Počítač

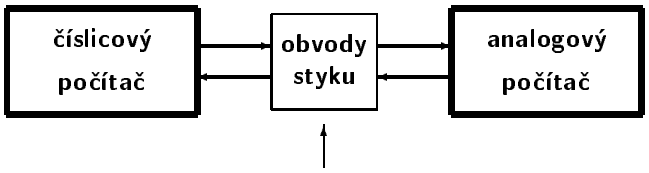
vstupní data (údaje) $\longrightarrow$ **výstupní data (údaje)**

zobrazení dat:

1. **spojité = analogové**
2. **nespojité = diskrétní = číslicové = digitální**

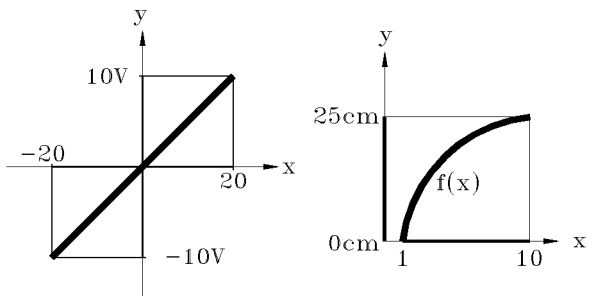
počítač:

1. **analogový** ... spojité (analogové) zobrazení
2. **číslicový** ... nespojité (číslicové) zobrazení
3. **hybridní:**

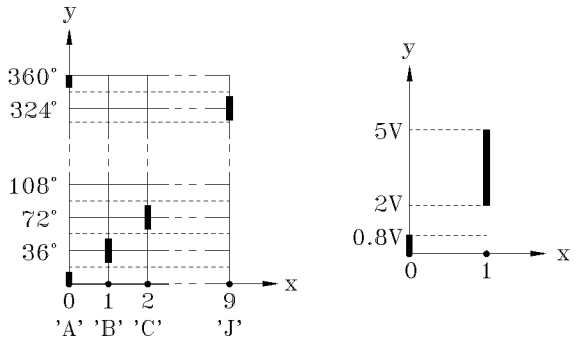


převodníky: Č/A = D/A
A/Č = A/D

zobrazení spojité



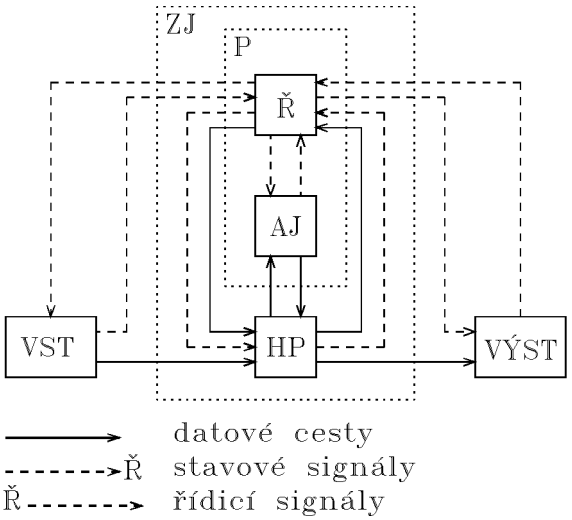
zobrazení nespojité



von Neumannova koncepce

- Instrukce a data jsou uloženy v téže paměti.
- Paměť je organizována lineárně (tzn. jednorozměrně) a je rozdělena na stejně velké buňky, které se adresují celými čísly (zprav. 0, 1, 2, 3, ...).
- Data ani instrukce nejsou explicitně označeny. Explicitně nejsou označeny ani různé datové typy.
- Pro reprezentaci dat i instrukcí se používají dvojkové signály.
- V instrukci zpravidla není uváděna hodnota operandu, ale jeho adresa.
- Instrukce se provádějí jednotlivě, a to v pořadí, v němž jsou zapsány v paměti, pokud není toto pořadí změněno speciálními instrukcemi (nazývanými skoky).
- Počítač tvoří: paměť,
  řadič,
  aritmetická jednotka,
  vstupní a výstupní jednotky.

Počítač von Neumannova typu



Ř ... řadič
AJ ... aritm. jedn.
VST ... vst. zař.
HP ... hlavní pam.
VÝST ... výst. zař.

P ... procesor
ZJ ... zákl. jedn.

Ř – Řadič **[CU – Control Unit]**

AJ – Aritmetická Jednotka **[ALU – Arithmetic and Logic Unit]**
neboli
Aritmeticko–logická Jednotka

HP – Hlavní Paměť **[MM – Main Memory]**
(operační paměť)

P – Procesor **[Processor]**

$$\mathbf{P} \stackrel{?}{=} \begin{cases} \mathbf{Ř + AJ} \\ \mathbf{AJ} \\ \mathbf{Ř + AJ + HP} \end{cases}$$

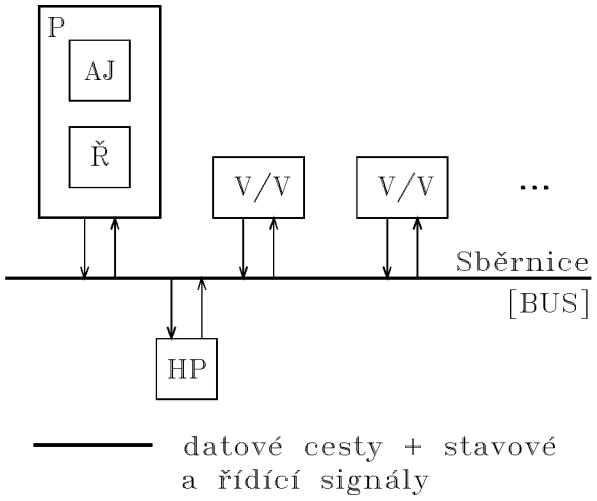
ZJ – Základní Jednotka **[CPU – Central Processing Unit]**

$$\mathbf{ZJ} \stackrel{?}{=} \begin{cases} \mathbf{P + HP} \\ \mathbf{P} \end{cases}$$

VST – VSTupní zařízení **[I – Input devices]**

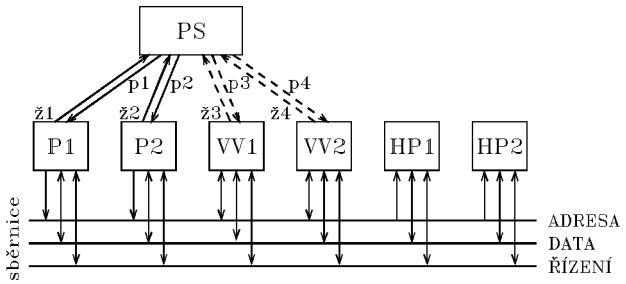
VÝST – VÝSTupní zařízení **[O – Output devices]**

Propojení jednotek přes sběrnice



V/V — Vstupní a/nebo výstupní zařízení neboli
periferní zařízení, popř.
přídavná zařízení

Přidělování sběrnic jednotkám



PS – přidělovač sběrnice **[bus arbiter]**
P1, P2 – procesory
VV1, VV2 – řídicí jednotky periferních zařízení
HP1, HP2 – moduly hlavní paměti
ž1, . . . , ž4 – žádosti o přidělení sběrnice **[requests]**
p1, . . . , p4 – přiděleno (potvrzení) **[grants]**

Funkci přidělovače může zastávat např. procesor (zvl. u jednoprocesorových počítačů)

Hlavní (operační) paměť

obsahuje:

- **Instrukce = příkazy k provedení *základních operací* počítače**
  soubor instrukcí: *program* — předpis (v kódované formě), jak se mají transformovat vstupní data na data výstupní
- **Data**
  - **numerická — *čísla***
  - **nenumerická:**
    - ***texty* — posloupnosti písmen, číslic a jiných znaků** (char, string apod.)
    - **pravdivostní hodnoty** — např. true a false (Boolean, logical apod.)
    - **obrazové informace, jejichž prvky** (PIXELs — picture elements) **obsahují barvu, jas apod.**
    - **aj.**

Všechny instrukce a všechny typy dat jsou zakódovány dvojkově — jako čísla !!!

Numerická data — *čísla*:

Čísla:

- **v pevné řádové čárce** **[fix point]**
  — obv. **celá čísla** (integer, byte, word, int apod.)
- **v pohyblivé řádové čárce** **[floating point]**
  — **reálná čísla** (přesněji: některá racionální)
  (real, float apod.)

Čísla:

- **dvojková** **[binary]**
- **desítková** **[decimal]**

Čísla:

- **bez znaménka**, tzn. pouze nezáporná
  (byte, word, unsigned apod.) **[unsigned]**
- **se znaménkem**, tzn. záporná i nezáporná
  (integer, shortint, signed apod.) **[signed]**

Čísla:

různě "dlouhá" — přesněji: různý rozsah hodnot
(např. shortint, integer, longint nebo byte, word)

Organizace hlavní paměti

Hlavní paměť je rozdělena na buňky neboli paměťová místa, kterým jsou přiřazena vzájemně různá nezáporná čísla (0, 1, 2, 3, ...) nazývaná adresy.

Obsah paměťového místa (závisí na procesoru):

- **slovo** **[word]**
  — např. 16 b, 32 b, 60 b anebo třeba 37 b, kde
  b označuje **bit**
- **slabika** („česky“ — bajt)
  - označení: **B** **[byte]**
  - zprav. **1 B = 8 b**
  - **určitý počet slabik** se někdy nazývá **slovo**
    (např. u procesorů 80x86: 1 slovo = 2 B)

Obsah paměťového místa na adrese a bývá někdy označován $\langle a \rangle$; nehrozí-li nedorozumění píše se však často a místo $\langle a \rangle$.

Slabiková organizace paměti

Př.: 1 slabika = 1B
1 slovo = 2B
1 dvojité slovo = 4B [DWord – Double Word]
↓
32b ... 8 šestnáckových číslic
Od adresy 5678 má být uloženo dvojité slovo 1234 ABCD:

	1.způsob	2.způsob
5678	12	CD
5679	34	AB
567A	AB	34
567B	CD	12

1. big-endian (IBM 360, Motorola 68000)
2. little-endian (Intel 80x86, DEC Alpha)
oba způsoby (Motorola 88 110)

Harvardská architektura

např. Motorola: 680x0, Intel: 8051

paměť dat ≠ paměť instrukcí

harvardská architektura $\overset{=}{\underset{\neq}{?}}$ **von Neumann**

non von Neumann

zkr.: ***non von***

počítače řízené tokem dat [data flow]
(versus: řízené tokem instrukcí — von Neumann)

počítače řízené tokem požadavků [demand driven]
(tzv. redukční počítače)

aj.
↑
paralelní zpracování informací — multiprocesorové systémy

Strojový kód

instrukce = příkaz (zakódovaný jako „číslo“)

Obsahuje (popř. může obsahovat) tyto informace:

① co se má provést (jaká operace)

② s čím se to má provést (operandy) a kam se má uložit výsledek

③ kde se má pokračovat (adr. násl. instr.)

Tyto informace mohou být obsaženy:

- explicitně *v instrukci*

  Př.: počítač SAPO — 5 adresové instrukce:
  - ① ... tzv. operační znak
  - ② ... 2+1 adresa
  - ③ ... 2 adresy — následující instrukce při záporném a nezáporném výsledku
- zčásti explicitně *v instrukci*, zčásti určeny *implicitně* architekturou procesoru, např.:
  - ① ... operační znak — OZ
  - ② ... adresová část instrukce

  } instrukce
  - ③ ... von Neumannova koncepce ⇒ další instrukce na následující adrese ← architektura

operační kód = *soubor instrukcí*: OZ ∼ operace

dále předpokl.: von Neumannova koncepce ⇒

⇒ *programový čítač* PC [Program Counter] obsahuje adresu následující instrukce

⇒ skoky (nepodmíněné a podmíněné) „neproduktivní“ operace

⇒ kratší instrukce

3 adresová instrukce:

OZ	a_1	a_2	a_3

- „nejpřirozenější“: 2 operandy + 1 výsledek

  např.: $\langle a_1\rangle - \langle a_2\rangle \to a_3$
- poměrně dlouhá

2 adresová instrukce:

OZ	a_1	a_2

- výsledek se ukládá na místo prvního operandu

  např.: $\langle a_1\rangle - \langle a_2\rangle \to a_1$
- je třeba zavést „neproduktivní“ operaci přesun: $\langle a_2\rangle \to a_1$

Př.: $\langle x\rangle - \langle y\rangle \to z \quad \equiv \quad \begin{cases} \langle x\rangle \to z \\ \langle z\rangle - \langle y\rangle \to z \end{cases}$

1 adresová instrukce:

OZ	a

- zvl. registr — *střadač* S [Accumulator]

  1. operand a výsledek např.: $\langle S\rangle - \langle a\rangle \to S$
- operace přesunu: $\langle a\rangle \to S$ a $\langle S\rangle \to a$

Př.: $\langle x\rangle - \langle y\rangle \to z \quad \equiv \quad \begin{cases} \langle x\rangle \to S \\ \langle S\rangle - \langle y\rangle \to S \\ \langle S\rangle \to z \end{cases}$

více střadačů ⇒ číslo střadače ∈ instrukce

⇒ lze zavést operace mezi střadači

Př.: Motorola — řada 68000

datové registry D_0, D_1, ..., D_7 à 32 b

odčítání 32 b: $D_n - \text{paměť} \to D_n$

OZ:

9	n	0	B	9

instrukce:

OZ	adresa

$\langle D_3\rangle - \langle$18 FF20÷18 FF23$\rangle \to D_3$:

96B9 0018 FF20

Zápis programu

strojový kód: strojové instrukce — „čísla“

Př.: Intel 8086 (zjednodušeno)

AX — střadač 16 b

A10401 $\langle 104\div105\rangle \to$ AX

2B060601 $\langle$AX$\rangle - \langle 106\div107\rangle \to$ AX

A30201 $\langle$AX$\rangle \to 102\div103$

- pracné programování i změny
- nepřehledný zápis

vyšší programovací jazyk (VPJ)

(např. Pascal)

Př.: a := b − c

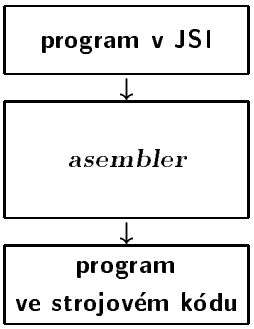
- omezení (zejm. pro „speciální“ účely)
- méně efektivní (?)

jazyk symbolických instrukcí (JSI):

operační znak i adresy zapsány symbolicky

Př.:
```
MOV AX,b
SUB AX,c
MOV a,AX
```

Vytváření programů ve strojovém kódu



vstupní data

program

výstupní data

VPJ: analogicky
místo asembleru jiný program
— *překladač (kompilátor)*

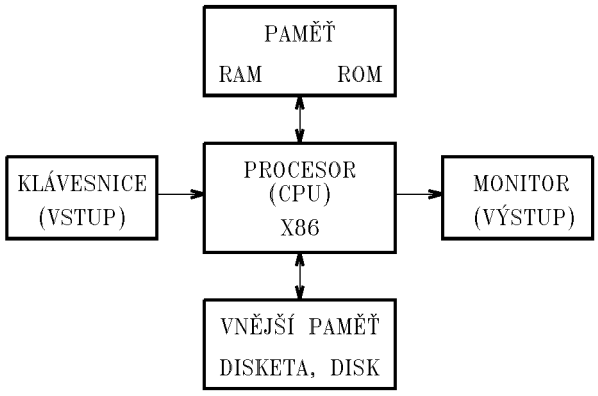
někdy:

program v JSI
program ve VPJ
} → překladač → modul

moduly → spojovací program [linker] → stroj. kód

Ideové schéma technické platformy

(IBM PC–kompatibilní osobní počítač)



Řada procesorů Intel 80x86

rok		tranz. [tis.]	MIPS	při MHz	dat.	adr.
1971	*4004	2,3	0.06	0,108	4	10
1978	8086	29	0.33	5	16	20
1979	8088	29	0.33	5	8	20
1982	80286	134	1.2	8	16	24
1985	80386DX	275	6	16	32	32
1988	80386SX	275	2.5	16	16	24
1989	i486DX	1200	20	25	32	32
1993	Pentium	3100	112	66	64	32
1995	Pentium Pro	5500	300	133	64	36
1997	Pentium II	7500		300	64	36
1999	Pentium III	28000		733	64	36

1981 IBM PC
1984 IBM PC-AT

Typy instrukcí X86 (=INSTRUKČNÍ SOUBOR)

- přesun dat
  (MOV,...,IN,OUT,...PUSH,POP,CLI,..)
- aritmetické i. (ADD,...,INC,..)
  logické i. (AND,...,XOR,...)
  posuvy (SHL,...,RCR,...)
- skoky (JMP,...,JC,JNC,....)
  cykly (LOOP,...,LOOPZ,...)
  podprogramy (CALL,RET,...)
  přerušení (INT,IRET,....)

- instrukce pro práci s řetězci
- pseudoinstrukce (např. deklarace proměnných DB,DW,..)
- makroinstrukce

Registry procesorů řady 80x86

16 bitů = 2 × 8 bitů

datové registry

0	AX	4	AH	AL	0 [Accumulator]
1	CX	5	CH	CL	1 [Counter]
2	DX	6	DH	DL	2 [Data]
3	BX	7	BH	BL	3 [Base]

ukazatelé

4	SP	Ukazatel zásobníku [Stack Pointer]
5	BP	Bázový registr [Base Pointer]
6	SI	Indexregistry [Source Index]
7	DI	Indexregistry [Destination Index]

segmentové registry

0	ES	[Extra Segment]
1	CS	program [Code Segment]
2	SS	zásobník [Stack Segment]
3	DS	data [Data Segment]

IP	Programový čítač [Instruction Pointer]
FLAGS	Registr příznaků [Status Flags]

registr příznaků — FLAGS

				O	D	*I*	T	S	Z		*A*		P		C
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

0	C ≡ CF	přenos	[Carry]
2	P ≡ PF	parita	[Parity]
4	***A*** ≡ ***AF***	*pomocný přenos*	[Auxiliary carry]
6	Z ≡ ZF	nula	[Zero]
7	S ≡ SF	znaménko	[Sign]
8	T ≡ TF	krokování	[Trap]
9	***I*** ≡ ***IF***	*přerušení*	[Interrupt]
10	D ≡ DF	směr	[Direction]
11	O ≡ OF	přeplnění	[Overflow]

IF, TF a DF: **ovlivňují činnost procesoru**

ostatní: **nastavovány některými instrukcemi informace o výsledku operace**

vybrané instrukce 8086

(v JSI)

přesuny dat

MOV *kam,co* ... *kam* := *co*

kam: datový reg., ukazatel, segment. reg., paměť

co: datový reg., ukazatel, segment. reg., paměť, konstanta

nelze: segment. reg. := konstanta
paměť := paměť

Př.:
```
MOV AL,DH        MOV DS,BX
MOV SI,65        MOV prom,AX
```

konstanta — tzv. *přímý operand* [immediate]

různé způsoby zápisu hodnot:

65535 = 0FFFFH = 1111111111111111B
65 = 41H = 01000001B = 1000001B = 'A'
3142H = '1B'

zápis čísla musí začínat desítkovou číslicí:
FFFFH — špatně 0FFFFH — dobře

nutný souhlas délek — platí i pro další instrukce !!!
Př.: MOV AX,BL — špatně (16 b versus 8 b)

deklarace proměnných:

— vyhrazení místa v paměti (pro data)
— přidělení symbolických jmen
— příp. jejich inicializace (počáteční hodnoty)

tzv. pseudoinstrukce:

- **DB** [Define Bytes] — slabiky (1 B = 8 b)
- **DW** [Define Words] — slova (2 B = 16 b)
- **DD** [Define Dwords] — dvojitá slova (4 B = 32 b)

⋮

Příklad:
```
bflm  DB   ?    ; poč. hodn. není definována
psvz  DB   5    ; poč. hodn. = 5
tab   DB   0FH, 5, '='   ; nahrazuje 3 deklarace
text  DB   'UPS'   ; místo 'U', 'P', 'S'
pole  DB   3 DUP (8, ?)   ; místo 8, ?, 8, ?, 8, ?
prom  DW   ?
      DB   1011B
adr   DD   bflm, prg   ; „adresy“
prg:  ⋮
      MOV AL,psvz   ; bflm = ?
      MOV bflm,AL   ; bflm = 5
```

aritmetické operace

Operandy a výsledek lze interpretovat jako:

- **čísla v doplňkovém kódu, např. (pro 1 B = 8 b):**
  $FF_{16} \sim$ (-1) $\in \langle -128_{10}; 127_{10} \rangle$
- **nezáporná čísla, např. (pro 1 B = 8 b):**
  $FF_{16} \sim FF_{16} = 255_{10} \in \langle 0; 255_{10} \rangle$

ADD α,β ... $\alpha := \alpha + \beta$

α:	β:
datový reg.	datový reg.
ukazatel	ukazatel
paměť	paměť
	konstanta

nelze: paměť := paměť + paměť

příznaky **CF, OF, SF a ZF:**

Př.: ADD AL,BL

nezáporná č.			doplňkový kód			příznaky			
AL	BL	AL+BL	AL	BL	AL+BL	CF	OF	SF	ZF
7A	78	0 F2	∼7A	∼78	∼(-E)	0	1	1	0
7A	FF	179	∼7A	∼(-1)	∼79	1	0	0	0

pozn.: dále se nastavují příznaky PF a AF

ADC α,β ... $\alpha := \alpha + \beta + CF$

SUB α,β ... $\alpha := \alpha - \beta$

analogické ADD, až na ***příznak*** **CF:**

$\alpha < \beta \iff CF := 1$ (α, β — nezáporná čísla)

zde: **CF ... tzv. výpůjčka** [borrow]

SBB α,β ... $\alpha := \alpha - \beta - CF$

CMP α,β ... $\alpha - \beta$

stejné jako SUB, ale neukládá se výsledek — nastaví se však ***příznaky !***

NEG α ... $\alpha := 0 - \alpha \quad = -\alpha$

(uvažuje se doplňkový kód)

Př.: MOV AL,1 ... 0000 0001
NEG AL ... 1111 1111 = 0FFH
CF=1, ZF=0, SF=1, OF=0

INC α ... $\alpha := \alpha + 1$ **nemění se příznak CF**

DEC α ... $\alpha := \alpha - 1$ **nemění se příznak CF**

logické operace

AND α,β ... $\alpha := \alpha \wedge \beta$ **logický součin**

OR α,β ... $\alpha := \alpha \vee \beta$ **logický součet**

XOR α,β ... $\alpha := \alpha \oplus \beta$ **součet modulo 2**

příznaky: CF := 0
SF ... výsl. < 0
ZF ... výsl. = 0
OF := 0

	0	1	0	1
	0	0	1	1
AND	0	0	0	1
OR	0	1	1	1
XOR	0	1	1	0

TEST α,β ... $\alpha \wedge \beta$ **stejné jako AND, ale neukládá se výsledek — nastaví se však** ***příznaky !***

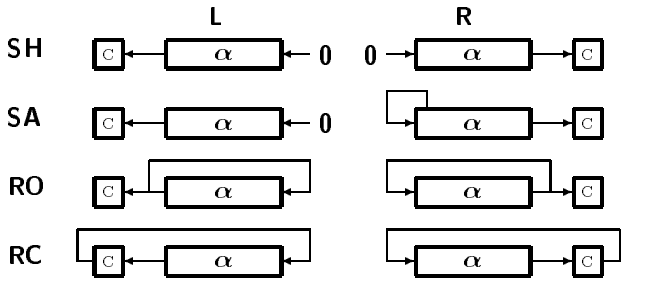
NOT α ... $\alpha := \overline{\alpha}$ **negace** ($\overline{0} = 1; \overline{1} = 0$)

Př.: MOV DH,13H ... 0001 0011
XOR DH,86H ... 1000 0110
1001 0101 = 95H
ZF=0, SF=1
NOT DH ... 0110 1010 = 6AH
ZF=0, SF=0

posuvy

posuv
- **logický** **SH** [SHift]
- **aritmetický** **SA** [Shift Arithmetic]
- **cyklický** **RO** [ROtate]
- **cyklický přes C** **RC** [Rotate through C]

posuv
- **vlevo** **L** [Left]
- **vpravo** **R** [Right]



SHL α, *o_kolik* $\quad o\_kolik = \begin{cases} 1 \\ CL \end{cases}$

analogicky: SHR, SAL, ...RCL

kromě příznaku C ≡ CF se mění některé další příznaky

Př.: MOV AX,1234H ... AX = 1234H
MOV CL,4
ROL AX,CL ... AX = 2341H CF=1

skoky

nepodmíněný skok (nejjednodušší varianta):

JMP *náv* — *náv* ... návěští instrukce

```
Př.:            MOV  AX,1      ①
                JMP  NAVESTI   ②
        COSI:   ADD  BX,CX
        NAVESTI: RCR AX,1      ③
```

podmíněné skoky:

JC *náv* ... je-li C=1, skok na *náv* (jinak nic)

JNC *náv* ... je-li C=0, skok na *náv* (jinak nic)

Př.: AX := max (BX,CX) — nezáporná čísla:

```
        MOV  AX,BX
        CMP  CX,BX
        JC   OK
        MOV  AX,CX
OK:
```

analogicky: JZ, JNZ, JS, JNS, JO, JNO, JP, JNP

podmíněné skoky orientované na srovnání:

relace	doplňkový kód	nezáporná čísla
$<$	JL	JB
$\leq$	JLE	JBE
$=$	JE	JE
$\neq$	JNE	JNE
$\geq$	JGE	JAE
$>$	JG	JA

E ... Equal
L ... Less
G ... Greater
B ... Bellow
A ... Above

podmínka skoku ⇐ příslušné příznaky, např.:

JB ... CF = 1 $\Longrightarrow$ JB ≡ JC
JBE ... CF∨ZF = 1
JL ... SF⊕OF = 1
JGE ... SF⊕OF = 0 atd.

pozn.: v předch. příkl. lze JC nahradit JB

Př.: AX := max (BX,CX) — doplňkový kód :
řešení: v předch. příkl. JC nahradit JL

rozsah podm. skoků: −128 ÷ 127 vůči IP

Př.:
```
        JC   OK
        MOV  AX,CX   ←IP   ↑128↑   ↓127↓
```

„delší“ skoky: opačný podm. skok + nepodm. skok

cykly

```
Př.:            MOV  CX,300
        CYKL:   ⋮
                DEC  CX      ; mění příznaky
                JNZ  CYKL
```

LOOP *náv* CX := CX − 1;
je-li CX ≠ 0, skok na *náv*

nemění příznaky

```
Př.:            MOV  CX,300
        CYKL:   ⋮
                LOOP CYKL    ; nemění příznaky
```

LOOPE *náv* CX := CX − 1;
je-li CX ≠ 0 a ZF = 1, skok na *náv*

LOOPNE *náv* CX := CX − 1;
je-li CX ≠ 0 a ZF = 0, skok na *náv*

LOOPZ *náv* jako LOOPE *náv*

LOOPNZ *náv* jako LOOPNE *náv*

rozsah skoku stejný jako u podmíněných skoků

Adresový prostor 8086

programový čítač (registr IP) ... adr. násl. instr.:

16 b ... $2^{16} = 64 \times 1\,024 = 64\,\mathrm{K} = 65\,536$

max. kapacita HP (pro 8086): ? perspektivní ?

