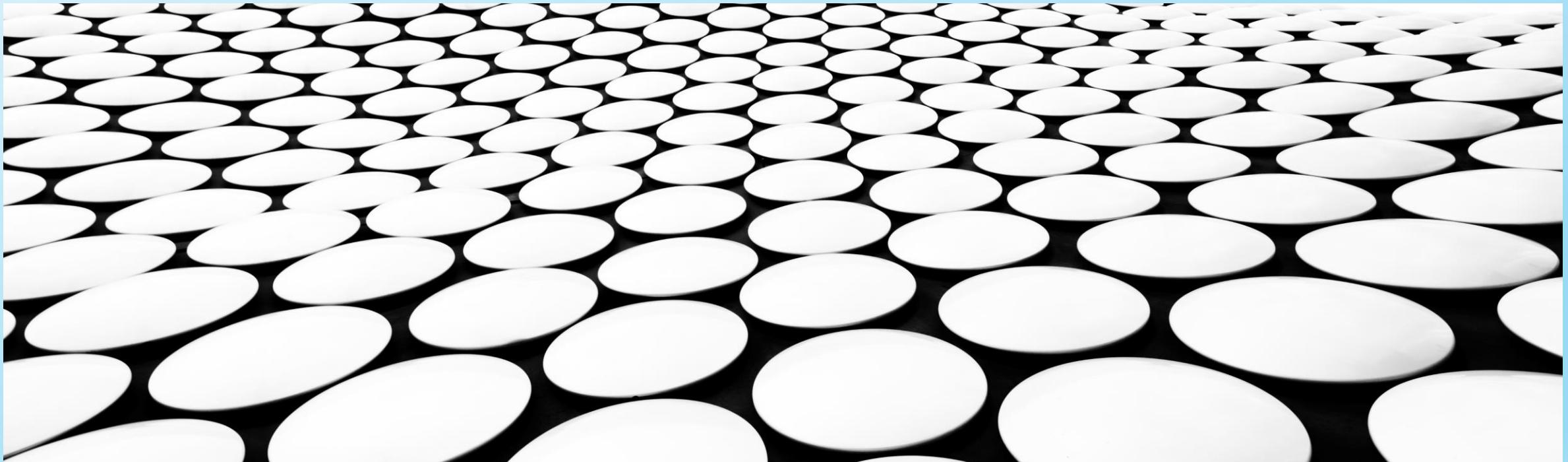

ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



ARHITECTURA SISTEMELOR DE CALCUL

- Conf.Dr. Florin Stanculescu



- Lect.Dr. Alecsandru Chiroscă



- Asist.Dr. Gianina Chiroscă

Adrese institutionale:
florin.stanculescu@unibuc.ro
alecsandru.chiroscă@unibuc.ro
gianina.chiroscă@unibuc.ro

Adrese pentru comunicare informala:
asc@fpce1.fizica.unibuc.ro

SITE-UL CURSULUI

<http://asc.fizica.unibuc.ro/>

BIBLIOGRAFIE RECOMANDATA

Computer Organization and Design

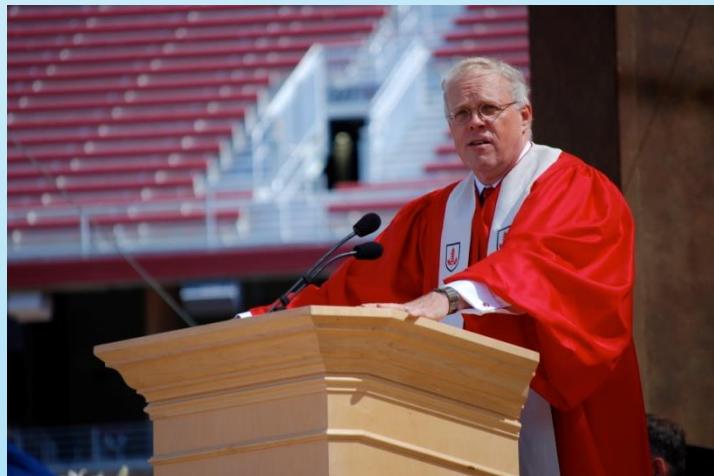
The hardware/Software interface

David Patterson, John Hennessy, Fifth edition, Elsevier 2014

Computer Architecture

A quantitative Approach

John Hennessy, David Patterson, Fifth edition, Elsevier 2012



John L. Hennessy:

- President of Stanford University (2010-2016)
- co-founder of MIPS Computer Systems Inc Marc Andreessen called him "the godfather of Silicon Valley.

Arhitectura si performantele sistemelor de calcul

Arhitectura: Felul în care este construit sau alcătuit ceva
Presupune o concepție vizuală

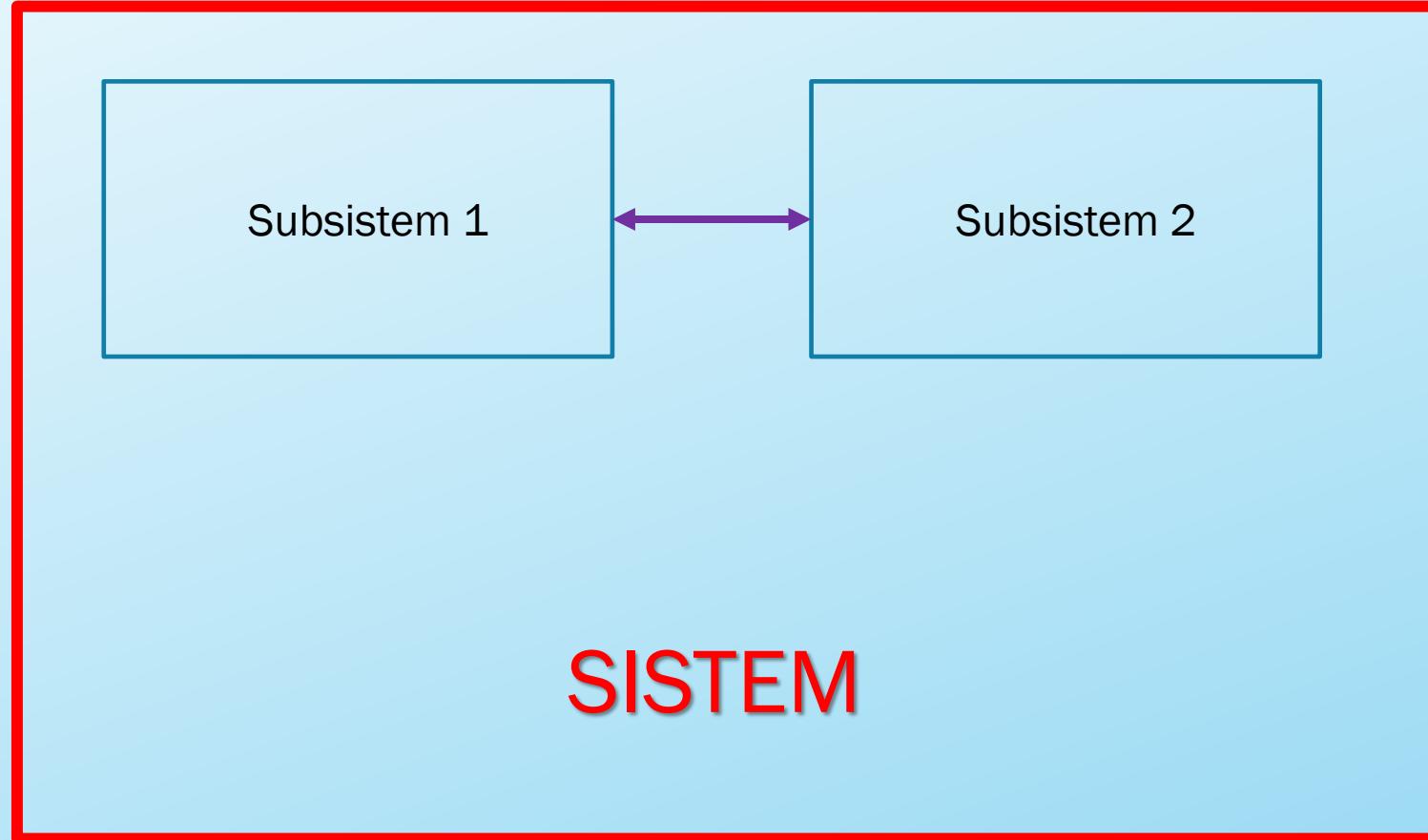
SISTEM: Ansamblu de elemente dependente între ele și
formând un întreg organizat

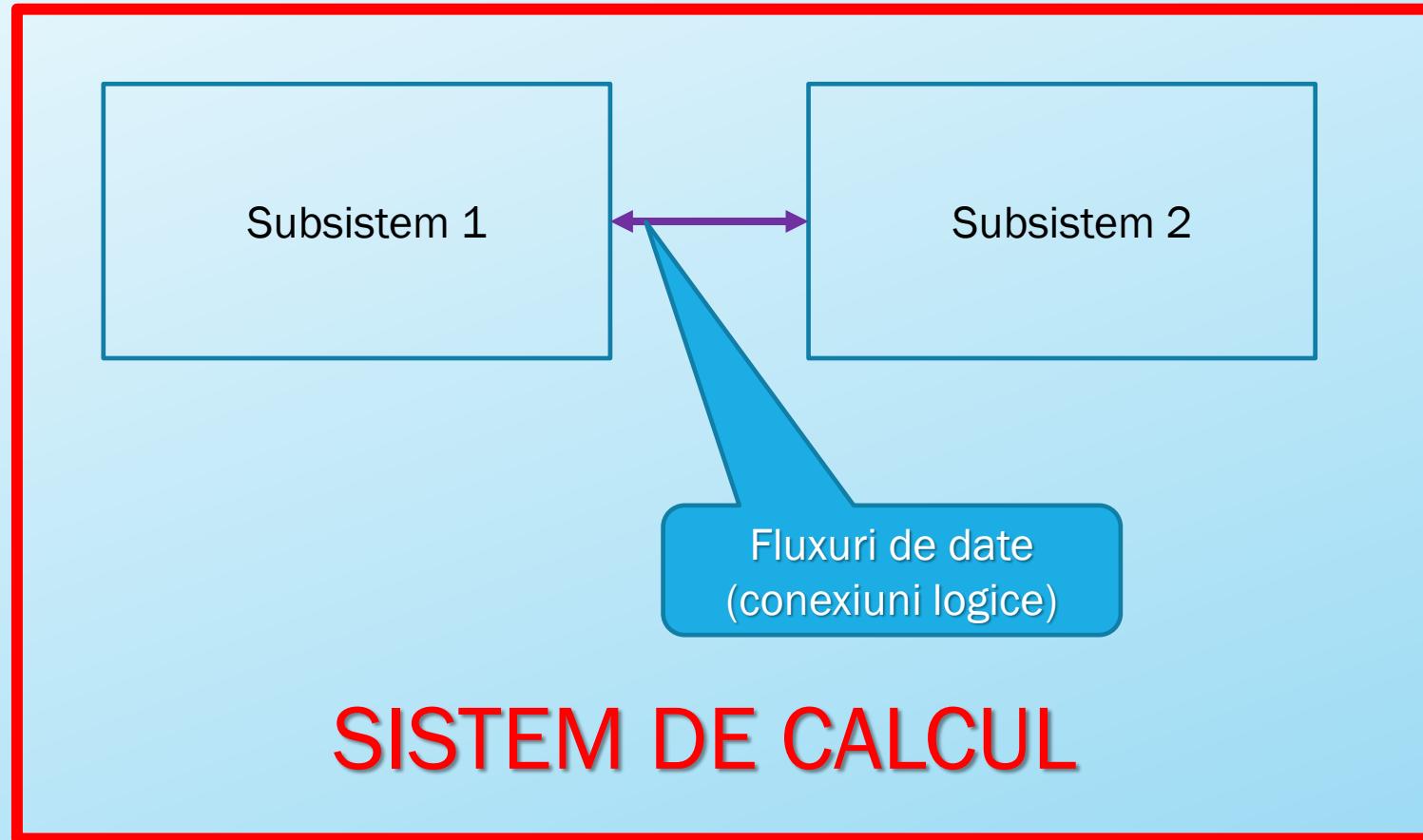
Presupune existența mai multor elemente sau subsisteme;
Minim două elemente

Subsistem 1

Subsistem 2

SISTEM







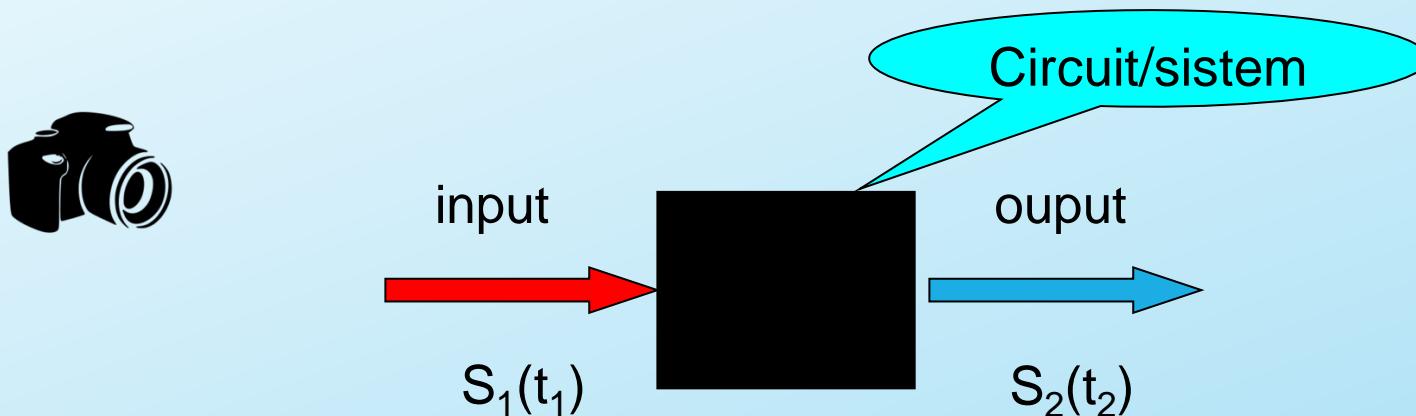
SISTEM DE CALCUL

Canalul fizic de comunicatie este un subsistem al unui sistem de calcul

Un sistem de calcul presupune executia succesiva (fara intrerupere) a unui sir de instructiuni
Performantele se refera, de regula, la executia instructiunilor

- Performantele unui sistem de calcul se pot masura folosind un numar foarte mare de criterii.
 - Criteriile se impart in mai multe categorii.
- De regula, performantele se masoara prin **raportare** la performantele unui **sistem de referinta**.

Latență



S_1 semnalul la intrare; S_2 semnalul la ieșire;
 t_1 momentul de inceput al intrării; t_2 momentul de inceput al ieșirii

$$\text{Latență} = t_2 - t_1 \quad \text{Largimea de Banda (debit)} = 1/L$$

Definiția performanței (asociată cu latență)

$$\boxed{\text{Performanta}(X) = \frac{1}{\text{Latenta}(X)}}$$

Aceasta definitie generala este aplicabila si unui sistem de calcul (X):

De exemplu: t_1 momentul lansarii in executie a unei instructiuni,
 t_2 momentul lansarii in executie a urmatoarei instructiuni

Sporul de performanta (lui Y in raport cu X) =

$$= \frac{\text{Performanta}(Y)}{\text{Performanta}(X)} = \frac{\text{Latenta}(X)}{\text{Latenta}(Y)} = n$$

Sistemul **Y** este de **n** ori mai rapid decat sistemul **X**

Timpul de execuție

- Pentru un sistem de calcul în locul latentei se folosește termenul: **timp de execuție**.
- iar performanța se exprima prin **viteza de execuție**, având ca unitate de măsură:
MIPS (Millions of Instructions Per Second)
 - (A nu se confunda cu ISA MIPS)
- Timpul de execuție se poate referi la diverse activități.
- **Timpul de execuție al unui program** (P) de către un sistem (S) este:

$$timp\ de\ executie(P, S) = \frac{\text{numar\ de\ instructiuni}(P)}{\text{viteza\ de\ executie}(S)}$$

- În anii 1970' viteza unui sistem de calcul era raportată la un calculator VAX 11/780;
 - viteza unui astfel de calculator era considerată 1 milion de instrucțiuni pe secundă.
- Viteza de calcul depinde de frecvența ceasului intern (generatorul de tact)

Ecuăția performanței (procesorului)

$$\frac{\text{timp}}{\text{program}} = \frac{\text{timp}}{\text{ciclu}} \times \frac{\text{cicluri}}{\text{instructiune}} \times \frac{\text{instructiuni}}{\text{program}}$$

Ciclu: ciclul de ceas (generatorul de tact)

timp/ciclu: este *inversul frecvenței* generatorului de tact

cicluri/instrucțiune: este numarul mediu de cicluri de ceas pe instrucțiune

timp/program: timpul consumat de CPU pentru execuția programului

Sunt excluse perioadele de timp în care CPU face altceva.

In aceste condiții se mai introduce o marime:

Numarul de cicluri pe instructiune (CPI) [Cycles Per-Instruction]

- Observatie: Timpul de executie al fiecarui tip de instructiune este diferit!

CPI este o marime medie (**globală**): $CPI = \sum_{i=1}^n F_i \cdot CPI_i$

unde CPI_i este numarul de cicluri al fiecarui tip de instructiune, iar F_i este frecventa de utilizare a instructiunii tip i.

Ansamblul $\{F_i\}$ determina compozitia in instructiuni a programului. [instruction mix]

Compozitia este specifica atat fiecarui program cat si fiecarei ISA

Există o anumita procedura de determinare a compozitiei numita: workload characterization

Exemplu:

Operatia	F_i	CPI_i	$F_i \times CPI_i$
Aritmetica	40%	1	0.4
Cu memoria	40%	4	1.6
De salt	20%	2	0.4
TOTAL (CPI)	100%		2.4

$$nr\ de\ cicluri(P, S) = numarul\ de\ instructiuni(P) \times CPI(S)$$

$$timp\ executie\ program(P) = \frac{nr\ de\ cicluri(P, S)}{frecventa\ generatorului\ de\ tact(S)} =$$

$$= \frac{numarul\ de\ instructiuni(P) \times CPI(S)}{frecventa\ generatorului\ de\ tact(S)}$$

Alta unitate de masura pentru viteza de execuție

FLOPS (scris și flops sau flop/s) este un acronim ce provine de la expresia engleză **floating point operations per second** (tradus: operații în virgulă mobilă pe secundă).

Name	FLOPS
<u>yotta</u> FLOPS	10^{24}
<u>zetta</u> FLOPS	10^{21}
<u>exa</u> FLOPS	10^{18}
<u>peta</u> FLOPS	10^{15}
<u>tera</u> FLOPS	10^{12}
<u>giga</u> FLOPS	10^9
<u>mega</u> FLOPS	10^6
<u>kilo</u> FLOPS	10^3

- Estimare
 - Majoritatea procesoarelor contemporane sunt caracterizate de 4 FLOPS/ciclu ceas
 - În aceste condiții un single-core la 2.5GHz va avea o performanță maximă de 10 GFLOPS

Legea Amdahl

Se referă la creșterea în performanță a unui sistem de calcul atunci când este îmbunătățita numai o porțiune (componentă) din sistem.

$$S = \frac{1}{(1 - f) + f/K}$$

S: este factorul global de creștere a performanței;

f: fracția din activitate realizată de componentă îmbunătățită,

K: creșterea în performanță a componentei

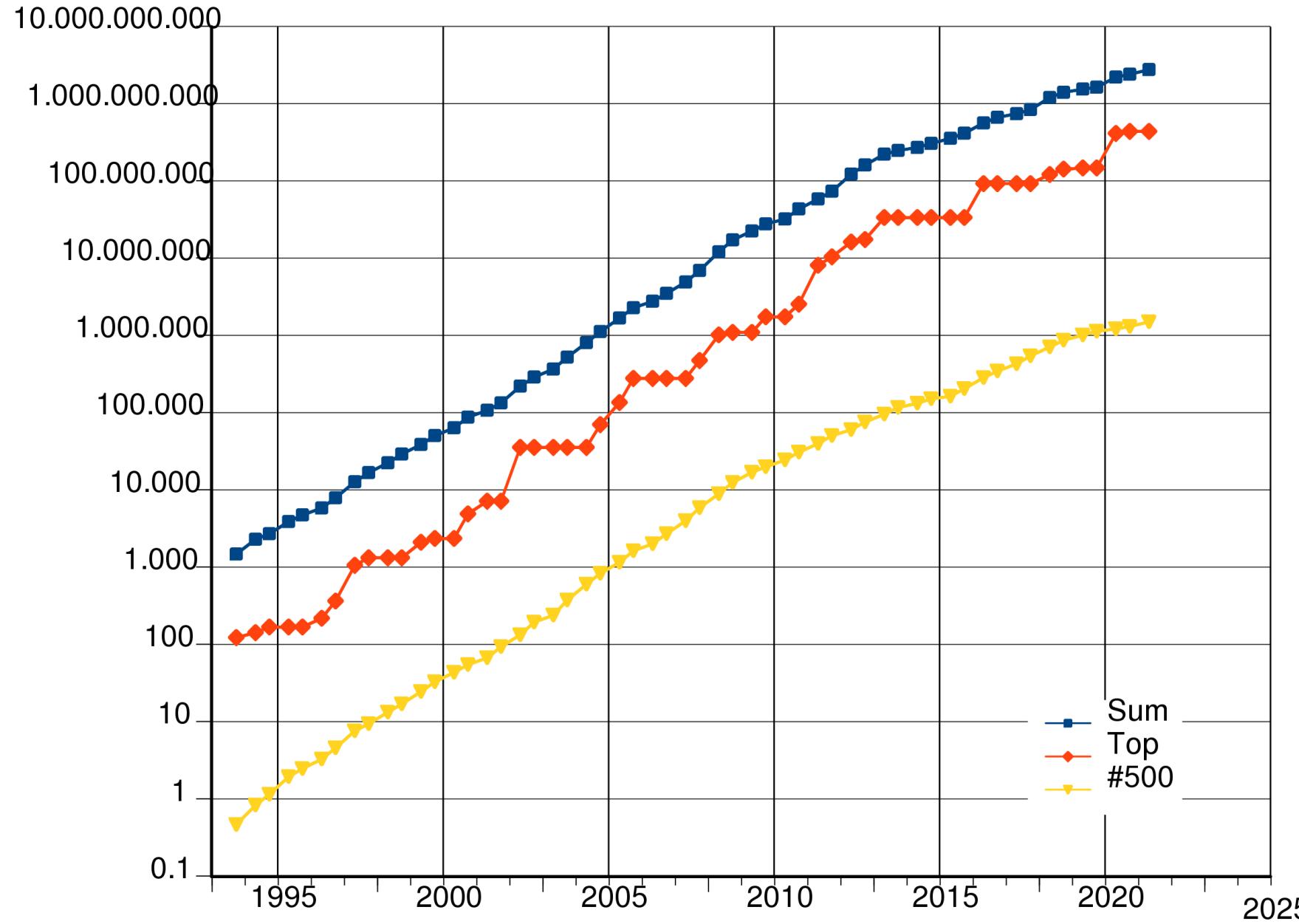
In particular se poate calcula impactul cresterii performantei unei magistrale asupra performantei intregului sistem

Performantele supercomputerelor

1993	Intel Paragon XP/S 140	0.143 teraflops
1997	Intel's ASCI Red	1.338 teraflops
2002	NEC Earth Simulator	35.86 teraflops
2007	IBM Blue Gene/L	478 teraflops
2008	Cray XT Jaguar	1.64 petaflops
2009	Cray Jaguar	1.75 petaflops
2010, oct	Tianhe-IA,(China)	2.56 petaflops
2011, nov.	Fujitsu K computer (Japan)	10.51 petaflops
2012, iun.	IBM's Sequoia	16.32 petaflops
2012, nov.	Titan/Cray Inc	17.59 petaflops
2013, iunie	China's Tianhe-2	33.86 petaflops
2016	Sunway TaihuLight (Wuxi, China)	93.01 petaflops
2018	IBM Summit	122.3 petaflops
2020	Supercomputer Fugaku (Fujitsu)	537 petaflops
2023	EI Capitan (Cray)	1.5 exaflops

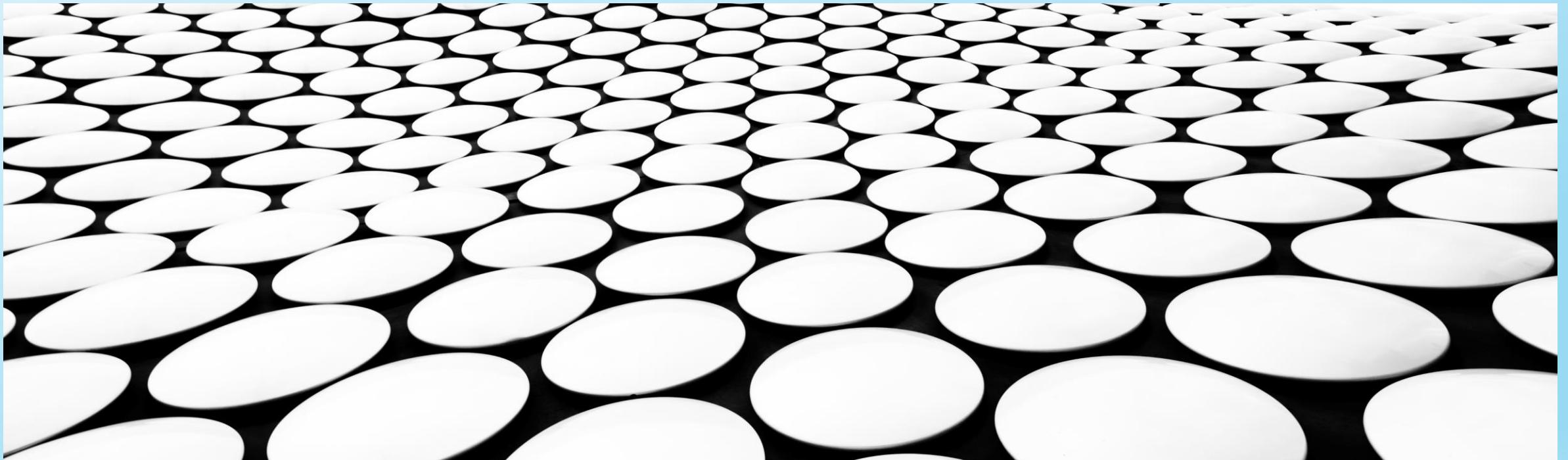
sursa:
Top500

Gflops



ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



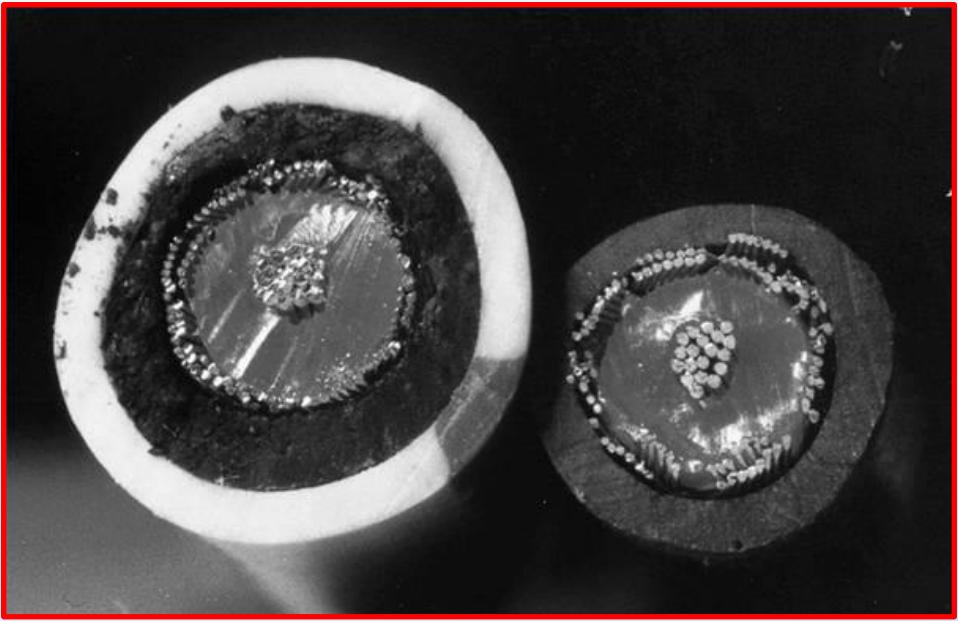
Linii de transmisie

- Caracteristicile si aplicatiile liniilor de transmisie
- Fizica liniilor de transmisie
 - Reflexii si alte distorsiuni in liniile de transmisie si metode de a le reduce efectele.

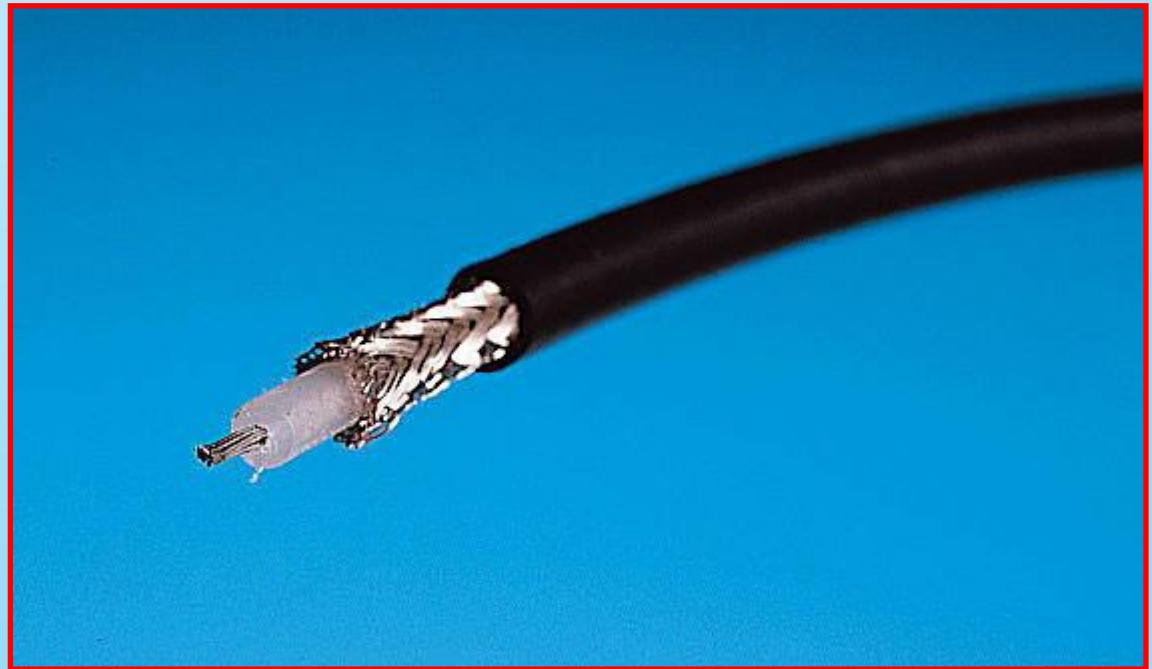
exemple



**LINIE DE TRANSMISIE COAXIALA
CONECTORI
TERMINATOR DE 50Ω**

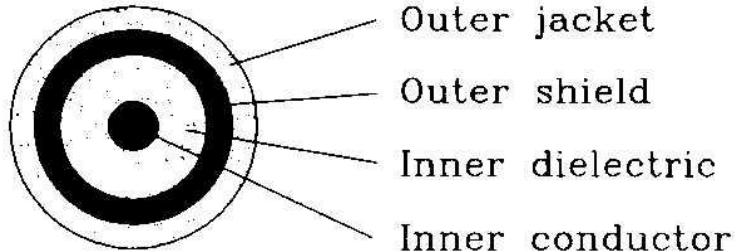


SECTIUNE IN CABLUL COAXIAL

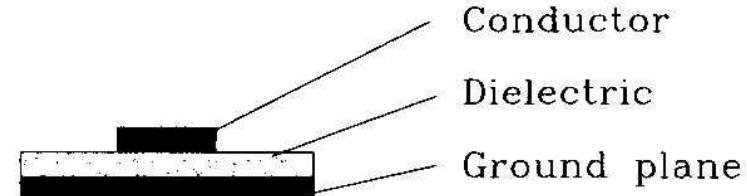


Principalele tipuri de linii de transmisie

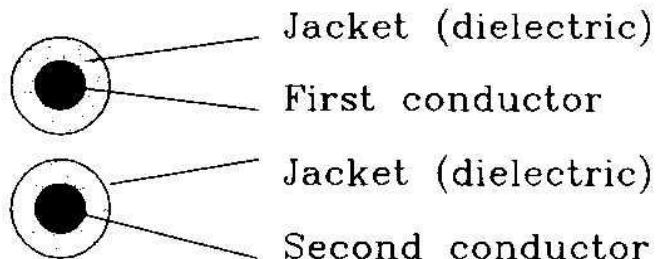
Coaxial cable



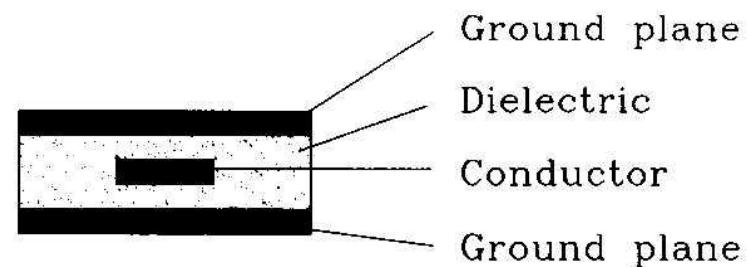
Microstrip



Twisted pair



Stripline

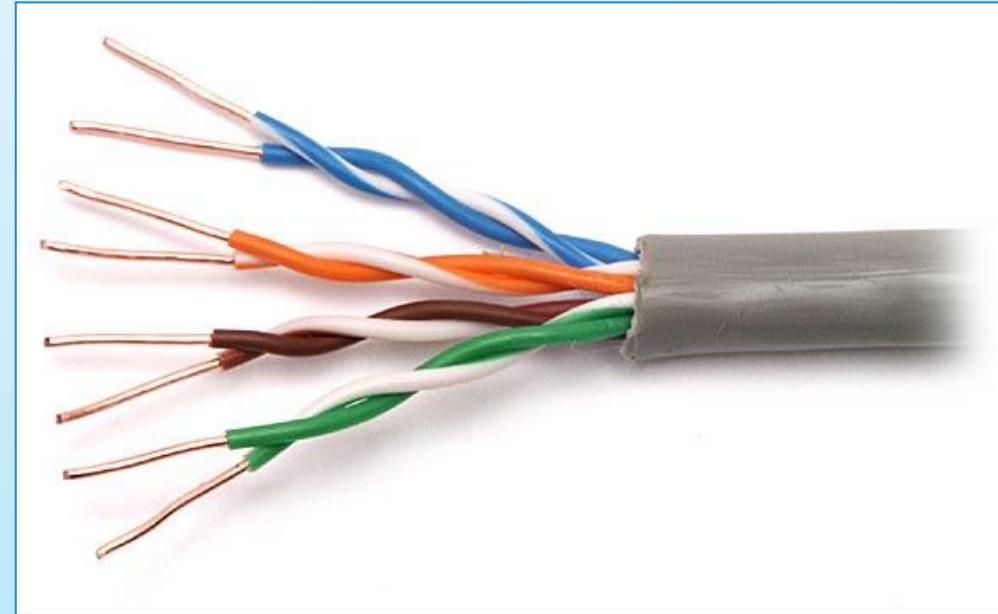


CABLU COAXIAL;

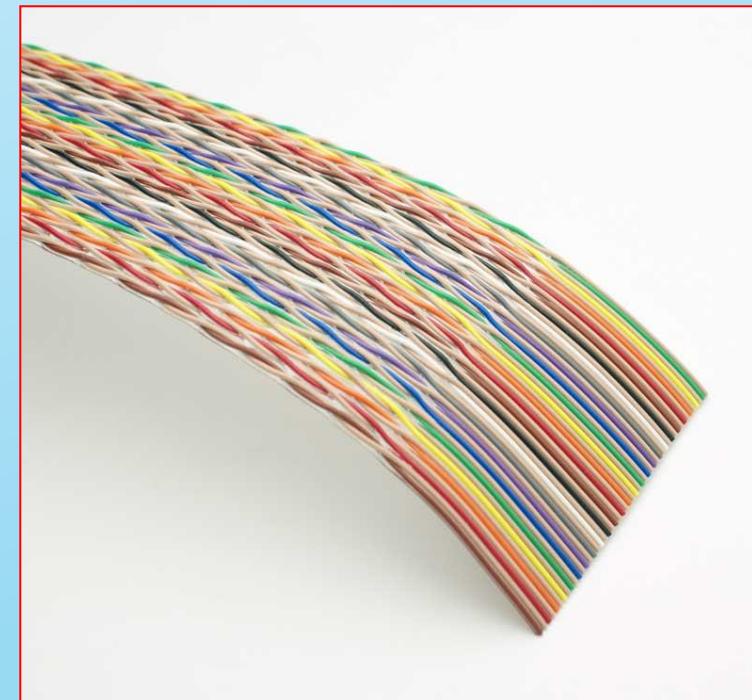
MICROBANDA

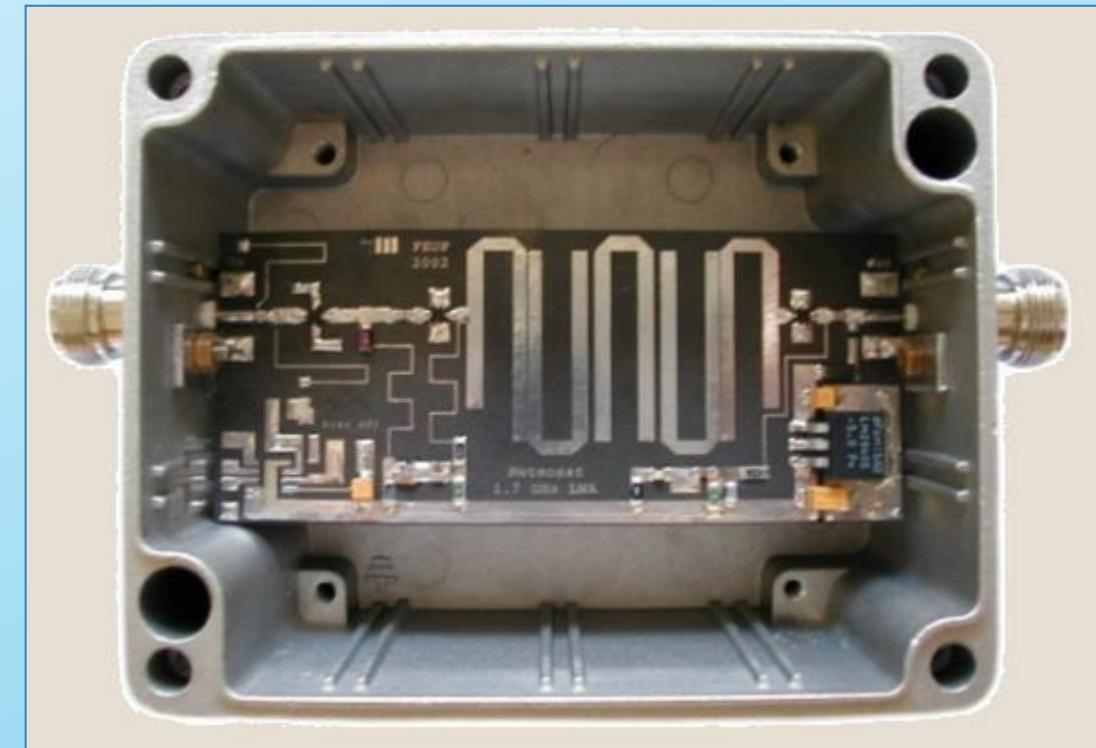
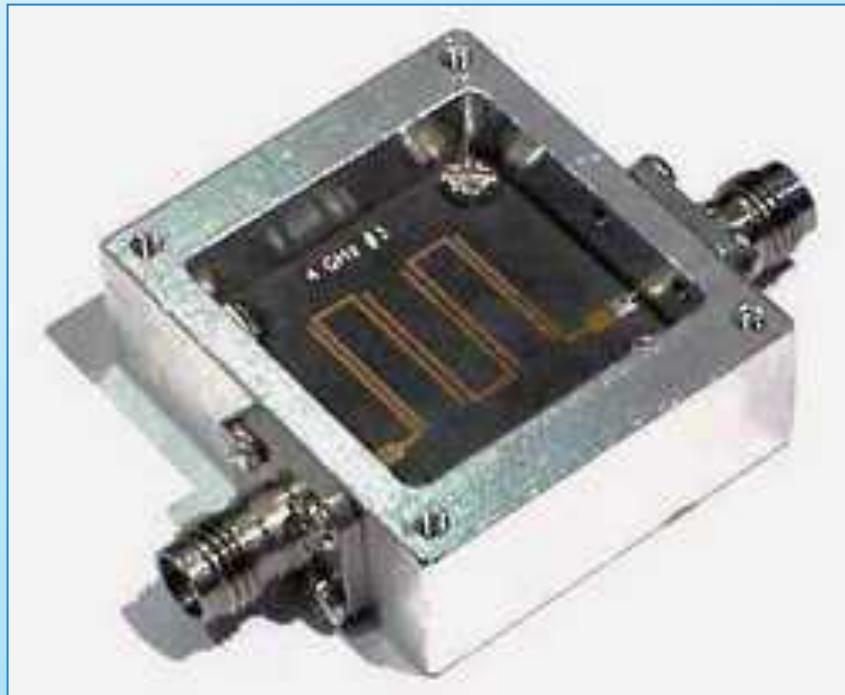
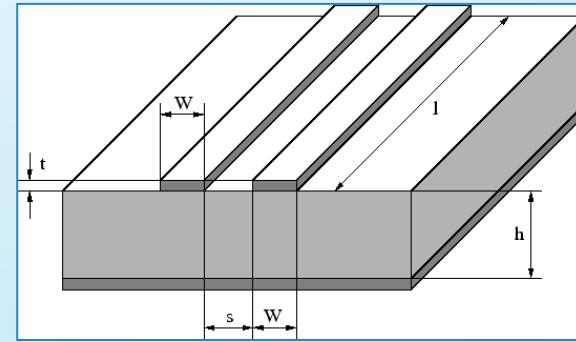
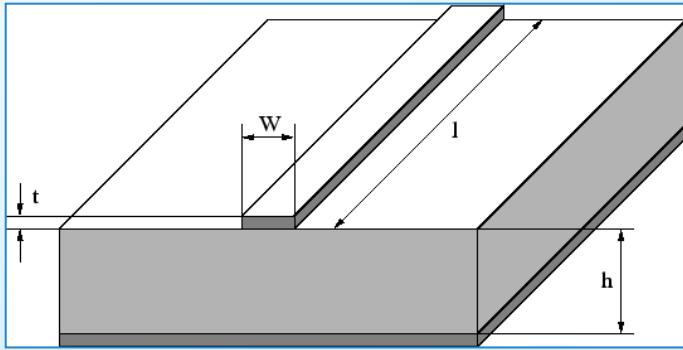
CABLU TORSADAT

BANDA



CABLURI TORSADATE

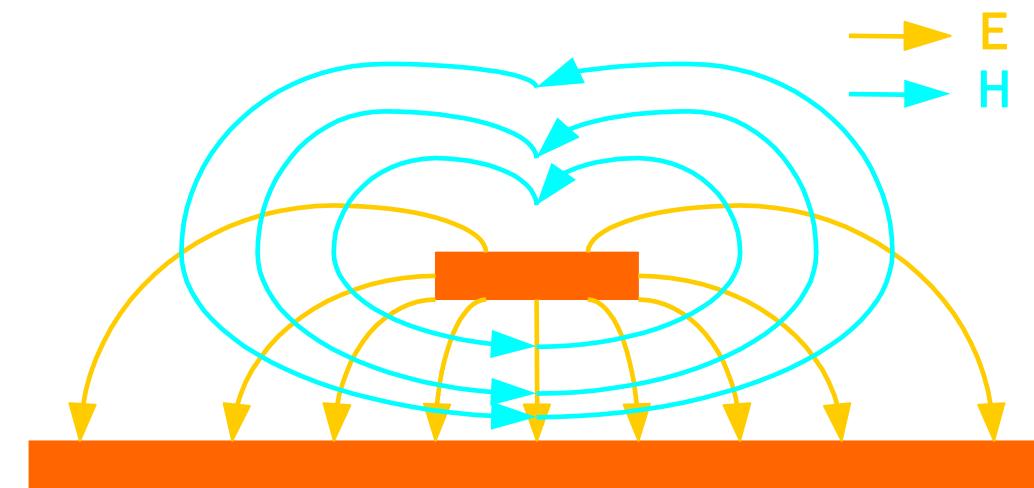




MICROBANDA (MICROSTRIP)

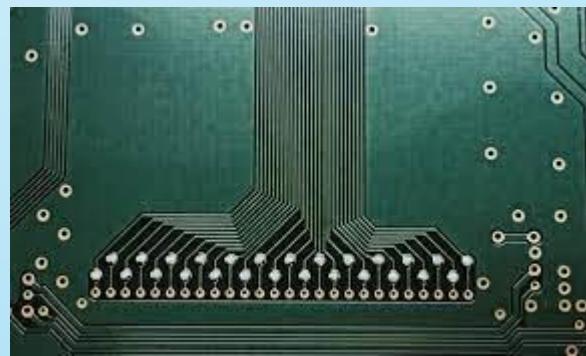
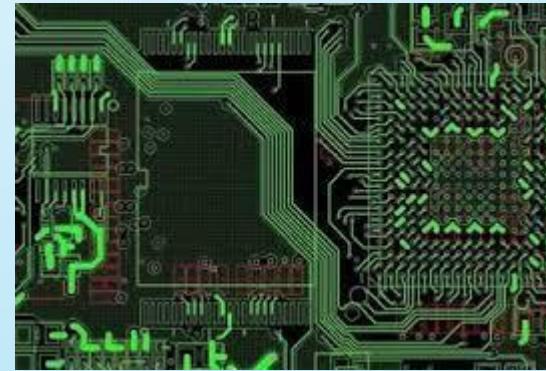
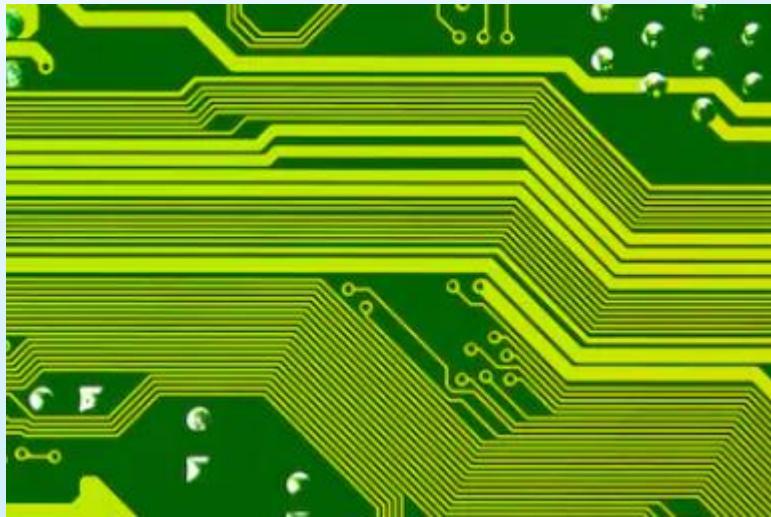


x

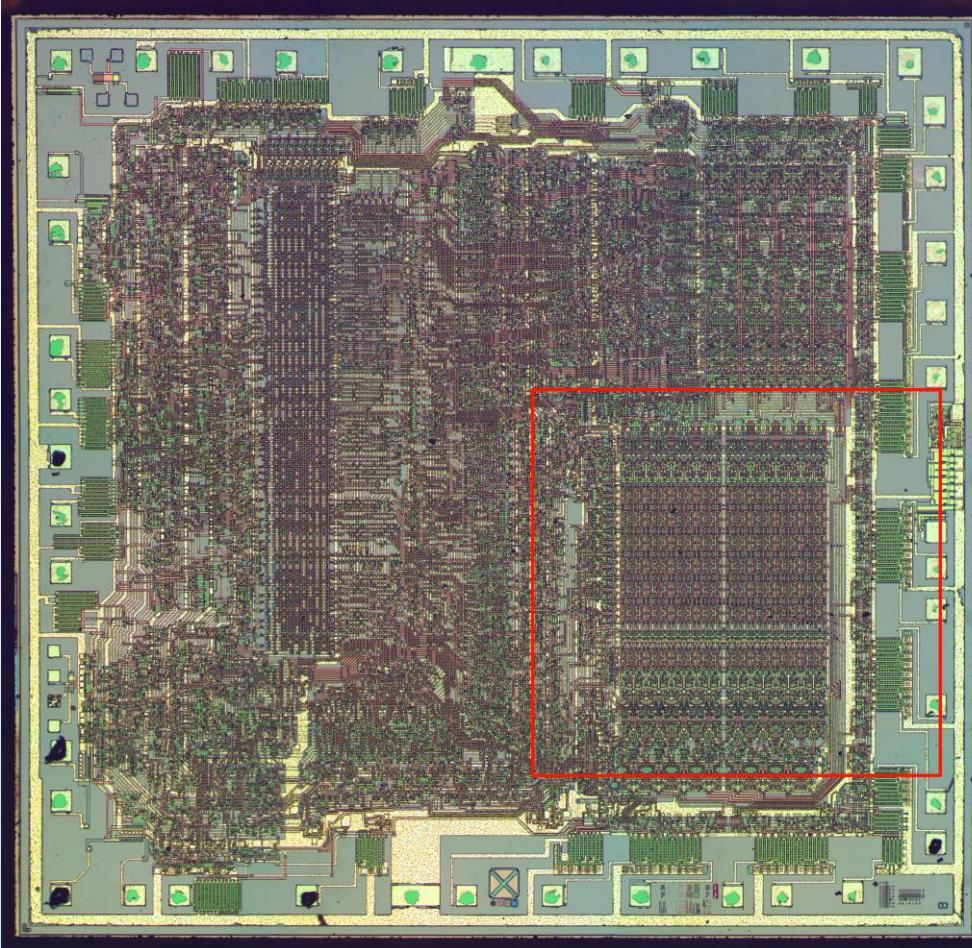


E

H



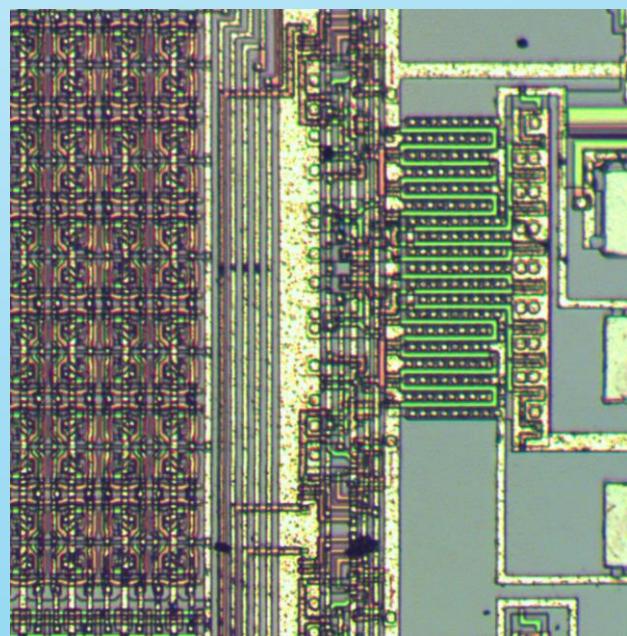
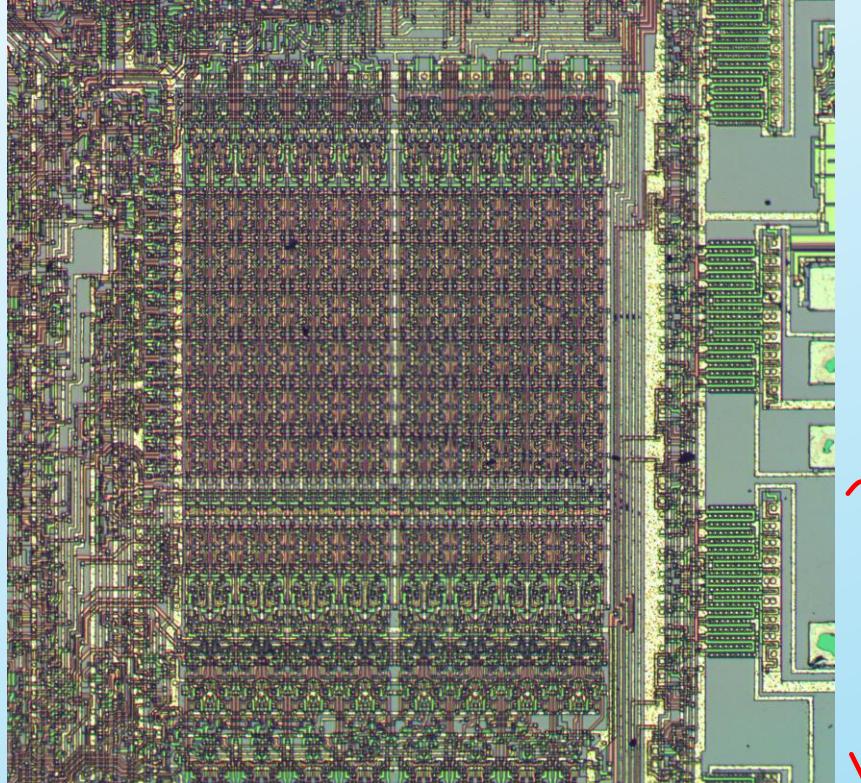
Diverse arhitecturi microstrip pe placi de baza
in calculatoare sau in alte dispozitive de calcul.



Microprocesor Z80A

Tehnologie $5\mu\text{m}$ (1976)

Dimensiune cip $4\ 950 \times 4\ 720\ \mu\text{m}$



microstrip



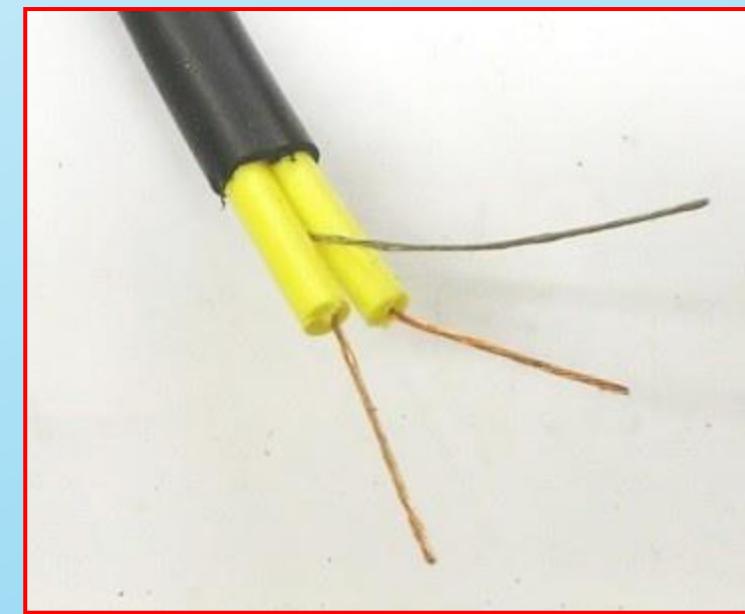
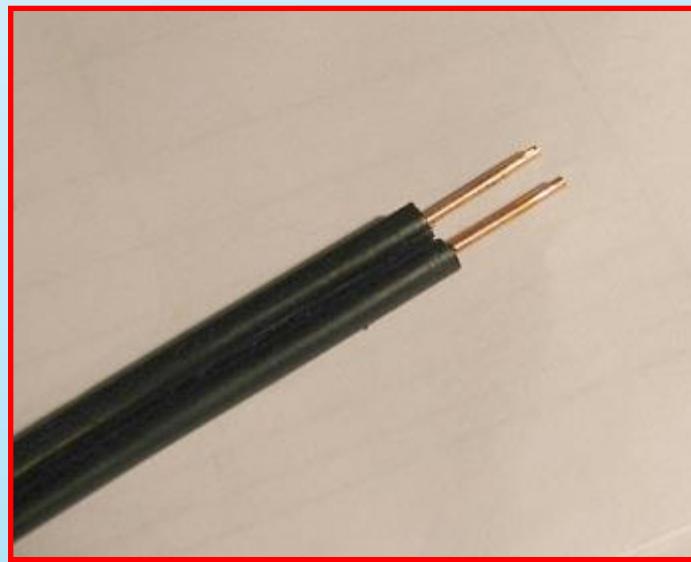
STRIPLINE



Store No . 317123



LINIE PARALELA (TWIN LEAD)



CABLURI TORSADATE

AVANTAJE

Distorsiuni mici

Inter-influente mici

Radiatie redusa

DEZAVANTAJE

Consum de energie

APLICATII

Cabluri de retea

Cabluri usb

Conexiuni cablate lungi (circuite imprimante)

Characteristicile liniei ideale

Fara pierderi (de energie);

Lungime infinita;

Transmisie nedistorsionata a semnalelor;

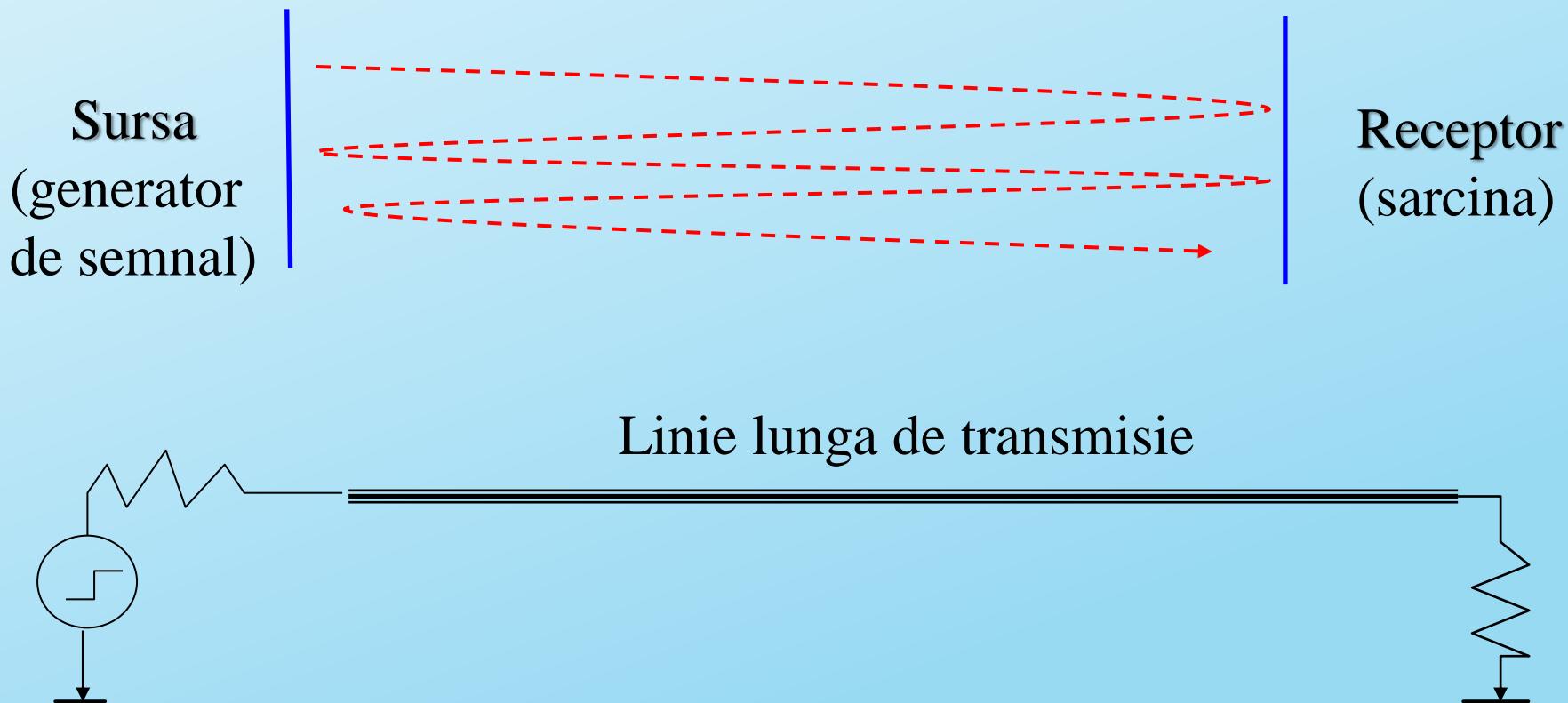
Semnale transmise cu intarziere. Timpul de intarziere este proportional cu distanta parcursa.

$$\tau_l = \sqrt{L_l \cdot C_l}$$

τ_l este timpul de intarziere (s/m) L_l inductanta pe unitatea de lungime (H/m), iar C_l capacitatea pe unitatea de lungime (F/m)

Problema liniei de transmisie (țiuitul)

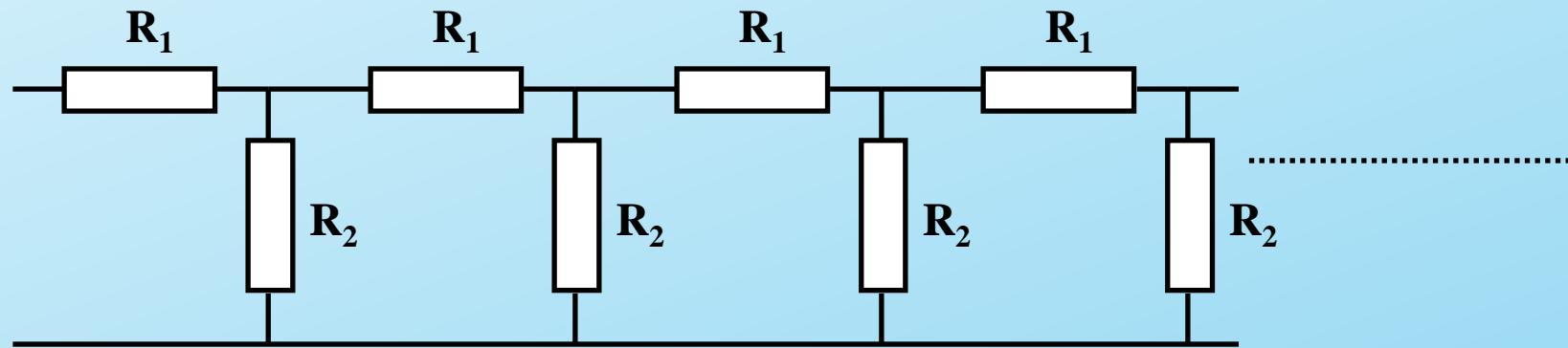
SEMNALUL NU SE TRANSMITE INSTANTANEU. SE TRANSMITE SUB FORMA DE UNDE.
APARE FENOMENUL DE REFLEXIE MULTIPLA.



Linia cu pierderi

LINIA OHMICA

Doua fire metalice echidistante, separate de un izolator, la frecventa f. joasa.

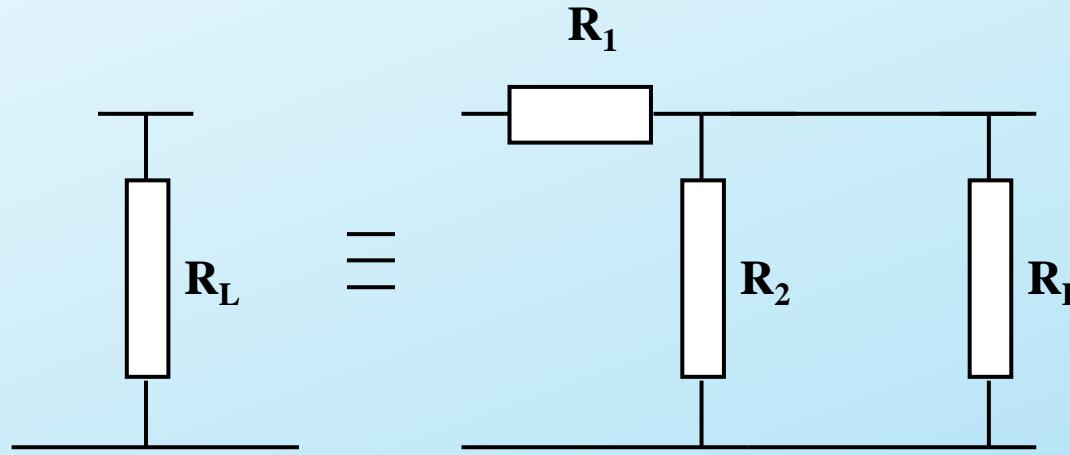


Semnificatii

R_1 rezistenta firelor pe unitatea de lungime

R_2 rezistenta de izolatie pe unitatea de lungime

LINIA ESTE ECHIVALENTA CU O REZISTENTA R_L



Daca adaugam un grup suplimentare R_1 , R_2 , rezistenta liniei nu se schimba, deoarece linia este infinita (R_L)

$$R_1 + \frac{R_2 \cdot R_L}{R_2 + R_L} = R_L$$

$$R_L = \frac{R_1}{2} \left[1 + \sqrt{1 + 4 \frac{R_2}{R_1}} \right]$$

In cazul unei linii reale $R \rightarrow Z$ (complex), iar formula se pastreaza.

In cazul liniei ideale (fara pierderi)

$R_1 \rightarrow \omega L$ (bobina ideală), iar $R_2 \rightarrow 1/\omega C$ (condensator ideal)

$$Z_L = \frac{Z_1}{2} \left[1 + \sqrt{1 + 4 \frac{Z_2}{Z_1}} \right]$$

$$Z_L = \frac{\omega L_1}{2} \left[1 + \sqrt{1 + 4 \frac{1}{\omega C_2 \omega L_1}} \right] \quad \frac{1}{C_2 L_1} = \omega_r^2$$

$$\omega \ll \omega_r \quad Z_L \cong \frac{\omega L_1}{2} \left[\sqrt{4 \frac{1}{\omega C_2 \omega L_1}} \right] = \sqrt{\frac{L_1}{C_2}}$$

ω_r este frecventa de rezonanta a liniei

Impedanta liniei de transmisie nu depinde de frecventa la frecvente mult mai mici decat frecventa de rezonanta.

**Cazul general: linie reală, cu pierderi: bobina reală
(inductanță în serie cu rezistență) și condensator real
(capacitate în paralel cu o rezistență)**

$$R_1 \rightarrow R_1 + j\omega L_1 \quad 1/R_2 \rightarrow 1/R_2 + j\omega C_2 = G_2 + j\omega C_2$$

$$Z_L = \sqrt{(R_1 + j\omega L_1)/(G_2 + j\omega C_2)}$$

$$v(z, t) = V(x)e^{j\omega t} \quad V(x) = V_0 e^{-\gamma z} + V_1 e^{+\gamma z}$$

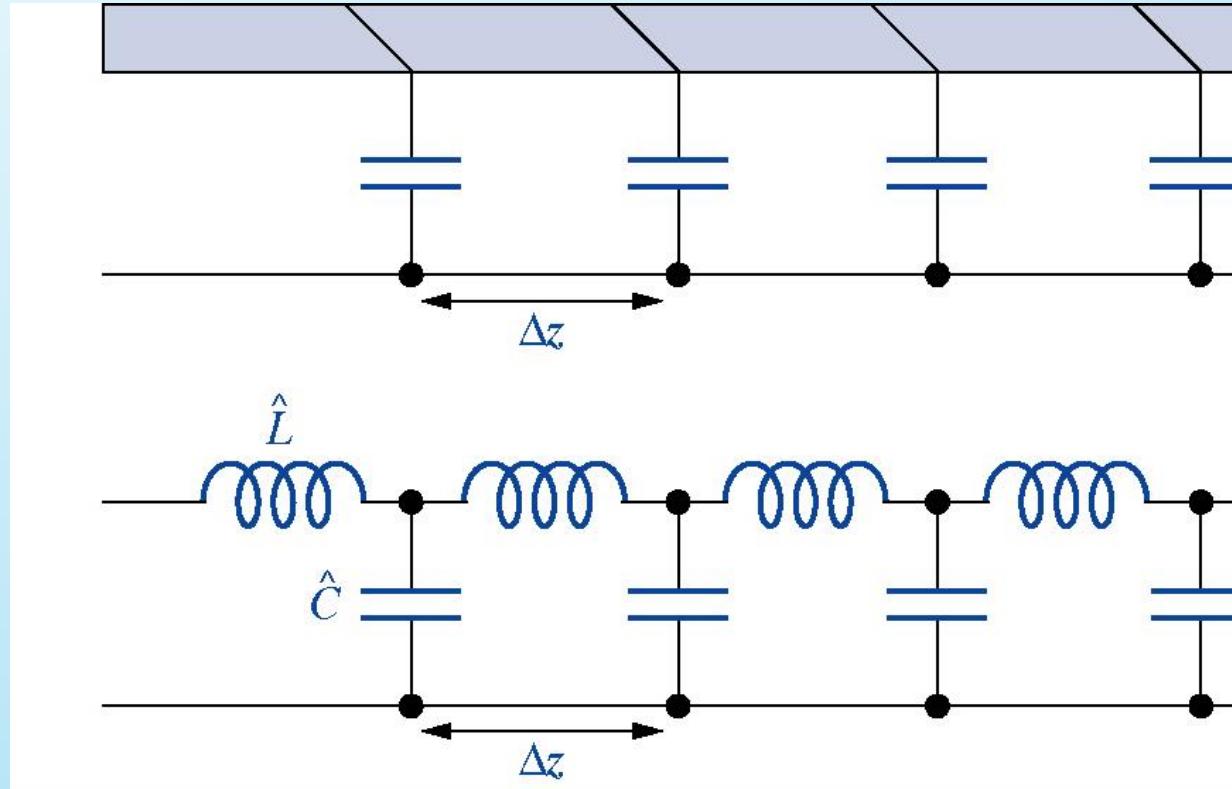
$$\gamma = \sqrt{(R_1 + j\omega L_1) \cdot (G_2 + j\omega C_2)}$$

In cazul liniei ideale Z_L este de natură **rezistivă**.
(partea imaginara este nula)

Conditia de reflexie 0 la capetele liniei:

$$Z_{\text{sursa}} = Z_{\text{linie}} = Z_{\text{sarcina}} \text{ (toate 3 rezistive)}$$

In caz contrar apare o unda inversa
care preia o parte din energia transmisa prin linie.



$$L = \hat{L} \cdot \Delta z$$

$$C = \hat{C} \cdot \Delta z$$

Line	Inductance	Capacitance
Coaxial cable	$L = \frac{\mu}{2\pi} \ln\left(\frac{b}{a}\right) \Delta z$	$C = \frac{2\pi\epsilon}{\ln(b/a)} \Delta z$
Microstrip line	$L = \frac{\mu d}{w} \Delta z$	$C = \frac{\epsilon w}{d} \Delta z$
Twin lead	$L = \frac{\mu}{\pi} \cosh^{-1}\left(\frac{D}{2a}\right) \Delta z$	$C = \frac{\pi\epsilon}{\cosh^{-1}(D/2a)} \Delta z$

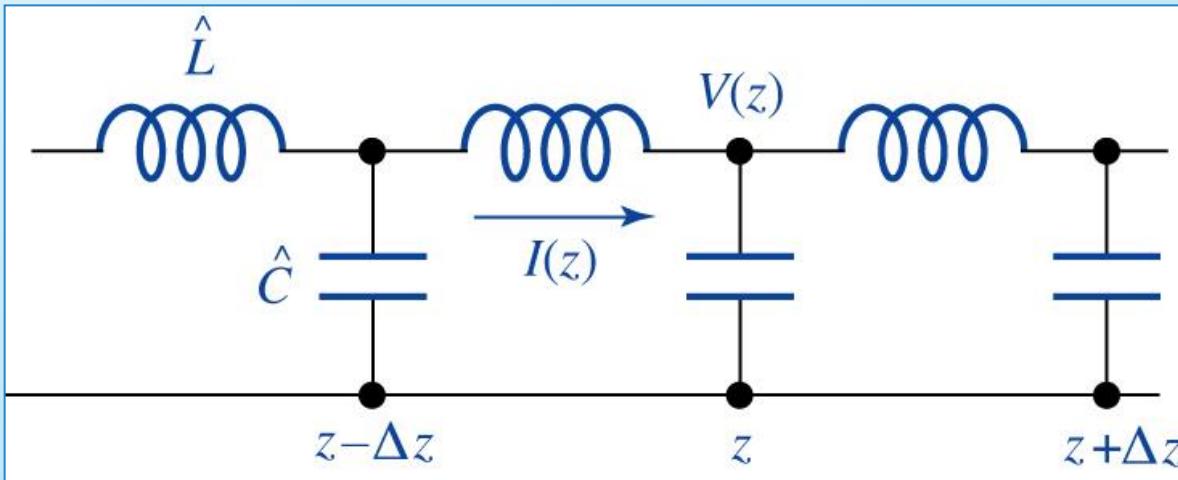
μ permeabilitatea magnetica

ϵ permitivitatea electrica

A, b, d, D caracteristici de dimensiune (in sectiune)

Δz lungime linie

Ecuatiile telegrafistului (ec. Heaviside)



Cazul liniei fara pierderi

$$\frac{\partial I(z,t)}{\partial z} = -\hat{C} \frac{\partial V(z,t)}{\partial t}$$

$$\frac{\partial V(z,t)}{\partial z} = -\hat{L} \frac{\partial I(z,t)}{\partial t}$$

Ecuatiile anterioare se pot rescrie astfel:

$$\frac{\partial^2 I(z,t)}{\partial z^2} - \hat{L}\hat{C} \frac{\partial^2 I(z,t)}{\partial t^2} = 0$$

Ecuatia undei de curent

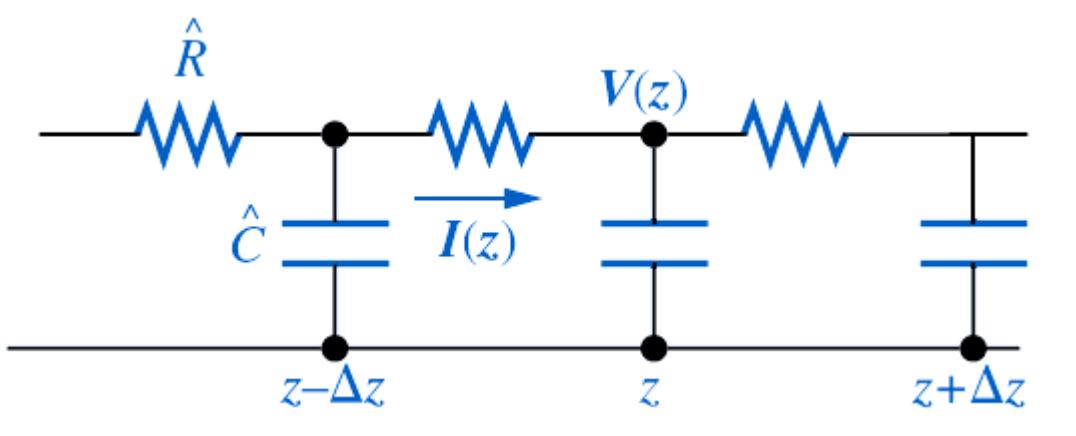
$$\frac{\partial^2 V(z,t)}{\partial z^2} - \hat{L}\hat{C} \frac{\partial^2 V(z,t)}{\partial t^2} = 0$$

Ecuatia undei de potential

$$v = \frac{1}{\sqrt{\hat{L}\hat{C}}}$$

Viteza de propagare undelor

Linie cu pierderi



$$\Rightarrow \frac{\partial V(z,t)}{\partial z} = I(z,t) \hat{R}$$
$$\frac{\partial I(z,t)}{\partial z} = \hat{C} \frac{\partial V(z,t)}{\partial t}$$

Aceste ecuatii se pot recombină astfel:

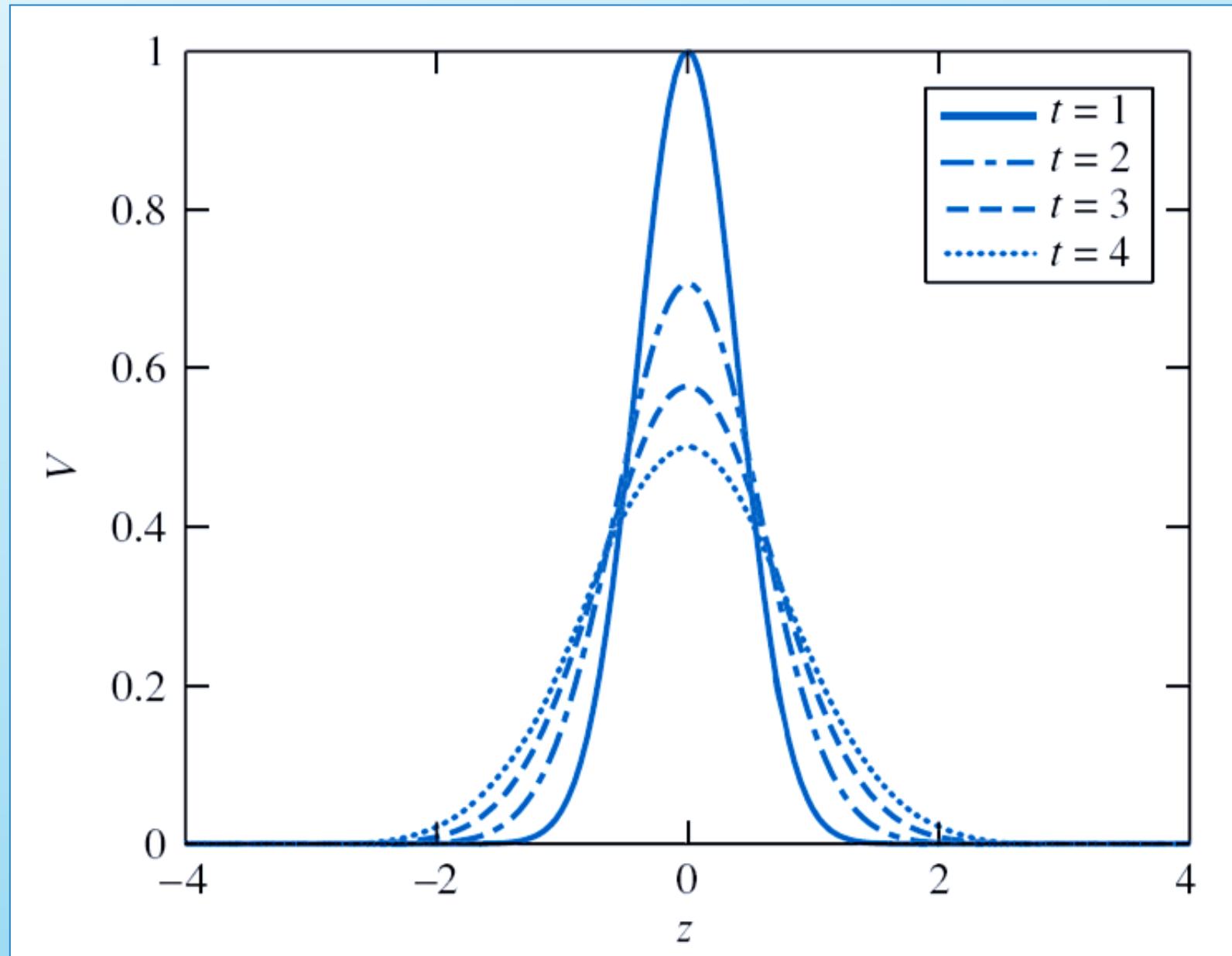
$$\frac{\partial^2 V(z,t)}{\partial z^2} = \hat{R} \frac{\partial I(z,t)}{\partial z} = \hat{R} \left(\hat{C} \frac{\partial V(z,t)}{\partial t} \right) = \hat{R} \hat{C} \frac{\partial V(z,t)}{\partial t}$$

$$D = \frac{1}{\hat{R} \hat{C}}$$

PROPAGAREA UNUI PULS DE TENSIUNE (GAUSS)

$$V(z,t) = \frac{1}{2\sqrt{D\pi t}} e^{-\frac{z^2}{4Dt}}$$

De-a lungul unei linii disipative,
un puls de tensiune se lateste.



Unde sinusoidale intr-o linie ideală

Apar daca la capatul liniei cuplam o sursa sinusoidală de tensiune

$$V(z,t) = \operatorname{Re} \left\{ V(z) e^{j\omega t} \right\}; \quad I(z,t) = \operatorname{Re} \left\{ I(z) e^{j\omega t} \right\}$$

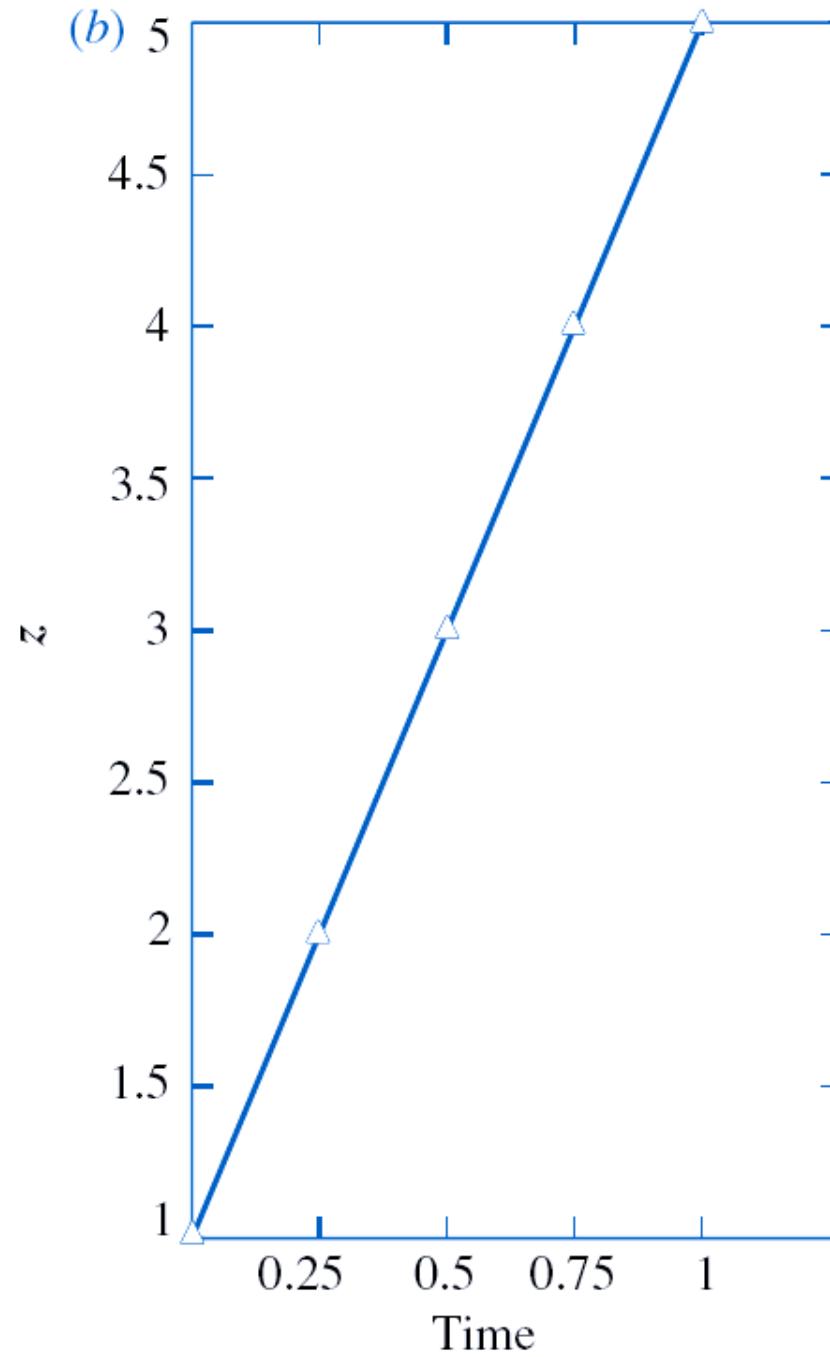
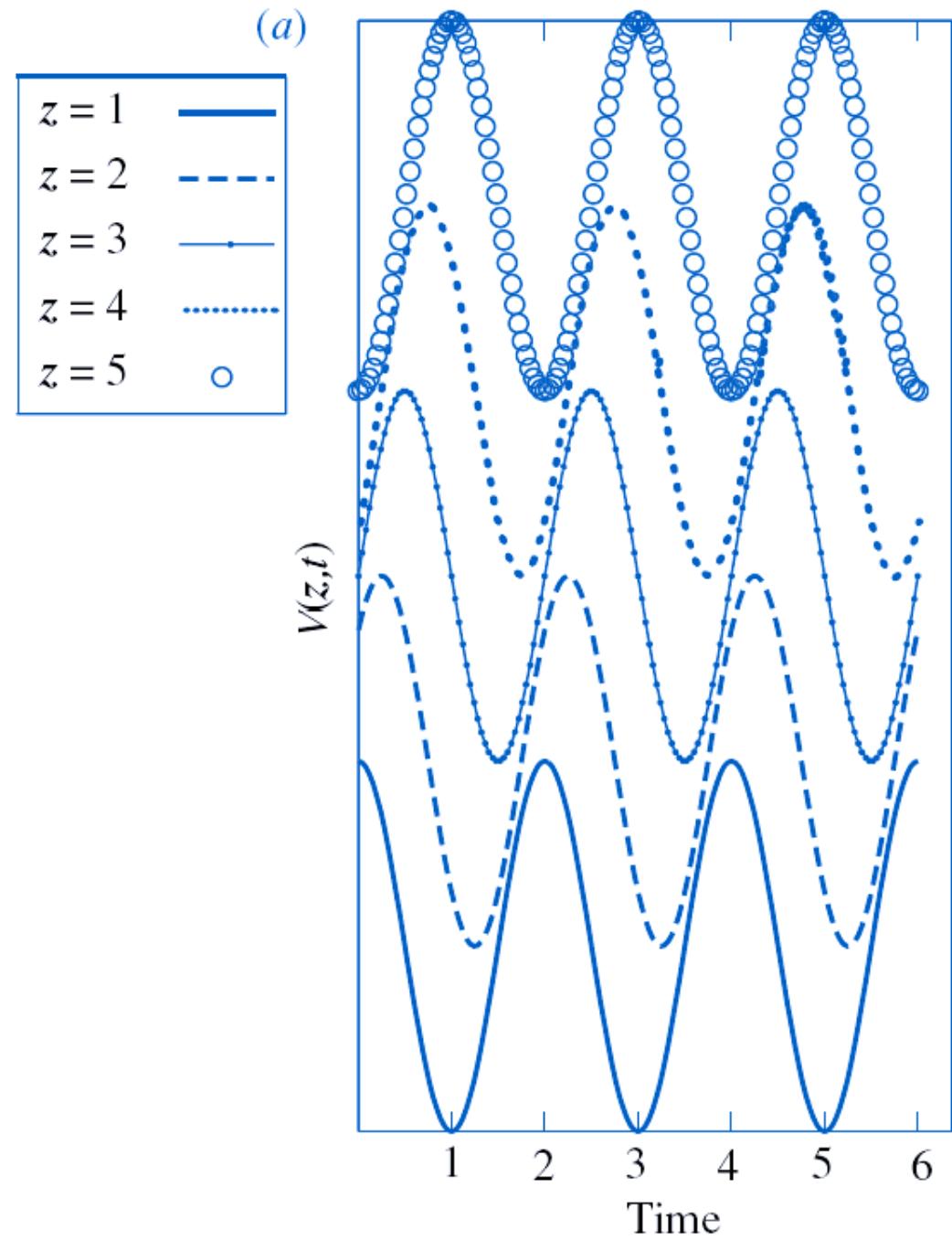
$$k = \frac{\omega}{v} = \frac{2\pi}{\lambda}$$

$$\frac{d^2 V(z)}{dz^2} + k^2 V(z) = 0$$

$$\frac{d^2 I(z)}{dz^2} + k^2 I(z) = 0$$

$$V(z) = A_1 \cos kz + B_1 \sin kz$$

$$V(z) = A_2 e^{-jkz} + B_2 e^{+jkz}$$



Stabilirea impedantei liniei

$$V(z)=V_0e^{-jkz}$$

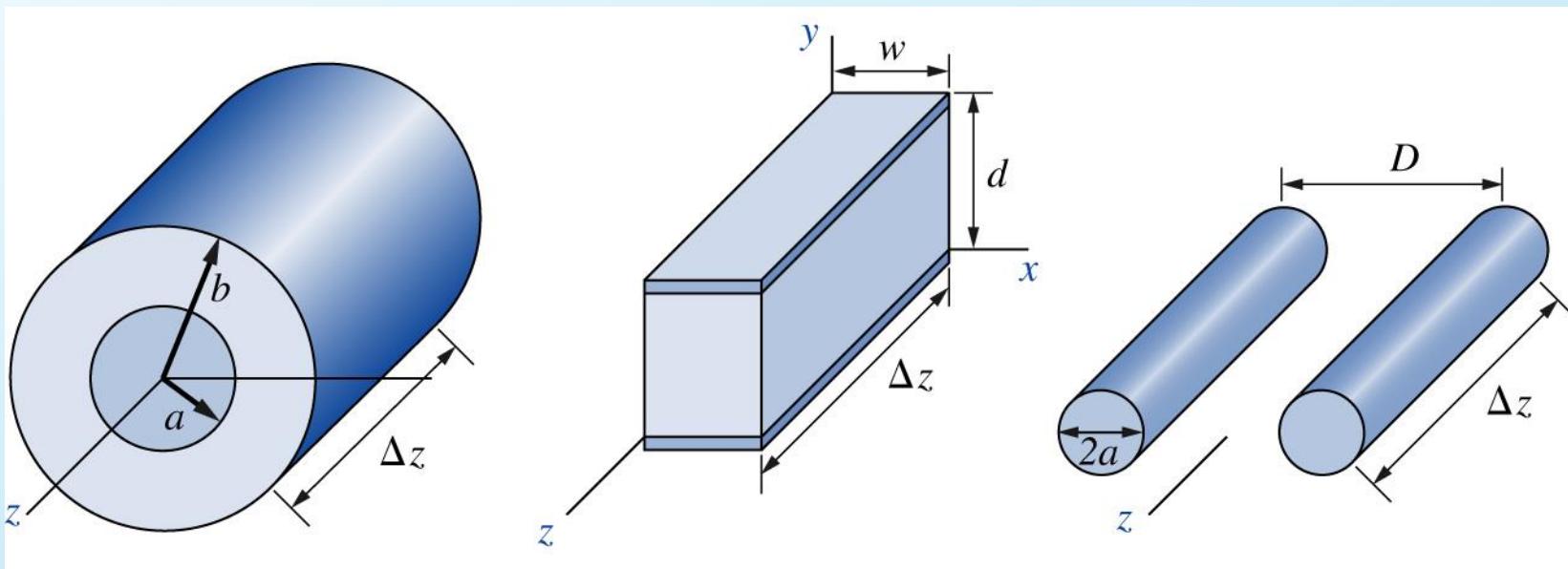
$$\frac{dV(z)}{dz}=-jkV(z)=-j\omega \hat{L}I(z)$$

$$I(z)=\frac{k}{\omega \hat{L}}V(z)=\frac{k}{\omega \hat{L}}V_0e^{-jkz}$$

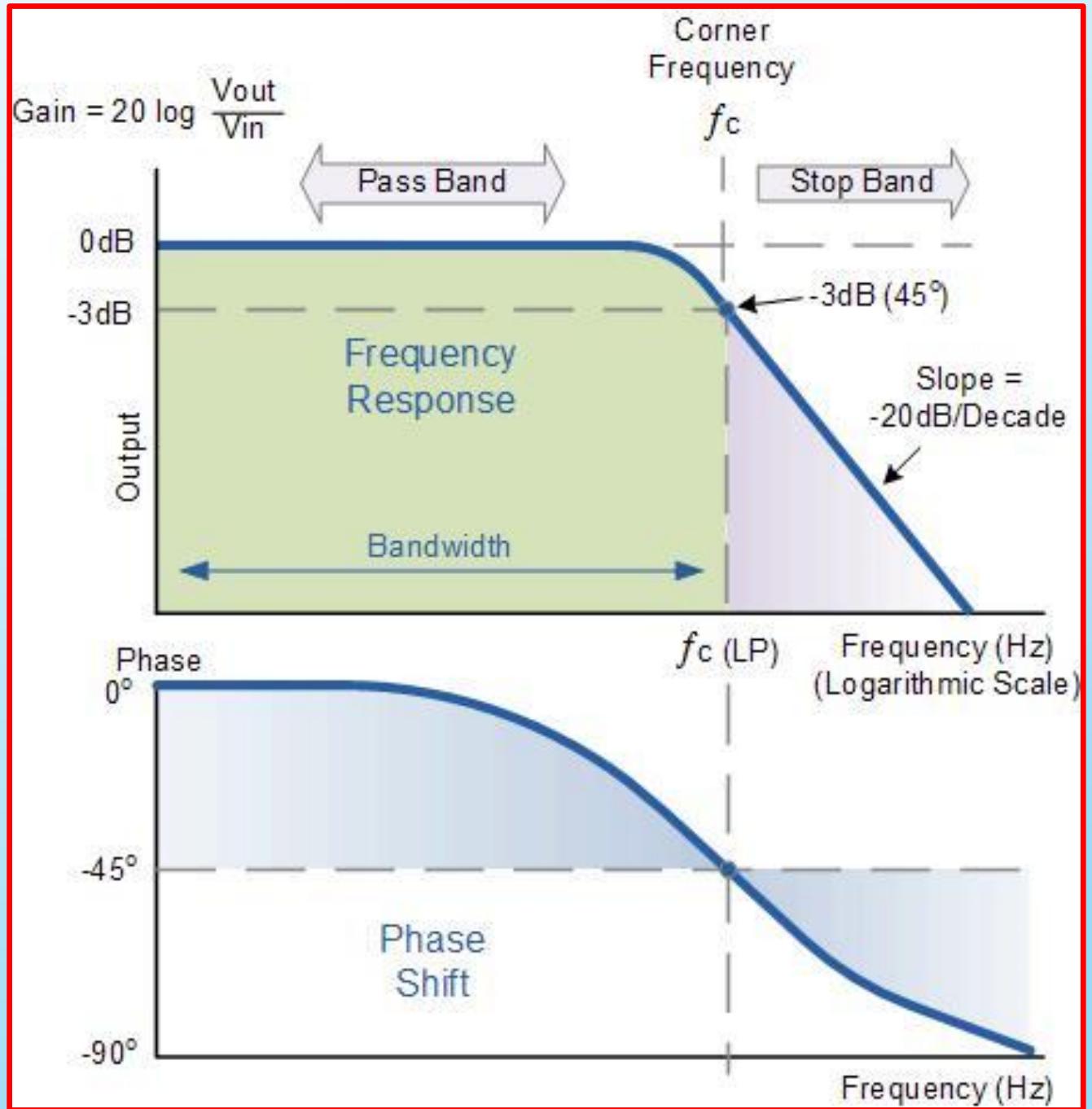
$$Z_c=\frac{V(z)}{I(z)}=\frac{\omega \hat{L}}{k}$$

$$k=\frac{\omega}{v}\qquad\qquad v=\frac{1}{\sqrt{\hat{L}\hat{C}}}$$

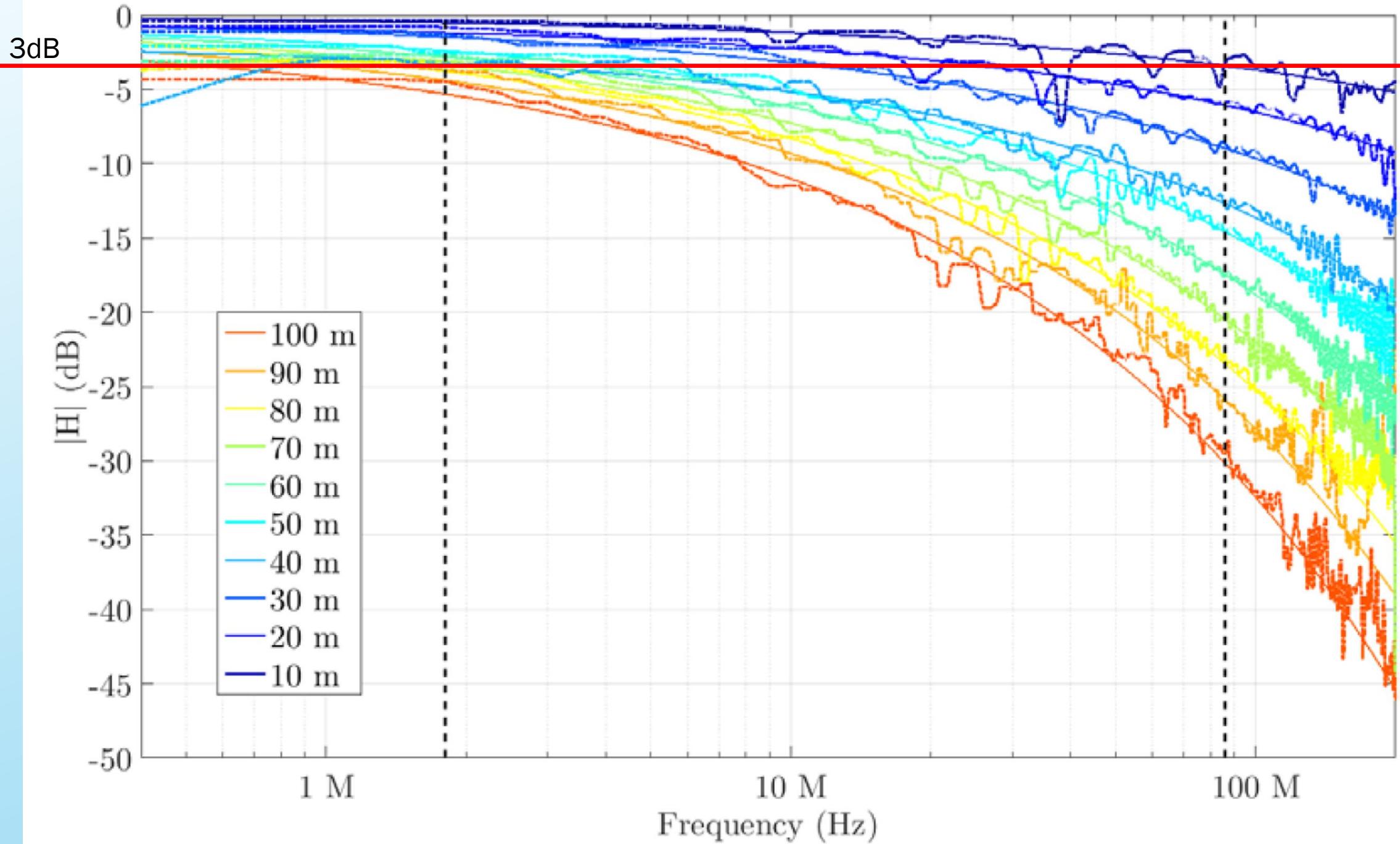
$$Z_c=\sqrt{\frac{\hat{L}}{\hat{C}}}\,\,[\Omega]$$



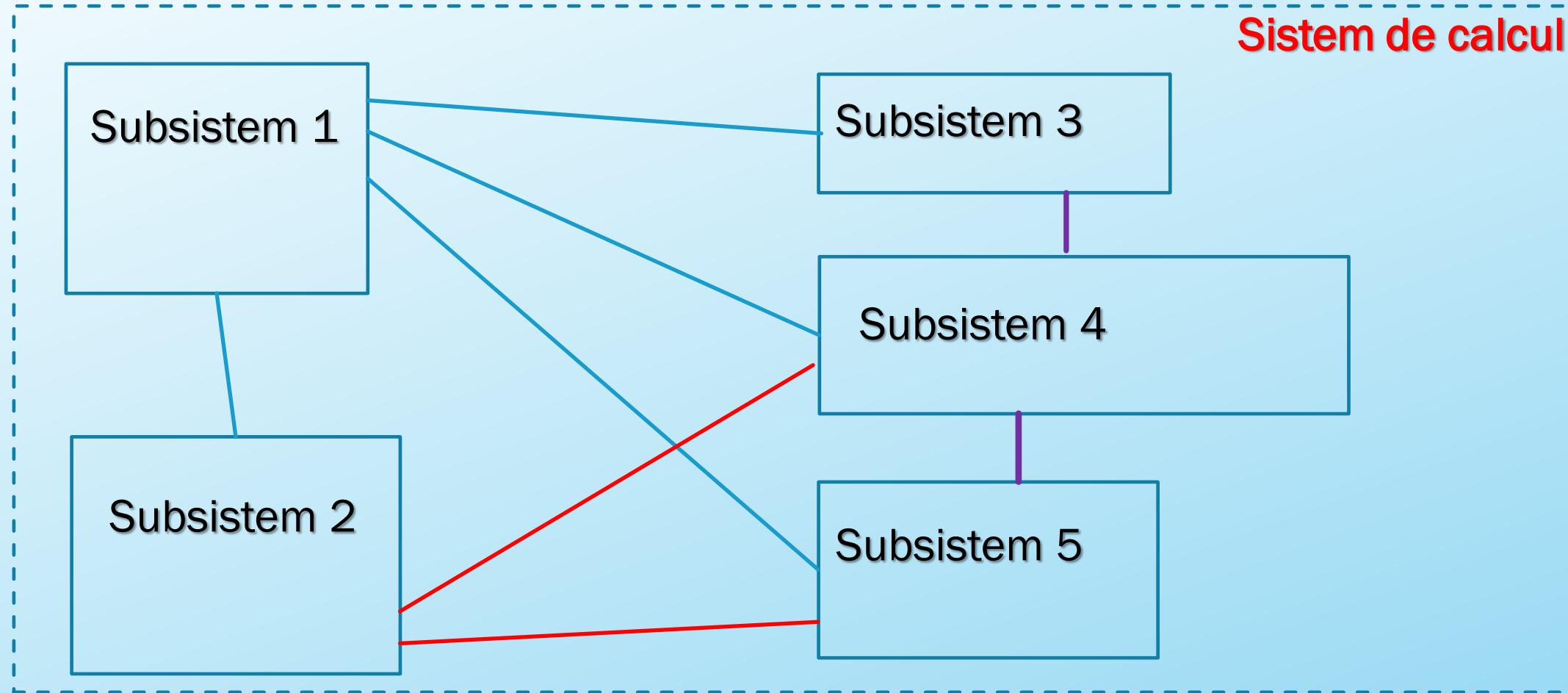
Line	Velocity of propagation	Characteristic impedance
Coaxial cable	$v = \frac{1}{\sqrt{\hat{L}\hat{C}}} = \frac{1}{\sqrt{\mu\varepsilon}}$	$Z_c = \sqrt{\frac{\hat{L}}{\hat{C}}} = \sqrt{\frac{\mu}{\varepsilon}} \left(\frac{\ln(b/a)}{2\pi} \right)$
Microstrip line	$v = \frac{1}{\sqrt{\hat{L}\hat{C}}} = \frac{1}{\sqrt{\mu\varepsilon}}$	$Z_c = \sqrt{\frac{\hat{L}}{\hat{C}}} = \sqrt{\frac{\mu}{\varepsilon}} \left(\frac{d}{w} \right)$
Twin lead	$v = \frac{1}{\sqrt{\hat{L}\hat{C}}} = \frac{1}{\sqrt{\mu\varepsilon}}$	$Z_c = \sqrt{\frac{\hat{L}}{\hat{C}}} = \sqrt{\frac{\mu}{\varepsilon}} \left(\frac{\cosh^{-1}(D/2a)}{\pi} \right)$



Raspunsul în frecvență al unei linii torsadate



Sistem de calcul



Subsistemele ce compun un sistem de calcul comunica prin intermediul liniilor de transmisie

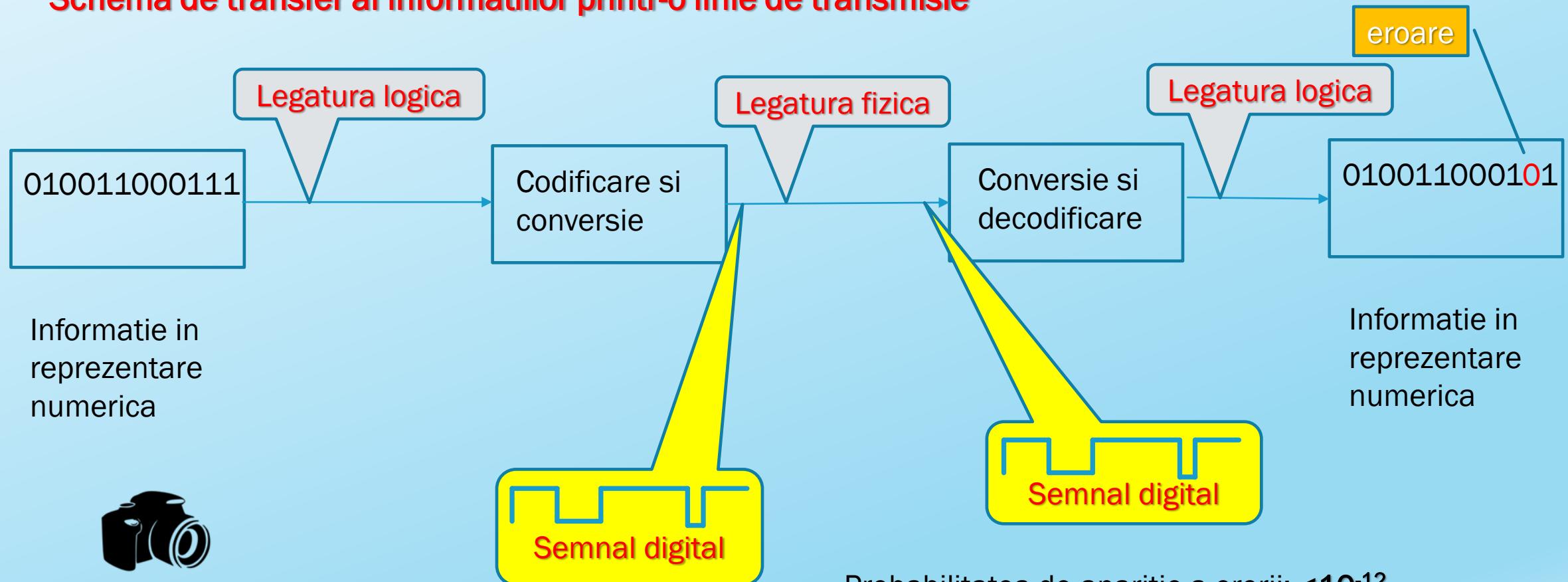
Subsistem: dispozitiv, unitate, bloc, modul, etc

In aceasta reprezentare liniile de transmisie au o largime de banda sufficient de mare incat sa nu franeze transmisia de date intre subsisteme

Caracteristici importante ale liniilor de transmisie

- Viteza finita de transmitere a semnalelor
- Transmitere distorsionata a semnalelor analogice
- Banda de frecvente finita pentru semnale analogice (existenta unei frecvente maxime)
- Capacitate limitata de transmitere a informatiei prin semnale digitale

Schema de transfer al informatiilor printr-o linie de transmisie



Caracteristici digitale ale liniilor de transmisie

Rata de transfer: numarul de biti transferati in unitatea de timp; unitatea de masura: bps

Largimea de banda: rata maxima de transfer; unitatea de masura: bps (cate o data se utilizeaza Hz)

Uzual se utilizeaza multiplii unitatilor de masura:

Kbps; kHz

Mbps; MHz

Gbps; GHz

Observatii:

Unitatea de masura Hz se foloseste numai pentru largimea de banda!

In unele situatii se foloseste si definitia:

Rata de transfer: numarul de bytes transferati in unitatea de timp; unitatea de masura: Bps

Exemple de transformare de unitati:

1 Bps = 8 bps

1 MBps = 8 MHz

O linie de transmisie poate fi formata din unul sau mai multe canale.

Pot fi canale fizice si/sau logice.

In cel de al doilea caz se utilizeaza frecvent denumirea de culoar (lanes).

Linia de transmisie cu mai multe canale fizice consta din mai multe fire (electrice) de-a lungul carora se transmite in paralel informatia (se transmit simultan biti diferiti pe fire diferite)

In cazul mai multor canale logice se partajeaza acelasi canal fizic intre mai multe canale logice (multiplexare)

Cabluri telefonice



Cablu PATA

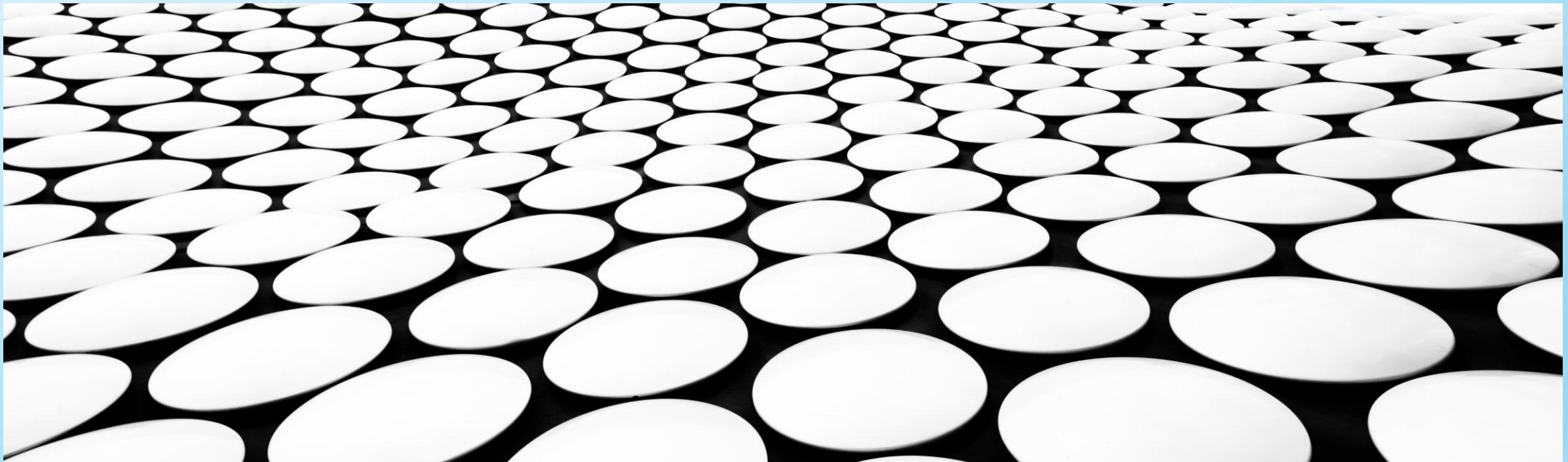
Daca o linie de comunicatie este formata din **N** canale fizice atunci
(Largimea de banda totala) < N x (largimea de banda a unui canal izolat).

Aceasta limitare apare datorita interferentelor intre canale.

Este una din explicatiile tranzitiei de la PATA la SATA.

ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



MAGISTRALE

O magistrala este un mediu de comunicatie ce leaga doua sau mai multe componente (subsisteme) ale unui sistem de calcul. Este un subsistem al sistemului de calcul.

Este un mediu de transmisie partajat:

La un moment dat numai o componenta (un modul) poate transmite informatie.

O magistrala poate contine una sau mai multe linii de transmisie fizica.

Daca magistrala contine mai multe linii comunicatia se face in paralel.

Semnalul fiind digital se transmit simultan (in paralel) mai multi biti.

Exemplu:

O magistrala de 8 biti transmite simultan un pachet de 8 biti, folosind 8 linii de transmisie.

Un sistem de calcul poate incorpora mai multe magistrale, formand o structura ierarhica a componentelor legate la magistrale.

■ Clase de magistrale:

- Magistrale de sistem
 - Conectaza CPU cu componentele de baza ale sistemului
 - ex.: MULTIBUS, ISA, PCI, PCIe
- Magistrale specializede
 - Optimizeaza transferul de date cu anumite tipuri de periferice
 - ex.: VESA, PATA(IDE), SATA, SCSI, USB, GPIB, GPIO, I2C

■ -

- Magistrale cu transmisie paralela
- Magistrale cu transmisie seriala

- Magistrale sincrone
- Magistrale asincrone

O magistrala contine:

- un set de linii de date (magistrala de date)
- un set de linii de adrese (magistrala de adrese)
- un set de linii de control

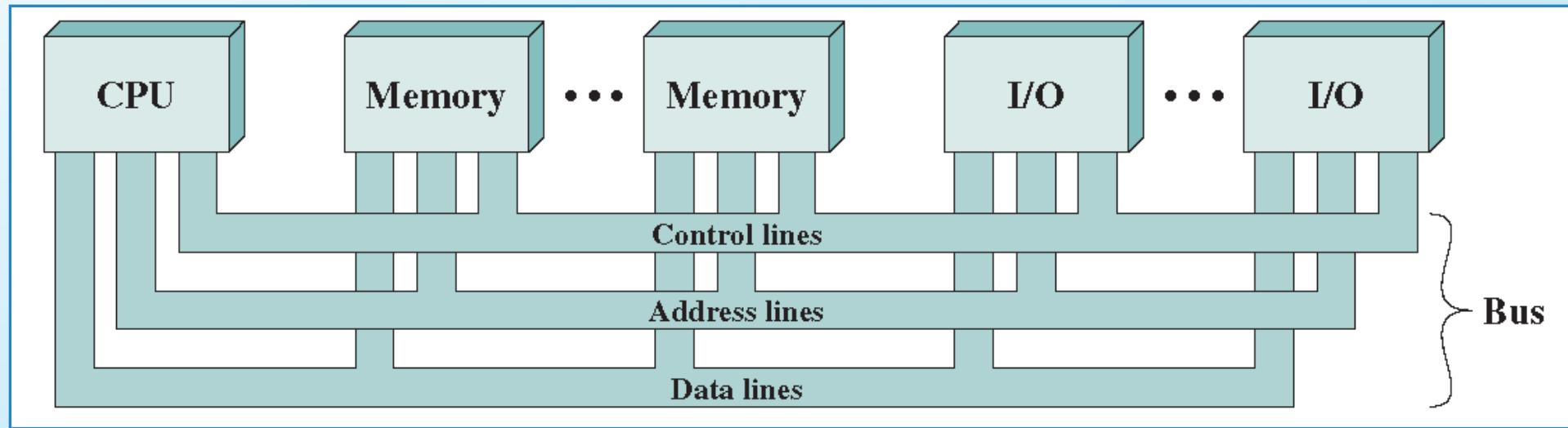
Semnalele de control transmit informatii de comanda si informatii de timing

Semnalele de comanda specifica operatiunile ce trebuie efectuate

Semnalele de timing indica validitatea informatiilor de date si adrese

OBSERVATIE

Liniile sunt fizice sau logice.



CPU = Central Processing Unit (Unitatea Centrală de Prelucrare)

Bus = Magistrala

I/O = bloc de intrare / ieșire



Magistrala functioneaza astfel:

- Dacă un modul dorește să trimită date către un altul, atunci el trebuie să facă două lucruri:
 - (1) să obțină dreptul de utilizare a magistralei și
 - (2) să transfere date prin intermediul magistralei.
- Dacă un modul dorește să solicite date de la un alt modul, atunci el trebuie:
 - (1) să obțina dreptul de utilizare a magistralei și
 - (2) să transfere o cerere către celălalt modul prin intermediul liniilor de control și adrese adecvate.
 - (3) să aștepte ca modulul al doilea să trimită datele.

Observatie: în aceasta descriere modul = CPU, memorie, modul I/O .

In raport cu majoritatea comenzilor toate modulele au aceleasi drepturi.

Există un număr limitat de comenzi care pot fi inițiate numai de CPU.

Unele magistrale au un bloc special, cuplat la magistrala, numit controler de magistrala, care gestionează drepturile de utilizare. În caz contrar gestionarea este realizată de CPU.

Exemple de comenzi

- **Memory write:** causes data on the bus to be written into the addressed location
- **Memory read:** causes data from the addressed location to be placed on the bus
- **I/O write:** causes data on the bus to be output to the addressed I/O port
- **I/O read:** causes data from the addressed I/O port to be placed on the bus
- **Transfer ACK:** indicates that data have been accepted from or placed on the bus
- **Bus request:** indicates that a module needs to gain control of the bus
- **Bus grant:** indicates that a requesting module has been granted control of the bus
- **Interrupt request:** indicates that an interrupt is pending
- **Interrupt ACK:** acknowledges that the pending interrupt has been recognized
- **Clock:** is used to synchronize operations
- **Reset:** initializes all modules.

- **Memory write**: face ca datele din magistrală să fie scrise în locația adresată
- **Memory read**: face ca datele din locația adresată să fie plasate pe magistrala
- **I/O write**: determină ca datele de pe magistrală să fie trimise la portul **I/O** adresat
- **I/O read**: face ca datele din portul **I/O** adresat să fie introduse pe magistrala
- **Transfer ACK**: indică faptul că datele au fost acceptate dinspre, sau introduse pe magistrala
- **Bus request**: indică faptul că un modul trebuie să obțină controlul asupra magistralei
- **Bus grant**: indică faptul că un modul solicitant a primit controlul asupra magistralei
- **Interrupt request**: indică faptul că o solicitare de întrerupere este în aşteptare
- **Interrupt ACK**: admite ca solicitarea de întrerupere în aşteptare a fost recunoscută
- **Clock**: se utilizează pentru sincronizarea operațiilor
- **Reset**: initializează toate modulele.

