

PB151 Výpočetní systémy

Michal Brandejs brandejs@fi.muni.cz

Katedra počítačových systémů a komunikací FI MU Centrum výpočetní techniky FI MU

17. dubna 2022

PB151 Výpočetní systémy / PB150 Architektury výpočetních systémů

Michal Brandejs
brandejs@fi.muni.cz

17. dubna 2022

Obsah

| | | |
|----------|--|-----------|
| 1 | Pojmy | 5 |
| 1.1 | Základní pojmy | 5 |
| 1.2 | Koncepce Johna von Neumanna | 9 |
| 2 | Číselné soustavy | 10 |
| 2.1 | Číselné soustavy, Polyadické soustavy | 10 |
| 2.2 | Převody mezi polyadickými soustavami | 11 |
| 2.3 | Číselné soustavy v počítačové praxi | 12 |
| 2.4 | Aritmetika ve dvojkových kódech | 15 |
| 3 | Kódy | 15 |
| 3.1 | BCD kód | 15 |
| 3.2 | Vnější, detekční a opravné kódy | 17 |
| 3.3 | Kódová vzdálenost a její znázornění | 25 |
| 3.4 | Vztah kódové vzdálenosti a počtu chyb | 26 |
| 4 | Obvody | 26 |
| 4.1 | Booleova algebra | 26 |
| 4.2 | Obvodové znázornění Booleovy algebry | 29 |
| 4.3 | Shefferova algebra, Peirceova algebra | 30 |
| 4.4 | Fyzikální postata signálů, zakázané pásmo | 31 |
| 4.5 | TTL, invertor v TTL, NAND a NOR v TTL | 33 |
| 4.6 | Kombinační logické obvody - základní logické členy | 34 |
| 4.7 | Kombinační logické obvody - ostatní logické členy | 35 |
| 4.8 | Logické obvody | 36 |
| 4.9 | Dekodér, sčítačky | 37 |
| 4.10 | Úplná sčítačka, vícemístná sčítačka | 38 |
| 4.11 | Sekvenční logické obvody, klopný obvod RS | 39 |
| 4.12 | Klopný obvod D | 41 |
| 4.13 | Typické sekvenční obvody | 42 |
| 4.14 | Přenos informací v systému | 42 |
| 4.15 | Sčítačka v BCD kódu | 45 |
| 4.16 | Násobičky | 45 |
| 4.17 | Rotace bitů, logický a aritmetický posun | 46 |
| 4.18 | Obvod pro rotaci bitů vpravo, vlevo a beze změny, komparátor | 47 |

| | | |
|-----------|---|-----------|
| 5 | Paměti | 48 |
| 5.1 | Parametry | 48 |
| 5.2 | Vnitřní paměti | 49 |
| 5.3 | Permanentní paměti | 50 |
| 5.4 | Asociativní paměť | 51 |
| 6 | Procesor | 52 |
| 6.1 | Struktura, fáze, registry, typy instrukcí, instrukce, Little/Big-Endian | 52 |
| 6.2 | Instrukce a mikroinstrukce, příznaky | 54 |
| 6.3 | Větvení | 56 |
| 6.4 | Zásobník, volání podprogramu, V/V operace | 59 |
| 6.5 | Přerušení | 62 |
| 7 | Virtuální paměť | 63 |
| 7.1 | Algoritmus LRU | 64 |
| 8 | Vyrovnávací (cache) paměť | 66 |
| 8.1 | Použití cache paměti | 66 |
| 9 | Architektura procesorů Intel – Procesor 8086 | 67 |
| 9.1 | Zapojení procesorů 8086/88 | 67 |
| 9.2 | Typy dat zpracovávané procesory Intel | 68 |
| 9.3 | Adresace paměti procesoru 8086 | 68 |
| 9.4 | Registry procesoru | 69 |
| 9.5 | Příznakový registr | 70 |
| 9.6 | Zásobník | 70 |
| 9.7 | Přerušení | 71 |
| 9.8 | Rezervovaná přerušení 8086, počáteční nastavení procesoru | 72 |
| 9.9 | Adresovací techniky | 73 |
| 9.10 | Instrukce MOV | 74 |
| 9.11 | Aritmetické instrukce | 74 |
| 9.12 | Logické instrukce | 76 |
| 9.13 | Rotace, posuvy | 77 |
| 9.14 | Větvení programu | 78 |
| 9.15 | Podmíněné skoky | 79 |
| 9.16 | Zásobník | 81 |
| 9.17 | Volání a návrat z podprogramu | 81 |
| 9.18 | Příznakový registr | 82 |
| 9.19 | Přerušení | 82 |
| 9.20 | Cykly | 83 |
| 9.21 | Ovládání V/V | 83 |
| 9.22 | Další instrukce přesunů dat | 84 |
| 9.23 | Řetězcové instrukce | 85 |
| 9.24 | Instrukce pro podporu BCD aritmetiky | 85 |
| 10 | Architektura procesorů Intel – Procesor 80186 | 86 |
| 11 | Architektura procesorů Intel – Procesor 80286 | 86 |
| 11.1 | Zapojení | 86 |
| 11.2 | Režimy | 87 |
| 11.3 | Příznakový registr | 87 |
| 11.4 | Registr MSW | 87 |
| 11.5 | Adresace paměti v chráněném režimu | 88 |
| 11.6 | Tabulky popisovačů segmentů | 88 |
| 11.7 | Popisovač datového segmentu | 89 |
| 11.8 | Popisovač instrukčního segmentu | 90 |
| 11.9 | Popisovač systémového segmentu | 90 |

| | | |
|-----------|--|------------|
| 11.10 | Segmentové registry | 90 |
| 11.11 | Registry GDTR a LDTR | 91 |
| 11.12 | Použití GDTR, LDTR a segmentových registrů | 91 |
| 11.13 | Úrovně oprávnění | 91 |
| 11.14 | Zpřístupnění datového segmentu | 92 |
| 11.15 | Předání řízení do instrukčního segmentu | 92 |
| 11.16 | Brána pro předání řízení | 92 |
| 11.17 | Použití brány pro předání řízení | 93 |
| 11.18 | Předávání parametrů pomocí brány | 93 |
| 11.19 | Privilegované instrukce | 94 |
| 11.20 | Segment stavu procesu | 94 |
| 11.21 | Segment stavu procesu - obsah | 95 |
| 11.22 | Brána zpřístupňující TSS | 95 |
| 11.23 | Přerušení | 95 |
| 11.24 | Brány pro přerušení | 96 |
| 11.25 | Informace ukládaná do zásobníku | 96 |
| 11.26 | Formát chybového slova | 96 |
| 11.27 | Rezervovaná přerušení | 97 |
| 11.28 | Počáteční nastavení procesoru | 97 |
| 11.29 | Zapnutí chráněného režimu | 98 |
| 12 | Architektura procesorů Intel – Procesor 80386 | 98 |
| 12.1 | Popis signálů a registry | 98 |
| 12.2 | Adresace v chráněném režimu a řídicí registry | 99 |
| 12.3 | Popisovače segmentů | 100 |
| 12.4 | Stránkování, Translation Look-aside Buffer (1) | 102 |
| 12.5 | Mapa přístupných V/V bran, přerušení | 104 |
| 13 | Architektura procesorů Intel – Procesor 80486 | 104 |
| 13.1 | Zapojení | 105 |
| 13.2 | Rysy procesoru, příznakový registr | 105 |
| 13.3 | Schéma činnosti vyrovnávací paměti | 106 |
| 13.4 | Řídicí registry CR0, CR3 | 106 |
| 13.5 | Stránkování | 107 |
| 13.6 | Interní vyrovnávací paměť | 107 |
| 13.7 | Pseudo-LRU | 108 |
| 14 | Architektura procesorů Intel – Procesor Pentium | 109 |
| 14.1 | Rysy procesoru | 109 |
| 14.2 | Blokový diagram | 109 |
| 14.3 | Zřetěžené provádění instrukcí | 110 |
| 14.4 | Předvídaní podmíněných skoků | 110 |
| 14.5 | Párování instrukcí | 111 |
| 14.6 | Režim správy systému | 111 |
| 15 | Architektura x86-64 | 112 |
| 15.1 | Principy architektury x86-64 | 112 |
| 16 | RISC | 114 |
| 16.1 | Principy architektury i860 | 114 |
| 16.2 | Jednotky a principy i860 | 115 |

| | |
|---|------------|
| 17 IEEE 754 | 116 |
| 17.1 Matematický koprocessor 80x87 | 116 |
| 17.2 Typy dat | 116 |
| 17.3 Zobrazení reálného čísla (1) | 117 |
| 17.4 Zobrazení reálného čísla (2), zvláštní čísla | 117 |
| 17.5 Převod čísel do a z IEEE 754 | 118 |
| 17.6 Zvláštní čísla (2) | 118 |
| 17.7 Určení rozsahu, přesnosti a rozlišitelnosti | 119 |
| 17.8 Typy operací koprocessoru | 119 |
| 18 Připojování V/V zařízení | 120 |
| 18.1 Centronics | 120 |
| 18.2 RS-232-C (V.24), zapojení, signály | 121 |
| 18.3 RS-232-C (V.24), průběhy, formáty | 122 |
| 18.4 Proudová smyčka | 122 |
| 18.5 Nullmodem | 123 |
| 18.6 USB Universal Serial Bus | 123 |

1 Pojmy

1.1 Základní pojmy

Pojmy

- **Technické vybavení počítače – Hardware – HW** – souhrnný název pro veškerá fyzická zařízení, kterými je počítač vybaven
- **Programové vybavení počítače – Software – SW**
- **Firmware** – programy tvořící součást technického vybavení počítače

Pojmy

- **1 bit** (z anglického BInary digiT) 1 b – základní jednotka informace.
- **1 bajt** (byte, oktet, slabika) 1 bajt = 1 B – skupina 8 bitů.

Proč právě 8 bitů?

Proč je bajt 8bitový?

- Bajt je jednotka ukládající typicky 1 znak, se kterým pracuje člověk.
- Písmen naší abecedy A-Z je 26 bez diakritiky, plus malá písmena, číslice, další znaky ... potřeba celkem cca 100 různých znaků.
- Na zobrazení 100 různých znaků je potřeba 7 bitů, protože $2^7 = 128$.

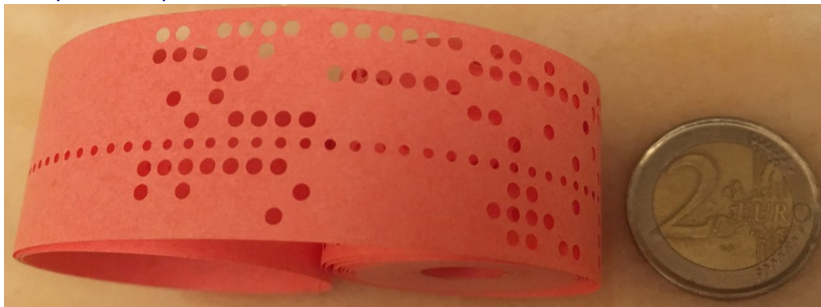
| | | | | | |
|--------|-----|--------------------|--------------------|--------------------|---------------------|
| 1 bit | 0/1 | 2 možnosti = 2^1 | | | |
| 2 bity | 0/1 | 0/1 | 4 možnosti = 2^2 | | |
| 3 bity | 0/1 | 0/1 | 0/1 | 8 možností = 2^3 | |
| 4 bity | 0/1 | 0/1 | 0/1 | 0/1 | 16 možností = 2^4 |
| ... | | | | | |

Znaky (písmena, číslice a jiné) se kódují (transformují) do čísel. Každý znak má svoje číslo. Příklad **ASCII**

ASCII kódování/transformace znaků na čísla

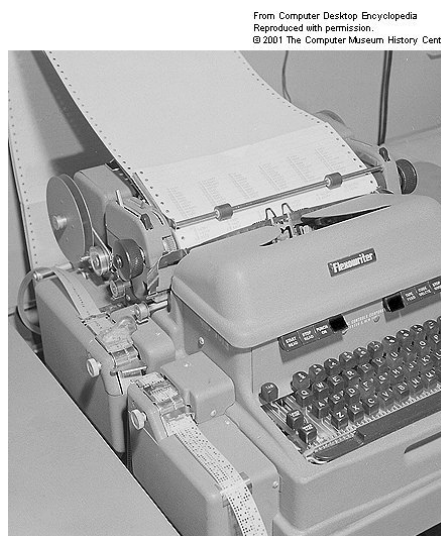
| Decimal | Binary | Octal | Hex | ASCII | Decimal | Binary | Octal | Hex | ASCII | Decimal | Binary | Octal | Hex | ASCII |
|---------|----------|-------|-----|-------|---------|----------|-------|-----|-------|---------|----------|-------|-----|-------|
| 32 | 00100000 | 040 | 20 | SP | 64 | 01000000 | 100 | 40 | @ | 96 | 01100000 | 140 | 60 | ` |
| 33 | 00100001 | 041 | 21 | ! | 65 | 01000001 | 101 | 41 | A | 97 | 01100001 | 141 | 61 | a |
| 34 | 00100010 | 042 | 22 | " | 66 | 01000010 | 102 | 42 | B | 98 | 01100010 | 142 | 62 | b |
| 35 | 00100011 | 043 | 23 | # | 67 | 01000011 | 103 | 43 | C | 99 | 01100011 | 143 | 63 | c |
| 36 | 00100100 | 044 | 24 | \$ | 68 | 01000100 | 104 | 44 | D | 100 | 01100100 | 144 | 64 | d |
| 37 | 00100101 | 045 | 25 | % | 69 | 01000101 | 105 | 45 | E | 101 | 01100101 | 145 | 65 | e |
| 38 | 00100110 | 046 | 26 | & | 70 | 01000110 | 106 | 46 | F | 102 | 01100110 | 146 | 66 | f |
| 39 | 00100111 | 047 | 27 | ' | 71 | 01000111 | 107 | 47 | G | 103 | 01100111 | 147 | 67 | g |
| 40 | 00101000 | 050 | 28 | (| 72 | 01001000 | 110 | 48 | H | 104 | 01101000 | 150 | 68 | h |
| 41 | 00101001 | 051 | 29 |) | 73 | 01001001 | 111 | 49 | I | 105 | 01101001 | 151 | 69 | i |
| 42 | 00101010 | 052 | 2A | * | 74 | 01001010 | 112 | 4A | J | 106 | 01101010 | 152 | 6A | j |
| 43 | 00101011 | 053 | 2B | + | 75 | 01001011 | 113 | 4B | K | 107 | 01101011 | 153 | 6B | k |
| 44 | 00101100 | 054 | 2C | , | 76 | 01001100 | 114 | 4C | L | 108 | 01101100 | 154 | 6C | l |
| 45 | 00101101 | 055 | 2D | - | 77 | 01001101 | 115 | 4D | M | 109 | 01101101 | 155 | 6D | m |
| 46 | 00101110 | 056 | 2E | . | 78 | 01001110 | 116 | 4E | N | 110 | 01101110 | 156 | 6E | n |
| 47 | 00101111 | 057 | 2F | / | 79 | 01001111 | 117 | 4F | O | 111 | 01101111 | 157 | 6F | o |
| 48 | 00110000 | 060 | 30 | 0 | 80 | 01010000 | 120 | 50 | P | 112 | 01110000 | 160 | 70 | p |
| 49 | 00110001 | 061 | 31 | 1 | 81 | 01010001 | 121 | 51 | Q | 113 | 01110001 | 161 | 71 | q |
| 50 | 00110010 | 062 | 32 | 2 | 82 | 01010010 | 122 | 52 | R | 114 | 01110010 | 162 | 72 | r |
| 51 | 00110011 | 063 | 33 | 3 | 83 | 01010011 | 123 | 53 | S | 115 | 01110011 | 163 | 73 | s |
| 52 | 00110100 | 064 | 34 | 4 | 84 | 01010100 | 124 | 54 | T | 116 | 01110100 | 164 | 74 | t |
| 53 | 00110101 | 065 | 35 | 5 | 85 | 01010101 | 125 | 55 | U | 117 | 01110101 | 165 | 75 | u |
| 54 | 00110110 | 066 | 36 | 6 | 86 | 01010110 | 126 | 56 | V | 118 | 01110110 | 166 | 76 | v |
| 55 | 00110111 | 067 | 37 | 7 | 87 | 01010111 | 127 | 57 | W | 119 | 01110111 | 167 | 77 | w |
| 56 | 00111000 | 070 | 38 | 8 | 88 | 01011000 | 130 | 58 | X | 120 | 01111000 | 170 | 78 | x |
| 57 | 00111001 | 071 | 39 | 9 | 89 | 01011001 | 131 | 59 | Y | 121 | 01111001 | 171 | 79 | y |
| 58 | 00111010 | 072 | 3A | : | 90 | 01011010 | 132 | 5A | Z | 122 | 01111010 | 172 | 7A | z |
| 59 | 00111011 | 073 | 3B | ; | 91 | 01011011 | 133 | 5B | [| 123 | 01111011 | 173 | 7B | { |
| 60 | 00111100 | 074 | 3C | < | 92 | 01011100 | 134 | 5C | \ | 124 | 01111100 | 174 | 7C | |
| 61 | 00111101 | 075 | 3D | = | 93 | 01011101 | 135 | 5D |] | 125 | 01111101 | 175 | 7D | } |
| 62 | 00111110 | 076 | 3E | > | 94 | 01011110 | 136 | 5E | ^ | 126 | 01111110 | 176 | 7E | ~ |
| 63 | 00111111 | 077 | 3F | ? | 95 | 01011111 | 137 | 5F | _ | 127 | 01111111 | 177 | 7F | DEL |

Osmistopá děrná páska

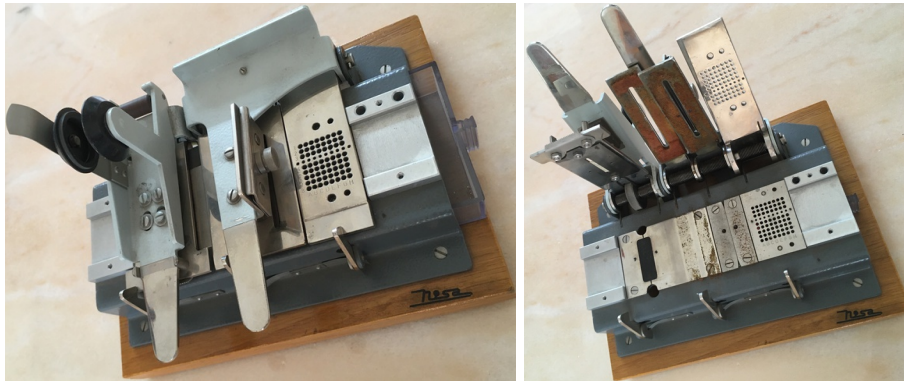


- ASCII pokrývá 7 bitů.
- 8. bit je tzv. **paritní bit** k jednoduché detekci chyby v bajtu.
- Na obrázku tzv. **sudá parita**, počet jedniček (tj. děr) je v bajtu vždy sudý.
- Jak je kontrola parity spolehlivá ochrana informace?

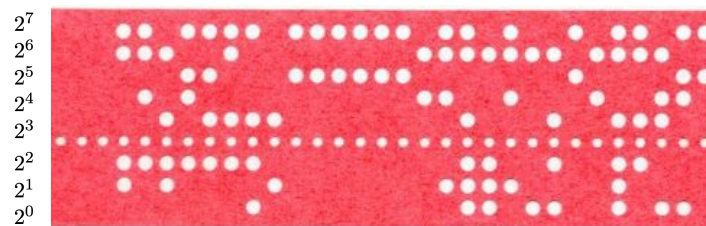
Požizovač dat na děrnou pásku - elektrický psací stroj



Ruční opravy děrné pásky



Jak číst děrnou pásku?



1. znak: $1\ 000\ 110_2 = 106_8 = \mathbf{F}$ 2. znak: $1\ 010\ 100_2 = 124_8 = \mathbf{T}$ 3. znak: $1\ 001\ 110_2 = 116_8 = \mathbf{N}$ 4. znak: $0\ 110\ 100_2 = 064_8 = \mathbf{4}$ 5. znak: $0\ 101\ 100_2 = 054_8 = \mathbf{.}$ 6. znak: $1\ 001\ 100_2 = 114_8 = \mathbf{L}$ 7. znak: $0\ 001\ 101_2$

| | | | | | | | | | |
|----|----------|-----|----|---|----|----------|-----|----|---|
| 38 | 00100110 | 046 | 26 | & | 70 | 01000110 | 106 | 46 | F |
| 39 | 00100111 | 047 | 27 | ' | 71 | 01000111 | 107 | 47 | G |
| 40 | 00101000 | 050 | 28 | (| 72 | 01001000 | 110 | 48 | H |
| 41 | 00101001 | 051 | 29 |) | 73 | 01001001 | 111 | 49 | I |
| 42 | 00101010 | 052 | 2A | * | 74 | 01001010 | 112 | 4A | J |
| 43 | 00101011 | 053 | 2B | + | 75 | 01001011 | 113 | 4B | K |
| 44 | 00101100 | 054 | 2C | . | 76 | 01001100 | 114 | 4C | L |
| 45 | 00101101 | 055 | 2D | - | 77 | 01001101 | 115 | 4D | M |
| 46 | 00101110 | 056 | 2E | . | 78 | 01001110 | 116 | 4E | N |
| 47 | 00101111 | 057 | 2F | / | 79 | 01001111 | 117 | 4F | O |
| 48 | 00110000 | 060 | 30 | 0 | 80 | 01010000 | 120 | 50 | P |
| 49 | 00110001 | 061 | 31 | 1 | 81 | 01010001 | 121 | 51 | Q |
| 50 | 00110010 | 062 | 32 | 2 | 82 | 01010010 | 122 | 52 | R |
| 51 | 00110011 | 063 | 33 | 3 | 83 | 01010011 | 123 | 53 | S |
| 52 | 00110100 | 064 | 34 | 4 | 84 | 01010100 | 124 | 54 | T |

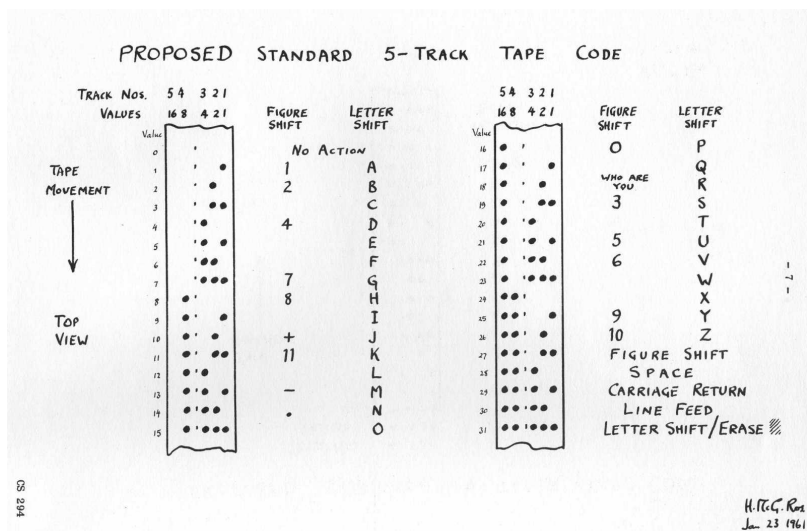
$= 015_8 = \mathbf{CR}$ 8. znak: $0\ 001\ 010_2 = 012_8 = \mathbf{LF}$

5stopá děrná páska

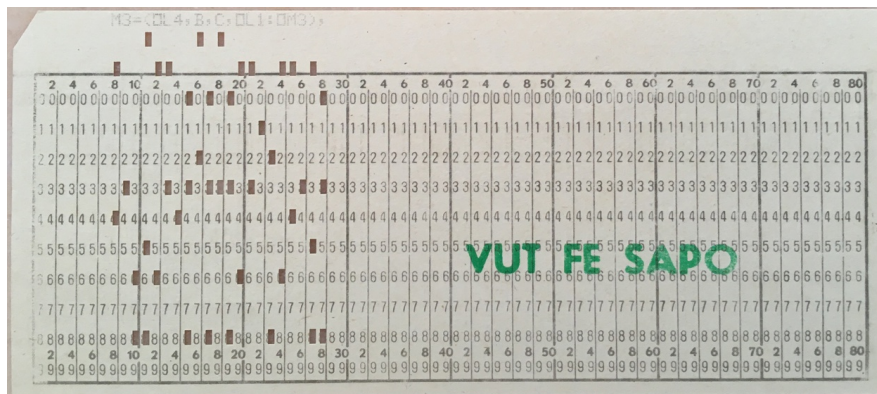
- Historicky starší 5stopá děrná páska.
- $2^5 = 32$ neumožňuje současně zobrazit ani velká písmena a číslice.
- Řešení pomocí písmenové a číslicové/znakové změny (letter and figure shift), vizte následující slajd.
- Použití v tzv. **dálnopisech**.



5stopá děrná páska



Děrný štítek – jiný způsob transformace/kódování znaků



Není ASCII, je jde o binárně kódované desítkové číslice EBCDIC.

Pojmy

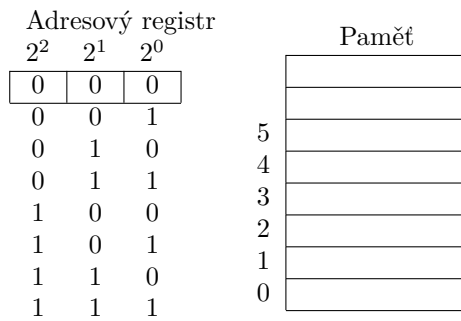
Protože např. pro uložení čísla je 1 bajt málo, používá se:

- **1 slovo** = 1 word – několik (2, 4, 6, 8, 10 apod.) bajtů.

Další pojmy:

- **Paměť** – zařízení pro uchovávání informace (konkrétně binárně kódovaných dat).
- **Operační paměť** – paměť uvnitř počítače/zařízení pro aktuálně procesorem/y zpracovávané instrukce a data.
- **Adresa v paměti** – číselné označení místa v paměti.
- **Nejmenší adresovatelná jednotka** – kapacita místa v paměti, které má vlastní adresu (bajt, slovo).

Kapacity pamětí v mocninách čísla 2



Kapacita paměti

- 1 KB = 1 KiB (kibi) = 2^{10} bajtů = 1.024 bajtů (kilobyte)
- 1 MB = 1 MiB (mebi) = 2^{20} bajtů = 1.048.576 bajtů
- 1 GB = 1 GiB (gibi) = 2^{30} bajtů = 1.073.741.824 bajtů
- 1 TB = 1 TiB (tebi) = 2^{40} bajtů (terabyte)
- 1 PB = 1 PiB (pebi) = 2^{50} bajtů (petabyte)
- 1 EB = 1 EiB (exbi) = 2^{60} bajtů (exabyte)
- 1 ZB = 1 ZiB (zebi) = 2^{70} bajtů (zettabyte)
- 1 YB = 1 YiB (yobi) = 2^{80} bajtů (yottabyte)

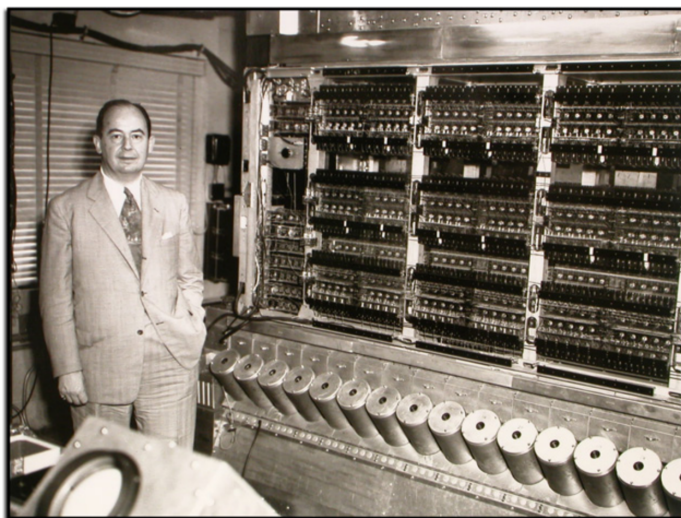
Způsob zápisu (mezera mezi číslem a zkratkou):

- Kapacita paměti je 1 GB.
- 1GB paměť.
- **RAM** – paměť pro čtení i zápis
- **ROM** – paměť pouze pro čtení
- **Paměť s přímým přístupem**
- **Paměť se sekvenčním přístupem**
- **Vnitřní (operační) paměť**
- **Vnější (periferní) paměť**
- **Registr**
- **V / V zařízení** (I / O Equipment)
- **Řadič** (Controller) – zařízení převádějící příkazy v symbolické formě (instrukce) na posloupnost signálů ovládajících připojené zařízení

1.2 Koncepte Johna von Neumanna

1945: Architektura „von Neumann“

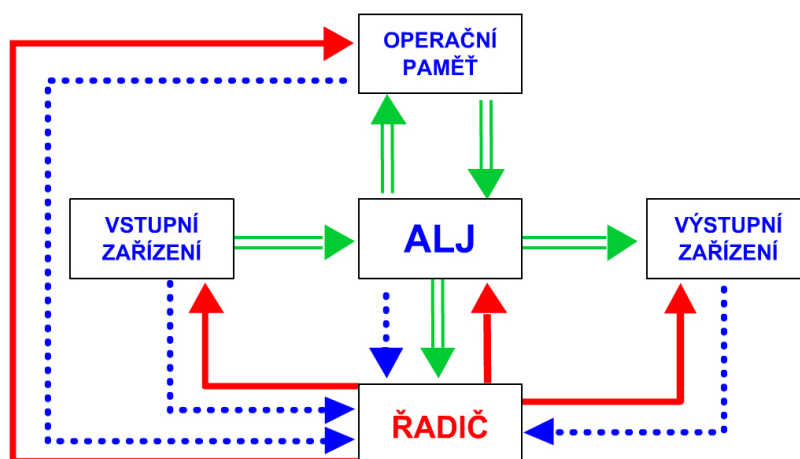
John von Neumann, Princeton Institute for Advanced Studies (IAS) Computer, cca 1952



1945: Architektura „von Neumann“

1. Počítač obsahuje operační paměť, ALJ, řadič, V/V zařízení.
2. Předpis pro řešení úlohy je převeden do posloupnosti instrukcí.
3. Údaje a instrukce jsou vyjádřeny binárně.
4. Údaje a instrukce se uchovávají v paměti na místech označených adresami.
5. Ke změně pořadí provádění instrukcí se používají instrukce podmíněného a nepodmíněného skoku.
6. Programem řízené zpracování dat probíhá v počítači samočinně.

1945: Architektura „von Neumann“



Pozn. "Konkurenční" Harvardská architektura má zvlášť paměť pro instrukce a zvlášť paměť pro data.

2 Číselné soustavy

2.1 Číselné soustavy, Polyadické soustavy

Polyadické soustavy

Polyadické, nebo též poziční soustavy. Hodnota/řád číslice je dán její pozicí v čísle.

Celá čísla kladná

zápis

číslo = součet mocnin základu vynásobených číslicemi

$$A = a_n \cdot z^n + a_{n-1} \cdot z^{n-1} + \dots + a_1 \cdot z^1 + a_0 \cdot z^0$$

$$A = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$$

zhuštěný zápis

běžná je forma zhuštěného zápisu:

$$A = a_n a_{n-1} \dots a_1 a_0$$

$$A = 123$$

$$A = 123_{10}$$

- **Reálná čísla kladná:** $A = a_n \cdot z^n + \dots + a_0 \cdot z^0 + a_{-1} \cdot z^{-1} + a_{-2} \cdot z^{-2} + \dots + a_{-m} \cdot z^{-m}$
- **Zobecnění pro záporná čísla** – přidáním znaménka (pro počítače nevhodné)

Soustavy užívané v počítačové praxi

$$z = 2$$

Dvojková

Binární

$$0, 1 \quad z = 8$$

Osmičková

Oktalová

0, 1, 2, 3, 4, 5, 6, 7 $z = 16$

Šestnáctková

Hexadecimální

0, 1, ..., 9, A, ..., F

2.2 Převody mezi polyadickými soustavami

| $z = 10$ | $z = 2$ | $z = 8$ | $z = 16$ |
|----------|---------|---------|----------|
| 0 | 000000 | 0 | 0 |
| 1 | 001 | 1 | 1 |
| 2 | 010 | 2 | 2 |
| 3 | 011 | 3 | 3 |
| 4 | 100 | 4 | 4 |
| 5 | 101 | 5 | 5 |
| 6 | 110 | 6 | 6 |
| 7 | 111 | 7 | 7 |
| 8 | 1000 | 10 | 8 |
| 9 | 1001 | 11 | 9 |
| 10 | 1010 | 12 | A |
| 11 | 1011 | 13 | B |
| 12 | 1100 | 14 | C |
| 13 | 1101 | 15 | D |
| 14 | 1110 | 16 | E |
| 15 | 1111 | 17 | F |
| 16 | 10000 | 20 | 10 |
| 17 | 10001 | 21 | 11 |
| ... | | | |
| 32 | 100000 | 40 | 20 |

Převody mezi soustavami

Číslo v soustavě o základu z^k (kde z a k jsou přirozená čísla) lze převést do soustavy o základu z jednoduše.

Převody

| | |
|------------------------|-------------------------|
| $2 \leftrightarrow 8$ | $8 \nleftrightarrow 16$ |
| $2 \leftrightarrow 16$ | $2 \nleftrightarrow 10$ |

Každou k -tici číslic nižší soustavy nahradíme číslicí soustavy vyšší.

Převod z 10 soustavy do 2

$$12, 2_{10} = ?_2$$

Rozdělit na celou a desetinnou část čísla:

| | | | |
|----|-------|-----|--------------|
| | | 0, | 2×2 |
| | | 0 | 4 |
| 12 | $: 2$ | 0 | 8 |
| 6 | 0 | 1 | 6 (0,6×2) |
| 3 | 0 | 1 | 2 (0,2×2) |
| 1 | 1 | 0 | 4 |
| 0 | 1 | 0 | 8 |
| | | 1 | 6 |
| | | ... | ... |

$$12, 2_{10} = 1100, 0011001100 \dots_2$$

Převod z 2 soustavy do 10

$$1100, 0011001100_2 = ?_{10}$$

Celá část:

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 = 12$$

Desetinná část:

$$0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} + 0 \times 2^{-5} + 0 \times 2^{-6} + 1 \times 2^{-7} + 1 \times 2^{-8} + \dots = 0 \times 0,5 + 0 \times 0,25 + 1 \times 0,125 + 1 \times 0,0625 + \dots = 0,19999\dots$$

Řešení: zaokrouhlení dle poslední číslice rozvoje.

Obecný algoritmus převodu

```

/* Algoritmus pro převod celé části desítkového čísla
   do soustavy z */
integer i := 0 ; /* Řád číslice */
integer Číslo := celé_číslo_bez_znaménka ; /* Převáděné číslo */
byte Základ := z ;
byte Číslice [] ; /* Vektor převedených číslic */
while Číslo > 0
begin
    Číslice [i] := Číslo MOD Základ ;
    Číslo := Číslo DIV Základ ;
    i := i + 1 ;
end;
/* V poli Číslice[0] až Číslice[n] jsou uloženy
   hodnoty číslic v obráceném pořadí */

/* Algoritmus pro převod necelé části desítkového čísla
   do soustavy z */
integer i ; /* Řád číslice */
real Číslo := necelá_část_čísla ; /* Převáděné číslo */
byte Základ := z ;
byte Číslice [] ; /* Vektor převedených číslic */
real Součin ; /* Pracovní proměnná */
for i := 1 to požadovaný_počet_číslíc
begin
    Součin := Číslo * Základ ;
    Číslice := TRUNC ( Součin ) ;
    Číslo := Součin - Číslice [i] ;
end;
/* V poli Číslice[1] až Číslice[požadovaný_počet_míst] jsou uloženy
   hodnoty číslic necelé části výsledného čísla v přímém pořadí */

```

2.3 Číselné soustavy v počítačové praxi

Zobrazení celého čísla v počítači v binárním tvaru

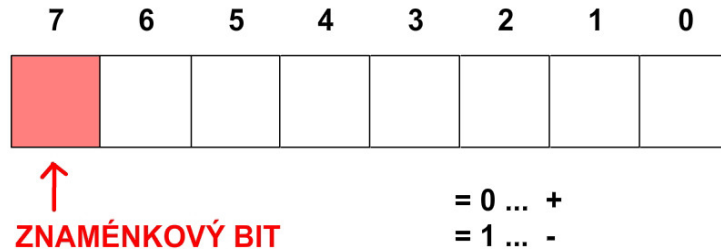
číslo bez znaménka Číslo pouze kladné v intervalu $\langle 0; 2^n - 1 \rangle$

Zobrazení např. na 4 bitech ($n = 4$):

| | | | | |
|---|---|---|---|----|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 9 |
| 1 | 1 | 1 | 1 | 15 |

číslo se znaménkem Kladné i záporné číslo.

Zobrazení celého čísla se znaménkem v binárním tvaru



Zobrazení kladných čísel

Rozsah zobrazení je $\langle 0; 2^{n-1} - 1 \rangle$

pro $n = 8$: $\langle 0; 127 \rangle$

Zobrazení záporných čísel: Přímý kód

Přímý kód

- v absolutní hodnotě
rozsah zobrazení je $\langle -2^{n-1} + 1; -0 \rangle$
pro $n = 4$: $\langle -7; -0 \rangle, \langle +0; 7 \rangle$
pro $n = 8$: $\langle -127; -0 \rangle, \langle +0; 127 \rangle$

| | | | | |
|---|---|---|---|----|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 7 |
| 0 | 0 | 1 | 1 | 3 |
| 1 | 0 | 1 | 1 | -3 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | -0 |

Zobrazení záporných čísel: Inverzní kód

Inverzní kód

- inverze bitů (jedničkový doplněk)
rozsah zobrazení je $\langle -2^{n-1} + 1; -0 \rangle$
pro $n = 4$: $\langle -7; -0 \rangle, \langle +0; 7 \rangle$
pro $n = 8$: $\langle -127; -0 \rangle, \langle +0; 127 \rangle$

| | | | | |
|---|---|---|---|----|
| 0 | 0 | 1 | 1 | 3 |
| 1 | 1 | 0 | 0 | -3 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | -0 |

Zobrazení záporných čísel: Doplnkový kód

Doplnkový kód

- dvojkový doplněk = inverze všech bitů a přičtení jedničky
rozsah zobrazení je $\langle -2^{n-1}; 2^{n-1} - 1 \rangle$
pro $n = 4$: $\langle -8; 7 \rangle$
pro $n = 8$: $\langle -128; 127 \rangle$, pro $n = 16$: $\langle -32768; 32767 \rangle$

| | | | | |
|---|---|---|----|----|
| 0 | 0 | 1 | 0 | 2 |
| 1 | 1 | 0 | 1 | |
| | | | +1 | |
| 1 | 1 | 1 | 0 | -2 |
| 0 | 0 | 0 | 1 | |
| | | | +1 | |
| 0 | 0 | 1 | 0 | 2 |

Doplnkový kód - dvě nuly?

| | | | | |
|-------|---|---|----|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | |
| | | | +1 | |
| [1] ← | 0 | 0 | 0 | 0 |

Přenos ze znaménkového bitu se ignoruje.