$1\,\mathrm{M} = 2^{20} = 1\,048\,576$ ... 20 b

Řešení? *segmentace*:

IP neobsahuje „absolutní adresu“, ale tzv. její
posunutí [offset]
vůči 16násobku obsahu segm. reg. CS, tzn.

posunutí = adresa − 16 × ⟨segm. reg.⟩

tedy:

adresa = posunutí + 16 × ⟨segm. reg.⟩

Př.: IP = 1234_{16} CS = $9ABC_{16}$

1234	0001 0010 0011 0100	pos.
9ABC0	1001 1010 1011 1100 0000	segm.
9BDF4	1001 1011 1101 1111 0100	adr.

Př.: Je-li CS = 9ABC, pak:
minim. adr.: 9ABC0 = 9ABC0 + 0
maxim. adr.: AABBF = 9ABC0 + FFFF
tzv. *segment*
??? adresy 9ABBF nebo AABC0 apod. ???
nutno změnit obsah segmentového registru

Segmentace

hlavní paměť: rozdělena na *paragrafy* po 16 B
1 MB = 64 K paragrafů

segment. registr obsahuje číslo prvního paragrafu

segment: začíná na začátku některého paragrafu
64 KB = 4 K paragrafů

program: segm. reg. CS ⇒ programový segment
(někdy: kódový segment)

data: segm. reg. DS ⇒ datový segment
analogicky jako programový segment:
V instrukci není uvedena adresa dat,
ale její posunutí vůči 16násobku DS

další segmentové registry:
SS — tzv. zásobník (všimneme si později)
ES — data ve speciálních případech

implicitní segm. reg.: CS — program
DS — data (obvykle)
ES — data (někdy)
SS — zásobník

počáteční nastavení registrů:
techn. prostředky (HW) — CS, IP, DS, SS, ES, FLAGS
progr. prostředí (SW) — CS, IP, SS, SP (zprav.)
ostatní — nutno zajistit v programu !!!

Jednoduchý program

(1 datový a 1 programový segment)

```
    .MODEL SMALL      ; 1 dat. a 1 progr. segm.
    .STACK 100H       ; zásobník
    .DATA             ; datový segment
A   DW   12 DUP (?)
B   DW   1
C   DW   ?
      :
    .CODE             ; programový segment
      :
S:  MOV  AX, @DATA    ; start
    MOV  DS, AX       ; nastavení DS
      :
K:  MOV  AX,4C00H     ; stop
    INT  21H
      :
    END  S            ; S = startovaci adresa
```

.MODEL, .STACK, .DATA, .CODE, END } tzv. direktivy (příkazy) asembleru

@DATA = první paragraf dat. segmentu (konstanta)

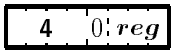
Strojový kód 8086

(příklady)

① CLI ... IF := 0

② $reg \in \{AX, \ldots DI\}$ — 2 slabiky (2 B)

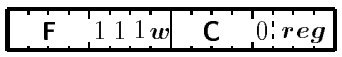
INC reg



př.: INC SI ... 46

③ reg ... dat. reg. nebo ukazatel — 1 nebo 2 B

INC reg

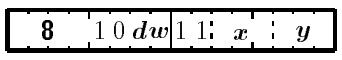


[word]

w	
0	1 B
1	2 B

př.: INC DH ... FE C6 INC SI ... FF C6

④ MOV $r1, r2$



[direction]

d	
0	$x \rightarrow y$
1	$x \leftarrow y$

př.: MOV CH,AL ... 88 C5 nebo 8A E8

Strojový kód 8086 (příklady) 2

⑤ MOV *reg, konst*

B	*w*	*reg*	*konst*

př.: MOV CH,41H ... B5 41
MOV CX,123H ... B9 23 01

⑥ MOV *reg, pam* a MOV *pam, reg*

8	1 0 *dw*	0 0	*reg*	1 1 0	*posunut*

d	
0	*reg* → *pam*
1	*reg* ← *pam*

př. předp.: posunutí(Beta) = $12_{10} = C_{16}$
MOV CX,Beta ... 8B 0E 0C 00
MOV Beta,CX ... 89 0E 0C 00

⑦ JNZ *nv*

7	5	*vzd*

př.: 0004F: 40 CYKL: INC AX
00050: 75 FD JNZ CYKL
00052: ?? ???
4F − 52 = −3 ∼ FD (doplňkový kód)

Nepodmíněné skoky (JMP)

- tzv. **krátké** — JMP SHORT *náv*
  vzd = „vzdálenost“: $-128 \le vzd \le 127$

E	B	*vzd*

(2 B)

IP := IP + *vzd*

srov.: **podmíněné skoky** (všechny jsou „krátké“)

- tzv. **blízké** — JMP NEAR PTR *náv*
  uvnitř segmentu (programového)

E	9	*vzd*

(3 B)

IP := IP + *vzd*

- tzv. **daleké** — JMP FAR PTR *náv*
  mezi segmenty (programovými)

E	A	*posunutí*	*segment*

(5 B)

IP := *posunutí* [offset]
CS := *segment*

Pozn.: existují ještě jiné nepodmíněné skoky
(„cílová“ adresa je v registru nebo v proměnné)

Pozn.: Asembler zprav nevyžaduje, aby se uvádělo „NEAR PTR“

Způsoby adresace

operandy/výsledek: 1. registry
2. místa v hlavní paměti

ad 2:

- *přímé adresy*: dosud uvažované adresy, např.:
  ADD AX,xyz nebo ADD AX,xyz+3
- *indexované adresy* — reg. SI a DI, např.:

```
Arr     DB    8,3,5,4
        :
        MOV   SI,0          ; SI = 0
        MOV   AL,0
        MOV   CX,4
cyklus: ADD   AL, Arr[SI]   ; AL = 8, 11, 16, 20
        INC   SI            ; SI = 1, 2, 3, 4
        LOOP  cyklus
```

lze však také použít např. 7[DI] nebo [SI] ≡ 0[SI]

- *bázované adresy* — reg. BX a BP
  analogicky — např. [BX], 3[BX] anebo Arr[BP]
- *kombinace* — např. [SI][BX], 5[BP][DI]

Pozn.: Jsou možné i jiné zápisy, např.
5[BP+DI] ≡ 5[BP][DI]

datové segmenty

implicitní segmentový registr — DS

výjimky: adresy bázované registrem BP — SS
některé operady operací s řetězci — ES

Př.: SI = 1, BP = 2, DS = 3, SS = 4
8[SI] ∼ 00039 8[BP] ∼ 0004A

explicitní určení sementového registru:

prefixové instrukce (krátce — prefixy):

ES:	SEGES	(26)
CS:	SEGCS	(2E)
SS:	SEGSS	(36)
DS:	SEGDS	(3E)

Př.: SI = 1, BP = 2, DS = 3, SS = 4
SS:8[SI] ∼ 00049 DS:8[BP] ∼ 0003A

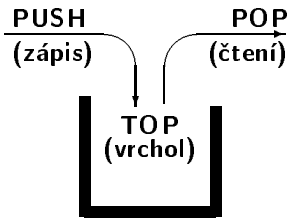
Př.: mov ax,ES:qwert ≡ SEGES mov ax,qwert

Př.:	JSI	stroj. kód
	mov ax,1[si]	8B 44 01
	mov ax,ES:1[si]	26 8B 44 01

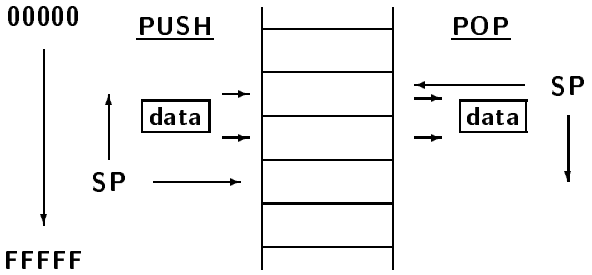
Zásobník

(zásobníková paměť, sklípková paměť, paměť LIFO)

[stack, push-down, ..., Last-In-First-Out]



zásobník simulovaný v hlavní paměti



adresa v hlavní paměti = SP + 16 × SS
SP ... posunutí [offset]

Instrukce PUSH a POP

PUSH γ ... $\gamma \rightarrow$ *zásobník*; SP := SP − 2

γ: registr — 2 slabiky — kromě IP a FLAGS
paměť — 2 slabiky

predekrementace SP — nejprve snížit, pak zapsat

Př.: SS = 8A74 SP = 20

PUSH AX

1. SP := 1F
2. AH → 8A75F
3. SP := 1E
4. AL → 8A75E

POP δ ... $\delta \leftarrow$ *zsobnk* ; SP := SP + 2

δ: registr — 2 slabiky — kromě CS, IP a FLAGS
paměť — 2 slabiky

postinkrementace SP — nejprve číst, pak zvýšit

Př.: SS = 8A74 SP = 1E

POP AX

1. ⟨8A75E⟩ → AL
2. SP := 1F
3. ⟨8A75F⟩ → AH
4. SP := 20

Instrukce CALL

Skok do podprogramu (=volání procedury)

CALL u (u je adresa začátku podprogramu)

Př

deklarace podprogramu:

```
MAX     PROC              ;max (BX,CX) → AX
max1:   MOV     AX,BX     ;srov. UPS2•17
        CMP     CX,BX
        JNC     jinak
        RET               ;návrat z podprogramu
jinak:  MOV     AX,CX
        RET               ;návrat z podprogramu
MAX     ENDP              ;konec zápisu
                          ;podprogramu
```

volání podprogramu:

```
        MOV     BX,data1  ;zadání 1. parametru
        MOV     CX,data2  ;zadání 2. parametru
        CALL    MAX       ;volání podprogramu
        MOV     vysl,AX   ;uložení výsledku
```


CALL

Deklarace a volání podle umístění podprogramu vzhledem k místu volání (viz nepodmíněný skok):

	uvnitř segmentu	mezi segmenty
deklarace	PROC NEAR ENDP	PROC FAR ENDP
volání délka instr. zásobník	CALL NEAR PTR 3 B "PUSH IP"	CALL FAR PTR 5 B PUSH CS "PUSH IP"
návrat zásobník	RET "POP IP"	RET "POP IP" "POP CS"

Předávání parametrů: (viz "nahrazení formálních parametrů skutečnými" ve vyšších programovacích jazycích) zajišťuje programátor prostřednictvím:

- registrů
- paměti
- zásobníku

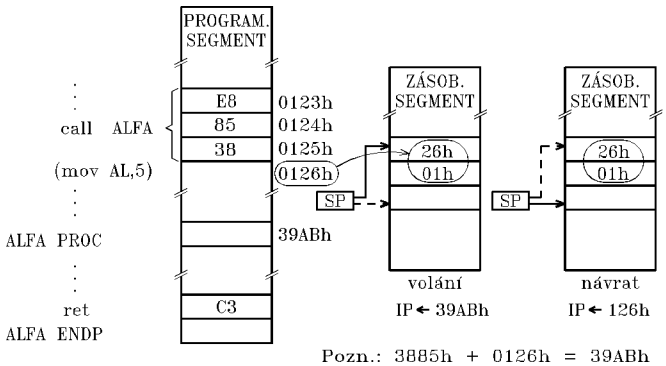
Př.1 (uvnitř segmentu)

Nelze volat z jiného segmentu!

```
39AB    alfa    PROC NEAR   ;pseudoinstrukce
                ...
                RET         ;stroj. kód - C3
        alfa    ENDP        ;pseudoinstrukce
```

0123 CALL NEAR PTR alfa

0126 další instrukce (např. MOV AL,5)

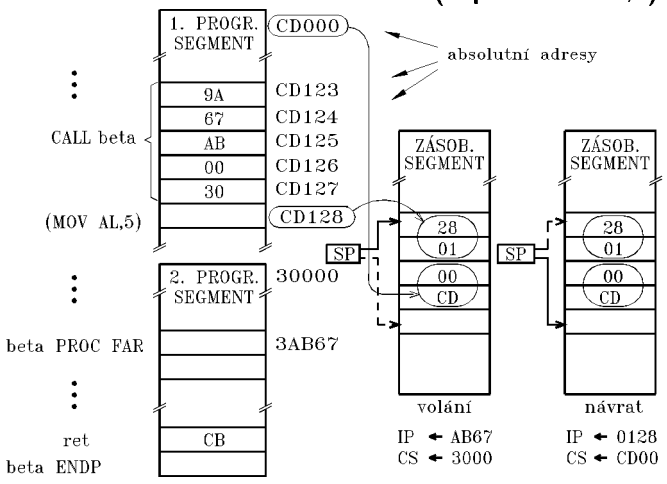


Pozn.: 3885h + 0126h = 39ABh

Př.2 (deklarace je v jiném segmentu než volání)

Při volání vždy použít FAR - i ve stejném segm.!

```
3000:AB67   beta    PROC FAR    ;pseudoinstrukce
                    ...
                    RET         ;stroj. kód - CB
            beta    ENDP        ;pseudoinstrukce
CD00:0123           CALL FAR PTR beta
     0128           další instr. (např. MOV AL,5)
```



Instrukce IN, OUT

obsluha vstupů a výstupů

vstup:

IN Ac, brána Ac ← brána

IN Ac, DX Ac ← [DX]

výstup:

OUT brána, Ac brána ← Ac

OUT DX, Ac [DX] ← Ac

kde:

Ac - AX nebo AL (tzn. 16 nebo 8 bitů)

brána - konstanta z intervalu ⟨ 0, 0FFh ⟩

DX - 16 bitů ⟨ 0, 0FFFFh ⟩

brány [ports] = logické obvody (registry) + pravidla

adresy těchto registrů tvoří nezávislý datový prostor (=vstupní/výstupní adresový prostor), oddělený od paměti pro programy a data, který je přímo adresovatelný (bez segmentových registrů), přístupný pomocí instrukcí IN a OUT

PŘERUŠENÍ

způsobí, že procesor přestane (dočasně) provádět právě probíhající program a místo něj začne provádět jiný program, který přerušení tzv. obslouží (tj. reaguje na jev, který přerušení vyvolal)

vnější - periferie, uživatel, havarijní stavy, ...

- nemaskovatelné vstup NMI
- maskovatelné (z řadiče přerušení) INTR

vnitřní

- chyby operandů, výsledku, krokování, ...,
- instrukce INT n (n je 8bitová konstanta)

- před obsluhou přerušení se uloží na zásobník informace o tom, jaký program se právě prováděl (FLAGS, CS, IP)
- zakáže se další přerušení (CLI)
- zjistí se, jak daný typ přerušení obsloužit - nastaví se nové CS a IP

- při návratu z přerušení je třeba obnovit informace o původním programu (instrukce IRET - ze zásobníku se vyzvednou IP, CS, FLAGS)

maskování přerušení - zákaz/povolení přerušení pomocí příznaku IF (jen maskovatelná přerušení)

instrukce: STI povolení přerušení IF:=1

CLI zákaz přerušení IF:=0

vektor přerušení - dvojité slovo (32 bitů) obsahující nové CS a IP (určení začátku podprogramu pro zpracování přerušení)

adresa vektoru přerušení se získá vynásobením tzv. typu (čísla) přerušení 4:

Př. INT 10h → na adresách 00040h až 00043h je uložen IP a CS pro zpracování tohoto typu přerušení

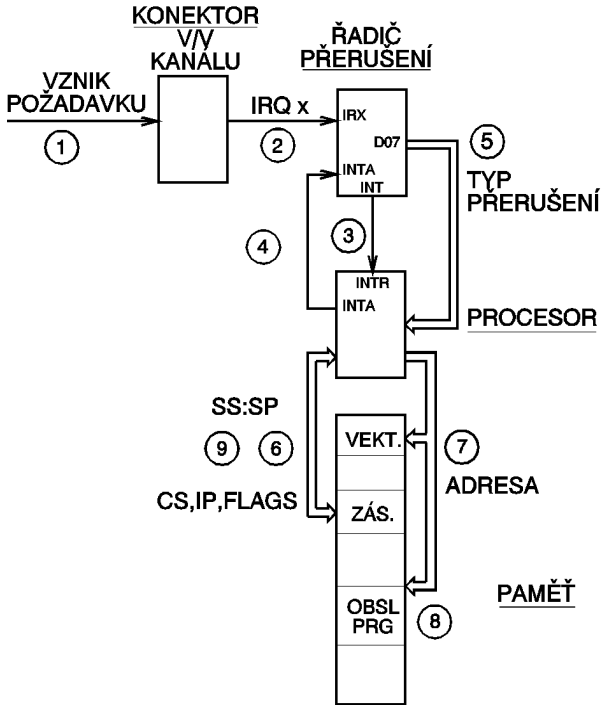
V paměti na nejnižších adresách je vyhražen prostor 0 ÷ 3FFh, kde je 256 vektorů - 32bitových adres podprogramů pro zpracování daného typu přerušení.

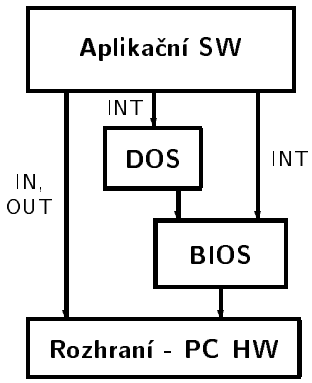
řadič přerušení - logický obvod (IO 8259A), který přijímá signály od V/V identifikuje požadavky na přerušení podle jejich priority [interrupt request - IRQ] generuje přerušovací signál (INT → INTR)

POSLOUPNOST ČINNOSTÍ PŘI OBSLUZE ŽÁDOSTI O VNĚJŠÍ PŘERUŠENÍ U PC (viz UPS4-9)

HW	1-3	vznik žádosti o přerušení
	4	rozhodnutí o obsluze (je-li IF=1 a INTA)
	5	identifikace příčiny přerušení (podle čísla - typu)
	6	uložení stavové informace (FLAGS), segmentového registru (CS) a čítače instrukcí (IP) na zásobník
	7	nalezení adresy začátku programu pro obsluhu daného typu přerušení pomocí vektoru přerušení - nové CS, IP
SW	8	provedení programu obsluhy přerušení
	9	návrat do přerušeného programu (IRET) obnovení IP, CS, FLAGS ze zásobníku

POSLOUPNOST ČINNOSTÍ PŘI OBSLUZE ŽÁDOSTI O PŘERUŠENÍ U PC





BIOS - v paměti ROM, nejnižší úroveň SW v PC (Basic Input/Output System)
(obsluha obrazovky, diskových jednotek, tiskáren, klávesnice, získání informací o konfiguraci systému, velikosti paměti, ...)

DOS - v paměti RAM po natažení s disku (diskety) (Disk Operating System)

Přerušení způsobené instrukcí INT:

V podstatě volání podprogramů, tzn. včetně předávání parametrů (před instrukcí INT) - zde přes registry

Př.1.

využití služeb BIOSu (Basic Input/Output System) pro zobrazení znaku na obrazovku na okamžitou pozici kurzoru

INT 10h

```
MOV   AH, 09h      ;číslo funkce do reg. AH
MOV   AL, znak     ;zobrazovaný znak do reg. AL
MOV   BH, strana   ;číslo stránky do reg. BH
MOV   BL, atrib    ;atribut do reg. BL
MOV   CX, opak     ;kolikrát má byt znak opakován
INT   10h          ;volání BIOSu
```

Poznámka:
zobrazení jednoho znaku (pro zobrazení více znaků je třeba použít cyklu), neposouvá se kurzor

Př.2.

Využití služeb jádra operačního systému (volání funkcí jádra DOSu) pro zobrazení řetězce znaků na obrazovku

INT 21h

```
text  DB     'toto je text na obrazovce$'
      .
      .
      MOV    DX,offset text   ;adresa řetězce
      MOV    AH, 09           ;číslo funkce
      INT    21h              přerušení DOS
```

Poznámka:
řetězec musí být ukončen symbolem $

Rozdíl INT 10h a INT 21h ?!
(mohu/nemohu, nebo musím/nemusím ovládat vše)

Výrokový počet

Pravdivostní hodnoty:

pravda	P	ano	+	*true*	1	aj.
nepravda	N	ne	−	*false*	0	aj.

Př.: $M1$ ... 1. osoba je muž
$M2$ ... 2. osoba je muž
$S1$ ... 1. osoba je svobodná
$S2$ ... 2. osoba je svobodná
V ... mohou se vzít

$$V = \{ [(\boxed{ne}\ M1) \boxed{a}\ M2] \boxed{nebo}\ [M1 \boxed{a} (\boxed{ne}\ M2)] \} \boxed{a}\ S1 \boxed{a}\ S2$$

Konvence:

$X \boxed{a} Y \rightarrow X \cdot Y$
$X \boxed{nebo} Y \rightarrow X + Y$
$\boxed{ne}\ X \rightarrow \overline{X}$
pravda $\rightarrow$ 1
nepravda $\rightarrow$ 0

Př.: $\boxed{V = (\overline{M1} \cdot M2 + M1 \cdot \overline{M2}) \cdot S1 \cdot S2}$

Boolova algebra (logická algebra)

logický $\stackrel{?}{=}$ Boolův (boolovský) [G. Boole]

Boolovy (logické) funkce (operace):

X	$\overline{X}$
0	1
1	0

X	Y	$X \cdot Y$	$X + Y$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

$\overline{X}$ funkce NE, NOT, negace, inverze
$X \cdot Y$ funkce I, AND, logický součin, konjunkce, ...
$X + Y$ funkce NEBO, OR, logický součet, disjunkce, ...

