

Elektronika a počítače

Radek Hnilica

Elektronika a počítače

Radek Hnilica

Working Vydání

Vydáno

Copyright © 2007, 2008, 2009, 2010, 2011 Radek Hnilica

Vše od základů elektroniky přes logické obvody až k procesorům.

Pracovní sešit na téma elektronika a počítače. Postihuje vše od základních věcí až po pokročilou digitální techniku. Tedy sem si píšete vše co do tohoto tématu spadá, a píšete to jak vám to přijde pod ruku a máte čas a potřebu si příslušnou informaci zaznamenat.

Tento dokument je k dispozici v několika různých formátech. Jako vícestránkový HTML dokument², postscriptový³ či PDF⁴ soubor formátovaný na velikost papíru A4. Pokud některý z těchto formátů nenaleznete, nebo bude neaktuální dejte mi vědět, připravím jej pro vás.

Aktuální verze knihy je vystavena na www.hnilica.cz⁵, www2.hnilica.cz⁶ a na penguin.cz/~radek⁷. Některé z těchto webů nemusí být dosažitelné.

Pokud budu mít možnost vystavit tento dokument i jinde, rád tak učiním.

Počet stran v Postscriptové⁸ a PDF⁹ verzi: 793 .

ěščřžýáíéúů

Příšerně žlutoučký kůň upěl d'ábelské ódy.

Hled', tot' přízračný kůň v mátožné póze šíleně upí.

Toto dílo smíte užívat dle podmínek licence CC BY-NC-SA¹.



Přehled revizí

Revize 3	2010-08-23	Revidoval: rh
Změna licence.		
Revize 2	2009-07-07	Revidoval: rh
Aktuální pracovní verze.		
Revize 1.19	2009/01/28 18:30:17	Revidoval: rfh
Aktuální pracovní verze.		

Věnování

FIXME:napsat věnování

Obsah

Předmluva.....	xvii
1. Úvod	1
I. Součástky, komponenty a materiály	1
2. Fyzikální veličiny	2
2.1. Základní fyzikální veličiny	2
2.2. Odvozené fyzikální veličiny	5
2.3. Zákony a vztahy mezi veličinami	5
3. Materiály a jejich vlastnosti	7
3.1. Vodiče	7
3.2. Polovodiče	8
4. Pasivní součástky.....	12
4.1. Odpor	12
4.2. Kondenzátor.....	12
4.3. Cívka.....	13
4.4. Termistory a teplotní čidla.....	13
5. Mechanické součástky.....	15
5.1. Tiltswitch	15
6. Zvláštní a neobvyklé součástky.....	16
6.1. Parametron.....	16
II. Integrované obvody.....	17
7. Stabilizátory napětí a proudu.....	19
7.1. LM317	19
7.2. LM7805	19
7.3. LE50A.....	19
7.4. LT3802.....	20
8. Konvertory napětí.....	21
8.1. Konvertory z nižšího na vyšší napětí (step-up).....	21
8.2. Konvertory z vyššího napětí na nižší (step-down).....	22
8.3. Konvertory napětí Step-Up / Step-Down.....	23
8.4. Sběr energie (<i>Energy Harvesting</i>)	24
9. Operační zesilovače.....	26
10. Zvláštní obvody.....	27
10.1. Funkční generátory	27
10.2. NE555, NE556.....	28
10.3. Rádiové přijímače či jejich části.....	29
10.4. Spínací tranzistorová pole.....	30
10.5. Řízení motorů	31
10.6. TAA661 (MAA661)	32
10.7. LM3909	32
11. Zesilovače.....	35
11.1. LM386 (<i>Low Voltage Audio Power Amplifier</i>)	35
11.2. TDA2822	36
III. Číslicová technika.....	38
12. Hradla	39
12.1. Přehled typů hradel	39
12.2. Konstrukce digitálních hradel.....	44
13. Kombinační obvody	56
13.1. Booleova Algebra	56

13.2. Sčítání	57
14. Sekvenční obvody	60
14.1. Klopný obvod RS.....	60
15. Dual Rail logika	62
16. Přehled některých logických IO	63
16.1. Řady/rodiny logických obvodů.....	68
16.2. Zesilovače, převodníky úrovní.....	69
16.3. Logické obvody / hradla	71
16.4. Dekodéry.....	77
16.5. Klopné obvody.....	79
16.6. Čítače	82
16.7. Posuvné registry.....	84
16.8. Záchytné registry	86
16.9.	87
16.10. Zvláštní, nebo zatím nezařazené obvody	87
17. Obvody pro MCU	89
17.1. Paměti	89
17.2. MMU	89
17.3. Obvody reálného času.....	90
17.4. LED Drivers.....	90
18. Zapojení s digitálními obvody.....	92
18.1. Oscilátory	92
18.2. MMU	92
18.3. Level Shifter	92
IV. Obvody a zapojení	96
19. Rezonanční obvody	97
19.1. LC obvod	97
20. Oscilátory	98
21. Tranzistorové klopné obvody	99
21.1. Astabilní klopný obvod, multivibrátor.....	99
21.2. Bistabilní klopný obvod / Dělička kmitočtu	99
21.3. RS klopný obvod	101
22. Zobrazovače	106
23. Velmi jednoduchá zapojení	107
23.1. Joule Thief	107
24. Zdroje	108
24.1. Stabilizátory napětí	108
V. Komunikace a kódování.....	109
25. Reprezentace čísel	110
25.1. Vyjádření celých čísel.....	110
26. Kódy	112
26.1. ASCII.....	112
26.2. GCR	113
27. IR komunikce	116
VI. Počítačová architektura	117
28. Poznámky	118
29. Návrh procesorů	119
29.1. 32 bitový procesor orientovaný na Forth	119
29.2. Procesor se sériovou ALU	122
29.3. SOC-8	123

29.4. SC-4	133
29.5. SOC-240	137
29.6. SOC-168	139
29.7. SOC-22	150
29.8. Různé úvahy na téma 12 bitových CPU	151
29.9. FJE5	159
29.10. SPEW	160
29.11. MPROZ	161
29.12. ZPU	162
29.13. Úvaha na jednoduchou konstrukci SC-2	163
29.14. Varianty SC3	165
29.15. Jednoduchý procesor orientovaný na forth	165
30. Návrh instrukční sady procesoru	170
30.1. Instrukční sady s malým počtem instrukcí a jednou či více adresami	170
31. Binární aritmetika	173
VII. Mikroprocesory a mikrořadiče	174
32. MuP21 a F21	175
33. Mikroprocesor Z80	176
33.1. Obvody rodiny Zilog Z80	177
33.2. Instrukční sada Z80	180
33.3. Programové vybavení pro Z80	180
33.4. Zapojení pro Z80	182
33.5. Konstrukce počítačů s Z80	188
34. RCA CDP1802 (1805)	192
34.1. Obvody rodiny CDP 1802	196
34.2. Instrukce CDP 1802 (1805)	199
35. Motorola 6809	209
35.1. Registry	209
35.2. Programovací nástroje a prostředí	209
35.3. Konstrukce založené na 6809	210
36. 6502	211
37. SC/MP	212
38. MIPS32	213
38.1. Registry	213
39. PIC	214
39.1. Programátory pro PIC	214
39.2. Nástroje a prostředí pro vývoj programů	214
39.3. Konstrukce s PICy	215
39.4. Řešení různých problémů	215
40. AVR	217
40.1. Integrované periferie a části procesoru	221
40.2. Popis vybraných procesorů	224
40.3. Programování AVR	240
40.4. Nástroje a prostředí pro vývoj programů	244
40.5. Instrukční sada	251
40.6. Konstrukce a zapojení s AVR	262
40.7. Knihovna kódů	269
40.8. Arduino	280
40.9. Processing	325
41. Propeller	335
42. PSoC	337

43. Projekty s mikrořadiči	338
43.1. Využití malých obvodů s nízkým počtem vývodů	338
43.2. Komunikace mezi řadiči pomocí LED	338
43.3. Generování videosignálu	339
43.4. Čtení analogových hodnot	340
43.5. RFID	340
43.6. Čítače	340
43.7. Rozšíření počtu I/O	340
43.8. Obsluha LED v maticovém zapojení	341
44. Virtuální procesory	343
44.1. Zork	343
VIII. Periferie a rozhraní	344
45. Sérió-paralelní registry	345
46. SD/MMC	346
46.1. Připojení MMC k AVR ATmega	346
46.2. Připojení k CDP 1802	346
47. LCD a LED displeje	348
47.1. Řadiče zobrazovacích jednotek	348
47.2. Displeje z mobilů a jiných zařízení	348
48. JTAG	357
49. USB	358
49.1. Fyzická vrstva	358
49.2. Čipy	358
49.3. Konstrukce	359
49.4. Převodníky USB \longleftrightarrow RS232	359
50. Prostorová a orientační čidla	364
50.1. Kompas	364
51. Sériová komunikace	365
51.1. RS232	365
51.2. RS422, TIA/EIA-422	368
51.3. RS485, EIA/TIA-485	368
51.4. MIDI	370
51.5. CAN	370
52. Síťová komunikace	373
52.1. Ethernet	373
52.2. Modbus	374
52.3. Profibus	375
53. Záznamová média	377
53.1.	377
54. Rozhraní pro videosignál	378
54.1. SCART	378
55. Zvukové generátory a syntezátory	380
55.1. Programovatelné zvukové generátory (PSG)	380
55.2. FM a OPN	380
56. Ostatní rozhraní a sběrnice	382
56.1. SPI	382
56.2. Ethernet	382
56.3. 1 Wire	382
56.4. TWI, I ² C	382
56.5. BlueTooth	382
56.6. DMX512	382

57. Připojený výkonných motorických prvků	384
57.1. Ovládání servomotorů	384
58. Analog Digital převodníky	385
IX. Konstrukce, většinou amatérské	386
59. Komponenty	387
59.1. Experimentální PCB	387
60. Doma dělané a zvláštní počítače či procesory	388
60.1. Kenbak-1	389
60.2. Mark-8	389
60.3. EDUC-8 Microcomputer	389
60.4. BMOW 1 (<i>Big Mess o' Wires I</i>)	390
60.5. Turing Machine	391
60.6. AppleCrate II	391
61. Konstrukce s CPU	393
61.1. Intel 8080	393
62. Repliky	394
62.1. PDP-2	394
X. Vybrané architektury	395
63. PDP-5, PDP-8, IC6100, HD-6120	396
63.1. Jednotlivé modely počítačů	396
63.2. Instrukční sada	400
63.3. Základní a rozšířené periferie	502
63.4. Varianty a rozšíření minipočítače	508
63.5. Programové vybavení	510
63.6. Datové formáty	516
63.7. Simulátory PDP-8	518
63.8. Technologie programování PDP-8	520
Literatura	523
64. VAX	525
64.1. Instrukční sada	525
65. IBM System 360	530
65.1. Formáty instrukcí	530
65.2. Instrukce	532
65.3. Nástroje pro tvorbu programů	540
XI. Robotika a mechatronika	541
66. Roboti	542
66.1. Humanoidní roboti	542
66.2. Vozítka	542
66.3. Létadla	542
67. Obráběcí a tvořící stroje	544
67.1. RepRap a jiné 3d tiskárny	544
67.2. Vrtací a frézovací stroje	544
67.3. Plotry	545
68. Díly a komponenty	547
68.1. Krokové motory a jejich řízení	547
68.2. Servomotory	548
68.3. Contraptor	550
68.4. Konstrukční materiály	551
69. Poznámky	553
70. Mechanika	554

70.1. Různé poznámky	554
XII. Programové vybavení a operační systémy	555
71. Softwarové nástroje	556
71.1. gEDA	556
71.2. TkGate	560
71.3. EMC2.....	561
71.4. SketchUp.....	561
71.5. Art Of Illusion.....	562
71.6. SIMH (simulátor starých počítačů)	562
72. Assemblery a překladače.....	564
73. Femto OS.....	565
74. UniFLEX.....	566
75. UCSD Pascal	567
XIII. Historie výpočetní techniky	568
76. Poznámky k historii.....	569
76.1. Minipočítače	570
77. Výrobci počítačů	572
77.1. Artronix.....	572
77.2. Control Data Corporation	572
77.3. Data General	574
77.4. Digico	575
77.5. Digital Equipment Corporation	611
77.6. Elliott Bros (London) Ltd.	639
77.7. English Electric Ltd.	642
77.8. Ferranti.....	643
77.9. GRI Computer Corporation.....	647
77.10. Honeywell.....	647
77.11. IBM (International Business Machines Corporation).....	647
77.12. LEO Computers Ltd.	655
77.13. Norsk Data	655
77.14. Nuclear Data, Inc.	657
77.15. Philco	686
77.16. Univac.....	686
78. Historie Sovětské výpočetní techniky	690
79. Historie výpočetní techniky v ČSSR a zemích východního bloku	691
80. Mechanické počítače a konstrukce.....	692
81. Réleové počítače.....	693
81.1. Z3 (Konrad Zuse).....	693
81.2. R500/7T Relay Computer.....	693
81.3. Elektrický mozek Simon.....	693
82. Elektronkové počítače a jiné stroje	694
82.1. Colosus	694
82.2. ABC	694
82.3. ENIAC	694
82.4. EDSAC	694
82.5. UNIVAC	695
82.6. EDVAC	695
82.7. LEO.....	696
82.8. IAS.....	696
82.9. MANIAC	696
82.10. SSEM (Small Scale Experimental Machine) [1948-06-21]	697

83. Počítače druhé generace	699
83.1. IBM 1401	699
83.2. ELLIOT 803	699
84. Doba elektronková	700
84.1. Whirlwind (MIT)	700
85. Doba transistorová	723
85.1. TX-0	723
85.2. CDC 160	727
85.3. CDC 160A	732
85.4. ARC-1 — The Average Response Computer	733
85.5. LINC [1962]	734
85.6. L-1	735
85.7. Control Data Corporation Model 1604	737
85.8. Eliot	737
85.9. Další počítače	737
86. Doba mikroprocesorová	739
86.1. I8008	739
86.2. RCA1802	739
86.3. I8080	740
86.4. Z80	740
86.5. MC6800	741
86.6. 6502	741
86.7. Altair	742
86.8. IMSAI	742
87. Nezařazené stroje	744
87.1. Philco 212	744
88. Historie na videu	745
89. Osobnosti	747
89.1. Howard Aiken	747
89.2. Sergei Alexeevich Lebedev (1902-1974)	747
89.3. David L. Jones (*19xx)	747
89.4. Grace Hopper (*1906, †1992)	748
89.5. Hans R. Camenzind (*1934)	748
89.6. Harlan Anderson	748
89.7. Jeri Ellsworth	748
89.8. Ken Olson	748
89.9. Saul Dinman	749
89.10. Seymour Cray	749
89.11. Steve Wozniak (*1950-08-11)	749
89.12. William Friedman	749
90. Staré časopisy	751
91. Operační systémy a softwarové vybavení	752
91.1. OS-9/6809	752
91.2. FLEX	752
92. Počítače ve vesmíru a v letadlech	753
92.1. 1750A	753
92.2. IBM AP-101	755
92.3. Magic I	755
92.4. TRADIC	756
93. Počítače ve zbraních (raketách, bombách, ...)	757
93.1. Minuteman-I Autonetics D-17 flight computer	757
93.2.	757

94. Úvod	758
94.1. Zdroje informací	758
94.2. Nezaříděné a nezpracované informace	758
95. Velmi staré počítače	760
95.1. Burroughs B-200	760
95.2. SAGE	760
95.3. Ostatní počítače	760
96. Minipočítače	761
96.1. Interdata	761
97. Blíže neidentifikované počítače	762
97.1. RCA 110	762
97.2. Central Air Data Computer	762
98. Všechno, náměty a co se jinam nevešlo	763
98.1. CoreWars	763
98.2. Zajímavé a velmi podivné projekty	763
98.2.1. Fusion reactor	763
98.2.2. Elektronový mikroskop	763
98.3. Různé nezařazené informace	764
98.4. Ham Radio	764
98.5. Nápady na experimenty	764
98.5.1. Laser	764
98.5.2. OLED	764
98.6. Výrobci součástek	764
98.6.1. National Semiconductor	764
A. Vybavení laboratoře/dílny	766
A.1. Nepájivá kontaktní pole (<i>Breadboard</i>)	766
A.2. Moduly pro nepájivá kontaktní pole a experimenty	767
A.3. Meřicí přístroje	767
A.4. Napájecí zdroje	767
A.4.1. Využití standardních počítačových zdrojů	767
B. Technologie a postupy	769
B.1. SMT / SMD	769
B.2. DPS (PCB)	769
C. Odkazy na weby a blogy	772
C.1. České stránky	772
C.2. Ostatní stránky	772
C.3. Obchody	772
D. Různá doporučení	774
D.1. Jeri Ellsworth	774
D.2. Dave Jones	774
D.2.1. Rigol DS1052E a další	774
E. Zajímavý hardware	??
E.1. Ben NanoNote	??
E.2. Malé počítače, samostatné desky	??

Seznam tabulek

2-1. Tabulka základních jednotek	2
2-2. Násobky a jejich značení	2
2-3. Jednotky délky	3
3-1. AWG	7
3-2.	10
6-1. PC-1	16
7.	17
10-1.	31
12-1. Funkční tabulka hradla YES	39
12-2. Funkční tabulka invertoru	40
12-3. Funkční tabulka AND	40
12-4. Funkční tabulka OR	41
12-5. Tabulka výstupu dvou vstupového hradla NAND	42
12-6. Tabulka výstupu tří vstupového hradla NAND	42
12-7. Funkční tabulka XOR	43
13-1. Pravdivostní tabulka některých binárních operací	57
13-2. Pravdivostní tabulka některých binárních operací	57
13-3. Funkční tabulka poloviční sčítačky	57
15-1. Logické hodnoty na dual rail	62
16-1. Použité zkratky	63
16-2. Tabulka obvodů 74 podle značení	63
16-3. Tabulka obvodů 4000 podle značení	67
16-4. Přehled řad/rodin logických obvodů	68
16-5. Vybrané parametry vybraných rodin	69
16-6. Pravdivostní tabulka klopného obvodu D v IO CD4013	81
16-7. Funkční tabulka 74165	85
17-1.	90
17-2. Dostupnost TLC5940	91
25-1.	110
26-1. Význam řídicích kódů ASCII	112
26-2. Commodore GCR Codes	113
26-3. GCR: (0,2) RLL	113
26-4. 4B5B Encoding Table	114
29-1. Registry procesoru	119
29-2. Struktura instrukčního slova	119
29-3.	120
29-4. Základní instrukce pro umístění do tabulky atomických instrukcí	122
29-5. Registry procesoru SOC-8	123
29-6. Bity stavového slova	124
29-7.	125
29-8. Adresní módy instrukcí SOC-8	127
29-9. Adresní módy instrukcí SOC-8	128
29-10. Adresní módy	130
29-11. Některé instrukce SOC-8	132
29-12.	132
29-13. Plánované instrukce	138
29-14. Návrh prostoru instrukčních kódů:	138
29-15. Registry procesoru SOC-168	139
29-16. Tabulka Instrukcí SOC-168	143

29-17. Význam bitů pole adresního módu instrukce SOC-168	144
29-18. Registry SOC-22.....	150
29-19. Adresní módy	152
29-20. Instrukce	152
29-21. Tabulka instrukcí	154
29-22. Tabulka instrukcí	156
29-23.	157
29-24.	158
29-25. Registry procesoru FJE5.....	159
29-26. Instrukce procesoru EFJ5	159
29-27. Registry procesoru SPEW	160
29-28. Registry procesoru SPEW	161
29-29.	161
29-30.	164
29-31. Tabulka instrukcí nanoForth	167
29-32. Tabulka instrukcí Nagro VM	167
30-1. Ukázka návrhu instrukční sady.....	171
33-1. Základní instrukce	180
34-1. Přehled obvodů rodiny CDP 1802.....	197
34-2. Operační kódy instrukcí CDP1802 (1805) řazené podle operačního kódu	199
38-1. Konvence pojmenování a použití registrů	213
40-1.	218
40-2. Přehled vybraných obvodů řady AVR ATtiny	218
40-3. Přehled vybraných obvodů řady AVR ATmega.....	219
40-4. Obvody s USB rozhraním.....	221
40-5. SREG — The AVR Status Register.....	222
40-6. Timer/Counter Control Register A - TCCR0A	223
40-7. Dostupnost ATtiny 24,44,84.....	225
40-8. Dostupnost ATtiny 25,45,85.....	225
40-9. Dostupnost ATmega8	227
40-10. Vektory přerušení ATmega162	228
40-11. Registry.....	228
40-12. Dostupnost ATmega162	229
40-13. Registry čítače/časovače 0 procesoru ATmega162.....	229
40-14.	230
40-15.	231
40-16. Popis funkce vývodů portu B	231
40-17. Dostupnost ATmega8	232
40-18. Dostupnost ATmega 16/32/64	232
40-19. Některé parametry mikropočítačů ATmega48 až 328	233
40-20. I/O registry.....	233
40-21. Dostupnost ATmega 48/88/169/328 P	234
40-22. Compare Output Mode, non-PWM Mode.....	236
40-23. Compare Output Mode, Fast PWM Mode.....	236
40-24. Compare Output Mode, Phase Correct PWM Mode.....	236
40-25. Waveform Generation Mode Bit Description.....	236
40-26. Dostupnost ATmega 164/324/644/1284 P/PA	240
40-27. Přesné bitové šířky jednotlivých typů.....	248
40-28. Vektory přerušení.....	249
40-29. Operační kódy instrukcí AVR řazené podle operačního kódu.....	251
40-30.	288
40-31.	289

40-1.	293
41-1. Cog RAM Special Purpose Registers	335
42-1. Dostupnost součástek:	337
47-1. Informace o monochromatický LCD displejích	349
47-2. Parametry displejů	349
47-3. Zapojení vývodů Nokia3310 na připájeném plochem vodiči	353
47-4. Připojení displeje k MCU AVR	353
47-5.	354
49-1. USB kabel/konektor	358
49-2. Použitelné kabely	359
49-3. Zapojení konektoru nokia	360
49-4. Zapojení kabelu CA-42	361
49-5. Zapojení kabelu DKU-5	361
49-6. Zapojení kabelu	362
51-1. Barvy vodičů	368
51-2.	369
52-1.	374
52-2. Dostupnost ENC28J60	374
52-3.	374
52-4. Kódy funkcí	375
56-1. Startovací kódy	383
63-1. Přehled jednotlivých modelů PDP-8	396
63-2.	397
63-3. Soubor základních instrukcí PDP-5	398
63-4. Doba provádění instrukcí	399
63-5. Základní instrukce PDP-8	401
63-6. Četnosti užití instrukcí v programu FOCAL-8	402
63-7. Standardní adresy některých I/O zařízení PDP-8	402
63-8. Kombinace bitů mikroinstrukcí pro rotace ve skupině 1	404
63-9. Group 1 Microinstructions od PDP-8	405
63-10. Group 2 Microinstructions od PDP-8	406
63-11. Group 3 Microinstructions od PDP-8	407
63-12. Operační kódy instrukcí PDP-8 řazené podle operačního kódu	409
63-1. DCLR příkazy	425
63-14. Příkazy pro práci s konzolou	502
63-15. Instrukce DC02-F	504
63-16. FFP-12 Active Parameter Table Format	505
63-17. IOT instrukce pro FPP-12	506
63-18. Instruction Set	506
63-19. Instrukce pro práci s RK8	507
63-20. KT8-A Compatible Instructions	508
63-21. KT8-A Expanded instructions	509
63-22. Důležití/systémoví uživatelé TSS/8	514
63-23. Simulovaný hardware	519
63-24. Nastavování procesoru	519
64-1.	525
64-2. Instrukce řazené podle operačního kódu	525
65-1. Délky a formáty instrukcí architektury System 360	530
65-2. Obsazení paměti IBM System 360	531
65-3. Seznam instrukcí System 360 podle operačních kódů	532
68-1. NEMA	547
68-2. Připojení modelářského serva	549

68-3. Turnigy TG9	549
68-4. HEXTRONIK HTX 900.....	550
77-1. CDC Display code characters (64-character character set version)	572
77-2. Přehled základních instrukcí.....	576
77-3. Operační kódy instrukcí Micro 16V řazené podle operačního kódu.....	577
77-4. Stručný přehled modelů PDP firmy Digital.....	611
77-5. Přehled instrukcí PDP-1	612
77-6. Přehled instrukcí PDP-4	613
77-7. Přehled instrukcí PDP-7	614
77-8. Registry PDP-5, PDP-8	615
77-9. Základní instrukce PDP-5, PDP-8.....	616
77-10. Modely PDP-8	616
77-11. Modely kompatibilní od jiných výrobců	617
77-12. Operační kódy instrukcí PDP-11	621
77-13. Počítače Elliot a data spuštění	639
77-14.	642
77-15. Registry.....	642
77-16.	643
77-17. Ferranti Pagasus Memory Map.....	644
77-18. Ferranti Atlas Index Registers	646
77-19. Počítače IBM	647
77-20. Sumary of AN/FSQ-8 instructions	650
77-21. Instrukce IBM 701.....	651
77-22. Kombinace bitů v paměťové buňce IBM 1620.....	652
77-23. IBM 1400 series.....	653
77-24.	654
77-25. Single-Word instrukce	658
77-26.	659
77-27. Single-Word Memory Reference Instructions	660
77-28. Modifikace přístupu k paměti	660
77-29. Two-Word Memory Referenc Instructions	661
77-30.	664
77-31.	666
77-32. Operační kódy instrukcí ND812 řazené podle operačního kódu.....	666
77-33. Registry UNIVAC 1218.....	687
77-34. Adresy paměti UNIVAC 1218.....	687
82-1. SSEM Instruction Set	697
84-1. Instrukce počítače Whirlwind.....	700
85-1. Registry TX-0	724
85-2. TX0 Combined Operate Class Commands.....	726
85-3. Registry modifikovaného TX-0	726
85-4. Základní instrukce modifikovaného TX-0.....	727
85-5. Tabulka instrukcí CDC-160.....	728
85-6. Adresní módy CDC-160.....	728
85-7.	729
85-8. Instruction table for CDC 160	730
85-9.	732
85-10. Základní instrukce (Full Address) LINC.....	734
85-11. Základní instrukce LINC	735
85-12.	735
85-13.	737
87-1. Instrukce	744

B-1. Třídy PCB	769
B-2. Označení materiálů pro PCB	770
B-3. Rozměry některých eurokaret.....	770

Předmluva

FIXME:napsat nebo nechat napsat předmluvu.

Kapitola 1. Úvod

Odkazy:

- Lessons In Electric Circuits¹

Původně jsem tento dokument začal psát, abych si utřídil informace o digitálních obvodech. A těch tu opravdu příliš informací není. Později jsem pak začal sbírat informace o historické výpočetní technice a něco málo jsem se věnoval jednočipovým mikropočítačům, zejména AVR firmy Atmel.

Obrázky vytvářím několika programy. Obrázky instrukcí jsou vytvářeny pomocí programu **gpic** jenž přináleží ka programu **groff**. Schémata jsem pak jak šel čas vytvářel v programech TkGate, Electric a naposled a těch je nejvíce, gEDA.

Poznámky

1. <http://www.ibiblio.org/obp/electricCircuits/index.htm>

I. Součástky, komponenty a materiály

Kapitola 2. Fyzikální veličiny

*

2.1. Základní fyzikální veličiny

*

Odkazy:

- Desítková soustava¹ na české Wikipedii
- Předpona soustavy SI²
-
-

Při konstrukci elektronických zařízení pracujeme s reálnými materiály. S reálnou hmotou, která má rozměry, hmotnost a další vlastnosti. Abychom mohli vyjádřit číselně vlastnosti hmoty, musíme mít měřítko, tedy soustavu jednotek, k vyjádření.

Tabulka 2-1. Tabulka základních jednotek

veličina	označení	jednotka	označení	poznámka
délka		metr	m	
hmotnost		kilogram	kg	
čas	t	sekunda	s	
náboj		Coulomb	C	
napětí	U	Volt	V	
proud				
odpor	R	Ohm	Ω	
kapacita	C	Farad	F	
indukčnost	L	Henry		

Tabulka 2-2. Násobky a jejich značení

název	značení	číslo	poznámka
exa	E	10^{18}	
peta	P	10^{15}	
tera	T	10^{12}	
giga	G	10^9	

název	značka	hodnota	poznámka
mega	M	1 000 000	
kilo	k	1 000	
hekto	h	100	
deka	da	10	
deci	d	0,1	
centi	c	0,01	
mili	m	0,001	
mikro	μ	0,000 001	
nano	n	10^{-9}	
piko	p	10^{-12}	
femto	f	10^{-15}	
atto	a	10^{-18}	

2.1.1. Délka, rozměr

Délka je vzdálenost mezi dvěma body v prostoru.

Může to být vzdálenost jednoho konce tyče od druhého konce tyče. Vzdálenost jednoho rohu krabice od druhého rohu krabice, atd.

Tabulka 2-3. Jednotky délky

popis	zkratka		
metr	m		
.	mil	1000 mil = 1 in	
palec <i>inch</i>	in	1 in = 25.4 mm	
stopa <i>foot</i>	ft.		
míle .			

Základní délkovou jednotkou podle soustavy SI je metr. V době svého vzniku byl definován jako jedna desetimilióntina zemského kvadrantu. Protože tato definice přestala časem vyhovovat, byla nahrazena jinou, přesnější. Podle té je jeden metr definován jako 1 650 763,73 násobek vlnové délky záření ve vakuu které přísluší přechodu mezi energetickými hladinami $2p_{10}$ a $5d_5$ atomu kryptonu 86. Poslední definice je z roku 1983 a v ní je metr definován jako vzdálenost, kterou urazí světlo ve vakuu za dobu $1/299\,792\,458$ sekundy.

Ačkoliv je metry v soustavě SI standardizován již více než 200 let, přesto se v některých oborech používají odlišné, tradiční jednotky. Pro elektroniku je důležitá jednotka palec (inch) a jeho zlomek mil. Palec je definován jako 25,4 milimetru přesně a 1 mil je jedna tisícina palce. Od těchto měr jsou tradičně odvozeny například rotněry součástek a tím pádem se používají při konstrukci DPS.

2.1.2. Náboj

*

Odkazy:

- Electric charge³ na Wikipedii
- Coulomb⁴ na Wikipedii
-
-

Elektrický náboj je měřen v jednotkách coulomb

- $1 \text{ Coulomb} = 6,24151 \times 10^{18} \times p^+$; kde p^+ je elektrický náboj protonu. Můžeme také použít elektrický náboj elektronu e^- , protože $p^+ = -e^-$

2.1.3. Napětí

* *Attributy: id="voltage"*

Odkazy:

- Voltage⁵ na Wikipedii
- Electric potential⁶ na Wikipedii
-
-

U

2.1.4. Proud

*

Odkazy:

- Elektrický proud⁷ na české Wikipedii
- Electric current⁸ na Wikipedii
-

I

2.1.5. Odpor

Odkazy:

- Odpor (rezistor) jako součástka
- Electrical resistance⁹ na Wikipedii
-
-

Odpor jako veličina je schopnost materiálu klást odpor průchodu elektrického proudu. Označuje se písmenem R. Jednotka odporu je Ω .

V elektrickém obvodu je vztah mezi odporem, napětím a proudem dán ohmovým zákonem.

2.1.6. Kapacita

* *Attributy: id="Capacitance"*

Odkazy:

- Capacitance¹⁰ na Wikipedii
- 4.2
-

Rovnice 2-1. Rovnice

$$C = \frac{Q}{V}$$

Energie náboje q.

$$dW = \frac{q}{C} dq$$

2.1.7. Indukčnost

*

Odkazy:

- Inductance¹¹ na Wikipedii
- 4.3
-

Rovnice 2-2. Napětí na cívce

$$v = \frac{d\phi}{dt} = L \frac{di}{dt}$$

Rovnice 2-3. Proud protékající cívkou

$$i = \frac{1}{L} \int u dt$$

2.2. Odvozené fyzikální veličiny

* *Rozmyslet co dát do této sekce.*

2.3. Zákony a vztahy mezi veličinami

*

2.3.1. Ohmův zákon

* *id="ohms-law"*

Odkazy:

- Ohmův zákon¹² na české Wikipedii
- Ohm's law¹³ na Wikipedii
-

$$I = V / R$$

$$V = IR$$

$$R = V / I$$

2.3.2.

*

Poznámky

1. http://cs.wikipedia.org/wiki/Desítková_soustava
2. http://cs.wikipedia.org/wiki/Předpona_soustavy_SI
3. http://en.wikipedia.org/wiki/Electric_charge
4. <http://en.wikipedia.org/wiki/Coulomb>
5. <http://en.wikipedia.org/wiki/Voltage>
6. http://en.wikipedia.org/wiki/Electric_potential
7. http://cs.wikipedia.org/wiki/Elektrický_proud
8. http://en.wikipedia.org/wiki/Electric_current
9. http://en.wikipedia.org/wiki/Electrical_resistance
10. <http://en.wikipedia.org/wiki/Capacitance>
11. <http://en.wikipedia.org/wiki/Inductance>
12. http://cs.wikipedia.org/wiki/Ohmův_zákon
13. http://en.wikipedia.org/wiki/Ohm's_law

Kapitola 3. Materiály a jejich vlastnosti

*

V konstrukcích používáme řadu různých materiálů s odlišnými fyzikálními vlastnostmi.

Z hlediska vodivosti elektrického proudu rozlišujeme tři základní kategorie látek, jsou to:

- vodiče
- nevodiče - izolanty
- polovodiče

* Rozepsat se blíže.

3.1. Vodiče

*

Odkazy:

- Wire Gauge and Current Limits¹
- American wire gauge² na Wikipedii
- Standard wire gauge³ — BS 3737:1964, Imperial Wire Gauge or British Standard Gauge
-
-

Vodič je materiál, jenž snadno vede elektrický proud. Jako vodiče se používají kovy, většinou železo, hliník a měď. V elektronice je to téměř výhradně měď.

AWG je American Wire Gauge. Jedná se o standard popisující elektrické kabely.

$$D(\text{AWG}) = 0.005 * 92^{((36-\text{AWG})/39)} \text{ inch}$$

Tabulka 3-1. AWG

gauge	průměr		odpor				
	palce	mm	Ω / 1000 ft.	Ω / km			
OOOO	0.46	11.684	0.049	0.16072	380	302	125 Hz
OOO	0.4096	10.40384	0.0618	0.202704	328	239	160 Hz
OO	0.3648	9.26592	0.0779	0.255512	283	190	200 Hz
0	0.3249	8.25246	0.0983	0.322424	245	150	250 Hz
1	0.2893	7.34822	0.1239	0.406392	211	119	325 Hz
16	0.0508	1.29032	4.016	13.17248	22	3.7	11 kHz
17	0.0453	1.15062	5.064	16.60992	19	2.9	13 kHz
18	0.0403	1.02362	6.385	20.9428	16	2.3	17 kHz
19	0.0359	0.91186	8.051	26.40728	14	1.8	21 kHz
20	0.032	0.8128	10.15	33.292	11	1.5	27 kHz
21	0.0285	0.7239	12.8	41.984	9	1.2	33 kHz

gauge	průměr		odpor				
	palce	mm	Ω / 1000 ft.	Ω / km			
22	0.0254	0.64516	16.14	52.9392	7	0.92	42 kHz
23	0.0226	0.57404	20.36	66.7808	4.7	0.729	53 kHz
24	0.0201	0.51054	25.67	84.1976	3.5	0.577	68 kHz
25	0.0179	0.45466	32.37	106.1736	2.7	0.457	85 kHz
26	0.0159	0.40386	40.81	133.8568	2.2	0.361	107 kHz
27	0.0142	0.36068	51.47	168.8216	1.7	0.288	130 kHz
28	0.0126	0.32004	64.9	212.872	1.4	0.226	170 kHz
29	0.0113	0.28702	81.83	268.4024	1.2	0.182	210 kHz
30	0.01	0.254	103.2	338.496	0.86	0.142	270 kHz

3.2. Polovodiče

*

3.2.1. PN přechod

*

3.2.2. Tranzistor

* *Attributy:* id="transistor"

* Přemístit do kapitoly polovodiče

Odkazy:

- Tranzistor⁴ na české Wikipedii
- Transistors Tutorial⁵ Tony van Roon
- Tranzistor a tranzistorový jev⁶
- EARLY TRANSISTOR HISTORY AT GE⁷ Hugh R. Lowry
- Instruments of Amplification, Gallery 2⁸ H. P. Friedrichs (AC7ZL) Homepage —
-
-
-

Tranzistor je polovodičová nelineární součástka se třemi vývody. Nejčastěji se pod pojmem tranzistor rozumí bipolární tranzistor.

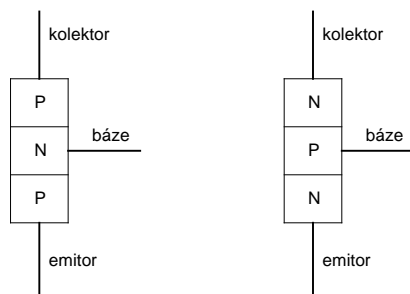
* *Bipolar Junction Transistor*

Tranzistorový jev objevili Dr. John Bardeen, Dr. Walter Brattain a Dr. William Shockley. První tranzistor zkonstruovali v prosinci 1947, v Bellových laboratořích v Murray Hill, NJ. Za objev tranzistorového jevu jim byla v roce 1956 udělena Nobelova cena za Fyziku. Objev tranzistoru způsobil převrat v aplikované elektrotechnice a vědeckotechnickou revoluci. Odstartoval miniarizaci elektronických součástek která vyvrcholila v konstrukci dnešních počítačů.

Bipolární tranzistor sestává ze dvou PN přechodů. Správné vytvoření těchto přechodů bylo klíčovým problémem v konstrukci tranzistoru. Pořadí P a N vrstev v tranzistoru umožňuje konstrukci dvou typů a to PNP a

NPN. Jednotlivé vrstvy jsou připojeny k vývodům označeným emitor (E), báze (B) a kolektor (C). Principem činnosti tranzistoru je jev, kdy proud tekoucí do báze mění "odpor" mezi emitorem a kolektorem a tím ovlivňuje velikost proudu tekoucího těmito vývody.

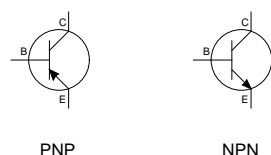
Obrázek 3-1. Bipolární tranzistor



* 320×240; EPS scale=50

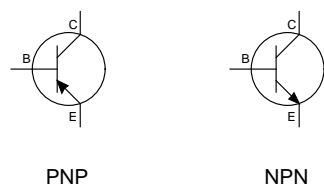
Schematická značka tranzistoru ukazuje všechny tři vývody. Emitor je znázorněn šipkou a směr šipky ukazuje směr proudu.

Obrázek 3-2. Schematická značka pro bipolární tranzistor



* 320×240; EPS scale=50

Obrázek 3-3. Schematická značka pro bipolární tranzistor (scale="60")



3.2.2.1. Základní zapojení tranzistoru

*

Známe tři základní zapojení tranzistoru, lišící se tím, který vývod je připojen přímo ke napájecímu zdroji. A to ať již zemnicímu vodiči nebo napájecímu vodiči. Tato zapojení se nazývají

- zapojení se společným emitorem
- zapojení se společným kolektorem
- zapojení se společnou bází

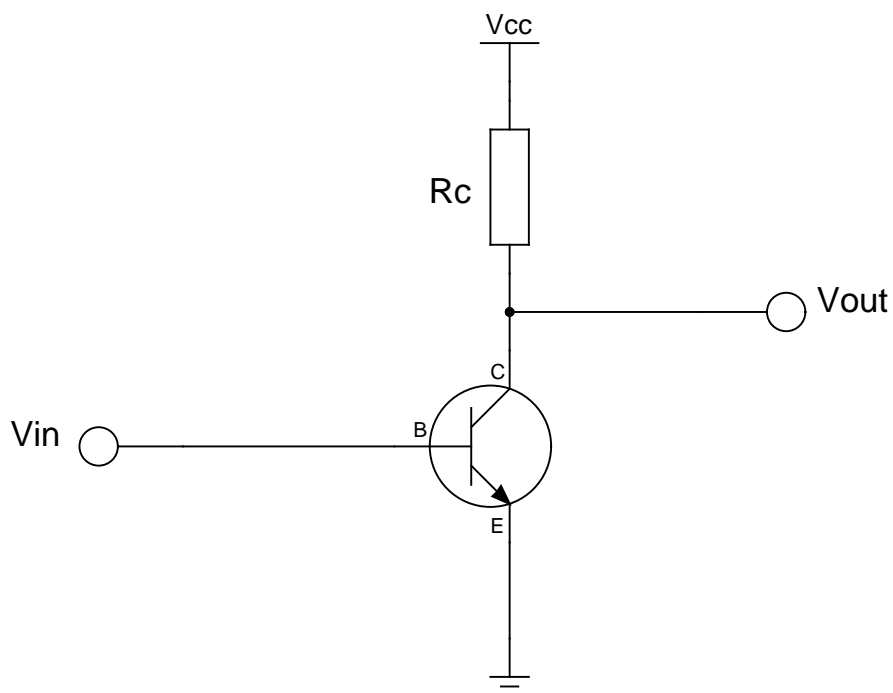
3.2.2.1.1. Zapojení se společným emitorem

*

Odkazy:

- Common emitter⁹
-

Obrázek 3-4.



* gEDA image schematic Transistor_Common_Emitter.sch: 320×240 EPS, PNG

3.2.2.2. Přehled známých tranzistorů

*

Tabulka 3-2.

typ	V_{CE0} [V]	I_c [mA]	P_{tot} [mW]	f_T [MHz]	
2N3904	40	200	625	100	General Purpose Transistor (NPN), complementary to 2N3906
2N3906	40	200	625	250	Small Signal transistor (PNP), komplementární k 2N3904
2N4401	40	600	625	250	Small Signal Transistor (NPN)
2N4036					
2N4037					PNP

Poznámky

1. http://www.powerstream.com/Wire_Size.htm
2. http://en.wikipedia.org/wiki/American_wire_gauge
3. http://en.wikipedia.org/wiki/Standard_wire_gauge
4. <http://cs.wikipedia.org/wiki/Tranzistor>
5. <http://www.sentex.ca/~mec1995/tutorial/xtor/xtor.html>
6. http://www.z-moravec.net/ext_el/tranz/tranz1.php
7. http://semiconductormuseum.com/Transistors/GE/OralHistories/Lowry/Lowry_Index.htm
8. <http://www.hpfriedrichs.com/bks-ioa-gallery2.htm>
9. http://en.wikipedia.org/wiki/Common_emitter

Kapitola 4. Pasivní součástky

*

Odkazy:

- [Electronic Components](#)¹
- [Electronic component](#)²
- [Basic Passive Components](#)³
-
-

FIXME: Pasivní součástky jsou takové, které nepotřebují napájení ke své činnosti.

Pasivní součástky nemají zesílení ani směrovost. V analýze.

Pasivní součástky jsou například odpory, kondenzátory a cívky.

4.1. Odpor

* *Attributy: id="component.resistor"*

Odkazy:

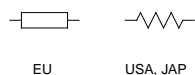
- Odpor jako veličina
- Odpor⁴ a vodivost⁵ na české Wikipedii
- Elektrotechnická součástka rezistor⁶ na české Wikipedii
- Resistor⁷ na Wikipedii
- Praktická elektronika/Lineární součástky⁸

Odpor je nejjednodušší součástka. Má dva vývody a jeho hlavní vlastnost je odpor který klade průtoku elektrického proudu. Odpor se značí znakem R, měří v jednotkách Ohm (Ω). Někdy se v souvislosti s vedením elektrického proudu mluví o vodivosti jenž se značí G a měří v jednotkách Siemens. Odpor je v přímém vztahu s vodivostí určeném vzorcem:

$$\text{odpor} = 1 / G$$

Schématická značka je jednoduchá, na následující obrázku je způsob kterým kreslí odpor v Evropě a způsob který je používán v Americe a Japonsku.

Obrázek 4-1. Schématická značka pro resistor



4.2. Kondenzátor

* *Attributy: id="Capacitor"*

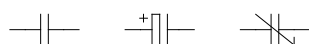
Odkazy:

- [Capacitance measurement](#)⁹

- Kapacita
- 2.1.6
- Praktická elektronika/Lineární součástky/Kondenzátor¹⁰ na Wikiknihách
-

Kondenzátor je součástka jejíž hlavní veličinou je 2.1.6. Existuje řada konstrukcí kondenzátoru. Po kapacitě je nejdůležitější informací polarita. Běžné kondenzátory nejsou polarizovány a nezáleží na tom jaké napětí (co do polarity) na něj přivedeme. Kondenzátory s větší kapacitou bývají vyráběny jako elektrolytické, ty mají stanovenou polaritu a nesmíme na ně přivést záporné napětí. Elektrolytické kondenzátory proto mají označenu polaritu na schématu i na samotné součástce.

Obrázek 4-2. Schematická značka pro kondenzátor



4.3. Cívka

*

Odkazy:

- 2.1.7
- Inductor¹¹ na Wikipedii
- Ferrite bead¹² na Wikipedii
- Electronics/Inductors¹³ na Wikibooks
-

4.4. Termistory a teplotná čidla

*

Odkazy:

- LM35/LM45/TMP35¹⁴ Precision Centigrade Temperature Sensor (10mv/°C, -55°C až 150°C)
- LM34/TMP34¹⁵ (Fahrenheit output)
- LM50/TMP36¹⁶
- MCP9700/MCP9701¹⁷
-
-

4.4.1. LM35 (*Precision Centigrade Temperature Sensor*)

*

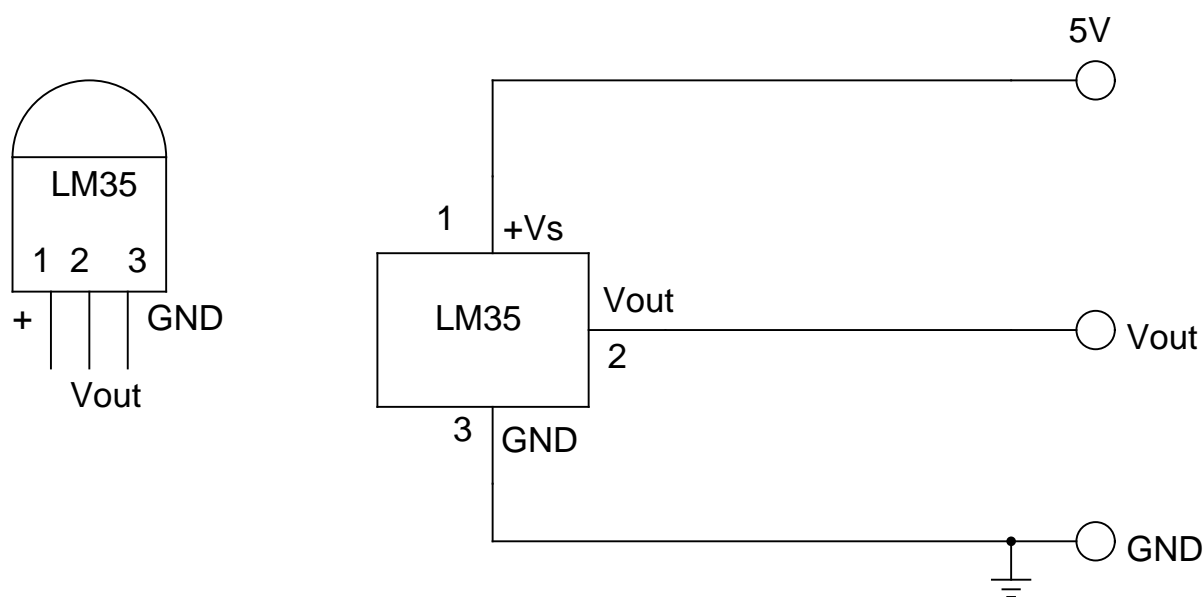
Odkazy:

- Teploměr k multimetru¹⁸
-

Zapojení s LM35 a podobnými čidly.

Čidlo teploty LM35 je obvod se třemi vývody pro měření teploty. Vyrábí jej National Semiconductor

Obrázek 4-3. Zapojení LM35



* Image size 640*480 PNG, EPS.

Poznámky

1. http://www.st-andrews.ac.uk/~www_pa/Scots_Guide/info/comp/comp.htm
2. http://en.wikipedia.org/wiki/Electronic_component
3. <http://sound.westhost.com/beginners.htm>
4. http://cs.wikipedia.org/wiki/Elektrick%C3%BD_odpor
5. http://cs.wikipedia.org/wiki/Elektrick%C3%A1_vodivost
6. <http://cs.wikipedia.org/wiki/Rezistor>
7. <http://en.wikipedia.org/wiki/Resistor>
8. http://cs.wikibooks.org/wiki/Praktick%C3%A1_elektronika/Line%C3%A1rn%C3%BD_sou%C4%85%C3%A1stky#Rezistor
9. <http://www.answers.com/topic/capacitance-measurement>
10. http://cs.wikibooks.org/wiki/Praktick%C3%A1_elektronika/Line%C3%A1rn%C3%BD_sou%C4%85%C3%A1stky#Kondenz%C3%A1tor
11. <http://en.wikipedia.org/wiki/Inductor>
12. http://en.wikipedia.org/wiki/Ferrite_bead
13. <http://en.wikibooks.org/wiki/Electronics/Inductors>
14. <http://www.national.com/mpf/LM/LM35.html#Overview>
- 15.
- 16.
- 17.
18. <http://pokusy.chytrak.cz/schemata/LM35.htm>

Kapitola 5. Mechanické součástky

*

5.1. Tiltswitch

*

Odkazy:

- Build your own tilt switch¹ na Instructables
-

Poznámky

1. <http://www.instructables.com/id/Build-your-own-tilt-switch/>

Kapitola 6. Zvláštní a neobvyklé součástky

6.1. Parametron

Odkazy:

- Historical Computers in Japan¹
- The first computers: history and architectures² strana 429

Parametron byl vynalezen Dr. Eiichi Goto v roce 1954. Byl levný a spolehlivý a na jeho základě vyvinula laboratoř profesora Hidetosi Takahasi počítač PC-1 (Parametron Computer 1).

Tabulka 6-1. PC-1

zahájení prací	září 1957
dokončen	březen 1958
demontován	květen 1964
počet parametronů	4200
paměť	magnetická jádra, 512 slov
čas sčítání/odčítání	0.4 ms
čas násobení	2.6ms (short), 4.4ms (long)
čas dělení	16.1ms (short/long)
frekvence hodin	10kHz
vstup	fotoelektrická čtečka papírové pásky
výstup	Tele-typewriter

Citace

This was a new logic element invented by Goto Eiichi in 1954. At the time, Goto was a graduate student in the laboratory of Takahasi Hidetosi at the Department of Physics, the University of Tokyo. The parametron was a logic circuit element which utilized the parameter excitation phenomenon. When the L or C in LC resonance circuit is varied by a frequency almost twice the resonance frequency f using parameter excitation. Goto noticed that this oscillation had two phases, 0 or π , and that binary numbers could be represented by that phase. He adopted "parametron" as the name of this new logic element for electronic computers, and made his first research report in July 1954 before the Electronic Computer Research Group of the Institute of Telecommunications Engineers. An electronic computer called the PC-1 was prototyped using parametrons in the Takahasi's laboratory.

* *Parametron*³, *Historical Computers in Japan, Early Computers*.

Goto invented a quantum flux parametron (QFP), an ultra high speed logical element, based on the Josephson junction phenomena in the cryogenic state. He promoted this research as a leader of JST's Goto Quantum Magneto-Flux Logic Project.

Poznámky

1. http://www.ipsj.or.jp/katsudou/museum/computer/0080_e.html
2. <http://books.google.com/books?id=nDWPW9uwZPAC>
3. <http://museum.ipsj.or.jp/en/computer/dawn/0007.html>

II. Integrované obvody

Odkazy:

- Wikipedia_{EN}: Integrated circuit⁵⁵
-

Integrovaný obvod, čip, mikročip, monolitický integrovaný obvod, jsou názvy pro konstrukční jednotky v jednom „malém“ pouzdru. Konstruují jednotky obsahující více aktivních a pasivních prvků. První integrované obvody obsahovaly doslova jen několik tranzistorů na jedné křemíkové destičce, zapojených do nějakého obvodu. Dnešní integrované obvody mohou obsahovat takových tranzistorů stovky milionů.

* Ověřit dnešní počty tranzistorů v CPU a RAM!

Odkazy:

- Kniha Designing Analog Chips⁵⁶ by Hans Camenzind
- The VCO Chip Cookbook⁵⁷ by Thomas Henry
-
-

Nedlouho po objevu tranzistoru začal výzkum polovodičových materiálů vedený záměrem efektivně vyrábět tranzistor. Ale tady se vývoj nezastavil. Pokus umístit několik tranzistorů na stejný kousek křemíku vedl k objevu/konstrukci integrovaného obvodu.

Počátky integrovaných obvodů jsou datovány do roku 1949, kdy německý inženýr Werner Jacobi (Siemens AG) podal žádost o patent⁵⁸ na polovodičový integrovaný zesilující obvod obsahující pět tranzistorů na společném substrátu zapojených jako dvoustupňový zesilovač. Jako typické komerční užití Jacobi popisuje naslouchadlo pro nedoslýchavé. Není známo žádné komerční užití Jacobiho patentu.

Předchůdci skutečných integrovaných obvodů byly obvody složené z několika miniaturních samostatných součástek na malé keramické destičce. Tato myšlenka vedla k programu Mikromodulů. Ale než se stačily mikromoduly vážněji prosadit, došlo k průlomu a konstrukci skutečného integrovaného obvodu na jednom kousku polovodiče.

Firma IBM používala SLT⁵⁹ obvody v konstrukcích počítačů System/360.

* Ověřit a doplnit odkaz na SLC moduly od IBM použité v konstrukcích jejich počítačů. Tyto obvody, nazývané Solid Logic Technology⁶⁰ byly zavedeny do výroby v roce 1964.

1. Značení integrovaných obvodů

*

V označení/názvech integrovaných obvodů je jistý řád. Skládá se z označení výrobce následované kódem vlastního obvodu. Protože názvy nejsou určovány žádnou centrální autoritou, vyvinul se systém značení obvodů tak jak se vyvinul. Pro usnadnění v orientaci uvádím následující tabulku prefixů výrobců.

Tabulka 7.

Prefix	Výrobce
AD	Analog Devices
Am	Advanced Micro Devices
Bx	Sony
CA	RCA (nyní Harris)
CD	RCA (nyní Harris)
Cx	Sony

Prefix	Výrobce
DM	National Semiconductor
F	Fairchild (nyní National Semiconductor)
FSS	Ferranti
HA	Harris
HA	Hitachi
HD	Hitachi
HG	Hitachi
HI	Harris
IR	Sharp
KA	Samsung
LF	National Semiconductor
LM	National Semiconductor
LT	Linear Technology
M	Mitsubishi
MB	Fujitsu
MC	Motorola
MH	Tesla
MM	Motorola
NE	Signetics
PM	Precision Monolithics
SN	Texas Instruments
T	Toshiba
TL	Texas Instruments
TMS	Texas Instruments
XR	Exar
μ A	FSC
μ PB	NEC

* Tabulka je převzata z *The Forrest Mims engineer's notebook*⁶¹ doplněných podle mých znalostí.

Tak jak šla doba, a někteří výrobci byly pohlceni jinými, zachovávali noví majitelé stejné označení integrovaného obvodu.

Poznámky

55. http://en.wikipedia.org/wiki/Integrated_circuit

56. <http://www.designinganalogchips.com/>

57. <http://matrixsynth.blogspot.com/2007/06/vco-chip-cookbook.html>

58. <http://v3.espacenet.com/publicationDetails/biblio?CC=DE&NR=833366C&KC=C&FT=D&date=19520630>

59. http://www-03.ibm.com/ibm/history/exhibits/vintage/vintage_4506VV3081.html

60. http://en.wikipedia.org/wiki/IBM_Solid_Logic_Technology

61. http://books.google.cz/books?id=a_B4dCFFL2oC

Kapitola 7. Stabilizátory napětí a proudu

IO pro napájecí zdroje

K dispozici je velká řada integrovaných obvodů.

Součástky k prozkoumání

- Linear technology LT3085 — Low drop, 500mA, napětí nastavitelné jedním odporem

7.1. LM317

Odkazy:

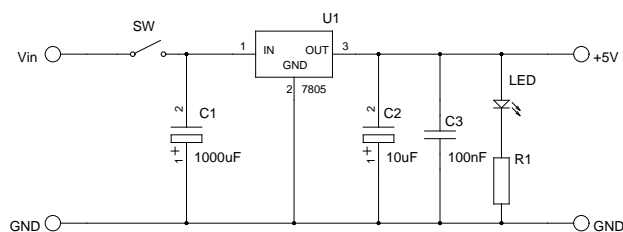
- LM317T Variable Voltage Regulator¹
- Using The LM317T To Regulate Voltage²
- LM317 Small Power Supply³

7.2. LM7805

Odkazy:

-
-
-

Obrázek 7-1. Zapojení obvodu LM7805

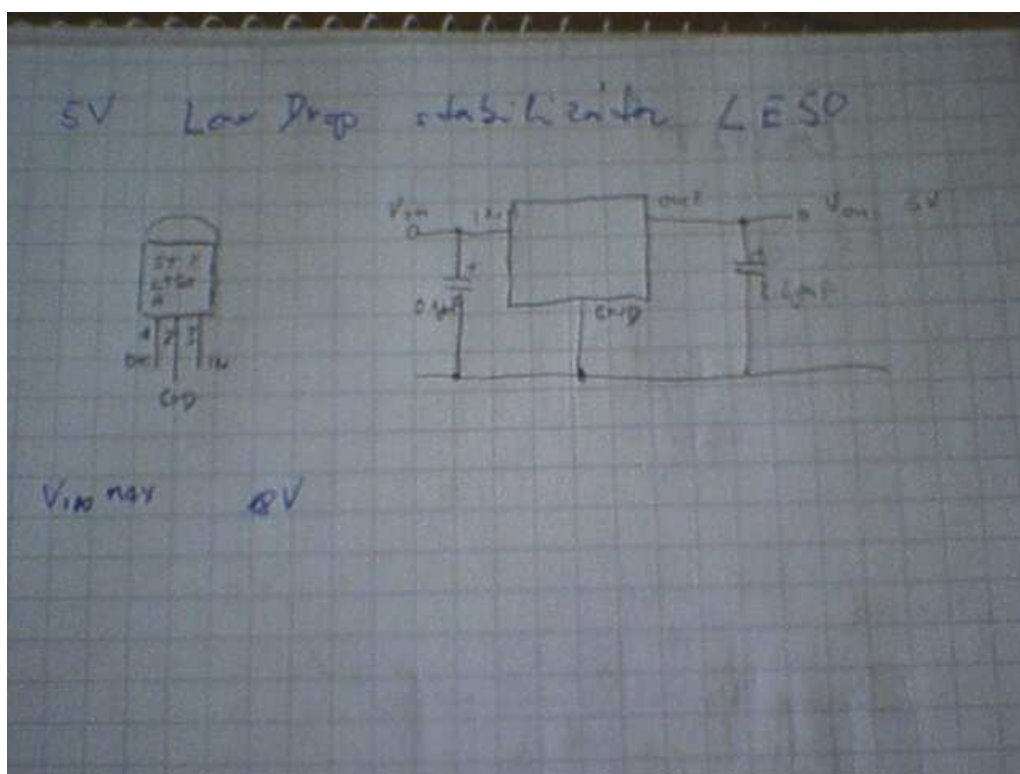


7.3. LE50A

Odkazy:

-
-
-

Obrázek 7-2. LE50 v pouzdře TO82 a zapojení



7.4. LT3802

*

Odkazy:

- LT3082 - 200mA Single Resistor Low Dropout Linear Regulator⁴
-

Poznámky

1. http://ourworld.compuserve.com/homepages/Bill_Bowden/page12.htm
2. <http://www.reuk.co.uk/Using-The-LM317T-To-Regulate-Voltage.htm>
3. <http://www.instructables.com/id/LM317-Small-Power-Supply/>
4. <http://www.linear.com/product/LT3082>

Kapitola 8. Konvertory napětí

Obvody pro změnu velikosti napětí.

Další, zatím nepopsané, obvody:

- intersil ICL7660, 7660Y, 7660S — invertor napětí
- ICL7662 — invertor napětí
- KA34063A SMPSC
- LT1303 — StepUp min cca 1.8V in. Výstupní napětí se řídí napětovým děličem. [cena v TME 169Kč bez DPH
- LT1303-5 — varianta LT1303 se zabudovaným napětovým děličem s nastavením na 5V.
-

8.1. Konvertory z nižšího na vyšší napětí (step-up)

Zvyšovače napětí typu Boost converter:

- Boost converter¹ na Wikipedii
- DC-DC converter to step up input voltage²
- A simple step-up converter³
- Convert IC with Integrated Schottky Diode Elektor Electronics rok 2005, číslo 7-8, strana 56. Popisovaný obvod je LT3464
- Run an uC from an AA-battery⁴ — článek popisující generátor napětí z jedné AA baterie, jako MCU je použit ATtiny13
- DC-DC Converter Basics⁵
- Arduino based Switchmode Voltage Regulator⁶
- Boost Switching Converter Design Equations⁷
- AVR forum - DC-DC booster and ATtiny internal ADC voltage reference⁸
-
-

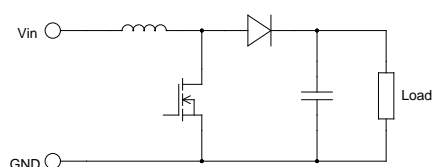
Součástky / IO:

- LT1300 - Micropower High Efficiency 3.3/5V Step-Up DC/DC onverter⁹
- LT1615-1 - Micropower Step-Up DC/DC Converters in ThinSOT¹⁰

Konstrukce:

- Minty Boost, How to: How to¹¹
-

Obrázek 8-1. Boost konvertor



8.1.1. MAXIM MAX756, MAX757

Odkazy:

- MAX756, MAX757 3.3V/5V/Adjustable-Output, Step-Up DC-DC Converters¹²
-

3.3V/5V/Adjustable-Output, Step-Up DC-DC Converters

For New Designs, We Recommend Improved Performance Devices, MAX1674/MAX1675/MAX1676

8.1.2. MAXIM MAX1674, MAX1675, MAX1676

Odkazy:

- MAX1674, MAX1675, MAX1676 High-Efficiency, Low-Supply-Current, Compact, Step-Up DC-DC Converters¹³
-
-

High-Efficiency, Low-Supply-Current, Compact, Step-Up DC-DC Converters

Compact, Low-Cost, Low-Noise Design: Inductor Damping Switch Eliminates Ringing, Improves EMI

8.1.3. LM2621 - Low Input Voltage, Step-Up DC-DC (National Semiconductor)

Odkazy:

- LM2621 - Low Input Voltage, Step-Up DC-DC¹⁴
-

Vlastnosti:

- Small Mini-SO8 Package (Half the Footprint of Standard 8-Pin SO Package)
- 1.09 mm Package Height
- Up to 2 MHz Switching Frequency
- 1.2V to 14V Input Voltage
- 1.24V - 14V Adjustable Output Voltage
- Up to 1A Load Current
- 0.17Ω Internal MOSFET
- Up to 90% Regulator Efficiency
- 80μA Typical Operating Current
- <2.5μA Guaranteed Supply Current In Shutdown

8.2. Konvertory z vyššího napětí na nižší (step-down)

Odkazy:

- Low-cost Step-down Converter with Wide Input Voltage Range¹⁵ — Elektor Electronics rok 2005, číslo 7-8, strana 103
- 1958: Invention of the Integrated Circuit¹⁶
- The first monolithic integrated circuits¹⁷

- LT1074¹⁸
-

8.3. Konvertory napětí Step-Up / Step-Down

Odkazy:

- DC/DC měniče s MC34063¹⁹
-
-
-

Zapojení a integrované obvody které mohou konvertovat napětí jak směrem dolů, tak směrem nahoru.

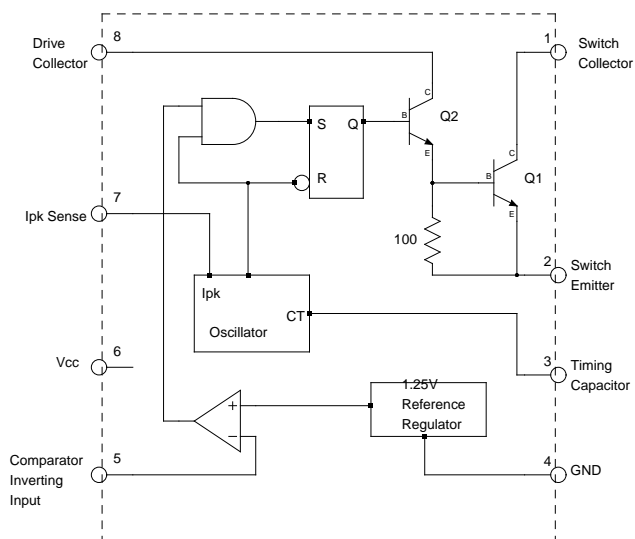
Obvod který doporučuje Dave L. Jones: MC34063. Vyrábí nejen Motorola ale další výrobci. Vyhneme se tím závislosti pouze na jednom dodavateli. Další výhodou pro takové to domácí kutění je dostupnost obvodu v pouzdru DIL-8.

Další obvody typu 34063:

- MIK34063A
- PJ34063 — GM měl vadnou sérii viz Pozor na radice DC-DC menicu 34063 z GM²⁰ [2006-03-10]
- IT34063
- MC33063
-

Obvod je v pouzdře s osmi vývody a jeho vnitřní zapojení je na následujícím obrázku.

Obrázek 8-2. Vnitřní zapojení obvodu MC34063



Zapojení obvodu se může jevit náročnější než zapojení jiných obvodů. Ale pokud projdeme počlivě jednotlivé výpočty, dosáhneme kýženého cíle.

Vlastnosti obvodu:

- Pracovní vstupní napětí od 3.0V do 40V.
- Maximální spínaný proud 1.5A.
- Kmitočet 100kHz

-
-
-

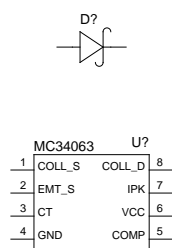
Zapojení Step-Up

Návrh STEP-UP konvertoru s Davem Jonesem na YouTube²¹.

$$V_{IN} = 5V \pm 10\%$$

$$V_{OUT} = 15V \text{ } 100mA \text{ (100mV Ripple)}$$

Obrázek 8-3. Vnitřní zapojení obvodu MC34603



8.4. Sběr energie (*Energy Harvesting*)

Odkazy:

- LTC3588²²
- LTC3109²³
-

Poznámky

1. http://en.wikipedia.org/wiki/Boost_converter
2. <http://www.rowan.sensation.net.au/electronics/stepup.html>
3. <http://www.xs4all.nl/~odu/dcdc.html>
4. <http://spritesmods.com/?art=ucboost&page=1>
5. http://www.powerdesignersusa.com/InfoWeb/design_center/articles/DC-DC/converter.shtm
6. http://www.powerdesignersusa.com/InfoWeb/design_center/articles/DC-DC/converter.shtm
7. <http://www.daycounter.com/LabBook/BoostConverter/Boost-Converter-Equations.phtml>
8. <http://www.avrfreaks.net/index.php?name=PNphpBB2&file=printview&t=60473&start=0>
9. <http://www.linear.com/pc/productDetail.jsp?navId=H0,C1,C1003,C1042,C1035,P1449>
10. <http://www.linear.com/pc/productDetail.jsp?navId=H0,C1,C1003,C1042,P1775>
11. <http://www.ladyada.net/make/mintyboost/process.html>
12. http://www.maxim-ic.com/quick_view2.cfm/qv_pk/1167/t/do
13. http://www.maxim-ic.com/quick_view2.cfm/qv_pk/1878/t/al

14. <http://www.national.com/pf/LM/LM2621.html>
- 15.
16. <http://www.pbs.org/transistor/background1/events/icinv.html>
17. <http://homepages.nildram.co.uk/~wylie/ICs/monolith.htm>
- 18.
19. <http://amarokcz.wz.cz/MC34063.htm>
20. <http://list.hw.cz/pipermail/hw-list/2006-March/269035.html>
21. <http://www.youtube.com/watch?v=qGp82xhybs4>
- 22.
- 23.

Kapitola 9. Operační zesilovače

*

Odkazy:

- Operační zesilovač¹
- Zapojení s operačním zesilovačem²
- Operational amplifier³ na Wikipedii

-
-
-
-

Odkazy:

- 1458⁴
-

Poznámky

1. http://cs.wikipedia.org/wiki/Opera%C4%8Dn%C3%AD_zesilova%C4%8D
2. http://cs.wikipedia.org/wiki/Zapojen%C3%AD_s_opera%C4%8Dn%C3%ADm_zesilova%C4%8Dem
3. http://en.wikipedia.org/wiki/Operational_Amplifier
- 4.

Kapitola 10. Zvláštní obvody

Ostatní obvody:

- TDA0161 — Proximity Detector
-

Odkazy:

- METAL PROXIMITY DETECTOR¹ s TDA0161
-

10.1. Funkční generátory

*

Odkazy:

- High Frequency Function Generator with the Maxim MAX038²
-

Existuje několik specializovaných obvodů, které plní funkce generátorů.

Obvody:

- Maxim MAX038³ — High-Frequency Waveform Generator, již se nevyrábí
- ICL8038⁴ [\$23.90]
- XR2206⁵ — Monolithic Function Generator [2011-03-28, 65.58Kč, Farnel]
- XR2209⁶ — Voltage-Controlled Oscillator [2011-03-28, 35.50Kč, Farnel]
- AD537⁷
- NTE864⁸ Integrated Circuit Precision Waveform Generator. Low Frequency Drift with Temperature: 250ppm/C. Low Distortion: 1%. Wide Frequency Range: 0.001Hz to 300kHz. Variable duty Cycle 2% to 98%. High Level Outputs: TTL to 28V.
- AD9833⁹ — programable Waveform Generator (build with an DDS) from 0 to 12.5 MHz [2011-03-28, 252.39Kč, Farnel]
- AD9834¹⁰
- AD9835¹¹
- AD9854¹²
- 8038PCD¹³
- XR8038¹⁴
- AD821¹⁵
-
-

Stavebnice a konstrukce:

- Funkční generátor s XR2206¹⁶
- Bauplan Schaltplan XR2206 Funktionsgenerator¹⁷
- Triangle and Squarewave Generator¹⁸ z 1458
- Analog Waveform Generator Module for the HB Bench¹⁹
- Simple Function generator with LM566²⁰ www.elecfree.com
- Function Generator using LF351²¹
-
-
-
-

* AD9851

* Obvod ICL8038 lze asi za \$5 koupit v iStore²².

10.2. NE555, NE556

Odkazy:

- Wikipedia: 555 timer IC²³
- Kniha Designing Analog Chips²⁴ od Hanse Camenzinda, dizajnéra NE555
- 555 Contest²⁵ „You’ve got 8 pins... and one shot.“
- learn about the 555²⁶ na Instructables. Mimo jiné jsou zde i často opakované chyby v zapojení.
- Obvod NE555²⁷ podrobný rozbor zapojení v čteně odvození rovnic

Nezpracované odkazy:

- AF variable frequency generator with a 555²⁸
- 555 Timer Tutorial²⁹
- THE 555 TIMER³⁰
- Wikibooks: Praktická elektronika/Generátory/555³¹
- A Stylophone³² — jednoduchý hudební nástroj s NE555 a LM386
- Know Your IC: 555 Timers³³
- The 555 Precision Timer IC³⁴
-

Zajímavé konstrukce z 555 Contest³⁵:

- 555 Adding Machine³⁶ — klasická stará mechanická sčítací kalkulačka realizovaná elektronicky s obvody 555
- A DIY Square Wave Signal Generator with Pulse Width Modulation³⁷
- PWM Generators³⁸
- 555 Function Generator³⁹
- Pre-Lab Exercise 6⁴⁰
- 555 Oscillator⁴¹
- 555 Timer Voltage-Controlled Oscillator⁴²
- DC Voltage Polarity Inverter⁴³
-

Integrovaný obvod, časovač 555, byl navržen Hansem Camenzindem v roce 1970 a na trh uveden v roce 1971 firmou Signetics (později koupenou Philipsem). Jedná se o jednoduchý a velmi univerzální integrovaný obvod.

Původní označení bylo SE555 pro obvod v kovovém kulatém pouzdru a NE555 v plastickém DIP pouzdru. Popis byl "The IC Time Machine".

Obvod NE556 je integrovaný obvod obsahující dva časovače 555 v jednom pouzdru.

Obvod NE558 v 16-ti vývodovém DIP pouzdře obsahuje 4 mírně modifikované časovače 555.

Na trhu jsou také implementace časovače s velmi nízkou spotřebou, vyrobené technologií CMOS. (7555, TLC555).

* Schematický obrázek vnitřního zapojení.

* Detailní obrázek vnitřního zapojení.

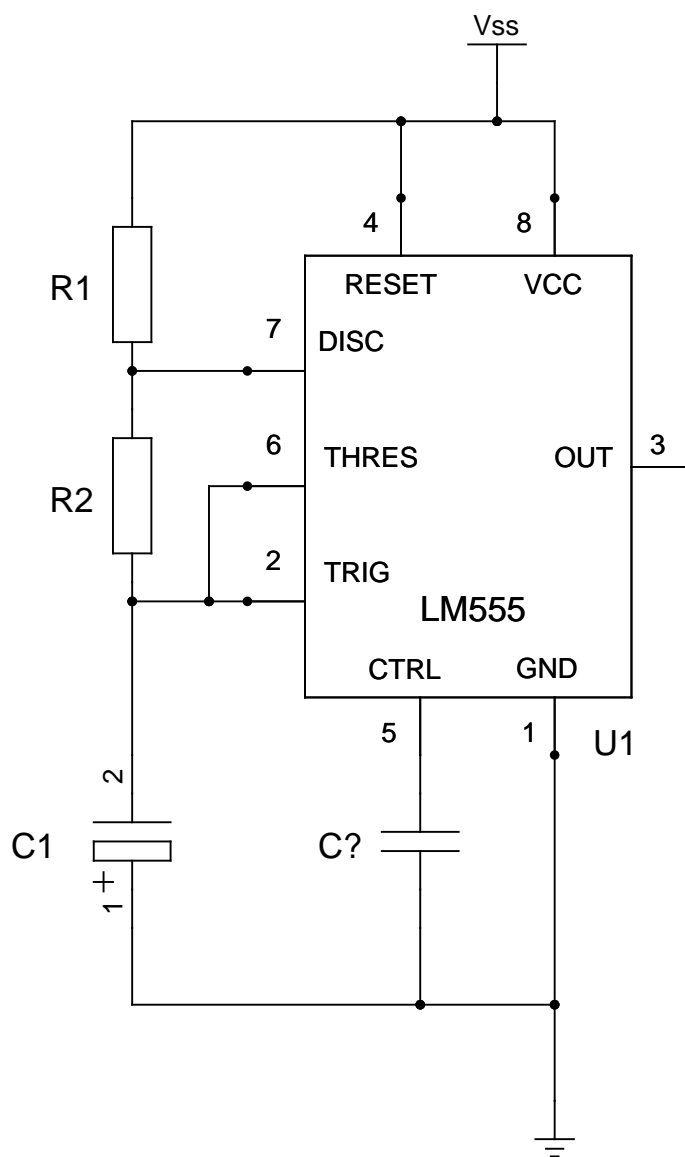
* Základní zapojení astabilního klopného obvodu.

Kmitočet (f) a perioda (T) kmitů je určena vzorcem:

$$T = 0.7 * (R1 + 2 * R2) * C1$$

$$f = 1.4 / [(R1 + 2 * R2) * C1]$$

Obrázek 10-1. Astabilní klopný obvod s NE555



* Základní zapojení monstabilní klopný obvod

10.3. Rádiové přijímače či jejich části

*

Odkazy:

-
-

Jiné obvody k prozkoumání:

- Z416E

- MC2833 FM Transmitter
- BA14014 FM Stereo Transmitter
- MC3362P FM Receiver
- LM372
-

10.3.1. ZN414 one chip AM radio

* *Attributy: id="ZN414"*

Odkazy:

- ZN414⁴⁴ na Wikipedii
- MK484⁴⁵ na Wikipedii
- ZN414 Portable Receiver⁴⁶ Andy Collinson
- MK484 MW Receiver⁴⁷
- MK484 Radio with Loudspeaker⁴⁸ Chad Castagana
- A simple MK 484 IC tuned rf (trf) radio⁴⁹
- ZN414 AM Receiver IC⁵⁰
- Přijímač sv, dv⁵¹
-

Podobné či ekvivalentní čipy:

- MK484
- ZN484
- TA7642
- ZN414Z
- YS414
-
-
-

* *Zjistit jestli se dá sehnat a zkusit jednoduchou konstrukci. Farnell nemá tento obvod v nabídce! Vyskytuje se na Ebay za cca £5.0.*

* *ZN414 je nahrazen novějším kompatibilním obvodem MK484. Farnell tento obvod nemá v nabídce!*

Další náhrada je TA7642.

10.3.2. NE602N, NE612AN (SA602AN, SA612AN)

Odkazy:

- AM přijímač s NE612⁵²
-
-
-

SA612AN (NE612N) — Double-balanced mixer and oscillator

SA612AD Integrovaný obvod VHF Mix/Osc Bandwith 500MHz SO8, [85Kč TME]

10.4. Spínací tranzistorová pole

*

Jiné obvody k prozkoumání:

- ULN2803A — aktivní výstup 0V
-
-
-

10.4.1. ULN2803

*

Obvody ULN280x obsahují 8 darlingtonových párů. Tyto obvody jsou určeny ke spínání větších proudů a napětí než zvládnou běžné digitální obvody či mikrořadiče.

* *Looks fine, most use a ULN2803 to do the same thing as it has the diode clamps and higher gain than the 2N2222A plus a nice simple DIP package.*

Tabulka 10-1.

ULN2801A	PMOS-CMOS	
ULN2802A	14-15V PMOS	
ULN2803AP/AFW	TTL, 5V CMOS	
ULN2804AP/AFW	6~15V PMOS, CMOS	
ULN2805	TTL	

Odkazy:

- Výstupní proud 500mA
- Maximální spínané napětí 50V
- Power Dissipation 1.47W
-

10.5. Řízení motorů

*

Čipy:

- SN754410⁵³ H-bridge
- L293D⁵⁴ — Standard Quadruple Half-H Driver
- L298⁵⁵ DUAL FULL-BRIDGE DRIVER, celkový proud 4A
- UDN2547B⁵⁶ 600mA
-
-

10.5.1. Řízení krokových motorů

*

Odkazy:

- Alegro A3967⁵⁷ Microstepping Driver with Translator. Osazená deska je v nabídce Sparkfun jako ROB-10267⁵⁸ za \$14.95
- UDX2916 Dual Full-Bridge PWM Motor Driver⁵⁹
- LMD18245⁶⁰ 3A, 55V DMOS Full-Bridge Motor Driver
-

10.6. TAA661 (MAA661)

*

Odkazy:

- MAA661⁶¹
-
- Přijímač FM s MAA661⁶²
-

Mezifrekvenční zesilovač. Obvod sestává z několika bloků: vf zesilovače, koincidenčního detektoru, nf zesilovače.

10.7. LM3909

*

Odkazy:

- LED Counter⁶³ na Instructables
- LM3909⁶⁴
-
-

Univerzální integrovaný obvod pro blikáče.

Poznámky

1. <http://lets makerobots.com/node/24057>
2. http://pcbheaven.com/projectpages/High_Frequency_Function_Generator/?p=0&topic=worklog
- 3.
- 4.
- 5.
- 6.
- 7.
- 8.
- 9.
- 10.
- 11.
- 12.

- 13.
- 14.
- 15.
16. http://pandatron.cz/?1247&funkcni_generator_s_xr2206
17. <http://www.loetstelle.net/projekte/xr2206/xr2206.php>
18. <http://www.eleccircuit.com/triangle-and-squarewave-generator/>
19. http://www.ac0c.com/main/page_antennas__homebrew_xr2206_sig_gen.html
20. <http://www.wiparat.com/simple-function-generator-with-lm566/>
21. <http://www.wiparat.com/function-generator-using-lf351/>
22. http://iteadstudio.com/store/index.php?main_page=product_info&cPath=41_48&products_id=193
23. http://en.wikipedia.org/wiki/555_timer_IC
24. <http://www.designinganalogchips.com/>
25. <http://www.555contest.com/>
26. <http://www.instructables.com/id/learn-about-the-555/>
27. <http://alzat.szm.sk/Oscilat/generato/ako/ne555.htm>
28. <http://users.belgacom.net/hamradio/schemas/555osc1.gif>
29. <http://www.matni.com/Arabic/Elec-Info/NE555%20DETAILS/555.html>
30. http://www.ee.nmt.edu/~thomas/ee322_s07/labs/555_timer.html
31. http://cs.wikibooks.org/wiki/Praktick%C3%A1_elektronika/Gener%C3%A1tory/555
32. <http://www.instructables.com/id/A-Stylophone/>
33. <http://www.instructables.com/id/Know-Your-IC-555-Timers/>
34. <http://tronixstuff.posterous.com/the-555-precision-timer-ic>
35. <http://www.555contest.com/>
36. <http://www.vk2zay.net/article/258>
37. http://www.rmcybernetics.com/projects/DIY_Devices/homemade_signal_generator2.htm
38. http://www.m.case.btinternet.co.uk/html/pwm_generators.html
39. <http://forum.allaboutcircuits.com/showthread.php?t=12405>
40. <http://www.ee.nmt.edu/~tubesing/ee101/labs/6lab/6lab.html>
41. http://www.electronics-tutorials.ws/waveforms/555_oscillator.html
42. <http://www.ecelab.com/circuit-vco-555.htm>
43. <http://www.ecelab.com/circuit-polarity-inv.htm>
44. <http://en.wikipedia.org/wiki/ZN414>
45. <http://en.wikipedia.org/wiki/MK484>
46. <http://www.zen22142.zen.co.uk/Circuits/rf/ZN414.htm>
47. <http://www.vk2zay.net/article/116>
48. <http://www.zen22142.zen.co.uk/Circuits/rf/chad484.htm>
49. <http://bellsouthpwp2.net/w/u/wuggy/mk484.htm>

50. <http://cool386.tripod.com/zn414/zn414.html>
51. <http://www.pigibastleni.jex.cz/menu/schemata/prijimace/prijimac-sv-dv>
52. http://pandatron.cz/?614^am_prijimac_s_ne612
53. <http://focus.ti.com/docs/prod/folders/print/sn754410.html>
- 54.
55. <http://www.st.com/stonline/books/pdf/docs/1773.pdf>
- 56.
57. http://www.allegromicro.com/en/Products/Part_Numbers/3967/
58. <http://www.sparkfun.com/products/10267>
59. http://www.allegromicro.com/en/Products/Part_Numbers/2916/index.asp
60. <http://www.national.com/mpf/LM/LMD18245.html>
61. http://www.edunet.souepl.cz/~weisz/dilna/en_kat/maa661.php
62. <http://www.jerrycb.wz.cz/schemata/prij04.htm>
63. <http://www.instructables.com/id/LED-Counter>
64. http://pandatron.cz/?207&lm3909_-_1.dil

Kapitola 11. Zesilovače

Odkazy:

-

Čipy k prozkoumání:

-
- TDA7268 (?stereo, DIP16)
-
-

11.1. LM386 (*Low Voltage Audio Power Amplifier*)

*

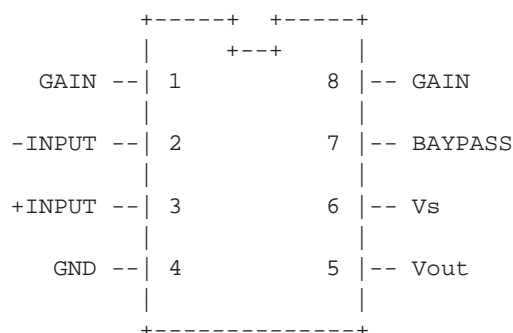
Odkazy:

- National Semiconductor LM386 Low Voltage Audio Power Amplifier¹
- NF zesilovač LM386²
- NF zesilovač s LM386 s NÍZKÝM ŠUMEM³
- LM386 Low Voltage Audio Power Amplifier⁴
- Small audio amplifiers⁵
-
-
-
- Zesilovač s LM386- 1Wattový monozesilovač⁶
- How to get 74dB by LM386⁷
- 74 dB na LM386⁸

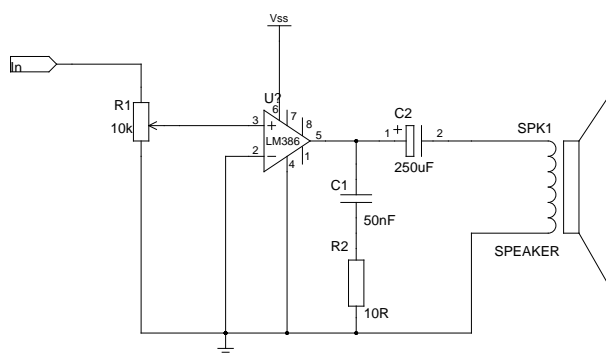
Konstrukce s LM386:

- Zesilovač s LM386⁹ [2008-10-09] OK1KVK / OL7C
-
-

Obrázek 11-1. Zapojení vývodů pouzdra LM386

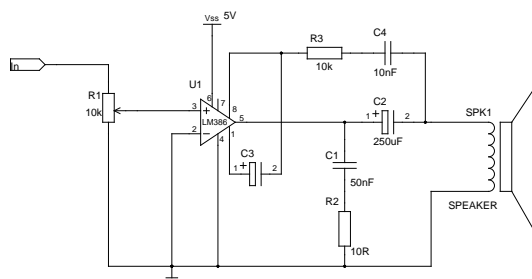


Obrázek 11-2. Zapojení zesilovače s LM386



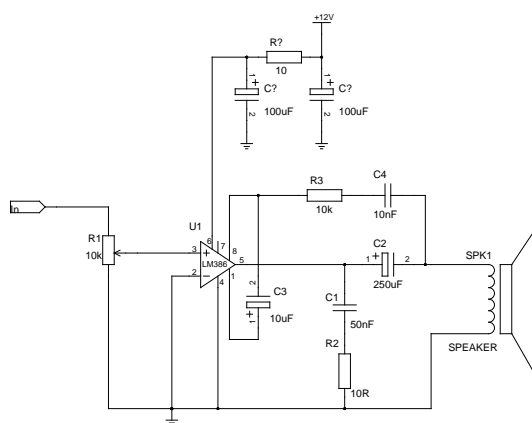
Mezy vývody 1 a 8 se zapojují součástky určující zisk. Bez zapojení jakýchkoliv součástek je zisk 20. Pokud mezi vývody zapojíme elektrolytický kondenzátor o kapacitě $10\mu\text{F}$, bude zisk 200. Pokud sériově s tímto kondenzátorem zapojíme rezistor o odporu $1.2\text{k}\Omega$, bude zisk 50.

Obrázek 11-3. Zapojení zesilovače s LM386 s nízkým šumem



* GEDA, Width*Height: 800x600.

Obrázek 11-4. Zapojení zesilovače s LM386



* GEDA, Width*Height: 800x600.

11.2. TDA2822

*

Odkazy:

- TDA2822M - Simple low power stereo amplifier¹⁰
- More TDA2822 Amplifier Developments¹¹
- KA2209 / TDA2822M¹² Mini Stereo Amplifier
-

Poznámky

1. <http://www.national.com/mpf/LM/LM386.html>
2. <http://www.8bitu.cz/clanek/nf-zesilovac-lm386/>
3. <http://www.radio.qrp.cz/2/clanky/lm386.htm>
4. <http://www.electroniccircuits.com/electronic-circuits/lm386-low-voltage-audio-power-amplifier/>
5. <http://ludens.cl/Electron/audioamps/AudioAmps.html>
6. http://www.volta.estranky.cz/clanky/houkacky_-sireny/zesilovac_slm_386
7. <http://www.intio.or.jp/jf10zl/LM386.htm>
8. http://ok1like.c-a-v.com/soubory/74_lm386.htm
9. <http://www.ok1kvk.cz/web/index.php/technika-a-bastelni/41-bastleni/405-zesilova-s-lm386>
10. <http://www.electro-dan.co.uk/electronics/TDA2822.html>
11. http://www.electro-dan.co.uk/electronics/more_tda2822.html
12. <http://yeungmea.com/KA2209.htm>

III. Číslicová technika

Odkazy:

- Smithsonian The Chip Collection¹⁶⁹
- IEEE STANDARD SYMBOLS¹⁷⁰ — pdf dokument
-
-

Odkazy:

- Základy číslicové techniky¹⁷¹
-
-
-
-
-

Normy pro kreslení:

Odkazy:

- MIL-STD-806 [1950]
- ANSI/IEEE Std 91-1984
- ANSI/IEEE Std 91a-1991
- IEC 60617-12
- EN 60617-12:1999 (Evropská norma)
- BS EN 60617-12:1999 (Britská norma)

Číslicová technika je obor elektroniky, který se zabývá zpracováním nespojitých signálů jenž nabývají dvou úrovní, nazývaných logické úrovně. Její základy můžeme najít v réleové technice, kde elektromagnetická relé byla užita jako základní komponenty jednoduchých logických (digitálních) obvodů. Největší nasazení číslicové techniky přichází s nástupem číslicových počítačů a trvá dodnes.

V této části se zabývám číslicovou technikou od základů. Tedy je zde nějaká matematika, popisy základních hradel a také mě známé konstrukce hradel.

* *TODO: vyhledat historii číslicové techniky!*

* *<http://www.electronicresourceonline.com/digital-electronics/> The inception of digital electronics started with the electronic circuit or gate which was invented in 1835 by Joseph Henry.*

Poznámky

169. <http://smithsonianchips.si.edu/>

170. www.ddpp.com/DDPP3_pdf/IEEEsyms.pdf

171. <http://telefon.unas.cz/e/>

Kapitola 12. Hradla

Základní stavební kameny digitálních obvodů

*

12.1. Přehled typů hradel

*

Odkazy:

- Logic gate¹ na Wikipedii

We have a couple basic gates which can be used to build any component.

* Zde ukážu základní hradla a jejich funkci.

12.1.1. Zesilovač, hradlo YES, Buffer

* *Attributy: id="gate.yes"*

Odkazy:

- 16.3.1

Tento obvod se někdy ani za hradlo nepovažuje. Hodnota kterou dává na výstupu je stejná jako ta na vstupu. Někdy se mu říká buffer, nebo zesilovač. Hlavní použití je jako výstupní člen, kdy výstupní obvody hradla mohou pracovat s větším (jiným) napětím než vstupní obvod a také mohou dodávat větší proud.

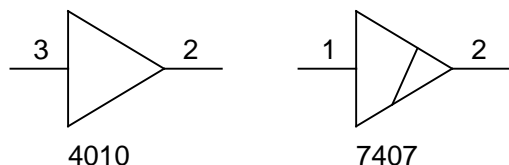
Tento logický člen má jen jeden vstup a jeden výstup. Funkce je dána tabulkou.

Tabulka 12-1. Funkční tabulka hradla YES

A	Y
0	0
1	1

Praktické použití je jako zesilovač, oddělovač, převodník napětíových úrovní.

Obrázek 12-1. Americký symbol pro hradlo YES



* gEDA image: size 320*240 PNG, EPS.

12.1.2. Invertor, hradlo NOT

* *Attributy: id="gate.not"*

Odkazy:

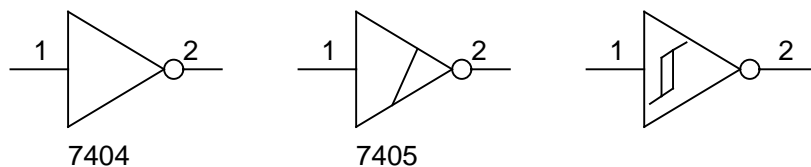
- 16.3.2

Tento logický člen má jen jeden vstup a jeden výstup. Funkce je dána tabulkou.

Tabulka 12-2. Funkční tabulka invertoru

A	Y
0	1
1	0

Obrázek 12-2. Americký symbol pro invertor NOT



* Image size 640*480 PNG, EPS.

12.1.3. Hradlo AND

* *Attributy: id="gate.and"*

Odkazy:

- 16.3.3

Rovnice 12-1. Rovnice

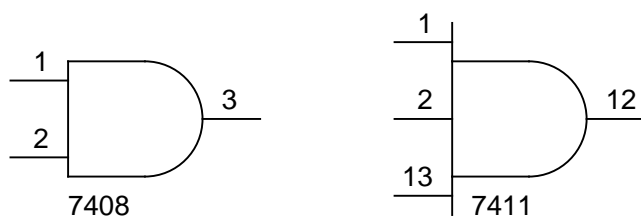
$$Y = A \& B$$

Tabulka 12-3. Funkční tabulka AND

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Schematické značky hradla AND

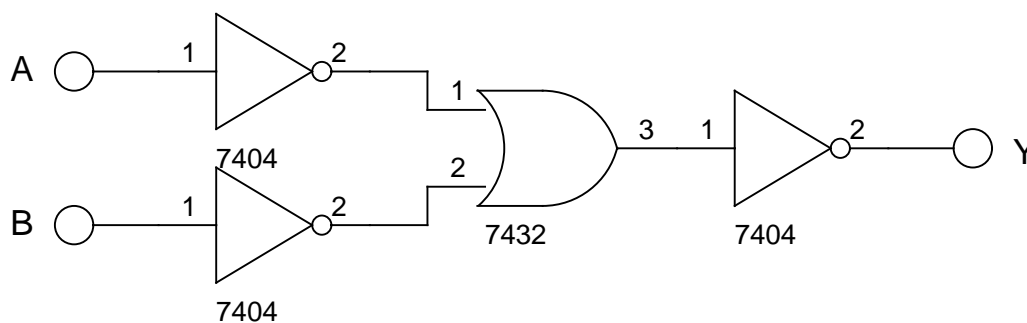
Obrázek 12-3. Americký symbol pro dvou a třívstupové hradlo AND



* Image size 320*240 PNG, EPS.

Ekvivalentní schéma hradla AND.

Obrázek 12-4. Náhrada AND



* Image size 640*480 PNG, EPS.

12.1.4. Hradlo OR

* *Attributy:* id="gate.or"

Odkazy:

- 16.3.5

Rovnice 12-2. Rovnice

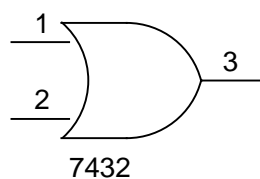
$$Y = A \vee B$$

Tabulka 12-4. Funkční tabulka OR

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Schematické značky hradla OR

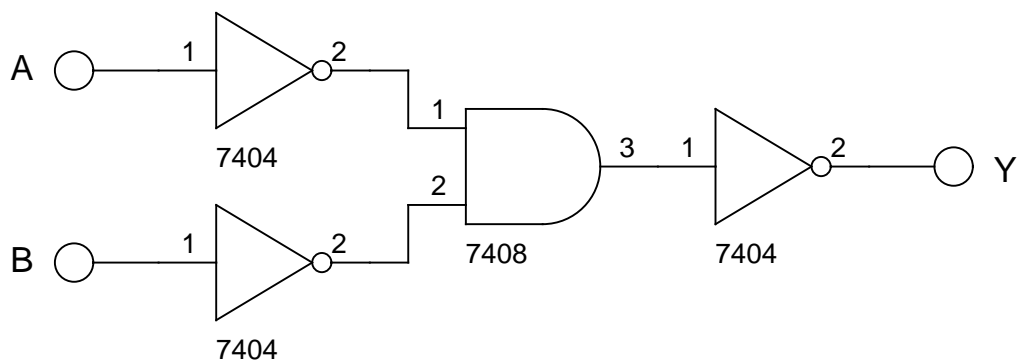
Obrázek 12-5. Americký symbol pro dvou a třívstupové hradlo OR



* Image size 320*240 PNG, EPS.

Ekvivalentní schéma hradla OR.

Obrázek 12-6. Náhrada OR



* Image size 640*480 PNG, EPS.

12.1.5. Hradlo NAND

*

Odkazy:

- [xref linkend="ic.logic.nand"/](#)
-

Tabulka 12-5. Tabulka výstupu dvouvstupového hradla NAND

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Tabulka 12-6. Tabulka výstupu třívstupového hradla NAND

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Obecně pro jakákoliv další vícevstupová hradla platí že hradlo dává v případě samých 1 na vstupu výstup 0. Všechny ostatní kombinace vstupů dávají výstup 1.

12.1.6. Hradlo NOR

*

12.1.7. Hradlo XOR

* *Attributy: id="gate.xor"*

Odkazy:

- 16.3.7

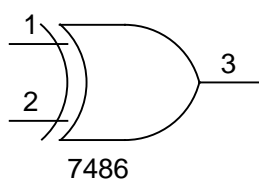
-

Tabulka 12-7. Funkční tabulka XOR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Schematická značky hradla XOR

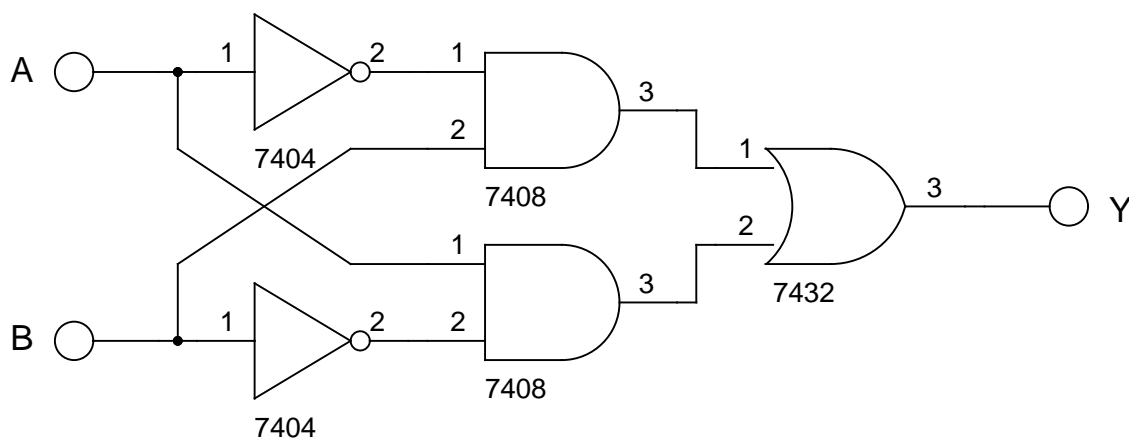
Obrázek 12-7. Americký symbol pro dvouvstupové hradlo XOR



* *Image size 320*240 PNG, EPS.*

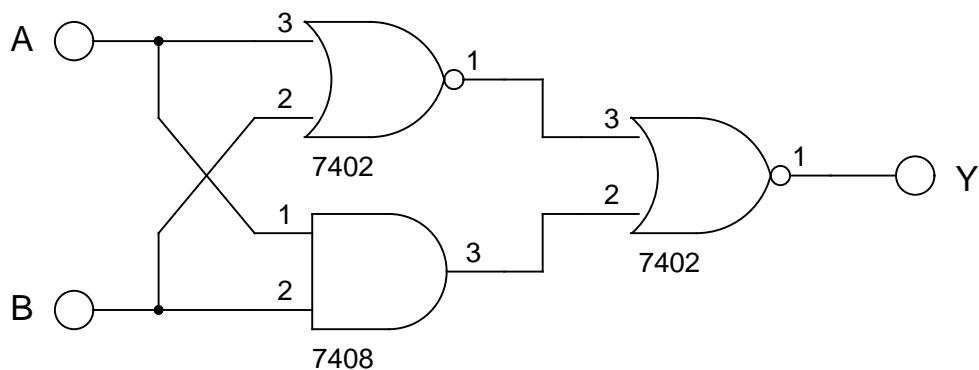
Ekvivalentní schéma hradla XOR.

Obrázek 12-8. Náhrada hradla XOR



* *Image size 640*480 PNG, EPS.*

Obrázek 12-9. Náhrada hradla XOR



* Image size 320*240 PNG, EPS.

12.1.8. Hradlo XNOR

*

12.1.9. Hradlo C

Odkazy:

- C-element² na Wikipedii

12.1.10. Asymetrické hradlo C

Odkazy:

- Asymmetric C-element³ na Wikipedii

12.2. Konstrukce digitálních hradel

- How to build your own logic inverter⁴

Digitální hradla lze konstruovat mnoha způsoby. A také tomu tak v minulosti bylo. V rámci hledání nejlepší cesty byly konstruovány nejdříve jednoduchá zapojení, v dobách kdy cena součástek byla určující, až po složitá zapojení s mnoha součástkami na křemíkovém substrátu, když byla zvládnuta výroba integrovaných obvodů.

12.2.1. DL (*Diode Logic*)

* *Attributy:* id="Diode_Logic"

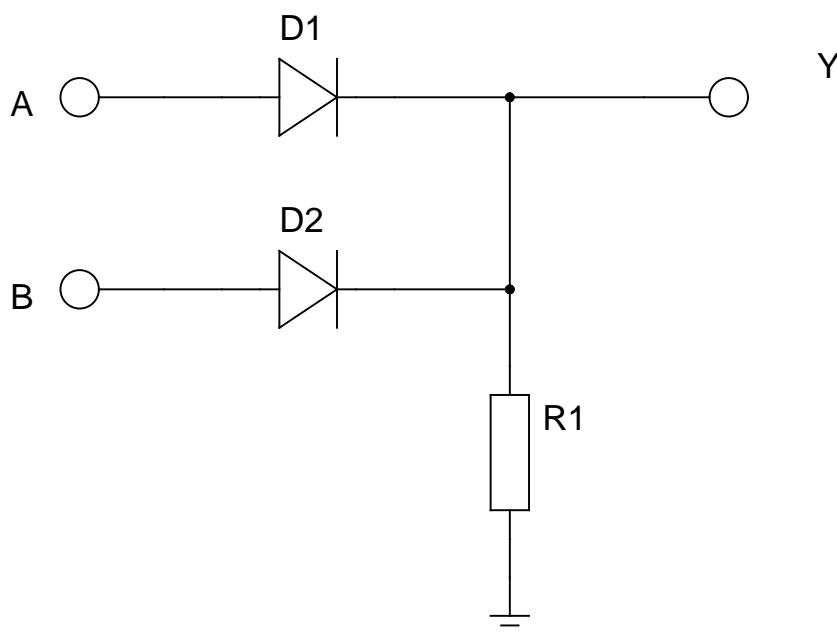
Odkazy:

- Wikipedia: Diode logic⁵
- Číslicová technika - 10 - Diodová logika⁶ a 11 - Diodová logika 2⁷

- Základní logické členy⁸
-
-

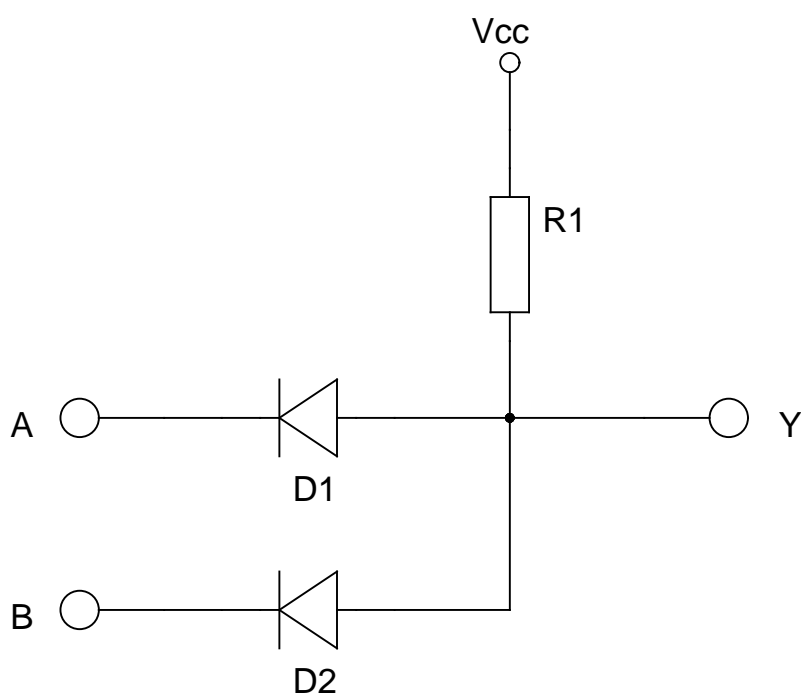
Diodová logika je jednou z nejjednodušších. Využívá ve své funkci diod a jejich schopnosti v jednom směru proud propouštět a v opačném ne. K její realizaci si tedy vystačíme jen s polovodičovými diodami a odpory.

Obrázek 12-10. Zapojení hradla OR diodové logice



* gEDA image: 320×240 EPS, PNG

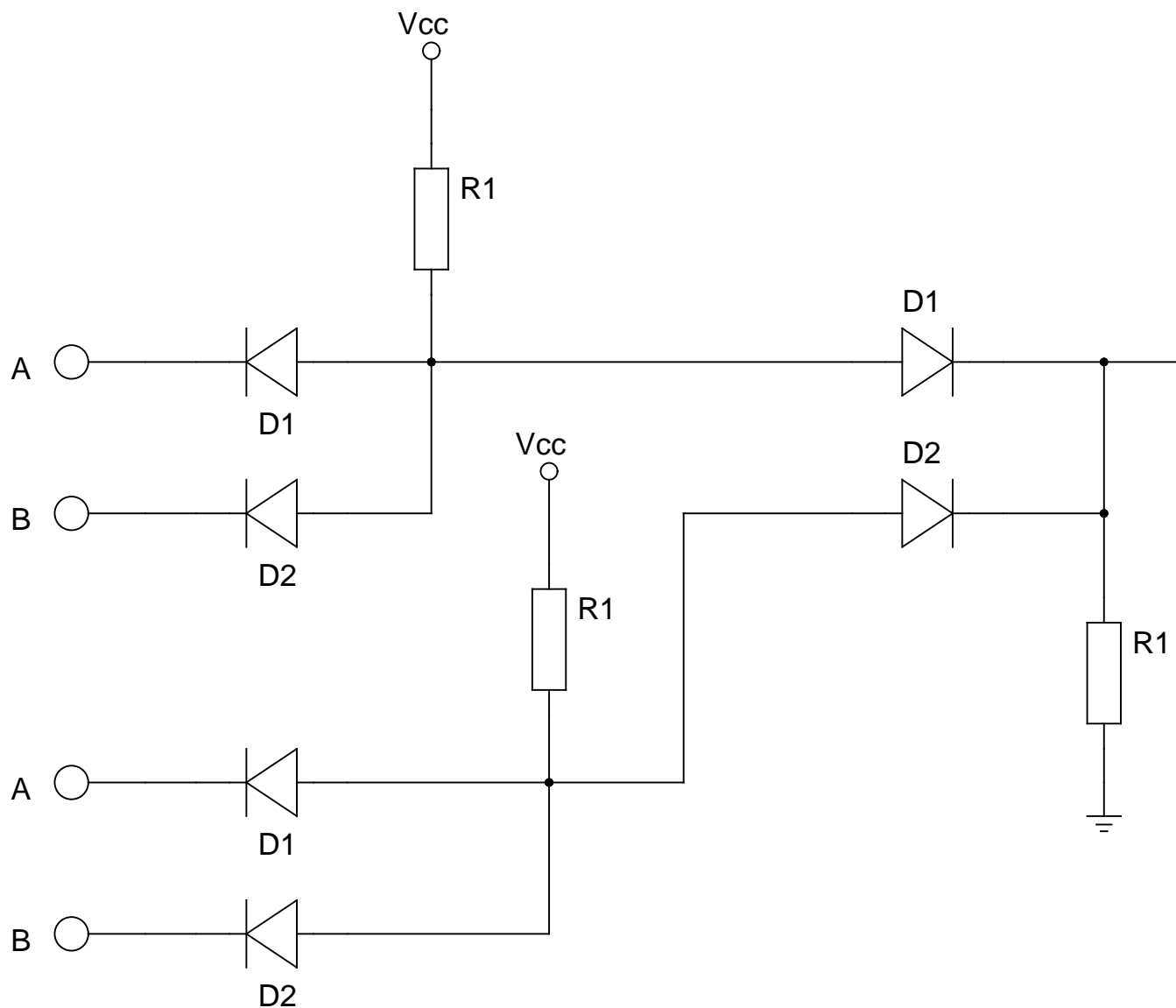
Obrázek 12-11. Zapojení hradla AND diodové logice



* gEDA image Diode_Logic_AND.sch: 320×240 EPS, PNG

Vzhledem k principům realizace diodových hradel je není možno kombinovat libovolným způsobem. A protože v diodové logice není možno realizovat invertor, není ani možné realizovat jiná hradla než AND a OR.

Obrázek 12-12. Zapojení kombinace hradel AND a OR v diodové



* gEDA image Diode_Logic_AND_OR.sch: 640×480 EPS, PNG

12.2.2. DTL (*Diode Transistor Logic*)

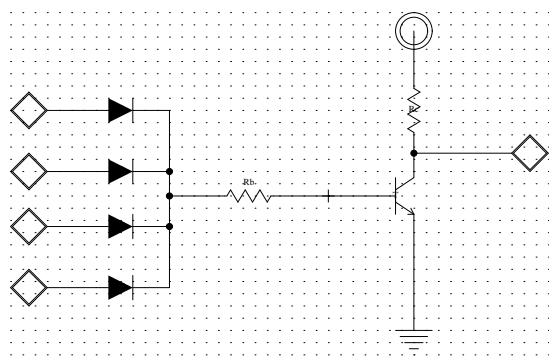
* *Attributy: id="DTL"*

Odkazy:

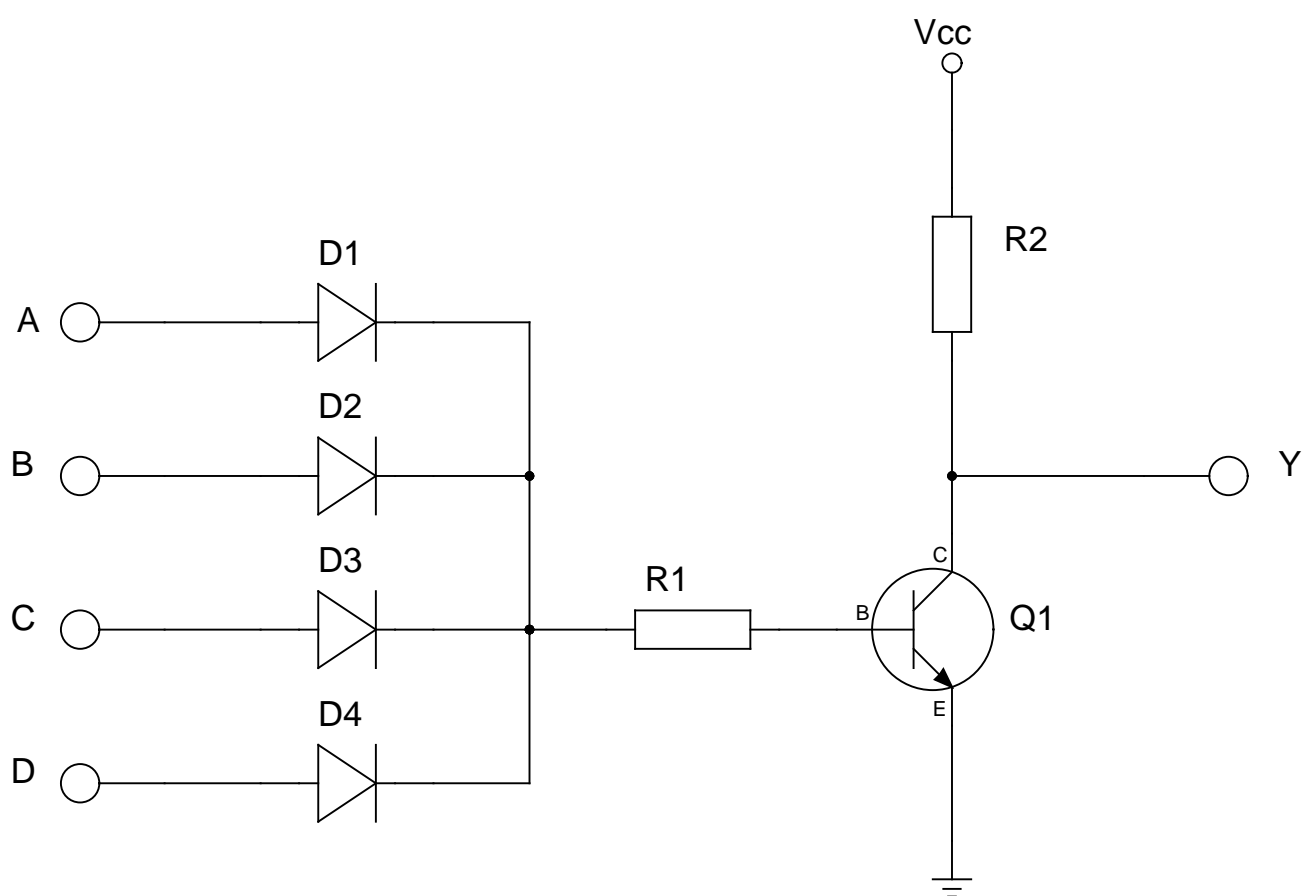
- DIODE-TRANSISTOR LOGIC⁹
- Casio AL-1000 Calculator Technical Description: Electronic Implementation of Logic¹⁰

Přidáním tranzistoru, jako aktivního prvku zajišťujícího funkci invertoru, se z diodové logiky stává DTL logika. V této logice, protože má v sobě prvek invertoru, již můžeme realizovat všechna hradla. Je tedy úplná.

Obrázek 12-13. DTL hradlo NOR (v Electric)

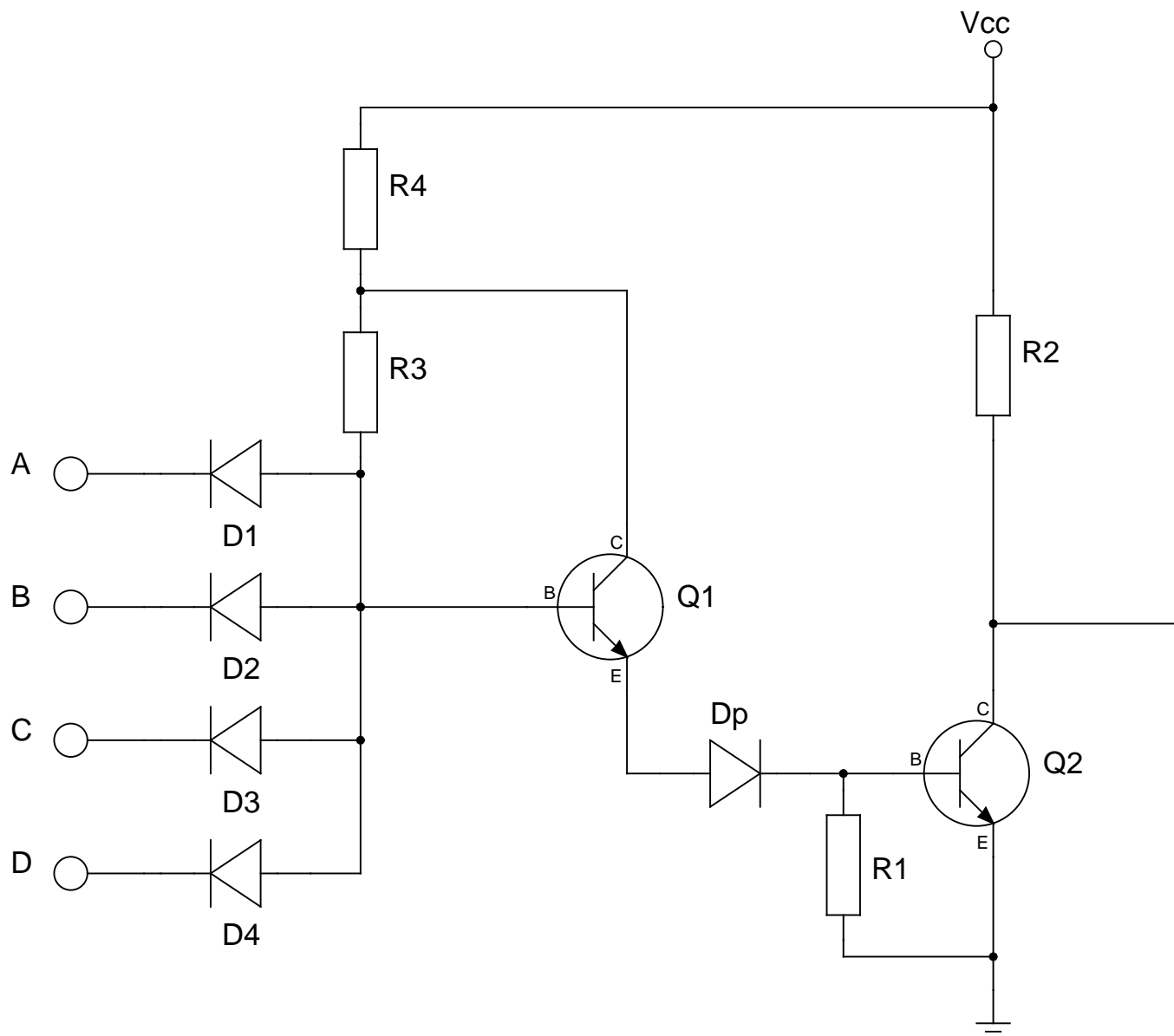


Obrázek 12-14. Čtyřvstupové hradlo NOR v DTL



* gEDA image DTL_NOR:sch: 640×480 EPS, PNG

Obrázek 12-15. Čtyřvstupové hradlo NAND v DTL

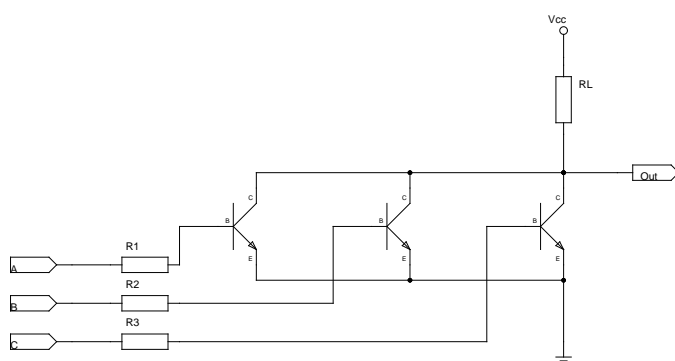


* gEDA image DTL_NAND:sch: 640×480 EPS, PNG

12.2.3. Resistor-Transistor Logic (RTL)

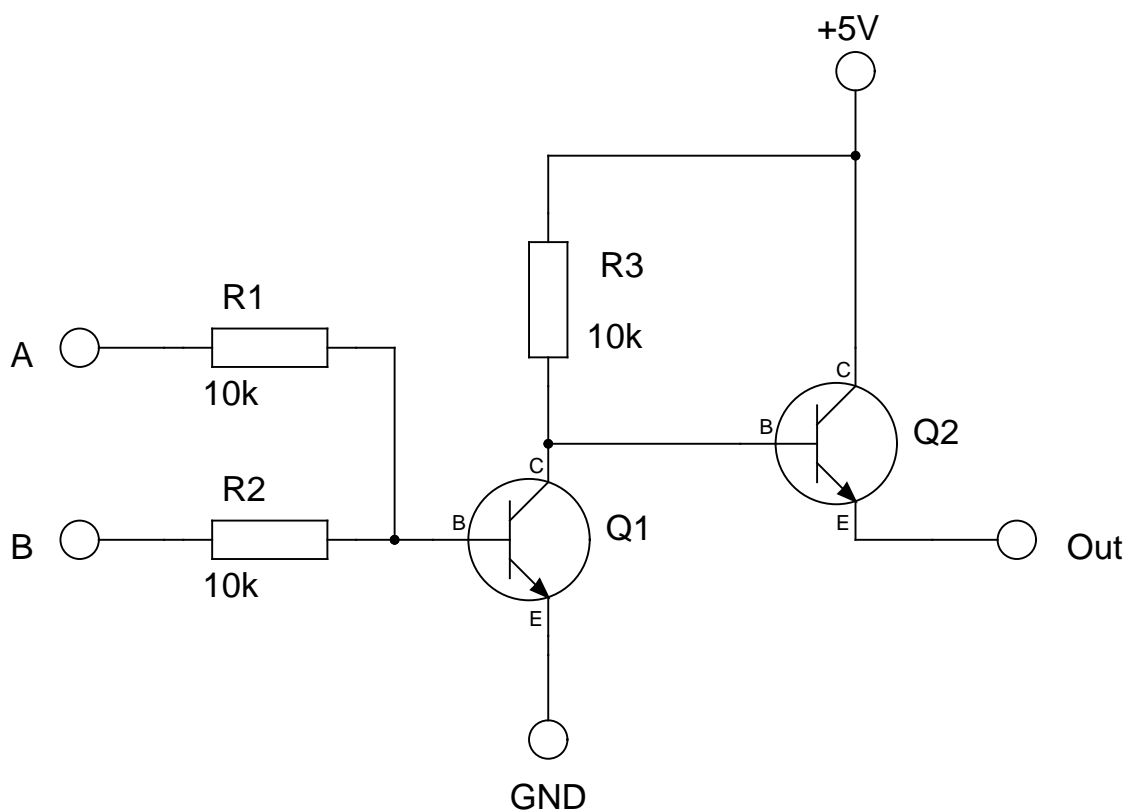
Odkazy:

- Resistor-transistor logic¹¹ na Wikipedii
-

Obrázek 12-16. Třívstupové hradlo NOR v technologii RTL

Hradla této konstrukce byla použita v integrovaných obvodech v AGC (Apollo Guidance Computer).

Další ukázka NOR hradla pochází z Instructables¹².

Obrázek 12-17.

* 320*240

12.2.4. Transistor Logic

* Následující informace pochází z http://www.6502.org/users/dieter/mt15a/mt15a_3.htm.

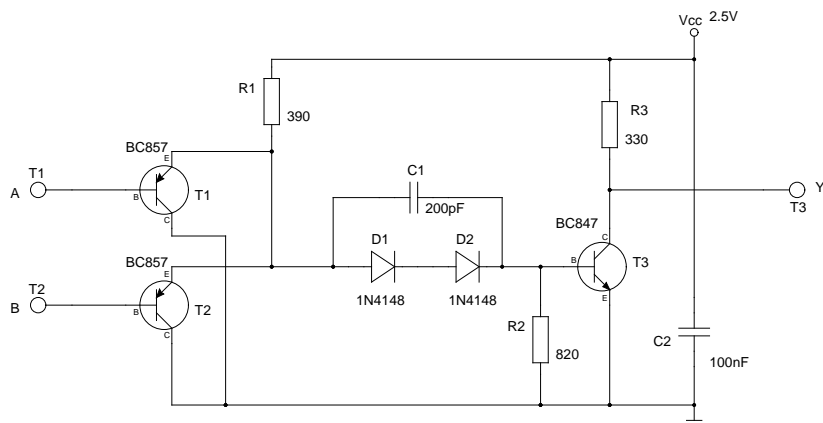
Zatím, co DTL logika, následovaná RTL logikou byla implementována v integrovaných obvodech a vedla ke konstrukci TTL a ECL, o realizaci logických obvodů z diskretních součástek již tolik informací není. V této

části shraňují schémata a implementace logických hradel realizovaných pomocí tranzistorů a dalších diskretních součástek.

* K realizaci logiky pomocí tranzistorů jsem zatím nenašel mnoho informací. Nejzajímavější co jsem objevil je reálně použitá konstrukce hradla uvedená na obrázku. Obvod byl použit v počítači s hodinovým kmitočtem 0.5 MHz.

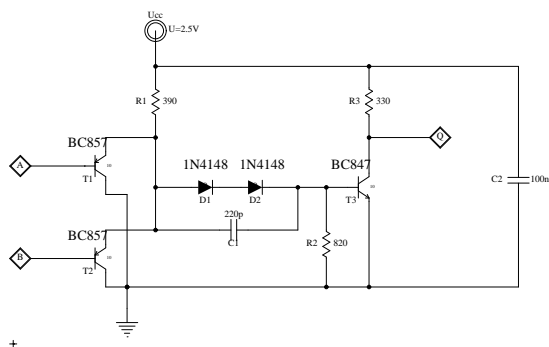
Následující hradlo NAND pochází z konstrukce počítače MT15.

Obrázek 12-18. Tranzistorové hradlo NAND počítače MT15



* gEDA image schematic:sch: 640×480 PNG, EPS scale=40

Obrázek 12-19. Tranzistorové hradlo NAND (v Electric)



Toto hradlo lze snadno upravit na zapojení s otevřeným kolektorem. Pouze se vypustí odpor R3. Rovněž lze snadno přidáním dalších tranzistorů na vstup pod T1 a T2 rozšířit počet vstupů. V Dietrově konstrukci¹³ se vyskytují 4 a 8 vstupové hradla NAND s otevřeným kolektorem. Použití otevřeného kolektoru pak umožňuje spojovat výstupy hradel v montážním součtu. Dosáhneme tak nižšího průměrného počtu tranzistorů na logické hradlo v zapojení, protože koncový tranzistor T3 se tak stane součástí hradla OR.

Threshold napětí je okolo 1.3V, takže lze toto hradlo budít bez problémů integrovanými obvody 74HCT/ACT.

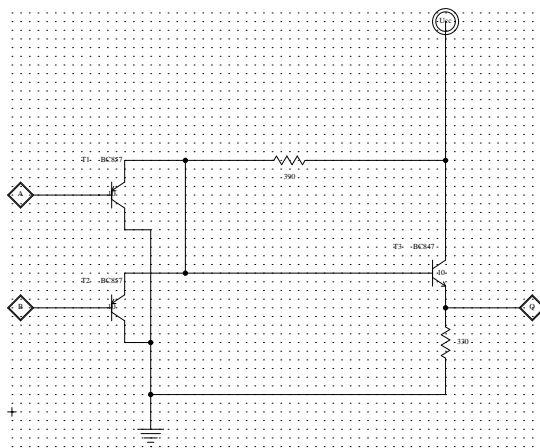
Napájecí napětí by mělo být v rozsahu od 2.4 do 2.6V. U každého hradla musí být na napájení kondenzátor C2.

12.2.5. CTL (Complementary Transistor Logic)

* *Attributy:* id="CTL"

Podstatou CTL je použití jak NPN tak PNP tranzistorů. Vazba mezi nimi, jak je vidět na schématu hradla NAND, je přímá.

Obrázek 12-20. CTL NAND v Electric



12.2.6. TTL

Odkazy:

- TTL NAND and AND gates¹⁴

12.2.7. To be done:ECL (SECL)

12.2.8. To be done:DECL (Differetntial ECL)

12.2.9. Technologie CMOS

Odkazy:

- CMOS¹⁵ anglický článek na Wikipedii
- CMOS¹⁶ český článek na Wikipedii

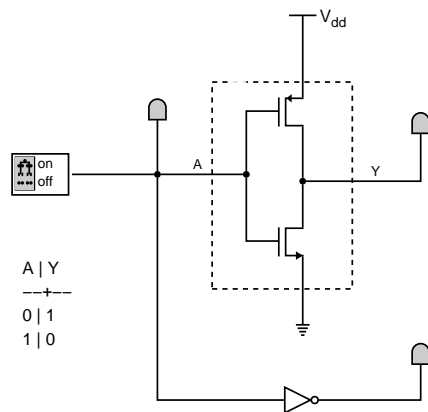
12.2.9.1. Invertor

Odkazy:

- CMOS¹⁷ na Wikipedii .cz
- CMOS¹⁸ na Wikipedii

Invertor je základní logické hradlo známé také pod názvem NOT. Jeho funkce je poskytovat na výstupu logickou hodnotu jenž je negací logické hodnoty na vstupu.

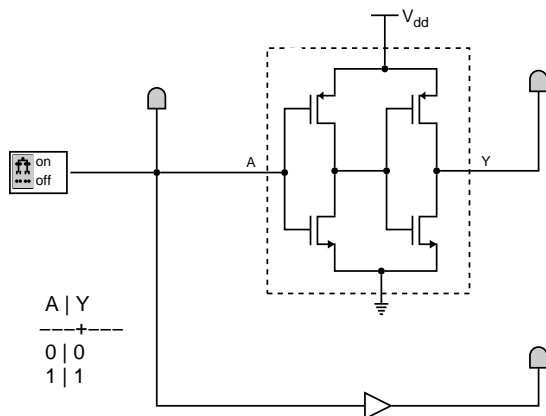
Obrázek 12-21. Testovací schéma CMOS invertoru v TkGate



12.2.9.2. Buffer

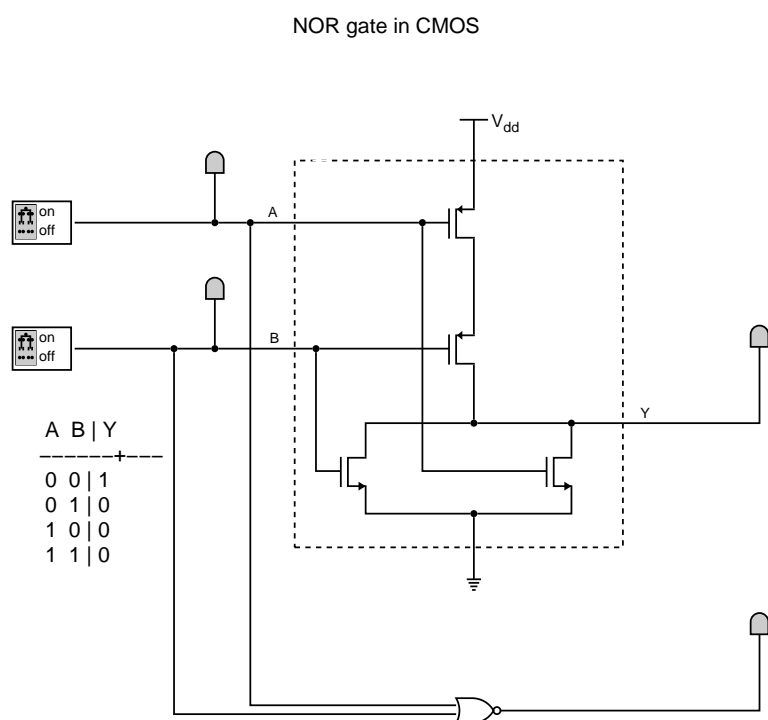
Obrázek 12-22. Testovací schéma CMOS bufferu v TkGate

CMOS Buffer. Also known as YES gate.

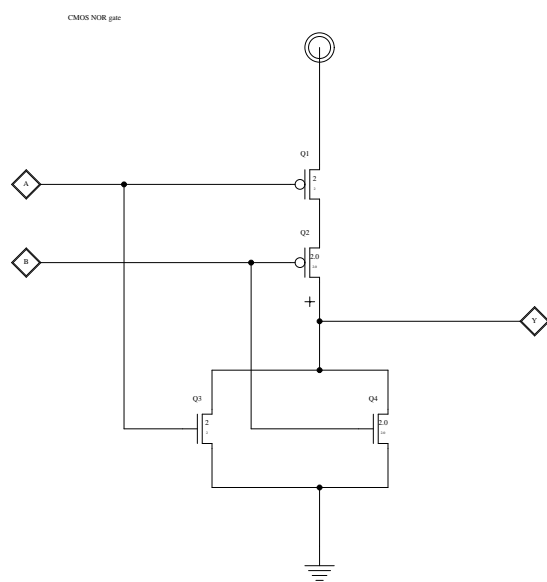


12.2.9.3. Hradlo NOR

Obrázek 12-23. Testovací schéma CMOS hradla NOR v TkGate

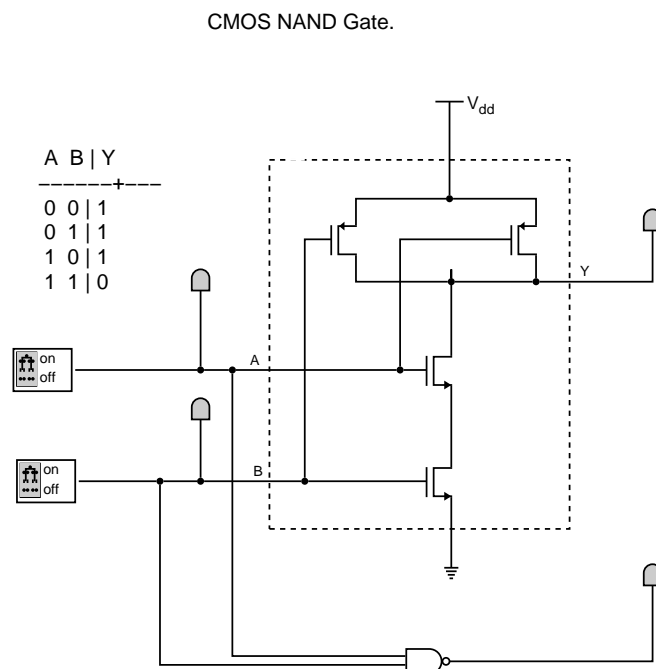


Obrázek 12-24. Testovací schéma CMOS hradla NOR v Electric

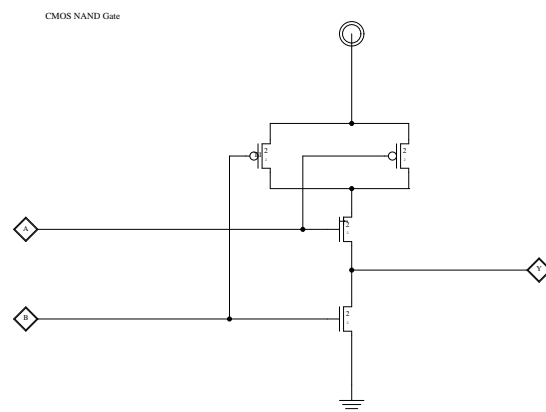


12.2.9.4. Hradlo NAND

Obrázek 12-25. Testovací schéma CMOS hradla NAND v TkGate



Obrázek 12-26. Testovací schéma CMOS hradla NAND v Electric



12.2.10. Mechanická hradla

Odkazy:

- Z1 - Calculation with Metal Sheets¹⁹
-

Poznámky

1. http://en.wikipedia.org/wiki/Logic_gate
2. <http://en.wikipedia.org/wiki/C-element>
3. http://en.wikipedia.org/wiki/Asymmetric_C-element
4. <http://www.flatspike.com/projects/electronics/single-transistor-logic-inverter/>
5. http://en.wikipedia.org/wiki/Diode_logic
6. <http://www.8bitu.cz/clanek/cislicova-technika-10-diodova-logika/>
7. <http://www.8bitu.cz/clanek/cislicova-technika-11-diodova-logika-2/>
8. http://www.copsu.cz/mikrop/didakticka_pomucka/cislicova_technika/zakl_log_cleny/obecne/obecne.html
9. <http://mycircuits.blogspot.com/2007/06/diode-transistor-logic.html>
10. <http://people.cs.ubc.ca/~hilpert/eec/misc/CasioAL1000Tech/electronics.html>
11. http://en.wikipedia.org/wiki/Resistor-transistor_logic
12. <http://www.instructables.com/id/Create-a-NOR-Gate/>
13. http://www.6502.org/users/dieter/mt15/mt15_sch.htm
14. http://www.allaboutcircuits.com/vol_4/chpt_3/5.html
15. <http://en.wikipedia.org/wiki/CMOS>
16. <http://cs.wikipedia.org/wiki/CMOS>
17. <http://cs.wikipedia.org/wiki/CMOS>
18. <http://en.wikipedia.org/wiki/CMOS>
19. http://user.cs.tu-berlin.de/~zuse/Konrad_Zuse/en/z1-metal-sheets.html

Kapitola 13. Kombinační obvody

* *WORKING: Editovat.*

13.1. Booleova Algebra

Odkazy:

- Číslicové počítače. Jan Blatný, Karel Křišťoufek, Zdeněk Pokorný, Ján Kolenička. SNTL 1982. Strana 62.
- Booleova algebra¹ na Wikipedii
-
-

Booleova algebra je množina B o alespoň dvou prvcích (0 a 1), nad níž jsou definovány dvě binární (\wedge a \vee) a jedna unární operace (\neg), splňující následující axiomy. Nejdříve si ale povíme něco více o použitých pojmech.

Je zvykem označovat prvky množiny B znaky

- První prvek: 0 , O , L
- Druhý prvek: 1 , I , H

Označení číslicemi 0 a 1 je obvyklé v matematice a výpočetní technice. Označení písmeny L a H je obvyklé v číslicové technice. Použití písmene O místo 0 a písmene I místo 1 je z historických důvodů kdy se pro tyto číslice častou používali písmena abecedy jim podobná.

Dvě binární operace nazývané jako sčítání a násobení jsou pojmenovávány a značeny jako

- logický součet, \vee , \wedge , $+$, OR
- logický součin, \wedge , \cdot , \cdot , \cdot , AND

Unární operace je nazývána negace a označována znaky \neg , $-$, $/$, $!$, NOT, nebo tak, že se nad výrazem udělá vodorovná čára přes celý výraz.

Axiomy

1. výsledkem všech tří operací \vee , \wedge a \neg jsou prvky množiny B
 - 2.
 3. komutativita: $a \vee b = b \vee a$, $a \wedge b = b \wedge a$
 4. asociativita: $a + (b \cdot c) = (a + b) \cdot (a + c)$, $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
 - 2a:** \exists prvek 0 , pro který platí $a + 0 = a$
 - 2b:** \exists prvek 1 , pro který platí $a \cdot 1 = a$
 - 4a:** $a + (b \cdot c) = (a + b) \cdot (a + c)$
 - 4b:** $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
 - 5:** Jsou-li prvky $0, 1$ z axiomu 2 jediné, pak pro každý prvek a existuje prvek $\neg a$ $\in B$, který splňuje rovnice $a \cdot \neg a = 0$, $a + \neg a = 1$
 - 6:** Množina B obsahuje alespoň dva navzájem různé prvky a, b .
- REMOVE: **1a:** $a + b \in B$
- REMOVE: **1b:** $a \cdot b \in B$
- REMOVE: **3a:** $a + b = b + a$
- REMOVE: **3b:** $a \cdot b = b \cdot a$

Věty:

- 1a:** Prvek 0 v axiomu 2a je jediný.
- 1b:** Prvek 1 v axiomu 2b je jediný.

2a: $a + a = a$

2b: $a \cdot a = a$

3a: $a + 1 = 1$

3b: $a \cdot 0 = 0$

4a: $a + a \cdot b = a$

4b: $a \cdot (a + b) = a$

5: $\neg a$ je určen jednoznačně

6: $\neg(\neg a) = a$

7a: $\neg(a + b) = \neg a \cdot \neg b$

7b: $\neg(a \cdot b) = \neg a + \neg b$

8a: $(a + b) + c = a + (b + c)$

8b: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

9a: $a + \neg a \cdot b = a + b$

9b: $a \cdot (\neg a + b) = a \cdot b$

10: $(a + b) \cdot (\neg a + c) = a \cdot c + \neg a \cdot b$

11a: $\neg(a \cdot c + b \cdot \neg c) = \neg a \cdot c + \neg b \cdot \neg c$

11b: $\neg((a + c) \cdot (b + \neg c)) = (\neg a + c) \cdot (\neg b + \neg c)$

* *Elementy:* exist \exists , forall \forall , isin \in , ni \notin , notin \notin , sdot \cdot

* *Chyba je v TeX souboru v řádce:* `{\boldsymbol{1a:\endSeq{}}\endNode{}} a + b \Character{8714} B\endSeq{}}\endNode{}}\endPar{}}\endDisplayGroup{}}\endNode{}}\Node%`

* *znak isin je podle ISOTECH² unicode 02208_{HEX}. Popis znaku: "/in R: =set membership"*

* *element &element; nefunguje*

Tabulka 13-1. Pravdivostní tabulka některých binárních operací

a	b	$a \wedge b$	$a \vee b$	$\neg a$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Tabulka 13-2. Pravdivostní tabulka některých binárních operací

a	b	$a \wedge b$	$a \vee b$	$\neg a$
O	O	O	O	I
O	I	O	I	I
I	O	O	I	O
I	I	I	I	O

13.2. Sčítání

*

Poloviční sčítačka je obvod se dvěma vstupy a dvěma výstupy. Sčítá dvě jedobitová čísla. Její funkční tabulka vypadá následovně.

Tabulka 13-3. Funkční tabulka poloviční sčítačky

A	B	Cout	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Zde jsou rovnice, popisující chování poloviční sčítačky.

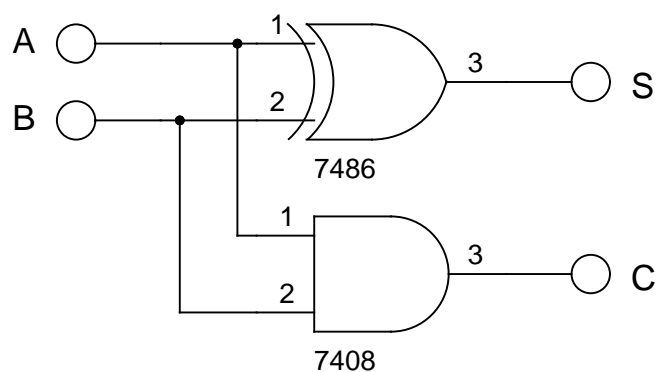
$$S = A \text{ xor } B = (A \wedge \neg B) \vee (\neg A \wedge B)$$

$$C = A \wedge B$$

Jak tabulka napovídá a je vidět v rovnicích, můžeme poloviční sčítačku realizovat s použitím dvou hradel. Hradlo XOR vytváří součet S vstupů A a B, a hradlo AND pak generuje přenos do vyššího řádu.

Ukázka realizace poloviční sčítačky pomocí základních logických hradel.

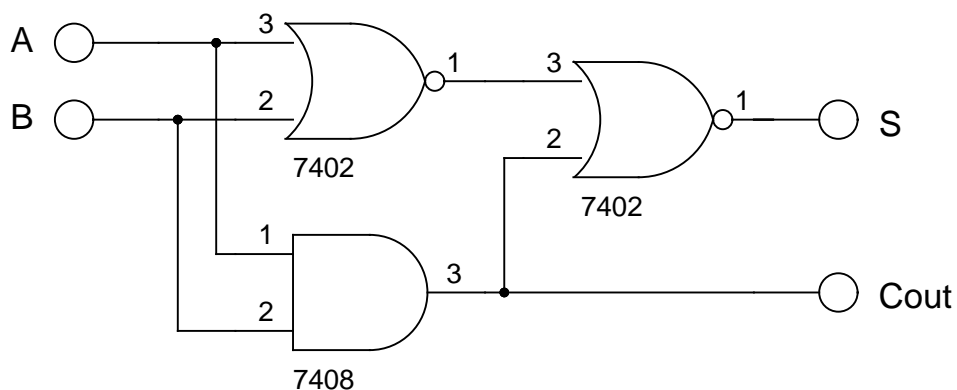
Obrázek 13-1. Poloviční sčítačka z hradla XOR a AND



* Image size 320*240 PNG, EPS.

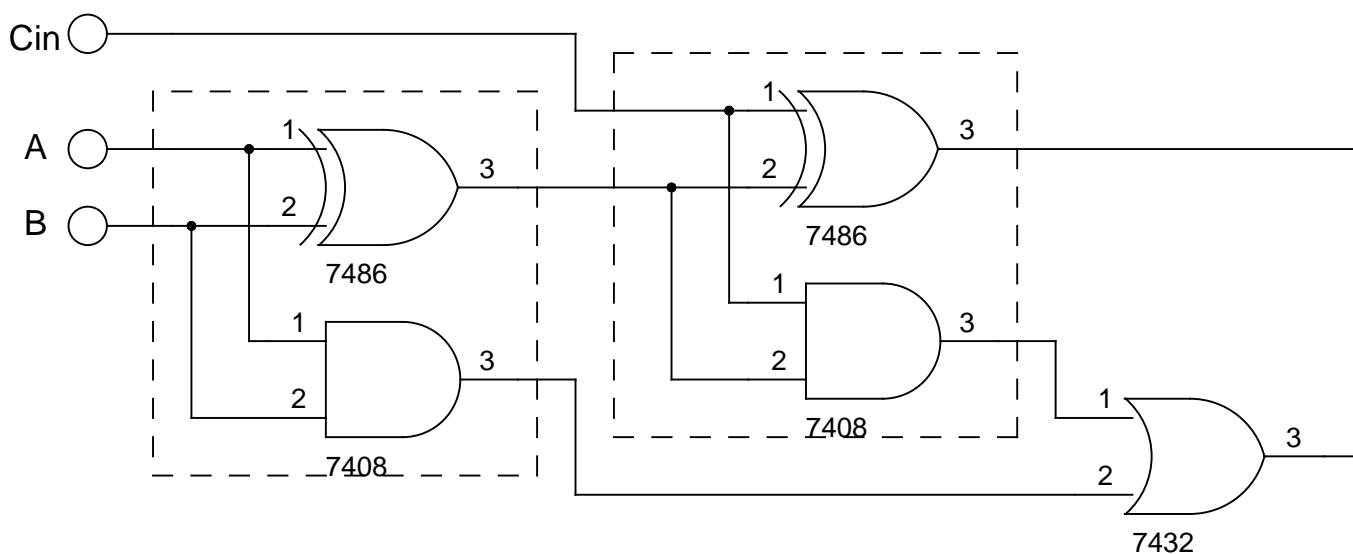
Tomuto sčítacímu obvodu se říká poloviční, protože pokud potřebujeme celou jednobitovou sčítačku, musíme použít kombinaci dvou uvedených obvodů. Tedy každý tvoří „polovinu“ celé sčítačky.

