SAP – BI-SPOL-28

Architektura číslicového počítače, instrukční cyklus počítače, základní třídy souborů instrukcí (ISA). Paměťový subsystém počítače, paměťová hierarchie, cache.

Obsah

1 Architektura číslicového počítače **2**
1.1 Harward vs. Von Neuman 2

2 Instrukční cyklus počítače **2**

3 Základní třídy souborů instrukcí (ISA - Instruction Set Architecture) **3**
3.1 Střadačová (acumulator) 4
3.2 Zásobníková (stack) 4
3.3 Registrová (GPR - General Purpose Registers) 5

4 Paměťový subsystém počítače **5**
4.1 Paměťová hierarchie 5
4.2 Cache 5

1 Architektura číslicového počítače

- základní části počítače:
  1. Datová část (ALU) - součást procesoru
  2. Řídící část (řadič) - součást procesoru
  3. Hlavní paměť - často mimo procesor
  4. vstupní zařízení
  5. výstupní zařízení

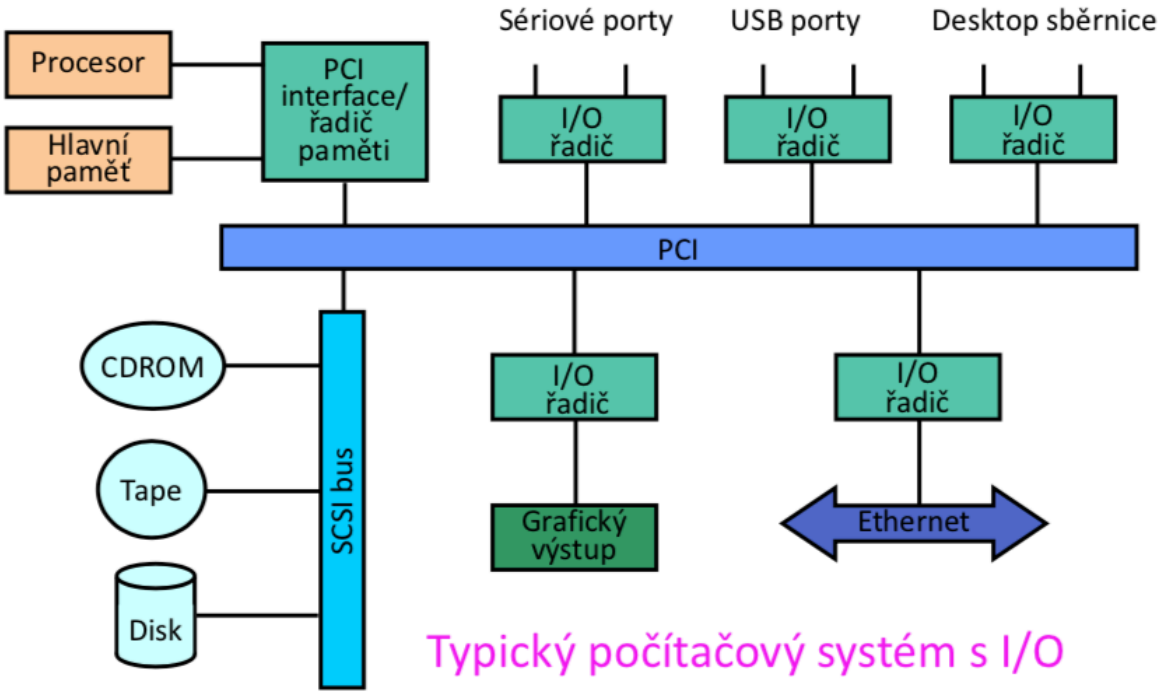

Figure 1: Harwarová architektura počítače

1.1 Harward vs. Von Neuman

- Harward má oddělenou paměť instrukcí od paměti dat, Von Neuman nikoli
- Harward používaný častěji u malých jednočipových počítačů (ATmega169,...)

2 Instrukční cyklus počítače

- instrukční cyklus
  1. IF - čtení instrukce (instruction fetch)
  2. ID - dekódování instrukce (instruction decode)
  3. OF - čtení operandů (operand fetch)
  4. IE - provedení instrukce (instruction execute)
  5. WE - uložení výsledku (write back)
  6. přerušení? (interrupt)
- nejnižší "úroveň" na které může programátor pracovat - staví se pomocí ní SW
- **instrukce** - příkaz zakódovaný jako číslo, musí obsahovat:
  - co se má provést (instrukce)

  - s čím se to má provést (operandy)
  - kam uložit výsledek
  - kde pokračovat (např. instrukce *ret* pokračuje jinde než *add*)

3 Základní třídy souborů instrukcí (ISA - Instruction Set Architecture)

- zahrnuje:
  - typy a formáty instrukcí
  - datové typy, kódování, reprezentace a způsob uložení dat v paměti
  - módy adresování a přístup do paměti dat/instrukcí
  - mimořádné stavy
- umožňuje:
  - abstrakci (různé implementace stejné architektury)
  - definici rozhraní mezi SW a HW
  - standardizuje instrukce
- adresace operandů:
  - přímá - pracuje se přímo s registrem nebo adresou v operandu
  - nepřímá - v registru/paměti je adresa na data se kterými se pracuje
  - relativní - offset od určité adresy (v registru nebo immediate)
  - indexovaná - báze + offset
    * autoinkrementace/autodekrementace