Shannonův expanzní teorém:

$$\boxed{f(a,b,\ldots,c) = a \cdot f(1,b,\ldots,c) + \overline{a} \cdot f(0,b,\ldots,c)}$$

Důkaz: vztah platí pro všechna a (pro $\forall a \in \{0,1\}$)

Důsledek: $f(a,b,\ldots,c) = a \cdot g(b,\ldots,c) + \overline{a} \cdot h(b,\ldots,c)$

Každá logická funkce se dá zapsat pomocí logického součinu, součtu a negace

Zákony Booleovy algebry

⊙ komutativní ⊙

$\boxed{x + y = y + x}$ $\boxed{x \cdot y = y \cdot x}$

asociativní

$\boxed{(x+y)+z = x+(y+z)}$ $\boxed{(x \cdot y) \cdot z = x \cdot (y \cdot z)}$

⊙ distributivní ⊙

$\boxed{(x+y) \cdot z = x \cdot z + y \cdot z}$ $\boxed{(x \cdot y) + z = (x+z) \cdot (y+z)}$

⊙ o neutrálnosti 0a 1⊙

$\boxed{x + 0 = x}$ $\boxed{x \cdot 1 = x}$

o agresivnosti 0a 1

$\boxed{x \cdot 0 = 0}$ $\boxed{x + 1 = 1}$

o idempotenci prvků

$\boxed{x + x = x}$ $\boxed{x \cdot x = x}$

⊙ vyloučeného třetího ⊙

$\boxed{x + \overline{x} = 1}$ $\boxed{x \cdot \overline{x} = 0}$

o dvojí negaci

$\boxed{\overline{\overline{x}} = x}$

De Morganova pravidla

$\boxed{\overline{x} + \overline{y} = \overline{x \cdot y}}$ $\boxed{\overline{x} \cdot \overline{y} = \overline{x + y}}$

⊙ Huntingtonovy axiomy ⊙

Princip duality:

platí zákon $\mathcal{Z} \Rightarrow$ platí duální zákon $\mathcal{Z}^*$

$\mathcal{Z} \rightarrow \mathcal{Z}^*$:

+	→	·
·	→	+
0	→	1
1	→	0

Př. — viz zákony Booleovy algebry

Př.: $x + x \cdot y = x \cdot (1 + y) = x \cdot 1 = x$

Př.: $x + \overline{x} \cdot y = (x + \overline{x}) \cdot (x + y) = 1 \cdot (x + y) = x + y$

Další zákony Booleovy algebry

absorpce

$\boxed{x + x \cdot y = x}$ $\boxed{x \cdot (x + y) = x}$

absorpce negace

$\boxed{x + \overline{x} \cdot y = x + y}$ $\boxed{x \cdot (\overline{x} + y) = x \cdot y}$

Pozorování:

1. Je-li číslo dělitelné 2 **nebo** 3, není to prvočíslo.
   X ... **Číslo je dělitelné 2.**
   Y ... **Číslo je dělitelné 3.**
   X nebo Y: **1 nebo 1 = 1**
   „obyčejné“ nebo $X + Y$

2. Bude hodný, **nebo** dostane pár facek.
   X ... **Bude hodný.**
   Y ... **Dostane pár facek.**
   X nebo Y: **1 nebo 1 = 0**
   Tzv. ***vylučovací nebo*** $X \oplus Y$

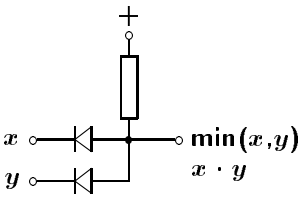
závěr: **nebo ≠ nebo**

Další logické funkce

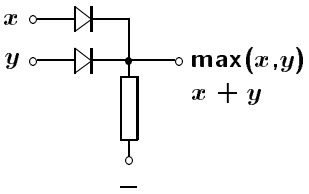
X	Y	1	2	3	4	5	···
0	0	0	1	1	1	1	···
0	1	1	0	1	0	1	···
1	0	1	0	1	0	0	···
1	1	0	0	0	1	1	···

1. **vylučovací NEBO, XOR** [eXclusive OR], **nonekvivalence, součet modulo 2, ...**
   $X \oplus Y = X \cdot \overline{Y} + \overline{X} \cdot Y$
2. **funkce ANI, NOR, Pierceova funkce, ...**
   $X \downarrow Y = \overline{X + Y}$
3. **funkce NAND, Shefferova funkce, ...**
   $X \uparrow Y = \overline{X \cdot Y}$
4. **ekvivalence**
   $X \sim Y = X \cdot Y + \overline{X} \cdot \overline{Y}$
5. **implikace**
   $X \rightarrow Y = \overline{X} + Y$

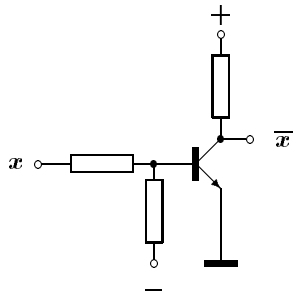
tzv. ***pozitivní logika***



x	y	min(x,y)
0	0	0
0	1	0
1	0	0
1	1	1

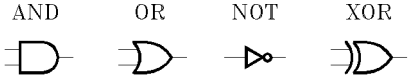


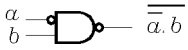
x	y	max(x,y)
0	0	0
0	1	1
1	0	1
1	1	1

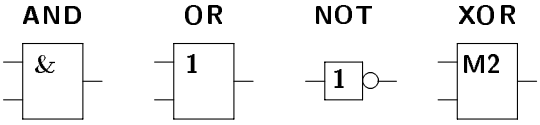


x	$\overline{x}$
0	1
1	0

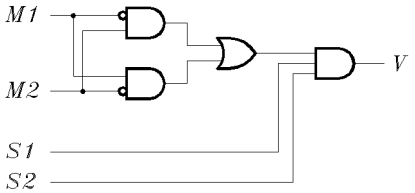
Schematické značky logických členů



negace: 



Př.: $V = (\overline{M1} \cdot M2 + M1 \cdot \overline{M2}) \cdot S1 \cdot S2$



Př.: **Majorita** ze tří — je rovna 1 právě, když **většina** proměnných (2 nebo 3) je rovna 1;
označuje se např. $M_3(x, y, z)$ nebo $x\#y\#z$.

$M_3 = \overline{x}\ y\ z + x\ \overline{y}\ z + x\ y\ \overline{z} + x\ y\ z$

1×OR4 + 4×AND3 + 3×NOT

8 logických členů 19 vstupů

$M_3 = \overline{x}\ y\ z + x\ \overline{y}\ z + x\ y\ (\overline{z} + z)$

$M_3 = \overline{x}\ y\ z + x\ \overline{y}\ z + x\ y$

1×OR3 + 2×AND3 + 1×AND2 + 2×NOT

6 logických členů 13 vstupů

$xyz = xyz + xyz + xyz$

$M_3 = \overline{x}yz + xyz + x\overline{y}z + xyz + xy\overline{z} + xyz$

$M_3 = y\ z + x\ z + x\ y$

1×OR3 + 3×AND2

4 logické členy 9 vstupů

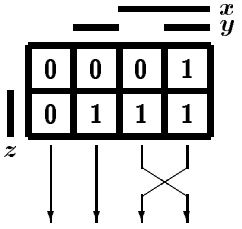
Př.: majorita ze tří — $M_3(x, y, z)$

Pravdivostní tabulka

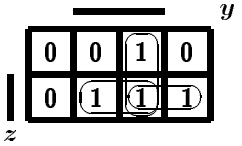
i	x	y	z	M_3
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

i ... tzv. stavový index

Svobodova mapa

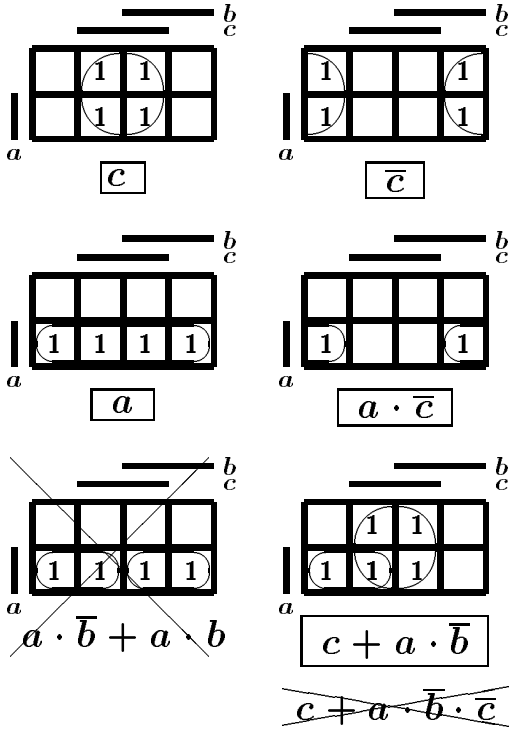


Karnaughova mapa



Algebraický výraz: $M_3(x, y, z) = xy + xz + yz$

Stavové indexy pro $f(\ldots) = 1$: $M_3 \sim \{3, 5, 6, 7\}$



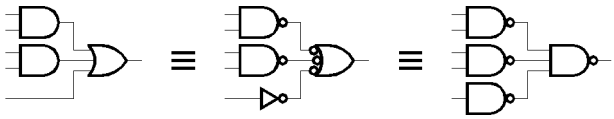
c , $\overline{c}$, a , ... $a \cdot \overline{c}$, $a \cdot \overline{b} \cdot \overline{c}$

implikanty neboli **termy** (součinové)

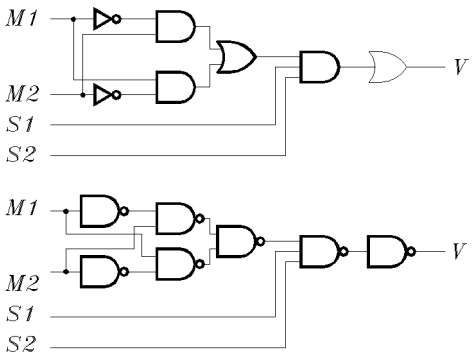
Použití členů NAND

De Morganovo pravidlo:

Z toho plyne:

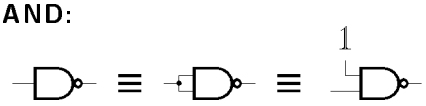


Př.:



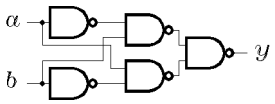
Použití členů NAND

1 vstupový člen NAND:

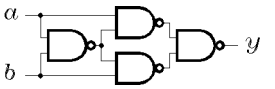


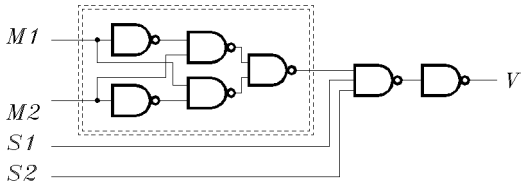
XOR:

$$y = a \oplus b = \\ = a\overline{b} + \overline{a}b$$



$$y = a \oplus b = \\ = a\overline{b} + 0 + \overline{a}b + 0 = \\ = a\overline{b} + a\overline{a} + \overline{a}b + \overline{b}b = \\ = a(\overline{a}+\overline{b}) + (\overline{a}+\overline{b})b \\ = a \cdot \overline{a\,b} + \overline{a\,b} \cdot b$$





Sčítačky

a, b bity sčítanců p přenos z nižšího řádu

s bit součtu q přenos do vyššího řádu

úplná sčítačka

a	b	p	q	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$q = M_3(a, b, p)$

$s = a \oplus b \oplus p$

$q = ab + ap + bp$

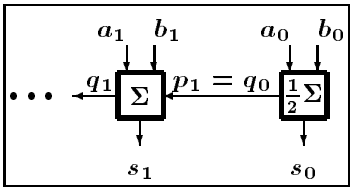
$s = \overline{a}\overline{b}p + \overline{a}b\overline{p} + \\ + a\overline{b}\overline{p} + abp$

půlsčítačka (poloviční sčítačka)

a	b	q	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$q = a \cdot b$

$s = a \oplus b = \\ = \overline{a}b + a\overline{b}$

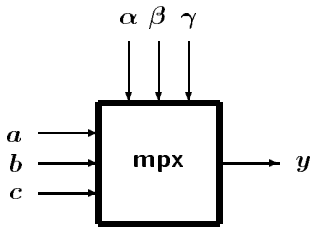


Dekodér

a	b	c	y_0	y_1	y_2	...	y_7
0	0	0	1	0	0	...	0
0	0	1	0	1	0	...	0
0	1	0	0	0	1	...	0
0	1	1	0	0	0	...	0
1	0	0	0	0	0	...	0
1	0	1	0	0	0	...	0
1	1	0	0	0	0	...	0
1	1	1	0	0	0	...	1

$$y_0 = \overline{a}\,\overline{b}\,\overline{c} \\ y_1 = \overline{a}\,\overline{b}\,c \\ y_2 = \overline{a}\,b\,\overline{c} \\ \vdots \\ y_7 = a\,b\,c$$

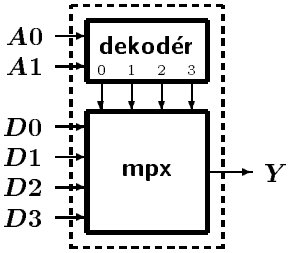
Multiplexory



α	β	γ	y
0	0	0	0
1	0	0	a
0	1	0	b
0	0	1	c
jinak			?

$$y = a \cdot \alpha + b \cdot \beta + c \cdot \gamma$$

MPX

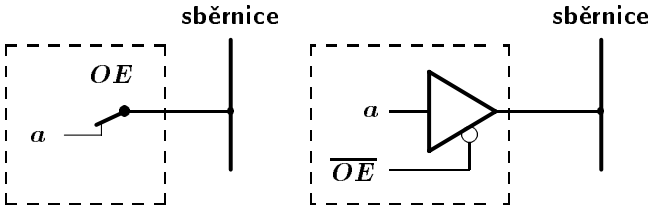


„adresa“ $A1$	$A0$	Y
0	0	$D0$
0	1	$D1$
1	0	$D2$
1	1	$D3$

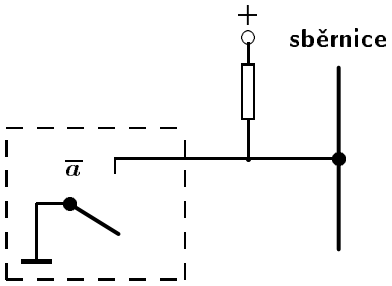
Připojení obvodů ke sběrnicím

3stavový výstup: 0, 1, Z

Z = stav vysoké impedance

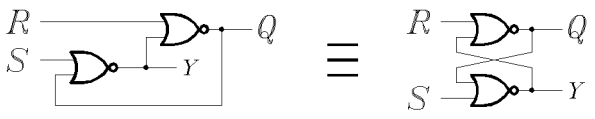


otevřený kolektor:



Sekvenční obvody

Př



R	S	Q	Y
1	1	0	0
1	0	0	1
0	1	1	0
0	0	?	?

$Q = \varphi(Y)$ a $Y = \psi(Q)$

pozorování:

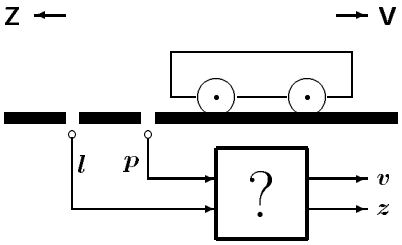
R	S	Q	Y
1	0	0	1
0	0	0	1
0	1	1	0
0	0	1	0

závěr: výstupy závisí na „historii“

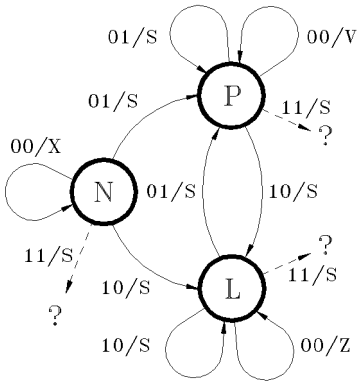
sekvenční obvody **(dosud: kombinační obvody)**

zpětná vazba !?!

Návrh sekvenčního obvodu (příklad)



graf přechodů



výstupy:

v	z		
0	0	X	???
0	1	Z	západ
1	0	V	východ
1	1	S	střed

vnitřní stavy:

N	...	Nevím
L	...	vLevo
P	...	vPravo

Návrh sekvenčního obvodu (příklad) **2**

graf přechodů → ***tab. přechodů*** **(a výstupů):**

(l,p)	00	10	11	01
N	N	L	?	P
L	L	L	?	P
P	P	L	?	P

00	10	11	01
X	S	S	S
Z	S	S	S
V	S	S	S

stavy: • **vnitřní** (krátce jenom **stavy**): **N, L, P**
• **vstupní: 00, 01, 11, 10**
• **výstupní: X, Z, V, S**

kódování stavů:

stav (vnitřní, ...) ∼ hodnoty logických proměnných
logické proměnné: vnitřní, vstupní, výstupní

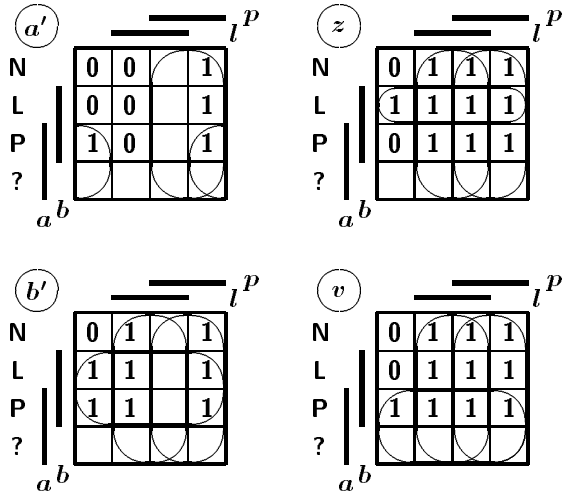
- **vstupní stavy: byly kódovány již při zadání**
- **výstupní stavy: viz předchozí folii**
- **vnitřní stavy:**

	a	b
N	0	0
L	0	1
P	1	1

a a b jsou vnitřní proměnné

Návrh sekvenčního obvodu (příklad) **3**

kódované tabulky přechodů a výstupů



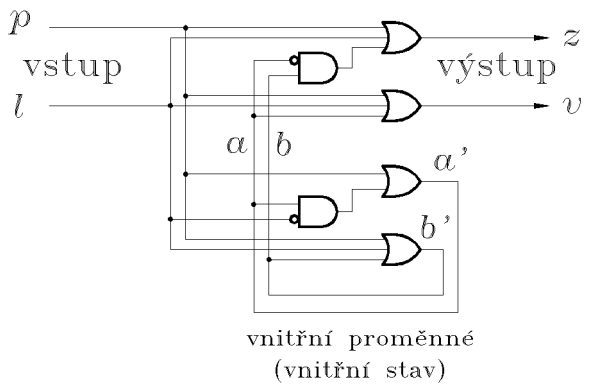
$a' = p + a \cdot \bar{l}$ $\quad z = p + l + \bar{a} \cdot b$

$b' = p + l + b$ $\quad v = p + l + a$

samostatný úkol: Proveďte zpětné doplnění tabulky přechodů a grafu přechodů (tzn. nahraďte otazníky výslednými hodnotami).

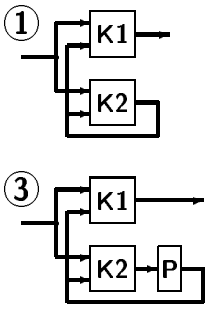
Návrh sekvenčního obvodu (příklad) 4

schéma zapojení



samostatný úkol: Pokuste se doplnit „nulování“ (tzn. nastavení do stavu N).

struktura sekvenčních obvodů

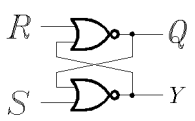


K1, K2 — kombinační obvody

P — paměťové členy

paměťové členy — **klopné obvody (asynchronní)** [flip-flop]

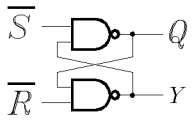
typ R–S [Reset–Set]



R	S	Q'	Y'
0	0	Q	$\overline{Q'}$
0	1	1	$\overline{Q'}$
1	0	0	$\overline{Q'}$
1	1	0	0

„pamatuje“
zápis 1
zápis 0
???

typ $\overline{R}$–$\overline{S}$



$\overline{R}$	$\overline{S}$	Q'	Y'
1	1	Q	$\overline{Q'}$
1	0	1	$\overline{Q'}$
0	1	0	$\overline{Q'}$
0	0	1	1

R a S, popř. $\overline{R}$ a $\overline{S}$ — tzv. ***budicí funkce***

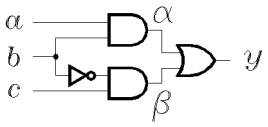
některé problémy:

fundamentální režim:

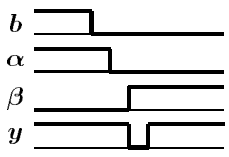
- **změna jediné vstupní proměnné**
- **další změna: až po ustálení výstupů**

hazardy **(příklady):**

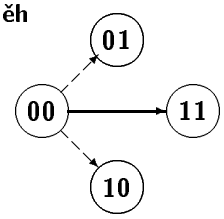
1 **tzv. statický hazard v 1**



$a = 1$ a $c = 1 \Rightarrow$
$\Rightarrow y = a \cdot b + \overline{b} \cdot c = 1$



2 **tzv. souběh**



„řešení“: tzv. **synchronní sekvenční obvody**
(dosud: tzv. **asynchronní)**

synchronní sekvenční obvody

do zpětných vazeb se zařadí tzv.

synchronní paměťové členy (klopné obvody)

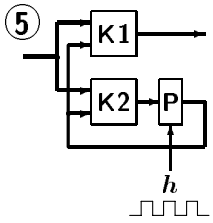
tj. obvody, jejichž vstupy jsou vzorkovány tzv.

hodinovými pulsy

(krátce „hodinami“) [clock]

⇒ přechodové děje jsou ignorovány,

⇒ „diskrétní čas“ — takty.

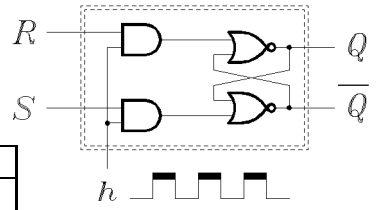


synchronní paměťové členy (klopné obvody):

- **hladinové (řízené hladinou)** [latch]
- **hranové (řízené hranou)** [flip-flop]

hladinové klopné obvody [latch]

1. typ R–S [Reset – Set]

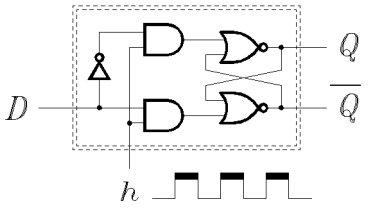


h	
0	„pamatuje se“
1	vst. $\rightsquigarrow$ výst.

$R \cdot S = 1$ & $h: 1 \to 0$ $\Longrightarrow$ $Q = ?$

proto: $\boxed{R \cdot S \neq 1}$ — nutno zajistit při návrhu

2. typ D [Data]

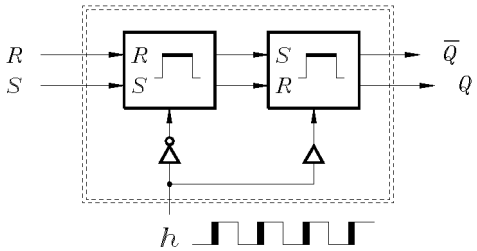


problém: šířka hodin. pulsů → hranové klop. obv.

použití hladin. klop. obvodů: mimo zpětné vazby (registry — někdy)

hranové klopné obvody

tzv. obvody „*master – slave*“ (typ R–S)



změna výstupů (Q, popř. $\overline{Q}$): jen při náběžné hraně hodinových pulsů

společné hodiny $\Longrightarrow$

- hodiny ↗ — změna stavu všech klop. obv.
- potom: přechodový děj v kombinačních obv.
- hodiny ↗ — ⋮ atd.

Pozn.:

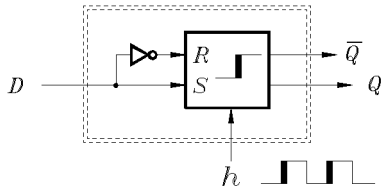
1. modifikace uvedeného zapojení: závěrná hrana místo náběžné hrany (↘ místo ↗)
2. existují i jiné možnosti realizace hran. klop. obvodů
3. i zde nutno zajistit $\boxed{R \cdot S \neq 1}$ → jiné typy

typy hranových klopných obvodů

1. R–S [Reset – Set]
(viz předchozí folie)

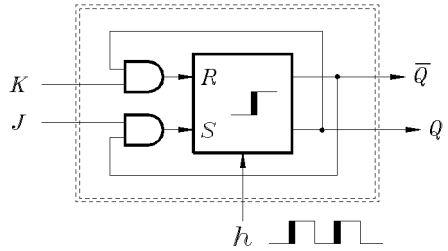
R	S	Q^{t+1}
0	0	Q^t
0	1	1
1	0	0
1	1	?