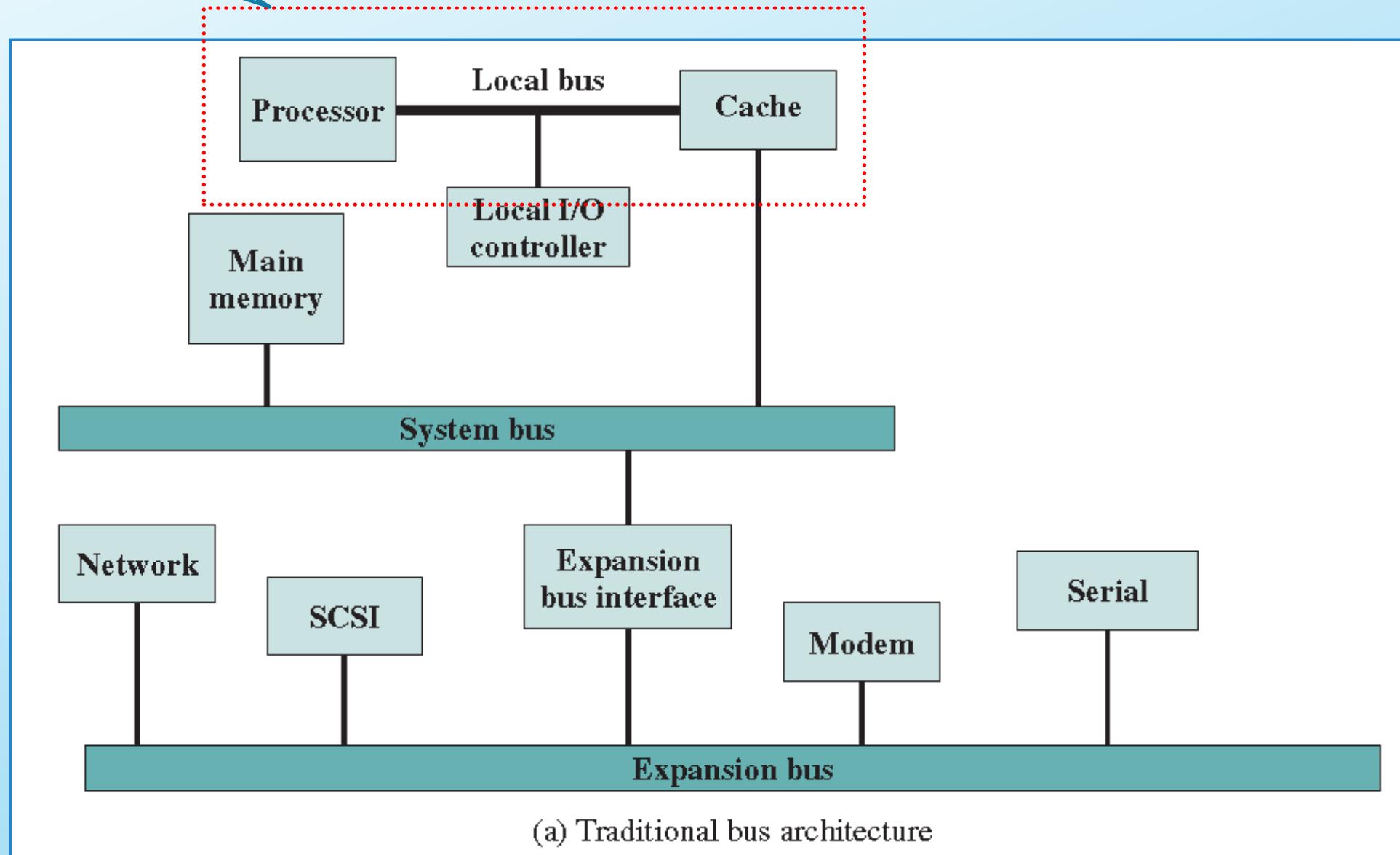
ARHITECTURI MULTI-MAGISTRALA

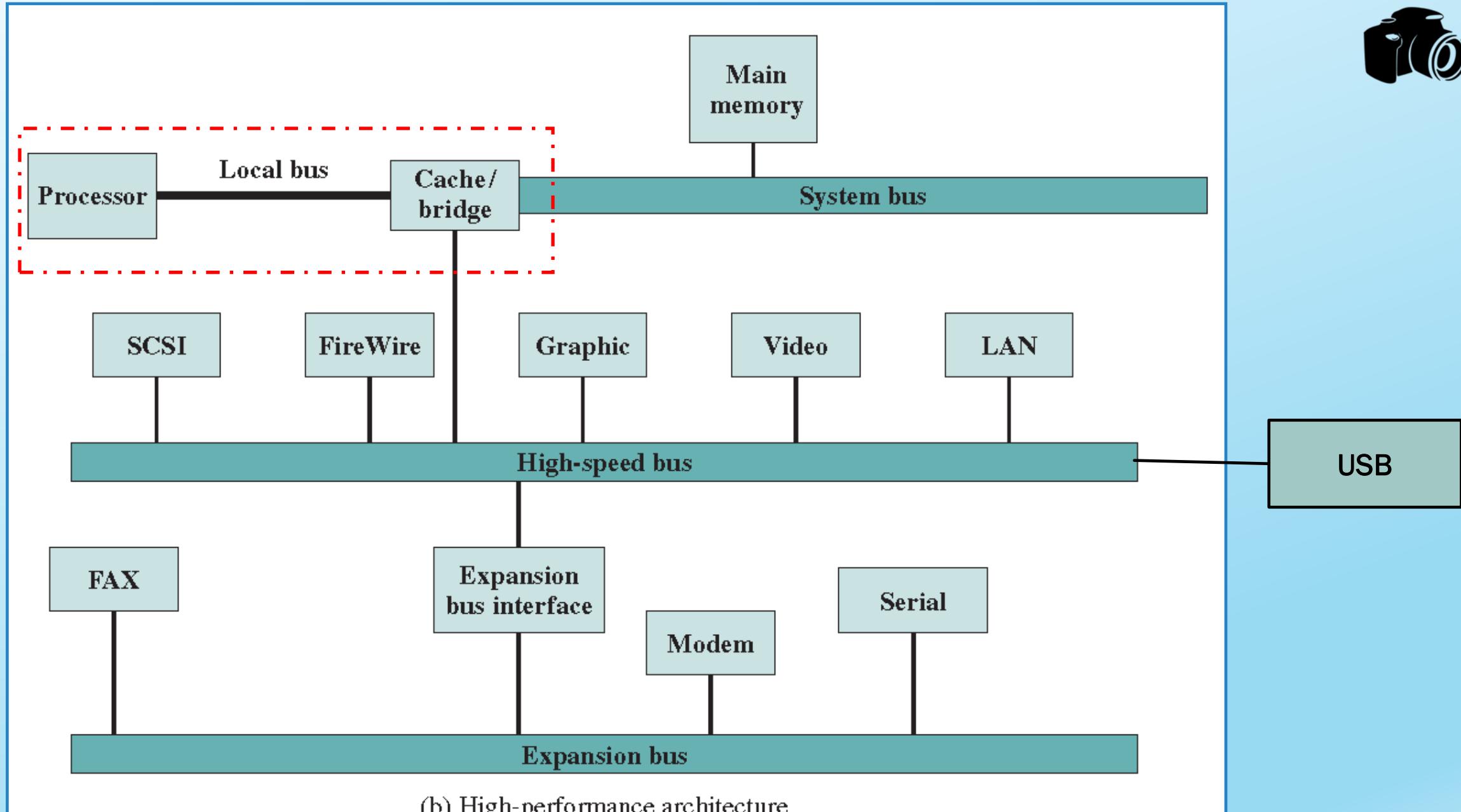
- Dacă sunt conectate multe dispozitive la magistrala, performanța se va înrauăti.
- Există două cauze principale:
 - 1. Cu cât sunt mai multe dispozitive atașate la magistrala, cu atât este mai mare lungimea magistralei și deci este mai mare **întârzierea propagării**. Această întârziere determină timpul necesar pentru ca dispozitivele să-si coordoneze utilizarea magistralei. Atunci când controlul magistralei trece de la un dispozitiv la altul, apare o **întârziere de comutare**. Dacă aceste întârzieri sunt frecvente ele pot afecta semnificativ performanța.
 - 2. Magistrala poate gătui transferul, deoarece cererea de transfer de date aggregate atinge **capacitatea magistralei**. Această problemă poate fi contracarată într-o oarecare măsură prin:
 - **creșterea ratei de transfer pe care magistrala o poate transporta** și prin
 - **utilizarea unor magistrale mai largi (de exemplu, creșterea magistralei de date de la 32 la 64 de biți)**.
- Amană două problemele sunt evitate prin utilizarea mai multor magistrale interconectate (ierarhizate)



Exemple de arhitecturi de magistrale interconectate

micro
procesor





Magistrala memoriei principale

Este o magistrala de sistem
Toate liniile sunt linii fizice

Liniile magistralei sunt impartite in trei seturi de linii:

- Linii de date
- Linii de adrese
- Linii de comanda si control

Setul de linii de date (magistrala de date)

- Tipic: 32, 64 sau 128 linii
- Numarul de linii poarta denumirea de: **latimea magistralei**
- Latimea magistralei este un factor determinant al performantei globale a unui sistem de calcul

Daca o instructiune are lungimea de 64 biti, atunci CPU trebuie sa acceseze memoria de doua ori pentru a citi o instructiune, pentru o latime de magistrala de 32 biti.

Liniile de adrese (magistrala de adrese)

Latimea magistralei de adrese determina capacitatea maxima de memorie a unui sistem

O magistrala de adrese cu largimea de 8 biti se poate adresa unei memorii cu maxim 256 de locatii

Daca largimea este de 16 biti atunci numarul maxim de locatii este 65 536 (65 mii)

Pentru 32 biti avem 4 294 967 296 locatii (4 miliarde)

Pentru 64 biti 18 446 744 073 709 551 616 (18 miliarde de miliarde)

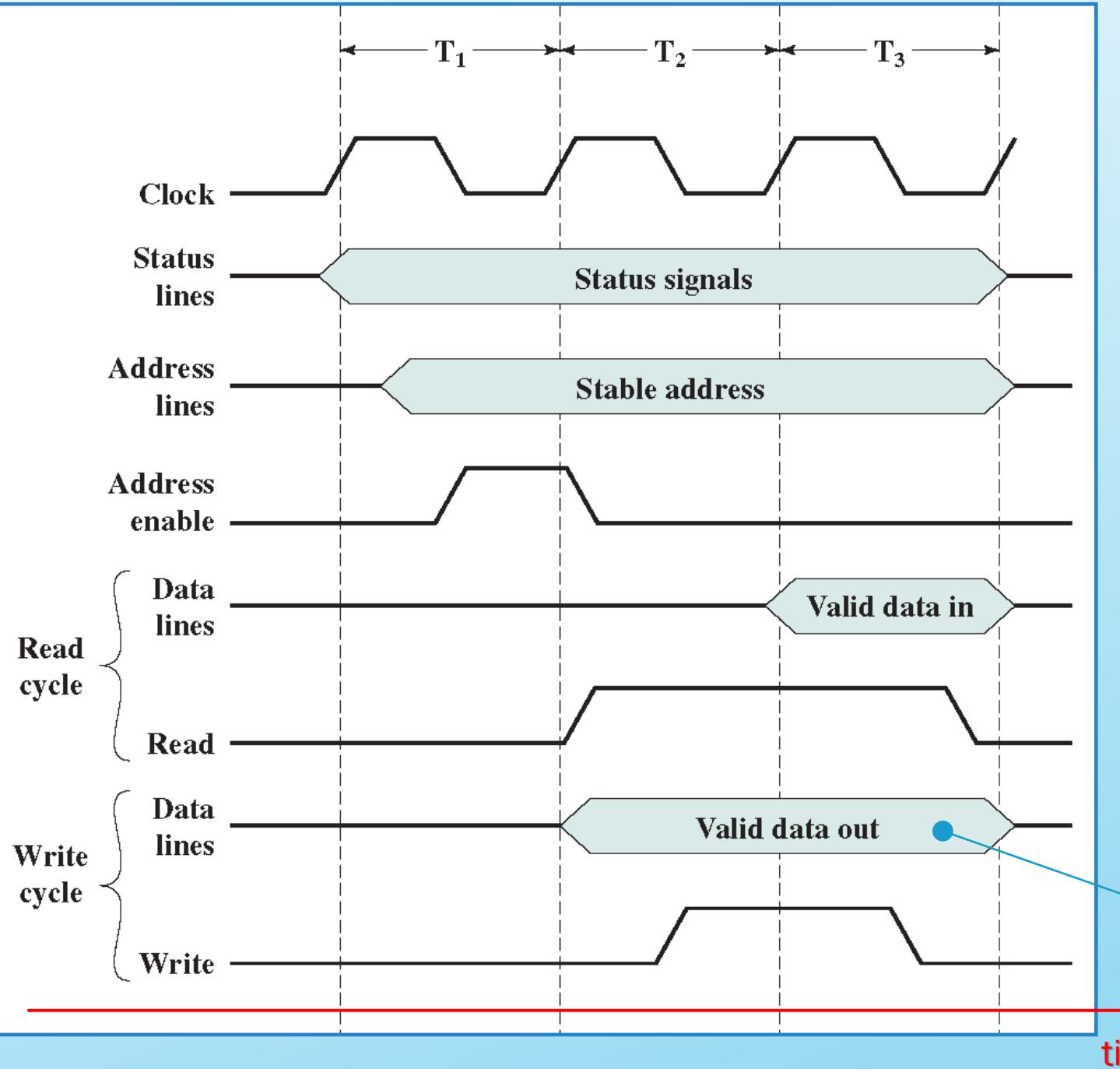
Exista doua tipuri de magistrale de memorie principala:

Magistrala sincrona

Magistrala asincrona



Magistrala sincronă



Clock: semnal dreptunghiular generat de ceasul magistralei, si trimis pe o linie dedicata.

Toate evenimentele starteaza la inceputul ciclului de ceas.

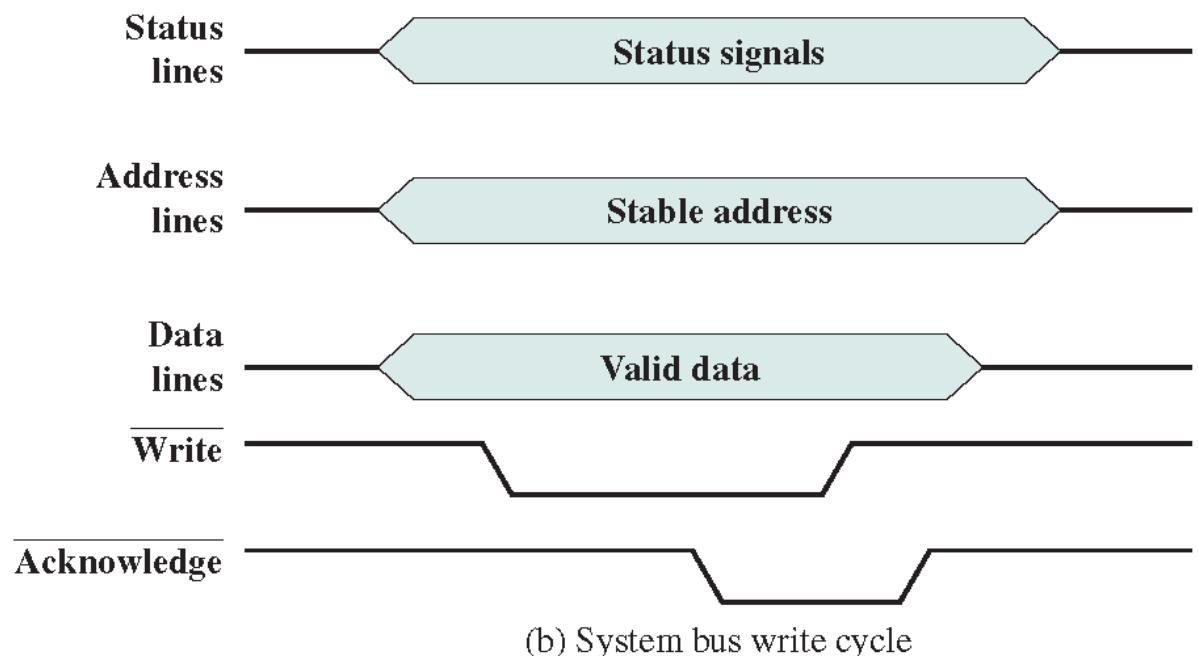
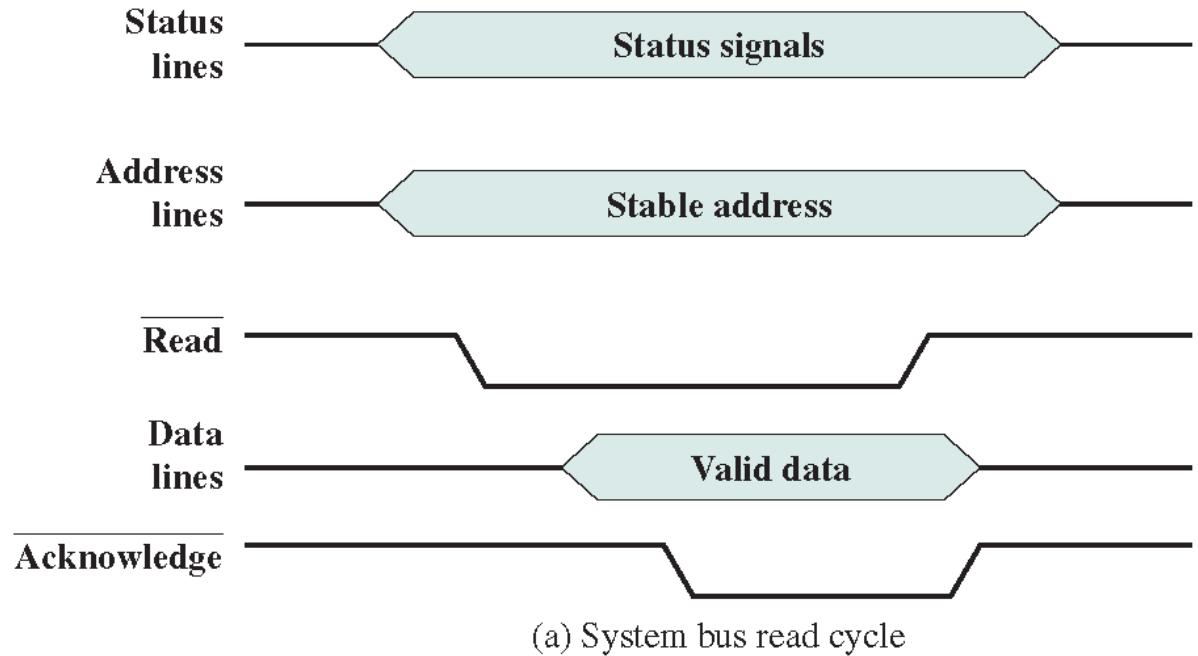
Address enable: semnal de activare a adresei

Read: semnalul de comanda a citirii

Write: semnal de comanda a scrierii:

Pachet de semnale
timpul

Magistrala asincrona



Preluarea controlului nu este indicata in diagrama

In general semnalele de stare indica disponibilitatea memoriei pentru operatiuni

Acknowledge: semnal de confirmare

Pentru citirea memoriei:

In prima etapa sunt trimise semnalele de stare si adresa

In etapa a doua este trimisa comanda de citire

In etapa a treia apar datele valide pe liniile de date

In etapa a patra memoria confirma finalizarea operatiunii

Pentru scrierea in memorie:

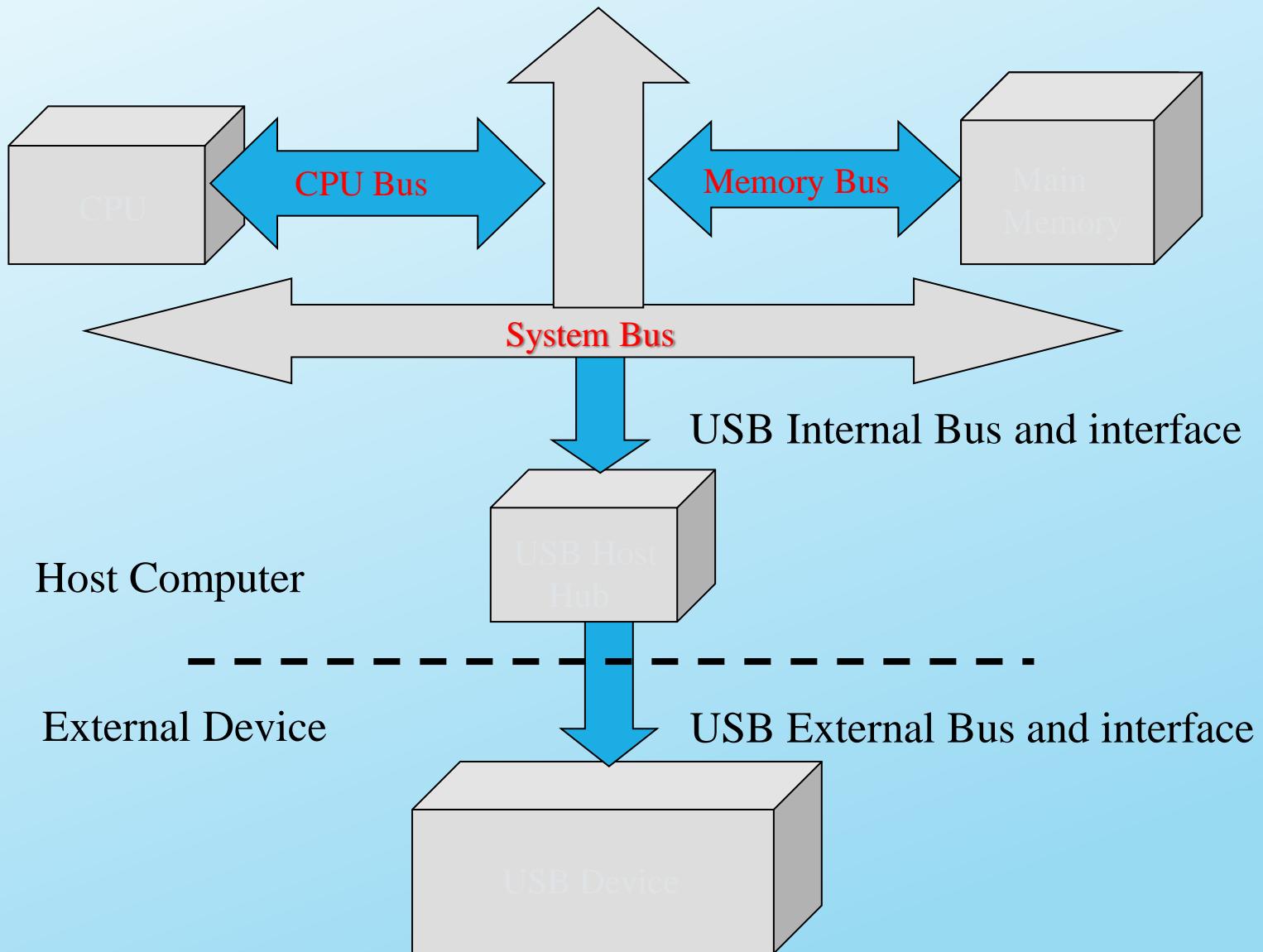
In prima etapa sunt trimise semnalele de stare si adresa; de asemenea sunt trimise datele destinate scrierii pe liniile de date

In etapa a doua este trimisa comanda de scriere

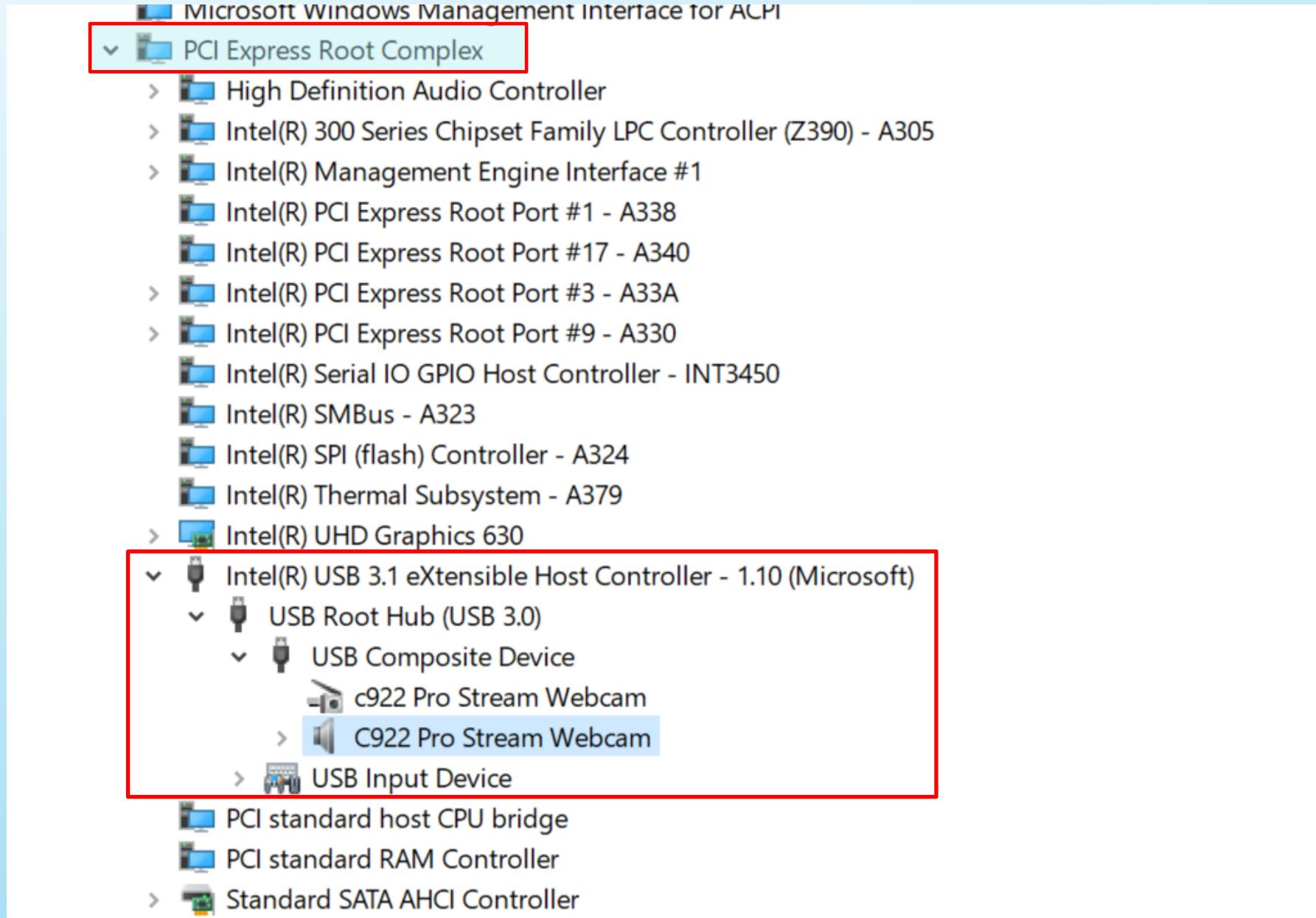
In etapa a treia memoria confirma finalizarea operatiunii

Magistrale USB

Ierarhia magistralelor

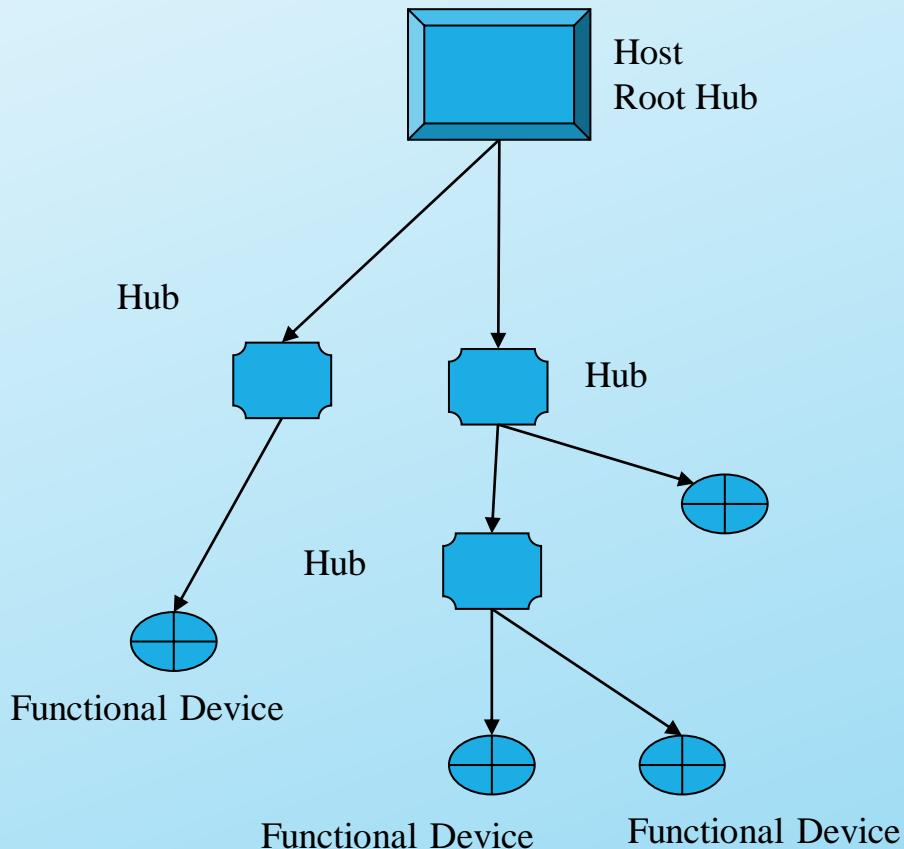


Ierarhia magistralelor poate fi parțial observată în Device Manager



Topologia magistralei

- Conectează computerul la dispozitive periferice.
- **Topologie stelară pe mai multe niveluri**



- Toate dispozitivele sunt conectate la un punct comun denumit hub rădăcină.
- Specificația permite până la 127 ($2^7 - 1$) dispozitive diferite.
- Cablul cu patru fire servește la interconectare: alimentare, împământare și două linii pentru semnal diferențial.
- USB este o magistrală cu interogare; toate tranzacțiile sunt inițiate de gazdă.

CARACTERISTICI GENERALE ALE MAGISTRALEI USB

■ Ușor de utilizat pentru utilizatorul final

- Model unic pentru cablare și conectori
- Contactele electrice izolate de utilizatorul final (de exemplu, terminalele de magistrală)
- Periferice autoidentificate, mapare automată a funcțiilor și a configurației
- Periferice atașabile dinamic și reconfigurabile

■ O gamă largă de sarcini de lucru și aplicații

- Potrivită pentru lățimi de bandă ale dispozitivelor, de la câțiva kb/s la câțiva Mb/s (și chiar mai mult)
- Suportă tipuri de transfer isocron și asincron pe același set de fire
- Suportă funcționarea simultană a mai multor dispozitive (conexiuni multiple)
- Suportă transferul de fluxuri multiple de date și de mesaje între gazdă și dispozitivele atașate

■ Implementare low-cost

- Sub-canal cu cost redus la 1,5 Mb/s
- Potrivit pentru dezvoltarea de periferice low-cost
- Cabluri și conectori ieftini

■ Calea de upgrade

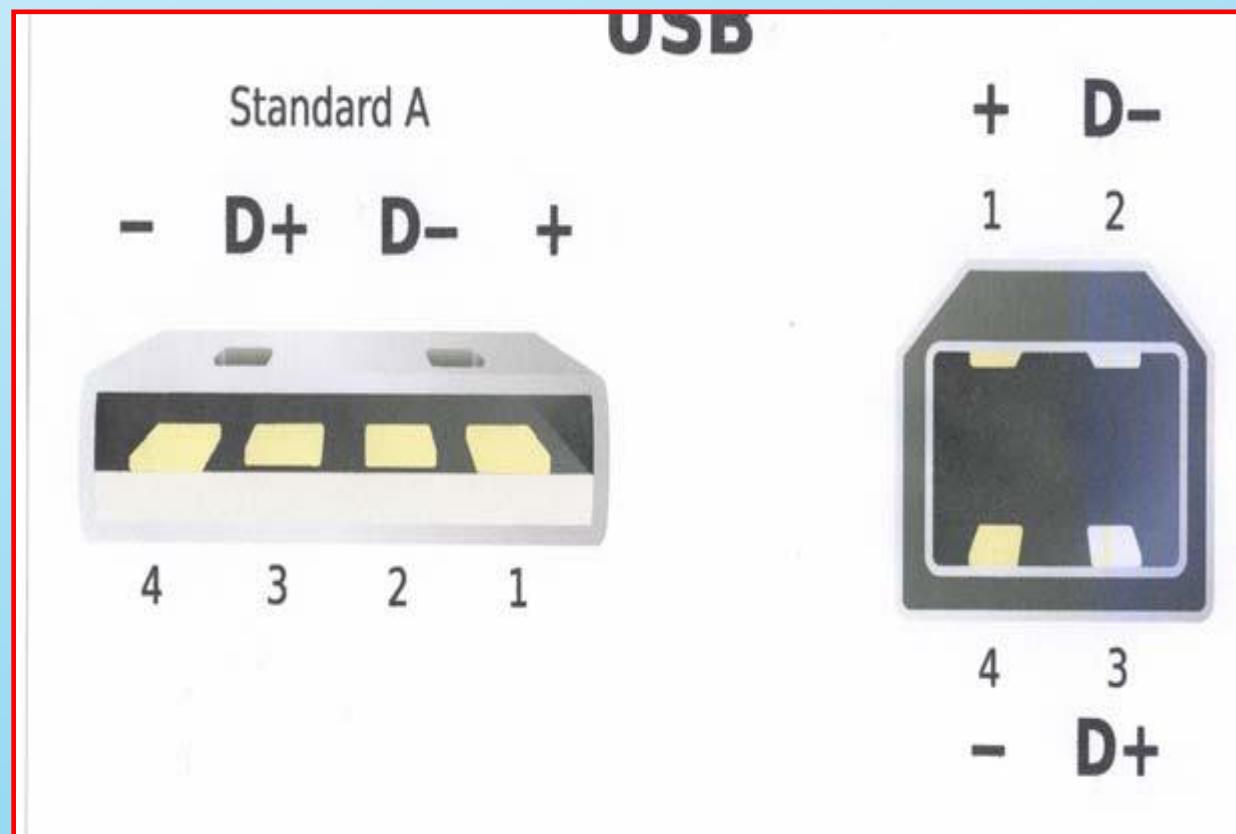
- Arhitectură upgradabilă pentru a suporta mai multe controlere USB de gazdă într-un sistem

USB

- Rapida
- Bidirecțională
- Izocrona
- cost scăzut
- interfață serială atașabilă dinamic
- Este în concordanță cu cerințele platformei PC de astăzi și de mâine

USB 2

- Patru fire (+5V, Return, data twisted pair)
- Până la 5 m lungime
- Conexiunile mai lungi folosesc hub-uri sau extensii active



USB

- Specificația USB 1.0 introdusă în 1994
- Specificația USB 2.0 a fost finalizată în 2001
- A devenit populară datorită avantajului cost / beneficiu
 - Spre deosebire, IEEE 1394 - lățime de bandă mare dar cost ridicat
- Trei generații de USB
 - USB 1 : Full-speed devices (12 Mbps)
 - USB 2 : maxim 480 Mbps
 - USB 3 : 6Gbps și WUSB
 - USB 4 : 40Gbps

CARACTERISTICI ALE MAGISTRALEI USB (CONTINUARE)

■ Lățimea de bandă izocronă

- Lățime de bandă garantată și latențe reduse, adecvate pentru telefonie, aplicatii audio etc.
- Sarcina de lucru izocronă poate folosi întreaga lățime de bandă a magistralei

■ Flexibilitatea

- Suportă o gamă largă de dimensiuni de pachete, care permite o gamă de opțiuni pentru memoria tampon a dispozitivului
- Permite o gamă largă de rate de transfer de date prin adaptarea dimensiunii memoriei tampon și latenței pachetelor
- Controlul debitului pentru manipularea tamponului este integrat în protocol

■ Robustete

- Tratarea erorilor / mecanismul de recuperare a informatiei este integrat în protocol
- Inserarea și îndepartarea dinamică a dispozitivelor se desfășoară în timp real
- Asigura identificarea dispozitivelor defecte

COMPARAȚIE

5 m = 16.4 ft

Interface	Format	Number of Devices (maximum)	Length (maximum, feet)	Speed (maximum, bits/sec.)	Typical Use
USB	asynchronous serial	127	16 (or up to 96 ft. with 5 hubs)	1.5M, 12M, 480M	Mouse, keyboard, disk drive, modem, Audio, printer, scanners, etc
RS-232 (EIA/TIA-232)	asynchronous serial	2	50-100	20k (115k with some hardware)	Modem, mouse, instrumentation
Parallel Printer Port	parallel	2 (8 with daisy-chain support)	10–30	8M	Printers, scanners, disk drives

BENEFICII PENTRU UTILIZATORI

■ Ușurință de utilizare

Ușurința de utilizare a fost un obiectiv major de proiectare pentru USB, iar rezultatul este o interfață plăcută la utilizare din mai multe motive:

■ O singura interfață pentru mai multe dispozitive.

USB-ul este suficient de versatil pentru a putea fi utilizat cu multe tipuri de periferice. În loc să aibă tipuri diferite de conectori și support hardware diferit pentru fiecare periferic, o singura interfață servește multora.

■ Configurare automată.

Când un utilizator conectează un periferic USB la un computer, sistemul său de operare detectează automat perifericul și încarcă driverul software corespunzător.

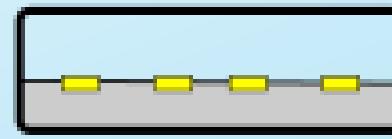
■ Conectabil la cald

Putem conecta și deconecta un periferic oricând dorîți, indiferent dacă sistemul și perifericul sunt alimentate sau nu, fără a deteriora computerul sau perifericul. Sistemul de operare detectează când este atașat un dispozitiv și îl pregătește pentru utilizare.

■ Nu este necesară o sursă de alimentare (uneori).

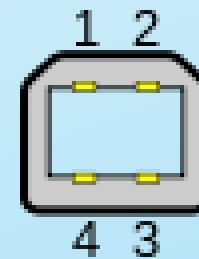
Un periferic care necesită până la 500 de miliamperi poate extrage toată puterea sa din magistrala în loc să aibă propria sursă de alimentare.

ASTECUL CONECTORILOR USB 2



4 3 2 1

Type A



1 2
4 3

Type B



5 4 3 2 1

Mini-A



5 4 3 2 1

Mini-B



1 2 3 4 5

Micro-A



1 2 3 4 5

Micro-B

- Conectori de tip A pe dispozitivele gazdă care furnizează energie
- Conectori de tip B pe dispozitivele țintă care primesc alimentare.

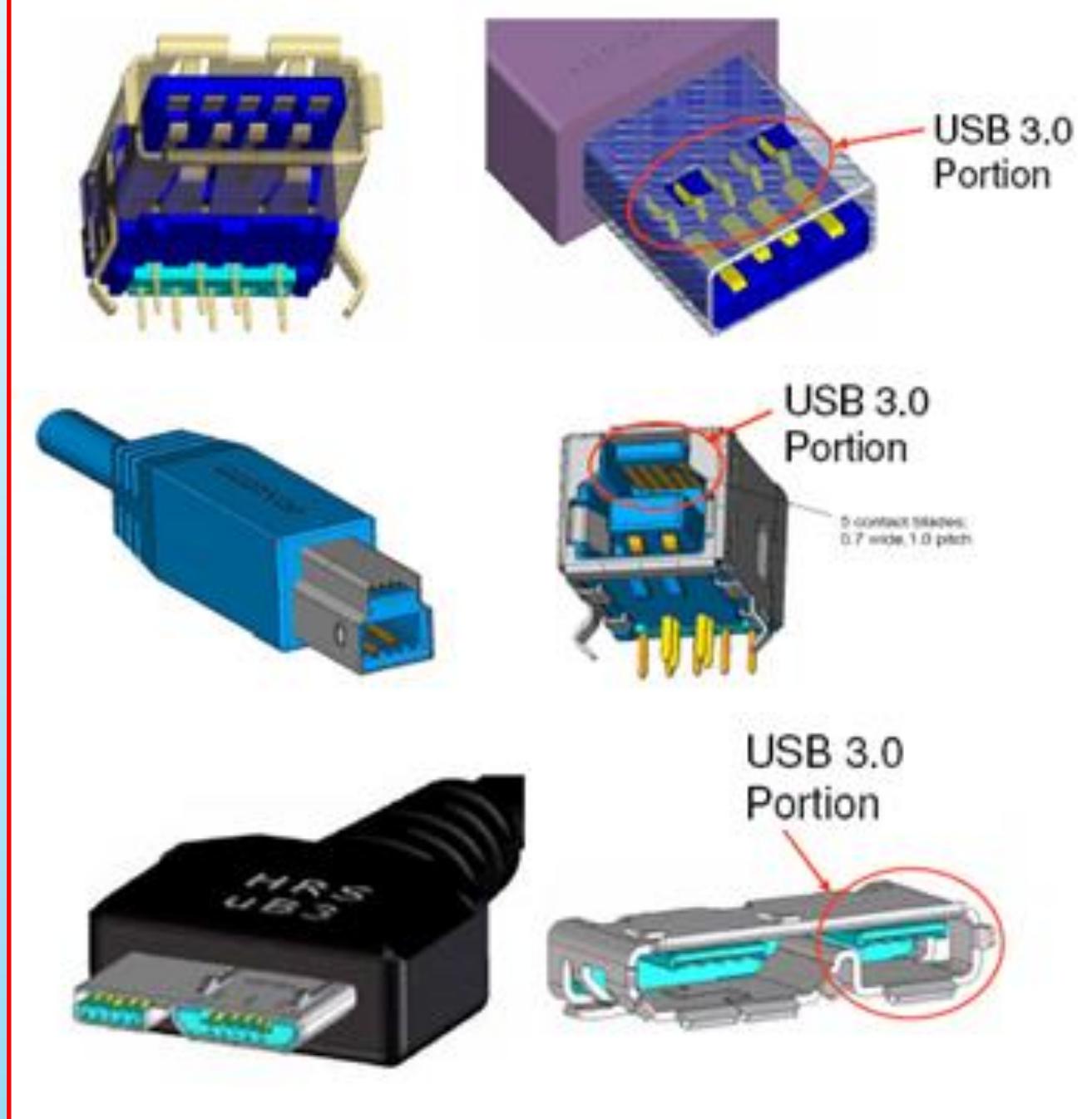
USB 3.0

- Denumit și SuperSpeed USB
- Viteză de 10 ori mai mare decât 2.0 (5 Gbps în mediu de testare controlat)
 - Transfer de fișier de 25 GB în aproximativ 70 de secunde (vezi tabelul)
- Extensibil - Proiectat pentru scalare > 25Gbps
- Eficiență energetică optimizată
 - Fără interogare dispozitiv (notificări asincrone)
 - Cerințe mai mici de putere activă și de mers în gol
- Compatibil cu USB 2.0
 - Dispozitivul USB 2.0 va funcționa cu gazda USB 3.0
 - Dispozitivul USB 3.0 va funcționa cu gazda USB 2.0

	Song / Pic	256 Flash	USB Flash	SD-Movie	USB Flash	HD-Movie
	4 MB	256 MB	1 GB	6 GB	16 GB	25 GB
USB 1.0	5.3 sec	5.7 min	22 min	2.2 hr	5.9 hr	9.3 hr
USB 2.0	0.1 sec	8.5 sec	33 sec	3.3 min	8.9 min	13.9 min
USB 3.0	0.01 sec	0.8 sec	3.3 sec	20 sec	53.3 sec	70 sec

CONECTORI USB 3.0

- S-au adăugat pini pentru semnalele USB SuperSpeed
- Compatibilitate pentru conectorii USB 2.0
- Conectorul USB 3.0 Standard B (mijloc) conține pinii de alimentare și de împământare pentru dispozitivul de alimentare

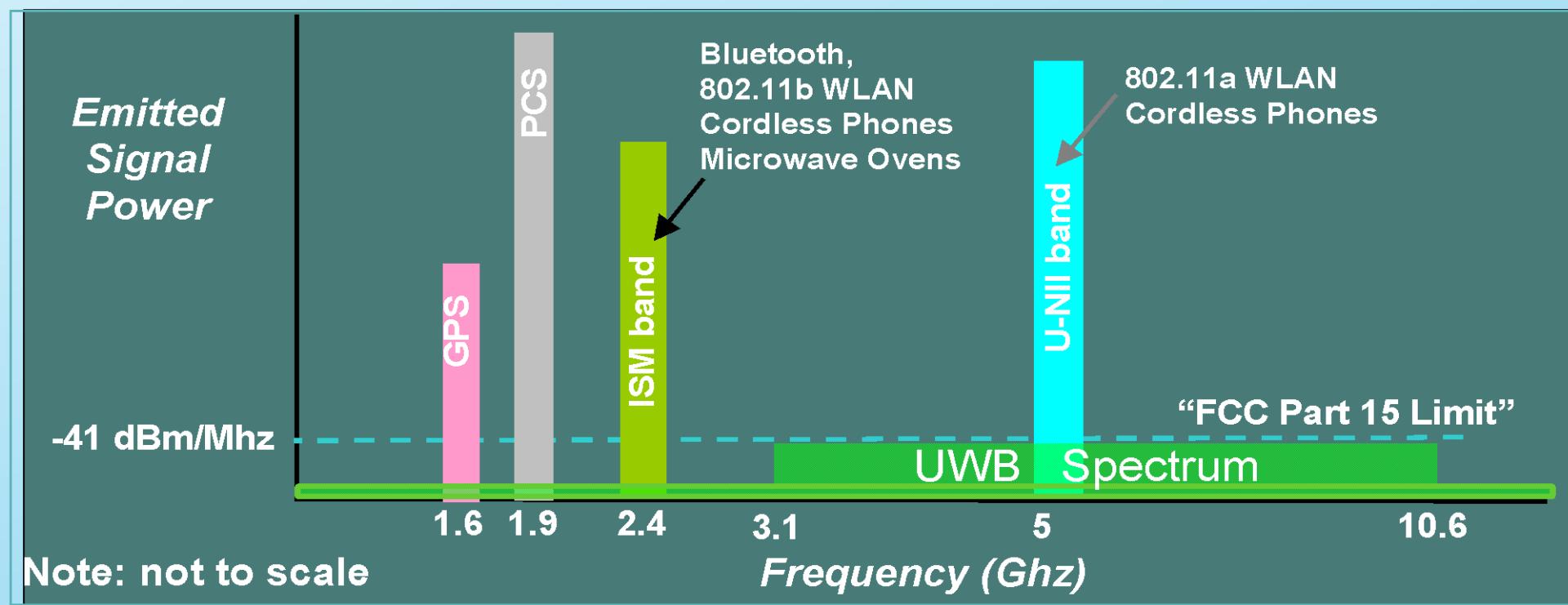


WIRELESS USB

- WUSB este o formă de tehnologie USB care utilizează undele radio (RF).
- Tehnologia WUSB se bazează pe platforma radio comună WiMedia Ultra-Wideband.
- WUSB poate oferi rate de transfer de până la 480 Mbps (la 3 m) sau 110 Mbps (la 10 m).
- WUSB permite, maxim 127 de dispozitive conectate la un singur controler gazdă.

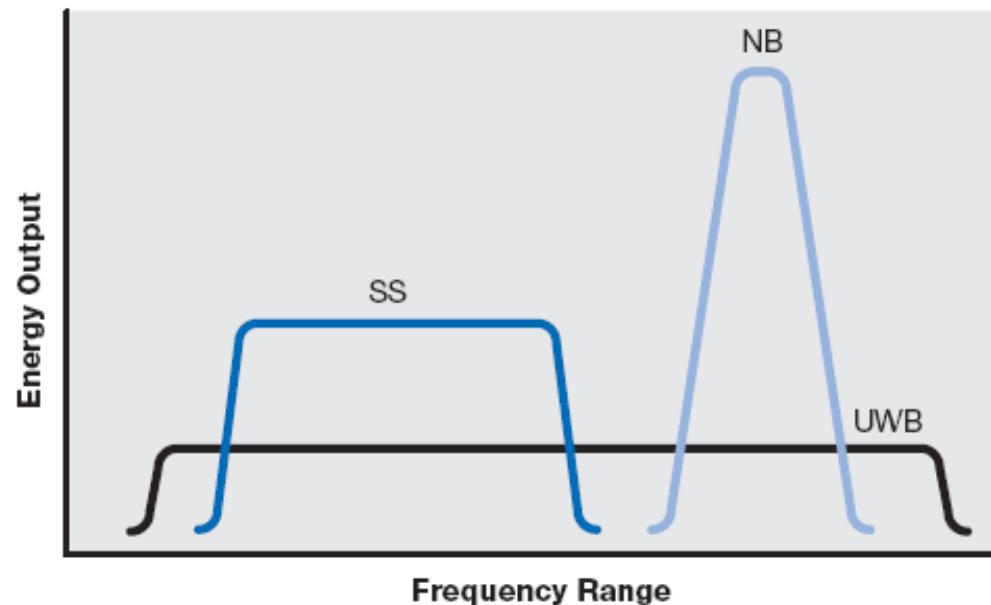
CE ESTE ULTRA-WIDEBAND

- UWB este o tehnologie radio care poate fi utilizată la niveluri de energie foarte scăzute pentru comunicații cu lățime de bandă cu rază scurtă de acțiune utilizând o porțiune mare a spectrului radio
- Spectrul mai larg și puterea redusă îmbunătățesc viteza și reduc interferențele altor dispozitive



- UWB diferă substanțial de alte RF și SS în bandă îngustă, cum ar fi:
 - Tehnologie Bluetooth
 - 802.11a / b / g.
- De asemenea, permite mai multe transferuri de date într-o anumită perioadă de timp.

Figure 2. Comparison of narrowband (NB), spread spectrum (SS), and ultra-wideband (UWB) signal concepts



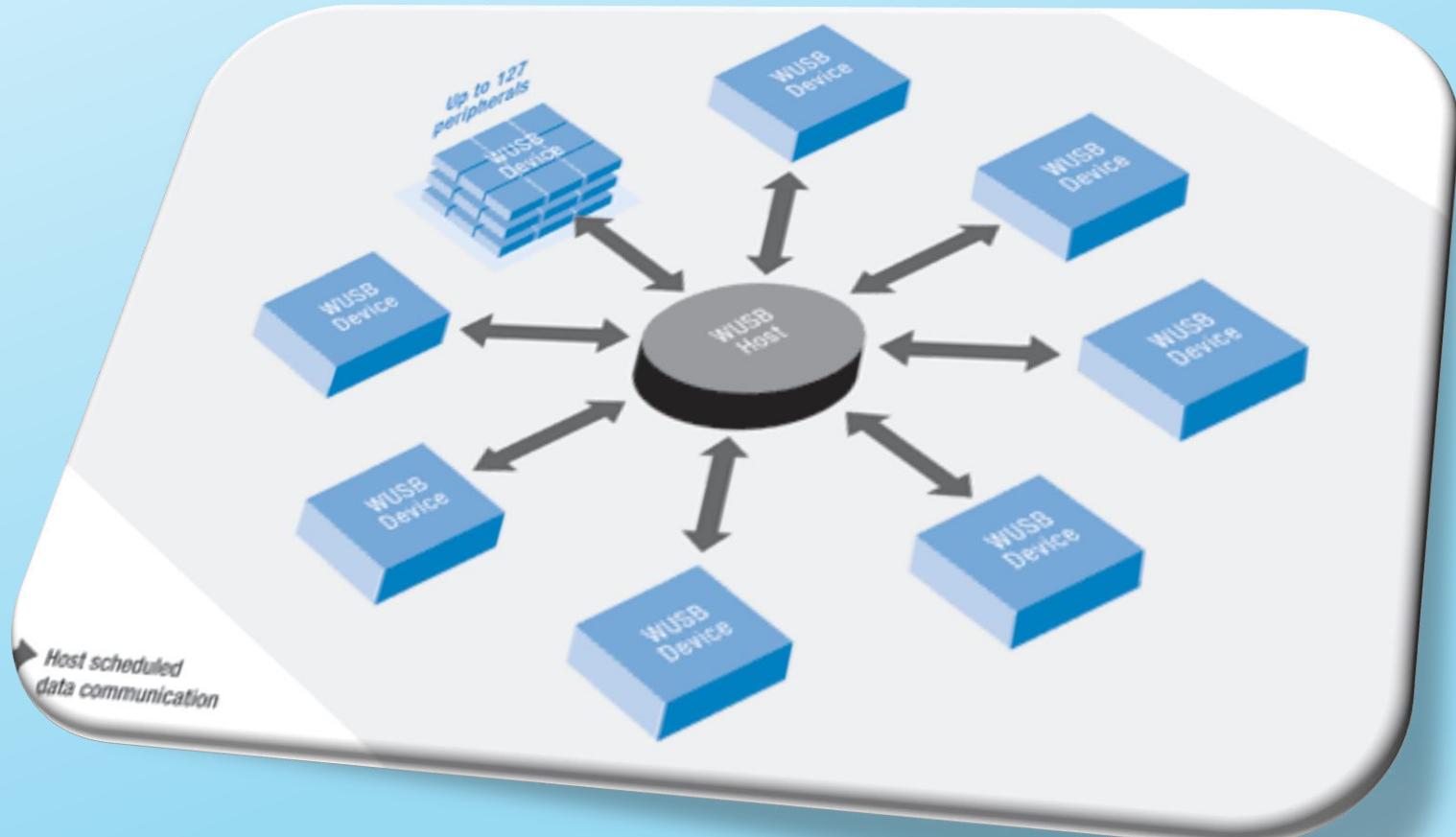
SS = Spread Spectrum
NB = Narrowband
UWB = Ultra-Wideband

DE CE USB WIRELESS

- Cererea sporită de conectivitate fără dezordine
- Păstrează arhitectura stratificată USB 2.0 și fluxul de comunicării
 - Punct la punct (Point-to-point)
 - Aceleași tipuri de transfer etc.
- Interfața WUSB oferă în continuare capacitatea Plug and Play, precum și componente hardware care pot fi schimbată la cald
- Menține compatibilitatea înapoi (1.0 și 2.0)

TOPOLOGIA

- WUSB folosește un model radial
- Gazda WUSB este „hub-ul” și dispozitivele stau la capătul unor raze ”
- Fiecare raza oferă o conexiune punct-la-punct

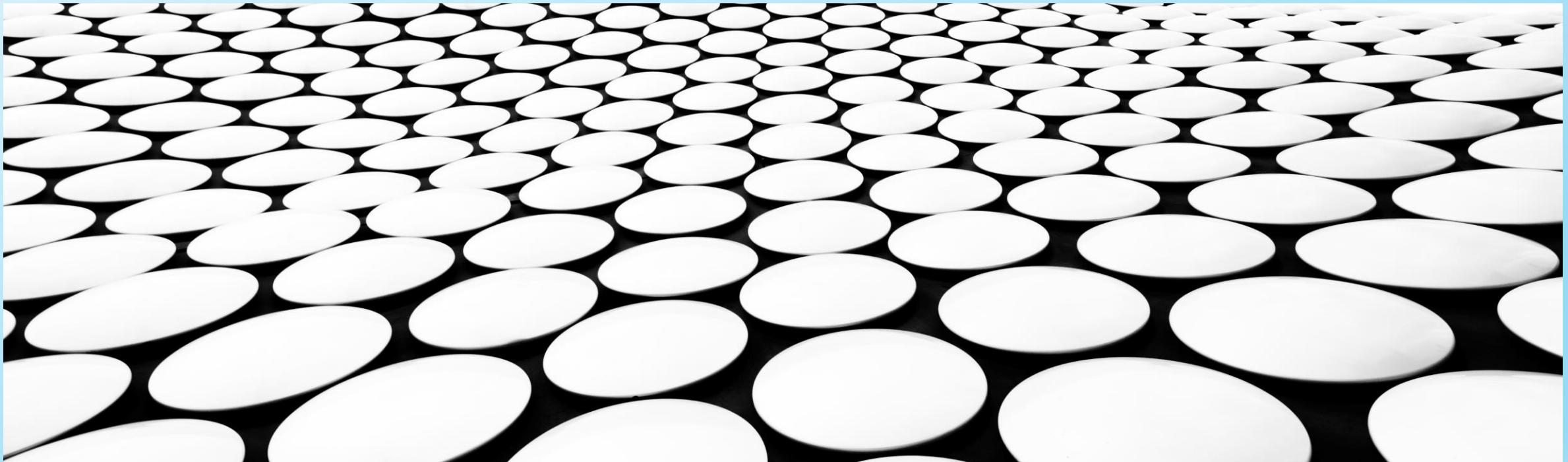


PREZENTAREA GENERALĂ A ARHITECTURII USB

- Un sistem USB constă dintr-o gazdă și un număr de dispozitive care funcționează toate împreună pe aceeași bază de timp și interconectare logică.
- Sistemul USB poate fi descris prin trei zone definitorii:
 - Interconectare USB
 - Dispozitive USB
 - Gazdă USB
- Interconectarea USB este modul în care dispozitivele USB sunt conectate și comunică cu gazda.
- Aceasta include următoarele:
 - Topologie
 - Modele de flux de date
 - Planificare USB

ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



Magistrale USB

(continuare)

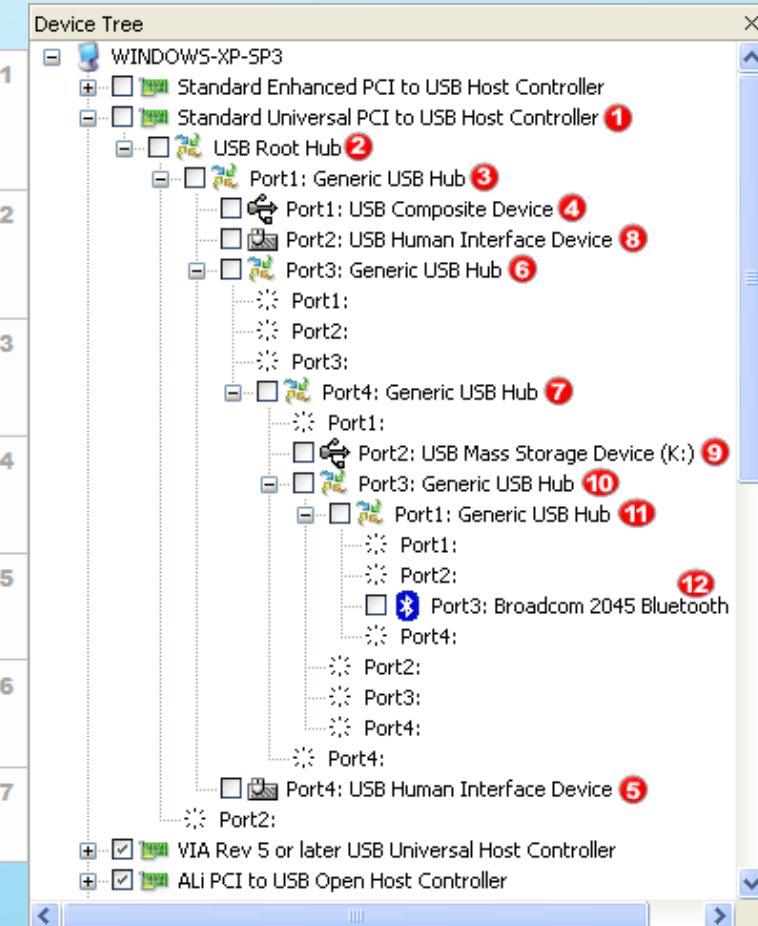
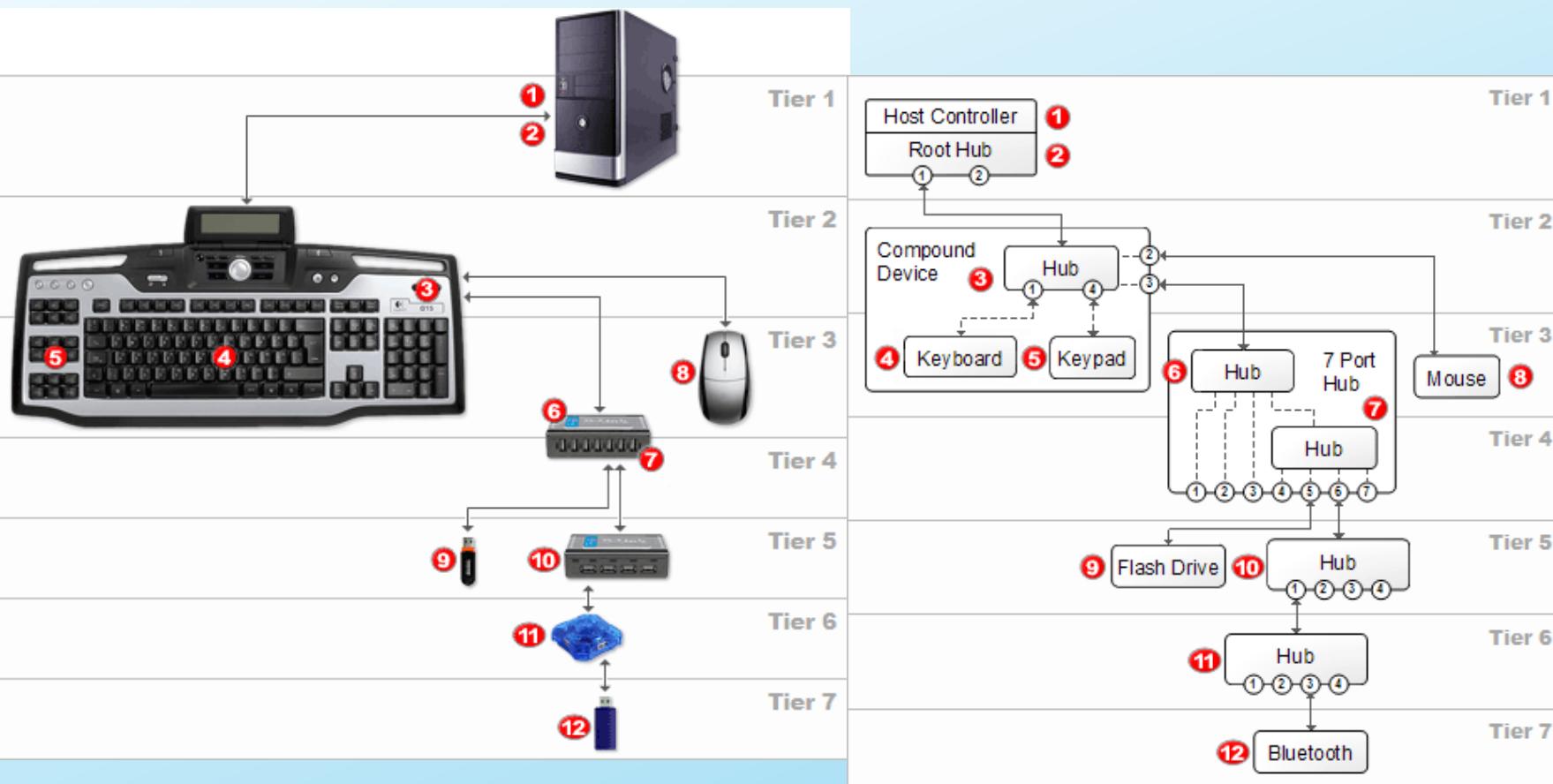
PREZENTAREA GENERALĂ A ARHITECTURII USB

Un sistem USB constă dintr-o **gazdă** și un număr de **dispozitive** care funcționează toate împreună pe aceeași bază de timp și interconectare logică.

Sistemul USB poate fi descris prin trei zone definitorii:

- Interconectare USB
 - Dispozitive USB
 - Gazdă USB
- **Interconectarea USB** este modul în care dispozitivele USB sunt conectate și comunică cu gazda.
- Zona de interconectare include următoarele:
 - Topologia
 - Modele de flux de date
 - Planificarea USB

Exemple de topologie USB



Alimentarea dispozitivelor USB

■ Dispozitive alimentate prin magistrala:

1. de putere redusă

- Consum mai mic de 100 mA
- Compatibile cu alimentarea prin magistrala

2. de putere ridicată

- Între 100 mA și 500 mA
 - Porturile cu **alimentare completă** pot alimenta aceste dispozitive (**Full-powered ports**)
- Pot fi proiectate pentru a avea propria lor sursa de putere
- Opereaza în trei moduri:
 - Configurat (500 mA)
 - Neconfigurat (100 mA)
 - Suspendat (about 2.5 mA)

Alimentarea dispozitivelor USB

■ Huburi USB

■ Alimentate de magistrală

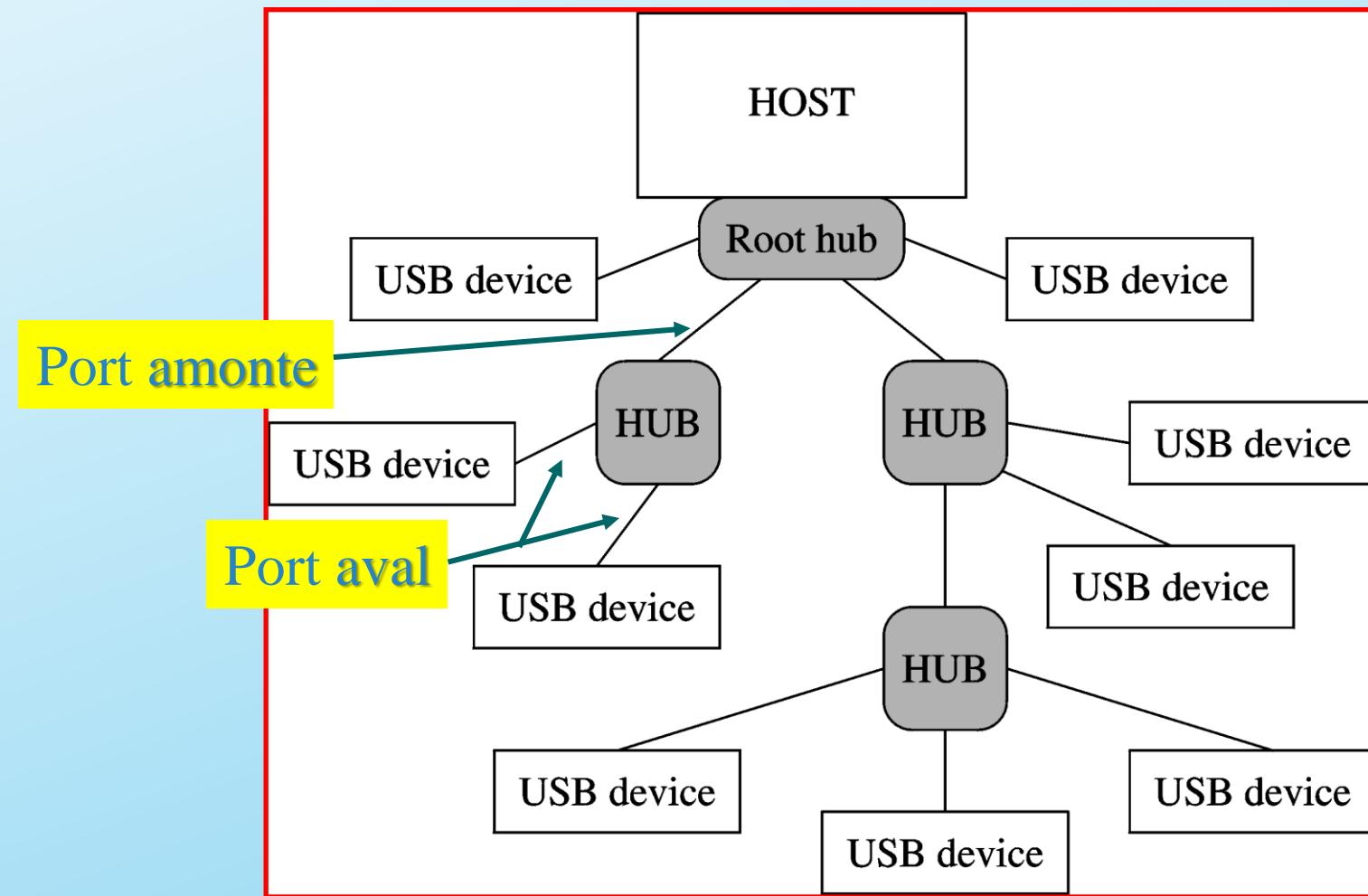
- Nu este necesară o sursă de alimentare suplimentară
- Trebuie conectat în amonte la un port care poate oferi 500 mA
- Porturile din aval pot furniza doar 100 mA
 - Numărul de porturi este limitat la patru
 - Suportă numai dispozitive cu consum redus de energie

■ Auto-alimentate

- Suportă 4 dispozitive de putere ridicată
- Suportă 4 hub-uri USB alimentate prin magistrală
- Majoritatea hub-urilor cu 4 porturi sunt alimentate dual

Topologia USB

Huburile pot fi folosite pentru extinderea magistralei



Endpoints

- Cea mai simplă formă de comunicare USB este prin intermediul unui **punct de capat** (*endpoint*)
 - Este unidirecțională: se transportă date într-o singură direcție
 - De la gazdă la dispozitiv (punctul final OUT)
 - De la dispozitiv la gazdă (punct final IN)

Endpoints

În USB, informațiile circulă între gazdă și dispozitiv.

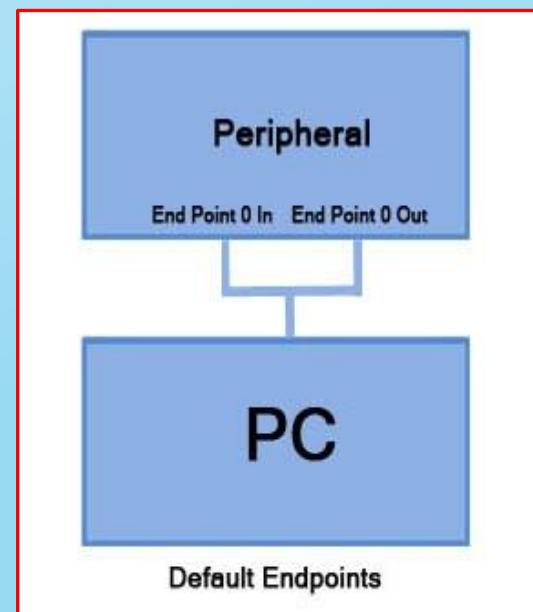
Punctele de capat (*Endpoints*) sunt sursa sau “aspiratorul” informațiilor într-un canal de comunicare. (canal logic)

Pe un periferic sunt atașate Endpoint-uri în perechi (*in + out*).

Cele două *Endpoints* din pereche au același număr (index), dar direcții diferite.

La conectarea dispozitivului numai *Endpoint-ul implicit 0* este accesibil. Acest *Endpoint* primește cerere de control și de stare de la gazdă în timpul procesului de enumerare.

Celelalte *Endpoints* sunt declarate conform cerințelor după configurarea dispozitivului.



Endpoint este unitatea fundamentală de comunicare în USB. Toate datele sunt transferate prin conducte virtuale între gazdă și aceste *endpoints*. Toată comunicarea dintre o gazdă USB și un dispozitiv USB este adresată unui *endpoint* specific de pe dispozitiv. Fiecare *endpoint* al dispozitivului este un receptor sau un emițător de date unidirecțional; fiecare este specificat ca expeditor sau receptor de date de la gazdă.

O **conductă** (pipe) reprezintă o cale de date între gazdă și dispozitiv. O **conductă** poate fi *unidirectională* (constând dintr-un singur punct final) sau *bidirectională* (constând din două puncte finale în direcții opuse).

O **conductă** specială este **conducta de control** implicită. Aceasta constă atât din *endpoint-ul 0* de intrare, cât și *endpoint-ul 0* de ieșire. Este obligatorie pentru toate dispozitivele și trebuie să fie disponibila imediat după ce dispozitivul este alimentat. Gazda folosește această conductă pentru a identifica dispozitivul și punctele sale finale și pentru a configura dispozitivul.

Endpoints nu sunt toate la fel. *Endpoints* specifică cerințele lor de lățime de bandă și modul în care transferă date.

Exista **4 tipuri** de puncte de capat

- de CONTROL
- de INTRERUPERE
- pentru transfer de BLOCURI (BULK)
- pentru transfer IZOCRON

ENDPOINTS

■ CONTROL

- Sunt folosite pentru configurarea dispozitivului, recuperarea informațiilor și transmiterea stării dispozitivului sau trimiterea comenziilor către dispozitiv
- Fiecare dispozitiv are un punct final de control numit punct final 0
 - Este folosit de nucleul USB pentru a configura dispozitivul la momentul inserării
 - Transferurile sunt garantate cu lățime de bandă rezervată

ENDPOINTS

■ INTERRUPT

- Transferă cantități mici de date la o rată fixă
- Este utilizat pentru tastatura și mouse USB
- De asemenea, este utilizat pentru a controla dispozitivul
- Nu este folosit pentru transferuri mari
- Lățime de bandă rezervată garantată

ENDPOINTS

■ BULK

- Transfer de cantități mari de date
- Fără pierderi de date
- Nu este garantat timpul
- Un pachet BULK ar putea fi împărțit pe mai multe transferuri
- Folosit pentru imprimante, stocare și dispozitive de rețea

ENDPOINTS

- IZOCRON
 - Transfera o cantitate mare de date
 - Pentru cantitati de date în timp real, dispozitive A/V
 - Spre deosebire de punctele finale BULK, nu există garanții (pierderi potențiale de date)
- Punctele finale CONTROL și BULK sunt utilizate pentru transferuri de date asincrone
- Punctele finale INTRERUPT și ISOCHRON sunt pentru transferuri periodice (repetate) cu lățime de bandă rezervată

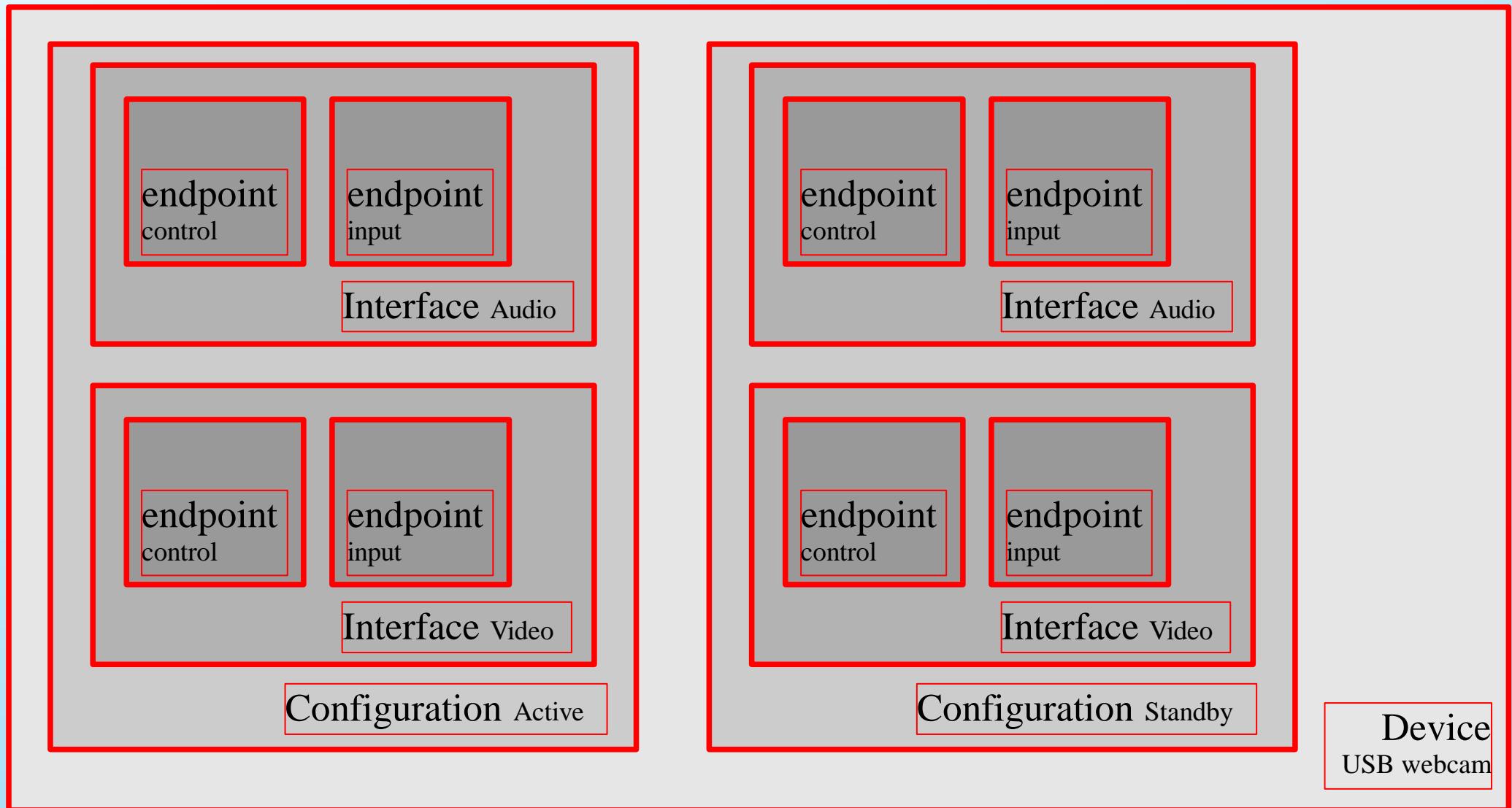
INTERFETE

- Punctele finale USB sunt grupate în interfețe
 - O interfață gestionează un singur tip de conexiune logică (E.g., un mouse)
 - Unele dispozitive au mai multe interfețe
 - E.g., un speaker
 - O interfață pentru butoane și una pentru fluxul audio
- Interfața USB poate avea setări alternative
 - E.g., setări diferite pentru a rezerva diferite lățimi de bandă pentru dispozitiv

CONFIGURATII

- Interfețele USB sunt grupate în configurații
- Un dispozitiv USB poate avea mai multe configurații
 - Doar una poate fi activă la un moment dat
 - Poate comuta între ele

EXEMPLU DE DISPOZITIV WEBCAM USB



Clasa dispozitivului

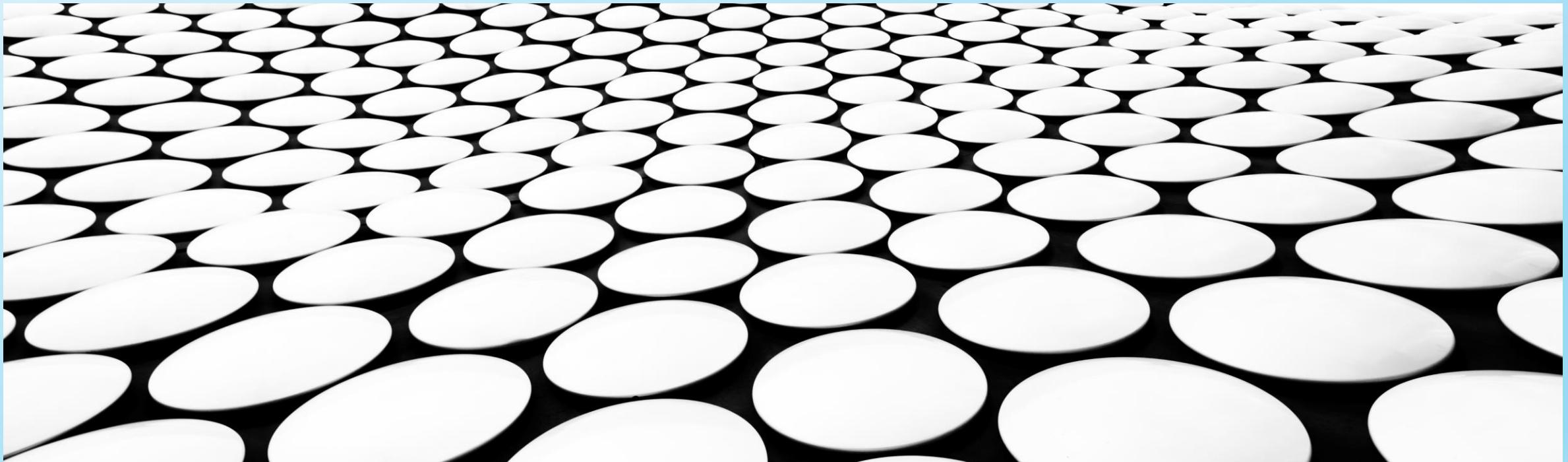
Fiecare dispozitiv USB se incadreaza intr-o clasă de dispozitive care definește funcționalitatea și scopul dispozitivului respectiv.

Gazda încarcă driverul potrivit în funcție de clasa dispozitivului. Cele mai frecvente clase de dispozitive sunt:

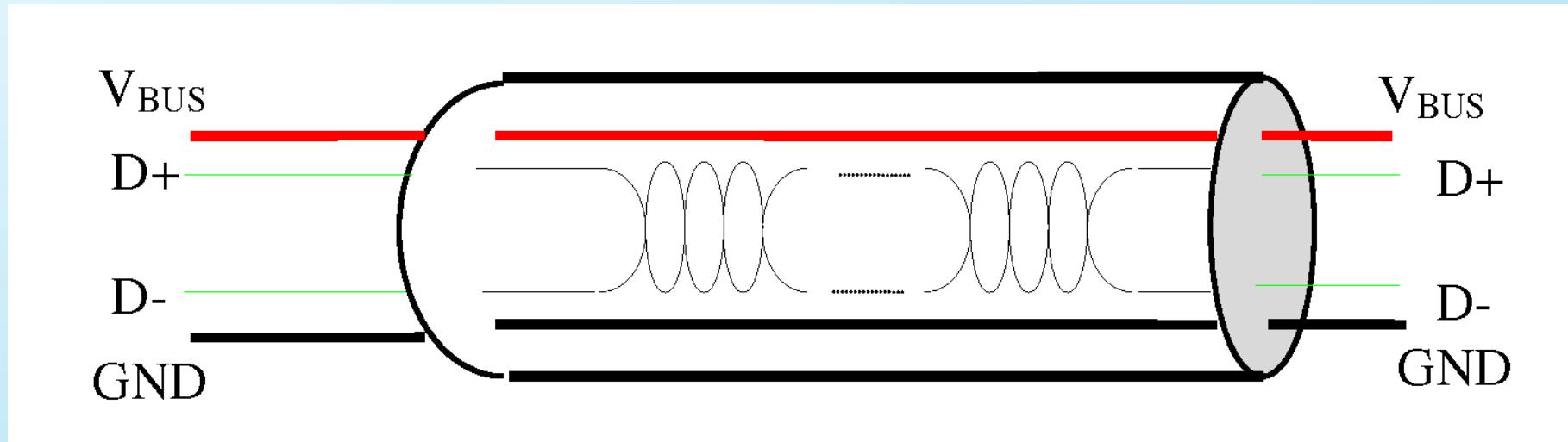
Common USB Device Classes			
Class	Usage	Description	Examples
01h	Interface	Audio	Speaker , microphone , sound card
03h	Interface	Human Interface Device	Keyboard , mouse , joystick
07h	Interface	Printer	Laser printer , inkjet printer , CNC machine
08h	Interface	Mass storage	USB flash drive , memory card reader , digital audio player , digital camera , external
09h	Device	USB hub	Full bandwidth hub
0Bh	Interface	Smart Card	USB smart card reader
0Dh	Interface	Content Security	Finger Print Reader
0Eh	Interface	Video	Webcam

ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



Cablu USB 2

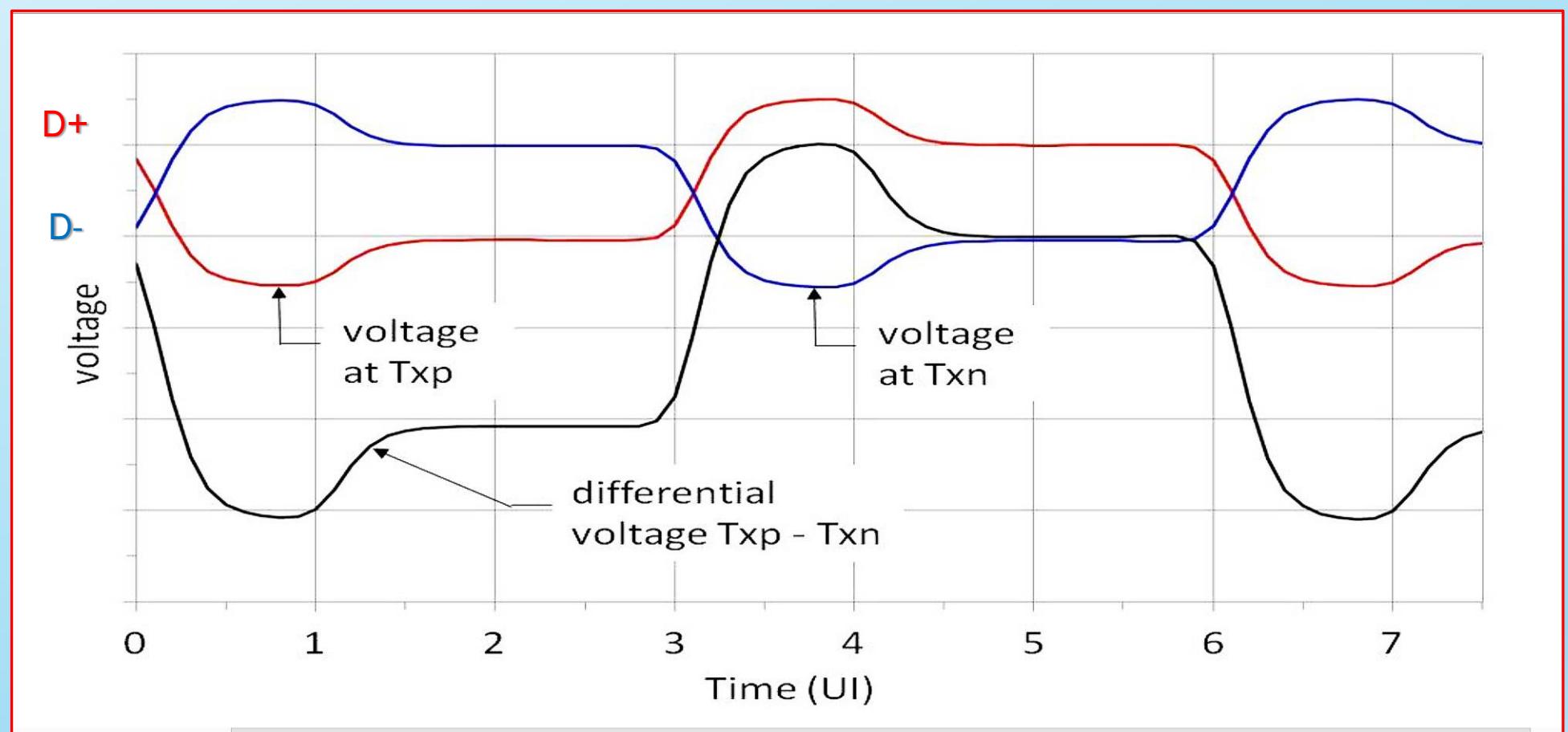
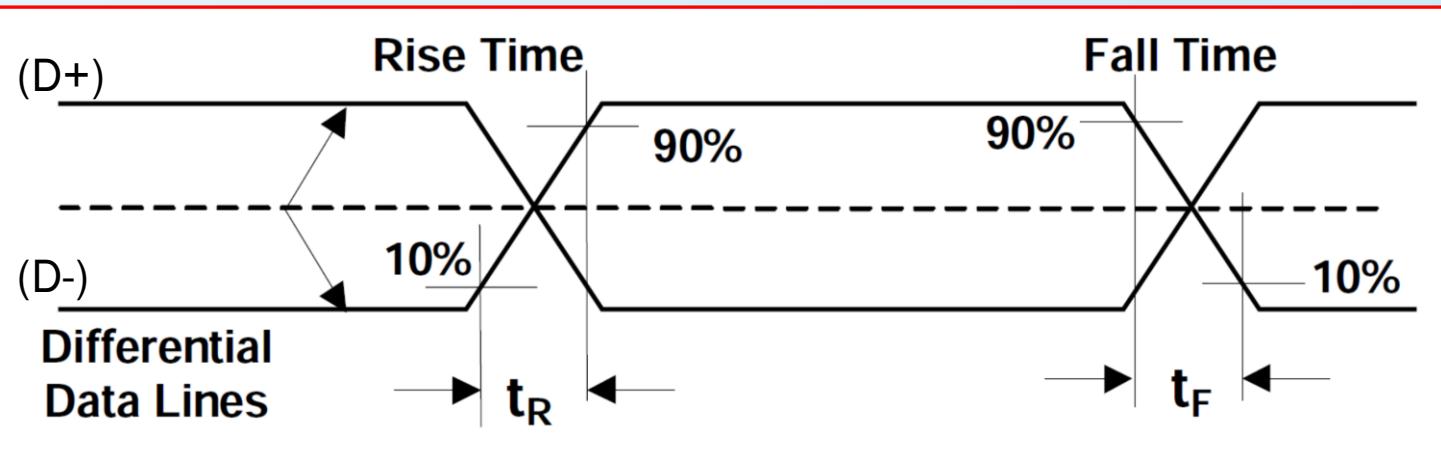


2 perechi de fire:

- o pereche pentru date (D+, D-): semnal differential
- o pereche pentru alimentare cu energie: (V_{BUS}, GND)

Semnale USB

In semnalul diferential
se codifica informația



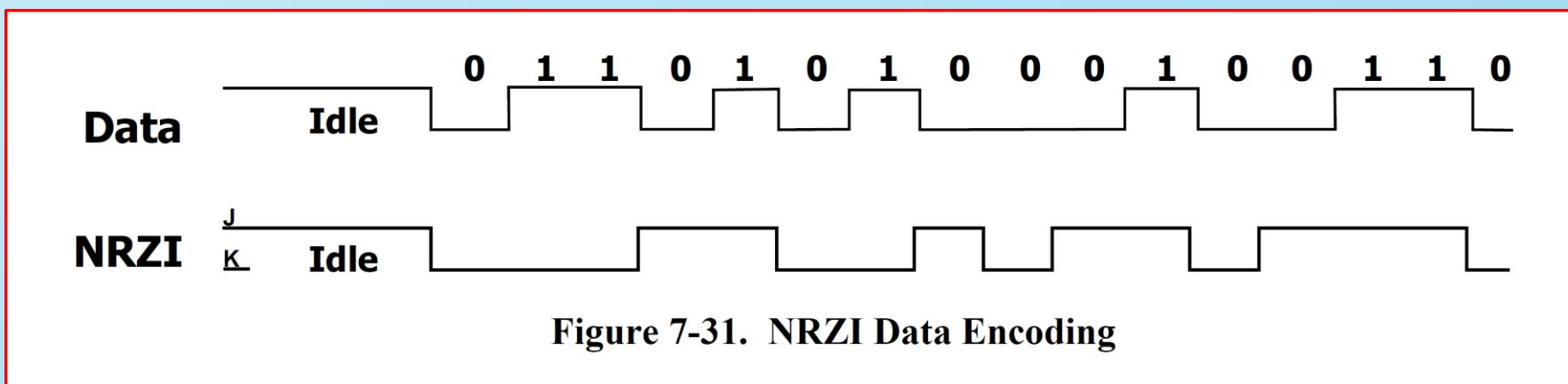
Codificarea/Decodarea datelor

USB-ul folosește **codificarea NRZI** la transmiterea pachetelor de date.

În codificarea NRZI,

un „1” este reprezentat absenta modificarii a nivelului semnalului differential și

un „0” este reprezentat de o schimbare a nivelului.



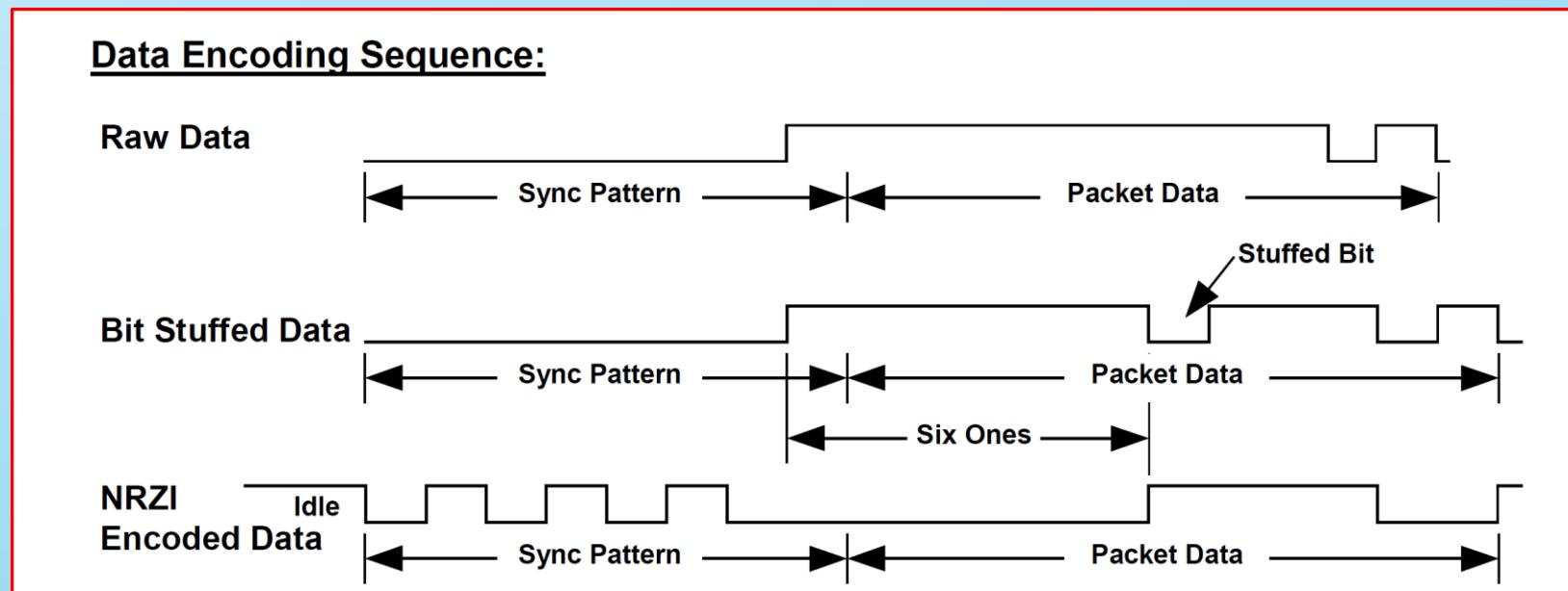
Nivelul înalt reprezintă starea J pe liniile de date.

Un sir de zerouri face ca datele NRZI să comute de fiecare dată pe bit.

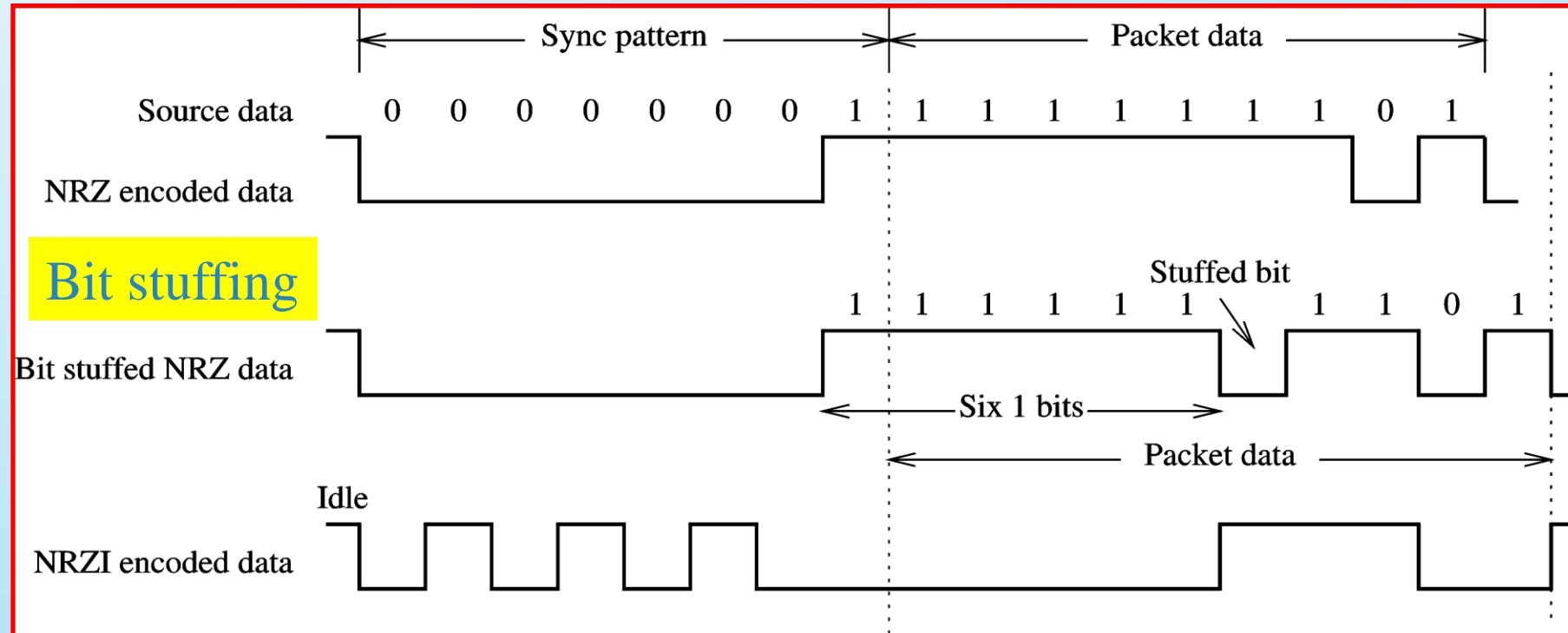
Un sir de unu cauzează perioade lungi fără tranziții în sirul NRZI de date.

Pentru a asigura tranziții adecvate ale semnalului, dispozitivul de transmisie folosește umplutura de biți atunci când trimite a pachet pe USB

Un zero este introdus după fiecare șase biti consecutivi în sirul de date, înainte ca datele să fie codificate NRZI, pentru a forța o tranziție în fluxul de date NRZI.

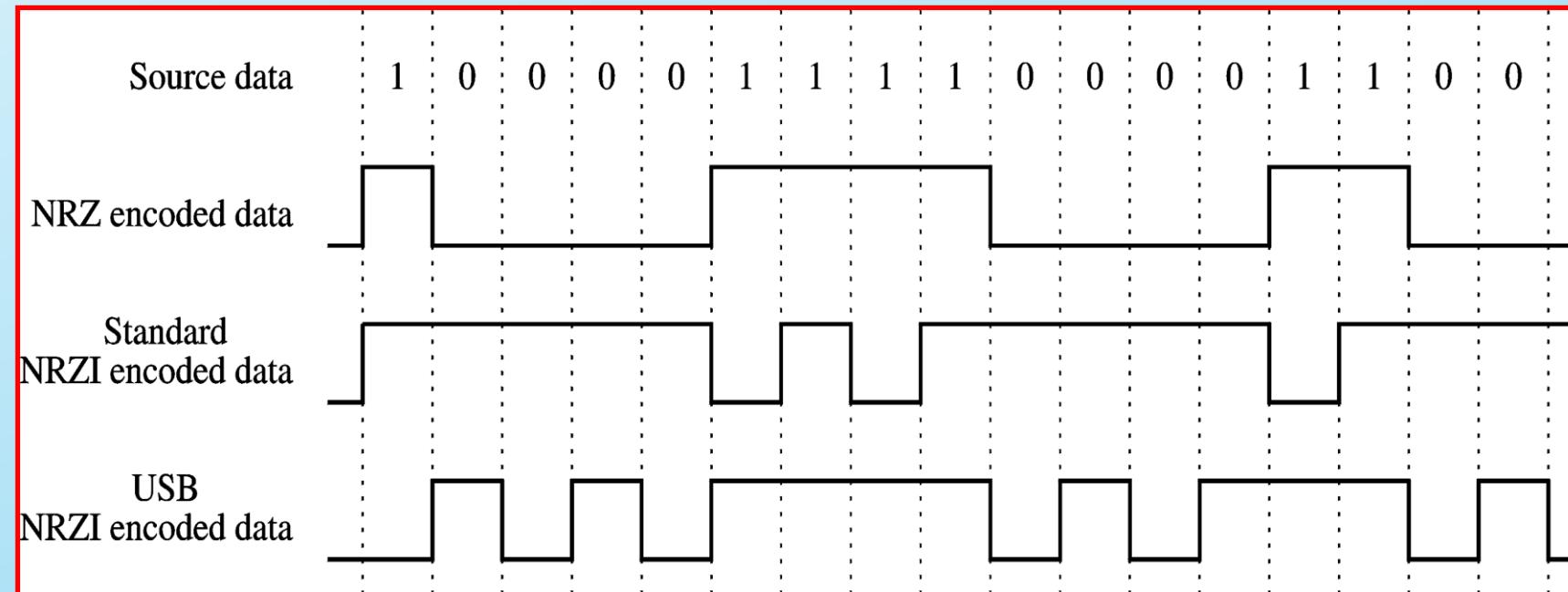


USB



■ Un alt exemplu de codificare USB

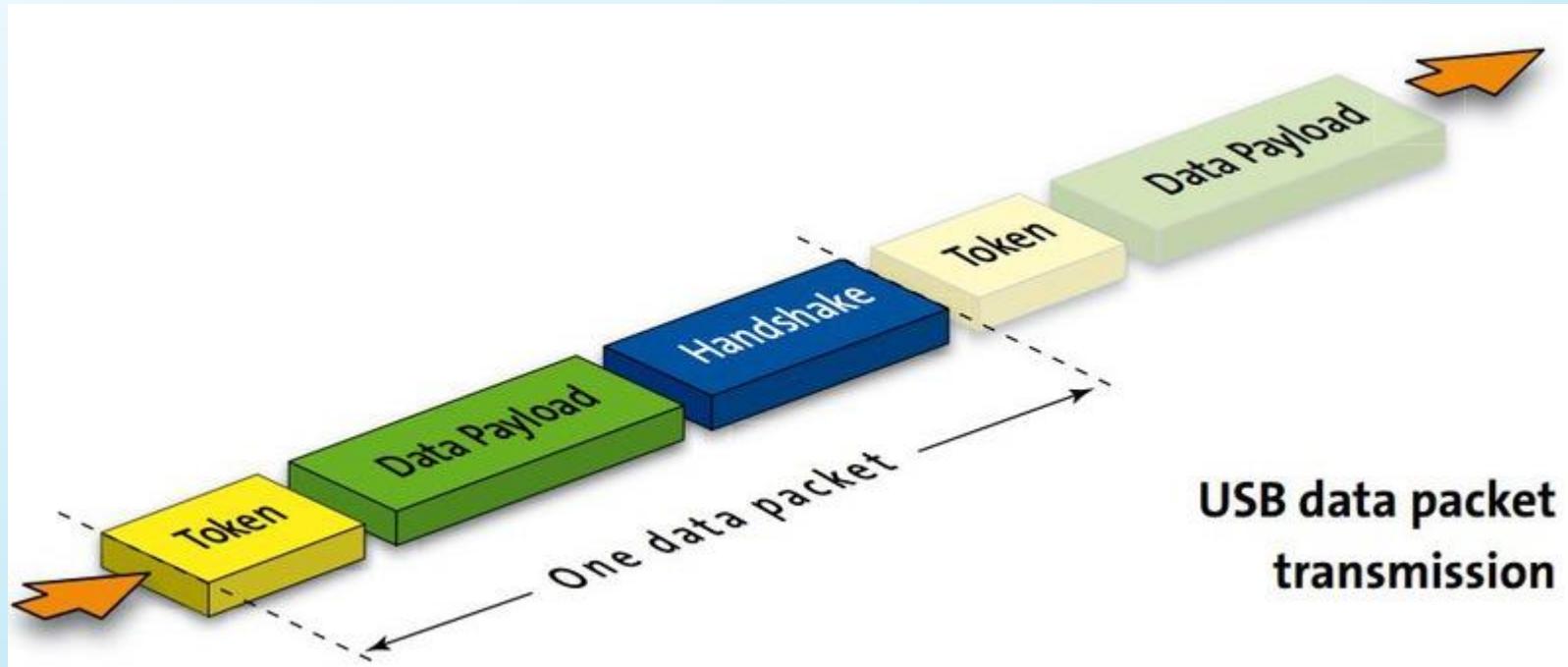
- Se foloseste codificarea **NRZI**
 - Non-Return to Zero-Inverted



MANIPULAREA PACHETELOR USB

- Toate transferurile de date sunt inițiate de gazdă (host)
 - Un dispozitiv USB nu va incepe niciodata sa trimita date fara sa fie mai intai invitat de HOST
- Huburile pot fi organizate in maxim 5 straturi in adâncime de (127 de dispozitive)
- **Pachetele sunt direcționate, NU sunt difuzate**
- Huburile utilizează proceduri de „stocare și redirecționare”
 - Pachete sunt reținute de hub daca sunt direcționate către un port inactiv
- Pachetele directionate aval folosesc un sir de rutare pentru a naviga către dispozitiv
- Pachetele directionate amonte conțin întotdeauna gazda ca destinație





Transferul pachetelor

Tipuri de pachete USB

Fiecare protocol de comunicare implică schimbul de pachete. Același lucru este cazul cu USB. Aceste pachete încapsulează informațiile într-un mod organizat standard.

Aceste pachete conțin în general informații legate de:

- Controlul schimbului de date
- Schimbul de date sub formă de sarcină utilă reală
- Detectarea și corectarea erorilor prin verificarea stării

Câmpurile pachetelor USB

În USB, LSB-ul pachetului este transmis mai întâi. Un pachet USB conține câmpuri diferite. Ele sunt:

- **Sync:** este un câmp obligatoriu care apare la începutul pachetului. Acest câmp sincronizează ceasul receptorului cu emițătorul. Pentru modul de viteză mică și maximă, acest câmp are o lungime de 8 octeți și pentru modul de viteză mare are o lungime de 32 de octeți.
- **PID:** PID înseamnă ID-ul pachetului. Indică tipul de pachet care este trimis. Acest câmp are o lungime de **8 biți**. Cei patru biți superioiri identifică tipul de pachet, iar cei patru biți inferiori sunt complimentul bit-wise al celor patru biți superioiri. Cei patru biți inferiori ajută la detectarea erorilor.
- **ADDR:** acest câmp conține adresa de destinație a dispozitivului USB. Este **de 7 biți**, ceea ce înseamnă că poate suporta $2^7 = 127$ dispozitive.
- **ENDP:** Acest câmp specifică numărul *endpoint*. Este de **4 biți**, ceea ce înseamnă că poate indica $2^4 = 16$ puncte finale posibile.
- **CRC:** CRC înseamnă Cyclic Redundancy Check. Acest câmp este utilizat pentru a verifica datele din pachet pentru orice eroare în procesul CRC. Pentru pachetele token, se utilizează CRC pe **5 biți**, iar pentru pachetele de date se utilizează CRC pe **16 biți**.
- **EOP:** EOP înseamnă End of Packet. Acest câmp semnalizează liniile de date pentru Single Ended Zero (SE0) timp de aproximativ 2 biți, urmat de starea J (stare inactivă) timp de 1 biți.

Packet Type	PID Value	Packet Identifier
Token	0001	OUT Token
	1001	IN Token
	0101	SOF Token
	1101	SETUP Token
Data	0011	DATA0
	1011	DATA1
	0111	DATA2
	1111	M DATA
Handshake	0010	ACK
	1010	NAK
	1110	STALL
	0110	NYET
Special	1100	Preamble
	1100	ERR
	1000	Split
	0100	Ping

1. Token packets: Aceste pachete sunt trimise numai de către gazdă.

Structura pachetului conține
un octet PID,
11 biți de adresă și
un CRC de 5 biți.

Tipuri de pachete de *token*:

- *In* - Acest pachet notifică dispozitivul USB că gazda dorește să citească informații.
- *Out* - Acest pachet notifică dispozitivul USB pe care gazda dorește să scrie.
- *Setup* - Acest pachet este utilizat pentru a porni transferul de control
- *Ping* - Înainte de a trimite perechea de pachete OUT / DATA, acest token solicită dispozitivului USB sa confirme dacă este gata să primească perechea de pachete OUT / DATA.
- *Split* - Acest token este utilizat pentru a comunica cu un dispozitiv cu viteză mică / maximă pe o magistrală de mare viteză

Sync	PID	Device Address	End Point	CRC 5-Bit	EOP
------	-----	----------------	-----------	-----------	-----

2. Data packets: Există două tipuri de pachete de date, *Data0* și *Data1*. Structura pachetului conține:
un octet PID,
câmp de date și
CRC pe 16 biți.

Câmpul de date poate transporta 0-1023 octeți de date. Datele trebuie trimise întotdeauna în multipli de octeți.

- Pentru dispozitivele cu viteză redusă, câmpul maxim de date este de 8 biți.
- Pentru dispozitivele cu viteză maximă, câmpul maxim de date este de 1023 biți.
- Pentru dispozitivele de mare viteză, câmpul maxim de date este de 1024 biți

După USB2.0, s-au adăugat încă două tipuri *Data2* și *MData*.

Acestea sunt utilizate numai în transferul de mare viteză cu transfer lățimii de bandă izocronice atunci când este nevoie să se transfere mai mult de 1024 octeți la 8192 kB / s.



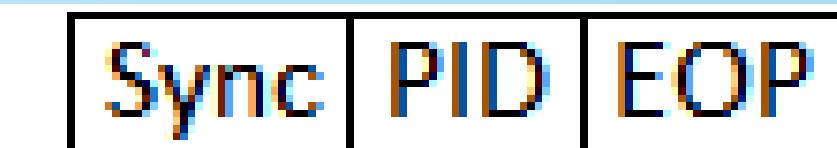
3. Handshake Packets: Aceste pachete sunt trimise în mare parte ca răspuns la pachetele de date. Ele constau pur și simplu dintr-un octet PID.

Există trei tipuri de pachete de strângere de mână:

- ACK - Confirmare pentru pachetul primit
- NAK - indicând faptul că pachetele nu pot fi primite sau trimise temporar. De asemenea, este folosit pentru a indica faptul că nu există date de transmis
- STALL - indicând faptul că dispozitivul este în stare de eroare și are nevoie de intervenția gazdei

Cu USB 2.0, au fost adăugate încă două pachete:

- NYET - indicând că tranzacția Split nu este încă finalizată.
- ERR - indicând eșecul tranzacției



4. Start of Frame packets (SOF): Pachetul SOF constă dintr-un număr de cadre incremental de 11 biți.

Pe o magistrală USB cu viteză maximă, acest pachet este trimis de gazdă la fiecare 1 ms și pe magistrala USB de mare viteză la fiecare 125 de μ s. Acest pachet este utilizat pentru sincronizarea transferului izocron.