Obrázek 13-2. Alternativní zapojení poloviční sčítačky



Nyní si ukážeme zapojení celé sčítačky.

Obrázek 13-3. Zapojení úplné sčítačky



Poznámky

1. http://cs.wikipedia.org/wiki/Booleova_algebra
2. <http://www.w3.org/2003/entities/iso8879doc/isotech.html>

Kapitola 14. Sekvenční obvody

Sekvenční obvody jsou takové obvody, kde stav výstupů závisí nejen na vstupních hodnotách, ale i na předchozím stavu celého obvodu. Sekvenční obvody totiž mají paměť.

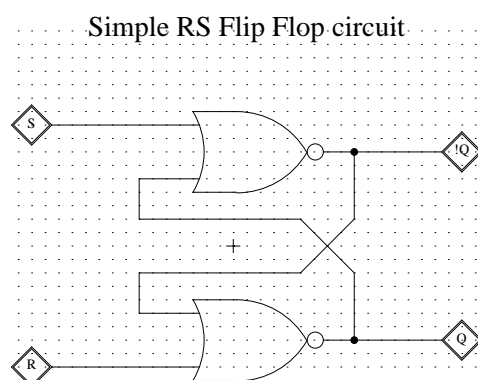
14.1. Klopný obvod RS

Odkazy:

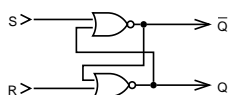
- The Basic RS NOR Latch¹
- The Basic RS NAND Latch²
- Basic flip-flop³
- A Transistor RS Flip Flop Tutorial⁴
- Transistor Pushpul RS Flipflop⁵
- 3. - BISTABLE TRIGGER CIRCUITS⁶
- Flip-Flops⁷
-
-

Klopný obvod je nejjednodušším sekvenčním obvodem a jeho princip je základem funkčnosti celé řady klopných obvodů (D, T, JK, ...). Sestává ze dvou hradel typu NOR

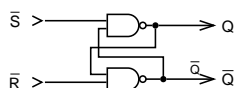
Obrázek 14-1. Schéma klopného obvodu RS v Electric



Obrázek 14-2. Schéma klopného obvodu RS v TkGate



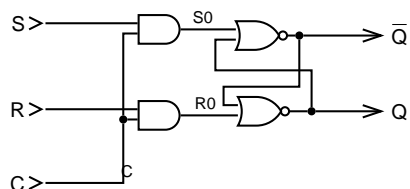
Místo hradel NOR můžeme použít hradla NAND

Obrázek 14-3. Schéma klopného obvodu RS z hradel NAND

Existuje velké množství variant klopného obvodu který jsme si právě ukázali. Některé z nich mají i vlastní název. Dále se pokusím některé z nich ukázat a upozornit na základní odlišnosti a použití těchto obvodů.

14.1.1. Klopný obvod RS s hradlovanými vstupy

Tato úprava klopného obvodu připouští změnu jen v případě že vstupní signál C má hodnotu "1".

Obrázek 14-4. Hradlovaný klopný obvod RS z hradel NOR

Tento obvod použijeme například při realizaci paměťového registru. Vstup C v takovém případě slouží pro výběr registr do kterého se bude zapisovat informace.

Poznámky

1. http://www.play-hookey.com/digital/rs_nor_latch.html
2. http://www.play-hookey.com/digital/rs_nand_latch.html
3. http://www.tpub.com/content/neets/14181/css/14181_119.htm
4. http://www.hobbyprojects.com/flip_flop/a_transistor_RS_flip_Flop.html
5. http://www.microclub.ch/~jmasur/electronics/Transistor_Pushpull_RS_Flipflop.png
6. http://www.premiumorange.com/daniel.robert9/anglais/Digit/Digit_4TS2.html
7. <http://www.facstaff.bucknell.edu/mastascu/elessonshtml/Logic/Logic4.html>

Kapitola 15. Dual Rail logika

Klasická číslicová logika používá k prezentaci logických hodnot jednoho vodiče. Ten, většinou napětíovou úrovní, určuje logickou "0" či "1". Dual Rail logika má však pro každou logickou hodnotu samostatný vodič.

Tabulka 15-1. Logické hodnoty na dual rail

0	(0,1)
1	(1,0)
quiet	(0,0)
alarm	(1,1)

Kapitola 16. Přehled některých logických IO

Odkazy a zdroje:

- Přehled obvodů řady TTL 7400. Díl I. 7400 až 7499
- Přehled obvodů řady TTL 7400. Díl II. 74100 až 74199
- Přehled obvodů řady CMOS 4000. Díl I. 4000 ... 4099
- Přehled obvodů řady CMOS 4000. Díl II. 41xx, 43xx, 45xx, 40xxx
- Chip Directory¹
- Chip Directory (mirror)²
- Original Frank's The Giant Internet IC Masturbator (GIICM)³
- Mirror⁴ of Frank's GIICM
- GIICM⁵ verze Jana Řeháka
- 7400 Series Description⁶
- Frank's Hospital Workshop⁷
- List of 7400 series integrated circuits⁸ na Wikipedii
- List of 4000 series integrated circuits⁹
-

Tato kapitola je přehledem vybraných logických integrovaných obvodů.

Následující tabulka je přehledem logických IO podle značení 74 které zavedla firma Texas Instruments. V prvním sloupci je značení podle firmy TI, v druhém pak kategorie do které obvod patří. Ve třetím sloupci je zkrácený popis obvodu s použitím řady zkratk. V předposledním sloupci je CD kód CMOS varianty obvodu jenž zavedla firma RCA. Poslední sloupec pak obsahuje případné odkazy na jiné informační zdroje k danému obvodu.

Tabulka 16-1. Použité zkratky

OC	Obvod s výstupy s otevřeným kolektorem. Může být doplněno informací o maximálním napětí v případě obvodů jenž slouží jako budiče.
Schmitt	Obvod má vstupech Schmittovy klopné obvody.
EXP	Obvod má Expanded vstupy výstup. Tyto slouží pro propojení obvodů za účelem vytvoření komplikovanějších funkcí. Typicky FIXME : doplnit příklad.
GATED	Výstup obvodu je „hradlován“ zvláštním vstupem značným G

* Následující 5(4) sloupcová tabulka je pokus o tabulku 74xx obvodů s doplňkovými informacemi.

Following table is in “compressed” for because of a space.

OC	Open collector output
GATED	output is gated by G input
Schmitt	Schmitt Trigger input gate
EXP	gate has X,Y inputs for expander

* Následující tabulka je po novu generována z informací uvedených na jiných místech. Druhá část tabulky jsou původní ručně zadaná data která jsou postupně odstraňována.

Tabulka 16-2. Tabulka obvodů 74 podle značení

kód SN	kat.	popis	kód CD	links
7400	logic	4x 2-input NAND gate		CD ₁₀ GM ₁₀

kód SN	kat.	popis	kód CD	links
7401	logic	4x 2-input NAND gate OC		CD ₁₀ GM ₁₀
7402	logic	4x 2-input NOR gate	4001	CD ₁₀ GM ₁₀
7403	logic	4x 2-input NAND gate OC		CD ₁₀ GM ₁₀
7404	buffer	6x INVERTOR	4069	CD ₁₀ GM ₁₀
7405	buffer	6x INVERTOR OC		CD ₁₀ GM ₁₀
7406	buffer	6x NOT gate OC 30V		CD ₁₀ GM ₁₀
7407	buffer	6x buffer OC, 30V, 40mA		CD ₁₀ GM ₁₀
7408	logic	4x 2-input AND gate	4081	CD ₁₀ GM ₁₀
7409	logic	4x 2-input AND gate OC		CD ₁₀ GM ₁₀
7410	logic	3x 3-input NAND gate	4023	CD ₁₀ GM ₁₀
7411	logic	3x 3-input AND gate	4073	CD ₁₀ GM ₁₀
7412	logic	3x 3-input NAND gate OC		CD ₁₀ GM ₁₀
7413	logic	2x 4-input NAND gate Schmitt	4093	CD ₁₀ GM ₁₀
7414	buffer	6x NOT gate Schmitt	4584	CD ₁₀ GM ₁₀
7415	logic	3x 3-input AND gate OC		CD ₁₀ GM ₁₀
7416	buffer	6x NOT gate OC 15V		CD ₁₀ GM ₁₀
7417	buffer	6x NOT gate OC 15V		CD ₁₀ GM ₁₀
7418	logic	2x 4-input NAND gate Schmitt		CD ₁₀ GM ₁₀
7419	buffer	6x NOT gate Schmitt		CD ₁₀ GM ₁₀
7420	logic	2x 4-input NAND gate		CD ₁₀ GM ₁₀
7424	logic	4x 2-input NAND gate Schmitt	4093	CD ₁₀ GM ₁₀
7430	logic	1x 8-input NAND gate		CD ₁₀ GM ₁₀
7432	logic	4x 2-input OR gate	4xxx	CD ₁₀ GM ₁₀
7437	logic	4x 2-input NAND gate Power		CD ₁₀ GM ₁₀
7438	logic	4x 2-input NAND gate OC Power		CD ₁₀ GM ₁₀
7440	logic	2x 4-input NAND gate Power		CD ₁₀ GM ₁₀
7442	decoder	Dekodér BCD na 1 z 10		CD ₁₀ GM ₁₀
7450	logic	2x 4-input AND-OR-INVERT-EXPAND gate		CD ₁₀ GM ₁₀
7451	logic	2x 4-input AND-OR-INVERT gate		CD ₁₀ GM ₁₀
7453	logic	1x 8-input AND-OR-INVERT-EXPAND gate		CD ₁₀ GM ₁₀
7454	logic	1x 8-input AND-OR-INVERT gate		CD ₁₀ GM ₁₀
7470	flip-flop	J-K Flip-Flop s nulováním a nastavením		CD ₁₀ GM ₁₀
7471	flip-flop	J-K Flip-Flop s nulováním a nastavením		CD ₁₀ GM ₁₀
7472	flip-flop	J-K Flip-Flop s nulováním a nastavením		CD ₁₀ GM ₁₀
7473	flip-flop	2x J-K Flip-Flop s nulováním		CD ₁₀ GM ₁₀
7474	flip-flop	2x D Flip-Flop s nulováním a nastavením		CD ₁₀ GM ₁₀

kód SN	kat.	popis	kód CD	links
7475	flip-flop	4x D Flip-Flop		CD ₁₀ GM ₁₀
7476	flip-flop	2x J-K Flip-Flop s nulováním a nastavením		CD ₁₀ GM ₁₀
7478	flip-flop	2x J-K Flip-Flop s nulováním a nastavením		CD ₁₀ GM ₁₀
7486	logic	4x 2-input XOR gate	4030,4070	CD ₁₀ GM ₁₀
7490	counter	desítkový čítač s nulováním a nastavením		CD ₁₀ GM ₁₀
7493	counter	čtyřbitový čítač s nulováním		CD ₁₀ GM ₁₀
7496	shift	pětibitový posuvný registr		CD ₁₀ GM ₁₀
74122	buffer	monostable ff		
74123	buffer	2x monostable ff		
74123	osc	VCO		
74125	buffer	4x buffer 3 state		
74126	buffer	4x buffer 3 state		
74136	logic	4x 2-input XOR gate OC		
74141	decoder	Dekodér BCD na 1 z 10 HV		
74147	logic	priority encoder		
74150	decoder	16-Multiplexer		
74151	decoder	8-Multiplexer		
74154	decoder	16-Demultiplexer		
74164	shift	8-mi bitový posuvný registr s paralelním výstupem		
74165	shift	8-mi bitový posuvný registr PISO		
74192	counter	dekadický synchronní vratný čítač s nulováním a nastavením		
74193	counter	čtyřbitový synchronní vratný čítač s nulováním a nastavením		
74266	logic	4x 2-input XNOR gate OC		
74324	osc	VCO		
74325	osc	VCO		
74326	osc	VCO		
74327	osc	VCO		
74386	logic	4x 2-input XOR gate		
74543	latch	8-mi bitový, třístavový, obousměrný latch		GM ₁₀
74544	latch	8-mi bitový, invertující, třístavový, obousměrný latch		GM ₁₀
74573	latch	8-mi bitový, třístavový latch		GM ₁₀

kód SN	kat.	popis	kód CD	links
74595	shift	8-mi bitový posuvný registr SIPO s 3-stavovým paralelním výstupem		GM ₁₀
74610	logic	mmu		GM ₁₀
74611	logic	memory management unit		GM ₁₀
74612	logic	memory management unit		GM ₁₀
74613	logic	memory management unit		GM ₁₀
74624	osc	VCO		GM ₁₀
74625	osc	VCO		GM ₁₀
74626	osc	VCO		GM ₁₀
74627	osc	VCO		GM ₁₀
74628	osc	VCO		GM ₁₀
74629	osc	VCO		GM ₁₀
747266	logic	4x 2-input XNOR gate	4077	
<i>Následuje původní, ručně zadávaná část tabulky.</i>				
7421	logic	2x 4-input AND gate		CD ₁₀ , GM ₁₀
7422	logic	2x 4-input NAND gate OC		CD ₁₀ , GM ₁₀
7423	logic	2x 4-input NOR gate GATED, EXPANDED		CD ₁₀
7425	logic	2x 4-input NOR gate GATED		CD ₁₀ , GM ₁₀
7426	logic	4x 2-input NAND gate OC		CD ₁₀ , GM ₁₀
7427	logic	3x 3-input NOR gate		CD ₁₀ , GM ₁₀
7428	logic	4x 2-input NOR gate		CD ₁₀ , GM ₁₀
7429	<i>no circuit uses this number</i>			
7431	logic	2x YES, 2x NOT, 2x NAND gate DELAY		CD ₁₀ , GM ₁₀
7432	logic	4x 2-input OR		CD ₁₀ , GM ₁₀
7433				CD ₁₀ , GM ₁₀
7434				CD ₁₀ , GM ₁₀
7435				CD ₁₀ , GM ₁₀
74543	latch	8-bit 3-state nonverting 2-way register-transceiver(!LEBA, !GBA, !CEBA, !CEAB, !LEAB, !GAB)		CD ₁₀ , GM ₁₀
74544	latch	8-bit 3-state iverting 2-way register-transceiver(!LEBA, !GBA, !CEBA, !CEAB, !LEAB, !GAB)		CD ₁₀ , GM ₁₀
				CD ₁₀ , GM ₁₀

kód SN	kat.	popis	kód CD	links
7450729		2x flip-flop D	n/a	CD ₁₀
-----><-----				
Následuje původní verze tabulky zmenším počtem sloupců. Jednotlivé řádky jsou postupně konvertovány.				
IC	popis			
139	dual 2-to-4 demultiplexer			
244	dual 4-bit 3-state noninverting buffer/line driver (!OE, !OE))			
245	8-bit 3-state noninverting bus transceiver (DIR, !EN)			
273	8-bit D flip-flop with reset (!RST, CLK)			
373	8-bit 3-state transparent latch (!OE, LE)			
374	8-bit 3-state D flip-flop (!OE, CLK)			
377	8-bit D flip-flop with clock enable (!CLKEN, CLK)			
540	8-bit 3-state inverting buffer/line driver (!OE1, !OE2)			
541	8-bit 3-state noninverting buffer/line driver (!OE1, !OE2)			
573	8-bit 3-state transparent latch (!OE, LE)			
574	8-bit 3-state D flip-flop, latch (CLK, !OE)			
575	8-bit 3-state D flip-flop with reset (!RST, !OE, CLK)			
576	8-bit 3-state inverting D flip-flop (!OE, CLK)			
577	8-bit 3-state inverting D flip-flop with reset (!RST, !OE, CLK)			
580	8-bit 3-state inverting transparent latch (!OE, LE)			
646	?			

Tabulka 16-3. Tabulka obvodů 4000 podle značení

kód CD	kat.	popis	kód SN	links
4000	logic	2x 3-input NOR gate + one Invertor		
4001	logic	4x 2-input NOR gate		
4002	logic	2x 4-input NOR gate		
4011	logic	4x 2-input NAND gate		
4013	flip-flop	2x klopný obvod D		
4028	decoder	Dekodér 1 z 10		
4029	counter	čtyřbitový/dekadický synchronní vratný čítač s nastavením		
4049	buffer	6x NOT, CMOS to TTL		
4050	buffer	6x Buffer, CMOS to TTL		
4077	logic	4x 2-input XNOR gate		

kód CD	kat.	popis	kód SN	links
4093	logic	4x 2-input NAND gate Schmitt		

* Následující sekce popisují jednotlivé kategorie obvodů. V sekcích pak mohou být další informace a popisy konkrétních obvodů dané kategorie. Rovněž je možno obvody v sekcích dále dělit. U popisu každého obvodu se nachází <?db ?> věta. Informace z této věty jsou pak využity při vytváření různých souhrnných a přehledových tabulek.

Čipy k prozkoumání:

- CD4060 / 74HC4060¹⁰ oscilátor a 14 stupňový asynchronní čítač
-

16.1. Řady/rodiny logických obvodů

Odkazy:

- STANDARD LOGIC¹¹
- Logic Family Choices¹²
- About TTL Logic families¹³
-

Tabulka 16-4. Přehled řad/rodin logických obvodů

rodina	název	popis
TTL		5V
S	Shottky Logic	5V
AS		5V
L		5V
LS	Low Power Schottky Logic	5V
ALS		5V
CDS Default	Cadence Design Systems CMOS Default	5V
C	CMOS	5V
AC	Advanced CMOS Logic, inputs CMOS compatible	5V
ACT	Advanced CMOS Logic, inputs TTL compatible	5V
ACTQ	Advanced CMOS Logic	5V
AHC	Advanced High Speed CMOS Logic	5V, 3.3V
AHCT	Advanced High Speed CMOS Logic	5V, 3.3V
HC	High Speed CMOS Logic	5V
HCT	High Speed CMOS Logic	5V
VHC		5V
ABT	Advanced BiCMOS Technology	5V
ALVC	Advanced Low Voltage CMOS Technology	3.3V
ALVCH	Advanced Low Voltage CMOS Technology	3.3V

rodina	název	popis
ALVT	Advanced Low Voltage CMOS Technology	3.3V
HSTL	High Speed Transceiver Logic	3.3V
LV/LVC	Low Voltage CMOS Logic	3.3V
ALB	Advanced Low Voltage BiCMOS Technology	2.5V

Tabulka 16-5. Vybrané parametry vybraných rodin

rodina	t_{pd} [ns]	f_{clk} [MHz]	V_{supply} [V]
CMOS @10V	30	5	3-18
CMOS @5V	50	2	3-18
AC	3	125	2-6
ACT	3	125	4.6-5.5
HC	9	30	2-6
HCT	9	30	4.6-5.5
AS	2	105	4.5-5.5
F	3.5	100	4.75-5.25
ALS	4	34	4.5-5.5
LS	10	25	4.75-5.25

4000 Series CMOS Technolog

16.1.1. Propojení obvodů s odlišným napětím

Odkazy:

- Bidirectional I²C Level Shifter — Elektor Electronics, rok 2005, číslo 7-8, strana 64¹⁴
- Philips Semiconductors Application Note AN97055¹⁵
-
-
-

16.2. Zesilovače, převodníky úrovní

Odkazy:

- 16.3.1
- 16.3.2
-

Tyto obvody slouží jako rozhraní k „jiné“ elektronice než je vlastní řada obvodu. Jedná se například o budiče displejů o vyšším napětí (například 7407), případně o budiče pro připojení k jiné technologické řadě obvodů (např. **FIXME:**).

16.2.1. 74HC126, 74HC125 Quad buffer/line driver 3-state

* *Attributy:*

Tento obvod obsahuje čtveřici hradel-zesilovačů s třístavovým výstupem. Každé hradlo má vlastní řízení třístavového výstupu. Obvody 125 a 126 sse liší jen polaritou signálů přepínajících výstupy hradel do třetího stavu.

16.2.2. 4049 CMOS to TTL inverter

* *Attributv: id="cd4049"*

Obvod je možno použít jako invertor který snese větší zátěž (nutno konzulovat datasheet) nebo jako oddělovací invertor pro převod signálu z CMOS do TTL. Pro zvýšení zátěže je možno inventory řadit paralelně.

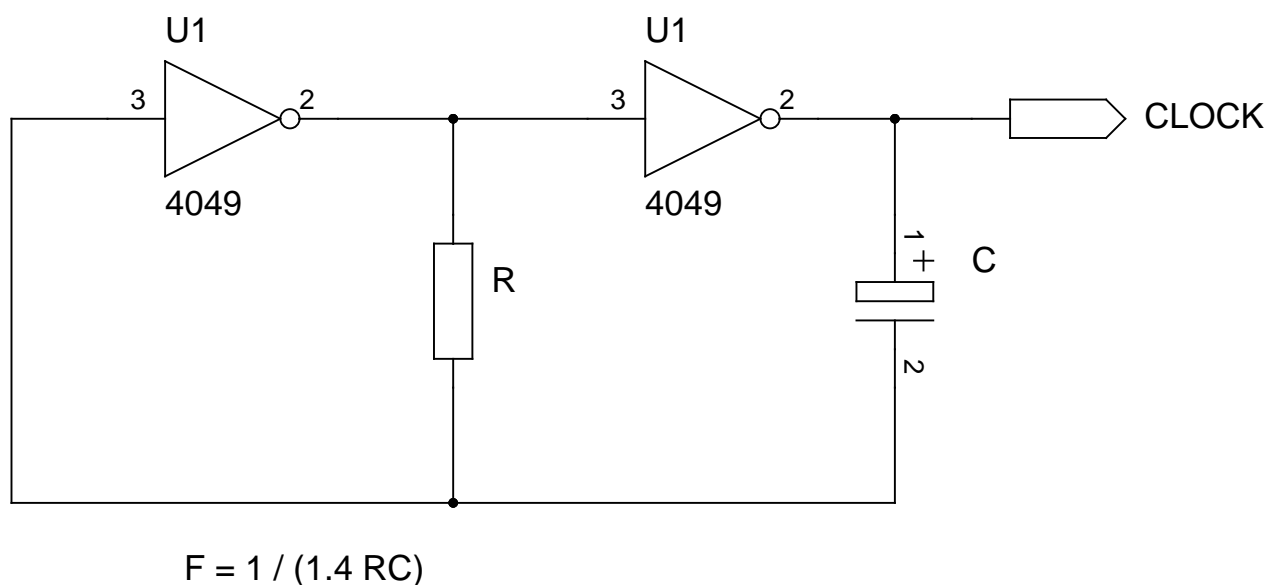
Obrázek 16-1. 4049 invertující oddělovač CMOS-TTL

		+	---	\	/	---	+	
Vcc	-	1		16		-	nc	
Y1	-	2		15		-	Y6	
A1	-	3		14		-	A6	
Y2	-	4		13		-	nc	
A2	-	5		12		-	Y5	
Y3	-	6		11		-	A5	
A3	-	7		10		-	Y4	
GND	-	8		9		-	A4	
				+	-----	+		

* Simple Pulse Generator circuit by CD4049¹⁶

* IC 4049 Clock Pulse Generator¹⁷

Obrázek 16-2. RC Oscilátor s CD4049



* *Image size 640*480 PNG, EPS.*

Uvedený oscilátor dává výstupní kmitočet podle vzorce.

$$F = 1 / (1.4 * R * C)$$

R cca 100K, C 0.01 až 10 μ F.

16.2.3. CD4050 CMOS to TTL buffer

* *Attributy: id="cd4050"*

Odkazy:

- ST7565 LCDs: Graphical LCDs!¹⁸
-
-
-

Obvod 4050 obsahuje šestici neinvertujících zesilovačů, konstruovaných jako převodníky napět'ových úrovní CMOS→TTL. Napájení obvodu musí být zapojené na stejné napětí jako jsou logické obvody připojené k výstupům tohoto obvodu. Obvod můžeme použít i k převodu signálů z 5V logiky na 3.3V logiku.

16.3. Logické obvody / hradla

Odkazy:

- 12.1
- Logic gate¹⁹ na Wikipedii

Jedním ze základních logických hradel a současně i prvním členem řady obvodů 7400 je obvod s označením 7400. Jedná se o čtveřici dvouvstupových hradel NAND.

Dalším obvodem je šestice inventořů v pouzdře 7404.

Řada z výše uvedených obvodů existuje v provedení s výstupem s otevřeným kolektorem. Jedná se například ale nejen o obvody 7401, 7405,

16.3.1. Zesilovač / buffer / hradlo YES

* *Attributy: id="ic.logic.yes"*

Odkazy:

- 12.1.1
- 16.2
- 16.3.2

Obvod 7407 obsahuje šestici zesilovačů s otevřeným kolektorem schopným spínat napětí do 30V.

16.3.2. Invertor NOT (7404, 05, 06, 14, 16, 17, 19)

* *Attributy: id="ic.logic.not"*

Odkazy:

- 12.1.2

Obvod 7404 obsahuje 6 invertorů.

Obvod 7405 obsahuje 6 invertorů s výstupy s otevřeným kolektorem.

Obvod 7414 obsahuje šestici invertorů se schmittovým klopným obvodem.

Obrázek 16-3. 7404, 7414 NOT

+---\ /---+			
A1	- 1	14	- V _{CC}
Y1	- 2	13	- A6
A2	- 3	12	- Y6
Y2	- 4	11	- A5
A3	- 5	10	- Y5
Y3	- 6	9	- A4
GND	- 7	8	- Y4
+-----+			

16.3.3. Hradla AND (7408, 7409, 7411, 7415)

* *Attributy: id="ic.logic.and"*

Odkazy:

- 12.1.3

Obrázek 16-4. 7408, 7409 AND

+---\ /---+			
A1	- 1	14	- V _{CC}
B1	- 2	13	- B4
Y1	- 3	12	- A4
A2	- 4	11	- Y4
B2	- 5	10	- B3
Y2	- 6	9	- A3
GND	- 7	8	- Y3
+-----+			

Trojice třívstupových hradel AND

Obrázek 16-5. 7411, 7415 3x 3input AND

+---\ /---+			
A1	- 1	14	- V _{CC}
B1	- 2	13	- C1
A2	- 3	12	- Y1
B2	- 4	11	- C3
C2	- 5	10	- B3
Y2	- 6	9	- A3
GND	- 7	8	- Y3
+-----+			

16.3.4. Hradla NAND (00, 01, 03, 10, 12, 13, 18, 20, 30, 37, 38, 40, 4011)

Odkazy:

- 12.1.5
-

* Obrázek zapojení vývodů obvodu v pouzdře DIL.

Obrázek 16-6. 7400, 7401, 7403, 7424 NAND

+---\ /---+			
A1 -	1	14	V _{CC}
B1 -	2	13	B4
Y1 -	3	12	A4
A2 -	4	11	Y4
B2 -	5	10	B3
Y2 -	6	9	A3
GND -	7	8	Y3
+-----+			

Trojice třívstupových hradel NAND

Obrázek 16-7. 7410, 7412 3x 3-input NAND

+---\ /---+			
A1 -	1	14	V _{CC}
B1 -	2	13	C1
A1 -	3	12	Y1
B2 -	4	11	C3
C2 -	5	10	B3
Y2 -	6	9	A3
GND -	7	8	Y3
+-----+			

7413 — Dvojice čtyřvstupových hradel NAND se Schmitovým klopným obvodem

7420 — Dvojice čtyřvstupových hradel NAND.

Obrázek 16-8. 7413, 7420 2x 4-input NAND

+---\ /---+			
A1 -	1	14	V _{CC}
B1 -	2	13	D2
nc -	3	12	C2
C1 -	4	11	nc
D1 -	5	10	B2
Y1 -	6	9	A2
GND -	7	8	Y2
+-----+			

Obrázek 16-9. 7430 8-input NAND

+---\ /---+			
A	-	1	14 - V _{CC}
B	-	2	13 - n _C
C	-	3	12 - H
D	-	4	11 - G
E	-	5	10 - n _C
F	-	6	9 - n _C
GND	-	7	8 - Y
+-----+			

Integrovaný obvod CD4011 obsahuje čtveřici dvouvstupových hradel NAND, stejně jako obvod CD4093. CD4093 jsou ovšem hradla se Schmittovými klopnými obvody. Zapojení pozder je stejné viz obrázek.

Obrázek 16-10. 4011,4093 — 4* 2-input NAND

+---\ /---+			
A1	-	1	14 - V _{CC}
B1	-	2	13 - A4
Y1	-	3	12 - B4
Y2	-	4	11 - Y4
B2	-	5	10 - Y3
A2	-	6	9 - B3
GND	-	7	8 - A3
+-----+			

16.3.5. Hradla OR (7432)

* *Attributy:* id="ic.logic.or"

16.3.6. Hradla NOR (7402, 4000, 4001, 4002, 4025, 4078)

* *Attributy:* id="ic.logic.nor"

Obrázek 16-11. 7402 NOR

+---\ /---+			
A1	-	1	14 - V _{CC}
B1	-	2	13 - B4
Y1	-	3	12 - A4
A2	-	4	11 - Y4
B2	-	5	10 - B3
Y2	-	6	9 - A3
GND	-	7	8 - Y3
+-----+			

Obvod 4000 obsahuje dvě nezávislá třívstupová hradla NOR a jeden invertor.

Obrázek 16-12. 4000 NOR

+---\ /---+			
NC	-	1	14 - V _{CC}
NC	-	2	13 - C2
A1	-	3	12 - B2
B1	-	4	11 - A2
C1	-	5	10 - Y2
Y1	-	6	9 - !X
GND	-	7	8 - X
+-----+			

Obvod 4001 obsahuje čtyři nezávislá dvouvstupová hradla NOR.

Obrázek 16-13. 4001 NOR

+---\ /---+			
A1	-	1	14 - V _{CC}
B1	-	2	13 - A4
Y1	-	3	12 - B4
Y2	-	4	11 - Y4
B2	-	5	10 - Y3
A2	-	6	9 - B3
GND	-	7	8 - A3
+-----+			

Obvod 4002 obsahuje dvě nezávislá čtyřvstupová hradla NOR.

Obrázek 16-14. 4001 NOR

+---\ /---+			
Y1	-	1	14 - V _{CC}
A1	-	2	13 - Y2
B1	-	3	12 - A2
C1	-	4	11 - B2
D1	-	5	10 - C2
NC	-	6	9 - D2
GND	-	7	8 - NC
+-----+			

16.3.7. Hradla XOR

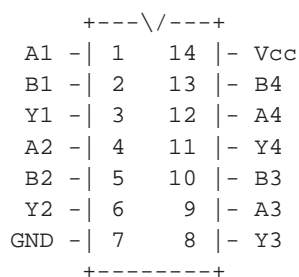
* *Attributy: id="ic.logic.xor"*

Odkazy:

- 12.1.7

Exklusivní součet. K dispozici jsou čtveřice dvouvstupových hradel v obvodu 7486 a hradla s otevřeným kolektorem na výstupu 74136.

Obrázek 16-15. 7486, 74136 XOR

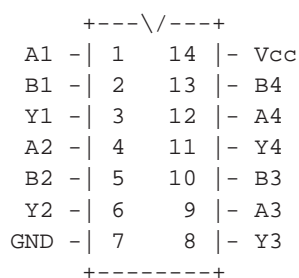


16.3.8. Hradla XNOR

* *Attributy: "ic.logic.xnor"*

Negace exkluzivního součtu. K dispozici jsou čtveřice dvouvstupových hradel v obvodu 4077.

Obrázek 16-16. 74266, 747266 XNOR

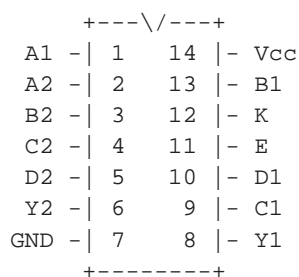


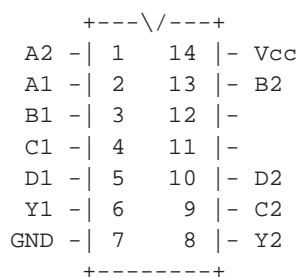
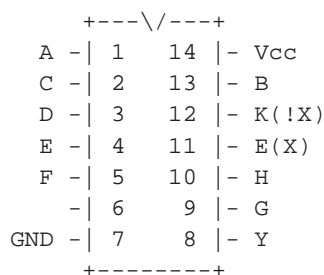
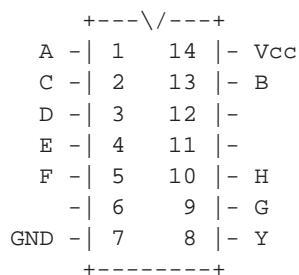
16.3.9. AND-OR-INVERT (50, 51, 53, 54)

*

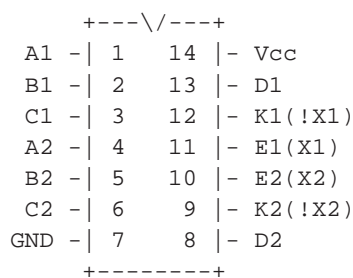
Dvojitý logický člen AND-OR-INVERT s jednou rozšířenou sekcí

Obrázek 16-17. 7450 2x AND-OR-INVERT-EXPAND



Obrázek 16-18. 7451 2x AND-OR-INVERT

Obrázek 16-19. 7453 1x AND-OR-INVERT-EXPAND

Obrázek 16-20. 7454 1x AND-OR-INVERT


Obvod 7460 je expander pro obvody 7450, 7453

Obrázek 16-21. 7460 2x AND-EXPANDER


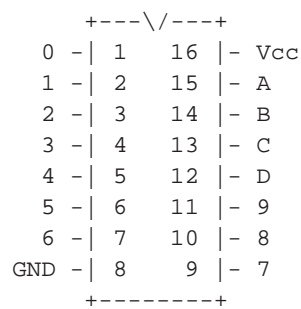
16.4. Dekodéry

*

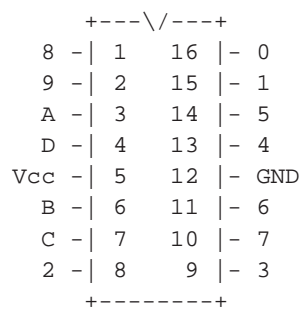
16.4.1. Dekodér BCD na 1 z 10 (42, 141, 4028)

* *Attributy: id="ic.decoder.bcd"*

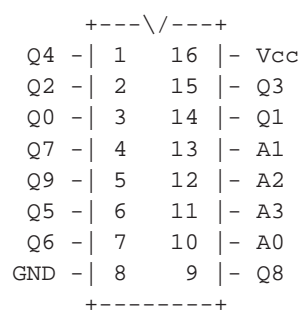
Obrázek 16-22. 7442 Dekodér BCD na 1 z 10



Obrázek 16-23. 74141 Dekodér BCD na 1 z 10 pro digitrony



Obrázek 16-24. 4028 Dekodér BCD na 1 z 10



16.4.2. Multiplexer

*

Obrázek 16-25. Šestnáctikanálový multiplexer 74150

		+-----+	+-----+	
7	-	1	+---+	24 V _{CC}
6	-	2		23 - 8
5	-	3		22 - 9
4	-	4		21 - 10
3	-	5		20 - 11
2	-	6		19 - 12
1	-	7		18 - 13
0	-	8		17 - 14
S	-	9		16 - 15
W	-	10		15 - A
D	-	11		14 - B
GND	-	12		13 - C
		+-----+		

Obrázek 16-26. Osmikanálový multiplexer 74151

		+---\ /---+	
D3	-	1	16 - V _{CC}
D2	-	2	15 - D4
D1	-	3	14 - D5
D0	-	4	13 - D6
Y	-	5	12 - D7
W	-	6	11 - A
S	-	7	10 - B
GND	-	8	9 - C
		+-----+	

16.4.3. Demultiplexer

*

Obrázek 16-27. Šestnáctikanálový demultiplexer 74154

		+-----+	+-----+	
0	-	1	+---+	24 V _{CC}
1	-	2		23 - A
2	-	3		22 - B
3	-	4		21 - C
4	-	5		20 - D
5	-	6		19 - G2
6	-	7		18 - G1
7	-	8		17 - 15
8	-	9		16 - 14
9	-	10		15 - 13
10	-	11		14 - 12
GND	-	12		13 - 11
		+-----+		

16.5. Klopné obvody

*

V této části popisují různé druhy jednoduchých klopných obvodů.

16.5.1. Klopné obvody RS

* **Attributy:** *id="ic.ff.rs"*

16.5.2. Klopné obvody D (75)

```

GND -| 7      8 | - S2
      +-----+

```

Tabulka 16-6. Pravdivostní tabulka klopného obvodu D v IO CD4013

CL	D	R	S	Q	!Q	komentář
↑	L	L	L	L	H	
↑	H	L	L	H	L	
↓	X	L	L	Q	!Q	
X	X	H	L	L	H	reset — mazání
X	X	L	H	H	L	set — nastavení
X	X	H	H	H	H	zakázaná kombinace

16.5.3. Klopné obvody JK (70,71,72,73,76,78, 4013)

*

Obvod 7470 obsahuje jeden klopný obvod typu J-K s nastavením a nulováním. Vstupy J a K jsou vícenásobné, spojené v logickém součinu ($J = J1 \wedge J2 \wedge !J3$).

Obrázek 16-31. 7470 J-K FlipFlop s RS

```

      +---\ /---+
NC  -| 1      14 | - Vcc
!R  -| 2      13 | - !S
J1  -| 3      12 | - CP
J2  -| 4      11 | - K2
!J3 -| 5      10 | - K1
!Q  -| 6       9 | - !K3
GND -| 7       8 | - Q
      +-----+

```

Obvod 7471 obsahuje jeden klopný obvod typu J-K s nastavením a nulováním. Vstupy J a K jsou vícenásobné, spojené v součtu logických součinů AND-OR ($J = (J1 \wedge J2) \vee (J3 \wedge J4)$).

Obrázek 16-32. 7471 J-K FlipFlop s RS

```

      +---\ /---+
J1  -| 1      14 | - Vcc
J2  -| 2      13 | - !CP
J3  -| 3      12 | - K4
J4  -| 4      11 | - K3
!S  -| 5      10 | - K2
Q   -| 6       9 | - K1
GND -| 7       8 | - Q
      +-----+

```

Obrázek 16-33. 7472 JK FlipFlop

		+	---	\	/	---	+	
nc	-		1		14	-	Vcc	
R	-		2		13	-	S	
J1	-		3		12	-	CL	
J2	-		4		11	-	K3	
J3	-		5		10	-	K2	
!Q	-		6		9	-	K1	
GND	-		7		8	-	Q	
			+	-----	+			

16.5.3.1. 7473

✻

		+	---	\	/	---	+		
CP1	-		1		14	-		J1	
!R1	-		2		13	-		!Q1	
K1	-		3		12	-		Q1	
Vcc	-		4		11	-		GND	
CP2	-		5		10	-		K2	
!R2	-		6		9	-		Q2	
J2	-		7		8	-		!Q2	
		+	-----	+					

16.5.3.2. 7476

*

16.5.4. Monostabilní klopné obvody (122,123)

* **Attributy:** *id="ic.ff.monostable"*

Obvod 74123 obsahuje dvojici monostabilních klopných obvodů.

Obvod 74122 obsahuje jeden monostabilní klopný obvod.

16.6. Čítače

*

16.6.1. Desítkový čítač (90)

*

Obrázek 16-34. 7490 desítkový čítač s nulováním a nastavením

+---\ /---+			
!CP2	-	1 14	- !CP1
MR1	-	2 13	-
MR2	-	3 12	- Q0
	-	4 11	- Q3
Vcc	-	5 10	- GND
MS1	-	6 9	- Q1
MS2	-	7 8	- Q2
+-----+			

Obrázek 16-35. dekadický synchronní vratný čítač (74192)

+---\ /---+			
B	-	1 16	- Vcc
Qb	-	2 15	- A
Qa	-	3 14	- B
CD	-	4 13	- BO
CU	-	5 12	- CA
Qc	-	6 11	- L
Qd	-	7 10	- C
GND	-	8 9	- D
+-----+			

16.6.2. Čtyřbitový čítač (93, 193, 4029)

*

Obrázek 16-36. 7493 čtyřbitový čítač s nulováním

+---\ /---+			
!CP2	-	1 14	- !CP1
MR1	-	2 13	-
MR2	-	3 12	- Q0
	-	4 11	- Q3
Vcc	-	5 10	- GND
	-	6 9	- Q1
	-	7 8	- Q2
+-----+			

Obrázek 16-37. čtyřbitový synchronní vratný čítač (74193)

+---\ /---+			
B	-	1 16	- Vcc
Qb	-	2 15	- A
Qa	-	3 14	- B
CD	-	4 13	- BO
CU	-	5 12	- CA
Qc	-	6 11	- L
Qd	-	7 10	- C
GND	-	8 9	- D
+-----+			

Obrázek 16-38. čtyřbitový synchronní vratný čítač (4029)

+---\ /---+			
PL -	1	16 -	V _{CC}
Q3 -	2	15 -	CP
P3 -	3	14 -	Q2
P0 -	4	13 -	P2
!CE -	5	12 -	P1
Q0 -	6	11 -	Q1
!TC -	7	10 -	UP / !DN
GND -	8	9 -	BIN / DEC
+-----+			

16.6.3.

*

16.7. Posuvné registry

*

16.7.1. 5bit posuvný registr (96)

*

Obrázek 16-39. 5bit posuvný registr (7496)

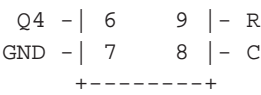
+---\ /---+			
CP -	1	16 -	!MR
D0 -	2	15 -	Q0
D1 -	3	14 -	Q1
D2 -	4	13 -	Q2
V _{CC} -	5	12 -	GND
D3 -	6	11 -	Q3
D4 -	7	10 -	Q4
PE -	8	9 -	DS
+-----+			

16.7.2. 8bit posuvné registry (164,165,595)

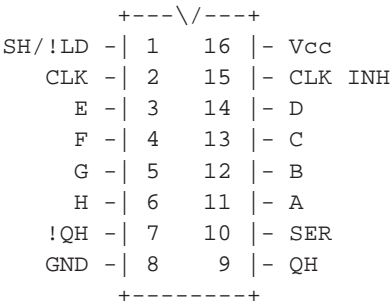
*

Obrázek 16-40. 8bit posuvný registr (74164)

+---\ /---+			
A -	1	14 -	V _{CC}
B -	2	13 -	Q8
Q1 -	3	12 -	Q7
Q2 -	4	11 -	Q6
Q3 -	5	10 -	Q5



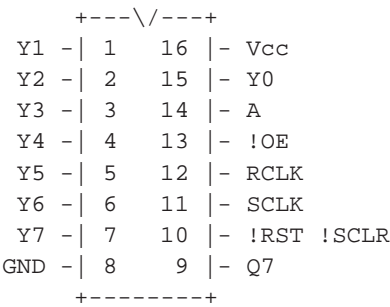
Obrázek 16-41. 8bit posuvný registr (74165)



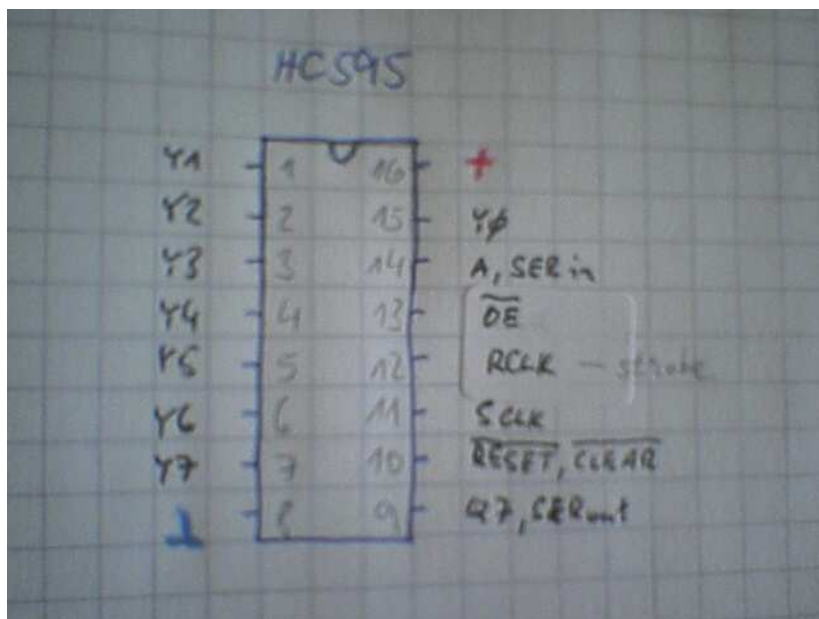
Tabulka 16-7. Funkční tabulka 74165

INPUTS			FUNCTION
SH/!LD	CLK	CLK INH	
L	X	X	Parallel load
H	H	X	No change
H	X	H	No change
H	L	↑	Shift
H	↑	L	Shift

Obrázek 16-42. 74595



Obrázek 16-43. Vývody pouzdra DIL16



Obvod se dá krásně použít například pro rozšíření počtu výstupních vývodu jednočipového mikropočítače.

Odkazy:

- Using shift registers to extend the microcontroller I/O²⁰
-

Celková zátěž na všechny výstupy 74HCT595 je v součtu 70mA maximálně. Na jeden výstup je to cca 20-35mA.

Cena TPIC6B595 v roce 2007 cca 1.2€.

SN74HC595N, MM74HC595N, MC74HC595N

16.8. Záchytné registry

Tyto obvody slouží k zapamatování informace a jejímu poskytování. V principu se jedná o klopné obvody typu D.

16.8.1. Obousměrný 8-mi bitový, třístavový záchytný registr

Obvod se vyskytuje ve dvou variantách, neinvertovaný 74543 a invertovaný 74544.

Obrázek 16-44. 74543, 74544

	+-----+	+-----+	
!LEBA -	1	24	- VCC
!GBA -	2	23	- !CEBA
A1 -	3	22	- B1
A2 -	4	21	- B2
A3 -	5	20	- B3
A4 -	6	19	- B4

A5	-	7	18	-	B5
A6	-	8	17	-	B6
A7	-	9	16	-	B7
A8	-	10	15	-	B8
!CEAB	-	11	14	-	!LEAB
GND	-	12	13	-	!GAB

+-----+

Poznámka: Obvod 543, 544 je prakticky nesehnatelný a nenašel jsem jej u žádného dodavatele.

16.8.2. Třístavový záchytný osmibitový registr

*

Obrázek 16-45. 74573

!OE	-	1	20	-	V _{CC}
D0	-	2	19	-	O0
D1	-	3	18	-	O1
D2	-	4	17	-	O2
D3	-	5	16	-	O3
D4	-	6	15	-	O4
D5	-	7	14	-	O5
D6	-	8	13	-	O6
D7	-	9	12	-	O7
GND	-	10	11	-	LE

+-----+

D₀-D₇

datové vstupy

LE

vstup latch enable aktivní v logické 1

!OE

vstup "3-state output enable", aktivní v logické 0

Q₀-Q₇

třístavové výstupy, aktivují se signálem !OE = 0

16.9.

*

16.10. Zvláštní, nebo zatím nezařazené obvody

16.10.1. VCO 74LS 124,624,625

*

16.10.2. 74147 — priority encoder

16.10.3. LS7222 — keypad programmable digital lock

Poznámky

1. <http://www.chipdir.org/>
2. <http://www.xs4all.nl/~ganswijk/chipdir/>
3. <http://www.kingswood-consulting.co.uk/giicm/>
4. <http://www.mil.ufl.edu/3701/pinouts/index.html>
5. <http://hw.cz/docs/giicm/GIICM.html>
6. http://www.capetronics.com/digital_ic_names.htm
7. http://frankshospitalworkshop.com/electronics/ic_7400.html
8. http://en.wikipedia.org/wiki/List_of_7400_series_integrated_circuits
9. http://en.wikipedia.org/wiki/List_of_4000_series_integrated_circuits
- 10.
11. http://www.avnet.co.za/Semi_log.htm
12. <http://www.piclist.com/techref/logic/family.htm>
13. <http://wiesi.dyndns.org/logic.html>
- 14.
- 15.
16. <http://www.eleccircuit.com/simple-pulse-generator-circuit-by-cd4049/>
17. <http://www.wiparat.com/ic-4049-clock-pulse-generator/>
18. <http://www.instructables.com/id/ST7565-LCDs-Graphical-LCDs/>
19. http://en.wikipedia.org/wiki/Logic_gate
20. <http://homepages.which.net/~paul.hills/Software/ShiftRegister/ShiftRegisterBody.html>

Kapitola 17. Obvody pro MCU

*

17.1. Paměti

*

17.1.1. SRAM 32KB

*

Obrázek 17-1. Zapojení pouzdra DIL32 paměti 62256

	+-----+	+-----+	
A14 -	1	+---+	28 - VCC
A12 -	2		27 - !WE
A7 -	3		26 - A13
A6 -	4		25 - A8
A5 -	5		24 - A9
A4 -	6		23 - A11
A3 -	7		22 - !OE
A2 -	8		21 - A10
A1 -	9		20 - !CS
A0 -	10		19 - D7
D0 -	11		18 - D6
D1 -	12		17 - D5
D2 -	13		16 - D4
GND -	14		15 - D3
	+-----+		

17.2. MMU

17.2.1. SN74610 - 613

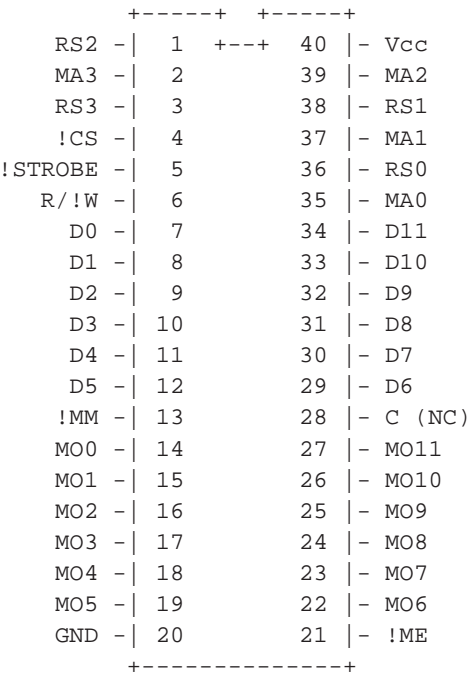
*

Odkazy:

- 74LS612 - MMU¹
-
-
-

Obvody SN74LS610,611,612 a 613 mají speciální použití jako MMU (*Memory Management Unit*) pro počítače. Obsahují paměť o celkové kapacitě 16 slov, každé 12 bitů velké.

Obrázek 17-2.



Tabulka 17-1.

Device	Output Latched	Map Output Type
LS610	Yes	3-State
LS611	Yes	Open-Collector
LS612	No	3-State
LS613	No	Open-Collector

17.3. Obvody reálného času

*

Odkazy:

- PCF8583² Clock and calendar with 240*8-bit RAM
- .
- .
- .

17.4. LED Drivers

17.4.1. TLC5940 od Texas Instruments

16 CHANNEL LED DRIVER WITH DOT CORRECTION AND GRAYSCALE PWM CONTROL

Odkazy:

- Review – Texas Instruments TLC5940 16-channel LED driver IC³
-

Tento obvod dokáže řídit 16 LED diod, nebo matici diod se společnou anodu.

Na straně připojení k mikrořadiči se tento obvod jeví jako posuvný registr. Přesněji dva posuvné registry, jeden 96 bitů dlouhý a druhý 192 bitů. Rozlišení mezi těmito registry se děje nastavením pinu VPRG. Jako posuvný registr je také kaskádovatelný a můžeme tak zapojit více obvodů za sebe a ovládat je stejně.

Data o jasu LED diod se přenášejí do 192 bitů dlouhého posuvného registru. Pro každou diodu jedno 12-ti bitové slovo. Data se přenášejí jako Big-Endian. Nejvyšší bit (MSB) diody 15 až nejnižší bit (LSB) diody 12 a pokračuje se až k diodě 0.

Další a podobné obvody:

- TLC5952⁴ — 24 kanálový budič LED pro 8 RGB LED diod.
-

Tabulka 17-2. Dostupnost TLC5940

typ	obchod/prodejce	datum	cena	na skladě
TLC5940	Farnell 12263065	2010-09-23	103.95 bez DPH	366

Poznámky

1. <http://www.baltissen.org/newhtml/74ls612.htm>
- 2.
3. <http://tronixstuff.wordpress.com/2010/07/19/review---texas-instruments-tlc5940-16-channel-led-driver-ic/>
- 4.

Kapitola 18. Zapojení s digitálními obvody

*

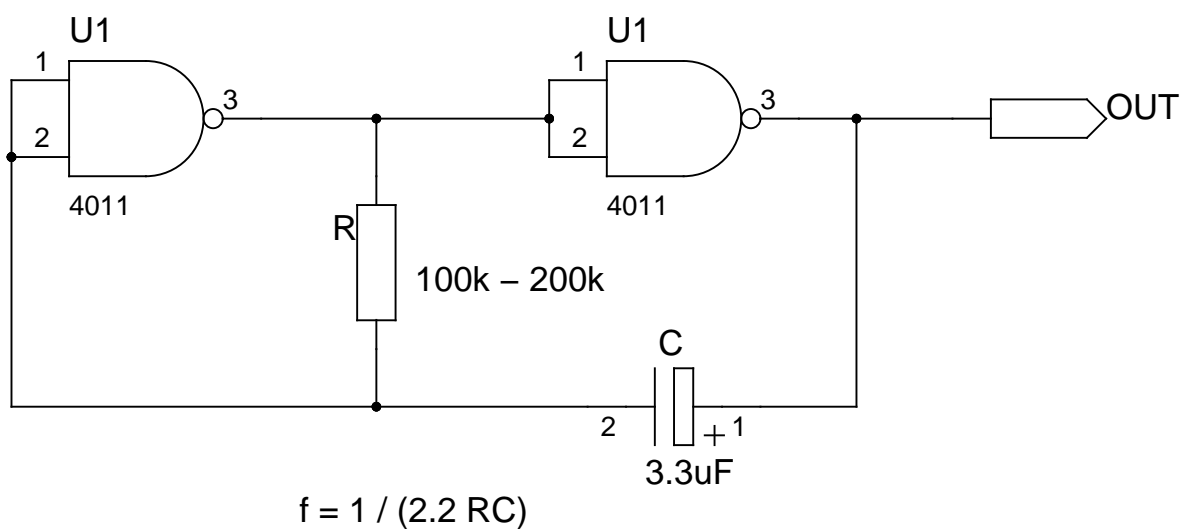
18.1. Oscilátory

*

Odkazy:

- Construct Square Wave Oscillator using CMOS Logic Element¹
- CMOS Logic Experiments²
-

Obrázek 18-1. RC Oscilátor s CD4011



18.2. MMU

*

Odkazy:

- Build your own Memory Management Unit³
- Virtual Address space for 6502 Microprocessors⁴
- <http://www.sxlist.com/techref/mem/dram/slide3.html>
-

18.3. Level Shifter

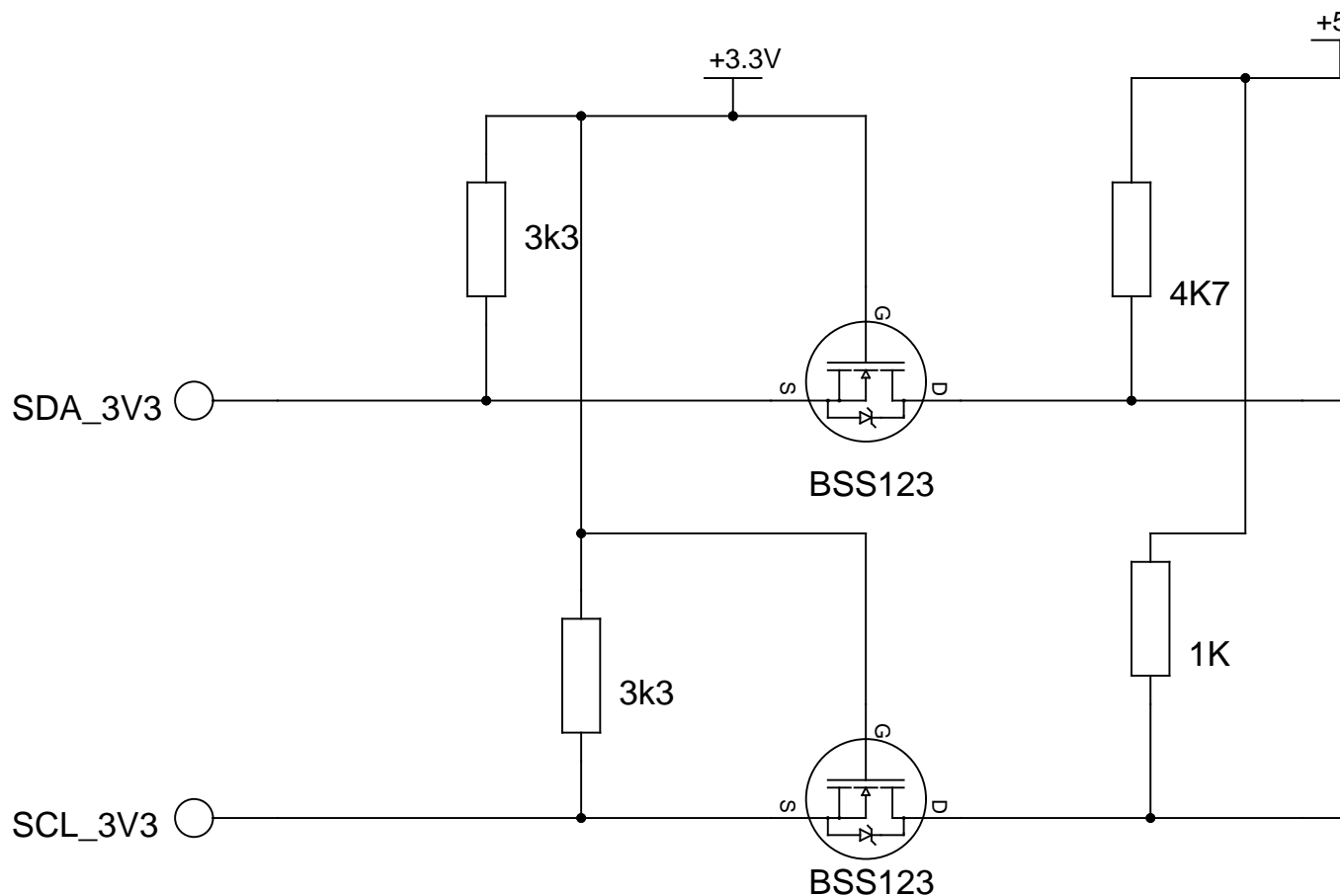
*

Odkazy:

- I2C Mixed-Voltage System⁵ na AVR freaks
- NXP⁶
- MAXIM APPLICATION NOTE 3007 Logic-Level Translation⁷

- ADG3304⁸
- breakout board, level-shifter, 3V/5V, 14-pin wide DIP⁹
- Texas Instruments TXB0104¹⁰

Obrázek 18-2.



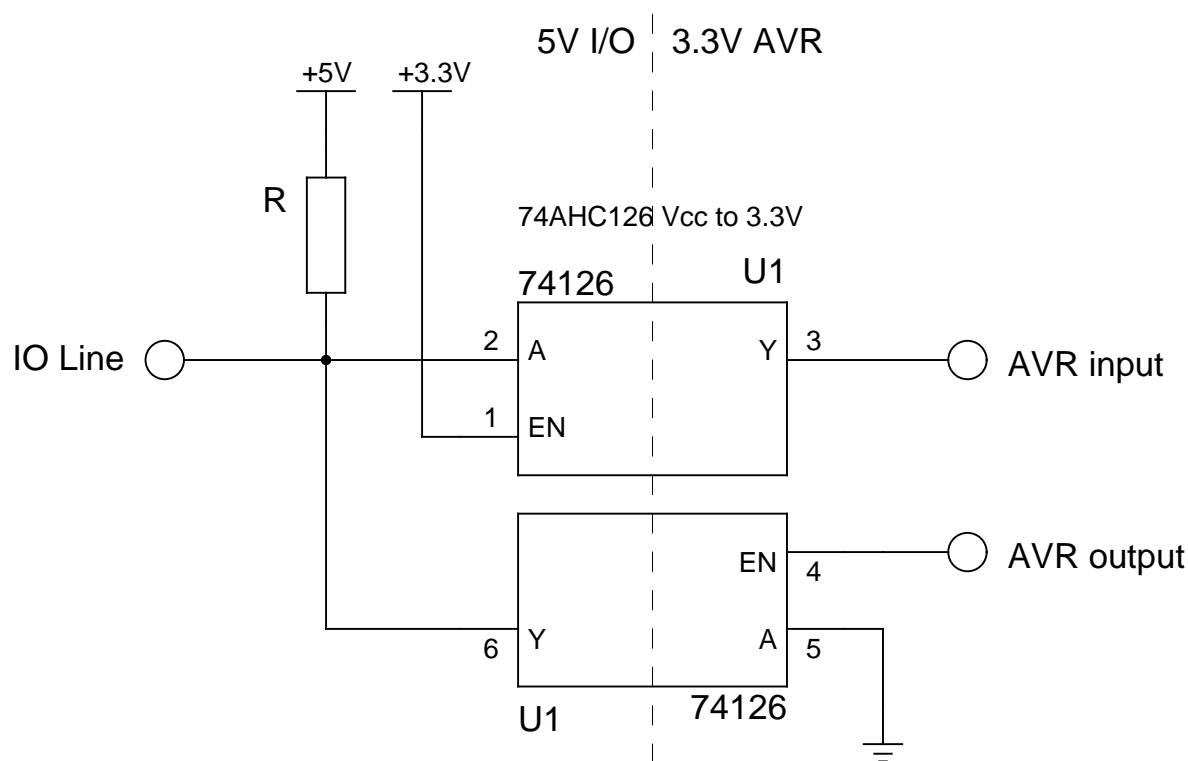
* gEDA image schematic: I2C_FET_level_shifter.sch: 640×480 EPS, PNG

Pro podobné účely lze použít obvody MAXIM: MAX3000E, MAX3001E, MAX3002, MAX3003, MAX3004, MAX3005, MAX3006, MAX3007, MAX3008, MAX3009, MAX3010, MAX3011, MAX3012¹¹. A možná taky obvody: LM3350 a QS3861.

Obvod: LVCC3245A.

Připojení 5V periferie k 3.3V MCU.

Obrázek 18-3.



* gEDA image schematic: 5V_IO_to_3V_AVR.sch: 640×480 EPS, PNG

Na uvedeném schématu je třeba použít obvod 74126 technologie ACT,HCT,AHCT. Důvodem jsou nižší hodnoty V_{IH} (2.0V) a V_T (1.5V). Údaje o napěťových hodnotách mám z Napěťové úrovně logických obvodů¹² ze serveru MCU.CZ.

* Pro napěťové přizpůsobení/oddělení I²C lze také použít obvod PCA9515.

* Pro 5 ↔ 3.3 V se dá použít obvod 74LVC245.

* Re: 3V ↔ 5V Level shifter¹³ When I run into problems of that sort, I usually use one of the buffer/drivers with 5V tolerant I/O. Something like a 74LCX245 would work.

* 74LVC4245A (5V ↔ 3.3V)

Poznámky

1. <http://datasheetoo.com/semiconductor-article/construct-square-wave-oscillator-using-cmos-logic-element.html#more-838>
2. <http://www.qrp.pops.net/cmos1.asp>
3. <http://www.baltissen.org/newhtm/mmu.htm>
4. <http://www.6502.org/users/andre/icapos/mmu65.html>
5. <http://www.avrfreaks.net/index.php?name=PNphpBB2&file=viewtopic&t=38133>
6. http://www.nxp.com/news/backgrounders/bg_esc9727/index.html
7. <http://www.maxim-ic.com/app-notes/index.mvp/id/3007>
- 8.

9. <http://www.nanocore12.com/products/details/27/4/accessories/breakout-board,-level-shifter,-3v/5v,-14-pin-wide-dip.html>
10. <http://focus.ti.com/docs/prod/folders/print/txb0104.html>
11. <http://www.maxim-ic.com/datasheet/index.mvp/id/3672>
12. <http://mcu.cz/news.php?extend.2495.3>
13. <http://www.embeddedrelated.com/usenet/embedded/show/5-1.php>

IV. Obvody a zapojení

* *Attributy: id="obvody"*

Různá zapojení a obvody.

Odkazy:

- JF1OZL QRP,HAND MADE AMATEUR RADIO,paper model,Boomerang;name:Kazuhiro Sunamura²³⁷
- Circuits for the Hobbyist²³⁸ VA3AVR
-
-

Poznámky

237. <http://www.intio.or.jp/jf10zl/>

238. <http://www.sentex.ca/~mec1995/circ/circuits.htm>

Kapitola 19. Rezonanční obvody

*

19.1. LC obvod

*

Odkazy:

- LC circuit¹ na Wikipedii
-
-
-

Poznámky

1. http://en.wikipedia.org/wiki/LC_circuit

Kapitola 20. Oscilátory

Odkazy:

- The 'Kitsilano' Oscillator Circuit¹
-
-

Poznámky

1. <http://catseye.tc/projects/kitsilano/kitsilano.html>

Kapitola 21. Tranzistorové klopné obvody

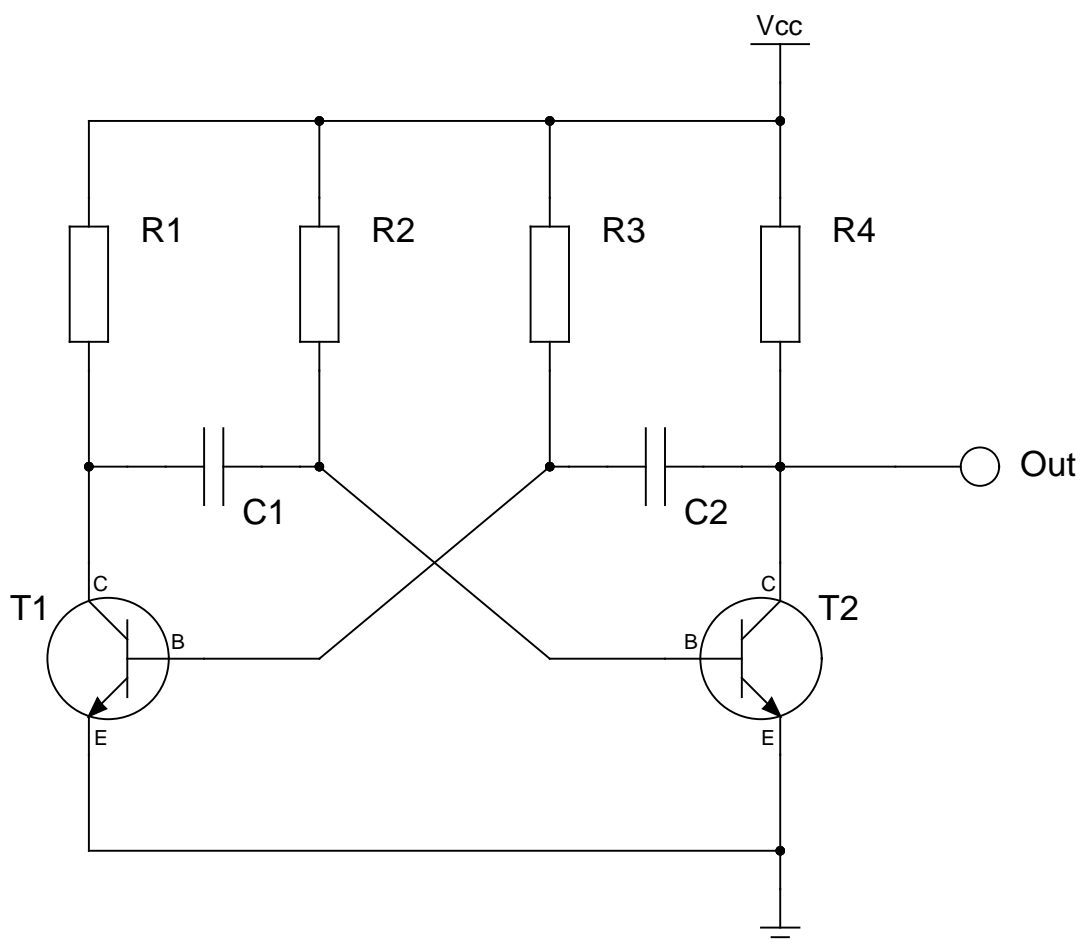
Odkazy:

- AR A/2 z roku 1986 strana 49
- Basic Transistor Circuits¹

21.1. Astabilní klopný obvod, multivibrátor

Zapojení základních transistorových oscilačních obvodů.

Obrázek 21-1. Tranzistorový multivibrátor



* *multivibrator2.sch, 640x480*

* *Obrázek multivibrátoru například z AR 5/66 strana 18.*

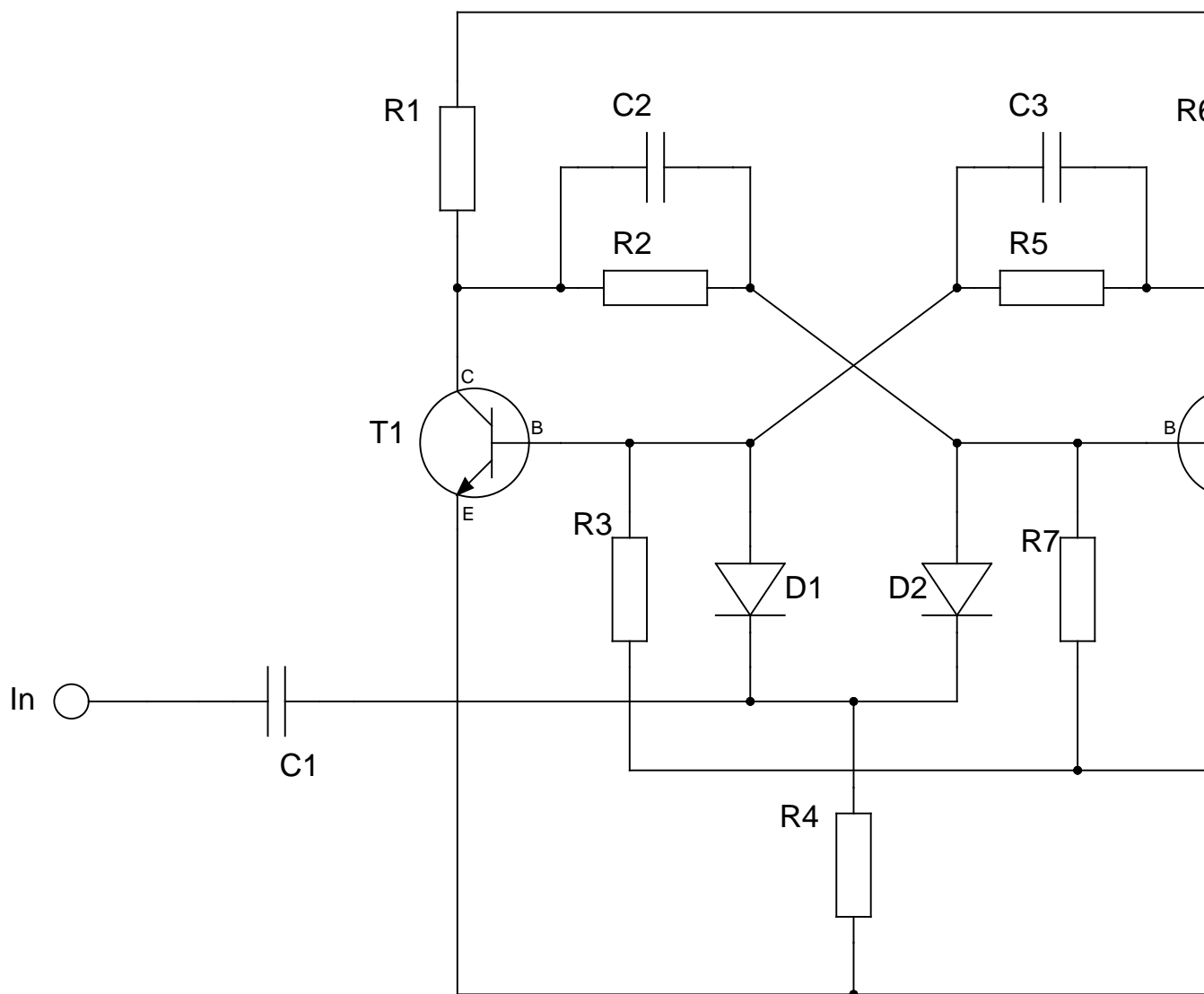
Tento typ multivibrátoru byl v literatuře popsán již mnohokrát.

21.2. Bistabilní klopný obvod / Dělička kmitočtu

Odkazy:

- Bistabilní preklápací obvod [BKO]² — velmi dobrý popis s teorií
-

Obrázek 21-2. Tranzistorový bistabilní klopný obvod jako dělička kmitočtu

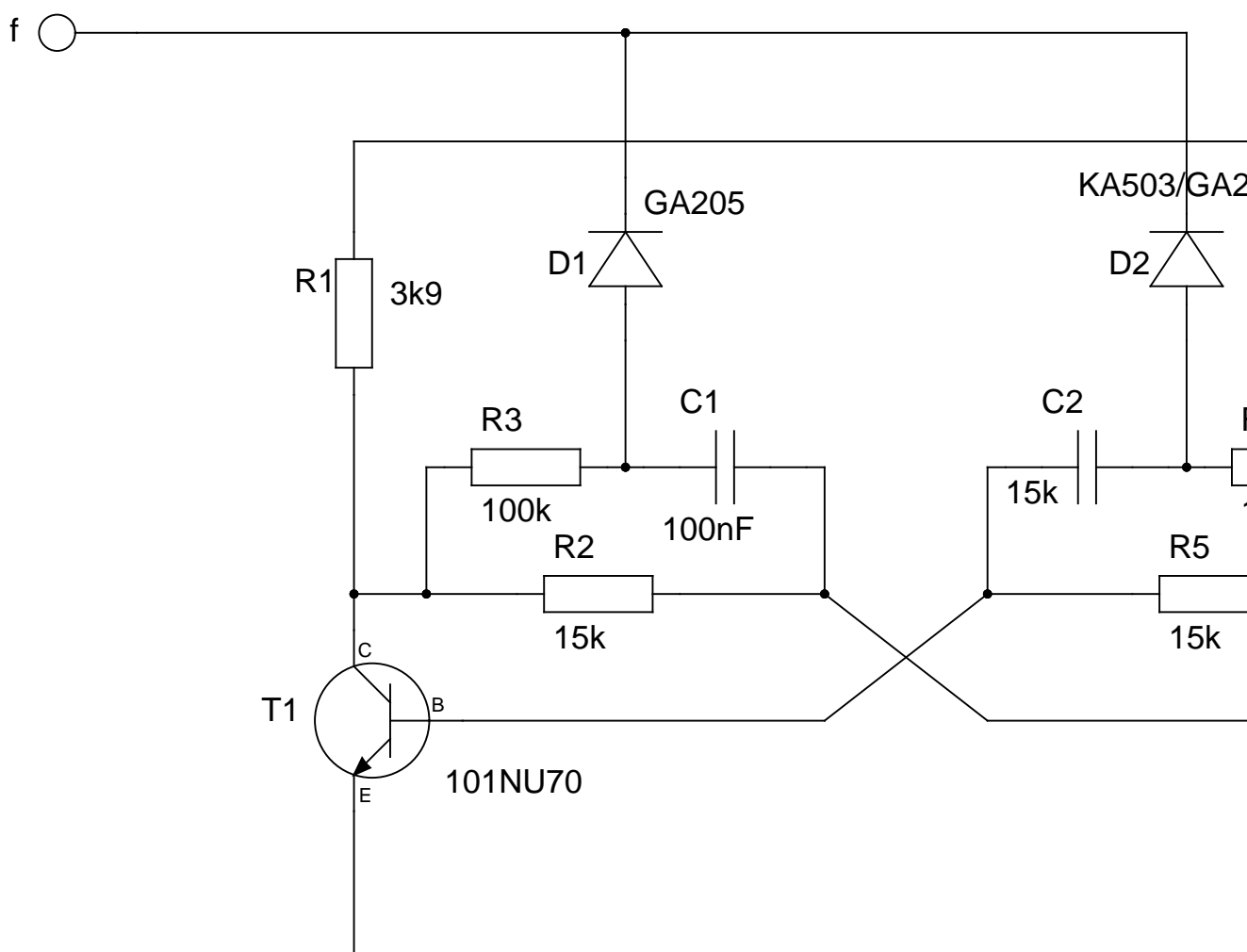


* *delicka-kmitoctu.sch*, 640x480

* Schéma převzato z AR 5/66 strana 19.

Další dělička kmitočtu pochází z AR 6/71 strana 212 z článku Elektronické varhany Herlod.

Obrázek 21-3. Dělička kmitočtu s germaniovými tranzistory



* *delic-kmitoctu2.sch, 640x480*

Tento dělič je rovněž použit v konstrukci elektronických varhan popsané v RK 6/72 na straně 5.

Další dělič kmitočtu je v konstrukci Elektronické minivarhany od Ing. Eduarda Moravce publikované v AR 1/75 na stranách 15-19.

Jiné zapojení děliče kmitočtu je v AR B 3/77 na straně 89.

* *Překreslit toto zapojení.*

Další varianta tranzistorového děliče kmitočtu je v AR A 5/77 na straně 185. V zapojení nejsou použity diody ale kondenzátory.

21.3. RS klopný obvod

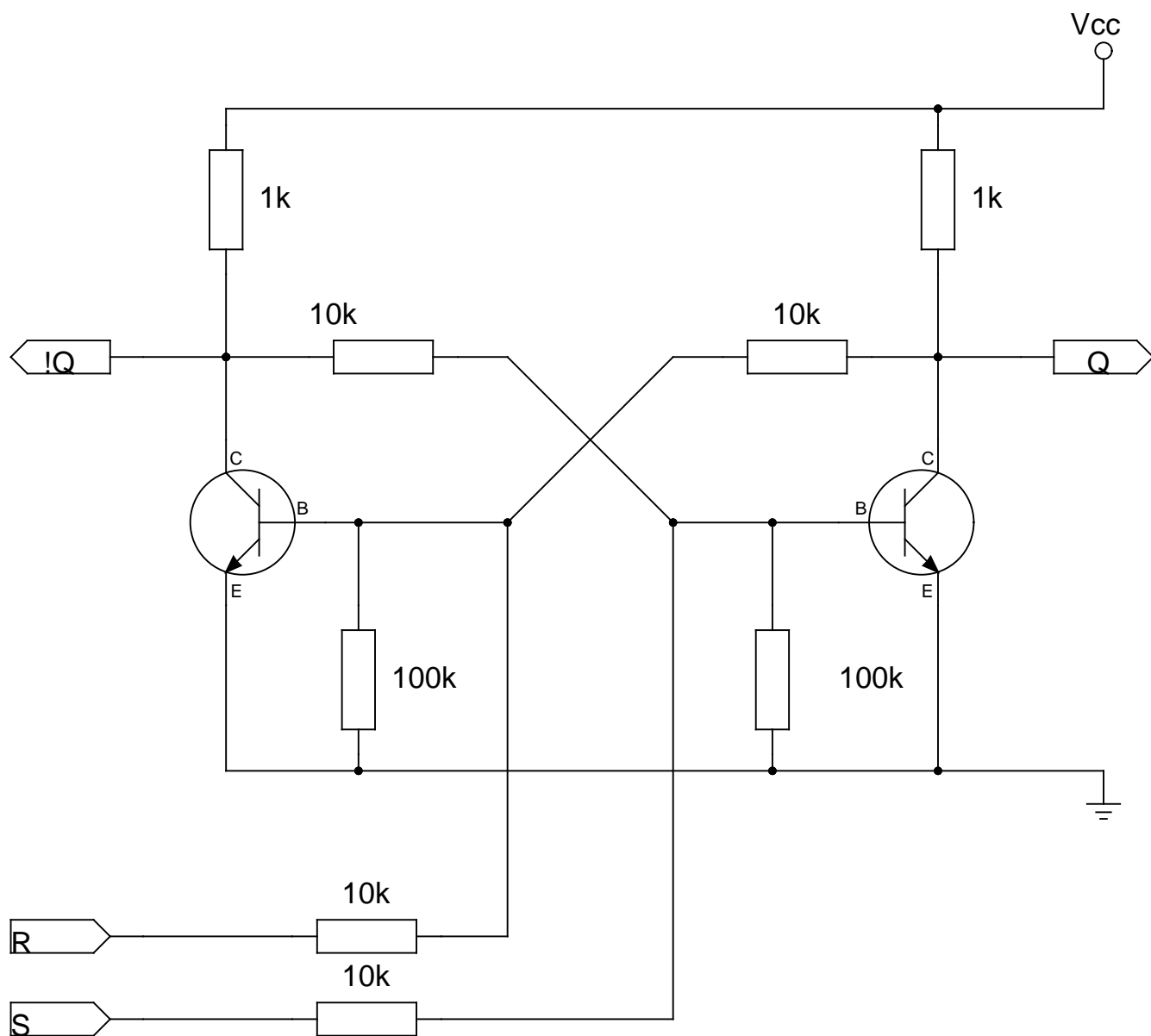
*

Odkazy:

- Na stránce Basic Transistor Circuits³ v částí RS Flip Flop (RSFF)
-

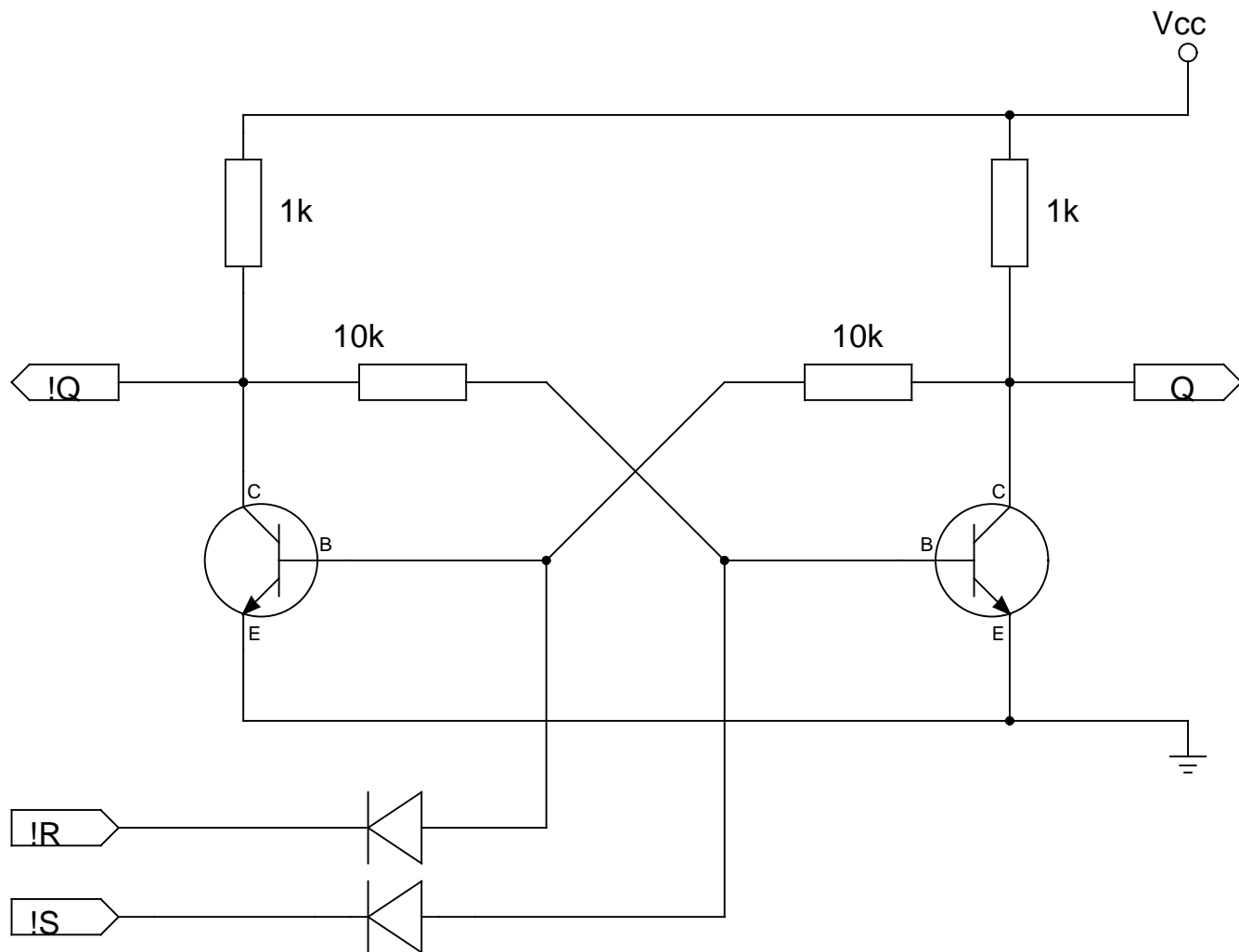
* http://pcbheaven.com/scripts/imagepresent.php?filename=%2Fwikipages%2Fimages%2Ftransistorcircuits_1235488532.jpg

Obrázek 21-4.



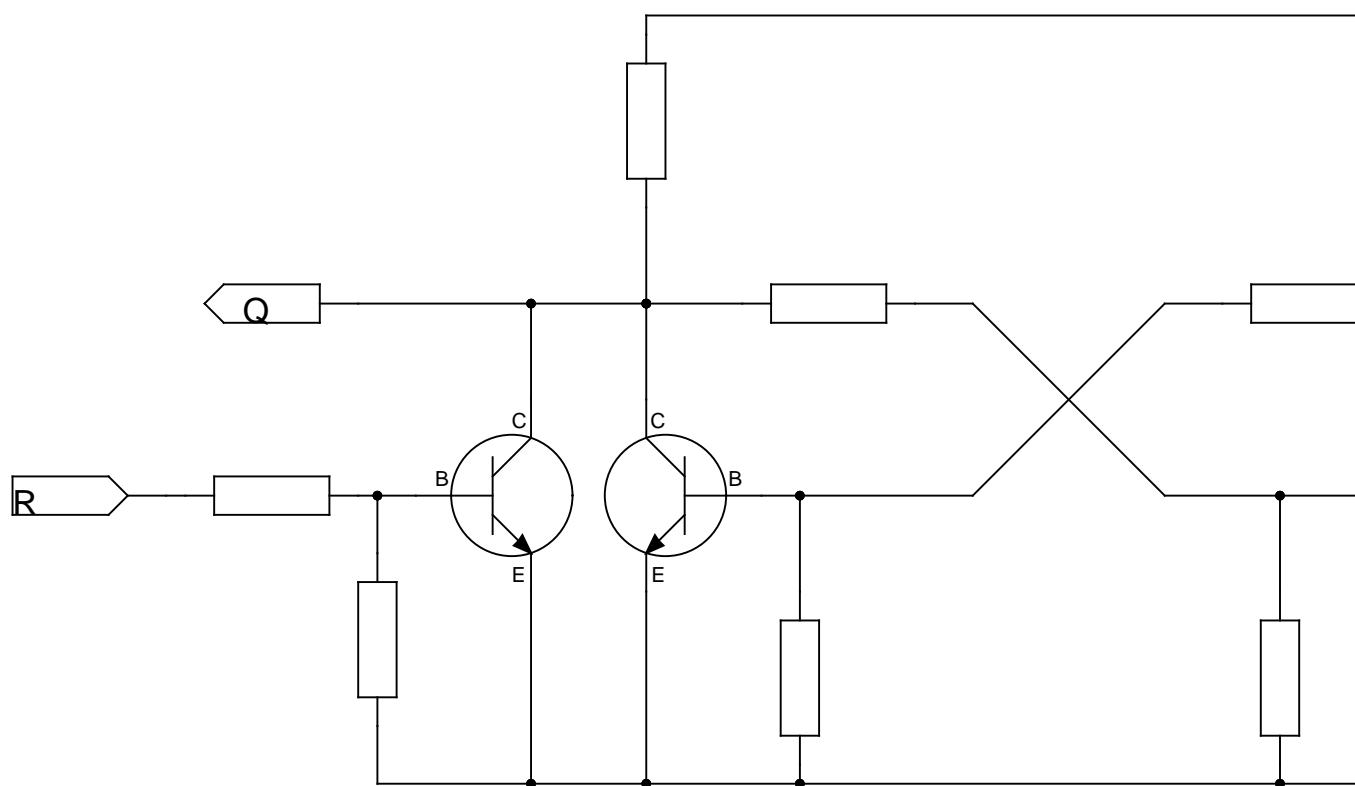
* gEDA image schematic Transistor_RS_FlipFlop.sch: 640×480 EPS, PNG

Obrázek 21-5.



* gEDA image schematic Transistor_RS_FlipFlop_2.sch: 640×480 EPS, PNG

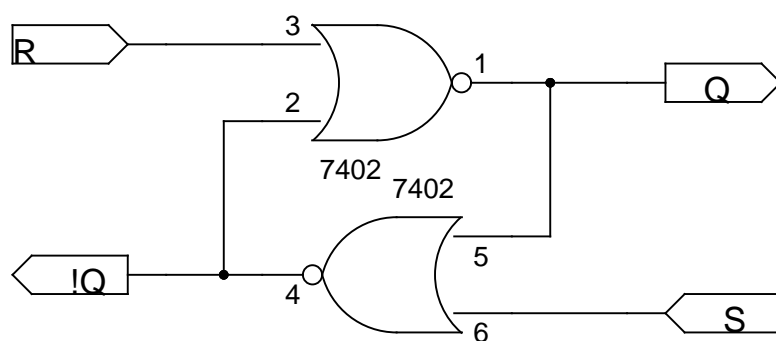
Obrázek 21-6.



* gEDA image schematic Transistor_RS_FlipFlop_3.sch: 640×480 EPS, PNG

Dvojice tranzistorů v tomto klopném obvodu tvoří vlastně samostatná hradla NOR.

Obrázek 21-7.



* gEDA image schematic RS_FF_NOR.sch: 320×240 EPS, PNG

Poznámky

1. http://pcbheaven.com/wikipages/Transistor_Circuits/

2. <http://alzat.szm.com/Oscilat/generato/bko/bko.htm>
3. http://pcbheaven.com/wikipages/Transistor_Circuits/

Kapitola 22. Zobrazovače

Odkazy:

- Handmade LED Clock 2¹ — zajímavé požití LED diod k vytvoření zobrazovače připomínajícího digitrony
-

Poznámky

1. <http://www.flickr.com/photos/barsprojects/sets/72157624179771160/>

Kapitola 23. Velmi jednoduchá zapojení

*

23.1. Joule Thief

*

Odkazy:

- MAKE A JOULE THIEF¹
-

Poznámky

1. <http://www.emanator.demon.co.uk/bigclive/joule.htm>

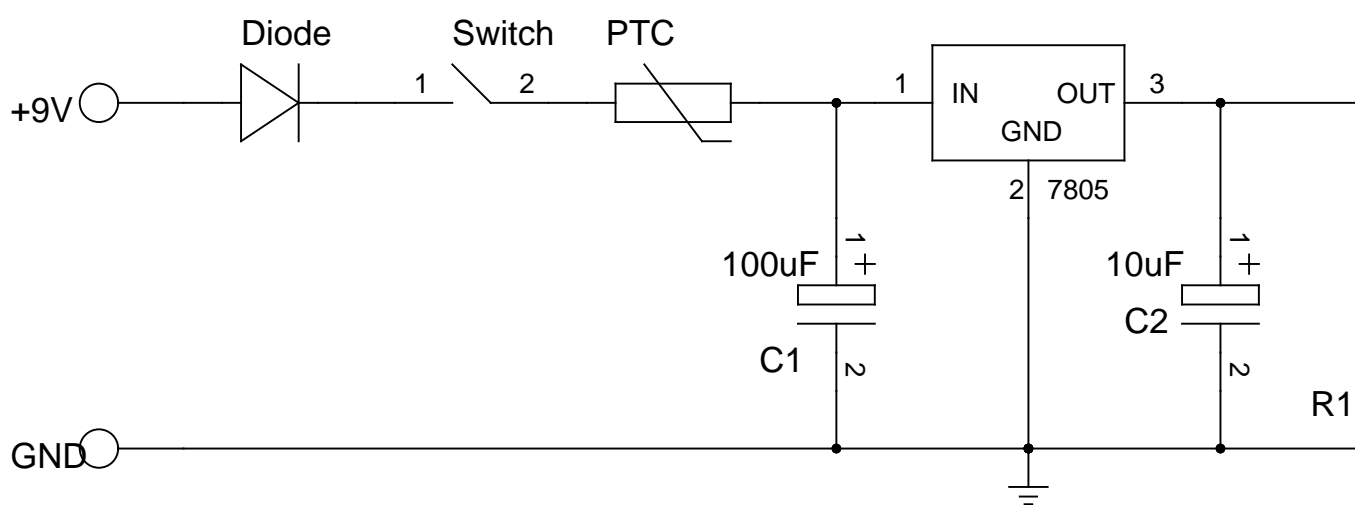
Kapitola 24. Zdroje

*

24.1. Stabilizátory napětí

*

Obrázek 24-1. Zapojení lineárního stabilizátoru napětí



* Image size 640*480 PNG, EPS.

V. Komunikace a kódování

Kapitola 25. Reprezentace čísel

*

Celá přirozená čísla v čteně nuly můžeme snadno reprezentovat v přímém binárním kódování takto.

$\dots 0000 = 0$
 $\dots 0001 = 1$
 $\dots 0010 = 2$
 $\dots 0011 = 3$
 $\dots 0100 = 4$
 $\dots 0101 = 5$
 \vdots

Hodnota bitů v jednotlivých pozic číselného vyjádření vyjadřuje násobky základu číselné soustavy. Tedy podobně jako v dekadické soustavě (soustavě o základu 10) jednotlivé číslice určují násobky 1, 10, 100, ..., podle pozice na které se nacházejí.

Rovnice 25-1. Vyjádření čísla v dvojkové soustavě

$$\sum_{i=0}^k m_i 2^i$$

V uvedeném vzorci je k počet bitů v reprezentaci čísla a m_i hodnota bitu na i -té pozici.

25.1. Vyjádření celých čísel

*

Celá čísla můžeme vyjádřit několika způsoby

- *V přímém binárním kódu se znaménkem.* Kdy nejvyšší bit v reprezentaci čísla vyjadřuje znaménko čísla. Obvykle 0 vyjadřuje kladná čísla a 1 záporná čísla.
- *V inverzním kódu,* kdy kladná čísla vyjadřujeme v přímém binárním kódování a záporná čísla jsou vytvořena inverzí, t.j. negováním všech bitů v reprezentaci čísla.
- *Doplňkovým kódem, ve kterém jsou kladná čísla vyjádřena v přímém binárním kódování, a záporná čísla k nim vytvoříme tak že provedeme inverzi všech bitů a následně přičteme hodnotu 1*

Tolik ve zkratce.

Tabulka 25-1.

binární reprezentace		význam v kódu
\pm	bity	přímém se znaménkem
0	0 ... 0 0	0
0	0 ... 0 1	1
0	0 ... 1 0	2
	\vdots	

binární reprezentace		význam v kódu
\pm	bity	přímém se znaménkem
0	1 ... 1 1	+MAX
1	0 ... 0 0	-0
1	0 ... 0 1	-1
	\vdots	
1	1 ... 1 0	-MAX +1
1	1 ... 1 1	-MAX

Kapitola 26. Kódy

26.1. ASCII

Odkazy:

- [Understanding ASCII Codes¹](#)
-

Tabulka 26-1. Význam řídicích kódů ASCII

název	dec	hex	popis
NUL	0	00	
SOH	1	01	
STX	2	02	
ETX	3	03	
EOT	4	04	
ENQ	5	05	
ACK	6	06	
BEL	7	07	
BS	8	08	Back Space
HT	9	09	Horizontal Tab
LF	10	0A	Linefeed
VT	11	0B	Vertical tab
FF	12	0C	Formfeed — page eject
CR	13	0D	
SO	14	0E	
SI	15	0F	
DLE	16	10	
DC	17	11	
DC2	18	12	
DC3	19	13	
DC4	20	14	
NAK	21	15	
SYN	22	16	
ETB	23	17	
CAN	24	18	
EM	25	19	
SUB	26	1A	
ESC	27	1B	
FS	28	1C	
GS	29	1D	

název	dec	hex	popis
RS	30	1E	
US	31	1F	
ALT	125	7D	}
PRE	126	7E	~
DEL	127	7F	

26.2. GCR

Odkazy:

- The Apple Story, Part 2²
- The Complete Commodore Inner Space Anthology page 49³
- Group code recording⁴
- Line code⁵

Tabulka 26-2. Commodore GCR Codes

Hex	GCR	Binary	Dec
\$00	01010	0000	0
\$01	01011	0001	1
\$02	10010	0010	2
\$03	10011	0011	3
\$04	01110	0100	4
\$05	01111	0101	5
\$06	10110	0110	6
\$07	10111	0111	7
\$08	01001	1000	8
\$09	11001	1001	9
\$0A	11010	1010	10
\$0B	11011	1011	11
\$0C	01101	1100	12
\$0D	11101	1101	13
\$0E	11110	1110	14
\$0F	10101	1111	15

Poznámka: GCR is the method in which disk data is magnetically stored. It is based on transitions (ie. 1 to 0, or 0 to 1) A transition is decoded as 0, no transition decodes to a 1.

Tabulka 26-3. GCR: (0,2) RLL

Nybble	Code	Nybble	Code
0000	11001	1000	11010

Nybble	Code	Nybble	Code
0001	11011	1001	01001
0010	10010	1010	01010
0011	10011	1011	01011
0100	11101	1100	11110
0101	10101	1101	01101
0110	10110	1110	01110
0111	10111	1111	01111

Tabulka 26-4. 4B5B Encoding Table

Name	4b	5b	Description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
Q	—	00000	Quiet (signal lost)
I	—	11111	Idle
J	—	11000	Start #1
K	—	10001	Start #2
T	—	01101	End
R	—	00111	Reset
S	—	11001	Set
H	—	00100	Halt

Poznámky

1. http://www.nadcomm.com/ascii_code.htm
2. <http://apple2history.org/museum/articles/byte8501.html>
3. <http://www.zimmers.net/anonftp/pub/cbm/manuals/anthology/p049.jpg>

4. http://en.wikipedia.org/wiki/Group_code_recording
5. http://en.wikipedia.org/wiki/Line_code

Kapitola 27. IR komunikce

*

Sony SIRC:

- Sony SIRC Protocol¹
- SONY infrared remote protocol²
-
-

Poznámky

1. <http://www.sbprojects.com/knowledge/ir/sirc.htm>
2. <http://users.telenet.be/davshomepage/sony.htm>

VI. Počítačová architektura

Odkazy:

- Paper processor. What is fetch, decode, and execute?²⁵³
-

Poznámky

253. <https://sites.google.com/site/kotukotuzimiti/>

Kapitola 28. Poznámky

Odkazy:

- Homemade computers¹
- Computer Architecture²
- Cybernetic Computer and Electronic Brain³ The fascinating story of how computers work in clear, non-mathematical language

Poznámky

1. <http://www.holmea.demon.co.uk/Links.htm#Homemade>
2. http://www.rdrop.com/~cary/html/computer_architecture.html
3. <http://www.lauftext.de/cybernetic-computer/index.htm>

Kapitola 29. Návrh procesorů

29.1. 32 bitový procesor orientovaný na Forth

Tento procesor může být realizován i jako virtuální stroj zakomponovaný do jádra Forthu.

Při návrhu vycházím z poznatků z procesorů SOP32, F18, ...

Tabulka 29-1. Registry procesoru

název	popis
IP	Ukazatel instrukcí (zvaný také PC). Ukazuje na následující instrukci
IR	instrukční registr obsahuje právě dekodované instrukční slovo
SP	datový zásobník
RP	zásobník návratových adres (>R R> R@)
A	adresní registr který používají instrukce A@+, ...
W	pomocný pracovní registr, není přímo přístupný

Nejdříve instrukce. Z pohledu procesoru existují atomické instrukce, atomy a instrukční slova. Slovo obsahuje obvykle několik atomických instrukcí. Protože je nutno skloubit malý počet instrukcí a potřebu velkého počtu bitů pro adresu při volání poprogramu, je instrukční slovo rozděleno do několika skupin bitů. Jedná se o dlouhé instrukční pole, a dva jednobitové příznaky, příznak návratu a příznak adresy (skoku).

Základní struktura instrukčního slova:

```
+-----+-----+-----+
|      instrukční pole      |Adr|Ret|
+-----+-----+-----+
```

Tabulka 29-2. Struktura instrukčního slova

instrukční pole	A	R	popis
adresa	0	0	instrukce volání podprogramu
adresa	0	1	instrukce skoku
pole atomických instrukcí	1	0	atomické instrukce
pole atomických instrukcí	1	1	atomické instrukce s návratem

Při vytváření instrukční sady jsem vycházel z následujících úvah:

- Nejdříve jsem popsal instrukci volání podprogramu. Protože podprogramy mohou být adresovány jedinečně na celá 32-bitová slova, musí být spodní dva bity adresy nulové. Rozhodl jsem se tedy pro jednoduchost že dva spodní nulové bity kódují volání a vyšších 30 bitů tvoří vyšších 30 bitů adresy doplněné zprava dvěma nulovými bity. Vzhledem k tomu že instrukce má spodní dva bity nulové je tedy přímo uvedená adresa adresou podprogramu.
- Potřeboval jsem se rozhodnout, jaký význam přidělím spodním dvěma bitům. Protože jedna z potřebných informací je příznak návratu, rozhodl jsem se nevytvářet dvoubitové pole a jednotlivým kombinacím přidělovat význam, ale přímo přidělit význam jednotlivým bitům. Tyto dva bity jsem pojmenoval R —

Return a A — *Address* a přidělil jim následující význam.

Bit	Název	Význam
0	Ret	Implicitní return, je-li v poli 1, dojde po vykonání instrukce/instrukcí k návratu z podprogramu.
1	Adr	Adresní bit, 0 - v instrukčním poli je adresa, 1 - v instrukčním poli jsou atomické instrukce

- Pak jsem si napsal jednotlivé kombinace bitů přehledně do tabulky abych se nad nimi mohl zamyslet.

b₃₁-b₂ (I)	b₁ (A)	b₀ (R)	mnemo	Význam
adresa	0	0	CALL	volání podprogramu na adrese
adresa	0	1	JUMP	předání řízení na adresu
atomy	1	0		atomické instrukce
atomy	1	1		atomické instrukce s návratem

Tři kombinace my byly hned jasné, bylo to volání podprogramu a atomické instrukce s návratem a bez. Nad čtvrtou kombinací jsem se na chvíli zamyslel. Uvědomil jsem si že ji můžu použít ve významu *tail call*. Což je technicky obyčejný skok na adresu. Takováto instrukce použitá na konci Forth slova způsobí že se předá řízení jinému Forth slovu a již se nebude provádět návrat neb by to bylo zbytečné.

- V takovémto posloupnosti instrukcí se mohou vyskytovat i jiná data, jak bude popsáno dále. V principu se ale bude vždy jednat o parametry atomické instrukce.
- Atomy jsou v instrukci umístěny v pořadí od nižších bitů. To dovoluje vybírat následující atom pomocí jednoduchého posunu bitů v instrukci do prava.
- Atomu, ve kterém jsou všechny bity nulové, jsem přiřadil význam atomické instrukce NOP, která neprovádí nic. To mi dovoluje takové instrukce ignorovat. Rovněž mi to dovolí snadno detekovat, že již byly zpracovány všechny atomy v instrukci. Pokud je instrukční registr prázdný (=0) byly všechny atomy již vykonány.

U tohoto konceptu jsem přemýšlel, jestli nemá význam změnit pořadí polí Instrukce/Adresa, A a R. Zdá se mi, že pro hardwarovou realizaci je to naprosto jedno. Proto jsem raději zůstal u této kombinace.

Nyní k atomickým instrukcím. Vzhledem k počtu bitů jenž je kódují (30) se můžeme rozhodnout, kolik atomů a jak velikých v instrukci bude. Ostatní procesory podobného typu používají instrukce kódované minimálně 5 bity. Protože mi to tak pěkně vyšlo a $30 = 6 * 5$, mohu se rozhodnout jestli v instrukci bude 6 atomů po pěti bitech, nebo 5 atomů po šesti bitech. Toto je ponecháno k dalším úvahám. Jediné co vyžadují, aby kombinace kdy všechny bity jsou nulové měla význam instrukce NOP. V následující tabulce, do které si budu vepisovat jednotlivé atomické instrukce, počítám proto se 6-ti bity.

Tabulka 29-3.

b₅	b₄	b₃	b₂	b₁	b₀	mnemo	význam
0	0	0	0	0	0	NOP	žádná operace, nedělá se nic
0	0	0	0	0	1		
0	0	0	0	1	0		
0	0	0	0	1	1		
0	0	0	1	0	0		
0	0	0	1	0	1		

b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	mnemo	význam
0	0	0	1	1	0		
0	0	0	1	1	1		
0	0	1	0	0	0		
0	0	1	0	0	1		
0	0	1	0	1	0		
0	0	1	0	1	1		
0	0	1	1	0	0		
0	0	1	1	0	1		
0	0	1	1	1	0		
0	0	1	1	1	1		
0	1	0	0	0	0		
0	1	0	0	0	1		
0	1	0	0	1	0		
0	1	0	0	1	1		
0	1	0	1	0	0		
0	1	0	1	0	1		
0	1	0	1	1	0		
0	1	0	1	1	1		
0	1	1	0	0	0		
0	1	1	0	0	1		
0	1	1	0	1	0		
0	1	1	0	1	1		
0	1	1	1	0	0		
0	1	1	1	0	1		
0	1	1	1	1	0		
0	1	1	1	1	1		
1	0	0	0	0	0		
1	0	0	0	0	1		
1	0	0	0	1	0		
1	0	0	0	1	1		
1	0	0	1	0	0		
1	0	0	1	0	1		
1	0	0	1	1	0		
1	0	0	1	1	1		
1	0	1	0	0	0		
1	0	1	0	0	1		
1	0	1	0	1	0		
1	0	1	0	1	1		
1	0	1	1	0	0		
1	0	1	1	0	1		
1	0	1	1	1	0		
1	0	1	1	1	1		

b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	mnemo	význam
1	1	0	0	0	0		
1	1	0	0	0	1		
1	1	0	0	1	0		
1	1	0	0	1	1		
1	1	0	1	0	0		
1	1	0	1	0	1		
1	1	0	1	1	0		
1	1	0	1	1	1		
1	1	1	0	0	0		
1	1	1	0	0	1		
1	1	1	0	1	0		
1	1	1	0	1	1		
1	1	1	1	0	0		
1	1	1	1	0	1		
1	1	1	1	1	0		
1	1	1	1	1	1		

Tabulka 29-4. Základní instrukce pro umístění do tabulky atomických instrukcí

instr.	zásobník	název	popis
!	n addr →	STORE	uloží hodnotu n do buňky na adrese addr, adresa musí respektovat zarovnání buňek v paměti
@	addr → n	FETCH	
+	n ₁ n ₂ → n	ADD	
-	n ₁ n ₂ → n	SUB	odečte n ₂ od n ₁
>R	n →		uloží n do návratového zásobníku (Return Stack)
R>	→ n		vyzvedne n z návratového zásobníku (Return Stack)
AND	n ₁ n ₂ → n		
OR	n ₁ n ₂ → n		
XOR	n ₁ n ₂ → n		
NOT	n → n		
DROP	n →		
DUP	n → n n		
OVER	n ₁ n ₂ → n ₁ n ₂ n ₁		
SWAP	n ₁ n ₂ → n ₂ n ₁		
[IF]	n →		
[CALL]			
[EXIT]			
[LIT]			

29.2. Procesor se sériovou ALU

Jedním ze způsobů jak snížit počet hradel v procesoru na úkro rychlosti je realizovat ALU jako jednobitovou a provádět všechny operace sériově. Tím se mimo alu zjednoduší datová případně i adresová sběrnice. Velkou daní za snížení počtu hradel je i snížení rychlosti. Každá operace bude vyžadovat minimálně tolik cyklů, kolik bitů mají zpracovávaná slova.

29.3. SOC-8

* Zvážit název počítače/procesoru. Názvy k zamyšlení: OC-8 (Octal Computer 8), TO-8 (Transistor Octal 8), ROC-8 (Radek's Octal Computer 8), ...

Následující text je ve stádiu rozpracování. Text není konzistentní protože do něj postupně zapracovávám nové myšlenky. Na druhou stranu se některých myšlenek zbavuji a nemusle jsem je odstranit ze všech částí textu.

Prvotní motivací pro tuto konstrukci bylo navrhnout počítač inspirovaný počítači TX-0, PDP-8 a CDC-160.

Kritéria návrhu:

- Šířka datového/instrukčního slova je 12 bitů.
- Šířka adresy bez jednotky MMU je rovněž 12 bitů. To umožňuje adresovat 2^{12} slov tedy 4 kiW.
- Jednoduchá konstrukce. Do té míry jednoduchá že ji bude možno realizovat s pomocí tranzistorů. Případně jen základních logických IO. (Ale jednodušší než níže zmíněné PDP-8 a TX-0 nebude.)
- Vychází ideově z PDP-8 a TX-0.
- Orientace na osmičkovou soustavu. Návrh instrukční sady tak aby bylo lze jednoduše dekodovat instrukce.
- Odstranění adresního módu s přístupem do aktuální stránky. Tento mód nahrazuji částečně módem *P Relative*.

Po řadě úvah a několikerém přepracování návrhu jsem přišel s následující konstrukcí.

29.3.1. Registry

Popis počítače začínám popisem jeho registrů. Někde začít musím. Tak jak postupně uzrává návrh modifikuji i tento seznam registrů.

Procesor má 4 programátorovi přístupné registry P, A, L a C. První tři jsou dvanáctibitové, poslední je jednobitový. Další registry jsou součástí konstrukce a nejsou přímo přístupné programátorovi. Tyto registry jsou využity v průběhu vykonávání instrukcí.

* Předmětem úvah je zavedení registru S jako ukazatele na zásobník a jako náhradu za registr L. Další možností je přidání dalšího akumulátoru B. Rovněž je předmětem úvah zavedení registru příznaků F známého z jiných architektur také jako PSW a začlenění jednobitového registru C do něj.

Tabulka 29-5. Registry procesoru SOC-8

#	symbol	název	poznámka
0	A, AC	první střadač (akumulátor A)	Pracovní registr. Je cílem nebo zdrojem dat řady instrukcí.
1	B, BC	druhý střadač (akumulátor B)	Pracovní registr. Je cílem nebo zdrojem dat řady instrukcí.

#	symbol	název	poznámka
2	P, PC	čítač instrukcí	Registr obsahuje adresu následující instrukce. Při čtení instrukce z paměti je vždy zvětšen o 1.
3	L, LINK	spojovací registr (link)	Využívá se při volání podprogramů a návratu z nich.
4	S, SP	ukazatel zásobníku	
5	T, TP	ukazatel zásobníku	
6			
7	PSW, PCW	registr příznaků	Obsahuje příznaky C,Z,N. Jednobitové registry IE, ...
	EA	efektivní adresa	Do tohoto registru je počítána efektivní adresa operandu v paměti.
	OP	operand	Do tohoto registru je načítán operand, převážně z efektivní adresy.
	MAR		Memory Address Register
	MDR		Memory Data Register

* Ve sloupečku název je doplněn seznam alternativních názvů registru.

Obrázek 29-1.

11	10	9	8	7	6	5	4	3	2	1	0
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+											
SUP		:		: IE		: N		Z		C :	
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+											

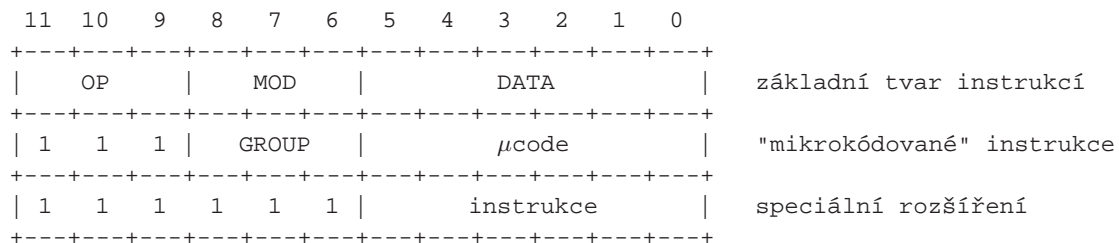
Tabulka 29-6. Bity stavového slova

bit	sym- bol	název	poznámka
0			
1			
2			
3	C	carry	
4	Z	zero	
5	N	negative	
6	I, IE	interrupt enabled	Informuje zdali je povoleno přerušení.
7			
8			
9			
10			
11	SUP	supervisor	Indikuje že běží v režimu supervisor a smí vykonávat všechny instrukce. V uživatelském režimu je repertoár instrukcí omezen.

29.3.2. Instrukční sada

O struktuře a jednoduchosti návrhu vypoví instrukční sada. Nejdříve tedy jak vypadá formát instrukcí. Po úvaze jsem přišel s takovým rozdělením polí v instrukčním slově které odpovídá oktalovým číslicím. Tedy dělím instrukční slovo po trojicích bitů. Toto dovoluje velmi snadno „číst“ instrukce přímo z oktalového výpisu paměti.

Obrázek 29-2. Formáty instrukcí SOC-8



* Obrázek je velmi orientační.

Pro jednoduchost, tentokrát nikoliv konstrukční ale jednoduchost programování ve strojovém kódu, jsem umístil všechna pole tak aby byla na hrnicích trojic bitů. Instrukce lze pak velmi snadno číst i psát v oktalovém zápisu. Je tomu tak proto, že jsem se rozhodl prodloužit počet bitů adresního módu ze dvou které má PDP-8 na tři. Zvětšeným počtem adresních módů sice naroste počet logických hradel v obvodu dekodování a výpočtu adresy a získávání operandu. Získám tak ovšem variabilnější architekturu která nebude mít některé vlastnosti které mi na PDP-8 vadily. Jedná se zejména o adresní režimy v aktuální stránce. Tyto nahrazuji režimy *PC Relative*.

Tabulka 29-7.

instrukce	název	popis
000 rzi sss sss	LD	nahrávání, operand $\rightarrow A$
001 rzi sss sss	AD	sčítání: operand+(A,C) $\rightarrow (C,A)$
010 rzi sss sss	AN	logický součin: operand $\wedge A \rightarrow A$
011 rzi sss sss	OR	logický součet: operand $\vee A \rightarrow A$
100 rzi ddd ddd	ST	ukládání: $A \rightarrow \text{mem}[EA]$
101 0zi sss sss	DSZ	Decrement operand and Skip if Zero
101 lzi sss sss	ISZ	Increment operand and Skip if Zero
110 0pi aaa aaa	B, BAL	skok: (PC $\rightarrow L$;) EA $\rightarrow PC$
110 1pi aaa aaa	B, BAL	skok: (PC $\rightarrow L$;) EA $\rightarrow PC$
111 xxx xxx xxx	μ code	

* Tabulka je prvním pokusem sestavit instrukční sadu základních instrukcí. Vzhledem k tomu že na kódování základních instrukcí mám k dispozici jen 3 bity jsem omezen 8-mi kombinacemi. Vycházím tedy z instrukční sady PDP-8 kterou se snažím různě modifikovat.

Rozdílly oproti TX-0 jsou v počtu bitů kódujících základní instrukce. V tomto návrhu používám tři bity (TX0 používá jen 2).

Další zajímavou modifikací je instrukce skoku BL (*Branch and Link*). Právě použitím registru L jako jednorovňového hardwérového zásobníku umožňuje realizovat jednoduše podprogramy. Registr L je přístupný pro další práci a je tak možno programově realizovat víceúrovňový zásobník v operační paměti počítače. Tímto sjednocením instrukce skoku a volání podprogramu jsem oproti PDP-8 ušetřil jeden operační kód. Rovněž jsem se rozhodl vypustit instrukci ISZ PDP-8 a taktéž vstupně výstupní operaci IOT. ISZ lze realizovat dvojicí instrukcí, například

```

        AD #const
SKP Z

```

Ve skutečnosti je možností nečekaně více protože přičtením záporné konstanty realizujeme vlastně snižování hodnoty v A. Rovněž můžeme použít logickou operaci a provádět cyklus podle logické podmínky. Zkrátka můžeme si nakombinovat libovolnou operaci AD, AN, OR s libovolnou podmínkou v mikrooperaci SKP.

Devět bitů základní instrukce, následujících za kódem instrukce se dále dělí na dvě části. Adresní mód a číslo. Pro jednoduchost jsem se rozhodl **FIXME**:

29.3.2.1. Základní instrukce

Jak je vidět na obrázku 29-2 v základním tvaru instrukce jsou pro operační kód k dispozici pouze 3 bity. To nám dává 8 možných kombinací. Minimálně jedna musí být použita pro další tvari instrukcí jako jsou například „mikrokódované“ instrukce. Tím se prostor ještě zmenší. Při výběru jsem vycházel z instrukční sady PDP-8 kterou jsem se snažil modifikovat. PDP-8 má tyto instrukce:

```

0xxx  AND    ; operand  $\wedge$  AC  $\longrightarrow$  AC
1xxx  TAD    ; operand + AC  $\longrightarrow$  AC
2xxx  ISZ    ; increment operand and skip if zero
3xxx  DCA    ; AC  $\longrightarrow$  pamet', 0  $\longrightarrow$  AC
4xxx  JMS    ; skok do podprogramu
5xxx  JMP    ; skok
6xxx  IOT    ; I/O instrukce
7xxx  OPR    ; mikrokódované instrukce

```

Nejprve jsem zvážil odstranit instrukci IOT. Domnívám se, že se I/O zařízení dají připojit jako část paměti. Ušetřím tím právě jednu kombinaci pro jinou instrukci. Pokud bych chtěl instrukci IOT mohl bych ji realizovat v prostoru mikrokódovaných instrukcí například jako:

```

 11 10 9 8 7 6 5 4 3 2 1 0
+---+---+---+---+---+---+---+---+---+---+---+---+
| 1 1 1 | IOT | DEV | FCE | "mikrokódované" instrukce
+---+---+---+---+---+---+---+---+---+---+---+

```

Oproti PDP-8 snížím množství dostupných zařízení z 64 na 8.

Další úsporou v instrukční sadě je sjednotit instrukce JMS a JMP do jedné instrukce BAL. Tato instrukce je modelována podle BAL instrukce například z procesoru MIPS. Jedná se o instrukci skoku která do speciálního registru L uloží adresu instrukce následující po instrukci skoku. Tedy návratovou adresu. Cíl skoku, je-li to procedura či funkce, je pak zodpovědný za uschování této adresy.

```

...
BAL FCE ; volání podprogramu
NOP     ;  $\leftarrow$  zde se pokračuje po návratu
...
FCE:    ; ulož L do zásobníku
...     ; proved' co je potřeba
FLOOP:  ; nějaká smyčka
BAL FLOOP
; obnov L ze zásobníku
; proved' skok na adresu v L

```


29.3.2.2. Adresní módy

Při výběru adresních módů jsem vycházel z návrhu adresních módů PDP-8. Procesor, který má jen jeden pracovní registr, v našem případě střadač A, potřebuje nedostatek registrů nějak kompenzovat. PDP-8 tak činí adresním módem s přístupem k nulté stránce paměti a s přístupem k aktuální stránce paměti. Tím se počet míst která jsou procesoru snadno dostupná a dají se považovat v určitém smyslu za registry podstatně zvýší. V případě PDP-8 je to 128 slov v nulté stránce a 128 slov v aktuální stránce.

Adresní mód nulté stránky mi plně vyhovuje, je to něco co se objevuje například i v procesorech 6502 a 6800. Adresní mód přístupu k aktuální stránce je mi značně proti srsti. Hlavní námitku kterou proti němu mám lze ukázat na jednoduché věci. V následujícím kódu, který vznikl například po přidání instrukce nebo jiné změně která posunula instrukce TAD a AND, se instrukce AND již odkazuje na úplně jinou adresu než původně. Je to proto že se posunula do další stránky.

```

1016:  VAR1:  DW 6
1018:  VAR3:  DW 0
...
1176:          CLA CLC
1177:          TAD 16
           ; Změna aktuální stránky
1200:          AND 18
...
```

Kód pro PDP-8 tedy není imunní vůči posouvání v paměti. Znamená to že se význam instrukce mění, podle toho kde se v paměti nachází. Víím že v případě PDP-8 se jedná o obrovské zjednodušení konstrukce. Dnes použiji místo tohoto adresního módu mód jiný, relativní k hodnotě čítače instrukcí. Toto řešení je hezčí pro programátora ale klade větší nároky na konstrukci procesoru a tak se vůbec nedivím že jej v době konstruování PDP-8 nepoužili. Asi by výrazně prodražil konstrukci.

Pokud budu ochoten obětovat určité zvýšení komplexity obvodu generování adresy a přístupu k operandu, dostanu se k následujícím adresním módům.

Obrázek 29-3. Pole bitů adresního módu počítače SOC-8

```

      8      7      6
+---+---+---+
| R | I | Z |
+---+---+---+
```

Bity v tomto poli mají následující význam.

Tabulka 29-8. Adresní módy instrukcí SOC-8

bit	sym- bol	hodnoty	popis
8	R	0-A, 1-B	vybírám jeden ze dvou střadačů A,B u instrukcí které pracují se střadači
7	I	1-Indirect	Pokud je tento bit nastaven, je použito nepřímé adresování. Na efektivní adresu operandu EA je použita operace: EA=mem[EA].
6	Z	0-Rel, 1-ZP	Má-li bit Z hodnotu 1 jsou bity b ₆ -b ₀ adresou slova v nulté stránce paměti. Má-li hodnotu nula jsou tyto bity znaménkově rozšířeny a přičteny k PC.

Následující tabulka uvádí všechny uvažované adresní módy. Pouze některé budou použity a jsou jim přiděleny

kombinace bitů.

Tabulka 29-9. Adresní módy instrukcí SOC-8

mód	zápis	název	popis
000	n	Zero Page	EA=0...0b ₅ ...b ₀ ; OP=mem[EA]
001	(n)	Zero Page indirect	EA=0...0b ₅ ...b ₀ ; EA=mem[EA]; OP=mem[EA]
	(n)+	ZP Post Increment	EA=0...0b ₅ ...b ₀ ; EA=mem[EA]; OP=mem[EA]; EA=EA+1; mem[0...0b ₅ ...b ₀]=EA
	-(n)	ZP Pre Decrement	EA=0...0b ₅ ...b ₀ ; EA=mem[EA]; EA=EA-1; mem[0...0b ₅ ...b ₀]=EA; OP=mem[EA]
	#0	přímá konstanta	OP = SignExtended(b ₅ ...b ₀)
010	P,n	P relative	EA=SignExtended(b ₅ ...b ₀); EA=EA+PC; if EA=0 then PC=PC+1; OP=mem[EA]
010 0...0	#nn	Long literal	Jiný zápis pro zvláštní případ P relative s posunutím 0.
011	P,(n)	P Relative Indirect	EA=SignExtended(b ₅ ...b ₀); EA=EA+PC; if EA=0 then PC=PC+1; EA=mem[EA]; OP=mem[EA]
011 0...0	P,(nn)	P Relative Indirect	EA=SignExtended(b ₅ ...b ₀); EA=EA+PC; if EA=0 then PC=PC+1; EA=mem[EA]; OP=mem[EA]

Použití jen dvou bitů pro specifikování adresního módu, jak to má PDP-8 (I a Z), by příliš omezilo množství adresních módů.

název	zápis	význam
Zero Page	n	EA=(0,...0,b ₅ ...b ₀); OP=mem[EA]
Zero Page Indirect	(n)	EA=(0,...0,b ₅ ...b ₀); EA=mem[EA]; OP=mem[EA]
P Relative	P,n	EA=SignExtended(b ₅ ...b ₀); EA=EA+P; if EA=0 then PC=PC+1; OP=mem[EA]

Mód *P Relative* má jednu zvláštnost, která je patrná na následující ukázce.

```
1234: 1x00      AD P,0
1235: 2345      DW 2345
```

Co instrukce AD na adrese 1234 provádí? V průběhu načítání instrukce z adresy 1234 byla zvětšena hodnota v registru P o jedničku, aby P ukazoval na následující instrukci. P tedy obsahuje hodnotu 1235. Co se děje dále? Při výpočtu efektivní adresy EA z výrazu P,0 spočte EA=P+0=1235. Poté se z paměti přečte operand OP=mem[1235] což je 2345. Tímto způsobem můžeme snadno realizovat mód s velkými přímými literály. Samozřejmě že je to ideální mód pro skoky přes celou paměť a pro volání podprogramů. Například volání podprogramu na adrese 620 realizujeme instrukcí:

```
1252: 6x00      BL P,0
1253: 0620      DW 0620
```

Zvláštnost na kterou je třeba upozornit je to, že po ukončení instrukce ukazuje P na slovo 1235, v případě skoku na 1253. Na těchto adresách je očekávána další instrukce. Proto je v tomto zvláštním případě, kdy je čtena hodnota relativně k P s posunutím 0, provedena korekce P zvětšením o jedničku.

Tabulku adresních módů tedy můžeme rozšířit o další řádek, což není nic jiného než jiný zápis pro speciální

variantu jiného adresního módu.

název	zápis	význam
Long Literal	#nn	EA=SignExtended(bs...b ₀); EA=EA+P; if EA=0 then PC=PC+1; OP=mem[EA]

Dalším adresním módem, který považuji za velmi užitečný, je malý přímý literál. Tedy hodnota přímo zapsaná do instrukce. Dolních 6 bitů se v tomto módu interpretují jako malé číslo kterému se rozšíří znaménko do zbývajících bitů. Můžeme tedy do instrukce přímo zapsat malá čísla v rozsahu od -32_{DEC} do $+31_{\text{DEC}}$. Ačkoli bychom mohli použít právě zavedený mód *Long Literal*, vložení čísla přímo do instrukce zkrátí programový kód a rovněž jej zrychlí.

název	zápis	význam
Short Literal	#n	OP=SignExtended(bs...b ₀); EA není definováno

Uvedené adresní módy, které pokrývají 4 základní kombinace se dají vyjádřit použitím pouhých dvou bitů. Mám tedy na vybranou, zdali se u těchto 4 (5) adresních módů zastavím, nebo zdali využiji třetí bit k jejich rozšíření.

Jedním z rozšíření adresních módů, které zavedli u PDP-8 je nepřímé adresování se zvětšujícím ukazatele před použitím. Tato varianta je zavedena pomocí zvláštních adres v adresním prostoru. Jedná se o adresy 0010 až 0017. Kdykoliv je čten obsah paměti prostřednictvím těchto adres, jsou tyto zvětšeny o 1. Je velmi lákavé zavést další dva adresní módy, a to *Indirect Pre Decrement* a *Indirect Post Increment*. Tyto módy by dovolovali velmi snadno realizovat zásobníky a práci s ukazateli. Na druhou stranu je možno použít přístup PDP-8 a realizovat oba módy vázáním na speciální adresy v paměti. Například mne napadá následující.

adresa	význam
0010...0017	Speciální adresy. Při použití těchto adres se chovají módy Zero Page a Zero Page Indirect normálně.
0020...0027	Při odkazu na tyto adresy se ve skutečnosti přistupuje na adresy 0010...0017 s tím, že po ukončení čtení se obsah zvětší o 1.
0030...0037	Při odkazu na tyto adresy se ve skutečnosti přistupuje na adresy 0010...0017 s tím, že před čtením se obsah zmenší o 1.

Uvedený přístup, jako jeden z možných, dovolí realizovat dodatečné adresní módy bez změny v procesoru navázáním speciálních vlastností na adresy v paměti. Výhodou je že není třeba měnit instrukční sadu a tím pádem procesor. Nevýhodou je že se sníží počet dostupných slov paměti v rozsahu nulté stránky. V tomto konkrétním případě dojde ke snížení počtu slov o 16_{DEC} z celkových 64_{DEC} což je nezanedbatelné číslo. Samozřejmě že je možné provést další optimalizaci, například místo adres 0010...0017 s navázáním speciálních funkcí na následující 0020...0037, Určit jako speciální jen adresy 0020...0027 s tím že při čtení dochází k Post Inkrementaci a s namapováním adresy 0030...0037 na 0020...0027 s tím že při čtení dochází k Pre Dekrementaci.

Oba dva módy, tedy *Indirect Post Increment* a *Indirect Pre Decrement* považuji za natolik důležité že je musím zavést. Otázkou zůstává jak. Volný bit v trojici bitů adresního módu totiž můžu využít tak, že jej přiřadím k datovému poli dolních 6-ti bitů, které tak prodloužím na 7. V tom okamžiku si zvětším velikost nulté stránky na dvojnásobek tedy na 128_{DEC} slov. To je velmi lákavé. Nebo můžu udělat úplně jinou věc. Rozšířit počet střadačů v procesoru z jednoho na dva. Toto by opět zvedlo možnosti v programování.

Pokud se omezím jen na zásobníky, lze ze specifickými registry zacházet tak, že je určen jeden rozsah, například od 0010 do 0017. Veškeré přístupy k těmto buňkám paměti se chovají normálně, speciální chování je navázáno jen na nepřímý adresní mód (n) *Zero Page Indirect*. Pokud zapisujeme do adresy buňkou určené, dojde před zápisem k dekrementaci hodnoty v buňce. Pokud naopak čteme, dojde po čtení k inkrementaci adresy v buňce. Tímto způsobem se nesnížíme počet použitelných buňek v nulté stránce paměti.

Domnívám se však, že pokud se mám podržet původní myšlenky jednoduchosti designu, kterou jsem dost narušil již zavedením módu *P Relative*, že nemá smysl zavádět další registr.

Pro tuto chvíli zanechám úvah na téma třetího bitu v adresním módu a tento bit ponechám „nevyužitý“ a volný pro případná další rozšíření počítače.

Tabulka 29-10. Adresní módy

000			
001			
010			
011			
100			
101			
110			
111		PC Relative Indirect	EA=SignExtended(b ₅ ...b ₀); EA=EA+PC; if EA=0 then PC=PC+1; EA=mem[EA]; OP=mem[EA]

Zero Page

Zero Page Indirect

Zero Page Indirect Pre Decrement

Zero Page Indirect Post Increment

P Relative

P Relative Indirect

Small Literal

Long Literal

29.3.2.3. Mikrokódové instrukce

Pokud mají bity operačního kódu instrukce hodnotu 111, jedná se o tzv mikrokódovou instrukci. Je to obdoba instrukce OPR monipočítače PDP-8.

Obrázek 29-4. Formát instrukce μ code SOC-8

11	10	9	8	7	6	5	4	3	2	1	0				
+---+---+---+---+---+---+---+---+---+---+---+---+															
	1	1	1		GROUP			μ code							
+---+---+---+---+---+---+---+---+---+---+---+---+															
	1	1	1		0	0	0		CLC	CPC	CLA	CPA	IAC	DAC	Clear and Complement (IACVDAC)
+---+---+---+---+---+---+---+---+---+---+---+---+															
	1	1	1		0	0	1		CLC	CPC	CLA	R/L	A/L	DBL	Rotate
+---+---+---+---+---+---+---+---+---+---+---+---+															
	1	1	1		0	1	0		Condition			!?	?!DBL	Skip Condition[,Dbl]	
+---+---+---+---+---+---+---+---+---+---+---+---+															
	1	1	1		0	1	1		Source		Destination		Move Rs→Rd R one of A, B, P, L, PSW, S, T		
+---+---+---+---+---+---+---+---+---+---+---+---+															
Move L→PC ≡ Return															
Zatím nevyužité kódy															
+---+---+---+---+---+---+---+---+---+---+---+---+															
	1	1	1		1	0	0		unused						
+---+---+---+---+---+---+---+---+---+---+---+---+															
	1	1	1		1	0	1		unused						
+---+---+---+---+---+---+---+---+---+---+---+---+															
Privilegované operace															
+---+---+---+---+---+---+---+---+---+---+---+---+															
	1	1	1		1	1	0		DEVICE		FUNCTION		IOT		
+---+---+---+---+---+---+---+---+---+---+---+---+															
	1	1	1		1	1	1		0	R/W	S/U	PAGE REG.		MMU (max 8 reg na režim)	
+---+---+---+---+---+---+---+---+---+---+---+---+															
	1	1	1		1	1	1		1	Processor Control					
+---+---+---+---+---+---+---+---+---+---+---+---+															
	1	1	1		1	1	1		1	CPU - Enable/Disable part of the CPU					
+---+---+---+---+---+---+---+---+---+---+---+---+															
Stack operations															
+---+---+---+---+---+---+---+---+---+---+---+---+															
								:P:P:A/L:S/T							
+---+---+---+---+---+---+---+---+---+---+---+---+															

Obrázek 29-5. Volání podprogramu

```

BAL PProg
...

PProg:  PUSH L,S
        ...
        POP  L,S
        Move L,PC ; Return

```

Přesně v intencích dělení instrukčního slova po trojcích bitů přiřadím první další skupině bitů význam skupiny (GROUP) tyto bity určují o jaký mikrokód se jedná. Máme následující skupiny:

skupina	název	význam
000	Clear and Complement	Mazání a nastavování bitů C a registru A. CLA, CLC, CMA, CMC, IAC, ...

skupina	název	význam
001	ROT	Rotace a posuvy. LEFT/RIGHT, ARIT/LOGIC, DOUBLE, . . . Instrukce: ROR, ROL, ASL, ASR, . . .
010	Skip	Skupina podmíněných přeskoků. Bity mikrokódu specifikují podmínku přeskočení následující instrukce.
011	Move	Přesuny mezi registry. Je možno provést přesun mezi libovolnými dvěma registry. (A, B, P, PSW, L, S, T, U, . . .)
100		
101		
110		
111	CPU+MMU	Řízení procesoru a MMU

Tabulka 29-11. Některé instrukce SOC-8

code	mnemo	popis
7000	NOP	no operation
7001	DAC	decrement AC
7002	IAC	increment AC
7003		unused/prohibited

29.3.2.3.1. Processor Control and MMU

Instrukce začínající bity 111 111 jsou privilegované instrukce pro řízení procesoru a práci s jednotkou MMU.

Tabulka 29-12.

kód	blok	popis
111 111 000 xxx	MMU	Load DPage with x
111 111 001 xxx		Load IPage with x
111 111 010 000		Load Dpage to AC
111 111 010 001		Load Ipage to AC
111 111 010 010		Load Dpage with AC
111 111 010 011		Load Ipage with AC
111 111 100 000	CPU	Interrupt Enable
111 111 100 001		Interrupt Disable
111 111 100 010		
111 111 100 011		
111 111 100 100		
111 111 100 101		
111 111 100 110		
111 111 100 111		CPU Halt

29.3.3. Rozšíření paměti (MMU)

Protože 4KiW operační paměti není zrovna mnoho, je možné SOC8 rozšířit pomocí jednotky správy paměti schopnost adresovat další RAM.

Jednotka MMU má 8 stránkových registrů. Pro každý režim procesoru. Tyto registry jsou široké 12 bitů. Tyto registry jsou adresovány dvěma nejvyššími bity adresy na adresní sběrnici a dalším signálem I/D který rozlišujeme mezi čtením programu a čtením a zápisem dat. Tento signál efektivně rozšiřuje adresu ze 12 na 13 bitů a odděluje prostor programu od prostoru dat.

Takže tyto 3 bity adresují 8 stránkových registrů. Celý adresní prostor počítače je tak rozdělen na 1KiW stránky.

```

0000 +-----+
      | PAGE 0 | 1777
2000 +-----+
      | PAGE 1 | 3777
4000 +-----+
      | PAGE 2 | 5777
6000 +-----+
      | PAGE 3 | 7777
      +-----+

```

Každý stránkový registr obsahuje bity ochrany paměti a bity rozšířené adresy.

```

 11 10  9  8  7  6  5  4  3  2  1  0
+---+---+---+---+---+---+---+---+---+---+---+
|R/O| ? | ? |           :PAGE ADDRESS   :      |
+---+---+---+---+---+---+---+---+---+---+---+

```

Na uvedeném návrhu je pro adresu stránky vyhrazeno 9 bitů. To umožňuje adresovat celkem 512 stránek o velikosti 1KiW což je 512KiW operační paměti. To je na 12-ti bitový počítač docela dost.

Nejvyšší 3 bity stránkového registru pak specifikují vlastnosti stránky. V bitu b_{11} je informace zdali je stránka jen pro čtení. Pokud ano, je při pokusu o zápis do stránky vyvolána výjimka ochrany paměti. Zbývající dva bity zatím nemají přidělen žádný význam.

* Možné použití je například takové že do bitu b_{10} jednotka MMU zapíše jedna při každém přístupu na stránku. Je možné pak sledovat, zdali proces jenž má stránku k dispozici s ní někdy pracoval.

Operační systém počítače poskytuje procesům službu mapování stránek. Pokud proces o takovou službu požádá operační systém pro něj udržuje seznam přidělených stránek a umožňuje mu tyto přepínat.

Protože minimální počet režimů procesoru je v případě rozšířeného procesoru dva, t.j. uživatelský a systémový. Je celkový počet registrů MMU 16.

Poznámka: Rozšíření počtu stran paměti v adresním prostoru ze 4 na 8 by se mi líbilo. Celkové snížení adresovatelné paměti nevidím jako problém. Problém je, přepínání úloh. Čím menší velikost stránky a větší počet stránek v adresním prostoru procesoru, tím více je stránkových registrů v MMU. A to znamená že při přechodu na jinou úlohu je třeba vyměnit více informací.

29.4. SC-4

*

SC-4 je 12-ti bitový počítač. Je konstruován jako jednoduchý minipočítač, který je možno rozšířit o další funkční moduly.

Většina instrukcí SC-4 používá jedno slovo, které je rozděleno na 6 bitů operačního kódu a modifikátoru adresy a zbylých 6 bitů obsahuje data nebo část adresy. Mimo to existují instrukce jenž používají dvě slova. Takové instrukce obsahují ve druhém slově 12-ti bitovou adresu, nebo 12-ti bitové datové slovo. Základní instrukční formáty tedy vypadají takto:

Obrázek 29-6. Formáty instrukcí počítače SC-4

Základní formáty instrukcí **SC-4**.

Základní formát instrukčního slova	11	8	7	6	5	0		
	OPCODE		I	Z	DISP			
Základní instrukce s daty	11	8	7	6	5	0	11 data ne	
	OPCODE		I	1	0 0 0 0 0 0			
Dlouhá instrukce	11	OPCODE			6	5	0 11 podmínka adres	
Mikroinstrukce	11	8	7	μCode				0
	OPCODE							

Základní formát instrukčního slova je rozdělen na tři části. Směrem od nejvýznamnějších k nejméně významným bitům slova to jsou:

- operační kód o velikosti 4 bity
- modifikátor adresy sestávající z bitu I pro nepřímé adresování a z bitu Z pro adresování nulté stránky paměti
- adresy nebo posunutí o velikosti 6 bitů

Bit I určuje zdali je použito přímé adresování (hodnota 0), nebo nepřímé adresování (hodnota 1).

Bit Z určuje, jestli je následujících 6 bitů adresou slova v nulté stránce paměti (hodnota 0), nebo jestli se jedná o posunutí (DISP) relativně k hodnotě ukazatele instrukcí PC (hodnota 1).

Obrázek 29-7. Instrukce počítače SC-4

Návrh instrukcí SC-4:

	11	0 0 0 0 0 0				6	5	OPCODE							0		
MOV	11	0 0 0 0 0 1				6	5	Sreg			3	2	Dreg		0		
PUSH	11	0 0 0 0 1 0				6	5	AC	BC	SP	TP	PSW	PC				
POP	11	0 0 0 0 1 1				6	5	AC	BC	SP	TP	PSW	PC				
ROT	11	0 0 0 1 0 0				6	5	A/B	S/R	L/R	COUNT			0			
SVC	11	0 0 0 1 0 1				6	5	N							0		
I06	11	0 0 0 1 1 0				6	5	DISP							0		
LDC	11	0 0 0 1 1 1				6	5	DATA							0		
LDA	11	0 0 1 0		8	I	7	Z	6	5	DISP					0		
STA	11	0 0 1 1		8	I	7	Z	6	5	DISP					0		
ADD	11	0 1 0 0		8	I	7	Z	6	5	DISP					0		
SUB	11	0 1 0 1		8	I	7	Z	6	5	DISP					0		
AND	11	0 1 1 0		8	I	7	Z	6	5	DISP					0		
OR	11	0 1 1 1		8	I	7	Z	6	5	DISP					0		
I40	11	1 0 0 0		8	I	7	Z	6	5	DISP					0		
I44	11	1 0 0 1		8	I	7	Z	6	5	DISP					0		
SKIP	11	1 0 1 0		8	OFFSET		7	6	5	COND					0		
JMP	11	1 0 1 1		8	I	7	Z	6	5	DISP					0		
LJMP	11	1 1 0 0		8	I	7	Z	6	5	COND					0		
CALL	11	1 1 0 1		8	I	7	Z	6	5	COND					0		
OPR0	11	1 1 1 0		8	CLA	7	CLL	6	5	CPA	CPL	3	ROT		1	IA	0
OPR1	11	1 1 1 1		8	CLB	7	CPB	6	5	4	3	2	1	0			

Obrázek 29-8. Registry počítače SC-4

Návrh registrů SC-4.

A	ACCUMULATOR																0					
B	B REGISTER																0					
.																						
SP	STACK POINTER																0					
PC	PROGRAM COUNTER																0					
PSW	STATUS REGISTER																0					
.																						
	11	SUP		10	9			8	7	I	6	5	4		P	3	N	2	Z	1	L	0

29.4.1. MMU

*

Počítač SC-4 má v základní výbavě možnost adresovat 4096 slov operační paměti. Protože pro rozsáhlejší úlohy je to málo a rovněž se počítá s multiuživatelským prostředím či spouštěním více programů současně, je pro něj připravena jednotka správy paměti. Tato jednotka se instaluje mezi procesor a paměť. Jednotka kombinuje informaci na adresových vodičích A11 a A10 a informaci o právě probíhající instrukci s informací ve svých registrech a sestavuje fyzickou adresu jenž může obsahovat až 19 bitů. Maximální velikost připojené paměti je tedy 512K slov, což je více než dostatečné i pro multiuživatelský operační systém umožňující provozovat mnoho programů současně. V praktickém nasazení se počítá s mnohem menšími velikostmi operační paměti od 16K do 64K slov.

Nejvyšší dva bity adresy určují který ze 4 bloků paměti je adresován. Další informace která se používá k rozhodování je informace o právě probíhající instrukci. Jednotka MMU ví, jestli procesor čte instrukci, nebo jestli čte či zapisuje data. Toto umožňuje oddělit adresní prostor programu a dat. Efektivně má tedy jednotka k dispozici tři bity I/D, A11 a A10. Tyto určují který z osmi registrů se použije při tvorbě adresy. Obsah pole PAGE které má 9 bitů se připojí ke zbývajícím 10-ti bitům adresy a vytvoří tak 19-ti bitovou fyzickou adresu.

Obrázek 29-9. MMU počítače SC-4

Návrh MMU pro SC-4. Prostor adresovatelný CPU

•	BLOK 0	0000	1777
•	BLOK 1	2000	3777
•	BLOK 2	4000	5777
•	BLOK 3	6000	7777

Registry MMU

MMUCTL	EN 11	10	9	6	5	0
MMUB0I	11	HIT 10	INT 9	8	PAGE 0	
MMUB1I	11	HIT 10	INT 9	8	PAGE 0	
MMUB2I	11	HIT 10	INT 9	8	PAGE 0	
MMUB3I	11	HIT 10	INT 9	8	PAGE 0	
MMUB0D	RO 11	HIT 10	INT 9	8	PAGE 0	
MMUB1D	RO 11	HIT 10	INT 9	8	PAGE 0	
MMUB2D	RO 11	HIT 10	INT 9	8	PAGE 0	
MMUB3D	RO 11	HIT 10	INT 9	8	PAGE 0	

29.5. SOC-240

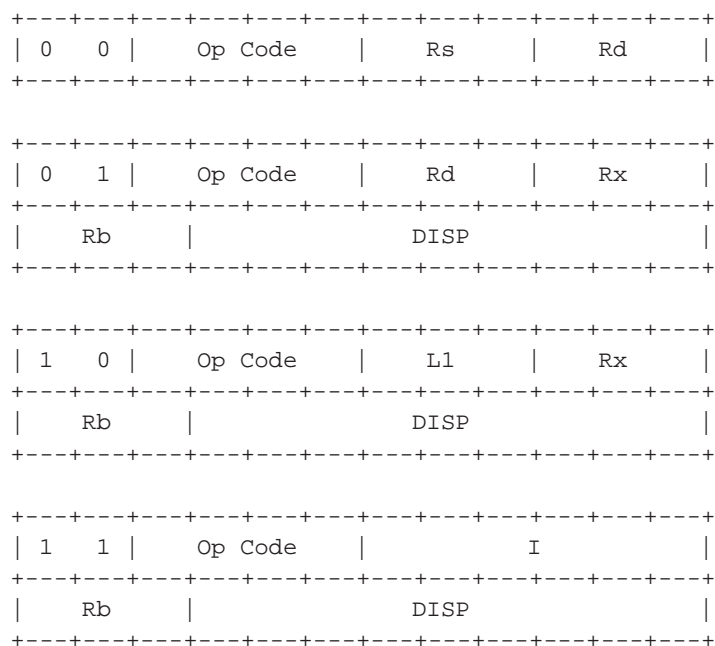
*

24 bitový procesor/počítač/architektura.

Při návrhu tohoto procesoru vycházím ideově z architektury 65.

