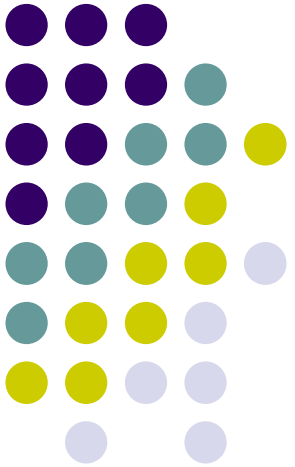
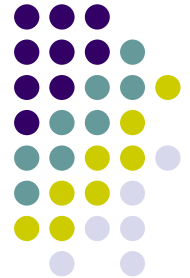
PPI

09.11.2011

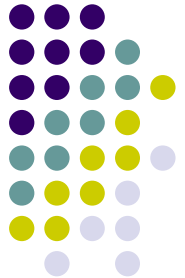


Procesor s architektúrou CISC (Complex Instruction Set Computer)



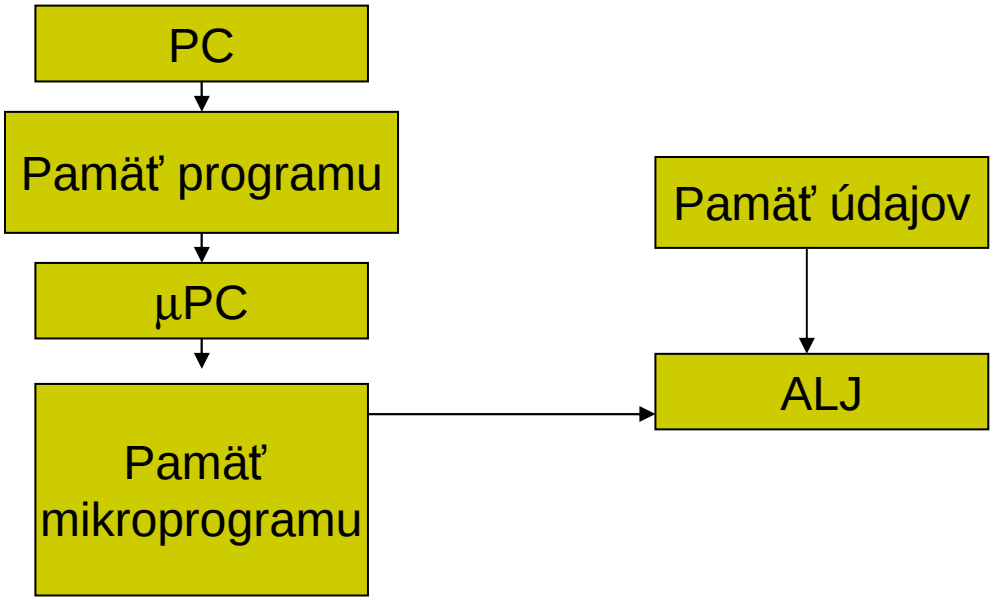
- *Zložitý inštrukčný súbor*-podporuje preklad z vyšších programovacích jazykov do strojového kódu procesora.
- *Veľa spôsobov adresácie operandov.*
- Inštrukcie realizujú aj *zložitejšie,* napr. reťazcové operácie,
- Implementovaná *priama podpora niektorých funkcií OS*
- Dodržiava sa k*ompatibilita inštrukčného súboru v rodinách procesorov* zdola nahor atď.
- Zložitá *mikroprogramová riadiaca jednotka.*
- Zložité inštrukcie si vyžadujú pre svoju realizáciu veľký súbor *mikroinštrukcií* a podporných technických prostriedkov

Procesor s architektúrou CISC (Complex Instruction Set Computer)