Sync	PID	Frame number	End Point	CRC	EOP
------	-----	--------------	-----------	-----	-----

Tipuri de transfer

- Există 4 tipuri de transfer

1. Transfer de intrerupere

- Foloseste sondajul
 - Intervalul de sondare poate varia de la 1 ms la 255 ms

2. Transfer izocron

- Folosit în aplicații în timp real care necesită o rată constantă de transfer de date
 - Exemplu: Citire audio de pe CD-ROM
- Aceste transferuri sunt planificate în mod regulat
- Nu folosesc detectarea și recuperarea erorilor

3. Transfer de control

- Folosit pentru a configura și initialize dispozitive USB
- Trei faze
 - Etapa de configurare
 - Trimite tipul de cerere adresată dispozitivului țintă
 - Etapa datelor
 - Etapa optională
 - Transfer de control care necesită utilizarea de date în aceasta etapa
 - Etapa de stare
 - Verifică starea operațiunii
- Se alocă o lățime de bandă garantată de 10%
- Se utilizează detectarea erorilor și recuperarea
 - Recuperarea se face prin reîncercări

4. Transfer în bloc

- Pentru dispozitive fără cerințe specifice privind rata de transfer de date
 - Exemplu: trimiterea datelor către o imprimantă
- Alocare a lățimii de bandă cu cea mai mică prioritate
 - Dacă celelalte trei tipuri de transferuri iau 100% din lățimea de bandă transferurile în bloc sunt amânate până când sarcina scade
- Se utilizează detectarea erorilor și recuperarea
 - Recuperarea se face prin reîncercări

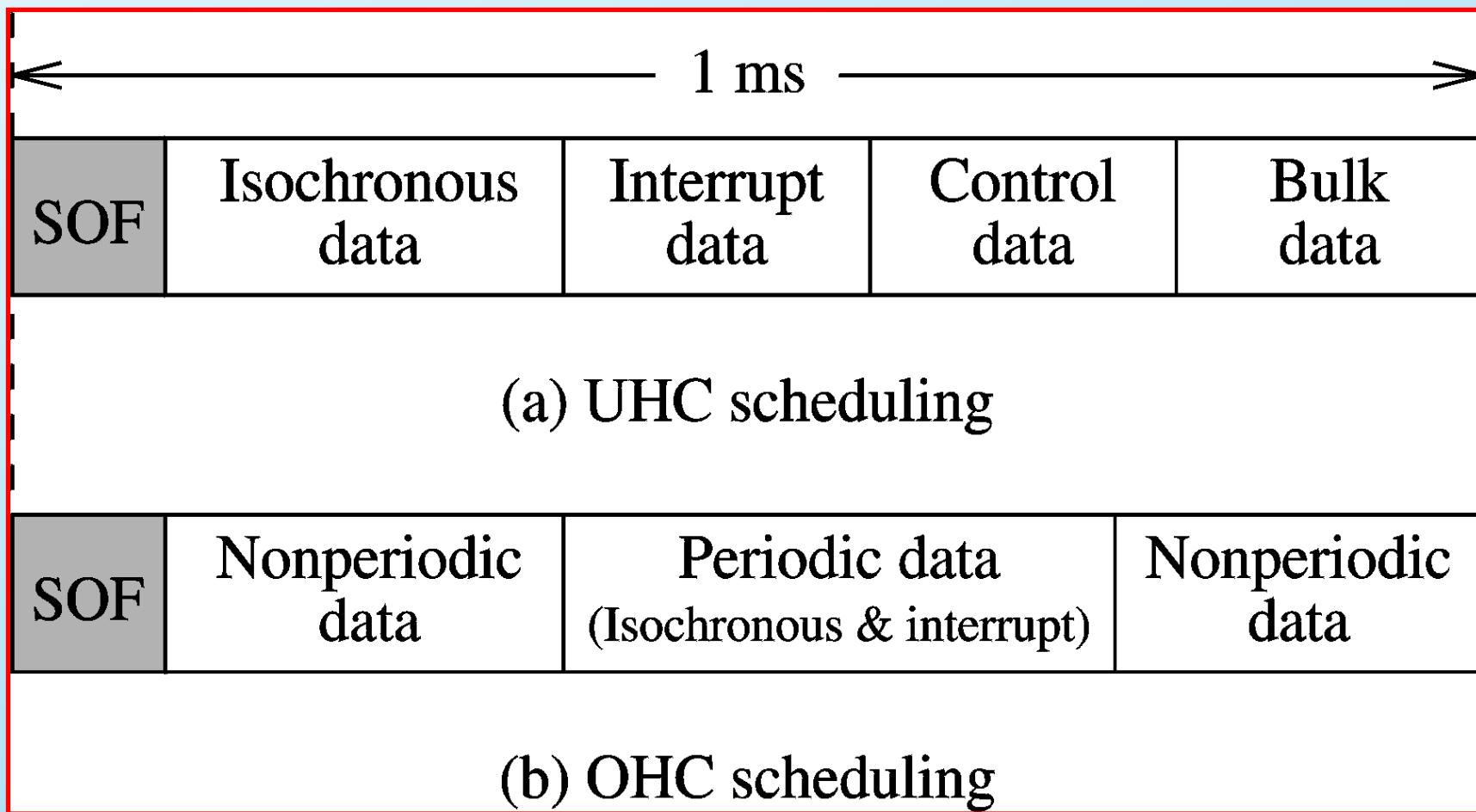
Arhitectura USB

- USB host controller
 - Inițiază tranzacții prin USB
- Root hub
 - Oferă puncte de conectare
- Două tipuri de controlere gazdă (host)
 - Open host controller (OHC)
 - Definit de Intel
 - Universal host controller (UHC)
 - Specificat de National Semiconductor, Microsoft, Compaq
 - Diferența dintre cele două
 - Cum planifică cele patru tipuri de transferuri

■ Planificarea la UHC

- Programează mai întâi transferurile periodice
 - Transferuri periodice: izocrone și de întrerupere
 - Poate ocupa până la 90% din lățimea de bandă
- Aceste transferuri sunt urmate de cele de control și transferurile în bloc
 - Transferurile de control sunt garantate cu 10% din lățimea de bandă
- Transferurile în bloc sunt programate numai dacă există lățime de bandă disponibilă

USB



Planificarea OHC

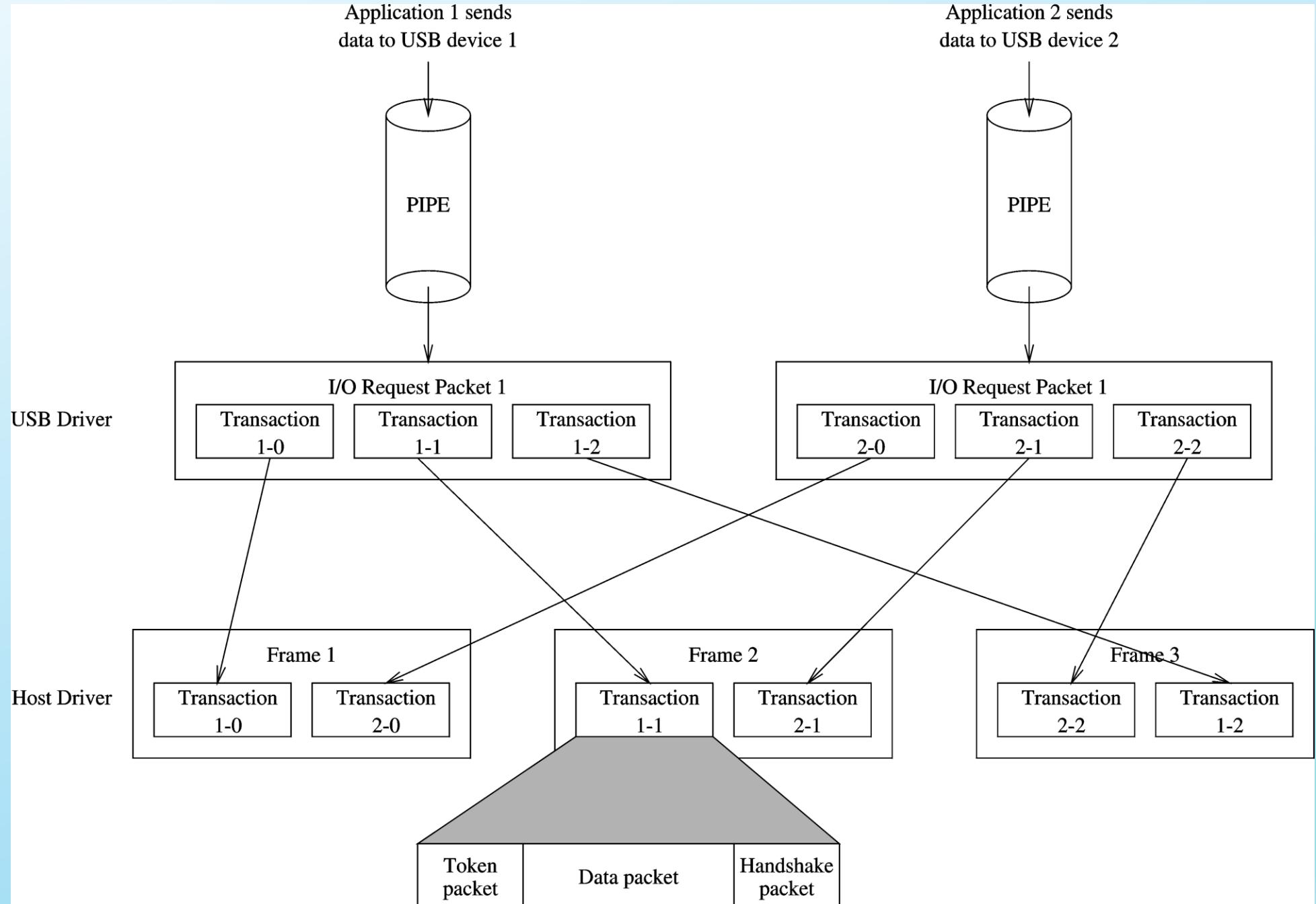
- Diferita de programarea UHC
- Rezervă mai întâi spațiu pentru transferuri non-periodice
 - Transferuri non-periodice: de control și in bloc
 - **10%** lățime de bandă rezervată
- Următoarele sunt programate transferurile periodice
 - Garantează lățimea de bandă de **90%**
 - Lățimea de bandă rămasă este alocată transferurilor non-periodice

Tranzactii USB

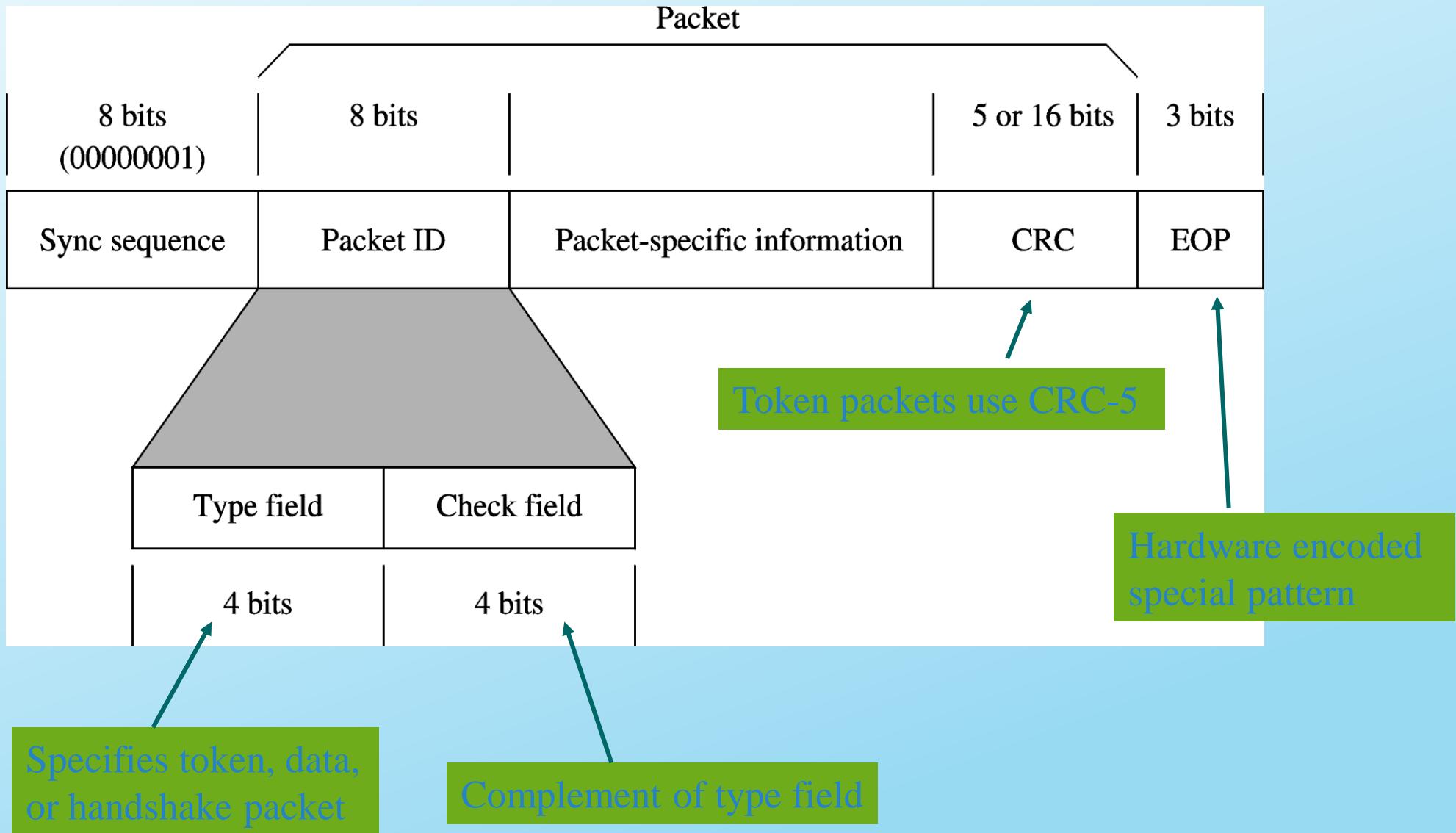
- Transferurile se fac în una sau mai multe tranzacții
 - Fiecare tranzacție constă din mai multe pachete
- Tranzacțiile pot avea între 1 și 3 faze
 - **Faza Token packet**
 - Specifică tipul tranzacției și adresa dispozitivului țintă
 - **Faza Data packet (optional)**
 - Se transferă maximum 1023 octeți
 - **Faza Handshake packet**
 - Cu excepția transferurilor izocrone, celelalte tipuri de transferuri folosesc detectarea erorilor pentru livrarea garantată
 - Oferă feedback cu privire la faptul ca datele au fost primite fără erori

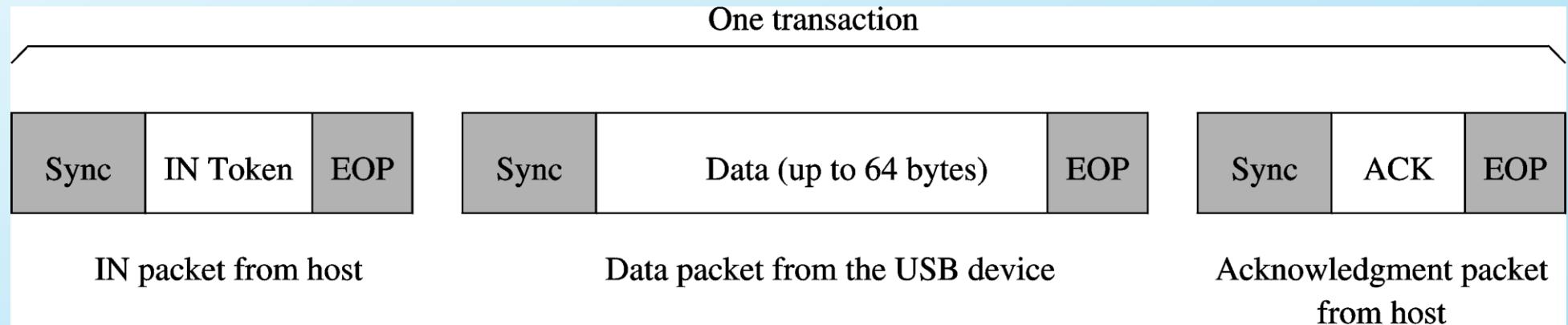
USB

USB IRP frame

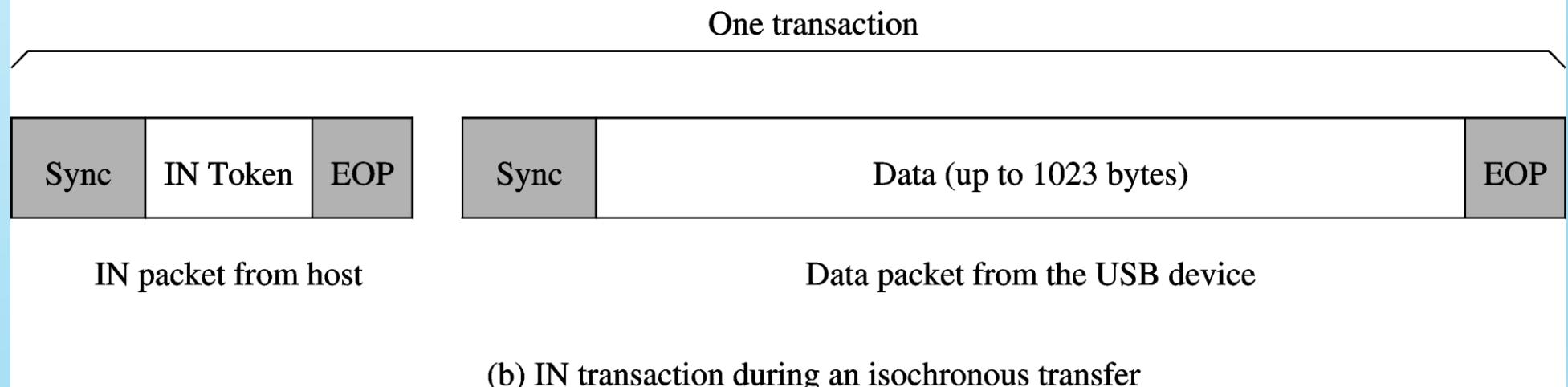


USB





(a) IN transaction without errors



(b) IN transaction during an isochronous transfer

USB

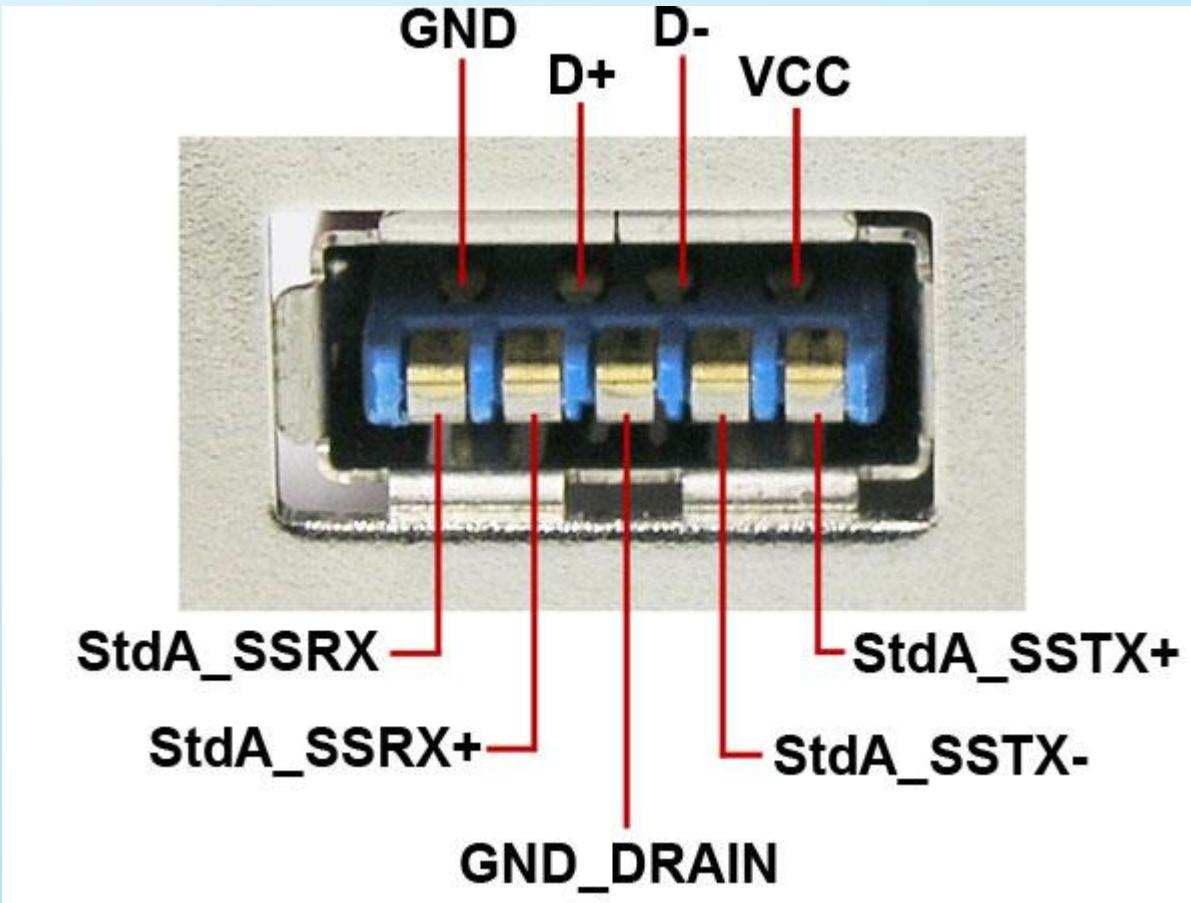
- USB 2.0

- Foloseste o singura pereche de fire pentru transferal bidirectional al datelor
- USB 1.1 foloseste frameuri de 1 ms
- USB 2.0 foloseste frameuri de $125 \mu\text{s}$
 - 1/8 din USB 1.1
- asigura rate de transfer de 40X
 - Pana la 480 Mbps
- Este competitiva cu
 - SCSI
 - IEEE 1394 (FireWire)

USB 3.0

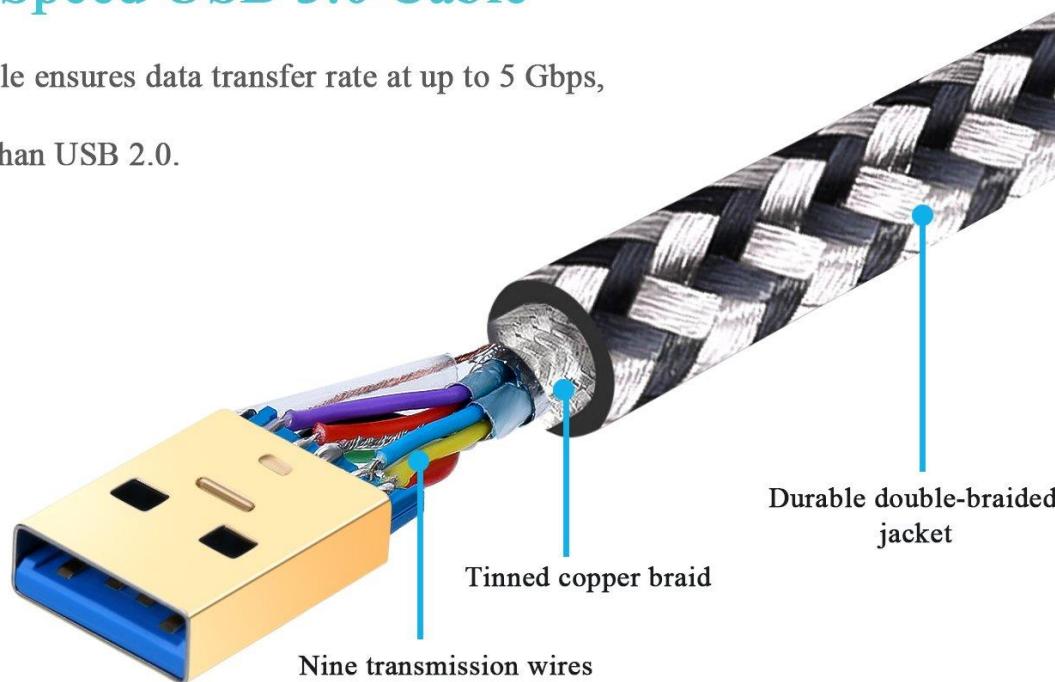
Semnalele USB 3.0 sunt transferate pe două perechi de fire dedicate, diferențiale pentru transmisie și recepție.

Datorită naturii full-duplex a USB 3.0, magistrala funcționează diferit la nivel fizic fata de USB 2.0.



Super Speed USB 3.0 Cable

Quality cable ensures data transfer rate at up to 5 Gbps,
10x faster than USB 2.0.

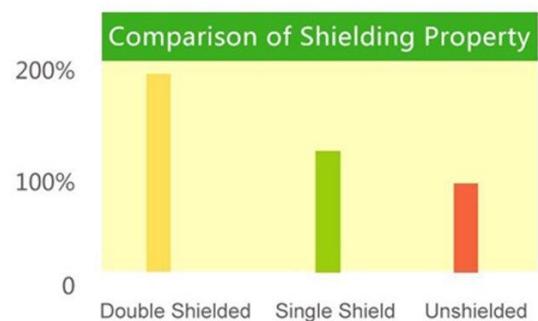


480Mbps

USB2.0

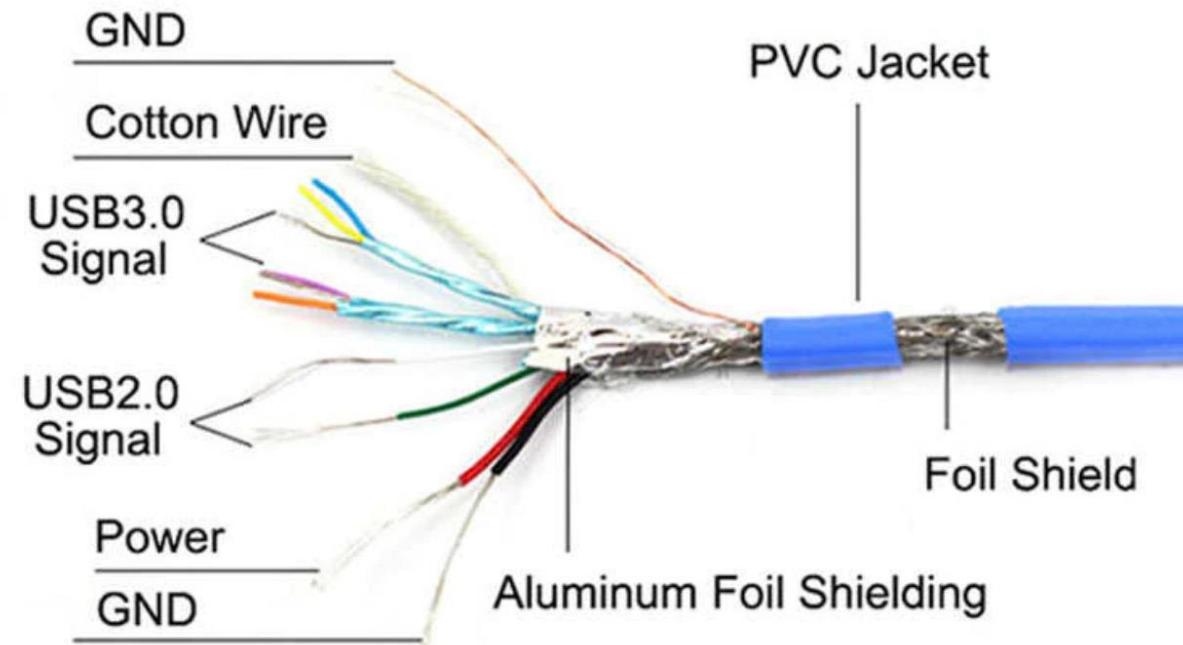
5Gbps

USB3.0

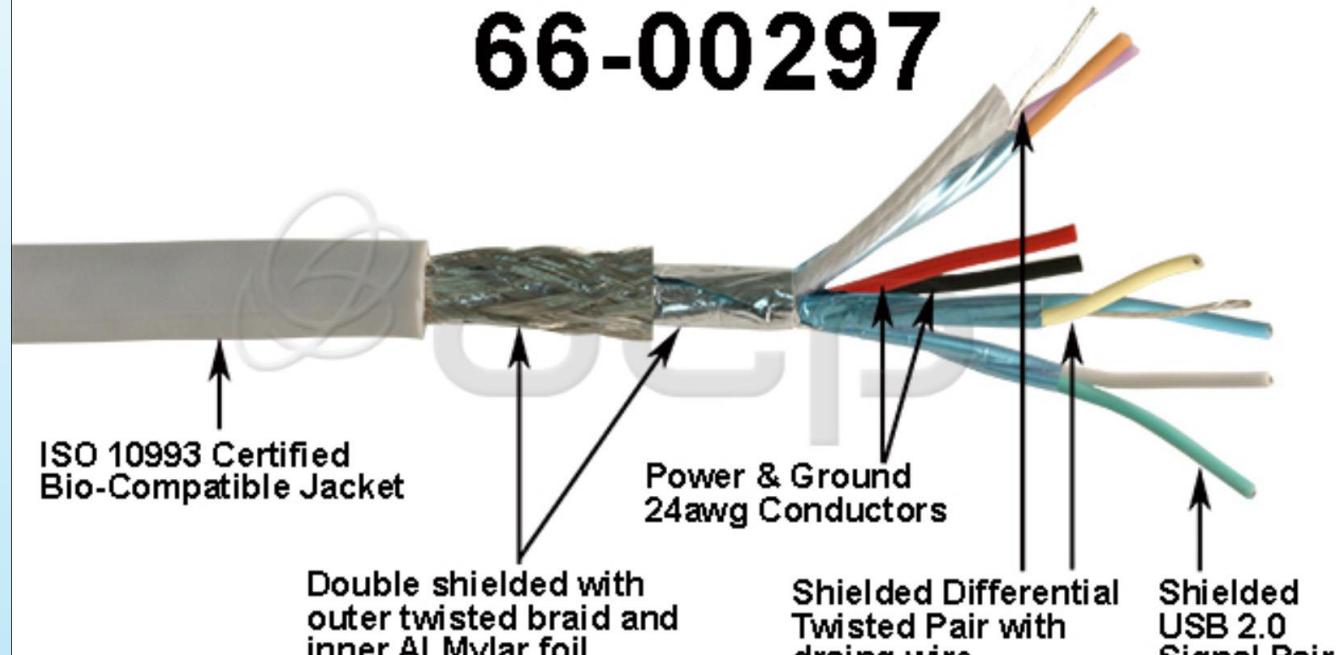


Multiple Protection Shield

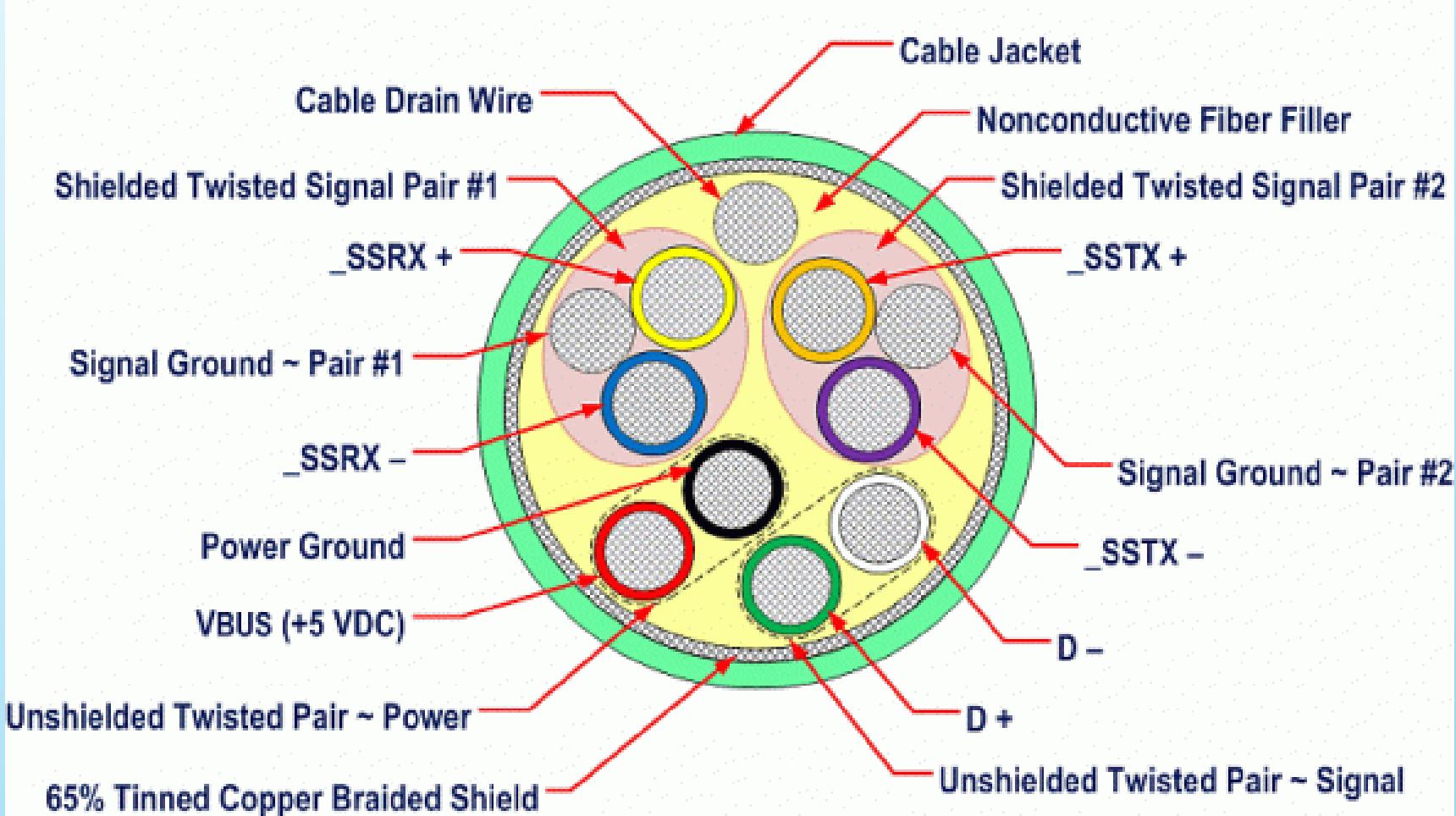
Three-layer Shielding, Aluminum Foil + Myra +
Aluminum And Magnesium Braided High-speed
Transmission Experience.



66-00297

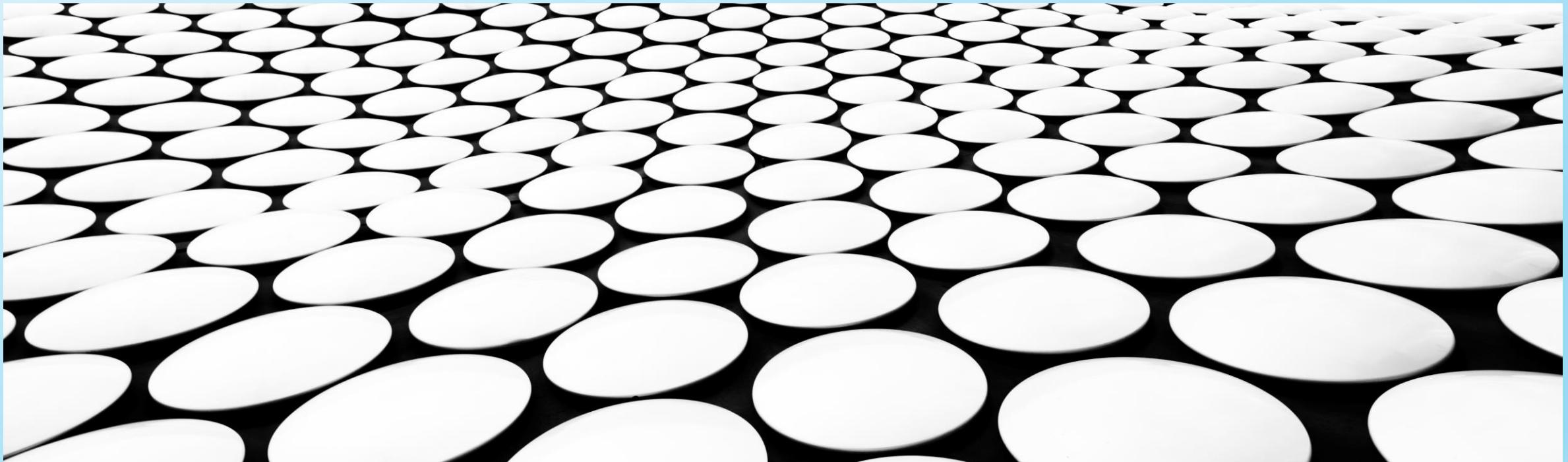


USB 3.0 Shielded Cable



ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



MAGISTRALE I2C

I2C este o magistrală serială, sincronă, multi-master, multi-slave, cu un singur capăt

Este o magistrală de banda îngusta și de distanță mică.

A fost inventată la Philips Semiconductor (acum NXP Semiconductors).

Este de obicei folosit pentru atașarea circuitelor integrate periferice cu viteză redusă la procesoare și microcontrolere

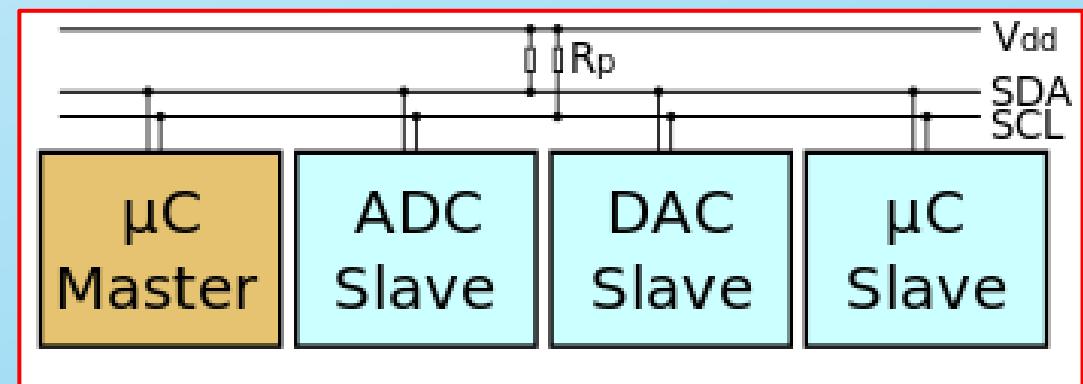
I2C folosește doar două linii bidirectionale,
linia de date în serie (**SDA**=Serial Data Line) și
linia de ceas serial (**SCL**=Serial Clock Line),

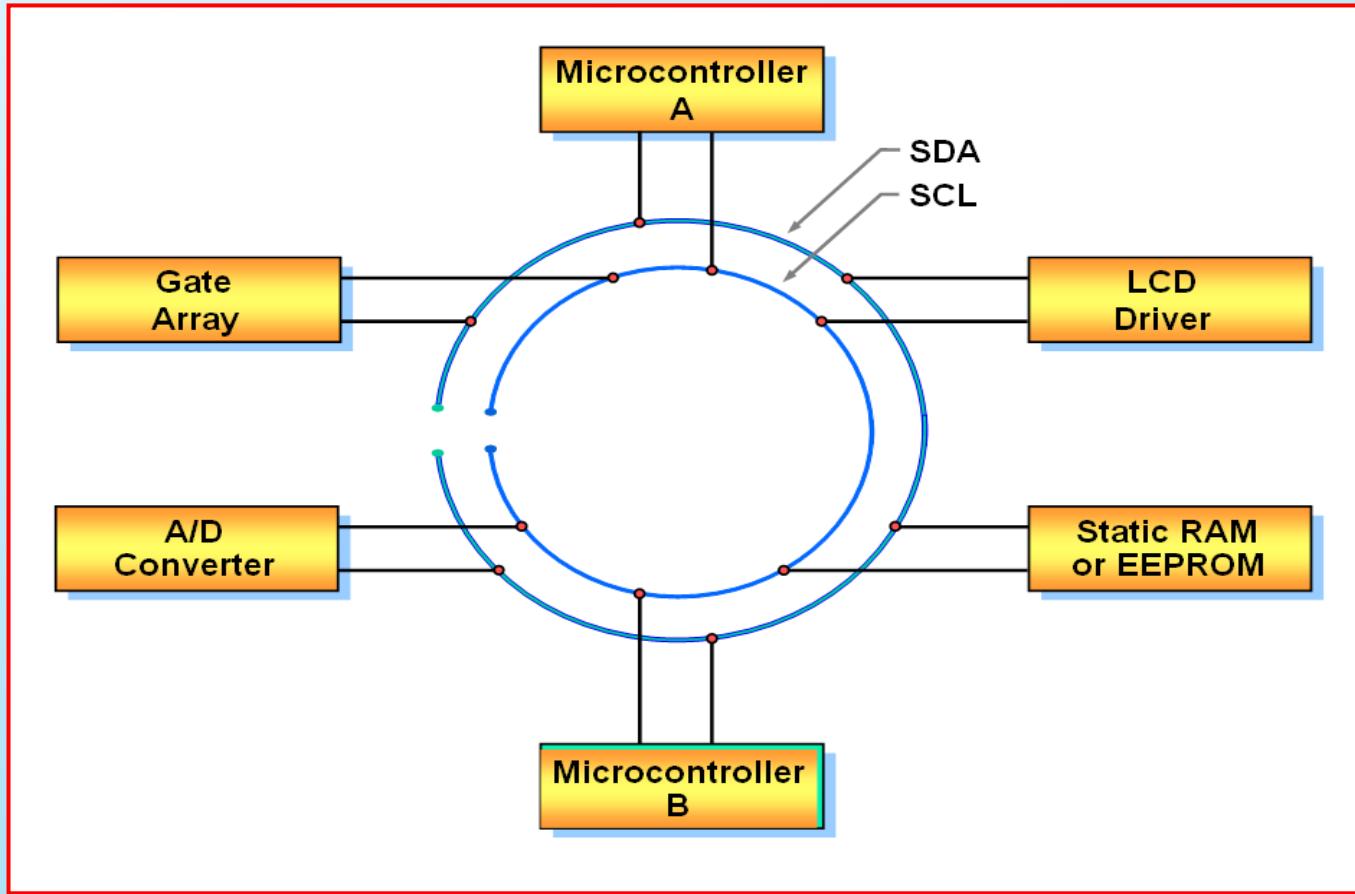
Tensiunile tipice utilizate (Vdd) sunt +5 V sau +3,3 V,
deși sunt premise și sistemele cu alte tensiuni.

Vitezele obișnuite ale magistralei I2C sunt:

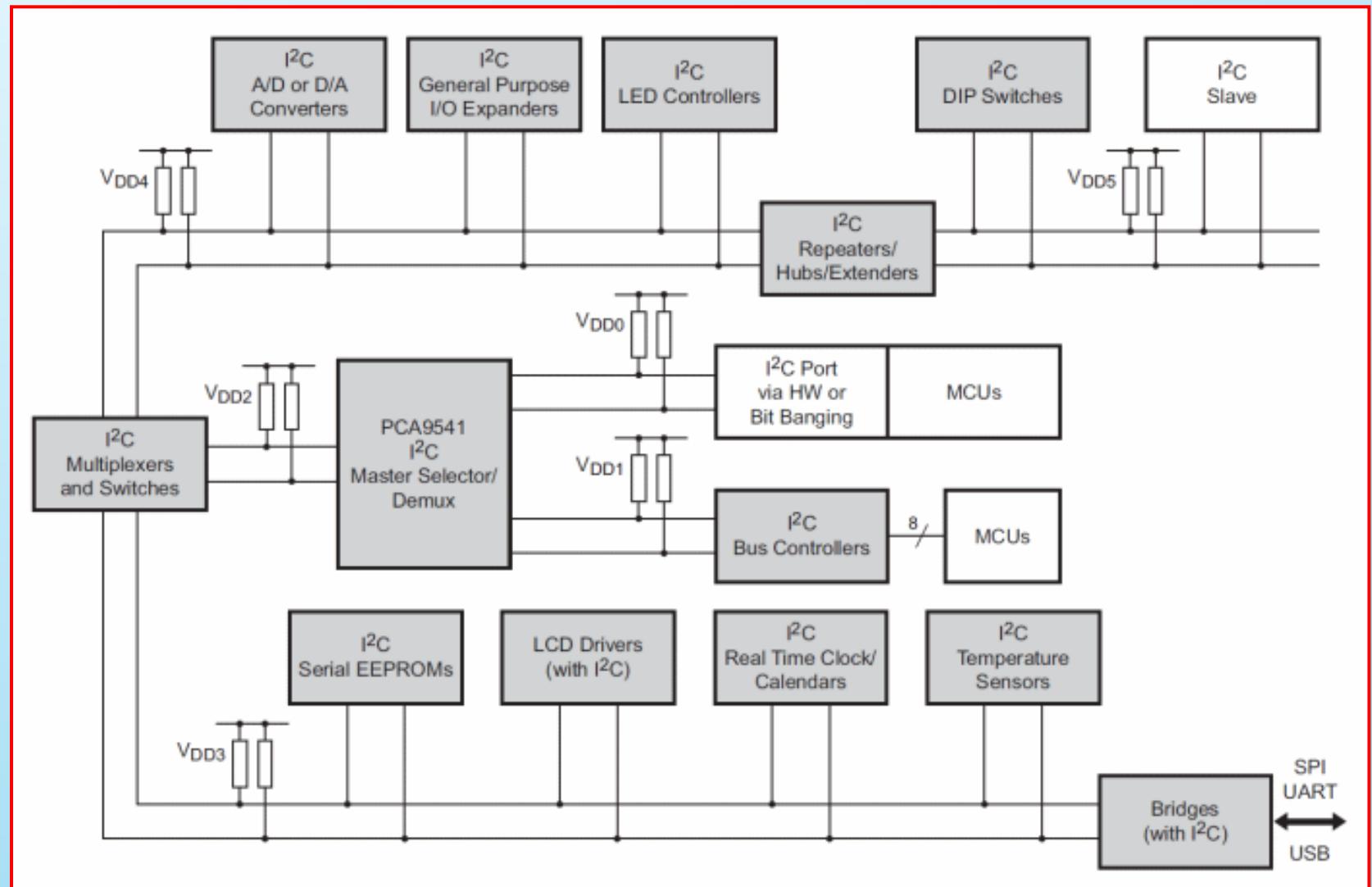
- modul de viteză redusă de 10 kbit/s,
- modul standard de 100 kbit/s
- Modul de viteza mare 3.4 Mbit/s
- sunt permise și frecvențe de ceas arbitrar scăzute.

Topologie simplă





Exemplu de
topologie complexă:



Magistrala I2C folosește două fire: date seriale (SDA) și ceasul serial (SCL).

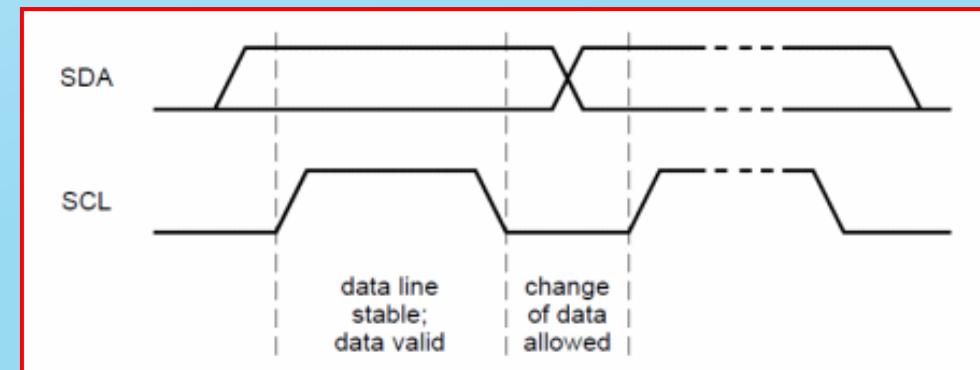
Toate dispozitivele *master* și *slave* I2C sunt conectate doar cu acele două fire.

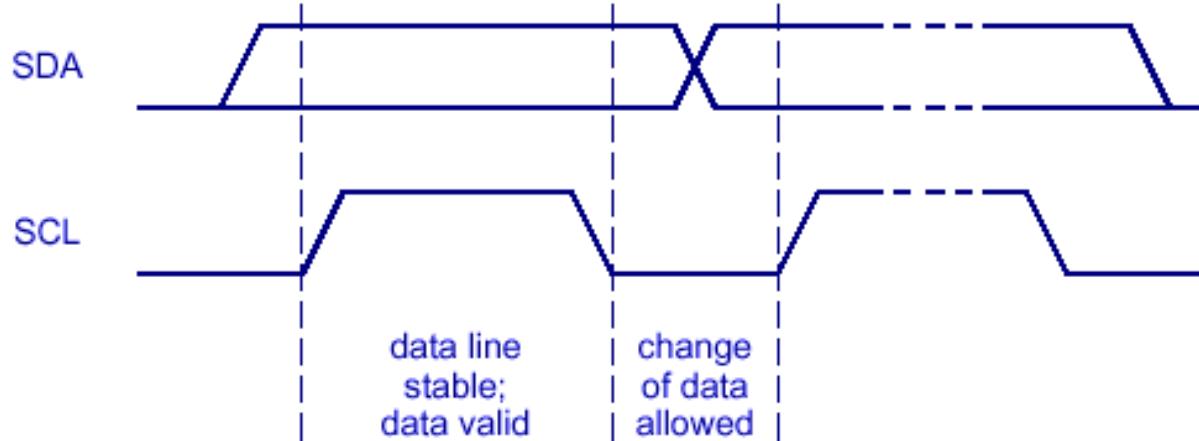
Fiecare dispozitiv poate fi un transmițător, un receptor sau ambele.

- Dispozitivele *master* generează semnalul de ceas și inițiază comunicarea pe magistrală,
- Dispozitivele *slave* răspund la comenzi de pe magistrală.

Pentru a se comunica cu un anumit dispozitiv, fiecare dispozitiv *slave* trebuie să aibă o adresă unică pe magistrală.

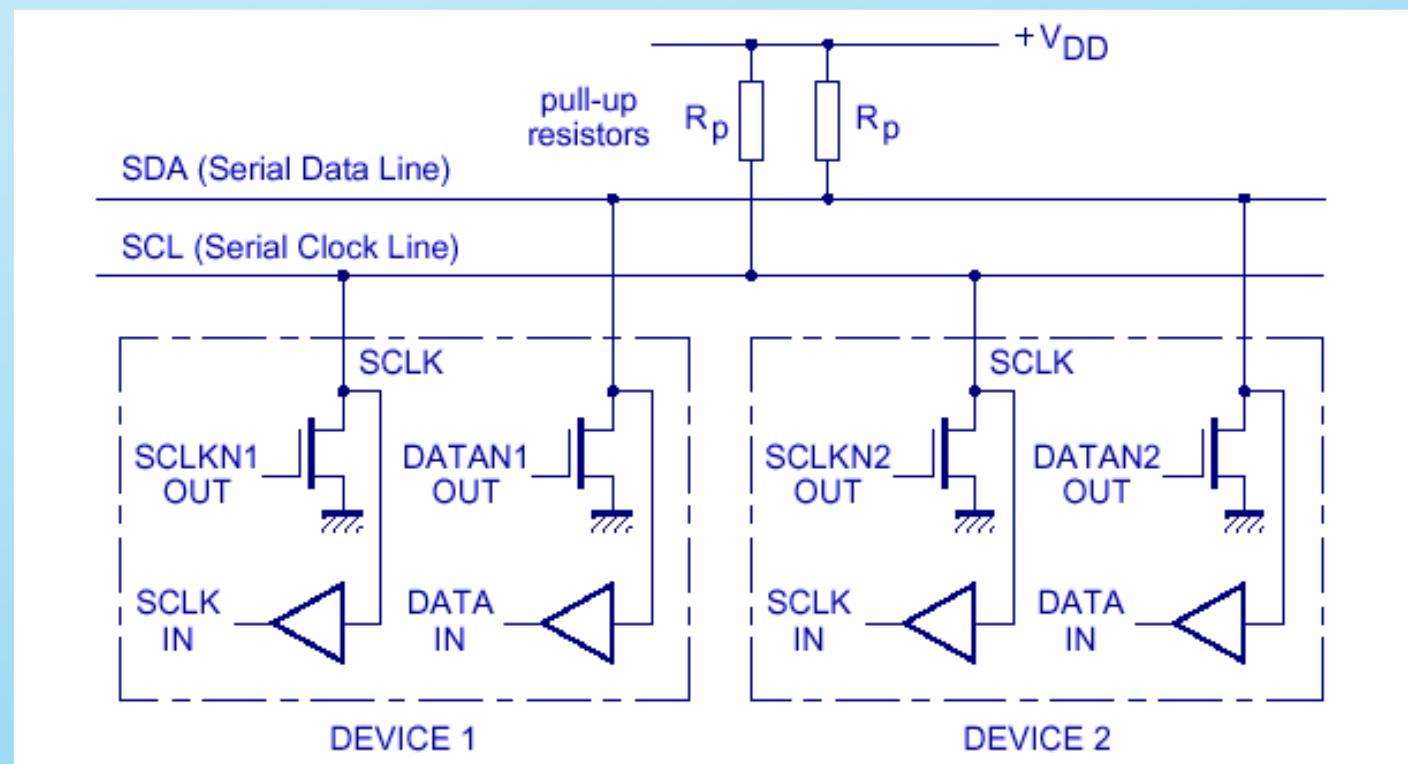
Dispozitivele *master* I2C (de obicei microcontrolere) nu au nevoie de o adresă, deoarece niciun alt dispozitiv (*slave*) nu trimit comenzi către master.





Dispozitivele I2C sunt conectate AND împreună, prin cablu.

Dacă un singur nod scrie zero, întreaga linie este zero



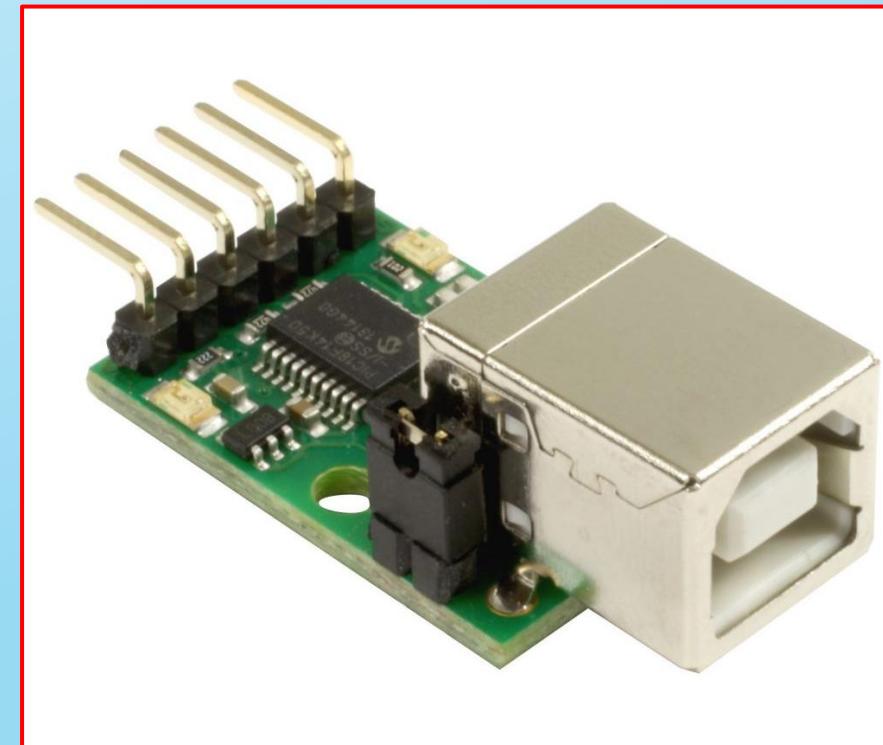
Sisteme de calcul dotate cu interfata I2C (dispozitiv master)

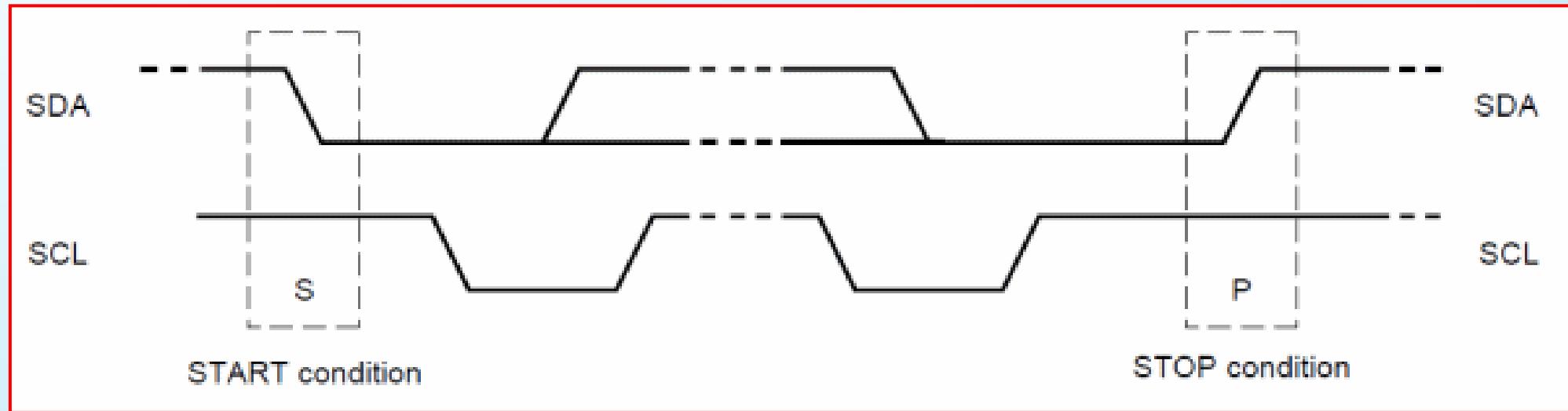
Raspberry PI

Arduino

Sisteme de calcul fara interfata I2C

Adaptor USB-I2C





Condițiile de pornire și oprire

Fiecare comandă I2C inițiată de dispozitivul *master*

începe cu o condiție START și se termină cu o condiție STOP.

Pentru ambele condiții, semnalul SCL trebuie să fie ridicat.

o tranziție **înalt spre scăzut** a semnalului SDA este considerată **START** și
o tranziție **scăzut spre înalt** ca **STOP**.

În ambele cazuri semnalul SCL este generat de *master*.

După condiția de pornire, magistrala este considerată ocupată și poate fi utilizată de un alt *master* numai după ce este detectată o condiție de oprire.

Transferul de date

Este generat un puls de ceas pentru fiecare bit transferat.

Două reguli simple (sincronizare) dictează modul de operare:

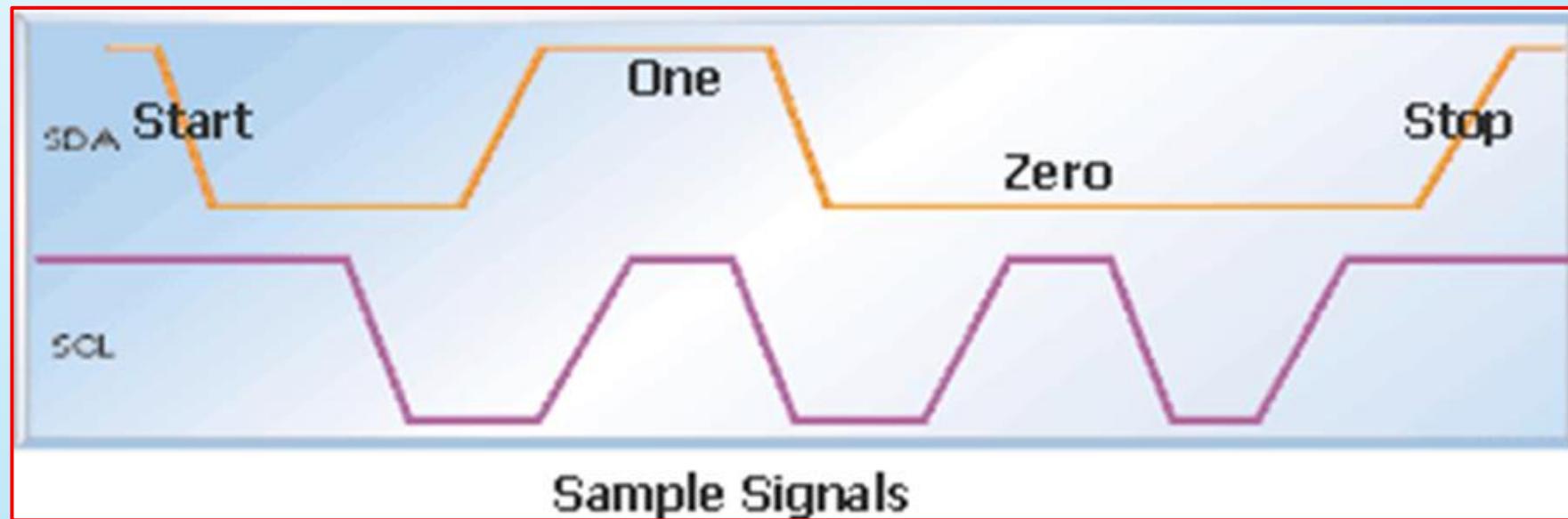
1. Când linia SCL are valoare scăzută starea liniei SDA se poate schimba.
2. Când linia SCL are valoare ridicată, starea liniei SDA indică valoarea unui bit.

Două excepții de la regula 2 creează condiții speciale care sunt folosite pentru a delimita începutul și sfârșitul fiecărei tranzacții între două dispozitive de pe magistrală.

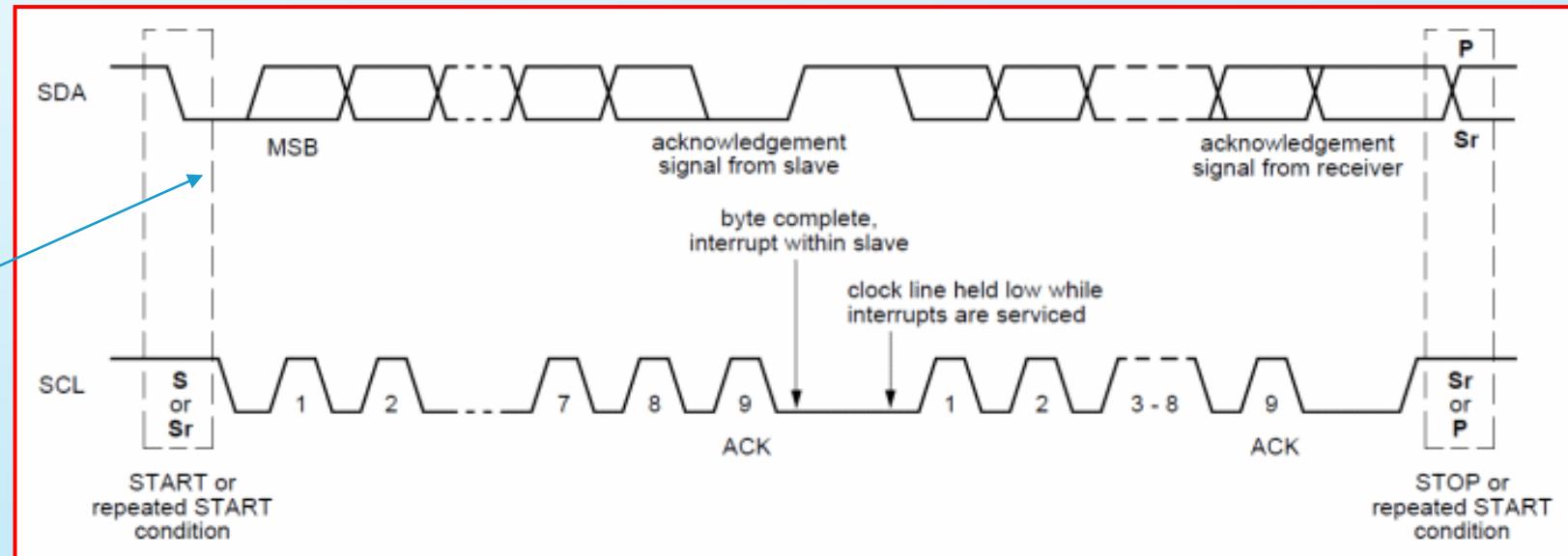
Când SCL este ridicat:

Condiția START este indicată de schimbarea liniei SDA de la ridicat la scăzut

Conditia STOP este indicată de schimbarea liniei SDA de la scăzut la ridicat



Most Significant Bit



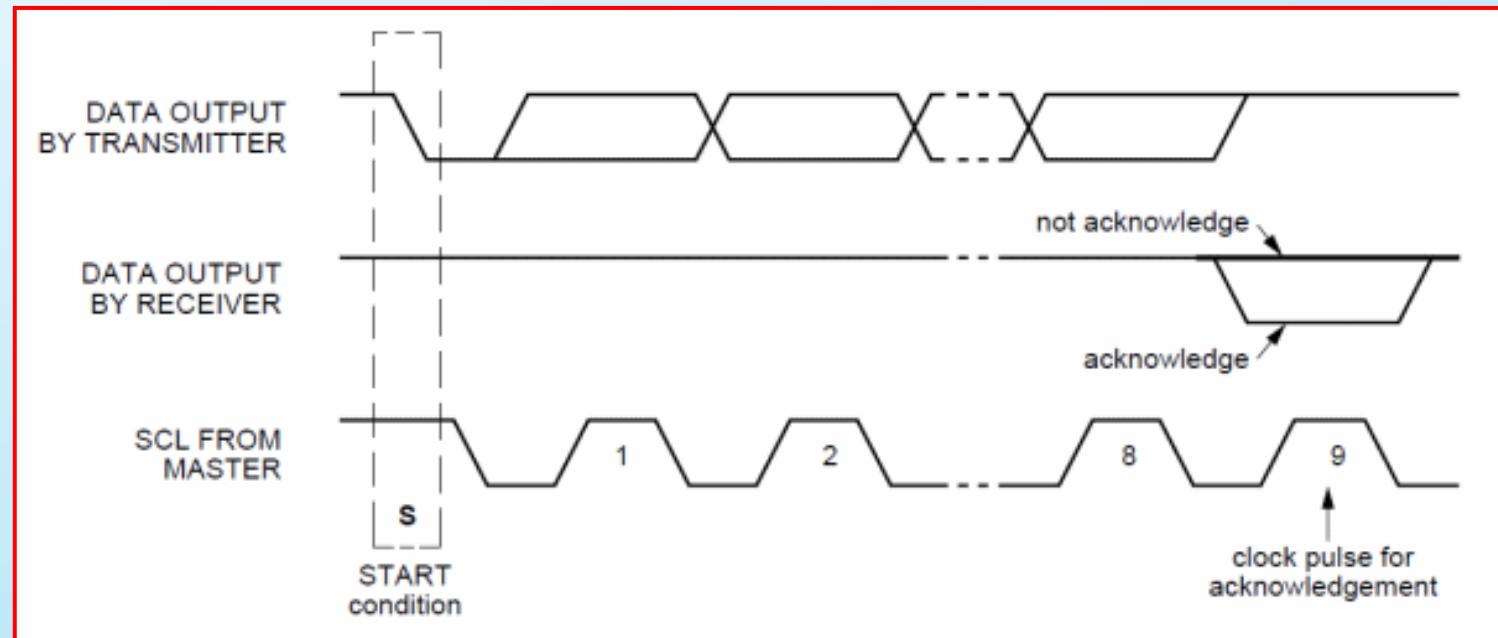
Datele de pe magistrala I2C sunt transferate în pachete de 8 biți (octeți).

Nu există nicio limitare a numărului de octeți, totuși, fiecare octet trebuie să fie urmat de un bit de confirmare. Acest bit semnalează dacă dispozitivul este pregătit să continue cu următorul octet.

Pentru toți biții de date, inclusiv bitul de confirmare, *masterul* trebuie să genereze impulsuri de ceas.

Dacă dispozitivul *slave* nu confirmă transferul, înseamnă că nu mai există date sau că dispozitivul nu este încă pregătit pentru transfer.

Dispozitivul *master* trebuie să genereze starea de oprire sau de pornire repetată.



Confirmarea

La al 9-lea bit recepționat dispozitivul *slave* pune SDA la nivel scăzut.

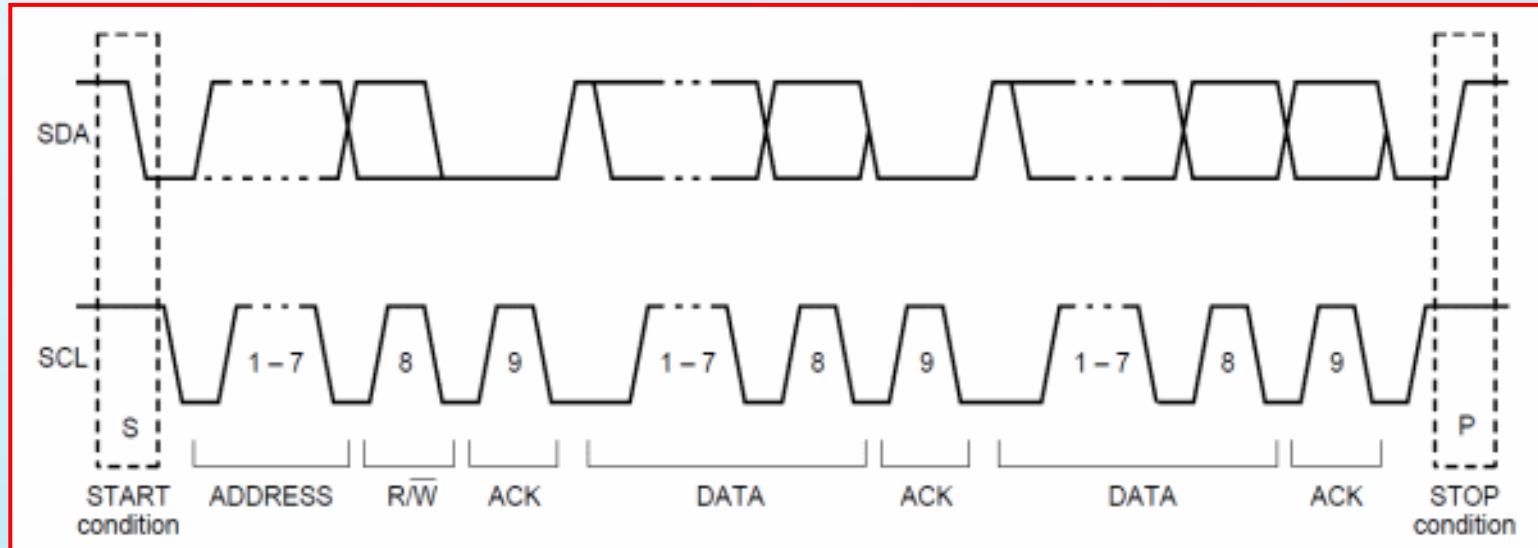
Dacă acest lucru nu se întâmplă *slave* nu a detectat date.

In acest caz trebuie reîncercata transmiterea datelor sau trebuie semnalata o eroare.

Sincronizarea ceasului și handshaking

Dispozitivele *slave* care au nevoie de ceva timp pentru a procesa octetul primit sau care nu sunt încă gata să trimită următorul octet, pot cobori semnalul de ceas pentru a semnala *masterului* că ar trebui să aștepte.

Odată ce semnalul ceasul este eliberat, *masterul* poate continua transmiterea următorului octet.



Comunicarea cu adrese I2C pe 7 biți

Fiecare dispozitiv *slave* de pe magistrală trebuie să aibă o adresă unică de 7 biți. (sau 10 biti)
 Comunicarea începe cu condiția Start, urmată de adresa dispozitivului *slave* pe 7 biți și bitul de direcție a datelor (citire/scriere).

Dacă acest bit este **0**, atunci *masterul* va **scrie** pe dispozitivul *slave*.

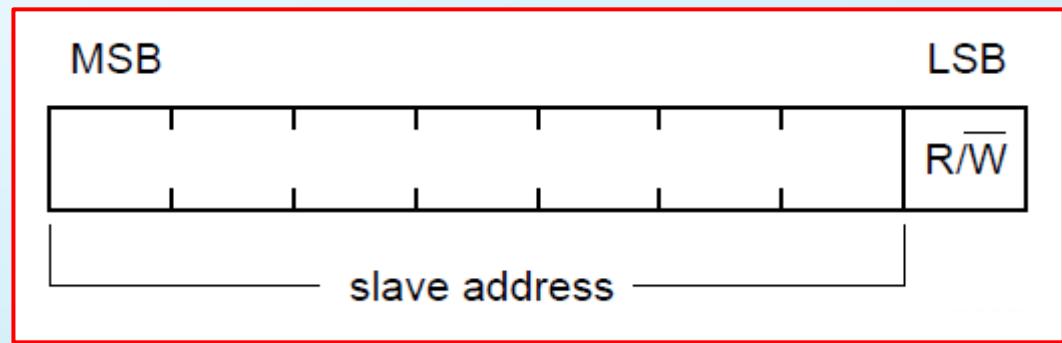
Dacă bitul de direcție a datelor este **1**, *masterul* va **citi** de pe dispozitivul *slave*.

După ce adresa *slave* și direcția datelor sunt trimise, masterul poate continua cu citirea sau scrierea.
 Comunicația se încheie cu condiția Stop care semnalează și că magistrala I2C este liberă.

Dacă masterul trebuie să comunice cu alți *slave*, acesta poate genera o pornire repetată cu o altă adresă *slave* fără generarea stării de oprire.

Toți octetii sunt transferați cu bitul MSB transportat mai întâi.

Adresarea I2C pe 7 biți



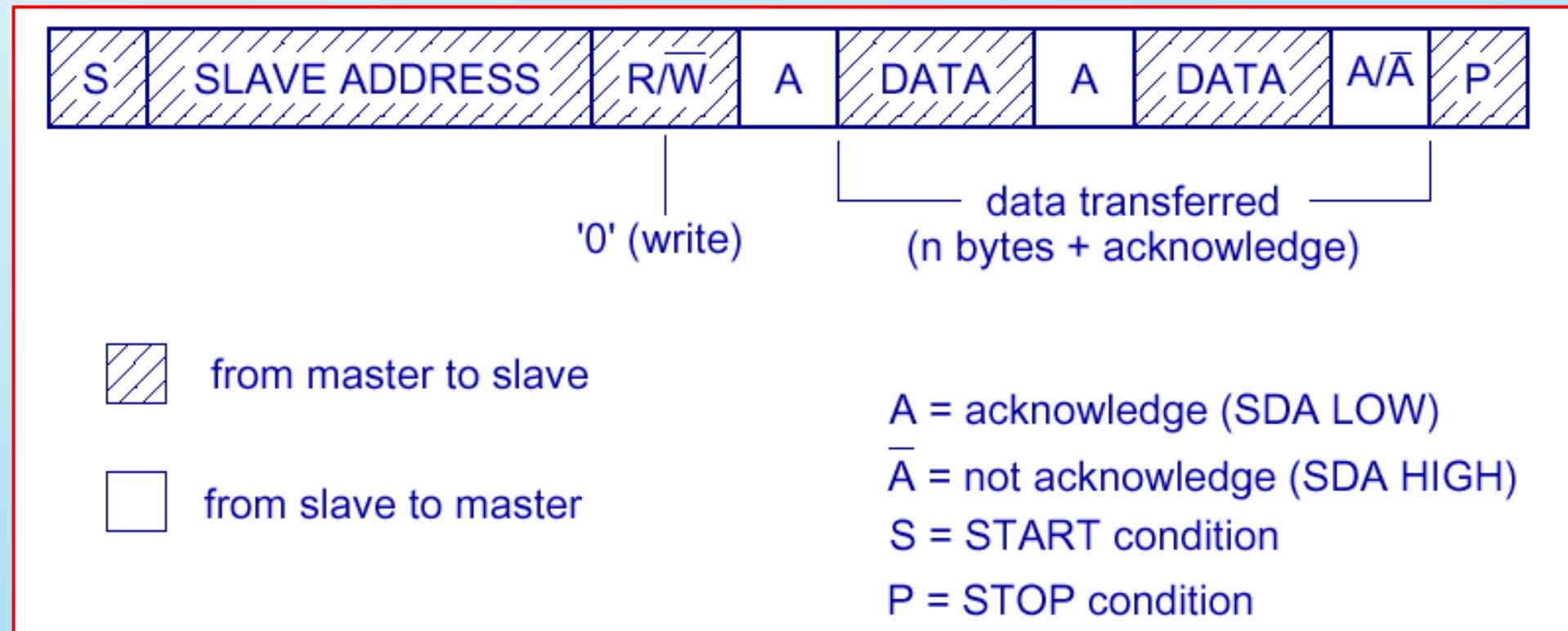
Alocarea adreselor I2C este administrată de **I2C Bus Committee** care se ocupă de alocări.

Două grupuri de 8 adrese I2C sunt rezervate pentru utilizări viitoare și o adresă este utilizată pentru adresarea I2C pe 10 biți.

SLAVE ADDRESS	R/W BIT	DESCRIPTION
0000 000	0	General call address
0000 000	1	START byte
0000 001	X	CBUS address
0000 010	X	Reserved for different bus format
0000 011	X	Reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	X	Reserved for future purposes
1111 0XX	X	10-bit slave addressing

Transfer de date de la Master-Transmitter la Slave-Receiver

Direcția de transmisie nu se schimbă niciodată.
Configurarea și transferul sunt simple

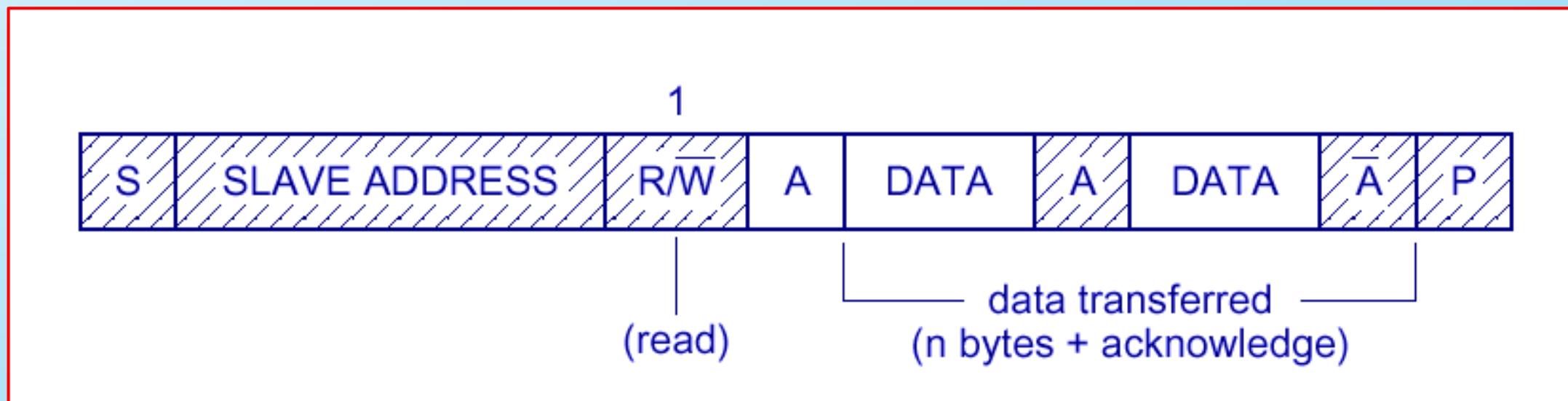


Transfer de date *Master-Receiver* și *Slave-Transmitter*

Masterul inițiază transferul de date prin generarea condiției START, urmată de octetul de pornire (cu bitul de citire/scriere setat la 1, adică modul de citire)

După prima confirmare de la slave, direcția datelor se schimbă și *masterul* devine receptor iar transmițător *slave*.

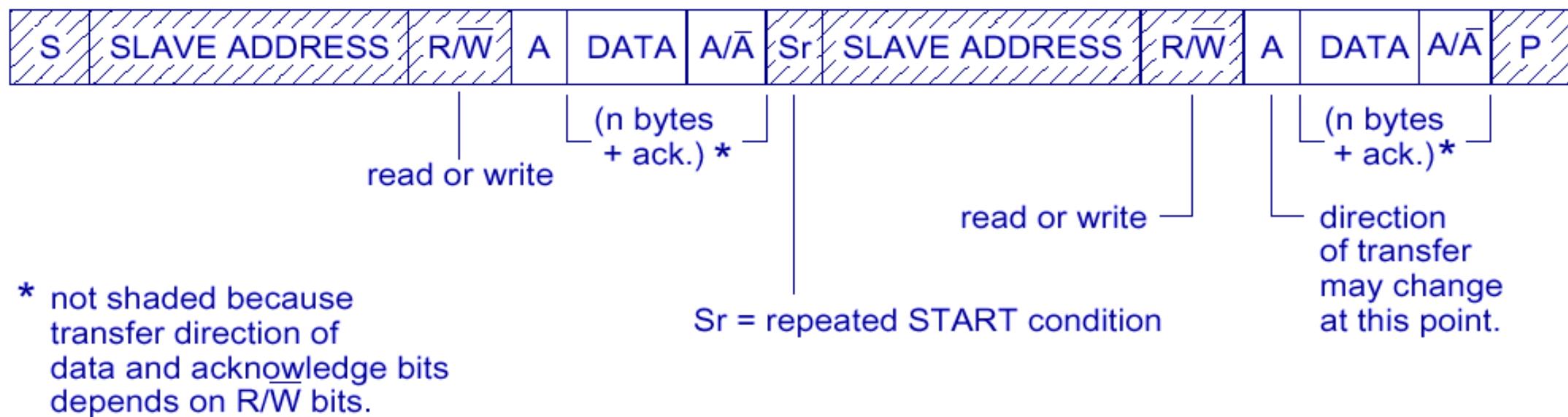
Condiția STOP este încă generată de *master* (*masterul* trimite not-ACK înainte de a genera STOP)



Citire și scriere în același transfer de date

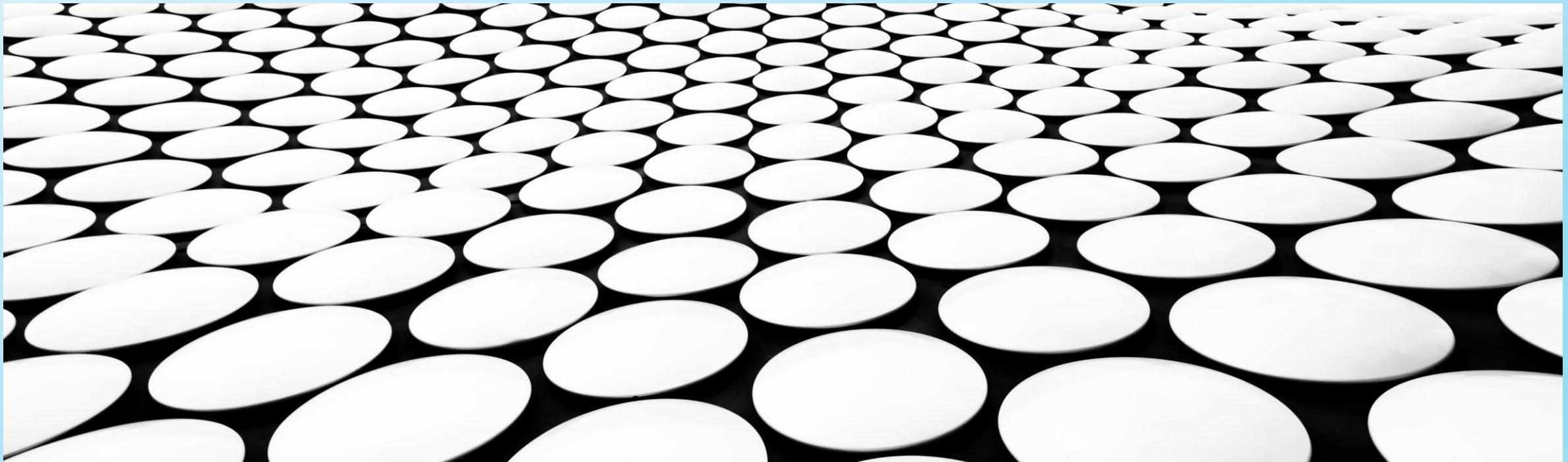
Schimbarea direcției transferului de date poate avea loc prin generarea de către *master* a unei alte condiții de START (numită condiție de START repetată) cu adresa *slave* repetată

Dacă *masterul* a fost un receptor înainte de schimbare, atunci *masterul* trimite un not-ack (A') înainte de condiția START repetată



ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



UNITĂȚI DE MEMORIE

MEMORIE = dispozitiv (bloc, subsistem) de stocare a informației

Caracteristici

- capacitate
- viteza
- tip de acces
- volatilitate
- tehnologie
- portabilitate
- pret

Clasificari dupa caracteristici

Clasificare dupa tipul de acces

- ROM (Read-Only Memories)
 - Pot fi numai citite
 - Conținutul nu poate fi alterat
 - Ex.: electronice; optice (CD/DVD)
- RAM (Random-Access Memories)
 - Memoriile pot fi citite, scrise, sterse
 - sunt cu acces direct (spre deosebire de cele cu acces secvential)
 - Ex.: electronice, magnetice (HDD)
- SAM (Sequential Access Memory)
 - Ex: benzi magnetice

Clasificare dupa volatilitate

- Volatile
 - Informația dispare la intreruperea alimentarii cu energie electrică
 - Ex.: memorii electronice
- Permanente (nevolatile)
 - Informația se pastrează și după intreruperea alimentarii
 - Ex.: memorii magnetice, memorii optice

Clasificare după tehnologie

- Memorii magnetice
- Memorii optice
- Memorii electronice
- Memorii cuantice

CATEGORII DE MEMORII

Memoria primara

- este accesata direct de CPU
- poate fi atat RAM cat si ROM
- stocheaza instructiuni si date cu care se lucreaza in mod curent
- este mai rapida

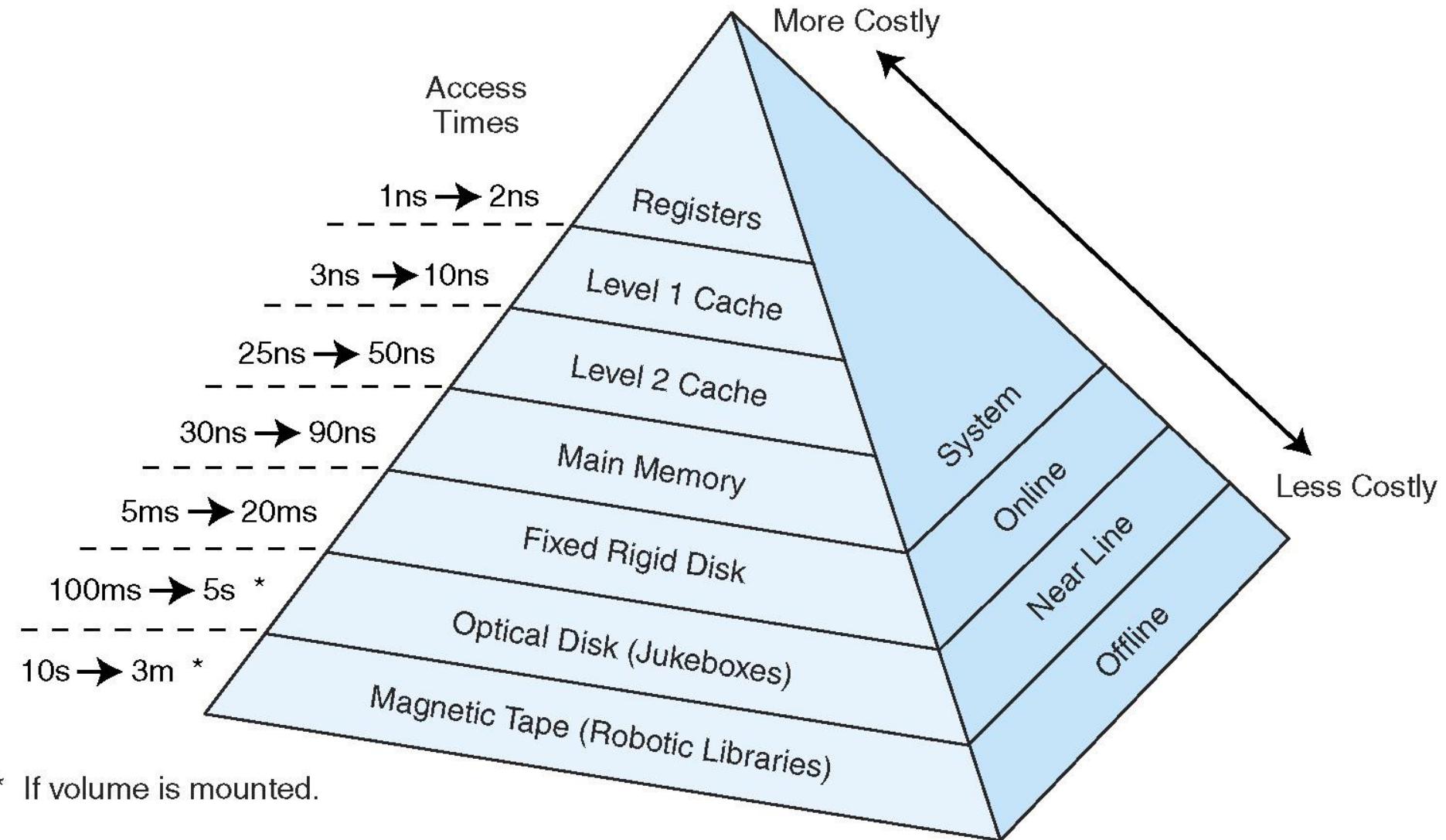
Memoria secundara

- accesul este intermediat de blocuri specialize de I/O
- este memorie permanenta
- este mai lenta

Hierarhia memoriei



Memoriile mai rapide
sunt mai scumpe
(pretul pe bit)



Organizarea memoriei

- Este divizata in locatii
- Locatia are dimensiune fixa: 1,2,4,8,16,32 biti, etc
 - Fiecare locatie are o adresa unica
 - La un moment dat o singura locatie este **selectata**
 - (pentru a fi citita sau scrisa)
 - Pot fi organizate liniar sau matriceal
 - In cazul matriceal, adresa este formata din doua componente concatenate
 - Capacitatea: numarul total de locatii
 - Spatiul de adresare: multimea adreselor locatiilor

- Frecvent memoria este imparțita in pagini.
 - Pagina este in bloc de memorie de lungime fixa ce contine 256-4096 locatii (2^8 - 2^{12})

Memorii magnetice

Memorii cu banda magnetica



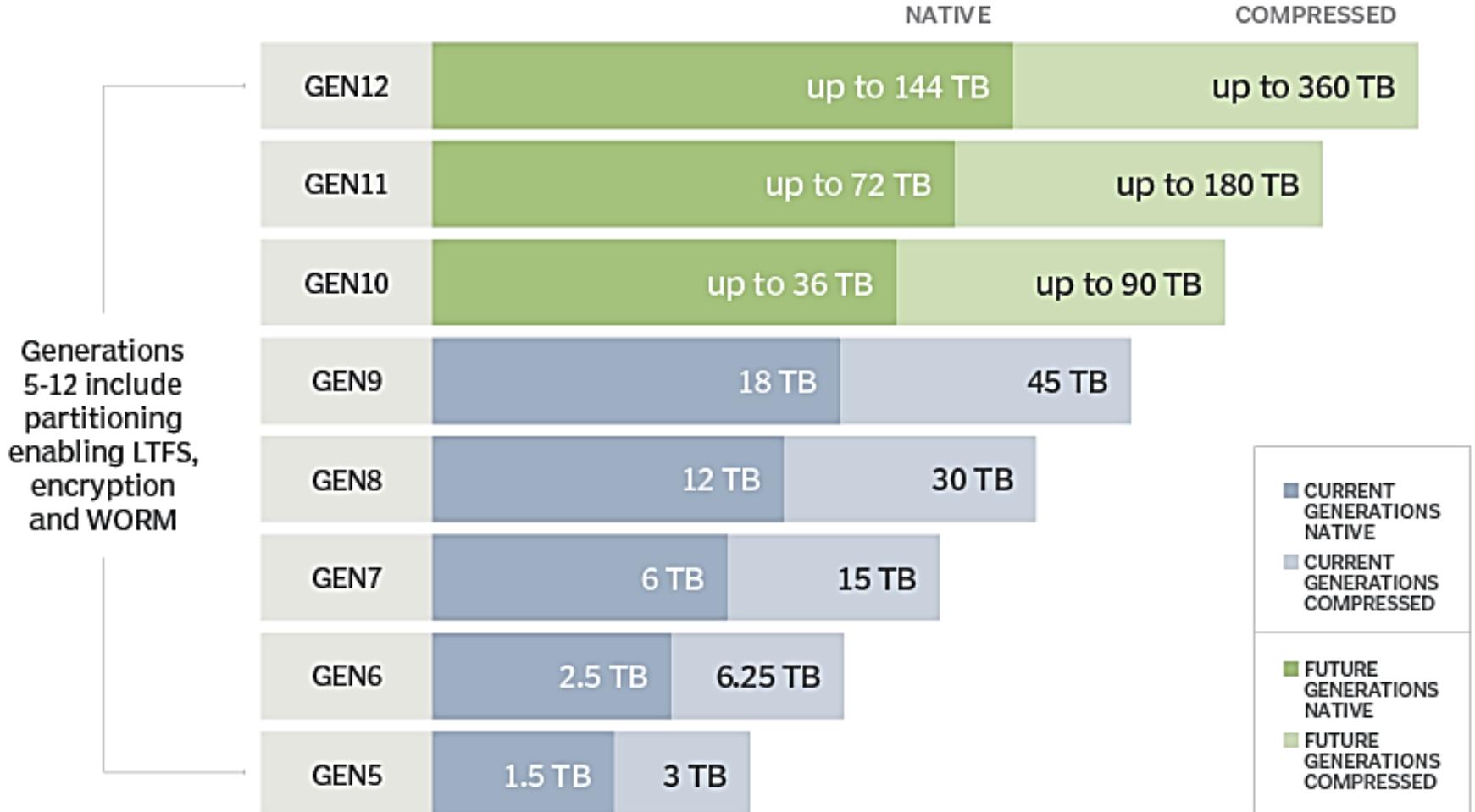
Memorie cu acces secvential

Tehnologia de inregistrare pe banda magnetica **LTO** (Linear Tape Open) asigura:
cel mai bun raport intre capacitatea stocata, performante si fiabilitate
la un pret per TB superior altor tehnologii.

2 tipuri de operare:

- Single-drive mode (de sine statatoare)
 - Library-mode (in automatizari cu roboti de arhiva)
-
- Generatia **LTO-7** ofera 6TB de stocare per caseta in regim necompresat,
respectiv 15TB de stocare (folosind compresie 2.5 : 1)
fiind echivalenta cu stocarea a pana la **800 de filme de rezolutie HD**.

LTO Ultrium roadmap



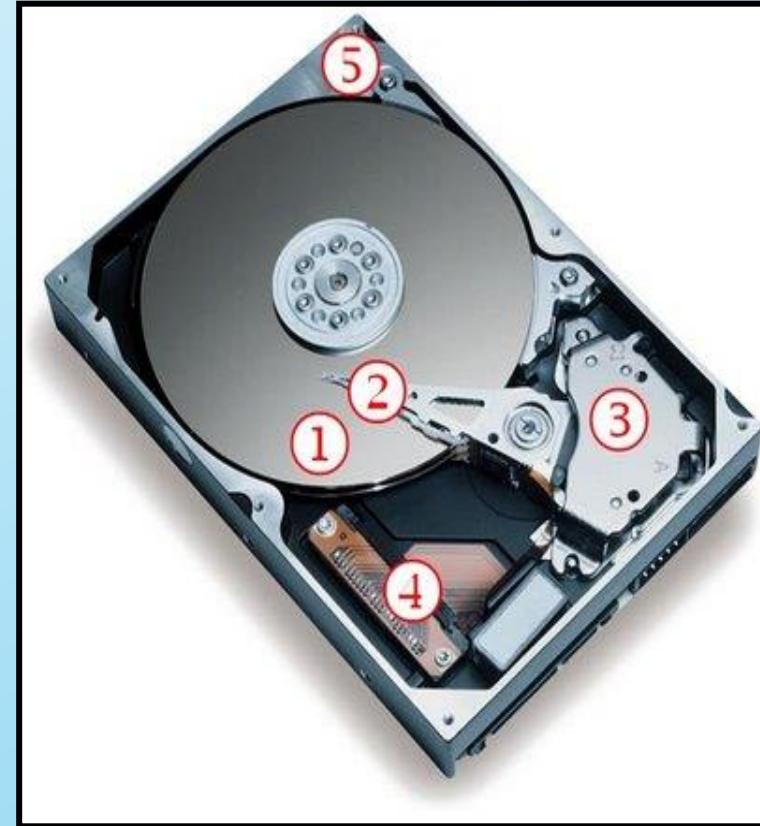
Memorii cu disc magnetic

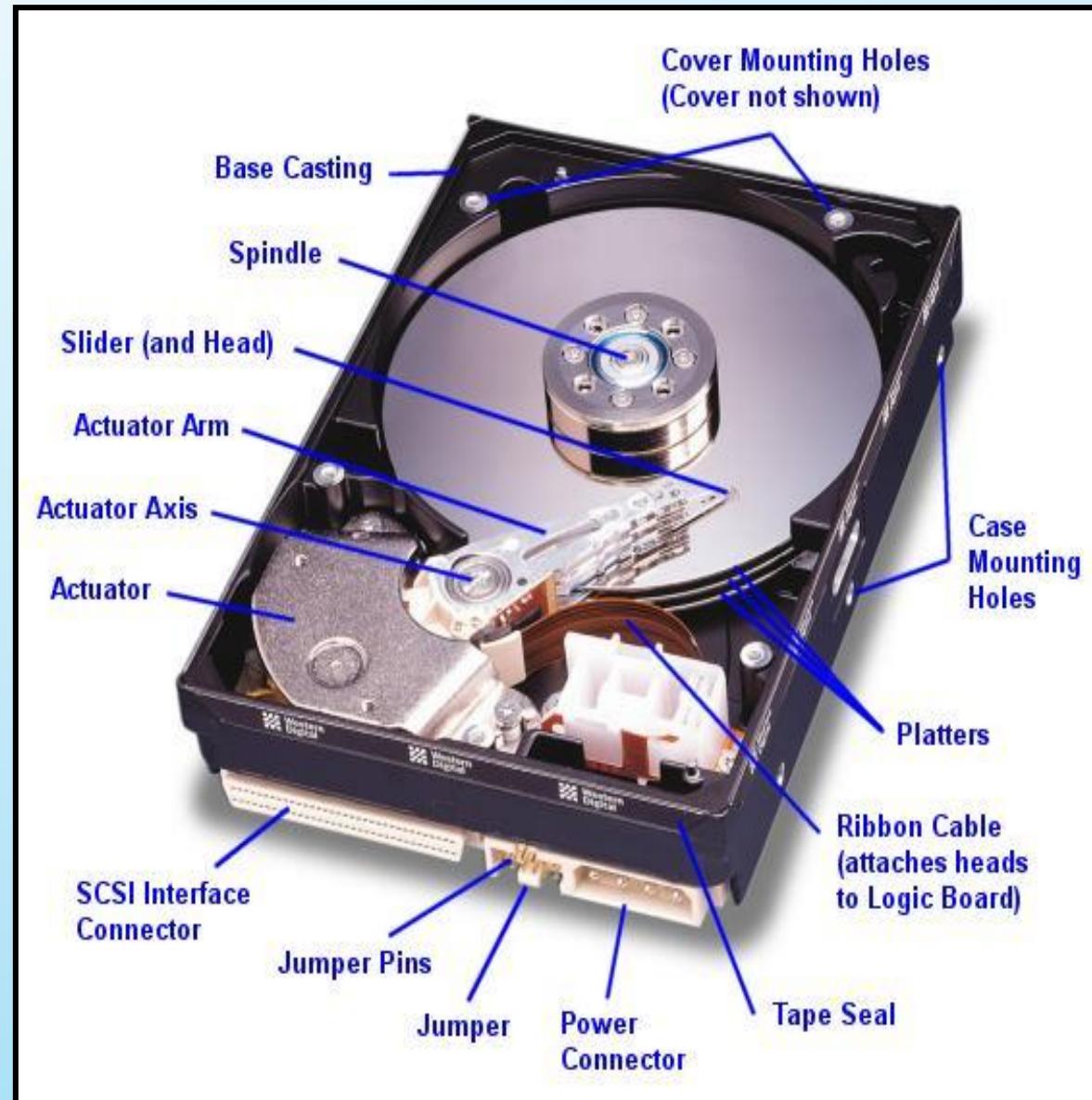
HDD = Hard Disk Drive

Unitate de memorie cu disc rigid

Componente de bază ale unității de hard-disk

1. Platanele (discurile)
2. Capetele de citire, respectiv de scriere
3. Dispozitivul de acționare a capului
4. Motorul de antrenare (al platanului)
5. Placa logica
6. Cabluri și conexiuni
7. Elemente de configurare (precum jumpere sau comutatoare).





Structura fizică a unui hard-disk constă din discuri rotative cu capete care se mișcă deasupra suprafețele lor și stochează date pe piste și sectoare.

O unitate de hard-disk conține platane rigide, în formă de disc, confecționate de obicei din aluminiu sau sticlă.

Pe platan este depus, sub forma de strat subțire, un material magnetic.

Platanele nu se pot curba sau îndoi.
În majoritatea unităților de hard-disk discurile nu se pot extrage, din acest motiv se numesc *unități cu disc fix*.



Structura internă

Motor liniar electrodinamic cu magneti permanenți în stator și bobină parcursă de curent ca rotor

Brațul port capete

Discuri de aluminiu acoperite cu o peliculă feromagnetică

Filtru pentru curățirea aerului antrenat de discuri

Carcasă antivibrării

Axul motorului care antrenează disurile în rotație prin CAV (Constant Angular Velocity)

Cablu panglică flexibila pentru transmiterea datelor citite de la capete la placa electronică. O primă prelucrare analogică se face chiar pe panglică.

Placa electronică (nu se vede din acest punct de vedere)

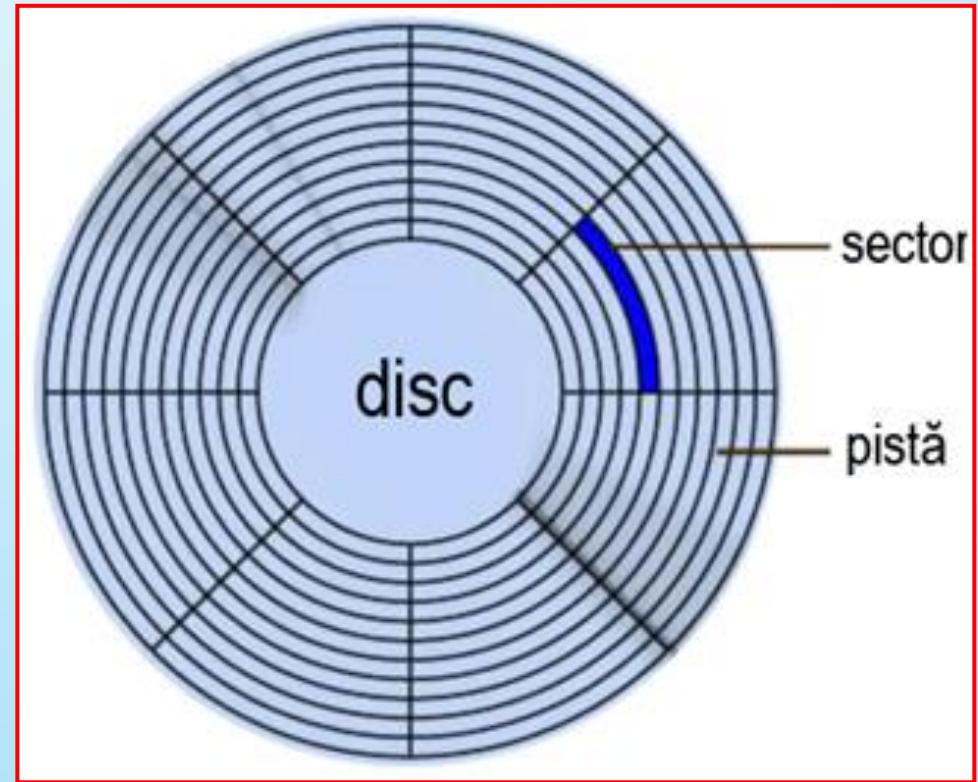
**Majoritatea unităților au turații de 5400, 5600, 6400, 7200 rot/min,
dar unele unitati ating 15000 de rot/min.**

Unitățile de hard-disk au de obicei mai multe discuri (platane) care sunt amplasate unul deasupra celuilalt și se rotesc solidar, fiecare avand două fețe, pe care unitatea stocheaza datele.





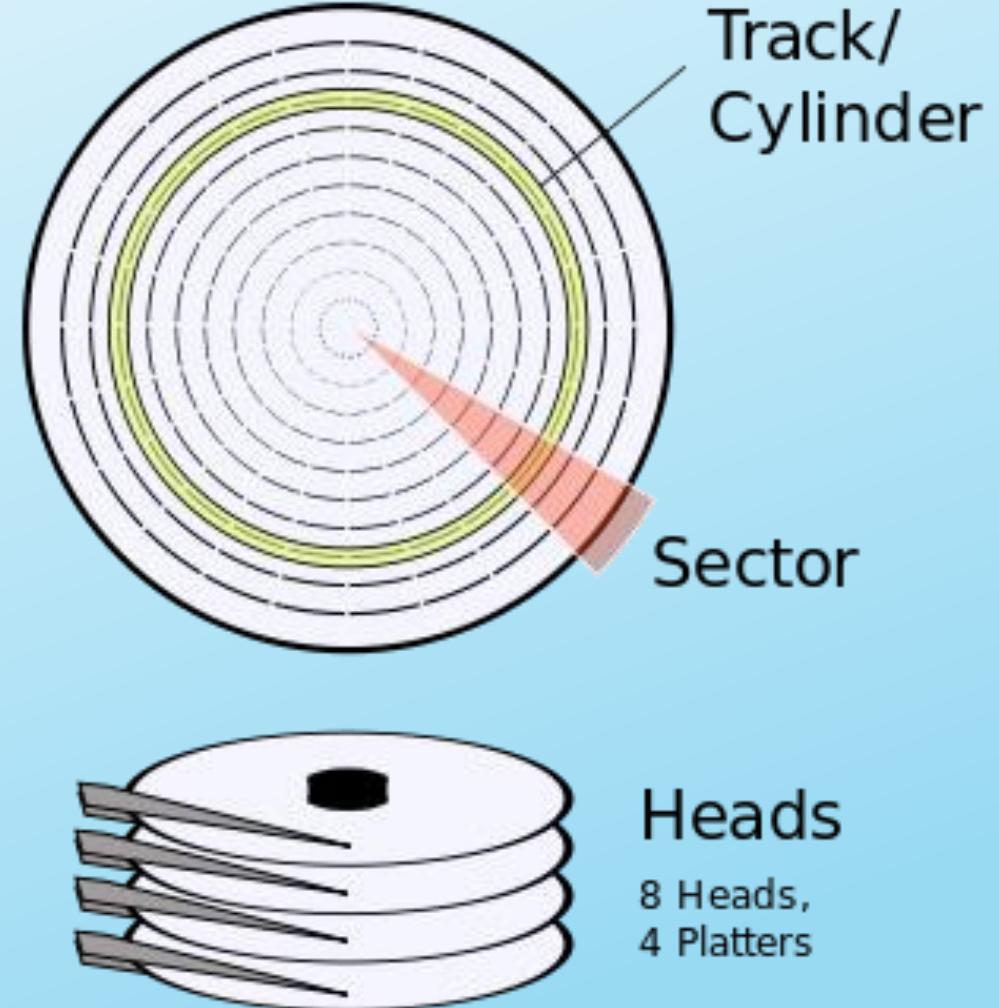
*Capetele citesc și scriu date
pe inele concentrice numite piste,
care sunt divizate în
segmente numite septoare,
conținând de obicei 512 octeți fiecare .*

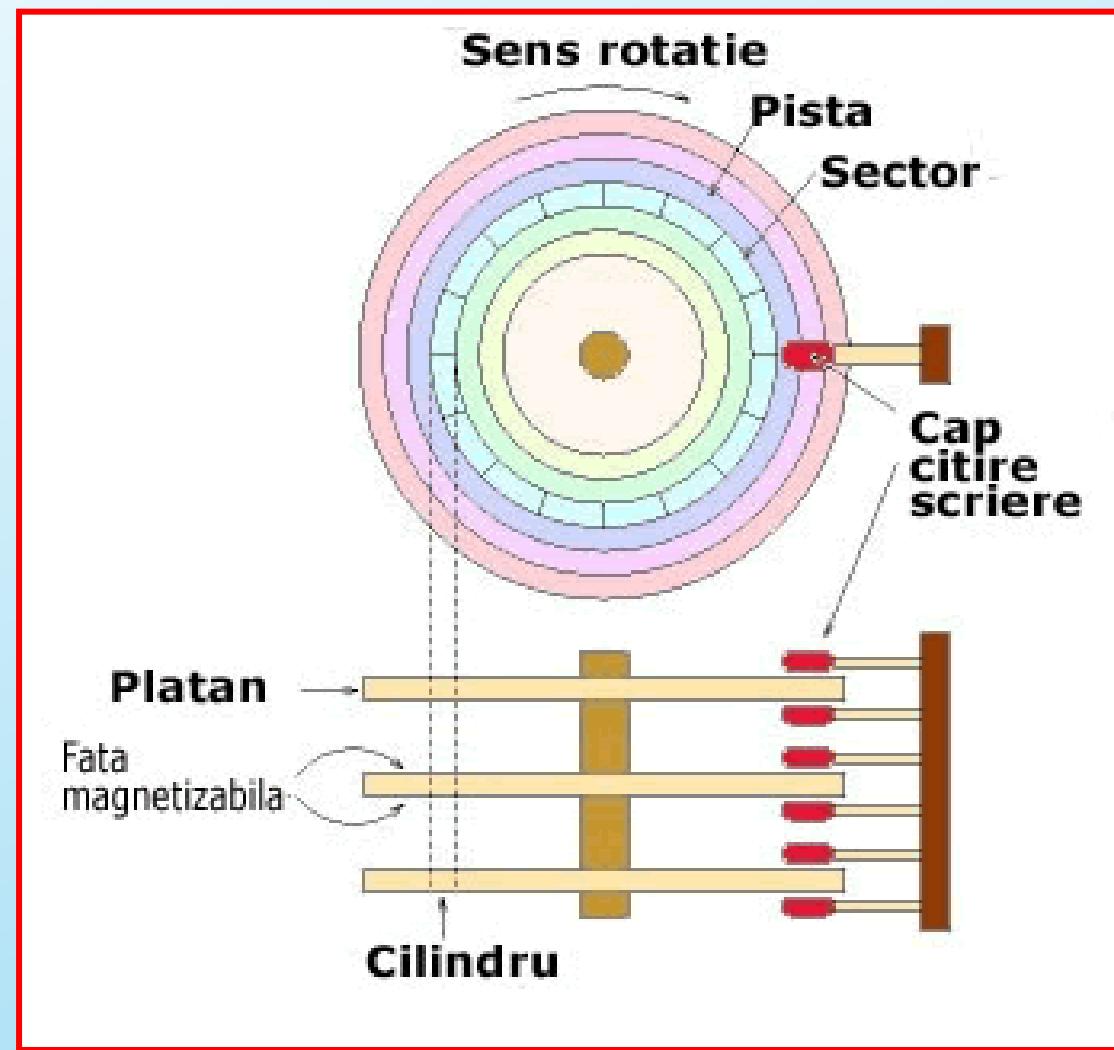


Desi sunt considerate memorii cu acces direct,
cantitatea minima de informatie care poate fi citita/scrisa
este cea inmagazinata intr-un sector (adica mai multe locatii de memorie).

Majoritatea unităților au două sau trei platane care dau patru sau șase fețe, dar unele unități au până la 11 sau mai multe platane.

Pistele aflate pe aceeași poziție, de pe fiecare față a fiecarui platan, luate împreună, alcătuiesc **un cilindru**.





Dimensiunea fizică a unei unități este exprimată prin dimensiunea platanelor (discurilor):

- **5 $\frac{1}{4}$ inch (130 mm)**
- **3 $\frac{1}{2}$ inch (95 mm)**
- **2 $\frac{1}{2}$ inch (63.5 mm)**
- **1.8 inch (45.7 mm)**



HDD 3.5"



HDD 2.5"



HDD 1.8"



Suporturi de înregistrare

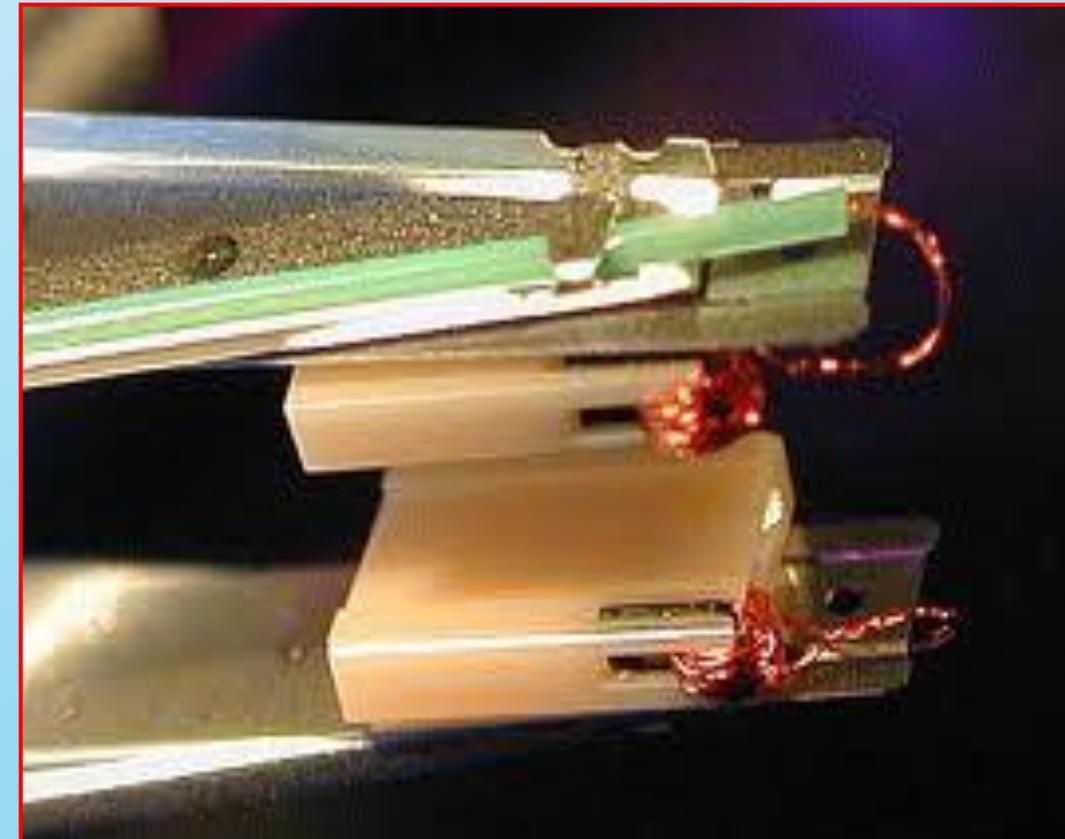
Indiferent de substratul folosit,
platanele sunt acoperite cu un strat subțire de substanță magnetică,
depusă uniform, numit suport, pe care se stochează informația.

Capetele de citire/scriere.

O unitate de hard-disc are câte un cap de citire/scriere pentru fiecare față de platan.

Aceste capete sunt conectate, sau *solidare cu* același mecanism de deplasare.

Fiecare cap se află pe un braț al dispozitivului de acționare, braț acționat de un resort pentru a presa capul pe platan.



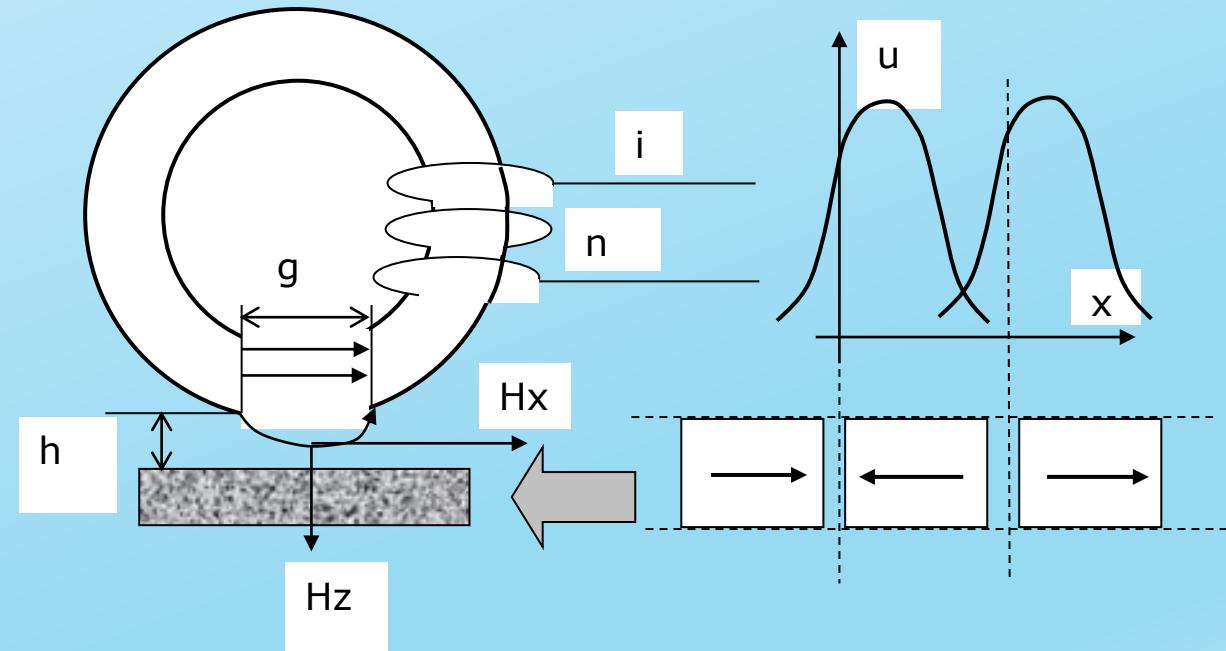
Principiul înregistrării magnetice

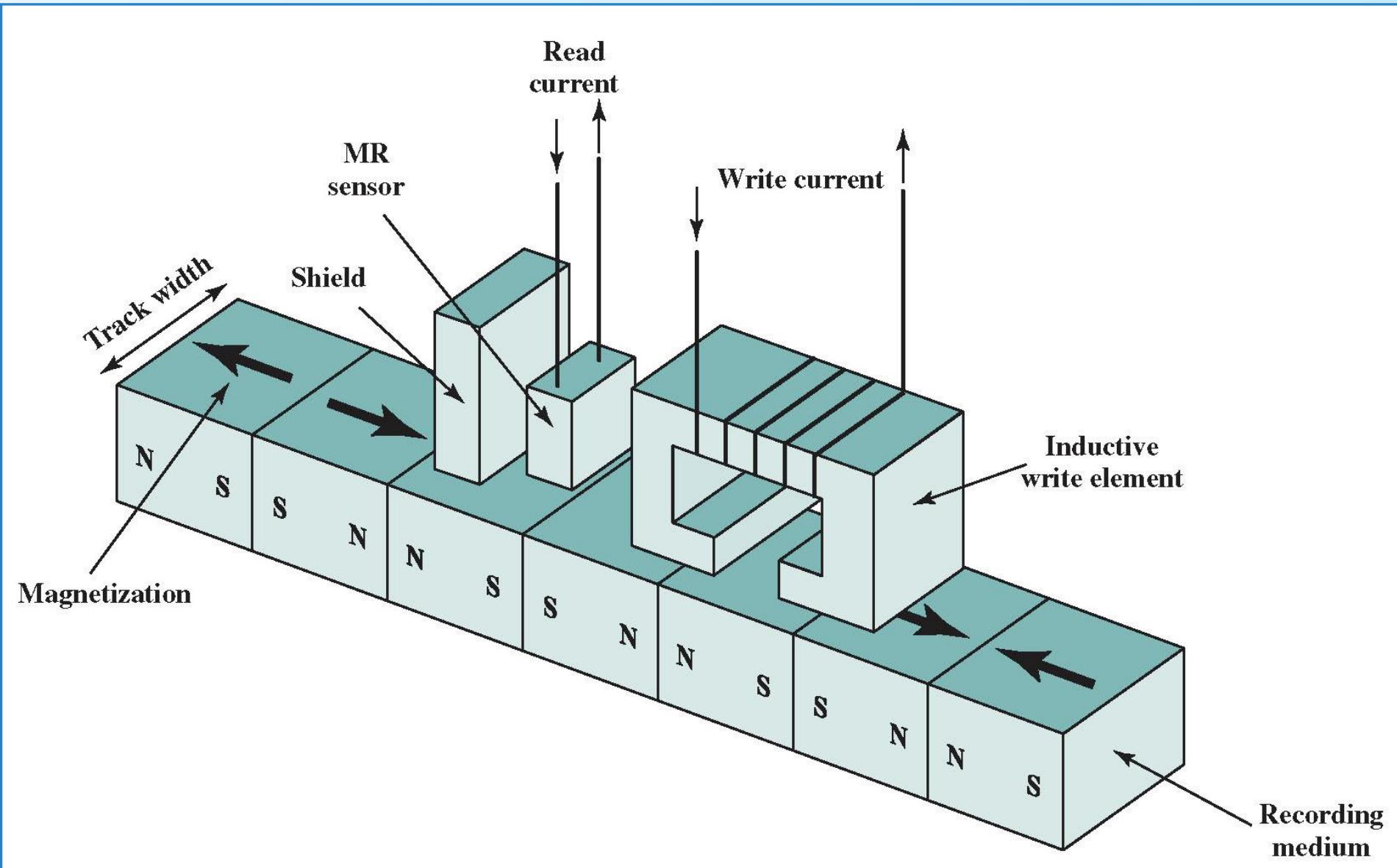
Capul de scriere/citire este un inel cu un întrefier și o înfășurare cu n spire străbătută de curentul i care crează un flux magnetic în vecinătatea spațiului interolar și care determină o magnetizare a stratului magnetic al mediului de stocare.

Scrierea este realizată prin fluxul de dispersie și nu de fluxul prin întrefier. Cu cât întrefierul g este mai mic cu atât densitatea de scriere poate fi mai mare.

Câmpul de dispersie trebuie să fie mai intens decât câmpul coerciv al suportului de înregistrare, iar câmpul în întrefier trebuie să fie mai mic decât câmpul de saturatie.

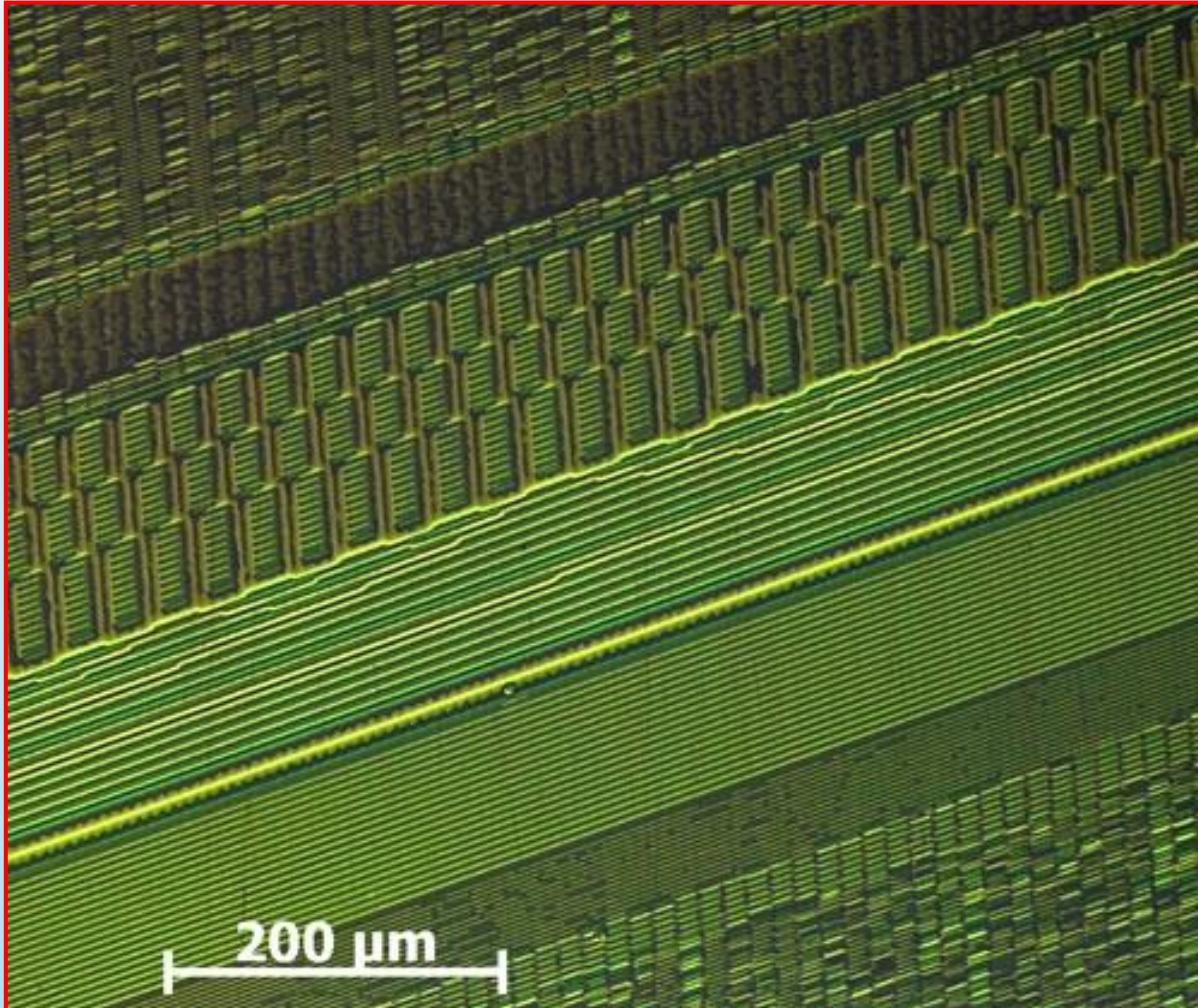
Influența fluxului de dispersie asupra suportului poate fi mărită prin micșorarea distanței între cap și mediu h .





