



# Architektura počítačů

zpracoval Martin Kuba

16. května 1995

## Obsah

<b>1 Úvodní tlachy</b>	<b>3</b>
1.1 Definice některých pojmů . . . . .	3
1.2 Historie počítačů . . . . .	4
1.3 Generace počítačů . . . . .	4
<b>2 Architektura počítačového systému</b>	<b>4</b>
<b>3 Číselné soustavy</b>	<b>5</b>
<b>4 Zobrazení dvojkového čísla v počítači</b>	<b>6</b>
4.1 Zobrazení záporných čísel . . . . .	6
4.2 Zobrazení reálných čísel . . . . .	7
4.2.1 zobrazení v pevné řádové čárce . . . . .	7
4.2.2 zobrazení v pohyblivé řádové čárce . . . . .	7
4.2.3 zobrazení logických hodnot . . . . .	7
4.2.4 kód BCD . . . . .	7
4.3 Vnější kódy . . . . .	8
<b>5 Logické obvody</b>	<b>8</b>
5.1 Kombinační logické obvody . . . . .	9
5.2 Sekvenční logické obvody . . . . .	10
5.2.1 Klopný obvod RS . . . . .	10
5.2.2 Typické sekvenční obvody v počítačích . . . . .	10
5.2.3 Bitové posuny . . . . .	10
<b>6 Paměti</b>	<b>11</b>
<b>7 Řadiče</b>	<b>12</b>
7.1 Direktivní ovládání . . . . .	12
7.2 Mikroprogramový řadič . . . . .	12
7.3 Zpětnovazební ovládání . . . . .	13

<b>8</b>	<b>Procesor</b>	<b>13</b>
8.1	Náš počítač . . . . .	14
8.1.1	instrukce a registry . . . . .	14
8.1.2	mikroinstrukce . . . . .	14
8.1.3	Zásobník . . . . .	14
8.1.4	Vstupně/Výstupní operace . . . . .	14
8.1.5	systém přerušení . . . . .	15
<b>9</b>	<b>Řada procesorů Intel 86</b>	<b>15</b>
<b>10</b>	<b>Správa paměti</b>	<b>16</b>
10.1	CACHE paměť . . . . .	16
10.2	Stránkování operační paměti . . . . .	16
10.3	Mapování operační paměti . . . . .	16
10.4	Virtuální paměť . . . . .	16
<b>11</b>	<b>Klasifikace paralelních procesorů</b>	<b>17</b>
<b>12</b>	<b>Periferie</b>	<b>17</b>
12.1	Záznam na magnetické vrstvy . . . . .	17
12.1.1	magnetická páska . . . . .	17
12.1.2	magnetické disky . . . . .	17
12.1.3	diskety . . . . .	17
12.1.4	optické diskové paměti . . . . .	18
12.2	Tiskárny . . . . .	18
12.3	ostatní . . . . .	18
<b>13</b>	<b>Struktura a funkce OS</b>	<b>19</b>
13.1	Strukturální popis . . . . .	19
13.2	základní funkce řídicích programů . . . . .	19
13.3	základní aplikační programy . . . . .	19

# 1 Úvodní tlachy

## 1.1 Definice některých pojmů

**počítač** – stroj na zpracování informací

**informace** — údaj, který je přisouzen datům

**data** — znaky, číslice, num. hodnoty, symboly, grafy atd.

**program** — algoritmus řešení nějakého problému, posloupnost instrukcí

**instrukce** — předpis pro provedení základní operace realizované technickým vybavením počítače

**HARDWARE** — technické vybavení počítače, souhrnný název fyzických zařízení počítače

**SOFTWARE** — programové vybavení počítače

**FIRMWARE** — programy, které jsou technicky vestavěny, tvoří součást hardware

**jednotky informace** —

**1 bit (Binary Unit)** – 1b

**1 slabika** – 1 byte – 8 bitů – 1B

**1 slovo** – 1 word – 2,4,8,6 bytů

**paměť** — zařízení pro uchovávání informací. Rozdělujeme ji

- podle funkce
  - RAM** – čtení a zápis
  - ROM** – pouze čtení
- podle přístupu
  - s **přímým přístupem** – disky, operační paměť
  - se **sekvenčním přístupem** – magnetická páska
- podle velikosti a doby přístupu
  - periferní paměť** – velká kapacita, dlouhá doba přístupu
  - operační paměť** – vždy s přímým přístupem
  - cache paměť**
  - registry** – několik bytů, s velmi krátkou dobou odezvy
- a na
  - virtuální**
  - fyzickou**

**řadič** – (controller) hardware převádějící instrukce na posloupnost signálů. Těmito signály se řídí zařízení připojená k řadiči.

**signály** jsou *číslicové (digitální)* nebo *spojité (analogové)*

## 1.2 Historie počítačů

Volně podle skript Jan Staudek, Vladimír Drábek — Počítačové systémy.

První **číslicovou kalkulačku** sestrojil v roce 1623 profesor W. Schickard. Blaise Pascal sestrojil v roce 1645 svoji kalkulačku "Pascaline", ze které se poučil Leibnitz a v roce 1671 navrhl kalkulačku, jejíž principy se používali dalších třista let.

Roku 1725 pan B. Bouchon použil pro řízení tkalcovského stavu **děrnou pásku**. Pan Falcet ji nahradil **děrnými štítky**.