2. D [Data]



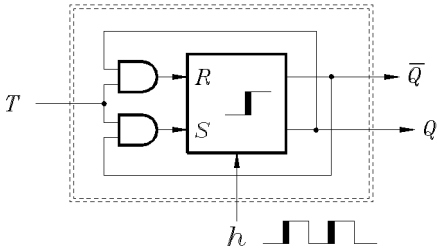
D	Q^{t+1}
1	1
0	0

3. J–K [Jordan – Eccles]



J	K	Q^{t+1}
0	0	Q^t
0	1	0
1	0	1
1	1	$\overline{Q^t}$

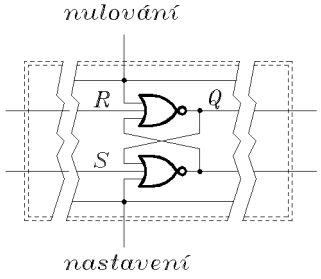
4. T [Trigger]



T	Q^{t+1}
1	$\overline{Q^t}$
0	Q^t

aynchronní nastavení a nulování

synchronních klopných obvodů



návrh synchronních sekvenčních obvodů

Postupuje se analogicky jako při návrhu asynchronních sekvenčních obvodů (viz příklad), **ale:**

- **fundamentální režim se nevyžaduje**
- **hazardy** vůbec **nevadí**
- pojem **stabilní stav ztrací smysl**
- **klopné obvody**
  - **typu D: klopné obvody se „prostě" vloží do zpětných vazeb (schéma ① nebo ②)**
  - **ostatní: výstupy K2 (viz ⑤ a ⑥) — budicí funkce (např. J a K) klopných obvodů**

Pozn.: i u asynchronních sekv. obvodů ③ a ④: výstupy K2 — budicí funkce (R a S)

Rekapitulace:

logické obvody { **kombinační**; **sekvenční** { **asynchronní**; **synchronní** }

Konečný automat (KA) : matem. model SLO

- šestice : (X, Y, Q, Q_0, δ, λ)

X .... množina možných kombinací hodnot vstup. proměnných KA; př: 3 vstup. prom. ⇒ X obs. 2^3 = 8 kombinací

Y .... množina možných kombinací hodnot výstupních proměnných KA

Q .... množina možných kombinací hodnot vnitřních proměnných KA (množina stavů)

Q_0 .... počáteční stav (kombinace hodnot vnitřních proměnných KA v počáteč. stavu)

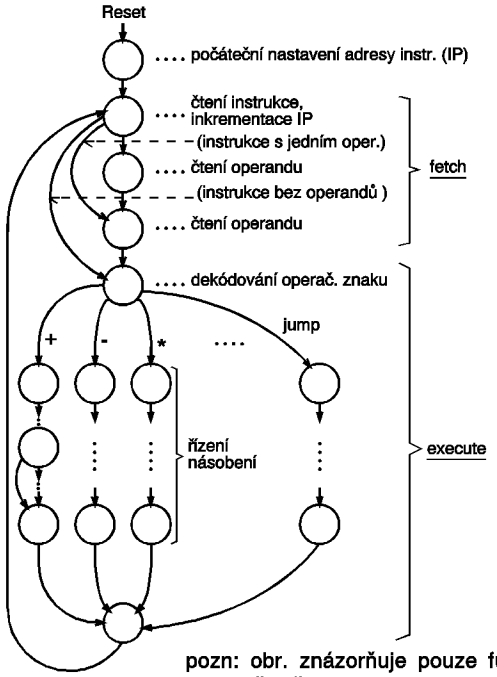
δ .... stavově přechodová funkce :
$\delta : X \times Q \rightarrow Q$ ....definuje příští vnitřní stav KA

λ .... výstupní funkce :
λ: a) $X \times Q \rightarrow Y$ .... typ Mealy
b) $Q \longrightarrow Y$ .... typ Moore

Formy popisu KA : - graf přechodů
- tabulky pro δ a λ

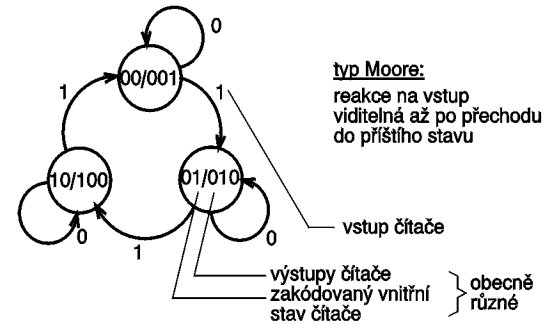
Ukázky automatů

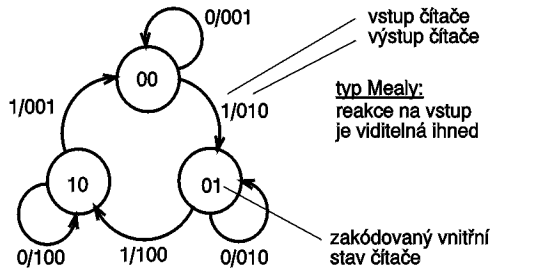
1)řadič číslicového počítače



pozn: obr. znázorňuje pouze funkci řadiče, nejsou zde uvedeny jeho vstupy ani výstupy

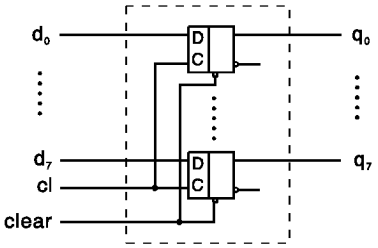
2) Čítač mod 3: tři výstupy - 001,010,100 ⇒ tři výstupní proměnné
tři vnitřní stavy - 00,01,10 ⇒ dva klopné obvody





pozn.: Výstup obou čítačů je v kódu 1 ze 3.
Obvykle však bývá kódován dvojkově (00,01,10).
Pak lze v našem případě u typu Moore ztotožnit vnitřní a výstupní proměnné.

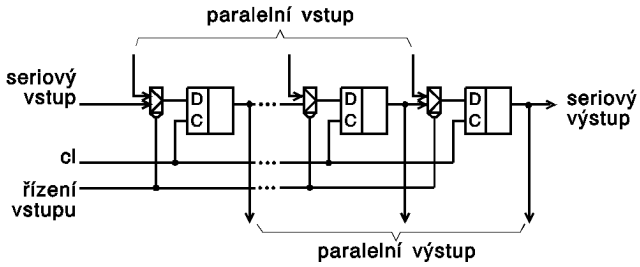
a) paralelní registry (dočasná paměť)



b) seriové registry (převodníky)

-serioparalelní: seriový vstup
paralelní výstup

-paralelněseriové: paralelní vstup
seriový výstup



Součástková základna číslicové techniky

SSI (small scale integration)..... do 10^2 hradel/čip
MSI (medium scale integration).... 10^2 až 10^3 hradel/čip
LSI (large scale integration)..... 10^3 až 10^4 hradel/čip
VLSI (very LSI) ..... nad 10^4 hradel/čip

SSI - elementární logické členy:
- and, nand, or, nor, xor
- paměťové obvody D, JK, ...

MSI - často používané universální obvody:
- multiplexory: 2 x 1, 4 x 1, 2 x 2, 16 x 1, ...
- čítače: mod2, mod5, mod16
- registry

LSI a VLSI :

- speciální: mikroprocesory
  mikroprocesorové řezy
  paměti (ROM, PROM, EPROM, RAM, ...)
  různé podpůrné obvody

- universální (s programovatelnou strukturou)
  výhody vzhledem k SSI a MSI:
  - vyšší rychlost
  - vyšší spolehlivost
  - nižší náklady na montáž
  - nižší spotřeba elektr. energie

implementace:

- plně zákaznický návrh: veškeré propojení je plně realiz. již při výrobě ⇒ složitá spolupráce výrobní závod -uživatel
- polozákaznický návrh:
  výroba - připraví polotovar
  uživatel - propojení požadované struktury
  - PLA (programmable logic array)
  - PAL (programmable and logic)
  - FPGA (field programmable gate arrays)
  - FPLD (field programmable logic devices)

- technologie spojovacích bodů:

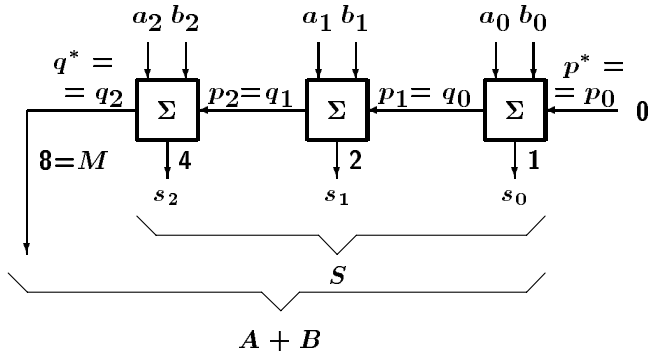
  - tavné bipolární pojistky ⇒ jednorázové naprogramování

  - elektronické spínače řízené:
    a) transistory CMOS s izolovaným hradlem ⇒ vymazání světlem

    b) bistabilními klopnými obvody ⇒ nutnost konfigurace po každém zapnutí elektrické energie programem z pevné paměti (EPROM). Př. XILINX

Sčítání ve dvojkové soustavě

$A \sim a_2 a_1 a_0$
$B \sim b_2 b_1 b_0$
} **nezáporná čísla** (bez znaménka)



p_i ... **přenos do řádu** i **[carry]**

q_i ... **přenos z řádu** i

$M = 8 = 2^3$ ... **modul sčítačky**

$$S = \begin{cases} A + B, & \text{je-li } q^* = 0 \\ A + B - M, & \text{je-li } q^* = 1 \end{cases}$$

$q^* = 1 \iff$ **přeplnění** **[overflow]**

$M = 1000_2 = (111 + 1)_2$

Odčítání ve dvojkové soustavě

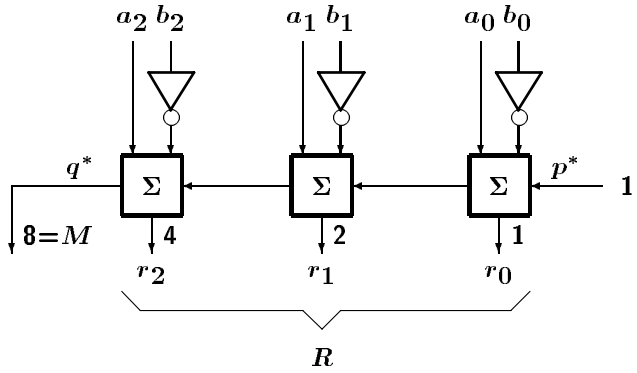
$A \sim a_2 a_1 a_0$
$B \sim b_2 b_1 b_0$
} **nezáporná čísla** (bez znaménka)

Pozorování: $B = 101 \quad \overline{B} = 010$

$B + \overline{B} = 111 = 1000 - 1 = M - 1$

$-B = \overline{B} + 1 - M$

$$A - B = A + \overline{B} + 1 - M$$



$R = A - B$, **je-li** $q^* = 1$

$p^* = 1$ ... **„horká jednička“** **[hot one]**

sčítání ... **přenos** **[carry]**

odčítání ... **výpůjčka** **[borrow]**

v^* ... **výpůjčka pro nejvyšší řád**

$$v^* = 1 \iff \text{záporný výsledek}$$

$X = A - B$

$A + \overline{B} + 1 = M + (A - B) = M + X$

$X \geq 0 \quad \Rightarrow \quad q^* = 1 \quad \Rightarrow \quad R = X$

$X < 0 \quad \Rightarrow \quad q^* = 0 \quad \Rightarrow \quad R = M + X$

$$\begin{array}{lllll} q^* = 1 & \iff & v^* = 0 & \iff & A - B \geq 0 \\ q^* = 0 & \iff & v^* = 1 & \iff & A - B < 0 \end{array}$$

$$v^* = \overline{q^*}$$

Kódy pro záporná čísla

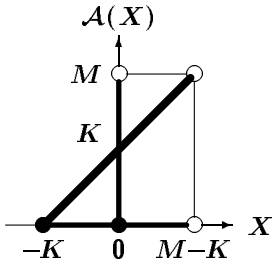
(přesněji: **pro záporná i nezáporná čísla,**
tzn. **pro „čísla se znaménkem“**)

Ve standardní polyadické soustavě (dvojkové, desítkové a pod.) **nelze zapsat záporná čísla.**

Proto se používají *kódy*:

1. **přímý** **[sign–magnitude]**
2. **doplňkový (dvojkový doplněk)** **[2's complement]**
3. **aditivní (s posunutou nulou)** **[biased]**
4. **inverzní** **[1's complement]**

Př.: aditivní kód



$$\mathcal{A}(X) = X + K$$

K — **„vhodná“ konstanta**

$$-K \leq X < M - K$$

M — **tzv.** *modul řádové mřížky*

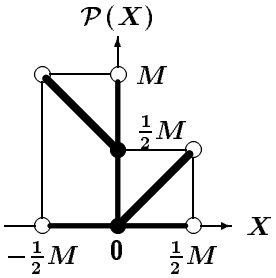
modul řádové mřížky — **zobecnění modulu sčítačky i na jiné aplikace** (např. obsah registrů)

Přímý kód

znaménkový bit: $0 \sim +$, $1 \sim -$

Př.: $M = 1000_2 \implies$ 3bitová čísla

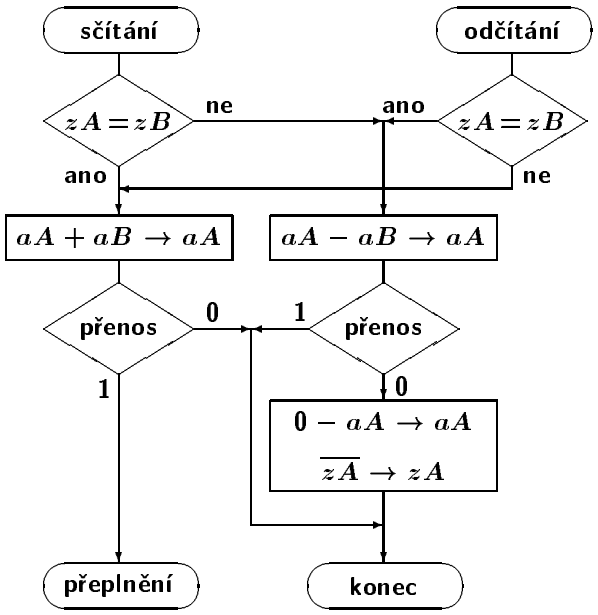
X	$\mathcal{P}(X)$	
$+0$	0 0 0	← kladná nula
$+1$	0 0 1	
$+2$	0 1 0	
$+3$	0 1 1	
-0	1 0 0	← záporná nula
-1	1 0 1	
-2	1 1 0	
-3	1 1 1	



$|X| < \frac{1}{2}M$

Sčítání a odčítání v přímém kódu

$A \sim (zA, aA)$, $B \sim (zB, aB)$ $\longrightarrow$? $\begin{cases} A+B \to A \\ A-B \to A \end{cases}$



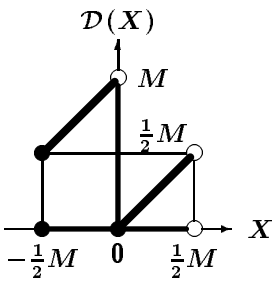
Doplňkový kód

$$\mathcal{D}(X) = \begin{cases} X, & \text{je-li } X \geq 0 \\ M + X, & \text{je-li } X < 0 \end{cases}$$

(srov. odčítání nezáporných čísel: $R = \cdots$)

Př.: $M = 1000_2 \implies$ 3bitová čísla

X	$\mathcal{D}(X)$	
0	0 0 0	
1	0 0 1	
2	0 1 0	
3	0 1 1	
-4	1 0 0	← !
-3	1 0 1	
-2	1 1 0	
-1	1 1 1	



Znaménko je určeno *prvním bitem* (zleva); tento bit je však organickou částí obrazu (na rozdíl od přímého kódu)

$$-\tfrac{1}{2}M \leq X < \tfrac{1}{2}M$$

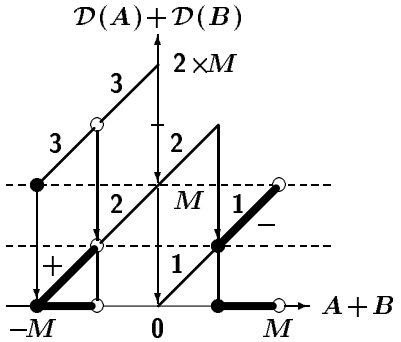
Sčítání v doplňkovém kódu

		$\mathcal{D}(A)+\mathcal{D}(B)$	$\mathcal{D}(A+B)$
1	$A \geq 0$ $B \geq 0$	$A+B$	$A+B$
2	$A \geq 0$ $B < 0$ $A < 0$ $B \geq 0$	$A+B+M$	$A+B$ $A+B+M$
3	$A < 0$ $B < 0$	$A+B+M+M$	$A+B+M$

$$\mathcal{D}(A+B) = \begin{cases} \mathcal{D}(A) + \mathcal{D}(B) \\ \mathcal{D}(A) + \mathcal{D}(B) - M \end{cases}$$

Sečtou se obrazy a ignoruje se přenos.

? přeplnění ? (přeplnění ≠ přenos)



přeplnění: $+ + \to -$, $- + \to +$

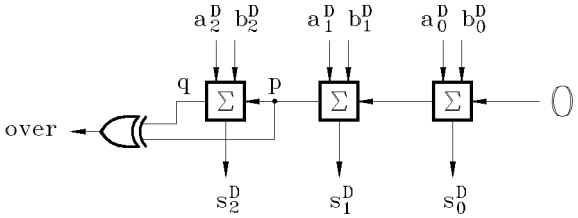
Sčítání v doplňkovém kódu 2

přeplnění: $\boxed{+} + \boxed{+} \longrightarrow \boxed{-}$; $\boxed{-} + \boxed{-} \longrightarrow \boxed{+}$

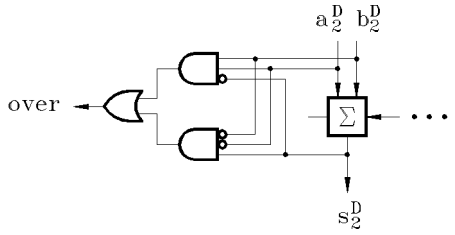
pravdivostní tabulka úplné sčítačky:

$a=0 \quad b=0 \quad s=1$
$a=1 \quad b=1 \quad s=0$
$\Big\} \iff p \neq q$

přeplnění: **v nejvyšším řádu** $p \neq q$



jiná alternativa:



Odčítání v doplňkovém kódu

$$\boxed{A - B = A + (-B)}$$

$\mathcal{D}(B) + \mathcal{D}(-B) = B + (-B) + M = M$
$\mathcal{D}(-B) = M - \mathcal{D}(B)$

srovnej „Odčítání nezáporných čísel“:

$\mathcal{D}(-B) = \overline{\mathcal{D}(B)} + 1$

Odčítačka: **stejná jako u nezáporných čísel, ale *detekce přeplnění* jako u sčítačky v doplňkovém kódu.**

Doplňkový kód v desítkové soustavě

Př.: 3místná desítková čísla → $M = 1000_{10}$

X	$\mathcal{D}(X)$	X	$\mathcal{D}(X)$
0	000	−500	500
1	001	−499	501
⋮	⋮	⋮	⋮
499	499	−1	999

Znaménko **je určeno *první číslicí* (zleva):**

0 ÷ 4	...	+
5 ÷ 9	...	−

Př.: $\mathcal{D}(X) + \mathcal{D}(-X) = 1000 = 999 + 1$
$\mathcal{D}(-X) = 999 - \mathcal{D}(X) + 1$
označme: $\widetilde{a} = 9 - a$
$\mathcal{D}(X) = 499 \Longrightarrow \mathcal{D}(-X) = \widetilde{\widetilde{499}} + 1$
$\mathcal{D}(-X) = 500+1 = 501$

tzv. **desítkový doplněk** **[10's complement]**

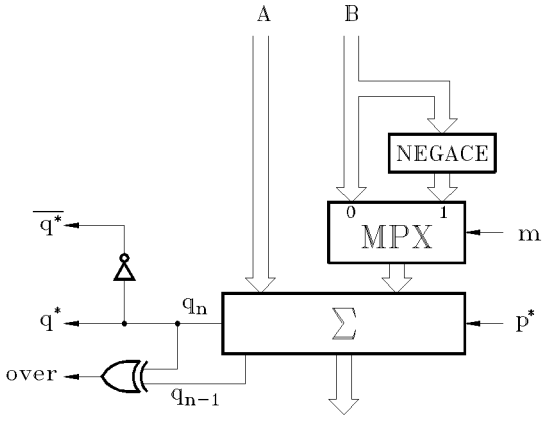
ve dvojkové soustavě: **dvojkový doplněk**

Pozn.: existuje také **jedničkový a devítkový doplněk**
(týká se tzv. inverzního kódu)

Sčítání a odčítání — 80x86

(základní princip)

- **nezáporná čísla (bez znaménka) —** ***byte, word***
- **doplňkový kód —** ***shortint, integer***



	m	p^*	CF	OF
ADD	0	0	q^*	*over*
ADC	0	CF	q^*	*over*
SUB	1	1	$\overline{q^*}$	*over*
SBB	1	$\overline{CF}$	$\overline{q^*}$	*over*

Příznaky a vícenásobná aritmetika

(např. ***longint***)

- ***nižší řády:***
  - OF — nemá význam
  - CF — přenos / výpůjčka pro další řády
    - ADD → ADC → ... → ADC
    - SUB → SBB → ... → SBB
- ***nejvyšší řád:***

	CF	*OF*
byte, word	přeplnění	—
shortint, integer	—	přeplnění

Rozšíření řádové mřížky

v doplňkovém kódu

$$\mathcal{D}(X) = \begin{cases} X \\ X + M \end{cases} \qquad \mathcal{D}'(X) = \begin{cases} X \\ X + N \end{cases}$$

M a N ... moduly řádových mřížek: $M < N$

$$\mathcal{D}'(X) = \mathcal{D}(X) + \begin{cases} 0 & \text{pro } X \geq 0 \\ N - M & \text{pro } X < 0 \end{cases}$$

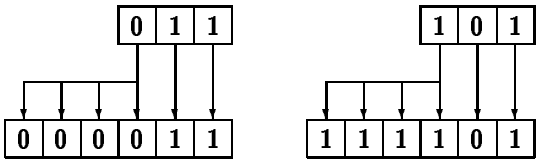
$M = 2^m, \quad N = 2^n \implies N - M = 2^n - 2^m$

$N - M = (2^{n-m} - 1) \cdot 2^m$

Př.: $M = 1000_2$ ... 3bitová čísla

$N = 1000000_2$ ... 6bitová čísla

$N - M = 111000_2$



tzv. ***znaménkové rozšíření*** [sign extension]

Posuvy

Př.: 001,01 × 100 = 101,00

101,00 : 100 = 001,01

obecně:

$A \times 2^k = A \ll k$	posuv vlevo	[shift left]
$A : 2^k = A \gg k$	posuv vpravo	[shift right]

$$A \gg k = A \ll -k$$

problémy: (a) Vypadávají některé bity (číslice).

(b) Co vložit na uvolněná místa ?

logický posuv

(a) ignoruje se

(b) nuly

aritmetický posuv

(a) **detekce přeplnění** (popř. ztráty přesnosti)

(b) **takové bity** (číslice), **aby výsledek odpovídal** (pokud možno) **příslušnému násobení nebo dělení**

cyklický (kruhový) posuv [rotation]

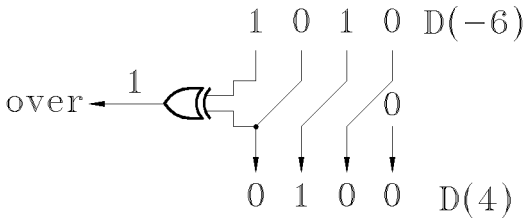
Vypadávající číslice se ve stejném pořadí vkládají na uvolněná místa.

Aritmetický posuv

v doplňkovém kódu

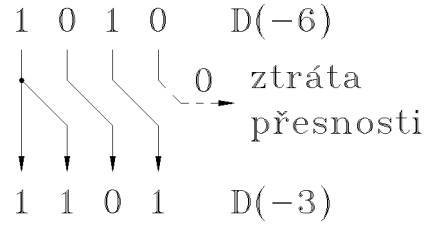
$$A \ll 1 \equiv A \times 2 = A + A$$

stejní sčítanci $\implies$ stejné znaménko



$A \gg 1$ — opak $A \ll 1$

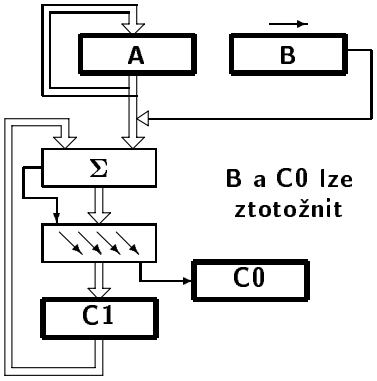
bez přeplnění



Násobení

Př.:

A	×	B	=	C
1 1 1	×	1 0 1	=	1 0 0 *0 1 1*
0 0 0		↓ ↓ ↓		C1 C0
1 1 1		←		
0 1 1 *1*				
0 0 0		←		
0 0 1 *1*				
1 1 1		←		
1 0 0 *0*				



Dělení

0 , 1 0 1 : 0 , 1 1 0
↓

	1 0 1 : 1 1 0 = 0 , 1 1 0		podíl
−	1 1 0		
	−1	→ 0 ,	
+	1 1 0		návrat
	1 0 1 *0*		
−	1 1 0		
	1 0 0 *0*	→ 1	
−	1 1 0		
	1 0 *0*	→ 1	
−	1 1 0		
	−1 0	→ 0	
+	1 1 0		návrat
	1 0 0		zbytek

návrat (přes nulu) — restaurace nezáporného zbytku: pomocná operace obnovující původní dílčí zbytek pro následující odčítání

Dělení bez návratu přes nulu

(bez restaurace nezáporného zbytku)

návrat = přičtení jistého čísla X → $+\ X$

následuje odečtení čísla $X \gg 1$ → $\dfrac{-X/2}{+X/2}$

	1 0 1 : 1 1 0 = 0 , 1 1 0		podíl
−	1 1 0		
	−1 *0*	→ 0 ,	
+	1 1 0		
	1 0 0 *0*	→ 1	
−	1 1 0		
	1 0 *0*	→ 1	
−	1 1 0		
	−1 0	→ 0	
+	1 1 0		návrat
	1 0 0		zbytek

návrat — pouze *v posledním kroku*, je-li zbytek záporný, a to *jenom* v případě, že *třeba získat* také *správný zbytek*.