Modele de capete de citire/scriere:

- Cu ferită
- Peliculare (Thin Film)
- Cu întrefier metalizat (MIG— Metal-In-Gap)
- **Magneto-rezistive (MR)**



Capetele pot fi:

- **În contact, la viteza între 0,5-3,5m/s**
- **Plutitoare, la viteza mai mare de 3,5m/s. Se asigură distanțe între cap și suport mai mari de $0,2\mu m$.**

La capetele plutitoare, stratul de aer antrenat de discul rigid în rotație, care se scurge pe profilul de plutire al capului, crează o forță aerodinamică care îndepărtează capul de disc.

Forța elastică a brațului apasă capul spre disc și astfel, la echilibru se obține o distanță constantă între disc și cap.

Dispozitivul de acționare a capului.

Acum dispozitivul deplasează capetele pe deasupra discului și le poziționează cu precizie deasupra cilindrului dorit.

Dispozitivul de acționare a capului determină cele mai importante caracteristici ale unei unități (caracteristici de performanță și fiabilitate).

Motoare de antrenare pentru platane

Motorul care rotește platanele este numit **motor de antrenare**, pentru că este conectat la axul în jurul căruia se rotesc platanele.

Motorul de antrenare trebuie să aibă viteza precis controlată. Platanele din unitățile de hard-disc se rotesc cu viteze între 3.600 și 10.000 rot/min sau mai mult, iar motorul are un circuit de control cu o buclă de feedback pentru a urmări și a controla precis această viteză.

Plăcile logice

Plăcile logice conțin circuite electronice care controlează:

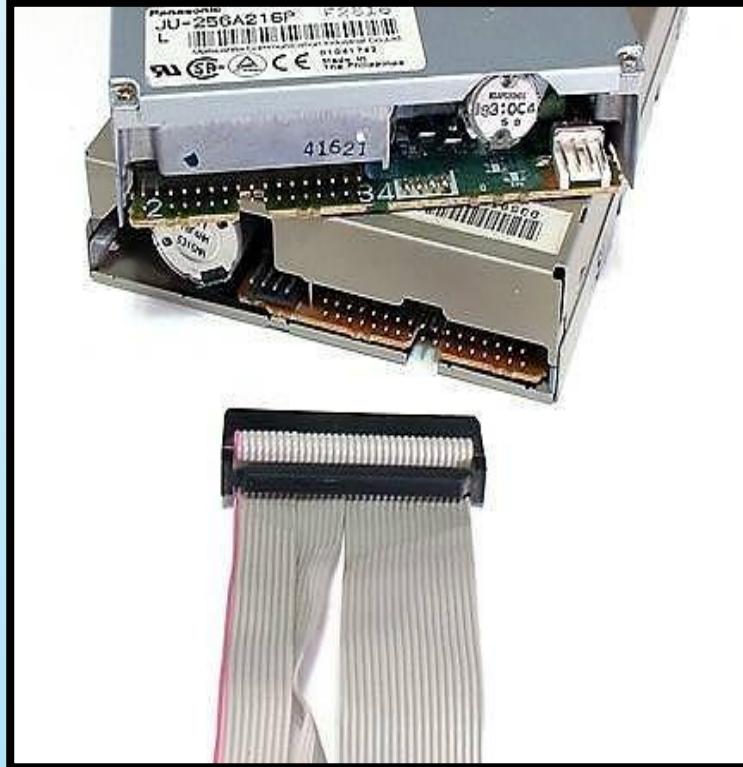
sistemul de antrenare al platanelor și dispozitivul de acționare al capului și pun la dispoziția controlerului date, într-o formă potrivită.



Cabluri și conexoare

Majoritatea unităților au cel puțin aceste 3 tipuri de conexoare :

- Conexoare de interfață (transfer de date)
- Conexoare de alimentare
- Conector opțional de legare la masă





Caracteristici

Timpul de acces este timpul care trece de la lansarea unei comenzi de acces până cand datele solicitate sunt accesibile.

Timpul de acces apare ca urmare a inertiei mecanice a brațului care poartă capetele și a sistemului de rotație a discurilor.

Timpul de poziționare este timpul necesar poziționării capetelor pe cilindrul dorit.

Timpul mediu de poziționare la acționările actuale cu motor de curent continuu electrodinamic este de 3-20ms.

Rata (viteza) de transfer caracterizeaza rata de transfer a datelor seriale preluate de pe disc odată ce capetele au fost poziționate și evident depinde de viteza de rotație.

La un HDD de 7200 rpm viteza tipică este 1Gbps.

Viteza de transfer poate fi dată și pentru transferul datelor din buffer-ul HDD în calculatorul gazdă, interfața SATA permite 6Gbps.

Timpul de latență este timpul necesar ca sistemul de rotație să aducă sectorul solicitat în dreptul capetelor. Acest timp depinde de viteza de rotație a discurilor, conform tabelului următor.

Rotational speed (rpm)	Average latency (ms)
15,000	2
10,000	3
7,200	4.16
5,400	5.55
4,800	6.25

Interfața SATA- caracteristici

Interfața serială SATA este formată din:

- 2 perechi de fire cu transmisie LVDS (Low Voltage Differential Signaling, 250mV),
 - o pereche pentru date emise,
 - o pereche pentru date recepționate, transmisia fiind diferențială.

Datele sunt codificate 8b/10b ca și Ethernet

Gigabit, PCIe sau Fibre Channel.

Legătura SATA este o legătură punct la punct,
fiecare drive este conectat printr-un cablu serial
la placa de bază.

transfer de date cu 150 MB/s.



Prima generație SATA (SATA 150, versiuni- 1.x) comunică cu viteza de 1,5Gbps, ceea ce înseamnă circa 1,2Gbps informație utilă, adică **150M**octeți/s.

SATA 2(SATA 300, versiuni- 2.x) asigură o viteză de 3Gbps(300 MB/s) compatibil cu SATA 150 la inițializare.

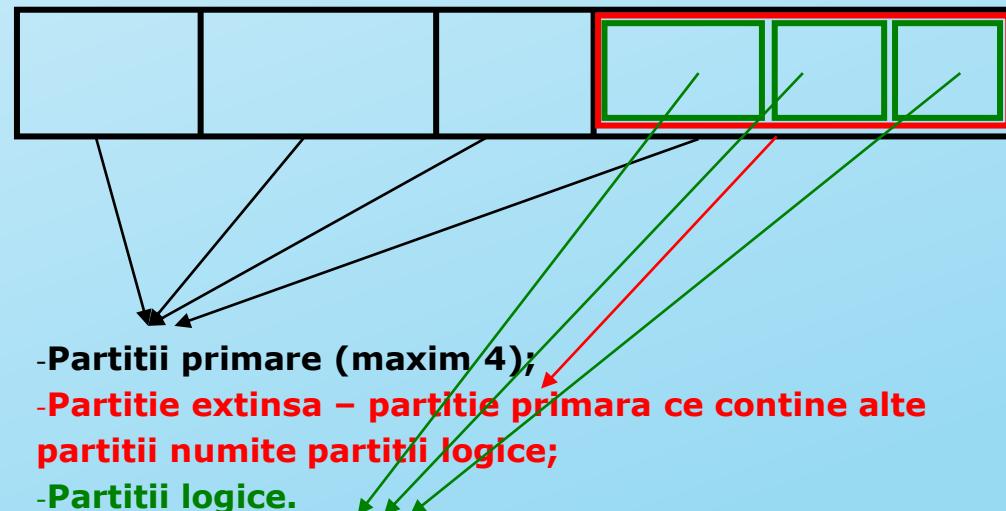
SATA 3(SATA 300, versiuni- 3.x) are latimea de banda **600MB/s**



STRUCTURA LOGICA A UNUI DISC

Din punct de vedere logic, un disc este divizat in partiții:

- primare;
- extinse;
- logice.



O partitie contine un anumit numar de sectoare.

Partitionarea

Crearea unor partiții pe hard-disc îi permite acestuia să gazduiască sisteme de fișiere distincte, fiecare în partitie.

Formatarea

Prin formatare un mediu de stocare este pregătit pentru stocarea de fișiere.

Formatarea constă în impărțirea spațiului de stocare disponibil în compartimente ce pot reține o cantitate fixă de date.

Aceste compartimente, denumite **clustere**, sunt manipulate de către sistemul de operare pentru stocarea și accesarea sistematică a informației

Formatarea discului.

Pentru un hard-disc sunt necesare 3 operații separate de formatare:

* Formatarea de nivel jos (Low-Level Formatting-LLF)

* Partiționarea

* Formatarea de nivel înalt (High-Level Formatting-HLF)

Formatarea de nivel jos

În cursul unei formatări de nivel jos, programul de formatare **împarte** pistele hard-discului într-un număr precizat de **sectoare**, creând **intervale de siguranță** între sectoare și între piste și înscriind informație în preambulul și postambulul sectorului.

De regula aceasta operatie se face în fabrică.

Aceste sectoare sunt **sectoare fizice**.

Pistele exterioare conțin mai multe sectoare decât pistele interioare pentru că sunt mai lungi.

*Toți cilindrii dintr-o anumită zonă au același număr de sectoare pe pistă.
Numărul de zone diferă de la o unitate la alta.*

S.M.A.R.T.

Self-Monitoring, Analysis and Reporting Technology

Este un standard industrial pentru predictia indicatorilor de fiabilitate

- Se refera la metodele de semnalizare (comunicare) intre senzorii din unitatea de memorie externa (HDD, SSD) si calculatorul gazda.
- Protocolul de comunicatie este standardizat, formatul/continutul raportului nu este standardizat.
- Detecteaza si raporteaza diversi indicatori de fiabilitate ai unitatii de memorie cu scopul de a anticipa defectiuni hardware iminente.

```
smartctl 6.6 2017-11-05 r4594 [x86_64-linux-4.19.0-17-amd64] (local build)
Copyright (C) 2002-17, Bruce Allen, Christian Franke, www.smartmontools.org
```

```
==== START OF INFORMATION SECTION ===
```

```
Model Family:      Western Digital Purple
Device Model:    WDC WD20PURX-64P6ZY0
Serial Number:   WD-WCC4M6UXSA3Z
LU WWN Device Id: 5 0014ee 2b83f6fda
Firmware Version: 80.00A80
User Capacity:   2,000,398,934,016 bytes [2.00 TB]
Sector Sizes:    512 bytes logical, 4096 bytes physical
Rotation Rate:   5400 rpm
Device is:        In smartctl database [for details use: -P show]
ATA Version is:   ACS-2 (minor revision not indicated)
SATA Version is:  SATA 3.0, 6.0 Gb/s (current: 6.0 Gb/s)
Local Time is:    Thu Nov 10 01:41:17 2022 EET
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
```

```
==== START OF READ SMART DATA SECTION ===
```

```
SMART overall-health self-assessment test result: PASSED
```

SMART Attributes Data Structure revision number: 16

Vendor Specific SMART Attributes with Thresholds:

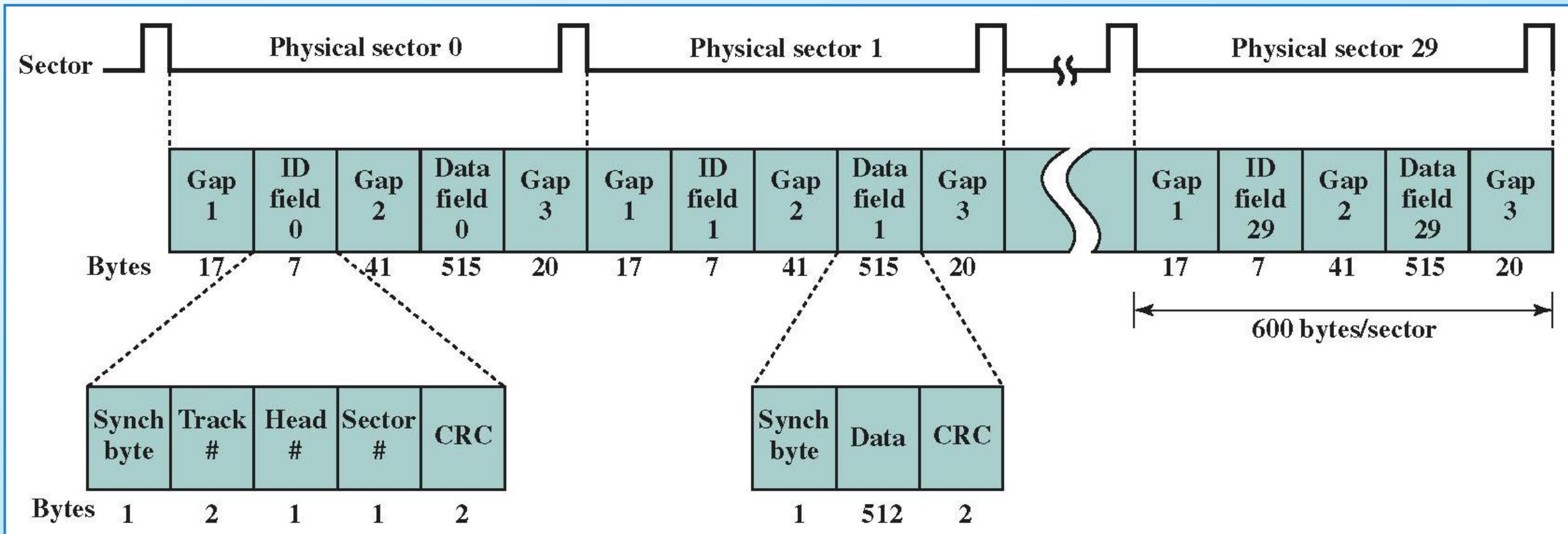
ID#	ATTRIBUTE_NAME	FLAG	VALUE	WORST	THRESH	TYPE	UPDATED	WHEN FAILED	RAW_VALUE
1	<u>Raw_Read_Error_Rate</u>	0x002f	200	200	051	Pre-fail	Always	-	0
3	Spin_Up_Time	0x0027	179	174	021	Pre-fail	Always	-	4050
4	Start_Stop_Count	0x0032	085	085	000	Old_age	Always	-	15129
5	<u>Reallocated_Sector_Ct</u>	0x0033	200	200	140	Pre-fail	Always	-	0
7	Seek_Error_Rate	0x002e	200	200	000	Old_age	Always	-	0
9	<u>Power_On_Hours</u>	0x0032	062	062	000	Old_age	Always	-	27843
10	Spin_Retry_Count	0x0032	100	100	000	Old_age	Always	-	0
11	Calibration_Retry_Count	0x0032	100	100	000	Old_age	Always	-	0
12	Power_Cycle_Count	0x0032	098	098	000	Old_age	Always	-	2729
192	Power-Off_Retract_Count	0x0032	200	200	000	Old_age	Always	-	86
193	Load_Cycle_Count	0x0032	195	195	000	Old_age	Always	-	15042
194	<u>Temperature_Celsius</u>	0x0022	111	107	000	Old_age	Always	-	36
196	<u>Reallocated_Event_Count</u>	0x0032	200	200	000	Old_age	Always	-	0
197	Current_Pending_Sector	0x0032	200	200	000	Old_age	Always	-	0
198	Offline_Uncorrectable	0x0030	100	253	000	Old_age	Offline	-	0
199	UDMA_CRC_Error_Count	0x0032	200	200	000	Old_age	Always	-	0
200	Multi_Zone_Error_Rate	0x0008	100	253	000	Old_age	Offline	-	0

SMART Error Log Version: 1

No Errors Logged

=1160 zile=3.2 ani (9.5 ani munca)

STRUCTURA A UNUI SECTOR FIZIC



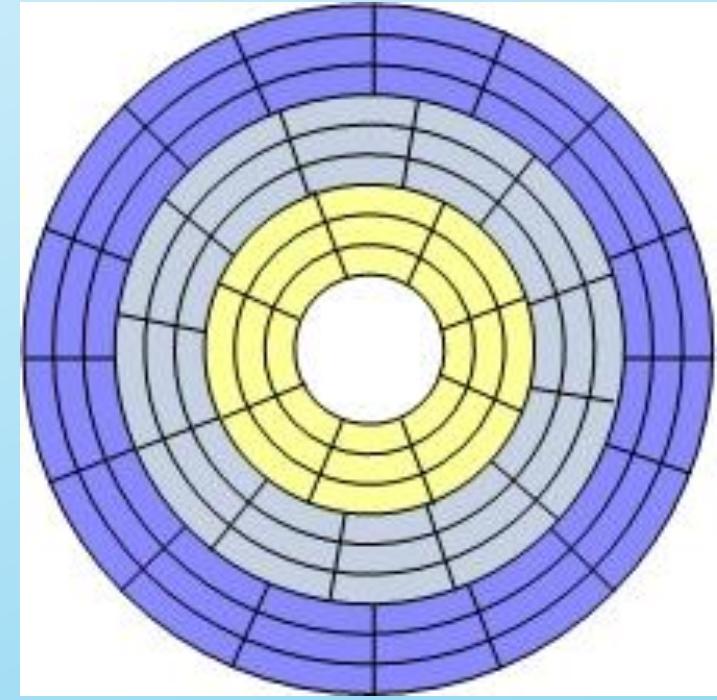
2¹⁶ piste
2⁸ capete de citire
2⁸ sectoare
= ~2TB

Zone Bit Recording

Pentru a mări cantitatea de informație care se poate stoca, suprafața platanului se împarte în 3 (sau mai multe zone) cu număr diferit de sectoare, mai multe înspre exterior.

Acest mod de împărțire pe zone se mai numește și [Zone Constant Angular Velocity](#) (Zone CAV, Z-CAV sau ZCAV).

Pentru ca datele să fie scrise sau citite corect trebuie ca viteza de rotație (unghiulară) să fie variabilă funcție de zona în care sunt datele accesate. (Viteza de parcurgere: sectoare/secunda este același)

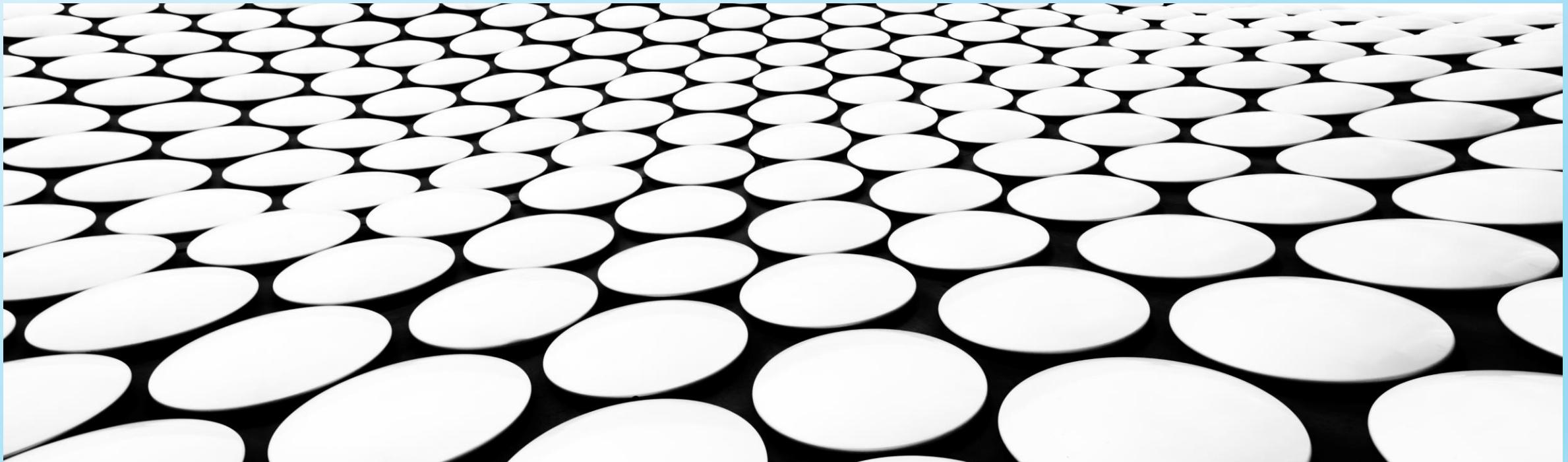


Alocarea fizica a sectoarelor

Placa logica face legatura intre sectoarele fizice si sectoarele logice.
Sistemul de operare are acces numai la sectoarele logice.

ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



Memorii electronice

Memorii electronice

■ Caracteristici

- **Capacitatea** memoriei: in multipli de biți/octeti etc
- **Geometria**: lungimea unui **cuvânt** (locatie); aranjarea si modul de adresare al cuvintelor
- **Timpul de acces**: timpul dupa care se obtin datele la iesire in raport cu momentul aplicarii adresei la intrare
- **Ciclul memoriei**; timpul necesar pentru scrierea sau citirea unei locatii
- **Puterea** consumata, totala (W) sau specifica (μ W/bit)
- **Tehnologia** de realizare: bipolară, MOS , etc

Memorii ROM electronice

■ Tipuri

- ROM
 - Informația este inclusă în designul circuitului electronic al memoriei
- PROM (Programmable Read-Only Memory)
 - Memoria poate fi scrisă o singură dată
- EPROM (Erasable Programmable Read Only Memory)
 - Înregistrari repetitive
 - Tipuri
 - UVEPROM
 - Stergere totală prin iradiere cu UV (~100 cicluri)
 - EEPROM
 - Stergere totală prin aplicarea unui semnal electric (~100 000 cicluri)

Memoria RAM electronica

■ TIPURI

- Statica (**SRAM**)
 - Informația se conserva atat timp cat memoria este alimentata cu energie
- Dinamica (**DRAM**)
 - Informația dispare dupa un anumit interval de timp, si de aceea trebuie regenerata (timp tipic de regenerare: 2 ms)

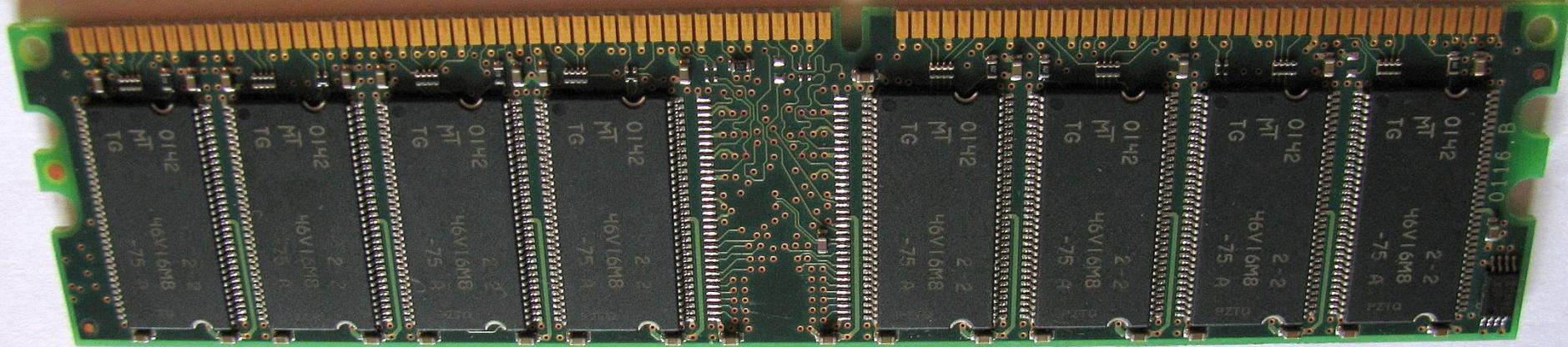
comparație

SRAM	DRAM	Unit. tipica
Mai rapida	Mai lenta	MB/s
Densitate mai mica	Densitate mai mare	MB/cm ²
Mai scumpă	Mai ieftină	\$/MB
Consum energetic mai mare	Consum energetic mai mic	W/MB

Tipuri/tehnologi de memorii DRAM

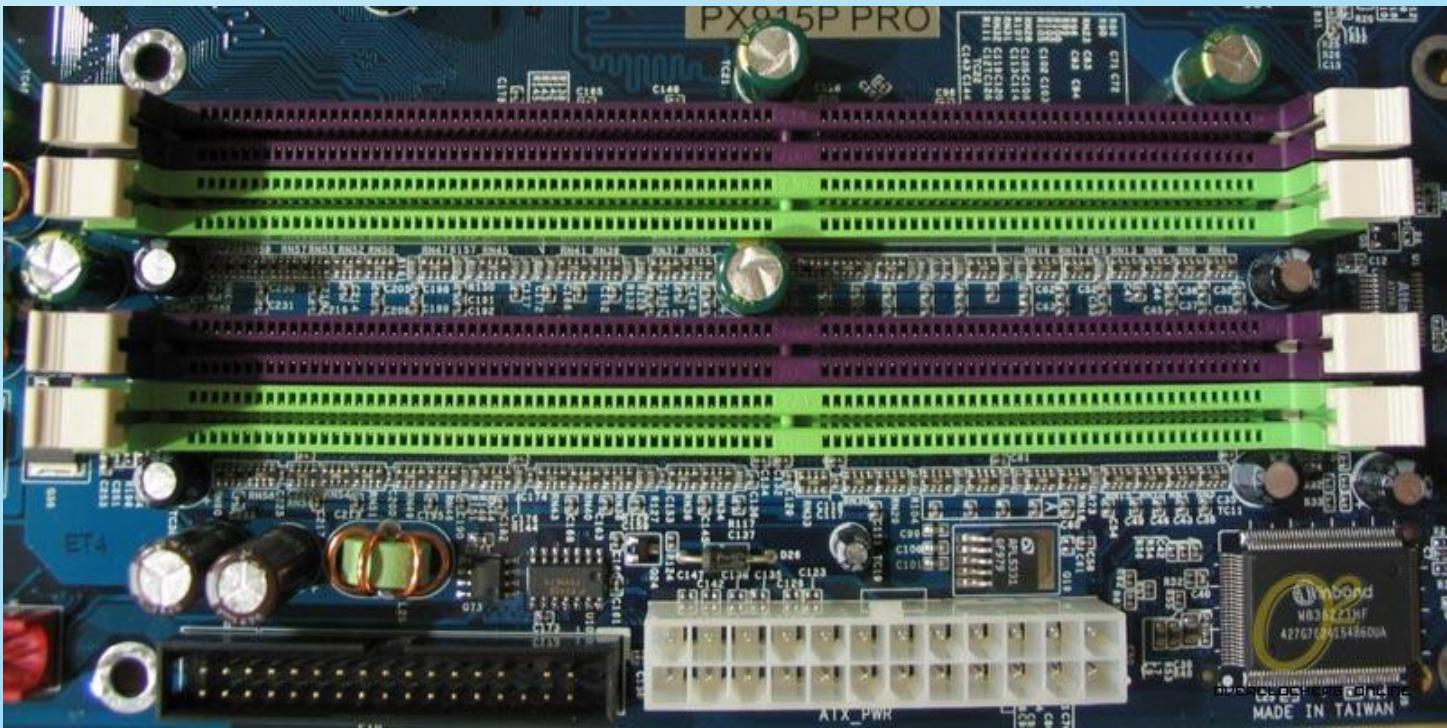
- **FPM DRAM** (fast page mode)
 - Memorie matriceala paginata
 - Descarca rapid o pagina
- **EDO DRAM** (extended data output)
 - Timp de acces redus la citire
- **SDRAM** (synchronous)
 - Este organizata in **blocuri** care pot lucra in paralel
 - Sincronizare cu semnalul de ceas (magistrala)
- **DDR SDRAM** (double data rate)
 - La un ciclu ceas au loc doua transferuri

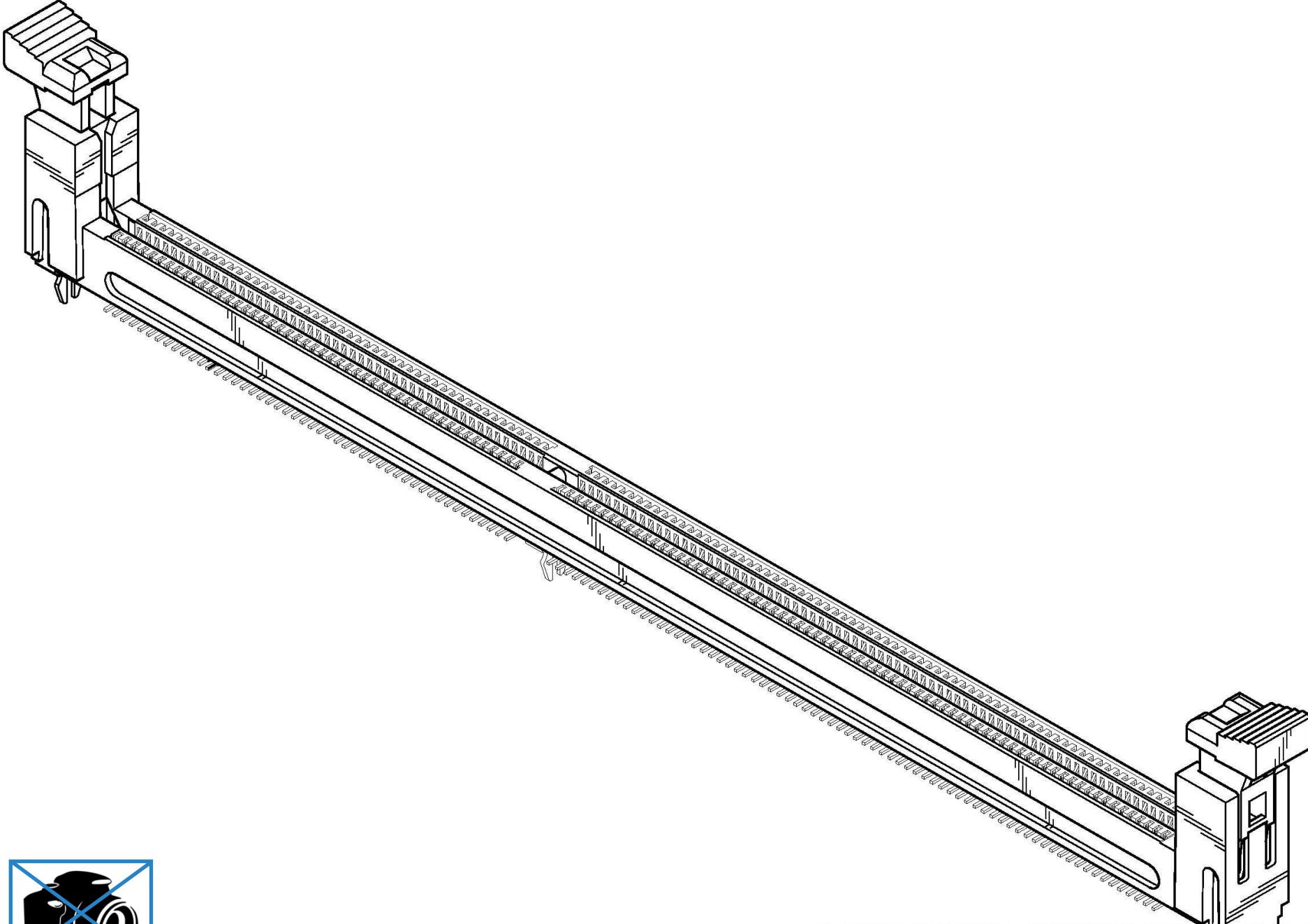
Nota: este vorba despre ceasul memoriei (care este distinct de ceasul microprocesorului)



184-pin DIMM (dual in-line memory module)

DDR SDRAM





STRUTPATENT.COM

DDR SDRAM Standard	Bus clock (MHz)	Internal rate (MHz)	Transfer Rate (MT/s)	Voltage	DIMM pins	SO-DIMM pins	MicroDIMM pins
DDR (2001)	100–200	100–200	200–400	2.5/2.6	184	200	172
DDR2 (2006)	200–533	100–266	400–1066	1.8	240	200	214
DDR3 (2011)	400–1066	100–266	800–2133	1.5	240	204	214
DDR4 (2016)	800–1600	200–400	1600–3200	1.2	288	256	-
DDR5 (2021)	1600–3200	200–400	3200–6400	1.1	288	262	-
DDR5 (2025?)							

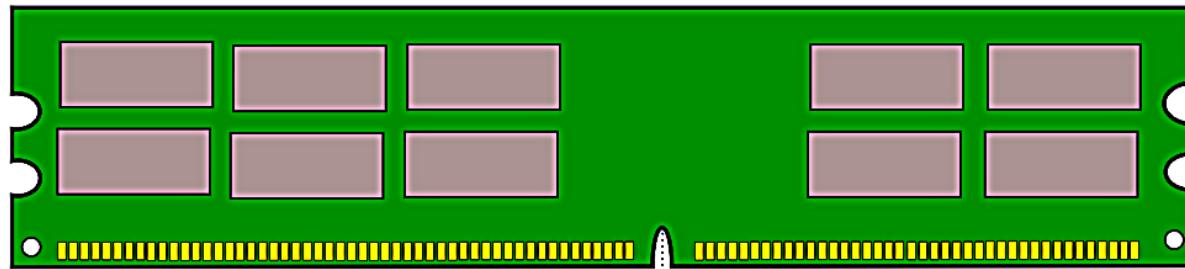
MegaTransfers per second (MT/s)

SO-DIMM: small outline dual in-line memory module (laptops/notebooks)

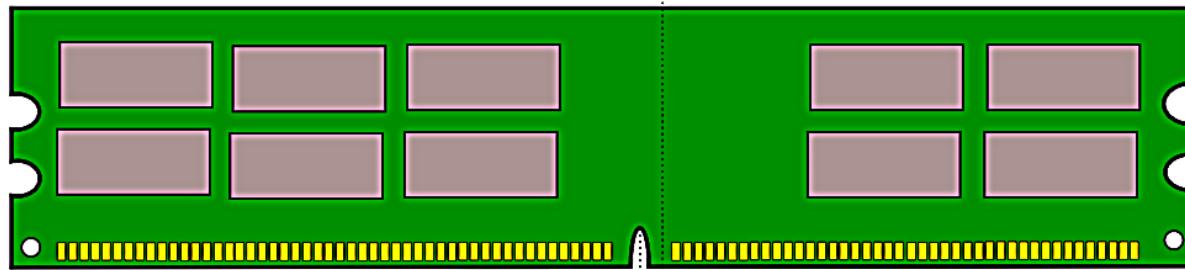
MicroDIMM: (tablets)



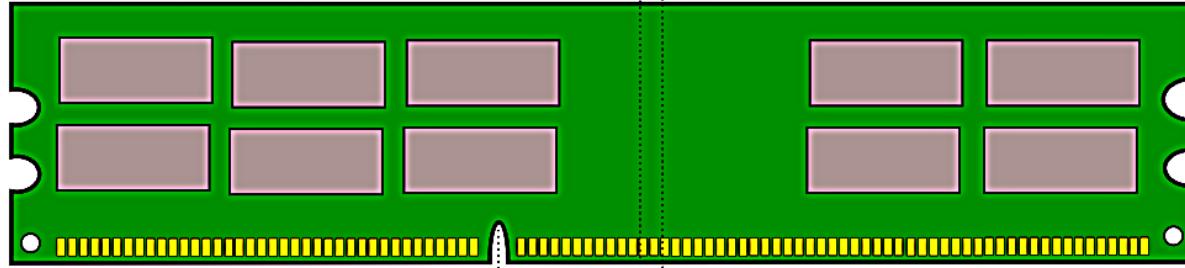
DDR



DDR 2



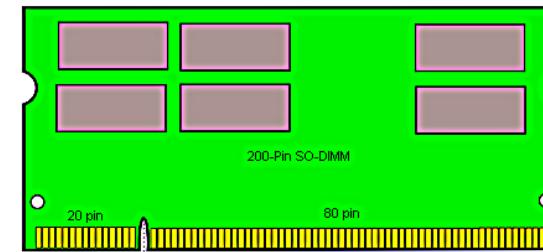
DDR 3



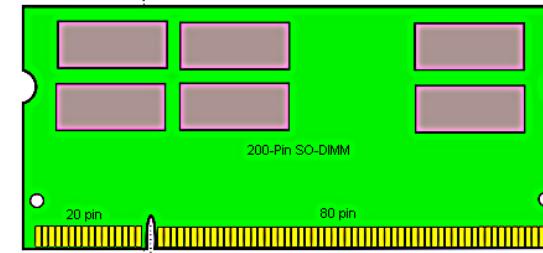
cm.



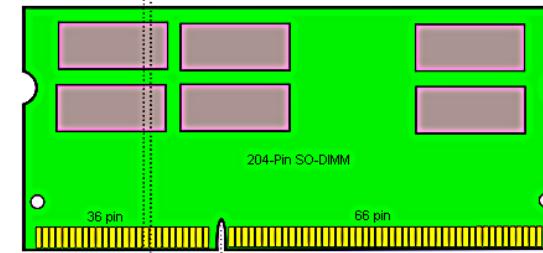
SO-DIMM DDR



SO-DIMM DDR 2



SO-DIMM DDR 3



Memoria Flash

Combinăție intre RAM si EEPROM

- Este un ansamblu de blocuri EEPROM (ce contin tranzistori MOS cu poarta flotanta)
- Nu se poate sterge o locație, ci numai un bloc
- Performanțe intre HDD si RAM
- Numar limitat de rescrieri
- Este implementat un mecanism de protectie (ascuns) care scoate din uz blocurile cu numarul de cicluri epuizat sau defecte

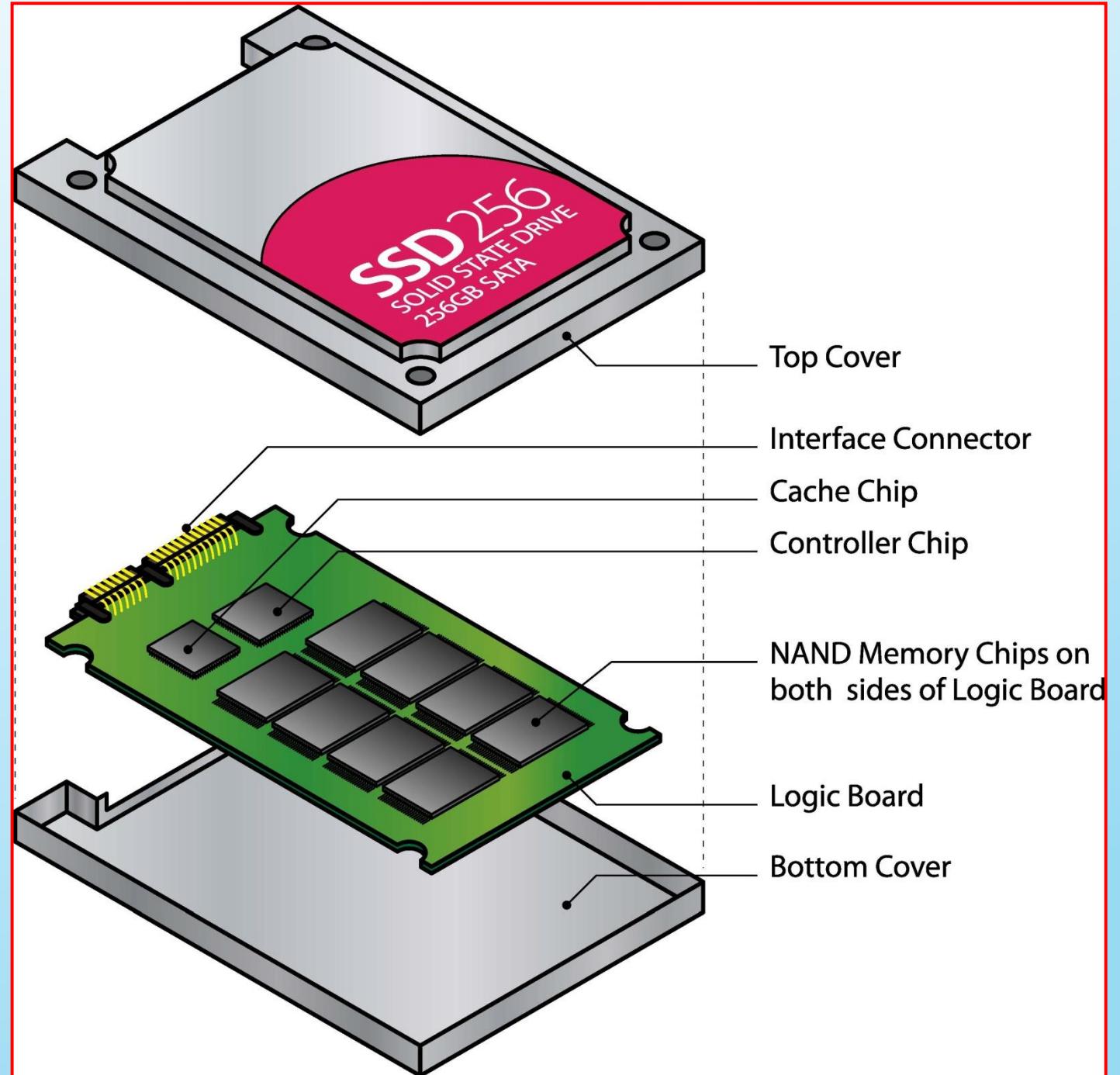
- Înlocuiesc HDD sub numele **HDD SSD** (solid state devices)
- Au depasit 16 TB
- **Microcarduri** pentru telefoane și aparate foto
- **Stick-uri USB**

Solid State Drive SSD

SSD este un dispozitiv de stocare nevolatilă a informației construit cu memorii semiconductoare.

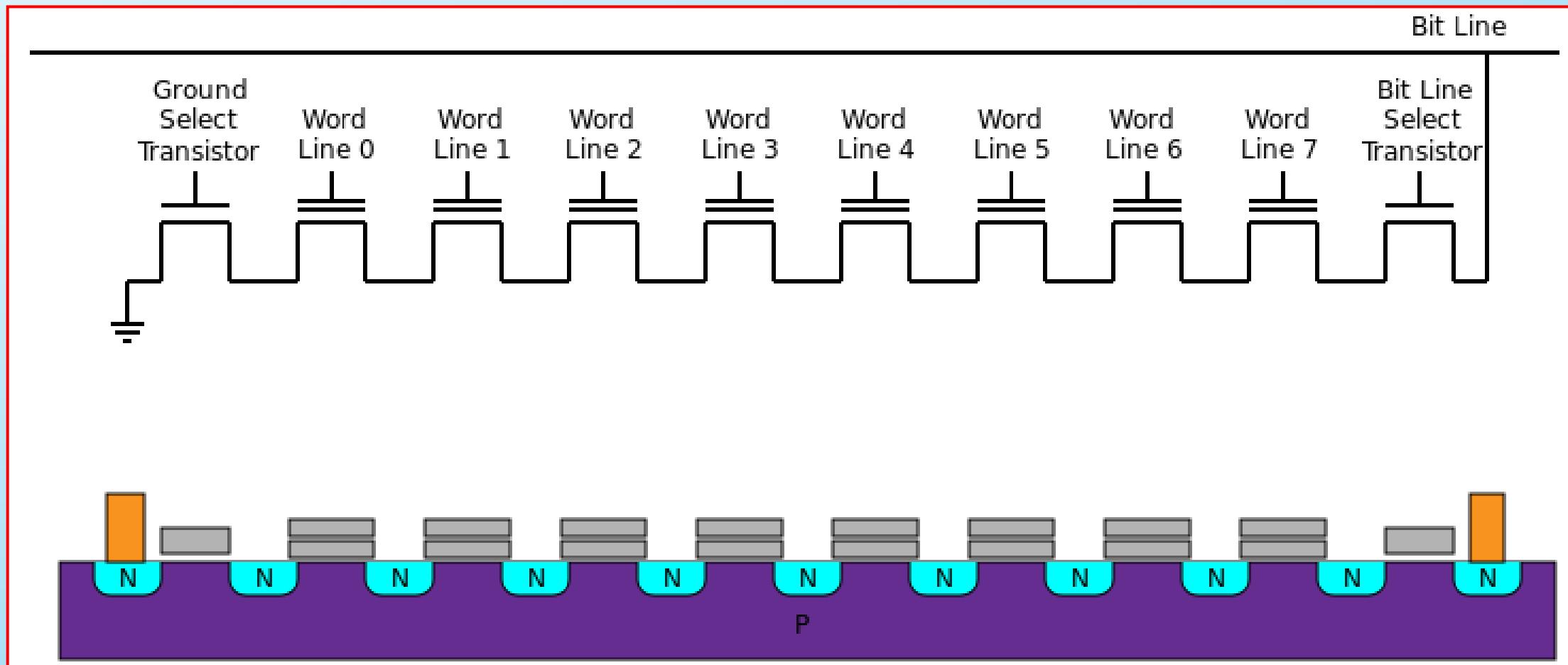
Pentru a putea fi utilizat în loc de HDD dimensiunile carcasei și interfața sunt standard.

SSD este mai silențios și mai fiabil decât un HDD datorită faptului că nu are piese în mișcare.



Solid State Drive SSD- structura

O variantă este cu memorii Flash, varianta NAND Flash fiind prezentată în figură.



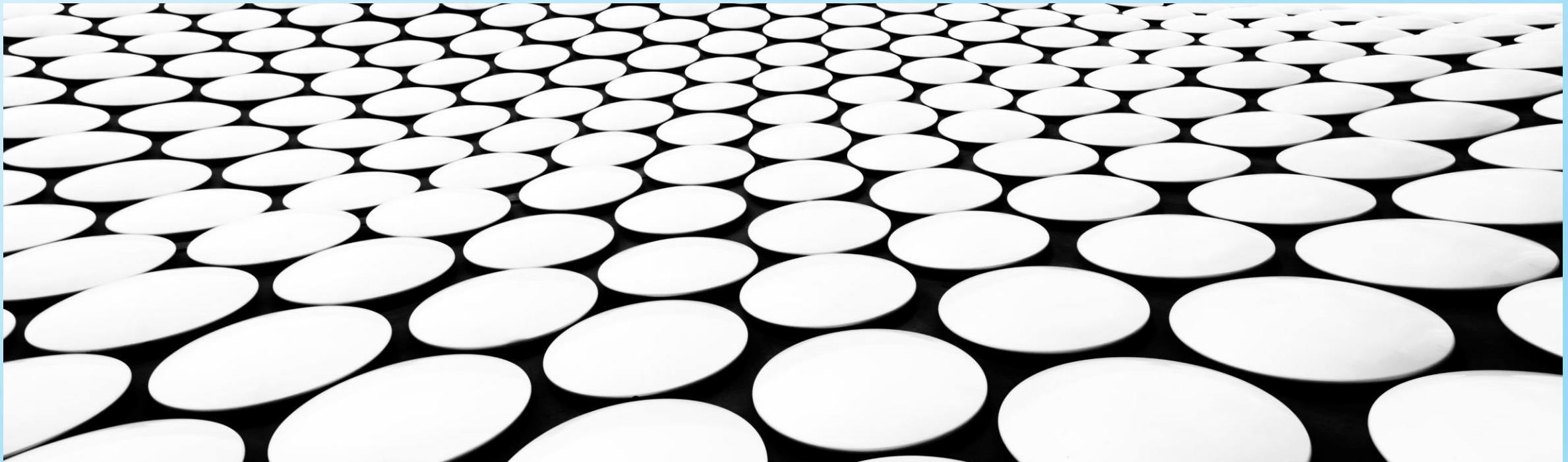
Analiza comparativă

Device	Critical feature-size F	Area (F^2)	Density (Gbit /sq. in)
Hard Disk	50 nm (MR width)	1.0	250
DRAM	45 nm (half pitch)	6.0	50
NAND (2 bit)	43 nm (half pitch)	2.0	175
NAND (1 bit)	43 nm (half pitch)	4.0	87
Blue Ray	210 nm ($\lambda/2$)	1.5	10

Această analiză comparativă arată unitatea modurilor de stocare a informației. Principiile de stocare cu semiconductori, optice și magnetice pot fi comparațe prin densitate și preț.

ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



Arhitecturi RAID

Ce înseamnă RAID?

- RAID (Redundant Array of Independent Disks sau Redundant Array of Inexpensive Disks)
 - reprezintă un sistem de HDD(SSD)-uri care are ca scop:
 - creșterea vitezelor scriere / citire,
 - protecția datelor
 - combinația lor.
- Există mai multe tipuri de RAID. Cele mai folosite și cunoscute sunt RAID 0, RAID 1, RAID 5 și RAID 10.
- De obicei se utilizează mai multe discuri identice pentru o implementare RAID

RAID 0

Tipul de configurație **RAID 0** este utilizat în scopul îmbunătățirii **vitezei** de scriere și citire a datelor.

- Exemplu:

O configurație RAID 0 cu 3 HDD-uri.

Salvăm un fișier care este alcătuit din 3 părți (A,B și C).

Salvarea se efectuează în felul următor:

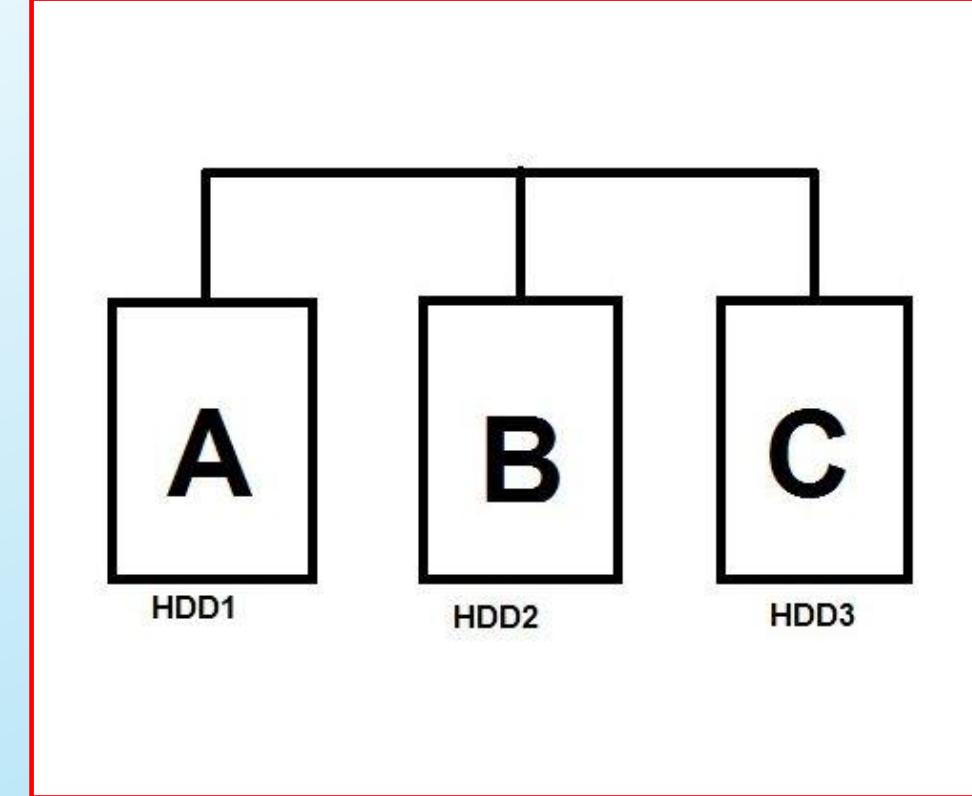
partea A a fișierului este salvată pe primul HDD,

partea B pe al doilea HDD, iar

partea C pe al treilea HDD.



File=A+B+C



Scrierea/citirea se face **simultan** pe mai multe HDD-uri, mărindu-se performanța totală a sistemului.

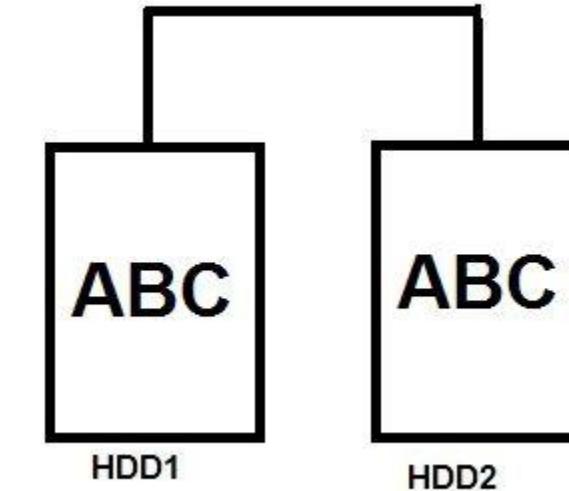
- RAID 0 este cea mai **rapidă** metodă de copiere, stocare și citire a datelor însă
- defectarea unuia dintre cele 3 HDD-uri atrage după sine **pierderea totală** a informațiilor salvate.

RAID 1

- RAID 1 se folosește mai ales când dorim o **securitate sporită** a datelor importante.
- Dezavantajul acestei configurații este că spațiul de stocare este de jumătate din suma capacitaților HDD-urilor.



FILE=A+B+C



Exemplu: Se folosesc 2 HDD-uri pentru o configurație de tip **RAID 1**.

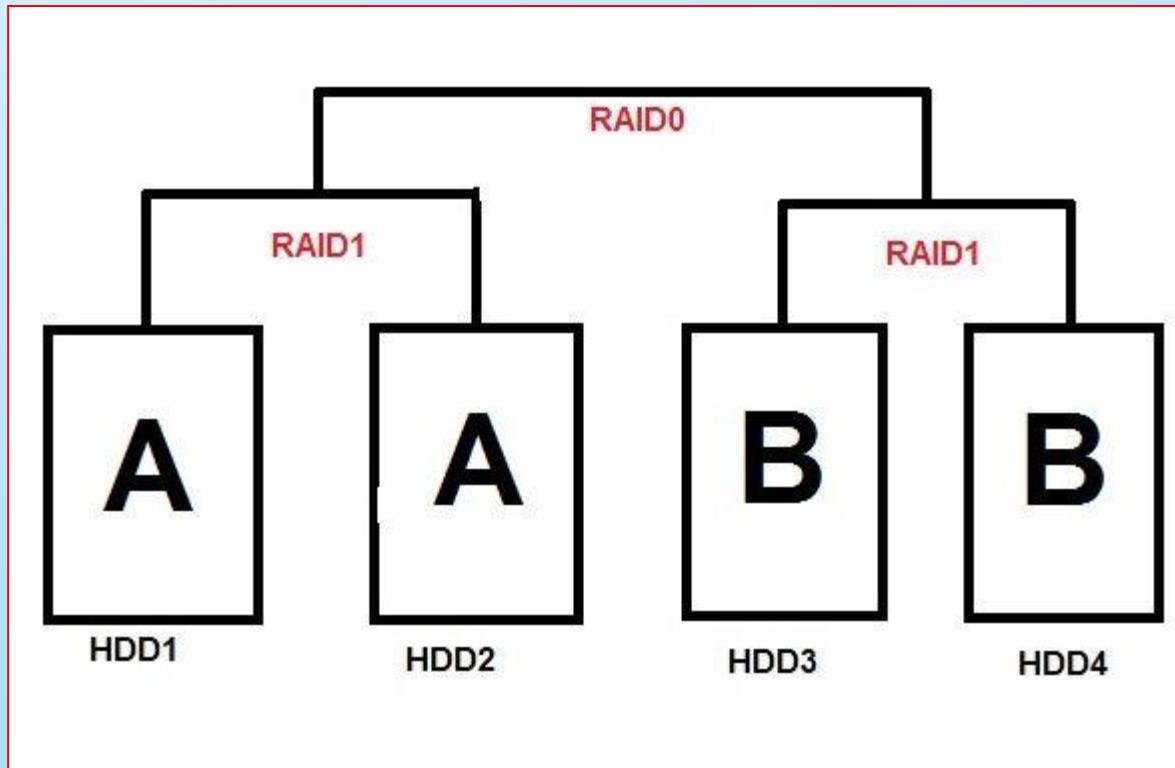
- Salvam același fișier cu 3 părți (A,B și C) ca și în exemplul anterior. Salvarea se face în felul următor:
Părțile A, B și C ale fișierului sunt salvate pe fiecare HDD în parte.
- Cu alte cuvinte fișierul este duplicat, îl avem pe ambele HDD-uri în aceeași formă. Dacă unul dintre cele 2 HDD-uri va ceda, atunci integritatea datelor (în exemplul nostru fișierul alcătuit din cele 3 părți: A, B și C) rămâne în siguranță deoarece fișierul există pe ambele HDD-uri.

RAID 10

- Această configurație este o combinație între RAID0 și RAID1.
- Este nevoie de **minim 4 HDD-uri**.
- RAID 10 oferă citirea datelor după modelul RAID 0, iar stocarea se face ca la RAID 1.
În acest fel beneficiem simultan de **viteza RAID0 și siguranța RAID1**.
- Dezavantajul este că putem folosi numai jumătate din capacitatea celor 4 HDD-uri.



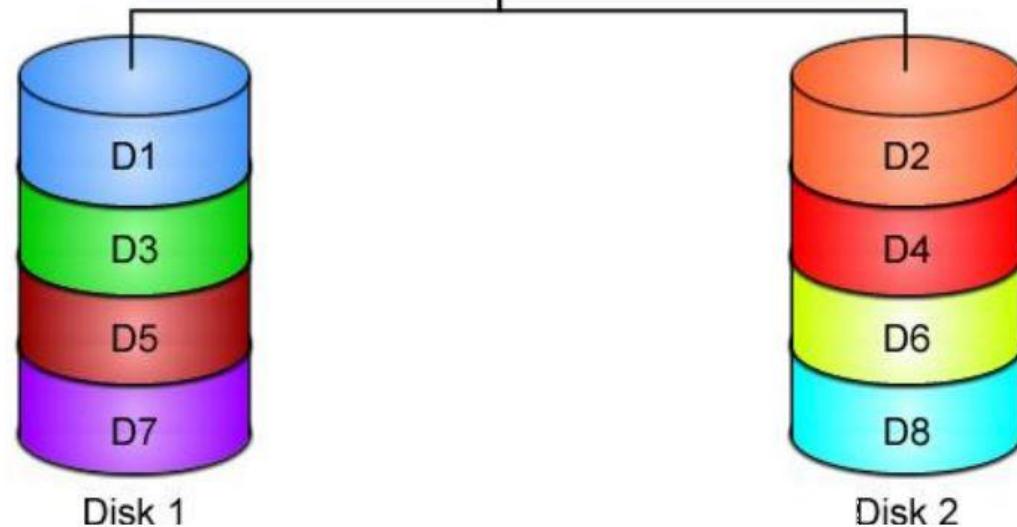
FILE=A+B



RAID 0

- RAID 0 nu este o arhitectură RAID în adevăratul sens al cuvântului deoarece nu asigură nicio redundanță a datelor.
- RAID 0 este folosit strict pentru maximizarea performanțelor de viteza în lucrul cu harddisk-urile.
- Datele sunt întrețesute (stripping) în mod secvențial pe mai multe discuri și sunt tratate ca un singur disc (sau **volum**) virtual.
- De obicei 4 discuri formează un volum.
- Implementările RAID 0 divizează volumele în **blocuri** și scriu datele în blocuri consecutive, localizate fizic pe discuri diferite din cadrul ariei de discuri.
 - de exemplu un bloc conține 512 **septoare**, iar un sector 512 octeti

RAID 0



Felii: D1,D1D3,D4,D5,D6,D7,D8...

- Operațiile de scriere și citire au loc în paralel fiind executate SIMULTAN pe toate discurile aflate în volum.
- Mărimea blocurilor este definită de către utilizator și poate fi de 512 sectoare, spre exemplu.

- Utilizând această tehnică de întrețesere se obține o rată de transfer a datelor foarte ridicată, îmbunătățindu-se în felul acesta și performanța efectivă a sistemului.
- În eventualitatea defectării chiar și a unui singur disc, RAID 0 nu oferă redundanță datelor.
- În lipsa posibilității de regenerare a discului, datele ce se găsesc pe acesta se vor pierde. Din acest motiv, RAID 0 nu este folosit în aplicații de înaltă disponibilitate.



RAID 1

- RAID 1 asigură redundanță datelor prin oglindirea acestora (mirroring).
- Metoda aceasta creează două sau mai multe copii ale aceluiași volum de date și le plasează pe discuri separate. Fiecare copie este o imagine oglindită a celeilalte.
- RAID 1 poate scădea performanța generală a sistemului deoarece trebuie să efectueze tot atâtea scrieri câte copii sunt.
- Performanța citirilor este însă mult îmbunătățită deoarece **cererile sunt trimise simultan** controlerelor de discuri ale tuturor copiilor. Discul care răspunde primul cererii de citire va returna datele sistemului.



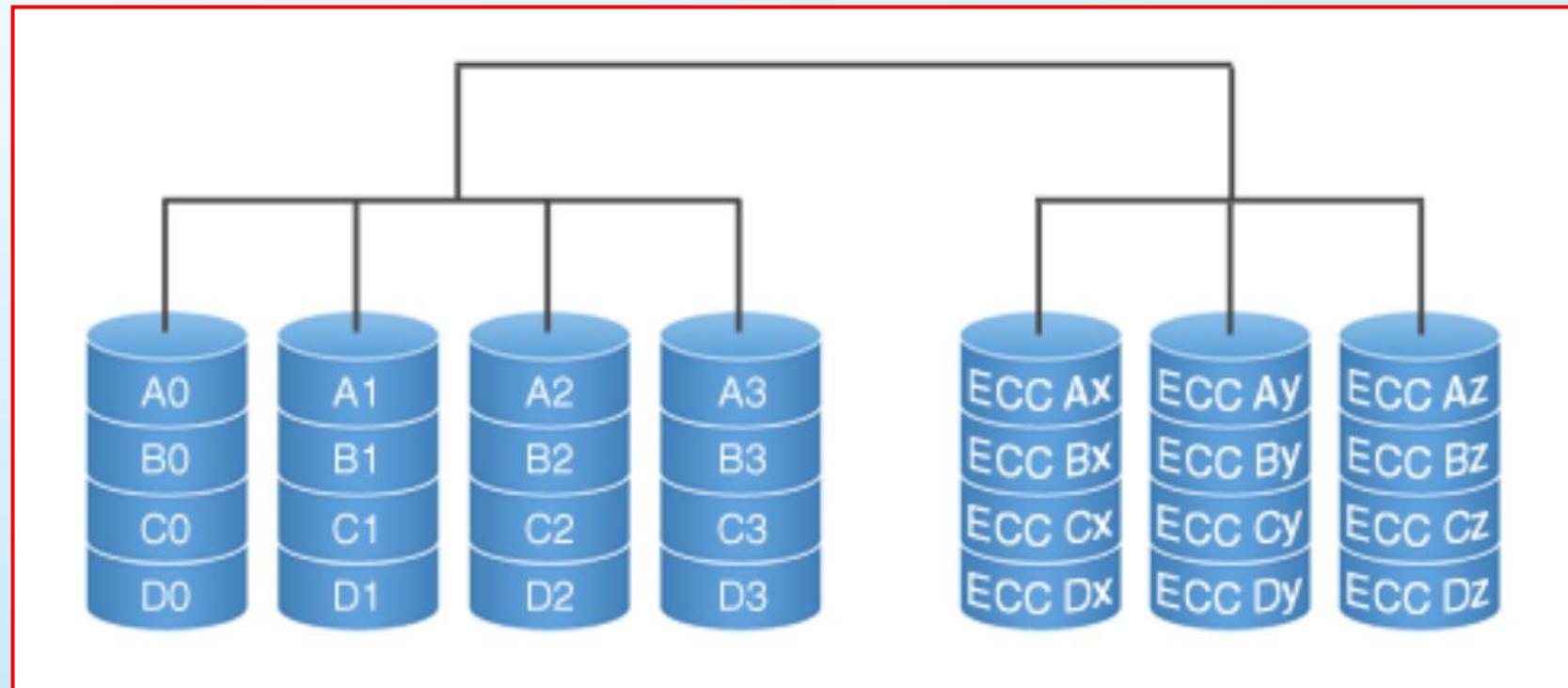
RAID 1



- RAID 1 este cea mai scumpă implementare deoarece **asigură redundanță** datelor în proporție de 100%
- Poate suporta **mai multe defecte** simultan.
În caz de defectare, datele nu trebuie reconstruite ci pur și simplu **copiate**.
- asigură **cea mai mare disponibilitate** a datelor prin utilizarea celui mai mic număr de discuri din configurație.

RAID 2

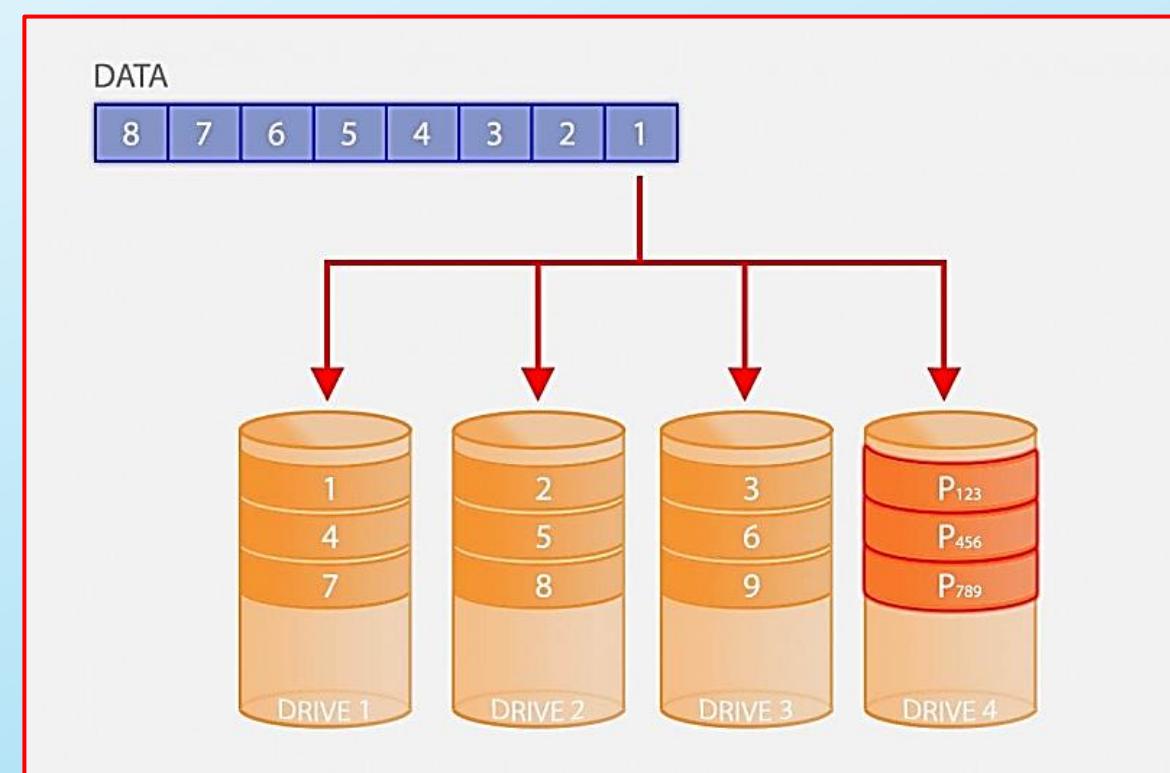
- Acest model presupune că se distribuie datele pe mai multe discuri în paralel, efectuându-se o **întrețesere la nivel de bit**,
 - În cadrul ariei de discuri sunt utilizate și discuri de verificare.
 - Citirea și scrierea datelor se face utilizând **tehnici de codificare** bazate pe coduri corectoare de erori de tip **Hamming** pentru a asigura detectia și corecția erorilor.



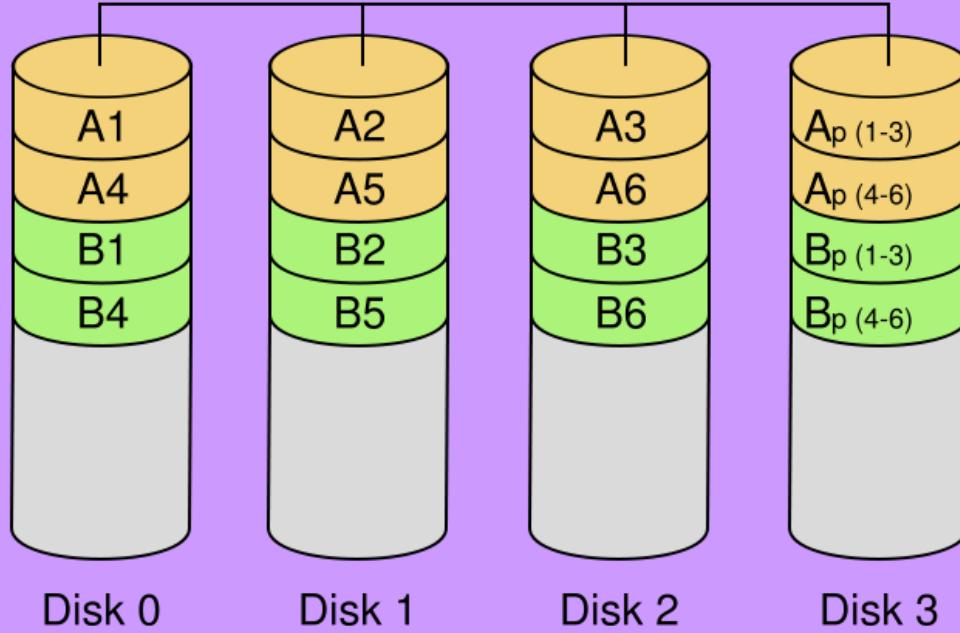
- RAID 2 are mai multe dezavantaje decât avantaje.
- Printre acestea trebuie amintit faptul că:
 - utilizeaza tehnici de codare bazate pe coduri corectoare de erori de tip Hamming
 - necesită utilizarea unor grupuri mari de discuri pentru a asigura consistență (ex: 4 discuri de verificare pentru 10 discuri de date, 5 discuri de verificare pentru 25 de discuri de date).
 - rata de transfer este cel mult egala cu cea a unui disc
- Din cauză că această tehnică de codificare este foarte complexă și scump de implementat, RAID 2 nu prezintă un real interes pentru mediul comercial.

RAID 3

- RAID 3 folosește doar **un singur disc** dedicat pentru memorarea **informațiilor de paritate**, celelalte discuri din cadrul ariei de discuri fiind folosite pentru memorarea informației utile ce este împrăștiată pe toate discurile ariei.
(exact ca și în cazul RAID 2).
- Acesta efectuează un **SAU EXCLUSIV** asupra datelor și nu un cod corector de erori.
- RAID 3 lucreaza la nivel de bit



RAID 3



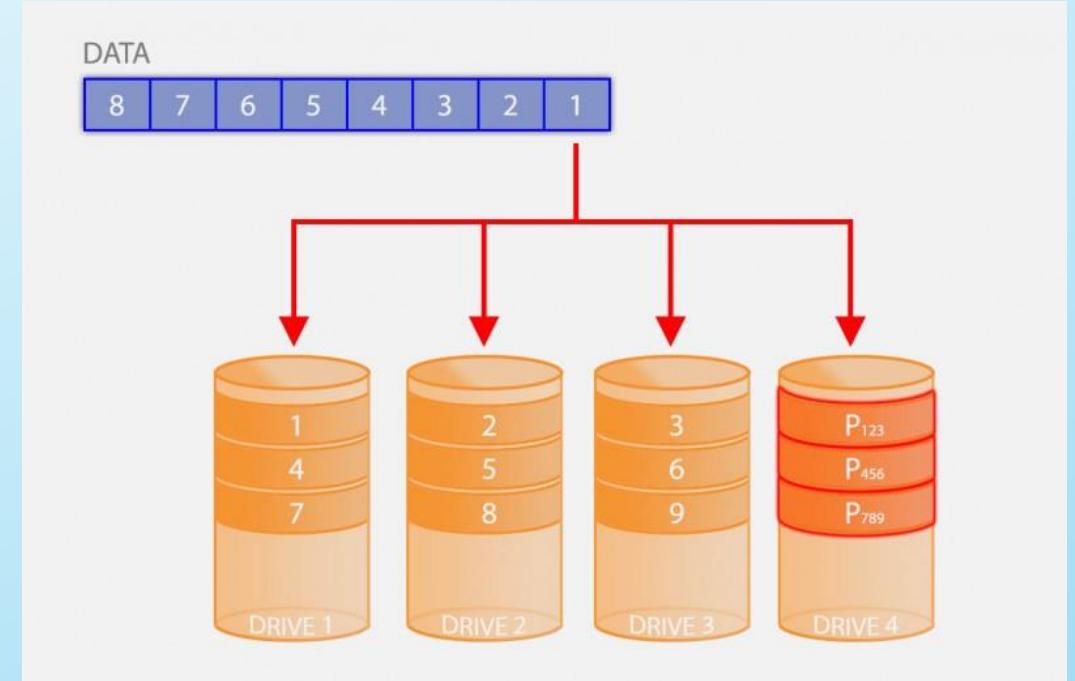
- Cantitatea minimă de date care este scrisă sau citită într-o operație de intrare sau de ieșire este egală cu numărul de discuri din aria înmulțit cu numărul de octeți ai sectorului discului.
 - Aceasta este cunoscută ca **unitate de transfer**.
- ✖ RAID 3 oferă și posibilitatea **înlocuirii la cald** a unui disc și recuperarea datelor.

- La un moment de timp dat nu poate fi activă decât o singură cerere de intrare sau de ieșire deoarece capetele de acces se mișcă în paralel în aria de discuri.
- Acest lucru ne asigură **o rată bună** de transfer a datelor atunci când datele accesate sunt **dispuse secvențial pe disc**, dar face ca RAID 3 să nu fie suficient de practic pentru aplicații cu date dispuse aleator în aria de discuri.
- O rată de transfer foarte bună se obține atunci când unitatea de transfer are aceeași mărime cu dimensiunea blocurilor de date care se scriu sau se citesc. Cu cât blocurile transferate sunt de dimensiuni mai mici, cu atât rata de transfer scade.



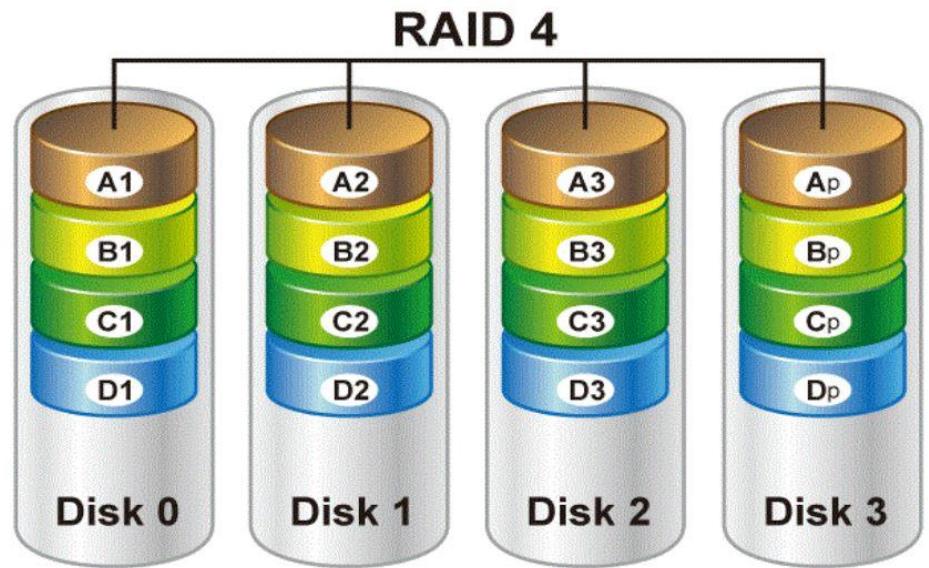
RAID 4

- În cazul RAID 4 datele sunt distribuite la fel ca și în cazul RAID 3 pe mai multe discuri de date și pe un disc de paritate.
- Diferența constă în faptul că **un bloc** de date se află **pe un singur disc**, astfel încât citirea se face accesând un singur disc și nu mai multe.
- Aria de discuri poate prelucra mai multe cereri deodata începând cu unitatea 4.
- RAID 4 lucrează la nivel de **bloc**



Date scrise astfel: blocul 1 merge la unitatea 1, blocul 2 la unitatea 2, blocul 3 la unitatea 3 și paritatea calculată la unitatea 4.

Dacă o unitate eşuează, datele „lipsă” pot fi recalculate pe baza datelor rămase.



Discul de paritate devine un element de scădere a performanței deoarece este implicat în toate operațiile de scriere.

Acest lucru se întâmplă deoarece paritatea, pentru fiecare operație de scriere, se calculează având în vedere și informația de pe celelalte discuri, în sectoarele de date asociate.

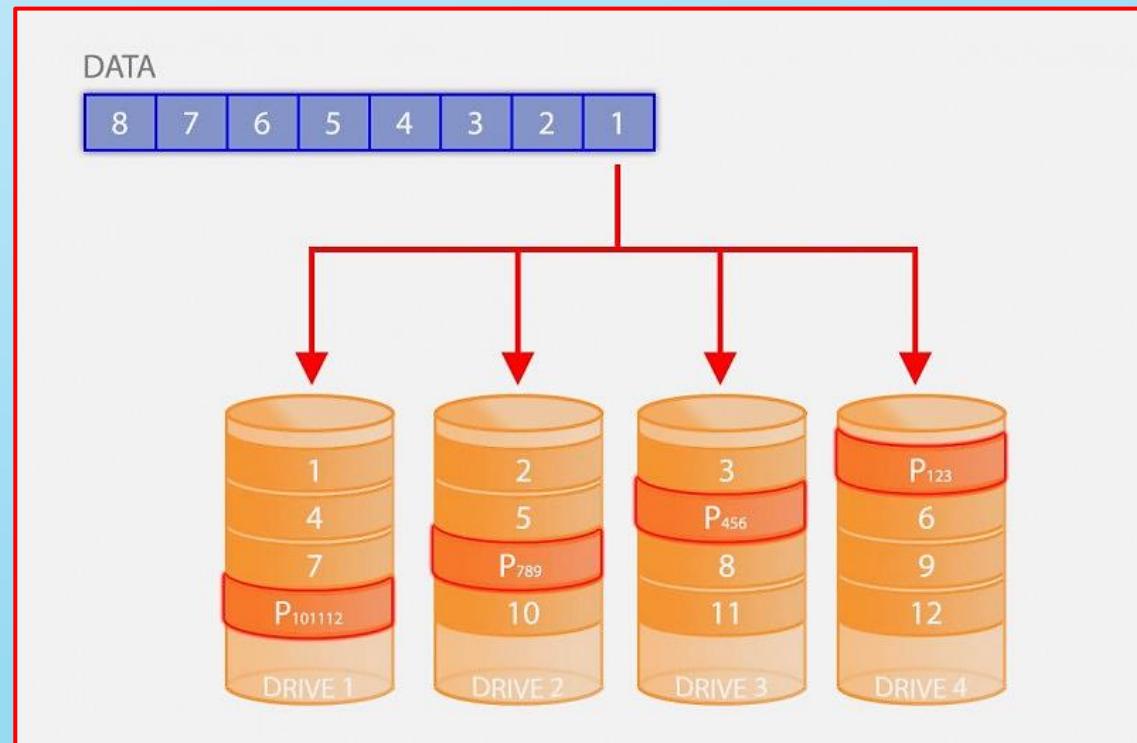
Această procedură face ca RAID 4 să fie considerat nepractic.

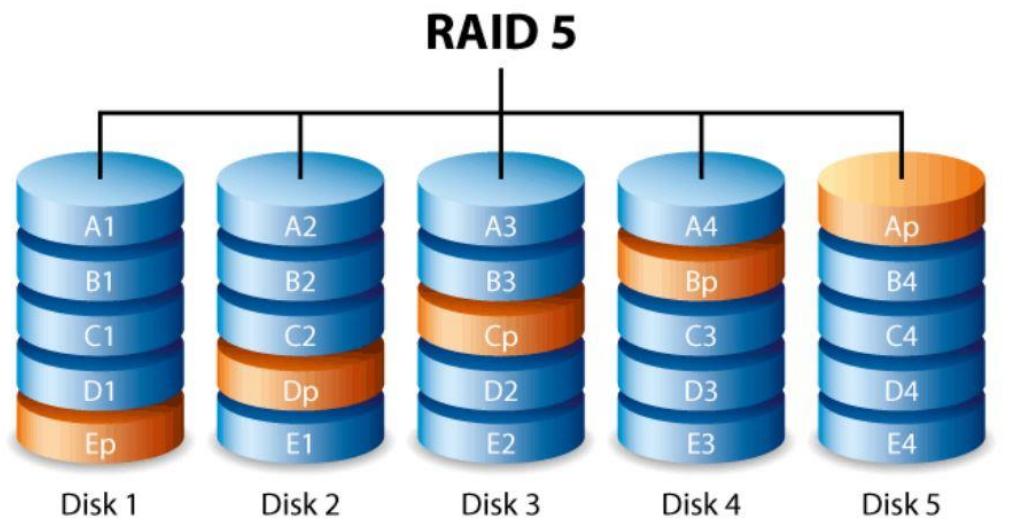
RAID 5

- În cazul RAID 5 datele sunt distribuite pe mai multe discuri de date și pe un disc de paritate.
- RAID 5 nu are un disc dedicat special numai pentru informația de **paritate**, ci distribuie informațiile de tip date și informațiile de tip paritate pe toate discurile ariei.
- RAID 5 permite **mai multe operațiuni de acces concurente** la dispozitivele din cadrul ariei de discuri, fiind satisfăcute astfel multiple cereri concurente de intrare sau ieșire.

Datele se recuperă ca și în cazul RAID 4

Aria suportă un singur disc defect.





Procesorul efectuează toate calculele pentru paritate, iar în cazul unei defecțiuni, operațiunile de intrare sau de ieșire sunt de până la de 3 ori mai consumatoare de resurse.

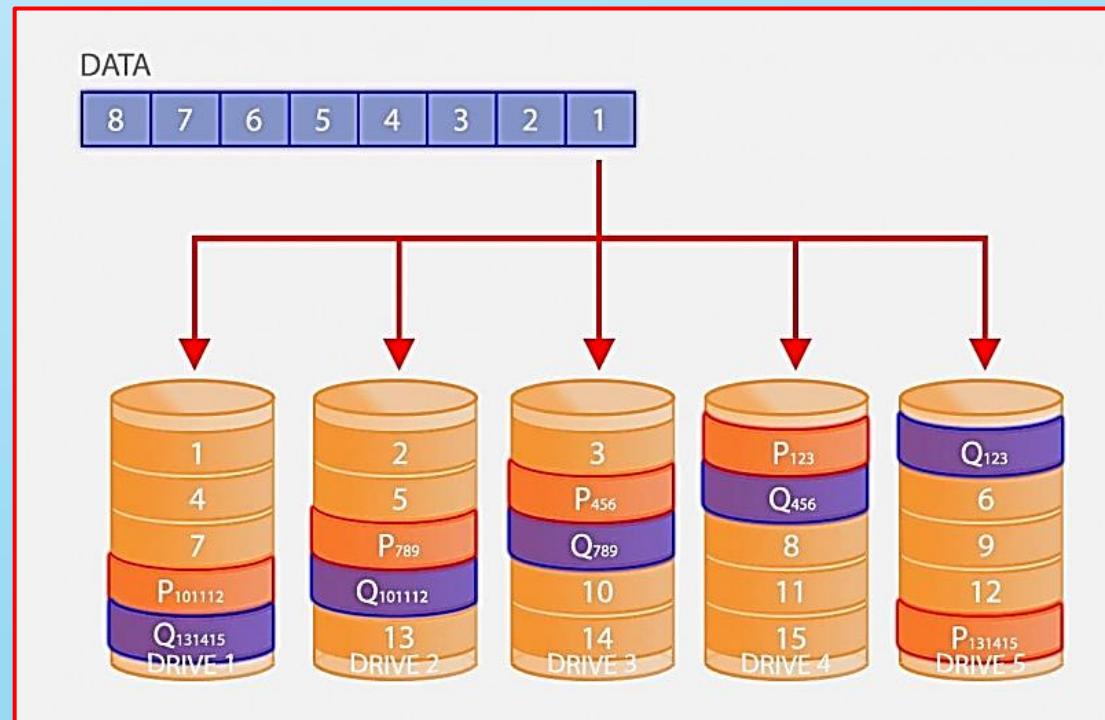
Având un cost relativ scăzut per GB utili și o performanță medie foarte bună, RAID 5 este utilizat cu succes pentru toate tipurile de transfer (servere pentru baze de date rationale, servere de fișiere)

- Astfel, **performanța** transferurilor pentru aceste arii de discuri crește semnificativ, acestea devenind cele mai potrivite și utile în cazul operării cu blocuri de dimensiune mică, sau cu datele dispuse **în mod aleator** pe discuri. Ciclul de scriere se realizează în 3 pași:
 - Este citită paritatea existentă și data ce urmează a fi schimbată. Vechea dată din valoarea parității existente este scăzută și apoi se memorează paritatea corectă într-un buffer.
 - Este calculată noua paritate pe baza valorii din buffer și a valorii noii date de memorat și se scriu pe discurile corespunzătoare noua dată și paritatea actualizată.
- Implementările software ale RAID 5 sunt suportate de către multe sisteme moderne de operare (Linux, Windows Server etc.).



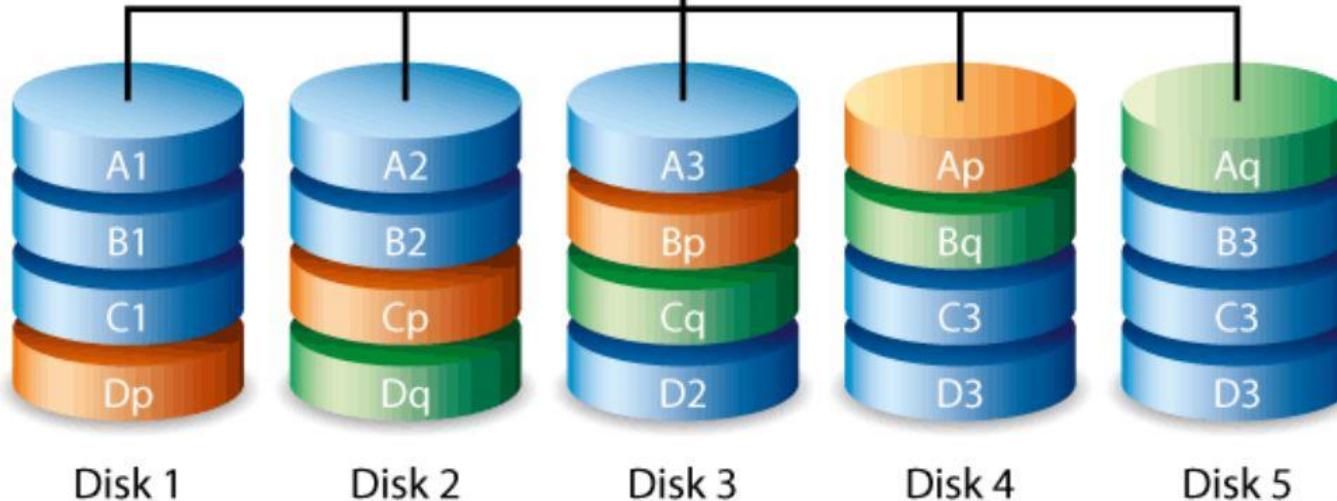
RAID 6

- Ceea ce diferențiază nivelul RAID 6 de nivelul RAID 5 este faptul că se calculează și se stochează paritate dublă pentru date.
- Sunt necesare **minim 4 discuri**, performanța scăzând datorită calculului dublu de paritate.
- Redundanța crește, aria suportând două discuri defecte.





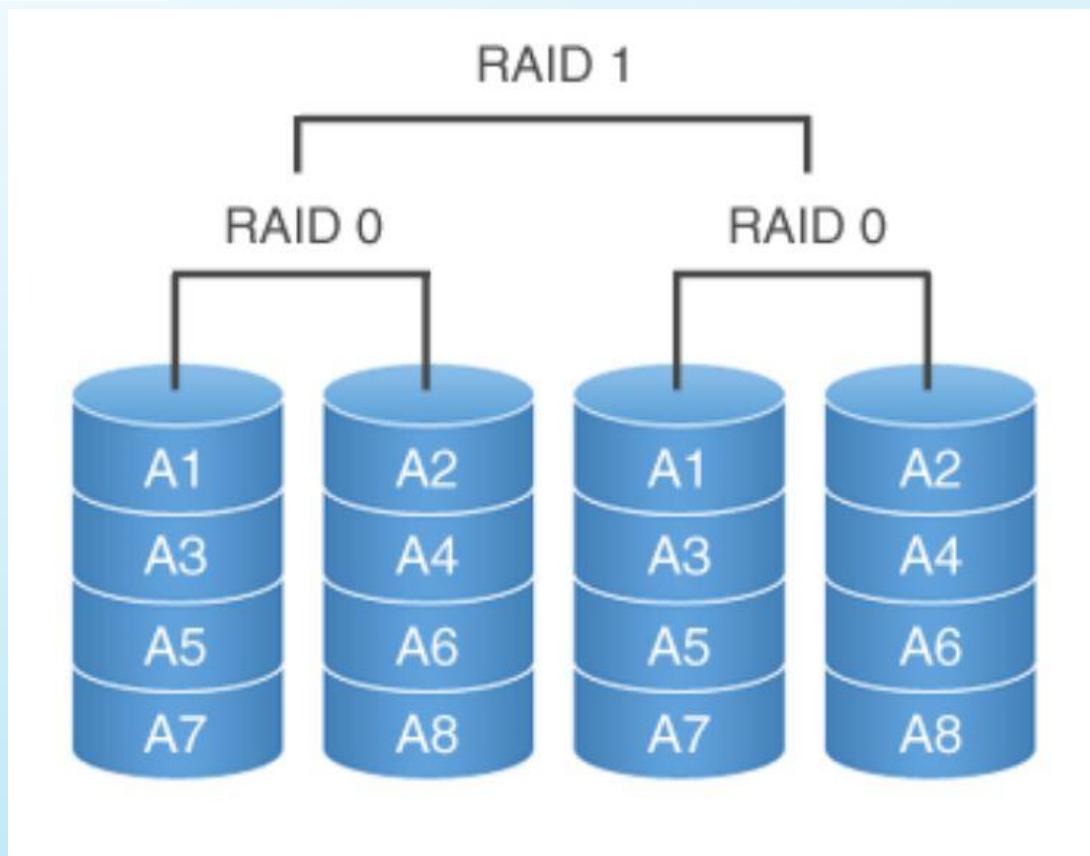
RAID 6



- Reconstituirea datelor poate fi un mare consumator de resurse, performanța fiind puternic afectată, dar redundanța dublă se compensează prin posibilitatea de amânare a procesului de rebuilding pentru un plus de viteză.
- Pentru a crea nivele etajate de RAID, oricare nivel poate fi combinat cu alt nivel, dar cel mai des sunt folosite nivelele 0 și 1.

RAID 01

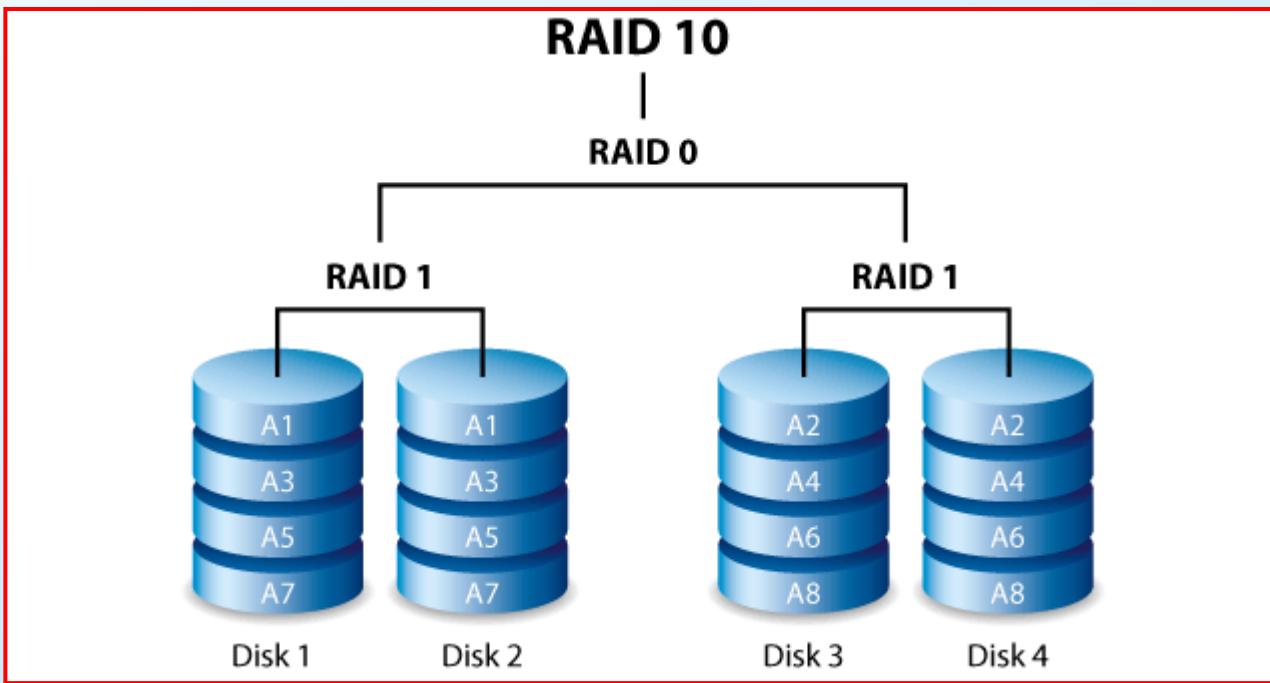
- Acest nivel dispune **minim 4 discuri** grupate în fâșii (oglindite),
- Înlătură neajunsurile și îmbunătățește performanțele, însă crește complexitatea.
- RAID 0+1 creează un al doilea set de fâșii oglindite cu un set primar de fâșii de discuri.



- Matricea continuă să funcționeze cu unul sau mai multe unități de stocare afectate în aceeași oglindă stabilită, dar dacă driverul cedează de pe ambele părți ale oglinziei, datele de pe sistemul RAID sunt pierdute

RAID 10

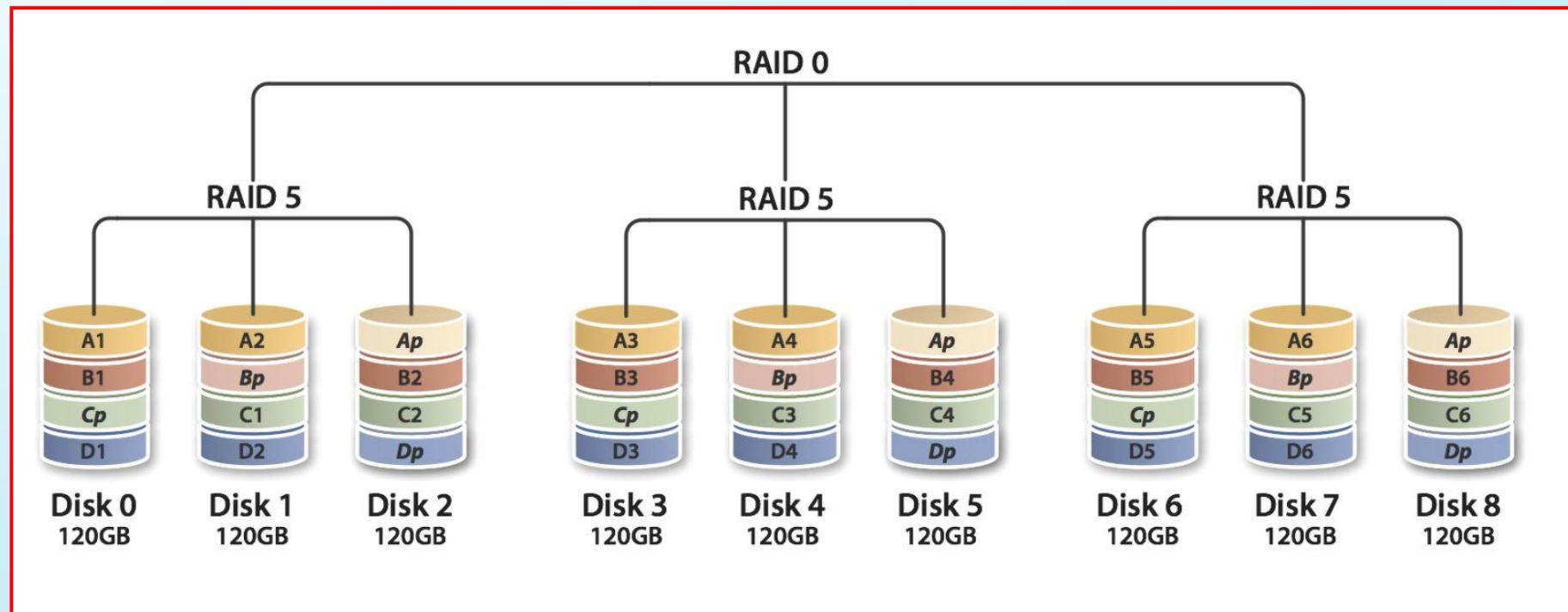
- Acest nivel dispune **minim 4 discuri** grupate în fâșii (oglindite), înlătură neajunsurile și îmbunătățește performanțele, însă crește complexitatea.
- Diferența față de RAID 0+1 este că RAID 1+0 creează discuri cu date întrețesute de la o serie de unități de stocare în oglindă.



- Într-o situație de eșuare, RAID 1+0 funcționează mai bine, deoarece toate celelalte discuri continuă să fie utilizate.
- Matricea poate susține mai multe discuri pierdute, atât timp cât oglinda nu pierde toate unitățile sale

RAID 50

- RAID50 combină feliile de blocuri de la nivelul RAID0 cu paritatea distribuită de RAID5
- Aceasta este de fapt o matrice RAID 0 întrețesută de-a lungul elementelor RAID 5.
- Pentru a realiza un nivel hibrid RAID 50 avem nevoie de **cel putin 6 discuri**.
- Avantajul acestei configurații este că ca pot eșua câte un drive din fiecare set, fără să se piardă date.

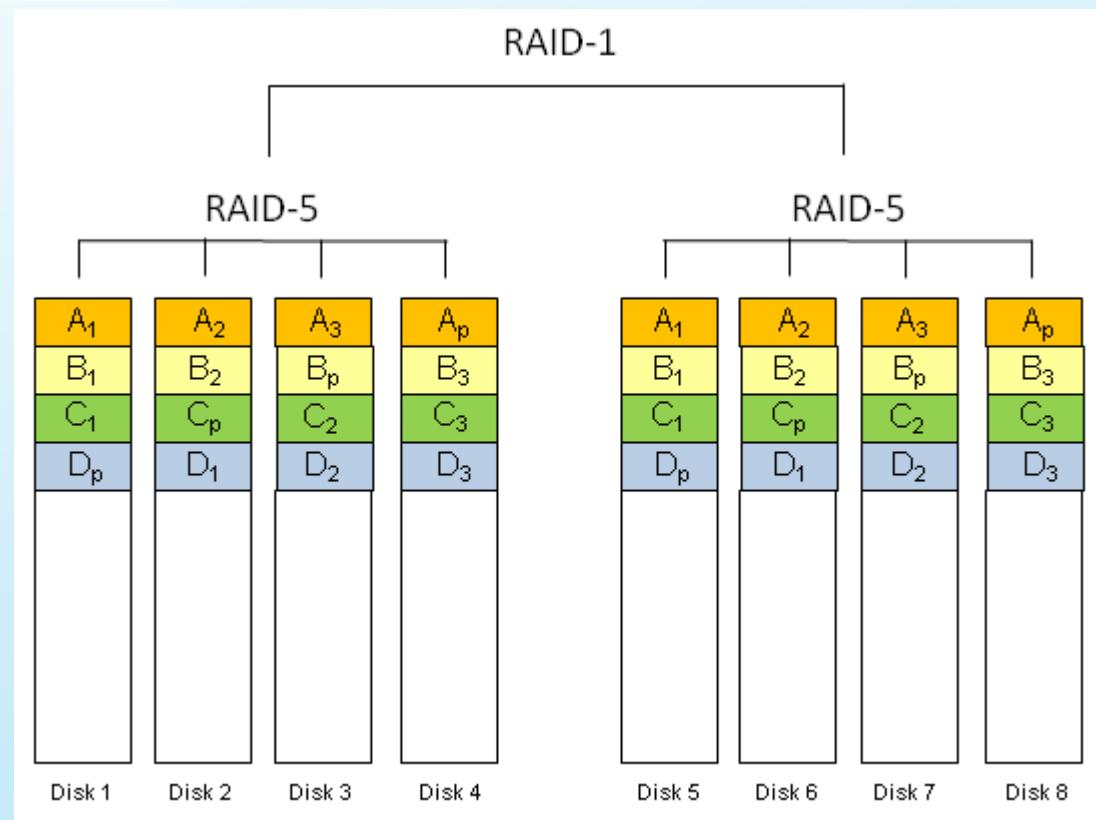


- Unitățile rămase în acel set devin un punct slab pentru întreaga matrice până când discul defect este înlocuit.
- Este suficient să se mai defecteze un singur disc (pe lângă cel inițial) și întreg sistemul cedează, datele stocate în întreaga matrice fiind pierdute.
- Timpul petrecut în procesul de rebuilding reprezintă o perioadă de vulnerabilitate a setului RAID.



RAID 51

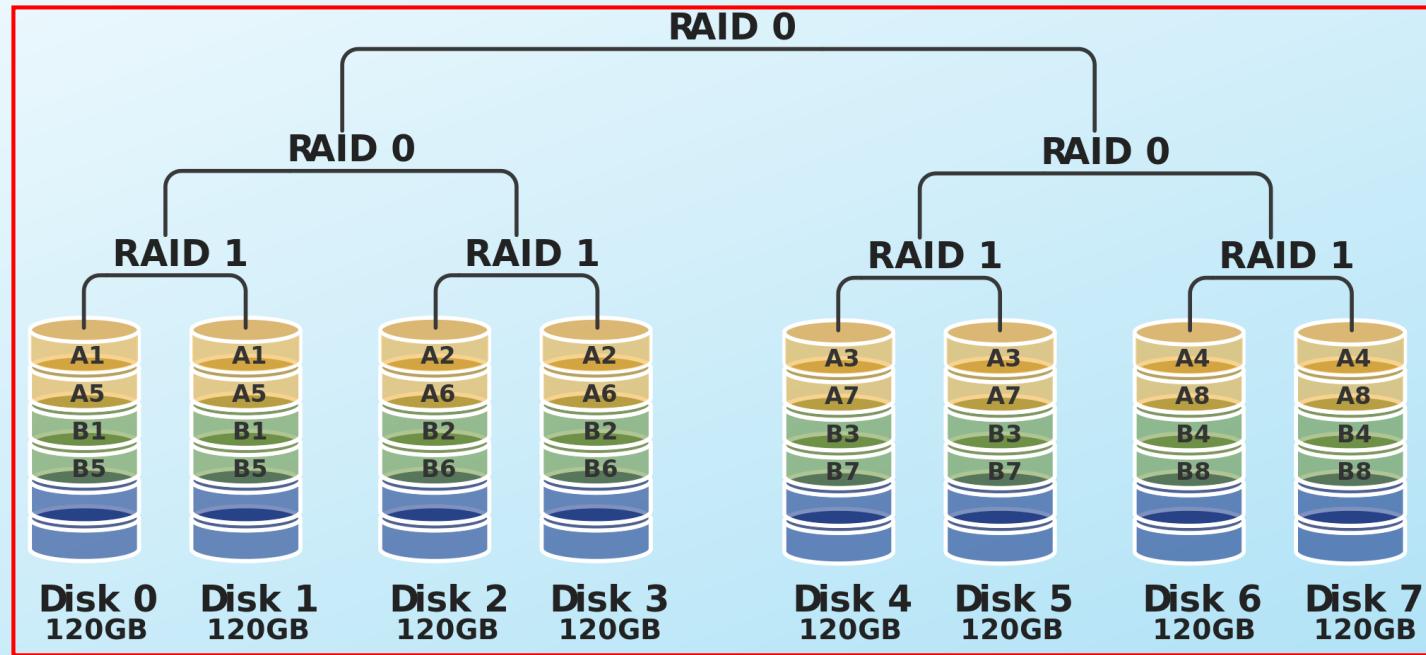
- Acest nivel hibrid este o matrice formată din două matrice RAID 5, care sunt oglindite una față de cealaltă.
- În general această configurație este folosită astfel încât fiecare set de RAID 5 se află pe un controller separat.
- Scrierile și citirile sunt echilibrate în ambele matrice RAID 5.



- Unele controlere suportă RAID 51 pe mai multe canale pentru a ține sincronizate părțile diferite.
- Această configurație poate susține eșecul oricărui disc din orice matrice, plus încă un disc suplimentar din cealaltă matrice înainte să existe pierderi de date.
- Spațiul maxim ce se poate găsi pe un RAID 51 este egal cu dimensiunea unui set individual RAID 5

RAID 100

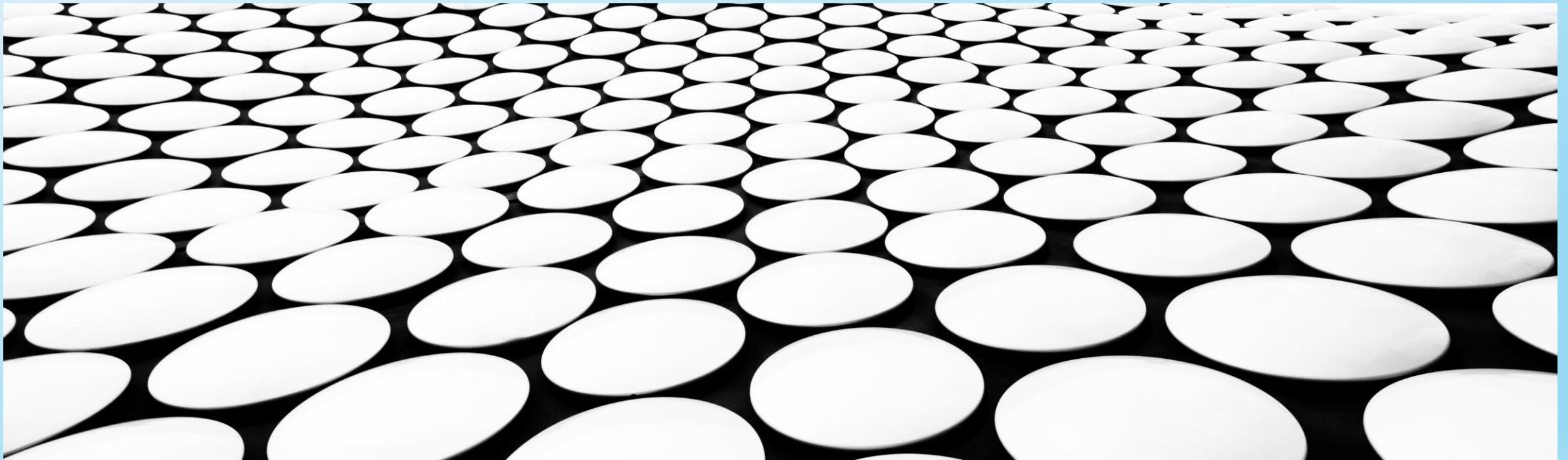
- Un raid 100, numit uneori și RAID 10+0, este o felie de tipuri RAID 10.
- Din punct de vedere logic el este de fapt o matrice RAID 10 implementată cu ajutorul software-ului RAID 0 peste hardware-ul de RAID 10.