Roku 1837 vynalezl S. F. B. Morse **telegraf**, který byl uveden do provozu mezi Washingtonem a Baltimorem roku 1844 a v roce 1858 byla Evropa spojena s Amerikou podmořským kabelem.

Praotcem **univerzálních počítačů** je Charles Babbage (1791-1871), který chtěl zkonstruovat stroj, který měl měnit průběh dalšího výpočtu v závislosti na výsledku předchozí operace. Příliš předběhl dobu a ani po čtyřicetiletém úsilí svůj Analytical Engine nedokončil.

**Děrnostřítkový stroj** sestrojil v roce 1888 Herman Hollerith pro sčítání lidu v USA. Jeho společnost Tabulating Machine Company později fúzovala a přejmenovala se na International Business Machine Corporation – IBM.

Plně fungující **programovatelné počítače** vznikly za druhé světové války v Berlíně, kde Konrád Zuse sestrojil v roce 1941 počítač Z3. Používal se pro balistické výpočty raket "V".

Harvardský fyzik Howard H. Aiken sestrojil **elektromagnetický počítač** Mark I řízený instrukcemi z děrné pásky. Řešil obyčejné diferenciální rovnice. Násobení trvalo 6 sekund. Zdokonalený Mark II násobil již za 0.4s.

Z vojenských důvodů vznikl v Pensylvánii v roce 1946 slavný **ENIAC**. Obsahoval 18 000 elektronek a chladily jej dvě letecké vrtule. Dokázal 5000 součtů za sekundu, násobení bylo 14krát pomalejší.

V té době byly stanoveny tři základní rysy počítačů, a to binární aritmetika, sekvenční zpracování instrukcí a společná paměť pro data i programy.

Následovaly počítače EDVAC (1945), BINAC (1950), UNIVAC (1951), IAS (1952), IBM604. Prvním **sériově vyráběným počítačem** byl počítač IBM model 701 (1952). V Československu byl první reléový počítač SAPO uveden do provozu na pracovišti ČSAV (VÚMS - Výzkumný Ústav Matematických Strojů) v roce 1958.

## 1.3 Generace počítačů

Generace	rok uvedení	konfigurace	rychlost	součástky
nultá	1940	velký počet skříní	jednotky operací	relé
1.	1950	desítky skříní	100 - 1000	elektronky
2.	1958	do deseti skříní	tisíce	tranzistory
3.	1964	kolem pěti skříní	desítky tisíc	integrované obvody
3.5	1972	jedna skříň	stovky tisíc	obvody větší integrace
4.	1981	jedna skříň	desítky miliónů	obvody větší integrace

## 2 Architektura počítačového systému

Počítač se skládá z

**CPU** – Central Processing Unit – procesor – je integrovaný obvod obsahující:

**ALU** – aritmeticko-logická jednotka. Obsahuje sčítačky, násobičky, střadače

**řadič** – na základě instrukcí vydává signály řídící ostatní části počítače

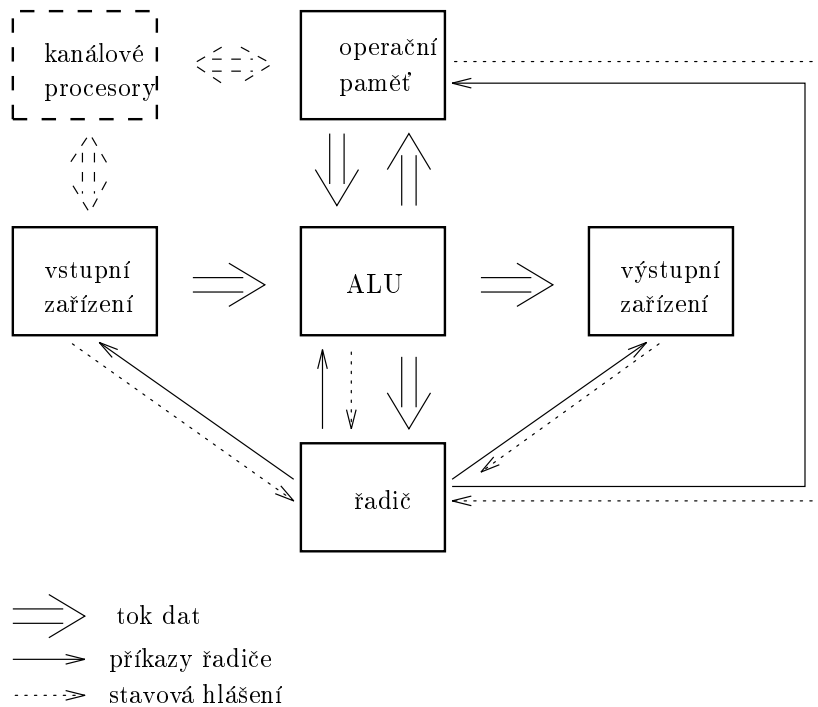
**registry** – zápisníková paměť. Několik málo velice rychlých paměťových buněk

**operační paměť** – středně velká adresovatelná paměť s přímým přístupem

**vstupně/výstupní zařízení**

Podle druhu komunikace mezi těmito částmi dělíme architektury počítačů na dva zásadně odlišné druhy:

- sběrnicové
- kanálové



Obrázek 1: Von Neumannova architektura počítače

U sběrnice architektury jsou všechny části navěšeny na jedné sběrnici, po které mohou komunikovat vždy jen dvě zařízení najednou, tedy např. procesor s pamětí nebo disk s pamětí, a ostatní zařízení musí čekat.

