

# Evaluarea performantelor calculatoarelor

## Benchmarks

LAB 11

ANCA APATEAN - AC - UTCN

# Tematica : Benchmarks

ANCA APATEAN - AC - UTCN

## Introducere

Introducere – reprezentarea informatiei in calculator. Tipuri de calculatoare

Istoria calculatoarelor. Calculatorul von Neumann.

Arhitectura calculatoarelor personale/ PC. Istoria familiei de procesoare x86

## Din interiorul PC-ului

Memoria in PC.  
Memoria cache.

Echipamente periferice.  
Control transfer de date.  
Polling, Intreruperi,  
DMA.

Bus-uri si interfete folosite in PC.  
ISA, PCI, PCIe,  
IDE/ATA, SCSI, RS232,  
USB, IEEE 1284, etc

## Din exteriorul PC-ului

Echipamente de stocare date: FD, HDD, SSD, CD/DVD, Flash USB

Echipamente de intrare-iesire: monitor, tastatura, mouse, interfata grafica, interfete audio, etc

Identificarea si configurarea resurselor PC

## Cresterea performantelor PC-ului

Clasificarea Flynn.  
Paralelismul in prelucrarea datelor.  
CISC vs RISC

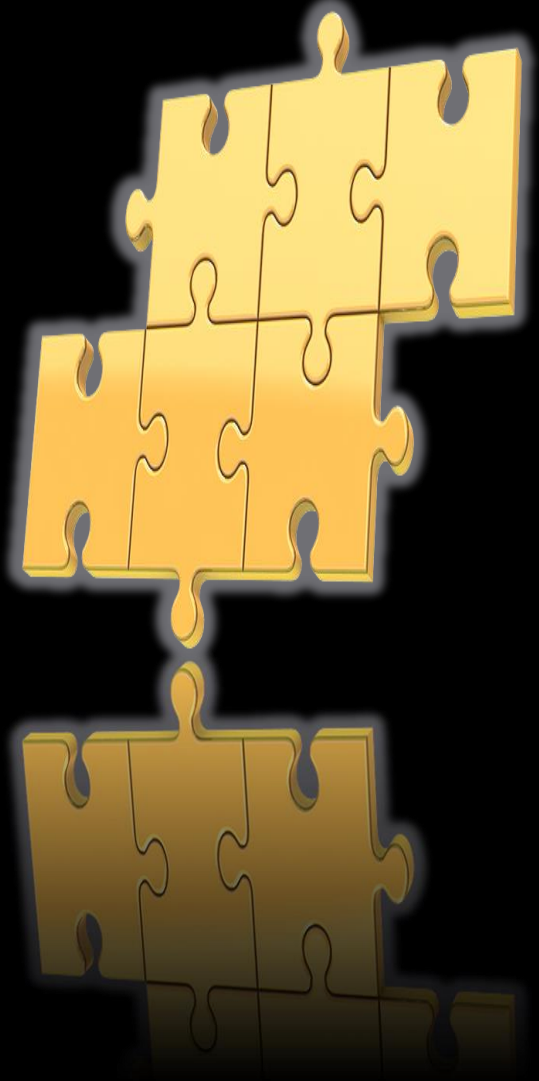
Arhitecturi VLIW, EPIC, Prelucrare secventiala si secvential-paralela

Pipeline/ Superpipeline.  
Scalar/Superscalar

Comparatie arhitecturi procesoare de uz general (GPP), DSP, MicroC, DSC, SoC.

## Evaluarea performantelor PC-ului

Evaluarea performantelor calculatoarelor. Benchmark-uri.



- Prezentarea notiunilor principale despre modul **cum se pot compara performantele unui PC**
- Familiarizarea cu metodele de evaluare existente

Familiarizarea cu Metricile uzuale folosite la evaluarea performantelor PC-urilor, precum : **Timpul de execuție , Timpul CPU,**

**MIPS, MFLOPS, Legea lui Amdahl**

si indrumarea spre exemple de calcul folosind acesti indicatori

- **urmarirea principalelor aspecte legate de folosirea unor programe reale de evaluare a performantelor PC-urilor**

### Sistemul A este mai bun decat sistemul B ?

Important dpdv al: **inginerului** proiectant de sistem,  
dar si al **vanzatorului** sau al **cumparatorului** dintr-o firma,  
sau al **utilizatorului** obisnuit

=>cunoasterea si utilizarea unor

Programe reale de evaluare a performanțelor

Care se pot baza pe metrici precum:

1. Timpul de execuție
2. Timpul CPU
3. MIPS
4. MFLOPS
5. Legea lui Amdahl

## Informatia in Evaluarea performantelor calculatoarelor

Metrici folosite

1. Timpul de execuție

2. Timpul CPU

3. MIPS

4. MFLOPS

5. Legea lui Amdahl

Programe reale de evaluare a performantelor

Evaluarea performantelor calculatoarelor nu poate avea

(nu trebuie sa aiba) legatura cu:

#### Informatia incompleta

“Produsul nostru a fost testat si rezultatele publicate. Compania rivala X nu a publicat rezultatele aceluiasi test, deci produsul nostru e mai bun (adica mai rapid).”

- nu exista nici o informatie relevanta, de incredere - nu exista date concrete !

#### Informatia vaga si masuratorile insuficiente sau necorespunzatoare

- folosirea cuvintelor imprecise, precum “mai mult/mare”, “mai putin/mic”, “aproape/practic (la fel de)”, sau alte constructii sinonime, fara asocierea cu date concrete -> semnal alarma !

#### Apelarea la popularitate

“Produsele noastre sunt folosite de 80% din cele mai puternice 100 companii din lume”

- nu inseamna decat ca au un renume asigurat, nu neaparat si ca respectivul produs se potriveste nevoilor utilizatorului care vede reclama

Un sistem cu procesor la 2,5 GHz – inseamna ca e mai bun ca un sistem cu procesor la 2,0 GHz?

NU neaparat

Si atunci ... Cum se evalueaza performanta calculatoarelor ?

Performanța unui calculator **nu este ușor de evaluat** având în vedere numărul și diversitatea arhitecturilor existente pentru sistemele de calcul.

- Mai multe **fatete ale performanței** unui calculator:

- din prisma unui utilizator obișnuit:

se măsoară performanța pe baza **timpului** necesar pt a executa un anumit program (o sarcină, un task) = *timpul<sub>executie</sub>* sau *timpul<sub>raspuns</sub>*

(“Care sistem a terminat mai repede? A sau B?”)

- din prisma unui inginer (sau al unui manager de callcenter):

performanța prin **banda/debit** (rata=bandwidth, putere de calcul=throughput) :  
cantitatea totală de sarcini executate în unitatea de timp

(“Care sistem a executat mai multe sarcini într-o zi?”)

În [Patterson2013] – analogia cu sistemul de transport aerian:

**Care din cele 3 sisteme e mai bun?**

- **Concorde** – **viteza** cea mai ridicată

- **DC-8** – **curșa/gama** cea mai lungă

- **747** – **capacitatea** cea mai mare

Performanța – măsurată prin diferite aspecte! Dacă se cunosc cele 3 aspecte pentru fiecare din cele 3 sisteme, se poate calcula un coeficient precum **rata la care avionul transporta pasageri** = *capacitatea* × *viteza* (ignorând deci curșa/gama)

- pe baza acestui coeficient se poate oarecum decide care e mai bun

A. Care e cel mai bun sistem **dpdv al unui singur pasager**? B. Dar **in medie pe un număr mare de pasageri**?

Răspuns: A. **Concorde** (viteza de cel puțin 2x mai ridicată ca 747), B. **747** deoarece are capacitate mai mare de transport

Performanța unui calculator **nu este ușor de evaluat** având în vedere numărul și diversitatea arhitecturilor existente pentru sistemele de calcul.

- Mai multe **fatete ale performanței** unui calculator:

- din prisma unui utilizator obișnuit:

se măsoară performanța prin  **timpul**  necesar pt a executa un anumit program (o sarcină, un task) = *timpul<sub>execuție</sub>* sau *timpul<sub>raspuns</sub>*  
(“Care sistem a terminat mai repede? A sau B?”)

- din prisma unui inginer (sau al unui manager de callcenter):

se măsoară performanța prin **banda** sau **debit** (bandwidth, putere de calcul=throughput) :  
cantitatea totală de **sarcini executate** în unitatea de timp  
(“Care sistem a executat mai multe sarcini într-o zi?”)

Dar în general, **performanța** este evaluată din prisma **timpului**

**necesar PROCESORULUI** să execute o anumită sarcină (un task) predefinită,

- există și alte metrici (**utilizarea memoriei** sau **consumul de energie**)

la fel de importante în anumite aplicații.

