CURS 0x0A

GPU

- **este o unitate separată de calcul**
- **este conectată la un BUS de comunicare**
- **complementează capacitatea de calcul a unui CPU**
- **conțin un număr mare unități de procesare separate**
  - perfecte pentru procesarea (uniformă) unui bloc mare de date
- **sunt specializați pe un anumit set de instrucțiuni**
  - în general, nu pot realiza tot ce poate un CPU
  - dar ce pot executa, executa mult mai repede
    - SIMD/MIMD
    - hardware multithreading
    - Instruction Level Parallelism (ILP)
- **are propria sa memorie (on chip)**
  - este comparabilă cu cea a CPU (dar de obicei mai mică)
  - ierarhizare memoriei este mai simplă
  - timpii de acces nu sunt cea mai importantă caracteristică
  - bandwidth-ul (lățimea benzii de acces) este mai importantă

TIMPUL DE EXECUȚIE

- **o definiție inițială pentru performanță: timp de execuție**
- **rulăm un program A pe un sistem de calcul X**
  - performanța$_X$ = (timpul de execuție a lui A pe X)$^{-1}$
  - timp de execuție mai mic → performanță mai mare
  - în general, vrem să comparăm și să spunem: sistemul de calcul X este de *n* ori mai rapid decât sistemul de calcul Y
    - performanța$_X$ (performanța$_Y$)$^{-1}$= *n*
  - timpul de execuție îl măsurăm în secunde
    - se numește *wall-clock time*, *response time* sau *elapsed time*
    - este timpul în “lumea reală”
    - de ce e complicat? avem mai mulți timpi?
      - un procesor execută simultan mai multe programe
      - *CPU time* = cât timp de execuție a fost alocat pe CPU

PERFORMANȚA CPU

- **pentru CPU**
  - frecvența (*clock rate*), ex. 4GHz
  - ciclu de ceas (*clock cycle*), ex. la fiecare 250 pico secunde (ps)
    - (frecvența)$^{-1}$

- **CPU time pentru A = ciclii de ceas pentru A / frecvența**
  - deci, putem micșora timpul de execuție pentru A dacă
    - reducem numărul de ciclii de ceas necesar pentru a executa A
    - mărim frecvența procesorului

- **ciclii de ceas pentru A = număr instrucțiuni în A × număr ciclii necesar pentru a executa o instrucțiune (în medie)**

CPU time pentru A = număr instrucțiuni în A × număr ciclii necesar pentru a executa o instrucțiune (în medie) / frecvența

CISC VS. RISC

- **Complex Instruction Set Computers**
- **Reduced Instruction Set Computers**

CISC	RISC
ISA original	ISA apărută în anii ’80, popularitate ridicată acum (RISC-V)
hardware complicat	software complicat
instrucțiuni complicate (au nevoie de mai mulți cicli de ceas), lungime variabilă pentru instrucțiuni	instrucțiuni simple (fiecare are nevoie de un singur ciclu de ceas), lungime fixă pentru instrucțiuni
multe operații au loc memorie-memorie (citirea/scrierea în memorie este incorporată în instrucțiuni)	multe operații au loc registru-registru
suportă multe metode de adresare	metode simple (și puține) de adresare
cod scurt (puține instrucțiuni)	cod lung (multe instrucțiuni)
logica este complexă (tranzistori mulți)	logica este simplă (tranzistorii sunt alocați pentru memorie, etc.)

Criteriul PERFORMANȚA PER WATT

RISC domină clar CISC

Complex Instruction Set Computers

- **x86**
- **Motorola**

Reduced Instruction Set Computers

- **MIPS (Microprocessor without Interlocked Pipelined Stages)**
- **Power PC**
- **Atmel AVR (mașini Harvard)**
- **PIC Microchip**
- **ARM (Advanced RISC Machine)**
- **RISC-V**

Lectura suplimentară

- **1.2 Eight Great Ideas in Computer Architecture** ✅
  - **Design for Moore’s Law**
  - **Use Abstraction to Simplify Design**
  - **Make the Common Case Fast**