Arhitectura sistemelor de calcul

- Prelegerea 13 -

4-DS (Calculatoare)

Ruxandra F. Olimid

Facultatea de Matematică şi Informatică
Universitatea din Bucureşti

Cuprins

4-DS (Calculatoare)

- Introducem încă un *ciclu* la sistemele cu 3 cicluri (3-DS) şi obţinem sistemele *4-DS*

- Prin închiderea unui ciclu peste un **procesor** şi o memorie **RAM** se obţine un **calculator**

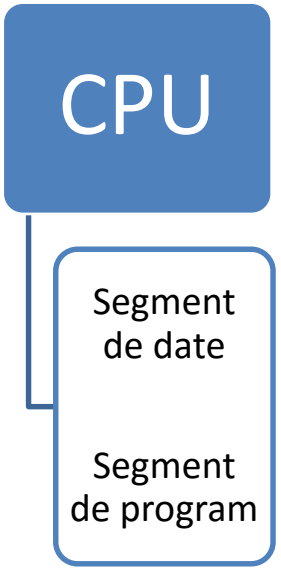
- Prin închiderea unui ciclu peste un **procesor** şi o memorie **ROM** se obţine un **microcontroller** (one-chip computer)

- Există şi alte tipuri de închidere a ciclului 4; spre exemplu, se poate închide un ciclu peste 2 procesoare (de fapt un procesor şi un co-procesor specializat pentru anumite funcţii)

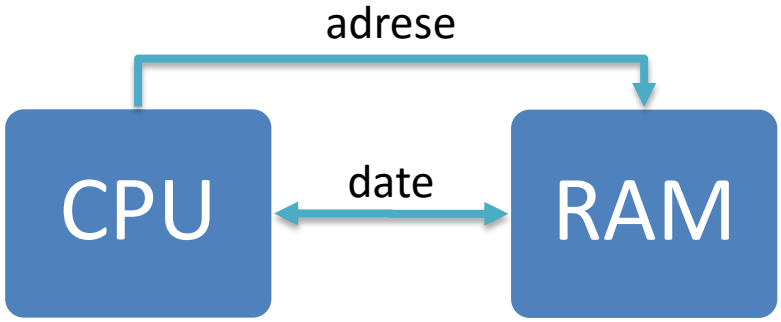
[Notă: Maşinile de calcul actuale implementează mai mult de 4 cicluri, pentru a creşte autonomia anumitor subsisteme, dar 4 cicluri sunt suficiente pentru un calculator]

4-DS (Calculatoare)

- Arhitectura *Von Neumann* este chiar acest system 4-DS care închide un ciclu peste un processor şi RAM:



Arhitectura
Von Neumann
(curs 1)