U kanálové architektury, která je mnohem dražší, jsou některé dvojice zařízení propojeny *kanálovým procesorem*, který umožňuje přenos dat i v době, kdy procesor pracuje. Například procesor dá instrukci pro přenos bloku dat kanálovému procesoru pro přenos mezi diskem a pamětí a zatímco kanálový procesor přenáší data, procesor pracuje na něčem jiném nezměněnou rychlostí.

Krok sběrnice architektury směrem ke kanálové je tzv. **DMA** – Direct Memory Access, což je zařízení také pověšené na společné sběrnici, které ale dokáže provádět přenosy mezi pamětí a diskem (nebo mezi pamětí a pamětí) rychleji než procesor. Takže procesor zadá DMA příkaz pro přenesení bloku dat a čeká na jeho dokončení, nicméně přenos bude rychlejší, než kdyby ho prováděl sám procesor.

Sběrnice se dělí na tři části, na

- řídicí
- adresovou
- datovou

Pro ilustraci – při zápisu do operační paměti procesor nastaví na adresové sběrnici adresu paměťové buňky, na datové sběrnici nastaví zapisovanou hodnotu a po řídicí sběrnici dá paměti povel k zápisu.

### 3 Číselné soustavy

Obecný algoritmus převodu:

Celá část:

```

číslo := celá_část;
  i := 0; {řád číslice}
základ := ...;

```

```

WHILE (číslo<>0) DO
  BEGIN
    číslice[i] := číslo MOD základ;
    číslo := číslo DIV základ;
    i := i+1;
  END

  Desetinná část:

číslo := desetinná_část;
základ := ...;
FOR i := 1 TO požadovaný_počet_míst DO
  BEGIN
    součin := číslo * základ;
    číslice[i] := TRUNC(součin);
    číslo := součin - číslice[i];
  END

```

## 4 Zobrazení dvojkového čísla v počítači

**rozsah zobrazení** – interval, který je omezený zleva nejmenším a zprava největším zobrazitelným číslem

**rozišitelnost** – nejmenší kladné nenulové zobrazitelné číslo

**přesnost** – pouze u reálných čísel

### 4.1 Zobrazení záporných čísel

**přímý kód** — jako kladná se záporným znaménkem. Má dvě nuly: +0, -0. Rozsah zobrazení  $\langle -2^{n-1} + 1; 2^{n-1} - 1 \rangle$ , např.  $\langle -127, -0 \rangle \langle +0; 127 \rangle$ .

**inverzní kód** — inverze všech bitů, zase dvě nuly

**dvojkový doplňkový kód** — dvojkový doplněk je inverze všech bitů a přičtení jedničky. Má jen jednu nulu, rozsah  $\langle -128, 127 \rangle$ .

Kódová kombinace		Význam		
		přímý	inverzní	doplňkový
0	0...00	0	0	0
0	0...01	1	1	1
0	0...10	2	2	2
...	...	...	...	...
0	1...11	+MAX	+MAX	+MAX
1	0...00	-0	-MAX	-MAX-1
1	0...01	-1	-MAX+1	-MAX
1	0...10	-2	-MAX+2	-MAX+1
1	0...11	-3	-MAX+3	-MAX+2
...	...	...	...	...
1	1...10	-MAX+1	-1	-2
1	1...11	-MAX	-0	-1

Sčítání v doplňkovém kódu:

- všechny bity se sčítají stejně
- přenos z nejvyššího bitu se ignoruje
- *přetečení* nastane tehdy, pokud se přenos *do* znaménkového bitu nerovná přenosu *ze* znaménkového bitu.

## 4.2 Zobrazení reálných čísel

### 4.2.1 zobrazení v pevné řádové čárce

*Měřítka* je posun mezi čárkou strojovou a čárkou logickou.

Př.  $0.^1101\ 100^2\ 000\ 000$  kde <sup>1</sup> je místo strojové čárky a <sup>2</sup> místo logické čárky. Měřítka je 6.

Pravidla:

- sčítat lze jen čísla se stejným měřítkem
- měřítko součinu je součet měřítek
- měřítko podílu je rozdílem měřítek

### 4.2.2 zobrazení v pohyblivé řádové čárce

Typ `real`. Číslo rozdělíme na *mantisu* a *exponent*. Mantisa je ve tvaru  $\pm 0.123$ . Číslo pak vyjádříme jako

$$\pm \textit{mantisa} \cdot Z^{\pm \textit{exponent}}$$

.

**Rozsah zobrazení**  $(-2^{+maxexp}, +2^{+maxexp})$

**Rozlišitelnost**  $+2^{-maxexp}$

**Přesnost** počet bitů mantisy

Normalizovaná mantisa – za strojovou čárkou musí být první významná číslice.

### 4.2.3 zobrazení logických hodnot

Typ `boolean`. Nejsnadněji testovatelný bit.

### 4.2.4 kód BCD

Binary coded decimal. Používá 4 bity na jednu desítkovou číslici.

