

# 31 APO

**Architektura počítače. Koncepce a techniky CPU. Porovnání přístupů RISC a CISC procesorů. Sítě procesorů a paralelní architektury. Hierarchický koncept pamětí. Přerušovací a vstupně-výstupní podsystém počítače, řízení vstupů a výstupů**

## 1 Architektura počítače

Klasický von Neumannovský počítač má:

- Procesor
  - Řadič
  - ALU (Arithmetic-Logical Unit)
- Paměť
- I/O Systém

Řadič se dále skládá z části datové s vlastní řídicí částí. Datovou lze navíc rozdělit na registry a další obvody.

## 2 Základní cyklus počítače

Důležité registry:

- Program Counter (PC)
- Instruction Register (IR)
- Stack Pointer (SP)

1. Inicializace registrů apod

2. Čtení instrukce

- PC → adresa Hlavní Paměti (HP)
- Čtení obsahu
- Uložení přečtených dat do IR
- Inkrementace PC o délku instrukce

3. Přečtení opcode (kód instrukce - typicky několik prvních bitů v závislosti na velikosti instrukční sady)

4. Provedení instrukce - zahrnuje i další věci jako čtení operandu

5. Zpracování případného přerušení

6. Skok na bod 2

## 3 Přerušení

Jsou dva druhy přerušení a u obou dochází k přerušení sekvence vykonávaného kódu a přechází se na obsluhu.

**Vnější přerušení** asynchronní obsluha vnější události, např. reakce OS na event ze vstupní periferie.

**Vnitřní přerušení (výjimka)** přímo od CPU, které takto reaguje na problém při zpracování instrukce, např. dělení nulou

**Synchronní soft. přerušení** záměrné přerušení vzniklé vložením patřičné instrukce na místo v kódu.

## 4 Instrukční sady

Každá instrukce se skládá z opcode a operandu. Operandy mohou být registry (resp. jejich "indexy"), adresa (např. pro skoky).

### 4.1 RISC

Reduced Instruction Set Computer je architektura CPU, ve které je velice omezena instrukční sada (IS) např. jen na sčítání, bitové posuny, nějaké čtení/ukládání atd. Zbytek se realizuje softwarově. Všechny instrukce mají pevný formát a délku a také stejnou dobu vykonání. Velikost IS taky implikuje potřebu mnohem menšího počtu tranzistorů, což je mimo jiné důvod, proč třeba ARMy moc nežerou. Zástupci: ARM, MIPS, PowerPC.

### 4.2 CISC

Complex Instruction Set Computer je architektura (prozatím) většiny uživatelských CPU. Instrukční sada bývá často poměrně velká, instrukce nemusí být stejně dlouhé a jejich vykonání může trvat různě dlouho. Zástupci: x86, amd64.

## 5 Sítě procesoru a paralelní architektury

Existují 4 typy struktur procesoru:

**Single instruction single data (SISD)** Klasický procesor tj. jednoduchý tok instrukcí i dat (používá se ve většině pc s von Neumannovou архитектурou)

**Single instruction multiple data (SIMD)** Procesorové pole - jednoduchý tok instrukcí, vícenásobný tok dat. např. GPU

**Multiple instruction single data (MISD)** Pipeline

**Multiple instruction multiple data (MIMD)** Multiprocesorové pole

### 5.1 Propojovací sítě

Zajišťují propojení a komunikaci mezi procesory v architekturách s více procesory. Dělí se na statické a dynamické, přičemž ve statických jsou spoje neměnné a v dynamických naopak. U dynamických jsou spojovací spínače řízeny buď lokálně, kdy má každá skupina svůj řadič, nebo centrálně, kdy existuje jen jeden.

Statické propojovací sítě jsou: lineární, stromová, kruhová, mřížová, hvězdicová, krychlová a polygonální (úplný graf).

**Skalární procesory** běžné procesory

**Vektorové procesory** provádí jednu operaci nad několika operandy, tím se odstraňuje režie spojená s indexováním jednotlivých prvků vektoru.

**Paralelizované SISD** jsou systémy postavené na architektuře VLIW, zálohované systémy a systémy používající pipelining.

**Very Long Instruction Word (VLIW)** je architektura s dlouhými instrukcemi, které jsou rozděleny na několik částí a zpracovávány paralelně. Jednotlivé exekuční jednotky jsou propojeny, operační paměť je rozdělena mezi exekuční jednotky.