**1. Timpul de execuție**

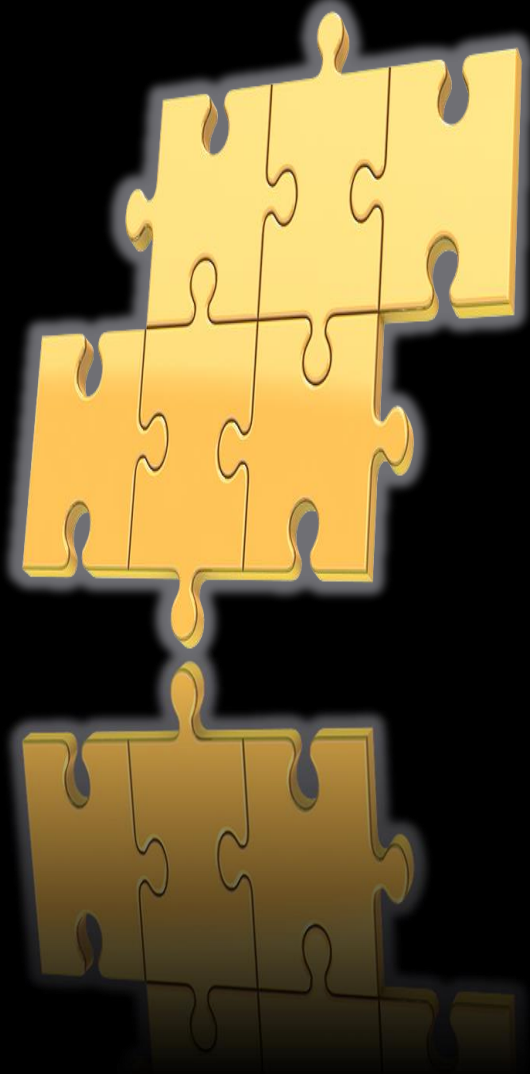
**2. Timpul CPU**

**3. MIPS**

**4. MFLOPS**

**5. Legea lui Amdahl**





## Informația în Evaluarea performanțelor calculatoarelor

### Metrice folosite

1. Timpul de execuție

2. Timpul CPU

3. MIPS

4. MFLOPS

5. Legea lui Amdahl

Programe reale de evaluare a performanțelor

**Timpul de execuție** este măsura cel mai des folosită pentru evaluarea performanțelor sistemelor de calcul: calculatorul care obține timpul de execuție (**măsurat în secunde**) cel mai scurt la executarea unui set de operații, este considerat cel mai performant.

Timpul de execuție poate lua diferite forme particulare în măsurarea duratei anumitor evenimente, un astfel de exemplu fiind **timpul de răspuns**

- definit ca **timpul necesar terminării unei sarcini**

(de ex. timpul în care sistemul reușește să furnizeze un rezultat) și cuprinde:

- timpul necesar accesării memoriei,
- timpul necesar operațiilor de I/O și
- timpul necesar operațiilor executate de sistemul de operare (S.O.).

În cazul sistemelor care folosesc multiprogramarea, acești timpi se întrepătrund, nemaiputând fi evaluați corect => este necesară introducerea unui alt indicator ( $timpul_{CPU}$ ) care să indice **timpul efectiv în care CPU execută un program**, fără a lua în considerare **timpul de așteptare pentru operații de I/O** sau **timpul în care CPU execută alte programe**.

De asemenea,  $timpul_{CPU}$  poate fi divizat în:

- $timpul_{CPU_{utilizator}}$  corespunzător **execuției programului utilizator** și
- $timpul_{CPU_{sistem}}$  necesar **execuției funcțiilor apelate de S.O.**

! ceea ce percepe utilizatorul NU este  $timpul_{CPU}$

Atunci când se dorește **compararea a două sisteme de calcul (A și B)**:

- se spune despre sistemul A ca este **mai rapid** decât sistemul B dacă **timpul de execuție al unui program este mai scurt** pentru sistemul A decât pentru sistemul B:

- “computerul A este **de n ori mai rapid** decât computerul B”:

$$\frac{t_E(B)}{t_E(A)} = n \quad (1)$$

Astfel, raționamentul „calculatorul A este **cu n% mai rapid** decât calculatorul B” poate fi exprimat de relația:

$$\frac{t_E(B)}{t_E(A)} = 1 + \frac{n}{100} \quad (2)$$

unde  $t_E$  = timpul de execuție.

- timpul de execuție = inversul performanței ( $t_E = 1/P$ ):

$$\frac{t_E(B)}{t_E(A)} = \frac{P(A)}{P(B)} = 1 + \frac{n}{100} \quad (3)$$

**Creșterea performanței** (notată  $n$ ) este exprimată ca diferența dintre performanța calculatorului mai rapid și cea a calculatorului mai lent, raportată la performanța calculatorului mai lent:

$$n = \frac{P(A) - P(B)}{P(B)} \cdot 100 = \frac{t_E(B) - t_E(A)}{t_E(A)} \cdot 100 \quad (4)$$

s-a presupus că dintre cele două calculatoare A și B, cel mai lent este B

Atunci când se dorește compararea a două sisteme de calcul:

- se spune despre sistemul A ca este mai rapid decât sistemul B dacă *timpul de execuție al unui program este mai scurt* pentru sistemul A decât pentru sistemul B:

- computerul A este de **n** ori **mai rapid** decât computerul B

Dacă se folosește **banda**, atunci spunem:

**Banda** (rata sau puterea de calcul) lui A e **de n ori mai mare** decât **banda** lui B

și înseamnă că nr de sarcini rezolvate în unitatea de timp folosind sistemul A este de n ori mai mare decât cele rezolvate folosind sistemul B.

- în gen,  $n > 1$  (supraunitar)

- timpul în gen se raportează la completarea unui task (sarcini) precum:

acces la disc,

acces la memorie,

activitatea unui periferic I/O



**Exemplul 00h:**

Dacă **calculatorul A** execută un program în **15 secunde**, iar **calculatorul B** execută același program în **20 secunde**, cu cât va fi mai rapid calculatorul A față de calculatorul B?

**Soluție:** Se calculează creșterea performanței după relația (4) astfel:

$$n = \frac{P(A) - P(B)}{P(B)} \cdot 100 = \frac{t_E(B) - t_E(A)}{t_E(A)} \cdot 100$$

$$n = \frac{t_E(B) - t_E(A)}{t_E(A)} \cdot 100 = \frac{20 - 15}{15} \cdot 100 = 33 \text{ [%]}$$

**=> calculatorul A este cu 33 % mai rapid decât calculatorul B.**



### Exemplul 01h:

Dacă un calculator A rulează un anumit program în 10 secunde și un calculator B rulează același program în 15 secunde, se cere:

- De câte ori e mai rapid calculatorul A decât calculatorul B?
- Cu cât e mai rapid calculatorul A decât calculatorul B?

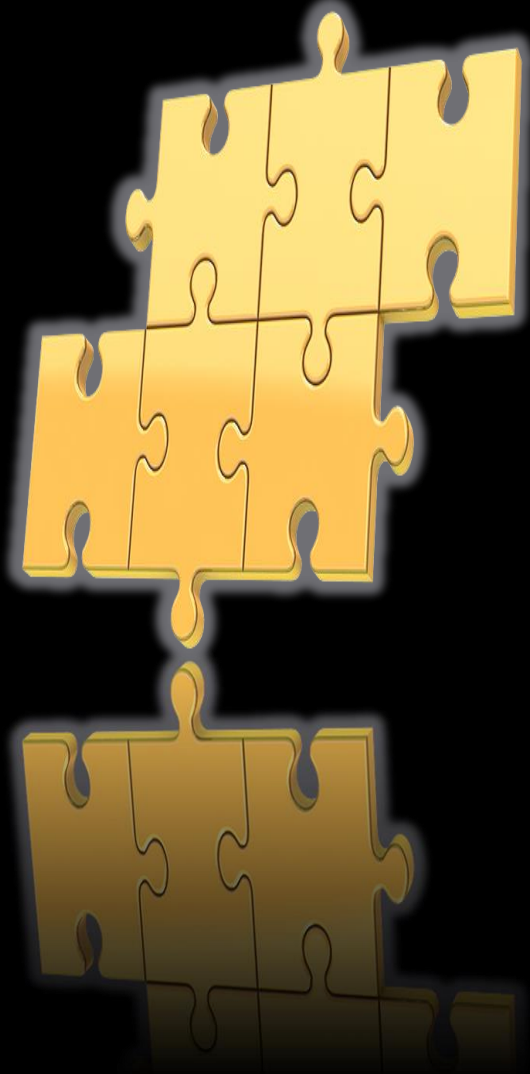
### Soluție:

$$a) n = \frac{t_E(B)}{t_E(A)} = \frac{15s}{10s} = 1,5 \Rightarrow n = 1,5$$

$\Rightarrow$  calculatorul A este de 1,5 ori mai rapid decât calculatorul B.

$$b) n = \frac{t_E(B) - t_E(A)}{t_E(A)} \cdot 100 = \frac{15 - 10}{10} \cdot 100 = \frac{5}{10} \cdot 100 = 50 \Rightarrow n = 50\%$$

$\Rightarrow$  calculatorul A este cu 50% mai rapid decât calculatorul B.



## Informația în Evaluarea performanțelor calculatoarelor

### Metrici folosite

1. Timpul de execuție

2. Timpul CPU

3. MIPS

4. MFLOPS

5. Legea lui Amdahl

Programe reale de evaluare a performanțelor

## 11. Evaluarea performanțelor calculatoarelor

### Metriци – timpul de execuție al procesorului

ANCA APATEAN - AC - UTCN

Sistemul de calcul folosește un semnal de ceas pentru a raporta și sincroniza execuția diferitelor operații.  
- acest semnal de ceas este definit de **durata ciclului de ceas** (exprimată în secunde) sau de **frecvența ceasului** (exprimată în Hz).

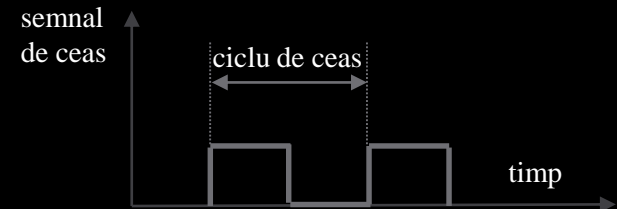
Pentru un program timpul de execuție al procesorului (*timpul<sub>CPU</sub>*) (pt un program) poate fi exprimat:

[secunde/program]

$$t_{CPU} = C_{CPU} \cdot T_C \quad (5)$$

unde  $C_{CPU}$  = numărul ciclurilor de ceas ale CPU necesare pt execuția programului, iar

$T_C$  = durata (perioada) ciclului de ceas [secunde/ciclu ceas].