Desítkové kódy

k bitů/číslice ... k bitový kód

$2^k \geq 10 \implies \boxed{k \geq 4}$

10 bitový kód: 1 z 10

5 bitové kódy: bezpečnostní kódy (2 z 5, 3 z 5 a zejm. Brownův kód $3N+2$)

4 bitové kódy: $\binom{16}{10} \cdot 10! = 29\,059\,430\,400$ kódů

	BCD	+3	2 4 2 1	8,4,-2,-1
0	0 0 0 0	0 0 1 1	0 0 0 0	0 0 0 0
1	0 0 0 1	0 1 0 0	0 0 0 1	0 1 1 1
2	0 0 1 0	0 1 0 1	0 0 1 0	0 1 1 0
3	0 0 1 1	0 1 1 0	0 0 1 1	0 1 0 1
4	0 1 0 0	0 1 1 1	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 0 0	1 0 1 1	1 0 1 1
6	0 1 1 0	1 0 0 1	1 1 0 0	1 0 1 0
7	0 1 1 1	1 0 1 0	1 1 0 1	1 0 0 1
8	1 0 0 0	1 0 1 1	1 1 1 0	1 0 0 0
9	1 0 0 1	1 1 0 0	1 1 1 1	1 1 1 1

- váhové kódy, např. BCD (8,4,2,1); 2,4,2,1; aj.
- Stibitzův (+3, též XS3), Aikenův (2,4,2,1) a Rubinoffův kód (8,4,-2,-1) — doplněk do 9 se získá inverzí všech bitů (sčítačka → odčítačka)

Kód BCD — sčítání

Jednomístná desítková sčítačka

a, b číslice sčítanců p přenos z nižšího řádu

s číslice součtu q přenos do vyššího řádu

Označme $y = a+b+p$; pak $\boxed{y \geq 10 \iff q = 1}$

? $y \to s$?

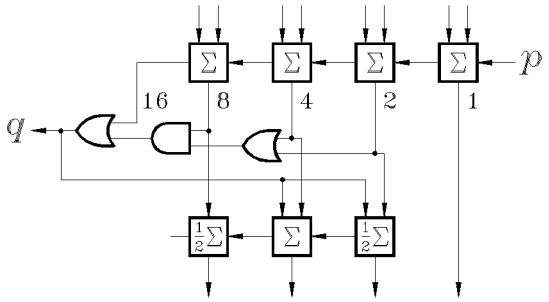
$$\text{korekce} = \begin{cases} -10 & \text{pro } q = 1 \\ 0 & \text{pro } q = 0 \end{cases}$$

$$\boxed{-10 = -16 + 6}$$

Př.: 3+2+0 → 0 5 ... 0 0101

5+6+1 → 1 2 ... 0 1100 + 0110 ... 1 0010

8+9+0 → 1 7 ... 1 0001 + 0110 ... 1 0111



Kód BCD — sčítání — 80x86

ADD/ADC: AF [Auxiliary Flag] přenos z řádu 3

CF [Carry Flag] přenos z řádu 7

Po **ADD AL,**··· nebo po **ADC AL,**··· lze použít

DAA: **korekce +0 nebo +6** (obě číslice)

vytvoření CF (desítkový přenos)

Př.:

8 b.	8 b.		8 b.	8 b.		8 b.	8 b.	8 b.
x1	x0	+	y1	y0	→	s2	s1	s0
4 9	2 2	+	6 7	1 8	=	0 1	1 6	4 0

```
mov  AL,x0   ; 22h → AL
ADD  AL,y0   ; 22h+18h= 3Ah → AL
DAA          ; korekce AL: +06h  (040)
mov  s0,AL   ; 40h → s0

mov  AL,x1   ; 49h → AL
ADC  AL,y1   ; 49h+67h+0= B0h → AL
DAA          ; korekce AL: +66h  (116h)
mov  s1,AL   ; 16h → s1

mov  AL,0    ; 0 → AL
ADC  AL,0    ; 0+0+1= 01h → AL
mov  s2,AL   ; 01h → s2
```


Pohyblivá řádová čárka

Dosud uvažovaná reprezentace čísel:

Pevná řádová čárka [fix point]

Nejobvyklejší typy řádových mřížek:

1. typ: modul $M = 2$ ... abs. hodnota ≤ 1
2. typ: celá čísla

Př.: 1. typ podle obrázku

přímý kód $\Rightarrow -1 < A < 1$

$C = A + B = 1 + 2 = ?$

A ani B není zobrazitelné !?!

měřítko $S = 1{:}16 = 1{:}10000_2$ [Scale]

$A^* = A \cdot S = 0{,}0001_2$

$B^* = B \cdot S = 0{,}0010_2$

$C^* = C \cdot S = A^* + B^* = 0{,}0011_2$

$C = C^*/S = 11_2 = 3$

Pohyblivá řádová čárka

Měřítko:

$$\boxed{\begin{aligned} A + B &= (A^* + B^*) \,/\, S \\ A - B &= (A^* - B^*) \,/\, S \\ A * B &= (A^* * B^*) \,/\, S^2 \\ A \,/\, B &= (A^* \,/\, B^*) \end{aligned}}$$

Vhodné měřítko (!?!)

lze bez větších problémů předem stanovit pro výpočty z oblasti tzv. zpracování hromadných dat (účetnictví, sklady apod.) — variabilita potřebných měřítek je „minimální". **Vhodné měřítko však prakticky nelze předem stanovit pro tzv. vědecko-technické výpočty** — opačná situace.

Řešení ???

↓

Ať se snaží počítač — má na to HW i SW !!!

↓

reprezentace čísel v pohyblivé řádové čárce

Pohyblivá řádová čárka
[floating point]

Obrazem čísla A je taková uspořádaná dvojice

$$\boxed{(m, e)},$$

že platí $A = m \cdot z^{\,e}$, kde z je základ soustavy (dále budeme uvažovat $z = 2$); tedy:

$$\boxed{A \;=\; m \cdot 2^e}$$

m **mantisa** ... 1. typ řádové mřížky (velmi často)
e **exponent** ... 2. typ řádové mřížky (vždy)
(též **charakteristika**)

Příklad: mantisa i exponent v doplňkovém kódu
$A = -13 = -1101_2 = (-0{,}1101 \cdot 10^{100})_2$
$D(m) = D(-0{,}1101) = 1{,}0011\,0 \cdots 0$
$D'(e) = D'(100) = 0100$

0	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0
exponent				mantisa											

4980_{16}

Operace
zjednodušeno

sčítání/odčítání

1. upravit operandy tak, aby měly stejný exponent
2. sečíst/odečíst mantisy

srov.: $500 - 3 = 0{,}5 \cdot 10^3 - 0{,}3 \cdot 10^1 =$
$= 0{,}5 \cdot 10^3 - 0{,}003 \cdot 10^3 =$
$= 0{,}497 \cdot 10^3 =$
$= 497$

násobení/dělení

- vynásobit/podělit mantisy
- sečíst/odečíst exponenty

srov.: $500 \times 3 = 0{,}5 \cdot 10^3 \times 0{,}3 \cdot 10^1 =$
$= 0{,}15 \cdot 10^4 = 1500$

---

výsledný exponent:

$>$ největší možný $\Rightarrow$ *přeplnění* [overflow]
$<$ nejmenší možný $\Rightarrow$ *nenaplnění* [underflow]

srov.: $0{,}8 \cdot 10^{-99} = 0{,}000 \cdots 008 \to 0$

Normalizace

Témuž číslu může příslušet několik reprezentací.
Srov.: $8 = 0{,}8 \cdot 10^1 = 0{,}08 \cdot 10^2 = \cdots$

> Číslo má normalizovaný tvar, není-li možné posunout mantisu doleva.

Příklad: mantisa i exponent v doplňkovém kódu
$A = -13 = -1101_2 = (-0{,}1101 \cdot 10^{100})_2 =$
$= (-0{,}01101 \cdot 10^{101})_2$

$D(m)$	$D'(e)$	
$1{,}001\,100 \cdots 0$	100	normalizovaný tvar
$1{,}100\,110 \cdots 0$	101	nenormalizovaný tvar

> normalizovaný tvar $\Rightarrow$ jednodušší algoritmy

Na konci každé operace je třeba provést tzv.
normalizaci výsledku !!!

Skrytá jednička

Nenulová mantisa:
- přímý kód
- normalizovaný tvar

$\Downarrow$

Bit v nejvyšším řádu je vždy roven **1**.
Tento bit lze ze zápisu nenulových čísel vypustit.

$\Downarrow$

skrytá jednička [hidden one]

Je použita např. v reprezentaci podle
ANSI/IEEE Std 754 – 1985

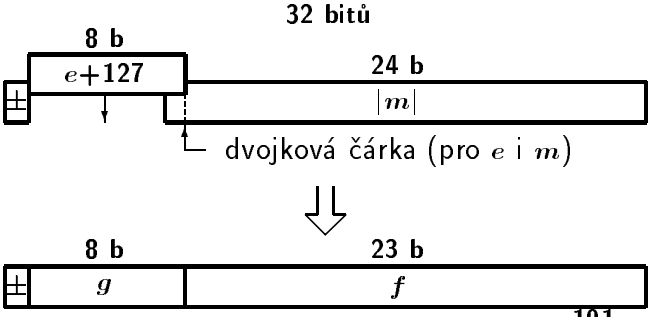
základní formáty:

	znaménko	exponent	mantisa
32 b	1 b	8 b	23 (24) b
64 b	1 b	11 b	52 (53) b

V obou formátech je pro mantisu (včetně znaménka) použit modul $\boxed{M = 4}$, tzn. $\boxed{|m| < 2}$.

ANSI/IEEE Std 754 – 1985

32 bitů



Př.: $-58_{10} = -11\,1010_2 = (-1{,}11010 \times 10^{101})_2$

$|m| = 1{,}11010 \quad \Rightarrow \quad f = 110100\ldots0$

$e = 101 \quad \Rightarrow \quad g = 1000\,0100$

1 1000 0100 1101 0000 0 ... 00

| C | 2 | 6 | 8 | 0 0 0 0 |

little endian: 00 00 68 C2

***konvence*:** **00000000 je obrazem nuly !!!**

(neúplné) $g = 255$ **a** $f = 0$: ∞

$g = 255$ **a** $f \neq 0$: **NaN**

NaN [Not a Number] — např. výsledek dělení 0:0

Alfanumerické kódy

(Abecedně-číslicové kódy)

kódy pro zobrazení textů

tj. písmen, číslic (cifer) a dalších znaků

26 písmen anglické (latinské) **abecedy**

$\times$ 2 (velká/malá)

10 číslic

další znaky (mezera, čárka, tečka, ..., plus, ...)

⇓

minimálně 6 bitů/znak (raději aspoň 7 bitů/znak)

5bitový kód: CCITT 2 (ITA 2, MTA 2)

dvojí interpretace znaků „přepínaná“ speciálními znaky

8bitový kód: EBCDIC (DKOI)

[Extended Binary-Coded-Decimal Interchange Code (dvojičnyj kod dlja obměny i obrabotky informacij)]

mezera	**40**		
'a' ... 'i'	**81 ... 89**	**'A' ... 'I'**	**C1 ... C9**
'j' ... 'r'	**91 ... 99**	**'J' ... 'R'**	**D1 ... D9**
's' ... 'z'	**A2 ... A9**	**'S' ... 'Z'**	**E2 ... E9**
		'0' ... '9'	**F0 ... F9**

Alfanumerické kódy - II.

7bitový kód: ASCII , též **ASCII-7** , popř. USASCII **(CCITT 5, ISO-7, KOI-7)**

[American Standard Code for Information Interchange, popř. USA Standard Code for Information Interchange (Commité consultatif international télégraphique et téléphonique, International Standard Organization, kod dlja obměny i obrabotky informacij)]

8bitový kód: **ASCII-8** (popř. ISO-8, KOI-8)

00 ... 7F — ASCII-7 ■

80 ... FF — tzv. „národní abecedy“

„české abecedy“ (pouze některé z používaných): ■

KOI ◇ **KOI-8 čs** (ČSN 36 9103)

Kam ◇ **kód bratří Kamenických** (KEYBCZ, CP895)
Ælu*f*ouçkÿ kûñ úpêl âábelské ódy.
[**Žluťoučký kůň úpěl ďábelské ódy.**]

852 ◇ **IBM – page 852** (Latin 2)

ISO ◇ **ISO 8859–2** (ČSN ISO/IEC 8859–2 36 9111)

1250 ◇ **windows 1250**

MAC ◇ **apple–ce**

uni ◇ **Unicode**

Pozn.: Lze „vymyslet“ cca $n \times 11 \cdot 10^{90}$ dalších „abeced“!

Kód ASCII-7 (šestnáctkový zápis)

	0.	1.	2.	3.	4.	5.	6.	7.
0	NUL	DLE		0	@	P	‘	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	”	2	B	R	b	r
3	ETX	DC3	#	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	EH	)	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

	KOI	Kam	852	ISO	1250	MAC	uni
á	C1	A0	A0	E1	E1	87	00E1
ä	D1	84	84	E4	E4	8A	00E4
č	C3	87	9F	E8	E8	8B	010D
ď	C4	83	D4	EF	EF	93	010F
é	D7	82	82	E9	E9	8E	00E9
ě	C5	88	D8	EC	EC	9E	011B
í	C9	A1	A1	ED	ED	92	00ED
Í	CB	8D	92	E5	E5	BE	013A
ľ	CC	8C	96	B5	BE	BC	013E
ň	CE	A4	E5	F2	F2	CB	0148
ó	CF	A2	A2	F3	F3	97	00F3
ô	D0	93	93	F4	F4	99	00F4
ö	CD	94	94	F6	F6	9A	00F6
ŕ	C6	AA	EA	E0	E0	DA	0155
ř	D2	A9	FD	F8	F8	DE	0159
š	D3	A8	E7	B9	9A	E4	0161
ť	D4	9F	9C	BB	9D	E9	0165
ú	D5	A3	A3	FA	FA	9C	00FA
ů	CA	96	85	F9	F9	F3	016F
ü	C8	81	81	FC	FC	9F	00FC
ý	D9	98	EC	FD	FD	F9	00FD
ž	DA	91	A7	BE	9E	EC	017E

	KOI	Kam	852	ISO	1250	MAC	uni
Á	E1	8F	B5	C1	C1	E7	00C1
Ä	F1	8E	8E	C4	C4	80	00C4
Č	E3	80	AC	C8	C8	89	010C
Ď	E4	85	D2	CF	CF	91	010E
É	F7	90	90	C9	C9	83	00C9
Ě	E5	89	B7	CC	CC	9D	011A
Í	E9	8B	D6	CD	CD	EA	00CD
Ĺ	EB	8A	91	C5	C5	BD	0139
Ľ	EC	9C	95	A5	BC	BB	013D
Ň	EE	A5	D5	D2	D2	C5	0147
Ó	EF	95	E0	D3	D3	EE	00D3
Ô	F0	A7	E2	D4	D4	EF	00D4
Ö	ED	99	99	D6	D6	85	00D6
Ŕ	E6	AB	E8	C0	C0	D9	0154
Ř	F2	9E	FC	D8	D8	DB	0158
Š	F3	9B	E6	A9	8A	E1	0160
Ť	F4	86	9B	AB	8D	E8	0164
Ú	F5	97	E9	DA	DA	F2	00DA
Ů	EA	A6	DE	D9	D9	F1	016E
Ü	E8	9A	9A	DC	DC	86	00DC
Ý	F9	9D	ED	DD	DD	F8	00DD
Ž	FA	92	A6	AE	8E	EB	017D

TYPY PAMĚTÍ A JEJICH KONSTRUKČNÍ PRINCIPY

Paměť: zařízení pro uchování programů a dat v počítači
Základní funkce: zápis a čtení (ex. paměti s pevným obsahem, které lze pouze číst)

ROZDĚLENÍ PAMĚTÍ

-podle použití v počítači

např. -operační (hlavní) paměť
-vnější paměť
(existují i další - viz příští přednáška)

-podle fyzikálního principu

-polovodičové
-magnetické
-optické

-podle způsobu výběru datových položek

-s adresovým výběrem (adresové)
-s postupným výběrem (seriové)
-asociativní - výběr podle části uložené informace, tzv. klíče (CAM)
-zásobník - LIFO
-fronta - FIFO

-podle možností a způsobu změny uložené informace

-Paměti pro čtení a zápis *RWM*
-obsah lze libovolně přepisovat *RAM*
SRAM -statická
DRAM -dynamická

Paměti RAM jsou energeticky závislé (volatilní). Uložená informace zanikne po vypnutí napájení.

-Paměti permanentní
-Obsah určen buď při výrobě *ROM*
-nebo jednorázovým naprogramováním *PROM*

-paměti semipermanentní
-obsah určen naprogramováním, lze je vymazat a přeprogramovat *EPROM*
EEPROM

Permanentní a semipermanentní paměti jsou energeticky nezávislé. Informace zůstane uložena i po vypnutí napájení dokud není přepsána.

V této a příští přednášce se budeme zabývat pouze polovodičovými paměťmi s adresovým výběrem, které se používají pro konstrukci hlavní paměti počítače. Vnější paměti patří mezi periferní (přídavná) zařízení - viz přednáška 11.

ZÁKLADNÍ POJMY

Paměťová buňka - základní stavební blok paměti, sloužící k záznamu jednoho bitu.

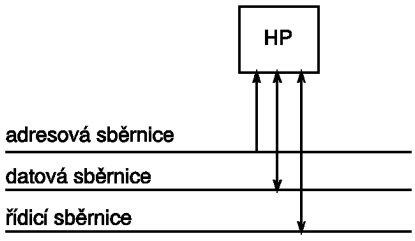
Paměťové místo - skupina paměťových buněk, které lze současně zapisovat nebo číst (jejich počet je dán šířkou slova paměti).

Položka - obsah paměťového místa.

Adresa - číselné označení (index) paměťového místa jimž lze vybírat jednotlivé položky. Počet položek se nazývá kapacita paměti.

Paměťová matice - skupina paměťových míst uspořádaná tak, že je lze vybírat adresou.

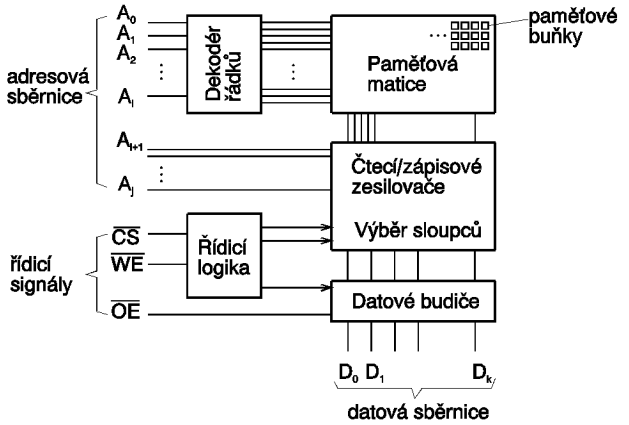
Hlavní paměť komunikuje s ostatními jednotkami počítače prostřednictvím adresové, datové a řídicí sběrnice (opakování z 1. přednášky).



Hlavní paměť je sestavena z paměťových obvodů (v dnešní době zpravidla integrované obvody vyrobené technologií CMOS).

VNITŘNÍ STRUKTURA PAMĚŤOVÝCH OBVODŮ A JEJICH VLASTNOSTI

BLOKOVÉ SCHÉMA TYPICKÉHO PAMĚŤOVÉHO OBVODU



Kapacita paměťového obvodu je dána šířkou jeho adresové a datové sběrnice. V tomto případě 2^{i+1} slov po k+1 bitech.

Dekodér řádků: dekóduje binární kód na kód 1 z n (přesněji 1 z 2^{i+1})
Obvod výběru sloupců: jeden multiplexer pro každý datový bit
Paměťová buňka: např. bistabilní klopný obvod u statické paměti RAM - podrobněji viz dále
Řídicí signály:
$\overline{OE}$ - aktivace výstupních třístavových budičů datové sběrnice
$\overline{WE}$ - povolení zápisu (u permanentních pamětí odpadá)
$\overline{CS}$ - výběr čipu - aktivní úroveň (Ø) podmiňuje provedení zápisu nebo čtení

TYPICKÁ ORGANIZACE PAMĚŤOVÝCH OBVODŮ

(Pozn. n=j+1)

2^nx8 ... obvody SRAM, ROM, EPROM, některé EEPROM, FLASH
např. 32kx8, 64kx8, 128kx8, ... 512kx8

2^nx1 ... obvody DRAM
např. 256kx1, 1Mx1, ... 16Mx1

2^nx9, 2^nx36 ... paměťové moduly DRAM
např. 256kx9, 1Mx9, ... 4Mx36

.
.
.

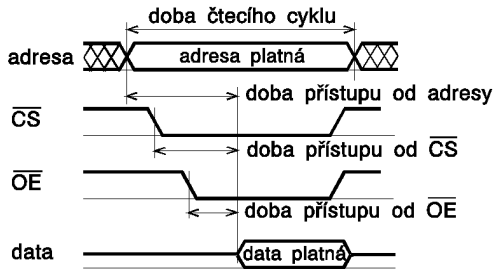
16MBx36 ... 60ns
parita:
4x8+4=36

SIMM - Single In-line Memory Module
DIMM - Dual In-line Memory Module

CHOVÁNÍ PAMĚŤOVÝCH OBVODŮ A JEJICH PARAMETRY

Pro bezchybné provedení čtení a zápisu je nutné dodržet minimální (výjimečně i maximální) zpoždění mezi aktivací jednotlivých signálů a dalšími událostmi - např. čtení z datové sběrnice

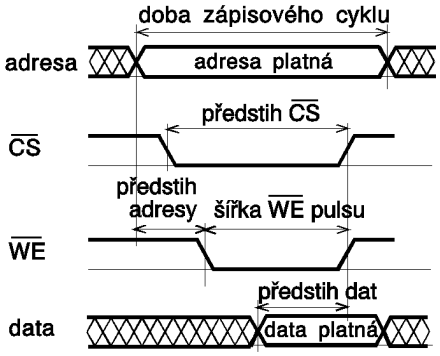
Typický průběh operace čtení ze statické paměti RAM



Podobný časový diagram má i většina permanentních a semipermanentních pamětí.

Typické parametry moderních pamětí SRAM:
doba přístupu od adresy ... 15-70ns=100%
min. doba čtecího cyklu ... ≥ 100%
doba přístupu od CS ..... ≤ 100%
doba přístupu od OE ..... cca 50%

Typický průběh operace zápisu do statické paměti RAM

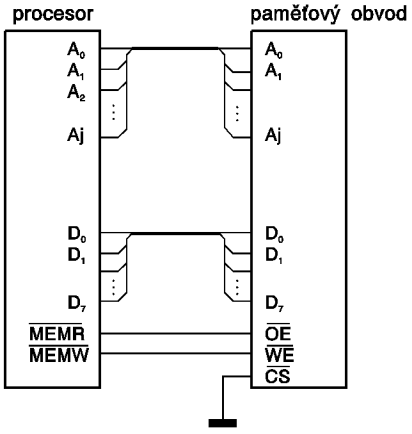


Typické parametry: min. doba zápisového cyklu ... 100%
(shodná s dobou čtecího cyklu)
předstih CS .....cca 80%
šířka WE pulsu .....cca 80%
předstih dat .....cca 50%

POUŽITÍ PAMĚŤOVÝCH OBVODŮ PŘI KONSTRUKCI HLAVNÍ PAMĚTI

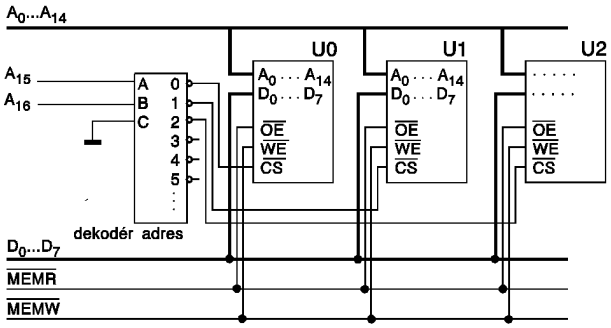
PŘIPOJENÍ PAMĚŤOVÉHO OBVODU NA SBĚRNICI MIKROPROCESORU

Pro jednoduchost předpokládáme hypotetický mikroprocesor, který má stejné rozhraní jako sběrnice počítače PC-XT. Paměťový obvod je typu SRAM a má kapacitu 32kx8bitů



ROZŠÍŘENÍ ADRESOVÉHO PROSTORU

Mějme za úkol navrhnout modul hlavní paměti o kapacitě 96kB (s organizací 96kx8) s použitím paměťových obvodů s organizací 32kx8.