**Zálohované systémy** v sobě mají několik SISD, které všechny provádí stejný výpočet nad stejnými daty, přičemž výsledek všech je porovnáván. Cílem je zvýšení spolehlivosti a bezpečnosti.

**Paralelní systémy SIMD** jsou ty, ve kterých se zpracovávají dobře rozdělitelná data. Ideálním případem jsou matice. Princip práce spočívá v současném zpracování několika prvků, kdy procesory z procesorového pole provádějí synchronně stejnou instrukci. Procesory jsou řízeny společným řadičem.

**SIMD s lokální pamětí** Procesorové pole (PP) je řízeno univerzálním počítačem/procesorem, který zpracovává nadřazený program (rozhoduje o maticových úlohách a zabezpečuje přenos dat na procesory v poli)

- Řadič PP sám zpracovává skalární a řídicí instrukce, zatímco vektorové nechává zpracovat PP
- Každý procesor má svou vlastní paměť operandu
- Procesory si mezi sebou posílají data přes propojovací síť

**SIMD se sdílenou pamětí** Na rozdíl od SIMD s lokální pamětí je v tomto případě paměť od procesoru oddělena a komunikace probíhá přes propojovací síť

- Přidělování paměti do procesoru zajišťuje řadič
- Počet paměťových modulů může být jiný než počet procesorů

**Paralelní systémy MIMD** Každý procesor zpracovává instrukce a data svého vlastního programu.

**Těsně vázané** Procesory mají sdílenou operační paměť, která je oddělená od procesoru a připojena spolu s periferiemi na propojovací síť, přes kterou komunikují s CPU

- Každý procesor má malou vyrovnávací paměť pro data
- Periferie mají malou autonomii
- Propojovací síť umožňuje libovolné propojení

**Volně vázané** Procesory mají vlastní (lokální) paměť a vlastní periferie, přičemž lokální paměť obsahuje jak program, tak data

- Propojovací síť bývá statická
  - Hierarchická organizace sběrnic - procesory nejnižší úrovně spolu s pamětmi seskupeny do clusteru, které jsou připojeny komunikačními moduly na sběrnice vyšší hierarchie
  - Organizace do n-rozměrné krychle (nebo mřížky) - každý procesorový modul má 8(4) komunikačních procesorů pro připojení. Části krychle lze dynamicky přidělovat pro různé úlohy. Je vyžadován nadřazený počítač.
- Řízení je složitější než u volně vázaných sítí, ale na druhou stranu jsou tyto systémy odolnější vůči poruchám a výpočty lze navíc zněkolikanásobit podle potřeby. Jsou používány tam, kde je potřeba spolehlivost.

**NUMA (Non-Uniform Memory Access)** Používá se u MIMD systémů, kde má zajistit kratší čekání na zápis nebo čtení do paměti tím, že každému procesoru je poskytnuta samostatná paměť. Používá těsnější vazbu více CPU v uzlu, které jsou dále propojené do větších celků.

## 6 Hierarchie paměti

Není možné mít všechna data uložená v té nejrychlejší paměti co nejblíže procesoru. Taková paměť by byla drahá a nevešla by se tam, proto máme několik stupňů paměti v různé velikosti a vzdálenosti od procesoru. Takový systém se rychlostí blíží tomu, který by využíval jen nejrychlejší paměti.

Hlavní myšlenkou je, že běžící programy používají v daný okamžik ke svému běhu jen část adresního prostoru. Navíc je pravděpodobné, že použijí lokální data:

**Časová lokalita** Nedávno použitá data se použijí znovu (proměnné, data v cyklu...)

**Prostorová lokalita** Použijí se položky blízké těm současným (další řádek v poli)

Paměťová hierarchie se skládá (směrem od procesoru) z: L1 cache, L2 cache, někdy L3 cache, operační paměti a pevného disku. Data se v paměti hledají nejprve v L1 cache, v případě neúspěchu se hledá v dalších cache a v operační paměti.

### 6.1 Virtualizace paměti

Virtuální paměť (VP) je úroveň abstrakce postavená nad všemi zdroji dostupné paměti. Umožňuje procesu/programu dotazovat se pouze na logické adresy (LA) a nezabývat se, jaká je jejich fyzická adresa (FA) a jestli jsou data uložena v RAM nebo na HDD. VP může být daleko větší, než je velikost fyzické paměti.

Převod z LA na FA zajišťuje procesor (hardwarově). Běžně je to implementováno pomocí stránkování, kde je stránkovaná jak operační paměť, tak místo na HDD (více viz otázka 1 z okruhu Informatika a počítačové vědy). Jednotka prostoru se nazývá v logickém prostoru “stránka” a ve fyzickém prostoru “rámec”, obě mají stejnou velikost.

