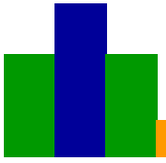


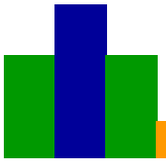
Lecţia 2:

Evaluarea performantei calculatoarelor

G Stefănescu — Universitatea Bucureşti

Arhitectura sistemelor de calcul, Sem.1
Octombrie 2016—Februarie 2017

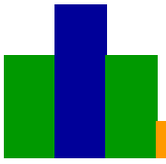
După: D. Patterson and J. Hennessy, Computer Organisation and Design



Performanta calculatoarelor

Cuprins:

- *Executia programelor*
- Progresul hardware-ului
- Masurarea performantei
- Programe de test (benchmark-uri)
- Concluzii

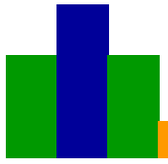


Executia programelor

1. Sursa: Program sursă

```
swap(int v[], int k)
{int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

- în vectorul `v` se permută poziţiile `k` şi `k+1`

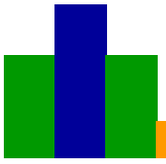


..Executia programelor

2. Compilare: Program în limbajul de ansamblare (tip MIPS)

```
swap:
    muli $2, $5,4
    add  $2, $4,$2
    lw   $15, 0($2)
    lw   $16, 4($2)
    sw   $16, 0($2)
    sw   $15, 4($2)
    jr   $31
```

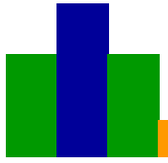
- `multi` - înmulţire imediată; `add` - adunare; `lw/sw` - load/store word
- `$4` - adresa lui `v`; `$5` - adresa lui `k`; `$31` - adresa de continuare; `$2,$15,$16` - regiştri de lucru



..Executia programelor

3. Asamblor: Program în limbaj maşină binar (tip MIPS)

```
11111100101000100000000000000100     multi:  63,5,2,4
00000000100000100001000000100001   add:  0,4,2,2,0,33
10001100010011110000000000000000       lw:  35,2,15,0
10001100010100000000000000000100       lw:  35,2,16,4
10101100010100000000000000000000       sw:  43,2,16,0
10101100010011110000000000000100       sw:  43,2,15,4
00000011111100000000000000001000    jr:  0,31,0,0,0,8
```



..Executia programelor

Asamblor (cont.)

- Codul de sus e fictiv - nu e cod MIPS real, e.g.,

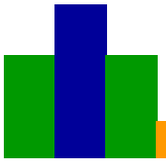
  ```
  multi $2, $5,4
  ```

  trebuie simulat cu alte instrucţiuni, spre exemplu