Okrajové hodnoty rozsahu zobrazení

| | | | | |
|---|---|---|----|----|
| 1 | 1 | 1 | 1 | -1 |
| 0 | 0 | 0 | 0 | |
| | | | +1 | |
| 0 | 0 | 0 | 1 | 1 |

Okrajové hodnoty rozsahu zobrazení

| | | | | |
|---|---|---|----|------------------------|
| 1 | 0 | 0 | 0 | co je to za číslo? |
| 0 | 1 | 1 | 1 | |
| | | | +1 | |
| 1 | 0 | 0 | 0 | nemá absolutní hodnotu |
| | | | +1 | |
| 1 | 0 | 0 | 1 | ? |
| 0 | 1 | 1 | 0 | |
| | | | +1 | |
| 0 | 1 | 1 | 1 | $7 \approx +MAX$ |
| 1 | 0 | 0 | 1 | $-7 \approx -MAX$ |
| 1 | 0 | 0 | 0 | $-8 \approx -MAX-1$ |

Vztahy mezi zobrazeními

| ± | Kódová kombinace | Význam v kódu | | |
|---|------------------|---------------|----------|-----------|
| | | Přímý | Inverzní | Doplnkový |
| 0 | 0 ... 00 | 0 | 0 | 0 |
| 0 | 0 ... 01 | 1 | 1 | 1 |
| 0 | 0 ... 10 | 2 | 2 | 2 |
| | ... | ... | ... | ... |
| 0 | 1 ... 11 | +MAX | +MAX | +MAX |
| 1 | 0 ... 00 | -0 | -MAX | -MAX - 1 |
| 1 | 0 ... 01 | -1 | -MAX + 1 | -MAX |
| | ... | ... | ... | ... |
| 1 | 1 ... 10 | -MAX + 1 | -1 | -2 |
| 1 | 1 ... 11 | -MAX | -0 | -1 |
| | šířka n-bitů | | | |

$$MAX \approx 2^{n-1} - 1$$

2.4 Aritmetika ve dvojkových kódech

Aritmetika ve dvojkových kódech

- Základní operace – **součet**
- **Přetečení (přeplnění)** = výsledek operace spadá mimo rozsah zobrazení

Součet v doplňkovém kódu

- všechny bity se sčítají stejně (včetně znaménkového)
- vznikne-li přenos ze znaménkového bitu, tak se ignoruje
- přetečení nastane, pokud se přenos do znaménkového bitu nerovná přenosu ze znaménkového bitu

$$\text{Př.: } \begin{array}{r|l} 0 & 110 \\ 0 & 101 \\ \hline {}^01 & {}^1011 \end{array} \quad \begin{array}{r|l} 1 & 010 \\ 1 & 011 \\ \hline {}^10 & {}^0101 \end{array}$$

Přetečení může nastat, jen pokud sčítáme čísla se stejným znaménkem. Po přetečení má výsledek znaménko opačné.

Čísla bez znaménka: přenos indikuje přetečení.

Součet v inverzním kódu

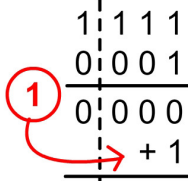
- problém dvou nul,
- nutnost provádět tzv. *kruhový přenos* = přičtení přenosu z nejvyššího řádu k výsledku.

Na číselné ose:

$$-3 \quad -2 \quad -1 \quad -0 \quad +0 \quad +1 \quad +2 \quad +3$$

se součtem např. $-1 + 2 = 1$ musíme posunout o jedno číslo navíc.

$$\begin{array}{r} -0 \\ +1 \\ \hline 1 \ 1 \ 1 \ 1 \\ 0 \ 1 \ 0 \ 0 \ 1 \\ \hline 0 \ 1 \ 0 \ 0 \ 0 \\ \hline 0 \ 1 \ 0 \ 0 \ 1 \\ \hline \end{array}$$

Př.: 

3 Kódy

3.1 BCD kód

Kód BCD (Binary Coded Decimal)

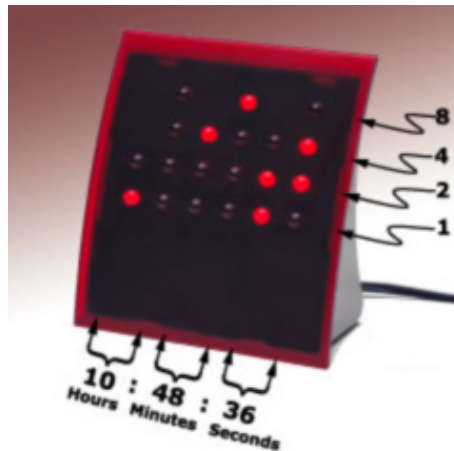
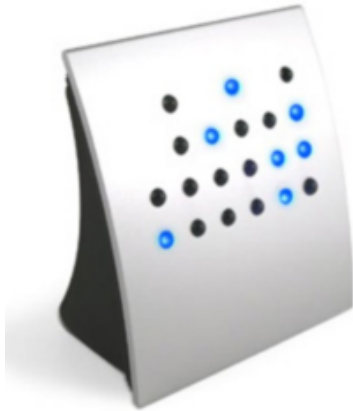
| | | | | |
|---|---|---|---|------------------|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 1 | 0 | <i>nepoužito</i> |
| 1 | 0 | 1 | 1 | <i>nepoužito</i> |
| 1 | 1 | 0 | 0 | <i>nepoužito</i> |
| 1 | 1 | 0 | 1 | <i>nepoužito</i> |
| 1 | 1 | 1 | 0 | <i>nepoužito</i> |
| 1 | 1 | 1 | 1 | <i>nepoužito</i> |

Jedna desítková číslice se zobrazuje vždy na 4 bitech:

Pojem: Nibble (půlbajt)

Nibble, půlbajt, buď spodní nebo horní 4 bity bajtu.

BCD hodiny



Historické dělení počítačů na binární vs. BCD

Počítače pro vědeckotechnické výpočty

protože v operacích převažuje aritmetika, používala se čistá binární soustava.

Počítače pro hromadné zpracování dat (např. pro sčítání lidu)

protože převažovaly vstupní a výstupní operace, používala se BCD soustava.

Nyní je toto dělení už bezpředmětné, protože zvýšením výkonu počítačů přestal být problém s převodem znaků na čísla.

Zobrazení čísel v BCD kódu

Rozvinutý tvar (unpacked decimal)

- mezitvar, nepoužívá se k výpočtům
- ekvivalentní název = zónový tvar desítkového čísla
- **zóna** = horní půlbajt
 - standardně F_{16}
 - $+ C_{16}$
 - $- D_{16}$

| | Desítkově | Rozvinutý tvar |
|------|---------------|-------------------|
| Př.: | 71346_{10} | $F7F1F3F4C6_{16}$ |
| | -71346_{10} | $F7F1F3F4D6_{16}$ |

Zhuštěný tvar (packed decimal)

- základní zobrazení pro výpočty
- vypouští se všechny zóny kromě nejpravější

| | Desítkově | Zhuštěný tvar |
|------|---------------|---------------|
| Př.: | 71346_{10} | $71346C_{16}$ |
| | -71346_{10} | $71346D_{16}$ |

3.2 Vnější, detekční a opravné kódy

Vnější kódy (tj. uložení písmen, číslic aj. znaků)

- Každý znak má svoji *ordinální (numerickou) hodnotu*.
- Jednobajtová kódování
- Vícebajtová kódování

Jednobajtová kódování

Vlastnosti zobrazení znak – ordinální hodnota:

- lexikální uspořádání
- snadný převod desítkových číslic na numerickou hodnotu

ASCII American Standard Code for Information Interchange
7bitové kódování, z roku 1963

ASCII – 7bitové kódování

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|-----|-------------|-------|---|---|---|---|-----|
| 0 | NUL | DLE | space | 0 | @ | P | ` | p |
| 1 | SOH | DC1 XON | ! | 1 | A | Q | a | q |
| 2 | STX | DC2 | " | 2 | B | R | b | r |
| 3 | ETX | DC3 XOFF | # | 3 | C | S | c | s |
| 4 | EOT | DC4 | \$ | 4 | D | T | d | t |
| 5 | ENQ | NAK | % | 5 | E | U | e | u |
| 6 | ACK | SYN | & | 6 | F | V | f | v |
| 7 | BEL | ETB | ' | 7 | G | W | g | w |
| 8 | BS | CAN | (| 8 | H | X | h | x |
| 9 | HT | EM |) | 9 | I | Y | i | y |
| A | LF | SUB | * | : | J | Z | j | z |
| B | VT | ESC | + | ; | K | [| k | { |
| C | FF | FS | , | < | L | \ | l | |
| D | CR | GS | - | = | M |] | m | } |
| E | SO | RS | . | > | N | ^ | n | ~ |
| F | SI | US | / | ? | O | _ | o | del |

ASCII: Řídící znaky I.

| Dec | Hex | Oct | Rotace | Znak | Kláv | Graficky | Název znaku |
|-----|-----|-----|--------|------|--------|----------|---------------------|
| 0 | 00 | 000 | 000000 | NUL | CTRL+@ | | null |
| 1 | 01 | 001 | 000400 | SOH | CTRL+A | ⊙ | start of heading |
| 2 | 02 | 002 | 001000 | STX | CTRL+B | ⦿ | start of text |
| 3 | 03 | 003 | 001400 | ETX | CTRL+C | ♡ | end of text |
| 4 | 04 | 004 | 002000 | EOT | CTRL+D | ◇ | end of transmission |
| 5 | 05 | 005 | 002400 | ENQ | CTRL+E | ♣ | enquiry |
| 6 | 06 | 006 | 003000 | ACK | CTRL+F | ♠ | acknowledge |
| 7 | 07 | 007 | 003400 | BEL | CTRL+G | • | bell |
| 8 | 08 | 010 | 004000 | BS | CTRL+H | ◻ | backspace |
| 9 | 09 | 011 | 004400 | HT | CTRL+I | ◊ | tab horizontally |
| 10 | 0A | 012 | 005000 | LF | CTRL+J | ◻ | line feed |
| 11 | 0B | 013 | 005400 | VT | CTRL+K | ♁ | tab vertically |
| 12 | 0C | 014 | 006000 | FF | CTRL+L | ♀ | form feed |
| 13 | 0D | 015 | 006400 | CR | CTRL+M | ♪ | carriage return |
| 14 | 0E | 016 | 007000 | SO | CTRL+N | ♯ | shift out |
| 15 | 0F | 017 | 007400 | SI | CTRL+O | * | shift in |

ASCII: Carriage Return, Line Feed

Carriage Return návrat vozíku, posun na začátek (též) řádku

Line Feed posun o řádek (beze změny sloupce)

Windows systémy ukončují řádek dvojicí CR LF.



UNIXové systémy ukončují řádek LF.

ASCII: Řídící znaky II.

| | | | | | | | |
|----|----|-----|--------|-----|--------|----|------------------------|
| 16 | 10 | 020 | 010000 | DLE | CTRL+P | ▷ | data link escape |
| 17 | 11 | 021 | 010400 | DC1 | CTRL+Q | ◁ | device control 1 X-ON |
| 18 | 12 | 022 | 011000 | DC2 | CTRL+R | ↑ | device control 2 TAPE |
| 19 | 13 | 023 | 011400 | DC3 | CTRL+S | !! | device control 3 X-OFF |
| 20 | 14 | 024 | 012000 | DC4 | CTRL+T | ¶ | device control 4 TAPE |
| 21 | 15 | 025 | 012400 | NAK | CTRL+U | § | negative acknowledge |
| 22 | 16 | 026 | 013000 | SYN | CTRL+V | - | synchronous idle |
| 23 | 17 | 027 | 013400 | ETB | CTRL+W | ↓ | end of transm. block |
| 24 | 18 | 030 | 014000 | CAN | CTRL+X | ↑ | cancel line |
| 25 | 19 | 031 | 014400 | EM | CTRL+Y | ↓ | end of medium |
| 26 | 1A | 032 | 015000 | SUB | CTRL+Z | → | substitute |
| 27 | 1B | 033 | 015400 | ESC | CTRL+[| ← | escape |
| 28 | 1C | 034 | 016000 | FS | CTRL+\ | └ | file separator |
| 29 | 1D | 035 | 016400 | GS | CTRL+] | ↔ | group separator |
| 30 | 1E | 036 | 017000 | RS | CTRL+^ | △ | record separator |
| 31 | 1F | 037 | 017400 | US | CTRL+_ | ▽ | unit separator |

8bitová kódování

- IBM PC
- Kameničtí
- PC-Latin2
- KOI-8čs
- Windows-1250
- ISO-8859-2 (ISO-Latin2)

| Dec | Hex | Oct | Rotace | PC | Kam | PCLat | Koi | Win | ISOLat2 |
|-----|-----|-----|--------|----|-----|-------|-----|-----|---------|
| 128 | 80 | 200 | 100000 | Ç | Č | Ç | | | |
| 129 | 81 | 201 | 100400 | ü | ü | ü | | | |
| 130 | 82 | 202 | 101000 | é | é | é | | , | |
| 131 | 83 | 203 | 101400 | â | d' | â | | | |
| 132 | 84 | 204 | 102000 | â | â | â | | » | |
| 133 | 85 | 205 | 102400 | à | Đ | ù | | ... | |
| 134 | 86 | 206 | 103000 | à | Ť | é | | † | |
| 135 | 87 | 207 | 103400 | ç | č | ç | | ‡ | |
| 136 | 88 | 210 | 104000 | ê | ë | î | | | |
| 137 | 89 | 211 | 104400 | ë | È | ë | | ‰ | |
| 138 | 8A | 212 | 105000 | è | Ĺ | Ó | | Š | |
| 139 | 8B | 213 | 105400 | ı | Í | σ | | < | |
| 140 | 8C | 214 | 106000 | ı | ı | ı | | Š | |
| 141 | 8D | 215 | 106400 | ı | Í | Ž | | Ť | |
| 142 | 8E | 216 | 107000 | Ă | Ă | Ă | | Ž | |
| 143 | 8F | 217 | 107400 | Ă | Ă | Č | | Ž | |

| | | | | | | | | | |
|-----|----|-----|--------|---|---|---|--|---|--|
| 144 | 90 | 220 | 110000 | É | É | É | | | |
| 145 | 91 | 221 | 110400 | æ | ž | Ĺ | | ‘ | |
| 146 | 92 | 222 | 111000 | Æ | Ž | Í | | ’ | |
| 147 | 93 | 223 | 111400 | ô | ô | ô | | “ | |
| 148 | 94 | 224 | 112000 | ö | ö | ö | | ” | |
| 149 | 95 | 225 | 112400 | ó | Ó | E | | • | |
| 150 | 96 | 226 | 113000 | ù | ù | ı | | — | |
| 151 | 97 | 227 | 113400 | ú | Ú | Š | | — | |
| 152 | 98 | 230 | 114000 | ÿ | ÿ | ś | | | |
| 153 | 99 | 231 | 114400 | Ö | Ö | Ö | | ™ | |
| 154 | 9A | 232 | 115000 | Ü | Ü | Ü | | § | |
| 155 | 9B | 233 | 115400 | đ | Š | Ť | | > | |
| 156 | 9C | 234 | 116000 | £ | E | ı | | ś | |
| 157 | 9D | 235 | 116400 | ¥ | Ý | L | | ı | |
| 158 | 9E | 236 | 117000 | ƒ | Ř | × | | ž | |
| 159 | 9F | 237 | 117400 | f | ı | č | | ž | |

| Dec | Hex | Oct | Rotace | PC | Kam | PCLat | Koi | Win | ISOLAT2 |
|-----|-----|-----|--------|----|-----|-------|-----|-----|---------|
| 160 | A0 | 240 | 120000 | á | á | á | | | |
| 161 | A1 | 241 | 120400 | ı | ı | ı | | ˘ | Ą |
| 162 | A2 | 242 | 121000 | ó | ó | ó | | ˘ | ˘ |
| 163 | A3 | 243 | 121400 | ú | ú | ú | | L | L |
| 164 | A4 | 244 | 122000 | ñ | ñ | Ą | | □ | □ |
| 165 | A5 | 245 | 122400 | Ñ | Ñ | ą | | Ą | E |
| 166 | A6 | 246 | 123000 | a | Ů | Ž | | | Ś |
| 167 | A7 | 247 | 123400 | o | Ô | ž | | § | § |
| 168 | A8 | 250 | 124000 | ı | s | Eı | | ” | ” |
| 169 | A9 | 251 | 124400 | ı | ı | ę | | © | Š |
| 170 | AA | 252 | 125000 | ı | ı | ę | | § | § |
| 171 | AB | 253 | 125400 | ½ | Ř | ž | | << | Ť |
| 172 | AC | 254 | 126000 | ¼ | ¼ | Č | | ı | Ž |
| 173 | AD | 255 | 126400 | i | § | s | | - | - |
| 174 | AE | 256 | 127000 | << | << | << | | ® | Ž |
| 175 | AF | 257 | 127400 | >> | >> | >> | | Ž | Ž |

| | | | | | | | | |
|-----|----|-----|--------|---|---|---|---|---|
| 176 | B0 | 260 | 130000 | ⋮ | ⋮ | ⋮ | | |
| 177 | B1 | 261 | 130400 | ⋮ | ⋮ | ⋮ | ± | ± |
| 178 | B2 | 262 | 131000 | ⋮ | ⋮ | ⋮ | ± | ± |
| 179 | B3 | 263 | 131400 | | | | ± | ± |
| 180 | B4 | 264 | 132000 | ± | ± | ± | ± | ± |
| 181 | B5 | 265 | 132400 | ± | ± | Á | μ | Γ |
| 182 | B6 | 266 | 133000 | | | Â | ¶ | § |
| 183 | B7 | 267 | 133400 | π | π | Ë | · | · |
| 184 | B8 | 270 | 134000 | ± | ± | § | · | · |
| 185 | B9 | 271 | 134400 | ± | ± | ± | ± | ± |
| 186 | BA | 272 | 135000 | | | | § | § |
| 187 | BB | 273 | 135400 | ± | ± | ± | » | ± |
| 188 | BC | 274 | 136000 | ± | ± | ± | E | ± |
| 189 | BD | 275 | 136400 | ± | ± | ± | " | " |
| 190 | BE | 276 | 137000 | ± | ± | ± | Γ | ± |
| 191 | BF | 277 | 137400 | ± | ± | ± | ± | ± |

| Dec | Hex | Oct | Rotace | PC | Kam | PCLat | Koi | Win | ISOLat2 |
|-----|-----|-----|--------|----|-----|-------|-----|-----|---------|
| 192 | C0 | 300 | 140000 | ± | ± | ± | | Á | Á |
| 193 | C1 | 301 | 140400 | ± | ± | ± | á | Á | Á |
| 194 | C2 | 302 | 141000 | ± | ± | ± | | Á | Á |
| 195 | C3 | 303 | 141400 | ± | ± | ± | ± | Á | Á |
| 196 | C4 | 304 | 142000 | ± | ± | ± | d' | Á | Á |
| 197 | C5 | 305 | 142400 | ± | ± | ± | e | Á | Á |
| 198 | C6 | 306 | 143000 | ± | ± | ± | f | Á | Á |
| 199 | C7 | 307 | 143400 | | | | ch | Ç | Ç |
| 200 | C8 | 310 | 144000 | ± | ± | ± | u | Ç | Ç |
| 201 | C9 | 311 | 144400 | ± | ± | ± | i | É | É |
| 202 | CA | 312 | 145000 | ± | ± | ± | ù | Ej | Ej |
| 203 | CB | 313 | 145400 | ± | ± | ± | í | È | È |
| 204 | CC | 314 | 146000 | ± | ± | ± | r | È | È |
| 205 | CD | 315 | 146400 | = | = | = | ø | Í | Í |
| 206 | CE | 316 | 147000 | ± | ± | ± | n | Î | Î |
| 207 | CF | 317 | 147400 | ± | ± | ± | ó | Ï | Ï |

| | | | | | | | | | |
|-----|----|-----|--------|---|---|---|----|---|---|
| 208 | D0 | 320 | 150000 | ± | ± | ± | ø | Ð | Ð |
| 209 | D1 | 321 | 150400 | ± | ± | ± | a | Ñ | Ñ |
| 210 | D2 | 322 | 151000 | ± | ± | ± | r | Ñ | Ñ |
| 211 | D3 | 323 | 151400 | ± | ± | ± | s | Ó | Ó |
| 212 | D4 | 324 | 152000 | ± | ± | ± | d' | Ô | Ô |
| 213 | D5 | 325 | 152400 | ± | ± | ± | ú | Õ | Õ |
| 214 | D6 | 326 | 153000 | ± | ± | ± | í | Ö | Ö |
| 215 | D7 | 327 | 153400 | ± | ± | ± | e | × | × |
| 216 | D8 | 330 | 154000 | ± | ± | ± | a | Ř | Ř |
| 217 | D9 | 331 | 154400 | ± | ± | ± | y | Ů | Ů |
| 218 | DA | 332 | 155000 | ± | ± | ± | z | Ú | Ú |
| 219 | DB | 333 | 155400 | ± | ± | ± | | Û | Û |
| 220 | DC | 334 | 156000 | ± | ± | ± | · | Ü | Ü |
| 221 | DD | 335 | 156400 | ± | ± | ± | | Ý | Ý |
| 222 | DE | 336 | 157000 | ± | ± | ± | ^ | Ť | Ť |
| 223 | DF | 337 | 157400 | ± | ± | ± | | ß | ß |

| Dec | Hex | Oct | Rotace | PC | Kam | PCLat | Koi | Win | ISOLat2 |
|-----|-----|-----|--------|------------|------------|-------|-----|-----|---------|
| 224 | E0 | 340 | 160000 | α | α | Ó | ´ | í | í |
| 225 | E1 | 341 | 160400 | β | β | ß | Á | á | á |
| 226 | E2 | 342 | 161000 | Γ | Γ | Ò | | â | â |
| 227 | E3 | 343 | 161400 | π | π | Ñ | Č | ǎ | ǎ |
| 228 | E4 | 344 | 162000 | Σ | Σ | ń | Ď | ã | ã |
| 229 | E5 | 345 | 162400 | σ | σ | ň | Ě | í | í |
| 230 | E6 | 346 | 163000 | μ | μ | Š | Ř | é | é |
| 231 | E7 | 347 | 163400 | τ | τ | š | CH | ç | ç |
| 232 | E8 | 350 | 164000 | Φ | Φ | Ř | Û | č | č |
| 233 | E9 | 351 | 164400 | Θ | Θ | Ú | Í | é | é |
| 234 | EA | 352 | 165000 | Ω | Ω | ř | Û | ę | ę |
| 235 | EB | 353 | 165400 | δ | δ | Ů | Ł | ë | ë |
| 236 | EC | 354 | 166000 | ω | ω | ý | E | è | è |
| 237 | ED | 355 | 166400 | ϕ | ϕ | Ý | Ö | í | í |
| 238 | EE | 356 | 167000 | ϵ | ϵ | ť | Ñ | î | î |
| 239 | EF | 357 | 167400 | ∩ | ∩ | ´ | Ó | d' | d' |

| | | | | | | | | | |
|-----|----|-----|--------|-----|-----|---|---|---|---|
| 240 | F0 | 360 | 170000 | ≡ | ≡ | - | Ô | đ | đ |
| 241 | F1 | 361 | 170400 | ± | ± | " | Ă | ń | ń |
| 242 | F2 | 362 | 171000 | ≥ | ≥ | ˘ | Ř | ň | ň |
| 243 | F3 | 363 | 171400 | ≤ | ≤ | ˘ | Š | ó | ó |
| 244 | F4 | 364 | 172000 | ∫ | ∫ | ˘ | Ť | ô | ô |
| 245 | F5 | 365 | 172400 | ∫ | ∫ | § | Ú | ø | ø |
| 246 | F6 | 366 | 173000 | ÷ | ÷ | ÷ | | ö | ö |
| 247 | F7 | 367 | 173400 | ≈ | ≈ | ˘ | É | ÷ | ÷ |
| 248 | F8 | 370 | 174000 | ° | ° | ˘ | Á | ř | ř |
| 249 | F9 | 371 | 174400 | • | • | ˘ | Ý | ù | ù |
| 250 | FA | 372 | 175000 | · | · | ˘ | Ž | ú | ú |
| 251 | FB | 373 | 175400 | √ | √ | ř | | ů | ů |
| 252 | FC | 374 | 176000 | n | n | Ř | | ü | ü |
| 253 | FD | 375 | 176400 | 2 | 2 | ř | | ý | ý |
| 254 | FE | 376 | 177000 | ▪ | ▪ | ▪ | | ț | ț |
| 255 | FF | 377 | 177400 | | | | | ˘ | ˘ |

EBCDIC

Extended Binary Coded Decimal Interchange Code (EBCDIC)

- pro BCD kódování
- IBM mainframe operating systems (OS/390, VM, OS/400)

| | 00 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | A0 | B0 | C0 | D0 | E0 | F0 |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0 | | | | sp | & | - | | | | | | | | | | 0 |
| 1 | | | | | | / | | | | | | | A | J | | 1 |
| 2 | | | | | | | | | | | | | B | K | S | 2 |
| 3 | | | | | | | | | | | | | C | L | T | 3 |
| 4 | | | | | | | | | | | | | D | M | U | 4 |
| 5 | | | | | | | | | | | | | E | N | V | 5 |
| 6 | | | | | | | | | | | | | F | O | W | 6 |
| 7 | | | | | | | | | | | | | G | P | X | 7 |
| 8 | | | | | | | | | | | | | H | Q | Y | 8 |
| 9 | | | | | | | | | | | | | I | R | Z | 9 |
| A | | | | | | | | | | | | | : | | | |
| B | | | | | | | | | | | | | . | # | | |
| C | | | | | | < | * | % | @ | | | | | | | |
| D | | | | | | (|) | ' | | | | | | | | |
| E | | | | | | + | > | = | | | | | | | | |
| F | | | | | | ? | " | | | | | | | | | |

Starší než ASCII, před rokem 1960. Hlavně pro děrné štítky.

Opakování a shrnutí

Máme-li v 1 bajtu uloženou binární kombinaci

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

co všechno tato kombinace představuje?

- osmičkové celé číslo bez znaménka: **350**
- šestnáctkové celé číslo bez znaménka: **0e8**
- desítkové celé číslo bez znaménka: **232**
- desítkové celé číslo v přímém kódu: **-104**
- desítkové celé číslo v inverzním kódu: **-23**
- desítkové celé číslo ve dvojkovém doplňkovém kódu: **-24**
- dvě BCD číslice to nemohou být (nemáme číslici E)
- písmeno 'h' v ASCII kódu se sudou paritou
- písmeno 'č' v kódování Windows-1250 nebo ISO-8859-2

Big-Endian a Little-Endian

Který bajt slova je uložen na nižší adrese?

Big-Endian – Big end first Bajt nejvyššího řádu je uložen na nejnižší adrese. Příklad uložení čísla 123456_{16} ve 32bitovém slově big-endian:

| | | | | |
|--------|----|----|----|----|
| Adresa | 00 | 01 | 02 | 03 |
| | 00 | 12 | 34 | 56 |

Používají např. sálové počítače IBM 370, Motorola 68000 a Sun Sparc.

Little-Endian – Little end first Bajt nejnižšího řádu je uložen na nejnižší adrese. Příklad uložení čísla 123456_{16} ve 32bitovém slově little-endian:

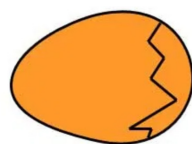
| | | | | |
|--------|----|----|----|----|
| Adresa | 00 | 01 | 02 | 03 |
| | 56 | 34 | 12 | 00 |

Používá např. INTEL x86(současná PC), DEC Alpha.

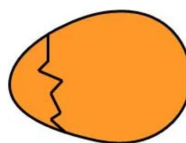
Middle-Endian – Pořadí bajtů 3 – 4 – 1 – 2 nebo 2 – 1 – 4 – 3.

Bi-Endian – Např. procesor PowerPC (Power Macintosh) umožňuje pracovat s Big-Endian i Little-Endian.

- Bity v bajtu jsou big-endian bez ohledu na pořadí bajtů.
- E-mailová adresa je little-endian. Americký způsob zápisu data mm/dd/yy je middle-endian, evropský dd/mm/yy little-endian, japonský yy/mm/dd big-endian pro evropské/americké datum.
- Označení big-endian a little-endian převzato z románu Jonathana Swifta Gulliverovy cesty: nepochopené nařízení vládcce rozbít vejce na menším konci, zatímco tradičně se vejce rozbíjelo na konci větším.



BIG ENDIAN - The way people always broke their eggs in the Lilliput land



LITTLE ENDIAN - The way the king then ordered the people to break their eggs

Způsoby popisu zvoleného vnějšího kódování

Nejčastěji parametrem **Charset** v **MIME type** (Multipurpose Internet Mail Extensions) dle RFC 2045 a následujících.

Např. použití v hlavičce e-mailu:

Content-Type: text/plain; charset=us-ascii

Content-Type: text/plain; charset=iso-8859-2

Content-Type: text/plain; charset=windows-1250

Content-Type: text/plain; charset=UTF-8

Vícebajtová kódování

UNICODE • <http://www.unicode.org/>

- standard definuje i název znaku, převodní tabulky malá-velká písmena atp.
- nevýhody: násobná délka textu, větší znaková sada

UCS-2 • (Universal Coded Character Set)

- základní způsob zápisu Unicode znaků, 2 bajty na znak
- zastaralé, používá se UTF-16

UCS-4 • 4 bajty na znak

- zastaralé, používá se UTF-32, ale je to totéž

UTF-8 (Unicode Transformation Format)

- Nejpoužívanější zobrazení Unicode znaků.
- Pokud je znak ve standardním ASCII-7, zobrazí se beze změny v 1 bajtu, tzn. nejvyšší bit bajtu je roven nule.
- Tzn. že UTF-8 je zdola kompatibilní s ASCII.
- Pokud znak není v ASCII, je zapsán dvěma až čtyřmi bajty, tzn. 21 datových bitů, cca 2 mil. znaků (před Unicode 4.0 šesti bajty).
 - 1. bajt: počet jedniček zleva vyjadřuje délku sekvence, následuje nula, která je oddělovač.
 - Další bajty: v nejvyšších dvou bitech vždy 10.

| Unicode kód od – do | Binární zápis znaku v UTF-8 |
|-----------------------|-------------------------------------|
| 0000 0000 - 0000 007F | 0xxxxxxx |
| 0000 0080 - 0000 07FF | 110xxxxx 10xxxxxx |
| 0000 0800 - 0000 FFFF | 1110xxxx 10xxxxxx 10xxxxxx |
| 0001 0000 - 001F FFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx |

- České znaky mají malé hodnoty, lze všechny zapsat dvěma bajty.

- Příklad: 'Příliš'

| | | | | | |
|----|-------|-------|----|----|-------|
| 50 | c5 99 | c3 ad | 6c | 69 | c5 a1 |
| P | ř | í | l | i | š |

- ř: 1100 0101 1001 1001 'ř' v unicode U+0159 (hexa)
- Ordinální hodnota se zapisuje $U + 00ED$ (tj. U+hexa číslice).

| | | | | | | | | |
|---|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 5 | ą 0105 | ě 0115 | ĥ 0125 | ĵ 0135 | Ń 0145 | ř 0155 | ť 0165 | ŵ 0175 |
| 6 | Ć 0106 | Ė 0116 | Ħ 0126 | Ḷ 0136 | ŋ 0146 | Ŕ 0156 | Ŧ 0166 | Ŷ 0176 |
| 7 | ć 0107 | ė 0117 | ħ 0127 | ḷ 0137 | ņ 0147 | ŗ 0157 | ț 0167 | ŷ 0177 |
| 8 | Ĉ 0108 | Ė 0118 | Ĩ 0128 | κ 0138 | ň 0148 | Ř 0158 | Ũ 0168 | Ÿ 0178 |
| 9 | ĉ 0109 | ę 0119 | ĩ 0129 | ł 0139 | ń 0149 | ř 0159 | ũ 0169 | ź 0179 |
| A | Ċ 010A | Ě 011A | Ī 012A | Í 013A | Ń 014A | Ś 015A | Ū 016A | Ż 017A |
| B | ċ 010B | ě 011B | ī 012B | ł 013B | ŋ 014B | ś 015B | ū 016B | ż 017B |

Postup transformace znaku ř do UTF-8

Znak ř má v UNICODE ordinální hodnotu $U + 0159$ (je zapsána hexadecimálně).

Ordinální hodnota znaku ř binárně:

0000 0001 0101 1001

Binární hodnota nemá více než 11 významných bitů, proto ji lze uložit do 2bajtového UTF-8. Dvoubajtové UTF-8 se vkládá do následující formy:

110x xxxx **10**xx xxxx

Binární ordinální hodnotu znaku ř dosadíme do dvoubajtové formy na místa znaků x. Začínáme vždy zprava, tj. od řádu 2^0 . Nadbytečné levostranné nuly ignorujeme.

1100 0101 1001 1001

Převáděno do hexadecimální soustavy: **C5 99**

- Pomocí UTF-8 lze zobrazit maximálně 2^{21} možných znaků.
- Jak rozpoznám úvodní bajt od bajtů následujících?
- Nikdy nejsou pro zakódování znaku použity bajty s hodnotou 0FE (11111110) a 0FF (11111111)
- Proto se někdy (Windows) používá kód $U + FFFF$ jako označení, že následuje UTF-8. Nazývá se **BOM** (Byte-Order Mark). Nalezne-li se $U + FFFE$, jde o Little-Endian uložení. Ovšem pro UTF-8 nemá endianita smysl, protože se ukládá po jednotlivých bajtech.

UTF-16 • rozšíření základní šířky z 8 na 16 bitů

UTF-32 • rozšíření základní šířky z 8 na 32 bitů

UTF-7 • pro sedmibitový přenos e-mailem (jako Base64)

Detekční kódy

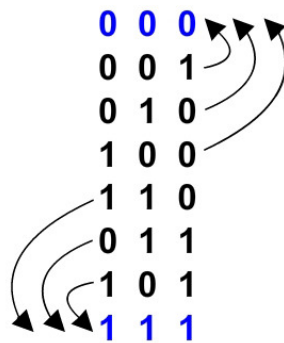
- 1 chyba znamená inverzi hodnoty jednoho bitu
- zavedení nadbytečnosti (redundance)
- *parita*
 - sudá (even)
 - lichá (odd)
- Kód 2 z 5:

| | | |
|---|--|-----------|
| 0 | | 0 0 0 1 1 |
| 1 | | 0 0 1 0 1 |
| 2 | | 0 0 1 1 0 |
| 3 | | 0 1 0 1 0 |
| 4 | | 0 1 1 0 0 |
| 5 | | 1 0 1 0 0 |
| 6 | | 1 1 0 0 0 |
| 7 | | 0 1 0 0 1 |
| 8 | | 1 0 0 0 1 |
| 9 | | 1 0 0 1 0 |

Zachytí všechny 1násobné chyby a 40 % 2násobných chyb.

Opravné kódy

Př.: Ztrojení



Dokáže opravit 1 chybu a detekovat 2 chyby.

3.3 Kódová vzdálenost a její znázornění

Kódová (Hammingova) vzdálenost

Definice

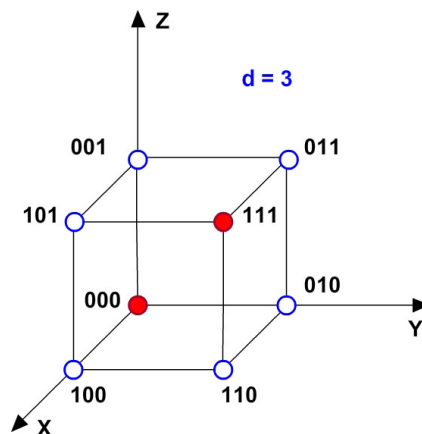
Kódová vzdálenost d = počet bitů, v nichž se liší dvě sousední platné kódové kombinace.

Znázornění pomocí Hammingovy krychle:

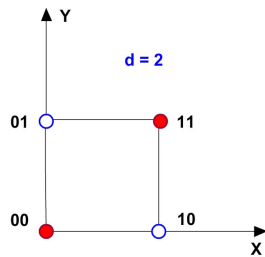
- trojrozměrná (xyz)
- dvojrozměrná (xy)
- jednorozměrná (x)

- trojrozměrná

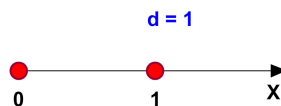
Dvě platné kódové kombinace (xyz): 000 a 111



- dvojrozměrná



- jednorozměrná



3.4 Vztah kódové vzdálenosti a počtu chyb

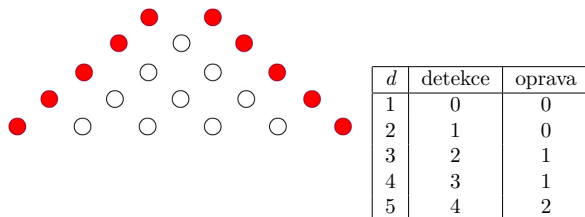
Detekce a oprava k chyb

Vztahy

Detekce k chyb: $d \geq k+1$

Oprava k chyb: $d \geq 2k+1$

- Jiné znázornění:



4 Obvody

4.1 Booleova algebra

Booleova algebra

- GEORGE BOOLE (1815 - 1864) - Irský matematik, v roce 1854 zvláštní druh algebry (uplatnění až v roce 1938).
- Boolova algebra je nauka o operacích na množině $\{0,1\}$.

Definice

Def.: B.A. je množina B o alespoň 2 prvcích nad níž jsou definovány operace sčítání, násobení a negace splňující tyto axiomy:

- (předp.: $a, b, c \in B$) :
 - $a + b \in B$
 - $a \cdot b \in B$
- Existuje prvek 0, pro který platí: $a + 0 = a$
- Existuje prvek 1, pro který platí: $a \cdot 1 = a$

Pokračování definice

- Komutativní zákon:
 - $a + b = b + a$
 - $a \cdot b = b \cdot a$
- $a + (b \cdot c) = (a + b) \cdot (a + c)$
- $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
- Pro každý prvek a existuje prvek $\bar{a} \in B$:
 - $a \cdot \bar{a} = 0$
 - $a + \bar{a} = 1$

Základní operace

B. A. užívá jen 3 základní operace:

- Logický (Booleův) součin *AND* $\wedge \cap \times$.
- Logický (Booleův) součet *OR* $\vee \cup +$
- Negace \bar{x} *NOT* $\neg \sim$ (*před operandem*)

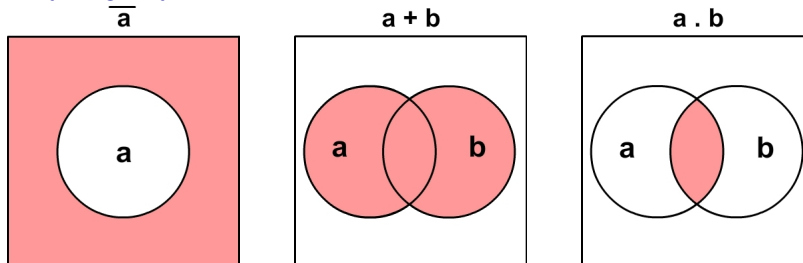
Způsoby popisů:

- Pravdivostní tabulka
- Graficky v rovině = Vennovy diagramy
- Matematický aparát

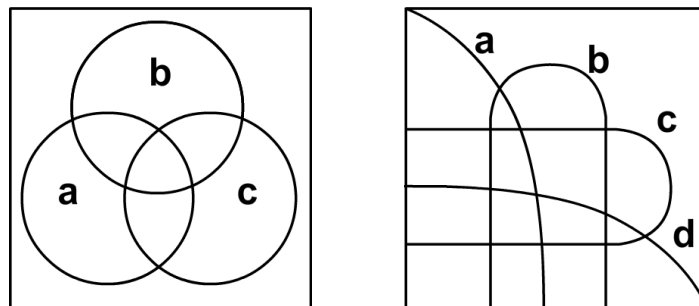
Pravdivostní tabulka:

| a | b | $a + b$ | $a \cdot b$ | \bar{a} |
|-----|-----|---------|-------------|-----------|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Vennovy diagramy



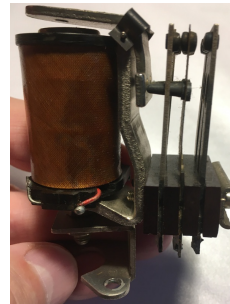
Vennovy diagramy



Využití Booleovy algebry

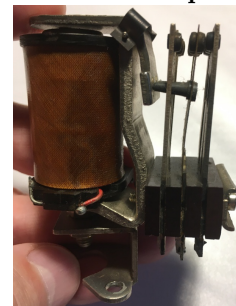
- 1938 - Shannon pro popis průchodnosti kontaktního zapojení
- Základní prvek: **relé**

Relé **rozepnuto**.



