IV. Počítače a sítě

Update: 3. května 2018

1 Architektura univerzálních procesorů. Principy urychlování činnosti procesorů.

Architektura počítačů je náčrt struktury a funkčnosti systému. Je charakterizována výčtem **registrů** a jejich funkcí, vnitřních a vnějších **sběrnic**, způsobem **adresování** a **instrukčním souborem**.
Registr je malé úložiště dat v mikroprocesoru s rychlým přístupem, které slouží jako **pracovní paměť** během výpočtů.
Sběrnice je soustava vodičů pro **přenos informací** mezi více účastníky na principu „jeden vysílá, ostatní přijímají." Podle typu přenášené informace je dělíme na *datové*, *adresové* a *řídící*. V praxi však díky multiplexu může jít o jedny dráty.

1.1 Procesory CISC a RISC

V dnešní době se ustálilo dělení počítačů do dvou základních kategorií podle typu používaného procesoru:

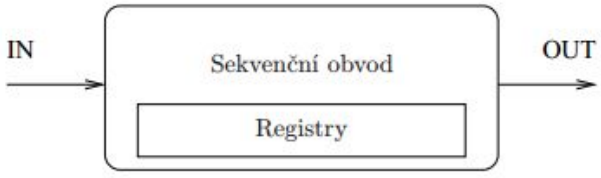
- **CISC** – počítač se složitým souborem instrukcí (*Complex Instruction Set Computer*)
- **RISC** – počítač s redukovaným souborem instrukcí (*Reduced Instruction Set Computer*)

1.1.1 CISC

- procesory s **komplexním instrukčním souborem**
- instrukce mají **proměnlivou délku** i **dobu vykonání**
- **vysoká složitost instrukcí** → nutný systematický návrh řadiče procesoru
- vykonání strojové instrukce probíhá posloupností mikrooperací (předepsána mikroinstrukcí v řídící paměti)
- procesor obsahuje relativně **nízký počet registrů**
- operace provedená i **složenou instrukcí** (např. násobení) může být **nahrazena** sledem jednodušších strojových instrukcí (*sčítání a bitové posuny*) → mohou být ve výsledu vykonány rychleji, než hardwarově implementovaná složená varianta

– Označení CISC bylo zavedeno jako **protiklad** až poté, co se prosadily procesory RISC, které mají instruční sadu naopak maximálně redukovanou (pouze jednoduché operace, tj. žádné složené, jsou stejně dlouhé a jejich vykonání trvá stejnou dobu).

– Obvyklou chybou je domněnka, že procesory CISC mají více strojových instrukcí, než procesory RISC. Ve skutečnosti nejde o absolutní počet, ale o počet různých druhů operací, které procesor sám přímo umí vykonat na hardwarové úrovni (tj. již z výroby). Procesor CISC tak může například paradoxně obsahovat jen jednu strojovou instrukci pro danou operaci (např. *logické operace*), zatímco procesor RISC může tuto operaci obsahovat jako několik strojových instrukcí, které stejnou operaci umí provést nad různými registry.



Obrázek 1: Procesor (CISC) jako sekvenční obvod

Krok		Význam
1.	**VI**	Výběr Instrukce
2.	**DE**	Dekódování
3.	**VA**	Výpočet Adresy
4.	**VO**	Výběr Operandu
5.	**PI**	Provedení Instrukce
6.	**UV**	Uložení Výsledku

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}
VI	I_1						I_2						...
DE		I_1						I_2					
VA			I_1						I_2				
VO				I_1						I_2			
PI					I_1						I_2		
UV						I_1						I_2	

Tabulka 4: Postup provádění instrukcí procesorem CISC

1.1.2 RISC

– počet instrukcí a způsobů adresování je malý, ale zůstává úplný, aby bylo možno provést vše → v tomhle se liší od CISC

– instrukce jsou vytvořeny pomocí obvodu → jednodušší na výrobu než CISC

– širší sběrnice, rychlejší tok instrukcí a dat do procesoru

– instrukce jen nad registry

– navýšený počet registrů → delší program

– instrukce mají **jednotný formát** – délku i obsah

– komunikace s pamětí pouze pomocí instrukcí **LOAD / STORE**

– **každý strojový cyklus znamená dokončení jedné instrukce**

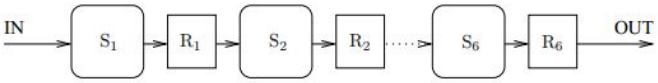
– používá se **zřetězené zpracování instrukcí**