- Principalul avantaj al RAID 100 față de un singur nivel RAID este răspândirea încărcării pe mai multe controllere, obținându-se astfel performanță mai bună la citiri aleatoare și atenuarea riscurilor pe matrice.
- RAID 100 reprezintă cea mai bună alegere când vine vorba de baze de date foarte mari, unde controllerele hardware de RAID limitează numărul de discuri fizice permise în fiecare matrice standard

ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2021-2022

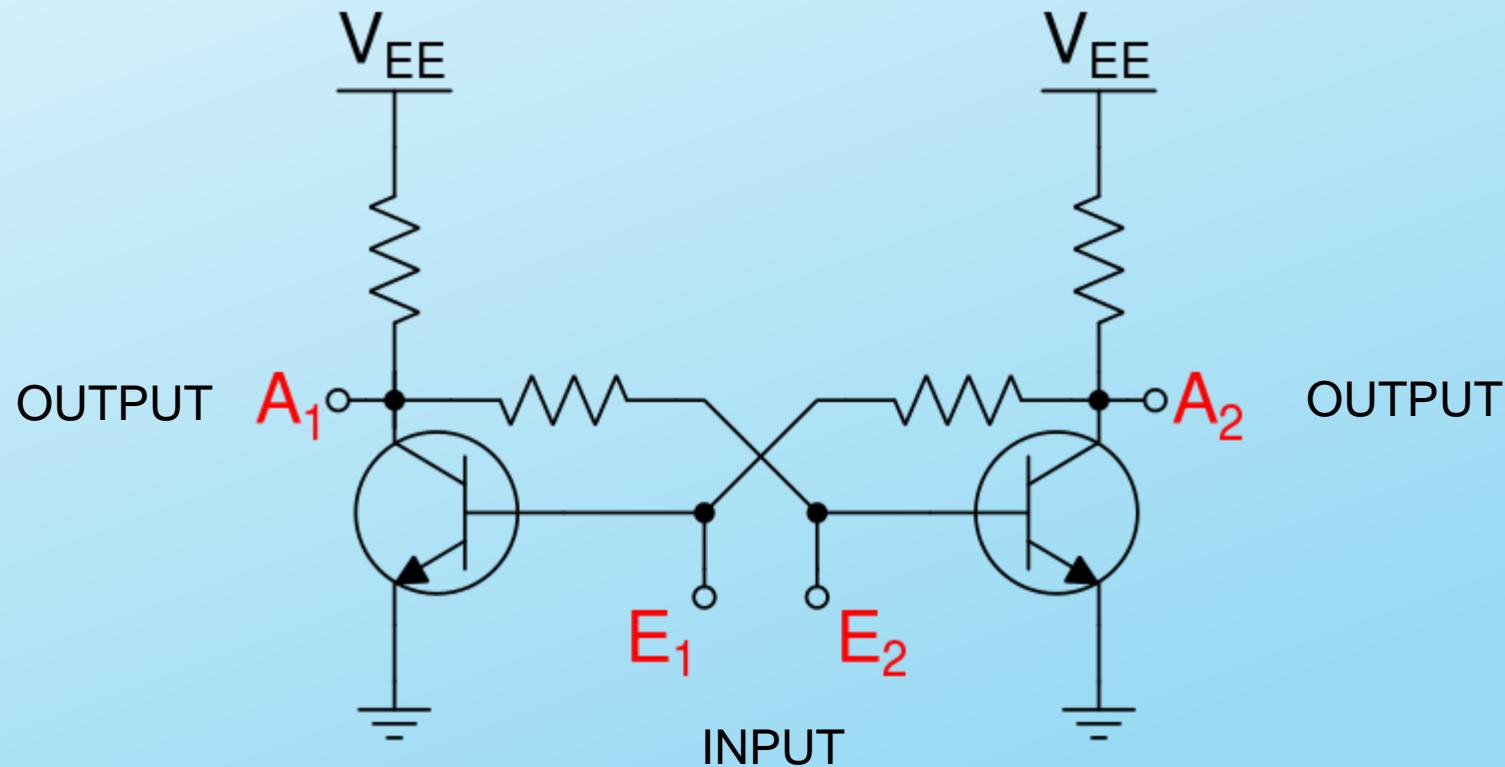


BLOCURI DE MEMORIE ELECTRONICA

CEA MAI SIMPLA ARHITECTURA DE MEMORIE RAM SCALABILA

Circuite elementare de memorie

- ### ■ Circuitul bistabil



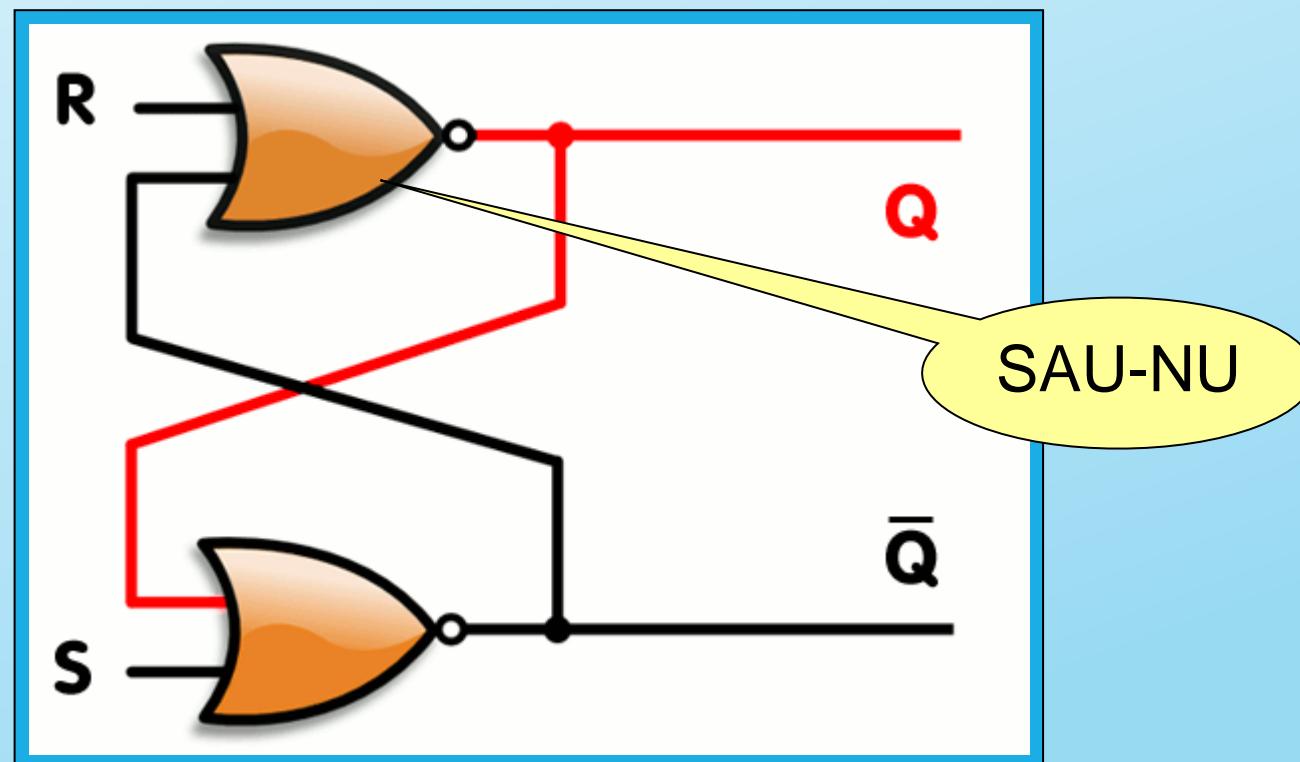
Exemplu de circuit cu tranzistori bipolari

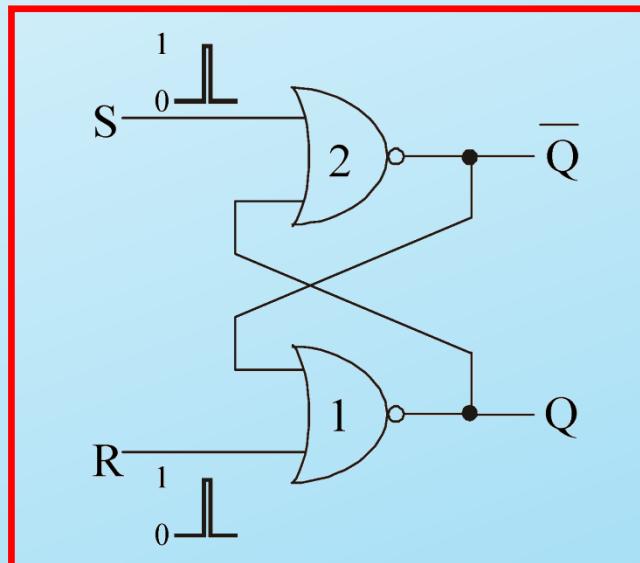
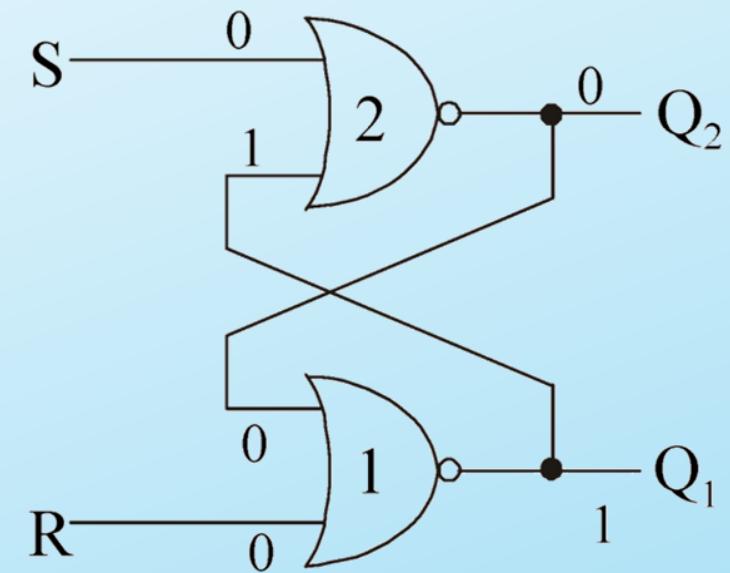
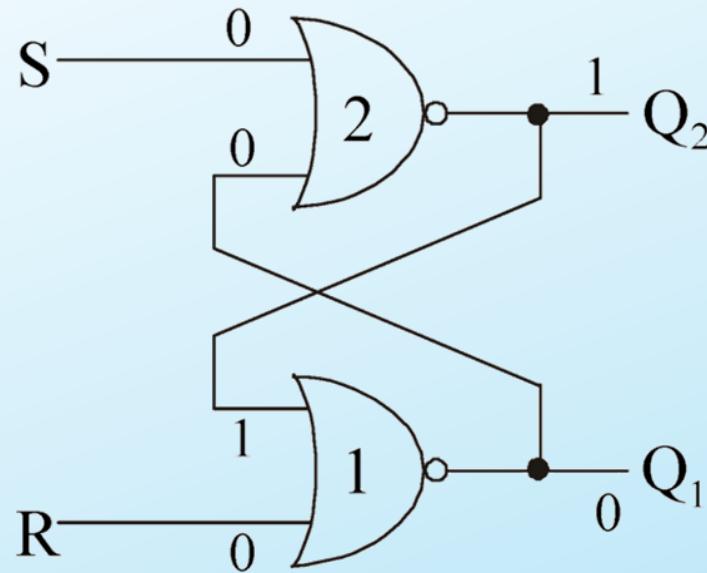
Schema logica echivalenta

INPUT:

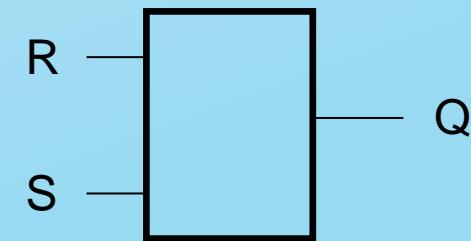
S – SET

R -RESET





- Un impuls pozitiv pe **S** aduce iesirea **Q1** in starea 1
- Un impuls pozitiv pe **R** aduce iesirea **Q1** in starea 0



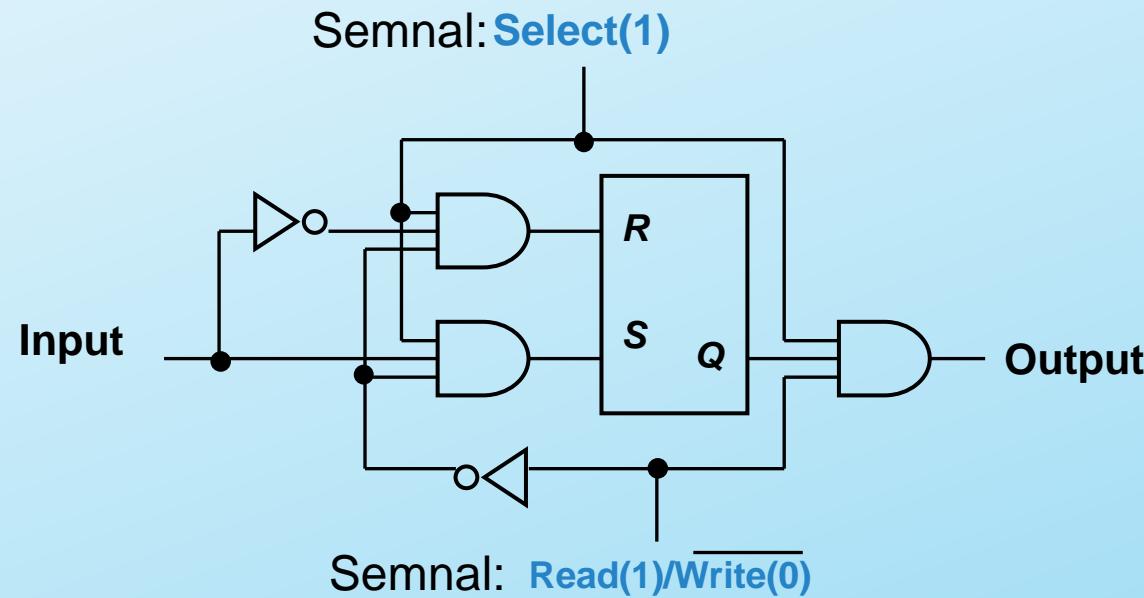
Echivalentul mecanic al memoriei



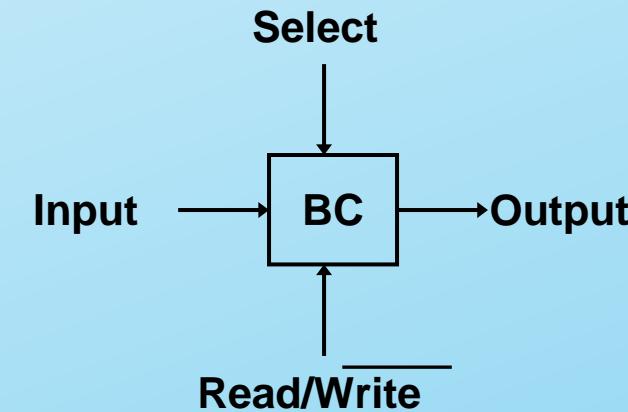


CELULA DE MEMORIE SRAM DE 1 bit

Diagrama logica pentru **celula de 1 bit**



Logic diagram



Block diagram

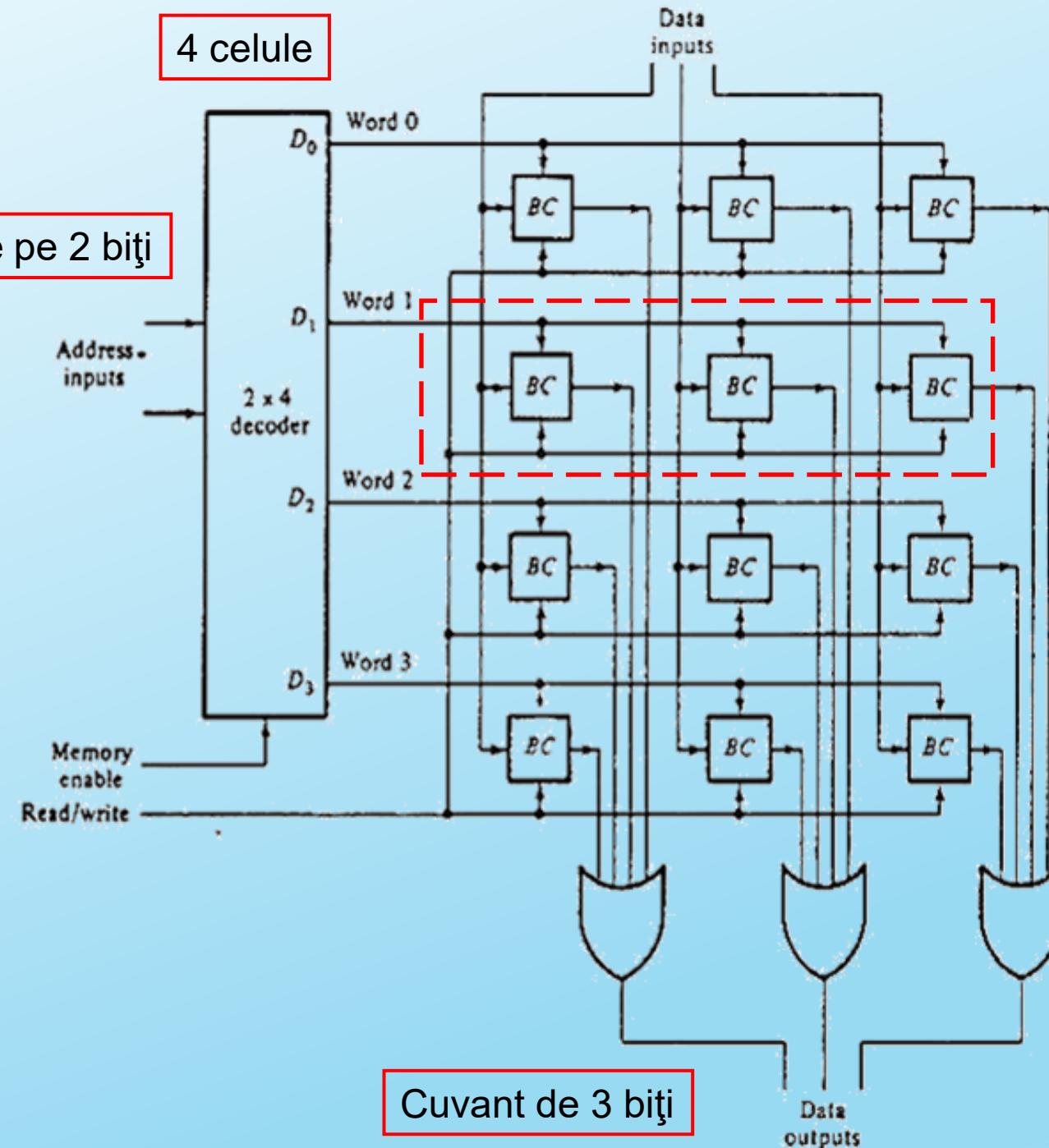
Matrice de celule de memorie 4x3 (4 locatii de 3 biti)

4x3 RAM

Biti din cuvant transferati in paralel

Cuvinte transferate succesiv (functie de adresa)

informatica de adresa se transforma in semnal de selectie



Bloc de memorie: 1K x 8-bit RAM (chip)

- $1K = 1024$ cuvinte $= 2^{10}$

pentru 1024 cuvinte este nevoie de o dimensiune de adresa de 10 biti

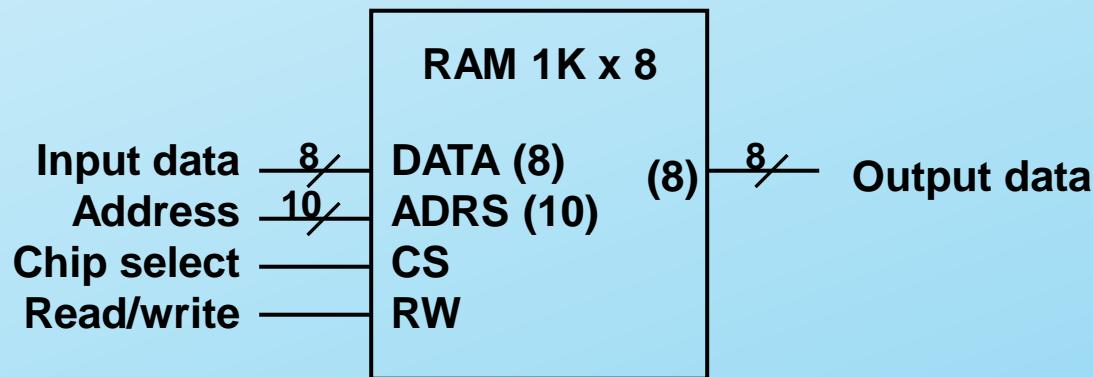
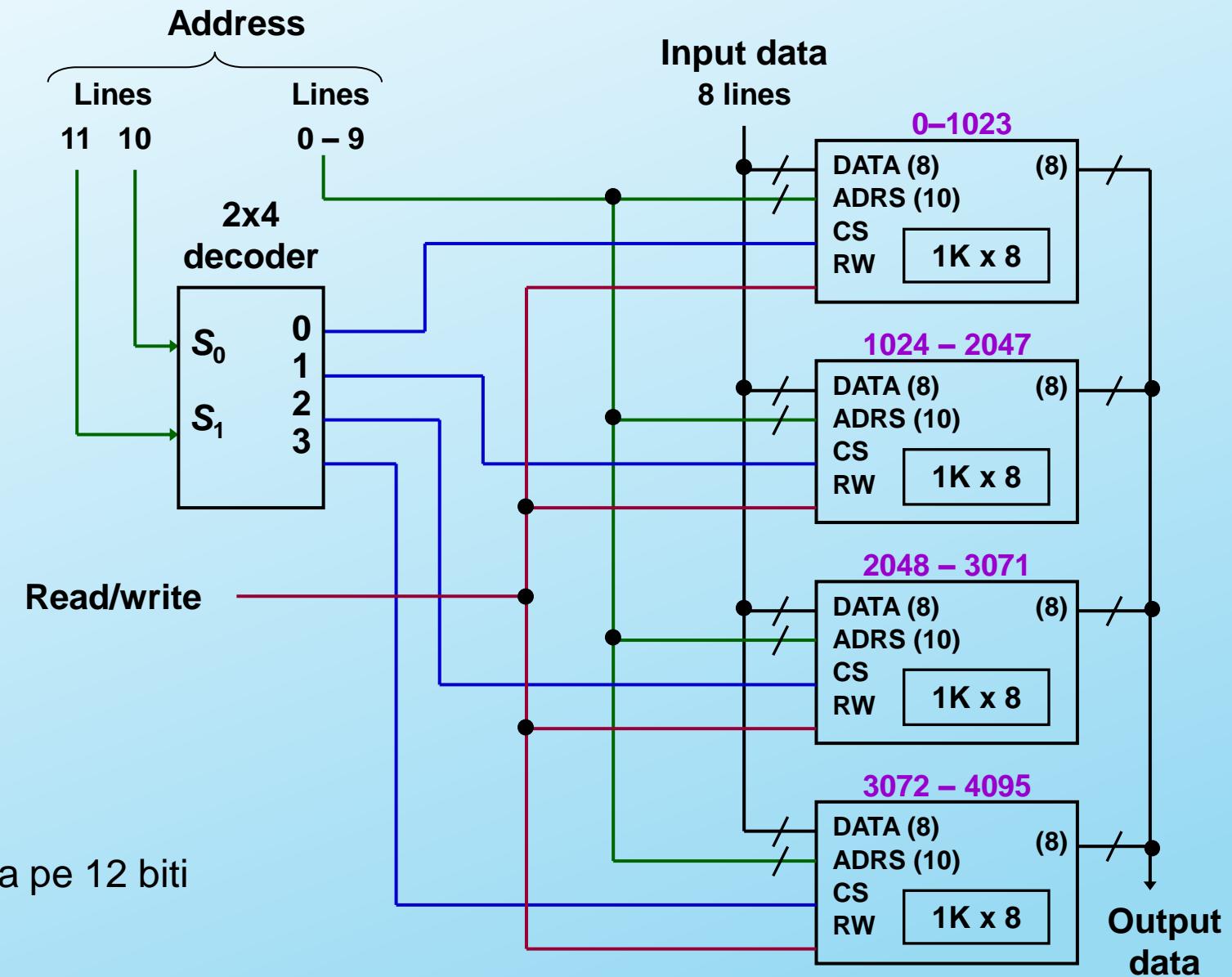


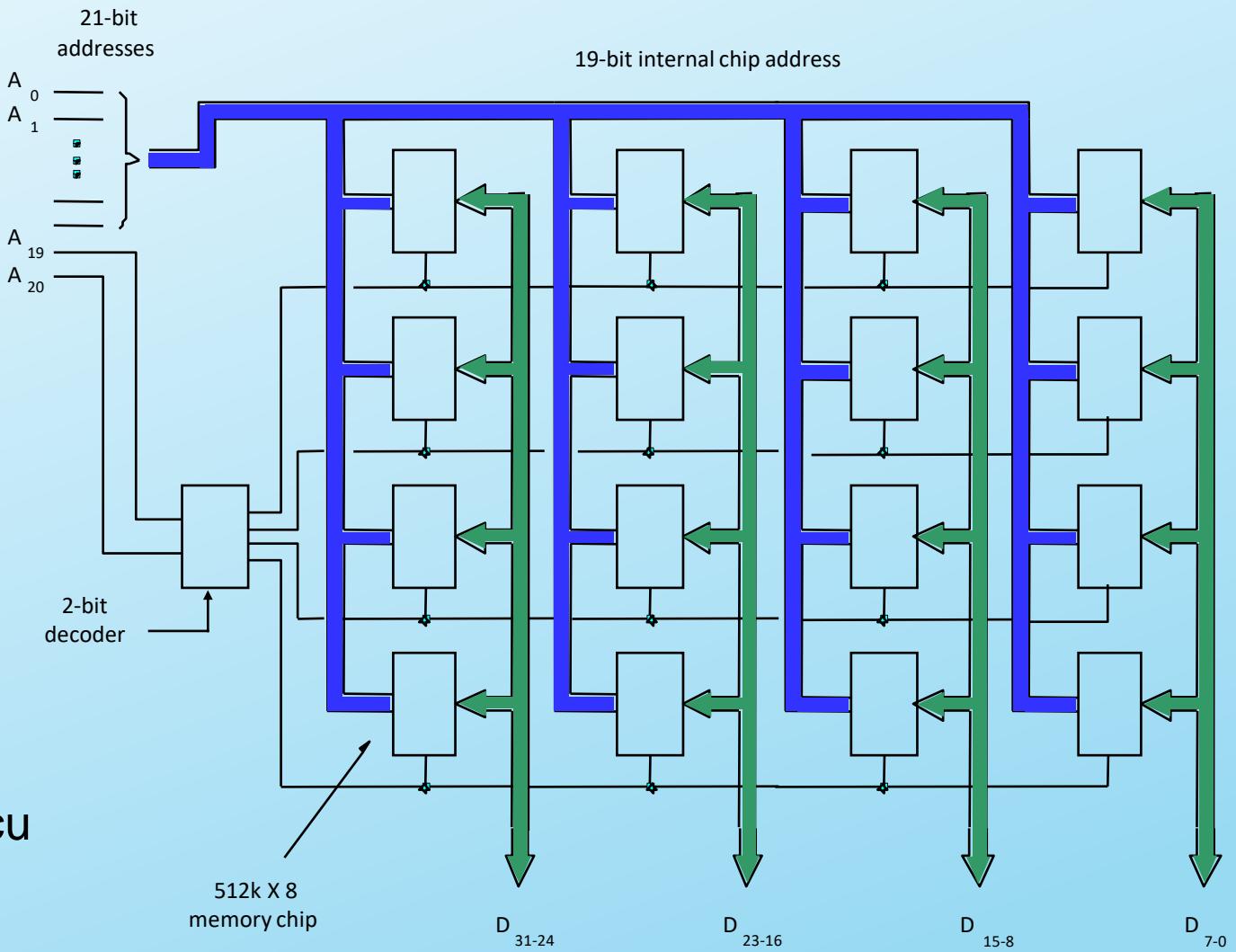
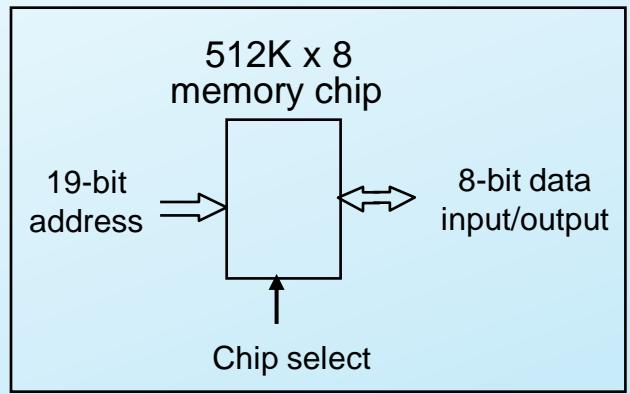
Diagrama bloc a unui
chip RAM de 1K x 8b





Blocurile OR de colectare sunt omis pentru simplificarea desenului

Tema: redesenati schema, incluzand si blocurile de colectare



Alt exemplu:
Modul de memorie de **2M × 32b** asamblat cu
512K × 8b static memory chips.

Semnalul: Read/Write este omis

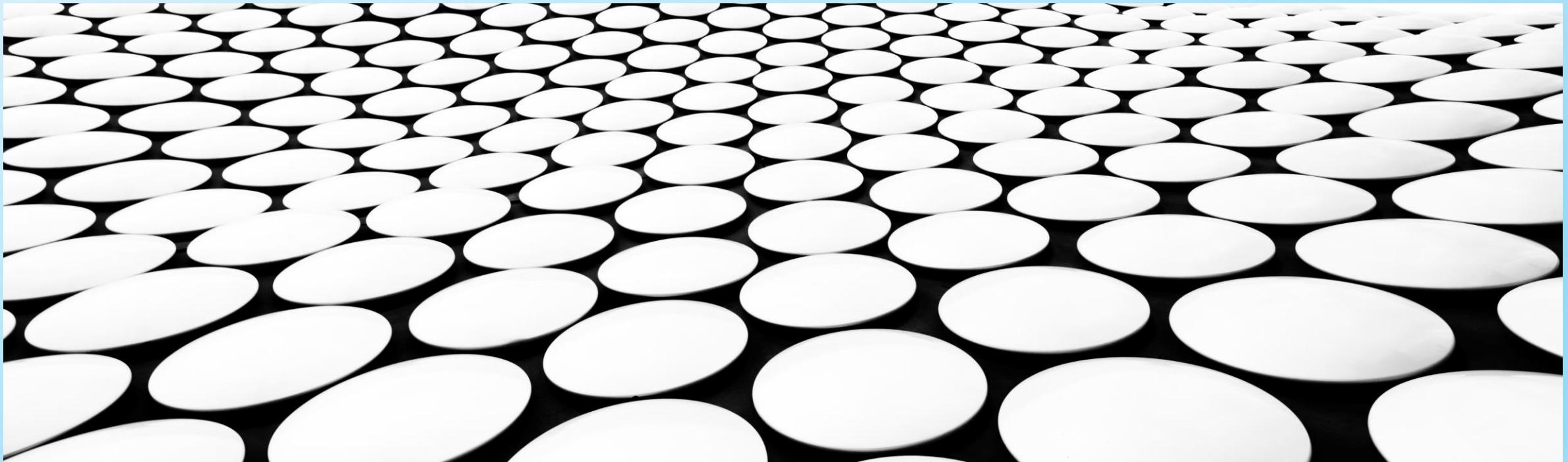
Tema: redesenati schema, incluzand toate blocurile si semnalele omis

TEMA

Desenati schema unui modul de memorie de $2G \times 64b$,
folosind cipuri de memorie de $512M \times 32b$

ARHITECTURA SISTEMELOR DE CALCUL

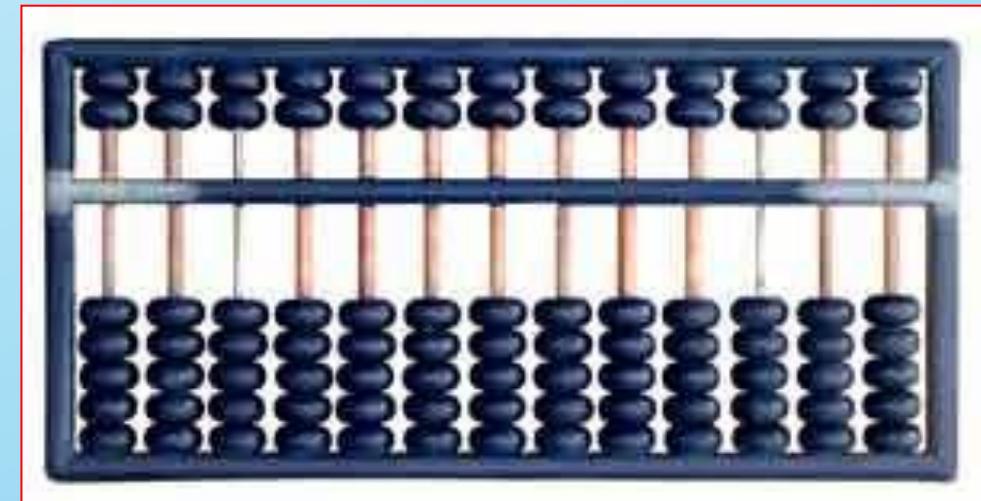
UB, FMI, CTI, ANUL III, 2022-2023



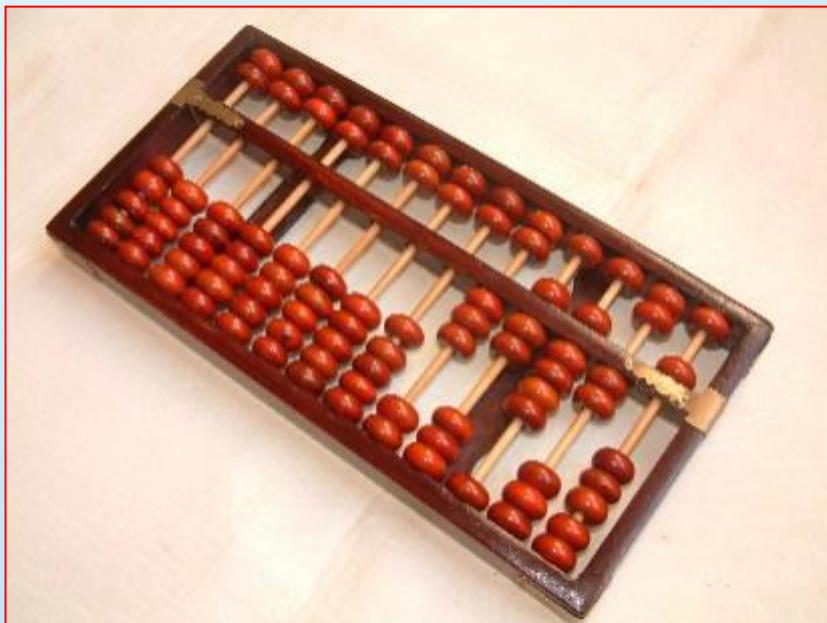
Istoria sistemelor de calcul

CALCULATOARE MECANICE

- Abacul
 - Prima datare 3000 î.Hr
 - Utilizat de babilonieni
 - Este cel mai vechi calculator digital mecanic



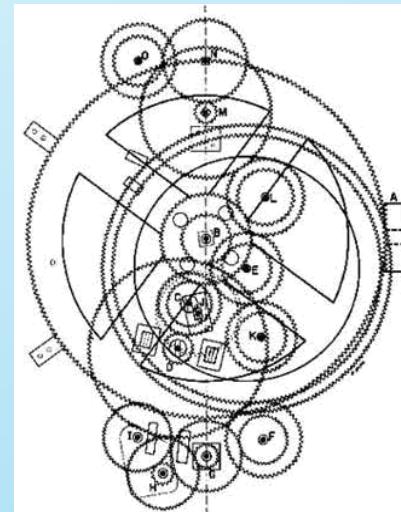
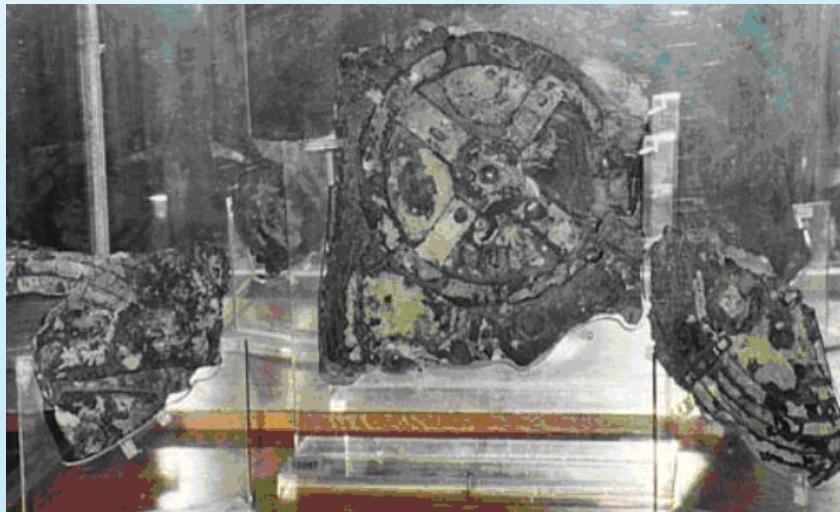
Abac asiatic



Abac roman



PRIMELE MECANISME DE CALCUL



- Calculatorul antic din **Antikythera**:
 - Calcula cu precizie cicluri astronomice
 - Determina data la care avea loc urmatoarea olimpiada
 - Avea roti dintate si cilindri de bronz (80 bucati)
 - A fost descoperit in **1901** de catre un pescar grec

Calculatorul din Antikythera, reconstituit

Este cel mai vechi mecanism complex cunoscut in istorie.
Mecanismul din Antikythera are o dimensiune asemanatoare cu
a unui ceas de perete Victorian.

Se presupune ca mecanismul ar fi fost realizat in **secolul II i.e.n.**

Mecanismul afisa Soarele si Luna, dar si cele cinci planete
vizibile cu ochiul liber de pe Pamant: Mercur, Venus, Marte,
Jupiter si Saturn.

Se presupune ca avea mai multe cadrane in care erau afisate
fazele Lunii, eclipsele solare si lunare.



Primul calculator mecanic

1623: Ceasul calculator (Calculating Clock)

- astronomul si matematicianul **Wilhelm Schickard** [Germania]
- Adunare si scadere de numere cu 6 digits
- Tehnologie cu clichei si cremaliera



Anterior, Leonardo da Vinci schitase planurile unui calculator care au fost suficient de complete si corecte dupa care inginerii moderni au reusit sa construiasca un calculator.

1642: Prima masina (mecanica) de calcul: **pascaline**

- Blaise Pascal
- Pascalina efectua adunari si scaderi.

In decurs de 10 ani au fost construite mai mult de 50 de Pascaline, devenind prima masina de calcul folosita la scara larga in afaceri.

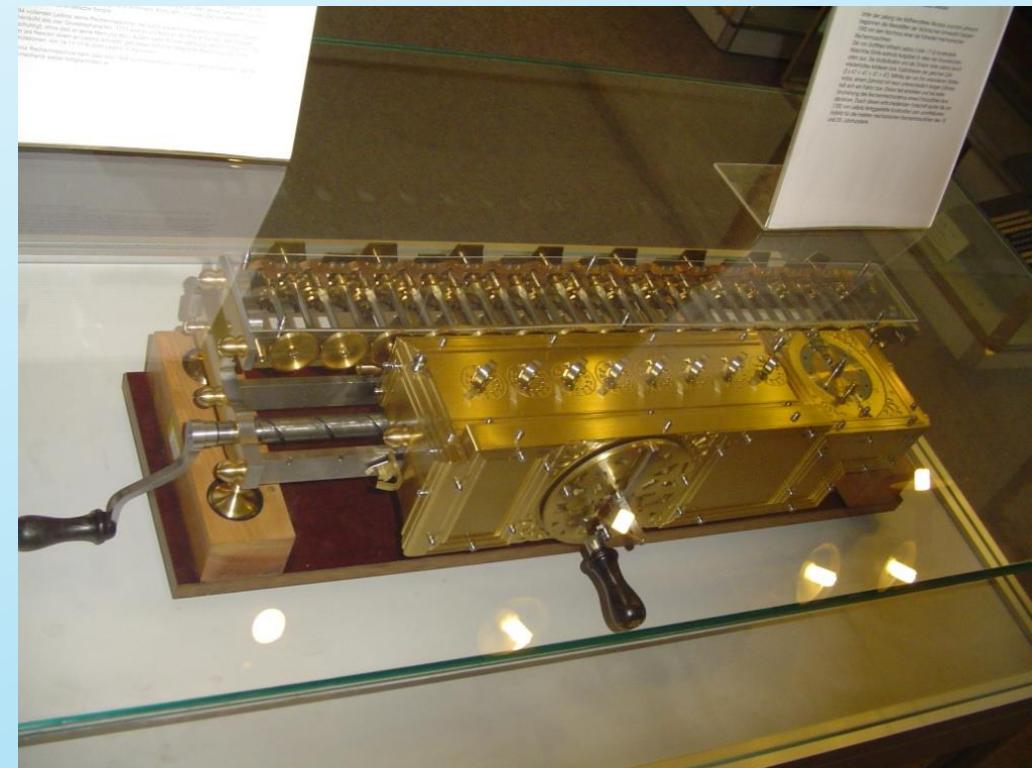


Prima masina universală pentru calcul digital

1671 Gottfried Leibniz ("Stepped Reckoner")

Poate efectua cele **patru operatii** aritmetice

Rezultatul poate avea maxim 16 digits



Leibniz a fost cel mai mare sustinator al sistemului numeric binar.

Masini de calculat utilizate in activitati comerciale

- Dezvoltate in **sec. XVIII**
- Operatii: adunare, scadere, multiplicare
- Numere cu 4-6 digits
- Sursa de energie: **manivela**

1820: Charles Xavier Thomas de Colmar, un francez inventiv, a construit Arithmometrul, primul dispozitiv de calcul comercial produs in masa.

Cu un aritmometru se puteau efectua operatiile matematice elementare, fiind aproape un secol cea mai eficienta masina de calcul.

INSTALATII MECANICE DE CALCUL

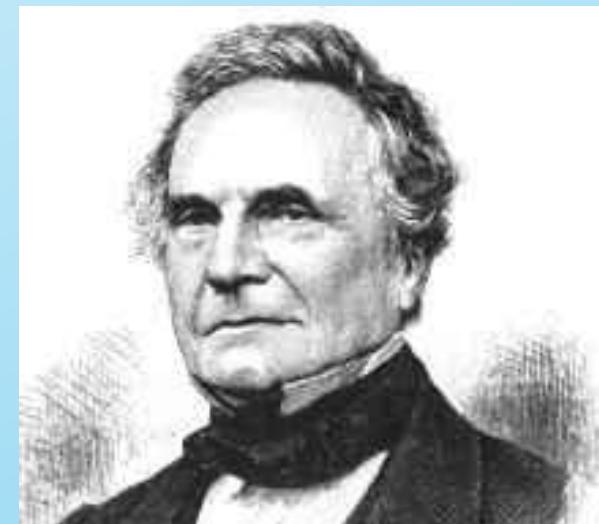
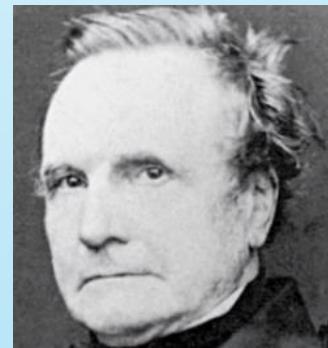
(contineau mai multe masini/masinari)

Razboiul de tesut controlat (programat)

- Joseph-Marie Jacquard (1752-1854) [Franta]
- Foloseste cartele perforate
- Sursa de energie: o masina cu abur.

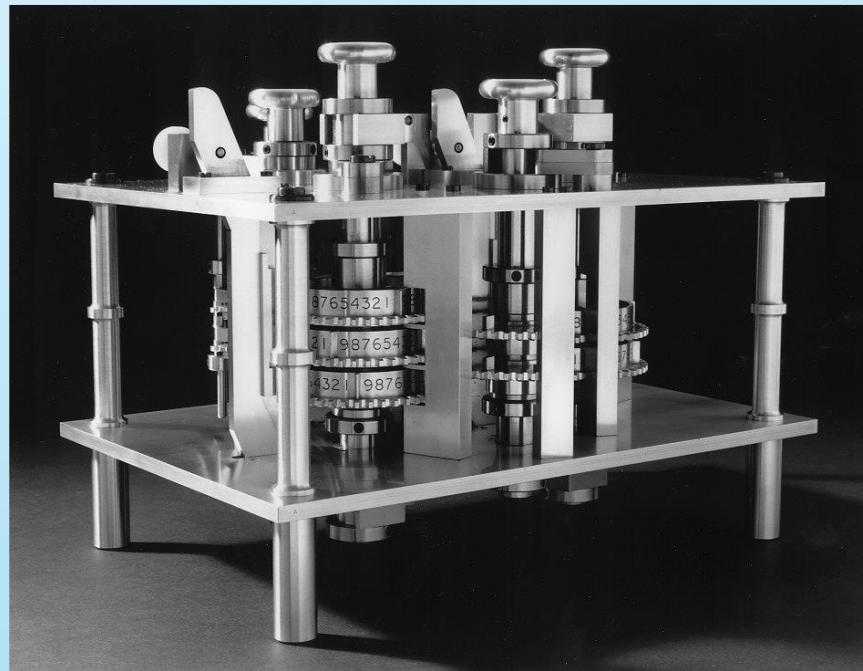


- Charles Babbage: “Tatal calculatoarelor”
 - 1791-1871 [Anglia]
 - Introduce noțiuni fundamentale în teoria mașinilor de calcul:
 - ROM,
 - Programare
 - CPU
 - Proiectează și realizează calculatoare (mașini de calcul de complexitate crescută);
 - Realizează **prima mașină** de calcul **programabilă**



- Masina pentru diferențe de ordinul 6

- 1828-32
- Numere cu 20 de digits
- Rezolvă ecuații polinomiale



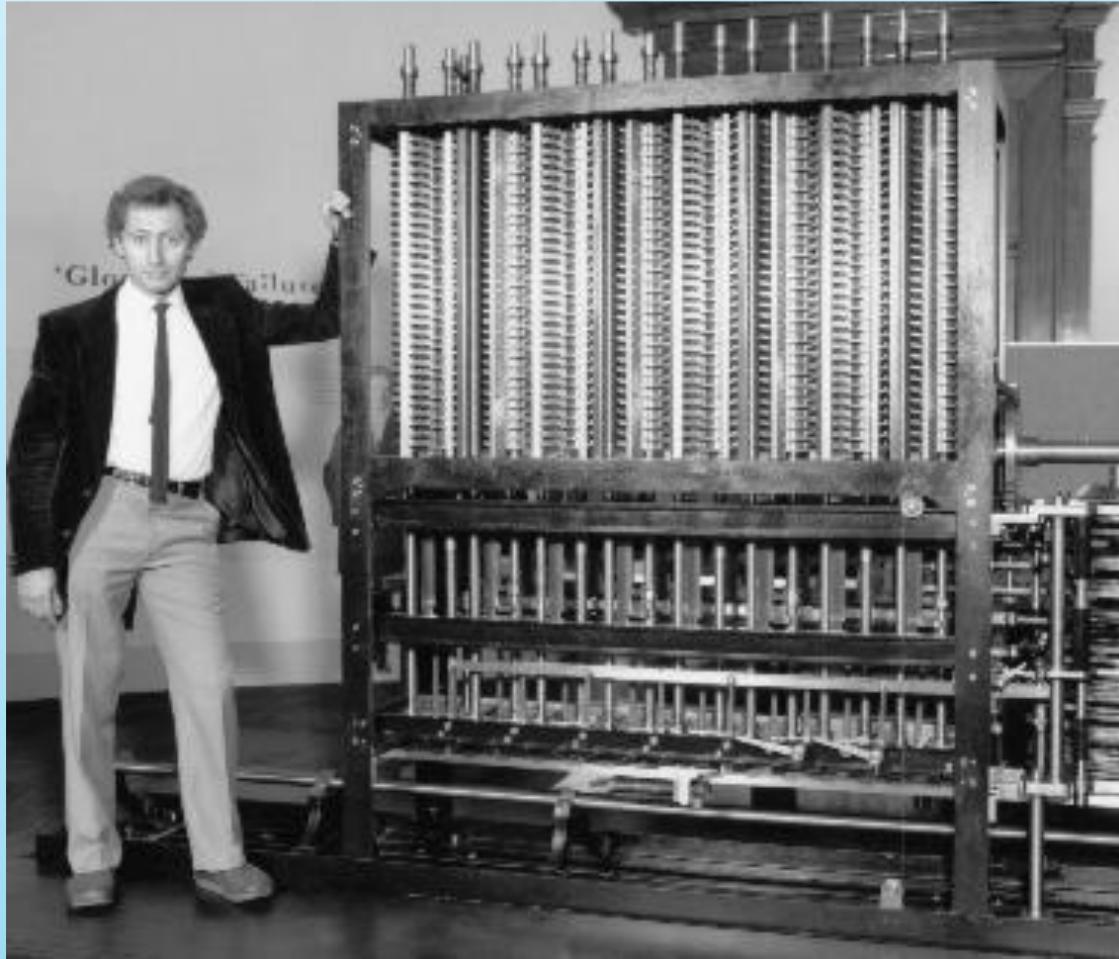
Analytical Engine a lui Charles Babbage

- Program stocat pe cartele perforate
- CPU de 40 digits
- Stocheaza 100 numere
- Citeste cod (program) si date
- Efectueaza adunari si scaderi in 3 sec
- Efectueaza inmultiri in 2-4 min



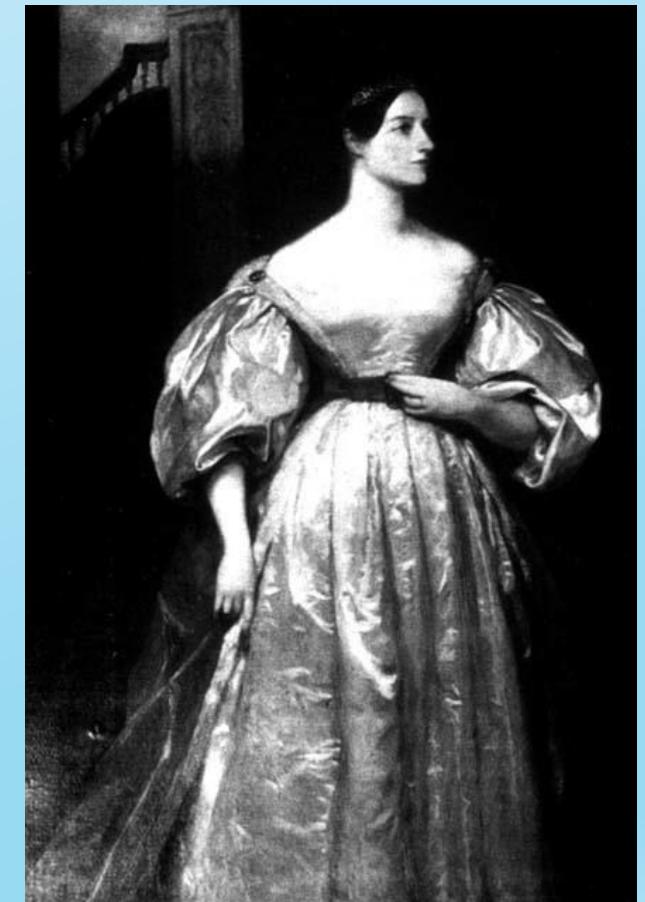
Unul dintre proiecte a fost prea complicat, nu au putut fi realizat in epoca sa.

Proiectul sau fundamental a putut fi construit abia in **1961**, la Science Museum din Londra



Lady Augusta Ada, Contesa de Lovelace (1815-1852)

- Primul programator
 - Fiica poetului Byron
- Lucreaza cu masinile Charles Babbage (Analytical Engine)
- Scrie cod pentru manipularea datelor, folosind cartele perforate



Calculatoare electro-mecanice

Calculator tabular pentru recensamant

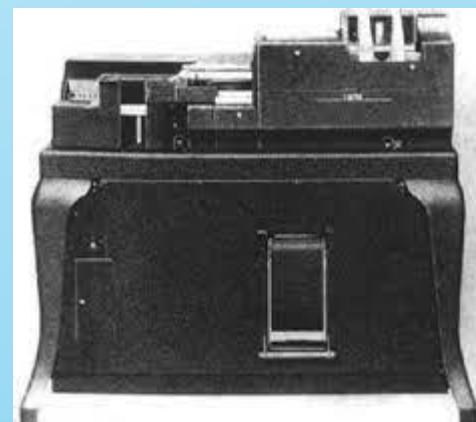
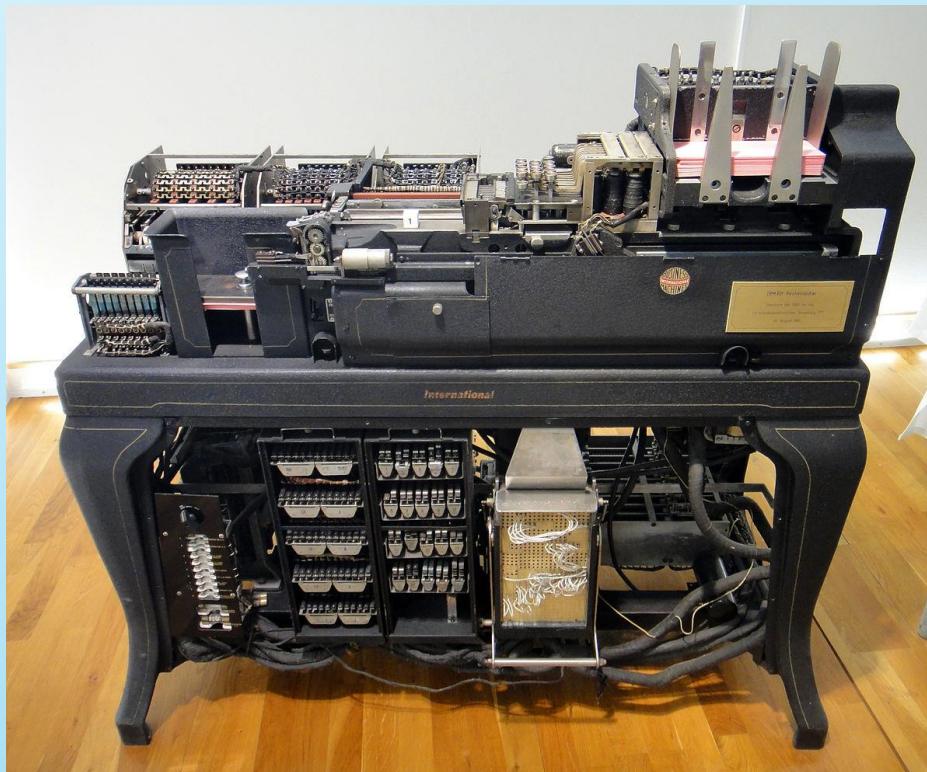
- Herman Hollerith (1860-1929)
- Foloseste cartele perforate
- Recensamantul din **1890** din SUA
- prin utilizarea calculatorului prelucrarea a durat **1 an**, spre deosebire de recensamantul anterior, unde durata a fost de **8 ani**

L	A	B	C	A	B	C	L	C	n	G	A	C	C	S	M	I	H	M	W	A	G	E	F	u	d	
Cn	D	E	F	D	E	F	L	C	n	G	A	C	C	S	M	I	H	M	W	A	G	E	F	u	d	
Le	G	H	I	G	H	I	L	C	n	G	A	C	C	S	M	I	H	M	W	A	G	E	F	u	d	
Cm	K	L	M	K	L	M	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
CS	N	O	P	N	O	P	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
LS	Q	R	S	Q	R	S	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Ka	*	*	*	*	*	*	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	
RN	*	*	*	*	*	*	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	
QC	g	h	i	g	h	i	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	
AV	*	*	*	*	*	*	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	
Sc	*	*	*	*	*	*	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	

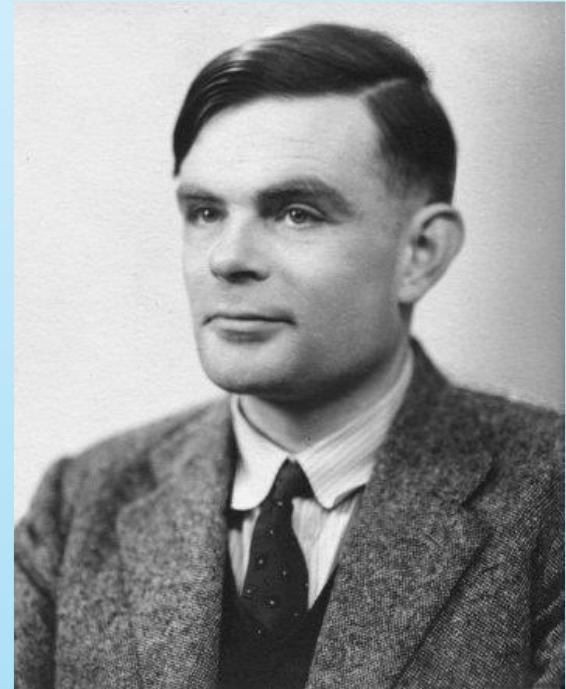


■ Primul calculator electric:

- **1901 (IBM 601)**
- Cu relee si tuburi electronice
- Stocare pe cartele perforate
- Operatii de multiplicare finalizate intr-o secunda



- Alan M. Turing (1912-1954, Anglia)
 - Tatal stiintei moderne a calculatoarelor
 - Turing a elaborat in detaliu conceptele de baza ale unei masini universale de calcul
 - **1936**: Masina Turing (a-machine)
 - Este un dispozitiv teoretic

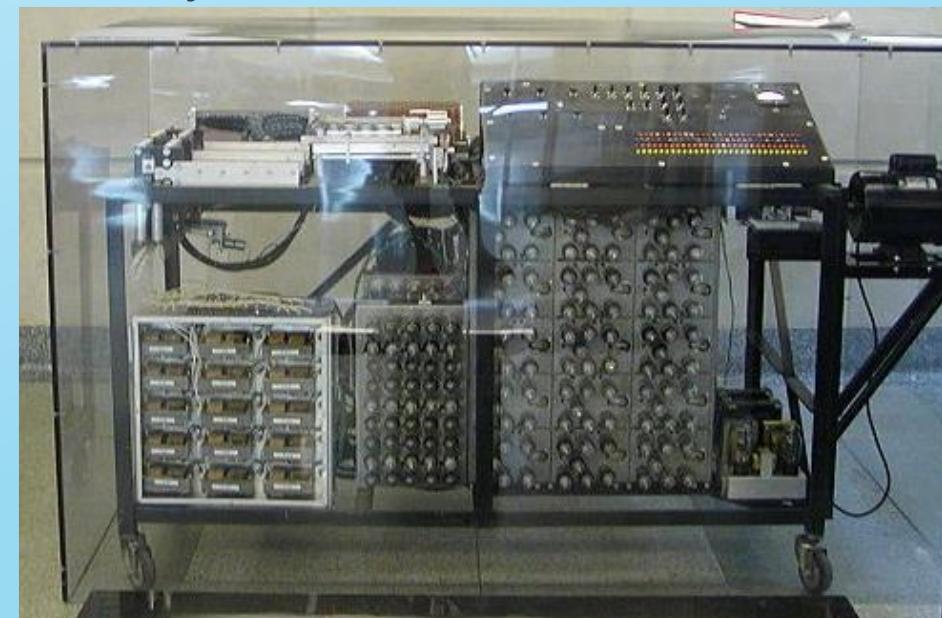
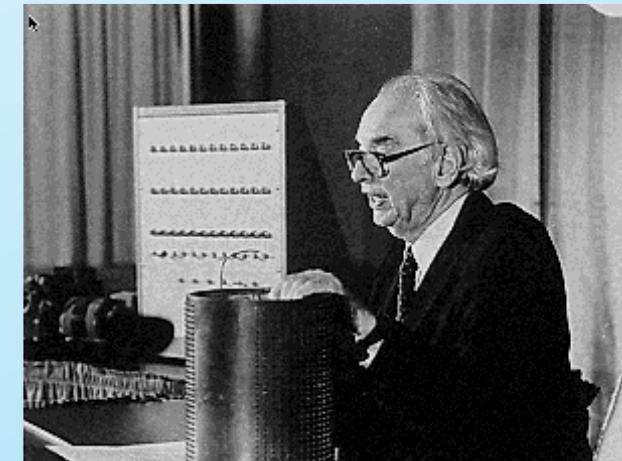


Calculatoare electronice

The Atanasoff-Berry Computer (ABC)

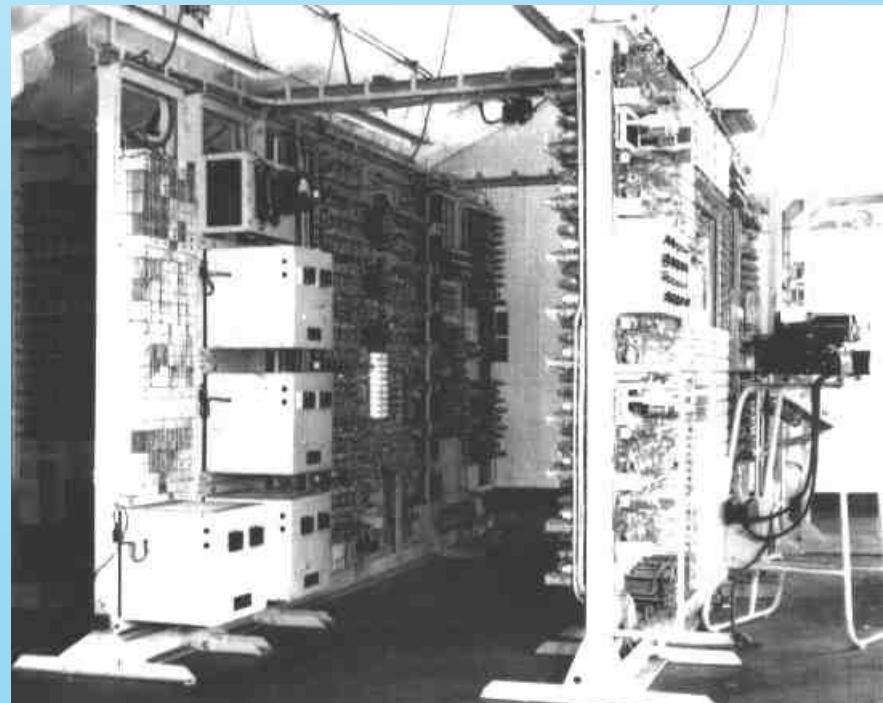
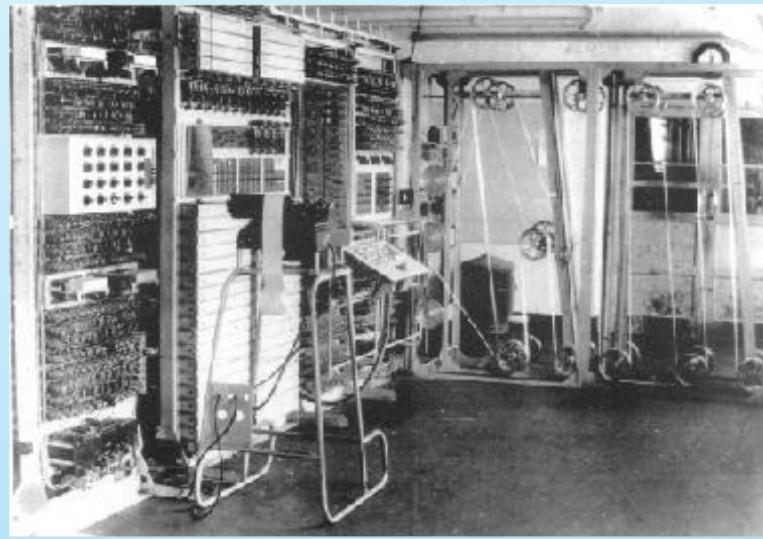
- **1939:** Primul calculator electronic digital

- Cu tuburi electronice (300 tuburi)
- Primul ALU electronic
- Memorie: 3000 b, pentru date
- Nu era programabil!
- Nu contineau elemente active mecanice sau electro-mecanice
- Putea rezolva sisteme de ecuatii liniare
- John Vincent Atanasoff (1903-1995), Physics Prof. at Iowa State University,
- Clifford Berry (1918-1963), PhD student of Dr. Atanasoff's



Colossus [Anglia]

- **1943**: Bletchley Park's
- A fost construita o serie de calculatoare sub aceasta emblema
- Intrare: cititor optic de banda de hartie perforata
- Iesire: memorie buffer cu relee si masina de scris electrica, pe rola de hartie
- Ceas intern **5kHz**
- **2500** de tuburi
- **4,5 kW**
- Asistenta pentru spargerea codurilor de criptare



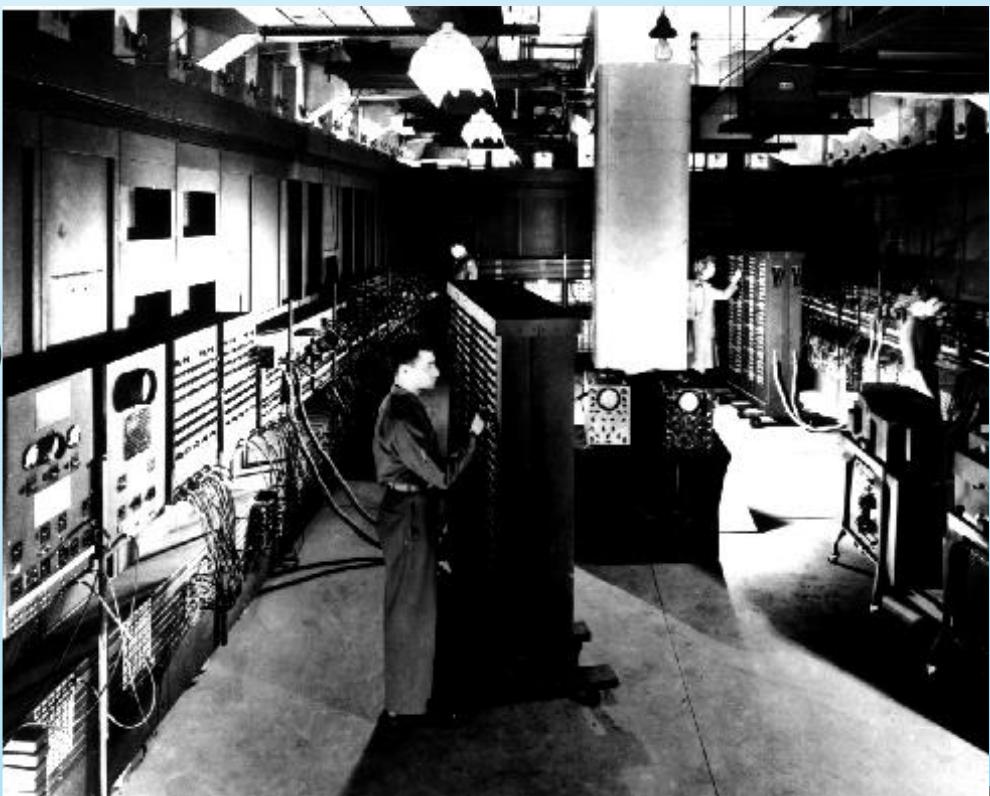
Harvard Mark I

- **1944**: Harvard, IBM
- Automatic Sequence Controlled Calculator
- General purpose **electromechanical computer**
- Calcule balistice
- Simulari numerice pentru bomba atomica (von Neumann)
 - 765 000 componente electromecanice
 - Numere cu 23 digits
 - inmultirea: 6 sec.
 - Impartirea : 15 sec.



ENIAC

- 1946: Moore School, University of Pennsylvania,
- Electronic Numerical Integrator and Computer (ENIAC)
- Integral electronic digital
- 30 tone
- 18 000 tuburi
- 100 000 calcule / sec
- Proiectat initial pentru calcule de artillerie.
- John Presper Eckert (1919-1995) and John Mauchly (1907-1980) of the University of Pennsylvania, Moore School of Engineering



Prima generatie de calculatoare (electronice) comerciale

- 1945-1958
- CPU cu tuburi electronice
- Programe stocate pe cartele perforate
- Date stocate pe sisteme cu capacitoare
- I/O cartele perforate
- Fiecare tip de calculator avea limbajul propriu de programare

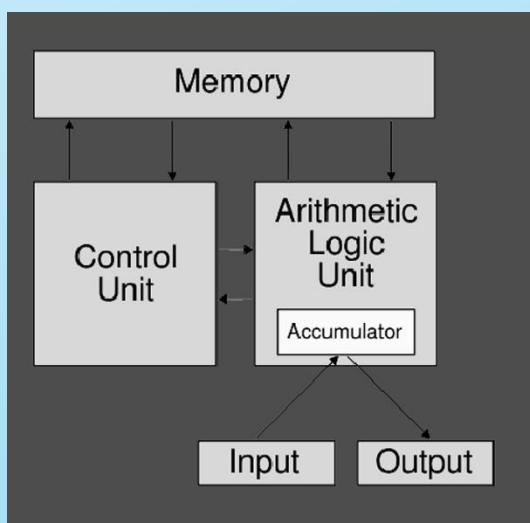
First Generation: Von Neuman Machine



- John von Neumann (1903-1957)
 - Studii de matematica: Budapest si Berlin
 - Princeton University
 - Contributii la dezvoltarea mecanicii cuantice

- Incepand din 1948

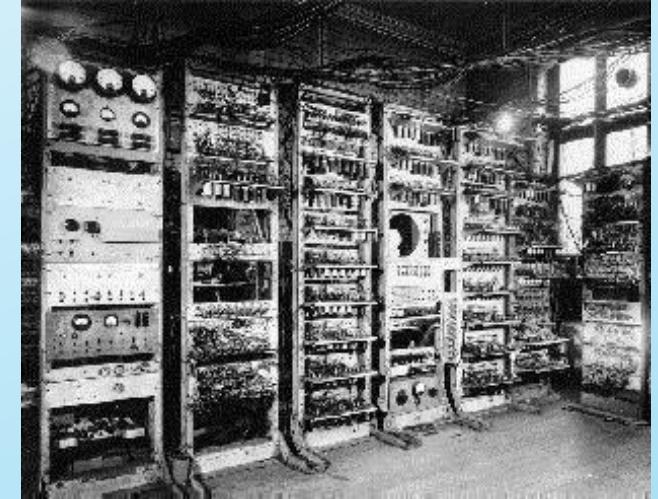
Von Neuman Architecture (single-memory stored program architecture)



- Stored Program Computer
- Programmable Instruction Set
- Architecture
- Memory bandwidth

Manchester Mark I

- 1948
- Implementeaza complet o masina Turing
- Pentru prima data se stocheaza programul
- 25 kW
- Cuvinte de 40 biti
- Timpul de executie a unei instructiuni standard: 1.8 ms



1950'

- În lume se utilizează 100 de calculatoare electronice
- Se dezvoltă limbajele FORTRAN și LISP
- Se dezvoltă memoriile magnetice
- Se realizează imprimantele matriciale

1951: UNIVAC I

- UNIVersal Automatic Computer I (UNIVAC I),
- Eckert and Mauchly
- first commercially successful computer

Generatia a doua de calculatoare

- **1959-1964**
- Cu tranzistori si circuite imprimate
- I/O banda perforata
- Se folosesc limbaje de programare de nivel inalt: limbaje nespecifice masinii
- Ex.: IBM 360

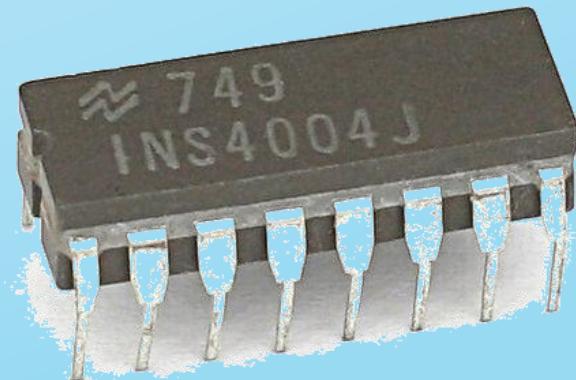
A treia generatie de calculatoare

- **1964-1975**
- Bazate pe circuite integrate
- Timesharing (partajarea timpului de lucru al CPU)
 - Mai multe sarcini rezolvate in paralel
 - Mai multi utilizatori la terminale diferite
 - Comunicare intre calculatoare (retea)
- Ex: PDP-8

- În aceeași perioadă:
 - Se inventează mouse-ul
 - Se dezvoltă limbajul BASIC
 - Se realizează prima rețea de calculatoare
 - Se dezvoltă sistemul de operare UNIX
 - Se realizează primul microprocesor

A patra generatie (1975- prezent)

- Se bazeaza pe circuite integrate la scara larga (LSI; VLSI; ULSI)
 - CI/VLSI: 5000 tranzistori
- Microprocesoare
 - Un CPU intr-un CI
- Primul microprocesor **INTEL 4004** (pe un singur cip era integrata intraga unitate de control si intreaga unitate aritmetica a unui calculator)
 - Cateva caracteristici
 - Cuvinte de 4 biti
 - Instructiuni de 8 biti
 - Adrese de 12 biti
 - Set 46 de instructiuni
 - 16 registrii de cate 4 biti
 - Ceasul intern 740 kHz
 - Durata ciclului de instructiune 10,8 μ s
 - 8 cicluri de ceas pe ciclu de instructiune
 - 2300 tranzistori
 - Tehnologie: 10 μ m/pMOS



- Se dezvolta arhitecturi de tip PC (personal computer)
 - Permite productia unui numar foarte mare de calculatoare, avand drept consecinta scaderea dramatica a pretului unui calculator
- Se dezvolta interfetele grafice
- Apare CD-ROM
- HD 3.5" disks
- Apare WWW

Generatia 4,5

- Sisteme multimedia
- Sisteme multiprocesor
- Sisteme cluster, grid, cloud etc.
- Inteligenta artificiala
- Calculatoare portabile/ultraportabile

Generatia 5



Quantum & Molecular Computers

■ Moores Law (1965)

- La fiecare 2 ani numarul de tranzistori pe cip se dubleaza
 - Gordon Moore: co-fondator INTEL (Intel Corporation)

■ Metcalfe Law (1973)

- Valoarea unei retele este proportionala cu patratul numarului de calculatoare din retea.
 - De exemplu numarul de conexiuni posibile in retea.
 - Robert Metcalfe: co-inventatorul eternetului

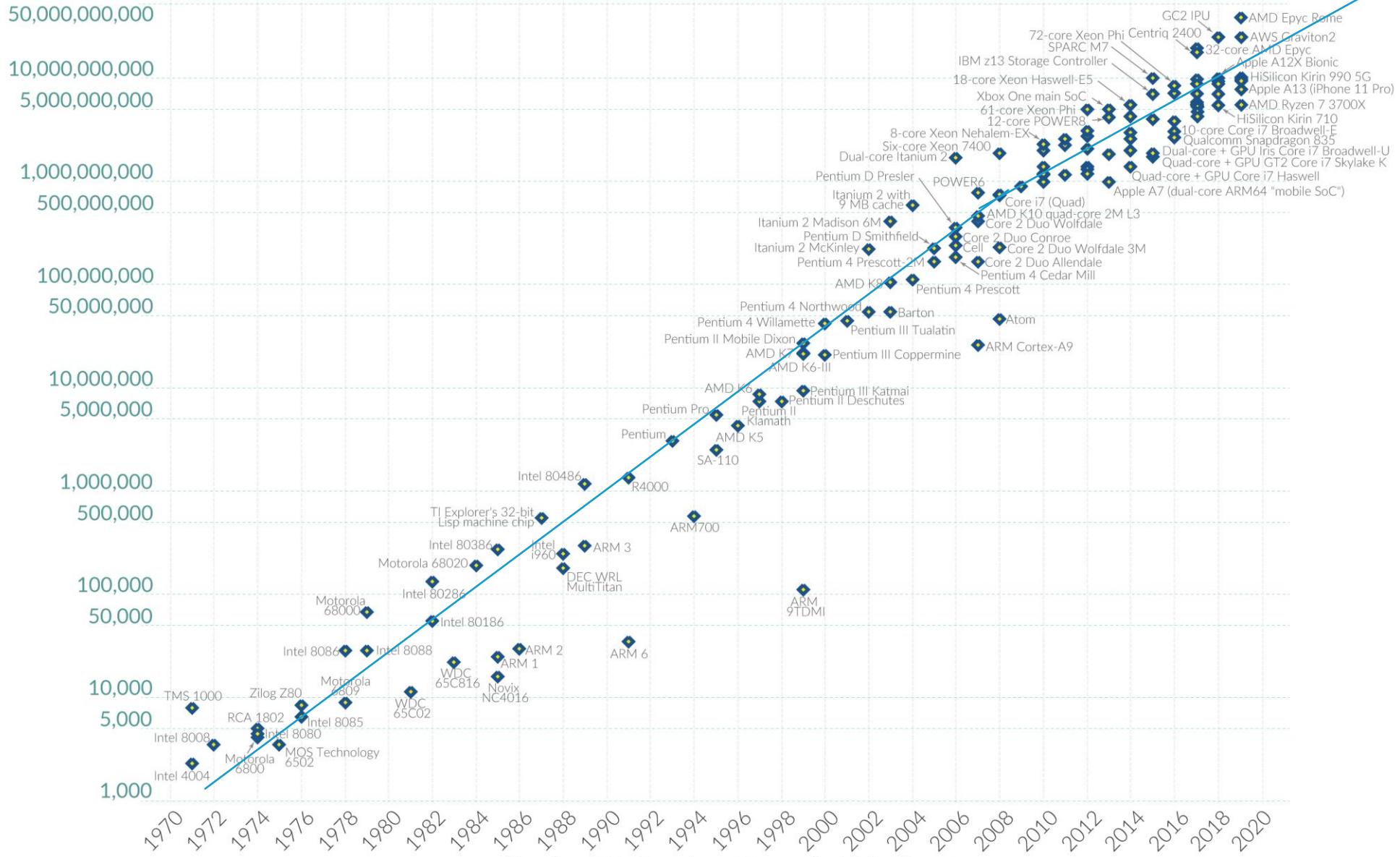
Moore's Law: The number of transistors on microchips doubles every two years

Our World
in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.

This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

Transistor count



Istoria calculatoarelor în România

- Prof. Horia Hulubei
(Rector al Universitatii din Bucuresti 1941-1944)



Infiinteaza succesiv:

1949: Institutul de Fizica al Academiei

1956: Institutul de Fizica Atomica

AGERPRES

PRIMUL CALCULATOR ROMANESC

CIFA 1

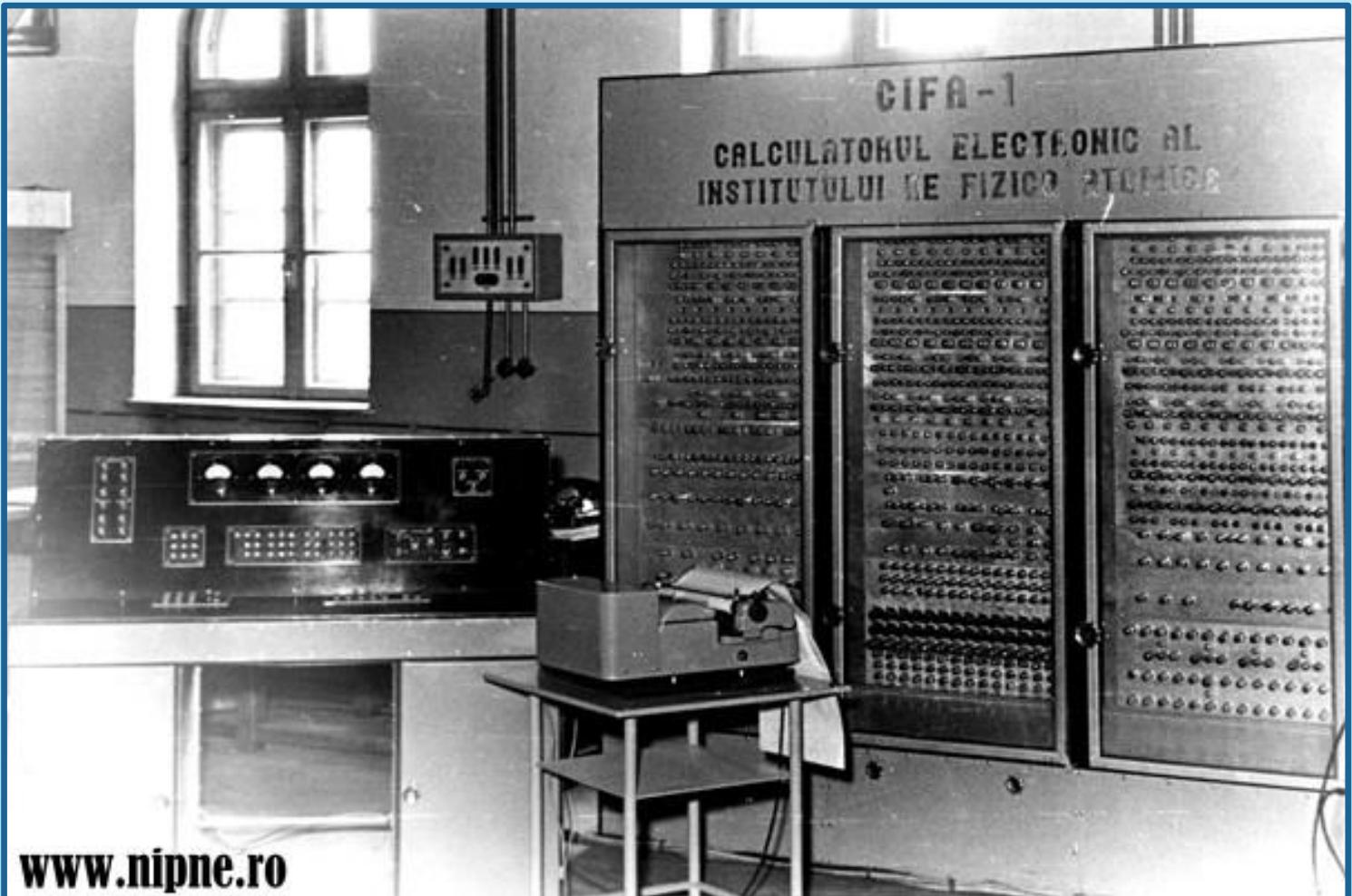
1954-1957

Victor TOMA

IFA - Magurele

Punere in functiune: aprilie 1957

A avut utilizare mai mult pentru demonstratii
A ajutat la formarea primilor specialisti romani in programare.



România a fost cea de a 8-a țară din lume care a reușit proiectarea și realizarea unui calculator electronic.



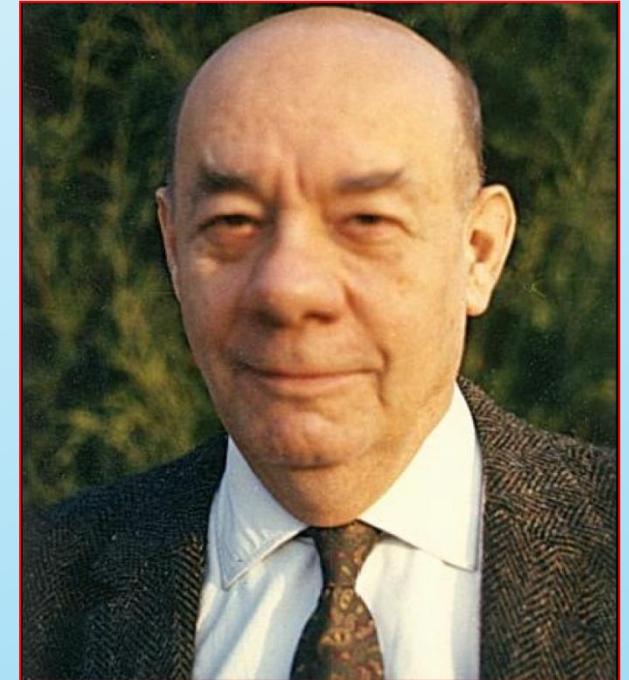
Victor Toma

- În 1959 s-a înființat **secția „Mașini de Calcul”**
la Facultatea de Matematică și Fizică din București,
din inițiativa academicianului Grigore Constantin Moisil,
- În 1962 s-a creat Centrul de Calcul al Universității din București.

Sala principală: sub amfiteatrul Spiru Haret

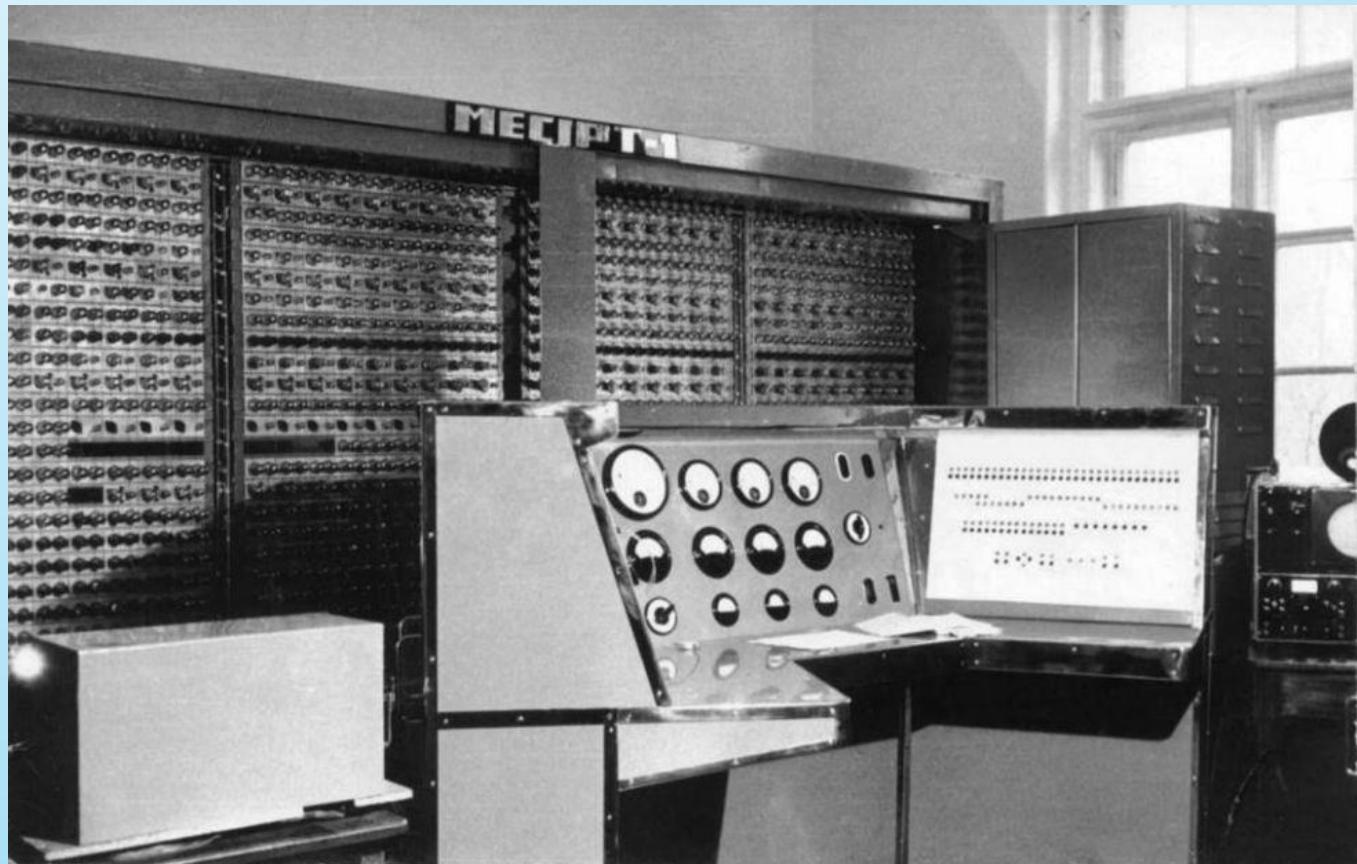
PROF. GRIGORE MOISIL (1906-1973)

Coordonatorul colectivului de specialisiti in
programare din IFA
(1956-1973)



PRIMUL CALCULATOR DE GENERATIA A DOUA

1961:
MECIPT-1
Timisoara



Utilizari semnificatie:

Reproiectarea cupolei pavilionului expozițional de la Piața Scânteii din București, în iulie 1963, (care se prăbușise în anul precedent).

Proiectarea barajul hidrocentralei de la Lacul Vidraru (jud. Argeș).

- În ianuarie **1968**, reprezentanții filialei de la Viena a companiei americane IBM au sosit la București cu un calculator **IBM 360/30**.
- Acesta a fost prezentat timp de câteva luni la o expoziție organizată la **Centrul de Calcul al Universității București** și, la sfârșitul acelui an, a fost cumpărat de autoritățile române, împreună cu alte două calculatoare identice – care au ajuns la Ministerul Agriculturii și la Comisia Națională de Informatică.

1967: A fost înființat
Institutul de Tehnica de Calcul

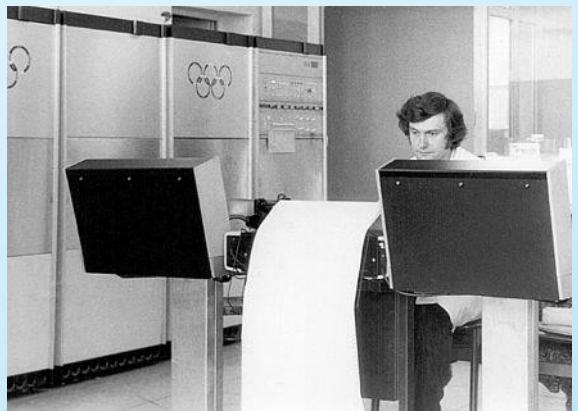
1968: Înființarea Fabricii de Calculatoare
Import de tehnologie din Franța (generația 2,5)

1973: Primul calculator de serie fabricat în România
FELIX C-256

Nicolae Ceaușescu a hotărât în același an achiziționarea licenței de fabricație a calculatorului francez IRIS 50 (renumit FELIX C256, în România).

Aceasta a fost o decizie politică și, din punct de vedere tehnic, s-a dovedit costisitoare, deoarece modelul respectiv nu a putut fi dezvoltat în același ritm, în România și în Franța, cu calculatoarele concepute și fabricate de compania americană IBM.

FELIX C-256



1968: Întreprinderea pentru Întreținerea și Repararea Utilajelor de Calcul

1975: Fabrica de Echipamente Periferice

1971: Fabrica de Memorii Timișoara

1980: CUB-Z (Calculator Universal de Birou), folosea microprocesorul Z80
a fost accesibil pentru utilizare largă în perioada 1987-1989,



PRIMUL PC ROMANESC

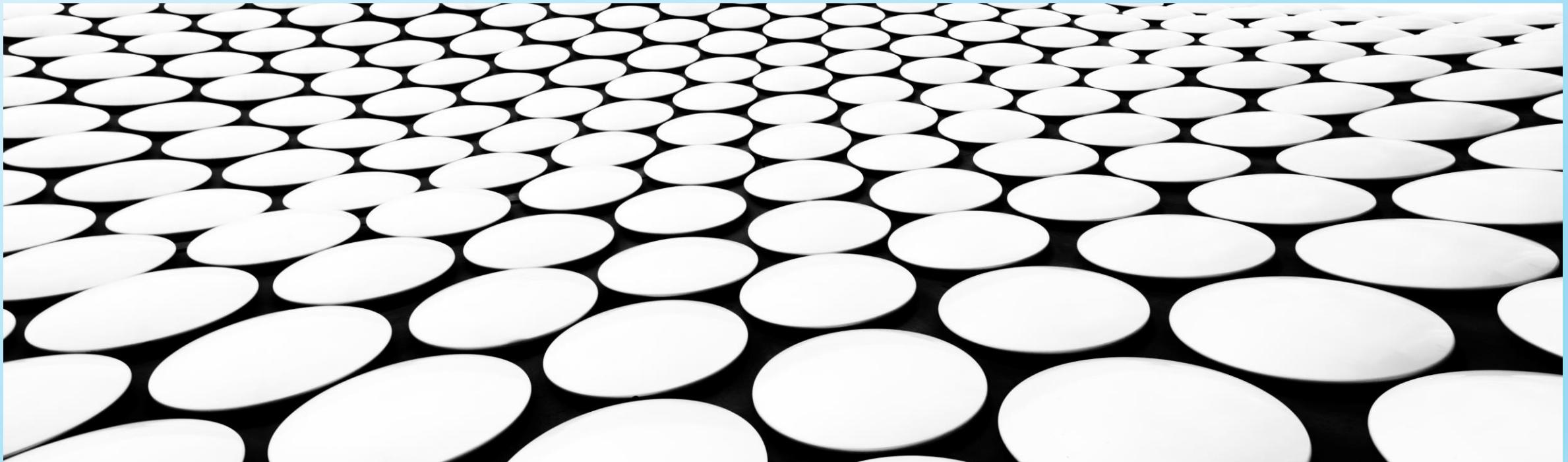


FELIX-PC

1985

ARHITECTURA SISTEMELOR DE CALCUL

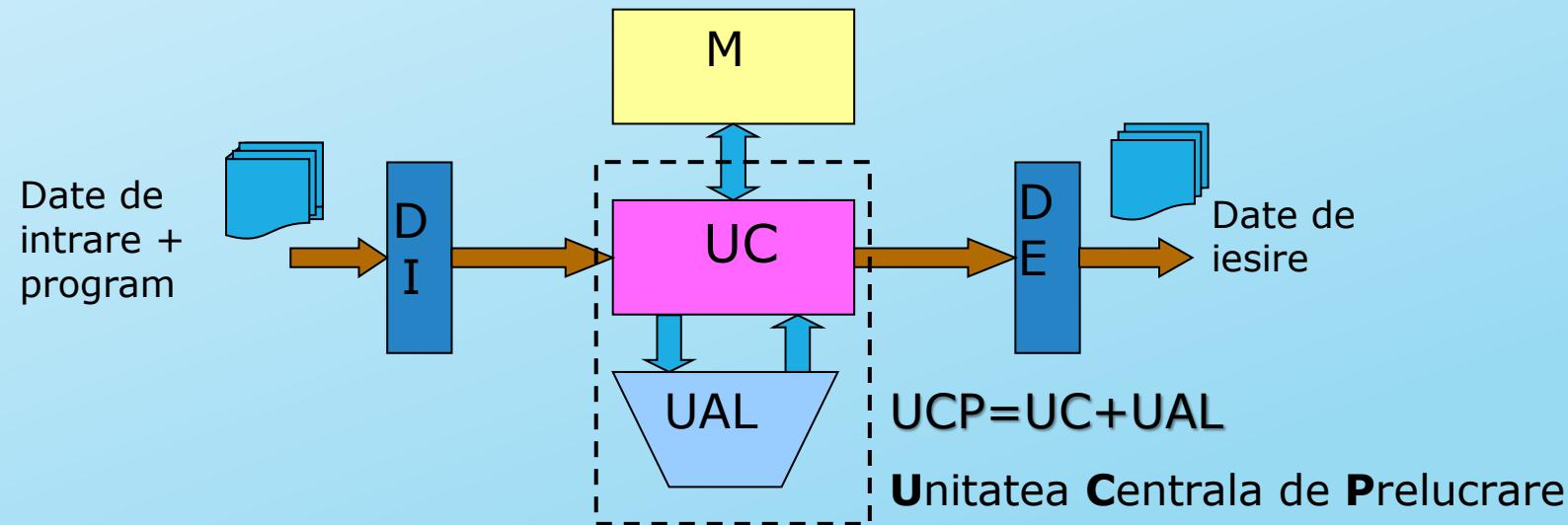
UB, FMI, CTI, ANUL III, 2022-2023



Funcționarea unui sistem de calcul

Modelul von Neumann

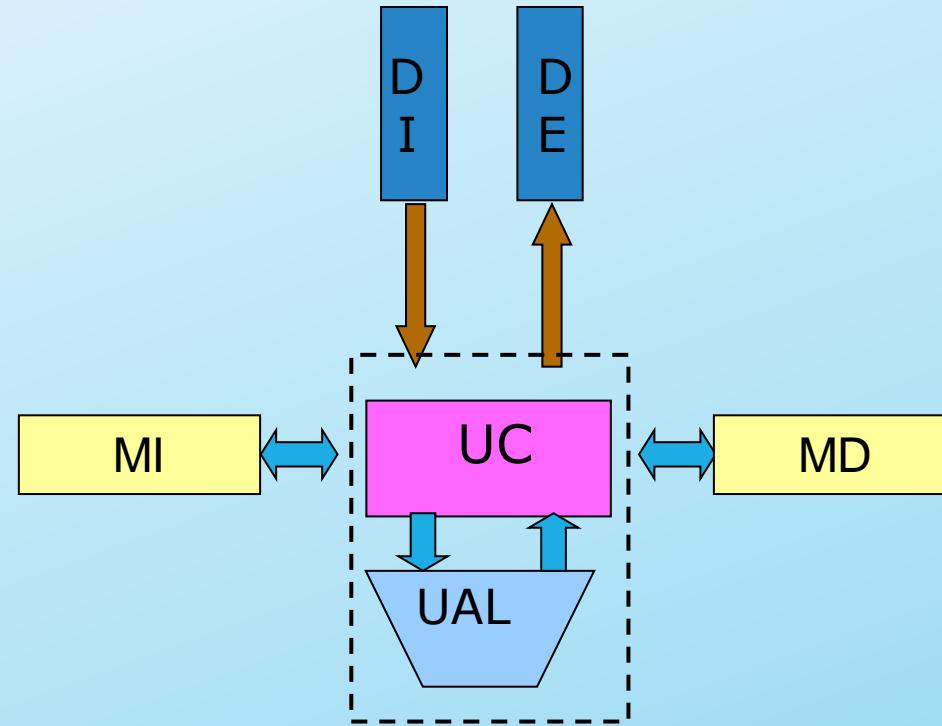
- + Unitatea de control (UC)
- + Unitatea aritmetico-logica (UAL)
- + Memoria (M)
- + Dispozitiv(e) de intrare (DI)
- + Dispozitiv(e) de iesire (DE)



In engleza CPU (Central Processing Unit)



Modelul Harvard



Memorie pentru instructiuni (MI)

Memorie pentru date (MD)



Componentele unui sistem de calcul

Unitatea centrală de prelucrare(UCP) [CPU]

- Unitatea de Control (UC) [CU]
 - controlează toate componente, executând instrucțiunile unui program
- Unitatea de calcul Aritmetic și Logic (UAL) [ALU]
 - efectuează calculele aritmetice și logice

Memoria (UM)

- stocheaza programele în curs de execuție și datele asociate lor

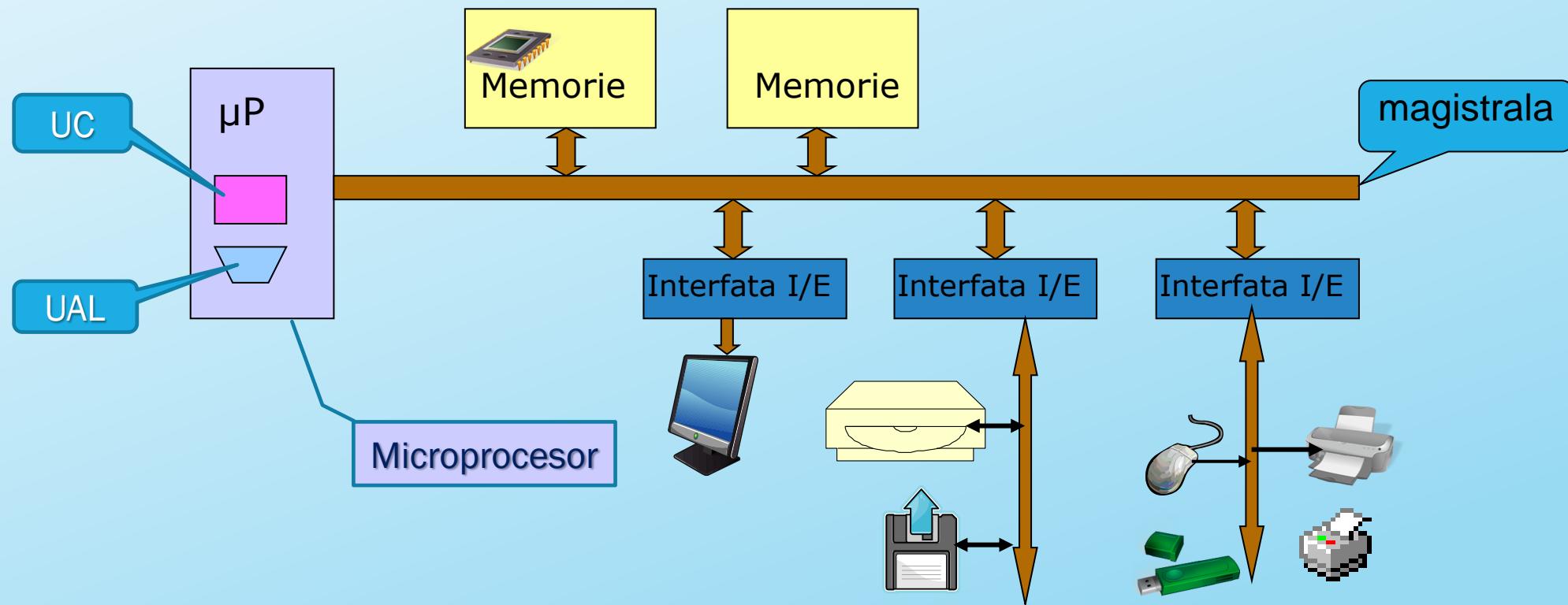
Unitatea de Intrare/Ieșire (U I/E)

Leagă sistemul cu lumea externă prin intermediul unităților periferice:

- ecran,
- tastatură,
- discuri,
- benzi magnetice,
- rețele etc.

Componentele unui sistem de calcul

Sistem de calcul bazat pe un microprocesor



Principiul de funcționare al unui calculator

- În UM există programe, fiecare program având un număr de instrucțiuni, precum și date destinate prelucrării.
- Execuția unui program înseamnă execuția succesivă a instrucțiunilor din care este alcătuit.
- Fiecare instrucțiune este executată în mai multe cicluri (etape) succesive.

Ciclurile (etapele) de executie ale unei instructiuni

- 1. extragerea instrucțiunii**
- 2. identificarea operanzilor**
- 3. transferul operanzilor**
- 4. execuția propriu-zisă**
- 5. transferul rezultatului**

Ciclul ‘**extragere instrucțiune**’ (instruction fetch).

- UC lanseaza o citire a memoriei la adresa la care se află instrucțiunea. Instrucțiunea are un număr de biți, în funcție de arhitectura calculatorului, de obicei multiplu de 8.
- Instrucțiunea citită este transferata prin magistrală și depusă într-un registru al UC-ului.

registru = memorie

Ciclul de **identificare** a operanzilor.

- În această fază trebuie identificate adresele unde se găsesc operanții. Aceștia se pot găsi în două tipuri de locații:
 - în registrele generale ale UC-ului;
 - la o adresă de memorie.
- La sfârșitul acestui ciclu, în UC trebuie să existe adresele fizice ale operanților participanți la instrucțiune.

Ciclul de **transfer al operanzilor** în UC

- În acest ciclu se aduc operanții participanți la instrucțiune de la adresele determinate în ciclul anterior. Ei sunt aduși din registrele generale sau de la adresele de memorie în registrele funcționale.

Ciclul de execuție propriu-zisă

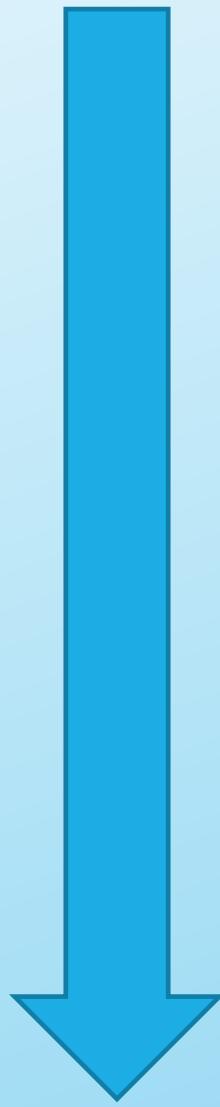
În acest ciclu are loc execuția propriu-zisă a instrucțiunii, dată de codul instrucțiunii

Orice instrucțiune are ca scop final aflarea unui rezultat, care poate fi:

- un **operand** în cazul instrucțiunilor aritmetice (de exemplu suma pentru cod de *adunare*, produsul pentru cod de *înmulțire*) sau:
- poziționarea unor **indicatori** în cazul instrucțiunilor logice
 - de exemplu, în cazul unei instrucțiuni de comparație între doi operanzi, poziționarea indicatorului $z=1$ corespunde la identitatea (egalitatea) celor doi operanzi.

Ciclul de transfer al rezultatului.

- La sfârșitul acestui ciclu, care înseamnă și sfârșitul executării instrucțiunii,
 - Se calculeaza adresa unde va fi transferat rezultatul
 - Se transfera rezultatul
 - Se calculează adresa de la care va fi adusă instrucțiunea următoare.





Sistemul de masini virtuale

Exemplul 1

Nivel n **Masina virtuala Mn cu limbajul masina Ln**

Programele exprimate in Ln pot fi interpretate de un interpretor ruland pe o masina inferioara sau pot fi traduse (compilate) intr-un limbaj inferior

Nivel 3 **Masina virtuala M3 cu limbajul masina L3**

Programele exprimate in L2 pot fi interpretate de un interpretor ruland pe M1 sau M0 sau pot fi traduse in L1 sau L0

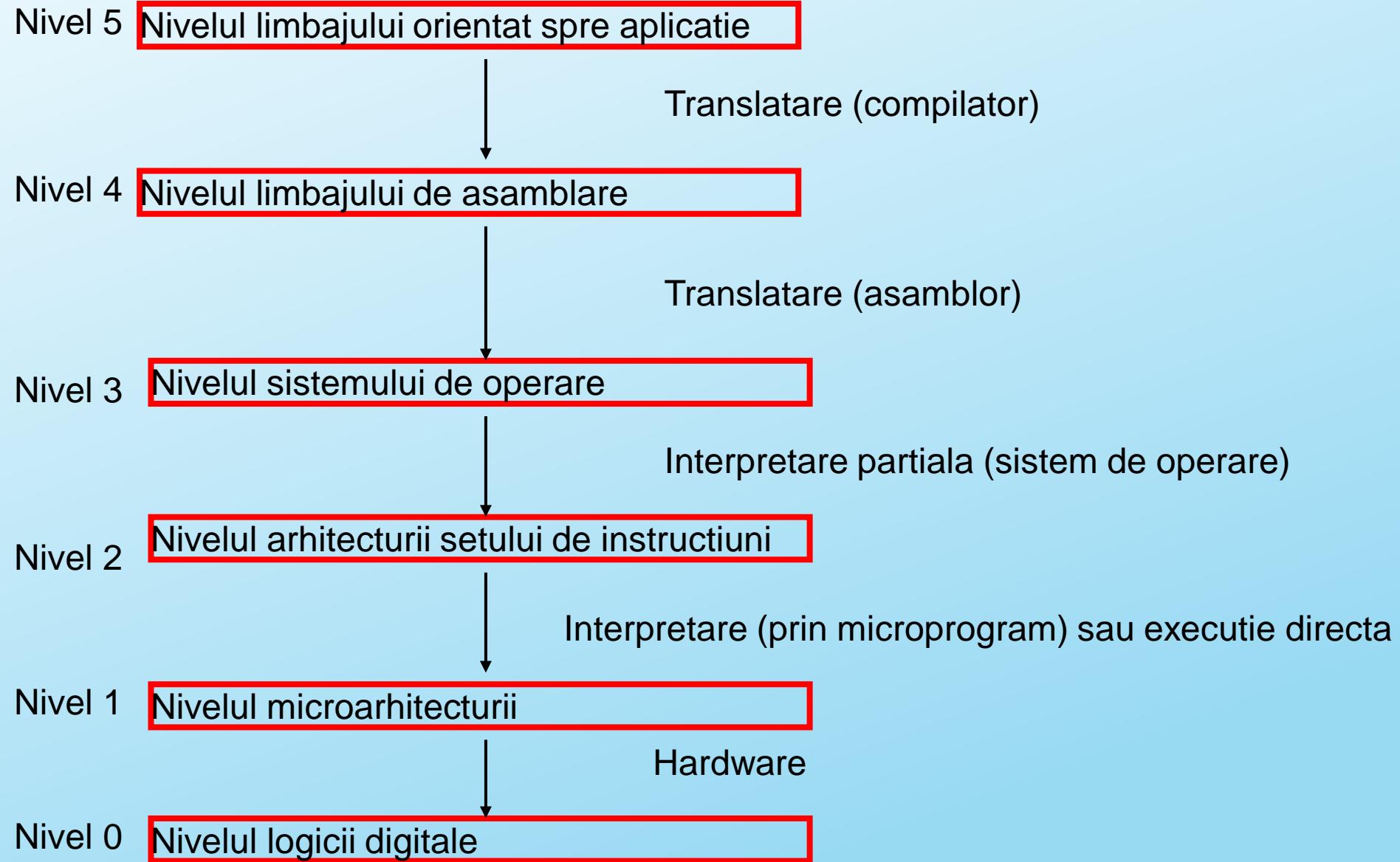
Nivel 2 **Masina virtuala M2 cu limbajul masina L2**

Programele exprimate in L1 pot fi interpretate de un interpretor ruland pe M0 sau pot fi traduse in L0

Nivel 1 **Masina virtuala M1 cu limbajul masina L1**

Programele exprimate in L0 pot fi executate direct de circuitele electronice

Nivel 0 **Masina reala M0 cu limbajul masina L0**



Nivelul definirii problemei (nivelul 7)

- Se identifica o problema complexa din viata reala si se formuleaza in asa fel incat sa poata fi rezolvata cu calculatorul (abstractizare/modelare)
- Se opereaza simplificari (se neglijeaza anumite detalii)
- Se folosesc modele matematice

Nivelul algoritmului (nivelul 6)

- Se defineste o procedura in pasi succesivi
- Pasii trebuie sa fie bine definiti pentru a putea fi executati fara ambiguitate de o masina (virtuala)
- Dezvoltarea algoritmului este un proces creativ!
- Este necesar un numar finit de pasi pentru rezolvarea problemei.

Nivelul limbajului de nivel inalt (nivelul 5)

- Ex: C/C++, Java, Python, FORTRAN, LISP, Ada, etc
- Utilizate de programatorii de aplicatie
- Compilatoarele efectueaza si alte sarcini in afara translatiei.

Nivelul limbajului de asamblare (nivelul 4)

- Cuprinde instructiuni mai primitive (elementare/simple)
- Foloseste o “versiune englezescă” a limbajului masina

Nivelul arhitecturii setului de instructiuni ISA (nivelul 3)

- Este interfata dintre hardware si software de nivel coborat
 - Avantaje: accepta diverse implementari pentru aceeasi arhitectura
 - Dezavantaje: in unele situatii frâneaza inovarea
- Exemple de arhitecturi:
 - IA-32(i386), PowerPC, MIPS, SPARC, ARM
- Este nivelul cel mai coborat ce poate fi “vazut” de programator.

Instructiunile

- Reprezinta limbajul masinii
- Sunt specifice platformei
- Se folosește un set limitat de comenzi in limbaj masina (ce pot fi intelese de hardware):
 - ex,: ADD, LOAD, STORE, RET

Nivelul microarhitecturii (nivelul 1)

- Interpretarea este realizata de Unitatea de Control
- Implementarea este realizata folosind resurse tehnice extrem de sofisticate:
 - In Pentium 4 este implementata ISA x86
 - In Motorola G4 este implementata ISA PowerPC
- Implementarea este optimizata in raport cu obiective de tip cost/performanta.
- Acest nivel este impenetrabil pentru programator

Nivelul Logic-Design (nivelul 0)

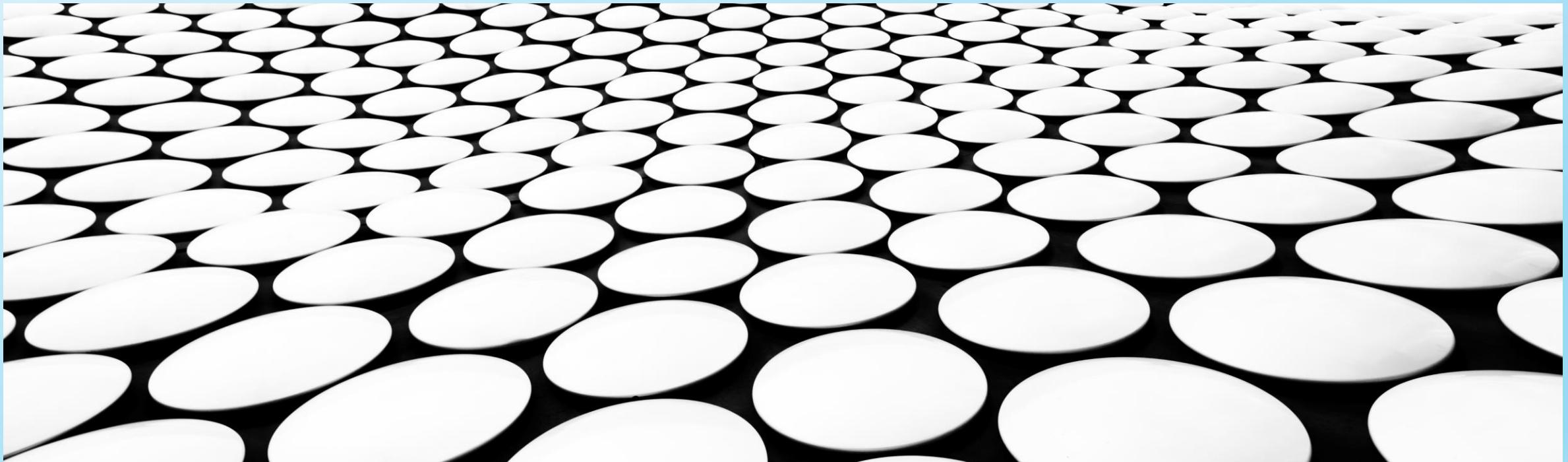
- ('nivelul electronicii digitale')
 - Porti logice
 - Multiplexoare, decodoare, PLA (programmable logic array)

Nivelul de dispozitiv

- Tranzistori, fire de legatura, etc
 - Implementeaza portile logice digitale
- La acest nivel masina se comporta mai de graba analogic dacat digital.
- Exemple de tehnologii:
 - Si CMOS, Si bipolar, GaAs

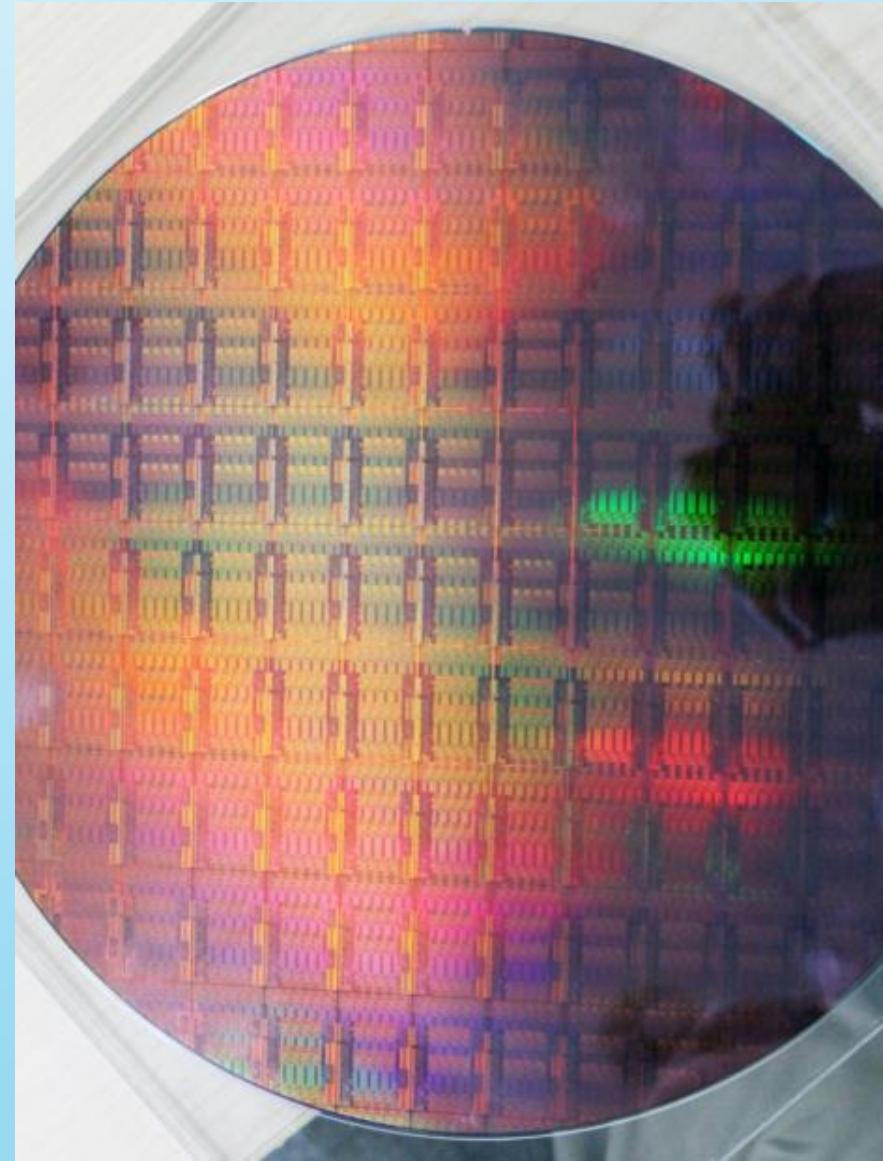
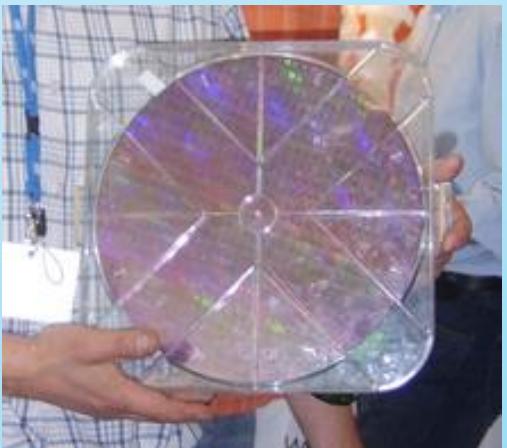
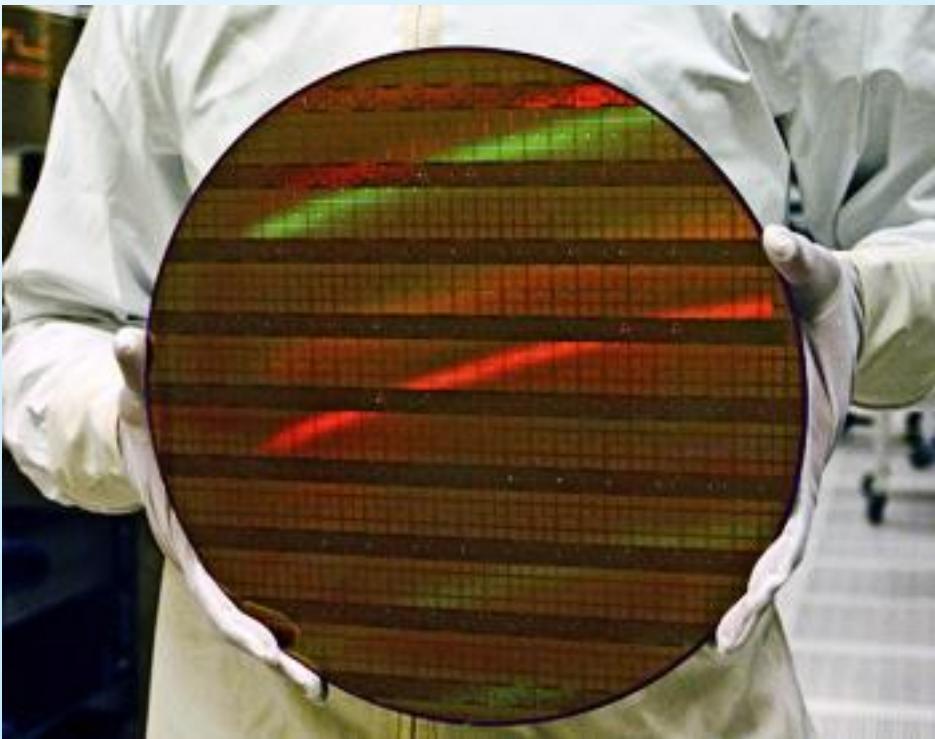
ARHITECTURA SISTEMELOR DE CALCUL

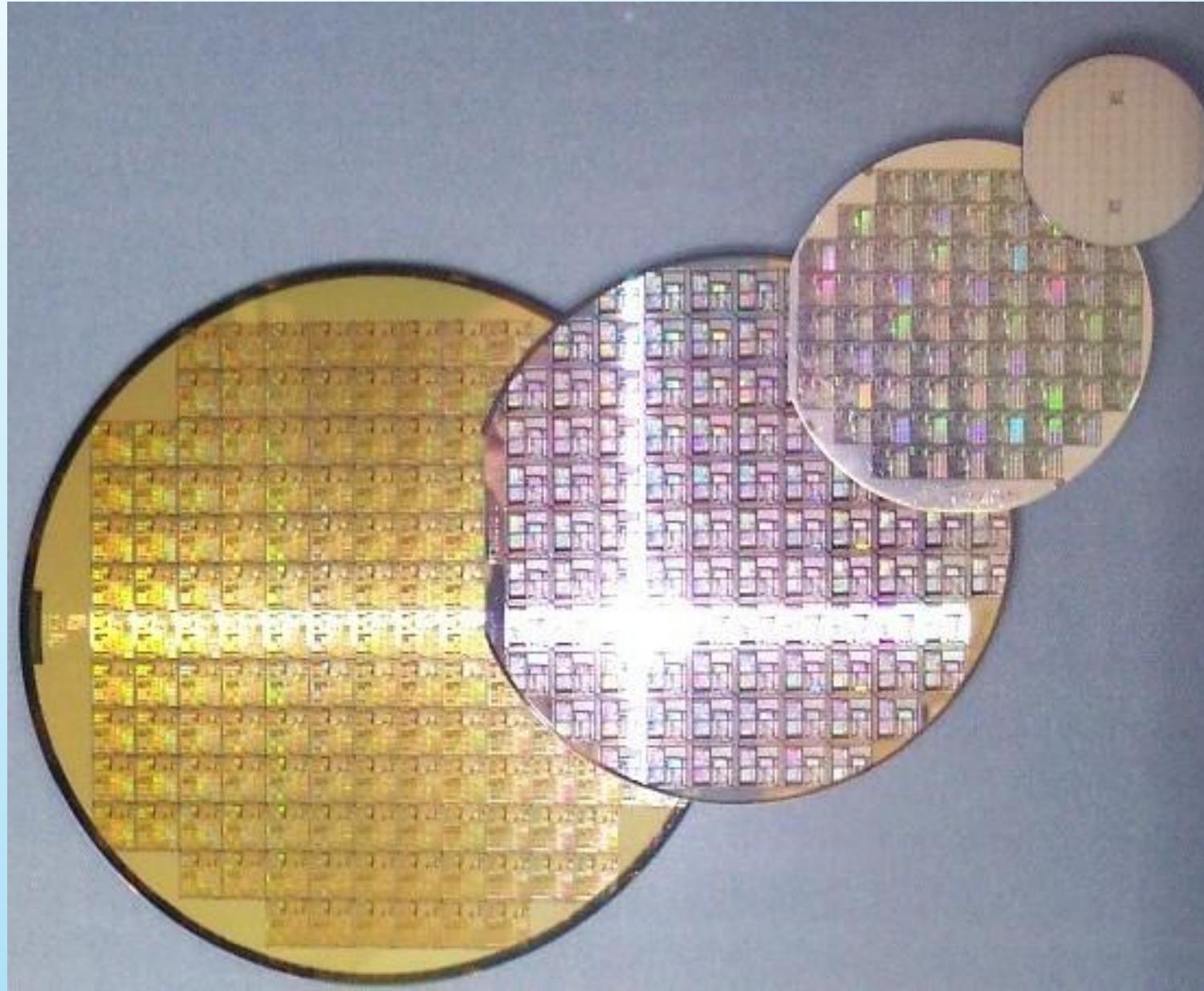
UB, FMI, CTI, ANUL III, 2022-2023

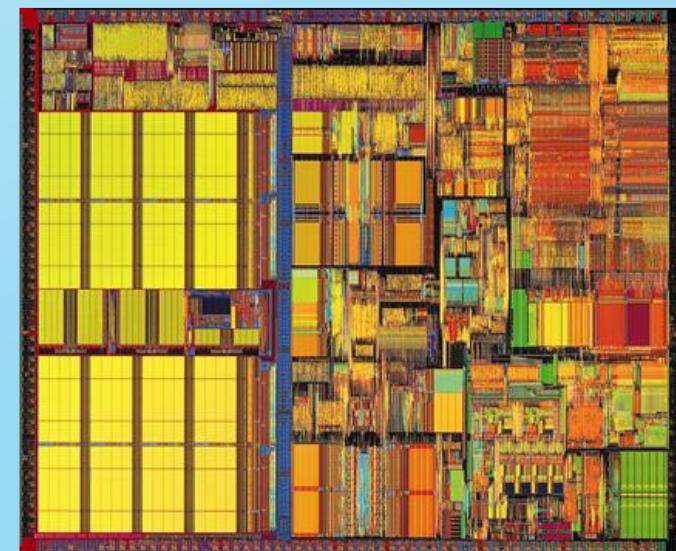
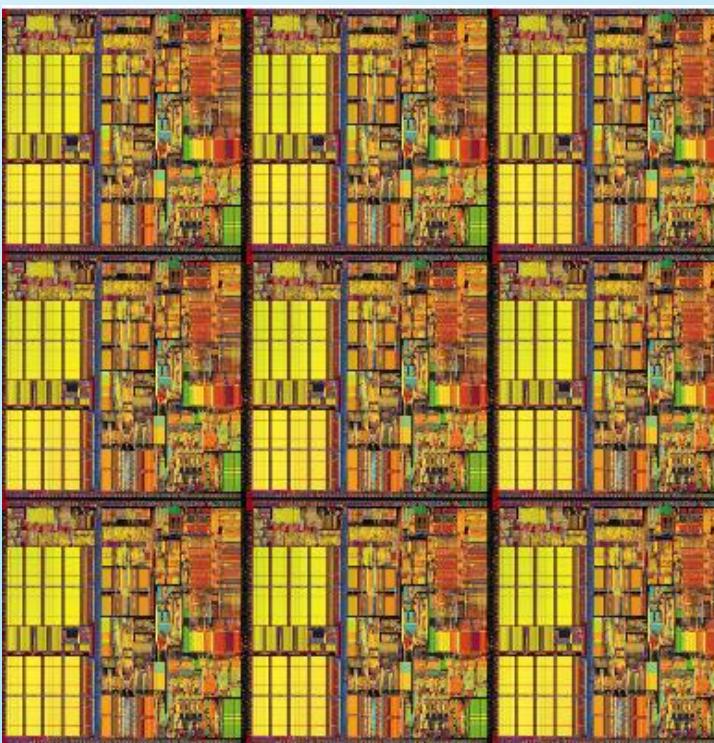
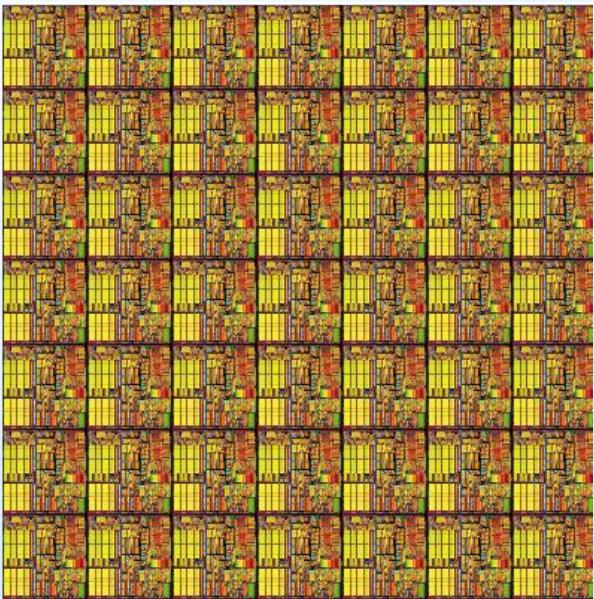


UNITATEA DE CONTROL

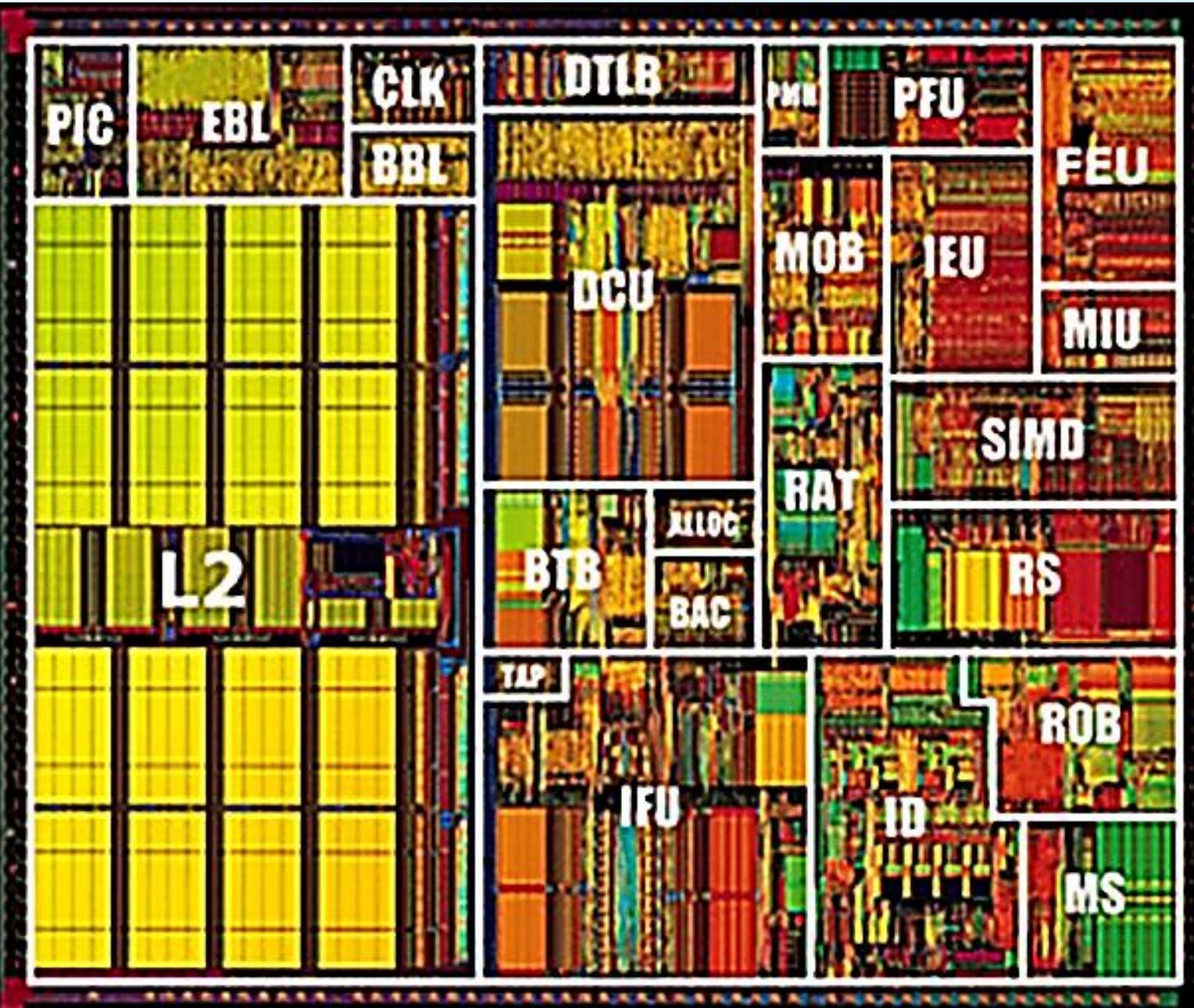
Cum se fabrica un microprocesor!