Výchozí požadavky, pracovní verze:

- Nejmenší adresovatelná velikost paměti 6, 9 nebo 12 bitů.
- Registry CPU délky 18 nebo 24 bitů. Větší registry by konstrukci poněkud prodražili.
- Počet univerzálních registrů 8 nebo 16.
- Instrukce s variabilní délkou v násobcích 9 nebo 12 bitů.
- Schopnost zpracovávat textové informace s 6, 9 nebo 12-ti bity na znak.
- Velikost adresovatelného prostoru dána 24 bitovou adresou. Tedy 16M základních adresovatelných jednotek.
-
-

Obrázek 29-10. Formát instrukčního slova SOC-120**Tabulka 29-13. Plánované instrukce**

mnemo	formáty	popis
ADD		
SUB		
AND	RR	logický bitový součin
OR	RR	logický bitový součet
XOR	RR	logický bitový exkluzivní součet
NOT	RR	bitová negace
ROT	RR	Rotace
CMP	RR	porovnání dvou hodnot a nastavení příznaků
MOVE	RR	přesun mezi registry
LOAD		
STORE		
LA		load address
BRANCH		skok na adresu
BAL		volání podprogramu

Tabulka 29-14. Návrh prostoru instrukčních kódů:

code	mnemo	format	operands	name
00				
01	SVC	I	nn	Supervisor Call

code	mnemo	format	operands	name
02				
03				
04				
05				
06				
07				
10				
11				
12				
13				
14				
15				
16				
17				
20				
21				
22				
23				
24				
23				
...				
76				
77				

29.6. SOC-168

Jednoduchý 12-ti bitový procesor/počítač inspirovaný architekturou 12-ti bitových počítačů jako jsou CDC-160 a PDP-8. Procesor je projektován jako experiment v simulaci procesorů. Elektronický návrh není předmětem.

Předpoklady, výchozí podmínky:

- 12-ti bitová architektura
- šířka datového/instrukčního slova 12 bitů
- šířka adresy 12 bitů

29.6.1. Registry

Procesor má řadu registrů. Nejdříve uvedu souhrnnou tabulku a pak je podrobně popíši.

Tabulka 29-15. Registry procesoru SOC-168

AC1	První akumulátor
AC2	Druhý akumulátor
PC	Čítač instrukcí. Ukazatel na instrukci.
LINK	Návratová adresa z podprogramu.

Rozhodl jsem se že v procesoru implementuji dva kumulátory. Očekávám že mi to dá větší prostor k vyjádření při programování. Jsou označeny symbolickými jmény AC1 a AC2.

Registr PC, tedy ukazatel na instrukci, vždy ukazuje na další instrukci. K jeho zvětšení (o 1) dochází v průběhu operace FETCH, tedy v průběhu nahrávání instrukce z paměti. V okamžiku vykonávání instrukce ukazuje již na instrukci následující.

29.6.2. Úvahy, jimiž se konstrukce SOC168 řídila

Základní představou bylo navrhnout jednoduchý, procesor inspirovaný procesory/počítači CDC 160 a 63.1.2. Výchozími požadavky je, aby byl použitelný alespoň v takovém rozsahu jako zmiňované dva počítače. Dalším požadavkem bylo vyhnout se některým aspektům návrhu PDP-8 a CDC-160 které mi velmi vadily. Začnu tedy postupný sled úvah jenž mají vyvrcholit konstrukcí simulátoru.

Jak jsem již začal, inspirovali mne 12-ti bitové počítače. Oba dva mají v základní charakteristice 12-ti bitové datové slovo a schopnost adresace 4KiW operační paměti 12-ti bitovou adresou. V prvním plánu neuvažuji o nějakém rozšíření operační paměti o mechanismus paměťových bank.

Stejně jako předlohy z nichž vycházím, i SOC168 bude mít minimum registrů. Tyto jsou:

Registr	Obsah	Popis
P	Program Counter	Ukazatel instrukcí programu. Ukazuje vždy na instrukci která je následující po právě zpracovávané instrukci.
A	Accumulator	Střadač. Cíl všech aritmeticko-logických instrukcí.
L	Link Address	Speciální registr kam je ukládána návratová adresa při volání podprogramu.
PSW	Program Status Word	Registr příznaků a dalších nastavení.

Nedostatek registrů chci kompenzovat mechanismem přístupu k takzvané Zero Page. Oblasti na začátku paměti. Tato paměť je dostupná tak snadno že ji lze považovat za registry.

A postupně se dostávám k instrukcím. Velmi se mi líbí jednoduchost instrukcí počítače PDP-8 jehož instrukční slovo má velmi jednoduchou strukturu.

```

+-----+-----+-----+
|  op   | mode |      addr      |
+-----+-----+-----+
```

Počet bitů pole `op` je v případě PDP-8 3 v případě CDC-160 4. To u PDP-8 dává prostor pro 8 instrukcí. PDP-8 jich má jen 7. Kód osmé instrukce má dolišný význam. Z těchto 7 instrukcí jsou 2 aritmeticko/logické (ADD a AND), 1 (STORE) ukládá obsah akumulátoru do paměti, 3 instrukce skoku (JSZ, JMS a JMP) a poslední je I/O instrukce IOT.

Minimalizace instrukčního souboru je velmi chytrá. Čím méně instrukcí, tím jednodušší realizace obvodů ALU a dekodéru instrukcí. Velmi hezké je vypuštění instrukce LOAD jenž se dá nahradit dvojicí instrukcí.

CLA / $0 \rightarrow AC$
TAD M / $A + mem(M) \rightarrow A$

Mohli bychom také požit jinou dvojici instrukcí, založenou nikoliv na sčítání ale na operaci AND.

STA / $7777 \rightarrow AC$
AND M / $A \text{ and } mem(M) \rightarrow AC$

Před dalšími úvahami se pokusím napsat seznam naprosto nezbytných instrukcí. PDP-8 má celkem 8 instrukčních kódů. Jsou to: AND, TAD, ISZ, DCA, JMS, JMP, IOT a OPR. Osobně se domnívám, že lze instrukce JMS a JMP sjednotit do jedné instrukce skoku a to BAL s následující sémantikou:

$PC \rightarrow LINK; \text{addr} \rightarrow PC$

Kód na který se skáče má pak možnost obsah registru LINK ignorovat, Nebo v případě že se jedná o proceduru jej uschovat.

```
BAL FCE
...
FCE:  LINK → AC
      ST STACK--
      # Kód podprogramu
LD ++STACK
AC → LINK
RET      # LINK → PC
```

Dalším možným zdrojem úspory v operačních kódech je změnit fungování I/O z instrukce IOT na práci s pamětí. Prostě zrušit kompletně IOT a nahradit ji periferiemi mapovanými do adresního prostoru.

Po všech změnách se pak můžeme dostat k následujícímu seznamu instrukcí.

opcode	mnemonic	popis
000	ADD	$\text{operand} + AC \rightarrow AC$
001	AND	$\text{operand} \wedge AC \rightarrow AC$
010	ST	$AC \rightarrow \text{memory}(\text{operand})$
011	BAL	$PC \rightarrow LINK; \text{operand} \rightarrow PC$
100	OR	
101	XOR	
110	MOP	Bitfield <u>textmu</u> code operation
111	CO	Coprocessor operation.

<u>textmu</u> OP		
opcode	mnemonic	popis
0xx xxx xxx	MO1	Rotate operations
10c ccc fff	MO2	Conditional operations
11x xxx xxx	MO3	Extended operations

MO0		
opcode	mnemonic	popis

MO0		
opcode	mnemonic	popis
000 000 000	NOP	No Operation
010 000 000	CLA	$0 \rightarrow A$
001 000 000	CLC	$0 \rightarrow C$
000 100 000	NOTA	$\neg A \rightarrow A$
000 010 000	NOTC	$\neg C \rightarrow C$
000 001 000	IA	$A+1 \rightarrow A$
000 000 100	ROR	Rotate CA Right
000 000 010	ROL	Rotate CA Left
000 000 001	T	Rotate Twice if ROR or ROL
000 000 001	BSW	Byte Swap A if not ROR or ROL

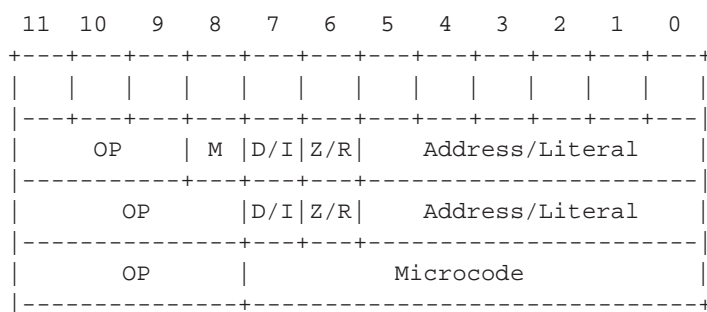
MO1		
opcode	mnemonic	popis
10c ccc fff	MO2	
Conditions		
100 000 fff		Never
100 001 fff		
100 010 fff		
100 011 fff		
100 100 fff		
100 101 fff		
100 110 fff		
100 111 fff		
101 000 fff		
101 001 fff		
101 010 fff		
101 011 fff		
101 100 fff		
101 101 fff		
101 110 fff		
101 111 fff		Always
Functions		
10c ccc 000		
10c ccc 001	S	Skip on condition
10c ccc 010		
10c ccc 011		
10c ccc 100	TLA	$L \leftarrow A$
10c ccc 101	TAL	$A \leftarrow L$
10c ccc 110	RET	$L \rightarrow P$
10c ccc 111	HLT	Halt

MO1		
opcode	mnemonic	popis
110 000 000	MO2	
110 000 001		
110 000 010		
110 000 011		
110 000 100		
100 000 101		
100 000 110		
100 000 111		
100 111 111	HLT	Halt processor.

Pokusím se nejprve navrhnout instrukční slovo. Zkusil jsem vyjít z PDP-8 s tím že:

- Adresní mód pracující s aktuální stránkou jsem nahradil módem relativním k PC
- V módu relativním k PC je jedna speciální výjimka, která umožňuje načíst 12 bitový Literál ze slova následujícího za instrukcí.
- Pokusně jsem rozšířil pole operačního kódu ze tří na 4 bity a to na úkor adresního pole.
-

Výsledek těchto změn je vidět na následujícím obrázku.



OP - 4-bit opcode

D/I - Direct/Indirect

Z/R - Zero(Direct) Page / PC Relative

M - modifier

Address/Literal - 6 bit address / 6 bit sign extended literal

Microcode - Bit filed microcode similar to PDP-9 OPR

* *EDITMARK*:

Instrukční slovo sestává z několika částí. Spodních 6 bitů slouží jako hodnota, adresa či offset. Záleží na jednotlivých instrukcích ale hodnota zde by měla být v rozsahu od -32 do 31 vyjádřená dvojkovým doplňkem. Horních 6 bitů tvoří operační kód. Tento je v případě některých instrukcí dále strukturován.

Tabulka 29-16. Tabulka Instrukcí SOC-168

Opcode	Mnemonic	Popis
000 0mm xxx xxx		unused
000 1mm xxx xxx		unused
001 0mm aaa aaa	LD	Load AC

Opcode	Mnemonic	Popis
001 1mm aaa aaa	ST	Store AC
010 0mm aaa aaa	AND	Value \wedge AC \longrightarrow AC
010 1mm aaa aaa	OR	Value \vee AC \longrightarrow AC
011 0mm aaa aaa	XOR	Value xor AC \longrightarrow AC
0111 mm aaaaaa		Value oper AC \longrightarrow AC
1000 mm aaaaaa	ADD	Value oper AC \longrightarrow AC
1001 mm aaaaaa	SUB	Value oper AC \longrightarrow AC
1010 mm aaaaaa		Value oper AC \longrightarrow AC
1011 mm aaaaaa		Value oper AC \longrightarrow AC
1100 mm aaaaaa		Value oper AC \longrightarrow AC
1101 mm aaaaaa		Value oper AC \longrightarrow AC
1110 mm aaaaaa		Value oper AC \longrightarrow AC
1111 mm aaaaaa		Value oper AC \longrightarrow AC
110 cc m		Jump and Link
111		nevyužito, rezerva

Zvažoval jsem různé adresní módy pro instrukce. Z úvah vyšlo že chci určité mód přímý (immediate) při kterém je 6 bitů adresního pole chápáno jako přímá hodnota v rozsahu od -32 do 31. Bit 5 slouží jako znaménko a pro účely instrukcí se replikuje do bitů 6 až 11 na 12-ti bitovou hodnotu.

Dále potřebuji adresovat hodnoty přímo v paměti. 6 bitů ovšem umožňuje adresovat jen 64 slov. Velmi se mi líbí relativní adresní mód kdy je těchto 6 bitů po rozšíření na 12 přičteno k čítači instrukcí a takto vzniklá adresa je použita k získání hodnoty z adresovaného slova.

Třetí možnost, Zero Page, "rozšiřuje" počet registrů o snadno dosažitelných 64 slov na začátku paměti na adresách 0000-0077.

Poslední, čtvrtá možnost, kombinace adresních bitů 11 není zatím implementována a je určena pro budoucí rozšíření.

Tabulka 29-17. Význam bitů pole adresního módu instrukce SOC-168

bity Z,I	název	význam
00	Direct Page	mem(DP+Address) \longrightarrow AC
01	Direct Page Indirect	mem(mem(DP+Address)) \longrightarrow AC
10	Immediate	sign extended(Literal) \longrightarrow AC
11 Addr \neq 0	PC Relative Indirect	mem(PC+Address) \longrightarrow AC
11 Addr=0	Immediate 12bit Literal in following Word	mem(PC) \longrightarrow AC; PC+1 \longrightarrow PC

Adresní mód PC Relative má jednu zvláštnost. Pokud je hodnota adresního pole 0, tedy když se odkazuje na slovo následující za právě vykonávanou instrukcí, dojde po načtení hodnoty z tohoto slova k automatické inkrementaci čítače instrukcí PC.

```
xxx r 01 000000    Instruction PC-Relative 0
vvvvvv vvvvvv      Přímá 12-ti bitová hodnota
```

V krátké tabulce instrukcí je jen jedna instrukce skoku. Tato se používá nejen jako instrukce skoku ale také jako

instrukce volání podprogramu. Funguje tak že obsah čítače instrukcí PC se uloží do speciálního registru LINK a poté je do PC nahrána cílová adresa. Podprogramy pak mají možnost hodnotu registru LINK uschovat, například na zásobník.

29.6.2.1. Strukturovaný soubor instrukcí

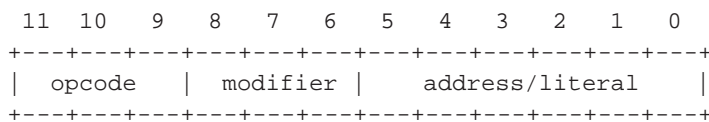
Strukturovaný soubor instrukcí mne napadl při prvním pokusu os sestavení instrukčního souboru pro SOC168. Práce sestává ze dvou kroků.

1. Definování polí
2. Rozmístění polí do několika typů instrukcí

Nejdříve samotná pole. Snažím se, aby byla tříbitová. V následujícím kroku se budu snažit aby byla zarovnána s trojemi bity v instrukčním slově tak, že každá okatlová číslice bude kódovat nějakou hodnotu pole. Tento přístup trochu zkomplikuje vytvoření instrukční sady ale na druhou stranu velmi usnadní „vhled“ programátora přímo do strojového kódu počítače.

29.6.2.1.1. Operační kód

Tato část instrukce kóduje samotnou instrukci a formát instrukce. Máme k dispozici 8 kódů které a 1 instrukční formát.



V operačním kódu jsou kódovány instrukce, jenž potřebují přímý přístup do paměti, nebo přímou literální hodnotu. Vzhledem k omezenému počtu možností je třeba vybírat uvážlivě. Proto proberu jednotlivé možnosti.

AD

Instrukce sčítání. Obvykle má procesor dvě základní instrukce sčítání a to ADD a ADC. První provádí sčítání $A+val \rightarrow C,A$ a druhá provádí $C,A+val \rightarrow C,A$. Liší se tedy tím, jestli je do sčítání zahrnut i byt přenosu C (Cary). Domnívám se že v praxi se lze omezit jen na jednu formu instrukce a to ADC. Vhodného efektu dosáhneme tím, jestli před sčítání C bit nastavíme, smažeme nebo necháme být.

```
# Implementace ADD
CLC
AD value
```

SB

Instrukce odečítání. Jako obvykle má procesor dvě základní instrukce pro odečítání a to SUB ($A-value \rightarrow C,A$) a SBC/SBB ($C,A-value \rightarrow C,A$). Vzhledem k nízkému počtu operačních kódů navrhuji tuto instrukci nazavádět. Lze ji totiž realizovat pomocí negace a sčítání.

```
# Implementace SBC/SBB
ST     TMP     # uschování -1 do TMP
CLA                     # 0 → A
AD     value
NA                     # -A → A
AD     TMP

# Implementace SBC/SBB immediate (C,A - immediate → C,A)
AD     -const
```

```
# Implementace SBC/SBB
ST      TMP
LD      value
NA      # -A → A
AD      TMP
```

AN

AND je jedna ze skupiny logických instrukcí (AND,OR,XOR,NOT).

OR

Instrukci OR je možné realizovat softwarově.

LD

Instrukce Load nahraje hodnotu do akumulátoru. ($\text{value} \rightarrow A$). Tuto instrukci je možné realizovat softwarově, stejně tak jako u počítače PDP-8 který ji nemá v základním souboru instrukcí.

```
# LD pomocí AN
CLA, NA      # 7777 → A (všechny bity 1)
AN      value

# LD pomocí AD
CLA      # 0 → A (všechny bity 0)
AD      value
```

ST

Instrukce Store uloží obsah akumulátoru do paměti.

BAL

Instrukce **BAL** je jediná instrukce skoku. Je udělána tak, aby pokryla všechny potřeby skoků. Používá standardní adresní možnosti jako všechny základní instrukce. Funkce vykonává postupně následující operace.

1. Operand \rightarrow MAR
2. P \rightarrow L
3. MA \rightarrow P

Uložení adresy instrukce následující za instrukcí skoku do registru L umožňuje realizovat podprogramy.

opcode	mnemo	popis
000	AD	
001	AN	
010	OR	
011	LD	
100		
101	ST	
110	BAL	
111	μ code	

Přemýšlím, které operační kód ještě do tabulky zahrnout. Jestli vložit OR,LD, jestli definovat instrukci pro „koprocesor“, atd.

Zbývajících 9 bitů instrukce od bitu 8 do bitu 0 určují operand. Horní tři bity 8-6 určují adresní mód a v závislosti na jejich hodnotě určují spodní bity přímou hodnotu, adresu nebo offset.

Mode	Name	Syntax	Effective Address	Operand Value
000	Literal	#n	???	Sign Extended [$b_5 \dots b_0$]
001	Zero Page Register	n	addr	mem[addr]
010	Zero Page Indirect	(n)	mem[addr]	mem[mem[addr]]
011	ZPI Post Increment	(n)+	mem[addr]; Inc(mem[addr])	mem[mem[addr]]
100	ZPI Pre Decrement	-(n)	Dec(mem[addr]); mem[addr]	mem[mem[addr]]
101	Register Indirect		A+offset	mem[A+offset]
110	PC Relative		PC+offset	mem[PC+offset]
111	PC Relative Indirect		mem[PC+offset]	mem[mem[PC+offset]]

```

000  n      ; ZP
001  (n)    ; ZP
010  (n)+   ; ZP
011  -(n)   ; ZP
100  #n
101  ((n))  ; ZP
110  ([n])  ; PC    mem[n+PC]
111  [n]    ; PC

```

Podrobný popis adresních módů.

Přímý krátký literál.

Obsah adresního pole, bity $b_5 \dots b_0$ je interpretován jako hodnota. Před použitím je znaménkový bit b_5 replikován do všech bitů vyššího bajtu. Tedy dojde k rozšíření znaménka. Tímto způsobem je možné přímo zapsat hodnoty v rozsahu od -32 do 31.

$$M \text{ undefined} \leftarrow \text{Sign Extend } b_5 \dots b_0$$

Zero Page Register

$$M \leftarrow 000000b_5 \dots b_0$$

$$O \leftarrow \text{mem}[M]$$

Zero Page Indirect

$$M \leftarrow 000000b_5 \dots b_0$$

$$O \leftarrow \text{mem}[M]$$

$$M \leftarrow O$$

$$O \leftarrow \text{mem}[M]$$

Zero Page Indirect Post Increment

$$M \leftarrow 000000b_5 \dots b_0$$

$$O \leftarrow \text{mem}[M]$$

$$M \leftarrow O$$

$$O \leftarrow \text{mem}[M]$$

$$\dots$$

Zero Page Indirect Pre Decrement

```

M ← 000000b5...b0
O ← mem[M]
O ← O-1
mem[M] ← O
M ← O
O ← mem[M]

...

M ←
O ← mem[M]

...

M ←
O ← mem[M]

...

M ←
O ← mem[M]

```

29.6.2.2. Tak trochu odlišný přístup

Tato sada instrukcí mne napadla po náhledu na instrukce PDP-11.

```

# Registry procesoru
R0
R1
R2
R3
R4
R5
R6
R7 - PC

```

Dvouoperandové instrukce.

```

+---+---+---+---+---+---+---+---+---+---+---+---+
| Opcode | Dest Reg | Addr Mode / Src Reg |
+---+---+---+---+---+---+---+---+---+---+---+---+

```

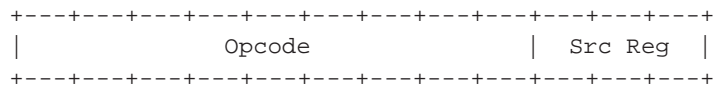
Jednooperandové instrukce.

```

+---+---+---+---+---+---+---+---+---+---+---+---+
| Opcode | Addr Mode / Src Reg |
+---+---+---+---+---+---+---+---+---+---+---+---+

```

Jednoregistrové instrukce.



Přehled instrukcí.

```

000 ddd sss sss  AD Rd,src
001 ddd sss sss  AN Rd,src
010 ddd sss sss  OR Rd,src
011 ddd sss sss  LD Rd,src
100 sss ddd ddd  ST Rs,dst
101 111 sss sss  BAL Rl,src    ;R7→Rl, src→R7

110 000 ddd ddd
110 001 ddd ddd
110 010 ddd ddd
.....
110 111 000 rrr RET Rr          ;Rr→P7
110 111 001 rrr
110 111 010 rrr
; Rotace a posuvy
111 000 nnn rrr ROR
111 001 nnn rrr ROL
111 010 nnn rrr ASR
111 011 nnn rrr ASL
; Podmíněné instrukce přeskoků.
111 100 ccc ofs SKIP ccc, ofs
111 101 rrr rrr XOR
111 110 rrr rrr
111 111 rrr rrr

```

29.6.2.3. RISC like Load/Store architektura

Pokus modelovat instrukční sadu podle procesorů třídy RISC.

```

00f fff ddd sss  ffff(Rs,Rd)→Rd
010 fff fff ddd  fffffff(Rd)→Rd
011 fff fff fff
100 ddd sss sss  Load Rd, src
101 sss ddd ddd  Store Rs, dst
110 LLL ddd ddd  BAL Rl, dst
111 ... ..

```

29.6.2.4. Jednoduchost návrhu

Nejdříve registry počítače.

- A — akumulátor
- P — Program Counter / Instruction Pointer

- L — Link Address (one level hardware stack)
- MAR — Memory Address Register, jediný registr umožňující adresovat operační paměť
- MDR — Memory Data Register, datový registr pro čtení a zápis do operační paměti.
-

Realizace adresního módu *Zero Page Indirect Post Increment*.

```

; Zero Page Indirect Post Increment
000000b5...b0 → MAR
mem[MAR] → MDR ; Získání obsahu slova v ZP.
MDR → MAM
MDR → MAR ; Získané slovo je adresa
mem[MAR] → MDR ; Přečteme data z uvedené adresy
MDR → OP2
; Post Increment
inc(MAM)
MAM → MDR
000000b5...b0 → MAR
MDR → mem(MAR)

```

29.7. SOC-22

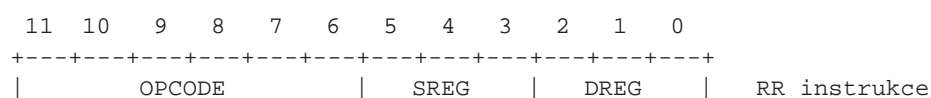
Předběžný návrh procesoru inspirovaného PDP-11.

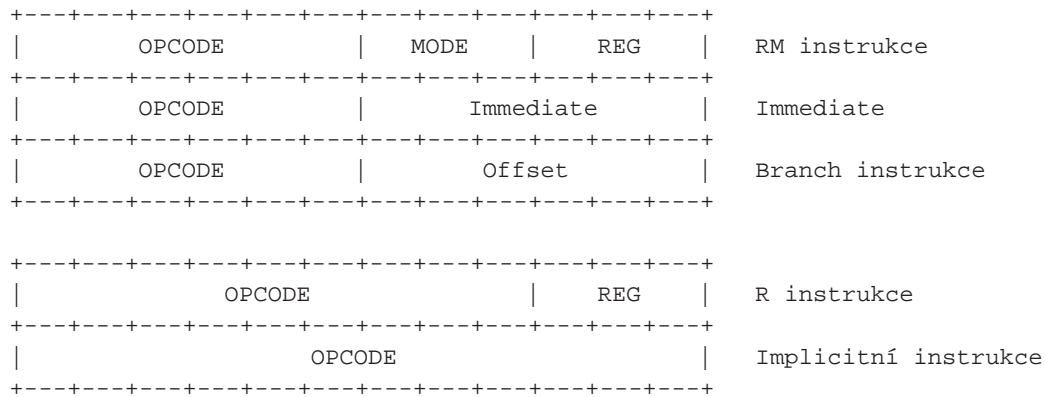
Procesor obsahuje cca 8 základních registrů

Tabulka 29-18. Registry SOC-22

regist	funkce	popis
R0	Accumulator	implicitní zdroj/cíl instrukcí které nespecifikují registr
R1		
R2		
R3		
R4		
R5	Frame Pointer	
R6	Stack Pointer	Ukazatel na vrchol zásobníku návratových adres
R7	Program Counter	čítač instrukcí, ukazatel na vykonávanou instrukci

Obrázek 29-11. Formáty instrukcí SOC-8





ADD R,R

ADC R,R

SUB R,R

SBC R,R

AND R,R

OR R,R

XOR R,R

INC R

DEC R

BR CC, offset

PUSH R

POP R

JMP Addr

JMP (Addr)

JMP R

CALL Addr

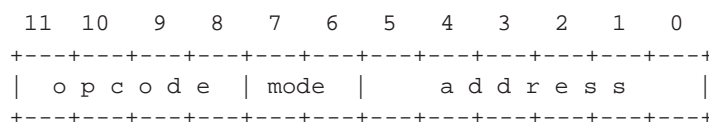
RET

29.8. Různé úvahy na téma 12 bitových CPU

29.8.1. Akumulátorový, jednoduchý procesor

Při svých úvahách jsem se zabýval i možností konstrukce procesoru s jedním akumulátorem. Opět inspirován jednoduchostí PDP-8 jsem přišel s následujícím instrukčním formátem.

Obrázek 29-12.



Stejně jako u PDP-8 je instrukční slovo rozděleno na tři části.

- operační kód
- adresní mód
- adresa

Oproti PDP-8 jsem se rozhodl kódovat adresní mód jiným způsobem. Místo přiřazení vlastností jednotlivým bitům, jsem se rozhodl očíslovat možné kombinace bitů a těm přiřadit adresní mód. To mi umožní zavést adresní módy které u PDP-8 nejsou možné, případně na které by bylo nutno rozšířit počet bitů specifikujících adresní mód.

Zvažoval jsem jaké módy zavést. Protože jednoakumulátorový procesor trpí nedostatkem registrů, je nutné tento nějak kompenzovat. V případě PDP-8 je tak činěno pomocí adresního módu nulté stránky paměti a aktuální stránky paměti. Rovněž je potřeba realizovat nepřímé adresování (*Indirect, Deferred*). Zcela určite jsem se rozhodl použít přímé a nepřímé adresování v nulté stránce.

Dále potřebuji možnost jak zavést jednoduše velké literály. U PDP-8 se musí tyto uložit buď to do nulté stránky v paměti, nebo do kódu v aktuální stránce. Pro jejich načtení se pak použije mód nulté nebo aktuální stránky. Nutlá stránka je malá a aktuální stránku nechci zavádět. Jednou z možností je použít následující buňku za instrukcí. Toto mě vede k adresnímu módu relativnímu k čítači instrukcí. Načtení dlouhého literálu z buňky za instrukcí je pak speciální případ adresování relativně k PC.

Tabulka 29-19. Adresní módy

mód	syntaxe	popis	název
00	n	$EA \equiv 0n$; $OP = \text{mem}[0n]$	Zero Page
01	(n)	$EA \equiv \text{mem}[0n]$; $OP = \text{mem}[EA]$	Zero Page Indirect
10	PC(n)	$EA \equiv PC+n$; $OP \equiv \text{mem}[EA]$	PC Relative
10	#nn	$EA \equiv PC+0$; $OP \equiv \text{mem}[PC+0]$; $PC++$	Long Literal
11		$OP = \text{SignEx}(b5-b0)$	Nevyužito, nebo přímý krátký literál.

Protože jsem ponechal pro kód instrukce 4 bity, je možno zavést větší počet instrukcí než má PDP-8. Chtěl bych zavést instrukce pro sčítání a odčítání, pro logické operace OR, AND, NOT, skoky, rotace.

Tabulka 29-20. Instrukce

kód	zápis	popis	poznámka
0000	ADD op	$A + op \longrightarrow A$	
0001	SUB op	$A - op \longrightarrow A$	
0010	LD op	$op \longrightarrow A$	
0011			
0100	OR op	$op \vee A \longrightarrow A$	
0101	AND op	$op \wedge A \longrightarrow A$	
0110	XOR op	$op \text{ xor } A \longrightarrow A$	
0111	NOT op	$\neg op \longrightarrow A$	
1000	INC op	$\text{mem}[EA]--$	
1001	DEC op	$\text{mem}[EA]++$	

Obrázek 29-15. Adresní módy

+-----+-----+		
	mod	syntax
+-----+-----+		
	000	R
	001	(R)
	010	(R) +
	011	-(R)
+-----+-----+		
	100	
	101	
	110	
	111	
+-----+-----+		

29.8.3. Architektura 12-ti bitová, bajtově orientovaná

Asi hloupý nápad. Architektura která má 6-ti bitový bajt a paměť organizovanou do 2_{12} 6-ti bitových bajtů.

Oprototi architektuře $2_{12} * 12$ má taková architektura v podstatě poloviční paměť. Komunikace po 6-ti bitové sběrnici bude mít taky poloviční výkon.

Ale instrukční sada by mohla být velmi, velmi hezká pokud by se prealizovala po bajtech, stejně jako u IBM 360 a DEC VAX. Tedy první bajt je operační kód. Podle prvních dvou bitů operačního kódu se pozná kolik formát a tím pádem kolik dalších bajtů, popisujících operandy se bude načítat. Jednotlivé operandy pak mohou být specifikovány jedním nebo dvěma bajty následovně.

5 4 3 2 1 0			
+---+---+---+---+---+---+			
	O P C O D E		Implicit
+---+---+---+---+---+---+			
5 4 3 2 1 0			
+---+---+---+---+---+---+			
	O P C O D E		+---+---+---+---+---+---+
+---+---+---+---+---+---+			R E G 1 R E G 2 RR
			+---+---+---+---+---+---+
5 4 3 2 1 0			
+---+---+---+---+---+---+			
	O P C O D E		+---+---+---+---+---+---+
+---+---+---+---+---+---+			R E G 1 R E G 2
			+---+---+---+---+---+---+
			+---+---+---+---+---+---+
			I N D E X OFFSET RX
			+---+---+---+---+---+---+
5 4 3 2 1 0			
+---+---+---+---+---+---+			
	M O D		R E G
+---+---+---+---+---+---+			R
+---+---+---+---+---+---+			
	M O D		R E G
+---+---+---+---+---+---+			+---+---+---+---+---+---+
			I N D E X OFFSET RX
			+---+---+---+---+---+---+

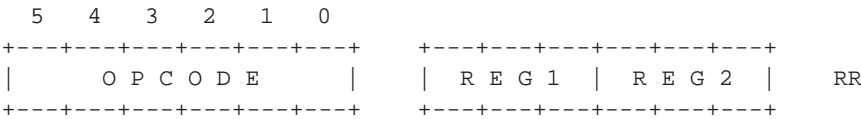
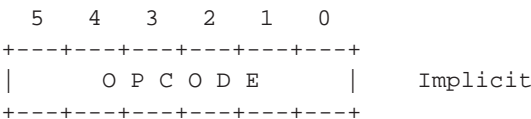
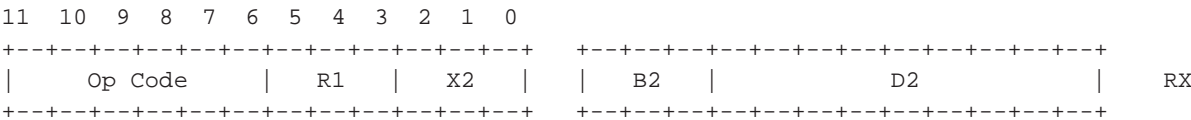
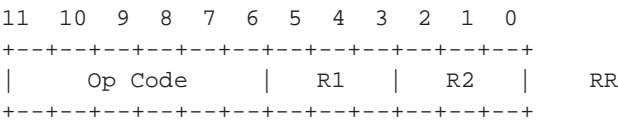
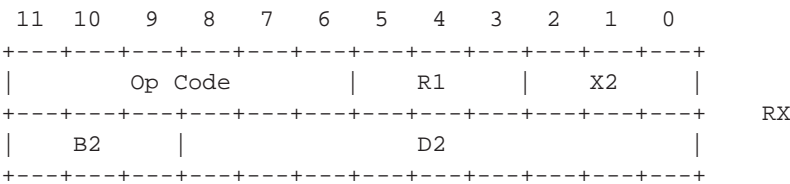
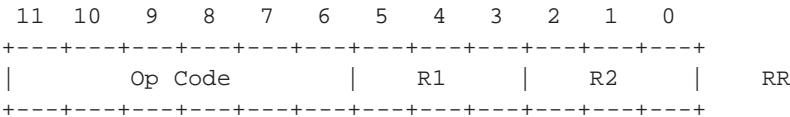
Instrukce by tedy mohla mít délku od 1 až do třeba 7 bajtů. Ovšem toho času co by si to vzalo než by se taková instrukce dekodovala až k získání zdrojových operandů a efektivní adresy kam uložit výsledek.

Tabulka 29-21. Tabulka instrukcí

kód	mnemo	popis	poznámka

29.8.4. IBM like

Architektura inspirovaná IBM.



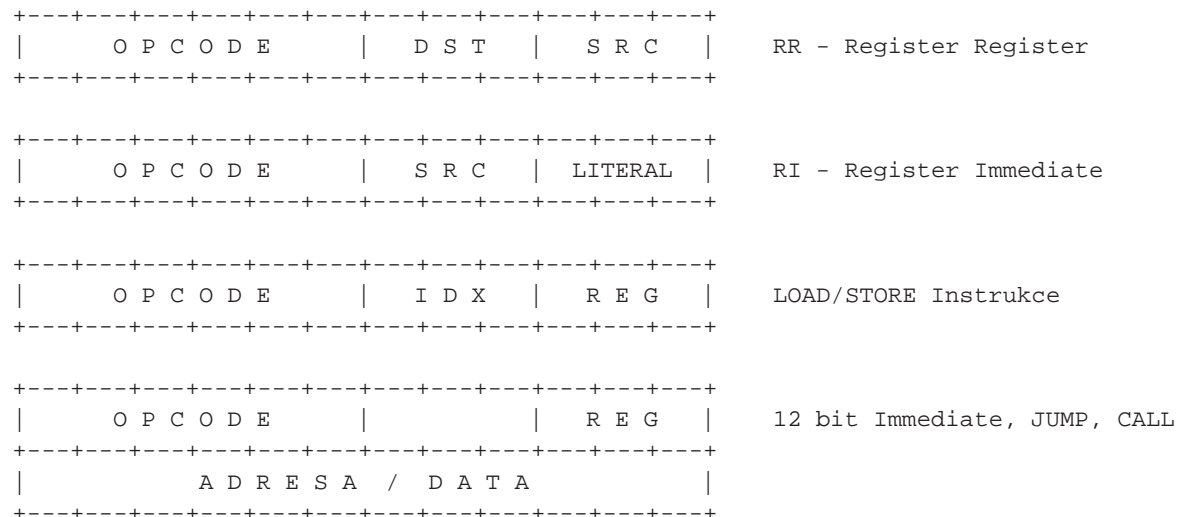
29.8.5. RISC inspirovaná Load/Store architektura

Výchozí požadavky jsou:

- Žádné složité adresní módy. Jedna instrukce smí pracovat jen s jednou paměťovou buňkou.
- Více registrů.
-

Tedy žádné složité adresní módy u všech instrukcí. Architektura s více registry a LOAD/STORE přístupem k paměti.

Obrázek 29-16.



Tabulka 29-22. Tabulka instrukcí

kód	mnemo	popis	poznámka
000 000			
000 001			
000 010 ddd sss	ADD Rs,Rd	$Rd + Rs \rightarrow Rd$	
000 011 ddd sss	ADC Rs,Rd	$Rd + Rs + C \rightarrow Rd$	
000 100 ddd sss	SUB Rs,Rd	$Rd - Rs \rightarrow Rd$	
000 101 ddd sss	SBC Rs,Rd	$Rd - Rs - C \rightarrow Rd$	
000 110 bbb aaa	CMP Ra,Rb		Compare Ra,Rb and set flags
000 111			
001 000 ddd sss	AND Rs,Rd	$Rd \wedge Rs \rightarrow Rd$	
001 001 ddd sss	OR Rs,Rd	$Rd \vee Rs \rightarrow Rd$	
001 010 ddd sss	XOR Rs,Rd	$Rd \text{ xor } Rs \rightarrow Rd$	
001 011 ddd sss	NOT Rs,Rd	$\neg Rs \rightarrow Rd$	
001 100 000 rrr	INC Rr		
001 100 001 rrr	DEC Rr		
001 100 010 rrr	ROR Rr		
001 100 011 rrr	ROL Rr		
001 100 100 rrr	ASL Rr		
001 100 101 rrr	ASR Rr		
001 100 110 rrr			
001 100 111 rrr			
001 101			

kód	mnemo	popis	poznámka
001 110			
001 111			
010 ccc ddd ddd	Bc d	If condition then $PC+d \rightarrow PC$	Podmíněný skok relativní.
011 ccc 000 sss	Bc d	If condition then $Rs \rightarrow PC$	Podmíněný skok.

29.8.6. Řadiče

Jedním ze způsobů „vylepšení“ architektury 12-ti bitových počítačů by mohl být systém řadičů.

Řadič je samostatný procesor se specializovaným hardwre a s vlastním programem. Funkcionalita řadiče je dána jednak jeho konstrukcí, použitím a propojením speciálních funkčních bloků. Druhým způsobem modifikace funkcionality je programování řadiče. Tedy ty funkce které je třeba vykonávat často a rychle jsou realizovány specializovaným hardware. Zbytek je realizován programem. Tím získáme vysoký výkon specializovaných operací při velké variabilitě funkčnosti dané programem.

Řadič tvoří most mezi vnějším prostředím a počítačem. Je to takový komunikační kanál na jehož vnějším konci je V/V zařízení. Druhou stranou je připojen přímo do paměti počítače. Je toho dosaženo buď to tak, že řadič sdílí část operační paměti počítače. Nebo pro zvýšení výkonu má vlastní paměť, kterou může druhou branou adresovat počítač. V takové konfiguraci může být počítač připojen k většímu počtu plně samostatně fungujících a nijak se neovlivňujících řadičů pracujících každý na plný výkon.

Pro jednoduchost navázání na architekturu počítače je šířka slova v paměti řadiče stejná jako šířka slova hlavního počítače, tedy 12 bitů. Jejich adresní schopnosti jsou ovšem jednodušší. Tato jednoduchost je dána tím, že řadič je specializovaný na nepotřebuje velký objem paměti a na druhou stranu umožní zjednodušit instrukční soubor a elektroniku řadiče tak aby při tom podávala velký výkon. Předpokládám jen instrukce dvou základních tvarů, jak je uvedeno na obrázku.

Obrázek 29-17. Instrukční formáty pro 12-ti bitový řadič

11	0	
+-----+-----+		Obecný tvar instrukce s adresou.
Kód	Adresa	Kóduje běžné instrukce a instrukci F.
+-----+-----+		
110	ff p	IO instrukce
+-----+-----+		
111	g μ kód	Mikroinstrukce bez adresy.
+-----+-----+		

Instrukční soubor je velmi inspirován počítačem PDP-8. Hlavním důvodem je jednoduchost hardware řadiče. Předpokládám více specializovaných řadičů v počítači a není důvod aby jejich elektronika byla zbytečně komplikovaná tam kde nemá být. Výkon řadiče je v specializovaném hardware a ne v univerzálním procesoru.

Tabulka 29-23.

kód	název	popis
000 aaa aaa aaa	ADD a	Sčítání: $AC + \text{mem}[a] \rightarrow AC$
001 aaa aaa aaa	AND a	Logický součin: $AC \wedge \text{mem}[a] \rightarrow AC$
010 aaa aaa aaa	OR a	Logický součet: $AC \vee \text{mem}[a] \rightarrow AC$

kód	název	popis
011 aaa aaa aaa	F a	Specializovaná funkce. F(mem[a])
100 aaa aaa aaa	ST	
101 aaa aaa aaa	B a	Pokračování programu na adrese a.
110 fff fff ggg	IO f,p	IO operace f na portu p
111 gbb bbb bbb	μ CODE	μ bits. První bit nebo bity mohou specifikovat mikroinstrukční řetězec.

Instrukce ADD umožňuje realizovat základní aritmetické instrukce jako je sčítání, odčítání, inkrementace a dekrementace.

```

// Inkrementace.
.data
ONE: DW 1 // 0001
.code
ADD ONE // AC+1 → AC

// Dekrementace
.DATA
NEGONE: DW -1 // 7777
.CODE
ADD NEGONE // AC-1 → AC

```

Jinou ukázkou řadiče je více mikroprogramování a méně univerzálních instrukcí.

Obrázek 29-18. Instrukční formáty pro 12-ti bitový μ řadič

```

11                                0
+-----+-----+-----+
| kód | podmínka | data |
+-----+-----+-----+
| kód |       $\mu$ bits      |
+-----+-----+-----+

```

Tabulka 29-24.

kkk ccc ddd ddd	IF C LOAD r,d	

Řadič má buďto oddělený prostor instrukční a datové paměti. Tedy taková Hardwarová architektura. Nebo mohou být tyto prostory namapovány přes sebe. Řadiče jsou technologicky samostatné procesory, ale jejich instrukční sada je zjednodušená a jsou orientovány na jednodušší nebo specializované úkoly. Nejedná se tedy o univerzální procesory.

Použití řadiče může být k obsluze vybraných periférií, či přímo jako „kanálového“ procesoru zajišťujícího samostatnou komunikaci, přenos a předzpracování dat pro vlastní procesor.

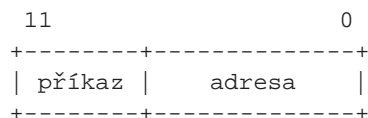
Hlavní procesor komunikuje s řadičem několika málo jednoduchými způsoby.

1. Pomocí paměti data/programu. Procesor čte a zapisuje do datové paměti řadiče, kterou vidí jako blok vlastní paměti. Procesor zapisuje do paměti programu řadiče, čímž modifikuje funkčnost řadiče.
2. Pomocí stavového slova řadiče. Stav řadiče je prezentován jedním stavovým slovem. Bity tohoto slova svou hodnotou vypovídají o jednotlivých částech řadiče a jejich stavu.

3. Pomocí řídicích instrukcí. Procesor má možnost několika málo instrukcemi: zastavit řadič, spusutit řadič.

Z hlediska architektury jsou řadiče specializované koprocesory se sdílenou pamětí.

Řídicí slovo, tedy způsob ovládání řadiče procesorem může vypadat například takto:



Řídicí instrukce jsou velmi jednoduché a je jich velmi málo. Potřebujeme jen instrukce pro spuštění řadiče a jeho zastavení. Protože startovací adresa programu řadiče je součástí startovací instrukce, můžeme tak použitím odlišných startovacích adres volat odlišné funkce řadiče dané jeho aktuálním programem.

kód	název	popis
110 aaa aaa aaa	START a	Adresa je nahrána do čítače instrukcí řadiče a tento je spuštěn.
001 xxx xxx xxx	STOP	Řadič je zastaven po dokončení právě vykonávané instrukce.
010 xxx xxx xxx	CONT	Opětovné spuštění bez zadání adresy. Řadič pokračuje v práci tam kde přestal.

29.9. FJE5

Odkazy:

- Class FJE5¹
-
-

*

Jednoduchý pětibitový procesor. Má tyto registry:

Tabulka 29-25. Registry procesoru FJE5

název	šířka [bit]	popis
A	5	akumulátor
I	5	index registr, obsahuje 5 vyšších bitů 10-ti bitové adresy
SP	10	ukazatel na zásobník
PC	10	ukazatel na instrukce

Tabulka 29-26. Instrukce procesoru EFJ5

mnemonic	kód	popis
NOP	00000	PC+1 → PC
XAI	00001	A ↔ I; PC+1 → PC; //vymění se obsahy registrů A a I

mnemonic	kód	popis
LDA n	00010 nnnnn	$n \rightarrow A$; $PC+2 \rightarrow PC$; //hodnota n se uloží do střadače A
LDI n	00011 nnnnn	$n \rightarrow I$; $PC+2 \rightarrow PC$;
LDAI bias	00100 nnnnn	$32 * I + n \rightarrow A$; $PC+2 \rightarrow PC$
LDAD bias	00101 bbbbb	$mem[mem[\{I,b\}]] \rightarrow A$; $PC+2 \rightarrow PC$
STAI bias	00110 bbbbb	$A \rightarrow mem[\{I,b\}]$; $PC+2 \rightarrow PC$; // Store A indirectly
STAD bias	00111 bbbbb	$A \rightarrow mem[mem[\{I,b\}]]$; $PC+2 \rightarrow PC$; // Store A double indirectly
PUSH	01000	$A \rightarrow mem[SP-1]$; $I \rightarrow mem[SP-2]$; $SP-2 \rightarrow SP$; $PC+1 \rightarrow PC$; // Push A and I on Stack
POP	01001	$mem[SP] \rightarrow I$; $mem[SP+1] \rightarrow A$; $SP+2 \rightarrow SP$; $PC+1 \rightarrow PC$; // Pop A and I from Stack
CALL addr	01010 hhhhh llll	$PC \rightarrow mem[SP]$; $SP-2 \rightarrow SP$; $hl \rightarrow PC$; // Call a subroutine
RET	01011	$\{mem[SP], mem[SP+1]\} \rightarrow PC$; $SP+2 \rightarrow SP$; // Return from subroutine
LDS adr	01100 hhhhh llll	$hl \rightarrow SP$; $PC+3 \rightarrow PC$; // Load SP
JP addr	01101 hhhhh llll	$hl \rightarrow PC$
JPZ addr	01110 hhhhh llll	if $A=0$ then $hl \rightarrow PC$ else $PC+3 \rightarrow PC$;
JPNZ addr	01111 hhhhh llll	if $A \neq 0$ then $hl \rightarrow PC$ else $PC+3 \rightarrow PC$;
ADD bias	10000 bbbbb	$A + mem[I,bias] \rightarrow A$; $PC+2 \rightarrow PC$;
SUB bias	10010 bbbbb	$A - mem[I,bias] \rightarrow A$; $PC+2 \rightarrow PC$
AND bias	10100 bbbbb	$A \text{ and } mem[I,bias] \rightarrow A$; $PC+2 \rightarrow PC$
OR bias	10110 bbbbb	$A \text{ or } mem[I,bias] \rightarrow A$; $PC+2 \rightarrow PC$
XOR bias	11000 bbbbb	$A \text{ xor } mem[I,bias] \rightarrow A$; $PC+2 \rightarrow PC$
SARA	11010	$A \gg 1 \rightarrow A$; $PC++$; //Shift Arithmetically Right A
SLRA	11100	$A \ggg 1 \rightarrow A$; $PC++$; //Shift Logically Right A
SLA	11110	$A \ll 1 \rightarrow A$; $PC++$; // Shift Left A
HALT	11111	Zastaví vykonávání programu.

29.10. SPEW

Odkazy:

- SPEW: A Fictitious Processor with 4K of Memory²
-
-

Fiktivní procesor s 4KB operační paměti.

Tabulka 29-27. Registry procesoru SPEW

název	šířka [bit]	adresa	popis
A	8	F00	accumulator
STATUS	8	F01	status register
STK	16	F02,F03	stack pointer

název	šířka [bit]	adresa	popis
PC	16	F04,F05	program counter

Paměť má velikost 4096 bajtů a je obsazena následovně.

Tabulka 29-28. Registry procesoru SPEW

adresa(y)	label	popis	
000-0FF		inicializováno hodnotami 00-FF	
100-EFF		core/data area	
F00	A	accumulator	
F01	STATUS	status register	
F02,F03	STK	stack pointer	
F04,F05	PC	program counter	
F06-FFF	STACK	default stack area	

Instrukce

Hex	Mnemonic	Brief description
----	-----	-----
00xx	OSCALL xx	Use an OS call (int 21h service)
0xxx	JP xxx	Jump to new PC address
1000	RETURN	Return from a GOSUB call
1xxx	GOSUB xxx	Go (call) a sub-routine
2xxx	PUSHB [xxx]	Push a byte onto the stack
3xxx	POPB [xxx]	Pop a byte from the stack
4xxx	LDA [xxx]	Load A (accumulator) from memory
5xxx	STA [xxx]	Store A into memory
6xxx	RDI [(xxx)]	Read an indirect byte using mem-ptr
7xxx	WRI [(xxx)]	Write an indirect byte using mem-ptr
8xxx	RDSYS [0000:0xxx]	Read byte from system ram[0000:0xxx]
9xxx	ADDW [xxx],A	Add sign-extended A to word variable
Acpp	JPcc +pp	Jump if the condition is true
Bxxx	ADCA [xxx]	A + mem[xxx] + CF
Cxxx	SBBA [xxx]	A - mem[xxx] - CF
Dxxx	ORA [xxx]	A OR mem[xxx]
Exxx	ANDA [xxx]	A AND mem[xxx]
Fxxx	XORA [xxx]	A XOR mem[xxx]

29.11. MPROZ

Odkazy:

- Index pro <ftp://137.193.64.130/pub/mproz/>³
-
-

Velmi jednoduchý procesor. Má jen 3 registry.

Tabulka 29-29.

jméno	šířka [bit]	popis
PC	15	Ukazatel na instrukci
A	16	Středač
F	1	Jednabitový příznakový registr

29.11.1. Instrukce

Obrázek 29-19. br adr

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 1 |                                     adresa |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Instrukce skoku **br adr** provádí následující:

1. If F=0 then adr → PC
2. 0 → F

Obrázek 29-20. add adr1,adr2,adr3

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 |                                     adr 1 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 |                                     adr 2 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 |                                     adr 3 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

mem[adr1]+mem[adr2] → mem[adr3]
F=carry

Obrázek 29-21. nadd adr1,adr2,adr3

```

+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 |                                     adr 1 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 |                                     adr 2 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0 |                                     adr 3 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

not(mem[adr1]+mem[adr2]) → mem[adr3]
if result = 0 then F=1 else F=0

29.12. ZPU

* *Attributy:* id="ZPU"

Odkazy:

- Zylin CPU⁴
-

Architektura ZPU je bezoperandová. Všechny instrukce mají fixní velikost 8-mi bitů.

Opcode	Name	Description
0000 0000	BREAKPOINT	
1xxx xxxx	IM	Immediate value
010x xxxx	STORESP	
011x xxxx	LOADSP	
0001 xxxx	ADDSP	
001x xxxx	EMULATE	
	PUSHPC	
0000 0100	POPPC	
0000 1000	LOAD	
0000 1100	STORE	
0000 0010	PUSHSP	
0000 1101	POPSP	
0000 0101	ADD	
0000 0110	AND	
0000 0111	OR	
0000 1001	NOT	
0000 1010	FLIP	
0000 1011	NOP	
61	PUSHSPADD	
57	POPPCREL	
49	SUB	
50	XOR	
51	LOADB	
52	STOREB	
34	LOADH	
35	STOREH	
36	LESSTHAN	
37		

29.13. Úvaha na jednoduchou konstrukcí SC-2

*

Odkazy:

- TX-0
- L-1

Pokud chci realizovat počítač s jednodušších logických IO či přímo z diskretních součástek (tranzistorů), musím toto zohlednit v konstrukci. Čím méně komponent bude konstrukce mít, tím snáze a rychleji ji postavím a bud v ní také méně chyb. Proto bych jako první konstrukcí začal s obměnou TX-0.

Obrázek 29-22. První návrh instrukčního formátu SC-2

Instrukce s adresou	Op Code	Adresa
	0 1 2	n
Mikrokódová instrukce	OPR	μCode
	0 1 2	n

J	00	adresa
	11 10 9	0
A	01	adresa
	11 10 9	0
S	10	adresa
	11 10 9	0
O	11	CL CP RR RL SKL 4 3 RDC WRC 0
	11 10 9 8 7 6 5 4 3 2 1 0	

Myšlenkou je co nejjednodušší dekodování instrukcí. Instrukční slovo počítače je rozděleno do dvou polí. První pole obsahuje bity operačního kódu a ve druhém poli se nachází adresa operandu. V případě mikrooperace se v druhém poli nachází mikrokód.

Instrukční dekodér je pak složen ze dvou obvodů, demultiplexeru 1 ze 4 který dekoduje operační kód a posuvného registru který pomáhá generovat pulzy pro mikroinstrukce.

Nyní tedy jak přidělit instrukcím kódy. Máme k dispozici jen dva bity a to jsou tedy čtyři možné kombinace.

Tabulka 29-30.

OpCode	Mnemonic	popis
00	JMP	addr → PC
01	ADD	AC + Mem[addr] → AC
10	STO	AC → Mem[addr]
11	OPR	proved' mikrooperace sériově

Kód pro instrukci skoku byl zvolen tak aby adresa buňky mohla sloužit jako adresa skoku. Máme-li v AC adresu a chceme-li skočit začít vykonávat instrukce na ní, pak to provedem následujícím kódem:

```
325:    STO 326
326:    0
```

Instrukce **STO** na adrese 325 uloží adresu cílového programu v AC do paměti na adresu 326. Hned poté se pokračuje instrukcí na adrese 326. Protože ta má oba nejvyšší byty nastaveny na 0 je to instrukce skoku.

Za úvahu stojí, jestli instrukci STO nenahradit instrukcí obdobnou DCA, tedy uloží obsah AC na uvedenou adresu a poté vymaže obsah AC.

Instrukce skoku je podmíněnou instrukcí. Skok se provede v závislosti ob obsahu jednobitového příznaku LINK.

Mikrooperace instrukce OPR je třeba vhodně zvolit. Jednak to musí být ty správné a potom musí být ve správném pořadí aby umožňovali smysluplné kombinace.

```
1. . . . . CLA (CLAC)
.1. . . . . CLL (CLC)
. . . . .
. . . NOT AC
      NOT LINK

      ROL
```

ROR

HALT

29.14. Varianty SC3

*

Obrázek 29-23. První návrh instrukčního formátu SC-3

Návrh instrukčních variant SC-3.

Instrukce s adresou	Op Code	Adresa
	_n _{n2} _{n3}	₀
Mikrokódová instrukce	OPR	µCode
	_n _{n2} _{n3}	₀

V případě 12-ti bitového slova s 10-ti bitovou adresou.

JUMP	11	0 0 0	9	8										adresa										0
ADD	11	0 0 1	9	8										adresa										0
STORE	11	0 1 0	9	8										adresa										0
AND	11	0 1 1	9	8										adresa										0
I4	11	1 0 0	9	8										adresa										0
I5	11	1 0 1	9	8										adresa										0
I6	11	1 1 0	9	8																				0
OPER	11	1 1 1	9	CL	CPA	CPL	RR	RL	SKL	SK?	IOR	IOW	8	7	6	5	4	3	2	1	0			

29.15. Jednoduchý procesor orientovaný na forth

*

Odkazy:

- Forth⁵
-

* <http://groups.google.com/group/comp.lang.forth/msg/10872cb68edcb526>

FORTH Primitives Comparison (use a fixed width font)

```

-----
3      primitives - Frank Sargent's "3 Instruction Forth"
9      primitives - Mark Hayes theoretical minimal Forth bootstrap
9,11   primitives - Mikael Patel's Minimal Forth Machine (9 minimum, 11 full)
13     primitives - theoretical minimum for a complete FORTH (Brad Rodriguez)
16,29  primitives - C. Moore's word set for the F21 CPU (16 minimum, 29 full)
20     primitives - Philip Koopman's "dynamic instruction frequencies"
```

```

23    primitives - Mark Hayes MRForth
25    primitives - C. Moore's instruction set for MuP21 CPU
36    primitives - Dr. C.H. Ting's eForth, a highly portable forth
46    primitives - GNU's GFORTH for 8086
58-255 functions - FORTH-83 Standard (255 defined, 132 required, 58 nucleus)
60-63 primitives - considered the essence of FORTH by C. Moore (unknown)
72    primitives - Brad Rodriguez's 6809 CamelForth
74-236 functions - FORTH-79 Standard (236 defined, 147 required, 74 nucleus)
94-229 functions - fig-FORTH Std. (229 defined, 117 required, 94 level zero)
133-?  functions - ANS-FORTH Standard (? defined, 133 required, 133 core)
200    functions - FORTH 1970, the original Forth by C. Moore
240    functions - MVP-FORTH (FORTH-79)
~1000  functions - F83 FORTH
~2500  functions - F-PC FORTH
FIXME  27 ?      - C. Moore's MachineForth

```

eForth primitives (9 optional)

```

doLIT doLIST BYE EXECUTE EXIT next ?branch branch ! @ C! C@ RP@ RP! R> R@ >R
SP@ SP! DROP DUP SWAP OVER 0< AND OR XOR UM+ TX!
?RX !IO $CODE $COLON $USER D$ $NEXT COLD IO?
9 MRForth bootstrap theoretical

```

```

@ ! + AND XOR (URSHIFT) (LITERAL) (ABORT) EXECUTE
9 Minimal Forth (3 optional)

```

```

>r r> 1+ 0= nand @ dup! execute exit

```

drop dup swap

23 MRForth primitives

```

C@ C! @ ! DROP DUP SWAP OVER $>$R R$>$ + AND OR XOR (URSHIFT) 0$<$ 0=
(LITERAL) EXIT (ABORT) (EMIT) (KEY)
20 Koopman high execution, Dynamic Freq.

```

```

CALL EXIT EXECUTE VARIABLE USER LIT CONSTANT 0BRANCH BRANCH I @ C@ R> >R
SWAP DUP ROT + = AND
46 Gforth

```

```

:DOCOL :DOCON :DODEFER :DOVAR :DODOES ;S BYE EXECUTE BRANCH ?BRANCH LIT @ !
C@ C! SP@ SP! R> R@ >R RP@ RP! + - OR XOR AND 2/ (EMIT) EMIT? (KEY) (KEY?)
DUP 2DUP DROP 2DROP SWAP OVER ROT -ROT UM* UM/MOD LSHIFT RSHIFT 0= =
36 eForth

```

```

BYE ?RX TX! !IO doLIT doLIST EXIT EXECUTE next ?branch branch ! @ C! C@ RP@
RP! R> R@ >R SP@ SP! DROP DUP SWAP OVER 0< AND OR XOR UM+ $NEXT D$ $USER
$COLON $CODE

```

29.15.1. nanoForth

*

Odkazy:

- Nano Forth - A tiny dialect of the Forth programming language⁶
-
-
-

V rámci hledání co nejjednoduššího procesoru pro jazyk Forth jsem našel nanoForth.

Tabulka 29-31. Tabulka instrukcí nanoForth

kód	instr.	slovo	PSP efekt	RSP efekt	popis
0	add	+	(n ₁ n ₂ → n ₃)	(→)	addition
1	nand	NAND	(n ₁ n ₂ → n ₃)	(→)	negative and
2	xor	XOR	(n ₁ n ₂ → n ₃)	(→)	exclusive-or
3	ashr	2/	(n ₁ → n ₂)	(→)	arithmetic shift right, sign retained
4	fetch	@	(addr → n)	(→)	fetch cell at addr
5	store	!	(n addr →)	(→)	store n to cell at addr
6	lit	LIT	(→ n) "value"	(→)	push inline cell, PSP++
7	dup	DUP	(n → n n)	(→)	duplicate
8	swap	SWAP	(n ₁ n ₂ → n ₂ n ₁)	(→)	exchange
9	drop	DROP	(n →)	(→)	discard
A	tor	>R	(n →)	(→ n)	pop parameter stack to return stack
B	rfrom	R>	(→ n)	(n →)	pop return stack to parameter stack
C	enter	CALL	(→)	(→ PC)	DOCOL, NEST - call subroutine
D	exit	SEMIS	(→)	(addr →)	RETURN - return from subroutine
E	zjmp	JZ	(n →)	(→)	jump if false
F	iptor	BEGIN	(→)	(→ ip)	push IP (next instruction fetch addr) to R
	NOP		(→)	(→)	no operation

29.15.2. Minimalistické verze

*

Odkazy:

- A 3-INSTRUCTION FORTH FOR EMBEDDED SYSTEMS WORK⁷
-

29.15.2.1. Nagro VM

*

Odkazy:

- Nagro VM Dual Stack Virtual Machine⁸
-

Tabulka 29-32. Tabulka instrukcí Nagro VM

kód	instr.	slovo	PSP efekt	RSP efekt	popis
0	NOP	nop			does nothing
1	LIT	nop			push a literal to the stack

kód	instr.	slovo	PSP efekt	RSP efekt	popis
2	DUP	dup			duplicate the top value on the stack
3	DROP				
4	SWAP				
5	PUSH	>R			
6	POP	R>			
7	CALL				
8	JUMP				
9	;				return from a subroutine
10	>JUMP				
11	<JUMP				
12	!JUMP				
13	=JUMP				
14	@				
15	!				
16	+				
17	-				
18	*				
19	/MOD				
20	AND				
21	OR				
22	XOR				
23	<<				
24	>>				
25	0;				
26	1+				
27	1-				
28	IN				
29	OUT				
30	WAIT				

Poznámky

1. <http://jfrace.sourceforge.net/javadoc/jfrace/cpu/FJE5.html>
2. <http://www.ddj.com/embedded/210601643>
3. <ftp://137.193.64.130/pub/mproz/>
4. <http://opensource.zylin.com/zpudocs.html>
5. <http://forth/index.html>
6. <http://www.quirkle.com/misc/forth.htm>

7. <http://pygmy.utoh.org/3ins4th.html>
8. <https://ngaro.jottit.com/instructionset>

Kapitola 30. Návrh instrukční sady procesoru

Tato kapitola je o navrhování procesoru od instrukční sady, tedy sestavení instrukční sady.

30.1. Instrukční sady s malým počtem instrukcí a jednou či více adresami

30.1.1. Instrukční sada bez instrukcí

Odkazy:

- The SUBLEQ URISC (Ultimate RISC) / OISC (One Instruction Set Computer) Architecture¹
- One instruction set computer²
- SUBLEQ³
- One Instruction Set Computer (OISC)⁴
- The Mark II OISC Self-Interpreter⁵
-
-

Tato možnost odpovídá tomu, že všechny bity instrukčního slova specifikují adresu. Pro specifikaci instrukce nemáme žádný bit. To odpovídá instrukční sadě s jednou implicitní instrukcí. Tato instrukce pak musí být taková aby se s ní dalo "programovat". Pro tento formát je ovšem jedna adresa nedostatečná. Instrukční slovo musí obsahovat adresy dvě.

30.1.2. Instrukční sada se dvěma instrukcemi

Tato možnost odpovídá případu kdy pro specifikování operačního kódu použijeme jen jeden bit a zbytek bitů kóduje adresu. Můžeme tak mít maximálně dvě instrukce. Aby byla instrukční sada použitelná, musí obě instrukce splňovat následující předpoklady.

První instrukce, nazvěme ji LOAD provádí principiálně převod hodnoty z paměti do akumulátoru. Protože nemáme žádnou další možnost jak zpracovávat informace, musím ji zkombinovat s nějakou aritmetickou operací.

```
SUB a    # AC - mem[a] → AC
STO a    # AC → mem[a]
```

Poznámka: Tady zjišťuji že to asi nepůjde. jedna adresa je málo. Nevím jak specifikovat skok. Je nutné mít alespoň dvě adresy v instrukci. První je adresa operandu, druhá je alternativní adresa následující instrukce, tedy adresa skoku.

30.1.3. Instrukční sada se čtyřmi instrukcemi

Tato možnost odpovídá případu kdy operační kód specifikují dva bity instrukčního slova. Zbytek instrukčního slova pak tvoří adresu buňky v paměti. Takto byl konstruován počítač TX-0.

Instrukce jsou v zásadě takovéto:

ADD a # {AC=AC+mem[a]}, instrukce slouží křenosu informace z paměti do akumulátoru a je zkombinována s aritmetickou instrukcí pro sčítání.

ST a # {mem[a]=AC}, instrukce uloží obsah střadače AC do paměti na adresu a.

BZ a # {if AC=0 then PC=a} podmíněný skok

OPR a # mikrokódová operace. každý bit pole a specifikuje nějakou operaci s hodnotou ve střadači AC.

30.1.4. Instrukční sada s osmi instrukcemi

Tento případ odpovídá třem bitům operačního kódu. Takto je konstruován například počítač PDP-8.

30.1.5. Instrukční sada s šestnácti instrukcemi

Instrukce se čtyřmi bity na operační kód umožňují již celkem velkou sadu až 16 instrukcí. Nyní, když jsme prošli návrhy s 8 a menším počtem instrukcí, musí nám těchto 16 zdát hodně. A to také je. Při umísťování instrukcí do tabulky můžeme usnadnit programátorovi život tak, že se podíváme na běžně psyný kód pro procesor s menším počtem instrukcí. Pokusíme se vyhledat často opakované vzory a těm přidělit vlastní operační kód. Takto jsem do tabulky umístil instrukce LD, SUB, OR a XOR.

S tím jak se tabulka plnila jsem se rozhodl instrukce v ní přeskupit. Myšlenka přeskupení instrukcí je rozmístit je tak, aby bylo možno realizovat jejich dekodování jednodušeji, s menším počtem hradel. První myšlenka byla oddělit od sebe instrukce které z paměti pouze čtou, od instrukcí které do paměti i zapisují. Takto jsem se rozhodl umístit do prvních 8 pozic, první a do dalších 4 druhé. Je-li tedy první bit operačního kódu 0 může obvod přípravy operandu zahájit čtení operandu z paměti nezávisle na dekodování ostatních bitů operačního kódu. Vzniklo tedy rozdělení:

0...	operand=mem[a]
1	...; →mem[a]

Toto jsem dále upravil po zakreslení instrukcí skoků.

0...	operand=mem[a] # load
10	...; →mem[a] #store, load modify store
11	... →PC #skoky

Do tabulky jsem také umístil instrukce pro zmenšení a zvětšení hodnoty v paměti (INC a, DEC a). A poté ještě i instrukce s nepřímým adresováním (LDD a, STD a). Na závěr jsem se rozhodl doplnit skoky instrukcí určenou pro programování cyklů (DJNZ)

Zůstaly mi volné dvě místa v tabulce s operačními kódy 0111 a 1111. Do těch jsem umístil mikrokódovanou instrukci ve stylu instrukce OPR a poslední pozici nechal volnou pro případnou další mikrokódovanou instrukci nebo pro rozšíření procesoru.

Výsledná tabulka vypadá následovně.

Tabulka 30-1. Ukázka návrhu instrukční sady

kód	zápis	popis operace
0000	LD a	AC=mem[a]; {ZN}
0001	LDD a	AC=mem[mem[a]]; {ZN}

kód	zápis	popis operace
0010	ADD a	$AC = AC + \text{mem}[a]$; {CZN}
0011	SUB a	$AC = AC - \text{mem}[a]$; {CZN}
0100	AND a	$AC = AC \wedge \text{mem}[a]$; {ZN}
0101	OR a	$AC = AC \vee \text{mem}[a]$; {ZN}
0110	XOR a	$AC = AC \oplus \text{mem}[a]$; {ZN}
0111	OPR x	μCode CLA, CLC, CPA, CPC, ROT(Arit/Logic), ROT(Left, Right), INCA, ...
1000	ST a	$\text{mem}[a] = AC$
1001	STD a	$\text{mem}[\text{mem}[a]] = AC$
1010	INC a	$\text{mem}[a] = \text{mem}[a] + 1$; {CZ}
1011	DEC a	$\text{mem}[a] = \text{mem}[a] - 1$; {CZ}
1100	JSR a	save PC; $a \rightarrow PC$
1101	JMP a	$a \rightarrow PC$
1110	DJNZ a	$AC = AC - 1$; if $AC \neq 0$ then $a \rightarrow PC$
1111		<i>volné pro další rozšíření</i>

V nevyužitých polích mohou být mikrokódované instrukce. Ve skutečnosti taková instrukce musí být alespoň jedna. Tato pak specifikuje operace nad střadačem AC a příznakem C, (CLA, CLC, CPA, CPC). Další operace jsou rotace a posuvy (ROL, ROR, ASL, ASR). Poté doplňkové instrukce jako například zastavení procesoru, speciální instrukce jako například prohození (swap) dolní a horní poloviny střadače, atd.

Uvedený příklad instrukcí je jen orientační. Instrukční sada není založena na důkladné analýze kódu a je to jen "střelba od boku". Už teď mě napadají drobná vylepšení.

Úprava LDS a STD instrukcí pro práci se zásobníkem. Tyto instrukce bych přejmenoval a předefinoval takto:

	LDS	$\text{mem}[a] = \text{mem}[a] + 1$; $AC = \text{mem}[\text{mem}[a]]$ # POP
	STS	$\text{mem}[\text{mem}[a]] = AC$; $\text{mem}[a] = \text{mem}[a] - 1$; # PUSH

Poznámky

1. <http://techtinkering.com/articles/?id=20>
2. http://en.wikipedia.org/wiki/One_instruction_set_computer
3. <http://mazonka.com/subleq/index.html>
4. <http://www.sccs.swarthmore.edu/users/06/adem/engin/e25/final/>
5. <http://eigenratios.blogspot.com/2006/09/mark-ii-oisc-self-interpreter.html>

Kapitola 31. Binární aritmetika

Poznámky k binární aritmetice. Vše co je nutné vědět, abychom mohli počítat binárně, tedy pomocí digitálních obvodů se dvěma úrovněmi signálu. Rovněž jso zde základy pro programování aritmetických operací na počítačích.

Odkazy:

- Binary Arithmetic¹ Addition, subtraction, multiplication, and division notes.
- Binary Multiplication²
- Binary Division by Shift and Subtract³
-
-
-

Poznámky

1. http://academic.evergreen.edu/projects/biophysics/technotes/misc/bin_math.htm
2. <http://courses.cs.vt.edu/csonline/NumberSystems/Lessons/Multiplication/index.html>
3. <http://courses.cs.vt.edu/~cs1104/BuildingBlocks/divide.030.html>

VII. Mikroprocesory a mikrořadiče

Tato část je o mikrořadičích, někdy také nazývaných jednočipové mikropočítače. Podrobněji popisuje jen některé architektury jako je AVR.

Odkazy:

- MSP430²⁷³

-

Tato část obsahuje více či méně detailní informace o vybraných mikroprocesorech, a to tak jak mě zaujali, nebo jak jsem potřeboval. Největší část je věnovaná 8-mi bitovým mikroprocesorům.

1. 8-mi bitové mikroprocesory

Toto jsou nejdůležitější mikroprocesory všech dob, protože byly první jenž dosáhly významného rozšíření. Rovněž je na nich patrné hledání nových cest a způsobů jako je práce se zásobníkem, volání podprogramů atd.

Dovolil bych si vybrat nejdůležitější obvody. Dle mého názoru jsou to 34 jakožto první procesor v CMOS provedení a rovněž pro jeho odlišnou architekturu. Pak procesor 6502, který pro svou cenu dosáhl velkého rozšíření v počítačích jako AppleII, Commodore C64, Atari 800 a dalších. Posledně zmíněným procesorem na který chci upozornit je 33 který rovněž velmi přispěl k rozšíření počítačů mezi veřejnost ve strojích jako ZX81, ZX Spectrum, Sharp MZ800 a strojích s operačním systémem CP/M.

2. Forth mikroprocesory

Tyto mikroprocesory v ničem nepřipomínají běžné procesory, protože jsou orientované na programovací jazyk Forth. V některých ohledech mohou být velmi zvláštní například architekturou připomínající RISC či naopak umístěním několika instrukcí v jednom paměťovém slově.

Poznámky

273. http://en.wikipedia.org/wiki/TI_MSP430

Kapitola 32. MuP21 a F21

Transfer Instructions: JUMP, CALL, RET, JZ, JCZ
Memory Instructions: LOAD, STORE, LOADP, STOREP, LIT
ALU Instructions: COM, XOR, AND, ADD, SHL, SHR, ADDNZ
Register Instructions: LOADA, STOREA, DUP, DROP, OVER, NOP

The F21 had 27 instructions to the MuP21's 24. (Only 23 are listed above, hmm.) They were renamed:

Code	Name	Description	Forth (with a variable named A)
00	else	unconditional jump	ELSE
01	T0	jump if T0-19 is false w/ no drop	DUP IF
02	call	push PC+1 to R, jump	:
03	C0	jump if T20 is false	CARRY? IF
06	RET	pop PC from R (subroutine return)	;
08	@R+	fetch from address in R, increment	R R@ @ R> 1+ >R
09	@A+	fetch from address in A, increment	A A @ @ 1 A +!
0A	#	fetch from PC+1, increment PC	LIT
0B	@A	fetch from address in A	A @ @
0C	!R+	store to address in R, increment R	R@ ! R> 1+ >R
0D	!A+	store to address in A, increment A	A @ ! 1 A +!
0F	!A	store to address in A	A @ !
10	com	complement T	-1 XOR
11	2*	left shift T, 0 to T0	2*
12	2/	right shift T, T20 to T19	2/
13	++	add S to T if T0 is true	DUP 1 AND IF OVER + THEN
14	-or	exclusive-or S to T	XOR
15	and	and S to T	AND
17	+	add S to T	+
18	pop	pop R, push to T	R>
19	A	push A to T	A @
1A	dup	push T to T	DUP
1B	over	push S to T	OVER
1C	push	pop T, push to R	>R
1D	A!	pop T to A	A !
1E	nop	delay 2ns	NOP
1F	drop	pop T	DROP

Kapitola 33. Mikroprocesor Z80

Attributy: id="Z80"

Odkazy:

- Počítače a mikropočítače s procesorem Z80
- Home of the Z80 CPU¹
- Z80² na české Wikipedii
- Zilog Z80³ na wikipedii
- stránky firmy Zilog⁴
- Zilog Z80 Microprocessor Oral History Panel⁵
-
-

Software pro Z80/CPM:

- Legacy Z-80⁶ — software od MICROCode Consulting
- CP/M⁷
- 11. Introduction to CP/M 3.0⁸

Mikroprocesor Z80 a jeho rodina podpůrných obvodů z dílny firmy Zilog.

Krátký přehled obvodů:

- Z80, Z80-CPU, Z80A-CPU, Z80B-CPU, Z8400, ...
-
-

* *Historie:*

Z80 přichází na svět když Federico Faggin⁹ zakládá firmu Zilog poté co pracoval na procesoru I8080 ve firmě Intel, kterou na konci roku 1974 opustil. Firmu Zilog založil spolu s Ralphem Ungermanem a v červnu roku 1976 vypustili Z80 na trh.

Z80 byl konstruován jako binárně kompatibilní s procesorem I8080, takže byl schopen vykonávat téměř všechny programy psané pro tento procesor, hlavně nejdůležitější aplikaci a to operační systém CP/M. K dizajnu Z80 přispěl také Masatoshi Shima¹⁰, spoluautor I4004 a I8080.

Z80 poskytnul mnoho vylepšení proti I8080.

- rozšířenou sadu instrukcí, včetně instrukcí pro manipulaci s bity a pro práci s bloky dat
- nové indexové registry IX a IY spolu s adresními módy které je používají
- lepší systém přerušení, Z80 má tři módy obsluhy přerušení a navíc nemaskovatelné přerušení NMI
- druhou sadu registrů ke standardním registrům
- používá jen jedno napájecí napětí oproti třem u I8080, což zjednodušilo konstrukce počítačů
- jednofázové hodiny což odstranilo potřebu externího čipu jaký používá I8080 pro generování dvoufázového hodinového signálu
- zabudovanou podporu občerstvování pamětí typu DRAM což opět zjednodušilo konstrukce počítačů
- nemultiplexované sběrnice, což odstranilo používání externího čipu pro dekódování sběrniceových signálů jaký používá procesor I8080
-
-
-

•

Z80 je skrátka ve všech ohlede výhodnější pro konstrukci počítačů než čip I8080 ze kterého ideově vychází. Dává programátorovi k dispozici více registrů a instrukcí, a konstruktérovi umožňuje zjednodušit konstrukci, což se odrazí na ceně.

Procesor se rychle usadil na trhu a nahradil I8080 i jeho následníka I8085 a stal se jedním z nejúspěšnějších 8-mi bitových procesorů na trhu.

Procesor Zilog Z80 byl zkonstruován a prodáván firmou Zilog od června 1976. Firma procesor licencovala mnoha dalším. Mimo nelicencované kopie z Ruska a východní evropy vyráběla procesor Z80 také Toshiba jako jeden z největších dodavatelů tohoto procesoru. Zabudovaný mechanismus pro občerstvování pamětí DRAM byla pravděpodobně jedna z nejdůležitějších vlastností, která přispěla k velkému úspěchu Z80 protože snížila počet komponent potřebných ke konstrukci počítače s dynamickými pamětmi.

První verze procesoru Z80 byly konstruovány technologií NMOS a pracovali na kmitočtu 2,5MHz. Postupně jak se zlepšovala technologie zvyšoval se i kmitočet na 4MHz u Z80A, 6MHz u Z80B a 8MHz u Z80H. Byla také zkonstruována CMOS verze procesoru pracující na kmitočtech od 4MHz až po 20MHz. Později vyvinuté procesory HD64180/Z180 a eZ80 pracují na kmitočtech až 33 či 50MHz.

33.1. Obvody rodiny Zilog Z80

Z80-CPU
Z8400
Z84C00

Z80C015

Z80 DMA
Z8410
Z84C10

Z80 PIO
Z8420
Z84C20

Z80 CTC
Z8430
Z84C30

Z80 SIO
Z8440
Z84C40

Z80 SIO/0

Z80 SIO/1

Z80 SIO/2

Z84C90

Z80 KIO

SIO, PIO a CTC v jednom pouzdru

Z8470 DART

Dual Asynchronous Receiver/Transmitter

Z8530 SCC

Zilog Serial Communications Controller

33.1.1. Z80 CPU

*

Obrázek 33-1. Zapojení vývodů Z80 v pouzdru DIL40

+-----+		+-----+	
A11 -	1	___	40 - A10
A12 -	2		39 - A9
A13 -	3		38 - A8
A14 -	4		37 - A7
A15 -	5		36 - A6
CLK -	6		35 - A5
D4 -	7		34 - A4
D3 -	8		33 - A3
D5 -	9		32 - A2
D6 -	10		31 - A1
+5V -	11		30 - A0
D2 -	12		29 - GND
D7 -	13		28 - !RFSH
D0 -	14		27 - !M1
D1 -	15		26 - !RESET
!INT -	16		25 - !BUSREQ
!NMI -	17		24 - !WAIT
!HALT -	18		23 - !BUSACK
!MREQ -	19		22 - !WR
!IORQ -	20		21 - !RD
+-----+			

Popis jednotlivých signálů/vývodů.

A15-A0

Address Bus (output, active High, tristate)

!BUSACK

Bus Acknowledge (output, active Low). Potvrzení předání sběrnice. CPU se odpojí od adresové (A0-A15), datové (D0-D7) a řídicí (!MREQ, !IORQ, !RD, !WR) sběrnice. Sběrnici může používat jiný procesor. Signál !BUSACK je odpovědí na žádost o předání sběrnice !BUSREQ.

!BUSREQ

Bus Request (input, active Low). Žádost o sběrnici. Tento signál má přednost před !NMI a je vždy rozpoznáván na konci aktuálního strojového cyklu. !BUSREQ žádá CPU aby se odpojil od adresové, datové a řídicí sběrnice. Jakmile procesor rozpozná !BUSREQ, odpojí se od uvedených sběrnic (obvody přejdou do stavu s vysokou impedancí) a následně vydá potvrzení !BUSACK.

D7-D0

Data Bus (input/output, active High, tristate)

HALT**INT****IORQ****M1****MREQ****NMI****RD****RESET****RFSH****!WAIT**

WAIT (input, active Low). Tento signál říká procesoru že adresovaná paměť nebo periferní zařízení není připraveno k přenosu dat. Procesor vkládá čekací cykly dokud je !WAIT aktivní.

WR

CLK

33.1.2. Z80 SIO

*

33.2. Instrukční sada Z80

Tabulka 33-1. Základní instrukce

code	
00	
00	
FF	RST 38

```

0
+---
0 |
1 |
2 |
3 |
4 |

```

33.3. Programové vybavení pro Z80

FIXME:

Pro vývoj v assembleru (jazyk symbolických adres) existuje následující software:

- Sarcasm Z80 Assembler¹¹ napsaný v Perlu. Překladač sestává ze dvou souborů: vlastního překladače `sarcasm.pl` v Perlu a tabulky instrukcí `opcode`. Sarcasm má neobvyklou syntaxi, dovoluje například zápis více instrukcí na řádek. Jeho celková velikost je cca 550 řádek v Perlu, takže s ním lze experimentovat a relativně snadno si ho upravit.
- ZASM¹²
- spasm¹³
-
-

33.3.1. Křížové překladače, vývojová prostředí a emulátory

- Z80 Assembly IDE¹⁴ — prostředí pro vývoj v assembleru. Neobsahuje assembler! Používá externí assembler spasm
- z80-asm
- Z80-SIM¹⁵
-

Vzhledem k výkonu procesoru Z80 a současné době, je pravdepodobné že při vývoji programů budete používat prostředí jenž běží na jiných, výkonnějších procesorech. Zejména pokud jste zvyklí na grafická prostředí.

Z80 Assembly IDE¹⁶ je takové grafické prostředí. Samotné prostředí obsahuje jen editor. Jako překladač se používá upravený SPASM, jehož zdrojové kódy jsou v balíčku rovněž obsaženy. Taktéž je přiložen wabbitsign. Grafické prostředí je postaveno nad knihovnou GTK.

Dalším zajímavým prostředím je z80-asm/z80-mon jenž původně vznikl jako školní projekt (zápočťák) Petra Kulhavého, ale nyní jej udržuje Achim Flammenkamp. Toto prostředí není prostředím ve smyslu jednoho programu jako předcházející, ale několika programů jenž pracují jen ve znakovém terminálu. Čím vyniká, tak to je přítomnost simulátoru s rozsáhlými ladicími funkcemi. Rovněž dovoluje pomocí konfiguračních souborů sestavit konkrétní hardwarovou konfiguraci a pomocí dalších programů k tomuto hardware připojit periferie.

Pasmo je assembler jenž se nachází ve standardní distribuci Debian Etch.

z80asm je assembler jenž se nachází ve standardní distribuci Debian Etch.

33.3.1.1. Z80-ASM

Odkazy:

- <http://wwwwhomes.uni-bielefeld.de/achim/z80-asm.html>
-

33.3.1.2. Z80AsmIDE

Odkazy:

- wxwabbitemu¹⁷ a TI-8x emulator based on Wabbitemu
- Wabbitcode¹⁸ (již zde nejsou žádné zdrojové kódy)
-

Z adresy Wabbitcode¹⁹ si stáhneme zdrojové kódy pro Linux.

```
~/work$ tar xzvf ~/archive/z80asmide-1.2.tar.gz
```

Requirements:

GTK+ 2.8 or 2.10

OpenSSL

libgtk2.0-dev-[version], libssl-dev-[version]

Upravit config.mk

PREFIX=/home/radek/local

```
~/work$ make
```

```
~/work$ make install
```

33.3.1.3. Samostatné emulátory

*

Odkazy:

- Z80 Java Emulator fully extensible²⁰
-
-
-

33.3.2. Operační systémy

33.3.2.1.

*

33.3.2.2. UZI

*

Odkazy:

- UZI²¹ — Doug Braun (NA1DB, formerly N1OWU and SV/N1OWU)
-

33.3.2.3. UZIX

*

Odkazy:

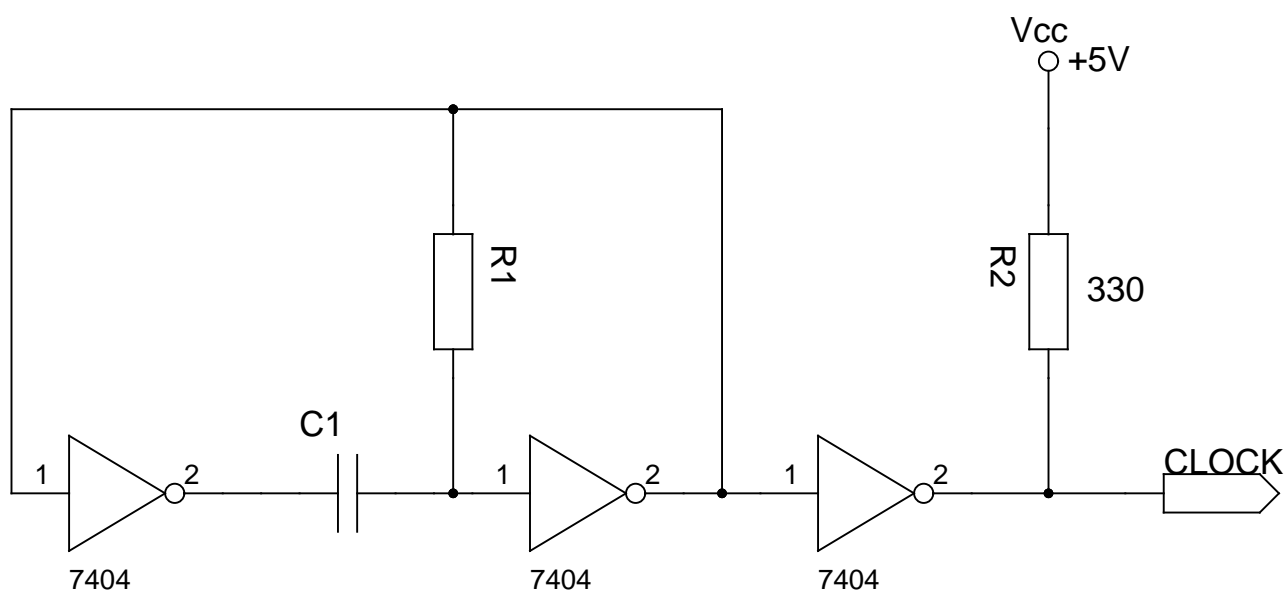
- UZIX²² [2005]
-

33.4. Zapojení pro Z80

Různá zapojení v souvislosti s konstrukcí počítače s procesorem Z80.

33.4.1. Vytváření hodinového signálu

*

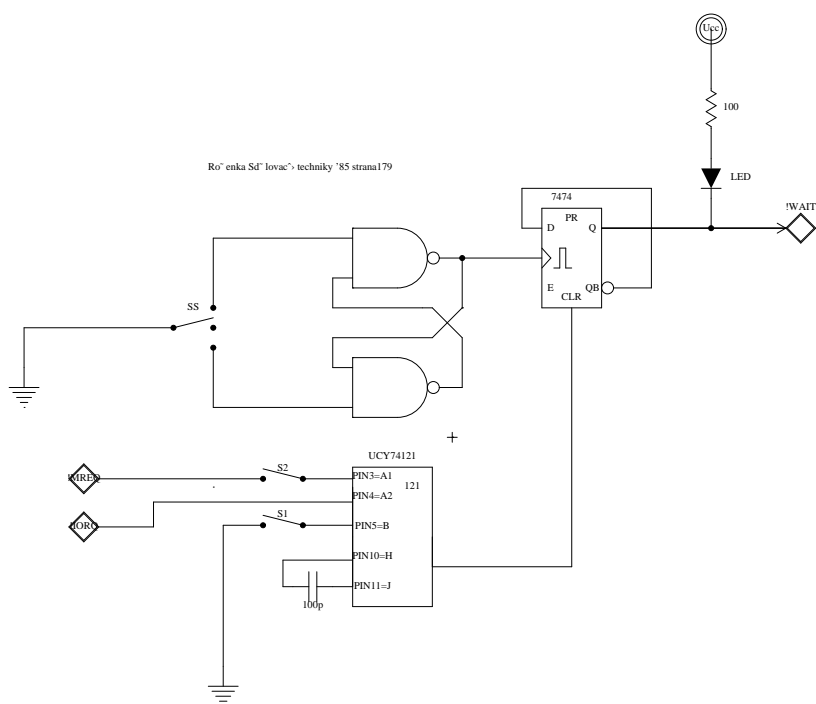
Obrázek 33-2. Vytváření hodinového signálu pomocí jednoduchého RC oscilátoru z invertorů

33.4.2. Krokování procesoru

Při ladění programů a hardware se nám může hodit obvod pro krokování procesoru. Tento obvod zajišťuje vykonání jen jedné instrukce na stisk tlačítka. Můžeme tedy přímo sledovat, s pomocí obvodů zobrazujících obsah adresové a datové sběrnice, stav v jednotlivých krocích.

První schéma je z Ročenky Sdělovací techniky '85 strana 179. Tlačítko SS slouží pro spuštění jednoho kroku. Spínač S1 v rozepnutém stavu zapíná krokovací obvod. V sepnutém stavu běží mikroprocesor normálně plnou rychlostí. Spínač S2 omezuje v rozpojeném stavu krokování jen na IO instrukce.

Obrázek 33-3. Obvod krokování pro mikroprocesor Z80

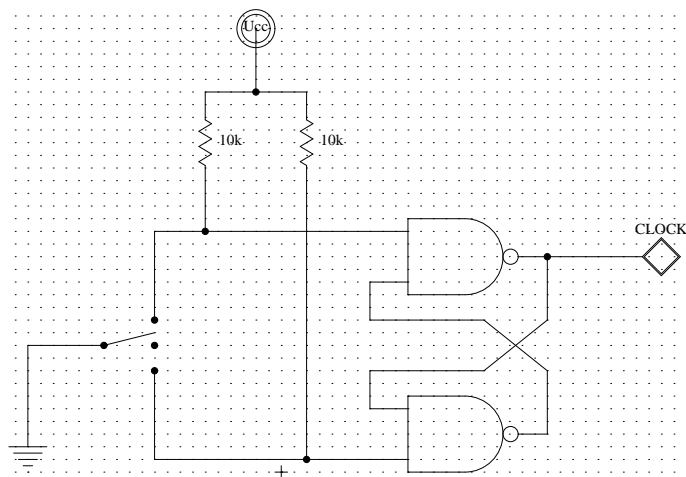


Ve stejné ročence je na další straně uveden krokovací obvod jenž umožňuje nastavit „break point“, bod přerušení. Tímto bodem je adresa instrukce.

Dalším možným způsobem krokování je ruční ovládání hodinového signálu. Mikroprocesor Z80 je vnitřně plně statický což nám dovoluje zastavit generování hodinového signálu, případně tento generovat ručně. A to je případ který řeší následující obvod.

* *The absolute minimum Z80 CPU test circuit ever seen*²³

Obrázek 33-4. Ruční ovládání hodinového signálu

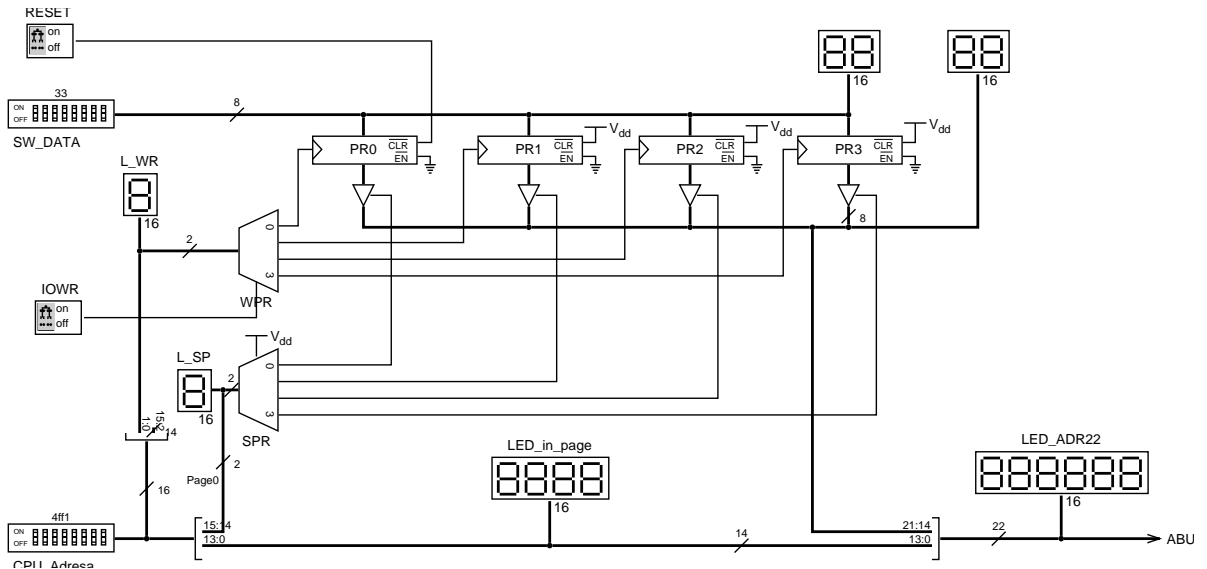


Oproti předcházejícím obvodům, tento krokuje jednotlivé takty. Umožňuje nám tedy sledovat podrobněji dění na sběrnici v průběhu každého strojového cyklu.

33.4.3. MMU21

Tato jednotka správy paměti pracuje na zcela jednoduchém principu. Pamet'ový prostor spravovaný mikroprocesorem je rozdělen na 4 stejně velké oblasti. Při schopnosti mikroprocesoru Z80 adresovat 64kiB paměti jsou oblasti velké 16kiB. Pro každou z oblastí existuje stránkový registr PR jenž obsahuje číslo stránky ve fyzické paměti. Při velikost tohoto registru 8 bitů lze adresovat maximálně 256 stránek, t.j. 4MiB.

Obrázek 33-5. MMU-21 pro Z80



Protože obsah stránkových registrů není po zapnutí počítače definován, je třeba vyřešit jakým způsobem se procesor dostane k základnímu softwarovému vybavení. Použil jsem proto na místě registru PR0 takový obvod, jenž má nulovací vstup. Tento nulovací vstup pak spojuji se signálem RESET mikroprocesoru. Toto zapojení zajistí, že vždy po resetu a taky po zapnutí počítače je ve stránkovacím registru PR0 hodnota 0. Procesor tedy začne vykonávat program ve stránce 0 namapované do bloku 0.

Na místě PR0 se tedy užije obvod HC575, pro ostatní registry pak obvody HC574.

Poznámka: Předtím, než procesor začne pracovat s pamětí v jiném než nultém bloku, je třeba provést správné nastavení příslušného stránkovacího registru.

Obrázek 33-6. Mapa paměti pro MMU21

CPU 64kB		page	ADDR	
+-----+			+-----+	
FFFF		255	3FFFFFF	RAM 1MB
C000	PR3	192	300000	
+-----+			+-----+	
BFFF		191	2FFFFFF	same RAM
8000	PR2	128	200000	R/O
+-----+			+-----+	
7FFF		127	1FFFFFF	Special devices
4000	PR1	64	0FFFFFF	VRAM, ...
+-----+			+-----+	

3FFF		63 0FFFFFF	ROM, FLASH
0000	PRO	0 000000	1MB
+-----+		+-----+	

33.4.4. MMU4M

MMU4M vznikla jako rozšíření MMU21. Základem je přidání pátého řídicího registru do MMU. V tomto jsou další informace.

33.4.5. Chráněný mód Z80

Mikroprocesor Z80, nemá žádný chráněný mód. Ať již uživatelský program, nebo operační systém, můžou pracovat s libovolnou součástí počítače bez omezení. O tom, jak zavést chráněný mód je tento článek.

Nejdříve se zmíním, k čemu je vlastně chráněný mód. Procesor který rozlišuje mezi chráněným módem a uživatelským módem omezuje programy v uživatelském módu v možnosti manipulovat libovolně s celým počítačem. Tato možnost je vyhrazena jen pro programy běžící v chráněném módu, což je například jádro operačního systému.

Není-li přímo v procesoru implementována logika chráněného módu, zůstává nám jediná možnost. Implementovat tuto logiku vně procesoru. Nemůžeme tak program v uživatelském módu omezit v instrukcích, ale můžeme mu efektivně zamezit v manipulaci s připojeným hardware. To učiníme například blokováním I/O pokud není procesor v chráněném módu.

Logika přepínání mezi chráněným a uživatelským módem je tedy vystavena kolem jednobitového registru PMR (*Processor Mode Register*). Řeší dvě základní otázky. Jak zajistit bezpečné přepnutí z uživatelského módu do chráněného módu, a jak zajistit přepnutí z chráněného módu do uživatelského módu.

Protože do chráněného módu se přepínáme jen při volání služeb jádra, můžeme naši logiku napojit právě na tato volání. Stačí nám jediné, detekovat že dochází k volání jádra.

Přiřaďme volání jádra instrukci **RST 30**, a popišme si jak toto volání probíhá. Uživatelský program, který chce použít službu jádra, provede nastavení parametrů volání. Jakým způsobem tak učiní nechme na později. Po nastavení těchto parametrů provede instrukci **RST 30**. Tato instrukce způsobí uložení čítače instrukcí PC na zásobník a přechod na adresu 0030h. Vykonání instrukce na adrese 0030h detekujeme jako provedení cyklu M1 s hodnotou 0030h na adresové sběrnici. Toto je přesně ten impuls, který použijeme k nastavení PMR registru na hodnotu 1.

Na adrese 0030h se nachází počátek programu pro obsluhu volání jádra. Abychom zamezili uživatelskému program v modifikaci tohoto prostoru, je v *User Mode* blokován zápis do paměti na těchto adresách.

* **FIXME:TBD**

Přechod z uživatelského módu do chráněného módu se uskuteční pomocí instrukce **RST 30**. Tato způsobí vykonání kódu na adrese 0x0030. Elektronika detekuje vykonání instrukce na této adrese a změní stav PMR na chráněný mód. Paměťový prostor na adresách 0030 až 0037 je mapován na elektroniku chráněného módu. To znamená že uživatelský program nemůže obsah těchto adres měnit. Program umístěný na těchto adresách musí bezpečně zajistit dokončení přechodu do bezpečného módu.

```

; Při načítání instrukce PUSH AF dojde dojde k přepnutí do chráněného
; Následující manipulace s registry v I/O prostoru tedy nebude blokována.
0030:      PUSH AF      ; uschování akumulátoru a příznaků na uživatelský zásobník

```

```

; Nyní je třeba změnit mapování paměti a nastavit si stránku s jádrem
; operačního systému. Předpokládejme tedy že toto jádro je ve stránce
; 44 a má být mapováno do prostoru PR0 (0000h - 3FFFh).
0031:          LD A, 44    ; Do stránkového registru pro stránku 0
0033:          OUT PR0    ; uložíme číslo stránky s jádrem

; Skok na podprogram obsluhy bolání jádra.
0035: .. 00 06          JMP SUPER ;

; Obsluha přechodu do chráněného režimu
SUPER:
; víme že PR0 obsahoval hodnotu CU_PR0
; dokončíme mapování stránek OS
LD A, 81h

OUT PR1
LD A, 82h

OUT PR2
LD A, 83h
OUT PR3
; uložíme si hodnotu SP do CU_SP
0600: ED 73 nn nn      LD (CU_SP), SP

; uschování hodnot registrů

LD SP, CU_REGS

PUSH BC, DE, HL, IX, IY
EXX AF

PUSH AF

EXX
PUSH BC, DE, HL

```

Přepnutí z chráněného do uživatelského módu lze uskutečnit například přes I/O.

```

; Přepnutí z chráněného do uživatelského módu a návrat
; do uživatelského programu

... obnovení PR registrů mapování paměti

LD A, 1
OUT PMR

```

Jednou z funkcionalit kterou potřebujeme ještě vyřešit je obsluha přerušení. Protože uživatelský program může pomocí instrukce **DI** zakázat přerušení, je jediným způsobem jak použít přerušení nemaskovatelné přerušení NMI. To nás omezuje v možnostech obsluhy více přerušení, neboť je nutno programově zjišťovat zdroj přerušení.

33.4.6. Použití MCU jako periferie k Z80

*

Myšlenka použít jednočipový mikrořadič jako je 40, 39 či jiný je velmi lákavá. Daly by se takto realizovat velmi inteligentní a sofistikované periferní obvody. Problémem ale je, jak propojit MCU se sběrnicí Z80. Většina MCU není totiž vybavena rozhraním kterým by se na sběrnici mohla připojit. Takové rozhraní jsem našel pouze u některých ovodů PIC firmy Microchip. Rozhraní se jmenuje PSP (*Parallel Slave Port*).

Pokud budu chtít realizovat periférii procesorem 40, bude to náročnější. Pro přenos směrem od Z80 do AVR, tedy když se Z80 pokusí zapsat data do AVR, můžeme použít jednoduchý klopný obvod. Výstup tohoto obvodu generuje signál WAIT. V normální situaci je výstup neaktivní, pokud je detekována operace zápisu na adresu obsluhovanou AVR, klopný obvod se překlápí a procesor zustane v čekacím stavu. AVR má nyní čas prozkoumat obsah adresní a datové sběrnice. Poté co zjistí potřebné informace, signálem na jednom pinu překlápí klopný obvod a stav WAIT je ukončen.

Operaci čtení ještě nemám promyšlenou. Začíná ovšem stejně, obvody detekují čtení z AVR a Z80 je vnučen stav WAIT. MCU připraví hodnotu pro datovou sběrnici, uvolní signál wait. MCU nyní musí počkat dostatečně dlouho aby Z80 mohl hodnotu z datové sběrnice přečíst, ale ne déle.

33.5. Konstrukce počítačů s Z80

* *Attributy: id="Z80.constructions"*

Konkrétní konstrukce:

- V6Z80P+²⁴ [2011-03-12, £88 +S&H]
- ZX97 Main Page²⁵
- Mildred²⁶ — Designed and constructed by Chris Pressey [2003-01-10]
- Coprolite Z-80²⁷
-
-
-
- Z80 test²⁸ — jednoduchá konstrukce
- The absolute minimum Z80 CPU test circuit ever seen²⁹
- Rudi's Z80 Computer³⁰ — Z80, ROM, RAM, UART
- Z80 mini System designed by M.Kimura³¹
- V6Z80P³² — v ceně L85 (85 Liber)
- Z80 Microcomputer³³ —
- Z80 Project Homepage³⁴ — cosam.org
- Hans Summers Z80 Computer³⁵ —
- ZX-Badaloc Reloaded³⁶ — klon ZX-Spectrum implementovaný v Spartan-3E FPGA
- Z80 SBC³⁷ — zde nic není jen odkaz na knihu Build your Own Z80 Computer³⁸
- A Homebuilt Z-80 Computer³⁹ —
- Homebrew Z80 Computer⁴⁰ —
- —
-

Počítače schopné provozu CP/M

- CP/M on breadboard⁴¹ —
- The eZ80 Single-Board Computer⁴² — \$250
- MIC80⁴³ — postaveno na EZ80
- Z80 SBC⁴⁴ — rovněž jednoduchá konstrukce jednodeskového počítače
- —

N8VEM

- The N8VEM Home Brew Computer Project⁴⁵ — konstrukce počítače ve stylu 70-tých let.
- N8VEM Single-board Computer⁴⁶
- The N8VEM eight-bit homebrew PC⁴⁷ na Bit-Tech.Net
- Rolf Harrmann⁴⁸

- ECB-Bus⁴⁹
- Eurocard (printed circuit board)⁵⁰
- The N8VEM Home Brew Computer Project⁵¹
- —
- —

Knihy:

- Build your own Z80 computer:⁵² design guidelines and application notes by Steve Ciarcia — kniha volně ke čtení a stažení
-
-

Odkazy:

- Combining a Z80 and an ATmega644P to boot CP/M⁵³ [2010-06-14]
- 86.4
-

Na konstrukci popsané v článku Combining a Z80 and an ATmega644P to boot CP/M⁵⁴ je několik velmi zajímavých momentů.

- nemá ROM paměť. ATmega je zodpovědný za inicializaci RAM při startu počítače.
- jeden klopný obvod obsluhující IORQ a generující WAIT umožňuje použití ATmega jako inteligentní periferie
- hodinový signál pro Z80 je generován z hodinového signálu ATmega dělením dvěma
-

ATmega nemůže kompletně obloužit celou sběrnici Z80. Proto jsou použity dva čipy MCP23S08

Poznámky

1. <http://www.z80.info/>
2. <http://cs.wikipedia.org/wiki/Z80>
3. http://en.wikipedia.org/wiki/Zilog_Z80
4. <http://www.zilog.com/>
5. <http://www.computerhistory.org/collections/accession/102658073>
6. <http://microcodeconsulting.com/z80/>
7. <http://duckduckgo.com/CP/M>
8. http://www.commodore.ca/manuals/128_system_guide/sect-11.htm
9. http://en.wikipedia.org/wiki/Federico_Faggin
10. http://en.wikipedia.org/wiki/Masatoshi_Shima
11. <http://www.ecstaticlyrics.com/electronics/z80/sarcasm/>
12. <http://k1.dyndns.org/www/little-bat.de/prog/zasm/>
13. <http://svn.revsoft.org/spasm/>
14. <http://www.revsoft.org/projects/z80-assembly-ide/>

15. <http://www.unix4fun.org/z80pack/>
16. <http://www.revsoft.org/projects/z80-assembly-ide/>
17. <http://code.google.com/p/wxwabbitemu/>
18. <http://www.revsoft.org/projects/z80-assembly-ide/>
19. <http://www.revsoft.org/projects/z80-assembly-ide/>
20. <http://www.viara.cn/en/j80.htm>
21. <http://www.dougbraun.com/uzi.html>
22. <http://uzix.sourceforge.net/>
23. <http://www.z80.info/z80test0.htm>
24. <http://www.retroleum.co.uk/v6z80p/>
25. http://www.angelfire.com/ab6/rodneynknaap/zx97_e.html
26. <http://catseye.tc/projects/mildred/>
27. <http://www.coprolite.com/z80.html>
28. <http://www.z80.info/gfx/z80test.gif>
29. <http://www.z80.info/z80test0.htm>
30. http://www.knoerig.de/Einplatinencomputer_en.html
31. <http://www.z80.info/z80core4.htm>
32. <http://www.retroleum.co.uk/electronics-articles/v6z80p/>
33. <http://blake-foster.com/project.php?p=16>
34. <http://www.cosam.org/projects/z80/>
35. <http://www.hanssummers.com/newz80.html>
36. http://www.zxbada.bbk.org/badaloc_fpga/index.htm
37. <http://retro.hansotten.nl/index.php?page=z80-sbc>
38. <http://z80.retro8bits.com/z80docs/BuildYourOwnZ80.pdf>
39. <http://cpuville.com/Z80.htm>
40. <http://digitalworld.uw.hu/z80.htm>
41. <http://home.micros.users.btopenworld.com/cpm/index.htm>
42. <http://www.ez80sbc.com/index.htm>
43. <http://www.shaels.net/index.php/mic80>
44. http://tinymicros.com/wiki/Z80_SBC
45. <http://n8vem-sbc.pbworks.com/>
46. <http://www.classiccmp.org/cini/n8vem.htm>
47. <http://www.bit-tech.net/news/2009/01/12/the-n8vem-eight-bit-homebrew-pc/1>
48. <http://www.hd64180-ecb.de/html/n8vem.html>
49. <http://de.wikipedia.org/wiki/ECB-Bus>
50. [http://en.wikipedia.org/wiki/Eurocard_\(PCB\)](http://en.wikipedia.org/wiki/Eurocard_(PCB))
51. <http://n8vem-sbc.pbworks.com/>

52. <http://books.google.cz/books?id=mVQnFgWzX0AC>
53. <http://benryves.com/journal/3662496>
54. <http://benryves.com/journal/3662496>

Kapitola 34. RCA CDP1802 (1805)

* *Attributy: id="cdp1802"*

Odkazy:

- <ftp://ftp.comlab.ox.ac.uk/pub/Cards/txt/1802.txt>
- "Membership Card" tests by Todd Decker¹
- Lee Hart's 1802 "Membership Card"²
- Micro-Elf Kits³
- elf-emulation.com/hardware⁴
- RCA 1802⁵ na Wikipedii
- RCA COSMAC 1802⁶ na The Antique Chip Collector's Page
- A Short Course In Programming⁷ by Tom Pittman
- COSMAC CDP1802 Processor⁸
- Build a PERSONAL MICROCOMPUTER for \$100⁹
-
-
-

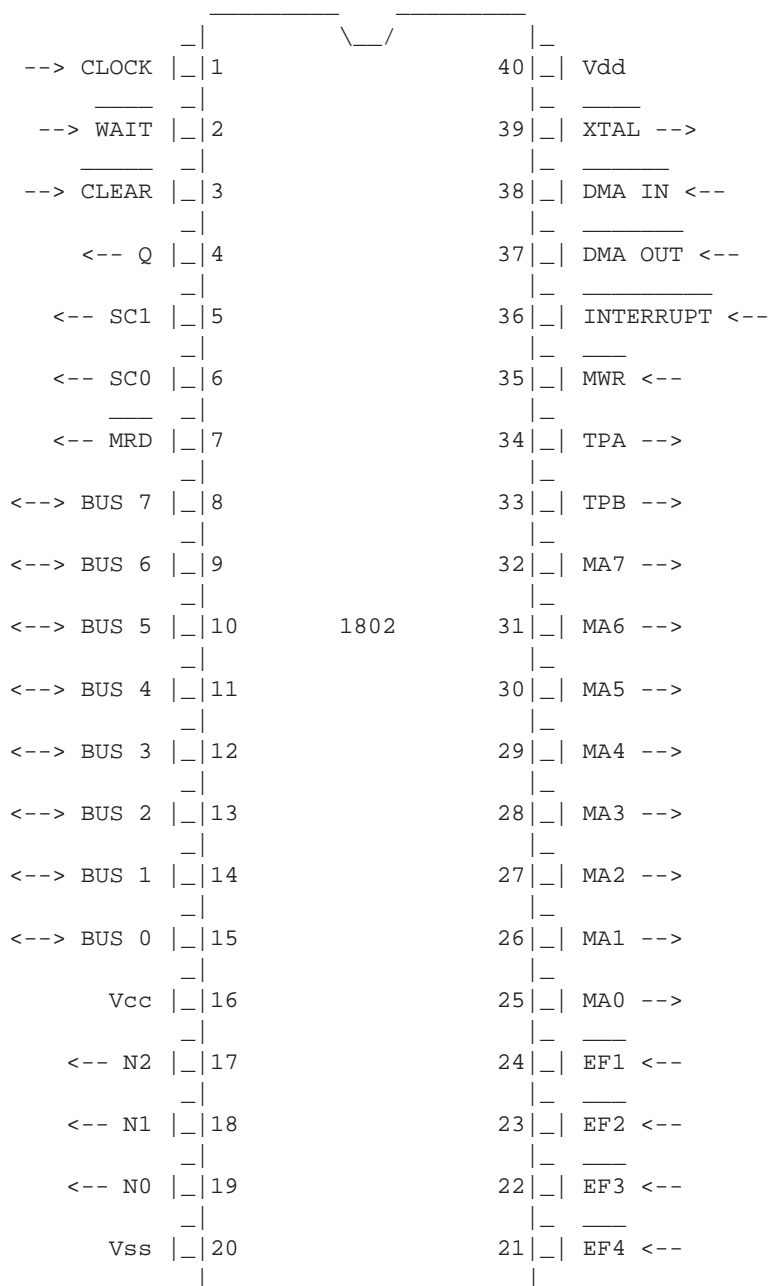
Počítače s procesory 1802:

- RCA COSMAC VIP¹⁰
- COSMAC ELF¹¹
- 1802 World¹²
- COMX 35¹³
- the Cosmac 1802 handheld computer¹⁴
-
-
-
-

První mikroporcesor vyrobený CMOS technologií, alespoň co já vím. Tento mikroporcesor má, tak jako řada stařších mikroprocesorů a počítačů, trochu zvláštní a na dnešní dobu neobvyklou konstrukci/strukturu. Jeho největší zvláštností je to že nemá pevně stanovený ukazatel na instrukce (*program counter*). Má 16 16-ti bitových registrů z nich každý může být ukazatelem na instrukce. Dále tento procesor nemá zásobník a instrukce pro volání a návrat z podprogramu.

Obrázek 34-1. CDP1802 COSMAC Microprocessor Instruction Set Summary

RCA									
1	88888	000	22222						
11	8 8	0 0	2 2						
1	8 8	0 0 0	2						
1	88888	0 0 0	222						
1	8 8	0 0 0	2						
1	8 8	0 0	2						
111	88888	000	2222222						
CDP1802 COSMAC Microprocessor Instruction Set Summary									



Written by Jonathan Bowen
 Programming Research Group
 Oxford University Computing Laboratory
 8-11 Keble Road
 Oxford OX1 3QD
 England

Kapitola 34. RCA CDP1802 (1805)

	Tel +44-865-273840	
Created	August 1981	
Updated	April 1985	
Issue	1.3	Copyright (C) J.P.Bowen 1985

Mnem.	Op	F	Description	Notes
ADC	74	*	Add with Carry	{DF,D}=mx+D+DF
ADCI	i 7C	*	Add with Carry Immediate	{DF,D}=mp+D+DF,p=p+1
ADD	F4	*	Add	{DF,D}=mx+D
ADI	i FC	*	Add Immediate	{DF,D}=mp+D,p=p+1
AND	F2	*	Logical AND	D={mx}&D
ANI	i FA	*	Logical AND Immediate	D={mp}&D,p=p+1
B1	a 34	-	Branch if EF1	If EF1=1 BR else NBR
B2	a 35	-	Branch if EF2	If EF2=1 BR else NBR
B3	a 36	-	Branch if EF3	If EF3=1 BR else NBR
B4	a 37	-	Branch if EF4	If EF4=1 BR else NBR
BDF	a 33	-	Branch if DF	If DF=1 BR else NBR
BGE	a 33	-	Branch if Greater or Equal	See BDF
BL	a 38	-	Branch if Less	See BNF BR else NBR
BM	a 38	-	Branch if Minus	See BNF
BN1	a 3C	-	Branch if Not EF1	If EF1=0 BR else NBR
BN2	a 3D	-	Branch if Not EF2	If EF2=0 BR else NBR
BN3	a 3E	-	Branch if Not EF3	If EF3=0 BR else NBR
BN4	a 3F	-	Branch if Not EF4	If EF4=0 BR else NBR
BNF	a 38	-	Branch if Not DF	If DF=0 BR else NBR
BNQ	a 39	-	Branch if Not Q	If Q=0 BR else NBR
BNZ	a 3A	-	Branch if D Not Zero	If D=1 BR else NBR
BPZ	a 33	-	Branch if Positive or Zero	See BDF
BQ	a 31	-	Branch if Q	If Q=1 BR else NBR
BR	a 30	-	Branch	pl=mp
BZ	a 32	-	Branch if D Zero	If D=0 BR else NBR
DEC	r 2N	-	Decrement register N	n=n-1
DIS	71	-	Disable	{X,P}=mx,x=x+1,IE=0
GHI	r 9N	-	Get High register N	D=nh
GLO	r 8N	-	Get Low register N	D=nl
IDL	00	-	Idle (wait for DMA or int.)	Bus=m0
INC	r 1N	-	Increment register N	n=n+1
INP	d 6N	-	Input (N=d+8=9-F)	mx=Bus,D=Bus,Nlines=d
IRX	60	-	Increment register X	x=x+1
LBDF	a C3	-	Long Branch if DF	If DF=1 LBR else LNBR
LBNF	a C8	-	Long Branch if Not DF	If DF=0 LBR else LNBR
LBNQ	a C9	-	Long Branch if Not Q	If Q=0 LBR else LNBR
LBNZ	a CA	-	Long Branch if D Not Zero	If D=1 LBR else LNBR
LBQ	a C1	-	Long Branch if Q	If Q=1 LBR else LNBR
LBR	a C0	-	Long Branch	p=mp
LBZ	a C2	-	Long Branch if D Zero	If D=0 LBR else LNBR
LDA	r 4N	-	Load advance	D=mn,n=n+1
LDI	i F8	-	Load Immediate	D=mp,p=p+1
LDN	r 0N	-	Load via N (except N=0)	D=mn
LDX	F0	-	Load via X	D=mx
LDXA	72	-	Load via X and Advance	D=mx,x=x+1
LSDF	CF	-	Long Skip if DF	If DF=1 LSKP else NOP
LSIE	CC	-	Long Skip if IE	If IE=1 LSKP else NOP
LSKP	C8	-	Long Skip	See NLBR

LSNF	C7	-	Long Skip if Not DF	If DF=0 LSKP else NOP
LSNQ	C5	-	Long Skip if Not Q	If Q=0 LSKP else NOP
LSNZ	C6	-	Long Skip if D Not Zero	If D=1 LSKP else NOP
LSQ	CD	-	Long Skip if Q	If Q=1 LSKP else NOP
LSZ	CE	-	Long Skip if D Zero	If D=0 LSKP else NOP
MARK	79	-	Push X,P to stack (T={X,P})	m2={X,P}, X=P, r2=r2-1
NBR	38	-	No short Branch (see SKP)	p=p+1
NLBR a	C8	-	No Long Branch (see LSKP)	p=p+2
NOP	C4	-	No Operation	Continue
OR	F1	*	Logical OR	D={mx}vD
ORI i	F9	*	Logical OR Immediate	D={mp}vD, p=p+1
OUT d	6N	-	Output (N=d=1-7)	Bus=mx, x=x+1, Nlines=d
PLO r	AN	-	Put Low register N	nl=D
PHI r	BN	-	Put High register N	nh=D
REQ	7A	-	Reset Q	Q=0
RET	70	-	Return	{X,P}=mx, x=x+1, IE=1
RSHL	7E	*	Ring Shift Left	See SHLC
RSHR	76	*	Ring Shift Right	See SHRC
SAV	78	-	Save	mx=T
SDB	75	*	Subtract D with Borrow	{DF,D}=mx-D-DF
SDBI i	7D	*	Subtract D with Borrow Imm.	{DF,D}=mp-D-DF, p=p+1
SD	F5	*	Subtract D	{DF,D}=mx-D
SDI i	FD	*	Subtract D Immediate	{DF,D}=mp-D, p=p+1
SEP r	DN	-	Set P	P=N
SEQ	7B	-	Set Q	Q=1
SEX r	EN	-	Set X	X=N
SHL	FE	*	Shift Left	{DF,D}={DF,D,0}<-
SHLC	7E	*	Shift Left with Carry	{DF,D}={DF,D}<-

Mnem.	Op	F	Description	Notes
SHR	F6	*	Shift Right	{D,DF}=->{0,D,DF}
SHRC	76	*	Shift Right with Carry	{D,DF}=->{D,DF}
SKP	38	-	Short Skip	See NBR
SMB	77	*	Subtract Memory with Borrow	{DF,D}=D-mx-{~DF}
SMBI i	7F	*	Subtract Mem with Borrow Imm	{DF,D}=D-mp-~DF, p=p+1
SM	F7	*	Subtract Memory	{DF,D}=D-mx
SMI i	FF	*	Subtract Memory Immediate	{DF,D}=D-mp, p=p+1
STR r	5N	-	Store via N	mn=D
STXD	73	-	Store via X and Decrement	mx=D, x=x-1
XOR	F3	*	Logical Exclusive OR	D={mx}.D
XRI i	FB	*	Logical Exclusive OR Imm.	D={mp}.D, p=p+1
		-	Interrupt action	T={X,P}, P=1, X=2, IE=0
	??		8-bit hexadecimal opcode	
	?N		Opcode with register/device in low 4/3 bits	
		-	DF flag unaffected	
		*	DF flag affected	
mn			Register addressing	
mx			Register-indirect addressing	
mp			Immediate addressing	
R()			Stack addressing (implied addressing)	
DFB n(,n)			Define Byte	
DFS n			Define Storage block	

Kapitola 34. RCA CDP1802 (1805)

DFW n(,n)	Define Word
D	Data register (accumulator, 8-bit)
DF	Data Flag (ALU carry, 1-bit)
I	High-order instruction digit (4-bit)
IE	Interrupt Enable (1-bit)
N	Low-order instruction digit (4-bit)
P	Designates Program Counter register (4-bit)
Q	Output flip-flop (1-bit)
R	1 of 16 scratchpad Registers(16-bit)
T	Holds old {X,P} after interrupt (X high, 8-bit)
X	Designates Data Pointer register (4-bit)
mn	Memory byte addressed by R(N)
mp	Memory byte addressed by R(P)
mx	Memory byte addressed by R(X)
m?	Memory byte addressed by R(?)
n	Short form for R(N)
nh	High-order byte of R(N)
nl	Low-order byte of R(N)
p	Short form for R(P)
pl	Low-order byte of R(P)
r?	Short form for R(?)
x	Short form for R(X)
R(N)	Register specified by N
R(P)	Current program counter
R(X)	Current data pointer
R(?)	Specific register
a	Address expression
d	Device number (1-7)
i	Immediate expression
n	Expression
r	Register (hex digit or an R followed by hex digit)
+	Arithmetic addition
-	Arithmetic subtraction
*	Arithmetic multiplication
/	Arithmetic division
&	Logical AND
~	Logical NOT
v	Logical inclusive OR
.	Logical exclusive OR
<-	Rotate left
->	Rotate right
{ }	Combination of operands
?	Hexadecimal digit (0-F)
-->	Input pin
<--	Output pin
<-->	Input/output pin

34.1. Obvody rodiny CDP 1802

*

Odkazy:

- 1800-Series Chip Identification¹⁵
-

Tabulka 34-1. Přehled obvodů rodiny CDP 1802

označení	popis
CDP1801CRD	"Microprocessor Register", registry a alu
CDP1801CUD	"Microprocessor Control", řízení (59 instrukcí)
CDP1802	mikroprocesor
CDP1804	mikroprocesor, 113 instrukcí, vylepšená verze 1802, ROM 2KB, RAM 64B
CDP1805	mikroprocesor, 123 instrukcí, rozšířená verze 1802, 64B RAM
CDP1806	mikroprocesor, varianta 1805, místo nožičky !ME má GND.
CDP1821	RAM 1K*1
CDP1822	RAM 256*4
CDP1823	RAM 128*8
CDP1824	RAM 32*8
CDP1825	RAM 1K*4
CDP1831	ROM 512*8 (maskou programovaná)
CDP1832	ROM 512*8 (maskou programovaná)
CDP1833	ROM 1K*8 (maskou programovaná)
CDP1834	ROM 1K*8 (maskou programovaná)
CDP1835	ROM 2K*8 (maskou programovaná)
CDP18U42	UV EPROM 256*8
CDP1851	PIO (Programmable I/O)
CDP1852	Byte I/O - 8-bit I/O port
CDP1853	Decoder 1 z 8
CDP1854A	UART
CDP1855	MDU (Multiply/Divide Unit)
CDP1856	Buffer 4- bit
CDP1857	Buffer 4- bit
CDP1858	Latch/Decoder - 4-bit
CDP1859	Latch/Decoder - 4-bit
CDP1861	VDC (Video Display Controller) "Pixie"
CDP1862	Color Generator Circuit
CDP1863	Programmable Frequency Generator
CDP1864	VDC (PAL Video Display Controller)
CDP1866	Latch/Decoder - 4-bit
CDP1867	Latch/Decoder - 4-bit

označení	popis
CDP1868	Latch/Decoder - 4-bit
CDP1869	VIS (Video Interface System)
CDP1870	VIS (Video Interface System)
CDP1871A	Keyboard Encoder, ASCII/HEX
CDP1872	High-Speed Input Port - 8-bit
CDP1873	High-Speed Decoder - 1 of 8
CDP1874	High-Speed Input Port - 8-bit
CDP1875	High-Speed Output Port
CDP1876	VIS (Video Interface System)
CDP1877	Programmable Interrupt Controller
CDP1878	Dual-Counter Timer
CDP1879	Real Time Clock
CDP6402	Industry Standard UART
CDP6551	UART with baud rate generator

34.1.1. CDP 1802

*

Obrázek 34-2. Vývody CDP 1802 v pouzdru DIL40

	+-----+	+-----+	
CLOCK -	1	___	40 - Vdd
!WAIT -	2		39 - !XTAL
!CLEAR -	3		38 - !DMA IN
Q -	4		37 - !DMA OUT
SC1 -	5		36 - !INTERRUPT
SC0 -	6		35 - !MWR
!MRD -	7		34 - TPA
BUS 7 -	8		33 - TPB
BUS 6 -	9		32 - MA7
BUS 5 -	10		31 - MA6
BUS 4 -	11		30 - MA5
BUS 3 -	12		29 - MA4
BUS 2 -	13		28 - MA3
BUS 1 -	14		27 - MA2
BUS 0 -	15		26 - MA1
Vcc -	16		25 - MA0
N2 -	17		24 - !EF1
N1 -	18		23 - !EF2
N0 -	19		22 - !EF3
Vss -	20		21 - !EF4
	+-----+		

TPA

Signál oznamující že na MA0-MA7 jsou bity 8-15 adresy

TPB

Signál oznamující že na MA0-MA7 jsou bity 0-7 adresy

!CLEAR

!WAIT

Nastavují procesor do jednoho ze čtyř módů.

!CLEAR	!WAIT	MODE
L	L	LOAD
L	H	RESET
H	L	PAUSE
H	H	RUN _x

34.1.2. CDP 1804

*

Obvod obsahuje procesor CDP 1802, má stejné zapojení vývodů. Navíc obsahuje maskou programovatelnou paměť ROM 2KB, paměť RAM 64B a čítač/časovač.

Obrázek 34-3.

	+-----+	+-----+	
CLOCK -	1	___	40 - Vdd
!WAIT -	2		39 - !XTAL
!CLEAR -	3		38 - !DMA IN
Q -	4		37 - !DMA OUT
SC1 -	5		36 - !INTERRUPT
SC0 -	6		35 - !MWR
!MRD -	7		34 - TPA
BUS 7 -	8		33 - TPB
BUS 6 -	9		32 - MA7
BUS 5 -	10		31 - MA6
BUS 4 -	11		30 - MA5
BUS 3 -	12		29 - MA4
BUS 2 -	13		28 - MA3
BUS 1 -	14		27 - MA2
BUS 0 -	15		26 - MA1
!EMS -	16		25 - MA0
N2 -	17		24 - !EF1
N1 -	18		23 - !EF2
N0 -	19		22 - !EF3
Vss -	20		21 - !EF4
	+-----+		

34.2. Instrukce CDP 1802 (1805)

Tabulka 34-2. Operační kódy instrukcí CDP1802 (1805) řazené podle operačního kódu

název instrukce	mnemo	strojový kód				popis
		bin	oc- tal	dec	hexa	
ldn	LDN	0000 nnnn	000	0	0n	Load via N
bdf	BDF	0011 0011	063	51	33	Branch if DF
bge	BGE	0011 0011	063	51	33	Branch if Greater or Equal
bl	BL	0011 1000	070	56	38	Branch if Less
irx	IRX	0110 0000	140	96	60	Increment register X
adc	ADC	0111 0100	164	116	74	Add with Carry
adci	ADCI i	0 1111 100i	370	248	7c ii	Add with Carry Immediate
nop	NOP	1100 0100	304	196	c4	No Operation
ldx	LDX	1111 0000	360	240	f0	Load via X
and	AND	1111 0010	362	242	f2	Logical And
add	ADD	1111 0100	364	244	f4	Add
ani	ANI	1 1111 010i	764	500	fa ii	Logical And Immediate
adi	ADI i	1 1111 100i	770	504	fc ii	Add Immediate

ADC

Jméno

ADC — Add with Carry

Přehled

ADC

Popis

$\{DF, D\} \leftarrow mx + D + DF$

Odkazy

Související instrukce: cdp1802.isa.add;, ADCI, cdp1802.isa.adi;.

·
·

ADCI

Jméno

ADCI — Add with Carry Immediate

Přehled

ADCI

Popis

$\{DF, D\} \leftarrow mx + D + DF, p = p + 1$

Odkazy

Související instrukce: ADC, cdp1802.isa.add;, cdp1802.isa.adi;.

·
·

ADD

Jméno

ADD — Add

Přehled

ADD

Popis

$\{DF,D\} \leftarrow mx+D$

Odkazy

Související instrukce: ADC, ADCI, cdp1802.isa.adi;.

·
·

ADI

Jméno

ADI — Add Immediate

Přehled

ADI i

Popis

$\{DF,D\} \leftarrow mx+D, p=p+1$

Odkazy

Související instrukce: ADC, ADCI, ADD.

·
·

AND

Jméno

AND — Logical AND

Přehled

AND i

Popis

$D \wedge \text{Mem}[\text{Reg}[X]] \longrightarrow D$

Odkazy

Související instrukce: ADD.

·
·

ANI

Jméno

ANI — Logical AND Immediate

Přehled

ANI i

Popis

$D \wedge \text{Mem}[\text{Reg}[P]] \longrightarrow D, R_p = R_p + 1$

Odkazy

Související instrukce: ADI, AND.

·
·

BDF

Jméno

BDF , BGE — Branch if DF

Přehled

BDF

BGE

Popis

If DF=1 then branch;

Odkazy

Související instrukce: LDX.

·
·

BL

Jméno

BL — Branch if Less

Přehled

BL

Popis

If ... then branch;

Odkazy

Související instrukce: BDF, BGE.

·
·

IRX

Jméno

IRX — Increment register X

Přehled

IRX

Popis

Odkazy

Související instrukce: LDX.

•
•

LDN

Jméno

LDN — Load via N

Přehled

LDN

Popis

$M[R_n] \longrightarrow D$; for n not 0

Odkazy

Související instrukce: LDX.

•
•

LDX

Jméno

LDX — Load via X

Přehled

LDX

Popis

Odkazy

Související instrukce: IRX.

·
·

NOP

Jméno

NOP — No Operation

Přehled

NOP

Popis

Odkazy

Související instrukce: IRX.

•
•

Poznámky

1. http://www.retrotechnology.com/memship/mem_decker.html
2. <http://www.retrotechnology.com/memship/memship.html>
3. <http://www.elf-emulation.com/microelf.html>
4. <http://www.elf-emulation.com/hardware.html>
5. http://en.wikipedia.org/wiki/CDP_1802
6. <http://www.antiquetech.com/chips/RCA1802.htm>
7. <http://www.ittybittycomputers.com/IttyBitty/ShortCor.htm>
8. <http://www.tedrossin.0sites.net/Electronics/RCA/RCA.html>
9. <http://penguincentral.com/retrocomputing/ElfII/>
10. <http://oldcomputers.net/rca-cosmac-vip.html>
11. <http://www.cosmacelf.com/>
12. <http://ken.krother.com/1802/>
13. <http://www.old-computers.com/museum/computer.asp?st=1&c=110>
14. <http://ringcomps.co.uk/hhc/>
15. <http://www.decodesystems.com/cosmac/#chipset>

Kapitola 35. Motorola 6809

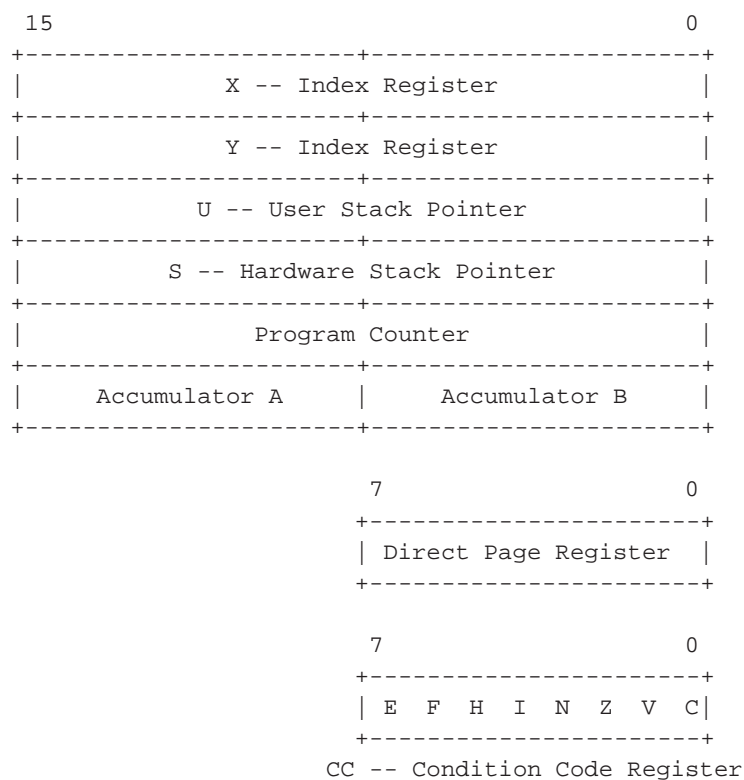
Odkazy:

- [Motorola 6809¹](#) From Wikipedia. (2²)
- [6809E 8-bit microprocessor³](#) — vývody a základní specifikace
-
-
-

Tento procesor je následníkem modelu 6800. Je to jeden z nejlépe navržených osmibitových mikroprocesorů vůbec. Oproti ostatním se liší zejména v možnosti psát kód nezávislý na absolutních adresách.

35.1. Registry

Obrázek 35-1. Registry MC6809



35.2. Programovací nástroje a prostředí

Odkazy:

- [ciforth for 6809⁴](#)
-
- [Homebrew 6809 Computer: First Test⁵](#)
- [Homebrew 6809 Computer: Hello World⁶](#)

- Homebrew 6809 Computer + Terminalscope: Monitor, Disassembler⁷
-

35.3. Konstrukce založené na 6809

Odkazy:

-

Ultimate 8-Bit Computer:

- [ultim809⁸](#)
- Homebrew 6809 Computer: First Test⁹ — na YouTube
- Homebrew 6809 Computer: Hello World¹⁰
- Homebrew 6809 Computer (work in progress)¹¹ — obrázky na flickr.com
-

-

Ultime

Poznámky

1. <http://en.wikipedia.org/wiki/index.html?curid=20326>
2. <http://en.wikipedia.org/wiki/6809>
3. <http://www.msarnoff.org/chipdb/6809E>
4. <http://home.hccnet.nl/a.w.m.van.der.horst/m6809forth.html>
5. <http://www.youtube.com/watch?v=IXu8P2qGoj8>
6. <http://www.youtube.com/watch?v=4bU1iyKxkCI>
7. <http://www.youtube.com/watch?v=BP7sjnVzTqw>
8. <http://www.msarnoff.org/projects/6809/>
9. <http://www.youtube.com/watch?v=IXu8P2qGoj8>
10. <http://www.youtube.com/watch?v=4bU1iyKxkCI>
11. <http://www.flickr.com/photos/74hc595/sets/72157623669618653/>

Kapitola 36. 6502

Odkazy:

- An expandable 6502 SBC¹
- Visual Transistor-level Simulation of the 6502 CPU² and other chips!
-
-

Poznámky

1. <http://members.lycos.co.uk/leedavison/6502/sbc/index.html>
2. <http://www.visual6502.org/welcome.html>

Kapitola 37. SC/MP

Odkazy:

-
- SC/MP revisited¹ A SC/MP micro-computer with NIBL Tiny-BASIC
- ISP-8A/600 single-chip 8-bit n-channel Microprocessor (SC/MP 2)²

Poznámky

1. <http://www.dos4ever.com/SCMP/SCMP.html>
2. <http://uk.randominter.net/mymk14.co.uk/paulRobson/scmp.htm>

Kapitola 38. MIPS32

Architektura MIPS patří mezi původní staré RISC architektury.

38.1. Registry

Instrukce procesoru umožňují pracovat s 32 obecnými registry, každý 32 bitů veliký. Tyto registry jsou označovány čísla jako \$0, \$1, ... \$31. Některé z těchto registrů mají speciální význam a použití ostatní vymezují konvence.

\$0

Tento registr obsahuje na pevnou hodnotu nula. Při čtení vždy přečteme nulu a zápis do tohoto registru je ignorován.

\$31

Link registr. Do tohoto registru ukládá instrukce jal návratovou adresu.

Čítač instrukcí, není registr. Protože architektura MIPS používá proudové zpracování instrukcí, není rozumné uvažovat v termínech jako je čítač instrukcí.

Tabulka 38-1. Konvence pojmenování a použití registrů

číslo registru	jméno	použití
0	zero	konstanta 0
1	at	(assembly temporary) dočasný registr
2-3	v0, v1	value hodnoty vracené podprogramem, výsledky
4-7	a0-a3	arguments parametry podprogramu, argumenty
8-15	t0-t7	dočasné hodnoty, k volnému užití podprogramem
24,25	t8, t9	dočasné hodnoty, k volnému užití podprogramem
16-23	s0-s7	
26,27	k0, k1	reservováno pro užití obsluhou přerušení
28	gp	Global pointer
29	sp	Stack pointer
30	s8/fp	
31	ra	Return address

Kapitola 39. PIC

*

Odkazy:

- David Tait's PIC Archive¹
- PICLIST.COM²
- CHEAPIC³
- PikLab⁴ — IDE for PIC microcontrollers
- Intro to Programming PIC Microcontrollers in Linux⁵
- Pic Cross Development Support Tools⁶
- Eric's PIC Page⁷
- PICPROJECTS⁸ for 12F & 16F PIC Microcontrollers
-
-

39.1. Programátory pro PIC

Odkazy:

- ZPL: a Zero Pin Loader for the PICmicro 18F family⁹
- PikDev Hardware¹⁰
- HOODMICRO¹¹ (\$1.95 za PCB, \$60 komplet)
- Byron Jeff's Trivial HVP Programmer¹² — na paralelní port
- Open Programmer v0.7.x¹³
-

39.1.1.

*

Odkazy:

- UsbPicProg.org¹⁴
- usbpicprog: Simple USB PIC programmer¹⁵
-

39.2. Nástroje a prostředí pro vývoj programů

*

Odkazy:

- How-to: Program PICs using Linux¹⁶ na HACK A DAY [2010-11-03]
-

39.2.1. Forth

Odkazy:

- FLASHFORTH for the PIC18F and the dsPIC30F¹⁷
-

39.3. Konstrukce s PICy

To be done:

- DL4YHF's Audio-Utilities: A/D converter for the serial port¹⁸
-

39.3.1. Čítač / měřič kmitočtu

Odkazy:

- 2.5 GHz Frequency counter¹⁹
- Frequency counter with a PIC and minimum hardware²⁰ by Wolfgang "Wolf" Büscher, DL4YHF
- The Weeder Frequency Counter PIC 16F84 port²¹ by Peter Cousens
- Universal frequency meter²²
- PIC frequency meters/digital scales²³
-

PIC

PIC <http://www.foxdelta.com/products/fc1.htm>

39.4. Řešení různých problémů

Odkazy:

- PIC - RAM Extension with 32k/64k SRAM²⁴

Poznámky

1. <http://www.piclist.com/techref/microchip/davidtait/index.html>
2. <http://www.piclist.com/techref/piclist/index.htm>
3. <http://www.piclist.com/techref/piclist/cheapic/index.htm>
4. <http://piklab.sourceforge.net/>
5. <http://www.micahcarrick.com/pic-programming-linux.html>
6. <http://www.cs.uiowa.edu/~jones/cross/pic/>
7. <http://www.brouhaha.com/~eric/pic/>
8. <http://picprojects.org.uk/>
9. <http://www.circuitcellar.com/flash2002/Honorable/M285-abstract.htm>
10. <http://pikdev.free.fr/hardware.html>
11. <http://www.k9spud.com/hoodmicro/>
12. <http://www.finitesite.com/d3jsys/proghvp.html>
13. <http://openprog.altervista.org/>

14. <http://usbpicprog.org/>
15. <http://sourceforge.net/projects/usbpicprog/>
16. <http://hackaday.com/2010/11/03/how-to-program-pics-using-linux/>
17. <http://flashforth.sourceforge.net/>
18. <http://freenet-homepage.de/dl4yhf/soundutl/serpicad.htm>
19. <http://hem.passagen.se/communication/frcpll.html>
20. http://freenet-homepage.de/dl4yhf/freq_counter/freq_counter.html#cc_or_ca_display
21. <http://www.piclist.com/techref/piclist/weedfreq.htm>
22. http://www.qsl.net/yo5ofh/pic/freq_counter_1300mhz/vhf&uhf_universal_frequency_meter.htm
23. <http://www.qsl.net/om3cph/counter/counter.html>
24. <http://www.captain.at/electronics/pic-ram/>

Kapitola 40. AVR

Odkazy:

- ATMEL AVR® 8-Bit RISC¹ — domovské stránky AVR na webu firmy ATMEL
- AVR on Wikipedia²
- AVR Beginners NET (asm tutorial)³
- AVR freaks⁴
- Programujeme jednočipy⁵ (Wikikniha)

Konstrukce:

- RGB lampička⁶

Vývojové nástroje a programátory:

- Linux Development Tools for Atmel AVR⁷
- UISP - AVR In-System Programmer⁸
- AVR Bootloader and Programmer⁹
- Download page¹⁰
- Programming the AVR microcontroller with GCC, libc 1.0.4¹¹
- AVR MCU programmer¹² ultra low-cost programmer for AT90S1200, 2313, 2323, 2343, 4414 4434 8515 8535
- PonyProg serial device programmer¹³
- ispRE: Do-it-yourself AVRISP-compatible Programmer¹⁴
- Using AVR microcontrollers: Minimalist target boards¹⁵
- Simple AVR Programmers¹⁶
-

DAPA programátory na paralelní port

- Programator ISP¹⁷
- DaPa == Direct AvR Parallel Access¹⁸
- The Arduino Boot-Cloner¹⁹
- Flashen für jedermann mit dem ISP-Programmieradapter²⁰
- Programátor na paralelní port²¹
- Programming the AVR Microcontroller with GCC²²
- Atmel AVR pod Linuxom²³
- AVR STK Parallel Port Dongle Programmer²⁴
- Trio AVR ISP: tři zařízení zároveň²⁵
- AVR In-Circuit Programmer²⁶
- AVR Programming Adaptor using 74LS244²⁷
- Sample Electronics Programmer²⁸

•

DASA programátory na sériový port

- HOWTO get started with an AVR at minimal cost²⁹
- Getting Started With Avrs, Too much info!!³⁰
- Beginner's Microcontroller Programming-III: (Further) Simplification of the LancOs's SI Prog the Serial Port Porgrammer³¹

•

Programátory na USB port

- USB AVR-ISP³²
- USBtiny³³

- ISP Header - 6 way IDC - Atmel SPI pin-out³⁴
- AVR ISP programmer Adapter KIT 10 to 6 pin³⁵
- The I106 is an ISP converter for AVR ISP programmer³⁶
- STK to breadboard adapter³⁷

Procesory AVR se dělí do několika základních rodin, pokusil jsem se je seřadit v tabulce. Rodina ATtiny zahrnuje nejmenší obvody. Zde se nacházejí i obvody s 8 vývody. Jsou určeny pro jednoduché aplikace kde není potřeba k MCU připojovat více obvodů.

Další rodinou je nejrozšířenější rodina ATmega. Zde jsou běžné obvody s bohatým periferním vybavením a dostatkem paměti programu i pro náročnější aplikace.

Další dvě rodiny jsou specializované, rodině XMEGA má větší paměť programu a poslední rodina zvláštních procesorů zahrnuje procesory s programovatelným logickým polem na jednom čipu.

Tabulka 40-1.

rodina	paměť programu	instrukční sada	vývody	periferie
tiny	1-8KB	základní	8-32	omezené
mega	4-256KB	rozšířená	28-100	rozšířené
XMEGA	16-384KB	rozšířená	44-100	rozšířené
zvláštní				

Následující tabulka obsahuje vybrané parametry vybraných procesorů. Není rozhodně v žádném směru úplná.

Tabulka 40-2. Přehled vybraných obvodů řady AVR ATtiny

typ	IO	paměť			Vcc [V]	f [MHz]	pouzdro	periferie
		Fla.	RAM	EEP.				
4	4	512B	32		0-4 MHz @ 1.8-5.5V 0-8 MHz @ 2.7-5.5V 0-12 MHz @ 4.5-5.5V	0-20 SOT-23 TQFP-28 TQFP-44 ADC		
5								
9		1024B						
10							ADC	
11L	6	1KB			2.7 - 5.5	0 - 2	PDIP8	zastaralý, nemá ISP
11					4.0 - 5.5	0 - 6		
12V	6	1KB	32B	64B	1.8 - 5.5	0 - 1.2	PDIP8	zastaralý
12L					2.7 - 5.5	0 - 4		
12					4.0 - 5.5	0 - 8		
13	6	1KB	64B	64B	2.7 - 5.5	0 - 20	PDIP8	4-ch 10-bit A/D, WDT, Timer with 2 PWM channels
13V					1.8 - 5.5	0 - 10		

typ	IO	paměť			Vcc [V]	f [MHz]	pouzdro	periferie
		Fla.	RAM	EEP.				
13A					1.8 - 5.5	0 - 20		% + picoPower, 190 μ A při 1.8V a 1MHz
15L	6	1KB	-	64	2.7 - 5.5	1.6 int.	PDIP8	timer, 4ch 10bit ADC, WDT, zastaralý, nahrazen ATtiny2
22L	6	2KB	128B	128B				zastaralý
2323								
2343								
24	12	2KB	128B	128B			PDIP14	2 timers, 8 ADC, WDT, US
44		4KB	256B	256B				
84		8KB	512B	512B				
25	6	2KB	128B	128B	2.7 - 5.5	0 - 16	PDIP8	2 timers, 4 ADC, WDT, US 24/44/84 v PDIP8
45		4KB	256B	256B				
85		8KB	512B	512B				
26	16	2KB	128B	128B	2.7 - 5.5	0 - 16		zastaralý
28	16	2KB			1.8 - 5.5	0 - 4		zastaralý
261	16	2KB	128B	128B	2.7 - 5.5	0 - 16	PDIP20	2 timers +3PWM, 11 ADC 10bit, WDT, USI
461		4KB	256B	256B				
861		8KB	512B	512B				
2313A	18	2KB	128B	128B	1.8 - 5.5	0 - 20	PDIP20, SOIC-20, MLF/VQFN-20	8-bit Timer/Counter, 16-bit Timer/Counter, 4*PWM, Analog Comparator, WDT, USI, Full Duplex USART
4313		4KB	256B	256B				

Zjednodušený přehled ATtiny:

- **ATtiny 24/44/84** — obvody s 12 GPIO. Pouzdro s 14 vývody (SOIC a PDIP). Periferie: T0 8-bit čítač s 2 PWM, T1 16-bit čítač s 2 PWM, 10-bit ADC, WDT, USI, dW, ISP via SPI
- **ATtiny 25/45/85** — obvody s 6 GPIO. Nejmenší pouzdro s 8-mi vývody. Periferie: T0 8-bit čítač/časovač s 2 PWM, T1 HS 8-bit čítač, USI, 10-bit ADC, WDC, dW, ISP via SPI
- **ATtiny 261/461/861** — obvody s 16 GPIO. Pouzdro s 20 vývody (SOIC). Periferie: T0 8/16-bit čítač/časovač, T1 HS 8/10-bit čítač s PWM, USI, 10-bit ADC, WDC, dW, ISP via SPI

Tabulka 40-3. Přehled vybraných obvodů řady AVR ATmega

typ	IO	paměť			napájení [V]	takt [MHz]	pouzdro	periferie
		Flash	RAM	EEPROM				
48P	23	4K	512	256	2.7 - 5.5	0 - 20	PDIP28S	2*8bit Tim/C, 1*16bit Tim/C, 6*PWM, 6-8 10bit ADC, USART, SPI, TWI, WDT
88P		8K	1K	512				
168P		16K	1K	512				
328P		32K	2K	1K				
164P/PA	32	16K	1K	512B	1.8 - 5.5	0 - 20	PDIP40W, TQFP44, VQFN/QFN- 44, DRQFN- 44, VFBGA- 49	2* 8-bit counters, 16-bit counter, RTC, 6*PWM, 8*10-bit ADC, I ² C, 2*USART, SPI, WDT
324P/PA		32K	2K	1K				
644P/PA		64K	4K	2K				
1284(P)		128K	16K	4K				
8A	23	8KB	1KB	512B	2.7 - 5.5	0-16	PDIP28S, TQFP-32, QFN/MLF- 32	
16	32	16KB	1KB	512B	4.5 - 5.5 V	0 - 16	PDIP40W, TQFP-44, MLF-44	zastaralý
16L					2.7 - 5.5	0 - 8		
32A	32	32KB	2KB	1KB	2.7 - 5.5	0 - 16	PDIP-40, TQFP-44, QFN/MLF- 44	zastaralý
32					4.5 - 5.5	0 - 16		
32L					2.7 - 5.5	0 - 16		
64	53	64KB	4KB	2KB	4.5 - 5.5	0 - 16	TQFP-64, QFN/MLF- 64	zastaralý
64L					2.7 - 5.5	0 - 16		
128								
162V	35	16KB	1KB	512B	1.8 - 5.5	0 - 8	PDIP40W, TQFP-44, MLF-44	XMEM, 2*8bit Timer/counter, 2*16bit Timer/Counter
162					2.7 - 5.5	0 - 16		

Zjednodušený přehled ATmega:

- **ATmega8/16/32/64/128**
- **ATmega** — obvody s x GPIO. Pouzdro s x vývody (SOIC). Periferie:
- **ATmega** — obvody s x GPIO. Pouzdro s x vývody (SOIC). Periferie:
- **ATmega** — obvody s x GPIO. Pouzdro s x vývody (SOIC). Periferie:

Tabulka 40-4. Obvody s USB rozhraním

typ	Flash	RAM	EEPROM	GPIO	pouzdra
AT90USB82	8KB	512B	512B	22	TQFP32, VQFN32
AT90USB162	16KB	512B	512B	22	TQFP32, VQFN32
ATmega16U4	16KB	2.5KB	1KB	26	TQFP44, VQFN44
ATmega32U4	32KB	2.5KB	1KB	26	TQFP44, VQFN44
AT90USB646	64KB	4KB	2KB	48	TQFP64, VQFN64
AT90USB647	64KB	4KB	2KB	48	TQFP64, VQFN64
AT90USB1286	128KB	8KB	4KB	48	VQFN64
AT90USB1287	128KB	8KB	4KB	48	TQFP64, VQFN64

Překladače a assemblyery

- avra
- gcc-avr
- simulavr

Nástroje pro ladění programů

- AVaRICE³⁸

40.1. Integrované periferie a části procesoru

Procesory obsahují řadu integrovaných jednotek pro různé účely. Je třeba připomenout, že ne každý obvod má všechny periferie a i když mají periferie v obvodu stejné označení, jako například Timer/Counter0, mohou různé procesory obsahovat odlišné implementace časovače s odlišnou funkcí. Pokusím se zde popsat některé obvody i s ukázkami jejich použití a rovněž upozornit na to, v kterých procesorech se vyskytují.

Odkazy:

- Atmega8 Timers³⁹
-

40.1.1. Jádru procesoru

40.1.1.1. SREG — Status Register

*

Tabulka 40-5. SREG — The AVR Status Register

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Obrázek 40-1. SREG — The AVR Status Register

bit	7	6	5	4	3	2	1	0
	+	+	+	+	+	+	+	+
	I	T	H	S	V	N	Z	C
	+	+	+	+	+	+	+	+
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit 7 — I: Global Interrupt Enable

Global Interrupt Enable bit musí být nastaven aby byla povolena přerušení. Povolení jednotlivých přerušení se nastavuje v odpovídajících registrech periferního zařízení. Tento bit řídí/povoluje přerušení jako celek. Bit I je smazán

Bit 6 — T: Bit Copy Storage

Bit 5 – H: Half Carry Flag

40.1.2. Timer/Counter0 with PWM — čítač a časovač číslo 0 s PWM

Odkazy:

- AVR Timers - An Introduction⁴⁰
- Timers in Compare Mode - Part I⁴¹
- Timers in Compare Mode - Part II⁴²
-

TCNT0

OCR0A

OCR0B

TIFR0

TIMSK0

TCCR0A

Timer/Counter Control Register A

Tabulka 40-6. Timer/Counter Control Register A - TCCR0A

bit	7	6	5	4	3	2	1	0
	COM0A1	COM0A0	COM0B1	COM0B0	—	—	WGM01	WGM00
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W

TCCR0B

40.1.3. 8-bit Timer/Counter with Separate Prescaler and Compare Mode

ATtiny48/88

Vyskytuje se v obvodech:

- ATtiny48/88

40.1.4. PWM

Odkazy:

- AVR Learning - Pulse Width Modulation⁴³
-
-

40.1.5. Analogově digitální převodník

Odkazy:

- Analog to Digital Conversion in AVR⁴⁴
-

$$V_{in}[V] = (ADCH * 256 + ADCL) * V_{ref}[V] / 1024$$

$$V_{in}[V] = ADCH * V_{ref}[V] / 256$$