  ```
  add $2, $5,$5
  add $2, $2,$2
  ```

- codul 0/1 depinde de regiştri alocaţi şi de codul instrucţiunilor; există convenţii standard pentru regiştri şi operaţii (e.g., `$29 = sp` - stack pointer; `$31 = ra` - return address; `op = 35` - `lw`; `op = 43` - `sw`, etc.)

- instrucţiunea a 2-a şi ultima sunt de tip R, restul sunt de tip J; mai există instrucţiuni de tip I



..Executia programelor

Asamblor (cont.)

Există 3 tipuri de formate MIPS (pe 32 biti)

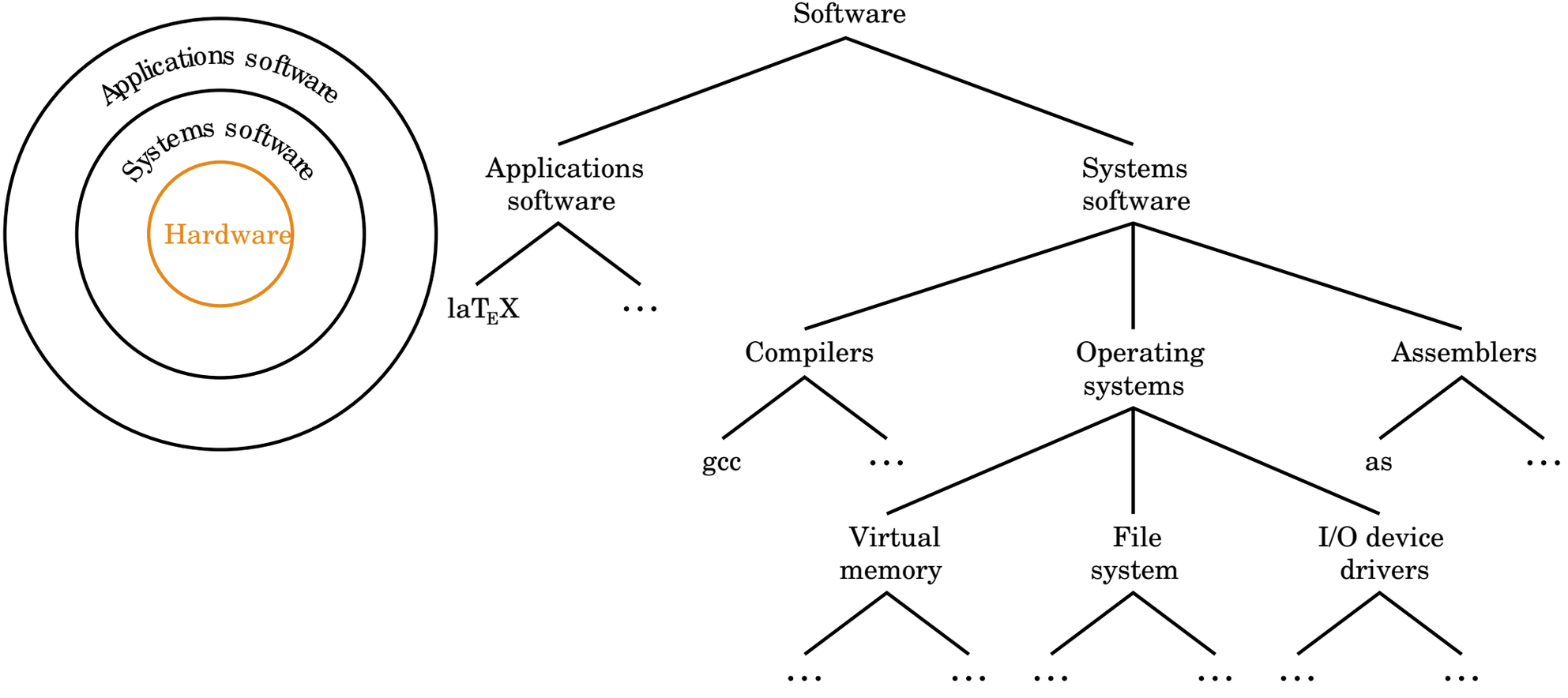
Nume	Campuri						Comentarii
Camp	6 biti	5 biti	5 biti	5 biti	5 biti	6 biti	toate instructiunile
R-format	op	rs	rt	rd	shamt	funct	instructiuni aritmetice/logice
I-format	op	rs	rt	adresa/val.imediata			transfer, branch, inst. imediată
J-format	op	adresa destinatie					instructiuni salt

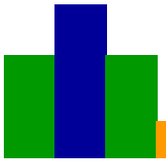
unde

- R-format: op = cod operaţie de bază; rs = registrul sursă 1; rt = registrul sursă 2; rd = registrul destinaţie; shamt = “shift amount”; funct = cod funcţie
- I-format: conţine un “shift” de adresă, ori o valuare imediată;
- J-format: adresă de salt

..Executia programelor

Nivele Hardware/Software: O vedere simplificată (stânga) şi câteva exemple (dreapta)

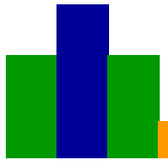




Performanta calculatoarelor

Cuprins:

- Executia programelor
- *Progresul hardware-ului*
- Masurarea performantei
- Programe de test (benchmark-uri)
- Concluzii



Componentele calculatoarelor

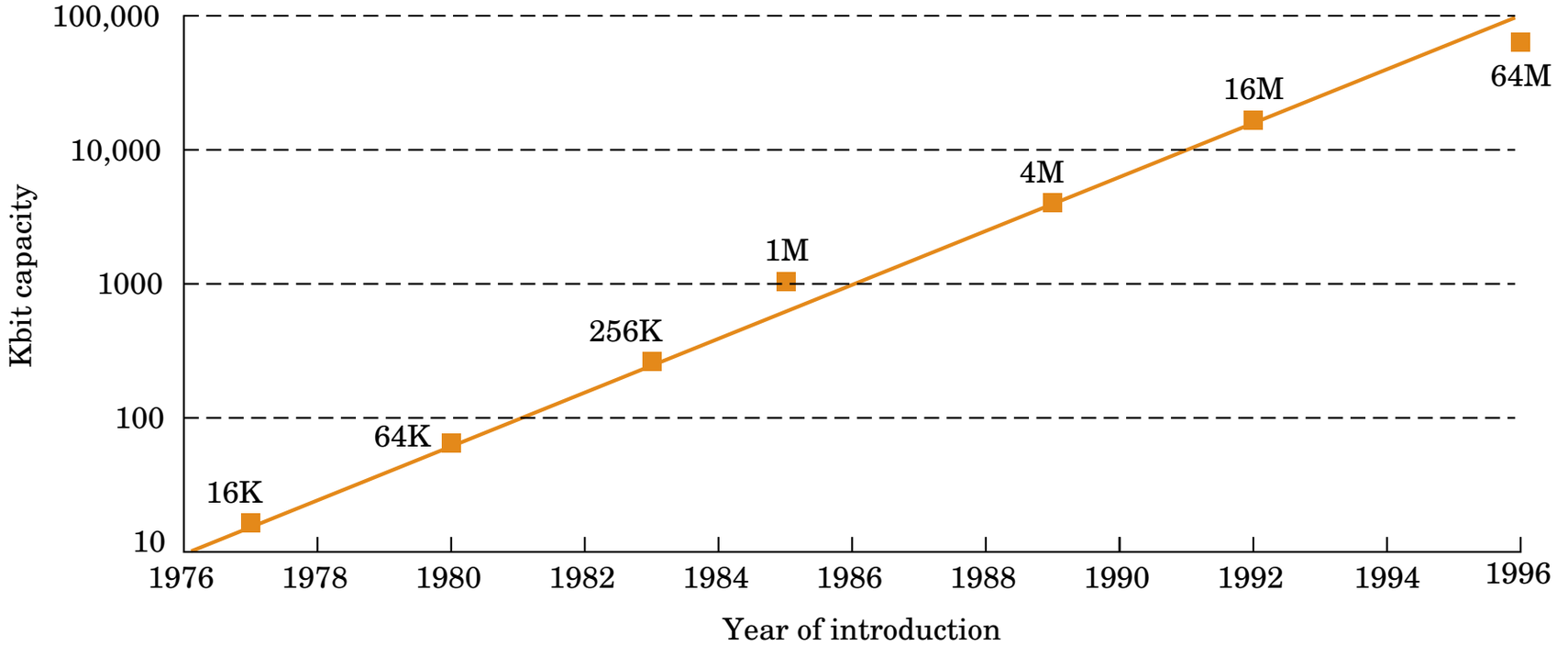
Cele 5 componente clasice ale computer-ului:

- *calea de date [datapath]*
- *control*
- *memorii*
- *intrări*
- *ieşiri*

Combinaţia “datapath + control” se numeşte *procesor*. Toate componentele din calculatoarele din trecut şi de azi intră într-una din aceste clase.

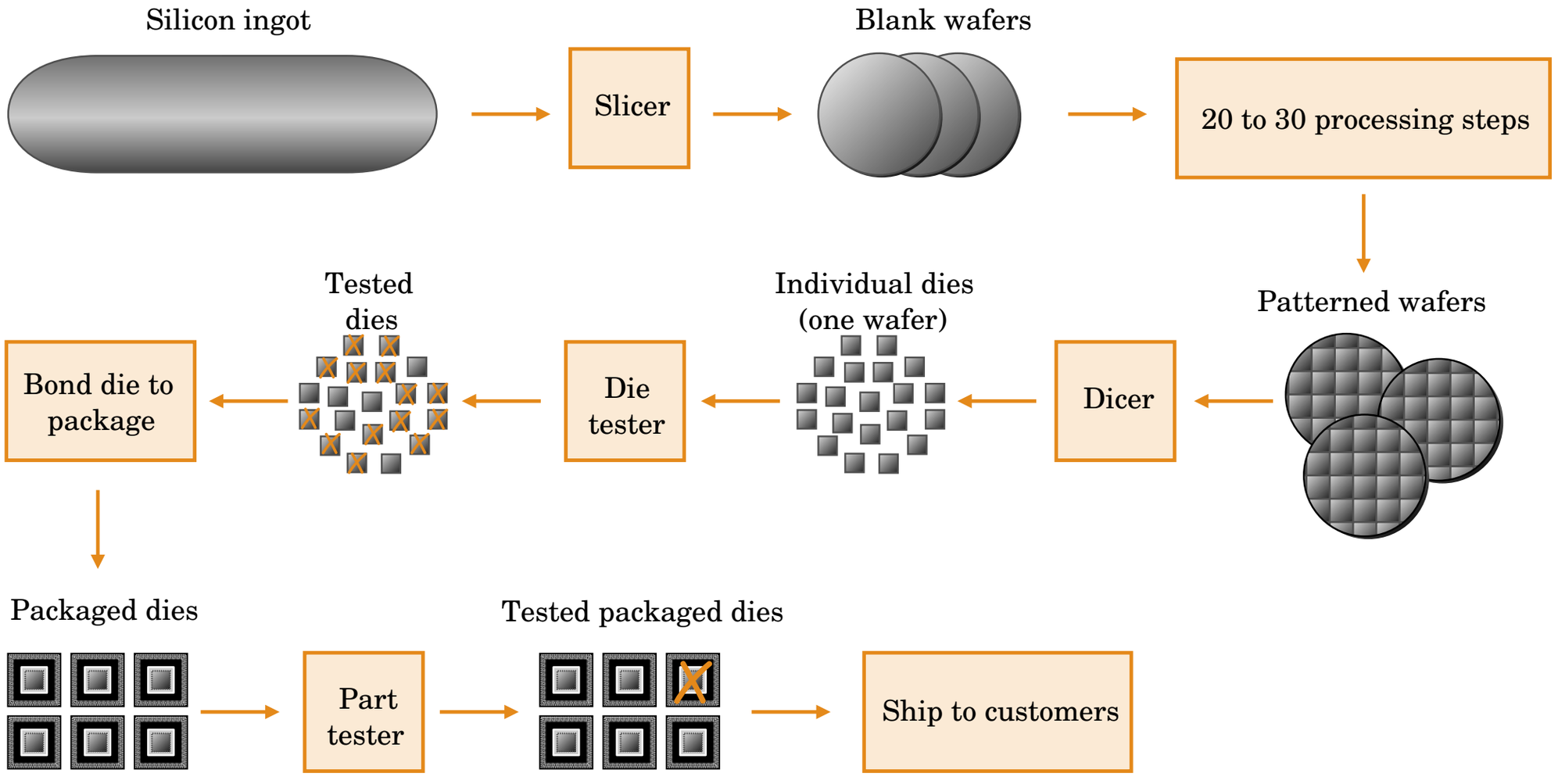
Progresul circuitelor integrate