– řešení problémů s frontou instrukcí

– mikroprogramový řadič může být nahrazen rychlejším obvodem

– přenášejí složitost technologického řešení do programu (překladače)

– představitelé *ARM, MOTOROLA 6800, INTEL i960, MIPS R6000*



Obrázek 2: Zřetězené zpracování (v procesoru RISC)

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
VI	I_1	I_2	I_3	I_4	I_5	I_6	I_7	...				
DE		I_1	I_2	I_3	I_4	I_5	I_6	I_7				
VA			I_1	I_2	I_3	I_4	I_5	I_6	I_7			
VO				I_1	I_2	I_3	I_4	I_5	I_6	I_7		
PI					I_1	I_2	I_3	I_4	I_5	I_6	I_7	
UV						I_1	I_2	I_3	I_4	I_5	I_6	I_7

Tabulka 5: Zřetězené provádění instrukcí procesorem RISC

1.2 Von Neumannovo schéma počítače

John Von Neumann definoval v roce **1945** základní koncepci počítače (EDVAC) **řízeného obsahem paměti**. Od té doby se objevilo několik odlišných modifikací, ale v podstatě se **počítače v dnešní době** konstruují podle tohoto modelu. Ve svém projektu si von Neumann stanovil určitá kritéria a principy, které musí počítač splňovat, aby byl použitelný univerzálně. Můžeme je ve stručnosti shrnout do následujících bodů:

- Počítač se skládá z paměti, řídící jednotky, aritmeticko–logické jednotky, vstupní a výstupní jednotky.

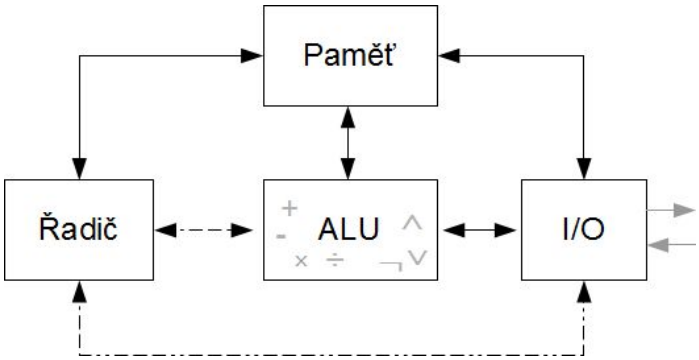
  – **ALU** - aritmeticko–logická jednotka (aritmetic–logic unit) - jednotka provádějící veškeré aritmetické výpočty a logické operace. Obsahuje sčítačky, násobičky a komparátory.

  – **Operační paměť** - slouží k uchování zpracovávaného programu, zpracovávaných dat a výsledků výpočtu

  – **Řídící jednotka** - řídí činnost všech částí počítače. Toto řízení je prováděno pomocí řídících signálů, které jsou zasílány jednotlivým modulům. Řadiči jsou pak zpět zasílané stavové hlášení. Dnes řadič spolu s ALU tvoří jednu součástku, a to procesor neboli CPU (Central Processing Unit).

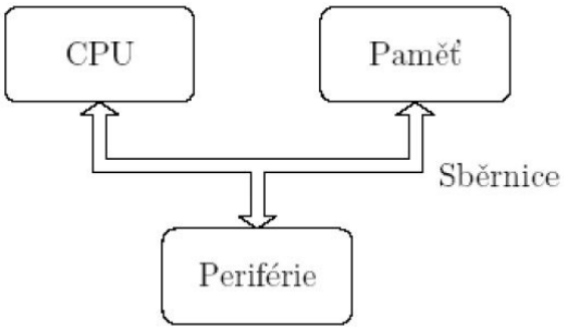
  – **Vstup/ Výstup** – zařízení určené pro vstup dat, a výstup zpracovaných výsledků.

- Struktura pc je **nezávislá na typu řešené úlohy** (univerzálnost), počítač se programuje obsahem paměti.
- Následující krok počítače je závislý na kroku předešlém.
- **Instrukce** a **data** jsou v téže paměti.
- Paměť je rozdělena do **paměťových buněk stejné velikosti (Byte)**, jejichž pořadová čísla se využívají jako adresy.
- Program je tvořen posloupností instrukcí, které se vykonávají jednotlivě v pořadí, v jakém jsou zapsány do paměti.
- Změna pořadí prováděných instrukcí se provádí **skokovými instrukcemi** (podmíněné nebo nepodmíněné skákání na adresy).
- Čísla, instrukce, adresy a znaky se značí v **binární soustavě**.