Cívkou neprotéká proud, pružina kotvičku odtahuje od cívky.

Relé **sepnuto**.

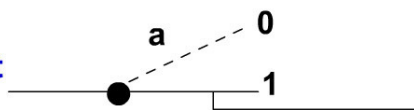


Cívkou protéká proud a vzniklé magnetické pole přitáhne kotvičku k cívice.

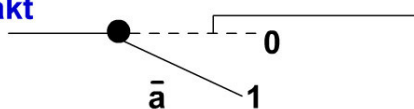
Využití Booleovy algebry

- Kontakt ovládaný dvouhodnotovou proměnnou a

1. spínací kontakt



2. rozpínací kontakt

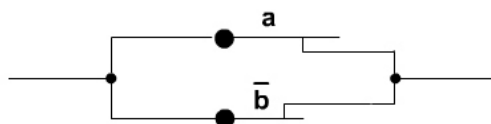


Zapojení kontaktů

sériové $a \cdot \bar{b}$



paralelní $a + \bar{b}$

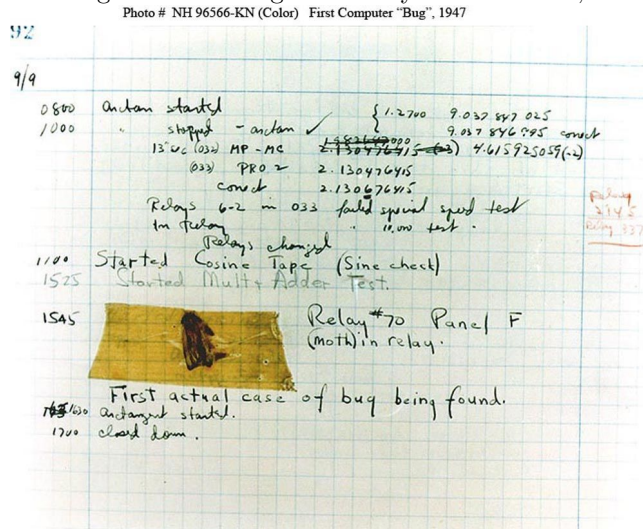


Příklady:

- Pračka smí začít topit, když je napuštěn dostatek vody (a) a současně nejsou otevřena dvířka (b), tj. $a \cdot \bar{b}$
- Automobil musí automaticky rozsvítit světla, když prší (a) nebo není dostatek světla (b), tj. $a + \bar{b}$

Jedna z nevýhod použití relé

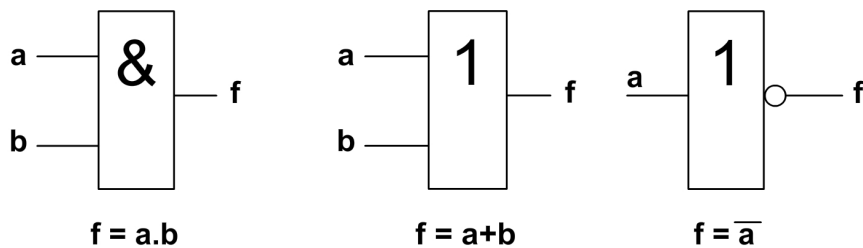
Thomas Edison reported “bugs” in his designs as early as the 1800s, but this was the first bug



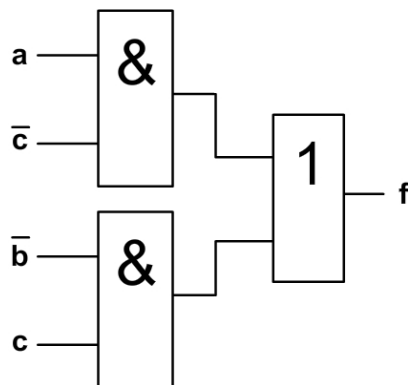
identified in a computer.

4.2 Obvodové znázornění Booleovy algebry

Obvodové znázornění Booleovy algebry



Př: $f = a \cdot \bar{c} + \bar{b} \cdot c$



Minimalizace počtu operací B-algebry

Proč minimalizovat?

Méně výrazů znamená menší počet součástek, nižší cenu a rychlejší zpracování.

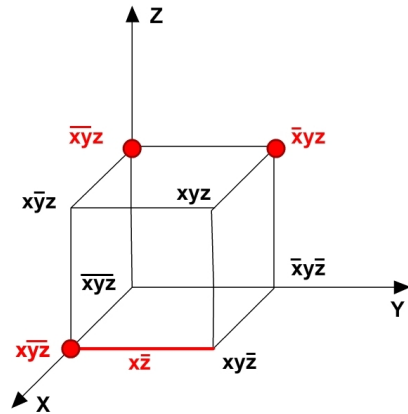
Jak minimalizovat?

- **1. Matematické úpravy**

Př: $\bar{x}yz + \bar{x}\bar{y}z = \bar{x}z(\bar{y} + y) = \bar{x}z$

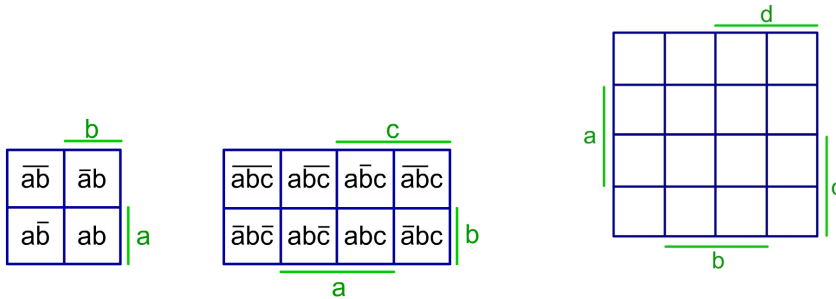
• 2. Využitím jednotkové krychle

Př: $f = \overline{x}yz + \overline{x}yz + x\overline{y}z + x\overline{z}$



$f = \overline{x}z + x\overline{z}$

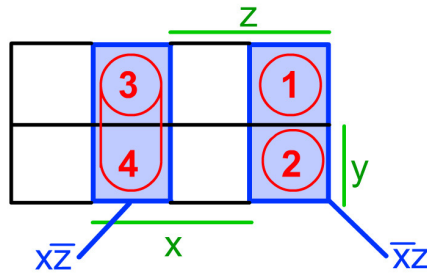
• 3. Karnaughova mapa - normalizací Vennova diagramu



Pro vyšší řády nejsou

souvislé prostory proměnných.

Př: $f = \overline{x}yz + \overline{x}yz + x\overline{y}z + x\overline{z}$



$f = \overline{x}z + x\overline{z}$

B-algebra je nevhodná pro technickou realizaci - příliš velký počet operací (\cdot , $+$, $-$)

4.3 Shefferova algebra, Peirceova algebra

Shefferova algebra

- Je vybudovaná na jedné logické funkci = *negace logického součinu NAND*
- Pro libovolný počet proměnných $f = \overline{x \cdot y}$
- Pravidla:
 - $\overline{x \cdot x} = \overline{x}$
 - $\overline{x \cdot 0} = 1$

- $\overline{x \cdot 1} = \bar{x}$
- $\overline{\overline{x \cdot y} \cdot 1} = \overline{\overline{x \cdot y}} = x \cdot y$
- $\overline{\overline{x \cdot x} \cdot \overline{y \cdot y}} = \overline{\overline{x} \cdot \overline{y}} = x + y$

- Pomocí operace NAND lze realizovat všechny operace Booleovy algebry
- Platí zákon komutativní: $\overline{x \cdot y} = \overline{y \cdot x}$
- *Neplatí* zákon asociativní: $\overline{\overline{x \cdot y} \cdot z} \neq \overline{x \cdot \overline{y \cdot z}} \neq \overline{x \cdot y \cdot z}$

Peirceova algebra

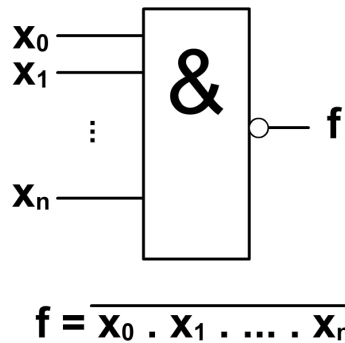
- Vystavěna na operaci *negace logického součtu* NOR - obdobné jako S-algebra.
- **Převod minimalizované formy B-algebry na S-algebru:**

Opakovanou aplikací de Morganových pravidel - $a + b = \overline{\overline{a} \cdot \overline{b}}$

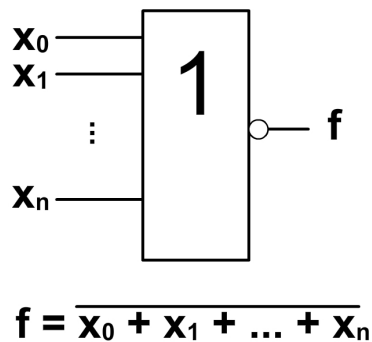
Př: $f = \overline{\overline{b \cdot c} + \overline{a \cdot c} + \overline{abd}}$

$$f = \overline{\overline{b \cdot c} + \overline{a \cdot c} + \overline{abd}} = \overline{\overline{b \cdot c} \cdot \overline{a \cdot c} \cdot \overline{abd}}$$

Obvodové znázornění S-algebry



Obvodové znázornění P-algebry



4.4 Fyzikální postata signálů, zakázané pásmo

Fyzikální podstata signálů

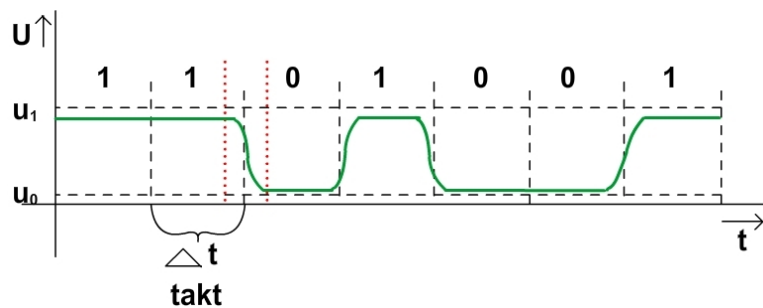
| LOGIC. HODNOTA | HLADINOVÉ | | | IMPULSOVÉ | |
|----------------|----------------------|-----------------|------------------|--------------------------|----------------------|
| | RELÉ | | | | |
| 1 | Proud prochází | Větší proud | Vyšší napětí | Přítomnost impulsu | Kladná polarita |
| 0 | Proud neprochází | Menší proud | Nižší napětí | Nepřítomnost impulsu | Záporná polarita |

Fyzikální podstata signálů

| LOGIC. HODNOTA | AMPLITUDA | KMITOČET | FÁZE |
|----------------|---------------------|--------------------|----------|
| 1 | Vyšší amplituda | Vyšší kmitočet | Fáze |
| 0 | Nižší amplituda | Nižší kmitočet | Fáze |

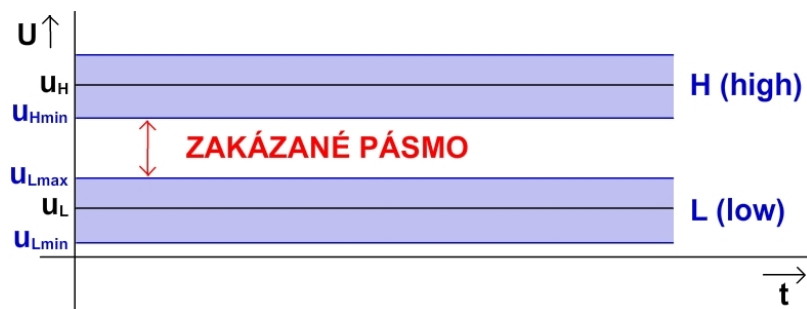
Průběh signálu

Změny signálu v praxi nejsou diskrétní, nýbrž spojitě.



Zakázané pásmo

Hodnotu signálu nelze číst (vzorkovat), když se hodnota signálu mění.



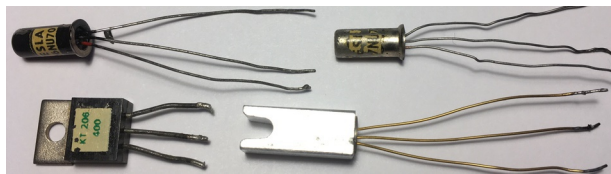
- Hodnoty jsou stanoveny pro každou výrobní technologii zvlášť.

- $L \sim 0$ $H \sim 1$ – Pozitivní logika
- $L \sim 1$ $H \sim 0$ – Negativní logika

4.5 TTL, invertor v TTL, NAND a NOR v TTL

Technologie TTL (transistor–transistor logic)

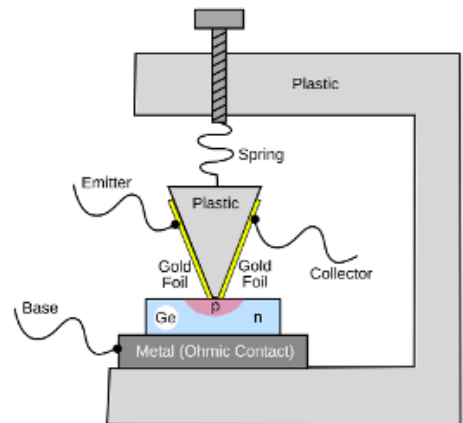
- Základní stavební prvek je tranzistor NPN.
- Parametry TTL:
 - napájecí napětí + 5V
 - $L < 0,8V$ $L \sim 0,4V$
 - $H > 2,0V$ $H \sim 2,4V$
- Tři vývody: báze, kolektor, emitor.
- Malou změnou proudu mezi bází a emitorem ovlivňujeme velké změny proudu mezi kolektorem a emitorem.



Napájecí napětí se s postupem vývoje technologií snižuje: 3,3 V, 2,5 V, 1,8 V a 1,2 V.

První tranzistor 1947

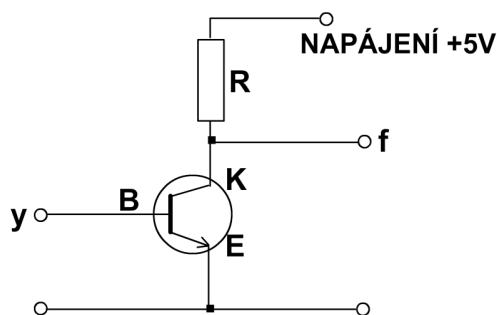
John Bardeen, Walter Brattain a William Shockley v Bell Laboratories, rok 1947. Velikost cca 10 cm.



Základem germánium. Později křemík.

Invertor v TTL

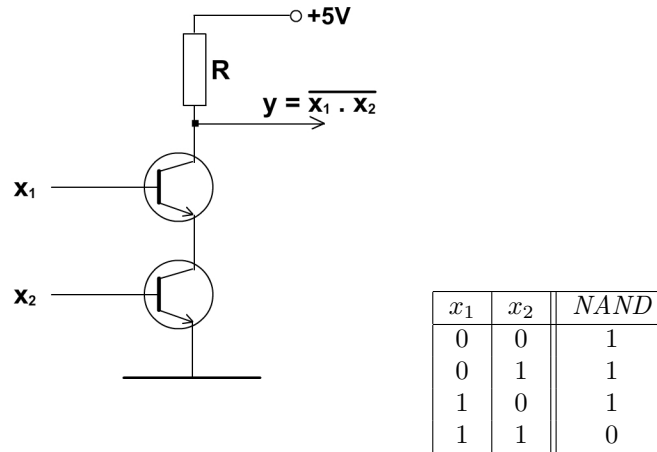
$$f = \bar{y}$$



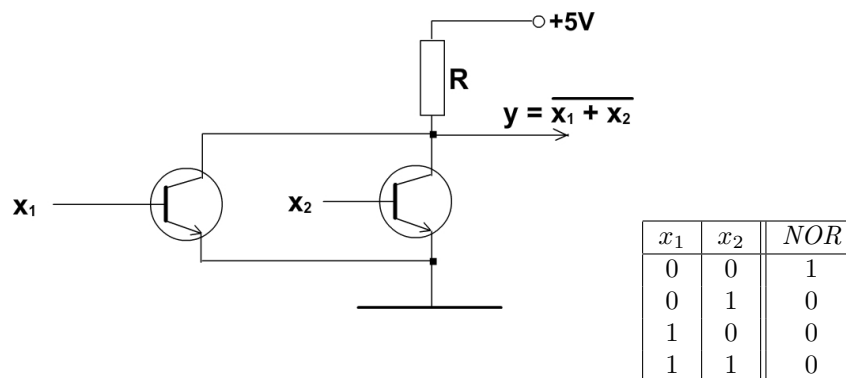
Nízká (do 0,8 V) přivedená na bázi uzavírá průchod kolektor-emitor, čímž se na výstup f dostane napájecí napětí.

Vysoká (od 2,0 V) přivedená na bázi otevírá průchod kolektor-emitor, čímž se výstup f "zkratuje" na úroveň země, tj. 0 V.

NAND pomocí dvou tranzistorů

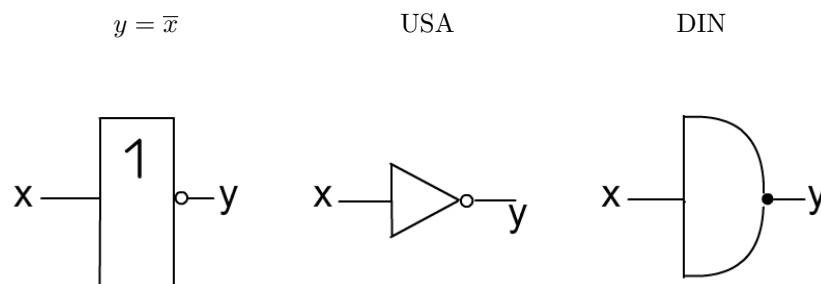


NOR pomocí dvou tranzistorů



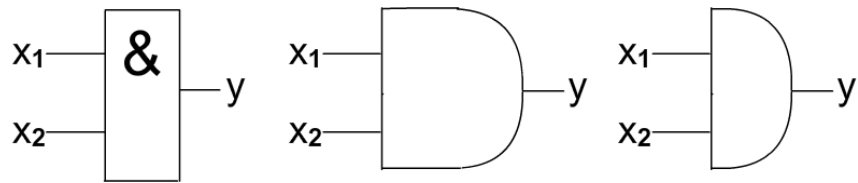
4.6 Kombinační logické obvody - základní logické členy

Invertor



AND



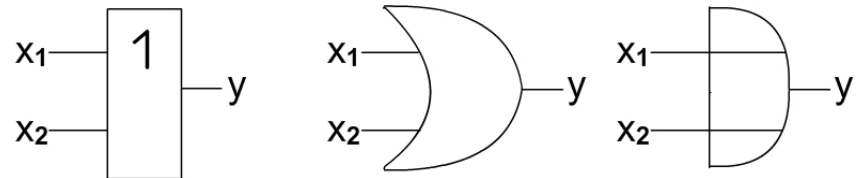


OR

$$y = x_1 + x_2$$

USA

DIN

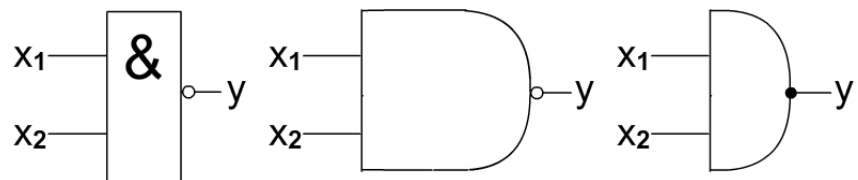


NAND

$$y = \overline{x_1 \cdot x_2}$$

USA

DIN

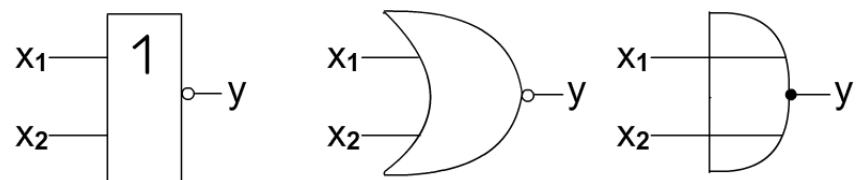


NOR

$$y = \overline{x_1 + x_2}$$

USA

DIN



4.7 Kombinační logické obvody - ostatní logické členy

Nonekvivalence – XOR

- Značení:
 - \neq
 - \oplus
 - =1
 - M2

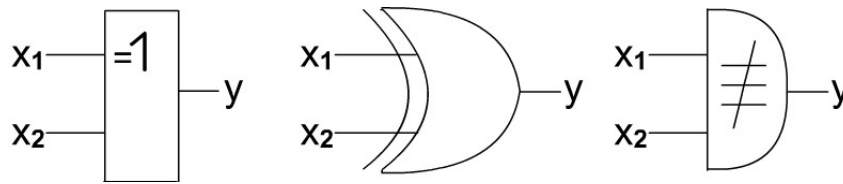
| x_1 | x_2 | y |
|-------|-------|-----|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Nonekvivalence – XOR

$$y = \overline{x_1 \oplus x_2}$$

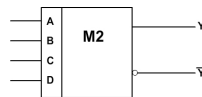
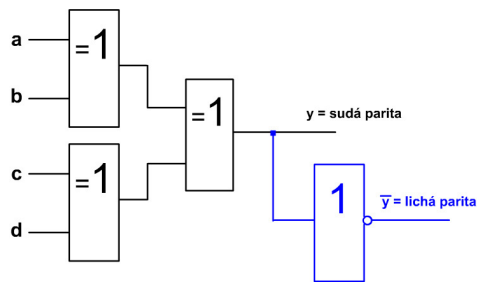
USA

DIN



Využití obvodu nonekvivalence

- Př: Generátor parity : $y = a \oplus b \oplus c \oplus d = (a \oplus b) \oplus (c \oplus d)$

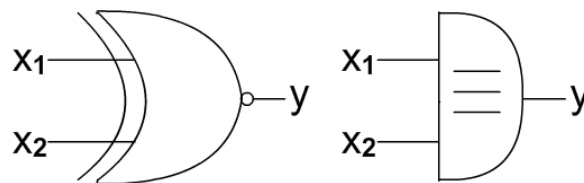


- Schématická značka:

Ekvivalence - NOXOR

USA

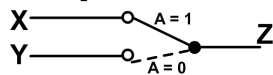
DIN



4.8 Logické obvody

Logické obvody

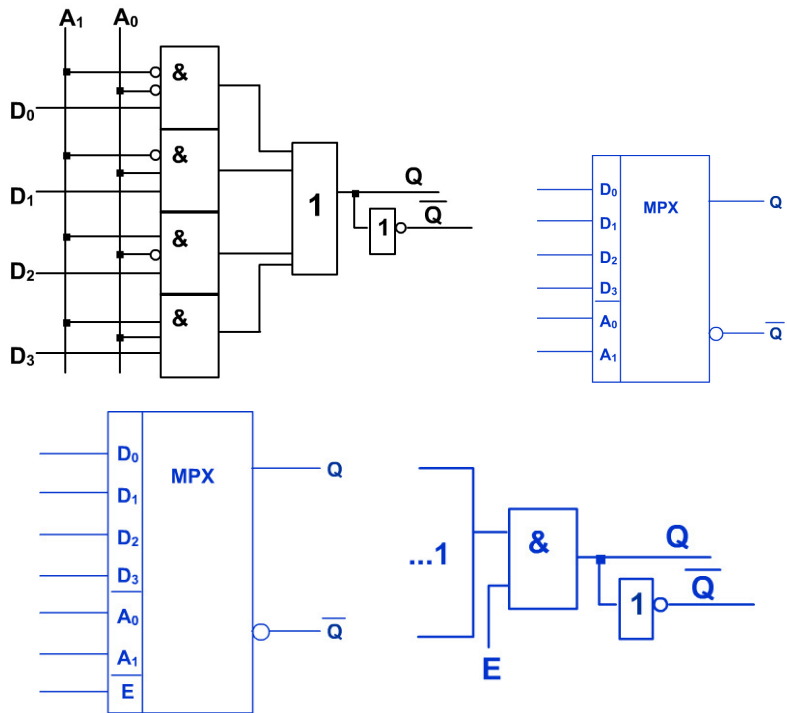
- **Multiplexor** : $Z = A \cdot X + \bar{A} \cdot Y$



- 4vstupý multiplexor – 4 datové vstupy, 2 adresové vstupy

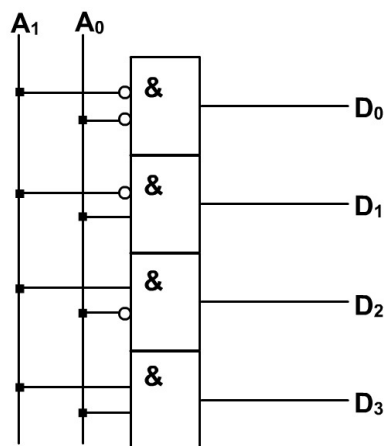
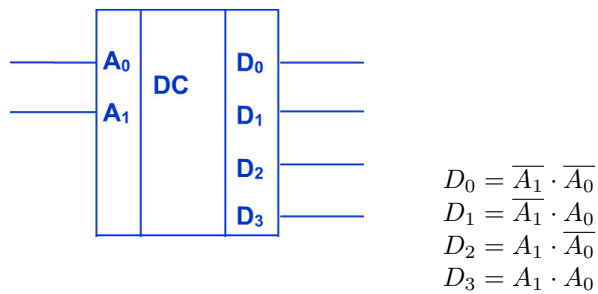
| A_1 | A_0 | Q |
|-------|-------|-------|
| 0 | 0 | D_0 |
| 0 | 1 | D_1 |
| 1 | 0 | D_2 |
| 1 | 1 | D_3 |

- $Q = \bar{A}_1 \cdot \bar{A}_0 \cdot D_0 + \bar{A}_1 \cdot A_0 \cdot D_1 + A_1 \cdot \bar{A}_0 \cdot D_2 + A_1 \cdot A_0 \cdot D_3$

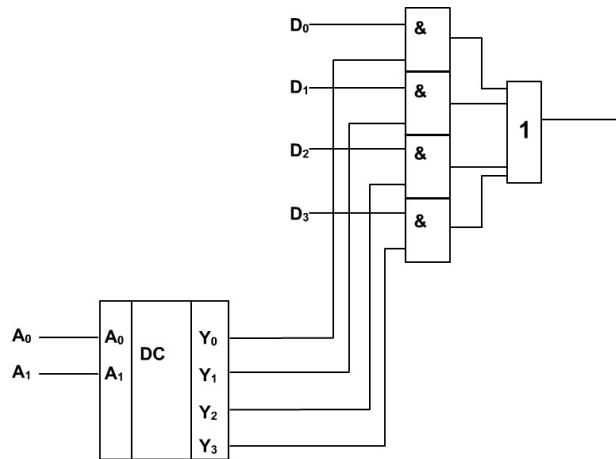


4.9 Dekodér, sčítačky

Dekodér



Realizace MPX pomocí dekodéru:



Sčítačky

- **Sčítačka MODULO 2** $x + y = z$

– *tabulka*

| x | y | z |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

– *rovnice*

$$z = \bar{x} \cdot y + x \cdot \bar{y}$$

Sčítačky

- **Polosčítačka**

– *tabulka*

| x | y | S | P |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

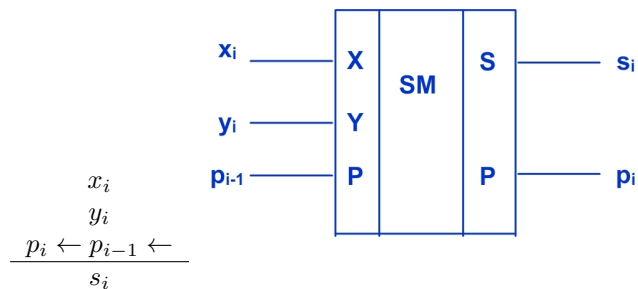
– *rovnice*

$$S = \bar{x} \cdot y + x \cdot \bar{y}$$

$$P = x \cdot y$$

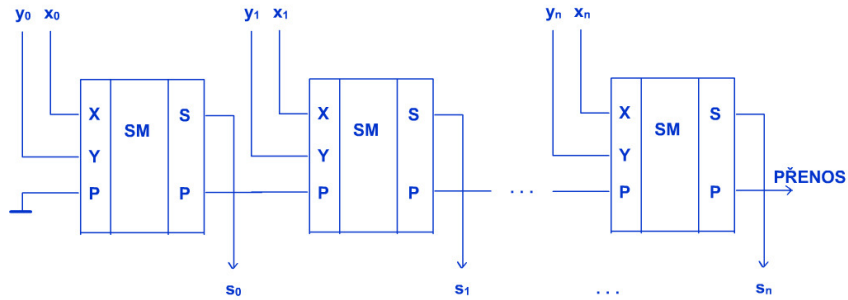
4.10 Úplná sčítačka, vícemístná sčítačka

Úplná sčítačka pro jeden binární řád



| x_i | y_i | p_{i-1} | s_i | p_i |
|-------|-------|-----------|-------|-------|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Vícemístná sčítačka



- **Př.:** Navrhněte sčítačku pro 32 řádů a zapište pravdivostní tabulku (2×32 vstupů, 32 výstupů). kniha 45 ř./s. a 500 stran = 22 500 ř.

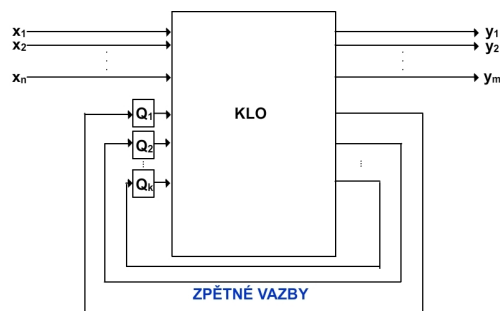
knihovna na celou stěnu: 2 000 knih
knihoven : 400 miliard
protože pravd. tabulka by měla 2^{64} řádků = 18×10^{18}

4.11 Sekvenční logické obvody, klopný obvod RS

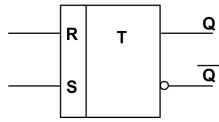
Sekvenční logické obvody

Předchozí obvody (and, or, not, xor, multiplexor, sčítačky apod.) byly tzv. **kombinační logické obvody**, ve kterých výstup závisel jen na aktuální hodnotě vstupů.

Sekvenční logické obvody mají výstup závislý nejenom na aktuální hodnotě vstupů, ale také i na posloupnosti změn, které předcházely.



- **Základní paměťový člen:** Klopný obvod RS
 - R ...RESET (nulování)
 - S ...SET (nastavení)



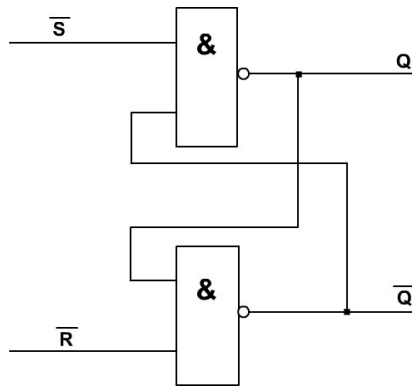
| | | | |
|---|---|-----------------|----------------------|
| R | S | Q_i | \overline{Q}_i |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 0 | Q_{i-1} | \overline{Q}_{i-1} |
| 1 | 1 | <i>zakázaný</i> | <i>stav</i> |

Obvod řízený jedničkami

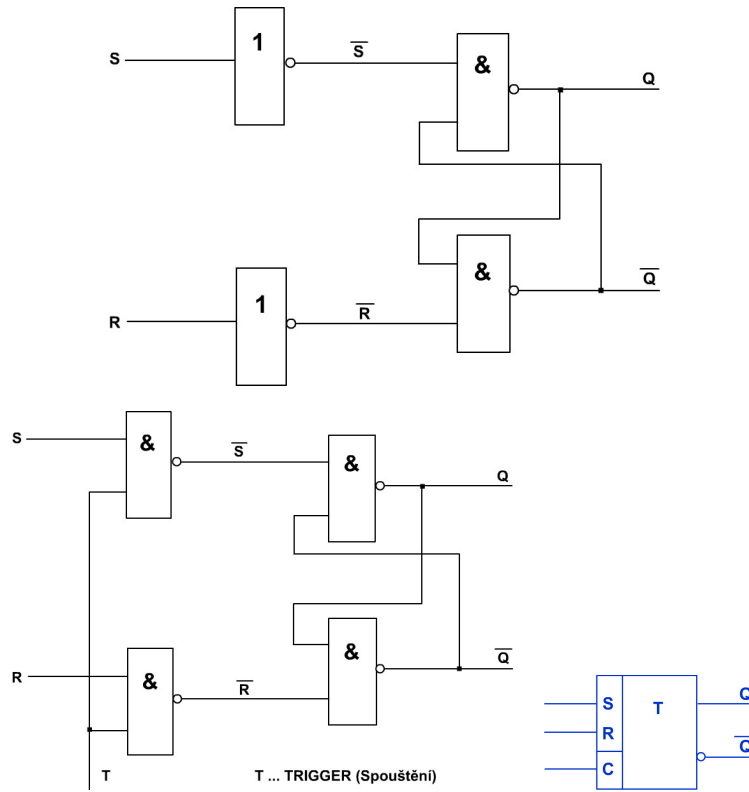
- RS řízený nulami:

| | | |
|----------------|----------------|----------------------|
| \overline{R} | \overline{S} | Q_i |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | Q_{i-1} |
| 0 | 0 | <i>Zakázaný stav</i> |

RS řízený nulami



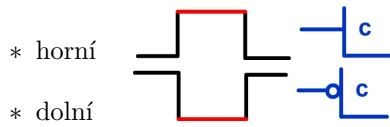
RS řízený jedničkami



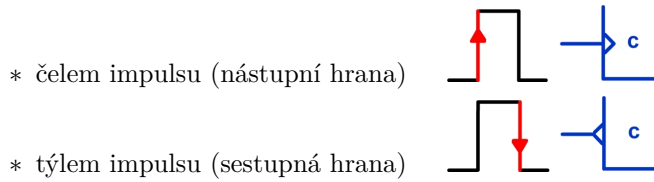
Obvod RS řízený jedničkami s časovou synchronizací.

• Klopný obvod řízený

– hladinou



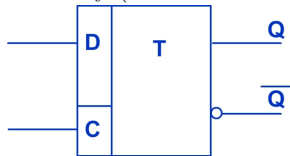
– hranou



4.12 Klopný obvod D

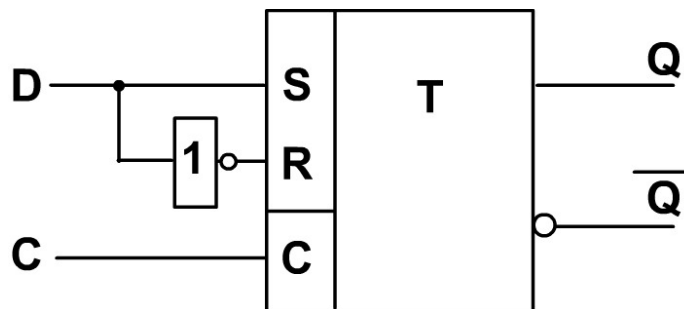
Klopný obvod D

- Ddelay (vzorkovací K.o.)

