Jádro procesoru – MO 11

Klasifikace procesorů

- Procesory lze klasifikovat podle různých kritérií, jako je architektura, výkon, účel použití nebo výrobní technologie
- Mohou být rozděleny podle počtu jader (jednojádrové, vícejádrové), podle výrobního procesu (například 14nm, 7nm), podle architektury (x86, ARM), nebo podle určení (procesory pro mobilní zařízení, serverové procesory atd)
- Klasifikace umožňuje lepší porovnání vlastností procesorů a volbu toho nejvhodnějšího pro konkrétní potřeby

Výkon procesoru

- Výkon procesoru je jeho schopnost vykonávat instrukce za určitý časový úsek
- Měří se různými parametry, včetně rychlosti (GHz), počtu jader, vyrovnávací paměti (cache), architektury (např superskalární architektura umožňující zpracování více instrukcí současně), a efektivity zpracování instrukcí (IPC - Instructions Per Cycle)
- Výkon procesoru je klíčový pro rychlost vykonávání programů a úloh

Programátorský model procesoru

- Programátorský model procesoru definuje způsob, jakým programy komunikují s procesorem
- Zahrnuje instrukční sadu, která specifikuje, jaké instrukce může procesor provádět, a také architekturu registru a paměti, s níž programy interagují
- Programátorský model určuje způsob psaní softwaru a rozhraní mezi hardwarem a softwarem

Kompatibilita na úrovni strojového kódu *** doplnit arm aarm x64 x32

- Procesory mají specifický instrukční soubor, který chápou a provádějí
- Kompatibilita na úrovni strojového kódu se týká schopnosti procesoru provádět instrukce napsané pro konkrétní instrukční sadu
- Například, pokud má procesor instrukční sadu x86-64, bude schopen provádět instrukce napsané pro tento typ sady

Šířka slova procesoru

- Šířka slova označuje počet bitů, které procesor může zpracovat najednou
- Čím širší slovo, tím více informací může procesor zpracovat za jeden cyklus
- Šířka slova ovlivňuje výkon procesoru a jeho schopnost pracovat s daty a instrukcemi
- Například procesory s 32bitovou nebo 64bitovou šířkou slova mají různé schopnosti práce s daty a instrukcemi

Evoluce instrukční sady

- Instrukční sada (*ISA - Instruction Set Architecture*) je základní rozhraní mezi hardwarem a softwarem, určující, jaké instrukce může procesor vykonávat
- Evoluce ISA byla významná, od jednoduchých sad instrukcí (*RISC - Reduced Instruction Set Computing*) po složitější (*CISC - Complex Instruction Set Computing*) a specializované instrukční sady, jako například pro vektorové zpracování