**Rozvinutý tvar** (UNPACKED DECIMAL)

- mezitvar, nepoužívá se k výpočtům
- zónový tvar desítkového čísla
- zóna – horní půlslabika standardně  $F_{16}$ , pro kladná  $C_{16}$ , pro záporná čísla  $D_{16}$ .

	desítkově	rozv. tvar
Příklad:	$71346_{10}$	F7 F1 F3 F4 $C_{16}$
	$-71346_{10}$	F7 F1 F3 F4 $D_{16}$

**zhuštěný tvar** (PACKED DECIMAL)

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

### 4.3 Vnější kódy

Každý znak má svoji ordinální (pořadovou) hodnotu. Požadované vlastnosti přiřazení ordinálních hodnot znakům:

- zachovat abecední uspořádání
- snadný převod desítkových číslic na numerickou hodnotu

**ASCII** – American Standard Code for Information Interchange

**EBCDIC** – Extended Binary Coded Decimal Interchange Code

**detekční kódy** – zavedením nadbytečnosti (redundance) odhalují chyby způsobené vnějšími vlivy

**opravné kódy** – umí chybu i opravit

**Hammingova vzdálenost** – počet bitů, v nichž se liší dvě kódové kombinace

**Hammingův kód** – samoopravný kód

$I_7$	$I_6$	$I_5$	$K_4$	$I_3$	$K_2$	$K_1$
-------	-------	-------	-------	-------	-------	-------

$\oplus$  = XOR

$$K_1 = I_3 \oplus I_5 \oplus I_7$$

$$K_2 = I_3 \oplus I_6 \oplus I_7$$

$$K_4 = I_5 \oplus I_6 \oplus I_7$$

$$S_1 = K_1 \oplus I_3 \oplus I_5 \oplus I_7$$

$$S_2 = K_2 \oplus I_3 \oplus I_6 \oplus I_7$$

$$S_4 = K_4 \oplus I_5 \oplus I_6 \oplus I_7$$

syndrom ( $S_4 S_2 S_1$ ) je číslo sloupce s chybou

## 5 Logické obvody

Idea logických obvodů vychází z Booleovy algebry, konkrétní implementace pak využívají její speciální případy:

- SHEFFEROVA ALGEBRA – postavena na NAND pravidla:

$$\star \overline{xx} = \overline{x}$$

$$\star \overline{x0} = 1$$

$$\star \overline{x1} = \overline{x}$$

$$\star \overline{\overline{xy}1} = \overline{\overline{xy}} = xy$$

$$\star \overline{\overline{xx} \overline{yy}} = \overline{\overline{xy}} = x + y$$

- PEICEROVA ALGEBRA – postavena na NOR

**Technologie TTL** (Tranzistor-Tranzistor Logic)

vlastnosti TTL:

- řízena proudem
- napájecí napětí 5V
- $L < 0.8V$
- $H > 2.4V$
- velká rychlost

- velká spotřeba
- menší integrace
- dražší

**Unipolární technologie** má opačné vlastnosti:

- je řízena napětím,
- napájecí napětí 5,10,15V
- $L < 0.1V$
- $H = \text{napájecí napětí}$
- malá rychlost
- malá spotřeba
- nebezpečí statické elektřiny

Tranzistor MOS – Metal-Oxid-Semiconductor (hliník-izolant-polovodič), PMOS odpovídá PNP, NMOS odpovídá NPN, CMOS je oba.

## 5.1 Kombinační logické obvody

hodnota výstupu závisí pouze na momentální hodnotě vstupu

Základní logické členy:

- INVERTOR
- AND
- OR
- NAND
- NOR

Ostatní logické členy:

- XOR
- NOXOR

Logické obvody:

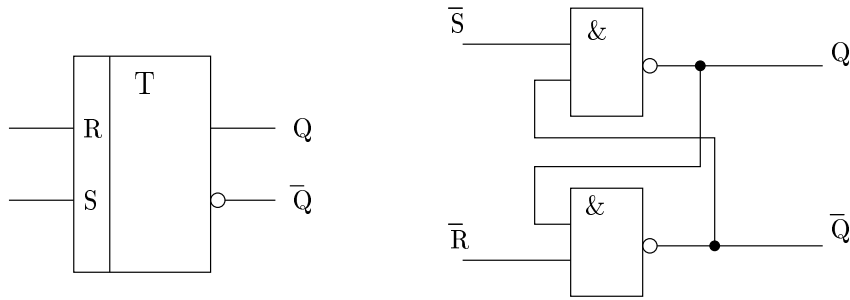
**MULTIPLEXOR** — adresové vstupy udávají, který datový vstup půjde na výstup

- dvouvstupový multiplexor – 2 datové vstupy  $X, Y$ ; jeden adresový  $A$ ; provádí funkci  $Z = A.X + \bar{A}.Y$
- čtyřvstupový multiplexor – 4 datové  $D_0, D_1, D_2, D_3$ , 2 adresové  $A_1, A_2$  provádí funkci  $Z = \bar{A}_1.\bar{A}_2.D_0 + \bar{A}_1.A_2.D_1 + A_1.\bar{A}_2.D_2 + A_1.A_2.D_3$

**DEKODÉR** — adresové vstupy udávají, který z výstupů se nastaví na 1

**KOMPARÁTOR** — jedna pokud jsou porovnávány byty stejné

**SČÍTAČKY** — kombinační sčítačka se implementuje jen pro jeden řád, vícemístné sčítačky sčítají ve více taktech - jsou sekvenční.



