

# ARP – architektura počítačů

Jakub Nečásek ([jakub.necasek@tul.cz](mailto:jakub.necasek@tul.cz))

podmínky pro zápočet:

- vypracování samostatné úlohy
- aktivní účast na cvičeních – 2 povolené absence
- literatura – viz syllabus
- dotazy...?

# Výkonnost počítačů, Amdahlův zákon, výkonnostní rovnice CPU

# Výkonnost počítačů

---

Základní požadavek kladený na počítač je schopnost provádět zpracování informací. Tuto schopnost označujeme jako **výkonnost počítače**.

Výkonnost je obtížné hodnotit jediným číslem – objektivnější je použít tzv. **vektor výkonnosti**, jehož struktura se vyvíjí.

Základem bývá **počet operací (příp. instrukcí) za sekundu**, bud' v pevné nebo pohyblivé řádové čárce.

Dalšími složkami mohou být **propustnost systému**, doba odezvy, stupeň využití, aj.

Hodnocení výkonnosti by mělo být podkladem pro optimalizaci

# Výkonnost a propustnost systémů

---

**Výkonnost  $P_T(T)$**  – inverzní hodnota doby  $T$  provedení jednoho úkonu (programu)

$$P_T(T) = \frac{1}{T}$$

**Propustnost  $P_R(n, T)$**  – počet  $n$  úkonů (úloh) za čas  $T$

$$P_R(n, T) = \frac{n}{T}$$

„úkonem“ často bývá instrukce, ale problém je v různých instrukčních souborech

# Metriky výkonnosti

---

**MIPS** (Million Instructions Per Second)

**MOPS** (Million Operations Per Second)

**MFLOPS** (Million FLoating point Operations Per Second)

$$P_{MIPS} = \frac{IC}{T_{CPU}} \times 10^{-6} \quad IC \dots \text{počet instrukcí}$$

$$P_{MOPS} = \frac{OC}{T_{CPU}} \times 10^{-6} \quad T_{CPU} \dots \text{doba výpočtu}$$

$$P_{MFLOPS} = \frac{OC_{FP}}{T_{CPU}} \times 10^{-6} \quad OC \dots \text{počet operací}$$

$OC_{FP} \dots \text{počet operací}$   
v pohyblivé řádové čárce

užívají se také GIPS (BIPS), GFLOPS, TFLOPS, PFLOPS ...

# Doba jedné instrukce

---

**Výkonnost:**  $P = 1/T_p$  [MIPS,  $\mu$ s]

$T_p$  ... čas potřebný na provedení jedné průměrné strojové instrukce

Pro zjištění  $T_p$  je třeba sestavit tabulku četnosti výskytu jednotlivých instrukcí při „běžném provozu“ počítače. Každá instrukce má přiřazenu svoji váhu  $a_i$ , která vyjadřuje pravděpodobnost výskytu instrukce v programu.

$$T_p = \frac{\sum_{i=1}^n a_i t_i}{\sum_{i=1}^n a_i} \quad [s]$$

$t_i$  ... doba provádění  $i$ -té instrukce  
 $a_i$  ... váha  $i$ -té instrukce  
 $n$  ... počet instrukcí zařazených do mixu

# Dhystone, Whetstone, DMIPS

---

Všechny instrukce netrvají stejně dlouho => lépe určit trvání dané úlohy, kolik udělá CPU práce.

**Dhystone** – test používající jen celočíselné operace;

**Whetstone** – test používající operace v pohyblivé řád. čárce;

**DMIPS** – udává, kolik umí CPU spočítat Dhystone za sekundu (udává se v miliónech) – DMIPS nelze jednoznačně přepočítat na MIPS (závisí na tom, kolik instrukcí potřebuje daný CPU na výpočet Dhystone (DMIPS nezávisí na architektuře – oproti MIPS));

**DMIPS/MHz** – udává jaký výkon má CPU při 1 MHz.

# Výkonnost DSP procesorů

---

Signálové procesory jsou orientované na optimální provádění součtu součinů (číslicová filtrace, FFT, ...)

Výkon se často uvádí v počtu provedených akumulovaných součinů:  $A = A + (B \times C)$ , v jednotkách **MMAC/s** (Million Multiply Accumulates per Second) - milion násobení a mezisoučtů za sekundu.

Někdy se redukuje na **max. teoretický výkon**  
= počet násobiček  $\times$  max. hod. frekvence násobičky

Další používanou jednotkou výkonnosti je **MSPS** (Million Samples Per Second)

$$\text{MSPS} = \frac{\text{max. hod. frekvence}}{\text{počet hod. taktu na vzorek}}$$

# Amdahlův zákon

---

Významný zákon informatiky

Popisuje výpočet výkonového zisku (zrychlení  $S$ ) dosaženého vylepšením nějaké části počítače.

**Zrychlení  $S$**  je číslo, které udává kolikrát je rychlejší běh úlohy na počítači s vylepšením oproti běhu stejné úlohy na původním počítači.

$$S = \frac{\text{výkonnost při využití vylepšení}}{\text{výkonnost bez využití vylepšení}} = \frac{P_{NEW}}{P_{OLD}}$$

$$S = \frac{\text{doba výpočtu bez využití vylepšení}}{\text{doba výpočtu při využití vylepšení}} = \frac{T_{OLD}}{T_{NEW}}$$

# Poměry $F_E$ a $S_E$

---

Definujme si poměry:

$F_E$  ... udává, jakou část výpočtu lze vylepšit

$$F_E = \frac{\text{původní doba výpočtu zlepšené části úlohy}}{\text{původní celková doba výpočtu}} \leq 1$$

$S_E$  ... udává, kolikrát se zrychlil výpočet zlepšené části úlohy

$$S_E = \frac{\text{původní doba výpočtu zlepšené části úlohy}}{\text{doba výpočtu zlepšené části úlohy}} > 1$$

# Doba výpočtu

---

Doba výpočtu na vylepšeném počítači se bude skládat z:

$(1 - F_E) T_{OLD}$  = doba výpočtu té části úlohy, kterou nelze vylepšit

$\frac{F_E}{S_E} T_{OLD}$  = doba výpočtu vylepšené části úlohy

Tedy **doba výpočtu**  $T_{NEW}$  na vylepšeném počítači je:

$$T_{NEW} = T_{OLD} \left( (1 - F_E) + \frac{F_E}{S_E} \right)$$

# Celkové zrychlení (Amdahlův z.)

---

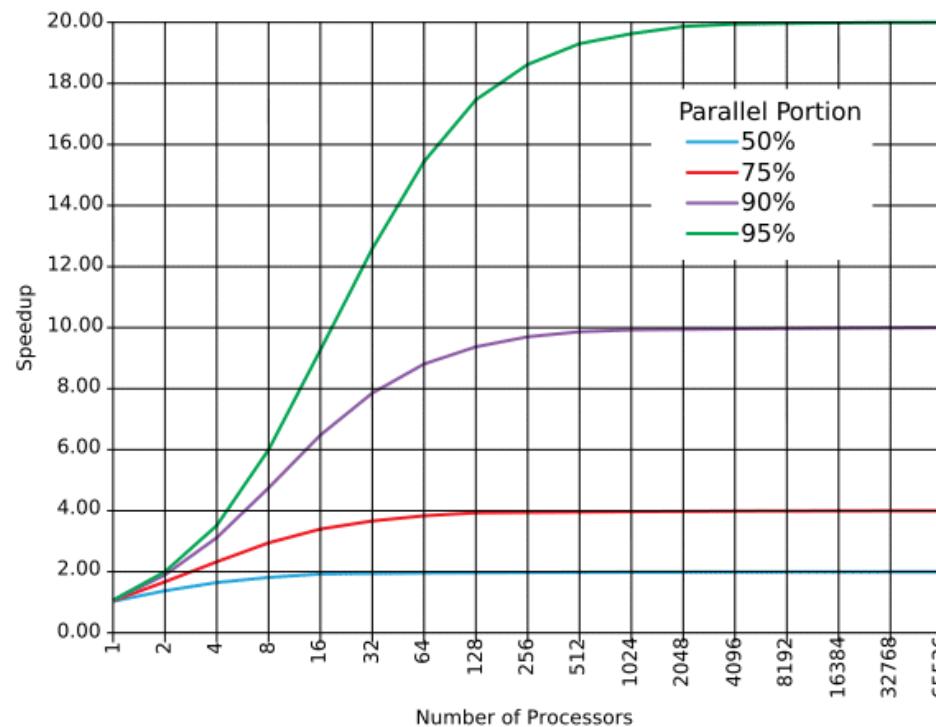
**Celkové zrychlení**  $S_{OVERALL}$  odpovídající danému vylepšení:

$$S_{OVERALL} = \frac{T_{OLD}}{T_{NEW}} = \frac{1}{(1 - F_E) + \frac{F_E}{S_E}}$$

# Zrychlení víceprocesor. systémů

---

Vždy je určitá část výpočtu provedena sekvenčně (pokud nelze úloha výrazně paralelizovat, víceprocesorové systémy nepřinesou zrychlení)



# Amdahlův z. pro víceproc. systémy

Celkové zrychlení výpočtu multiprocesorového systému:

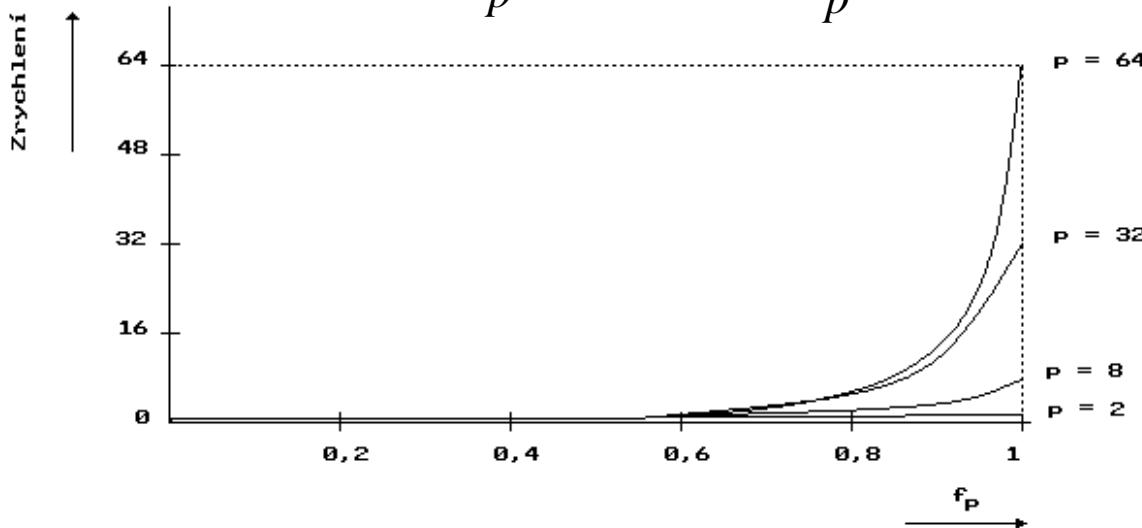
$p$  počet procesorů

$f_s$  část programu proveditelná jen jediným procesorem

$f_p$  paralelizovatelná část výpočtu

$$f_s + f_p = 1$$

$$S = \frac{1}{f_s + \frac{(1-f_s)}{p}} = \frac{1}{(1-f_p) + \frac{f_p}{p}}$$



# Instrukční parallelismus - zrychlení

---

U superskalárních architektur lze využít Amdahlův zákon pro víceprocesorové systémy, který zjednodušeně předpokládá části algoritmu, které lze/nelze paralelizovat.

Zavedením stupně  $p_x$  paralelizace jednotlivých částí (zde dvou) algoritmu můžeme zobecnit:

$$S_p = \frac{1}{\frac{1-f}{p_1} + \frac{f}{p_2}}$$

# Paralelismus algoritmu

---

**Zrychlení** (speedup)  $S(n,p)$

poměr doby výpočtu nejlepšího známého sekvenčního algoritmu a doby výpočtu paralelního algoritmu na témže (paralelním) počítači, využíváme-li  $p$  procesorů

$$S = \frac{T_{\text{serial}}}{T_{\text{parallel}}}$$

**Paralelní účinnost** (efficiency)  $E(n,p)$

jedná se o zrychlení dělené počtem použitých procesorů

$$E = \frac{S}{p}$$

# Výkonnostní rovnice procesoru

---

Výkonnost CPU závisí na:

- počtu instrukcí ( $IC$  – Instruction Count)
- (průměrném) počtu taktů na instrukci ( $CPI$  – Cycles Per Instruction)
- periodě hodinového signálu ( $T_{clk}$ ) – doba cyklu (taktu)

Doba provádění programu  $T_{CPU}$  je dána počtem hod. cyklů během programu násobená dobou cyklu  $T_{clk}$  (je-li konst.)

$$T_{CPU} = IC \times CPI \times T_{clk} \quad (\text{platí pro systémy bez cache})$$

$$P_{MIPS} = \frac{IC}{T_{CPU}} \times 10^{-6} = \frac{10^{-6}}{CPI \times T_{clk}} = \frac{f_{clk}}{CPI} \times 10^{-6}$$

# Přesnější výkonnostní rovnice

---

$$T_{CPU}(prg) = \sum (IC_i \times CPI_i) \times T_{clk}$$

Součet se provádí přes všechny instrukce z architektury instrukčního souboru

$T_{CPU}(prg)$  ... doba provádění programu  $prg$  procesorem

$IC_i$  ... počet provedení instrukcí  $i$  programu  $prg$

$CPI_i$  ... (průměrný) počet hodinových cyklů instrukce  $i$

# Vlivy na výkonnostní rovnici

---

	IC	CPI	T <sub>clk</sub>
Program	X		
Překladač	X	X	
ISA	X	X	X
Architektura počítače		X	X
Technologie výroby			X

ISA (Instruction Set Architecture) – architektura souboru instrukcí

# Příklad 1

---

Počítač zpracovává program, který má 5 milionů 1-CPI (jednotaktových instrukcí), 1 milion 2-CPI a 1 milion 3-CPI. Kmitočet hodinových taktů je 100 MHz. Jaká je jeho výkonnost v MIPS?

# Příklad 1

---

Počítač zpracovává program, který má 5 milionů 1-CPI (jednotaktových instrukcí), 1 milion 2-CPI a 1 milion 3-CPI. Kmitočet hodinových taktů je 100 MHz. Jaká je jeho výkonnost v MIPS?

$$IC = 5 \times 10^6 + 1 \times 10^6 + 1 \times 10^6 = 7 \times 10^6$$

$$N_{clk} = 5 \times 10^6 + 2 \times 10^6 + 3 \times 10^6 = 10 \times 10^6$$

Potřebný čas:  $T_{CPU} = \frac{N_{clk}}{f_{clk}} = \frac{10 \times 10^6}{100 \times 10^6} = 0.1 \text{ s}$

$$P_{MIPS} = \frac{IC}{T_{CPU}} \times 10^{-6} = \frac{7 \times 10^6}{0.1} \times 10^{-6} = 70 \text{ MIPS}$$

## Příklad 2

---

Výpočetní úloha je rozdělena na 4 části, z nichž každá trvá daný čas ( $P_1 = 11\%$ ,  $P_2 = 18\%$ ,  $P_3 = 23\%$  a  $P_4 = 48\%$ ). Jaké je celkové zrychlení, jestliže se nám část  $P_1$  nepodaří zrychlit, část  $P_2$  zrychlíme  $5\times$ , část  $P_3$  zrychlíme  $20\times$  a část  $P_4$  zrychlíme  $1,6\times$ ?

## Příklad 2

---

Výpočetní úloha je rozdělena na 4 části, z nichž každá trvá daný čas ( $P_1 = 11\%$ ,  $P_2 = 18\%$ ,  $P_3 = 23\%$  a  $P_4 = 48\%$ ). Jaké je celkové zrychlení, jestliže se nám část  $P_1$  nepodaří zrychlit, část  $P_2$  zrychlíme  $5\times$ , část  $P_3$  zrychlíme  $20\times$  a část  $P_4$  zrychlíme  $1,6\times$ ?

$$T_{OLD} = 1$$

$$T_{NEW} = \frac{0,11}{1} + \frac{0,18}{5} + \frac{0,23}{20} + \frac{0,48}{1,6} = 0,4575$$

$$S_{OVERALL} = \frac{T_{OLD}}{T_{NEW}} = \frac{1}{0,11 + \frac{0,18}{5} + \frac{0,23}{20} + \frac{0,48}{1,6}} \cong 2,19$$

(v podstatě zobecněný Amdhalův zákon)

## Příklad 3

---

Jak se zrychlí výpočet dvouprocesorového systému, jestliže 80% výpočetního algoritmu lze paralelizovat?

## Příklad 3

---

Jak se zrychlí výpočet dvouprocesorového systému, jestliže 80% výpočetního algoritmu lze paralelizovat?

$$S = \frac{1}{(1-f_p) + \frac{f_p}{p}} \quad \text{pro } p = 2 \text{ a } f_p = 0,8$$

$$S = \frac{1}{(1 - 0,8) + \frac{0,8}{2}} = 1,67$$

## Příklad 4

---

Předpokládejme vylepšení procesoru pro web. Nový CPU je  $10\times$  rychlejší pro webové aplikace než nynější. Dále víme, že nyní je CPU zaměstnán ze 40% výpočty a 60% času čeká na I/O operace. Jaké bude celkové zrychlení po plánovaném vylepšení?

## Příklad 4

---

Předpokládejme vylepšení procesoru pro web. Nový CPU je  $10\times$  rychlejší pro webové aplikace než nynější. Dále víme, že nyní je CPU zaměstnán ze 40% výpočty a 60% času čeká na I/O operace. Jaké bude celkové zrychlení po plánovaném vylepšení?

$$F_E = 0,4 \quad \text{část, kterou lze zlepšit} (\leq 1)$$

$$S_E = \frac{1}{0,1} = 10 \quad \text{kolikrát se zrychlil výpočet zlepšené části}$$

$$S_{OVERALL} = \frac{1}{(1 - F_E) + \frac{F_E}{S_E}} = \frac{1}{(1 - 0,4) + \frac{0,4}{10}} = 1,56$$

## Příklad 5

---

Předpokládejme, že při FP výpočtech v programu, operace odmocniny FPSQRT odpovídá 20% a všechny FP instrukce odpovídají 50% doby výpočtu úlohy. Úkolem je rozhodnout, zda je výhodnější  $10\times$  zrychlit provádění operace FPSQRT nebo  $1,6\times$  zrychlit provádění všech FP instrukcí.

## Příklad 5

---

Předpokládejme, že při FP výpočtech v programu, operace odmocniny FPSQRT odpovídá 20% a všechny FP instrukce odpovídají 50% doby výpočtu úlohy. Úkolem je rozhodnout, zda je výhodnější  $10\times$  zrychlit provádění operace FPSQRT nebo  $1,6\times$  zrychlit provádění všech FP instrukcí.

$$S_{FPSQRT} = \frac{1}{(1-0,2)+\frac{0,2}{10}} = 1,22 \quad F_E = 0,2 \quad S_E = 10$$

$$S_{FP} = \frac{1}{(1-0,5)+\frac{0,5}{1,6}} = 1,23 \quad F_E = 0,5 \quad S_E = 1,6$$

⇒ výhodnější je tedy zrychlit vše i když jen „nepatrнě“

## Příklad 6

---

DSP pracuje s frekvencí 80 MHz a k provedení jedné instrukce vyžaduje 4 hodinové takty. Během jedné instrukce udělá 2 FP aritmetické operace (součet a součin), 3 přístupy do paměti (přečte 2 operandy a uloží výsledek), obnoví jeden FP registr a inkrementuje 3 adresové ukazatele (čítače). Jaký je výkon procesoru v MIPS a MOPS?

## Příklad 6

---

DSP pracuje s frekvencí 80 MHz a k provedení jedné instrukce vyžaduje 4 hodinové takty. Během jedné instrukce udělá 2 FP aritmetické operace (součet a součin), 3 přístupy do paměti (přečte 2 operandy a uloží výsledek), obnoví jeden FP registr a inkrementuje 3 adresové ukazatele (čítače). Jaký je výkon procesoru v MIPS a MOPS?

$$P_{MIPS} = \frac{f_{clk}}{CPI} \times 10^{-6} = \frac{80}{4} = 20 \text{ MIPS}$$

$$\begin{array}{lcl} 2 \text{ FP operace} & & = 40 \text{ MOPS} \end{array}$$

$$\begin{array}{lcl} 3 \text{ přístupy do paměti} & & = 60 \text{ MOPS} \end{array}$$

$$\begin{array}{lcl} \text{obnova FP registru} & & = 20 \text{ MOPS} \end{array}$$

$$\begin{array}{lcl} 3 \text{ inkrementace} & & = 60 \text{ MOPS} \end{array} \quad \text{CELKEM tedy } \underline{180 \text{ MOPS}}$$

## Příklad 7

---

Přepokládejme, že 32% algoritmu lze paralelizovat stupněm 2, 45% algoritmu lze paralelizovat stupněm 5 a zbývající část algoritmu stupněm 6. Jaké je celkové zrychlení?

## Příklad 7

---

Přepokládejme, že 32% algoritmu lze paralelizovat stupněm 2, 45% algoritmu lze paralelizovat stupněm 5 a zbývající část algoritmu stupněm 6. Jaké je celkové zrychlení?

$$S = \frac{1}{\frac{0,32}{2} + \frac{0,45}{5} + \frac{0,23}{6}} \cong 3,47$$

## Příklad 8

---

Nejlepší sekvenční algoritmus se počítá na uvažovaném počítači 10 s, výpočet hodnoceného paralelního algoritmu pro 4 procesory trvá 5 s. Jaké je zrychlení a jaká je paralelní účinnost?

## Příklad 8

---

Nejlepší sekvenční algoritmus se počítá na uvažovaném počítači 10 s, výpočet hodnoceného paralelního algoritmu pro 4 procesory trvá 5 s. Jaké je zrychlení a jaká je paralelní účinnost?

$$S = \frac{T_{OLD}}{T_{NEW}} = \frac{10}{5} = 2$$

$$E = \frac{S}{p} = \frac{2}{4} = 0,5 \Rightarrow 50\%$$

# Zobrazování čísel v počítačích

# Čísla v počítačích

---

Čísla v počítačích je třeba ukládat do logických obvodů (registrů, pamětí, ...), příp. přenášet po sběrnicích  
⇒ užívání dvojkové soustavy (polyadická soustava o základu  $z = 2$ ):

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

Polyadické soustavy zobrazují pouze nezáporná čísla  
⇒ k zobrazování záporných čísel používáme transformace (*číselné kódy*); nejčastější:

- přímý se znaménkem
- inverzní
- doplňkový
- aditivní

# Řádová mřížka

---

Rozdělení na celou a zlomkovou část označujeme jako *řádová mřížka*, číslo zapsané v mřížce je *slovo*

Nelze-li zapsat číslice v nejvyšších řádech, mluvíme o *přeplnění* (přetečení, overflow), v nejnižších řádech o *ztrátě přesnosti*

$n-1$	$n-2$		$0$	$-1$			$-m$
-------	-------	--	-----	------	--	--	------

$n-1$  ... nejvyšší řád řádové mřížky (celá část má velikost  $n$  bitů)

$-m$  ... nejnižší řád řádové mřížky (zlomková část má velikost  $m$  bitů)

$N$  ... délka řádové mřížky (počet obsažených řadů)  $N = n + m$

$\varepsilon = 2^{-m}$  ... jednotka řádové mřížky (nejmenší zobrazitelné číslo)

$M = 2^n$  ... modul řádové mřížky (nejmenší číslo, které již v řádové mřížce není zobrazitelné)

# Formáty nezáporných čísel

---

Čísla bez znaménka (**unsigned**)

- *Celočíselný formát* (tj.  $m = 0, N = n$ ):

$$U = u_{n-1} \cdot 2^{n-1} + \dots + u_0 \cdot 2^0$$

- *Zlomkový (fraction) formát* – řádová čárka se umísťuje těsně za nejvyšší bit, který je řádu  $2^0$  ( $n = 1, N = m+1$ )

$$U = u_0 \cdot 2^0 + u_{-1} \cdot 2^{-1} + \dots + u_{-m} \cdot 2^{-m}$$

Z pohledu implementace aritmetických operací nezáleží na tom, kde je umístěna řádová čárka.

např.  $01001 + 00100 = 01101$

$$9 + 4 = 13 \quad \text{nebo} \quad 0,5625 + 0,25 = 0,8125$$

# Příklad

---

Převeďte číslo  $(258,125)_{10}$  do dvojkové soustavy

# Příklad

---

$$(258,125)_{10} : 2 = 0100000010,001$$

129

64

32

16

8

4

2

1

$$(0,125)_{10} \times 2 =$$

0,25

0,5

# Příklad

---

Převeďte číslo  $(1,1)_{10}$  do dvojkové soustavy

# Příklad

---

$$(1,1)_{10} : 2 = 1\textcolor{red}{|}0001101 \Rightarrow$$
$$\Rightarrow (1 + 0,0625 + 0,03125 + 0,0078125 = 1,1015625)_{10}$$

$$(0,1)_{10} \times 2 =$$

0,2

0,4

0,8

0,6

0,2

0,4

0,8... zaokrouhleno nahoru

# Příklad

---

Sečtěte  $(213)_{10}$  a  $(213)_{10}$

# Příklad

---

Sečtěte  $(213)_{10}$  a  $(213)_{10}$

$$\begin{array}{r} (213)_{10} = & 1101 \ 0101 \\ + (213)_{10} = & \underline{1101 \ 0101} \\ & 11010 \ 1010 = (426)_{10} \end{array}$$

=> násobení dvěma je posun o 1 bit vlevo a zvětšuje počet bitů o jeden

# Přímý kód se znaménkem

---

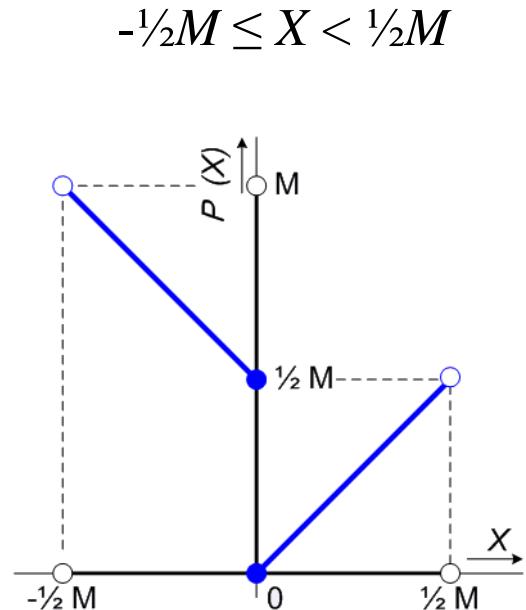
také „přirozený kód“

absolutní hodnota čísla se znaménkovým bitem;  $0 \sim (+)$ ,  $1 \sim (-)$

$$P(X) = X \quad \text{pro } X \geq 0 \quad P(X) = |X| + \frac{1}{2}M \quad \text{pro } X \leq 0$$

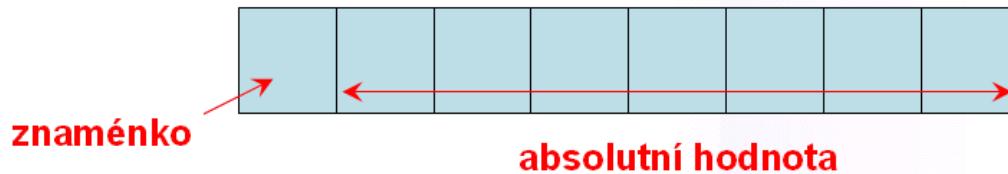
- složitá realizace aritmetických operací  
nejprve třeba otestovat znaménko  
pak se použije algoritmus operace  
(sčítání, odečítání)

- nevýhodou jsou dvě reprezentace nuly  
(nutno ošetřit)



# Přímý kód se znaménkem

---

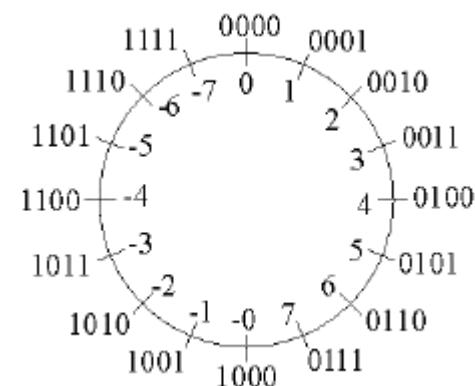


Např. pro 4-bitové slovo (n-bitové)

kladná čísla: 0 ... 7 ( $2^{n-1}-1$ )

záporná čísla: -7 ... 0

dvě nuly: 0000 a 1000



# Inverzní kód

---

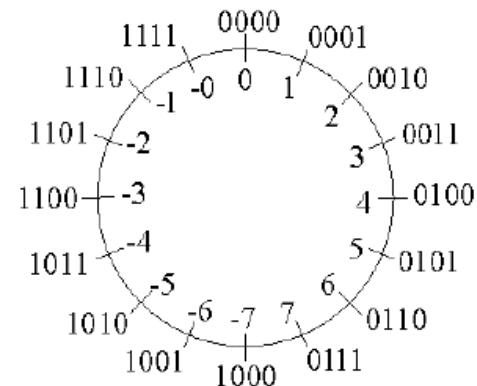
Vychází z jednotkového doplňku (one's complement)

$$I(X) = X \quad \text{pro } X \geq 0 \quad I(X) = M - \varepsilon + X \quad \text{pro } X \leq 0$$

- opět problém dvou nul (0000 a 1111)  $-\frac{1}{2}M \leq X < \frac{1}{2}M$
- obtížnější realizace aritmetických operací
- vznikne negací bitů daného slova
- MSB bit má opět charakter znaménka

Např.  $+0,8125 \sim 0,1101$

$-0,8125 \sim 1,0010$

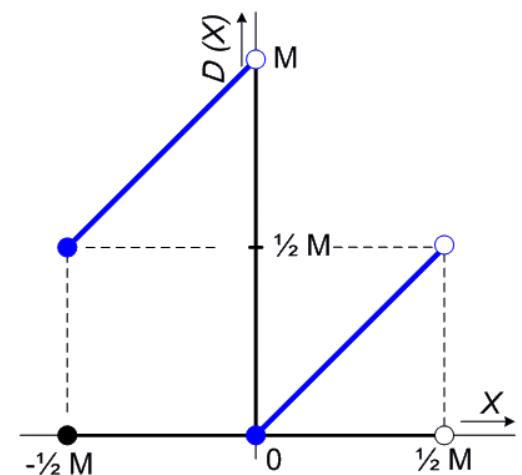


# Doplňkový kód

Vychází z dvojkového doplňku (two's complement)

$$D(X) = X \quad \text{pro } X \geq 0 \quad D(X) = M + X \quad \text{pro } X < 0$$

- nejvyšší bit má opět charakter znaménka  
(nenese informaci o hodnotě)  $-\frac{1}{2}M \leq X < \frac{1}{2}M$
- vznikne přičtením  $\varepsilon$  k inverznímu kódu
- max. zápis. číslo nemá kladný ekvivalent
- algoritmus odečítání je stejný jako sčítání  
(sečtou se obrazy a ignoruje se přenos)
- nejpoužívanější



# Doplňkový kód

---

Doplňkový kód je možné definovat také vztahem:

$$D(X) = S = -s_{n-1} \cdot 2^{n-1} + \sum_{i=-m}^{n-2} s_i \cdot 2^i \quad \begin{array}{l} \text{platí pro celý rozsah } X \\ -\frac{1}{2}M \leq X < \frac{1}{2}M \end{array}$$

Doplňkový kód chápeme jako **signed** (se znaménkem):

- *Celočíselný formát* (tj.  $m = 0, N = n$ ):

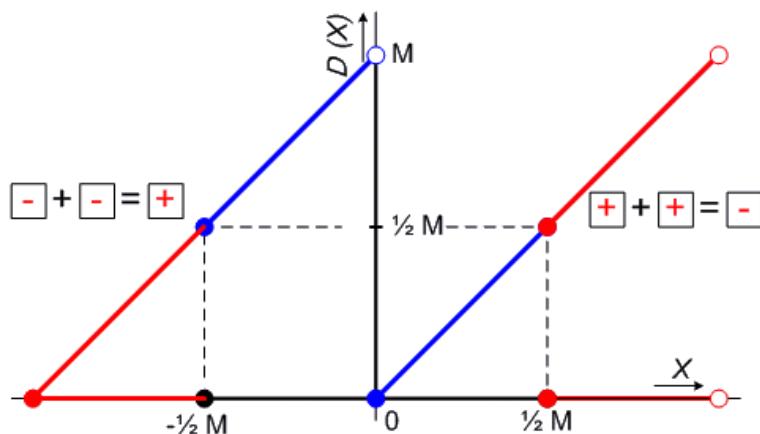
$$S = -s_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} s_i \cdot 2^i$$

- *Zlomkový (fraction) formát* (tj.  $n = 1, N = m+1$ )

$$S = -s_0 + \sum_{i=-m}^{-1} s_i \cdot 2^i$$

# Doplňkový kód - sčítání

Při operacích typu  $(+)+(+)$  a  $(-)+(-)$  může dojít k přetečení (výsledek je mimo rozsah platných bitů);  
v některých počítačích se přetečení detekuje a uchovává se,  
v jiných se nedetekuje – musí ohlídat programátor  
Při operacích typu  $(-)+(+)$  a  $(+)+(-)$  nemůže k přetečení dojít



# Příklady sčítání v doplňk. kódu

---

Chybné výsledky při malé délce n-bitového slova (přetečení):

$$\begin{array}{r} -9 \\ \underline{-8} \\ -17 \end{array} \quad \begin{array}{r} 10111 \\ \underline{11000} \\ 1|01111 \end{array} \quad \begin{array}{r} 9 \\ \underline{8} \\ 17 \end{array} \quad \begin{array}{r} 01001 \\ \underline{01000} \\ 10001 \end{array} \quad \begin{array}{r} 001001 \\ \underline{001000} \\ 010001 \end{array}$$

Také pro výsledek musí platit:  $-\frac{1}{2}M \leq X < \frac{1}{2}M$  ( $M = 2^n$ )

$$\begin{array}{r} 9 \\ \underline{-5} \\ 4 \end{array} \quad \begin{array}{r} 01001 \\ \underline{11011} \\ 1|00100 \end{array} \quad \begin{array}{r} -9 \\ \underline{4} \\ -5 \end{array} \quad \begin{array}{r} 10111 \\ \underline{00100} \\ 11011 \end{array}$$

Pokud je výsledek záporný, jeho absolutní hodnotu získáme opět pomocí dvojkového doplňku

# Příklad

---

Sečtěte  $(47)_{10}$  a  $(-20)_{10}$

# Příklad

---

Sečtěte  $(47)_{10}$  a  $(-20)_{10}$

$(20)_{10} = 0001\ 0100 \rightarrow$  invertovat  $\rightarrow 1110\ 1011 \rightarrow$  přičíst 1  $\rightarrow 1110\ 1100$

nebo opsat zprava až za první 1 a zbytek invertovat  $\rightarrow 1110\ 1100 = (-20)_{10}$

$$\begin{array}{r} (47)_{10} = 0010\ 1111 \\ +(-20)_{10} = \underline{\hspace{2cm} 1110\ 1100} \\ (1)0001\ 1011 = (27)_{10} \end{array}$$

# Příklad

---

Sečtěte  $(-100)_{10}$  a  $(-20)_{10}$

# Příklad

---

Sečtěte  $(-100)_{10}$  a  $(-20)_{10}$

$$(100)_{10} = 0110\ 0100 \rightarrow 1001\ 1100 = (-100)_{10}$$

$$(20)_{10} = 0001\ 0100 \rightarrow 1110\ 1100 = (-20)_{10}$$

$$\begin{array}{r} (-100)_{10} = & 1001\ 1100 \\ + (-20)_{10} = & \underline{1110\ 1100} \\ \hline (1)1000\ 1000 & = (-120)_{10} \end{array}$$

# Násobení v doplňkovém kódu

---

$$(+2).(-3) = (-6)$$

		0	0	1	0
		1	1	0	1
		0	0	1	0
		0	0	0	0
		0	0	1	0
		0	0	1	0
		0	0	1	0
		1	1	1	0
1	1	1	1	1	0
1	1	1	1	1	0

$$(-3).(+2) = (-6)$$

		1	1	0	1
		0	0	1	0
		0	0	0	0
		1	1	1	1
		0	0	0	0
		0	0	0	0
		1	1	1	1
		1	1	0	1
		1	1	1	0

Nakonec jsme přičetli  $(-2)$ ,  
výsledek je ve dvojk. doplňku

Vlevo šíříme znaménka mezi výsledků,  
výsledek je ve dvojkovém doplňku

# Příklady násobení – Boothův alg.

---

$$(+2) \cdot (-3) = (-6)$$

A 0010 0000 0 (= +2)

B 1110 0000 0 (= -2)

C 0000 1101 0 (= -3) „10“ -> C = C + B a arit. posun vpravo

C 1110 1101 0

C 1111 0110 1 „01“ -> C = C + A a arit. posun vpravo

C 0001 0110 1

C 0000 1011 0 „10“ -> C = C + B a arit. posun vpravo

C 1110 1011 0

C 1111 0101 1 „11“ či „00“ -> opíše a arit. posun vpravo

C 1111 0101 1

C 1111 1010 (= -6)

# Dělení

---

Obtížnější realizace, obecně zdlouhavé.

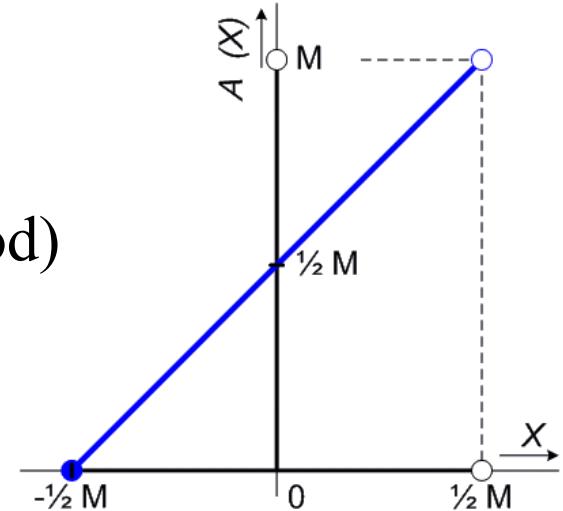
Nejčastější algoritmy:

- *s restaurací nezáporného zbytku* (s návratem přes nulu),  
odečítá se příslušně posunutý dělitel od dělence
- *bez restaurace nezáporného zbytku* (bez návratu přes  
nulu) – obdoba předešlého algoritmu, rychlejší
- *násobení převrácenou hodnotou*
- *podmíněné odečítání* – zjištujeme kolikrát se dělitel vejde  
do dělence, porovnávání a odečítání

# Aditivní (posunutý) kód

Kód s posunutou nulou – k číslu připočteme známou konstantu  $K$ :  $A(X) = X + K$  (binary offset)

- zachovává relace ( $</>$ )
- 2 varianty:
  - $\Rightarrow K = 2^{n-1}$ , tj.  $\frac{1}{2}M$  (sudý aditivní kód)
  - $K = 2^{n-1} - \varepsilon$  (lichý aditivní kód)



- před násobením se musí odečíst  $K$
- převod do doplňkového kódu provedeme negací nejvyššího bitu (pro sudý aditivní kód)
- použití: reprezentace exponentu reálných čísel, ADC...