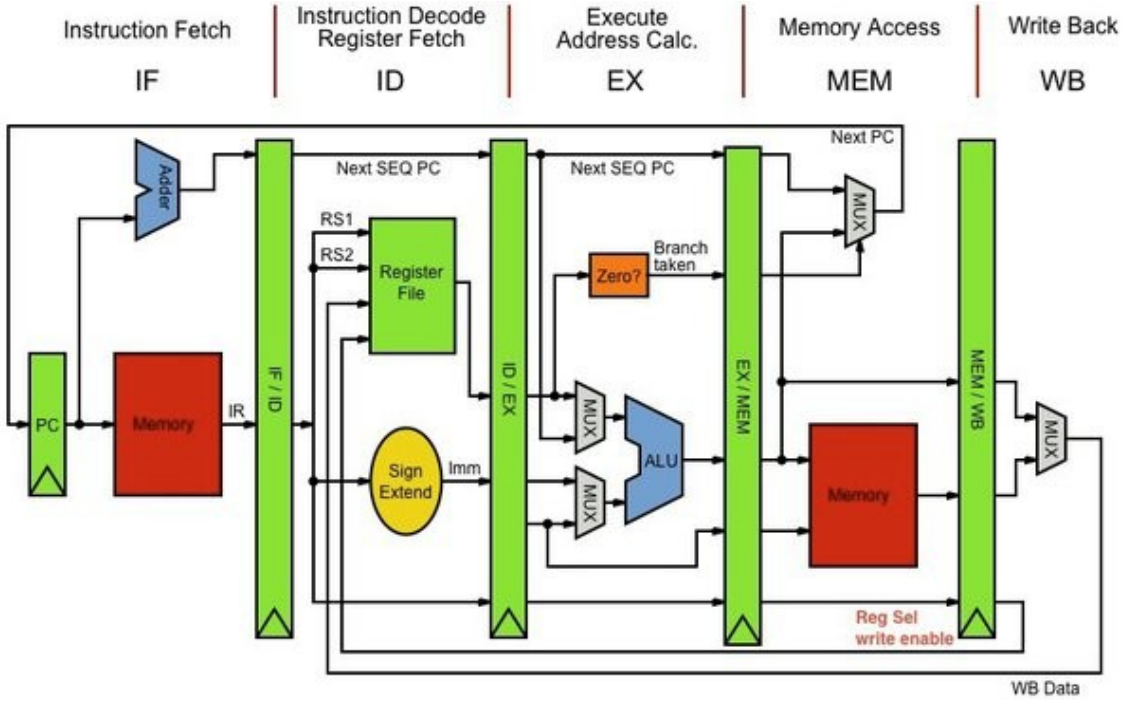
- **Vliv na hardware**
  - S rozvojem ISA se musel vyvíjet i hardware
  - Složitější instrukční sady vyžadují pokročilejší a často složitější procesorové architektury, což ovlivňuje design, výrobní procesy a efektivitu procesorů Například implementace ISA podporující paralelní zpracování nebo vektorové operace vyžaduje specifické hardwarové jednotky uvnitř procesoru
- **Vliv na software**
  - Vývoj ISA také ovlivňuje software
  - Nové instrukční sady umožňují efektivnější a rychlejší zpracování, ale vyžadují, aby software byl přizpůsoben nebo přepracován tak, aby z těchto vylepšení mohl těžit
  - To může zahrnovat aktualizace kompilátorů, operačních systémů a aplikací, aby plně využívaly nové instrukce

Organizace jednočipu

- Organizace jednočipu se týká uspořádání různých komponent procesoru na jednom integrovaném obvodu (čipu)
- To zahrnuje jádra procesoru, vyrovnávací paměť (cache), řadič paměti, řadiče vstupu/výstupu a další
- Toto uspořádání je klíčové pro výkon a efektivitu procesoru
- Optimalizace, jako je umístění jader blízko cache paměti nebo efektivní komunikace mezi jádry a pamětí, jsou důležité pro rychlou a efektivní práci procesoru

Důvody nasazení

- Důvody nasazení určitých technologií nebo architektur v počítačových systémech mohou být různé
- **Například**:
  - Výkon: Vyšší výkon pro náročné aplikace jako jsou databáze, hry, nebo software pro strojové učení
  - Energetická efektivita: Důležité pro mobilní zařízení a datacentra, kde je potřeba šetřit energií
  - Cena: Optimalizace nákladů pro určité tržní segmenty
  - Specifické potřeby aplikací: Například grafické karty pro vysoce výkonné grafické zpracování nebo specializované procesory pro AI a strojové učení
  - Bezpečnostní požadavky: Zahrnutí bezpečnostních funkcí přímo do hardwaru, jako jsou šifrovací moduly nebo bezpečnostní oddělení



1) **IF (Instruction Fetch)**
   - PC (Program Counter): Ukazuje na adresu instrukce, která se má načíst
   - Memory: Uchovává instrukce a data
   - Adder: Používá se pro výpočet adresy následující instrukce

2) **ID (Instruction Decode/Register Fetch)**
   - IF/ID (Instruction Fetch/Instruction Decode): Registr mezi IF a ID fází, uchovává načtenou instrukci
   - Register File: Uchovává hodnoty registrů a poskytuje je pro dekódování instrukcí
   - Sign Extend: Rozšiřuje krátké hodnoty na plnou délku pro další zpracování

3) **EX (Execute/Address Calculation)**
   - ID/EX (Instruction Decode/Execute): Registr mezi ID a EX fází, uchovává dekódovanou instrukci a data z registrů
   - ALU (Arithmetic Logic Unit): Provádí aritmetické a logické operace
   - MUX: Multiplexory v této fázi umožňují výběr mezi různými zdroji dat pro ALU

4) **MEM (Memory Access)**
   - EX/MEM (Execute/Memory Access): Registr mezi EX a MEM fází, uchovává výsledky z ALU a další informace pro přístup k paměti
   - Memory: Používá se pro čtení nebo zápis dat