Memorii: In medie, capacitatea memoriilor a crescut de circa 4 ori la fiecare 3 ani



..Progresul circuitelor integrate

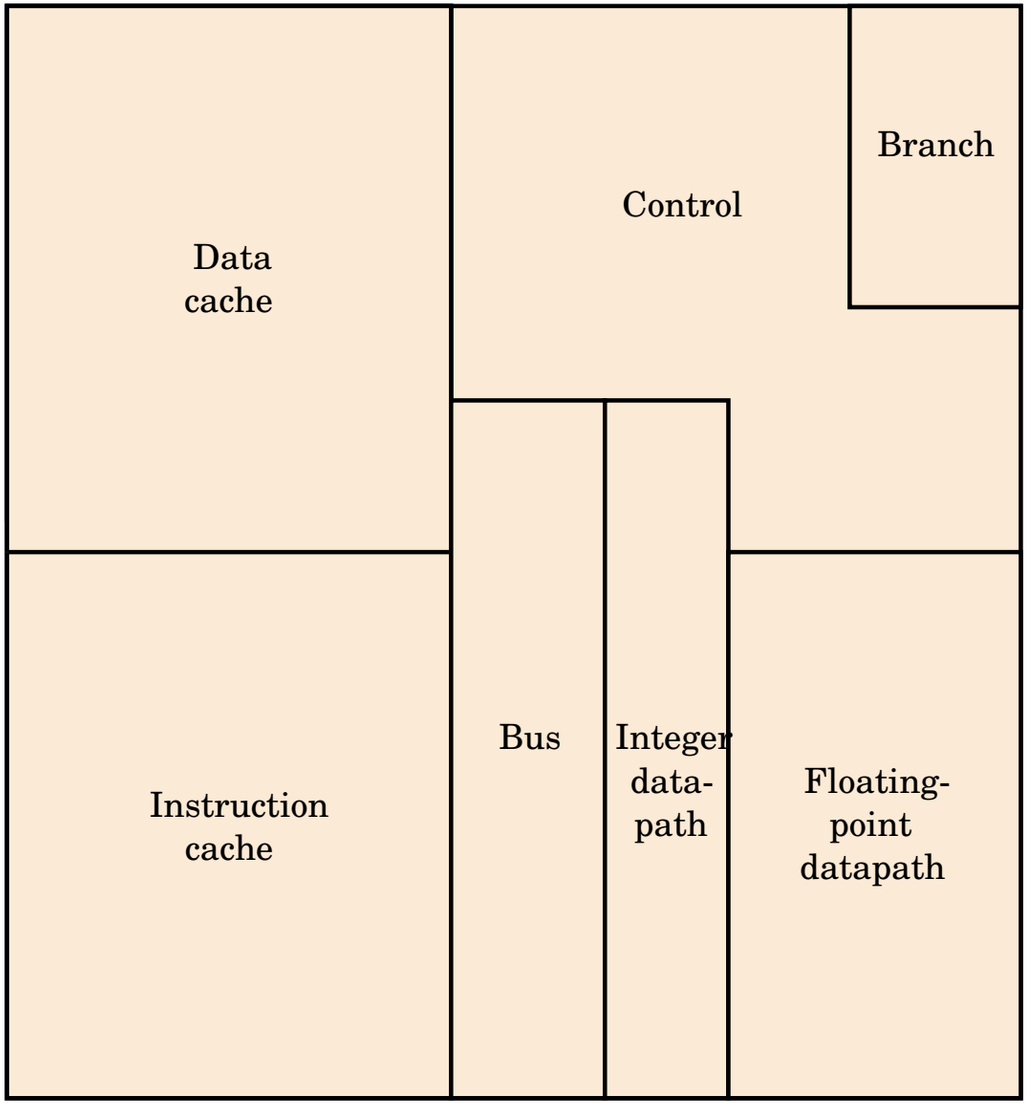
Chip-uri: Procedura de fabricaţie a chip-urilor; numai circa 25% din cele produse corespund standardelor



..Progresul circuitelor integrate

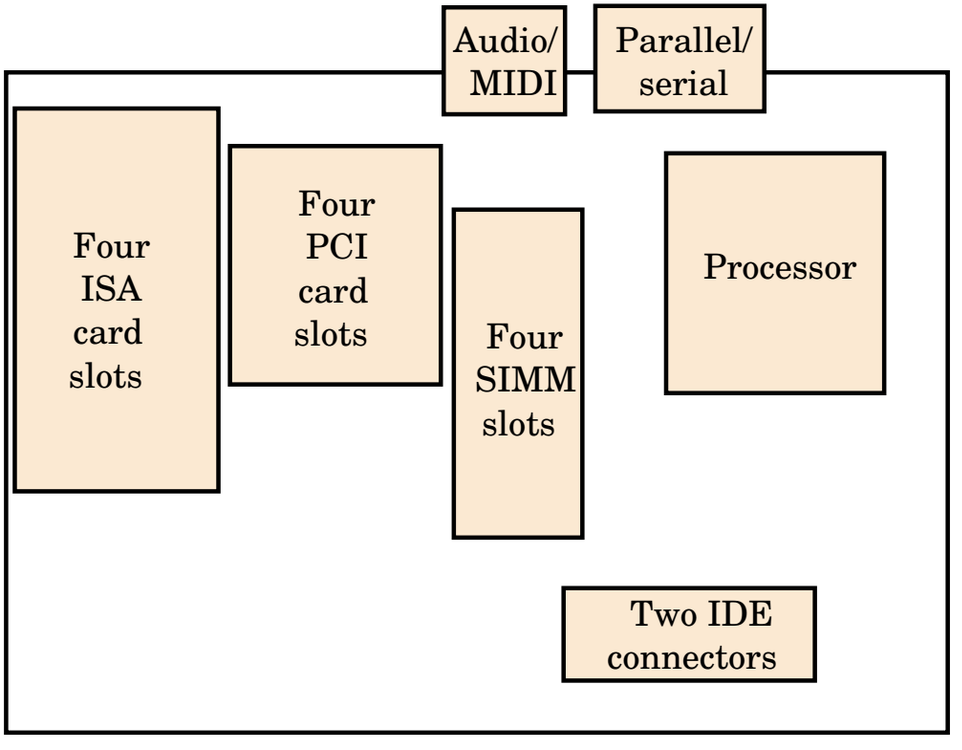
Procesorul Intel Pentium:

- Pe 91 mm^2 sunt 3.3mil tranzistori
- Sunt indicate componentele şi plasarea lor



..Progresul circuitelor integrate

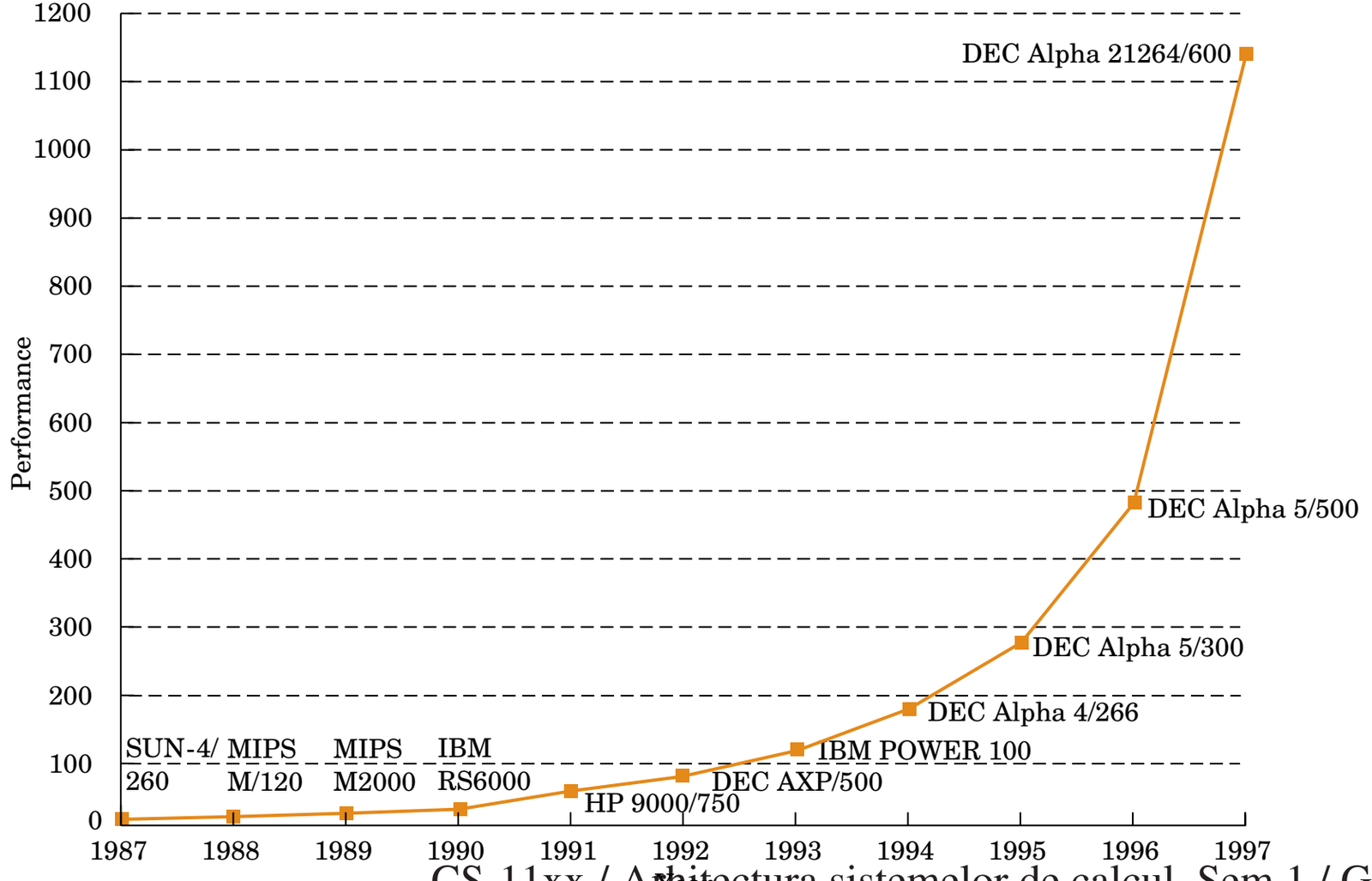
Placă de bază Intel Pentium Pro:

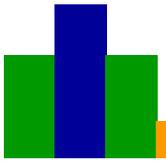


Glosar: IDE (Integrated Drive Electronics), ISA (Industry Standard Architecture - slot inlocuit de PCI), MIDI (Music Instrument Digital Interface), PCI (Peripherical Components Interconnect), SIMM (Single In-line Memory Module); IDE Devices (hard drive, CD-ROM)

..Progresul circuitelor integrate

Performanta statiilor de lucru (1987-1997): creştere cu circa 50% pe an (măsurate relativ la SPEC'92, programele de tip întreg şi comparate cu VAX-11/780)



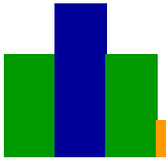


..Progresul circuitelor integrate

Tendinte tehnologice:

- Circuite integrate - densitatea tranzistorilor creşte cu 35% pe an, mărindu-se de 4 ori în 3 ani; mărimea capsulei creşte cu 10%-20%; cumulate, dau o creştere a numărului de tranzistori de circa 55% pe an;
- DRAM - creşte densitatea cu circa 40%-60% pe an; timpul de acces se îmbunătăţeşte mai lent, scăzând cu 1/3 la 10 ani;
- tehnologia de disk magnetic - în ultima vreme, mai mult de 100% pe an;
- tehnologia de reţea - 10 ani pentru trecerea de la 10Mb Ethernet la 100Mb, dar numai 5 ani de la 100Mb la 1Gb.

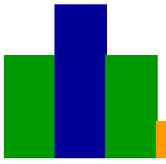
Glosar: DRAM (Dynamic Random-Access Memory)



Pret

Estimarea costul unui PC (1000 $, anul 2001):