# Aditivní (posunutý) kód

---

sudý:  $A(X) = X + 2^{5-1} = X + 16$

lichý:  $A(X) = X + 2^{5-1}-1$

<b>X<sub>10</sub></b>	<b>X<sub>2</sub></b>	<b>A<sub>2</sub></b>
3	00011	10011
2	00010	10010
1	00001	10001
0	00000	10000
-1		01111
-2		01110
-3		01101

<b>X<sub>10</sub></b>	<b>X<sub>2</sub></b>	<b>A<sub>2</sub></b>
3	00011	10010
2	00010	10001
1	00001	10000
0	00000	01111
-1		01110
-2		01101
-3		01100

# Příklad

---

	přímý	inverzní	doplňkový	aditivní sudý	aditivní lichý
9					
-9					
9/16					
-9/16					

# Příklad

---

	přímý	inverzní	doplňkový	aditivní sudý	aditivní lichý
9	0 1001	0 1001	0 1001	1 1001	1 1000
-9					
9/16					
-9/16					

# Příklad

---

	přímý	inverzní	doplňkový	aditivní sudý	aditivní lichý
9	0 1001	0 1001	0 1001	1 1001	1 1000
-9	1 1001	1 0110	1 0111	0 0111	0 0110
9/16					
-9/16					

# Příklad

---

	přímý	inverzní	doplňkový	aditivní sudý	aditivní lichý
9	0 1001	0 1001	0 1001	1 1001	1 1000
-9	1 1001	1 0110	1 0111	0 0111	0 0110
9/16	0, 1001	0, 1001	0, 1001	1, 1001	1, 1000
-9/16					

# Příklad

---

	přímý	inverzní	doplňkový	aditivní sudý	aditivní lichý
9	0 1001	0 1001	0 1001	1 1001	1 1000
-9	1 1001	1 0110	1 0111	0 0111	0 0110
9/16	0, 1001	0, 1001	0, 1001	1, 1001	1, 1000
-9/16	1, 1001	1, 0110	1, 0111	0, 0111	0, 0110

# Pohyblivá řádová čárka

---

Floating point (FP) – oproti FX zvětšení rozsahu (nezvyšuje se počet rozlišitelných hodnot)

*Mantisa (M)* – informace o hodnotě čísla (určuje přesnost)  $X_{FP} = M \cdot z^E$

*Exponent (E)* – informace o pozici řádové čárky (určuje rozsah)

$z \dots$  základ použité číselné soustavy (nejčastěji 2)

Dvě řádové podmřížky: většinou každá v různém formátu - mantisa ve zlomkovém tvaru v přímém kódu, exponent celočíselný většinou v kódu s posunutou nulou

# IEEE 754 - formát reálného čísla

---

- nejvyšší bit mantisy je vždy 1 a nezobrazuje se (hidden one);  
toto neplatí, je-li obraz exponentu nulový (zobrazení 0)
- myšlená řádová čárka je za nejvyšším bitem mantisy
- posunutí exponentu je  $K = 2^7-1=127; 2^{10}-1=1023\dots$
- kladné začíná 0, záporné 1 (znaménkový bit  $S$ )

Jednoduchá přesnost (binary32 – single prec.): 32 bitů (4 byte)

0|00000000|00000000000000000000000000000000

1b znaménko mantisy, 8b exponent, 23b mantisa

Dvojnásobná přesnost (binary64 – double prec.): 64 bitů

1b znaménko mantisy, 11b exponent, 52b mantisa

# IEEE 754

---

Hodnota exponentu v aditivním kódu:  $exp = E + K$

Zpětná transformace:  $X_{FP} = (-1)^S \cdot 2^{exp-K} \cdot (1, M_{IEEE})$

Formát IEEE 754 má speciální hodnoty (výjimky):

$exp = \text{max. } (255)$  a  $M_{IEEE} = 0 \Rightarrow \pm\infty$

$exp = \text{max. } (255)$  a  $M_{IEEE} \neq 0 \Rightarrow \text{neplatné č. „NaN“}$

$exp = 0$  a  $M_{IEEE} = 0 \Rightarrow \pm 0$  (daná  $S$ )

$exp = 0$  a  $M_{IEEE} \neq 0,$   $\Rightarrow E = E + 1$  a zároveň

(denormalizovaná čísla, subnormal numbers) nula před  $M_{IEEE}$

# Příklad – IEEE 754

---

Zobrazte ve formátu IEEE na 4 bytech reálné číslo  $(-258,125)_{10}$

$$(258,125)_{10} = (100000010,001)_2 = 1,00000010001 \cdot 2^8$$

exponent:  $2^7 - 1 + 8 = (10000111)$

$$\begin{aligned} (-258,125)_{10} &= (\textcolor{red}{1} \textcolor{green}{100} \textcolor{green}{0011} \textcolor{blue}{1000} \textcolor{blue}{0001} \textcolor{blue}{0001} \textcolor{blue}{0000} \textcolor{blue}{0000} \textcolor{blue}{0000})_{\text{IEEE}} = \\ &= (\text{ C } \quad 3 \quad \quad 8 \quad \quad 1 \quad \quad 1 \quad \quad 0 \quad \quad 0 \quad \quad 0 \quad )_{16} \end{aligned}$$

Pozn.:

konvence uspořádání vícebytových slov (pořadí v paměti):

*big endian* – nejvýznamnější byte první (C3 81 10 00)

*little endian* – nejnižší byte první (00 10 81 C3)

# Příklad

---

Převeďte číslo (BCDE 0000)<sub>16</sub> do tvaru  $\pm M \cdot 2^E$  (dle IEEE 754)

# Příklad

---

Převeďte číslo (BCDE 0000)<sub>16</sub> do tvaru  $\pm M \cdot 2^E$  (dle IEEE 754)

$$X_{FP} = (-1)^S \cdot 2^{exp-k} \cdot (1, M_{IEEE})$$

B	C	D	E	0	0	0	0
1011	1100	1101	1110	0000	0000	0000	0000

$$X_{FP} = (-1)^1 \cdot 2^{121-127} \cdot 1,101111 = -1,101111 \cdot 2^{-6}$$

# Příklad

---

Převeďte číslo  $(4291\ 4000)_{16}$  do dekadické soustavy (dle IEEE 754)

# Příklad

---

Převeďte číslo  $(4291\ 4000)_{16}$  do dekadické soustavy (dle IEEE 754)

4	2	9	1	4	0	0	0
0100	0010	1001	0001	0100	0000	0000	0000

$$X_{FP} = (-1)^0 \cdot 2^{133-127} \cdot 1,001000101 = 1001000,101 \cdot 2^6$$
$$X_{FP} = (72,625)_D$$

# Příklad

---

Převeďte číslo  $(47,0625)_{10}$  do binární soustavy (dle IEEE 754)

# Příklad

---

Převeďte číslo  $(47,0625)_{10}$  do binární soustavy (dle IEEE 754)

$$(47,0625)_{10} = (101111,0001)_2 = 1,011110001 \cdot 2^5$$

exponent:  $5+127 = (10000100)$

$$\begin{aligned}(47,0625)_{10} &= (\textcolor{red}{0}\textcolor{green}{100}\textcolor{green}{0010}\textcolor{blue}{0011}\textcolor{blue}{1100}\textcolor{blue}{0100}\textcolor{blue}{0000}\textcolor{blue}{0000}\textcolor{blue}{0000})_2 = \\ &= (\begin{array}{cccccccc} 4 & 2 & 3 & C & 4 & 0 & 0 & 0 \end{array})_{16}\end{aligned}$$

# Dekadická korekce

---

- používá se při výpočtech ve dvojkově-desítkové soustavě (číslice jsou nejčastěji v BCD kódu – kód 8421)
- pokud při sčítání je nižší nibble větší než 9, přičteme 6; obdobně při odečítání od mezivýsledku odečteme 6, je-li nibble větší než 9

$$\text{Př.: } 26 + 35 = 61$$

0010 0110

0011 0101

---

0101 1011 (= 5Bh)

---

0000 0110 (+ 6)

---

0110 0001 (= 61)

$$71 - 56 = 15$$

0111 0001

- 0101 0110

---

0001 1011 (= 1Bh)

---

- 0000 0110 (- 6)

---

0001 0101 (= 15)

# Převod z binární soustavy do BCD

---

Celé nezáporné binární číslo  $N$  lze vyjádřit:

$$N = a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \dots + a_1 \cdot 2^1 + a_0$$

$$N = ((a_n \cdot 2 + a_{n-1}) \cdot 2 + \dots + a_1) \cdot 2 + a_0$$

Číslo  $N$  lze vytvořit postupným násobením dvěma (posun vlevo) a přičítáním následujícího bitu (0 nebo 1). Díváme se na číslo jako dekadické, používáme dekadickou korekci.

Někdy se používá metoda postupného odečítání mocnin základu (doba převodu závisí se převáděném čísle) nebo metoda postupného dělení základem - hodnotou 10 (relativně zdlouhavé).

# Převod z binární soustavy na BCD

---

Převeďte binární číslo 0110100 ( $a_6 \dots a_0$ ) na číslo dekadické v BCD kódu

0000 0000 počáteční stav

0000 0000 přičtení  $a_6$

0000 0000 ~~dekadická korekce~~

0000 0001  $a_6 \cdot 2 + a_5$

0000 0001 ~~dekadická korekce~~

0000 0011  $(a_6 \cdot 2 + a_5) \cdot 2 + a_4$

0000 0011 ~~dekadická korekce~~

0000 0110  $((a_6 \cdot 2 + a_5) \cdot 2 + a_4) \cdot 2 + a_3$

0000 0110 ~~dekadická korekce~~

0000 1101  $(((a_6 \cdot 2 + a_5) \cdot 2 + a_4) \cdot 2 + a_3) \cdot 2 + a_2$

0001 0011 dekadická korekce

0010 0110  $(((a_6 \cdot 2 + a_5) \cdot 2 + a_4) \cdot 2 + a_3) \cdot 2 + a_2 \cdot 2 + a_1$

0010 0110 ~~dekadická korekce~~

0100 1100  $((((a_6 \cdot 2 + a_5) \cdot 2 + a_4) \cdot 2 + a_3) \cdot 2 + a_2) \cdot 2 + a_1 \cdot 2 + a_0$

0101 0010 dekadická korekce ... výsledek **52**

# Příklad

---

Sečtěte  $(154)_D$  a  $(271)_D$  v BCD kódu.

# Příklad

---

Sečtěte  $(154)_D$  a  $(271)_D$  v BCD kódu.

$$\begin{array}{r} (154)_D = & 0001 & 0101 & 0100 \\ + (271)_D = & \underline{0010} & \underline{0111} & \underline{0001} \\ \hline & 0011 & \textcolor{red}{1100} & 0101 \\ & + & 0110 & \\ & 0100 & 0010 & 0101 & = & (425)_D \end{array}$$

# Příklad

---

Sečtěte  $(571)_D$  a  $(-356)_D$  v BCD kódu.

# Příklad

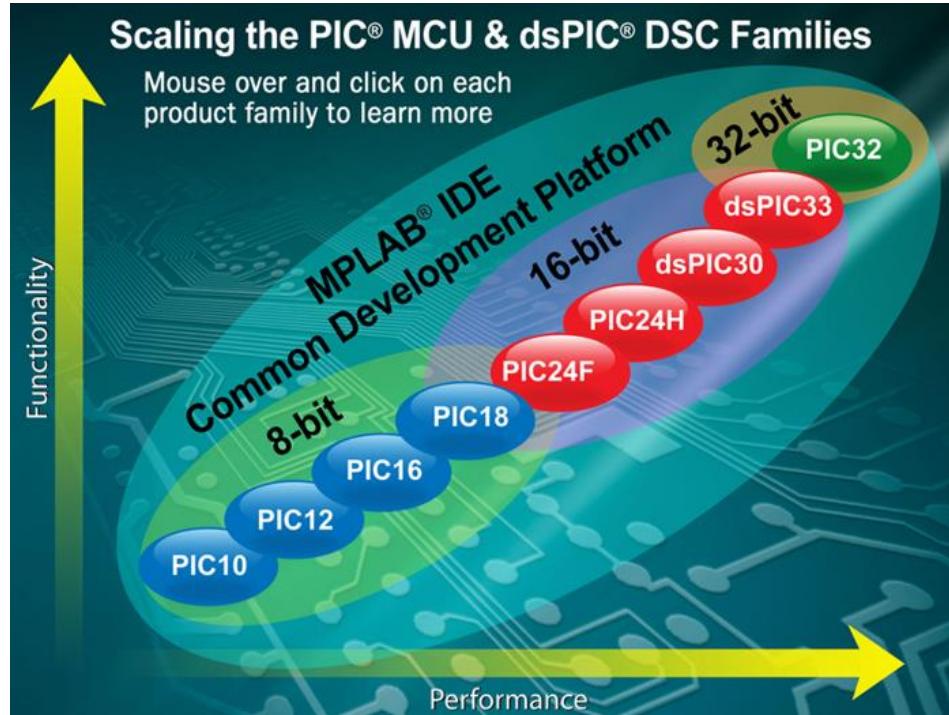
---

Sečtěte  $(571)_D$  a  $(-356)_D$  v BCD kódu.

$$\begin{array}{r} (571)_D = & 0101 & 0111 & 0001 \\ \underline{- (356)_D =} & 0011 & 0101 & 0110 \\ + & \underline{\textcolor{blue}{1101}} & \underline{\textcolor{blue}{1011}} & \underline{\textcolor{blue}{1010}} \\ & 0010 & 0010 & \textcolor{red}{1011} \\ & - & & 0110 & \text{(dekadická korekce)} \\ & + & \underline{\textcolor{blue}{1010}} \\ \hline & 0010 & 0010 & 0101 \\ & - & 0001 & & \text{(přenos)} \\ & + & \underline{\textcolor{blue}{1111}} \\ \hline & 0010 & 0001 & 0101 & = (215)_D \end{array}$$

Přenos jen při dekadické korekci. Jednodušší je však rovnou odčítat...

# Mikrořadiče PIC



# Mikrořadiče Microchip PIC16

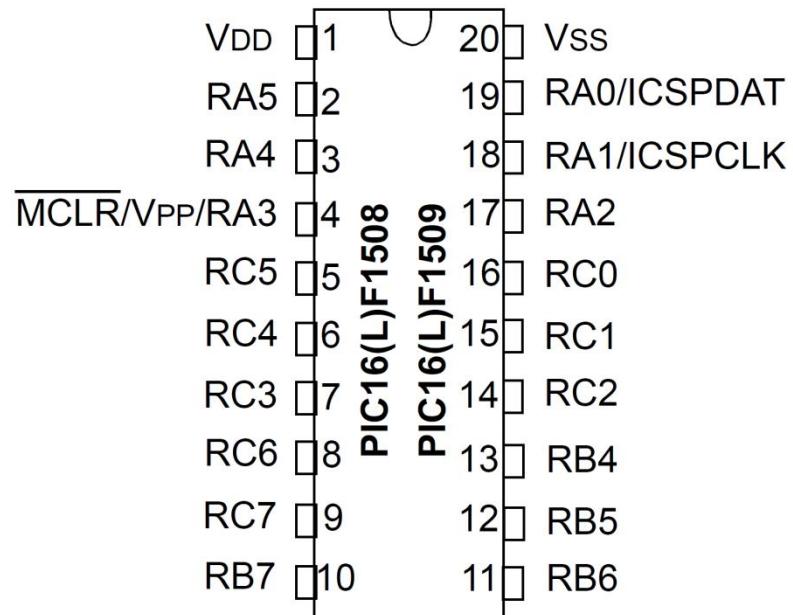
---

- RISC mikrořadiče v technologii CMOS
- harvardská arch.: programová sběrnice 14 bitů, datová 8 bitů
- všechny instrukce jsou jedno-cyklové (kromě skokových - podle výsledku operace jsou jedno- až dvou-cyklové)
- jeden strojový takt trvá 4 hodinové pulsy
- dvoustupňový pipelining (fetch, execute)
- velmi nízká proudová spotřeba
- rychlosť je až 12 MIPS (48 MHz)

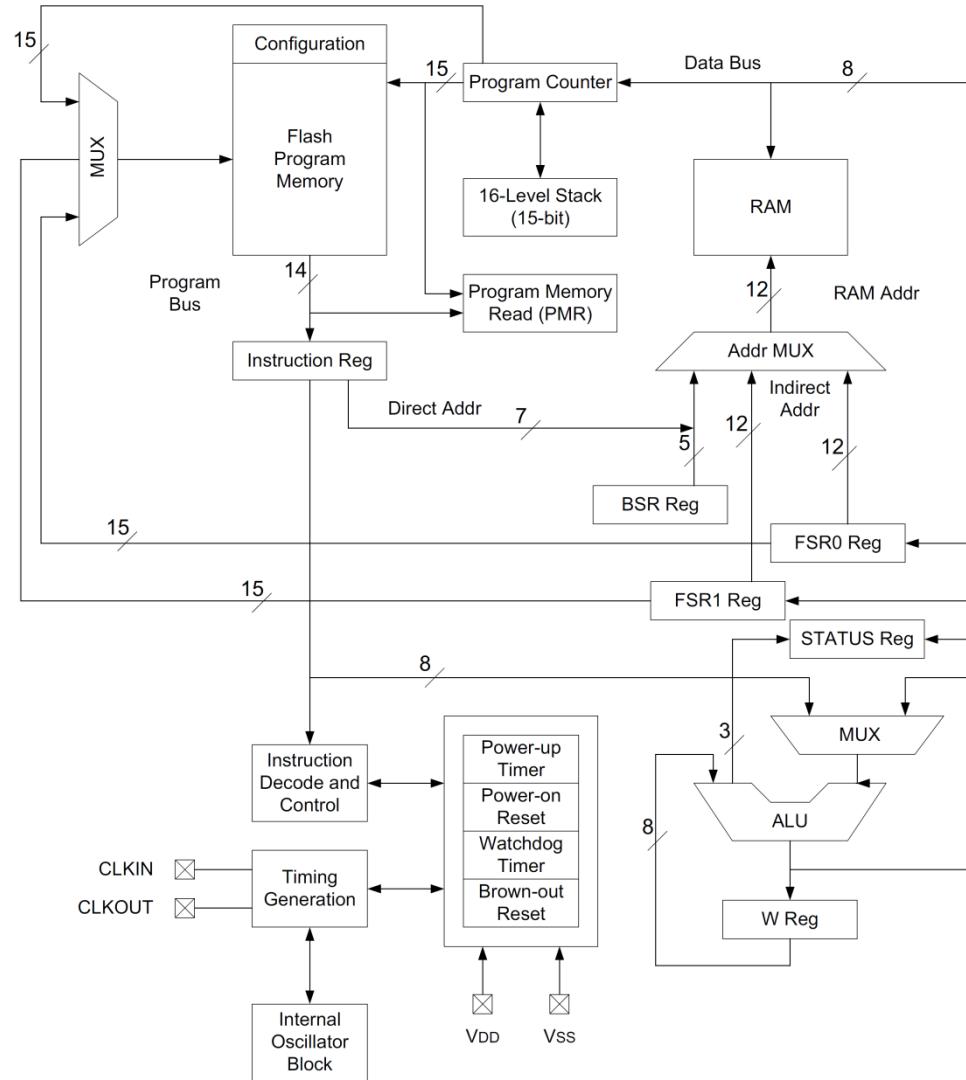
# Mikrořadič PIC16F1508

---

- pouzdro s 20 vývody
- 4096 slov programové paměti - FLASH
- 16-úrovňový zásobník
- 256 B Data RAM
- 128 B Data HEF (EEPROM)
- max. hod. kmitočet 20 MHz
- 200 ns min. instr. cyklus
- napájení 2,3 – 5,5 V
- ~1,1 mA @ 5 V @ 16 MHz
- programování ICSP po 5 pinech
- odběr z výstupů až  $\pm 25$  mA



# Jádro a paměti PIC16F1508



# Watchdog Timer (WDT)

---

*Hlídací časovač* – obvod zajišťující kontrolu správného běhu programu (musí být vynulován/nastaven dříve než dojde k jeho přetečení)

- pokud není periodicky resetován, při přetečení provede reset mikrořadiče
- používá hlavní hodinový signál nebo nezávislý interní RC oscilátor
- doba plného načítání obvykle volitelná
  - zde časy od 1 ms do 256 s (kroky po mocninách 2)
- slouží pro vzpamatování aplikace po záběhnutí programu

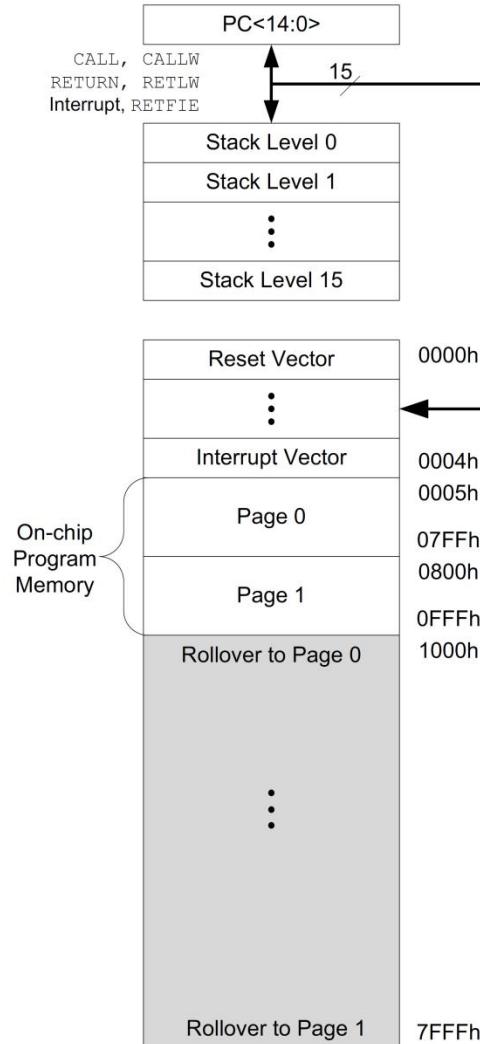
# Programová paměť'

Program Counter – viz dále

Stack – viz dále

Programová paměť' – FLASH

- velikost  $32\text{k} \times 14$  bitů  
(0000h až 7FFFh)
- min. 10 000 přepisů  
(min. 100 000 pro HEF)



# Stack (zásobník)

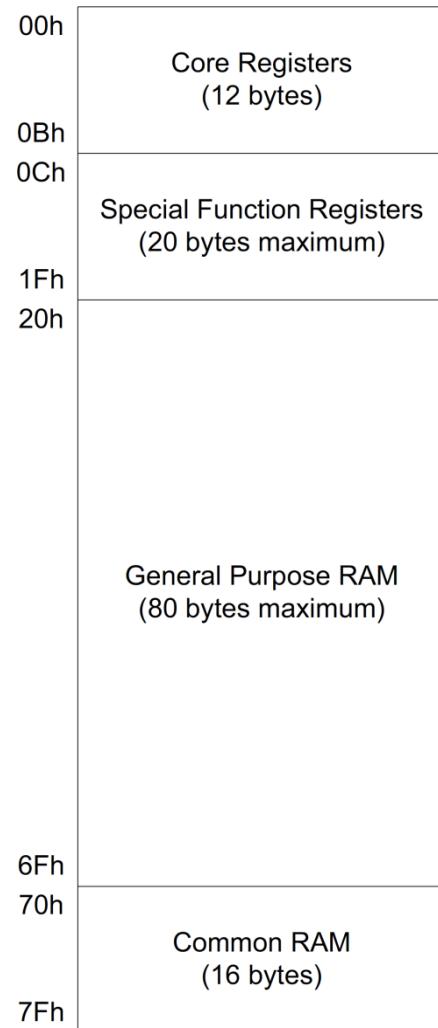
---

- 16 úrovní  $\times$  15 bitů (je to adresní zásobník)
- ukládá návratovou adresu při skoku do podprogramu (*CALL*, *CALLW* nebo přerušení)
- obnovuje návratovou adresu při návratu z podprogramu (*RETURN*, *RETLW* nebo *RETFIE*)
- není součástí adresového prostoru pro program nebo data
- SP (Stack Pointer, ukazatel zásobníku) nelze SW ovládat (neexistují zde instrukce *PUSH* a *POP*)
- přetečení/podtečení ošetřeno – *RESET* mikrokontroleru

# Paměť RAM

---

- rozdělena na 32 bank
- každá banka rozdělena na:
  - 12 core registers (kopie napříč bankami)
  - až 20 SFR (dle periferií)
  - až 80 B volné RAM (vždy různých)
  - 16 B volné RAM (kopie napříč bankami)



# Paměť RAM – banky 0-7

BANK 0		BANK 1		BANK 2		BANK 3		BANK 4		BANK 5		BANK 6		BANK 7		
000h	Core Registers (Table 3-2)	080h	Core Registers (Table 3-2)	100h	Core Registers (Table 3-2)	180h	Core Registers (Table 3-2)	200h	Core Registers (Table 3-2)	280h	Core Registers (Table 3-2)	300h	Core Registers (Table 3-2)	380h	Core Registers (Table 3-2)	
00Bh	PORTA	08Bh	TRISA	10Bh	LATA	18Bh	ANSELA	20Bh	WPUA	28Bh	—	30Bh	—	38Bh	—	
00Ch	PORTB	08Dh	TRISB	10Dh	LATB	18Dh	ANSELB	20Dh	WPUB	28Ch	—	30Ch	—	38Ch	—	
00Dh	PORTC	08Eh	TRISC	10Eh	LATC	18Eh	ANSELc	20Eh	—	28Dh	—	30Dh	—	38Dh	—	
00Fh	—	08Fh	—	10Fh	—	18Fh	—	20Fh	—	28Eh	—	30Eh	—	38Eh	—	
010h	—	090h	—	110h	—	190h	—	210h	—	290h	—	30Fh	—	38Fh	—	
011h	PIR1	091h	PIE1	111h	CM1CON0	191h	PMADRL	211h	SSP1BUF	291h	—	310h	—	390h	—	
012h	PIR2	092h	PIE2	112h	CM1CON1	192h	PMADRH	212h	SSP1ADD	292h	—	311h	—	391h	IOCAP	
013h	PIR3	093h	PIE3	113h	CM2CON0	193h	PMADTL	213h	SSP1MSK	293h	—	312h	—	392h	IOCAN	
014h	—	094h	—	114h	CM2CON1	194h	PMADTH	214h	SSP1STAT	294h	—	313h	—	393h	IOCAF	
015h	TMR0	095h	OPTION_REG	115h	CMOUT	195h	PMCON1	215h	SSP1CON1	295h	—	314h	—	394h	IOCBP	
016h	TMR1L	096h	PCON	116h	BORCON	196h	PMCON2	216h	SSP1CON2	296h	—	315h	—	395h	IOCBN	
017h	TMR1H	097h	WDTCON	117h	FVRCON	197h	VREGCON	217h	SSP1CON3	297h	—	316h	—	396h	IOCBF	
018h	T1CON	098h	—	118h	DAC1CON0	198h	—	218h	—	298h	—	317h	—	397h	—	
019h	T1GCON	099h	OSCCON	119h	DAC1CON1	199h	RCREG	219h	—	299h	—	318h	—	398h	—	
01Ah	TMR2	09Ah	OSCSTAT	11Ah	—	19Ah	TXREG	21Ah	—	29Ah	—	319h	—	399h	—	
01Bh	PR2	09Bh	ADRESL	11Bh	—	19Bh	SPBRG	21Bh	—	29Bh	—	31Ah	—	39Ah	—	
01Ch	T2CON	09Ch	ADRESH	11Ch	—	19Ch	SPBRGH	21Ch	—	29Ch	—	31Bh	—	39Bh	—	
01Dh	—	09Dh	ADCON0	11Dh	APFCON	19Dh	RCSTA	21Dh	—	29Dh	—	31Ch	—	39Ch	—	
01Eh	—	09Eh	ADCON1	11Eh	—	19Eh	TXSTA	21Eh	—	29Eh	—	31Dh	—	39Dh	—	
01Fh	—	09Fh	ADCON2	11Fh	—	19Fh	BAUDCON	21Fh	—	29Fh	—	31Eh	—	39Eh	—	
020h	General Purpose Register 80 Bytes	0A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h	Unimplemented Read as '0'	220h	Unimplemented Read as '0'	2A0h	Unimplemented Read as '0'	320h	Unimplemented Read as '0'	3A0h	Unimplemented Read as '0'	
06Fh	Common RAM (Accesses 70h – 7Fh)		0EFh	0F0h	Common RAM (Accesses 70h – 7Fh)	16Fh	170h	Common RAM (Accesses 70h – 7Fh)	1EFh	1F0h	Common RAM (Accesses 70h – 7Fh)	26Fh	270h	Common RAM (Accesses 70h – 7Fh)	2EFh	2F0h
070h	Common RAM (Accesses 70h – 7Fh)		0FFh	0FFh	Common RAM (Accesses 70h – 7Fh)	17Fh	1FFh	Common RAM (Accesses 70h – 7Fh)	27Fh	2FFh	Common RAM (Accesses 70h – 7Fh)	36Fh	370h	Common RAM (Accesses 70h – 7Fh)	37Fh	3FFh

Legend:

= Unimplemented data memory locations, read as '0'.

# Paměť RAM – banky 8-29

BANK 8		BANK 9		BANK 10		BANK 11		BANK 12		BANK 13		BANK 14		BANK 15	
400h	Core Registers (Table 3-2)	480h	Core Registers (Table 3-2)	500h	Core Registers (Table 3-2)	580h	Core Registers (Table 3-2)	600h	Core Registers (Table 3-2)	680h	Core Registers (Table 3-2)	700h	Core Registers (Table 3-2)	780h	Core Registers (Table 3-2)
40Bh	—	48Bh	—	50Bh	—	58Bh	—	60Bh	—	68Bh	—	70Bh	—	78Bh	—
40Ch	—	48Ch	—	50Ch	—	58Ch	—	60Ch	—	68Ch	—	70Ch	—	78Ch	—
40Dh	—	48Dh	—	50Dh	—	58Dh	—	60Dh	—	68Dh	—	70Dh	—	78Dh	—
40Eh	—	48Eh	—	50Eh	—	58Eh	—	60Eh	—	68Eh	—	70Eh	—	78Eh	—
40Fh	—	48Fh	—	50Fh	—	58Fh	—	60Fh	—	68Fh	—	70Fh	—	78Fh	—
410h	—	490h	—	510h	—	590h	—	610h	—	690h	—	710h	—	790h	—
411h	—	491h	—	511h	—	591h	—	611h	PWM1DCL	691h	CWG1DBR	711h	—	791h	—
412h	—	492h	—	512h	—	592h	—	612h	PWM1DCH	692h	CWG1DBF	712h	—	792h	—
413h	—	493h	—	513h	—	593h	—	613h	PWM1CON	693h	CWG1CON0	713h	—	793h	—
414h	—	494h	—	514h	—	594h	—	614h	PWM2DCL	694h	CWG1CON1	714h	—	794h	—
415h	—	495h	—	515h	—	595h	—	615h	PWM2DCH	695h	CWG1CON2	715h	—	795h	—
416h	—	496h	—	516h	—	596h	—	616h	PWM2CON	696h	—	716h	—	796h	—
417h	—	497h	—	517h	—	597h	—	617h	PWM3DCL	697h	—	717h	—	797h	—
418h	—	498h	NCO1ACCL	518h	—	598h	—	618h	PWM3DCH	698h	—	718h	—	798h	—
419h	—	499h	NCO1ACCH	519h	—	599h	—	619h	PWM3CON	699h	—	719h	—	799h	—
41Ah	—	49Ah	NCO1ACCU	51Ah	—	59Ah	—	61Ah	PWM4DCL	69Ah	—	71Ah	—	79Ah	—
41Bh	—	49Bh	NCO1INCL	51Bh	—	59Bh	—	61Bh	PWM4DCH	69Bh	—	71Bh	—	79Bh	—
41Ch	—	49Ch	NCO1INCH	51Ch	—	59Ch	—	61Ch	PWM4CON	69Ch	—	71Ch	—	79Ch	—
41Dh	—	49Dh	—	51Dh	—	59Dh	—	61Dh	—	69Dh	—	71Dh	—	79Dh	—
41Eh	—	49Eh	NCO1CON	51Eh	—	59Eh	—	61Eh	—	69Eh	—	71Eh	—	79Eh	—
41Fh	—	49Fh	NCO1CLK	51Fh	—	59Fh	—	61Fh	—	69Fh	—	71Fh	—	79Fh	—
420h	—	4A0h	—	520h	—	5A0h	—	620h	—	6A0h	—	720h	—	7A0h	—
46Fh	Unimplemented Read as '0'	4EFh	Unimplemented Read as '0'	56Fh	Unimplemented Read as '0'	5EFh	Unimplemented Read as '0'	66Fh	Unimplemented Read as '0'	6EFh	Unimplemented Read as '0'	76Fh	Unimplemented Read as '0'	7EFh	Unimplemented Read as '0'
470h	Accesses 70h – 7Fh	4F0h	Accesses 70h – 7Fh	570h	Accesses 70h – 7Fh	5F0h	Accesses 70h – 7Fh	670h	Accesses 70h – 7Fh	6F0h	Accesses 70h – 7Fh	770h	Accesses 70h – 7Fh	7F0h	Accesses 70h – 7Fh
47Fh	—	4FFh	Accesses 70h – 7Fh	57Fh	Accesses 70h – 7Fh	5FFh	Accesses 70h – 7Fh	67Fh	Accesses 70h – 7Fh	6FFh	Accesses 70h – 7Fh	77Fh	Accesses 70h – 7Fh	7FFh	Accesses 70h – 7Fh

# Paměť RAM – banky 30 a 31

Bank 30		Bank 31	
F0Ch	—	F8Ch	
F0Dh	—	FE3h	Unimplemented Read as '0'
F0Eh	—	FE4h	STATUS_SHAD
F0Fh	CLCADATA	FE5h	WREG_SHAD
F10h	CLC1CON	FE6h	BSR_SHAD
F11h	CLC1POL	FE7h	PCLATH_SHAD
F12h	CLC1SEL0	FE8h	FSR0L_SHAD
F13h	CLC1SEL1	FE9h	FSR0H_SHAD
F14h	CLC1GLS0	FEAh	FSR1L_SHAD
F15h	CLC1GLS1	FEBh	FSR1H_SHAD
F16h	CLC1GLS2	FECh	—
F17h	CLC1GLS3	FEDh	STKPTR
F18h	CLC2CON	FEEh	TOSL
F19h	CLC2POL	FEFh	TOSH
F1Ah	CLC2SEL0		
F1Bh	CLC2SEL1		
F1Ch	CLC2GLS0		
F1Dh	CLC2GLS1		
F1Eh	CLC2GLS2		
F1Fh	CLC2GLS3		
F20h	CLC3CON		
F21h	CLC3POL		
F22h	CLC3SEL0		
F23h	CLC3SEL1		
F24h	CLC3GLS0		
F25h	CLC3GLS1		
F26h	CLC3GLS2		
F27h	CLC3GLS3		
F28h	CLC4CON		
F29h	CLC4POL		
F2Ah	CLC4SEL0		
F2Bh	CLC4SEL1		
F2Ch	CLC4GLS0		
F2Dh	CLC4GLS1		
F2Eh	CLC4GLS2		
F2Fh	CLC4GLS3		
F30h	Unimplemented Read as '0'		
F6Fh			

# Paměť RAM – Core Registers

---

INDF – data z nepřímé adresace

- operace s INDF je operací na registru, jehož adresa je ve FSR

PCL, STATUS – viz dále

BSR – Bank Select Register

WREG (nyní už má svou adr.)

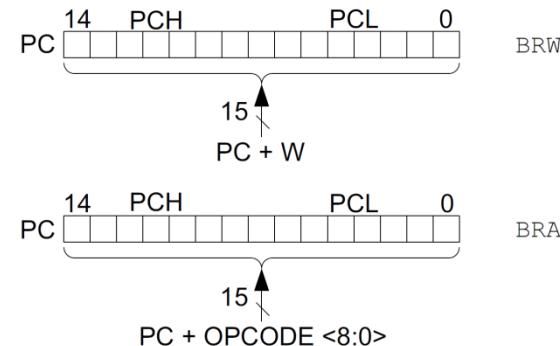
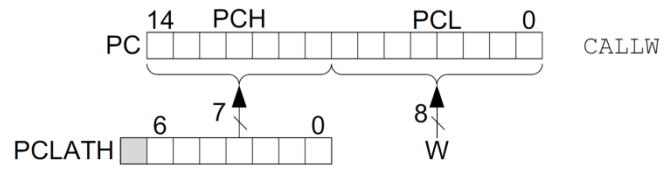
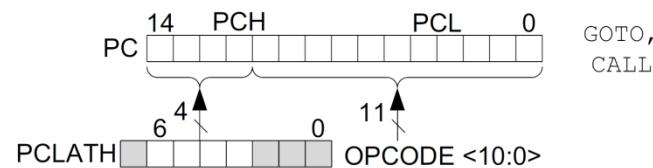
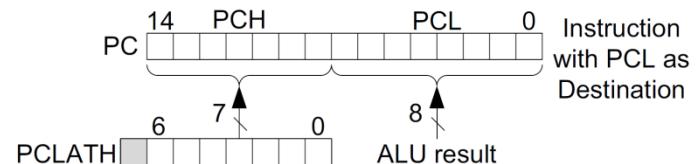
PCLATH – viz dále

INTCON – nastavení přerušení

Addresses	BANKx
x00h or x80h	INDF0
x01h or x81h	INDF1
x02h or x82h	PCL
x03h or x83h	STATUS
x04h or x84h	FSR0L
x05h or x85h	FSR0H
x06h or x86h	FSR1L
x07h or x87h	FSR1H
x08h or x88h	BSR
x09h or x89h	WREG
x0Ah or x8Ah	PCLATH
x0Bh or x8Bh	INTCON

# Program Counter (čítač instrukcí)

- PC vždy ukazuje na následující instrukci
- 15-bitový (~ 0 - 7FFFh)
- nižších 8 bitů je pro čtení i zápis přístupno v registru PCL
- horních 7 bitů není přímo přístupno – pro jejich změnu je nutno použít registr PCLATH, jehož obsah se přenáší do vyšších bitů PC při operaci s PCL



# STATUS registr (stav, příznaky)

---

C (Carry/!Borrow) – přenos při sčítání, odečítání a posuvu

DC (Digit carry) – přenos mezi 3. a 4. bitem

Z (Zero) – nulovost aritmetické, logické či přesun. operace

!PD (Power Down) – 0 po instrukci *SLEEP*

!TO (Time-Out) – 0 po přetečení hlídacího časovače (WDT)

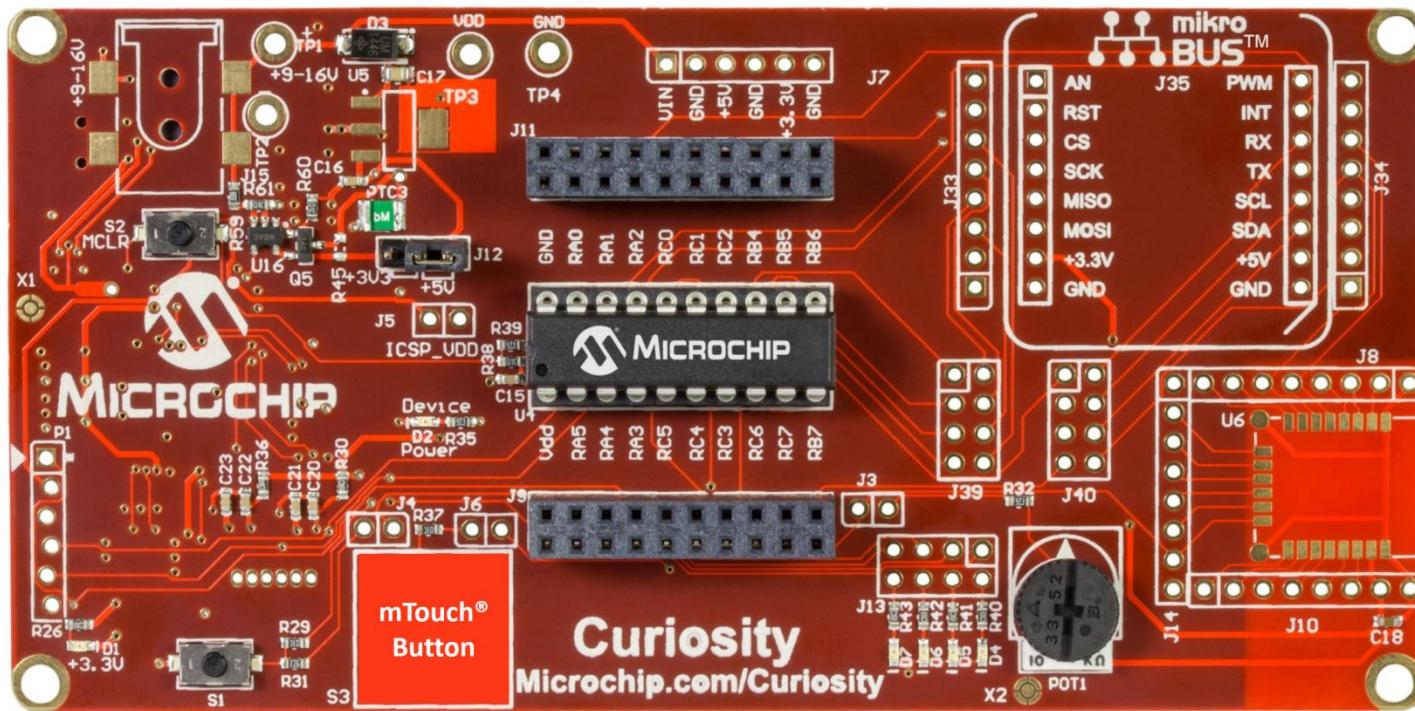
U-0	U-0	U-0	R-1/q	R-1/q	R/W-0/u	R/W-0/u	R/W-0/u
—	—	—	TO	PD	Z	DC <sup>(1)</sup>	C <sup>(1)</sup>
bit 7						bit 0	

# Microchip Curiosity

---

Vývojová deska pro vybrané 8b mikrokontrolery PIC

- komunikace s PC přes USB
- integrovaný PICkit3 pro programování a ladění
- pro kompatibilní µC v pouzdrech DIP od 8 do 20 pinů



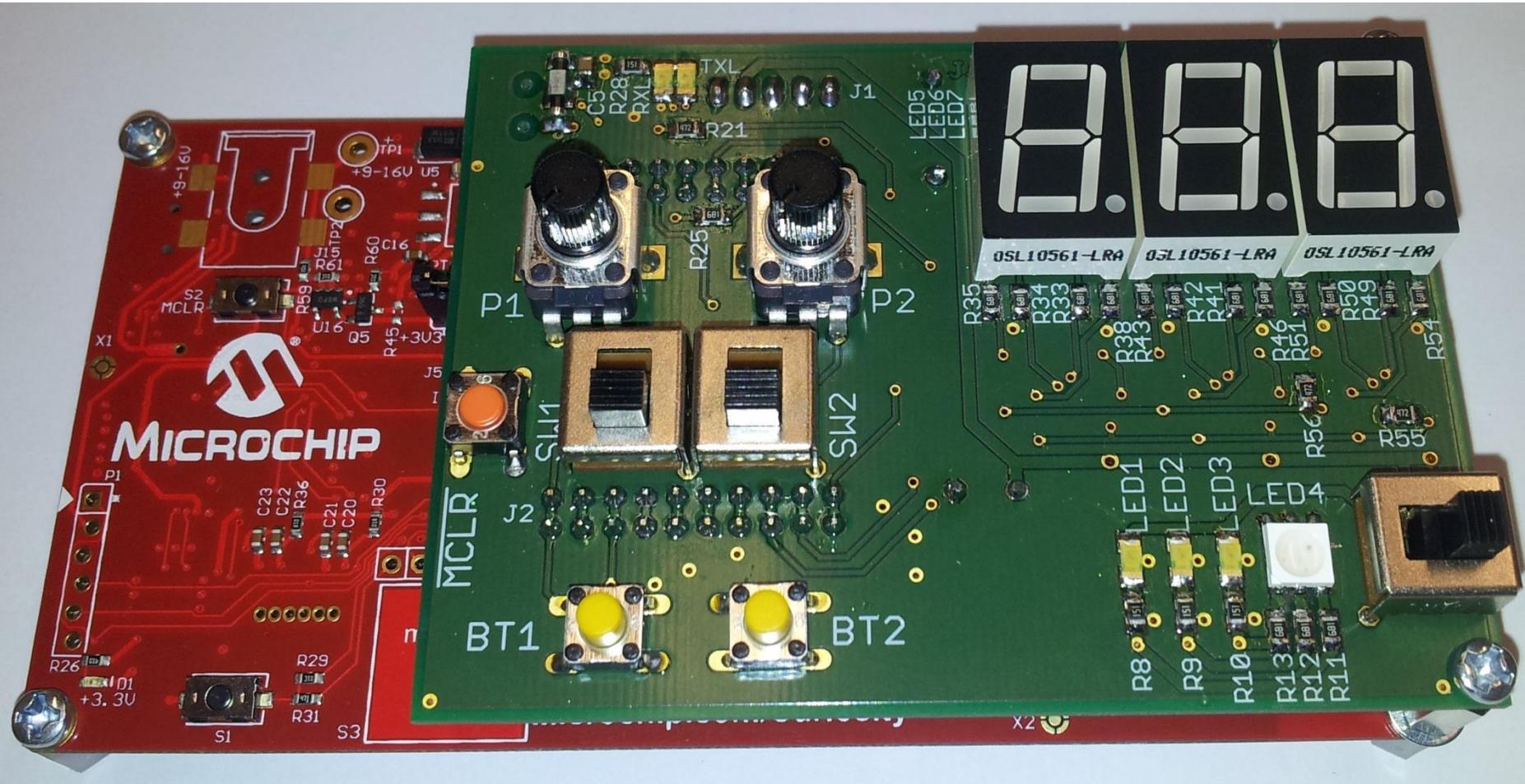
# Rozšiřující deska s periferiemi

---

Výuková deska připojená na Microchip Curiosity

- 2 tlačítka a 2 přepínače
- 3 7-segmentové (+ tečka) LED displeje po SPI
- převodník UART na USB – virtuální COM port
- 3 LED nebo 1 RGB LED (s možností PWM)
- 2 potenciometry pro ADC
- piezo-měnič