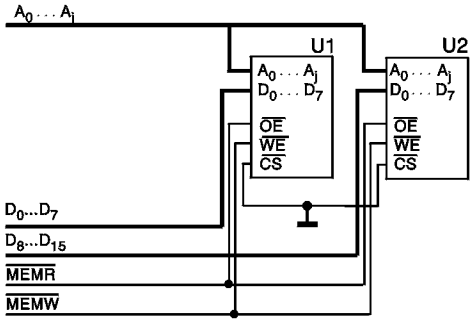


- adresové a datové vývody všech paměťových obvodů jsou spojeny paralelně a připojeny na odpovídající vodiče sběrnice
- vývody pro řízení čtení a zápisu jsou rovněž připojeny k vodičům MEMR a MEMW řídicí sběrnice
- jednotlivé paměťové obvody jsou vybírány aktivací vstupů CS připojených k výstupům dekodéru adres

ROZŠÍŘENÍ SLOVA HLAVNÍ PAMĚTI

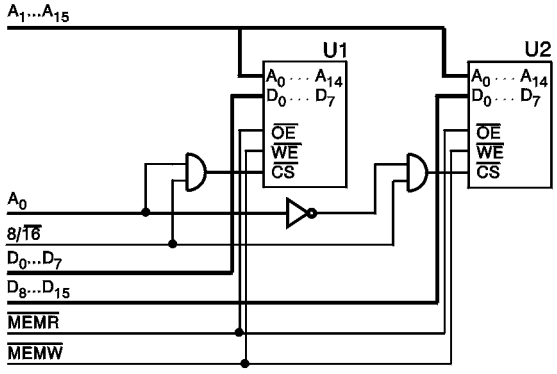
Mějme za úkol realizovat modul HP s organizací 32kx16 (t.j. šířka sběrnice paměťového modulu bude 16bitů) z paměťových obvodů 32kx8.

a) *paměť adresovaná po slovech* - nejmenší adresovatelnou jednotkou je 16-bitové slovo



- adresové a řídicí vývody obou paměťových obvodů jsou spojeny paralelně a připojeny na odpovídající vodiče sběrnice
- datové vývody jsou vyvedeny samostatně a připojeny na nižších (U1) a vyšších (U2) 8 vodičů datové sběrnice. Obvod U1 ukládá nižší a obvod U2 vyšší slabiku slova.

b) *paměť adresovaná po slabikách (ale se 16-bitovou datovou sběrnicí).* Podobně je řešena např. adresace paměti v počítačích PC-AT a vyšších.

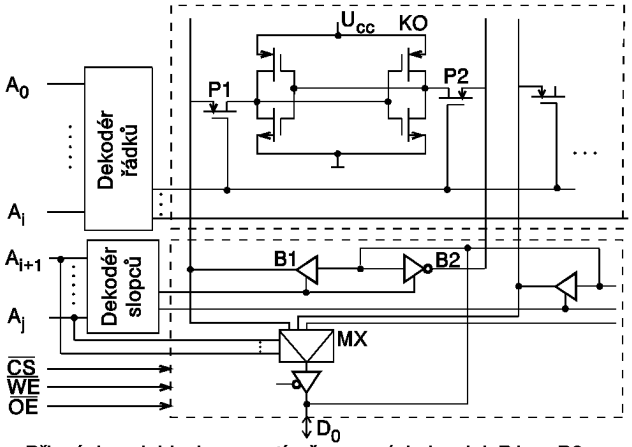


- vzhledem k adresaci po slabikách se mezi U1 a U2 přepíná signálem A0. Adresové vodiče obou obvodů jsou spojeny paralelně a na adresovou sběrnici jsou připojeny s "posunutím" o 1bit (A0 paměti na A1 sběrnice atd.).
- signál 8/16 svojí nulovou hodnotou povoluje 16-bitový přenos (vybrány jsou současně U1 a U2, na hodnotě signálu A0 pak nezáleží); jedničková hodnota signálu 8/16 povoluje 8-bitový přenos. Ten se podle hodnoty signálu A0 uskuteční po nižších (při A0=0) nebo vyšších (při A0=1) osmi bitech datové sběrnice.

KONSTRUKČNÍ PRINCIPY POLOVODIČOVÝCH PAMĚTÍ

STATICKÁ PAMĚŤ RAM

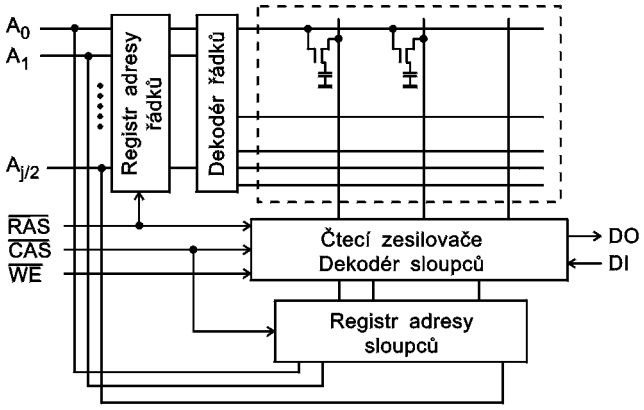
Paměťová buňka je realizována jako bistabilní klopný obvod - v CMOS logice jako dvojice invertorů



Při zápisu dojde k sepnutí přenosových hradel P1 a P2 a současně k aktivaci budičů B1 a B2. Tím se hodnota z vodiče D0 zapíše do klopného obvodu KO, protože přenosová hradla a budiče jsou "silnější" (mají menší impedanci v sepnutém stavu) než tranzistory v klopném obvodu. Při čtení se stav klopného obvodu KO přenese přenosovým hradlem P1 na první vstup multiplexeru MX a je-li tento vstup vybrán, objeví se na vodiči D0.

DYNAMICKÁ PAMĚŤ RAM

Jednotranzistorová paměťová buňka, data jsou uchována ve formě náboje na paměťovém kondenzátoru.



adresa je časově multiplexována, polovina adresy při $\overline{RAS}$=0 (řádek), druhá polovina adresy při $\overline{CAS}$=0 (sloupec)

Zápis: na datový (sloupcový) vodič se přivede zapisovaná úroveň a aktivuje se zvolený řádek. Paměťový kondenzátor se nabije nebo vybije.

Čtení: při výběru řádku se kondenzátory vybijí do vstupů čtecích zesilovačů (čtení je destruktivní, přečtenou informaci je nutno bezprostředně zapsat zpět).

Obnovení: Stejně jako čtení. Protože čtecí zesilovače jsou Umístěny ve všech sloupcích, obnovují se všechny sloupce jednoho řádku najednou.

POROVNÁNÍ VLASTNOSTÍ STATICKÝCH A DYNAMICKÝCH PAMĚTÍ RAM

- dynamické paměti jsou při stejné kapacitě podstatně levnější než statické.
  dynamická pam. buňka ..... 1 tranzistor
  statická pam. buňka ..... 6 tranzistorů

- dynamické paměti vyžadují externí logické obvody pro řízení obnovování, které musí periodicky generovat všechny adresy řádků

- dynamické paměti jsou poněkud pomalejší než statické vzhledem k tomu, že po destruktivním čtení musí být právě přečtená informace zapsána zpět do příslušných paměťových buněk (zajišťuje interní logika paměťových obvodů)

- dynamické paměti mají větší spotřebu v klidovém stavu, protože při obnovování dochází neustále k nabíjení a vybíjení paměťových kondenzátorů

ORGANIZACE PAMĚŤOVÉHO SYSTÉMU POČÍTAČE

HIERARCHIE PAMĚŤOVÉHO SYSTÉMU

je několikaúrovňové uspořádání pamětí různých kapacit a rychlostí s cílem dosáhnout výhodného poměru výkonnosti a ceny.

Cena paměti je: - přímo úměrná kapacitě
- přibližně nepřímo úměrná době přístupu

Paměťová hierarchie

typ paměti	Typická realizace	Doba přístupu	Kapacita
registry	klopné obvody	jednotky ns	desítky-stovky B
vyrovnávací paměť	statická RAM	10-20ns	stovky kB-jednotky MB
hlavní paměť	dynamická RAM	50-70ns	desítky-stovky MB
vnější paměť	pevný disk	5-15ms	Jednotky-desítky GB
záložní paměť	optický disk mag. páska ZIP drive	stovky ms- Stovky s	Stovky GB- jednotky TB

Hierarchické uspořádání paměti řeší konflikt mezi požadavky na rychlost paměti a na její kapacitu.

VIRTUÁLNÍ PAMĚŤ

je systém několika pamětí s různými parametry (kapacita, rychlost), který je řízen tak, aby vytvářel adresové prostory potřebné velikosti pro programy a pro data.

Virtuální paměť umožňuje:

- realizaci jednoho nebo několika logických (virtuálních) adresových prostorů, z nichž každý může být větší než je skutečná kapacita hlavní paměti označovaná jako fyzický adresový prostor. Logické adresové prostory jsou ve skutečnosti realizovány ve vnější paměti. Části programu a položky dat jsou automaticky přesouvány do hlavní paměti požaduje-li k nim procesor přístup.
- úsporné využití hlavní paměti tím, že v ní jsou přítomny jen ty části programu a dat, se kterými procesor právě pracuje.
- přemístění částí programu v hlavní paměti bez nutnosti je znovu překládat.
- vzájemnou ochranu jednotlivých programů v paměti a ochranu dat před neoprávněným přístupem a modifikací.

V režimu virtuální paměti pracuje program vždy s logickými (virtuálními) adresami.

Hlavní paměť se adresuje fyzickými adresami.

Překlad logických adres na fyzické zajišťuje mechanismus virtuální paměti.

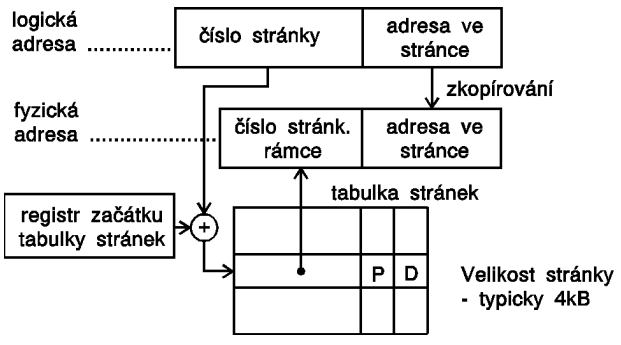
STRÁNKOVÁNÍ

je zaměřeno na realizaci velkého logického (virtuálního) adresového prostoru při malé kapacitě hlavní paměti.

Logický adresový prostor je rozdělen na úseky pevné délky nazývané stránky nebo také logické stránky.

Fyzický adresový prostor je rozdělen na stejně velké úseky nazývané stránkové rámce nebo také fyzické stránky.

Logický adresový prostor je realizován ve vnější paměti. Data (úseky programu) se přesouvají do hlavní paměti po jednotlivých stránkách tehdy, jsou-li v průběhu výpočtu požadována a pokud se již příslušná stránka v hlavní paměti nenachází. Stránkovací mechanismus pracuje s datovou strukturou - tabulkou stránek - uloženou též v hlavní paměti. Každé stránce odpovídá 1 položka v tabulce stránek. Ta obsahuje informaci o tom, zda se kopie příslušné stránky nachází v hlavní paměti a pokud ano tak ve kterém stránkovém rámci.



Problém: Tabulka stránek musí obsahovat 1 položku pro každou stránku v logickém adresovém prostoru i když tato stránka není použita. Je-li velký poměr mezi velikostí logického a fyzického adresového prostoru, může tabulka stránek zabírat podstatnou část hlavní paměti.

Příklad: virt. paměť 4GB 32b
hlavní paměť 4MB 22b
velikost stránky 4kB 12b

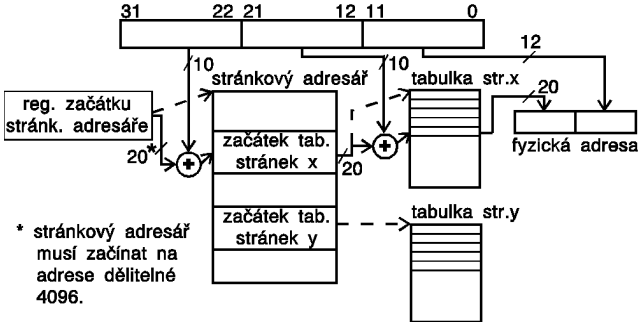
Tabulka stránek má 2^{20} položek

1 položka ... 10 bitů horní část fyzické adresy } 2B pro
1 bit platnost } 1 položku

⇒ tabulka stránek zabírá 2MB - nepřípustné.

Řešení:

Dvouúrovňová organizace tabulky stránek (použito v 80386 a vyšších)



Tabulka stránek musí obsahovat:

- horní část fyzické adresy (10 bitů)
- příznak přítomnosti stránky v hlavní paměti
- příznak indikující, že do stránky bylo po dobu přítomnosti v hlavní paměti zapisováno (Dirty bit)
- další příznaky - viz literatura

Funkce stránkovacího mechanismu

- je-li stránka přítomna v hlavní paměti (viz příznak přítomnosti stránky), přeloží se logická adresa na fyzickou
- není-li stránka přítomna, vyvolá se přerušení. Jeho obslužný program vyvolá načtení požadované stránky z vnější paměti.
- pokud při načítání nové stránky není volný žádný stránkový rámec v hlavní paměti, je třeba některý uvolnit. Toho se dosáhne přesunem vhodné stránky (např. nejdéle nepoužité) zpět do vnější paměti. Nebylo-li však po dobu přítomnosti této stránky v hlavní paměti do ní zapisováno (viz příznak), není třeba ji ukládat a stačí ji pouze přepsat nově načtenou stránkou.

SEGMENTACE

umožňuje dosáhnout úspory kapacity hlavní paměti tím, že se program do HP zavádí po částech.

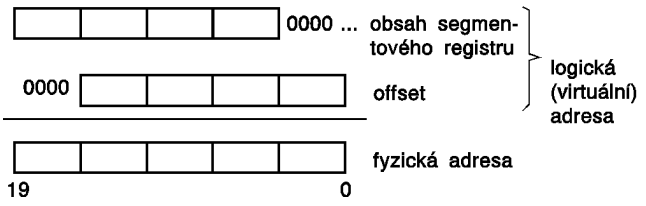
Segmenty jsou funkčně samostatné části programu proměnné délky, které lze zavádět do hlavní paměti v případě potřeby.

Adresy v segmentu jsou relativní vůči začátku (bázi) segmentu ⇒ přemístitelnost segmentů v hlavní paměti.

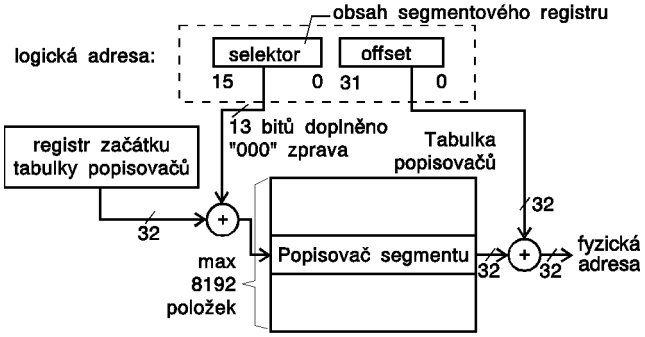
Logická adresa se skládá z obsahu segmentového registru a offsetu (posunutí).

Báze segmentu může být uložena

a) v segmentových registrech
   - např. 8086 a další procesory řady 80x86 v reálném módu (opakování)



b) v tabulkách adresovaných segmentovými registry
   - v chráněném režimu procesoru 80386 a vyšších



Popisovač segmentu obsahuje (u 80386 a vyšších, délka 8B)

- 32-bitovou bázi segmentu
- 20-bitový limit segmentu (max. délka)
- tzv. bit G (Granularity), udávající zda limit segmentu je vyjádřen ve slabikách nebo v blocích à 4096 slabik (4kB)
- přístupová práva

Výhody řešení ad b)

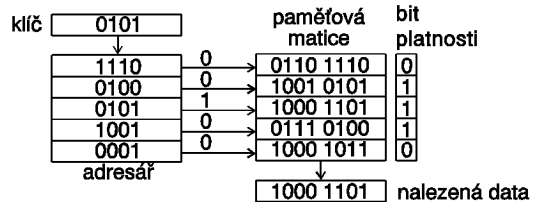
- fyzický adresový prostor (velikost hlavní paměti) až 4GB
- logický adresový prostor max 8192 segmentů à max 4GB
- umožňuje přiřadit segmentům přístupová práva - vzájemná ochrana jednotlivých běžících programů a dat
- hodnota selektoru není v přímém vztahu k fyzické adrese. Slouží jako index do předem připravené tabulky popisovačů jednotlivých segmentů.

RYCHLÁ VYROVNÁVACÍ PAMĚŤ (Cache)

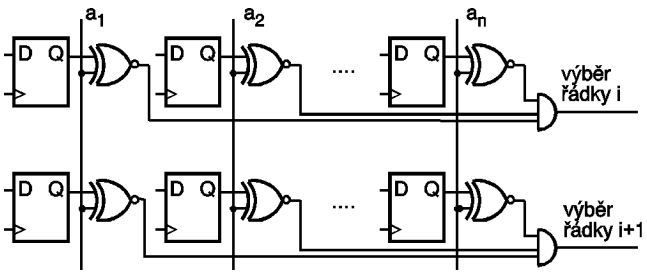
- rychlá paměť s malou kapacitou (desítky-stovky kB) zařazená mezi procesor a hlavní paměť
- obsahuje kopie nejčastěji používaných položek z hlavní paměti
- realizuje se jako statická paměť RAM - doba přístupu typicky 10-15ns
- pro její realizaci je nutná asociativní paměť - paměť adresovaná obsahem

PLNĚ ASOCIATIVNÍ PAMĚŤ

- paměť typu RAM se speciální strukturou lišící se od adresovatelné paměti RAM
- adresuje se částí datové položky, která se má vyhledat - tzv. klíčem
- na rozdíl od adresovatelné paměti RAM má místo adresového dekodéru tzv. adresář



Struktura adresáře asociativní paměti



Každá položka adresáře obsahuje logické obvody umožňující najednou prohledat všechny položky adresáře.

POUŽITÍ PLNĚ ASOCIATIVNÍ PAMĚTI JAKO RYCHLÉ VYROVNÁVACÍ PAMĚTI

- data zapsaná v paměťové matici asociativní paměti budou kopie "často" používaných položek dat z hlavní paměti
- klíčem bude adresa, která každou položku jednoznačně identifikuje

Jak bude probíhat čtení ?

- pokusíme se současně o čtení z cache a z hlavní paměti. Pokud se položka v cache nalezne, použije se a cyklus hlavní paměti se nedokončí. V opačném případě se data přečtou z hlavní paměti (a zpravidla se současně uloží do cache)

.... a zápis ?

- pokud položka v cache není, zapíše se (zpravidla) jen do hlavní paměti
- pokud tam je, lze postupovat dvěma způsoby:
  - zapsat novou hodnotu současně do cache a do hlavní paměti - tzv. průběžný zápis (write through)
  - zapsat novou hodnotu jen do cache - tzv. odložený zápis (write back)

Metoda odloženého zápisu je složitější v tom, že při uvolňování položky cache musíme někdy její původní obsah zapsat zpět do hlavní paměti (tehdy byla-li položka v cache modifikována).

POUŽITÍ PAMĚTI S OMEZENÝM STUPNĚM ASOCIATIVITY

Nevýhody plně asociativní paměti:

- pro konstrukci adresáře nelze použít standardní obvody RAM
- při stejné kapacitě cca trojnásobná plocha čipu (a tudíž i cena) oproti standardní RAM

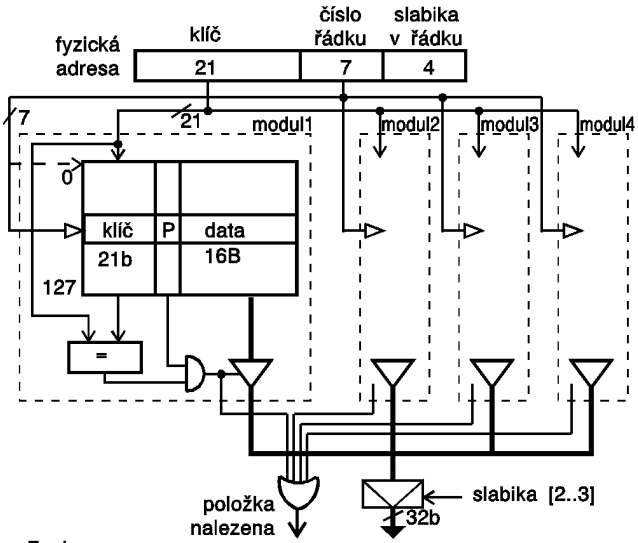
Řešení: Paměť s omezeným stupněm asociativity

Každé položce z HP je podle její adresy přiděleno jedno pevné místo (případně několik* míst), kde se v cache může nacházet. Adresář cache lze pak realizovat běžnou pamětí RAM, přítomnost položky se zjistí porovnáním požadovaného klíče s klíčem uloženým v adresáři (případně s několika* klíči).

Poznámka: Pro zvýšení efektivnosti se data z HP do cache nepřenášejí po jednotlivých slabikách nebo slovech, ale po delších blocích zvaných též řádky o délce např. 16 slabik.

*počet míst na nichž se konkrétní položka může v cache nacházet se nazývá stupeň asociativity

Řešení interní cache v procesoru 80486



Funkce

Při pokusu o čtení z cache se v každém modulu porovná klíč uložený v příslušném řádku (udávající adresu bloku dat, jehož kopie se v tomto řádku nachází) s klíčem odvozeným z adresy hledané položky dat. Shoda klíčů znamená, že hledaná položka byla v příslušném modulu cache nalezena.