```
void init_adc(void) {

    /* Měříme na portu ADC0, připravíme si jen ten. */
    PORTC &= ~_BV(PC0); DDRC  &= ~_BV(PC0);
    /** Choose clock and start ADC */
    ADCSRA |= _BV(ADEN) | _BV(ADPS2) | _BV(ADPS1) | _BV(ADPS0) | _BV(ADSC);

    /* wait for the value */
    loop_until_bit_is_clear(ADCSRA, ADSC);
}

voltage_t measure_voltage() {
    voltage_t result;

    /* ADC0/PC0 přepneme na vstupní, pro jistotu. */
    PORTC &= ~_BV(PC0); DDRC  &= ~_BV(PC0);

    ADMUX = 0b11000000; /* 2.56V Rref, noADLAR, select ADC0 */
    ADCSRA |= _BV(ADEN) | _BV(ADSC); /* Start Conversion */
    loop_until_bit_is_clear(ADCSRA, ADSC);
    result = (voltage_t) ADCL;
    result |= (((voltage_t) ADCH) << 8);
    ADCSRA &= ~_BV(ADEN); /* Turn of ADC */

    return result;
}
```

40.2. Popis vybraných procesorů

•

ATtiny15

- The ATtiny15 Processor⁴⁵ on WEB-DOT-DEV

•

•

•

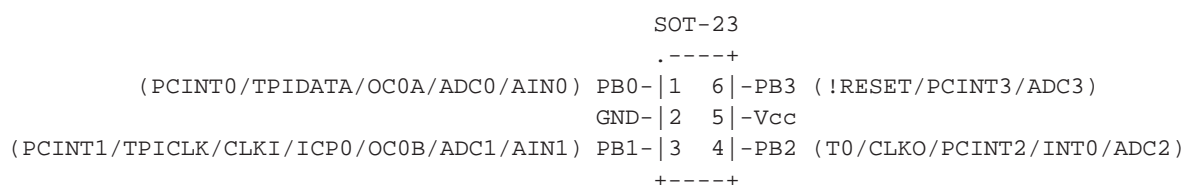
40.2.1. ATtiny4,5,9,10

Odkazy:

- Datasheet ATtiny4/5/9/10 Preliminary ⁴⁶

Skupina procesorů v 6-ti vývodovém pouzdře pro povrchovou montáž SOT-23. Procesory mají jen omezenou sadu 16-ti registrů R16 až R32. Mají 32B RAM a ukazatel zásobníku.

Obrázek 40-2. Zapojení vývodů pouzdra ATtiny4, ATtiny5, ATtiny9 a ATtiny10



40.2.2. ATtiny15

40.2.2.1. Generování přerušení čítačem 1

;

40.2.3. ATtiny48/88

To be done:

40.2.4. ATtiny 24,44,84

Tabulka 40-7. Dostupnost ATtiny 24,44,84

typ	obchod/prodejce	datum	cena	na skladě
ATtiny24-20PU	TME	2010-09-03	56.50 bez DPH	47
ATtiny24A-PU	TME	2010-09-03	50.20 bez DPH	17

40.2.5. ATtiny 25,45,85

Tabulka 40-8. Dostupnost ATtiny 25,45,85

typ	obchod/prodejce	datum	cena	na skladě
ATtiny25-20PU	TME	2010-09-03	40.10 bez DPH	267
ATtiny25V-10PU	TME	2010-09-03	50.20 bez DPH	120
ATtiny45-20PU	TME	2010-09-03	46.40 bez DPH	396

40.2.6. ATtiny261/461/861

* *Status: Preliminary*

Základní vlastnosti obvodu

- Podle typu 2, 4 nebo 8KB FLASH paměti programu, ISP. Životnost minimálně 10,000 Write/Erase cyklů.
- Podle typu 128, 256 nebo 512B EEPROM paměti. Životnost minimálně 100,000 Write/Erase cyklů.
- Podle typu 128, 256 nebo 512B SRAM paměti.
- Pracovní napětí: 2.7 až 5.5V
-

Peripheral Features:

- 8/16-bit Timer/Counter with Prescaler
- 8/10-bit High Speed Timer/Counter with Separate Prescaler
- Universal Serial Interface with Condition Detector
-

10-bit ADC

- 11 Single Ended Channels
- 16 Differential ADC Channel Pairs
- 15 Differential ADC Channel Pairs with Programmable Gain (1x, 8x, 20x, 32x)

Obrázek 40-3. Zapojení vývodů pouzdra ATtiny261/461/861

		+-----+-----+	
(MOSI/DI/SDA/!OC1A/PCINT8)	PB0	1 +--+ 20	PA0 (ADC0/DI/SDA/PCINT0)
(MISO/DO/OC1A/PCINT9)	PB1	2	19 PA1 (ADC1/DO/PCINT1)
(SCK/USCK/SCL/!OC1B/PCINT10)	PB2	3	18 PA2 (ADC2/INT1/USCK/SCL/PCINT2)
(OC1B/PCINT11)	PB3	4	17 PA3 (AREF/PCINT3)
VCC	5	16	AGND
GND	6	15	AVCC
(ADC7/!OC1D/CLKI/XTAL1/PCINT12)	PB4	7	14 PA4 (ADC3/ICP0/PCINT4)
(ADC8/OC1D/CLKO/CTAL2/PCINT13)	PB5	8	13 PA5 (ADC4/AIN2/PCINT5)
(ADC9/INT0/T0/PCINT14)	PB6	9	12 PA6 (ADC5/AIN0/PCINT6)
(ADC10/!RESET/PCINT15)	PB7	10	11 PA7 (ADC6/AIN1/PCINT7)
		+-----+-----+	

Popis vývodů

VCC

Napájecí napětí.

GND

Zem, 0V.

AVCC

Napájecí napětí analogové části.

AGND

Zem analogové části obvodu.

Port A (PA0..PA7)

8-mi bitový obousměrný I/O port s vnitřnímu pull-up odpory (můžou být aktivovány pro každý pin zvlášť).

Port B (PB0..PB7)

8-mi bitový obousměrný I/O port s vnitřnímu pull-up odpory (můžou být aktivovány pro každý pin zvlášť).

!RESET

Nízké napětí (logická 0) na tomto vstupu resetuje procesor. Nízké napětí musí trvat dostatečně dlouho, krátké pulzy nemusí provést reset obvodu.

Programovací hlava se zapojuje na vývody MOSI (PB0), MISO (PB1), SCK (PB2), !RESET (PB7) a GND

To be done:

40.2.7. ATtiny2313

Starší model, který se pořád vyrábí a vylepšuje. Původní varianty ATtiny2313 a ATtiny2313V byly nahrazeny novou variantou ATtiny2313A. Rovněž přišel nový model ATtiny4313 který má dvojnásobné velikosti pamětí.

Obrázek 40-4. ATtiny2313A, ATtiny4313 v pouzdře PDIP20/SOIC20

			+-----+ +-----+	
(PCINT10/!RESET/dW)	PA2-	1	+++	20 -Vcc
(PCINT11/RXD)	PD0-	2		19 -PB7 (USCK/SCL/SCK/PCINT7)
(PCINT12/TXD)	PD1-	3		18 -PB6 (MISO/DO/PCINT6)
(PCINT9/XTAL2)	PA1-	4		17 -PB5 (MOSI/DI/SDA/PCINT5)
(PCINT8/CLKI/XTAL1)	PA0-	5		16 -PB4 (OC1B/PCINT4)
(PCINT13/CKOUT/XCK/INT0)	PD2-	6		15 -PB3 (OC1A/PCINT3)
(PCINT14/INT1)	PD3-	7		14 -PB2 (OC0A/PCINT2)
(PCIBT15/T0)	PD4-	8		13 -PB1 (AIN1/PCINT1)
(PCINT16/OC0B/T1)	PD5-	9		12 -PB0 (AIN0/PCINT0)
GND-		10		11 -PD6 (ICPI/PCINT17)
			+-----+ +-----+	

40.2.8. ATtiny43U

*

Tento obvod se od všech ostatních obvodů liší přítomností boost konvertoru. S jeho použitím může obvod pracovat při napětí baterie 0.7V.

Tabulka 40-9. Dostupnost ATmega8

typ	obchod/prodejce	datum	cena	na skladě
ATtiny43U-SU	TME	2010-09-12	56.50 bez DPH	5

40.2.9. ATmega162 (XMEM)

Tento procesor je jedním z těch, které mají rozhraní XMEM pro připojení vnější paměti RAM. Procesor je dostupný taktéž v pouzdře PDIP. Dá se s ním tedy jednoduše experimentovat na nepájivém poli.

Řadič dále obsahuje 2 jednotky USART, rozhraní SPI, 4 čítače/časovače, 6 PWM, 8 AD kanálů, rozhraní XMEM.

Obvod nemá rozhraní I²C!

Tabulka 40-10. Vektory přerušení ATmega162

#	addr	source	Interrupt Definition
1	0x000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	0x002	INT0	
3	0x004	INT1	
4	0x006	INT2	
5	0x008	PCINT0	
6	0x00A	PCINT1	
26	0x032	EE_RDY	EEPROM Ready
27	0x034	ANA_COMP	Analog Comparator
28	0x036	SPM_RDY	Store Program Memory Ready

Tabulka 40-11. Registry

0x15	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0

0x18	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0

Obrázek 40-5. ATmega162 v pouzdře PDIP40

		+-----+	+-----+
(OC0/T0) PB0-	1		40 -Vcc
(OC2/T1) PB1-	2		39 -PA0 (AD0/PCINT0)
(RXD1/AIN0) PB2-	3		38 -PA1 (AD1/PCINT1)
(TXD1/AIN1) PB3-	4		37 -PA2 (AD2/PCINT2)
(!SS/OC3B) PB4-	5		36 -PA3 (AD3/PCINT3)
(MOSI) PB5-	6		35 -PA4 (AD4/PCINT4)
(MISO) PB6-	7		34 -PA5 (AD5/PCINT5)
(SCK) PB7-	8		33 -PA6 (AD6/PCINT6)
!RESET-	9		32 -PA7 (AD7/PCINT7)
(RXD0) PD0-	10		31 -PE0 (ICP1/INT2)
(TXD0) PD1-	11		30 -PE1 (ALE)
(INT0/XCK1) PD2-	12		29 -PE2 (OC1B)
(INT1/ICP3) PD3-	13		28 -PC7 (A15/TDI/PCINT15)
(TOSC1/XCK0/OC3A) PD4-	14		27 -PC6 (A14/TDO/PCINT14)
(OC1A/TOSC2) PD5-	15		26 -PC5 (A13/TMS/PCINT13)
(!WR) PD6-	16		25 -PC4 (A12/TCK/PCINT12)
(!RD) PD7-	17		24 -PC3 (A11/PCINT11)
XTAL2-	18		23 -PC2 (A10/PCINT10)
XTAL1-	19		22 -PC1 (A9/PCINT9)
GND-	20		21 -PC0 (A8/PCINT8)
		+-----+	

Tabulka 40-12. Dostupnost ATmega162

typ	obchod/prodejce	datum	cena	na skladě
ATmega162-16PU	TME	2010-09-03	169.00 bez DPH	0

40.2.9.1. Periferie

40.2.9.1.1. Timer/Counter 0

8-mi bitový čítač časovač.

Tabulka 40-13. Registry čítače/časovače 0 procesoru ATmega162

TCCR0	Timer/Counter Control Register
TCNT0	Timer/Counter Register
OCR0	Output Compare Register
TIMSK	Timer/Counter Interrupt Mask Register

TIFR				Timer/Counter Interrupt Flag Register				
bit	7	6	5	4	3	2	1	0
TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Tabulka 40-14.

CS02	CS01	CS00	popis
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk _{I/O} (No prescaling)
0	1	0	clk _{I/O} /8
0	1	1	clk _{I/O} /64
1	0	0	clk _{I/O} /256
1	0	1	clk _{I/O} /1024
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

40.2.9.1.2. Serial Peripheral Interface (SPI)

```

void SPI_MasterInit(void) {
    /* Set MOSI and SCK output, all others input */
    DDR_SPI = (1<<DD_MOSI) | (1<<DD_SCK);
    /* Enable SPI, Master, set clock rate fck/16 */
    SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0);
}

void SPI_MasterTransmit(char cData) {
    /* Start transmission */
    SPDR = cData;
    /* Wait for transmission complete */
    while (!(SPSR & (1<<SPIF))) ;
}

```


40.2.10. ATmega8

Odkazy:

- ATmega8A⁴⁷ na webu fy ATMEL
-
-

Co říká ATMEL o ATmega8A

8KB self-programming Flash Program Memory, 1KB SRAM, 512 Byte EEPROM, 6 or 8 Channel 10-bit A/D-converter. Up to 16 MIPS throughput at 16 Mhz. 2.7 - 5.5 Volt operation.

Fakta:

- velikost paměti programu Flash : 8KiB
- velikost paměti EEPROM: 512 B
- velikost pamětí RAM: 1024 Byte
- Maximální počet I/O vývodů: 23
- Maximální taktovací kmitočet: 16MHz
- Napájecí napětí v rozsahu 2.7 až 5.5 V
- Dodávaný v pouzdrech: MLF(VQFN)32, PDIP28, TQFP32
-

Obrázek 40-6. ATmega8 v pouzdře PDIP28

		+-----+ +-----+			
(RESET)	PC6-	1	+	28	-PC5 (ADC5/SCL)
(RXD)	PD0-	2		27	-PC4 (ADC4/SDA)
(TXT)	PD1-	3		26	-PC3 (ADC3)
(INT0)	PD2-	4		25	-PC2 (ADC2)
(INT1)	PD3-	5		24	-PC1 (ADC1)
(XCK/T0)	PD4-	6		23	-PC0 (ADC0)
	VCC-	7		22	-GND (?AGND)
	GND-	8		21	-AREF
(XTAL1/TOSC1)	PB6-	9		20	-AVCC
(XTAL2/TOSC2)	PB7-	10		19	-PB5 (SCK)
(T1)	PD5-	11		18	-PB4 (MISO)
(AIN0)	PD6-	12		17	-PB3 (MOSI/OC2)
(AIN1)	PD7-	13		16	-PB2 (SS/OC1B)
(ICP1)	PB0-	14		15	-PB1 (OC1A)

Mám na stole ATmega8-16PU, ale na webu ATMELu se vyskytuje již jend datasheet k ATmega8A. Ve vlastním archu jsem ještě našel datasheet pro ATmega8, ATmega8L. Oba datasheety se liší počtem stran. Starší pro ATmega8(L) má 308 stran a novější pro ATmega8A má 307 stran. Jediné rozdíly které jsem zatím objevil jsou v napájecím napětí a maximálním kmitočtu.

Tabulka 40-15.

MCU	Operating Voltages	Clock
ATmega8L	2.7 — 5.5 V	0 - 8 MHz
ATmega8	4.5 - 5.5 V	0 - 16 MHz
ATmega8A	2.7 - 5.5 V	0 - 16 MHz

Tabulka 40-16. Popis funkce vývodů portu B

číslo	název	alternativní funkce
PB7	XTAL2 (Chip Clock Oscillator pin 2), TOSC2 (Timer Oscillator pin 2)	
PB6	XTAL1 /Chip Clock Oscillator pin 1 or External clock input), TOSC1 (Timer Oscillator pin 1)	
PB5	SCK (SPI Bus Master clock Input)	
PB4	MISO (SPI Bus Master Input/Slave Output)	
PB3	MOSI (SPI Bus Master Output/Slave Input), OC2 (Timer/Counter2 Output Compare Match Output)	
PB2	SS (SPI Bus master Slave select), OC1B (Timer/Counter1 Output Compare Match B Output)	
PB1	OC1A (Timer/Counter1 Output Compare Match A Output)	
PB0	ICP1 (Timer/Counter1 Input Capture Pin)	

Tabulka 40-17. Dostupnost ATmega8

typ	obchod/prodejce	datum	cena	na skladě
ATmega8-16PU	TME	2010-09-03	125.00 bez DPH	6306
ATmega8A-PU	TME	2010-09-03	81.00 bez DPH	0
ATmega8L-8PU	TME	2010-09-03	99.90 bez DPH	2322

40.2.11. ATmega16/32/64

Tabulka 40-18. Dostupnost ATmega 16/32/64

typ	obchod/prodejce	datum	cena	na skladě
ATmega16A-PU	TME	2010-09-12	99.90 bez DPH	231
ATmega32A-16PU	TME	2010-09-12	112.00 bez DPH	2690
ATmega32A-PU	TME	2010-09-12	131.00 bez DPH	0
ATmega32L-8PU	TME	2010-09-12	175.00 bez DPH	443

40.2.12. ATmega 48,88,168,328

*

Tabulka 40-19. Některé parametry mikropočítačů ATmega48 až 328

MCU	Flash	EEP-ROM	SRAM
ATmega48PA	4 KB	256 B	512 B
ATmega88PA	8 KB	512 B	1 KB
ATmega168PA	16 KB	512 B	1 KB
ATmega328P	32 KB	1024 B	2 KB

Obrázek 40-7. ATmega48,88,168,328 v pouzdře PDIP28

		+-----+ +-----+		
(PCINT14/RESET)	PC6-	1	+--+	28 -PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD)	PD0-	2		27 -PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXT)	PD1-	3		26 -PC3 (ADC3/PCINT11)
(PCINT18/INT0)	PD2-	4		25 -PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1)	PD3-	5		24 -PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0)	PD4-	6		23 -PC0 (ADC0/PCINT8)
	Vcc-	7		22 -GND (?AGND)
	GND-	8		21 -AREF
(PCINT6/XTAL1/TOSC1)	PB6-	9		20 -AVcc
(PCINT7/XTAL2/TOSC2)	PB7-	10		19 -PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1)	PD5-	11		18 -PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0)	PD6-	12		17 -PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1)	PD7-	13		16 -PB2 (SS/OC1B/PCINT2)
(PCINT0/CLKO/ICP1)	PB0-	14		15 -PB1 (OC1A/PCINT1)
			+-----+	

Tabulka 40-20. I/O registry

adresa	číslo	název	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x1E		R30	General Purpose Register R30							
0x1F		R31	General Purpose Register R31							
0x20	0x00	Reserved								
0x21	0x01	Reserved								
0x22	0x02	Reserved								
0x23	0x03	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
0x24	0x04	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0

adresa	číslo	název	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x25	0x05	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
0x26	0x06	PINC		PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
0x27	0x07	DDRC		DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
0x28	0x08	PORTC		PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
0x29	0x09	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
0x2A	0x0A	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
0x2B	0x0B	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
0x2C	0x0C	<i>Reserved</i>								
0x2D	0x0D	<i>Reserved</i>								
0x2E	0x0E	<i>Reserved</i>								
0x2F	0x0F	<i>Reserved</i>								
0x30	0x10	<i>Reserved</i>								
0x31	0x11	<i>Reserved</i>								
0x32	0x12	<i>Reserved</i>								
0x33	0x13	<i>Reserved</i>								
0x34	0x14	<i>Reserved</i>								
0x35	0x15	TIFR0						OCF0B	OCF0A	TOV0
0x36	0x16	TIFR1			ICF1			OCF1B	OCF1A	TOV1
0x37	0x17	TIFR2						OCF2B	OCF2A	TOV2
0x38	0x18	<i>Reserved</i>								
0x39	0x19	<i>Reserved</i>								
0x3A	0x1A	<i>Reserved</i>								
0x3B	0x1B	PCIFR						PCIF2	PCIF1	PCIF0
0x3C	0x1C	EIFR							INTF1	INTF0
0x3D	0x1D	EIMSK							INT1	INT0
0x3E	0x1E	GPIOR0	General Purpose I/O Register 0							
0x3F	0x1F	EECR			EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE
0x40	0x20	EEDR	EEPROM Data Register							

* I/O Registry na adresách 0x20-0xFF jsou podle tabulky Register Summary z dokumentu ATmega48A/48PA/88A/88PA/168A/168PA/328/328P 8271C-AVR-08/10 strana 532-535

Tabulka 40-21. Dostupnost ATmega 48/88/169/328 P

typ	obchod/prodejce	datum	cena	na skladě
ATmega328P-PU	Farnell	2011-03-31	130.56, 83.63 (10ks)	1848
ATmega328P-AU	GME	2010-09-15	91.00 s DPH	0
ATmega168-20PU	TME	2010-09-03	99.90 bez DPH	504
ATmega168P-20PU	TME	2010-09-03	112.00 bez DPH	10
ATmega168PA-PU	TME	2010-09-03	99.90 bez DPH	0
ATmega328P-PU	TME	2010-09-03	188.00 bez DPH	0
ATmega88-20PU	TME	2010-09-03	106.00 bez DPH	0
ATmega88PA-PU	TME	2010-09-03	87.30 bez DPH	110
ATmega48-20PU	TME	2010-09-03	62.80 bez DPH	294

40.2.12.1. 8-bit Čítač/Casovač2 s PWM a asynchronním fungováním

* 8-bit Timer/Counter2 with PWM and Asynchronous Operation

* Tuto část by bylo vhodné přesunout někam mezi popis hardwarových jendotek protože by měla být společná více procesorům.

Vlastnosti Timer2:

- Jednakanálový čítač (pochopit, vysvětlit)
- Čítač může být načten při shodě porovnání (*Auto Reload*)
- *Glitch-free* PWM
- 10-bitový předřadný dělič (*Prescaler*)
- Přetečení (*Overflow*) a shoda porovnání (*Compare Match*) mohou vyvolávat přerušení TOV2, OCF2A a OCF2B
- Může používat externí 32kHz hodinkový krystal nezávislý na I/O hodinovém signálu

* Výše uvedené vlastnosti jsou jen přepisem z datasheetu.

Samotný čítač je 8-mi bitový a jeho aktuální hodnota je přístupná jako registr TCNT2. Tato hodnota je porovnávána s hodnotami v registrech OCR2A a OCR2B. Okamžik kdy je hodnota čítače stejná jako hodnota v porovnávacím registru generuje signál OC2A, OC2B který může být vyveden na výstupní pin mikrořadiče, použit k tvorbě PWM, nebo vyvolat přerušení.

Vstupním signálem který čítač čítá může být signál T2 z nožičky mikrořadiče, nebo výstup předřadné deličky. Správným nastavením zdroje děličky a dělicího poměru získáme vstupní signál o požadovaném kmitočtu.

Popis registrů

TCCR2A

Obrázek 40-8. TCCR2A — Timer/Counter Control register A

TCCR2A -- Timer/Counter Control Register A

(0xB0)	COM2A1 7	COM2A0 6	COM2B1 5	COM2B0 4	- 3	- 2	WGM21 1	WGM20 0
Read/Write	R/W 7	R/W 6	R/W 5	R/W 4	R 3	R 2	R/W 1	R/W 0
Initial Value	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0

- Bity 7:6 — COM2A1:0: Compare Match Output A Mode.

Tyto bity ovládají chování Output Compare pinu (OC2A).

Tabulka 40-22. Compare Output Mode, non-PWM Mode

COM2A1	COM2A0	pis
0	0	normální funkce portu, OC0A odpojeno
0	1	změn OC2A při shodě srovnání <i>Compare Match</i>
1	0	smaž (<i>Clear</i>) OC2A při shodě srovnání
1	1	nastave (<i>Set</i>) OC2A při shodě srovnání

Tabulka 40-23. Compare Output Mode, Fast PWM Mode

COM2A1	COM2A0	pis
0	0	normální funkce portu, OC0A odpojeno
0	1	WGM22=0: normální funkce portu, OC0A odpojeno. WGM22=1: změn (<i>Toggle</i>) OC2A při shodě srovnání <i>Compare Match</i>
1	0	smaž (<i>Clear</i>) OC2A při shodě srovnání
1	1	nastave (<i>Set</i>) OC2A při shodě srovnání

Tabulka 40-24. Compare Output Mode, Phase Correct PWM Mode

COM2A1	COM2A0	pis
0	0	normální funkce portu, OC2A odpojeno
0	1	WGM22=0: normální funkce portu, OC2A odpojeno. WGM22=1: změn (<i>Toggle</i>) OC2A při shodě srovnání <i>Compare Match</i>
1	0	smaž (<i>Clear</i>) OC2A při shodě srovnání
1	1	nastave (<i>Set</i>) OC2A při shodě srovnání

- Bity 5:4 — COM2B1:0: Compare Match Output B Mode

Stejně možnosti jako při nastavování COM2A1:0.

Následující tabulka popisuje jednotlivé mody čítače podle nastavení bitů WGM1 a WGM0 v registru TCCR2A a bitu WGM2 v registru TCCR2B.

Tabulka 40-25. Waveform Generation Mode Bit Description

mód	WGM2	WGM1	WGM0	Operace	TOP	update of OCRx at	TOV Flag Set on
0	0	0	0	Normal	0xFF	ihned	0xFF
1	0	0	1	PWM, Phase Correct	0xFF	TOP	0x00
2	0	1	0	CTC	OCRA	ihned	0xFF

Následují podrobnější popisy jednotlivých módů čítače.

Normal Mode

Nejjednodušší mód fungování (WGM22:0 = 0). V tomto módu se čítá jen nahoru (up, incrementing), a čítač není nikdy nulován. Při překročení maximální hodnoty 0xFF přirozeně přejde na 0x00.

Clear Timer on Compare Match (CTC) Mode

V režimu vymazání čítače při shodě s OCR2A registrem (WGM22:0 = 2), určuje registr OCR2A rozlišení čítače. V tomto módu je nastaven čítač na 0, dosáhne li TCNT2 hodnoty v registru OCR2A. OCR2A tedy definuje horní hodnotu čítače. Hodnota čítače TCNT2 se tedy pohybuje v rozsahu od 0 do OCR2A v čteně.

V tomto módu může být povoleno a generováno přerušení TOV1, ...

TCCR2B

Druhý ze dvojce registrů ovládajících chování čítače/časovače Timer2.

Obrázek 40-9. TCCR2B — Timer/Counter Control register B

TCCR2B -- Timer/Counter Control Register B

(0xB1)	FOC2A 7	FOC2B 6	- 5	- 4	WGM22 3	CS22 2	CS21 1	CS20 0
Read/Write	W 7	W 6	R 5	R 4	R 3	R/W 2	R/W 1	R/W 0
Initial Value	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0

TCNT2

TCNT2

OCR2A

OCR2B

TIMSK2

TIFR2

ASSR

Obrázek 40-10. ASSR — Asynchronous Status Register

ASSR - Asynchronous Status Register

(0xB6)	- 7	EXCLK 6	AS2 5	TCN2UB 4	OCR2AUB 3	OCR2BUB 2	TCR2AUB 1	TCR2BUB 0
Read/Write	R 7	R/W 6	R 5	R 4	R 3	R 2	R 1	R 0
Initial Value	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0

Bit 7 — rezervován pro budoucí použití

Tento bit má při čtení vždy hodnotu 0.

Bit 6 — EXCLK: Enable External Clock Input

Povolení externího hodinového signálu

Je-li tento bit nastaven na 1, je vybrán asynchronní externí hodinový signál. Tento signál musí být přiveden na pin TOSC1 obvodu. Je-li tento bit nastaven na 0, pak je nastartován vnitřní krystalový oscilátor 32kHz a tento je použit jako hodinový signál.

Varování

Vnitřní krystalový oscilátor 32kHz běží jen, je-li tento bit 0. V obvodech ATmega328P je tento oscilátor velmi nepřesný. Pokud chcete více, nahlédněte do poslední revize datasheetu.

Bit 5 — AS2: Asynchronous Timer/Counter2

Asynchronní čítač/časovač 2

Je-li tento bit 0, Čítač čítá z vnitřního signálu clk_{IO} . Je-li tento bit 1, čítač čítá z krystalového oscilátoru připojeného na vývod TOSC1. Při změně hodnoty bitu AS2 může být poškozen obsah registrů TCNT2, OCR2A, OCR2B, TCCR2A a TCCR2B.

AS2 bit vybírá, jestli je do desetibitové předděličky jako signál clk_{T2S} pouštěn clk_{IO} (AS2=0) nebo TOSC1 (AS2=1).

Bit 4 — TCN2UB: Timer/Counter2 Update Busy

Bit 3 — OCR2AUB: Output Compare Register2 Update Busy

Bit 2 — OCR2BUB: Output Compare Register2 Update Busy

Bit 1 — TCR2AUB: Timer/Counter Control Register 2 Update Busy

Bit 0 — TCR2BUB: Timer/Counter Control Register 2 Update Busy

GTCCR — General Timer/Counter Control Register

40.2.13. ATmega 164/324/644/1284 P/PA

Odkazy:

- Datasheet 8272A-AVR-01-10⁴⁸ z ledna 2010
-
-
-

Tato skupina procesorů má velkou výbavu periferiemi a je k dispozici v 40 vývodovém pouzdře PDIP a taktéž v 44 vývodových pouzdrech pro povrchovou montáž.

Fakta:

- Pouzdra: PDIP40, TQFP-44, VQFN/QFN/MLF-44, DRQFN-44, VFBGA-49
- Pracovní napětí od 1,8 do 5,5 V
- Maximální kmitočet se liší podle velikosti napájecího napětí.

0 - 4	1.8 - 5.5
0 - 10	2.7 - 5.5
0 - 20	4.5 - 5.5

-
-
-
-
-
-

Obrázek 40-11. ATmega 164/324/644/1284 P/PA v pouzdře PDIP40W

+-----+ +-----+
 (PCINT8/XCK0/T0) PB0- | 1 +---+ 40 | -PA0 (ADC0/PCINT0)

(PCINT9/CLKO/T1) PB1-	2	39	-PA1 (ADC1/PCINT1)
(PCINT10/INT2/AIN0) PB2-	3	38	-PA2 (ADC2/PCINT2)
(PCINT11/OC0A/AIN1) PB3-	4	37	-PA3 (ADC3/PCINT3)
(PCINT12/OC0B/!SS) PB4-	5	36	-PA4 (ADC4/PCINT4)
(PCINT13/MOSI) PB5-	6	35	-PA5 (ADC5/PCINT5)
(PCINT14/MISO) PB6-	7	34	-PA6 (ADC6/PCINT6)
(PCINT15/SCK) PB7-	8	33	-PA7 (ADC7/PCINT7)
!RESET-	9	32	-AREF
Vcc-	10	31	-GND
GND-	11	30	-AVcc
XTAL2-	12	29	-PC7 (TOSC2/PCINT23)
XTAL1-	13	28	-PC6 (TOSC1/PCINT22)
(PCINT24/RXD0) PD0-	14	27	-PC5 (TDI/PCINT21)
(PCINT25/TXD0) PD1-	15	26	-PC4 (TDO/PCINT20)
(PCINT26/RXD1/INT0) PD2-	16	25	-PC3 (TMS/PCINT19)
(PCINT27/TXD1/INT1) PD3-	17	24	-PC2 (TCK/PCINT18)
(PCINT28/XCK1/OC1B) PD4-	18	23	-PC1 (SDA/PCINT17)
(PCINT29/OC1A) PD5-	19	22	-PC0 (SCL/PCINT16)
(PCINT30/OC2B/ICP) PD6-	20	21	-PD7 (OC2A/PCINT31)
+-----+			

Tabulka 40-26. Dostupnost ATmega 164/324/644/1284 P/PA

typ	obchod/prodejce	datum	cena
ATmega1284P-PU	Farnell	2010-11-15	225.17 bez DPH
ATmega644P-20PU	TME	2010-09-03	188.00 bez DPH
ATmega644PA-20PU	TME	2010-09-03	194.00 bez DPH
ATmega644P-20PU	TME	2010-02-10	154.09 bez DPH
ATmega644P-20AU	TME	2010-02-04	179.88 bez DPH
ATmega644-20AU	TME	2010-02-04	160.54 bez DPH

40.3. Programování AVR

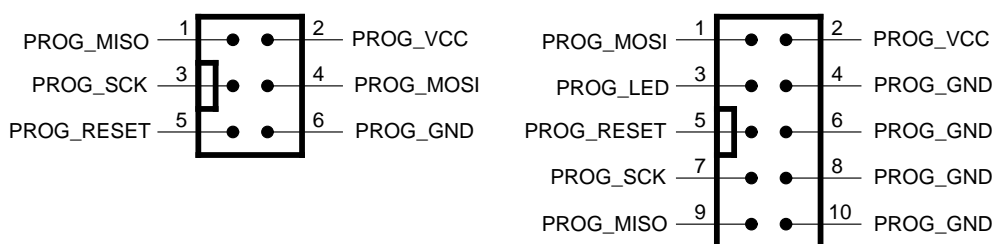
Aneb jak tam dostat program

* *Attributy: id="avr.programming"*

Odkazy:

- ISP Header - 6-way IDC - Atmel SPI pin-out⁴⁹
- ISP Header - 10-way IDC - Atmel SPI pin-out⁵⁰

Když už máme program pro AVR napsán a přeložen, potřebujeme ho do toho malého brouka nahrát. A tady stojíme před otázkou jak. Buď to si opatříme hotový programátor pro obvody AVR, nebo si jej postavíme. Konstrukce je celá řada. Já začnu těmi nejjednoduššími. Ale nejdříve si ukážeme jak vypadá ISP (*In System Programming*) konektor.

Obrázek 40-12. Programovací konektory pro procesory AVR

Bežně se objevují tři druhy konektorů, lišící se provedním a počtem pinů. První dva jsou IDC konektory (hřebínky) s dvěma řadami nožiček. Jeden má celkem 6 vývodů a druhý 10. Třetím konektorem je hřebínek s šesti nožičkami v jedné řadě.

Programy:

- AVRDUDE⁵¹
-

40.3.1. Jednoduchý programátor na paralelní port

Odkazy:

- Atmel AVR ATmega16 / ATmega32 Programmer - Cheap Parallel Port Programmer⁵²
- Parallel Port Programmer⁵³
- Jednočipy pod Linuxem - I⁵⁴
- ATmega16/32 Programmer⁵⁵ — od Kartik Mohta
- AVR - ATmega8 #1⁵⁶
- First contact with ATmega8 microcontroller - part 2⁵⁷
- AVR 4: Arduino - nahrani bootloaderu⁵⁸
-
-

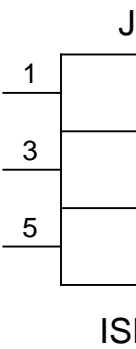
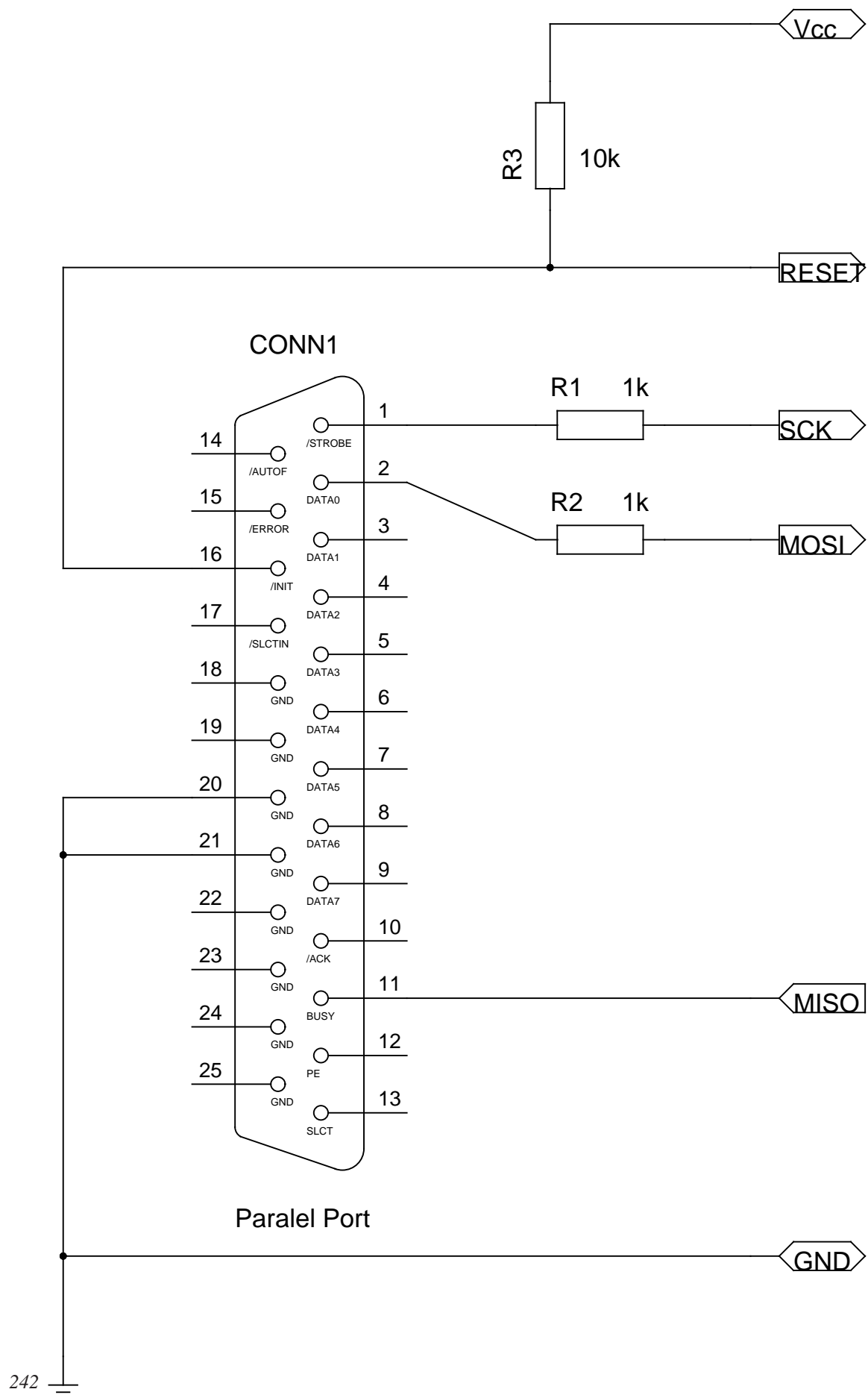
Tento programátor patří mezi nejjednodušší které jsem viděl. Stačí nam jen konektor na paralelní port, několik odporů a drátky. Je tak jednoduchý, že jsem se rozhodl začít s ním. Ano, paralelní porty, sériové porty a další standardní rozhraní postupně z notebooků mizí a protože můj hlavní počítač je již mnoho let notebook, neodzkouším paralelní programátor na něm. Ale nějaký starší počítač s paralelním portem doma mám, takže hurá na něj.

Popis zapojení

- pin 1 (Strobe) paralelního konektoru připojíme přes odpor 1kΩ na pin SCK procesoru
- pin 2 (D0) paralelního konektoru připojíme přes odpor 1kΩ na pin MOSI procesoru
- pin 11 (Busy) paralelního konektoru připojíme přes odpor nebo bez něj na pin MISO procesoru
- pin 16 (Init) paralelního konektoru připojíme na vývod RESET procesoru
- pin 18 (GND) paralelního portu připojíme na GND procesoru.
- piny 20 a 21 spojíme a připojíme na zem procesoru (GND).
-

Hodnoty odporů se v zapojeních která jsem našel liší. Sahají od zhruba 1kΩ až přímému propojení bez odporů.

Obrázek 40-13. ISP programátor pro AVR na paralelní port



40.3.2. Programátor na paralelní port s IO

Odkazy:

- LancOS PC paralel port programmer⁵⁹
- AVR - ATmega8 #1⁶⁰
-

40.3.3. Jednoduchý programátor na sériový port

Odkazy:

- 3 Socket DASA/Serial AVR Programmer In A Box!⁶¹
- Part III - Making Programmer and Target.⁶²
- DASA programátor OLIMEX⁶³
- AVR In-Circuit Serial Programmer⁶⁴ konstrukce která mne zaujala svou miniaturizací bez použití SMD součástek.

Po hledání na internetu, jsem se rozhodl založit svou konstrukci jednoduchého DASA programátoru pro AVR MCU na schématu uvedeném v AVR In-Circuit Serial Programmer⁶⁵ (původně OLIMEX⁶⁶). Rovněž mne přímo uchvátilo zabudování tohoto programátoru přímo do sériového konektoru. Rozhodl jsem se ale použít trochu odlišnou montáž. Plošný PCB chci instalovat nikoliv kolmo ale na ležato a zasunout jej mezi vývody DB9 konektoru. Spodní stranu PCB s měděnými kontakty připájet k pěti vývodům 1 až 5 konektoru. Deska bude mít nepravidelný tvar. Zatím nevím, jestli se mi takovou konstrukci podaří realizovat.

40.3.4. Programátory na USB port

Odkazy:

•

Software USB:

- USBasp - USB programmer for Atmel AVR controllers⁶⁷
- Building and installing USBASP-USB programmer of AVR microcontrollers⁶⁸
- One click project with USBasp programmer⁶⁹
-
-
-

•

S USB-Serial převodníkem, nebo hardwarovým řešením USB:

- AvrUsb500v2 -- an open source Atmel AVR Programmer, stk500 V2 compatible, with USB interface⁷⁰
- ISPProgrammingAnAVRopendous⁷¹
-

Velmi zvláštní programátor HUB ISP - Solving the USB-Only "Chicken or Egg" Problem⁷², je sestaven jen z čtyřportového USB HUBu, obvodu 74HC00, čtyř odporů dvou kondenzátorů a dvou LED diod.

Z českých konstrukcí bych uvedl UsbProg⁷³.

Diskuse⁷⁴ na ABC Linuxu o programátorech AVR.

AVR Programátor USB⁷⁵

40.3.5. avrdude

Instalace nové verze ze zdrojů. Zdrojové kódy nahrajeme do adresáře `/usr/local/download`.

```
# cd /usr/local/src
# tar xzvf ../download/avrdude-5.10.tar.gz
# cd avrdude-5.10
```

V rozbalených zdrojích si můžeme přečíst README, NEWS a INSTALL. Pak přistoupíme k překladu.

```
# ./configure
# make
# make install
make[1]: Entering directory `/usr/local/src/avrdude-5.10'
make[2]: Entering directory `/usr/local/src/avrdude-5.10'
test -z "/usr/local/bin" || /bin/mkdir -p "/usr/local/bin"
  /usr/bin/install -c 'avrdude' '/usr/local/bin/avrdude'
Backing up avrdude.conf in /usr/local/etc
test -z "/usr/local/etc" || /bin/mkdir -p "/usr/local/etc"
  /usr/bin/install -c -m 644 'avrdude.conf' '/usr/local/etc/avrdude.conf'
test -z "/usr/local/share/man/man1" || /bin/mkdir -p "/usr/local/share/man/man1"
  /usr/bin/install -c -m 644 './avrdude.1' '/usr/local/share/man/man1/avrdude.1'
make[2]: Leaving directory `/usr/local/src/avrdude-5.10'
make[1]: Leaving directory `/usr/local/src/avrdude-5.10'
```

40.4. Nástroje a prostředí pro vývoj programů

Odkazy:

- AVR a Linux⁷⁶
- AvrMon - an interactive debugging aid⁷⁷
- The AVR Assembler Site⁷⁸
- Home of Toms Linux AVR Assembler⁷⁹ — poslední verze 1.22 je z ledna 2005
- Beginners Programming in AVR Assembler⁸⁰

Tutoriály o programování:

- A Brief Tutorial on Programming the AVR without Arduino⁸¹
- AVR 8-bit Microcontrollers⁸²
- A simple project⁸³

K dispozici je řada nástrojů. Já zde zmíním jen ty se kterými mám zkušenosti, nebo o kterých vím. V první řadě jsou to assemblyry. Považuji znalost assembleru za docela základní. Chci mít přehled o tom, co se vlastně na úrovni strojového kódu v procesoru děje a assembler proto považuji za jeden ze základních prostředků.

40.4.1. Assembler

40.4.1.1. avra

Odkazy:

- avra⁸⁴ on SourceForge
- AVRA⁸⁵ — advanced macro assembler for Atmel AVR family

40.4.1.1.1. Překlad a instalace avra 1.2.3a

*

avra je kvalitní assembler kompatibilní s AVRASM1. Je již trochu starší a chybí v něm definice novějších procesorů. Ale i to je řešitelné. Pokud nepoužijeme direktivu `.device` musíme se sami ohlídat v používaných prostředích, paměti, instrukcích. Dalším problémem jsou inc soubory s definicemi pro jednotlivé procesory. Tyto jsou v novějších verzích vývojového prostředí pro AVR procesory pouze pro AVRASM2. V první chvíli jsem přišel s jednoduchým řešením které prověří čas a složitější programy. Inc soubory jsem zbavil všech # direktiv. Můžeme je například protlačit filtrem podle následující ukázky.

```
$ grep -v ^# orig/tn461def.inc >tn461def.inc
```

Jak jsem zmínil, avra ve verzi 1.2.3.1 nezná v direktivě `.device` novější procesory, jako je například výše uvedený ATtiny461. Můžeme si pomoci několika způsoby. Můžeme odstranit direktivy `.device` pro procesory s nimiž má avra problém. Můžeme požit podobné procesory. Například pro zmíněný ATtiny461 by připadal v úvahu ATtiny45. Druhou možností je použít novější verzi avra. Já jsem pro vlastní potřebu zasáhl do zdrojových kódů a v dobré víře dle svých nejlepších znalostí provedl úpravy. Tyto jsou k dispozici ke stažení například na Download⁸⁶

* *FIXME: Doplnit url a dát do něj zdroje/patch.*

Jak jsem říkal, čas ukáže, jak mnoho budu lámat avru než zkusím jiný nástroj.

Postup překladu avra ze zdrojů. Stáhneme si zdroje a umístíme do adresáře `/usr/local/download` nebo jiného. Zdroje rozbalíme.

```
# cd /usr/local/src
# tar xjvf /usr/local/download/avra-1.2.3a-src.tar.bz2
```

Na mém systému, kde jsem překládal řadu programů mám všechny potřebné nástroje nainstalovány. Takže v tuhle chvíli nevím které všechny to jsou. Určite to budou programy/balíčky jako `autoconf`, `automake` samozřejmě `gcc`

```
# cd avra-1.2.3
# ./automake-compile
```

Tot' vše. Tedy překladač máme a podáváme li se blíže do výpisu uvidíme že se nainstaloval jako `/usr/local/bin/avra`.

40.4.1.1.2. Překlad a instalace avra 1.3.0

Stáhneme si z sourceforge verzi 1.3.0. Tu rozbalíme do adresáře. Projdeme rozbalené soubory a přepneme se do adresáře `src`. Nakopírujeme si správný `makefile` a přeložíme.

```
$ cp makefiles/Makefile.linux Makefile
$ make
$ cp avra ~/bin/
```

40.4.1.2. gavrasm

Odkazy:

- gavrasm⁸⁷
-
-

40.4.2. Ada

Odkazy:

- AVR-Ada⁸⁸ na SOURCEFORGE.NET
- AdaCore⁸⁹ The GNAT Pro Company

40.4.2.1. Poznámky k překladu překladače

Pro první pokus jsem použil build script podle Building and Installing an Ada Cross Compiler using the Build Script⁹⁰. Tento skript se například nachází v balíčku avr-ada-1.0.3.tar.bz2 na SourceForge.NET⁹¹. Balíček je datovaný 2009-02-15 15:00. Skript se nachází v adresáři tools/builds/build-avr-ada-gcc-4.3.x.sh. Ten jsem si zkopírovan do adresáře ~/work/ada-avr který jsem pro účel překladu vytvořil. V souboru jsem pak provedl pár úprav viz následující diff.

```
$ diff -u build-avr-ada-gcc-4.3.x.sh.orig build-avr-ada-gcc-4.3.x.sh
--- build-avr-ada-gcc-4.3.x.sh.orig      2009-05-20 12:27:37.857108587 +0200
+++ build-avr-ada-gcc-4.3.x.sh          2009-05-28 14:13:42.269296012 +0200
@@ -67,7 +67,7 @@

OS=`uname -s`
if test $OS = "Linux" ; then
-   PREFIX="/opt/avr"
+   PREFIX="${HOME}/lib/avr"
else
    PREFIX="/mingw/avr"
fi
@@ -82,7 +82,7 @@
VER_MPFR=2.4.0
VER_GMP=4.2.4
VER_LIBC=1.6.2
-VER_AVRADA=1.0.2
+VER_AVRADA=1.0.3

FILE_BINUTILS="binutils-$VER_BINUTILS"
FILE_GCC="gcc-$VER_GCC"
```

Při haváriích překladu jsem podle chyby v deníku odhadl balíček který mi v systému chybí a doinstaloval. Bohužel jsem si nepoznamenal které balíčky to byly. Z deníku /var/log/aptitude vyjímám jména balíčků které jsem ten den instaloval a měly by s překladem souviset.

Přiinstalované balíčky

- gnat-4.3
- binutils-source
- flex
- autoconf

- bison
- libmpfr-dev

Poznámka: Instaloval jsem na Debian 5.0 Lenny.

40.4.3. C

Odkazy:

- C Programming and the ATmega16 Microcontroller.⁹²
-
-
-

Sériové rozhraní:

- <http://www.captain.at/electronic-atmega16-serial-port.php>
-
-

Odkazy:

- AVR Atmega Microcontroller Tutorial 4 Blink LED(Tishitu Electronics)⁹³ na YouTube
-

40.4.3.1. gcc-avr

V Debian 5.0 Lenny, stejně jako v předchozí verzi je balíček `gcc-avr`. Balíček starší verze Debian Etch jen nepozná některé novější MCU (například ATtiny25).

```
# aptitude install gcc-avr avr-libc
```

K dispozici máme rovněž balíčky:

- `gdb-avr` — s ladicím programem
- `simulavr` — simulátor

Odkazy:

- A simple project⁹⁴
- Programming the AVR Microcontroller with GCC⁹⁵
- Programming the AVR microcontroller with GCC, libc 1.0.4⁹⁶
- Tuxgraphics AVR C-programming tutorial⁹⁷
- Electrons :: Articles :: AVR :: AVR-GCC Programming Guide⁹⁸ — Learn how to code for these nifty microcontrollers using the excellent AVR-GCC.
- Atmega series of AVR microcontrollers⁹⁹
-
-

40.4.3.2. Tutoriál

* Krátké čtení o tom jak obsloužit různé části procesoru a různé periferie v jazyce C či assembleru a další potřebné znalosti.

* <http://www.joshknows.com/avr>

Tabulka 40-27. Přesné bitové šířky jednotlivých typů

Width	Signednes	Typedef	Const Cast
8	signed	int8_t	INT8_C
8	unsigned	uint8_t	UINT8_C
16	signed	int16_t	INT16_C
16	unsigned	uint16_t	UINT16_C
32	signed	int32_t	INT32_C
32	unsigned	uint32_t	UINT32_C
64	signed	int64_t	INT64_C
64	unsigned	uint64_t	UINT64_C

* http://www.nongnu.org/avr-libc/user-manual/group__avr__fuse.html

```
#include <avr/io.h>

FUSES =
{
    .low = LFUSE_DEFAULT,
    .high = (FUSE_BOOTSZ0 & FUSE_BOOTSZ1 & FUSE_EESAVE & FUSE_SPIEN & FUSE_JTAGEN),
    .extended = EFUSE_DEFAULT,
};
```

40.4.3.2.1. Vstupně výstupní a jiné porty

```
/* Data Direction Register for port C */
DDRC |= _BV(PC5) | _BV(PC4) | _BV(PC3);

PORTC = 0;

PORTC |= _BV(PC3); /* Set pin PC3 to 1 */
PORTC &= _BV(PC4); /* Set pin PC4 to 0 */
```

40.4.3.2.2. Obsluha přerušení

Odkazy:

- Introduction to avr-libc's interrupt handling¹⁰⁰
-

Funkce pro obsluhu přerušení se deklaruje makrem ISR. Parametrem tohoto makra je vektor přerušení které chceme obsluhovat.

```
#include <avr/interrupt.h>

ISR(ADC_vect) {
    // kód obsluhy přerušení
```

}

Tabulka 40-28. Vektory přerušení

název vektoru	popis
ADC_vect	Byla dokončena digitalizace analogové hodnoty.
ANALOG_COMP_0_vect	
ANALOG_COMP_1_vect	
ANALOG_COMP_1_vect	
ANALOG_COMP_vect	
ANA_COMP_vect	
CANIT_vect	
EEPROM_READY_vect	
EE_RDY_vect	

40.4.4. Forth

Odkazy:

- Mailová skupina avrForth¹⁰¹ na Yahoo, několik let bez aktivity

Různé implementace Forthu a jazyků z Forthu vycházejících.

40.4.4.1. TinyBoot

Odkazy:

- Tiny Open Firmware¹⁰² — doména tinyboot.com již neexistuje

40.4.4.2. amforth

Odkazy:

- amforth¹⁰³ na SourceForge
- Amforth - ATmega FORTH¹⁰⁴ na μ electronics info [2008-12-06]
- amforth¹⁰⁵

```
amforth 4.1 ATmega328P Forthduino
> 4 5 6 swap .s
0 2217 5
1 2219 6
2 2221 4
> : hello-kitty ." Hello kitty." cr ;
ok
> hello-kitty
Hello kitty.
ok
```

40.4.4.3. avrforth

Odkazy:

- domovská stránka avrforth¹⁰⁶
-
-

Tato implementace forthu je vlastně takový bezbarvý colorForth¹⁰⁷. Přebírá z něj řadu prvků a implementuje v nebarevném prostředí alfanumerického terminálu.

Projekt je dále zajímavý tím, že celý avrforth je napsán ve forthu a překlad, tedy vytvoření binárního obrazu pro daný procesor se provádí s pomocí gforth¹⁰⁸.

40.4.4.4. PFAVR

Odkazy:

- PFAVR -- An ANS Forth Implementation for the Atmel AVR¹⁰⁹
-
-

PFAVR je 16-ti bitová implementace Forthu pro procesory/mikrořadiče ATMEL AVR. Potřebuje 13K slov FLASH a 32K bajtů RAM.

40.4.5. Occam- π

Odkazy:

- concurrency.cc¹¹⁰ Parallel programming for tiny computers
- Fascinating concurrency¹¹¹ na Jee Labs

40.4.6. side4linux

- side4linux¹¹²

40.4.7. Grafická IDE

Odkazy:

- KontrollerLab IDE development software for AVR under Linux¹¹³
- avrLab¹¹⁴ [2003-01-02]
-
-
-

40.4.8. Simulátory a analyzátory

Odkazy:

- [simavr](#)¹¹⁵
- [Avrora](#)¹¹⁶ The AVR Simulation and Analysis Framework
-

40.5. Instrukční sada

Přehled instrukcí a popis vybraných instrukcí. Tento přehled je velmi neúplný a obsahuje jen instrukce které jsem z nějakého důvodu potřeboval blíže popsat. Pokud chcete úplný popis všech instrukcí, najdete jej na webu Atmelu.

Tento přehled instrukcí není rozhodně kompletní ani se jím nesnaží být. Popisuji zde opravdu jen některé instrukce.

Při úvahách jak mám instrukce popsat jsem zůstal u toho, že první popis instrukce je v angličtině. Je tak přímo vidět proč se instrukce jmenuje jak se jmenuje, protože její název je skratka z anglické věty která ji popisuje. Dále pak popisuji instrukci česky. Bohuže v DocBook který používám, nelze k jedné instrukci mít dva popisky. Odpadá tak možnost použít současně původní anglický popis a hned vedle něj český popis.

* `colname="name" colwidth="0.8in", colname="mnemo" colwidth="0.8in", colname="bin" colwidth="1.7in", colname="hex" colwidth="0.4in", colname="desc" colwidth="*"`

Tabulka 40-29. Operační kódy instrukcí AVR řazené podle operačního kódu

instrukce	mnemo	strojový kód		popis
		bin	hex	
and	AND Rd,Rr	0010 00rd dddd rrrr	2000	Logical AND
sbr	SBR Rd,K	0110 KKKK dddd KKKK	6000	Set Bits in Register
andi	ANDI Rd,K	0111 KKKK dddd KKKK	7000	Logical AND with Immediate
lds	LDS Rd,k	1001 000d dddd 0000 kkkk kkkk kkkk kkkk	90000000	Load Direct from Data Space
lpm	LPM Rd,Z	1001 000d dddd 0100	9004	Load Program Memory
lpm	LPM Rd,Z+	1001 000d dddd 0101	9005	Load Program Memory
break	BREAK	1001 0101 1001 1000	9598	Break
lpm	LPM	1001 0101 1100 1000	95c8	Load Program Memory
adiw	ADIW Rd+1:Rd,K	1001 0110 KKdd KKKK	9600	Add Immediate to Word
cbi	CBI A,b	1001 1000 AAAA Abbb	9800	Clear Bit in I/O Register

instrukce	mnemo	strojový kód		popis
		bin	hex	
sbic	SBIC A,b	1001 1001 AAAA Abbb	9900	Skip if Bit in I/O Register Cleared
sbi	SBI A,b	1001 1010 AAAA Abbb	9a00	Set Bit in I/O Register
sbis	SBIS A,b	1001 1011 AAAA Abbb	9b00	Skip if Bit in I/O Register is Set
lds	LDS Rd,k	1010 0kkk dddd kkkk	a000	Load Direct from Data Space
bst	BST Rd,b	1111 101d dddd 0bbb	fa00	Bit Store from Bit in Register to T Flag in SREG
sbrc	SBRC Rr,b	1111 110r rrrr 0bbb	fc00	Skip if Bit in Register is Cleared
sbrs	SBRs Rr,b	1111 111r rrrr 0bbb	fe00	Skip if Bit in Register is Set

Instrukční slovo má několik různých tvarů a sestává z řady polí. Tato se opakují na různých instrukcích. Proto je popisují zde na společném místě abych je zbytečně nevypisoval u každé instrukce.

Parametry instrukcí

port

Číslo (adresa) I/O registru. Protože *port* může nabývat hodnot od 0 do 31, můžeme jím adresovat jen prvních 32 portů jenž se nacházejí v RAM na adresách 0x20 až 0x3F (32 až 63 dekadicky).

bit

číslo bitu v I/O registru

Rd

Registr. Jeden z registrů R16 až R31. Parametrem *Rd* adresujeme libovolný horních 16-ti registrů procesoru.

Rr

Registr. Jeden z registrů R0 až R31. Parametrem *Rr* adresujeme libovolný z registrů procesoru.

Rs

Registr. Jeden z registrů R0 až R31. Parametrem *Rd* adresujeme libovolný z registrů procesoru.

K

Osmibitová konstanta která je přímo součástí instrukce.

ADIW

Jméno

ADIW — Add Immediate to Word

Přehled

ADIW Rd+1:Rd,K

Parametr *Rd* a *K*.

Popis

Instrukce přičte šestibitovou konstantu K ($0 \leq K \leq 63$) ke dvojici registrů (25:24, 27:26, 29:28, 31:30). Dvojice registrů je chápána jako jeden 16-ti bitový registr.

Obrázek 40-1. Formát instrukce CBI

0	1001 1000	7 8	AAAAA	12 13	bbb	15
---	-----------	-----	-------	-------	-----	----

Odkazy

Související instrukce: **FIXME**:avr.isa.sbi;.

- *8-bit AVR Instruction Set Rev. 0856I-AVR-07/10*, strana 18

AND

Jméno

AND — Logical AND

Přehled

AND Rd,Rr

Parametr *Rd*, parametr *Rr*.

Popis

Instrukce smaže (nastaví na 0) zadaný bit v určeném I/O registru.

Odkazy

Související instrukce: **FIXME**:avr.isa.sbi,.

- *8-bit AVR Instruction Set Rev. 0856I-AVR-07/10*, strana 19

ANDI

Jméno

ANDI — Logical AND with Immediate

Přehled

ANDI dD,K

Parametr *Rd* ($16 \leq d \leq 31$), parametr *K* ($0 \leq K \leq 255$).

Popis

Instrukce provede logický součin registru *Rd* s osmibitovou konstantou a výsledek uloží opět do registru *Rd*.

Odkazy

Související instrukce: AND.

- *8-bit AVR Instruction Set Rev. 0856I-AVR-07/10*, strana 20

BREAK

Jméno

BREAK — Break

Přehled

BREAK

Instrukce nemá žádné parametry.

Popis

Instrukci používá On-chip Debug system.

Odkazy

Související instrukce: **FIXME**:avr.isa.nop;.

- *8-bit AVR Instruction Set Rev. 0856I-AVR-07/10*, strana 28

BST

Jméno

BST — Bit Store from Bit in Register to T Flag in SREG

Přehled

BST *Rd*,*bit*

Parametr *Rd* určuje jeden z registrů R0 až R31.

Popis

Instrukce smaže (nastaví na 0) zadaný bit v určeném I/O registru.

Obrázek 40-1. Formát instrukce BST

0	1111 101	6	7	dddd	11	12	0	13	KKK	15
---	----------	---	---	------	----	----	---	----	-----	----

Odkazy

Související instrukce: SBI.

- 8-bit AVR Instruction Set Rev. 0856I-AVR-07/10, strana 46

CBI

Jméno

CBI — Clear Bit in I/O Register

Přehled

CBI *port*,*bit*

Parametr *port* určuje číslo I/O registru, a parametr *bit* pak bit v tomto registru.

Popis

Instrukce smaže (nastaví na 0) zadaný bit v určeném I/O registru.

Obrázek 40-1. Formát instrukce CBI

1001 1000								AAAAA				bbb		
0						7	8				12	13		15

Odkazy

Související instrukce: SBI, SBIC, SBIS.

- 8-bit AVR Instruction Set Rev. 0856I-AVR-07/10, strana 48

LDS

Jméno

LDS — Load Direct from Data Space

Přehled

LDS Rd,K

Parametr *Rd* ($16 \leq d \leq 31$), parametr *K* ($0 \leq K \leq 127$),

Popis

Naplní registr *Rd* hodnotou z RAM (I/O, SRAM, REG) adresy *K*. Přístup je omezen na RAM adresy od 0x40 do 0xbf.

Odkazy

Související instrukce: LPM.

- *8-bit AVR Instruction Set Rev. 0856I-AVR-07/10*, strana 96

LPM

Jméno

LPM — Load Program Memory

Přehled

LPM

LPM Rd,Z

LPM Rd,Z+

První parametr funkce je jeden z 30 registrů procesoru. ($0 \leq Rd \leq 29$). Pro registry R30 a R31, není chování instrukce definováno.

Popis

LPM je jediná instrukce, která nám umožňuje číst obsah paměti programu. Funguje tak, že do registru R0 nebo *Rd* načte jeden bajt z Flash paměti programu, bajt na který ukazuje registr *Z*.

Je nutné připomenout, že paměť programu je organizována po 16-ti bitových slovech. Instrukce LPM ji však čte po bajtech. Registr *Z* tedy dokáže adresovat prvních 64kB paměti programu, což je prvních 32kW. Pokud

procesor obsahuje registr RAMPZ, jsou jeho bity součástí adresy před bity registru Z. Dokážeme tak adresovat 2^{24} bajtů tedy 2^{23} slov paměti programu.

Varování

Varianta instrukce LPM Rd,Z+ při přetečení Z nezvyšuje RAMPZ.

Bajty ve slově jsou adresovány od nižšího, jak je vidět v následující tabulce.

Z	PC
0	0 LO Byte
1	0 HI Byte
2	1 LO Byte
3	1 HI Byte

Ukázky

```
ldi ZH, high(CodeTable >> 1) ; Initialize Z-pointer
ldi ZL, low(CodeTable >> 1)  ;+
lpm r16, Z                    ; Load from program memory
...
```

```
CodeTable:
    .dw 0x4281, 0x4316, 0x45a9
```

Odkazy

Související instrukce: **FIXME**:avr.isa.nop;.

- 8-bit AVR Instruction Set Rev. 0856I-AVR-07/10, strana 97

RET

Jméno

RET — návrat z podprogramu

Přehled

RET

SBI

Jméno

SBI — Set Bit in I/O Register

Přehled

SBI *port,bit*

Instrukce má dva parametry: *port* ($0 \leq \text{port} \leq 31$) a *bit* ($0 \leq \text{bit} \leq 7$).

Popis

Instrukce nastaví zadaný bit v určeném I/O registru.

Instrukce se provádí po dobu dvou hodinových cyklů. Na procesorech ATtiny s redukováným jádrem (např. ATtiny4, ATtiny5, ATtiny9, ATtiny10) trvá vykonání instrukce jen jeden hodinový cyklus.

Odkazy

Související instrukce: CBI, SBIS.

- *8-bit AVR Instruction Set Rev. 0856G-AVR-07/08*, strana 117
- *8-bit AVR Instruction Set Rev. 0856I-AVR-07/10*, strana 123

SBIC

Jméno

SBIC — Skip if Bit in I/O Register Cleared

Přehled

SBIC *port,bit*

Instrukce má dva parametry: *port* a *bit*.

Popis

Odkazy

Související instrukce: CBI, SBI, SBIS.

- *8-bit AVR Instruction Set Rev. 0856I-AVR-07/10*, strana 124

SBIS

Jméno

SBIS — Skip if Bit in I/O Register is Set

Přehled

SBIS *port,bit*

Instrukce má dva parametry: *port* a *bit*.

Popis

Istrukce zkoumá stav bitu *bit* na I/O adrese *port*. Má-li tento bit hodnotu "1", tedy je-li nastaven, je následující instrukce přeskočena a pokračuje si instrukcí za ní.

Není-li podmínka skoku splněna, trvá provedení instrukce 1 hodinový cyklus. Je-li podmínka splněna a docází k přeskoku, trvá provedení instrukce 2 až 3 hodinové cykly podle toho jestli se přeskakuje instrukce dlouhá jedno nebo dvě slova.

Odkazy

Související instrukce: CBI, SBI, SBIC.

- *8-bit AVR Instruction Set Rev. 0856I-AVR-07/10*, strana 125

SBR

Jméno

SBR — Set Bits in Register

Přehled

SBR Rd, K

Parametr *Rd* určuje se kterým registrem se pracuje, a parametr *K* je 8-mi bitová maska.

Popis

Instrukce nastaví na 1 bity v registru *Rd* podle masky *K*.

$Rd \longrightarrow Rd \vee K$

Obrázek 40-1. Formát instrukce SBR

0	0100	3	4	KKKK	7	8	dddd	11	12	KKKK	15
---	------	---	---	------	---	---	------	----	----	------	----

Odkazy

Související instrukce: SBI, CBI.

- *8-bit AVR Instruction Set Rev. 0856I-AVR-07/10*, strana 127

SBRC

Jméno

SBRC — Skip if Bit in Register is Cleared

Přehled

SBRC Rr,bit

Parametr *Rr* a parametr *bit*.

Popis

Je-li bit v registru *Rr* smazán ($== 0$), pak je následující instrukce přeskočena.

Odkazy

Související instrukce: SBI, SBR, SBRS.

- *8-bit AVR Instruction Set Rev. 0856I-AVR-07/10*, strana 128

SBRS

Jméno

SBRS — Skip if Bit in Register is Set

Přehled

SBRS *Rr*,*bit*

Parametr *Rr* a parametr *bit*.

Popis

Je-li bit v registru *Rr* nastaven ($== 1$), pak je následující instrukce přeskočena.

Odkazy

Související instrukce: SBI.

- *8-bit AVR Instruction Set Rev. 0856I-AVR-07/10*, strana 129

40.6. Konstrukce a zapojení s AVR

* *Attributy: id="avr.constructions"*

Zdroje a odkazy:

- AVR ATtiny15 sinus led pulse¹¹⁷ by Lefinnois¹¹⁸ on YouTube
- Sample Atmel AVR Code for the Spark Fun Nokia-like graphic color LCD¹¹⁹

- Nokia 6610 LCD + Arduino¹²⁰
- The gnuusb hardware¹²¹
- RobotikaWiki¹²² — obsahuje informace a několik konstrukcí s AVR
- Stealth USB CapsLocker¹²³
- USB Physical Hit Counter based on AVR ATtiny25¹²⁴
- LCD2USB¹²⁵ s ATmega8
- InfraHID¹²⁶ s ATmega8
- USBTenki: USB Temperature sensors and more¹²⁷ s ATmega8
- USB-LED-Fader¹²⁸ s ATmega8
- i2c-tiny-usb¹²⁹
- USBasp¹³⁰ — USB programmer for Atmel AVR controllers
- USBtiny¹³¹
- Teensy USB Development Boards¹³² s AT90USB162 a AT90USB646
- LUFA (Formerly MyUSB) (2008)¹³³

UART:

- Enabling AVR UART tx¹³⁴
- Enabling AVR UART Rx¹³⁵

Bluetooth a bezdrátová komunikace:

- BLUE controller.com¹³⁶ — experimentální modul s ATmega328 (168,88,48) a Bluetooth a 868/434MHz radio
- The Jee Labs Shop¹³⁷ (JeeNode)

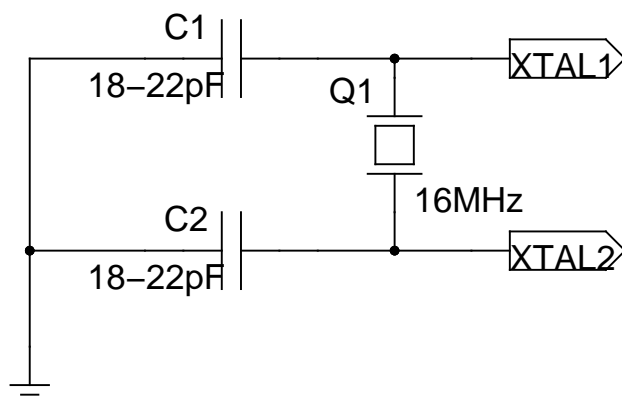
* <http://www.electrooons.com/2009/07/31/programmers-for-avr-serial-parallel/>

* <http://www.electrooons.com/2009/07/25/using-external-interrupt-in-avr/>

* <http://www.dharmanitech.com/2009/01/sd-card-interfacing-with-atmega8-fat32.html>

Na napájení MCU je třeba mít připojený *decoupling capacitor*. Kondenzátor který poskytuje MCU energii při špičkových odběrech a snižuje tak šíření rušení způsobeného těmito špičkovými odběry po napájení.

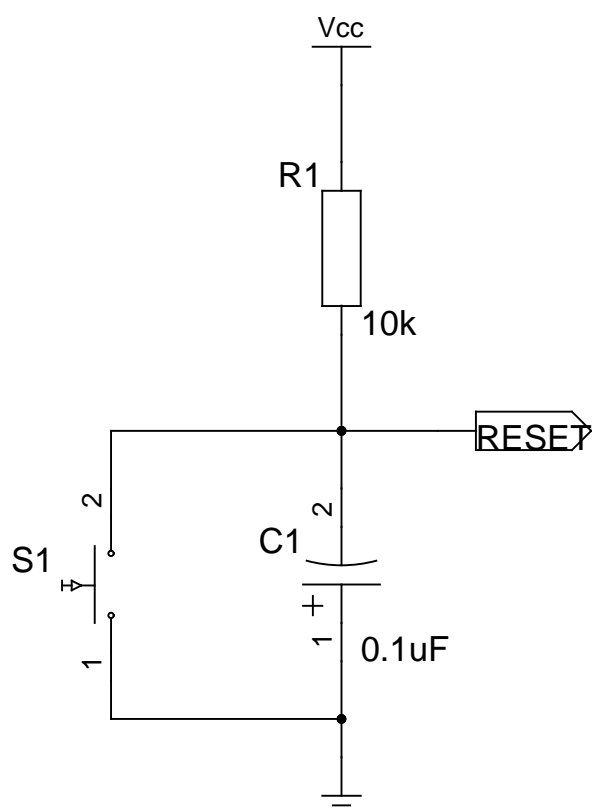
Obrázek 40-18. Připojení krystalu k MCU AVR



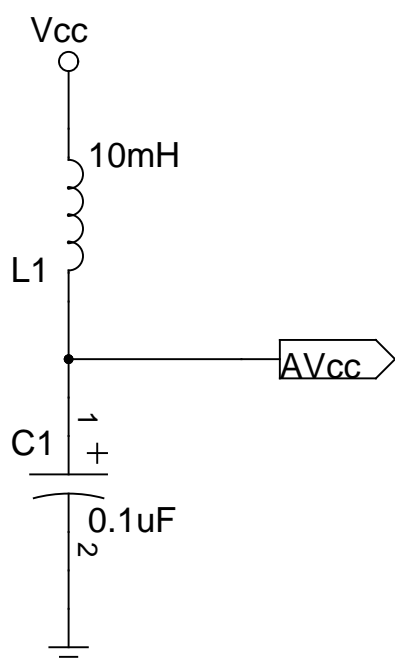
Kapacita kondenzátorů je určena vzorcem

$$C_1 = C_2 = 2 * (CL - 5pF)$$

Obrázek 40-19. Připojení resetovacího tlačítka



Obrázek 40-20. Filtr napájení AD převodníků



40.6.1. Experimentální desky a jiné nástroje

Odkazy:

- ATmega8 board¹³⁸
- Nálepky¹³⁹
- ATmega8 board a potisk PCB¹⁴⁰ [06.11.2008]
- ATmegaXX8 Target Board¹⁴¹ a ATtiny2313 Target Board¹⁴² od Evil Mad Science
-
-
- Diavolino¹⁴³ \$13
- AVR Starter Packs¹⁴⁴ \$110 - \$140
-
-

Pracujeme-li s nepájivým kontaktním polem, velmi se nám budou hodit udělátká.

Udělátká k nepájivému poli:

- ISP-Adapter for the breadboard¹⁴⁵
- ATtiny breadboard headers¹⁴⁶
- Bread-Power¹⁴⁷

Odkazy:

- <http://www.pocketmagic.net/?p=375>
-

40.6.2. Čítač / měřič kmitočtu

Zdroje a odkazy:

- Build your 40 MHz Frequency meter¹⁴⁸ by Ibrahim Kamal, Last update 2008-09-05
- Countermeasures¹⁴⁹ by Jasperh
- RS-232 Freq. Meter/Pulse Generator Based on Atmel AT902313¹⁵⁰ by Richard Capps
- A Little More Serious Frequency Meter (ATtiny2313)¹⁵¹ by Richard Capps
- A microprocessor based frequency counter up to 70MHz¹⁵²
-
-

```

;
        .org      0
        rjmp     reset

reset:
        ; Initialize SP to ramend
        ldi      temp,low(ramend)
        out      spl,temp
        ; Initialize baudrate for uart
        ldi      temp,baudconstant
        out      ubrr,temp
        :

; command interpretation loop for frequency meter

```

```

loop:

:
:
reset:
    ...
    ldi    temp,0b10000110        ; Enable input to 16 bit counter (Timer/Counter 1).
    out    tcclrb,temp
    ...

    ; Enable the watchdog
    wdr                                ; must be
    ldi    temp,0b00001111        ; WDT enable
    out    wdtcr,temp            ;+

    sei                                ; Enable interrupts
    rjmp   Main                    ; Start execution of main loop.

; Measure frequency for 1, 10 and 100 seconds.
measurefreq:
    ldi    temp,0                    ; Clear 16 bit counter
    out    tcntlh,temp
    out    tcntl1,temp
    clr    ZL                        ; Clear ZL which is the overflow counter.

    sbi    PORTD,4                    ; Pulse prescaler reset
    cbi    PORTD,4

    ldi    temp,0b10000000        ; Enable overflow interrupts from 16 bit timer.
    out    TIMSK,temp

    cbi    DDRD,3                    ; Make gate pin high impedance
    cbi    PORTD,3                ; Gate signal into prescaler - turn off pullup

AnotherSecond:
    rcall   Delay1Second            ; Delay 1 second.
    dec     iterations              ; Measure more than 1 second if necessary
    brne    AnotherSecond

    sbi     DDRD,3                    ; Make gate signal low Z.
    sbi     PORTD,3                ; Disable signal into prescaler.

    clr     temp                    ; Disable overflow interrupts.
    out     TIMSK,temp              ;+ Note: using tcclrb to disable counter causes 1+.
    ret

```

40.6.3. Připojení TV či monitoru

Odkazy:

- tvText¹⁵³ a video¹⁵⁴
- Composite Video Text/Graphics Display¹⁵⁵
- AVGA AVR Based color video game development platform¹⁵⁶
- Simple VGA/Vide adapter¹⁵⁷
- Simple VGA/Video adapter¹⁵⁸ na CircuitDB
- Text on TV¹⁵⁹

40.6.4. Software USB

Odkazy:

- V-USB¹⁶⁰ Virtual USB port for AVR microcontrollers
- Hardware Considerations¹⁶¹
- AVR-CDC¹⁶²
- AVR 1-Key-Keyboard Project¹⁶³
- 49

40.6.5. Připojení externí RAM

Odkazy:

- Simple evaluation board for ATmega162 + 32KB external RAM¹⁶⁴
- Adding external memory to Atmega128¹⁶⁵

Externí paměť RAM lze připojit k AVR několika způsoby. Když pominu případné sériové RAM s rozhraním SPI či možná I2C, a pominu možnost připojení dynamické RAM, budu se zabývat možnostmi připojení statické RAM paměti.

Nejdříve se ve zkratce zmíním o rozhraní XRAM. Toto rozhraní mají zabudovány některé MCU. Z MCU v pouzdru DIL je to ATmega162. Rozhraní sestává ze dvou osmibitových portů a jednoho tříbitového portu. Na osmibitových portech jsou na prvním PC signály A8-A15, a na druhém PA multiplexovány D0-D7 s A0-A7. Další tři bity jsou ALE pro multiplexování adresy a dat. A signály !RD a !WR. XRAM funguje velmi rychle, je jen o málo pomalejší než přístup k interní RAM.

Nyní připojení externí RAM na MCU které nemají podporu ve formě XRAM rozhraní. Tou nejpomalejší možností je použít sério-paralelních převodníků, obvodů HC595 a HC165. Současně tento způsob připojení je nejméně náročný na počet I/O pinů.

Pokud nám jde o rychlost, a nemáme obvod s velkým počtem pinů, můžeme se inspirovat řešením XRAM ale zajít ještě o kousek dále. Na jednom portu realizujeme datovou obousměrnou sběrnici. To je 8 pinů. Na tuto sběrnici budeme multiplexovat adresové informace. Tyto budeme zachytávat do obvodů HC573 a získáme tak kompletní 16-ti bitovou adresovou sběrnici. Přidáním dalšího obvodu HC573 můžeme rozšířit sběrnici na 24 adresových vodičů.

Chci použít MCU řady 48/88/168/328. Na tomto MCU jsou vyvedeny 3 porty PB, PC a PD. Funkce jednotlivých vývodů ovšem omezuje naše možnosti. PC není úplný ale má jen 6 vývodů. PC6 je použit jako RESET. PB je sice 8-mi bitový, ale na PB6 a PB7 je připojen krystal. PD je také osmibitový, ale PD0 a PD1 jsou obsazeny sériovým rozhraním RX/TX které potřebuji. Optimální by bylo užití většího obvodu ve 40-ti vývodovém pouzdrě jako je například ATmega644.

40.6.6. PWM

Odkazy:

- ATmega8 Basics, PWM¹⁶⁶
-
-

40.6.7. Počítač postavený okolo ATmega162

Jedná se o předběžný návrh.

Parametry:

- Připojená externí paměť přes XMEM rozhraní. Kapacita 512KB, stránkováno.
- SPI sběrnice, adresováno 8 zařízení
- 2*RS323
-
-

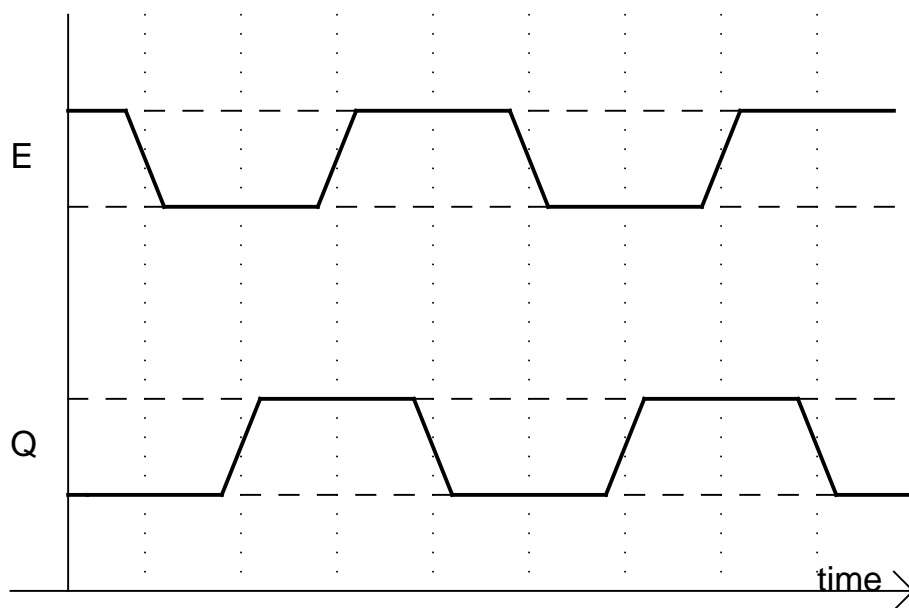
40.6.8. Generátor hodinových signálů pro 6809E

Varování

Tato konstrukce nebyla nikdy postavena. Jedná se jen o úvodní rozvahu.

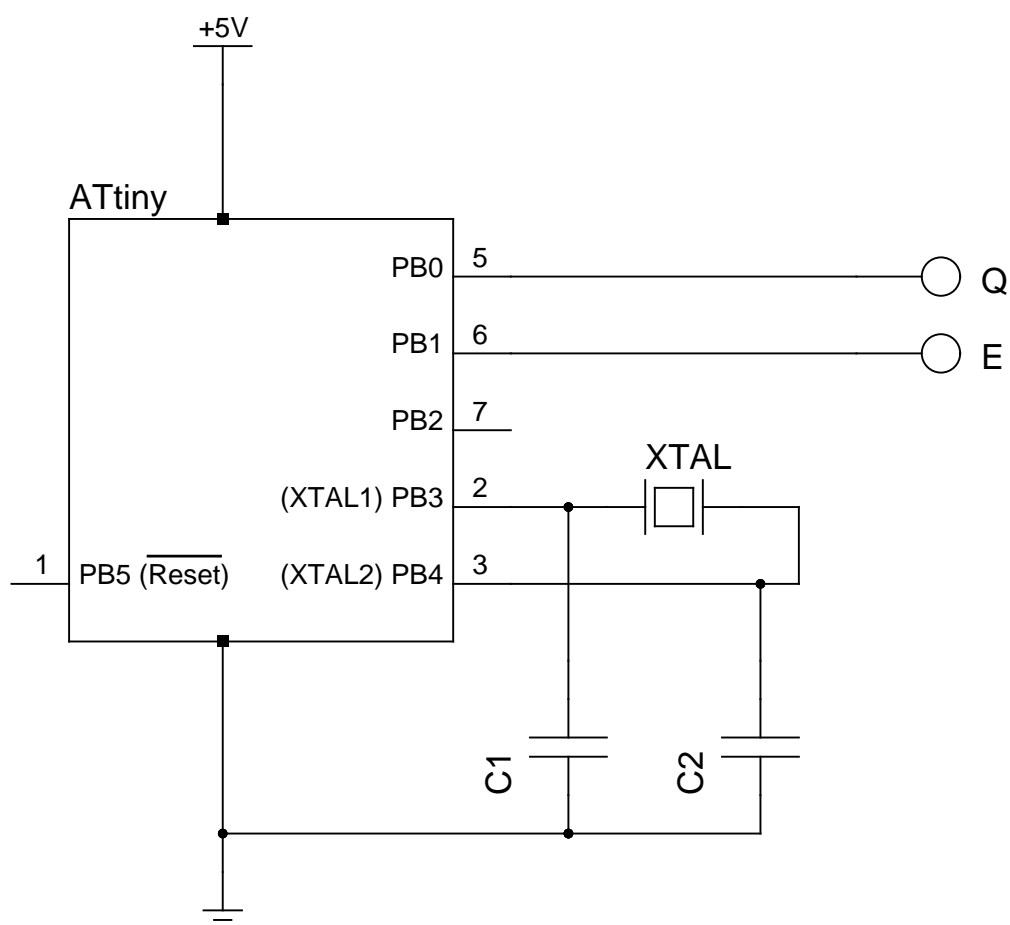
Procesor 35 je variantou 35 která nemá vnitřní oscilátor řízený vnějším krystalem, ale potřebuje ke své činnosti vnější zdroj dvoufázových hodin. Tyto hodiny se přivádějí na vstupní piny E a Q mikroprocesoru. Průběh dvoufázového signálu je na obrázku.

Obrázek 40-21. Dvoufázové hodiny pro MC6809E



Ke generování signálu lze použít jakýkoliv procesor, jenž je dostatečně rychlý, tedy 16MHz. Protože stačí malý počet vývodů pouzdra, použil jsem ATtiny.

Obrázek 40-22. Generování Dvoufázových hodiny pro MC6809E s ATtiny



```

LOOP:  SBI B,1      ; E ↑
        NOP
        NOP
        SBI B,0     ; Q ↑
        NOP
        NOP
        CBI B,1     ; E ↓
        NOP
        NOP
        CBI B,0     ; Q ↓
        RJMP LOOP

```

40.6.9. Konstrukce počítačů založených na MCU AVR

*

Odkazy:

- FIGnition¹⁶⁷ ()
-

40.7. Knihovna kódů

Knihovna kódů které jsem našel, vymyslel nebo odzkoušel. Knihovna není ve stavu přímo použitelném. Jedná se spíše o kousky kódu realizující zajímavé funkce. Při použití v nějakém projektu je třeba je přizpůsobit.

40.7.1. Morse Talker

Při experimentech s ATtiny15L jsem zjistil, že nemám volné vývody na připojení nějakého zobrazovače hodnoty. Například LCD displeje. Dostal jsem tedy nápad a k jednomu volnému pinu PB4 připojil sluchátko. Hlavní myšlenka je, že když nemohu hodnotu zobrazovat, můžu ji odvyšlat morseovkou.

Následující kód je místy velmi divoký a řadu instrukcí jsem nahradil jen komentáři. Jsou to totiž instrukce značně závislé na způsobu kterým měřím čas. Rovněž tento kód není moc přehledný. Ke své lítosti jsem opravdu nemohl použít podprogramy, které by kód velmi zpřehlednily. Program vznikl a ladil jsem jej na MCU ATtiny15L, který má zásobník návratových adres jen 3 úrovně hluboký. Počítaje přerušení které si může vzít kdykoliv jednu úroveň, a proceduru TX_Message která vysílá celý řetězec znaků a používá pro vysílání jednotlivých znaků tuto proceduru TX_Symbol. Tím jsem vyčerpal všechny úrovně zásobníku návratových adres.

```
; Transmit one morse symbol (character) given in register ARG.
TX_SYMBOL:
    ; Code to handle 0x20 (Space) as a special case.
    CPI    Arg, 0x20
    BREQ   Send2T

    ; Initialize Z pointer to MorseTab
    LDI    ZL, low(MorseTab << 1)
    LDI    ZH, high(MorseTab << 1)

    ; Make some correction to arg. The table begins with space, code=0x20.
    SUBI   Arg, 0x20

    ; Add Arg to Z, so Z will point to character given by Arg.
    ADD    ZL, Arg
    BRCC   LoadCode
    INC    ZH

LoadCode:
    LPM                                ; Now R0 contains encoded symbol.
    MOV    Seq, R0

SendLoop:
    CLC                                ; Must be cleared before ROR
    ROL    Seq
    ; The C contains di/dat, but only if Seq is non zero.
    ; In such case Seq is empty and C contains stop mark.
    BREQ   EndSymbol

    ; We can send a . or - depending on the Carry value.
    SBR    BITF, 0b00000001 ; Sound On
    BRCC   SendDih
```



```

; Vysíláme čárku (dah). Protože zvuk je zaplý.

;Wait 1T
;Wait 1T
;Wait 1T

CBR      BITF, 0b00000001 ; Sound Off

;Wait 1T
RJMP     SendLoop

EndSymbol:
        ;Wait 1T
        ;Wait 1T
        RET

```

Na kódu je zvláštní jedna věc. Se zdrojem časových impulsů a zdrojem tónu komunikuje přes bity v registru BitF. Bit 0 tohoto registru určuje, zdali je vytvářen tón. 0 znamená že nikoliv a je tedy ticho, 1 znamená že se generuje tón. Další bit 1 tohoto registru slouží ke komunikaci se zdrojem času. Zdroj času totiž tento bit v pravidelných intervalech 1T nastavuje na 1. Všechna čekání jsou tedy krátké posloupnosti tří instrukcí podobné této:

```

        ; Wait 1T
CBR      BitF, 0b00000010 ; smaž bit 1
Loop:    SBRS   BitF, 1      ; čekej dokud jej zdroj času opět nenastaví
        RJMP   Loop

```

Kód programu je rovněž „zkrácen“ vzájemným nakombinováním různých větví do sebe. Například vysílání čárky a tečky je udělané tak že tečka je poslední třetina čárky.

```

        Sound On
        If tečka Go tecka
        Wait 1T
        Wait 1T
tecka:   Wait 1T
        Sound Off
        Wait 1T

```

Za každou tečku nebo čárku je rovněž ihned vysláno ticho o délce 1T což mi zjednodušuje vysílání posloupnosti teček a čárek. Na to pak musím myslet, když dělám mezeru mezi písmeny, že kousek ticha o délce 1T už byl odvysílán a že zbývají jen dva kousky ticha.

Poslední částí, která je nutná k pochopení programu je tabulka morse kódů.

```

MorseTab:
        .db 0b00000000, 0b10101110, 0b01001010, 0b00000000 ; SP ! " #
        .db 0b00000000, 0b00000000, 0b01000100, 0b01111010 ; $ % & '
        .db 0b10110100, 0b10110110, 0b00000000, 0b01010100 ; ( ) * +
        .db 0b11001110, 0b10000110, 0b01010110, 0b10010100 ; , - . /

        ; 0x30 - 0x39 Numbers
        .db 0b11111100, 0b01111100, 0b00111100, 0b00011100 ; 0 1 2 3
        .db 0b00001100, 0b00000100, 0b10000100, 0b11000100 ; 4 5 6 7
        .db 0b11100100, 0b11110100 ; 8 9

        ; 0x3A - 0x3F

```

```
.db 0b11100010, 0b10101010, 0x00000000, 0b10001100 ; : ; < =
.db 0b00000000, 0b00110010 ; > ?

; 0x40 Alphabet
.db 0b10000000, 0b01100000, 0b10001000, 0b10101000 ; @ A B C
.db 0b10010000, 0b01000000, 0b00101000, 0b11010000 ; D E F G
.db 0b00001000, 0b00100000, 0b01111000, 0b10110000 ; H I J K
.db 0b01001000, 0b11100000, 0b10100000, 0b11110000 ; L M N O
.db 0b01101000, 0b11011000, 0b01010000, 0b00010000 ; P Q R S
.db 0b11000000, 0b00110000, 0b00011000, 0b01110000 ; T U V W
.db 0b10011000, 0b10111000, 0b11001000, 0b00000000 ; X Y Z [
.db 0b00000000, 0b00000000, 0b00000000, 0b00110110 ; \ ] ^ _
```

Vymyslel jsem spoustu složitých kódování a způsobů jak pospat morse pro jedno písmeno v jednom bajtu. Ale pak se mi rožlo a já byl osvícen. Kódování se kterým jsem přišel je velmi jednoduché a ke všemu ještě dokáže kódovat o jednu tečku nebo čárku více než nejsložitější se kterým jsem přišel. Princip je takový že tečku kóduji nulovým bitem a čárku jedničkovým bitem. Jediné co potřebuji jednoznačně detekovat, je konec posloupnosti. A v tom je ten ftip. Konec posloupnosti je označen 1 po které následují již jen samé 0. Protože při vysílání posouvám kódové slovo jedním směrem, v mém případě do leva, a zajišťuji že se zprava plní nulami je detekce konce dána tím že posledním bitem který se do Carry načetl je 1 a v registru jsou jen samé nuly. Toto čtení kódu je realizováno jen třemi instrukcemi:

```
SendLoop:
    CLC ; Must be zero before ROR
    ROR    Seq
    BREQ    EndSymbol
```

40.7.2. Různé posbírané kousky

40.7.2.1. Main loop

```
//Include the default AVR io-header
//This file(avr/io.) holds all of the informations about the mcu's ports, setup-registers and
#include <avr/io.h>

//The normal main-routine
int main(void) {
    //Our main-loop
    for(;;) {
        //Do nothing
    }
    //The compiler requires us to return something, even though we never get here...
    return 0;
}
```

40.7.2.2. Čtení a psaní do portů

* AVR/Examples/IO¹⁶⁸

```
#include <avr/io.h>

int main(void) {
```

```

        DDRB = 0xff;          /* Set PORTB as output */
5       DDRA = 0x00;          /* Set PORTA as input */

        for(;;) {
            PORTB = PINA;      /* Set PORTB as PORTA. Copy bits from A to B */
        }
10     return 0;
    }

```

Jak jse vidět, uvedený program čte co je na portu A a podle toho nastavuje piny portu B.

40.7.2.3. Použití AD převodníku

* AVR/Examples/AD — Analog-Digital Example¹⁶⁹

```

uint16_t ReadADC() {
    uint16_t result = 0;          /* Temporary variable */
    PRR ^= (1<<PRADC);           /* Set the Power Reduction ADC-bit to 0 */

    ADMUX = (0<<MUX3)|(1<<MUX2)|(0<<MUX1)|(0<<MUX0); /* ADC4 and AREF voltage reference */
    DIDR0 = (1<<ADC4D); /* Disable the digital buffer at this pin */

    ADCSRA = (1<<ADEN)|(1<<ADSC); /* Start conversion */
    while(ADCSRA & (1<<ADSC))      /* Wait until the conversion is finished */
        ;

    result = (uint16_t)ADCL;        /* Get the lower 8 bits of the result */
    result += (((uint16_t)ADCH)<<8); /* upper 2 bits */

    ADCSRA = 0x00;                 /* Turn off the ADC module */
    PRR |= (1<<PRADC);

    return result;
}

```

40.7.3. Emulace jiného procesoru

Výkon AVR je takový, že se můžeme pokusit emulovat chování jiného procesoru. Zde uváděné kousky kódu se snaží realizovat některé operace, které při tom budeme potřebovat.

40.7.3.1. Instruction Fetch and decode

Načítání a dekódování instrukce

* **FIXME:**Udělat instrukce aktivními odkazy.

Jednou z operací kterou je třeba realizovat, je nahrání instrukce simulovaného procesoru z paměti a dekódování této instrukce. Následující příklad vychází z předpokladu, že paměť simulovaného procesoru je v RAM našeho AVR, nebo že je použita externí RAM mechanismy AVR například u čipu ATmega162.

Operace „Fetch“, tedy přečtení instrukce z operační paměti je v podstatě jednoduché přečtení buňky v paměti RAM na adrese dané naším čítačem instrukcí PC a zapsání této hodnoty do registru `Instr` kde se bude provádět dekódování instrukce. Nezbytnou součástí je také posunutí PC na další instrukci.

```
mem[PC] → Instr
PC = PC+1
```

Z instrukcí, které umožňují přístup do paměti RAM, se nám hodí jen:

Instrukce	Popis	Operace
LD Rd,X	Load Indirect	$Rd \leftarrow (X)$
LD Rd, X+	Load Indirect and Post-Increment	$Rd \leftarrow (X); X \leftarrow X+1$
LD Rd, Y	Load Indirect	$Rd \leftarrow (Y)$
LD Rd, Y+	Load Indirect and Post-Increment	$Rd \leftarrow (Y); Y \leftarrow Y+1$
LD Rd, Z	Load Indirect	$Rd \leftarrow (Z)$
LD Rd, Z+	Load Indirect and Post-Increment	$Rd \leftarrow (Z); Z \leftarrow Z+1$

Musíme se tedy rozhodnout, v kterém registru budeme uchovávat hodnotu čítače instrukcí PC. Registr (registrový pár) z používají rovněž instrukce LPM, ELPM a SPM pro práci s pamětí programu. Z toho důvodu bych je nepoužil pro registr PC. Zbývají registry X a Y jenž jsou rovnocenné. Mohu si tedy vybrat kterýkoliv z nich. V dalším pak používám jako PC registrový pár X.

Kód pro Fetch tedy vypadá takto

```
Fetch:
    ld Instr, X+          ; Přečtení instrukce z RAM
```

Takhle jednoduché je to proto, že používáme instrukce pro práci s vnitřní či vnější pamětí RAM, a taky proto, že jsme mohli vyhradit registr X pro čítač instrukcí. Pokud bychom si nemohly vyhradit registr X, a museli používat jeden z X,Y,Z jen dočasně, mohli bychom si vypomoci instrukcí MOVW.

```
Fetch:
    movw    Z, PC          ; načtení hodnoty PC do indexového registru
    ld      Instr, Z+       ; přečtení instrukce z RAM
    movw    PC, Z           ; uložení nové hodnoty Z do PC
```

Výsledný program tedy bude o tyto dvě MOVW instrukce pomalejší. Protože každá si veme jeden hodinový takt, kód je o dva takty delší. Samotná instrukce LD spotřebuje další dva takty pro čtení z vnitřní RAM. V případě externí paměti RAM může dojít a patrně dojde ještě k dalšímu spomalení. Pro konkrétní hodnoty a použití externí RAM najdete informace o potřebném počtu taktů hodinového signálu v dokumentaci daného procesoru.

Dalším krokem je dekodování instrukčního kódu načteného do registru Instr. Tady hodně záleží na instrukční sadě našeho virtuálního procesoru. Některé mikroprocesory, jako je například CDP 1802, mají takovou instrukční sadu, že se kód instrukce rozpadá na dvě pole. První určuje instrukci a druhé registr či variantu instrukce. Takovou instrukční sadu bych pravděpodobně dekoval programově.

Při bližším pohledu do instrukční sady AVR, nalezneme instrukce:

Instrukce	Popis	Operace	Takty
IJMP	Indirect Jump to (Z)	$PC(15:0) \leftarrow Z;$ $PC(21:16) \leftarrow 0$	2
EIJMP	Indirect Jump to (Z)	$PC(15:0) \leftarrow Z;$ $PC(21:16) \leftarrow EIND$	2
ICALL	Indirect Call to (Z)	$PC(15:0) \leftarrow Z;$ $PC(21:16) \leftarrow 0; \dots$	3/4

Instrukce	Popis	Operace	Takty
EICALL	Indirect Call to (Z)	PC(15:0) \leftarrow Z; PC(21:16) \leftarrow EIND; ...	4

Jsou to jediné instrukce, které dovolují nepřímý skok nebo volání podprogramu. S použitím těchto instrukcí mohou napsat něco jako:

```
Decode: ; Dekódování instrukce.
        ldi ZL, low(OpTab)
        ldi ZH, high(OpTab)
        add ZL, Instr          ; získání adresy v tabulce OpTab
        jmp          ; skok na odpovídající řádek v tabulce
```

Místo instrukce ICALL/EICALL jsem použil IJMP, případně EIJMP. Důvodem je, že neplánuji kód jednotlivých virtuálních instrukcí volat ještě z jiného místa v programu, a chci ušetřit pár taktů. Proto místo, pro někoho možná logického

```
...
icall
...
```

Kdy jednotlivé podprogramy končí instrukcí RET. Použiji

```
...
ijmp
NextInstruction:
...
```

A jednotlivé "podprogramy" ukončím instrukcí **RJMP NextInstruction**. Tato instrukce je rovněž rychlejší než instrukce **RET**.

Pro maximální zrychlení emulace, provádím řadu optimalizací v nejčastěji používaném kódu, t.j. ve zde uvedeném kódu nahrávání a dekódování instrukce. Dekódování provádím skokem na proceduru podle obsahu instrukce (OpCode). Tabulka skoků má 256 položek. Pro emulování jednoduchého procesoru stačí jedna takováto tabulka. Pro emulování procesoru s velmi bohatou sadou instrukcí, jako je například Z80 bych takovýchto tabulek potřeboval více. Důvodem jsou prefixové instrukce tohoto procesoru.

Ve svém prvním pokusu se omezují na jednoduchý procesor Forthovského typu.

```
; Pracovní registr, mě oblíbené jméno je W. Tento registr musí být mezi
; horními 16 (tedy 16-31) protože je často plněn instrukcí ldi.
.def      W          = R16

; Registry použité při emulaci virtuálního procesoru.
.def      Zero       = R12    ; Konstanta 0 použitá k urychlení některých výpočtů.
.def      OpCode     = R13    ; Registr obsahující načtenou instrukci k dekódování.
.def      OpTabL     = R14    ; Dvojce registrů tvořící ukazatel na tabulku operací.
.def      OpTabH     = R15    ;+

; Před spuštěním emulace je třeba naplnit některé registry počátečními
; hodnotami. Jedná se zejména o registry Zero a OpTab jenž slouží jako
; konstanty pro urychlení operací.
        ; Naplnění OpTabL:OpTabH adresou OpTab
        ldi      W, low(OpTab)
        mov      OpTabL, W
        ldi      W, high(OpTab)
```

```

mov      OpTabH, W

; Naplnění Zero konstantou 0
ldi      W, 0
mov      Zero, W

; Čítač instrukcí virtuálního procesoru je v registrovém páru X.

Fetch:   ld      OpCode, X++      ; Získání instrukce z paměti
; Dekódování instrukce pomocí OpTab
movw     ZL, OpTabL              ; Move OpTab address into Z
add      ZL, OpCode              ; Z = Z+OpCode
adc      ZH, Zero                ;+
; Nyní ukazuje Z na n-tou položku OpTab podle hodnoty OpCode
ijmp     ; Předáme řízení na příslušnou proceduru.

; Tabulka OpTab. Pro každou hodnotu OpCode je zde adresa (skok) na rutinu
; realizující příslušnou instrukci virtuálního procesoru. Tabulka obsahuje
; maximálně 256 položek. Pokud obsahuje méně než 256 položek, měli bychom
; zajistit že nebude dekódována instrukce která není v tabulce.
OpTab:
rjmp     INOP                    ; OpCode=0, NOP
rjmp     IDUP                    ; OpCode=1, DUP
rjmp     IDROP                   ; OpCode=2, DROP
rjmp     IADD                    ; OpCode=3, ADD
...
rjmp     ILAST                   ; OpCode=255, LAST

; Instrukce NOP neprovádí nic, pouze se vrátí zpět do do smyčky emulátoru.
INOP:    rjmp     Fetch

IDUP:    ...
...
rjmp     Fetch

```

Poznámky ke kódu

- Při výpočtu hodnoty registru Z vycházím z toho že v tabulce OpTab zabírá každá instrukce jedno slovo. Pokud použiji instrukci **avr.isa.rjmp**, je tento předpoklad splněn. Je třeba si uvědomit, že instrukce skoku RJMP může předat řízení/skočit v rozsahu -2048 / +2047 slov od aktuální adresy. Procedury realizující jednotlivé instrukce virtuálního procesoru se tedy musí nacházet v této vzdálenosti od své položky v OpTab.

Varování

Uvedený kód nebyl testován. Podařilo se mi ho bez problémů přeložit assemblerem, ale není zaručena jeho funkčnost. Je třeba ještě zkontrolovat několik věcí.

40.7.4. Měření napětí (použití ADC)

Odkazy:

- ATtiny261/461/861 datasheet, strana 143
- AVR a AD převodník¹⁷⁰ (2007-05-25)
- ADC (Avr)¹⁷¹
- Using the "Mini 3 digit display" — Not only 2.56V and higher¹⁷²

bit	7	6	5	4	3	2	1	0
name	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

bit	7	6	5	4	3	2	1	0
name	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

ADEN zapíná a vypíná AD převodník. (1 = zapnuto)

ADSC startuje převod v průběhu převodeu čteme 1. Pokud se bit změní na 0 je převod ukončen.

```

adc_init:
    ADMUX =
    ADCSRA =
    rts

adc_read:
    admux = kanal
    adcsra |= 0x40;
    while ((adcsra & 0x10) == 0); // Wait
    ADCSRA |= 0x10;
    // result in ADCW

#include <avr/io.h>
#include <inttypes.h>

uint16_t readADC(uint8_t channel) {
    uint8_t i;
    uint16_t result = 0;

    // Den ADC aktivieren unr Teilungsfaktor auf 64 stellen
    ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1);

    // Kanal des Multiplexers waehlen
    ADMUX = channel;

    // Interne Referenzspannung verwenden (also 2.56 V)
    ADMUX |= (1<<REFS1) | (1<<REFS0);

    // Den ADC initialisieren und einen sog. Dummyreadout machen
    ADCSRA |= (1<<ADSC);
    while (ADCSRA & (1<<ADSC));

    for (i=0; i<3; i++) {
        ADCSRA |= (1<<ADSC);
        while (ADCSRA & (1<<ADSC));
    }
}

```

```

        result += ADCW;
    }
    ADCSRA &= ~(1<<ADEN);
    result /= 3;
    return result;
}

int main(void) {
    uint16_t result = readADC(0)
    return 0;
}

```

Měření napětí

```

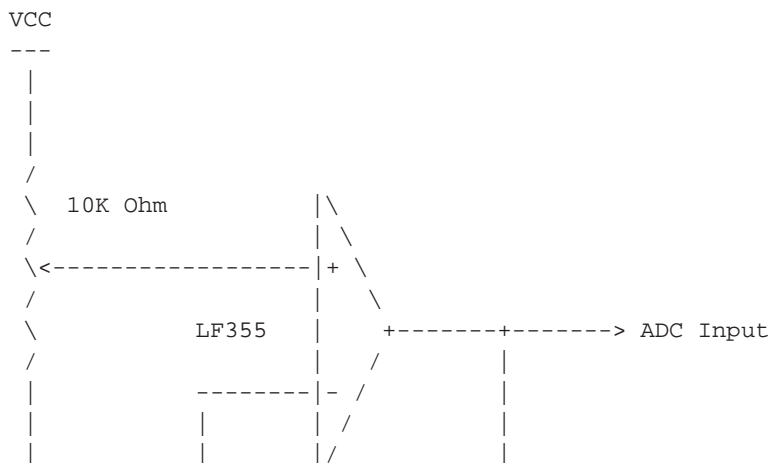
#include <avr/io.h>
#include <inttypes.h>

/*
 * Inicializace ADC převodníku. Tato inicializace je specifická pro
 * naši aplikaci a způsob použití převodníku.
 */
void initADC(void) {
    ADCSR |= (1<<ADPS2) | (0<<ADPS1) | (0<<ADPS0); /* Prescale = 16 */
    ADMUX |= (1<<REFS1) | (1<<REFS0); /* Voltage refference 0b11 = 2.56V */
    // ADCSR |= (1<<ADFR); /* Enable free-running mode */
    ADMUX = currentChannel; /* Initial channel selection */
    // ADCSR |= (1<<ADIE); /* Enable ADC conversion complete interrupt */
    // _SEI(); /* Enable global interrupts */
    // ADCSR |= (1<<ADSC); /* Start first conversion
} /* initADC() */

uint16_t readADC(uint8_t channel) {
    ADMUX = channel;
    ADCSRA |= (1<<ADSC);
    ADCSRA |= (1<<ADEN); /* Start ADC conversion */
    while (ADCSRA & (1<<ADSC));
    result = ADCW;
    ADCSRA &= ~(1<<ADEN); /* Stop ADC conversion */
}

```

Obrázek 40-23. Oddělení vstupu





40.7.5. USB

Odkazy:

- LUFA (Formerly MyUSB) (2008)¹⁷³
- V-USB¹⁷⁴ — Virtual USB port for AVR® microcontrollers

Knihovna LUFA řeší/realizuje:

- Audio In Device
- Audio Out Device
- CDC Device
- CDC Host
- Dual CDC
- General Library Test Application
- Joystick Device
- Keyboard Device
- Keyboard and Mouse Device
- Keyboard Host
- Mass Storage Device
- Mass Storage Host
- MIDI Device
- Mouse Device
- Mouse Host
- RNDIS (CDC) Ethernet
- Still Image Host
- USB-RS232 Device (based on CDC Device demo)

Podporované obvody jsou:

- AT90USB1286
- AT90USB1287
- AT90USB646
- AT90USB647
- AT90USB162
- AT90USB82
- ATmega16U4
- ATmega32U4

Knihovna V-USB řeší/realizuje/projekty:

- CDC Device
- Keyboard and Mouse Device
- USB-RS232 Device (based on CDC Device demo)
- AVR-CDC¹⁷⁵

40.7.6. Manipulace s bity

*

Odkazy:

- Bitmanipulation¹⁷⁶ na mikrocontroller.net [německy]
- AVR GCC Tutorial (1) – Basic I/O Operations¹⁷⁷
-
-

Práce s jednotlivými bity a částmi bajtu má svá úskalí. A o tom se pokouším rozepsat.

Ve strojovém kódu, a tím pádem assembleru, máme instrukce: sbrc, sbrs, SBIC, SBIS, sbi, cbi, bst, bld, ...

40.7.6.1. Makra pro bitové operace

Odkazy na projekty:

- Makra pro bitové operace¹⁷⁸

```
#define BIT(x) (1 << (x))
#define SETBITS(x,y) ((x) |= (y))
#define CLEARBITS(x,y) ((x) &= ~(y))
#define SETBIT(x,y) SETBITS((x), (BIT(y)))
#define CLEARBIT(x,y) CLEARBITS((x), (BIT(y)))
#define BITSET(x,y) ((x) & (BIT(y)))
#define BITCLEAR(x,y) !BITSET((x), (y))
#define BITSSET(x,y) (((x) & (y)) == (y))
#define BITSCLEAR(x,y) (((x) & (y)) == 0)
#define BITVAL(x,y) (((x)>>(y)) & 1)
```

40.8. Arduino

* *Attributy: id="Arduino"*

Odkazy:

- arduino¹⁷⁹ na Google Code
- Arduino¹⁸⁰ na CS Wikipedii
- Arduino: První program - blikání LED¹⁸¹ na blogu Techdot.EU
- The World Famous Index of Arduino & Freeduino Knowledge¹⁸²
- Arduino Development Using NetBeans IDE¹⁸³ — takové Arduino bez Arduina
- Arduino: The Documentary¹⁸⁴ (film)
- arduino.tw¹⁸⁵ (čísky)
-

Články:

- Série článků¹⁸⁶ na ROOT.CZ [2010-08-12]
- Overclocking to 22.1184MHz¹⁸⁷
- Temperature Sensor + Arduino¹⁸⁸ [2008-07-05] (LM35)
-

Knihy:

- Practical Arduino¹⁸⁹
- Arduino Cookbook¹⁹⁰ Pub. Date: 2011-03-29, ISBN: 978-0-596-80247-9

-
-
-

Tutoriály a výukové materiály:

- Arduino Tutorial¹⁹¹
- Arduino & Microcontroller Workshop¹⁹² (MIDI)
-

Kde koupit:

- Snail Instruments¹⁹³ [CZ] [2010-09-08]
- HW Kitchen¹⁹⁴ — Open Source Electronics Cooking [2010-09-08]
- CzechDUINO.cz¹⁹⁵ český obchod s Arduinem a příslušenstvím
- Freeduino nano V1.0 with ATMEGA328P-AU¹⁹⁶
- Adafruit Industries¹⁹⁷
- PJRC¹⁹⁸
- The Jee Labs Shop¹⁹⁹ (JeeNode)
- SparkFun²⁰⁰
- Sanguino v1.0 PCB²⁰¹ v MakerBot STORE
- TINKER SOUP²⁰² [2011-03-25] DE €
-
-

Blogy které se alespoň částečně věnují Arduinu:

- Harvie píše blog²⁰³ [2008]
- ArduinoRS.net O primeiro site brasileiro dedicado ao Arduino²⁰⁴
-
-

Zajímavé konstrukce:

- A credit card sized Ethernet Arduino compatible controller board²⁰⁵
- The Arduino AA Undershield²⁰⁶ s obvodem MAX756
-
- Arduino Frequency Counter Library²⁰⁷
- Arduino Realtime Audio Processing²⁰⁸
- Input Channel Extender²⁰⁹
- Bitlash Online²¹⁰ — interpretovaný shell
- A credit card sized Ethernet Arduino compatible controller board²¹¹ — založeno na chipu ENC28J60-DIL
- Building a hygrometer with a HS1101²¹²
-

Arduino je softwarová a hardwarová platforma. Po hardwarové stránce se jedná o minimální konfiguraci postavenou okolo procesoru Atmel ATmega328 nebo podobného. Existují i implementace pro procesory v pozdřech s velkým počtem vývodů. Po softwareové stránce je Arduino vývojové prostředí (IDE) naprogramované v Javě, které generuje přímo binární kód a nahrává jej do mikrořadiče po sériové lince. Arduino tedy nepoužívá ISP.

Samotný název získalo Arduino po Italském králi Arduinovi²¹³.

Nezatržiděné odkazy:

- Rhyduino - Arduino via Managed Code²¹⁴
-

Obrázek 40-24. Zapojení vývodů podle MCU ATmega328

						+-----+		+-----+		
	(PCINT14/RESET)	PC6-		1	+---+	28		-PC5 (ADC5/SCL/PCINT13)	--a5	
rx/d0	-----	(PCINT16/RXD)	PD0-		2			27		-PC4 (ADC4/SDA/PCINT12)--a4
tx/d1	-----	(PCINT17/TXT)	PD1-		3			26		-PC3 (ADC3/PCINT11)-----a3
d2	-----	(PCINT18/INT0)	PD2-		4			25		-PC2 (ADC2/PCINT10)-----a2
(PWM) d3	--	(PCINT19/OC2B/INT1)	PD3-		5			24		-PC1 (ADC1/PCINT9)-----a1
d4	-----	(PCINT20/XCK/T0)	PD4-		6			23		-PC0 (ADC0/PCINT8)-----a0
		Vcc-		7				22		-GND (?AGND)
		GND-		8				21		-AREF
	(PCINT6/XTAL1/TOSC1)	PB6-		9				20		-AVcc
	(PCINT7/XTAL2/TOSC2)	PB7-		10				19		-PB5 (SCK/PCINT5)-----d13 (LED)
(PWM) d5	----	(PCINT21/OC0B/T1)	PD5-		11			18		-PB4 (MISO/PCINT4)-----d12
(PWM) d6	---	PCINT22/OC0A/AIN0	PD6-		12			17		-PB3 (MOSI/OC2A/PCINT3)--d11 (PWM)
d7	-----	(PCINT23/AIN1)	PD7-		13			16		-PB2 (SS/OC1B/PCINT2)-----d10 (PWM)
(PWM) d8	---	(PCINT0/CLKO/ICP1)	PB0-		14			15		-PB1 (OC1A/PCINT1)-----d9 (PWM)
						+-----+				

40.8.1. Instalace

*

Integrované vývojové prostředí Arduino je napsáno v Javě. Musíme tedy mít na počítači běhové prostředí javy nainstalováno. Na stránce Download the Arduino Software²¹⁵ si vybereme balíček podle operačního systému a stáhneme.

40.8.2. Tutoriál

*

Odkazy:

- eNTý raNTs Arduino - seznámení²¹⁶, Arduino - první projekt²¹⁷ [2008-11-13]
- Arduino 5 Minute Tutorials²¹⁸
-
-

tronixstuff tutoriál:

- Getting started with Arduino! – Chapter Zero²¹⁹
- Getting Started with Arduino! – Chapter One²²⁰
- Getting Started with Arduino! – Chapter Two²²¹
- Getting Started with Arduino! – Chapter Three²²² [2010-04-20]
- Getting Started with Arduino! – Chapter Four²²³ [2010-04-30] (74HC595)
- Getting Started with Arduino! – Chapter Five²²⁴ [2010-05-06]
- Getting Started with Arduino! – Chapter Six²²⁵ [2010-05-14] (7 segment led)
- Getting Started with Arduino! – Chapter Six addendum²²⁶
- Getting Started with Arduino! – Chapter Seven²²⁷ [2010-05-20] (BCD, I2C, DS1307)
- Getting Started with Arduino! – Chapter Eight²²⁸ [2010-05-26] (DS1307)
- Getting Started with Arduino! – Chapter Nine²²⁹ [2010-06-06] (LED matrix, 74HC505,)
- Getting Started with Arduino – Chapter Ten²³⁰
- Getting Started with Arduino – Chapter Eleven²³¹
- Getting Started with Arduino – Chapter Thirteen²³² [2010-07-24]
- Moving Forward with Arduino – Chapter 14 – XBee introduction²³³ [2010-08-06]
- Moving Forward with Arduino – Chapter 15 – RFID Introduction²³⁴ [2010-08-18]

Odkazy:

- Tutorial 10 for Arduino: Interrupts + Debouncing²³⁵
-

Kostra programu sestává ze dvou funkcí, funkce `setup`, která se volá jen jednou v rámci startování programu, a funkce `loop` která se opakovaně volá v nekonečné smyčce. Toto je princip fungování většiny sketchů na arduinu.

```
void setup(void)
{
}

void loop(void)
{
}
```

Abychom si mohli napsat první malý zkušební program, potřebujeme znát ještě tři knihovní funkce. První je funkce `pinMode`, tato nastaví příslušný pin jako výstupní (OUTPUT nebo vstupní (INPUT). Druhá funkce je `digitalWrite`, která zapíše digitální hodnotu na výstupní pin. Poslední je `delay`, která pozastaví vykonávání programu na zadaný počet milisekund. Nyní již máme vše a můžeme napsat první program.

```
int    ledPin = 13;      // Interní led
int    value;

void setup(void)
{
    pinMode(ledPin, OUTPUT);
}

void loop(void)
{
    value = digitalRead(ledPin);
    value = value==HIGH ? LOW : HIGH;
    digitalWrite(ledPin, value);
    delay(500);
}
```

40.8.2.1. Čtení digitální hodnoty z pinu a zápis

Odkazy:

- Ready, Set, Oscillate! The Fastest Way to Change Arduino Pins²³⁶
-

Arduino má řadu nožiček označených d2-d13. Jedná se o digitální vstupy a výstupy. Pomocí těchto nožiček můžeme číst digitální informaci z okolí, například je-li stisknuto tlačítko. A naopak je můžeme použít jako výstupní a například rozsvítit LED diodu. Abychom věděli jak funkce použít, musíme mít alespoň základní představu jak digitální vstupy a výstupy v mikrořadiči fungují.

Nejdříve potřebujeme nakonfigurovat pin a oznámit hardwéru jestli jej chceme mít jako vstupní nebo výstupní. To uděláme pomocí volání funkce `pinMode`. Například když budeme chtít blikat se zabudovanou LED diodou v Boarduinu, která je připojená na d13, nastavíme si pin 13 jako výstupní.

```
int ledPin = 13;

void setup(void)
{
    pinMode(ledPin, OUTPUT);
}
```

```
}
```

V této chvíli již můžeme s LED diodou svítit.

```
digitalWrite(ledPin, HIGH);
```

A nyní si připojme tlačítko ke vstupu d12.

```
int tlacitkoPin = 12;
```

A ukážeme si malý program, který čte tlačítko a podle toho co přečte tak rozsvítí LED diodu.

```
int ledPin = 13; // Interní LED dioda
int tlacitkoPin = 2;

void setup(void) {
    pinMode(ledPin, OUTPUT);
    pinMode(tlacitkoPin, INPUT);
}

void loop(void) {
    digitalWrite(ledPin, digitalRead(tlacitkoPin));
    delay(100);
}
```

Ačkoliv máme k dispozici nej digitální piny 2-13, můžeme stejným způsobem použít i analogové piny a0-a5. Ty jsou číslovány v řadě digitálních pinů jako d14-d19. Pokud používáte jinou variantu Arduina, například Sanguino, podívejte se do dokumentace jak a které piny můžete použít. V následujícím ukážce zablikám LED diodou připojenou na a5.

```
int ledPin = 19;
void setup(void) {
    pinMode(ledPin, OUTPUT);
}

void loop(void) {
    digitalWrite(ledPin, !digitalRead(ledPin));
    delay(500);
}
```

40.8.2.2. Zvuk

Přímo v základní knihovně, určitě to má Arduino 18, jsou funkce `tone`, a `noTone`. První z funkcí generuje tón o stanoveném kmitočtu na daném pinu. Délka tónu je buďto předána jako třetí parametr `tone`, nebo zní tón pořád dokud není vypnut funkcí `noTone`. Pomocí těchto funkcí můžeme generovat najednou jen jeden tón.

40.8.2.3. Přerušení

Odkazy:

- [Arduino Interrupts²³⁷](#) na μ C Hobby
- [AVR data sheets²³⁸](#)
- `<avr/interrupt.h>`: [Interrupts²³⁹](#)
-

V procesorech Atmel jenž jsou srdcem Arduina máme řadu dedikovaných hardwarových bloků které plní specializované úlohy. Jsou to například sériový komunikační modul (USART), modul I²C sběrnice, modul SPI, čítače/časovače. Všechny tyto bloky mohou vyvolávat různá přerušení. Nejen tyto bloky, ale přerušení může být vyvoláno i změnami digitálních vstupů. Celý systém přerušení existuje proto, aby jednotlivé části hardware mohli "přítáhnout pozornost" procesoru a býti přednostně a rychle obslouženy. Například hlavní program provádí dlouhý výpočet, a jednotka časovače vyvolá přerušení. Procesor zanechá vykonávání hlavního výpočtu a provede kód obsluhy přerušení časovače. Poté se procesor vrátí k hlavnímu výpočtu a pokračuje v práci kde byl přerušen.

V kódu zapisujeme proceduru/funkci obsluhy přerušení podle následující předlohy. Jako parametr makra ISR uvedeme název vektoru přerušení, který chceme obsloužit.

```
ISR(přerušovací_vektor) {
    kód obsluhy přerušení
}
```

Varování

Jeden vektor přerušení může ukazovat jen na jednu funkci/proceduru obsluhy přerušení.

```
#include <avr/interrupt.h>
byte v, dv;

ISR(TIMER2_OVF_vect) {
    TCNT2 = dv;
    PORTD = v;
}
```

V MCU ATmega328 a ATmega168 jsou tři čítače/časovače. Její propojení s piny a možnosti jsou následující:

Timer0

Systémový časovač, PWM na pinech d5 a d6

Tento čítač/časovač se používá k sledování času. Používá jej například funkce millis. Rovněž je tento čítač/časovač použit ke generování PWM signálu na pinech d5 a d6.

Timer1

Generuje PWM na pinech d9 a d10

Timer2

Generuje PWM na pinech d3 a d11

Ačkoli jsou užity všechny čítače/časovače, jen Timer0 má k přerušení přiřazenu přerušovací rutinu.

Použití čítače/časovače Timer2

```
#define TIMER_CLOCK_FREQ 2000000.0    // 16MHZ / 8

unsigned char SetupTimer2(float timeoutFrequency) {
    unsigned char result;

    result = (int)((257.0-(TIMER_CLOCK_FREQ/timeoutFrequency))+0.5);

    TCCR2A = 0;
    TCCR2B = 0<<CS22 | 1 <<CS21 | 0<<CS20;
```

```

    TIMSK2 = 1<<TOIE2;    // Timer2 Overflow Interrupt Enable
    TCNT2 = result;        // load the timer fot its first cycle

    return(result);
}

```

Zápis procedury volané přerušením od časovače.

```

#define TOGGLE_IO 9

ISR(TIMER2_OVF_vect) {
    digitalWrite(TOGGLE_IO, !digitalRead(TOGGLE_IO));

    latency = TCNT2;
    TCNT2 = latency + timerLoadValue;
}

```

Ukázka kódu z gonium.net

* <http://gonium.net/md/2006/12/23/arduino-timer-interrupt/>

```

#include <avr/interrupt.h>
#include <avr/io.h>

#define INIT_TIMER_COUNT 6
#define RESET_TIMER2 TCNT2 = INIT_TIMER_COUNT

int ledPin = 13;
int int_counter = 0;
volatile int second = 0;
int oldSecond = 0;
long starttime = 0;

// Aruino runs at 16 Mhz, so we have 1000 Overflows per second...
// 1/ ((16000000 / 64) / 256) = 1 / 1000
ISR(TIMER2_OVF_vect) {
    RESET_TIMER2;
    int_counter += 1;
    if (int_counter == 1000) {
        second+=1; int_counter = 0;
    }
};

void setup() {
    Serial.begin(9600);
    Serial.println("Initializing timerinterrupt");
    //Timer2 Settings: Timer Prescaler /64,
    TCCR2A |= (1<<CSS22);
    TCCR2A &= ~((1<<CS21) | (1<<CS20));
    // Use normal mode
    TCCR2A &= ~((1<<WGM21) | (1<<WGM20));
    // Use internal clock - external clock not used in Arduino
    ASSR |= (0 << AS2);
    //Timer2 Overflow Interrupt Enable
    TIMSK2 |= (1<<TOIE2) | (0 << OCIE2A);

    sei();
}

```



```

//Timer2 Overflow Interrupr Enable
TIMSK2 =
Serial.print(millis() - starttime);
Serial.println(".");
digitalWrite(ledPin, HIGH);
delay(100);
digitalWrite(ledPin, LOW);
oldSecond = second;
}
}

```

40.8.2.4. Čítače/časovače

- * V této sekci shromažd'uji informace a podklady pro článek o použití časovačů k přerušení a měření času.
- * Z projektu "clock" DCF77.

```

// SETUP
//Timer2 Settings: Timer Prescaler /64.
TCCR2B |= (1<<CSS22); // turn on CSS22 bit
TCCR2B &= ~(1<<CS21 | 1<<CS20); // turn off CS21 and CS20 bits
// Use normal mode
TCCR2A &= ~(1<<WMG21 | 1<<WMG20); // turn off WMG21 and WMG20 bits
TCCR2B &= ~(1<<WMG22); // turn off WMG22
// Use internal clock - external clock not used in Arduino
ASSR |= (0<<AS2);
TIMSK2 |= (1<<TOIE2) | (0<<OCIE2A); // Timer2 Overflow Interrupt Enable

RESET_TIMER2;

attachInterrupt(0, int0handler, CHANGE);

```

Kód obsluhy přerušení od přetečeného časovače TIMER2

```

/**
 * The interrupt routine for counting seconds - increment hh:mm:ss.
 */
ISR(TIMER2_OVF_vect) {
    RESET_TIMER2;
    tick_counter += 1;
    if (tick_counter == 1000) {
        ss++;
        if (ss==60) {
            ss=0;
            mm++;
            if (mm==60) {
                mm=0;
                hh++;
                if (hh==24) {
                    hh=0;
                }
            }
        }
        tick_counter = 0;
    }
}

```

Chceme-li generovat přerušení v pravidelných intervalech a toto přerušení obsluhovat vlastním kódem, můžeme vyjít z následující ukázky.

* Ukázka je ve stádiu psaní a ladění!

```
void setup() {
    ...
    /*
     * Setup Timer2 to interrupt mode with no PWM.
     */
    noInterrupts();
    // Use internal clock from clkI/O. Must be set before TCNT2,
    // OCR2A, OCR2B, TCCR2A and TCCR2B, because of possible
    // corruption of these! Better to write in sequence and not
    // in one command because of datasheet!
    ASSR &= ~(1<<EXCLK);    // EXCLK=0
    ASSR &= ~(1<<AS2);      // AS2=0
    // Clock Select = clkT2S/1024 (From prescaler) [CS2 2:1:0 = 111]
    TCCR2B |= 0b00000111;    // clkT2S is 16MHz/1024 = 15625Hz
    // Waveform Generation = CTC [WGM2 2:1:0 = 010]
    TCCR2A &= 0b1111010; TCCR2A |= 0b00000010
    TCCR2B &= 0b11110111;
    // Compare Match Output A and B Mode = Normal [COM2A1:0 COM2B1:0 = 0000]
    TCCR2A &= 0b00001111;
    // Set TOP
    OCR2A = 38; // TOP value for mode CTC. Should generate 400Hz
    // Allow interrupt only from Timer2 Compare A Match. Overflow
    // does not work as I expected initially.
    // OCIE2B=0, OCIE2A=1, TOIE2=0
    TIMSK2 = 0b00000010;
    // Clear the Timer/Counter2 Interrupt Flags
    TIFR2 = 0b00000000;
    interrupts();
    ...
}
```

Obsluha přerušení

```
ISR(TIMER2_COMPA_vect, ISR_NOBLOCK) {
    ... udělej co musíš ale snaž se co nejrychleji
}
```

40.8.2.5. Typy hodnot

*

Tabulka 40-30.

typ	bytes	rozsah	užití
int	2	-32768 až 32767	
unsigned int	2	0 až 65535	kladná čísla
long	4	-2147483648 až 2147483647	velmi velké čísla
unsigned long	4	0 až 4294967295	velmi velké kladné celé čísla

typ	bytes	rozsah	užití
float	4	3.4028235E+38 až -3.4028235E+38	čísla s pohyblivou řádovou čárkou
double	4	stejně jako float	na Arduino je double jen jiné jméno pro float
boolean	1	false (0) nebo true (1)	reprezentuje logické hodnoty
char	1	-128 až 127	reprezentuje znak, také reprezentuje číslo se znaménkem.
byte	1	0 až 255	podobně jako char ale pro hodnoty bez znaménka
string			řetězec znaků char
void			

40.8.2.6. Sériová UART komunikace

*

Na Arduino a jeho odvozeninách s malými (co do počtu vývodů) procesory je jen jedno sériové rozhraní. Na Arduino Mega jsou další tři.

Tabulka 40-31.

port	TX pin	RX pin
Serial	1	0
Serial1	18	19
Serial2	16	17
Serial3	14	15

```
void setup() {
    Serial.begin(9600);
}

int number = 0;

void loop() {
    Serial.print("");
    Serial.println(number);
    number++;
    delay(500);
}
```

Na straně počítače můžeme v Linuxu použít tyto programy:

- CuteCom²⁴⁰
- The moserial Project²⁴¹
- Putty²⁴²
- screen²⁴³
- minicom²⁴⁴
- gkterm²⁴⁵
-

```
Serial.println(chrValue, BYTE);
```

```
Serial.println(chrValue, DEC);  
Serial.println(chrValue, HEX);  
Serial.println(chrValue, OCT);  
Serial.println(chrValue, BIN);
```

40.8.3. Jak nahrát program/sketch

Odkazy:

- Burning sketches to the Arduino board with an external programmer²⁴⁶
-

Standardním způsobem jak dostat sketch do arduina je použít USB/Serial připojení které využívá bootloaderu nahraného v čipu.

Pokud chceme nahrávat sketche pomocí ISP konektoru a programátoru, musíme si upravit konfiguraci. To znamená ručně editovat soubor `~/.arduino/preferences.txt`. V tomto souboru vyhledáme řádek s volbou `upload.using` a přepíšeme hodnotu podle toho jaký programátor máme. Seznam podporovaných programátorů je v souboru `hardware/arduino/programmers.txt` v distribuci IDE Arduino. V mém případě, kdy jsem si postavil programátor na základě arduina jsem tam uvedl:

```
upload.using=arduinoisp
```

Pokud se budeme chtít opět vrátit k sériovému bootloaderu, vrátíme volbu `upload.using` na původní hodnotu.

```
upload.using=bootloader
```

Musíme také nahrát do čipu bootloader!

40.8.4. Jak to funguje

```
int main() {  
    init();  
    setup();  
    for (;;)   
        loop();  
    return 0;  
}
```

Funkce `setup` a `loop` definujeme sami.

40.8.5. Standardní funkce

Odkazy:

- Language Reference²⁴⁷
-

Tato část obsahuje popis některých funkcí.

analogRead

Jméno

`analogRead` — Čte analogovou hodnotu z analogového pinu

Přehled

```
void analogRead(int pin);
```

pin

Číslo analogového pinu ze kterého se čte napětí. Na většině konstrukcí je to 0-5, na Arduino Mini a Nano je to 0-7 a na Mega je to 0-15.

Popis

Funkce čte napětí na analogovém pinu. Napětí v intervalu 0 až referenční napětí (vybrané voláním `analogReference`) se převádí na číslo v rozsahu 0 až 1023. Doba převodu trvá asi 100 μ s, a maximální počet čtení za sekundu je asi 10 000.

Příklad

* *Tento příklad je opsaný, zatím jsem jej nezkoušel.*

```
int analogPin = 3;
int val = 0;

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    val = analogRead(analogPin);
    Serial.println(val);
}
```

Odkazy

Související informace: `analogReference`, `analogWrite`, `map`

- `analogRead()`¹
- Analog Input Pins²

Poznámky

1. <http://arduino.cc/en/Reference/AnalogRead>
2. <http://arduino.cc/en/Tutorial/AnalogInputPins>

analogReference

Jméno

`analogReference` — Vybírá referenční napětí pro A/D převodník

Přehled

```
void analogReference( type );
```

Funkce nastavuje vybrané referenční napětí pro potřeby A/D převodníku. Možné hodnoty jsou

- `DEFAULT` — jako reference je použito napájecí napětí (5 nebo 3,3V podle konstrukce)
- `INTERNAL` — je použit interní generátor referenčního napětí (1,1V na ATmega168/328, 2,56V na ATmega8)
- `INTERNAL1V1` — vestavěný zdroj referenčního napětí 1.1V (pouze na Arduino Mega)
- `INTERNAL2V56` — vestavěný zdroj referenčního napětí 2.56V (pouze na Arduino Mega)
- `EXTERNAL` — je použito externí referenční napětí, přivedené na pin AREF

Popis

Varování

Pokud máme zapojen externí zdroj referenčního napětí, musíme to oznámit Arduinou voláním `analogReference(EXTERNAL)` předtím než provedeme `analogRead`! Jestli to neuděláme, uvedeme vnitřní zdroj referenčního napětí do skratu z externím zdrojem.

Odkazy

Související informace: `analogRead`, `analogWrite`

- `analogReference()`¹
- Analog Input Pins²

Poznámky

1. <http://arduino.cc/en/Reference/AnalogReference>
2. <http://arduino.cc/en/Tutorial/AnalogInputPins>

analogWrite

Jméno

`analogWrite` — Write an analog value (PWM wave) to a pin

Přehled

```
void analogWrite(int pin, int value);
```

Posílá na pin analogový signál ve formě PWM. Hodnota *value* je v rozsahu 0 (vypnuto) až 255 (plně zapnuto).

Parametr *pin* smí nabývat na Arduinu jen hodnot 9, 10 a 11. Souvisí to s tím, že jen k těmto pinům jsou připojeny obvody generování PWM signálu. Na jiných mikropočítačích ATmega to mohou být jiné piny, a může jich být také jiný počet.

Popis

Pomocí `analogWrite` můžeme Arduinem generovat analogový signál. Mikropočítač ATmega použitý v Arduinu nemá analogový výstup. Analogový signál se vytváří integrováním PWM signálu na kondenzátoru. Funkce `analogWrite` zajišťuje tedy generování PWM signálu o stanovené střídě.

Protože se PWM signál generuje pomocí vnitřních obvodů čítačů/časovačů, můžeme použít jen takové piny, na které je signál OCn mikropočítače vyveden.

PWM signál je generovaný na kmitočtu cca 490Hz.

PWM signál generovaný na pinech 5 a 6 je v konfliktu s funkcemi `millis()` a `delay`.

Tabulka 40-1.

čip	PWM piny
ATmega8	d9,d10,d11
ATmega168	3, 5, 6, d9,d10,d11
ATmega328	3, 5, 6, d9,d10,d11
Arduino Mega	2 — 13

```
int pin = 11;    // LED
int pulseWidth = 127;    // Libovolná hodnota mezi 0 až 255

void setup()
```

```
{  
    /* pro analogWrite není potřeba žádná inicializace */  
}  
  
void loop()  
{  
    analogWrite(pin, pulseWidth);  
}
```

Odkazy

Související informace: `digitalWrite`, `map`, `pinMode`

- `analogWrite()`¹
- Arduino Pulse Width Modulation²

Poznámky

1. <http://arduino.cc/en/Reference/AnalogWrite>
2. <http://principialabs.com/arduino-pulse-width-modulation/>

attachInterrupt

Jméno

`attachInterrupt` — Aktivuje funkci obsluhy přerušení na externí přerušení

Přehled

```
void attachInterrupt(int pin, function, int mode);
```

mode

konfiguruje zdroj přerušení. Máme čtyři možné hodnoty:

- `LOW` způsobí přerušení když je na pinu logická 0
- `CHANGE` způsobí přerušení kdykoliv se na pinu změní logická hodnota
- `RISING` způsobí přerušení jen při změně logické hodnoty z 0 na 1
- `FALLING` způsobí přerušení jen při změně logické hodnoty z 1 na 0

Popis

Příklad

* Tento příklad je opsaný, zatím jsem jej nezkoušel.

```
int pin = 13;
volatile int state = LOW;

void setup(void)
{
    pinMode(pin, OUTPUT);
    attachInterrupt(0, blink, CHANGE);
}

void loop(void)
{
    digitalWrite(pin, state);
}

void blink(void)
{
    state = !state;
}
```

Odkazy

Související informace: `Arduino.detachInterrupt`, `interrupts`, `noInterrupts`

- `attachInterrupt()`¹

Poznámky

1. <http://arduino.cc/en/Reference/AttachInterrupt>

constrain

Jméno

`constrain` — omezení hodnoty

Přehled

```
long constrain(long value, long low, long high);
```

- *value* — hodnota k omezení
- *low, high* — dolní a horní hodnota intervalu

Popis

Funkce omezuje číslo do zadaného intervalu.

Příklad

Tento příklad ukazuje použití funkce `constrain`.