5) **WB (Write Back)**
   - MEM/WB (Memory Access/Write Back): Registr mezi MEM a WB fází, uchovává data, která mají být zapsána zpět do registrů
   - MUX (Multiplexory): V této fázi se vybírá, která data budou zapsána zpět do register file

Detailní popis obrázku:

- **PC**:
  - Jedná se o registr, který uchovává adresu následující instrukce, která má být načtena z paměti
  - Přidává k aktuální adrese hodnotu 1 nebo skokovou adresu podle potřeby
- **Memory**:
  - Paměťová jednotka, kde jsou uloženy instrukce a data
  - Umožňuje čtení (pro načítání instrukcí nebo dat) a zápis (pro ukládání dat zpět do paměti)
- **Adder**:
  - Provádí sčítání, typicky se využívá k inkrementaci PC pro načítání následující instrukce
- **IF/ID**:
  - IF/ID je registr mezi fázemi IF (Instruction Fetch) a ID (Instruction Decode)
  - Uchovává načtenou instrukci pro dekódování v ID fázi
- **ID/EX**:
  - ID/EX je registr mezi fázemi ID a EX
  - Uchovává dekódovanou instrukci a data z registrů pro provedení výpočtu v EX fázi
- **EX/MEM**:
  - EX/MEM je registr mezi fázemi EX a MEM
  - Uchovává výsledky výpočtu z ALU a další informace potřebné pro přístup k paměti v MEM fázi
- **MEM/WB**:
  - MEM/WB je registr mezi fázemi MEM a WB
  - Uchovává data, která mají být zapsána zpět do registrů po dokončení operací v MEM fázi
- **MUX**:
  - Multiplexory slouží k výběru mezi různými datovými zdroji
  - V tomto kontextu MUX může mít různé funkce, jako je výběr vstupů pro ALU nebo řízení datového toku
- **Sign Extend**:
  - Rozšiřuje krátké hodnoty, jako například Immediate hodnoty z instrukcí, na plnou délku pro další zpracování
- **Zero?**:
  - Tento blok se zdá být součástí kontrolní logiky
  - Pravděpodobně zjišťuje, zda je výsledek operace nulový a může ovlivnit následující operace
- **ALU**:
  - Provádí aritmetické a logické operace (například sčítání, odečítání, logické AND/OR apod) na základě instrukcí

Zápis do registru

- 1) Identifikace registru
  - Procesor identifikuje konkrétní registr, do kterého má být zapsána hodnota
  - Tato informace je obvykle součástí strojového kódu instrukce
- 2) Příprava dat k zápisu
  - Hodnota, která má být uložena v registru, je buď připravena v předchozích fázích (například výsledek operace z ALU) nebo je to immediate hodnota přímo ze strojového kódu
- 3) Adresování registru
  - Vnitřní logika procesoru určí fyzickou adresu konkrétního registru, kam má být data zapsána
- 4) Samotný zápis dat
  - Data jsou zapsána do identifikovaného registru
  - Ve vícestupňovém procesu může být zajištěno, že zápis je proveden synchronně, aby se zabránilo kolizím a zaručila správnost operace

Čtení z registru

- 1) Identifikace registru
  - Procesor identifikuje konkrétní registr, ze kterého má být hodnota přečtena
  - Tato informace je opět obvykle součástí strojového kódu instrukce
- 2) Adresování registru
  - Vnitřní logika procesoru určí fyzickou adresu konkrétního registru, odkud má být data načtena
- 3) Čtení dat
  - Hodnota v registru je načtena a předána dalším částem procesoru pro další zpracování
  - Stejně jako u zápisu může být čtení synchronizováno, aby se zabránilo konfliktům a zajišťovala se konzistence dat

Kombinační logika
- je přítomna v ALU a různých multiplexorech, které provádějí výpočty a výběr signálů bez ukládání stavu
- Paměťové prvky jsou reprezentovány registry (např IF/ID, ID/EX, EX/MEM, MEM/WB) a register file, které uchovávají data mezi jednotlivými fázemi

Synchronní stroj
- Celý procesor funguje jako synchronní stroj, kde přechod mezi jednotlivými fázemi je řízen hodinovým signálem

Jádro procesoru
- Každá část jádra procesoru se specializuje na určitou fázi zpracování instrukcí, od načtení přes dekódování, výpočet až po přístup k paměti
- To umožňuje efektivní a paralelní zpracování instrukcí v rámci procesoru