Adresy se skládají ze tří částí. V první části je adresa stránky, která se pomocí tabulky stránek přeloží na adresu rámce. V druhé části je adresa uvnitř stránky/rámce, neboli “offset”, která se překladem nezmění. Ve třetí části jsou uloženy příznaky: validity bit, access rights, dirty bit.

Pokud rámec není přítomný v cache, tak se tam načte z disku. Pokud tam není místo, musí se nejprve rozhodnout, jaký jiný rámec se smaže. Mazaný rámec se vybere pomocí některého z algoritmů, nejčastěji se používá Last Recently Used (LRU). V případě, že byla data mazaného rámce změněna (mají aktivní dirty bit), tak se nejprve zapíše na disk. Nakonec se aktualizuje page table.

## 7 I/O podsystém

Druhy periférií:

1. výstupní
2. vstupní
3. obousměrné - např. HDD

### 7.1 Metody přenosu z/na periferie

**Programový kanál - pooling** Program čeká ve smyčce, dokud nejsou data dostupná. Je to nejjednodušší a zároveň nejloupežší řešení.

**Programový kanál s přerušením** IO operace je zahájena na žádost programu pomocí OS. Existují dvě varianty:

- synchronní - program čeká na dokončení IO operace
- asynchronní - program nečeká na dokončení IO a může běžet souběžně s ní. Jakmile jsou data dostupná, periferie vyvolá přerušení a dojde k jejich přečtení.

**Direct memory access (DMA)** Přenos je realizovaný speciální jednotkou bez přímé účasti OS. Ten jen nastaví parametry přenosu (kolik bajtů, do jakého bufferu a na jaké adrese). Následně řadič periferie inicializuje DMA přenos, a ten probíhá, dokud nejsou data přečtena. Po ukončení přenosu je vyvolané přerušení. Výhodou je, že velké datové přenosy nevytěsňují data z cache. DMA řadič umí pracovat s různými periferiemi, třeba i se síťovým rozhraním.

**Autonomní kanál (Bus Master DMA)** Podle slidů je téměř totožný s DMA, ale na wiki toho je víc. Periferie má vlastní řadič, a může dostat kontrolu nad paměťovou sběrnici. Dokáže tedy zapsat přímo do paměti bez jakéhokoli zapojení ze strany CPU. Průběh přenosu:

1. Nadržený procesor vloží sekvenci datových bloků do paměti
2. Nakonfiguruje nebo přímo naprogramuje řadič periferie, ta provede sekvenční přenos
3. Po úplném nebo částečném přerušení je informován procesor

## 8 Výjimky a přerušení

- Výjimky
  - pro MIPS např. matematické přetečení, načtení neznámé instrukce, systémové volání
  - nedostupnost dat nebo selhání zápisu
- Přerušení
  - maskovatelné - lze zakázat ve stavovém slovu CPU
  - nemaskovatelné - ošetření HW chyb, hlídacích obvodů

Výjimky jsou přijaty téměř vždy, přerušení jen pokud jsou nemaskovatelná nebo povolena.

Zpracování výjimky/přerušení:

1. Stavové slovo (PSW) a PC se uloží na zásobník nebo do speciálního registru
2. PC se nastaví na adresu obslužné rutiny připadající dané výjimce (případně i číslu zdroje přerušení). Rutina je pak vykonána.
3. V závislosti na typu výjimky/přerušení dojde ke specifickému zpracování
4. Provede se instrukce CPU pro uvedení do stavu před zpracováním výjimky/přerušení (instrukce návratu z přerušení) a obnoví se původní registry (PC a PSW)

Při správném obslužení výjimky by původní program neměl přímo poznat, že k přerušení došlo.

## 8.1 Určení zdroje výjimky/přerušení

**Soft. hledání** veškerá přerušení a výjimky spouštějí rutinu od stejné adresy. Rutina zjistí důvod přerušení ze stavového registru

**Vektorová obsluha přerušení** Číslo zdroje zjistí CPU. V paměti se nachází na pevném místě (specifikované řídicím registrem VBR) tabulka vektoru přerušení, CPU převede číslo zdroje na index a z něj načte v poli slovo, které vloží do PC

**Jiné**

- Nvektorová obsluha více pevně určených adres podle priorit
- Často i kombinované

Asynchronní vs. synchronní přerušení: asynchronní nejsou vázané na instrukci, zatímco synchronní ano.