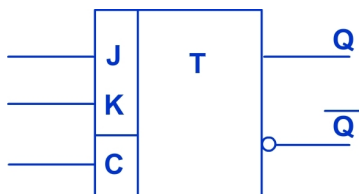


| D | C | Q_i |
|---|---|-----------|
| 1 | ⏏ | 1 |
| 0 | ⏏ | 0 |
| ? | — | Q_{i-1} |

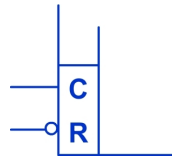
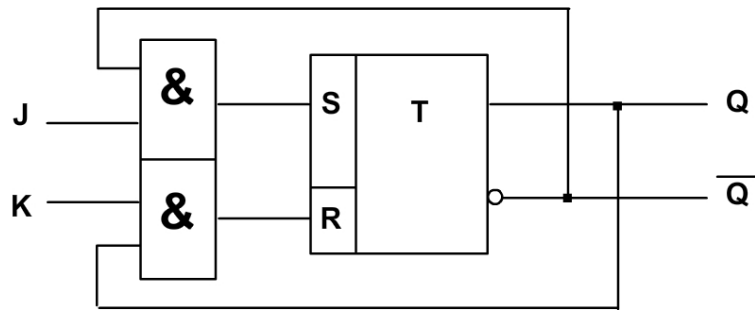
- Realizace D-KO pomocí RS:



Klopný obvod JK



| J | K | Q_i |
|---|---|----------------------|
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 0 | Q_{i-1} |
| 1 | 1 | $\overline{Q_{i-1}}$ |



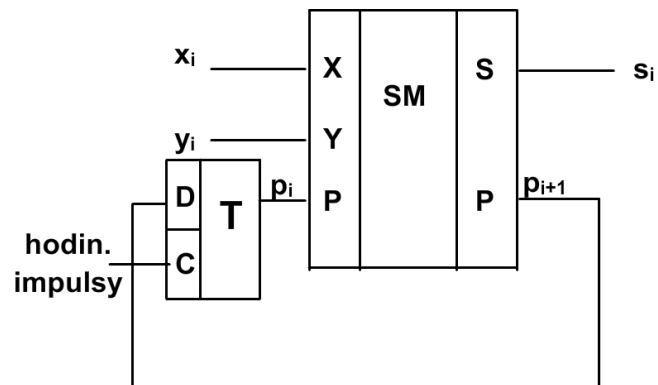
U většiny KO navíc:

R....reset

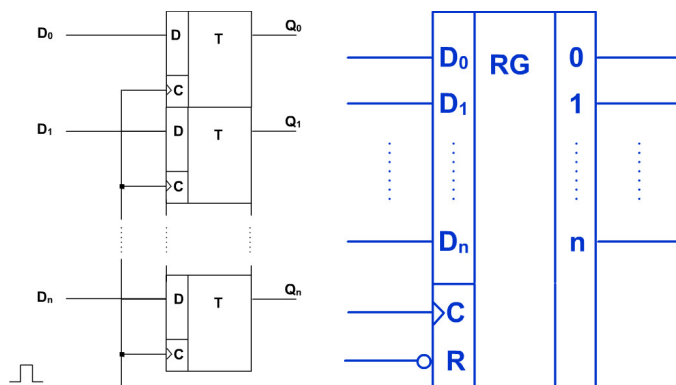
4.13 Typické sekvenční obvody

Typické sekvenční obvody v počítačích

- Sériová sčítačka:



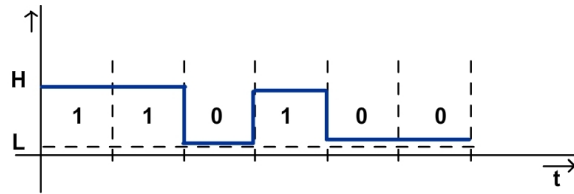
- Paralelní registr = střadač:



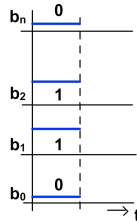
4.14 Přenos informací v systému

Přenos informací v systému

- Sériový:



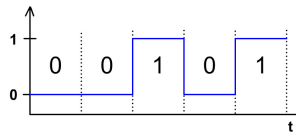
- Paralelní:



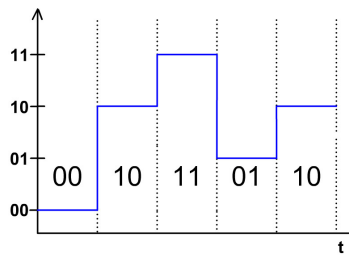
Převod sériová informace → paralelní pomocí posuvného registru

Sériový přenos

Dvoustavová komunikace

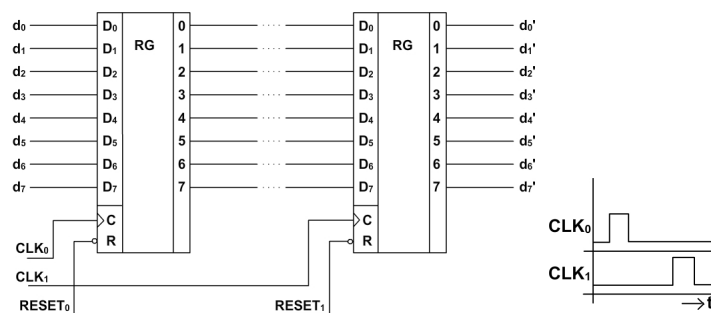


Čtyřstavová komunikace

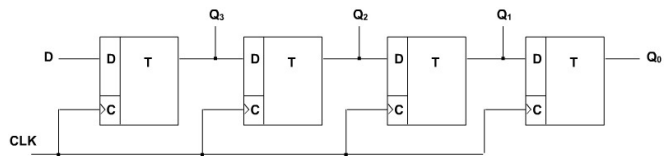


Přenosová rychlost

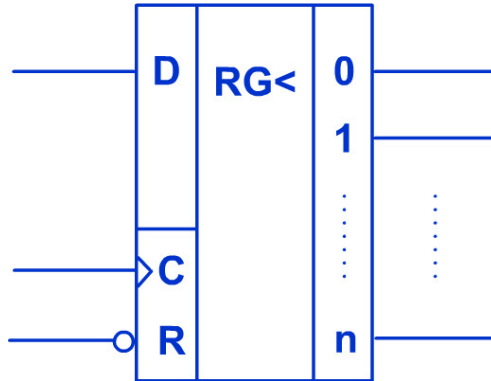
- v bitech za sekundu
- v počtu změn stavu za sekundu (baud rate, Bd)
- **Uvnitř počítače přenos paralelně pomocí sběrnice.**
Využití paralelních registrů:



- Sériový registr (posuvný registr):

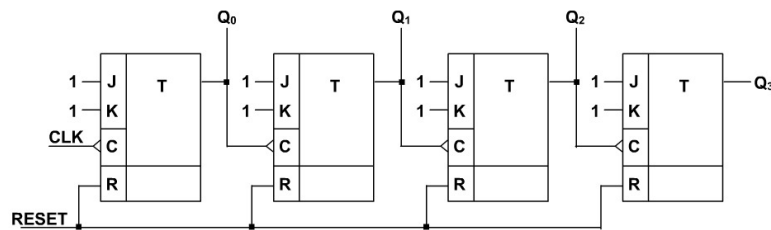


Jedním taktém signálu CLK se informace posune o *jeden* D-KO.

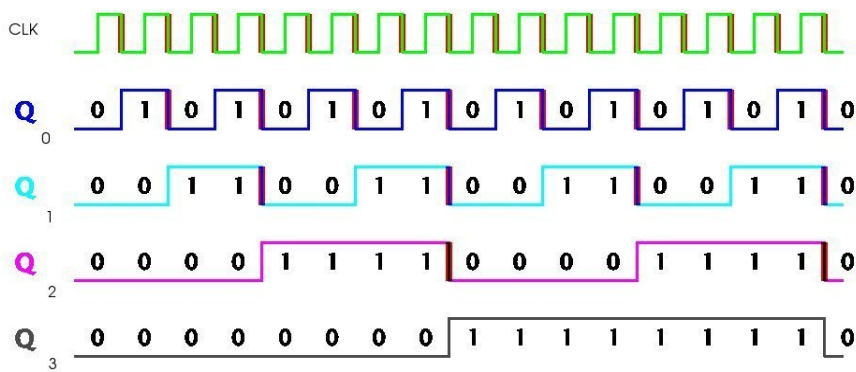


Sériový registr (posuvný registr)

- Čítače:

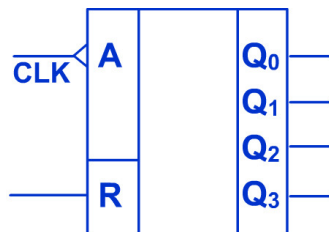


Dvojkový čítač 0...15, 0...15, ...



Řízen sestupnou hranou

impulsu! Každá sestupná hrana vyvolá reakci.



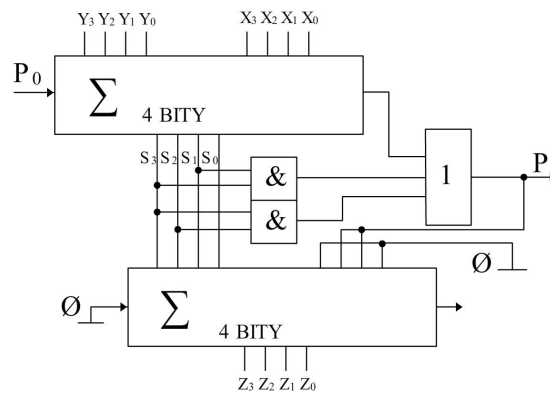
Dvojkový čítač 0...15

4.15 Sčítačka v BCD kódu

Sčítačka v BCD kódu

- Součet dvou čísel vyjádřený:

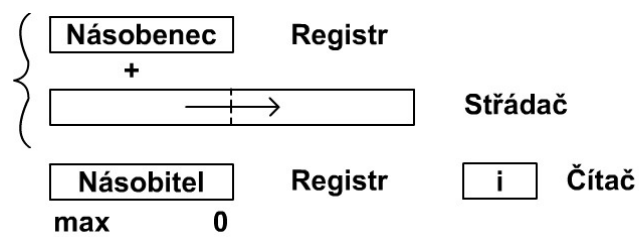
| Dvojkově: | BCD: | Desítkově: |
|-----------|------|------------|
| 0 | 0000 | 0 |
| 0 | 0001 | 1 |
| ... | ... | ... |
| 0 | 1001 | 9 |
| 0 | 1010 | 10 |
| 0 | 1011 | 11 |
| 0 | 1100 | 12 |
| 0 | 1101 | 13 |
| 0 | 1110 | 14 |
| 0 | 1111 | 15 |
| 1 | 0000 | 16 |
| 1 | 0001 | 17 |
| 1 | 0010 | 18 |
| 1 | 0011 | 19 |



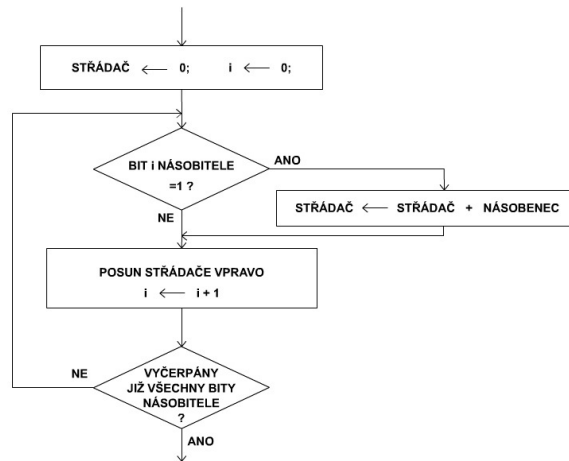
4.16 Násobičky

Násobičky

- Sekvenční násobení (bez znaménka)



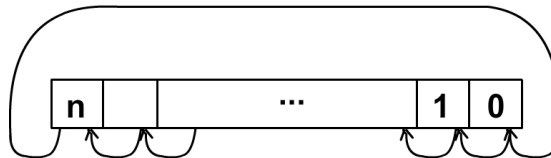
Kombinační násobička



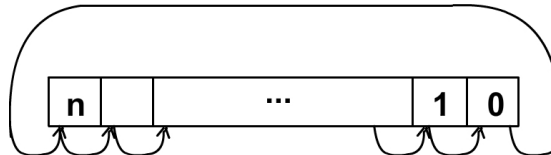
4.17 Rotace bitů, logický a aritmetický posun

Rotace bitů

- Doleva

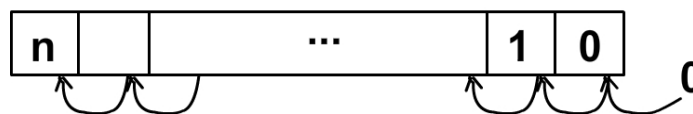


- Doprava

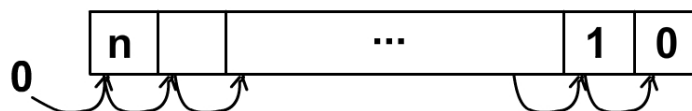


Logický posun (Logical shift)

- Doleva

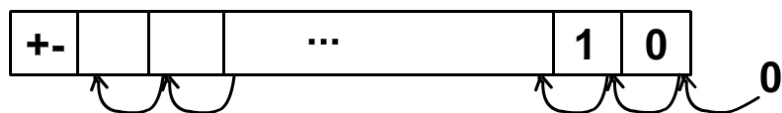


- Doprava



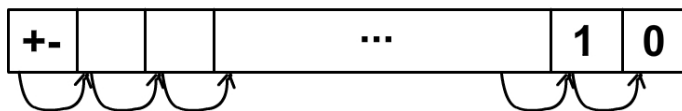
Aritmetický posun (Arithmetic shift)

- Doleva



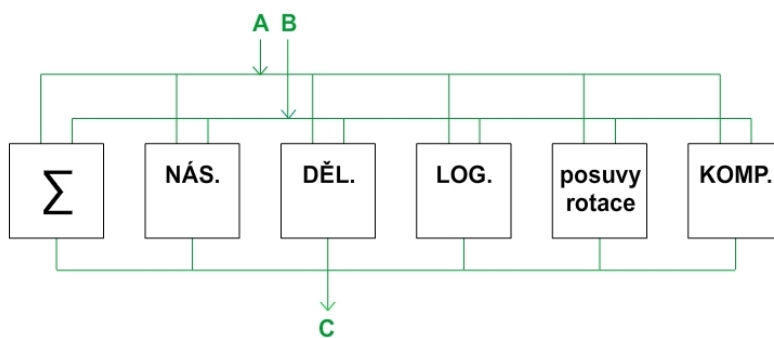
- Znaménkový bit se nemění !
- ~ násobení $\times 2$

- Doprava



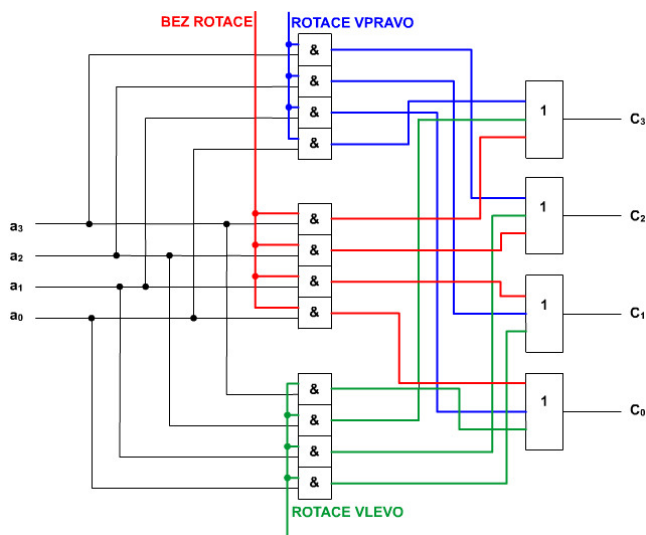
- Znaménkový bit se kopíruje do nižšího řádu.
- ~ dělení $/2$

Blok operační jednotky

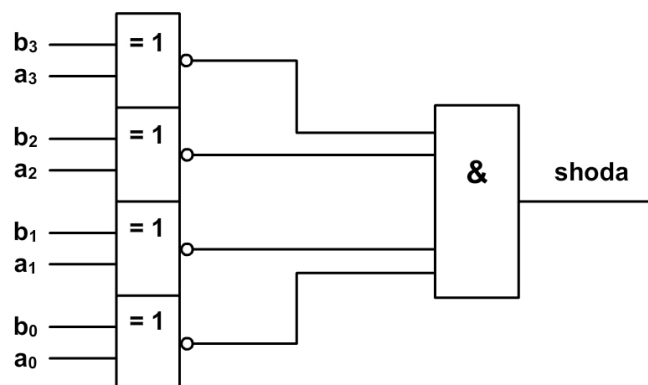


4.18 Obvod pro rotaci bitů vpravo, vlevo a beze změny, komparátor

Obvod pro rotaci vlevo, vpravo a beze změny



Komparátor



5 Paměti

5.1 Parametry

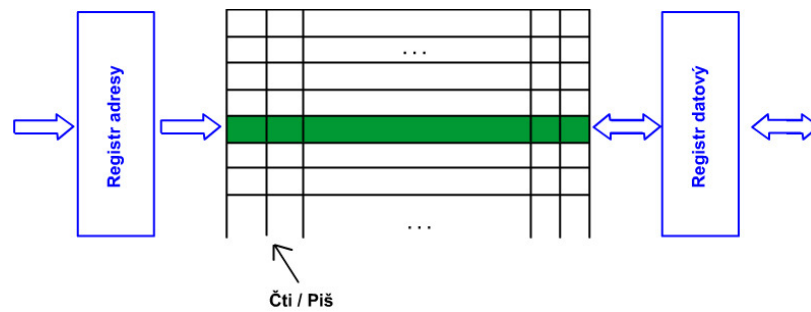
Paměti

- Máme tyto druhy:
 - vnější paměti
 - vnitřní paměti
 - registry
- **Parametry pamětí:**
 - vybavovací doba (tj. čas přístupu k záznamu v paměti) = 10 ns...100 ms
 - rychlost toku dat (tj. počet přenesených bitů za sekundu)
 - kapacita paměti (tj. počet bitů, bajtů, slov)
 - cena za bit
 - přístup
 - * přímý
 - * sekvenční
 - destruktivnost při čtení
 - energetická závislost a nezávislost
 - statika a dynamika
 - spolehlivost – definujeme v rozmezí teplot (např. 1 porucha za 5000 hodin, 1 chyba na 1013 bitů toku)
- **Parametry aplikované na typech pamětí:**
 - vnější paměti
 - vnitřní paměti
 - zápisníková paměť = sada registrů
 - řídicí paměť - pro zaznamenání stavu programů
 - vyrovnávací paměť (též cache) – k vyrovnání rozdílů v toku dat
 - * mezi procesorem a pamětí
 - * mezi procesorem a V/V zařízením

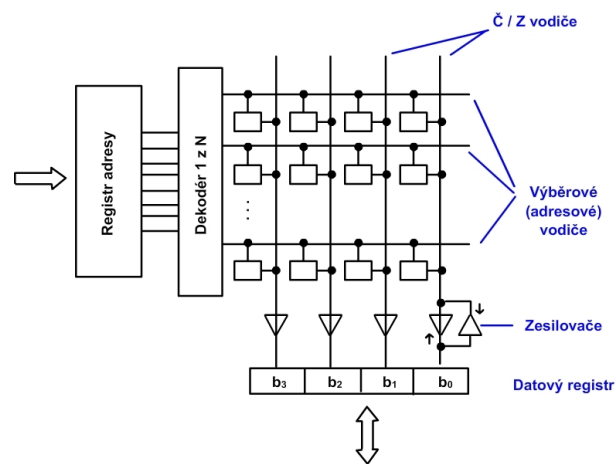
5.2 Vnitřní paměti

Vnitřní paměti

- Klasifikace pamětí podle způsobu čtení a zápisu:

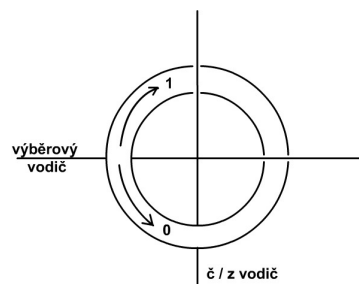


- Fyzická struktura paměti:



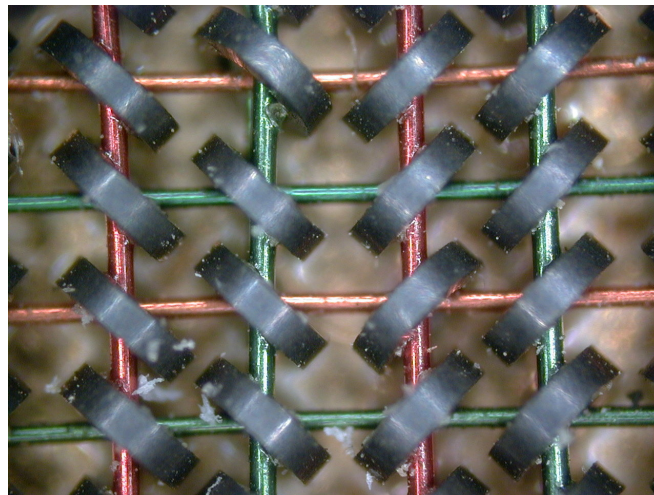
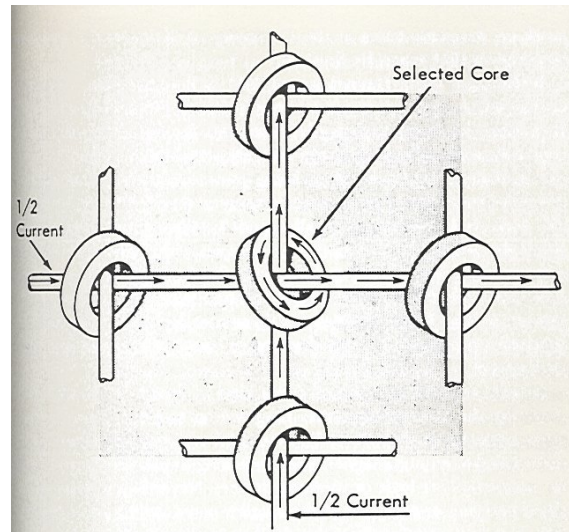
- Paměť pro čtení a zápis:
 - *RWM* (Read-Write Memory)
 - *RAM* (Random Access Memory)
- operační paměť počítačů
- nejrozšířenější – polovodičové paměti
 - Bipolární TTL
 - Unipolární NMOS, CMOS
 - SRAM, DRAM
 - energeticky závislé
 - nedestruktivní

Archaický typ – feritové paměti



ZÁPIS koexistencí proudů výběrového a č/z vodiče

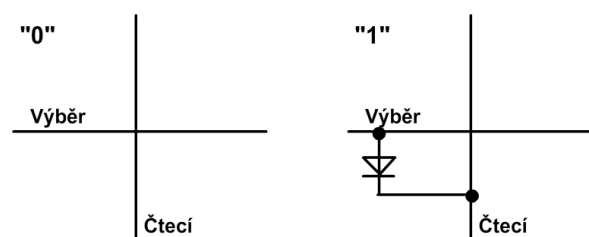
ČTENÍ zápisem „0“ se na č/z vodiči indukují vysoké nebo nízké napětí, původní hodnotu obnovit zpětným zápisem. → *Destructivní čtení*



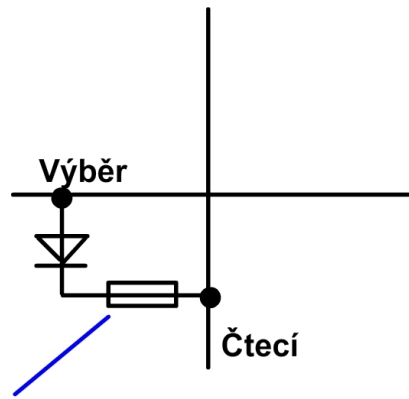
5.3 Permanentní paměti

Permanentní paměti

- paměti určené pouze pro čtení
- základem je *ROM* (read only memory)
- **ROM**



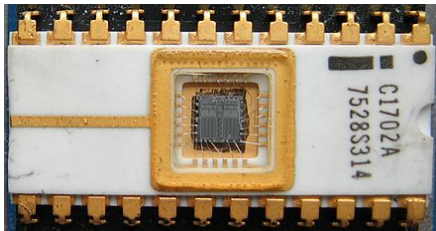
- **PROM** (programmable ROM)



Tavná spojka NiCr

OTP ROM (One Time Programmable)

- **EPROM** (erasable PROM)
 - mazání – působením UV záření (cca 20 minut speciální lampou) se obsah maže tím, že se elektrony rozptýlí.
 - programování – elektricky; elektrony se shromáždí na jedné straně přechodu.

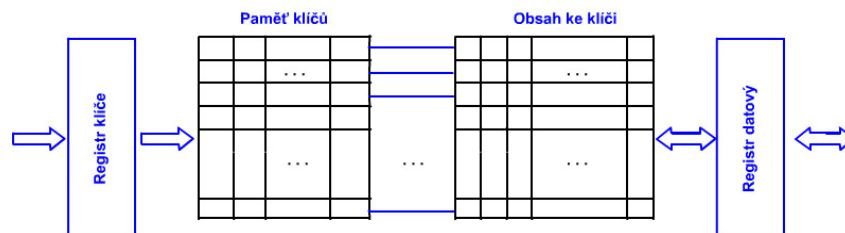


- **EEPROM** (electrically EPROM)
 - mazání elektrickým proudem = *RMM* (read mostly memory)

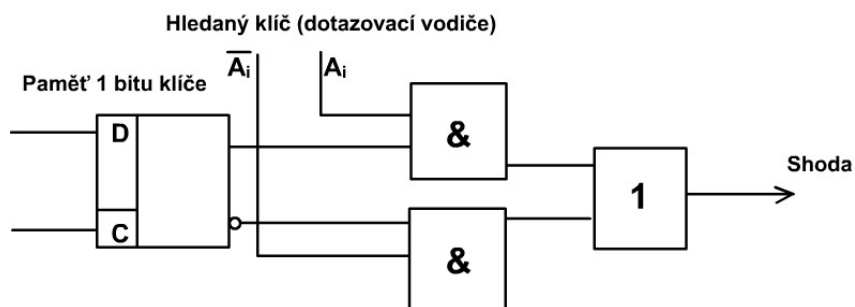
5.4 Asociativní paměť

Asociativní paměť

= CAM (contents addressable memory)



Zapojení 1 bitu klíče:



6 Procesor

6.1 Struktura, fáze, registry, typy instrukcí, instrukce, Little/Big-Endian

Procesor

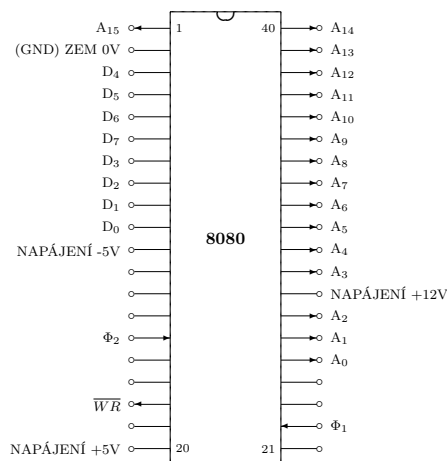
- Procesor je synchronní stroj řízený řadičem.
- Základní frekvence = *takt procesoru*
- *Strojový cyklus* = čas potřebný k zápisu (čtení) slova z paměti (např. 3 takty)
- *Instrukční cyklus* = čas potřebný pro výběr a provedení instrukce
- Příklad formátu instrukce:

| Operační kód (operační znak) | Adresa operandu / operand | Adresa 2. operandu / 2. operand |
|------------------------------|---------------------------|---------------------------------|
| | ne u všech instrukcí | |

- Fáze procesoru:
 - výběr
 - * operačního kódu z paměti
 - * operandu / adresy operandu z paměti
 - provedení instrukce
 - přerušení, ...
- Výběr instrukcí je řízen registrem:
 - čítač instrukcí (adres)
 - PC (Program Counter)
 - IP (Instruction Pointer, alternativní název k PC)
- Po provedení instrukce se zvyšuje o délku instrukce. Plní se např. instrukcí skoku, ...

Počítač

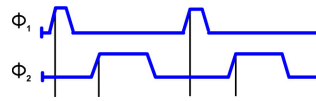
- pracující ve dvojkovém doplňkovém kódu
- registry
 - *A* - stádač - 8bitový (bajt) (Accumulator)
 - *PC* - čítač instrukcí - 16bitový (slovo) (Program Counter)
- paměť
 - 64 KB
 - adresovatelná jednotka = bajt
 - *PC* - čítač instrukcí - 16bitový (slovo)
 - data - 8bitová
- příklad - zapojení mikroprocesoru Intel 8080 (1979) - 8bitový procesor



8bitový procesor

\overline{WR} – Write, řízení zápisu do paměti

Φ_1, Φ_2 – impulsy vnějších hodin



I. část instrukčního souboru

- *LDA adresa* - Load A Direct
 - naplní registr A obsahem bajtu z paměti
 - uložení instrukce v paměti:

| | | |
|---------------|------------------------|-------------------|
| operační znak | 16bitová adresa paměti | |
| 3Ah | nižší bajt adresy | vyšší bajt adresy |

Little Endian

I. část instrukčního souboru

- *STA adresa* - Store A Direct
 - Uloží reg. A do paměti
 - uložení instrukce v paměti:

| | | |
|---------------|------------------------|-------------------|
| operační znak | 16bitová adresa paměti | |
| 32h | nižší bajt adresy | vyšší bajt adresy |

I. část instrukčního souboru

- *JMP adresa* - Jump Unconditional
 - nepodmíněný skok na adresu
 - uložení instrukce v paměti:

| | | |
|---------------|------------------------|-------------------|
| operační znak | 16bitová adresa paměti | |
| 0DAh | nižší bajt adresy | vyšší bajt adresy |

- Příklad: $X := Y$;
LDA 101h

STA 100h

- Proměnné v paměti:
100h X

101h Y

- Instrukce v paměti:
200h LDA 101h

203h STA 100h

206h ...

| | | | | | | | |
|--------|------|------|------|------|------|------|------|
| adresa | 200h | 201h | 202h | 203h | 204h | 205h | 206h |
| obsah | 3Ah | 01 | 01 | 32h | 00 | 01 | ... |

6.2 Instrukce a mikroinstrukce, příznaky

Interní registry (programátorovi neviditelné)

- IR** • instrukční registr (8bitový)
 - je napojen na dekodér instrukcí (řadič)
- DR** • datový registr (8bitový)
 - registr pro čtení/zápis dat z/do paměti
- AR** • adresový registr (16bitový)
 - adresa pro čtení/zápis z/do paměti
- TA = (TA_H, TA_L)** • Temporary Address Register (16bitový), skládá se z: TA_H (TA High - 8 bitů), TA_L (TA Low - 8 bitů)

Fáze procesoru (mikroinstrukce)

- *Fáze instrukce LDA adresa*

| | |
|--|------------------------|
| $200 \rightarrow PC$ | počáteční nastavení PC |
| $PC \rightarrow AR, 0 \rightarrow WR, DR \rightarrow IR$ | výběr operačního znaku |
| $PC + 1 \rightarrow AR, 0 \rightarrow WR, DR \rightarrow TA_L$ | výběr operandu |
| $PC + 2 \rightarrow AR, 0 \rightarrow WR, DR \rightarrow TA_H$ | výběr operandu |
| $TA \rightarrow AR, 0 \rightarrow WR$ | |
| $DR \rightarrow A$ | provedení instrukce |
| $PC + 3 \rightarrow PC$ | aktualizace PC |

Fáze procesoru (mikroinstrukce)

- *Fáze instrukce STA adresa*
 - $PC \rightarrow AR, 0 \rightarrow WR, DR \rightarrow IR$
 - $PC + 1 \rightarrow AR, 0 \rightarrow WR, DR \rightarrow TA_L$
 - $PC + 2 \rightarrow AR, 0 \rightarrow WR, DR \rightarrow TA_H$
 - $A \rightarrow DR$
 - $TA \rightarrow AR, 1 \rightarrow WR$
 - $PC + 3 \rightarrow PC$

Fáze procesoru (mikroinstrukce)

- *Fáze instrukce JMP adresa*
 - $PC \rightarrow AR, 0 \rightarrow WR, DR \rightarrow IR$

 - $PC + 1 \rightarrow AR, 0 \rightarrow WR, DR \rightarrow TA_L$
 - $PC + 2 \rightarrow AR, 0 \rightarrow WR, DR \rightarrow TA_H$
 - $TA \rightarrow PC$

- další registry: B, C, D, E, H, L (8bitové)
- instrukce přesunu mezi registry:
MOV r1,r2 $r_i = \{A,B,C,D,E,H,L\}$ $r1 \leftarrow r2$
kódování - 1 bajt (kombinace registrů je součástí operačního znaku)

- *Fáze MOV r1,r2*
 $PC \rightarrow AR, 0 \rightarrow WR, DR \rightarrow IR$
 $r2 \rightarrow r1$
 $PC + 1 \rightarrow PC$

Aritmetické instrukce

- *Fáze instrukce INR r (Increment Register)*
 $PC \rightarrow AR, 0 \rightarrow WR, DR \rightarrow IR$
 $r + 1 \rightarrow r$
 $PC + 1 \rightarrow PC$
- *Fáze instrukce ADD r (Add Register to A)*
 $PC \rightarrow AR, 0 \rightarrow WR, DR \rightarrow IR$
 $A + r \rightarrow A$
 $PC + 1 \rightarrow PC$
- *Fáze instrukce CMA (Complement A = inverze všech bitů)*
 $PC \rightarrow AR, 0 \rightarrow WR, DR \rightarrow IR$
 $\bar{A} \rightarrow A$
 $PC + 1 \rightarrow PC$

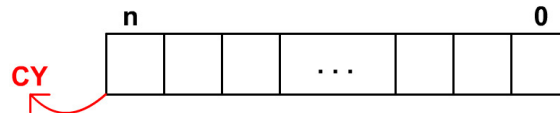
Příznakový registr F procesoru I8080

Z (Zero) • = 1 při nulovém výsledku operace

- = 0 při nenulovém

S (Sign) Kopie znaménkového bitu výsledku operace

CY (Carry) Kopie bitu přenášeného z nejvyššího řádu výsledku operace

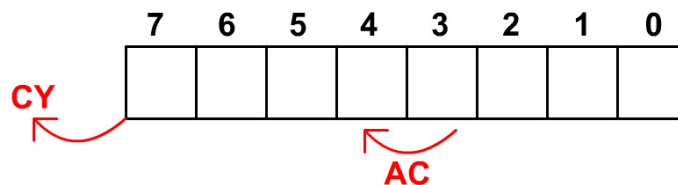


Příznakový registr F procesoru I8080

P (Parity) • = 1 při sudé paritě výsledku

- = 0 při liché paritě výsledku

AC (Auxiliary Carry) přenos mezi bitem 3 a 4 výsledku



Příznaky

Příznaky **nastavují** instrukce: ADD, INR (INR nenastavuje CY)

Příznaky **nemění** instrukce: LDA, STA, JMP, MOV, CMA

- *Fáze instrukce CMP r (Compare Register with A)*
 $PC \rightarrow AR, 0 \rightarrow WR, DR \rightarrow IR$
 $A - r \rightarrow$ nastavení příznaků
 $PC + 1 \rightarrow PC$

6.3 Větvení

Podmíněné skoky

- tj. skoky podle obsahu příznakového registru
- Vzor instrukce: Jpodmínka adresa

- PC \rightarrow AR, 0 \rightarrow WR, DR \rightarrow IR
if podmínka then
PC + 1 \rightarrow AR, 0 \rightarrow WR, DR \rightarrow TA_L
PC + 2 \rightarrow AR, 0 \rightarrow WR, DR \rightarrow TA_H
TA \rightarrow PC
else
PC + 3 \rightarrow PC
fi

Instrukce

| | |
|-----|------|
| JC | CY=1 |
| JNC | CY=0 |
| JZ | Z=1 |
| JNZ | Z=0 |
| JP | S=0 |
| JM | S=1 |

Příklady

- X ... 100h
- Y ... 101h

$X := X + Y;$

LDA 100h

MOV B,A

LDA 101h

ADD B

STA 100h

$X := X - Y;$

LDA 100h

MOV B,A

LDA 101h

CMA

INR A

ADD B

STA 100h

Příklady

if X=Y then ANO else NE;

LDA 100h

MOV B,A

LDA 101h

CMP B

JNZ NE

ANO:

...

JMP VEN

NE:

...

VEN:

if X<Y then ANO else NE;

LDA 101h

MOV B,A

LDA 100h

CMP B ; X-Y

JP NE

ANO:

...

JMP VEN

NE:

...

VEN:

Odečítat X-Y nebo Y-X ?

| | X < Y | | X-Y | Y-X |
|-----|-------|---|-----|-----|
| ANO | 3 | 5 | -2 | 2 |
| | 3 | 4 | -1 | 1 |
| NE | 3 | 3 | 0 | 0 |
| | 3 | 2 | 1 | -1 |
| | 3 | 1 | 2 | -2 |

Příklady

if $X \leq Y$ *then* ANO *else* NE;

```
LDA 100h
```

```
MOV B,A
```

```
LDA 101h
```

```
CMP B      ; Y-X
```

```
JM NE
```

ANO:

```
...
```

```
JMP VEN
```

NE:

```
...
```

VEN:

while $i \geq X$ *do* BLOK;

```
102h i
```

```
...
```

OPAKUJ: LDA 100h

```
MOV B,A
```

```
LDA 102h
```

```
CMP B      ; i-X
```

```
JP BLOK
```

```
JMP KONEC
```

BLOK:

```
...
```

```
JMP OPAKUJ
```

KONEC:

Uložení instrukcí v paměti

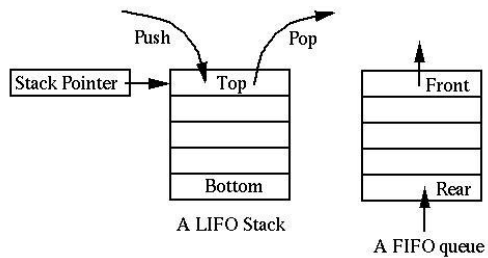
- *for* $i := 1$ *to* X *do* B ;

| | |
|------|-----------------------|
| 0FFh | 1 |
| 100h | X |
| 102h | i |
| ... | |
| ... | |
| 200h | LDA 0FFh |
| 203h | STA 102h ; i:= 1 |
| 206h | MOV B, A ; reg. B:= i |
| 207h | LDA 100h |
| 20Ah | CMP B ; X - i |
| 20Bh | JM 30Ah |
| 20Eh | blok B |
| ... | |
| ... | |
| 300h | LDA 102h |
| 303h | INR A |
| 304h | STA 102h ; i:= i + 1 |
| 307h | JMP 206h |
| 30Ah | |

6.4 Zásobník, volání podprogramu, V/V operace

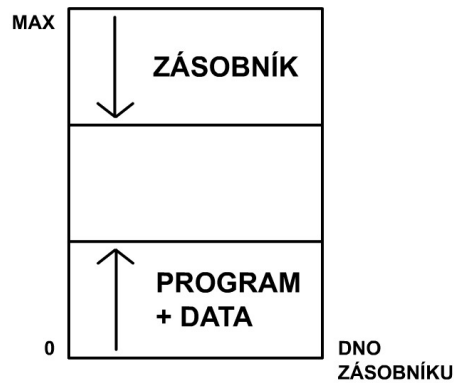
Zásobník

- Struktura Last - in, First - out (LIFO)
- Umístěn kdekoli v operační paměti



Zásobník roste shora dolů

Zásobník roste od vyšších adres k nižším:



Zásobník

- *Registr SP* Stack Pointer (16bitový)
Plnění SP instrukcí *LXI SP, hodnota* Load Immediate
Fáze instrukce:
PC → AR, 0 → WR, DR → IR
PC + 1 → AR, 0 → WR, DR → TA_L
PC + 2 → AR, 0 → WR, DR → TA_H
TA → SP
PC + 3 → PC

Práce se zásobníkem

- *Instrukce:*

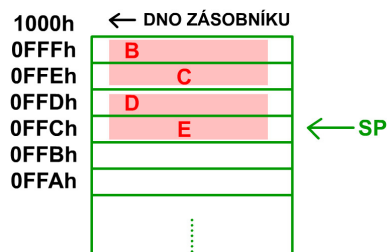
PUSH B | D | H | PSW

POP B | D | H | PSW

- Fáze instrukce **PUSH B|D|H|PSW**
 PC → AR, 0 → WR, DR → IR
 SP - 1 → AR, B | D | H | A → DR, 1 → WR
 SP - 2 → AR, C | E | L | Fl → DR, 1 → WR
 SP - 2 → SP
 PC + 1 → PC
- Fl (Flags, příznaky uspořádané do registru)
- Fáze instrukce **POP B|D|H|PSW**
 PC → AR, 0 → WR, DR → IR
 SP → AR, 0 → WR, DR → C | E | L | Fl
 SP + 1 → AR, 0 → WR, DR → B | D | H | A
 SP + 2 → SP
 PC + 1 → PC

Příklad

- LXI SP,1000h
 PUSH B
 PUSH D



- **Pozor, žádná kontrola podtečení !**

Zásobník a volání podprogramu

- Instrukce:
 - **CALL** adresa
 - **RET**
- Příklad:



- *Fáze instrukce CALL*
 PC → AR, 0 → WR, DR → IR
 PC + 3 → TA
 SP - 1 → AR, TA_H → DR, 1 → WR
 SP - 2 → AR, TA_L → DR, 1 → WR
 SP - 2 → SP
 PC + 1 → AR, 0 → WR, DR → TA_L
 PC + 2 → AR, 0 → WR, DR → TA_H
 TA → PC
- *Fáze instrukce RET*
 PC → AR, 0 → WR, DR → IR
 SP → AR, 0 → WR, DR → TA_L
 SP + 1 → AR, 0 → WR, DR → TA_H
 SP + 2 → SP
 TA → PC

Instrukce nepřímého ...

LDAX 100h

100h 200h

200h 5

A:= 5

STAX ...

JMPX ...