Sistem	Subsistem	Fractiune de cost
Carcasa	Cutie metal	2%
	Alimentator, ventilator	2%
	Cabluri, etc.	1%
	Ambalaj, transport	1%
	Subtotal	**6%**
Placa cu procesor	Procesor	22%
	DRAM (128)	5%
	Card video	5%
	Placă de baza cu I/O si retea	5%
	Subtotal	**37%**
I/O devices	Tastatură, mouse	3%
	Monitor	19%
	Hard disk (20GB)	9%
	DVD drive	6%
	Subtotal	**37%**
Software	**OS + MS Office (de baza)**	**20%**



..Pret

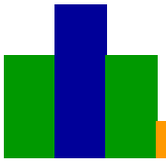
Formarea pretului PC-urilor:

- costul componentelor $P1$
- $20\% \cdot P1 =$ cost direct (de fabricaţie); $P1 + 20\% \cdot P1 = P2$
- $33\% \cdot P2 =$ "gross margin" (R&D, marketing, sales, cheltuieli fixe [cladire, etc], finanţare, taxe); $P2 + 33\% \cdot P2 = P3$ [preţ mediu de vânzare]
- $33\% \cdot P3 =$ "discount" (reducere pentru promoţii, cumparari directe, etc); $P3 + 33\% \cdot P3 = P4$ [list price]

In concluzie, în preţul de vânzare $P4$ [list price] intră

47% componente +10 % costuri directe + 19 % gross margin + 25% discount

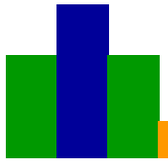
Uzual, cel mult 4%-12% din venitul unei companii merge pe R&D.



Performanta calculatoarelor

Cuprins:

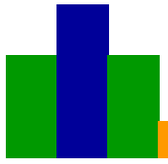
- Executia programelor
- Progresul hardware-ului
- *Masurarea performantei*
- Programe de test (benchmark-uri)
- Concluzii



Masurarea performantei

Masurarea performantei:

- Un utilizator este interesat în *timpul de răspuns* (ori *timpul de execuţie*), adică diferenţa dintre *începutul* şi *sfârşitul* unui eveniment
- Un manager de baze de date este interesat în *gradul de utilizarea* a serverului (throughput), adică cantitatea de servicii prestate într-o unitate de timp.



..Masurarea performantei

Masurarea performantei (cont.)

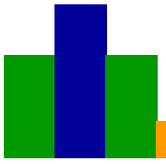
- In Unix/Linux se poate folosi comanda `time` spre a returna timpul, e.g.