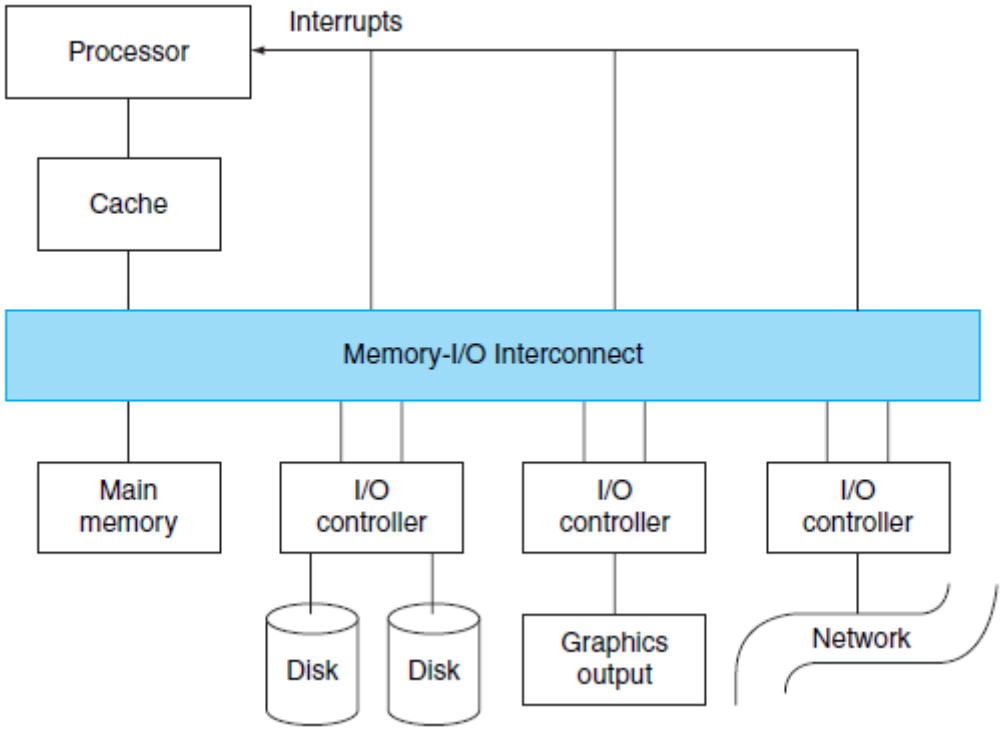


Arhitectura
Von Neumann
(curs 13)

[Nota: am ignorat semnalele de Read / Write / Wait…]

Calculatoare

- De fapt, pentru noi un calculator arată mai degrabă aşa:



[COD]

Dispozitive I/O

- Despre memorie (cache, memoria principală) am mai discutat, ne concentrăm asupra sistemelor de intrare-ieşire (I/O)

- *Întrebare:* Cum comunică sistemele I/O cu procesorul, memoria şi sistemul de operare (OS)?

- OS comunică cu dispozitivele I/O şi nu permite programelor utilizator să comunice direct cu acestea din diferite motive:
  - ✓ Un program nu trebuie să poate accesa decât dispozitivele I/O pentru care are drepturi (ex.: nu orice program poate scrie pe disc)
  - ✓ OS realizează accesul la dispozitivele I/O partajate pentru a creşte productivitatea (foloseşte eficient resursele partajate)

Dispozitive I/O

- Procesorul trebuie să comunice cu dispozitivele I/O, i.e. să trimită comenzi; există 2 metode: *comenzi I/O în memorie* şi *instrucţiuni I/O speciale*

- *Instrucţiuni I/O speciale:*
  - ✓ sunt transmise direct de la processor la dispozitivul de I/O pe magistrale
  - ✓ conţin atât dispozitivul căreia i se adresează cât şi comanda (sau locaţia comenzii în memorie)
  - ✓ programele utilizator nu pot transmite comenzi special I/O pentru că necesită modul *superuser* (*kernel*)

Dispozitive I/O

- *Comenzi I/O în memorie:*
  - ✓ spaţiul de adrese prezintă zone dedicate dispozitivelor I/O şi citirea sau scrierea din aceste locaţii sunt interpretate drept comenzi
  - ✓ unitatea de control a dispozitivului I/O consideră informaţia din zona de memorie ca o secvenţă de instrucţiuni pentru dispozitivul I/O (datele pot fi de asemenea transmise similar)
  - ✓ programele utilizator nu pot transmite comenzi pentru că OS le restricţionează accesul la zona de memorie destinată dispozitivelor I/O

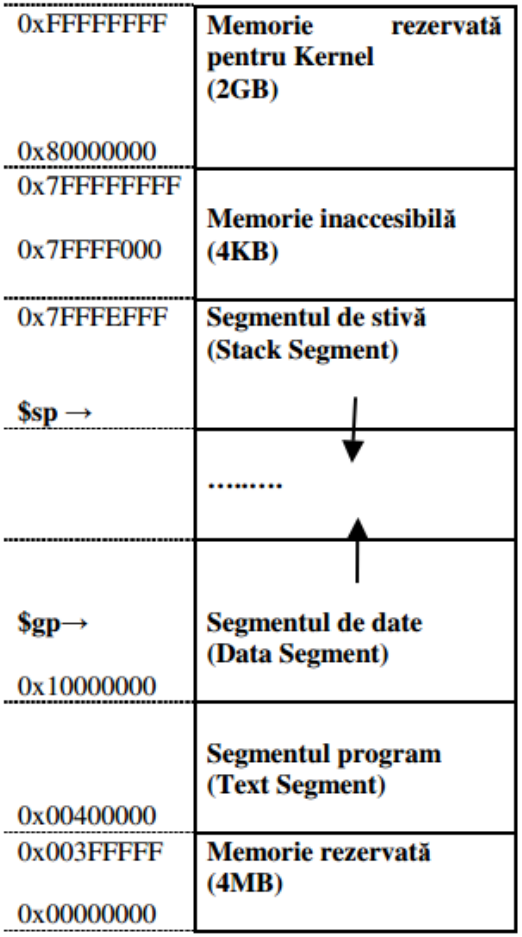
- Spre exemplu, fie o imprimantă care expune 2 regiştri:
  - ✓ **Status** – conţine un bit care să indice că a terminat operaţia (**done bit**) şi unul care să indice o eroare (**error bit**)
  - ✓ **Data** – conţine caracterul care se trimite la printat (pe 8 biţi)