*

```
void loop() {  
    int val = analogRead(0);  
    val = constrain(val, 10, 150);  
    ...  
}
```

Odkazy

Související informace: `max`, `Arduino.ref.min`;

- `constrain()`¹

Poznámky

1. <http://arduino.cc/en/Reference/Constrain>

delay

Jméno

`delay` — Spozdí vykonávání programu a zadaný počet milisekund

Přehled

```
void delay(int milliseconds);
```

Parametr `delay` určuje dobu čekání v milisekundách.

Popis

```
void loop()
{
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
}
```

Odkazy

Související informace: `Arduino.noDelay`;

- `delay()`¹

Poznámky

1. <http://arduino.cc/en/Reference/Delay>

detachInterrupt

Jméno

`detachInterrupt` — Odpojí funkcy obsluhy přerušení

Přehled

```
void detachInterrupt(int interrupt);
```

`interrupt`

číslo přerušení (0 nebo 1).

Popis

Příklad

Odkazy

Související informace: `attachInterrupt`, `interrupts`, `noInterrupts`

- `detachInterrupt()`¹

Poznámky

1. <http://arduino.cc/en/Reference/DetachInterrupt>

digitalRead

Jméno

`digitalRead` — Reads value from specified digital pin

Přehled

```
int digitalRead( pin );
```

*

Popis

```
void loop()  
{  
    button = digitalRead(12);  
    :  
}
```

Odkazy:

Související informace: `digitalWrite`, `pinMode`

- 40.8.2.1
- `digitalRead`¹

Poznámky

1. <http://arduino.cc/en/Reference/DigitalRead>

digitalWrite

Jméno

`digitalWrite` — Write digital value to pin

Přehled

```
void digitalWrite(pin, value);
```

Funkce nastavuje digitální pin na logickou hodnotu.

Popis

```
int ledPin = 11;

void setup()
{
    pinMode(ledPin, OUTPUT);
}

void loop()
{
    digitalWrite(ledPin, HIGH);
    digitalWrite(ledPin, LOW);
}
```

Odkazy

Související informace: `analogWrite`, `digitalRead`, `pinMode`

- `digitalWrite()`¹

Poznámky

1. <http://arduino.cc/en/Reference/DigitalWrite>

interrupts

Jméno

`interrupts` — Znovu povolí přerušení

Přehled

```
void interrupts(void);
```

Popis

Příklad

* *Tento příklad je opsaný, zatím jsem jej nezkoušel.*

```
void setup() {}

void loop()
{
    noInterrupts();
    interrupts();
}
```

Odkazy

Související informace: `attachInterrupt`, `Arduino.detachInterrupt`, `noInterrupts`

- `interrupts()`¹

Poznámky

1. <http://arduino.cc/en/Reference/Interrupts>

map

Jméno

`map` — mapování hodnoty z jednoho rozsahu do jiného

Přehled

```
long map(long value, long fromLow, long fromHigh, long toLow, long toHigh);
```

- *value* — hodnota k převedení
- *fromLow, fromHigh* — dolní a horní hodnota intervalu z kterého se převádí
- *toLow, toHigh* — dolní a horní hodnota intervalu do kterého se převádí

Popis

Funkce mapuje číslo z jednoho rozsahu na druhý. Ppkud potřebujeme převést hodnotu v jedné soustavě do jiné soustavy, musíme si napsat příslušné rovnice a z nich odvodit převodní funkci. Pro usnadnění tohoto úkolu máme k dispozici funkci `map`.

Příklad

Tento příklad ukazuje použití funkce `map` při převodu analogové hodnoty. Čtená hodnota je v rozsahu $\langle 0, 1023 \rangle$ a potřebujeme ji převést (namapovat) na rozsah $\langle 0, 255 \rangle$.

*

```
void setup() {
}

void loop() {
    int val = analogRead(0);
    val = map(val, 0, 1023, 0, 255);
    analogWrite(9, val);
}
```

Zajímavou možností je otočit intervaly. Tím myslím zápis:

```
int val = analogRead(0);
val = map(val, 0, 1023, 255, 0);
analogWrite(9, val);
```

Tedy nejnižší hodnou ze vstupního intervalu mapujeme na nejvyšší hodnotu výstupního intervalu. To znamená že když se vstupní hodnota zvětšuje, výstupní hodnota klesá.

Dodatek

Pro zajímavost, funkce `map` je definována takto

```
long map(long x, long in_min, long in_max, long out_min, long out_max) {
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

Odkazy

Související informace: `Arduino.constrain`;

- `map()`¹

Poznámky

1. <http://arduino.cc/en/Reference/Map>

max

Jméno

`max` — výběr největšího ze dvou čísel

Přehled

```
long max(long x, long y);
```

Popis

Toto volání vybírá největší ze dvou čísel. Obvykle se volání `max` implementuje jako makro. Z toho vyplývá, že nevíme v jakém pořadí a kolikrát budou jeho argumenty vyhodnocovány.

Příklad

```
val = max(value, 100);
```

Odkazy

Související informace: `constrain`, `min`

- `max()`¹

Poznámky

1. <http://arduino.cc/en/Reference/Max>

millis

Jméno

`millis` — Počet milisekund od posledního restartu

Přehled

```
unsigned long millis(void);
```

Funkce nemá žádný parametr. Jako návratovou hodnotu vrací počet milisekund který uplynul od posledního restartu.

Popis

Příklad

*

Odkazy

Související informace: `interrupts`, `noInterrupts`

- `attachInterrupt()`¹

Poznámky

1. <http://arduino.cc/en/Reference/AttachInterrupt>

min

Jméno

`min` — výběr nejmenšího ze dvou čísel

Přehled

```
long min(long x, long y);
```

Popis

Toto volání vybírá nejmenší ze dvou čísel. Obvykle se volání `min` implementuje jako makro. Z toho vyplývá, že nevíme v jakém pořadí a kolikrát budou jeho argumenty vyhodnocovány.

Příklad

```
val = min(value, 100);
```

Odkazy

Související informace: `constrain`, `max`

- `min()`¹

Poznámky

1. <http://arduino.cc/en/Reference/Min>

noInterrupts

Jméno

`noInterrupts` — Zakáže přerušení

Přehled

```
void noInterrupts(void);
```

Popis

Příklad

* Tento příklad je opsaný, zatím jsem jej nezkoušel.

```
void setup() {}

void loop()
{
    noInterrupts();
    Interrupts();
}
```

Odkazy

Související informace: `attachInterrupt`, `Arduino.detachInterrupt`, `interrupts`

- `noInterrupts()`¹

Poznámky

1. <http://arduino.cc/en/Reference/NoInterrupts>

noTone

Jméno

`noTone` — Zastaví generování zvuku na pinu

Přehled

```
void noTone(int pin);
```

pin

Parametr *pin* je vývod na kterém je generován tón.

Popis

Odkazy

Související informace: `tone`

- `noTone()`¹

Poznámky

1. <http://arduino.cc/en/Reference/NoTone>

pinMode

Jméno

`pinMode` — Konfiguruje zadaný pin buďto jako vstupní nebo výstupní

Přehled

```
void pinMode(pin, mode);
```

Parametr *pin* určuje vývod který konfigurujeme. Parametr *mode* může být `INPUT`, když chceme pin jako vstupní, nebo `OUTPUT`, když jej chceme nakonfigurovat jako výstupní.

Popis

Funkce konfiguruje pin jako vstupní nebo výstupní. Jako vstupní nebo výstupní můžeme konfigurovat všechny digitální piny a většinu analogových. Na běžném Arduinu všechny analogové.

Poznámka: Při zapnutí Arduina jsou všechny piny standardně nastaveny jako vstupní..

```
int ledPin = 11;
int buttonPin = 10;

void setup(void)
{
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}

void loop(void)
{
    digitalWrite(ledPin, digitalRead(buttonPin));
}
```

Odkazy

Související informace: `digitalRead`, `digitalWrite`

- `pinMode()`¹
- Digital Pins²
- Analog Input Pins³

Poznámky

1. <http://arduino.cc/en/Reference/PinMode>
2. <http://arduino.cc/en/Tutorial/DigitalPins>
3. <http://arduino.cc/en/Tutorial/AnalogInputPins>

tone

Jméno

`tone` — Generates a square wave of the specified frequency (and 50% duty cycle) on a pin

Přehled

```
void tone(pin, frequency);
```

```
void tone(pin, frequency, duration);
```

Parametr *pin* určuje vývod na kterém bude generován tón. Parametr *frequency* pak kmitočet tónu v Hz. Pokud je použit i třetí parametr *duration*, tak udává dobu trvání tónu v milisekundách.

Popis

Funkce generuje tón jenž má obdélníkový průběh a střihu 50%. Pokud použijeme variantu bez parametru *duration*, funkce nastaví hardware pro generování tónu a hned se vrátí. Tón zní, dokud jej nezměníme dalším voláním `tone` na ten samý pin, nebo dokud nezavoláme funkci `noTone`. Pokud použijeme variantu volání se třemi parametry, přehrává se tón tak dlouho jak jsme nastavili ve třetím parametru *duration*.

Varování

Použití funkce `tone` je v konfliktu s použitím PWM na pinech 3 a 11.

Odkazy

Související informace: `Arduino.noTone`;

- `tone()`¹

Poznámky

1. <http://arduino.cc/en/Reference/Tone>

40.8.6. Knihovny

40.8.6.1. Servo

Odkazy:

- Arduino Examples → Servo Library²⁷¹
- Open source software from TinkerLondon.com and Tinker.it²⁷² (TrueRandom, Cantarino, DmxSimple, Auduino, Liquidware Touchshield demo, Firmata for the Ethernet Shield)
-
-

Knihovna pro práci se servomotory

```
// by BARRAGAN <http://barraganstudio.nom>
#include <Servo.h>
Servo myservo;           // create servo object
int pos = 0;             // servo position

void setup() {
    myservo.attach(9);    // attaches the servo on pin 9
}

void loop() {
    for (pos = 0; pos < 180; pos += 1) {
        myservo.write(pos);
        delay(15);
    }
    for (pos = 180; pos >= 1; pos -= 1) {
        myservo.write(pos);
        delay(15);
    }
}
```

Knihovna má 6 funkcí, které dále popíši.

attach

Jméno

attach —

Přehled

```
void attach(int pin);
void attach(int pin, int min, int max);
```

Funkce připojí objekt servo k zadanému digitálnímu pinu *pin*. Parametry *min* a *max* určují šířku pulsu pro 0 stupňové natočení (*min*=544) a pro 180 stupňové natočení (*max*=2400).

Popis

read

Jméno

read — Read the current angle of the servo.

Přehled

```
int read();
```

Vrátí úhel natočení serva. Tedy poslední hodnotu zapsanou příkazem `write`.

Popis

Odkazy:

Související metody:

- `Servo.read()`¹

Poznámky

1. <http://arduino.cc/en/Reference/ServoRead>

write

Jméno

`write` — Position servo at angle.

Servo

Přehled

```
void write(int angle);
```

Nastaví servo do úhlu *angle*. Úhle musí být v intervalu od 0 do 180, parametr je ve stupních.

Popis

Odkazy:

Související metody: `Servo.read()`

- `Servo.write()`¹

Poznámky

1. <http://arduino.cc/en/Reference/ServoWrite>

writeMicroseconds

Jméno

`writeMicroseconds` — Writes a value in microseconds to the servo.

Servo

Přehled

```
void writeMicroseconds(int microseconds);
```


Na servo je poslán impuls dané délky *microseconds*. Délka impulsu určuje natočení serva. Pro standardní serva platí že 1000 μ s je natočení nejvíce v levo, a 2000 μ s je natočení nejvíce v pravo.

Popis

Odkazy:

Související metody: write()

- Servo.writeMicroseconds()¹

Poznámky

1. <http://arduino.cc/en/Reference/ServoWriteMicroseconds>

40.8.6.2. Bitlash

* *Attributy: id="Bitlash"*

Odkazy:

- Bitlash Documentation Index²⁷⁶
-

Něco málo o velmi zajímavé a užitečné knihovně Bitlash a o jejím použití. Co to Bitlash je? Je to knihovna která v mikropočítači realizuje shell. Tedy takovou příkazovou řádku, ze které můžeme manipulovat s hardware mikropočítače, zadávat jednoduché skripty a v rámci paralelizmu i tyto "na pozadí" spouštět. Tedy věci do té doby nevídané.

Jak knihovnu použijeme? V principu na začátku sketche napíšeme potřebný `#include` a pak použijeme dvě volání. V rámci inicializace voláme `initBitlash` kterému jako parametr dáme komunikační rychlost sériového portu. Ve smyčce programu pak voláme `runBitlash`. Minimální kód tedy vypadá například takto.

```
#include "bitlash.h"

void setup(void) {
    initBitlash(57600);
}

void loop(void) {
    runBitlash();
}
```

40.8.6.3. HC595

*

Odkazy:

- Home of the hc595²⁷⁷ on the github — má vlastní knihovna

- Using shift registers to extend the microcontroller I/O²⁷⁸ — velmi podrobný článek popusující použití HC595 (výstup) a HC165 (vstup)

Tato knihovna řeší připojení jednoho (právě jednoho) čipu 595 k mikropočítači.

40.8.6.4. Arduino Tone Library

*

Odkazy:

- Arduino Tone Library²⁷⁹
-

Knihovna která dokáže na jednu generovat několik tónů, podle množství dostupných časovačů v mikropočítači. Její zjednodušená verze schopná hrát najednou jen jeden tón je přímo součástí vývojového prostředí arduina.

40.8.6.5. Serial

* *Attributy:* `id="Arduino.lib.Serial"`

Knihovna / třída `Serial` je standardní třídou jenž je přímo k dispozici, bez nutnosti importovat nějaký hlavní soubor.

Na Arduinu máme k dispozici nejméně jeden sériový port s hardwarovou podporou. Jako periferní zařízení je v datasheetu mikrořadiče označen kódem UART nebo USART. Je vyveden na piny d0 (RX, vysílání) a d1 (TX, příjem).

Přes tento, první, sériový port se arduino programuje. Pro potřeby ladění je na něm implementován sériový monitor. A můžeme jej použít ke komunikaci naší hotové aplikace s jiným počítačem či zařízením.

Jiné modely/varianty Arduina, ty s většími procesory, mohou mít další sériové porty. Tyto jsou zastoupeny objekty `Serial1`, `Serial2` a `Serial3`.

* *Zbývá popsat metody:*

- `flush`
- `print`
- `println`
- `write`

available

Jméno

`available` — Zjistí kolik znaku je v přijímacím bufferu

`Serial`

Přehled

```
int available();
```

Vrací počet přijatých znaků, které jsme si zatím nepřečetli.

Popis

Metoda slouží k zjištění, kolik znaků čeká na přečtení.

Příklady užití

*

```
int znak = 0;

void setup() {
    Serial.begin(9600);
}

void loop() {
    if(Serial.available() > 0) {
        znak = Serial.read();

        Serial.print("Prijal jsem: ");
        Serial.println(znak, DEC);
    }
}
```

Odkazy

Související informace: `flush`, `read`

- `available()`¹

Poznámky

1. <http://arduino.cc/en/Serial/Available>

begin

Jméno

`begin` — Inicializace komunikačního hardware a nastavení přenosové rychlosti

`Serial`

Přehled

```
void begin(long speed);
```

Parametr *speed* určuje přenosovou/modulační rychlost v baudot. Pro komunikaci s počítačem použijte jednu z hodnot: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 a 115200. Ale můžete nastavit i jinou rychlost.

Popis

Metoda slouží k otevření sériového portu a nastavení komunikační rychlosti.

Příklady užití

*

```
void setup(void) {  
    Serial.begin(9600);    // otevře sériový port a nastaví rychlost 9600 bps  
}
```

Odkazy

Související informace: `end`

- `begin()`¹

Poznámky

1. <http://arduino.cc/en/Serial/Begin>

end

Jméno

`end` — Ukončuje/zakazuje sériovou komunikaci na portu

Serial

Přehled

```
void end();
```

Metoda nemá žádný parametr a ani nevrací žádnou hodnotu.

Popis

Metoda slouží k ukončení sériové komunikace na portu.

Odkazy

Související informace: `begin`

- `end()`¹

Poznámky

1. <http://arduino.cc/en/Serial/End>

flush

Jméno

`flush` — Vyprázdní vstupní buffer

Serial

Přehled

```
void flush();
```

Metoda nemá žádný parametr a ani nevrací žádnou hodnotu.

Popis

Volání `flush` maže/vyprazdňuje vstupní buffer. Všechny znaky, které v něm byly a čekaly na přečtení, jsou nenávratně ztraceny.

Odkazy

Související informace: `available`, `read`

- `flush()`¹

Poznámky

1. <http://arduino.cc/en/Serial/Flush>

print

Jméno

`print` — Vypíše informace v textové podobě na sériový port

Serial

Přehled

```
void print(data);  
void print(data, format);
```

Metoda má jeden až dva parametry.

data

Hodnota která se vypisuje. Může to být celočíselná hodnota, reálné číslo, nebo řetězec. Funkce sama rozpozná co jí předáváme. V případě čísla můžeme, ale nemusíme, přidat formátovací parametr.

format

Určuje jakým způsobem se číslo bude formátovat. Pro celá čísla můžeme použít formátovací hodnoty: BYTE, BIN, OCT, DEC a HEX. První volba převádí číslo na ascii znak. Zbylé čtyři volby vypisují číslo v binárním, oktalovém, dekadickém a hexadecimálním tvaru.

V případě čísel v plovoucí řádové čárce, float, formátovací parametr specifikuje na kolik desetinných míst se číslo vypisuje. Jako formátovací parametr se tedy předává celé číslo od 0 výše.

Popis

Metoda vypisuje data v ASCII formě do sériového portu. Metoda má jeden nebo dva parametry. První parametr jsou data, druhý, nepovinný parametr, popisuje formát dat.

```
Serial.print('a');    // => "a"  
  
Serial.print(78, BYTE); // => "N"  
Serial.print(78, BIN);  // => "1001110"  
Serial.print(78, OCT);  // => "116"  
Serial.print(78, DEC);  // => "78"  
Serial.print(78, HEX);  // => "4E"  
  
Serial.print(1.23456, 0); // => "1"  
Serial.print(1.23456, 2); // => "1.23"  
Serial.print(1.23456, 4); // => "1.2346" zaokrouhlení!
```

Pro zajímavost, aktuální hodnoty formátovacích konstant v Arduinu verze 0022 jsou:

```
#define BYTE    0  
#define BIN     2  
#define OCT     8  
#define DEC    10  
#define HEX    16
```

Odkazy

Související informace: **FIXME:**Arduino.lib.Serial.write;, **FIXME:**Arduino.lib.Serial.println;, read

- `print()`¹

Poznámky

1. <http://arduino.cc/en/Serial/Print>

println

Jméno

`println` — Vypíše informace v textové podobě na sériový port

Serial

Přehled

```
void println(data);
void println(data, format);
```

Metoda má jeden až dva parametry. Parametry jsou stejné, jako v případě metody `print`.

Popis

Metoda vypisuje data v ASCII formě do sériového portu. Na konci přidá "konec řádku", jako posloupnost znaků ASCII 13 ('\r') a ASCII 10 ('\n').

Odkazy

Související informace: **FIXME:** `Arduino.lib.Serial.write`, `print`, `read`

- `println()`¹

Poznámky

1. <http://arduino.cc/en/Serial/Println>

read

Jméno

`read` — Čtení jednoho znaku ze vstupního bufferu

Serial

Přehled

```
int read();
```

Vrací přijatý znak. Je-li vstupní buffer prázdný, vrací -1.

Popis

Metoda slouží k zjištění, kolik znaků čeká na přečtení.

Příklady užití

*

```
int znak = 0;

void setup() {
    Serial.begin(9600);
}

void loop() {
    if(Serial.available() > 0) {
        znak = Serial.read();

        Serial.print("Prijal jsem: ");
        Serial.println(znak, DEC);
    }
}
```

Odkazy

Související informace: `available`, **FIXME:**`Arduino.lib.Serial.flush`;

- `read()`¹

Poznámky

1. <http://arduino.cc/en/Serial/Read>

write

Jméno

`write` — Odešle binární data na sériový port

Serial

Přehled

```
void write(val);
void write(str);
void write(buf, int len);
```

Metoda má jeden až dva parametry.

`val`

Na sériový port je odeslán jeden byte.

`str`

Na sériový port je odeslán řetězec bajtů.

`buf`

Pole bajtů k odeslání.

`len`

Délka pole `buf`.

Popis

Metoda vypisuje binární data do sériového portu.

Odkazy

Související informace: `print`, `println`, `read`

- `write()`¹

Poznámky

1. <http://arduino.cc/en/Serial/Write>

40.8.7. Různé drobnosti

*

40.8.7.1. Drobnost

*

40.8.8. Programové nástroje

Odkazy:

- Fritzing²⁸⁸
-

40.8.8.1. ADABoot

Odkazy:

- ADABOOT - an ATmega Arduino(tm) compatible bootloader²⁸⁹ na webu The Shoppe at Wulfdon
- Bootloader for Atmega328²⁹⁰
-
-

S bootloaderem z webu AdaIndustries jsem měl problém. Nahrát do MCU šel, po resetu LED blikla, ale nedokázal jsem s ním navázat komunikaci. Nakonec jsem našel zavadeč ADABOOT²⁹¹ se kterým mi vše funguje.

40.8.9. HW doplňky a shieldy

*

Odkazy:

- Arduino Shield List²⁹²
-

Shields:

- Ethernet Shield With PoE²⁹³ (Wiznet W5100, freetronics, AU\$44.95)
- ProtoShield Basic for Arduino²⁹⁴
-
-
-

Komponenty:

- Stackable Arduino Shield Headers²⁹⁵
- shield stacking headers for Arduino²⁹⁶ \$1.5
-

Odkazy:

- Prototino ATmega328 Arduino compatible prototyping board.²⁹⁷
- The Prototino™ is an Arduino clone with a built in prototyping area.²⁹⁸
-
-
-
-

40.8.9.1. EthernetShield ENC28J60

*

Odkazy:

- Library with example code for the nuelectronics Arduino EtherShield (ethernet shield)²⁹⁹ [2009-05-22]
- Ethernet Shield V1.0 for Arduino³⁰⁰

- New Arduino ENC28J60 Ethershield library³⁰¹ [2009-06-22]
- ew EtherShield Library status update³⁰² [2010-08-26]
- Introduction to the tuxgraphics TCP/IP stack, 3rd generation³⁰³
- Embedded stuff³⁰⁴
- Welcome to the Café³⁰⁵
-
-

A HowTo get a etherShield to Work. Pro Arduino 0017

```
$ mkdir ethershield
$ cd ethershield
$ git clone http://github.com/jonoxer/ethershield.git

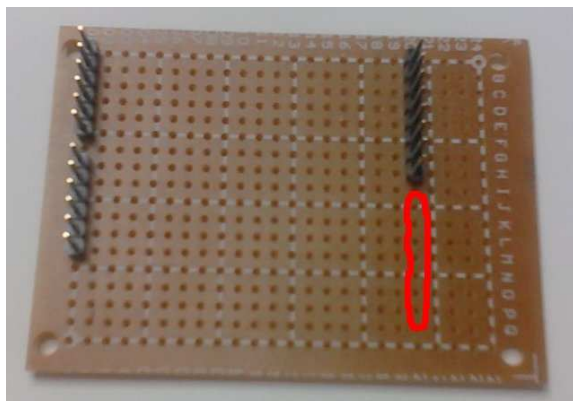
$ cd ethershield
$ git pull
$ cp -r -u -remove-destination . /usr/share/arduino-ide/hardware/libraries/ethershield
```

40.8.9.2. Experimentální PCB

*

Jedním ze způsobů jak experimentovat, je použít experimentální PCB. Tady ovšem narazíme na jeden problém. Vývody D8-D13, GND a AREF na jednom z konektorů arduina jsou posunuté oproti standardní matici 2.54mm. Toto posunutí způsobí, že nelze jednoduše zapájet nožičky do experimentálního PCB a nasunout na Arduino. U všech ostatních kontaktů to lze.

Obrázek 40-25. Arduino experimental PCB



40.8.10. Zapojení a konstrukce s Arduinem

Zde uvádím jak konstrukce nalezené v síti, tak vlastní vylepšení a vlastní konstrukce a nápady. Kód mých projektů se nachází v adresáři `examples/avr/Arduino/`. Některé projekty jsou umístěny na Github.com jako například:

Odkazy:

- Led8x8³⁰⁷
-

40.8.10.1. Arduino AVR ISP programátor

*

Odkazy:

- AVR ISP programming via Arduino³⁰⁸ na Hack a Day [2009-07-15]
- MEGA-ISP Arduino Shield³⁰⁹
- mega-isp In System Programmer using the AVR Mega8³¹⁰ na Google Code
- MEGA-ISP Shield Schematics and Board Layout for Arduino³¹¹
- New MEGA-ISP shield connected³¹²
- Using an Arduino as an AVR ISP (In-System Programmer)³¹³
- Make an Arduino NG into an AVR-ISP programmer (for good)³¹⁴
-
-

Popis zapojení, schéma a plošný spoje je publikován v článku MEGA-ISP Shield Schematics and Board Layout for Arduino³¹⁵. Skeč In System Programmer using the AVR Mega8³¹⁶ pro programátor je k dispozici na Google Code.

```
10: slave reset
11: MOSI
12: MISO
13: SCK
```

```
9: Heartbeat LED (pomrkává když programátor funguje)
8: Error: (červená led) nastala nějaká chyba
7: Programování - právě komunikuje s podřizným obvodem po ICSP
```

A teď pár ukázek použití. Programátor je typu `-c avrisp` a vidíme ho na některém `-P /dev/ttyUSB1` usb portu. V tomto případě na `ttyUSB1` protože `ttyUSB0` je obsazen jiným zařízením. Komunikační rychlost je `-b 19200` baudot. Mám připojený procesor ATmega8-16PU `-p m8`.

```
$ avrdude -pm8 -cavrisp -P /dev/ttyUSB1 -b19200
avrdude: AVR device initialized and ready to accept instructions
Reading | ##### | 100% 0.10s
avrdude: Device signature = 0x1e9307
avrdude: safemode: Fuses OK
avrdude done. Thank you.
```

Funguje. Nyní něco praktičtějšího. Do "prázdných" ATmega328P nahraji bootloader. Mám

```
$
```

40.8.11. Různé verze a varianty Arduina

*

Různé varianty Arduina:

- chibiArduino³¹⁷ [2010-11-06] s bezdrátovým rozhraním FreakLabs Freakduino Chibi Wireless Arduino Compatible Board³¹⁸ \$33
- Arduino Hardware³¹⁹ [2011-03-22]
- Free Duino . ORG³²⁰ — The World Famous Index of Arduino & Freeduino Knowledge
- Arduino Nano 3.0 NO-PIN (ATMEGA328)³²¹ [2011-03-22 \$32.99]
- Bare Bones Board Kit³²² [2011-03-22 \$16.95]
- adfruit industries DC Boarduino³²³ [2011-03-22 \$17.50]

- seed studio Seeeduino V2.21³²⁴ [2011-03-22 \$22.50]
- Teensy USB Development Board³²⁵ [2011-03-22] Teensy 2.0 \$18, Teensy++ 2.0 \$24
- Seeeduino release notice ver 1.0b³²⁶
- Introducing Duino644³²⁷
- Boarduino³²⁸ — Solderless Breadboard Arduino Clone
- Perfboard Hackduino (Arduino-compatible circuit)³²⁹ — minimalistická verze na experimentálním PCB připomínajícím nepájivé kontaktní pole
- Starduino³³⁰ [2010-09-07]
- Paperduino: Arduino-PCB³³¹, Arduino-Perfboard³³², Arduino-Stripboard³³³
- BHASHA Technologies³³⁴ [\$20.65]
- Nanode³³⁵ — ATmega328+ENC28J60 [2011-05-02]

Sanguino:

- Sanguino³³⁶

Podobné konstrukce ne s AVR:

- The Maple³³⁷ (STM32F103RB - ARM)
- ZPUINO³³⁸ 32-bit procesor v FPGA na 100MHz (Zylin's ZPU 32-bit processor core)
-

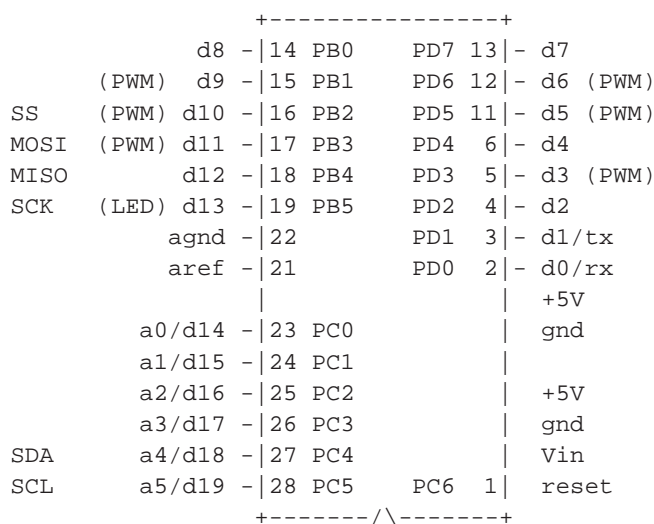
40.8.11.1. Boarduino

Odkazy:

- Bootloader Hacks³³⁹
-

Jednoduché řešení od Lady Ada³⁴⁰ prodávané na Adafruit³⁴¹.

Obrázek 40-26. Zapojení vývodů Boarduina



40.8.12. Kousky kódu a řešení

40.8.12.1. Heartbeat

Odkazy:

- Heartbeat Sketch³⁴² na Arduino.cc
-
-
-

Heart beat je technologie práce s LED diodou, kdy poblikáváním této diody dává zařízení najevo, že je v pořádku.

Můžeme samozřejmě obyčejně blikat led, ale často potřebujeme lepší efekt než pouhé blikání, a navíc toto musíme zaintegrovat do aplikace. Inspirován podobnou funkcí v 40.8.10.1 a ... jsem se rozhodl vyřešit problém trochu jinak. Výchozí myšlenka je že chci aby se jas LED diody měnil podél funkce \sin , případně podle jiné. Další podmínkou je, že si funkce sama musí hlídat čas a nemá tedy používat `delay`. Přišel jsem s řešením, kdy hodnoty funkce jasu je uložena v poli `heartbeat_array`. To mi dovolí si předem připravit libovolnou funkci a v průběhu volání `heartbeat` jinž neztrácím část výpočty v plovoucí řádové čárce. Dalším prvkem řešení je hlídání času. Funkce ji třeba volat rozumě často a ona sama rozhodně jestli již uběhl požadovaný čas `STEP`.

```
#define STEP 50

unsigned long lastTime;
uint8_t heartbeat_array[] = {
#include "sintable.h"
};

void heartbeat(int led) {
    unsigned long now = millis();
    static uint8_t table_index=0;

    /* Check if time is enough, return else */
    if (now < lastTime + STEP) {
        return;
    }

    analogWrite(led, heartbeat_array[table_index++]);
    if (table_index >= sizeof(heartbeat_array)) {
        table_index = 0;
    }
    lastTime += STEP;
}
```

Hodnoty pole `heartbeat_array` vytvářím krátkým programem v Ruby. Jednoduchou úpravou kdy se funkce `sin` zamění za jinou, dosáhneme požadovaného průběhu.

```
#!/usr/bin/env ruby
# This program creates table of output signal for function heart beat.

def generate_table steps, minvalue, maxvalue
    puts "/*
    * Sin x-PI/2 table in #{steps} steps with minumum at #{minvalue} and maximum at #{maxvalue}.
    * Generated with commad:
    * ./generate_table #{steps} #{minvalue} #{maxvalue}
    */"

    puts (0..steps-1) \
        .collect{|deg| (deg.to_f * 2.0 * Math::PI) /steps} \
```

```

        .collect{|r| (Math::sin(r - Math::PI/2)+1)/2 * (maxvalue-minvalue) +minvalue} \
        .collect{|n| "%3d" % n.round} \
        .each_slice(8).collect{|row| "\t" + row.join(', ')} .join "\n"
    end

    if $0 == __FILE__ then
        generate_table ARGV[0].to_i, ARGV[1].to_i, ARGV[2].to_i
    end
end

```

Poznámka: Pokud nebude funkce `heartbeat` volána dostatečně často, vznikne rozdíl v čas `lastTime` a aktuálním čase. To způsobí zrychlené "dohánění" času.

Bylo by hezké, kdyby buďto sama funkce poznala že nebyla dlouhý čas volána a toto ošetřila.

Bylo by hezké kdybych vyřešil náběh LED z nuly při prvním cyklu, případně doběh po posledním cyklu. Klidová poloha by nemusela být nula ale plný jas.

40.9. Processing

*

Odkazy:

- [Processing³⁴³](#) — domovská stránka projektu
- [OpenProcessing³⁴⁴](#)
-
-
-

Poznámky

1. <http://www.atmel.com/products/avr/>
2. http://en.wikipedia.org/wiki/Atmel_AVR
3. <http://www.avrbeginners.net/>
4. <http://www.avrfreaks.net/>
5. http://cs.wikibooks.org/wiki/Programujeme_jedno%C4%8Dipy
6. <http://www.sharksoft.wz.cz/rbglamp.html>
7. <http://www.theiling.de/avr.html>
8. <http://www.nongnu.org/uisp/index.html>
9. <http://www.jiggerjuice.net/electronics/projects/firmware-bootloader.html>
10. <http://avr.fenceline.de/download.html>
11. <http://www.tldp.org/linuxfocus/English/November2004/article352.shtml>
12. <http://www.serasidis.gr/circuits/avrprog/avrprog.htm>
13. <http://www.lancos.com/prog.html>
14. http://gandalf.arubi.uni-kl.de/avr_projects/avrispre/index.html

15. <http://www.evilmadscientist.com/article.php/avrtargetboards>
16. http://elm-chan.org/works/avr/works/avrreport_e.html
17. <http://dybkowski.net/elka/ispprog.html>
18. <http://www.cccac.de/cgi-bin/wiki.pl?action=browse&diff=1&id=DaPa>
19. <http://www.arduino.cc/playground/BootCloner/BootCloner>
20. <http://www.heise.de/ct/Redaktion/cm/klangcomputer/index1.htm>
21. <http://www.instructables.com/files/deriv/FWA17KH/FCHYNWU2/FWA17KH/FCHYNWU2.MEDIUM.jpg>
22. <http://www.linuxfocus.org/English/March2002/article231.shtml>
23. <http://www.linuxos.sk/clanok/242/index.html>
24. http://www.sparkfun.com/commerce/product_info.php?products_id=13
25. <http://www.abclinuxu.cz/blog/vejsplechty/2007/9/trio-avr-isp-tri-zarizeni-zaroven>
26. <http://sharon.esrac.ele.tue.nl/mirrors/z11bpu/micro/ENV/PGM.htm>
27. <http://sharon.esrac.ele.tue.nl/mirrors/z11bpu/micro/CNTR/AVRprog.gif>
28. <http://www.avrprojects.net/articles.php?lng=en&pg=511>
29. <http://www.zen9658.zen.co.uk/avr-doc/HOWTO-get-started.html>
30. <http://www.dutchforce.com/~eforum/index.php?s=5031a7d43400e67a9b4f027558834f38&showtopic=16538>
31. <http://sajiduc.blogspot.com/2008/04/beginners-microcontroller-programming.html>
32. <http://www.ullihome.de/index.php/USB AVR-ISP>
33. <http://www.xs4all.nl/~dicks/avr/usbtiny/>
34. <http://www.equinox-tech.com/Products/details.asp?ID=362>
35. <http://www.nkcelectronics.com/avr-isp-programmer-adapter-kit-10-to-6-.html>
36. <http://www.ere.co.th/default.aspx?RedirectPage=Products&RedirectPage1=ProductsDetail&ProductID=52>
37. <http://www.nkcelectronics.com/stk-to-breadboard-adapter.html>
38. <http://avarice.sourceforge.net/>
39. <http://dlangenberg.googlepages.com/atmega8timers>
40. <http://extremeelectronics.co.in/avr-tutorials/avr-timers-an-introduction/>
41. <http://extremeelectronics.co.in/avr-tutorials/timers-in-compare-mode-part-i/>
42. <http://extremeelectronics.co.in/avr-tutorials/timers-in-compare-mode-part-ii/>
43. <http://www.avrprojects.net/articles.php?lng=en&pg=73>
44. <http://winavr.scienceprog.com/atmel-avr-microcontrollers/analog-to-digital-conversion-in-avr.html>
45. <http://www.webdotdev.com/nvd/content/view/187/>
46. http://www.atmel.com/dyn/resources/prod_documents/doc8127.pdf
47. http://www.atmel.com/dyn/products/product_card_mcu.asp?part_id=4585
48. http://www.atmel.com/dyn/resources/prod_documents/doc8272.pdf
49. <http://www.equinox-tech.com/Products/details.asp?ID=362>
50. <http://www.equinox-tech.com/Products/details.asp?ID=359>
51. <http://www.bsddhome.com/avrdude/>

52. <http://www.captain.at/electronics/atmel-programmer/>
53. <http://www.arduino.cc/en/Hacking/ParallelProgrammer>
54. <http://www.abclinuxu.cz/clanky/hardware/jednocipy-pod-linuxem-i>
55. <http://kartikmohta.com/wiki/Tech/ATmega16Programmer>
56. <http://blackhole.sk/topicavr-atmega8-1>
57. <http://my.opera.com/CrazyTerabyte/blog/2007/10/26/first-contact-with-atmega8-microcontroller-part-2>
58. <http://martin.vancl.eu/AVR-4-Arduino-nahrani-bootloaderu>
59. <http://www.lancos.com/e2p/avrisp-siprog.gif>
60. <http://blackhole.sk/topicavr-atmega8-1>
61. <http://sonicvanajr.com/projs/avrbox.html>
62. <http://extremeelectronics.co.in/avr-tutorials/part-iii-making-programmer-and-target/>
63. <http://www.olimex.com/dev/images/avr-pg1b-sch.gif>
64. http://electronics-diy.com/avr_programmer.php
65. http://electronics-diy.com/avr_programmer.php
66. <http://www.olimex.com/dev/images/avr-pg1b-sch.gif>
67. <http://www.fischl.de/usbasp/>
68. <http://www.scienceprog.com/building-and-installing-usbasp-usb-programmer-of-avr-microcontrollers/>
69. <http://www.scienceprog.com/one-click-project-with-usbasp-programmer/>
70. <http://tuxgraphic.org/electronics/200705/article07052.shtml>
71. <http://code.google.com/p/avropendous/wiki/ISPProgrammingAnAVRopendous>
72. http://www.pjrc.com/hub_isp/
73. <http://www.smrz.chrudim.cz/usbprog/>
74. <http://www.abclinuxu.cz/poradna/linux/show/212720>
75. <http://www.falcom.sk/projects/otherprojects/58-avr-isp-stk500v2.html>
76. http://www.abclinuxu.cz/blog/Boldakuv_blok/2008/5/avr-a-linux
77. http://www.xs4all.nl/~sbolt/e-spider_prog.html
78. <http://avr-asm.tripod.com/>
79. <http://www.tavrasm.org/>
80. http://www.avr-asm-tutorial.net/avr_en/beginner/index.html
81. <https://www.mainframe.cx/~ckueth/avr-c-tutorial/>
82. <http://www.joshknows.com/avr>
83. http://www.nongnu.org/avr-libc/user-manual/group__demo__project.html
84. <http://avra.sourceforge.net/>
85. <http://www.suprafluid.com/avra/>
- 86.
87. http://www.avr-asm-tutorial.net/gavrasm/index_en.html
88. <http://sourceforge.net/projects/avr-ada/>

89. <http://www.adacore.com/home/>
90. <http://avr-ada.wiki.sourceforge.net/BuildScript>
91. <http://sourceforge.net>
92. <http://www2.tech.purdue.edu/ecet/courses/referencematerial/atmel/>
93. <http://www.youtube.com/watch?v=ghC2Q3MQzJw>
94. http://www.nongnu.org/avr-libc/user-manual/group__demo__project.html
95. <http://www.linuxfocus.org/English/March2002/article231.shtml>
96. <http://www.linuxfocus.org/English/November2004/article352.shtml>
97. <http://tuxgraphics.org/electronics/200904/avr-c-programming.shtml>
98. <http://electrons.psychogenic.com/modules/arms/art/3/AVRGCCProgrammingGuide.php>
99. <http://winavr.scienceprog.com/general-avr/atmega-series-of-avr-microcontrollers.html>
100. http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html
101. <http://tech.groups.yahoo.com/group/avrForth/>
102. <http://www.tinyboot.com/avr.html>
103. <http://sourceforge.net/projects/amforth/>
104. <http://www.uelectronics.info/amforth-atmega-forth>
105. <http://amforth.sourceforge.net/>
106. <http://krue.net/avrforth/>
107. <http://www.colorforth.com>
108. <http://www.jwdt.com/~paysan/gforth.html>
109. <http://claymore.engineer.gvsu.edu/~steriana/Software/pfavr/index.html>
110. <http://concurrency.cc/>
111. <http://news.jeelabs.org/2010/02/04/fascinating-concurrency/>
112. <http://side4linux.sourceforge.net/>
113. <http://www.scienceprog.com/kontrollerlab-ide-development-software-for-avr-under-linux/>
114. <http://avrlab.sourceforge.net/>
115. <http://gitorious.org/simavr>
116. <http://compilers.cs.ucla.edu/avrora/cfg.html>
117. http://www.youtube.com/watch?v=e3Ns8_VwZb4
118. <http://www.youtube.com/user/Lefinnois>
119. <http://www.idleloop.com/robotics/ColorLCD/>
120. <http://blog.lukrop.bplaced.net/archives/Nokia-6610-LCD-+-Arduino.html>
121. <http://gnusb.sourceforge.net/hardware/>
122. <http://robotika.cz/wiki/RobotikaWiki>
123. <http://macetech.com/blog/?q=node/46>
124. <http://www.bobhobby.com/2008/04/22/usb-physical-hit-counter-based-on-avr-attiny25/>
125. <http://www.harbaum.org/till/lcd2usb/index.shtml>

126. <http://vamposdecampos.googlepages.com/infracid.html>
127. http://www.raphnet.net/electronique/usbtenki/index_en.php
128. <http://www.schatenseite.de/index.php?id=216&L=2>
129. http://www.harbaum.org/till/i2c_tiny_usb/index.shtml
130. <http://www.fischl.de/usbasp/>
131. <http://www.xs4all.nl/~dicks/avr/usbtiny/>
132. <http://www.pjrc.com/teensy/index.html>
133. <http://www.fourwalledcubicle.com/LUFA.php>
134. <http://www.electroons.com/2009/11/09/enabling-avr-uart-tx/>
135. <http://www.electroons.com/2009/11/10/enabling-avr-uart-rx/>
136. <http://bluecontroller.com/>
137. <http://shop.jeelabs.com/>
138. <http://projects.adamh.cz/cs:atmega8board>
139. <http://projects.adamh.cz/avrnaepky>
140. <http://blog.adamh.cz/index.php?clanek=elektrotechnika/96-atmega8-board-a-potisk-pcb>
141. <http://evilmadscience.com/tinykitlist/74-atmegaxx8>
142. <http://evilmadscience.com/tinykitlist/112-tiny2313>
143. <http://evilmadscience.com/productsmenu/tinykitlist/180-diavolino>
144. <http://evilmadscience.com/productsmenu/tinykitlist/176-avrstarter>
145. <http://www.schatenseite.de/ispadapter.html?&L=2>
146. <http://tinkerlog.com/2009/01/18/attiny-breadboard-headers/#more-383>
147. <http://www.schatenseite.de/breadpower.html?&L=2>
148. http://www.ikalogic.com/freq_meter_2.php
149. <http://www.myplace.nu/avr/countermeasures/index.htm>
150. <http://cappels.org/dproj/AVRfpls/fmpg.htm>
151. <http://www.cappels.org/dproj/30MHzfmeter/30MhzFmtr.html>
152. http://www.dl5neg.de/freq_counter/freq_counter.html
153. <http://benryves.com/projects/tvtext>
154. <http://www.youtube.com/watch?v=RFjIAvW1OcQ>
155. <http://sbc.rictor.org/io/vid3.html>
156. <http://avga.prometheus4.com/index.php?p=1-0>
157. http://www.serasidis.gr/circuits/AVR_VGA/avr_vga.htm
158. <http://www.circuitdb.com/circuits/id/69>
159. http://www.batsocks.co.uk/readme/art_SerialVideo_1.htm
160. <http://www.obdev.at/products/vusb/index.html>
161. <http://vusb.wikidot.com/hardware>
162. <http://www.recursion.jp/avrcdc/>

163. <http://aboutmicrocontroller.blogspot.com/2008/11/avr-1-key-keyboard-project.html>
164. <http://avr-mcu.dxp.pl/atmega162+ram+eval+board.html>
165. <http://www.scienceprog.com/adding-external-memory-to-atmega128/>
166. <http://dlangenberg.googlepages.com/atmega8basics>
167. <http://sites.google.com/site/libby8dev/fignition>
168. <http://omegav.ed.ntnu.no/wiki/index.php/AVR/Examples/IO>
169. <http://omegav.ed.ntnu.no/wiki/index.php/AVR/Examples/AD>
170. <http://programujte.com/index.php?akce=clanek&cl=2007041902-avr-a-ad-prevodnik>
171. [http://www.rn-wissen.de/index.php/ADC_\(Avr\)](http://www.rn-wissen.de/index.php/ADC_(Avr))
172. <http://tuxgraphics.org/electronics/200903/three-digit-display.shtml#0lindex0>
173. <http://www.fourwalledcubicle.com/LUFA.php>
174. <http://www.obdev.at/products/vusb/index.html>
175. <http://www.recursion.jp/avrcdc/index.html>
176. <http://www.mikrocontroller.net/articles/Bitmanipulation>
177. <http://iamsuhasm.wordpress.com/tutsproj/avr-gcc-tutorial/>
178. <http://meloun.esportuj.cz/index.php?text=16-makra-pro-bitove-operace>
179. <http://code.google.com/p/arduino/>
180. <http://cs.wikipedia.org/wiki/Arduino>
181. <http://www.techdot.eu/index.php/2008/02/01/arduino-prvni-program-blikani-led/>
182. <http://www.freeduino.org/>
183. <http://java.dzone.com/news/arduino-development-using>
184. <http://arduinothedocumentary.org/>
185. <http://arduino.tw>
186. <http://www.root.cz/n/arduino/>
187. <http://feilipu.posterous.com/freetronics-2010-arduino-overclocking-and-rev>
188. <http://www.danielandrade.net/2008/07/05/temperature-sensor-arduino/>
189. <http://www.freetronics.com/products/practical-arduino>
190. <http://my.safaribooksonline.com/book/hobbies/9781449399368>
191. <http://www.ladyada.net/learn/arduino/index.html>
192. <http://www.google.cz/search?q=Arduino+ATmega640&btnG=Hledat&hl=cs&sa=2>
193. http://shop.snailinstruments.com/index.php?main_page=index&cPath=68_134
194. <http://www.hwkitchen.com/arduino/>
195. <http://www.czechduino.cz/>
196. http://www.nuelectronics.com/estore/index.php?main_page=product_info&products_id=15
197. <http://www.adafruit.com/>
198. <http://pjrc.com/>
199. <http://shop.jeelabs.com/>

200. <http://www.sparkfun.com/commerce/categories.php>
201. <http://store.makerbot.com/sanguino-v1-0-pcb.html>
202. <http://www.tinkersoup.de/>
203. <http://blog.harvie.cz/tag/arduino/>
204. <http://www.arduinoors.net>
205. <http://www.instructables.com/id/A-credit-card-sized-Ethernet-Arduino-compatible-co/>
206. <http://www.instructables.com/id/The-Arduino-AA-Undershield/>
207. <http://interface.khm.de/index.php/lab/experiments/arduino-frequency-counter-library/>
208. <http://interface.khm.de/index.php/labor/experimente/arduino-realtime-audio-processing/>
209. <http://interface.khm.de/index.php/lab/experiments/input-channel-extender/>
210. <http://bitlash.net/wiki/start>
211. <http://www.instructables.com/id/A-credit-card-sized-Ethernet-Arduino-compatible-co/>
212. <http://www.stillhq.com/arduino/000007.html>
213. http://en.wikipedia.org/wiki/Arduin_of_Italy
214. <http://sourceforge.net/projects/rhyduino/>
215. <http://arduino.cc/en/Main/Software>
216. <http://blog.tepper.cz/archive/2008/11/13/arduino---seznaacutemeniacute.aspx>
217. <http://blog.tepper.cz/archive/2008/11/13/arduino---prvni-projekt.aspx>
218. <http://www.robotshop.ca/arduino-5min-tutorial-1.html>
219. <http://tronixstuff.wordpress.com/2010/04/04/getting-started-with-arduino-chapter-zero/>
220. <http://tronixstuff.wordpress.com/2010/04/10/getting-start>
221. <http://tronixstuff.wordpress.com/2010/04/15/getting-started-with-arduino-chapter-two/>
222. <http://tronixstuff.wordpress.com/2010/04/20/getting-started-with-arduino-chapter-three/>
223. <http://tronixstuff.wordpress.com/2010/04/30/getting-started-with-arduino---chapter-four/>
224. <http://tronixstuff.wordpress.com/2010/05/06/getting-started-with-arduino-chapter-five/>
225. <http://tronixstuff.wordpress.com/2010/05/14/getting-started-with-arduino---chapter-six/>
226. [http://tronixstuff.wordpress.com/2010/05/16/getting-started-with-arduino---chapter six addendum/](http://tronixstuff.wordpress.com/2010/05/16/getting-started-with-arduino---chapter-six-addendum/)
227. <http://tronixstuff.wordpress.com/2010/05/20/getting-started-with-arduino---chapter-seven/>
228. <http://tronixstuff.wordpress.com/2010/05/26/getting-started-with-arduino---chapter-eight/>
229. <http://tronixstuff.wordpress.com/2010/06/06/getting-started-with-arduino---chapter-nine/>
230. <http://tronixstuff.wordpress.com/2010/06/14/getting-started-with-arduino-chapter-ten/>
231. <http://tronixstuff.wordpress.com/2010/06/26/getting-started-with-arduino---chapter-eleven/>
232. [http://tronixstuff.wordpress.com/2010/07/24/getting-started-with-arduino---chapter thirteen/](http://tronixstuff.wordpress.com/2010/07/24/getting-started-with-arduino---chapter-thirteen/)
233. <http://tronixstuff.wordpress.com/2010/08/06/moving-forward-with-arduino---chapter-14-xbee-introduction/>
234. [http://tronixstuff.wordpress.com/2010/08/18/moving-forward-with-arduino---chapter-15---rfid-introduction/](http://tronixstuff.wordpress.com/2010/08/18/moving-forward-with-arduino---chapter-15-rfid-introduction/)
235. <http://jeremyblum.com/2011/03/07/arduino-tutorial-10-interrupts-and-hardware-debouncing/>

- 236. <http://www.billporter.info/?p=308>
- 237. <http://www.uchobby.com/index.php/2007/11/24/arduino-interrupts/>
- 238. http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf
- 239. http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html
- 240. <http://cutecom.sourceforge.net/>
- 241. <http://live.gnome.org/moserial>
- 242.
- 243.
- 244.
- 245.
- 246. <http://arduino.cc/en/Hacking/Programmer>
- 247. <http://arduino.cc/en/Reference/HomePage>
- 1. <http://arduino.cc/en/Reference/AnalogRead>
- 2. <http://arduino.cc/en/Tutorial/AnalogInputPins>
- 1. <http://arduino.cc/en/Reference/AnalogReference>
- 2. <http://arduino.cc/en/Tutorial/AnalogInputPins>
- 1. <http://arduino.cc/en/Reference/AnalogWrite>
- 2. <http://principlalabs.com/arduino-pulse-width-modulation/>
- 1. <http://arduino.cc/en/Reference/AttachInterrupt>
- 1. <http://arduino.cc/en/Reference/Constrain>
- 1. <http://arduino.cc/en/Reference/Delay>
- 1. <http://arduino.cc/en/Reference/DetachInterrupt>
- 1. <http://arduino.cc/en/Reference/DigitalRead>
- 1. <http://arduino.cc/en/Reference/DigitalWrite>
- 1. <http://arduino.cc/en/Reference/Interrupts>
- 1. <http://arduino.cc/en/Reference/Map>
- 1. <http://arduino.cc/en/Reference/Max>
- 1. <http://arduino.cc/en/Reference/AttachInterrupt>
- 1. <http://arduino.cc/en/Reference/Min>
- 1. <http://arduino.cc/en/Reference/NoInterrupts>
- 1. <http://arduino.cc/en/Reference/NoTone>
- 1. <http://arduino.cc/en/Reference/PinMode>
- 2. <http://arduino.cc/en/Tutorial/DigitalPins>
- 3. <http://arduino.cc/en/Tutorial/AnalogInputPins>
- 1. <http://arduino.cc/en/Reference/Tone>
- 271. <http://arduino.cc/en/Tutorial/Sweep>
- 272. <http://code.google.com/p/tinkerit/>

1. <http://arduino.cc/en/Reference/ServoRead>
1. <http://arduino.cc/en/Reference/ServoWrite>
1. <http://arduino.cc/en/Reference/ServoWriteMicroseconds>
276. <http://bitlash.net/wiki/docindex>
277. <http://github.com/radekh/hc595>
278. <http://homepages.which.net/~paul.hills/Software/ShiftRegister/ShiftRegisterBody.html>
279. <http://code.google.com/p/rogue-code/wiki/ToneLibraryDocumentation>
1. <http://arduino.cc/en/Serial/Available>
1. <http://arduino.cc/en/Serial/Begin>
1. <http://arduino.cc/en/Serial/End>
1. <http://arduino.cc/en/Serial/Flush>
1. <http://arduino.cc/en/Serial/Print>
1. <http://arduino.cc/en/Serial/Println>
1. <http://arduino.cc/en/Serial/Read>
1. <http://arduino.cc/en/Serial/Write>
288. <http://fritzing.org>
289. <http://www.wulfden.org/TheShoppe/freduino/ADABOOT.shtml>
290. <http://www.ladyada.net/library/arduino/bootloader.html>
291. <http://www.wulfden.org/TheShoppe/freduino/ADABOOT.shtml>
292. <http://shieldlist.org/>
293. <http://www.freetronics.com/products/ethernet-shield-with-poe>
294. <http://www.freetronics.com/products/protoshield-basic>
295. <http://www.freetronics.com/products/stackable-arduino-shield-headers>
296. http://www.adafruit.com/index.php?main_page=product_info&cPath=52&products_id=85
297. <http://www.flickr.com/photos/spikenzie/4095424939/in/photostream/>
298. <http://www.spikenzielabs.com/SpikenzieLabs/Prototino.html>
299. <https://github.com/jonoxer/etherShield>
300. http://www.nuelectronics.com/estore/index.php?main_page=product_info&cPath=1&products_id=4
301. <http://blog.thiseldo.co.uk/?p=329>
302. <http://blog.thiseldo.co.uk/?p=493>
303. <http://www.tuxgraphics.org/electronics/200905/embedded-tcp-ip-stack.shtml>
304. <http://www.tucsni.nl/jee/>
305. <http://jeelabs.net/projects/cafe/wiki>
306. examples/avr/Arduino/
307. <http://github.com/radekh/Led8x8>
308. <http://hackaday.com/2009/07/15/avr-isp-programming-via-arduino/>
309. <http://www.flickr.com/photos/drug123/3718355976/in/pool-76206823@N00>

- 310. <http://code.google.com/p/mega-isp/>
- 311. <http://drug123.org.ua/mega-isp-shield/>
- 312. <http://www.flickr.com/photos/drug123/3718360064>
- 313. <http://arduino.cc/en/Tutorial/ArduinoISP>
- 314. <http://tinkerlondon.com/now/2007/02/19/make-an-arduino-ng-into-an-avr-isp-programmer-for-good/>
- 315. <http://drug123.org.ua/mega-isp-shield/>
- 316. <http://code.google.com/p/mega-isp/>
- 317. <http://freaklabs.org/index.php/chibiArduino.html>
- 318. http://www.freaklabsstore.com/index.php?main_page=product_info&cPath=22&products_id=187
- 319. <http://www.arduino.cc/en/Main/Hardware>
- 320. <http://www.freduino.org/>
- 321. <http://www.gravitech.us/arna30wiatn.html>
- 322. <http://shop.moderndevice.com/products/bbb-kit>
- 323. <http://www.adafruit.com/>
- 324. <http://seeedstudio.com/depot/>
- 325. <http://www.pjrc.com/teensy/>
- 326. <http://www.seeedstudio.com/blog/2008/09/15/seeeduino-release-notice-ver10b/>
- 327. <http://timewitharduino.blogspot.com/2009/09/introducing-duino644.html>
- 328. <http://www.ladyada.net/make/boarduino/>
- 329. <http://www.instructables.com/id/Perfboard-Hackduino-Arduino-compatible-circuit/>
- 330. <http://telinks.wordpress.com/2010/09/07/starduino/>
- 331. <http://txapuzas.blogspot.com/2010/07/paperduino-pcb.html>
- 332. <http://txapuzas.blogspot.com/2010/07/paperduino-perfboard.html>
- 333. <http://txapuzas.blogspot.com/2010/07/paperduino-stripboard.html>
- 334. http://www.bhasha.co.cc/product.php?id_product=34
- 335. <http://wiki.hackspace.org.uk/wiki/Project:Nanode>
- 336. <http://sanguino.cc/start>
- 337. <http://leaflabs.com/devices/maple/>
- 338. <http://www.alvie.com/zpuino/index.html>
- 339. <http://www.ladyada.net/library/arduino/bootloader.html>
- 340. <http://www.ladyada.net/>
- 341. http://www.adafruit.com/index.php?main_page=index&cPath=19
- 342. <http://www.arduino.cc/playground/Main/HeartbeatSketch>
- 343. <http://processing.org/>
- 344. <http://www.openprocessing.org/>

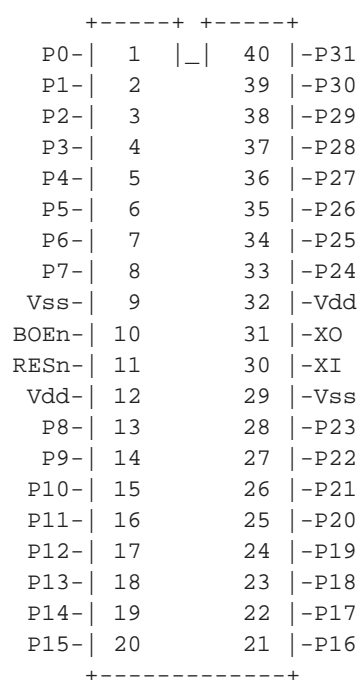
Kapitola 41. Propeller

Odkazy:

- [Propeller General Information](#)¹
- [Parallax Propeller](#)² na Wikipedii
-

Propeller od firmy Parallax³ je zajímavý obvod obsahující 8 paralelně běžících 32 bitových procesorů COG propojených kruhovou sběrnici a potřebnou paměť. K dispozici jsou i nástroje pod Linux, jak se lze dočíst v IDE for linux / Mac (and Windows) users - Ver 0.13 - The "Mr Sheen" release⁴.

Obrázek 41-1. P8X32A-D40



Tabulka 41-1. Cog RAM Special Purpose Registers

adresa	název	typ	popis
1F0	PAR	Read-Only	Boot Parameter
1F1	CNT	Read-Only	System Counter
1F2	INA	Read-Only	Input States for P31-P0
1F3	INB	Read-Only	Input States for P63-P32
1F4	OUTA	Read/Write	Output States for P31-P0
1F5	OUTB	Read/Write	Output States for P63-P32
1F6	DIRA	Read/Write	Direction States for P31-P0

adresa	název	typ	popis
1F7	DIRB	Read/Write	Direction States for P63-P32
1F8	CTRA	Read/Write	Counter A Control
1F9	CTRB	Read/Write	Counter B Control
1FA	FRQA	Read/Write	Counter A Frequency
1FB	FRQB	Read/Write	Counter B Frequency
1FC	PSHA	Read/Write	Counter A Phase
1FD	PSHB	Read/Write	Counter B Phase
1FE	VCFG	Read/Write	Video Configuration
1FF	VSCL	Read/Write	Video Scale

Pokud se vám zdají ceny kitů s procesory P8X32A vysoké, můžete si takový udělat sami podle How to make a Propeller Development Kit⁵.

Poznámky

1. <http://www.parallax.com/propeller/>
2. http://en.wikipedia.org/wiki/Parallax_Propeller
3. www.parallax.com
4. <http://forums.parallax.com/forums/default.aspx?f=25&m=298620&p=1&ord=d>
5. <http://www.longhornengineer.com/DIY/PropDev>

Kapitola 42. PSoC

Odkazy:

- Embedded Systems/Cypress PSoC Microcontroller¹
- CYPRESS²

Nástroje a prostředí

- - Forth**
 - Forth/PSoC Forth³
-
-
- Ostatní:
 - m8cutils⁴

Tabulka 42-1. Dostupnost součástek:

typ	prodejce	datum	cena
CY8C27143	TME	2010-02-15	115 Kč bez DPH
CY8C27443-24PXI	TME	2010-02-15	115 Kč bez DPH
CY8C29466-24PXI	TME	2010-02-15	173 Kč bez DPH

Poznámky

1. http://en.wikibooks.org/wiki/Embedded_Systems/Cypress_PSoC_Microcontroller
2. <http://www.cypress.com/>
3. http://en.wikibooks.org/wiki/Forth/PSoC_Forth
4. <http://m8cutils.sourceforge.net/>

Kapitola 43. Projekty s mikrořadiči

Tato kapitola obsahuje náměty a projekty na zařízení, která je možno na mikrořadičích postavit.

Velmi zvláštní projekty:

- The AVR AM Radio Transmitter (and Plantenna)¹ [2010-10-18]
- DIY standalone internet radio receiver with Shoutcast/Icecast protocols² Alexander Yerezeyev [2010-02-20]
-
-

43.1. Využití malých obvodů s nízkým počtem vývodů

Odkazy na projekty:

- Stealth USB CapsLocker³ — ATtiny45
- Capslocker Rebuild⁴
- A Really Tiny tinyCylon⁵
- USB Physical Hit Counter based on AVR ATtiny25⁶
- the 1-Key-Keyboard project⁷ — ATtiny45/ATtin85
-

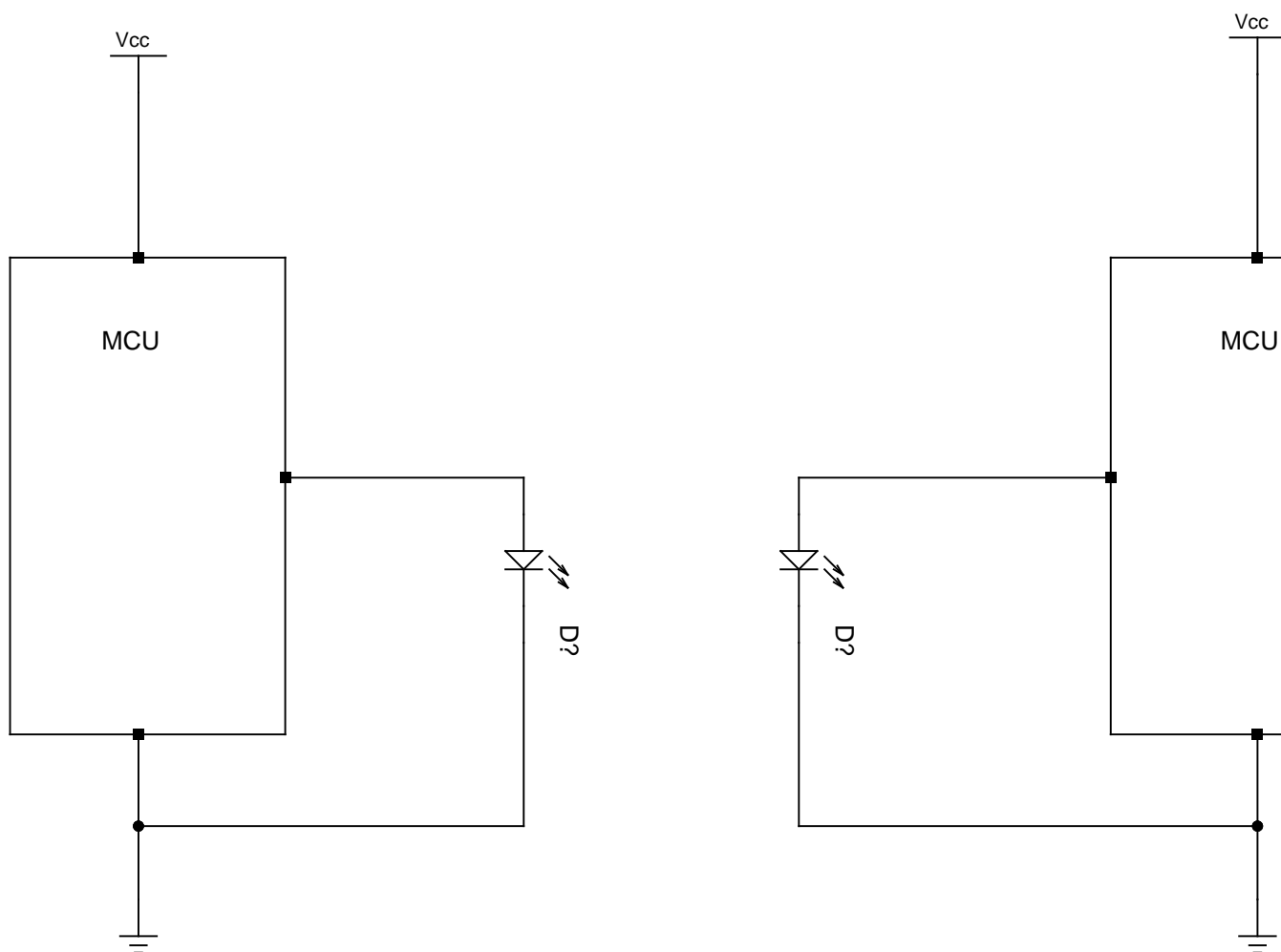
43.2. Komunikace mezi řadiči pomocí LED

Tento nápad je ve stádiu zkoumání.

Nosná myšlenka komunikace pomocí LED diody spočívá v tom, že LED dioda nejen že dokáže emitovat světlo, ale dokáže pracovat i obráceně. Tedy vytvářet elektrickou energii při osvětlení. Můžeme tedy vybavit dva mikrořadiče, každý jednou LED diodou. Tyto led diody přiložit k sobě tak aby do sebe svítily navzájem. Nyní již zbývá vyřešit jak tento hardware obsloužit programem.

Program přepíná mezi dvěma směry komunikace. Jedná se tedy o poloviční duplex. Při vysílání přivádíme na diodu napětí (logická 1, logická 0). Při příjmu měříme napětí na diodě pomocí vnitřního AD převodníku v MCU.

Obrázek 43-1. Komunikace dvou MCU pomocí LED diod



Ze způsobu přijímání, tedy použitím AD převodníku k měření fotoelektrického napětí, jsme limitováni na maximální komunikační rychlosti. Toto maximum předpokládám pro MCU Atmel AVR v řádu jednotek kilobitů.

Výhody:

- Zařízení nejsou galvanicky propojena.
- Je využit jen jeden pin MCU.

Nevýhody:

- Nízká rychlost komunikace.
- Nutnost řešit komunikaci od základů programově.
- Vyšší spotřeba energie než při přímém elektrickém propojení.

43.3. Generování videosignálu

Generování videosignálu pro televizní přijímač, nebo počítačový monitor.

Odkazy:

- How to generate video signals in software using PIC⁸
- PAL TV timing and voltages⁹

-
-
-

43.4. Čtení analogových hodnot

*

Odkazy:

- 58
- ATmega48 + LM335 + MAX232 = Serial Port Multi-Channel Temperature Measurement¹⁰ — Scott [2010-11-24]
-

Řada mikrořadičů má na čipu implementován AD převodník a je schopna odečítat analogová napětí na jednom či více vstupech. Většina AVR ATmega/ATtiny a také PIC mikrořadičů je má. Pokud ne, můžeme použít externí AD převodník v jednom čipu.

43.5. RFID

*

Odkazy:

- USING AN AVR AS AN RFID TAG¹¹ [2008-09-21]
- SIMPLEST RFID READER?¹² [2008-08-22] (Propeler)
- Proximity Security System¹³
- Building a RFID reader¹⁴ PIC16F628A
- Smart-Its RFID Tag Reader¹⁵
-
-

43.6. Čítače

*

Odkazy:

- Home-Brew Transceiver Taking Shape!¹⁶
- \$10 Frequency Counter Finished!¹⁷
-
-

43.7. Rozšíření počtu I/O

*

Odkazy:

- Using Serial Peripheral Interface (SPI) Master and Slave with Atmel AVR Microcontroller¹⁸
- MCP23S17 SPI I/O Expander¹⁹
- MCP23S17 addressable 16-Bit I/O Expander with SPI²⁰
-

Součástky:

- 16-42
- 16-41
- MCP23S08²¹ (SPI), MCP23008²² (I²C) — 8-Bit I/O Expander with Serial Interface
- MCP23S17²³ (SPI), MCP23017²⁴ (I²C) — 16-Bit I/O Expander with Serial Interface
- MCP23S09²⁵ (SPI), MCP23009²⁶ (I²C) — 8-Bit I/O Expander with Open-Drain Outputs
- MCP23S18²⁷ (SPI), MCP23018²⁸ (I²C) — 16-Bit I/O Expander with Open-Drain Outputs
- MCP23016²⁹ (I²C), nedoporučuje se pro nové konstrukce
- PCF8574³⁰
- MAX7317³¹ — 10-Port SPI-Interfaced I/O Expander
- MAX7318³² — I2C, 16-Bit
- MAX7301³³ — SPI, 20-Port
- M66010FP/GP³⁴
-

K dispozici jsou součástky od fy Microchip MCP23009/MCP23S09. Tyto součástky jsou 8-mi bitový expander. Každý pin 8-mi pinové brány může být nastaven jako vstupní nebo výstupní. Připojení k MCU je přes I²C (MCP23009) nebo SPI (MCP23S09)

Obvody MCP23018/MCP23S18 mají 16 I/O. Pouzdro 28 PDIP, 28 SOIC, 24 SSOP (MCP23018), 24 QFN

43.8. Obsluha LED v maticovém zapojení

*

Odkazy:

- tronixstuff Moving Forward with Arduino – Chapter 18 – RGB LED Matrix³⁵
-

Tedy jak obsloužit několik 7-mi segmentových, či 14-ti a 16-ti segmentových LED. Případně jak obsloužit matice 8*8 případně jiné rozměry.

Poznámky

1. <http://grieg.gotdns.com/blog/?p=312>
2. http://alyer.frihost.net/ethradio/eth_radio_en.htm
3. <http://www.macetech.com/blog/node/46>
4. <http://macetech.com/blog/node/81>
5. <http://www.popsci.com/diy/article/2009-01/really-tiny-tinycylon>
6. <http://www.bobhobby.com/2008/04/22/usb-physical-hit-counter-based-on-avr-attiny25/>
7. <http://blog.flipwork.nl/?x=entry:entry081009-142605>
8. <http://www.rickard.gunee.com/projects/video/pic/howto.php>
9. <http://www.retroleum.co.uk/electronics-articles/pal-tv-timing-and-voltages/>
10. <http://www.swharden.com/blog/2010-11-24-atmega48-lm335-max232-serial-port-multi-channel-temperature-measurement/>
11. <http://scanlime.org/2008/09/using-an-avr-as-an-rfid-tag/>
12. <http://scanlime.org/2008/08/simplest-rfid-reader/>

13. <http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2006/cjr37/Website/index.htm>
14. <http://memweb.newsguy.com/~rhuang/RFID/rfid.html>
15. <http://www.sics.se/~kalle/projects/daphne/RFID/>
16. <http://www.sw Harden.com/blog/2011-01-28-home-brew-transceiver-taking-shape/>
17. <http://www.sw Harden.com/blog/2011-03-14-frequency-counter-finished/>
18. <http://www.ermicro.com/blog/?p=1050>
19. http://www.senita.de/index.php?option=com_content&view=article&id=47
20. <http://mbed.org/users/romilly/notebook/mcp23s17-addressable-16-bit-io-expander-with-spi/>
- 21.
- 22.
- 23.
- 24.
- 25.
- 26.
- 27.
- 28.
- 29.
- 30.
- 31.
- 32.
- 33.
- 34.
35. <http://tronixstuff.wordpress.com/2010/09/28/moving-forward-with-arduino-%E2%80%93chapter-18-%E2%80%93rgb-led-matrix>

Kapitola 44. Virtuální procesory

Nevím kam zařadit tohle téma a tak jsem mu prozatím přiřadil samostatnou část v kapitole o elektronice ačkoliv je to více o software.

Virtuální procesory jak již název napovídá nejsou skutečné v tom významu že bychom je mohli vidět jako čipy. Jsou to jen konstrukce které jsou realizovány programem běžícím na skutečných procesorech. Jejich přínos je zejména v tom, že standardizují platformu, byť na virtuální úrovni a umožňují tak provozovat programy pro ně napsané na odlišných typech skutečných procesorů. Jako příklad bych uvedl virtuální stroje/procesory pro jazyk Java či Scheme. Nebo z jiné strany Zork vyvinutý pro psaní her firmou Infocom.

Odkazy:

- MIX¹
- MMIX²
- SECD³
-
-
-

44.1. Zork

Odkazy:

- Z-Machine Standards⁴
-
-
-

Poznámky

1. <http://en.wikipedia.org/wiki/MIX>
2. <http://en.wikipedia.org/wiki/MMIX>
3. http://en.wikipedia.org/wiki/SECD_machine
4. <http://www.inform-fiction.org/zmachine/standards/index.html>

VIII. Periferie a rozhraní

** Zvážit jestli by nebylo vhodné tento oddíl zahrnout do oddílu VII – „Mikroprocesory a mikrořadiče“.*

Kapitola 45. Sério-paralelní registry

Odkazy:

- Expanded I/O using 74165 and 74595¹

Zde jsou jednoduchá zapojení sériových registrů. Používají se pro rozšíření počtu vstupních či výstupních linek na úkor času. Používané obvody jsou:

- 74HC164 — 8-bit Serial-Input/Parallel-Output
- 74HC165
- 74HC166 — 8-bit Parallel-Load Shift Registers
-
- 74HC595
- 74HC597

Poznámky

1. http://www.blitzlogic.com/exp_io.htm

Kapitola 46. SD/MMC

Odkazy:

- Digital electronics and programing: Real schematik for interfacing One SD/MMC card to atmega microcontrollers¹

•

Konstrukce:

•

46.1. Připojení MMC k AVR ATmega

Konstrukce:

- Atmel ATmega (ATmega16 / ATmega32) - MMC (Multi Media Card) Flash Memory Extension²

•

Pokud má náš procesor jiné napájecí napětí, například 5V jako je v uvedené konstrukci, musíme rozhraní MMC karty přizpůsobit. Pokud nechceme riskovat poškození karty napětím 5V.

- Napájecí napětí pro kartu MMC/SD je získáno pomocí stabilizátoru, například Low Drop Voltage stabilizátoru LD1086V33 nebo jiného. Low Drop proto, aby mu na vstupu stačilo 5V jimiž je napájen procesor.
- Karta je připojena k sériovému SPI rozhraní MCU. Využijem tím hardware který máme v procesoru k dispozici a ušetříme výkon pro vlastní aplikaci.
- Vývody SCK, MOSI, a SS jsou připojeny na odpovídající vývody rozhraní karty (SCK→CLK, MOSI→DataIn, SS→!CS) jsou připojeny přes odporové děliče. Přesně na vývod procesoru je připojen odpor 1.8kΩ tento odpor je pak druhým vývodem připojen na rozhraní MCU a toto rozhraní je pak dalším odporem o hodnotě 3.3kΩ připojeno k zemi.
- Vývod DataOut rozhraní MMC/SD je připojen přímo na vstup procesoru MISO. Výstupní napětí na tomto vývodu spolehlivě stačí procesoru k rozeznání Log 1.

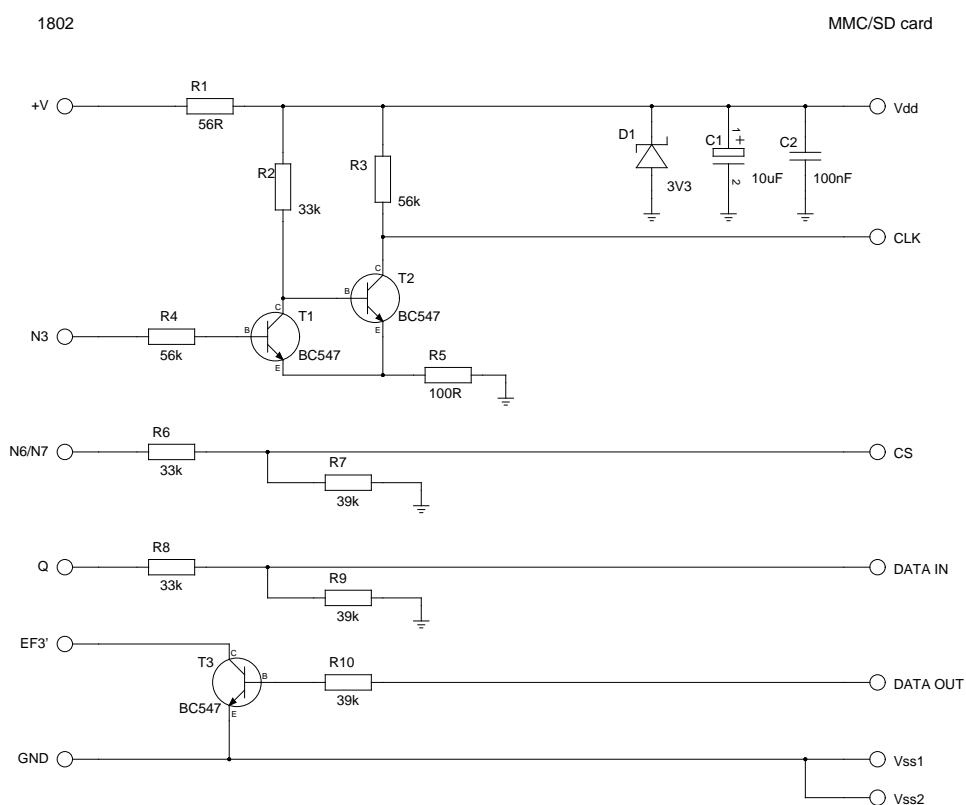
Ukázkové připojení je vidět například na <http://www.captain.at/electronic-atmega16-mmc-schematic.png>.

46.2. Připojení k CDP 1802

Odkazy:

- the Cosmac 1802 handheld computer³

Obrázek 46-1. The 1802 MMC/SD interface circuit



Poznámky

1. <http://digitalelectronicsandprograming.blogspot.com/2008/10/real-schematik-for-interfacing-one.html>
2. <http://www.captain.at/electronic-atmega-mmc.php>
3. <http://ringcomps.co.uk/hhc/>

Kapitola 47. LCD a LED displeje

Odkazy:

- Berty's Home Page¹
- LCDInfo.com²

47.1. Řadiče zobrazovacích jednotek

Odkazy:

- pcd8544³
-

* Až přibude řadičů, bude potřeba je rozřadit podle výrobců.

47.1.1. PCD8544

Tento řadič je použit v řadě malých displejů, zejména displejů v mobilních telefonech 47.2.2

TBD;

47.1.2. PCDF8814

Tento řadič je použit v displejích mobilních telefonů jako jsou LCD Nokia 1100

47.1.3. ST7548T

Odkazy:

- Datasheet pro ST7548T⁴

Tento řadič je použit v displejích mobilních telefonů LCD Nokia 3310

47.2. Displeje z mobilů a jiných zařízení

Odkazy:

- MT LCD - knihovny, schemata, odkazy⁵

Obchody kde se dá koupit:

- Náhradní díly a příslušenství na mobily⁶
- amilo s.r.o.⁷

Články:

- Epson S1D15G10 Nokia LCD (Work In Progress)⁸
-

•

Odkazy na displeje:

•

Nokia 5110/6110:

- Xmob.cz⁹ — 236Kč (2008-11-18)

•

•

Nokia 8210/8250:

- Xmob.cz¹⁰ — 240Kč (2008-11-18)

•

•

Nokia 3410/3350:

- Xmob.cz¹¹ — 298Kč (2008-11-18)

•

•

Siemens

- Using the Siemens S65-Display¹²

•

•

Nokia

- — xKč (2008-00-00)

•

Tabulka 47-1. Informace o monochromatických LCD displejích

typ, označení	použit v mobilech	používané řadiče	velikost v pixelech	poznámka
	LCD Nokia 3310	PCD8544, ST7548T	84*48	
	LCD Nokia 1100	PCF8814	96*65	
	Nokia 3510		98*67	

Tabulka 47-2. Parametry displejů

Typ	Rozměry	Barvy	Napájení	řadič
Nokia 1100	96*68	monochrome	2.8	PCF8814

Typ	Rozměry	Barvy	Napájení	řadič
Nokia 3310/3330	84*48	B&W	2.7-3.3	pcd5844
Nokia 5110, 6110				
Nokia 8210, 8850				
Nokia 6100, 7210, 6610, 7250, 6220	132*132	12b	3.3V	Epson S1D15G10, Philips PCF8833
Nokia 1110, 1110i				
Nokia 6210				
Nokia 6510				
Nokia 1112				
Nokia 3510i				
Nokia 5200, 6060, 6070, 6080, 6101, 6103, 6125, 6151, 7360	128*160	18b		
Nokia 3610, 5000, 5220, 7210, 7210	240*320			
Nokia 6200, 6301, 6120, 6124, 6555, 8600, 5310, 5320, 6121, 7500	240*320			
Nokia 6310, 6310i	96*65	monochrome		

- Nokia 5200¹³
- Nokia 5200¹⁴
- Originální LCD displej pro Nokia 5200, 6060, 6070, 6080, 6101, 6103, 6125, 6151, 7360 - vnitřní¹⁵
-
-

47.2.1. Nokia 1100

47.2.2. Display pro mobil Nokia 3310/3330

Odkazy:

- Nokia 3310 LCD¹⁶
- Connect Nokia 3310 LCD to LPT port¹⁷
- Kousek diskuze k problémům s displeji (nový řadič)¹⁸
- LCD Nokia 3310¹⁹ on YouTube
- LCD do PC z Nokie 3310²⁰
- NOKIA 3310 LCD interfacing with ATmega8²¹ (plošňáček)

Kde se dá display sehnat:

- Xmob.cz²² — 220Kč (2008-11-18)
- Nokia 3310 LCD Driver using a PIC²³
- LCD display pro Nokia 3310/3330²⁴ — 180Kč (2008-11-18)
- Nakupuj levně LCD display Nokia 3310²⁵
-

Konstrukce:

- Levný LCD teploměr²⁶
-
-
-

Technické parametry:

- Rozlišení displeje: 84*48 bodů.
- Napětí: 2.7 až 3.3V
- Použité řadiče: PCD8544 v původních displejích, později nahrazen ST7548T
-
-

V tomto mobilu je použit displej Phillips PCD8544.

Popis signálů

Vdd

Napájecí napětí 2.7 - 3.3V.

SCLK — *serial clock input*

SDIN — *serial data input*

D/!C — *data/command*

!SCE — *chip enable*

GND

Zem, 0V.

Vout

Mezi tento vývod a GND se připojí elektrolytický kondenzátor o kapacitě 1 až 10 μF

!RES — *external reset input*

Po připojení jednoho vzorku displeje k MCU a pokusu jej oživit jsem mimo jiné narazil na problém posunutého obrazu. Na první pohled to vypadalo, jako by displej zobrazoval o 3 řádky níž než by měl. Po důkladném prozkoumání jsem pojal podezření jenž se potvrdilo. Můj vzorek má zcela jiný řadič než js původní PCD8544 a to ST7548T. Tento řadič obslouží displeje až do velikosti 102*66 bodů.

Zapojení vývodů displeje Nokia 3310. Při pohledu ze zadu displeje, kdy konektor je nahoře, jsou kontakty číslované z leva do prava.

1. Vdd
2. Sck
3. Sdin
4. D/C
5. Sce
6. Gnd
7. Vout
8. Res

Obrázek 47-1. Ukázka displeje z Nokia 3310



Tabulka 47-3. Zapojení vývodů Nokia3310 na připájeném plochém vodiči

barva	signál
hnědá	Vdd
červená	Sck
oranžová	Sdin
žlutá	D/C
zelená	Sce
modrá	GND
fialová	Vout
šedá	Res

Veškerá komunikace s displejem se odehrává na vodičích !SCE, D/!C, SCLK, SDIN a RES. Vývod GND je spojen se zemí, na pájecí napětí cca 3V je přivedeno na Vcc a mezi Vout a GND se připojí elektrolytický kondenzátor o kapacitě 1 až 10 μ F.

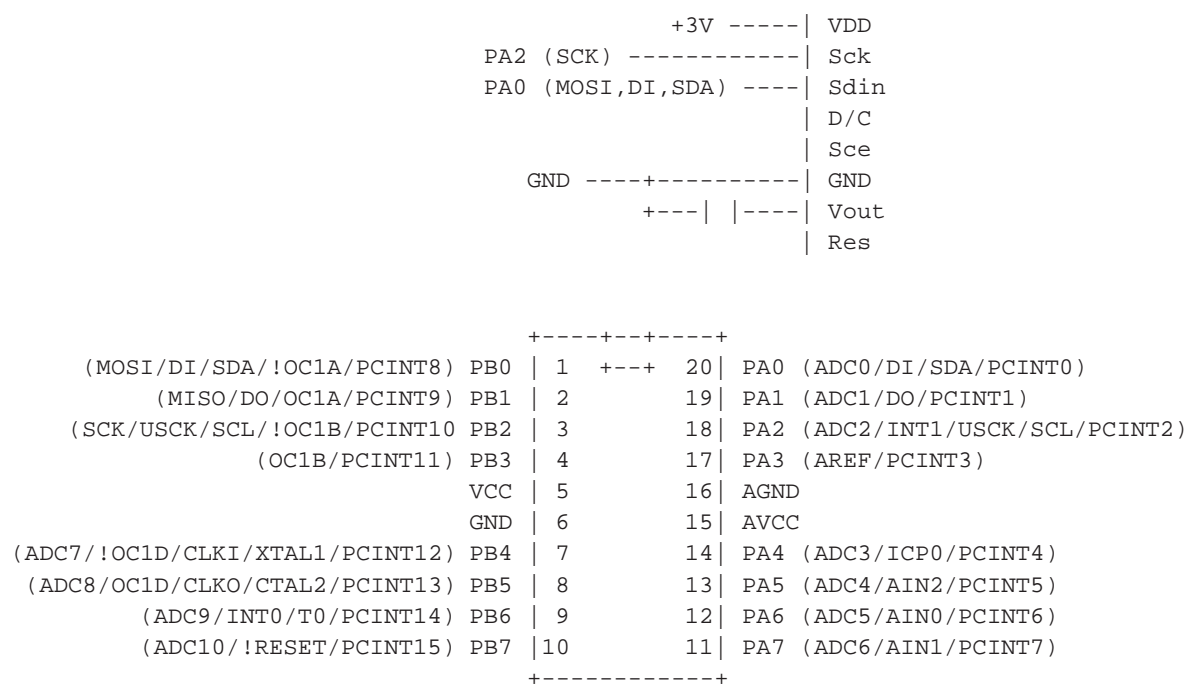
Tabulka 47-4. Připojení displeje k MCU AVR

displej		
VDD	+3V	
SCLK	SCK	
SDIN	MOSI	
D/!C		

displej		
SCE	!SS	
GND	GND	
VOUT	elyt.kond. 1-10 μ F proti GND	
!RST		

Pokud chceme použít vnitřní SPI hardware, tak záleží na připojení vývodů SCK-SCLK, MOSI-SDIN, u ostatních vy měly stačit libovolné volné piny.

Obrázek 47-2. Připojení N3310 k ATtiny461



47.2.3. Nokia 3510, 3510i LCD

Odkazy:

- Nokia 3510, 3510i LCD Pinouts²⁷
- Nokia 3510 Specifications²⁸
- Nokia Farbdisplay 3510i²⁹

Parametry N3510i:

- Velikost obrazu: 98*67 bodů
- Počet barev: 4096
- Technologie: CSTN

Parametry N3510:

- Velikost obrazu: 98*67 bodů
- Počet barev: B/W
- Technologie: FSTN
- Řadič: PCF8814

Tabulka 47-5.

Pin	Signál
1	!Reset
2	!Cs
3	Gnd
4	Sdata
5	ScIk
6	Vdigital 1.8V
7	Bbooster in 2.7V
8	Vlcd max 12V

47.2.4. Siemens S65

Odkazy:

- Using the Siemens S65-Display³⁰ by Christian Kranz, October 2005
-
-

* Siemens used for the mobile phones S65, M65, CX65 and SK65 a 16-bit color TFT display with 132x176 pixel.

Poznámky

1. <http://sanding.tripod.com/Bertys.html>
2. <http://forum.lcdinfo.com/index.php>
3. <http://serdisplib.sourceforge.net/ser/pcd8544.html>
4. <http://www.sitronix.com.tw/sitronix/product.nsf/Doc/ST7548T>
5. <http://www.mcu.cz/forum/viewtopic.php?f=513&t=1344>
6. <http://www.dily-na-mobily.cz/>
7. <http://www.amilo.cz/>
8. <http://www.arduino.cc/playground/S1D15G10NokiaLCD/S1D15G10NokiaLCD>
9. http://www.xmob.cz/k_16-Nokia-LCD-displeje-flex-kabely-folie-na-display/p_nlcd653-Originalni-LCD-displej-pro-Nokia-5110-6110
10. http://www.xmob.cz/k_16-Nokia-LCD-displeje-flex-kabely-folie-na-display/p_nlcd659-Originalni-LCD-displej-pro-Nokia-8210-8850
11. http://www.xmob.cz/k_16-Nokia-LCD-displeje-flex-kabely-folie-na-display/p_nlcd608-Originalni-LCD-displej-pro-Nokia-3410-3350-komplet
12. http://www.superkranz.de/christian/S65_Display/DisplayIndex.html
13. <http://mobil.idnes.cz/nokia-5200-c57-/katalog.asp?navez=5200&sa=HLEDEJ&id=497>
14. <http://www.esato.com/phones/index.php/phone=258>
15. http://www.xmob.cz/k_16-Nokia-LCD-displeje-flex-kabely-folie-na-display/p_nlcd610-Originalni-LCD-displej-pro-Nokia-5200-6060-6070-6080-6101-6103-6125-6151-7360-vnitri

16. <http://www.myplace.nu/mp3/nokialcd.htm>
17. <http://www.scienceprog.com/connect-nokia-3310-lcd-to-lpt-port/>
18. <http://www.edaboard.com/ftopic324477.html>
19. <http://www.youtube.com/watch?v=OZgKUa2y9tg>
20. http://portal.sedf.net/modules.php?name=Downloads&d_op=viewdownload&cid=28
21. <http://www.dharmanitech.com/2008/09/nokia-3310-lcd-interfacing-with-atmega8.html>
22. http://www.xmob.cz/k_16-Nokia-LCD-displeje-flex-kabely-folie-na-display/p_nlcd651-Originalni-LCD-displej-pro-Nokia-3310-3330
23. http://www.100acre.org/elec/nokia_lcd/
24. <http://www.amilo.cz/index.php?productID=1418>
25. <http://nakupujlevne.eu/mobilni-telefony/nokia/lcd-display-nokia-3310/>
26. <http://hw.cz/teorie-praxe/konstrukce/art2134-levny-lcd-teplomer.html>
27. <http://sanding.tripod.com/ni3510.html>
28. <http://stats.getjar.com/device/Nokia/3510>
29. <http://www.xmail.net/martin-k/display.htm>
30. http://www.superkranz.de/christian/S65_Display/DisplayIndex.html

Kapitola 48. JTAG

Odkazy:

- JTAG¹
-
-

Poznámky

1. <http://www.freelabs.com/~whitis/electronics/jtag/>

Kapitola 49. USB

Odkazy:

- USB¹
- USB Designer Links²
- USB1.1 Integrated Circuits & Development Boards³ from Beyond Logic
- USB Made Simple⁴ — A Series of Articles on USB
- USB in NutShell⁵
-

49.1. Fyzická vrstva

Fyzicky je USB realizováno 4 vodičovým kabelem se speciálními, tzv. USB konektory. Těchto konektorů je více typů a velikostí.

Kabelem jsou propojena vždy dvě USB zařízení. Pokud potřebujeme připojit více zařízení, využíváme tzv. USB HUB který má více konektorů a je tím pádem základním stavebním kamenem stromové struktury.

4 vodiče USB kabelu jsou tyto:

Tabulka 49-1. USB kabel/konektor

pin	barva	význam/funkce
1	červená	V _{BUS} (5 Volt)
2	bílá	D-
3	zelená	D+
4	černá	Ground

49.2. Čipy

Součástky které nám pomohou s realizací USB rozhraní.

Odkazy:

•

USB Slave

- PDIUSB12 FAQ⁶
- Micro-USB Module⁷ — USB bus to micro adapter built on CP2102 chip
- FT245⁸ — USB FIFO (USB Parallel)
- FT232⁹ — USB to Serial
- FT2232¹⁰ — USB to Serial
-
-

•

USB Host

- FTDI VNC1L T/R - Vinculum USB Host Controller Device #16620¹¹
-

49.3. Konstrukce

- World' Smallest Low-speed USB Analyzer (Atapchi)¹² — pužit SX Scenix (kompatibilní z PIC)
- California Dreamin SX USB Keyboard Demo¹³

49.4. Převodníky USB ↔ RS232

*

49.4.1. USB ↔ RS232TTL z kabelu NOKIA

*

Odkazy:

- Identificando pines del DATA CABLE de USB a serial¹⁴
- DIY USB to Serial Cable For \$3!¹⁵
- Use a Nokia Serial Cable on an ARM9 Linkstation¹⁶
- Cost effective (cheap!) USB Serial cable for any project¹⁷
- AddASerialPort¹⁸ Why Add A Serial Port? Because you can!!
- Nokia Pop-port connector pinout¹⁹
- Pinouts for Nokia Pop-Port²⁰
- HwB Pop-Port²¹
- FAQ1: Nokia Data / Connectivity Cables DKU-2 DKU-5 CA-42 CA-53²²
-
-
-

Převodníky USB RS232 TTL se dají postavit z USB čipů. Dají se i koupit hotové ve formě kabelů či malých destiček. Cena takových řešení se obvykle pohybuje od \$10 do \$30. Cenově zajímavou alternativou je použít kabely k mobilním telefonům NOKIA. Jejich cena je, při nákupu u čínských obchodníků na Ebay, okolo \$2.

Tabulka 49-2. Použitelné kabely

typ	počet vodičů	
CA-42	3	Podle HwB Pop-Port ²³ se jedná o novější variantu DKU-5 jenž má maximální rychlost 230400 bps.
DKU-5	4	Maximální rychlost 115200 bps
C-261		nezkoušeno
DKU-2		čistý USB kabel
CA-53		čistý USB kabel
CA-70		čistý USB kabel

Nejdříve zapojení konektoru na straně mobilního telefonu. Našel jsem následující obrázek.

Obrázek 49-1. Zapojení konektoru NOKIA



Podařilo se mi najít stránku se zapojením konektoru Nokia²³, z této stránky je následující tabulka.

Tabulka 49-3. Zapojení konektoru nokia

pin	název	popis
1	Vin	Charger input
2	GND	Charger ground
3	ACI	Accessory Control Interface / Auto Connect Ignition, (short with pin 2 for handsfreerecognition)
4	V Out / Vdd+	For Hansfree (ex. HS-23): microchip power supply
5	USBowDet	USB Power Detection, (USB Vcc +5V)
6	FBus Rx/USB D+	F-Bus or USB
7	FBus Tx/USB D-	F-Bus or USB
8	GND	Data GND (USB GND)
9	X Mic -	Audio in - External Microphone
10	X Mic +	Audio in - External Microphone
11	HS Ear L-	Audio out - External Audio out - left channel
12	HS Ear L+	Audio out - External Audio out - left channel
13	HS Ear R-	Audio out - External audio out - right channel
14	HS Ear R+	Audio out - External audio out - right channel
plášť	GND	shiled GND

49.4.1.1. CA-42

* *Attributy:* id="CA-42"

Toto je první kabel, který jsem zkoušel. Bylo to tak trochu omylem, chtěl jsem objednat DKU-5, ale na webu v

článku bylo zaměněné označení a tak jsem objednal CA-42. Po zapojení fungoval bez problémů, na první pokus.

* *Obrázky dodám jak budu "vyrábět" další kabel.*

Tabulka 49-4. Zapojení kabelu CA-42

barva	funkce	pin Nokia
modrá	phone TX	7
červená	phone RX	6
oranžová	GND	8

49.4.1.2. DKU-5

* *Attributy: id="DKU-5"*

Odkazy:

- serial cable analog (original DKU-5 is an USB cable with complicated schematic)²⁴

Tento kabel by měl mít i vodič s napájením z USB. Kabel je tedy čtyřžilový. Jak je na obrázku vidět, přestřižený kabel má čtyři vodiče. Koncovka kabelu a vlastně celý kabel je v černé barvě. Konektory na obou stranách jsou nerozebíratelné. Na USB konektoru není označení typu kabelu, jak je tomu v případě kabelu CA-42. Ale to může být způsobeno tím, že nálepka s označením nedržela, či upadla. Ostatně u CA-42 mi nejspíš upadne taky.

Obrázek 49-2. Pohled na přestřižený kabel DKU-5



Nyní identifikace jednotlivých vodičů. Jak již bylo psáno jinde na internetu, je třeba u každého kabelu prověřit zapojení vodičů. Nemůžeme se slepě spoléhat na to jak byl zapojený poslední kabel který jsme si upravovali.

Tabulka 49-5. Zapojení kabelu DKU-5

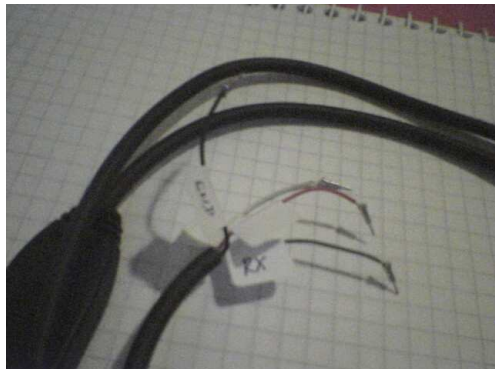
barva	funkce	pin Nokia
bílá	phone TX	7
červená	neznámá	5
zelená	phone RX	6
černá	GND	8

Tak takhle jsem nameřil konektor k Nokii. Co mě mate, že jsem čekal zapojení pinu 4 a ne pinu 5 jak jsem naměřil. Rozdílů je více. Podle odkazů, které jsem našel na internetu, má mít DKU-5 kabel pět vodičů, já vidím jen čtyři. Červený vodič vede na pin konektoru Nokie o kterém nic nevím. Pokud jsem zkouším měřit napětí na červeném vodiči po zasunutí kabelu do USB konektu, doměřil jsem se proti zemi 0.09 V. Podle popisu konektoru

k Nokii, jenž je uveden o pár stran dříve, má na tomto pinu být plné napětí USB.

Takž zapájet nožičky, používám nožičky z precizní patice, a odzkoušet. Na kontaktním poli mám něco složeného, takže jsem vyměnil jeden USB-RS232TTL kabel za tento. Funguje.

Obrázek 49-3. Pohled na nožičky hotového kabelu DKU-5



Papírky na vodičích používám kvůli označení. Na barvy se nelze spoléhat, různé kabely je mají jinak, viz kabel CA-42.

49.4.2. Kabel s PL2303HX na EBay

*

Odkazy:

- 2 x PL2303 USB/TTL/RS232 Serial Port Converter Module²⁵ u obchodníka cxw6712²⁶ na ebay
-

Tabulka 49-6. Zapojení kabelu

barva	signál
černá	GND
zelená	TxD
bílá	RxD
červená	+5V

Poznámky

1. http://www.faculty.iu-bremen.de/birk/lectures/PC101-2003/14usb/FINAL%20VERSION/usb_protocol.html
2. http://homepage.hispeed.ch/ibhdoran/usb_link.html
3. <http://www.beyondlogic.org/usb/usbhard.htm>
4. <http://www.usbmadesimple.co.uk/index.html>
5. <http://www.beyondlogic.org/usbnutshell/>
6. http://www.asalzadeh.netfirms.com/pdiusbd12_faq.htm

7. <http://www.dontronics-shop.com/4d-micro-usb.html>
- 8.
- 9.
- 10.
11. <http://www.dontronics-shop.com/ftdi-vnc11-t-r-vinculum-usb-host-controller-device.html>
12. <http://www.sxlist.com/techref/io/serial/usb/atapchi/index.html>
13. <http://www.sxlist.com/techref/scenix/lib/io/dev/keys/usbdemo-mh.htm>
14. http://www.microsln.com/msln/index.php?option=com_content&view=article&id=15:identificando-pines-del-data-cable-de-usb-a-serial
15. <http://www.uchobby.com/index.php/2009/10/04/diy-usb-to-serial-cable-for-3/>
16. http://buffalo.nas-central.org/index.php/Use_a_Nokia_Serial_Cable_on_an_ARM9_Linkstation
17. http://vikram.eggwall.com/computers/usb_cable/index.html
18. <http://www.nslu2-linux.org/wiki/HowTo/AddASerialPort>
19. http://pinouts.ru/CellularPhones-Nokia/nokia_pop_pinout.shtml
20. http://nokia-tuning.net/index.php?s=pinout_popport
21. <http://www.hardwarebook.info/Pop-Port>
22. <http://www.howardforums.com/showthread.php/832865-FAQ1-Nokia-Data-Connectivity-Cables-DKU-2-DKU-5-CA-42-CA-53>
23. http://nokia-tuning.net/index.php?s=pinout_popport
24. http://pinouts.ru/CellularPhonesCables/nokia_dku-5_cable_pinout.shtml
25. <http://cgi.ebay.co.uk/2-x-PL2303-USB-TTL-RS232-Serial-Port-Converter-Module-/220663968695>
26. <http://stores.ebay.co.uk/cxw6712>

Kapitola 50. Prostorová a orientační čidla

50.1. Kompas

- Finding The Way - HMC6352 Digital Compass¹
- Compass Module - HMC6352² na SparkFun

Poznámky

1. <http://www.nearfuturelaboratory.com/2009/03/14/finding-the-way-hmc6352-digital-compass>
2. http://www.sparkfun.com/commerce/product_info.php?products_id=7915

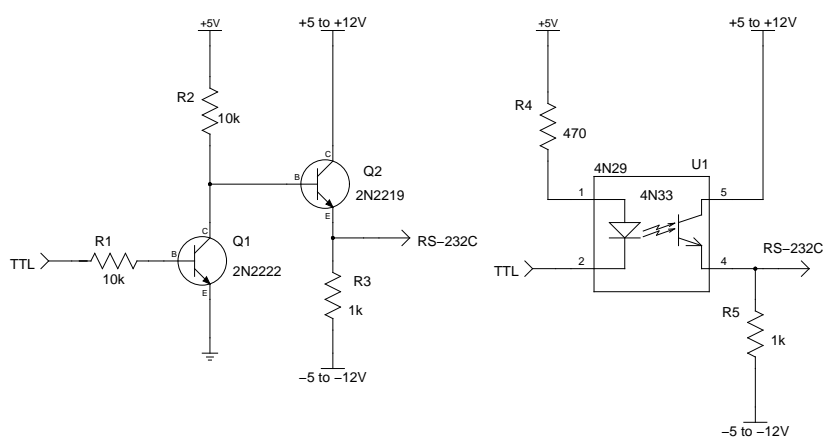
Kapitola 51. Sériová komunikace

51.1. RS232

Odkazy:

- RS-232¹ na české Wikipedii
- The RS232 STANDARD² A Tutorial with Signam Names and Definitions
- RS232 serial cables pinout³
- RS232 Data Interface⁴ a Tutorial on Data Interface and cables

Obrázek 51-1. Převodník TTL \longleftrightarrow RS232C s tranzistorů a optočlenu



* *images/geda/ciarcia_rs232c.sch, 800x600*

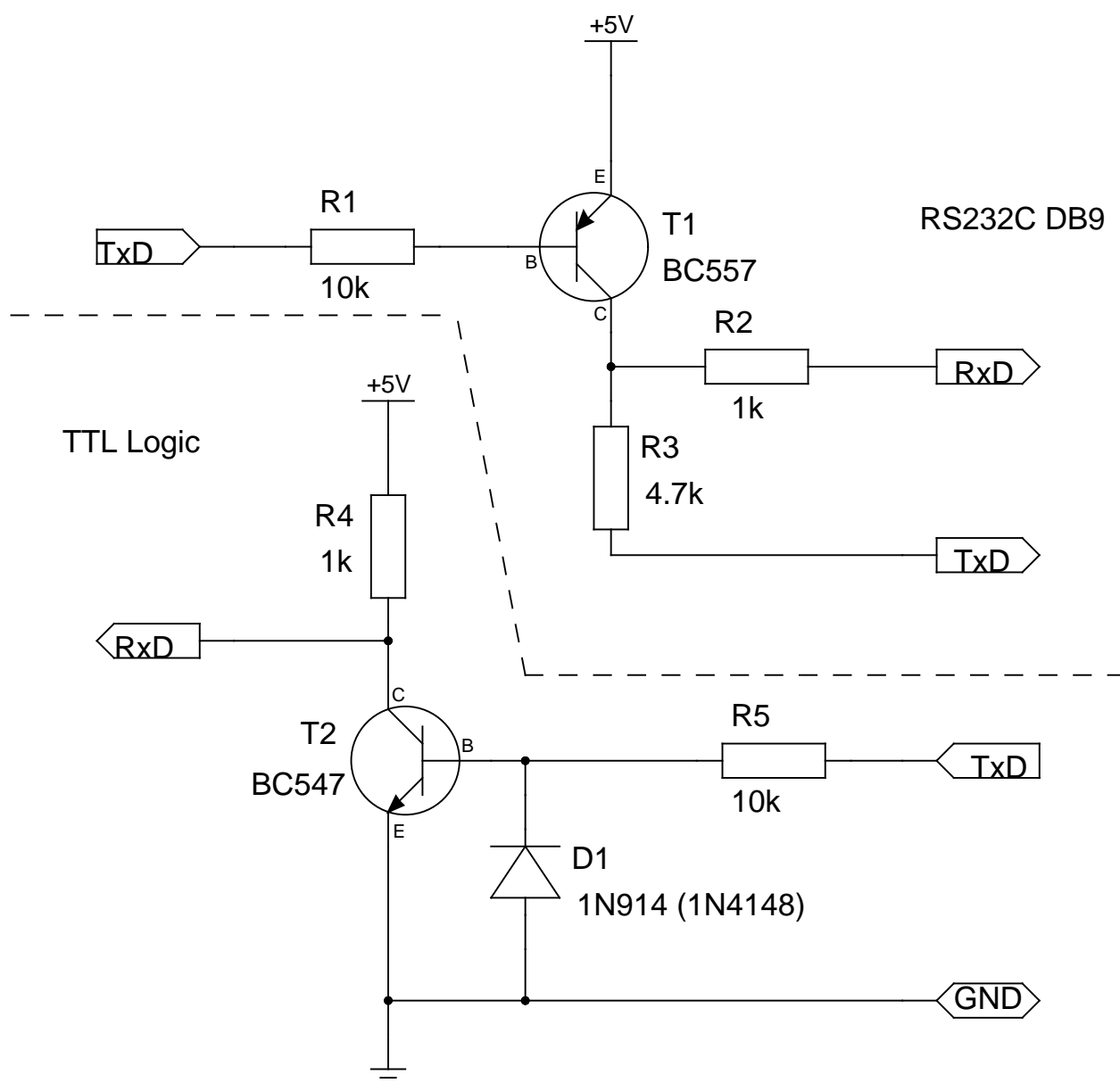
51.1.1. Převodník napěťových úrovní z tranzistorů

*

Odkazy:

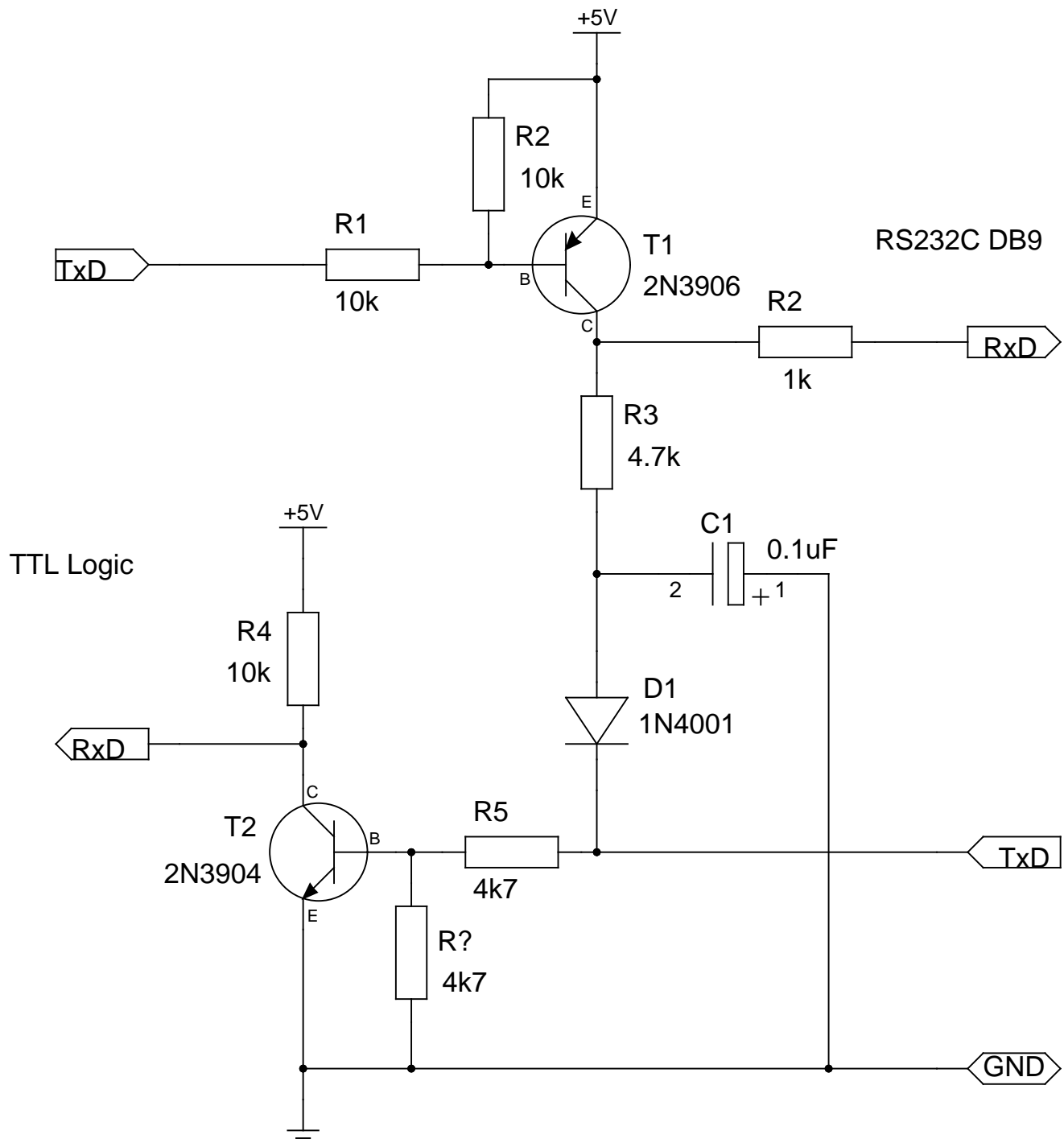
- Simple RS232C Level Converter using Transistors⁵
- TTL to RS232 adaptor Explained⁶
- Alternatives of MAX232 in low budget projects⁷
- Simple RS232 to logic level converter⁸ for PIC micros / TTL / CMOS
- TTL to RS232 Signal Conversion⁹
- RS232 Transceiver Circuits¹⁰
- RS232 Interface¹¹
- construction tips¹²
- RS232 transceiver circuits¹³

Obrázek 51-2. Jednoduchý převodník úrovní TTL↔RS232C



* images/geda/RS232C-level-converter, 640x480

Obrázek 51-3. Druhý jednoduchý převodník úrovní TTL↔RS232C



* *images/geda/RS232C-level-converter2, 640x480*

Další zapojení s tranzistory lze nalézt na konstrukci Single Sided Board¹⁴.

* *TODO: Překreslit v gschem!*

51.1.2. USB ↔ RS232-TTL sériový kabel

Odkazy:

- DIY USB to Serial Cable For \$3!¹⁵
- TTL-to-Serial for dummies (OpenWRT Serial Console)¹⁶
- Use a Nokia Serial Cable on an ARM9 Linkstation¹⁷
- How-To Convert a Cheap USB to Serial Cable For TTL¹⁸
-
-

Koupil jsem si kabel k telefonu CA-42 s tím, že jej podle návodu upravím na USB↔TTL pro Arduino.

Po zasunutí kabelu do počítače se v systémovém deníku objeví jak byl rozpoznán:

```
Aug  9 17:29:26 vanor kernel: [ 3796.986935] usb 2-2: new full speed USB device using uhci_hcd
Aug  9 17:29:26 vanor kernel: [ 3797.147235] usb 2-2: configuration #1 chosen from 1 choice
Aug  9 17:29:26 vanor kernel: [ 3797.149376] usb 2-2: New USB device found, idVendor=067b, idProduct=2303
Aug  9 17:29:26 vanor kernel: [ 3797.149382] usb 2-2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
Aug  9 17:29:26 vanor kernel: [ 3797.149387] usb 2-2: Product: USB-Serial Controller
Aug  9 17:29:26 vanor kernel: [ 3797.149390] usb 2-2: Manufacturer: Prolific Technology Inc.
Aug  9 17:29:26 vanor NetworkManager: <debug> [1281367766.561600] nm_hal_device_added(): New device
Aug  9 17:29:26 vanor kernel: [ 3797.509743] usbserial: USB Serial support registered for pl2303
Aug  9 17:29:26 vanor kernel: [ 3797.509743] pl2303 2-2:1.0: pl2303 converter detected
Aug  9 17:29:26 vanor kernel: [ 3797.521350] usb 2-2: pl2303 converter now attached to ttyUSB0
Aug  9 17:29:26 vanor kernel: [ 3797.521350] usbcore: registered new interface driver pl2303
Aug  9 17:29:26 vanor kernel: [ 3797.521350] pl2303: Prolific PL2303 USB to serial adaptor driver
```

Na zařízení jsem změřil který pin vede kterým kabelem.

Tabulka 51-1. Barvy vodičů

signál	barva
GND	oranžová
TX	červená
RX	modrá

51.2. RS422, TIA/EIA-422

Odkazy:

- TIA/EIA-422 Electrical Characteristics of Balanced Voltage Digital Interface Circuits¹⁹

51.3. RS485, EIA/TIA-485

Odkazy:

- RS-485 Bus²⁰
- RS-485 bus made from some DS3695's²¹
- Serial ports and making things²²
- EDAboard.com Forum Index -> Microcontrollers -> master/slave 8051²³

- Ten ways to bulletproof RS-485 interfaces²⁴ !!!
- THE ART AND SCIENCE OF RS-485²⁵ on Circuit Cellar
- DMX512
- Designing RS-485 Circuits²⁶ na Lakeview Research
- Basics of the RS-485 Standard²⁷
- Application Note 763: Guidelines for Proper Wiring of an RS-485 (TIA/EIA-485-A) Network²⁸
-
- RS485 card for gaming machines²⁹ [1997-10-18]
- Reduce Power In Your RS-485 Network³⁰ [2010-09-21]
- PDF Networking Topology (EIA/TIA/RS485)³¹ od Integrity Instruments Incorporated
-
-
-

Sériová sběrnice která pro přenos používá dvou (tři vodičů). Datový signál je přenášen symetricky po dvou vodičích, třetí vodič pak slouží jako společná zem. Dovoluje připojení až 32 zařízení (příjem/vysílání) na jeden segment. S použitím zařízení zvaných Repeaters (opakovače) je možno propojit více segmentů. Novější standard umožňuje za použití menších odporů připojení až 256 zařízení na společnou sběrnici (segment).

Maximální délka sběrnice je okolo 1200m a umožňuje přenosu rychlost do 200kbps. Nejvyšší rychlost je 10Mbps na sběrnici délky 50m.

Odkazy na součástky:

- - Maxim**
 - MAX3468/MAX3469 +5V, Fail-Safe, 40Mbps, Profibus RS-485 Transceivers³²
 - MAX3463/MAX3464 +5V, Fail-Safe, 20Mbps, Profibus RS-485 Transceivers³³
 - MAX3440E-MAX3444E +/- 15kV ESD-Protected, +/- 60V Fault-Protected, 10Mbps, Fail-Safe RS-485/J1708 Transceivers³⁴
 - MAX1480³⁵
 - MAX481,MAX483,MAX485,MAX487,MAX1487³⁶
 - Explanation of Maxim RS-485 Features³⁷
 - MAX489EPD,CPD (DIL14)³⁸
 - SN65HVD179 5B Full-Duplex RS-485/RS-422 DRIVER AND BALANCED RECEIVER³⁹
 - ADM1485 od Analog Devices⁴⁰
 - ADM485E/ADM487E/ADM1487R od Analog Devices⁴¹
 - DS96176 od National Semiconductor⁴²
 - LTC485⁴³
 - SN74LBC184⁴⁴
 - ST485⁴⁵
 -
 - Texas Instruments**
 - SN75176⁴⁶
 - SN75176BP v obchodě GES Electronics⁴⁷
 - LTC485IN8⁴⁸
 - DS3695⁴⁹
 - Analog Devices RS-485 chips⁵⁰

Tabulka 51-2.

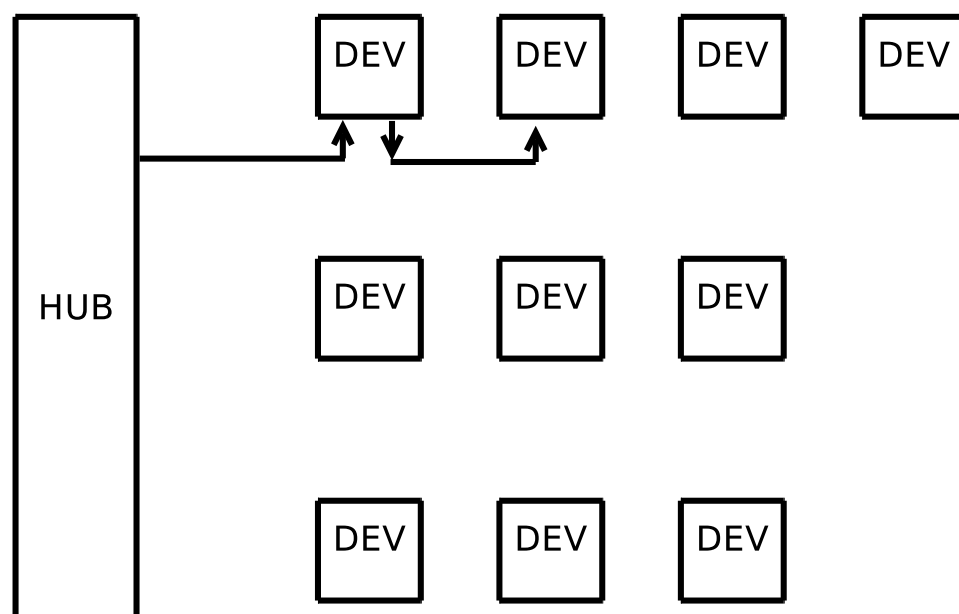
označení	zatížení	poznámka
SN75176	1	
ST485	1/8	

Zapojení:

- A good schematics⁵¹ using LTC485
- RS485 devices⁵² using SN75176 and PIC16F628
- RS485 Repeater⁵³
-
-

Obrázek 51-4.

Způsob propojení RS485 zařízení do sběrnice



51.4. MIDI

*

Odkazy:

- MIDI Information⁵⁴
- MIDI-interface for Arduino⁵⁵
- MIDI IN & OUT for the Arduino⁵⁶
- Arduino Midi Library⁵⁷
- CAN BUS: ARDUINO + MCP2515⁵⁸
-

51.5. CAN

* *Attributy:* id="CAN"

Odkazy:

- Controller area network⁵⁹
- MCP2515⁶⁰ — second generation stand-alone CAN controller
-
-

Poznámky

1. <http://cs.wikipedia.org/wiki/RS-232>
2. http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html
3. <http://www.lammerbics.nl/comm/cable/RS-232.html>
4. <http://www.arcelect.com/rs232.htm>
5. <http://www.kmitl.ac.th/~kswichit/ap275/ap275.htm>
6. <http://www.uchobby.com/index.php/2007/06/11/ttl-to-rs232-adaptor-explained/>
7. <http://www.scienceprog.com/alternatives-of-max232-in-low-budget-projects/>
8. <http://picprojects.org.uk/projects/simpleSIO/ssio.htm>
9. <http://techref.massmind.org/techref/io/serial/ttl2rs232.htm>
10. <http://www.circuitlake.com/rs232-transceiver-circuits.html>
11. http://www.botkin.org/dale/rs232_interface.htm
12. <http://www.elmelectronics.com/ictips.html>
13. http://www.coolcircuit.com/circuit/rs232_driver/
14. <http://arduino.cc/en/Main/ArduinoBoardSerialSingleSided>
15. <http://www.uchobby.com/index.php/2009/10/04/diy-usb-to-serial-cable-for-3/#more-492>
16. <http://neophob.com/2007/06/ttl-to-serial-for-dummies-openwrt-serial-console/>
17. http://buffalo.nas-central.org/index.php/Use_a_Nokia_Serial_Cable_on_an_ARM9_Linkstation
18. <http://sites.google.com/site/brandonselectronic/usbtottl>
19. http://www.interfacebus.com/Design_Connector_RS422.html
20. http://www.interfacebus.com/Design_Connector_RS485.html
21. <http://www.flickr.com/photos/cyberspace/3017168987/>
22. http://www.wirewd.com/make/blog/serial_ports/
23. <http://www.edaboard.com/viewtopic.php?t=197764&highlight=multipoint>
24. <http://www.edn.com/ednmag/archives/1996/080196/16df5.htm>
25. <http://www.circuitcellar.com/library/ccofeature/perrin0799/index.asp>
26. http://www.lvr.com/rs-485_circuits.htm
27. http://www.bb-elec.com/tech_articles/rs485_basics.asp
28. <http://www.maxim-ic.com/app-notes/index.mvp/id/763>
29. <http://www.rjcom.com/6811/jackpot/rs485-commspec.html>

30. <http://electronicdesign.com/article/design-solutions/Reduce-Power-In-Your-RS-485-Network.aspx>
31. <http://www.rs-485.com/download/485%20network%20topology.pdf>
- 32.
- 33.
- 34.
- 35.
- 36.
37. http://www.maxim-ic.com/appnotes.cfm/an_pk/367
- 38.
- 39.
- 40.
- 41.
- 42.
- 43.
44. <http://www.flickr.com/photos/35089060@N00/166469510/>
- 45.
- 46.
47. <http://www.ges.cz/?page=index&gesid=GES05000610>
- 48.
- 49.
50. <http://www.analog.com/en/interface/rs-485/products/index.html>
51. http://www.mikroe.com/pdf/rs485_board_schematic.pdf
52. <http://www.picbasic.co.uk/forum/attachment.php?attachmentid=2388&d=1204738043>
53. <http://www.picbasic.co.uk/forum/showthread.php?t=10136&s=0029d598831673d7a747b47d999f77e9>
54. http://tomscarff.110mb.com/midi_info.htm
55. http://www.mrstockinterfaces.com/groups/wiki/wiki/99af5/Arduino___Microcontroller_Workshop.html
56. http://tomscarff.110mb.com/MIDI_IN_OUT_ARDUINO/midi_in_out_arduino.htm
57. http://timothywillman.com/itp_blog/?page_id=240
58. <http://real2electronics.blogspot.com/2010/06/can-bus-mcp2515.html>
59. http://en.wikipedia.org/wiki/Controller_area_network
60. <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010406>

Kapitola 52. Síťová komunikace

*

52.1. Ethernet

*

52.1.1. ENC28J60

Odkazy:

- ENC28J60¹ na stránkách Microchip
- ENC28J60² na české Wikipedii
-
-
-

Fakta/Vlastnosti:

- RAM Buffer 8192B
- obsluha dvou LED
- napájecí napětí 3.3V
- Hodinový kmitočet 25MHz
- Rozhraní směrem k MCU je SPI
- Programovatelná MAC adresa, MAC adresa není defaultně nastavena.
- Ethernet rozhraní 10Base-T
- Je k dispozici varianta v pouzdře DIL.
-
-

Obvod ENC28J60 výrobce Microchip je ethernetový řadič k použití s jednoduchými MCU. K připojení MCU k ethernetu stačí tento obvod, ethernetová zásuvka s vestaveným transformátorem, krystal a několik pasivních součástek.

BreakUp boardy, Bread Board komponenty a moduly:

- ENC28J60-H WORLD SMALLEST ENC28J60 ETHERNET CONTROLLER DEVELOPMENT BOARD³ — od firmy Olimex
- SPINET - Ethernet ↔ SPI⁴
- BOB-00765⁵ — Ethernet Interface Board - ENC28J60 na Sparkfun
-
-
-

Konstrukce a návody:

- <http://members.home.nl/bzijlstra/software/examples/enc28j60.htm>
- An AVR microcontroller based Ethernet device⁶
- A watering controller that can be home networked⁷ na Instructables
- A credit card sized Ethernet Arduino compatible controller board⁸ na Instructables
- A Remotely Programmable Relay Controller (Christmas Lights or Home Automation Controller)⁹
-
-

•

Pokud sháním RJ45 zásuvku k ENC28J60, hledáme "RJ45 filter", se zabudovaným transformátorem. Další vyhledávací řetězec je "RJ45 magnetics".

* 08B0-1X1T-06-F: AutoMDIX Ethernet Magjack with bright LEDs and magnetics¹⁰

Co všechno potřebuju. Tedy seznam všech součástek k vytvoření ethernetového rozhraní:

Tabulka 52-1.

název	počet	kód Farnell	poznámka
ENC28J60-I/SP	1	1564402	
ethernet konektor s transformátorem, RJMG163218101NR	1	1357435 ₁₁	
STEWART CONNECTOR SI-60062-F		1137983 ₁₁	
odpory 49.9Ω 1%	4	9468609 ₁₁	min 50ks
odpory 23.2kΩ 1%	1	9466746 ₁₁	min 50ks

* Hledání konektorů s transformátorem. Zkouším hledat řetězec "RJ45 filtered". Musím dále vyhodnotit tyto:

* RJ45-8N3-B od TYCO (1126569) — žádné informace o transformátoru, je li nějaký.

* RJMG201022610NR od AMPHENOL (1357422) — gigabit, filtry na všech párech.

* RJ45-8N-S od TYCO ELECTRONICS / CORCOM () —

* RJMG185118101LR od AMPHENOL (1357442) — vypadá dobře ale je SMT.

* RJMG163218101NR od AMPHENOL (1357435) —

* RJMG163217101NR

* RJMG163118101NR

Tabulka 52-2. Dostupnost ENC28J60

typ	obchod/prodejce	datum	cena	sklad
ENC28J60-I/SP (28SDIP)	Farnell	2010-09-16	82.78	1017
ENC28J60/SP (28SDIP)	Farnell	2010-09-16	74.88	16

52.2. Modbus

*

Odkazy:

- Modbus¹¹ na Wikipedii
- Simply Modbus¹²
- modbusmaster¹³ — Arduino class library for communicating with Modbus slaves over RS232/485
-

Tabulka 52-3.

Register	Data	Type	Table
1-9999	0000 - 270E	RW	Discrete Output
10001-19999	0000 - 270E	RO	Discrete Input
30001-39999	0000 - 270E	RO	Analog Input
40001-49999	0000 - 270E	RW	Analog Output

Tabulka 52-4. Kódy funkcí

Funkce	Akce	Název tabulky
01 (0x01)	Read	Discrete Output Coils
05 (0x05)	Write single	Discrete Output Coil
15 (0x0F)	Write multiple	Discrete Output Coils
02 (0x02)	Read	Discrete Input Contacts
04 (0x04)	Read	Analog Input Registers
03 (0x03)	Read	Analog Output Holding Registers
06 (0x06)	Write single	Analog Output Holding Register
16 (0x10)	Write multiple	Analog Output Holding Registers

52.3. Profibus

*

Odkazy:

- Profibus¹⁴ na Wikipedii
-

Poznámky

1. <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en022889>
2. <http://cs.wikipedia.org/wiki/ENC28J60>
3. <http://www.olimex.com/dev/enc28j60-h.html>
4. http://pandatron.cz/?170&spinet_-_ethernet_-_spi
5. http://www.sparkfun.com/commerce/product_info.php?products_id=765
6. <http://www.tuxgraphics.org/electronics/200606/article06061.shtml>
7. <http://www.instructables.com/id/A-watering-controller-that-can-be-home-networked/>
8. <http://www.instructables.com/id/A-credit-card-sized-Ethernet-Arduino-compatible-co/>
9. <http://www.instructables.com/id/A-Remotely-Programable-Relay-Controller-Christmas/>
10. http://shop.tuxgraphics.org/electronic/detail_magjack.html
11. <http://en.wikipedia.org/wiki/Modbus>
12. <http://www.simplymodbus.ca/index.html>

13. <http://code.google.com/p/modbusmaster/>

14. <http://en.wikipedia.org/wiki/Profibus>

Kapitola 53. Záznamová média

*

53.1.

*

Odkazy:

- Papertape¹
-

Poznámky

1. <http://homepages.cwi.nl/~dik/english/codes/punched.html>

Kapitola 54. Rozhraní pro videosignál

*

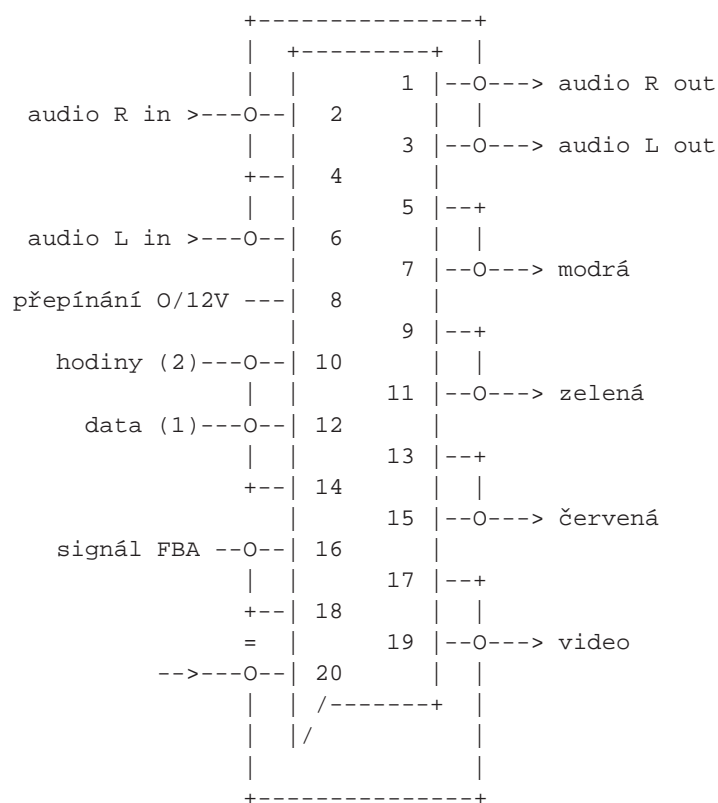
54.1. SCART

*

Odkazy:

- Ročenka sdělovací techniky 1989, strana 203
- SCART¹ v obrázcích na Google
-

Obrázek 54-1. Zapojení konektoru scart



FBA

Farb Bildinhalt Austast Signal. Kompozitní videosignál.

červená

zelená

modrá

Barvové signály. Maximální amplituda signálů je 1V. Vstupní impedance barevných signálů je 76 Ω

Poznámky

1. <http://www.google.cz/images?q=SCART>

Kapitola 55. Zvukové generátory a syntezátory

Odkazy:

- Sound generators of the 1980s home computers¹
-
-

55.1. Programovatelné zvukové generátory (PSG)

*

Další obvody:

- Yamaha YM2149
-
-
-

55.1.1. AY-3-8910

Odkazy:

- General Instrument AY-3-8910²
-
-
-

Tento zvukový obvod se objevil v době osmibitových počítačů. V řadě z nich byl použit. Například v novějších variantách ZX-Spectrum.

Obvody je zapouzdřený v pouzdře DIL-40. Součástí obvodu jsou dvě osmibitové paralelní brány A a B.

Později se objevily varianty tohoto obvodu:

- AY-3-8912 — Vypuštění paralelní brány B umožnilo snížit počet vývodů a obvod je v pouzdře DIL-28.
- AY-3-8913 — Vypuštěny oba dva paralelní porty, obvod je v pouzdře DIL-24
- Yamaha YM2149 (SSG) — obvod je zapojen stejně jako AY-3-8910 s několika malými rozdíly.
- AY-3-8914 — varianta obvodu se změnami pro Mattela
- AY-3-8930 — vylepšená varianta obvodu AY-3-8910

55.1.2. SN76489

*

Odkazy:

- Texas Instruments SN76489³ na Wikipedii
- SN76489 sound chip details⁴
- SN76489⁵ na SMS POWER
-

55.2. FM a OPN

Odkazy:

- Yamaha YM2203⁶ OPN (FM Operator Type N) 6-ti kanálový zvukový čip
- Yamaha YM2608⁷ 16-ti kanálový zvukový čip s FM syntézou
- Yamaha YM2612⁸ 6-ti kanálový zvukový čip, vyráběný také v CMOS verzi jako YM3438
- Yamaha YM2610⁹ — 15-ti kanálový zvukový čip (4 FM kanály, 3 SSG kanály, 1 šumový kanál, 6 ADPCM 18.5kHz, jeden ADPCM 1.8-55.5kHz, dva čítače, LFO
-
-
-

Poznámky

1. <http://www.srcf.ucam.org/~rga24/computer/music/>
2. http://en.wikipedia.org/wiki/General_Instrument_AY-3-8910
3. http://en.wikipedia.org/wiki/Texas_Instruments_SN76489
4. <http://web.inter.nl.net/users/J.Kortink/home/articles/sn76489/>
5. <http://www.smspover.org/Development/SN76489>
6. http://en.wikipedia.org/wiki/Yamaha_YM2203
7. http://en.wikipedia.org/wiki/Yamaha_YM2608
8. http://en.wikipedia.org/wiki/Yamaha_YM2612
9. http://en.wikipedia.org/wiki/Yamaha_YM2610

Kapitola 56. Ostatní rozhraní a sběrnice

*

Tento oddíl je zatím jen prázdným výčtem kapitol a tak slouží jen k velmi základní orientaci.

* *Sloučit tuto část s předchozí částí.*

Odkazy:

- Serial buses information page on ePanorama.net¹

56.1. SPI

* *Attributy: id="SPI"*

Odkazy:

- Rozhraní SPI² povídání na meloun.eSportuj.cz
-

56.2. Ethernet

56.3. 1 Wire

56.4. TWI, I²C

I²C, některými výrobci nazývané TWI je rozhraní kterým se připojují periferní obvody s pomocí dvou vodičů. (společnou zemi nepočítám). Jeden signál se nazývá SCL a druhý SDA. SCL je hodinový signál, který generuje ten obvod na sběrnici který má funkci master. SDA je obousměrný datový signál. Data se přenáší po oktetech (skupina 8-mi bitů). Adresování periférií je v prvním bajtu přenosu. I²C periferní obvody mají část adresního pole nastaveno na pavno od výrobce a část adresy je možno obvykle nastavit na několika vývodech obvodu. To nám umožní mít ke sběrnici připojeno několik identických obvodů.

Sběrnice I²C je tzv multimaster. To znamená, že k ní může být připojeno několik master obvodů, které používají společné periferie. Pokud chcete používat tuto vlastnost I²C, nastudujte si chování všech čipů ke sběrnici připojených.

56.5. BlueTooth

56.6. DMX512

Odkazy:

- DMX512³ na Wikipedii

- Welcome to the DMX-512 mini-FAQ⁴
- WHAT IS DMX512?⁵
- OpenDMX.net⁶
- OLA: An open framework for DMX lighting control⁷
- rdmx⁸ knihovna pro Ruby
- DMX512: část 1. - digitální výstupy⁹
- Řízení světelných efektů hudebním signálem¹⁰ Jaroslav Nušl, projekt.cvut.org
- DMX512 přijímač - je nutné pro UART pozměnit DMX512 paket?¹¹

DMX512 je sériová sběrnice původně pro ovládání světel a efektových zařízení na scénách. Na hardwarové úrovni je realizována jako EIA-485 a s použitím konktorů XLR.

Komunikace po sběrnici je jednosměrná. Je jeden pán sběrnice a zbytek jsou ovládaná zařízení. Je možno ovládat/přenášet data pro maximálně 512 kanálů, odtud číslo v názvu sběrnice.

Protokolově je přenos realizován tak, že po odvysílání startovací sekvence obsahující "Start Code" který určuje funkci. Pak následuje maximálně 512 bajtů hodnot pro jednotlivé kanály. Každé zařízení/kanál pozná hodnotu určenou pro něj podle pozice vzhledem k startovací sekvenci.

Tabulka 56-1. Startovací kódy

0x00	NULL	
0x17		Text packets
0xCF		System Information Packets
0xCC	RDM	RDM extension

Poznámky

1. <http://www.epanorama.net/links/serialbus.html>
2. <http://meloun.esportuj.cz/index.php?text=12-rozhrani-spi>
3. <http://en.wikipedia.org/wiki/DMX512>
4. <http://www.alia.com.au/links/dmx-faq.htm>
5. <http://www.dmx512-online.com/whats.html>
6. <http://www.opendmx.net/index.php/OpenDMX.net>
7. <http://code.google.com/p/linux-lighting/>
8. <http://github.com/heisters/rdmx>
9. <http://www.mcu.cz/news.php?extend.1026.2>
10. <http://projekt.cvut.org/>
11. <http://forum.mcontrollers.com/viewtopic.php?t=1757>

Kapitola 57. Připojený výkonných motorických prvků

*

57.1. Ovládání servomotorů

*

Odkazy:

- 68.2
- How RC Servos Works¹
- Drive 10 Servos using only 2 Arduino Pins² — Pete Burniht [2010-11-28]
-

Poznámky

1. http://pcbheaven.com/wikipages/How_RC_Servos_Works/
2. <http://diydrones.com/profiles/blogs/drive-10-servos-using-only-2>

Kapitola 58. Analog Digital převodníky

*

Jedná se o zařízení jimž na vstup přivedeme analogovou hodnotu ve formě napětí či proudu a na výstupu dostaneme číselnou hodnotu. V řadě mikrořadičů je taková jednotka přímo integrována do řadiče. Můžeme jej sestavit z diskretních komponent, nebo můžeme použít specializované čipy, tzv. AD převodníky.

Čipy:

- LTC2400¹ — 24-bit ~~textmu~~Power ADC,
- MCP3201² — 12-ti bitový, 1 kanálový s rozhraním SPI, 100k vzorků za sekundu, nízký odběr 0.5mA, pouzdro DIP8. [2011-04-13, TME, 83.03Kč]
- MCP3202³ — 12-ti bitový, 2 kanálový s rozhraním SPI, 100k vzorků za sekundu, nízký odběr 0.5mA, pouzdro DIP8. [2011-04-06, Farnell, 83.03Kč]
- MCP3204/MCP3208⁴ — varianty MCP3202 se 4-mi / 8-mi vstupy, pouzdro DIP14/DIP16. [2011-04-06, Farnell, cena cca 110/127 Kč]
- MCP3001, MCP3004, MCP3008⁵ — 10 bitový, 1/4/8 kanálový
-

Konstrukce:

- Connect a LTC2400 High Precision 24 Bit Analog to Digital Converter⁶ — Lab3
- Propellor Analog Input using MCP 3202s (12Bit 2channel)⁷
-
-
-

Poznámky

- 1.
- 2.
- 3.
- 4.
- 5.
6. <http://interface.khm.de/index.php/lab/experiments/connect-a-ltc2400-high-precision-24-bit-analog-to-digital-converter/>
7. <http://letsmakerobots.com/node/23713>

IX. Konstrukce, většinou amatérské

Part Intro.

Kapitola 59. Komponenty

59.1. Experimentální PCB

Odkazy:

- Universal through-hole and SMD prototyping board¹ [2009-02-20]

Poznámky

1. <http://www.whitewing.co.uk/protoboard.html>

Kapitola 60. Doma dělané a zvláštní počítače či procesory

Někteří lidé se rozhodnou, že si navrhnu a postaví počítač od základů sami. Někdy dokonce bez použití mikroprocesorů, či dokonce jen s použitím velmi jednoduchých součástek.

Následující odkazy jsou na stránky nadšenců kteří se zabývají konstrukcí počítačů.

Odkazy:

- Homebrew one-bit microcomputer¹
- SIX BIT MACHINE² by Jack Eisenmann
- DIY Calculator³ — More Cool Stuff
- Kniha Bit-Slice Design: Controllers and ALUs⁴ by Donnamaie E. White
- DIY Calculator⁵ ???
- Homemade computers⁶ — seznam odkazů
-

Mechanické konstrukce:

- LEGO Logic Gates⁷ — mechanická hradla z lega.
- Rube Goldberg machine⁸ — vysvětlení termínu na Wikipedii
- Heath Robinson Rube Goldberg Computer⁹
- Two Types of Mechanical Reversible Logic¹⁰
- YouTube video K'nex Adding Machine¹¹
-

Réleové počítače a konstrukce:

- KoP Calculator¹²
- Harry Porter's Relay Computer¹³
- Der Relaisrechner - Dokumentation¹⁴ [německy]
- Harry Porter's Relay Computer¹⁵ releový počítač
- www.relaiscomputer.de¹⁶ Informationen zu den Relaisrechnern von Kilian Leonhardt
- Zusie - My Relay Computer¹⁷

Konstrukce z tranzistorů:

- Dieterovy konstrukce MT15 a TREX¹⁸
- 4 bit computer¹⁹ — přímý odkaz na YouTube²⁰ (4 bitová sčítačka)

Konstrukce z integrovaných obvodů:

- Homebrew CPU Magic 1²¹
- DUO 128 ELITE²², DUO Elite Architecture Description²³ [2010-05-30]
- QR6: The Quintessential RISC²⁴
- heritage/1²⁵

S použitím různých verzí programovatelných logických obvodů:

-
-

Mikroprocesorové a mikropočítačové konstrukce:

- NP1 - MCU matrioška²⁶
- William's Homemade Computer²⁷ (486DX@20MHz)
- Breadboard a Computer²⁸ [6504]
- Homebrew 8080 microcomputer²⁹

•

The Heath Robinson Rube Goldberg Computer:

- Part 1: Implementing a computer using a mixture of technologies from relays to fluidic logic³⁰
- Part 2: Partitioning the system³¹
- Part 3: Introducing the HRRG emulator³²
- Part 4: The battle to make the virtual cabinets work³³

60.1. Kenbak-1

Odkazy:

- Kenbak-1³⁴
- KENBAK-1 Computer³⁵

60.2. Mark-8

Odkazy:

- The Mark-8 Minicomputer³⁶

Konstrukce postavená na procesoru Intel 8008.

60.3. EDUC-8 Microcomputer

* *Attributy: id="EDUC-8"*

Odkazy:

- Popis konstrukce EDUC-8 Microcomputer³⁷
- Dokumentace The EDUC-8 Microcomputer³⁸ Lawrence Wilkinson
- The Electronics Australia EDUC-8 microcomputer³⁹
- Educ-8⁴⁰ na Wikipedii

Mikropočítač, inspirovaný architekturou PDP-8 postavený z TTL obvodů. Použity jsou obvody série 7400 a 9000. První „hoby“ počítač postavený v Austrálii.

Architektura tohoto počítače a instrukční sada vychází z PDP-8. Ale protože má instrukční slovo jen 8 bitů, je jednodušší. K dispozici jsou jen dvě periferní zařízení a to sériové porty. Celková velikost operační paměti je 256 slov, každé 8 bitů veliké.

Konstrukce byla publikována v sérii článků v časopise Electronics Australia od srpna 1974 do srpna 1975.

Fakta:

- Paměť programu a dat 256 slov 8 bitů širokých.
- Postaven z obvodů TTL séri 74 nebo 9000. Obsahuje 98 obvodů.
- Paměť sestavena ze statických RAM čipů 1K * 1bit.
- Hodinový kmitočet 500kHz. Vykoná asi 50tisíc instrukcí za sekundu.
- Sériové zpracování.
- It was designed by Jim Rowe
- construction details were published in Electronics Australia magazine between August 1974 and January 1975. Additional peripherals were described in further articles up to August 1975.

Počítač má stejné registry jako PDP-8, jen registr L není implementován.

Instrukční slovo sestává ze tří polí. První pole o šířce tří bitů kóduje instrukci. K dispozici je 6 instrukcí s adresou, jeden kód 110 pro komunikaci s periferiemi a jeden kód 111 pro dvě skupiny mikroinstrukcí. Druhé pole o šířce jeden bit je příznak nepřímého adresování. Poslední čtyřbitové pole obsahuje adresu.

Z pohledu programátora je operační paměť rozdělena na 16 stránek každá má 16 slov. Adresa v instrukci se odkazuje na paměť ve stejné stránce jako je samotná instrukce.

Obrázek 60-1. Formát instrukčního slova počítače EDUC-8

7	Opcode	5	Ind	4	3	Address	0
---	--------	---	-----	---	---	---------	---

Obrázek 60-2. Přehled instrukcí počítače EDUC-8

AND	7	0 0 0	5	Ind	4	3	Address				0		
TAD	7	0 0 1	5	Ind	4	3	Address				0		
ISZ	7	0 1 0	5	Ind	4	3	Address				0		
DCA	7	0 1 1	5	Ind	4	3	Address				0		
JMS	7	1 0 0	5	Ind	4	3	Address				0		
JMP	7	1 0 1	5	Ind	4	3	Address				0		
IOT	7	1 1 0	5	I/O	4	Dev	3	Clr	2	Sh	1	Skp	0
OPR	7	1 1 1	5	0	4	CLA	3	CMA	2	RAL	1	IAC	0
OPR	7	1 1 1	5	1	4	SZA	3	SMA	2	RAR	1	HLT	0

60.3.1. Ukázkové programy

*

Počítadlo.

```

START,  LA          / clears AC
INCR,   IAC          / increments AC
BACK,   NOP          / delay
        NOP          / delay
        ISZ INDX     / loop on INDX
        JMP BACK     / if INDX non-zero
        ISZ INDY     / loop on INDY
        JMP INCR     / if INDY non-zero
        HLT          / INDY is zero
INDX,   0             / Delay counter
INDY,   0             / Follows AC

```


60.4. BMOW 1 (*Big Mess o' Wires 1*)

* *Attributy: id="BMOW"*

Odkazy:

- BIG MESS O' WIRES⁴¹ A HOME-BUILT CPU, AND OTHER MESSY ELECTRONICS ADVENTURES
-

Jedná se o originální procesor postavený jen z běžných logických obvodů řady 7400. Procesor má 8-mi bitovou datovou sběrnici a 24 bitovou adresovou sběrnici. Čtyři osmibitové registry pro obecné použití a tři 24 bitové registry: čítač programu, ukazatel zásobníku a pracovní registr. Instrukční sada vychází z instrukcí procesoru 6502.