Procesor Pentium III



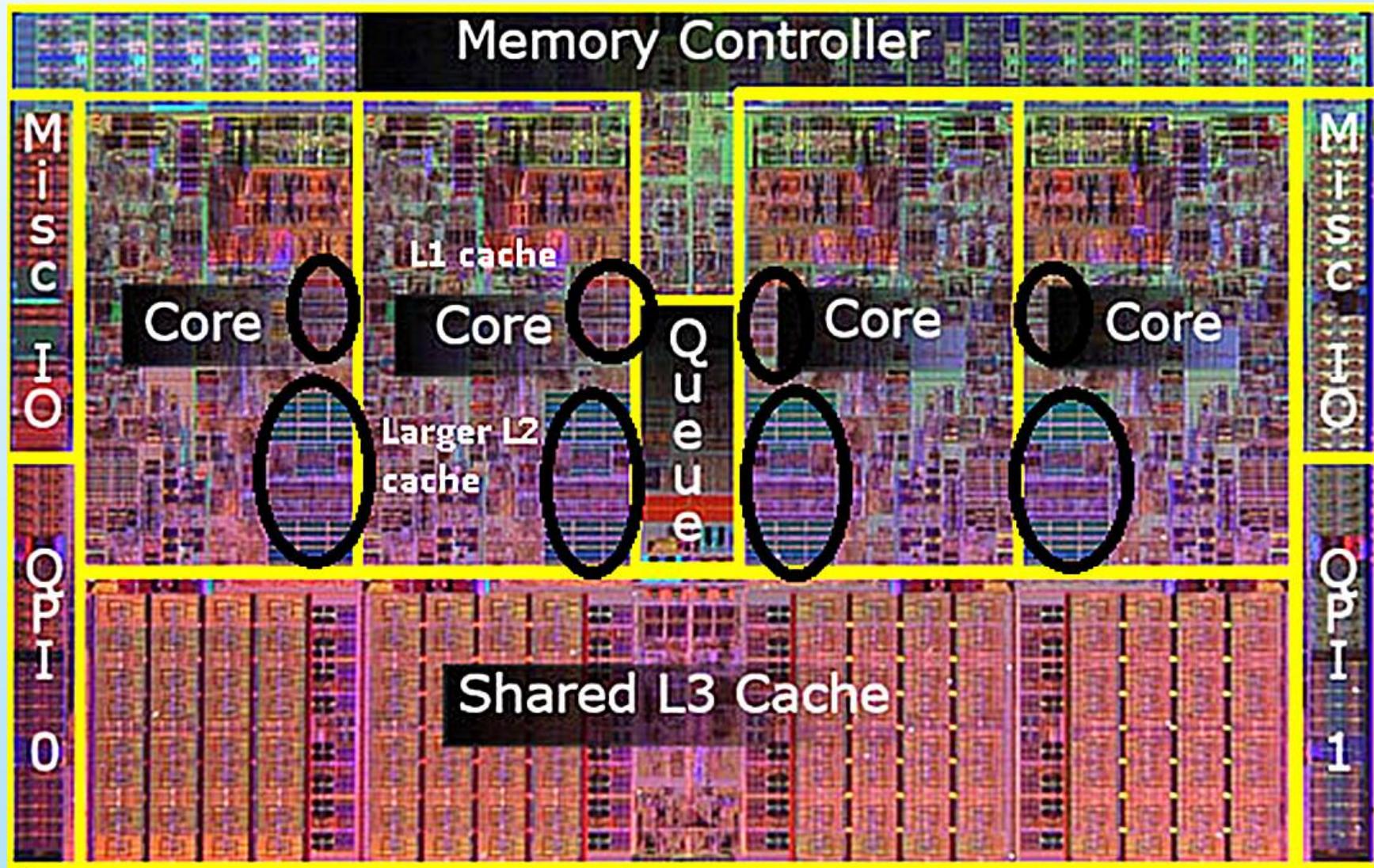
- 0.18 micron,
6-layer metal CMOS process technology
- 28.1 M transistors
- 3-way superscalar out-of-order execution
micro-architecture
- 256K Level 2 Cache
- 133 MHz IO bus

FEU - Floating point Execution Unit.

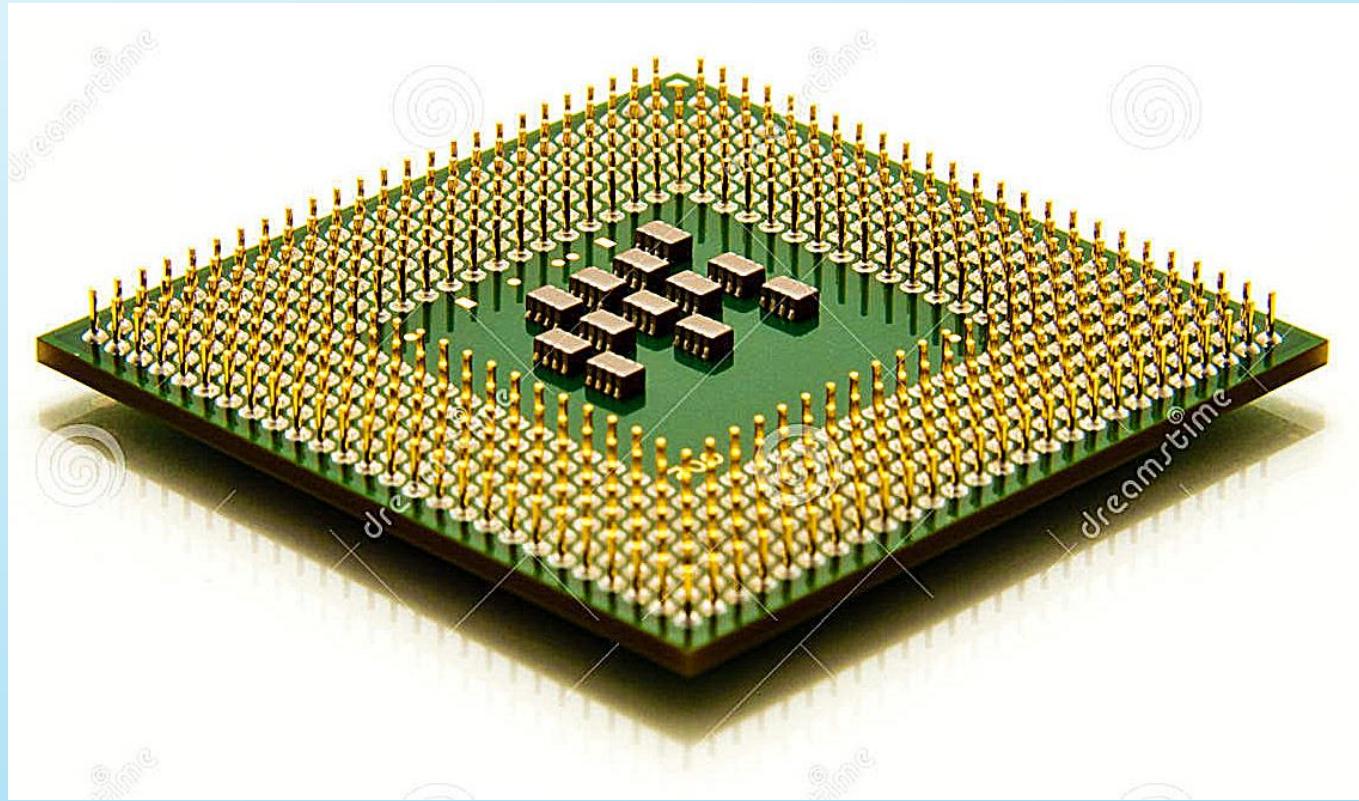
IEU - Integer Execution Unit.

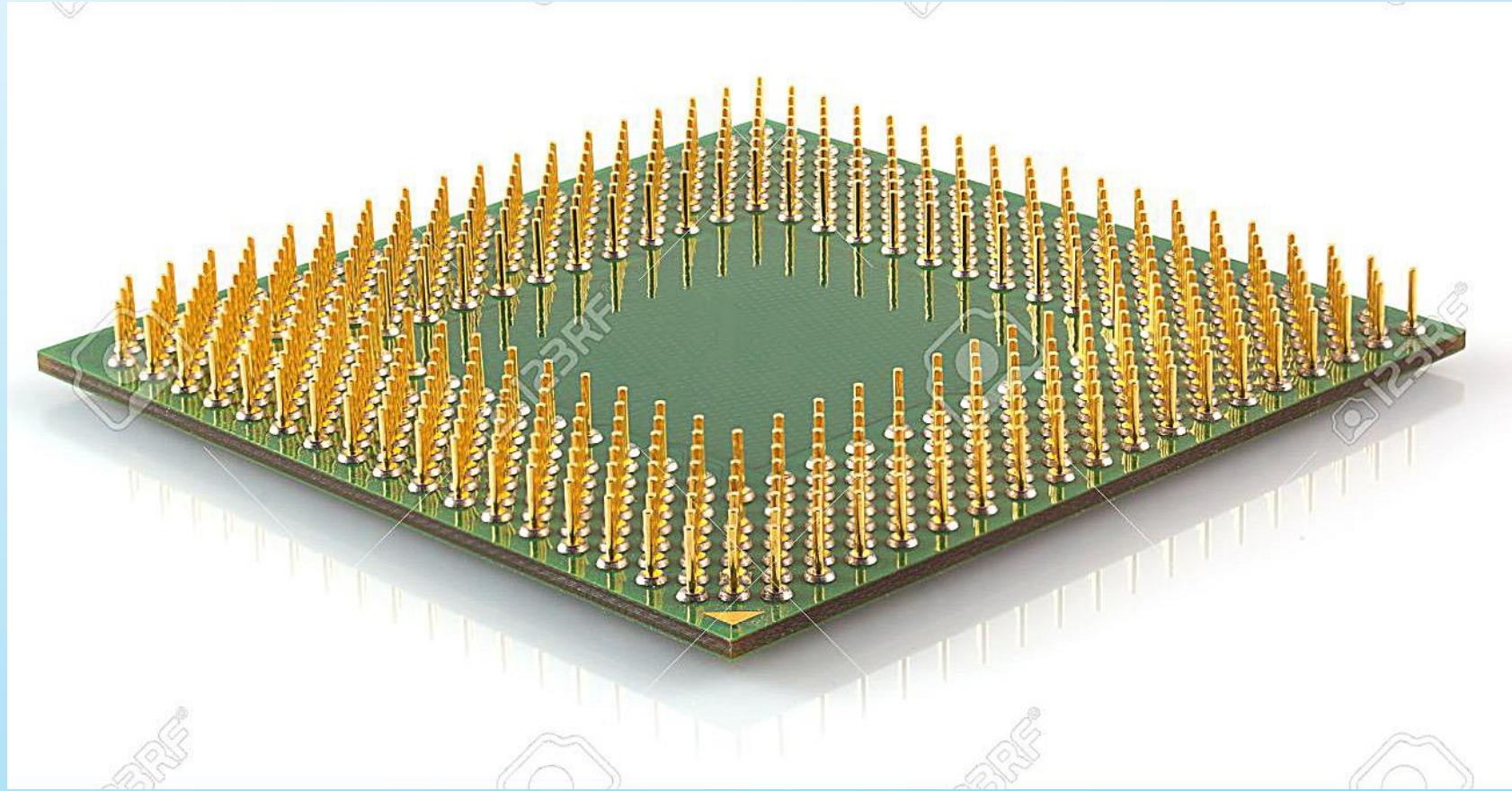
ID - Instruction Decoder.

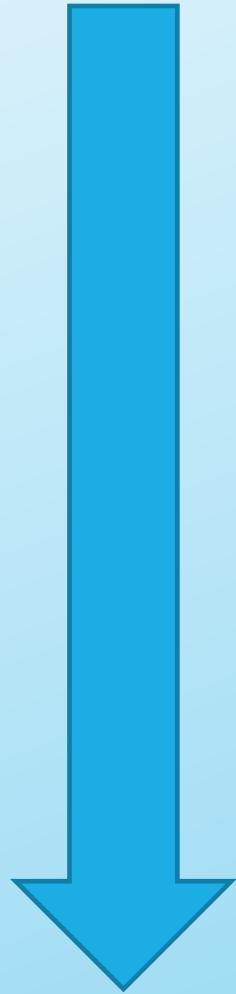
- **IFU - Instruction Fetch Unit.** Instruction fetch logic and a 16K Byte 4-way set-associative level one instruction cache resides in this block. Instruction data from the IFU is then forwarded to the ID.
- **ID - Instruction Decoder.** This unit is capable of decoding up to 3 instructions per cycle.
- **IEU - Integer Execution Unit.** This is responsible for ALU functionality of scalar integer instructions. Address calculations for memory referencing instructions are also performed here along with target address calculations for jump related instructions.
- **FEU - Floating point Execution Unit.** This performs floating point related calculations for both existing scalar instructions along with support for some of the SIMD-FP instructions.
- **DCU - Data Cache Unit.** Contains the non-blocking 16K Byte 4-way set-associative level one data cache along with associated fill and write back buffering.
- **PIC - Programmable Interrupt Controller.** Local interrupt controller logic for multi-processor interrupt distribution and boot-up communication.
- **CLK - Clocking.** Contains phase lock loop and other clocking control circuitry.
- **L2 - Level 2 Cache.** 256K unified level two cache.











UNITATEA DE CONTROL

Generatorul de Tact (GT)

Numaratorul de Program (NP)

Registrul de Instructiuni (RI)

Decodificatorul de Instructiuni (DI)

Generatorul de Faze (GF)

Registrul de Stare (RS)

Blocul Circuitelor de Comanda (BCC)



UNITATEA DE CONTROL

- Executa pe rand instructiunile din program
- Este formata din:
 - Generatorul de Tact (GT)
 - Numaratorul de Program (NP)
 - Registrul de Instructiuni (RI)
 - Decodificatorul de Instructiuni (DI)
 - Generatorul de Faze (GF)
 - Registrul de Stare (RS)
 - Blocul Circuitelor de Comanda (BCC)

Cadenta schimbarilor de stare

pentru toate circuitele secentiale din structura calculatorului
este data de catre **GENERATORUL DE TACT [GT]**

NP, RI, DI, GF

- NUMARATORUL DE PROGRAM [NP] pastreaza si ofera adresa, din memoria principală, a instructiunii in curs de executie.
- Codul instructiunii este citit din memoria principală si inscris in REGISTRUL DE INSTRUCTIUNI [RI].
- GENERATORUL DE FAZE [GF] construieste succesiunea de faze necesare pentru executia instructiunii din RI. Este, de regula, parte a decodificatorului de instructiuni [DI]

FAZE, RS

- Generarea fazei urmatoare este determinata de 3 elemente:
 - Tipul instructiunii
 - Faza curenta
 - Continutul registrului de stare.
- In **REGISTRUL DE STARE [RS]** sunt pastrate informatii despre modul de efectuare a operatiilor comandate de BCC, sau sunt pastrate informatii despre continutul registrilor generali

BCC

(Blocul Circuitelor de Comanda)

- Toate operatiunile elementare care se realizeaza pe parcursul executiei unei instructiuni sunt comandate de catre semnalele generate de **BCC** (microoperatiuni).
 - Operatiunile elementare se numesc **microoperatiuni**
 - Semnalele care comanda microoperatiunile se numesc: **microcomenzi**
 - **Microcomenzile nu sunt accesibile programatorului**

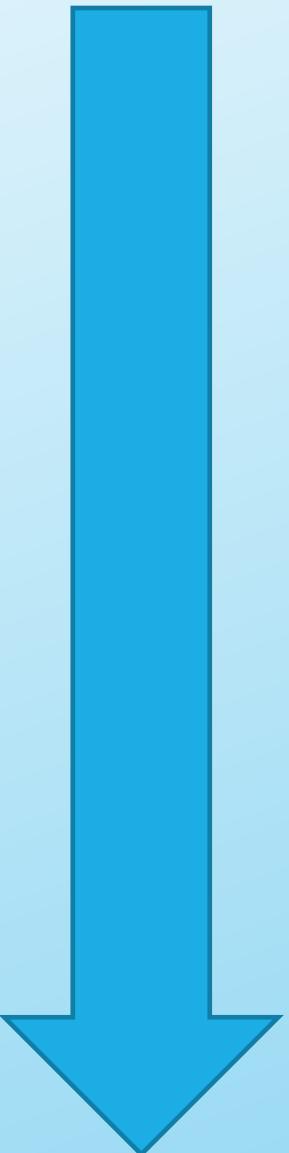
- Microcomenzile sunt **trimise** elementelor de executie din structura calculatorului:
 - registre, UAL, memorie, porturi, etccare **realizeaza** operatiuni elementare cum ar fi:
 - inscriere, validare iesire, stergere, deplasare stânga, etc.
- Executia unei instructiuni se desfasoara sub forma unei succesiuni de microoperatii.
- Toate microoperatiile care se executa in acelasi timp definesc o situatie in executia instructiunii numita **faza**.

Executia instructiunilor de catre UC (ciclul extragere-decodificare-executie)

Pentru fiecare instructiune UC executa urmatoarea secventa de pasi:

1. extrage din memorie de la adresa specificata in **NP** o instructiune si o depoziteaza in **RI**
2. determina tipul instructiunii din **RI**
3. determina adresele operanzilor daca instructiunea foloseste date din memorie
4. incarca datele, de la adresele specificate (daca exista) in **registri generali**
5. executa instructiunea
6. inscrie rezultatul executiei in locul indicat de instructiune
7. modifica numaratorul de program (**NP**) pentru a indica urmatoarea instructiune
8. sare la pasul 1.

- Multimea de instructiuni disponibile unui programator pentru un anumit nivel de masina (virtuala) se numeste **set de instructiuni** al acelei masini. Setul de instructiuni difera de la o masina la alta, de la un nivel la altul.
- Masinile cu un set redus de instructiuni sunt numite calculatoare **RISC** (*Reduced Instruction Set Computer*)
- Setul de instructiuni contine urmatoarele **tipuri** de instructiuni:
 - Aritmetice
 - Logice
 - Salt
 - Apel
 - Deplasare (registru-registru, memorie-registru, memorie-memorie)
 - Oprit



ISA (Instruction Set Architecture) Arhitectura Setului de Instructiuni

ISA contine toate operatiile si componentele calculatorului care pot fi “**vizibile**” programatorului.

Arhitectura include:

- Organizarea memoriei
 - Spatiul de adrese (cate locatii pot fi adresate)
 - Adresabilitate (cati biti pe locatie)
- Setul de registri
 - Cat de multi, ce dimensiuni, mod de utilizare
- Setul de instructiuni
 - Opcodes (**operation code**)
 - Tipuri de date
 - Moduri de adresare

Tipuri de ISA (Instruction Set Architecture)

- Arhitectura cu **stiva** (stack)
 - Fara operanzi expliciti
- Arhitectura cu **acumulator**
 - Un operand explicit
- Arhitecturi cu **registri** de uz general
 - 3 operanzi expliciti
 - 3 subtipuri
 - Memorie-memorie
 - Registru-memorie
 - Registru-registru

Arhitectura cu stiva

- 0 operanzi expliciti
- Toti operanzii sunt in stiva

$$A = B * (C + D * B)$$

1. Push B
2. Push D
3. *
4. Push C
5. +
6. Push B
7. *
8. Pop A

SP → 3

5	0.	2.	4.	8.
4		D	C	B
3	B		DB	A
2	V3	V3	V3	V3
1	V2	V2	V2	V2
0	V1	V1	V1	V1

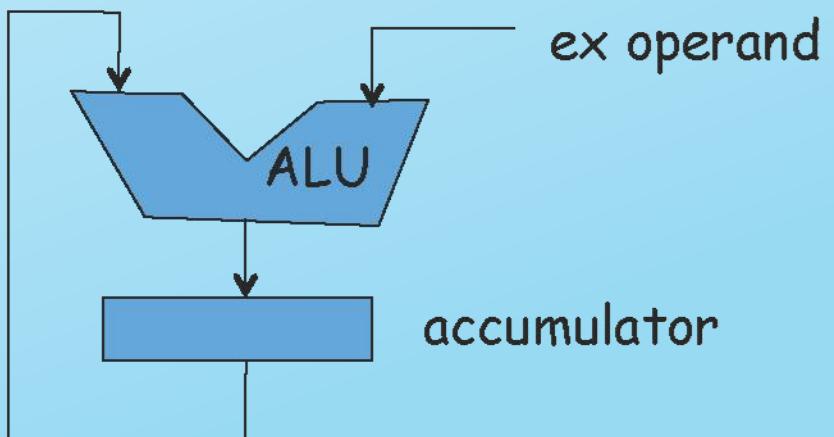
Arhitectura cu accumulator

- Un operand explicit
- Celelalte doi operanzi sunt in accumulator (o sursa si o destinatie)

O memorie de tip accumulator este o memorie aditiva: retine suma dintre valoarea nou introdusa si valoarea initiala.

$$A = B * (C + D * B)$$

1. Load B
2. Mult D
3. Add C
4. Mult B
5. Store A



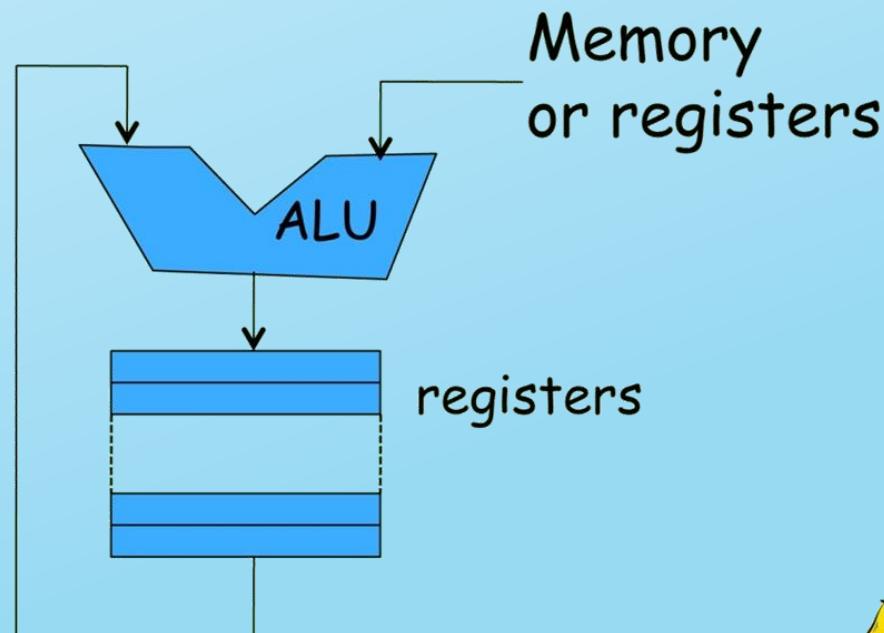
Arhitecturi cu registri de uz general

Arhitectura registru-memorie

- Doi operanzi: un registru (pentru o sursa si pentru destinatie) si o locatie de memorie

$$A = B * (C + D * B)$$

1. Load R0, B
2. Mult R0, D
3. Add R0, C
4. Mult R0, B
5. Store R0, A

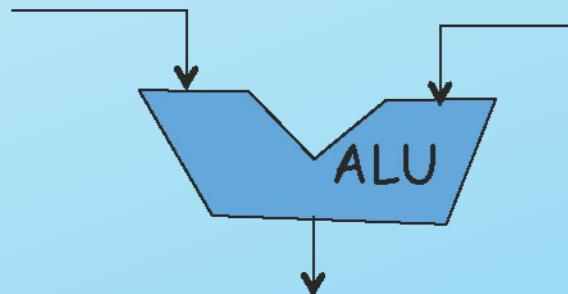


Arhitectura memorie-memorie

- Toti operanzii pot fi localizati: fie in memorie fie intr-un registru

$$A = B * (C + D * B)$$

1. Mult R0, D, B
2. Add R0, R0, C
3. Mult A, B, R0

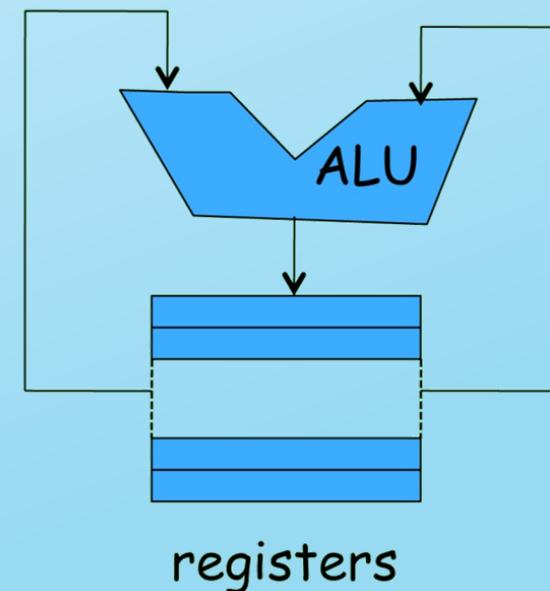


Arhitectura regiszru-regiszru (load/store)

- Cei trei operanzi trebuie sa fie in registri separati

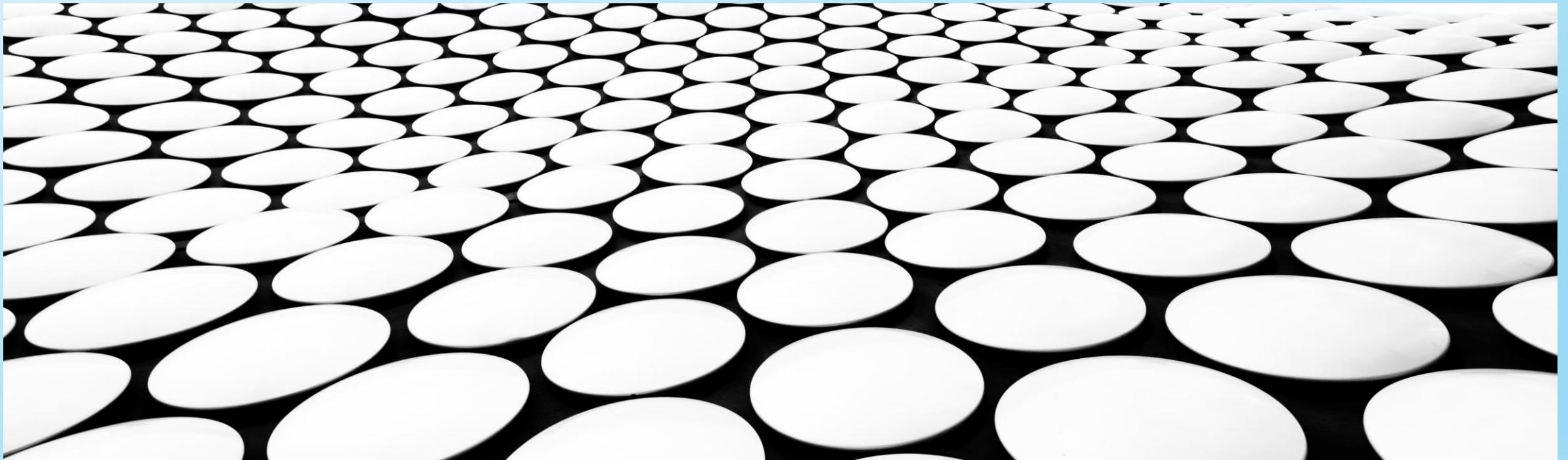
$$A = B * (C + D * B)$$

1. Load R0, B
2. Load R1, D
3. Mult R2, R0, R1
4. Load R3, C
5. Add R3, R3, R2
6. Mult R4, R3, R0
7. Store R4, A



ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



MEMORII CACHE

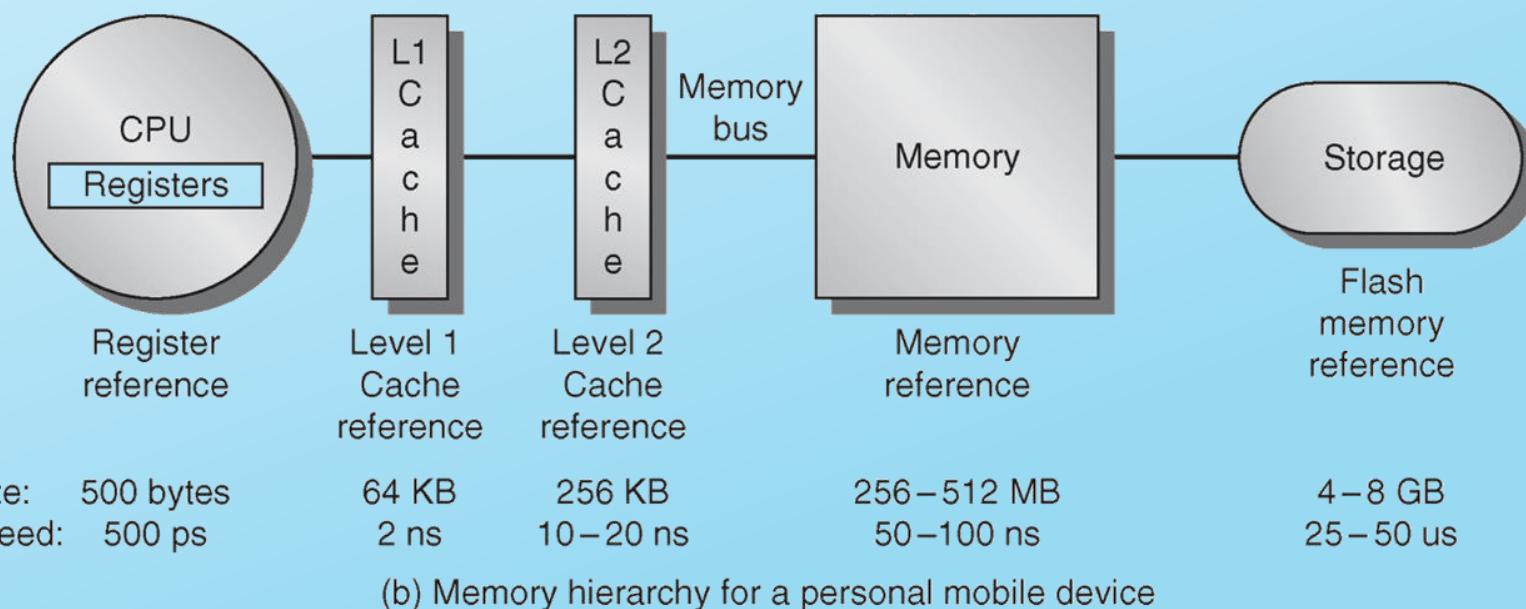
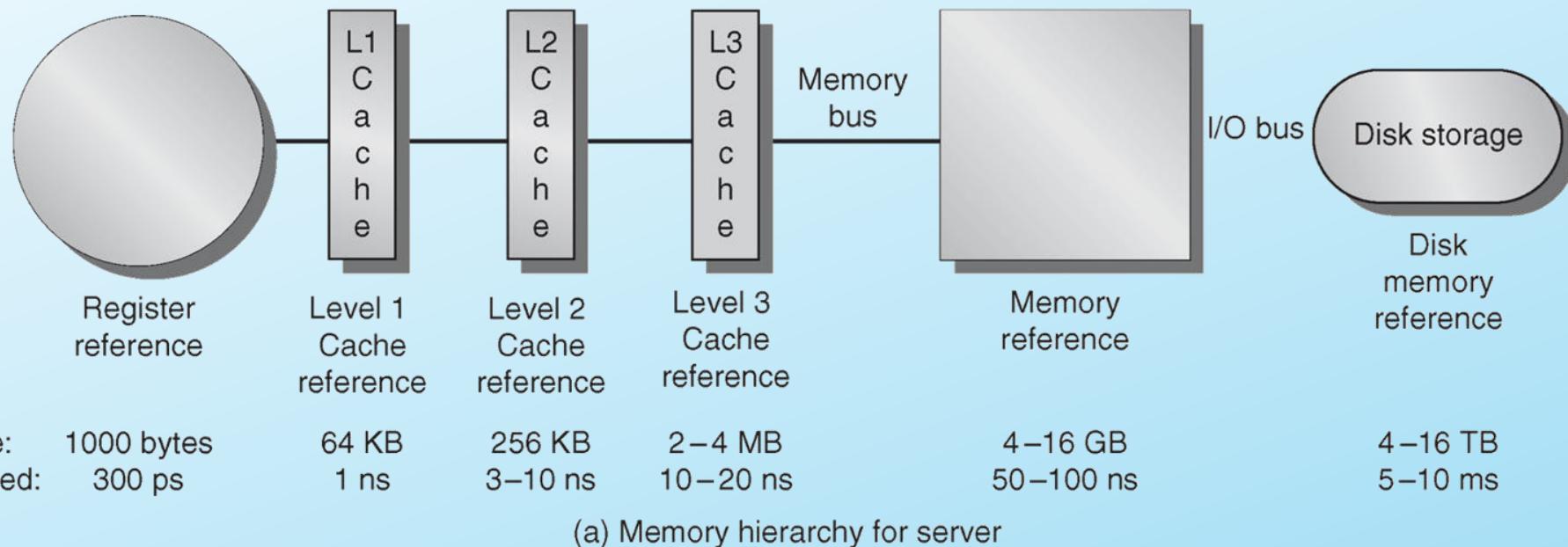
Ce este memoria cache?

Este o memorie, rapida, intermediara, temporara, care stocheaza informatiile mai frecvent utilizate.

Informatia necesara UC este cautata mai intai in memoria cache si apoi in memoria principala.

Daca exista mai multe memorii cache (ierarhizate) atunci informatia este cautata, mai intai in memoriile cache, in ordine ierarhica, si apoi daca nu este gasita, in memoria principala

Arhitecturi cu memorii cache



```
Mentest32 v4.3.7          : Intel Core2 Duo E7500 @ 2.93GHz
CPU_C1k : 2926 MHz         : Pass 61% #####
L1 Cache: 64K 39161 MB/s   : Test 20% #####
L2 Cache: 3072K 18239 MB/s  : Test #9 [Modulo 20, Random pattern]
L3 Cache: None             : Testing: 1024K - 2048M 2047M of 3037M
Memory : 3037M 2312 MB/s    : Pattern: c0b84181-5

CPU: 01                    : CPUs_Found: 2      CPU_Mask: ffffffff
State: W\                   : CPUs_Start: 2      CPUs_Active: 1

Time 0:14:49 Iterations: 2 AdrsMode:64Bit Pass: 0 Errors: 0

(ESC)exit (c)onfiguration (Space)scroll_lock (Enter)scroll_unlock
```

2007

Memtest86 v4.3.7

AMD Athlon II X3 440
CPU C1k : 3010 MHz | Pass 28% #####
L1 Cache: 128K 35264 MB/s | Test 35% #####
L2 Cache: 512K 15747 MB/s | Test #7 [Moving inversions, 32 bit pattern]
L3 Cache: None | Testing: 1024K - 2008M 2007M of 1964M
Memory : 1964M 4158 MB/s | Pattern: ffffff7f

CPU: 012 | CPUs_Found: 3 CPU_Mask: ffffffff
State: IWW | CPUs_Started: 3 CPUs_Active: 1

Time 0:19:20 Iterations: 3 AdrsMode:64Bit Pass: 1 Errors: 0

Pass complete, no errors, press Esc to exit

(ESC)exit (c)configuration (Space)scroll_lock (Enter)scroll_unlock

2011

Memtest86 v4.3.7

CPU Clk : 3601 MHz : Intel Core i7-9700K @ 3.60GHz
L1 Cache: 64K 294204 MB/s : Pass 7% ##
L2 Cache: 256K 125492 MB/s : Test 72% #####
L3 Cache: 12288 67177 MB/s : Test #4 [Moving inversions, 8 bit pattern]
Memory : 16G 21230 MB/s : Testing: 14G - 16G 2048M of 16G
Pattern: 40404040

CPU: 01234567 : CPUs_Found: 8 CPU_Mask: ffffffff
State: WWWWW/WWW : CPUs_Started: 8 CPUs_Active: 1

Time 0:01:35 Iterations: 1 AdrsMode:64Bit Pass: 0 Errors: 0

-

(ESC)exit (c)configuration (Space)scroll_lock (Enter)scroll_unlock

2019

Cateva definiții

- **cache hit:**

- eveniment de identificare a informației căutate în memoria cache
(succes)

- **cache miss:**

- eveniment cauzat de lipsa informației căutate în memoria cache
(insucces)

- **hit rate:**

- Procentul de accesări **cu** succes a memoriei cache, din totalul accesărilor

- **miss rate:**

- Procentul de accesări **fără** succes a memoriei cache, din totalul accesărilor

- Rata tipică de succes (hit rate): $\geq 95\%$

Design simplu de memorie cache

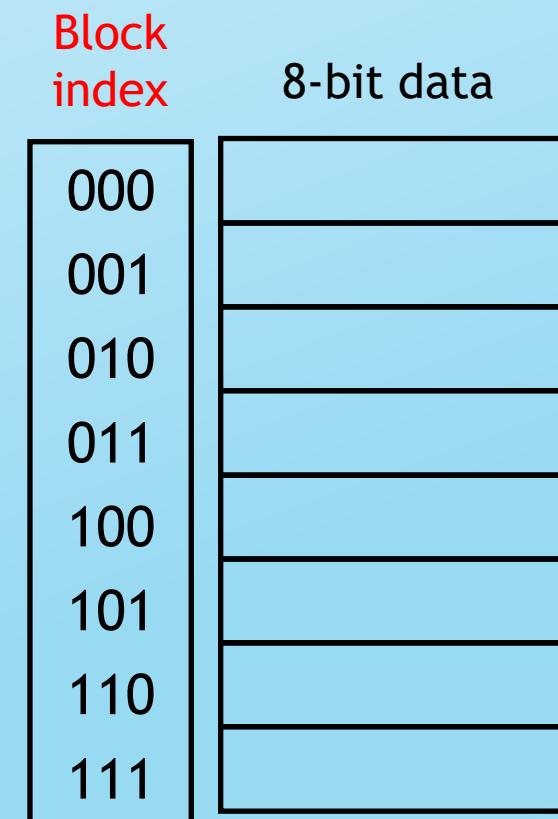
Memoria cache este divizata in blocuri, de dimensiuni egale.

- Numarul de blocuri in memoria cache este, uzuial, o putere a lui 2.

- Exemplu simplu

- fiecare bloc conține un byte.
- Index cu dimensiunea de 3 biti
- 8 blocuri

$$2^3=8$$



Fiecare memorie cache are atasat un **controler de memorie**.
Ce face controlerul de memorie?

4 intrebari importante:

1. Cand este copiat un bloc de date, din memoria principala in cache, **unde** este plasat el?

2. **Cum se stabileste** daca un cuvant (bloc) este deja in cache, sau trebuie adus mai intai din memoria principala?

3. Cache-ul se poate umple.
Pentru a incarca un nou bloc, trebuie sa inlocuim unul existent!
Care va fi acela?

4. Cum se desfasoara o operatie de scriere in memorie?

TIPURI DE MEMORIE CACHE

Memorie cache cu:

Mapare directă

Mapare complet asociativa

Mapare directă în seturi asociative

Memoria cache cu mapare directă

(cea mai simplă structură)

Fiecarei adrese din memoria principală îi corespunde o poziție preciză în memoria cache.

Exemplu:

Memorie principală de 16 byte

Memorie cache de 4 byte

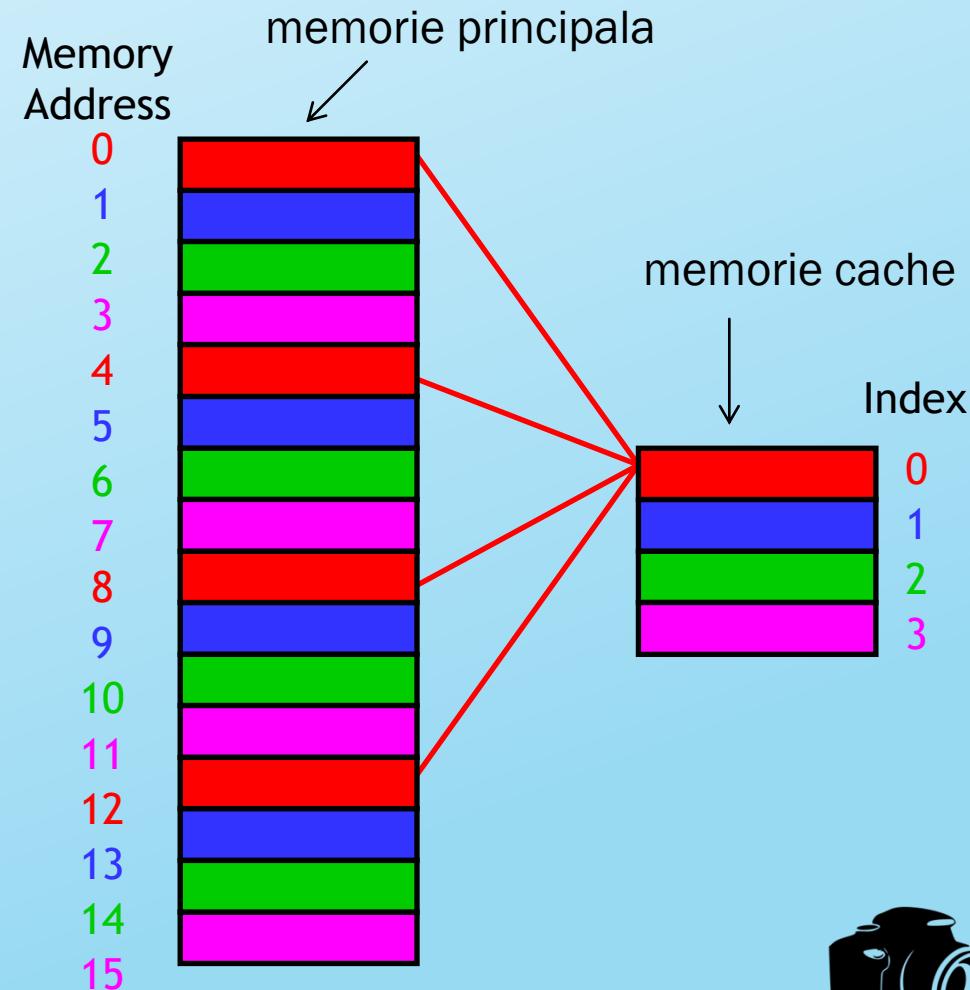
Locațiile de memorie **0, 4, 8 și 12**

vor fi mapate în blocul **0**

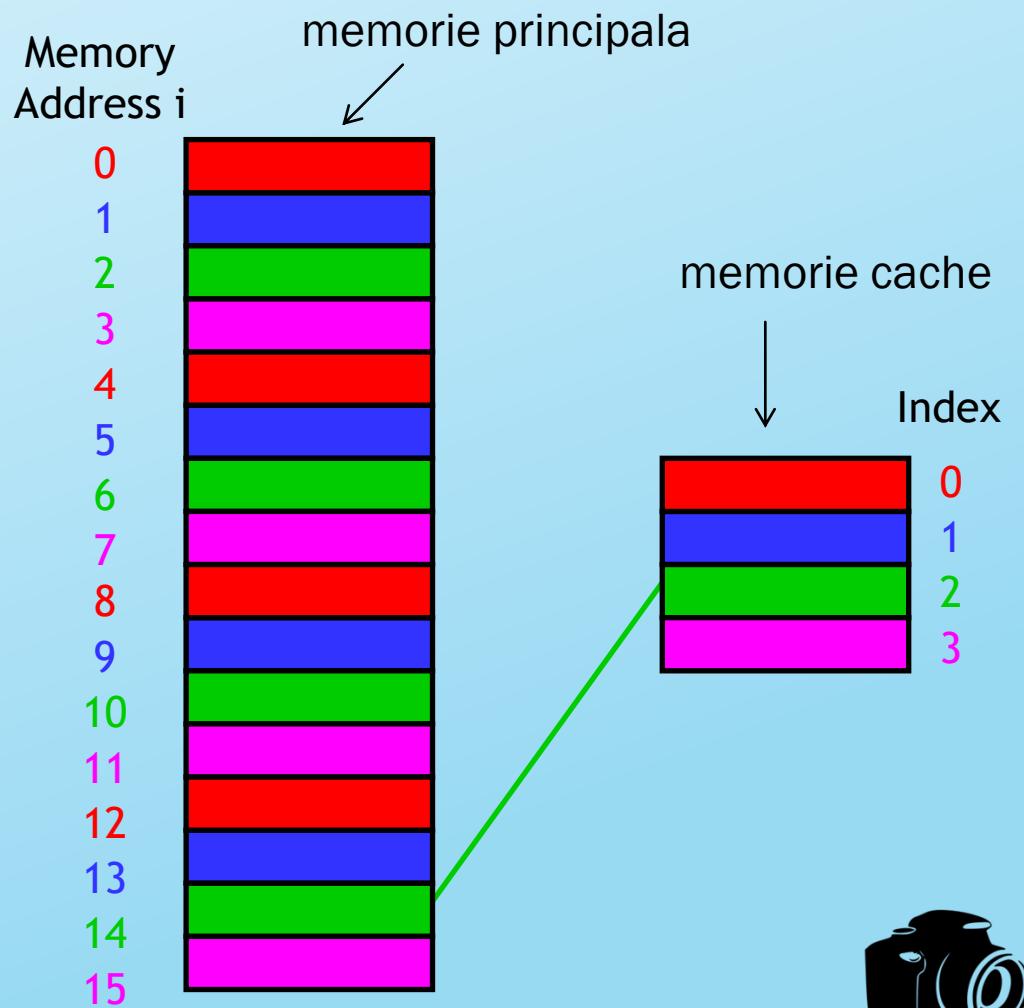
Locațiile de memorie **1, 5, 9 și 13**

vor fi mapate în blocul **1**

Cum se calculează?



- Se foloseste operatorul de calculare a restului!
- Daca memoria cache contine 2^k blocuri atunci cuvantul aflat in memoria principala la adresa i va fi copiat in cache in locatia indicata de *index*:
 - $index = i \bmod 2^k$
 - $(i \% 2^k)$
 - In exemplul nostru adresa 14 va fi mapata in blocul 2.
 - $14 \bmod 4 = 2$



O cale echivalentă: identificarea celor mai putin semnificativi biti din adresa

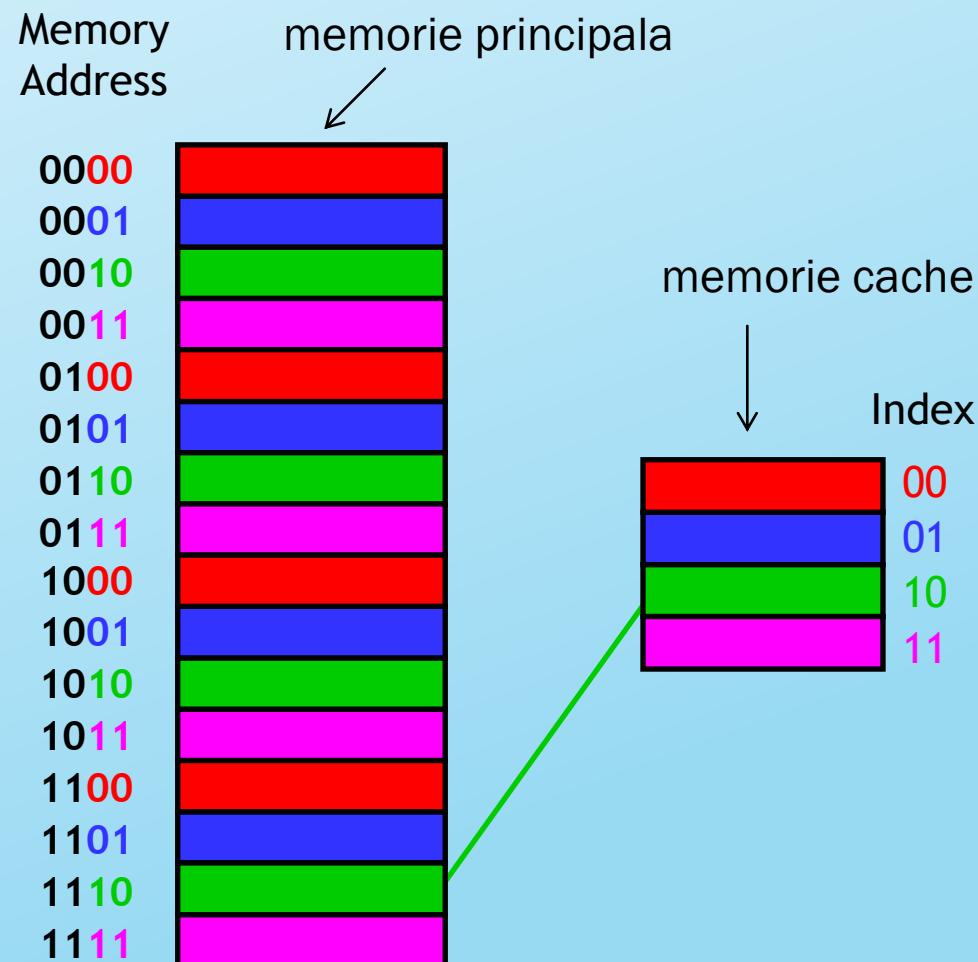
In cazul nostru: ultimii 2 biti

Se poate observa ca

adresa 14 (1110 in binar)

este mapata in
blocul 2 (10 in binar).

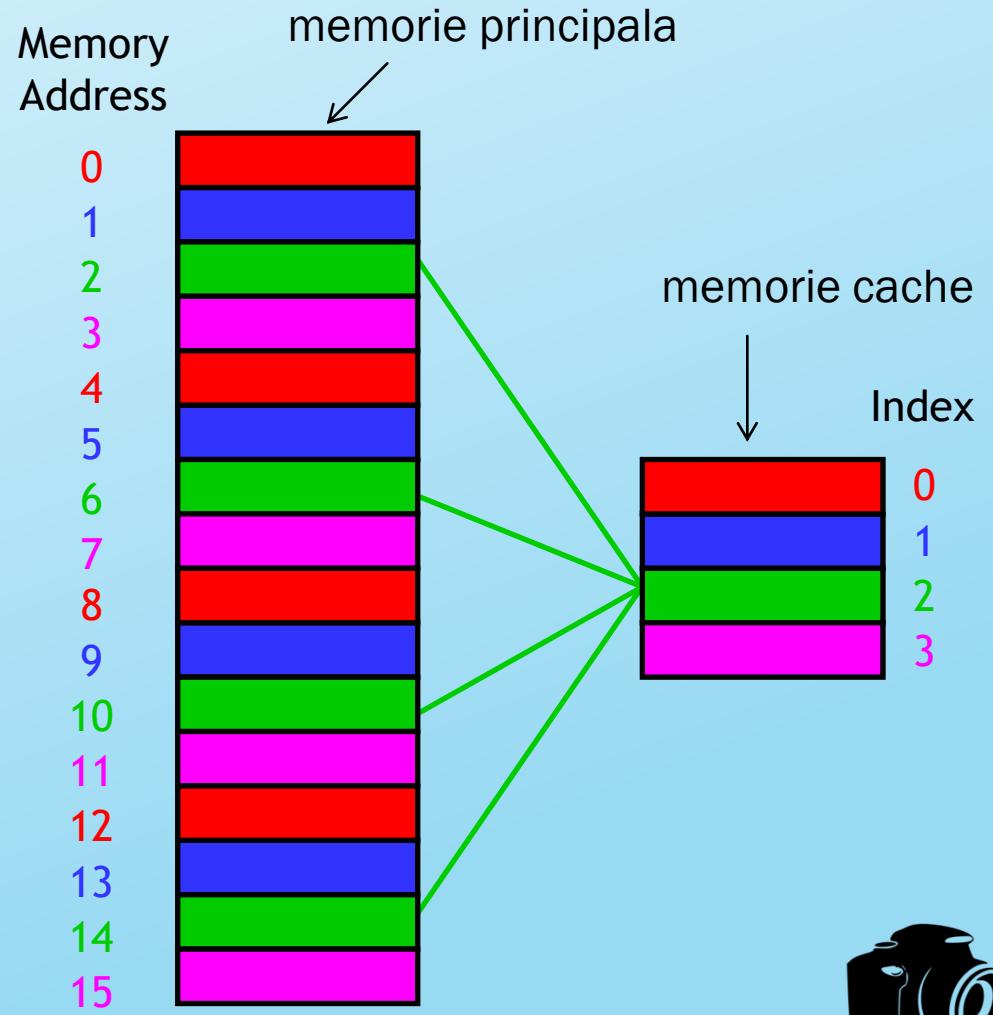
A lua cei mai putin semnificativi k biti
dintr-o valoare binara inseamna a
calcula mod 2^k .



Cum se gasesc datele in cache?

Prin folosirea metodei inverse: soluția este nedeterminata (nu este unica!)

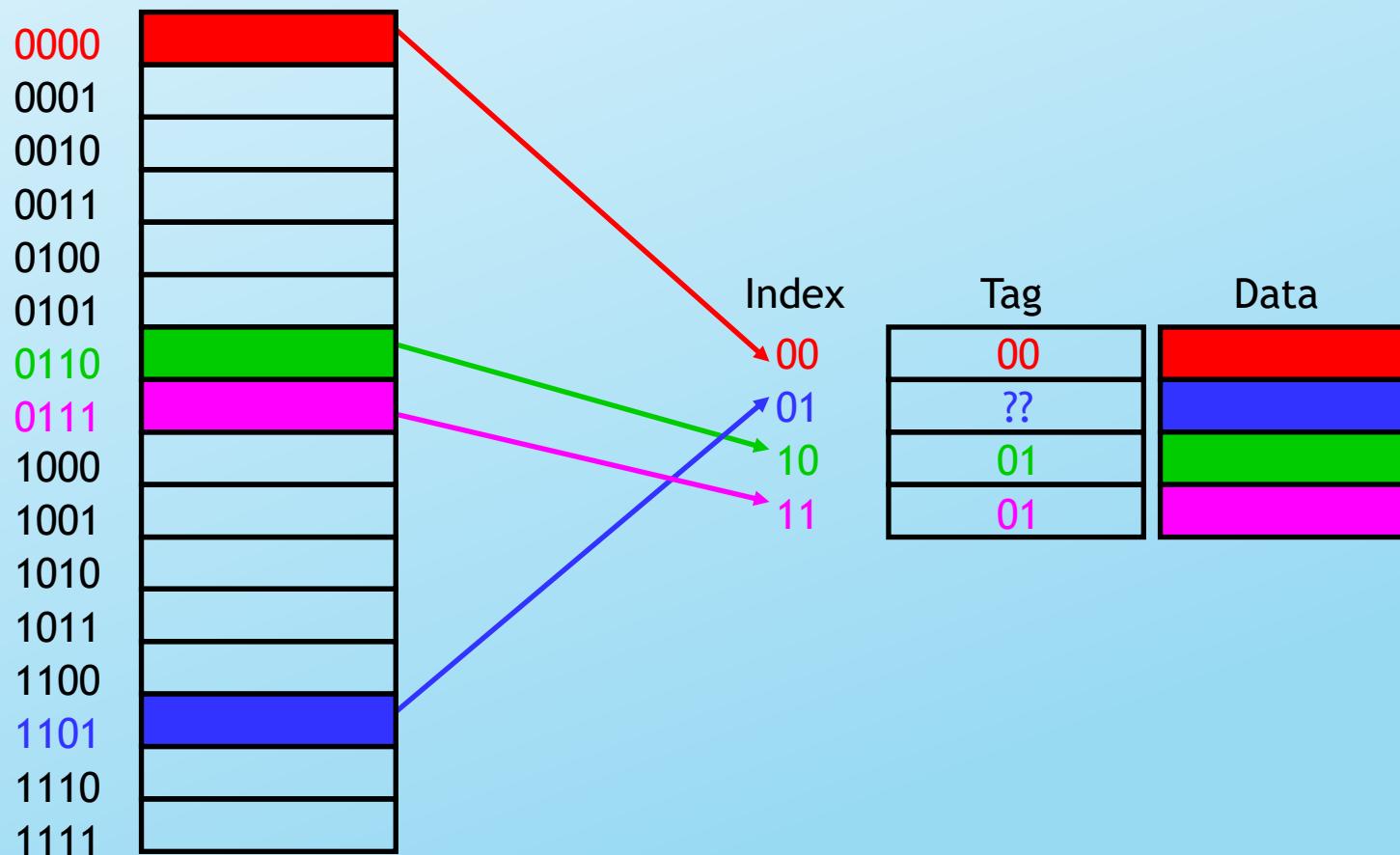
De exemplu: blocul 2
din cache poate conține date de
la adresele 2, 6, 10 sau 14.

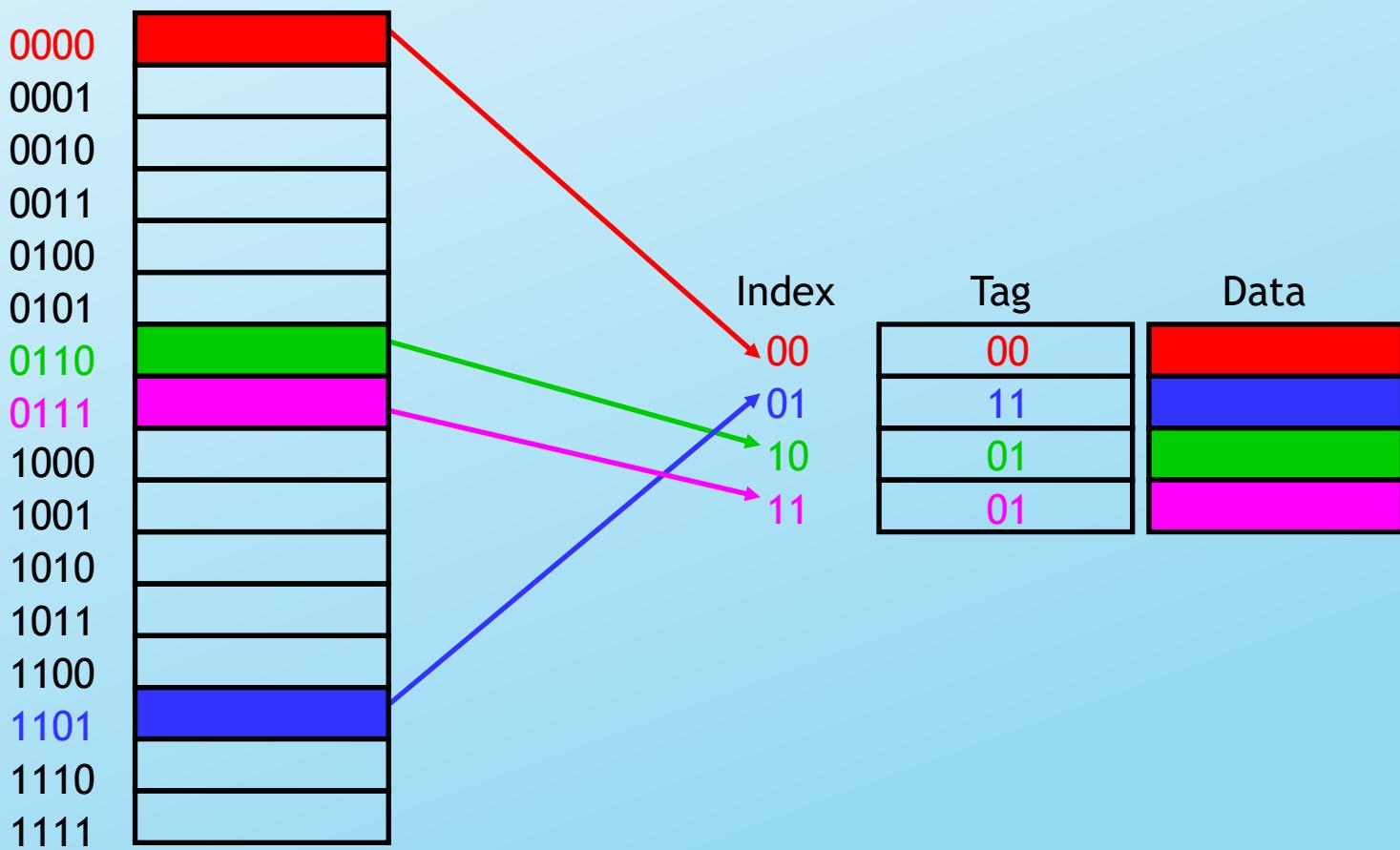


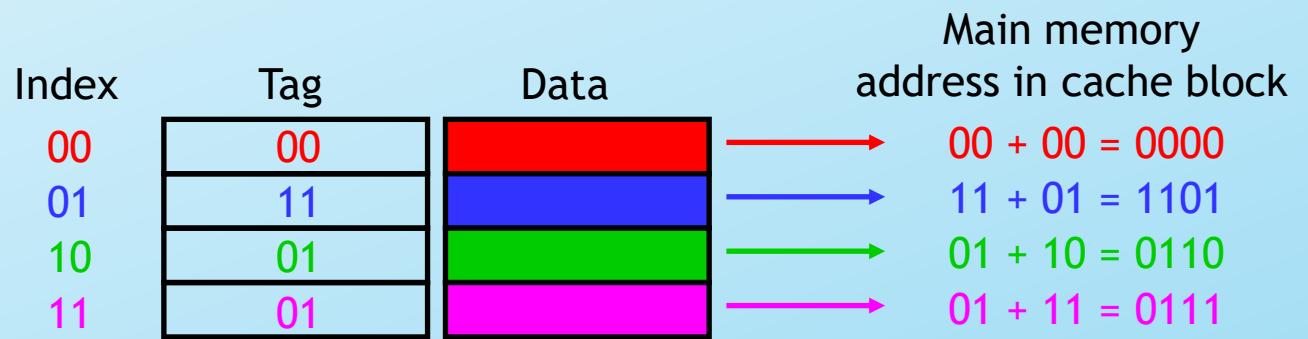
Trebuie introdus in cache o informatie suplimentara: **tag**-ul.

Suplineste lipsa informatiilor (restul de adresa)

Distinge locatia de memorie din care provine blocul.







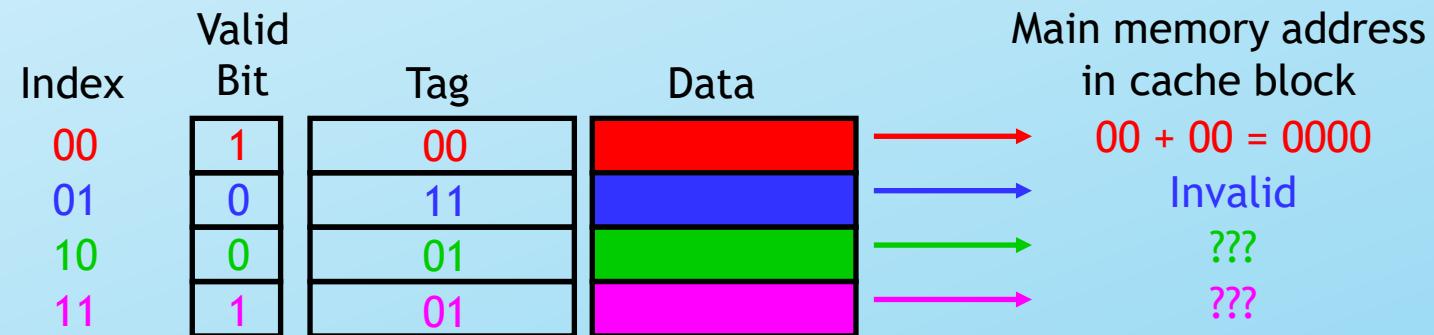
Tag+Index=adresa



O informație suplimentara bitul de prezență (**valid bit**)

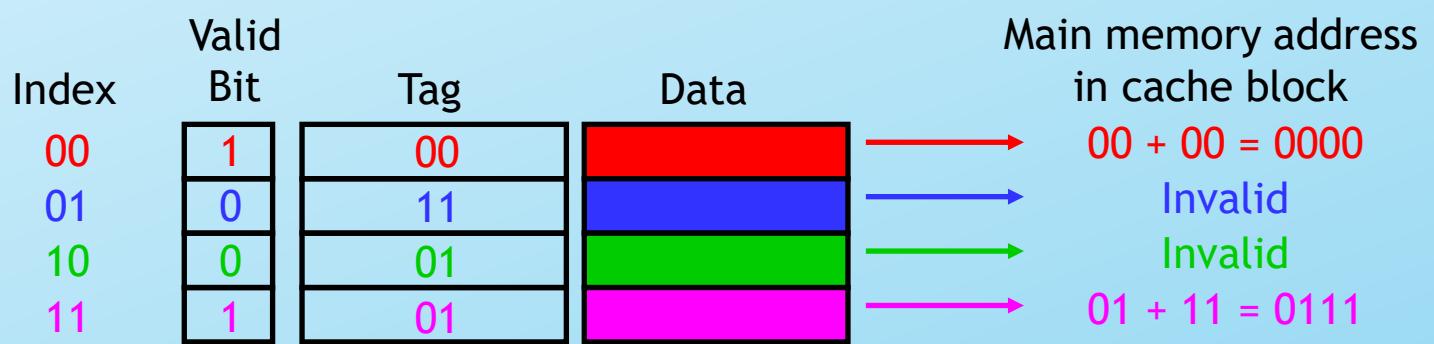
“lupta împotriva gunoiului”

In cache pot ramane informații din operațiuni anterioare care nu mai sunt valide (de exemplu fac parte din alt program, cu execuție finalizată, sau care au fost înlocuite în memoria principală)



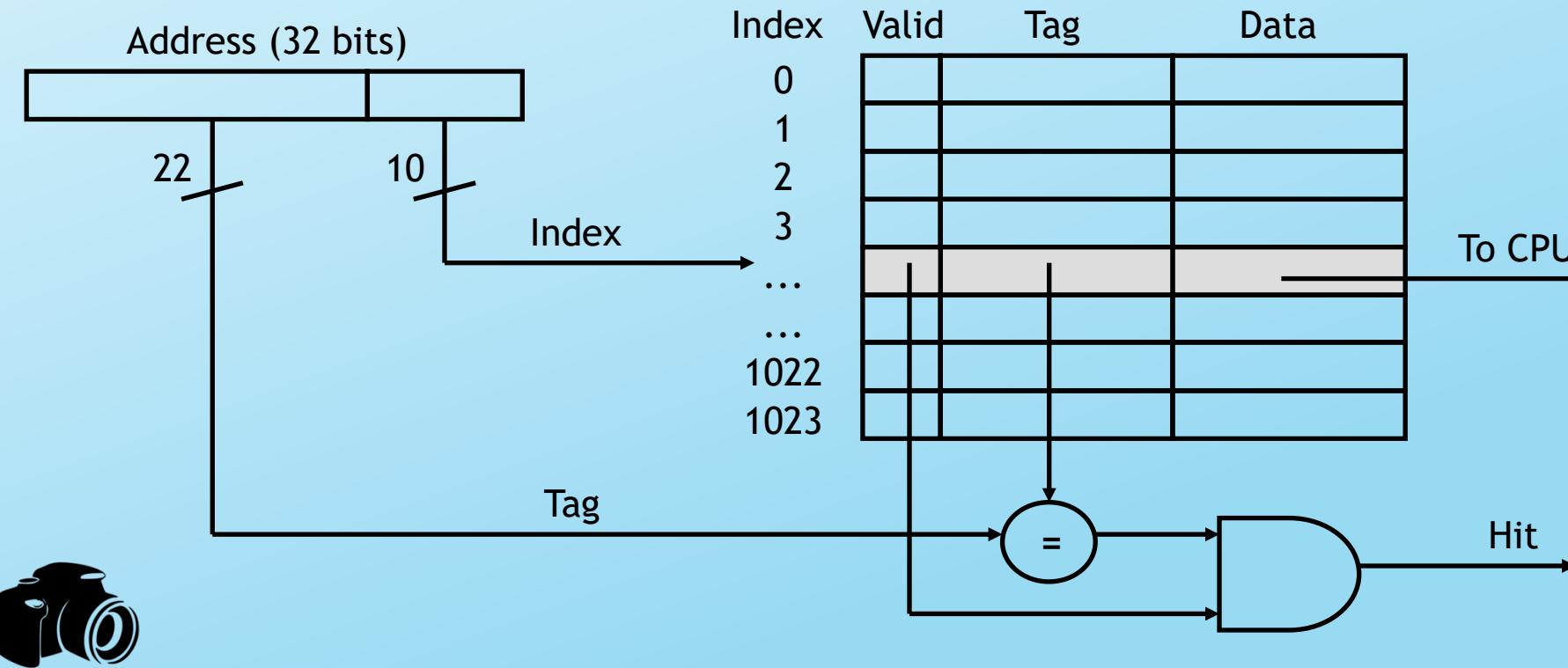
Bitul de prezență are valoarea 1 pentru date valide!





Ce se intampla in cazul unui **cache hit**

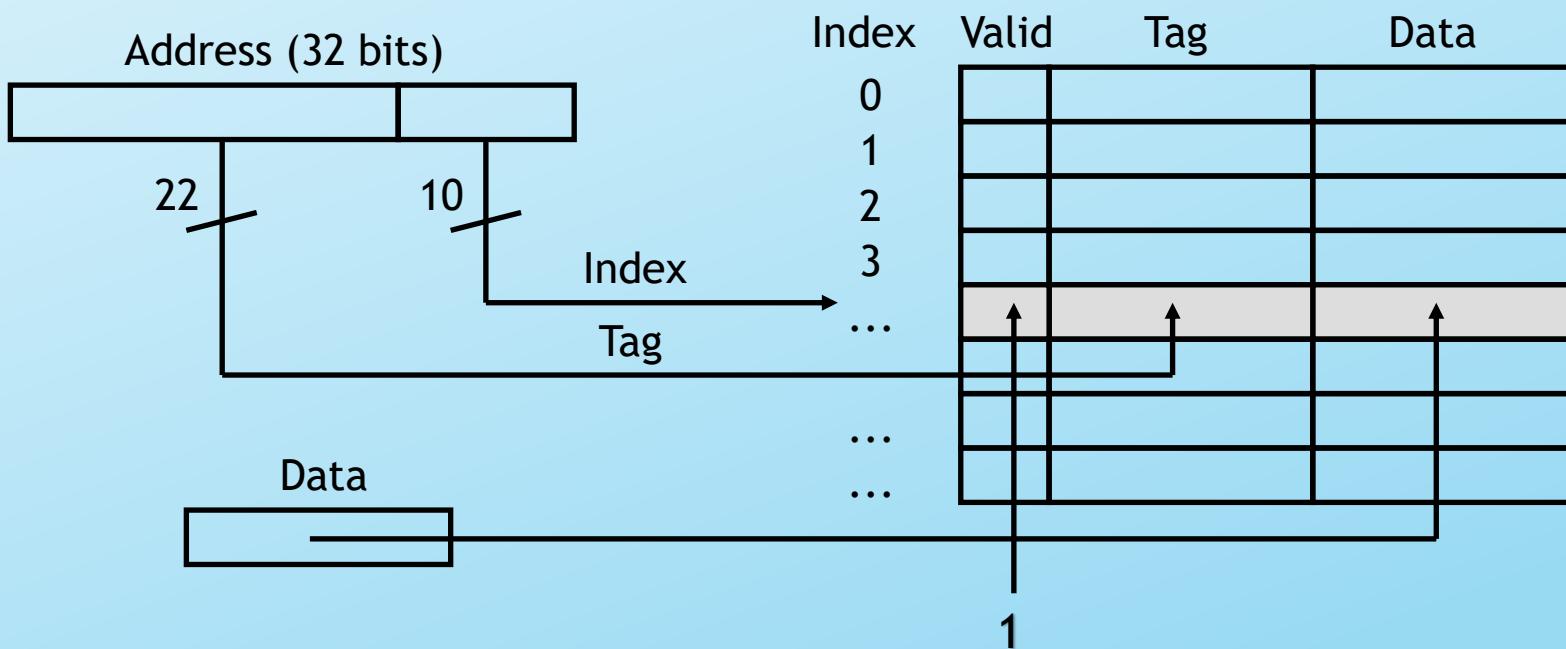
- Cand CPU initiaza citirea din memorie, adresa (m biți) este trimisa unui **controler de cache**.
 - Cei mai putin semnificativi k biți sunt transformati in **index** (cache) si se deschide accesul catre blocul corespunzator.
 - Daca blocul este **valid** si tag-ul se potriveste cu cei mai semnificativi ($m-k$) biti informația va fi trimisa catre CPU (impreuna cu un semnal de **hit**).



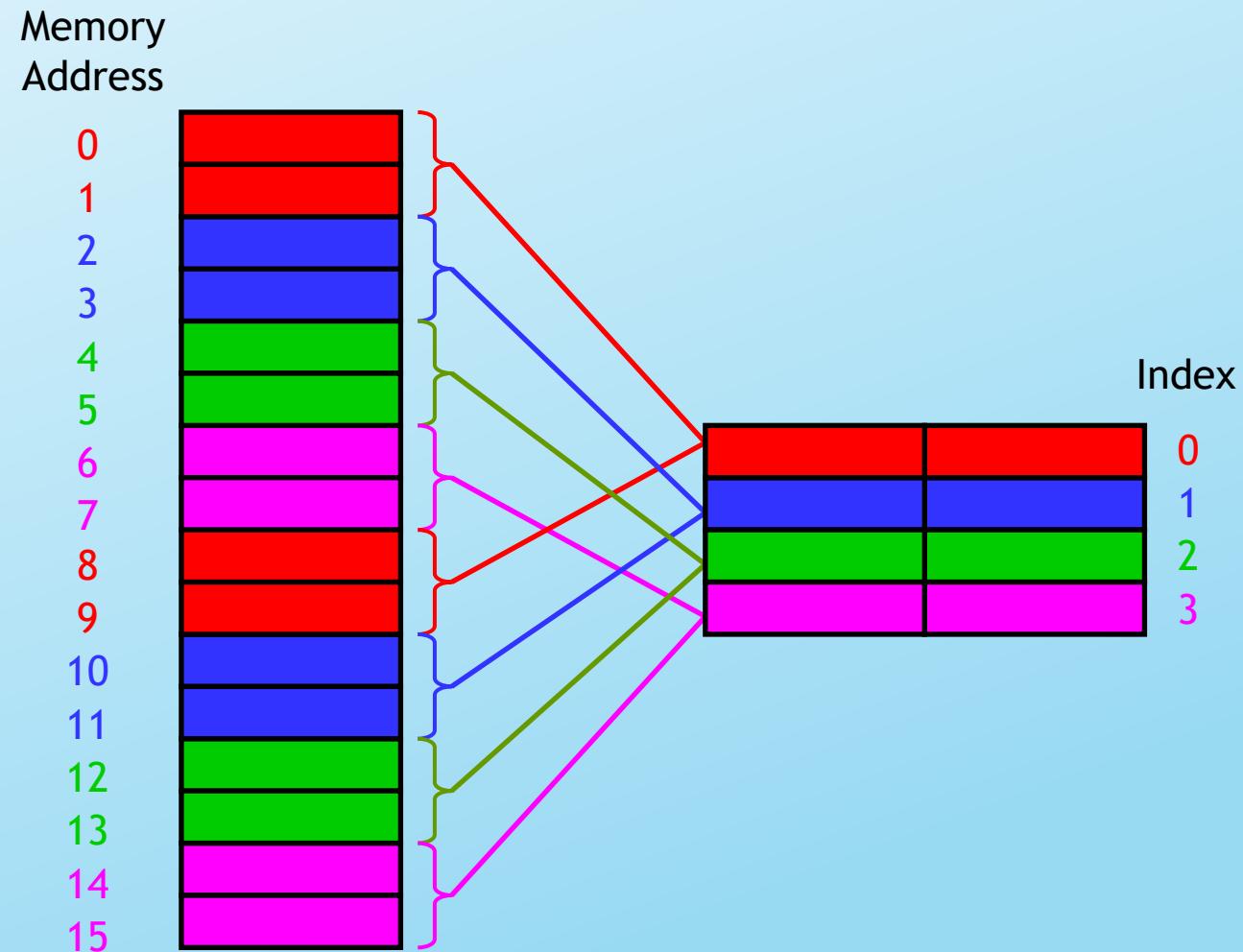
Ce se intampla in caz de **cache miss**

- Daca CPU **nu** primeste semnalul hit, (adica primeste 0 si nu 1) initiaza operatiunea de citire din memoria de rang imediat inferior. (in cazul sistemului cu un singur cache, citeste direct din memoria principală)

Incarcarea unui bloc in cache

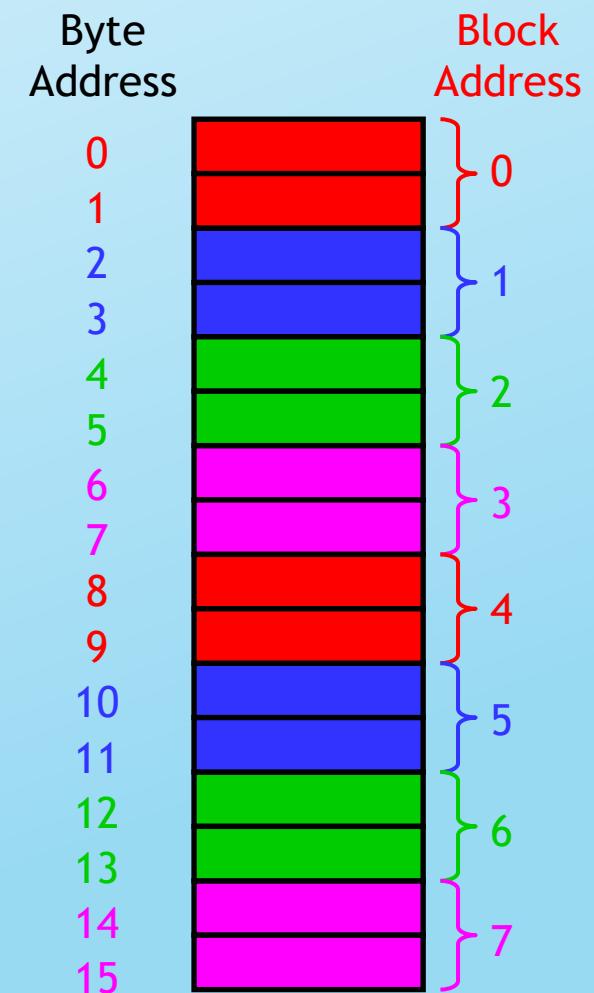


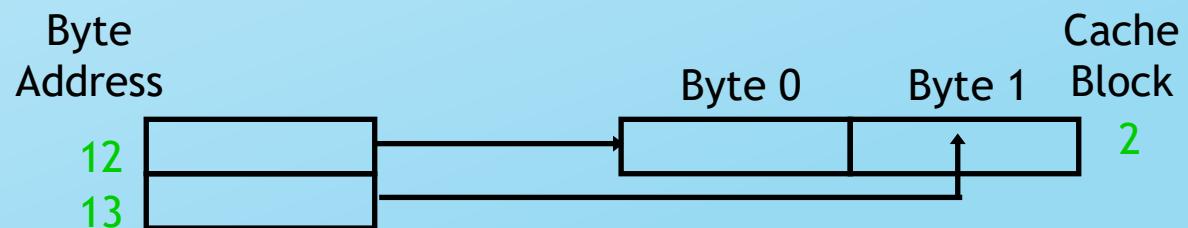
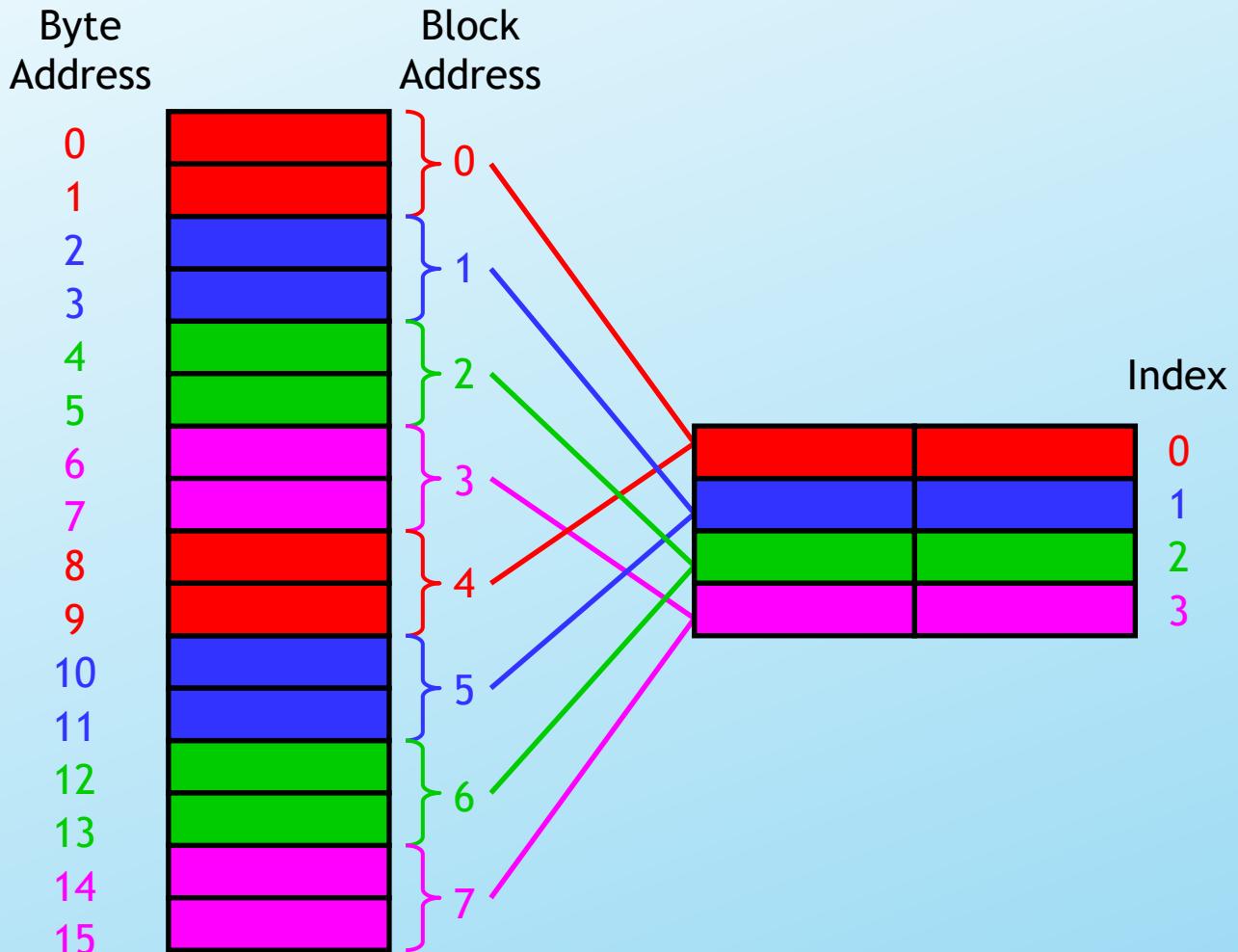
Memorii cache cu blocuri mai mari de 1 byte



Adrese de bloc

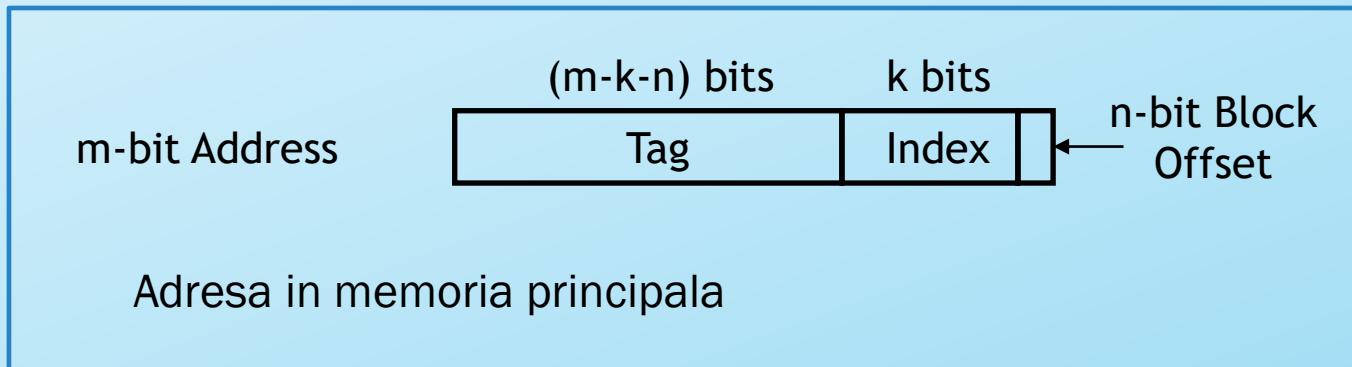
- Daca un bloc din cache are 2^n byte, putem sa divizam memoria principala in fragmente de 2^n byte.
- Pentru a determina adresa (indexul) blocului din cache, adresa din memoria principala i , se imparte la 2^n (impartire intreaga)
 $i / 2^n$
- In exemplul nostru:
- Putem sa gandim o memorie principala de 16 byte ca o memorie cu 8 blocuri .
- De exemplu, adresele 12 si 13 corespund adresei de bloc 6, deoarece $12 / 2 = 6$ and $13 / 2 = 6$.





Localizarea datelor in cache

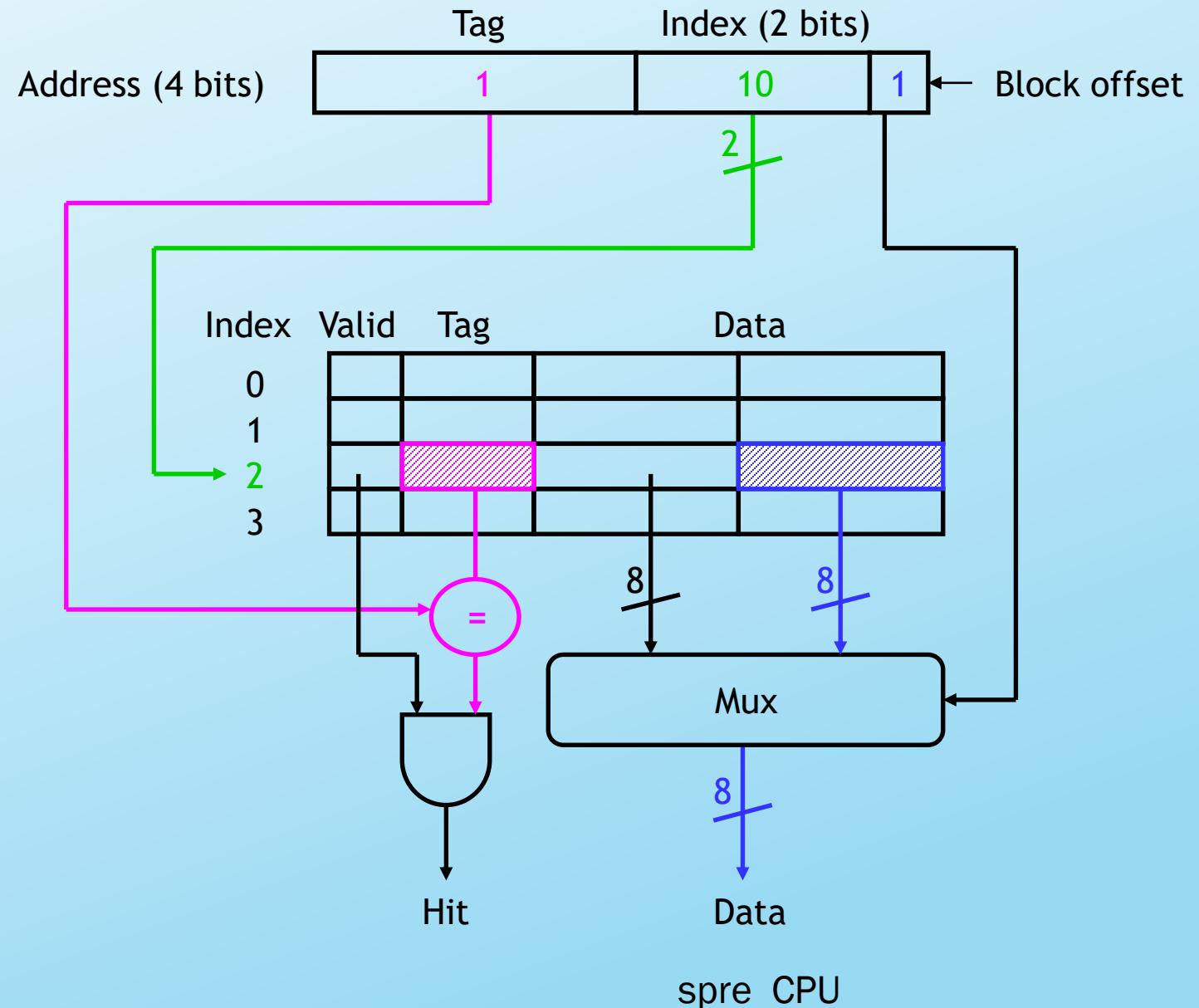
- Sa presupunem ca avem un cache cu 2^k blocuri continand 2^n byte.
- Putem determina pozitia in cache a unui octet plecand de la adresa sa din memoria principala.
 - Cei mai putin semnificativi n biti vor determina **offset**-ul care decide pozitia octetului in bloc.
 - Cei k biti vor identifica unul din cele 2^k blocuri din cache (index)



In exemplul nostru folosim un cache cu 2^2 blocuri si 2^1 byte per bloc. Astfel continutul adresei 13 (1101) va fi stocata in locatia 1 din blocul 2.



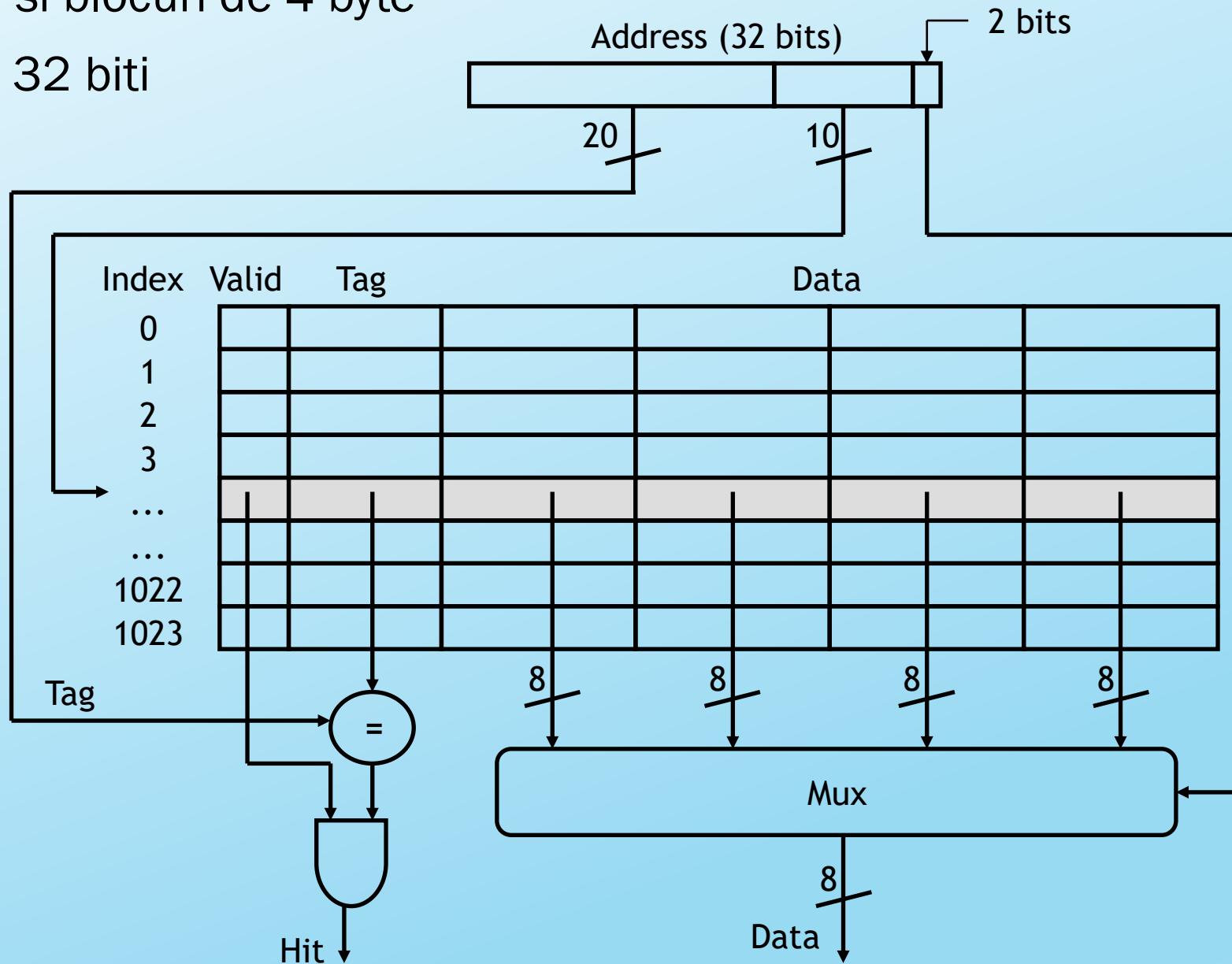
dinspre CPU

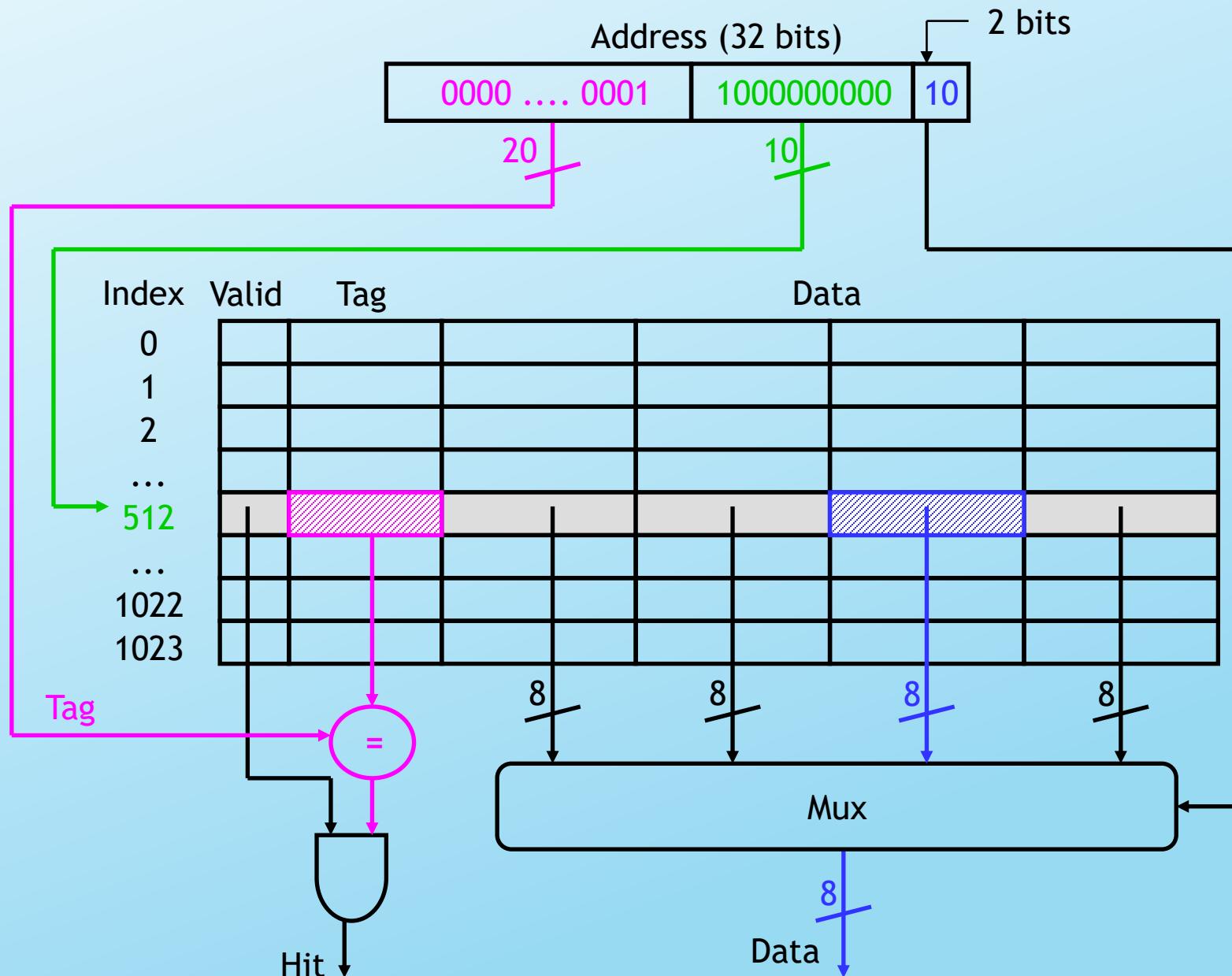


Un alt exemplu

Cache cu 1024 blocuri si blocuri de 4 byte

Memorie cu adrese de 32 biti



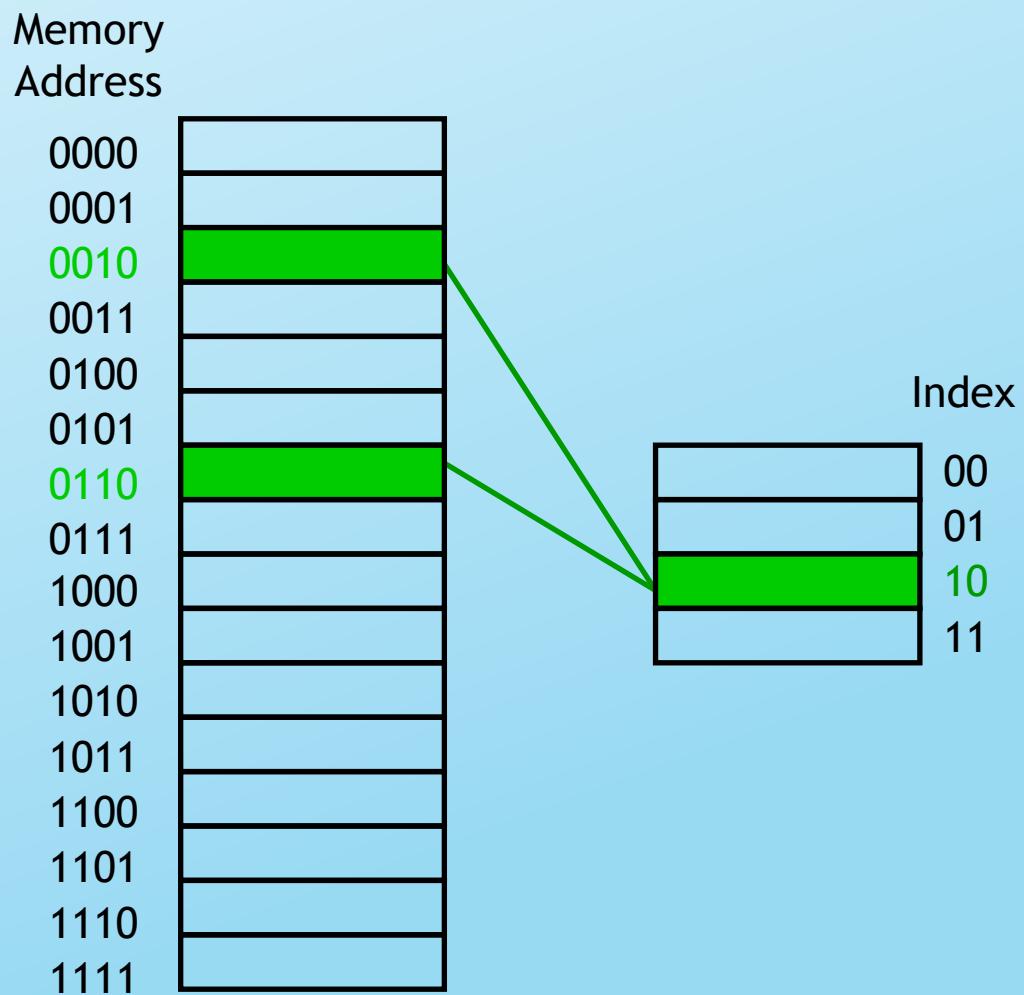


Dezavantajele maparii directe

Daca programul utilizeaza succesiv
adresele 2, 6, 2, 6, 2,

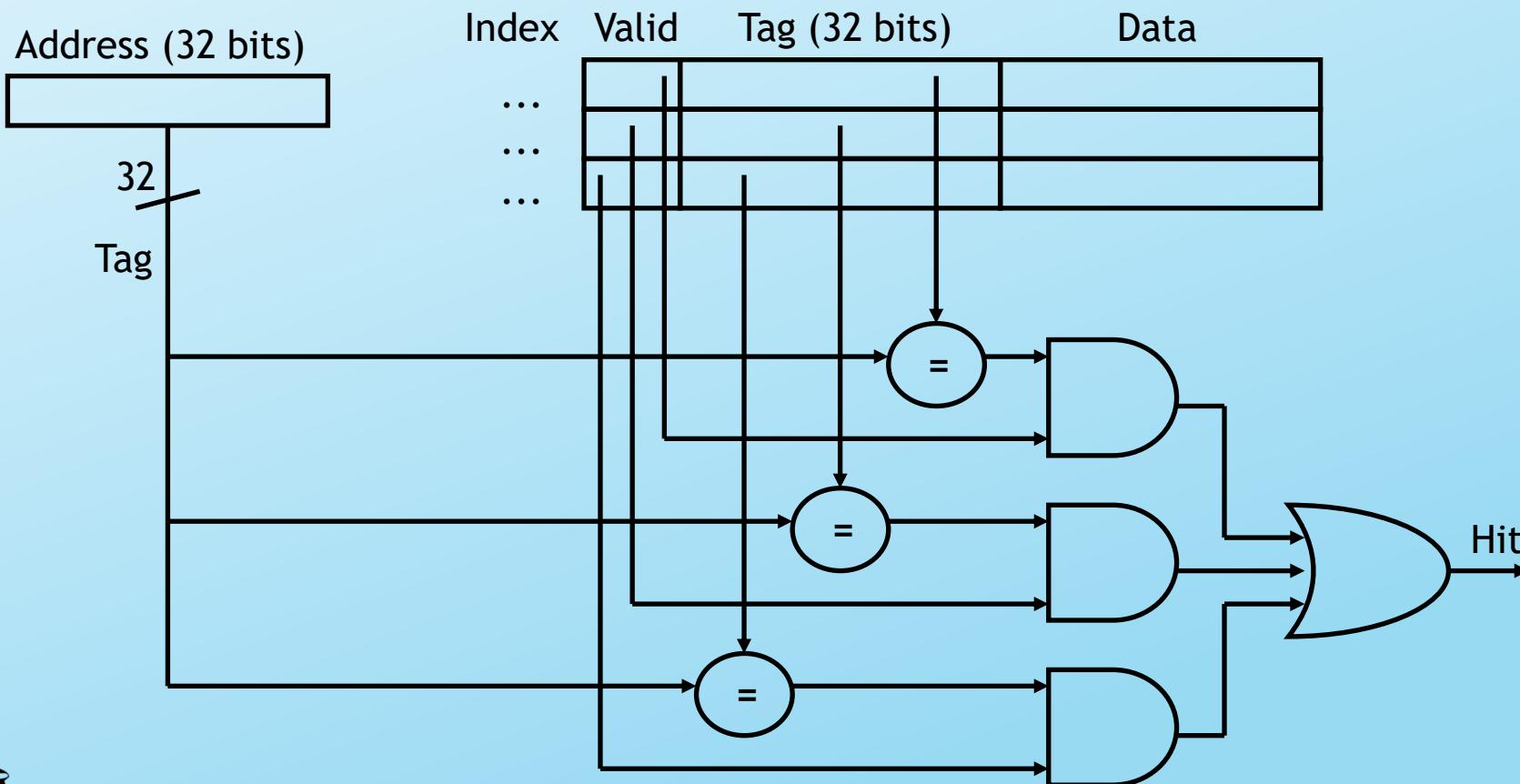
Rezulta **cache miss** de fiecare data

Ingreuneaza accesul la memorie



Maparea complet asociativa

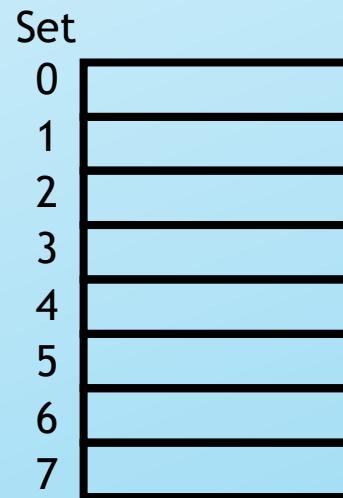
- Datele pot fi amplasate în orice bloc
 - tag = adresa completa



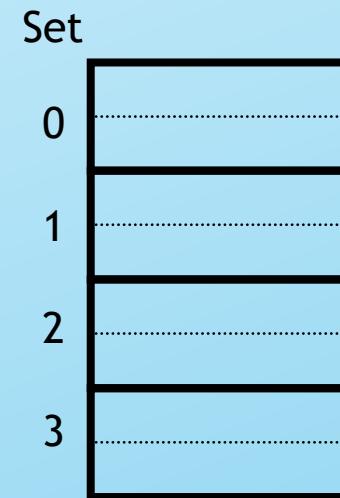
Maparea directă în seturi asociative

Blocurile sunt grupate în seturi. Datele sunt înmagazinate orice bloc dintr-un set, dar într-un anumit set.