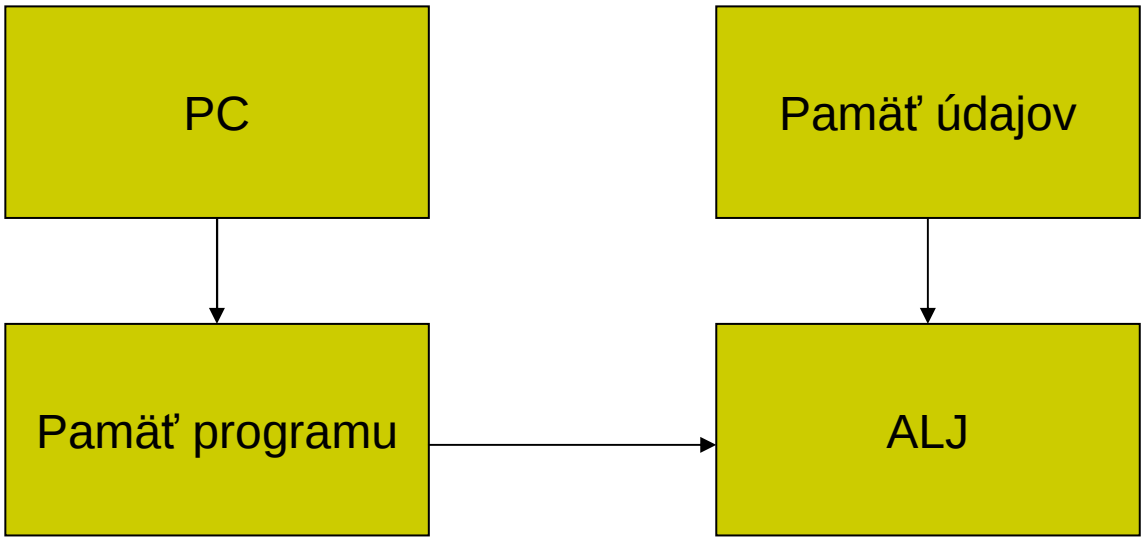


- Technické prostriedky na podporu mikroprogramovania zaberajú na čipe veľkú plochu, čím je daná aj vyššia cena
- Realizácia inštrukcií vzhľadom na použitie mikroprogramovej riadiacej jednotky trvá dlhšiu dobu.
- Iba malá podmnožina inštrukčného súboru tvorí väčšiu časť programov, pričom zložitejšie inštrukcie sa využívajú málo. Zložité inštrukcie dokonca môžu nepriaznivo ovplyvňovať optimalizáciu prekladu zložitejších jazykových konštrukcií vyšších programovacích jazykov.
- Predstaviteľmi procesorov *CISC* sú napr. procesory počítačov radu *IBM 43xx, VAX 11/780* alebo mikroprocesory rodiny *Intel 80x86*

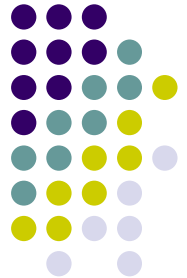
Procesor s architektúrou RISC (Reduced Instruction Set Computer)



Procesor s architektúrou RISC vs. CISC

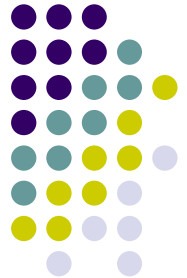


Procesor s architektúrou RISC (Reduced Instruction Set Computer)



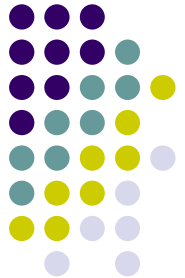
- Redukovaný inštrukčný súbor
- Inštrukcie sú jednoduché, ich vykonanie krátke (typicky v 1 strojovom cykle)
- Používa sa málo spôsobov adresácie operandov (obyčajne iba inštrukcie typu čítanie/zápis z/do hlavnej pamäte),
- Používa sa väčší počet univerzálnych registrov (desiatky až stovky).
- Riadiaca jednotka je jednoduchšia, obyčajne je to riadiaca jednotka s pevnou logikou.
- Na čipe sa uvoľnila značná časť plochy na implementáciu numerického koprocesora a vyrovnávacej pamäte Cache.
- Vykonanie inštrukcie, uloženej vo vyrovnávacej pamäti, je rýchlosťou porovnateľné s vykonaním mikroinštrukcie.
- Predstaviteľmi procesorov RISC sú napr. procesory SPARC, Motorola 88000, Transputer fy INMOS atď. (Apple)

Procesor s architektúrou RISC vs. CISC



- Program, v ktorom sa nachádzajú aj zložitejšie inštrukcie, je síce pri procesore *CISC* kratší, ako pri procesore *RISC* (procesor *RISC* musí zložitejšie inštrukcie nahradiť postupnosťou svojich jednoduchých inštrukcií), ale vzhľadom na réžiu procesora *CISC* (musí inštrukciu dekomponovať na postupnosť mikroinštrukcií) je jeho vykonanie procesorom *RISC* rýchlejšie.