VNĚJŠÍ PAMĚTI

charakteristiky ve srovnání s hlavní pamětí:

- větší kapacita
- nízká cena za bit
- energetická nezávislost a nedestruktivní čtení dat
- výměnné paměťové médium - archivace dat
- dlouhá vybavovací doba
- složité mechanické zařízení
- spolehlivost, údržba

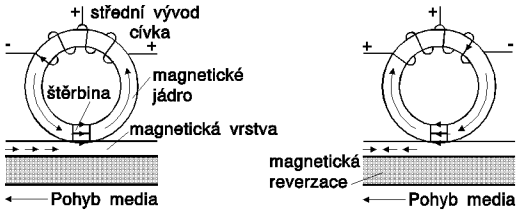
ZÁKLADNÍ POUŽÍVANÉ TYPY:

- magnetická disková paměť: pevný disk (výměnný, Winchestrovský), pružný disk (disketa)
- magnetická pásková paměť
- optická disková paměť

ZABEZPEČENÍ DAT:

např. CRC, ECC
(Cyclic Redundancy Check, Error Correcting Codes)
připojení kontrolních bitů - tj. zbytku po dělení bloku dat daným polynomem (stupně 16-56)

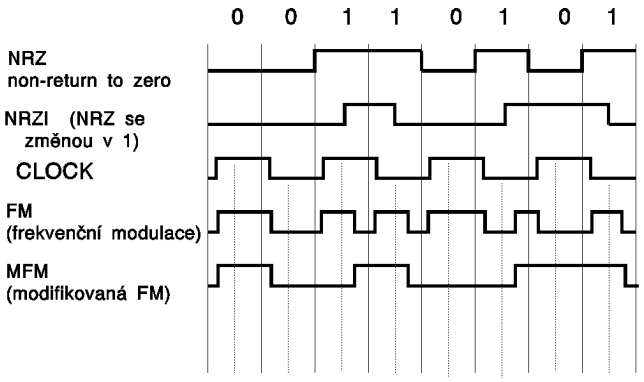
MAGNETICKÝ ZÁZNAM



podélný záznam
(existuje též vertikální záznam, siločáry ve štěrbině jsou kolmo k médiu, jiný tvar hlavy, větší hustota záznamu, větší šumová imunita)

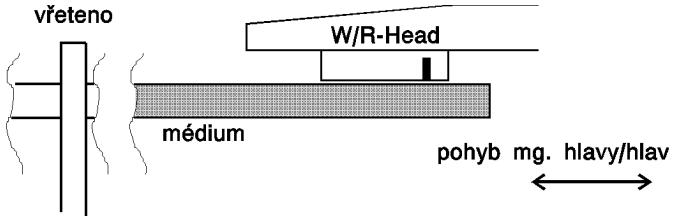
Záznam jednoho bitu - pomocí magnetických reverzací (čím méně reverzací, tím větší hustota záznamu)
měřítko hustoty záznamu - podélná hustota záznamu (data jsou zapsána sériově ve stopě)
jednotky: počet bitů na mm, počet bitů na palec
bpi bit per inch
záznam jednoho bitu zaujímá plochu několika μm^2 magnetické vrstvy
(hustota záznamu: např. 600 bitů/mm=15 000 bpi)

Kódování dat



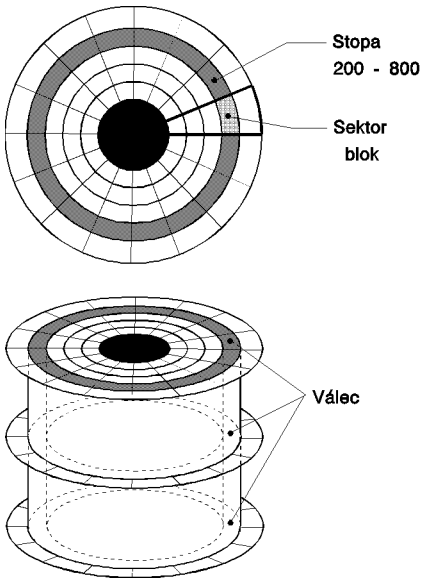
hrany znamenají změnu magnetizace - reverzaci
použití:
NRZI - magnetická páska (8 stop + lichá parita)
MFM - diskety
existují i další možnosti, např.:
RLL (Run-Length-Limited) - kódování ne bitů, ale vícebitových vzorů tak, aby se snížil počet jedniček a aby nebyly vedle sebe (složitější řadič)
např. RLL(2,7) - mezi dvěma jedničkami je min. 2 a max 7 nul (používá se i u optických pamětí)

Čtecí a záznamová hlava



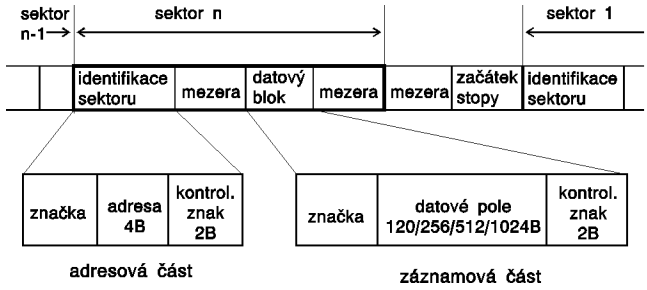
obvyklé hodnoty	vzdálenost hlava-medium	otáčky/min
klasický disk	$\approx 1\mu m$	
Winchester	$\approx 0.4\mu m$	3600-7200
disketa	v kontaktu	360
páska	v kontaktu	
optický disk	1mm	200-500

Magnetická disková paměť



Pozn: Délka stop je různá, ale počet zanamenaných bitů ve stopě je konstatní, tzn. hustota záznamu je na vnitřních stopách větší než na vnějších
(u optických pamětí - stopa ve tvaru spirály, sektor má stejnou délku na všech průměrech)

Příklad formátu záznamu na stopě



- začátek stopy - indexová značka (zářez)
- mezery slouží k čítání sektorů a jejich nalezení na stopě
- adresová část: adresa stopy (válce), číslo hlavy, adresa náhradní stopy

Přístupová doba (vybavovací doba):

- čekací doba pro otočení disku do polohy, kde jsou pod hlavou zaznamenaná data (rotační zpoždění)
- doba přestavení hlavy z jedné stopy na druhou (seek time)

OPTICKÝ ZÁZNAM

princip:

čtecí hlava vyšle laserový parsek a přijme paprsek odražený, který sebou nese informaci o tom, čím prošel

záznam informace:

- vytvoření prohlubní (děr) a ostrůvků
- vylisováním ..... CD-ROM
- odpařením (laserem s vyšší intenzitou) ....WORM
- změnou krystalické struktury ..... CD-R

materiál:

- polykarbonátový substrát
- reflexní kovová vrstva
- ochranný lak
- nejnověji ještě organické materiály, které zahřátím (laserem) mění svoji krystalickou strukturu nebo se stávají amorfní a jsou pak buď průhledné nebo neprůhledné (mezi substrátem a reflexní vrstvou)

typy optických médií

CD-ROM (Compact Disc)
pouze čtení, průměr 12cm, 680MB

CD-WORM (Write Once Read Many times)

CD-R (Compact Disc - Recordable)
optická paměť RWM, použití organických materiálů (cyanin) vložených mezi plykarbonátový substrát a reflexní kovovou vrstvu

DVD (Digital Video Disc též Digital Versatile Disc)
konec r. 1996 - nová generace CD disků s využitím hlavně pro video a multimediální aplikace
průměr 12cm, kapacita 4.7 GB, ale i "malé" produkty - 8cm, 1.4GB
používá laser s kratší vlnovou délkou a vícevrstvý záznam (tloušťka disku je poloviční ve srovnání s klasickým CD - je možné využít dvoustranný záznam)
DVD-WO, DVD-RAM 1997

PŘIPOJENÍ VNĚJŠÍCH PAMĚTÍ KE SBĚRNICI POČÍTAČE

normalizace rozhraní:
IDE (Integrated Driveand Electronics)
připojení 2 diskových jednotek typicky ke sběrnici ISA
do vzdálenosti 0.5 m
rychlost 2-3 MB/s

E-IDE (Enhanced IDE)
připojení až 4 jednotek (2 disky a 2 pomalejší jednotky, např. CD-ROM)
nejčastěji používané pro osobní a přenosné počítače

SCSI (Small Computer Systems Interface)
připojení až 7 jednotek
rychlost přenosu 2MB/s (SCSI-2 5MB/s)
použití u serverů a výkonnějších počítačů

funkce řídící jednotky:

- řízení obousměrného přenosu dat mezi operační pamětí a mechanikou diskové paměti
- převod dat při záznamu z paralelního tvaru do sériového (serializace) a při čtení ze sériového do paralelního (deserializace)
- kódování a převod dat do normalizovaného tvaru vhodného pro záznam na disk
- výpočet kontrolních znaků CRC, ECC
- synchronizace záznamu a čtení dat
- řízení mechaniky paměti

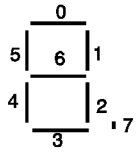
.....

zde neřešeno např.
RAID - Redundant Array of Independent Discs

VSTUPNÍ a VÝSTUPNÍ ZAŘÍZENÍ

spojení uživatele s počítačem

- klávesnice
- zobrazovací prvky
- tiskárny
- kreslící zařízení
- paměťové medium - disketa, děrná páska, ..
- myš, kulový ovladač
- ..... (joystik, hlasový vstup, ...)



zobrazovací prvky: LED diody (indikace 0 a 1)
sedmisegmentový displej
obrazovková jednotka

Základní princip zobrazení na obrazovce obrazovkové jednotky:
v jednom okamžiku svítí jen jeden bod, ten se musí rozsvítit 50krát za s, aby byl dobře čitelný a nekmital.

rozlišovací schopnost - počet zobrazitelných bodů (pixelů)
typicky 1024 x 768, ale i 1260 x 1024, lze zvolit méně, např. 800 x 600 - záleží na velikosti - úhlopříčce monitoru: 15" - 20"

videopaměť - uchování toho, co se zobrazuje - velký objem dat na každý bod souřadnice x, y a informace o jasu a barvě
např. 1024x768x8 tj. cca 1MB

zjednodušení: alfanumerická zobrazovací jednotka - zobrazení pouze znaků na předem určených pozicích - např. SM 7202 umožňoval zobrazit 80x24 znaků daného tvaru v rastru 5x7 bodů

TISKÁRNY

bodové (jehličkové, mozaikové) - typ je složen z bodů, čím více bodů, tím je kvalitnější UPS11-13

úderové (obrysové) - psací stroj UPS11-14

řádkové - řetězové, válcové - typový řetěz nebo válec
tisk celé řádky najednou

laserové - nepřímý elektrostatický princip tisku
rychlý a kvalitní tisk UPS11-15

tryskové (inkoustové) - kvalitní barevný tisk
typy jsou tvořeny buď rastrově nebo elektrosttickým vychylováním dráhy kapek UPS11-16

tepelné - tisk na tepelně citlivý papír

elektrostatické - tisk na speciální elektrografický papír

Důležité vlastnosti tiskáren: rychlost a kvalita tisku,
rozměry, spolehlivost, obsluha,
cena - pořizovací, provozní, tj. cena papíru, toneru, ..

MOZAIKOVÁ (JEHLIČKOVÁ) TISKÁRNA

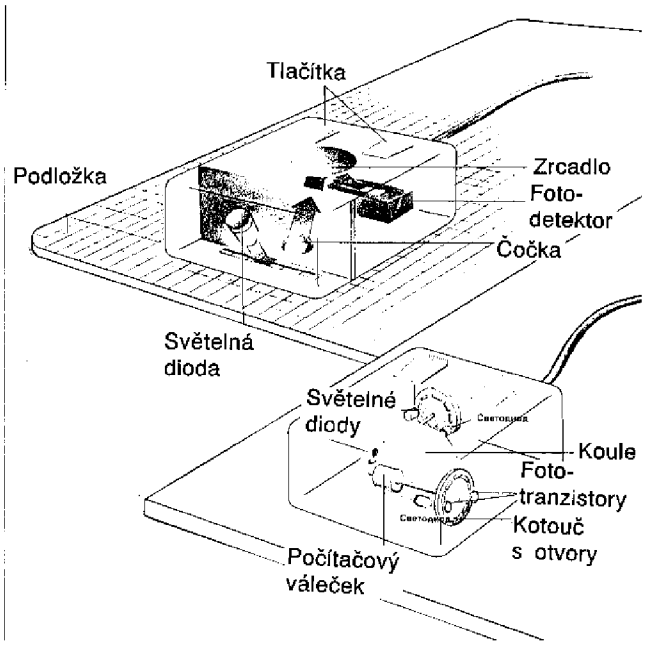
ÚDEROVÁ OBRYSOVÁ TISKÁRNA

LASEROVÁ TISKÁRNA

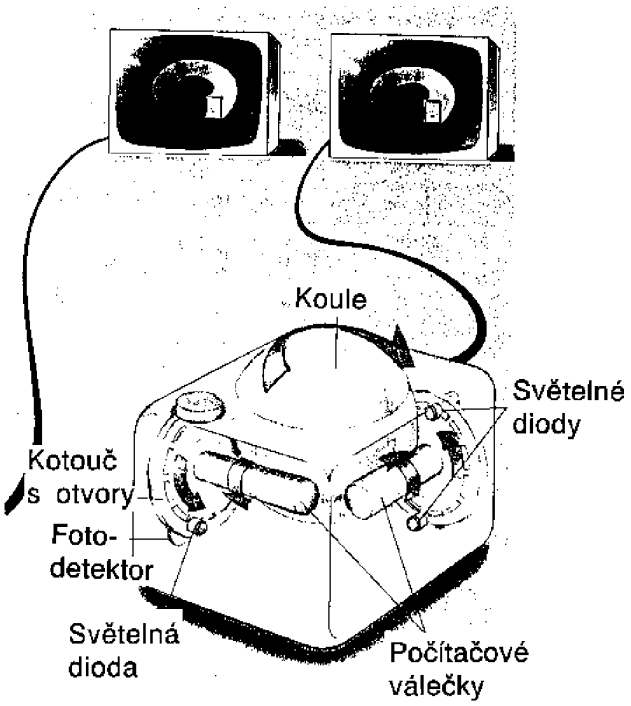
1 ... Laser
2 ... Otočné zrcadlo
3 ... Vybíjení fotokonduktivního válce
4 ... Nanášení toneru
- barviva
5 ... Obraz na válci
6 ... Zásobník papíru
7 ... Přenesený obraz
8 ... Fixace - zahřátí
9 ... Výstupní zásobník tisků

INKOUSTOVÁ TISKÁRNA

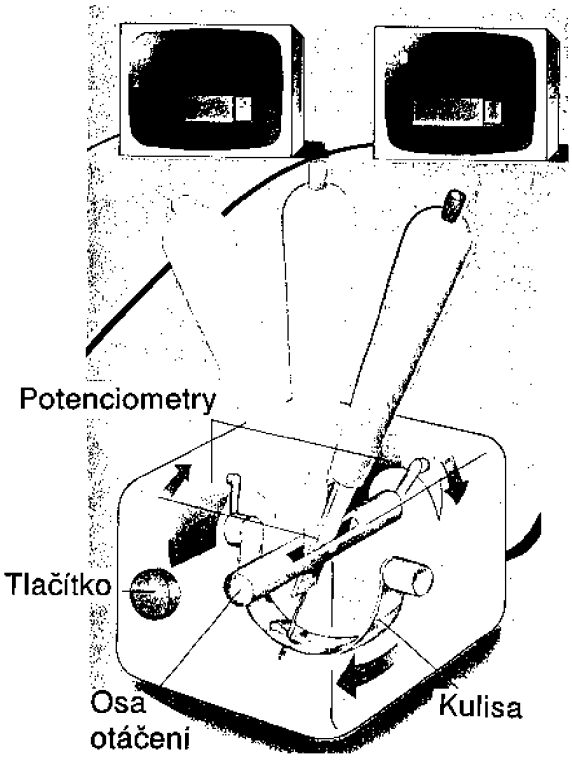
KLASICKÁ A SVĚTELNÁ MYŠ



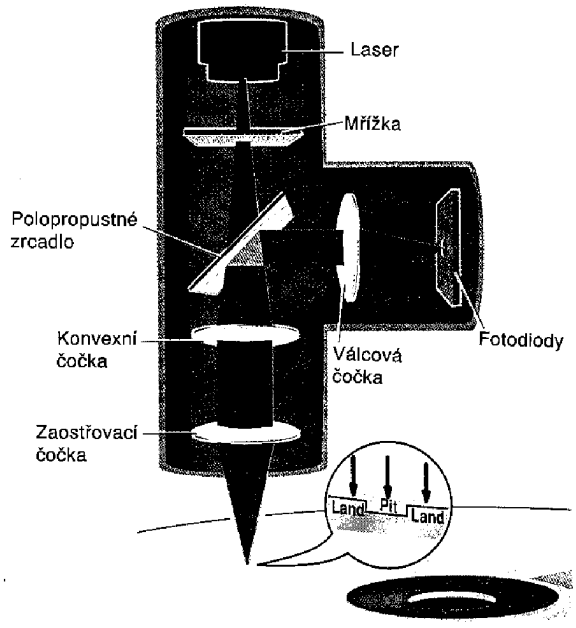
KULOVÝ OVLADAČ



PÁKOVÝ OVLADAČ (JOYSTIK)



ČTECÍ HLAVA OPTICKÉ PAMĚTI (CD-ROM)



Řadič

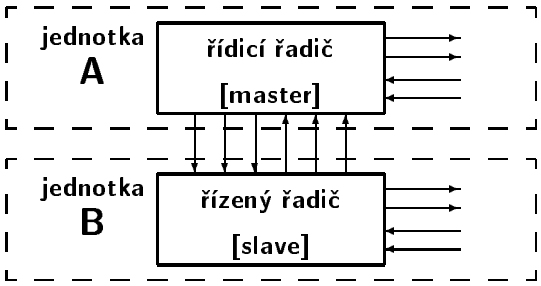
- v užším slova smyslu: **řídicí jednotka počítače** (viz schéma počítače von Neumannova typu) **[control unit]**
- v širším slova smyslu: **řídicí jednotka vůbec** (např. řadič tiskárny, řadič aritmetické jednotky, řadič počítače apod.) **[controller]**

sekvenční obvod — **výstupy: řídicí signály**
vstupy: stavové signály

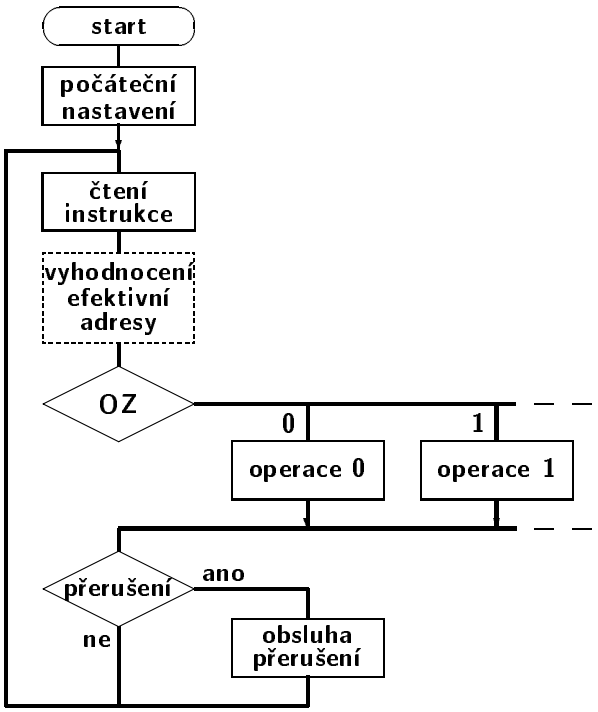
řídicí a stavové signály:
- **samostatné vodiče**
- **řídicí sběrnice**

---

hierarchie řadičů:



Řadič počítače: základní cyklus (instrukční cyklus)

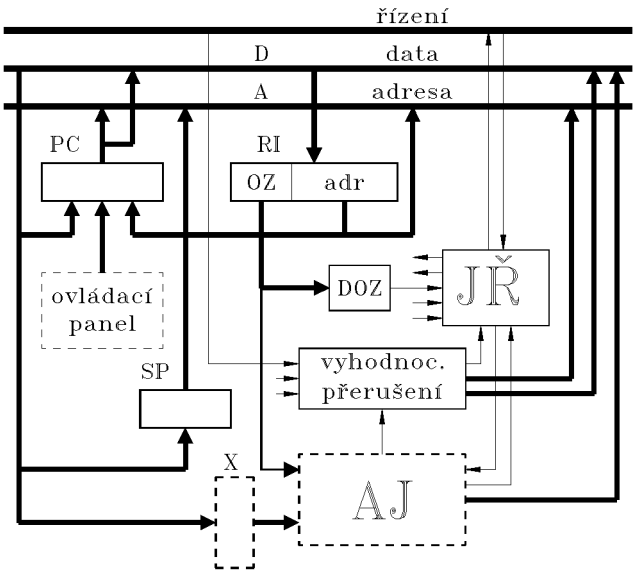


Poznámky k vývojovému diagramu:

- **Vývojový diagram je třeba modifikovat tak, aby vyhovoval požadovanému chování konkretního procesoru a co nejvíce vyhovoval jeho realizaci.**
- **V rámci realizace konkrétních operací se provádí také příp. čtení operandů a uložení výsledku; součástí těchto akcí může být i příp. vyhodnocení efektivních adres.**
- **Operační znak bývá dekódován postupně,** např.:
  - **skupina aritmetických a logických operací se dekóduje společně;**
  - **přečtou se operandy;**
  - **dekóduje se a provede se konkrétní operace.**
- **Někdy je třeba přejít na obsluhu přerušení dříve než je naznačeno ve vývojovém diagramu — chybí-li např. potřebná virtuální stránka v hlavní paměti, nelze instrukci přečíst** (natož dekódovat).
- **Test požadavku na přerušení a jeho obsluha bývají někdy zařazeny na začátek základního cyklu.**

Řadič počítače — příklad:

instrukce = 1 slovo (pojaté obecně, např. 32 b)
šířka datové sběrnice: 1 slovo



PC ...programový čítač, RI ...registr instrukcí, DOZ ...dekodér operačního znaku, JŘ ...jádro řadiče, SP ...ukazatel zásobníku, AJ ...aritm. j.

čtení instrukce
žádost o přidělení sběrnic
ne
přiděleno
ano
PC → SB.A
PC → SB.A
čtení z HP
SB.D → RI
ne
přečteno
ano
PC+1 → PC
konec

SB ... sběrnice
SB.A ... adresová sběrnice
SB.D ... datová sběrnice
HP ... hlavní paměť

$\chi \rightarrow$ SB.ψ $\Rightarrow$ vysílat signál „sběrnice obsazeny“

Některé aritmetické a logické operace a přesun:

operace typu

S ⊙ <adr> → S

0 + <adr> → S (přesun)

S ... střadač — obsažen v AJ

⊙ ... +, −, *and*, *or*, *xor* apod.

Tyto operace lze provést v 1 taktu.

aritm./log./přesun
RI.adr → SB.A
RI.adr → SB.A
čtení z HP
SB.D → X
ne
přečteno
ano
start AJ
konec

na vstupy AJ se přivádí mj. OZ ↓

Skoky:

podmíněný skok
podm.
splněna
nesplněna
nepodmíněný skok
RI.adr → PC
konec

skok do podprogr.
SP − 1 → SP
SP → SB.A
SP → SB.A
PC → SB.D
zápis do HP
ne
zapsáno
ano

obsluha přerušení
SP − 1 → SP
SP → SB.A
SP → SB.A
DataP → SB.D
zápis do HP
ne
zapsáno
ano
zákaz přerušení
AdrP → SB.A
AdrP → SB.A
čtení z HP
SB.D → PC
ne
přečteno
ano
konec

DataP Data Přerušení (kontext): 1. PC
⋮
Umožňují přesnou identifikaci příčiny přerušení a znovuspuštění přerušeného programu (jeho pokračování).

AdrP Adresa, na níž je uložena Přerušovací adresa pro příslušnou příčinu přerušení.