TAX = (TAX_H, TAX_L)

DŮ: mikro-INSTRUKCE

Programování V / V operací

- Instrukce
OUT zapíše obsah A na V/V sběrnici
IN přečte obsah V/V sběrnice do A
START zahájí V /V operaci
FLAG adresa skok na adresu, není-li operace hotova



Příklady

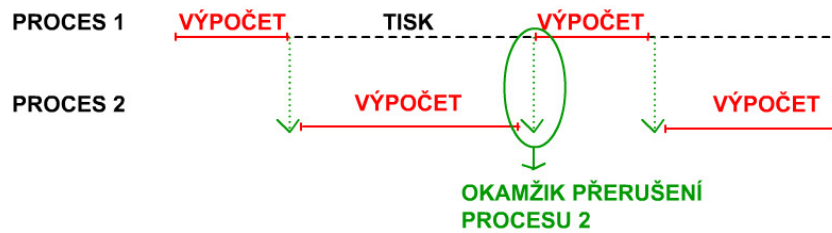
- Přenos A_{100h} do výstupního zařízení

| | |
|-------|------------|
| 1000h | LDA 100h |
| 1003h | OUT |
| 1004h | START |
| 1005h | FLAG 1005h |
| 1008h | |
- Čtení vstupního zařízení a uložení do A_{100h}

| | |
|-------|------------|
| 1000h | START |
| 1001h | FLAG 1001h |
| 1004h | IN |
| 1005h | STA 100h |
| 1008h | |
- Pojem *time-out*

6.5 Přerušování

Multiprogramové zpracování



Přerušovací systém (Interrupt System)

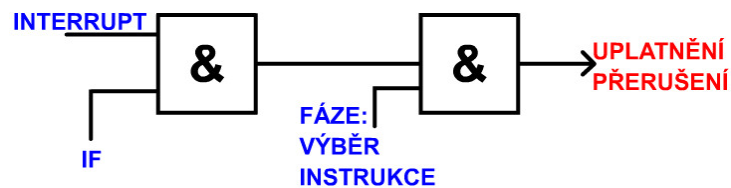
- program (statický) vs. proces (dynamický)
- umožňuje přerušování běžícího procesu a aktivuje rutinu pro obsluhu přerušování

Činnost při přerušování:

1. Přerušování provádění procesu
2. Úklid PC, A,
3. Provedení obslužné rutiny
4. Obnovení PC, A a tím pokračování v provádění procesu

Kdy lze přerušit proces?

- pouze po provedení instrukce (nikoli během ní → instrukce musí dokončit všechny své fáze)
- je-li to povoleno (každý procesor má příznak, kterým se přerušování zakazuje a povoluje).
Např. *IF* (*Interrupt FLAG*).
Instrukce *STI* (přerušování povoleno, tj. $IF=1$)
Instrukce *CLI* (přerušování zakázáno, tj. $IF=0$)
- procesor nelze přerušit bezprostředně po zahájení obsluhy předchozího přerušování
- přerušování se vyvolá signálem *Interrupt* (žádost o přerušování)



- Při přerušování se uplatní tyto fáze:

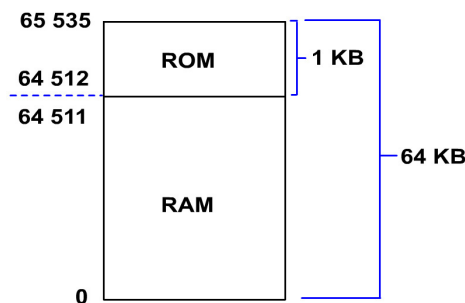
PC → TA
SP - 1 → AR, TA_H → DR, 1 → WR
SP - 2 → AR, TA_L → DR, 1 → WR
SP - 2 → SP
0 → IF
adresa programu pro obsluhu přerušování → PC

Příklad konstrukce programu pro obsluhu přerušení

| | | |
|------|----------|--------------------------------|
| 100h | PUSH PSW | úklid registru A a příznaků |
| 101h | PUSH B | úklid registrů B a C |
| 102h | PUSH D | úklid registrů D a E |
| 103h | PUSH H | úklid registrů H a L |
| 104h | ... | obsluha přerušení |
| ... | ... | ... |
| ... | POP H | obnovení registrů H a L |
| ... | POP D | obnovení registrů D a E |
| ... | POP B | obnovení registrů B a C |
| ... | POP PSW | obnovení registru A a příznaků |
| ... | STI | povolení přerušení |
| ... | RET | návrat do přerušeného procesu |

Signál RESET

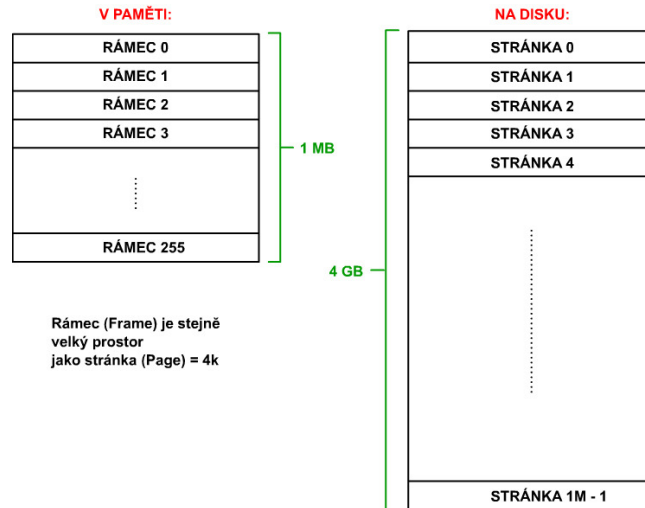
- Nastavení počítače do počátečních podmínek a předání řízení zaváděcímu programu v permanentní paměti
- Příklad: Rozdělení paměti 'našeho' pomyslného počítače:



- Signál Reset se uplatní kdykoli - tj. i uvnitř fází instrukce
- **Fáze RESET:**
 - 0 → IF (zakázání přerušení)
 - 64512 → PC (skok do ROM)
- **Činnosti po zapnutí počítače:**
 1. vyčkání asi 1s (doba náběhu a ustálení zdroje)
 2. generování signálu RESET

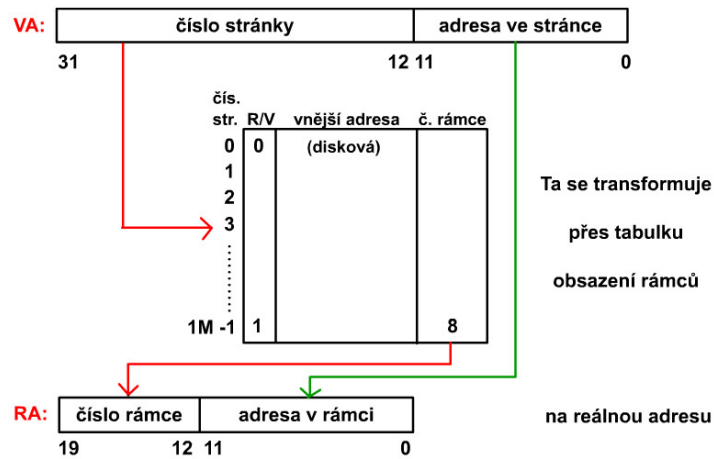
7 Virtuální paměť

Virtuální paměť



Virtuální paměť

- Každý odkaz na paměť obsahuje virtuální adresu:



7.1 Algoritmus LRU

Algoritmus LRU - Least Recently Used

Výběr nejdéle nepoužívané položky:

1. Ve VP vybavit každý blok čítačem, který se při:
 - volání daného bloku nuluje
 - volání jiného bloku inkrementuje o jedničku

V případě potřeby se vyřadí blok s nejvyšší hodnotou

| | 1 | 2 | 3 | 4 | |
|---------------|---|---|---|---|----------------|
| | 1 | 0 | 1 | 1 | |
| bloky: | 2 | 1 | 0 | 2 | volání: |
| | 3 | 2 | 0 | 3 | |
| | 4 | 3 | 1 | 0 | |

- bl. 2
- bl. 3
- bl. 3
- bl. 4

Algoritmus LRU

2. Pomocí neúplné matice s prvky nad hlavní diagonálou

- každý prvek je jednobitová paměť
- při volání bloku i se:
 - jedničkuje i -tý řádek
 - nuluje i -tý sloupec
- nejdéle nepoužité paměťové místo má:
 - v řádku nuly
 - ve sloupci jedničky

Realizace LRU

| | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |

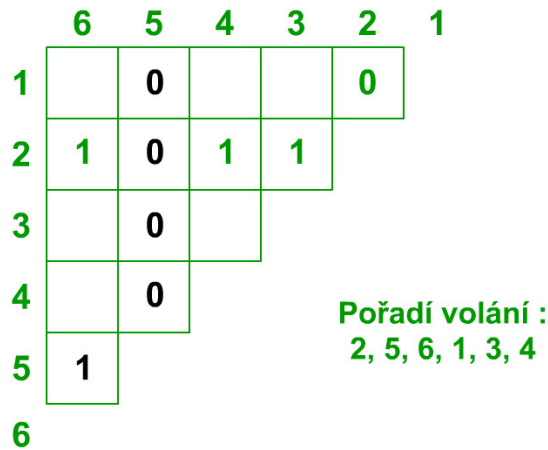
Pořadí volání :
2, 5, 6, 1, 3, 4

Realizace LRU

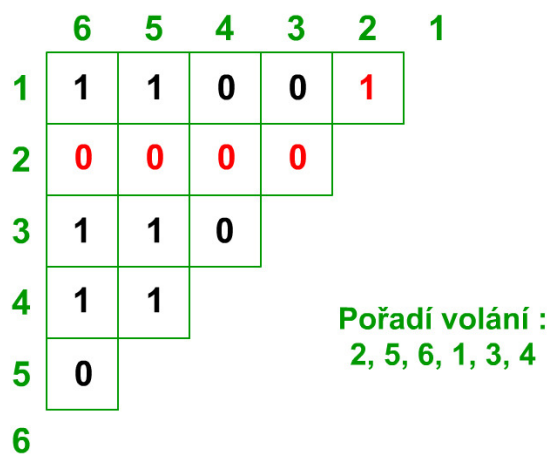
| | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|
| 1 | | | | | 0 | |
| 2 | 1 | 1 | 1 | 1 | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |
| 6 | | | | | | |

Pořadí volání :
2, 5, 6, 1, 3, 4

Realizace LRU

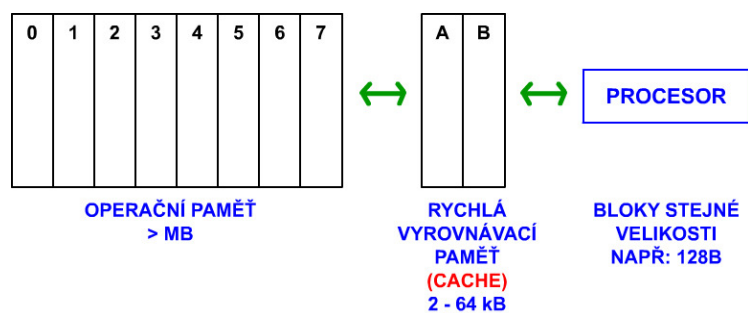


Realizace LRU



8 Vyrovňovací (cache) paměť

Vyrovňovací (cache) paměť, použití LRU



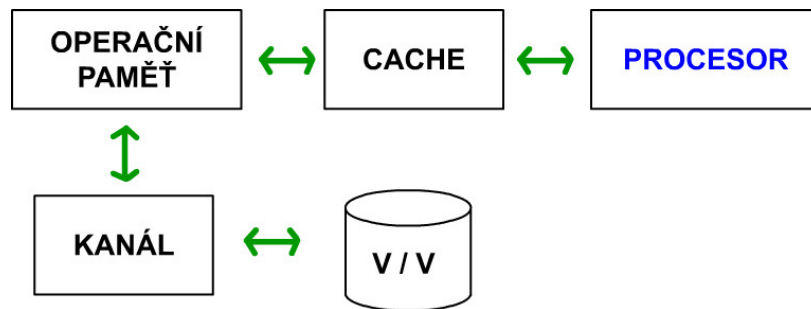
- Není nutné vždy přepisovat blok z VP zpět do OP.
- Který blok při zaplnění VP vyhodit? (použití LRU)

8.1 Použití cache paměti

Jedna paměť, jedna cache a dva různé přístupy

V cache může být nevalidní informace, pokud je do paměti přístup jinou cestou, než přes cache:

- Napojení OP na VP a na kanál:



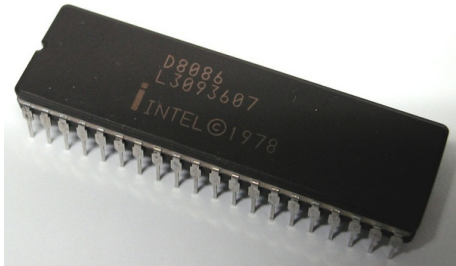
- V multiprocesorových systémech při sdílení jedné paměti více procesory.

9 Architektura procesorů Intel – Procesor 8086

Procesor Intel 8086 a 8088

Procesor 8086

- 16bitový procesor
- 1978 – 1982
- základní procesor řady INTEL x86
- frekvence max. 10 MHz



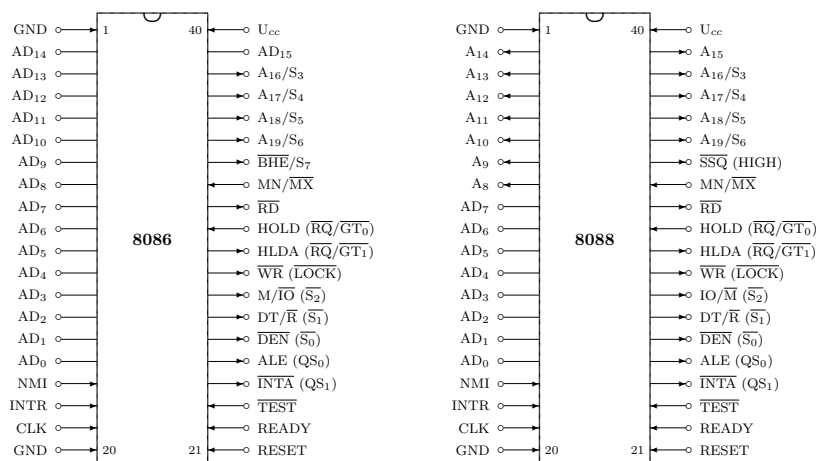
Procesor 8088

- 16bitový procesor do 8bitového prostředí
- 1979 – 1982



9.1 Zapojení procesorů 8086/88

Zapojení procesorů 8086/88



INTR Signál žádosti o maskovatelné přerušení.

TEST Signál testovatelný instrukcí WAIT. Při $\overline{\text{TEST}}=L$ program pokračuje další instrukcí.

NMI Signál nemaskovatelného přerušení.

RESET Signál okamžitě ukončující aktivitu CPU a předávající řízení instrukci na adrese 0FFFF0h.

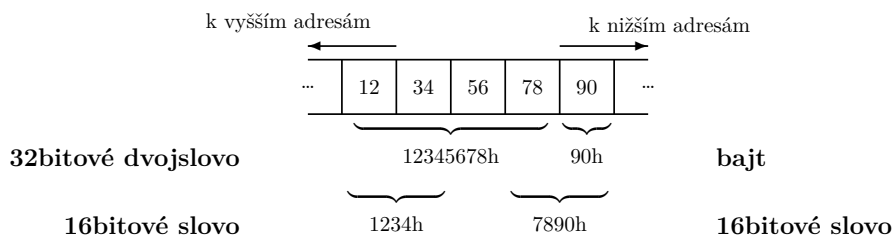
LOCK Uzamčení sběrnice pro procesor, který nastavil $\overline{\text{LOCK}}=L$ instrukčním prefixem LOCK.

M/IO Rozlišuje, zda adresa patří paměti nebo V/V v procesoru 8086.

9.2 Typy dat zpracovávané procesory Intel

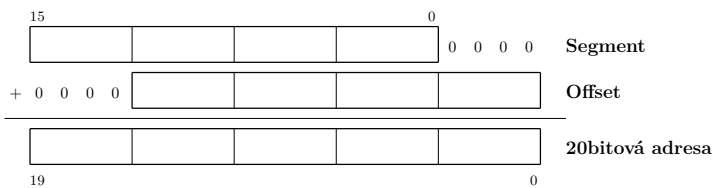
Typy dat zpracovávané procesory Intel

Little-Endian (na nižší adrese nižší řád):



9.3 Adresace paměti procesoru 8086

Adresace paměti procesoru 8086



Adresu zapisujeme ve tvaru

segment : offset.

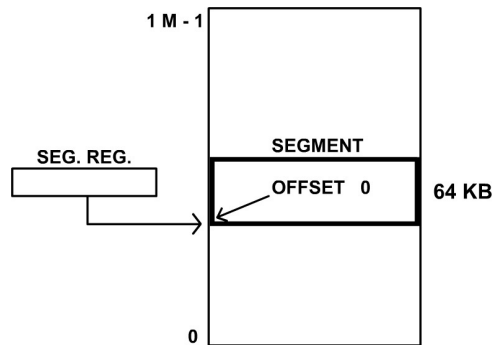
Zápis $01A5:0012_{16}$ představuje tedy dvacetibitovou adresu $01A62_{16}$.

Procesor 8086 pro uložení segmentu poskytuje čtyři 16bitové **segmentové registry**:

- **CS** (Code Segment) je určen pro výpočet adresy instrukce,
- **DS** (Data Segment) slouží pro výpočet adresy dat,
- **SS** (Stack Segment) se použije při přístupu k zásobníku a
- **ES** (Extra Segment) je může obsahovat pomocný datový segment.

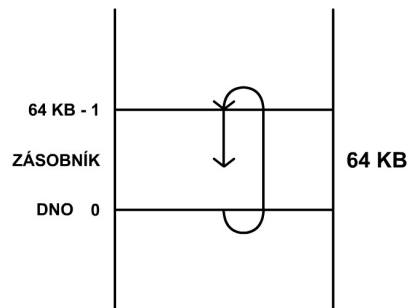
Umístování procesu/segmentu do paměti

Do instrukcí typu 'LDA adresa' se na místo adresy vkládá offset. Segmentovým registrem se určuje, kde je segment umístěn v paměti.



Zásobník v paměti

Instrukcemi PUSH/POP se mění pouze offset, nikoli segment. Zásobník proto „nevyteče“ ze 64KB segmentu.



Zásobník viz dále.

9.4 Registry procesoru

Registry procesoru 8086

| <i>Všeobecné registry</i> | | | | <i>Segmentové registry</i> | |
|---------------------------|----|----|-------------------|----------------------------|---------------------|
| AX | AH | AL | Accumulator | CS | Code Segment |
| BX | BH | BL | Base | DS | Data Segment |
| CX | CH | CL | Counter | ES | Extra Segment |
| DX | DH | DL | Data | SS | Stack Segment |
| SI | | | Source Index | | |
| DI | | | Destination Index | | |
| BP | | | Base Pointer | | |
| SP | | | Stack Pointer | | |
| 15 | 0 | | | | |
| <i>Řídící registry</i> | | | | | |
| | | | | IP | Instruction Pointer |
| | | | | F | Flags |
| | | | | 15 | 0 |

Implicitní přiřazení segmentových registrů

| Při přístupu k | se použije registr | Operace |
|---------------------|---------------------------|--|
| instrukcím | CS (Code Segment) | Výběr operačního kódu nebo přímého operandu. |
| zásobníku | SS (Stack Segment) | Při všech přístupech k zásobníku nebo ve spojitosti s registrem BP. |
| datům | DS (Data Segment) | Při všech přístupech k datům v paměti vyjma zásobníku a přímých operandů. V řetězcových operacích segmentuje zdrojový operand. |
| alternativním datům | ES (Extra Segment) | V řetězcových operacích pro segmentování cílového operandu. |

| Registr s offsetem | Implicitně použitý segmentový registr |
|-----------------------------|---------------------------------------|
| SP | SS |
| BP | SS |
| BX | DS |
| SI | DS |
| DI | DS (ES v řetězcových operacích) |
| BP v kombinaci s SI nebo DI | SS |
| BX v kombinaci s SI nebo DI | DS |

Explicitní přiřazení segmentového registru offsetovému lze zadat např.:

MOV AH,CS:[BX] Nepřímá adresa CS:BX (nikoli DS:BX)
 ADC AH,ES:Adresa Přímá adresa segmentovaná přes ES

9.5 Příznakový registr

Příznakový registr 8086

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|---|----|---|----|---|----|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | OF | DF | IF | TF | SF | ZF | | AF | | PF | | CF |

CF (Carry Flag) obsahuje přenos z nejvyššího bitu, a to jak při práci s 8 nebo 16bitovým operandem.

PF (Parity Flag) se nastaví na jedničku, pokud dolní osmice bitů výsledku právě provedené operace obsahuje sudý počet „1“ (sudá parita výsledku).

AF (Auxiliary Carry Flag) je rozšířením příznaku CF pro přenos přes hranici nejnižšího půlbajtu operandu (vždy z bitu 3 do 4 bez ohledu na šířku operandu). Má význam v BCD aritmetice.

ZF (Zero Flag) je nastaven při nulovém výsledku právě dokončené operace.

SF (Sign Flag) je kopii znaménkového bitu výsledku operace.

Příznakový registr 8086 - pokračování

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|---|----|---|----|---|----|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | OF | DF | IF | TF | SF | ZF | | AF | | PF | | CF |

OF (Overflow Flag) se nastaví na jedničku, pokud při právě dokončené operaci došlo k aritmetickému přeplnění (výsledek spadá mimo rozsah zobrazení).

TF (Trap Flag) uvádí procesor do krokovacího režimu, ve kterém je po provedení první instrukce generováno přerušení (INT 1). Příznak lze nastavit pouze přes zásobník instrukcí IRET.

IF (Interrupt Enable Flag) nulový zabrání uplatnění vnějších maskovatelných přerušení (generovaných signálem INTR).

DF (Direction Flag) řídí směr zpracovávání řetězcových operací.

9.6 Zásobník

Zásobník

- Zásobník procesor implementuje jako strukturu LIFO kdekoli v operační paměti. Všechny odkazy na zásobník jsou segmentovány přes registr SS.

- Příklad: Dno zásobníku je na adrese `SS:0A1A`. Zásobník byl do současného stavu naplněn posloupností instrukcí, které zapsaly hodnoty: `0AA01`, `11AA`, `3C00`.

| | | |
|----------------------|--------------------|--|
| <code>SS:0A1A</code> | Dno zásobníku | |
| <code>0A18</code> | <code>0AA01</code> | |
| <code>0A16</code> | <code>11AA</code> | |
| <code>0A14</code> | <code>3C00</code> | ← Vrchol zásobníku: <code>SS:SP = SS:0A14</code> |
| <code>0A12</code> | | |
| <code>0A10</code> | | |
| | ⋮ | |

- Výběr a zápis do zásobníku řídí registr **SP** (Stack Pointer), který obsahuje adresu právě zapsané položky.

PUSH

Instrukce **PUSH** provede činnosti v následujícím pořadí:

- 1. sníží obsah SP o dvě
- 2. na adresu `SS:SP` uloží obsah 16bitového operandu.

POP

Instrukce **POP** provede tyto akce:

- 1. operand naplní 16bitovým obsahem adresy `SS:SP`
- 2. zvýší obsah SP o dvě

- Procesor 8086 nemá žádný prostředek, kterým by hlídal maximální mez naplnění zásobníku.

9.7 Přerušování

Přerušování v 8086

- **Vnější** (gen. technickými prostředky)
 - nemaskovatelná (signál NMI)
 - maskovatelná (signál INTR)
- **Vnitřní** (gen. programově)
 - instrukcí `INT n`
 - chybou při běhu programu

Vektor adres rutin obsluhujících přerušování

| | | |
|---------------------|------------------------------|-------------------|
| Adresa v paměti | | Číslo přerušování |
| <code>0:0000</code> | <i>offset</i> <i>segment</i> | INT 0 |
| <code>0:0004</code> | <i>offset</i> <i>segment</i> | INT 1 |
| <code>0:0008</code> | <i>offset</i> <i>segment</i> | INT 2 |
| <code>0:000C</code> | <i>offset</i> <i>segment</i> | INT 3 |
| | ⋮ ⋮ | |
| <code>0:03FC</code> | <i>offset</i> <i>segment</i> | INT 0FFh |

Každé přerušování provede akce v tomto pořadí:

- 1. do zásobníku se uloží registr příznaků (F),
- 2. vynulují se příznaky IF a TF,
- 3. do zásobníku se uloží registr CS,
- 4. registr CS se naplní 16bitovým obsahem adresy $n \times 4 + 2$,

- 5. do zásobníku se uloží registr IP,
- 6. registr IP se naplní 16bitovým obsahem adresy $n \times 4$.

Návrat do přerušného procesu

Návrat do přerušného procesu a jeho pokračování zajistí instrukce IRET, která provede činnosti v tomto pořadí:

- 1. ze zásobníku obnoví registr IP,
- 2. ze zásobníku obnoví registr CS,
- 3. ze zásobníku obnoví příznakový registr (F).

Srovnání návratu z přerušného procesu

- „Náš“ procesor:

...

POP PSW

STI

RET

- Procesor 8086:

...

POP AX

IRET

9.8 Rezervovaná přerušení 8086, počáteční nastavení procesoru

Rezervovaná přerušení 8086

| INT n | Význam |
|---------|---|
| 0 | Celočíselné dělení nulou (Divide by Zero) |
| 1 | Krokový režim (Single-Step) |
| 2 | Nemaskovatelná přerušení (NMI) |
| 3 | Ladící bod (Breakpoint Trap) |
| 4 | Přeplnění (Overflow Trap) |

INT 0 při dělení nulou v instrukcích DIV a IDIV. Obsah CS:IP uložený do zásobníku ukazuje **za** (v 80286 a vyšších **na**) instrukci, která přerušení způsobila.

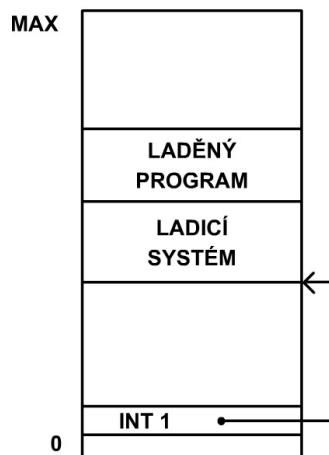
INT 1 po provedení instrukce, je-li TF=1.

INT 2 po přijetí signálu NMI (v 8086 pouze chyba parity v paměti), který nelze zakázat nulovou hodnotou příznaku IF.

INT 3 se používá společně s přerušením INT 1 v ladících systémech. Přerušení 03h se vygeneruje po dekodování speciální jednobajtové instrukce INT 3 (s operačním kódem 0CCh). Přerušení uloží do zásobníku obsah CS:IP ukazující na bajt bezprostředně za touto instrukcí.

INT 4 provede instrukce INTO (Interrupt on Overflow), je-li v okamžiku jejího dekodování nastaven příznak OF=1. CS:IP ukazuje na bajt za touto instrukcí.

Krokovací režim (TF=1)



Počáteční spuštění krokovacího režimu

MOV AX, 0x0100 ;nastavení TF=1 PUSH AX MOV AX, segmentová část adresy 1. instrukce PUSH AX MOV AX, 0 ;offsetová část adresy 1. instrukce PUSH AX IRET

Pozn. IRET provede:[1mm] POP IP POP CS POP F

Počáteční nastavení procesoru

Procesor je inicializován aktivní úrovní signálu RESET.

| Registr | Obsah |
|--------------------|--------|
| Příznakový registr | 0 |
| IP | 0 |
| DS, ES, SS | 0 |
| CS | 0FFFFh |

Tzn., že první instrukce, kterou bude procesor po inicializaci signálem RESET zpracovávat, je umístěna na adrese 0FFFF:0000h, tj. 0FFFF0h (15 bajtů od konce).

9.9 Adresovací techniky

Adresovací techniky

Instrukce: **Op. kód** **Operand(y)**

Registr

Přímý operand

Přímá adresa

(totožné s MOV AH,[PROM])

Neřímá adresa

Bázovaná adresa

Indezovaná adresa

(SI, DI)

Báze + Index

(BP, BX + SI, DI)

Přímá + Báze + Index

nebo

nebo

nebo

PROM

nebo

(SI, DI, SP, BP, BX)

(BP, BX)

nebo

nebo

nebo

nebo

nebo

nebo

nebo

Příklad:

MOV AH,BL

MOV AH,50

DB ?

MOV AH,PROM

MOV AH,DS:[101]

MOV AH,[BX]

MOV AH,[BP+PROM]

MOV AH,[PROM+SI]

MOV AH, PROM[SI]

MOV AH,[BX][DI]

MOV AH,[BP+DI]

MOV AH,PROM[BX][DI]

MOV AH,[PROM+BX+DI]

MOV AH,[PROM+BX+DI+1]

SEGMENT: PŘÍMÁ ADRESA + BÁZE + INDEX

9.10 Instrukce MOV

Instrukce MOV

Instrukce MOV příznaky nemění!!

| | | |
|------------------------------|----------------------------|----------|
| MOV r/m8,r8 | MOV AL,BL | AL:=BL |
| | MOV Bajt,CH | Bajt:=CH |
| MOV r/m16,r16 | MOV BX,CX | |
| MOV r8,r/m8 | | |
| MOV r16,r/m16 | | |
| MOV r/m16,segmentový registr | MOV AX,CS | |
| MOV segmentový registr,r/m16 | MOV DS,AX | |
| | Nelze MOV CS,.. !!! | |
| MOV r/m8,imm8 | MOV Bajt,10 | |
| MOV r/m16,imm16 | | |

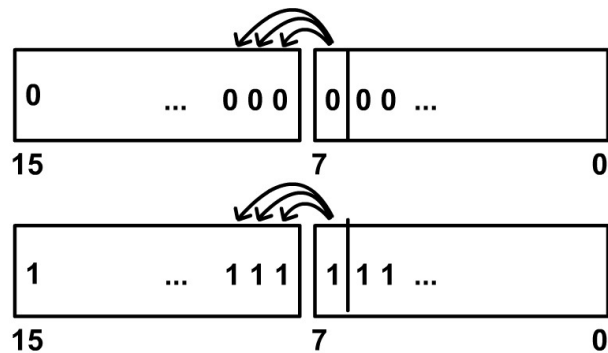
Po instrukci *MOV SS,..* je po dobu trvání následující instrukce zakázáno přerušení !

9.11 Aritmetické instrukce

Rozšíření s respektováním znaménka

Číslo z 8bitového registru rozšířit do 16bitového:

Znaménkový bit objektu, který se má rozšířit, se zkopíruje do všech bitů objektu, o který se rozšiřuje.



Aritmetické instrukce (celočíslné)

- ADD*

| | | | | | |
|---|---|---|---|---|---|
| * | * | * | * | * | * |
| O | S | Z | A | P | C |

r/m8,imm8 ADD AL,80 AL:=AL+80

ADD Bajt,-10

r/m16,imm16 ADD CX,10000

r/m16,imm8 ADD CX,10 ← rozšíření s respektováním znaménka

r/m8,r8 ADD CH,CL

r/m16,r16 ADD AX,BX

r8,r/m8

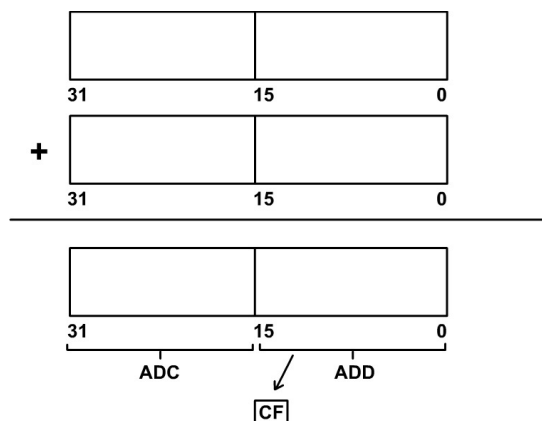
r16,r/m16

- ADC* – ADD WITH CARRY

... ADC AL,CL AL:=AL+CL+CF

Použití instrukce ADC

Instrukce ADC se používá na sčítání objektů, jejichž šířka je větší než běžně zpracovávaná šířka objektů.



Aritmetické instrukce (celočíslné)

- *SUB* – SUBTRACTION
... SUB AL,CL AL:=AL-CL
- *SBB* – SUBTRACTION WITH BORROW
... SBB AL,CL AL:=AL-CL-CF
- *CMP* – COMPARE
... CMP AL,CL F:=AL-CL
- *INC* – INCREMENT

| | | | | | |
|---|---|---|---|---|--|
| * | * | * | * | * | |
|---|---|---|---|---|--|

| | | | | | |
|---|---|---|---|---|---|
| O | S | Z | A | P | C |
|---|---|---|---|---|---|

 INC r/m8 INC Bajt Bajt:=Bajt+1
 INC r/m16 INC DX

Aritmetické instrukce (celočíslné)

- *DEC* – DECREMENT
... DEC Bajt Bajt:=Bajt-1
- *NEG* – DVOJKOVÝ DOPLNĚK

| | | | | | |
|---|---|---|---|---|---|
| * | * | * | * | * | * |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| O | S | Z | A | P | C |
|---|---|---|---|---|---|

 ... NEG Bajt Bajt:=-Bajt
- *CBW* – CONVERT BYTE TO WORD

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
|--|--|--|--|--|--|

| | | | | | |
|---|---|---|---|---|---|
| O | S | Z | A | P | C |
|---|---|---|---|---|---|

 ... CBW AX:=AL se zachováním znaménka

Aritmetické instrukce (celočíslné)

- *CWD* – CONVERT WORD TO DOUBLEWORD

| | |
|------------------------|-------------------|
| DX&AX := AX | |
| ↙ | ↘ |
| vyšší | nižší bity |

 ... CWD

Aritmetické instrukce (celočíslné)

- *IMUL – SIGNED MULTIPLICATION* *respektuje znaménka!*

| | | | | | |
|---|---|---|---|---|---|
| * | ? | ? | ? | ? | * |
| O | S | Z | A | P | C |

| | | |
|------------|------------|-------------------|
| IMUL r/m8 | IMUL BL | AX:=AL * BL |
| | IMUL Bajt | AX:=AL * Bajt |
| IMUL r/m16 | IMUL CX | DX&AX:=AX * CX |
| | IMUL Slovo | DX&AX:=AX * Slovo |

- *MUL – UNSIGNED MULTIPLICATION* – *Neuvažuje znaménkový bit, jinak stejné jako IMUL*

Rozdíl mezi IMUL a MUL je jen ve zpracování horní poloviny výsledku (IMUL znaménkově rozšiřuje) a nastavování příznaků OF a CF.

Aritmetické instrukce (celočíselné)

- *IDIV – SIGNED DIVIDE*

| | | | | | |
|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? |
| O | S | Z | A | P | C |

| | | |
|------------|------------|---|
| IDIV r/m8 | IDIV BL | AL:=AX ÷ BL AH:=AX modulo BL |
| | IDIV Bajt | AL:=AX ÷ Bajt AH:=AX modulo Bajt |
| IDIV r/m16 | IDIV CX | AX:=DX&AX ÷ CX DX:=DX&AX modulo CX |
| | IDIV Slovo | AX:=DX&AX ÷ Slovo DX:=DX&AX modulo Slovo |

Je-li podíl větší než maximální rozsah zobrazení → INT 0. Zbytek má stejné znaménko jako dělenec.

- *DIV – UNSIGNED DIVIDE* – *Neznaménkové*

9.12 Logické instrukce

Logické instrukce

- *AND – logický součin po bitech*

| | | | | | |
|---|---|---|---|---|---|
| 0 | * | * | ? | * | 0 |
| O | S | Z | A | P | C |

| | | |
|---------------------|-----------------|----------------------|
| kombinace parametrů | AND AL,7 | AL:=AL ∧ 7 |
| viz „ADD“ | AND Slovo,1FFFh | Slovo:=Slovo ∧ 1FFFh |

- *OR – logický součet*
OR AL,7 AL:=AL ∨ 7

- *XOR – nonekvivalence*
XOR AL,7 AL:=AL ⊕ 7

Logické instrukce

- *NOT – inverze bitů (jedničkový doplněk)*
- příznaky nemění

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| O | S | Z | A | P | C |

| | | |
|-----------|-----------|----------------------------|
| NOT r/m 8 | NOT AH | AH:= \overline{AH} |
| | NOT Bajt | Bajt:= \overline{Bajt} |
| NOT r/m16 | NOT SI | SI:= \overline{SI} |
| | NOT Slovo | Slovo:= \overline{Slovo} |

Logické instrukce

- *TEST* – LOGICAL COMPARE

| | | | | | |
|---|---|---|---|---|---|
| 0 | * | * | ? | * | 0 |
| O | S | Z | A | P | C |

TEST r/m8,imm8 TEST AL,7 F:=AL \wedge 7
 TEST Bajt,15 F:=Bajt \wedge 15

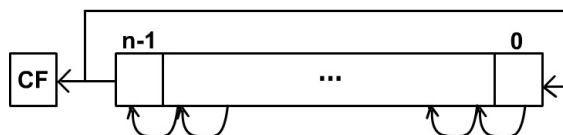
TEST r/m16,imm16
 TEST r/m8,r8
 TEST r/m16,r16

AND, OR, XOR r/m16,imm8 znaménkové rozšíření

9.13 Rotace, posuvy

Rotace

- *ROL* – ROTATE LEFT



ROL r/m8,1
 ROL r/m8,CL
 ROL r/m16,1
 ROL r/m16,CL

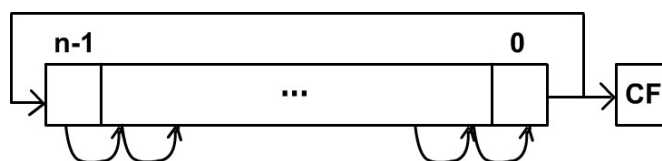
8086 : CL neomezeno

286,.. : CL \wedge 1Fh

- OF je definováno pouze při rotaci o 1 bit:
- ROL: OF:=CF \oplus bitn-1
- tj. OF se nastaví, pokud se hodnota CF nerovná novému nejvyššímu bitu.