În acest exemplu, procesorul trimite pentru printare un caracter, apoi verifică periodic done bit şi când este setat poate transmite caracterul următor

Dispozitive I/O - MIPS

- MIPS foloseşte comenzi I/O în memorie
- Spaţiul de adrese este partajat între memorie şi dispozitivele de I/O: unele adrese reprezintă locaţii de memorie, altele regiştrii ale dispozitivele de I/O
- Astfel nu sunt necesare instrucţiuni speciale, ci operaţiile I/O se realizează folosind instrucţiuni care pot referenţia memoria
- În MIPS adresele 0xffff0000 - 0xffffffff sunt rezervate pentru dispozitivele I/O

Dispozitive I/O - MIPS



- Ultimii 64KB sunt rezervaţi pentru dispozitivele de I/O si pot fi accesaţi doar de catre kernel

```
# Citire de la un dispozitiv I/O
lbu     $t0, 0xffff0000

# Scriere catre un dispozitiv I/O
sb      $t0, 0xffff0004
```

Mai multe info:
http://www.cs.uwm.edu/classes/cs315/Bacon/Lecture/HTML/ch14s03.html

Dispozitive I/O

- Procesorul trebuie să verifice periodic statusul dispozitivele I/O; există 2 metode: *interogări periodice (pooling)* şi *întreruperi*

- *Interogări periodice (pooling) :*
  - ✓ dispozitivele I/O modifică informaţia din registrul de stare, care este citit periodic de către processor
  - ✓ un exemplu este cel cu imprimanta de mai înainte
  - ✓ dezavantajul este că procesorul poate să citească de multe ori registrul de status până când dispozitivul este gata să preia o nouă comandă (procesorul este mai rapid decât dispozitivele I/O şi deci se consumă din timpul procesorului)

Dispozitive I/O

- *Întrebare:* Presupunem că o operaţia de polling durează 400 cicli pentru un procesor are 500 MHz. Care este rata de utilizare a procesorului la interograrea periodică a unui mouse accesat de 30 ori/s? Dar a unui hard disk care transferă câte 16B, cu rata 4 MB/s?

- *Răspuns:*

Mouse: $nr.cicli = 30 \cdot 400 = 12000;\ rata\ util. = \frac{12000}{500 \cdot 10^6} = 0.002\%$

Hard disk: $nr.accese\ per\ secunda = 4 \cdot 10^6/16 = 0.25 \cdot 10^6;$

$nr.cicli = 0{,}25 \cdot 10^6 \cdot 400 = 10^8; rata\ util. = \frac{10^8}{500 \cdot 10^6} = 20\%$

- *Observaţie*: pentru sistemele lente, procesorul face multe verificări fără succes

Dispozitive I/O

- *Întreruperi:*

  - ✓ un dispozitiv I/O transmite o întrerupere când vrea să notifice procesorul că a finalizat o operaţie sau necesită ceva

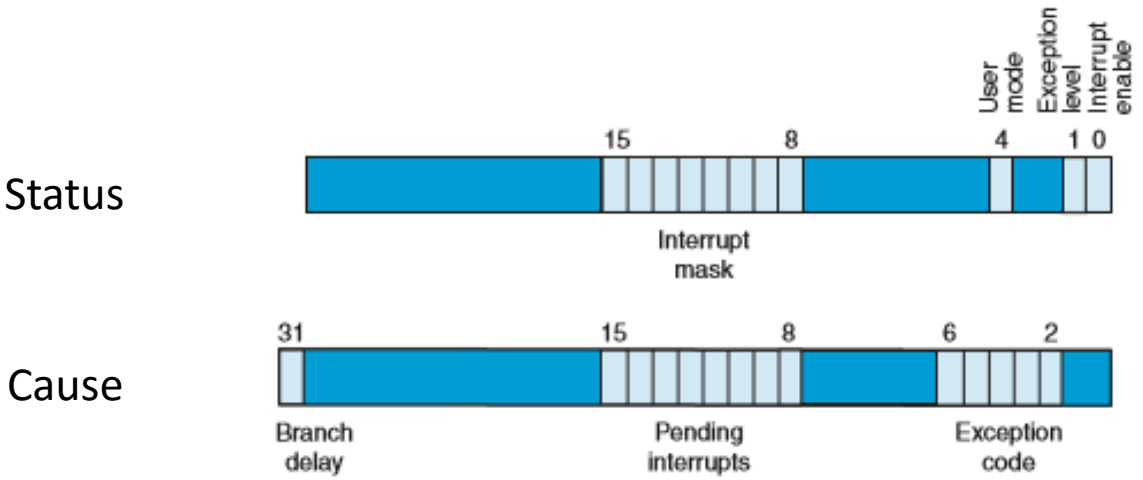
  - ✓ o întrerupere I/O este similar întreruperilor associate instrucţiunilor (ex.: overflow) cu 2 diferenţe:

