M10 Základní cyklus počítače

#technicke_vybaveni_pocitacu

- obvykle se nazývá "fetch-decode-execute cycle"
- popisuje zk. kroky opakované při každé instrukci
- architektura RISC má omezený počet instrukcí na základní a jednoduché

1. čtení *(fetch)*
   - instrukce se načtou z paměti
   - adresa instrukce k provedení je uložena v registru Program Counter *(PC)*
   - instrukce se načte adresy z PC do Instruction Register *(IR)*
2. dekódování *(decode)*
   - načtená informace *(v IR)* obsahuje operační kód *(opcode)* a další informace
   - během dekódování jsou jednotlivé části instrukce identifikovány pro další zpracování
   - pomocí opcode je určeno, jakou operaci instrukce představuje
   - pokud instrukce potřebuje operandy, dekódování je identifikuje a připraví k použití
3. provedení *(execute)*
   - vykonává operaci definovanou dekódovanou instrukcí (např.: aritmetické operace, logické operace, přesuny dat, skoky nebo další)
   - po provedení instrukce se aktualizují stavové registry obsahující informace o procesoru (např. přetečení)
   - výsledky operací jsou zapsány do registrů nebo do paměti
4. zásah do paměti
   - umožňuje procesoru přistoupit k paměti
   - adresa je buď předem určena nebo je vypočtena na základě aktuálních hodnot v registrech; adresa je následně odeslána na paměťovou sběrnici
   - hodnota z interních *(pipeline)* registrů je zapsána na adresu
   - po zápisu se aktualizují stavové bity *(bity ovlivňující následující průběh programu)*
   - dojde k aktualizaci PC aby ukazovala na další instrukci v paměti

- aktualizace PC
  - PC může být automaticky inkrementován po provedení každé instrukce (např. instrukce programu jsou uloženy za sebou v paměti, po provedení instrukce se PC zvedne o jeden → ukazuje na další instrukci v paměti)
  - při skoku (`jump`) nebo větvení (`branch`) se hodnota PC mění na základě logické podmínce; pokud je podmínka splněna, PC ukazuje na adresu, kde má program pokračovat
  - při volání (`call`) se adresa následující instrukce (adresa návratu) uloží na zásobník a adresa PC je změněna ukazujíc na první instrukci podprogramu; po provedení podprogramu je (návratová) adresa další instrukce načtena ze zásobníku do PC
  - při přerušení je PC aktualizován na adresu uloženou při volání podprogramu
  - v případě výjimky je adresa PC změněna na adresu obslužné rutiny výjimky; po obsluze výjimky může být adresa PC změněna na pokračování běhu programu

Výjimečné stavy při běhu CPU

- stavy které mohou vyžadovat speciální pozornost či manipulaci
- přerušení *(interrupt)*
  - přerušují běžný tok programu
  - vyžadují okamžitou pozornost procesoru
  - vyvolána externím zařízením, chybou programu nebo samotným programem úmyslně
  - pro obsluhu přerušení musí procesor přepnout kontext a reagovat na příslušné události
- výjimky *(exception)*
  - podobné přerušení
  - jedná se o chybový stav

    - vyžadují zvláštní opatření
    - procesor musí přepnout na obsluhu výjimky a přijmout opatření k řešení problému
    - např.: dělení nulou, přetečení při aritmetických operacích nebo přístup k neplatné paměti, broken pipeline
- přepínání kontextu *(context switch)*
    - nastává když běžící proces na CPU je pozastaven a CPU přepíná svůj kontext na jiný proces
    - informace (registry, program counter, adt.) pozastaveného procesu jsou uloženy do paměti
    - informace jsou následně aktualizovány aby odpovídaly novému procesu
- stav úspory energie *(halt)*
    - nastává když CPU přechází do režimu nízké spotřeby nebo je dočasně zastaven
    - neprovádí žádné instrukce a čeká na další pokyny
- bezpečnostní režim *(privileged mode)*
    - CPU má vyšší úroveň oprávnění než v normálním uživatelském režimu
    - má přístup k systémovým zdrojům (speciální registry, instrukce nebo přímý přístup k hardwaru)
- chyby přístupu do daměti *(memory access violations)*
    - program přistoupí k neplatné paměti nebo s ní provede nepovolenou operaci
    - např. pokus o čtení nebo zápis do neexistující adresy paměti