  ```
  time latex l02.tex
  ```

  returnează

  ```
  real 0m1.490s
  user 0m0.191s
  sys  0m0.009s
  ```

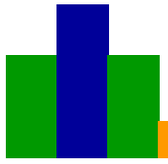
- Timpul total (real) conţine totul, inclusiv I/O, execuţia altor programe, etc.; $(0.191 + 0.009)/1.490 = 0.134$; deci doar 13.4% din timpul total este CPU
- Multe măsurători ignoră timpul sistemului (sys), căci poate depinde de maşină, de sistemul de operare, etc. ceea ce face dificilă comparaţia între calculatoare diferite;



..Masurarea performantei

Masurarea performantei (cont.) Autori folosesc convenţia că:

- *performanţa sistemelor* este în genere raportată prin timpul real când calculatorul este *neâncărcat* cu alte job-uri;
- *performanţă CPU* este "user CPU time" pe un calculator *neâncărcat*.



Masuri de performanta

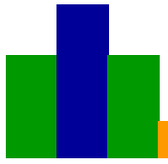
Măsuri de performanţă: Cuvintele *performanţă* şi *timp de execuţie* se pot interschimba, mai exact

$$\text{Performanta}_X = \frac{1}{\text{Timp Executie}_X}$$

Exemple:

- $\text{Performanta}_X > \text{Performanta}_Y$ dnd
  $\text{Timp Executie}_X < \text{Timp Executie}_Y$
- X este de n ori mai rapid ca Y dacă

$$\frac{\text{Performanta}_X}{\text{Performanta}_Y} = \frac{\text{Timp Executie}_Y}{\text{Timp Executie}_X} = n$$



..Masuri de performanta

Masuri de performanta (cont.) Alternativ, se poate folosi *ceasul calculatorului*, (CPU clocks, ticks, etc.) sub forma:

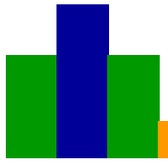
- *perioada ceasului* (cât durează un ciclu de ceas), e.g., 2ns (NanoSecunde)
- *frecvenţa ceasului* (câte cicluri se fac în unitatea de timp), e.g. 500 MHz (MegaHertz-i)

Legatura între *timpul de execuţie* şi *ceas* se face folosind *numărul de cicluri CPU* pentru un program. Atunci

$$\begin{aligned}&\text{Timpul de execuţie (CPU)}\\ &= \text{Nr. de cicluri de ceas} \times \text{Durata unui ciclu}\end{aligned}$$

ori

$$\text{Timpul de execuţie (CPU)} = \frac{\text{Nr. de cicluri de ceas}}{\text{Frecvenţa ceasului}}$$



..Masuri de performanta

Masuri de performanta (cont.)

- Să notăm că numărul de cicluri de ceas pentru un program depinde de numărul de instrucţiuni, dar şi de *numărul mediu CPI de cicluri necesare pentru o instrucţiune* (CPI = Cycles Per Instruction)

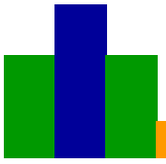
- Obţinem

$$\text{Timp CPU} = \text{Nr. instrucţiuni} \times \text{CPI} \times \text{Durata unui ciclu}$$

ori

$$\text{Timp CPU} = \frac{\text{Nr. instrucţiuni} \times \text{CPI}}{\text{Frecvenţa ceasului}}$$

Pentru o formulă mai exactă, se poate folosi numărul exact *de cicluri pentru fiecare instrucţiune.*



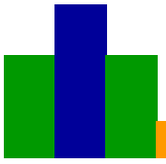
..Masuri de performanta

Masuri de performanta (cont.)