- pt definierea timpului: ciclul de ceas (clock cycle) = timpul între 2 fronturi crescătoare (sau descrescătoare) consecutive ale unui semnal de ceas periodic

= măsura utilizată pt a specifica timpul necesar execuției unui anumit task

- *timpul<sub>CPU</sub>* poate fi exprimat în funcție de **frecvența ceasului**  $f_C = 1/T_C$ :

$$t_{CPU} = \frac{C_{CPU}}{f_C} \quad (6)$$

- fie  $N$  = numărul de instrucțiuni executate de program

=> **CPI** = numărul mediu de cicluri de ceas pe instrucțiune:

$$CPI = \frac{C_{CPU}}{N} \quad (7)$$

Din (5) =>

$$t_{CPU} = N \cdot CPI \cdot T_C \quad (8)$$

sau

$$t_{CPU} = \frac{N \cdot CPI}{f_C} \quad (9)$$



Uneori este utilă calcularea **numărului total de cicluri de ceas ale CPU**, acesta obținându-se din relația:

$$C_{CPU} = \sum_{i=1}^n (CPI_i \cdot I_i) \quad (10)$$

unde  $CPI_i$  = numărul mediu de cicluri de ceas necesar pentru execuția instrucțiunii  $i$ , iar  
 $I_i$  = numărul de execuții ale instrucțiunii  $i$  într-un program.

=> timpul  $t_{CPU}$  din relația (5) poate fi exprimat

$$t_{CPU} = T_C \cdot \sum_{i=1}^n (CPI_i \cdot I_i) \quad (11)$$

iar nr total de cicluri pe instrucțiune  $CPI$  din relația (6) devine:

$$CPI = \frac{\sum_{i=1}^n (CPI_i \cdot I_i)}{N} = \sum_{i=1}^n \left( CPI_i \cdot \frac{I_i}{N} \right) = \sum_{i=1}^n (CPI_i \cdot F_i) \quad (12)$$

- unde  $F_i$  = frecvența cu care instrucțiunea  $i$  apare în program



### Exemplul 02h:

Fie frecvențele de utilizare a instrucțiunilor pe care le poate executa un sistem de calcul cele exprimate în tabelul următor:

Tip instrucțiune	Frecvența	Cicluri ceas
UAL	50%	1
Încărcare (load)	20%	2
Memorare (store)	10%	2
Salt	10%	3
altele	10%	2

Care va fi numărul total mediu de cicluri pe instrucțiune ?

### Soluție:

Numărul mediu de cicluri pe instrucțiune se calculează cu relația (12) astfel:

$$CPI = 1 \cdot 0,5 + 2 \cdot 0,2 + 2 \cdot 0,1 + 3 \cdot 0,1 + 2 \cdot 0,1 = 1,6 \text{ [cicluri/instrucțiune]}$$

deci *în medie*, pentru execuția unei instrucțiuni, vor fi necesare 1,6 cicluri de ceas.

**Exemplul 03h**

Un program rulează în **10 secunde** pe calculatorul **A**, care are frecvența de ceas de **2GHz**. Un alt calculator **B**, rulează același program în doar **7 secunde**. Pentru a obține această reducere a timpului, inginerul proiectant a descoperit că este necesară o creștere a frecvenței ceasului, afectând restul arhitecturii CPU. Dacă sistemul B va necesita **de 1,4 ori mai multe cicluri ceas** decât sistemul A pentru acel program, care este frecvența ceasului pe care ar trebui să o urmărească proiectantul ?

**Soluție:**

Numărul ciclurilor de ceas necesare pt rularea programului pe calculatorul A este:

$$t_{CPU_A} = \frac{C_{CPU_A}}{f_{C_A}} \Rightarrow 10s = \frac{C_{CPU_A}}{2 \cdot 10^9 \frac{\text{cicluri}}{\text{secundă}}} \Rightarrow C_{CPU_A} = 10 \cdot 2 \cdot 10^9 \text{ cicluri} = 20 \cdot 10^9 \text{ cicluri}$$

Numărul ciclurilor de ceas necesare pt rularea programului pe calculatorul B este de 1,4 ori mai mare.

$$t_{CPU_B} = \frac{1,4 \cdot C_{CPU_A}}{f_{C_B}} \Rightarrow 7s = \frac{1,4 \cdot 20 \cdot 10^9 \text{ cicluri}}{f_{C_B}} \Rightarrow f_{C_B} = \frac{1,4 \cdot 20 \cdot 10^9 \text{ cicluri}}{7s} = \frac{0,2 \cdot 20 \cdot 10^9 \text{ cicluri}}{s} \Rightarrow f_{C_B} = 4 \cdot 10^9 \frac{\text{cicluri}}{s}$$

$$\Rightarrow f_{C_B} = 4GHz.$$

Pentru a rula programul în 7 secunde, calculatorul B trebuie să dubleze frecvența de ceas corespunzătoare calculatorului A.



### Exemplul 04h

Fie două implementări ale aceluiași set de instrucțiuni.

Calculatorul A are perioada ciclului de 0,3 ns și execută în medie 2 cicluri pe instrucțiune pentru un anumit program, iar calculatorul B are perioada ciclului 0,6ns și execută 1,2 cicluri pe instrucțiune pentru același program.

Care calculator este mai rapid pentru acest program și de câte ori?

### Soluție:

Știm că numărul de instrucțiuni al programului este același = N.

$$C_{CPU} = CPI \cdot N \Rightarrow C_{CPU_A} = N \cdot 2$$

$$C_{CPU_B} = N \cdot 1,2$$

$$t_{CPU} = C_{CPU} \cdot T_C = CPI \cdot N \cdot T_C \Rightarrow t_{CPU}(A) = N \cdot 2 \cdot 0,3ns = 0,6N(ns)$$

$$t_{CPU}(B) = N \cdot 1,2 \cdot 0,6ns = 0,72N(ns)$$

⇒ calculatorul A este mai rapid

$$\frac{\text{performanța CPU}_A}{\text{performanța CPU}_B} = \frac{t_E(B)}{t_E(A)} = \frac{720N}{600N} = 1,2$$

⇒ calculatorul A este de 1,2 ori mai rapid decât calculatorul B pentru acel program



### Exemplul 05h

Un proiectant de compilatoare încearcă să se decidă între **două secvențe de cod** pentru un anumit calculator. Proiectantul hardware i-a dat următoarele informații:

Se cere:

- Care secvență de cod execută mai multe instrucțiuni?
- Care secvență de cod va fi mai rapidă?
- Cât este CPI pentru fiecare secvență?

**Soluție:**

- Secvența 1 execută:  $2 + 1 + 3 = 6$  instrucțiuni (=N1)  
 Secvența 2 execută:  $4 + 2 + 1 = 7$  instrucțiuni (=N2)  
 $\Rightarrow$  secvența 2 execută mai multe instrucțiuni.

- $C_{CPU} = \sum_{i=1}^n (CPI_i \cdot N_i) \Rightarrow C_{CPU_1} = (2 \cdot 1) + (1 \cdot 2) + (3 \cdot 3) = 2 + 2 + 9 = 13$  cicluri  
 $C_{CPU_2} = (4 \cdot 1) + (2 \cdot 2) + (1 \cdot 3) = 4 + 4 + 3 = 11$  cicluri  
 $\Rightarrow$  secvența de cod 2 este mai rapidă, deși execută cu o instrucțiune mai mult.

c)

$$CPI = \frac{C_{CPU}}{N} \Rightarrow CPI_1 = \frac{13}{6} = 2,16$$

$$CPI_2 = \frac{11}{7} = 1,57$$

*CPI pt fiecare clasă de instrucțiuni*

	A	B	C
CPI	1	2	3

*Nr de instrucțiuni pt fiecare clasă de instrucțiuni*

Secvența de cod	A	B	C
1	2	1	3
2	4	2	1



### Exemplul 06h

O aplicație (un program) rulează în 20 secunde pe un calculator.

Apare o nouă versiune a aplicației care necesită doar 0,7 instrucțiuni din numărul primei versiuni. Din păcate, acesta crește valoarea CPI cu 10% din valoarea CPI inițială.

- a) Cât de rapid ne așteptăm să ruleze această nouă versiune?
- b) Cat timp se castiga cu aceasta noua versiune?

### Soluție:

$$t_E = N_{nou} \cdot CPI_{nou} \cdot T_C = 0,7N \cdot 1,1CPI \cdot T_C = 0,7 \cdot 1,1 \cdot 20 \Rightarrow 15,4 \text{ secunde}$$

20sec ... 100%

(20-15,4) sec ... X  $\Rightarrow$  se castiga 23% din timp



ANCA APATEAN - AC - UTCN

### Exemplul 07h:

Pentru sistemul de calcul din **exemplul 02h** -> se pp. că **20%** dintre **instrucțiunile UAL** vor fol un operand încărcat din memorie (este deja încărcat în unul dintre regiștrii și nu mai e nevoie de o operație de Load).

Ca o metodă de îmbunătățire a performanțelor acestui sistem de calcul se propune **adăugarea unor instrucțiuni UAL** care **au un operand sursă în memorie** (deci nu mai implica si instrucțiuni de load), dar care vor necesita **2 cicluri de ceas**. Se pp. că **setul extins de instrucțiuni va crește cu 1 numărul ciclurilor de ceas pentru execuția instrucțiunilor de salt**, dar nu afectează durata ciclului de ceas.

Care va fi noul număr mediu de cicluri pe instrucțiune  $CPI_{nou}$ ? Precizați dacă modificarea propusă va crește performanța CPU.