Procesor s architektúrou RISC vs. CISC

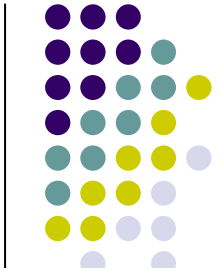


- RISC
- JA Jump if Above
- JAE Jump if Above or Equal
- JB Jump if Below ...
- JPO Jump if Parity Odd
- JS Jump if Sign
- JZ Jump if Zero (príznakovom registri)
- CISC
- **Branch and Branch with Link.**
- BLEQ Branch with Link if EQual

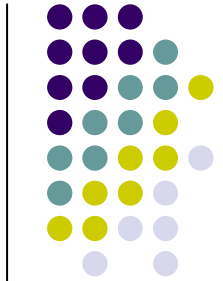
NISC,

```
;Dokaze previest binarne,
;hexadecimalne a decimalne cisla.
;
;Vstup: reg. BX obsahuje offset adresy
;cisla v pamati.
;
;Vystup: reg. DX obsahuje cisel. hodnotu
;-------------------------------
readnum   proc near
          mov al,byte ptr [cs:bx]
          inc bx
          mov dx,0
          mov cl,16
          xor ch,ch
          cmp al,"#"
          jz readnum3
          mov cl,2
          xor ch,ch
          cmp al,"%"
          jz readnum3
          xor ch,ch
          mov cl,10
          dec bx
readnum3: mov al,byte ptr [cs:bx]
          sub al,"0"
          cmp al,10
          jc readnum4
          sub al,"A"-"9"-1
readnum4: cmp al,16
          jc readnum6
```

RISC



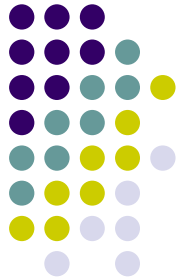
```
# Compute first twelve Fibonacci numbers and put in array, then print
        .data
fibs:   .word   0 : 12          # "array" of 12 words to contain fib values
size:   .word  12               # size of "array"
        .text
        la   $t0, fibs          # load address of array
        la   $t5, size          # load address of size variable
        lw   $t5, 0($t5)        # load array size
        li   $t2, 1             # 1 is first and second Fib. number
        add.d $f0, $f2, $f4
        sw   $t2, 0($t0)        # F[0] = 1
        sw   $t2, 4($t0)        # F[1] = F[0] = 1
        addi $t1, $t5, -2       # Counter for loop, will execute (size-2) times
loop:   lw   $t3, 0($t0)        # Get value from array F[n]
        lw   $t4, 4($t0)        # Get value from array F[n+1]
        add  $t2, $t3, $t4      # $t2 = F[n] + F[n+1]
```



NISC

Bez inštrukcií, bez dokódovania
Iba riadiace slová 2-3 krát dlhšie ale každé
vykonáva 2-3 inštrukcie RISC

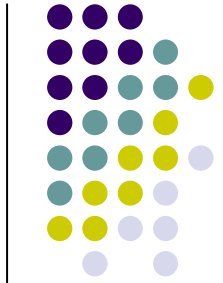
Prerušovací systém procesora



- Rýchla odozva na externé udalosti
- Prerušenie po udalosti, keď je nevyhnutné začať vykonávať nový program tzv. obslužný program, prerušenia

- Dnešný prerušovací systém je prioritný

Prerušovací systém procesora

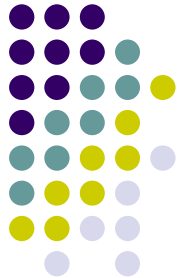


- Kasifikácia prerušení
  - A Externé (z okolia procesora)
  - B Interné (súvisia s vykonávaním inštrukcií)