# Rozšiřující deska s periferiemi



# MPLABX IDE

---

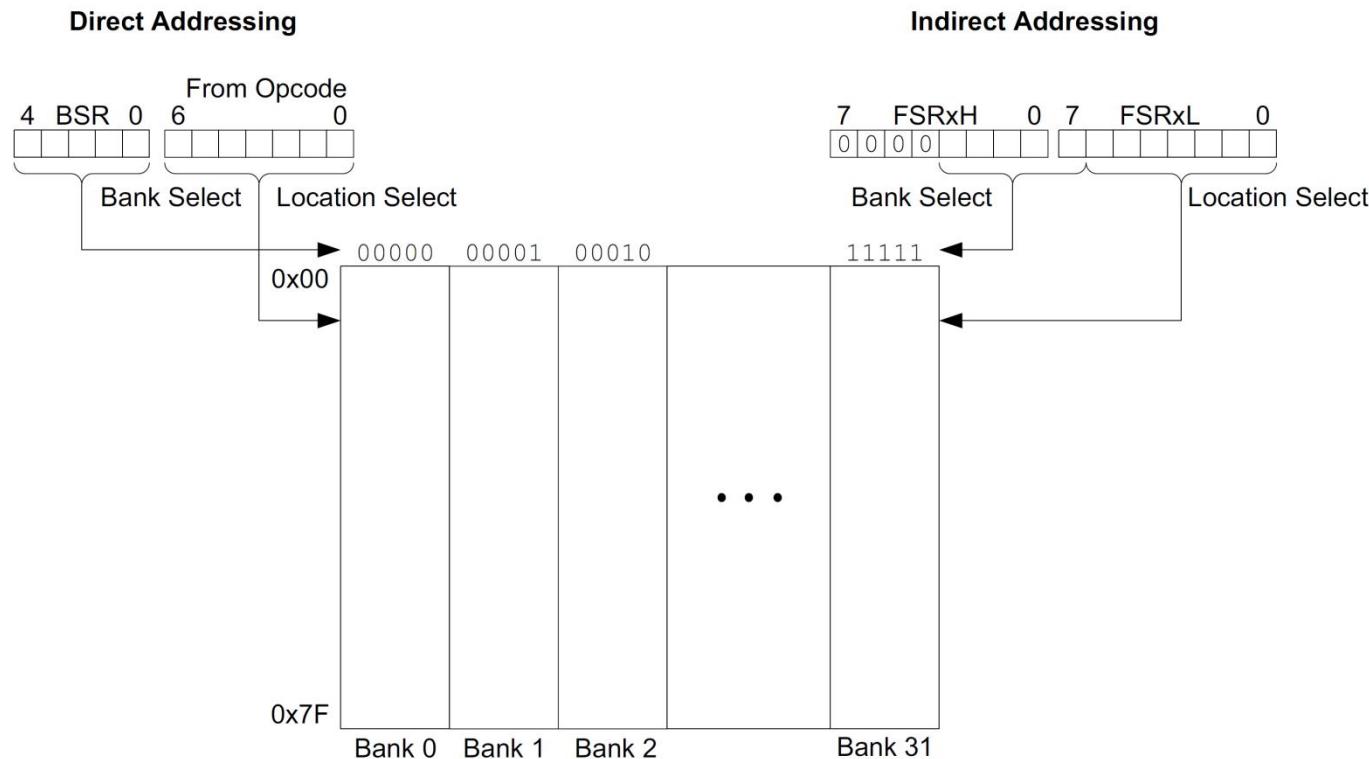
Integrované vývojové prostředí Microchip

- volně ke stažení (bez registrace) pro různé platformy
- pro naše účely stačí samotné bez dopl. rozšíření
- POZOR verze max 5.35 – novější již neobsahují ASM

# Adresování – přímé a nepřímé

## Adresování paměti dat

- *přímé* (7-bitová konstanta „k“ je součástí instrukce)
- *nepřímé* (ukazatelem jsou FSR a obsah je v INDF)



# Vstupně výstupní linky

Obousměrné, 3-stavové výstupy

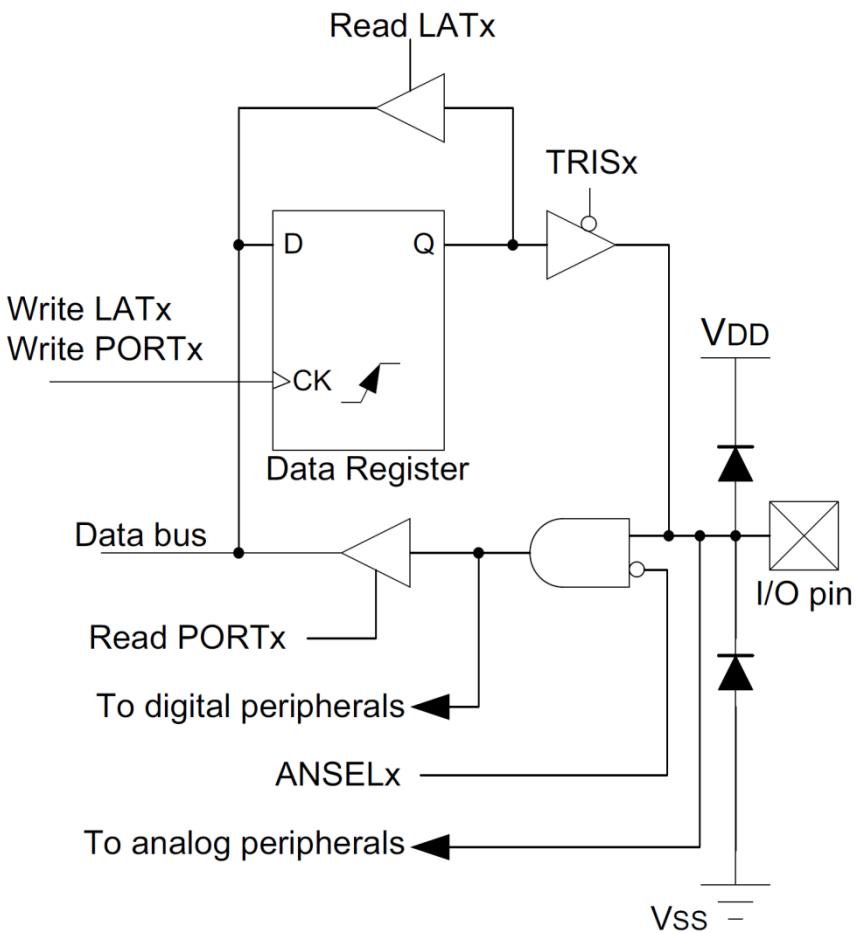
Registr TRISx řídí směr vývodu  
bitem na příslušné pozici

- 0 ... 0out, výstup
- 1 ... 1n, vstup (HiZ)

*Čtení* registru PORTx vrací bity  
podle úrovně na vývodech

*Čtení* registru LATx vrací bity  
výstupního bufferu

*Zápis* bitů do registrů PORTx či  
LATx nastavuje odpovídající  
úroveň na vývodech



# Asembler MPASM

---

Překladač z jazyka symbolických adres do strojového kódu

Vyjádření čísel v číselných soustavách:

desítková	.123	(přepínání bank)
hexadecimální	0x7B	(adresy)
binární	b'01110101'	(nast. registrů po bitech)
ASCII znaky	'A'	

Doporučení pro označování symbolů:

bity, konstanty	velká písmena	CY, MAX
registry	malá písmena	citac, y
návěští	počáteční velké	Skok, Funkce1
	dvojtečka za návěštím (Skok: goto Skok)	

Komentář ;toto je komentář

# Základní pseudoinstrukce

---

- **list**            **p = typ\_procesoru**  
                      ;informace překladači, pro jaký procesor je kód
- **#include**     “*soubor*”  
                      ;vloží soubor s def., podprg., knihovnami...
- **#define**      *název registr, bit*  
                      ;definuje název pro bit registru
- *název*          **EQU**    *hodnota*                    ;definice konstanty  
**ORG**    *hodnota*  
                      ;nastaví adresu pro uložení následujícího kódu
- *název*          **macro**  
                      *instrukce*  
                      ...
- **endm**            ;vytvoří makroinstrukci z inst. v bloku

# Základní pseudoinstrukce

---

- **cblock**      *hodnota*  
                  *název*  
                  ...  
**endc**      ;definuje postupný seznam konst s poč. „hodnota“
- **END**      ;konec kódu
- **goto \$-1**    ;skok zpět o 1 instrukci

# Instrukce přesunu dat

---

MOVF f,d	obsah registru f přesune do W (je-li d=0) nebo zpět do f (je-li d=1, slouží k testování); registr f ~ 0 až 7Fh (127)
MOVLW k	registr W je naplněn konstantou k (8 bitů)
MOVWF f	obsah registru W se přesune do registru f
RLF f,d	rotuje obsah f doleva přes bit C, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
RRF f,d	rotuje obsah f doprava přes bit C, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
SWAPF f,d	zamění spodní a horní 4 bity registru f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)

# Aritmetické a logické operace

---

ADDLW k	sečte registr W s konstantou k, výsledek uloží do registru W
ADWF f,d	sečte W s registrem f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
ANDLW k	logický součin W a k, výsledek uloží do W
ANDWF f,d	logický součin W a f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
DECF f,d	odečte jedničku od f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
INCF f,d	přičte jedničku k f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
COMF f,d	jedničkový doplněk f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)

# Aritmetické a logické operace

---

IORLW k	logický součet W a k, výsledek se uloží do W
IORWF f,d	logický součet W a f, výsledek se uloží do W (je-li d=0) nebo do f (je-li d=1)
SUBLW k	odečte W od konstanty k, výsledek uloží do W (řeší se přičtením dvojkového doplňku W; je-li výsledek $\geq 0$ , pak C=1)
SUBWF f,d	odečte W od registru f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)
XORLW k	nonekvivalence W a k, výsledek uloží do W
XORWF f,d	nonekvivalence W a f, výsledek uloží do W (je-li d=0) nebo do f (je-li d=1)

# Instrukce nulování a nastavení

---

BCF f,b	vynuluje bit b v registru f
BSF f,b	nastaví do log.1 bit b v registru f
CLRF f	vynuluje obsah registru f
CLRW	vynuluje obsah registru W
CLRWD	nuluje čítač WDT a jeho předděličku (je-li k WDT připojená)

# Instrukce skoků v programu

---

BTFSC f,b	je-li bit b v registru f v log.0, následující instrukce se neprovede
BTFSS f,b	je-li bit b v registru f v log.1, následující instrukce se neprovede
DECFSZ f,d	odečte jedničku od obsahu registru f a výsledek uloží do W (je-li d=0) nebo do f (je-li d=1). Je-li výsledek 0, následující instrukce se neprovede
INCFSZ f,d	přičte jedničku k obsahu registru f a výsledek uloží do W (je-li d=0) nebo do f (je-li d=1). Je-li výsledek 0, následující instrukce se neprovede
GOTO a	adresa „a“ se uloží na spodních 11 bitů PC (zbývající 2 b se doplní z 3. a 4. bitu PCLATH) program pokračuje na adresu PC

# Podprogramy a přerušení

---

CALL a	návratovou adresu PC+1 uloží do zásobníku a pokračuje do podprogramu
RETURN	naplní PC ze zásobníku (návrat z podprogramu)
RETLW k	naplní PC ze zásobníku a registr W naplní osmibitovou konstantou k
RETFIE	naplní PC ze zásobníku a povolí se přerušení nastavením bitu GIE do log.1 (návrat z přerušení)

# Zvláštní instrukce

---

NOP                neprovede nic

SLEEP              procesor přejde do stavu „spánku“

Většina instrukcí trvá jeden strojový cyklus

Dva cykly zaberou vždy instrukce skoků CALL, GOTO,  
RETURN, RETLW, RETFIE a nastane-li skok také  
instrukce BTFSC, BTFSS, DECFSZ, INCFSZ

# RESET mikrořadiče

---

RESET znamená nový start procesoru a může být vyvolán z těchto důvodů:

- zapnutí napájení
- nulování vstupního signálu !MCLR (RESET)
- přetečení WDT
- volání instrukce RESET

Při resetu je vynulován čítač instrukcí (program začíná na adrese 0000h), PCLATH je nastaven na 0, všechny porty jsou nastaveny jako analogové(!) vstupní, všechna přerušení jsou zakázána, zápis do EEPROM je zakázán.

# Konfigurační slova

---

Procesor obsahuje *konfigurační slova*, která jsou přístupná pouze během programování procesoru a slouží k nastavení různých režimů, příkaz `_CONFIG`

MPLABX má pro jejich nastavení jednoduché a intuitivní rozhraní s automatickým generováním kódu:

Window → PIC memory views → Configuration bits

Volba zdroje taktu, Watchdog Timer, Power-up Timer, Brown-out, Code & Data Protection...

# Příklad – V/V operace

---

;RS-KO s BT1 a BT2 vstupy a LED1 jako výstupem

list p=16F1508

#include "p16f1508.inc"

```
#define SETBT    PORTA,4
#define RESETBT  PORTA,5
#define LED1      PORTC,5
```

```
_CONFIG _CONFIG1,0xC9E4
_CONFIG _CONFIG2,0xFFFF
```

;\*\*\*\*\*

ORG 0x00

goto Start

ORG 0x04

nop

retfie

;ošetření náhodného skoku do přerušení

Start:

```
movlb .1                      ;Bank1
movlw b'01101000'              ;4 MHz
movwf OSCCON                  ;nastavení hodin
call Config_Ios               ;volání podprogramu pro nastavení portů
```

# Příklad – V/V operace

---

Loop:

```
movlb .0          ;Bank0
btfsC SETBT       ;test stisku SETBT
bsf LED1          ;rozsvítí LED1
btfsC RESETBT    ;test stisku RESETBT
bcf LED1          ;zhasne LED1
goto Loop          ;a stále dokola...
```

```
#include „Config_IOs.inc“
```

```
END
```

# Architektura počítačů

Milan Kolář  
Ústav mechatroniky a technické informatiky



Projekt ESF CZ.1.07/2.2.00/28.0050  
Modernizace didaktických metod  
a inovace výuky technických předmětů.

INVESTICE DO ROZVOJE Vzdělávání

2

## Literatura

- E-learningový portál FM: <https://elearning.tul.cz>
- Šimeček, I.: Moderní počítačové architektury a optimalizace implementace algoritmů. ČVUT FIT Praha, 2015.
- Kubátová, H.: Struktura a architektura počítačů s řešenými příklady. ČVUT FIT Praha, 2013.
- Hlavíčka, J.: Architektura počítačů. ČVUT FEL Praha, 2001.
- Skalický, P.: Přístrojové aplikace mikropočítačů. ČVUT FEL Praha, 2004.
- Pluháček, A.: Projektování logiky počítačů. ČVUT FEL Praha, 2000.
- Pinker, J.: Mikroprocesory a mikropočítače. BEN, Praha, 2004.
- Smékal, Z. – Sysel, P.: Signálové procesory. Sdělovací technika, Praha, 2006.



## Definice architektury

### Architektura počítačů – více definic

- technický obor zaměřený na návrh a konstrukci zařízení na zpracování dat;
- soubor pravidel a metod, které popisují **funkčnost, organizaci a implementaci** počítačových systémů;
- zahrnuje návrh **architektury instrukční sady**, návrh logiky, **návrh mikroarchitektury** (počítačové organizace) a **implementaci** v konkrétním počítači.

Znalost architektury má být prostředkem pro vytváření nových systémů, má být podkladem pro hodnocení kvality výsledku.



## Počítač

Stroj na číslicové zpracování informací (dat).

Zařízení, které provádí výpočty nebo řídí operace, které jdou popsat čísla nebo logickými výrazy (Oxfordský slovník)

Vrstvy abstrakce počítače:



## Části počítače

CPU (Central Processing Unit) - procesor

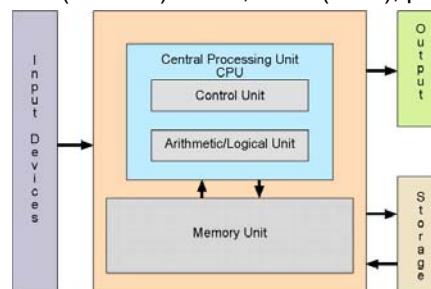
ALU (Arithmetical and Logical Unit) – aritmeticko-logická jednotka

CU (Control / Central Unit) – řadič, řídící jednotka

I/O (Input / Output) Devices – vstupně / výstupní zařízení

Memory – paměť ( operační)

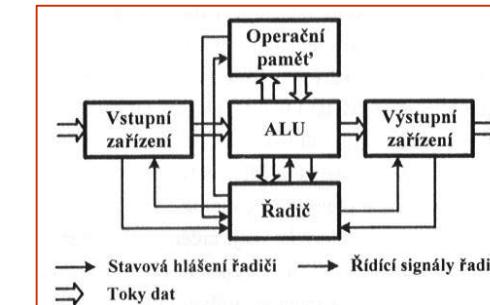
Storage – paměť (archivní) – disk, karta (flash), páška



5

## Von Neumannova architektura

představena v roce 1946



6

## Von Neumannova architektura

Charakteristické vlastnosti lze shrnout do následujících bodů:

- 1) struktura počítače je nezávislá na typu řešené úlohy, počítač se programuje obsahem paměti (lze řešit jakýkoli algoritmicky řešitelný problém);
- 2) instrukce a operandy jsou v téže paměti;
- 3) paměť je rozdělena do buněk stejné velikosti, jejich pořadová čísla se používají jako adresy;
- 4) program je tvořen posloupností elementárních příkazů (instrukcí), které se provádějí jednotlivě v pořadí, v němž jsou zapsány do paměti;
- 5) změna pořadí provádění instrukcí se vyvolá instrukcí podmíněného nebo nepodmíněného skoku;

7

## Von Neumannova architektura

- 6) pro reprezentaci instrukcí i čísel se používají dvojkové signály a dvojková číselná soustava;
- 7) programem řízené zpracování dat probíhá v počítači samočinně (tok dat řídí řadič);
- 8) zpracování dat probíhá v tzv. diskrétním režimu (během výpočtu nelze s počítačem komunikovat);
- 9) vstupy (resp. výstupy) jsou koncipovány jako datové zdroje (resp. výsledky) a jsou tedy přímo napojeny na ALU.

Nevýhody:

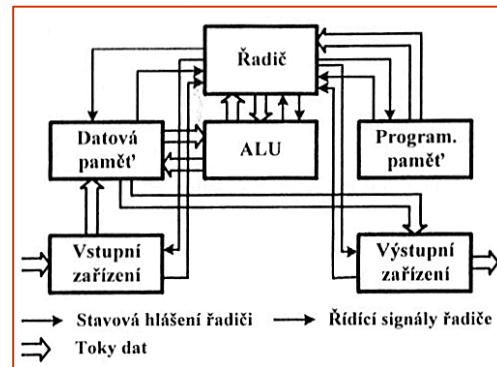
Může být obtížnější ladění programu, méně bezpečné, možnost mylně interpretovat data jako program, data i instrukce se přenáší po stejné sběrnici.

8

## Harvardská architektura

vznikla v roce 1943 (koncepce IBM Harvard MARK1)

někdy označována jako princetonská architektura



9

## Harvardská architektura

Základní principy (rozdíly vůči von Neumannové architektuře):

- 1) paměť programu je oddělena od paměti dat
  - možnost ve stejném okamžiku načítat instrukci a přistupovat k datové paměti,
  - datová a programová paměť mohou mít odlišnou organizaci,
  - paměť programu může být non-volatilní;
- 2) oddělené sběrnice (datová, instrukční, adresové);
- 3) řízení procesoru je odděleno od řízení vstupních a výstupních jednotek (nejsou napojeny přímo na ALU).

Vhodné pro zpracování většího objemu dat (rychlejší).

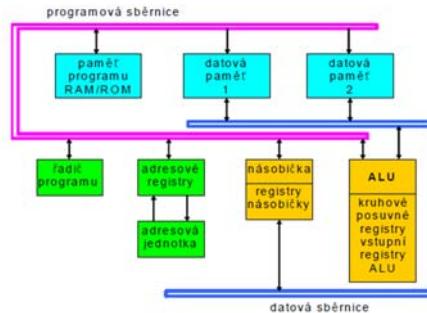
Použití zejména v mikrořadičích či signálových procesorech a v souvislosti s redukovánou instrukční sadou (RISC).

10

## Modifikované architektury počítačů

V moderních architekturách existují různé modifikace obou zmínovaných architektur, např.:

- oddělená paměť dat a paměť programu, avšak společná adresová a datová sběrnice (umožňuje zacházet s instrukcemi jako daty, např. přenést část kódu do paměti dat);
- uvnitř procesoru je použita harvardská architektura, kde se cache dělí na paměť instrukcí a paměť pro data, ale celý počítač se zvenku tváří jako von Neumann, protože načítá data i program z hlavní paměti najednou;
- sdílení programové sběrnice i pro přenos dat.



11

## Generace počítačů

Každá generace je charakteristická svou konfigurací, rychlosí počítače a základním stavebním prvkem

Generace	0	1	2	3	4
Rok	1940	1951	1957	1964	1981
Prvky	relé	elektronky	tranzistory	IO SSI	IO (V)LSI
Hlavní paměť		buben	ferity	ferity	LSI
Kapacita paměti		1 kB	10 kB	1 MB	10 MB
MIPS	0,001	0,01	0,1	1	10
Příklad	Mark I	Univac 1	IBM 7090	IBM 360	Intel 4004

12

## Čtvrtá generace

Její vývoj prožíváme dodnes;

základem je centrální procesorová jednotka (CPU) označovaná jako mikroprocesor (vesměs z křemíku);

IO LSI a VLSI (až  $10^{18}$  tranzistorů na čipu);

malé rozměry (technologie 45 → 32 → 22 → 14 nm);

velká rychlosť – využití paralelismu a zavádění programovacích prostředků, které paralelismus podporují;

Již mnoho let se mluví o 5. generaci – orientace na využití umělé inteligence, nalézá algoritmy řešení, přímý styk s uživatelem na úrovni přirozeného jazyka, textu a obrazů, distribuovaný HW.



## Výkonnost počítače

Viz 1. cvičení

**Výkonnost** – převrácená hodnota doby vykonání jednoho úkonu

**Propustnost** – počet úkonů za jednotku času (množství vykonané práce za jednotku času)

Do celkové výkonnéosti počítače z pohledu uživatele je třeba počítat i čekání na I/O operace, režii OS (včetně sdílení zdrojů s jinými uživateli) apod.

**Výkonnost procesoru** – dána výkonnéostní rovnicí CPU

$$T_{CPU} = IC \cdot CPI \cdot T_{clk} \quad (\text{jednoprocесоровý systém bez cache})$$

Není vhodné porovnávat jednotlivé veličiny samostatně (např. MIPS)

Průměrný počet taktů na instrukci (CPI) závisí na **instrukčním mixu**.

## Instrukční mixy

**Instrukční mixy** jsou seznamy (tabulky) nejfrekventovanějších instrukcí ohodnocených pravděpodobnostmi jejich výskytu v rámci daného typu zátěže

Nevýhody:

- instrukční mixy jsou zpravidla závislé na konkrétním procesoru a architektuře – obtížná přenositelnost;
- frekvence použití jednotlivých instrukcí závisí subjektivně na programátorech, na druhu zpracovávaných úloh i na souborech vstupních zpracovávaných dat;
- dobu instrukcí ovlivňuje i operační systém.

Nejstarší způsob hodnocení propustnosti číslicových systémů (dnes v podstatě nepoužívaný)



## Zkušební úlohy (benchmarky)

**Zkušební úloha** je vzorek zátěže, který má ověřit propustnost počítače v rámci určité aplikaci oblasti.

Výhodou je komplexnost – na jejich běhu se nepodílí jen procesor (jako u mixů), ale jsou ovlivněny i operačním systémem, překladačem, vstupy a výstupy, atd.

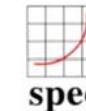
Skupiny zkušebních úloh:

**Reálné (přirozené) aplikace** – převzaté z běžného provozu počítače (překladač, text. editor, komprimace) - problém s přenositelností (závislost na OS nebo překladači);

**Umělé (synthetic benchmarks)** – vzniklé pouze za účelem zjišťování výkonnéosti (v současnosti nejčastější) např. Whetstone (WHIPS), Dhrystone (DIPS).

## SPEC

(Standard Performance Evaluation Corporation)



Eliminace slabin jedné zkušební úlohy vedla k vytvoření sad zkušebních úloh => SPEC - skládá se z reálných aplikací z různých vědeckých a inženýrských aplikací:

- pro osobní počítače (desktop benchmarks),
- pro servery (server benchmarks),
- pro vestavěné poč. systémy (embedded benchmarks), ...

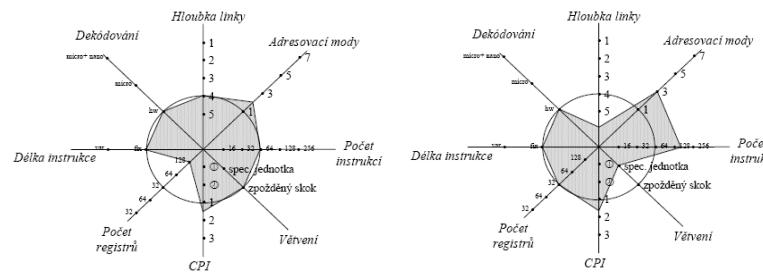
Původně hlavně pro výkonnost CPU, dnes i GPU, cloudů, databází, web serverů, emailových systémů apod.

Vývoj: speciální benchmarky pro různé typy HW (SPEC CPU2017, SPECviewperf 13, SPECmail2009, SPEC Cloud\_IaaS 2018, SPEC SFS 2014, ... (více na [www.spec.org](http://www.spec.org))

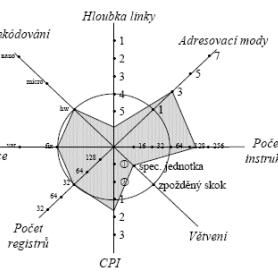
17

## Hodnocení vyváženosti architektur

**Kiviatův graf** – obecně  $n$  radiálních os ( $n$  vlastností s různými metrikami) – pro charakterizaci procesorů



Sparc



IBM RS/6000

## Monitorování výkonnosti počítačů

Na rozdíl od zkušebních úloh cílem monitorování je zjištění skutečného chování počítače v čase prostřednictvím hodnot několika předem zvolených stavových proměnných (např. stav procesoru, paměti, sběrnice, V/V zařízení)

- *programový monitor* – snadná realizace, ale nepatrne zdržuje vlastní činnost počítače
- *obvodový monitor* – samostatný funkční blok, který neovlivňuje počítač a je s ním spojen sondami
- *kombinovaný monitor*

Monitory mohou sledovat nejen činnost OS, ale i požadavky uživatele nebo vyváženosť konfigurace počítače.

18

# Procesory I.

Milan Kolář  
Ústav mechatroniky a technické informatiky



Projekt ESF CZ.1.07/2.2.00/28.0050  
Modernizace didaktických metod  
a inovace výuky technických předmětů.

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

2

## Procesor - definice

**Procesorem** se rozumí základní jednotka počítače, tj. logický automat pro zpracování informací, obsahující aritmetickou jednotku a řadič (počítač bez periferních zařízení a bez hlavní paměti); chování je definováno programem.

**Mikroprocesor** je malý procesor vyráběný technologií velké integrace, určen především na výpočty a logické funkce (od 70. let 20. století);

Je-li mikroprocesor orientován pouze na logické bitové operace, mluvíme o *logickém (booleovském) mikroprocesoru*.

V hrubém členění procesor obsahuje:

- datovou cestu (reprezentuje tok dat v CPU; především ALU);
- řízení toku dat (řadič, stavový automat).

**Koprocesor** – přídavný procesor provádějící specializované operace (úzce spolupracuje s CPU) – grafický, matematický, V/V, signálový ...



## Kategorie procesorů

- Univerzální (Intel, AMD, ...)
- Grafické (Nvidia, ATI, ...)
- Signálové (TI, AD, ...)
- Aplikační (pro mobilní telefony, ...)
- Multimedialní (TI, Mpact, ...)
- Speciální (šifrovací, kompresní, hračí, ...)

## Dělení podle platformy (použití)

- Osobní počítače
- Notebooks (mobilní technika)
- Pracovní stanice
- Servery (datové, komunikační, tiskové, databázové)
- Palmtopy, handheldy, pocket PC, PDA
- Grafické karty
- Embedded (vestavěné) aplikace
- Herní konzole
- Automobilová technika
- Telekomunikační technika
- ...

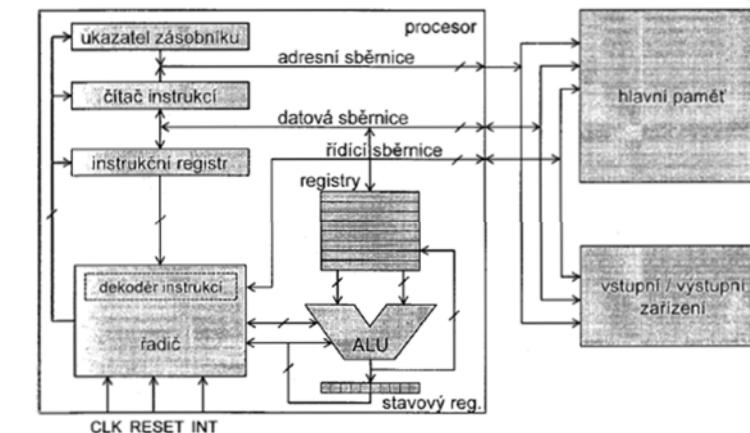
## Složení CPU

Procesor se skládá z:

- **řídící jednotky CU** (Control Unit) – řadič;
- **aritmeticko-logické jednotky ALU** (Arithmetic Logic Unit)
- **sady registrů RS** (Register Set), které uchovávají různé hodnoty během práce počítače (zápisníková paměť)  
ALU + RS je někdy označuje jako **operační jednotka**;
- **programového čítače PC** (Program Counter) – často se uvádí jako jeden registr RS nebo součást řadiče;
- **vnitřní sběrnice** – řeší spojení mezi bloky CPU (typy - datová, adresová, řídící), od každého typu může být v architektuře i více sběrnic; někdy se vyčleňuje **systémová sběrnice** (mezi vyrovnávací pamětí a operační pamětí, příp. můstkem).

5

## Blokové schéma CPU



6

## Řadič

Část procesoru řídící vykonávání operace a chod celého procesoru podle instrukcí programu.

Obsahuje **registrový instrukcí**, který uchovává operační znak instrukce po dobu jejího vykonávání, a **dekodér instrukcí**, který dekóduje a generuje řídící signály pro procesor.

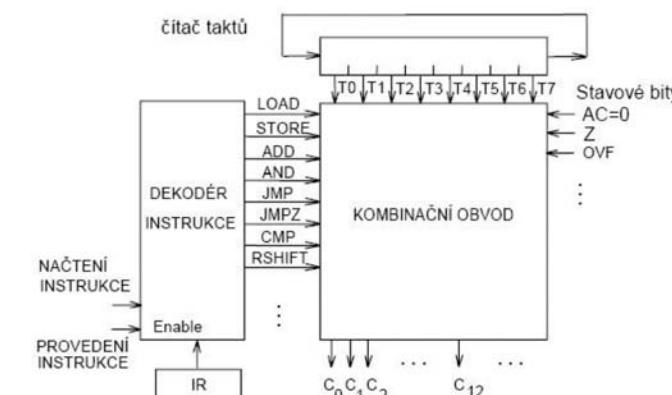
Existují 2 koncepce řadičů:

- 1) řadič je speciální sekvenční automat, který má čítač a dekodér (je dražší, ale rychlejší) – **obvodový řadič** (založen na kombinačních logických obvodech);
- 2) dekódování operačního znaku vykonává řadič paměť, ve které jsou mikroprogramy uloženy – **mikroprogramový řadič** (založen na výběru z paměti ROM).

7

## Obvodový řadič

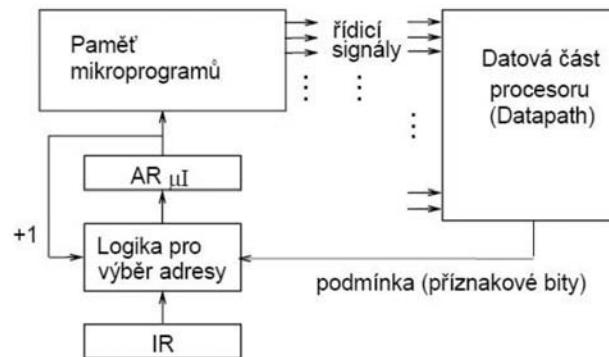
IR – instrukční registr (uchovává adresu další instrukce)



8

## Mikroprogramový řadič

AR – adresový registr (uchovává adresu pro čtení/zápis z/do paměti)



9

## Řadiče - rozčlenění

Na řadič lze pohlížet jako na soustavu místních řadičů:

- řadič provedení instrukce ALU s pevnou řádovou čárkou,
- řadič provedení instrukce ALU s pohyblivou řád. čárkou,
- řadič přerušení,
- řadič I/O (řadič kanálu),
- řadič paměti,
- řadič DMA (Direct Memory Access),
- řadiče jednotlivých periferií
- ...

10

## Aritmeticko-logická jednotka

**ALU** je část procesoru, ve které se provádějí všechny aritmetické (např. sčítání, násobení, bitový posuv, ...) a logické (logický součin, negace, ...) výpočty.

- HW řešen jako číslicový obvod (kombinační či sekvenční)
- součástí obvykle bývá **příznakový (status) registr**

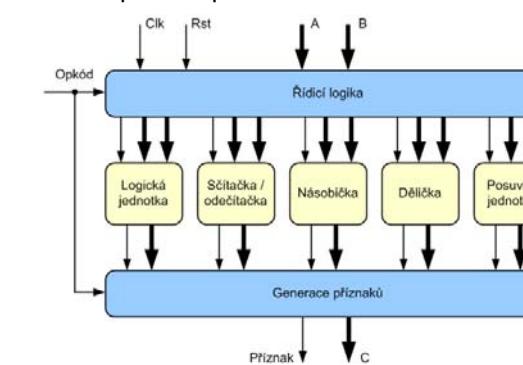
Procesor může mít více než jednu ALU (stále častější) - obvykle rozdelené na jednotky pro práci s celočíselnými operandy a jednotky pro práci s operandy v plovoucí řádové čárce (ty se někdy označují jako **FPU**, floating-point unit).

Jednotlivé ALU pracují relativně nezávisle, mohou i souběžně řešit operace v různých instrukcích. Větší samostatnost specializovaných výpočtů může vést až k použití **koprocesorů** (nezávislé na CPU, může samostatně zpracovávat instrukce).

11

## Blokové schéma ALU

Liší se zejména možnostmi aritmetických operací, formátem operandů (pevnou a pohyblivou řádovou čárkou), mírou paralelizace, počtem hod. taktů na operaci apod.



12

## Příznakový (stavový) registr

Příznakový registr (PSW – Program Status Word) zpravidla obsahuje:

- **C** (CY, carry) – přenos z MSB  
někdy zvlášť **B** (BO, borrow) – výpůjčka při odečítání
- **Z** (zero) - nulovost operace
- **S** (sign), **N** (negative) – záporné znaménko
- **V** (OV, overflow) – přetečení čísla se znaménkem, dělení 0
- **H** (half-carry), **AC** (auxiliary carry) – poloviční  
(pomocný) přenos – při součtu, mezi 3. a 4. bitem
- **P** (parity) – parita, tj. sudý nebo lichý počet jedniček

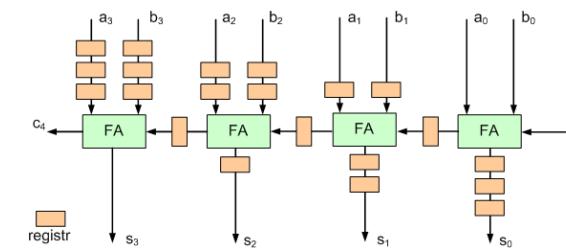
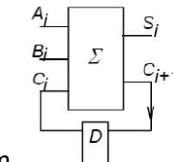
Příznaky (Flags) se nastavují podle výsledků operací v ALU, pokud tyto operace příznaky mění (zvláštní výstup z ALU)  
- využívají se pro větvění programu.

13

## HW realizace sčítáčky

Sčítáčky:

- úplné
- neúplné (polosčítáčky)
- sériové (SLO)
- sério-paralelní (SLO)
- paralelní (KLO) - např. s postupným přenosem
- s proudovým zpracováním

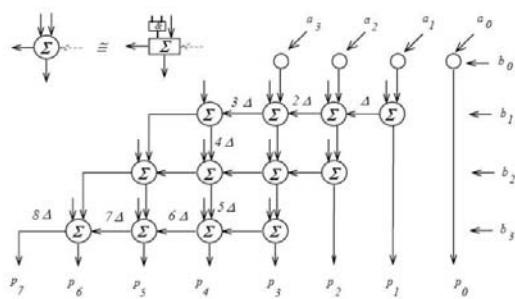


14

## HW realizace násobičky

Násobičky:

- sériové (sekvenční) – obsahují SLO a násobení trvá více taktů (typické pro procesory CISC)
- paralelní (kombináční) – obsahují KLO a násobení se provede na jeden takt (typické pro procesory RISC)



15

## Aritmetika FP

Sčítání, odečítání, porovnávání

- srovnat exponenty, provést danou operaci s mantisami

Násobení - sečist exponenty, vynásobit mantisy

Dělení - odečist exponenty, vydělit mantisy

Posuv - zvětšení/zmenšení exponentu

Normalizace

- posuv mantisy co nejvíce vlevo a úprava exponentu
- normalizace operandů nezaručuje normalizovaný tvar výsledku (nutno provést po každé operaci)

16

## Instrukce

Specifikace jednoduché činnosti, kterou má provést technický prostředek (nejčastěji procesor).

Binární tvar instrukce se skládá většinou:

- a) z operačního kódu (opkód) - určuje typ instrukce
- b) z parametrů, které mohou být:
  - do instrukce zakódované konstanty,
  - označení registrů, odkud vzít hodnotu, příp. kam zapsat,
  - adresa paměti, odkud načíst hodnotu, příp. kam zapsat,
  - adresa paměti, kam má skočit další provádění programu.

Instrukce lze zapisovat v různých tvarech:

- strojový zápis ve formě binárních čísel (strojový kód),
- častěji zápis v jazyku symbolických adres (mov ah, 56h).

Soubor instrukcí tvoří program.

17

## Mikroinstrukce

Jedna instrukce (příkaz) se v moderních procesorech nevykonává přímo, ale nejdříve se rozloží na jednodušší úkony, kterým se říká **mikroinstrukce**,  
=> vzniká odlišná externí a interní instrukční sada.

Dochází ke zvýšení výkonu a přitom se zachovává zpětná kompatibilita procesorů.

Mikroinstrukce je nejčastěji uložena v jedné buňce paměti ROM a její vykonávání trvá jeden hodinový takt (velké nároky vybavovací dobu paměti).

18

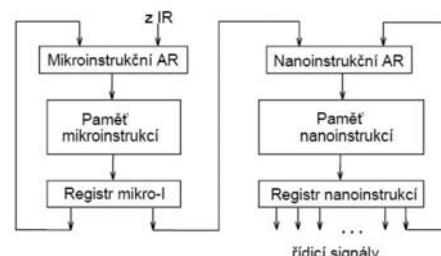
## Nanoinstrukce

U některých procesorů má mikroinstrukce funkci adresy do sekundární řídící paměti (nanopaměť), ze které se teprve čtou řídicí signály  
=> dvouúrovňový paměťový systém.

Obsahy buněk paměti nanoinstrukcí označujeme jako **nanoinstrukce**.

Přínos: - snížení kapacity mikropaměti

- zjednodušení návrhu mikroprocesoru



19

## Kódování instrukce

Kódování instrukce je kvalitativním popisem ISA

V každé instrukci musí mít zakódován

- druh operace (nejjednodušší – libovolný jedinečný kód)
- umístění operandů a výsledku (závisí na módech adresování, které ISA podporuje)

Obecná instrukce je sledem binárně zakódovaných informací:

operace	operand 1	operand2	...	operand n / přímý operand / adresa
---------	-----------	----------	-----	------------------------------------

Operandem bývá zpravidla adresa registru, adresa paměti, nebo hodnota / konstanta.

20

## Kódování instrukcí (pokračování)

U více operandů (GPR ISA) je první operand zpravidla cíl, další operandy jsou zdroje operací, poslední segment může být také přímý operand nebo adresa.