**Soluție:** sunt cu 20% mai puține instrucțiuni UAL care nu vor mai avea nevoie nici de încărcarea unor operanzi din memorie (pentru că sunt deja disponibili)

=> se vor elimina și 20% din instrucțiunile de încărcare (Load).

În schimb, vor fi adăugate 20% instrucțiuni UAL cu regiștrii și memoria care dureaza 2 cicluri.

Instrucțiunile de memorare contribuie în același fel ca în exemplul 02h,

însă la instrucțiunile de salt se modifică numărul de cicli din 3 în 4.

=>numărul de instrucțiuni la care se raportează noua valoare se modifică: vor fi cu 20% mai puține instrucțiuni UAL.

Tip instrucțiune	Frecvența	Cicluri ceas
UAL	50%	1
Încărcare (load)	20%	2
Memorare (store)	10%	2
Salt	10%	3
altele	10%	2



ANCA APATEAN - AC - UTCN

### Exemplul 07h: Soluție (continuare)

- se reduc 20% din instruct UAL cu 1 ciclu  
=> se reduce și nr instruct Load cu 20%
- instruct de memorare și altele contribuie la fel ca în exemplul 02h,
- la instrucțiunile de salt se modifică numărul de cicli din 3 în 4.
- + trebuie adăugate noile instrucțiuni UAL cu 2 cicli
- => numărul de instrucțiuni la care se raportează noua valoare: cu 20% mai puține

Frecvențele de utilizare a instrucțiunilor		
Tip instrucțiune	Frecvența	Cicluri ceas
UAL	50%	1
Încărcare (load)	20%	2
Memorare (store)	10%	2
Salt	10%	3
altele	10%	2

$$CPI_{nou} = \frac{1 \cdot [0,5 - (0,2 \cdot 0,5)] + 2 \cdot [0,2 - (0,2 \cdot 0,5)] + 2 \cdot (0,2 \cdot 0,5) + 2 \cdot 0,1 + 4 \cdot 0,1 + 2 \cdot 0,1}{1 - (0,2 \cdot 0,5)}$$

$$CPI_{nou} = \frac{1,6}{0,9} = 1,78$$

[cicluri/instrucțiune]

Performanța sistemului înainte și după modificarea propusă se va calcula astfel:

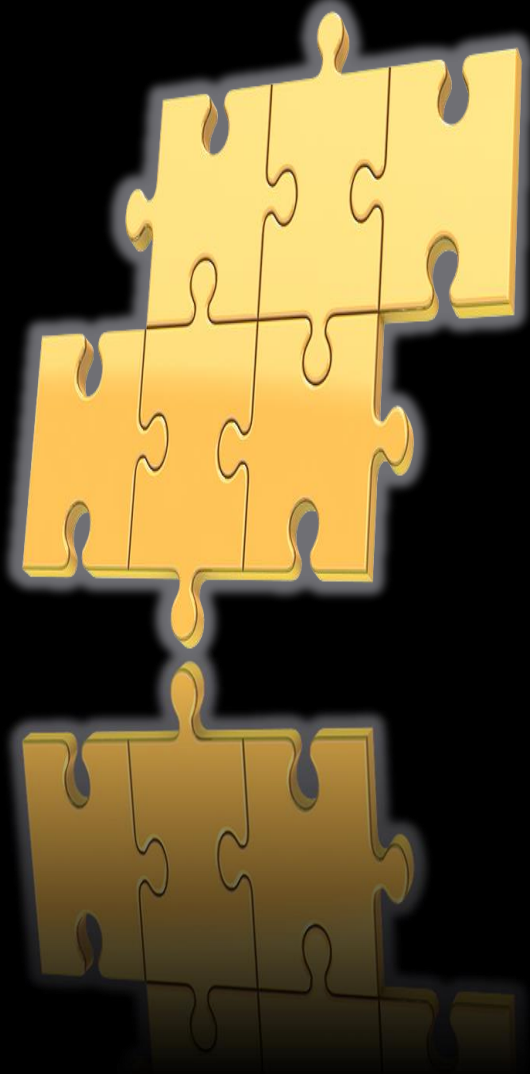
$$t_{CPU\_init} = N_{init} \cdot CPI_{init} \cdot T_{C\_init} = 1,6 \cdot N_{init} \cdot T_{C\_init}$$

$$t_{CPU\_nou} = N_{nou} \cdot CPI_{nou} \cdot T_{C\_nou} = ((1 - 0,2 \cdot 0,5) \cdot N_{init}) \cdot 1,78 \cdot T_{C\_init} = 0,9 \cdot 1,78 \cdot N_{init} \cdot T_{C\_init}$$

$$t_{CPU\_nou} = 0,9 \cdot 1,78 \cdot N_{init} \cdot T_{C\_init} = 1,6 \cdot N_{init} \cdot T_{C\_init}$$

Răspuns: prin efectuarea modificărilor propuse, reducerea cu 20% a instrucțiunilor UAL nu modifică timpii de execuție a salturilor, deci performanța sistemului nu va crește.





## Informația în Evaluarea performanțelor calculatoarelor

### Metrice folosite

1. Timpul de execuție

2. Timpul CPU

3. MIPS

4. MFLOPS

5. Legea lui Amdahl

Programe reale de evaluare a performanțelor

**Timpul de execuție** este considerat **cel mai important indicator de performanță**, fiind **consecvent și viabil**, însă de-a lungul timpului s-au mai utilizat și alți indicatori, precum **MIPS**, **MFLOPS**, etc.

**MIPS (Millions of instructions per second)** (“Milioane de instrucțiuni pe secunda”)

= o alternativă la **timpul de execuție**,

= **numărul de instrucțiuni medii pe care un calculator le poate executa pe secundă**.

=> pentru un program dat, *MIPS* se definește după relația:

$$MIPS = \frac{N}{t_E \cdot 10^6} \quad (13)$$

- Scriind relația (13) în funcție de frecvența ceasului și numărul de cicluri pe instrucțiune, și dacă  $t_E = t_{CPU}$  se obține:

$$MIPS = \frac{t_{CPU} \cdot f_C}{CPI \cdot t_E \cdot 10^6} = \frac{f_C}{CPI \cdot 10^6} \quad (14)$$

Exprimând timpul de execuție în funcție de numărul de instrucțiuni *N* și *MIPS*, se obține:

$$t_E = \frac{N}{MIPS \cdot 10^6} \quad (15)$$

În cazul în care un calculator poate executa peste un miliard de instrucțiuni pe secundă, în locul *MIPS* se poate folosi **BIPS (Billions of instructions per second)** sau **GIPS (G-giga)**, definit în mod similar.

Indicatorul *MIPS* este o metrică ușor de înțeles și folosit în evaluarea performanțelor unui sistem de calcul, însă

**NU** este indicată utilizarea lui *exclusivă* când se dorește

**compararea performanțelor mai multor sisteme care nu aparțin aceleiași familii !!!**

(adică nu au același set de instrucțiuni), deoarece:

- ***MIPS* depinde de setul de instrucțiuni**
- ***MIPS* poate varia pentru programe diferite ale aceluiași calculator**
- ***MIPS* poate varia invers proporțional cu performanța !**
- ***MIPS* nu ia în considerare timpul de execuție !**



ANCA APATEAN - AC - UTCN

### Exemplul 08h:

Pentru sistemul de calcul din **exemplul 02h** -> să se calculeze *MIPS*, știind că ceasul sistemului are **frecvența de 400 MHz**.

### Soluție:

$$\text{MIPS} = \frac{f_c}{\text{CPI} \cdot 10^6} = \frac{400 \cdot 10^6}{1,6 \cdot 10^6} = 250$$

### Frecvențele de utilizare a instrucțiunilor

Tip instrucțiune	Frecvența	Cicluri ceas
UAL	50%	1
Încărcare (load)	20%	2
Memorare (store)	10%	2
Salt	10%	3
altele	10%	2

**Exemplul 09h:**

Se dau următoarele măsurători de performanță pentru un program:

Valori obținute pentru	Calculator A	Calculator B
Număr de instrucțiuni	8 mld	6 mld
Frecvență ceas	3,6GHz	3,6GHz
CPI	1,0	1,1

Se cere:

Care calculator are MIPS mai mare?

Care calculator este mai rapid?

**Soluție:**

$$MIPS_A = \frac{f_{c(A)}}{CPI_{(A)} \cdot 10^6} = \frac{3600 \cdot 10^6}{1 \cdot 10^6} = 3600$$

$$MIPS_B = \frac{f_{c(B)}}{CPI_{(B)} \cdot 10^6} = \frac{3600 \cdot 10^6}{1,1 \cdot 10^6} = \frac{3600}{1,1} = 3272,7272$$

⇒ calculatorul A are MIPS mai mare.

$$t_{E(A)} = N_A \cdot CPI_A \cdot T_{C_A} = 8 \cdot 10^9 \cdot \frac{1}{3,6 \cdot 10^9} \cdot 1 = 2,22s$$

$$t_{E(B)} = N_B \cdot CPI_B \cdot T_{C_B} = 6 \cdot 10^9 \cdot \frac{1}{3,6 \cdot 10^9} \cdot 1,1 = 1,66 \cdot 1,1 = 1,83s$$

⇒ calculatorul B este mai rapid, deși calculatorul A are MIPS mai mare.

# 11. Evaluarea performanțelor calculatoarelor

## Metrici - MIPS (5)



ANCA APATEAN - AC - UTCN

**Exemplul 0Ah:** (exemplu în care MIPS nu e un indicator viabil în evaluarea performanțelor calculatoarelor)

Se pp. că un compilator care efectuează optimizarea codului este folosit pentru sistemul de calcul având frecvența instrucțiunilor dată în tabelul din **exemplul 02h**.

