Přerušení

25. listopadu 2008
10:50

Přerušení = tok instrukcí je přerušen nějakou akcí, a pak se vrátíme k původnímu proudu (zlepšení Von Neumannovské koncepce, ta s tím nepočítala)

Vnější příčiny - provedu až dokončím předchozí
- Význačné stavy I/O zařízení
  - To nám umožňuje asynchronní procesy, přerušení je jedno z metod, jak je implementovat, když potřebujeme něco, tak se zařízení samo ozve interruptem, bude obslouženo (tím zlikvidujeme periodické zkoumáni co se děje) => i rychlejší odezva po nastání akce (jinak bysme se pořád ptali klávesnice jestli něco nezmáčkl uživatel)
  - U výstupních.. Chceme mu poslat nějak data, tak bysme se museli pořád ptát, jestli můžeme poslat,
- Význačné stavy čítačů
  - No n-té instrukci chceme např. poslat přerušení.. Např. počítač změny stavu na vstupním pinu
  - Watchdog je speciální případ (zabránění nekonečným cyklům, po x provede reset)
- Zásahy uživatele
- Poruchy HW

Vnitřní příčiny - je třeba je provádět okamžitě při provádění (např. nemám data)
- Neznámá instrukce
  - Procesor neví co má dělat, proto to nechá řešit na někom jiném (sice zase procesorem, ale už podle scénáře co se má dělat při neznáme instrukci)
  - Např. provedení neprivilegované instrukce
- Speciální instrukce (INT, SYSCALL)
  - INT - prostě vygeneruje interrupt
  - SYSCALL - pro přístup k rutinám OS - chceme-li něco vykonat a nevíme kde to je (vím že chci přerušení 21, a nevím kde OS schovává pro obsluhu procesu, tak pomocí SYSCALL)
  - Dalo se použít pro ladění programů
    - Když si ve vývojovým prostředí dáme breakpoint, tak debugger na danou instrukci dal interrupt, na něj si počkal, koukl se na kterou instrukci přepsal, a mezitím vykreslil break, když měl pokračovat dál, našel co přepsal.. 1 bytová interrupt instrukce
- Výpadek (stránky/segmentu)
  - Mám instrukci, ale nemám data se kterýma ji mám provést
  - OS si to odchytí, zjistí z jaké adresy to mělo jít, zjistí kde ta data jsou (např. ze SWAPu) a dá zpět OS

Průběh přerušení - po obsloužení se musíme vrátit zpět, k tomu, co jsme dělali předtím
- Vznik, vyslání žádosti - to ještě neznamená, že se tím přerušením bude někdo zabývat
- Rozhodnutí o přijetí / nepřijetí (+ maskování přijetí)
- Identifikace zdroje - kdo něco chce
- Určení adresy obslužného programu - čím obsloužit
- Úschova aktuálního stavu CPU (abych se pak mohl vrátit zpět)
- Provedení obslužného programu
- Obnova stavu CPU (do původního)
- Návrat do přerušeného programu

- Přijetí
  - Vnitřní - exception = nutno řešit hned
  - Vnější - interrupt = stačí řešit mezi instrukcemi (pro zachování definovaného stavu CPU)
- Výběr mezi žadateli o přerušení
  - Zřetězená zařízení
    - Mám zařízení spojena do logického řetízku, a ten posílá žádost o přerušení do procesoru, a procesor bud vyhoví, a z řetízku se pak někdo ozve (většinou první, kdo si o přerušení řekl) jde o prioritu
      - Výhodou, že procesor zajišťuje jen jedno přerušení a zbytek si zajistí ostatní

        - Nevýhodou, ze musíme dodat logiku, kdo si ohlídá kdo je na přerušení na řadě
    - Řadič přerušení,
        - Pro procesor zajišťuje pomocné akce pro přerušení, pokud procesor povolí
        - Dá se dobře konfigurovat a měnit
        - V drtivé většině počítačů
        - Klidně to může být součást počítače, ale je to oddělená jednotka
- Identifikace zdroje
    - Čistě programová
        - Jediný obslužný program, zjistí si sám kdo si o přerušení žádal (pružné, ale pomalé, např. Motorola 6800)
        - Podle potřeby dynamicky můžu měnit kód starající se o interrupy
    - S pomocí technických prostředků
        - Žadatel poskytne vektor přerušení, podle něj se rozhodne o obslužném programu (u x86 je pod adresou 0 uložen vektor přerušení, což je zde tabulka adres, kam se má skočit, dojde li k přerušení),
          Tam může být:
            - Adresa obslužného programu
            - Index do tabulky adres
            - Strojová instrukce - typické pro mikrokontrolery
        - Významné kvůli dynamičnosti - můžu kompletně řídit co se má dělat při přerušení (můžu nahradit vlastní rutinou, která bude na míru pro nás; prostor pro viry, kdysi)