Délky jednotlivých segmentů mohou být různé:

- délka segmentu s typem operace musí být schopna zakódovat všechny druhy operací (nejčastěji 8 bitů);
- délka „registrověho“ operandu závisí na počtu registrů;
- délka přímého operandu závisí na délce slova procesoru;
- délka adresy či segmentu adresy závisí na šířce adresové sběrnice procesoru a způsobu adresování.

21

## Hlavní módy adresování

	Adresování	Příklad	Poznámky
Nultého řádu	Implicitní	OP Rx	Jeden z operandů (v případě ZO ISA všechny) je implicitním cílem, resp. zdrojem operace
	Přímým operandem	OP Rx, Num	Operand (číslo, adresa) je přímo obsažen v instrukci
	Registrové	OP Rx, Ry	Zdrojem operace (u GPR ISA také cílem) jsou registry
Prvního řádu	Nepřímo registrem	OP [Rx]	Hodnota registru Rx je ukazatelem na paměť (mem[Rx])
	Přímo adresou	OP Address	Adresa v instrukci je ukazatelem paměťové buňky (mem[Address])
Druhého řádu	Nepřímo pamětí	OP [Address]	Adresa v instrukci je ukazatelem na ukazatel v paměti (mem[mem[Address]])

22

## Formáty kódování instrukcí

### Proměnná délka kódu instrukce

- lepší hustota kódování (úspora paměti)
- každý operand může mít svůj adresní mód
- složité dekódování (vyžaduje čas nebo plochu čipu)
- typické pro CISC procesory  
(např. délka kódu instrukce IA32 (x86) je od 1B do 13B)

### Pevná délka kódu instrukce

- rychlé a jednoduché dekódování
- snazší implementace proudového zpracování (pipeline)
- adresní mód je součástí kódu operace
- omezená implementace adresních módů
- programy zabírají více paměti
- typické pro RISC procesory

23

## Instrukční sada

Seznam instrukcí procesoru, datových typů a dostupných režimů,  
seznam registrů, seznam pravidel (adresování, přerušení, ...)

Procesory mající stejnou instrukční sadou jsou vzájemně kompatibilní.

Možno dělit podle různých pohledů:

**Podle konceptu architektury** – obecný popis organizačních, funkčních a provozních principů procesoru  
(ISA – Instruction Set Architecture)

**Podle rozsahu instrukční sady:**  
– RISC, CISC

**Podle implementace (mikro)architektury:**  
– x86, IA-32 (i386), IA-64, SPARC, ARM, ...

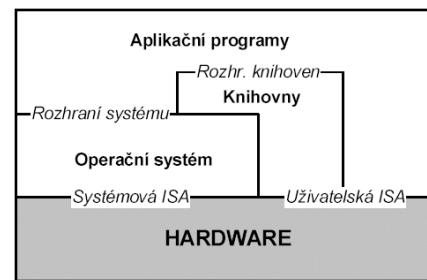
**Podle míry paralelnosti:**  
– subskalární, skalární, superskalární (např. VLIW)

24

## ISA

**Architektura souboru instrukcí** – představuje v podstatě rozhraní mezi SW a HW počítače

Souhrn vlastností počítačového systému viděného z pohledu programátora v strojovém jazyce (koncept struktury, funkčního chování).



25

## ISA

Musí být definováno:

- způsob kódování instrukcí (pořadí operací, operandů a adres),
- zacházení s operandem (způsoby adresování),
- výčet možných operací - instrukcí (aritmetických, logických, skoky, ...),
- způsob ukládání výsledku (střadač, registr, paměť, zásobník),
- datové typy a velikosti operandů,
- výběr následujících instrukcí (větvení) – varianty podmíněných skoků, volání a návraty z podprogramu, způsoby přerušení.

26

## Typy ISA

- střadačově (akumulátorově) orientovaná ISA (Accumulator)
- zásobníkově orientovaná ISA (Stack)
- ISA s univerzálními registry (registrová) (GPR – General Purpose Register)

Díky zpětné binární kompatibilitě nelze ISA často měnit.

ISA souvisí s vývojem počítačového HW, s jeho rostoucí hustotou integrace, zrychlováním a zlevňováním.

27

## Střadačově orientovaná ISA

Implicitním operandem aritmeticko-logických instrukcí (zdrojovým i cílovým) je speciální registr uchovávající hodnotu předchozí operace – **střadač** (akumulátor).

Druhým (explicitním) operandem může být univerzální registr nebo paměťová buňka ( $\Rightarrow$  jednoadresní architektura).

Charakteristické vlastnosti:

- snadno kódovatelné a krátké instrukce
- rychlé přepínání kontextu (omezený vnitřní stav CPU)
- častý přístup do paměti
- problematická realizace paralelismu mezi instrukcemi

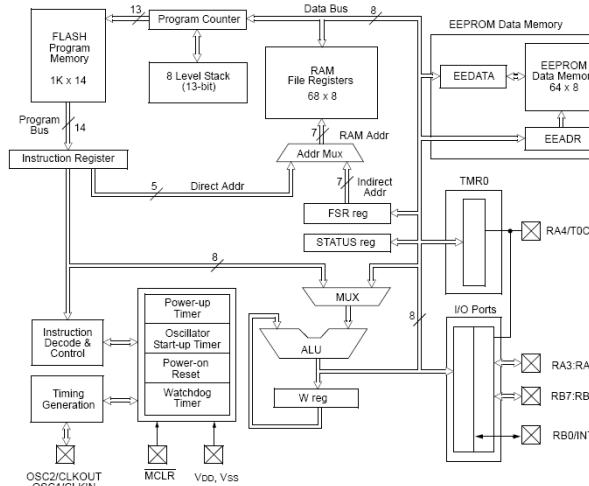
Vede na poměrně jednoduchý hardware (dekodér instrukcí).

Typický představitel – Intel 8080, 8051, 68HC05

28

## Střadačově orientovaná ISA

Příklad:  
PIC16F84



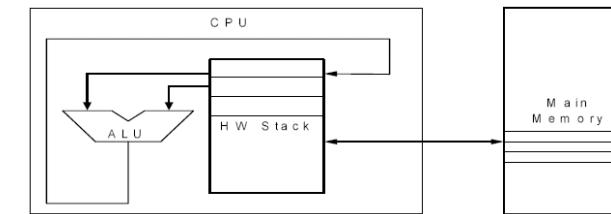
29

## Zásobníkově orientovaná ISA

ZO ISA se používá výjimečně (používá se SW implementace - Java Virtual Machine).

**Bezadresní** - základní instrukce: PUSH (načtení dat z paměti do zásobníku), POP (přesun dat ze zásobníku do paměti) a operace ALU.

Někdy se označuje **MISC** (Minimal Instruction Set Computer).



30

## Zásobníkově orientovaná ISA

Zdrojem a cílem všech operací je sekvenční HW zásobník (registrově pole s ukazovátkem vrcholu – top).

Výhody: jednoduché a rychlé instrukce (bez operandů), vysoká hustota kódování (krátké programy), rychlá implementace či emulace.

Nevýhody: komplikovaný přístup k datům v paměti (díky sekvenčnosti zásobníku, obtížný paralelismus operací, nemožnost náhodného přístupu k lokálním proměnným).

Řadic musí řešit přetečení nebo podtečení HW limitovaného zásobníku (přesun dat mezi zásobníkem a hlavní pamětí - stack spilling); nekonečný SW zásobník v hlavní paměti.

31

## Zásobníkově orientovaná ISA

Příklad – spočítat výraz:  $a * a * (b + 4 * a) + 3 * 2$

```

PUSH a
PUSH a
PUSH b
PUSH 4
PUSH a
MUL      ; 4 * a
ADD      ; 4 * a + b
MUL      ; (4 * a + b) * a
MUL      ; (4 * a + b) * a * a
PUSH 3
PUSH 2
MUL      ; 3 * 2
ADD      ; 3 * 2 + (4 * a + b) * a * a

```

Výsledek zůstává na vrcholu zásobníku

32

## ISA s univerzálními registry

- nejpoužívanější současná architektura (od 80. let min. století)

Základem je soubor velmi rychlých univerzálních registrů (GPR), které mohou být jak zdroji, tak cíli vykonávaných operací (mohou obsahovat mezivýsledky i proměnné)

Počet registrů se pohybuje od 8 do 128

Instrukce může mít 2–3 operandy (**2 až 3 adresní architekt.**)

Výhody:

- registry jsou rychlejší než paměť – včetně cache
- přístup k registrům může být náhodný
- méně časté přístupy do paměti → urychlení
- snadná implementace paralelismu

33

## ISA s univerzálními registry

Nevýhody:

- limitovaný počet registrů
- složitý překladač optimalizující využití registrů
- registry nemohou uchovávat složené datové struktury (záznamy, pole ...)
- přepnutí kontextu trvá delší dobu (ukládání více registrů)

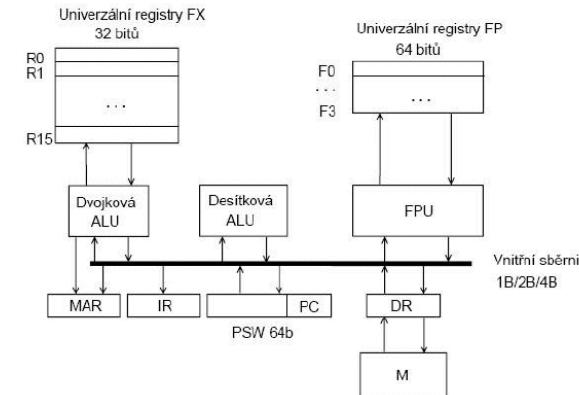
Typický představitel – Intel x86 (do i386 označovaná „multiple accumulator ISA“ - vícestřadačová)

Registry nemusí být všechny nutně univerzální a mohou se specializovat – datové, adresové, speciální (např. pro řízení periferií), kontrolní a stavové (zaznamenávají chod programu – např. programový čítač, ukazatel zásobníku, stavový registr).

35

## ISA s univerzálními registry

Příklad GPR architektury



34

## ISA s univerzálními registry

## ISA s univerzálními registry

Rozlišujeme varianty:

**R-R** ... oba operandy musí být v registrech; výhodou je jednoduché kódování instrukčního souboru, pevná délka instrukce, konstantní CPI, nevýhodou může být vyšší počet instrukcí výsledného programu; typické pro RISC.

*Architektura Load/Store – 2 kategorie instrukcí: instrukce pro přístup k paměti a instrukce pro práci s daty (registry a ALU).*

**R-M** ... jeden operand může být v paměti; výhodou je přímý přístup k datům bez meziúkládání, dobrá hustota kódu, nevýhodou jsou různé CPI; typické pro CISC.

**M-M** ... oba operandy mohou být v paměti; úspora registrů, proměnná délka instrukce, paměť může být úzké místo, typické pro kompletní CISC (zastaralé).

36



## Srovnání architektur ISA

Příklad výpočtu  $C = A + B$       A, B, C ... buňky v paměti

Střadačová	Zásobníková	Registráv (R-M)	Registráv (R-R)
Load A	Push A	Load R1, A	Load R1, A
Add B	Push B	Add R1, B	Load R2, B
Store C	Add	Store C, R1	Add R3, R1, R2
	Pop C		Store C, R3

37

Příklad výpočtu  $R1 = R1^2 + R1^3$

Střadačová ISA		Registráv ISA	
Instrukce	Význam	Instrukce	Význam
LD R1	$A \leftarrow R1$	MUL R2, R1, R1	$R2 \leftarrow R1 * R1$
MUL R1	$A \leftarrow A * R1$	MUL R1, R2, R1	$R1 \leftarrow R2 * R1$
ST R2	$R2 \leftarrow A$	ADD R1, R1, R2	$R1 \leftarrow R1 + R2$
MUL R1	$A \leftarrow A * R1$		
ADD R2	$A \leftarrow A + R2$		
ST R1	$R1 \leftarrow A$		

38

## CISC

*Complex Instruction Set Computer*

Dříve byly operační paměti výrazně pomalejší než procesory

(doba přístupu byla několik taktů CPU),

- zpomalování výpočtu opakováním načítání instrukcí,

- ⇒ snaha rozšiřovat instrukční soubor,

- ⇒ mnoho složitých instrukcí používaných jen zřídka;

- kratší programy (méně instrukcí) => úspora drahých pamětí,

- méně instrukcí znamená méně přístupů do (pomalé) paměti,

- používali se programovací jazyky na nižší úrovni,

- byly méně dokonalé komplikátory,

=> přesun složitosti ze SW do HW (řadiče), menší rozdíl v používání úrovni programovacího jazyka.

## CISC

- proměnná délka instrukcí,
- zpracování instrukcí ve více strojových cyklech ( $CPI \sim 5-10$ ),
- velký počet adresovacích módů,
- díky vysoké složitosti byl řadič navržen na principu paměti s mikroprogramy (ROM),
- implementované mikroprogramy je možné snadno změnit,
- řídicí obvody (dekodér instrukcí) zabírají na čipu velký prostor,
- pro překlad programů bývá zpravidla jednodušší překladač,
- používá se zpravidla GPR ISA (varianta R-M, M-M),
- s postupem doby se začíná používat zřetězené zpracování (zejména s rozkladem na mikroinstrukce).

39

40



## RISC

*Reduced Instruction Set Computer* (použit poprvé 1974) počátkem 80. let první RISC procesory.

Snaha přesunout některé složité a zřídka používané CISC instrukce z mikroprogramů do programů (z HW do SW)

- malý počet relativně jednoduchých instrukcí (důležitý není až tak počet, ale jednoduchost) (asi 40–150),
- zvětšení počtu instrukcí v programu, ale snížení CPI (pod 1,5). (oproti CISC je CPI výrazně nižší než nárůst IC => výkon roste),
- jednoduché instrukce navíc umožňují vyšší frekvenci  $T_{clk}$ ,
- instrukce mají většinou pevnou délku a malý počet formátů,
- podpora proudového zpracování instrukcí,
- důraz na dokonalejší komplikátory (včetně podpory pipeline),

41



## Post-RISC

Do této kategorie je možné zařadit většinu současných CPU

- někdy označovány jako CRISC (*Complex RISC*),
- kombinace CISC a RISC (navenek CISCOVÉ, ale vnitřní konstrukci mají RISCovou),
- instrukce trvají různě dlouhou dobu,
- instrukce se rozloží na jednoduché mikroinstrukce,
- proudové zpracování mikroinstrukcí,
- větší množství paralelních operací (větší míra paralelu),
- spekulativní provádění instrukcí,
- zpracování instrukcí mimo pořadí,
- v 1 taktu lze vykovat i více instrukcí (i nesuperskalární CPU),
- na čip se vejdu složitější dekodéry instrukcí,
- nadále dochází k dalšímu rozširování instrukční sady, zaměřené hlavně do multimediální oblasti a grafiky,
- většinou se zachovává zpětná kompatibilita.

43



## Základní rysy RISC

- řadič s pevnou logikou místo mikroprogramování (rychlé),
- řídicí obvody zabírají méně místa (obvodový řadič) oproti CISC,
- velký počet programově dostupných registrů (32–192),
- operace s daty pouze nad registry (2 zdrojové, 1 cílový),
- registry jsou víceúčelové (jednodušší překladače),
- přístup do paměti pouze pomocí instrukcí přesunu (mluvíme o architektuře Load/Store – instrukce Load a Store),
- malý počet adresových módů (3–5),
- ortogonální instrukční soubor (ve všech instrukcích, které používají registr procesoru jako zdrojový nebo cílový operand, lze použít libovolný registr),
- nejčastěji harvardská architektura.

42



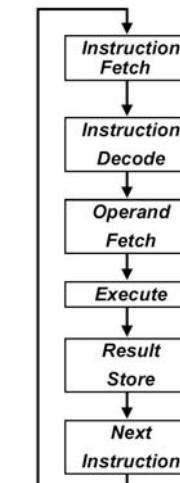
## Instrukční cyklus

Definován jako skupina všech dob nezbytných k uskutečnění jedné strojové instrukce.

Periodu instrukčního cyklu lze rozdělit na několik fází:

### načtení instrukce (Instruction Fetch – IF)

- CPU pošle na AB (Address Bus) adresu instrukce z PC (Program Counter);
- paměť předá na DB (Data Bus) obsah buněk;
- kód instrukce je z DB zapsán do IR (Instruction Register).



44

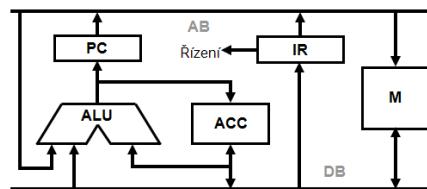
## Instrukční cyklus (pokračování)

### dekódování instrukce (Instruction Decode – ID)

- instrukční kód je řadičem dekódován a jsou generovány řídicí signály (zejména pro ALU),
- do PC se připraví nová hodnota;

### načtení operandů (Operand Fetch – OF), bývá součástí ID

- IR připraví na AB adresu operandu v paměti M,
- M vystaví hodnotu operandu na DB (lze načíst do ACC);



45

## Instrukční cyklus (pokračování)

### vykonání operace (Execution – EX)

- podle signálů z IR je v ALU vykonána operace (ACC může být jak zdrojovým, tak cílovým registrem);

### obsluha paměti (Memory operation – MEM), příp. obsluha přerušení

- nemusí být součástí instrukčního cyklu;

### zápis výsledku (Write Back – WB), také Result Store

- IR vystaví na AB adresu M, kam má být zapsán výsledek (výsledek je na DB, může být přechodně v ACC);

Některé instrukce nevyužijí všechny fáze, příp. mohou mít jiné:

např. procesory CISC: IF-ID-AG-CA-EX-WB  
(AG – Address Generation, CA – Cache Access, jeden operand v paměti)

46

## Kombinační (jednotaktový) CPU

Model procesoru, který může uskutečnit instrukční cyklus

Model rozděluje paměť na datovou a programovou část

Současně provádí:

- adresaci zápisníkové paměti (pro čtení i zápis)
- nastavení sběrnice datových zdrojů
- vybuzení obvodů hlavní paměti (adresa, data, čtení, zápis)
- nastavení sběrnice zpětného zápisu
- přípravu nové adresy čítače

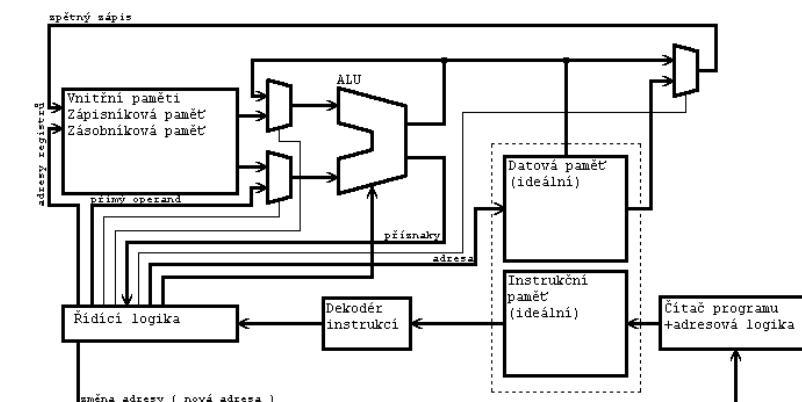
Doba vykonávání jedné instrukce je rovna součtu dob ustálení jednotlivých kombinačních obvodů ( $CPI = 1$ ) :

$$T_{clk} = T_{set}^{IF} + T_{set}^{ID} + T_{set}^{EX} + T_{set}^{MEM} + T_{set}^{WB}$$

47

## Kombinační (jednotaktový) CPU

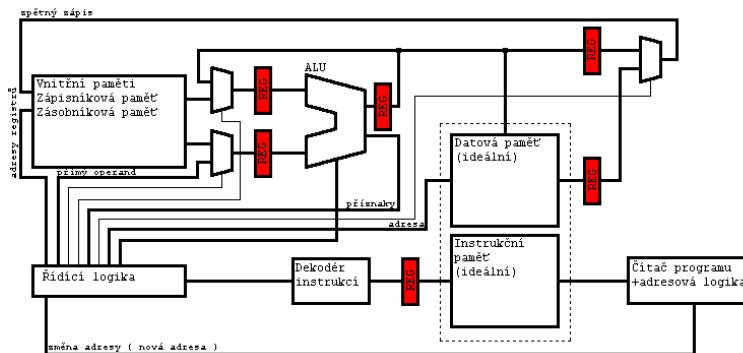
Jednoduchý řadič, ale dlouhá doba periody instrukčního cyklu



48

## Vícetaktový procesor

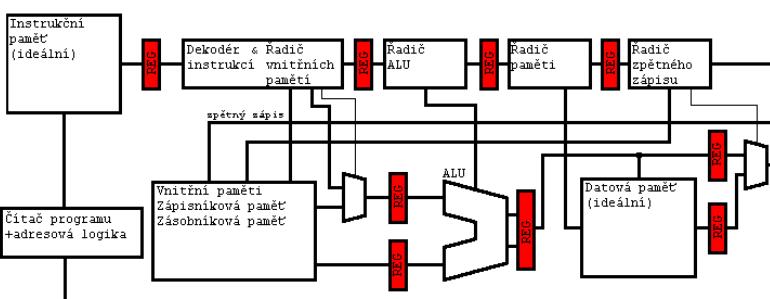
Vložení dodatečných registrů k oddělení jednotlivých fází instrukčního cyklu



49

## Zřetězený procesor

Řadič rozdělíme na menší řadiče jednotlivých fází instrukčního cyklu a mezi jednotlivé části vložíme registry



51

## Vícetaktový procesor

Rozdělení pomocí registrů umožnuje kontrolovat každou část procesoru odděleně a v důsledku toho některé fáze instrukčního cyklu vynechat

$CPI$  vzroste méněkrát než kolikrát lze zvýšit  $T_{clk}$   
 $\Rightarrow$  výkon je nepatrně vyšší než u jednotaktového CPU ( $\sim 1,2x$ )

Vícetaktový procesor je univerzálnější, protože umožňuje implementovat i komplexní funkce např. pomocí mikroprogramování

Jednotaktový i vícetaktový CPU jsou **plně sekvenční**  
(instrukce je dokončena než je započata následující)

Zvýšení výkonu sekvenčních CPU lze dosáhnout **zřetězením** nebo **replikací** (opakováním) funkčních jednotek

50

## Zřetězený procesor

Tím mluvíme o zřetězené struktuře – **pipeline** a umožníme **proudové zpracování instrukcí (pipelining)**

Oddělením jednotlivých stupňů zvýšíme vytížení jednotlivých komponent CPU (nesníží se čas potřebný k vykonávání jedné instrukce, ani nesnížíme dobu ustalování KLO)

Nejvyšší vytížení je možné získat trvalým přísunem nových instrukcí  
 $\Rightarrow$  nejlépe instrukcí s jednotnou délkou

Není možné zpracovávat komplexní instrukce (musí se rozložit na sekvenci jednodušších)

$T_{clk}$  je dána dobou ustálení nejpomalejšího KLO v CPU:

$$T_{clk} = \max\{T_{set}^{IF}, T_{set}^{ID}, T_{set}^{EX}, T_{set}^{MEM}, T_{set}^{WB}\}$$

52

## Zřetězené zpracování informací

Typy zřetězení: zřetězení aritmetické (na úrovni ALU), zřetězení instrukcí (na úrovni CPU), zřetězení procesů (na úrovni počítače).

Předpoklady zřetězení:

- nepřetržitý přísun údajů,
- operaci lze rozdělit na sekvenci nezávislých kroků (fází),
- trvání jednotlivých kroků by mělo být přibližně stejné.

Počet stupňů v řetězu označujeme jako **hloubka zřetězení**.

Doba průchodu instrukce všemi stupni řetězu je tzv. **latence řetězu (pipeline)**, tj. hloubka násobená dobou taktu.

Zřetězené zpracování může být **asynchronní** nebo **synchronní**.

53

## Zřetězené (proudové) zpracování

Procesory jsou vesměs synchronní, řízeny hodinovým signálem.

Procesor s jednoduchými fázemi, ale s větším počtem fází, může operovat na vyšší frekvenci.

Procesory s velkým počtem fází (2 až 20) častěji vznikají kvůli různým závislostem prázdná místa (tzv. bubliny), které snižují výkon;

⇒ vyšší frekvence CPU nezaručuje větší výkonnost (nelze porovnávat CPU různých architektur podle frekvence);

⇒ dlouhá pipeline s vysokou frekvencí může být neefektivní z hlediska spotřeby (Pentium 4 mělo  $P_{tot}$  až 100W a hloubku zřetězení 22 stupňů).

Pipeline zavádí v CPU určitý paralelismus zpracování instrukcí, ale navenek se CPU tváří sekvenčně => vznikají hazardy.

54

## Hazardy zřetězeného zpracování

**Hazardy** – narušení vykonávání programu (nezaměňovat s hazardy v číslicových obvodech):

- **strukturní hazardy** – kolize sdílených (omezených) prostředků (paměť, registry, funkční jednotky, sběrnice) – současný přístup k prostředku z více stupňů pipeline;
- **datové hazardy** – důsledek datových závislostí v programu (instrukce čeká na výsledek předchozí instrukce, nemůže uložit data do neuvolněného registru apod.).  
Hazardy: RAW, WAW a WAR (viz dále).
- **řídicí hazardy** – vznikají při provádění řídicích instrukcí
  - nutno činit rozhodnutí před vykonáním instrukce (např. podmíněně skoky – statisticky velmi časté (1/6))
  - čím větší je hloubka zřetězení, tím se zvyšují ztráty.

55

## Hazardy zřetězeného zpracování

Datové závislosti:

RAW (Read after Write) – výstup instrukce je použit jako vstup instrukce následující;

WAW (Write after Write) – dvě instrukce zapisují na stejně místo;

WAR (Write after Read) – zatímco jedna instrukce zpracovává data, další instrukce tato data změní.

Řešení hazardů:

- **Statické** (ve fázi komilace) – přeskládáním instrukcí (provádění „nezávislých“ instrukcí mimo pořadí) nebo vkládání prázdných instrukcí NOP;

- **Dynamické** (za běhu programu) – vkládání prázdných taktů (bublin) řadičem – pozastavení pipeline („pipeline stall“).

Výkonnost pipeline závisí na počtu fází a četnosti pozastavování.

56

## Příklad

Určete CPI proudově pracujícího procesoru, kde je dána četnost hazardů a způsobené pozastavení:

- RAW hazardy nastanou v 10 % instrukcí, průměrné pozastavení 2 taky;
- WAW hazardy nastanou v 0,5 % instrukcí, průměrné pozastavení 4 taky;
- řídicí hazardy v důsledku provedených skoků nastanou v 9 % instrukcí, průměrné zdržení jsou 3 taky;
- strukturální hazardy v důsledku nedostupné jednotky či registrového zápisového portu nastanou v 3 % instrukcí, průměrné zdržení jsou 2 taky.

Vycházíme z předpokladu, že pro ideální proudově pracující procesor platí: CPI = 1 (1 instrukce za jeden strojový takt).

57

## CPU a instrukční cyklus

Vývoj procesorů lze rozdělit do 3 fází:

- 1) **Subskalární (sekvenční) procesory** - tradiční sekvenční provádění instrukcí (celková doba provádění programu je dána aritmetickým součtem časů trvání jednotlivých instrukcí).
- 2) **Skalární procesory** – sekvenční vykonávání nahrazeno paralelním (buď zřetězené zpracování nebo replikace);
  - v jednom taktu byla k vykonání v některé jednotce dána nejvýš jedna instrukce ( $IPC = 1$ , Instruction Per Cycle)
  - vykonávání instrukcí probíhá překrytí (i paralelně);
  - dalším vývojem paralelismus navýšen i použitím několika zřetězených funkčních jednotek.

58

## CPU a instrukční cyklus

- 3) **Superskalární procesory** – paralelní vydávání i paralelní zpracovávání instrukcí ( $IPC > 1$ ); dosahuje se duplicitou vybraných částí procesoru (ne více procesorových jader); problém je se závislostí jednotlivých instrukcí.

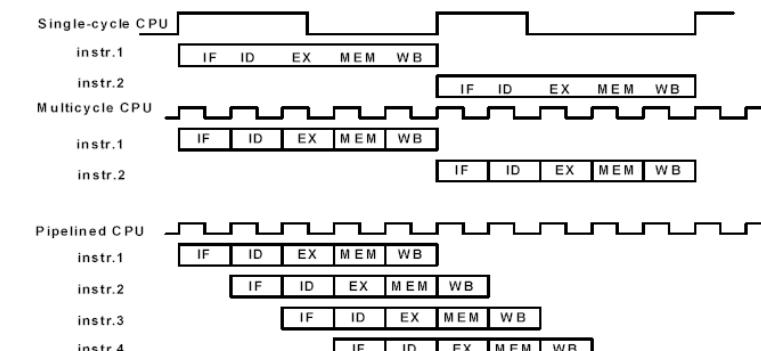
Plánování paralelního zpracovávání rozdělujeme na:

- statické** – paralelní vydávání instrukcí naplánovaný komplátorem (např. architektura VLIW); někdy se neřadí mezi superskalární architektury;
- dynamické** – o paralelismu rozhodují technické prostředky za běhu programu (složitější, ale zachovává se binární kompatibilita);

Podobné trendy paralelizace jsou uplatňovány v architekturách počítačů na různých úrovních abstrakce (mikrořadič, procesor, počítačový systém), v architekturách RISC i CISC.

59

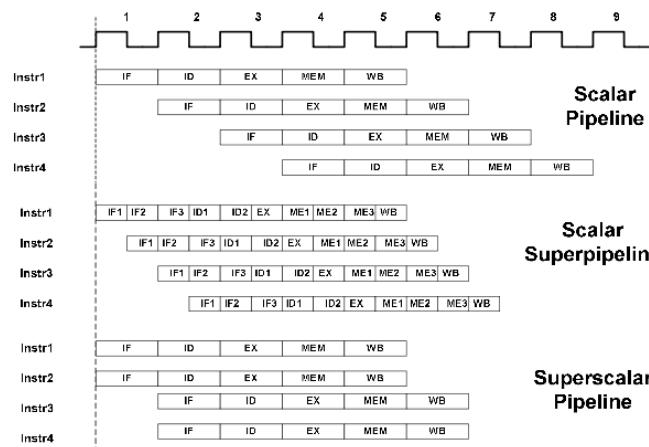
## Porovnání typů procesorů



rozdelení i počet jednotlivých fází instrukčního cyklu je u různých procesorů různé

60

## Zřetězené procesory



61

## Zřetězené zpracování informace

Frekvence výstupu  $k$ -stupňového řetězce (počet výsledků za jednotku času):

$$(t_d \dots \text{zpoždění ve vyrovnávacím registru}) \quad f = \frac{1}{\max(t_i)^k + t_d}$$

Časový interval na vykonání  $k$ -stupňového řetězce  $n$  úloh:

$$T_k = [k + (n - 1)](\tau + t_d)$$

$k(\tau + t_d) \dots$  čas na zaplnění řetězce (kompletní vykonání první úlohy)  
 $(n-1)(\tau + t_d) \dots$  čas na vykonání zbývajících  $(n-1)$  úloh  
 $\tau \dots$  max. zpoždění stupně (bez oddělovacího registru)

Čas na vykonání nezřetězené úlohy:  $T_s = k.n.\tau$

62

## Zrychlení při zřetězení

Zrychlení výpočtu  $n$  úloh ve zřetězeném a nezřetězeném procesoru:

$$S_k = \frac{T_s}{T_k} = \frac{n.k.\tau}{[k + (n - 1)](\tau + t_d)}$$

Teoretické maximální zrychlení pro velmi velká  $n$  se blíží  $k$ , tzn. že lineárně zřetězeným procesorem je možno dosáhnout zrychlení blížící se až počtu stupňů tohoto procesoru (pro  $n >> k$  platí  $[k + (n - 1)] \rightarrow n$ , tedy  $CPI \rightarrow 1$  pro  $n \rightarrow \infty$ , a pro  $t_d \ll \tau$  platí  $S_k = k$ ).

Zvyšování počtu stupňů  $k$  mírně zvětšuje zpoždění vstup-výstup (vlivem přídavných registrů), a tím také cenu (větší plocha čipu). Obecně přináší zřetězení velké zrychlení bez velkých nároků na přídavný HW.

63

## Zrychlení při superzřetězení

Superzřetězení (superpipelining) – jednotlivé fáze zpracování instrukce (IF, ID, ...) jsou dále děleny na určitý počet  $m$  kratších taktů (hloubka zřetězení se tak zvyšuje na řadově desítky i více) => zvýšení využití logiky a možnost zvyšování  $T_{clk}$

Zrychlení vůči normálnímu zřetězení:

$$S_{mk} = \frac{T_k}{T_{mk}} = \frac{(k + n - 1).(\tau + t_d)}{(m.k + n - 1).(\tau / m + t_d)}$$

Pro  $n \rightarrow \infty$  a pro  $t_d \ll \tau$  je  $S_{mk} = m$

Je-li zpoždění registrů srovnatelné se zpožděním stupně

$$\tau/m \approx t_d \text{ a pro } n \rightarrow \infty, \text{ pak } S_{mk} = (m+1)/2$$

64

## Zřetězení, superzřetězení

Zřetězená linka nebývá plně využita => průměrné využití stupňů linky  
(dosažená účinnost):

$$E_k = S_k / k$$

Výsledky úkolu budou k dispozici u nezřetězeného zpracování  
v intervalech  $t$ , zatímco u zřetězené linky v intervalech:

$$T = \tau + t_d = t / k + t_d = T_{clk}$$

Doba zpracování 1 instrukce je (v ideálním případě):

- pro zřetězené zpracování:  $CPI.[t / k + t_d]$
- pro superzřetězené zpracování:  $CPI.[t / k.m + t_d]$
- pro superskalární zpracování:  $CPI.[t / k + t_d] / IPC$
- pro superzřetězené i superskalární:  $CPI.[t / k.m + t_d] / IPC$

65

## Superzřetězení vs. superskalárnost

Trendem je kombinace obou technik – nejrychlejší.

Při rovnosti  $IPC = m$  vychází superskalární řešení rychlejší než  
superzřetězené.

Počet tranzistorů na čipu se zvyšuje rychleji než jejich rychlosť  
=> od paralelismu na úrovni bitů (8 až 64bitové CPU)  
přecházíme k paralelismu na úrovni instrukcí ( $IPC$  se  
zvyšuje od 1 do 8) a dále k paralelismu na úrovni vláken  
a procesů.

Zvyšování  $IPC$  naráží na paralelizovatelnost algoritmů.

Při zvyšování  $m$  se výkon zvyšuje teoreticky lineárně (při  $t_d \ll \tau$ ), ale  
jsme omezeni technologií (např. 2x delší pipeline neumožňuje 2x  
zvýšit frekvenci), rostou hazardy (režie) a ztrátový výkon.

66

## Příklady

1) Jaká je dosažená účinnost zřetězené 4stupňové linky při zpracování  
6 instrukcí ( $t_d \ll \tau$ ), je-li diagram vytížení dán tabulkou  
(„x“ značí čekání na mezivýsledek, „–“ značí nevyužitost) :

Čas	1	2	3	4	5	6	7	8	9	10
S1	i1	i2	i3	i4	x	i5	i6			
S2		i1	i2	i3	x	i4	i5	i6		
S3			i1	i2	x	i3	i4	i5	i6	
S4				i1	i2	–	i3	i4	i5	i6

2) Porovnejte zrychlení dvou ideálních instrukčních linek:  
a) superzřetězené linky se 4 stupni, každý stupeň se 3 podstupni,  
b) superskalární zřetězené linky se 4 stupni s četností vydávání  
3 instrukcí/takt,  
proti nezřetězenému zpracování  $t = 100$  ns na jednu instrukci.  
Uvažujme počet instrukcí  $IC \rightarrow \infty$ , zpoždění registrů  $t_d = 5$  ns.

67

## Zrychlení při ztrátových cyklech

Předpokládejme, že  $S_k = k$  (pro velký počet instrukcí).

Pokud některé instrukce nelze zřetězit, dostáváme obdobu  
Amdahlova zákona:

$$S_k = \frac{1}{(1-g) + g/k} \quad \begin{array}{l} k \dots \text{počet stupňů řetězce;} \\ g \dots \text{část instrukcí, které lze řetězit.} \end{array}$$

### Příklad:

Máme 6stupnovou zřetězenou linku. Jaké bude zrychlení linky, když  
13 % instrukcí bude mít 4 ztrátové (čekací) cykly, 25 % instrukcí  
bude mít jeden ztrátový cyklus a zbývající počet instrukcí bude  
bez ztrátových cyklů?

68

## Fronta instrukcí

Jedna z nejstarších technik urychljení procesorů.

Před instrukční registr (za interní sběrnici) se vložila **fronta instrukcí**.

Různé instrukce trvají různou dobu - složitější instrukce trvají delší dobu a mezi nimi se můžou do fronty načítat další instrukce; naopak u krátkých instrukcí (inkrementace, porovnávání, součet registrů) se naopak nemusí čekat na vybavení instrukce z relativně pomalé paměti.

Při změně běhu programu (příchod přerušení, provedení instrukce skoku apod.) se musí fronta vyprázdnit (zahodit).

69

## Zpracování instrukcí mimo pořadí

*Out of Order* - metoda, při níž se instrukce vykonávají v jiném pořadí, než jak uvádí program uložený v operační paměti. Procesor si sám rozhodne (jeho řadič, resp. dekódér instrukcí) a poskládá instrukce tak, aby byly zpracovány v co nejkratším čase při maximálním využití všech částí procesoru. Toto je důsledek toho, že ne všechny instrukce se dají zpracovávat současně a ne všechny se dají stejně rozpracovat. Kdybychom zpracovávali instrukce v původním pořadí, byly by některé části mikroprocesoru po určitou dobu nevyužity.

Někdy se procesory s OoO značí jako MLP (Memory Level Parallelism). V dnešní době ji využívá většina procesorů (někdy se OoO z CPU odstraňuje z důvodu snížení spotřeby energie).

70

## Spekulativní vykonávání instrukcí

*Speculative execution* – spočívá v odhadu vykonávání instrukcí dopředu. Instrukce se vykonávají v době, kdy je procesor méně vytížen, a jednotlivé výsledky zpracování se ukládají v CPU, aby byly v okamžiku potřeby k dispozici (někdy se nevyužijí a zahodí se).

Problém skokových instrukcí (5–20 % podle typu programu)

- zhruba v 5/6 případů se skok provede,
- nečeká se na vyhodnocení podmínky,
- v případě nesprávné predikce se výsledky zahodí,
- v predikci mohou být další podmíněné skoky,
- => úroveň (stupeň) spekulace je omezena, aby „úklid“ po špatné predikci netrval příliš dlouho.

Některé CPU vykonávají obě větve při podmínce s tím, že jednu z nich po vyhodnocení podmínky zahodí.

71

## Přejmenovávání registrů

*Register renaming* - technika omezující nebo odstraňující závislosti následujících instrukcí, které pracují se stejnými registry procesoru. Během zpracování instrukcí je k dispozici sada dočasných registrů, které zastupují skutečné registry procesoru.

(Používají-li například dvě následující instrukce stejný registr, je pro každou z nich přidělen dočasný registr s kopí původního registru. Když druhá instrukce změní hodnotu registru, je původní hodnota stále ještě v dočasném registru pro potřeby první instrukce, takže druhá instrukce může být vykonávána mimo pořadí před první instrukcí.)

72



## Multi-threading (MT)

Skupina technologií umožňujících zvýšit počet vláken v jednom procesoru.

*Vertikální multi-threading (VMT)* – zvýšení využití procesoru zmenšením čekacích cyklů – pokud CPU na něco čeká, přepne se jiné vlákno; CPU zpracovává v jednom okamžiku jediné vlákno (skalárnost z pohledu vláken); různé varianty podle rychlosti přepnutí (počtu taktů);

*Horizontal multi-threading (HMT)* – lepší vytízení/využití funkčních jednotek (současně se vykonává více než jedno vlákno – zpravidla dvě, někdy až 8) – superskalárnost z pohledu vláken; opět různé varianty (**hyper-threading**, simultaneous MT – do pipeline se zavádějí instrukce různých vláken, které na sobě nebudou závislé).

73

## Hyper-threading (HT)

Přináší dva programové vstupy, takže je možné najednou zpracovávat dva toky instrukcí (jeden fyzický procesor se tváří jako dva procesory logické) – např. 4jádrový CPU má 8 logických procesorů.

Ve skutečnosti zdvojeno to, co určuje aktuální stav jednoho výpočetního procesu (registry, instrukční čítač); dodatečné prostředky zabírají asi jen 5% plochy čipu (ALU i L2 cache jsou sdílené).

Díky HT jsou výkonné jednotky procesoru vytěžovány větší měrou – optimálně lze využít u vícevláknových úloh (výkonový zisk až 30%), náročnější správa.

HT byl prvotně implementován u jednoprocесорových systémů (od Pentia 4), u dvoujádrových architektur vynechána, u vícejádrových CPU opět implementována (u některých řad).

74

# Mikrořadiče

**Milan Kolář**  
*Ústav mechatroniky a technické informatiky*



Projekt ESF CZ.1.07/2.2.00/28.0050  
Modernizace didaktických metod  
a inovace výuky technických předmětů.

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

## Mikropočítač ( $\mu$ P)

Obecný (PC, PDA, server, superpočítač)

- samostatný (sám o sobě) nebo oddělený,
- důraz na HMI (Human-Machine Interface), tj. rozhraní člověk-stroj nebo na komunikační připojení
- provádí operace obecného účelu.

Embedded (vestavný, zabudovaný, vložený, řídící)  
do systému (zařízení, stroje, přístroje), jednoúčelový?

- dedikovaný počítač pracující v reálném čase,
- důraz na M2M (Machine-to-Machine), tj. rozhraní k obsluhovanému objektu (snímače, akční členy),
- pro uživatele není  $\mu$ P přímo viditelný,
- obsahuje často rozšířené možnosti úspory energie,
- více specializován na operace zaměřené na konkrétní aplikace,
- program se většinou vyvíjí na jiném systému.