Compilatorul **elimină 50% din instrucțiunile UAL**, dar nu poate reduce numărul celorlalte tipuri de instrucțiuni. Dacă se cunoaște că **durata ciclului de ceas este 2,5 ns**, care este valoarea MIPS și timpul de execuție al procesorului pentru codul neoptimizat și pentru cel optimizat ?

Tip instrucțiune	Frecvența	Cicluri ceas
UAL	50%	1
Încărcare (load)	20%	2
Memorare (store)	10%	2
Salt	10%	3
altele	10%	2

**Soluție:**

$$CPI_{neopt} = 1,6$$

=>

$$MIPS_{neopt} = \frac{400 \cdot 10^6}{1,6 \cdot 10^6} = 250$$

$$CPI_{opt} = \frac{1 \cdot (0,5 \cdot 0,5) + 2 \cdot 0,2 + 2 \cdot 0,1 + 3 \cdot 0,1 + 2 \cdot 0,1}{1 - (0,5 \cdot 0,5)} = \frac{1,35}{0,75} = 1,8$$

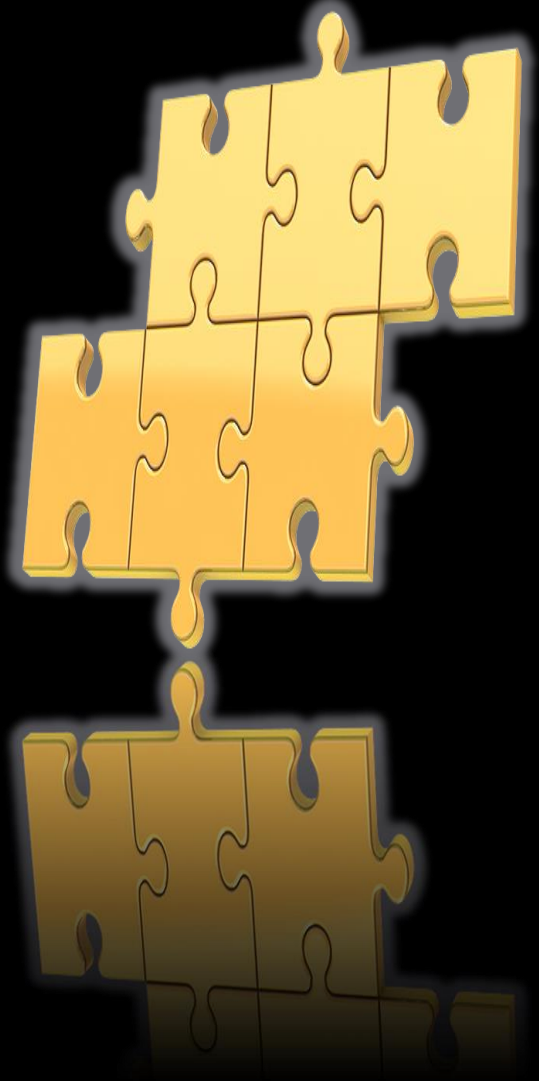
$$MIPS_{opt} = \frac{400 \cdot 10^6}{1,8 \cdot 10^6} = 222,22$$

Performanța codului neoptimizat, respectiv optimizat este:

$$t_{CPU\_neopt} = N_{neopt} \cdot CPI_{neopt} \cdot T_C = N_{neopt} \cdot 1,6 \cdot 2,5 \cdot 10^{-9} = 4 \cdot 10^{-9} \cdot N_{neopt}$$

$$t_{CPU\_opt} = N_{opt} \cdot CPI_{opt} \cdot T_C = (1 - 0,5 \cdot 0,5) \cdot N_{neopt} \cdot 1,8 \cdot 2,5 \cdot 10^{-9} = 3,375 \cdot 10^{-9} \cdot N_{neopt}$$

Codul optimizat este cu **15.6%** mai rapid (raportat la codul neoptimizat), deși valoarea MIPS coresp. este mai mică.



## Informația în Evaluarea performanțelor calculatoarelor

### Metrice folosite

1. Timpul de execuție

2. Timpul CPU

3. MIPS

4. MFLOPS

5. Legea lui Amdahl

Programe reale de evaluare a performanțelor

Metrica *MIPS* nu este adecvată procesoarelor specializate pentru **calcule matematice** precum procesoarele vectoriale.

- se folosește indicatorul **MFLOPS** (Millions of floating-point operations per second) sau **GFLOPS** (Billions of floating-point operations per second) pentru a măsura numărul operațiilor de calcul în virgulă mobilă pe care le poate executa sistemul de calcul într-o secundă.

$$MFLOPS = \frac{N_{VM}}{t_E \cdot 10^6} \quad (15)$$

unde  $N_{VM}$  este numărul de operații în VM (virgulă mobilă) dintr-un program

Similar cazului metricii *MIPS*, utilizarea *MFLOPS* ridică două probleme importante:

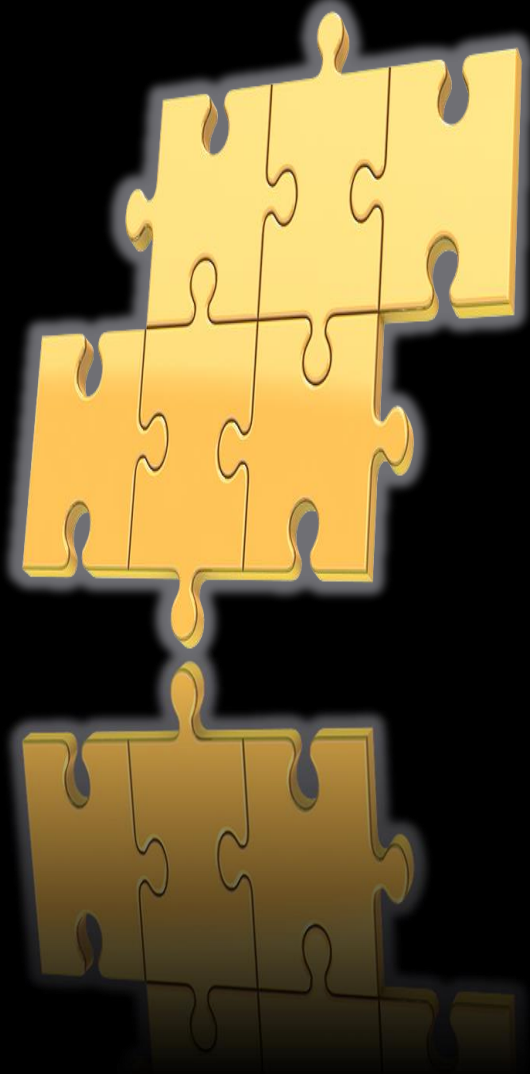
- setul de operații în virgulă mobilă diferă de la un calculator la altul și
- *MFLOPS* se modifică în funcție de diferite combinații între operații cu numere întregi și/sau în VM

**Soluția** pt rezolv acestor impedimente: fol operațiilor normalizate în virgulă mobilă.

*MIPS* și *MFLOPS* pot fi utili la compararea sistemelor de calcul aparținând aceleiași familii, cu același set de instrucțiuni și având același ciclu de ceas.

**Studiați: GFLOPS, TFLOPS, PFLOPS**





## Informația în Evaluarea performanțelor calculatoarelor

### Metrice folosite

1. Timpul de execuție

2. Timpul CPU

3. MIPS

4. MFLOPS

5. Legea lui Amdahl

Programe reale de evaluare a performanțelor

**Legea lui Amdahl** se referă la creșterea performanței care se poate obține prin **îmbunătățirea unei anumite părți/componente** a unui sistem de calcul.

Se presupune că se poate efectua îmbunătățirea unui sistem de calcul prin îmbunătățirea performanței la execuția unui task.

**Legea lui Amdahl** definește creșterea **performanței** ce se poate obține prin utilizarea acestei îmbunătățiri ca fiind:

$$\Delta v = \frac{P_{imb}}{P_{neimb}} \quad (16)$$

unde  $P_{imb}$  este performanța obținută utilizând îmbunătățirea atunci când este posibil, iar  $P_{neimb}$  este performanța obținută fără utilizarea îmbunătățirii.

Îmbunătățirea -  $\rightarrow$  dpdv al creșterii **vitezei de execuție** în funcție de timpul de execuție:

$$\Delta v = \frac{t_{E\_neimb}}{t_{E\_imb}} \quad (17)$$

unde  $t_{E\_neimb}$  reprezintă **timpul de execuție fără utilizarea îmbunătățirii**, iar  $t_{E\_imb}$  este **timpul de execuție obținut utilizând îmbunătățirea când e posibil**.

**Creșterea vitezei** depinde de:

- fracțiunea timpului de execuție a calculatorului **original**

în care se poate folosi îmbunătățirea ( $F_{imb}$ ), unde  $F_{imb} \leq 1$ ,

- îmbunătățirea obținută prin utilizarea noului mod de execuție,

deci creșterea vitezei care s-ar obține dacă s-ar utiliza numai noul mod de execuție ( $\Delta v_{imb}$ ), unde  $\Delta v_{imb}$  = raportul dintre - timpul de execuție în modul original și  
- timpul de execuție în modul îmbunătățit, cu  $\Delta v_{imb} > 1$ .