Lze provést velmi rozmanité modifikace, např.:

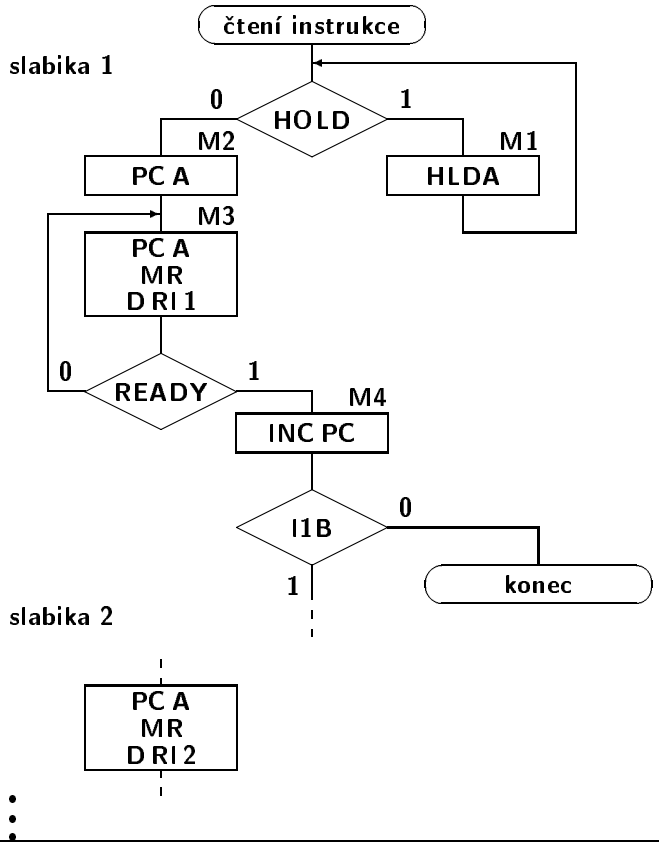
I. principiální modifikace:

1. **délka instrukce — proměnná** (1 nebo více slabik)
2. **šířka datové sběrnice: 8b = 1B** (1 slabika)
3. **funkci přidělovače sběrnic zastává procesor**
4. **reg. RI rozdělen na „podregistry" RI 1, RI 2, ...**

II. čistě formální modifikace:

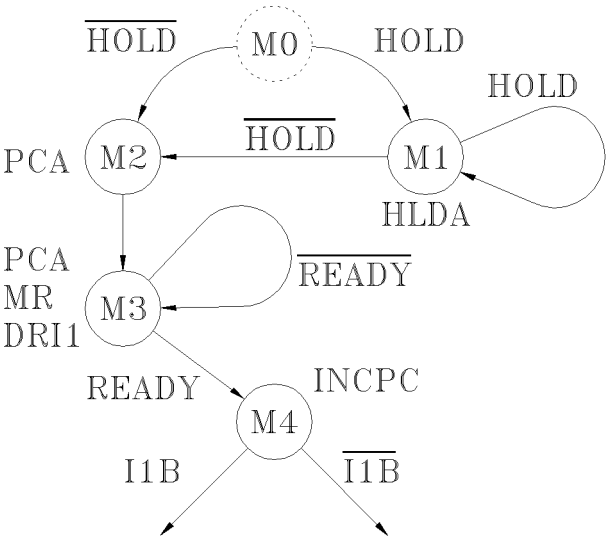
Místo dílčích operací (tzv. mikrooperací) i místo podmínek budou uváděny příslušné řídicí a stavové signály, např.:

MR	**čtení z HP** [Memory Read]
PC A	**PC → SB.A**
INC PC	**inkrementace PC** (PC+1 → PC)
D RI 1	**SB.D → RI 1**
D RI 2	**SB.D → RI 2**
HLDA	**sběrnice volná** [HoLD Ackowledgement]
READY	**hotovo** (přečteno/zapsáno)
HOLD	**uvolnit sběrnice (žádost)**
I1B	**délka instrukce > 1B**
I2B	**délka instrukce > 2B**



Návrh řadiče (popř. jádra řadiče)

vývojový diagram — jiná forma grafu přechodů:

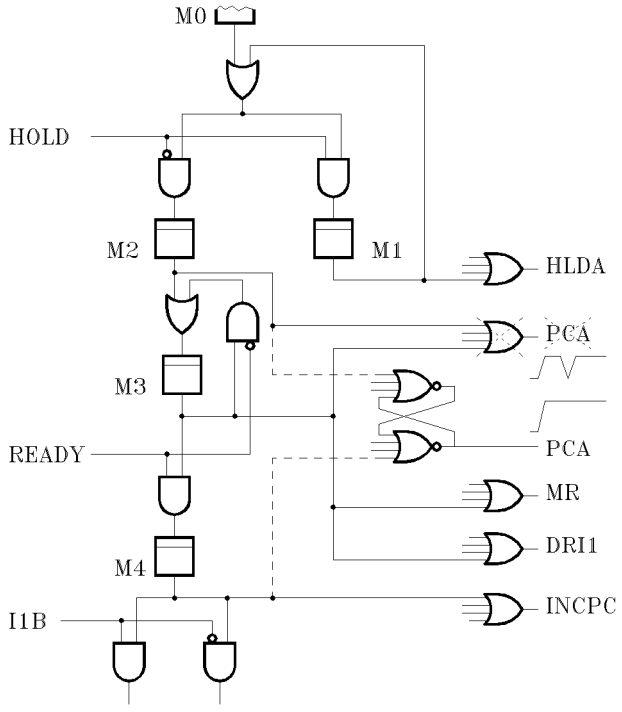


graf přechodů → sekvenční obvod — tzv.

„klasický" (nebo ***obvodový) řadič***

Řadič s řídicími řetězci

(jedna z možností realizace klasického řadiče)



mikroprogramovaný řadič

Provedení 1 operace (např. provedení instrukce, včetně přečtení, dekódování atd. — u řadiče počítače) **sestává z provedení řady dílčích operací — tzv. mikrooperací. Příkazy k provedení mikrooperací lze zapsat ve formě připomínající instrukce — tzv. mikroinstrukce.**
Ty tvoří tzv. mikroprogram, který lze uložit do tzv. řídicí paměti.
Lze navrhnout „klasický" řadič (nebo i jednodušší obvod)**, který bude zajišťovat postupné provádění mikroinstrukcí. „Počítač" takto řízený bude fungovat jako tzv. mikroprogramovaný řadič.**

mikroprogram — „nekonečný cyklus"
(viz základní cyklus počítače)

řadič počítače (jádro řadiče):
program
↓
instrukce ... **mikroprogram (1 průchod cyklem)**
↓
mikroinstrukce

soubor všech mikroprogramů daného procesoru:
mikroprogramové vybavení **[firmware]**

- **vertikální mikroprogramování:**
  - **krátké mikroinstrukce (např. 16b) obdobné instrukcím → čítač adres mikroinstrukcí**
  - **1 mikroinstrukce ... několik taktů**
- **horizontální mikroprogramování:**
  - **dlouhé mikroinstrukce (64 a víc bitů)**
  - **1 mikroinstrukce ... 1 takt**
  - **v mikroinstrukci přímo určeny řídicí signály**
  - **omezená volba adresy násl. mikroinstrukce**

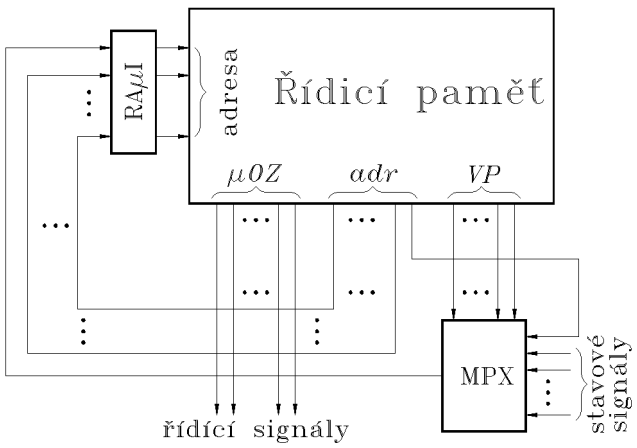
---

horizontální mikroprogramování

μOZ	adr	VP

- μOZ — **mikrooperační znak — hodnoty řídicích signálů**
- adr — **adresa následující mikroinstrukce**
- VP — **výběr podmínky**

struktura mikroprogramovaného řadiče — horizontální organizace



poslední bit adresy := hodnota vybrané podmínky

- **podmínka = některý stavový signál ⇒ větvení: adresa $\alpha\beta\cdots\gamma 0$ nebo $\alpha\beta\cdots\gamma 1$**
- **podmínka = poslední bit adr ⇒ nedochází k větvení**

Proudové zpracování
[Pipelining]

několik jednotek v kaskádě (srov. výrobní pás):
— každá jednotka provede část operace
— jednotky pracují současně

triviální případ — předčítání instrukcí:
jedna instrukce se provádí, další se čte a dekóduje

typická organizace:

jednotka	instrukce					
čtení instrukce	1	2	3	4	5	6
dekódování OZ		1	2	3	4	5
čtení operandů			1	2	3	4
provedení operace				1	2	3
uložení výsledku					1	2

čas ⟶

ideální případ: každý takt dokončena 1 instrukce

konflikty:

- **datový: potřebná data dosud neuložena**
- **skokový: adresu skoku prozatím nelze určit**

nejjednodušší řešení: počkat

Počítače typu RISC

Jak navrhnout „rychlý“ procesor?

Řešení: Co „nejvýkonnější“ instrukce.

???

Statistika:

Př.: fa DEC: vývoj MicroVAX–32 ← VAX 11/780
98% instrukcí v typických úlohách (?) používá:
58% instrukcí zařazených do operačního kódu
15% mikroprogramového vybavení

Př.: fa Motorola (68000)

přesuny :	**cca 33%**	cca 33%
skoky a srovnávání :	**cca 35%**	cca 68%
$+ \; - \;$ *and* *or* :	**cca 11%**	cca 79%
posuvy :	**cca 3%**	cca 82%
násobení :	**cca 0,6%**	
dělení :	**cca 0,1%**	

Závěr: Velmi „výkonné“ instrukce se používají příliš málo!

Co s tím?

Co „nejtriviálnější“, ale co „nejrychlejší“ instrukce.

Počítače typu RISC

[Reduced Instruction Set Computers]

(Počítače s redukovaným souborem instrukcí)

charakteristické rysy:

- **Poměrně malý počet instrukcí** (≤ 128 ?), **a to „velmi jednoduchých“**
- **Velmi krátká doba provedení instrukce (1 instrukce zprav. dokončena v 1 taktu)**
- **„Klasický“** (tj. obvodově realizovaný) **řadič**
- **Proudové zpracování**
- **1 instrukce — 1 slovo**
- **Malý počet formátů instrukcí** (≤ 4)
- **Malý počet způsobů adresace** (≤ 4)
- **Velký počet registrů** (≥ 32)
- **Komunikace s hlavní pamětí: pouze instrukcemi „přesun“**
  (aritmetické/logické operace: pouze mezi registry)

„protipól“ počítačů typu RISC:

Počítače typu CISC

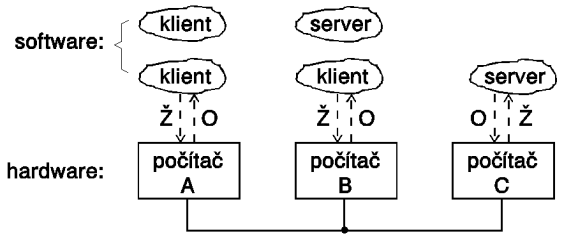
[Complex Instruction Set Computers]

(Počítače se „rozsáhlým“ souborem instrukcí)

POČÍTAČOVÉ SÍTĚ

terminálová síť (předchůdce počítačových sítí):
- množina terminálů připojená k hostitelskému počítači
způsob spolupráce: centralistický
veškeré aplikace jsou prováděny na hostitelském počítači
typický případ - střediskový počítač s multiuživatelským operačním systémem zpřístupňující výkonný počítač více uživatelům.

počítačová síť:
- množina vzájemně propojených autonomních počítačů různých typů
způsob spolupráce: klient - server
autonomní počítače si mohou poskytovat vzájemné služby:
klient - žadatel a příjemce služby (charakter aplikační úlohy - spuštění na žádost uživatele)
server- vykonavatel služby (charakter systémové úlohy - spuštění operačním systémem)
př.: file server, print server, atd.



Ž.....žádost o službu
O.....odpověď serveru

Přínos počítačových sítí:

a) prostředek pro komunikaci mezi účastníky sítě
- nejdůležitější aplikace klient - server v síti INTERNET: e-mail, ftp (file transfer protocol), telnet - spojení se vzdáleným počítačem, ping - zjištění funkčnosti vzdáleného systému, finger - zjištění uživatelů vzdáleného systému, talk -interaktivní spojení mezi vdálenými uživateli.
Nejpoužívanější služba: WWW (World Wide Web)

b) sdílení nákladných prostředků
- velkokapacitní diskové paměti, tiskárny, atd.

c) zajištění spolehlivosti
- zálohování (bankovnictví, armáda, řízení leteckého provozu, atd.)

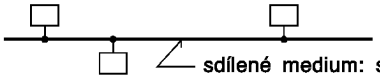
d) paralelní výpočetní prostředek (novější pohled):
- volně vázané počítače komunikující pomocí výměny zpráv:
př.: PVM (Parallel Virtual Machine) - programové vybavení umožňující využít síť počítačů pod OS UNIX jako paralelní výpočetní prostředek.

Typy počítačových sítí:

WAN ..... Wide Area Network (long haul networks)
LAN ..... Local Area Networks,
MAN ..... Metropolitan Area Netwoks: technologie LAN, ale zahrnující oblasti velkých měst (~200km)

POČÍTAČOVÉ SÍTĚ TYPU LAN
- stručná charakteristika

- sdílené medium (sítě typu "broadcast"): koaxiální kabel
dvoulinka
optický kabel

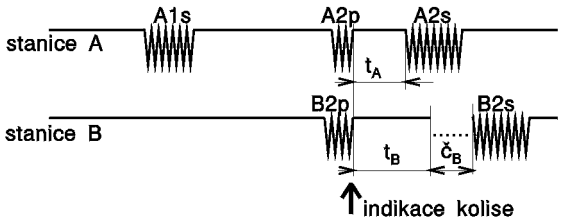


- rychlejší než WAN: 4Mb/sec ÷ 2Gb/sec

PŘIDĚLOVÁNÍ SDÍLENÉHO MEDIA:
chybí centrální přidělovač ⇒ distribuovaný přístup
a) nedeterministické přidělování
př.: CSMA/CD (carrier sense multiple access/collision detection)
použití: v síti Ethernet (Xerox r.1976; později IEEE standard 802.3)
Strategie přidělování:
- Má-li stanice připravená data k vysílání a je-li na sběrnici provoz, pak stanice čeká na jeho ukončení. Není-li na sběrnici provoz, pak zahájí okamžitě vysílání.
- V případě, že více stanic zahájí vysílání v úzkém časovém intervalu, pak nastane kolise. Po detekci kolise každá stanice ukončí okamžitě vysílání a generuje časový interval náhodné délky po jehož uplynutí se pokusí o nové vysílání původního paketu.
- V případě opakované kolise při vysílání téhož paketu se interval pro získání náhodné doby pro odmlčení stanice prodlužuje. Tímto způsobem se zmenšuje pravděpodobnost opakovaných kolisí v případě snahy velkého počtu stanic o přístup na sběrnici.

- důvod vzniku kolisí:
konečná rychlost šíření signálu na sběrnici

šíření signálů vzdálených stanic

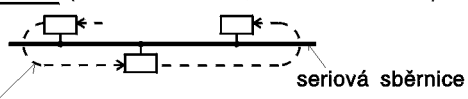


A1s .... správně vyslaný paket stanicí A bez výskytu kolise
A2s, B2p ..... pakety porušené při kolisi

t_A, t_B ..... náhodně generované doby odmlčení stanic A,B
$č_B$ ..... čekání stanice B na ukončení provozu na sběrnici
A2s, B2s ..... opakovaně a správně vyslané pakety po kolisi.

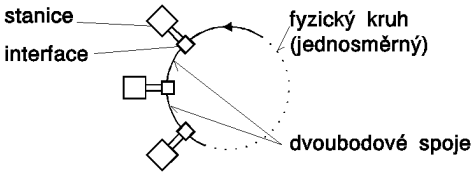
b) deterministické přidělování:
- dle pověření (pešek, token) ⇒ bezkolisní přístup

b1) token bus (Generals Motors; IEEE Std. 802.4)



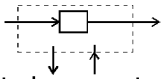
logický kruh pro předávání peška

b2) token ring (IBM; IEEE Std. 802.5)

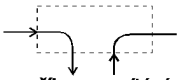


- interface - 2 módy:

listen mode - stanice není držitelem peška

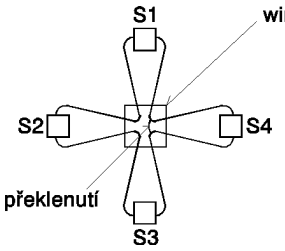


transmit mode - stanice je držitelem peška:



- každý vysílač likviduje svůj původní rámec ⇒ snadná kontrola spolehlivosti a snadné potvrzování příjmu přijímací stanice

- realizace kruhu:

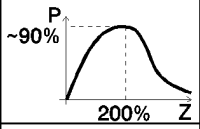
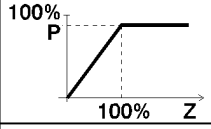


wire center: - umožňuje překlenutí chybné části a snadnou lokalizaci přerušeného místa

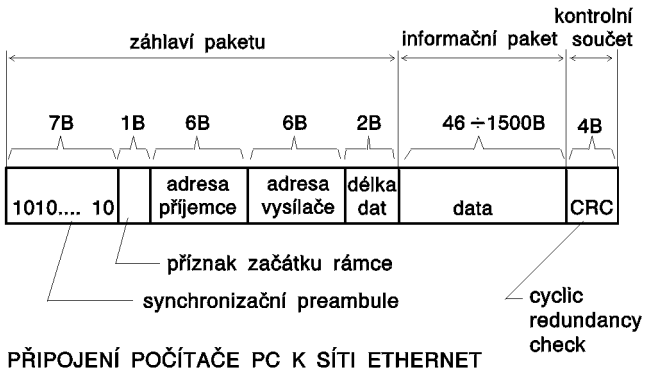
strategie přidělování:

- Stanice čeká na tzv. pověření.
- Po obdržení pověření stanice zahájí vysílání má-li připravená nějaká data. V opačném případě pošle pověření další stanici v kruhu.
- Doba držení pověření stanicí je omezena. Pokud stanice nestačí odeslat během této doby všechna data, pak přeruší vysílání a odešle příkaz další stanici v kruhu.

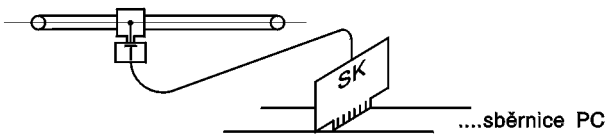
SROVNÁNÍ LOKÁLNÍCH SÍTÍ:

hledisko \ typ sítě	Ethernet	token bus/token ring
maximální doba přístupu stanice k médiu	nedeterministická	deterministická
závislost propustnosti sítě P na její zátěži Z		
chování sítě při malé zátěži	bez zpoždění	se zpožděním (čekání na pověření)
priority zpráv	nepodporuje	podporuje

STRUKTURA RÁMCE V SÍTI ETHERNET



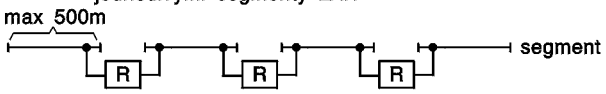
PŘIPOJENÍ POČÍTAČE PC K SÍTI ETHERNET



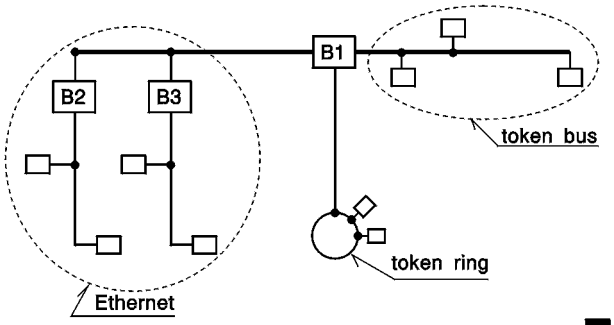
T ... tranceiver - funkce: detekce nosné + detekce kolise
SK ... síťová karta (síťový interface)
funkce: - převod kódů: Manchester ⟷ binární
- autonomní příjem a vysílání rámců:
- kontrola správnosti rámce (výpočet CRC)
- odstranění obálky rámce pro příjem
- vytváření obálky rámce při vysílání
- realizace strategie přístupu k médiu metodou CSMA/CD

PROPOJOVÁNÍ SÍTÍ

- opakovač (repeater): provádí pouhé kopírování všech bitů mezi jednotlivými segmenty LAN



- most (bridge): provádí směrování rámců mezi sítěmi typu LAN,
význam: - vzájemná isolace provozu mezi jednotlivými segmenty téže sítě (viz B2,B3)
- transformace a směrování rámců mezi sítěmi LAN odlišného typu (viz B1)



- brána (gateway, router): provádí směrování mezi sítěmi odlišného typu: LAN ↔ WAN
$WAN_1 \leftrightarrow WAN_2$

STRATEGIE SMĚROVÁNÍ MOSTU

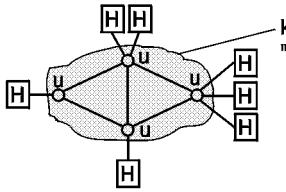
Informace potřebné pro směrování se vytváří dynamicky v paměti mostu na základě vlastního učení: most přijímá všechny rámce ze všech směrů a zapamatovává si odkud jednotlivé rámce přicházejí a tím získává informace o rozložení vysílacích stanic a sítí.

Možné situace:

1) Při příchodu rámce most nemá žádné informace o poloze cílové stanice tohoto rámce; odešle rámec do všech směrů vyjma směru příchozího. Zapamatuje směr příchodu rámce a přiřadí tomuto směru stanici, která byla zdrojem tohoto rámce.
2) Rámec přichází do mostu ze směru, který reprezentuje spojení mostu s oběma stanicemi (tj. zdrojem i cílem rámce). Most daný rámec ignoruje.
3) Rámec přichází do mostu ve směru, který se liší od směru spojujícího most s cílovou stanicí; most odešle rámec do směru cílové stanice.

POČÍTAČOVÉ SÍTĚ TYPU WAN
- stručná charakteristika

- klasické přepojování v určité množině dvoubodových spojů (zde není přidělování společného media jako v případě LAN)
- pomalejší než LAN: závisí na propojovacích spojích typicky 9,6kb/sec 45Mb/sec
- starší než LAN: př. síť ARPANET z r.1969 (základ INTERNETU)



u ... uzel (router, gateway, interface message processor)
H ... host computer

MOŽNÉ STRATEGIE PŘENOSU ZPRÁV

1) přepojování kanálů - vytvoří se fyzický spoj od zdroje do cíle pro celou dobu přenosu ⇒ vyžaduje stejnou přenos. rychlost po celé trase ⇒ nepoužívá se!
2) přepojování paketů: t.j. přenos zprávy po částech (paketech) po dvoubodových spojích s dočasným uložením v mezilehlých uzlech přenosové cesty ⇒ výhody:
   - buffery stejné délky
   - překrývání při přenosu

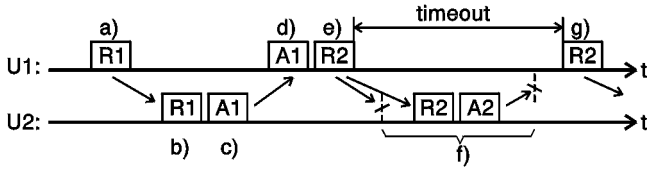
a) virtuální spoje (connection oriented service)
   - vyznačení určité cesty (otevření spojení)
   - přenos všech paketů po dané cestě
   - zrušení cesty (uzavření spojení)

   } analogie 1) (telefon)

b) datagramová služba (connectionless service)
   - každý paket je sítí přenášen samostatně individuální cestou (analogie dopisu)

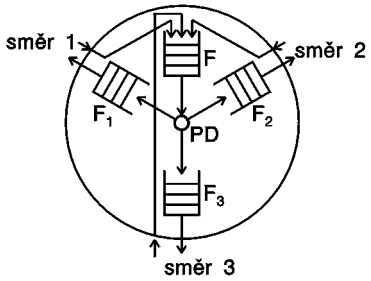
HLAVNÍ FUNKCE KOMUNIKAČNÍCH SÍTÍ:

1) Zajištění spolehlivého přenosu rámců po nespolehlivých spojích (např. telefonních)
   - použití bezpečnostních kódů pro detekci chyb při přenosu (CRC)
   - metoda ARQ (automatic repeat request) nejjednod. forma: protokol stop and wait



a) vysílání rámce č.1
b) příjem rámce č.1
c) vysílání potvrzení rámce č.1
d) příjem potvrzení rámce č.1
e) vysílání rámce č.2
f) ztráta rámce č.2 (nebo ztráta potvrzení rámce č.2)
g) opakované vysílání rámce č.2 po uplynutí časového intervalu timeout

2) Směrování paketů v síti
   - zkoumání cílové adresy přijatého paketu a jeho zařazení do příslušné fronty dle směrovacích tabulek uložených v paměti uzlu.
   - periodické obnovování směrovacích tabulek umožňuje přizpůsobit chování sítě změnám její topologie či zatížení této sítě.
   - zjednodušená vnitřní struktura uzlu:

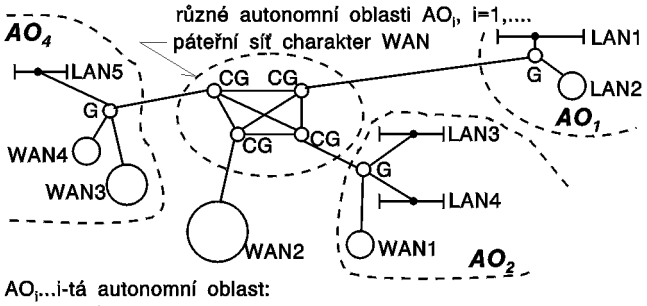


F ..... fronta paketů přicházejících do uzlu z různých směrů

PD ... proces zajišťující distribuci paketů do výstupních front F_1, F_2 , F_3 odpovídajících jednotlivým směrům 1, 2, 3 (dle směrovacích tabulek)

INTERNET

Charakteristika: - síť sítí - celosvětová síť propojující již existující heterogenní sítě (WAN,LAN)
- je realizována jako páteř (tvořená uzly "core gateways") ke ktaré jsou připojeny různé autonomní oblasti AO_i, i=1,....



AO_i...i-tá autonomní oblast:
- může se postupně dynamicky rozvíjet,
- každá oblast si udržuje svou centrální správu
- součástí každé autonomní oblasti je počítač, zde označený G (gateway), který sděluje adresy nově připojených sítí této autonomní oblasti počítačům CG z páteřní sítě a všechny rámce směřující z této oblasti odesílá do připojeného CG v páteřní síti

CG...core gateways:
- tvoří páteřní síť
- mají kompletní informace o všech možných sítích připojených k síti INTERNET a příslušné směrovací tabulky; dle těchto informací směrují pakety
- periodicky vyměňují mezi sebou nové informace (t.j. informace týkající se nově připojených sítí) pro zajištění směrování