2

## Mikroprocesor

Jednoobvodový (jednočipový) procesor, vyráběný technologií velké integrace

- univerzální

- do osobních počítačů, serverů, pracovních stanic
  - spíše CISC, větší šířka sběrnic (32/64bit.), velké taktovací frekvence (~ GHz), velký příkon;
- do embedded mikropočítačů
  - spíše RISC, šířka sběrnic 8–32 bitů, integr. periferie, frekvence (stovky MHz), často bateriové napájení;

- signálový (DSP – Digital Signal Processor)

- pro zpracování signálů (filtry, FFT, modulace),
- speciální architektury i instrukce;

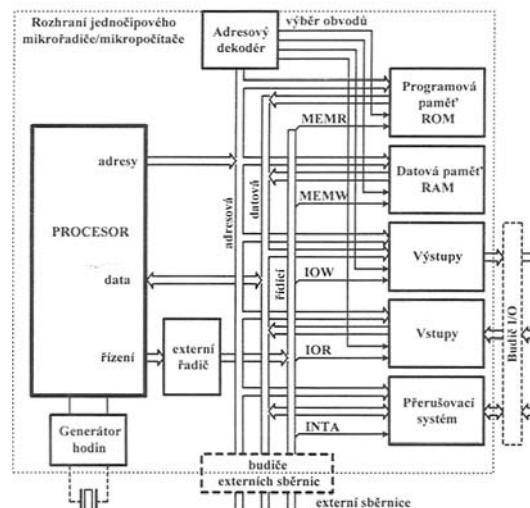
- grafické, multimedialní, ...

## Mikrořadič (Microcontroller – $\mu$ C)

jednoobvodový (jednočipový, monolitický)  $\mu$ P (mikroprocesor doplněný pamětími a periferiemi), MCU – Micro Controller Unit;

- převážně RISC architektura – jednocyklové instrukce, omezený soubor instrukcí (35–130);
- spíše Harvardská architektura;
- programová paměť – varianty Flash, EEPROM, omezené ROM nebo OTP (One-Time Programmable), až 512 kB;
- datová paměť – SRAM, Flash, EEPROM, externí SDRAM, až 128 kB;
- datová sběrnice 8, 16, 32 bitů (výjimečně 64 bitů);
- relativně nízká cena;
- široký rozsah napájecího napětí (od 1,8 V do 3,3 V);
- technologie CMOS;
- střadačová nebo registrová ISA (příp. kombinace).

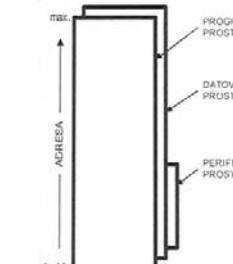
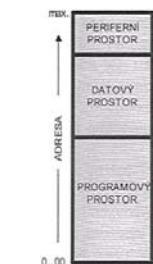
## Blokové schéma mikrořadiče



5

## Paměťový prostor

- rozsah adres: u 8bit.  $\mu$ C ~ 12–16 b; u 16bit. ~ 16–20 b; u 32bit. ~ 32 b
- vnitřní nebo vnější (jsou-li vyvedeny sběrnice),
- společný (lineární, typické pro Von Neumannovu architekturu) nebo vícenásobný (oddělený, typické pro Harvardskou architekturu; mohou existovat stejné adresy; přístup je odlišen řídicími signály),
- u jednodušších  $\mu$ P jsou do datové paměti mapovány všeobecné registry.



6

## Vnitřní části architektury

**Časovače / čítače** – programovatelné děliče / čítače (vstupem je nejčastěji signál z oscilátoru); slouží nejčastěji k odměřování časových událostí; přetečením se vyvolá obvykle přerušení.

**Watchdog Timer** – sleduje správný běh programu (nutno jej periodicky nastavovat / nulovat, aby nepřetekl) a zajíšťuje reset mikrořadiče při „zběhnutí“ programu; bývá většinou interní (výjimečně externí);

**Porty (vstupy a výstupy)** – jsou obecně organizovány do paralelních 8bitových bran (některá z bran může být neúplná); většinou bývají obousměrné (směr je třeba programově nastavit); některé porty mohou být sériové, příp. analogové.

**Zdroj hodinového signálu** – většinou externí oscilátor nebo alespoň externí RC článek nebo krystal (oscilátor interní);

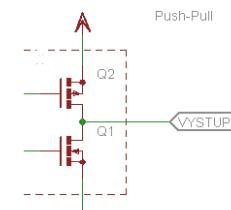
**Přerušovací systém** – reaguje na asynchron. události v běhu programu, menší počet úrovní priorit, většinou vnitřní i vnější přerušení.

7

## Výstupy mikrořadičů

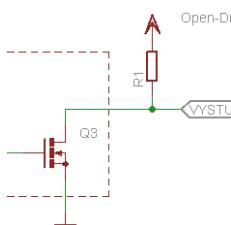
### Dvojčinný výstupní budič třístavový (Push-Pull)

- drží log. úroveň 0 nebo 1,
- vysoká rychlosť přeběhu,
- rychlé nabíjení/vybíjení parazitních kapacit,
- mohou se nastavit do „vysoké impedance“.



### Jednočinný výstupní budič s otevřeným kolektorem (Open-Drain)

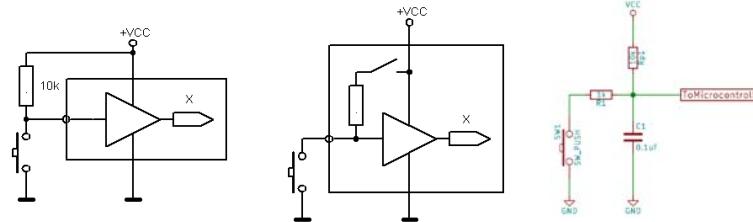
- proud pouze odebírá,
- potřeba rezistor (interní nebo externí),
- možnost připojit na jiné napětí než je napětí čipu (např. jiný napěťový standard),
- možnost vytvářet montážní součiny.



8

## Vstupy mikrořadičů

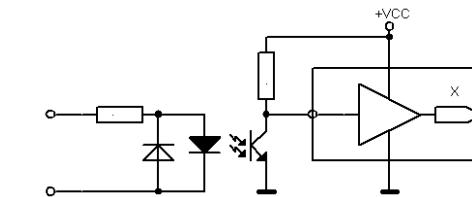
- s pull-up rezistorem (vnější nebo vnitřní)
- s hysterezí (Schmittovy)
- s pull-down rezistorem (příliš se nepoužívá)
- galvanicky oddělený (s otronem)
- nezapojen (NC – no connect), třetí stav



9

## Galvanicky oddělený vstup

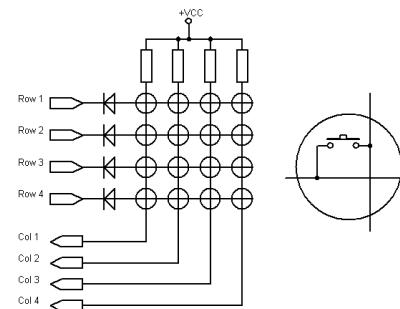
- pro logické (číslicové) signály,
- chrání vstupy mikrořadiče před poškozením,
- může pracovat na jiném potenciálu (rozmezí 4–30 V),
- je možno navrhnut i pro buzení střídavým napětím.



10

## Maticová klávesnice

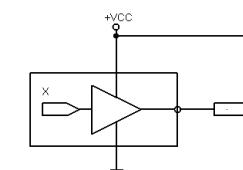
Při pripojení více kontaktů – nutno programově scanovat (multiplexovat) – postupně nastavovat řadu po řadě do log.0 (ostatní jsou v log.1) a číst stavu na sloupcích (diody zabraňují zkratu při stisku dvou tlačítek v jednom sloupci).



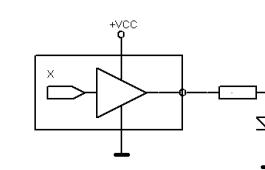
11

## Připojení LED

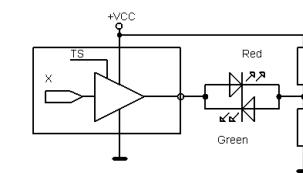
Obdobně je možné zapojit otrony, optotriaky apod.



svítí při log.0



svítí při log.1



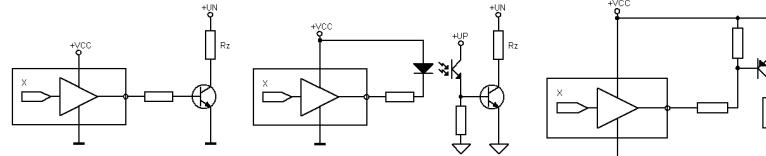
dvojitá LED – při log.0 svítí zelená, při log.1 červená, ve třetím stavu žádná

12

## Připojení větších zátěží

Zátěž vyžadující vyšší proudy nebo napětí, např. relé, stykač, reproduktor, elektromagnet, motorek, odporové opení, žárovka, ...

- tranzistorové spínače (možné i Darlingtonovo zapojení)
- použití otronů (omezení rušení, ochrana µC)
- možnost napájení zátěže z napájecího zdroje µC



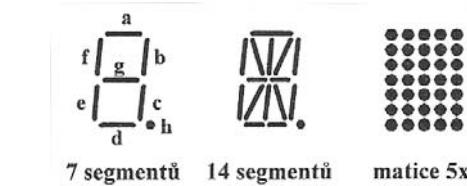
13

## Připojení displeje

Jednotlivé segmenty spínáme přímo nebo přes budiče

Většinou se snažíme redukovat počet vývodů

- BCD dekodér (nelze zobrazit libovolné znaky),
- komunikace přes sériové rozhraní,
- multiplexní řízení.



7 segmentů

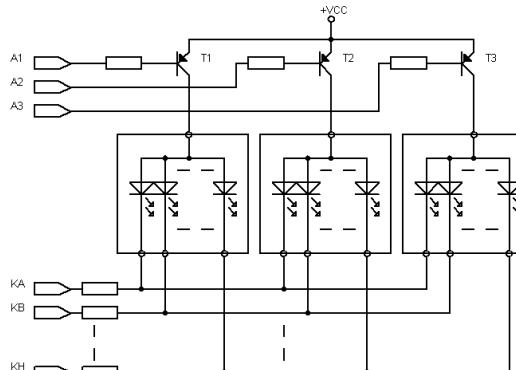
14 segmentů

matice 5x7

14

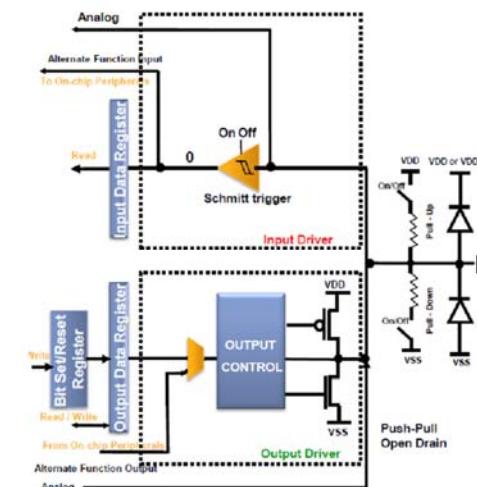
## Multiplexní řízení displeje

Cyklicky rozsvěcime jednotlivé znaky



15

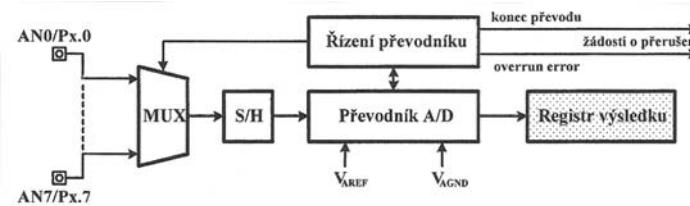
## Příklad obousměrného pinu



16

## A/D převodníky (ADC)

- rozlišení 8, 10, výjimečně 12 bitů,
- většinou na principu postupné approximace,
- rychlosti převodu relativně malé (řádově 10<sup>2</sup> kS/s),
- vstupní analogový multiplexer (6-16 kanálů),
- vzorkovací obvod S/H (Sample/Hold),
- referenční napětí (napájecí nebo externí).



17

## Další periferie

**Analogový komparátor** – 2 vstupy přivedeny na interní OZ, výstup čten v SFR, příp. aktivuje přerušení nebo zaveden na výstup  
- s vnějšími obvody umožňuje realizovat ADC s dvojitou integrací.

**D/A převodník (DAC)** – až u novějších typů (dříve jen PWM s dolní propustí nebo externí)  
- nejčastěji 8, 10 nebo 12 bitů.

**Obvod reálného času (RTC – Real Time Circuit)** – obvod se záložním bateriovým napájením a krystalem (32768 Hz)  
- z jednotlivých registrů se čtou časové údaje (sec až roky).

**Vstupy pro zachycování času a počítání událostí (capture unit).**

LCD řadiče, JTAG, SPI, I<sup>2</sup>C, UART, USART, PWM, CAN, Ethernet, DMA řadič, PLL, ...

18

## Přehled mikrořadičů

Mikroprocesory pro embedded systémy lze dělit na 2 kategorie:

- **nezávislé procesorové rodiny** (produkovaný více než jedním výrobcem) – liší se zejména množstvím periferií, velikostí interní paměti, frekvencí;  
– např. řada Intel 8051, řada ARM, řada MIPS aj.
- **procesorové rodiny jednoho výrobce** – závislost na jediném výrobci, většinou mají určitou unikátní vlastnost;  
– řada PIC (Microchip), řada MSP430 (TI), řada Freescale, řada Renesas aj.

19

## Mikrořadiče a Internet věcí

Množství populárních platform pro vývoj mikrokontrolérů aplikací:

**Arduino** – jednoduchá platforma (převážně 8bit CPU Atmel, varianty i s CPU ARM 32 bitů), množství variant a rozšiřujících karet;

**Raspberry PI** – výkonné vícejádrové procesory, velká modularita, podpora OS Linux a Android;

**Orange PI** – open-source jednodeskové počítače, výkonné vícejádrové procesory ARM Cortex-A7;

**WeMos** – množství malých modulů s procesory, obdoba Arduina, integrovaná WiFi, velká modularita;

**Espressif** – platforma kompatibilní s Arduino, výkonnější procesory, komunikační moduly (nejen WiFi);

**BBC micro:bit** – jednoduchá platforma pro začátečníky v programování (32bit ARM), bluetooth rozhraní.

20

## Řada ARM

vyvinuté firmou ARM Limited (Advanced RISC Machine)

- postupně verze ARM1 (1984) až ARM11, pak Cortex;
- označování ARM ISA (jádro): ARMv1 ... ARMv8 (nepřehledné);
- 32bitová architektura RISC, objevují se již 64bitové verze;
- výborný poměr výkon/spotřeba (pro bateriové aplikace);
- nyní Harvardská architektura (do verze ARM7 s ARMv4 spíše von Neumann – společný adresní prostor, 1 sběrnice);
- periferie: Ethernet, ADC, UART, CAN, SPI, I2C, PWM, ...
- v rámci licence mnoho výrobců (Luminary Micro, ST, Atmel, NXP, ...)
- v současnosti nejpočetnější skupina procesorů (desítky mld. CPU)
  - řada Cortex-A (vysoká výkonnost, práce pod OS)
  - řada Cortex-R (práce v reálném čase, rychlá odezva)
  - řada Cortex-M (mikrokontroléry s nízkou spotřebou)

21

## Řada ARM Cortex-Mx

založena na ISA ARMv7 (novější verze ARMv8), určeno především pro DSP embedded aplikace (mikrořadiče)  
 různé varianty: M3, M4, M7, M23, M33, M35P, M55 (2020)  
 32bitové varianty, většinou Harvardská architektura

ARM Cortex-M7:

- výkon 2,14 DMIPS/MHz; max. frekvence 216 MHz (462 DMIPS);
- programová Flash 512 kB, datová SRAM 256 kB (Harvard);
- duální 6úrovňová pipeline (superskalární CPU);
- až 5 paralelních modulů pro vykonávání instrukcí (execute) – ALU1, ALU2, MAC, FPU, Load/Store;

22

## Řada ARM Cortex-Axx

založena na ISA ARMv8-A, 32/64bitové, pro vysoce výkonné aplikace

(netbooky, mobilní telefony, multimédia); patří mezi MCU?

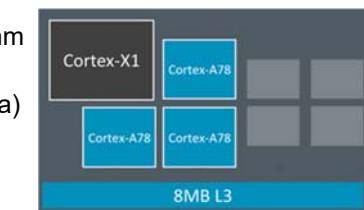
- různé varianty – A35, A53, A57, A72, A73, A75, A76, A77, A78;
- i více CPU jader na čipu (1 – 4);
- 31 vnitřních 64bitových registrů;
- L1 (D-cache, I-cache), L2 společná pro jádro, příp. L3;
- frekvence od 0,5 – 3,0 GHz;
- výkon 1,6 – 2,5 DMIPS/MHz na 1 jádro;
- spotřeba 0,12 až 0,08 mW/MHz;
- energetická efektivita 5 – 20 DMIPS/mW;
- pipeline 8 až 18úrovňová;
- technologie 28 nm → 20 nm → 14 nm → 10 nm.
- podpora množství OS (Linux, Symbian, Windows CE).

23

## Nejnovější ARM Cortex-X1

Představena na jaře 2020, zatím nejvýkonnější verze

- vychází z A78 (ale zvýšení výkonu cca 20 %)
- ISA ARMv8.2, 64bitová verze
- frekvence 3,0 GHz
- 64 kB L1, až 1 MB L3, až 8 MB L3
- 13stupňová pipeline
- superskalární (5 instrukcí/takt)
- technologie 10 nm → 7 nm → 5 nm
- předpokládá se integrace s jinými jádry (big.little architektura) zejména pro mobilní telefony.



24

# Paměti

Milan Kolář  
Ústav mechatroniky a technické informatiky



Projekt ESF CZ.1.07/2.2.00/28.0050  
Modernizace didaktických metod  
a inovace výuky technických předmětů.

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

2

## Pojmy a dělení

**Paměť** – zařízení, které slouží pro uchování informací (konkrétně binárně kódovaných dat).

**Registr** – velmi rychlé paměťové místo malé kapacity (jednotky bytů) umístěné většinou uvnitř procesoru počítače.

- **Neadresovatelné paměti** (LIFO, FIFO) – nezadává se u nich, kam zapisovat či odkud číst.
- **Adresovatelné paměti** (RAM, ROM) – paměť bývá rozdělena na **buňky** určité velikosti, z nichž každá je jednoznačně identifikována svým číslem. Toto číslo se nazývá **adresa** paměti a velikost takového buňky, která má svou vlastní adresu, se označuje jako nejmenší adresovatelná jednotka.
- **Obsahem adresovatelné paměti** (CAM) – asociativní.

## Dělení pamětí

Podle materiálu a fyzikálních principů:

- feritové (zastarálé),
- magnetické (informaci uchovává směr magnetizace),
- magnetooptické (světlo mění magn. vlastnosti materiálu),
- polovodičové (nejpoužívanější),
- optické (založeny např. na odrazu světla).

Podle schopnosti uchovávat obsah při odpojení napájení (podle energetické závislosti):

- **volatilní** (nestálé) - ztratí obsah,
- **non-volatile** (stálé) - drží obsah.

## Dělení pamětí

**Paměť s přímým přístupem**: paměť, která dovoluje přistoupit okamžitě k místu s libovolnou adresou  
RAM (Random Access Memory);

**Paměť se sekvenčním přístupem**: paměť, u které je nutné při přístupu k místu s adresou  $n$  nejdříve postupně přečíst všechna předcházející místa (0 až  $n-1$ ) např. magnetické pásky;

Dělení pamětí z hlediska čtení a zápisu:

- **RWM** (Read Write Memory) - RAM: paměť určená ke čtení i zápisu dat;
- **ROM** (Read Only Memory): paměť určená pouze ke čtení dat.

## Typy RAM

**DRAM** (Dynamic RAM) – dynamická RAM (nejběžnější);

**SDRAM** (Synchronous DRAM) – rychlé paměti synchronizované systémovými hodinami;

**SRAM** (Static RAM) – statická paměť založená na KO (varianty synchronní a asynch. se systémovými hodinami);

**FPM RAM** (Fast Page Mode RAM) – logika paměti předpokládá, že další požadavek na čtení bude ležet v sousedství;

**EDO ROM** (Extended Data Out RAM) – vylepšení FPM;

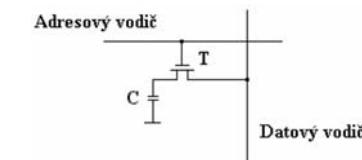
**RDRAM** (Rambus DRAM) – vylepšení od firmy Rambus.

## DRAM

Dynamická RAM – tvořena tranzistorem s velmi malým kondenzátorem (řádově femtofarady)

- buňka zabírá malou plochu,
- buňky je třeba každé cca 4 ms obnovovat (refresh).

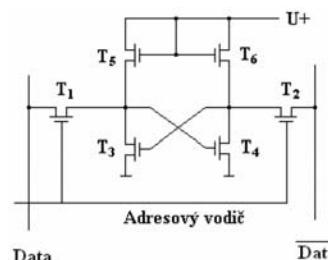
Čtení/zápis probíhá tak, že se přečte a vymaže celý řádek, uloží se do vyrovnávací paměti a pak se zapíše celý řádek z vyrovnávací paměti zpět do paměti.



## SRAM

Statická RAM – buňka tvořena bistabilním klopným obvodem, který je relativně složitý  $\Rightarrow$  menší kapacita, ale rychlé.

Při zápisu i čtení – na adresový vodič log.1, T1 a T2 se otevřou, zapisuje-li se log.0 (Data = 1), T4 se otevře, T3 zavře; čteme-li pak tuto hodnotu, Data za T2 jsou v log.0 (negace).



## Typy ROM

**ROM** – obsah paměťové buňky je určen při výrobním procesu (naprogramována výrobcem);

**PROM** (Programmable ROM) – programuje se v programátoru přepalováním propojek; někdy označována OTP (One Time Programmable);

**WORM** (Write Once – Read Many) – magneto-optický princip;

**EPROM** (Erasable PROM) – informace v podobě náboje hradla tranzistoru MOSFET; lze celé smazat (UV zářením) a opět naprogramovat (pouze omezený počet);

**EEPROM** (Electrically EEPROM) – může se el. impulsy po buňkách;

**Flash-EEPROM** – buňku tvoří 1 unipolární tranzistor se dvěma vzájemně izolovanými hradly (Control a Floating Gate), programuje se po blocích, levnější než EEPROM.

## Rozdělení pamětí

**Vnitřní (primární):** paměť sloužící pro uchování momentálně zpracovávaných dat a programů. Realizovaná většinou pomocí polovodičových součástek (RAM)

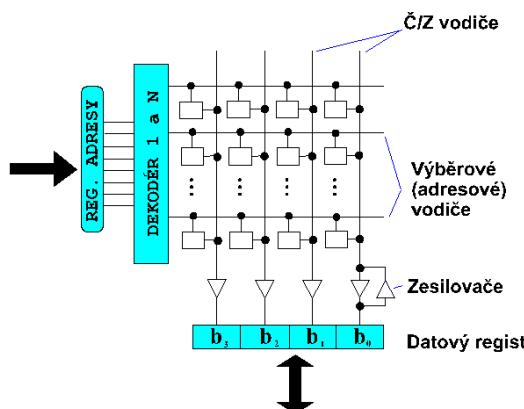
- registry,
- vyrovnávací paměti,
- hlavní paměť (operační).

**Vnější (periferní, sekundární):** paměť sloužící pro dlouhodobější uchování dat. Realizovaná většinou na principu magnetickém, optickém nebo Flash paměti. Ve srovnání s operační pamětí bývá přístup k jejím datům výrazně pomalejší.

**Archivní (záložní):** pro velké objemy dat, pomalé (mg. pásky).

## Struktura adresovatelné paměti

Paměťové buňky uspořádány do matice, napojeny na adresové a datové vodiče (sběrnice).



## Charakteristické parametry

**Vybavovací doba (access time)** – udává rychlosť s jakou paměť zapíše nebo vyhledá data (ns);

**Doba cyklu (cycle time)** – minimální časový interval, který musí uplynout mezi dvěma po sobě následujícími požadavky na čtení nebo zápis; je dána vybavovací dobou a časem ustálení přechodových dějů na sběrnících (AS, DS); stále se zkracuje (ale pomaleji než se zkracují taktys procesorů);

**Kapacita paměti** – jaké množství informace je možné v paměti uchovat (kB, MB, GB, TB, 1kB = 1024 bytů) – vhodné udávat ve tvaru respektujícím organizaci paměti (součin počtu paměť. míst a délky paměťového místa, např. 4Mx4 byty);  
 $Ki$  (kibi) =  $2^{10}$ ,  $Mi$  (mebi) =  $2^{20}$ ,  $Gi$  (gibi) =  $2^{30}$ ,  $Ti$  (tebi) =  $2^{40}$

**Přenosová rychlosť** – udává počet datových jednotek přenesených do nebo z paměti za sekundu.

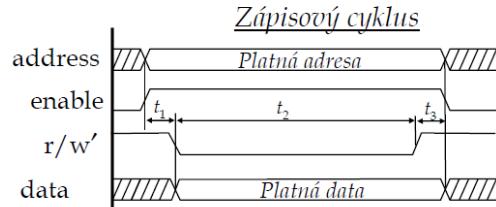
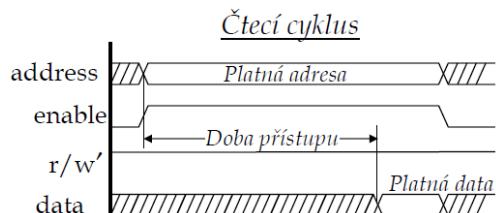
## Časování asynchronní statické RAM

Časování zajišťuje řadič (paměti / procesoru)

Doba přístupu – doba, za kterou získáme platná data (vybavovací doba pro čtení);

Čtecí cyklus je dán dobou přístupu a přečtením dat;

Zápisový cyklus:  $t_1+t_2+t_3$



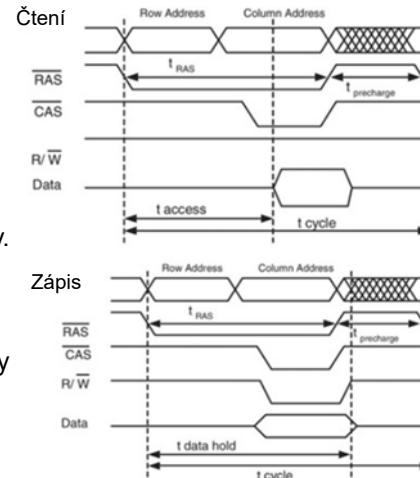
## Časování dynamické RAM

Adresa časově multiplexovaná (nejprve řádek a potom sloupec)  $\Rightarrow$  úspora vodičů.

RAS (Row Access Strobe) - řídící signál řádkové adresy,

CAS (Column Access Strobe) - řídící signál sloupcové adresy.

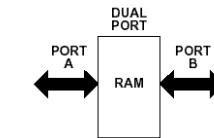
**Synchronní paměti** – všechny řídící signály, adresové vstupy i datové I/O se mění v rytmu CLK  $\Rightarrow$  lepší spolupráce s dalšími obvody počítače.



13

## Víceportová paměť RAM

Víceportové (vícebranové) paměti mají obecně  $N$  adresových vstupů a  $N$  (nebo méně) datových vstupů či výstupů



Nejčastěji 2-portové:

- *simple (pseudo) dual-port* (jeden port pro čtení a druhý pro zápis do rozdílných buněk, často na stejně frekvenci),
- *true dual-port* (dva plnohodnotné porty, ze kterých lze libovolně provádět čtení či zápis při dvou různých frekvencích).

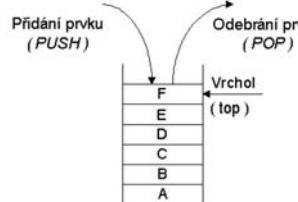
U víceportových pamětí je třeba zjistit chování při současném zápisu a čtení z téže adresy (buňky).

14

## LIFO

*Last In First Out* – zásobníková paměť (stack)

- čte se poslední zapsaná položka – ovládání instrukcemi PUSH (zápis) a POP (čtení)
- pro manipulaci s uloženými daty se udržuje tzv. ukazatel zásobníku (vrchol zásobníku), který udává relativní adresu poslední přidané položky
- obsahem zásobníku mohou být jakékoli datové struktury

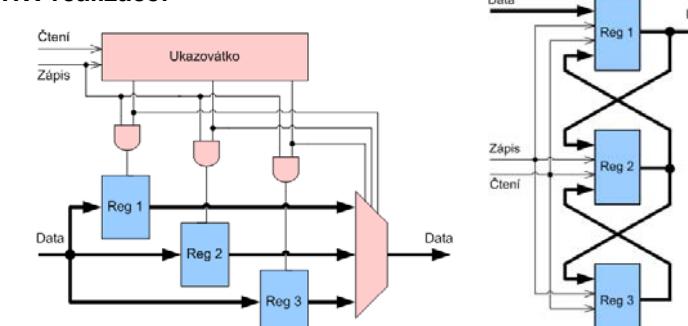


15

## LIFO - realizace

Realizace může být softwarová i hardwarová;

### HW realizace:

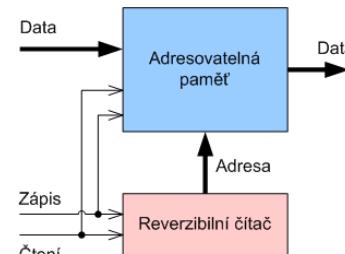


a) Registry určené ukazovátkem

b) Sériově řazené registry

16

## LIFO – realizace (pokračování)



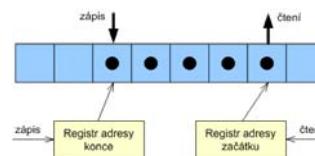
c) Adresovatelná paměť s reverzibilním čítačem

Nejčastější využití v procesoru – do zásobníku jsou ukládány návratové adresy a příznaky stavu procesoru při přerušení a skocích do podprogramů.

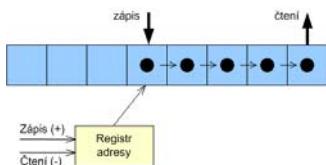
17

## FIFO – principy funkce

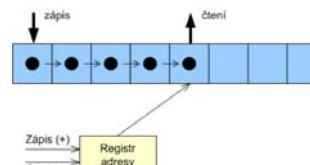
- bez přesouvání obsahu:



- s přesouváním obsahu:



a) přesouvání při čtení

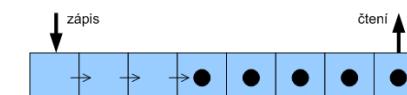


b) přesouvání při zápisu

19

## FIFO – principy funkce

- c) s přesouváním obsahu při zápisu i při čtení  
(fronta s probubláváním – posouvá každou položku po zápisu asynchronně až do posledního volného místa)  
u každého paměťového místa musí existovat indikátor obsazenosti (klopny obvod).



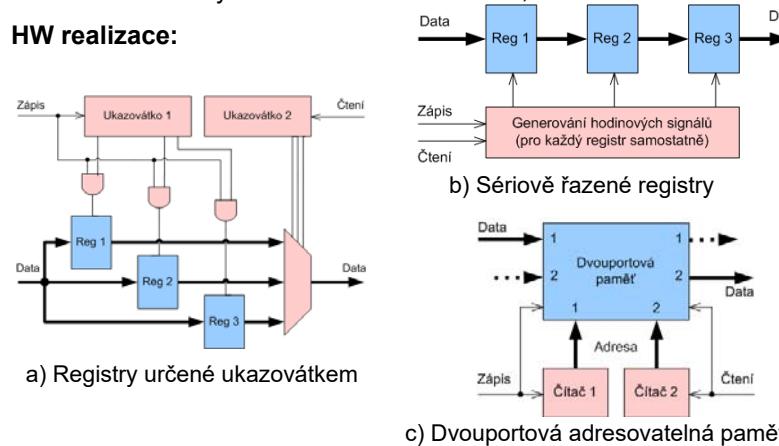
V principu je možné zápisy a čtení provádět jediným hodinovým signálem (Single Clock FIFO) nebo dvěma (Dual Clock).

20

## FIFO - realizace

Realizace může být softwarová i hardwarová;

### HW realizace:

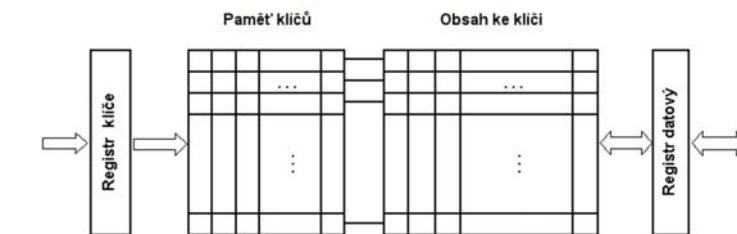


21

## Asociativní paměť'

CAM (Content Addressable Memory) – obsahem adresovatelná paměť'

- tvořeny tabulkou, která obsahuje sloupec s klíči (tagy), podle kterých se v asociativní paměti vyhledává; v další tabulce jsou uchovávaná data, popř. další informace nutné ke správné funkci (platnost dat aj.);
- použití ve vyrovnávacích a virtuálních pamětech.



22

## Vyrovnávací paměť' (cache, buffer)

Obecně rychlá polovodičová paměť umístěná mezi rychlým zařízením (procesorem) a zařízením pomalejším (např. operační paměť, HDD), která vyrovnává rozdíly v rychlosti

- zvyšuje výkon systému (komunikace s rychlými paměti typu SRAM, data se přenáší po blocích – ne náhodně, velikost bloků je různá podle úrovně cache – několik bytů až cca 1 kB);
- obvykle se dělí na část pro data a pro instrukce;
- pohyb v paměti cache řídí *řadič cache* - umístěn přímo v CPU (L1, L2) nebo je součástí čipové sady (L2, L3);
- Vyrovnávací paměť může být i softwarová (SMARTDRV, VCACHE), zvl. vyrovnávací paměť pro pomalé periferie (HDD, vypalovací mechaniky, síťové adaptéry) – lépe označovat jako buffer.

23

## Cache v procesorových systémech

Nejčastěji rozdělujeme cache na:

- **cache první úrovně L1** (first level, primární)
  - co nejbliže ALU (součástí CPU),
  - velmi rychlá,
  - relativně malá (velikost cca 32 kB – 128 kB);
- **cache druhé úrovně L2** (second level, sekundární)
  - pomalejší vůči L1,
  - velikost cca 256kB – 1MB,
  - u vícejádrových CPU každé jádro samostatná L2;
- **cache třetí úrovně L3**
  - u vícejádrových CPU sdílená všemi jádry,
  - velikost jednotky MB.

24

## Rozdělení cache podle zápisu

Podle způsobu práce při zapisování dat:

- 1) *Write-Through* – (zápis skrz cache, přímý zápis) k zápisu do operační paměti dochází ihned po zápisu CPU do cache (o další osud dat se stará cache) – průběžné „propisování dat“ (obecně pomalejší, ale jednodušší realizace).
- 2) *Write-Back* – (opožděný zápis) data jsou zapisovány do operační paměti až když je cache zaplněna – vyšší výkon (větší pravděpodobnost, že v cache jsou potřebná data).

Uvádí se, že operace zápisu představují zhruba 5-30 % paměťových operací.

## Implementace cache

Na principu asociativních pamětí; při přístupu do cache je nutné zadat adresu, z níž data požadujeme – tato adresa (buď celá nebo její část) je považována za tag

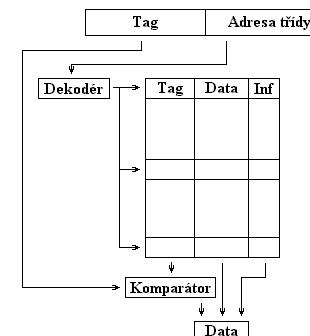
Druhy: přímo mapované  
n-cestně asociativní  
plně asociativní

## Přímo mapovaná (1cestná) cache

Zadaná adresa se rozdělí na 2 části: tag + adresa třídy;

Dekodér vybere podle adresy třídy jeden tag (řádek), jeho tag je následně porovnán s tagem adresy; není-li shoda, je z OP natažen celý řádek dat (více, např. 64 B dat).

Jednoduché, ale málo efektivní.



## Plně asociativní cache

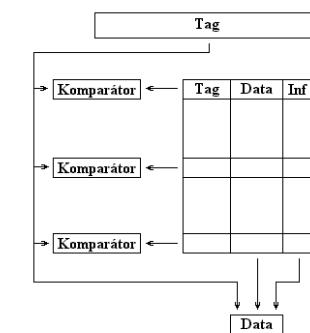
Celá adresa (ze které se budou číst data, resp. na kterou se budou zapisovat) je brána jako tag (porovnává se s tagy v cache – shoda znamená přítomnost dat v paměti).

Nevýhody:

- mnoho komparátorů
- nutná velká kapacita paměti pro tagy

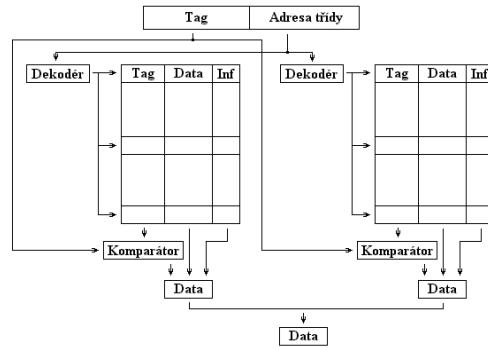
Výhody:

- velká rychlosť



## n-cestně asociativní cache

Adresa třídy pomocí dekodérů vybere jeden řádek v každé tabulce, jehož tagy se porovnávají (nejpoužívanější);  
- pro velká „n“ se blíží plně asociativní variantě



## Parametry cache

Parametry vypovídající o kvalitě cache

- pravděpodobnost úspěchu (*hit ratio*)
- pravděpodobnost neúspěchu (*miss rate*), neboli pravděpodobnost výpadku bloku (*fault*).
- lze definovat zvlášť pro čtení a zápis, pro data a instrukce.

Doba nalezení dat je v případě úspěchu přístupová doba cache, v případě neúspěchu se přičítá ztrátová doba (*miss penalty*), což je doba potřebná na přisunutí bloku, tj. doba potřebná na uvolnění místa + přístupová doba k OP + doba přesunu bloku.

Výpadky cache lze minimalizovat i softwarově (překladačem).

## Velikost bloků cache

Vlastnosti větších bloků:

- větší blok obsahuje více slov v blízkosti požadovaného slova, více instrukcí jdoucích po sobě ⇒ způsobuje méně výpadků
- vzrůstá miss penalty (hlavně na přenos bloků)
- při velkých blocích a dané velikosti cache je v cache málo bloků ⇒ roste miss rate



## Strategie řízení přesunu dat

Použití v cache a při přesunu stránek/segmentů virtuální paměti;

Dojde-li k zaplnění (a je třeba zavést další blok), který z bloků má paměť opustit:

- *LRU* (Least Recently Used) – nejdéle nepoužívaný blok; nejčastější algoritmus
- *NFU* (Not Frequently Used) – blok nejméně používaný
- *FIFO* (Aging - stárnutí) – nejstarší blok
- *RAND* (Random) – náhodně vybraný blok

Pro správu algoritmů jsou nutné další obvodové prvky, např. čítače pro udržování času použití.

## Efektivita cache

*AMAT* (Average Memory Access Time) – průměrná doba přístupu do paměti (cache + OP):

$$AMAT = HT + MR \cdot MP$$

*HT* ... Hit Time – čas pro nalezení a získání položky v cache

*MP* ... Miss Penalty – čas získání dat z nižší úrovně paměť. hierarchie

*MR* ... Miss Rate – pravděpodobnost neúspěchu (výpadku bloku)

$$HT = (1 - MR) \cdot T_C = HR \cdot T_C \quad T_C \dots \text{vybavovací doba cache}$$

$$MR = 1 - HR$$

*HR* ... Hit Rate – poměrná úspěšnost nalezení dat v cache k celkovému počtu přístupů do paměti

Např.  $T_C = 10 \text{ ns}$ ,  $MP = 180 \text{ ns}$ ,  $MR = 0,05 \Rightarrow AMAT = 18,5 \text{ ns}$   
zlepšení vůči *MP* (úměrné vybavovací době OP) je cca 10x

## Výkonnostní rovnice CPU s cache

$$\begin{aligned} T_{CPU}(prg) &= (cyc_{CPU} + cyc_{MEM}) \cdot T_{clk} = \\ &= IC \cdot (CPI + MAPI \cdot MR \cdot MP) \cdot T_{clk} \end{aligned}$$

kde:  $T_{CPU}(prg)$  ... doba provádění programu prg  
 $cyc_{CPU}$  ... počet taktů, které procesor stráví výpočtem  
 $cyc_{MEM}$  ... počet taktů, které zabere práce s pamětí  
 $T_{clk}$  ... perioda hodin procesoru  
 $IC$  ... počet instrukcí programu prg (Instruction Count)  
 $CPI$  ... (průměrný) počet hod. taktů pro zpracování jedné instrukce (Cycles Per Instruction)  
 $MAPI$  ... (průměrný) počet přístupů do paměti na jednu instrukci (Memory Access Per Instruction)

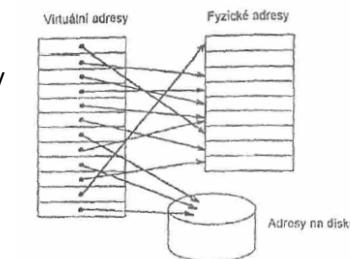
34

## Virtuální paměť (adresace)

Zvláštní způsob správy operační paměti (OP), umožňuje operačnímu systému využívat vnitřní paměť, která je větší než skutečná fyzická velikost paměti. V danou chvíli nepotřebná paměť se odkládá na pevný disk. Virtuální adresní prostor je tedy realizován jak „rychlou“ OP, tak diskem.

Účelem je:

- vytvořit možnost efektivního sdílení paměti mnoha programy i více uživateli;
- odstranění omezení fyzické velikosti paměti (OP je dražší než prostor na disku).



35

## Virtuální paměť - charakteristika

- pracuje se s fyzickým a logickým adresním prostorem;
- o všechny adresy se stará správa virtuální paměti, která rozhoduje o tom, která část bude zavedena do OP a která bude odložena do odkládacího prostoru (swap);
- běžící procesy mohou využívat větší paměť;
- každý běžící proces má k dispozici vlastní paměťovou oblast (pouze on sám);
- paměť se z hlediska procesu jeví jako lineární (i když fyzicky je na různých místech paměti i odkládacího prostoru);
- může dojít ke ztrátě výpočetního výkonu (častou výměnou dat mezi pamětí a diskem).