60.5. Turing Machine

Odkazy:

- A Turing Machine In The Classic Style⁴²
-
-
-

60.6. AppleCrate II

*

Odkazy:

- AppleCrate II: A New Apple II-Based Parallel Computer⁴³
-

"Masivně" paralelní počítač stvořený propojením 17 základních desek Apple IIe.

Poznámky

1. <http://userwww.sfsu.edu/~hl/c.homebrew2.html>
2. <http://web.mac.com/teisenmann/sixbit/main.html>
3. <http://www.diycalculator.com/cool.shtml>
4. http://www10.edacafe.com/book/parse_book.php?article=BITSLICE/bitslcP.html
5. <http://www.diycalculator.com/>
6. <http://www.holmea.demon.co.uk/Links.htm#Homemade>
7. <http://goldfish.ikaruga.co.uk/logic.html>
8. http://en.wikipedia.org/wiki/Rube_Goldberg_machine
9. <http://www.diycalculator.com/sp-hrrgcomp.shtml>
10. <http://www.zyvex.com/nanotech/mechano.html>
11. <http://www.youtube.com/watch?v=3vXIQZvS-nM>
12. <http://www.carobotics.org/projects/calculator/>

13. <http://web.cecs.pdx.edu/~harry/Relay/>
14. <http://www.schlaefendorf.de/relaisrechner/dokumentation/index.html>
15. <http://web.cecs.pdx.edu/~harry/Relay/index.html>
16. <http://www.relaiscomputer.de/>
17. <http://www.nablaman.com/relay/>
18. <http://6502.org/users/dieter/>
19. http://www.waitingforfriday.com/index.php/4-Bit_Computer
20. <http://www.youtube.com/watch?v=xISG4nGTQYE>
21. <http://www.homebrewcpu.com/>
22. <http://web.mac.com/teisenmann/iWeb/elitepage/menu.html>
23. <http://www.youtube.com/watch?v=a91XTeh8M5c>
24. <http://www.efton.sk/qr6/index.htm>
25. <http://www.armandoacosta.com/cpu/index.php>
26. <http://www.mcu.cz/news.php?extend.1377.2>
27. <http://dubel.org/computer/>
28. <http://mysite.du.edu/~jcalvert/tech/6504.htm>
29. <http://userwww.sfsu.edu/~hl/c.homebrew1.html>
30. <http://www.ibm.com/developerworks/power/library/pa-hrrg1/index.html>
31. <http://www.ibm.com/developerworks/power/library/pa-hrrg2/index.html>
32. <http://www.ibm.com/developerworks/power/library/pa-hrrg3/index.html>
33. <http://www.ibm.com/developerworks/power/library/pa-hrrg4/index.html>
34. http://www.bytecollector.com/kenbak_1.htm
35. <http://www.kenbak-1.net/index.htm>
36. <http://web.archive.org/web/20040216021509/http://www.his.com/~jlewczyk/adavie/mark8b.html>
37. <http://www.sworld.com.au/steven/educ-8/index.html>
38. <http://www.ljw.me.uk/educ8/educ8.html>
39. <http://www.ljw.me.uk/educ8/>
40. <http://en.wikipedia.org/wiki/Educ-8>
41. <http://www.stevechamberlin.com/cpu/bmow1/>
42. <http://aturingmachine.com/>
43. <http://home.comcast.net/~mjmahon/AppleCrateII.html>

Kapitola 61. Konstrukce s CPU

*

Konstrukce založené na známých mikroprocesorech.

61.1. Intel 8080

*

Odkazy:

- Nostalcomp 8080¹
-

Poznámky

1. <http://www.nostalcomp.cz/nostal8080.php>

Kapitola 62. Repliky

62.1. PDP-2

Odkazy:

- The PDP-2 processor¹ — ve skutečnosti není replika ale fantazie, DEC nikdy nevyprojektoval ani nevyrobil PDP-2
-

Poznámky

1. <http://locl.net/homes/pdp2/>

X. Vybrané architektury

Bližší informace o některých počítačových architekturách. Tyto informace jsou natolik rozsáhlé, že jsem jim vyhradil zvláštní díl.

Kapitola 63. PDP-5, PDP-8, IC6100, HD-6120

Architektura rodiny 12-ti bitových minipočítaču firmy DEC.

* Všechny instrukce jsou značeny kódem `pdp8.isa`. Je to proto, že původně jsem začal s popisem minipočítače PDP-8 a pak tento je nejznámějším členem této rodiny.

Odkazy:

- The Digital Equipment Corporation PDP-8¹
- PDP-8² na Wikipedii
- PDP-8/X³
- Computer Structures: Principles and Examples⁴
- <http://www.grc.com/pdp-8/isp-musings.htm>
-

Odkazy:

- Programming Languages (for the PDP-8)⁵
-

Rodina 12-ti bitových počítačů firmy DEC.

Další z 12-ti bitových počítačů. Paměť počítače byla velká 4K 12-ti bitových slov. V pozdějších verzích byl CPU rozšířen o možnost práce s 8*4K slov (32K slov).

Jeden z nejrozšířenějších minipočítačů své doby.

Přehled jednotlivých modelů:

Tabulka 63-1. Přehled jednotlivých modelů PDP-8

model		
PDP-5		předchůdce PDP-8
PDP-8		první model PDP-8
PDP-8/S		zjednodušený (sériový) model, pomalý
PDP-8/I		použity obvody TTL
pdp-8/e		
pdp-8/f		
pdp-8/m		

63.1. Jednotlivé modely počítačů

Odkazy:

- PDP-8 Summary of Models and Options⁶
- PDP-8 Frequently Asked Questions⁷

Tato část je přehledem různých modelů počítačů 12-ti bitové rodiny minipočítačů firmy DEC. Prvním minipočítačem byl typ PDP-5, na kterém byly vymodelovány základní principy. Minipočítačů PDP-5 se mnoho neprodalo, podle informací které se mi podařilo sehnat to bylo asi 116 kusů (PDP-8 Frequently Asked Questions⁸).

63.1.1. PDP-5

Odkazy:

- PDP-5⁹
-

Fakta:

- Předvedeno 1963-08-11 na WESCONu.
- Výroba ukončena začátkem roku 1967.
- Celkem vyrobeno 116 kusů.
- Cena: \$27 000.

První počítač 12-ti bitové architektury firmy DEC. Má základní charakteristiky této 12-ti bitové architektury jako je instrukční sada.

Počítač byl sestaven z modulů které firma DEC vyráběla. Jednalo se o logické moduly sestavené z transistorů a dalších jednoduchých komponent. Napájecí napětí byla dvě, +10 a -15 voltů. Logické úrovně 0 (0 V) a 1 (-3V).

Inspirací pro práci na tomto počítači byl úspěch CDC-160, 12-ti bitového minipočítače od Seymoura Craye, a úspěch počítače LINC postaveného zákazníky fy DEC z tranzistorových logických modulů které firma DEC prodávala. Tyto úspěchy ukázaly volné nezaplněné místo na trhu, které se firma DEC pokusila zaplnit modelem PDP-5 se zaváděcí cenou \$27 000.

Ken Olson řekl že PDP-5 nebyl zamýšlen jako počítač. Byl navrhován pro společnost která potřebovala automatické řízení průmyslového provozu. Ken jim řekl, že pro ně mohou udělat programovatelné řízení místo pevné zadrátované logiky o kterou původně požádali. Zákazník se nebyl jist rovnicemi podle nichž mělo být zařízení konstruováno a tak akceptovali myšlenku programovatelného řadiče. Výsledkem byl počítač PDP-5.

S nástupem počítače PDP-8 jenž byl výkonnější a byl levnější byla ukončena výroba PDP-5.

Všechny počítače rodiny vychází ze stejných základů, a tak jsou do jisté míry kompatibilní. Určité odlišnosti tady ale jsou. Počítač PDP-5 nemě čítač instrukcí jako samostatný registr, ale pro tento účel sloužilo paměťové slovo v hlavní paměti počítače na adrese 0000. Také přerušení je zpracováváno odlišně než u PDP-8. Instrukce IAC a CMA ve skupině 1 μ instrukcí OPR1 nemohou být kombinovány s μ instrukcemi pro rotaci RAR, RAL, RTR a RTL.

Obsluha přerušení funguje tak, že nejprve uloží obsah čítač instrukcí na adresu 0001 v paměti, a poté začne vykonávat program obsluhy přerušení který začíná na adrese 0002.

Tabulka 63-2.

adresa	symbol	popis
0000	PC	
0001	SAVEPC	Zde je uchována hodnota PC při přerušení.
0002	INT	Na této adrese je první instrukce obsluhy přerušení.
0010-0017		Při nepřímém adresování pomocí těchto adres je jejich obsah zvětšen o 1.

Počítač se standardně dodával se pamětí velikou 1KiW nebo 4KiW.

K počítači bylo možno připojit EAE (Extended Arithmetic Element) který se ovládal přes IOT instrukce. Toto rozšíření tím pádem není kompatibilní s EAE u dalších počítačů jako PDP-8.

Obrázek 63-1. Formát instrukčního slova PDP-5

PDP-5 Instrukce	Op Code	IA	P	In Page Address
	0 2	3	4	5 11

Tabulka 63-3. Soubor základních instrukcí PDP-5

kód	název	popis
0YYY	AND Y	
1YYY	TAD Y	
2YYY	ISZ Y	
3YYY	DCA Y	
4YYY	JMS Y	
5YYY	JMP Y	
6ddf	IOT DF	
7MMM	OPR M	

Obrázek 63-2. Formáty OPR μ instrukcí PDP-5

Skupina OPR1	1 1 1	0	CLA	CLL	CMA	CML	RAR	RAL	0	IAC
	0 2	3	4	5	6	7	8	9	10	11
	0 2	3	4	5	6	7	RTR 8	RTL 9	1 10	 11

Skupina OPR2	1	1	1	1	1	1	0	0	0	0
	0	2	3	4	5	6	7	8	9	10
	0	2	3	4	5	6	7	8	9	10

Obrázek 63-3. Formáty OPR instrukcí PDP-5

0	1	2	3	4	5	6	7	8	9	10	11	
1	1	1	0	CLA	CLL	CMA	CML	RAR	RAL	0	IAC	Skupina 1
								RTR	RTL	1		
1	1	1	1	CLA	SMA	SZA	SNL	0	OSR	HLT		Skupina 2
					SPA	SNA	SZL	1				

63.1.2. PDP-8

Odkazy:

- Micro 16V
- PDP-5
- TX-0
- ND812

Tento minipočítač se stal legendou. První počítač na světě který dosáhl takového rozšíření.

Fakta:

- Představen 1965-03-22 v New Yorku.
- Výroba ukončena v roce 1968.
- Celkem vyrobeno 1450 kusů.
- Zaváděcí cena \$18 000.

Tento počítač je také znám pod názvy:

- Clasic PDP-8
- Straight-8
- PCP-88 — používán firmou Foxboro Corporation
- AN/GYK-6 (Army-Navy Ground-based (Y)data-processing (K)Computer 6)

Byl postaven hlavně z logických modulů DEC série R. Byly pužity rovněž moduly sérií S a B. Napájecí napětí byla +10 a -15 voltů stejně jako v PDP-5 a logické úrovně rovněž -3 V logická jednička a 0V logická nula.

63.1.3. PDP-8/S

Levnější verze PDP-8. Používá sériovou aritmetiku. Tím se snížil počet obvodů ale také se významě snížila rychlost vykonávání instrukcí.

63.1.4. PDP-8/I

Fakta:

- PDP-8 postavený s pomocí integrovaných obvodů TTL rodiny SN7400 v logických modulech M.
- Paměťový cyklus trvá 1,5 μ s.
-
-
-

Tabulka 63-4. Doba provádění instrukcí

instrukce	stavy	doba	stavy	doba
AND Y	F,E	3 μ s	F,D,E	4,5 μ s
TAD Y	F,E	3 μ s	F,D,E	4,5 μ s
ISZ Y	F,E	3 μ s	F,D,E	4,5 μ s
DCA Y	F,E	3 μ s	F,D,E	4,5 μ s
JMS Y	F,E	3 μ s	F,D,E	4,5 μ s
JMP Y	F	1,5 μ s	F,D	3 μ s
IOT	F,?	?		
OPR	F,?	1,5 μ s?		

63.1.4.1. Hlavní stavy procesorové jednotky

Procesorová jednotka PDP-8/I obsahuje 6 klopných obvodů pro hlavní stavy procesoru. Jsou to: Fetch, Defer, Execute, Word Count, Current Address a Break

63.1.5. IM6100

PDP-8 na jednom čipu. Na zakázku DEC vyrobila firma Intersil.

63.1.6. HD-6120

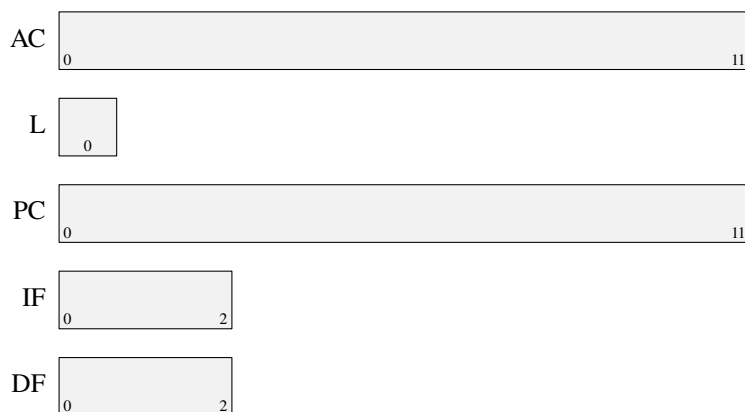
Další implementace PDP-8 v mikroporcesoru. Tentokrát na zakázku vyrobeno firmou Harris. Implementuje instrukční sadu PDP-8/E s rozšířeními.

63.2. Instrukční sada

Velikost slova minipočítače PDP-8 je 12 bitů, a základní velikost paměti je 4K slov. Procesor počítače obsahuje 3 registry:

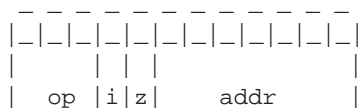
- PC — čítač programu, 12 bitů
- AC — akumulátor, 12 bitů
- L — link bit, 1 bit, udržuje 13-tý bit výsledku sčítání, v dnešní terminologii by byl nazývaný C (Carry)

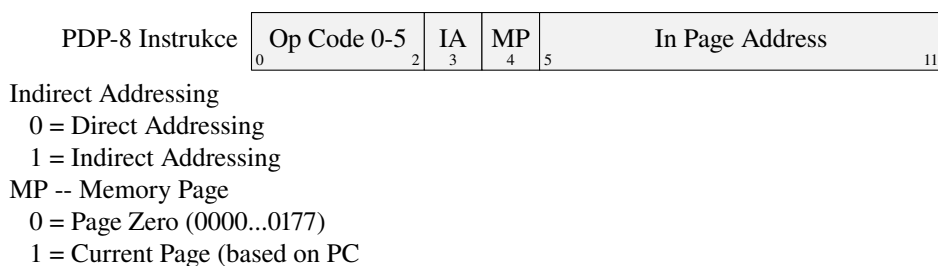
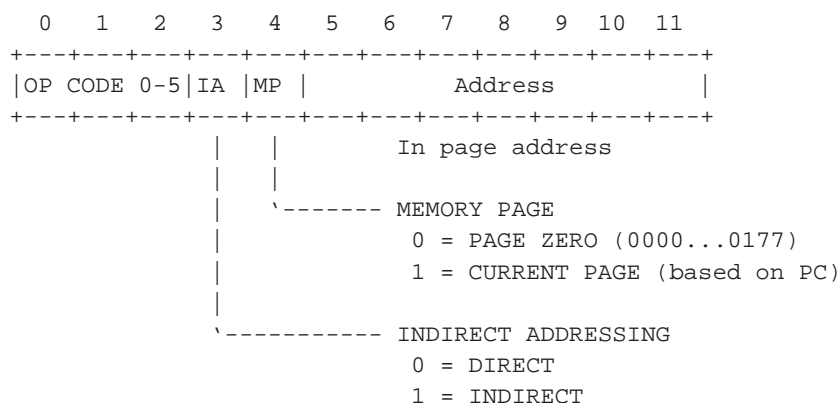
Obrázek 63-4. Registry PDP-8 přístupné programátorovi



TBD

Obrázek 63-5. Instrukční slovo PDP-8



Obrázek 63-6. Formát instrukčního slova PDP-8**Obrázek 63-7. Základní formát instrukčního slova PDP-8**

pole / bit	význam
op	operační kód
ia	indirect bit (0 = direct, 1 = indirect)
zp	page bit (0 = zero page, 1 = current page)
addr	adresa v stránce

Tabulka 63-5. Základní instrukce PDP-8

kód	název	popis
000	AND	logical AND: operand \wedge AC \longrightarrow AC
001	TAD	2's complement add: operand + \langle L,AC $\rangle \longrightarrow \langle$ L,AC \rangle
010	ISZ	increment operand and skip if zero
011	DCA	deposit and clear AC: AC \longrightarrow paměť, 0 \longrightarrow AC
100	JMS	skok do podprogramu
101	JMP	skok
110	IOT	in-out transfer, manipulace v/v zařízeními
111	OPR	mikrokódovaná operace

ISZ a další přeskokové instrukce podmíněně přeskočí následující instrukci.

Instrukce volání podprogramu JMS ukládá návratovou adresu (aktuální hodnotu PC) do prvního slova podprogramu. Návrat z podprogramu je přes nepřímý skok na tuto uloženou adresu.

V dokumentu 690304 Proposed PDP-11 Re-Organization¹⁰ jsem našel následující informaci vypovídající jak často jsou použity jednotlivé instrukce v programu FOCAL-8. Zběžným pohledem je vidět že součástí tabulky není instrukce ISZ. Proč to nevím ale nedovedu si představit program bez této instrukce.

Tabulka 63-6. Četnosti užití instrukcí v programu FOCAL-8

Instrukce	počet OCT	počet DEC
AND	500	320
TAD	1000	512
DCA	200	128
JMS	500	320
JMP	600	384
IOT	20	16
OPR1	300	192
OPR2	400	256

63.2.1. IOT (Input-Output Transfer) instrukce

Obrázek 63-8. Formát instrukce IOT minipočítače PDP-8

PDP-8 IOT Instrukce	1 1 0	Device	Function
	0 2 3	8 9	11

Bity 0 až 2 kódují instrukci IOT a obsahují oktalovou hodnotu 6. Bity 3 až 8 slouží k výběru zařízení. Některé hodnoty jsou standardizované a osazují se podle plánu:

Tabulka 63-7. Standardní adresy některých I/O zařízení PDP-8

	oktalově	
110 000 000 fff	00	Tuto adresu používá procesor
110 000 001 fff	01	čtečka děrné pásky
	02	děrovačka děrné pásky
	03	klávesnice konzoly (KRS, KCC, KSF
	04	tiskárna konzoly
	05	VC8-E CRT Display Control
	05-07	VW01 Writing Tablet
	10	KP8-E Power Fail Detect
	11	DP8-EP Redundancy Check Option
	11	63.3.2
	12	
	13	DK8-EP Programmable Real Time Clock
	13	DK8-EA Line Frequency Clock
	13	DK8-EC Crystal Clock
	20,25-27	KM8-E Time-Share

	oktalově	
	35,36	Universal Digital Controller (UDC)
	36,37	BB08-P General Purpose Interface Unit
	40,41	Synchronous Modem Interface
	50	XY8-E Incremental Plotter Control
	5x	DR8-EA 12-Channel Buffered Digital I/O
	5x	DR8-E Interprocessor Buffer
	53	AD8-EA A/D Converter
	53	AD01A 12-Bit Analog-to-Digital Converter
	53,54	AFC-8 Low-Level Analog Multiplexer
	54,56,57	AF04A Guarded Scanning Integrated Digital Voltmeter (IDVM)
	55	AA50 Digital-to-Analog Converter
	55,56	AA05A/AA07 Digital-to-Analog Converter and Control
	57	DKC8-AA/LA180 Line Printer Interface
	60-62	DF32-D Disk File and Control
	61	MP8A-Memory Parity
	61,62,64	RF08 Disk File
	63,67	CR8-E Card Reader and Control or CM8-E Optical Mark Card Reader and Control
	66	LE-8 Line Printer
	70-72	TC58 DECmagtape System
	70-72	TM8-E/F Control
	73-75	RK01 Disk Drive and Control
	74	RK8
	76,77	TC08-P DECTape Control

Bity 9 až 11 kódují funkci pro dané zařízení. Pro čtečku a děrovačku pásky a pro konzoli mají obvykle následující význam.

Operační kódy v rozsahu od 6200 do 6277 používá modul správy a stránkování paměti.

- Bit 11 caused the processor to skip the next instruction if I/O device was ready
- Bit 10 smazat AC
- Bit 9 moved a word between AC and the device, initiated another I/O transfer, and cleared the device's "ready" flag.

63.2.2. OPR (OPeRate) instrukce

OPR není instrukce ale tři skupiny mikrokódových operací. Základním kódem je 111 xxx xxx xxx. Devět bitů instrukčního slova kódují jednu ze tří skupin a jednotlivé bity spouští jednotlivé mikrooperace ve skupině. Tento způsob kódování mikrooperací je použit mimo jiné například v počítači TX-0.

A nyní k rozlišení jednotlivých skupin. Prvotním dělícím prvkem je hodnota bitu b_3 . Pokud má tento bit hodnotu 0, jedná se o skupinu 1. Pokud má hodnotu 1 jedná se o skupinu 2 nebo 3. Mezi skupinami 2 a 3 pak rozlišuje hodnota bitu b_{11} , hodnota 0 znamená skupinu 2 a hodnota 1 pak skupinu 3 a EAE rozšíření.

Obrázek 63-9. Rozdělení OPR na skupiny a koncentrovaný popis mikroinstrukcí (PDP-8)

0	1	2	3	4	5	6	7	8	9	10	11	
1	1	1	0	CLA	CLL	CMA	CML	RAR	RAL	0	IAC	Skupina 1
								0	0	BSW		nepovolená kombinace
								1	1	x		
1	1	1	1	CLA	SPA	SNA	SZL	1	OSR	HLT	0	Skupina 2
				0	0	0	SKP					
1	1	1	1	CLA	MQA	SCA	SQL	EAE	opcode	1		Skupina 3 (EAE)

Bity mikrokódové instrukce spouštějí jednotlivé mikroinstrukce. Tyto ale mohou, a také ovlivňují stejné registry. Například instrukce CLA, CMA, RAR, RAL a IAC všechny mění obsah registru AC. Protože jsou přípustné i kombinace mikroinstrukcí, jsou tyto rozděleny do sekvencí které se vykonávají postupně. Takto je zajištěno, že například kombinace mikroinstrukcí CLA IAC skončí s hodnotou 1 v akumulátoru AC. Pořadí sekvencí je uvedeno v seznamu:

Skupina 1

•

Sekvence 1: Clear

- CLA — clear AC
- CLL — clear L

•

Sekvence 2: Complement

- CMA — ones complement AC
- CML — complement L bit

•

Sekvence 3: Increment

- IAC — increment <L,AC>

•

Sekvence 4: Rotate

- RAR — rotate <L,AC> right
- RAL — rotate <L,AC> left
- RTR — rotate <L,AC> right twice
- RTL — rotate <L,AC> left twice
- BSW — byte swap

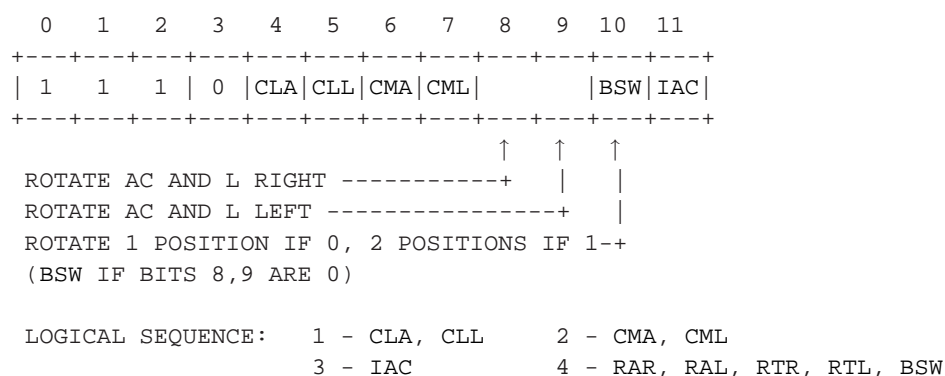
Mikroinstrukce v jednotlivých sekvencích jsou voleny tak, aby mohly být v rámci sekvence vykonávány všechny najednou, paralelně. U bitů ve čtvrté sekvenci popisujících rotaci je tomu jinak, protože všechny tři tyto bity kódují jen jednu operaci.

Tabulka 63-8. Kombinace bitů mikroinstrukcí pro rotace ve skupině 1

b ₈	b ₉	b ₁₀	instrukce
0	0	0	NOP
0	0	1	BSW
0	1	0	RAL
0	1	1	RTL
1	0	0	RAR
1	0	1	RTR
1	1	0	ilegální kombinace
1	1	1	ilegální kombinace

Instrukční slovo skupiny 1 vypadá takto.

Obrázek 63-10. Group 1 microinstructions of PDP-8



Tabulka 63-9. Group 1 Microinstructions od PDP-8

MNEMONIC	OPCODE	OPERATION
NOP	7000	No Operation. This instruction causes a 1-cycle delay in program execution, without affecting anything. It may be used for timing synchronization or as a convenient means of deleting another instruction from program.
IAC	7001	Increment Accumulator. $AC+1 \rightarrow AC$.
BSW	7002	Byte Swap. $AC_{0-5} \leftrightarrow AC_{6-11}$. Value in lower six bits in AC is swapped with value in upper six bits in AC.
RAL	7004	Rotate Accumulator Left. $AC_{1-11} \rightarrow AC_{0-10}$, $L \rightarrow AC_{11}$, $AC_0 \rightarrow L$. $\langle AC, L \rangle$ is rotated left.
RTL	7006	Rotate Two Left. Same as RAL, RAL.
RAR	7010	Rotate Accumulator Right. $AC_{0-10} \rightarrow AC_{1-11}$, $L \rightarrow AC_0$, $AC_{11} \rightarrow L$. $\langle AC, L \rangle$ is rotated right.
RTR	7012	Rotate Two Right. Same as RAR, RAR.
CML	7020	Complement Link. $\neg L \rightarrow L$.
CMA	7040	Complement Accumulator. $\neg AC \rightarrow AC$.
CLL	7100	Clear Link. $0 \rightarrow L$.
CLA	7200	Clear Accumulator. $0 \rightarrow AC$.

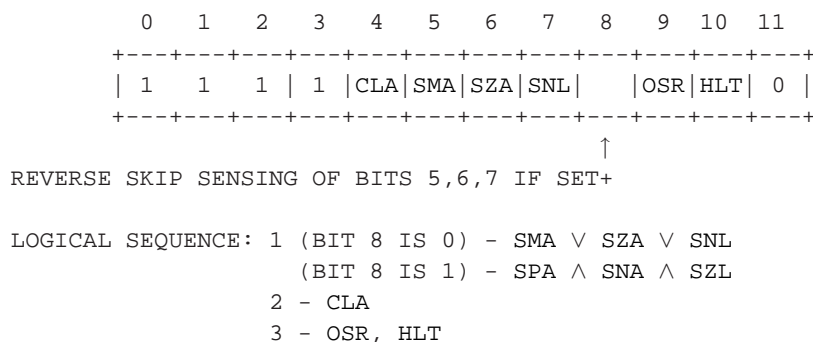
MNEMONIC	OPCODE	OPERATION
CIA	7041	Complement and Increment Accumulator. CMA IAC. $-\text{AC} \rightarrow \text{AC}$.
STL	7120	Set the Link. CLL CML. $1 \rightarrow \text{L}$.
STA	7240	Set the Accumulator. CLA CMA. $7777 \rightarrow \text{AC}$.
GLK	7204	Get the Link. CLA RAL. $\text{L} \rightarrow \text{AC}_{11}, 0 \rightarrow \text{L}$.

Skupina 2

- SMA — skip on $\text{AC} < 0$ (or group)
- SZA — skip on $\text{AC} = 0$ (or group)
- SNL — skip on $\text{L} \neq 0$ (or group)
- SKP — skip uncoditiopnally
- SNA — skip in $\text{AC} \geq 0$ (and group)
- SZL — skip on $\text{L} = 0$ (and group)
- CLA — clear AC
- OSR — logically 'or' front-panel switches with AC
- HLT — halt

Instrukční slovo skupiny 2 vypadá takto.

Obrázek 63-11. Group 2 microinstructions of PDP-8



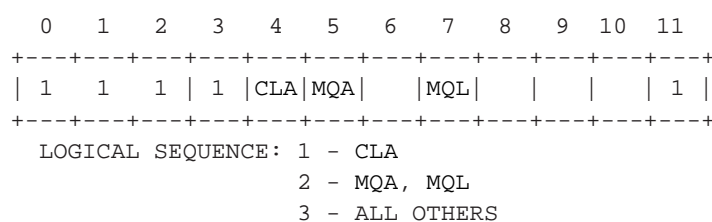
Tabulka 63-10. Group 2 Microinstructions od PDP-8

MNEMONIC	OPCODE	OPERATION
HLT	7402	Halt. Clears the run flip-flop so that program execution stops at the end of TP4 of the current machine cycle.
OSR	7404	Logical OR with Switch Register. The content of th programmer's console switch register (SR) is combined with the content of the AC by a bitwise OR operation. The result is left in the AC and the original content of the AC is lost. The content of the SR is not affected.
SKP	7410	Skip. The content of the PC is incremented by 1, to skip the next sequential instruction.
SNL	7420	Skip on Non-Zero Link. The conetent of the link is sampled. If the link contains a 1, the content of the PC is incremented to skip the next sequential instruction. If the link contains a 0, the next instruction is executed.

MNEMONIC	OPCODE	OPERATION
SZL	7430	Skip on Zero Link. The content of the link is sampled. If the link contains a 0, the content of the PC is incremented to skip the next sequential instruction. If the link contains a 1, the next instruction is executed.
SZA	7440	Skip on Zero Accumulator. The content of each bit of the AC is sampled. If every bit contains a 0, the content of the PC is incremented to skip the next sequential instruction. If any bit contains a 1, the next instruction is executed.
SNA	7450	Skip on Non-Zero Accumulator. If $AC \neq 0$ then $PC+1 \rightarrow PC$. So the next sequential instruction is skipped. Otherwise execution continues with the next sequential instruction.
SMA	7500	Skip on Minus Accumulator. If $AC_0=1$ then $PC+1 \rightarrow PC$.
SPA	7510	Skip on Positive Accumulator. If $AC_0=0$ then $PC+1 \rightarrow PC$.
CLA	7600	Clear Accumulator. $0 \rightarrow AC$.
SZA SNL	7460	Skip if $AC=0$ or $L=1$.
SNA SZL	7470	Skip if $AC=0$ and $L=0$.
SMA SNL	7520	Skip if $AC < 0$ or $L=1$.
SPA SZL	7530	Skip if $AC \geq 0$ and $L=0$.
SMA SZA	7540	Skip if $AC \leq 0$.
SPA SNA	7550	Skip if $AC > 0$.
SMA SZA SNL	7560	Skip if $AC \leq 0$ or $L=1$.
SPA SNA SZL	7570	Skip if $AC > 0$ or $L=0$.

Nepoužité kombinace OPR byly dány do třetí skupiny mikroprogramových akcí.

Obrázek 63-12. Group 3 microinstructions of PDP-8



Tabulka 63-11. Group 3 Microinstructions od PDP-8

MNEMONIC	OPCODE	OPERATION
CLA	7601	Clear Accumulator. Each bit of the AC is loaded with a binary 0.
MQL	7421	Multiplier Quotient Load. The content of the AC is loaded into the MQ. The AC is cleared and the original content of the MQ is lost.

MNEMONIC	OPCODE	OPERATION
MQA	7501	Multiplier Quotient into Accumulator. The content of the MQ is combined with the content of the AC by a bitwise logical OR operation, and the result is loaded into the AC. The original content of the AC is lost, but the original content of the MQ is not affected. Note that this instruction provides the programmer with a direct inclusive OR operation.
SWP	7521	Swap Accumulator and Multiplier Quotient. The content of the AC and the content of the MQ are exchanged. This is a microprogrammed combination of MQA and MQL.
CAM	7621	Clear Accumulator and Multiplier Quotient. Each bit of both the AC and the MQ loaded with a binary 0. This is microprogrammed combination of CLA and MQL.

63.2.3. Nultá stránka paměti

Tato část paměti od adresy 0000 do adresy 0177 (oktalově) má význam rozšířených registrů. Procesor má jen 3 registry ale adresovací mód zero page dovoluje použít část paměti jako dalších 128 registrů. Některá slova této paměti mají ovšem zvláštní význam a chování.

Slova na adresách 0010 až 0017 se při použití s nepřímým adresováním automaticky inkrementují (zvětšují o jedničku) a ještě před použitím.

63.2.4. Kousky kódu

Vynulování části paměti od adresy LOC v délce CNT slov.

```

CLA
TAD      LOC-1
DCA      X1      / X1 = LOC-1
TAD      -CNT
DCA      TMP      / TMP = -CNT

LOOP,    / do {
DCA I    X1      / M[++X1] = 0
ISZ      TMP      / TMP++
JMP      LOOP    / } while (TMP!=0)

```

X1 je automaticky inkrementovaný registr (slovo paměti na nulté stránce).

Přesun bloku v paměti z adresy SRC na adresu DST v délce CNT slov.

```

CLA
TAD      SRC-1
DCA      X1      / X1 = SRC-1
TAD      DST-1
DCA      X2      / X2 = DST-1
TAD      -CNT
DCA      TMP      / setup TMP = -CNT

LOOP,    / do {
TAD I    X1

```

```

DCA I   X2      /   M[++X2] = M[++X1]
ISZ     TMP     /   TMP++
JMP     LOOP    /   } while (TMP!=0)

```

63.2.4.1. RIM Loader Programs

Příklad 63-1. RIM Loader pro Low-Speed Reader

```

                                / RIM LOADER
7756:                          *7756
7756: 6032  RIM, +----->KCC
7757: 6031      | +--->KSF
7760: 5357      | |<---JMP 7757
7761: 6036      | |      KRB          / read upper byte
7762: 7106      | |      CLL RTL      / rotate left 4*
7763: 7006      | |      RTL          / LA je 8 765 432 10. ba9
7764: 7510      | | +<SPA          / skip b7=0
7765: 5357      | +--|-JMP 7757
7766: 7006      |      +>RTL          / move upper byte in place A je 543 210 .ba 987
7767: 6031      | +->+<KSF          / čekání na další byte
7770: 5367      | +--|-JMP 7767      /+
7771: 6034      |      +>KRS          / read lower byte
7772: 7420      |      +<SNL
7773: 3776      |      | DCA I 7776   / (PTR)=A
7774: 3376      |      +>DCA 7776    / PTR=0
7775: 5356      +-----JMP 7756
7776: 0000  PTR,      0000

```

63.2.5. Abecední seznam instrukcí PDP-8

Tabulka 63-12. Operační kódy instrukcí PDP-8 řazené podle operačního kódu

název instrukce	mnemo	kód	
		bin	octal
And	AND y	000 iza aaa aaa	0000
And	AND I y	000 0za aaa aaa	0000
And	AND I	000 1za aaa aaa	0400
Two's Complement Add	TAD y	001 iza aaa aaa	1000
Two's Complement Add	TAD y	001 0za aaa aaa	1000
Two's Complement Add	TAD I y	001 1za aaa aaa	1400
Increment and Skip if Zero	ISZ y	010 iza aaa aaa	2000
Increment and Skip if Zero	ISZ y	010 0za aaa aaa	2000
Increment and Skip if Zero	ISZ I y	010 1za aaa aaa	2400
Deposit and Clear AC	DCA y	011 iza aaa aaa	3000
Deposit and Clear AC	DCA y	011 0za aaa aaa	3000
Deposit and Clear AC	DCA I y	011 1za aaa aaa	3400

název instrukce	mnemo	kód	
		bin	octal
Jump to Subroutine	JMS	100 iza aaa aaa	4000
Jump to Subroutine	JMS	100 0za aaa aaa	4000
Jump to Subroutine	JMS I	100 1za aaa aaa	4400
Jump	JMP	101 iza aaa aaa	5000
Jump	JMP	101 0za aaa aaa	5000
Jump	JMP I	101 1za aaa aaa	5400
Input/Output Transaction	IOT f	110 000 000 000	6000
Skip if Interrupt On	SKON	110 000 000 000	6000
Interrupt Turn On	ION	110 000 000 001	6001
Interrupt Turn Off	IOF	110 000 000 010	6002
Skip on Interrupt Request	SRQ	110 000 000 011	6003
Get Flags	GTF	110 000 000 100	6004
Restore Flags	RTF	110 000 000 101	6005
Skip if Greater Than	SGT	110 000 000 110	6006
Clear All Flags	CAF	110 000 000 111	6007
Set Reader/Punch Interrupt Enable	RPE	110 000 001 000	6010
RSF	RSF	110 000 001 001	6011
RRB	RRB	110 000 001 010	6012
RFC	RFC	110 000 001 100	6014
Read Buffer and Fetch new Character	RCC	110 000 001 110	6016
PCE	PCE	110 000 010 000	6020
Skip on Punch Flag	PSF	110 000 010 001	6021
Keyboard Clear Flags	KCF	110 000 011 000	6030
Skip on Keaboard Flag	KSF	110 000 011 001	6031
Clear Keyboard Flag	KCC	110 000 011 010	6032
Read Keyboard Buffer Static	KRS	110 000 011 100	6034
Keyboard Interrupt Enable	KIE	110 000 011 101	6035
Read Keyboard Buffer Dynamic	KRB	110 000 011 110	6036
Skip on No Memory Parity Error	SMP	110 001 000 001	6101
Skip On Power Low	SPL	110 001 000 010	6102
Clear Memory Parity Error Flag	CMP	110 001 000 100	6104
Skip on Key Board Flag	MKSF	110 001 001 001	6111
Clear Receive Flag	MKCC	110 001 001 010	6112
Read Transmitter Flag	MTPF	110 001 001 011	6113
Receive Operation	MKRS	110 001 001 100	6114
Set Interrupt Flip-Flop	MINT	110 001 001 101	6115
Select Specified Station	MTON	110 001 001 111	6117
Skip on Transmitter Flag	MTSF	110 001 010 001	6121
Clear Transmitter Flag	MTCF	110 001 010 010	6122
Read Receiver Flag Status	MTKF	110 001 010 011	6123

název instrukce	mnemo	kód	
		bin	octal
Transmit Operation	MTPC	110 001 010 100	6124
Skip on Interrupt Request	MINS	110 001 010 101	6125
Read Station Status	MTRS	110 001 010 111	6127
Change to Data Field	CDF	110 010 nnn 001	6201
Change Instruction Field	CIF	110 010 nnn 010	6202
Push PC on Stack 1	PPC1	110 010 000 101	6205
Enter Panel Mode	PR0	110 010 000 110	6206
Move SP1 to AC	RSP1	110 010 000 111	6207
Read Data Field	RDF	110 010 001 100	6214
Push AC on Stack 1	PAC1	110 010 001 101	6215
Enter Panel Mode	PR1	110 010 001 110	6216
Load SP1 from AC	LSP1	110 010 001 111	6217
Read Instruction Field	RIF	110 010 010 100	6224
Return from Stack 1	RTN1	110 010 010 101	6225
Enter Panel Mode	PR2	110 010 010 110	6226
Move SP2 to AC	RSP2	110 010 010 111	6227
Pop AC from Stack 1	POP1	110 010 011 101	6235
Enter Panel Mode	PR3	110 010 011 110	6236
Load SP2 from AC	LSP2	110 010 011 111	6237
Push PC on Stack 2	PPC2	110 010 100 101	6245
Write to Switch Register	WSR	110 010 100 110	6246
Push AC on Stack 2	PAC2	110 010 101 101	6255
Get Current Fields	GCF	110 010 101 110	6256
Return from Stack 2	RTN2	110 010 110 101	6265
Pop AC from Stack 2	POP2	110 010 111 101	6275
Disk Skip on Flag	DSKP	110 111 100 001	6741
Disk Clear	DCLR	110 111 100 010	6742
Load Address and Go	DLAG	110 111 100 011	6743
Load Current Address	DLCA	110 111 100 100	6744
Read Status	DRST	110 111 100 101	6745
Load Command	DLDC	110 111 100 110	6746
Maintenance Instruction	DMAN	110 111 100 111	6747
Operate	OPR b	111 bbb bbb bbb	7000
Operate Group 1	OPR1 b	111 0bb bbb bbb	7000
No Operation	NOP	111 000 000 000	7000
Increment Accumulator	IAC	111 000 000 001	7001
Byte Swap	BSW	111 000 000 010	7002
Rotate Accumulator Left	RAL	111 000 000 100	7004
Rotate Two Left	RTL	111 000 000 110	7006
Rotate Accumulator Right	RAR	111 000 001 000	7010

název instrukce	mnemo	kód	
		bin	octal
Rotate Two Right	RTR	111 000 001 010	7012
Rotate AC Left 3	R3L	111 000 001 100	7014
Complement Link	CML	111 000 010 000	7020
Complement Accumulator	CMA	111 000 100 000	7040
Set the printer/punch Flag	TLF	111 000 100 000	7040
Skip on Teleprinter Flag	TSF	111 000 100 001	7041
Complement And Increment Accumulator	CIA	111 000 100 001	7041
Clear Teleprinter Flag	TCF	111 000 100 010	7042
Load Teleprinter and Print	TPC	111 000 100 100	7044
Skip if reader or printer interrupt Flag	TSK	111 000 100 101	7045
Load Teleprinter Sequence	TLS	111 000 100 110	7046
Clear Link	CLL	111 001 000 000	7100
Set Link	STL	111 001 010 000	7120
Clear Accumulator	CLA	111 010 000 000	7200
Get Link	GLK	111 010 000 100	7204
Step Counter OR with AC	SCA	111 010 010 001	7221
Set Accumulator	STA	111 010 100 000	7240
Operate Group 2	OPR2 b	111 1bb bbb bb0	7400
Operate Group 3	OPR3 b	111 1bb bbb bb1	7401
EAE	EAE f	111 100 00f ff1	7401
Halt	HLT	111 100 000 010	7402
OR With Switch Register	OSR	111 100 000 100	7404
Unconditional Skip	SKP	111 100 001 000	7410
Skip on Non-Zero Link	SNL	111 100 010 000	7420
Load Multiplier Quotient	MQL	111 100 010 001	7421
Skip on Zero Link	SZL	111 100 011 000	7430
Skip on Zero Accumulator	SZA	111 100 100 000	7440
Skip on Non-Zero Accumulator	SNA	111 100 101 000	7450
Skip on Minus Accumulator	SMA	111 101 000 000	7500
Multiplier Quotient Load into Accumulator	MQA	111 101 000 001	7501
Skip on Positive Accumulator	SPA	111 101 001 000	7510
Swap MQ and AC	SWP	111 101 010 001	7521
Clear Accumulator	CLA	111 110 000 000	7600
Clear Accumulator	CLA	111 110 000 001	7601
Load AC from switch register	LAS	111 110 000 100	7604
Clear Accumulator and Multiplier Quotient	CAM	111 110 010 001	7621
Load AC with contents od MQ	ACL	111 111 000 001	7701

ACL

Jméno

ACL — Load AC with contents of MQ

Přehled

ACL

L
.

Popis

Střadač AC je naplněn obsahem registru MQ. Instrukce je kombinací instrukcí CLA a MQA.

MQ → AC

mnemo	kód	
	binary	octal
ACL	111 111 000 001	7701

Odkazy

Podobné instrukce: CLA, MQA, MQL.

AND

Jméno

AND — Logická operace AND (\wedge)

Přehled

AND op

AND I op

L
-

Popis

Nad operandem instrukce a střadačem je vykonána operace logického součinu AND a výsledek je uložen do střadače AC.

$$AC \wedge \text{operand} \longrightarrow AC$$

mnemo	kód	
	binary	octal
AND y	000 iza aaa aaa	0000
AND I	000 1za aaa aaa	0400
AND I y	000 0za aaa aaa	0000

Ukázky

```

1          / File: db/pdp8.code/and-example.pal  -- mode:asm; --
2
3      0030          *0030
4 00030 2525  GLBA,  2525
5 00031 6002  GLBPTR, 6002
6
7      0400          *0400
8 00400 0206  START,  AND LOCA
9 00401 0030          AND GLBA
10 00402 0607          AND I LOCPTR
11 00403 0431          AND I GLBPTR
12 00404 0032          AND Z 32
13 00405 7402          HLT
14
15 00406 5252  LOCA,  5252
16 00407 6001  LOCPTR, 6001
17          $
      No detected errors

```

Odkazy

Podobné instrukce: DCA, OSR, TAD.

BSW

Jméno

BSW — Byte Swap

PDP-8/E

Přehled

BSW

Poznámky k tvaru instrukce

L
-

Popis

Prohodí hodnoty v horní a dolní polovině střadače.

$$AC_{0\dots5} \longleftrightarrow AC_{6\dots11}$$

mnemo	kód	
	binary	octal
BSW	111 000 000 010	7002

Odkazy

Podobné instrukce: RAL, RAR, RTL, RTR.

CAF

Jméno

CAF — Clear all Flags

PDP-8/E

Přehled

CAF

L
-

Popis

Instrukce provede to samé, jako bychom na předním panelu zmáčkli tlačítko CLEAR. Nastaví AC a L na 0 a vygeneruje INITIALIZE signál na sběrnici OMNIBUS a na I/O rozhraní.

Instrukce nesmí být vykonána pokud jsou periferní zařízení aktivní. Například nesmí být vykonána v době 100 ms po instrukci TLS

0 → AC
0 → L
0 → GT
0 → IEFF ; disable interrupt
signal INITIALIZE, IOCLR

mnemo	kód	
	binary	octal
CAF	110 000 000 111	6007

Poznámka: Tato instrukce byla implementována až v PDP-8/E. Je rovněž implementována v mikroprocesoru HD-6120.

Odkazy

Podobné instrukce: CLL, CLA, GTF, SGT, RTF.

- Small Computer Handbook 1973¹, strana 4-8, 4-16.

Poznámky

1. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/Small_Computer_Handbook_1973.pdf

CAM

Jméno

CAM — Clear Accumulator and Multiplier Quotient

Přehled

CAM

L
.

Popis

Instrukce nastaví obsah střadače AC a registru MQ na hodnotu 0. Instrukce je kombinací instrukcí CLA a MQL

0 → AC
0 → MQ

mnemo	kód	
	binary	octal
CAM	111 110 010 001	7621

Odkazy

Podobné instrukce: CLA, MQL.

CDF

Jméno

CDF — Change Data Field

Přehled

CDF

Popis

Obsah bitů b_6 - b_8 instrukce je uložen do registru DF. Protože obsah registru DF je přidáván k adrese při čtení dat z paměti, další instrukce bude číst z paměťové banky v tomto nově nastaveném registru DF.

$$MB_{6-8} \longrightarrow DF$$

mnemo	kód	
	binary	octal
CDF	110 010 nnn 001	6201

Ukázky

```

1          / File: db/pdp8.code/ex_dfif.pal  -*- mode:asm; -*-
2
3      0400      *0400          / program start at address 400 in field 0
4 00400 7200  START,  CLA
5 00401 6224      RIF          / Read IF into AC bits 6-8
6 00402 3211      DCA LASTIF
7 00403 6214      RDF          / Read DF into AC bits 6-8
8 00404 3212      DCA LASTDF
9
10 00405 6221      CDF 20      / Change DF to 2
11 00406 6212      CIF 10      / Change IB to 1
12 00407 5300      JMP 0500    / IB->IF; Jump to 0500 in IF 1
13 00410 7402      HLT
14
15 00411 0000  LASTIF, 0
16 00412 0000  LASTDF, 0
17
18          FIELD 1
19 10200 7402      HLT
20          $

No detected errors
```

Odkazy

Podobné instrukce: CIF, RDF, RIF.

- Small Computer Handbook 1970¹, strana 51
- Small Computer Handbook 1973², strana 5-14

Poznámky

1. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/SmallComputerHandbook_1970.pdf
2. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/Small_Computer_Handbook_1973.pdf

CIA

Jméno

CIA — Complement And Increment Accumulator

Přehled

CIA

L
*

Popis

Spočet spočte zápornou hodnotu k číslu ve střadači AC, a výsledek uloží zpět do střadače AC. Instrukce je kombinací instrukcí CMA a IAC.

$-AC \rightarrow AC$

mnemo	kód	
	binary	octal
CIA	111 000 100 001	7041

Odkazy

Podobné instrukce: CML, CIA.

CIF

Jméno

CIF — Change Instruction Field

Přehled

CIF

Popis

Obsah bitů b_6 - b_8 instrukce je uložen do registru IB. Následná instrukce JMP nebo JMS zapíše obsah registru IB do registru IF a způsobí tak přechod na banku z programem určenou registrem IB.

$$MB_{6-8} \longrightarrow IB$$

mnemo	kód	
	binary	octal
CIF	110 010 nnn 010	6202

Ukázky

```

1          / File: db/pdp8.code/ex_dfif.pal  -*- mode:asm; -*-
2
3          0400          *0400          / program start at address 400 in field 0
4 00400 7200  START,  CLA
5 00401 6224          RIF          / Read IF into AC bits 6-8
6 00402 3211          DCA LASTIF
7 00403 6214          RDF          / Read DF into AC bits 6-8
8 00404 3212          DCA LASTDF
9
10 00405 6221          CDF 20          / Change DF to 2
11 00406 6212          CIF 10          / Change IB to 1
12 00407 5300          JMP 0500          / IB->IF; Jump to 0500 in IF 1
13 00410 7402          HLT
14
15 00411 0000  LASTIF, 0
16 00412 0000  LASTDF, 0
17
18          FIELD 1
19 10200 7402          HLT
20          $

No detected errors
```

Odkazy

Podobné instrukce: CDF, RDF, RIF.

- Small Computer Handbook 1970¹, strana 51
- Small Computer Handbook 1973², strana 5-14

Poznámky

1. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/SmallComputerHandbook_1970.pdf
2. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/Small_Computer_Handbook_1973.pdf

CLA

Jméno

CLA — Clear Accumulator

Přehled

CLA

L
-

Popis

Všechny bity střadače AC jsou nastaveny na 0. V střadači je tedy hodnota 0000_{OCT} .

$0 \rightarrow \text{AC}$

Instrukce se dá vyjádřit více kódy. Je totiž kódována bitem b_4 ve skupinách μ operací 1, 2 i 3.

mnemo	kód	
	binary	octal
CLA	111 010 000 000	7200
CLA	111 110 000 000	7600
CLA	111 110 000 001	7601

Odkazy

Podobné instrukce: CLL, CMA, DCA, GLK, STA.

- 63.2.2

CLL

Jméno

CLL — Clear Link

Přehled

CLL

L
*

Popis

Instrukce smaže obsah jednobytového registru L. T.j. nastaví jej na hodnotu 0.

$0 \longrightarrow L$

mnemo	kód	
	binary	octal
CLL	111 001 000 000	7100

Odkazy

Podobné instrukce: CLA, CML, STL.

CMA

Jméno

CMA — Complement Accumulator

Přehled

CMA

L
-

Popis

Jedničkový doplněk střadače AC. Operace bitově neguje obsah střadače AC.

$\neg AC_j \longrightarrow AC_j$

mnemo	kód	
	binary	octal
CMA	111 000 100 000	7040

Odkazy

Podobné instrukce: CLA, CML, CIA.

CML

Jméno

CML — Complement Link

Přehled

CML

L
*

Popis

Operace neguje obsah jednobitového registru L.

$\neg L \longrightarrow L$

mnemo	kód	
	binary	octal
CML	111 000 010 000	7020

Odkazy

Podobné instrukce: CLL, CMA.

CMP

Jméno

CMP — Clear Memory Parity Error Flag

Přehled

CMP

L
-

Popis

Příznak chyby parity paměti je smazán.

0 → Memory Parity Error Flag

mnemo	kód	
	binary	octal
CMP	110 001 000 100	6104

Odkazy

Podobné instrukce: SMP.

DCA

Jméno

DCA — Deposit and Clear AC

Přehled

DCA op

DCA I op

L
-

Popis

Obsah střadače je uložen do paměti určené operandem a poté je obsah střadače vymazán.

AC → mem[operand]
0 → AC

mnemo	kód	
	binary	octal
DCA y	011 iza aaa aaa	3000
DCA y	011 0za aaa aaa	3000
DCA I y	011 1za aaa aaa	3400

Odkazy

Podobné instrukce: AND, TAD.

DCLR

Jméno

DCLR — Disk Clear

Přehled

DCLR

L
*

Popis

Instrukce fykoná funkce podle nastavení bitů 10 a 11 střadače AC.

Tabulka 63-1. DCLR příkazy

AC ₁₀	AC ₁₀	Mnemo	popis
0	0	DCLS	Clear AC and Status Register
0	1	DCLC	Clear AC and Control, Major registers are cleared. This instruction will stop the control even if it is rewriting the header.
1	0	DCLD	Clear AC, Recalibrate Selected Drive to track 000 and clear Status Register.

mnemo	kód	
	binary	octal
DCLR	110 111 100 010	6742

Odkazy

Podobné instrukce: DLAG, DLCA, DLDC, DMAN, DRST, DSKP.

- RK8
- Minicomputer Handbook strana 9-116

DLAG

Jméno

DLAG — Load Address and Go

Přehled

DLAG

L
*

Popis

Instrukce nahraje adresu sektoru z registru AC a provede instrukci v příkazovém registru.

mnemo	kód	
	binary	octal
DLAG	110 111 100 011	6743

Odkazy

Podobné instrukce: DCLR, DLCA, DLDC, DMAN, DRST, DSKP.

- RK8
- Minicomputer Handbook strana 9-116

DLCA

Jméno

DLCA — Load Current Address

Přehled

DLCA

L
*

Popis

Instrukce nahraje do adresního registru obsah AC. AC je poté vymazán.

mnemo	kód	
	binary	octal
DLCA	110 111 100 100	6744

Odkazy

Související instrukce: DCLR, DLAG, DLDC, DMAN, DRST, DSKP.

- RK8
- Minicomputer Handbook strana 9-116

DLDC

Jméno

DLDC — Load Command

Přehled

DLDC

L
*

Popis

Instrukce naplní příkazový registr obsahem střadače AC, a vymaže jak střadač AC tak i stavový registr.

mnemo	kód	
	binary	octal
DLDC	110 111 100 110	6746

Odkazy

Související instrukce: DCLR, DLAG, DLCA, DMAN, DRST, DSKP.

Odkazy:

- RK8
- Minicomputer Handbook strana 9-116

DMAN

Jméno

DMAN — Maintenance Instruction

Přehled

DMAN

L
*

Popis

Servisní instrukce, bližší informace mi nejsou známy.

mnemo	kód	
	binary	octal
DMAN	110 111 100 111	6747

Odkazy

Související instrukce: DCLR, DLAG, DLCA, DLDC, DRST, DSKP.

- RK8
- Minicomputer Handbook strana 9-116

DRST

Jméno

DRST — Read Status

Přehled

DRST

L
*

Popis

Instrukce naplní střadač AC obsahem stavového registru.

mnemo	kód	
	binary	octal

mnemo	kód	
	binary	octal
DRST	110 111 100 101	6745

Odkazy

Související instrukce: DCLR, DLAG, DLCA, DLDC, DMAN, DSKP.

- RK8
- Minicomputer Handbook strana 9-116

DSKP

Jméno

DSKP — Disk Skip on Flag

Přehled

DSKP

L
*

Popis

Přeskočí následující instrukce je-li nastaven příznak ukončení přenosu (TRANSFER DONE) nebo příznak chyby (ERROR).

mnemo	kód	
	binary	octal
DSKP	110 111 100 001	6741

Odkazy

Podobné instrukce: DCLR, DLAG, DLCA, DLDC, DMAN, DRST.

- RK8

- Minicomputer Handbook strana 9-116

GCF

Jméno

GCF — Get Current Fields

HD-6120

Přehled

GCF

L
-

Popis

Do jednotlivých bitů střadače AC je nahrán obsah řady příznaků a malých registrů..

$L \rightarrow AC_0$
 $GT \rightarrow AC_1$
 $INTEREQ \rightarrow AC_2$; 1 if INTEREQ low, 0 if high
 $PWRON \rightarrow AC_3$
 $IEFF \rightarrow AC_4$
 $0 \rightarrow AC_5$
 $IF \rightarrow AC_{6-8}$
 $DF \rightarrow AC_{9-11}$

mnemo	kód	
	binary	octal
GCF	110 010 101 110	6256

Poznámka: Tato instrukce je implementována v mikroprocesoru HD-6120.

Odkazy

Podobné instrukce: CAF, RTF, SGT.

- HD6120 Specifications, strana 4-16

GLK

Jméno

GLK — Get Link

Přehled

GLK

L
-

Popis

Instrukce GLK je kombinací instrukcí CLA a RAL. Instrukce udělá to že načte hodnotu registru L do středače AL.

if L=0 then 0→AC

if L=1 then 1→AC

mnemo	kód	
	binary	octal
GLK	111 010 000 100	7204

Odkazy

Podobné instrukce: CLA, RAL.

GTF

Jméno

GTF — Get Flags

PDP-8/E, HD-6120

Přehled

GTF

L
-

Popis

LINK \longrightarrow AC₀
 GT \longrightarrow AC₁
 INTREQ \longrightarrow AC₂
 PWRON \longrightarrow AC₃
 1 \longrightarrow AC₄
 0 \longrightarrow AC₅
 ISF \longrightarrow AC₆₋₈
 DSF \longrightarrow AC₉₋₁₁

mnemo	kód	
	binary	octal
GTF	110 000 000 100	6004

Poznámka: Tato instrukce byla implementována až v PDP-8/E.

Odkazy

Podobné instrukce: CAF, GCF, RTF, SGT.

- HD6120 Specifications, strana 4-17

HLT

Jméno

HLT — Halt

Přehled

HLT

L
-

Popis

Instrukce zastaví procesor. Tato instrukce je kódována bitem b_{10} v mikrokódu instrukce OPR2.

mnemo	kód	
	binary	octal
HLT	111 100 000 010	7402

Odkazy

Podobné instrukce: NOP, OPR, OPR2.

IAC

Jméno

IAC — Increment Accumulator

Přehled

IAC

L
-

Popis

Obsah střadače AC je zvětšen o jedničku.

$$AC+1 \longrightarrow AC$$

mnemo	kód	
	binary	octal
IAC	111 000 000 001	7001

Odkazy

Podobné instrukce: NOP, ISZ.

IOF

Jméno

IOF — Interrupt Turn Off

Přehled

IOF

L
-

Popis

Instrukce zakáže přerušení.

mnemo	kód	
	binary	octal
IOF	110 000 000 010	6002

Odkazy

Podobné instrukce: ION, SKON.

- Small Computer Handbook 1973¹, strana 4-8

Poznámky

1. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/Small_Computer_Handbook_1973.pdf

ION

Jméno

ION — Interrupt Turn On

Přehled

ION

L
-

Popis

Instrukce povolí přerušení.

mnemo	kód	
	binary	octal
ION	110 000 000 001	6001

Odkazy

Podobné instrukce: IOF, SKON.

IOT f

Jméno

IOT f — Input/Output Transaction

Přehled

ISZ op

ISZ I op

L
*

Popis

Obsah paměti určené operandem je zvětšen o 1. Je-li výsledná hodnota 0 je přeskočena následující instrukce.

operand + 1 \rightarrow operand; If operand=0 then PC+1 \rightarrow PC

mnemo	kód	
	binary	octal
ISZ y	010 iza aaa aaa	2000
ISZ y	010 0za aaa aaa	2000
ISZ I y	010 1za aaa aaa	2400

Odkazy

Podobné instrukce: IAC, JMP, JMS, SKP, TAD.

JMP

Jméno

JMP — Jump

Přehled

JMP op

JMP I op

L
-

Popis

Do PC je nahrána adresa určená operandem. Vykonávání programu pokračuje instrukcí na této adrese.

operand \rightarrow PC

mnemo	kód	
	binary	octal
JMP	101 1za aaa aaa	5000
JMP	101 0za aaa aaa	5000
JMP I	101 1za aaa aaa	5400

Odkazy

Podobné instrukce: ISZ, JMS, SKP.

JMS

Jméno

JMS — Jump to Subroutine

Přehled

JMS op

JMS I op

L
-

Popis

Obsah PC je uložen do paměti určené operandem a poté je do PC nahrána adresa o jedničku větší než kam byl uložen jeho obsah.

PC \rightarrow mem[operand]
operand+1 \rightarrow PC

mnemo	kód	
	binary	octal
JMS	100 1za aaa aaa	4000
JMS	100 0za aaa aaa	4000
JMS I	100 1za aaa aaa	4400

Odkazy

Podobné instrukce: JMP, SKP.

KCC

Jméno

KCC — Clear Keaboard Flag

Přehled

KCC

L
-

Popis

V přípravě na čtení znaku z terminálu je vymazán příznak klávesnice a střadač.

0 → AC
0 → Keyboard Flag

mnemo	kód	
	binary	octal
KCC	110 000 011 010	6032

Odkazy

Podobné instrukce: KCF, KIE, KRB, KRS, KSF.

KCF

Jméno

KCF — Keyboard Clear Flags

Přehled

KCF

L
-

Popis

Příznak klávesnice oznamující že klávesnice má připravená data je smazán.

Tato instrukce podporována na modelech starších než PDP-8/E. Instrukci je možno použít se zařízeními:

zařízení	vstup	výstup
Console TTY	03	04
Second TTY	40	41
Serial printer	65	66
VT78 serial #1	30	31
VT78 serial #2	32	33
DECmate printer	32	33

0 →

mnemo	kód	
	binary	octal
KCF	110 000 011 000	6030

Odkazy

Podobné instrukce: KCC, KSF, KRS, KIE, KRB.

Odkazy:

- [1] strana 6-5
-

KIE

Jméno

KIE — Keyboard Interrupt Enable

Přehled

KIE

L
-

Popis

Obsah střadače AC je nahrán do řídicího registru zařízení.

Tato instrukce podporována na modelech starších než PDP-8/E. Instrukcei je možno použít se zařízeními:

zařízení	vstup	výstup
Console TTY	03	04
Second TTY	40	41
Serial printer	65	66
VT78 serial #1	30	31
VT78 serial #2	32	33
DECmate printer	32	33

0 →

mnemo	kód	
	binary	octal
KIE	110 000 011 101	6035

Odkazy

Podobné instrukce: KCC, KCF, KSF, KRS, KRB.

KRB

Jméno

KRB — Read Keyboard Buffer Dynamic

Přehled

KRB

L
-

Popis

Je vymazán střadač a příznak klávesnice. Poté je k obsahu střadače přičten znak přečtený z klávesnice.

0 → AC
 0 → Keyboard Flag
 TTI + AC → AC

mnemo	kód	
	binary	octal
KRB	110 000 011 110	6036

Ukázky

```

1          / File: db/pdp8.code/keybinput.pal -- mode:asm; --
2
3          0200 *200
4 00200 6032 INPUT, KCC          /CLEAR KEYBOARD FLAG
5 00201 4204          JMS LISN    /ENTER SUBROUTINE
6 00202 3211          DCA STORE    /STORE ASCII CHARACTER
7 00203 7402          HLT
8
9 00204 0000 LISN, 0          /LISN SUBROUTINE
10 00205 6031          KSF          /KEYBOARD FLAG RAISED YET?
11 00206 5205          JMP .-1      /NO: CHECK AGAIN
12 00207 6036          KRB          /YES: READ THE CHARACTER
13 00210 5604          JMP I LISN   /RETURN TO MAINLINE
14
15 00211 0000 STORE, 0
16          $

```

No detected errors

Odkazy

Podobné instrukce: KCC, KCF, KIE, KRS, KSF.

Odkazy:

- [1] strana 6-4
-

KRS

Jméno

KRS — Read Keyboard Buffer Static

Přehled

KRS

L
-

Popis

Je přečten znak z klávesnice a jeho hodnota přičtena k hodnotě střadače. Výsledek je uložen do střadače.

$TTI + AC \longrightarrow AC$
 $0 \longrightarrow \text{Keyboard Flag}$

mnemo	kód	
	binary	octal
KRS	110 000 011 100	6034

Odkazy

Podobné instrukce: KCC, KCF, KIE, KRB, KSF.

KSF

Jméno

KSF — Skip on Keaboard Flag

Přehled

KSF

L
-

Popis

Je zkoumán příznak klávesnice. Má-li hodnotu 1, je přeskočena následující instrukce.

If Keyboard Flag = 1 then PC+1→PC

mnemo	kód	
	binary	octal
KSF	110 000 011 001	6031

Odkazy

Podobné instrukce: KCC, KCF, KIE, KRB, KRS.

LAS

Jméno

LAS — Load AC from switch register

Přehled

LAS

L

L
.

Popis

Instrukce je bitovou kombinací instrukcí CLA a OSR.

SR \longrightarrow AC

mnemo	kód	
	binary	octal
LAS	111 110 000 100	7604

Odkazy

Podobné instrukce: HLT.

LSP1

Jméno

LSP1 — Load SP1 from AC

HD-6120

Přehled

LSP1

L
*

Popis

Obsah střadače AC je uložen do registru SP1 ukazatele na první zásobník.

AC \longrightarrow SP1
0 \longrightarrow AC

mnemo	kód	
	binary	octal
LSP1	110 010 001 111	6217

Odkazy

Podobné instrukce: BSW, RAL, RAR, RTL, RTR.

- HD6120 Specifications strana 4-13

LSP2

Jméno

LSP2 — Load SP2 from AC

HD-6120

Přehled

LSP2

L
*

Popis

Obsah střadače AC je uložen do registru SP2 ukazatele na druhý zásobník.

AC \longrightarrow SP2

0 \longrightarrow AC

mnemo	kód	
	binary	octal
LSP2	110 010 011 111	6237

Odkazy

Podobné instrukce: BSW, RAL, RAR, RTL, RTR.

- HD6120 Specifications strana 4-13

MINS

Jméno

MINS — Skip on Interrupt Request

Přehled

MINS

L
.

Popis

mnemo	kód	
	binary	octal
MINS	110 001 010 101	6125

Odkazy

Podobné instrukce: MINT, MKCC, MKRS, MKSF, MTCF, MTKF, MTON, MTPC, MTPF, MTRS, MTSF.

- [1] strana D-12
- 63.3.2

MINT

Jméno

MINT — Set Interrupt Flip-Flop

Přehled

MINT

L
.

Popis

mnemo	kód	
	binary	octal
MINT	110 001 001 101	6115

Odkazy

Podobné instrukce: MINS, MKCC, MKRS, MKSF, MTCF, MTKF, MTON, MTPC, MTPF, MTRS, MTSF.

- [1] strana D-12
- 63.3.2

MKCC

Jméno

MKCC — Clear Receive Flag

Přehled

MKCC

L
.

Popis

mnemo	kód	
	binary	octal
MKCC	110 001 001 010	6112

Odkazy

Podobné instrukce: MINS, MINT, MKRS, MKSF, MTCF, MTKF, MTON, MTPC, MTPF, MTRS, MTSF.

- [1] strana D-12
- 63.3.2

MKRS

Jméno

MKRS — Receive Operation

Přehled

MKRS

L
.

Popis

mnemo	kód	
	binary	octal

mnemo	kód	
	binary	octal
MKRS	110 001 001 100	6114

Odkazy

Podobné instrukce: MINS, MINT, MKCC, MKSF, MTCF, MTKF, MTON, MTPC, MTPF, MTRS, MTSF.

- [1] strana D-12
- 63.3.2

MKSF

Jméno

MKSF — Skip on Key Board Flag

Přehled

MKSF

L
.

Popis

mnemo	kód	
	binary	octal
MKSF	110 001 001 001	6111

Odkazy

Podobné instrukce: MINS, MINT, MKCC, MKRS, MTCF, MTKF, MTON, MTPC, MTPF, MTRS, MTSF.

- [1] strana D-12

- 63.3.2

MQA

Jméno

MQA — Multiplier Quotient Load into Accumulator (V)

Přehled

MQA

L
.

Popis

Instrukce spočte logický součet (operace V) hodnoty ve střadači AC a hodnoty v registru MQ. Výsledek je uložen do střadače AC.

Tato instrukce je jediný způsob jak přechíst hodnotu z registru MQ. Současně se touto instrukcí dá jednodušeji spočítat logický součet OR (V).

$MQ \vee AC \longrightarrow AC$

mnemo	kód	
	binary	octal
MQA	111 101 000 001	7501

Odkazy

Podobné instrukce: CAM, CLA, MQL, SWP.

MQL

Jméno

MQL — Load Multiplier Quotient

Přehled

MQL

L
.

Popis

Do registru MQ se nahraje obsah střadače AC. Střadač je poté vynulován.

AC \longrightarrow MQ
0 \longrightarrow AC

mnemo	kód	
	binary	octal
MQL	111 100 010 001	7421

Odkazy

Podobné instrukce: ACL, CLA, MQA, SWP, CAM.

MTCF

Jméno

MTCF — Clear Transmitter Flag

Přehled

MTCF

L
.

Popis

mnemo	kód	
	binary	octal
MTCF	110 001 010 010	6122

Odkazy

Podobné instrukce: MINS, MINT, MKCC, MKRS, MKSF, MTKF, MTON, MTPC, MTPF, MTRS, MTSF.

- [1] strana D-12
- 63.3.2

MTKF

Jméno

MTKF — Read Receiver Flag Status

Přehled

MTKF

L
.

Popis

mnemo	kód	
	binary	octal

mnemo	kód	
	binary	octal
MTKF	110 001 010 011	6123

Odkazy

Podobné instrukce: MINS, MINT, MKCC, MKRS, MKSF, MTCF, MTON, MTPC, MTPF, MTRS, MTSE.

- [1] strana D-12
- 63.3.2

MTON

Jméno

MTON — Select Specified Station

Přehled

MTON

L
.

Popis

mnemo	kód	
	binary	octal
MTON	110 001 001 111	6117

Odkazy

Podobné instrukce: MINS, MINT, MKCC, MKRS, MKSF, MTCF, MTKF, MTPC, MTPF, MTRS, MTSE.

- [1] strana D-12

- 63.3.2

MTPC

Jméno

MTPC — Transmit Operation

Přehled

MTPC

L
.

Popis

mnemo	kód	
	binary	octal
MTPC	110 001 010 100	6124

Odkazy

Podobné instrukce: MINS, MINT, MKCC, MKRS, MKSF, MTCF, MTKF, MTON, MTPF, MTRS, MTSF.

- [1] strana D-12
- 63.3.2

MTPF

Jméno

MTPF — Read Transmitter Flag

Přehled

MTPF

L
.

Popis

mnemo	kód	
	binary	octal
MTPF	110 001 001 011	6113

Odkazy

Podobné instrukce: MINS, MINT, MKCC, MKRS, MKSF, MTCTF, MTKF, MTON, MTPC, MTRS, MTSF.

- [1] strana D-12
- 63.3.2

MTRS

Jméno

MTRS — Read Station Status

Přehled

MTRS

L
.

Popis

mnemo	kód	
	binary	octal
MTRS	110 001 010 111	6127

Odkazy

Podobné instrukce: MINS, MINT, MKCC, MKRS, MKSF, MTCF, MTKF, MTON, MTPC, MTPF, MTSF.

- [1] strana D-12
- 63.3.2

MTSF

Jméno

MTSF — Skip on Transmitter Flag

Přehled

MTSF

L
.

Popis

mnemo	kód	
	binary	octal
MTSF	110 001 010 001	6121

Odkazy

Podobné instrukce: MINS, MINT, MKCC, MKRS, MKSF, MTCF, MTKF, MTON, MTPC, MTPF, MTRS.

- [1] strana D-12

- 63.3.2

OPR

Jméno

OPR — Operate

Jméno

NOP — No Operation

Přehled

- NOP
- OPR *b*
- OPR1 *b*
- OPR2 *b*
- OPR3 *b*
- EAE *f*

Jedná se o různé tvary instrukce OPR. NOP je instrukce OPR bez nastavených bitů mikrokódu, proto ji uvádím zde. OPR1, OPR2 a OPR3 jsou různé skupiny lišící se nastavením bitů 3 a 11 v mikrokódu. EAE je zvláštní forma nebo podskupina operací skupiny OPR3.

L
-

Popis

Instrukce OPR je základní tvar třídy instrukcí OPR. Její speciální forma je pak instrukce NOP. Instrukce OPR se dělí do několika skupin. Mezi těmito skupinami se rozlišuje pomocí bitů b_3 a b_{11} . Případně dalších bitů.

Obrázek 63-1. Přehledová tabulka mikrokódu instrukce OPR počítače PDP-8

0	1	2	3	4	5	6	7	8	9	10	11	
1	1	1	0	CLA	CLL	CMA	CML	RAR	RAL	0	IAC	Skupina 1
								RTR	RTL	1		
								0	0	BSW		
1	1	1	1	CLA	SMA	SZA	SNL	0	OSR	HLT	0	Skupina 2

```

|          |   |   | SPA|SNA|SZL|SKP|          |
+---+---+---+---+---+---+---+---+---+---+
| 1  1  1 | 1 | CLA|MQA|SCA|MQL|EAE opcode | 1 | Skupina 3 (EAE)
+---+---+---+---+---+---+---+---+---+---+

```

mnemo	kód	
	binary	octal
NOP	111 000 000 000	7000
OPR b	111 bbb bbb bbb	7000
OPR1 b	111 0bb bbb bbb	7000
OPR2 b	111 1bb bbb bb0	7400
OPR3 b	111 1bb bbb bb1	7401
EAE f	111 100 00f ff1	7401

Odkazy

Podobné instrukce: HLT.

* **FIXME**:zajistit automatické vygenerování odkazů na konkrétní instrukce.

OSR

Jméno

OSR — OR With Switch Register

Přehled

OSR

L
-

Popis

Provede operaci logického součtu mezi AC a SR. Výsledek uloží do AC.

$SR \vee AC \longrightarrow AC$

mnemo	kód	
	binary	octal
OSR	111 100 000 100	7404

Odkazy

Podobné instrukce: AND, NOP.

PAC1

Jméno

PAC1 — Push AC on Stack 1

HD-6120

Přehled

PAC1

L
*

Popis

Obsah akumulátoru je uložen na vrchol prvního zásobníku.

AC \longrightarrow mem[SP1]

SP1-1 \longrightarrow SP1

mnemo	kód	
	binary	octal
PAC1	110 010 001 101	6215

Odkazy

Podobné instrukce: PPC1, PAC2, RTN1, POP1, RSP1, LSP1.

- HD6120 Specifications strana 4-16

PAC2

Jméno

PAC2 — Push AC on Stack 2

HD-6120

Přehled

PAC2

L
*

Popis

Obsah akumulátoru je uložen na vrchol druhého zásobníku.

AC \longrightarrow mem[SP2]
SP2-1 \longrightarrow SP2

mnemo	kód	
	binary	octal
PAC2	110 010 101 101	6255

Odkazy

Podobné instrukce: PPC2, PAC1, RTN2, POP2, RSP2, LSP2.

- HD6120 Specifications strana 4-16

PCE

Jméno

PCE — Clear Reader/Punch Interrupt Enable

Přehled

PCE

L
.

Popis

mnemo	kód	
	binary	octal
PCE	110 000 010 000	6020

Odkazy

Podobné instrukce: RPE.

- DEC-08-XINPA-A-D_intrPgm_75.pdf strana D-7
-

POP1

Jméno

POP1 — Pop AC from Stack 1

HD-6120

Přehled

POP1

L
*

Popis

Hodnota z prvního zásobníku SP1 je vyjmuta a vložena do střadače AC.

SP1+1 \longrightarrow SP1
 mem[SP1] \longrightarrow AC

mnemo	kód	
	binary	octal
POP1	110 010 011 101	6235

Odkazy

Podobné instrukce: PPC1, PAC1, RTN1, POP2, LSP1, RSP1.

- HD6120 Specifications strana 4-16

POP2

Jméno

POP2 — Pop AC from Stack 2

HD-6120

Přehled

POP2

L
*

Popis

Hodnota z druhého zásobníku SP2 je vyjmuta a vložena do střadače AC.

SP2+1 \longrightarrow SP2

mem[SP2] \longrightarrow AC

mnemo	kód	
	binary	octal
POP2	110 010 111 101	6275

Odkazy

Podobné instrukce: PPC2, PAC2, RTN2, POP1, RSP2, LSP2.

- HD6120 Specifications strana 4-16

PPC1

Jméno

PPC1 — Push PC on Stack 1

HD-6120

Přehled

PPC1

L
*

Popis

Obsah čítače instrukcí AC je uložen na vrchol prvního zásobníku.

PC \longrightarrow mem[SP1]

SP1-1 \longrightarrow SP1

mnemo	kód	
	binary	octal
PPC1	110 010 000 101	6205

Odkazy

Podobné instrukce: LSP1, PAC1, POP1, PPC2, RSP1, RTN1.

- HD6120 Specifications strana 4-16

PPC2

Jméno

PPC2 — Push PC on Stack 2

HD-6120

Přehled

PPC2

L
*

Popis

Obsah čítače instrukcí PC je uložen na vrchol druhého zásobníku.

PC \longrightarrow mem[SP2]

SP2-1 \longrightarrow SP2

mnemo	kód	
	binary	octal
PPC2	110 010 100 101	6245

Odkazy

Podobné instrukce: LSP2, PAC2, POP2, PPC1, RSP2, RTN2.

- HD6120 Specifications strana 4-16

PR0

Jméno

PR0, PR1, PR2, PR3 — Enter Panel Mode

HD-6120

Přehled

PR0

PR1

PR2

PR3

L
.

Popis

Nastaví PNLTRP a následující instrukce je vykonána v Panel módu.

1 → PNLTRP

mnemo	kód	
	binary	octal
PR0	110 010 000 110	6206
PR1	110 010 001 110	6216
PR2	110 010 010 110	6226
PR3	110 010 011 110	6236

Poznámka: Tato instrukce je implementována v mikroprocesoru HD-6120.

Odkazy

Podobné instrukce: OSR.

- HD6120 Specifications, strana 4-17

PSF

Jméno

PSF — Skip on Punch Flag

Přehled

PSF

L
.

Popis

mnemo	kód	
	binary	octal
PSF	110 000 010 001	6021

Odkazy

Podobné instrukce: PCE, RPE.

- DEC-08-XINPA-A-D_intrPgm_75.pdf strana D-7
-

R3L

Jméno

R3L — Rotate AC left 3 places

HD-6120

Přehled

R3L

L
*

Popis

Pseudokód

$$\begin{aligned} AC_j &\longrightarrow AC_{j-3} \\ AC_0 &\longrightarrow AC_9 \\ AC_1 &\longrightarrow AC_{10} \\ AC_2 &\longrightarrow AC_{11} \end{aligned}$$

mnemo	kód	
	binary	octal
R3L	111 000 001 100	7014

Odkazy

Podobné instrukce: BSW, RAL, RAR, RTL, RTR.

- HD6120 Specifications strana 4-13

RAL

Jméno

RAL — Rotate Accumulator Left

Přehled

RAL

L
*

Popis

$$\begin{aligned} AC_j &\longrightarrow AC_{j-1} \\ AC_0 &\longrightarrow L \end{aligned}$$

$$L \longrightarrow AC_{11}$$

mnemo	kód	
	binary	octal
RAL	111 000 000 100	7004

Odkazy

Podobné instrukce: BSW, RAR, RTL, RTR.

RAR

Jméno

RAR — Rotate Accumulator Right

Přehled

RAR

L
*

Popis

$$\begin{aligned} AC_j &\longrightarrow AC_{j+1} \\ AC_{11} &\longrightarrow L \\ L &\longrightarrow AC_0 \end{aligned}$$

mnemo	kód	
	binary	octal
RAR	111 000 001 000	7010

Odkazy

Podobné instrukce: BSW, RAL, RTL, RTR.

RCC

Jméno

RCC — Read Buffer and Fetch new Character (RRB, RFC)

Přehled

RCC

L
.

Popis

mnemo	kód	
	binary	octal
RCC	110 000 001 110	6016

Odkazy

Podobné instrukce: RFC, RPE, RRB, RSF.

- DEC-08-XINPA-A-D_intrPgm_75.pdf strana D-7
-

RDF

Jméno

RDF — Read Data Field

Přehled

RDF

Popis

Je spočten logický součet obsahu registru DF s bity 6,7,8 střadače AC. Výsledek je uložen ve střadači AC. Další bity střadače AC zůstávají nezměněny.

$$DF \vee AC_{6-8} \longrightarrow AC_{6-8}$$

mnemo	kód	
	binary	octal
RDF	110 010 001 100	6214

Ukázky

```

1          / File: db/pdp8.code/ex_dfif.pal  -*- mode:asm; -*-
2
3      0400      *0400          / program start at address 400 in field 0
4 00400 7200  START,  CLA
5 00401 6224      RIF          / Read IF into AC bits 6-8
6 00402 3211      DCA LASTIF
7 00403 6214      RDF          / Read DF into AC bits 6-8
8 00404 3212      DCA LASTDF
9
10 00405 6221      CDF 20      / Change DF to 2
11 00406 6212      CIF 10      / Change IB to 1
12 00407 5300      JMP 0500    / IB->IF; Jump to 0500 in IF 1
13 00410 7402      HLT
14
15 00411 0000  LASTIF, 0
16 00412 0000  LASTDF, 0
17
18          FIELD 1
19 10200 7402      HLT
20          $

```

No detected errors

Odkazy

Podobné instrukce: CDF, CIF, RIF.

- Small Computer Handbook 1970¹, strana 51
- Small Computer Handbook 1973², strana 5-14

Poznámky

1. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/SmallComputerHandbook_1970.pdf
2. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/Small_Computer_Handbook_1973.pdf

RFC

Jméno

RFC — Reader Fetch Character

Přehled

RFC

L
.

Popis

mnemo	kód	
	binary	octal
RFC	110 000 001 100	6014

Odkazy

Podobné instrukce: RPE, RRB, RSF.

- DEC-08-XINPA-A-D_intrPgm_75.pdf strana D-7
-

RIF

Jméno

RIF — Read Instruction Field

Přehled

RIF

Popis

Je spočten logický součet obsahu registru IF s bity 6,7,8 střadače AC. Výsledek je uložen ve střadači AC. Další bity střadače AC zůstávají nezměněny.

$$IF \vee AC_{6-8} \longrightarrow AC_{6-8}$$

mnemo	kód	
	binary	octal
RIF	110 010 010 100	6224

Ukázky

```

1          / File: db/pdp8.code/ex_dfif.pal  -*- mode:asm; -*-
2
3      0400      *0400          / program start at address 400 in field 0
4 00400 7200  START,  CLA
5 00401 6224      RIF          / Read IF into AC bits 6-8
6 00402 3211      DCA LASTIF
7 00403 6214      RDF          / Read DF into AC bits 6-8
8 00404 3212      DCA LASTDF
9
10 00405 6221      CDF 20      / Change DF to 2
11 00406 6212      CIF 10      / Change IB to 1
12 00407 5300      JMP 0500    / IB->IF; Jump to 0500 in IF 1
13 00410 7402      HLT
14
15 00411 0000  LASTIF, 0
16 00412 0000  LASTDF, 0
17
18          FIELD 1
19 10200 7402      HLT
20          $

```

No detected errors

Odkazy

Podobné instrukce: CIF, CDF, RDF.

- Small Computer Handbook 1970¹, strana 52
- Small Computer Handbook 1973², strana 5-15

Poznámky

1. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/SmallComputerHandbook_1970.pdf
2. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/Small_Computer_Handbook_1973.pdf

RPE

Jméno

RPE — Set Reader/Punch Interrupt Enable

Přehled

RPE

L
.

Popis

mnemo	kód	
	binary	octal
RPE	110 000 001 000	6010

Odkazy

Podobné instrukce: HLT.

- DEC-08-XINPA-A-D_intrPgm_75.pdf strana D-7
-

RRB

Jméno

RRB — Read Reader Buffer

Přehled

RRB

L
.

Popis

mnemo	kód	
	binary	octal
RRB	110 000 001 010	6012

Odkazy

Podobné instrukce: RPE, RSF.

- DEC-08-XINPA-A-D_intrPgm_75.pdf strana D-7
-

RSF

Jméno

RSF — Skip on Reader Flag

Přehled

RSF

L
.

Popis

mnemo	kód	
	binary	octal
RSF	110 000 001 001	6011

Odkazy

Podobné instrukce: RPE, RRB.

- DEC-08-XINPA-A-D_intrPgm_75.pdf strana D-7
-

RSP1

Jméno

RSP1 — Load AC from SP1

HD-6120

Přehled

RSP1

L
*

Popis

Hodnota ukazatele zásobníku SP1 je zapsána do střadače AC.

SP1 → AC

mnemo	kód	
	binary	octal
RSP1	110 010 000 111	6207

Odkazy

Podobné instrukce: BSW, RAL, RAR, RTL, RTR.

- HD6120 Specifications strana 4-13

RSP2

Jméno

RSP2 — Load AC from SP2

HD-6120

Přehled

RSP2

L
*

Popis

Hodnota ukazatele zásobníku SP2 je zapsána do střadače AC.

SP2 → AC

mnemo	kód	
	binary	octal
RSP2	110 010 010 111	6227

Odkazy

Podobné instrukce: BSW, RAL, RAR, RTL, RTR.

- HD6120 Specifications strana 4-13

RTF

Jméno

RTF — Restore Flags

PDP-8/E, HD-6120

Přehled

RTF

L
-

Popis

Obnovení příznaků a řady menších registrů z hodnoty ve střadači AC.

$AC_0 \rightarrow L$
 $AC_1 \rightarrow GT$
 $AC_4 \rightarrow IEFF$
 $AC_{6-8} \rightarrow IB$
 $AC_{9-11} \rightarrow DF$
 $1 \rightarrow IIFF$
 $0 \rightarrow AC$

mnemo	kód	
	binary	octal
RTF	110 000 000 101	6005

Poznámka: Tato instrukce byla implementována až v PDP-8/E. Je rovněž implementována v mikroprocesoru HD-6120.

Odkazy

Podobné instrukce: CAF, GTF, SGT.

RTL

Jméno

RTL — Rotate Two Left

Přehled

RTL

L
*

Popis

Pseudokód

$$\begin{aligned}AC_j &\longrightarrow AC_{j-2} \\ AC_1 &\longrightarrow L \\ AC_0 &\longrightarrow AC_{11} \\ L &\longrightarrow AC_{10}\end{aligned}$$

* *db=pdp8.isa id='111 000 000 110' mnemonic='RTL' name='Rotate Two Left' linkend=pdp8.isa.rtl*

Odkazy

Podobné instrukce: BSW, RAL, RAR, RTR, R3L.

RTN1

Jméno

RTN1 — Return from Stack 1

HD-6120

Přehled

RTN1

L
*

Popis

Obsah čítače instrukcí je obnoven z prvního zásobníku.

$SP1+1 \longrightarrow SP1$
 $mem[SP1] \longrightarrow PC$

mnemo	kód	
	binary	octal
RTN1	110 010 010 101	6225

Odkazy

Podobné instrukce: PPC1, PAC1, RTN2, POP1, RSP1, LSP1.

- HD6120 Specifications strana 4-16

RTN2

Jméno

RTN2 — Return from Stack 2

HD-6120

Přehled

RTN2

L
*

Popis

Obsah čítače instrukcí PC je obnoven z druhého zásobníku.

$SP2+1 \longrightarrow SP2$

mem[SP2] \longrightarrow PC

mnemo	kód	
	binary	octal
RTN2	110 010 110 101	6265

Odkazy

Podobné instrukce: PAC2, PPC2, RTN1, POP2, RSP2, LSP2.

- HD6120 Specifications strana 4-16

RTR

Jméno

RTR — Rotate Two Right

Přehled

RTR

L
*

Popis

$$\begin{aligned} AC_j &\longrightarrow AC_{j+2} \\ AC_{11} &\longrightarrow AC_0 \\ AC_{10} &\longrightarrow L \\ L &\longrightarrow AC_1 \end{aligned}$$

mnemo	kód	
	binary	octal
RTR	111 000 001 010	7012

Odkazy

Podobné instrukce: BSW, RAL, RAR, RTL, R3L.

SCA

Jméno

SCA — Step Counter OR with AC

KE8-E Mode A

Přehled

SCA

L
.

Popis

Obsah čítače Step Counter je logicky sečten s AC_7-AC_{11} a uložen do AC_7-AC_{11} . AC_0-AC_6 zůstává nezměněno.

mnemo	kód	
	binary	octal
SCA	111 010 010 001	7221

Odkazy

Podobné instrukce: HLT.

SGT

Jméno

SGT — Skip if Greater Than

PDP-8/E, HD-6120

Přehled

SGT

L
-

Popis

Přeskočí následující instrukci, pokud je nastave příznak/registr GT.

If GT==1 then PC+1 → PC

mnemo	kód	
	binary	octal
SGT	110 000 000 110	6006

Poznámka: Tato instrukce byla implementována až v PDP-8/E, je implementována rovněž v mikroprocesoru HD-6120.

Odkazy

Podobné instrukce: CAF, GTF, RTF.

SKON

Jméno

SKON — Skip if Interrupt On

PDP-8/E

Přehled

SKON

L
-

Popis

Je-li povoleno přerušení je přeskočena následující instrukce.

If Interrupt Enabled then PC+1 \rightarrow PC

mnemo	kód	
	binary	octal
SKON	110 000 000 000	6000

Poznámka: Tato instrukce byla implementována až v PDP-8/E.

Odkazy

Podobné instrukce: SKP.

SKP

Jméno

SKP — Unconditional Skip

Přehled

SKP

L
-

Popis

Instrukce přeskočí následující instrukci. Vykonáván programu pokračuje až další instrukcí za instrukcí následující.

$$PC + 1 \longrightarrow PC$$

mnemo	kód	
	binary	octal
SKP	111 100 001 000	7410

Odkazy

Podobné instrukce: SNL.

SMA

Jméno

SMA — Skip on Minus Accumulator

Přehled

SMA

L
-

Popis

Pokud je hodnota registru AC záporná (t.j. nejvyšší bit AC_0 je 1), tak instrukce přeskočí následující instrukci. Vykonáván programu pokračuje až další instrukcí za instrukcí následující.

$$\text{If } AC_0=1 \text{ then } PC + 1 \longrightarrow PC$$

mnemo	kód	
	binary	octal
SMA	111 101 000 000	7500

Odkazy

Podobné instrukce: SKP, SMP, SNA, SNL, SPL, SRQ.

SMP

Jméno

SMP — Skip on No Memory Parity Error

Přehled

SMP

L
-

Popis

Má-li příznak chyby parity paměti hodnotu 0, t.j. nenastala-li žádná chyba parity paměti, je následující instrukce přeskočena.

If Memory Parity Error Flag = 0 then PC+1 \longrightarrow PC

mnemo	kód	
	binary	octal
SMP	110 001 000 001	6101

Odkazy

Podobné instrukce: CMP.

SNA

Jméno

SNA — Skip on Non-Zero Accumulator

Přehled

SNA

L
-

Popis

Pokud je hodnota registru AC různá od 0, tak instrukce přeskočí následující instrukci. Vykonáván programu pokračuje až další instrukcí za instrukcí následující.

If $AC \neq 0$ then $PC + 1 \longrightarrow PC$

mnemo	kód	
	binary	octal
SNA	111 100 101 000	7450

Odkazy

Podobné instrukce: SKP, SNL.

SNL

Jméno

SNL — Skip on Non-Zero Link

Přehled

SNL

L
-

Popis

Pokud je hodnota registru L 1, tak instrukce přeskočí následující instrukci. Vykonávání programu pokračuje až další instrukcí za instrukcí následující.

If L=1 then PC + 1 \longrightarrow PC

mnemo	kód	
	binary	octal
SNL	111 100 010 000	7420

Odkazy

Podobné instrukce: SKP, SNA.

SPA

Jméno

SPA — Skip on Positive Accumulator

Přehled

SPA

L
-

Popis

Pokud je hodnota registru AC kladná (t.j. nejvyšší bit AC_0 je 0), tak instrukce přeskočí následující instrukci. Vykonávání programu pokračuje až další instrukcí za instrukcí následující.

If $AC_0=0$ then PC + 1 \longrightarrow PC

mnemo	kód	
	binary	octal
SPA	111 101 001 000	7510

Odkazy

Podobné instrukce: SKP, SMA, SNL.

SPL

Jméno

SPL — Skip On Power Low

Přehled

SPL

L
.

Popis

mnemo	kód	
	binary	octal
SPL	110 001 000 010	6102

Ukázky

```

0000                                /STORAGE FOR PC AFTER PROGRAM INTERRUPT
0001    JMP FLAGS                  /INSTRUCTION EXECUTED AFTER PROGRAM INTERRUPT
FLAGS,  SPL                        /SKIP IF POWER LOW FLAG = 1
                                /INTERRUPT NOT CAUSED BY POWER LOW, CHECK OTHER FLAGS
                                JMP OTHER
                                DCA AC    /INTERRUPT WAS CAUSED BY POWER LOW, SAVE AC
                                RAR      /GET LINK
                                DCA LINK  /SAVE LINK

```

```

MQA                /GET MQ
DCA MQ             /SAVE MQ
TAD 0000           /GET PC
DCA PC             /SAVE PC
TAD RESTART        /GET RESTART INSTRUCTION
DCA 0000           /DEPOSIT RESTART INSTRUCTION IN 0000
HLT
RESTART, JMP ABCD  /ABCD IS LOCATION OF RESTART ROUTINE

0000    JMP ABCD
ABCD,   TAD MQ      /GET MQ
        MQL         /RESTORE MQ
        TAD LINK    /GET LINK
        CLL RAL     /RESTORE LINK
        TAD AC       /RESTORE AC
        ION         /TURN ON INTERRUPT
        JMP I PC     /RETURN TO INTERRUPTED PROGRAM

```

Odkazy

Podobné instrukce: HLT, SKP.

SRQ

Jméno

SRQ — Skip on Interrupt Request

PDP-8/E

Přehled

SRQ

L
-

Popis

Je-li žádost o přerušení je přeskočena následující instrukce.

If Interrupt then PC+1 \longrightarrow PC

mnemo	kód	
	binary	octal
SRQ	110 000 000 011	6003

Poznámka: Tato instrukce byla implementována až v PDP-8/E.

Odkazy

Podobné instrukce: IOF, ION, SKON.

STA

Jméno

STA — Set Accumulator

Přehled

STA

L
-

Popis

Instrukce STA nastaví všechny bity střadače na 1. Je to to samé jako říct že instrukce STA nahraje do střadače hodnotu 7777_{oct} .

Instrukce je kombinací instrukcí CLA a CMA.

$$1 \longrightarrow AC_j \equiv 7777 \longrightarrow AC$$

mnemo	kód	
	binary	octal
STA	111 010 100 000	7240

Odkazy

Podobné instrukce: CLA, CMA.

STL

Jméno

STL — Set Link

Přehled

STL

1 \rightarrow L

Instrukce STL je kombinací instrukcí CLL a CML.

L
*

Popis

mnemo	kód	
	binary	octal
STL	111 001 010 000	7120

Odkazy

Podobné instrukce: CLL, CML.

SWP

Jméno

SWP — Swap MQ and AC

PDP-8/E

Přehled

SWP

L
-

Popis

Jsou prohozeny obsahy registrů MQ a AC. Instrukce je kombinací instrukcí MQA a MQL.

AC \longleftrightarrow MQ

mnemo	kód	
	binary	octal
SWP	111 101 010 001	7521

Poznámka: Tato instrukce byla implementována až v KE8-I, ale poprvé dokumentována v PDP-8/E.

Odkazy

Podobné instrukce: MQA, MQL.

SZA

Jméno

SZA — Skip on Zero Accumulator

Přehled

SZA

L
-

Popis

Pokud je hodnota registru AC nulová, tak instrukce přeskočí následující instrukci. Vykonávání programu pokračuje až další instrukcí za instrukcí následující.

If AC=0 then PC + 1 \longrightarrow PC

mnemo	kód	
	binary	octal
SZA	111 100 100 000	7440

Odkazy

Podobné instrukce: SKP, SNA, SZL.

SZL

Jméno

SZL — Skip on Zero Link

Přehled

SZL

L
-

Popis

Pokud je hodnota registru $L = 0$, tak instrukce přeskočí následující instrukci. Vykonávání programu pokračuje až další instrukcí za instrukcí následující.

If L=0 then PC + 1 \longrightarrow PC

mnemo	kód	
	binary	octal
SZL	111 100 011 000	7430

Odkazy

Podobné instrukce: SKP, SZA.

TAD

Jméno

TAD — Two's Complement Add

Přehled

TAD op

TAD I op

L
*

Popis

Obsah paměti určené operandem je přičten ke střadači a výsledek je uložen do střadače.

AC + operand \longrightarrow AC

mnemo	kód	
	binary	octal
TAD y	001 iza aaa aaa	1000
TAD y	001 0za aaa aaa	1000
TAD I y	001 1za aaa aaa	1400

Odkazy

Podobné instrukce: AND, DCA, ISZ.

TCF

Jméno

TCF — Clear Teleprinter Flag

Přehled

TCF

L
-

Popis

Příznak teleprinter je vymazán.

0 → Teleprinter Flag

mnemo	kód	
	binary	octal
TCF	111 000 100 010	7042

Odkazy

Podobné instrukce: TLF, TLS, TPC, TSF, TSK.

TLF

Jméno

TLF — Set the printer/punch flag

Přehled

TLF

L
-

Popis

mnemo	kód	
	binary	octal
TLF	111 000 100 000	7040

Odkazy

Podobné instrukce: TCF, TLS, TPC, TSF, TSK.

TLS

Jméno

TLS — Load Teleprinter Sequence

Přehled

TLS

L
-

Popis

Je vymazán příznak Teleprinteru a do TT0 nahrán znak z AC. Tento znak je pak vytištěn na terminálu.

0 → Teleprinter Flag
AC_{4...11} → TT0

mnemo	kód	
	binary	octal
TLS	111 000 100 110	7046

Ukázky

```

1          / File: db/pdp8.code/tls-example.pal  -- mode:asm; --
2
3      0400      *0400
4 00400 6041  WAIT,  TSF          / SKIP WHEN FREE TO PRINT
5 00401 5200          JMP WAIT    / REPEAT OTHERWISE
6 00402 6046      TLS          / LOAD TTO, PRINT OR PUNCH
7 00403 7402      HLT
8          $

```

No detected errors

Odkazy

Podobné instrukce: TCF, TLF, TPC, TSF, TSK.

TPC

Jméno

TPC — Load Teleprinter and Print

Přehled

TPC

L
-

Popis

Do registru TTO je uložen obsah spodních 8-mi bitů AC. Tento znak je pak vytištěn na terminálu.

$AC_{4...11} \longrightarrow TTO$

mnemo	kód	
	binary	octal
TPC	111 000 100 100	7044

Odkazy

Podobné instrukce: TCF, TLF, TLS, TSF, TSK.

TSF

Jméno

TSF — Skip on Teleprinter Flag

Přehled

TSF

L
-

Popis

Je zkoumán příznak teleprinter. Má-li hodnotu 1, je přeskočena následující instrukce.

```
If Teleprinter Flag = 1 then PC+1→PC
```

mnemo	kód	
	binary	octal
TSF	111 000 100 001	7041

Odkazy

Podobné instrukce: TCF, TLF, TLS, TPC, TSK.

TSK

Jméno

TSK — Skip the next instruction if printer/punch interrupt or keyboard/reader interrupt is set

Přehled

TSK

L
-

Popis

mnemo	kód	
	binary	octal
TSK	111 000 100 101	7045

Odkazy

Podobné instrukce: TCF, TLF, TLS, TPC, TSF.

WSR

Jméno

WSR — Write to Switch Register

HD-6120

Přehled

WSR

L
.

Popis

Obsah střadače AC je nahrán do SR, poté je střadač vymazán.

AC \longrightarrow SR

0 \longrightarrow AC

mnemo	kód	
	binary	octal
WSR	110 010 100 110	6246

Poznámka: Tato instrukce je implementována v mikroprocesoru HD-6120.

Odkazy

Podobné instrukce: OSR.

- HD6120 Specifications, strana 4-16

63.3. Základní a rozšířené periferie

Minipočítač PDP-8 lze vybavit dalšími moduly z velikého výběru. V této části některé z nich popíši.

63.3.1. Klávesnice a tiskárna konzoly (03, 04)

*

Obrázek 63-15. Keyboard/Reader Instruction Format

0	1	2	3	4	5	6	7	8	9	10	11
1	1	0	0	0	0	0	1	1			
									^	^	^--
										+	---
									+	-----	KSF
									0	0	0
									1	0	1
									1	1	0
											KCF
											KIE
											KRB

Tabulka 63-14. Příkazy pro práci s konzolou

mmemo	ok- talově	binárně	popis
KCF	6030	110 000 011 000	Keyboard Clear Flags
KSF	6031	110 000 011 001	Skip on Keaboard Flag
KCC	6032	110 000 011 010	Clear Keyboard Flag
KRS	6034	110 000 011 100	Read Keyboard Buffer Static
KRB	6036	110 000 011 110	Read Keyboard Buffer Dynamic
TLF	7040	111 000 100 000	Set the printer/punch Flag
TSF	7041	111 000 100 001	Skip on Teleprinter Flag
TCF	7042	111 000 100 010	Clear Teleprinter Flag
TPC	7044	111 000 100 100	Load Teleprinter and Print
TSK	7045	111 000 100 101	Skip if reader or printer interrupt Flag
TLS	7046	111 000 100 110	Load Teleprinter Sequence

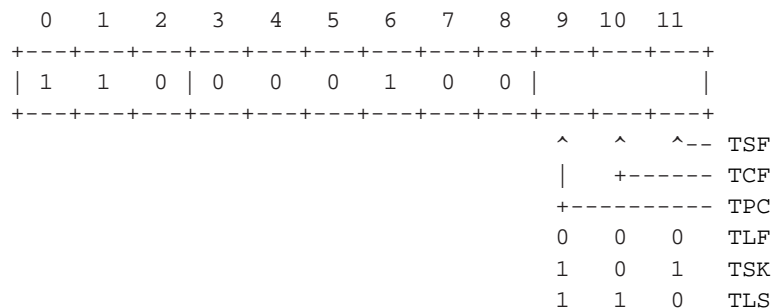
Příklad 63-2. Čtení jednoho znaku z konzoly

```

1          / File: db/pdp8.code/keybininput.pal -*- mode:asm; -*-
2
3      0200  *200
4 00200 6032 INPUT,  KCC          /CLEAR KEYBOARD FLAG
5 00201 4204          JMS LISN    /ENTER SUBROUTINE
6 00202 3211          DCA STORE   /STORE ASCII CHARACTER
7 00203 7402          HLT
8
9 00204 0000 LISN,  0          /LISN SUBROUTINE
10 00205 6031          KSF        /KEYBOARD FLAG RAISED YET?
11 00206 5205          JMP .-1    /NO: CHECK AGAIN
12 00207 6036          KRB        /YES: READ THE CHARACTER
13 00210 5604          JMP I LISN /RETURN TO MAINLINE
14
15 00211 0000 STORE,  0
16          $

```

No detected errors

Obrázek 63-16. Printer/Punch Instruction Format

Příklad 63-3. Vytisknutí jednoho znaku na konzoli

```

1          / File: db/pdp8.code/printoutput.pal -- mode:asm; --
2
3      0200 *200
4 00200 7300 OUTPUT, CLA CLL      /CLEAR ACCUMULATOR AND LINK
5 00201 6046      TLS            /RAISE PRINTER FLAG
6 00202 1213      TAD HOLD       /GET THE CHARACTER
7 00203 4205      JMS TYPE       /ENTER SUBROUTINE
8 00204 7402      HLT
9
10
11 00205 0000 TYPE, 0            /TYPE SUBROUTINE
12 00206 6041      TSF           /PRINTER FLAG RAISED YET?
13 00207 5206      JMP .-1       /NO: CHECK AGAIN
14 00210 6046      TLS           /YES: PRINT THE CHARACTER
15 00211 7300      CLA CLL       /CLEAR ACCUMULATOR AND LINK
16 00212 5605      JMP I TYPE    /RETURN TO MAINLINE
17
18 00213 0243 HOLD, 243         /STORED ASCII CHARACTER
19      $

```

No detected errors

63.3.2. DC02-F 8-Channel Multiple Teletype Control

*

O tomto zařízení jsem zatím nenašel žádnou jinou zmínku než jeho název a seznam instrukcí v knize [1] na straně D-12.

Tabulka 63-15. Instrukce DC02-F

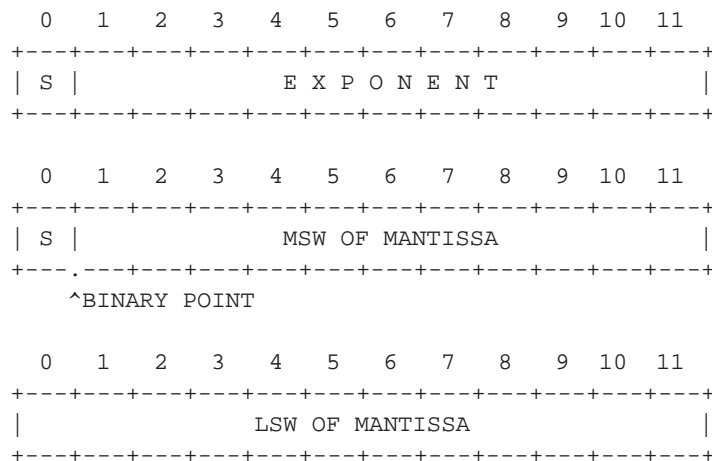
mnemo	ok- talově	binárně	popis
MTPF	6113	110 001 001 011	Read Transmitter Flag
MINT	6115	110 001 001 101	Set Interrupt Flip-Flop
MTON	6117	110 001 001 111	Select Specified Station
MTKF	6123	110 001 010 011	Read Receiver Flag Status
MINS	6125	110 001 010 101	Skip on Interrupt Request
MTRS	6127	110 001 010 111	Read Station Status
MKSF	6111	110 001 001 001	Skip on Key Board Flag
MKCC	6112	110 001 001 010	Clear Receive Flag
MKRS	6114	110 001 001 100	Receive Operation
	6116	110 001 001 110	Combined MKRS and MKCC
MTSF	6121	110 001 010 001	Skip on Transmitter Flag
MTCF	6122	110 001 010 010	Clear Transmitter Flag
MTPC	6124	110 001 010 100	Transmit Operation
	6126	110 001 010 110	Combined MTCF and MTPC

63.3.3. FFP-12 Floating Point Processor

Tento modul obsahuje samostatný procesor pro výpočty v plovoucí řádové čárce.

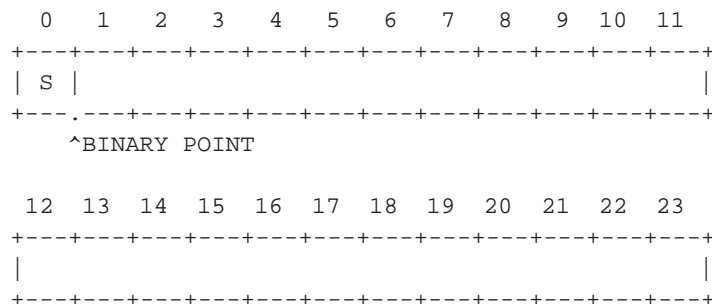
Data jsou ukládána ve vlastním 36-ti bitovém formátu s plovoucí řádovou čárkou. Jedno číslo je tedy reprezentováno třemi po sobě jdoucími slovy. První slovo obsahuje exponent a jeho znaménko. Druhé slovo obsahuje významnější část mantisy a její znaménko. Třetí slovo obsahuje 12 nejméně významných bitů mantisy. Pro vlastní výpočty je mantisa reprezentována jako 28 bitové slovo. Při ukládání do paměti se normalizuje na 24 bitů.

Obrázek 63-17. Formát dat FFP-12



Modul umožňuje také pracovat s čísly v pevné řádové čárce.

Obrázek 63-18. Formát Double Precision FFP-12



Tabulka 63-16. FFP-12 Active Parameter Table Format

P	Filed Bits of Operand Address	Field Bits of Base Reg.	Field Bits of Index Register Location	Field Bits of FPC
P+1	Lower 12 bits of FPC			
P+2	Lower 12 bits of index register location			
P+3	Lower 12 bits od Base Reg			
P+4	Lower 12 bits of operand address			
P+5	Exponent of FAC			
P+6	MSW of FAC			
P+7	LSW of FAC			

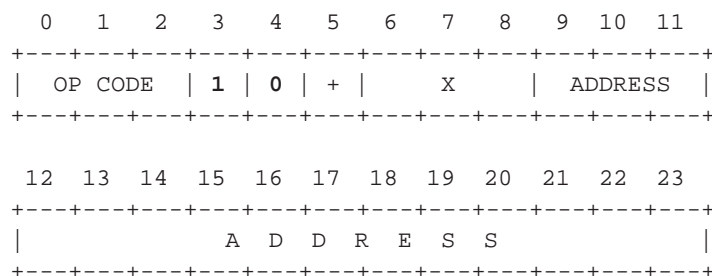
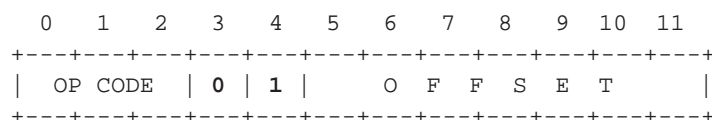
NOTE: APT address points to location P.

Tabulka 63-17. IOT instrukce pro FPP-12

název	kód_{oct}	popis
FPINT	6551	Skip on FPP "interrupt request" flag.
FPHLT	6554	Halt the processor at the end of the current instruction. Store active registers in core, set a status register bit, and the "interrupt request" flag.
FPCOM	6553	If the FPP is not running and the FPP "interrupt request flag" has been reset, set the command register to the contents of the AC. The three least significant bits of the AC set the field bits of the "Active Parameter Table" address. If the FPP is running or the interrupt request flag is set, the instruction is ignored.
FPICL	6552	Clear the FPP "interrupt request" flag.
FPST	6555	If the FPP is not running and the FPP "interrupt request flag" is reset, set the location of the "Active Parameter Table" to the contents of the AC, initiate the FPP and skip the next instruction. If the FPP is not running or the FPP "interrupt request flag" has not been reset, the instruction is ignored.
FPRST	6556	Read the FPP status register into the AC.
FPIST	6557	Skip on FPP "interrupt request" flag. If the skip is granted, clear the flag and read the FPP status request into the AC.

63.3.3.1. Instrukce FPP-12

Jak již bylo zmíněno, FPP-12 je zcela samostatný procesor která má také vlastní sadu instrukcí.

Obrázek 63-19. FPP-12 Double word direct addressing**Obrázek 63-20. FPP-12 Single-Word Addressing Format****Tabulka 63-18. Instruction Set**

kód	název	popis
0	FLDA	Load the FAC from the effective address.
1	FADD	Add the operand to the contents of FAC and store the result in the FAC.

kód	název	popis
5	FADDM	Add the operand to the contents of the FAC and store the results in the operand.
2	FSUB	Subtract the operand from the contents of the FAC and store the result in the FAC.
3	FDIV	Divide the operand into the contents of the FAC and store the results in the FAC.
4	FMUL	Multiply the contents of the FAC by the operand and store the result in the FAC.
7	FMULM	Multiply the contents of the FAC by the operand and store the result in memory.
6	FSTA	Replace the operand with the contents of the FAC.

63.3.4. RK8 (Cartridge Disk System)

* *Attributy: id="pdp8.RK8"*

Odkazy:

- PDP-8/a Minicomputer Handbook 1976-77, strana 9-116
-

Zařízení má obvyklé číslo 74, instrukce pro práci s ním mají tedy oktalový tvar 674x, kde x je funkce kterou se zařízením chceme provádět.

Zařízení RK8J-E sestává z řadiče RK8-E a z jednoho periferního zařízení RK05-J DECpack. Řadič RK8-E může obsluhovat až 4 jednotky RK05-J. V minipočítači mohou být dva řadiče RK8-E a tedy lze celkově obsluhovat 8 diskových jendotek.

Disk má 406 stop, a na stopě je 12 neb 16 sektorů. Jednotka RK05F má 812 stop. Rychlost otáčení disků ve všech zařízeních je 1500 rpm. Pro záznam je použita metoda NRZ. V zařízení RK05J může být uloženo 25 miliónů bitů a v zařízení RK05F dvojnásobek, tedy 50 miliónů bitů.

Řadič RK8-E má celkem čtyři 12-ti bitové registry:

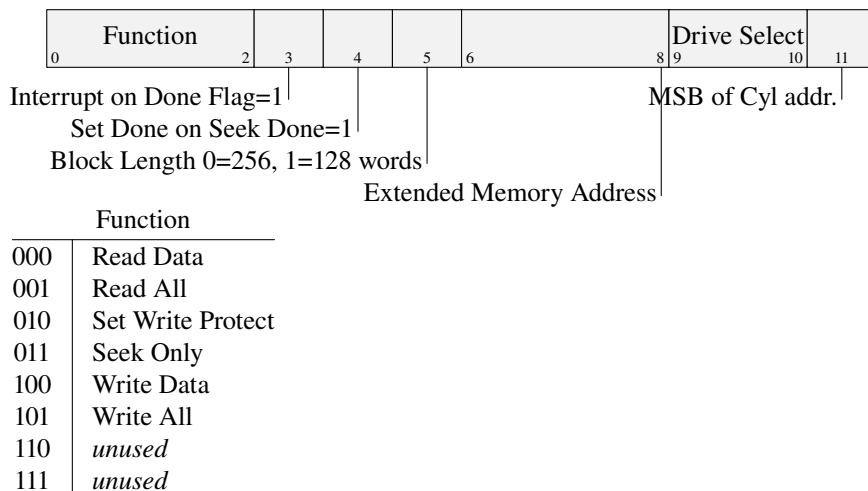
- Command Register
- Current Address Register
- Disk Address Register
- Status Register

Tabulka 63-19. Instrukce pro práci s RK8

mnemo	ok- talově	binárně	popis
DSKP	6741	110 111 100 001	Disk Skip on Flag
DCLR	6742	110 111 100 010	Disk Clear
DLAG	6743	110 111 100 011	Load Address and Go
DLCA	6744	110 111 100 100	Load Current Address
DRST	6745	110 111 100 101	Read Status
DLDC	6746	110 111 100 110	Load Command
DMAN	6747	110 111 100 111	Maintenance Instruction

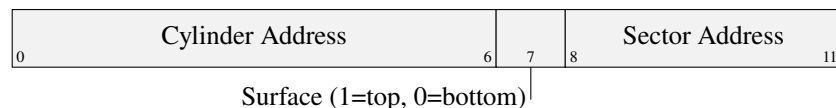
Obrázek 63-21. Příkazový registr RK8 (instrukce 6746)

Command Register (DLDC) **RK8**.



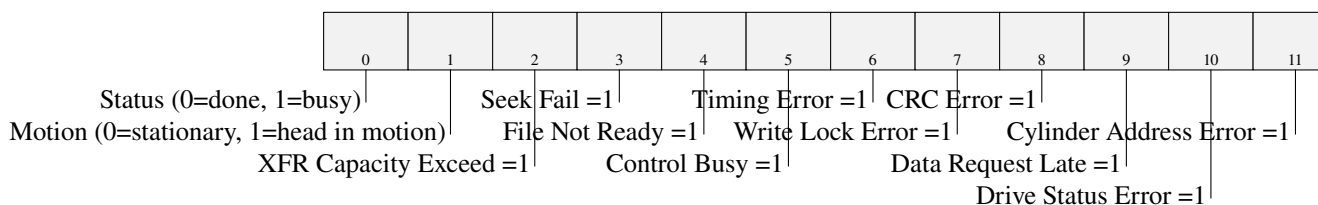
Obrázek 63-22. Adresový registr RK8 (instrukce 6743)

RK8-E Address Register (DLAG)



Obrázek 63-23. Stavový registr RK8 (instrukce 6745)

RK8-E Status Register (DRST)



63.4. Varianty a rozšíření minipočítače

63.4.1. PDP-8/A, KM8-A/KM8-E

63.4.1.1. Rozšíření paměti

Tato varianta minipočítače dokáže adresovat až 128k slov operační paměti.

Tabulka 63-20. KT8-A Compatible Instructions

kód	mnemonic	popis
62N1	CDF	Change Data Field.
62N2	CIF	Change Instruction Field Buffer
62N3	CDF CIF	Kombinace abou instrukcí CDF a CIF.
6004	GTF	Get Flags. Nahraje obsa save field register a CPU status bits do AC.
6005	RTF	ResTore Flags
6204	CINT	Clear user INTerrupt flag
6214	RDF	
6224	RIF	
6234	RIB	Read Interrupt Buffer
6244	RMF	Read Memory Field
6254	SINT	Skip if user INTerrupt flag is set
6264	CUF	Clear User mode flip-flop and buffer
6274	SUF	

Tabulka 63-21. KT8-A Expanded instructions

kód	mnemonic	popis
6 01x nnn y01	CDF	
6 01x nnn y10	CIF	
6 01x nnn y11	CDF CIF	
6170	LBM	
6171	RBM	
6172	RLB	
6173	RMR	
6174	MBC	
6175	RACA	
6176	RACB	
6177	RACC	
6204	CINT	
6210	GTS	
6214	RDF	
6220	RTS	
6224	RIF	
6230	RXM	
6234	RIB	
6240	LRR	
6244	RMF	

kód	mnemonic	popis
6250	RRR	
6254	SINT	
6260	LUSR	
6264	CUF	
6270	RUSR	
6274	SUF	
6101	SBE	
6102	SPL	
6103	CAL	

63.5. Programové vybavení

Odkazy:

- THIRD²¹
-
-
-

Operační systémy:

- OS/8
- RTS/8 — Real-Time Operating System
- TSS/8 — Time-Sharing System — multiuživatelský systém pro 16 uživatelů (v některých aplikacích pro 24)
-

63.5.1. OS/8

*

63.5.1.1. RX02 Bootstrap

1. Press in order the MD and DISP buttons to display memory data in the octal readout.
2. Press in order 0 and LXA to select memory field 0.
3. Press in order 20 and LA to start loading instructions at location 20.
4. Deposit the following octal values, termination each value with D NEXT.


```

1          / File db/pdp8.code/rx02boot.pal -- mode:asm; --
2      7301 AC1=CLL CLA IAC
3      7326 AC2=CLL CLA CML RTL
4      7327 AC6=CLL CLA CML IAC RTL          /RX02'S MUST RUN ON AN OMNI-BUS !!
5      7330 AC4000=CLL CLA CML RAR
6      7350 AC3777=CLL CLA CMA RAR
7      7346 AC7775=CLL CLA CMA RTL
8          /
9          / DEVICE IOT SYMBOLIC EQUATES
10         /
11      6751 LCD=6751          /LOAD COMMAND
12      6752 XDR=6752          /TRANSFER DATA
13      6753 STR=6753          /SKIP IF READY TO TRANSFER
14      6754 SER=6754          /SKIP ON ERROR
15      6755 SDN=6755          /SKIP ON DONE
16         /
17         /
18         /
19      0020          *20
20         /
21 00020 1061 READ, TAD UNIT /TRY NEXT COMBINATION OF DENSITY AND UNIT
22 00021 1046 TAD CON360 /ADDING IN 360
23 00022 0060 AND CON420 /KEEPING ONLY 420 BITS
24 00023 3061 DCA UNIT /CYCLES 400, 420, 0, 20, 400,,,,,
25 00024 7327 AC6 /COMMAND TO READ DISK
26 00025 1061 TAD UNIT /UNIT AND DENSITY
27 00026 6751 LCD /COMMAND TO CONTROLLER
28 00027 7301 AC1 /TO SET SECTOR AND TRACK TO 1
29 00030 4053 JMS LOAD /SECTOR TO CONTROLLER, LEAVES AC ALONE
30 00031 4053 JMS LOAD /AND TRACK
31 00032 7004 LITRAL, 7004 /LEAVING A 2 IN AC; SERVERS AS LITERAL
32         /
33         / FOLLOWING IS PART OF WAIT LOOP, SAME SECONDARY BOOTS, OLD PRIMARY
34         /
35 00033 6755 START, SDN /HAS DONE COME UP; CODE STARTS HERE!
36 00034 5054 JMP LOAD+1 /NO, GO CHECK FOR READY TO TRANSFER
37         /
38         / NOW, DONE OR ERROR
39         /
40 00035 6754 SER /SKIP ON AN ERROR, TRY ANOTHER DENSITY ETC.
41 00036 7450 SNA /NASTY, AC=2 FOR ABOUT TO DO SILO, 0 ON START
42 00037 5020 JMP READ /START-UP, GO SET UP UNIT, THEN READ TO SILO
43 00040 1061 TAD UNIT /AC ALREADY 2, PUT IN UNIT, DENSITY
44 00041 6751 LCD /TO EMPTY THE SILO
45 00042 1061 TAD UNIT /SET UP LOC 60 FOR OLD SECONDARY BOOT
46 00043 0046 AND CON360 /KEEPING ONLY DENSITY BIT
47 00044 1032 TAD LITRAL /ADDING IN 7004, BECAUSE THAT'S WHAT SYS WANT
48 00045 3060 DCA RX1SAV /OLD SECONDARY BOOT MOVES IT TO HANDLER
49 00046 0360 CON360, 360 /LITERAL; EXECUTES IN LINE AD A NO-OP
50         / /FALLS THRU TO NEXT PAGE OF LISTING
51         /
52         / FOLLOWING CODE SAME AS OLD PRIMARY BOOT
53 00047 4053 JMS LOAD /GRAB NEXT ITEM FROM SILO
54 00050 3002 DCA 2 /TRADITION; SECONDARY BOOT STARTS LOADING AT
55 00051 2050 ISZ 50 /INCREMENT LOAD ADDRESS
56 00052 5047 JMP 47 /GO BACK FOR ANOTHER

```

```

/
PALBART V2.3a Page 1

57 /
58 / SECONDARY BOOT LOADS OVER PRIMARY BOOT UNIT LOCATION 47 IS LOADED,
59 / THEN CONTROL PASSES TO SECONDARY BOOT
60 /
61 00053 0000 LOAD, 0 /SUBROUTINE TO GIVE AND TAKE DATA FROM CONTR
62 00054 6753 STR /IS HE READY TO TALK TO US?
63 00055 5033 JMP START /NO, IS HE PERHAPS DONE WITH SILO, OR IN ERRO
64 00056 6752 XDR /YES, DATA IN OR OUT;IF DATA TO CONTROLLER AC
65 00057 5453 JMP I LOAD /NO MAGIC, JUST EXIT FROM SUBROUTINE
66 /
67 / 60 GOES TO OLD SECONDARY BOOT
68 / 61 HAS DENSITY AND UNIT THAT BOOTED SUCCESSFULLY
69 /
70 CON420, /USE IT TO HOLD 420 LITERAL TO START OUT
71 00060 0420 RX1SAV, 420 /UNIT^20+7004 TO GO TO SYS HANDLER
72 00061 0020 UNIT, 20 /<DENSITY^400>+<UNIT^20> THAT BOOTED OK
73 $
No detected errors
```

- 5. After you have deposited all the values, press 0033 and LA to start the program at location 33.
- 6. To start the bootstrap program, press INIT and RUN.
- 7.

63.5.2. TSS/8

* *Attributy:* id="pdp8.TSS/8"

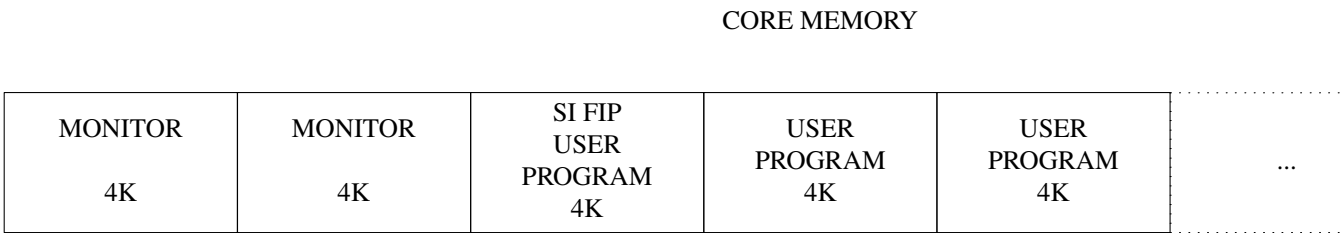
Odkazy:

- TSS-8²² na Wikipedii
- TSS8_8.24_ManagersGuide.pdf²³
-
-
-

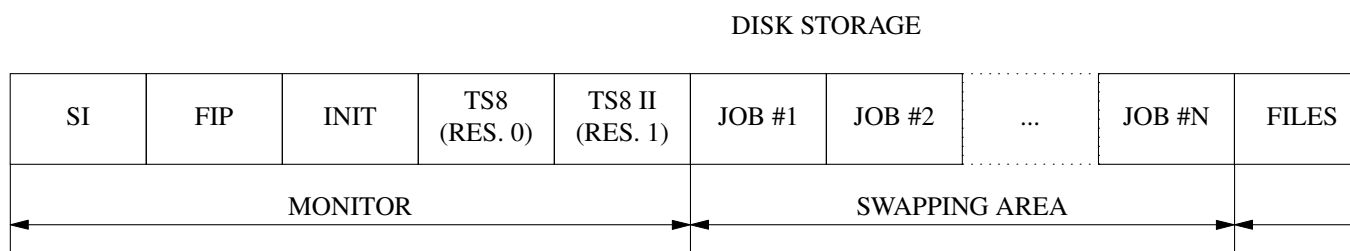
Poznámky k TSS/8

Paměť je přidělována jednotlivým procesům po celých 4KW stránkách. První dvě stránky paměti zaujímají samotné procesy Monitoru.

Obrázek 63-24. Mapa operační paměti



Obrázek 63-25. Mapa operační disku



- EduSystem 50
- EduSystem 30

Z pohledu uživatele má každý uživatel k dispozici terminál a PDP-8 s 4KW operační pamětí.

Protože nemusíme mít v reálném systému dostatek operační paměti pro všechny současně připojené uživatele, provádí TSS/8 odkládání. Pro každého uživatele je na disku 4K slov odkládací paměti kam se jeho operační paměť odloží když je potřeba vykonat proces jiného uživatele.

Hardware základního počítače je upraven. Je do něj přidáno rozlišení mezi uživatelským a systémovým režimem. V systémovém režimu neplatí žádná omezení. V uživatelském režimu jsou tato omezení:

- čtení pomocí OSR není dovoleno, způsobí přerušení
- zastavení procesoru instrukcí HLT není dovoleno, způsobí přerušení
- vykonání jakékoliv IOT instrukce způsobí přerušení

Protože vykonání IOT instrukce způsobí v uživatelském režimu přerušení, může TSS/8 zajistit sdílení aktuálního hardware. Nejen to, umožňuje simulovat virtuální hardware který není ke skutečnému počítači připojen. Služby systému TSS/8 jsou přístupny přes "virtuální" IOT instrukce.

Přesný název módů procesoru je *executive mode* a *user mode*.

Poznámka: Mikrokódované instrukce OPR2 s HLT a OSR proběhnou celé normálně s výjimkou HLT a OSR a teprve poté je vyvoláno přerušení.

Každý uživatel má přiřazeno identifikační číslo, podle kterého jej systém rozpozná a sleduje. Toto číslo se používá k přihlašování (LOGIN).

Minimální konfigurace pro běh EduSystem 50

- PDP-8, 8I nebo 8E s nejméně 12KW paměti.
- RF08 alespoň s jedním RS08, může být použit DF32 s dvěma disky
- Připojení terminálových linek pomocí jednoho nebo více adaptérů KL8E, PT08 nebo DC08A
- všechny konfigurace vyjma PDP-8I s DC08A potřebují zdroj "real-time clock"

Další, podporovaný, hardware

- až 32KW paměti
- DC08A smí být použit jen na PDP-8I a jen jako přídavek k PT08
- 689AG modemový řadič použitý jen s DC08A

- jsou podporována všechna standardní EAE rozšíření s výjimkou tradičního PDP-8 "step counter" který není uchovávan ani obnovován
- vysokorychlostní čtečka děrné pásky, je vyžadována v případě sestavování TSS/8. může být užita i pomalá (standardní) čtečka ale sestavení pak trvá dlouho
- vysokorychlostní děrovačka děrné pásky
- řádková tiskárna LP08/LE8 nebo LS08/LS8E
- páskové jednotky TC01 nebo TC08, může být připojeno až 8 páskových mechanik
- až čtyři disky RS08
- čtečka děrných štítků
- až čtyři jednotky RK8E

Software sestává z několika programů. Ve zkratce to jsou:

- INIT — inicializační program
- SI — System Interpreter
- FIP — File Phantom
- TS8 a TS8II — dvě sekce monitoru které jsou vždy v paměti v polích 0 a 1
-

Diskový prostor je dělen na "stopy" (*tracks*) o pevné velikosti 4KW. Segment je definován jako dvě stránky operační paměti, tedy 256 slov (400 oktalově). Velikosti všech souborů jsou měřeny v segmentech.

Zavádění a inicializace:

1. zapněte počítač (POWER ON)
2. zapněte terminál operátora (ON)
3. zapněte vysokorychlostní čtečku děrné pásky
4. zastavte počítač přepínačem HALT
5. přes pření panel zaved' te do počítače RIM zavaděč pro vysokorychlostní čtečku děrné pásky
6. odstartujte RIM
7. Po úspěšném zavedení INIT z děrné pásky se objeví výzva
LOAD, DUMP, START, ETC?
8. ⋮

Jak jsem již zmínil, uživatelé jsou označeni čísly. Důležití uživatelé mají tato čísla:

Tabulka 63-22. Důležití/systémoví uživatelé TSS/8

číslo	název
1	SYSTEM
2	LIBRARY
3	OPERATOR

Noví uživatelé moho být vytváření jen uživatelem 1 (SYSTEM). Každý uživatel má přidělena dvě čísla, číslo projektu a číslo programátora. Přioktalovém zápisu první dvě číslice (6 bitů) jsou číslo projektu, druhé dvě číslice

pak číslo programátora.

Založení nového účtu

1. Nejdříve se přihlásíme jako uživatel 1 (SYSTEM) jenž má v naší ukázce heslo SYST.

```
. LOGIN 1 SYST
```

2. Spustíme program pro správu účtů LOGID.

```
. R LOGID
```

- 3.

```
PLEASE ENTER DISK QUOTA: 200
*
```

63.5.2.1. Programy a nástroje TSS/8

*

Programy a nástroje ve zkratce

CAT

```
R CAT
R CAT:R      !!!
R CAT:L
```

LOADER

Nahrává soubory ve formátu BIN z disku do paměti. Když se ptá, zadáme mu seznam souborů oddělených čárkami. Volba (*option*) :D aktivuje ladění pomocí ODT. Pozor na adresy, LOADER nenahraje korektně oduřady 7767 až 7777. Je-li použit ODT musí se pro něj vyhradit adresy 4 a 7000 až 7777.

LOGID

Systémový manažer (účet 1) může pomocí tohoto programu pracovat s účty uživatelů.

LOGOUT

PIP

Peripheral Interchange Program.

Program přenáší soubory mezi médii. Použijte místo něj novější program PUTR.

SYSTAT

Program vypisuje stav systému.

PUTR

Program určený pro přenos souborů mezi zařízeními.

PTLOAD

TSS/8 verze binárního zavaděče.

GRIPE

Program umožňuje každému uživateli zanechat vzkaz pro systémového operátora.

BASIC

FOCAL

63.5.2.2. Monitor a jeho příkazy

*

Na jednom řádku může být více příkazů oddělených znakem ';'.

R

Spuštění programu

EXAMI adresa počet-slov

EXAM

EXA

DEPOSIT adresa hodnota

START adresa

63.6. Datové formáty

*

63.6.1. Formáty děrné pásky

*

Data na děrnou pásku jsou ukládána ve formátu pro zavaděče RIM a SS BIN. Formát pásky je následující:

1. Informace na pásce začínají a končí kódem 200_{OCT}.

2. Jako první je uložen samospustitelný binární zavaděč jenž je kódován v RIM.
3. Checksum of SS BIN or two frames of leader/trailer.
4. Leader/trailer or blank tape.
5. Program to be loaded, beginning with an origin setting. If it is to be loaded into a field other than the field of the loaders, it must also begin with a field setting.
6. An origin setting at the end of the program, if it is to be started by SS BIN.
7. Checksum of the program portion of the tape.
8. Leader/trailer.

Obrázek 63-26. Příklad

děrná páska	význam	poznámka
87 654 321		
10 000 000	loader/trailer	
11 011 000	field setting	Bity 4,5 a 6 obsahují číslo pole, v tomto případě 3.
01 000 010	origin setting	
00 011 100		
00 111 110	data word	
00 101 100		

63.6.2. RIM

* *Attributy: id="pdp8.rim"*

Odkazy:

- DEC-08-PMP0-D Readin Mode²⁴
- DEC-08-LRAA-D Read In Mode Loader²⁵
-

63.6.3. BIN

* *Attributy: id="pdp8.bin"*

Odkazy:

- DEC-08-LBAA-D Binary Loader²⁶
-

```

00 000 000
...
00 000 000
10 000 000  Loader START
01 000 010  Origin 0200
00 000 000

00 111 010  Contents of 0200 is 7200
00 000 000

```

```
00 011 010  Contents of 0201 is 3276
00 111 110

00 111 100  Contents of 0201 is 7402
00 000 010
:
00 000 100  Checksum.  Program determines this is checksum because
00 010 010  trailer follows.

10 000 000  Trailer
```

63.7. Simulátory PDP-8

*

63.7.1. SIMH

* *Attributy:* `id="pdp8.simh"`

Odkazy:

- Emulating a DEC PDP-8 with SimH²⁷
-

Seznam zařízení jenž jsou emulována.

```
sim> show dev
PDP-8 simulator configuration
```

```
CPU, idle disabled
TSC, disabled
CLK, 60Hz, devno=13
PTR, devno=01
PTP, devno=02
TTI, devno=03
TTO, devno=04
TTIX, devno=40-47
TTOX, 4 units
LPT, devno=66
RK, devno=74, 4 units
RL, disabled
RX, RX8E, devno=75, 2 units
DF, disabled
RF, devno=60-64
DT, devno=76-77, 8 units
TD, disabled
MT, devno=70-72, 8 units
CT, disabled
```

- CPU —
-
-
-

Tabulka 63-23. Simulovaný hardware

název	popis
CPU	PDP-8/E CPU s 4-32KW paměti, KE8E (EAE), KM8E (správa paměti a sdílení času)
TSC	TSC8-75, sdílení času pro operační systém ETOS
PTR, PTP	PCE8E, čtečka a děrovačka děrné pásky
TTI, TTO	KL8E terminál
TTIX, TTOX	KL8JA, přidané terminály
LPT	LE8E, řádková tiskárna
CLK	DK8E, line frequency clock (PDP-8/A kompatibilní
RK	RK8E/RK05, řadič kazetových disků, až 4 disky
RF	RF08/RS08, řadič disku s pevnými hlavami, 1-4 disky
DF	DF32/DS32, řadič pevných disku s pevnými hlavami, 1-4 disky
RL	RL8A/RL01, řadič kazetových disků, až 4 disky
RX	RX8E/RX01, RX28/RX02, řadič disket, 2 mechaniky
DT	TC08/TU56 DECtape, řadič až 8 páskových jednotek
TD	TD8E/TU56 DECtape, řadič 2 páskových jednotek
MT	TM8E/TU10, řadič 8-mi páskových jednotek
CT	TA8E/TU60, řadič kazetopáskových jednotek, 2 kazetové mechaniky

Zařízení povolíme a zakážeme v konfiguraci počítače příkazy:

```
simh> SET dev ENABLED
simh> SET dev DISABLED
```

63.7.1.1. CPU

Tabulka 63-24. Nastavování procesoru

příkaz	popis
SET CPU EAE	povolení EAE
SET CPU NOEAE	zakázání EAE
SET CPU 4K	nastavení velikosti operační paměti na 4K slov
SET CPU nK	nastavení velikosti operační paměti na 4, 8, 12, 16, 20, 24, 28 a 32K slov
SET CPU [NO]IDLE	zapnutí [vypnutí] detekce čekajícího (idle) procesoru
SET CPU HISTORY	
SHOW CPU HISTORY	

63.7.1.2. Nastavení KL8E (TTI,TTO)

*

63.8. Technologie programování PDP-8

* *Attributy:* id="pdp8.programming"

63.8.1. Podprogramy

Strojový kód PDP-8 má k dispozici instrukci JMS pro volání podprogramu, ale nemá instrukci pro návrat. Instrukce JMS funguje tak, že návratovou adresu ukládá do prostoru rutiny která je volána. Návrat z rutiny lze uskutečnit pomocí instrukce skoku JMP.

```

...
JMS ALFA
...

/ Podprogram (subroutine) ALFA
ALFA,    0000    / ← zde se ukládá návratová adresa
/ ... tělo podprogramu
JMP I ALFA / návrat z podprogramu skokem na návratovou adresu

```

Příklad 63-4. Příklad podprogramu pro PDP-8

```

1          / File db/pdp8.code/subroutine.pal -- mode:asm; --
2
3          0400          *0400
4 00400 7000  START,  NOP /...
5 00401 4206          JMS ALFA
6 00402 7000          NOP /...
7 00403 4206          JMS ALFA
8 00404 7000          NOP /...
9 00405 7402          HLT
10
11          / Podprogram (subroutine) ALFS
12 00406 0000  ALFA,    0000          / místo pro uložení návratové adresy
13 00407 7000          NOP          /... kód podprogramu
14 00410 5606          JMP I ALFA    / návrat skokem na uloženou návratovou adresu
15          $

No detected errors

```

Tento způsob realizace podprogramů neumožňuje rekurzi. Pokud bychom rekurzi potřebovali, musíme si zásobník návratových adres sami vytvořit a udržovat.

63.8.2. Logické operace

Odkazy:

- Other Logical Operations²⁸
- PDP-8 Instruction Set Musings²⁹

Příklad 63-5. Logický součet (OR) pro PDP-8

```

/ AC ∨ M → AC
/ De Morgan Law
/ AC ∨ M ≡ ¬(¬AC ∧ ¬M)
CMA                / ¬AC → AC
DCA      TMP      / AC → TMP
TAD      M        / (AC is zero from DCA) 0+M → AC
CMA                / ¬AC → AC
AND      TMP      / ¬M ∧ ¬TMP → AC
CMA                / ¬(¬M ∧ ¬AC) ≡ M or AC → AC

```

Příklad 63-6. Exclusive OR pro PDP-8

```

/ a xor b = (a+b) - 2*(a and b)
DCA      TMP
TAD      TMP
AND      M
CMA IAC
CLL RAL
TAD      TMP
TAD      M

```

Optimalizovaný kód pro výpočet XOR založený na vzorci:

$$A \text{ xor } B = A + B - 2*(A \text{ and } B)$$

Příklad 63-7. Exklusivní logický součet XOR, pro PDP-8

```

1          / File: db/pdp8.code/xor.pal  -*- mode:asm; -*-
2          / From: http://www.grc.com/pdp-8/isp-musings.htm
3
4          0400          *0400
5
6 00400 7200          CLA          / Load AC with ArgOne
7 00401 1207          TAD ArgOne    /+
8 00402 0210          AND ArgTwo    / Add ArgTwo. AC is ArgOne + ArgTwo
9 00403 7104          CLL RAL
10 00404 7041          CMA IAC
11 00405 1207          TAD ArgOne
12 00406 1210          TAD ArgTwo
13
14 00407 0000 ArgOne, 0
15 00410 0000 ArgTwo, 0
16          $

```

No detected errors

63.8.3. Aritmetické operace

V základním repertoáru jsou pouze sčítání a negace (-). Odčítání realizujeme kódem.

Příklad 63-8. Odčítání na PDP-8

DCA	TMP	/	$AC \rightarrow M, 0 \rightarrow AC$
TAD	M	/	$M \rightarrow AC$
CMA	IAC	/	$\neg AC \rightarrow AC$
TAD	TMP	/	$AC = AC - M$

63.8.4. Práce se zásobníkem

*

Odkazy:

- Software stack³⁰ na stránce PDP-8 na Wikipedii
-

Architektura PDP-8 nemá žádný hardwarový zásobník nebo instrukce které by při jeho realizaci v programu byly nápomocny. I přesto se ale v programu zásobník realizovat dá. Následující procedury realizují zásobníkové operace programově. V našem případě je zásobník oblast paměti od adresy 0377 směrem k nižším adresám. Zásobník tedy „roste“ směrem od vyšších k nižším adresám. Ukazatel zásobníku je v paměťové buňce označené SP a ukazuje vždy na poslední uloženou hodnotu. Proto se při inicializaci zásobníku nastavuje na adresu u jedničku vyšší než je paměť vyhrazená pro zásobník.

Příklad 63-9. Operace se zásobníkem, PUSH a POP, pro PDP-8

```

1          / File: db/pdp8.code/stack-push.pal -*- mode:asm; -*-
2          / From: http://en.wikipedia.org/wiki/PDP-8#Software_stack
3          0400          *400
4 00400 0000  PUSH,    0          / Value in AC
5 00401 3222          DCA DATA    / Temporary save value to push.
6 00402 7240          CLA CMA      / SP--
7 00403 1223          TAD SP        /+
8 00404 3223          DCA SP        /+
9 00405 1222          TAD DATA    / Load saved value to push
10 00406 3623         DCA I SP      / Store value to Mem[SP]
11 00407 5600         JMP I PUSH    /Return
12
13 00410 0000  POP,    0
14 00411 7300          CLA CLL      /
15 00412 1623          TAD I SP     / Load AC with Mem[SP]
16 00413 2223          ISZ SP       / SP++
17 00414 5610         JMP I POP     / Return, value in AC
18
19          / Stack initialization procedure
20 00415 0000  STKINI, 0
21 00416 7300          CLA CLL      / SP=STKBOT
22 00417 1224          TAD STKBOT   /+
23 00420 3223          DCA SP       /+
24 00421 5615         JMP I STKINI  / Return
25
26 00422 0000  DATA,  0          / Temporary place for pushed value
27 00423 0000  SP,    0          / Stack pointer. Must be initialized before using stack!
28 00424 0400  STKBOT, 400        / Bottom of the stack is 377.
29          $

```

No detected errors

63.8.5. Jednostranně vázaný seznam (*linked list*)

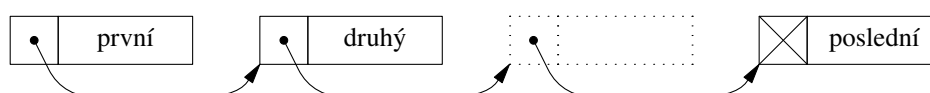
*

Odkazy:

- [Linked List](#)³¹ na stránce PDP-8 na Wikipedii
-

* Obrázek neodpovídá programu!

Obrázek 63-27. Linked List



Příklad 63-10. Práce s jednostranně vázaným seznamem na PDP-8

```

1          / File: db/pdp8.code/linked-list.pal  -- mode:asm; --
2          / From: http://en.wikipedia.org/wiki/PDP-8#Linked_list
3
4          0400          *0400
5 00400 0000  GETN,    0          /Gets the number pointed to and moves the pointer
6 00401 7300          CLA CLL          / AC = Mem[PTR]
7 00402 1610          TAD I PTR          /+
8 00403 3211          DCA TEMP          / Save the AC temporarily
9 00404 2210          ISZ PTR          / PTR++
10 00405 1610          TAD I PTR
11 00406 3210          DCA PTR
12 00407 5600          JMP I GETN          /Return
13 00410 0000  PTR,    0
14 00411 0000  TEMP,   0
15                                     $

```

No detected errors

Literatura

Knihy

[IntProg75] *introduction to programming*, Fifth Edition, April 1975, 1970, 1971, 1972, 1973, 1975.