Comentariu - există o interdependenţă strânsă între componentele de mai sus, ceea ce face extrem de dificilă creşterea performanţei.

Exemple:

- Putem micşora numărul de cicluri pe instrucţiune (CPI) folosind în hard instrucţiuni mai complexe; dar atunci ceasul va deveni mai lent, afectând performanţa globală
- Putem sacrifica unele instrucţiuni care se execută mai rar, în sensul de a se translata în cod neperformant; dacă câştigul în eficientizarea celor des folosite este mare, performanţa globală poate fi mai bună.



..Masuri de performanta

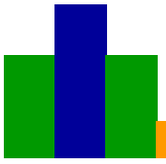
Exemplu: Dacă avem instrucţiuni de tip A,B,C care durează 1,2,3 ciclui, respectiv, care din translatările de mai jos a unei secvenţe de nivel înalt este mai bună:

1. cea cu 2 instrucţiuni de tip A, 1 tip B, şi 2 tip C, ori
2. cea cu 4 tip A, 1 tip B, 1 tip C?

Răspuns: Prima foloseşte 2+1+2=5 instrucţiuni, iar a doua 4+1+1=6. Totuşi,

1. Nr. Cicluri $\text{Ceas}_1 = 2 \times 1 + 1 \times 2 + 2 \times 3 = 10$, iar
2. Nr. Cicluri $\text{Ceas}_2 = 4 \times 1 + 1 \times 2 + 1 \times 3 = 9$

deci a 2-a translatare este mai performantă.



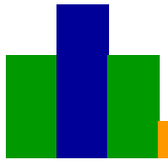
MIPS/FLOPS

MIPS

- MIPS = Millions Instructions Per Second
- $\text{MIPS} = \dfrac{\text{nr. instructiuni}}{\text{timp\_executie} \cdot 10^6}$
- O maşină mai rapidă are un MIPS mai mare

FLOPS

- FLOPS = Floarting Point Operations per Second
- $\text{MFLOPS} = \dfrac{\text{FLOPS}}{\text{timp\_executie} \cdot 10^6}$

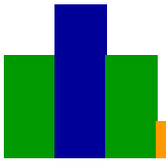


Performanţa de vârf

Performanţa de vârf (peak performance) - când sistemul este foarte încarcat)

- Intel I860
  - 2 operaţii floating point, respectiv 3 intreagi / pe ceas
  - la 50 MHz: 100 MFLOPS şi 150 MOPS
- MIPS R 3000
  - la 33 MHz: 16MFLOPS şi 33 MOPS
- Teoretic
  - I860 de circa 5 ori mai rapid
- SPEC benchmark
  - R3000 a fost cu 15% mai rapidă

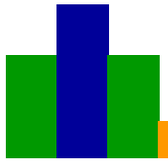
Concluzie: Performanţa de vârf nu este o măsură prea utilă!



Performanta calculatoarelor

Cuprins:

- Executia programelor
- Progresul hardware-ului
- Masurarea performantei
- *Programe de test (benchmark-uri)*
- Concluzii



Alegerea programelor de test

PC-uri:

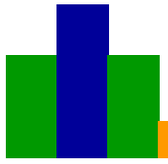
- SPEC - System Performance Evolution Cooperative (SPEC'89, '92, '95, 2000);
  URL: www.spec.org

Servere:

- TPC - Transaction Processing Council (TPC-A '85, TPC-C'92, etc.);
  URL: www.tcp.org

Embedded Computers:

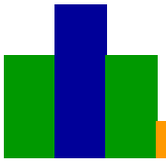
- EEMBC - Embedded Microprocessor Benchmark Consortium: pe categorii, e.g., auto/industrial, consumatori, retea, birou, telecomunicaţii.



..Alegerea programelor de test

Alegerea programelor de test pentru PC-uri: Se pot folosi

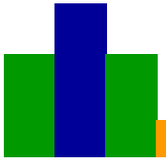
- *aplicaţii reale* - gcc, Word, Photoshop, etc.