36

## Virtuální paměť - implementace

Existují 2 metody implementace:

- **stránkování** (paměť je rozdělena na relativně velké bloky stejně velikosti, např. 4 až 64 kB; horší využití místa);
- **segmentace** (paměť je rozdělena na bloky různé velikosti, větší volnost při programování, netřeba např. přesouvat celou stránku, pokud stačí pár bytů);  
**segment** je skupina po sobě následujících paměť. míst, jejíž velikost lze měnit i během provádění programu a má jedinou počáteční adresu; obtížné umísťování segmentů do OP – musí zůstat souvislý.

37



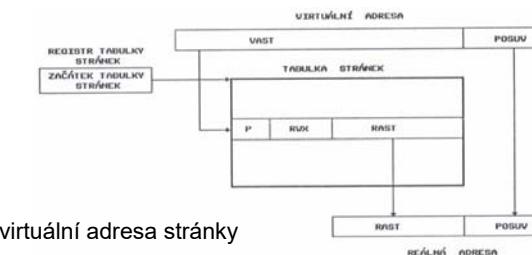
## Překlad adres

Překlad adres se realizuje tabulkou (uloženou také paměti); aby tabulka nebyla příliš rozsáhlá, nepřekládá se celá virtuální adresa na fyzickou, ale jen část:

- při překladu se kontroluje i oprávnění přístupu (autorizace) vyjádřený znakem RWX (read/write/execute) – odvozuje se z kódu instrukce a z režimu práce (čtení, zápis, spuštění programu);
- jestliže hledaná adresa stránky/segmentu není v OP, vyvolá se přerušení, uvolní se stránka/segment v OP a načež se z odkládacího prostoru požadovaná stránka/segment (mezitím může běžet jiná úloha). Strategie výběru stránky, která se odstraní z paměti (oběť), je obdobná jako u cache (FIFO, LRU, NFU,...).

38

## Překlad adres stránkové paměti



VAST ... virtuální adresa stránky

RAST ... reálná adresa stránky

P ... příznak určující, zda je stránka v operační paměti

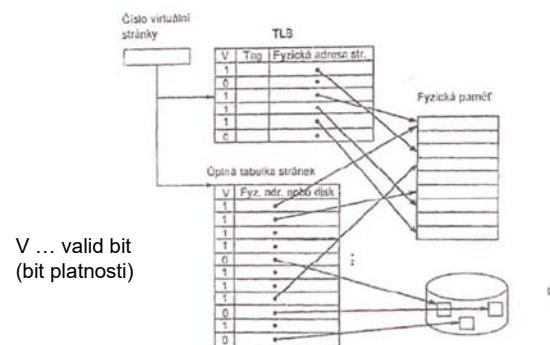
Překlad virtuálních adres u segmentů je analogický, pouze v tabulce segmentů je třeba mít informaci o jeho velikosti

39



## Tabulka přeložených adres

TLB (Translation Look-Aside Buffer) – používá se pro zrychlení překladu virtuálních adres pomocí asociativní paměti (pokud se vstupní adresa nenalezné v TLB, hledá se v úplné tabulce stránek; v TLB jsou nejčastěji volané stránky).

V ... valid bit  
(bit platnosti)

40

## ECC

(Error Correction Code, Error Checking and Correcting) - systém nadbytečné informace, která slouží k objevení a opravě vzniklých chyb – najde a opraví chybu v 1 bitu, detekuje chybu ve 2 bitech (kontrola v reálném čase), nutná podpora řadiče paměti (chipsetu); např. při 64 bitech dat je ECC 8 bitů.

Pokročilejší technologie:

multiple-bit error correction, Chipkill, memory scrubbing, Intel Single Device Data Correction (SDDC).

41



## Počítačové paměti DDRx DRAM

DDR (Double Data Rate) – synchronní dynamické paměti, u kterých dochází k přenosu dat na náběžnou i sestupnou hranu hodin.

**Časování paměti:** tCL–tRCD–tRP–tRAS, např. CL9-10-9-27

tCL (CAS Latency) – zpoždění výstupu dat po adresaci sloupce;  
tRCD (RAS to CAS Delay) – zpoždění mezi výběrem řádku a adresací sloupce;

tRP (RAS Precharge) – zpoždění po výběru paměti do adresace řádku;

tRAS (Row Active Time) – doba, jak dlouho musí zůstat RAS aktivní;

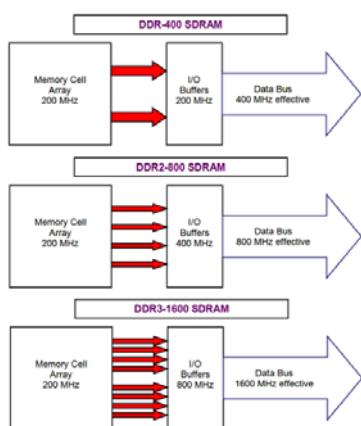
Casy se udávají v počtu hodinových taktů => čím nižší, tím lepší.

## Srovnání pamětí DRAM v PC

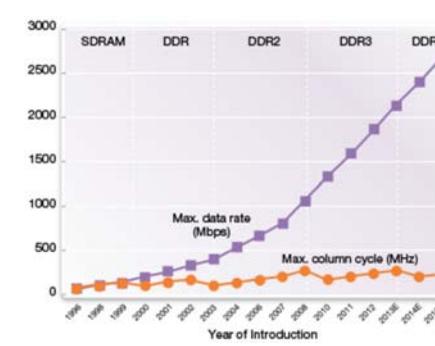
DRAM	Uvedení	Napětí	Propustnost	Frekvence
SDR	1993	3,3 V	1,6 GB/s	66-133 MHz
DDR	2000	2,5 V	3,2 GB/s	266-433 MHz
DDR2	2003	1,8 V	8,5 GB/s	533-1066 MHz
DDR3	2007	1,5 V	17 GB/s	1066-1600 MHz
DDR4	2013	1,2 V	25,6 GB/s	2133-3200 MHz
DDR5	2020	1,1 V	32 GB/s	4800-7200 MHz



## Paměti DDRx DRAM

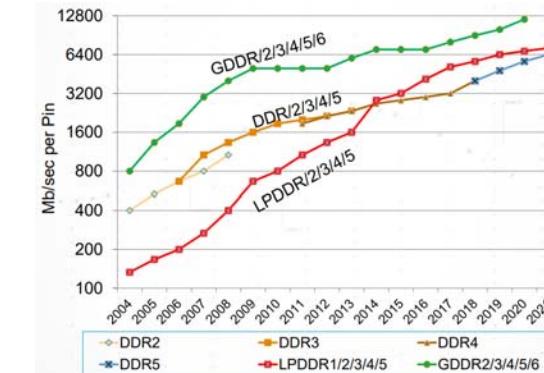


Rychlosť samotných paměťových čipů se v podstatě nemění, zrychlují se sběrnice (I/O buffery).



## Paměti DDRx DRAM

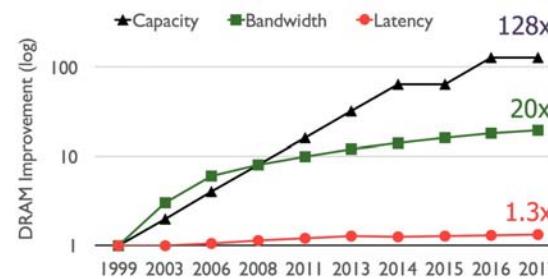
Zvyšují se přenosové rychlosti (ne šířky) na sběrnicích – velice náročné na vysokofrekvenční diferenciální spoje (na jejich impedanci, odrazy na vedeních, limity zvlnění a jitteru).



## Vývoj výkonnosti paměti

mikroprocesory – zvyšování výkonnosti 60%/rok (2x/1,5 roku)

paměti – zvyšování výkonnosti 9%/rok (2x/10 let)



46

## SSD disky (Solid State Drive)

Disk na principu převážně non-volatilních NAND flash pamětí

SLC (Single Level Cell) – v jedné buňce 2 úrovňě (1 bit),

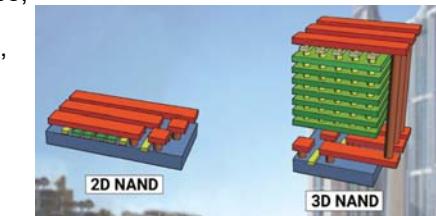
MLC (Multi Level Cell) – v jedné buňce 4 úrovni (2 bity),

TLC (Triple Level Cell) – v jedné buňce 8 úrovni (3 bity),

QLC (Quad Level Cell) – v jedné buňce 16 úrovni (4 bity).

Obecné platí: čím více bitů v buňce,  
tím větší kapacita,  
nižší cena při dané kapacitě,  
menší rychlosť,  
menší počet přepisů.

Přechod z 2D struktur na 3D  
(až 64 vrstev)  
3D MLC, 3D TLC, ...



Nejčastější rozhraní: SATA III, PCI Express nebo M.2.

47

## SSD disky (pokračování)

**Výhody:** velká přenosová rychlosť (150–1800 MB/s),  
nízká spotřeba energie (malý ztrátový výkon),  
minimální přístupová doba (0,1 ms),  
mechanicky odolné, tichý provoz, nízká hmotnosť i rozměry,  
není nutné defragmentovat.

**Nevýhody:** omezený počet přepisů flash pamětí (SLC řádově  $10^5$ ,  
TLC  $10^3$  cyklů) – nutné rovnoměrné rozložení zápisů (stárnutí)  
- v důsledku opotřebení se degraduje výkon;  
vyšší chybovost vyžaduje kód na detekci chyb (ECC);  
zápis nových dat je možné pouze do „vyčištěných“ buněk“  
=> nutný cyklus čtení-smazání-upravení-zápis  
=> pro udržení výkonu probíhá čištění uvolněných buněk na pozadí;  
nižší kapacity disků (řádově stovky GB až jednotky TB),  
vyšší cena na jednotku kapacity (ve srovnání s klasickými diskami).

48

## RAID pole

*Redundant Array of Independent Disks*

více pevných disků propojeno (převážně prostřednictvím HW řadiče)  
za účelem zvýšení rychlosti a spolehlivosti:

**RAID 0 (striping)** – data se ukládají prokládaně po blocích (sudé a  
lící bloky) na různé disky, žádný redundantní disk, zvýší se  
rychlosť (ne spolehlivost), min. 2 disky.

**RAID 1 (mirroring)** – data se ukládají duplicitně na dva disky, je třeba  
dvojnásobný počet disků, zvýší se spolehlivost (rychlosť se  
muže i snížit).

**RAID 2 (error detection and correction code)** – obdoba detekce chyb  
u paměti, nutná velká redundance.

**RAID 3 (Bit-interleaved parity)** – 1 disk paritní, je silně vytížen, data  
se dají z parity zrekonstruovat.

49



## RAID pole (pokračování)

**RAID 4** (Block-interleaving parity) – obdoba RAID3, jiný princip výpočtu parity, při zápis po blocích, paritní disk silně vytížen.

**RAID 5** (Distributed block-interleaved parity) – obdoba RAID 4, paritní bloky jsou na všech discích, 1 disk redundantní, potřeba min. 3 disky.

**RAID 6** (P+Q redundancy) – obdoba RAID 5, dva paritní disky (pomalejší zápis), dovoluje obnovu po 2 chybách.

**RAID 0+1** – dvě pole RAID 0 se spojí do RAID 1 (min. 4 disky), vyšší propustnost i spolehlivost.

**RAID 10** – pole RAID 1 se spojí do RAID 0 (min. 4 disky).

Obdobně **RAID 50** (min. 6 disků) nebo **RAID 60** (8 disků).

# Sběrnice

*Milan Kolář*

**Ústav mechatroniky a technické informatiky**



Projekt ESF CZ.1.07/2.2.00/28.0050  
**Modernizace didaktických metod a inovace výuky technických předmětů.**

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

2

## Co je sběrnice

Informační cesta umožňující přenos informace mezi jednotlivými funkčními bloky systému nebo mezi systémem a okolím.

Soustava vodičů, které přenáší data nebo signál stejného charakteru.

Sběrnice má obvykle řadič (master), kterému jsou ostatní části podřízené (slave).

Sběrnici lze definovat z hlediska:

- mechanického (tvar a rozměry mechanických dílů, konektorů)
- elektrického (úrovně U a I, přiřazení logických stavů)
- funkčního (význam jednotlivých signálů a jejich časové průběhy)
- operačního (informační kódy a formáty přenášených zpráv)

## Dělení sběrnic

Podle topologie:

- dvoubodovým spojem
- vícebodovým spojem

Podle výstupního obvodu:

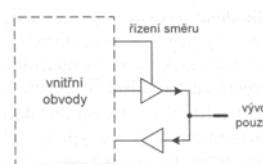
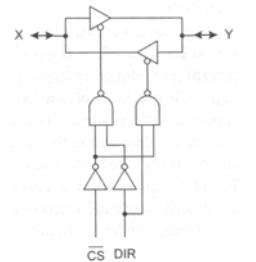
- třístavové
- s otevřeným kolektorem

Podle umístění:

- vnitřní (krátké, větší počet vodičů)
- vnější (pro malé i velké vzdálenosti)

Podle směru přenosu:

- jednosměrné
- obousměrné



3

## Dělení sběrnic

Podle úrovně signálu:

- asymetrické (SE – Single Ended)
- symetrické (LVD – Low Voltage Differential)

Podle způsobu přenosu:

- sériové
- paralelní
- kombinované (sériov-paralelní)

Podle funkce:

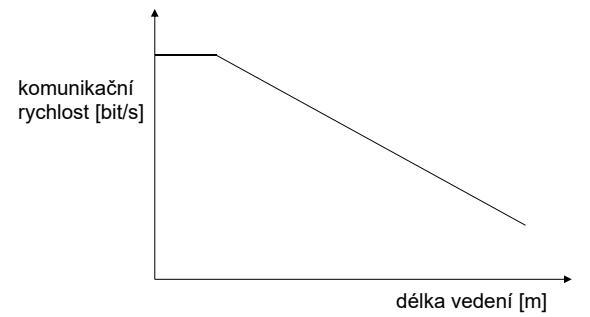
- datové
- adresové
- řídicí

4



## Technické parametry sběrnic

- maximální délka sběrnice,
- počet přípojných míst (zatížení),
- přenosová rychlosť (kapacita),
- impedance sběrnice (odpor),
- napěťové úrovně,
- konektory ...



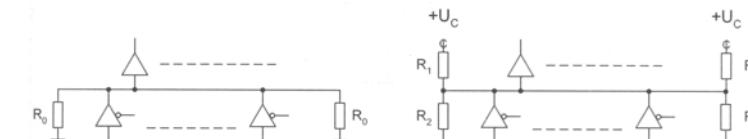
5

## Impedanční zakončení sběrnice

Na sběrnici je třeba pohlížet jako na **dlouhé vedení**, které je třeba impedančně zakončit **terminátory** – „právě jen“ na obou koncích (ideálně  $R_0 = Z_0 = R_1||R_2$ ).

U sběrnic s otevřeným kolektorem je třeba mít pull-up rezistory rovněž na obou koncích vedení (jako terminátory).

Terminátory zatěžují napájecí zdroj (nevzhodné zejména u vícevodičových sběrnic nebo u bateriového napájení).



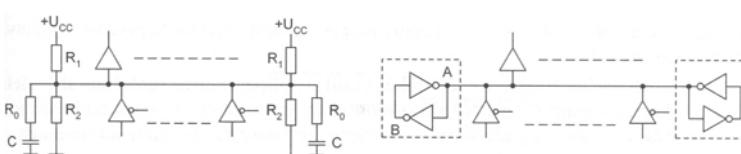
6



## Terminátory s malým příkonem

Pro snížení příkonu se používají sériové rezistory nebo doplňkové kondenzátory (pro střídavé signály zkrat) – časová konstanta  $R_0C$  se volí min. 4x větší než zpoždění signálu na vedení ( $R_1$  a  $R_2$  mají velkou hodnotu a zajišťují pouze definovanou úroveň při odpojení všech budičů).

Lze použít i **aktivní terminátory** („bus hold“) – princip bistabilního klopného obvodu (odčerpává energii přicházející vlně a již nedojde k odrazu).



7



## Synchronní a asynchronní přenos

**Synchronní přenos** – společně s daty se vysílají synchronizační impulsy, příp. speciální synchronizační vodič  
 - signál obsahuje jen celistvé násobky nejkratších charakteristických intervalů,  
 - výhodné pro velké objemy dat,  
 - v každém okamžiku konstantní přenosová rychlosť.

**Asynchronní přenos** - dávkový přenos dat

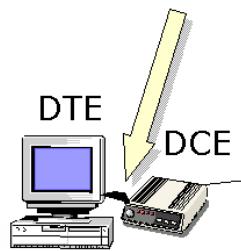
- přenosový rámec obsahuje synchronizační informace,
- vhodné při nepravidelných požadavcích na přenos,
- odolnější proti rušení.

**Arytmický přenos** – kombinace asynchronního a synchronního přenosu (v rámci jedné značky se pracuje synchronně, značky se přenášejí asynchronně).

8

## RS-232

- sériová komunikace arytmickým přenosem (sériový port);
- úrovně datových signálů v negativní logice;
- předpokládá dva typy zařízení:
  - DTE (Data Terminal Equipment) – koncové zařízení (např. počítač);
  - DCE (Data Circuit Terminating Equipment) – zařízení ukončující okruh (např. modem).



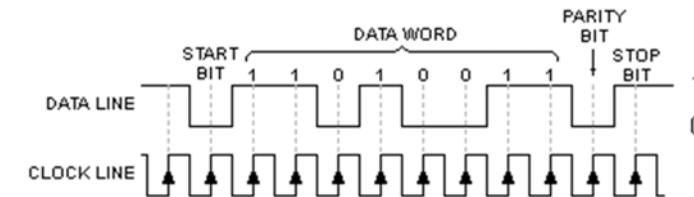
Datové signály		
Úroveň	Vysílač	Přijímač
Log. L	+5 V to +15 V	+3 V to +25 V
Log. H	-5 V to -15 V	-3 V to -25 V
Nedefinovaný	-3 V to +3 V	

Řídicí signály		
Signál	Driver	Terminátor
"Off"	-5 V to -15 V	-3 V to -25 V
"On"	5 V to 15 V	3 V to 25 V

9

## RS-232

- maximální rychlosti do 19200 bit/s (115,2 kbit/s);
- maximální délka sběrnice normována na 15 m nebo kapacitu 2,5 nF (možno dosáhnout až 50 m);
- přenos po rámcích (start bit, 5-8 datových bitů, paritní bit, 1-2 stop bity).

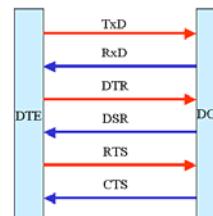


10

## RS-232

Významné signály rozhraní:

- RxD (Read Data) – vstupní data
- TxD (Transmit Data) – výstupní data
- DTR (Data Terminal Ready) – počítač je připraven
- DSR (Data Set Ready) – modem je připraven
- RTS (Request To Send) – žádost o možnost vyslat data
- CTS (Clear To Send) – povolení možnosti vyslat data



11

## I<sup>2</sup>C (Inter Integrated Circuit)

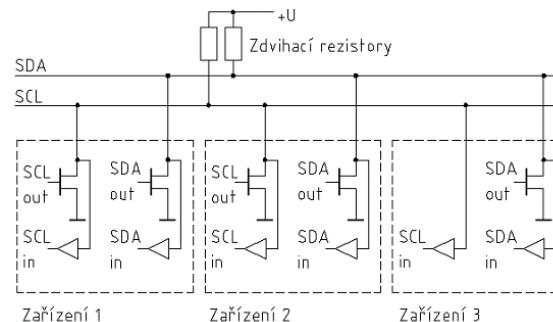
- navržen firmou Philips pro komunikaci obvodů (paměti, porty, A/D a D/A převodníky, snímače, budiče displeje, ...);
- dva typy zařízení – master a slave;
- sběrnice typu multi-master s detekcí kolizí (v každém okamžiku jen jediný master);
- každé zařízení je sw adresovatelné (má jedinečnou adresu);
- přenos je sériový, osmibitový, synchronní, poloduplexní:
  - Standard mode (100 kbit/s),
  - Fast mode (400 kbit/s),
  - High-speed mode (3,4 Mbit/s);
- maximální počet zařízení limitováno celkovou kapacitou sběrnice (nesmí překročit 400 pF), délka může být i jednotky metrů.



12

## I<sup>2</sup>C

- používá dva obousměrné vodiče – sériovou datovou linku SDA (Serial Data Line) a linku hodinového signálu SCL (Serial Clock Line);
- na sběrnici jsou budiče s otevřeným kolektorem + pull-up odpory.

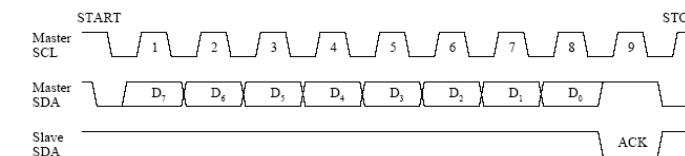


13

## I<sup>2</sup>C

Definované situace:

- **klidový stav** – SDA i SCL jsou v log.1 (neaktivní)
- **podmínka startu** – SDA je masterem stažena na log.0
- **podmínka stop** – SDA přejde z log.0 na 1, SCL v log.1
- **přenos dat** – daný vysílač přiveze na SDA 8 datových bitů (posun v rytmu SCL od mastera)
- **potvrzení (acknowledge)** – přijímač potvrdí příjem log.0 na SDA; může se přijímat další byte



14

## I<sup>2</sup>C

- neaktivní účastníci sběrnice jsou v log.1 a neustále vyhodnocují signály na sběrnici;
- je-li použit jen jeden master, vysílá SCL jen on;
- data může vysílat jak master, tak slave;
- přenos a potvrzování adres je stejné jako u dat;
- postup: master vytvoří podmínku startu a pak v bitech 7 až 1 přenese adresu součástky, v bitu 0 (R/W) je požadovaný směr přenosu; slave potvrdí adresu, pak se posílají data (každých 8 bitů je potvrzeno); přenos ukončí master podmínkou stop.



15

## SPI (Serial Peripheral Interface)

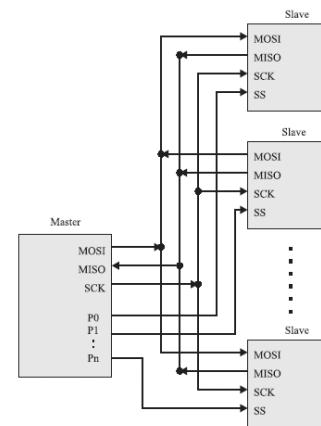
- navrženo firmou Motorola;
- dva typy zařízení – master a slave;
- synchronní sériové rozhraní se čtyřmi druhy vodičů:
  - datový výstup MOSI zařízení master (Master Out Slave In)
  - datový vstup MISO zařízení master (Master In Slave Out)
  - výstup hodinového signálu SCK z masteru
  - výběrový vodič SS (Slave Select) – aktivní v log.0 (z master do každého zařízení slave);
- rychlosť SCK je standardně 2 MHz, u variant High-speed SPI (HSSPI) až 20 MHz;

16

## SPI – připojení zařízení

Zařízení lze k SPI připojovat:

- **paralelně** (signály MISO musí být s třístavovými budiči, SS je rozveden individuálně)
- **sériově** do řetězce (k-násobné prodloužení datového rámce, aktivní všechna zařízení).

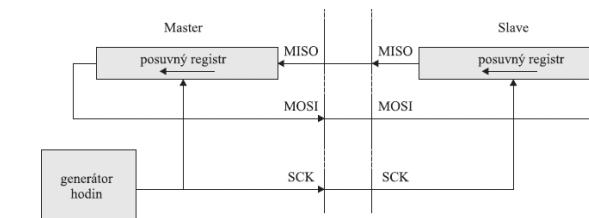


17

## SPI

Komunikace na sběrnici mezi zařízeními Master a Slave:

- obě zařízení obsahují 8-bitový posuvný registr, který postupně po 8 taktech čtou nebo do něj zapisují,
- master generuje hodinový signál SCK,
- přenos probíhá od MSB k LSB.



Princip komunikace po SPI

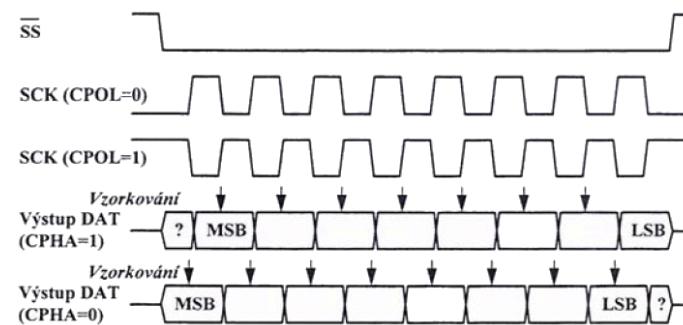
18

## SPI - módy

Konfigurační módy:

CPOL – určuje polaritu SCK

CPHA – posun datového signálu vůči hodinám



19

## USB (Universal Serial Bus)

Standard pro sériový přenos dat (4, resp. 8 vodičů – USB 3).

Charakteristické parametry:

- komunikační rychlosť
  - SuperSpeed Plus - až 10 Gbits/s (v.3.1)
  - SuperSpeed – až 5 Gbits/s (v.3.0),
  - High Speed - 480 Mbits/s (v.2.0),
  - Full Speed - 12 Mbits/s (v.1.1),
  - Low Speed - 1.5 Mbits/s (v.1.1),
- komunikační vzdálenost 1 až 5 m (se zesílením až 30 m),
- lze připojit až 127 zařízení, zpětná kompatibilita,
- log.0 ... 0-0.3 V, log.1 ... 2.8-3.6 V (diferenciální),
- zajišťuje správné přidělení prostředků (IRQ, DMA, ...).



Pin	Jméno	Barva	Popis
1	VBus	Red	+5 VDC
2	D-	White	Data -
3	D+	Green	Data +
4	GND	Black	Ground

20

## USB

- podpora ovladačů ve stávajících operačních systémech;
- autoidentifikace periferií a překonfigurovatelné periferie;
- HotSwap – připojení/odpojení za chodu počítače;
- možnost využití celé šířky pásma jedním zařízením;
- rozhraní obsahuje 5V napájení
  - zařízení mohou být napájena přímo ze sběrnice (do 100 mA, příp. 500 mA pro jedno zařízení, 900 mA pro v.3.0)
- pro vyšší rychlosti třeba stíněný kabel;
- každé USB zařízení má svoji adresu;
- konektory – typ A (USB Host), B (USB Device) a C (oboustranný).

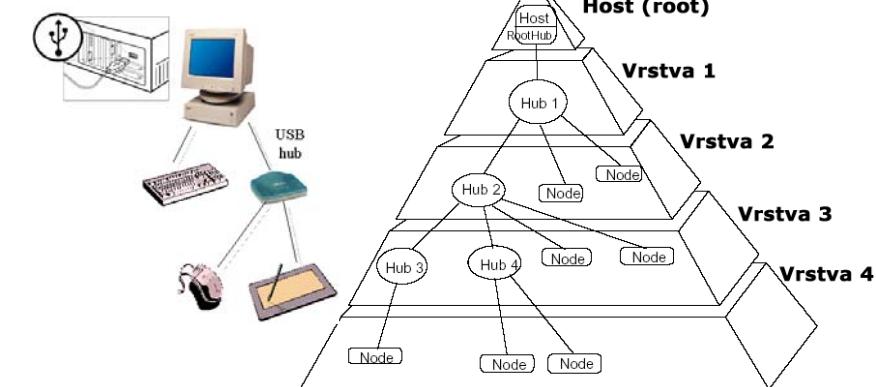


21

Modernizace didaktických metod a inovace výuky technických předmětů

Víceúrovňová hvězdicová struktura, pouze jeden hostitel

Dva typy zařízení: hub a periferie



22

## Druhy paralelních portů

- Standard Parallel Port (SPP) (Centronics)
- Enhanced Parallel Port (EPP)
- Extended Capabilities Port (ECP)

### Centronics:

- napěťové úrovně TTL,
- doporučená maximální délka kabelu 3 až 5 m,
- přenos na principu Handshake,
- osmibitový výstup pro tiskárnu,
- omezená možnost monitorování stavu tiskárny,
- rychlosť 100 - 200 kB/s.

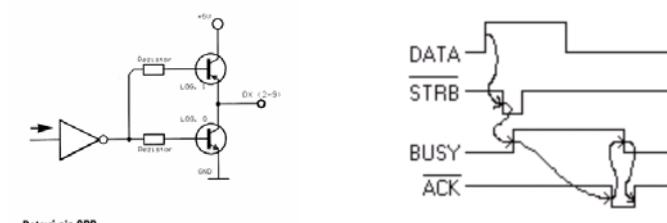
23

## Centronics (SPP)

Hlavní signály:

datové vodiče (standardně 8) – každý stíněný GND vodičem (z PC),

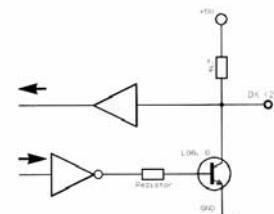
Strobe – řídicí signál oznamující platnost dat (z PC),

Busy – oznamuje, že zařízení není připraveno k přenosu  
(z periferie),Acknowledge – předávání stavových informací (z periferie),  
+ další (Error, PE, Select, Init, ...).

24

## EPP

- od roku 1991 (Intel, Xircom, Zenith DS),
- používá se pro interaktivní komunikaci s adaptérem LAN, mechanikou CD ROM, diskem, páskovou jednotkou
- 8-bitový vstup a 8-bitový výstup,
- rychlosť 500 - 2 000 kB/s.



Datový pin EPP a ECP

25

## ECP

- Hewlett-Packard LaserJet 4, Microsoft,
- obousměrný osmibitový přenos,
- vysokorychlostní přenos bloků dat,
- podpora datové komprese, kompresní poměr je 64:1,
- rychlejší než EPP (využívá DMA),
- rychlosť přenosu > 2 MB/s.

26

## Čipová sada (chipset)

Sestava integrovaných obvodů na základní desce - vlastnosti:

- řídí komunikaci na základní desce;
- určuje typy, příp. počet procesorů;
- určuje kapacitu a typy pamětí;
- určuje typ a počet rozšiřujících slotů, popř. integrované prvky (grafická, síťová nebo zvuková karta);
- obsahuje rozhraní pro FDD a HDD (P-ATA, S-ATA);
- podpora Plug and Play, USB, LAN, příp. další rozhraní;
- funguje jako řadič operační i vyrovnávací paměti;
- definuje rozhraní (systémovou sběrnici) mezi CPU a okolím;
- může řídit Power Management.

27

## Čipová sada - architektura

Nejrozšířenější: Intel, VIA, nVidia (nForce, GeForce), ATI (Radeon, CrossFire) - vzájemně nekompatibilní.

Architektura: **North Bridge – South Bridge – Super I/O**

*North Bridge* – rozbočovač řadiče pamětí (MCH – Memory Controller Hub) je přes systémovou sběrnici spojen s CPU;

- propojuje paměťovou a grafickou sběrnici;
- pracuje na frekvenci základní desky, pro níž je řadičem;
- zajišťuje rychlý přesun velkého množství dat (⇒ nutné chlazení);

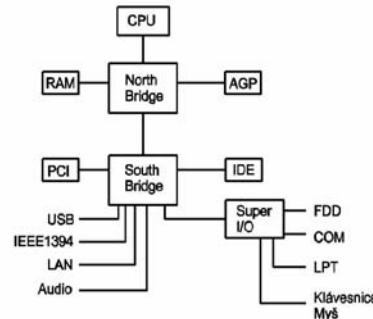
*South Bridge* – rozbočovač řadiče vstupů a výstupů (ICH – I/O Controller Hub), spojuje středně rychlé sběrnice PCI, USB, IDE, LAN, audio, zajišťuje funkce BIOSu.

28

## Blokové schéma chipsetu

**Super I/O** – pro práci s pomalejšími vstupy a výstupy (v současnosti jeho funkci přebírá South Bridge)

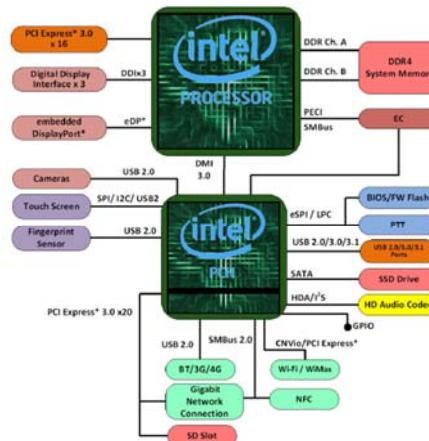
Trendem se přesouvání řadiče paměti DDRx, řadiče sběrnice PCI-e a grafického čipu přímo do CPU



29

## Současné schéma CPU a chipsetu

Severný můstek je součástí CPU, v CPU je i grafický čip, na základní desce je PCH (Platform Controller Hub) – Komunikace s CPU přes DMI (Direct Media Interface)



30

## Sběrnice FSB (Front Side Bus)

Systémová sběrnice pro komunikaci mezi CPU a základní deskou (North Bridge, respektive s paměťmi) – pomalejší než interní frekvence jádra procesoru, aby periferie zvládaly komunikaci.

**Násobitel** (Multiplier) - udává, kolikanásobně je procesor rychlejší než externí sběrnice FSB (rosté po hodnotách 0,5). Pokud chceme procesor přetaktovat (overclocking), musíme změnit buď frekvenci FSB nebo násobitel.

FSB přenáší data na sestupné i vzestupné hraně hodinového signálu, rychlosti se postupně zvyšují (do 1600 MHz).

Propustnost až 12,8 GB/s (1,6GHz x 2B x 2 (↑↓) x 2 směry).

31

## HyperTransport

Vylepšená sběrnice FSB od firmy AMD – duální jednosměrná sériová sběrnice (pro každý směr komunikace jedna sběrnice) založená na principu paketů (point-to-point)

- vysoká propustnost (mezi CPU a North Bridge),
- nízké zpoždění (latence),
- podpora více procesorů,
- nízké napájení (méně ztrátového tepla),
- může mít obecnější použití (nejen mezi CPU a North B.) - např. propojení CPU ve víceprocesorových systémech.

Použita poprvé v procesorech AMD v roce 2001, různé verze: 1.x (0,8GHz), 2.0 (1,4GHz), 3.0 (2,6GHz), 3.1 (3,2GHz).

V současnosti 32-bitová (pro každý směr), frekvence 3,2 GHz, ⇒ propustnost 51,2 GB/s (4B x 3,2GHz x 2 (↑↓) x 2 směry).

32

## QuickPath Interconnect (QPI)

Náhrada FSB od fy Intel (2008), obdoba HyperTransportu

Skládá se ze dvou jednosměrných sběrnic point-to-point

2 x 21 signálů (16 data + 4 řídicí + 1 hodinový),  
všechny signály jako diferenciální páry => 84 vodičů

Přenos dat na obě hrany hod. signálu.

Rychlosti: 2,4 GHz, 2,93 GHz, 3,2 GHz

propustnost 25,6 GB/s (2B x 3,2GHz x 2 (↓) x 2směry).

QPI sběrnic může být v CPU i více (zejména u vícejádrových)  
- distribuce původní systémové sběrnice.

V dnešních CPU neopouští QPI čip, uvnitř je i řadič PCI-e,  
vně vychází jen sběrnice DMI (Direct Media Interface) spojená  
s PCH (Platform Controller Hub) – v podstatě South Bridge.

Od 2017 nahrazeno Intel Ultra Path Interconnect (UPI).

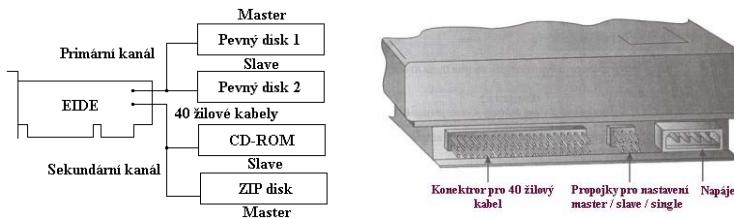
33

## P-ATA (IDE, ATA)

Integrated Drive Electronics,

(Parallel Advanced Technology Attachment)

- 40-žilový kabel (80-žilový), max. 45cm;
- na každý kanál lze připojit až 2 zařízení (master – slave);
- ATAPI (ATA Packet Interface) – pro CD-ROM, ZIP, LS-120.



35

## Infinity Fabric (IF)

Proprietární sběrnice od fy AMD (2017), nástupce HyperTransportu

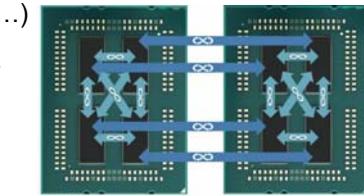
Univerzální vnitřní sběrnice pro přesuny dat na úrovni čipu nebo mezi čipy (mezi CPU, GPU nebo APU)

Škálovatelné z hlediska datové propustnosti (od 30 do 512 GB/s), obousměrná 256bitová

Kombinace datové a řídicí sběrnice – má 2 části:

- Scalable Data Fabric (přesuny dat)
- Scalable Control Fabric (přenosy příkazů, řízení čipu, zabezpečení, řízení spotřeby, ...)

V nových procesorech AMD řady Ryzen.



34

## P-ATA - přehled

Specifikace	ATA-1 IDE	ATA-2 E-IDE	ATA-3 Fast ATA	ATA-4 UltraATA	ATA-5 UltraDMA	ATA-6	ATA-7
Přenosové módy	PIO1	PIO4 DMA2	PIO4 DMA2	PIO4 DMA2	PIO4 DMA3 UDMA4	PIO4 DMA3 UDMA5	PIO4 DMA3 UDMA6
Max. přenosová rychlosť (MB/s)	8,3	16,7	16,7	33	66	100	133
Rok uvedení	1986	1994	1997	1998	1999	2000	2002
Charakteristické vlastnosti	Asynchronní přenos, 512 MB	Synchronní přenos, 4 zařízení	Technologie SMART	Přenos na obě hrany, CRC, 80 vodičů	80 vodičů	Nap. 3,3 V	

36

## S-ATA

Serial Advanced Technology Attachment

- přenosová rychlosť S-ATA – 150 MB/s (1,5 Gb/s),  
S-ATA II – 300 MB/s (3 Gb/s),  
S-ATA III – 600 MB/s (6 Gb/s)
- na jednom kabelu jedno zařízení (point-to-point)
- délka kabelu až 1 m, diferenciální přenos signálů, dva datové páry a 3 vodiče na stínění (celkem 7 vodičů)
- napájení cca 0,5 – 0,6 V (menší spotřeba) – není součástí rozhraní

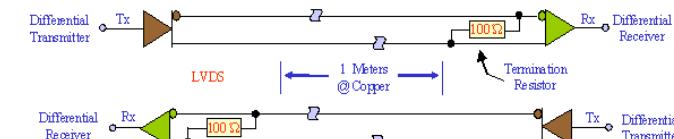


37

## PCI Express (PCI-e)

Sériová sběrnice složená ze dvou nízkonapěťových diferenciálních páru (vysílacího a přijímacího) - lane  
LVDS (Low-Voltage Differential Signaling);

- propustnost 2,5 Gb/s na 1 lane, kódování 8b/10b;
- architektura point-to-point (žádné sdílení sběrnice);
- vícestavová modulace (korekce ztrát).



38

## PCI Express - přehled

sdružování lane do linků – snadné zvyšování propustnosti

PCI-Express 1.0

Počet lanů v linku	x1	x2	x4	x8	x12	x16	x32
Rychlosť [Gb/s]	2,5	5	10	20	30	40	80
Propustnosť [MB/s]	250	500	1000	2000	3000	4000	8000

PCI-Express 2.0 – dvojnásobná propustnost vůči v. 1.0 (až 16 GB/s);

PCI-Express 3.0 – efektivnější kódování 128b/130b, zdvojnásobení propustnosti (až 32 GB/s);

39

## M.2

Relativně nové rozhraní (od 2013), někdy se označuje NGFF (Next Generation Form Factor)  
náhrada mini-SATA i mini-PCIe



Určeno pro disky, WiFi moduly, Bluetooth moduly, aj.

Velká propustnost a malá latence sběrnice  
(bez kabelů → spolehlivější)

M.2 socket 2 – rychlosť PCIe x2, propustnosť 2 GB/s

M.2 socket 3 – rychlosť PCIe x4, propustnosť 4 GB/s



40



## Rozhraní SAS

Serial Attached SCSI – nástupce SCSI rozhraní (topologie „point-to-point“), použití v serverech a diskových polích;

- rychlosť SAS: 3 Gb/s, SAS 2.0 – 6 Gb/s; SAS 3.0 – 12 Gb/s
- lze připojovat i odpojovat za provozu;
- každý disk má dva nezávislé komunikační kanály (vhodné pro redundanci např. v diskových polích – možné připojit na záložní řadič);
- rozšiřitelnost pomocí externích expandérů (stromová topologie);
- kompatibilní s normou SATA (lze kombinovat i v rámci jednoho řadiče) – disky SAS nelze připojit k řadiči SATA;
- HDD SAS mají větší otáčky (10-15 tis. ot/min), ale menší velikosti (stovky GB až jednotky TB) vůči HDD SATA.



# Speciální procesory (signálové, grafické, mobilní)

Milan Kolář  
Ústav mechatroniky a technické informatiky



Projekt ESF CZ.1.07/2.2.00/28.0050  
Modernizace didaktických metod  
a inovace výuky technických předmětů.

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

2

## Signálový procesor (DSP)

Mikroprocesor nebo mikroprocesorový systém, který je přizpůsoben pro efektivní realizaci algoritmů číslicového zpracování signálu v reálném čase s co nejmenšími vynaloženými náklady.

Architektury signálových procesorů mají různý stupeň paralelizmu zpracování instrukcí (závisí na množství instrukcí, které lze provést během jednoho cyklu, a na množství provedených operací v každé instrukci) a paralelního zpracování dat.