Timpul de execuție utilizând calculatorul original în modul îmbunătățit va fi  
timpul în care se utilizează partea nemodificată a calculatorului  
+ timpul în care se utilizează partea îmbunătățită:

$$t_{E\_nou} = t_{E\_init} \cdot \left( (1 - F_{imb}) + \frac{F_{imb}}{\Delta v_{imb}} \right) \quad (18)$$

**Creșterea totală a vitezei** va fi raportul dintre timpii de execuție:

$$\Delta v_{tot} = \frac{t_{E\_init}}{t_{E\_nou}} = \frac{1}{(1 - F_{imb}) + \frac{F_{imb}}{\Delta v_{imb}}} \quad (19)$$



#### Exemplul 0Bh:

Se presupune că un calculator ruleaza (in general) in 60% din timp operatii cu procesorul, iar in 40% din timp operatii cu hard discul si ca utilizatorul nu e multumit de performanta sistemului. Dupa studierea pietei de componente, utilizatorul a gasit 2 optiuni: prima, sa inlocuiasca procesorul cu unul de 1,5 ori mai bun pentru suma de 500 lei, iar a doua sa inlocuiasca unitatea de hard disc cu una de 2,5 ori mai buna, dar cheltuind 800 lei.

- Care este raportul pret/performance obtinut in ambele cazuri?
- Care optiune va fi considerata daca se doreste o investitie cat mai mica dpdv financiar dar cu raportul pret/performance cel mai bun?
- Care optiune va fi consid daca se doreste obt performantei cele mai bune, indiferent de suma cheltuita?

#### Soluție:

a) Inlocuind procesorul, este afectat 60% din timp si se obtine:

Performanta dupa imbunatatire :  $\Delta v = \frac{1}{1 - 0,6 + \frac{0,6}{1,5}} = 1,25 \Rightarrow$  o crestere a performantei cu 25%

Inlocuind unitatea de hard disc, este afectat 40% din timp si se obtine:

Performanta dupa imbunatatire :  $\Delta v = \frac{1}{1 - 0,4 + \frac{0,4}{2,5}} = 1,3158 \Rightarrow$  o crestere a performantei cu 31,58%

Raportul pret/performance:  $500/125=4,00$  vs  $800/131,58=6,08$

b) deoarece raportul pret/performance e mai bun in primul caz, se poate alege compromisul de a creste performanta doar cu 25% in loc de 31 % pentru a economisi 300 lei, deci se va alege inlocuirea procesorului

c) daca nu exista problema financiara, atunci se va alege inlocuirea hard discului deoarece performanta obtinuta este mai mare cu aproape 7% , desi raportul pret/performance e mai ridicat cu 50%



### Exemplul 0Ch:

Se presupune că un program rulează în 50 secunde pe un calculator, din care 40 de secunde sunt rulate operații de înmulțire.

Cu cât trebuie crescută viteza operațiilor de înmulțire pentru a rula programul de 5 ori mai rapid?

### Soluție:

$$t_{E \text{ initial}} = 50s$$

Îmbunătățirea vitezei 40s din 50s = 80%  $\Rightarrow F_{imb} = 0,8$

$$t_{E \text{ după îmbunătățire}} = 50s \left( (1 - 0,8) + \frac{0,8}{n} \right) = (100s - 80s) + \frac{40s}{n},$$

*n sau  $\Delta v_{\text{îmbunătățit}}$*

$$\text{de 5 ori mai rapid: } 10s = \frac{40s}{n} + 10s \Rightarrow 0 = \frac{40s}{n}$$

$\Rightarrow$  Nu există o astfel de cantitate cu care să îmbunătățim execuția operației de înmulțire pentru a obține creșterea performanței de 5 ori, dacă operația de înmulțire apare în 80% din instrucțiuni. p

**Exemplul 0Dh:**

Se presupune ca la un CPU se pot îmbunătăți anumite instrucțiuni care apar în 15% din cazuri, executându-se de 2 ori mai rapid. Care este creșterea totală a vitezei obținute?

**Soluție:**

$$F_{imb} = 0,15; \Delta v_{imb} = 2$$

$$\Delta v = \frac{t_{e\_neimb}}{t_{E\_imb}} = \frac{1}{1 - 0,15 + 0,15/2} = \frac{1}{0,925} = 1,08 \Rightarrow \text{viteza a crescut cu } 8\%$$

**Exemplul 0Eh:**

Se presupune ca se poate îmbunătăți performanța unui CPU prin creșterea vitezei de execuție cu un factor de 3, dar cu inconvenientul creșterii costului de 2 ori. Se mai presupune ca CPU e utilizat în doar 60% din timp, în rest având loc alte operații, precum cele cu dispozitivele I/O. Dacă costul CPU reprezintă 1/3 din costul întregului sistem, se pune întrebarea dacă e o investiție rentabilă (dpdv al raportului cost/performanță) creșterea vitezei cu un factor de 3.

**Soluție:**

$$\text{Creșterea vitezei de execuție va fi: } \Delta v = \frac{t_{e\_neimb}}{t_{E\_imb}} = \frac{1}{1 - 0,6 + 0,6/3} = \frac{1}{0,6} = 1,67$$

$$\Rightarrow \text{Costul noului sistem: } \frac{2}{3} \cdot 1 + \frac{1}{3} \cdot 2 = 1,33$$

deci de 1,33 ori mai mare față de costul calculatorului original. Astfel, creșterea costului (1,33) nu este mai mare decât creșterea performanței (1,67) și deci se îmbunătățește raportul cost/performanță.

### Informația în Evaluarea performanțelor calculatoarelor

#### Metrici folosite

1. Timpul de execuție

2. Timpul CPU

3. MIPS

4. MFLOPS

5. Legea lui Amdahl

Programe reale de evaluare a performanțelor

În general, **performanța unui sistem de calcul nu se poate caracteriza printr-o singură metrică**, deoarece aceasta depinde de interacțiunile diferitelor componente ale sale.

Pe măsură ce arhitectura calculatoarelor evoluează, devine tot mai dificil de comparat performanța diferitelor sisteme doar prin specificațiile date de producător => compararea performanțelor prin rularea unor **programe de testare a performanțelor (benchmarks) standardizate**

=> s-au dezvoltat diferite programe specializate (numite **benchmark-uri**) pt evaluarea performanțelor unor componente sau ale unor sisteme de calcul:

- UCP (Unitatea centrală de procesare)
- unitatea aritmetică în virgulă fixă / mobilă
- sistemul de memorie
- sistemul de I/O
- sistemul de operare

Metricile tradiționale precum MIPS sau MFLOPS uneori pot fi eronate și nu reflectă factori relevanți precum **timpul de execuție al aplicațiilor, utilizarea memoriei, energia consumată**.

Aplicațiile de tip benchmark suferă și ele unele limitări care împiedică compararea corectă a mai multor sisteme. Există totuși o **metodologie pe bază de nuclee (numite kernele)** și diferite aplicații ce furnizează **estimări foarte pertinente asupra performanței procesorului**, deoarece sunt **standardizate**

=> se folosesc **aplicații complete** sau chiar **seturi de aplicații**

ce permit inclusiv măsurarea timpului de utilizare a memoriei, energia consumată de procesor, ....



Categorii pt programele de test [Patterson09]:

1. **Programe reale**: ex :**compilatoare de C**, **programe de procesare text**, **programe de proiectare asistată (Spice)**.

Testarea: în condiții similare privind intrările, ieșirile și setarea opțiunilor.

Dezavantaj: uneori se pot întâlni probleme de portabilitate, mai ales datorită dependenței de S.O. folosit

2. **Aplicații modificate**: se fol aplicațiile reale ca blocuri componente ale unor programe de testare.

Modificările constau în eliminarea unor componente (de ex eliminarea componentelor de I/O dacă programul de test este destinat performanțelor UCP) sau prin introducerea unor componente care măresc gradul de portabilitate al programelor.

Se fol **Script-uri** pt a simula programe de aplicație și cu scopul de a reproduce o *comportare interactivă* (de ex prin **afișări pe ecranul calculatorului**) sau pt a simula *interacțiunea multi-utilizator* produsă pe un sistem de tip server.

3. **Nuclee (kernels)** din programe reale: s-au extras porțiuni semnificative (=nuclee) din programe reale, pt a fi folosite ca rutine de test. Exemple: **Livermore Loops**, **Linpack**.

4. **Toy benchmarks** (programe de test amuzante – jucărie- nu sunt ft concludente): sunt **programe scurte (maxim 100 de linii de cod)** care produc un *rezultat cunoscut înainte de rulare*, folosite doar pentru a testa *viteza de execuție*.

Exemplu: **Sieve of Erastosthenes**, **Puzzle**, **Quicksort**.

5. **Programe de test sintetice (synthetic benchmarks)**: programe create artificial, dar asemănătoare ca scop cu programele de test de tip “kernels” -> testarea: prin operații și operanzi specifici, cu frecvența medie de apariție în programele reale. Exemple: **Drystone** și **Whetstone**. Programul de test Dhrystone -> analiză statistică , fiind fol pt testarea performanțelor la aritmetica cu numere întregi și pt modul de transmitere a parametrilor la apelul unor funcții; testul Whetstone = o colecție de coduri ce testează performanța la rularea unor biblioteci matematice în virgulă mobilă; Programele ca Drystone și Whetstone - *testează amănunțit doar UCP*, dar nu testează performanța celorlalte componente ale sistemului (precum unitățile de disc, dispozitivul de afișare, etc).