Rotace

- *ROR* – ROTATE RIGHT



ROR: OF:=bitn-1 \oplus bitn-2

Rotace

- *RCL* – ROTATE LEFT THROUGH CARRY



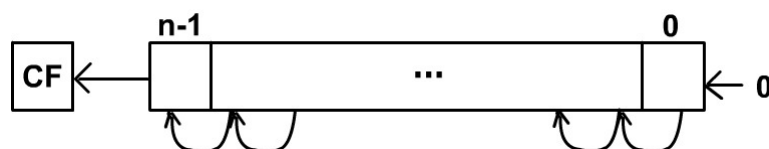
Rotace

- *RCR* – ROTATE RIGHT THROUGH CARRY



Posuvy

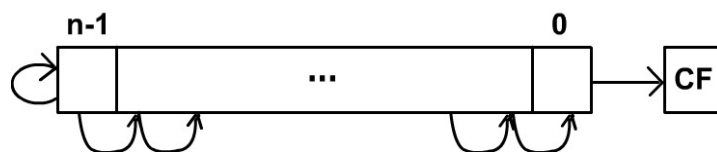
- *SAL* – SHIFT ARITHMETIC LEFT
- *SHL* – SHIFT LOGICAL LEFT
- obě provedou tutéž akci
- varianty viz instrukce „ROL“



- znaménko aritmetického násobení 2^n
- $OF := CF \oplus bit_{n-1}$

Posuvy

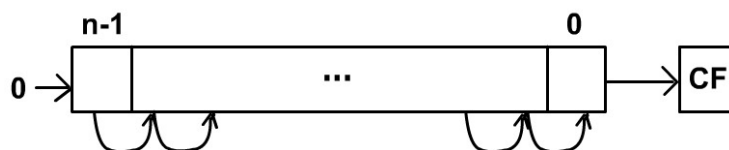
- *SAR* – SHIFT ARITHMETIC RIGHT



- $OF := 0$

Posuvy

- *SHR* – SHIFT LOGICAL RIGHT



- $OF := \text{původní } bit_n - 1$

9.14 Větvení programu

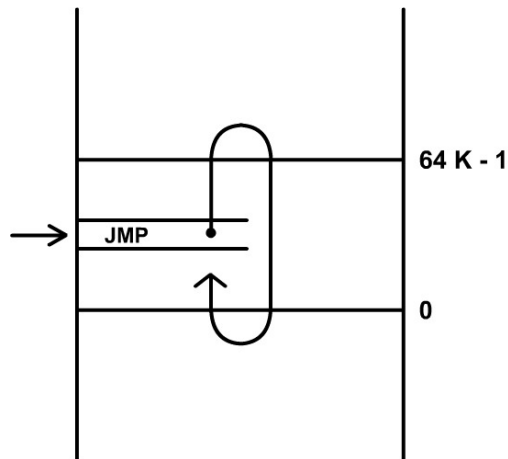
Větvení programu

JMP – jump, nepodmíněný skok

- přímý skok (cílová adresa je v instrukci)
- mění CS
- * vzdálený skok (*far jump*)

- * plní CS:IP
- nemění CS - IP := IP + vzdálenost
- * vzdálenost $\in \langle 0; 65535 \rangle$
sčítá se neznaménkově!!
= *blízký skok (near jump)*
- * vzdálenost $\in \langle -128; +127 \rangle$
sčítá se znaménkově!!
= *krátký skok (short jump)*

Blízký skok



Větvení programu

JMP – jump, nepodmíněný skok

- nepřímý skok : v instrukci je odkaz na:
 - Registr SI, DI, SP, BP nebo BX (obsahuje offset cílové adresy)
 - Paměť
 - * segment : offset
 - * offset

Větvení programu

| | | |
|--------------|--------------------------------|--|
| JMP rel8 | JMP SHORT návěští | krátký skok IP:=IP+vzdálenost návěští |
| JMP rel16 | JMP návěští | blízký skok IP:=IP+vzdálenost návěští |
| JMP ptr16:16 | JMP FAR _ PTR návěští | vzdálený skok CS:IP:= segment:offset návěští |
| JMP r/m16 | JMP [BX] | nepřímý blízký skok IP:=BX |
| JMP m16:16 | JMP [slovo] JMP [dvojslovo] | IP:=slovo CS:IP:=dvojslovo nepřímý skok vzdálený |

9.15 Podmíněné skoky

Podmíněné skoky

JUMPS CONDITIONAL

- Pouze krátké skoky: vzdálenost $\langle -128; +127 \rangle$

- Neuvádí se „short“

| JUMP SHORT IF.. | TESTOVANÁ PODMÍNKA | VÝSLEDEK POSLEDNÍ OPERACE |
|------------------------|-----------------------|------------------------------|
| <i>JE EQUAL</i> | ZF=1 | roven |
| <i>JZ ZERO</i> | ZF=1 | nulový |
| <i>JNE NOT EQUAL</i> | ZF=0 | různý |
| <i>JNZ NOT ZERO</i> | ZF=0 | nenulový |
| <i>JP PARITY</i> | PF=1 | sudá parita |
| <i>JPE PARITY EVEN</i> | PF=1 | sudá parita |
| <i>JNP NOT PARITY</i> | PF=0 | lichá parita |
| <i>JPO PARITY ODD</i> | PF=0 | lichá parita |

Podmíněné skoky

| JUMP SHORT IF.. | TESTOVANÁ PODMÍNKA | VÝSLEDEK POSLEDNÍ OPERACE |
|-------------------------|-----------------------|------------------------------|
| <i>JS SIGNUM</i> | SF=1 | záporný |
| <i>JNS NOT SIGNUM</i> | SF=0 | kladný nebo nulový |
| <i>JC CARRY</i> | CF=1 | nastal přenos |
| <i>JNC NOT CARRY</i> | CF=0 | nenastal přenos |
| <i>JO OVERFLOW</i> | OF=1 | nastalo přetečení |
| <i>JNO NOT OVERFLOW</i> | OF=0 | nenastalo přetečení |

Písmenné zkratky vyjádření podmínky

| | | |
|------------------|----|---|
| equal | EQ | = |
| not equal | NE | ≠ |
| less than | LT | < |
| less or equal | LE | ≤ |
| greater than | GT | > |
| greater or equal | GE | ≥ |

Programovací jazyk FORTRAN (vznik 1954-57, IBM)

Podmíněné skoky

| JUMP SHORT IF.. | TESTOVANÁ PODMÍNKA | VÝSLEDEK POSLEDNÍ OPERACE |
|---------------------------------|-----------------------|------------------------------|
| <i>JB BELOW</i> | CF=1 | nz. menší |
| <i>JNAE NOT ABOVE NOR EQUAL</i> | CF=1 | nz. menší |

Př: + 2:

| | | |
|---|---|---|
| 0 | 1 | 0 |
|---|---|---|

 } nz.
+ 5:

| | | |
|---|---|---|
| 1 | 0 | 1 |
|---|---|---|

CMP 2,5:

$$\begin{array}{r} 010 \\ - 101 \\ \hline 1101 \end{array}$$

CF = 1, SF = 0, OF = 0, ZF = 0
NZ: 2 < 5 ✓

Podmíněné skoky

| JUMP SHORT IF.. | TESTOVANÁ PODMÍNKA | VÝSLEDEK POSLEDNÍ OPERACE |
|-----------------------------------|-----------------------|------------------------------|
| <i>JL LESS</i> | SF ≠ OF | z. menší |
| <i>JNGE NOT GREATER NOR EQUAL</i> | SF ≠ OF | z. menší |

Př: + 2:

| | | |
|---|---|---|
| 0 | 1 | 0 |
|---|---|---|

 } z.
- 3:

| | | |
|---|---|---|
| 1 | 0 | 1 |
|---|---|---|

CMP 2,-3: CF = 1, SF = 1, OF = 1, ZF = 0
Z: 2 < -3

$$\begin{array}{r} 101 \\ - 010 \\ \hline 011 \end{array}$$

CF = 0, SF = 1, OF = 1, ZF = 0
NZ: -3 < 2 ✓

Podmíněné skoky

| JUMP SHORT IF.. | | TESTOVANÁ PODMÍNKA | VÝSLEDEK POSLEDNÍ OPERACE |
|-----------------|----------------------------|----------------------------|------------------------------|
| <i>JA</i> | <i>ABOVE</i> | $(CF=0) \wedge (ZF=0)$ | nz. větší |
| <i>JNBE</i> | <i>NOT BELOW NOR EQUAL</i> | $(CF=0) \wedge (ZF=0)$ | nz. větší |
| <i>JG</i> | <i>GREATER</i> | $(SF=OF) \wedge (ZF=0)$ | z. větší |
| <i>JNLE</i> | <i>NOT LESS NOR EQUAL</i> | $(SF=OF) \wedge (ZF=0)$ | z. větší |
| <i>JBE</i> | <i>BELOW OR EQUAL</i> | $(CF=1) \vee (ZF=1)$ | nz. menší nebo rovno |
| <i>JNA</i> | <i>NOT ABOVE</i> | $(CF=1) \vee (ZF=1)$ | nz. menší nebo rovno |
| <i>JLE</i> | <i>LESS OR EQUAL</i> | $(SF \neq OF) \vee (ZF=1)$ | z. menší nebo rovno |
| <i>JNG</i> | <i>NOT GREATER</i> | $(SF \neq OF) \vee (ZF=1)$ | z. menší nebo rovno |
| <i>JAE</i> | <i>ABOVE OR EQUAL</i> | $CF=0$ | nz. větší nebo rovno |
| <i>JNB</i> | <i>NOT BELOW</i> | $CF=0$ | nz. větší nebo rovno |

Podmíněné skoky

| JUMP SHORT IF.. | | TESTOVANÁ PODMÍNKA | VÝSLEDEK POSLEDNÍ OPERACE |
|-----------------|---------------------------|--|------------------------------|
| <i>JGE</i> | <i>GREATER OR EQUAL</i> | $SF=OF$ | z. větší nebo rovno |
| <i>JNL</i> | <i>NOT LESS</i> | $SF=OF$ | z. větší nebo rovno |
| <i>JCXZ</i> | <i>JUMP SHORT IF CX=0</i> | používá se pro řízení cyklů | |
| Jpodm rel8 | JZ návěští | krátký skok na návěští je-li ZF=1, jinak se pokračuje následující instrukcí | |
| JCXZ rel8 | JCXZ návěští | krátký skok na návěští je-li CX=0 | |

9.16 Zásobník

Zásobník

PUSH Uložení 16bitového objektu do zásobníku:

1. $SP := SP-2$
2. $[SS:SP] := \text{operand_16bitový}$

| | |
|--------------|------------|
| PUSH m16 | PUSH slovo |
| PUSH r16 | PUSH AX |
| PUSH segment | PUSH CS |

Zásobník

POP Výběr 16bitového objektu ze zásobníku:

1. $\text{operand_16bitový} := [SS:SP]$
2. $SP := SP+2$

| | |
|---------------|--|
| POP m16 | POP slovo |
| POP r16 | POP BX |
| POP segment | <i>NELZE: POP CS !!!</i> |
| <i>POP SS</i> | zakazuje přerušení na dobu provedení této a následující instrukce |

9.17 Volání a návrat z podprogramu

Volání a návrat z podprogramu

- *CALL*

| | | |
|---------------|----------------------|---------------------------|
| CALL rel16 | CALL návěští | IP:=IP+vzdálenost návěští |
| CALL ptr16:16 | CALL FAR PTR návěští | CS:IP:=ptr 16:16 |
| CALL r/m16 | CALL [BX] | IP:=BX |
| CALL m16:16 | CALL [dvojslovo] | CS:IP := dvojslovo |

- *CALL*

1. PUSH CS - pouze „FAR“ varianta
2. PUSH IP+délka_ instrukce
3. (CS):IP := operand

Volání a návrat z podprogramu

- *RET* – RETURN

| | | |
|------------|--------------|-----------------|
| RET | POP IP | blízký návrat |
| RETF | POP IP | vzdálený návrat |
| | POP CS | vzdálený návrat |
| RET imm16 | POP IP | |
| | SP:=SP+imm16 | |
| RETF imm16 | POP IP | |
| | POP CS | |
| | SP:=SP+imm16 | |

- Příklad:
RET 2 návrat z podprogramu
 s odstraněním 1 slova

PUSH parametr
CALL podprogram

9.18 Příznakový registr

Příznakový registr

| | | |
|--------------|--------------------|--|
| <i>PUSHF</i> | PUSH FLAG REGISTER | nemění příznaky |
| | PUSHF | uloží 16bitový registr F |
| <i>POPF</i> | POP FLAG REGISTER | mění příznaky |
| | POP F | vybere 16bitový objekt a uloží jej do F |
| <i>STI</i> | IF:=1 | povolení přerušení |
| <i>STD</i> | DF:=1 | řetězce odzadu |
| <i>STC</i> | CF:=1 | |
| <i>CLI</i> | IF:=0 | zákaz přerušení |
| <i>CLD</i> | DF:=0 | řetězce odpředu |
| <i>CLC</i> | CF:=0 | |

Poznámka: STI povolí přerušení až po provedení následující instrukce

9.19 Přerušení

Přerušení

INT INTERRUPT

INT imm8 délka 2 bajty

1. PUSHF
2. IF:=0, TF:=0
3. PUSH CS
4. CS:=[imm8 x 4 + 2]
5. PUSH IP + délka instrukce (*obsah zásobníku ukazuje za „INT imm8“!*)
6. IP:=[imm8 x 4]

Přerušení

INT 3 délka 1 bajt, operační kód: 0CCh

INTO INTERRUPT IF OVERFLOW

délka 1 bajt, operační kód 0CEh provede INT 4, je-li OF=1

IRET INTERRUPT RETURN

1. POP IP
2. POP CS
3. POP F

9.20 Cykly

Cykly

- *LOOP* – UNCONDITIONAL LOOP

- nemění příznaky
- registr **CX** ... čítač průchodů

| | | |
|-----------|-------------------------|--|
| Př: | MOV CX, počet_ průchodů | inicializace řídicí proměnné |
| | OPAKUJ: | |
| | . | tělo cyklu |
| | . | |
| | LOOP OPAKUJ | CX:=CX-1 pokud CX ≠ 0 ... SHORT skok na OPAKUJ |
| | zde, je-li CX=0 | |
| LOOP rel8 | LOOP OPAKUJ | 1) CX:=CX-1 2) je-li CX ≠ 0 ... SHORT skok |

Cykly

CONDITIONAL LOOP

- *LOOPE*
LOOPE rel8 1) CX:=CX-1
2) je-li (CX ≠ 0) ∧ (ZF=1) ... SHORT skok na rel8
- *LOOPZ*
stejně jako LOOPE
- *LOOPNE*
LOOPNE rel8 1) CX:=CX-1
2) je-li (CX ≠ 0) ∧ (ZF=0) ... SHORT skok na rel8
- *LOOPNZ*
stejně jako LOOPNE

9.21 Ovládání V/V

Ovládání V/V zařízení

- Na adresovou sběrnici „číslo“ V/V zařízení ≡ V/V brána ≡ port v intervalu 0 - 65535
- Na datovou sběrnici data

V/V brány jsou 8bitové – lze pracovat i s dvojicí bran na po sobě jdoucích adresách.

Ovládání V/V

- *IN* – INPUT FROM PORT

Přenos bajtu nebo slova ze V/V brány do registru AL nebo AX

číslo V/V brány:

| | |
|-------------|---------------------|
| IN AL, imm8 | imm8 ... < 0, 255 > |
| IN AX, imm8 | imm8 ... < 0, 255 > |
| IN AL, DX | DX ... < 0, 65535 > |
| IN AX, DX | DX ... < 0, 65535 > |

16bitový přenos: imm8, imm8+1
DX, DX+1

Ovládání V/V

- *OUT* – OUTPUT TO PORT

Přenos bajtu nebo slova z registru AL nebo AX do V/V brány.

| |
|--------------|
| OUT imm8, AL |
| OUT imm8, AX |
| OUT DX, AL |
| OUT DX, AX |

9.22 Další instrukce přesunů dat

Další instrukce přesunů dat

- *XCHG* – cílový, zdrojový ... zamění obsahu

| | |
|-----------------|----------------|
| XCHG r/m8, r8 | XCHG AL, AH |
| XCHG r8, r/m8 | XCHG AX, slovo |
| XCHG r/m16, r16 | |
| XCHG r16, r/m16 | |

- *XLAT* – provede AL:=DS:[BX+AL], tj. dej bajt pořadí AL

- *LEA* – r16, imm16

vlož offset, tj. totožné s: MOV r16, offset

MOV BX, offset Tabulka ≡ LEA BX, Tabulka

- *LDS*

r16, m16:16 LDS BX, Dvojslovo DS:BX:= obsah Dvojslovo

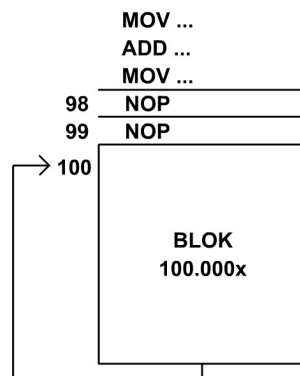
- *LES*

r16, m16:16 LES DI, Dvojslovo ES:DI:= obsah Dvojslovo

Instrukce NOP

- *NOP* – NO OPERATION operační kód 90h; jednobajtová; ekv. XCHG AX, AX

Využití např. pro optimalizaci umístění začátku těl cyklů:



Řídicí instrukce

- *HLT* – HALT zastavení procesoru
obnova:
 - NMI (návrat IRET je za instrukci HLT)
 - RESET
- *ESC* – Vzorek uvádějící instrukce 8087
- *WAIT* – Čekání na dokončení akce 8087
- *LOCK* – Instrukční prefix „zamykající“ sběrnici po dobu trvání instrukce
Příklad: LOCK ADD Slovo, DX ; Slovo:=Slovo+DX

9.23 Řetězcové instrukce

Řetězcové instrukce

| | | |
|--------------------|---|------------------|
| <i>MOVSB Bajt</i> | ES:[DI]:=DS:[SI] Je-li DF=0 : SI:=SI+1 ; DI:=DI+1 jinak : SI:=SI-1 ; DI:=DI-1 | žádný příznak |
| <i>MOVSW Slovo</i> | ES:[DI]:=DS:[SI] Je-li DF=0 : SI:=SI+2 ; DI:=DI+2 jinak : SI:=SI-2 ; DI:=DI-2 | žádný příznak |
| <i>CMPSB/CMPSW</i> | F:=DS:[SI] - ES:[DI] inc/dec SI, DI | všechny příznaky |
| <i>SCASB/SCASW</i> | F:=AL/AX - ES:[DI] inc/dec DI | všechny příznaky |

Řetězcové instrukce

| | | |
|--------------------|---|---------------|
| <i>LODSB/LODSW</i> | AL/AX:=DS:[SI] inc/dec SI | žádný příznak |
| <i>STOSB/STOSW</i> | ES:[DI]:=AL/AX inc/dec DI | žádný příznak |
| <i>REP</i> | instrukční prefix pro opakování řetězcových instrukcí | nastaví ZF |

REP

1. Je-li CX=0 ... konec
2. Uplatněno případné přerušování
3. Jedno provedení instrukce (řetězcové)
4. CX:=CX-1
5. Je-li *REP* ... jdi na 1
Je-li *REPZ* (*REPE*) a je-li ZF=1 ... jdi na 1. (*Má význam pouze u CMPS a SCAS*)
Je-li *REPZ* (*REPNE*) a je-li ZF=0 ... jdi na 1. (*Má význam pouze u CMPS a SCAS*)
Jinak neopakuj = KONEC.

9.24 Instrukce pro podporu BCD aritmetiky

Instrukce pro podporu BCD aritmetiky

BCD číslice 4 bity; 0 - 9 ; půlbajt (nibble)

Nezhuštěný tvar Unpacked Decimal

1 číslice v dolním půlbajtu, horní musí být rovna 0

Zhuštěný tvar Packed Decimal

2 číslice v jednom bajtu

- *AAA* – Ascii Adjust After Addition
Je-li AF=1 \vee AL>9 \Rightarrow AH:=AH+1; AL:=AL+6 ; AL1-4:=0; AF:=CF:=1
Jinak: AF:=CF:=0
- *AAD* – Ascii Adjust AX Before Division

10 Architektura procesorů Intel – Procesor 80186

Procesor Intel 80186

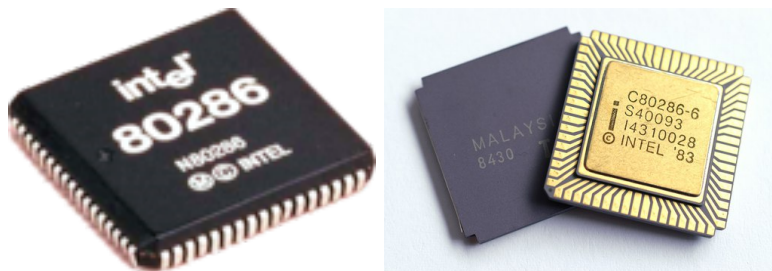
- mírně vylepšená 8086 (nebo 8088 ve verzi 80188),
- v počítačích PC se neuplatnil,
- používáno ve speciálních zařízeních.
- Má uvnitř integrován DMA řadič, řadič prerušení, ...
- Několik nových instrukcí ENTER, LEAVE, PUSHA, POPA, BOUND, UD2, INS, OUTS.



11 Architektura procesorů Intel – Procesor 80286

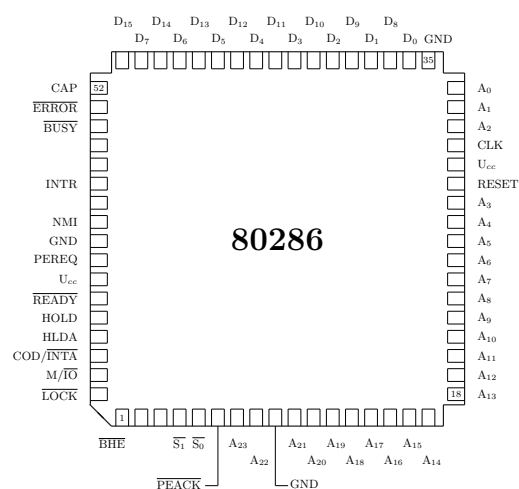
Procesor Intel 80286

- 16bitový procesor,
- od 1982, cca do 1990,
- frekvence 6 - 16 MHz,
- nové počítače PC AT,
- 24bitová adresová sběrnice, tj. 16 MB RAM.



11.1 Zapojení

Zapojení procesoru 80286



CAP Mezi tento vývod a vývod GND musí být zapojen kondenzátor kapacity $0,047 \mu\text{F} \pm 20\%$ 12V vyhlazující nežádoucí napěťové zákmity.

PEREQ Signálem koprocessor žádá procesor o vyslání operandu.

PEACK Signálem procesor oznamuje koprocessoru, že vysílá operand.

BUSY Aktivní úroveň signálu oznamuje, že koprocessor provádí výpočet. Signál je testován instrukcí WAIT.

ERROR Signálem koprocessor oznamuje chybový stav.

11.2 Režimy

Režimy procesoru 80286

- Reálný režim
 - Je nastaven po inicializaci procesoru.
 - Je slučitelný s procesorem 8086.
- Chráněný režim
 - Zapíná se programově z reálného režimu.
 - Adresuje 16 MB reálné paměti a 1 GB virtuální paměti.
 - Poskytuje prostředky 4úrovňové ochrany.
 - Nelze se vrátit z chráněného režimu zpět do reálného.

Rozdíl reálného režimu oproti 8086

24bitová adresová aritmetika 80286 vs. 20bitová adresová aritmetika 8086:

$$\begin{array}{r}
 \text{8086:} \quad \begin{array}{cccccc} & & \text{F} & \text{F} & \text{F} & \text{F} \\ + & & & \text{F} & \text{F} & \text{F} & \text{F} \\ \hline & & 0 & \text{F} & \text{F} & \text{E} & \text{F} \end{array} \quad (\text{tj. } 65\,519_{10}, 64 \text{ K je } 65\,536) \\
 \\
 \text{80286:} \quad \begin{array}{cccccc} & & \text{F} & \text{F} & \text{F} & \text{F} \\ + & & & \text{F} & \text{F} & \text{F} & \text{F} \\ \hline & & 1 & 0 & \text{F} & \text{F} & \text{E} & \text{F} \end{array} \quad (\text{tj. } 1 \text{ MB} + 65\,519_{10})
 \end{array}$$

11.3 Příznakový registr

Příznakový registr 80286

| | | | | | | | | | | | | | | | |
|----|----|------|----|----|----|----|----|----|---|----|---|----|---|----|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | NT | IOPL | OF | DF | IF | TF | SF | ZF | | AF | | PF | | CF | |

Příznakový registr je oproti 8086 rozšířen o příznaky NT a IOPL použitelné pouze v chráněném režimu: **NT** (Nested Task) určuje režim práce instrukce IRET. Je-li NT=0, provádí IRET klasický návrat z přerušení. Je-li NT=1, přepne se při provádění IRET proces podle zpětného ukazatele právě aktivního TSS.

IOPL (I/O Privilege Level) určuje úroveň oprávnění, při které může proces ještě provádět V/V instrukce. Vyšší hodnota představuje nižší úroveň oprávnění.

Ostatní příznaky mají stejný význam jako u procesoru 8086.

11.4 Registr MSW

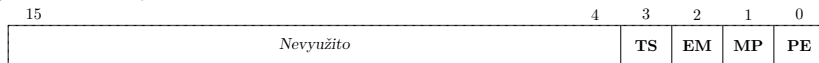
Registr MSW (Machine Status Word)

| | | | | | | | |
|-----------|---|---|---|----|----|----|----|
| 15 | 4 | 3 | 2 | 1 | 0 | | |
| Nevyužito | | | | TS | EM | MP | PE |

PE (Protected Mode Enable) zapíná *chráněný režim* procesoru. Po inicializaci procesoru (signálem RESET) je zapnut *reálný režim*. Nastavením tohoto příznaku se procesor přepne do *chráněného režimu*. Zpět do *reálného režimu* lze procesor vrátit pouze inicializací procesoru (RESET).

MP (Monitor Processor Extension) indikuje fyzickou přítomnost koprocessoru (např. matematického koprocessoru 80287).

Registr MSW - pokračování



EM (Emulate Processor Extension) zapíná programovou emulaci koprocесoru tehdy, není-li koprocесor instalován (je-li EM=1, způsobí instrukce koprocесoru přerušení INT 7).

TS (Task Switch) se nastavuje vždy přepnutím procesu. Je používán koprocесorem ke zjištění, že v procesoru se vyměnil „zadavatel“ úkolů.

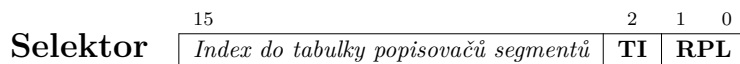
11.5 Adresace paměti v chráněném režimu

Adresace paměti v chráněném režimu 80286

Pojmy:

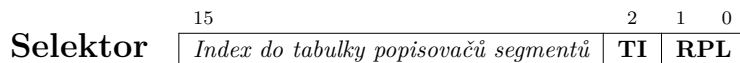
- **Proces**
- **Segment** je definován:
 - 1. bází segmentu (adresou začátku segmentu)
 - 2. limitem segmentu (délkou segmentu v bajtech – 1)
 - 3. přístupovými právy a typem segmentu.
- **Globální adresový prostor**
- **Lokální adresový prostor**
- **Virtuální adresa** (sektor:offset)

Segment selector



Selektor obsahuje 13 bitů (8192 kombinací) **indexu** do **tabulky popisovačů segmentů** lokálního nebo globálního adresového prostoru a další 3 informační bity

Segment selector - pokračování



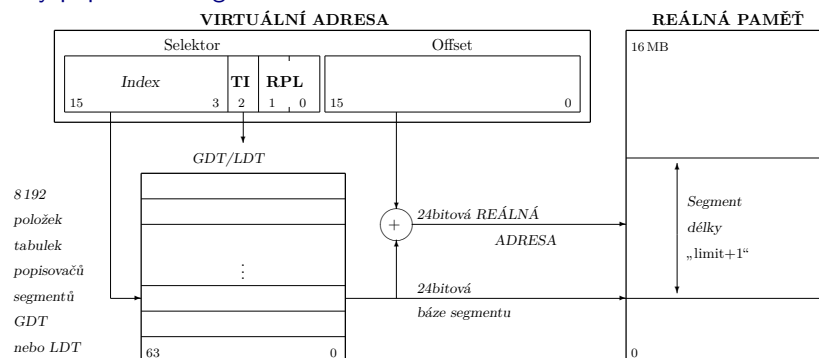
RPL (Requested Privilege Level) představuje úroveň oprávnění, kterou proces nabízí při přístupu k tomuto segmentu.

TI (Table Indicator) indikuje, ukazuje-li index do tabulky popisovačů segmentů lokálního adresovacího prostoru (TI=1) nebo globálního adresovacího prostoru (TI=0).

Kombinace Index=0 a zároveň TI=0 se nazývá **neplatný selektor** a má speciální význam.

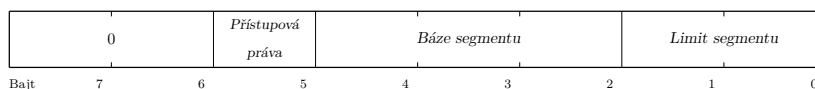
11.6 Tabulky popisovačů segmentů

Tabulky popisovačů segmentů



Transformace virtuální adresy

Transformace virtuální adresy na reálnou pomocí tabulek popisovačů segmentů v procesoru 80286
Virtuální adresový prostor 1GB: 14b. Selektor + 16b. Offset



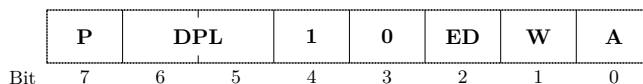
Položka tabulky popisovačů segmentů

Typ popisovaného segmentu je definován obsahem bajtu **přístupová práva**. Podle typu segmentu rozlišujeme v 80286 tyto 4 základní třídy popisovačů:

1. popisovač segmentu obsahujícího data (datový segment),
2. popisovač segmentu obsahujícího instrukce (instrukční segment),
3. popisovač segmentu obsahujícího informace pro systém (systémový segment),
4. popisovač brány.

11.7 Popisovač datového segmentu

Popisovač datového segmentu - pokračování

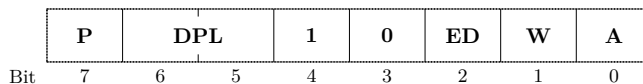


P (Segment Present) je nastaven na jedničku tehdy, je-li obsah segmentu uložen v reálné paměti. Není-li, je nulový.

DPL (Descriptor Privilege Level) určuje úroveň oprávnění přidělenou segmentu, který je popisovačem adresován.

W (Writable) je nastaven na 1, pokud je povoleno čtení i zápis do segmentu. Zásobník musí mít vždy $W=1$. Nulová hodnota bitu **W** zakazuje zápis dat a je povoleno pouze čtení.

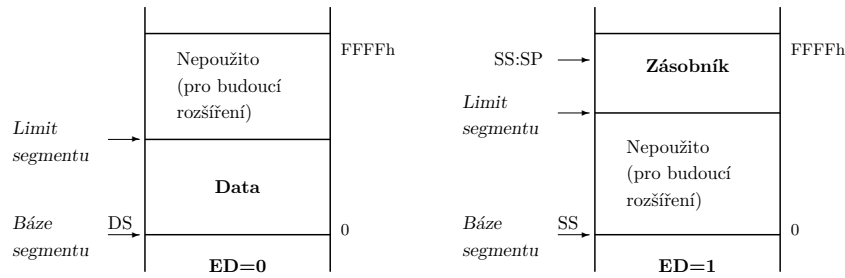
Popisovač datového segmentu



A (Accessed) nastavuje procesor na jedničku při každém přístupu k této položce v tabulce popisovačů segmentů (zavedení do segmentového registru nebo použití instrukce testující selektor). Procesor tento příznak nenuluje. Je určen operačnímu systému ke sledování četnosti přístupů ke konkrétním segmentům.

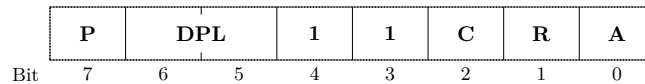
ED (Expansion Direction) indikuje, kterým směrem se bude obsah segmentu rozšiřovat. Datové segmenty mohou obsahovat klasická data nebo zásobníky.

- Je-li nastaveno $ED=0$ (data), bude se obsah segmentu rozšiřovat směrem k vyšším adresám. Data se ukládají (v rámci 64 KB segmentu) od adresy 0000 směrem k adrese 0FFFFh. Při požadavku na zvětšení obsahu se musí zvětšit hodnota limitu segmentu.
- Je-li nastaveno $ED=1$ (zásobník), bude se obsah segmentu rozšiřovat směrem k nižším adresám. Položky zásobníku se ukládají od adresy 0FFFFh směrem k adrese 0000 (uvnitř 64 KB segmentu). Při požadavku na zvětšení obsahu se musí zmenšit hodnota limitu segmentu (ten se totiž stále počítá od adresy 0).



11.8 Popisovač instrukčního segmentu

Popisovač instrukčního segmentu



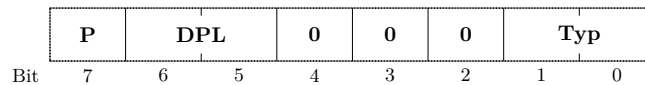
C (Conforming) nulový sděluje, že podprogramy volané v tomto segmentu budou mít nastavenou úroveň oprávnění odpovídající úrovni segmentu, v němž se nacházejí. Je-li $C=1$, bude volanému podprogramu v tomto segmentu přidělena úroveň oprávnění segmentu, z něhož je volán.

R (Readable) nulový zakazuje čtení obsahu segmentu. Je povoleno pouze obsah segmentu spustit. Jedničková hodnota bitu povoluje jak spuštění, tak i čtení segmentu.

11.9 Popisovač systémového segmentu

Popisovač systémového segmentu

Tento popisovač smí být umístěn pouze v GDT.



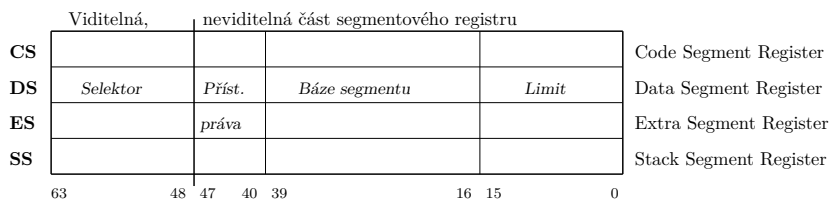
Typ=1 označuje segment stavu procesu (TSS – Task State Segment) pro právě neaktivní proces.

Typ=3 označuje segment stavu procesu pro právě aktivní proces.

Typ=2 označuje segment lokální tabulky popisovačů segmentů (LDT – Local Descriptor Table).

11.10 Segmentové registry

Segmentové registry



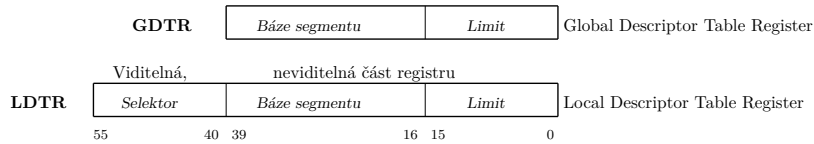
Plnění:

CS: JMP, CALL, RET – vše vzdálené (FAR) varianty.

DS, ES, SS: MOV, LES, LDS *selektor* .

11.11 Registry GDTR a LDTR

Registry GDTR a LDTR



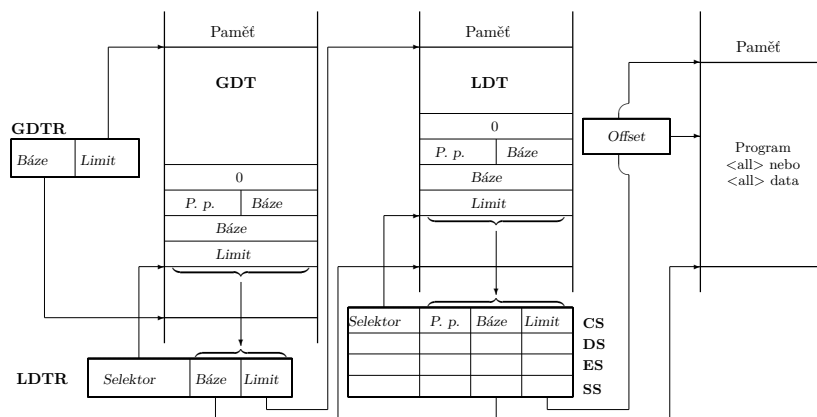
Plnění:

GDTR: LGDT *operand obsahující bázi (24b) a limit (16b)*

LDTR: LLDT *selektor(16b)*

11.12 Použití GDTR, LDTR a segmentových registrů

Použití GDTR, LDTR a segmentových registrů



Sdílení jednoho segmentu více popisovači

- Aplikační proces (instrukční segment a nutnost zápisu ladících bodů do kódu programu).
- Jeden proces plní vyrovnávací paměť, druhý proces ji smí pouze číst.
- Modifikace obsahu systémových segmentů.

11.13 Úrovně oprávnění

Úrovně oprávnění (Privilege Levels)

nejvyšší nejnižší

0 1 2 3

úroveň 0 ... jádro operačního systému (řízení procesoru, V/V operací),

úroveň 1 ... služby poskytované operačním systémem (plánování procesů, organizace V/V, přidělování prostředků),

úroveň 2 ... systémové programy a podprogramy z knihoven (systém obsluhy souborů, správa knihoven),

úroveň 3 ... uživatelské aplikace.

DPL (Descriptor Privilege Level) je uložen ve dvou bitech bajtu **přístupová práva** popisovače segmentu. Obsahuje úroveň oprávnění přidělenou obsahu segmentu.

CPL (Current Privilege Level) je zapsán ve dvou nejnižších bitech **selektoru CS** (tj. v poli označeném RPL). Představuje momentální úroveň oprávnění přidělenou právě prováděnému procesu.

RPL (Requested Privilege Level) je uložen v bitech 0 a 1 **selektoru segmentového registru** a obsahuje úroveň oprávnění, kterou proces nabízí při přístupu k určitému segmentu.

EPL (Effective Privilege Level) je numerické maximum CPL a RPL (tedy hodnota nižší úrovně oprávnění).

11.14 Zpřístupnění datového segmentu

Zpřístupnění datového segmentu

MOV DS,AX ; Naplnění a kontrola
; přístupových práv.
MOV DL,DS:Adresa ; Čtení datového segmentu.
MOV DS:Adresa,DL ; Zápis do datového
; segmentu (je-li W=1).

$$\mathbf{CPL \leq DPL}$$

$$\mathbf{RPL \leq DPL}$$

$$\mathbf{\underline{\underline{Max(CPL,RPL) \leq DPL}}}$$

$$\mathbf{EPL \leq DPL}$$

Proces přistupuje k datům pouze na stejné nebo nižší úrovni oprávnění.

11.15 Předání řízení do instrukčního segmentu

Předání řízení do instrukčního segmentu

JMP FAR PTR Navesti ; Skok do jiného
; instrukčního segmentu.
CALL FAR PTR Navesti ; Volání jiného
; instrukčního segmentu.
RET ; Návrat do jiného
; instruk. segmentu.
MOV DL,CS:Adresa ; Čtení instrukčního
; segmentu (je-li R=1).

$$\mathbf{CPL = DPL}$$

11.16 Brána pro předání řízení

Brána pro předání řízení (Call Gate)

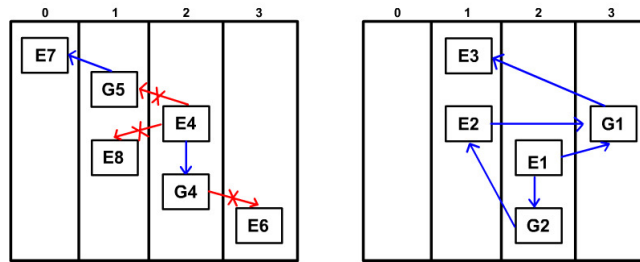
Brána je popisovač uložený v tabulce popisovačů segmentů.

CPL ≤ DPL brány

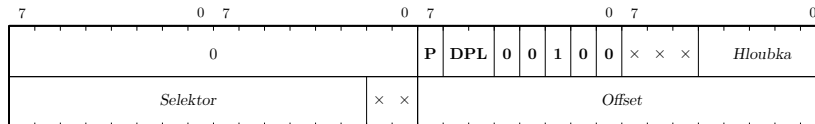
CPL ≥ DPL podprogramu

- Brána je na nižší úrovni oprávnění

- Podprogram na vyšší úrovni oprávnění

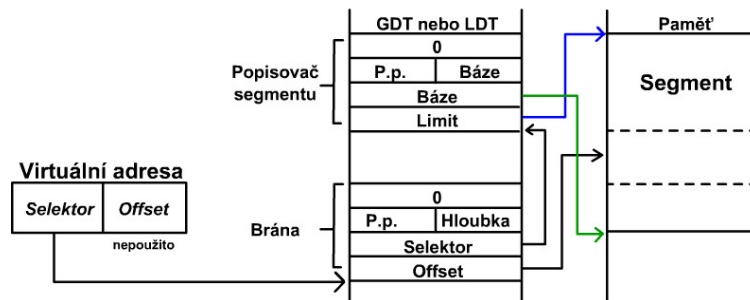


Popisovač brány pro předání řízení



11.17 Použití brány pro předání řízení

Použití brány pro předávání řízení



11.18 Předávání parametrů pomocí brány

Předávání parametrů pomocí brány

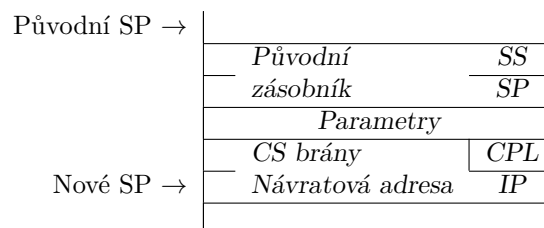
```

PUSH    Par1
PUSH    Par2
CALL    Podprogram
  
```

- Každý proces má vlastní zásobník.
- Každá úroveň oprávnění uvnitř procesu má vlastní zásobník.
- Parametry se do podprogramu předávají přes zásobník.
- Je-li podprogram na jiné úrovni oprávnění?