- úroveň paralelizmu omezuje technologie výroby IO.

Předpokládá se průběžné zpracování velkého množství dat „protékajících“ procesorem (na zpracování dat se aplikují různé algoritmy DSP).

## Požadavky

- zpracování toku dat v reálném čase (vysoký pracovní kmitočet, využití co nejvyššího stupně paralelizmu),
- velký matematický výkon,
- malé nároky na paměť (a tím malá plocha čipu a nízký příkon u zpracovávaných algoritmů),
- efektivní instrukční sada (úsporné programy),
- výkonné kompilační programy z vyšších programovacích jazyků,
- nízká cena, dostupnost na trhu,
- přenositelnost programů mezi generacemi mikroprocesorů,
- snadná modifikace zpracování toku dat.

## Operace signálových procesorů

Nejčastější algoritmy číslicového zpracování signálů:

- konvoluce
- číslicová filtrace (IIR, FIR)
- diskrétní transformace
- korelace
- práce s maticemi

$$y_n = \sum_{i=0}^M h_i \cdot x_{n-i}$$

$$y_n = \sum_{i=0}^M a_i \cdot x_{n-i} - \sum_{i=1}^L b_i \cdot y_{n-i}$$

$$r_n = \frac{1}{N} \sum_{i=0}^{N-1} x_i \cdot x_{n+i}$$

Důraz je kladen na násobení hodnot uložených v datové paměti konstantou nebo proměnnou a na akumulaci těchto dílčích součinů, dále aritmetické a logické posuny.

## Signálové procesory

1979 – první signálový procesor I2920 (Intel)  
výrobci – Texas Instruments, Analog Devices, Freescale (dříve Motorola).

Výpočetní výkon – hodnocen MIPS, MFLOPS, MOPS, MMACS.

Porovnání výkonnosti architektur signálových procesorů:

- množství instrukcí, které lze provést během jednoho hod. cyklu,
- počet instrukcí provedených paralelně,
- množství provedených operací v každé instrukci.

5

## Signálové procesory - dělení

Dělení podle (dynamického) rozsahu: 16, 24, 32 bitové  
(závisí na použití – zpracování hlasu v telefonii, zvuku  
v HIFI technice, obrazu, 3D grafiky).

Dělení podle periferií:

- DSP mikroprocesory* (neobsahují paměť ani další periferie),
- DSP mikrokontroléry* (obsahují rozličné periferie – např. A/D a D/A převodníky).

Částečně dochází k prolínání mikrořadičů a signálových procesorů  
=> hybridní mikrokontroléry.

6

## Dělení podle aritmetiky

Rozdelení podle typu zpracovávaných dat:

- *procesory v celočíselné aritmetice*  
levné, ale náročnější vývoj algoritmů, nutná tzv. normalizace;
- *procesory s pevnou řádovou čárkou*  
rychlejší, levnější, menší přesnost a rozsah (snadné přetečení), kompromis mezi celočíselnou a FP;
- *procesory s pohyblivou řádovou čárkou* (32-bitové)  
složitější struktura procesoru, větší spotřeba energie, malý kvantizační šum, větší rozsah, programování převážně ve vyšších programovacích jazycích, jednodušší algoritmy.

7

## Architektury signálových procesorů

DSP má většinou:

- harvardskou architekturu (zejména proto, že kód i data mají zvláštní sběrnice, možnost uchovávat data i v paměti programu);
- omezenou instrukční sadu (RISC);
- zřetězené zpracování dat (pipeline) => zvyšování kmitočtu;
- registrově orientovaná ISA architektura (množství registrů);
- kapacitní přenos operandů k ALU nebo MAC  
=> větší počet adresových a datových sběrnic (2 až 4) vně procesoru vyvedena většinou jen jedna AS a DS.
- vysoký vnitřní paralelismus (superskalární, speciální typy LIW a VLIW), někdy se využívají architektury paralelních systémů, příp. shlukování DSP do výpočetních sítí.

8

## Hlavní části běžných DSP

**Hlavní aritmetická jednotka** (jednotlivé části nezávislé):

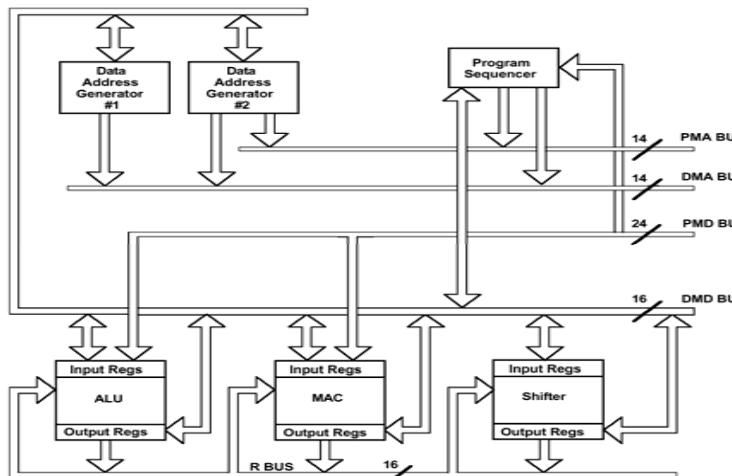
- jednotka MAC (Multiplier and Accumulator) – paralelní násobička a sčítáčka (na 1 strojový cyklus) – velikost střadače dána součtem velikostí operandů + 8 bitů;
- jednotka ALU (+, -, \*, /, inkrementace, dekrementace, log. fce);
- jednotka podporující paralelní posuny (barell-shifter).

**Generátory adres DAG** (Data Address Generator) - správa adres pro čtená data a konstanty, nejčastěji dva (pro konstanty a pro data) – podporují různé druhy adresování (inkrementace, dekrementace, reverzně bitové adresování, adresování v kruhovém zásobníku apod.).

**Čítač instrukcí** – může opakovat jednu či více instrukcí.

9

## Architektura běžných DSP



11

## Další možné části DSP

- nezávislé sběrnice (PMA, PMD, DMA, DMD, R bus);
- jedno- nebo dvouportová paměť RAM pro data, příp. několik nezávislých paměťových bank;
- rychlé sériové kanály (převážně synchronní);
- programová paměť – nejčastěji Flash, ale i ROM;
- obvodová zásobníková paměť;
- DMA kanály se speciálními sběrnicemi (i vícenásobné);
- přerušovací systém (+ zrcadlové sady registrů);
- čítače/časovače, A/D převodníky;
- další standardní rozhraní (RS-232, SPI, CAN, I2C, USB, Ethernet);
- generátor hodin (s externím krystalem), fázový závěs.

10

## Paralelní systémy

Třídění paralelních systémů podle počtu programů:

- **SI** (Single Instruction Stream) – v čase řešení problému běží jeden program;
- **MI** (Multiple Instruction Stream) – během řešení běží více programů paralelně.

Třídění podle toků dat (podle počtu zpracovávaných dat. souborů):

- **SD** (Single Data Stream) – jeden zpracovávaný tok dat;
- **MD** (Multiple Data Stream) – více zpracovávaných toků dat.

12



## Čtyři kategorie paralelních systémů

**SISD** – chápán jako klasický von Neumannův počítač s jedním programem a jedním sériově přiváděným tokem dat;

**MISD** – hypotetická kombinace několika programů zpracovávajících jeden tok dat (nezaměňovat se zřetězeným zpracováním instrukcí, kde je vždy aktivní pouze jeden program), lze uvažovat vektorové počítače;

**SIMD** – větší počet funkčních jednotek pracujících na řešení téhož programu (všechny jednotky provádí současně tutéž instrukci, ale každá s jinými daty);

**MIMD** – obecný typ paralelního systému, který obsahuje jednotky již tak samostatné, že každá z nich řeší samostatný program a zpracovávají jiná data.

13

14

## Paralelismus zpracování instrukcí

### Čtení s jedním přístupem (Single Issue Processing)

řídicí jednotka čte pouze jednu instrukci během jednoho hod. taktu, ale paralelismus je realizován sdružením několika operací do jedné instrukce (harvardská architektura)  
=> SIMD – paralelní zpracování dat

Tím je zahušťován programový kód a šetřena programová paměť, ale znamená to řadu omezení (které registry se musí použít pro které operace a které operace lze sdružit do jedné instrukce) – toto rovněž ztěžuje vytváření komplilátorů.

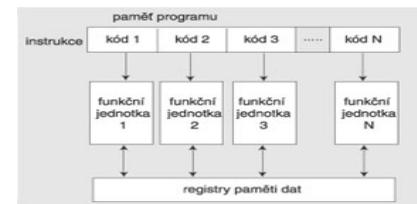


## Paralelismus zpracování instrukcí

### Metoda čtení s více přístupy (Multi-Issue Processing)

provádí se jedna operace v jednom instrukčním cyklu, přitom se čte skupina instrukcí paralelně (vícenásobné čtení)  
– paralelní zpracování instrukcí

Jedna instrukce je složena z několika částí, které současně ovládají skupinu funkčních jednotek



15

16

## Superskalární architektura

spíše typická u procesorů pro všeobecné použití

závislost registrů a dat na instrukcích řeší za běhu programu sám procesor – jednotka *Schedule Unit*  
=> jednodušší na programování, větší hustota kódu;

instrukce pro dílčí jednotky jsou jednoduché  
=> jednodušší komplilátory, větší rychlosť;

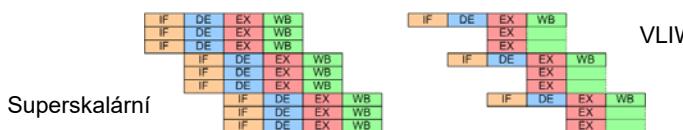
umožněna binární kompatibilita mezi generacemi procesorů (menší závislost na konkrétní architektuře procesoru);

programátor přesně neví, které instrukce procesor seskupí pro paralelní zpracování => problematický odhad celkového počtu instrukčních cyklů (problematická optimalizace).

## Architektura VLIW

VLIW (Very Long Instruction Word) – architektura se čtením s více přístupy (obdoba předchozí architektury pro signálové procesory) – delší instrukce sestavené z dílčích částí, které mohou být provedeny paralelně (instrukční packet) - např. 8 dílčích 32-bitových subinstrukcí (256 bitů)

- o paralelizaci instrukcí rozhoduje překladač, případně samotný programátor (HW nekontroluje hazardy);
- větší náročnost na programovou paměť (větší programy se synchronizačními NOPy);
- různé generace VLIW jsou obecně binárně nekompatibilní.



17

## Vícejádrové DSP procesory

Signálové procesory s velkým výkonem (jednojádrové):  
frekvence 200 až 1200 MHz, výkon řádově jednotky GIPS

Trend k vícejádrovým CPU (jako u univerzálních CPU)

Např. TMS320C6472 od firmy Texas Instruments

- CPU s pevnou řádovou čárkou, 6jádrový DSP, 32bitový
- frekvence až 700 MHz
- 8 instrukcí/cyklus v jádře →  $6.8.700 = 33600$  MIPS
- 8 MAC  $16 \times 16$  v jádře →  $6.8.700 = 33600$  MMACS
- cache L1 6 x 64kB, L2 4416 kB (768 kB sdílená)
- energetická spotřeba 0,15 mW/MIPS
- napětí jádra 1.2 V, I/O buňky 1.2 V, 1.5 V, 1.8 V, 3.3 V
- I2C, Ethernet 10/100/1000, 12 x 64bit. časovače,
- Serial RapidIO Link, DDR2 memory controller, JTAG

18

## Grafické procesory (GPU)

Speciální řídící procesory zajišťující vykreslování dat (nejčastěji data uložená v grafické paměti a jsou obrazovaná na monitoru) – zpracovává 3D geometrii na 2D obraz.

Grafický procesor může být mnohdy výrazně výkonnější než hlavní CPU počítače – architektura se přizpůsobuje charakteru obrazových dat – stream processing (SIMD – jeden postup nad velkým množstvím dat) – paralelní architektura.

Bývá na grafické kartě, v severním můstku nebo přímo v CPU.

Nejčastěji pracují s vlastní grafickou pamětí – velká a rychlá.

Nejvýznamnější výrobci GPU:

nVidia, AMD (ATI), Intel, VIA Technologies, Matrox.

19

## GPU – hlavní části

**Shadery** – plně programovatelné části GPU (speciálními programovacími jazyky) - spíše superskalární zřetězené architektury, operace nejčastěji v pohyblivé řádové čárce; někdy jsou shadery chápány jen jako programy řídící části GPU;

**Řadič paměti** – stará se o komunikaci grafickou pamětí a GPU;

**Jednotka TMU (Texture Mapping Unit)** – nanáší textury na objekty;

**Jednotka ROP (Render Output Unit)** – stará se o konečný výstup dat z grafické karty (2D).



20



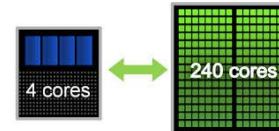
## Shadery

Vývoj z 2D a 3D grafických akcelerátorů, po roce 2000 se objevují shadery – zprvu samostatné specializované vertex a pixel, později geometry shadery:

- *Vertex shader* – provede danou transformaci na každém polygonu (vrcholu trojúhelníků), vytvoří 3D scénu, odstraní neviditelné prvky, nasvítí scénu;
- *Pixel shader* – provede operaci na každém pixelu (potahování texturou, stínování,obarvení, odlesky);
- *Geometry shader* – umožňuje přidávat a odebírat vrcholy (doplňení detailů existujícího modelu);
- *Shadery pro teselaci* – přidávání dalších detailů.

V současné době tzv. **unifikované shadery**

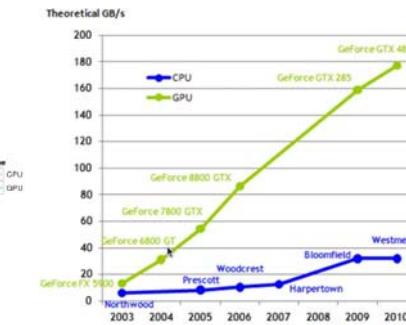
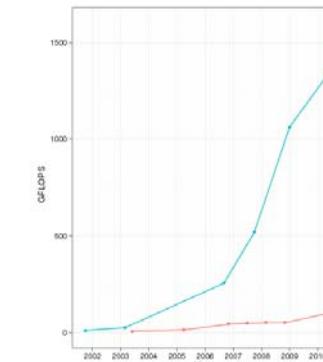
- zamezení přelévání dat mezi specializovanými shadery;
- velké množství jednodušších jader.



21

## CPU vs. GPU

Srovnání výkonu a propustnosti



22



## Mobilní procesory

Významný podíl procesorů ARM – velký výkon, nízká spotřeba, nízká cena (cca 98% mobilních zařízení – zejména mobilní telefony, navigace, tablety, ...)

Firma ARM procesory pouze vyvíjí, pak prodává licence => řada výrobců: největší asi Qualcomm – např. generace Snapdragon, Samsung (řada Exynos), Apple, Texas Instruments, nVidia, Intel (Atom), ...

Podpora v OS (Android, Windows CE, Symbian, ...).

Významnou částí mobilních procesorů jsou grafické čipy (např. řada Mali u ARM, řada Adreno u Qualcomm) nebo DSP jádra.

2011 – první mobilní čip se dvěma jádry (nVidia Tegra 2);

2012 – první 4jádrový CPU (nVidia Tegra 3).

V současnosti 8-10jádrové, 32/64bitové, 1,0-2,5 GHz.

23

## Mobilní procesory (pokrač.)

Největší důraz kladen na úsporu energie:

Architektura typu „big.little“ - kombinace různých jader s různou výkonností (není-li potřeba výkon, zapnou se pouze pomalejší a úspornější jádra) – přepínání jádra však vyžaduje množství hod. cyklů (řádově  $10^4$ ), což je energeticky a časově náročné.

Power gating – nevypínají se celá jádra, ale dochází ke snížení napětí nepoužívaných logických obvodů (až na nulu) nebo se vypínají jen určité části – pak se rychleji „probouzejí“.

Clock gating - snižování pracovní frekvence procesoru při nečinnosti.

Stále větší důraz na úsporu energie periferií ( displej, SRAM, Wi-Fi adaptér, grafika, ...).

24

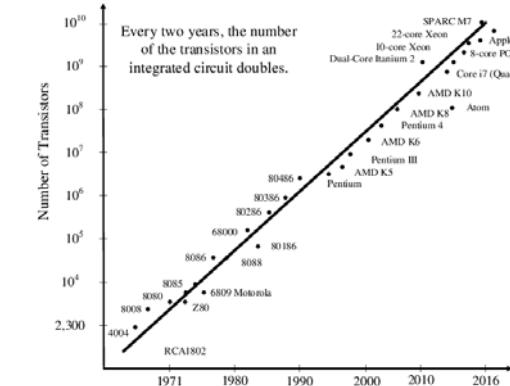
# Procesory II.

Milan Kolář

Ústav mechatroniky a technické informatiky

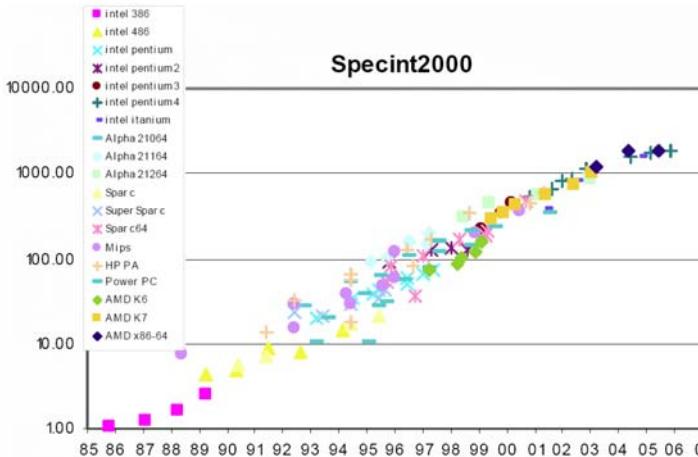
## Moorův zákon

1965 – Gordon Moore – počet tranzistorů na čipu se bude exponenciálně zvyšovat - zdvojnásobení každých cca 18 až 24 měsíců (a cena se sníží na polovinu) - stále platný



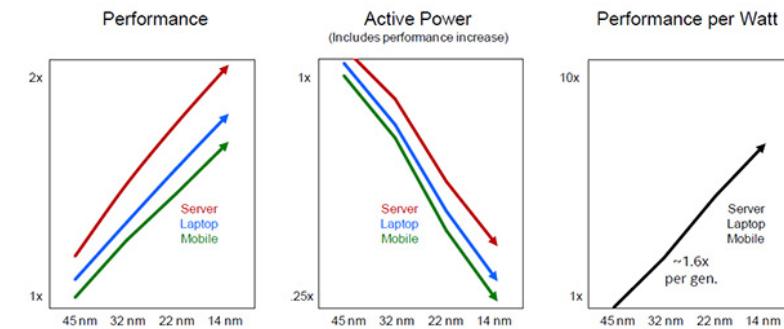
## Moorův zákon (modifikovaný)

zdvojnásobí se poměr výkon/cena



## Výkonnost nových technologií

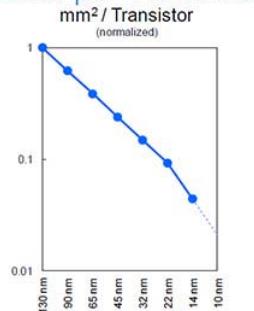
Výrazně se snižuje spotřeba energie a vzrůstá výkon vztažený na 1 W



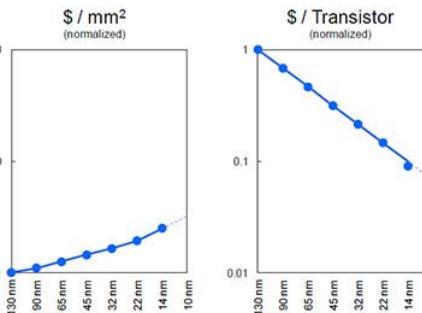
## Efektivnost nových technologií

Výrazně se zvyšují počty tranzistorů, ale klesá cena za tranzistor

Cost per Transistor



\$ / mm<sup>2</sup> (normalized)

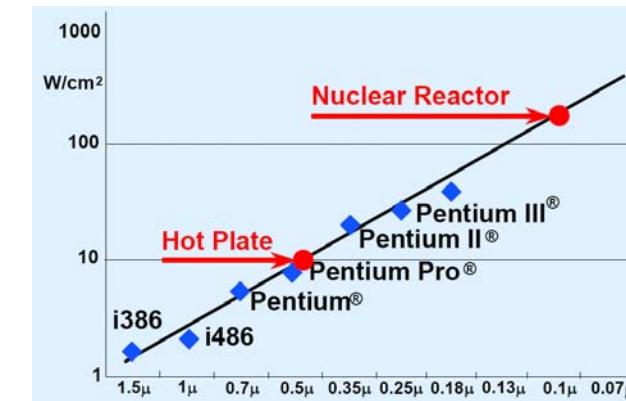


\$ / Transistor (normalized)

5

## Hustota výkonu

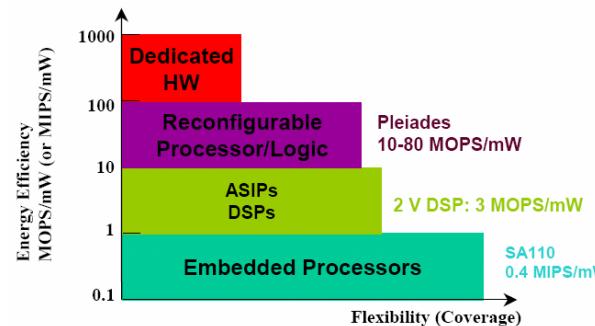
hlavní problém při zvyšování hustoty funkcí na čipu – nutno odvést ztrátové teplo z čipu (TDP – Thermal Design Power)



6

## Výpočetní (energetická) účinnost

Computational (energy) efficiency – počet operací (resp. instrukcí) za sekundu vztažený ke ztrátovému výkonu na daném čipu (energetická účinnost) – výhodnější je specializovaný HW

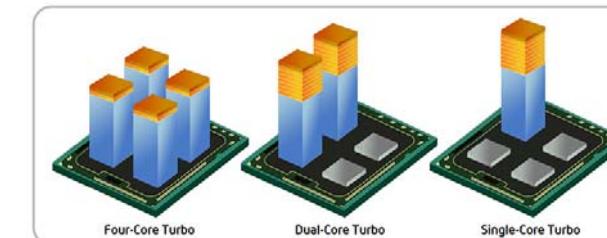


7

## Technologie Turbo Boost (Intel)

Technologie umožňující jednotlivým jádrům na vyšší frekvenci, pokud to okolnosti dovolují – závisí na počtu aktivních jader, na očekávané spotřebě a na teplotě procesoru (automaticky se zvyšuje násobič u jednoho či více jader).

Aktuální zejména při nerovnoměrném vytížení jednotlivých jader CPU (TB Max. 3.0 přetaktuje ještě více jen jedno jádro).

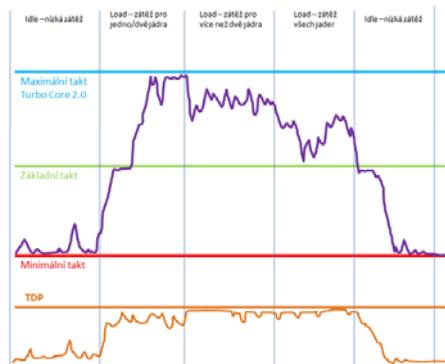


8

## Technologie Turbo Core (AMD)

Obdoba Turbo Boost pro AMD – změna frekvence jednotlivých jader (ne nutně všech) podle zátěže, aby se nepřekročil max. TDP

Při dobrém chlazení – XFR (Extended Frequency Range) – nad boost



9

## Technologie Anti-Theft (Intel)

HW podpora ochrany v CPU (+ placená služba) – nutná registrace; zablokování počítače při krádeži (uživatel musí aktivovat přes internet); sledování pohybu na základě lokace IP adresy; automatické zablokování, nedojde-li do určité doby k synchronizaci se serverem.

Po registraci si nainstalujeme SW (aktivace)

U CPU Core i3/i5/i7 (od Sandy Bridge), podpora v OS od Win 7



10

## Významné parametry procesorů

- velikost datové sběrnice ... 16, 32, 64 bitů;
- velikost cache L1 ... 8 – 128 kB (vztaženo na 1 jádro);
- velikost cache L2 ... 64 kB – 3 MB; 1 MB/jádro;
- velikost cache L3 ... 1 – 80 MB (většinou sdílená pro více jader);
- frekvence systémové sběrnice ... 1,6 – 3,2 GHz;
- frekvence jádra ... 1500 – 4500 MHz (v TB až 5,7 GHz);
- tepelný výkon ... 30 – 180 W;
- napětí jádra procesoru ... 0,9 – 2,5 V;
- maximální teplota jádra ... 70 – 95 °C;
- technologie výroby ... 45 nm, 32 nm, 22 nm, 14 nm, 10 nm, 7 nm;
- pouzdro (socket) ... 754, 775, 939, 1156, 1366, 2011, 2066, 4094 pinů

11

## Vývoj procesorů Intel

**Intel 8086** (1978), IBM PC XT, 16bitový, 2stupňová pipeline, skalární, CISC, 29 tisíc tranzistorů, 5-10 MHz, 0,33-0,66 MIPS, 3 µm, 20bitová AS (OP max. 1 MB), architektura „x86“.

**Intel 80286** (1982), IBM PC AT, 16bitový, 3stupňová pipeline, 134 tisíc tranzistorů, 8-25 MHz, 1,2-3,75 MIPS, 24bitová AS (OP max. 16 MB).

**Intel 80386** (1985), 32bitový, 3stupňová pipeline, 275 tisíc tranzistorů, 19-40 MHz, 6-15 MIPS, 1.5 µm, 32bitová AS (OP 4 GB), IA-32.

**Intel 80486** (1989), 32bitový, 5stupňová pipeline, 1,25 mil. tranzistorů, 25-50 MHz, 22-44 MIPS, 0,8 µm, 32bitová AS (OP 4 GB), sdružuje FPU, prvky RISC (nejčastější instrukce má přímo zahrnutý v log. obvodech - nepotřebuje pro ně mikroprogram)

12

## Intel Pentium 80586 + PRO

- 1993 – 3.1 mil. tranzistorů, 0.8 µm, 5 V, 60-66 MHz, 102-112 MIPS, L1 2x8 kB, dvoucestná, možnost multiprocesingu (2 CPU).  
 2 instrukce za takt (první superskalární procesor Intel);  
 32bitová AS, vnější datová sběrnice 64 bitů (vnitřně 32bitový), jednotka předvídání skoků.
- 1994 – 3.3 mil. tranzistorů, 0.6 µm, 3.3 V, 75-100 MHz, 120-160 MIPS.
- 1997 – 166-233 MHz, 0.28 µm, 4.5 mil. tranzistorů, L1 2x16 kB čtyřcestná.
- 1995 – **Pentium PRO**: 14násobná pipeline; 36bitová AS, 0.5-0.35 µm, 21 až 67 mil. tranzistorů, až 232 MIPS, provádění instrukcí mimo pořadí; spekulativní provádění instrukcí; přejmenovávání registrů; SMP max. 4 CPU,

13

## Další vývoj CPU Intel

- Intel Itanium** (2001) – plně 64bitový CPU; 10stupňová pipeline, technologie 0,18 µm, 800 MHz (málo vůči P4); výkon 13 Gflops; ztráta kompatibility z předešlými programy (nemůže nativně vykonávat instrukce x86, x87, MMX apod.) – velká instrukční sada by snižovala výkon; přidán překladač 32bitových instrukcí IA-32 na IA-64.

- Pentium Extreme Edition** (2005) – první dvoujádrový CPU Intel - v podstatě se tváří jako dva nezávislé procesory (je schopen provádět paralelně dvě sekvence instrukcí); 3.2-3.73 GHz, L1 2x16 kB, L2 2x1-2x2 MB, 90-65 nm, 230 mil. tranzistorů, 1.25 V, TDP 130 W; aplikace a OS musí dokázat využívat (vícevláknové aplikace).

15

## Intel Pentium II + III + 4

- Pentium II** (1997) – 233-450 MHz, 0.25 µm, 650 MIPS, L1 2x16kB, L2 256-512 kB;
- Pentium III** (1999) – 0.25-0.13 µm, 1.5-2 V, 450-1133 MHz, vícenásobná predikace větvění;
- Pentium 4** (2000) – 180-65 nm, 1.4-3.8 GHz, 42 mil. tranzistorů; 32bitový CPU, L1 8-16 kB, L2 256 kB až 1 MB; označení **NetBurst**; hyper zřetězení (20-31stupňová pipeline, výkon dosahován vysokými frekvencemi); L1 již neuchovává instrukce, ale mikroinstrukce (při opakování instrukce odpadá dekódování); mikrojádro s ALU pracuje na dvojnásobku nominální frekvence (velký TDP); technologie HT (Hyper-Threading) – u pozdějších variant P4.

14

## Mikro-architektury procesorů Intel

Postupné zvyšování výkonu a snižování spotřeby, podpora nových rozhraní, zvyšování frekvence pro RAM, zvyšování cache

- Core 2** (od 2006) – 64bitový, 65 nm, FSB, 1-2-4-6 jader, zkrácen pipelining (12-14 stupňový), snížení frekvence, snížení ztrátového výkonu (spojovalní řad pro PC a NB).

- Nehalem** (2008) – 45-32 nm, 750 mil. tranzistorů, nově QPI, DDR3, 2-4-8 jader, L1 2x32 kB, L2 256 kB/jádro, nově sdílená L3 8 MB, nové technologie Turbo Boost, integrace řadiče DDR3 a PCI-e do CPU, GPU na jiném čipu (1 pouzdro), řady i3/i5/i7, návrat k HyperThreadingu.

- Sandy Bridge** (2011) – 32 nm, výrazné vylepšení GPU (na 1 čipu), 2. generace i3/i5/i7; nové System Agent – obdoba North Bridge přímo v CPU (řadiče PCI-e 3.0, DMI, DDR3), Media Engine

16

## Mikro-architektury procesorů Intel

**Ivy Bridge** (2012) – 22 nm, 1,4 mld. tri-gate (3D) tranzistorů – snížení spotřeby (až 50 %), TDP 14-87 W, podpora PCI-e 3.0, DDR3L, USB 3.0, zvýšení výkonu GPU, 2-8 jader, 1.7-3.9 GHz.

**Haswell** (2013) – 22 nm, důraz na úsporu energie (TDP od 10 W), L3 2-20 MB, některé typy i L4 128 MB, 14-16stupňová pipeline, 2-18 jader, 4. generace i3/i5/i7.

**Broadwell** (2014) – 14 nm, důraz na nízkou spotřebu (pro mobilní zařízení – TDP od 3,5 W), zvýšení výkonu zejména u GPU, 5. generace i3/i5/i7.

**Sky Lake** (2015) – 14 nm, podpora DDR4, 6. generace i3/i5/i7

**Kaby Lake** (2016) – 14 nm, USB 3.1, 7. generace i3/i5/i7

**Coffee Lake** (2017) - 14 nm, 8. generace, Cannon Lake (10 nm)



## Mikro-architektury procesorů Intel

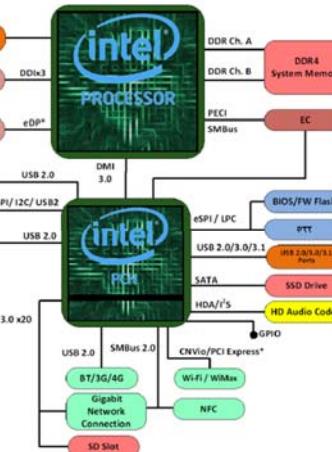
**Coffee Lake Refresh** (2018)  
14 nm, 9. generace i5/i7/i9

**Comet Lake** (2019), 14 nm,  
**Ice Lake**, 10 nm,  
10. generace i5/i7/i9

**Rocket Lake, Tiger Lake**  
(2020), 11. generace

**Alder Lake** (2022), 12. generace

Rozrůstá se množství kódových  
názvů pro desktopové a  
mobilní aplikace vyráběné stále  
se vylepšujícími technologiemi.



## Core i9-13900K (Intel)

Jeden z nejvýkonnějších CPU pro PC (září 2022):

- 24 jádrový (8 výkonných a 16 úsporných),  
frekvence 2.2/3.0 GHz, Boost 4.3/5.8 GHz;
- CPU 13. generace, Raptor Lake, 64bitová architektura;
- technologie Hyper-Threading (až 32 vláken současně);
- L2 cache 32 MB (celkem), L3 cache 36 MB;
- integrovaná PCI Express 5.0/4.0;
- GPU UHD Graphics 770 (podpora rozlišení 8K);
- podpora DDR5/DDR4 (5600/3200 MHz), max. 128 GB;
- technologie 10 nm; TDP 125 (max. 253 W), socket LGA 1700;



## Vývoj procesorů AMD

**Am286** (1984) – licence od Intel (požadavek IBM), kompatibilní s 80286

**Am386** (1991) – konkurence Intel, levnější

**Am486** (1993) – snaha o výkonnější varinty vůči Intel,  
později snížení nap. napětí na 3.3 V a zvýšení frekvence (ztráta  
kompatibility, rozdílné základní desky)

**K5** (1996) - superskalární procesor (konkurent Intel Pentium),  
0.5-0.35 µm, 4.3 mil. tranzistorů, až 120 MIPS, 75-166 MHz

**K6** (1997) – 0.35-0.25 µm, 8.8 mil. tranzistorů, 166-300 MHz,  
až 290 MIPS, FPU 8-17 MFlops, FSB 60-66 MHz;

**K6-2** (1998) – 266-550 MHz, 0.25 µm, 9.3 mil. tranz., FSB až 100 MHz;  
3DNow! – rozšíření o 21 instrukcí zrychlujících vytváření 3D scén.

**K6-3** (1999) - 21.3 mil. tranzistorů, 4cestná L2 256 kB, až 490 MIPS.



## Vývoj procesorů AMD (desktopy)

**K7** (1999) – Athlon, Athlon XP, Athlon MP, Duron, Sempron

**K8** (2003) – nová technologie x86-64 – plně kompatibilní s 32bitovou IA32 i 16bitovou x86, ale méně výkonná oproti IA-64; Athlon 64, Athlon 64 X2 (první vícejádrový CPU od AMD – 2005), Turion 64, Turion 64 X2, Sempron;

**K10 (Stars, Phenom, Opteron)** (2007) – 65 nm, 2-4 jádra, frekvence 1.8-2.6 GHz, L2 512 kB, L3 2 MB,

**Phenom II** (2008) – 45 nm, frekvence 2.5-3.6 GHz, 2-6 jader,

**Bulldozer** (2011) – 32 nm, L2 3-8 MB, L3 8MB, 4-16 jader, frekvence až 3.6/4.2 GHz, TDP 10-100 W, turbo core 2.0; na čipu CPU i GPU,

**Piledriver** (2012) – 32 nm, až 4.7/5.0 GHz, TDP 17-220 W, sc. AM3+;

21

## Vývoj procesorů AMD (desktopy)

**Streamroller** (2014) – 28 nm;

**Excavator** (2015) – 28 nm, DDR4;

**Zen (Ryzen, Threadripper, Epyc)** (2016) – 14 nm, až 32 jader, sběrnice Infinity Fabric,

**Zen 2 (EPYC 2)** (2018) – 14 nm a 7 nm (složen z více čipů), i 128bitové varianty, až 64 jader (128 vláken).

**Zen 3** (2020) – 7 nm, podpora DDR4 i DDR5, řady Ryzen 5, 7, 9.

**Zen 4** (2022) – 5 nm, podpora DDR5

Množství kódových jmen (Naples, Rome, Milan, ...); nejvýkonnější varianty spíše pro serverové aplikace (TDP i přes 200 W).

22

## RYZEN Threadripper 2990WX (AMD)

Jeden z nejvýkonnějších CPU AMD (pro PC)

- 32 jádrový (64 vláken), frekvence 3,0/4,2 GHz;
- 64bitová architektura;
- PCI Express 3.0 (až 64 linek);
- technologie simultánní multithreading (SMT)
- L2 512 kB/jádro, L3 cache 64 MB;
- výrobní technologie 12 nm;
- 4kanál. DDR4 (až 2933 MHz)
- socket TR4 (LGA, 4094 pinů);
- TDP 250 W.



23

## Trendy ve vývoji

- na jednom čipu CPU i GPU (APU – Accelerated Processing Unit);
- integrace grafických procesorů s vysokým rozlišením (4-8K);
- na čipu více jednodušších procesorů/jader a zlepšení komunikace mezi nimi, kombinace různých jader na čipu;
- důraz na energetickou efektivitu (v závislosti na výkonu), podpora mobilních procesorů (pro ultrabooky a tablety);
- hardwarové funkce pro ochranu dat (Intel Anti-Theft);
- hardwarová podpora vzdálené administrace (bez nutné podpory OS, propojeno s integrovanou grafickou kartou);
- nové instrukce pro standard pokročilého šifrování dat (Advanced Encryption Standard) – včetně dešifrování;
- hardwarová virtualizace – podpora více virtuálních počítačů v CPU.

24

## Spojení CPU a FPGA

Do jednoho pouzdra (SoC) se sdružují CPU (nejčastěji hardwarová jádra) společně s hradlovými poli FPGA (spojení architektur založených na instrukcích s paralelně orientovanými architekturami).

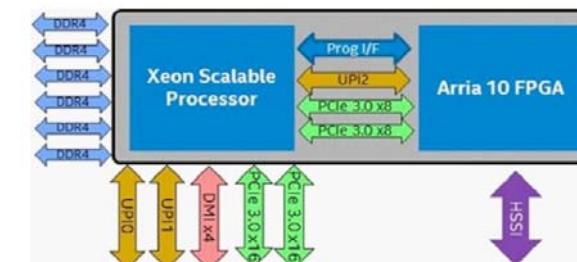


- FPGA mohou rychleji zvládnout paralelní výpočty;
- v FPGA lze vytvořit vhodné vstupně-výstupní periferie (s velkou šírkou pásma, nestandardní typy či větší množství);
- náklady (na vývoj i sériovou výrobu) bývají obvykle vyšší než samotné CPU (výkonnostní přínosy to musí vyvážit);
- speciální hardware je většinou energeticky účinnější (výpočet něčeho v přepočtu na energii) než univerzální architektura CPU.

25

## Hybridní CPU Xeon s FPGA (Intel)

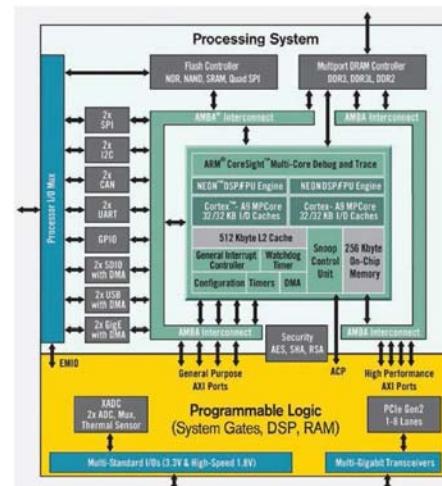
Hybridní spojení CPU Intel Xeon (Sky Lake, 20 jader, 2 GHz, TDP 125 W) s FPGA Arria 10 (70 W);  
vše vzájemně spojeno rychlou sběrnicí UltraPath Interconnect (UPI) s rychlosí až 10,4 GT/s (Giga Transfers per Second).



26

## ZYNQ-7000 (Xilinx)

Spojení 2jádrového procesoru ARM s jádrem Cortex-A9 a hradlového pole;  
- frekvence CPU 720 MHz  
- 85 tis. log. elementů



27

# Paralelní víceprocesorové systémy

Milan Kolář

Ústav mechatroniky a technické informatiky



Projekt ESF CZ.1.07/2.2.00/28.0050  
 Modernizace didaktických metod  
 a inovace výuky technických předmětů.

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

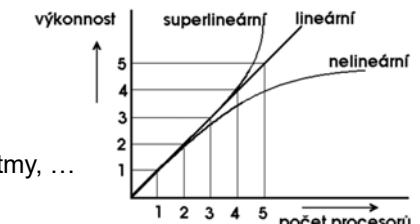
## Paralelní víceprocesorové systémy

Jako paralelní označujeme systém, v němž může probíhat několik procesů současně (paralelně)

- snaha zvyšovat výkonnost nad hranici danou technologií výroby součástek (mikroprocesorů);
- rozměry, cena a energetická náročnost elektronických prvků klesá rychleji než roste jejich rychlosť  $\Rightarrow$  vytváření paralelních víceprocesorových systémů.

Výkonnost většinou neroste lineárně s počtem procesorů (má spíše logaritmický průběh)

- vlivem komunikace CPU, synchronizace, nedokonalým vytížením, nevhodnými algoritmy, ...
- od jisté hranice je přidávání procesorů nerentabilní.



2



## Paralelismus

Paralelní systémy dále členíme podle formy (granularity, zrnitosti) paralelního procesu – zajímají nás procesy od určitého stupně složitosti (např. systémy pracují paralelně s celými slabíkami obvykle jako paralelní neoznačujeme).

Pro paralelní systémy je charakteristické, že procesory jsou obvykle soustředěny v poměrně malém prostoru (minimální časové ztráty při přenosu informací) – v opačném případě mluvíme o *rozloženém (distributed) systému* (počítačová síť).

Architektuře počítačového systému by měl být přizpůsobený i řešený *algoritmus*; sériové algoritmy se snažíme transformovat na paralelní (např. sériové procházení nezávislých cyklů lze přidělit více procesorům). Naopak pro víceprocesorové systémy jsou vhodné operace s vektory, maticemi či aritmetika vícemístných čísel.



## Ukazatele paralelizace algoritmů

Algoritmy posuzujeme obdobně jako HW počítače, např.:

- 1) *Časová složitost* (time complexity)  $O(n,p)$  - funkce, jejíž hodnota je pro konkrétní algoritmus úměrná maximální době jeho výpočtu.
- 2) *Zrychlení* (speedup)  $S(n,p)$  - poměr doby výpočtu nejlepšího známého sekvenčního algoritmu a doby výpočtu paralelního algoritmu na témaž (paralelním) počítači, využíváme-li  $p$  procesorů.
- 3) *Paralelní efektivita* (efficiency)  $E(n,p)$  - jedná se o zrychlení dělené počtem použitých procesorů.