  - A Asynchrónne (externé, nesúvisí s vykonávanými inštrukciami, možno ho zakázať)
  - B Synchrónne (súvisí s programom, nemožno ho zakázať, skok do operačného systému, chyba při vykonaní inštrukcie)
    - Softvérové prerušenie (číslo prerušenia)
    - Výnimka (exception, delenie nulou)

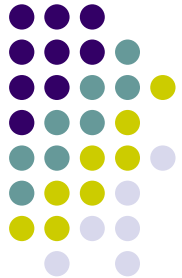
  - A Maskovateľné (niektoré externé, spravidla asynchrónne)
  - B Nemaskovateľné

Prerušovací systém procesora

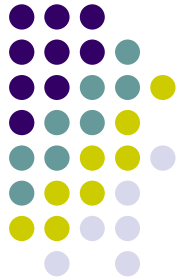


Prerušenie sa teda skladá z týchto krokov:

- prijatie požiadavky na prerušenie,
  (po vykonaní bežiacej inštrukcie)
- odloženie stavu procesora - PC, registre
  (zásobník)
- zistenie zdroja prerušenia, synchrónne prerušenia majú pevné štartovacie adresy
- vykonanie zodpovedajúceho obslužného programu prerušenia,
- obnovenie pôvodného stavu procesora,
- pokračovanie v prerušenom programe.



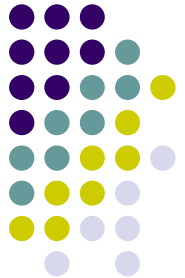
K bodu 1.
Požiadavka na *externé* prerušenie môže prísť v ľubovoľnom okamihu, t.j. aj uprostred vykonávania inštrukcie. **S obsluhou prerušenia** (t.j. odloženie stavu procesora atď.) **sa však začne až po dokončení práve vykonávanej inštrukcie.**
K bodu 2.
Okamžitý stav *procesora* je charakterizovaný obsahom všetkých registrov procesora. *Stav procesora sa odkladá do zásobníka.* Použitie zásobníka umožňuje aj *hniezdenie prerušení,* to znamená, že počas obsluhy jedného prerušenia môže prísť k akceptovaniu ďalšieho prerušenia *s vyššou prioritou.*



Asynchrónne prerušenie je prerušenie, ktoré priamo nesúvisí s vykonávanými inštrukciami a môže nastať kedykoľvek. Je to tzv. *externé (hardvérové) prerušenie* a typicky je požadované niektorým vstupno/výstupným zariadením, keď je toto *pripravené na prenos.*

Procesor má zvyčajne dva prerušovacie vstupy pre externé prerušenie:

- ***Vstup maskovateľného prerušenia.*** Inštrukčný súbor procesora obsahuje v tomto prípade špeciálne inštrukcie, ktoré umožňujú *povoliť* resp. *zakázať* prijatie požiadavky z tohto vstupu.
- ***Vstup nemaskovaieľného prerušenia.*** Toto prerušenie nie je možné zakázať a typicky sa používa pri obsluhe katastrofických situácií (napr. *výpadok napájacieho napätia*).