Činnost brány při předávání řízení:

1. Ukazatel vrcholu zásobníku (SS:SP) volajícího modulu (starý zásobník) se uloží do zásobníku volaného podprogramu (nový zásobník).
2. Ze starého zásobníku se zkopíruje hloubka slov do nového zásobníku.
3. Do nového zásobníku se vloží jako návratová adresa (CS:IP) adresa této brány. Tím může tento zásobník být použit volaným podprogramem.



11.19 Privilegované instrukce

Privilegované instrukce

- $CPL = 0$

| | |
|-------------|----------------------------------|
| LGDT | naplnění registru GDTR, |
| LIDT | naplnění registru IDTR, |
| LLDT | naplnění registru LDTR, |
| LTR | naplnění registru TR, |
| LMSW | naplnění registru MSW, |
| CLTS | nulování bitu TS v registru MSW, |
| HLT | zastavení procesoru. |

POPF a IRET směřují měnit IOPL pouze s $CPL=0$.

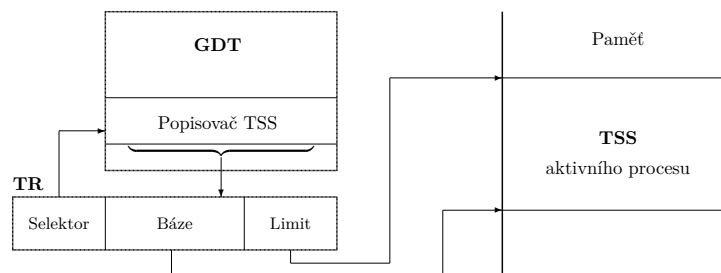
- $CPL \leq IOPL$

| | |
|--------------------------------|---------------------|
| IN, INS, INSB, INSW | čtení ze V/V brány, |
| OUT, OUTS, OUTSB, OUTSW | zápis na V/V bránu, |
| STI, CLI | změna příznaku IF, |
| prefix LOCK | blokování sběrnice. |

POPF smí měnit IF pouze s $CPL \leq IOPL$ (jinak se změna IF ignoruje). Provinění se trestá INT 13.

11.20 Segment stavu procesu

Segment stavu procesu (TSS) - Task State Segment



Na segment s TSS ukazuje popisovač systémového segmentu (smí být umístěn pouze v GDT).

- **Typ=3** sděluje, že jde o TSS právě aktivního procesu.
- **Typ=1** určuje, že jde o TSS právě neaktivního procesu.

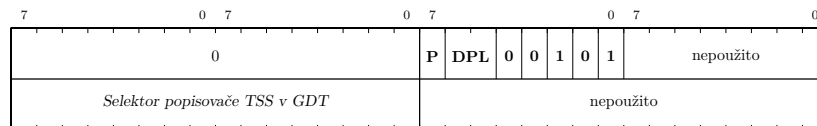
Pravidlo pro zpřístupnění systémového segmentu s TSS je stejné jako pro zpřístupnění datového segmentu, tj. $EPL \leq DPL$.

11.21 Segment stavu procesu - obsah

| | | |
|-----|-----------------|----|
| 15 | TSS | 0 |
| ... | | |
| | Selektor LDT | 42 |
| | Selektor DS | 40 |
| | Selektor SS | 38 |
| | Selektor CS | 36 |
| | Selektor ES | 34 |
| | DI | 32 |
| | SI | 30 |
| | BP | 28 |
| | SP | 26 |
| | BX | 24 |
| | DX | 22 |
| | CX | 20 |
| | AX | 18 |
| | F | 16 |
| | IP | 14 |
| | SS pro úroveň 2 | 12 |
| | SP pro úroveň 2 | 10 |
| | SS pro úroveň 1 | 8 |
| | SP pro úroveň 1 | 6 |
| | SS pro úroveň 0 | 4 |
| | SP pro úroveň 0 | 2 |
| | Zpětný ukazatel | 0 |

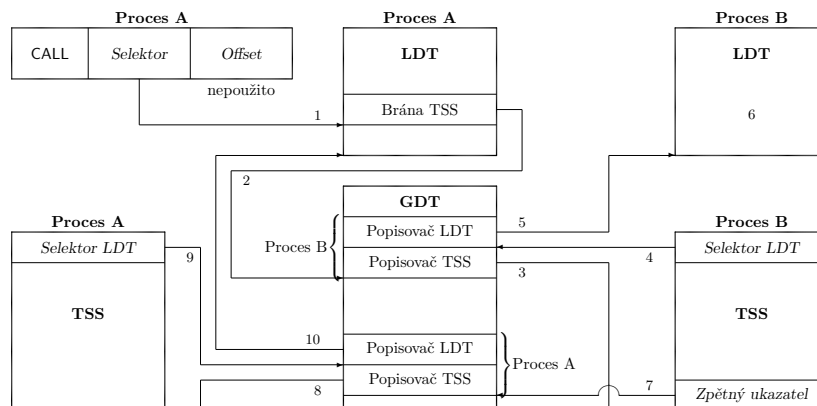
11.22 Brána zpřístupňující TSS

Brána zpřístupňující TSS



Přepnutí procesu může být vyvoláno:

- vzdáleným JMP nebo CALL, jehož selektor ukazuje na popisovač TSS nového procesu v GDT.
- vzdáleným JMP nebo CALL, jehož selektor ukazuje na bránu zpřístupňující TSS.
- IRET s nastaveným NT=1.
- přerušením, jehož přerušovací vektor ukazuje na bránu zpřístupňující TSS.



11.23 Přerušování

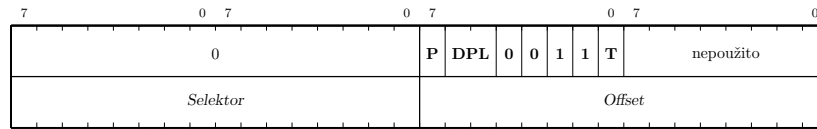
Přerušování

- Interrupt Descriptor Table (IDT) obsahuje až 256 popisovačů rutin obsluhujících přerušování.
- IDTR obsahuje adresu IDT (like GDTR).
- Popisovače v IDT jsou pouze tyto tři:

1. brána zpřístupňující TSS,
2. brána pro maskující přerušení (Interrupt Gate),
3. brána pro nemaskující přerušení (Trap Gate).

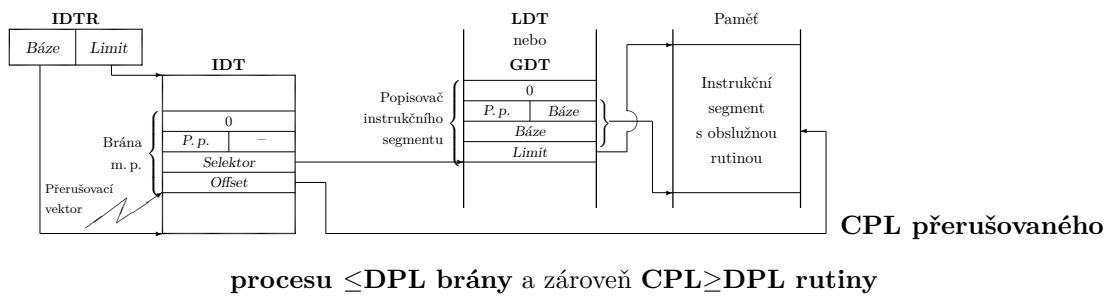
11.24 Brány pro přerušení

Brány pro přerušení



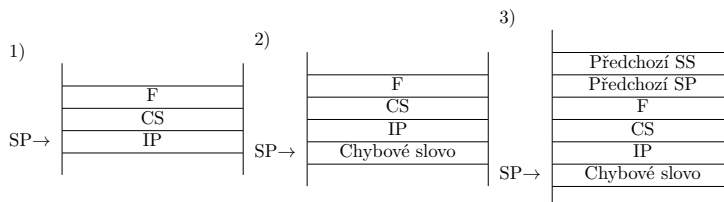
T=1 ... Brána pro nemaskující přerušení (nenuluje IF).

T=0 ... Brána pro maskující přerušení (nuluje IF).



11.25 Informace ukládaná do zásobníku

Informace ukládaná do zásobníku

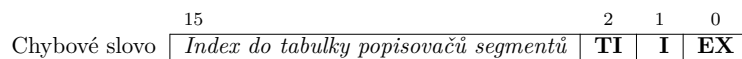


1. Žádné chybové hlášení.
2. Přerušení předává chybové slovo.
3. Přerušení předává chybové slovo a obsluha přerušení pracuje na jiné (vyšší) úrovni oprávnění – je uloženo původní SS:SP.

11.26 Formát chybového slova

Formát chybového slova

předávaného přerušeními 10 až 13



I index ukazuje do IDT (nikoli do GDT nebo LDT podle TI).

EX (External) přerušení bylo způsobeno vnější událostí bez zavinění procesu (např. INT 10: vnější přerušení přes bránu zpřístupňující TSS vyvolalo pokus o přepnutí na proces mající TSS s chybným obsahem).

11.27 Rezervovaná přerušení

Přerušení generovaná procesorem 80286 dělíme do tří kategorií:

- *Fault* do zásobníku uloží CS:IP ukazující **na instrukci**, která způsobila přerušení,
- *Trap* do zásobníku uloží CS:IP ukazující **za instrukci** (na následující instrukci), která přerušení způsobila,
- *Abort* v procesu nelze pokračovat a musí být násilně ukončen.

Rezervovaná přerušení 80286

| Číslo vektoru | Určení vektoru | Typ přerušení | Chybové slovo? |
|---------------|-----------------------------------|---------------|----------------|
| 0 | Dělení nulou | Fault | ne |
| 1 | Krokovací režim | Trap | ne |
| 2 | Nemaskovatelná přerušení | - | ne |
| 3 | Ladící bod | Trap | ne |
| 4 | Přeplnění | Trap | ne |
| 5 | Kontrola mezí | Fault | ne |
| 6 | Chybný operační kód | Fault | ne |
| 7 | Nedostupnost koprocessoru | Fault | ne |
| 8 | Dvojnásobný výpadek segmentu | Abort | ano (=0) |
| 9 | Překročení segmentu koprocessorem | Abort | ne |
| 10 | Chybný TSS | Fault | ano |
| 11 | Výpadek segmentu | Fault | ano |
| 12 | Výpadek segmentu se zásobníkem | Fault | ano |
| 13 | Obecná chyba ochrany | Fault | ano |
| 16 | Chyba koprocessoru | Fault | ne |

11.28 Počáteční nastavení procesoru

Počáteční nastavení procesoru

| Registr | Obsah |
|-------------|---------|
| F | 0002h |
| MSW | FFF0h |
| IP | FFF0h |
| Selektor CS | F000h |
| Selektor DS | 0000h |
| Selektor SS | 0000h |
| Selektor ES | 0000h |
| Báze CS | FF0000h |
| Báze DS | 000000h |
| Báze SS | 000000h |
| Báze ES | 000000h |
| Limit CS | FFFFh |
| Limit DS | FFFFh |
| Limit SS | FFFFh |
| Limit ES | FFFFh |
| Báze IDT | 000000h |
| Limit IDT | FFFFh |

Procesor provádí tyto činnosti:

- zakáže přerušení (IF:=0),
- nastaví reálný režim bez koprocessoru (PE:=0, MP:=0, EM:=0),
- IDT se naplní nulami,
- DS, ES a SS jsou naplněny tak, aby ukazovaly do prvních 64 KB paměti,
- obsah CS:IP ukazuje na první instrukci, která musí být na adrese FF0000h:FFF0h=FFFFFF0h,
- první instrukční segment zpřístupněný po inicializaci systému je posledních 64 KB paměti (CS=FF0000h).

- Bezprostředně po inicializaci procesoru jsou adresové vodiče $A_{20} \div A_{23}$ nastaveny na jedničky při všech přístupech adresovaných přes registr CS.
- Tento stav trvá do první změny obsahu CS, potom jsou vodiče $A_{20} \div A_{23}$ vynulovány.

11.29 Zapnutí chráněného režimu

Zapnutí chráněného režimu

1. do paměti zavést programy a odpovídající tabulky popisovačů,
2. nastavit GDTR a IDTR,
3. zapnout chráněný režim nastavením bitu PE:=1 registru MSW.
4. provést blízký skok JMP proto, aby se zrušil obsah interních front procesoru, ve kterých jsou uloženy předvybrané instrukce (výběr instrukcí totiž závisí na zvoleném režimu procesoru),
5. vytvořit TSS inicializačního procesu a nastavit obsah TR,
6. naplnit LDTR,
7. inicializovat ukazatel vrcholu zásobníku SS:SP,

Zapnutí chráněného režimu – pokračování

8. všechny segmenty v paměti označit P:=0,
9. nastavit příznakový registr F a registr stavu procesoru MSW,
10. inicializovat externí zařízení,
11. zabezpečit obsluhu všech možných přerušení,
12. naplnit DS:=0
ES:=0
CS → JMP FAR ...
13. povolit přerušení (IF:=1),
14. zahájit provádění prvního programu.

12 Architektura procesorů Intel – Procesor 80386

Procesor Intel 80386

- 32bitový procesor,
- od 1986 cca do 1994,
- 16 MHz až 40 MHz,
- „zakladatel“ architektury IA-32,
- 32bitová adresová sběrnice, tj. max. 4 GB RAM,
- 32bitová datová sběrnice,
- alternativní název i386DX,
- varianta 386SX s 16bitovou datovou a 24bitovou adresovou sběrníci,
- matematický koprocesor zvláště i387,
- i386SL pro laptop počítače, nižší spotřeba.

12.1 Popis signálů a registry

Popis signálů procesoru Intel 80386

Procesor je integrován do čtvercového keramického integrovaného obvodu, který má vývody na spodním povrchu (PGA – Pin Grid Array). Obvod má 132 vývodů.

$\overline{D}_0 \div \overline{D}_{31}$ 32bitová obousměrná datová sběrnice.

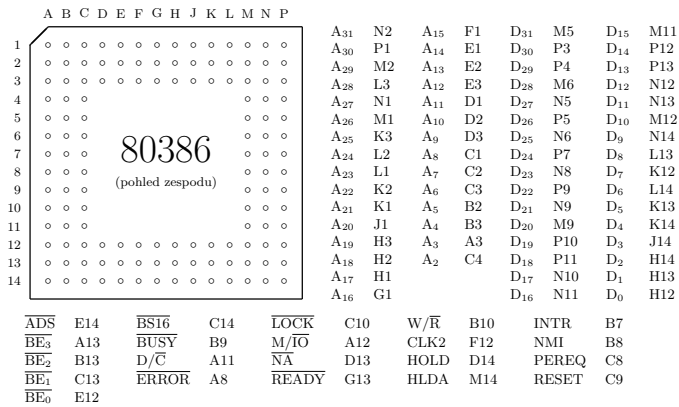
$\overline{A}_2 \div \overline{A}_{31}$ 32bitová adresová sběrnice adresující 32bitová dvojslova.

$\overline{BE}_0 \div \overline{BE}_3$ Bližší určení přenášených bajtů v rámci dvojslova.

$\overline{BS16}$ Volba 16bitového přenosu dat.

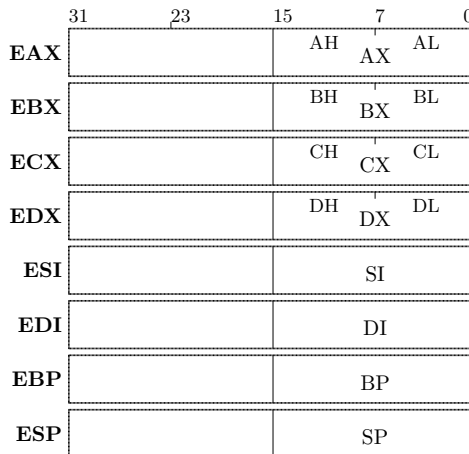
\overline{NA} (Next Address) Slouží k zahájení výběru obsahu další adresy při proudovém zpracování.

$\overline{D/C}$, \overline{ADS} , $\overline{W/R}$ jsou signály určené pro řízení sběrnice.

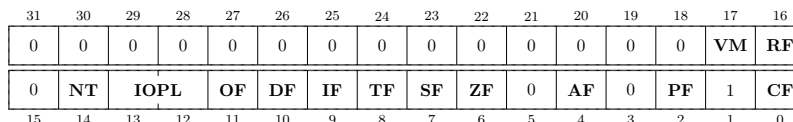


GND: A2 A6 A9 B1 B5 B11 B14 C11 F2 F3 F14 J2 J3 J12 J13 M4 M8 M10 N3 P6 P14
 U_{cc}: A1 A5 A7 A10 A14 C5 C12 D12 G2 G3 G12 G14 L12 M3 M7 M13 N4 N7 P2 P8

Registry procesoru 80386



EFLAGS:



VM (Virtual 8086 Mode) zapíná režim *virtuální 8086* pro proces, jemuž obsah příznakového registru náleží. Příznak VM smí programátor nastavovat pouze v chráněném režimu, a to instrukcí IRET, a jenom na úrovni oprávnění 0. Příznak je také modifikován mechanismem přepnutí procesu.

RF (Resume Flag) maskuje opakování ladícího přerušování.

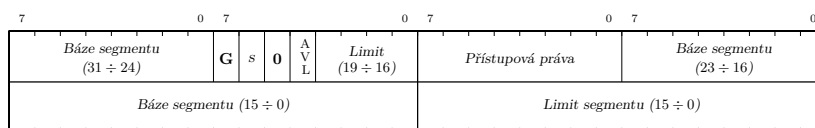
- Registry pro uložení selektoru datových segmentů: **DS**, **ES**, **FS** a **GS**
- Velikost viditelných částí registrů se nezměnila (selektor je stále 16bitový), ale zvětšila se neviditelná část tak, že báze segmentu je 32bitová.

12.2 Adresace v chráněném režimu a řídicí registry

Adresace v chráněném režimu 80386

- *Selektor* je stejný jako v 80286.
- *Offset* je 32bitový.
- *Límit segmentu* může mít velikost až 4 GB - 1.
- *Báze segmentu* je 32bitová (tj. 0 až 4 GB - 1).

Popisovače segmentů



G (Granularity)

- =0 ... jednotka limitu je 1 B (max. 1 MB),
- =1 ... jednotka limitu je 4 KB (max. 4 GB).

AVL (Available for Programmer Use)

s závisí na typu popisovače.

Popisovač datového segmentu

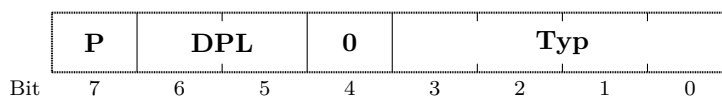
B (Big)

=0 ... segment podle pravidel 80286 (max. 64 KB), implicitní velikost položky ukládané do zásobníku je 16 bitů,

=1 ... segment podle pravidel 80386 (max. 4 GB), zásobník lze plnit od adresy FFFFFFFFh, implicitní velikost položky ukládané do zásobníku je 32 bitů.

Popisovač systémového segmentu

Bit s není použit.



Typ=0 ... nepovolená hodnota,

- 1 ... TSS neaktivního procesu 80286,
- 2 ... LDT 80286 a 80386,
- 3 ... TSS aktivního procesu 80286,
- 4 ... brána pro předání řízení 80286,
- 5 ... brána zpřístupňující TSS 80286 a 80386,
- 6 ... brána pro maskující přerušení 80286,
- 7 ... brána pro nemaskující přerušení 80286,
- 8 ... nepovolená hodnota,
- 9 ... TSS neaktivního procesu 80386,
- A ... nepovolená hodnota,
- B ... TSS aktivního procesu 80386,
- C ... brána pro předání řízení 80386,
- D ... nepovolená hodnota,
- E ... brána pro maskující přerušení 80386,
- F ... brána pro nemaskující přerušení 80386.

Popisovač instrukčního segmentu

D (Default)

- =0 ... implicitní velikost adres a operandů je 16 bitů,
- =1 ... implicitní velikost adres a operandů je 32 bitů,

Explicitní určení velikosti zajišťují instrukční prefixy:

66h mění implicitní velikost **operandu** a

67h mění implicitní velikost **adresy**.

| | | | | | | | | |
|----------------------------|----|-----|-----|-----|----|-----|-----|-----|
| D= | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| Prefix 66h (vel. operandu) | ne | ne | ano | ano | ne | ne | ano | ano |
| Prefix 67h (vel. adresy) | ne | ano | ne | ano | ne | ano | ne | ano |
| Velikost operandu v bitech | 16 | 16 | 32 | 32 | 32 | 32 | 16 | 16 |
| Velikost adresy v bitech | 16 | 32 | 16 | 32 | 32 | 16 | 32 | 16 |

V reálném režimu není, ani po použití prefixu změny velikosti adresy, povoleno adresovat větší segmenty než 64 KB. Offset, který by překročil hodnotu FFFFh, způsobí přerušení INT 13.

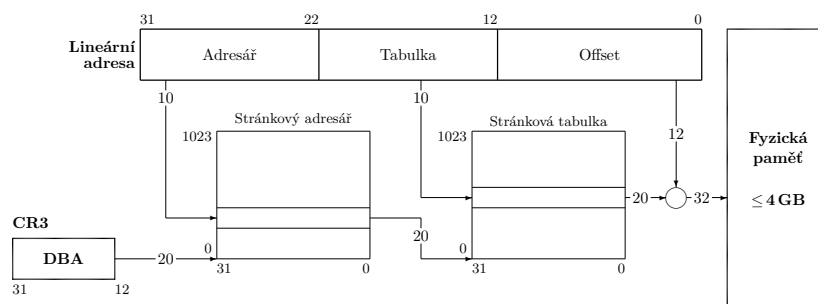
12.4 Stránkování, Translation Look-aside Buffer (1)

Stránkování

logická adresa → lineární adresa → fyzická adresa
Selektor₁₆ : Offset₃₂ *32 b* *32 b*

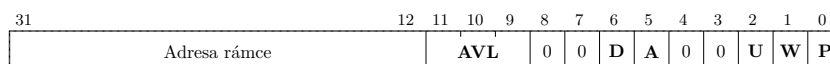
Rámec a stránka kapacity 4 KB

Zapnutí stránkování PG:=1 (bit v CR0)



Každý proces má vlastní stránkový adresář (CR3 je uloženo v TSS).

Položka stránkové tabulky a adresáře



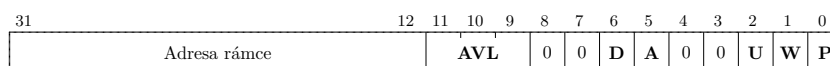
Adresa rámce je horních 20 bitů adresy rámce.

AVL (Available)

D (Dirty) nastavuje procesor při změně obsahu rámce. Ve stránkovém adresáři je tento bit nedefinován.

A (Accessed) nastavuje procesor při každém použití tohoto specifikátoru.

Položka stránkové tabulky a adresáře – pokračování



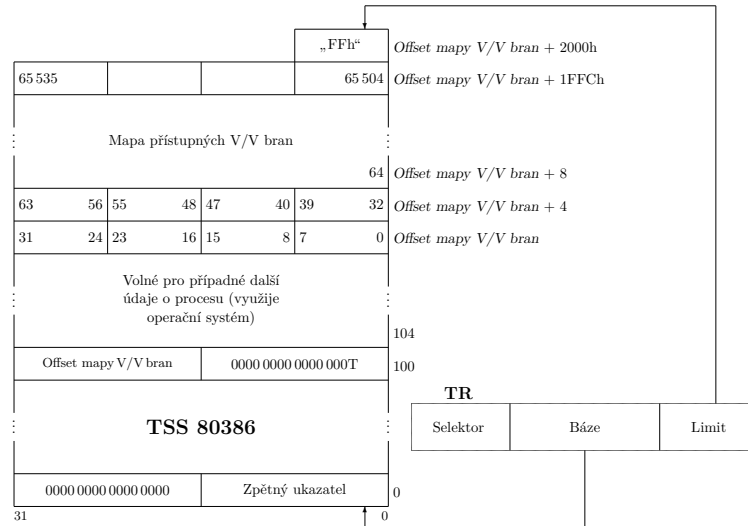
U (User Accesible) Pracuje-li proces na úrovni oprávnění CPL=3, smí k této stránce přistupovat při U=1. Procesy s CPL<3 smějí přistupovat ke všem stránkám bez ohledu na hodnotu bitu U.

W (Writeable) Pracuje-li proces na úrovni CPL=3, smí do této stránky zapisovat při W=1. Procesy s CPL<3 smějí zapisovat do všech stránek bez ohledu na hodnotu bitu W.

12.5 Mapa přístupných V/V bran, přerušení

Mapa přístupných V/V bran

- Pro kontrolování V/V instrukcí pouze tehdy, je-li CPL>IOPL.
- Je-li bit mapy =0 ... V/V operace se povolí,
- je-li bit mapu =1 ... generuje se INT 13.
- Pracuje-li V/V instrukce se slovem nebo dvojslovem ... testují se všechny odpovídající bity.



Rezervovaná přerušení

INT 1 Ladící přerušení (Debug Exceptions)

1. při čtení/zápisu z/do paměti byl detekován ladící bod (Trap),
2. při výběru instrukce byl detekován ladící bod (Fault),
3. po provedení instrukce v krokovacím režimu (Trap),
4. při přepnutí na proces mající v TSS T=1 (Trap),
5. nedovoleným přístupem k ladícím registrům při GD=1 (Fault).

Rezervovaná přerušení – pokračování

INT 14 Výpadek stránky (Page Fault)

(typ: Fault) Přerušení generuje stránkovačí jednotka při:

1. proces nemá dostatečnou úroveň oprávnění pro přístup ke stránce,
2. ve stránkovačích tabulkách je detekováno P=0.

Při přerušení je naplněn CR2 lineární adresou, která vyvolala přerušení. Chybové slovo má zvláštní tvar:

| | | | | |
|-----------|---|---|----------|----------|
| 31 | 3 | 2 | 1 | 0 |
| nepoužito | | | U | W |
| | | | P | |

P (Present) logický součin bitů P obou transformačních tabulek,

W (Write) přerušení vyvolal zápis (W=1) nebo čtení (W=0).

U (User Level) je-li =1, měl proces CPL=3.

13 Architektura procesorů Intel – Procesor 80486

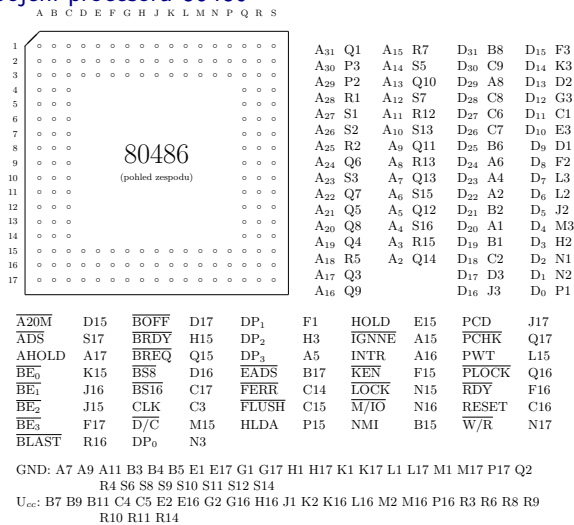
Procesor Intel 80486

- 32bitový procesor,

- od 1989 cca do 1993,
- 25 MHz až 120 MHz,
- 32bitová adresová sběrnice, tj. max. 4 GB RAM,
- 32bitová datová sběrnice,
- obsahuje jednotku operací v pohyblivé řádové čárce (koprocessor),
- obsahuje interní vyrovnávací paměť (cache),
- obsahuje novou technologii blízkou RISC,
- alternativa i486SX bez koprocessoru,
- i486DX novější verze; i486DX2 dvojnásobná frekvence,
- stále napájení 5 V, později DX4 už s jenom 3,3 V.

13.1 Zapojení

Zapojení procesoru 80486



$\overline{DP_0} \div \overline{DP_3}$ Paritní bit pro každý bajt přenášený po sběrnici.

\overline{PCHK} Chyba parity na sběrnici.

\overline{PLOCK} , \overline{ADS} , \overline{RDY} , \overline{BRDY} , \overline{BLAST} , \overline{BOFF} Signály pro řízení sběrnice.

\overline{AHOLD} , \overline{EADS} Signály pro řízení vnitřní vyrovnávací paměti.

\overline{KEN} Povoluje nebo zakazuje použití vnitřní vyrovnávací paměti.

\overline{FLUSH} Pokyn k vyprázdnění vnitřní vyrovnávací paměti.

PWT, PCD Signály přenášející hodnoty bitů PWT a PCD.

\overline{FERR} Signál oznamuje chybu koprocessoru (podobně jako \overline{ERROR}).

\overline{IGNNE} Ignorování chyb hlášených koprocessorem.

$\overline{A20M}$ Maskování adresy podle pravidel 8086.

13.2 Rysy procesoru, příznakový registr

- Jednotka operací v pohyblivé řádové čárce
 - **Floating-Point Unit** – Ovládá se stejně jako 80387. Je programově kompatibilní s předcházejícími typy matematických koprocessorů Intel.
- Interní vyrovnávací paměť
 - **Internal Cache** – Je společná pro data i instrukce, má kapacitu 8 KB.

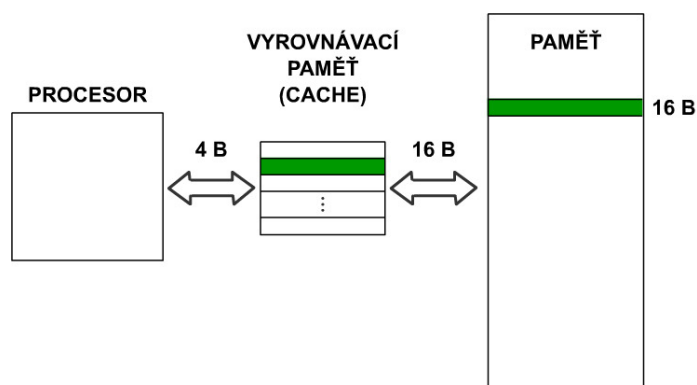
Příznakový registr i486

| | | | | | | | | | | | | | | | |
|----|----|------|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | AC | VM | RF |
| 0 | NT | IOPL | OF | DF | IF | TF | SF | ZF | 0 | AF | 0 | PF | 1 | CF | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

AC (Alignment Check) zapíná generování přerušení INT 17 při odkazu na paměť, který není „zarovnan“ na hranici odpovídající délce zpřístupňovaného objektu. Platí pouze pro proces s CPL=3.

13.3 Schéma činnosti vyrovnávací paměti

Schéma činnosti vyrovnávací paměti



Write-Through Write-Back

13.4 Řídící registry CR0, CR3

Řídící registr CR0 procesoru 80486

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PG | CD | NW | | | | | | | | | | | AM | | WP |
| | | | | | | | | | | NE | 1 | TS | EM | MP | PE |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

CD (Cache Disable) Je-li CD=1, je IVP (Interní vyrovnávací paměť) vypnuta tak, že položky, které při čtení nebyly ve vyrovnávací paměti nalezeny, se do ní nezapíší.

Po inicializaci procesoru signálem RESET je nastaveno CD=1.

Řídící registr CR0 procesoru 80486 – pokračování

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PG | CD | NW | | | | | | | | | | | AM | | WP |
| | | | | | | | | | | NE | 1 | TS | EM | MP | PE |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

NW (Not Write-Through) Je-li NW=1, není zápisem do paměti změněn obsah IVP ani tehdy, má-li adresa zapisovaného objektu svoji položku v IVP.

Po inicializaci procesoru signálem RESET je nastaveno NW=1.

Řídicí registr CR0 procesoru 80486 – pokračování

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PG | CD | NW | | | | | | | | | | | AM | | WP |
| | | | | | | | | | | NE | 1 | TS | EM | MP | PE |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

AM (Alignment Mask) AM=1 zapíná funkci AC.

WP (Write Protect) je-li WP=1, zakazuje zápis do stránek označených W=0 i procesům na úrovni oprávnění CPL<3.

NE (Numerics Exception) sděluje, jak se mají procesoru 80486 oznamovat chyby zjištěné v jednotce pohyblivé řádové čárky.

Řídicí registr CR3 procesoru 80486

| | | | | | | |
|-------------------------------------|----|----|---|---|---|-------|
| 31 | 12 | 11 | 4 | 3 | 0 | |
| Registr adresy stránkového adresáře | | | | | | |
| | | | | P | P | nepo- |
| | | | | C | W | užito |
| | | | | D | T | |

Bity **PWT** (Page Write-Through) a **PCD** (Page Cache Disable) slouží k řízení vyrovnávacích pamětí není-li zapnuto stránkování nebo se stránkování z nějaké příčiny obchází.

13.5 Stránkování

Stránkování i486

Specifikátor stránk. adresáře a tabulky

| | | | | | | | | | | | | | | | | | |
|--------------|----|----|----|---|---|---|---|---|---|-----|---|---|---|---|---|---|---|
| 31 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | |
| Adresa rámce | | | | | | | | | | AVL | 0 | 0 | D | A | P | P | |
| | | | | | | | | | | | | | C | W | U | W | P |
| | | | | | | | | | | | | | D | T | | | |

| U | W | WP | Proces CPL=3 | Proces CPL<3 |
|---|---|----|-------------------------|-------------------------|
| 0 | 0 | 0 | nepřístupná | čtení, zápis, provedení |
| 0 | 1 | 0 | nepřístupná | čtení, zápis, provedení |
| 1 | 0 | 0 | čtení, provedení | čtení, zápis, provedení |
| 1 | 1 | 0 | čtení, zápis, provedení | čtení, zápis, provedení |
| 0 | 0 | 1 | nepřístupná | čtení, provedení |
| 0 | 1 | 1 | nepřístupná | čtení, zápis, provedení |
| 1 | 0 | 1 | čtení, provedení | čtení, provedení |
| 1 | 1 | 1 | čtení, zápis, provedení | čtení, zápis, provedení |

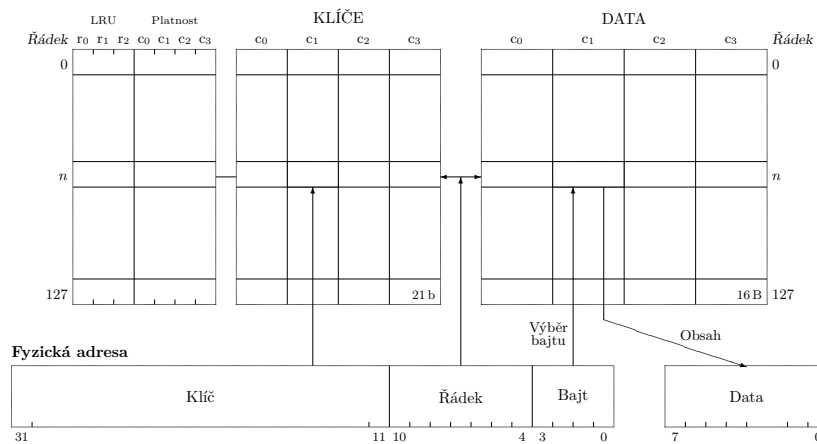
PWT (Page Write-Through) určuje způsob práce externí vyrovnávací paměti. Je-li PWT=1, provádí se zápis metodou Write-Through. Je-li PWT=0, provádí se zápis metodou Write-Back.

PCD (Page Cache Disable) vypíná činnost interní VP. Je-li PCD=0, je splněna jedna z podmínek zapínajících IVP. Další podmínky tvoří signál \overline{KEN} a bity CD a NW v registru CR0. Je-li PCD=1, je IVP vypnuta bez ohledu na ostatní podmínky.

Je-li stránkování zapnuto (PG=1) a je právě plněn stránkový adresář, čtou se bity PWT a PCD z CR3. Bity PWT a PCD konkrétního specifikátoru ze stránkového adresáře se čtou při plnění stránkové tabulky.

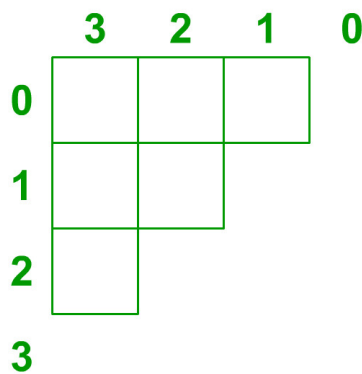
13.6 Interní vyrovnávací paměť

Interní vyrovnávací paměť (IVP)



Potřebný počet bitů na realizaci LRU

Kolik bitů potřebujeme na výběr nejdéle nepoužité položky ze čtyř položek algoritmem LRU?



13.7 Pseudo-LRU

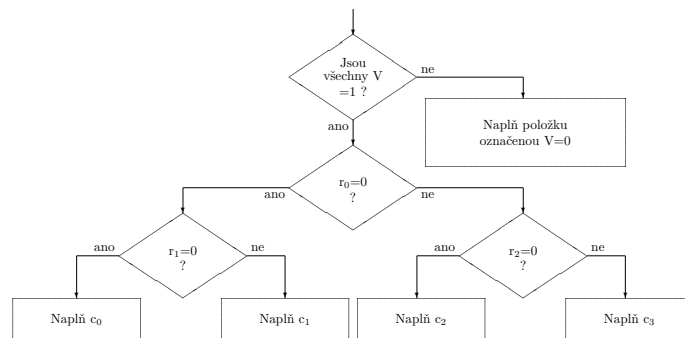
Pseudo-LRU IVP i486

Nastavování rozhodujících bitů

| Při použití položky z cesty | se nastaví rozhodovací bity | | |
|-----------------------------|-----------------------------|------------|------------|
| | r_0 | r_1 | r_2 |
| c_0 | 1 | 1 | beze změny |
| c_1 | 1 | 0 | beze změny |
| c_2 | 0 | beze změny | 1 |
| c_3 | 0 | beze změny | 0 |

Výběr nejdéle nepoužité položky

Při plnění položkou, která nemá svůj obraz v IVP, se nejprve podle bitů 4 až 10 fyzické adresy vybere řádek IVP. Potom se postupuje podle algoritmu:



14 Architektura procesorů Intel – Procesor Pentium

14.1 Rysy procesoru

Procesor Intel Pentium

- Pentium z řecky penta, tj. 5,
- 32bitový procesor,
- od 1993 do 1999,
- 60 MHz až 300 MHz,
- od 1995: Pentium MMX, Pro, II, III, 4, D, Xeon,
- postupně se liší parametry: technologie např. 0,25 μm , velikostí cache, počtem jader, ...