Nevýhody Von Neumannovy koncepce ve srovnání s dnešními PC

- Podle von Neumannova schématu počítač pracuje **vždy nad jedním programem**. Toto vede k velmi špatnému využití strojového času. Dnes je obvyklé, že počítač **zpracovává paralelně více programů** zároveň - tzv. **multitasking**
- Počítač může mít i více jak jeden procesor.
- Podle Von Neumanova schématu mohl počítač pracovat pouze v tzv. **diskrétním režimu**, kdy byl do paměti počítače zaveden program, data a pak probíhal výpočet. V průběhu výpočtu již nebylo možné s počítačem dále interaktivně komunikovat.
- Dnes existují **vstupní/výstupní** zařízení, např. pevné disky a páskové mechaniky, které umožňují vstup i výstup.
- Program se do paměti nemusí zavést celý, ale je možné zavést pouze jeho část a ostatní části zavádět až v případě potřeby.



Výhody

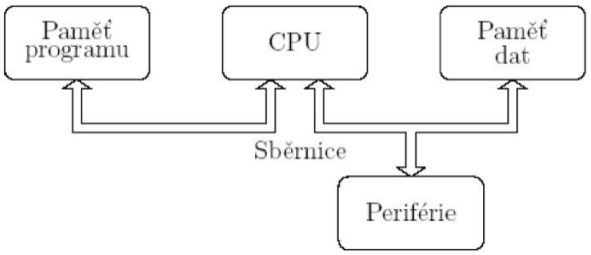
+ **Rozdělení paměti** pro kód a data určuje programátor, řídící jednotka přistupuje pro data i instrukce jednotným způsobem.
+ **Jedna sběrnice** -> jednodušší levnější výroba.

Nevýhody

– **Společné uložení dat a kódu** může mít za následek přepsání vlastního programu
– **Jedna sběrnice** je omezující.

1.3 Hardvardské schéma počítače

Několik let po von Neumannovi, přišel vývojový tým odborníků z Harvardské univerzity s vlastní koncepcí počítače, která se sice od Neumannovy příliš nelišila, ale odstraňovala některé její nedostatky. V podstatě jde pouze o **oddělení paměti pro data a program**. Abychom si mohli obě koncepce porovnat, můžeme vycházet ze zjednodušených schémat.



Výhody

+ **Program se nepřepíše** (oddělené paměti pro data a program).
+ Dvě sběrnice umožňují **paralelní** načítání instrukcí a dat.
+ Paměti mohou být vyrobeny **odlišnými technologiemi** a každá může mít jinou nejmenší adresovací jednotku (8 bitů pro instrukce a 8, 16 nebo 32 pro data).

Nevýhody

- 2 sběrnice mají **vyšší nároky na vývoj** řídící jednotky a jsou také dražší a složitější na výrobu.
- Paměť je **rozdělena** už od **výrobce**.
- Nevyužitou část dat **nelze využít** po program a obráceně.

1.4 Principy urychlování činnosti procesorů

Techniky urychlování výpočtu v hardwaru:

- speciální kódování dle potřeby dané úlohy
- speciální výpočetní jednotky dle potřeby dané úlohy (FFT – rychlý fourierova transformace)
- paralelní zpracování (násobné výpočetní jednotky)
- zřetězové zpracování instrukcí (pipelining)

1.4.1 Paralelní zpracování

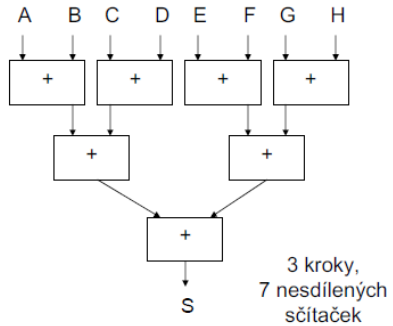
Zpracování více elementárních úloh běží současně.