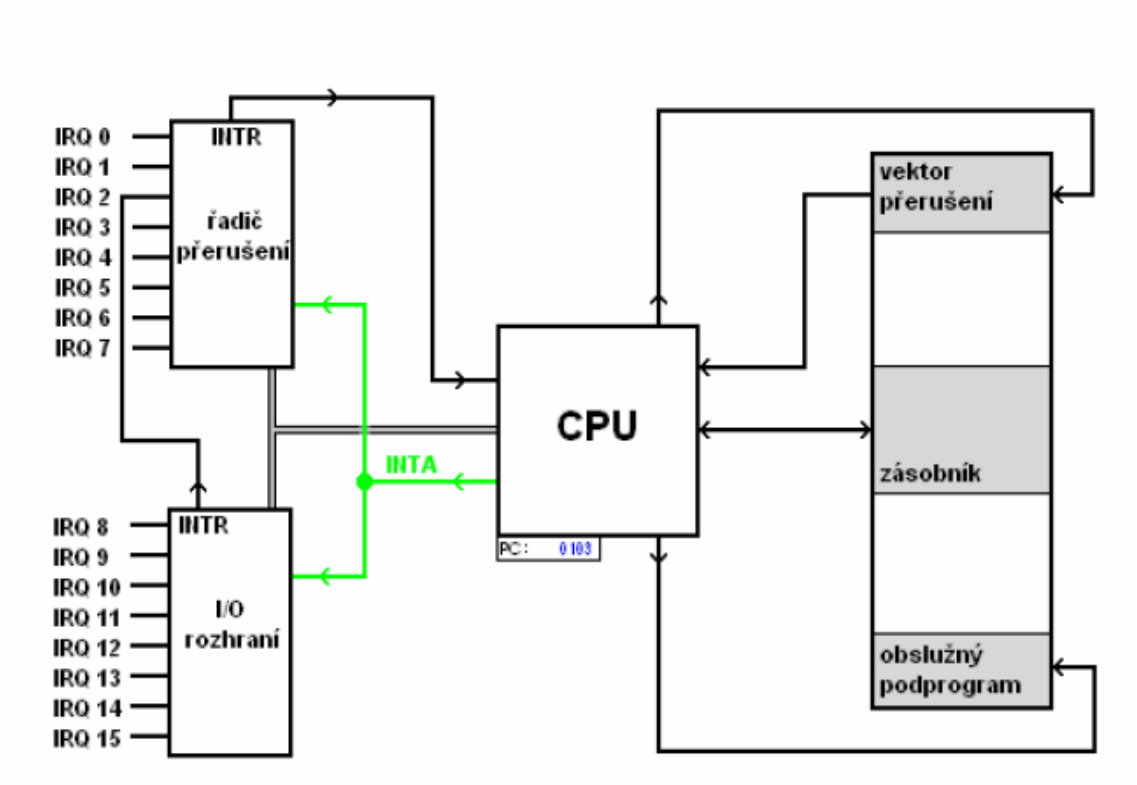
Synchrónne prerušenie

Synchrónne (interné) prerušenie priamo súvisí s vykonávanými inštrukciami a nie je ho možné zakázať. Synchrónne prerušenie je dvojaké:

•***Softvérové prerušenie.*** Softvérové prerušenie je generované po vykonaní špeciálnej *riadiacej inštrukcie.* Parametrom tejto inštrukcie je *číslo prerušenia,* ktoré sa má obslúžiť. Toto prerušenie sa typicky používa pri *volaní funkcií operačného systému.*

•***Výnimka (Exception).*** Výnimka sa generuje automaticky, ak nastane nejaká chyba pri vykonaní inštrukcie (napr. *nedefinovaný operačný kód, delenie nulou, pokus o zápis do oblasti, kde sú uložené inštrukcie, sprístupňovaný segment sa nenachádza v hlavnej pamäti* atď.).

1. **Průběh hardwarového přerušení**
2. Vnější zařízení vyvolá požadavek o přerušení
3. I/O rozhraní vyšle signál IRQ na řadič přerušení (na port IRQ 2)
4. Řadič přerušení vygeneruje signál INTR – „někdo“ žádá o přerušení a vyšle ho k procesoru.
5. Procesor se na základě maskování rozhodne obsloužit přerušení a signálem INTA se zeptá, jaké zařízení žádá o přerušení.
6. Řadič přerušení identifikuje zařízení, které žádá o přerušení a odešle číslo typu přerušení k procesoru
7. Procesor uloží stavové informace o právě zpracovávaném programu do zásobníku.
8. Podle čísla typu příchozího přerušení nalezne ve vektoru přerušení adresu příslušného obslužného podprogramu.
9. Vyhledá obslužný podprogram obsluhy přerušení v paměti a vykoná ho.
10. Po provedení obslužného programu opět obnoví uložené stavové informace ze zásobníku a přerušený program pokračuje dál.



http://www.dnp.fmph.uniba.sk/~kollar/pc_hw_sw/pc7.htm
http://cs.wikipedia.org/wiki/P%C5%99eru%C5%A1en%C3%AD