Obrázek 2: Klopný obvod řízený nulami

## 5.2 Sekvenční logické obvody

Výstupy nezáleží jen na momentální hodnotě vstupů, ale i na posloupnosti předcházejících změn.  
Základní paměťový člen – klopný obvod.

### 5.2.1 Klopný obvod RS

(SET/RESET) (nastavení/nulování)

- řízený jedničkami

R	S	$Q_i$	$\overline{Q}_i$
0	1	1	0
1	0	0	1
0	0	$Q_{i-1}$	$\overline{Q}_{i-1}$
1	1	zakázaný stav	

- řízený nulami

$\overline{R}$	$\overline{S}$	$Q_i$	$\overline{Q}_i$
1	0	1	0
0	1	0	1
1	1	$Q_{i-1}$	$\overline{Q}_{i-1}$
0	0	zakázaný stav	

### 5.2.2 Typické sekvenční obvody v počítačích

- sériová sčítačka
- paralelní registr – střadač
- posuvný registr (posuny bitů)
- čítače
- sčítačka BCD kódu
- násobičky

### 5.2.3 Bitové posuny

**ROTACE** — bity všechny rotují

**LOGICAL SHIFT** — bity se posouvají, do posledního se vkládá nula

**ARITHMETIC SHIFT** —

- doleva — znaménkový bit se nemění (=násobení x2)
- doprava — znaménkový bit se kopíruje do nižšího řádu (=dělení :2)

## 6 Paměti

- vnější paměti
- vnitřní paměti
- registry

Parametry pamětí:

- vybavovací doba (10ns až 100ms)
- rychlost toku dat
- kapacita paměti
- cena za bit
- přístup (přímý/sekvenční)
- destruktivnost při čtení
- energetická závislost a nezávislost
- statika a dynamika
- spolehlivost (v rozmezí teplot, na dobu provozu, na počet správných bitů)

Parametry aplikované na typech pamětí:

**vnější paměti** – vyb. doba velká, kapacita velká, cena nízká

**vnitřní paměti** – vyb. doba střední, kapacita menší, cena vyšší

**sada registrů** – (zápisníková paměť) kapacita byty, cena vysoká

**řídící paměť** – pro zaznamenání  $\mu$ -programů

**vyrovnávací paměť** – mezi procesorem a V/V zařízením

**CACHE** – mezi procesorem a pamětí

### Polovodičové paměti

bipolární TTL	1-10ns	1mW/bit
unipolární CMOS	10-100ns	0.1-0.01mW/bit

- SRAM -statická
- DRAM - dynamická, každé 2ms se musí přečíst, tím se obnoví hodnoty

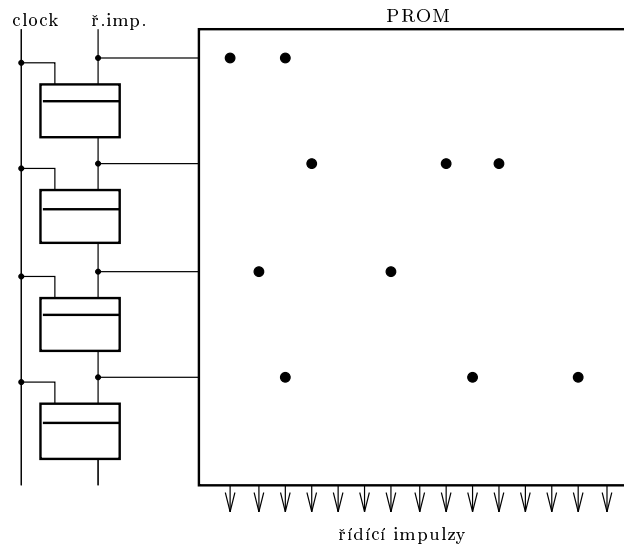
**Feritové paměti** Zápis koexistencí proudů V a Č/Z vodiče. Čtení – zápisem na V vodič se na Č/Z vodiči indukuje vysoké nebo nízké napětí — destruktivní čtení.

**Bublinové paměti** Nosič je nepohyblivá magnetická vrstva protkaná proudovými smyčkami, které realizují posuv informace.

## 7 Řadiče

### 7.1 Direktivní ovládání

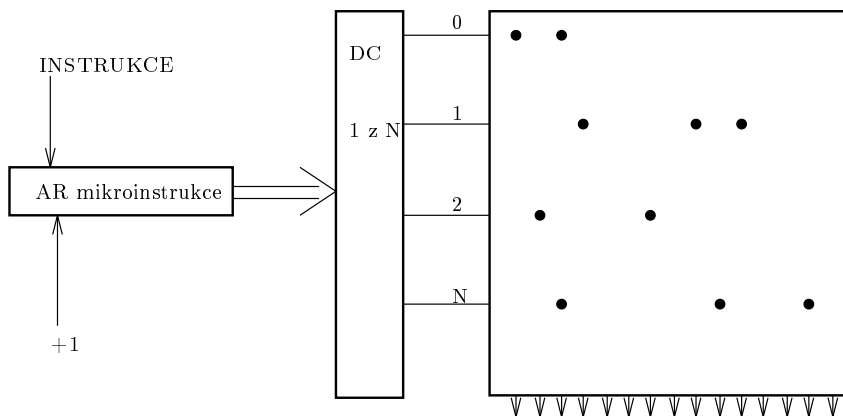
Nemá zpětnou vazbu, neměnný impulzní sled. Řadič sestává z permanentní paměti PROM, která je maticí propojení impulsů příšlých do řadiče a impulsů vysílaných řadičem.