#### Standarde in industrie

##### ***Business Applications Performance Corporation*** (BAPCo)

-> SYSmark 2012 (performanta generala a intregului sistem), TabletMark 2013 (pt dispozitive tactile), MobileMark 2012 (bateria si performanta sistemului)

- de studiat : [http://www.ginfo.ro/revista/11\\_3/tehno.pdf](http://www.ginfo.ro/revista/11_3/tehno.pdf)

##### ***Embedded Microprocessor Benchmark Consortium*** (EEMBC) (“embassy”)

- in domeniul sistemelor cu calculator integrat (Embedded Systems)

-> exista 5 tipuri de seturi de programe de test în funcție de domeniile de aplicație:

auto și industrial, bunuri de consum, interconectare, automatizări de birou, telecomunicații

##### ***Standard Performance Evaluation Corporation*** (SPEC), cu SPECint si SPECfp

- in 1988 - corporația SPEC (non-profit) – scop: *stabilirea, menținerea și garantarea* de seturi standardizate pt programe de test

– initial cuprindea firmele HP, DEC, MIPS și Sun; a crescut în timp, incluzând în prezent peste 60 de companii

- SPEC nu rulează programe de test (nu realizează testare de mașini de calcul), ci dezvoltă programe de test, analizează și publică rezultatele obținute și transmise către SPEC de membrii organizației și alte instituții licențiate

##### ***Transaction Processing Performance Council*** (TPC)

- măsoară capacitățile unui sistem în realizarea tranzacțiilor (accesari și actualizări de baze de date)

- Primul program de test: 1985 TPC-A;

intre timp: multe alte variante, de ex. TPC-W pt testarea performanțelor tranzacțiilor pe bază de Web.

***Coremark*** – pentru sisteme embedded

- *Linpack* = o colectie de rutine bazate pe algebra liniara si metode Gaussiene
  - de 2 ori pe an se publica pe *www.top500.org* o lista cu primele 500 computere care au cea mai ridicata performanta pe baza acestui benchmark [Patterson 2013]
  - > primul sistem din lista top500 e considerat “**cel mai rapid sistem din lume**”
- *SPECrate* = o metrica bazata pe debit – bazata pe benchmarkuri din suita SPEC CPU (2006 de exemplu)
  - masoara paralelismul la nivel de task (desi ruleaza mai multe copii ale unui program simultan, nu exista comunicare intre taskuri)
- *SPLASH* si *SPLASH 2* (Stanford Parallel Applications for Shared Memory) – Universitatea Stanford
  - similar SPEC CPU, dar fol. 2 seturi de date si aplica benchmarkuri paralele (kernele si aplicatii)

**SPEC (System Performance Evaluation Cooperative)** – o organizatie sprijinita de industria producatorilor de sisteme pentru a asigura **seturi standardizate de benchmarkuri**

In 1989: SPEC a creat un benchmark pt evaluarea performantei CPU – “SPEC89” - a evoluat in 5 generatii

...

In 2006: SPEC a creat “SPEC CPU2006” – un set de **12 benchmarkuri** pentru operatii intregi (CINT2006) si **17 benchmarkuri** pentru operatii in virgula mobila (CFP2006).

[Patterson2013]

# 11. Evaluarea performantelor calculatoarelor

## Programe reale de evaluare a performantelor (5)

ANCA APATEAN - AC - UTCN

SPEC2006 benchmark description	SPEC2006	SPEC2000	SPEC95	SPEC92	SPEC89
GNU C compiler					gcc
Interpreted string processing			perl		espresso
Combinatorial optimization		mcf			li
Block-sorting compression		bzip2		compress	eqntott
Go game (AI)	go	vortex	go	sc	
Video compression	h264avc	gzip	jpeg		
Games/path finding	astar	eon	m88ksim		
Search gene sequence	hmmer	twolf			
Quantum computer simulation	libquantum	vortex			
Discrete event simulation library	omnetpp	vpr			
Chess game (AI)	sjeng	crafty			
XML parsing	xalancbmk	parser			
CFD/blast waves	bwaves				fpppp
Numerical relativity	cactusADM				tomcatv
Finite element code	calculix				doduc
Differential equation solver framework	dealll				nasa7
Quantum chemistry	gamess				spice
EM solver (freq/time domain)	GemsFDTD			swim	matrix300
Scalable molecular dynamics (~NAMD)	gromacs		apsi	hydro2d	
Lattice Boltzman method (fluid/air flow)	lbm		mgrid	su2cor	
Large eddie simulation/turbulent CFD	LESlie3d	wupwise	applu	wave5	
Lattice quantum chromodynamics	milc	apply	turb3d		
Molecular dynamics	namd	galgel			
Image ray tracing	povray	mesa			
Spare linear algebra	soplex	art			
Speech recognition	sphinx3	equake			
Quantum chemistry/object oriented	tonto	facerec			
Weather research and forecasting	wrf	ampp			
Magneto hydrodynamics (astrophysics)	zeusmp	lucas			
		fma3d			
		sixtrack			

- programele de test SPEC = *seturi* de aplicații pentru testarea procesoarelor  
 Avantaj seturi: eventuala slăbiciune a uneia dintre aplicații poate fi compensată rulând celelalte aplicații din setul etalon

- există 5 generații de seturi de programe de test, pt operare în **virgulă fixă** (INT - întregi) și **virgulă mobilă** (reale, FP - Floating Point):  
 SPEC89, SPEC92, SPEC95, SPEC2000, SPEC2006 (vers curenta **V1.2** - Sept 2011)

- SPEC CPU2006 conține 2 serii de programe de test: CINT2006 , CFP2006 pt măsurarea și compararea performanțelor la calcule cu întregi (12 seturi), respectiv în virgulă mobilă (17 seturi).  
 Cele 2 -> măsoară performanța pt procesor, pt arhitectura memoriei și pt compilator.

[Hennessy2009]

# 11. Evaluarea performantelor calculatoarelor

## Programe reale de evaluare a performantelor (6)

ANCA APATEAN - AC - UTCN

SPEC a decis sa raporteze performanta SC la **un singur numar** – pt a sintetiza toate cele 12 benchmarkuri de tip intreg => s-a divizat timpul de executie al unui CPU de referinta la timpul de executie obtinut de sistemul testat = “normalizare”

=> **valoarea SPECratio - direct proportionala cu performanta sistemului (avantaj major)**

Valoarea rezultata din masuratorile pe CINT2006 sau CFP2006 se obtine din media geometrica a valorilor SPECratio

### Exemplu:

Benchmarkurile SPECINTC2006 rulate pe un procesor **Intel Core i7 920 la 2.66 GHz** [Patterson2013]

- timpul de executie (rel. 8) = produsul a 3 factori:

Nr de instructiuni (masurat in miliarde) = I coloana

Nr de cicluri pe instructiune adica CPI (se observa ca CPI variaza cu un factor >5) = a II-a coloana,

Durata ciclului de ceas masurata in nanosecunde (e aceeasi pt toate liniile din tabel  $T_c = 0.376 \text{ (sec} \times 10^{-9})$ ).

Ultima coloana: **SPECratio** = timpul de referinta (furnizat de SPEC – coloana IV) si divizat la timpul calculat cu relatia (8)

=> **SPECINTC2006** va fi **media geometrica** a valorilor de pe ultima coloana (SPECratio) = **25.7** [Patterson2013]

### Concluzie:

La compararea a 2 SC prin SPECratio,

se va folosi **media geometrica**

=> va furniza acelasi raspuns relativ

indiferent de calculatorul utilizat ac

referinta (adica pentru normalizare)

- daca s-ar folosi media aritmetica rezultatele ar

varia functie de SC de referinta

Nr instruct. x10 <sup>9</sup>	CPI	Timp executie (sec)	Timp de referinta (sec)	SPECratio
794	1.20	358	8050	22.5
2252	0.60	508	9770	19.2
221	2.66	221	9120	41.2
2390	0.70	629	9650	15.4
1274	1.10	527	10490	19.9
3793	0.50	713	22130	31.0
1250	1.00	470	7020	14.9
2616	0.60	590	9330	15.8
659	0.44	109	20720	190.0
367	2.10	290	6250	21.5
1948	0.80	586	12100	20.7
1045	0.70	275	6900	25.1

**Media geometrica**

**25.7**

### SPEC2006 benchmark description

GNU C compiler	gcc
Interpreted string processing	perl
Combinatorial optimization	mcf
Block-sorting compression	bzip2
Go game (AI)	go
Video compression	h264avc
Games/path finding	astar
Search gene sequence	hmmer
Quantum computer simulation	libquantum
Discrete event simulation library	omnetpp
Chess game (AI)	sjeng
XML parsing	xalanbmk

# “Fallacies and Pitfalls” [Patterson2013]

ANCA APATEAN - AC - UTCN

*Pitfalls* = “easily made mistakes”

= “generalizations of principles that are true only in a limited context”

“*Pitfall*: Expecting the improvement of one aspect of a computer to increase overall performance by *an amount proportional* to the size of that improvement.”

-> *Legea lui Amdahl*

“*Pitfall*: Using a *subset* of the performance equation as a performance metric.”