Formát instrukce

- AVR používá specifický formát založený na RISC formátu
- instrukce v AVR arch. jsou 16bitové
- instrukce
    - typu R *(-egistr)*
        - pracuje s registry
        - provádí aritmetické nebo logické operace
        - `opcode | Rd | Rr`
            - `opcode` → identifikační číslo operace
            - `Rd` → cílový registr (a zároveň zdrojový)
            - `Rr` → zdrojový registr
    - typu I *(-mmediate)*
        - pracuje s konstantami
        - konstanty jsou uloženy přímo v instrukci
        - `opcode | Rd | K`
            - `K` → konstanta
    - typu J *(-ump)*
        - pro řízení toku programu
        - `opcode | d`
            - `d` → adresa destinace skoku

Memory Access: Load

Op	Rs1	ws	Offset
4	3	3	6

Memory Access: Store

Op	Rs1	Rs2	Offset
4	3	3	6

Data Processing:

Op	Rs1	Rs2	ws	Offset
4	3	3	3	3

Branch:

Op	Rs1	Rs2	Offset
4	3	3	6

Jump:

Op	Offset
4	12

OP	
0000	Load Word
0001	Store Word
0002	Add
0003	Subtract
0004	Invert
0005	Logical Shift Left
0006	Logical Shift Right
0007	Bitwise AND
0008	Bitwise OR
0009	Set on less than
0010	Hamming Distance
0011	Branch on Equal
0012	Branch on NOT Equal
0013	Jump

Operační

jednotka

- aritmetické a logické operace provádí odděleně od těch ostatních
- instrukce jsou navrženy aby byly jednoduché; vyžadují minimální počet cyklů pro provedení
- každá instrukce provádí pouze jednu konkrétní operaci
- maximalizuje paralelní zpracování; navrženy pro nezávislé vykonávání vedoucí k rychlejšímu výpočtu
- pracuje pouze s daty v registrech; pokud data nejsou v registru, zapíše je
- pevně definovaná šířka slova

znak

- základní strojový kód říkající procesoru, jakou operaci má provést
- "*operátor*"
  - v kontextu programování se jedná o symbol či zkratku reprezentující operaci (např.: `ADD` - sčítání; `SUB` - odčítání; `AND` - logický AND; `OR` - logický OR; `MOV` - přesun dat; `JMP` - nepodmíněný skok)
  - operátory jsou následně převedeny do binární podoby
  - operátory závisí na konkrétním RISC procesoru

Pipeline registr

- speciální druh registrů používaný pro pipelining
- pipelining - technika umožňující provádění několik fází instrukce současně
- každá fáze má svůj registr; slouží k uložení mezikroků
- broken pipeline - situace, kdy je normální tok instrukcí v pipeline přerušen nebo narušen; obvykle vyžaduje nějakou formu opravy nebo obnovení pipeline do normálního stavu

Řadič

- dekóduje instrukce z paměti
- připravuje interní obvody pro provedení instrukcí
- řídí takt procesoru; určuje prioritu provedení instrukce
- spravuje přístup čtení a zápisu do paměti
- řídí tok dat v datových cestách
- detekuje stav výjimek a přerušení; popřípadě dokáže vyřešit neobvyklé stavy
- provádí skoky a podmíněné instrukce
- zodpovídá za plynulý a synchronní pohyb instrukcí skrz pipeline

Dekodér

- interpretuje instrukce z programové paměti a připravuje připravuje je k provedení
- čte instrukce a převádí je do procesorem čitelné formy
- identifikuje a připravuje operandy (např. načtení hodnot z registrů/paměti)
- počítá adresy pro skoky
- identifikuje výjimky
- dekóduje paralelně s pipeline a zajišťuje aby každá fáze proběhla správně