- *aplicaţii modificate* - pentru protabilitate, focus pe un aspect al performanţei, etc.
- *Kernels* - programe mici, intens utilizate din aplicaţii reale
- *Toy benchmarks* - benchmarks cu programe mici [10-100 linii de cod], e.g. ciurul lui Erathostenes, Quicksort, etc.
- *Benchmark-uri sintetice* - secvenţe de cod artificiale [nu au un rezultat util]



SPEC

Benchmark-uri SPEC: (vezi şi `www.spec.org`)

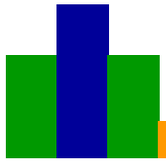
- Evalueaza atât calculatorul, cât şi compilatorul
- In tabelele de mai jos sunt incluse toate programele din SPEC'89, '92, şi '95;
- In SPEC'2000 din seturile '89 şi '92 a ramas doar câte un program (gcc şi swim), iar din setul '95 au rămas 4. Restul de 24 programe au fost abandonate pe parcurs.



..SPEC

Benchmark-uri SPECSPEC benchmarks (cont.1):

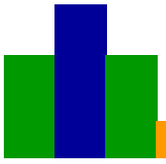
Benchmark name	Integer of FP	SPEC89	SPEC92	SPEC 95	SPEC 2000
gcc	integer	*adopted*	*modified*	*modified*	*modified*
espresso	integer	*adopted*	*modified*	*dropped*	
li	integer	*adopted*	*modified*	*modified*	*dropped*
eqntott	integer	*adopted*	*dropped*		
spice	FP	*adopted*	*modified*	*dropped*	
doduc	FP	*adopted*		*dropped*	
nasa7	FP	*adopted*		*dropped*	
fpppp	FP	*adopted*		*modified*	*dropped*
matrix300	FP	*adopted*	*dropped*		
tomcatv	FP	*adopted*		*modified*	*dropped*



..SPEC

Benchmark-uri SPECSPEC benchmarks (cont.2):

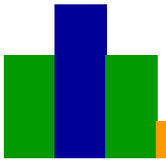
Benchmark name	Integer of FP	SPEC89	SPEC92	SPEC 95	SPEC 2000
compress	integer		*adopted*	*modified*	*dropped*
sc	integer		*adopted*	*dropped*	
mdljdp2	FP		*adopted*	*dropped*	
wave5	FP		*adopted*	*modified*	*dropped*
ora	FP		*adopted*	*dropped*	
mdljsp2	FP		*adopted*	*dropped*	
alvinn	FP		*adopted*	*dropped*	
ear	FP		*adopted*	*dropped*	
swm256 (aka swim)	FP		*adopted*	*modified*	*modified*
su2cor	FP		*adopted*	*modified*	*dropped*
hydro2d	FP		*adopted*	*modified*	*dropped*



..SPEC

Benchmark-uri SPEC (cont.3):

Benchmark name	Integer of FP	SPEC89	SPEC92	SPEC 95	SPEC 2000
go	integer			*adopted*	*dropped*
m88ksim	integer			*adopted*	*dropped*
ijpeg	integer			*adopted*	*dropped*
perl	integer			*adopted*	*modified*
vortex	integer			*adopted*	*modified*
mgrid	FP			*adopted*	*modified*
applu	FP			*adopted*	*dropped*
apsi	FP			*adopted*	*modified*
turb3d	FP			*adopted*	*dropped*

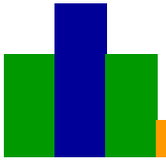


..SPEC

SPEC-CPU2000: Programe de tip întreg

Nume	Tip	Sursa	Descriere
gzip	integer	C	Compresie cu algoritmul Lempel-Ziv
vpr	integer	C	Plasare si rutare de circuite FPGA
gcc	integer	C	Compilator GNU cu optimizare pentru C
mcf	integer	C	Planificare combinatoriala a traficului public
crfty	integer	C	Program de jucat sah
parser	integer	C	Parser privind sintaxa limbii Engleze