Př. S = A + B + C + D + E + F + G + H

```
SW: sekvenčně

S = A + B;
S = S + C;
S = S + D;
S = S + E;
S = S + F;
S = S + G;
S = S + H;

7 kroků,
1 sdílená
sčítačka
```



1.4.2 Zřetězené zpracování instrukcí (pipelining)

Princip zřetězení se značně překrývá s principy procesorů RISC. Základní myšlenkou je **rozdělení zpracování jedné instrukce** mezi různé části procesoru a tím i dosažení možnosti **zpracovávat více instrukcí** najednou. Pro dosažení tohoto zřetězení je nutné rozdělit úlohu do posloupnosti dílčích úloh, z nichž každá může být vykonána **samostatně**, např. oddělit načítaní a ukládání dat z paměti od provádění výpočtu instrukce a tyto části pak mohou běžet souběžně. To znamená že musíme osamostatnit jednotlivé části sekvenčního obvodu tak, aby každému obvodu odpovídala jedna fáze zpracování instrukcí. Všechny fáze musí být **stejně časově náročné**, jinak je rychlost **degradována** na nejpomalejší z nich. Fáze zpracování je rozdělena minimálně na 2 úseky:

- **Načtení** a **dekódování** instrukce.
- **Provedení** instrukce a případné uložení výsledku.

Zřetězení se stále vylepšuje a u novějších procesorů se již můžeme setkat stále s více řetězci rozpracovaných informací (více pipelines), dnes je standartem 5 pipelines.

Problém

Největší problém spočívá v **plnění zřetězené jednotky**, hlavně při provádění podmíněných skoků, kdy během stejného počtu cyklů se vykoná více instrukcí. U pipelingu se instrukce následující po skoku vyzvedává dřív, než je skok dokončen. **Primitivní implementace** vyzvedává vždy **následující instrukci**, což vede k tomu, že se vždy mýlí, pokud je skok nepodmíněný. Pozdější implementace mají **jednotku předpovídání skoku (1bit)**, která vždy správně **předpoví nepodmíněný skok** a s použitím cache se záznamem předchozího chování programu se pokusí předpovědět i cíl podmíněných skoků nebo skoků s adresou v registru nebo paměť. V případě, že se predikce nepovede, bývá nutné vyprázdnit celou pipeline a začít vyzvedávat instrukce ze správné adresy, což znamená relativně **velké zdržení**. Související problémem je přerušení.

Plnění fronty instrukcí

Pokud se dokončí skoková instrukce, která odkazuje na jinou část kódu, musejí být instrukce za ní zahozeny (*problém plnění fronty instrukcí*)

- u malého zřetězení **neřešíme**
- používání bublin na vyprázdnění pipeline, **naplněníní prázdnými instrukcemi**
- **predikce skoku** – vyhrazen jeden bit předurčující, zda se skok provede či nikoliv
  - **Statická** – součást instrukce $\rightarrow$ řeší programátor nebo kompilátor
  - **Dynamická**
    - **jednobitová** – zaznamenává jestli se skok provedl, či ne (1/ 0)
    - **dvoubitová** – metoda zpožděného skoku $\rightarrow$ v procesoru řeší se např. tabulkou s 4 kB instrukcí

Zřetězené zpracování přináší urychlení výpočtu nejen v procesorech, ale i jiných číslicových obvodech (např. pro zpracování obrazu, bioinformatických dat apod.). Pokud použijeme zřetězené zpracování, musíme dodat řadu podpůrných obvodů a řešit řadu nových problémů. Moderní procesory používají kromě zřetězení i další koncepty:

- **superskalární architektura** (zdvojení) – když nastane podmíněný skok, začnou se vykonávat instrukce obou variant, nepotřebná část se pak zahodí. Tento způsob, pak vyžaduje vyřešit ukládání výsledku.

- **VLIW procesory** – má více ALU – tzn. může zároveň dělat více operací → k tomu složí dlouhé instrukce.
- **vektorové procesory** – je navržený tak, aby dokázal vykonávat matematické operace nad celou množinou čísel v daném čase. Je opakem skalárního procesoru, který vykonává jednu operaci s jedním číslem v daném čase.
- **multivláknové procesory**

2 Základní vlastnosti monolitických počítačů a jejich typické integrované periférie. Možnosti použití.

3 Struktura OS a jeho návaznost na technické vybavení počítače.

4 Protokolová rodina TCP/IP.

5 Metody sdíleného přístupu ke společnému kanálu.

6 Problémy směrování v počítačových sítích. Adresování v IP, překlad adres (NAT).

7 Bezpečnost počítačových sítí s TCP/IP: útoky, paketové filtry, stavový firewall. Šifrování a autentizace, virtuální privátní sítě.