Obrázek 3: Direktivní řadič

### 7.2 Mikroprogramový řadič

Instrukce je počáteční adresa mikroprogramu. Adresový registr AR obsahuje adresu právě prováděné mikroinstrukce, postupně se inkrementuje. DC je dekodér vybírající jeden z  $N$  výstupů podle adresy z AR. Mikroprogramový řadič je pomalejší než direktivní, ale programovatelný.

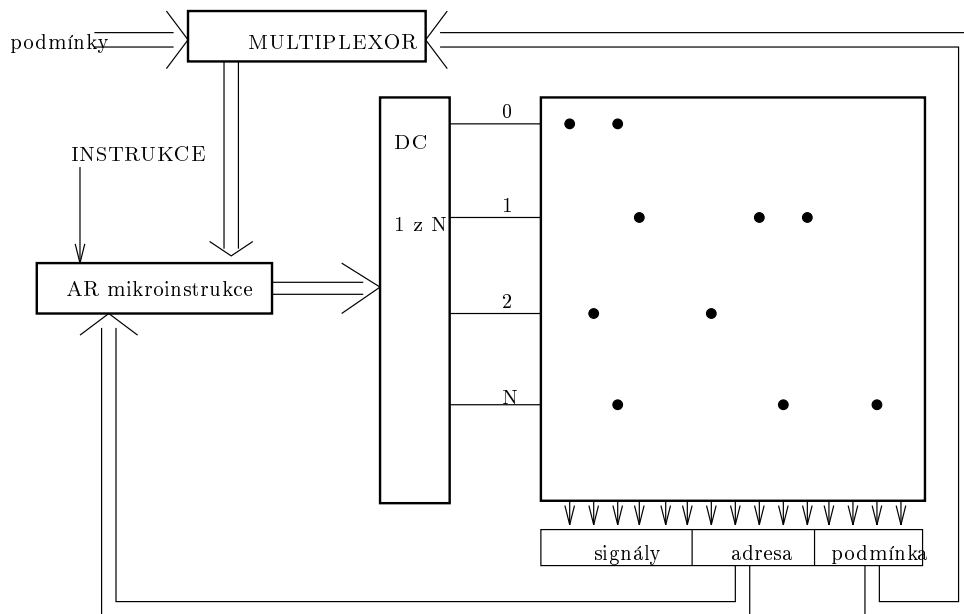


Obrázek 4: Mikroprogramový řadič

### 7.3 Zpětnovazební ovládání

V praxi průběh modifikují znaménka, stavy, ...

Je-li řadič mikroprogramový, vytváří nám hodnota podmínky novou adresu, na které bude řízení pokračovat.



Obrázek 5: Mikroprogramový řadič se zpětnou vazbou

## 8 Procesor

Procesor je synchronní stroj řízený řadičem.

**Takt procesoru** – základní synchronizační frekvence

**Strojový cyklus** – čas potřebný k zápisu slova do paměti

**Instrukční cyklus** – čas potřebný pro výběr a provedení instrukce

Fáze procesoru:

- výběr
  - ★ operačního kódu z paměti
  - ★ operandu z paměti
- provedení instrukce
- přerušení, ...

Výběr instrukcí je řízen registrem ČI,PC,IP (čítač instrukcí, program counter, instruction pointer), který obsahuje adresu právě prováděné instrukce. Po provedení instrukce se zvyšuje o délku instrukce.

## 8.1 Náš počítač

### 8.1.1 instrukce a registry

- pracuje ve dvojkovém doplňkovém kódu
- registry

A	střadač	16 bitů
PC	čítač instrukcí	16 bitů
SP	ukazatel zásobníku	16 bitů

- paměť 64kW
- instrukční soubor

<b>JMP</b> adresa	skok	$PC \leftarrow \text{adresa}$	JUMP
<b>LDA</b> adresa	načtení	$A \leftarrow (\text{adresa})$	LOAD A
<b>STA</b> adresa	uložení	$(\text{adresa}) \leftarrow A$	STORE A
<b>ADA</b> adresa	přičtení	$A \leftarrow A + (\text{adresa})$	ADD A
<b>CMA</b>	inverze	$A \leftarrow \bar{A}$	COMPLEMENT A
<b>ZERO</b> adresa	větvení	IF $A=0$ GOTO adresa	
<b>MINUS</b> adresa	větvení	IF (znam.bit $A=1$ ) $PC \leftarrow \text{adresa}$	

- interní registry procesoru

AR	adresový registr	drží adresu pro Č/Z z/do paměti
DR	datový registr	drží data pro Č/Z dat z/do paměti
IR	instrukční registr	dekódování instrukce