“commonly held misconceptions = *fallacies*” (*aberatii*)

“*Fallacy*: Computers at low utilization use little power” (*consum proportional*)

“*Fallacy*: Designing for performance and designing for energy efficiency are unrelated goals (*independent*)”

#### Open source benchmarks [Wikipedia]

DEISA Benchmark Suite: pt aplicatii stiintifice  
Dhrystone, Fhourstones: aritmetica intrega  
Whetstone: aritmetica in virgula mobila  
HINT: performanta procesorului si a memoriei  
Iometer: sistemul I/O  
Linpack: masoara FLOPS  
NBench: performanta pt numere intregi, in virgula mobila si memorie (sintetic)  
Geek Benchmark: performanta pt numere intregi, in virgula mobila si memorie  
POV-Ray: redare 3D  
TATP Benchmark: pt tranzactii in telecomunicatii  
TPoX: tranzactii pe baze date  
Rodinia: arhitecturi paralele (acceleratoare)  
Parsec: sisteme cu arhitecturi de memorie paralele  
Splash2: arhitecturi paralele  
STREAM: banda memoriei  
LLCbench: (Low Level Architectural Characterization Benchmark Suite) perform procesorului si a memoriei

#### Altele:

**BAPCo**: MobileMark, SYSmark, WebMark  
**Windows** System Assessment Tool  
Futuremark: 3DMark, PCMark  
**iCOMP** – fol de Intel  
**Performance Rating** – fol de AMD si Cyrix  
Sunspider – viteza browserului  
  
Worldbench  
PiFast  
SuperPrime  
Super PI, Hyper pi  
wPrime, IntelBurnTest, Prime95, Montecarlo superPI,  
OCCT , y-cruncher

Daca un SC e capabil sa calculeze pi (pana la pozitia 32 milioane dupa punctul zecimal) fara nici o greseala, se considera a fi stabil dpdv al memoriei si procesorului; testul poate dura ore in loc de minute pe unele sisteme

- super pi este calcul pt un singur thread, deci nu e relevant la evaluarea performantelor sistemelor multiprocesor

=> hyper pi – pt threaduri multiple

**Benchmarks = set de programe reprezentative pt evaluarea sistemelor de calcul**

- aceste programe efectueaza anumite operatii si masoara timpul în care acestea au fost efectuate:

Rezultatele sunt comparate cu cele obținute de un sistem de referinta,

iar sistemul testat primeste un punctaj in functie de rezultat (=scor)

- pt a det punctajul corespunzator unei aplicatii, se compara timpul de care a avut nevoie sistemul pt a îndeplini sarcina cu timpul în care a îndeplinit sarcina un sistem considerat *sistem etalon*

- Punctajul corespunzator unei categorii este determinat **calculând media geometrica** a punctajelor obtinute pentru fiecare aplicatie din categoria respectiva.

=> Punctajul general – det. tot pe baza unei medii geometrice, dar a punctajelor coresp. fiecarei categorii

O masura a performantei:

timpul de executie al unui set reprezentativ de programe

- timpul de executie poate fi: **timpul total de executie**

**media aritmetica** sau **geometrica a timpilor de executie**

**Sintetizarea si compararea performantei unui grup (set) de benchmarkuri**

- daca nu exista o relatie unitara intre masuratorile sau rezultatele obtinute, nu se poate defini performanta relativa intre 2 sau mai multe SC pe baza timpului de executie (pt fiecare program in parte):

- exemplu: un program ruleaza pe calculatorul A mai rapid decat pe B, dar un alt program ruleaza pe calculatorul B mai rapid decat pe A => setul resp nu este relevant



# 11. Evaluarea performantelor calculatoarelor

## Programe reale de evaluare a performantelor (9)

ANCA APATEAN - AC - UTCN

- **timpul TOTAL de executie**  - cea mai simpla metoda
  - se spune ca un calculator A este de n ori , unde  $n = tE(B)/tE(A)$  ,  
mai rapid decat calculatorul B pt un set de programe
- **media aritmetica a timpilor de executie** :  $MA = \frac{1}{n} \sum_{i=1}^n t_{Ei}$ , unde  $t_{Ei}$  este timpul de executie al programului i din setul total cu n programe de test
- **media aritmetica ponderata a timpilor de executie** 
  - se fol daca exista anumite programe care se executa mai des din set
  - fiecare program are atribuita o pondere ce indica frecventa sa de executie
  - alegerea ponderilor se face a.i. timpii de executie ponderati ai fiecarui program sa fie egali (pe un calculator de referinta)

	tE(A)	tE(B)	tE(A normalizat la A)	tE(B normalizat la A)	tE(A normalizat la B)	tE(B normalizat la B)
Progr 1	1	10	1	10	0,1	1
Progr 2	1000	100	1	0,1	10	1
Media aritm	500,5	55	1	5,05	5,05	1
Media geom	31,6	31,6	1	1	1	1

Se pare ca A este mai rapid ca B

Se pare ca B este mai rapid ca A

**La normalizare:** se consid media timpilor de executie normalizati

- daca se fol media aritmetica a timpilor de executie, rezultatul va depinde de alegerea calculatorului de referinta !

- **media geometrica** are avantajul ca este independenta de seria datelor folosite la normalizare:  $MG = \sqrt[n]{\prod_{i=1}^n t_{Ei}}$

=> rezultatul va fi acelasi, corect, indiferent de calculatorul ales la normalizare

- dezavantaj: nu anticipeaza performantele

**Programe de evaluare a performantelor:**

- se prefera folosirea unui set de aplicatii reale ca programe de evaluare

- programe artificiale (sintetice) de evaluare (kernele):

se creaza un program in care frecventele de executie ale instructiunilor sunt aceleasi cu cele dintr-un set de programe de evaluare

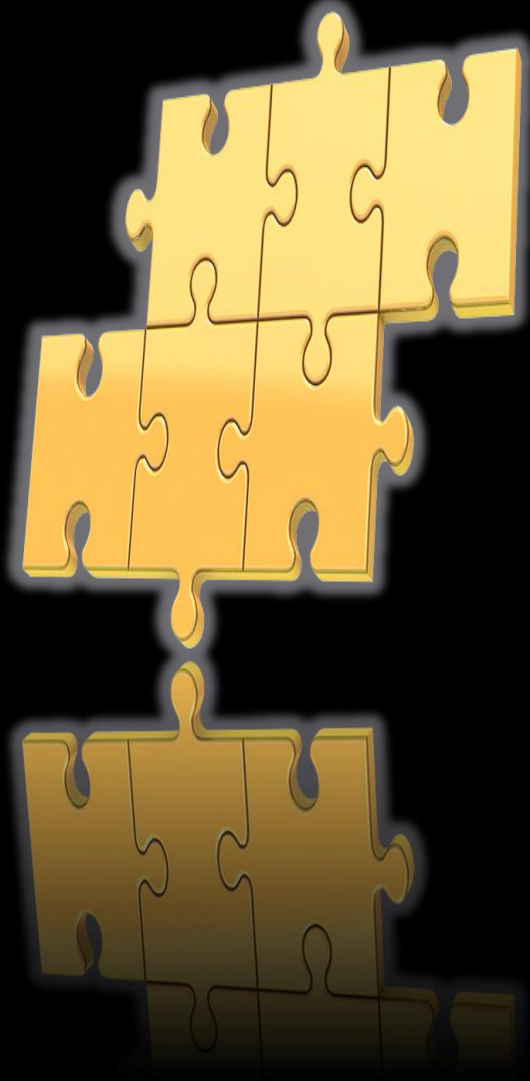
Diferiți **utilizatori** pot fi interesați de diferite **alte aspecte**, dintre care :

- consumul de energie – să fie unul cât mai redus;
- dimensiune redusă și greutate mică – mai ales la sistemele portabile;
- costul sistemului – uneori se raportează viteza sistemului la cost;
- performanța per watt – în special la sistemele paralele.

Alți factori de calitate care se urmăresc la evaluarea performanțelor unui SC sunt:

- **generalitatea** – gama de aplicații ce se pot rula pe o arhitectură;
- **simplitatea în utilizare** – ușurința dezvoltării de programe pentru arhitecturile respective;
- **expandabilitatea** – ușurința cu care se poate extinde o arhitectură (cu procesoare, memorii, dispozitive de I/E);
- **compatibilitatea** – în ce măsură pot fi executate (pe noile arhitecturi) programele dezvoltate pentru arhitecturile precedente din aceeași familie;
- **fiabilitatea** – probabilitatea de apariție a defectelor sau timpul mediu între defecte.

Pentru **producători**, este mai importantă **energia consumată** (ce determină durata de viață a bateriei) decât **puterea consumată**;  
se ține cont de managementul puterii.



# Concluzii

- Evaluarea performanțelor calculatoarelor nu trebuie să aibă legătură cu informația vagă, incompletă sau apelul la popularitate.
- Metricile uzuale folosite la evaluarea performanțelor PC-urilor sunt:  
**Timpul de execuție , Timpul CPU, MIPS, MFLOPS, Legea lui Amdahl**
- La folosirea unor programe reale de evaluare a performanțelor PC-urilor trebuie considerate următoarele aspecte:
  - se preferă folosirea unui set de aplicații reale ca programe de evaluare
  - programe artificiale (sintetice) de evaluare (kernele):  
se creează un program în care frecvențele de execuție ale instrucțiunilor sunt aceleași cu cele dintr-un set de programe de evaluare



- [Barr2005] – Mostafa Abd-El-Barr, Hesham El-Rewini  
– “Fundamentals of Computer Organization and Architecture”, 2005
- [Baruch2000] - Zoltan Baruch  
– “Arhitectura calculatoarelor”, Editura Toderescu, 2000
- [Brey1997] - Barry B. Brey  
– “The Intel Microprocessors”, 4<sup>th</sup> edition, 1997
- [Hennessy2009] - John Hennessy, David Patterson  
– “Computer Architecture – A quantitative Approach”, 2009, 5<sup>th</sup> edition
- [Hide2001] - Randall Hide  
– “The Art of Assembly Language”, beta edition
- [Lupu2012] – Eugen Lupu, Simina Emerich , Anca Apatean  
– “Initiere in Limbaj de Asamblare x86. Lucrari practice, teste si probleme”, Ed. Galaxia Gutenberg, 2012
- [Mueller2012] - Scott Mueller  
– “Upgrading and Repairing PCs”, 20<sup>th</sup> edition, 2012
- [Null2003] - Linda Null, Julia Lobur  
– “The essentials of Computer Organization and Architecture”, 2003
- [Patterson2009] – David Patterson, John Hennessy  
– “Computer Organization and Design – the hardware/software interface”, 4<sup>th</sup> edition, 2009
- [Patterson2013] – David Patterson, John Hennessy  
– “Computer Organization and Design – the hardware/software interface”, 5<sup>th</sup> edition, 2013
- [Tarnoff2007] - David Tarnoff  
– “Computer Organization and Design Fundamentals”, editia intai revizuita, 2007