Poznámky

1. <http://www.cs.uiowa.edu/~jones/pdp8/>
2. <http://en.wikipedia.org/wiki/PDP-8>
3. <http://homepage.mac.com/dgcx/pdp8x/>
4. http://web.archive.org/web/20021221184100/www.ulib.org/webRoot/Books/Saving_Bell_Books/SBN+Computer+Strucutres/c
5. <http://www.digibarn.com/collections/books/programming-languages-pdp8/index.html>

6. <http://www.faqs.org/faqs/dec-faq/pdp8/>
7. <http://www.faqs.org/faqs/dec-faq/pdp8/>
8. <http://www.faqs.org/faqs/dec-faq/pdp8/>
9. <http://ed-thelen.org/comp-hist/pdp-5.html>
10. http://www.bitsavers.org/pdf/dec/pdp11/memos/690304_Proposed_PDP-11_Re-Organization.pdf
1. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/Small_Computer_Handbook_1973.pdf
1. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/SmallComputerHandbook_1970.pdf
2. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/Small_Computer_Handbook_1973.pdf
1. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/SmallComputerHandbook_1970.pdf
2. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/Small_Computer_Handbook_1973.pdf
1. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/Small_Computer_Handbook_1973.pdf
1. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/SmallComputerHandbook_1970.pdf
2. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/Small_Computer_Handbook_1973.pdf
1. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/SmallComputerHandbook_1970.pdf
2. http://www.bitsavers.org/pdf/dec/pdp8/handbooks/Small_Computer_Handbook_1973.pdf
21. <http://anachrona.homeunix.com:8000/~rivie/third/>
22. <http://en.wikipedia.org/wiki/TSS-8>
- 23.
24. http://www.bitsavers.org/pdf/dec/www.computer.museum.uq.edu.au_mirror/DEC-08-PMP0-D_Readin_Mode_%5bRIM%5d_Punch.pdf
25. http://www.bitsavers.org/pdf/dec/www.computer.museum.uq.edu.au_mirror/DEC-08-LRAA-D_Read_In_Mode_Loader.pdf
26. http://www.bitsavers.org/pdf/dec/www.computer.museum.uq.edu.au_mirror/DEC-08-LBAA-D_Binary_Loader.pdf
27. <http://techtinkering.com/articles/?id=27>
28. <http://www.cs.uiowa.edu/~jones/pdp8/man/mri.html#logic>
29. <http://www.grc.com/pdp-8/isp-musings.htm>
30. http://en.wikipedia.org/wiki/PDP-8#Software_stack
31. http://en.wikipedia.org/wiki/PDP-8#Linked_list

Kapitola 64. VAX

*

64.1. Instrukční sada

64.1.1. Adresní módy

Tabulka 64-1.

číslo módu	název	zápis	popis
0-3	Literal	$S^{\#}literal$	Dolní dva bity módu se přidají k zbývajícím 4 bitům a formují 6-ti bitový literál. operand=ZEXT(literal)
4	Index		$OA = BOA + \{size * (R_x)\};$ operand=(OA)
5	Register	R_n	
6	Register Deferred	(R_n)	
7	Autodecrement	$-(R_n)$	$R_n \leftarrow R_n - size;$ $OA = R_n;$ operand=(OA)
8	Autoincrement		$OA = R_n;$ $R_n \leftarrow R_n + size;$ operand=(OA)
9	Autoincrement Deferred	$@(R_n)+$	$OA = (R_n);$ $R_n \leftarrow R_n + 4;$ operand=(OA)
10	Displacement B	$B^d(R_n)$	$OA = R_n + SEXT(displ);$ operand=(OA)
11	Displacement Deferred B	$@B^d(R_n)$	$OA = R_n + SEXT(displ);$ operand=(OA)
12	Displacement W	$W^d(R_n)$	$OA = (R_n + SEXT(displ));$ operand=(OA)
13	Displacement Deferred W	$@W^d(R_n)$	$OA = (R_n + SEXT(displ));$ operand=(OA)
14	Displacement L	$L^d(R_n)$	$OA = R_n + displ;$ operand=(OA)
15	Displacement Deferred L	$@L^d(R_n)$	$OA = (R_n + displ);$ operand=(OA)

64.1.2. Instrukce

Následující tabulka je velmi stručným přehledem instrukcí.

Tabulka 64-2. Instrukce řazené podle operačního kódu

opcode	mnemonic	name
58	ADAWI	Add Aligned Word Interlocked
78	ASHL	Arithmetic Shift Long
79	ASHQ	Arithmetic Shift Quad
80	ADDB2	Add Byte 2 Operand
81	ADDB3	Add Byte 3 Operand
88	BISB2	Bit Set Byte 2 Operand
8A	BICB2	Bit Clear Byte
8B	BICB3	Bit Clear Byte
A0	ADDW2	Add Word 2 Operand
A1	ADDW3	Add Word 3 Operand
AA	BICW2	Bit Clear Word
AB	BICW3	Bit Clear Word
C0	ADDL2	Add Long 2 Operand
C1	ADDL3	Add Long 3 Operand
CA	BICL2	Bit Clear Long
CB	BICL3	Bit Clear Long
D8	ADWC	Add With Carry
00	HALT	
01	NOP	
BA	POPR	
BB	PUSHR	
FC	XFC	

HALT

Jméno

HALT — Halt

Přehled

HALT

N	Z	V	C
.	.	.	.

Popis

Instrukce zastaví procesor. Tato instrukce je privilegovaná a pokud je použita v neprivilegovaném módu, způsobí výjimku.

opcode	mnemo	name
00	HALT	Halt

Odkazy

Podobné instrukce: HALT.

NOP

Jméno

NOP — No Operation

Přehled

NOP

N	Z	V	C
-	-	-	-

Popis

Nevykoná nic.

opcode	mnemo	name
01	NOP	No Operation

Odkazy

Podobné instrukce: NOP.

POPR

Jméno

POPR — Pop Registers

Přehled

POPR *mask.rw*

Poznámky k tvaru instrukce

N	Z	V	C
-	-	-	-

Popis

opcode	mnemo	name
BA	POPR	Pop Registers

Odkazy

Podobné instrukce: HALT.

PUSHR

Jméno

PUSHR — Push registers

Přehled

PUSHR *mask.rw*

Poznámky k tvaru instrukce

N	Z	V	C
-	-	-	-

Popis

opcode	mnemo	name
BB	PUSHR	Push Registers

Odkazy

Podobné instrukce: HALT.

XFC

Jméno

XFC — Extended Function Call

Přehled

XFC

XFC *src, dst*

Poznámky k tvaru instrukce

N	Z	V	C
0	0	0	0

Popis

opcode	mnemo	name
FC	XFC	Extended Function Call

Odkazy

Podobné instrukce: HALT.

Kapitola 65. IBM System 360

Vlastnosti:

- IBM 360 Architecture and Microprogramming¹ by Ivor Jones
-
-
-

Architektura IBM 360 je velmi stará, ale současně přetrvala až do dnešních dob. Proto jsem ji nezařadil přímo mezi historické počítače ale věnuji jí samostatnou kapitolu.

65.1. Formáty instrukcí

Odkazy:

- Operační systémy, Stuart E. Madnick, John J, Donovan, SNTL 198, strana 549
-
-

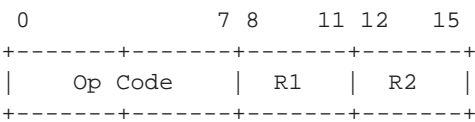
System 360 používá několik formátů instrukcí. Tyto formáty používají délku 16, 32 a 48 bitů. Tedy jedno, dvě a tři půlslova. Operační kód začíná v prvních osmi bitech. A nejvyšší dva z nich současně určují jak je instrukce dlouhá.

Tabulka 65-1. Délky a formáty instrukcí architektury System 360

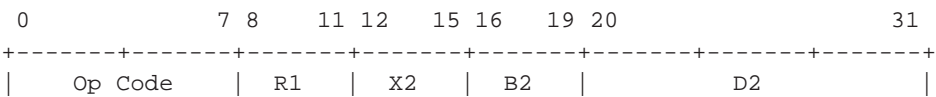
bity 0-1	délka in- strukce	formáty
00	1	I/E/ RR
01	2	RX
10	2	RI/RRE/RRF/RRR/ RS /RSI/RX/ SI /S
11	3	RIE/RIL/RSL/RSY/RXE/RXF/RXY/SIY/ SS /SSE/SSF

Poznámka: V tabulce jsou i formáty instrukcí z následníků System 360.

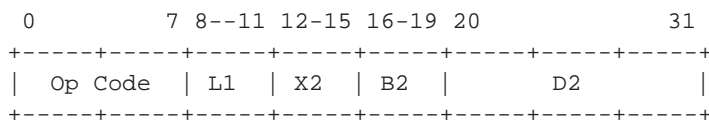
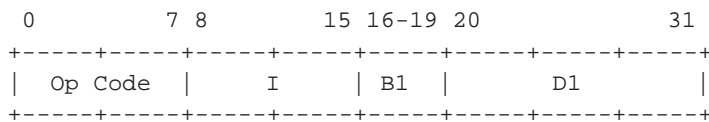
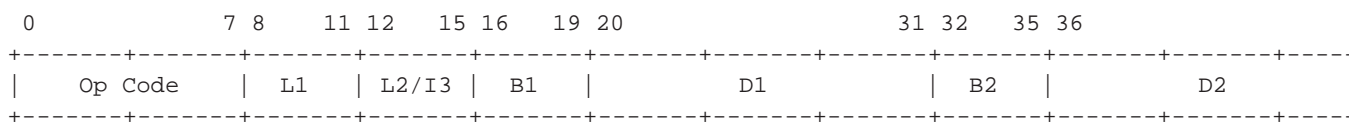
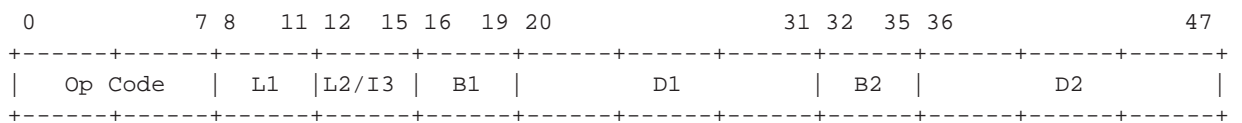
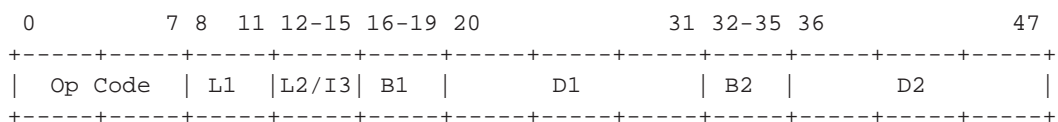
Obrázek 65-1. RR



Obrázek 65-2. RX



+-----+-----+-----+-----+-----+-----+-----+-----+

Obrázek 65-3. RS**Obrázek 65-4. SI****Obrázek 65-5. SS****Obrázek 65-6. SS****Obrázek 65-7. SS****Tabulka 65-2. Obsazení paměti IBM System 360**

Adresa DEC	Délka	Význam
0	double word	Initial program Loading PSW
8	double word	Initial program Loading CCW1
16	double word	Initial program Loading CCW2
24	double word	External old PSW
32	double word	Supervisor call old PSW
40	double word	Program old PSW
48	double word	Machine check old PSW
56	double word	Input/output old PSW
64	double word	Channel status word

Adresa DEC	Délka	Význam
72	word	Channel address word
76	word	Unused
80	word	Timer
84	word	Unused
88	double word	External new PSW
96	double word	Supervisor call new PSW
104	double word	Program new PSW
112	double word	Machine check new PSW
120	double word	Input/output new PSW
128	depends on model	Diagnostic scan-out area

65.2. Instrukce

Tabulka 65-3. Seznam instrukcí System 360 podle operačních kódů

Name	Mnemonic	Opcode	Format	Operands
Supervisor Call	SVC	0A	RR	R ₁ ,R ₂
Add	AR	1A	RR	R ₁ ,R ₂
Subtract	SR	1B	RR	R ₁ ,R ₂
Add Logical	ALR	1E	RR	R ₁ ,R ₂
Subtract Logical	SLR	1F	RR	R ₁ ,R ₂
Load Positive	LPR	10	RR	R ₁ ,R ₂
Load Negative	LNR	11	RR	R ₁ ,R ₂
Load and Test	LTR	12	RR	R ₁ ,R ₂
Load Complement	LCR	13	RR	R ₁ ,R ₂
Load	LR	18	RR	R ₁ ,R ₂
Compare	CR	19	RR	R ₁ ,R ₂
Add Halfword	AH	4A	RX	R ₁ ,D ₂ (X ₂ ,B ₂)
Subtract Halfword	SH	4B	RX	R ₁ ,D ₂ (X ₂ ,B ₂)
Load Halfword	LH	48	RX	R ₁ ,D ₂ (X ₂ ,B ₂)
Compare Halfword	CH	49	RX	R ₁ ,D ₂ (X ₂ ,B ₂)
Add	A	5A	RX	R ₁ ,D ₂ (X ₂ ,B ₂)
Subtract	S	5B	RX	R ₁ ,D ₂ (X ₂ ,B ₂)
Add Logical	AL	5E	RX	R ₁ ,D ₂ (X ₂ ,B ₂)
Subtract Logical	SL	5F	RX	R ₁ ,D ₂ (X ₂ ,B ₂)
Load	L	58	RX	R ₁ ,D ₂ (X ₂ ,B ₂)
Compare	C	59	RX	R ₁ ,D ₂ (X ₂ ,B ₂)
Load Multiple	LM	98	RS	R ₁ ,R ₃ ,D ₂ (B ₂)

A

Jméno

A, AR, AH, ALR, AL — Add, Add Halfword, Add Logical

Přehled

AR $R1, R2$

A $R1, D2(X2, B2)$

AH $R1, D2(X2, B2)$

ALR $R1, R2$

AL $R1, D2(X2, B2)$

Popis

Název	Mnemo	Kód	Formát	Operandy
Add	A	5A	RX	
Add	AR	1A	RR	
Add Halfword	AH	4A	RX	
Add Logical	AL	1E	RX	
Add Logical	ALR	5E	RR	

Odkazy

Podobné instrukce: S.

C

Jméno

C, CR — Compare

Přehled

CR $R1, R2$

C $R1, D2(X2, B2)$

Popis

Název	Mnemo	Kód	Formát	Operandy
Subtract	C	59	RX	
Subtract	CR	19	RR	

Odkazy

Podobné instrukce: S, CH.

CH

Jméno

CH — Compare Halfword

Přehled

CH $R1, D2(X2, B2)$

Popis

Název	Mnemo	Kód	Formát	Operandy
Subtract Halfword	CH	49	RX	

Odkazy

Podobné instrukce: S, C.

L

Jméno

L, LR — Load

Přehled

LR R_1, R_2

L $R_1, D_2(X_2, B_2)$

Popis

LR	18
L	58

Odkazy

Podobné instrukce: A, LH, LTR, LCR, LPR, LNR, LM.

LCR

Jméno

LCR — Load Complement

Přehled

LCR R_1, R_2

Popis

LCR	13
-----	----

Odkazy

Podobné instrukce: L, LR, LH, LTR.

LH

Jméno

LH — Load Halfword

Přehled

$\mathbf{LH} \ R_1, \ D_2(X_2, B_2)$

Popis

LH	48
----	----

Odkazy

Podobné instrukce: L, LR.

LM

Jméno

LM — Load Multiple

Přehled

$\mathbf{LM} \ R_1, \ R_3, \ D_2(B_2)$

Popis

Skupina registrů, počínaje registrem R_1 a konče registrem R_3 je naplněna obsahem paměti začínajícím na adrese danou $D_2(B_2)$.

Registry jsou plněny postupně od registru specifikovaného R_1 až k registru specifikovanému R_3 . Pokud je parametr R_1 stejný jako R_3 , naplní se jen jeden registr určený tímto číslem. Pokud je hodnota R_3 menší než hodnota R_1 , naplní se postupně registry od GPR_{R_1} až do GPR_{15} a poté se pokračuje registry GPR_0 až GPR_{R_3} .

$$-R_2 \longrightarrow R_1$$

LM	98
----	----

Odkazy

Podobné instrukce: L, LR, LH, LTR.

LNR

Jméno

LNR — Load Negative

Přehled

$$\text{LNR } R_1, R_2$$

Popis

Do R_1 je uložena hodnota $-R_2$.

$$-R_2 \longrightarrow R_1$$

LNR	11
-----	----

Odkazy

Podobné instrukce: L, LR, LH, LTR.

LPR

Jméno

LPR — Load Positive

Přehled

LPR R_1, R_2

Popis

Do R_1 je uložena absolutní hodnota R_2 .

$\text{abs}(R_2) \rightarrow R_1$

LPR	10
-----	----

Odkazy

Podobné instrukce: L, LR, LH, LTR.

LTR

Jméno

LTR — Load and Test

Přehled

LTR R_1, R_2

Popis

LTR	12
-----	----

Odkazy

Podobné instrukce: L, LR.

S

Jméno

S, SR, SH, SLR, SL — Subtract, Subtract Halfword, Subtract Logical

Přehled

SR $R1, R2$

S $R1, D2(X2, B2)$

SH $R1, D2(X2, B2)$

SLR $R1, R2$

SL $R1, D2(X2, B2)$

Popis

Název	Mnemo	Kód	Formát	Operandy
Subtract	S	5B	RX	
Subtract	SR	1B	RR	
Subtract Halfword	SH	4B	RX	
Subtract Logical	SL	1F	RX	
Subtract Logical	SLR	5F	RR	

Odkazy

Podobné instrukce: A, C.

SVC

Jméno

SVC — Supervisor Call

Přehled

SVC *I*

Popis

SVC	0A
-----	----

Odkazy

Podobné instrukce: **FIXME**:s360.isa.svc;.

65.3. Nástroje pro tvorbu programů

*

65.3.1. Assembler

*

Odkazy:

- z390 Portable Mainframe Assembler and Emulator Project² — v Javě
-
-
-
-
-

Poznámky

1. <http://www.cs.manchester.ac.uk/CCS/res/res23.htm#d>
2. <http://www.z390.org/>

XI. Robotika a mechatronika

Tato část knihy pojednává o robotice. Oboru který je kombinací elektroniky, mechaniky, číslicové elektroniky, počítačů, programování, umělé inteligence a řadě dalších.

Odkazy:

- robotika.cz¹⁰²¹
- [Hobbyrobot](http://www.hobbyrobot.cz/)¹⁰²²
- [Robot revue](http://www.robotrevue.cz/)¹⁰²³

Obchody:

- [Stavebnice.com](http://www.stavebnice.com)¹⁰²⁴

Robotika a mechatronika nebyla nikdy tak snadná a dostupná jako je dnes, a budoucnost vypadá ještě lépe. Jen se podívejte kolem sebe, kolik zařízení je vám k dispozici. Podívejte se na dnešní nabídku elektronických zařízení a kitů. Můžete konstruovat věci dříve pro běžného člověka nedosažitelné. Stačí vám k tomu jednoduchý řídicí element jako je například 40.8, pár periférií, motory a něco mechaniky a můžete si postavit 3D tiskárnu, obráběcí stroj, malého robota, Komponenty jako GPS přijímače, a elektronické gyroskopy a snímače zrychlení (gravitačního pole), snímače zemského elektromagnetického pole (kompasy). Velmi výkonné počítače v jednom čipu za pár korun (či euro). To jsou všechno věci o kterých jste před lety nemohli ani snít a teď jsou vám k dispozici a můžete si je koupit.

Poznámky

1021. <http://robotika.cz/>

1022. <http://www.hobbyrobot.cz/>

1023. <http://www.robotrevue.cz/>

1024. <http://www.stavebnice.com>

Kapitola 66. Roboti

* *Attributy: id="Robots"*

Informace o robotech na internetu, robotických stavebnicích i celých kompletech.

66.1. Humanoidní roboti

Odkazy:

- Robonova-I
 - Robonova-I Humanoid Robot Kit¹ on Robosavvy
- Bioloid
 - Educational Robot kit "Bioloid"²
 - Bioloid³
- RoboBuilder 5710K
 - RoboBuilder 5710K Intro⁴ on Robosavvy
- KHR-1
 - KHR-1 Kit Content⁵ on Robosavvy
- Kondo KHR-2HV
 - KONDO KHR-2HV Humanoid Robot Kit⁶ on Robosavvy
- -

66.2. Vozítka

Odkazy:

- POBbot
 - POB Educational Platform⁷
- Pololu 3pi Robot
 - Pololu 3pi Robot Kit⁸

66.3. Létadla

*

Odkazy:

-
-

66.3.1. Čtyřrotorový *Quadcopter*

* *Attributy: id="Quadcopter"*

Odkazy:

- Next Generation UAVP⁹
- Quadcopter Home¹⁰
- AeroQuad¹¹ The Open Source Quadcopter
-
-

Poznámky

1. http://robosavvy.com/site/index.php?option=com_content&task=view&id=135&Itemid=124
2. http://www.robotis.com/zbxe/bioloid_en
3. http://robosavvy.com/site/index.php?option=com_content&task=view&id=82&Itemid=81
4. http://robosavvy.com/site/index.php?option=com_content&task=view&id=159&Itemid=128
5. http://robosavvy.com/site/index.php?option=com_content&task=view&id=57&Itemid=43
6. http://robosavvy.com/site/index.php?option=com_content&task=view&id=127&Itemid=119
7. http://robosavvy.com/site/index.php?option=com_content&task=view&id=180&Itemid=133
8. <http://www.pololu.com/catalog/product/975>
9. <http://ng.uavp.ch/moin>
10. <http://www.quadcopter.org>
11. <http://aeroquad.com>

Kapitola 67. Obráběcí a tvořící stroje

Tato kapitola je věnována strojům a konstrukcím jenž mají praktický význam v tom že umožňují obrábět, či jiným způsobem zpracovávat hmotu.

Odkazy:

- Thingiverse¹ — blog publikovaných modelů pro různé typy obrábění a tisku
- CNC Router Zone — Great Info on CNC Routers²
-
-

67.1. RepRap a jiné 3d tiskárny

Odkazy:

- Fab@Home³
- RepRap Project⁴
-
-

Nástroje:

- Skeinforge⁵
-

67.2. Vrtací a frézovací stroje

Odkazy:

- MTM A-Z PCB Mill⁶ rovněž dřevěná konstrukce
- How to Make a Three Axis CNC Machine (Cheaply and Easily)⁷
- Fireball V90 CNC Router Assembly⁸ — kit za cca \$599
- ! Home-Made DIY CNC Machine using Step Motors, Dremel, and EMC2⁹ — [IL]
- CNC Cookbook¹⁰
- www.C-N-C.cz fórum¹¹
- Varsanyi Peter¹² — jen obrázky
-
-
-

Obchody:

- CNCshop.cz¹³ — CNC & automatizace
-
-
-
-

Odkazy na české konstrukce:

-
-

-

Instructables:

- How to make a mini milling machine- manual or CNC!¹⁴
- Build a CNC Router from Scratch (Part 1): Complete Video Tutorial¹⁵
- \$20 CNC Machine¹⁶
-

Vybraná videa na YouTube:

- Easy to build Desktop CNC Mill¹⁷ — konstrukce na Instructables.com¹⁸
-
-

67.2.1. Dřevěný 3-osý obráběcí stroj do \$100

*

Odkazy:

- 3D Obráběcí fréza, dřevěná konstrukce.¹⁹
-

Stroj má velmi primitivní elektroniku sestávající s tranzistorových spínačů na paralelním portu které přímo spínají vinutí tří krokových motorů.

Jako řídicí program slouží EMC2.

EMC2 odkazy:

- EMC Info²⁰
- emc2.git summary²¹
-
-

67.3. Plotry

Odkazy:

- Zapisovač Alfí²²
- Návod sestavení Alfí²³
- zapojení plotru alfi k pc²⁴
-
-

Poznámky

1. <http://www.thingiverse.com/>
2. <http://cncrouterzone.org/>
3. http://fabathome.org/wiki/index.php?title=Main_Page
4. http://en.wikipedia.org/wiki/RepRap_Project
5. <http://fabmethesus.crsndoo.com/wiki/index.php/Skeinforge>

6. http://mtm.cba.mit.edu/machines/mtm_az/index.html
7. <http://www.instructables.com/id/How-to-Make-a-Three-Axis-CNC-Machine-Cheaply-and-/>
8. <http://hackedgadgets.com/2009/04/26/fireball-v90-cnc-router-assembly/>
9. <http://www.lirtex.com/robotics/diy-cnc-machine>
10. <http://www.cnccookbook.com/CCResourcesInd.htm>
11. <http://www.c-n-c.cz/index.php>
12. http://www.ftp.cnhungary.com/Varsanyi_Peter/CNC%20gepek%20fotok/
13. <http://www.cncshop.cz/>
14. <http://www.instructables.com/id/Make-a-mini-milling-machine/>
15. <http://www.instructables.com/id/Build-a-CNC-Router-from-Scratch-Part-1:-Complete/>
16. <http://www.instructables.com/id/20-CNC-Machine/>
17. <http://www.youtube.com/watch?v=6drMZqmyXQc>
18. <http://www.instructables.com/id/Easy-to-Build-Desk-Top-3-Axis-CNC-Milling-Machine/>
19. <http://fab.cba.mit.edu/classes/MIT/863.09/people/dcarr/final/final.html>
20. <http://wiki.linuxcnc.org/cgi-bin/emcinfo.pl?git>
21. <http://git.linuxcnc.org/gitweb?p=emc2.git;a=summary>
22. <http://www.merkurtoys.cz/cz/robotika/stranky/alfi/stranka.htm>
23. <http://www.cerovsky.net/merkur/>
24. <http://merkur.kreteni.cz/diskuse/prispevek.php?od=0&kat=18>

Kapitola 68. Díly a komponenty

*

Obchody a dodavatelé:

- MICROCON¹
- Ing. Petr Spáčil - kovoobrábění²
- Torqspline Screws³
- cnc-multitool.com⁴
- MakerBot STORE⁵
- DUZI CNC Shop⁶ — I/O karty [CZ]
- CNC Shop⁷
- 68.3
-
-

Odkazy:

- High Speed 4 axis opto isolated interface PCB⁸
- PMinMO.com⁹
- T-Slot system¹⁰
-
-

68.1. Krokové motory a jejich řízení

*

Odkazy:

- MS MOTOR CO., LTD¹¹ (China)
- Krokové motory¹² na Alltronics.com
- Stepper Motor NEMA Sizes¹³
- stepper motor - use it cheapest and easiest¹⁴ — jak rozpoznat zapojení neznámého krokového motoru
- Tutoriál Control of Stepping Motors¹⁵ Douglas W. Jones
- Other Stepper Motors¹⁶

Některé krokové motory mají standardizované rozměry a uchycení které se nazývá NEMA.

Tabulka 68-1. NEMA

NEMA	vnější rozměr	uchycení	díra
8		16±0.2mm	
11		23.11mm	
14	35.2×35.2	26±0.2mm (25.91mm)	0.157
16			
17		1.22"×1.22" (30.99mm)	
23	2.3" × 2.3"	1.8560"×1.8560 (47.34mm)	DIA 0.2050"
24			
34		2.740×2.740 (69.6mm)	

NEMA	vnější rozměr	uchycení	díra
42			

Bipolární motor

Unipolární motor

68.1.1. Řízení krokových motorů

*

Odkazy:

- EasyDriver Stepper Motor Driver¹⁷
-

V <http://www.schmalzhaus.com> jsou kompletní podklady pro výrobu ovladačů dvoufázových krokových motorů EasyDriver. Tyto jsou založeny na čipu A3967SLB. Tyto desky jsou použity například v DIY CNC¹⁸ publikovaném na Instructables.

Brian Schmalz¹⁹ vytvořil konstrukci s A3967SLB proto, že se cena původně používaného čipu UCN5804B zvedla. Aktuálně [2011-04-17, GME] je nabízen za 696Kč s DPH.

Odkazy:

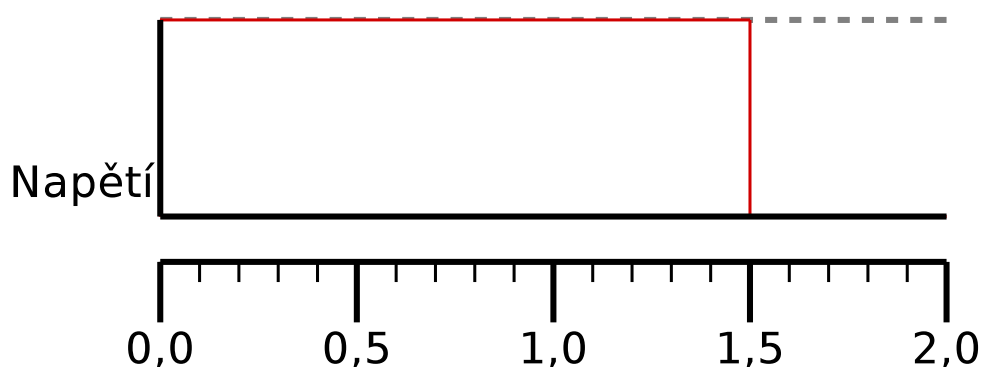
- EasyDriver v3.1 Tutorial²⁰
-

68.2. Servomotory

Odkazy:

- 57.1
- Whats a Servo?²¹
- RC servo controlling²²
- Generating PWM signals using Timers in the ATmega chip²³
-
-

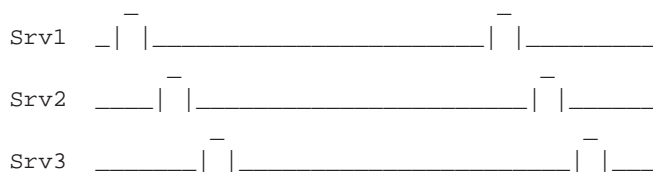
Modelářské servomotory jsou výkonné pohybové mechanismy řízené nejčastěji pulsním signálem. Puls vysílaný 50 krát v sekundě definuje svojí šířkou úhel natočení servopohonu. Minimální délka pulzu je cca 1ms a takový puls nastavuje mechanismus úplně v levo. Délka pulsu 1.5ms nastaví střední, neutrální, polohu. Maximální délka pulsu 2ms nastaví mechanismus do pravé krajní polohy.

Obrázek 68-1. Ovládací puls pro nastavení neutrální polohy na servomotoru

Pokud řídíme serva MCU AVR, můžeme je řídit přímo výstupy OC1A, OC1B. Ovládací program je pak velmi jednoduchý, protože o všechno se starají obvody prvního čítače/časovače. Jsme ale omezeni jen na dvě serva.

Pokud použijeme sofistikovanější softwarový přístup, můžeme ovládat až 20 servomotorů. Základem tohoto přístupu je, pulzy do serv mohou být navzájem posunuty.

Celý cyklus při 50Hz trvá 20ms. Impuls pro každé servo trvá od 1ms do 2ms. Tedy je zde pevná fixní část dlouhá 1ms. Algoritmus je pak následující. Každou ms budeme generovat startovací část impulsu serva číslo $n+1$ a současně nastavíme čítač časovač a budeme generovat proměnlivou část impulsu serva číslo n . Toto překrývání ná umožní ovládat s pomocí jednoho časovače až 20 serv.

Obrázek 68-2.

Pokud bychom mohli využít druhý registr OC1B u prvního časovače, mohli bychom ovládat dvě serva paralelně. To by zvedlo maximální počet ovládaných serv na 40.

Bežné modelářské servomotory mají jednoduché třívodičové připojení přivedené na konektor. Barvy vodičů a jejich význam je následující:

Tabulka 68-2. Připojení modelářského serva

vodič	barva
pulzní řízení	bílá, žlutá, nebo obecně svatlá
Ucc	červená
GND	černá, hnědá, nebo obecně tmavá

68.2.1. Parametry a dostupnost některých servomotorů

Tabulka 68-3. Turnigy TG9

--	--

hmotnost	9g
napětí	3 až 6 V stejnosměrné
rychlost	0.12s/60 stupňů při 4.8V
kroutící moment	1.6kg - cm
pracovní teplota	-30 až 60 stupňů Celsia

Odkazy:

- Micro servo — Turnigy TG9²⁴ na Adafruit Industries
- \$1.99 Servo Dissected²⁵
-

Tabulka 68-4. HEXTRONIK HTX 900

hmotnost	9g
velikost	21x12x22 mm
napětí	3 až 6 V
rychlost	0.12 sec/60 při 4.8V
kroutící moment	1.6 kg - cm
pracovní teplota	-30 až 60 stupňů

Odkazy:

- Hextronik HXT900 Micro Servo 9g / 1.6kg / .12sec²⁶
-

68.3. Contraptor

*

Odkazy:

- Contraptor²⁷ open source DIY rapid hardware prototyping
- Contraptor²⁸ na garageFab.cc
-
- SourceForge²⁹
-
-

Tato "stavebnice" je takovým větším Merkurem. Dírky jsou v rozteči jednoho palce.

Projekty:

- Vincent Rouilly/XYZ Table Project³⁰
-

Podobné "stavebnice":

•

Gríde Beam

- Grid Beamers³¹ [2007]
- Grid Beam³²

- Gridbeam³³
-
-

68.4. Konstrukční materiály

*

Odkazy:

- AluCad - hliníkové profily³⁴
- BETZ TECHNIK³⁵
- MAREK INDUSTRIAL a.s.³⁶
- Poltech³⁷
- ALUTEC K&K a.s.³⁸
-
-
-

Poznámky

1. <http://microcon.cz/>
2. <http://www.krutor.cz/manufacturing/aboutme/cz.htm>
3. <http://roton.com/Screw.aspx?line=Torqspline>
4. <http://www.cnc-multitool.com/shop-milling.asp>
5. <http://store.makerbot.com/>
6. http://www.duzi.cz/shop_cnc/
7. <http://www.cncshop.cz/>
8. <http://pminmo.com/4axisopto/4axisDIYopto.htm>
9. <http://pminmo.com/>
10. <http://www.8020.net/T-Slot-5.asp>
11. <http://www.ms-motor.com/>
12. <http://www.alltronics.com/cgi-bin/category/55>
13. <http://www.piclist.com/techref/io/stepper/nemasizes.htm>
14. <http://www.instructables.com/id/Unknown-stepper-motor-wiring/>
15. <http://www.cs.uiowa.edu/~jones/step/index.html>
16. <http://www.homes.doc.ic.ac.uk/~ih/doc/stepper/others/>
17. <http://www.schmalzhaus.com/EasyDriver/>
18. <http://www.instructables.com/id/DIY-CNC-1-2/>
19. <http://www.schmalzhaus.com/>
20. <http://danthompsonsblog.blogspot.com/2008/09/easydriver-v31-tutorial.html>
21. <http://www.seattlerobotics.org/guide/servos.html>

22. <http://www.epanorama.net/documents/motor/rcservos.html>
23. <http://mil.ufl.edu/~achamber/servoPWMfaq.html>
24. http://www.adafruit.com/index.php?main_page=product_info&cPath=34&products_id=169
25. <http://www.rcgroups.com/forums/showthread.php?t=1173308>
26. http://www.hobbytec.co.uk/product.php?id_product=16
27. <http://www.conraptor.org/>
28. <http://www.garagefab.cc/conraptor>
29. <http://sourceforge.net/projects/conraptor/>
30. http://openwetware.org/wiki/User:Vincent_Rouilly/XYZ_Table_Project
31. <http://www.gridbeamers.com/>
32. <http://gridbeam.biz/>
33. <http://www.gridbeamnation.com/>
34. <http://www.alucad.cz/standardni-hlinikove-profil-y-pro-siroke-pouziti/>
35. http://www.betz.cz/download/soubory_ke_stazeni.html
36. <http://www.marek.eu/cz/produkty/hlinikove-profil-y-konstrukce-doplňky-b-fm-systeme-b-fath/>
37. <http://www.poltech.cz/poltech/6-Hlinikove-Profil-y-Schnaithmann>
38. <http://www.aluteckk.cz/>

Kapitola 69. Poznámky

Co se týče rozhodnutí jestli hlavu s nástrojem umístit na most který se hýbe nad pracovním prostorem, tak takové řešení zabírá nejméně místa.

Druhé řešení, kdy pracovní stůl hýbe s obráběným předmětem ve dvou osách ve vodorovné rovině je klasičtější. Nejspíše má své výhody, mě napadají dvě.

První výhoda je ta, že se s obráběcí hlavicí hýbe jen v ose Z a tato může mít větší hmotnost a pevnější uchycení. Její hmotnost nezatěžuje most protože žádný nemáme.

Druhá výhoda která mne napadla je že mohu mít více obráběcích či měřících hlavic, které jsou na fixních pozicích a samy se pohybují jen v ose Z. Tímto způsobem je možno realizovat stroj s několika odlišnými nástrojovými hlavicemi které se nemusí vyměňovat, protože stůl s obráběným předmětem si najede pod právě potřebnou hlavici.

Kapitola 70. Mechanika

70.1. Různé poznámky

*

Odkazy:

- Open Source Engineering¹
-

Na Z-39 was "Yet Another Tracked Bot"² je detailní popis konstrukce pásového tankového mechanismu. Konstrukce je formou CNC frézovaných či řezaných komponent ze dřeva či podobného materiálu. Dílky se poskládají, některé části slepi a jednotlivé segmenty pásu se spojí čepy či v tomto případě šrouby.

Poznámky

1. http://p2pfoundation.net/Open_Source_Engineering
2. <http://letsmakerobots.com/node/25507>

XII. Programové vybavení a operační systémy

Kapitola 71. Softwarové nástroje

* *Attributy: id="software"*

V této kapitule popisují nebo zmiňují různé programy. Jedná se o programy pro práci se schématy, analýzu obvodů, simulaci či jiné.

Pro simulaci obvodů můžeme použít:

- Logisim¹ — simulátor digitálních obvodů. Tento program je napsaný v Javě, a proto jej můžeme použít i na MacOS X a MS Windows. Použitá licence je GPL 2.
- KLogic² []
- Logic Simulator³ [Java]
- A Digital Circuit Simulator/Designer program⁴
- KSimus⁵
- Qucs⁶
-

Logisim je simulátor digitálních obvodů naprogramovaný v Javě. Můžeme jej tedy provozovat jak na Linuxu, tak na Mac OSX a i na MS Windows. Program umožňuje vytvářet vlastní komponenty a ty používat při sestavování dalších komponent. Obvody jsou editovány za živa, dusledky každé změny v zapojení ihned vidíme. Propojovací vodiče svou barvou říkají jaká logická hodnota na nich je, není tedy třeba mít všude sondy. Největší nevýhodou kterou u programu spatřuji je nemožnost vyexportovat zapojení v EPS.

KLogic je další simulátor digitálních obvodů, tentokrát naprogramovaný pod KDE. Tento program rovněž nemá možnost exportovat vytvořený obvod do EPS.

71.1. gEDA

Odkazy:

- gEDA^{project7}
- gEDA⁸ na Wikipedii
- gEDA Project Wiki⁹

Protože s interní databází komponent instalovanou přímo s programem nemusíme vystačit, použijeme komponenty publikované na internetu. Tato je na webu gedasymbols.org¹⁰.

Videa na YouTube:

- Dvě videa uživatele alpacadee¹¹ ukazující práci v gschem a pcb [bez zvuku, japonsky]
- Kanál uživatele sonodrome¹², který publikoval asi půl tuctu videí
- YouTube kanál uživatele XbitElectronics¹³. Mají také web¹⁴. Uživatel slíbil publikovat nějaké videotutoriály.
- Kanál uživatele Some professors¹⁵
-
-

71.1.1. Instalace

71.1.1.1. Instalace verze 1.5 ze zdrojů

Nejdříve jsem si vytvořil adresář.

```
# mkdir -p /usr/local/src/geda
# chown radek:radek /usr/local/src/geda
```

Potom stáhl příslušné zdrojové soubory.

```
$ mkdir /usr/local/src/geda/download
$ cd /usr/local/src/geda/download
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/README
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/gaf-1.5.0-relnotes.html
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/gEDA-gaf-1.5.0.md5sum
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/Makefile
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/libgeda-1.5.0.tar.gz
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/geda-symbols-1.5.0.tar.gz
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/geda-gschem-1.5.0.tar.gz
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/geda-genlist-1.5.0.tar.gz
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/geda-gsymcheck-1.5.0.tar.gz
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/geda-gattrib-1.5.0.tar.gz
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/geda-utils-1.5.0.tar.gz
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/geda-docs-1.5.0.tar.gz
$ wget http://www.geda.seul.org/devel/v1.5/1.5.0/geda-examples-1.5.0.tar.gz
```

Balíčky rozbalíme

```
$ cd /usr/local/src/geda
$ for package in download/*.tar.gz; do tar xzvf $package; done
```

A všechno přeložíme

```
# make configure
# export PREFIX=/usr/local
# export LD_LIBRARY_PATH=$PREFIX/lib:$LD_LIBRARY_PATH
# export PATH=$PREFIX/bin:${PATH}
# export PKG_CONFIG_PATH=$PREFIX/lib/pkgconfig:$PKG_CONFIG_PATH
# make install >log 2>&1
```

71.1.1.2. Instalace verze 1.6 ze zdrojů na Debian Lenny

Pro překlad je třeba mít nainstalovanou verzi debhelper z Squeeze. Debhelper nemá další závislosti, takže to šlo bez problémů. Poté doinstalovat požadované vývojové balíčky.

```
$ apt-get -b source geda-gaf=1:1.6.0-4
```

* Tak jsem se zkusil podívat, není li k dispozici novější verze a zjisťil jsem že maličko novější, tedy 1.6.1 je.

```
$ apt-get -b source geda-gaf=1:1.6.1-4
```

71.1.1.3. Kopilace pcb z gitu

*

```
$ git clone git://git.gpleda.org/pcb.git
$ ./autogen.sh
$ ./configure --with-gui=gtk --prefix=$HOME
$ make
$ make install
```

71.1.2. Konfigurace

*

Po nainstalování se v našem domovském adresáři vytvoří podadresář `~/.gEDA` který obsahuje celou uživatelskou konfiguraci programu. Systémová konfigurace se nachází v adresáři `/etc/gEDA`. Tím „hlavním“ konfiguračním souborem, který si budeme upravovat, je `~/.gEDA/gschemrc`. Tento soubor je v lispovském (scheme) formátu.

71.1.2.1. Změna standardního černého pozadí na světlé

*

Protože mi lépe vyhovuje světlé pozadí, chtěl jsem si nakonfigurovat **gschem** tak aby ho používal standardně. Jedná se o volbu v menu `View` → `Light color scheme (v l)`.

Tohoto nastavení dosáhneme vložením následujícího kódu do `~/.gEDA/gschemrc`.

```
; Load up a color scheme
(load (build-path geda-rc-path "gschem-colormap-lightbg"))
```

71.1.3. Jak malovat schémata

*

No jednoduše. Ale teď vážněji. Je dobré si projít některý nebo více tutoriálů o práci s gEDA. Zde totiž popisují jen některé zvláštnosti.

71.1.3.1. Atributy

*

Každá komponenta má své atributy. Zde skusím některé atributy popsat.

Popisy některých atributů

`pintype`

pinlabel

Každý vývod (pin) součástky musí mít vyplněn tento atribut a jeho hodnota musí být jednoznačná. Každý pin součástku musí mít jinou hodnotu.

pinnumber

Číslo vývodu (pinu) součástky. Toto číslo odpovídá číslu vývodu na footprintu.

pinseq

Hodnota podobná pinnumber. Je důležitá pro PCB a Autorouter.

refdes

Identifikátor komponenty. Pokud komponentu vytváříme to co zde zapíšem je šablona identifikátoru. Například pro diodu uvedem "D?" kde znak "?" je pak nahrazen číslem.

device

Aktuální jméno součástky / integrovaného obvodu jako například "LM555".

footprint

71.1.3.2. Součástky a jejich footprinty

*

71.1.3.2.1. Jednoduché pájecí body a konektory

*

Na schématu používám s oblibou komponentu "terminal". Na PCB ji mohu realizovat

* `/user/kai_martin_knaak/symbols/connector/con_1x1.sym`

Odkazy:

- `1x1PIN.fp`¹⁶
-

71.1.4. Vytváření PCB

*

Odkazy:

- `gschem -> gsch2pcb -> PCB`¹⁷
-

Budeme potřebovat program **gsch2pcb** který je v standardní instalaci programů gEDA. Tento program vzme schéma a vloží součástky do předlohy plošného spoje. Tak jednoduché to není ale tím začínáme.

Proto aby vše fungovalo, musíme mít vyplněny u každé komponenty následující atributy:

- footprint — rozložení vývodů součástky na plošném spoji

- value — hodnota součástky (u odpor například 1k, u tranzistoru BC546, ...)
- refdes — identifikátor součástky (R2, U6, C5, ...)

Právě hodnota atributu footprint je pro mne záhadou kterou je třeba vyřešit. Začnu od jednodušších komponent.

Footprinty pro odpory jsou například: R025.

71.1.5. Knihovna symbolů

*

Knihovna symbolů instalovaná s editorem schémat se nachází v adresáři `/usr/share/gEDA/sym/`.

Centrální web pro symboly je [gedasymbols](http://gedasymbols.org)¹⁸.

Můžeme mít libovolný počet lokálních knihoven. Tyto musíme ale uvést v konfiguračním souboru `~/gEDA/gafrc`. Adresáře se symboly zavádíme příkazem `component-library`.

```
;; User symbol libraries
(define personal-library "/home/radek/lib/gEDA")
(component-library (build-path personal-library "sym/headers") "Headers (local)")
(component-library (build-path personal-library "sym/natsemi") "Nat.Semi. (local)")
```

71.1.6. Footprints

* *Attributy: id="gEDA.Footprints"*

Odkazy:

- Footprint creation for the open-source layout program "PCB"¹⁹ [2005]
-

Footprint je rozvržení kontaktů/děr na plošném spoji pro konkrétní součástku. Je to tedy tvar měděné vrstvy pro osazení konkrétní součástkou.

71.1.7. Otázky a odpovědi

Q: hello. I have problem with "Embed Component/Picture". It looks promising for what I need but it isn't. What I need is to embed component into schematic and then "Down Symbol" modify it. But it always try to edit the original symbol and not the embeded.

A: i think you need to save symbol in example in project dir, then modify gafrc to see such symbols, then modify it then embed

71.2. TkGate

Odkazy:

- TkGate.org²⁰
-

TkGate je program pro vytváření a simulaci digitálních obvodů. Základním stavebním kamenem jsou hradla, ale máme k dispozici i MOS tranzistory jenž jsou považovány za elektronicky ovládané spínače. Tedy žádná anal-

ogová techniky. Mimo jednoduchá hradla máme k dispozici a "větší" komponenty jako jsou registry multiplexery a demultiplexery, paměti, . . .

Poslední verze o které vím a kterou jsem zkoušel je 2.0b10.

Poznámka: Některá schémata v tomto dokumentu byla vytvářena v TkGate.

71.3. EMC2

* *Attributy: id="EMC2"*

Odkazy:

- LinuxCNC²¹
-

Enhanced Machine Controller project.

71.3.1. Instalace

*

Odkazy:

- Debian Lenny Compile EMC2²²
-

Na Debian Lenny postupujeme podle návodu Debian Lenny Compile EMC2²³.

```
$ ./configure --enable-run-in-place --enable-simulator \
  --with-tclConfig=usr/lib/tcl8.4/tclConfig.sh \
  --with-tkConfig=/usr/lib/tk8.4/tkConfig.sh
```

71.4. SketchUp

*

Odkazy:

- SketchUp²⁴ v knize o Google
- How to run Google Sketchup 7 on eeePC 901 with a Debian Lenny OS ?²⁵ na Zeitoun.NET
- Debian Bug report logs - #554804 wine-unstable: Google sketchup 7.1 fails to install²⁶
-
-
-
-

71.4.1. Skriptování v Ruby

*

Odkazy:

- suplugins (WebConsole)²⁷
-

71.5. Art Of Illusion

*

Odkazy:

- Art of Allusion²⁸
- vasek.wu.cz²⁹
-
-
-

71.6. SIMH (simulátor starých počítačů)

*

Odkazy:

- The Computer History Simulation Project³⁰
-

Poznámky k verzím programu

3.8-1

Poslední verze vydaná 2009-02-08.

3.7-0

Verze v Debian 5.0 Lenny.

Přehled strojů které SIMH simuluje:

- Data General Nova, Eclipse
- Digital Equipment Corporation: PDP-1, PDP-4, PDP-7, PDP-8, PDP-9, PDP-10, PDP-11, PDP-15, VAX
- GRI Corporation GRI-909, GRI-99
- IBM 1401, 1602, 1130, 7090/7094, System 3
-
- Hewlett-Packard 2114, 2115, 2116, 2100, 21MX, 1000
-
-

Program SIMH který napsal Bob Supnik

Poznámky

1. <http://ozark.hendrix.edu/~burch/logisim/>
2. <http://www.a-rostin.de/index.html>
3. http://www.tetzl.de/java_logic_simulator.html
4. <http://www.sandeepdeb.com/software/dcd/index.shtml>
5. <http://ksimus.berlios.de/>
6. <http://qucs.sourceforge.net/>

7. <http://www.gpleda.org/>
8. <http://en.wikipedia.org/wiki/GEDA>
9. <http://www.geda.seul.org/wiki/>
10. <http://www.gedasymbols.org/>
11. <http://www.youtube.com/user/alpacadee>
12. <http://www.youtube.com/user/sonodrome>
13. <http://www.youtube.com/user/XbitElectronics#g/u>
14. <http://www.xbitinc.com/news/2/New-Products%2C-Tutorials%2C-Website-Updated.html>
15. <http://www.youtube.com/user/someprofs>
16. http://www.gedasymbols.org/user/kai_martin_knaak/footprints/connector/1x1PIN.fp
17. http://www.geda.seul.org/wiki/geda:gsch2pcb_tutorial
18. <http://www.gedasymbols.org/>
19. http://www.brorson.com/gEDA/land_patterns_20050129.pdf
20. <http://www.tkgate.org/>
21. <http://linuxcnc.org/>
22. http://wiki.linuxcnc.org/cgi-bin/emcinfo.pl?Debian_Lenny_Compile EMC2
23. http://wiki.linuxcnc.org/cgi-bin/emcinfo.pl?Debian_Lenny_Compile EMC2
24. <http://google.com/SketchUp.html>
25. <http://www.zeitoun.net/articles/google-sketchup-7-with-debian-lenny-on-eeepc-901/start>
26. <http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=554804>
27. <http://code.google.com/p/suplugins/>
28. <http://artofillusion.org/>
29. <http://vasek.wu.cz/artilusion/AoIManual/>
30. <http://simh.trailing-edge.com/>

Kapitola 72. Assemblery a překladače

*

Poznámky k překladačům.

Odkazy:

- [vasm](http://sun.hasenbraten.de/vasm/)¹ a portable and retargetable assembler
-

Poznámky

1. <http://sun.hasenbraten.de/vasm/>

Kapitola 73. Femto OS

Odkazy:

- Femto OS¹
- Femto OS shell application²
- Femto OS interrupt application³
- Femto OS queue application⁴
- Femto OS hook application⁵
-

Poznámky

1. <http://www.femtoos.org/>
2. <http://www.youtube.com/watch?v=CvtM8J9ZpJU>
3. <http://www.youtube.com/watch?v=UgocLNmY2Ro>
4. <http://www.youtube.com/watch?v=GFcYjVAh8hs>
5. <http://www.youtube.com/watch?v=KcSxaGPQ8UU>

Kapitola 74. UniFLEX

Odkazy:

- The Missing 6809 UniFLEX Archive¹ UniFLEX'09 for SWTP & Gimix - Plus ALL Products Ever Developed
- 6809 Emulation Page²
-
-

Poznámky

1. <http://www.rtmx.com/UniFLEX/>
2. <http://koti.mbnet.fi/~atjs/mc6809/>

Kapitola 75. UCSD Pascal

Odkazy:

- http://www.bitsavers.org/bits/UCSD_Pascal

UCSD Pascal je implementace překladače programovacího jazyka Pascal do virtuálního prostředí a rovněž implementace interpretu tohoto virtuálního počítače pro konkrétní HW architekturu. Svým způsobem je to velmi vzdálený předek Javy.

XIII. Historie výpočetní techniky

Muzea a podobné odkazy:

- Norsk Datahistorisk Forenings¹¹³⁸
- Glenn's Computer Museum¹¹³⁹
- The History of Computing Project¹¹⁴⁰
- Facts and stories about Antique (lonesome) Computers¹¹⁴¹
-
-
-

Poznámky

1138. <http://www.nodaf.no/index.php/Hovedside>

1139. <http://www.glennsmuseum.com/>

1140. <http://www.thocp.net/index.html>

1141. <http://ed-thelen.org/comp-hist/index.html>

Kapitola 76. Poznámky k historii

Odkazy:

- Computer Pioneers - Pioneer Computers Part 1¹
- Computer Pioneers - Pioneer Computers Part 2²
- On-Line Documents³
- Columbia University Computing History⁴
- The History of Computers⁵
- Computer Chronology⁶
- History of Computer Technology⁷

1936 Konrad Zuse — základní návrh počítače řízeného děrnou páskou. (Z3)

1936 Allan Turing — teoretické základy

Fyzik Howard Aiken

John Antanasov

1940 BTL1: Bell Labs⁸: George Stibitz — experimentoval s relé, binární jednobitová sčítací. Výsledkem projektu byl kalkulátor BTL1. Kalkulátor byl sprovozněn v roce 1939, vyřazen z provozu v roce 1949. Cena projektu cca \$20.000 byla důvodem pro zastavení dalších prací. Velikost BTL1 byla 8*5*1 stopa. Paměť obsahovala 2 komplexní čísla. Rychlost 0.2 operace za sekundu. Do konce války postavili v Bell Labs dalších 6 strojů. Největší z nich, dokončený v roce 1946 obsahoval 9.000 relé v několika místnostech. Potřeboval 20kW energie. Používal aritmetiku v plovoucí čárce, a stál \$500.000.

Ve videu Computer Pioneers - Pioneer Computers Part 1⁹ na 00:12:02 je ukázka mechanického logického obvodu AND.

1938 Z1: 64 slov paměti. Byl postaven z elektronických relé.

1941 Z3: Sprovozněn koncem roku 1941, zničen v roce 1944 v průběhu II. světové války. 1963 byla postavena jeho kopie. Počítač postavil vlastními náklady sám Konrad Zuse. Odhadovaná cena \$6.500. Velikost 2 moduly o rozměrech 6*3*1 stopa. Počítač měl paměť vlikou 64 slov, každé 22 bitů veliké. Používal aritmetiku v plovoucí řádové čárce. Program byl vyděrován do 35mm širokého filmu. Zvládal 0.5 operace za sekundu. Použití: analýza obvodů, obrana, prototyp. Sečtení dvou čísel v plovoucí řádové čárce trvalo asi 2 sekundy.

1942-1945 Z4. Zrekonstruován v 1950. V roce 1951 měl Zuse 1000 slov mechanické paměti s přístupovou dobou 1,5s. Logika reallizována v relé. Šírka slova 42 bitů. Aritmetika v plovoucí řádové čárce. V roce 1950 modifikován a přidána možnost podmíněného větvení. Byl používán až do roku 1959.

Po válce vybudoval Zuse firmu která stavěla počítačové stroje. Tato byla později absorbována firmou Siemens.

1942 ABC: John Vincent Atanasoff¹⁰ spolu se spolupracovníkem {Berry} uvedl do chodu stroj používající sériovou binární aritmetiku a kondenzátorovou paměť. Cena stroje byla \$7.000, a velikost asi 6*3*3 stopy. Použitá technologie: elektronky, kondenzátory, děrné štítky. Paměť 64 50-ti bitových slov. Rychlost 32 operací za sekundu. Použití: Lineární algebra. Na snímku Computer Pioneers - Pioneer Computers Part 1¹¹ je část jeho přednášky 00:22:25 až 00:28:30. Atanasoff byl první v USA kdo použil binární aritmetiku a elektronky k počítání. Jeho práce byla použita ke zrušení patentu ENIACu.

Harvard MARK I (ASCC): Howard Aiken a Grace Hopper. Počítač byl postaven v IBM. Sprovozněn roku 1943, vyřazen z provozu 1959. Cena cca \$500.000, velikost asi 51*8*8 stop. Byl dlouhý 51 stop 8 stop vysoký a hluboký 8 stop. Měl prosklenou přední stěnu.. Použité technologie: mechanické prvky, relé, děrné štítky propojovací desky (plugboards). Paměť měla 72 slov o šířce 23 číslic (na pozici 24 bylo znaménko). Program uložen na děrných štítcích nebo pásce. Rychlost 3.3 operací za sekundu. Použití: obrana. Na snímku Computer Pioneers - Pioneer Computers Part 1¹² 00:31:08 - 00:42:06 je část přednášky Grace Hopper. Na této ukázce zmiňuje i první počítačovou chybu (bug). MARK I pracoval 24 hodin denně, 7 dní týdně.

Počítač měl 72 registrů, každý měl vlastní sčítačku.

- Harvard Mark I¹³
- Inventors of the Modern Computer¹⁴
- ASCC operational manual¹⁵

SSEC IBM: {Wallace Eckert}, sprovozněn 1948, plocha 25*50 stop. Technologie: elektronky, relé, dřevné štítky. Paměť 160 číslic (elektronková), 300 (releová), 400,000 na dřevné pásce. Program na dřevných štítcích. Rychlost 50 násobení za sekundu. Použití: astronomie, obrana.

Inženýři stavějící SSEC později postavili IBM650.

Programátoři SSEC se později stali členy skupin programátorů 701.

76.1. Minipočítače

*

Odkazy:

- Minicomputer¹⁶ na Citizendii
- Computer Automation LSI 2 minicomputer¹⁷
-
-

První tranzistorové minipočítače

- PDP-8 ¶ předchůdci PDP-5, LINC, TX-0, TX-2 a PDP-1
-
-
-

Poznámky

1. <http://www.youtube.com/watch?v=qundvme1Tik>
2. <http://www.youtube.com/watch?v=wsirYCAocZk>
3. <http://ed-thelen.org/comp-hist/on-line-docs.html>
4. <http://www.columbia.edu/acis/history/index>.
5. http://www.web-friend.com/help/general/pc_history.html
6. <http://www.thesciencebookstore.com/compchron.asp>
7. http://blogs.siliconindia.com/parthawiproit/History_of_Computer_Technology-bid-KO10ZgsH13241631.html
8. http://cs.wikipedia.org/wiki/Bellovy_laborato%C5%99e
9. <http://www.youtube.com/watch?v=qundvme1Tik>
10. <http://www.columbia.edu/~td2177/JVAtanasoff/JVAtanasoff.html>
11. <http://www.youtube.com/watch?v=qundvme1Tik>
12. <http://www.youtube.com/watch?v=qundvme1Tik>
13. http://en.wikipedia.org/wiki/Harvard_Mark_I

14. <http://inventors.about.com/library/weekly/aa052198.htm>
15. http://www.bitsavers.org/pdf/harvard/MarkI_operMan_1946.pdf
16. <http://www.citizendia.org/Minicomputer>
- 17.

Kapitola 77. Výrobci počítačů

Firmy, školy a ústavy které projektovali a vyráběli počítače či jejich komponenty

77.1. Artronix

Odkazy:

- Artronix¹
-

77.1.1. PC12

Odkazy:

- PC12 minicomputer²
-

77.2. Control Data Corporation

Odkazy:

- http://en.wikipedia.org/wiki/Control_Data_Corporation

77.2.1. Různé

77.2.1.1. CDC display code

Odkazy:

- CDC display code³
-
-
-

Tabulka 77-1. CDC Display code characters (64-character character set version)

Binary	Decimal	Octal	ASCII	CDC	Name
000 000	0	00	:	:	colon
000 001	1	01	A	A	
000 010	2	02	B	B	
000 011	3	03	C	C	
000 100	4	04	D	D	
000 101	5	05	E	E	

Binary	Decimal	Octal	ASCII	CDC	Name
000 110	6	06	F	F	
000 111	7	07	G	G	
001 000	8	10	H	H	
001 001	9	11	I	I	
001 010	10	12	J	J	
001 011	11	13	K	K	
001 100	12	14	L	L	
001 101	13	15	M	M	
001 110	14	16	N	N	
001 111	15	17	0	0	
010 000	16	20	P	P	
010 001	17	21	Q	Q	
010 010	18	22	R	R	
010 011	19	23	S	S	
010 100	20	24	T	T	
010 101	21	25	U	U	
010 110	22	26	V	V	
010 111	23	27	W	W	
011 000	24	30	X	X	
011 001	25	31	Y	Y	
011 010	26	32	Z	Z	
011 011	27	33	0	0	
011 100	28	34	1	1	
011 101	29	35	2	2	
011 110	30	36	3	3	
011 111	31	37	4	4	
100 000	32	40	5	5	
100 001	33	41	6	6	
100 010	34	42	7	7	
100 011	35	43	8	8	
100 100	36	44	9	9	
100 101	37	45	+	+	
100 110	38	46	-	-	
100 111	39	47	*	*	
101 000	40	50	/	/	
101 001	41	51	((
101 010	42	52))	
101 011	43	53	\$	\$	
101 100	44	54	=	=	
101 101	45	55			blank
101 110	46	56	,	,	
101 111	47	57	.	.	

Binary	Decimal	Octal	ASCII	CDC	Name
110 000	48	60	#	≡	equiv
110 001	49	61	[[
110 010	50	62]]	
110 011	51	63	%	%	
110 100	52	64	"	≠	not eq
110 101	53	65	_	→	concat
110 110	54	66	!	∨	log OR
110 111	55	67	&	∧	log AND
111 000	56	70	'	↑	super
111 001	57	71	?	↓	sub
111 010	58	72	<	<	
111 011	59	73	>	>	
111 100	60	74	@	≤	
111 101	61	75	\	≥	
111 110	62	76	^	¬	NOT
111 111	63	77	;	;	

77.3. Data General

Odkazy:

- Data General⁴
-

77.3.1. Data General Nova

Odkazy:

- Data General Nova⁵
-

Populární 16-ti bitový počítač.

Obrázek 77-1. Tvar instrukcí počítače Nova

NO ACCUMULATOR-EFFECTIVE ADDRESS															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+															
0	0	0	OP CODE	@	INDEX										
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+															
								0	0	(0; DISP)	ZERO PAGE				
								0	1	(PC+ DISP)	PC RELATIVE				
								1	0	(AC2+ DISP)	INDEXED BY AC2				
								1	1	(AC3+ DISP)	INDEXED BY AC3				

ONE ACCUMULATOR-EFFECTIVE ADDRESS

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+															
0	OP CODE	AC	@	INDEX	DISPLACEMENT										
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+															

TWO ACCUMULATOR-MULTIPLE OPERATION

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+															
1	ACS	ACD	OP CODE	SH	C	#	SKIP								
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+															

INPUT/OUTPUT

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+																
0	1	1	AC		OP CODE		CONTROL		DEVICE CODE							
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+																

Při nepřímém přístupu s pomocí adres $20_{\text{OCT}}-27_{\text{OCT}}$ je adresa v těchto slovech automaticky zvětšena o 1 (*AutoIncrement*). Při nepřímém přístupu pomocí adres $30_{\text{OCT}}-37_{\text{OCT}}$ je adresa v těchto slovech automaticky zmenšena o 1 (*AutoDecrement*).

77.4. Digico

*

Odkazy:

- digico working group⁶
- Section 50. Manufacturers not connected to the ICL family⁷
- Most important machines and inventions⁸

Stroje:

- Digiac [1966] — zkonstruovaný speciálně pro zpracování dat z hmotového spektrometru. První stroj byl postaven na vánoce roku 1966. Celkem bylo prodáno 19 strojů.
- Micro 16S [1968] — První univerzální počítač postavený firmou Digico. Na zakázku NRDC (National Research Development Corporation). Bez jakékoliv britské konkurence se rychle stal úspěšným.
- Micro 16P [1970] — Náhrada zastarávajícího Micro 16S.
- Micro 16V [1972] — Postavený posledními technologiemi své doby s výhledem na 10 let. Zkonstruován z integrovaných obvodů. Rovněž do té doby běžná feritová paměť byla nahrazena polovodičovou.

77.4.1. Micro 16V

* *Attributy: id="Micro16V" ents=Micro_16V*

Odkazy:

-
-

Můj odhad: Minipočítač Micro 16 je jedním z ideových dědiců PDP-8. Firma Digico inspirovaná úspěchem PDP-8 se rozhodla postavit minipočítač, kterým by konkurovala PDP-8.

Obrázek 77-2. Formáty instrukcí a dat Micro 16V

Addressing Instructions	Operation	Address										
Non-Addressing Instructions	0 0 0 0	Function										
Input/Output Instructions	0 0 0 0	1	Function Code						Peripheral Cannel Number			
Shift Instructions	0 0 0 0	0 0	Function Code						Shift Required			
Shift Instructions	0 0 0 0	0 0	Function Code						NC	Shift Required		

Tabulka 77-2. Přehled základních instrukcí

mnemo	binary	octal	hex	popis
	0000.....	00....	0...	Instrukce bez adresy
INC y	0 001 nnn nnn nnn nnn	01nnnn	1nnn	Increment Store and Test
JPU y	0 010 nnn nnn nnn nnn	02nnnn	2nnn	Unconditional Jump
JPZ y	0 011 nnn nnn nnn nnn	03nnnn	3nnn	Jump if Zero
GET y	0 100 nnn nnn nnn nnn	04nnnn	4nnn	Get Store into Accumulator
STO y	0 101 nnn nnn nnn nnn	05nnnn	5nnn	Copy Accumulator into Store
GTI y	0 110 nnn nnn nnn nnn	06nnnn	6nnn	Indirect Get
STI y	0 111 nnn nnn nnn nnn	07nnnn	7nnn	Indirect Store
AND y	1 000 nnn nnn nnn nnn	10nnnn	8nnn	Logical AND
DEC y	1 001 nnn nnn nnn nnn	11nnnn	9nnn	Decrement Store and Test
JPS y	1 010 nnn nnn nnn nnn	12nnnn	Annn	Jump to a Subroutine
JPN y	1 011 nnn nnn nnn nnn	13nnnn	Bnnn	Jump if Negative
ADD y	1 100 nnn nnn nnn nnn	14nnnn	Cnnn	Addition
SUB y	1 101 nnn nnn nnn nnn	15nnnn	Dnnn	Subtraction
ADI y	1 110 nnn nnn nnn nnn	16nnnn	Ennn	Indirect Add
JSI y	1 111 nnn nnn nnn nnn	17nnnn	Fnnn	Indirect Jump ro a Subroutine

77.4.1.1. Sada instrukcí Micro 16V

*

Tabulka 77-3. Operační kódy instrukcí Micro 16V řazené podle operačního kódu

název instrukce	mnemo	strojový kód				popis
		bin	oc- tal	dec	hexa	
not	NOT	0 000 000 000 000 000		0	0000	Do Nothing Instruction
cry	CRY	0 000 000 000 000 001		1	0001	Read Carry Instruction
swb	SWB	0 000 000 000 000 010		2	0002	Get Switchbank Register
hlt	HLT	0 000 000 000 100 000		32	0020	Halt Instruction
fin	FIN	0 000 000 000 111 000		56	0038	Forbid interrupts
cru	CRU	0 000 000 001 000 000		64	0040	Unset Carry Register
cla	CLA	0 000 000 001 010 000		80	0050	Zeroise Accumulator
crs	CRS	0 000 000 010 000 000		128	0080	Set Carry Register
lkj	LKJ a	0 000 000 100 000 000		166	0100	Link Jump
shr	SHR c	0 000 001 000 00c ccc		152	0200	Shift Right
rci	RCI c	0 000 001 000 1cc ccc		144	0220	Right Circulate Invert
oca	OCA	0 000 001 000 110 000		160	0230	Ones Complement Accumulator

název instrukce	mnemo	strojový kód				popis
		bin	oc- tal	dec	hexa	
tca	TCA	0 000 001 000 110 001	0010 0551	0231		Twos Complement Accumulator
cir	CIR c	0 000 001 001 0cc ccc	0011 0066	0240		Circulate Right
rss	RSS c	0 000 001 010 ccc ccc	0012 0640	0280		Arithmetic Shift Right
shl	SHL c	0 000 001 011 ccc ccc	0013 0040	02c0		Shift Left
rsc	RSC c	0 000 001 100 ccc ccc	0014 0068	0300		Right Shift Carry
nca	NCA	0 000 001 100 010 000	0014 0084	0310		Negate Carry Instruction
reo	REO	0 000 100 000 xxx xxx	0040 0048	00800		Read Character Or
rso	RSO	0 000 100 001 xxx xxx	0041 0012	00840		Reader OR and Reader Start
sri	SRI	0 000 100 001 xxx xxx	0041 0012	00840		Skip if RI not set
prt	PRT	0 000 100 011 xxx xxx	0043 0040	008c0		Print/Punch
eci	ECI	0 000 100 100 111 111	0044 0067	0093f		Twos Complement Accumulator
red	RED	0 000 101 000 xxx xxx	0050 0060	00a00		Read Character
rst	RST	0 000 101 001 xxx xxx	0051 0062	00a40		Read and Reader Start
kbd	KBD	0 000 101 010 xxx xxx	0052 0068	00a80		Get Keyboard Character

název instrukce	mnemo	strojový kód				popis
		bin	oc- tal	dec	hexa	
skb	SKB	0 000 111 010 xxx xxx	0072	30120	e80	Skip if Keaboard not Bussy
srb	SRB	0 000 111 100 xxx xxx	0074	30400	f00	Skip if Reader is not Busy
inc	INC a	0 001 nnn nnn nnn nnn	0100	40961	1000	Increment Store and Test
jpu	JPU a	0 010 nnn nnn nnn nnn	0200	80922	2000	Unconditional Jump
jpz	JPZ a	0 011 nnn nnn nnn nnn	0300	10288	8000	Jump if Zero
get	GET a	0 100 nnn nnn nnn nnn	0400	10638	4000	Get
sto	STO a	0 101 nnn nnn nnn nnn	0500	20488	8000	Copy Accumulator into Store
gti	GTI a	0 110 nnn nnn nnn nnn	0600	24576	6000	Indirect Get
sti	STI a	0 111 nnn nnn nnn nnn	0700	28672	72000	Indirect Store
and	AND a	1 000 nnn nnn nnn nnn	1000	32768	8000	Logical AND
dec	DEC a	1 001 nnn nnn nnn nnn	1100	36864	9000	Decrement Store and Test
jps	JPS a	1 010 nnn nnn nnn nnn	1200	40968	a000	Jump to a Subroutine
jpn	JPN a	1 011 nnn nnn nnn nnn	1300	46056	b000	Jump if Negative
add	ADD a	1 100 nnn nnn nnn nnn	1400	49152	c000	Addition

název instrukce	mnemo	strojový kód				popis
		bin	oc- tal	dec	hexa	
sub	SUB a	1 101 nnn nnn nnn nnn	15000	60240	0000	Subtraction
adi	ADI a	1 110 nnn nnn nnn nnn	16000	60340	0000	Indirect Add
jsi	JSI a	1 111 nnn nnn nnn nnn	17000	60440	0000	Indirect Jump to a Subroutine

ADD

Jméno

ADD — Addition

Přehled

ADD addr

Popis

Instrukce přičte ke střadači obsah paměti určené adresními bity instrukčního slova. Překročí-li výsledek velikost slova, je nastaven příznak přenosu C.

$ACC + Mem[addr] \longrightarrow C, ACC$

mnemo	kód		
	binary	octal	hex
ADD a	1 100 nnn nnn nnn nnn	140000	c000

Odkazy

Související instrukce: ADI, GET, SUB.

- *micro16V computer manual*, strana 9

ADI

Jméno

ADI — Indirect Add

Přehled

ADI `addr`

Popis

mnemo	kód		
	binary	octal	hex
ADI a	1 110 nnn nnn nnn nnn	160000	e000

Odkazy

Související instrukce: ADD, GTI, STI.

•
•

AND

Jméno

AND — Logical AND

Přehled

AND `addr`

Popis

Instrukce spočte logický součin (AND) obsahu střadače ACC a paměti. Výsledek uloží do střadače.

$$\text{ACC} \wedge \text{Mem}[\text{addr}] \longrightarrow \text{ACC}$$

mnemo	kód		
	binary	octal	hex
AND a	1 000 nnn nnn nnn nnn	100000	8000

Odkazy

Související instrukce: ADD.

- *micro16V computer manual*, strana 10

CIR

Jméno

CIR — Circulate Right

Přehled

CIR c

Popis

Obsah střadače A je rotován do prava. Nejnižší bit b0 je přemístněn na místo bitu nejvyššího b15.

mnemo	kód		
	binary	octal	hex
CIR c	0 000 001 001 0cc ccc	001100	0240

Odkazy

Související instrukce: RCI, RSC, RSS, SHL, SHR.

•
•

CLA

Jméno

CLA — Zeroise Accumulator

Přehled

CLA

Popis

mnemo	kód		
	binary	octal	hex
CLA	0 000 000 001 010 000	000120	0050

Odkazy

Související instrukce: SHR.

•
•

CRS

Jméno

CRS — Set Carry Register

Přehled

CRS

Popis

mnemo	kód		
	binary	octal	hex
CRS	0 000 000 010 000 000	000200	0080

Odkazy

Související instrukce: SHR.

•
•

CRU

Jméno

CRU — Unset Carry Register

Přehled

CRU

Popis

mnemo	kód		
	binary	octal	hex
CRU	0 000 000 001 000 000	000100	0040

Odkazy

Související instrukce: SHR.

•
•

CRY

Jméno

CRY — Read Carry Instruction

Přehled

CRY

Popis

mnemo	kód		
	binary	octal	hex
CRY	0 000 000 000 000 001	000001	0001

Odkazy

Související instrukce: SHR.

•
•

DEC

Jméno

DEC — Decrement Store and Test

Přehled

DEC addr

Popis

Instrukce zmenší obsah paměti na adrese daní bity 11-0 instrukčního slova o jedničku. Je-li výsledek 0, přeskočí se následující instrukce.

$\text{Mem}[\text{addr}] - 1 \longrightarrow \text{Mem}[\text{addr}]; \text{ If } \text{Mem}[\text{addr}] = 0 \text{ then } P+1 \longrightarrow P$

mnemo	kód		
	binary	octal	hex
DEC a	1 001 nnn nnn nnn nnn	110000	9000

Odkazy

Související instrukce: INC, SUB.

- *micro16V computer manual*, strana 10

ECI

Jméno

ECI — Enable Clock Interrupts

Přehled

ECI

Popis

Je spuštěn časovač a povoleno přerušení od časovače. První přerušení následuje 10ms po vykonání instrukce ECI.

mnemo	kód		
	binary	octal	hex

mnemo	kód		
	binary	octal	hex
ECI	0 000 100 100 111 111	004477	093f

Odkazy

Související instrukce: SHR.

•
•

FIN

Jméno

FIN — Forbid interrupts

Přehled

FIN

Popis

mnemo	kód		
	binary	octal	hex
FIN	0 000 000 000 111 000	000070	0038

Odkazy

Související instrukce: SHR.

•
•

GET

Jméno

GET — Get Store into Accumulator

Přehled

GET *addr*

Popis

Instrukce naplní střadač ACC obsahem paměti na adrese uvedené v bitech 11-0 instrukčního slova. Horní bity adresy se doplní z registru aktuální stránky.

$\text{Mem}[\text{addr}] \longrightarrow \text{ACC}$

mnemo	kód		
	binary	octal	hex
GET <i>a</i>	0 100 <i>nnn nnn nnn</i> <i>nnn</i>	040000	4000

Odkazy

Související instrukce: GTI, STO, STI.

- *micro16V computer manual*, strana 9

GTI

Jméno

GTI — Indirect Get

Přehled

GTI *addr*

Popis

Mem[Mem[addr]] \longrightarrow A

mnemo	kód		
	binary	octal	hex
GTI a	0 110 nnn nnn nnn nnn	060000	6000

Odkazy

Související instrukce: LDX.

•
•

HLT

Jméno

HLT — Halt Instruction

Přehled

HLT

Popis

mnemo	kód		
	binary	octal	hex
HLT	0 000 000 000 100 000	000040	0020

Odkazy

Související instrukce: SHR.

•
•

INC

Jméno

INC — Increment Store and Test

Přehled

INC **addr**

Popis

Instrukce zvětší obsah paměti na adrese určené v bitech 11-0 instrukčního slova. Je-li výsledné číslo 0, přeskočí se následující instrukce.

$\text{Mem}[\text{addr}] + 1 \longrightarrow \text{Mem}[\text{addr}]; \text{ If } \text{Mem}[\text{addr}] = 0 \text{ then } P+1 \longrightarrow P$

mnemo	kód		
	binary	octal	hex
INC a	0 001 nnn nnn nnn nnn	010000	1000

Odkazy

Související instrukce: ADD, DEC.

- *micro16V computer manual*, strana 10

JPN

Jméno

JPN — Jump if Negative

Přehled

JPN addr

Popis

addr \longrightarrow P_{11:0}

mnemo	kód		
	binary	octal	hex
JPN a	1 011 nnn nnn nnn nnn	130000	b000

Odkazy

Související instrukce: JPU, JPZ.

·
·

JPS

Jméno

JPS — Jump to a Subroutine

Přehled

JPS addr

Popis

Registr P je zvětšen o 1 takže ukazuje na následující instrukci za JPS. Poté je jeho obsah uložen na adresu zadanou v instrukci. Zde slouží jako návratová adresa pro návrat z podprogramu. Poté je do registru P zavedena adresa o jedničku větší než je adresa v instrukci. Pokračuje se instrukcí na kterou ukazuje P.

P+1 \longrightarrow P

mnemo	kód		
	binary	octal	hex
JPS a	1 010 nnn nnn nnn nnn	120000	a000

Ukázky

```

; Hlavní program
JPS Q

Q-1:    LKJ          ; Instrukce umístněná těsně před návratovou adresu.
Q:      ; návratová adresa
        ; tělo podprogramu
JPU Q-1      ; Návrat z podprogramu skokem na instrukci LKJ.
```

Odkazy

Související instrukce: JPU, JPZ.

•
•

JPU

Jméno

JPU — Unconditional Jump

Přehled

JPU addr

Popis

addr \longrightarrow P_{11:0}

mnemo	kód		
	binary	octal	hex
JPU a	0 010 nnn nnn nnn nnn	020000	2000

Odkazy

Související instrukce: ADI, STI, STO.

•
•

JPZ

Jméno

JPZ — Jump if Zero

Přehled

JPZ addr

Popis

addr \longrightarrow P_{11:0}

mnemo	kód		
	binary	octal	hex
JPZ a	0 011 nnn nnn nnn nnn	030000	3000

Odkazy

Související instrukce: ADI, STI, STO.

•
•

JSI

Jméno

JSI — Jump to a Subroutine

Přehled

JSI *addr*

Popis

Registr P je zvětšen o 1 takže ukazuje na následující instrukci za JPS. Poté je jeho obsah uložen . . . Zde slouží jako návratová adresa pro návrat z podprogramu. Poté je do registru P zavedena adresa o jedničku větší než je adresa v instrukci. Pokračuje se instrukcí na kterou ukazuje P.

$P+1 \longrightarrow P$

mnemo	kód		
	binary	octal	hex
JSI a	1 111 nnn nnn nnn nnn	170000	f000

Ukázky

```

; Hlavní program
P:   JSI Q
.
.
.
Q:   adresa R
```

```

LKJ ; Instrukce umístěná těsně před návratovou adresou.
R:   ; návratová adresa P+1
      ; tělo podprogramu
JPU R-1 ; Návrat z podprogramu skokem na instrukci LKJ.
```

Odkazy

Související instrukce: JPU, JPZ.

•
•

KBD

Jméno

KBD — Get Keyboard Character

Přehled

KBD

Popis

mnemo	kód		
	binary	octal	hex
KBD	0 000 101 010 xxx xxx	005200	0a80

Odkazy

Související instrukce: SHR.

•
•

LKJ

Jméno

LKJ — Link Jump

Přehled

LKJ

Popis

Registr P je naplněn adresou ze slova následujícího instrukci LKJ. Jinstrukce slouží jako skok na libovolnou adresu v paměti.

$\text{Mem}[P+1] \longrightarrow P$

mnemo	kód		
	binary	octal	hex
LKJ a	0 000 000 100 000 000	000400	0100

Odkazy

Související instrukce: JPS, JSI.

•
•

NCA

Jméno

NCA — Negate Carry Instruction

Přehled

NCA

Popis

mnemo	kód		
	binary	octal	hex

mnemo	kód		
	binary	octal	hex
NCA	0 000 001 100 010 000	001420	0310

Odkazy

Související instrukce: SHR.

•
•

NOT

Jméno

NOT — Do Nothing Instruction

Přehled

NOT

Popis

mnemo	kód		
	binary	octal	hex
NOT	0 000 000 000 000 000	000000	0000

Odkazy

Související instrukce: SHR.

•
•

OCA

Jméno

OCA — Ones Complement Accumulator

Přehled

OCA

Popis

mnemo	kód		
	binary	octal	hex
OCA	0 000 001 000 110 000	001060	0230

Odkazy

Související instrukce: SHR.

·
·

PRT

Jméno

PRT — Print/Punch

Přehled

PRT

Popis

mnemo	kód		
	binary	octal	hex
PRT	0 000 100 011 xxx xxx	004300	08c0

Odkazy

Související instrukce: SHR.

•
•

RCI

Jméno

RCI — Right Circulate Invert

Přehled

RCI c

Popis

mnemo	kód		
	binary	octal	hex
RCI c	0 000 001 000 1cc ccc	001040	0220

Odkazy

Související instrukce: SHR.

•
•

RED

Jméno

RED — Read Character

Přehled

RED

Popis

Akumulátor je promazán a do jeho bitů 0-7 je nahrán znak z konzoly. Rovněž je smazán příznak RI.

mnemo	kód		
	binary	octal	hex
RED	0 000 101 000 xxx xxx	005000	0a00

Odkazy

Související instrukce: SHR.

•
•

REO

Jméno

REO — Read Character Or

Přehled

REO

Popis

Do akumulátoru je uložen logický součet akumulátoru a 8-mi bitů z konzoly. Rovněž je smazán příznak RI.

mnemo	kód		
	binary	octal	hex
REO	0 000 100 000 xxx xxx	004000	0800

Odkazy

Související instrukce: SHR.

•
•

RSC

Jméno

RSC — Right Shift Carry

Přehled

RSC c

Popis

mnemo	kód		
	binary	octal	hex
RSC c	0 000 001 100 ccc ccc	001400	0300

Odkazy

Související instrukce: SHR.

•

.

RSO

Jméno

RSO — Reader OR and Reader Start

Přehled

RSO

Popis

... Rovněž je smazán příznak RI. Poté je spuštěna čtecí operace konzoly.

mnemo	kód		
	binary	octal	hex
RSO	0 000 100 001 xxx xxxx	004100	0840

Odkazy

Související instrukce: SHR.

.

.

RSS

Jméno

RSS — Arithmetic Shift Right

Přehled

RSS c

Popis

mnemo	kód		
	binary	octal	hex
RSS c	0 000 001 010 ccc ccc	001200	0280

Odkazy

Související instrukce: SHR.

·
·

RST

Jméno

RST — Read and Reader Start

Přehled

RST

Popis

Akumulátor je promazána do jeho dolních 8-mi bitů načten znak z konzoly. Rovněž je smazán příznak RI. Poté je spuštěna čtecí operace konzoly.

mnemo	kód		
	binary	octal	hex
RST	0 000 101 001 xxx xxx	005100	0a40

Odkazy

Související instrukce: SHR.

•
•

SHL

Jméno

SHL — Shift Left

Přehled

SHL *c*

Popis

mnemo	kód		
	binary	octal	hex
SHL <i>c</i>	0 000 001 011 <i>ccc</i> <i>ccc</i>	001300	02 <i>c</i> 0

Odkazy

Související instrukce: JPS, JSI.

•
•

SHR

Jméno

SHR — Shift Right

Přehled

SHR c

Popis

Instrukce posouvá bity střadače do prava. Nejméně významný bit střadače je vždy zahazován. Nejvýznamnější bit střadače se plní nulou.

mnemo	kód		
	binary	octal	hex
SHR c	0 000 001 000 00c ccc	0010cc	0200

Odkazy

Související instrukce: JPS, JSI.

·
·

SKB

Jméno

SKB — Skip if Keaboard not Bussy

Přehled

SKB

Popis

mnemo	kód		
	binary	octal	hex
SKB	0 000 111 010 xxx xxx	007200	0e80

Odkazy

Související instrukce: SHR.

•
•

SRB

Jméno

SRB — Skip if Reader is not Busy

Přehled

SRB

Popis

mnemo	kód		
	binary	octal	hex
SRB	0 000 111 100 xxx xxx	007400	0f00

Odkazy

Související instrukce: SHR.

•
•

SRI

Jméno

SRI — Skip if RI not set

Přehled

SRI

Popis

mnemo	kód		
	binary	octal	hex
SRI	0 000 100 001 xxx xxx	004100	0840

Odkazy

Související instrukce: SHR.

•
•

STI

Jméno

STI — Indirect Store

Přehled

STI addr

Popis

$A \longrightarrow \text{Mem}[\text{Mem}[\text{addr}]]$

mnemo	kód		
	binary	octal	hex
STI a	0 111 nnn nnn nnn nnn	070000	7000

Odkazy

Související instrukce: ADI, STI, STO.

•
•

STO

Jméno

STO — Copy Accumulator into Store

Přehled

STO **addr**

Popis

Anstrukce uloží obsah střadače ACC na adresu danou bity 11-0 instrukce.

ACC \longrightarrow Mem[addr]

mnemo	kód		
	binary	octal	hex
STO a	0 101 nnn nnn nnn nnn	050000	5000

Odkazy

Související instrukce: GET, GTI, STI.

- *micro16V computer manual*, strana 9

SUB

Jméno

SUB — Subtraction

Přehled

SUB **addr**

Popis

Instrukce odečet obsah paměti určené adresními bity instrukčního slova od střadače, a výsledek uloží ve střadači. Dojde-li k podtečení, je nastaven příznak C.

$ACC - Mem[addr] \longrightarrow C, ACC$

mnemo	kód		
	binary	octal	hex
SUB a	1 101 nnn nnn nnn nnn	150000	d000

Odkazy

Související instrukce: ADD, GET, STO.

- *micro16V computer manual*, strana 9

SWB

Jméno

SWB — Get Switchbank Register

Přehled

SWB

Popis

mnemo	kód		
	binary	octal	hex
SWB	0 000 000 000 000 010	000002	0002

Odkazy

Související instrukce: SHR.

•
•

TCA

Jméno

TCA — Twos Complement Accumulator

Přehled

TCA

Popis

mnemo	kód		
	binary	octal	hex
TCA	0 000 001 000 110 001	001061	0231

Odkazy

Související instrukce: SHR.

•
•

77.5. Digital Equipment Corporation

Odkazy:

- Digital Equipment Corporation⁹
- dokumenty o počítačích firmy DEC¹⁰ na bitsavers.org
- architecture18b.pdf¹¹
- PDP-8 Frequently Asked Questions (posted every other month)¹²
- GBell's CyberMuseum for Digital Equipment Corp (DEC): Documents, Photo Albums, Talks, and Video-tapes about Computing History¹³

Tabulka 77-4. Stručný přehled modelů PDP firmy Digital

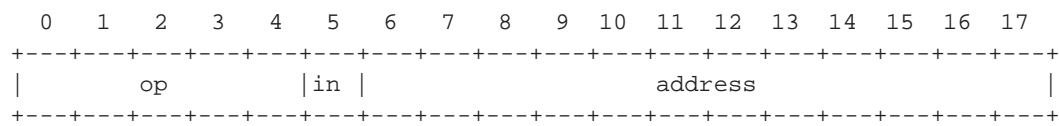
počítač	arch.	datum	prodáno	adresa	popis
PDP-1	18 bit	1960, listopad	50		
PDP-2	24				Never built? Prototype only?
PDP-3	36				One built by a customer, not by DEC
PDP-4	18 bit	1962, červenec	45	13	Předchůdce PDP-7
PDP-5	12 bit	1963		12	Předchůdce PDP-8
PDP-6	36 bit	1964	23	18	Většina postavena pro MIT.
PDP-7	18 bit	1964, prosinec	120	13	Široce používaný pro řízení v reálném čase
PDP-8	12 bit	1965	~50 000	12	Následník PDP-5
PDP-9	18 bit	1966, srpen	445	13	Upgrade PDP-7
PDP-10	36 bit	1967	~700	18	Následník PDP-6, prodáváný také pod označením DECSYSTEM-20
PDP-11	16 bit	1970	>600 000	16	První a jediný 16-ti bitový počítač fy DEC
PDP-12	12 bit			12	PDP-8 + LINC
PDP-13					13 nosí smůlu, toto značení nebylo nikdy použito
PDP-14					Programovatelný řadič s pamětí programu ROM
PDP-15	18 bit	1970, květen	790		TTL upgrade PDP-9
PDP-16	8/16	1972		9	A register-transfer module system.
PDP-X	16 bit				Prototyp. Jeden z kandidátů na PDP-11.

77.5.1. 18-ti bitová architektura

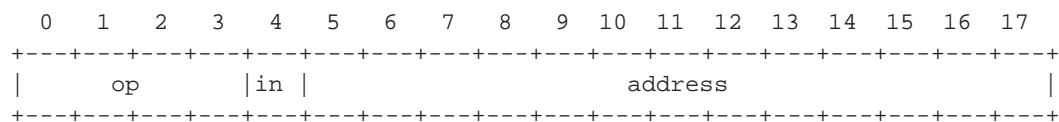
První počítač, který ve firmě DEC vznikl byl PDP-1. Byl to současně první čelní 18-ti bitové rodiny počítačů. Jeho následníci přinesli další vylepšení ale základní koncept zůstal.

Rodina 18-ti bitových počítačů začíná úplně prvním postaveným počítačem PDP-1 a pokračuje počítači PDP-4, PDP-7, PDP-9 a PDP-15.

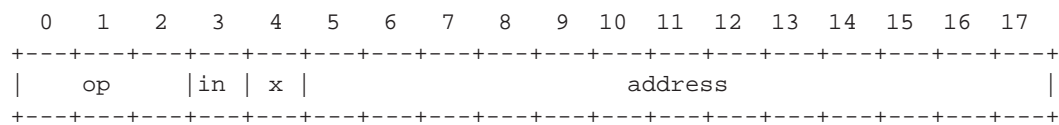
Obrázek 77-3. Základní instrukční slovo PDP-1



Obrázek 77-4. Základní instrukční slovo PDP-4



Obrázek 77-5. Základní instrukční slovo PDP-15



77.5.1.1. PDP-1

*

Tabulka 77-5. Přehled instrukcí PDP-1

Mnemo	Code	us	popis
	00		
and Y	02	10	
ior Y	04		
xor Y	06		
xct Y	10		
	12		
	14		
cal	16	10	
jda	17	10	
lac Y	20		
lio Y	22		
dac Y	24	10	
dap Y	26	10	
dip Y	30	10	
dio Y	32	10	
dzm Y	34		
	36		
add Y	40	10	

Mnemo	Code	us	popis
sub Y	42		
idx Y	44		
isp Y	46		
sad Y	50		
sas Y	52		
mus Y	54		
dis Y	56	10	
jmp Y	60		
jsp Y	62		
skp	64		
shift	66		
law N	70		
law -N	71		
iot	72		
	74		
opr	76		

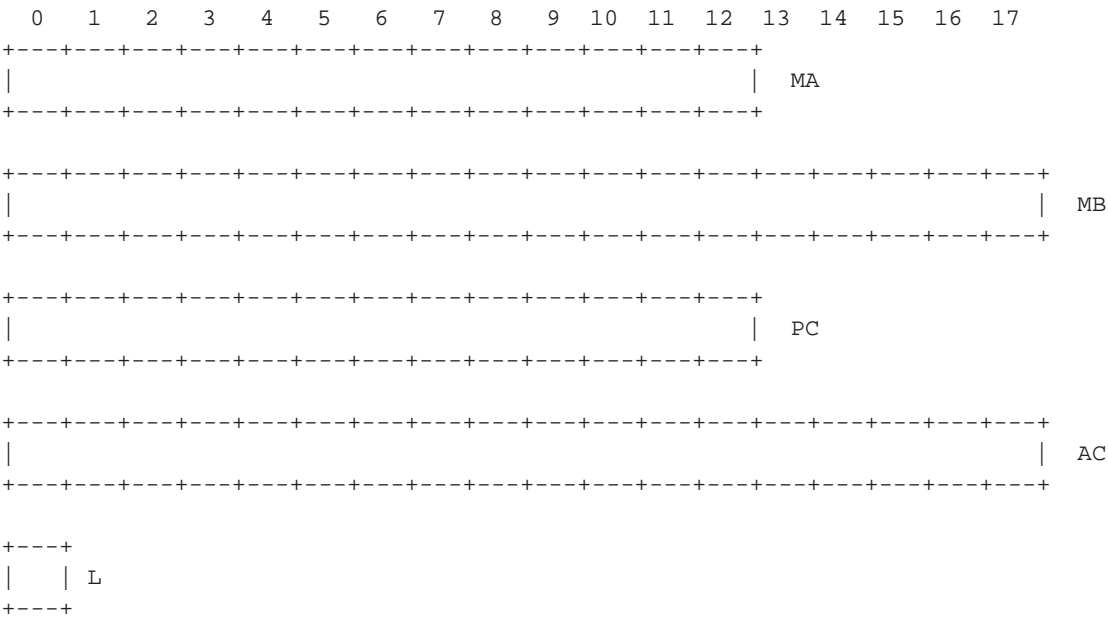
77.5.1.2. PDP-4

*

Fakta:

- Operační paměť má velikost 1024, 4096 nebo 8192 slov.
-
-

Obrázek 77-6. Registry PDP-4



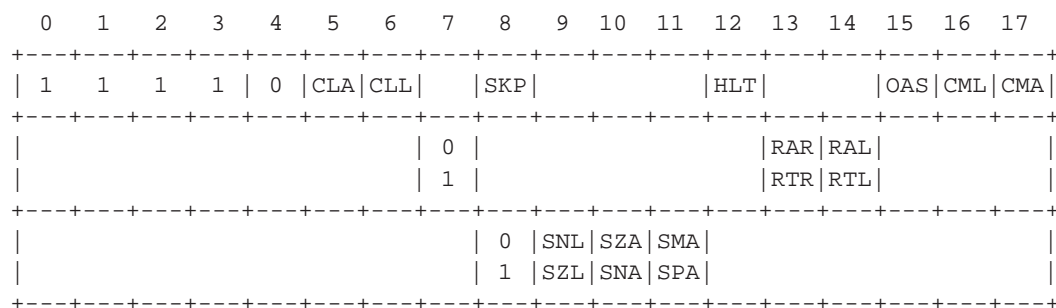
Tabulka 77-6. Přehled instrukcí PDP-4

Mnemo	Code	us	popis
cal Y	00	16	Call Subroutine
dac Y	04	16	Deposit AC
jms Y	10	16	Jump to subroutine
dzm Y	14	16	
lac Y	20	16	Load AC
xor Y	24	16	Exclusive OR: $AC \vee Y \rightarrow AC$
add Y	30	16	
tad Y	34	16	
xct Y	40	8+	Execute
isz Y	44	16	Index and skip if zero
and Y	50	16	AND
sad Y	54	16	Skip if AC and Y differ
jmp Y	60	8	Jump
	64		
iot	70		
opr	74		
law N	76	8	

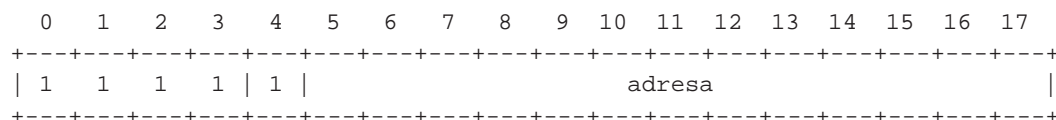
77.5.1.3. PDP-7

*

Obrázek 77-7. PDP-7 Operate Group 1



Obrázek 77-8. PDP-7 Operate Group 2 (LAW)



Tabulka 77-7. Přehled instrukcí PDP-7

Mnemo	Code	cycl	popis
cal Y	00	2	Call Subroutine
dac Y	04	2	Deposit AC
jms Y	10	2	Jump to subroutine
dzm Y	14	2	Deposit zero in memory.
lac Y	20	2	Load AC
xor Y	24	2	Exclusive OR: AC xor Y \rightarrow AC
add Y	30	2	Add (1's complement)
tad Y	34	2	Two's complement add.
xct Y	40	1+	Execute
isz Y	44	2	Increment and skip if zero
and Y	50	2	AND
sad Y	54	2	Skip if AC and Y differ
jmp Y	60	1	Jump
	64		
iot	70		
opr	74		
law N	76	8	

77.5.2. 12-ti bitová rodina

Odkazy:

- pdp8.hachti.de¹⁴
-

První z této rodiny byl PDP-5. Konstrukce tohoto počítače byla inspirována počítači CDC-160 a LINC. Tyto stroje mají 12-ti bitové slovo, adresují obvykle 4K slov paměti a mají jednoduchý ale silný soubor instrukcí. V době svého uvedení na trh to byl nejlevnější dostupný počítač.

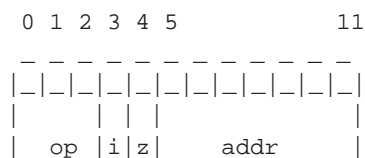
Další charakteristickou vlastností této rodiny počítačů je použití jednoho 12-ti bitového střadače a jedno-bitového registru kam se ukládá 13-tý bit při sčítání. Programátorem použitelné registry jsou:

Tabulka 77-8. Registry PDP-5, PDP-8

registr	použití
PC	12-ti bitový registr, ukazatel instrukcí.
AC	12-ti bitový střadač
L	link bit, 13-tý bit střadače

Velmi jednoduchý formát instrukčního slova umožňoval použití jen 8-mi instrukcí a 4 adresních módů.

Obrázek 77-9. Tvar instrukčního slova PDP-5 a PDP-8



op - the opcode.

- i - the indirect bit (0 = direct, 1 = indirect).

z - the page bit (0 = page zero, 1 = current page).

addr - the word in page.

Tabulka 77-9. Základní instrukce PDP-5, PDP-8

kód	mnemo	popis
000	AND	bitový operace AND mezi operandem a střadačem
001	TAD	operand je přičten ke střadači
010	ISZ	zvětšení operandu o 1 a přeskočení následující instrukce je-li výsledek 0
011	DCA	uloží obsah střadače do paměti a střadač vymaže
100	JMS	skok do podprogramu
101	JMP	skok
110	IOT	vstupní a výstupní instrukce
111	OPR	mikrokódované operace

Další vlastností je automatické zvětšení hodnoty v paměti na adresách 0010_{OCT} až 0017_{OCT} pokud jsou použity při nepřímém adresování. K tomuto zvětšení hodnoty dochází před použitím jejich hodnoty.

Tabulka 77-10. Modely PDP-8

model	výroba	prodáno	cena	technologie	poznámka
PDP-5	1963-67	116		Transistor	
PDP-8	1965-69	1450	\$18,500	Transistor	
LINC-8	1966-69	142	\$38,500	Transistor	
PDP-8/S	1966-70	1024	\$10,000	Transistor	velmi pomalý
PDP-8/I	1968-71	3698	\$12,800	TTL	
PDP-8/L	1968-71	3902	\$8,500	TTL	Ořezaný 8/I
PDP-12	1969-73?	3500?	\$27,900	TTL	LINC-8 + PDP-8
PDP-8/E	1970-78	>10,000	\$6,500	TTL MSI Omnibus	
PDP-8/F	1972-78?	>10,000	<\$5,000	TTL MSI Omnibus	založen na 8/E CPU
PDP-8/M	1972-78?	>10,000	<\$5,000	TTL MSI Omnibus	OEM verze 8/F
PDP-8/A	1975-84?	>10,000	\$1,317	TTL MSI Omnibus	nový CPU nebo 8/E CPU
VT78	1978-80		\$7,995	Intersil 6100	pracovní stanice

model	výroba	prodáno	cena	technologie	poznámka
DECmate I	1980-1984			Harris 6120	pracovní stanice
DECmate II	1982-1986		\$1,435	Harris 6120	pracovní stanice
DECmate III	1984-1990		\$2,695	Harris 6120	pracovní stanice
DECmate III+	1985-1990			Harris 6120	pracovní stanice

Tabulka 77-11. Modely kompatibilní od jiných výrobců

model	výroba	výrobce, poznámka
TPA1001	1969	Hungarian, KFKI, tranzistorový
TPA1001/i	1971	Hungarian, KFKI, IC verze TPA1001
TPA/i	1971	Hungarian, KFKI, přejmenovaný TPA1001/i
TPA/l	197?	Hungarian, KFKI, rozšířený TPA/i
TPA/l/128H	197?	Hungarian, KFKI, TPA/l s 128K paměti
TPA/s	197?	Hungarian, KFKI, založen na čipu Intersil
TPA Quadro	198?	Hungarian, KFKI, porovnatelný s DECmate
Electronica-100	?	Ruský, tranzistorový
Electronica-100I	?	Ruský, pravděpodobně klon PDP-8/I
Electrotechnica-100I	?	Jugoslávský, pravděpodobně klon PDP-8/I
Saratov-2	?	Ruský, postaven jako PDP-8/M
SPEAR u-LINC 100	?	SPEARm Inc, Waltham Mass (klon LINC)
SPEAR u-LINC 300	?	SPEARm Inc, Waltham Mass (klon LINC)
DCC-112	1970	Digital Computer Controls, klon PDP-8/L
DCC-112H	1971	Digital Computer Controls
MPS-1	1974	Fabritek, klon PDP-8/L
MP-12	1974	Fabritek, jiné značení pro MPS-1
6100 Sampler	1976?	Intersil, IM6100 kit
Intercept I	197?	Intersil, založen na IM6100
Intercept Jr	197?	Intersil, založen na IM6100
TLF MINI-12	1977	Založen na IM6100
PCM-12	197?	Pacific CyberMetrix, založen na Intercept bus
PCM-12A	1977	Pacific CyberMetrix, PCM-12 taktovaný na 4MHz
SBC-8	1984-88	CESI, založen na IM6120? SCSI bus

Bližší informace o tomto počítači jsou v kapitole PDP-8

77.5.3. PDP-11

Odkazy:

- PDP-11¹⁵ na Wikipedii

- PDP-11 a SMEP: Systém Malých Elektronických Počítačů¹⁶
- PDP-11 FAQ¹⁷
- Computer Organization and (PDP-11) Programming¹⁸ by Donald Fisk
-
-
-

Minipočítač PDP-11 patří mezi nejúspěšnější počítače z rodiny PDP firmy DEC. Je to první model s klasickou 8-mi/16-ti bitovou strukturou. Paměť je organizována po 8-mi bitových bajtech a je adresována 16-ti bitovými slovy. Minipočítač se vyznačuje velmi úhlednou a pravoúhlou sadou instrukcí. Je to počítač na kterém byl naprogramován první UNIX.

77.5.3.1. Instrukční sada PDP-11

Odkazy:

- What is the PDP-11 instruction set?¹⁹
- PDP-11 Processor Handbook²⁰
-
-

77.5.3.1.1. Operandy instrukcí

V řadě případů je operand instrukce specifikován 3-mi nebo 6-ti bity. Pokud je jen tříbitový, jedná se o registr. Pokud je 6-ti bitový, jedná se o informaci adresním módu a registr.

```

+---+---+---+---+---+---+
|   Mode   |   Reg   |
+---+---+---+---+---+---+

```

V poli Reg je číslo registru. Registry jsou číslovány od R0 až do R7. Registr R7 má zvláštní postavení mezi registry, je to čítač instrukcí PC, je to tedy ukazatel do paměti na další instrukci.

Mode	Name	Syntax	Effect
000	Register	Rn	op≡Rn
001	Register Deferred	(Rn)	op≡mem[Rn]
010	Autoincrement	(Rn)+	op≡mem[Rn]; inc Rn
011	Autoincrement Deferred	@(Rn)+	op≡mem[mem[Rn]]; inc Rn
100	Autodecrement	-(Rn)	dec Rn; op≡mem[Rn]
101	Autodecrement Deferred	@-(R)	dec Rn; op≡mem[mem[Rn]]
110	Index	x(Rn)	x=mem[PC]; inc PC; op=mem[x+Rn]
111	Index Deferred	@x(Rn)	x=mem[PC] inc PC; op≡mem[mem[x+Rn]]

Ponechání čítače instrukcí mezi běžnými registry nám poskytuje další adresní módy. Není třeba zavádět nové operace, protože žádaného výsledku dosáhneme použitím vybraného adresního módu společně s registrem R7. Pro tato použití je zavedena nová syntaxe.

Pokud potřebujeme načíst do vybraného registru, například R0 přímou hodnotu, zapíšeme to

```
MOV 23,R0
```

Tato instrukce zabírá v programu dvě slova a jedná se vlastně o použití registru R7 s autoinkrementací (mód 010).

```
MOV (R7)+, R0
DW 23
NOP
```

Technicky to znamená že se přečte obsah slova na adrese kam ukazuje R7, což je slovo následující za aktuální instrukcí MOV. Tento obsah se uloží do registru R0. Poté se R7 zvětší o 2 což zajistí že bude ukazovat na instrukci nacházející se za přečteným literálním slovem, tedy na instrukci NOP.

Mode	Name	Syntax	Effect
010	Immediate	#n	op≡mem[PC]; inc PC
011	Absolute	@#n	op≡mem[mem[PC]]; inc PC
110	Relative	n	x=mem[PC]; inc PC; op≡mem[x+PC]
111	Relative Deferred	@n	x=mem[PC]; inc PC; op=mem[mem[x+PC]]

Procesor nemá mezi registry žádný zásobníkový ukazatel. Ono ho ani není třeba. Jako zásobník se dá totiž vzhledem k bohatým adresním módům použít libovolný registr. Bývá zvykem jako zásobník používat rgeistr R6. Operace PUSH a POP nad zásobníkem se dají zapsata takto:

```
; PUSH R0
MOV R0, -(R6)
; POP R1
MOV (R6)+, R1
```

Instrukce návratu z podprogramu se pak dá zapsat takto:

```
; RETURN
MOV (R6)+, R7
```

77.5.3.1.2. Dvouoperandové instrukce

```

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| b | Opcode | Source | Destination |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Bit 15 specifikuje velikost operandů. Má-li hodnotu 0, pak se jedná o operandy velikosti slova (16 bitů). Má-li hodnotu 1, pak se jedná o operandy velikost bytu (8 bitů).

Opcode	Popis
b 000	Jiné než dvouoperandové instrukce
b 001 ssssss dddddd	MOV/MOVB
b 010 ssssss dddddd	CMP/CMPB
b 011 ssssss dddddd	BIT/BITB
b 100 ssssss dddddd	BIC/BICB
b 101 ssssss dddddd	BIS/BISB
b 110 ssssss dddddd	ADD(b=?)/SUB(b=?)

Opcode	Popis
b 111	Další aritmetické funkce

77.5.3.1.3. Jednooperandové instrukce

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+																
b 0 0 0				Instruction						Destination						
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+																

Bit 15 (b) má stejný význam jako u dvouoperandových instrukcí.

Instrukce	
0 000 000 000 000 000	HALT
0 000 000 000 000 001	WAIT
0 000 000 000 000 010	RTI
0 000 000 000 000 011	BPT
0 000 000 000 000 100	IOT
0 000 000 000 000 101	RESET
0 000 000 000 000 110	RTT
0 000 000 000 000 111	nepoužito
0 000 000 000	
0 000 000 000 111 111	
0 000 000 001 ddd ddd	JMP
0 000 000 010 000 rrr	RTS
0 000 000 010 001 000	nepoužito
0 000 000 010 010 ...	
0 000 000 010 010 111	
0 000 000 010 011 nnn	SPL
0 000 000 010 100 000	NOP
0 000 000 010 100 001	condition codes
0 000 000 010 1.. ..	
0 000 000 010 111 111	
b 000 000 011 dddddd	SWAB/BPL
b 000 101 000 dddddd	CLR/CLRB
b 000 101 001 dddddd	COM/COMB
b 000 010 010 dddddd	INC/INCB
b 000 101 011 dddddd	DEC/DECB
b 000 101 100 dddddd	NEG/NEGB
b 000 101 101 dddddd	ADC/ADCB
b 000 101 110 dddddd	SBC/SBCB

Instrukce	
b 000 101 111 dddddd	TST/TSTB
b 000 110 000 dddddd	ROR/RORB
b 000 110 001 dddddd	ROL/ROLB
b 000 110 010 dddddd	ASR/ASRB
b 000 110 011 dddddd	ASL/ASLB
b 000 110 100 dddddd	MARK/MPTS
b 000 110 101 dddddd	MPFI/MPFD
b 000 110 110 dddddd	MTPI/MTPD
b 000 110 111 dddddd	SXT/MFPS
0 000 000 1dd dddddd	BR
0 000 001 0dd dddddd	BNE
0 000 001 1dd dddddd	BEQ
0 000 010 0dd dddddd	BGE
0 000 010 1dd dddddd	BLT
0 000 011 0dd dddddd	BGT
0 000 011 1dd dddddd	BLE
1 000 000 0dd dddddd	BPL
1 000 000 1dd dddddd	BMI
1 000 001 0dd dddddd	BHI
1 000 001 1dd dddddd	BLOS
1 000 010 0dd dddddd	BVC
1 000 010 1dd dddddd	BVS
1 000 011 0dd dddddd	BCC≡BHS
1 000 011 1dd dddddd	BVS≡BLO

77.5.3.1.4. Rozšířená sada instrukcí EIS

Rozšířená sada instrukcí používá operační kódy začínající 0 111. Rozšířená sada obsahuje instrukce: MUL, DIV, ASH, ASHC, XOR

77.5.3.1.5. Přehled instrukcí podle operačního kódu

Tabulka 77-12. Operační kódy instrukcí PDP-11

0 000 000 000 000 000	HALT		
0 000 000 011 dddddd	SWAB		
0 000 101 000 dddddd	CLR		
0 000 101 001 dddddd	COM		
0 000 101 010 dddddd	INC		
0 000 101 011 dddddd	DEC		
0 000 101 100 dddddd	NEG		

0 000 101 111 dddddd	TST		
0 000 110 000 dddddd	ROR		
0 000 110 001 dddddd	ROL		
0 000 110 010 dddddd	ASR		
0 000 110 011 dddddd	ASL		
0 001 ssssss dddddd	MOV		
0 010 ssssss dddddd	CMP		
0 011 ssssss dddddd	BIT		
0 100 ssssss dddddd	BIC		
0 101 ssssss dddddd	BIS		
0 110 ssssss dddddd	ADD		
0 111 000 rrr ssssss	MUL		
0 111 001 rrr ssssss	DIV		
0 111 010 rrr ssssss	ASH		
0 111 011 rrr ssssss	ASHC		
0 111 100 rrr dddddd	XOR		
1 000 101 000 dddddd	CLRB		
1 000 101 001 dddddd	COMB		
1 000 101 010 dddddd	INCB		
1 000 101 011 dddddd	DECB		
1 000 101 100 dddddd	NEGB		
1 000 101 111 dddddd	TSTB		
1 000 110 000 dddddd	RORB		
1 000 110 001 dddddd	ROLB		
1 000 110 010 dddddd	ASRB		
1 000 110 011 dddddd	ASLB		
1 001 ssssss dddddd	MOVB		
1 010 ssssss dddddd	CMPB		
1 011 ssssss dddddd	BITB		
1 100 ssssss dddddd	BICB		
1 101 ssssss dddddd	BISB		
1 110 ssssss dddddd	SUB		

77.5.3.1.6. Abecední seznam instrukcí

ADD

Jméno

ADD — Hodnota zdrojového operandu je přičtena k cílovému operandu

Přehled

ADD *src, dst*

N	Z	V	C
*	*	*	*

Popis

ADD	0 110 ssssss ddddd
-----	--------------------

Ukázky

Odkazy

Podobné instrukce: SUB.

ASH

Jméno

ASH — Aritmetický posun doprava nebo doleva o více pozic

Přehled

ASH *src, R*

Obsah registru je posunut o počet bitů určený zdrojovým operandem. Ze zdrojového operandu se bere jen spodních 6 bitů a považuje se za čísla od -32 do +31. Záporná čísla způsobují posuv doprava, kladná doleva.

N	Z	V	C
*	*	0	*

Popis

ASH	0 111 010 rrr ssssss
-----	----------------------

Ukázky

ASH R3, R0 ; Obsah R0 je posunut o R3 míst.

Odkazy

Podobné instrukce: ASHC, ASL, ASR.

ASHC

Jméno

ASHC — Aritmetický posun 32 bitového čísla doprava nebo doleva o více pozic

Přehled

ASHC *src, R*

Obsah registru R_n a R_{n+1} je posunut o počet bitů určený zdrojovým operandem. Ze zdrojového operandu se bere jen spodních 6 bitů a považuje se za čísla od -32 do +31. Záporná čísla způsobují posuv doprava, kladná doleva.

N	Z	V	C
*	*	0	*

Popis

ASHC	0 111 011 rrr ssssss
------	----------------------

Odkazy

Podobné instrukce: ASH, ASL, ASR.

ASL

Jméno

ASL, ASLB — Aritmetický posun doleva

Přehled

ASL *dst*

ASLB *dst*

Všechny bity cílového operandu jsou posunuty o jedno místo doleva. Nejnižší bit je naplněn nulou a obsah nejvyššího bitu se přenáší do C.

N	Z	V	C
*	*	*	*

Popis

ASL	0 000 110 011 ddddd
ASLB	1 000 110 011 ddddd

Odkazy

Podobné instrukce: ASR, ASRB, ROL, ROLB.

ASR

Jméno

ASR, ASRB — Aritmetický posun doprava

Přehled

ASR *dst*

ASRB *dst*

Všechny bity cílového operandu jsou posunuty o jedno místo doprava. Obsah nejnižšího bitu je přenesen do C. Obsah znaménkového bitu (nejvyššího) se nemění.

N	Z	V	C
*	*	*	*

Popis

ASR	0 000 110 010 ddddd
ASRB	1 000 110 010 ddddd

Odkazy

Podobné instrukce: ASL, ASLB, ROR, RORB, SWAB.

BIC

Jméno

BIC, BICB — Nulování bitů v operandu

Přehled

BIC *src, dst*

BICB *src, dst*

V cílovém operandu jsou vynulovány bity, které jsou nastaveny v operandu zdrojovém. Zdrojový operand slouží jako bitová maska.

Popis

BIC	0 100 ssssss ddddd
BICB	1 100 ssssss ddddd

Odkazy

Podobné instrukce: BIT, BITB, BIS, BISB.

BIS

Jméno

BIS, BISB — Nastavování bitů v operandu

Přehled

BIS *src, dst*

BISB *src, dst*

V cílovém operandu jsou nastaveny bity, které jsou nastaveny v operandu zdrojovém. Zdrojový operand slouží jako bitová maska.

Popis

BIS	0 101 ssssss ddddddd
BICB	1 101 ssssss ddddddd

Odkazy

Podobné instrukce: BIC, BICB, BIT, BITB.

BIT

Jméno

BIT, BITB — Testování bitů v operandu

Přehled

BIT *src, dst*

BITB *src, dst*

Zdrojový a cílový operand jsou testovány logickým součinem (AND). Na základě výsledku jsou nastaveny podmínkové bity.

N	Z	V	C
*	*	0	-

Popis

BIT	0 011 ssssss ddddd
BITB	1 011 ssssss ddddd

Odkazy

Podobné instrukce: BIC, BICB, BIS, BISB, CMP, CMPB, TST, TSTB.

CLR

Jméno

CLR, CLRB — Clear Operand

Přehled

CLR *dst*

CLRB *dst*

Všechny bity operandu jsou nastaveny na "0".

Popis

CLR	0 000 101 000 ddddd
CLRB	1 000 101 000 ddddd

Odkazy

Podobné instrukce: BIC, BICB

CMP

Jméno

`CMP`, `CMPB` — Zdrojový a cílový operand jsou porovnány a podle výsledku jsou nastaveny příznaky N, Z, V a C

Přehled

`CMP src, dst`

`CMPB src, dst`

Je spočten rozdíl `src-dst` a podle výsledku jsou nastaveny podmínkové bity.

N	Z	V	C
*	*	*	*

Popis

CMP	0 010 ssssss ddddd
CMPB	1 010 ssssss ddddd

Ukázky

Odkazy

Podobné instrukce: SUB, BIT, BITB, TST, TSTB.

COM

Jméno

COM, COMB — Logický doplněk (*Complement*)

Přehled

COM *dst*

COMB *dst*

Všechny bity operandu jsou nahrazeny bity s opačnou hodnotou. Logická operace NOT.

Popis

COM	0 000 101 001 dddddd
COMB	1 000 101 001 dddddd

Odkazy

Podobné instrukce: CLR, CLRB, COM, COMB.

DEC

Jméno

DEC, DECB — Zmenšení cílového operandu o 1

Přehled

DEC *dst*

DECB *dst*

Popis

DEC	0 000 101 011 dddddd
-----	----------------------

DECB	1 000 101 011 dddddd
------	----------------------

Ukázky

Odkazy

Podobné instrukce: INC, INCB.

DIV

Jméno

DIV — Dělení 32 bitového čísla 16-ti bitovým

Přehled

DIV *src*, *R*

32 bitové čísl v R_n a R_{n+1} je děleno zdrojovým operandem. Výsledek je uložen v R_n a zbytek po dělení v R_{n+1} .

N	Z	V	C
*	*	0	*

Popis

DIV	0 111 001 rrr ssssss
-----	----------------------

Ukázky

```
MOV #20001,R1    ; Define low order 16 bits
STX R0           ; Rozšiř znaménko do dalších 16-ti bitů
DIV #2,R0        ; Dělení 16-bitů 2 do R0,R1
```

Odkazy

Podobné instrukce: MUL.

HALT

Jméno

HALT — Zastavit počítač

Přehled

HALT

Instrukce zastaví počítač.

Popis

HALT	0 000 000 000 000 000
------	-----------------------

Odkazy

pdp11.isa.wait;

INC

Jméno

INC, INCB — Zvětšení cílového operandu o 1

Přehled

INC *dst*

INCB *dst*

Popis

INC	0 000 101 010 dddddd
INCB	1 000 101 010 dddddd

Ukázky

Odkazy

Podobné instrukce: DEC, DECB.

MOV

Jméno

MOV, MOVB — Cílový operand bude přepsán zdrojovým operandem

Přehled

MOV *src, dst*

MOVB *src, dst*

U instrukce MOVB, je-li cílový operand registr, dochází k rozšíření znaménkového bitu. Tzv. *Sign Extension*.

N	Z	V	C
*	*	0	-

Popis

MOV	0 001 ssssss dddddd
MOVB	1 001 ssssss dddddd

MUL

Jméno

MUL — 16-ti bitové násobení s 32-dvou bitovým nebo 16-ti bitovým výsledkem

Přehled

MUL *src*, *R*

Pokud je číslo registru sudé, ukládá se 32 bitový výsledek do registru *Rn* a *Rn+1*. Pokud je číslo registru liché, ukládá se do něj jen spodních 16 bitů výsledku.

N	Z	V	C
*	*	0	*

Popis

MUL	0 111 000 rrr ssssss
-----	----------------------

Ukázky

```
CLC                ; Clear carry condition code
MOV #400,R1
MUL #10,R1         ; 16-bit product (R is odd), R0 untouched
BCS ERROR          ; Result to low or to big for 16 bits.
```

Odkazy

Podobné instrukce: DIV.

NEG

Jméno

NEG, NEGB — Dvojkový doplněk (*Negate*)

Přehled

NEG *dst*

NEGB *dst*

Operand bude nahrazen negativní hodnotou. T.j. 2 je nahrazeno -2, Aritmetická operace -.

Popis

NEG	0 000 101 100 dddddd
NEGB	1 000 101 100 dddddd

Odkazy

Podobné instrukce: CLR, CLRB, COM, COMB.

ROL

Jméno

ROL, ROLB — Rotace doleva přes C

Přehled

ROL *dst*

ROLB *dst*

Všechny bity cílového operandu jsou posunuty o jedno místo doleva. Na nejnižší místo se posune C a tento je nahrazen obsahem původně nejvyššího (nejlevějšího) bitu operandu.

N	Z	V	C
*	*	*	*

Popis

ROL	0 000 110 001 dddddd
ROLB	1 000 110 001 dddddd

Odkazy

Podobné instrukce: ASL, ASLB, ROR, RORB, SWAB.

ROR

Jméno

ROR, RORB — Rotace doprava přes C

Přehled

ROR *dst*

RORB *dst*

Všechny bity cílového operandu jsou posunuty o jedno místo doprava. Na nejvyšší místo se posune obsah C a tento je nahrazen obsahem původně nějnižšího (nejpravějšího) bitu operandu.

N	Z	V	C
*	*	*	*

Popis

ROR	0 000 110 000 ddddd
RORB	1 000 110 000 ddddd

Odkazy

Podobné instrukce: ASR, ASRB, ROL, ROLB, SWAB.

SUB

Jméno

SUB — Hodnota zdrojového operandu je odečtena od operandu cílového

Přehled

SUB *src, dst*

N	Z	V	C
*	*	*	*

Popis

SUB	1 110 ssssss ddddddd
-----	----------------------

Odkazy

Podobné instrukce: ADD, CMP, CMPB.

SWAB

Jméno

SWAB — V cílovém operandu je vyměněn obsah spodního a horního bajtu

Přehled

SWAB *dst*

N	Z	V	C
*	*	0	0

Popis

SWAB	0 000 000 011 ddddddd
------	-----------------------

Odkazy

Podobné instrukce: ASL, ASLB, ASR, ASRB, ROL, ROLB, ROR, RORB.

TST

Jméno

TST, TSTB — Na základě cílového operandu jsou nastaveny příznaky N a Z

Přehled

TST *dst*

TSTB *dst*

Operand je přečten a podle jeho hodnoty jsou nastaveny příznak N a Z.

Popis

TST	0 000 101 111 ddddd
TSTB	1 000 101 111 ddddd

Odkazy

Podobné instrukce: BIT, BITB, CMP, CMPB.

XOR

Jméno

XOR — Logický exkluzivní součet

Přehled

XOR *R, dst*

N	Z	V	C
*	*	0	-

Popis

XOR	0 111 100 rrr dddddd
-----	----------------------

Ukázky

XOR R0, R2 ; R0 xor R2 → R2

Odkazy

Podobné instrukce: FIXME: pdp11.isa.*;.

77.6. Elliott Bros (London) Ltd.

Odkazy:

-
-
-

Počítače:

- Elliott 152²¹
-
-

* Informace z: *Our Computer Heritage, Pilot Study*²²

Tabulka 77-13. Počítače Elliot a data spuštění

počítač	datum spuštění
Elliot 150 series	
Elliot 152	1950
Elliot Nicholas	1952
Elliot 153	1954
Elliot/GCHQ OEDIPUS (311)	1954
Elliot 400 series	
Elliot/NRDC 401	1953
Elliot 402	1955
Elliot 403 (WREDAC)	1956
Elliot 405	1956

počítač	datum spuštění
Elliot 800 series	
Elliot 802	1958
Elliot 803	1959
Elliot ARCH 1000, etc.	1962
Elliot 500 series	
Elliot 503	1963
Elliot 502	1964
Elliot 900 series	
Elliot 900 series	1963
Elliot 4100 series	
Elliot 4100 series	1966

77.6.1. Rodina počítačů Elliott 15x

The Admiralty's Medium Range System 5 (MRS5).

Tyto počítače byly sestavované pro Britskou admiraltu. Jejich původní účení byla střelba řízená radarem. K počítači nikdy nebylo připojeno dělo.

Vlastnosti počítače Elliott 152:

- Přenos dat uvnitř počítače: sériový
- ALU: rychlá násobička
- 40-ti bitový akumulátor
- Šířka datového slova 16 bitů
- Šířka instrukčního slova 20 bitů
- ALU založeno na subminiaturních pentodách
- Paměť: 64 slov (CRT - elektrostatická RAM)
- Paměť adresovaná jako dvě nezávislé banky, každá 32 slov.
- Main memory: 1024 slov 32 bitů každé. Paměť založena na 64 long delay lines
-

Instrukční slovo, 20 bitů široké, sestávalo ze dvou částí. 6-ti bitového instrukčního kódu ???

77.6.2. Elliott 40x

Počítač vnitřním sériovým přenosem. Délka slova 34 bitů, využitých bylo jen 32 které byly přístupny programátorovi.

Instrukční slovo dlouhé 32 bitů sestávalo z řady polí.

77.6.3. Elliott 800 series

Rodina tranzistorových počítačů menší až střední velikosti, vnitřní přenosy dat a zpracování sériové. Tato rodina byla projektována jako univerzální počítač a také pro online řízení procesů. Rodina sestávala s těchto typů:

801

Prototyp postavený v továrně. Byl postaven jen jeden prototyp okolo roku 1957.

802

Produkční varianta 801. Byl poprvé expedován v listopadu 1958. 33-bitový se sériovou aritmetikou. Používal paměť z feritových jader. Převážně tranzistorový ale používal elektronky v obvodech feritové paměti a některých částech logiky. V letech 1958 až 1961 bylo postaveno a expedováno sedm počítačů Elliott 802.

803A

Plně tranzistorová verze 802. Délka slova byla zvětšena na 39 bitů. Poprvé uveden do provozu v roce 1959, první dodávky zákazníkům někdy v roce 1960. Používal stejnou sériovou architekturu jako 802. Větší délka slova způsobovala díky sériové architektuře nižší rychlost než měl model 802.

803B

Vylepšená verze 803A. Poprvé spuštěn v roce 1960. V počítači bylo více paralelních cest a měl hardwarovou FPU. Prodejní cena v roce 1960 byla £29,000. Počítač byl jednoduchý a kompletně tranzistorový. Příkon byl pouze 3.6KW. Počítačů 803A a 803B bylo dohromady vyrobeno 211 kusů.

804

Návrh nikdy neopustil projekční kancelář.

805

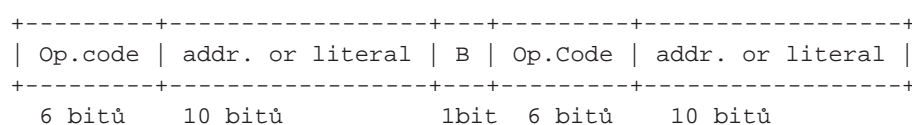
Návrh nikdy neopustil projekční kancelář.

Vlastnosti:

- Délka slova 33-39 bitů
- Dvě instrukce v jednom slově
- Schopný adresovat 8K slov.
-

Instrukční slovo modelu 802 má délku 33 bitů a obsahuje dvě 16-ti bitové instrukce. Mezi těmito instrukcemi se nachází bit B-line. Celé instrukční slovo sestává z několika polí.

Obrázek 77-10. Instrukční slovo počítače Elliott 802



Operační kódy jsou rozděleny do 8 skupin po 8 kódech.

Instrukční slovo modelu 803 má délku 39 bitů a obsahuje dvě 19-ti bitové instrukce. Vypadá obdobně jako instrukční slovo modelu 802 s tím rozdílem, že adresní pole byla rozšířena z 10 bitů na 13 bitů.

77.6.4. Elliott 500

Model 503 je následník modelu 803. 39-ti bitové slovo.

Model 502 je následník modelu 402. 20-ti bitové slovo. Navrhovaný pro letectvo.

77.6.5. Elliott 900 series

18-ti bitová architektura. Některé modely jsou zkráceny na 13 a 12 bitů.

Tabulka 77-14.

901	18-bit	
920A	18-bit	18-bit 8192-word store
920B	18-bit	18-bit 8192-word store
903	18-bit	
ARCH 9000	18-bit	
920M	18-bit	
920C	18-bit	
905	18-bit	
ARCH 9050	18-bit	
920ATC	18-bit	
MC1800	18-bit	
ARCH 102	13-bit	
902	12-bit	
102C	12-bit	
ARCH 105	12-bit	

Tabulka 77-15. Registry

Registr	šířka	Obsah
J	16	adresový register
M	18	memory buffer
I	4	function
A	18	accumulator
Q	18	pomocný registr

77.6.6. Elliott 4100

Architektura 24/48 bitů

77.7. English Electric Ltd.

77.7.1. KDF9

Vlastnosti:

- Hlavní paměť: 32768 slov každé 48 bitů dlouhé.
- 16 counter/index registrů, Q0 always zero
-
-

77.8. Ferranti

Tabulka 77-16.

model	rok	word	instr.	
Mark I	1951			
Mark Star	1953			
Mercury	1957	40	20	
Pegasus	1956	39	19	
Perseus	1959			
Sirius	1961			
Orion 1	1963			
Orion 2	1963			
Atlas 1 & 2	1962			
Apollo	1961	20	20	Air Traffic Control
Poseidon	1962?			RT data processing of Royal Navy
Argus	1963?			
Hermes				RT data processing of Royal Navy
F1600				binary compatible with Hermes
FM1600				binary compatible with Hermes

77.8.1. Ferranti Mercury

Vlastnosti:

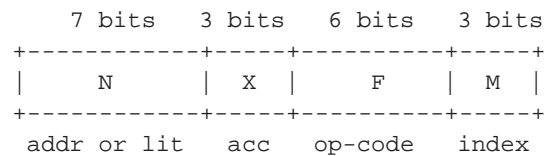
- high speed — 1MHz clock cycle
- hardware floating point arithmetic, add 180~~textmus~~, multiply 300~~textmus~~, division 3.5ms
- exceptionally large storage capacity
- Computational Store: 1k 40bit words, magnetic cores
- Main Store: 4k 40bit words, magnetic drum, 4-8 drums
- word length: 40 bits, 1 parity bit per 10bits.
- Instruction length: 20 bits (10 bit instruction, 10 bit address)

77.8.2. Ferranti Pegasus

Vlastnosti:

- Word size: 39 bit.
- Instruction size: 19 bit

Obrázek 77-11. Ferranti Pegasus Instruction Fields



Tabulka 77-17. Ferranti Pagasus Memory Map

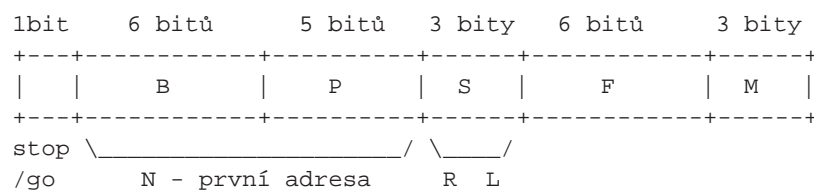
addr		description
0-7	0-7	Accumulators X0-X7, X0 always zero
8-14		nepoužito, 0
15	15	ruční přepínače (20 bits)
16	16	i/o 5bitů, checked
17	17	i/o 5bitů, unchecked
18-31		nepoužité, 0
32	32	konstanta -1.0
33	33	konstanta 1/2
34	34	konstanta 2^{-10}
35	35	konstanta 2^{-13}
36-63		nepoužito, 0
64-111	0.0 - 5.7	48 words program/data jako 6 bloků po 8 slovech
112-127		nepoužito, 0

77.8.3. Ferranti Perseus

Vlastnosti:

- Instrukce 24 bitů
- Slovo 72 bitů obsahuje 3 instrukce

Obrázek 77-12. Ferranti Perseus Instruction Fields



77.8.4. Ferranti Sirius

Vlastnosti:

- Word: 40 bits, 10 BCD číslic
-
-
-

Obrázek 77-13. Instrukční slovo



77.8.5. Ferranti Orion 1 a 2

Odkazy:

- Malcolm Bigg²³
-

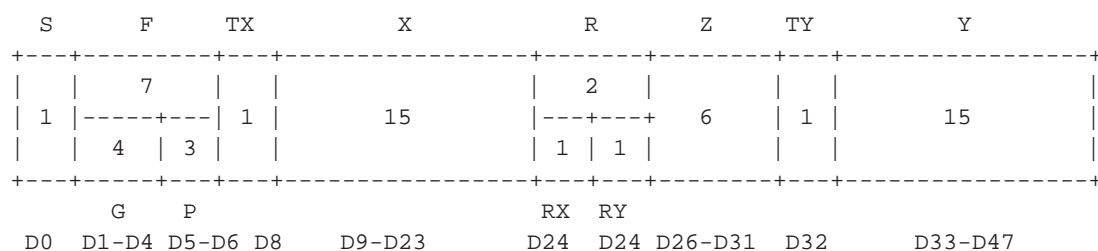
This was the first design of a system which used ballot box logic based on magnetic principles in the form of circuit packages called „neurons“ proposed by Gordon Scarrott then Ferranti's Chief Engineer.

Vlastnosti:

- Word length 48 bits
-

Protože se vývoj logických prvků pro Orion 1 opožďoval, postavila firma Orion 2 který měl stejnou architekturu a strukturu, jen byl postaven na dostupné technologii DTL.

Obrázek 77-14. Farranti Orion Instruction Format



77.8.6. Ferranti Atlas 1 a 2

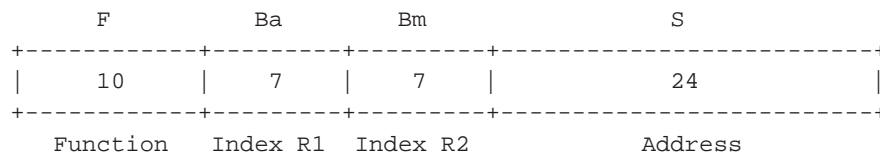
Celkem bylo postave 3 ks počítačů Atlas 1 a 3 ks počítačů Atlas 2.

Vlastnosti:

- FPU 48bits word

- 128 index registers, most of them 24 bits wide.
- Atlas 1 minimum main memory: 114,688 words (48 bits wide)
- Atlas 1 core memory: 16,384 words (48bit)

Obrázek 77-15. Ferranti Atlas Instruction Format



Počítač má 128 indexových registrů. Tyto jsou 24 bitové. Výjimku tvoří registry B120 až B127 jenž jsou speciální a mají šířku 48 bitů.

Tabulka 77-18. Ferranti Atlas Index Registers

B0		konstanta 0
B120		
B121		
B122		
B123		
B124		
B125		
B126		
B127		lokální čas

77.8.7. Ferranti Apollo

Vlastnosti:

- Word: 20 bits

Instrukce jsou dlouhé 20 bitů a sestávají ze tří polí. Operačního kódu v délce 6 bitů, modifikátoru v délce 2 bity a 12 bitů dlouhé adresy. Přímou bylo možno adresovat 4096 slov paměti z feritových jader.

77.8.8. Ferranti Poseidon

Počítač byl vytvářen pro Royal Navy.

Vlastnosti:

- Instrukce 24 bitů dlouhé
-

.

77.9. GRI Computer Corporation

Odkazy:

- .
- .

Fakta:

- Založil Saul Dinman poté co opustil DEC.
- Firma vyráběla počítače GRI 909, GRI 99 a GRI 919.
-

77.10. Honeywell

Odkazy:

- Honeywell²⁴
-
-

77.10.1. Honeywell Series 16

Odkazy:

- Honeywell 316²⁵
- Honeywell Series 16²⁶
- SimX16 -- Honeywell DDP-516 Minicomputer Simulation²⁷
-
-
-

77.11. IBM (International Business Machines Corporation)

*

Odkazy:

- History of IBM²⁸ na Wikipedii
-
-
-

Tabulka 77-19. Počítače IBM

rok	model	technologie	poznámka
1934	IBM 801		
1935	IBM 805		
1944	ASCC	elektronky	
1946	IBM 603	elektronky	
1946	IBM SSEC	elektronky	
1952, 1953-04-07	IBM 701 (Defense Calculator)	elektronky	36/18 bit
1953	IBM 702	elektronky	36/18 bit, komerční
1953	IBM 650		
1954	NORC (Naval Ordnance Research Computer)		
1954	IBM 704	elektronky	36bit, vědecký
1954	IBM 705	elektronky	36bit, komerční
1958	IBM 709	elektronky	36bit, vědecký
1958	SAGE (Semi-Automatic Ground Environment)	elektronky	
1959	IBM 1401		
1959	IBM 1403		
1959	IBM 7090		vědecký
1960	IBM 7070		dekadický
1961	IBM 7030 Stretch		superpočítač
1961	IBM 7074		dekadický
1961	IBM 7080		komerční
<=1962	IBM 1620		dekadický
1962	IBM 7010		high end verze IBM 1410
1962	IBM 7072		dekadický
1962	IBM 7094		vědecký
1963	IBM 7040		vědecký
1963	IBM 7044		vědecký
1964	IBM 7094 II		vědecký
1964-03-07	System/360		
1965	guidance computer for Gemini		
1966	System/4Pi		
1970	System/370		
1975	IBM 5100 Portable computer		
1988	AS/400		
1990	System/390		
	IBM 7090	tranzistory	36bit

rok	model	technologie	poznámka
	IBM 7094	tranzistory	36bit
	IBM 700/7000 series		
	System/3		
	System/38		
	IBM Series/1		
	PowerPC		
	RS/6000		
	zSeries		
	Cell processor		

77.11.1. AN/FSQ-7 (IBM)

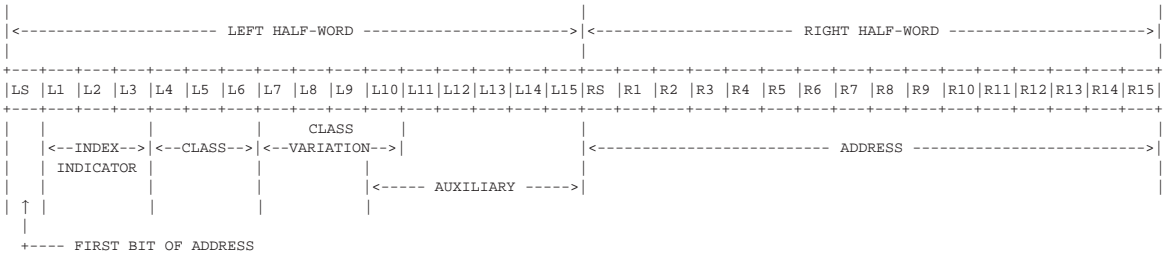
Odkazy:

- AN/FSQ-7²⁹ na Wikipedii
- AN/FSQ-7 "Whirlwind II" Intercept Computer³⁰
-
-

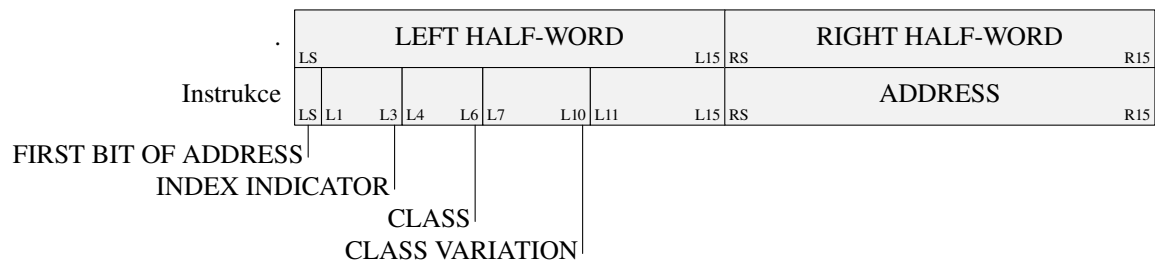
Fakta:

- Šířka slova je 32 bitů. Do paměti se ukládá 33, poslední bit je paritní.
- Jedno 32 bitové slovo sestává ze dvou 16-ti bitových půlslov.
- Používá 17-ti bitovou adresu, (AN/FSQ-8 16-ti bitovou)
-
-
-
-
-

Obrázek 77-16. Instrukční slovo AN/FSQ-7



Obrázek 77-17. Instrukční slovo AN/FSQ-7



Tabulka 77-20. Summary of AN/FSQ-8 instructions

Instruction Name	Mnemo. Name	Octal Code	Exec. Time	Indexable	Cause Overflow
Reset Index Register from Right Accumulator	XAC	764	6 μ s	No	No
Store Address	STA	340	18 μ s	Yes	No
Add Index Register	ADX	770	6 μ s	No	No
Clear and Add Magnitudes	CAM	160	12 μ s	Yes	No
Difference Magnitudes	DIM	164	12 μ s	Yes	No
Add B Registers	ADB	114	12 μ s	No	Yes
Multiply	MUL	250	17 \pm 0.5 μ s	Yes	No
Twin and Multiply	TMU	254	17 \pm 0.5 μ s	Yes	No
Divide	DVD	260	51.5 \pm 0.5 μ s	Yes	No
Twin and Divide	TDV	264	51.5 \pm 0.5 μ s	Yes	No
Shift Left and Round	SLR	024	Variable	No	Yes
Dual Shift Left	DSL	400	Variable	No	No
Dual Shift Right	DSR	404	Variable	No	No
Left Element Shift Right	LSR	440	Variable	No	No
Right Element Shift Right	RSR	444	Variable	No	No
Accumulators Shift Left	ASL	420	Variable	No	No
Accumulators Shift Right	ASR	424	Variable	No	No
Dual Cycle Left	DCL	460	Variable	No	No
Full Cycle Left	FCL	470	Variable	No	No
Extract	ETR	004	12 μ s	Yes	No
		000	μ s		

Inspirován architekturou Whirlwind II

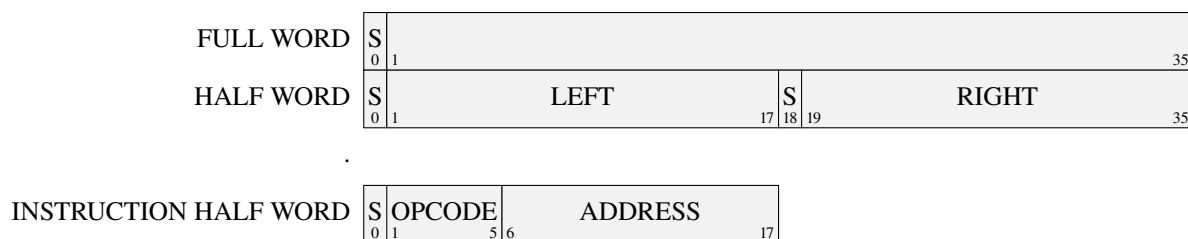
Součást SAGE

MIT and IBM developed the IBM AN/FSQ-7 computer to run the SAGE centers.

77.11.2. IBM 701

*

Obrázek 77-18. Formát dat a instrukcí IBM 701



Tabulka 77-21. Instrukce IBM 701

OCT	DEC	instrukce
00	00	STOP
01	01	TR
02	02	TR OV
03	03	TR +
04	04	TR 0
05	05	SUB
06	06	R SUB
07	07	SUB ABS
10	08	NO OP
11	09	ADD
12	10	R ADD
13	11	ADD ABS
14	12	STORE
15	13	STORE A
16	14	STORE MQ
17	15	LOAD MQ
20	16	MPY
21	17	MPY R
22	18	DIV
23	19	ROUND
24	20	L LEFT
25	21	L RIGHT
26	22	ACC LT
27	23	ACC RT
30	24	READ
31	25	READ B
32	26	WRITE
33	27	WR EOF
34	28	REWIND
35	29	SET DR
36	30	SENSE
37	31	COPY

77.11.3. IBM 704

*

Obrázek 77-19. Formát dat a instrukcí IBM 704

TYPE A INSTRUCTION	Prefix S 2 3	Decrement 17 18 20 21	Tag 18 20 21	Address 35
TYPE B INSTRUCTION	Operation S 11 12	Not Used 17 18 20 21	Tag 18 20 21	Address 35
INTEGER NUMBER	S S 1	Magnitude 35		
FLOAT POINT NUMBER	S S 1	Characteristic 8 9	Fraction 35	

77.11.4. IBM 1130

*

Fakta:

- Kapacita paměti 4, 8, 16 nebo 32K 16-ti bitových slov.
-
-
-

Obrázek 77-20. Formát dat a instrukcí IBM 1130

Single Precision Data Word Format	S 0 1	15													
Double Precision Data Word Format	S 0 1	15 0													
Short Instruction Format	OP 0	F 4	T 5	Displacement 6 7 8 15											
Long Instruction Format	OP 0	F 4	T 5	IA 6	Modifier Bits 7 8 9 15	Address 0 15									

77.11.5. IBM 1620

Jedná se o velmi starou konstrukci. Počítač pracuje s dekadickými číslicemi. Každá číslice je uložena v šestici bitů. Možné kombinace jsou pevně dány. Poslední čtyři bity, označené 8,4,2,1 kódují jednu dekadickou číslici. Bit F má zvláštní význam a označuje poslední číslici v řadě, záporné číslo, případně má jiný zvláštní význam. Bit C je Check, tedy kontrolní bit. V dnešních termínech je to paritní bit.

Jsou přípustné jenom takové kombinace bitů, které reprezentují číslice 0 až 9. V tabulce jsou uvedeny i další kombinace, které mají speciální význam.

Paměť počítače je adresována po jednotlivých dekadických číslicích, tedy po šesticích bitů. Její velikost je rovněž dekadická a má velikost od 20,000 do 60,000 číslic.

Tabulka 77-22. Kombinace bitů v paměťové buňce IBM 1620

	C	F	8	4	2	1
--	---	---	---	---	---	---

	C	F	8	4	2	1
0	X					
1						X
2					X	
3	X				X	X
4				X		
5	X			X		X
6	X			X	X	
7				X	X	X
8			X			
9	X		X			X
Record Mark	X		X		X	
Group Mark	X		X	X	X	X
Numerical Blank	X		X	X		

Obrázek 77-21. Struktura paměťové buňky IBM 1620

```

+---+---+---+---+---+---+
| C | F | 8 | 4 | 2 | 1 |
+---+---+---+---+---+---+

```

77.11.5.1. Instrukční sada

Instrukční slovo sestává z 12 dekadických číslic.

Obrázek 77-22. Formát instrukčního slova