### 8.1.2 mikroinstrukce

Jaké mikroinstrukce se provedou při 200 LDA 101

- $PC \leftarrow 200$  (číslo instrukce)
- $PC \rightarrow AR$ ,  $data \rightarrow DR$ ,  $DR \rightarrow IR$  (výběr instrukce)
- $PC+1 \rightarrow AR$ ,  $data \rightarrow DR$ , (výběr adresy operandu)
- $DR \rightarrow AR$ ,  $data \rightarrow DR$  (výběr hodnoty operandu)
- $DR \rightarrow A$  (provedení)
- $PC \leftarrow PC+2$

### 8.1.3 Zásobník

<b>PUSH</b>	uložení na zásobník	obsah A uloží na zásobník
<b>POP</b>	vyzvednutí ze zás.	vršek zásobníku do A
<b>CALL</b> adresa	volání podprogramu	PC na zásobník, adresa do PC
<b>RET</b>	návrat z podprogramu	adresa ze zásobníku do PC

### 8.1.4 Vstupně/Výstupní operace

<b>OUT</b>	výstup	zapiše obsah A na sběrnici
<b>IN</b>	vstup	sběrnici zapiše do A
<b>START</b>	zahájí V/V operaci	
<b>FLAG</b> adresa	skok na adresu, není-li operace skončena	

### 8.1.5 systém přerušení

Umožňuje přerušit běžícího procesu a aktivuje rutinu pro obsluhu přerušení.

Činnost při přerušení:

- přerušení provádění programu
- úklid registrů
- provedení obslužné rutiny
- obnovení registrů a pokračování v procesu

Kdy lze přerušit proces:

- pouze po provedení instrukce, nikoliv během ní
- je-li to povoleno – příznak IF (interrupt flag) instrukce

<b>STI</b>	povoluje přerušení	1 → IF
<b>CLI</b>	zakazuje přerušení	0 → IF

Instrukce jsou rozděleny na

- privilegované – STI, CLI, IN, OUT, START, FLAG
- neprivilegované – LDA, ADA, JMP, ZERO

Proces nelze přerušit bezprostředně po zahájení obsluhy předchozího přerušení. Přerušení se vyvolá signálem INTERRUPT na řídicí sběrnici.

Třídy přerušení

- I/O
- chybný výsledek instrukce
- vnější
- programová – volání služeb systému apod.
- hardwarové chyby počítače

## 9 Řada procesorů Intel 86

Podrobnosti jsou v knize

**Michal Brandejs: mikroprocesory INTEL 8086-80486, Grada 1991.** Shrnu zde jen základní rozdíly:

**I8086** –

- rok 1978
- 16bitový mikroprocesor
- 20bitová adresová sběrnice ⇒ paměť 1MB

**I80286** –

- rok 1983
- 16bitový mikroprocesor
- dva módy
  - ★ reálný – plně slučitelný s I8086, 1MB paměti
  - ★ chráněný – 16MB fyzické paměti, 1GB virtuální

- paralelní předzpracování instrukcí

#### I80386 –

- rok 1985
- 32bitový procesor
- tři módy
  - ★ reálný – plně slučitelný s I8086, 1MB paměti
  - ★ chráněný – 4GB fyzické paměti, 64TB virtuální
  - ★ virtuální 8086 – jako jeden z procesů v chráněném režimu

#### I80486 –

- rok 1989
- 32bitový procesor
- plně slučitelný s I80386
- 3x až 5x rychlejší než I80386
- integrovaný matematický koprocesor

## 10 Správa paměti

### 10.1 CACHE paměť

CACHE paměť je rychlá paměť s malou kapacitou vázaná mezi pomalou operační paměť s velkou kapacitou a rychlý procesor. Problém je, který z bloků CACHE paměti vyměnit za nový při jejím přeplnění.

Používá se algoritmus LRU (Least Recently Used – nejdéle nepoužívaný blok). Každý blok v CACHE paměti se doplní čítačem, který se nuluje při použití daného bloku a inkrementuje při použití jiného bloku.

Druhá možnost je použití neúplnou maticí s prvky pouze nad hlavní diagonálou. Každý prvek je jednobitová paměť, při volání  $i$ -tého bloku se jedničkuje  $i$ -tý řádek a nuluje  $i$ -tý sloupec. Nejdéle nepoužívaný blok pak má v řádku nuly a ve sloupci jedničky.

### 10.2 Stránkování operační paměti

používá se u počítačů s kratším prostorem pro uložení adresy (10b) než je adresový prostor (16b). Adresování v celém rozsahu se pak zajistí

- doplněním horních bitů z PC
- nepřímou adresací (ADT)

### 10.3 Mapování operační paměti

umožňuje adresování více paměti, než povoluje adresový prostor počítače. Logická adresa se rozdělí na *adresu logické stránky* a na *adresu slova ve stránce*. Adresa logické stránky se pak použije jako ukazatel do tabulky adres fyzických stránek, které mají více bitů než adresy logických stránek. Fyzická adresa se pak složí z adresy fyzické stránky a adresy slova ve stránce.

### 10.4 Virtuální paměť

prostředek, jak u počítačů s dostatečnou šířkou adresy zajistit možnost adresace plné kapacity bez celé paměti. U stránkování a mapování mám malou šířku adresy a dostatek fyzické paměti, u virtualizace naopak mám dostatečnou šířku adresy, ale málo fyzické paměti.