eon	integer	C++	Vizualizarea probabilista a trasarii umbrelor
perlmbk	integer	C	Perl cu 4 script-uri de intrare
gap	integer	C	Pachet pentru probleme de teoria grupurilor
vortex	integer	C	Sistem de baze de date orientate pe obiecte
bzip2	integer	C	Algoritm de compresie bazat pe sortare de blocuri
twolf	integer	C	Plasare si rutare VLSI cu “simulated annealing”

(cont.)



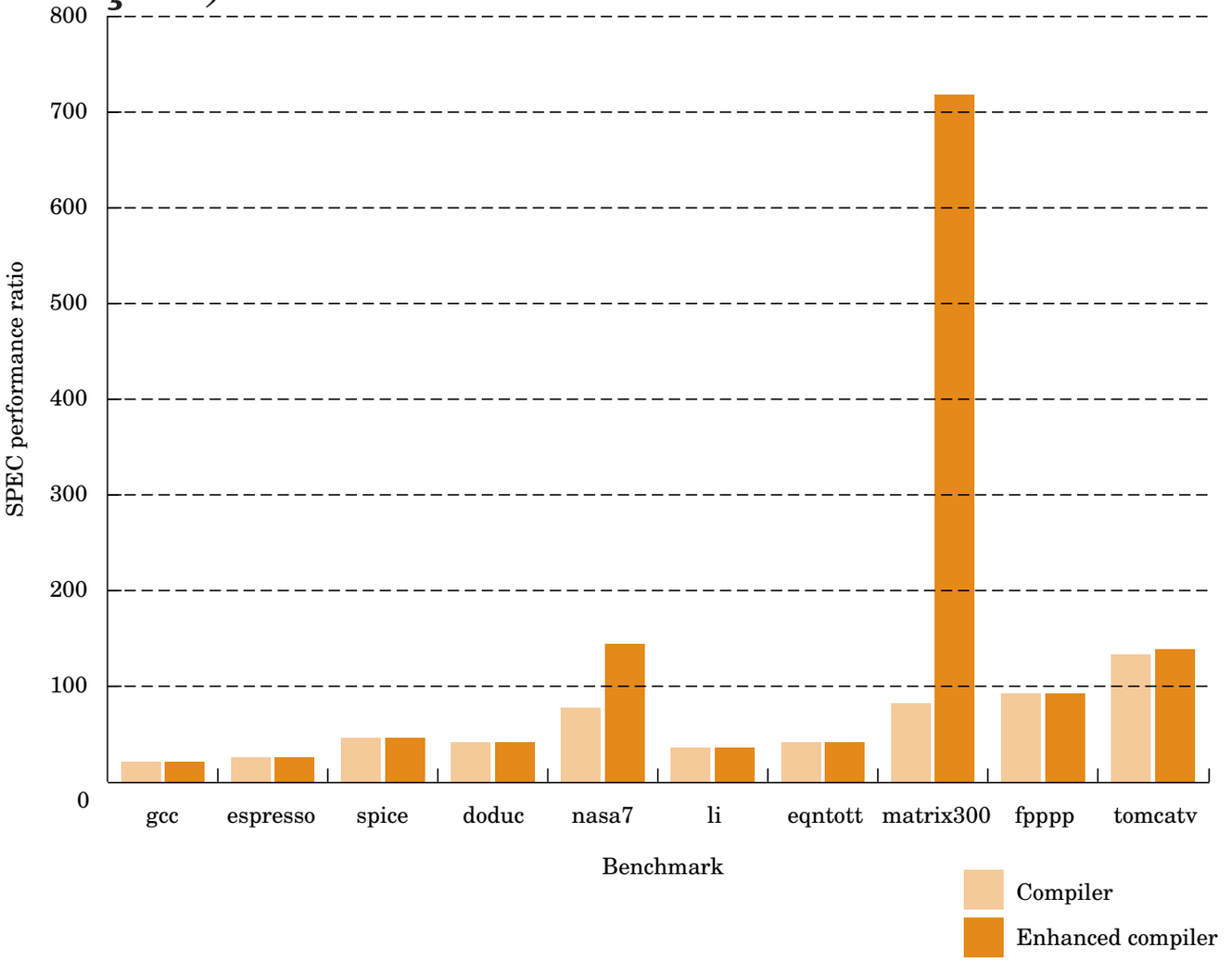
..SPEC

SPEC-CPU2000 (cont.) Programe de tip FP (virgulă-mobilă)

Nume	Tip	Sursa	Descriere
wupwise	FP	F77	Cromo-dinamica cuantica cu “lattice gauge theory”
swim	FP	F77	Miscari la suprafata apei rezolvate cu diferente finite
mgrid	FP	F77	Pachet multi-grid pentru 3-dimensiuni
apply	FP	F77	Pachet pentru ecuatii diferentiale parabolice si eliptice
mesa	FP	C	Biblioteca pentru grafica 3-dimensionala
galgel	FP	F90	Dinamica fluidelor
art	FP	C	Recunoastere de imagini cu retele neurale
equake	FP	C	Propragarea undelor seismice
facerec	FP	C	Recunoasterea fetelor cu “ondulete” & identificari de grafuri
ammp	FP	C	Dinamica proteinelor in apa
lucas	FP	F90	Numere prime Mersenne
fma3d	FP	F90	Simularea ciocnirilor cu metoda elementelor finite
sixtrack	FP	F77	Proiectarea acceleratoarelor in fizica energiilor inalte
apsi	FP	F77	Simularea poluarii atmosferice

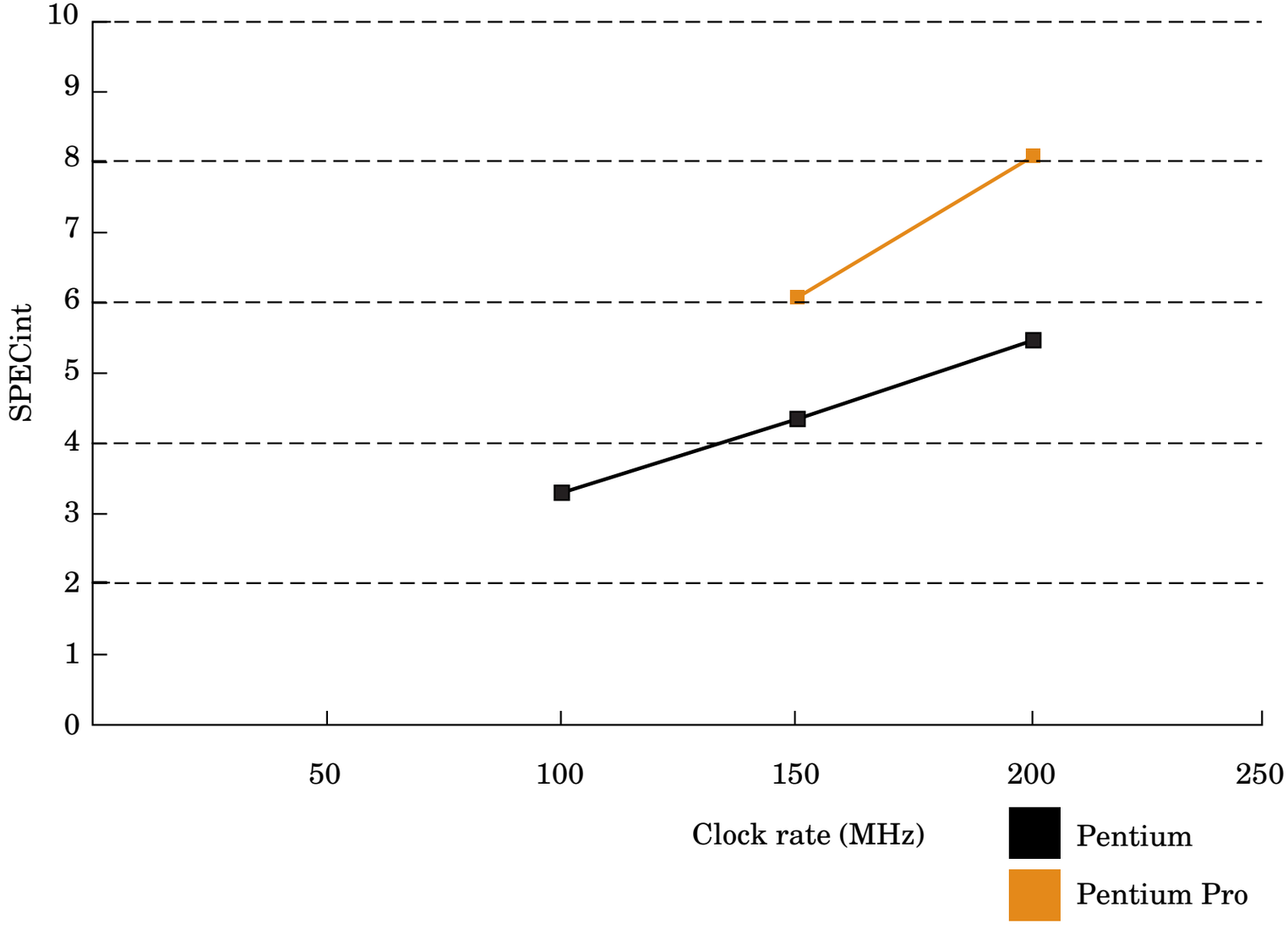
..SPEC

SPEC'89: Figura conţine rezultatele testelor SPEC'89, întregi pe staţia IBM Powerstation 550 cu 2 compilatoare ("performanţa" este "inversa timpului de execuţie")



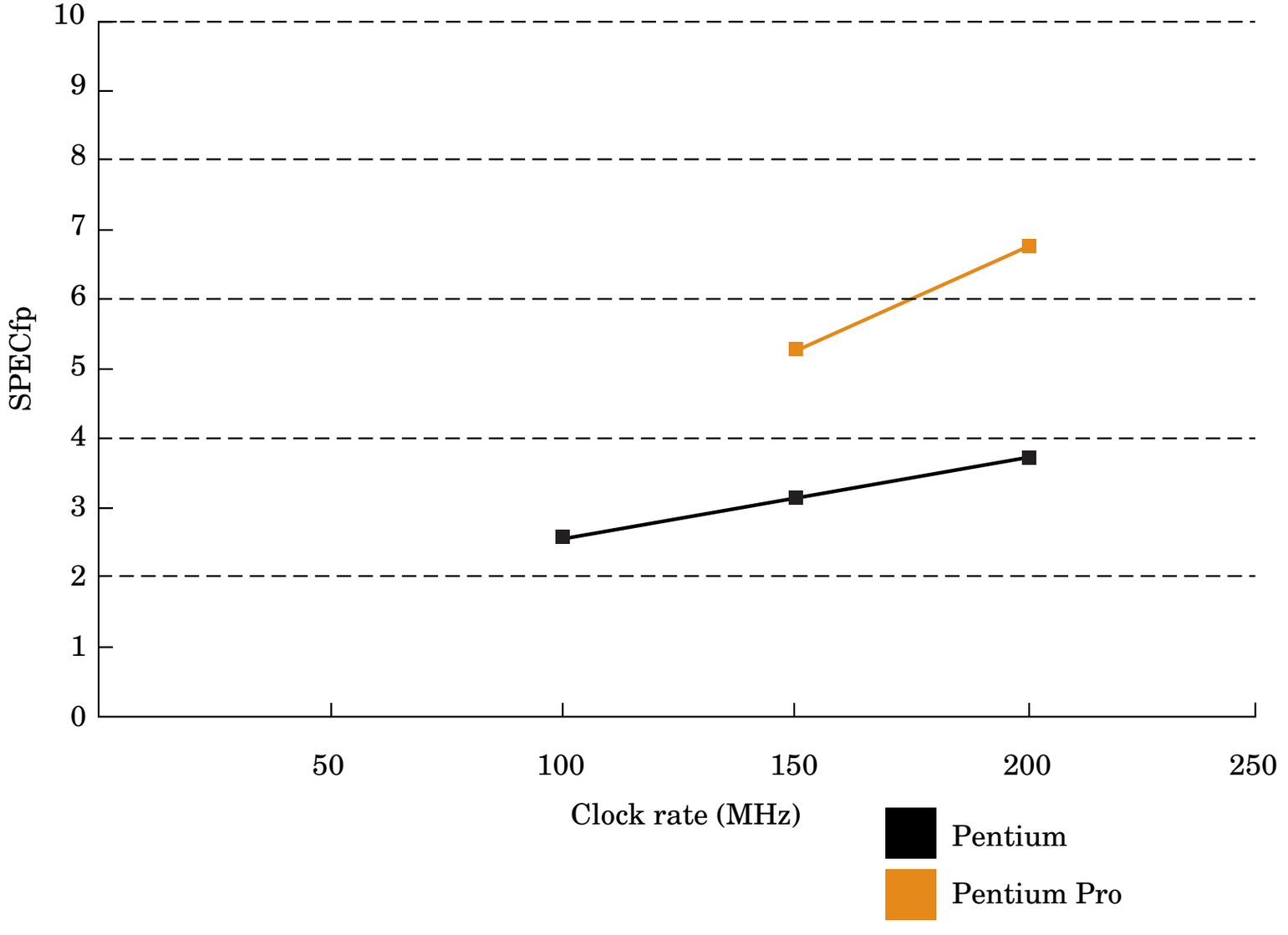
..SPEC

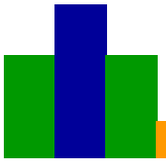
SPEC'95: Spec-Integer'95, Pentium vs. Pentium Pro



..SPEC

SPEC'95: Spec-FP'95, Pentium vs. Pentium Pro





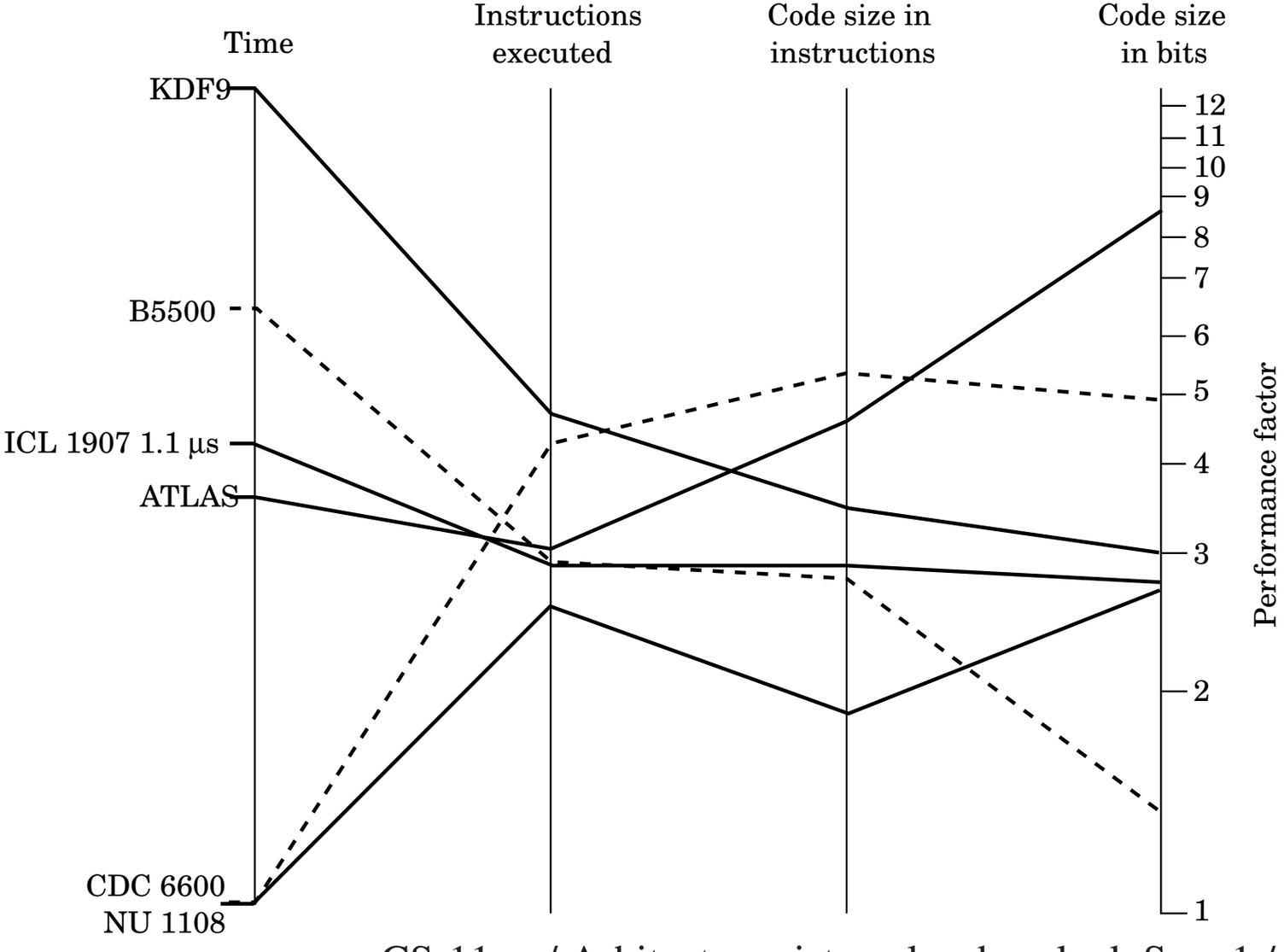
Performanta calculatoarelor

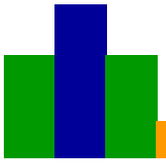
Cuprins:

- Executia programelor
- Progresul hardware-ului
- Masurarea performantei
- Programe de test (benchmark-uri)
- *Concluzii*

Concluzii

Lungime cod vs. performanţă: Lungimea codului şi timpul de execuţie sunt necorelate.





..Concluzii

Parte vs. global:

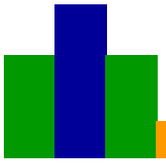
- Imbunătăţirea unei componente într-o proporţie nu îmbunătăţeşte în aceeaşi proporţie întregul.

Performanţa depinde de hardware:

- Metrici care neglijează hardware-ul de tipul lungime cod, etc. nu sunt total obiective.

MIPS-ul e neadecvat:

- Nu consideră ponderea instrucţiunilor, variază cu programul, şi poate varia invers proporţional cu performanţa!



..Concluzii

Benchmark-uri sintetice:

- In programele sintetice compilatoarele pot înlătura o mare parte din cod (25%).
- Nu au relevanţă pentru utilizatorul concret.
- Sfat: Execută rapid cazul frecvent!

Concluzie finală

- Timpul rămâne singura măsură obiectivă în evaluarea performanţei computerelor.