```

+---+---+---+---+---+---+---+---+---+---+---+---+
| O0 | O1 | P2 | P3 | P4 | P5 | P6 | Q7 | Q8 | Q9 | Q10 | Q11 |
+---+---+---+---+---+---+---+---+---+---+---+---+
| OP | P | Q |
| <-----> | <----- ADDRESS -----> | <----- ADDRESS-----> |
| CODE | | |

```

77.11.6. IBM 1400 serie

*

Odkazy:

- IBM 1400 series³¹ na Wikipedii
- IBM 1410³² na Wikipedii
-
-
-

Rodina počítačů s dekadickou aritmetikou a proměnlivou délkou slova. Počítače byly vyráběné z tranzistorů. Jedná se o druhou generaci počítačů.

Prvním modelem v této rodině byl IBM 1401. Další počítače jsou s ním tu více tu méně kompatibilní.

Tabulka 77-23. IBM 1400 series

rok	model	poznámka
1959-10-05	IBM 1401	výroba ukončena 1971-02-08
1960-09-12	IBM 1410	mnohem rychlejší než 1401, více paměti, registrů, ... měl přepínač kompatibility s 1401
1962	IBM 1420	bankovníctví
1962	IBM 1440	levná alternativa k 1401, není plně kompatibilní
1962	IBM 7010	mainframe verze 1410, rychlejší
1963	IBM 1240	bankovní systém
1963	IBM 1460	
1968	IBM 1450	databázový systém pro malé banky

Obrázek 77-23. Slovo IBM 1401

Byte

C	B	A	8	4	2	1	M
0	1	2	3	4	5	6	7

Programovací jazyky a prostředí pro rodinu 1400 obashuje Symbolic Programming System (SPS, assembler), Autocoder (assembler), COBOL, FORTRAN, Report Program Generator (RPG) a FARGO.

77.11.7. IBM 7070

*

Odkazy:

- IBM 7070³³ na Wikipedii
-
-
-

Tranzistorová náhrada za elektronkový IBM 650.

Tabulka 77-24.

rok	model
	IBM 7070
1961-11	IBM 7074
1962-11	IBM 7072

77.11.8. IBM 4020

*

Odkazy:

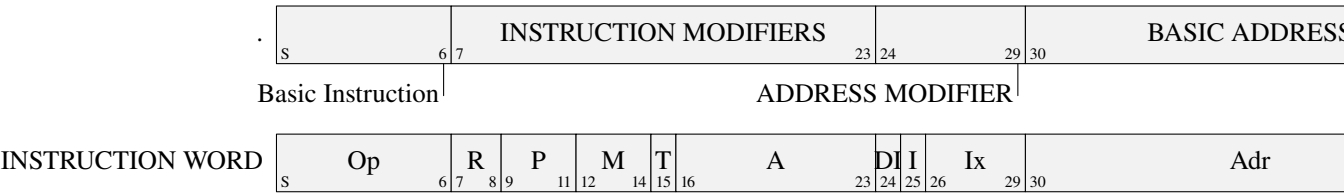
-
-

Vojenský počítač.

Fakta:

- Rychlost: do 400 000 instrukcí za sekundu
- Délka cyklu: 2.5μs
- Velikost slova: 48 bitů a dva paritní
- Velikost paměti: do 131 072 slov
-

Obrázek 77-24. Instrukční slovo IBM 4020



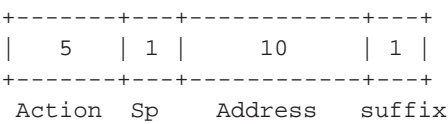
77.12. LEO Computers Ltd.

Odkazy:

- LEO Computers³⁴

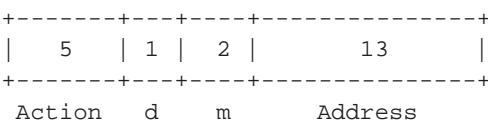
77.12.1. LEO I

Obrázek 77-25. LEO I Instruction Format



77.12.2. LEO III

Obrázek 77-26. LEO III Instruction Format



77.13. Norsk Data

*

Odkazy:

- Norsk Data³⁵ na Citizendii
- NORD-1 na Citizendii³⁶
-
-

Modely počítačů

- NORD-1 — první minipočítač vytvořený ve firmě. Počítač má 16-ti bitovou architekturu a první kus byl vyroben v roce 1967.
- NORD-10
- NORD-100 — 16-ti bitový univerzální jednodeskový počítač (*single board computer*). Schopný adresovat 64K slov (128KB), s MMU 16M slov (32MB) operační paměti. Je kompletně softwarově kompatibilní s NORD-10/S.
- ND-500 — 32 bitový systém
-
-
-

77.13.1. NORD-10

*

Obrázek 77-27. NORD-10

Word format	16-bit NORD-10 memory word																
	15															0	
Byte Format	Even Address								Odd Address								
	15							8	7							0	
Standard Floating Point Format	+/-	Exponent										Mantisa					
	47	46									32	31					
Instruction Word	Op.Code										X	I	B	Displacement			
	15										11	10	9	8	7	0	

77.13.2. NORD-100

*

Procesor obsahuje 16 sad registrů, každá sada pro jednu ze 16-ti úrovní programů. Sada obsahuje 8 univerzálních registrů a 8 registrů jen pro vnitřní použití mikroprogramem. Celkem tedy obsahuje procesor 256 registrů.

Pracovní registry NORD-100

Status register

obsahuje 8 indikátorů (příznaků)

A register

Hlavní pracovní registr.

D register

Dozširující pracovní registr.

T

Dočasný registr.

L

Link registr obsahující návratovou adresu podprogramu.

X

Index registr.

B

Bázový registr.

P

Čítač programu. Ukazatel na aktuálně prováděnou instrukci.

77.14. Nuclear Data, Inc.

* *Attributy: id="Nuclear_Data"*

Odkazy:

- Dokumenty o počítačích firmy Nuclear Date, Inc.³⁷ na BitSavers.org
-

Firma Nuclear Data, Inc. začala vyvíjet speciální počítače v roce 1957.

77.14.1. ND812

Odkazy:

- PDP-8
- Bunker Ramo BR2412
-

*

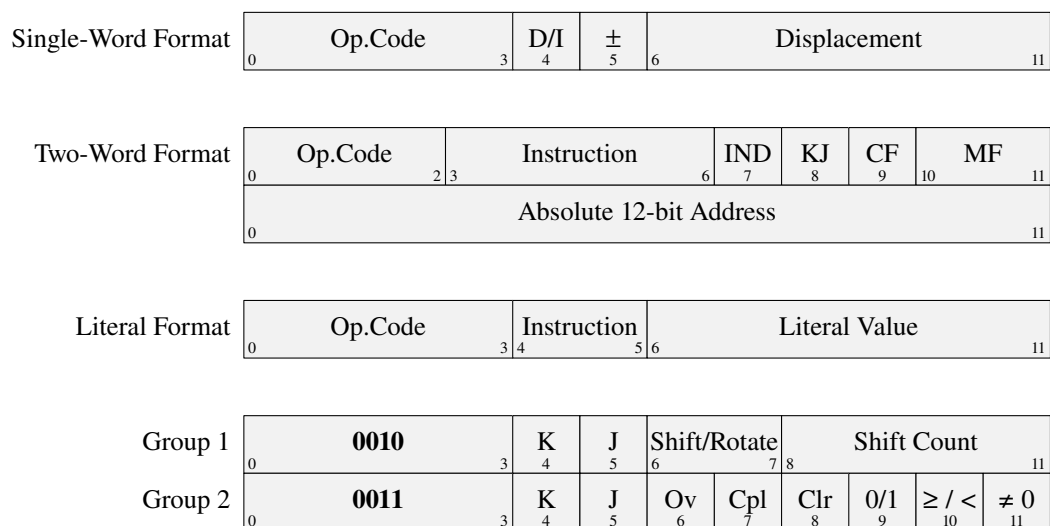
ALU obsahuje 4 dvanáctibitové registry (akumulátory): J, K, R, S

Paměťová jednotka obsahuje 8-16K 12-ti bitový slov. Jako paměťové médium slouží feritová jádra. Počítač je schopen přímo adresovat 16K slov paměti pomocí Two-Word instrukcí. Ve druhém slovu je spodních 12 bitů adresy a v prvním slovu v poli MF jsou dva nejvyšší bity adresy. Takto je získána celá absolutní 14-ti bitová adresa.

Počítač byl sestaven jako jednodeskový s pomocí TTL/DTL integrovaných obvodů.

Vyroběn v lednu roku 1971.

Obrázek 77-28. Formáty instrukcí



Tabulka 77-25. Single-Word instrukce

opcode	mnemo
00	<i>Two-Word Instruction</i>
04	<i>Two-Word Instruction</i>
10	Group 1
14	Group 2
20	SMJ
24	SMJ
30	DSZ
34	ISZ
40	SBJ
44	ADJ
50	LDJ
54	STJ
60	JMP
64	JPS
70	
74	

Obrázek 77-29. Status Word

Flag 0	OV 1	JPS 2 3		INT 4 5		IONL 6	IONA 7	IONB 8	IONH 9	Current Exec 10 11	
		MF0 2	MF1 3	MF0 4	MF1 5					MF0 10	MF1 11

77.14.1.1. Adresní módy

*

77.14.1.1.1. Auto Index

V každém paměťovém poli jsou vyhrazeny dvě adresy (slova). Tyto mají vlastnost, že adresovány nepřímo (*indirect*) instrukcí Single-Word, nejdříve zvětší (automaticky inkrementují) svůj obsah. Zakódování této operace se provádí nastavením bitů 6-11 (Displacement) na 0. Obsah bitu 5 (+/-) určuje které ze dvou autoinkrement slov se použije. Tyto autoincrement adresy jsou 0000 a 7777.

77.14.1.2. Instrukce po skupinách

*

Tabulka 77-26.

Op-Code	Mnemonic	Description	
BIN			OCT
0000	00		Two-Word Instruction Format
0001	04		
0010	10		
0011	14		
0100	20		Literal instruction format (ANDE, ANDL, ADDL, SUBL)
0101	24	skip if [J K]≠operand SMJ/SMK	
0110	30	DSZ	Dec(Mem[ea]); skip if =0
0111	34	ISZ	Inc(Mem[ea]); skip if =0
1000	40	SBJ/SBK	
1001	44	ADJ/ADK	

Op-Code	Mnemonic	popis	
BIN			OCT
1010	50	LDJ/LDK	[J K] → Mem[ea]
1011	54	STJ/STK	Mem[ea] → [J K]
1100	60	JMP	PC → ea
1101	64	JPS	Mem[ea] → PC; PC → Mem[ea] + 1
1110	70	XCT	execute out of order instruction Mem[ea]
1111	74	I/O	

77.14.1.2.1. Memory Reference Instructions

*

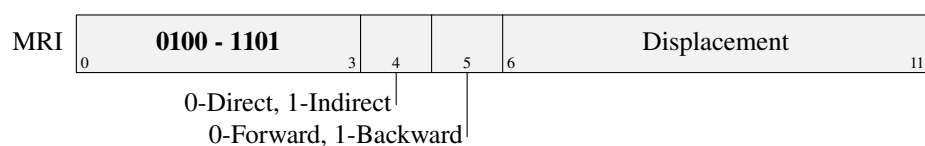
Odkazy:

- strana 4-
-

Tabulka 77-27. Single-Word Memory Reference Instructions

Mnemonic	Op-Code	popis
ANDF	2000	
SMJ	2400	Skip if J ≠ memory
DSZ	3000	
ISZ	3400	
SBJ	4000	
ADJ	4400	
LDJ	5000	
STJ	5400	
JMP	6000	
JPS	6400	
XCT	7000	Execute instruction n
I/O	7400	I/O instruction

Obrázek 77-30. Single-Word Memory Reference Instruction Format



Tabulka 77-28. Modifikace přístupu k paměti

operace	kód
Direct Forward	X0ZZ
Direct Backward	X1ZZ
Indirect Forward	X2ZZ
Indirect Backward	X3ZZ

77.14.1.2.2. Two-Word Memory Reference Instructions

*

Odkazy:

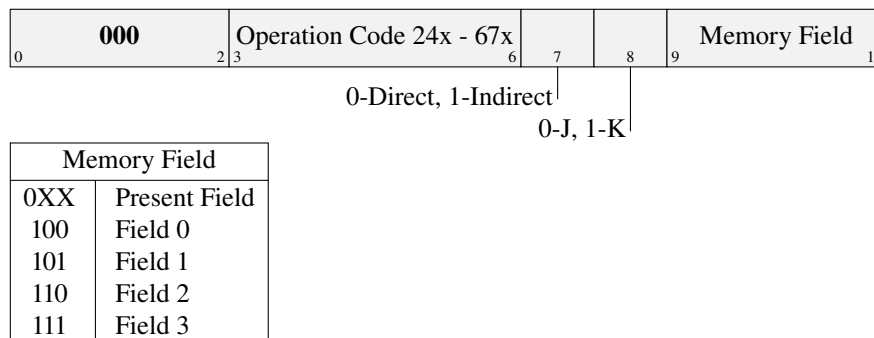
- ND821_MaintMan.pdf³⁸ strana 4-182
-

Tabulka 77-29. Two-Word Memory Referenc Instructions

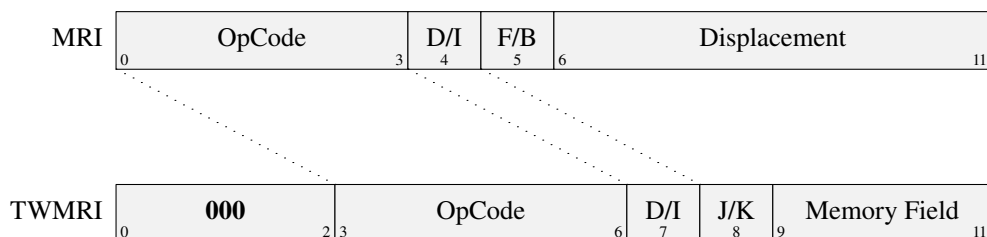
Mnemonic	Op-Code	popis
TWSMJ	0240	Skip if J \neq memory
TWSMK	0250	Skip if K \neq memory
TWDSZ	0300	Decrement memory and skip if =0
TWISZ	0340	Increment memory and skip if =0
TWSBJ	0400	Subtract memory from J
TWSBK	0410	Subtract memory from K
TWADJ	0440	Add memory to J
TWADK	0450	Add memory to K
TWLDJ	0500	Load memory into J
TWLDK	0510	Load memory into K
TWSTJ	0540	Store J to memory
TWSTK	0550	Store K in memory
TWJMP	0600	Jump unconditionally

MnemonicOp-Code	popis
TWJPS 0640	Jump to subroutine

Obrázek 77-31. Two-Word Memory Reference Instruction Format



Obrázek 77-32. Vztah mezi MRI a TWMRI instrukčním formátem



77.14.1.2.3. Group 1 Operate

*

Odkazy:

- ND821_Maint_Man.pdf³⁹ strana 4-62
-
- ROTD
- SFTZ

Skupina 1 instrukcí je rozpoznána podle charakteristické hodnoty 0010 v bitech 0-3 instrukce. Tyto instrukce se provádějí nad horním (J) a dolním (K) akumulátorem, podle hodnoty bitů 4 a 5 instrukčního slova.

Do této skupiny instrukcí patří podskupiny:

- MPY-DIV
- Logical AND
- Load / Exchange
- Add-Subtract
- Shift / Rotate
- Load / Reload

Obrázek 77-33. Formát instrukčního slova Group 1 Operate

Group 1	0010			Acc		Arithmetic	Neg.	SubAcc		Sub.
	0	3	4	5	6	7	8	9	10	11
Group 1	0010			K	J		Neg.	S	R	
	0	3	4	5	6	7	8	9	10	11
00-AND / 01-Arithmetic / 10-Shift / 11-Rotate							0-Add / 1-Sub			
AND						00				
	0	5	6	7	8	11				
Arithmetic						01				
	0	5	6	7	8	11				
Shift						10				
	0	5	6	7	8	11				
Rotate						11				
	0	5	6	7	8	11				
Multiply / Divide	0010			0	0	0	0	0		
	0	3	4	5	6	7	8	9	10	11
AND	0010			J/K	00	Mul/Div				
	0	3	4	5	6	7				

77.14.1.2.4. Rotace a posuvy (Group 1 Operate)

*

Odkazy:

- ND812_Prog.pdf⁴⁰ strana 4-9
- ND821_Maint_Man.pdf⁴¹ strana 4-62
-
- ROTD
- SFTZ

V jedné instrukci můžeme provést rotaci nebo posuv až o 15 bitových pozic. Počet pozic o které se posouvá je zakódován v instrukčním slově v bitech 8 až 11.

Obrázek 77-34. Formát instrukčního slova Shift/Rotate

	0	3	4	5	6	7	8	11
	0010			K	J	Shift/Rotate	Shift Count	
SFTZ	0	3	4	5	6	7	8	11
	0010			K	J	10	Shift Count	
ROTD	0	3	4	5	6	7	8	11
	0010			K	J	11	Shift Count	

77.14.1.2.5. Instrukce skupiny 2

*

Odkazy:

- ND812_Prog.pdf strana 3-8
-

-
- SET
- CLR
- INC
- SET

Skupina 2 instrukcí operate je identifikována hodnotou 0011 v bitech 0 až 3 instrukčního slova.

Skupina 2 kódovaných instrukcí slouží k nastavování, a testování hodnot v akumulátorech J a K a v příznaku O (Overflow).

Do této skupiny instrukcí patří podskupiny:

- Register Modify
- Register Sign Skip
- Register Zero Skip
- Status Modify
- Status Skip
- Cycle Delay (1400_{OCT})

Obrázek 77-35. Formáty instrukcí

Group 2	0011			K	J	Ov	Skip condition			
	0	3	4	5	6	7	8	9	11	
Func (01-Clear/10-Complement/11-Set)										
Register				K	J	O				
	0	3	4	5	6	7	11			
Operation							Oper			
	0	6	7	8	9	11				
Condition								0/1	≥/<	≠ 0
	0	3	4	8	9	10	11			
IDLE	0011			000			00		000	
	0	3	4	6	7	8	9	11		
INC	0011			K	J	1	00		100	
	0	3	4	5	6	7	8	9	11	
CLR	0011			K	J	Ov	01		000	
	0	3	4	5	6	7	8	9	11	
CMP	0011			K	J	Ov	10		000	
	0	3	4	5	6	7	8	9	11	
NEG	0011			K	J	Ov	10		100	
	0	3	4	5	6	7	8	9	11	
SET	0011			K	J	0	11		000	
	0	3	4	5	6	7	8	9	11	

Tabulka 77-30.

9	10	11	condition
0	0	0	
0	0	1	!=0 (non zero)

9	10	11	condition
0	1	0	<0 (positive)
0	1	1	!=0 AND >0 (non-zero and positive)
1	0	0	Increment
1	0	1	=0 (zero)
1	1	0	<0 (negative)
1	1	1	<=0 (zero or negative)

SET J 1530
 INC K 1604
 CLR K 1610
 CMP K 1620
 NEG K 1624
 SET K 1630
 CLR JK 1710
 CMP JK 1720
 SET JK 1730

SIP J 1502
 SIP K 1602
 SIP JK 1702
 SIN J 1506
 SIN K 1606
 SIN JK 1706

SNZ J 1501
 SNZ K 1601
 SNZ JK 1701

SIZ J 1505
 SIZ K 1605
 SIZ JK 1705

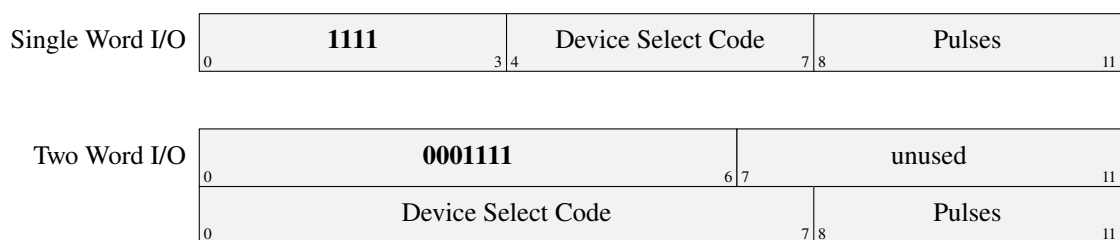
CLR 1410
 CLR O 1450
 CMP 1420 # Complement Flag Register
 CMP O 1460

SET 1430 # Set Flag Register to 1
 SET O 1470

77.14.1.2.6. I/O instrukce

*

Obrázek 77-36. Formáty instrukcí



Tabulka 77-31.

OpCode	Mnemonic
1003	IOFF
1004	IONH
1005	IONB
1006	IONA
1007	IONN
1440	SKPL
1500	PION
1600	PIOF

77.14.1.3. Abecední seznam instrukcí ND 812

*

Tabulka 77-32. Operační kódy instrukcí ND812 řazené podle operačního kódu

název instrukce	mnemo	kód	
		bin	octal
twdsz	TWDSZ	000 011 0i0 fff	0300
twisz		000 011 100 000	0340
twldj		000 101 000 000	0500
twldk		000 101 001 000	0510
twio	TWIO	000 111 100 000	0740
sftz		001 001 100 000	1140
rotd		001 001 110 000	1160
sftz		001 010 100 000	1240
rotd		001 010 110 000	1260
sftz		001 011 100 000	1340
rotd		001 011 110 000	1360
idle	IDLE	001 100 000 000	1400
clr		001 100 001 000	1410
neg		001 100 001 000	1410

název instrukce	mnemo	kód	
		bin	octal
neg		001 100 010 000	1420
set		001 100 011 000	1430
clr		001 100 101 000	1450
neg		001 100 101 000	1450
set		001 100 111 000	1470
clr		001 101 001 000	1510
neg		001 101 010 100	1524
set		001 101 011 000	1530
inc		001 101 100 000	1540
clr		001 101 101 000	1550
neg		001 101 101 000	1550
clr		001 110 001 000	1610
neg		001 110 010 100	1624
set		001 110 011 000	1630
inc		001 110 100 000	1640
clr		001 110 101 000	1650
neg		001 110 101 000	1650
clr		001 111 001 000	1710
neg		001 111 010 100	1724
set		001 111 011 000	1730
inc		001 111 100 000	1740
clr		001 111 101 000	1750
neg		001 111 101 000	1750
andf		010 000 xxx xxx	2000
dsz	DSZ	011 0i0 ddd ddd	3000
isz		011 100 000 000	3400
ldj		101 000 000 000	5000
tif	TIF	111 100 000 001	7401
tir	TIR	111 100 000 010	7402
trf	TRF	111 100 000 011	7403
tis	TIS	111 100 000 100	7404
toc	TOC	111 100 001 001	7411
top	TOP	111 100 001 010	7412
tcp	TCP	111 100 001 011	7413
tos	TOS	111 100 001 100	7414
hif	HIF	111 100 010 001	7421
hir	HIR	111 100 010 010	7422
his	HIS	111 100 010 100	7424

ANDF

Jméno

ANDF — (POPIS)

Přehled

ANDF

(**TVAR INSTRUKCE**) *src, dst*

Poznámky k tvaru instrukce

Popis

mnemo	kód	
	binary	octal
ANDF	010 000 xxx xxx	2000

Odkazy

Podobné instrukce: ANDF.

- ND821_MaintMan.pdf strana 4-242
-

CLR

Jméno

CLR — Clear register

Přehled

CLR

CLR J

CLR K

CLR JK

CLR O

Poznámky k tvaru instrukce

L
.

Popis

mnemo	kód	
	binary	octal
CLR	001 100 001 000	1410
CLR O	001 100 101 000	1450
CLR J	001 101 001 000	1510
CLR JO	001 101 101 000	1550
CLR K	001 110 001 000	1610
CLR KO	001 110 101 000	1650
CLR JK	001 111 001 000	1710
CLR JKO	001 111 101 000	1750

Obrázek 77-1. Kódování instrukcí CLR

0011	K	J	O	Cpl	1	0/1	≥/ <	≠ 0	
0	3	4	5	6	7	8	9	10	11

Odkazy

Podobné instrukce: nd812.isa.hlt;.

-
-

DSZ

Jméno

DSZ , TWDSZ — Decrement memory and skip next instruction if zero

Přehled

DSZ *op*

DSZ I *op*

TWDSZ *op*

TWDSZ I *op*

J	K	R	S	Ov
X	X	-	-	X

Popis

Overflow not possible.

mnemo	kód	
	binary	octal
CLR JKO	011 0i0 ddd ddd	3000
CLR JKO	000 011 0i0 fff	0300

Odkazy

Podobné instrukce: ISZ.

-
-

HIF

Jméno

HIF — HS Reader Fetch

Přehled

HIF

Popis

mnemo	kód	
	binary	octal
CLR JKO	111 100 010 001	7421

Odkazy

Podobné instrukce: HIR, HIS.

•
•

HIR

Jméno

HIR — Clear Flag; Read HS Buffer

Přehled

HIR

Popis

mnemo	kód	
	binary	octal
CLR JKO	111 100 010 010	7422

Odkazy

Podobné instrukce: HIS.

•
•

HIS

Jméno

HIS — Skip HS Reader Ready

Přehled

HIS

Popis

mnemo	kód	
	binary	octal
CLR JKO	111 100 010 100	7424

Odkazy

Podobné instrukce: HIR.

•
•

IDLE

Jméno

IDLE — One cycle delay

Přehled

IDLE

Popis

mnemo	kód	
	binary	octal
CLR JKO	001 100 000 000	1400

Odkazy

Podobné instrukce: IDLE.

•
•

INC acc

Jméno

INC acc — Increment accumulator

Přehled

INC J

INC K

INC JK

J	K	R	S	Ov
X	X	-	-	X

Popis

mnemo	kód	
	binary	octal
INC	001 101 100 000	1540
INC	001 110 100 000	1640
INC	001 111 100 000	1740

Odkazy

Podobné instrukce: nd812.isa.hlt;.

•
•

ISZ

Jméno

ISZ, TWISZ — Increment contents of effective address by one and skip next instruction if zero

Přehled

ISZ

TWISZ *dst*

Poznámky k tvaru instrukce

L
.

Popis

Overflow not possible.

mnemo	kód	
	binary	octal
ISZ	011 100 000 000	3400
TWISZ	000 011 100 000	0340

Odkazy

Podobné instrukce: DSZ.

•
•

LDJ

Jméno

LDJ, TWLDJ, TWLDK — Loads register J/K

Přehled

LDJ

TWLDJ

TWLDK

(TVAR INSTRUKCE) *src, dst*

Poznámky k tvaru instrukce

L
.

Popis

mnemo	kód	
	binary	octal
LDJ	101 000 000 000	5000
TWLDJ	000 101 000 000	0500
TWLDK	000 101 001 000	0510

Odkazy

Podobné instrukce: nd812.isa.hlt;.

-
-

NEG

Jméno

NEG — Clear register

Přehled

NEG

NEG J

NEG K

NEG JK

CMP O

Poznámky k tvaru instrukce

L
.

Popis

mnemo	kód	
	binary	octal
NEG	001 100 001 000	1410
CMP	001 100 010 000	1420
NEG O	001 100 101 000	1450
NEG J	001 101 010 100	1524
NEG JO	001 101 101 000	1550
NEG K	001 110 010 100	1624
NEG KO	001 110 101 000	1650
NEG JK	001 111 010 100	1724
NEG JKO	001 111 101 000	1750

Obrázek 77-1. Kódování instrukcí NEG

0011	K	J	O	Cpl	1	0/1	≥/ <	≠ 0	
0	3	4	5	6	7	8	9	10	11

Odkazy

Podobné instrukce: CLR.

•
•

ROTD acc

Jméno

ROTD *acc* — Rotate contents of accumulator by *n* bits left

Přehled

ROTD *J*

ROTD *K*

ROTD *JK*

(**TVAR INSTRUKCE**) *src, dst*

Poznámky k tvaru instrukce

J	K	R	S	Ov
x	x	-	-	-

Popis

Instrukce rotuje obsah akumulátoru (*J*, *K* nebo obou) o příslušný počet bitů do leva. Bity z akumulátoru neovlivňují nijak příznak přetečení Overflow.

Protože registry *J* a *K* mají 12-bitů, můžeme zakódovat operaci rotace do prava pomocí rotace do leva. Rotujeme o 12-*n* bitů. Rotace o 3 místa do prava se tedy zapíše jako rotace o 9 míst do leva.

mnemo	kód	
	binary	octal
ROTD <i>J</i>	001 001 110 000	1160
ROTD <i>K</i>	001 010 110 000	1260
ROTD <i>JK</i>	001 011 110 000	1360

Odkazy

Podobné instrukce: SFTZ.

•
•

SET acc/O

Jméno

SET acc/O — Set accumulator to -1

Přehled

SET J

SET K

SET O

SET JK

Poznámky k tvaru instrukce

L
.

Popis

mnemo	kód	
	binary	octal
SET	001 100 011 000	1430
SET J	001 101 011 000	1530
SET K	001 110 011 000	1630
SET O	001 100 111 000	1470
SET JK	001 111 011 000	1730

Obrázek 77-1. Kódování instrukcí SET

0	0011	3	K	J	O	Cpl	1	0/1	≥/<	≠ 0
			4	5	6	7	8	9	10	11

Odkazy

Podobné instrukce: nd812.isa.hlt;.

•
•

SFTZ acc

Jméno

SFTZ acc — Shift contents of accumulator by n bits left

Přehled

SFTZ J

SFTZ K

SFTZ JK

(TVAR INSTRUKCE) *src, dst*

Poznámky k tvaru instrukce

J	K	R	S	Ov
x	x	-	-	-

Popis

Instrukce posune obsah akumulátoru (J nebo K) o příslušný počet bitů do leva. Bity z akumulátoru neovlivňují nijak příznak přetečení Overflow.

mnemo	kód	
	binary	octal
SFTZ J	001 001 100 000	1140
SFTZ K	001 010 100 000	1240
SFTZ JK	001 011 100 000	1340

Odkazy

Podobné instrukce: nd812.isa.hlt;.

•
•

TCP

Jméno

TCP — Clear Flag, Print-Punch

Přehled

TCP

Popis

mnemo	kód	
	binary	octal
SFTZ JK	111 100 001 011	7413

Odkazy

Podobné instrukce: TIS.

•
•

TIF

Jméno

TIF — Keyboard-Reader Fetch

Přehled

TIF

Popis

Přečte další znak z klávesnicového bufferu. Nepřenáší žádnou informaci do akumulátoru J!

mnemo	kód	
	binary	octal
SFTZ JK	111 100 000 001	7401

Odkazy

Podobné instrukce: TIS.

·
·

TIR

Jméno

TIR — Load Keyboard Into J

Přehled

TIR

Popis

Obsah klávesového bufferu je přesunut do akumulátoru J.

mnemo	kód	
	binary	octal
SFTZ JK	111 100 000 010	7402

Odkazy

Podobné instrukce: TIS.

•
•

TIS

Jméno

TIS — Skip if Keyboard Ready

Přehled

TIS

Popis

Instrukce TIS přeskočí následující instrukci, je-li v terminálovém bufferu připraven znak z klávesnice.

mnemo	kód	
	binary	octal
SFTZ JK	111 100 000 100	7404

Odkazy

Podobné instrukce: TIR.

•
•

TOC

Jméno

TOC — Clear Flag

Přehled

TOC

Popis

Instrukce smaže příznak připravenosti terminálu.

mnemo	kód	
	binary	octal
SFTZ JK	111 100 001 001	7411

Odkazy

Podobné instrukce: TIS.

·
·

TOP

Jméno

TOP — Print-Punch

Přehled

TOP

Popis

mnemo	kód	
	binary	octal
SFTZ JK	111 100 001 010	7412

Odkazy

Podobné instrukce: TCP.

•
•

TOS

Jméno

TOS — Skip if Printer-Punch Ready

Přehled

TOS

Popis

Instrukce TOS přeskočí následující instrukci, je-li terminál připraven k přijetí dalšího znaku.

mnemo	kód	
	binary	octal
SFTZ JK	111 100 001 100	7414

Odkazy

Podobné instrukce: TIS.

•
•

TRF

Jméno

TRF — Keyboard Read-Fetch

Přehled

TRF

Popis

Kombinace instrukcí TIR a TIF. Přečte znak z klávesnicového bufferu do akumulátoru J a spustí čtení dalšího znaku.

mnemo	kód	
	binary	octal
SFTZ JK	111 100 000 011	7403

Odkazy

Podobné instrukce: TIF, TIR, TIS.

·
·

TWIO

Jméno

TWIO — Two Word I/O

Přehled

TWIO

Popis

mnemo	kód	
	binary	octal
SFTZ JK	000 111 100 000	0740

Odkazy

Podobné instrukce: IDLE.

-
-

77.15. Philco

*

Odkazy:

- Philco⁴² na Wikipedii
- A Brief History of Philco⁴³
-
-

Firma Philco vyráběla počítače Philco Model 212. Typ Philco 2000 Model 212 byl vybrán a instalován v Noradu, kde fungoval od roku 1962 do roku 1980. Podle některých údajů to byl Philco 200 S.

77.15.1. Číslicové počítače

*

Philco 2000: "první" plně tranzistorový velký počítač.

Počítač byl konstruován s ohledem na spolehlivost, jeho architektura je paralelní a asynchronní.

Paměť je rozdělena do modulů, každý obashující 8192 slov. Počítač mohl být osazen 1, 2 nebo 4-mi moduly. Maximální kapacita tedy byla 32768 slova. Šířka slova byla 48 bitů. Přesněji 56 bitů z kterých 8 bylo paritních a 48 obsahovalo hodnotu slova.

77.16. Univac

77.16.1. Univac 1218

* *Vojenský počítač*

Fakta:

- Doba přístupu k paměti je $4\mu s$
- Paměť je organizována v 18-ti bitových slovech
- Velikost paměti 4096, 8192, 16384 nebo 32768 slov.
- Část paměti o velikosti 32 slov má pevně daný obsah. Dnes bychom řekli že je to paměť ROM. Tyto část je na adresách 00200_{OCT} až 00237_{OCT} . Obsahuje zavaděč a konstanty.
- Repertoár 96 instrukcí.
-
-

Tabulka 77-33. Registry UNIVAC 1218

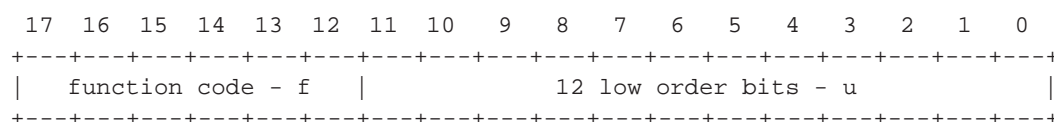
název	velikost	popis
A	36-bitů	akumulátor/sřadač
AU	18	horních 18-bitů registru A
AL	18	dolních 18-bitů registru A
B	18-bitů	aktuální index
ICR	3 bity	číslo aktuálního index registru
P		ukazatel na instrukce právě vykonávaného programu
SR	4 bity	special register

Tabulka 77-34. Adresy paměti UNIVAC 1218

addr _{OCT}	význam, funkce
000000	Fault Interrupt, Entrance Address
000001	index registr B1
000002	index registr B2
000003	index registr B3
000004	index registr B4
000005	index registr B5
000006	index registr B6
000007	index registr B7
000010	index registr B0
000011	Memory Word
000012	Memory Word
000013	Memory Word
000014	Memory Word
000015	Memory Word

addr _{oct}	význam, funkce
000016	Synchronizing Interrupt and RTC Entrance Address
000017	Scale Factor Shift Count Word
000020 - 000037	External Function Buffer Control (EFCB) Registers
000020	EFCB for Channel 0, Terminal Address Word
000021	EFCB for Channel 0, Current Address Word
000040 - 000057	Output Buffer Control (OBC) Registers
000060 - 000077	Input Buffer Control (IBC) Registers
000100 - 000117	External Interrupt (EI) Registers
000120 - 000137	External Function Monitor Interrupt (EFMI) Registers
000140 - 000157	Output Monitor Interrupt (OMI) Registers
000160 - 000177	Input Monitor Interrupt (IMI) Registers
00200 - 00237	Nondestructive Readout Memory Allocations
00240 - 77777	Instruction word and data storage organized in 10000 _{oct} word banks

Obrázek 77-40. Formát I instrukčního slova UNIVAC 1218



12 spodních bitů *u* slouží jako konstanta, nebo jako adresa.

Poznámky

1. <http://en.wikipedia.org/wiki/Artronix>
2. http://en.wikipedia.org/wiki/PC12_minicomputer
3. http://en.wikipedia.org/wiki/CDC_display_code
4. http://en.wikipedia.org/wiki/Data_General
5. http://en.wikipedia.org/wiki/Data_General_Nova
6. <http://digico-computers.com/index.htm>
7. <http://cedarsgw2.leeds.ac.uk/iclarch/arch50.html>
8. <http://www.thocp.net/timeline/uk.htm>
9. http://en.wikipedia.org/wiki/Digital_Equipment_Corporation

10. <http://www.bitsavers.org/pdf/dec/>
11. <http://www.bitsavers.org/pdf/simh/architecture18b.pdf>
12. <http://www.faqs.org/faqs/dec-faq/pdp8/>
13. <http://research.microsoft.com/en-us/um/people/gbell/Digital/DECMuseum.htm>
14. <http://pdp8.hachti.de/>
15. <http://en.wikipedia.org/wiki/PDP-11>
16. <http://www.root.cz/clanky/pdp-11-a-smep-system-malych-elektronickych-pocitacu/>
17. <http://www.village.org/pdp11/faq.html>
18. <http://web.onetel.net.uk/~hibou/Comp%20Org%20Lecture%20Course.html>
19. <http://www.village.org/pdp11/faq.pages/PDPinst.html>
20. <http://pages.cpsc.ucalgary.ca/~dsb/PDP11/>
- 21.
22. <http://www.ourcomputerheritage.org/wp/>
23. <http://malcolm.bigg.me.uk/>
24. <http://en.wikipedia.org/wiki/Honeywell>
25. http://en.wikipedia.org/wiki/Honeywell_316
26. <http://www.series16.adrianwise.co.uk/>
27. <http://www.theoengel.nl/ddpx16/simx16.html>
28. http://en.wikipedia.org/wiki/History_of_IBM
29. <http://en.wikipedia.org/wiki/AN/FSQ-7>
30. <http://www.radomes.org/museum/equip/fsq-7.html>
31. http://en.wikipedia.org/wiki/IBM_1400_series
32. http://en.wikipedia.org/wiki/IBM_1410
33. http://en.wikipedia.org/wiki/IBM_7070
34. <http://www.kzwp.com/lyons/leo.htm>
35. http://www.citizendia.org/Norsk_Data
36. <http://www.citizendia.org/NORD-1>
37. <http://www.bitsavers.org/pdf/nd/>
- 38.
- 39.
- 40.
- 41.
42. <http://en.wikipedia.org/wiki/Philco>
43. <http://www.olderadio.com/archives/hardware/philco.htm>

Kapitola 78. Historie Sovětské výpočetní techniky

Odkazy:

- [RUSSIAN VIRTUAL COMPUTER MUSEUM](#)¹
- [Tribute to the Soviet Computing - Part I](#)²
-
- [History of computer hardware in Soviet Bloc countries](#)³
-
-

Poznámky

1. <http://www.computer-museum.ru/english/index.php>
2. <http://www.youtube.com/watch?v=1Zx4NEAAAt-E>
3. http://en.wikipedia.org/wiki/History_of_computer_hardware_in_communist_countries

Kapitola 79. Historie výpočetní techniky v ČSSR a zemích východního bloku

Odkazy:

- Osmibitové muzeum¹
- Historie výpočetní techniky v Československu²
-

Poznámky

1. <http://osmi.tarbik.com/odkazy.html>
2. <http://www.historiepocitacu.cz/aritma-praha-a-dernostitkove-stroje.html>

Kapitola 80. Mechanické počítače a konstrukce

*

Odkazy:

- Category:Mechanical computers¹ na Wikipedii
- 1963: Difi-Comp I²
- Digi-Comp II³
- Digi-Comp II⁴
- Geniac⁵
- 1955 Geniac⁶
- Brainiac K-30⁷
- GENIACS: SIMPLE ELECTRIC BRAIN MACHINES, AND HOW TO MAKE THEM (1955)⁸
- DIGI-COMP I⁹
-
-
-
-

Poznámky

1. http://en.wikipedia.org/wiki/Category:Mechanical_computers
2. http://en.wikipedia.org/wiki/Digi-Comp_I
3. http://en.wikipedia.org/wiki/Digi-Comp_II
4. http://www.oldcomputermuseum.com/digicomp_2.html
5. <http://en.wikipedia.org/wiki/Geniac>
6. <http://www.oldcomputermuseum.com/geniac.html>
7. http://www.oldcomputermuseum.com/brainiac_k30.html
8. <http://www.computercollector.com/cgi-bin/exec/compcol/menu-st.cgi?directory=/archive/geniac/manual>
9. http://www.mindsontoy.com/kits.htm?dc1_main.htm

Kapitola 81. Réleové počítače

*

81.1. Z3 (Konrad Zuse)

*

Odkazy:

- Konrad Zuse's computing machine Z3¹
-

81.2. R500/7T Relay Computer

*

Odkazy:

- R500/7T Relay Computer²
-

81.3. Elektrický mozek Simon

Odkazy:

- Edmund Berkeley's Simon Relay Processor³
-

Časopis Radio-Electronics přinesl v říjnovém čísle v roce 1950 na titulní stránce obrázek releového počítače. Toto číslo obsahovalo první ze 13 ti článků o principech číslicové automatického počítání. Autor článků, Edmund Berkeley také napsal knihu "Giant Brains and Machines that Think"

Poznámky

1. <http://www.youtube.com/watch?v=2a6HMqsYnxk>
2. <http://www.youtube.com/watch?v=67jLony0mXg>
3. <http://people.cs.ubc.ca/~hilpert/e/simon/index.html>

Kapitola 82. Elektronkové počítáče a jiné stroje

Odkazy:

- List of vacuum tube computers¹
-

82.1. Colosus

*

Odkazy:

- Colossus - The First Electronic Computer - Pt1²
-

82.2. ABC

Odkazy:

- Reconstruction of the Atanasoff-Berry Computer (ABC)³
- The Atanasoff-Berry Computer⁴

82.3. ENIAC

Odkazy:

-
-

Fakta:

- Primární účel byly výpočty balistických křivek.
- Inventors: J. Presper Eckert a John Mauchly
- Builder: Moore School, University of Pennsylvania
- Uveden do provozu 14.února 1946
- Cena \$500.000
- Velikost: 40 panelů
- Technologie: 18.000 elektronek.
- Paměť: Flip-flops, function table, plugboard, cards
- Program: Propojení jednotlivých částí vodiči, plugboards, function table
- Rychlost: 5.000 operací za sekundu
-

*

ENIAC měl důležitý vliv na konstrukci svých následníků: 82.4, 82.8 a 82.5.

82.4. EDSAC

Odkazy:

-
-

Fakta:

- Inventors: Maurice Wilkes
- Builder: Cambridge University
- Uveden do provozu v roce 1949.
- Technologie: elektronky
- Paměť: 1024 17-ti bitových slov realizovaná rtuťovými spožďovacími linkami (mercury delay line)
- Program: single address
- Rychlost: 714 operací za sekundu.

82.5. UNIVAC

Odkazy:

-
-

Fakta:

- Inventors: J. Prespet Eckert a John Mauchly
- Builder: Remington Rand
- Cena: \$750.000
- Vysokorychlostní tiskárny \$185.000
- Velikost: 943 kubických stop.
- Uveden do provozu v roce 1951.
- Technologie: elektronky, zpožďovací linky, magnetická páska
- Paměť: 1024 slov, 12 číslic velikých, magnetická páska
- Rychlost: 1905 operací za sekundu.

82.6. EDVAC

Odkazy:

-
-

Fakta:

- Inventors: J. Presper Eckert a John Mauchly
- Builder: Moore School, University of Pennsylvania
- Uveden do provozu v roce 1951.
- Cena \$467.000
- Velikost: umístněn na ploše 480 čtverečních stop. Výha 17.000 liber.
- Technologie: 4.000 elektronek, rtuťové zpožďovací linky (mercury delay lines)
- Paměť: 1024 slov o šířce 44 bitů. 4608 slov bubnové paměti.
- Rychlost: 1.200 operací za sekundu.

- Použití: balistické výpočty, střelecké tabulky, redukce dat.
-

*

82.7. LEO

Odkazy:

-
-

Fakta:

- Komerční počítač
-
-
-

*

82.8. IAS

Odkazy:

-
-

Fakta:

- Inventors: John Von Neumann, Arthur Burks, Herman Goldstein
- Builder: Princeton Institute for Advanced Studies
- Uveden do provozu v roce 1952
- Cena \$500.000
- Počítač zabíral plochu 22 čtverečních stop a vážil 1.000 liber.
- Technologie: 4.600 elektronek, elektrostatické paměťové elektronky
- Paměť 1024 slov, každé 40 bitů velké. 16.384 slov bubnové paměti.
- Program: Single address
- Rychlost: 14.000 operací za sekundu.

82.9. MANIAC

Odkazy:

-
-

Fakta:

- Builder: Los Alamos
- Uveden do provozu v roce 1952
- Cena \$200.000
- Počítač zabíral objem 128 krychlových stop.
- Paměť elektrostatická 1024 slov, každé 40 bitů velké. Bubnová 10.000 slov.

- Rychlost: 12.000 operací za sekundu.
- Operation ratio: 83%

82.10. SSEM (Small Scale Experimental Machine)

[1948-06-21]

Odkazy:

- The Manchester Small Scale Experimental Machine -- "The Baby"⁵
- The SSEM Computer: Small Scale Experimental Machine ⁶

Počítač SSEM, přezdívaný „the Baby“ byl zkonstruován v universitě v Manchesteru za účele otestování paměťové elektronky Williams-Kilburn. Tyto elektronky byly připravovány pro konstrukci počítače Mark 1. Tyto elektronky si pamatovaly 32 slov každé 32 bitů dlouhé.

Počítač byl poprvé uveden do provozu 21. června 1948.

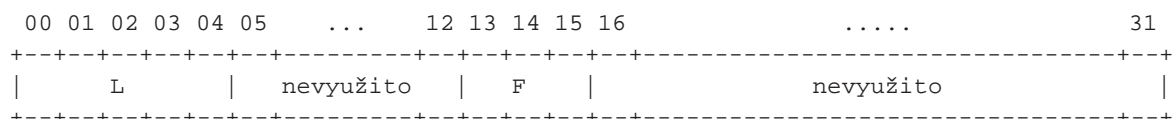
Časem byl počítač rozšířen a stal se z něj Manchester Mark 1 [1949]. Komerčně byl vyráběn od roku 1951 pod názvem Ferranti Mark 1.

82.10.1. Instrukční sada

Počítač byl velmi jednoduchý, používal 32 bitové slovo a měl jen 7 instrukcí.

Poznámka: V následujících obrázcích a tabulkách je nejméně významný bit číslo 0 zobrazen vždy vlevo v souladu s původní dokumentací. Z toho vychází i binární kódy instrukcí uvedené v tabulce instrukcí.

Obrázek 82-1. Formát instrukčního slova SSEM



Tabulka 82-1. SSEM Instruction Set

Code		Modern Mnemonic	Action	Description
Dec	Bin			
0	000	JMP	$S(L) \rightarrow CI$	Jump Indirect. Jump to one after the contents of the specified Store line.
1	100	JPR	$CI + S(L) \rightarrow CI$	Jump Relative. Add contents of specified Store line to CI
2	010	LDN	$-S(L) \rightarrow A$	Load Negative. Load the Accumulator with the negative of the contents of the specified Store line.
3	110	STO	$A \rightarrow S(L)$	Store. Copy contents of the Accumulator to the specified Store line.

Code		Modern Mnemonic	Action	Description
Dec	Bin			
4	001	SUB	$A - S(L) \rightarrow A$	Subtract. Subtract contents of specified Store line from Accumulator.
6	011	SKN	if $A < 0$ then $CI + 1 \rightarrow CI$	Skip if Negative. Skip next instruction if contents of Accumulator is negative.
7	111	HLT		Halt. Stop instruction execution. The Stop neon will light

Zvláštností modelu SSEM a i několika dalších je pořadí operací při vykonávání instrukce. Na rozdíl od obvyklého:

FETCH, Increment-PC, Execute, ...

Má SSEM pořadí:

Increment-PC, FETCH, Execute, ...

Tento drobný rozdíl, kdy ukazatel na instrukci je zvětšen předtím, než je instrukce načtena způsobí že program nezačíná na adrese 0, ale až na adrese 1. Rovněž pokud měníme tok instrukcí pomocí instrukce skoku, musíme uvést adresu o 1 menší než na kterou chceme skočit.

Poznámky

1. http://en.wikipedia.org/wiki/List_of_vacuum_tube_computers
2. <http://www.youtube.com/watch?v=O8WXNPn1QKo>
3. <http://www.scl.ameslab.gov/ABC/ABC.html>
4. <http://people.cs.ubc.ca/~hilpert/e/ABC/index.html>
5. <http://www.computer50.org/mark1/new.baby.html>
6. <http://people.cs.ubc.ca/~hilpert/e/SSEM/index.html>

Kapitola 83. Počítače druhé generace

Odkazy:

- POČÍTAČE DRUHÉ GENERACE - tranzistorové¹
-

Druhá generace počítačů je konstruována s pomocí transistorů.

83.1. IBM 1401

83.2. ELLIOT 803

Poznámky

1. http://sen.felk.cvut.cz/sen/index_cz.html?historie/gen2.html

Kapitola 84. Doba elektronková

*

Časové údaje:

- - 1955:
 - ZUSE Z22¹
 - Z22² — na Wikipedii

84.1. Whirlwind (MIT)

Odkazy:

- The pre-history of the Digital Equipment Corporation³
- Index of /pdf/mit/whirlwind⁴ na BitSavers.org

Počítač Whirlwind byl zkonstruován MIT jako hlavní počítač a součást projektu SAGE

Fakta:

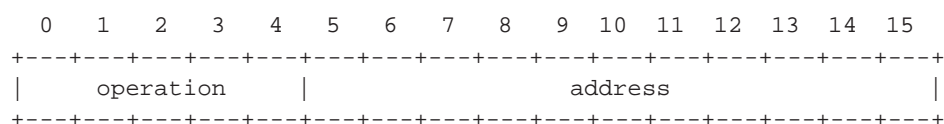
- 16-ti bitové slovo, na svou dobu nezvykle krátké
- instrukce má 5 bitů operačního znaku a 11 bitů adresy
-

Vzhledem k zadání, tedy že veškerá elektronika bude snadno dostupná byl počítač fyzicky řešen ve 2D prostoru. Z toho vzešly obrovské požadavky na zastavěnou plochu.

Počítač měl přes 5000 elektronek a 11 germaniových diod.

Jeden z neobvyklých vlastností bylo použití diodové matice k dekódování instrukcí. V té době se běžně používaly nestrukturované logické obvody (Random Logic).

Obrázek 84-1. Instrukční slovo Whirlwind



Tabulka 84-1. Instrukce počítače Whirlwind

Oper.	Function	Binary Code	Octal	Dec	Time	Oper.
clc n	cycle left and clear (BR)	11110-0	1.70	30	15+0.8n μ s	clc
clh n	cycle left and hold	11110-1	1.70	30	15+0.8n μ s	clh
md x	multiply digits with no roundoff (p)	11111	1.74	31	22 μ s	md
si pqr	select in-out unit/stop	00000	0.00	0	30 μ s	SI
ill	illegal instruction	00001	0.04	1	n/a	ILL
bi x	block transfer in	00010	0.10	2	max 16ms 1st; 16 μ s each word	BI

Oper.	Function	Binary Code	Octal	Dec	Time	Oper.
rd		00011	0.14	3	15 μ s	RD
bo x	block transfer out	00100	0.20	4	max 16ms 1st; 16 μ s each word	BO
rc x	record	00101	0.24	5	22 μ s	RC
sd x	sum digits	00110	0.30	6	22 μ s	SD
cf pqr	change fields	00111	0.34	7	15 μ s	CF
ts x	transfer to storage	01000	0.40	8	22 μ s	TS
td x	transfer digits	01001	0.44	9	29 μ s	TD
ta x	transfer address	01010	0.50	10	29 μ s	TA
ck x	check	01011	0.54	11	22 μ s	CK
ab x	add B	01100	0.60	12	29 μ s	AB
ex x	exchange	01101	0.64	13	29 μ s	EX
cp x	conditional program	01110	0.70	14	15 μ s	CP
sp x	sub-program	01111	0.74	15	15 μ s	SP
ca x	clear and add	10000	1.00	16	22 μ s	CA
cs x	clear and subtract	10001	1.04	17	22 μ s	CS
ad	add	10010	1.10	18	22 μ s	AD
su	subtract	10011	1.14	19	22 μ s	SU
cm	clear and add magnitude	10100	1.20	20	22 μ s	CM
sa x	special add	10101	1.24	21	26 μ s	SA
ao x	add one	10110	1.30	22	29 μ s	AO
dm x	difference of magnitudes	10111	1.34	23	22 μ s	DM
mr x	multiply and roundoff	11000	1.40	24	34-41 μ s	MR
mh x	multiply and hold	11001	1.44	25	34-41 μ s	MH
dv x	divide	11010	1.50	26	71 μ s	DV
slr x	shift left and roundoff	11011 0	1.54	27	15+0.8n μ s	SLR
slh x	shift left and hold	11011 1	1.54	27	15+0.8n μ s	SLH
srr x	shift right and roundoff	11100 0	1.60	28	15+0.8n μ s	SRR
srh x	shift right and hold	11100 1	1.60	28	15+0.8n μ s	SRH
sf x	scale factor	11101	1.64	29	30-78 μ s	SF

84.1.1. Abecední přehled instrukcí Whirlwind

AB

Jméno

AB — add B

Přehled

AB *x*

Popis

Add the contents of the B register to the contents of register x and store the result in th AC and register x. The contents of register x appear in AR. SAM is cleared. Overflow may occur giving an arithmetic check alarm if $|C(x)+C(B)| \geq 1$. (dm x or clh 16 puts C(AC) into B)

AB	01100 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

AD

Jméno

AD — add

Přehled

AD *x*

Popis

Add the contents of memory[x] to contents of AC, storing result in AC. The contents of memory[x] appear in AR. SAM is cleared. Overflow may occur giving an arithmetic check alarm if $|C(AC) + C(mem[x])| \geq 1$.

AD	10010 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

AO

Jméno

AO — add one

Přehled

AO *x*

Popis

Add the number 1 times 2^{-15} to contents of memory[*x*], storing the result in AC and in memory[*x*]. The original contents of memory[*x*] appear in AR. SAM is cleared. Overflow may occur giving an arithmetic check alarm if $\text{mem}[x] + (1 * 2^{-15}) = 1$.

AO	10110 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

BI

Jméno

BI — block transfer in

Přehled

BI *x*

Popis

Transfer a block of *n* words or characters from an in-out unit to MCM, where register *x* is the initial address of the block in MCM, and $\pm n$ times 2^{-15} is contained in AC. The computer is stopped while the transfer is taking place. After a block transfer, AC contains the address which is one greater than the MCM address at which the last word was placed; AR contains the initial address of the block in MCM.

BI	00010 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

BO

Jméno

BO — block transfer out

Přehled

BO x

Popis

Transfer block of n words from MCM to an in-out unit where x is the initial address of the block in MCM, and $\pm n$ times 2^{-15} is contained in AC. The computer is stopped while the transfer is taking place. After the block transfer, AC contains the address which is one greater than the MCM address from which the last word was taken and stored; AR contains the initial address of the block in MCM.

BO	00100 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

CA

Jméno

CA — clear and add

Přehled

CA x

Popis

Clear AC and BR, then obtain contents of SAM (+1, 0, or -1) times 2^{-15} and add contents of memory[x], storing result in AC. The contents of memory[x] appear in AR. SAM is cleared. Overflow may occur giving an arithmetic check alarm if $|C(x) + C(\text{SAM}) 2^{-15}| = 1$

CA	10000 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

CF

Jméno

CF — sum digits

Přehled

CF pqr

Popis

The address section does not refer to a register of storage in this instruction, but supplies information to the computer requesting a change in fields Group A and/or B. When the fields to be changed contains the program, it is necessary for the cf instruction to perform like an SP instruction. Digit 7 of the cf address section causes the contents of the accumulator to be read to the program counter (PC) prior to the field change; thus, program continuity can be preserved during field changes. The A-Register will contain the original PC address plus one upon completion of the cf instruction. The digit allocation for the cf word is as follows:

bity	význam
0-4	00111 — instrukce cf
5	nepoužito
6	examine feature — causes contents of core memory Group A control and Group B control registers to be read into the Accumulator.

bity	význam
7	sp enable — reads content of AC to PC to establish starting point of program in the new field.
8	change Group A field enable. (If, when the examine feature of digit 6 is requested, there is a "1" in digit 8, the content of Group A control will be changed before read-out to the A-Register takes place.
9	change Group B field enable. (If, when the cf examination feature is requested, there is a "1" in digit 9, the content of Group B control will be changed before read-out to the A-Register takes place.
10-12	contain field designation for Group A (registers 40-1777)
13-15	contain field designation for Group B (registers 2000-3777)

CF	00111 xxxAB aaa bbb
----	---------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

CK

Jméno

CK — check

Přehled

CK x

Popis

Operates in two modes depending upon position of console switch (at T.C.) labelled "Program Check Alarm on Special Mode." The NORMAL MODE is selected if the switch is off or down. Normal Mode compares contents of AC with contents of register x. If contents of AC are identical to contents of register x, proceed to next instruction; otherwise, stop the computer and give a "check register alarm." (Note: +0 is not identical to -0.) SPECIAL MODE is chosen if switch is on. Special Mode operates in same way as above if the numbers being checked agree. If there is disagreement, no check alarm will occur but the program counter (PC) will be indexed by one, causing the next instruction to be skipped.

CK	01011 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

CM

Jméno

CM — clear and add magnitude

Přehled

CM 

Popis

Clear AC and BR, then obtain contents of SAM(+1, 0, -1) times 2^{-15} and add magnitude of contents of memory[x], storing result in AC. The magnitude of contents of memory[x] appears in AR. SAM is cleared. Overflow may occur giving an arithmetic check alarm if $|C(\text{mem}[x]) + C(\text{SAM}) 2^{-15}| = 1$.

CM	10100 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

CP

Jméno

CP — conditional program

Přehled

CP 

Popis

If number in AC is negative, proceed as in SP. If number in AC is positive, proceed to next instruction, clear the AR and place in the last 11 digit positions of the AR the address of this next instruction.

CP	01110 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

CS

Jméno

CS — clear and subtract

Přehled

CS x

Popis

Clear AC and BR, then obtain contents of SAM (+1, 0, or -1) times 2^{-15} and subtract contents of memory[x], storing result in AC. The contents of memory[x] appear in AR. SAM is cleared. Overflow may occur giving an arithmetic check alarm if $|-C(x) + C(SAM) 2^{-15}| = 1$

CS	10001 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

DM

Jméno

DM — difference of magnitudes

Přehled

DM x

Popis

Subtract the magnitude of contents of memory[x] from the magnitude of contents of AC, leaving result in AC. The magnitude of memory[x] appears in AR. SAM is cleared. BR will contain the initial contents of the AC. If $|AC| = |\text{mem}[x]|$, the result is -0.

DM	10111 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

DV

Jméno

DV — divide

Přehled

DV x

Popis

Divide contents of AC AC by contents of memory[x], leaving 16 binary digits of the quotient in BR and ± 0 in AC according to the sign of the quotient. The instruction **SLR** 15 following the dv operation will roundoff the quotient to 15 binary digits and store it in AC. Let u and v be the numbers in AC and memory[x], respectively, when the instruction dv x is performed. If $|u| < |v|$, the correct quotient is obtained and no overflow can arise. If $|u| > |v|$, the quotient exceeds unity and a divide-error alarm will result. If $u = v \neq 0$, the dv instruction leaves 16

"ones" in BR; roundoff in subsequent slr 15 will cause overflow and give an arithmetic overflow check alarm. If $u = v = 0$, a zero quotient of the appropriate sign is obtained. The magnitude of contents of memory[x] appears in AR. SAM is cleared.

DV	11010 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

EX

Jméno

EX — exchange

Přehled

EX x

Popis

Exchange contents of AC with contents of memory on address x. (Original contents of AC in memory[x]; original contents of memory[x] in AC and AR.) Ex 0 will clear AC without clearing BR.

EX	01101 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

ILLEGAL

Jméno

ILLEGAL — illegal instruction

Přehled

ILLEGAL

Popis

If an attempt is made to use this operation, a check alarm will result.

ILLEGAL	00001 xxxxxxxxxxxx
---------	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

MH

Jméno

MH — multiply and hold

Přehled

MH x

Popis

Multiply contents of AC by contents of memory[x]. Retain the full product in AC and in the first 15 digit positions of BR, the last digit position of BR being cleared. The magnitude of contents of memory[x] appears in AR. SAM is cleared. Sign of AC is determined by sign of product. Result in (AC + BR) is double register product. The is determined the same as for MR.

MH	11001 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

MR

Jméno

MR — multiply and roundoff

Přehled

MR \times

Popis

Multiply contents of AC by contents of memory[x]. Roundoff result from BR to 15 significant binary digits and store it in AC. If $BR(0) = 1$, roundoff is 2^{-15} ; if $BR(0) = 0$, roundoff is 0. Clear BR. The magnitude of contents of memory[x] appears in AR. SAM is cleared. Sign of AC is determined by sign of product. The minimum time of $34 \mu s$ occurs if there are all 0's in |AC|; the maximum time of $41 \mu s$ occurs when there are all 1's in |AC|. The average time is $37.5 \mu s$.

MR	11000 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

RC

Jméno

RC — record

Přehled

RC \times

Popis

Transfer contents of AC via IOR to an in-out unit. IOR is cleared only after an rc used as a display instruction. The address section of the instruction has no significance except when used with the character generator.

RC	00101 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

RD

Jméno

RD — read

Přehled

RD \times

Popis

Transfer word from IOR to AC, then clear IOR. (Wait, if necessary, for information to arrive in IOR from an in-out unit.) Contents of AR are identical to contents of AC. The address section of the instruction has no significance.

RD	00011 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

SA

Jméno

SA — special add

Přehled

SA x

Popis

Add contents of memory[x] to contents of AC, storing fractional result in AC and retaining in SAM any overflow (including sign) for use with next CA, CS or CM instruction. Between sa and the next CA, CS or CM instruction for which the sa is a preparation, the use of any instruction which clears SAM will result in the loss of the overflow with no other effect on the normal function of the operation. The following operations clear SAM without using its contents: SD, SU, SA AO DM, MR, MH, DV, **FIXME:whirlwind.isa.sl**, **FIXME:whirlwind.isa.sr**; and **FIXME:whirlwind.isa.sf**. CA, CS or CM clear SAM after using its contents. If the overflow resulting from the sa is to be disregarded, care must be taken to destroy it before the next CA, CS or CM instruction. The contents of memory[x] appears in AR. SAM is cleared before, but not after, the addition is performed.

SA	10101 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME:whirlwind.isa.halt**;

SD

Jméno

SD — sum digits

Přehled

SD x

Popis

The sum of the original contents of digit i of AC and original contents of digit i of register x becomes stored in digit i of AC. The final value of digit i of AC is 0 if the values of digit i of AC and register x are alike; the final value of digit i of AC is 1 if the values of digit i of AC and register x are different.

SD	00110 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

SF

Jméno

SF — scale factor

Přehled

SF x

Popis

Multiply the contents of AC and BR by 2 often enough to make the positive magnitude of the product equal to or greater than 1/2. Leave the final product in AC and BR. Store the number of multiplications in AR and in last 11 digit places of memory[x], the first 5 digits being undisturbed in memory[x]. If all the digits in BR are zero and AC contains ± 0 , the instruction sf x leaves AC and BR undisturbed and stores the number 33 times 2^{-15} in AR and in the last 11 digit positions of memory[x]. Negative numbers are complemented in AC before and after the multiplication (by shifting); hence ones appear in the digit places made vacant by the shift. SAM is cleared. The time varies according to the number of 0's between the binary point and first 1 in magnitude of binary fraction represented by the number AC and BR. The minimum time of $30 \mu s$ occurs when $|AC + BR| \geq 1/2$; the maximum time of $78 \mu s$ occurs when $|AC+BR| = 0$.

SF	11101 xxxxxxxxxx
----	------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

SI

Jméno

SI — select in-out unit/stop

Přehled

SI pqr

Popis

Stop any in-out unit that may be running. Select a particular in-out unit and start it operating in a specified mode, designated by the digits p q r; or, stop the computer. Si 0 will stop the computer; si 1 will stop the computer only if the "Conditional Stop" switch is ON. A program alarm may occur if the computer is not ready to receive information transmitted to it from selected in-out unit (e.g., the use of an si address if si selects Magnetic Tape, or PETR without the necessary rd instruction following). Also an inactivity alarm may result when using an rc or bo with an si instruction which selects the read mode.

SI	00000 xxxxxxxxxx
----	------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

SLH

Jméno

SLH — shift left and hold

Přehled

SLH x

Popis

Shift contents of AC (except sign digit) and BR to the left n places. The positive integer n is treated modulo 32; digits shifted left out of AC₁ are lost. (Shifting left n places is equivalent to multiplying by 2ⁿ, with the result reduced modulo 1.) Leave final product in AC and BE. Do not eoundoff or clear BR. Negative numbers are complemented before the shift; hence, ones appear in the digit places made vacant by the shift of a negative number. Digit 6 (the 2⁹ = 512 digit of the address) of the instruction slh n must be a one to distinguish slh n from SLR n described below. SAM is cleared. The execution time depends upon the size of n.

SLH	11011 1xxxxxxxxxxx
-----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

SLR

Jméno

SLR — shift left and roundoff

Přehled

SLR \times

Popis

Shift fractional contents of AC (except sign digit) and B to the left n places. The positive integer n is treated modulo 32; digits shifted left out of AC_1 are lost. (Shifting left n places is equivalent to multiplying by 2^n , with the result reduced modulo 1.) Roundoff the result to 15 binary digits and store it in AC. Clear BR. Negative numbers are complemented before the shift and after the roundoff; hence, ones appear in the digit places made vacant by the shift of a negative number. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction slr n must be a zero to distinguish slr n from SLH n described below. The instruction slr 0 simply causes roundoff and clears BR. SAM is cleared. Roundoff (ρ) may cause overflow with a consequent arithmetic check alarm if $|F(AC+BR)2^n + \rho| = 1$. The execution time varies according to the size of n .

SLR	11011 0xxxxxxxxxx
-----	-------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

SP

Jméno

SP — sub-program (transfer control)

Přehled

SP x

Popis

The next instruction from register x. If the sp instruction was at address y, clear the AR and store y+1 in the last 11 digit positions of AR. x becomes the contents of the PC.

SP	01111 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

SRH

Jméno

SRH — shift right and hold

Přehled

SRH x

Popis

Shift contents of AC (except sign digit) and BR to the right n places. The positive integer n is treated modulo 32; digits shifted right out of BR_{15} are lost. (Shifting right n places is equivalent to multiplying by 2^{-n} .) Do not roundoff the result or clear BR. Result is stored in AC and BR. Negative numbers are complemented in AC before and after the shift; hence, ones appear in the digit places made vacant by the shift of a negative number. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction srh n must be a one to distinguish srh n from SRR n described below. SAM is cleared. The time depends upon the size of the integer n.

SRH	11100 1xxxxxxxxx
-----	------------------

Odkazy

Podobné instrukce: **FIXME**;whirlwind.isa.halt;.

SRR

Jméno

SRR — shift right and roundoff

Přehled

SRR *x*

Popis

Shift contents of AC (except sign digit) and BR to the right n places. The positive integer n is treated modulo 32; digits shifted right out of BR_{15} are lost. (Shifting right n places is equivalent to multiplying by 2^{-n} .) Roundoff the result to 15 binary digits and store it in AC. Clear BR. Negative numbers are complemented before the shift and after the roundoff; hence, ones appear in the digit places made vacant by the shift of a negative number. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction `srr n` must be a zero to distinguish `srr n` from `SRH n` described below. The instruction `srr 0` simply causes roundoff and clears BR. SAM is cleared. Roundoff (in an `srr0`) may cause overflow with a consequent arithmetic check alarm. The time depends upon the size of n .

SRR	11100 0xxxxxxxxxx
-----	-------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

SU

Jméno

SU — subtract

Přehled

SU *x*

Popis

Subtract contents of memory[x] from contents of AC, storing result in AC. The contents of memory[x] appear in AR. SAM is cleared. Overflow may occur giving an arithmetic check alarm if $|C(AC) - C(mem[x])| \geq 1$.

SU	10011 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

TA

Jméno

TA — transfer address

Přehled

TA x

Popis

Transfer last 11 digits of AR to last 11 digit positions of register x. The original contents of the last 11 digit positions of register x are destroyed. The ta operation is normally executed after an SP or **FIXME**:whirlwind.isa.cp; instruction in connection with sub-programming; less frequently after **FIXME**:whirlwind.isa.ao;, **FIXME**:whirlwind.isa.sf; or other operations.

TA	01010 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

TD

Jméno

TD — transfer digits

Přehled

TD *x*

Popis

Transfer last 11 digits of AC to last 11 digit positions of register *x*. The original contents of the last 11 digit positions of register *x* are destroyed.

TD	01001 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

TS

Jméno

TS — transfer to storage

Přehled

TS *x*

Popis

Transfer contents of AC to register *x*. The original contents of register *x* are destroyed.

TS	01000 xxxxxxxxxxxx
----	--------------------

Odkazy

Podobné instrukce: **FIXME**:whirlwind.isa.halt;.

Poznámky

1. http://user.cs.tu-berlin.de/~zuse/Konrad_Zuse/en/rechner_z22.html
2. <http://en.wikipedia.org/wiki/Z22>
3. <http://www.cs.man.ac.uk/CCS/res/res04.htm#f>
4. <http://www.bitsavers.org/pdf/mit/whirlwind/>

Kapitola 85. Doba transistorová

Vše co se mi podařilo sehnat o tranzistorových počítačích.

Odkazy:

- Přehled tranzistorových počítačů¹
-

12-ti bitové počítače

- CDC-160
- LINC
- PDP-5
- PDP-8

Historie:

-
-

1961:

- ZUSE Z23² — následník Z22
-

85.1. TX-0

* *Attributy: chapter id="tx-0" ent=TX-0*

Odkazy:

- Kolekce informací z MIT³
- Programy pro TX0⁴
- A Brief History of the Internet⁵
- The pre-history of the Digital Equipment Corporation⁶
- TX-0⁷ on Absolute Astronomy
- L-1
- Highlights from The Computer Museum Report⁸ Volume 8 — Spring 1984
-

Počítač TX-0 vznikl původně jako logických obvodů a feritových pamětí pro projektovaný tranzistorový počítač TX-2. Použité transistory byly nové SBT100 (surface barrier transistor) od firmy Philco. Jejich cena byla tehdy \$80. TX-0 obsahoval kolem 3 600 takových transistorů.

Velikost datového slova uloženého v paměti byla 18-bitů a paměť byla omezena 16-ti bitovou adresou na 64K slov. Instrukce byla celá uložena v jednom slově.

85.1.1. Original TX-0 Registers and Instruction Set

Instrukční sada

Odkazy:

- MIT TX-0 INSTRUCTION SET⁹

Původní TX-0 z Lincoln Labs měl operační kód dlouhý 2 bity. Také neměl žádný indexový registr. Později, v roce 1958 při stěhování na MIT byl operační kód rozšířen ze dvou na čtyři bity a byl přidán indexový registr.

Dva bity operačního kódu rozlišovaly 4 instrukce. Tyto instrukce jsou ve zkratce STORE, ADD, JUMP IF LESS THAN a OPERATE

```
000000+addr  STO  addr
200000+addr  ADD  addr
400000+addr  TRN  addr    // jump of AC0 is zero
600000+code  OPR  code
```

Instrukce OPR je vlastně celá třída „mikrokódovaných“ instrukcí. Jejím parametrem není adresa ale 16 bitů. Každý z těchto bitů ovlivňuje funkci některé části počítače a jejich vzájemnou kombinací vznikají sofistikované instrukce. Tento postup byl dále rozveden v takových počítačích jako je například 63.1.2

Instrukční slovo je široké 18 bitů. Z těchto 18-ti bitů tvoří instrukce první dva a zbylých 16 je adresa.

Tabulka 85-1. Registry TX-0

reg	popis
MBR	Memory Buffer Register (18 bits)
AC	Accumulator (18 bits)
MAR	Memory Address Register (16 bits)
PC	Program Counter (16 bits)
IR	IR Instruction Register (2 bits)
LR	Live Register (18 bits)
TBR	Toggle Switch Buffer Register (18 toggle switches)
TAC	Toggle Switch Accumulator (18 toggle switches)

IR ₀	IR ₁	ABBREVIATION	INSTRUCTION
0	0	sto x	Replace the contents of register x with the contents of the AC. Let the AC remain the same.
0	1	add x	Add the word in register x to the contents of the AC and leave the sum in the AC
1	0	trn x	If the sign digit of the accumulator (AC ₀) is negative (i.e., a one) take the next instruction from there. If the sign is positive (i.e., a zero) ignore this instruction and proceed to the next instruction.
1	1	opr x	Execute one of the operate class commands indicated by the number x.

85.1.1.1. Operate Class Commands

Popis instrukce, nebo lépe instrukcí třídy operate.

100 000	CLL	Clear left nine digital positions of the AC.
40 000	CLR	Clear right nine digital positions of the AC.

20 000	IOS	In-Out Stop. Stop machine so that an In-Out command (specified by digits 6,7, and 8 of MAR) may be executed.
30 000	HLT	Halt the computer.
7 000	P7H	Punch holes 1-6 in flexo tape specified by AC digital positions 2, 5, 8, 11, 14, and 17. Also punch a 7th hole in tape.
6 000	P6H	Same as P6H but no seventh hole.
4 000	PNT	Print one flexowriter character specified by AC digits 2, 5, 8, 11, 14, and 17.
1 000	R1C	Read ine line of flexo tape so that tape positions 1, 2, 3, 4, 5, and 6 will be put in the AC digital positions 0, 3, 6, 9, 12 and 15.
3 000	R3C	Read ine line of flexo tape so that tape into AC digits 0, 3, 6, 9, 12 and 15. Then cycle the AC one digital positions; read the next line on tape into AC digits 0, 3, 6, 9, 12 and 15, cycle the AC right one digital positions and read the third and last line into AC digits 0, 3, 6, 9, 12 and 15. (This command is equal to triple CYR-R1C.)
2 000	DIS	
2 000	DIS	Intensifi a point in the scope with x and y coordinates where x is specified by AC digits 0-8 with digit 0 being used as the sign and y is specified by AC digits 9-17 with digit 9 being used as the sign for y. The complement system is in effect when the signs are negative.
400	SHR	Shift the AC right in place, i.e., multiply the AC by 2 ⁻¹ .
600	CYR	Cycle the AC right one digital position (AC ₁₇ will become AC ₀)
200	MLR	Store the contents of the MBR (memory buffer register) in the live reg.
100	PEN	Read the light pen flip-flops 1 and 2 into AC ₀ and AC ₁ .
4	TAC	Insert a one in each digital position of the AC wherever there is a ine in the corresponding digital position of the TAC.
40	COM	Complement every digit in the accumulator.
20	PAD	Partial add AC to MBR, that is, for every digital position of th MBR that contains a one, complement the digit in the corresponding digital position of the AC. This is also called a half add.
10	CRY	
1	AMB	
3	TBR	
2	LMB	

Obrázek 85-1. Operate Fields

--1	---	---	---	---	---	CLL
---	1--	---	---	---	---	CLR
---	-10	---	---	---	---	IOS
---	-11	---	---	---	---	HLT
---	---	111	---	---	---	P7H
---	---	110	---	---	---	P6H
---	---	100	---	---	---	PNT
---	---	010	---	---	---	DIS
---	---	---	10-	---	---	SHR
---	---	---	11-	---	---	CYR
---	---	---	01-	---	---	MLR
---	---	---	--1	---	0--	PEN
---	---	---	--0	---	1--	TAC

```

--- 1--- COM
--- -1- PAD
--- --1 CRY
--- -01 AMB AC -> MBR
--- -11 TBR TBR -> MBR
--- -10 LMB LR -> MBR

```

85.1.1.2. Coded Operate Class Commands

700000	c11	Clear the left nine digital positions of the AC
640000	clr	Clear the right nine digital positions of the AC
620000	ios	In-Out Stop
630000	hlt	Halt the computer
607000	p7h	Punch holes 1-6 in flexo tape. Also punch a 7th hole on tape.
606000	p6h	Same as p7h but no seventh hole
CLA	opr 140000	Clear the accumulator
600002	lmb	Store the contents of the LR in the MBR

Tabulka 85-2. TX0 Combined Operate Class Commands

Opcode	Mnemo	Combination	Description
740 000	cla	c11 + clr	Clear the AC.
600 031	cyl	amb + pad + cry	Cycle the AC left one digital position.
740 040	clc	c11 + clr + com	Clear and complement AC
622 000	dis	ios + dis	Display (note IOS must be included for in-out cmds)
760 000		ios + c11 + clr	In out stop with AC cleared

85.1.2. Modified TX-0 Register and Instruction Set

Tabulka 85-3. Registry modifikovaného TX-0

reg	popis
AC	18-ti bitový akumulátor
XR	14-ti bitový indexový registr
LR	18-ti bitový in-out registr (live registr)
MBR	18-ti memory buffer register
TBR	Toggle Switch Buffer Regisster (18 přepínačů)
TAC	Toggle Switch Accumulator (18 přepínačů)
PF	program flag register (?)

Obrázek 85-2. Instrukční slovo modifikovaného TX-0

```

0 1 2 3 4 5 ..... 15
+---+---+---+---+---+---+---+---+---+---+---+---+
| OP  | X |           ADR           |
+---+---+---+---+---+---+---+---+---+---+---+---+

```

Tabulka 85-4. Základní instrukce modifikovaného TX-0

IR ₀₋₃ , X	mnemo	popis
00000	sto y	Store
00001	stx y	Store AC, Indexed
00010	sxa y	Store Index in Address
00011	ado y	Add One
00100	slr y	Store Live Register
00101	slx y	Store LR Indexed
1111		

85.2. CDC 160

* *Attributy:* id="cdc-160" xreflabel="CDC 160", ent=CDC-160 ent=CDC-160A

Odkazy:

- The Control Data Corporation 160 Computer¹⁰
- Control Data Corporation Model 160¹¹
- Characteristics Of The CONTROL DATA MODEL 160-A COMPUTER¹²
- CDC 160A¹³ na Wikipedii

Počítač CDC160 byl představen v roce 1960.

Používal 12-ti bitová slova, a adresa paměti byla rovněž 12-ti bitová. To dovolovalo adresovat 4K slov.

85.2.1. Registry

CDC-160 má následující registry:

S

Memory addressing. Hlavní paměť CDC-160 má 4K slova adresované 12-ti bitovým registrem S.

P

Program Counter. Čítač programu, obsahuje adresu následující instrukce. Čítač je inkrementován po načtení instrukce.

A

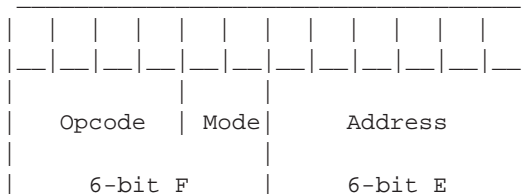
Accumulator. 12-ti bitový střadač. Obsahuje výsledky logických a aritmetických operací.

85.2.2. Instrukční sada

Většina instrukcí má následující formát

Obrázek 85-3. Formát instrukcí počítače CDC-160

11 10 09 08 07 06 05 04 03 02 01 00



Tabulka 85-5. Tabulka instrukcí CDC-160

opcode	význam	
0000	Immediate Instructions	
0001	Immediate Instructions	
0010	And	
0011	Or	
0100	Load	
0101	Load Complement	
0110	Add	
0111	Subtract	
1000	Store	
1001	Shift and Replace	
1010	Add and Replace	
1011	Add One and Replace	
1100		
1101		
1110		
1111		

Tabulka 85-6. Adresní módy CDC-160

00	Direct	Přímé adresování nulté stránky paměti. Tedy pamětových buněk na adresách 0 až 63.
01	Indirect	
10	Forward Relative	
11	Backward Relative	

Tabulka 85-7.

Opcode	Mnemonic	Description	
000 000 000 000	ERR	Halt	halt the cpu
000 000 000 ???	NOP		
000 000 000 111	NOP		
000 000 bbb vvv		bank set	b≠0
000 001 000 000	BLS		
000 001 000 001	PTA	Transmit Program Counter to Accumulator	$A \leftarrow P$
000 001 000 010	LS1	Shift Left	
000 001 000 011	LS2	Shift Left 2	
000 001 000 100	CBC		
000 001 000 101	ATE		
000 001 000 110	ATX		
000 001 000 111	ETA		
000 001 001 000	LS3	Shift Left 3	
000 001 001 001	LS6	Shift Left 6	
000 001 001 010	MUT	Multiply by 10	
000 001 001 011	MUH	Multiply by 100	
000 001 001 100	RS1	ROR A	
000 001 001 101	RS2	ROR A,2	
000 001 010 000	CIL		
000 001 011 000	CTA		
000 001 100 xxx	SBU		
000 001 101 xxx	STP		
000 001 110 xxx	STE		
0000 10		And immediate	$A \leftarrow A \& E$
0000 11		Or immediate	$A \leftarrow A E$
0001 00		Load	$A \leftarrow E$
0001 01		Load Complement	$A \leftarrow$
0001 10		Add immediate	$A \leftarrow A + E$
0001 11		Subtract immediate	$A \leftarrow A - E$
110x 00		Zero Jump	if x=0 then dst=P+E else dst=P-E

Opcode	Mnemonic	Description	
110x 01		Non-Zero Jump	if x=0 then dst=P+E else dst=P-E
110x 10		Positive Jump	jump if A is positive
110x 11		Negative Jump	jump if A is negative
1110 00		Jump Indirect	$P \leftarrow M[E]$
1110 01		Jump Forward Indirect	
1111 1111 1111		Error	halt the cpu, same as Halt

Addressing modes

No Address (n)

E is used as the lower 6 bits of a 12-bit operand, the upper 6 bits being zeros. No storage reference is made.

Direct Address (d)

E is used to address one of the first 64 storage locations from which a 12-bit operand is obtained.

Indirect Address (i)

E is used to address one of the first 64 storage locations. The 12-bit contents of that location represent another address within storage. This latter address contains the 12-bit operand to be used by the instruction.

Relative Address Forward (f)

E is added to the contents of P. The sum specifies a new address where the operand to be used by the current instruction will be found.

Relative Address Backward (b)

E is subtracted from the contents of P. The resultant difference specifies a new address where the operand to be used by the current instruction will be found; this new address cannot be further than 63 locations from the current instruction. P is unchanged.

Relative Forward Indirect Address (fi)

E is added to the contents of P in order to specify one of the 63 locations immediately following the current instruction address; the 12-bit contents of that location are used to address and operand anywhere on storage. P is unchanged.

Tabulka 85-8. Instruction table for CDC 160

mnemonic	opcode	name	addr mode	exec time	comments
ERR	00	Error		6.4	Stop operation
HLT	77	Halt		6.4	

mnemonic	opcode	name	addr mode	exec time	comments
SHA	01	Shift A	(n)	6.4	Shift A depends on value E for its action. Three possibilities are: <ul style="list-style-type: none"> • 0102 Shift the contents of A left end-around 1 binary position. • 0110 Shift the contents of A left end-around 3 binary positions. • 0112 Multiply the contents of A by octal 12 (decimal 10).
LPN	02	Logical Product	(n)	6.4	Form in A the logical product of original contents of A and operand. $A \leftarrow A \& \text{operand}$
LPD	10		(d)	12.8	
LPI	11		(i)	19.2	
LPF	12		(f)	12.8	
LPB	13		(b)	12.8	
LSN	03				
LSD	14				
LSI	15				
LSF	16				
LSB	17				
OUT	73	Output	(fi)	12.8 + $n \cdot 12.8$	Transmit to selected output device words of information from storage beginning at starting address and continuing up to but not including the termination address.
OTN	74	Output Direct	(n)	6.4	Transmit lower 6 bits (E) of this instruction to selected output device. If more than 6 bits are used by output equipment other bits will be zero.
EXF	75	External Function	(f)	12.8	Transmit external function code to external equipment. The code is specified by contents of storage location E positions following external function instruction.
INA	76	Input to A	(n)	6.4	Read into A one frame of data from selected input device, clearing previous contents of A.

85.2.2.1.

85.2.2.2. Podprogramy

CDC 160 nemá žádnou instrukci pro volání podprogramů. Toto lze ale realizovat následující posloupností instrukcí.

```
// Call p
x : 0100 1000 0000 -- Load Forward Relative
x+1:      x+6      -- (return address)
x+2: 1000 0100 0000 -- Store Indirect
x+3:      p        -- (destitnation return link)
x+4: 1110 0100 0000 -- Jump Forward Indirect
x+5      p+1      -- (destination address)

// Subroutine P
p-1: 1110 0100 0000 -- Jump Forward Relative
p :      -          -- (return address)
p+1:      ....      -- Entry Point of P
      ....          -- (function body)
p+n: 1110 0100 0000 -- Jump Forward Indirect
      :      p-1    -- (address of return jump)
```

85.3. CDC 160A

* *Attributy:* section id="cdc160a"

* *FIXME:*TBD.

Odkazy:

- CDC 160A¹⁴

85.3.1. Instrukční sada

*

Tabulka 85-9.

Opcode	Mnemonic	Description	Timing
0000	ERR	Error Stop	1
01..		Fixed Shift	
02..		And	
03..		Or	
04..		Load	
05..		Load complement	
06..		Add	
07..		Subtract	
10..		And (direct)	
11..		And (indirect)	
12..		And (relative forward)	
13..		And (relative backward)	
14..		Or (direct)	
15..		Or (indirect)	
16..		Or (relative forward)	
17..		Or (relative backward)	

F	E	G	Mnemonic	Name	Timing
Stop Commands					
001x			Shift relative (memory reference control to Bank x)	1	
Shift Commands					
01	EE			Fixed Shift	
02	EE			And	
03	EE			Or	
04	EE			Load	
05	EE			Load complement	
06	EE			Add	
77	00		HLT	Halt	1
77	77		HLT	Halt	1
Data Transmission Commands					
01	00	XXXX	BLS	Block Store	1-2
01	14		RS1	Right Shift One	1
01	15		RS2	Right Shift Two	1

85.4. ARC-1 — The Average Response Computer

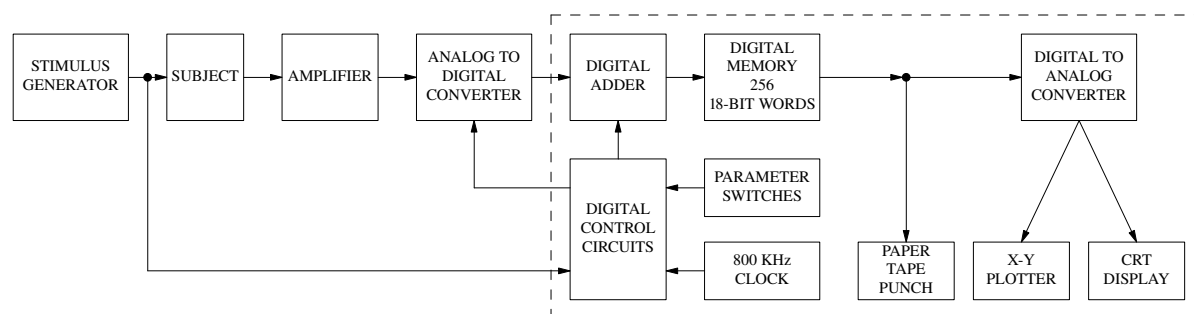
* *Attributy: id="ARC-1"*

Odkazy:

- The Genesis of a Technological Revolution¹⁵
-

Nástroj který vyvinul W.A.Clark (Lincoln Laboratory) na Center Development Office for Computer Technology in the Biomedical Sciences pro Mary A.B. Brazier. ARC-1 není univerzální počítač, ale specializovaný stroj pro neurofiziologické experimenty. Tento přístroj byl vyroben v Lincoln Laboratory, byl postaven z digitálních transistorových obvodů a byl dostatečně rychlý pro zpracování experimentálních dat v reálném čase.

Obrázek 85-4. Blokové schéma ARC-1



85.5. LINC [1962]

* *Attributy: id="LINC", ent=LINC*

Odkazy:

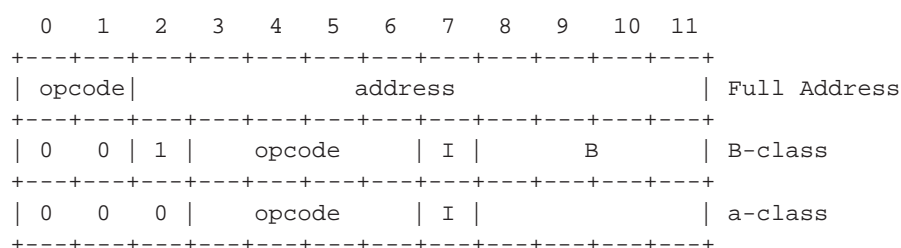
- LINC Computer¹⁶
- LINC MODE PROGRAMMING¹⁷ from PDP-12 User's Manual
- The LINC was early and small¹⁸ ACM PORTAL
- WES CLARK LECTURE NOVEMBER 18. 1981¹⁹
- The Genesis of a Technological Revolution²⁰
-
-

Fakta:

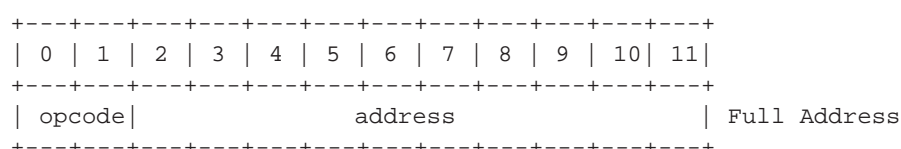
- Postaven v roce 1962 na MIT.
- Přímí předchůdci: TX-0, ARC-1 a L-1
- Konstrukce LINC ovlivnila PDP-4, a PDP-5.
- Bylo postaveno přes 1200 kusů.
-
-

LINC je zkratka pro Laboratory Instrument Computer.

Obrázek 85-5. Formáty instrukcí LINC-8



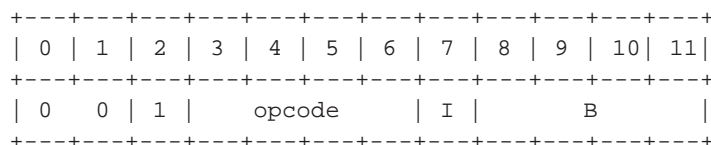
Obrázek 85-6. Direct Address Instruction Format of LINC



Tabulka 85-10. Základní instrukce (Full Address) LINC

kód	název	popis
00		Instrukce třídy B, C, ...
01y yyy yyy yyy	ADD Y	ACC+mem[Y]→ACC
10y yyy yyy yyy	STC Y	ACC→mem[Y]; 0→ACC
11y yyy yyy yyy	JMP Y	PC→mem[0]; Y→PC

Obrázek 85-7. iβ



Tabulka 85-11. Základní instrukce LINC

kód	název	popis
	ADD	
	STC	
	JMP	

Pokud B není 0 tak je to adresa jednoho z 15 B registrů. Ty jsou umístěny v paměti od adresy 0001 do 0017 (oktalově).

Tabulka 85-12.

0	00	adresa je čtena ze slova následujícího za instrukcí (absolutní mód)
1	00	data jsou čtena ze slova následujícího za instrukcí (immediate mode)
0	01-17	adresa je čtena z B registru
1	01-17	B registr je zvětšen o jedničku a je z něj přečtena adresa

85.6. L-1

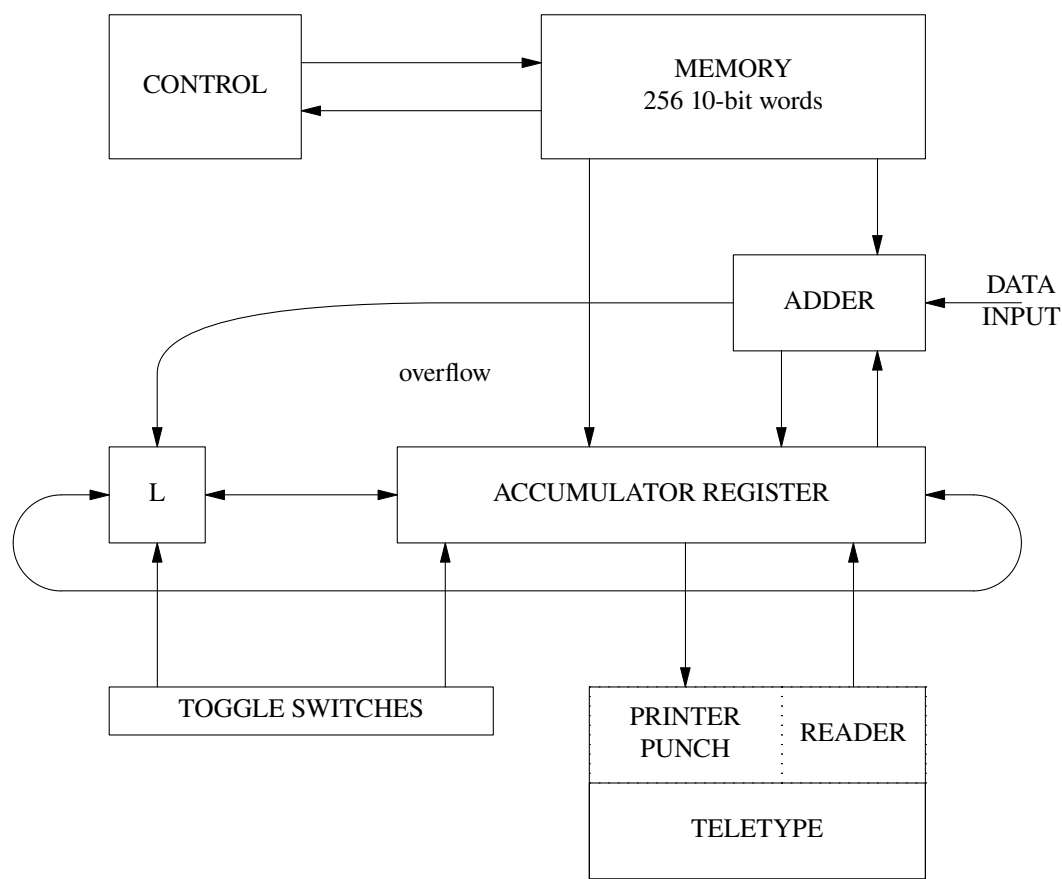
* *Attributy: id="L-1", ent=L-1*

Odkazy:

- A FUNCTIONAL DESCRIPTION OF THE L-1 COMPUTER²¹

Počítač postavený okolo roku 1960 v Lincoln Laboratory v Massachusetts Institute of Technology. Počítač je velmi malý, bez vstupně výstupního elektronického psacího stroje zabírá 4 kubické stopy. Vykoná až 80 tisíc 10-ti bitových operací za sekundu. Jeho instrukční sada vychází z konstrukce TX-0

Obrázek 85-8. Blokové schéma L-1



0	1	2	3	4	5	6	7	8	9																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
---	---	---	---	---	---	---	---	---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

JP adresa
AD adresa ($A \leftarrow A + M$)
ST adresa ($A \rightarrow M_{adresa}; 0 \rightarrow A$)

Obrázek 85-9. Instrukce L-1

JP adresa	0 0	adresa									
	0	1	2								9
AD adresa	0 1	adresa									
	0	1	2								9
ST adresa	1 0	adresa									
	0	1	2								9
operate	1 1	CL	CP	SR	SL	RS	SA	PR	RT		
	0	1	2	3	4	5	6	7	8	9	

Data se do počítače dostávají třemi kanály, dálnopisem s 5-ti stopou děrnou páskou, přepínači na předním panelu a vstupním datovým kanálem.

Výstup je realizován derovačem 5-ti stopé děrné pásky dálnopisným strojem.

Výpočet začíná spuštěním stroje v *load* módu. V tomto se načte prvních 32 znaků z děrné pásky a uloží do prvních 16-ti slov operační paměti. Poté je spuštěn právě načtený program od adresy 0.

Tabulka 85-13.

code	mnemo	name	description
00 xxxxxxxx	jp	jump	if $A_0=0$ then $x \rightarrow P$ else $P+1 \rightarrow P \pmod{256}$
01 xxxxxxxx	ad x	add	$A+M_x \rightarrow A$; Overflow $\vee L \rightarrow L$
10 xxxxxxxx	st x	store	$A \rightarrow M_x$; $0 \rightarrow A$
11 10000000	cl	clear	$0 \rightarrow A$; $0 \rightarrow L$
11 01000000	cp	complement	$\neg A \rightarrow A$
11 00100000	sr	shift right	$A_i \rightarrow A_{i+1} [i=0,1,\dots,8]$; $A_9 \rightarrow L$; $L \rightarrow A_0$
11 00010000	sl	shift left	
11 00001000	rs	read switches	$T \vee A \rightarrow A$; $T_{10} \vee L \rightarrow L$
11 00000100	sa	sample	$A+Y \rightarrow A$; Overflow $\vee L \rightarrow L$
11 00000010	pr	print	$A_{5-9} \rightarrow P$; 111111111 $\rightarrow A$ 1 $\rightarrow L$
11 00000001	rt	read tape	$R \rightarrow A_{0-4}$; $R \rightarrow A_{5-9}$

85.7. Control Data Corporation Model 1604

Odkazy:

- Control Data Corporation Model 1604²²
-

Velký (sálový) počítač. Používal 48-mi bitová slova. Instrukce byly 24 bitové. Do jednoho slova se tedy vešly dvě. Paměť s feritových prstýnků měla 32K slov po 48 bitech.

85.8. Eliot

Odkazy:

- The first computer I programmed²³
- Elliot 803²⁴ na Wikipedii
-
-
-
-

85.9. Další počítače

Odkazy:

-

•

85.9.1. IBM 1130

Odkazy:

- IBM1130.org²⁵

Paměť počítače byla 4 až 32K 16-ti bitových slov

Poznámky

1. http://en.wikipedia.org/wiki/List_of_transistorized_computers
2. http://www.computerhistory.org/projects/zuse_z23/
3. <http://www.bitsavers.org/pdf/mit/tx-0/>
4. <http://www.bitsavers.org/bits/MIT/tx-0/>
5. <http://www.businessweek.com/chapter/segaller.htm>
6. <http://www.cs.man.ac.uk/CCS/res/res04.htm#f>
7. <http://www.absoluteastronomy.com/topics/TX-0>
8. <http://ed-thelen.org/comp-hist/TheCompMusRep/TCMR-V08.html>
9. http://www.bitsavers.org/pdf/mit/tx-0/MIT_TX-0_InstructionSet.txt
10. <http://www.cs.uiowa.edu/~jones/cdc160/>
11. <http://ed-thelen.org/comp-hist/BRL61-c.html#CDC-160>
12. <http://ed-thelen.org/comp-hist/CDC-160-A-prel.html>
13. http://en.wikipedia.org/wiki/CDC_160A
14. <http://ed-thelen.org/comp-hist/cdc-160a.html>
15. http://history.nih.gov/exhibits/linc/docs/page_06.html
16. http://www.smecc.org/linc_computer.htm
17. <http://users.rcn.com/crfriend/museum/doco/PDP-12/UG-Chap3s3.html>
18. <http://portal.acm.org/citation.cfm?id=12178.12187>
19. <http://ed-thelen.org/comp-hist/lecture-Wes-Clark3.html>
20. <http://history.nih.gov/exhibits/linc/index.html>
21. http://www.bitsavers.org/pdf/mit/lincolnLaboratory/51G-0012_Functional_Description_of_the_L1_Computer_Mar60.pdf
22. <http://ed-thelen.org/comp-hist/BRL61-c.html#CDC-1604>
23. <http://www.g6lvb.com/803.htm>
24. http://en.wikipedia.org/wiki/Elliott_803
25. <http://ibm1130.org>

Kapitola 86. Doba mikroprocesorová

S příchodem mikroprocesorů se podstatně snížila velikost cena i dostupnost počítačů. Počítače přesunuli z oblasti exklusivních strojů zabírajících celé místnosti nebo alespoň dostatečně veliké či malé skříně do oblasti malých, jednodeskových strojů s masovým nasazením.

86.1. I8008

První 8-mi bitový mikroprocesor. Vyroben firmou Intel. Jedná se o předchůdce procesoru 8080. Je to jeden z prvních mikroprocesorů o kterém vím že byl nasazen v dostupné počítačové technice.

Počítače postavené na I8008:

- - **SCELB-8H [1973]:**
 - SCELB¹ na Wikipedii
 -
- - **MARK-8 [1974]:**
 - Mark-8² na Wikipedii
 - Mark-8 Minicomputer³

86.2. RCA1802

Počítače postavené na RCA1802:

- - **COSMAC MICROTUTOR [1976]:**
 - RCA COS/MAC MICROTUTOR⁴
 -
 -
- - **COSMAC ELF [1976,1977]:**
 - COSMAC ELF⁵
 - COSMAC ELF⁶ na Wikipedii
 - Elf 2000⁷
 - Build The COSMAC "ELF" A Low-Cost Experimenter's Microcomputer⁸ v Popular Electronics, srpen 1976
 -
 -
- - **COSMAC VIP [1977]:**
 - COSMAC VIP⁹ na Wikipedii
 - RCA Cosmac VIP¹⁰ na OLD-COMPUTERS.COM

*

86.3. I8080

* *TEMPLATE*

86.4. Z80

Počítače a mikropočítače s procesorem Z80. Zde jsou uvedeny jen komerčně vyráběné a prodávané, nebo velmi významné historické konstrukce. Pokud se zajímáte a aktuální konstrukce, ty jsou uvedeny v kapitole o mikroprocesoru Z80.

Odkazy ZX Spectrum a klony:

- 33
- Didaktik Gama 192KB¹¹ Nový Klon ZX Spectra 128KB
- PENTAGON-1024 SL¹²
- Sinclair ZX Spectrum FAQ¹³ —
- comp.sys.sinclair FAQ¹⁴ —
- Sprinter¹⁵ — stránka je v ruštině
-

Odkazy Sinclair ZX mix a různé:

- Computer case which is worth considering¹⁶
- Sinclair User Archive¹⁷
- ZX MMC¹⁸
- Sinclair User¹⁹
- CRASH²⁰
- Postavte si doma pořádný počítač!²¹
-

Fóra:

-
- WaveMate_Computers²²

Simulátory:

- Z80-SIM²³
-

86.4.1. Sinclair ZX80/ZX81 a klony

* *Attributy: id="ZX80"*

Odkazy ZX80:

- ZX80²⁴
- Another Sinclair ZX80 page!²⁵
- planet sinclair²⁶
- Sinclair ZX80²⁷ na oldcomputers.net
- ZX81 RULES OK!²⁸
-

Odkazy ZX81:

- ZX-Team²⁹
- Grafika na osmibitových počítačích firmy Sinclair³⁰ Pavel Tišnovský [2009-06-25]
- ZX81 / TIMEX 1000/1500 COMPETITIONS³¹
- ZX81 MMC Redux³²
- La Page de Didier, Club Zx81³³
- ORDI-5³⁴ [francouzsky]
- ZX81.de — ZX-TEAM-homepage³⁵
-
-

ZX80 je první počítač vyrobený firmou Cliva Sinclaira v Británii. Byl to první mikropočítač s nízkou cenou pro velmi široké maso. Jeho konstrukce postavená okolo procesoru Z80 je velmi jednoduchá. O videesignál je generován procesorem jen s minimální podporou hardware. Celá elektronika se vešla na jednu malou PCB.

ZX81 je vylepšená verze ZX80. Hlavním rozdílem je, že místo většiny obvodů byl použit zákaznický obvod ULA vyrobený firmou Feranti.

Fóra:

- Sinclair ZX80 / ZX81 Forums³⁶ — Discussion forums for users of the Sinclair ZX80 / ZX81 and their clones
- forum.tlienhard.com³⁷ — Das Forum für ZX81 und Amiga Freunde [německy]
-
-
-

Klony:

- TL801 italská replika ZX80
- Microace americká replika ZX80
-

86.4.1.1. Grant Searle ZX80

*

Odkazy:

- How to build your own ZX80/ZX81 [2010-01-22]³⁸
-

86.4.2. ZX Spectrum

Odkazy:

- ZX Design Info³⁹ — A site dedicated to the reverse engineering of the ZX Spectrum and related projects
- World of Spectrum Forums⁴⁰

86.5. MC6800

* *TEMPLATE*

86.6. 6502

* *TEMPLATE*

86.7. Altair

Odkazy:

- Welcome to AltairKit.com! Home of the New Altair 8800 Kit! ⁴¹
-

Altair 8080, původní, byl osazen procesorem 8080. Později byly zkonstruovány procesorové desky s jinými procesory. Proto jej uvádím zde a ne u procesoru 8080.

86.8. IMSAI

Odkazy:

- The Official Home Page of IMSAI⁴²
- IMSAI Documentation⁴³
-
-

Poznámky

1. <http://en.wikipedia.org/wiki/SCELBI>
2. <http://en.wikipedia.org/wiki/Mark-8>
3. http://www.bytecollector.com/mark_8.htm
4. <http://www.decodesystems.com/cosmac/index.html#microtutor>
5. <http://homepage.mac.com/ruske/cosmacelf/>
6. http://en.wikipedia.org/wiki/COSMAC_ELF
7. <http://www.sparetimegizmos.com/Hardware/Elf2K.htm>
8. http://incolor.inebraska.com/bill_r/elf/html/elf-1-33.htm
9. http://en.wikipedia.org/wiki/COSMAC_VIP
10. <http://www.old-computers.com/museum/doc.asp?c=543>
11. <http://electronics.mysteria.cz/gama192cz/index.htm>
12. <http://pentagon.nedopc.com/>
13. <http://www.nvg.ntnu.no/sinclair/faq/index.html>
14. <http://www.sinclairfaq.com/cssfaq/>
15. <http://winglion.ru/sprinter/>
16. <http://www.sincuser.f9.co.uk/023/hardwre.htm>
17. <http://www.sincuser.f9.co.uk/index.htm>
18. <http://www.robsonfamily.dsl.pipex.com/mmc/mmc.htm>

19. <ftp://ftp.worldofspectrum.org/pub/sinclair/magazines/SinclairUser/SinclairUserViewer.html>
20. <http://www.crashonline.org.uk/>
21. <http://www.mcu.cz/news.php?extend.1540.7>
22. http://groups.google.com/group/wavemate_computers
23. <http://www.unix4fun.org/z80pack/index.html>
24. http://www.zx81.de/english/zx80_e.htm
25. <http://www.xs4all.nl/~fjkraan/comp/zx80/>
26. <http://www.nvg.org/sinclair/computers/zx80/zx80.htm>
27. <http://www.oldcomputers.net/zx80.html>
28. <http://www.user.dccnet.com/wrigter/>
29. <http://www.fischerkai.de/zxteam/>
30. <http://www.root.cz/clanky/grafika-na-osmibitovych-pocitacich-firmy-sinclair/>
31. <http://zx81.republika.pl/>
32. <http://arduinoanut.blogspot.com/2010/02/zx81-mmc-redux.html>
33. <http://zx81.ordi5.free.fr/didier/>
34. <http://zx81.ordi5.free.fr/>
35. <http://www.zx81.de/>
36. http://www.rwapservices.co.uk/ZX80_ZX81/forums/
37. <http://forum.tlienhard.com/phpBB3/index.php>
38. <http://home.micros.users.btopenworld.com/zx80/zx80.html>
39. <http://www.zxdesign.info/>
40. <http://www.worldofspectrum.org/forums/index.php>
41. http://www.altairkit.com/creation_of_a_kit_story.html
42. <http://www.imsai.net/>
43. <http://www.hartetechnologies.com/manuals/IMSAI/>

Kapitola 87. Nezařazené stroje

*

Do této kapitoly umísťnuji popisy počítačů které jsem zatím nezařadil do předchozích kapitol.

87.1. Philco 212

*

Obrázek 87-1. Formáty dat a instrukcí Philco 212

FIXED-POINT DATA WORD	integer part											fraction part																								
FLOATING-POINT DATA WORD	mantisa																																			
BCD OR ALPHANUMERIC WORD	char					char					char					char					char					char										
	0 5 6					11 12					17 18					23 24					29 30					35 36										
.sp 1																																				
INSTRUCTION WORD	LEFT INSTRUCTION																							RIGHT INSTRUCTION												
:	ADDRESS FIELD															COMMAND								ADDRESS FIELD												
INPUT-OUTPUT ORDER WORD												NRS				TC		UNIT				NWR													NI	
	0 11 12 15											16 17		18 19		23 24				35 36																
ADDRESS-FIELD	0	V																																		
1																																				
	N	V																																		
3																	4																			
	15																																			

Tabulka 87-1. Instrukce

mnemonic	code ₄	popis
AM	1000	Add Memory
FAM	3000	
AMS	1001	Add memory and Store
FAMS	3001	
	0000	
	0000	

Kapitola 88. Historie na videu

*

Triumph of the Nerds:

- 1 - 1¹
- 1 - 2²
- 1 - 3³
- 1 - 4⁴
- 1 - 5⁵
- 1 - 6⁶
- 2 - 1⁷
- 2 - 2⁸
- 2 - 3⁹
- 2 - 4¹⁰
- 2 - 5¹¹
- 2 - 6¹²
- 3 - 1¹³
- 3 - 2¹⁴
- 3 - 3¹⁵
- 3 - 4¹⁶
- 3 - 5¹⁷
- 3 - 6¹⁸

How the Altair 8800 started the PC revolution:

- 1¹⁹
- 2²⁰

Pirates of silicon valley:

- 1²¹
- 2²²
- 3²³
- 4²⁴
- 5²⁵
- 6²⁶
- 7²⁷
- 8²⁸
- 9²⁹
- 10³⁰
- Steve Wozniak talks about Pirates of Silicon Valley³¹

Nerds 2.0.1

- episode 1 part 1³²
-

Poznámky

1. <http://www.youtube.com/watch?v=3jV3JdtaOGc>
2. <http://www.youtube.com/watch?v=k4SHjp0Z-7M>
3. <http://www.youtube.com/watch?v=Dc5-2unzD9A>
4. <http://www.youtube.com/watch?v=pyKlNjwR03M>

5. <http://www.youtube.com/watch?v=hZnZvOxg8Ks>
6. <http://www.youtube.com/watch?v=xi-g0ievM-4>
7. <http://www.youtube.com/watch?v=rKw3KM3MmLo>
8. <http://www.youtube.com/watch?v=viANWOeGc1I>
9. <http://www.youtube.com/watch?v=hvun38EreIE>
10. <http://www.youtube.com/watch?v=wBGJJdCnJMw>
11. <http://www.youtube.com/watch?v=R8xIdkw6Zvk>
12. <http://www.youtube.com/watch?v=lKk5a6hgbgM>
13. <http://www.youtube.com/watch?v=qrnMgBBfNI>
14. <http://www.youtube.com/watch?v=Yp6RMfYcrH0>
15. <http://www.youtube.com/watch?v=enZ4q96TQLE>
16. <http://www.youtube.com/watch?v=Sil44rEFzMc>
17. <http://www.youtube.com/watch?v=tL9mujB-L1A>
18. http://www.youtube.com/watch?v=D_QEso4nZ0c
19. <http://www.youtube.com/watch?v=WVwfDA4watU>
20. http://www.youtube.com/watch?v=pfRBwBP_uTA
21. <http://www.youtube.com/watch?v=xfIXMZL2stU>
22. <http://www.youtube.com/watch?v=sHM82fF2ta8>
23. <http://www.youtube.com/watch?v=BCgB5fITxA4>
24. http://www.youtube.com/watch?v=ZYDL_ZHMdvU
25. <http://www.youtube.com/watch?v=1ZU2ncF2NFw>
26. <http://www.youtube.com/watch?v=DcWjOodAtoE>
27. <http://www.youtube.com/watch?v=wIeSqEBGNHM>
28. <http://www.youtube.com/watch?v=J9GGdsepMGU>
29. <http://www.youtube.com/watch?v=4Vw0QVT40CM>
30. <http://www.youtube.com/watch?v=6DstZQIJx-o>
31. <http://www.youtube.com/watch?v=1lx9JsSTklI>
32. <http://www.youtube.com/watch?v=QpAlKeukMys>

Kapitola 89. Osobnosti

* *Attributy: id="osobnosti"*

* *Kapitola generovaná z databáze. Část informací je zde ručně.*

Jména a krátké informace o lidech jenž dělali počítače.

89.1. Howard Aiken

*

Pracoval s Grace Hopper na počítači MARK I (ASCC).

Příslušník armády.

89.2. Sergei Alexeevich Lebedev (1902-1974)

- Sergei Alexeevich Lebedev¹

Pod jeho vedením bylo vytvořeno 15 elektronických počítačích strojů.

V roce 1945 byl součástí týmu který vytvořil první analogový počítač v Sovětském Svazu.

Podílel se na stvoření počítačů MESM, SESM, BESM-2, M-20, BESM-6

89.3. David L. Jones (*19xx)

Odkazy:

- YouTube: EEVblog²
- EEVblog Electronics Community Forum³
- EEVBlog⁴ — Electronics Engineering Video Blog; No script, no fear, all opinion.

David L. Jones, známý také jako Dave Jones je Australský elektronik známý svým videoblogem o elektronice na YouTube.

Doporučené knihy:

- iWoz⁵
- High-Speed Signal Propagation⁶ Advanced Black Magic, Howard Johnson, Martin Graham
- High-Speed Digital Design⁷ (Handbook of Black Magic)
- The Scientist and Engineer's Guide to Digital Signal Processing⁸ By Steven W. Smith, Ph.D. (na webu zdarma jako pdf)
- The Art of Electronics⁹ Paul Horowitz, Winfield Hill
-
-

Doporučené weby:

- FindChips.com¹⁰
- ANT the power of less¹¹
-

•

89.4. Grace Hopper (*1906, †1992)

Odkazy:

- Wikipedia^{EN}: Grace Hopper¹²
- Wikipedie^{CZ}: Grace Hopperová¹³
- Grace Murray Hopper¹⁴
-

Grace Murrayová Hopperová, narozená 9. prosince 1906, zemřela 1. ledna 1991. Byla americká matematicka, počítačová vědkyně a důstojnice amerického námořnictva.

Pracovala spolu s Howardem Aikenem na počítači MARK I (ASCC). Zaměstnána u námořnictva USA (USA Navy).

Učila matematiku. V té době byla výuka matematiky (classified occupation)

Autorka prvního manuálu k počítači, ASCC operational manual¹⁵.

89.5. Hans R.Camenzind (*1934)

Odkazy:

- Hans R. Camenzind¹⁶
-

Hans R. Camenzind se narodil v Zurichu ve Švýcarsku v roce 1934. Vystudoval na elektro inženýra v North-eastern University¹⁷ a MBA na Santa Clara University¹⁸. Po několika letech práce v okolí Bostonu (USA) se přestěhoval na západní pobřeží a nastoupil do firmy Signetics¹⁹ a později založil vlastní firmu, Interdesign. Tu po několika letech prodal Ferranti²⁰. Od té doby pracuje jako nezávislý konsultant v navrhování analogových integrovaných obvodů.

Během své kariéry napsal tři učebnice a navrhl první integrovaný zesilovač ve třídě D. Implementoval koncept fázového závěsu do integrovaného obvodu, navrhl několik zákaznických integrovaných obvodů a stvořil známý 555. Do roku 2006 navrhl celkem 140 standardních a zákaznických integrovaných obvodů.

89.6. Harlan Anderson

Zakladatel firmy DEC. Tu založil spolu s Kenem Olsonem v roce 1957.

89.7. Jeri Ellsworth

Odkazy:

- Jeri Ellsworth²¹ na Wikipedii
- Jeri Ellsworth Dot Com²²

POPIS

89.8. Ken Olson

Zakladatel firmy DEC. Tu založil spolu s Harlanem Andersonem v roce 1957.

Olson s Andersonem pracovali v Lincoln Labs kde navrhli hlavní části AN/FSQ-7, TX-0 a TX-2. Chtěli mít vlastní firmu na vývoj a výrobu počítačů. Proto založili DEC.

89.9. Saul Dinman

Odkazy:

- <http://www.cs.uiowa.edu/~jones/arch/risc/>
-

Architekt PDP-8/S. Poté co opustil DEC, založil GRI Computer Corporation. Zkonstruoval počítač GRI-909.

89.10. Seymour Cray

Autor řady počítačů a superpočítačů.

- CDC-160
- CDC-1604
-

89.11. Stave Wozniak (*1950-08-11)

Odkazy:

- Woz.Org²³
- iWoz²⁴

Steve Wozniak (Woz) je konstruktér počítačů Apple I, Apple II a Apple III. Je jeden ze zakladatelů Apple Computer Inc.

Napsal knihy:

- iWoz – Steve Wozniak muž, který postavil první osobní počítač²⁵
-

Doporučené weby:

-
-

89.12. William Friedman

Americký kryptoanalytik.

Poznámky

1. http://www.thocp.net/biographies/lebedev_sergei.html
2. <http://www.youtube.com/user/EEVblog#p/u/187/u2yRR4G3yTA>
3. <http://www.eevforum.com/>
4. <http://www.eevblog.com/>
- 5.
- 6.
- 7.
8. <http://dspguide.com/>
- 9.
10. <http://www.findchips.com/>
11. <http://thisisant.com/>
12. http://en.wikipedia.org/wiki/Grace_Hopper
13. http://cs.wikipedia.org/wiki/Grace_Hopper
14. http://www.thocp.net/biographies/hopper_grace.html
15. http://www.bitsavers.org/pdf/harvard/MarkI_operMan_1946.pdf
16. http://en.wikipedia.org/wiki/Hans_R._Camenzind
17. http://en.wikipedia.org/wiki/Northeastern_University,_Boston
18. http://en.wikipedia.org/wiki/University_of_Santa_Clara
19. <http://en.wikipedia.org/wiki/Signetics>
20. <http://en.wikipedia.org/wiki/Ferranti>
21. http://en.wikipedia.org/wiki/Jeri_Ellsworth
22. <http://www.jeriellsworth.com/>
23. <http://www.woz.org>
24. <http://www.iwoz.org>
- 25.

Kapitola 90. Staré časopisy

Zde uvádím seznam historických časopisů věnovaných elektronice a výpočetní technice o jejichž existenci aspoň vím. Tam kde to jde uvádím odkazy na internetové archivy.

Creative Computing

Časopis byl publikován od roku 1974 až do roku 1985. Byl orientován na hobby a věnoval se všem platformám. S časopisem vyšla také rozsáhlé přílohy „The Best of Creative Computing“.

Odkazy:

- Creative Computing Magazine¹ na ATARIARCHIVES.ORG
- Creative Computing² na Wikipedii

Popular Electronics

Circuit Cellar

Nuts and Volts

Odkazy:

- Nuts and Volts³ the magazine for the electronics and hobbyist

Poznámky

1. <http://www.atarimagazines.com/creative/>
2. http://en.wikipedia.org/wiki/Creative_Computing
3. <http://www.nutsvolts.com>

Kapitola 91. Operační systémy a softwarové vybavení

*

91.1. OS-9/6809

*

Odkazy:

- NitroOS-9¹ OS-9 Level II pro Tandy CoCo3
- RTSI OS9 World Wide Archive²
-
-

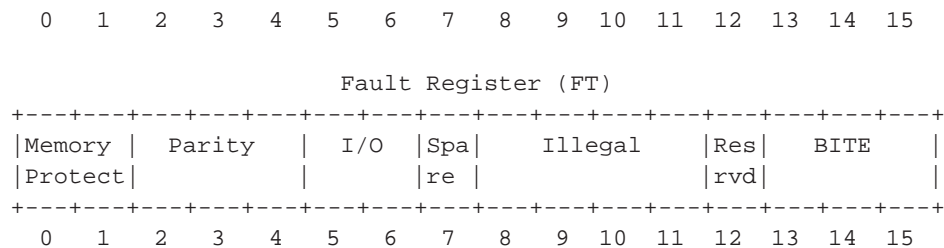
91.2. FLEX

Odkazy:

- FLEX Documentation³ na FLEX User Group
- The Missing 6809 UniFLEX Archive⁴

Poznámky

1. http://sourceforge.net/apps/mediawiki/nitros9/index.php?title=Main_Page
2. <http://www.rtsi.com/wwa.html>
3. <http://www.evenson-consulting.com/flexusergroup/fug4.htm>
4. <http://www.rtmx.com/UniFLEX/>

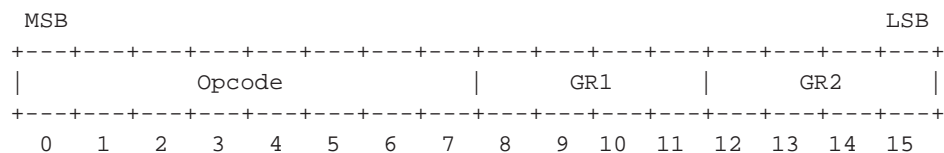


92.1.1. Istrukce MIL-STD-1750A

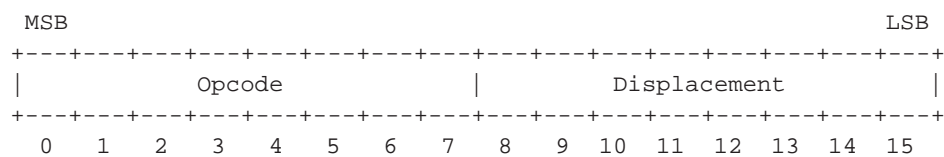
Odkazy:

- 4.2. Instruction Formats¹⁰
-
-

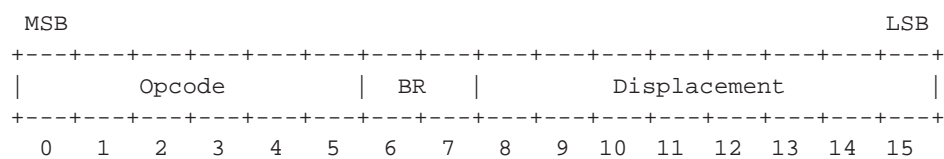
Obrázek 92-2. 1750A Register-to-Register Format



Obrázek 92-3. 1750A Counter Relative Format

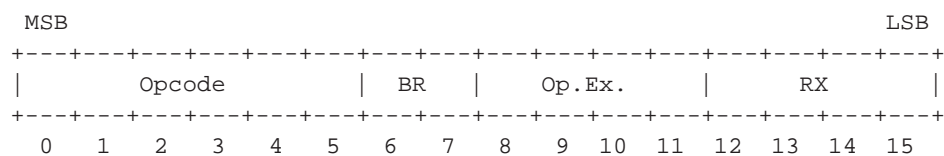


Obrázek 92-4. 1750A Base Relative Format

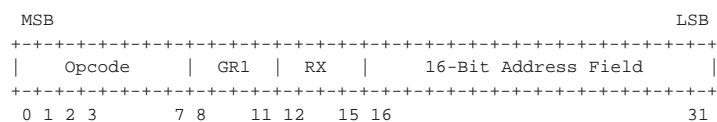
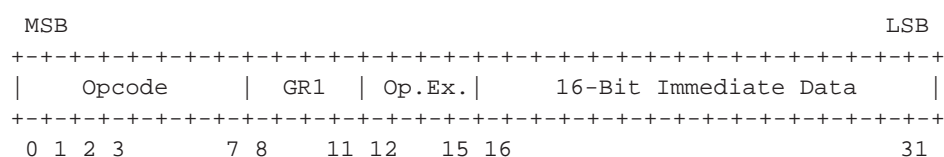
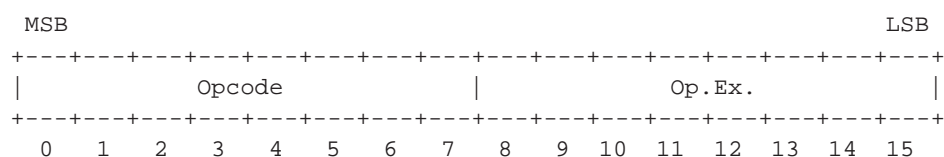


BR = 0 general register 12
 BR = 1 general register 13
 BR = 2 general register 14
 BR = 3 general register 15

Obrázek 92-5. 1750A Base Relative Indexed Format



BR = 0 general register 12
 BR = 1 general register 13
 BR = 2 general register 14
 BR = 3 general register 15
 RX = 0 no indexing

Obrázek 92-6. 1750A Long Instruction Format**Obrázek 92-7. 1750A Immediate Opcode Extension Format****Obrázek 92-8. 1750A Special Format**

92.2. IBM AP-101

Odkazy:

- IBM AP-101¹¹ na Wikipedii
-
-
-

*

Fakta:

- Jeden z nejmodernějších čelnů rodiny IBM System/4 Pi
- Architektura IBM 360
- Mikroprogramový
- Původně 16-ti bitová adresa, později rozšířena na 20 bitů. A tedy schopnost adresovat 1MB operační paměti.
- Použit zejména v raketoplánech, stíhačkách F15
-
-
-

92.3. Magic I

Odkazy:

- magic¹² na bitsavers.org

Počítač určený pro použití v letecké a raketové technice. Sériové zpracování data → pomalý. Paměť 4096 24-bitových slov. Instrukce 12 bitů. V jednom slově jsou dvě. Při konstrukci jsou použity obvody Micrologic firmy Fairchild. Toto jsou jedny z prvních integrovaných logických obvodů. Zapouzdření je v kulatých (tranzistorových) pouzdrech s více vývody (8 vývodů).

92.4. TRADIC

Odkazy:

- The Industrial Era 1952 - 1954¹³
-

Tento tranzistorový počítač, postavený firmou Bell Telephone Laboratories v roce 1953, byl instalován v bombardovacím letadle B52. Počítač obsahoval asi 800 tranzistorů.

Poznámky

1. http://klabs.org/history/build_agc/
2. <http://web.mit.edu/slava/space/introduction.htm>
3. <http://en.wikipedia.org/wiki/1750A>
4. <http://www.computer-museum.ru/english/argon17.htm>
5. http://www.teledyne-controls.com/newscenter/2004/adl-adc_lokcheed_martin.asp
6. http://www.aviationtoday.com/av/issue/feature/Russian-Airborne-Computers_12840.html
7. <http://www.solarstorms.org/SEUcomputers.html>
8. <http://www.xgc.com/manuals/m1750-ada/m1750/book1.html>
9. <http://en.wikipedia.org/wiki/JOVIAL>
10. <http://www.xgc.com/manuals/m1750-ada/m1750/x524.html>
11. http://en.wikipedia.org/wiki/IBM_AP-101
12. http://www.bitsavers.org/pdf/ac_delco/magic/
13. <http://www.thocp.net/timeline/1952.htm>

Kapitola 93. Počítače ve zbraních (raketách, bombách, ...)

*

Odkazy:

-
-

93.1. Minuteman-I Autonetics D-17 flight computer

*

Odkazy:

- LGM-30 Minuteman¹
-

Tento palubní/letový počítač používal jako paměť programu rotující magnetický disk. Program byl uložen v 2560 slovech na 20-ti stopách. Slova měla velikost 24 bitů. Po nahrání programu se deaktivovaly zápisové hlavy. Disk také obsahoval jednu modifikovatelnou stopu s 128 slovy. Otáčka disku trvala 10ms, a okolo toho se točilo veškeré zpracování data.

Palubí počítač byl umístěn v prstenci těsně za hlavicí. Podle obrázků byl realizován z diskretních komponent (tranzistorů a pasivních součástek) na deskách s plošnými spoji.

93.2.

Odkazy:

- Střelecký počítač²
-

Poznámky

1. http://en.wikipedia.org/wiki/LGM-30_Minuteman
2. http://en.wikipedia.org/wiki/M18_FADAC

Kapitola 94. Úvod

FIXME: napsat úvod.

94.1. Zdroje informací

Tento dokument je sestaven výlučně z informací posbíraných po Internetu. Se zde uvedenými stroji jsem se až na výjimky nikdy nepotkal a nebiděl je přímo v provozu.

Odkazy:

- Buletin Resurrection¹ of Computer Conservation Society
- Computers²
- BitSavers³
- The Virtual Museum of Computing⁴ (VMoC)
- Our Computer Heritage⁵ — historie britské výpočetní techniky
-
-
-

Odkazy:

- Academician Lebedev⁶
- Sergey Lebedev⁷

94.2. Nezatříděné a nezpracované informace

Odkazy:

- www.computerconservationsociety.org/media.htm
- Remington-Rand Presents the Univac⁸
- Man & Computer - IBM 1965⁹
- ibm 305 rama¹⁰
- Computer History - A British View - Part 1 of 3¹¹
-
-

Poznámky

1. <http://www.cs.man.ac.uk/CCS/res/>
2. <http://www.quadibloc.com/comp/compint.htm>
3. <http://bitsavers.org/>
4. <http://vmoc.museophile.org/>
5. <http://www.ourcomputerheritage.org/>
6. <http://www.ipmce.org/about/history/lebedev/>
7. <http://www.icfcst.kiev.ua/MUSEUM/LEBEDEV/Lebedev.html>
8. <http://www.youtube.com/watch?v=j2fURxbdIZs>

9. <http://www.youtube.com/watch?v=BUCZJWo9MZo>
10. <http://www.youtube.com/watch?v=zOD1umMX2s8>
11. <http://www.youtube.com/watch?v=nXFnCD72JpY>

Kapitola 95. Velmi staré počítače

95.1. Burroughs B-200

Odkazy:

- Burroughs B-200¹

95.2. SAGE

Odkazy:

- Cold War Computing - The SAGE System² — YouTube video
- INTRODUCTION³

V rámci SAGE byly vytvořeny a nasazeny počítače:

- Whirlwind
- Whirlwind II
- AN/FSQ-7

95.3. Ostatní počítače

Odkazy:

- The IBM 1620 Data Processing System⁴ 1959-1970
- IBM 610 Autopoint Computer⁵
- Soviet Computer URAL2⁶
- UNIVAC - Then and Now⁷
-
-

Poznámky

1. http://www.smecc.org/burroughs_b-200.htm
2. <http://www.youtube.com/watch?v=06drBN8nIWg>
3. <http://ed-thelen.org/SageIntro.html>
4. <http://www.columbia.edu/acis/history/1620.html>
5. <http://ed-thelen.org/comp-hist/BRL61-ibm06.html#IBM-610>
6. <http://www.youtube.com/watch?v=LChE-9fg7c8>
7. <http://www.youtube.com/watch?v=h4wQJfdhOIU>

Kapitola 96. Minipočítače

96.1. Interdata

Interdata, Inc. was a computer company, founded in 1966 and based in Oceanport, New Jersey, that produced a line of 16- and 32-bit minicomputers. They were loosely based on the IBM 360 architecture.

Odkazy:

- [Interdata 7/32 and 8/32¹ na Wikipedii](#)
- [Interdata² na Wikipedii](#)
-
-

96.1.1. Interdata 7/32

Vlastnosti:

-
-

96.1.2. Interdata 8/32

Vlastnosti:

-
-

Poznámky

1. http://en.wikipedia.org/wiki/Interdata_7/32_and_8/32
2. <http://en.wikipedia.org/wiki/Interdata>

Kapitola 97. Blíže neidentifikované počítače

V této části jsou částečné popisy počítačů jenž se mi zatím nepodařilo blože identifikovat a zařadit.

97.1. RCA 110

Odkazy:

- <http://www.quadibloc.com/comp/cp0303.htm>

Fakta:

- 12-ti bitová adresa, počítač měl jen 4K slov operační paměti
- 32K slov dat na bubnové paměti

97.2. Central Air Data Computer

Odkazy:

- Central Air Data Computer¹

První mikroprocesor. Protože tento procesor byl použit v stíhacích letounech F14A, byla celá práce a její výsledky utajeny. Ray Holt tedy nedošel uznání jako tvůrce prvního mikroprocesoru.

Poznámky

1. http://www.thocp.net/biographies/papers/holt_ray_f14a.htm

Kapitola 98. Všehochuť, náměty a co se jinem nevešlo

Různé věci vzdáleně související či vůbec nesouvisející s obsahem knihy.

Odkazy:

- CoreLife: The Linear Thinkers Nightmare¹

98.1. CoreWars

Odkazy:

- Redcode²
- CORE WAR GUIDELINES³
- <http://www.ecst.csuchico.edu/~pizza/koth/tutorial1.html>
- Introduction to Redcode⁴ — část 1
- Introduction to Redcode⁵ — část 2
- Annotated Draft of the Proposed 1994 Core War Standard⁶
- The ICWS'94 draft (extended)⁷
- KOTH.org⁸
-
-

*

98.2. Zajímavé a velmi podivné projekty

Projekty jenž s elektronikou nemusejí souviset ani okrajově.

98.2.1. Fusion reactor

Odkazy:

- Fusion Reactor's First Light!⁹
- Fusor¹⁰
- Fusion Achieved¹¹
-
-

98.2.2. Elektronový mikroskop

*

Odkazy:

- DIY scanning electron microscope - Overview video¹² [2011-03-21]
-

98.3. Různé nezařazené informace

Odkazy:

- Elektronické systémy¹³ — řešení a taháky
- Dieter's Hobby Projects¹⁴ — tranzistorový počítač MT15
- Coincident Current Ferrite Core Memories¹⁵ —
- Mark's free (legal) technical e-books links¹⁶
- —

98.4. Ham Radio

Odkazy:

- OK1IKE¹⁷
-

98.5. Nápady na experimenty

98.5.1. Laser

*

Odkazy:

- YouTube video Powerful Homemade Burning Laser Built From Computer Parts¹⁸

98.5.2. OLED

Odkazy:

- Making an OLED - Light from Carbon Compounds¹⁹
- Flexible Aluminum Electroluminescent Display - No Transparent²⁰
- Make Electroluminescent (EL) Ink at Home - Re-doping Glow Powder²¹
-

98.6. Výrobci součástek

*

98.6.1. National Semiconductor

*

Odkazy:

- National Semiconductor²²

•
Výrobce integrovaných obvodů, operačních zesilovačů, teplotních čidel, . . .

Poznámky

1. <http://corewar.co.uk/corelife/index.htm>
2. <http://en.wikipedia.org/wiki/Redcode>
3. <http://users.obs.carnegiescience.edu/birk/COREWAR/DOCS/guide2red.txt>
4. <http://www.ecst.csuchico.edu/~pizza/koth/tutorial1.html>
5. <http://www.ecst.csuchico.edu/~pizza/koth/tutorial2.html>
6. <http://www.ecst.csuchico.edu/~pizza/koth/icws94.html>
7. <http://www.corewar.info/lexicon/94draft.htm>
8. <http://www.koth.org/index.html>
9. <http://tidbit77.blogspot.com/2010/02/fusion-reactors-first-light.html>
10. <http://en.wikipedia.org/wiki/Fusor>
11. <http://tidbit77.blogspot.com/2011/03/fusion-achieved.html>
12. <http://benkrasnow.blogspot.com/2011/03/diy-scanning-electron-microscope.html>
13. <http://bruxy.regnet.cz/fel/35ES/>
14. <http://www.6502.org/users/dieter/index.htm>
15. <http://ed-thelen.org/comp-hist/Byte/76jul.html>
16. <http://www.freelabs.com/~whitis/ebooks/index.xhtml>
17. <http://oklike.c-a-v.com/>
18. <http://www.youtube.com/watch?v=3Kc4RyOMDjo>
19. <http://www.youtube.com/user/jeriellsworth#p/u/26/CAgRF8TibJ0>
20. <http://www.youtube.com/watch?v=ZmkzrX4T4Ec>
21. <http://www.youtube.com/watch?v=pmQdYrn9g8>
22. <http://www.national.com>

Příloha A. Vybavení laboratoře/dílny

*

Různé informace o tom co je třeba, nebo se alespoň hodí.

A.1. Nepájivá kontaktní pole (*Breadboard*)

*

Odkazy:

- Five Cheap Breadboard Prototyping Tips And Tricks¹
-
- cixi wanjie electronic co.,ltd²
- cxzhongyi³
- KEIFO Machinery & Electronics Co., LTD⁴
-
- breadboard(270 tie-points)⁵ na Alibaba.com
-

Bez tohoto si žádné experimentování nedovedu představit.

SYB-46

Kupuji v místním obchodě jednou za čas. Mají jen v kusovém množství. Zjistil jsem že tuhle velikost prodává SOS⁶ pod názvem BB-005. Tento druh a velikost má na všech 4 stranách "zámky" a dá se pěkně skládat do sebe ze všech čtyř stran. Dá se tak vytvořit pevně spojená poloha libovolného tvaru a velikosti. Samozřejmě že v případě většího množství spojených modulů je třeba je přichytit pevně k nějakému podkladu. Ať již šroubkem nebo oboustranou lepicí pěnou.

SYB-94

Tento model je více než dvakrát větší než SYB-46 a má taky zámkové. Vyšší modely již zámkové nemají.

SYB-120

Dá se koupit na Ebay.

SYB-130

Dá se sehnat na Ebay.

SYB-118T

SYB-500

SYB-800

ZY-46

ZY-446

ZY-100

ZY-101

ZY-102

ZY-128B

ZY-200

ZY-201

ZY-203

ZY-204

ZY-W202

A.2. Moduly pro nepájivá kontaktní pole a experimenty

*

Odkazy:

- Moduly do nepájivého kontaktního pole pro bastlení⁷
- Pomocníci při bastlení⁸ — článek na MCU.cz [2010-07-08]
-
-

A.3. Meřicí přístroje

*

Odkazy:

- Saleae logic⁹ — logický analyzátor k počítači
- OpenBench Logic Sniffer¹⁰ na Hack a Day. — 32 kanálový logický analyzátor
-

A.4. Napájecí zdroje

*

A.4.1. Využití standardních počítačových zdrojů

*

Odkazy:

- How to Convert a Computer ATX Power Supply to a Lab Power Supply¹¹
- YouTube video Lab power supply from computer power supply¹²
-
-
-
-

Poznámky

1. <http://www.43oh.com/2010/11/five-cheap-breadboard-prototyping-tips-and-tricks/>
2. http://wanjie.en.madeinchina.com/pg1441042_1/SOLDERLESS-BREADBOARD.shtml
3. http://www.cxzhongyi.com/en_product2.htm
4. http://www.keifo.com/product_view.asp?id=699
5. http://www.alibaba.com/product-gs/216567425/breadboard_270_tie_points_.html
6. <http://www.soselectronic.cz/?str=371&artnum=53508>
7. <http://mojekonstrukce.webpark.cz/moduly.htm>
8. <http://mcu.cz/comment.php?comment.news.1967>
9. <http://www.saleae.com/logic/>
10. <http://hackaday.com/2010/02/28/open-source-logic-analyzer-2/>
11. <http://www.wikihow.com/Convert-a-Computer-ATX-Power-Supply-to-a-Lab-Power-Supply>
12. <http://www.youtube.com/watch?v=1BDDk5K751E>

Příloha B. Technologie a postupy

*

B.1. SMT / SMD

*

Odkazy:

- Circuit Skills: Surface Mount Devices¹ z Collin's Lab na YouTube
-

Součástky (odpory, kondenzátory, ...) se vyrábějí v několika velikostech: 1206, 0805, 0603, 0402

Tento číselný kód specifikuje rozměr součástek v setinách palce. Tedy:

- 1206 má rozměr 0.12" × 0.06" (3.01 × 1.52 mm)
- 0805 má rozměr 0.08" × 0.05" (2.03 × 1.27 mm)
- 0603 má rozměr 0.06" × 0.04" (1.52 × 0.76 mm)
- 0402 má rozměr 0.04" × 0.02" (1.02 × 0.51 mm)

* SMD Test Kit na Jameco.com, part# 2124197.

B.2. DPS (PCB)

*

Odkazy:

- Plošný spoj² na Wikipedii
- Praktická elektronika/Výroba plošných spojů³ na Wikiknihách
- Konstrukce elektronických zařízení — návrh plošných spojů⁴
-

Odkazy:

- PCB Fabrication Sources⁵ [2011-02-10]
-

Jak vytvořit předlohu pro výrobu DPS:

- PCB Design Layout Rules Recommendation⁶
- THE BASICS OF PCB LAYOUT⁷
-

Desky s plošnými spoji. Česká zkratka je DPS a anglická PCB (*Printed Circuit Board*).

Tabulka B-1. Třídy PCB

třída	vzdálenost středů	min. šířka vodiče	Isol	počet vodičů mezi body
1	5mm / 5.08mm			0
2	3.54mm / 3.59mm	0.40mm		

třída	vzdálenost středů	min. šířka vodiče	Isol	počet vodičů mezi body
3	2.5mm / 2.54mm			0
4	2.5mm / 2.54mm	12mil	12mil	1
5	2.5mm / 2.54mm	8mil	8mil	2
6	2.5mm / 2.54mm	6mil	6mil	3

Isol

Isolační vzdálenost

W

Šířka spoje

Tabulka B-2. Označení materiálů pro PCB

oz- načení	popis
FR1	papír nasycený fenolovou pryskyřicí — laciný druh
FR2	papír nasycený fenolovou pryskyřicí – standardní provedení
FR3	papír nasycený epoxidovou pryskyřicí
FR4	tkanina ze skelných vláken sycená epoxidovou pryskyřicí – nejběžnější druh
FR5	tkanina ze skelných vláken sycená epoxidovou pryskyřicí – zvláště tepelně odolný druh

Odkazy:

-
-

Rozměry tzv eurokaret jsou určeny normou DIN 41494-2.

Tabulka B-3. Rozměry některých eurokaret

délka	šířka
100mm	55.5mm
100mm	80mm
100mm	100mm
160mm	100mm
220mm	100mm

Poznámky

1. <http://www.youtube.com/watch?v=ihoX7x0RBz8>
2. http://cs.wikipedia.org/wiki/Plo%C5%A1n%C3%BD_spoj
3. http://cs.wikibooks.org/wiki/Praktick%C3%A1_elektronika/V%C3%BDroba_plo%C5%A1n%C3%BDch_spoj%C5%AF
4. http://files.gamepub.sk/ET1/Konstrukce_elektronickych_zarizeni_navrh_plosnych_spoju.pdf
5. <http://digital-diy.com/General-Electronics/pcb-fabrication-sources.html>
6. <http://www.electronics-project-design.com/PCB-Design.html>
7. <http://www.smeps.us/layout.html>

Příloha C. Odkazy na weby a blogy

*

C.1. České stránky

*

Odkazy:

- fprik¹ — osobní WWW stránky
- Radioamatérská dílna²
- obvody.unas.cz³ [2008]
- DANYK⁴
-

C.2. Ostatní stránky

*

Odkazy:

- Electronic Lives Manufacturing⁵ by ChaN [JP]
- Dick Cappels' Project Pag⁶
- MadLab.ORG⁷ — MadLab – inventive electronic kits and exciting workshops
- CHD⁸ — informace o PDP-8, PDP-11, PDP-14, VAX a IBM S/34, a pár dalších zajímavostí
-

Nezpracované odkazy:

-
-

Velmi hezkým videoblogem je EEVblog⁹ Davida L. Jonese na YouTube. Blog je dosažitelný taktéž přes adresu EEVblog – Electronics Engineering Video Blog¹⁰.

Seznam dalších blogů:

- An off-the-cuff radio show for electronics enthusiasts and professionals¹¹
-
-

Knihy:

- Art of Electronics¹²
-
-
-

C.3. Obchody

*

Ačkoliv se snažím uvádět možné obchody u jednotlivých kapitol či témat, co zbylo píše sem.

Odkazy:

- ArcadeComponents.com¹³ — zde se dají koupit osmibitové procesory jako 6502, 6800, 6809, Z80, 8080, 8085 a spousta dalších čipů
-

Poznámky

1. <http://fprik.net/index.html>
2. <http://www.volny.cz/pjenicek/radio/index.htm>
3. <http://www.obvody.unas.cz/index.html>
4. <http://danyk.wz.cz>
5. <http://elm-chan.org/>
6. <http://cappels.org/dproj/Home.htm>
7. <http://www.madlab.org>
8. <http://www.chd.dyndns.org/>
9. <http://www.youtube.com/user/EEVblog>
10. <http://www.eevblog.com/>
11. <http://www.theamphour.com/>
- 12.
13. <http://www.arcadecomponents.com/index.html>

Příloha D. Různá doporučení

*

D.1. Jeri Ellsworth

*

Doporučené knihy ke studiu FPGA dizajnu

Odkazy:

- Rapid Prototyping of Digital Systems (VHDL, mouse controller, video controller, ...)
- Digital Design with CPLD Applications¹
- The Blue Book: HDL Chip Design²
-
-
-
-

D.2. Dave Jones

*

Základní vybavení laboratoře pro elektroniku.

Odkazy:

- Dva multimetry, ne příliš drahé, asi tak \$50 každý. Dva proto že občas potřebujeme měřit dvě hodnoty současně, například proud a napětí. Hodí se také si pořídit za cca \$20 multimetr v kapesním provedení.
- Jednoduchý doutnavkový indikátor síťového napětí. Prodává se také v provedení šroubovák. Cena asi \$20
- Digitální osciloskop, například RIGOL DS1052E³ za cca \$400
- starý dobrý použitý analogový osciloskop, pokud jej někde levně seženeme
- generátor funkcí, například instek GFG-8219A⁴ neb Instek SFG-1003
- laboratorní zdroj, nebo raději několik. Tento si můžete velmi jednoduše postavit sami.
- mikropájku
- odsávačku
- štipáčky, kombinačky, šroubováky, klíče
- ...

D.2.1. Rigol DS1052E a další

*

Odkazy:

- Hacking the Rigol DS1052E Oscilloscope with Linux⁵
- EEVblog #70 - Turn your Rigol DS1052E Oscilloscope into a 100MHz DS1102E (Hack)⁶
- EEVblog #77 - Rigol DS1052E DS1102E Oscilloscope Hack Update⁷
- Dvoukanálový digitální paměťový osciloskop 50MHz, 1GSa/s Rigol DS1052E⁸
-
-

Poznámky

- 1.
- 2.
- 3.
- 4.
5. <http://www.instructables.com/id/Hacking-the-Rigol-DS1052E-Oscilloscope-with-Linux/>
6. <http://www.youtube.com/watch?v=LnhXfVYWYXE>
7. <http://www.youtube.com/watch?v=R2dGKcMtAvg>
8. <http://drieg.sweb.cz/rigol/ds1052e.html>