Nemohu zde nenapsat, jak vznikla virtuální paměť. To měl takhle jednou do Kocourkova přijet Císařpán. Taková důležitá osoba musí jít z kočáru ke vchodu radnice po červeném koberci, jenže jako na potvoru v Kocourkově neměli

dostatečně dlouhý červený koberec. Nejdelší červený koberec, který měli, stačil tak na desetinu cesty. Sešli se páni radní, dumali, až jeden z nich dostal spásný nápad. Ten krátký koberec rozstříhli na několik dílů, které položili za sebe od kočáru směrem k radnici. Císařpán vystoupil a šel po koberci. Vždycky, když přešel přes další úsek koberce, tak ten kus vzali a přenesli před Císařpána a koberec nadstavili. Tak Císařpán šel po celou dobu po červeném koberci, i když neměli koberec na celou délku cesty.

Tenhle příběh je volně převyprávěný z manuálu z VÚMSu k počítači EC1027. Virtuální paměť funguje úplně stejně jako Kocourkovský koberec. Fyzická i virtuální paměť jsou rozděleny na bloky. Celá virtuální paměť je na disku a ve fyzické paměti jsou jen ty bloky virtuální paměti, které se zrovna používají.

## 11 Klasifikace paralelních procesorů

- počet zpracovávaných programů (současně)
  - ★ SI (single instruction)
  - ★ MI (multiple instruction)
- počet zpracovávaných množin dat
  - ★ SD (single data)
  - ★ MD (multiple data)

**SISD** – von Neumann

**MISD** – více procesorů nad stejnými daty – odolnost proti chybám

**SIMD** – maticové procesory

**MIMD** – obecný multiprocessorový paralelní systém

## 12 Periferie

### 12.1 Záznam na magnetické vrstvy

#### 12.1.1 magnetická páska

šířka 18.7mm

tloušťka podkladu 0.038mm, magnetické vrstvy 0.015mm

hustota 800-1600bpi

#### 12.1.2 magnetické disky

Kotouč z nemagnetického materiálu pokrytý magnetickou vrstvou. Disk má více *ploch*, na každé jsou jako soustředné kružnice *stopy* rozdělené na *sektory*. U disku trvá nejdéle vystavení hlaviček nad správnou stopu, doba čekání na po-otočení disku je krátká.

#### 12.1.3 diskety

**disketa 8"** –

**SSSD** – stopy 0-76, sektory 1-26, sektor 128B, **kapacita 250kB**

**SSDD** – stopy 0-76, sektory 1-52, sektor 128B, **kapacita 500kB**

**DSDD** – dva povrchy, bf kapacita 1MB

**disketa 5 $\frac{1}{4}$ "** –

**DSDD** – strany 0-1, stopy 0-39, sektory 1-9, sektor 512B, **kapacita 360kB**

**DSQD** – strany 0-1, stopy 0-79, sektory 1-9, sektor 512B, **kapacita 720kB**

**DSHD** – strany 0-1, stopy 0-79, sektory 1-15, sektor 512B, **kapacita 1.2MB**

**disketa 3.5"** –

**DSDD** – strany 0-1, stopy 0-79, sektory 1-9, sektor 512B, **kapacita 720kB**

**DSHD** – strany 0-1, stopy 0-79, sektory 1-18, sektor 512B, **kapacita 1.44MB**

#### 12.1.4 optické diskové paměti

1974 – compact disc "CD"

1982 – laserové přehrávače na trhu

- CD-ROM zapsané výrobcem
- WORM jedenkrát zapisovatelné
- mazatelné optické paměti

## 12.2 Tiskárny

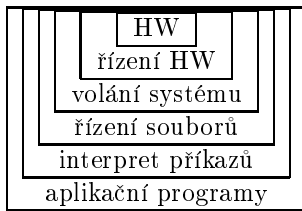
- úderové
  - ★ paralelní
    - s typovým válcem
    - s typovým řetězem
  - ★ sériové
    - bodové (maticové, jehličkové)
    - s typovou růžicí
- bezúderové
  - ★ laserové
  - ★ inkoustové
  - ★ tepelné

## 12.3 ostatní

- myš
- světelné pero
- trackball
- scanner
- digitizér a tablet
- hlasový vstup
- snímač čárkového kódu
- monitor

## 13 Struktura a funkce OS

### 13.1 Strukturální popis



### 13.2 základní funkce řídicích programů

- ŘÍZENÍ ÚLOH (Shell)
  - ★ zpracování příkazů uživatele
  - ★ práce se systémem souborů
  - ★ vyvolání aplikačního programu
- ŘÍZENÍ VÝPOČTU
  - ★ řízení paměti
  - ★ zavedení apl. programu do paměti
  - ★ komunikace s operátorem
  - ★ zprostředkování styku s HW
  - ★ souběžné zpracování programů
    - hospodaření se společnými prostředky
    - vzájemná ochrana paměti, souborů
    - vzájemná spolupráce (sdílení paměti, souborů)
    - ošetření chyb (HW, programů)
    - normální/abnormální ukončení
- ŘÍZENÍ SOUBORŮ – implementace přístupových metod

### 13.3 základní aplikační programy

- údržba knihoven
- překladače (compiler)
- spojovací program (linker)
- údržba disků (norton commander)
- konverze souborů (filtry, sort)
- sledování a testování (debugger)
- programovací systémy (TurboPascal)