3.1 Střadačová (acumulator)

- implicitním operandem ALU je vždy střadač
- byl populární v 50. a 70. letech, protože HW byl drahý a paměti rychlejší než CPU
- výhody
  - jednoduchý HW
  - minimální stav procesoru => rychlé přepínání kontextu
  - krátké instrukce (záleží na type daného operátoru)
  - jednoduché dekódování instrukcí
- nevýhody
  - častá komunikace s pamětí
  - omezený paralelismus mezi instrukcemi

3.2 Zásobníková (stack)

- využívání "HW zásobníku" při vykonávání programu
- např. instrukce ADD vezme 2 nejvyšší hodnoty na zásobníku a do vrchní uloží jejich součet
- tento typ byl využit např. u x87 FPU (floating point unit)
- výhody:
  - jednoduchá a efektivní adresace operandů
  - krátké instrukce
  - vysoká hustota kódu (tzn. krátké programy)
  - jednoduché dekódování instrukcí
  - snadné napsání překladače (tedy bez optimalizací)
- nevýhody:
  - nelze náhodně přistupovat k lokálním datům
  - zásobník je sekvenční (omezuje paralelizmus)
  - těžké omezit přístupy do paměti

3.3 Registrová (GPR - General Purpose Registers)

- dnes nejrozšířenější
- RISC a CISC
- výhody:
  - registry jsou rychlejší než paměť (dokonce i než cache)
  - lze k nim přistupovat náhodně
  - mohou obsahovat mezivýsledky a lokální proměnné
  - méně častý přístup do paměti => potenciální možnost zrychlení
- nevýhody:
  - registrů je omezený počet
  - složitější překladač (např. které hodnoty nechat v registrech…)
  - delší přepínání kontextu
  - registry nemohou obsahovat složitější datové struktury
  - k objektům v registrech nejde přistupovat přes ukazatele

4 Paměťový subsystém počítače

`TODO???`

4.1 Paměťová hierarchie

1. registry
2. caches - extrémně rychlé, drahé, kapacitou menší, umístěné co nejblíž k procesoru
   - primární cache
   - sekundární cache
3. hlavní paměť - rychlé, levnější, větší (např. paměť RAM)
4. vnější paměť - pomalé, obrovská kapacita, odkládání (např. pevný disk)

4.2 Cache

- řeší nízkou rychlost hlavní paměti
- většinou ve více vrstvách (L1, L2, …)
- nižší vrstvy jsou menší rychlejší a dražší
- často jsou využívané asociativní paměti
- čtení:
  - **cache hit** - data jsou v cache nalezena
    - **hit rate** - poměr *cache hit* a počet všech dotazů
    - **hit time** - doba nalezení údajů v cache a předání procesoru
  - **cache miss** - výpadek cache (je třeba načíst z nižší úrovně)
    - **miss rate** - četnost výpadků cache = 1-hit_rate
    - **miss penalty** - doba potřebná k získání z nižší paměti
- zápis:
  - pokud není v cache jde rovnou do paměti
  - **write throught** - nová hodnota se zapíše do cache i do hlavní paměti
  - **write back** - zapíše se do paměti pouze když by měla být z cache vyřazena
- více stupňů asociativity - více míst kam uložit paměť se stejným klíčem (použití LRU při kolizích pro vyhození)

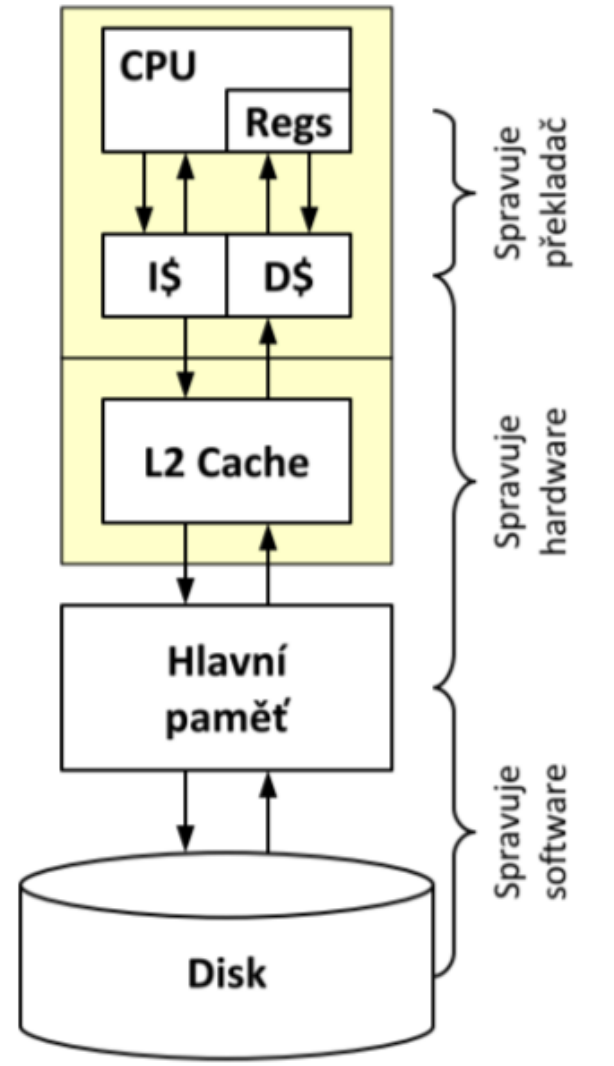

Figure 2: Paměťová hierarchie