Příklad výpočtu algoritmu pro součet  $n$  čísel:

- sekvenční algoritmus:  $O(n) = n$  (doba je lin. závislá na počtu čísel)
- paralelní součet prováděný na  $p = n/2$  procesorech:  $O(n,n/2) = \log n$
- zrychlení:  $S(n,n/2) = n / \log n$
- paralelní efektivita:  $E(n,n/2) = 2 / \log n$



## Amdahlův zákon

Udává maximální zrychlení pro víceprocesorové systémy:

$$S \leq \frac{T(n)}{T(n, p)} = \frac{1}{f_s + \frac{f_p}{p}} \leq \frac{1}{f_s} \quad f_s = \frac{t_s}{t_s + t_p}$$

$f_s$  ... část výpočtu, která musí být provedena sekvenčně

$f_p$  ... část výpočtu, kterou lze paralelizovat

$p$  ... počet procesorů

Předpokládá se:

- při výpočtu sekvenční části se ostatní CPU nevyužívají;
- velikost řešeného problému se při přidávání procesorů nemění
- zrychlení je shora omezeno ( $1/f_s$ ).

5

## Gustafsonův zákon

Udává maximální zrychlení pro paralelní systémy:

$$S \leq \frac{T(n)}{T(n, p)} = f_s + p \cdot (1 - f_s) \quad f_s = \frac{t_s}{t_s + \frac{t_p}{p}}$$

$f_s$  ... část výpočtu, která musí být provedena sekvenčně

$f_p$  ... část výpočtu, kterou lze paralelizovat

$p$  ... počet procesorů

Předpokládá se:

- za konstantní se považuje doba běhu (ne rozsah úlohy);
- GZ udává, kolikrát déle by trval výpočet bez paralelizace;
- s velkým počtem CPU se bude zrychlení neustále zvyšovat.

6



## Zdroje neefektivnosti

- a) nedostatek užitečné práce pro daný počet procesorů,
- b) velké komunikační náklady,
- c) příliš velká režie synchronizace procesorů,
- d) špatná distribuce práce (nerovnoměrné rozdělení práce) mezi procesory.

Možná řešení:

- **technologické** (rychlejší komunikační HW a komunikační režie, překrývání komunikačních a výpočetních operací);
- **algoritmické** (dobré mapování algoritmy na paralelní architekturu, volba vhodné zrnosti úkolů, rovnoměrné rozdělení výpočetní zátěže mezi CPU, volba komunikace,...).

7

## Klasifikace paralelních systémů

zavedl v roce 1966 J. M. Flynn (podle toku instrukcí a dat):

- **SISD** (Single Instruction stream, Single Data stream),
- **MISD** (Multiple Instruction stream, Single Data stream),
- **SIMD** (Single Instruction stream, Multiple Data stream),
- **MIMD** (Multiple Instruction stream, Multiple Data stream).

příliš hrubé dělení, někdy se používají varianty:

- **MSIMD** (Multiple SIMD) – systém, v němž pracuje několik pod systémů SIMD nezávisle na sobě;
- **SPMD** (Single Program, Multiple Data stream) – modifikace SIMD, kde všechny procesory sice provádějí stejný program (dělí se o instrukce), ale nezávisle na sobě (bez synchronizace).

8

## Klasifikace paralelních systémů

Podle způsobu řízení systému:

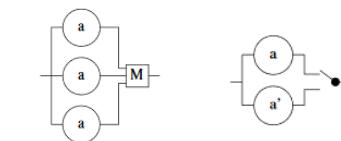
- *řízené tokem instrukcí* (dané pořadím instrukcí)
  - SIMD – s lokální nebo globální pamětí; vektorové a maticové procesory, příp. VLIW;
  - MIMD – volně nebo těsně vázané;
- *řízené tokem událostí* (k provedené operaci dochází v okamžiku, kdy to okolnosti umožňují nebo vyžadují)
  - řízení tokem dat (spustí se instrukce, jsou-li potřebné operandy), nemají čítač instrukcí;
  - řízení tokem požadavků (výsledek se počítá, je-li požadován);
- *bez centrálního řízení* (např. jednoúčelové systolické systémy, které jsou obvodově přizpůsobeny výpočetnímu algoritmu nebo neuronové sítě), MISD (několik procesů zpracovává jedny data).

9

## Zálohované systémy

Zálohované systémy jsou takové systémy, ve kterých jsou důležité části použity v nadbytečném (redundantním) počtu.

Nejsou určeny pro zvýšení výkonnosti systému, ale pro eliminaci poruch. Člení se na systémy s vysokou bezpečností a výše spolehlivé systémy (např. zálohování majoritou, zálohování přepínáním).



10

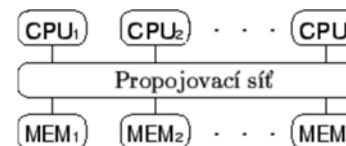
## Těsně vázané systémy

**Těsně vázané systémy** – paralelní systémy, v nichž procesory nejsou vybaveny lokální pamětí nebo je tato paměť velmi malá – pokud jsou všechny CPU stejněho typu a mají rovnocenný přístup k paměti, mluvíme o *symetrickém multiprocesoru* (Symmetric Multi-Processor, SMP).

*Propojovací síť* lze propojit libovolný CPU s libovolným paměťovým modulem.

Velké nároky na rozsah a rychlosť komunikace mezi CPU.

Sdílená paměť

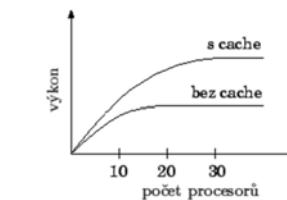


11

## Těsně vázané systémy

Při použití většího množství procesorů již nedochází k nárůstu výkonu  
⇒ použití cache paměti u procesoru.

Vzniká ale problém konzistence dat uložených v několika cache současně – řeší se např. metodou přímé signalizace změn (zrušení platnosti kopí ve všech cache) nebo rozdělením dat na taková, které lze a která nelze přesouvat do cache (přesouvají se jen data, která se nemění).



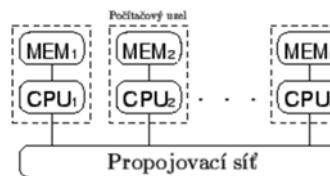
12

## Volně vázané systémy

Volně vázané systémy – každý procesor v systémech s distribuovanou pamětí je vybaven velkou lokální pamětí (a často i svými periferními zařízeními)

- každý CPU má značnou autonomii,
- někdy se tyto systémy označují jako *multipočítací*,
- velká rychlosť komunikace s lokální pamětí,
- počet CPU není v principu omezen,
- není problém s konzistencí dat v cache,
- komunikace formou zpráv,
- slabá interakce mezi CPU.

(masivně paralelní počítače - „clustery“ s rychlou propojovací sítí)



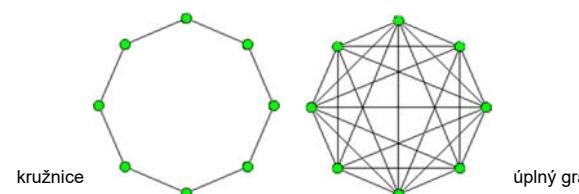
13

## Statické propojovací sítě

Strukturu obvykle vyjadřujeme grafem (uzel odpovídá modulu, hrana odpovídá komunikačnímu spoji).

Důležité vlastnosti:

- průměr grafu (největší vzdálenost dvou uzlů v grafu) – má vliv na celkovou rychlosť systému;
- stupeň uzlu (počet hran příslušejících uzlu) – určuje počet komunikačních adaptérů procesoru (souvisí s cenou).



15

## Propojovací sítě

Slouží ke komunikaci mezi jednotkami; dělí se na:

- *statické sítě* – neměnné, vhodné pro méně složité sítě, používají se zejména u volně vázaných systémů;
- *dynamické sítě* – obsahují spínací prvky - řídí se buď *globálně* (centralizované řízení, společný řadič, propojování okruhů, vhodné pro občasné velké objemy dat) nebo *lokálně* (distribuované řízení, propojování paketů, vhodné pro velký počet malých objemů)
  - křížové přepínače,
  - sítě typu promíchání s výměnou,
  - válcové posouvače,
  - sběrnice.

14

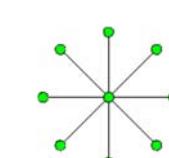
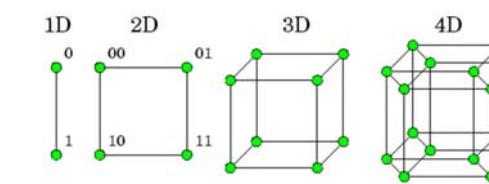
## Statické propojovací sítě

## Statické propojovací sítě

*Scalability* (rozšiřitelnost) – vyjadřuje, jak se změní složitost komunikace, přidáme-li další uzly.

Nejčastější topologie:

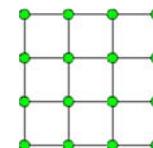
- kubická síť ( $n$ -rozměrná krychle)
  - každý vrchol má  $n$  sousedů (uzlově symetrické),
- binární strom (uzlově nesymetrické),
- hvězda (uzlově nesymetrické).



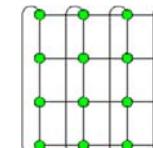
16

## Statické propojovací sítě

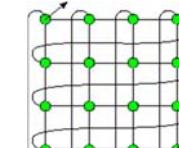
- čtvercová síť (mříž),
- trojúhelníková síť,
- válec,
- anuloid, torus, síť typu Illiac



čtvercová síť



válec

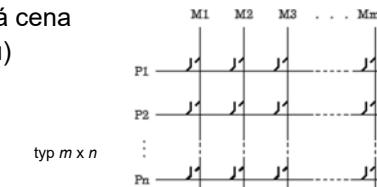


síť typu Illiac

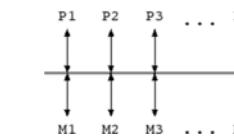
17

## Dynamické propojovací sítě

**Křízový přepínač** – informace prochází pouze jedním přepínačem, počet dvojic vstup-výstup není omezen, nevýhodou je vysoká cena (velký počet spínačů)



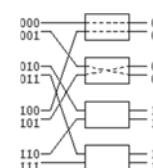
**Sběrnice** – může komunikovat každý s každým, v jednom okamžiku pouze jeden přenos, - omezená propustnost.



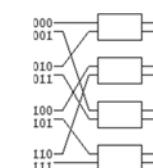
18

## Dynamické propojovací sítě

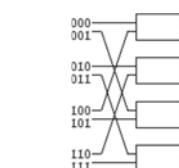
**Síť typu promíchání výměnou** – je tvořena různě propojenými elementárními přepínači, nejčastěji typu  $2 \times 2$  (mají dva pracovní stavy – identita a výměna); spojuje se několik přepínačů za sebou.



Dokonalé promíchání



Dokonalé oddělení



Propojení typu motýlek

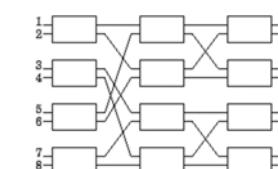
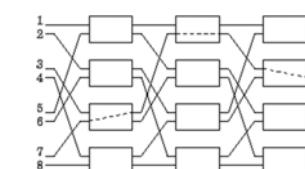
19

## Dynamické propojovací sítě

**Blokující sítě** – požadované spojení může být blokováno existujícími spojeními.

**Přestaviteľné sítě** – lze vždy realizovat všechna propojení, avšak někdy pouze za cenu přestavění stávajících spojení.

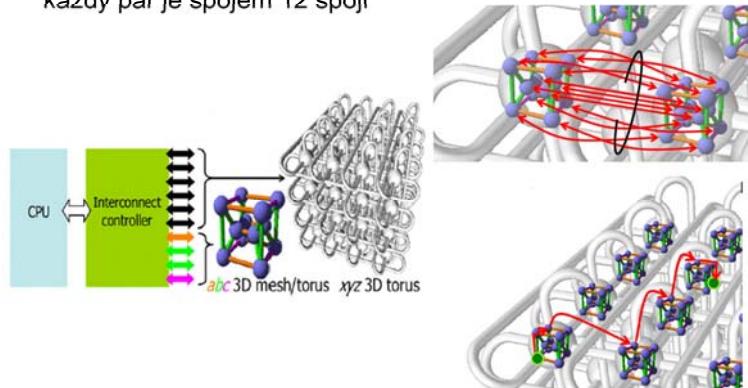
**Neblokující sítě** – umožňují vytvořit spojení libovolného vstupu s libovolným výstupem bez ohledu na to, jaká jiná spojení již jsou uskutečněna.



20

## TOFU propojovací síť'

TOFU (torus/fusion) – 6D mesh/torus topologie,  
každý uzel obsahuje 6+4 spojů (každý o rychlosti 10GB/s),  
každý pár je spojen 12 spoji



21

## Systolická pole

Multiprocesorový systém MISD (několik procesorů zpracovává jednu data), procesory komunikují vždy pouze s nejbližším sousedem v propojené síti, každý systolický prvek (SP) využívá pouze svou lokální paměť a vykonává stále stejnou úlohu (netřeba ředit);  
- SP může obsahovat buď kompletní CPU nebo dekodér instrukcí + procesní jednotku nebo jednoúčelovou pevně zadávanou buňku.

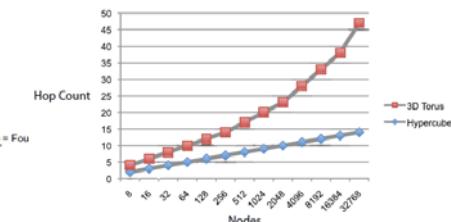
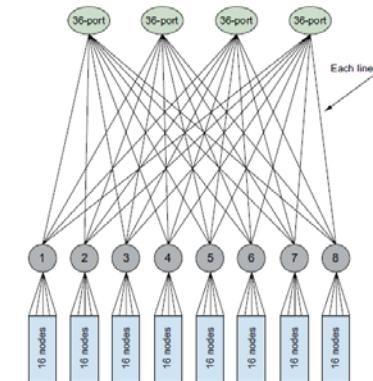
Vlastnosti:

- strukturou proudí vstupní data i mezičlánky (různými směry i rychlosťmi) – na rozdíl od zřetězeného zpracování;
- buňky jsou jednoho nebo několika málo typů;
- kombinace intenzivní lokální komunikace a výpočtu;
- nejčastěji mají dimenze 1 nebo 2.

23

## Propojovací síť' Hypercube

Stromová topologie Hypercube  
(InfiniBand)



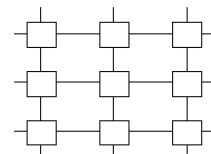
22

## Systolická pole

- Jednoúčelová – zadávané systolické prvky ušití na míru specifické aplikaci (může být i na jednom čipu);
- Univerzální – mohou se adaptovat na řadu aplikací rekonfigurací;
- Programovatelná – každá procesní jednotka je programovat.

Využití: manipulace s datovými strukturami nebo řetězci znaků, matematické operace, maticová aritmetika, zpracování obrazové informace, rozpoznávání řeči nebo obrazu.

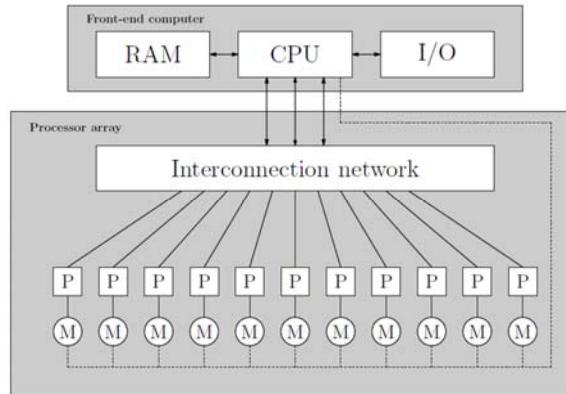
Systolickou síť' není nutné ředit programem.



24

## Procesorová pole

Skládají se z plnohodnotného počítače a několika jednoduchých výpočetních jednotek vybavených vlastní malou lokální pamětí.



25

## Procesorová pole (pokr.)

Hlavní CPU distribuuje při paralelním výpočtu data do lokálních pamětí a pošle výpočetním jednotkám instrukce k provedení.

- všechny jednotky provádějí ten samý kód (je možné je jen „vymaskovat“);
- mezi výsledky se pak sloučí (tzv. redukce);
- výhodou je nízká cena výpočetních jednotek;
- tato architektura je vhodná jen pro některé typy úloh (numerická matematika, vektorové výpočty, zpracování obrazových dat apod.);
- většinou jednouživatelské systémy;
- s rozvojem VLSI ztrácí na významu (nahrazeny nesymetrickými, či poslední dobou symetrickými multiprocesory).

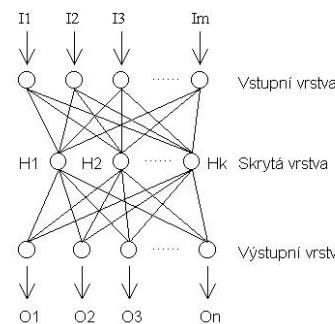
26

## Neuronové sítě (počítače)

Základním prvkem jsou neurony propojené do sítě (paralelní dynamický systém), mají 2 fáze činnosti:

- učení - nastaví se chování (váhy, práh neuronů),
- vybavování - aktivní činnost podle naučeného chování.

- nemusí se programovat
- výhodné u struktur s neznámým algoritmem



27

## Počítačové clustery

Seskupení volně vázaných počítačů, které spolu úzce spolupracují (navenek může pracovat jako jeden počítač)

- obvykle propojeny rychlou datovou sítí (např. Myrinet, cLAN, Gigabitový Ethernet, ATM, InfiniBand) pro zasílání zpráv mezi procesy, izolované od vnější sítové infrastruktury;
- volnější sdružení počítačů se nazývá *farma*.

Většinou jeden uzel stojí nad ostatními (master node) a provádí např. rozdělování úkolů (uzly clusteru nemají připojené displeje).

Typický cluster využívá „open source“ OS (např. Linux) – na všech uzlech identický OS.

Cluster se z hlediska uživatele (aplikativního programátora) jeví jako jeden velmi výkonný a homogenní počítač.

28

## Vlastnosti clusterů

- většinou sestaveny ze „standardních“ počítačů (x86);
- příznivý poměr ceny a výkonu;
- snadná rozšiřovatelnost;
- většinou homogenní uzly, mohou být i heterogenní;
- odolnost vůči výpadku, spolehlivost;
- některé uzly mohou být pasivní (záložní);
- nižší využitelnost výkonu (nižší využití procesorů);
- náročnější správa;
- úzká místa ve formě sdílených komponent (např. komunikační přepínač);
- vhodné pro paralelizovatelné slabě vázané procesy, které nepotřebují častou synchronizaci či výměnu výsledků.

29

## Typy clusterů (podle funkce)

**Výpočetní cluster** (high performance computing - HPC)  
– pro zvýšení výpočetního výkonu.

**Cluster s vysokou dostupností** (high availability, failover)  
– zajištění nepřetržité poskytování nějaké služby (při výpadku uzlu přebírá práci jiný uzel).

**Cluster s rozložením zátěže** (load balancing, scalable)  
– službu poskytuje několik počítačů, požadavky jsou zasílány na uzly podle jejich aktuálního vytížení.

**Úložný cluster** (storage) – zprostředkovává přístup k diskové kapacitě (většinou rozložena mezi více počítačů).

**Gridové clustery** – rozlehlé clustery zabezpečené tak, aby komunikace mohla probíhat v nechráněném prostoru internetu (základem jsou certifikáty uživatelů a počítačů) - heterogenní.  
Funkce clusterů se ve skutečnosti prolínají.

30

## Clustery pracovních stanic

Clustery většinou chápeme lokálně, ale propojovací síť může být např. i internet (COW – Clusters of Workstations, NOW – Network of Workstations).

Na stanicích je spuštěn *démon PVM* (Parallel Virtual Machine), na jednom z počítačů je spuštěn *hlavní proces*, který posílá démonům úkoly

- COW umožňuje heterogenost stanic (HW i SW),
- z hlediska programátora se COW jeví jako paralelní počítač s distribuovanou pamětí,
- minimální pořizovací náklady,
- nevýhodou je zejména nízká komunikační kapacita a její velká režie (časová náročnost),
- obtížné ladění paralelních aplikací,
- malá využitelnost instalovaného výkonu.

31

## Nejvýkonnější cluster na světě

**Superpočítač Frontier** – (Top500 – listopad 2022), výrobce HPE, 8,73 mil. jader (9.472 CPU AMD EPYC 64C (každý 64 jader) + 37.888 GPU Radeon Instinct MI250X (220 jáder); výkon 1.102 PFlops, spotřeba 21,1 MW; OS HPE Cray; síť HPE Slingshot-11; USA (Oak Ridge, Tennessee).



32

## Nejvýkonnější clustery v Ostravě

Národní superpočítacové centrum IT4Innovations, VŠB-TU Ostrava, [www.it4i.cz](http://www.it4i.cz) (2013 – Anselm, 2015 – Salomon, 2019 – Barbora, 2021 – nejvýkonnější superpočítač Karolina).

**Karolina** – 2021 (85. pořadí TOP500), výrobce HPE, výkon 15,7 Pflops, úložiště z rychlých flash disků 1,4 PB, vysokorychlostní síť Infiniband HDR 200 Gb/s.



33

## Nejvýkonnější cluster ve Škoda Auto

Jaro 2020 – Mladá Boleslav, datové centrum Škoda Auto; nejvýkonnější firemní superpočítač, výrobce HPE, výkon 3,2 PFLOPS, síť Infiniband 100 Gb/s.



35

## Nejvýkonnější cluster v ČR

Jaro 2019 – ČVUT v Praze, Karlovo náměstí, Výzkumné centrum informatiky (RCI), 41,6 mil. Kč

Heterogenní systém se skládá z 20 výpočetních uzlů se 480 CPU jader Intel Xeon Gold, 12 výpočetních uzlů NVIDIA GPU Supermicro, jednoho uzlu Lenovo ThinkSystem SR950 se 192 CPU jádry; výkon přes 6 PFLOPS; vysokorychlostní propojovací síť Infiniband EDR (100 Gb/s), rychlé SSD disky.



34

## Nejvýkonnější cluster na FM TUL

**Charon** (dokončen na jaře 2019), celkem 24 uzlů po 2 x 10-core Intel Xeon Silver 4114 2,2 GHz (480 jader), RAM 2,3 TB (4,8 GB na jádro), SSD 480 GB na uzel, diskové pole 64 TB, propojovací síť Omni-Path (InfiniBand od Intelu), pripojen do Metacentra ČR.

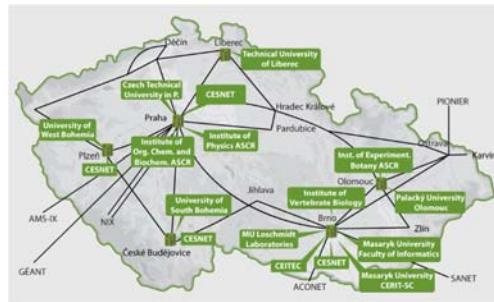
36

## Metacentrum

Virtuální organizace sdružující výpočetní a úložné kapacity vědecko-výzkumných institucí ČR

více na <https://metavo.metacentrum.cz>

Uživatelem se může stát každý zaměstnanec nebo student akademické instituce ČR (členství je zdarma).



# Komunikace procesoru se vstupně-výstupními zařízeními

Milan Kolář  
Ústav mechatroniky a technické informatiky



Projekt ESF CZ.1.07/2.2.00/28.0050  
Modernizace didaktických metod  
a inovace výuky technických předmětů.

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

2

## Komunikace CPU s I/O zařízeními

Vstupně výstupní brána (I/O port, V/V řadič) – zprostředkovává předávání dat mezi sběrnicí počítače a periferním zařízením.

V/V periferie jsou obecně pomalé oproti CPU, součinnost se realizuje těmito základními principy:

- 1) *přímou programovou obsluhou (pomocí sběrnic a I/O bran)*
- 2) *obsluhou s přerušením (interrupt, trap)*
- 3) *komunikací prostřednictvím DMA (kanály)*
- 4) *specializovanými V/V procesory (kanály)*

**Kanály** - schopné realizovat velké množství V/V operací (odlehčení procesoru).

## 1) Přímá programová obsluha

Styk CPU s periferiemi řízen programovými prostředky. Program prostřednictvím stavové vstupní brány postupně testuje, zda a která V/V zařízení jsou připravena vyslat, příp. přjmout data.

Při zjištění připravenosti procesor vyvolá V/V podprogram (*ovladač – driver*), který zajistí vstup nebo výstup dat; ovladač zajišťuje řídicí signály pro danou periferii.

Ve většině případů však používání přímé programové obsluhy počítač zdržuje, neboť je zatěžován periodickým testováním stavů (i když nedochází k přenosům dat).

Vhodná pro relativně rychlá zařízení.

## Přímá programová obsluha

V programu jsou instrukce vstupu nebo výstupu a komunikace s periferním zařízením synchronizovány:

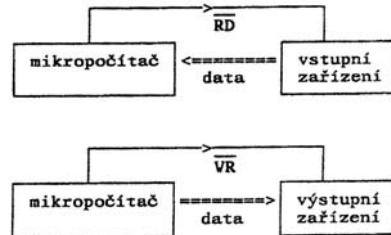
- vkládáním čekacích taktů
- pomocí tzv. *handshake* (před instrukcí V/V je vložena smyčka zjišťující, zda je zařízení schopno přjmout, resp. poskytnout informaci).

Programové řízení je výhodné z hlediska obvodového řešení. Pro zmenšení zátěže CPU se část stykové úlohy přenáší z programu na obvody, příp. komunikační procesor.

## Technika nepodmíněného V/V

Jednoduché, ale předpokládá, že periferní zařízení je stále připraveno komunikovat (pro rychlé periferie).

Při vstupu vyšle procesor bitový signál RD (Read), čímž vstupní zařízení předá data, při výstupu vyšle procesor signál WR (Write) a výstupní zařízení převeze data.

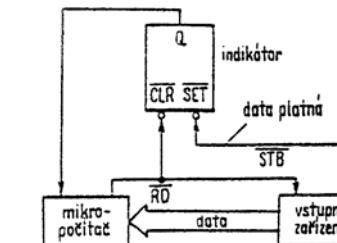


5

## Technika podmíněného vstupu

Jednosměrný korespondenční (neúplný) režim – o zahájení nebo ukončení přenosu je informován pouze μP a vysílač dat je povinen data udržovat.

Má-li vstupní zařízení platná data, nastaví pomocí signálu STB (Strobe) výstup klopného obvodu (Q = 1); μP následně impulsem RD převeze data a nuluje indikátor Q.

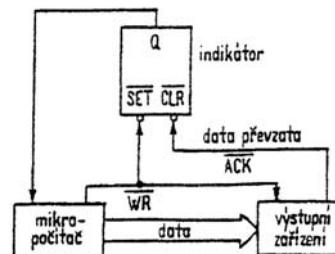


6

## Technika podmíněného výstupu

Opět jednosměrný korespondenční (neúplný) režim

Má-li μP platná výstupní data, vyšle impuls WR, který současně nastaví indikátor Q. Výstupní zařízení po převzetí dat impulsem ACK (Acknowledge) nuluje indikátor Q. Tím může μP vyslat další data.

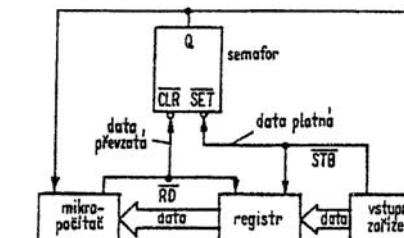


7

## Úplný podmíněný vstup

Obousměrný korespondenční (úplný) režim – využívá registr (vyrovňávací paměť) a klopný obvod pracující jako semafor, který je testován vysílačem i přijímačem dat.

Pokud je registr plný, semafor je ve stavu 1 a vstupní zařízení dáší data nevyšle. Je-li prázdný (Q = 0), lze signálem STB do registru vyslat data, čímž se semafor nastaví do 1. Při Q = 1 může μP impulsem RD data převzít a vynulovat Q.

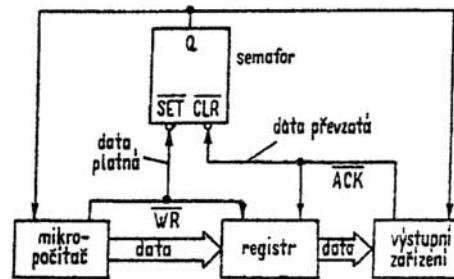


8

## Úplný podmíněný výstup

Opět obousměrný korespondenční (úplný) režim

Při výstupu μP testuje semafory. Je-li nulový, impulsem WR vyšle data do registru a nastaví semafory do stavu 1. Výstupní zařízení tak zjistí, že v registru jsou data a může je impulsem ACK převzít a semafory vynulovat.



9

## 2) Přerušení

„nestandardní“ událost v systému (vně i uvnitř procesoru), která způsobí přerušení právě probíhajícího programu (bez ohledu na právě prováděné místo v programu se dokončí pouze právě prováděná instrukce); procesor začne zpracovávat program obsluhy přerušení a po jeho dokončení se vrátí na přerušené místo.

Vhodné pro zařízení (resp. události), jejichž rychlosť (resp. četnost) je výrazně nižší než rychlosť procesoru (počítač nemusí jednotlivá zařízení, resp. události, periodicky testovat).

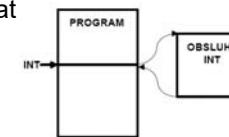
(IRQ – Interrupt Request – žádost o přerušení)

10

## Činnost při přerušení

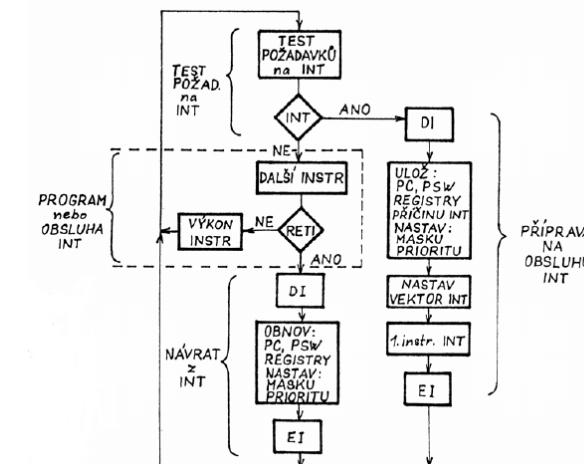
Objeví-li se žádost o přerušení, tak procesor:

- dokončí rozpracovanou instrukci
- znemožní přijetí dalších žádostí o přerušení (např. nulováním interního indikátoru, DI – Disable Interrupt)
- vyšle signál o akceptování požadavků (INTA)
- určí, který podprogram se bude vykonávat
  - a) vnučení adresy
  - b) vnučení instrukce
  - c) vnučení přerušovacího vektoru
- zpracuje se podprogram přerušení
- v podprogramu přerušení se opět povolí přerušení (instrukce EI – Enable Interrupt)
- nakonec se vrátíme na přerušené místo hlavního programu.



11

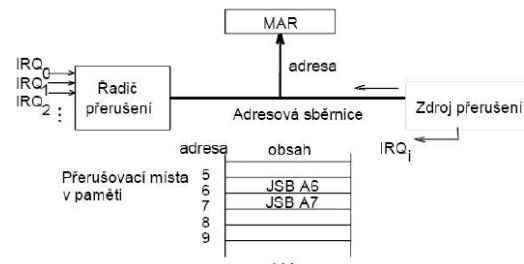
## Obsluha přerušení



12

## a) Vnucení adresy

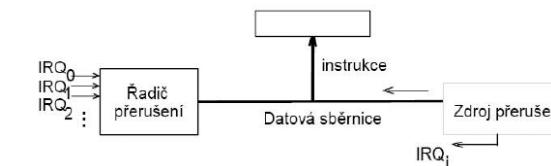
Každý zdroj je identifikován adresou, kterou je schopen při přerušení zaslat procesoru – do registru MAR (Memory Address Register) - následující instrukce pak bude načtena z této adresy



13

## b) Vnucení instrukce

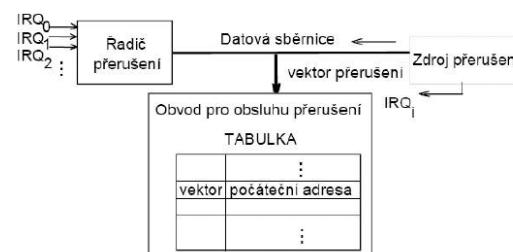
Jednotka vyžadující přerušení (příp. řadič přerušení) zašle procesoru po datové sběrnici instrukci, která se má provést (např. RST n, CALL n)



14

## c) Vnucení přerušovacího vektoru

Každý zdroj přerušení generuje přerušovací číslo, zvané *vektor přerušení*, které je ukazatelem do tabulky počátečních adres přerušovacích podprogramů (*tabulka vektorů přerušení*); nejčastější (většinou od adresy 0 v OP, např. 256 x 4B = 1kB).



15

## Typy přerušení

### a) Hardwarové

- *vnitřní* (od procesoru) – je vyvoláno chybou při provádění strojové instrukce (dělení nulou, přetečení, výpadek stránky cache, neplatná instrukce, apod.)
- *vnější* (od periferií vně procesoru) – nejčastěji od řadiče přerušení, V/V zařízení (IRQ n se označují linky od periferií k procesoru), od časovače aj. Procesor tyto vstupy neustále testuje (při každé instrukci).

Podle reakce na vnější přerušení:

- *maskovatelné* (INTR) – je možné dočasně zakázat (např. nastavením bitu IF – Interrupt Flag);
- *nemaskovatelné* (NMI – Non-Maskable Interrupt) zejména pro havarijná situace (výpadek napájení).

16

## Typy přerušení (pokračování)

### b) Softwarové (programové)

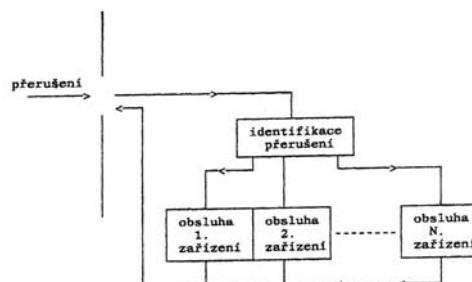
- přerušení je vyvoláno instrukcí volání přerušení umístěnou přímo v programu (volání služeb OS), (např. INT n – volá přerušovací podprogram n).
- Vesměs jde o nepřímé volání podprogramů; má vysokou (nejvyšší) prioritu.

Z hlediska časování:

- **asynchronní** přerušení (nejčastěji hardwarová vnější);
- **synchronní** přerušení (s během programu) – nejčastěji softwarová a hardwarová vnitřní) např. INT, TRAP.

## a) Programová identifikace

K přerušovacímu vstupu procesoru je připojen signál logického součtu externích signálů přerušení; po akceptování žádosti o přerušení se vyvolá přerušovací podprogram, který programově identifikuje přerušující zařízení (cyklické výzvy – *pooling*); podle pořadí lze určit i prioritu.



## Řešení priorit

Při větším počtu zdrojů žádostí o přerušení vzniká problém:

- počtu přerušovacích vstupů (zdrojů),
- priorit jejich obsloužení.

Více zdrojů přerušení lze řešit třemi způsoby:

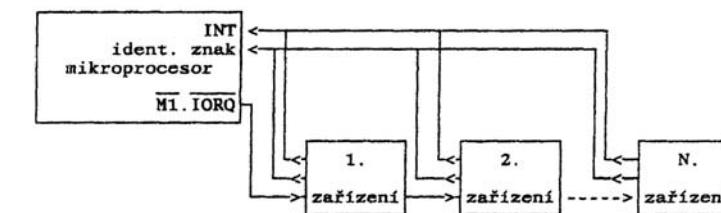
- programovou identifikací,
- sériovou obvodovou identifikací,
- řadičem přerušení (interrupt controller).

Každé přerušení má stanovenou prioritu - pro případ současně žádosti více komponent;

- standardní přerušení (řeší se po dokončení instrukce, během níž vznikl požadavek);
- výjimka vysoké úrovni (řeší se během provádění instrukce – nelze ji dokončit).

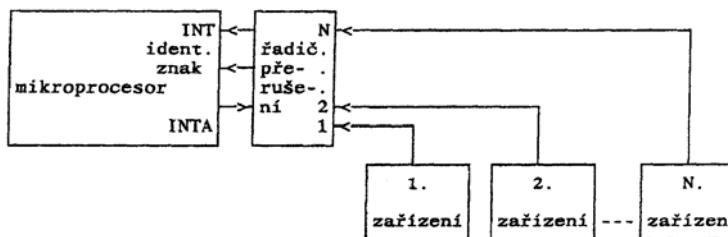
## b) Sériová obvodová identifikace

Procesor po příchodu přerušovacího signálu očekává identifikační znak generovaný stykovými obvody, který jej nasměruje do žádoucího podprogramu. Stykové obvody jsou spojeny sériově v pořadí priority (neuplatní se přerušení nižší priority) – *zřetězení (Daisy Chain)*.



## c) Řadič přerušení

Specializovaný stykový obvod, který soustřeďuje všechny žádosti ze všech přerušovacích zdrojů; periferní zařízení vysílají pouze požadavky na obsluhu do řadiče přerušení. Řadič požádá o přerušení i v případě, že nový požadavek má vyšší úroveň než právě probíhající.

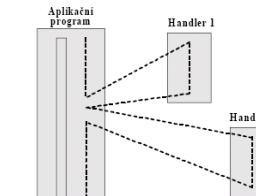


21

## Vícenásobná přerušení

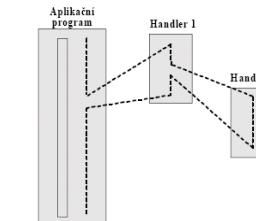
### Sekvenční zpracování

- během obsluhy jednoho přerušení se další požadavky nepřijímají (pozdržují se);
- jednoduché, ale ne vždy vhodné.



### Vnořené zpracování

- přijímají se přerušení s prioritou vyšší než je obsluhovaná priorita



22

## 3) DMA

### Direct Memory Access – přímý přístup do paměti

Metoda kopírování bloků dat mezi pamětí a portem (HDD, pamětí, ...) bez průchodu skrz procesor, bez dočasného ukládání těchto dat v pomocných registrech

- rychlejší,
- méně náročné na výkon.

Teoreticky přenos dat např. ze V/V portu do paměti nelze provést přímo (kvůli jedné adresové sběrnici) - je třeba nejprve data načíst do pomocného registru CPU a odtud (třeba ještě v rámci téže instrukce) do paměti (třeba min. dva cykly sběrnice)  
⇒ řešíme využitím DMA řadičů.

23

## Řadič DMA

### Musí umět:

- generovat adresy pro paměť,
- generovat příslušné řídicí signály,
- žádat CPU o uvolnění paměti (zabránění kolize v paměti),
- po ukončení činnosti vrací řízení sběrnic procesoru.

V praxi většinou DMA řadiče tvořeny několika samostatnými kanály vyhrazenými určitému zařízení, které umí adresovat.

Řadič potřebuje ke své činnosti 3 údaje:

- kolik dat má přenést,
- kterým směrem,
- ze/do kterého místa paměti (tj. od jaké počáteční adresy).

24

## Činnost řadiče DMA

- Postup: 1) zařízení zažádá o přímý přenos dat svůj řadič (při výstupu dat zažádá DMA řadič sám procesor);
- 2) DMA řadič požádá procesor o přidělení sběrnic;
- 3) po akceptování CPU vyšle DMA řadič adresu paměti na adresovou sběrnici a vygeneruje příslušné řídící signály - pro paměť (R/W) i zařízení;
- 5) zařízení vyšle data na datovou sběrnici, resp. začne číst data z této sběrnice;
- 6) po ukončení přenosu to zařízení oznámí řadiči;
- 7) DMA řadič vrátí sběrnice procesoru.

DMA řadič není řízen instrukcemi – je to jednoúčelové HW zařízení.

25

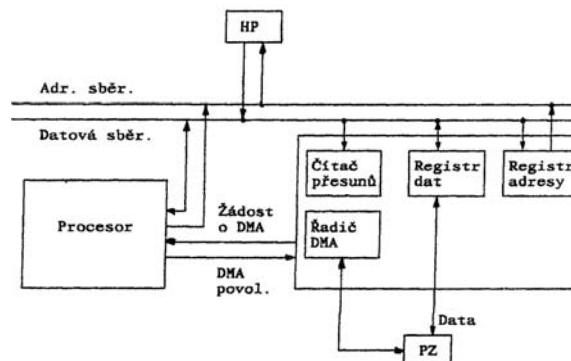
## Způsoby řízení přenosu DMA

- 1) CPU převede své sběrnice do neutrálního stavu, jejich řízení převeze řadič DMA, procesor nepokračuje v běžné činnosti.
- 2) Odpojí se generování hodinových signálů pro CPU, odpojí se jeho budiče sběrnic (nejčastěji na jediný dílčí přenos).
- 3) Dílčí přenosy se uskutečňují v časových intervalech, v nichž procesor pracuje, ale nekomunikuje po sběrnících (neovlivňuje rychlosť CPU); je třeba přesně synchronní činnost CPU a řadiče DMA.

26

## Spolupráce DMA - CPU

HP – hlavní paměť                    PZ – periferní zařízení



27

## 4) Specializované V/V procesory

Mezi procesor a řadič periferie se vkládá další jednotka (kanál), což je specializovaný V/V procesor schopný samostatně řídit V/V operace.

Používá se u rozsáhlejších počítačových systémů, které mohou mít i více kanálů (i více CPU).

Kanál je obdobou DMA, ale je více samostatný:  
 je řízen posloupností vlastních instrukcí (tzv. kanálovým programem)  
 - selektorový kanál (v jednom okamžiku může obsluhovat jen jedno zařízení, vhodné pro rychlé PZ);  
 - multiplexní kanál (může obsluhovat několik PZ současně, rozdělen na podkanály).

28

## Samostatné kanály

HP – hlavní paměť

PZ – periferní zařízení