Pentium

V procesoru Pentium jsou integrovány všechny vlastnosti procesoru Intel486. Navíc poskytuje tato významná rozšíření:

- superskalární architekturu,
- dynamické předvídání skoků,
- zřetěženou FPU,
- zkrácení doby provádění instrukcí,
- oddělené 8KB datové a instrukční vnitřní vyrovnávací paměti,
- protokol MESI pro řízení datové vyrovnávací paměti,
- 64bitovou datovou sběrnici,
- zřetězování cyklů sběrnice,

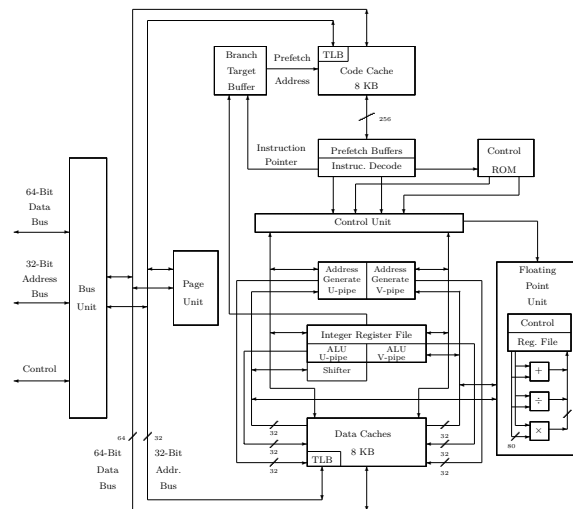
Pentium – další rysy

- adresové parity,
- vnitřní kontrolu parity,
- kontrolu správné funkce znásobením čipů s procesorem,
- sledování provádění,
- monitorování výkonnosti,
- ladění prostřednictvím IEEE 1149.1 Boundary Scan,
- režim správy systému a
- rozšíření v režimu V86.

Instrukční repertoár Pentia je plně kompatibilní s Intel486. Plně kompatibilní je také i správa paměti (MMU).

14.2 Blokový diagram

Blokový diagram procesoru Pentium



14.3 Zřetězené provádění instrukcí

Zřetězené provádění instrukcí

| | | |
|----|--------------------|----------------------|
| PF | Prefetch | Výběr instrukce |
| D1 | Instruction Decode | Dekódování instrukce |
| D2 | Address Generate | Generování adresy |
| EX | Execute | Provedení instrukce |
| WB | Write Back | Dokončení instrukce |

| | | | | | | | | |
|-----------|----|----|----|----|----|----|----|----|
| PF | I1 | I3 | I5 | I7 | | | | |
| | I2 | I4 | I6 | I8 | | | | |
| D1 | | I1 | I3 | I5 | I7 | | | |
| | | I2 | I4 | I6 | I8 | | | |
| D2 | | | I1 | I3 | I5 | I7 | | |
| | | | I2 | I4 | I6 | I8 | | |
| EX | | | | I1 | I3 | I5 | I7 | |
| | | | | I2 | I4 | I6 | I8 | |
| WB | | | | | I1 | I3 | I5 | I7 |
| | | | | | I2 | I4 | I6 | I8 |

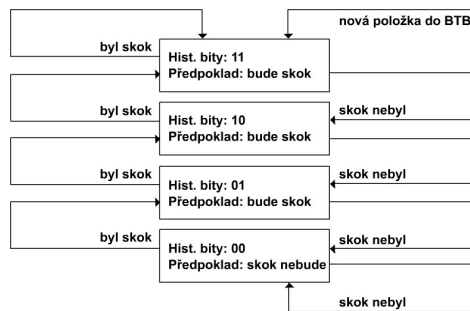
- Tyto dvě zřetězené fronty se nazývají „u“ a „v“.
- Proces souběžného zpracovávání instrukcí se nazývá „párování“.
- Ve zřetězené frontě „u“ lze provádět libovolnou instrukci, zatímco ve frontě „v“ lze provádět pouze jednoduché instrukce, popsané v pravidlech pro párování instrukcí.

14.4 Předvídání podmíněných skoků

Předvídání podmíněných skoků

Branch Target Buffer - BTB

Při výběru instrukce se testuje obsah BTB na shodu s adresou vybírané instrukce. Pokud se adresa v BTB najde, zkoumá se obsah bitů historie.



14.5 Párování instrukcí

Párování instrukcí

Instrukce mohou být spojeny do páru za splnění následujících podmínek.

- Obě instrukce v páru musí být „jednoduché“ podle dále uvedené definice.
- Mezi instrukcemi v páru nesmí být vztah „čtení až po zápisu“ nebo „zápis až po čtení“.
- Žádná z instrukcí nesmí mít výpočet adresy složený ze dvou částí: z přímé hodnoty a zároveň z přírůstku.
- Instrukce s prefixy (vyjma 0F před podmíněným skokem) lze provádět pouze ve frontě „u“.

Jednoduché instrukce

Jednoduché instrukce jsou ty, které nevyžadují mikro kód a provedou se během jednoho hodinového cyklu. Výjimkou jsou instrukce aritmeticko-logické jednotky (ALU) *mem,reg* a *reg,mem*, které se provádějí ve dvou nebo třech taktech a jsou považovány za jednoduché.

Za jednoduché se považují tyto instrukce určené pro celočíselné zpracování:

1. *mov reg, reg/mem/imm*
2. *mov mem, reg/imm*
3. *alu reg, reg/mem/imm*
4. *alu mem, reg/imm*
5. *inc reg/mem*
6. *dec reg/mem*
7. *push reg/mem*
8. *pop reg*
9. *lea reg,mem*
10. *jmp/call/jcond near*
11. *nop*

Podmíněné a nepodmíněné skoky smějí být párovány pouze jako druhé instrukce v páru.

14.6 Režim správy systému

Režim správy systému

System Management Mode – SMM – Režim SMM je transparentní (neviditelný) pro aplikace i operační systém z těchto důvodů:

- Jedinou možností, jak SMM zapnout, je externí nemaskovatelné přerušení přivedené speciálním signálem.
- Procesor zahájí provádění instrukcí určených pro SMM ze separátního adresového prostoru a separátní paměti (tzv. SMRAM – System Management RAM).

Režim správy systému - pokračování

- Při přepínání do SMM procesor ukládá obsah všech registrů do zvláštní části SMRAM.
- Všechna přerušení, která normálně operační systém či aplikace obsluhuje, jsou během SMM zakázána.
- Stav před přepnutím do režimu SMM se vrátí provedením instrukce RSM.

SMM je podobný reálnému režimu. Nejsou v něm úrovně oprávnění, privilegované instrukce nebo mapování adres. V SMM lze provádět V/V operace a adresovat celou 4GB kapacitu fyzické operační paměti.

15 Architektura x86-64

15.1 Principy architektury x86-64

x86-64 architektura

- *x86-64* tzv. *AMD64*
 - plnohodnotně 64bitová architektura
 - firma AMD (Advanced Micro Devices) od roku 1999
 - první procesor AMD Opteron 2003
 - zpětně kompatibilní s x86
 - procesory Opteron, Athlon, Turion, Sempron

Dvě architektury

- *IA-64*
 - firmy Intel a Hewlett-Packard
 - kompatibilní s x86
 - jiná instrukční sada v 64bitovém režimu: EPIC (Explicitly Parallel Instruction Computing)
 - možnost vkládat instrukce paralelně úkolující více jednotek; VLIW (Very Long Instruction Word) – tzv. instrukční paket, součástí instrukce jsou samostatné pokyny všem jednotkám procesoru
 - procesory Itanium

Obě architektury nejsou na 64bitové úrovni kompatibilní.

EM64T Extended Memory 64 Technology, tzv. Intel 4

- firma Intel převzala architekturu AMD64 (též pod označením Intel 4)
- procesory Pentium 4, Celeron D, Xeon, Core 2
- jsou drobné rozdíly mezi AMD64 a Intel 4

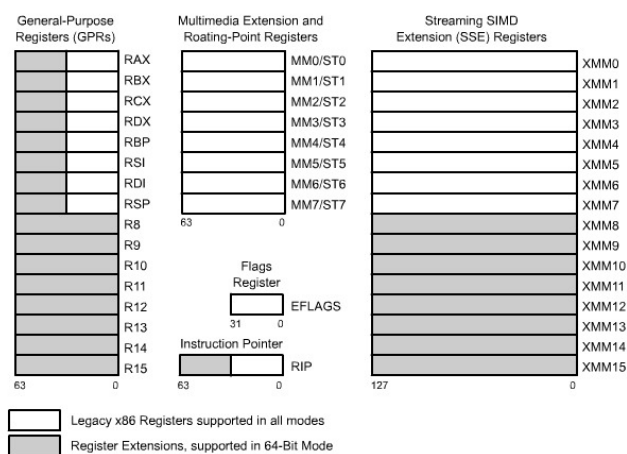
x86-64 architektura

long mode • **64bitový režim (Long Mode)** – plně 64bitový režim

- **kompatibilní režim** – 32/16bitový režim, omezená kompatibilita s x86 (pouze chráněný režim, žádný reálný, žádný V86 režim)

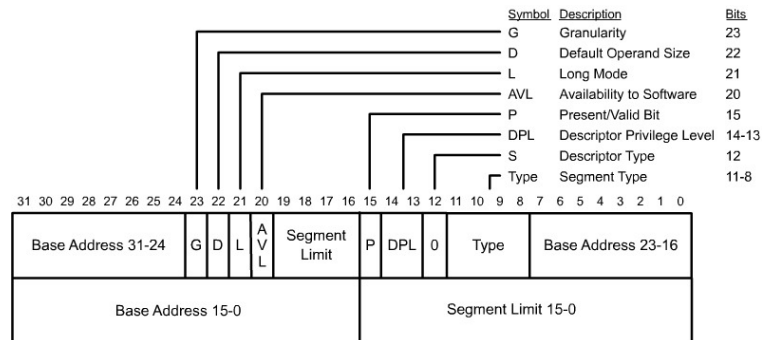
x86 mode plná kompatibilita s x86 32/16bitovým režimem (vč. reálného režimu, ...)

Registrová struktura



64bitový režim (Long mode)

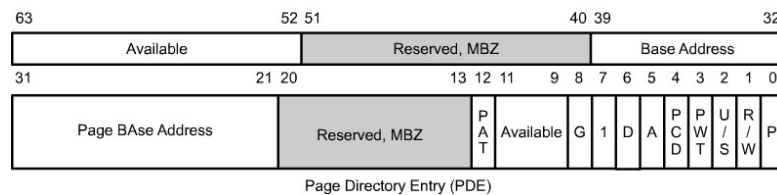
- 64bitová virtuální adresa, 52bitová fyzická adresa (4 petabyte)
- šířka adresy implicitně 64 bitů
- virtuální adresa je pouze 64bitový offset
- blízké skoky rozšířeny na 64 bitů
- flat 64bit virtuální adresový prostor
- potlačen význam segmentace jen na určování typů a práv segmentů
- místo segmentace zaveden non-executable bit u každé stránky
- GDTR a IDTR mají 64bitovou bázi a 16bitový limit
- LDTR a TR 64bitovou bázi, 32bitový limit a navíc 16bitový selektor
- navíc bit L (Long) v popisovači instrukčního segmentu:



- ignoruje se báze a limit v popisovači
- báze je vždy 0
- vždy 64bitová adresa, proto L=1 a D=1 je vyhrazeno pro budoucí použití
- obsah ES, DS a SS se ignoruje

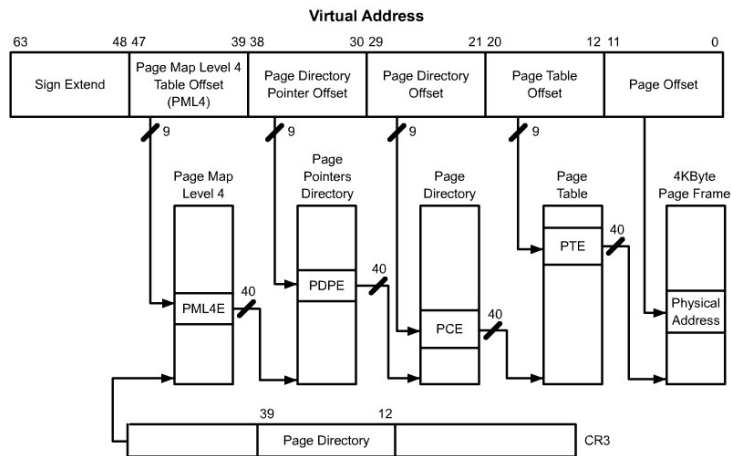
Stránkování

- podporují se 4KB nebo 2MB stránky
- velikost stránky určuje bit 7 stránkového adresáře
- položky stránkových tabulek jsou 64bitové
- formát položky stránkového adresáře pro 2MB stránky:

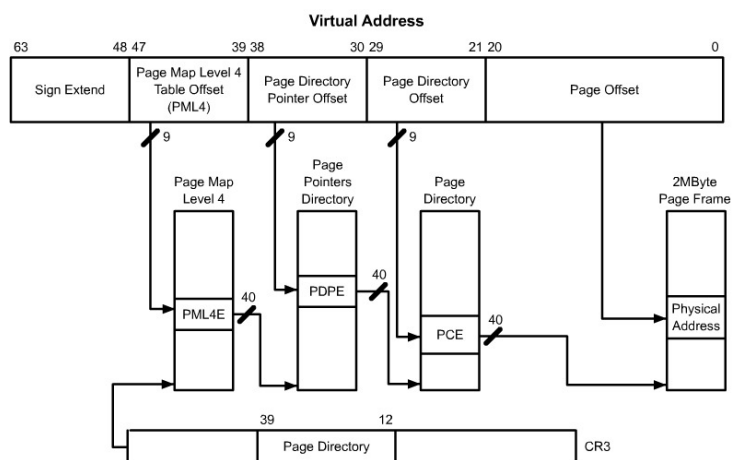


MBZ = Must be zero

4KB stránky:



2MB stránky:



V long módu i 16GB stránky.

16 RISC

16.1 Principy architektury i860

Mikroprocesor i860

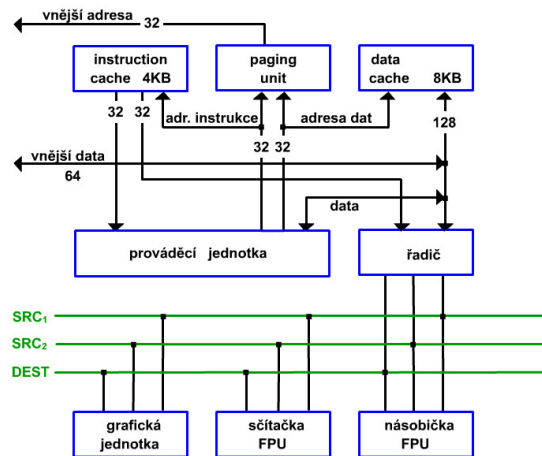
- 8086 - i486 = CISC - Complex Instruction Set Computer
- i860 = RISC - Reduced Instruction Set Computer

1989

Není kompatibilní na řadu x86

Výkonem odpovídá počítači Cray I. „Cray on a Chip“
64bitový procesor

Mikroprocesor i860



16.2 Jednotky a principy i860

Jednotky i860

Prováděcí jednotka:

- Registry $r_0 - r_{31}$ 32bitové
- r_0 je vždy = 0
- operace zápisu se ignoruje
- pro 64bitové operace se sdružují do dvojic

Jednotky i860

Techniky:

- *Registr Bypassing*
Je-li výsledek předchozí operace vstupem do další bere se ze sběrnice
- *Delayed Branch*
Před skokem se provede ještě následující instrukce
BR návěští
OR r_0, r_0, r_0
- 2 varianty podmíněného skoku:



Jednotky i860

FPU:

- Registry $f_0 - f_{31}$ 32bitové
- f_0, f_1 vždy = 0
- *Sčítačka* – sčítání a převody mezi jednoduchou a dvojnásobnou přesností
- *Násobička* – násobení a výpočet $1/x$
- *Duální instrukční mód* – jedna instrukce vyvolá dvě paralelní akce: jednu v násobičce a jednu ve sčítačce
- *Využití*: vyčíslování řad, FFT, ...

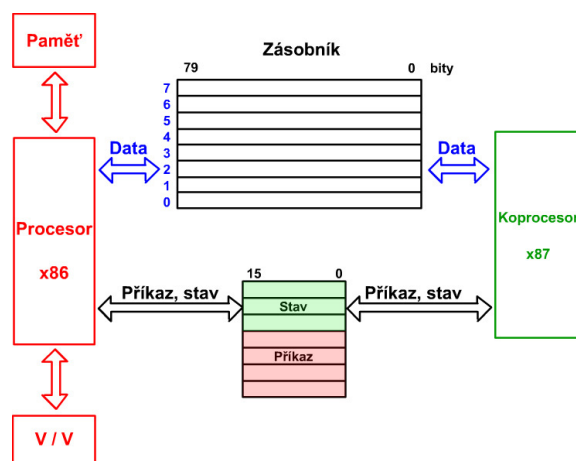
Jednotky i860

- **Stránkovací jednotka:** – Stejná jako v 80386
- **Grafická jednotka:** – Pro 3D grafiku z-Buffer (uložení souřadnice třetího rozměru)
- **Využití i860:**
 - grafické a unixové stanice
 - jako speciální grafický koprocesor (grafika v reálném čase)

17 IEEE 754

17.1 Matematický koprocesor 80x87

Matematický koprocesor Intel 80x87 k procesoru 80x86



17.2 Typy dat

Typy dat pro koprocesor 80x87

| Typ | Poč. bitů | Poč. des. číslic mant. | Rozsah zobr. v des. soustavě | | | | | | |
|-----------------|-----------|------------------------|--|---|-----------|------------------|----|----|---|
| INTEGER: | | | | | | | | | |
| WORD | 16 | 4-5 | $-32768 \leq x \leq +32767$ | | | | | | |
| SHORT | 32 | 9-10 | cca $-2.10^9 \leq x \leq +2.10^9$ | | | | | | |
| LONG | 64 | 18-19 | cca $-9.10^{18} \leq x \leq +9.10^{18}$ | | | | | | |
| BCD: | | | | | | | | | |
| PACKED | | | | | | | | | |
| DECIMAL | 73 | 18 | <table border="1"> <tr> <td>±</td> <td>nevyužito</td> <td>18 desít. číslic</td> </tr> <tr> <td>79</td> <td>71</td> <td>0</td> </tr> </table> | ± | nevyužito | 18 desít. číslic | 79 | 71 | 0 |
| ± | nevyužito | 18 desít. číslic | | | | | | | |
| 79 | 71 | 0 | | | | | | | |

Typy dat pro koprocesor 80x87

| Typ | Poč. bitů | Poč. des. číslic mant. | Rozsah zobr. v des. soustavě | | | | | | |
|--------------|---------------------|------------------------|---|-------|---------------------|-----------------|----|----|---------|
| REAL: | | | | | | | | | |
| SHORT | 32 | 6-7 | <table border="1"> <tr> <td>zn.m.</td> <td>exp. vč. zn. (8 b)</td> <td>mantisa bez zn.</td> </tr> <tr> <td>31</td> <td>30</td> <td>23 22 0</td> </tr> </table> $1.2 \times 10^{-38} \leq x \leq 3.4 \times 10^{38}$ | zn.m. | exp. vč. zn. (8 b) | mantisa bez zn. | 31 | 30 | 23 22 0 |
| zn.m. | exp. vč. zn. (8 b) | mantisa bez zn. | | | | | | | |
| 31 | 30 | 23 22 0 | | | | | | | |
| LONG | 64 | 16-17 | <table border="1"> <tr> <td>zn.m.</td> <td>exp. vč. zn. (11 b)</td> <td>mantisa bez zn.</td> </tr> <tr> <td>63</td> <td>62</td> <td>52 51 0</td> </tr> </table> $cca 10^{-308} \leq x \leq 10^{308}$ | zn.m. | exp. vč. zn. (11 b) | mantisa bez zn. | 63 | 62 | 52 51 0 |
| zn.m. | exp. vč. zn. (11 b) | mantisa bez zn. | | | | | | | |
| 63 | 62 | 52 51 0 | | | | | | | |

Typy dat pro koprocessor 80x87

| Typ | Poč. bitů | Poč. des. číslic mant. | Rozsah zobr. v des. soustavě | | |
|--|-----------|------------------------|------------------------------|---------------------------|----------------------------|
| REAL: | | | | | |
| TEMPORARY | 80 | 19-20 | zn.m. 79 | exp. vč. zn. (15 b) 78 | mantisa bez zn. 64 63 0 |
| cca $10^{-4932} \leq x \leq 10^{4932}$ | | | | | |

Reálná čísla jsou vždy automaticky transformována na typ temporary real, ve kterém se provádějí všechny výpočty.

V normě IEEE 754 existuje i 128bitová reprezentace s 15b exponentem a 112b mantisou, tj. zvýšená přesnost.

17.3 Zobrazení reálného čísla (1)

Formát čísel I.

FORMÁT:

- *INTEGER: WORD, SHORT, LONG ...*
... Dvojkový doplňkový kód
- *REAL: TEMPORARY (80bitový) ...*
... IEEE 754 (Institute of Electrical and Electronics Engineers)



$$\pm \text{Mantisa} \times 2^{\text{Exponent}}$$

S znaménko čísla, 0=kladné, 1=záporné

Mantisa v Přímém kódu (znaménko je v S)

Normalizovaný tvar Mantisy:

- První významná binární číslice je v nejvyšším bitu



↑
Binární číslice nejvyššího řádu

- Nejvyšší bit je vždy = 1 (vyjma případu číslo = 0)
- Nejvyšší binární číslici vynecháváme
- **Mantisu** vyjádříme ve tvaru:



Exponent číslem 2^{Exponent} vynásobíme Mantisu ve tvaru 1.xxxxxxx, abychom dostali zobrazované číslo.

17.4 Zobrazení reálného čísla (2), zvláštní čísla

Formát čísel II.

Exponent ... v kódu posunutě nuly:

| | | |
|------------------|---|---------------------|
| 000 ... 000 | - | max |
| 011 ... 111 | | 0 |
| 100 ... 000 | | +1 |
| <u>111...111</u> | | + max + 1 |
| n | | max = $2^{n-1} - 1$ |

K zapisovanému číslu přičítáme $2^{n-1} - 1$, tj. pro n = 8 přičítáme 127₁₀, tzn. 01111111₂.

17.5 Převod čísel do a z IEEE 754

- *Příklad 1:*

12.5_{10}

$$12.5_{10} = 1100.1_2 = 1.1001_2 \times 2^3$$

| | | |
|---|----------|----------------|
| 0 | 10000010 | 1001000 ... 00 |
|---|----------|----------------|

Exponent: $01111111 + 00000011 = 10000010$

- *Příklad 2:*

-0.3125_{10}

$$-0.3125_{10} = -0.0101_2 = -1.01_2 \times 2^{-2}$$

| | | |
|---|----------|---------------|
| 1 | 01111101 | 01000 0 |
|---|----------|---------------|

Exponent: $01111111 - 00000010 = 01111101$

- *Příklad 3:*

1.0

| | | |
|---|----------|----------------|
| 0 | 01111111 | 000000 0 |
|---|----------|----------------|

Exponent: $01111111 + 00000000 = 01111111$

Zvláštní čísla podle IEEE 754

| | | |
|----------|----------|----------------|
| θ | | |
| 0 | 00000000 | 000000 0 |
| 1 | 00000000 | 000000 0 |

FPU běžně produkuje kladnou nulu:

$$+ \theta = +1.000 \dots \times 2^{-127}, \text{ je-li počet bit; exponentu} = 8$$

$$- \theta = -1.000 \dots \times 2^{-127}$$

17.6 Zvláštní čísla (2)

Formát čísel III.

| | | | |
|------------------------------|---|----------|----------------|
| ∞ Kladné nekonečno | 0 | 11111111 | 000000 0 |
| Záporné nekonečno | 1 | 11111111 | 000000 0 |

$$+\infty = +1.0 \times 2^{128}, \text{ je-li počet bitů exponentu} = 8$$

$$-\infty = -1.0 \times 2^{128}$$

Nenormalizované číslo

- Při nutnosti zobrazit menší číslo v absolutní hodnotě než je $1.0 \times 2^{-2^{n-1}+1}$
- *O číslu musí být známo, že je nenormalizované !*
- Ke každému registru uschovávacímu číslo IEEE je 2bitový příznak

| | |
|-------------------|----------------------------------|
| 11 ... registr | je prázdný |
| 01 ... registr | = 0 |
| 10 ... registr | = ∞ (exponent = 11111111) |
| | = nenormalizované číslo |
| | (exponent = 00000000) |
| | = atd. |
| 00 ... „normální“ | obsah registru |

17.7 Určení rozsahu, přesnosti a rozlišitelnosti

Formát čísel IV.

- **Rozsah zobrazení:**
 $(-1.0 \times 2^{2^{n-1}}; +1.0 \times 2^{2^{n-1}})$ n ... počet bitů exponentu
Pro účely určování rozsahu zobrazení předpony Mantisy = 1.0
bitů na exponentu = 8 ... $(-2^{128}; 2^{128})$
- **Přesnost zobrazení:**
 - na kolik bitů lze zobrazit Mantisu
 - počet bitů Mantisy + 1 = $m + 1$

Formát čísel IV.

- **Rozlišitelnost:**
= nejmenší kladné nenulové zobrazitelné číslo
normalizované: $+1. \underbrace{00\dots000}_m \times 2^{-2^{n-1}+1}$
nenormalizované: $+0. \underbrace{00\dots001}_m \times 2^{-2^{n-1}+1}$
 $= 2^{-2^{n-1}+1-m}$

17.8 Typy operací koprocessoru

Typy operací koprocessoru 8087

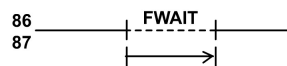
- **Přenos dat paměť86 ↔ zásobník87**
 - reálná čísla
 - celá čísla
 - BCD čísla
- **Aritmetické operace**
 $+$, $-$, $*$, $/$, \div , $\sqrt{\quad}$, mod , $round$, int , abs , \pm
- **Porovnávací operace**

Typy operací koprocessoru 8087

- **Výpočet transcendentních funkcí**
 - exponenciální funkce
 - logaritmické funkce
 - goniometrické funkce
 - cyklometrické funkce = (inverzní goniometrické fce.)
 - hyperbolické funkce
- **Plnění konstantami** $+0.0$, $+1.0$, $\log_2(10)$, $\log_2(e)$, $\log_{10}(2)$, $\log_e(2)$
- **Instrukce pro řízení 8087:**
např. FWAIT

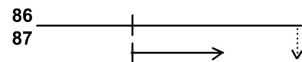
Dva typy činnosti:

1.



- naplní zásobník87
- zahaj operaci87
- FWAIT
- čti stav87
- čti zásobník87

2.



- naplní zásobník87
- zahaj operaci87
- jiná činnost bez 87
- čti stav87
- je-li hotovo, čti zásobník87

18 Připojování V/V zařízení

18.1 Centronics

Rozhraní Centronics (EPSON)

Paralelní rozhraní určené pro výstup informace

1. *Mechanická úroveň:* Konektor Cannon 25kolíkový, na počítači je zásuvka.



2. *Elektronická úroveň:*

- „0“ ... 0V až 0.4 V - úroveň TTL
- „1“ ... 2.4V až 5V - úroveň TTL

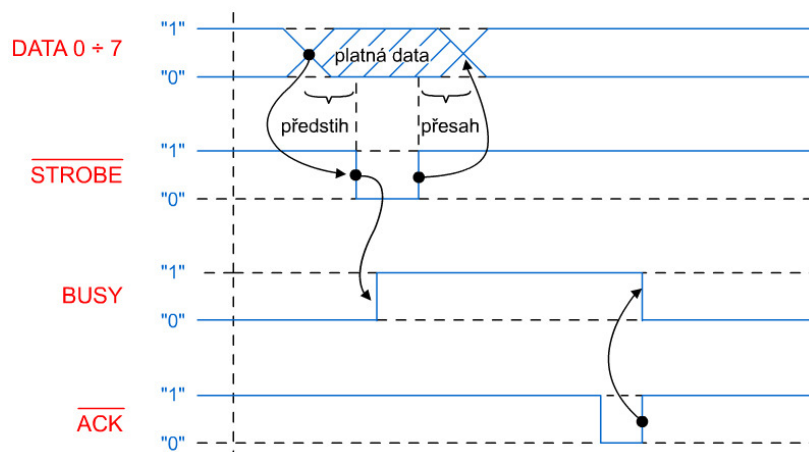
Rozhraní Centronics (EPSON)

Zapojení:

| Špička | Signál | Zdroj | Význam |
|--------|---------------------|-------|-----------------------|
| 1 | \overline{STROBE} | Poč. | platnost dat |
| 2 | DATA 0 | Poč. | DATA |
| . | . | . | DATA |
| . | . | . | DATA |
| . | . | . | DATA |
| 9 | DATA 7 | Poč. | DATA |
| 10 | \overline{ACK} | Tisk. | konec tisku znaků |
| 11 | BUSY | Tisk. | tiskárna obsazena |
| 12 | PE | Tisk. | paper empty |
| 13 | SLCT | Tisk. | připravenost tisku |
| 14 | AUTO | poč. | automat. LF po CR |
| 15 | \overline{ERROR} | Tisk. | chyba tiskárny |
| 16 | \overline{INIT} | Poč. | inicializace tiskárny |
| 17 | SLCT IN | Poč. | žádost o přípravu |
| 18-25 | GND | - | zem |

Rozhraní Centronics (EPSON)

3. Logická úroveň

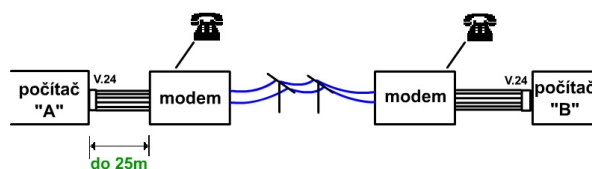


18.2 RS-232-C (V.24), zapojení, signály

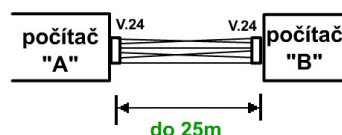
Rozhraní V.24 I. = RS-232C

Připojení:

- a)



- b)



V.24 je rozhraní přispůsobené pro telefonní linky:

1. *Mechanická úroveň:* Konektor Cannon 25 nebo 9kolíkový, na počítači je zástrčka.
2. *Elektronická úroveň:*

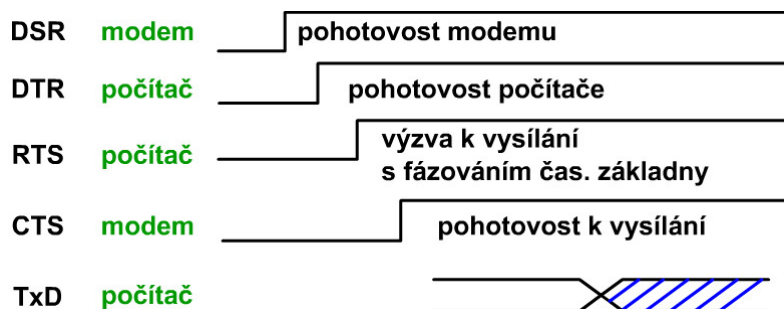
- „1“ ... -15V ÷ -3V
- „0“ ... 3V ÷ 15V

Zapojení:

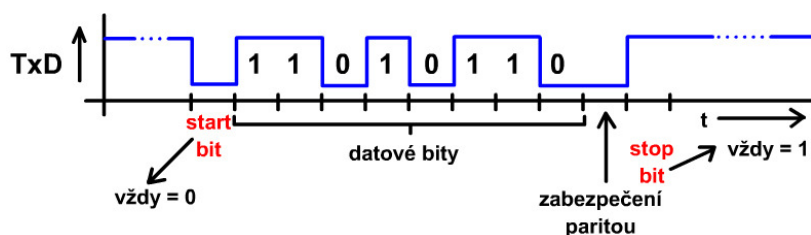
| Špička | Číslo obvodu | Označení | Zdroj | Význam |
|--------|--------------|--------------|---------|-----------------------|
| 1 | 101 | PG | | ochr. zem |
| 2 | 103 | $T \times D$ | počítač | vysílaná data |
| 3 | 104 | $R \times D$ | modem | přijímaná data |
| 4 | 105 | RTS | počítač | výzva k vysílání |
| 5 | 106 | CTR | modem | pohotovost k vysílání |
| 6 | 107 | DSR | modem | pohotovost modemu |
| 7 | 102 | SG | | signálová zem |
| 8 | 109 | DCD | modem | detekce nosné |
| 20 | 108 | DTR | počítač | pohotovost počítače |
| 22 | 125 | RI | modem | zvonek |
| . | . | . | . | . |
| . | . | . | . | . |
| . | . | . | . | . |

18.3 RS-232-C (V.24), průběhy, formáty

3. Logická úroveň



Formát přenosu dat:



Parametry přenosu dat:

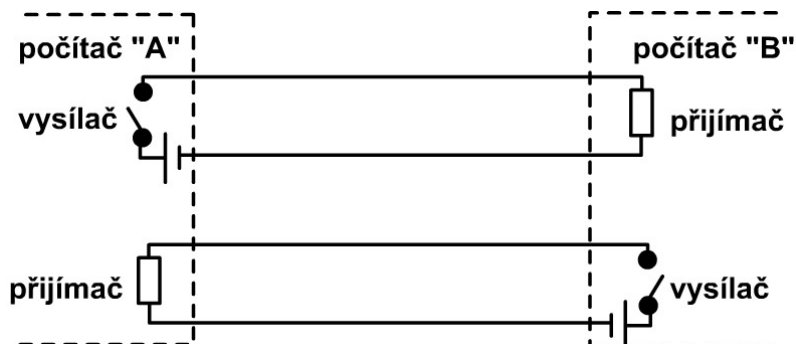
- rychlost: (v bitech za sekundu) 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200
- počet datových bitů: 5, 6, 7, 8
- zabezpečení: sudá parita (Even), lichá parita (Odd), žádné
- délka stop bitu: 1, 1.5, 2

18.4 Proudová smyčka

Rozhraní IRPS (proudová smyčka)

- název převzatý z dálkopisné sítě
- až do 2 km
- proud 20, 40 mA
- chybí přesná definice

Zapojení:

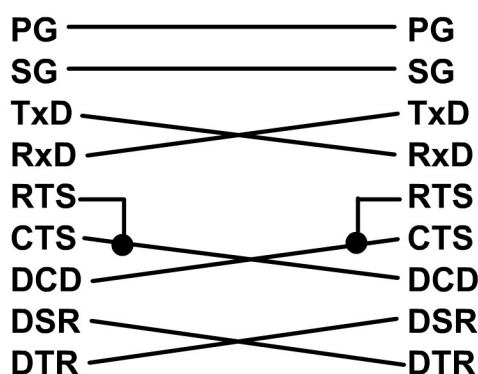


+ optické oddělení komunikujících počítačů

18.5 Nullmodem

Nullmodem

Nullmodem: propojení dvou počítačů bez modemu



18.6 USB Universal Serial Bus

USB Universal Serial Bus

Idea:

- Všechna typická zařízení se stejně připojují na společnou sběrnici.
- Náhrada připojení klávesnice, myši, RS232 zařízení, Centronics.
- Snadnost použití.
- Možnost připojování/odpojování bez vypnutí.
- Současně napájí a současně přenáší data.

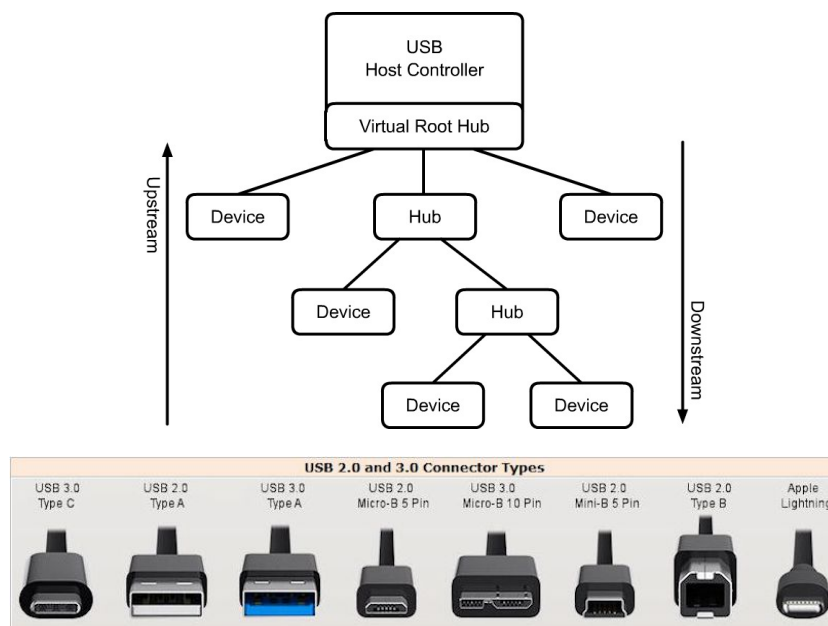
Master/Slave protokol Komunikace je řízena/vyvolávána počítačem (host)

Maximálně 127 zařízení

Ochrana proti zkratu a přepětí Dovoluje se připojení/odpojení zařízení bez vypnutí počítače. Nutnost ochrany proti elstat. výboji - člověk až 15 kV na koberci.

Napájení z USB Maximální proud je omezován dle typu připojeného zařízení. Při překročení proudu je port odpojen.

Počítač se opakovaně dotazuje rozbočovačů na jejich stav a nová zařízení.



USB 1.x (1996) Rychlost 1,5 Mbit/s (Low-Speed) nebo 12 Mbit/s (Full-Speed), 4 vodiče.

USB 2.0 (1999) Rychlost 480 Mbit/s (Hi-Speed).

USB 3.1 (2010) Rychlost 5 Gbit/s, 9 vodičů.

USB 3.1 generace 2 (2013) Rychlost 10 Gbit/s, konektor USB Type-C (oboustranný, 12 kontaktů na každé straně).

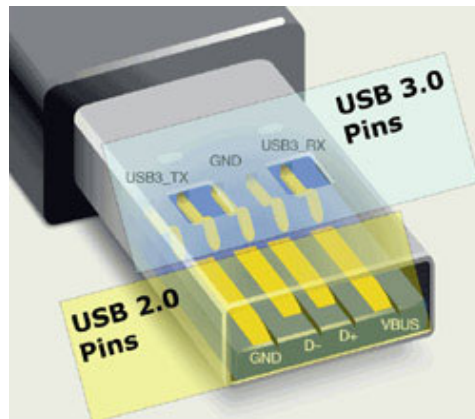
Čtyřdrátová sběrnice:

+5V

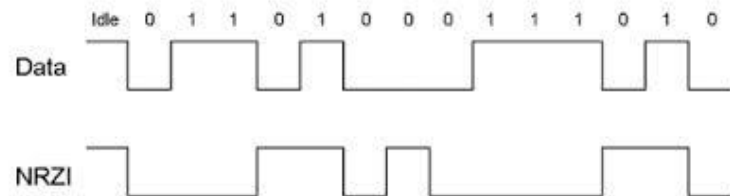
data -

data +

zem



- Kódování **NRZI** (Non return to Zero Invert, bez návratu k nule s inverzí).
- 0 invertuje hodnotu signálu, 1 ponechává beze změny.
- Není závislé na polaritě. Vhodné pro přenos stejného signálu dvěma skroucenými vodiči s opačnou



polaritou (tzv. diferenciální pár). Odolává rušení.

- Synchronizuje se úvodním bajtem 00000001.
- Po každé šesté 1 v řadě se automaticky vkládá 0.