1) este asincronă faţă de execuţia instrucţiunilor, i.e. nu este asociată unei instrucţiuni şi nu opreşte execuţia unei instrucţiuni

2) prezintă priorităţi diferite, în funcţie de dispozitivul care a generat-o

Dispozitive I/O - MIPS

- Regiştrii **Status** şi **Cause** în MIPS32:
  - ✓ dacă bitul Interrupt Enable este 0, nimic nu poate cauza întrerupere
  - ✓ un bit din Interrupt Mask (reg.Status) corespunde unui bit din Pending Interrupts (reg.Cause)
  - ✓ pentru a activa o întrerupere, în Interrupt Mask trebuie să fie 1 la bitul corespunzător
  - ✓ sistemul de operare găseşte motivul întreruperii în Exception Code



Dispozitive I/O - MIPS

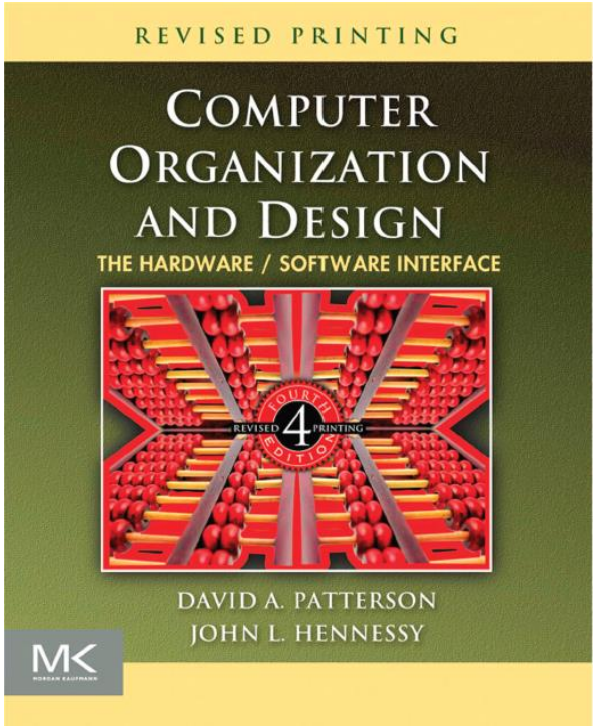
- Sistemul de operare implementează politica de tratare a priorităţilor (*IPL* = *I*nterrupt *P*riority *L*evels)
- Priorităţile sunt menţinute în memoria procesului, fiecare proces având asignată o prioritate
- Prioritate minimă înseamnă că orice întrerupere e permisă, prioritate maximă înseamnă că toate intreruperile sunt blocate; modificarea se face prin schimbarea Intrrrupt Mask în registrul Status
- În UNIX sunt 6 nivele de prioritate (în general în toate OS dispozitivele de viteză mare au prioritate mai mare)

Dispozitive I/O

- *DMA* (*D*irect *M*emory *A*ccess) este un mecanism care permite dispozitivelor I/O să transfere date direct spre / din memorie fără să fie necesară implicarea (permanentă) a procesorului

- Mecanismul de întrerupere este folosit între dispozitiv şi processor numai la **iniţializare** (când setează numele dispozitivului, adresa şi lungimea datelor de transfer), la **finalizare** şi dacă apar **erori**

Referințe bibliografice



[COD] D. Patterson and J. Hennessy, Computer Organisation and Design

[AAT] A. Atanasiu, Arhitectura calculatorului

Schemele [Xilinx - ISE] au fost realizate folosind
http://www.xilinx.com/tools/projnav.htm

Grafurile [JFLAP] au fost realizate folosind
http://www.jflap.org/