1-way associativity
8 sets, 1 block each



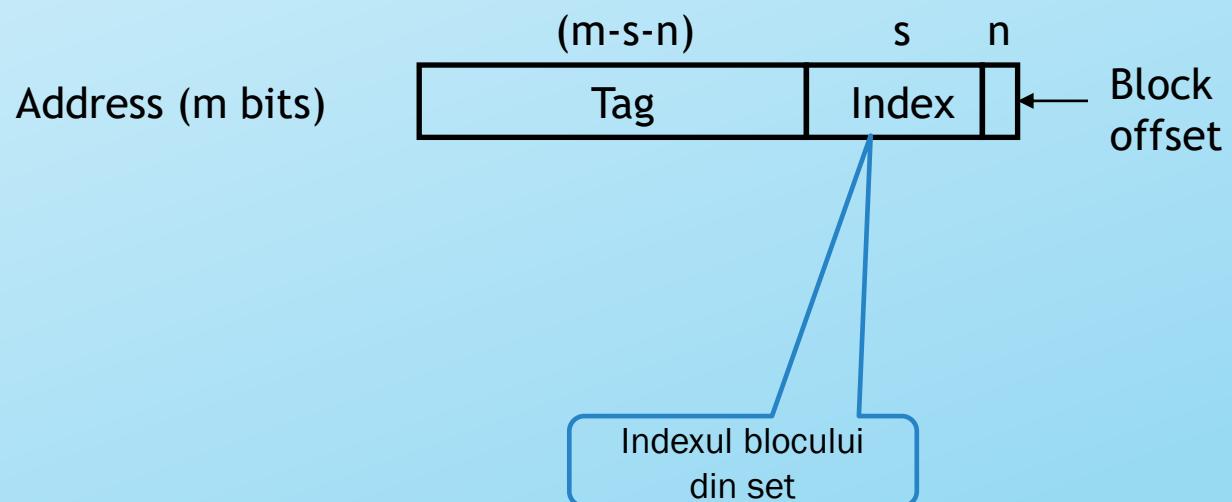
2-way associativity
4 sets, 2 blocks each



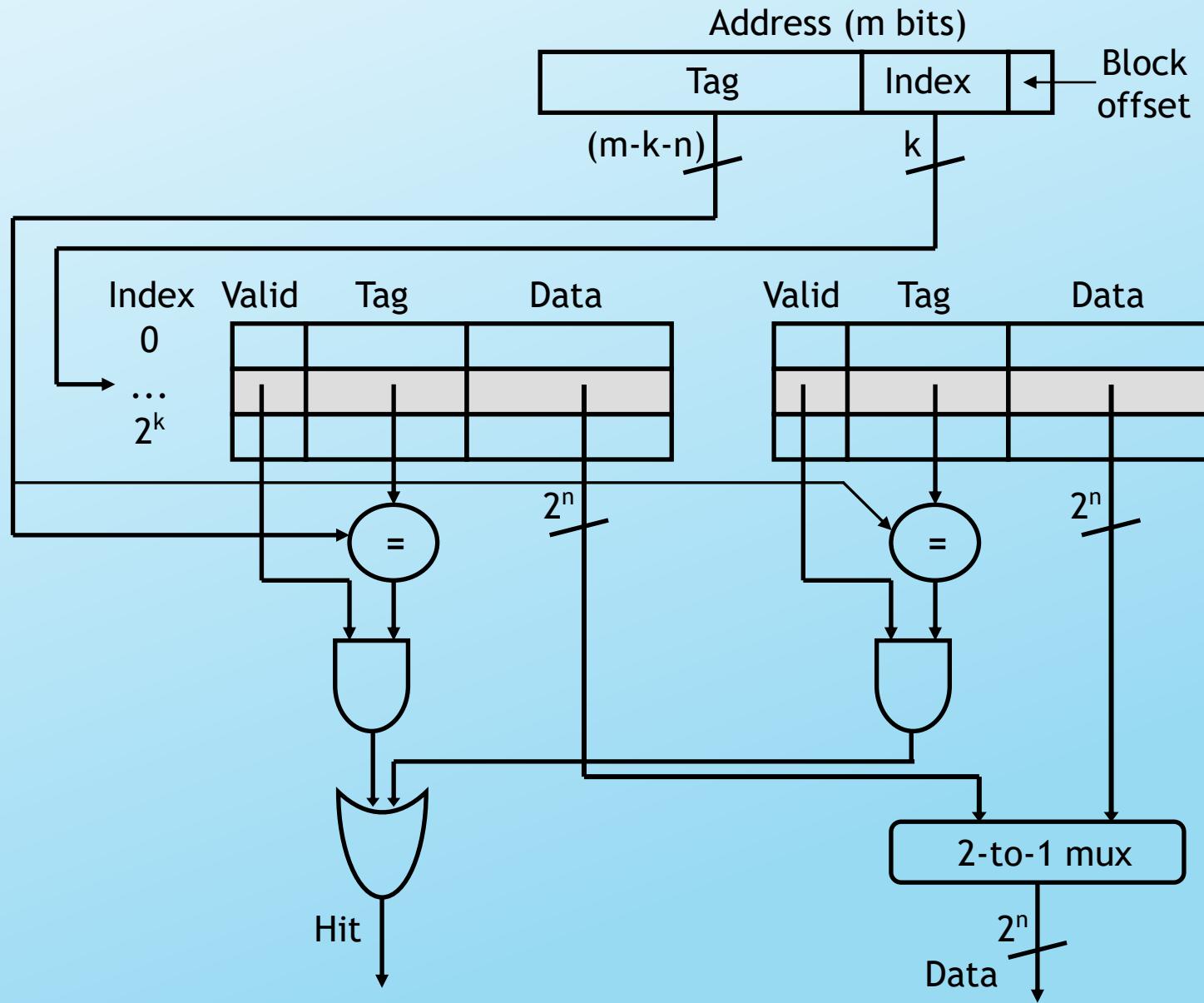
4-way associativity
2 sets, 4 blocks each

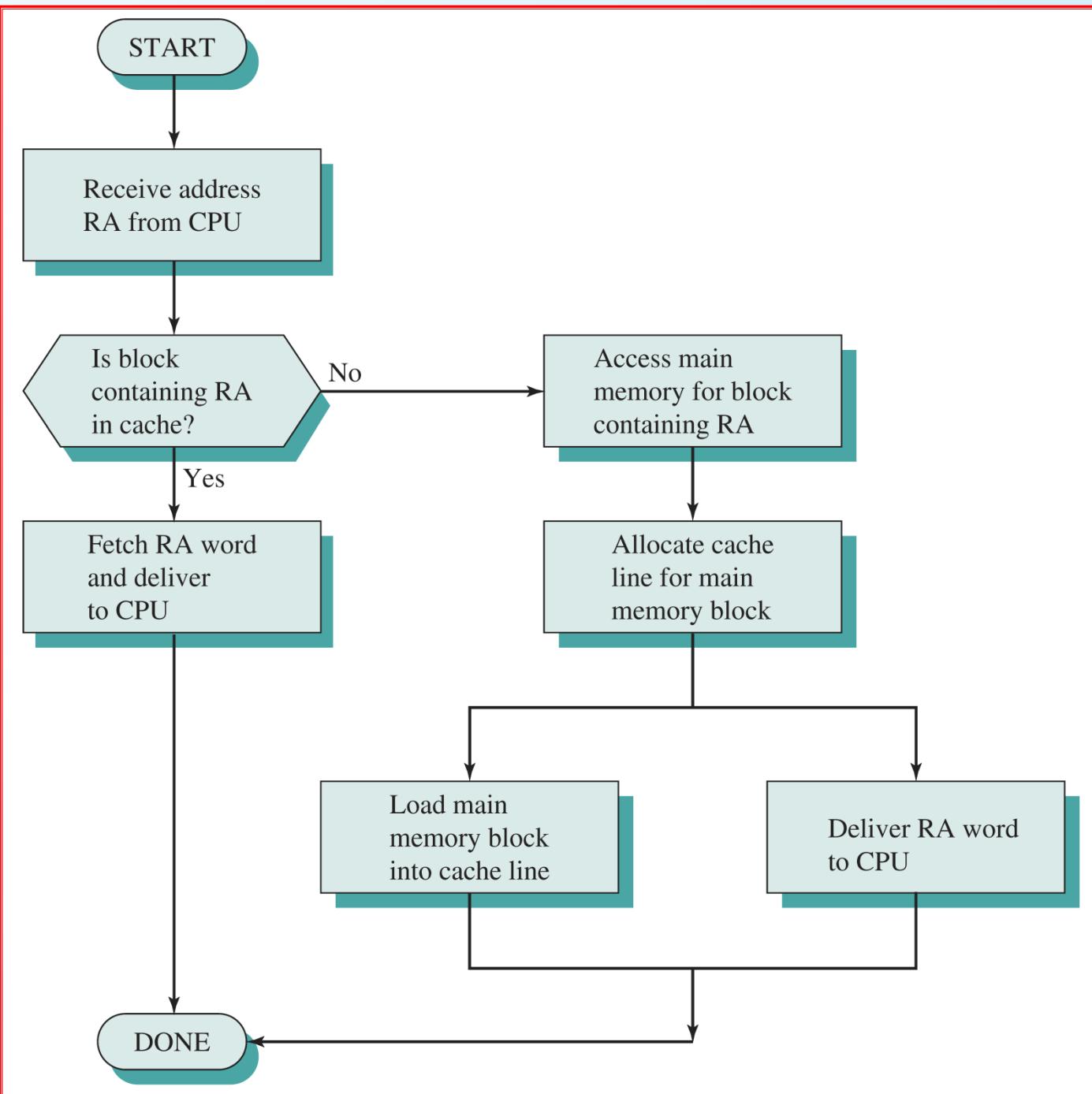


- Cache cu 2^s seturi ,
- Fiecare set cu 2^n blocuri (2^n -way associative cache)
- Fiecare bloc cu 2^m byte

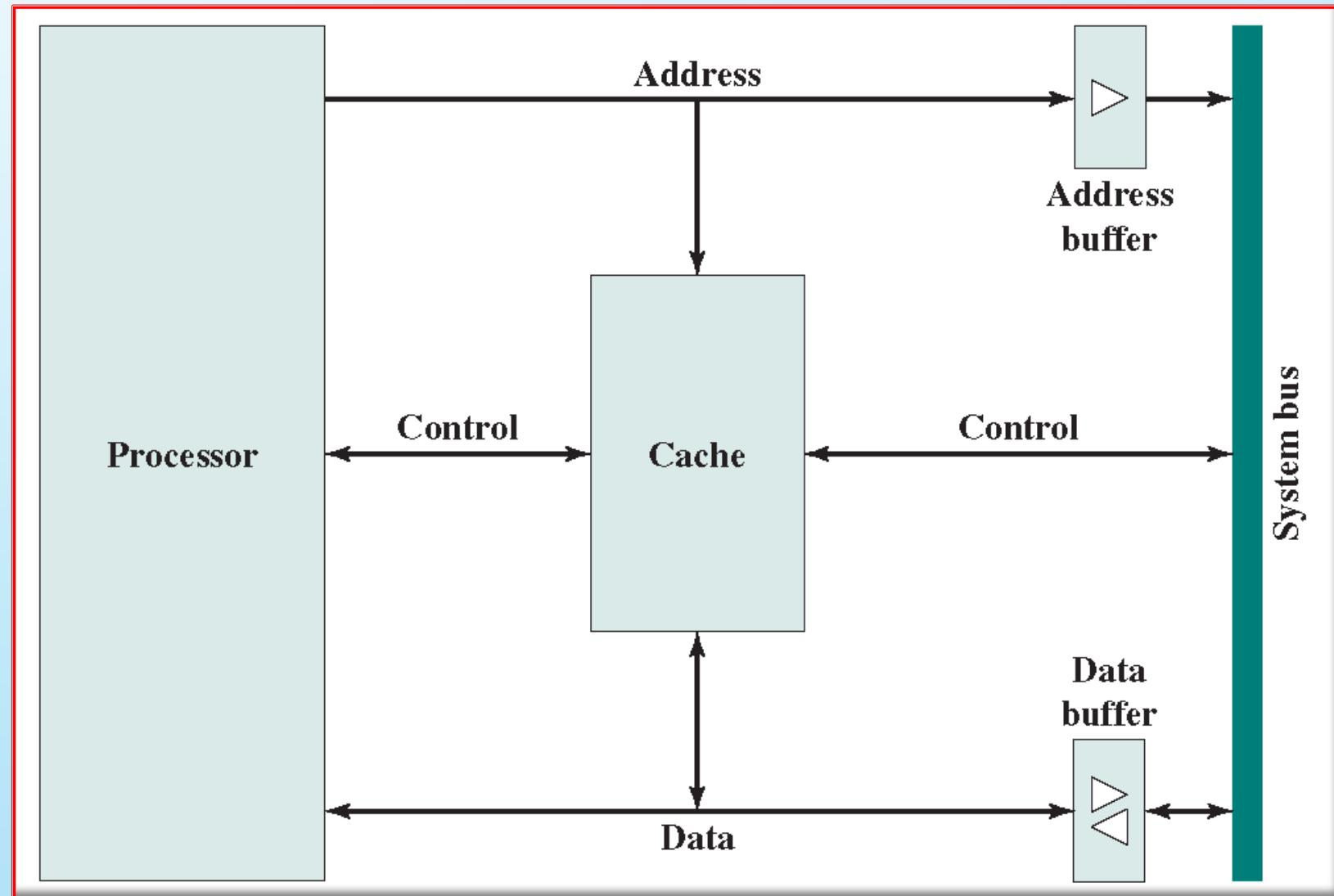


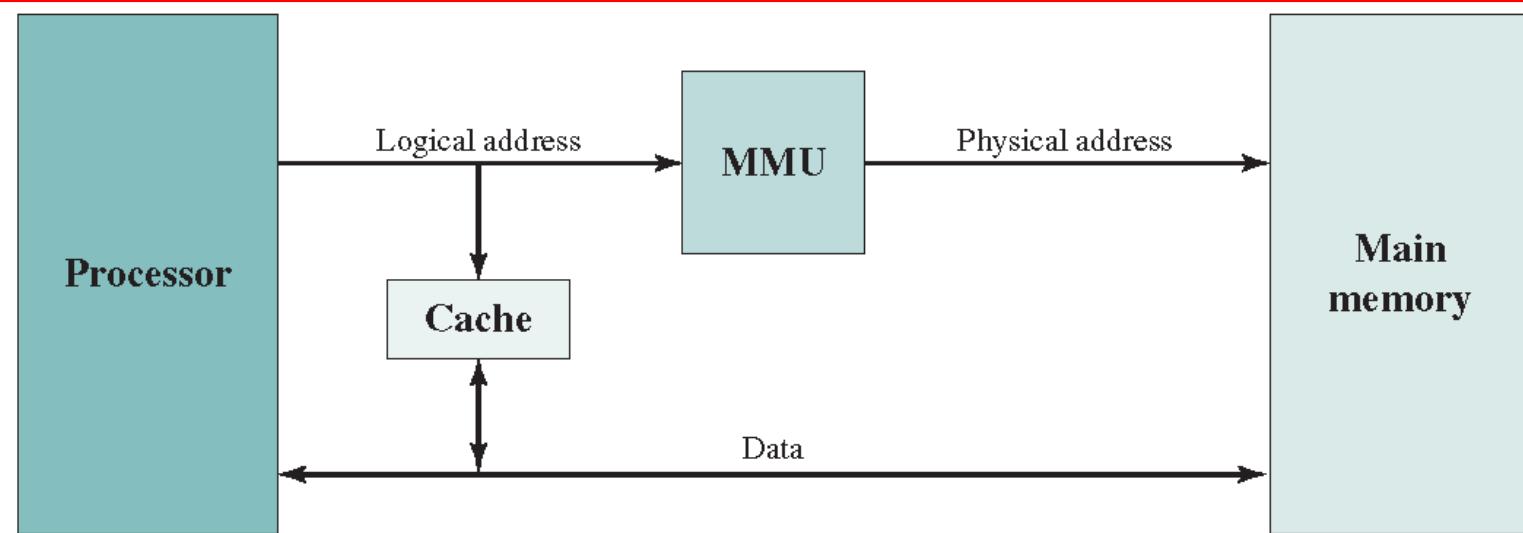
2-way set associative cache



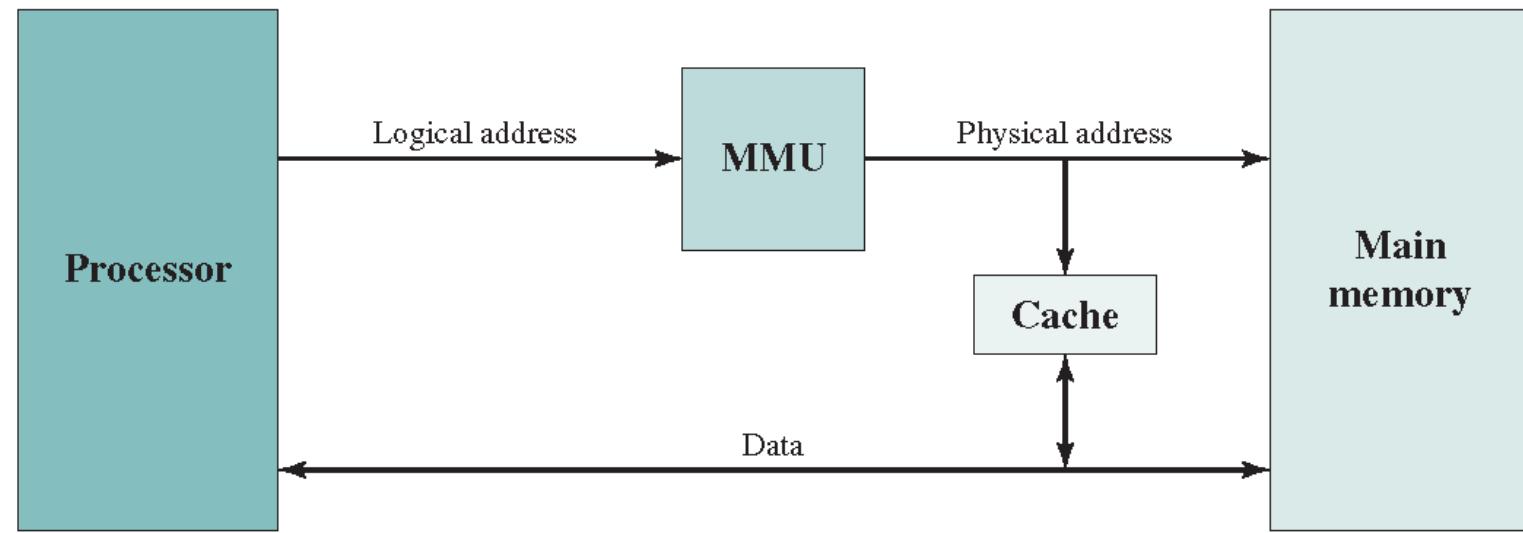


RA= read address





(a) Logical cache



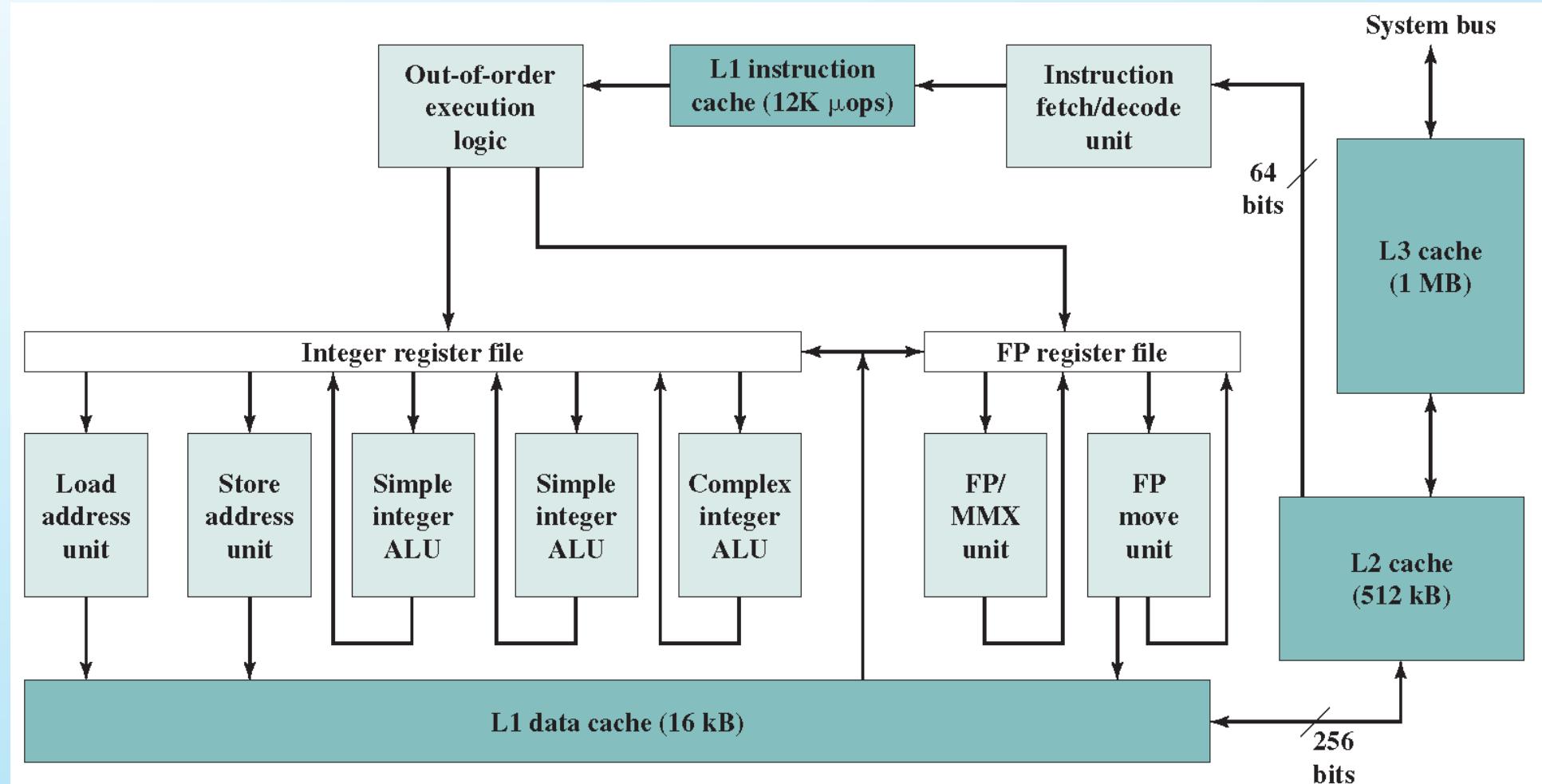
(b) Physical cache

MMU="MEMORY MANAGEMENT UNIT"

TRANSLATEAZA ADRESELE LOGICE IN ADRESE FIZICE

Processor	Type	Year of Introduction	L1 Cache^a	L2 Cache	L3 Cache
IBM 360/85	Mainframe	1968	16–32 kB	—	—
PDP-11/70	Minicomputer	1975	1 kB	—	—
VAX 11/780	Minicomputer	1978	16 kB	—	—
IBM 3033	Mainframe	1978	64 kB	—	—
IBM 3090	Mainframe	1985	128–256 kB	—	—
Intel 80486	PC	1989	8 kB	—	—
Pentium	PC	1993	8 kB/8 kB	256–512 kB	—
PowerPC 601	PC	1993	32 kB	—	—
PowerPC 620	PC	1996	32 kB/32 kB	—	—
PowerPC G4	PC/server	1999	32 kB/32 kB	256 kB to 1 MB	2 MB
IBM S/390 G6	Mainframe	1999	256 kB	8 MB	—
Pentium 4	PC/server	2000	8 kB/8 kB	256 kB	—
IBM SP	High-end server/ supercomputer	2000	64 kB/32 kB	8 MB	—
CRAY MTA ^b	Supercomputer	2000	8 kB	2 MB	—
Itanium	PC/server	2001	16 kB/16 kB	96 kB	4 MB
Itanium 2	PC/server	2002	32 kB	256 kB	6 MB
IBM POWER5	High-end server	2003	64 kB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 kB/64 kB	1 MB	—
IBM POWER6	PC/server	2007	64 kB/64 kB	4 MB	32 MB
IBM z10	Mainframe	2008	64 kB/128 kB	3 MB	24–48 MB
Intel Core i7 EE 990	Workstation/ server	2011	6 × 32 kB/ 32 kB	1.5 MB	12 MB
IBM zEnterprise 196	Mainframe/ server	2011	24 × 64 kB/ 128 kB	24 × 1.5 MB	24 MB L3 192 MB L4

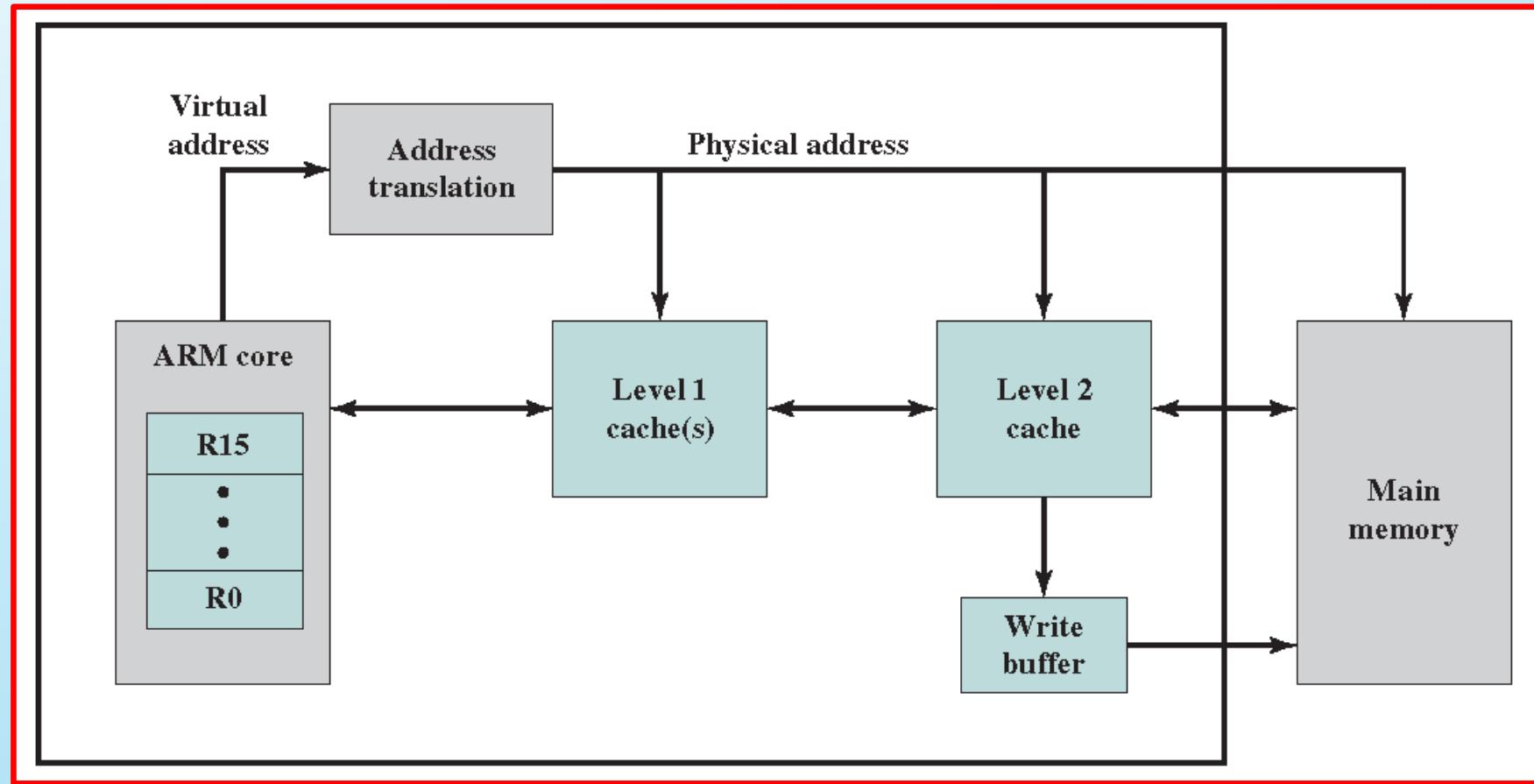




CACHE UTILIZAT DE MICROPROCESORUL PENTIUM 4

Core	Cache Type	Cache Size (kB)	Cache Line Size (words)			Write Buffer Size (words)
ARM720T	Unified	8	4	4-way	Logical	8
ARM920T	Split	16/16 D/I	8	64-way	Logical	16
ARM926EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	16
ARM1022E	Split	16/16 D/I	8	64-way	Logical	16
ARM1026EJ-S	Split	4-128/4-128 D/I	8	4-way	Logical	8
Intel StrongARM	Split	16/16 D/I	4	32-way	Logical	32
Intel Xscale	Split	32/32 D/I	8	32-way	Logical	32
ARM1136-JF-S	Split	4-64/4-64 D/I	8	4-way	Physical	32

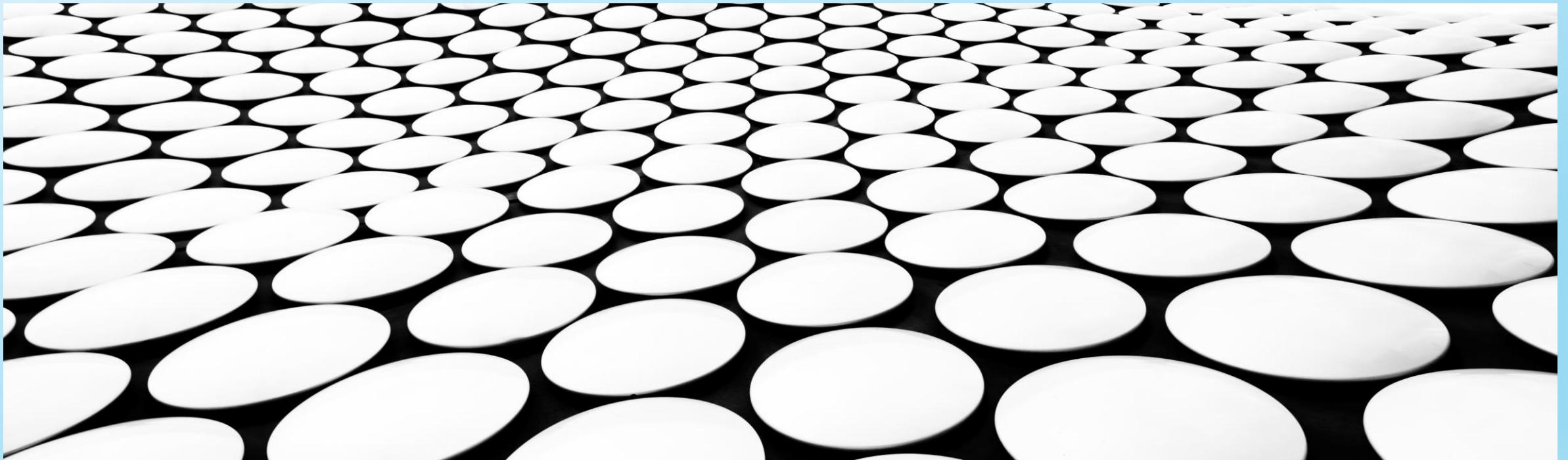
ARM



ARM

ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



Intreruperi

O **intrerupere** este un **semnal** trimis catre CPU si emis de catre un bloc hardware distinct (de exemplu un periferic) cu scopul de a intrerupe execuția secvenței curente de cod.

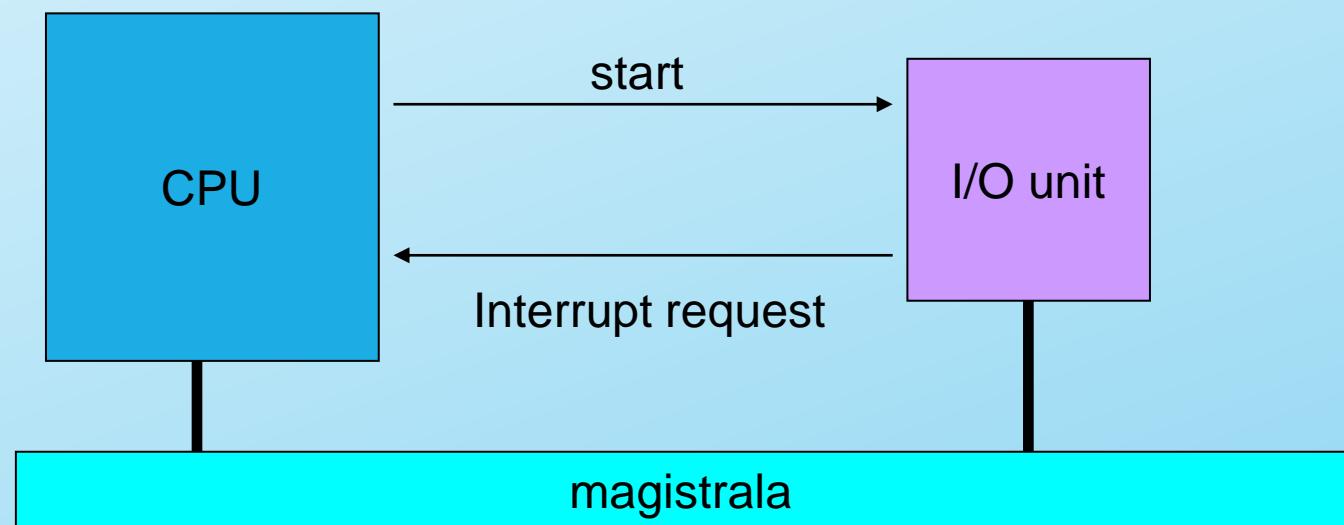
CPU raspunde prin:

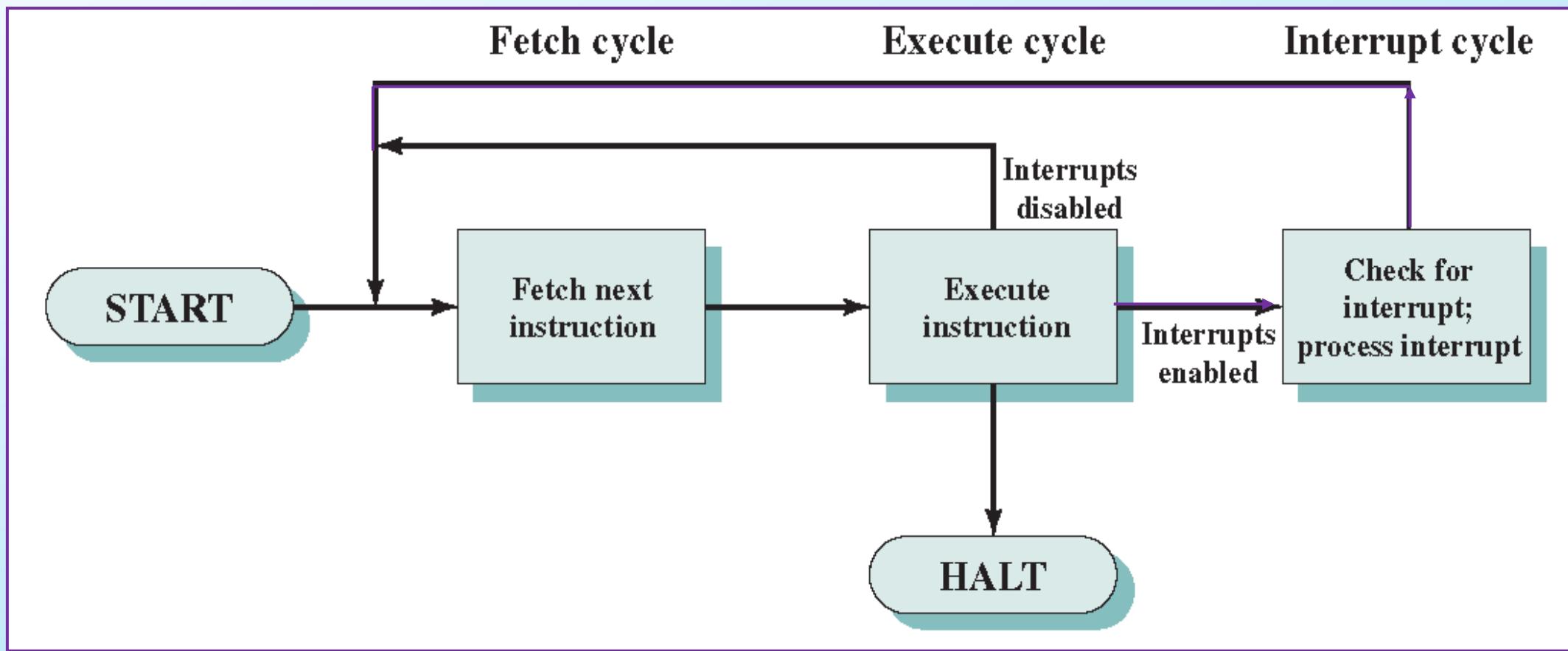
- suspendarea activitațiilor curente
- salvarea informațiilor de stare
- executarea unei funcții de tratare a intreruperii (o secvență de program)
(interrupt handler sau Interrupt Service Routine)
- Dupa executarea acestei funcții CPU revine la activitatea normală

- Primul sistem de calcul cu intreruperi:
 - Univac 1103/1103A (1953-1956)

Clase de intreruperi

- Intreruperi de program
 - Generate de execuția unei secvențe de cod (ex.: împărțirea prin 0)
- Intreruperi de timer
 - Generate de un timer intern (permite indeplinirea anumitor funcții la intervale regulate de timp)
- Intreruperi I/O
 - Generate de un controler de I/O (pt. a semnala terminarea unei operații, solicitarea unui serviciu, etc.)
- Intreruperi de avarie (Failure)
 - Apar în urma unei caderi de tensiune, a unei erori de memorie, etc
- Intreruperi mascabile/nemascabile
 - Intreruperi ce pot/nu pot fi ignorate
 - Intreruperile nemascabile intrerup totdeauna programul



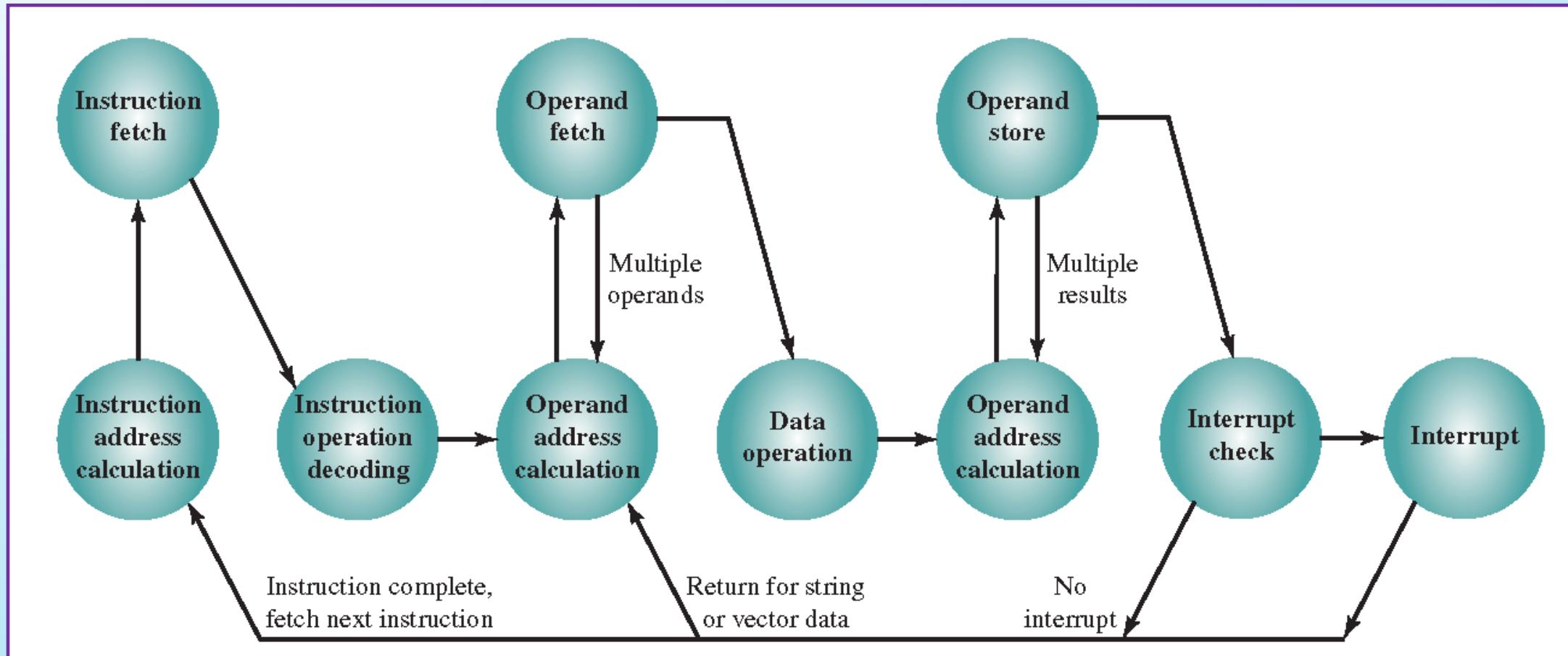


CICLUL INSTRUCȚIUNE CU INTRERUPERE

Daca semnalul de intrerupere este prezent:

- Programul curent este intrerupt
- Contextul este salvat (este salvata adresa instructiunii urmatoare, adica continutul curent al contorului de program, precum si alte date relevante)
- Este setat contorul de program la adresa de start a rutinei de intrerupere





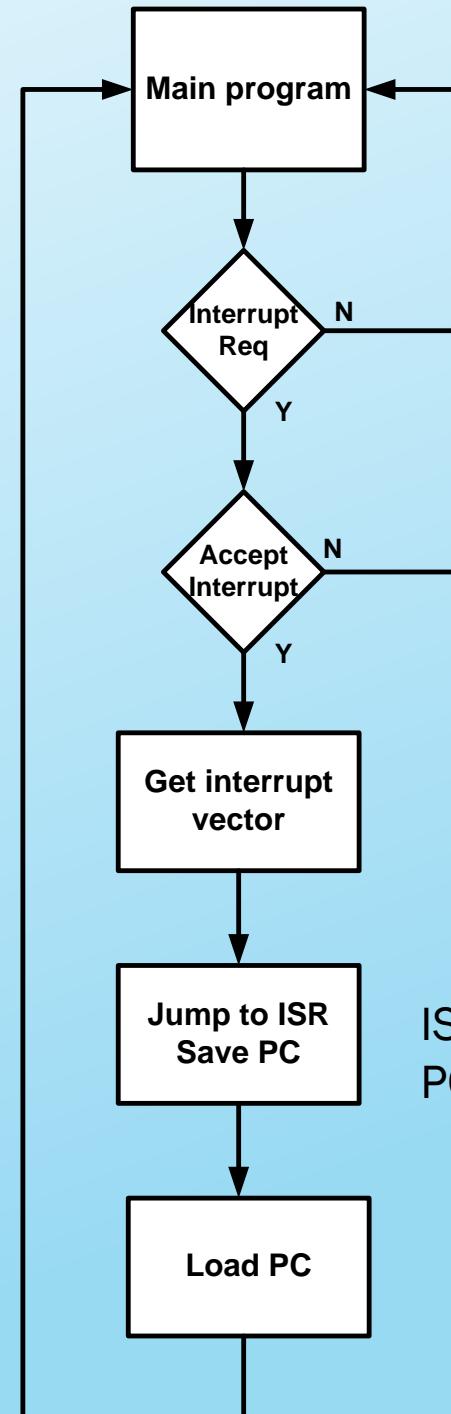
Ciclu instrucțiune cu intreruperi



Prezenta semnalului de intrerupere este verificata periodic.
Verificarea se face intre executiile a doua instructiuni successive.

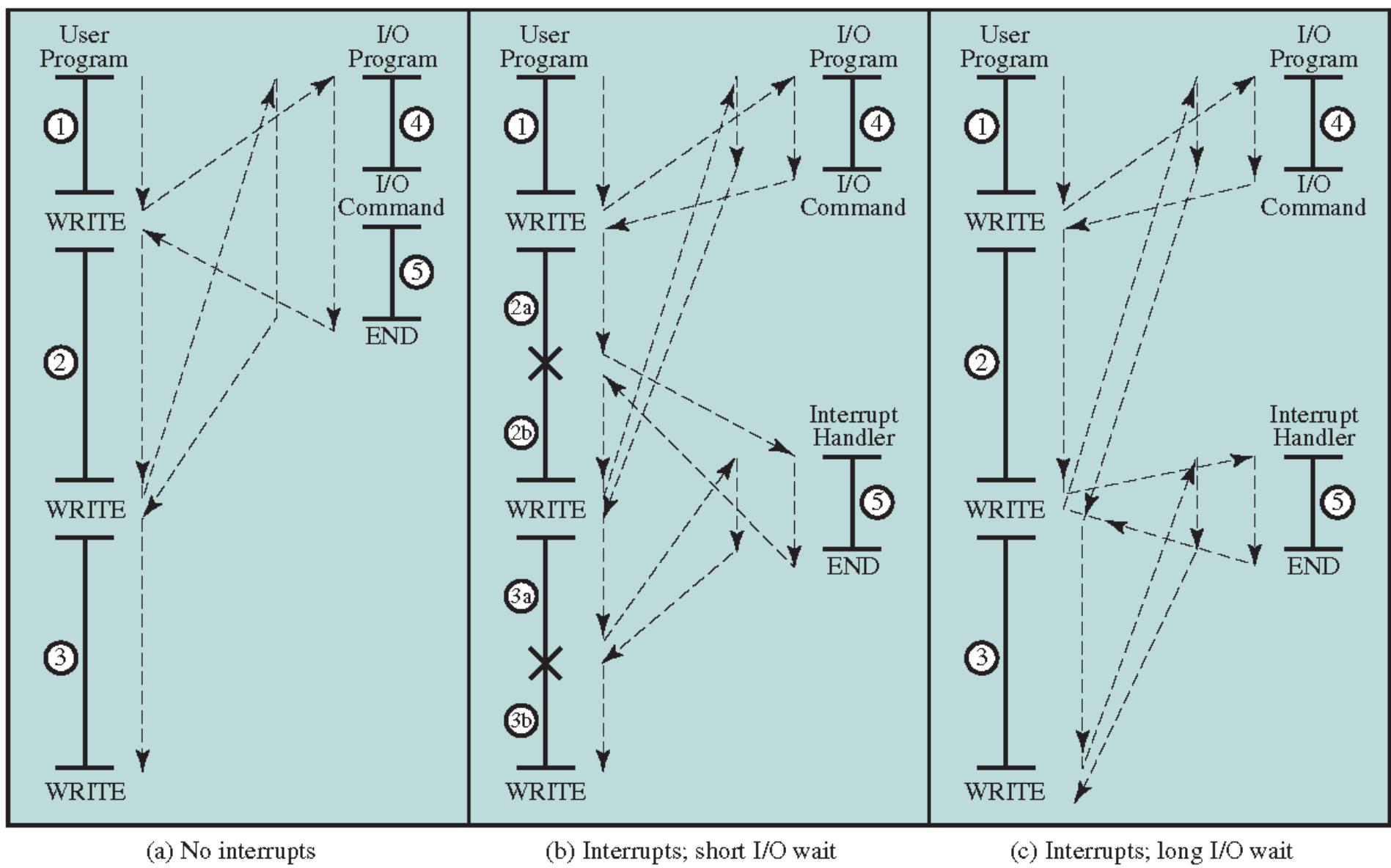
In timpul executiei unei instructiuni **nu se verifica** intreruperea.
Daca semnalul de intrerupere apare in timpul executiei unei
instructiuni, aceasta este executata pana la capat si apoi este
detectata intreruperea.

vector de intrerupere (interrupt vector):
este adresa de inceput a rutinei de
tratare a intreruperii (ISR)



ISR: **Interrupt Service Routine**
PC: **Program Counter**





Execuția unui program fără sau cu intreruperi



USER PROGRAM

$\langle \text{statement} \rangle$
 $\langle \text{statement} \rangle$
⋮
 $\langle \text{statement} \rangle$

} Code segment 1

WRITE

$\langle \text{statement} \rangle$
 $\langle \text{statement} \rangle$
⋮
 $\langle \text{statement} \rangle$

} Code segment 2

WRITE

$\langle \text{statement} \rangle$
 $\langle \text{statement} \rangle$
⋮
 $\langle \text{statement} \rangle$

} Code segment 3

I/O PROGRAM

$\langle \text{statement} \rangle$
 $\langle \text{statement} \rangle$
⋮
 $\langle \text{statement} \rangle$

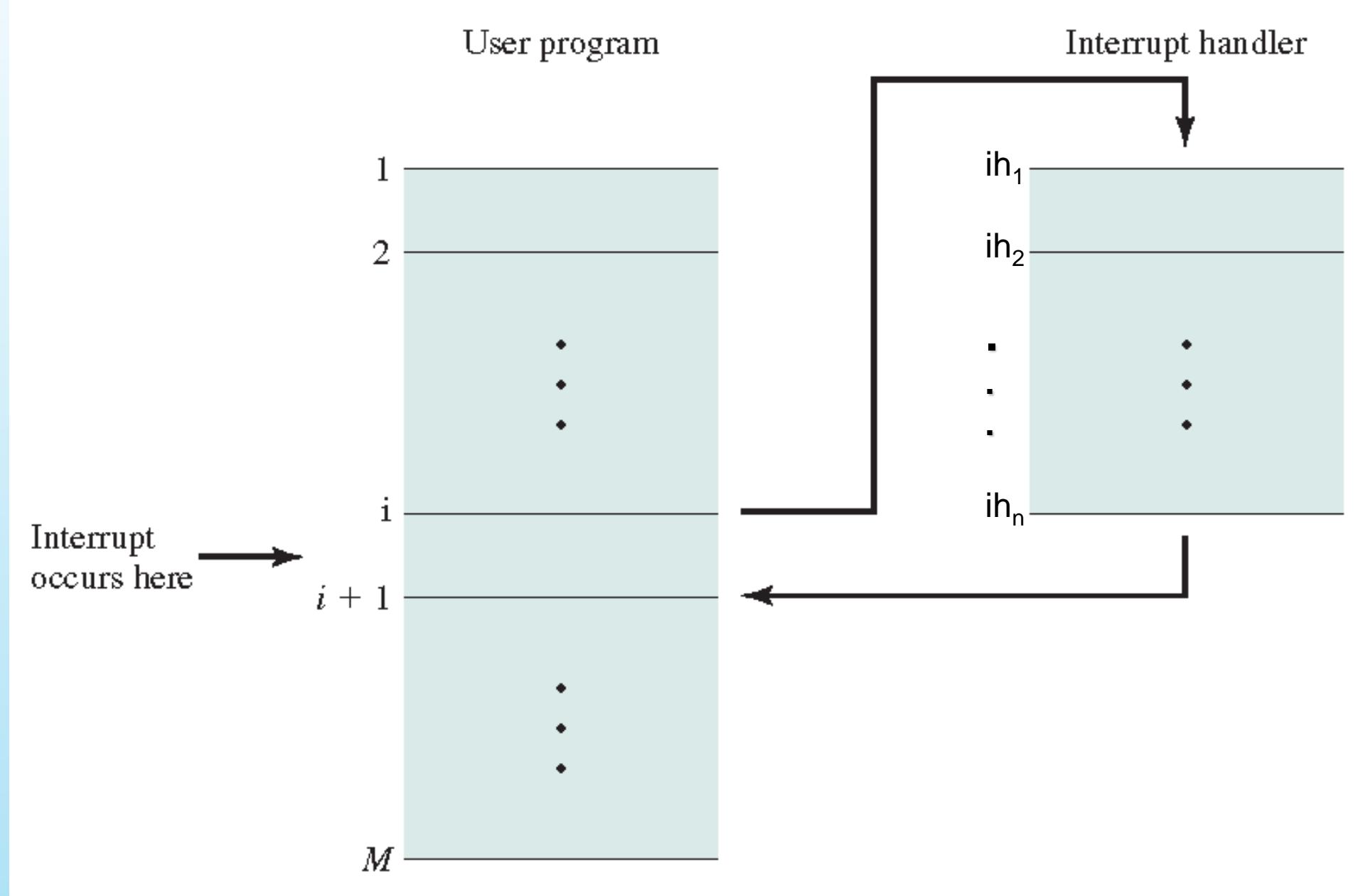
} Code segment 4

I/O command

$\langle \text{statement} \rangle$
 $\langle \text{statement} \rangle$
⋮
 $\langle \text{statement} \rangle$

} Code segment 5

- Segment de program



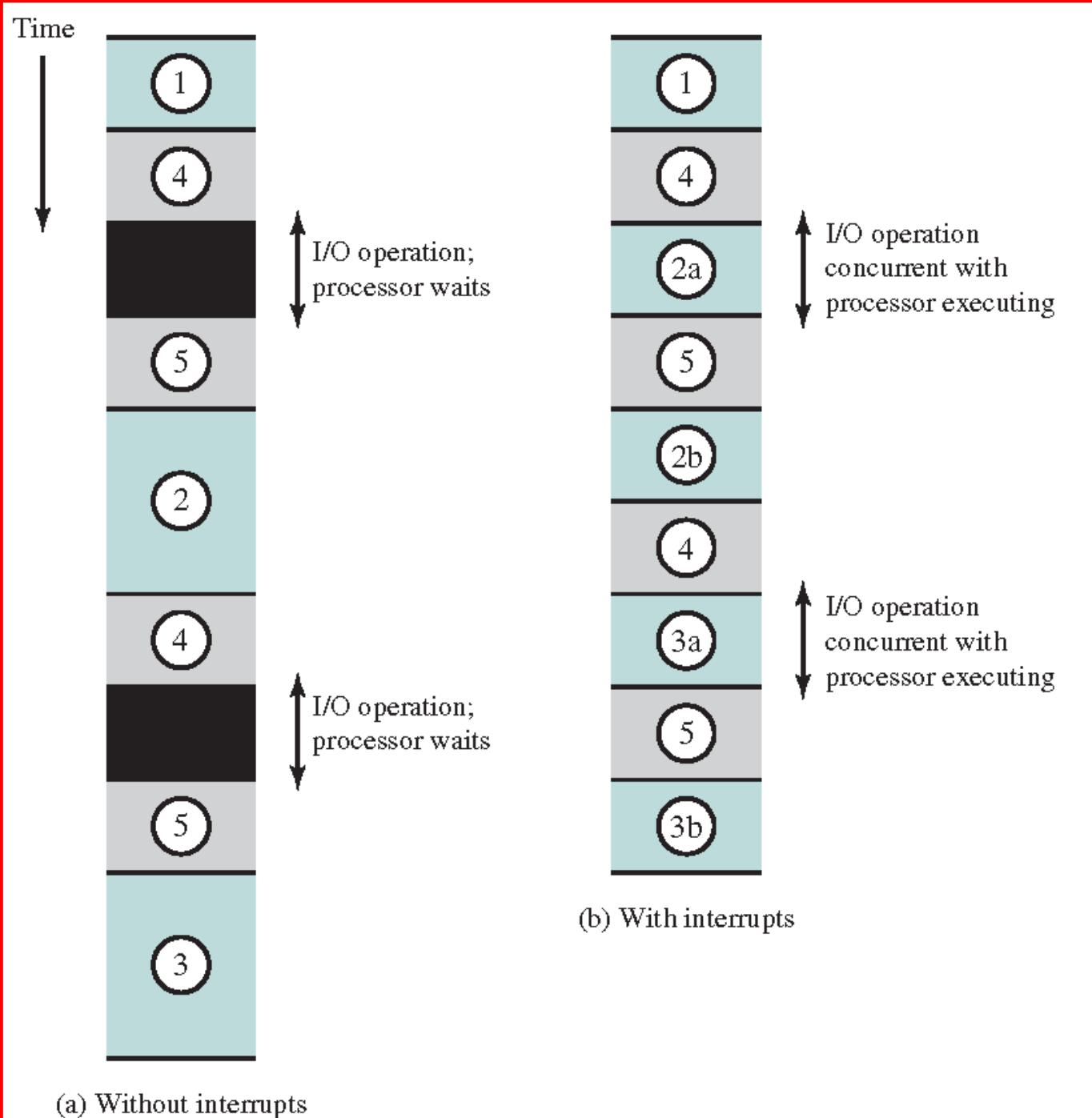
- Transferul controlului *via intrerupere*



- Rutina de întrerupere este, în general, **parte** a sistemului de operare.
- Acest program determină natura întreruperii și efectuează acțiunile necesare.
 - Instrucțiuni suplimentare trebuie executate (în rutina de întrerupere) pentru a determina natura întreruperii și pentru a decide asupra acțiunii corespunzătoare.
- Când rutina de întrerupere este finalizată, procesorul poate relua executarea programului utilizator din punctul de întrerupere.
- Datorită timpului relativ mare care ar fi pierdut prin simpla așteptare a unei operații de I / O, procesorul poate fi folosit mult mai eficient prin utilizarea întreruperilor.

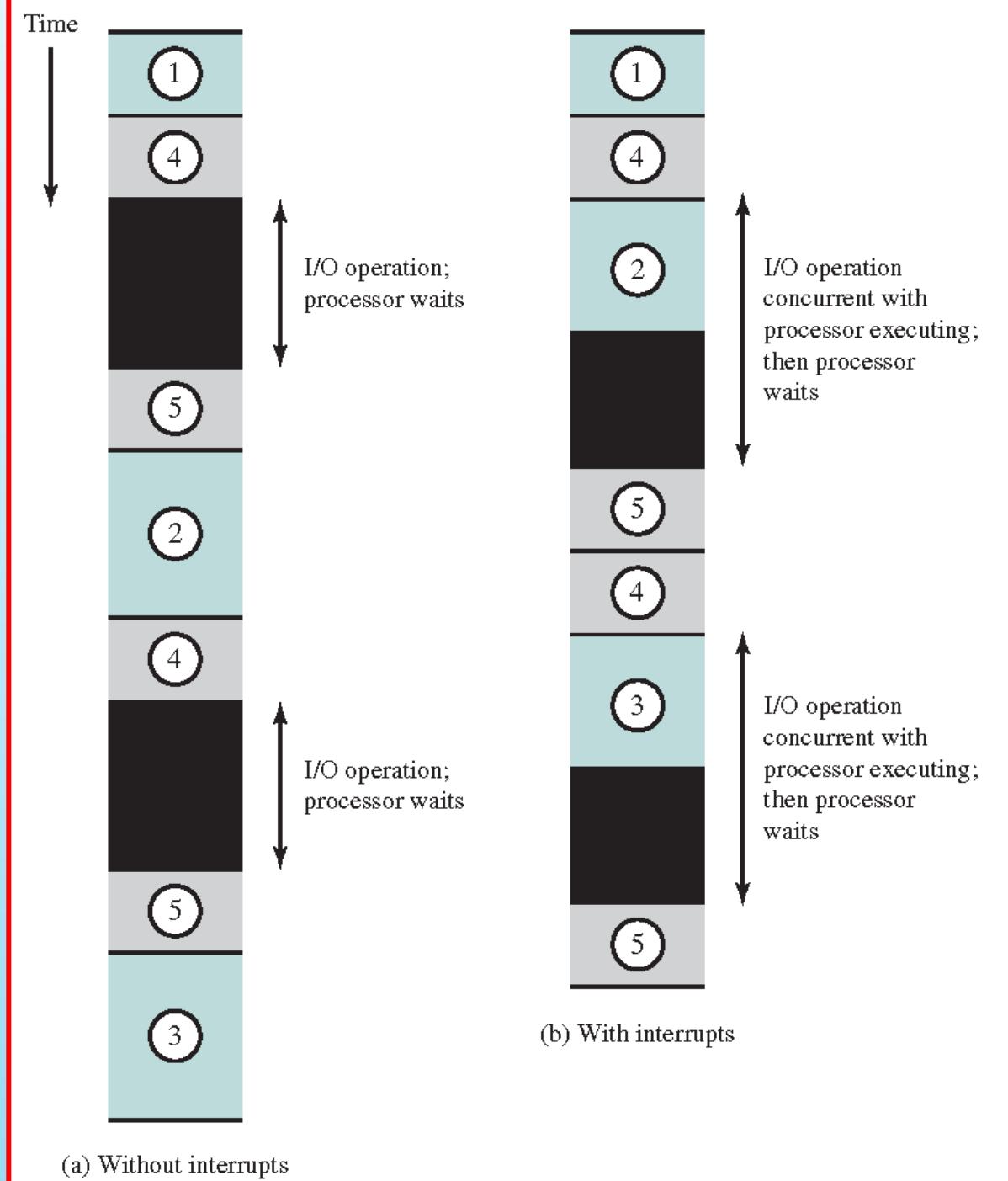


Bugetul de timp,
cu și fără intrerupere:
cazul așteptării scurte I/O





Budgetul de timp,
cu și fără intrerupere:
cazul așteptării lungi I/O



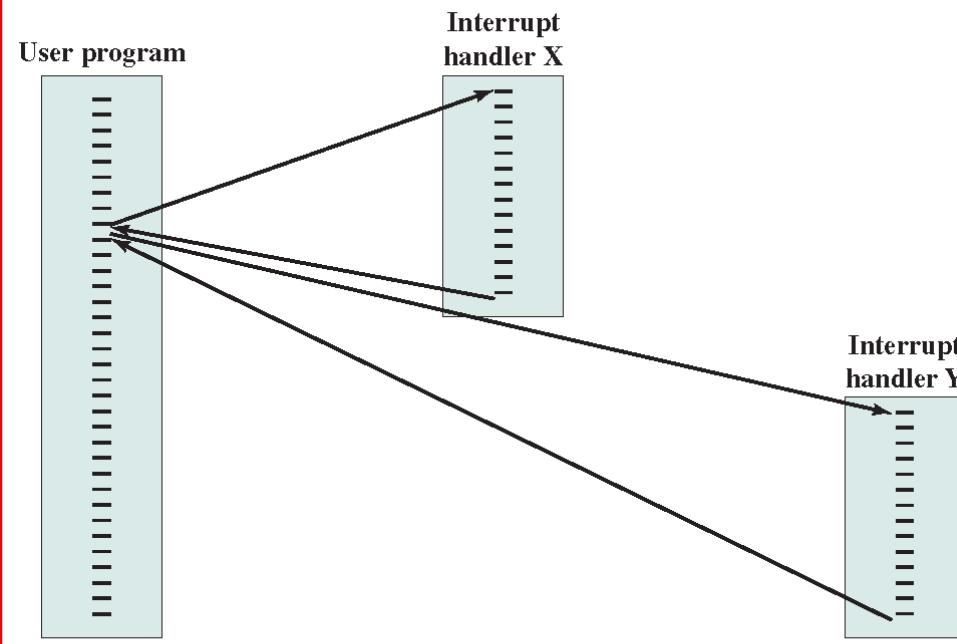
Prezență intreruperilor multiple

Tratament secvential

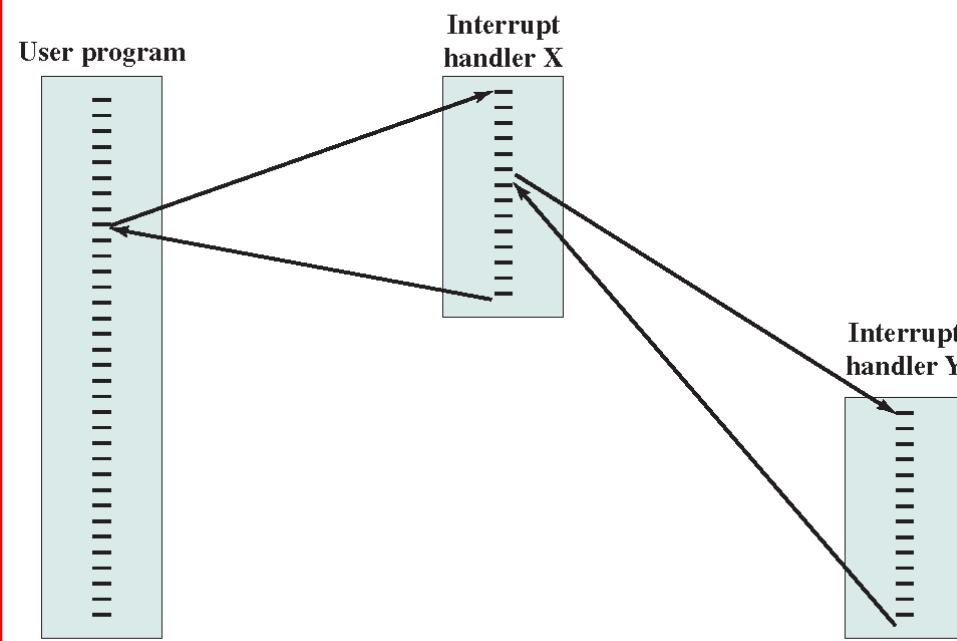
- În timpul în care o intrerupere este tratată celelalte intreruperi sunt dezactivate (CPU ignora semnalele de intrerupere)
- Urmează tratarea în ordine a următoarelor intreruperi

Tratament imbricat

- Sunt definite priorități pentru intreruperi
- Este tratată intreruperea cu prioritatea cea mai ridicată
- Este permisă blocarea unei secvențe de tratare a unei intreruperi de prioritate mică

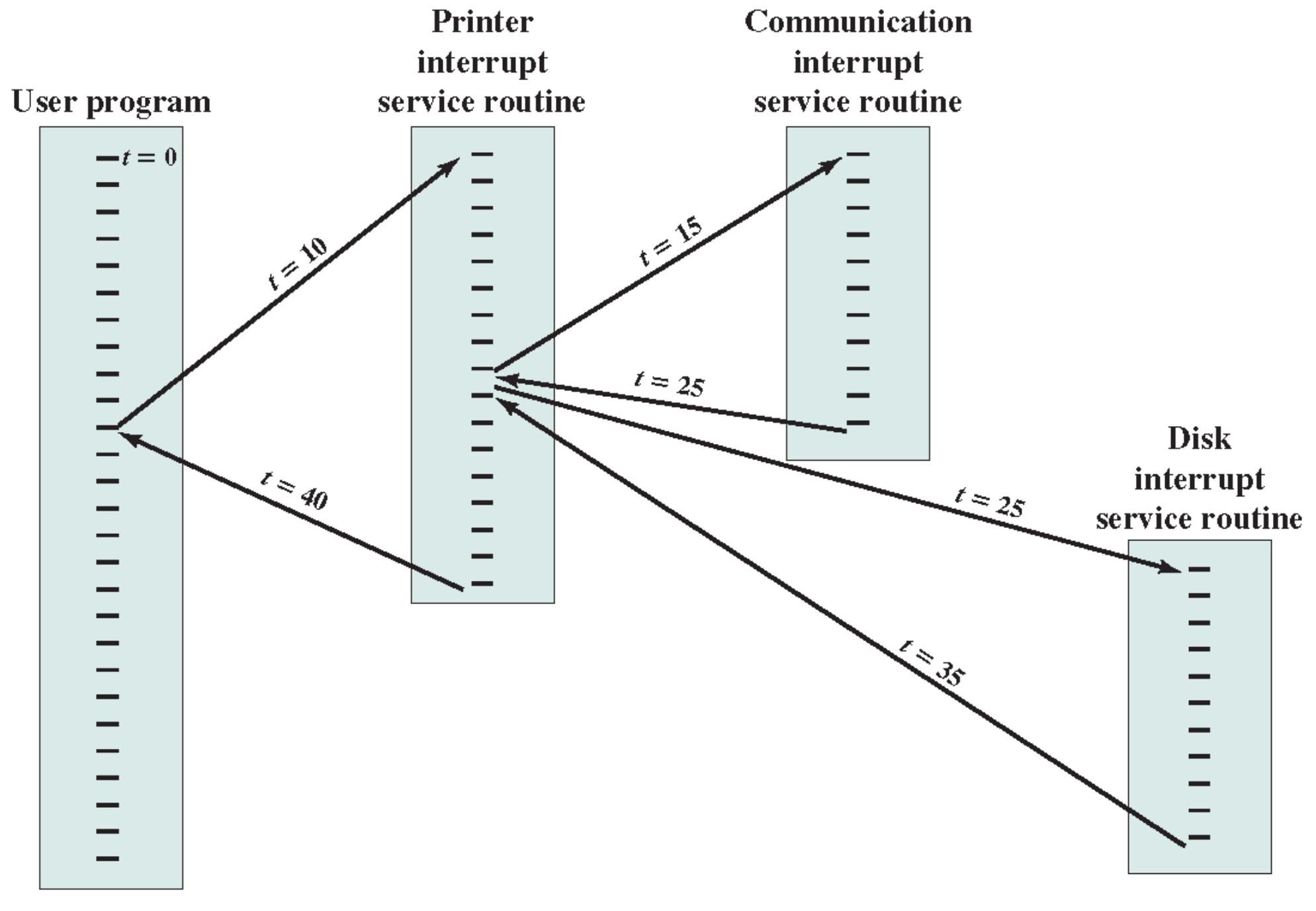


(a) Sequential interrupt processing



(b) Nested interrupt processing



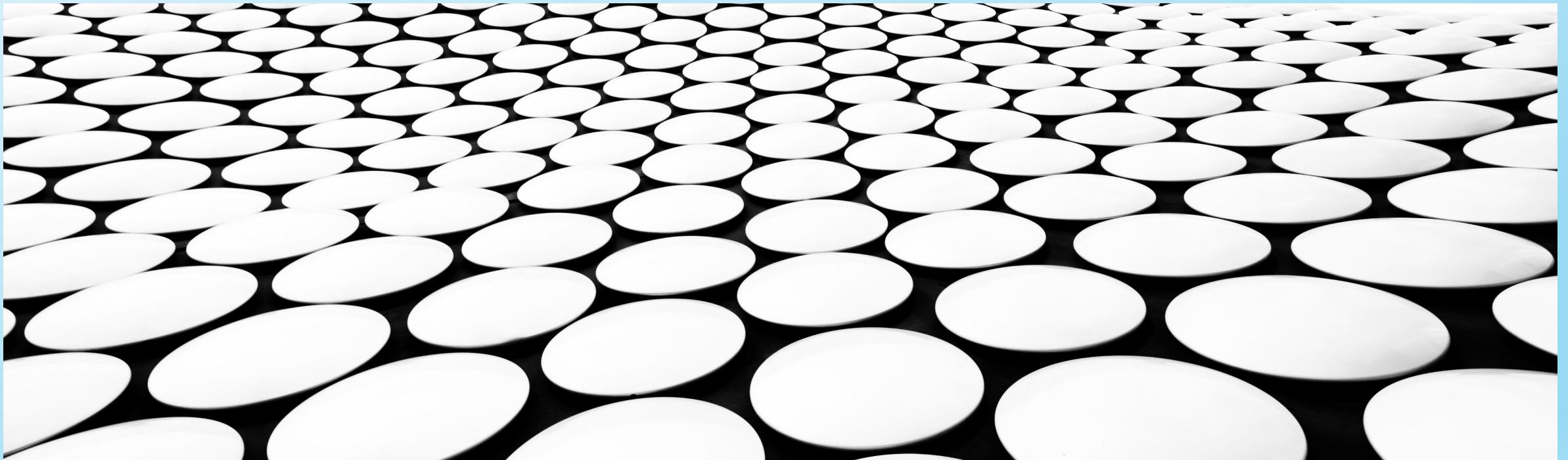


Prioritatile Intreruperilor

Interrupt type	Priority level
Internal	Highest
Non-maskable	Very high
Software	High
External (32 – 255)	low

ARHITECTURA SISTEMELOR DE CALCUL

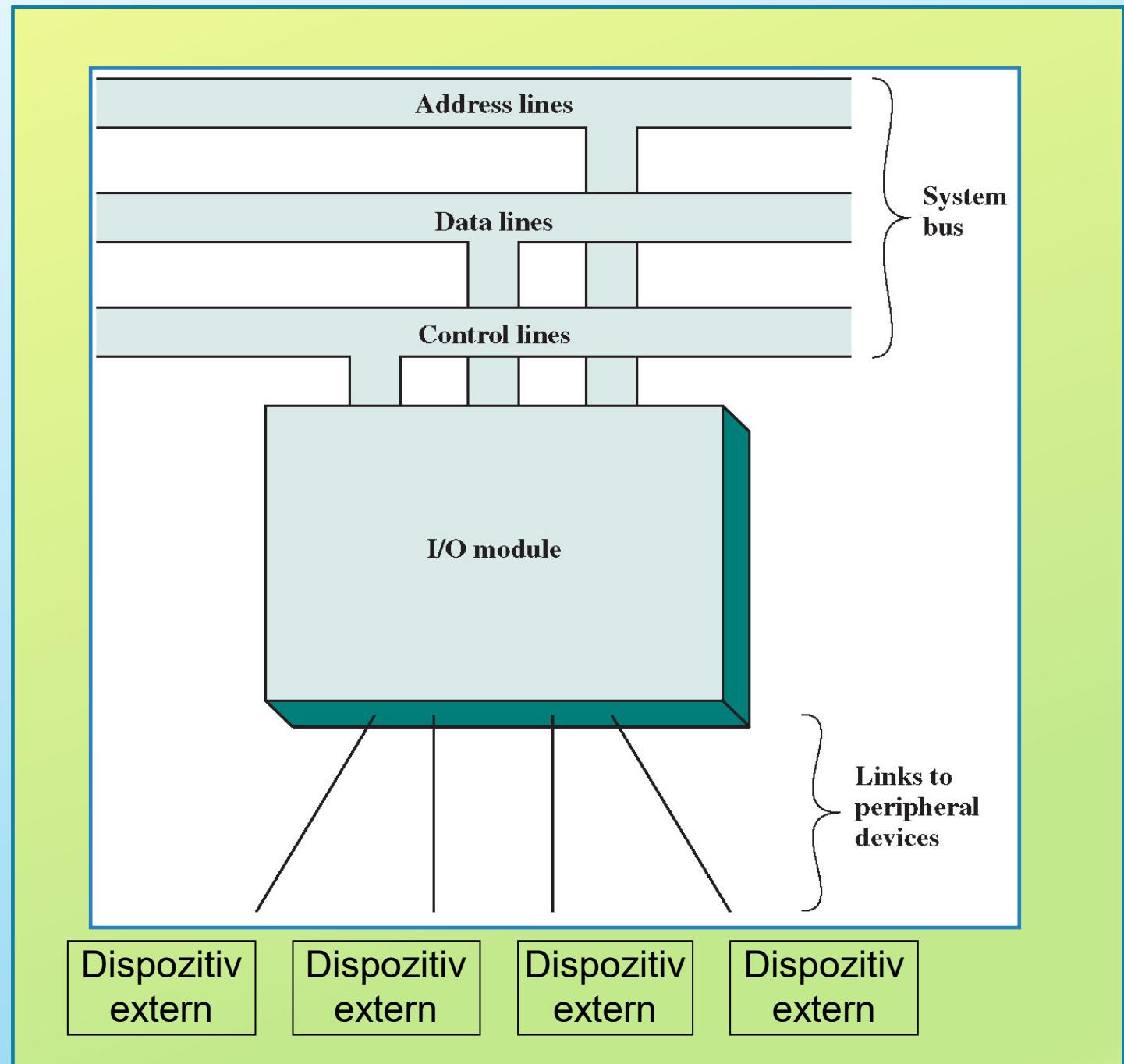
UB, FMI, CTI, ANUL III, 2022-2023



**BLOCURI
DE
INTRARE/IESIRE
(I/O)**

Modelul generic al unui bloc I/O

= modul I/O +
dispozitiv extern



Dispozitive externe

- Operațiile I/O sunt realizate printr-o gamă largă de dispozitive externe care asigura schimbul de date între mediul extern și computer.
- Un dispozitiv extern se atașează la computer printr-o legătură cu un modul I/O.
 - Legătura este utilizată pentru a transfera controlul, starea și datele între modulul **I/O** și dispozitivul extern.
 - Un dispozitiv extern conectat la un modul **I/O** este adesea denumit dispozitiv periferic sau pur și simplu periferic.

În general, putem clasifica **dispozitivele** externe în trei categorii:

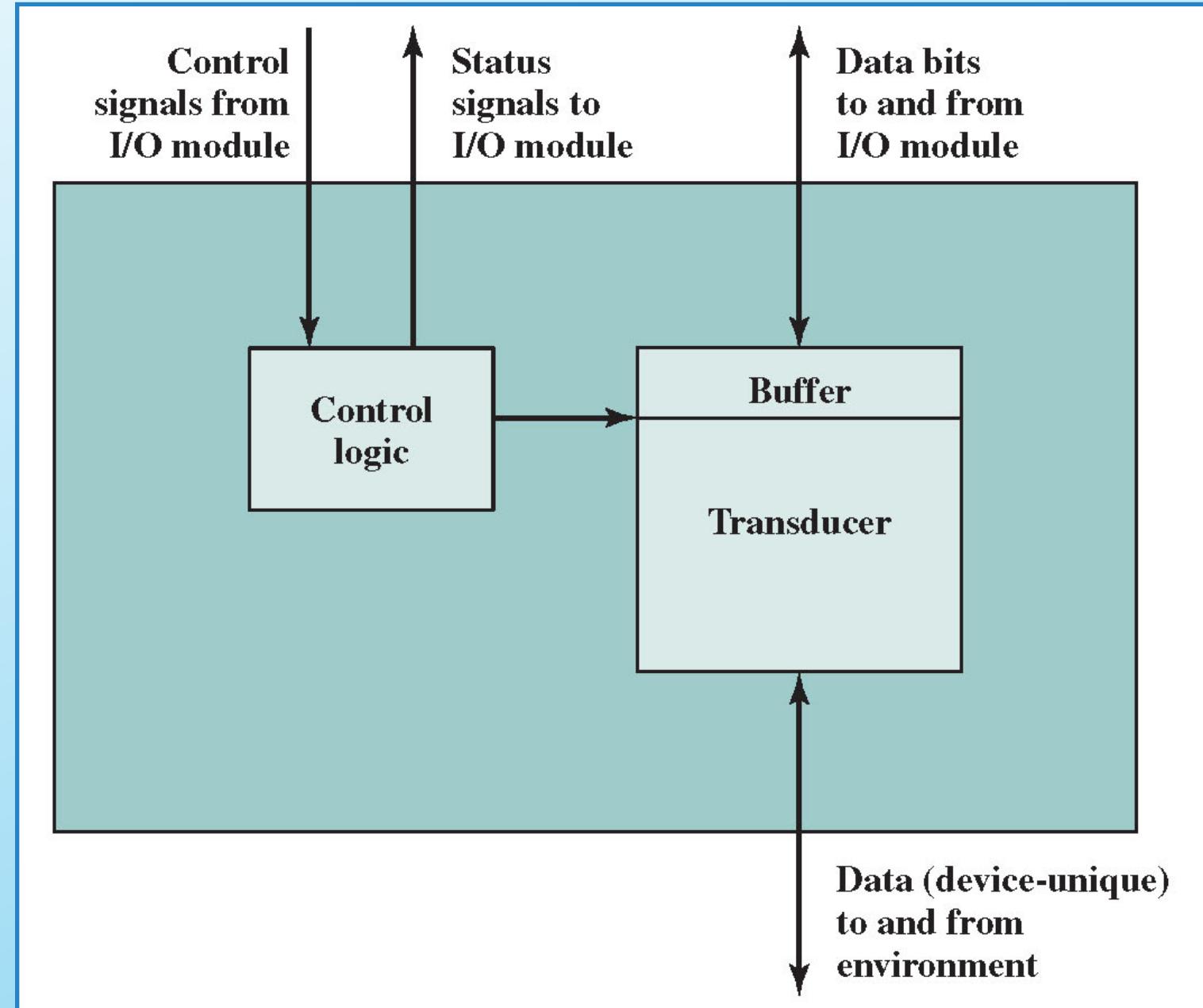
- **dispozitive care pot fi citite de om:** Adequate pentru comunicarea între utilizator si calculator
 - Video-terminal, imprimante
- **dispozitive care pot fi citite de masina:** specializate în comunicarea între echipamente si calculator
 - Discuri magnetice, senzori
- **dispozitive pentru comunicare:** Potrivite pentru comunicarea cu dispozitivele aflate la distanță

Structura unui dispozitiv extern

Blocul de control logic

Memorie tampon

Traductor



Interfața cu modulul I/O transmite semnale de control, stare și date.

- **Semnalele de control**

- determină operația pe care dispozitivul o va efectua:
 - trimitera datelor către modulul I/O (INPUT sau READ),
 - acceptarea datelor din modulul I/O (OUTPUT sau WRITE),
- indică starea rapoartelor
- comanda efectuarea unei anumite funcții de control la dispozitiv (de exemplu, poziționarea unui cap pe disc).

- **Semnalele de stare** indică starea dispozitivului.

- READY / NOT-READY pentru a arăta dacă dispozitivul este pregătit pentru transferul de date.
- **Datele** sunt grupate sub forma unor seturi de biți care trebuie trimise sau primite de la modulul I/O.

- **Blocul de control logic** controlează funcționarea dispozitivului ca răspuns la directiva primită de la modulul I/O.
- **Traductorul** convertește datele:
 - de la cele cu semnal electric (digital) la celelalte forme de semnal în timpul ieșirii și
 - de la alte forme de semnal la cele cu semnal electric (digital) în timpul intrării.
- **Memoria tampon** este asociată cu traductorul pentru a ține temporar datele transferate între modulul I/O și mediul extern; o dimensiune de buffer de 8 până la 16 biți este uzuala.

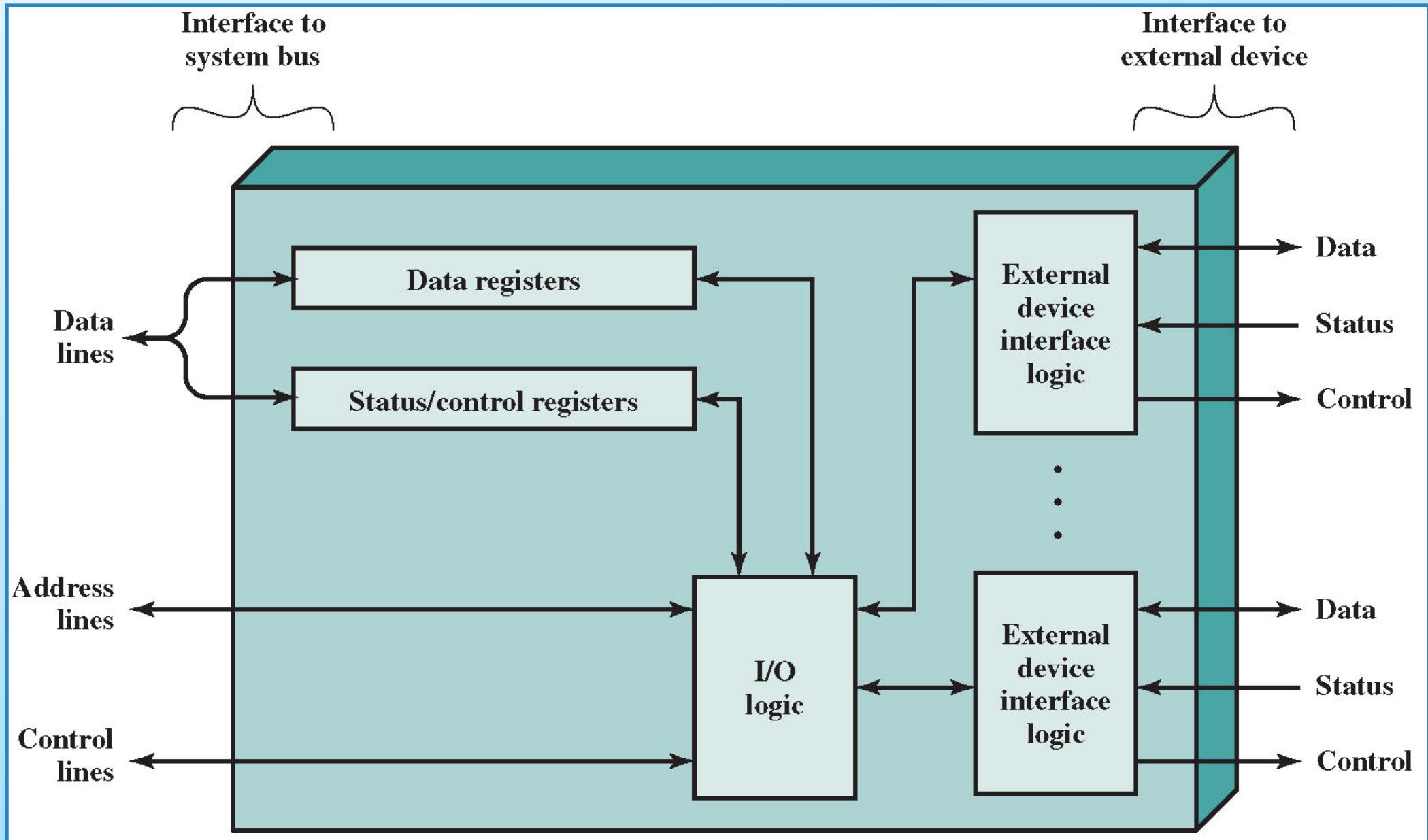
Module I/O

- Funcțiile modulului
 - Funcțiile sau cerințele majore pentru un **modul I/O** se încadrează în următoarele categorii:
 - Control și secvențiere
 - Comunicarea cu procesorul
 - Comunicarea cu dispozitivul periferic
 - Stocarea datelor în memorii tampon
 - Detectarea erorilor

Structura modulului I/O



Ex.:



- În orice moment de timp, procesorul poate (trebuie) să comunice cu unul sau mai multe dispozitive externe, în mod imprevizibil, dar în funcție de comenzi de I/O din program.
- Resursele interne, cum ar fi memoria principală și magistrala de sistem, trebuie partajate între mai multe activități, inclusiv transferul de date I/O.
- Astfel, *funcția I/O include o cerință de control și de sincronizare*, pentru a coordona fluxul de date între componente interne și dispozitivele externe.

Transferul de date **de la** un dispozitiv extern către procesor implica următoarea secvență de pași:

- 1.** Procesorul interoghează modulul I/O pentru a verifica starea dispozitivului atașat.
- 2.** Modulul I/O returnează starea dispozitivului extern.
- 3.** Dacă dispozitivul este operațional și este gata să transmită, procesorul solicită transferul de date prin intermediul unei comenzi către modulul I/O.
- 4.** Modulul I/O obține un calup de date (de exemplu, 8 sau 16 biți) de la dispozitivul extern.
- 5.** Datele sunt transferate de la modulul I/O la procesor.

Dacă sistemul utilizează o magistrală, atunci fiecare dintre interacțiunile dintre procesor și modulul I/O implică una sau mai multe arbitraje de magistrală.

Scenariul simplificat precedent ilustrează de asemenea faptul că modulul I/O trebuie să comunice **cu procesorul** și **cu dispozitivul extern**.

Comunicarea cu procesorul

Comenzi: Modulul I/O acceptă comenzi de la procesor, ca semnale pe magistrala de comandă.

- De exemplu, un modul I/O pentru o unitate de disc ar putea accepta următoarele comenzi: 'READ SECTOR', 'WRITE SECTOR', 'SEEK track number' și "CAN record ID". Ultimele două comenzi includ fiecare un parametru care este trimis pe magistrala de date.

Date: Datele sunt schimbate între procesor și modulul I/O pe magistrala de date.

Raportarea stării: deoarece perifericele sunt foarte lente, este important să se cunoască starea modulului I/O (respectiv a perifericului).

- De exemplu, dacă un modul de intrare / ieșire este rugat să trimită date procesorului (citire), este posibil să nu fie gata să facă acest lucru deoarece lucrează încă la o comandă I/O anterioară. Acest fapt poate fi raportat cu un semnal de stare.

Semnalele generale de stare sunt 'BUSY' și 'READY'. Pot să existe și semnale de raportare pentru diferite erori.

Recunoașterea adresei: Un modul I/O trebuie să recunoască cate o adresă unică pentru fiecare periferic pe care il controleaza.

Comunicarea cu dispozitivul extern

Această comunicare implică transfer de comenzi, informații despre stare și date

O sarcină esențială a unui modul I/O este **stocarea datelor în memorii tampon**. În timp ce rata de transfer catre sau dinspre memoria principală sau procesor este destul de mare, pentru multe dispozitive periferice rata este cu ordine de mărime mai mică și acoperă un domeniu larg.

- Datele provenite din memoria principală sunt trimise catre un modul I/O într-un ritm exploziv. Datele sunt stocate temporar în modulul I/O și apoi trimise la dispozitivul periferic într-o rata de transfer specifică.
- În direcția opusă, datele sunt stocate temporar astfel încât să nu oblige memoria la o operație de transfer lent. Astfel, modulul I/O trebuie să poată funcționa atât la viteze ale dispozitivului, cât și la memorie.
- În mod similar, dacă dispozitivul I/O funcționează cu o rată mai mare decât rata de acces la memorie, atunci modulul I/O efectuează operația de stocare temporara necesară.

- Un modul I/O este adesea responsabil pentru **detectarea erorilor** și pentru **răportarea lor** catre procesor.
 - O clasă de erori include **defecțiuni mecanice și electrice** raportate de dispozitiv (de exemplu, blocaj de hârtie, disc DVD defect).
 - O altă clasă constă în **modificări neintenționate ale informației**, în timpul transmisiei de la dispozitiv la modulul I/O. Unele forme de codificare permit detectarea erorilor de transmisie.
 - Un exemplu simplu este utilizarea unui bit de paritate pe fiecare caracter transmis. De exemplu, codul de caractere IRA ocupă 7 biți dintr-un octet. Al 8-lea bit este setat astfel încât suma totală a bitilor din byte să fie par (paritate pară) sau impară (paritate impară). Când un octet este primit, modulul I/O verifică paritatea pentru a determina dacă a apărut o eroare.

- Modulele I/O variază considerabil în ceea ce privește complexitatea și numărul de dispozitive externe pe care le controlează.
 - Modulul se conectează la restul calculatorului printr-un set de linii de semnal (de exemplu, linii de magistrala de sistem).
 - Logica din cadrul modulului interacționează cu procesorul printr-un set de linii de control. Procesorul utilizează liniile de control pentru a emite comenzi către modulul I/O.
 - Unele dintre liniile de control pot fi utilizate de către modulul I/O (de exemplu, pentru semnale de arbitraj și stare).
 - Datele transferate către și din modul sunt stocate în unul sau mai mulți registri de date.
 - Pot exista, de asemenea, unul sau mai multe registri de stare care furnizează informații despre starea curentă.
 - Un registru de stare poate funcționa, de asemenea, ca registru de control, pentru a accepta informații detaliate de control de la procesor.
 - De asemenea, modulul trebuie să poată recunoaște și genera **adrese asociate** cu dispozitivele pe care le controlează.
 - **Fiecare modul I/O are o adresă unică sau, dacă controlează mai multe dispozitive externe, un set unic de adrese.**
 - Modulul I/O conține logică specifică interfeței cu fiecare dispozitiv pe care îl controlează.

Un modul I/O funcționează permitând procesorului să vizualizeze o gamă largă de dispozitive **într-un mod simplu**.

- Modulul I/O poate ascunde detaliile de sincronizare, formate și de electromecanică ale unui dispozitiv extern astfel încât procesorul să poată funcționa în termeni simpli de comenzi de citire și scriere și, eventual comenzi de deschidere și închidere a fișierelor.
 - În forma sa cea mai simplă, modulul I/O poate lăsa încă o mare parte din munca de control (de exemplu, derularea inversă a benzii) vizibilă procesorului.
- Un modul I/O avansat, care preia cea mai mare parte a operațiilor detaliate de procesare, prezintând o interfață de nivel înalt procesorului, este denumit de obicei un **canal I/O** sau un **procesor I/O**.
- Un modul I/O care este destul de primitiv și necesită un control detaliat este denumit de obicei un **controler I/O** sau un **controler de dispozitiv**.
 - Controlerele I/O sunt frecvent observate pe microcomputere, în timp ce canalele I/O sunt utilizate pe mainframe.

■ Sunt posibile trei tehnici de operare I/O.

- **operarea I/O programata:** datele sunt schimbată între procesor și modulul I/O. Procesorul execută un program care îi oferă controlul direct al operației I/O, incluzând sesizarea stării dispozitivului, trimiterea unei comenzi de citire sau scriere și transferarea datelor.
 - Când procesorul emite o comandă către modulul I/O, trebuie să aștepte până la terminarea operației I/O.
Dacă procesorul este mai rapid decât modulul I/O, apare o pierdere de timp procesor.
- **operarea I/O cu întreruperi:** procesorul emite o comandă I/O, continuă să execute alte instrucțiuni și este **întrerupt** de modulul I/O când acesta din urmă și-a încheiat activitatea.
 - În operarea **I/O programata și cu întreruperi**, procesorul este responsabil:
 - cu extragerea datelor din memoria principală pentru ieșire și
 - cu stocarea datelor în memoria principală pentru intrare.
 - A treia alternativa de operare este cunoscută sub denumirea de **acces direct la memorie (DMA)**. În acest mod, modulul I/O și memoria principală schimbă datele direct, fără implicarea procesorului.

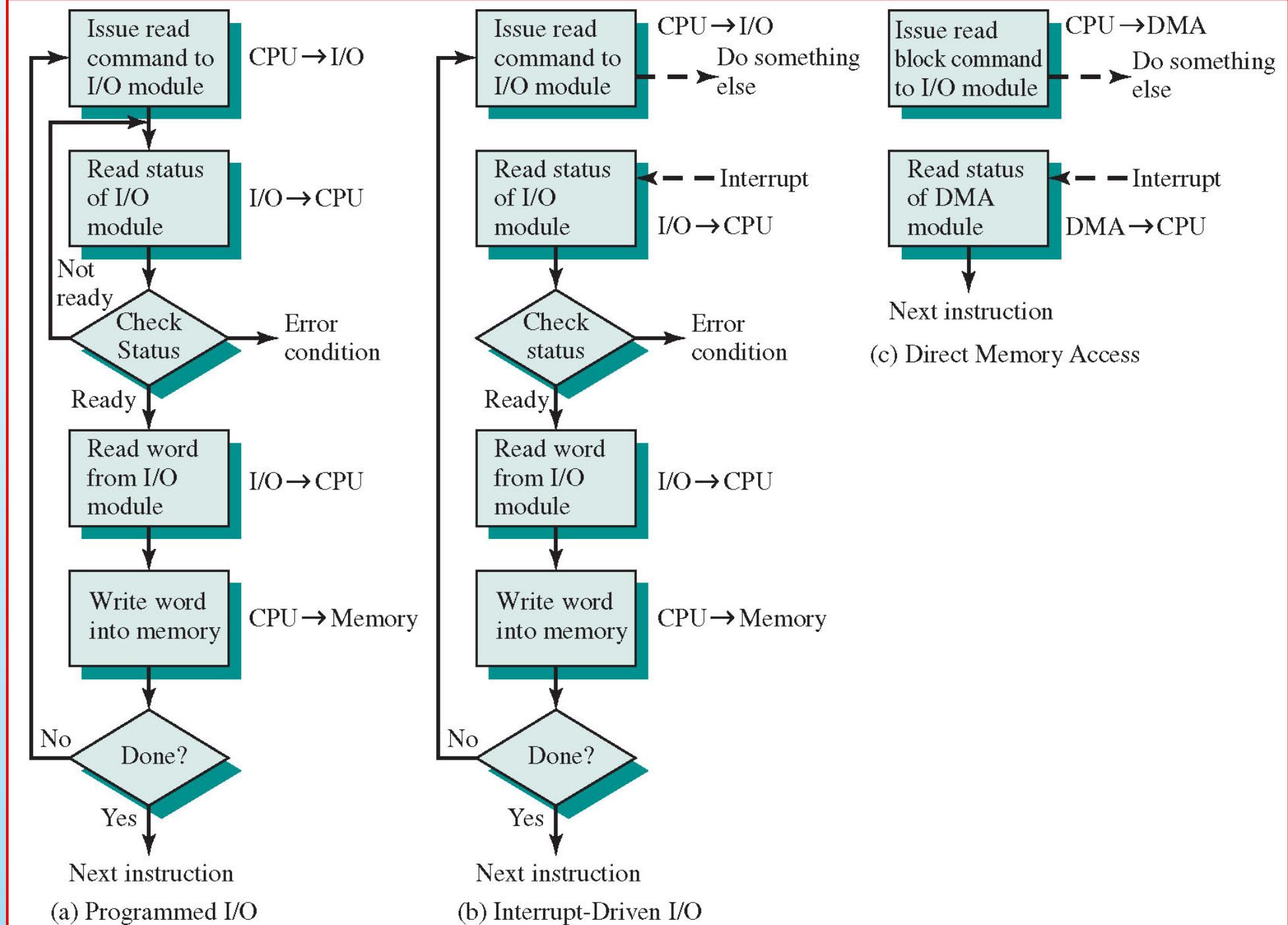
Operatiuni I/O programate

- Când procesorul execută un program și întâlnește o instrucțiune referitoare la I/O, execută acea instrucțiune prin emiterea unei comenzi către modulul I/O corespunzător.
 - Cu I/O programate, modulul I/O va efectua acțiunea solicitată și apoi va seta biții corespunzători în registrul de stare I/O.
 - Modulul I/O nu mai întreprinde acțiuni de alertare a procesorului. În special, **nu îintrerupe procesorul**.
 - Astfel, este responsabilitatea procesorului să verifice periodic starea modulului de intrare / ieșire până când constată că operația este completă.

Comenzi I/O

- Pentru a executa o instrucțiune referitoare la I/O, procesorul emite
 - o adresă, specificând modulul I/O și dispozitivul extern și
 - o comandă I/O.
- Există patru **tipuri** de comenzi I/O pe care un modul I/O le poate primi atunci când este adresat de un procesor:
 - **de Control:** Se utilizează pentru a activa un dispozitiv periferic și pentru a-i comunica ce să facă.
 - De exemplu, o unitate de bandă magnetică poate fi instruită pentru a derula înapoi sau în avans avans o banda. Aceste comenzi sunt adaptate tipului particular de dispozitiv periferic.
 - **de Test:** Folosit pentru a testa diferitele condiții de stare asociate cu un modul I/O și perifericele acestuia.
 - Procesorul va dori să știe că **perifericul** de interes este alimentat și **disponibil** pentru utilizare.
 - De asemenea, va dori să știe **dacă operațiunea I/O** cea mai recentă **este finalizată** și dacă au apărut erori.
 - **de Citire:** determină modulul I/O să obțină un calup de date de la periferic și să îl plaseze într-un buffer intern.
 - Procesorul poate obține apoi calupul de date solicitând ca modulul I/O să îl plaseze pe magistrala de date.
 - **de Scriere:** Obliga modulul I/O să preia un calup de date (octet sau cuvânt) din magistrala de date și apoi să transmită acel element de date către periferic.

Figura 1



- Figura 1a prezintă un exemplu de utilizare a I/O programate pentru citirea într-un bloc de date de la un dispozitiv periferic (de exemplu, o înregistrare din bandă) cu scriere în memorie.
 - Datele sunt citite sub forma unui singur cuvânt (ex.: 16 biți) la un moment dat.
 - Pentru fiecare cuvânt citit, procesorul trebuie să rămână într-un ciclu de verificare a statutului până când determină că cuvântul este disponibil în registrul de date al modulului I/O.
 - Această diagramă evidențiază principalul dezavantaj al acestei tehnici: este un proces consumator de timp care menține procesorul ocupat inutil.

Instructiuni I/O

- În cazul I/O programate, există o corespondență strânsă între instrucțiunile I/O pe care procesorul le extrage din memorie și comenzi I/O pe care procesorul le emite către un modul I/O pentru a executa instrucțiunile.
 - Instrucțiunile sunt ușor de tradus în comenzi I/O, și există adesea o relație unu-la-unu simplă. Forma instrucțiunii depinde de modul în care sunt abordate dispozitivele externe.
- Există multe dispozitive I/O conectate prin intermediul modulelor I/O la sistem.
 - Fiecare dispozitiv are un identificator sau o adresă unică. Când procesorul emite o comandă I/O, comanda conține adresa dispozitivului dorit.
 - Fiecare modul I/O trebuie să interpreteze liniile de adresă pentru a determina dacă comanda este pentru el însăși.

- Când procesorul, memoria principală și I/O partajeaza acceasi magistrala, sunt posibile două moduri de adresare:
 - **memorie mapată**
 - **memorie izolată.**
- Pentru **I/O cu memorie mapata**, există un singur spațiu de adrese pentru locațiile de memorie și dispozitivele I/O.
 - Procesorul tratează registrele de stare și date ale modulelor I/O ca locații de memorie și utilizează aceleași instrucțiuni pentru a accesa atât dispozitivele de memorie, cât și dispozitivele I/O.
 - De exemplu, o magistrala cu 10 linii de adrese (10 biti), poate adresa un total (combinat) de $2^{10} = 1024$ de locații de memorie și adrese I/O, în orice combinație.
 - Pentru I/O cu memorie mapata, sunt necesare: o singură linie de citire și o singură linie de scriere pe magistrala

I/O cu memorie izolata

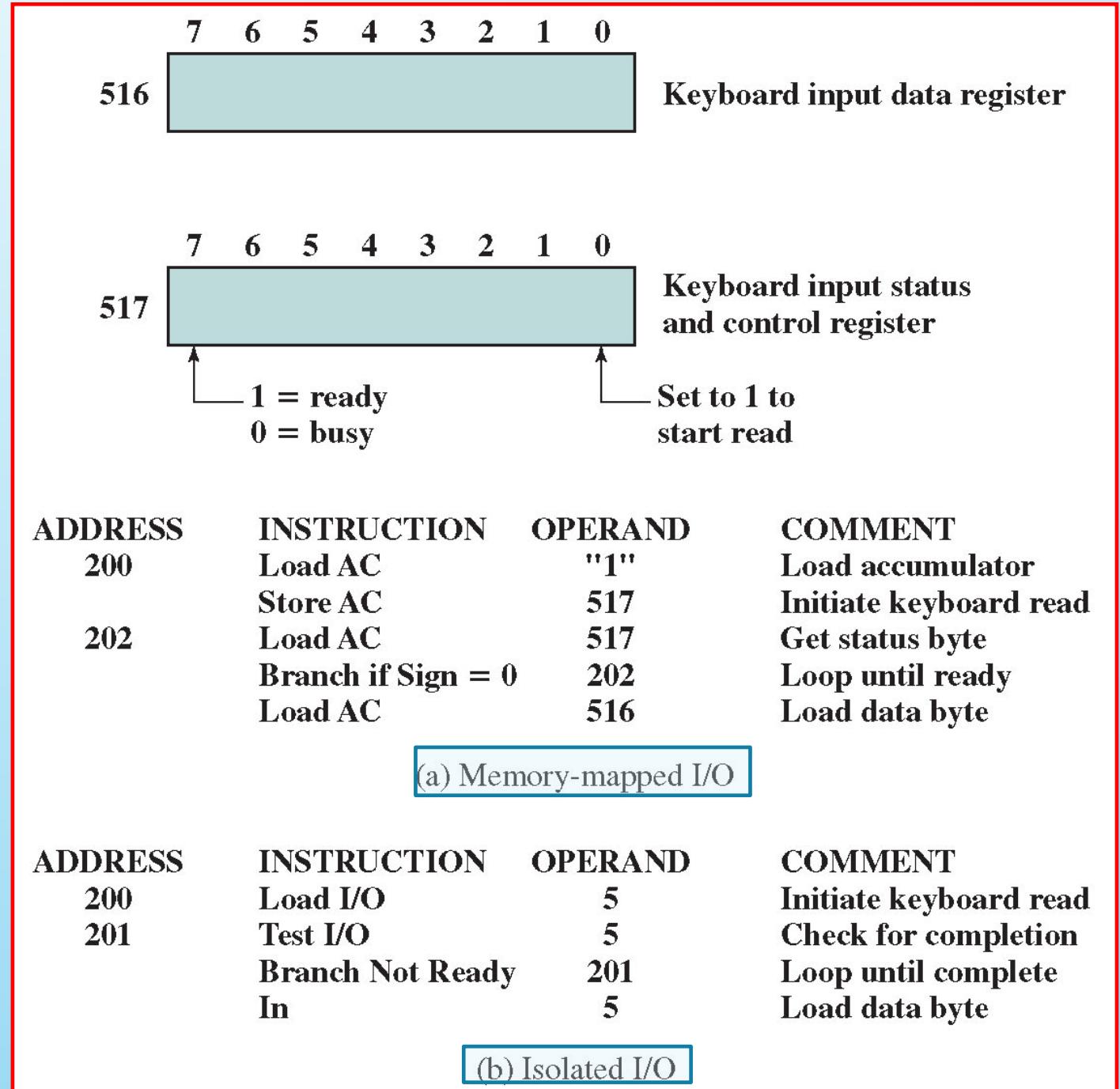
- Alternativ, magistrala poate fi echipată cu: linii de comandă de citire și scriere în memorie, plus linii de comandă de intrare și ieșire.
 - Acum, linia de comandă specifică dacă adresa se referă la o locație de memorie sau la un dispozitiv I/O.
 - Gama completă de adrese este disponibilă pentru ambele.
 - Din nou, cu 10 linii de adrese, sistemul poate suporta acum atât 1024 de locații de memorie, cât și 1024 adrese I/O.
- Deoarece spațiul de adrese pentru I/O este izolat de cel pentru memorie, acest lucru este denumit **I/O izolat**.

Exemplu

Figura (care urmeaza) compara aceste două tehnici I/O programate.

- Figura 2a arată modul în care interfața pentru un dispozitiv de intrare simplu, cum ar fi o tastatură, poate apărea unui programator care utilizează I/O cu memorie mapata.
 - Să presupunem o adresă pe 10 biți, cu memorie de 512-biți (locații 0-511) și până la 512 adrese I/O (locații 512-1023).
 - Două adrese sunt dedicate intrării de la tastatură. Adresa 516 se referă la registrul de date, iar adresa 517 se referă la registrul de stare, care funcționează și ca registrul de control pentru primirea comenziilor procesorului.
- Programul prezentat va citi 1 byte de date de la tastatură și îl va introduce într-un registru acumulator din procesor. Rețineți că procesorul ramane în bucla până când octetul de date este disponibil.

Figura 2



- Pentru I/O cu memorie izolata (**Figura 2 b**), porturile I/O sunt accesibile numai prin comenzi I/O speciale, care activează liniile de comandă I/O din magistrală.
- Pentru majoritatea tipurilor de procesoare, există un set relativ mare de instrucțiuni diferite pentru a adresa memoria. Dacă se utilizează I/O cu memorie izolată, există doar câteva instrucțiuni I/O.
- Astfel, un avantaj al I/O cu memorie mapată este că acest repertoriu mare de instrucțiuni poate fi folosit, permitând o programare mai eficientă.
- Un dezavantaj este că spațiul valoros de adrese de memorie este blocat. Atât I/O cu memorie mapată, cât și cele memorie izolată cu sunt utilizate în mod obișnuit.

Operațiuni I/O cu intrerupere

- Problema asociată cu I/O programată este că procesorul trebuie să aștepte mult timp modulul de I/O care este ocupat cu receptia sau transmiterea datelor.
 - Procesorul, în timp ce așteaptă, trebuie să interogheze în mod repetat starea modulului I/O.
 - Ca urmare, nivelul performanței întregului sistem este puternic degradat.
- O alternativă este ca procesorul să emită o comandă de I/O către modul și apoi să continue să facă alte activități utile.
- Modulul I/O va îintrerupe apoi procesorul pentru a solicita serviciul atunci când este gata să facă schimb de date cu procesorul.
- Procesorul execută apoi transferul de date, ca mai înainte, și apoi își reia prelucrarea anterioară.

Actiunea modulului I/O.

- Pentru operatiunea de intrare, modulul I/O primește o comandă READ de la procesor.
 - Modulul I/O va citi (in consecinta) date de la un periferic asociat.
- Odată ce datele se află în registrul de date al modulului, modulul semnalează printr-o îñtrerupere pe o linie de control spre procesor.
- Modulul așteaptă până când datele sale sunt solicitate de către procesor.
 - Când CPU face cererea, modulul își pune datele pe magistrala de date și este apoi pregătit pentru o altă operație I/O.

Acțiunea procesorului.

- Procesorul emite o comandă READ.
 - Apoi se opreste și face altceva (procesorul poate lucra la mai multe programe diferite în același timp).
- La sfârșitul fiecărui ciclu de instrucțiune, procesorul verifică întreruperile
- Când se produce întreruperea de la modulul I/O, procesorul salvează contextul (de exemplu, registrul de programe și registrul procesorului) programului curent și procesează întreruperea.
 - În acest caz, procesorul citește cuvântul de date din modulul I/O și îl stochează în memorie.
- Apoi restabilește contextul programului pe care lucra (sau alt program) și reia execuția.

Figura 1b prezintă utilizarea interogării I/O pentru citirea într-un bloc de date.

- I/O cu întrerupere este mai eficientă decât I/O cu programare, deoarece elimină aşteptarea inutilă.
- I/O cu întrerupere consumă totusi mult timp procesor, deoarece fiecare cuvânt de date care trece de la memorie la modulul I/O sau de la modulul I/O la memorie trebuie să treacă prin procesor.

Accesul direct la memorie (DMA)

Dezavantaje ale I/O programate și întrerupte

- I/O cu întreruperi, deși mai eficient decât I/O programat, necesită intervenția activă a procesorului pentru a transfera date între memorie și un modul I/O, iar orice transfer de date trebuie să traverseze o cale prin procesor.
- Astfel, ambele forme de I/O suferă de două dezavantaje inerente:
 - 1. Rata de transfer I/O este limitată de viteza cu care procesorul poate testa și servi un dispozitiv.
 - 2. Procesorul este legat la gestionarea unui transfer I/O; un număr de instrucțiuni trebuie executate pentru fiecare transfer I/O

Există un fel compromis între aceste două dezavantaje.

- Sa luam în consideratie transferul unui bloc de date.
 - Folosind I/O programate, procesorul este dedicat sarcinii I/O și poate muta date cu o rată destul de ridicată, cu costul de a nu face altceva.
 - Întreruperea I/O eliberează procesorul într-o oarecare măsură în detrimentul ratei de transfer I/O.
 - Cu toate acestea, ambele metode au un impact negativ atât asupra activității procesorului, cât și asupra ratei de transfer I/O.

Atunci când trebuie mutate volume mari de date, este necesară o tehnică mai eficientă: accesul direct la memorie (DMA).

Functia DMA

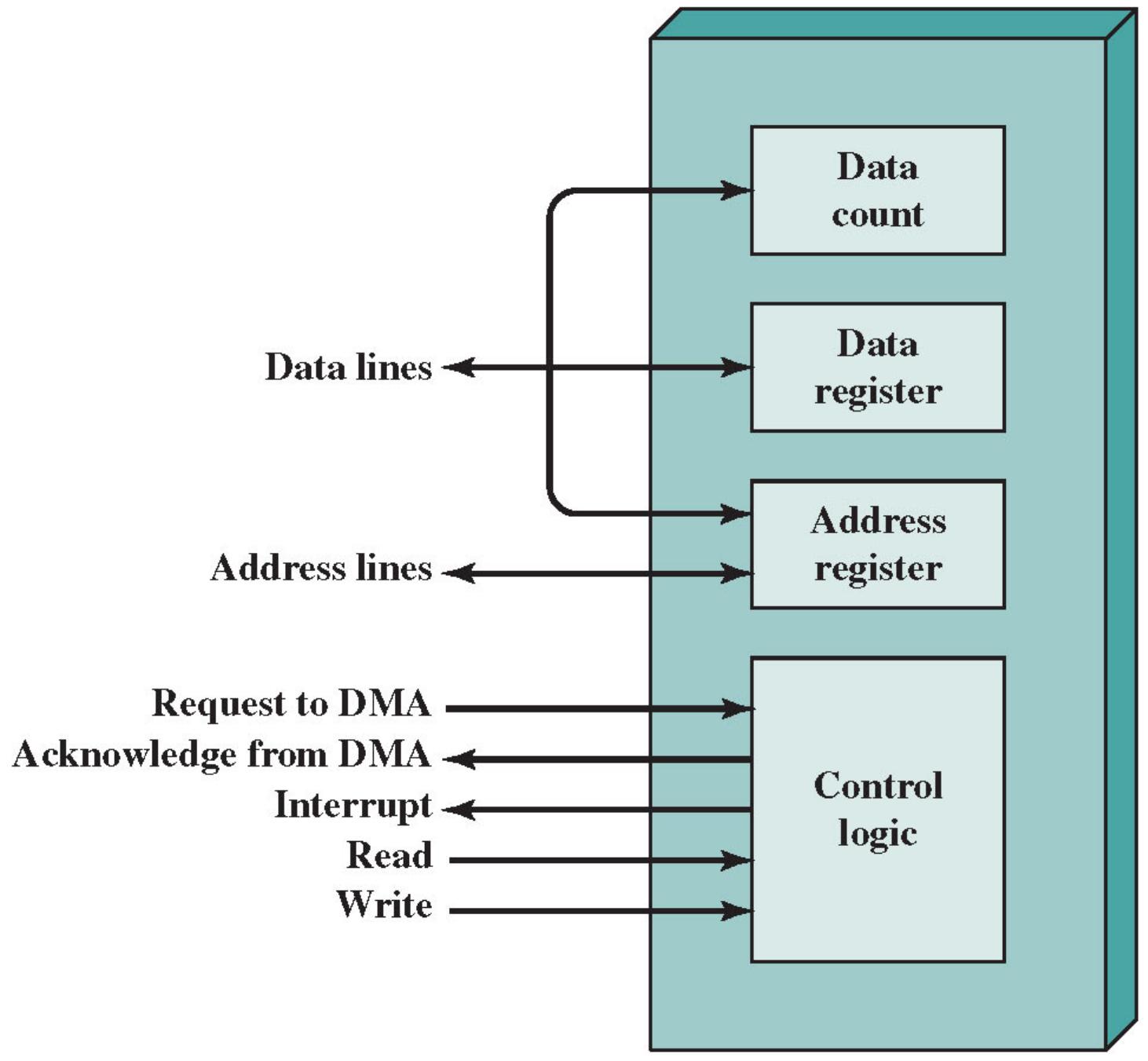
DMA implică existența unui modul suplimentar pe magistrala de sistem.

- **Modulul DMA** este capabil să imită procesorul și să preia controlul asupra sistemului de la procesor.
 - Trebuie să facă acest lucru pentru a transfera date către și din memorie prin magistrala de sistem.
 - În acest scop, modulul DMA
 - trebuie să utilizeze magistrala numai atunci când procesorul nu are nevoie de ea sau
 - trebuie să forțeze procesorul să își suspende temporar funcționarea.
 - Ultima tehnică este mai frecventă și este cunoscută ca furt de ciclu, deoarece modulul DMA în mod efectiv fură un ciclu de magistrala.

Când procesorul dorește să citească sau să scrie un bloc de date, emite o comandă către modulul DMA, trimițând către modulul DMA următoarele informații:

- Dacă se solicită citirea sau scrierea, folosind linia de control pentru citire sau scriere între procesor și modulul DMA
- Adresa dispozitivului I/O implicat, comunicat pe liniile de date
- Locația de pornire din memorie pentru citirea sau scrierea, comunicarea pe liniile de date și stocata de modulul DMA în registrul său de adrese
- Numărul de cuvinte care trebuie citite sau scrise, comunicate din nou prin linii de date și stocate în registrul de numărare a datelor

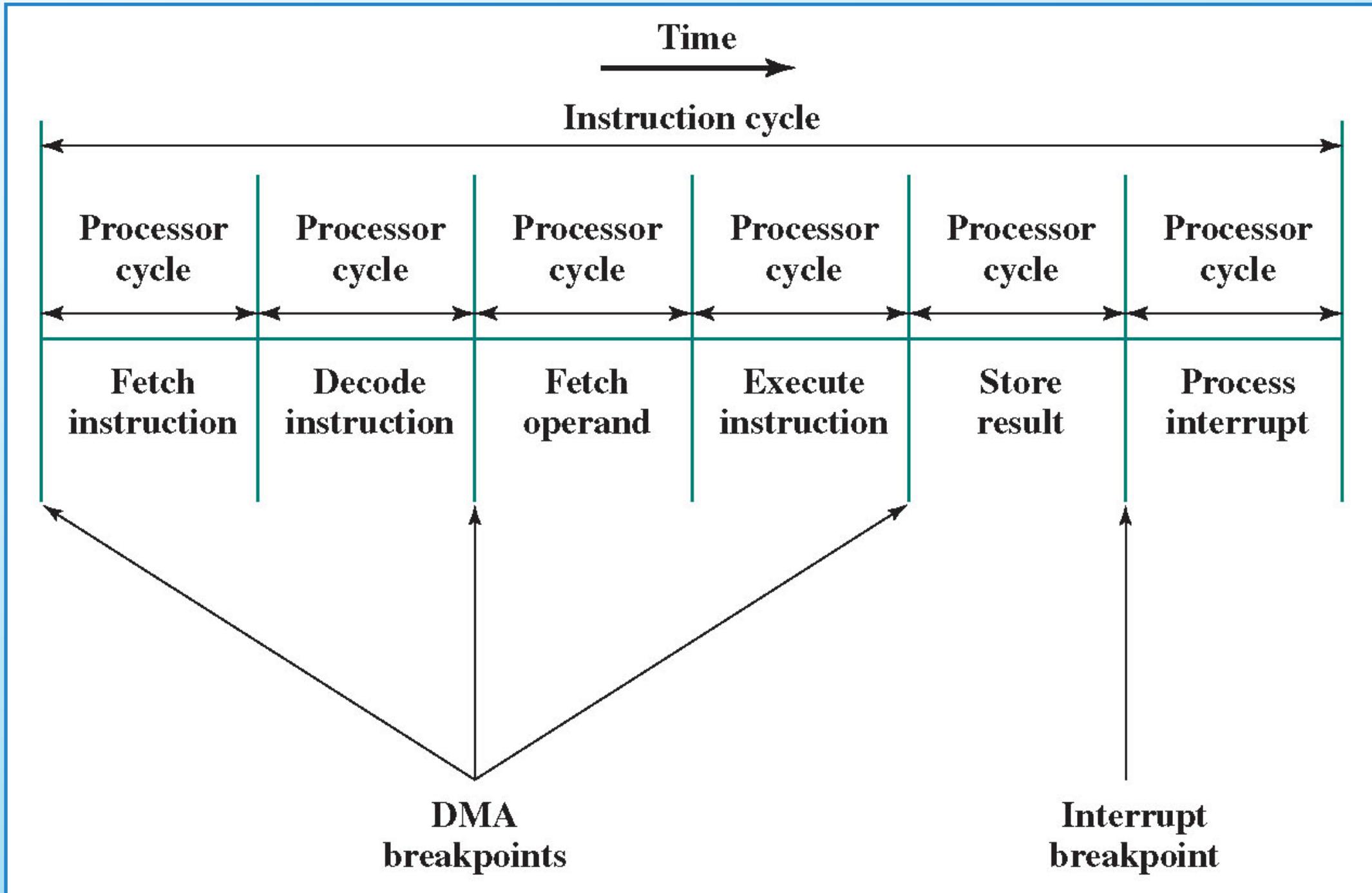
Figura 3



Procesorul continuă apoi cu alte lucrări.

- A delegat această operație I/O catre modulul DMA.
- Modulul DMA transferă întregul bloc de date, câte un cuvânt la un moment dat, direct către sau din memorie, fără a trece prin procesor.
- Când transferul este finalizat, modulul DMA trimite un semnal de întrerupere procesorului.
- Astfel, procesorul este implicat doar la începutul și la sfârșitul transferului.

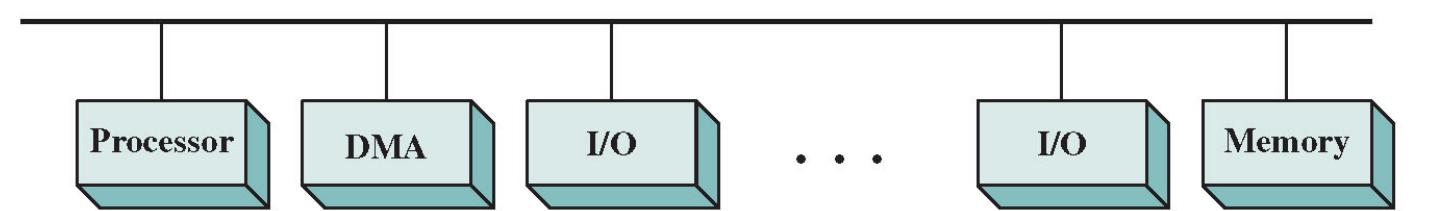
Figura 4



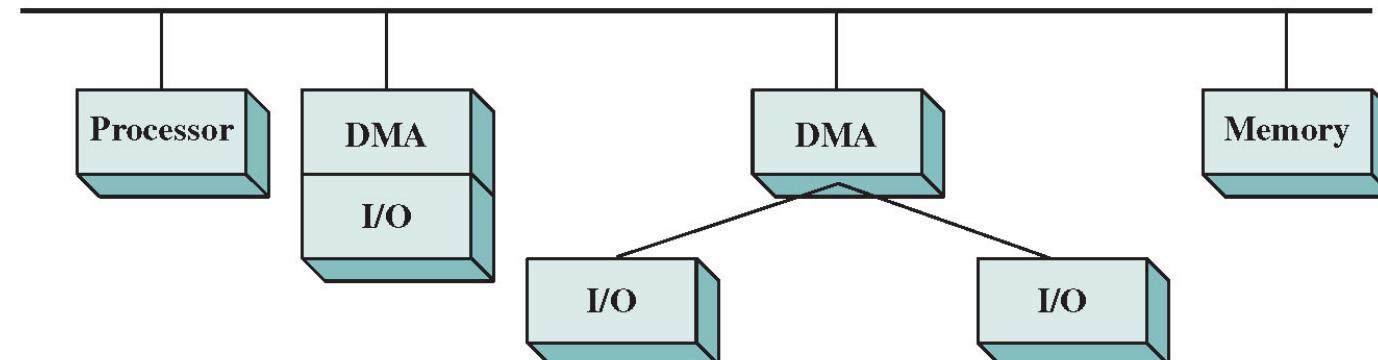
- Figura 4 arată unde, în ciclul instrucțiunii, procesorul poate fi suspendat.
 - În fiecare caz, procesorul este suspendat chiar înainte de a folosi magistrala.
- Modulul DMA transferă apoi un cuvânt și readuce controlul la procesor.
 - **Rețineți că aceasta nu este o întrerupere; procesorul nu salvează un context și nu face altceva.**
 - procesorul se oprește pentru un ciclu de magistrala.
- Efectul global este ca procesorul executa operațiile mai lent.
 - Pentru un transfer I/O cu mai multe cuvinte, DMA este mult mai eficient decât I/O programate sau cu întreruperi

- Mecanismul DMA poate fi configurat într-o varietate de moduri.
 - Unele posibilități sunt prezentate în Figura 5.
- În primul exemplu, toate modulele partajează aceeași magistrală de sistem.
 - Modulul DMA, care acționează ca un procesor surogat, utilizează I/O programat pentru a schimba date între memorie și un modul I/O prin modulul DMA.
 - Această configurație, deși poate fi ieftină, este în mod clar ineficientă.
 - Ca și în cazul I/O programate controlate de procesor, fiecare transfer al unui cuvânt consumă două cicluri de magistrală. (un transfer memorie-DMA și un transfer DMA-I/O)

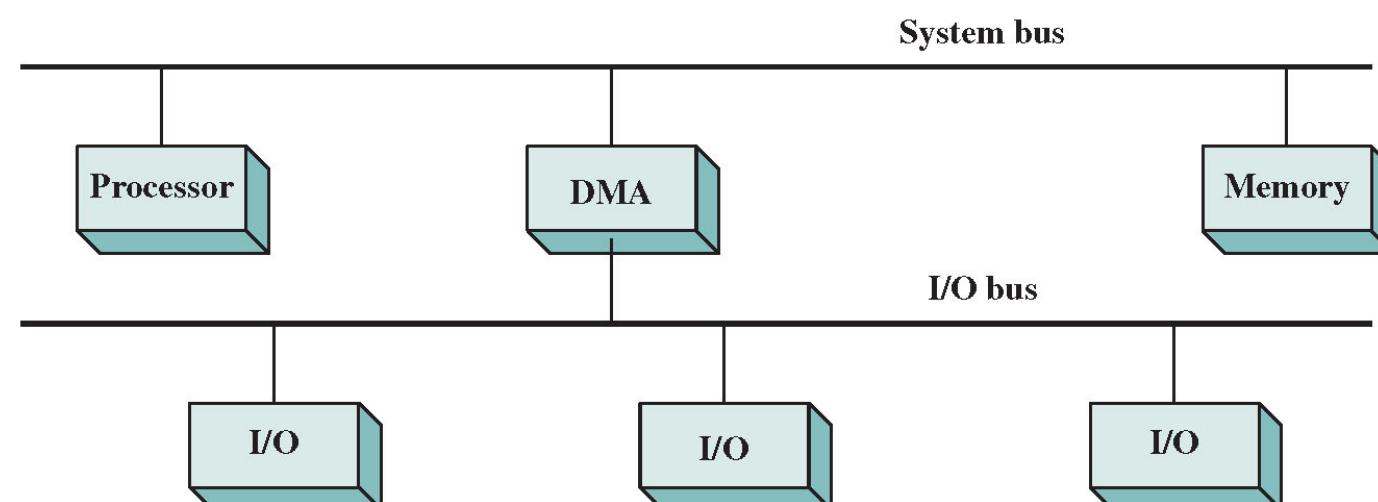
Figura 5



(a) Single-bus, detached DMA



(b) Single-bus, integrated DMA-I/O

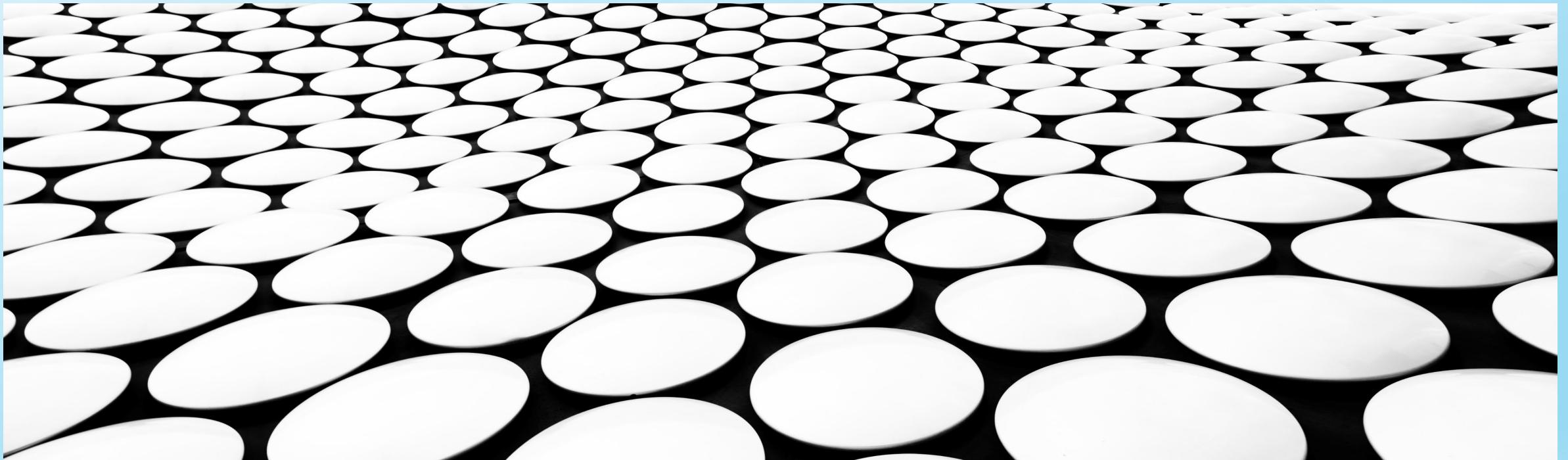


(c) I/O bus

- Numărul necesar de cicluri de magistrala poate fi redus substanțial prin integrarea funcțiilor DMA și I/O.
 - După cum indică **Figura 5b**, aceasta înseamnă că există o cale între modulul DMA și unul sau mai multe module I/O care nu include magistrala de sistem.
 - Logica DMA poate fi de fapt o parte a unui modul I/O sau poate fi un modul separat care controlează unul sau mai multe module I/O.
- Acest concept poate fi preluat cu un pas prin conectarea modulelor I/O la modulul DMA folosind o magistrală I/O (**Figura 5c**).
 - Aceasta reduce numărul de interfețe I/O din modulul DMA la una și asigură o configurație ușor de expandat.
 - În ambele cazuri (**Figurile 5b și 5c**), magistrala de sistem pe care modulul DMA o partajează cu procesorul și memoria este folosită de modulul DMA numai pentru a schimba datele cu memoria.
 - Schimbul de date între modulele DMA și I/O are loc în afara magistralei de sistem.

ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



Performante energetice

Care este interesul programatorului

Programatorii de succes au fost întotdeauna preocupați de performanța programelor lor, deoarece obținerea rapidă a rezultatelor pentru utilizator este esențială în crearea unui software de succes.

Anii 1960 - 1970 – memoria a fost constrângerea primară.

Anii 2000 - 2010 – Natura paralelă a procesoarelor și natura ierarhică a memoriilor.

Astăzi – Eficiența energetică a programelor devine o tinta relevanta.

Ce tehnici pot fi folosite de designerii de hardware pentru a îmbunătăți eficiența energetică?

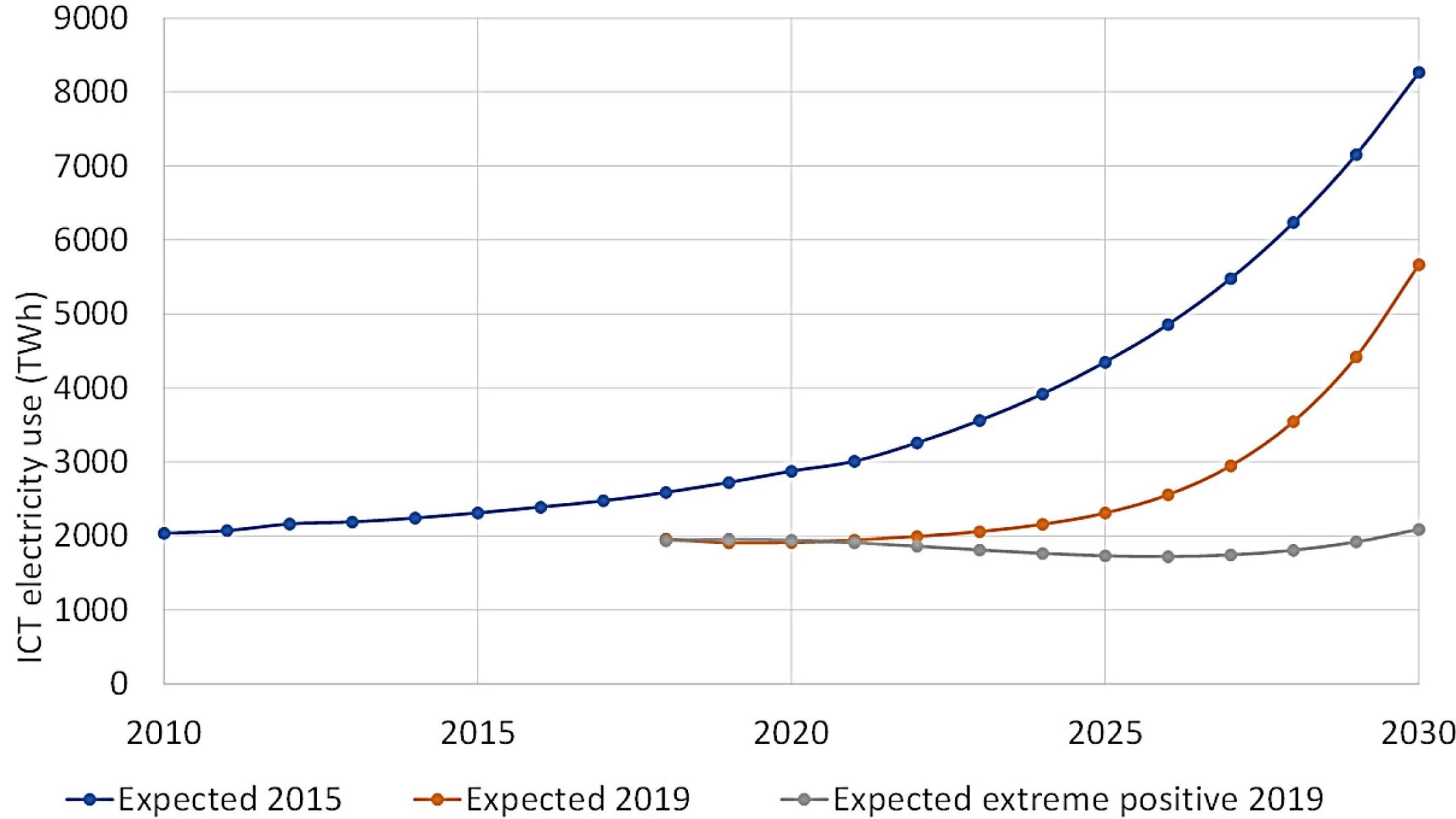
Modificarea frecvenței de ceas

Atât frecvența de ceas, cât și puterea au crescut rapid timp de zeci de ani, apoi creșterea s-a aplatizat relativ recent (începând 2004 cu Pentium IV Prescott).

Motivul pentru care cei două parametrii au crescut împreună este că sunt corelați.

Motivul pentru încetinirea recentă a creșterii este că ne-am confruntat cu apropierea de limita practică de putere impusă de racirea microprocesoarelor

Deși puterea manifestă o limită impusă ceea ce putem răci, în era PostPC resursa cu adevărat critică este **energia**.



La fel cum măsurarea timpului în secunde este o măsură mai certă a performanței programului decât o rată precum MIPS, măsurarea energiei în Jouli este o măsură mai bună decât o putere exprimată în Watt, care este Joul/secundă.

Tehnologia dominantă pentru circuitele integrate se numește:
CMOS (semiconductor complementar de oxid de metal)

Energia dinamică: energie care este consumată atunci când tranzistoarele comută între stările de 0 și 1 sau invers.

Energia ~ capacitatea electrică × tensiunea electrică²

Aceasta relație exprimă energia unui puls în timpul dublei tranzitii logice $0 \rightarrow 1 \rightarrow 0$ sau $1 \rightarrow 0 \rightarrow 1$.

Pentru o singură tranzitie energia este:

Energia ~ $\frac{1}{2} \times$ capacitatea electrică × tensiunea electrică²

Puterea necesară per tranzistor este doar produsul energiei unei tranziții cu frecvența tranzițiilor:

Puterea $\sim \frac{1}{2} \times$ capacitatea electrică \times tensiunea electrică² \times frecvența de comutare

Frecvența de comutare este funcție de frecvența de ceas.

Capacitatea per tranzistor este o funcție atât de numărul de tranzistori conectați la o ieșire cât și de tehnologie, care determină capacitatea atât a firelor, cât și a tranzistorilor.

Deși energia dinamică este sursa principală de consum de energie în CMOS, consumul de **energie statică** are loc din cauza **currentului de pierderi** care curge chiar și atunci când un tranzistor este oprit.

La servere, pierderile sunt de obicei responsabile pentru 40% din consumul de energie.

Astfel, creșterea numărului de tranzistori mărește disiparea puterii, chiar dacă tranzistoarele sunt întotdeauna oprite.

PERFORMANTE ENERGETICE

- Consumul de energie per comutare ($0 \rightarrow 1$ sau $1 \rightarrow 0$)

$$E_C = \frac{1}{2} C_l \cdot U^2$$

C_l este o capacitate specifică circuitului de comutare.

Aceasta capacitate este proporțională cu aria specifică medie a unui circuit de comutare și depinde de tehnologia microelectronică de fabricare a cipurilor.

Puterea P consumată de circuitul de comutare este proporțională cu frecvența de comutare (F_C):

$$P = E_C \cdot F_C$$

Modalități de scadere a consumului de energie

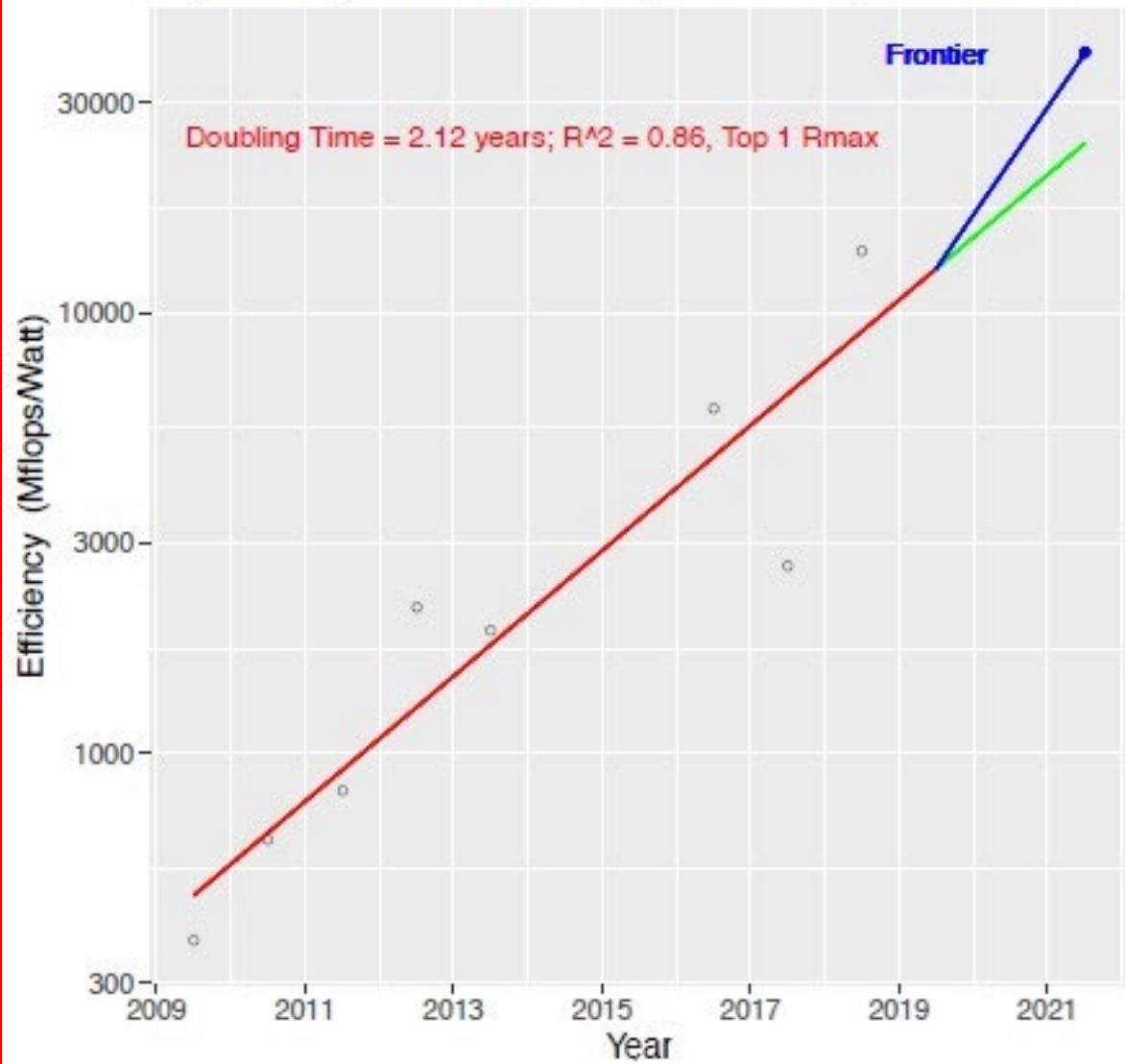
- Scaderea tensiunii de alimentare
- Miniaturizarea

Factori care determină creșterea consumului de energie

- Creșterea frecvenței de lucru
- Efectul de ambalare termică:

La creșterea temperaturii interne a circuitului crește capacitatea specifică C_1 și odată cu ea crește suplimentar consumul de energie. Aceasta ultima creștere are ca efect creșterea suplimentară temperaturii.

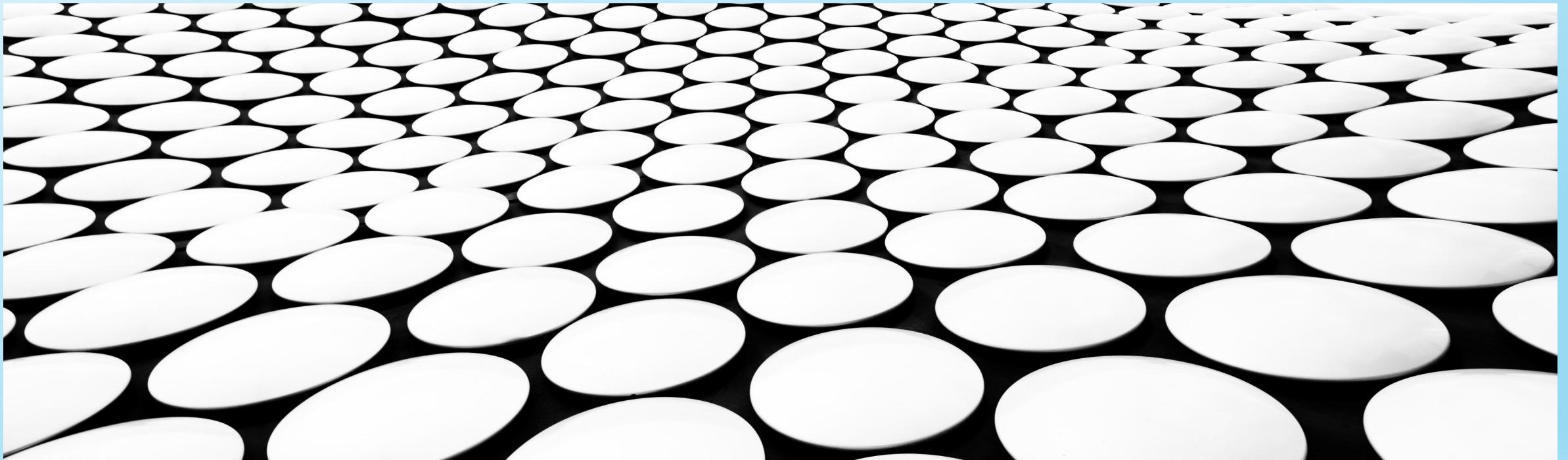
Top500 list, Efficiency of Top Performing Machines



	Processor Power (Watt)	Baseline Phases (Watt)	Atributable to Linpack Execution (Watt)	Rmax (GFLOPS)	Power Efficiency (GFLOPS/Watt)	RSD (Relative Standard Deviation)
i5-3210M (3rd gener.)	19.85 ± 0.06	11.90 ± 2.52	7.95 ± 1.69	32.64	4.1 ± 0.87	0.21
i5-4200H (4th gener.)	37.59 ± 0.25	12.19 ± 1.56	25.40 ± 3.27	74.64	2.9 ± 0.38	0.13
i7-6500U (6th gener.)	15.55 ± 0.17	1.30 ± 0.38	14.25 ± 4.18	63.92	4.5 ± 1.32	0.29
i5-10400T (10th gener.)	30.03 ± 0.06	1.14 ± 0.40	28.89 ± 10.14	112.28	3.9 ± 1.36	0.36
i7-1165G7 (11th gener.)	21.22 ± 1.94	2.31 ± 0.25	19.01 ± 2.40	148.47	7.8 ± 0.99	0.13

ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



Tipuri de formate de instrucțiuni masina

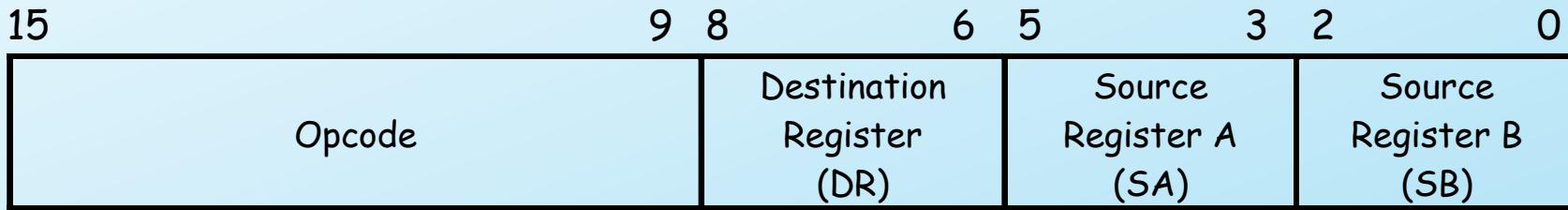
Tipuri de formate de instrucțiuni masina (3 tipuri)

- ✖ Instrucțiuni tip: **Register format**
 - + Necesita **doi registri** sursa
- ✖ Instrucțiuni tip: **Immediate format**
 - + Necesita **un registru** sursa si o constanta **operand** (inclusa in corpul instructiunii)
- ✖ Instrucțiuni tip: **Jump and branch format**
 - + Necesita **un registru** sursa si o constanta **adresa** (inclusa in corpul instructiunii)

Formatul instrucțiunilor

Exemplu pe 16 biti

Register format (16 biti)



Un camp de 7 biți pentru codul operațiunii (**opcode**)

Un camp de 3 biți pentru registrul destinație

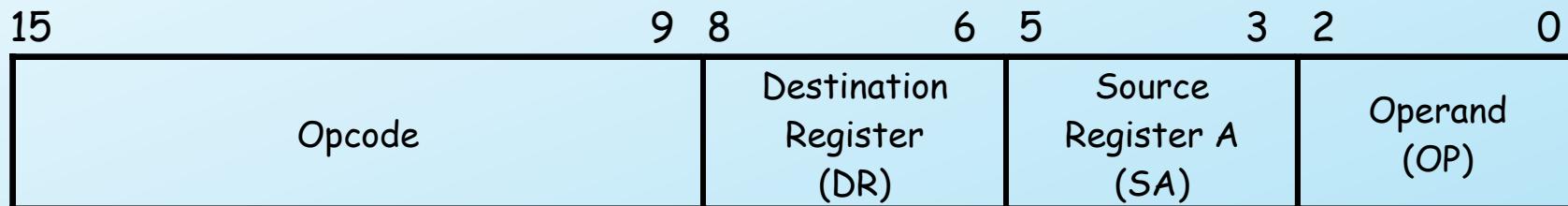
Doua campuri de 3 biți pentru registrele sursă

Aceste tipuri de instrucțiuni sunt valabile pentru arhitecturi cu pana la **8 registrii**

Exemplu

ADD R1 , R2 , R3

Immediate format



Un camp de 7 biți pentru codul operațiunii

Un camp de 3 biți pentru registrul destinație

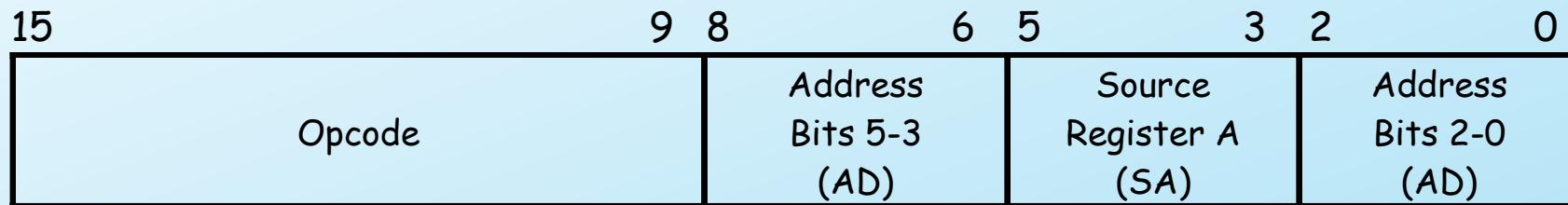
Un camp de 3 biți pentru registrul sursă

Un camp de 3 biți pentru constanta operand

Exemplu

ADD R1, R2, #3

Jump and branch format



Un camp de 7 biți pentru codul operațiunii

Un camp de 3 biți pentru registrul destinație

Un camp (compus din două segmente) de 6 biți pentru constanta adresa.

Segmentarea acestui camp se face pentru uniformitate.

Exemplu

BZ R3, -24
JMP 18

Instruction type
Register-format ALU operation
Register-format shift operation
Memory write (from registers)
Memory read (to registers)
Immediate ALU operation
Immediate shift operation
Conditional branch
Jump

Instruction type	Opcode bits		
	15	14	13
Register-format ALU operation	0	0	0
Register-format shift operation	0	0	1
Memory write (from registers)	0	1	0
Memory read (to registers)	0	1	1
Immediate ALU operation	1	0	0
Immediate shift operation	1	0	1
Conditional branch	1	1	0
Jump	1	1	1

Biții 12-9

Corespond codului de selecție a funcției ALU

FS	Operation
00000	$F = A$
00001	$F = A + 1$
00010	$F = A + B$
00011	$F = A + B + 1$
00100	$F = A + B'$
00101	$F = A + B' + 1$
00110	$F = A - 1$
00111	$F = A$
01000	$F = A \wedge B$ (AND)
01010	$F = A \vee B$ (OR)
01100	$F = A \oplus B$
01110	$F = A'$
10000	$F = B$
10100	$F = sr\ B$ (shift right)
11000	$F = sl\ B$ (shift left)

Instruction	Bits 15-9 (Opcode)	Bits 8-6	Bits 5-3	Bits 2-0	Format
LD R1, (R0)	011xxxxx	001	000	xxx	Immediate
BZ R1, +4	110x011	000	001	100	Jump/branch
SUB R5, R5, #1	1000100	101	101	001	Immediate
ADD R1, R0, R5	0000010	001	000	101	Register
JMP -3	111xxxxx	111	xxx	101	Jump/branch

Conversia si executia instructiunilor de catre Unitatea de Control

Unitatea de control converteste o instrucțiune într-un set de semnale recunoscute de:
ALU, registri și alte blocuri funcționale ale unui CPU.

- IR: Instruction Register

- PSR: Processor Status Register

N (negative)

The result of the last ALU operation is negative (MSB = 1)

Z (zero)

The result of the last ALU operation is zero

C (carry)

The result of the last ALU operation has a carry-out

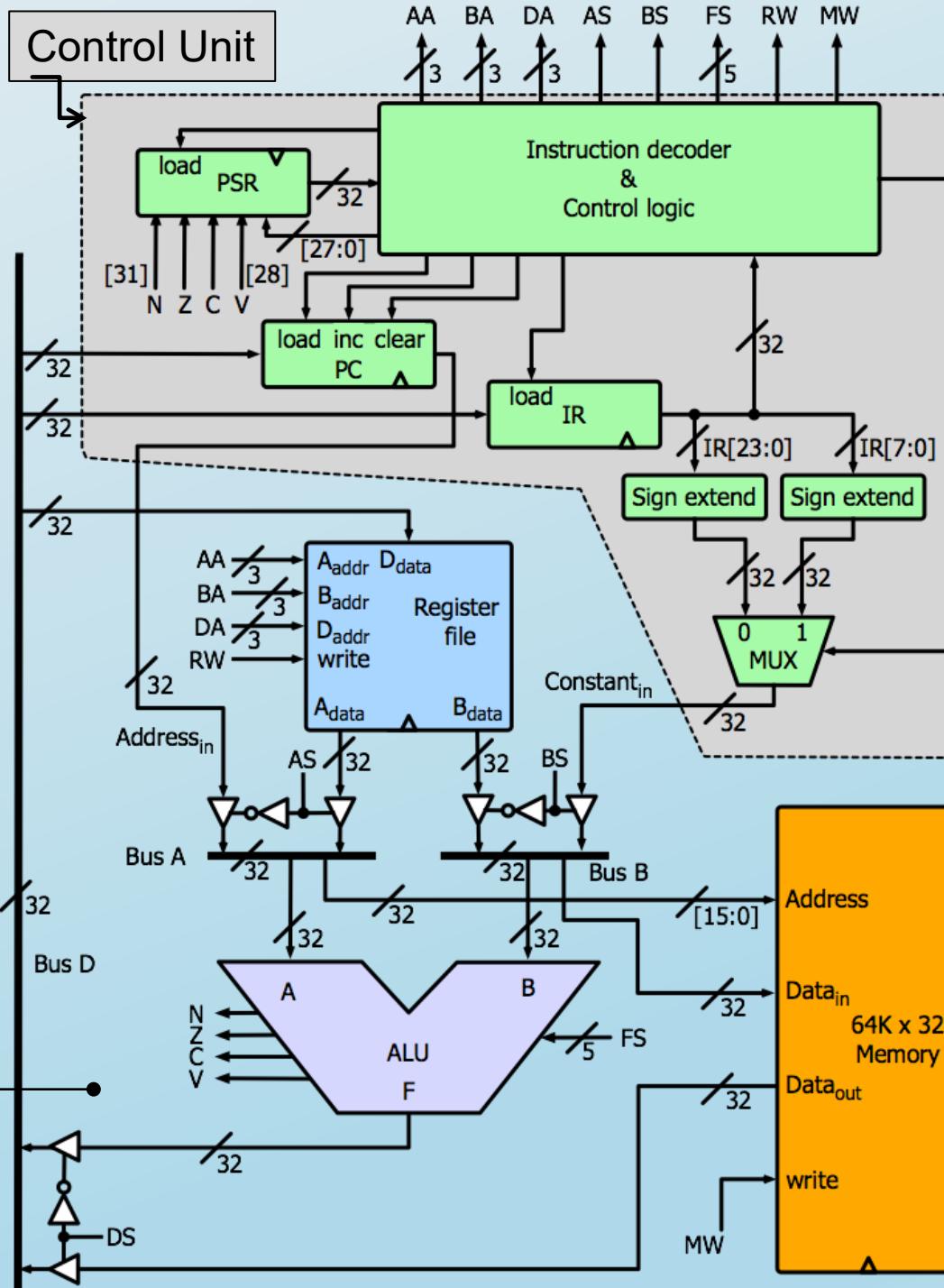
V (oVerflow)

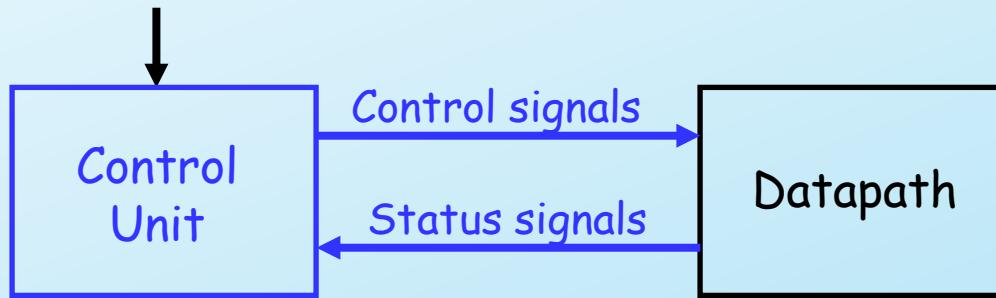
The result of the last ALU operation overflows

Decodorul de instructiuni citeste continutul IR;
sunt decodificati: **opcode** (din headerul
instructiunii) si **operanzii**

Apoi sunt generate semnalele corespunzatoare
operatiei specificate de opcode

Datapath

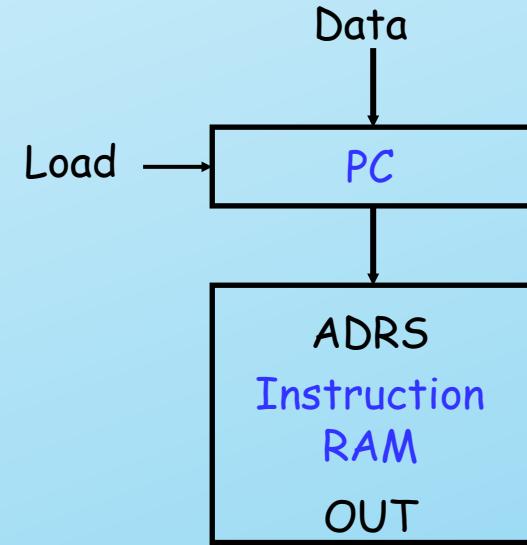




- ✖ Unitatea de control conecteaza programul cu blocul generic “Datapath”, care contine ALU, registrii si alte blocuri.
- ✖ Genereaza semnalele: WR, DA, AA, BA, MB, FS, MW, MD.
- ✖ Receptioneaza semnale dinspre Datapath cum ar fi: bitii de stare ALU: V, C, N, Z.

Program counter (PC)

- ✖ Executa două sarcini
- ✖ Load=0
 - + Adresa din PC este incrementată cu 1 (la fiecare ciclu de ceas)
- ✖ Load=1
 - + Adresa din PC este actualizată cu valoarea Data

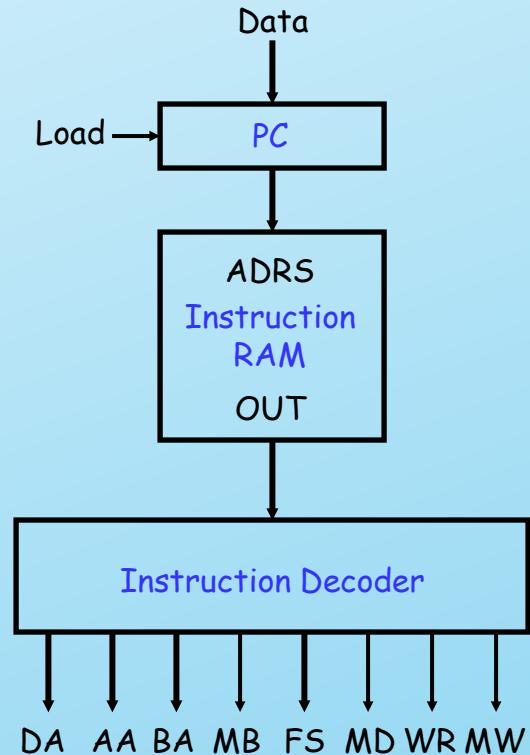


Instruction decoder

circuit combinațional

Genereaza semnale corespunzatoare adresei

Aceste semnale stabilesc ce registrii, ce locatie de memorie sau ce operatie ALU trebuie efectuata.

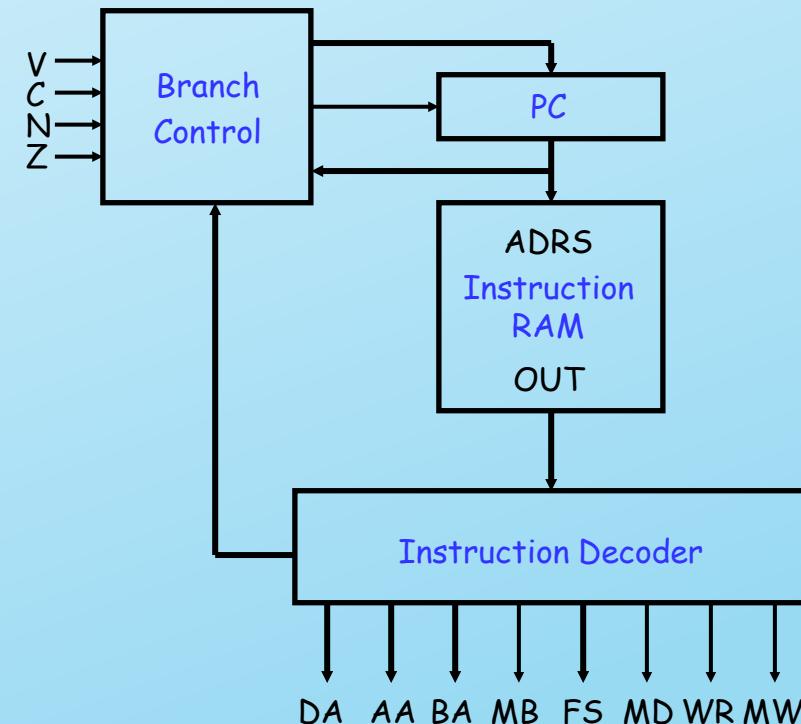


(to the datapath)



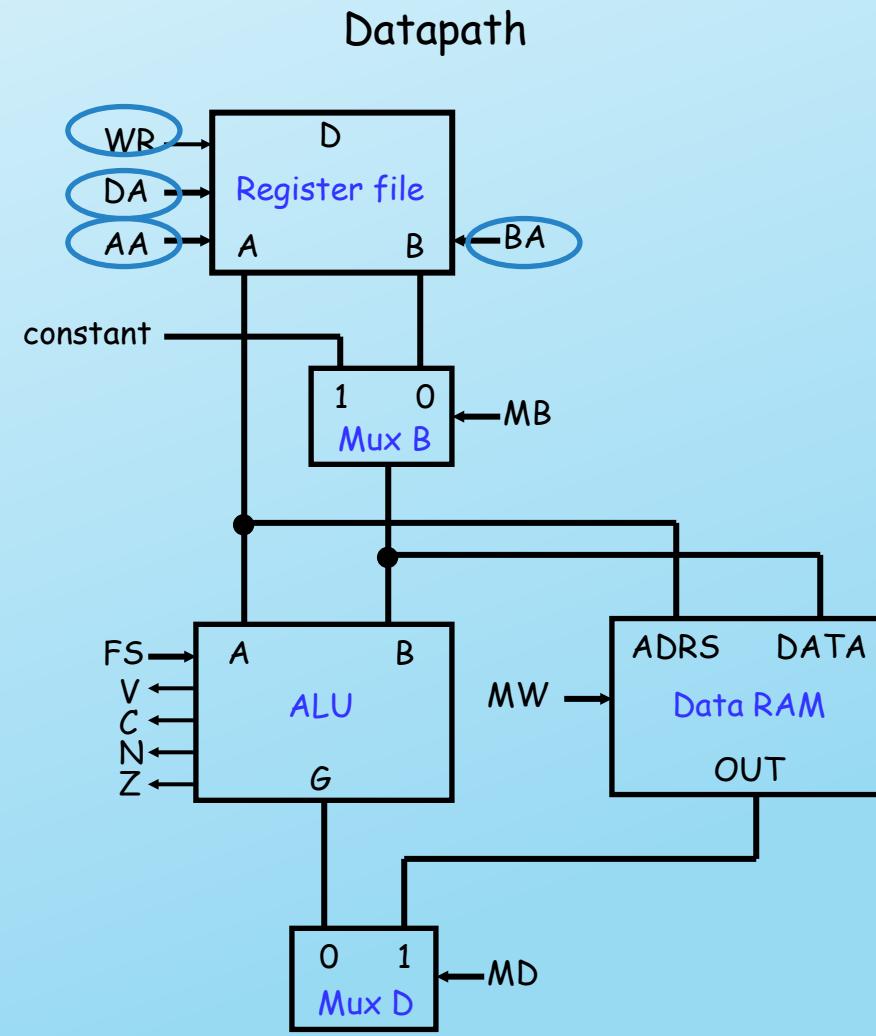
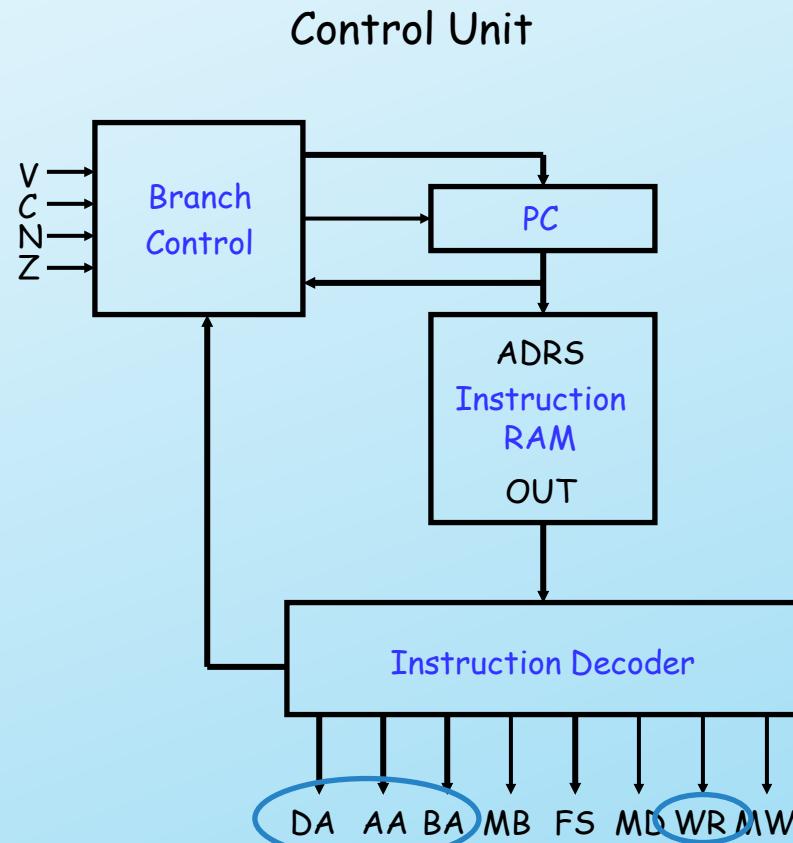
Jumps and branches

- ✖ branch control unit
- ✖ Decide ce valoare va fi incarcata in PC
 - + Jumps
 - ✖ PC va fi incarcata cu adresa tinta specificata in instructiune
 - + branch instructions
 - ✖ PC va fi incarcata cu adresa tinta numai daca bitul de stare este TRUE
 - + Pentru alte instructiuni
 - ✖ Este incrementat



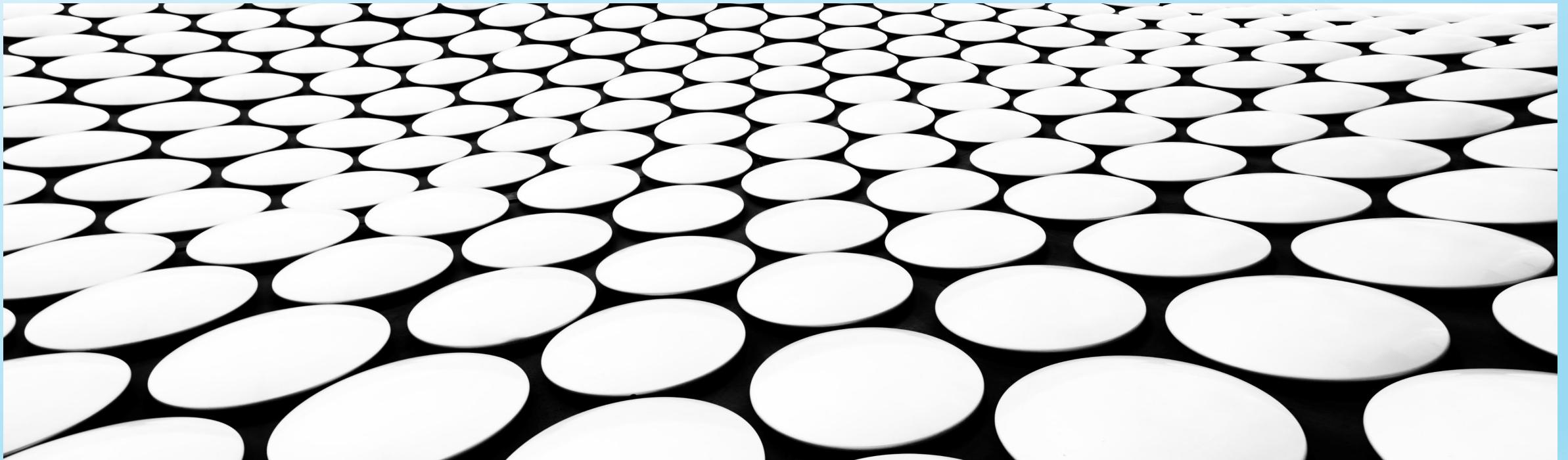


Intregul procesor



ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023



Calculatoare cuantice

Arhitectura calculatoarelor cuantice

Un computer cuantic are atât părți clasice, cât și cuantice.

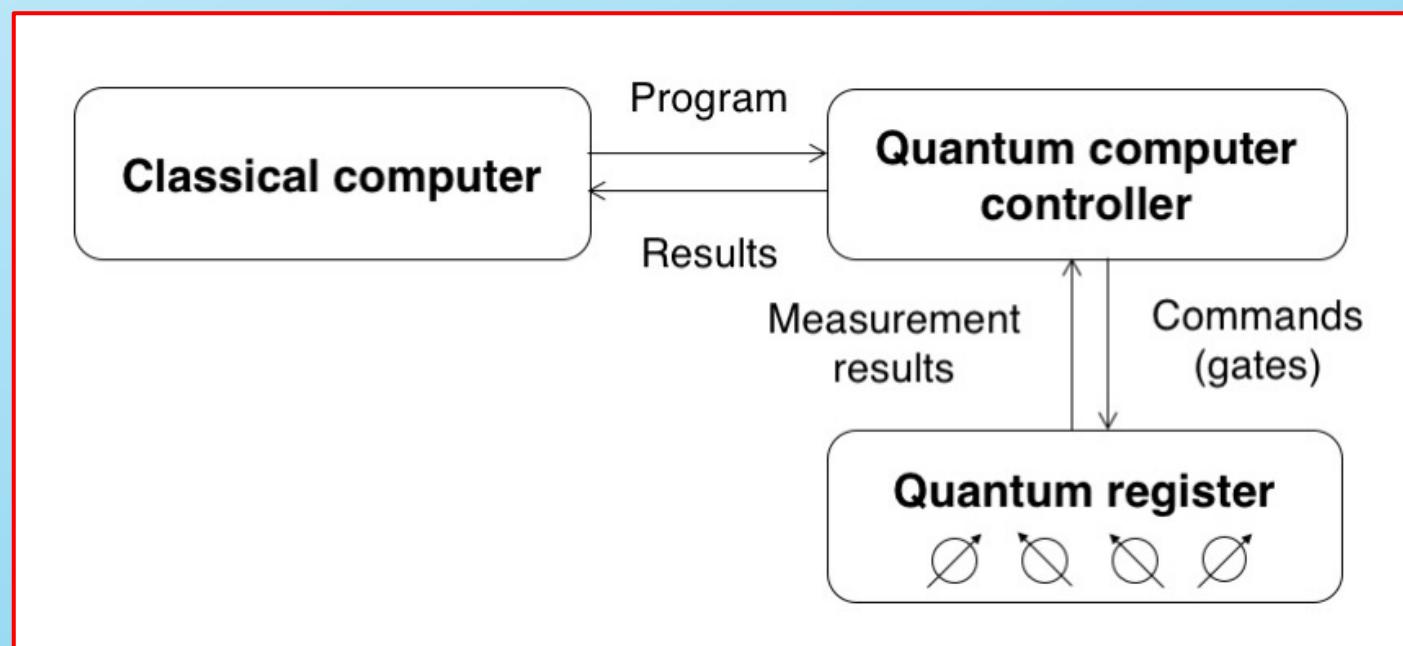
SIMULATOR: <https://algassert.com/quirk>

Modelul de bază al calculatoarelor cuantice

Atât computerul clasic, cât și cel cuantic constau în esență din trei părți :

- **Memoria** - Conține/stocheaza starea curentă a mașinii.
- **Un procesor sau un controler** - Efectuează operații elementare asupra stării mașinii.
- **Dispozitiv de intrare/ieșire** - Face posibilă definirea stării inițiale și obținerea stării finale de calcul.

- Registrii cuantici sunt memoria calculatoarelor cuantice. Dețin date cuantice pentru algoritm.
- Poțile cuantice sunt echivalentul instructiunilor.
- Controler de calculator deține programul și le spune dispozitivelor care controlează fiecare qubit să efectueze acțiuni conform instrucțiunilor.



Elemente de fizica calculatoarelor cuantice

INDISCERNABILITATE

1



2



Chiar da par identice, cele doua flori sunt diferite prin unele amanunte.

2



1

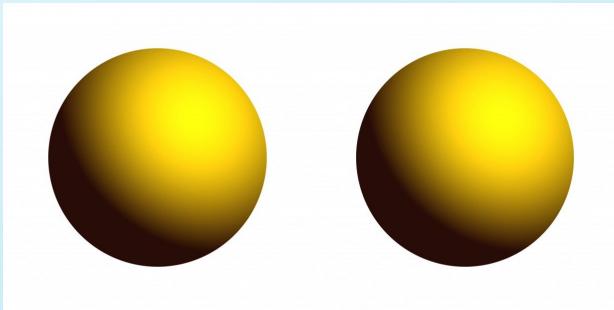


Florile de jos sunt inversate in pozitie

Cele doua flori
pot fi discernute

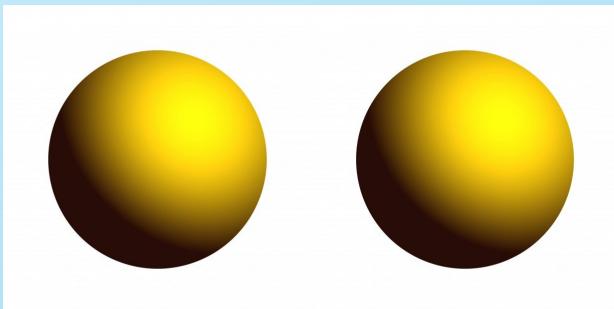
IN LUMEA MICROSCOPICA LUCRURILE NU STAU ASA

1



2

2



1

2 electroni nu pot sa difere

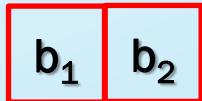
Ei pot sa isi schimbe pozitiile intre ei iar noi nu putem observa acest lucru

Aceste particule sunt indiscernabile

La fel se comporta si doi atomi identici sau doua molecule identice

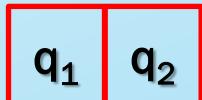
MEMORIA

Memorie clasica cu 2 biti



Cele doua celule de memorie sunt discernabile
(modelul balansoarului)
Putem sti daca cele doua celule se inverseaza, iar ele
pastreaza informatia initiala

Memorie cuantica (memorie care foloseste sisteme cuantice, microscopice)



In acest caz celulele sunt indiscernabile:
Nu putem sti daca cele doua celule si-au
inversat locul, si in consecinta nu putem sti
precis ce informatie contin

Una dintre problemele puse de
constructia unui calculator cuantic este
ridicarea (inlaturarea) indiscernabilitatii
intre celulele de memorie.

SETURI DE BITI

Intr-un calculator clasic:

$b_1, b_2, b_3, \dots b_n$ este un sir de biti memorati

poate fi scris $[b_1, b_2, b_3, \dots b_n]$ fiecare bit avand o pozitie precis definita

Intr-un calculator cuantic scriem seria de biti in felul urmator:

$|q_1, q_2, q_3, \dots q_n\rangle$

Prin aceasta scriere presupunem ca pozitia bitilor este bine definita (indiscernabilitatea este eliminata)

O celula de memorie de un bit, intr-un **calculator clasic**, poate inmagazina (la momente de timp diferite), doua valori diferite: **0** si **1**.

La un moment dat valoarea memorata este **cu certitudine** fie **0** fie **1**.

Intr-un calculator cuantic, valoarea memorata are **un anumit grad de incertitudine**.

Starea in care se gaseste celula de memorie este descrisa ca o superpozitie intre cele doua stari: $|0\rangle$ si $|1\rangle$

Ea este scrisa in felul urmator:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Expresia de mai sus descrie starea unui **qubit**.

Obiectul $|\psi\rangle$ este element al unui spatiu Hilbert. Expresia de mai sus deriva din legile fizicii cuantice.

α si β au valori complexe si respecta regula $|\alpha|^2 + |\beta|^2 = 1$

α si β se numesc amplitudini

Bitul din calculatorul clasic are drept corespondent **qubitul** in calculatorul cuantic.

In ambele cazuri reprezinta celula elementara de memorie.

$|\alpha|^2$ reprezinta probabilitatea ca qubitul sa se afle in starea $|0\rangle$

$|\beta|^2$ reprezinta probabilitatea ca qubitul sa se afle in starea $|1\rangle$

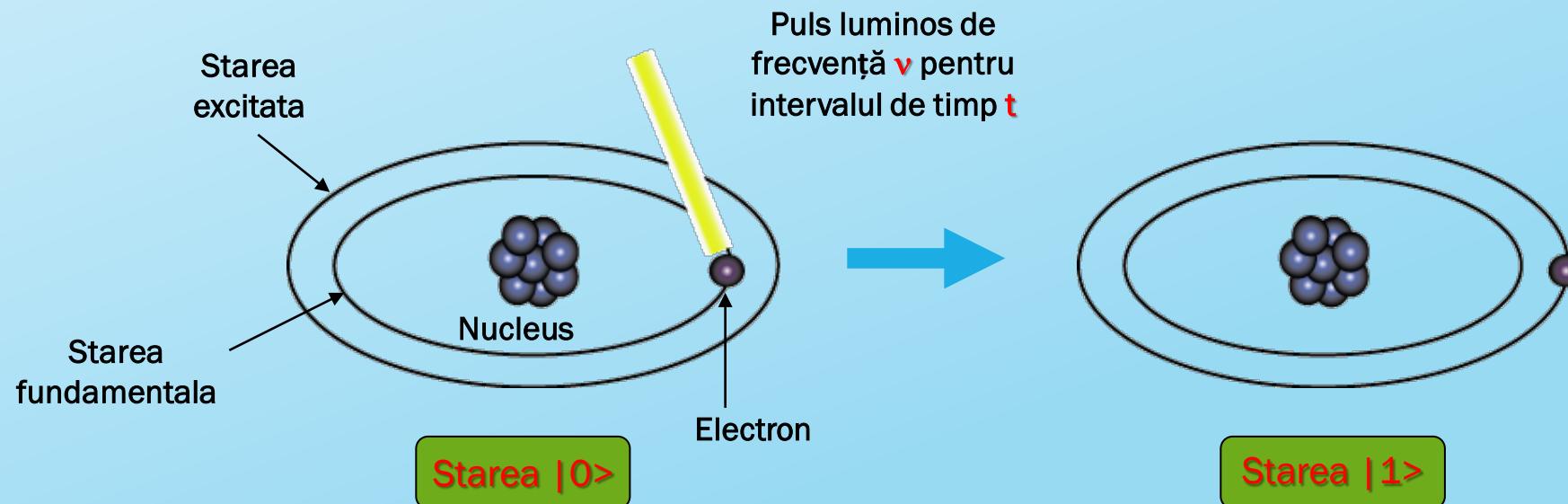
REPREZENTAREA DATELOR - QUBITII

O implementare fizică a unui qubit ar putea folosi cele două niveluri de energie ale unui atom.

O stare excitată reprezentând $|1\rangle$ iar o stare fundamentală reprezentând $|0\rangle$.

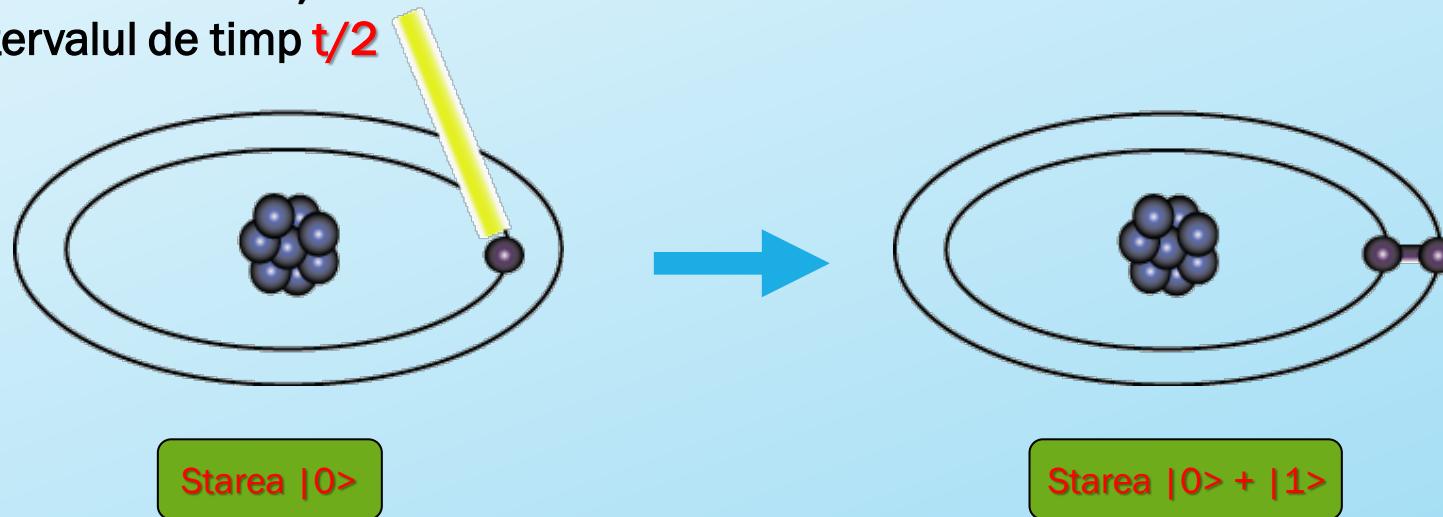
Un bit de date este reprezentat de un singur atom care se află într-una din cele două stări notate cu $|0\rangle$ and $|1\rangle$.

Un singur bit de această formă este cunoscut sub numele de qubit



REPREZENTAREA DATELOR - SUPRAPUNEREA

Puls luminos de frecvență ν
pentru intervalul de timp $t/2$



Avem o suprapunere (superpozitie) de stari.

Adica: sistemul se afla simultan in cele doua stari cu o anumita probabilitate

Daca se fac calcule complete atunci se gaseste ca:

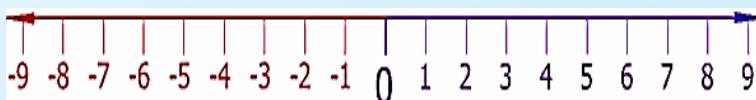
sistemul se afla in starea 1 cu o probabilitate de $\frac{1}{2}$ si
simultan in starea 2 cu o probabilitate de $\frac{1}{2}$

Comparatie: fizica clasica – fizica cuantica

Fizica clasica

Rezultate predictibile!

Marimile sunt continue



Fizica cuantica

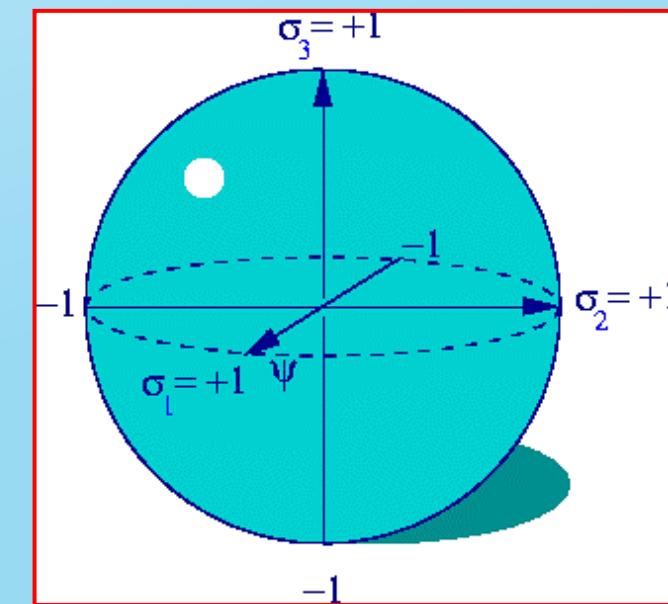
- Sunt violate legile fizici clasice la scara microscopica ($\sim h$, constanta Plank)
- Valorile sunt discrete $|0\rangle$ or $|1\rangle$
- Exista posibilitatea ca sistemul sa se afle simultan in mai multe stari (**superpozitie** de valori ,reprezentand toate starile simultan)

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

α si β sunt amplitudini cuantice

- Starile sunt nedeterminate pana la masurare.

Sfera Bloch reprezinta toate starile posibile in **Superpozitie**



Sisteme cuantice binare

Putem reprezenta un sistem binar (biți) mai degrabă în stări cuantice decât în stări clasice?

Putem găsi un sistem mecanic cuantic în care să putem folosi două „stări” pentru a reprezenta numerele noastre binare? DA

Qubiții pot fi transportați ca atomi, ioni, fotoni sau electroni **împreuna cu** dispozitivele lor de control cu care lucrează împreună pentru a acționa ca memorie de computer și procesor.



Entangled qubits allow multiple numbers to be represented simultaneously. $|000\rangle$

Entangled = incalcit, incurcat, combinat

$|000\rangle$ = 3 qubiți combinati

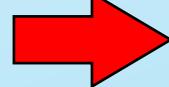
BIT VS QUBIT

* Bit clasic

* 0 or 1

* 101

Qubit

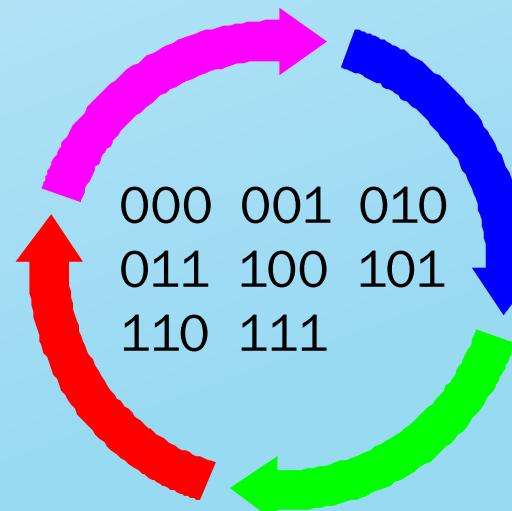
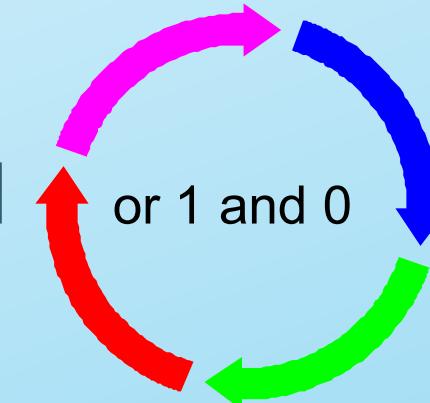


0 or 1

or 1 and 0



000 001 010
011 100 101
110 111



Care este diferență?

n biti pot stoca **unul din** 2^n numere la un moment dat

n qubiti pot stoca **toate** 2^n numere în același timp

AVANTAJELE QUBITILOR

Prin adaugare de qubiti capacitatea de stocare creste exponential

Se pot face operatii asupra tuturor superpozitiilor simultan...
(ca in cazul calculelor paralele)

O operatie matematica asupra a 2^n numere codificate in n biti necesita
 2^n pasi sau 2^n procesoare lucrand in paralel

Aceiasi operatie asupra 2^n numere codificate in n qubiti necesita **1** pas

Aceasta face ca problemele complexe sa fie rezolvate mult mai usor.

HOW QUANTUM COMPUTERS WORK

Today's Computers

Turing Machine- theoretical device that consists of tape of unlimited length that is divided into little squares. Each square can either hold a symbol (1 or 0) or be left blank.

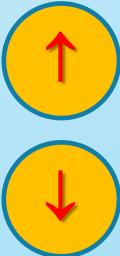
Today's computers work by manipulating bits that exist in one of two states: a 0 or a 1.

1 and 0's are carried and turned on by states of electrical current

• Quantum Computers

- Quantum computers aren't limited to two states like today's computers. They encode information as quantum bits, or **qubits**, which can exist in **superposition**.
- **Superposition**- quantum computers can represent both 0 and 1 as well as **everything in between** at the same time.
- **Qubits** can be carried as atoms, ions, photons or electrons and their respective control devices that are working together to act as computer memory and a processor.
- **Basically, a quantum computer can work on a million computations at once, while your desktop PC works on one.**

SUPERPOZITIA NUMERELOR



Sa presupunem o memorie cu 2 qubiți formata din doi atomi.

$$\Psi = 1/\sqrt{2}(|\uparrow\rangle + |\downarrow\rangle)$$

(primul atom este marcat cu \uparrow iar al doilea atom este marcat cu \downarrow)

poate fi scris ca:

$$\Psi = 1/\sqrt{2}(|0\rangle + |1\rangle)$$

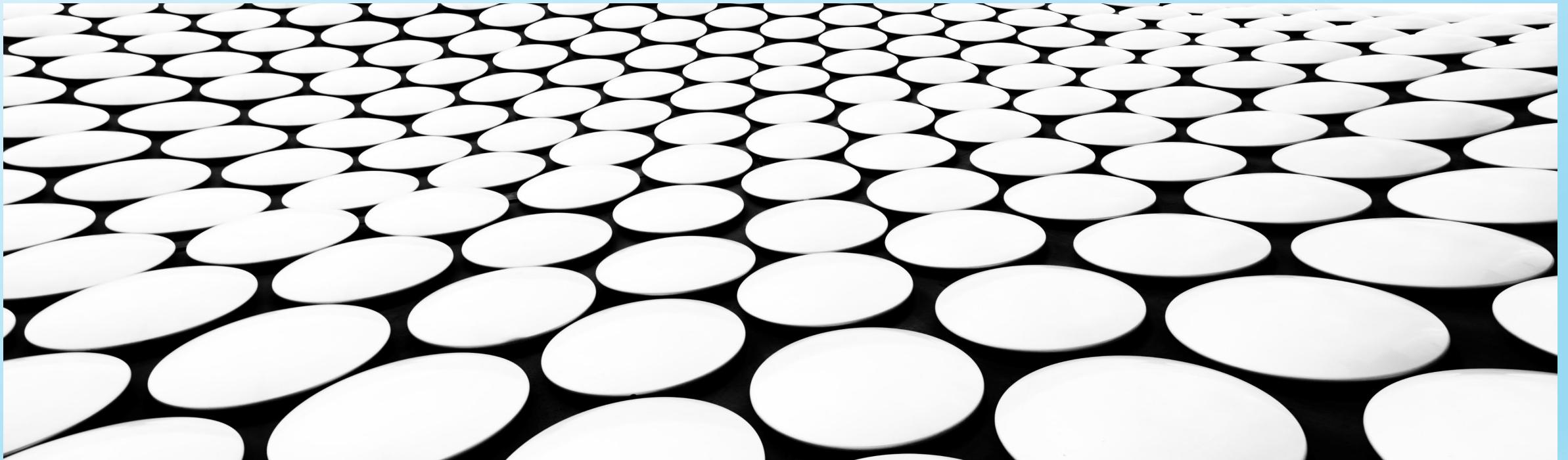
Aceasta poate fi expandată ca

$$|00\rangle + |01\rangle + |10\rangle + |11\rangle$$

Tocmai am scris o reprezentare binară în care numerele 0, 1, 2 și 3 sunt reprezentate simultan. Acest lucru este impresionant în comparație cu un computer clasic.

ARHITECTURA SISTEMELOR DE CALCUL

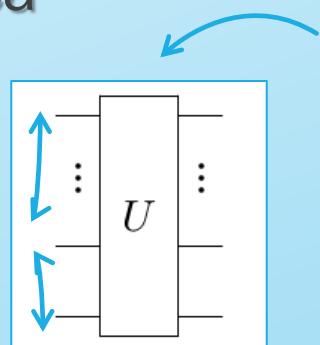
UB, FMI, CTI, ANUL III, 2022-2023



PORTI CUANTICE

- Actioneaza in mod similar
portile **clasice** manipuleaza cativa **biti** in acelasi timp,
portile **cuantice** manipuleaza cativa **qubiti** in acelasi timp
- Uzual sunt reprezentate prin **matrici unitare**
- Reprezentarea circuitistica

Firele descriu qubiți



*casetele și simbolurile diferite
descriu operații asupra qubiților*

*... moștenirea calculului clasic –
este mai bine să ne gândim la qubiți ca la particule și
la porți ca la procese fizice aplicate acestor particule*

Porti cuantice

$$|\Psi(t)\rangle = U |\Psi(0)\rangle \quad U U^* = 1$$

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad X|0\rangle = |1\rangle \quad X|1\rangle = |0\rangle$$

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad Z|0\rangle = |0\rangle \quad Z|1\rangle = -|1\rangle$$

$$\text{Hadamard} = H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

POARTA PAULI-X

Actioneaza asupra unui singur qubit

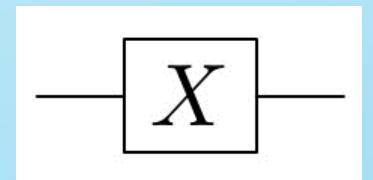
Notation Dirac

$$|0\rangle \rightarrow |1\rangle, \quad |1\rangle \rightarrow |0\rangle$$

Reprezentarea Matriceala

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Reprezentarea Circuitistica



Actionand asupra starilor pure devine o poarta **NOT**clasica

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \xrightarrow{\hspace{2cm}} \quad X \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \cdot 1 + 1 \cdot 0 \\ 1 \cdot 1 + 0 \cdot 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \xrightarrow{\hspace{2cm}} \quad X \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \cdot 0 + 1 \cdot 1 \\ 1 \cdot 0 + 0 \cdot 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$$

Notatia Dirac ...

$$X|0\rangle = |1\rangle$$

$$X|1\rangle = |0\rangle \quad \dots este evident mai convenabila pentru calcul$$

Stare pura: stare care nu este formata prin superpozitie

POARTA PAULI-X

- Actionand asupra unei stari generale a qubitului

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$X|\psi\rangle = \alpha|1\rangle + \beta|0\rangle = \beta|0\rangle + \alpha|1\rangle$$

- Este propriul său invers

$$XX = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

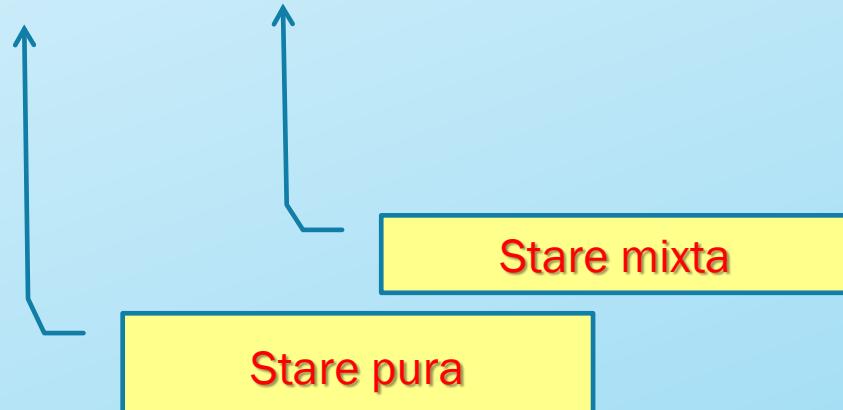
POARTA HADAMARD

- Actioneaza asupra unui singur qubit

Notatia Dirac

$$|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

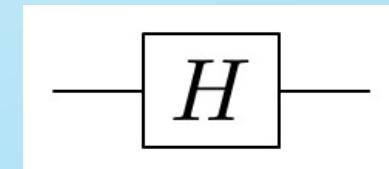
$$|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$



Matricea Unitara

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Reprezentarea Circuitistica



... evident, nici un echivalent clasic

- Una dintre cele mai importante porți pentru calculul cuantic

POARTA HADAMARD

Un exemplu interesant

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$
$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

Acționând asupra stărilor pure...
...oferă o suprapunere echilibrată...

$$|\alpha_0|^2 = \frac{1}{2} \quad |\alpha_1|^2 = \frac{1}{2}$$

... ambele stări, dacă sunt măsurate,
dau fie 0, fie 1 cu probabilitate egală

POARTA HADAMARD

Aplicând o altă poartă Hadamard

la primul rezultat

$$H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) + \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$H\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = |0\rangle$$

iar la al doilea rezultat

$$H\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) - \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$H\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) = |1\rangle$$

In acest caz se geneteaza stari pure

POARTA HADAMARD

Ambele stări dau probabilități egale atunci când sunt măsurate...

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad \begin{matrix} \swarrow & \searrow \\ & \end{matrix} \quad \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

*...dar când se aplică transformarea Hadamard,
aceasta produce două stări diferite*

*Exemplul oferă un răspuns la întrebarea:
de ce starea sistemului trebuie specificată cu amplitudini
complexe și nu poate fi specificată doar cu probabilități*

POARTA PAULI-Y

- Actioneaza **asupra unui singur qubit**

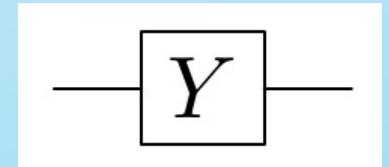
Notatia Dirac

$$|0\rangle \rightarrow i|1\rangle, \quad |1\rangle \rightarrow -i|0\rangle$$

Reprezentarea Matriceala

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

Reprezentarea Circuitistica



...o altă poartă fără echivalent clasic

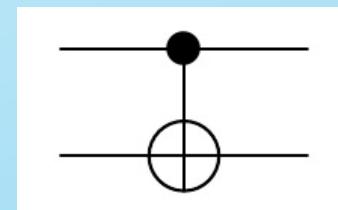
POARTA CNOT

- Poarta **NU Controlata**
- Acționează asupra doi qubiți

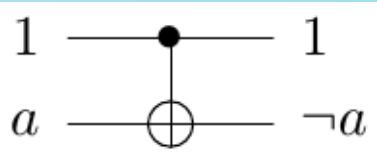
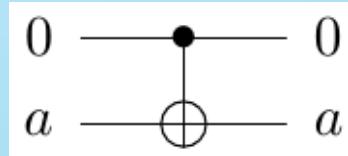
Reprezentarea Matriceala

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Reprezentarea Circuitistica



- Funcționarea clasică a portii



POARTA CNOT

- Exemplu de actionare la o suprapunere de stari

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

$$|00\rangle \rightarrow |00\rangle, \quad |01\rangle \rightarrow |01\rangle, \quad |10\rangle \rightarrow |11\rangle, \quad |11\rangle \rightarrow |10\rangle$$



$$CNOT|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$$

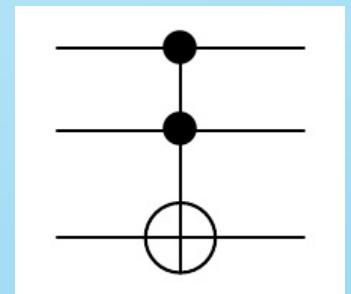
POARTA TOFFOLI

- Numit și **NU controlat controlat**
- Acționează pe trei qubiți

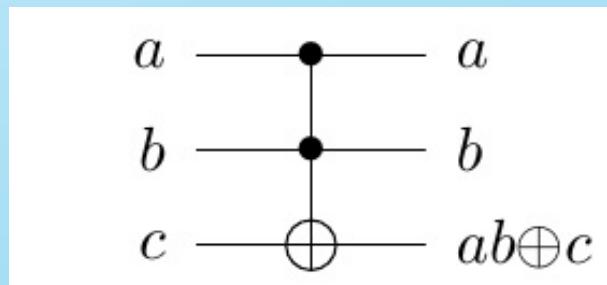
Reprezentarea Matriceala

$$TOFFOLI = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Reprezentarea Circuitistica

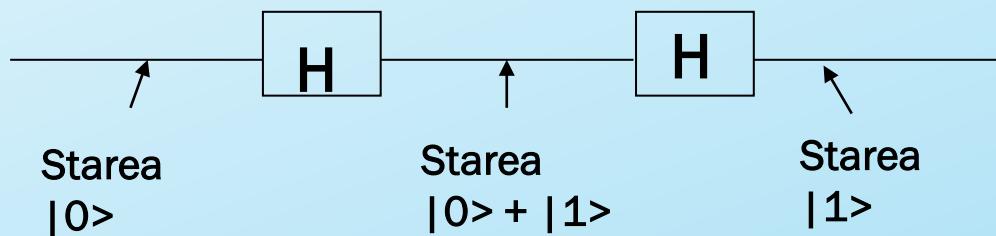


- Funcționarea clasică a porții



POARTA HADAMARD

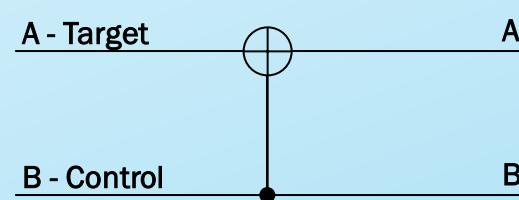
Folosit pentru a pune qubiți în superpozitie.



Nota: Două porți Hadamard utilizate
succesiv pot fi folosite ca poartă NOT

cunoscută și ca rădăcină pătrată a porții NOT.

- Dacă bitul de pe linia de control este 1, inversează bitul de pe linia țintă.



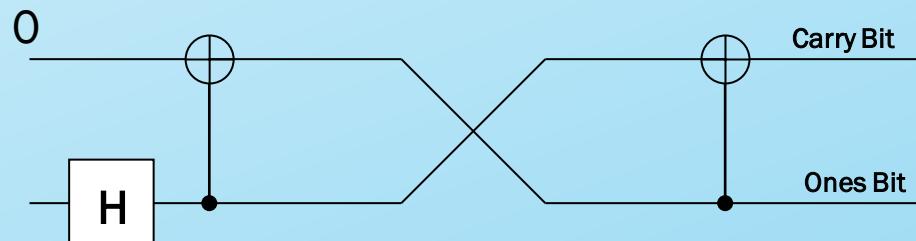
Input		Output	
A	B	A'	B'
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

Note: Poarta CN are un comportament similar cu poarta XOR cu unele informații suplimentare pentru a o face reversibilă.

EXAMPLE OPERATION - MULTIPLICATION BY 2

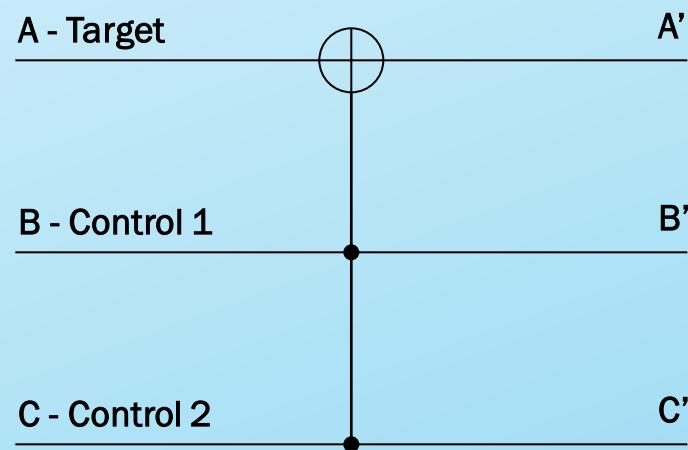
- Putem construi un circuit logic reversibil pentru a calcula înmulțirea cu 2 folosind porti CN aranjate în felul următor:

Input		Output	
Carry Bit	Ones Bit	Carry Bit	Ones Bit
0	0	0	0
0	1	1	0



CONTROLLED CONTROLLED NOT (CCN)

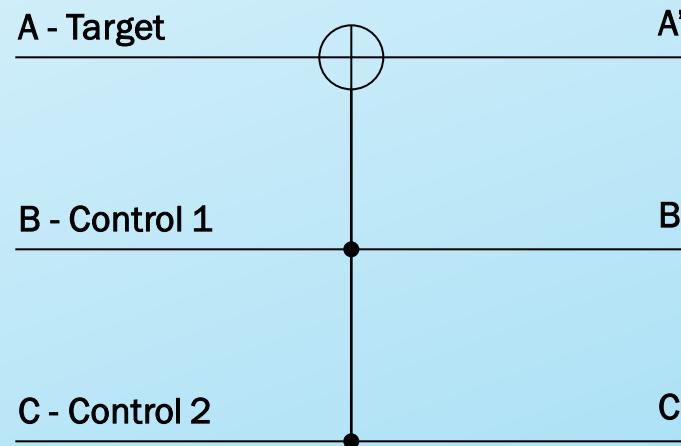
Dacă biții de pe ambele linii de control sunt 1, atunci bitul țintă este inversat.



Input			Output		
A	B	C	A'	B'	C'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	0	1	1

A UNIVERSAL QUANTUM COMPUTER

- Poarta CCN s-a dovedit a fi o poartă logică reversibilă universală, deoarece poate fi folosită ca poartă NAND.

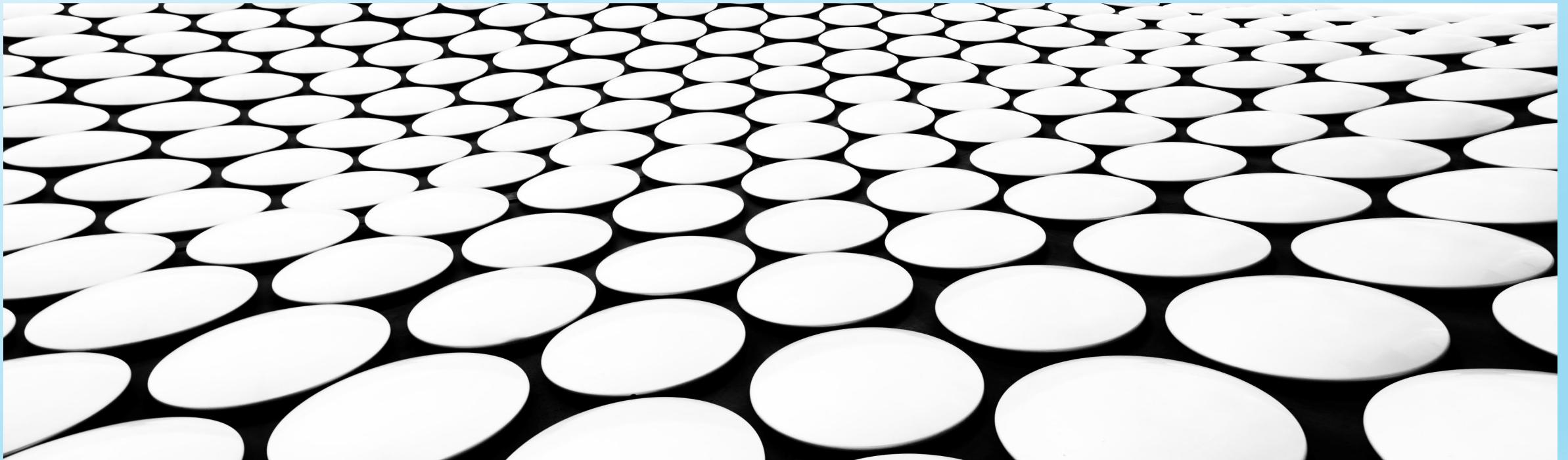


Input			Output		
A	B	C	A'	B'	C'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	0	1	1

Când intrarea noastră ţintă este 1, ieşirea noastră ţintă este rezultatul unui NAND aupra B și C.

ARHITECTURA SISTEMELOR DE CALCUL

UB, FMI, CTI, ANUL III, 2022-2023

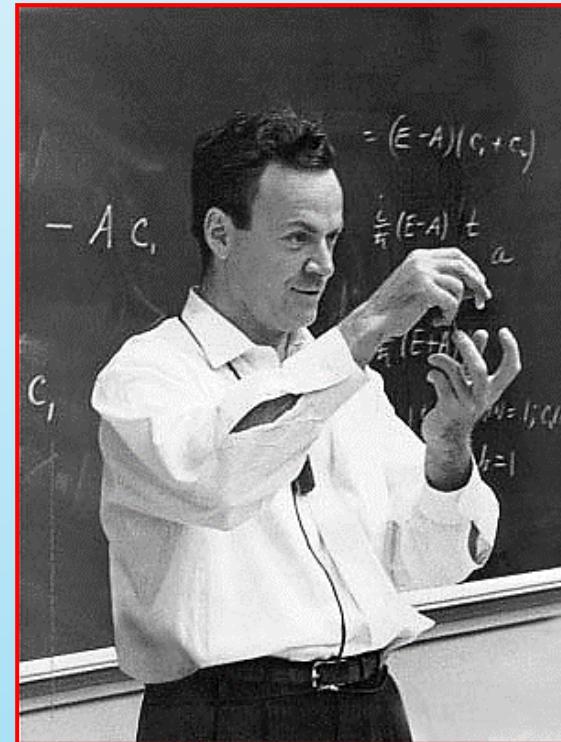


CALCULATOARE CUANTICE

Partea II

ISTORIA CALCULATOARELOR CUANTICE

- 1982 - **Feynman** a propus ideea de a crea mașini bazate pe legile mecanicii cuantice în loc de legile fizicii clasice.



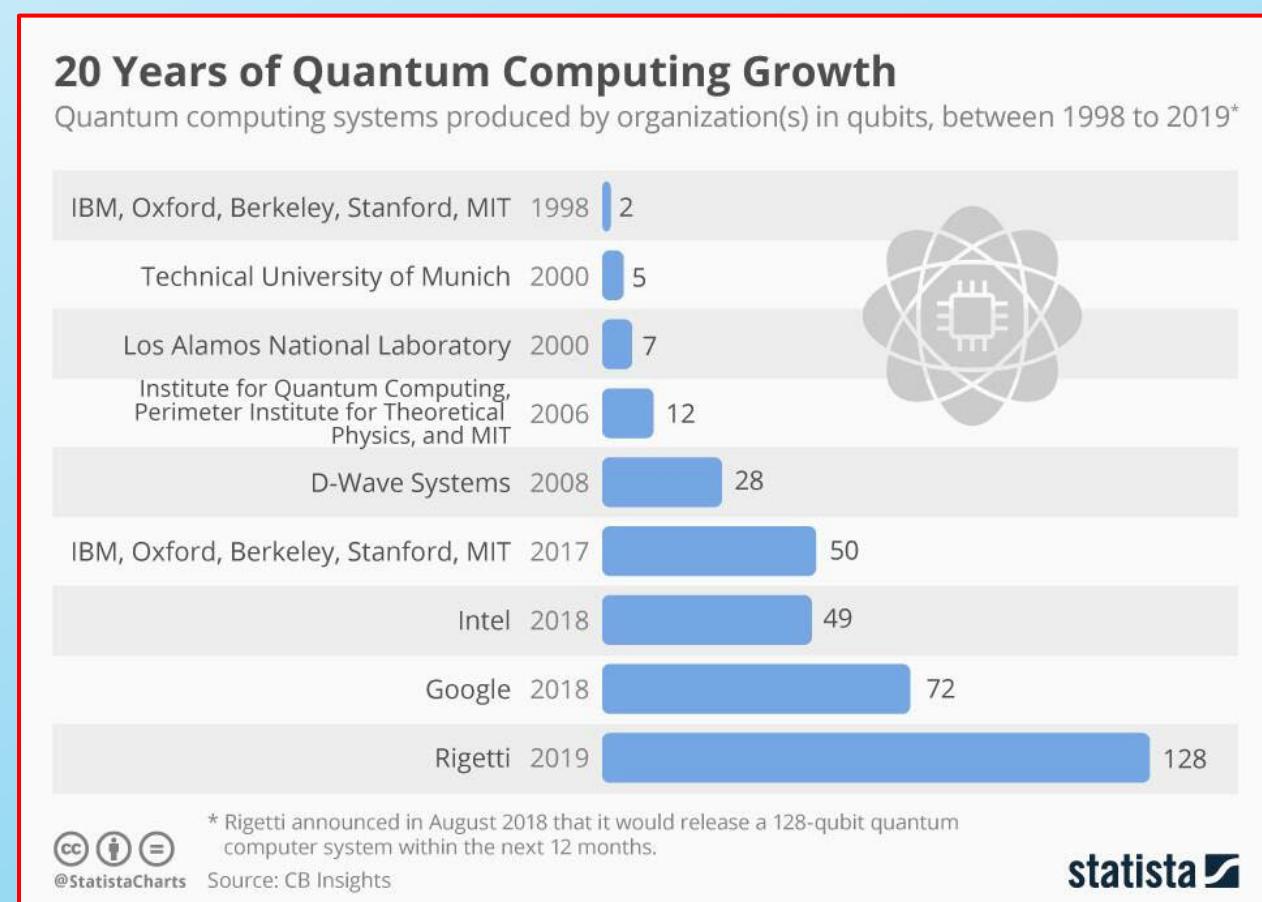
- 1985 - David Deutsch a dezvoltat **masina Turing cuantica**, arătând că circuitele cuantice sunt universale.
- 1994 - Peter Shor a venit cu un **algoritm cuantic** pentru a factoriza numere foarte mari.
- 1997 - Lov Grover dezvolta un algoritm cuantic de cautare cu o complexitate de $O(\sqrt{N})$

- **1973** - Alexander Holevo publishes paper showing that n qubits cannot carry more than n classical bits of information.
- 1976 - Polish mathematical physicist Roman Ingarden shows that Shannon information theory cannot directly be generalized to the quantum case.
- 1981 - *Richard Feynman determines that it is impossible to efficiently simulate a evolution of a quantum system on a classical computer.*
- **1985 - David Deutsch of the University of Oxford, describes the first universal quantum computer.**
- 1993 - Dan Simon, at Universite de Montreal, invents an oracle problem for which quantum computer would be exponentially faster than conventional computer. This algorithm introduced the main ideas which were then developed in Peter Shor's factoring algorithm.
- 1994 - Peter Shor, at AT&T's Bell Labs discovers *algorithm* to allow quantum computers to *factor large integers quickly*. Shor's algorithm could theoretically break many of the cryptosystems in use today.
- **1995** - Shor proposes the first scheme for quantum error correction.
- 1996 - Lov Grover, at Bell Labs, invents quantum database search algorithm.
- **1997 - David Cory, A.F. Fahmy, Timothy Havel, Neil Gershenfeld and Isaac Chuang publish the first papers on quantum computers based on bulk spin resonance, or thermal ensembles. Computers are actually a single, small molecule, storing qubits in the spin of protons and neutrons. Trillions of trillions of these can float in a cup of water.**
- **1998 - First working 2-qubit NMR computer** demonstrated at University of California, Berkeley.
- 1999 - First working 3-qubit NMR computer demonstrated at IBM's Almaden Research Center. First execution of Grover's algorithm.
- 2000 - First working 5-qubit NMR computer demonstrated at IBM's Almaden Research Center.

- 2001 - First working **7-qubit NMR** computer demonstrated at IBM's Almaden Research Center. First execution of Shor's algorithm. The number 15 was factored using 1018 identical molecules, each containing 7 atoms.

2015 Cambridge Quantum Computing releases **tket**, which could be the first operating system for quantum computing. This enables classical computers interface to quantum computers.

- Oct. 2019. Google claims to have achieved quantum supremacy with a **53-qubit programmable superconducting processor**. Named Sycamore, it's based on quantum logic gates. The machine completes a benchmark test in 200 seconds, what a classical supercomputer would take 10,000 years.
- However, IBM claims a supercomputer with more disk storage could solve the problem in 2.5 days.
- In **August 2020**, 12 qubits of Sycamore are used to simulate a chemical reaction.
- **Mar. 2020** Honeywell reports on a demonstration of a quantum computer architecture based on trapped-ion QCCD (Quantum Charge-Coupled Device).



Simulatoare:

<https://algassert.com/quirk>

<https://quantum-circuit.com/>

<https://qiskit.org/>

Documentatie:

<https://quantum-computing.ibm.com/docs/>

Joc:

<https://quantumgame.io/>

Limbaje de programare:

QCL (<http://tph.tuwien.ac.at/~oemer/qcl.html>)

Arhitectura calculatoarelor cuantice

O arhitectură a unui sistem este un model teoretic care arată structura, comportamentul și componentele sistemului. Prin ea sunt descompuse comportamentele complexe ale sistemelor într-un set gestionabil de operațiuni.

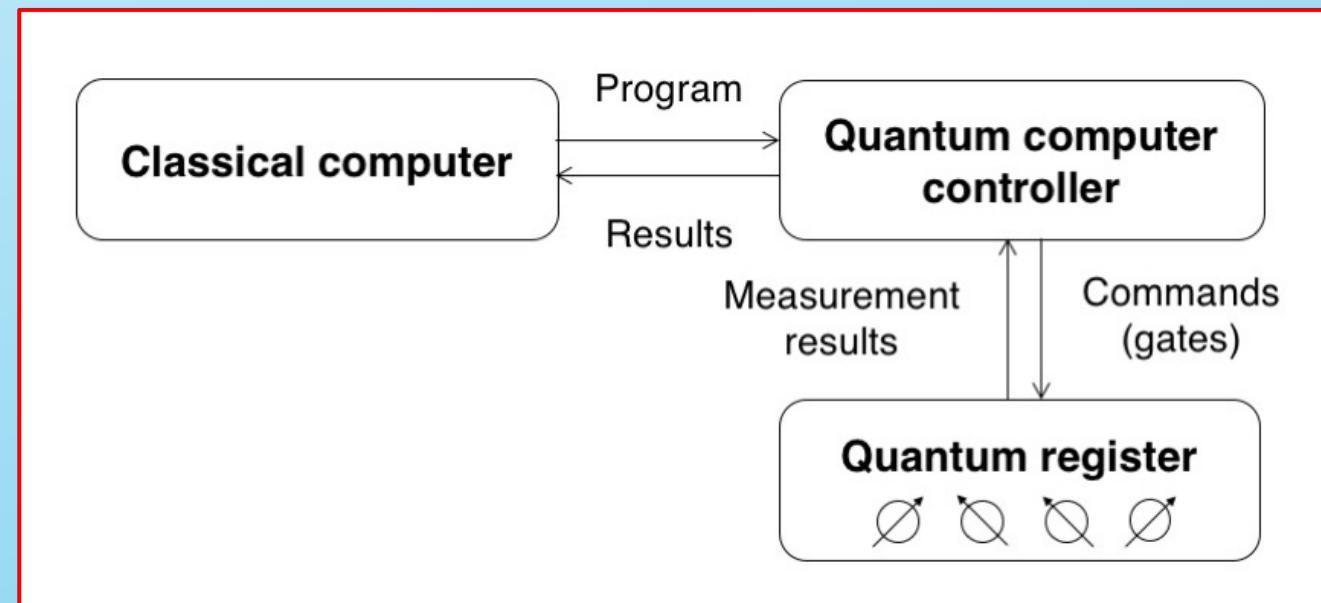
Un computer cuantic are atât părți clasice, cât și cuantice.

Modelul de bază al calculatoarelor cuantice

Atât computerul clasic, cât și cel cuantic constau în esență din trei părți :

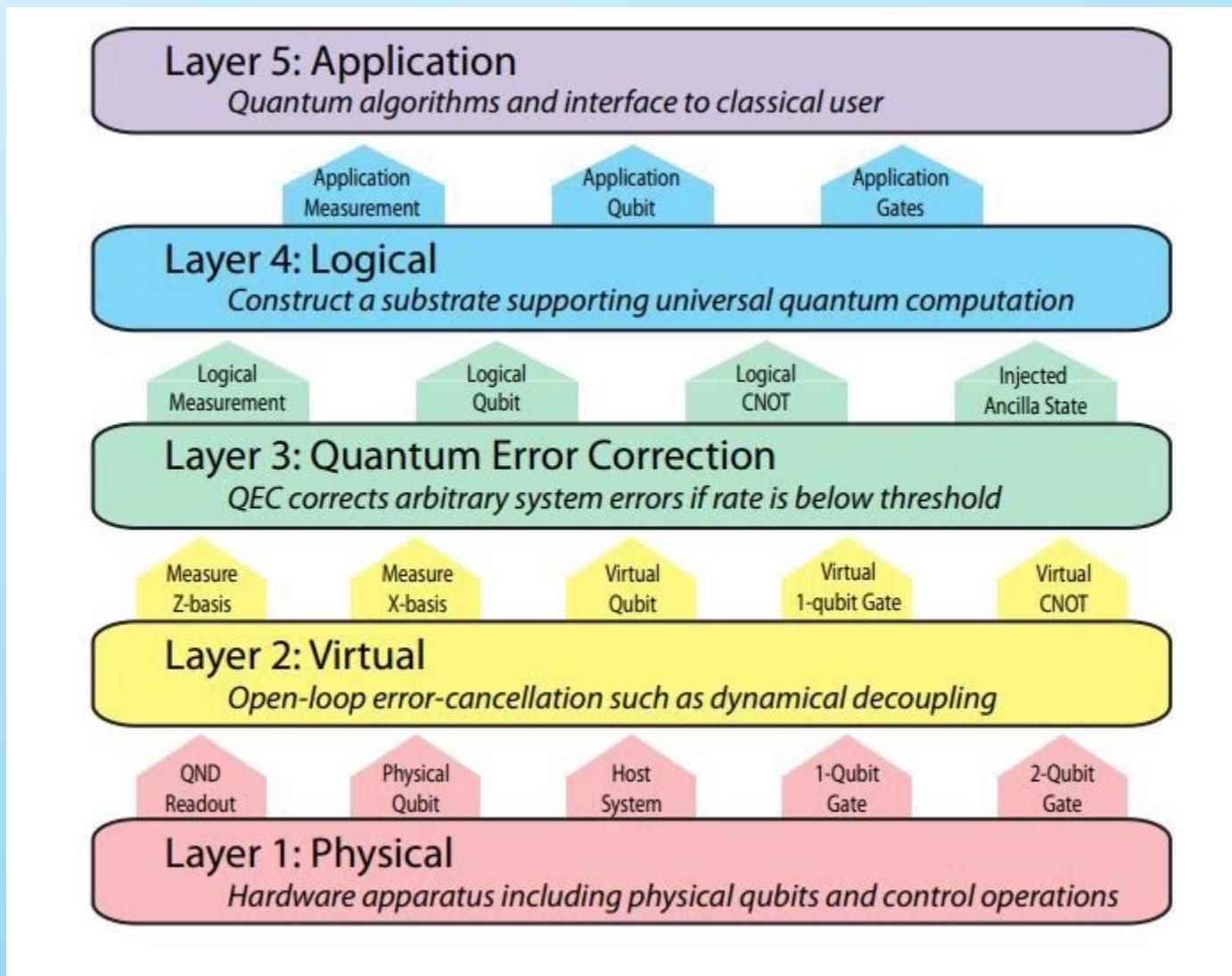
- **Memoria** - Conține/stocheaza starea curentă a mașinii.
- **Un procesor sau un controler** - Efectuează operații elementare asupra stării mașinii.
- **Dispozitiv de intrare/ieșire** - Face posibilă definirea stării initiale și obținerea stării finale de calcul.

- **Registrii cuantici** sunt memoria calculatoarelor cuantice.
Dețin date cuantice pentru algoritm.
- **Porțile cuantice** sunt echivalentul instructiunilor.
- **Controler de calculator** deține programul și le spune dispozitivelor care controlează fiecare qubit să efectueze acțiuni conform instrucțiunilor.



Arhitectură stratificată

- Arhitectura computerului cuantic constă din cinci straturi, în care fiecare strat are propriul set de sarcini sau funcții.
- Pentru a executa o operație, un strat trebuie să primească comanda sau instrucțiunile de la stratul de mai jos și să le proceseze în consecință.

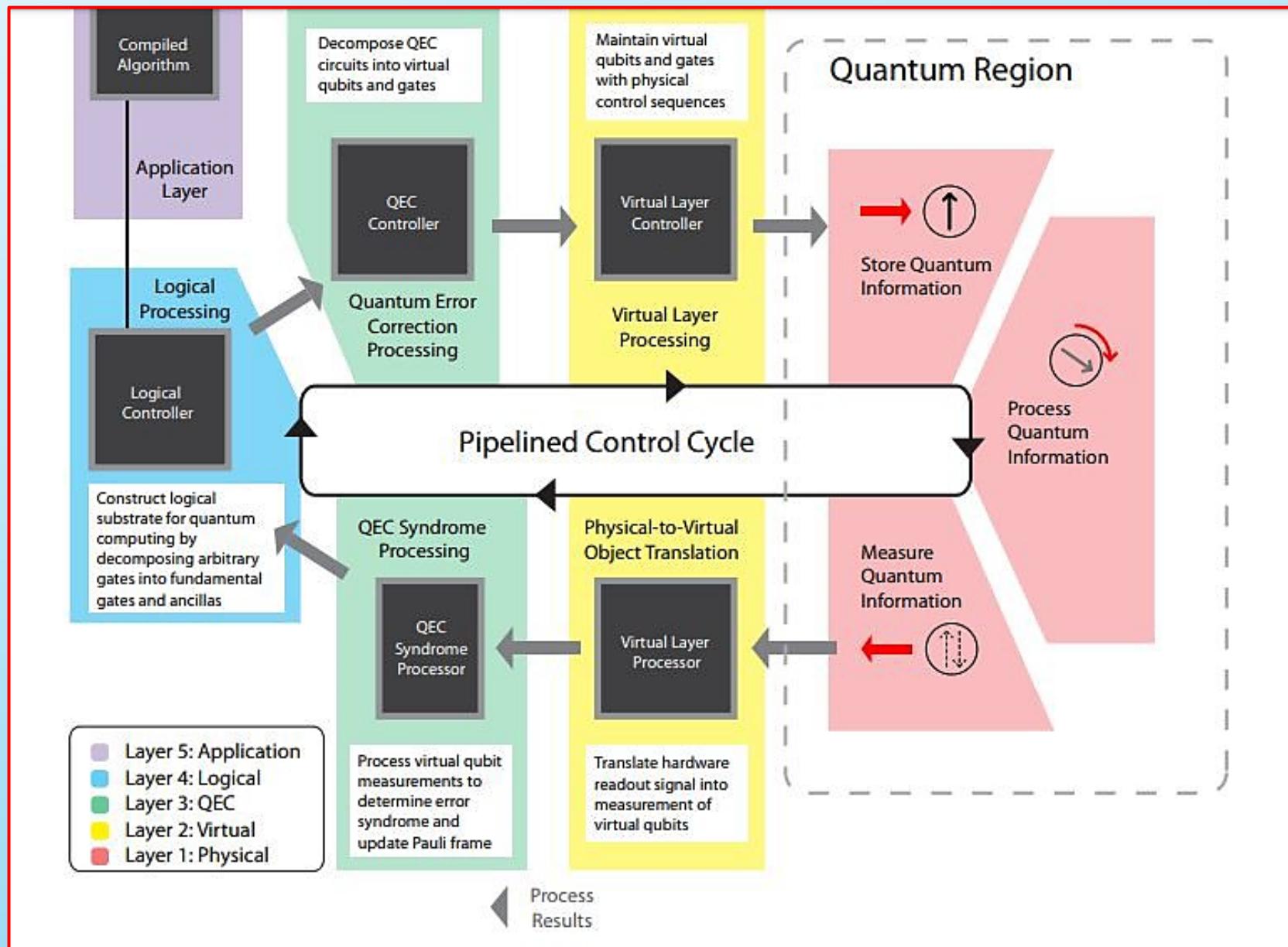


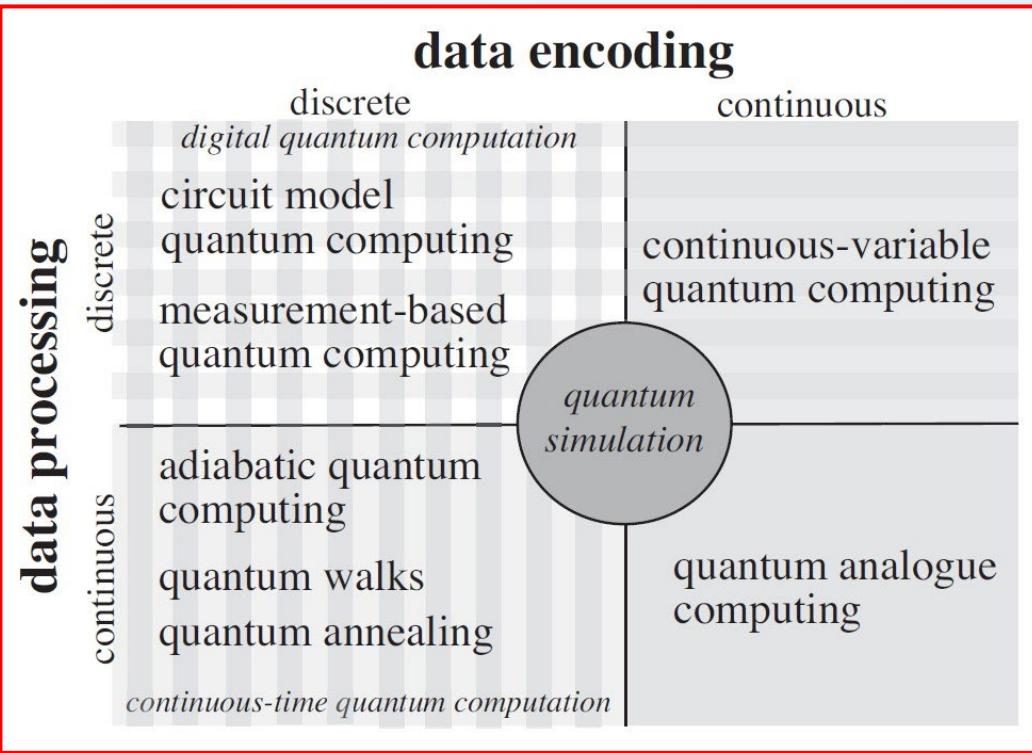
Ciclul de control primar definește comportamentul dinamic al computerului cuantic în această arhitectură, deoarece toate operațiunile trebuie să interacționeze cu această buclă.

Scopul principal al ciclului de control este implementarea cu succes a **corectării erorilor cuantice**.

Calculatorul cuantic trebuie să funcționeze suficient de rapid pentru a corecta erorile; totuși, unele operațiuni de control implică în mod necesar întârzieri, așa că acest ciclu nu emite pur și simplu o singură comandă și așteaptă rezultatul înainte de a continua - conducta este esențială.

Straturile 1 până la 4 interacționează în buclă, în timp ce stratul Aplicație interacționează numai cu stratul logic, deoarece este agnostic cu privire la designul de bază al computerului cuantic.





Analog quantum computers (including quantum annealer, adiabatic quantum computers, and direct quantum simulation).

These systems operate using **coherent manipulation** of the qubits, changing the analog values of the Hamiltonian.

It does not use quantum gates.

Fully error-corrected gate-based quantum computers.

Like NISQs, these are gate-based systems that operate on qubits.

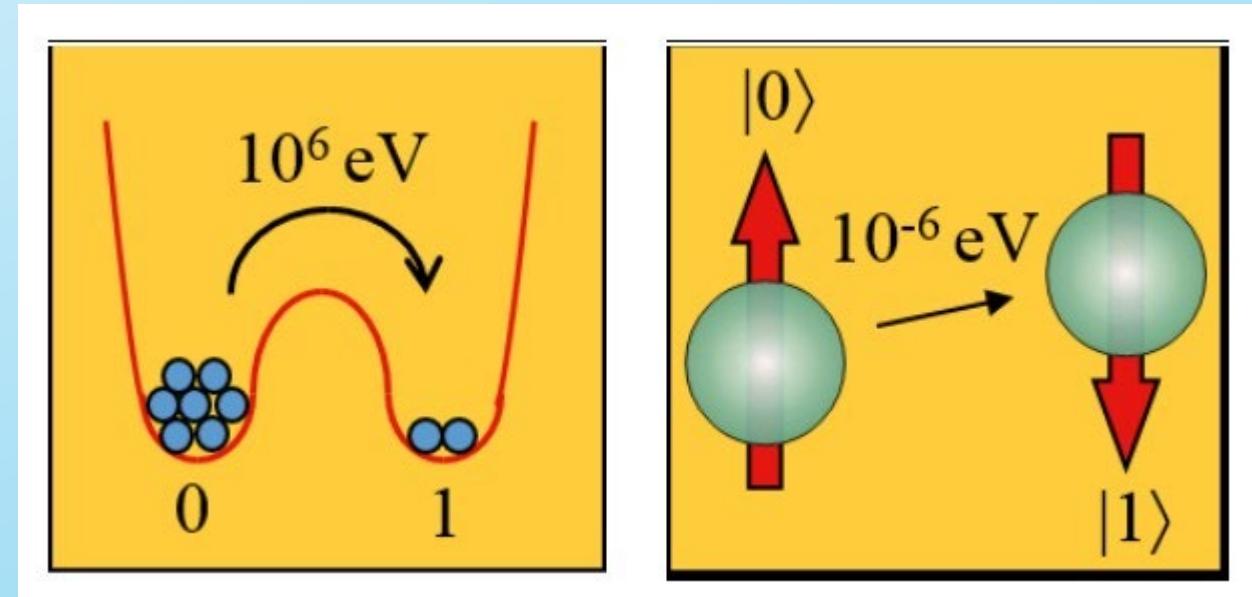
Still, these systems are complex and implement quantum error correction to eliminate the negative effects of system noise (including errors introduced by imperfect control signals or device fabrication or unintended coupling of qubits to each other or the environment).

These systems enable reductions in error probability rates sufficient that the computer is reliable for all computations.

Fully error-corrected gate-based quantum computers are expected to scale to thousands of logical qubits, enabling massive computational capabilities.

Decoerență cuantică este o sursă majoră de erori atunci când se lucrează cu computere cuantice. Orice interacțiune a qubiților cu mediul exterior în moduri care perturbă comportamentul lor cuantic va duce la decoerență.

Calculatoarele cuantice de astăzi pot funcționa doar pentru perioade scurte de timp (adesea mai puțin de o secundă, și chiar și design-urile bune funcționează doar câteva secunde) înainte ca decoerență să interfereze cu funcționarea lor.



Calculatoarele cuantice bazate pe porți pot avea diverse implementari fizice.

Cu toate acestea, orice realizare trebuie să îndeplinească criteriile DiVincenzo:

1. Un sistem fizic **scalabil** cu qubiți bine caracterizați
2. Abilitatea de a inițializa starea qubiților într-o stare cuantică simplă care poate fi reprodusă în mod fiabil cu variabilitate scăzută
3. Timpi relativ **lungi** de decoerență
4. Un set „universal” de porți cuantice
5. O capabilitate de măsurare specifică qubitului

Calculatoarele cuantice **analogice** au nevoie de toate cele de mai sus, cu excepția elementului 4, deoarece nu folosesc porți pentru a-și exprima algoritmii.

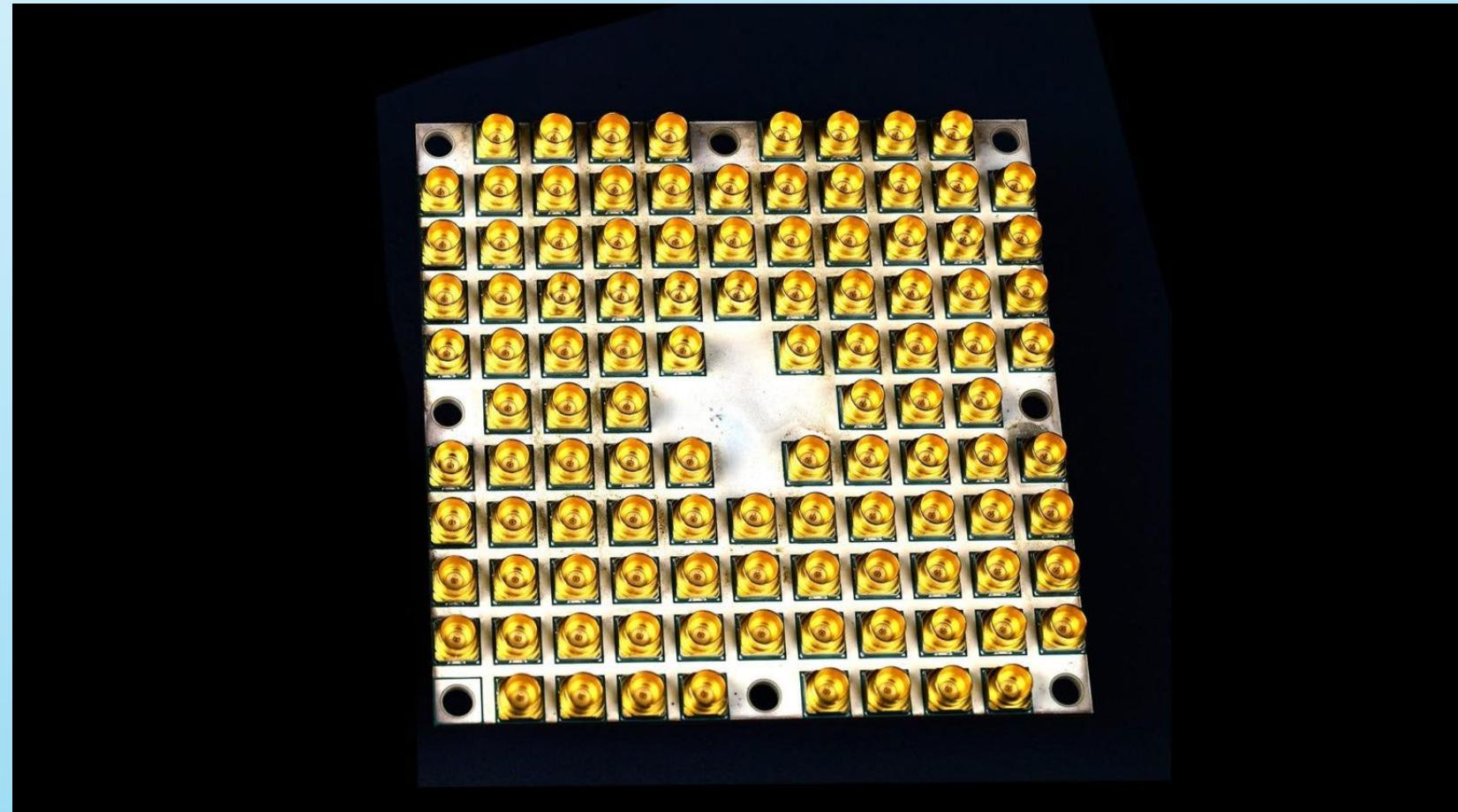
Cu toate acestea, decoerența joacă un rol foarte diferit în calculul cuantic analogic decât în modelul de poartă. De exemplu, o oarecare decoerență este tolerabilă în “recoacerea cuantică” și un anumit interval de relaxare energetică este necesar pentru ca “recoacerea cuantică” să reușească.

Calculatoarele cuantice au fost implementate folosind modele NISQ cuantice analogice și digitale.

Sistemele complet corectate de erori sunt mai dificil de realizat și sunt încă în curs de dezvoltare.

Research at Intel Labs has led directly to the development of Tangle Lake, a superconducting quantum processor that incorporates 49 qubits in a package manufactured at Intel's 300-millimeter fabrication facility in Hillsboro, Oregon.

This device represents **the third-generation quantum processors** produced by Intel, scaling upward from 17 qubits in its predecessor.



CANDIDATES FOR QUANTUM COMPUTERS

- Superconductor-based quantum computers
(including SQUID-based quantum computers)
- Ion trap-based quantum computers
- "Nuclear magnetic resonance on molecules in solution"-based
- "Quantum dot on surface"-based
- "Laser acting on floating ions (in vacuum)" -based (Ion trapping)
- "Cavity quantum electrodynamics" (CQED)-based
- Molecular magnet-based
- Fullerene-based ESR quantum computer
- Solid state NMR Kane quantum computer

QUANTUM COMPUTING PROBLEMS

- Current technology
 - ≈ 40 Qubit operating machine needed to rival current classical equivalents.
- Errors
 - **Decoherence** - the tendency of a quantum computer to decay from a given quantum state into an incoherent state as it interacts with the environment.
 - Interactions are unavoidable and induce breakdown of information stored in the quantum computer resulting in computation errors.
 - **Error rates** are typically proportional to the ratio of operating time to decoherence time
 - operations must be completed much quicker than the decoherence time.

Algoritmi cuantici

ALGORITMUL SHOR

Algoritm de factorizare a numerelor intregi

Exemplu de factorizare a unui număr întreg impar N (să alegem 15):

1. Alegem un intreg q astfel incat $N^2 < q < 2N^2$ **hai sa alegem 256**
2. Alegem x arbitrar astfel incat $\text{cmmdc}(x, N) = 1$ **hai sa alegem 7**
3. Creem doi registri cuantici (acesti registri trebuie sa fie “entangled” astfel incat colapsarea registru lui de intrare sa corespunda cu colapsarea registrului de iesire)
 - **Registrul de intrare:** trebuie sa contine suficienti qubiti pentru a stoca numere la fel de mari ca $q-1$.
pana la 255, este nevoie de 8 qubiti
 - **Registrul de iesire:** trebuie sa contine suficienti qubiti pentru a stoca numere la fel de mari ca $N-1$.
pana la 14, este nevoie de 4 qubiti

SHOR'S ALGORITHM - PREPARING DATA

4. Incarcam the registrul de intrare cu o superpozitie, de ponderi egale, a tuturor intregilor from 0 to $q-1$. **0 to 255**
5. Incarcam registrul de iesire cu zero peste tot.

The starea totala a sistemului in acest punct va fi:

$$\frac{1}{\sqrt{256}} \sum_{a=0}^{255} |a, 000\rangle$$

The diagram illustrates the state of the system as a superposition of two registers. On the left, there is an 'Input Register' represented by a vertical line. Above it, the expression $\frac{1}{\sqrt{256}} \sum_{a=0}^{255}$ is shown with a bracket underneath, indicating the state of the input register. On the right, there is an 'Output Register' represented by a vertical line. A bracket above the output register indicates its state. The entire expression $|a, 000\rangle$ is positioned between the two registers, with a vertical line connecting the two brackets.

Nota: virgula semnifica faptul ca registrii sunt “entangled”

SHOR'S ALGORITHM - MODULAR ARITHMETIC

6. Aplicam transformarea $x^a \text{ mod } N$ fiecarui numar din registrul de intrare, stocand rezultatul rezultatul fiecarui calcul in registrul de iesire.

Input Register	$7 \text{ Mod } 15^a$	Output Register
$ 0\rangle$	$7 \text{ Mod } 15^0$	1
$ 1\rangle$	$7 \text{ Mod } 15^1$	7
$ 2\rangle$	$7 \text{ Mod } 15^2$	4
$ 3\rangle$	$7 \text{ Mod } 15^3$	13
$ 4\rangle$	$7 \text{ Mod } 15^4$	1
$ 5\rangle$	$7 \text{ Mod } 15^5$	7
$ 6\rangle$	$7 \text{ Mod } 15^6$	4
$ 7\rangle$	$7 \text{ Mod } 15^7$	13

Nota: folosim numere zecimale numai pentru simplitate

SHOR'S ALGORITHM - SUPERPOSITION COLLAPSE

7. Acum aplicam masuratori asupra registrului de iesire. Aceasta va colapsa superpozitia catre unul din rezultatele transformarii, sa numim aceasta valoare **c**.

Registrul de iesire va colapsa catre una din urmatoarele stari:

$|1\rangle$, $|4\rangle$, $|7\rangle$, or $|13\rangle$

De dragul exemplului, să alegem **$|1\rangle$**

SHOR'S ALGORITHM - ENTANGLEMENT

8. Deoarece registri sunt “entangled”, masurind registrul de iesire, ca efect va apărea colapsarea of partiala a registrului de intrare într-o **superpozitie egala** a fiecarei stari intre 0 si $q-1$ care produce **c** (valoarea din registrul de iesire colapsat)

Deoarece registrul de iesire colapseaza catre $|1\rangle$, registrul de intrare va colapsa parcial catre:

$$\frac{1}{\sqrt{64}}|0\rangle + \frac{1}{\sqrt{64}}|4\rangle + \frac{1}{\sqrt{64}}|8\rangle + \frac{1}{\sqrt{64}}|12\rangle, \dots$$

Probabilitatile in acest caz sunt $\frac{1}{\sqrt{64}}$ deoarece registrul de intrare este acum intr-o superpozitie egala de 64 values (0, 4, 8, ... 252)

SHOR'S ALGORITHM - QFT

Acum aplicăm transformata Fourier cuantică (TFq) pe registrul de intrare parțial colapsat.

Transformata Fourier are efectul de a lua o stare $|a\rangle$ și de a o transforma într-o stare data de :

$$\frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} |c\rangle * e^{2\pi i ac / q}$$

SHOR'S ALGORITHM - QFT

$$\frac{1}{\sqrt{64}} \sum_{a \in A} |a\rangle, |1\rangle \xrightarrow{\text{QFT}} \frac{1}{\sqrt{256}} \sum_{c=0}^{255} |c\rangle * e^{2\pi i ac / 256}$$

Nota: A este multimea tuturor valorilor pentru care $7^a \bmod 15$ produce 1.
In cazul nostru $A = \{0, 4, 8, \dots, 252\}$

Astfel starea finala a registrului de intrare dupa aplicarea TFq este:

$$\frac{1}{\sqrt{64}} \sum_{a \in A} \frac{1}{\sqrt{256}} \sum_{c=0}^{255} |c\rangle * e^{2\pi i ac / 256}, |1\rangle$$

SHOR'S ALGORITHM - QFT

TF_q va genera, în esență, amplitudinile de probabilitate ca multipli întregi ai lui $q/4$ în cazul nostru $256/4$ sau 64 .

$$|0\rangle, |64\rangle, |128\rangle, |192\rangle, \dots$$

Deci nu mai avem o suprapunere egală de stări, amplitudinile de probabilitate ale stărilor de mai sus sunt acum mai mari decât celelalte stări din registrul nostru.

Măsurăm registrul și va colapsa cu mare probabilitate la unul dintre acești multipli de 64 , să numim această valoare p .

Cu cunoștințele noastre despre q și p , există metode de calculare a perioadei (o metodă este dezvoltarea fracției continue a raportului dintre q și p .)

SHOR'S ALGORITHM - THE FACTORS :)

10. Acum că avem perioada, factorii lui N pot fi determinați luând cel mai mare divizor comun al lui N față de $x^{(P/2)+1}$ și $x^{(P/2)-1}$.
În aceasta etapa calcul se va face pe un calculator clasic.

Calculam:

$$\text{cmmdc}(7^{4/2} + 1, 15) = 5$$

$$\text{cmmdc}(7^{4/2} - 1, 15) = 3$$

Am factorizat cu success 15.

SHOR'S ALGORITHM - PROBLEMS

- $T_F q$ poate oferi o perioada greșită.

Probabilitatea depinde de fapt de alegerea pentru q .

Cu cât este mai mare q , cu atât este mai mare probabilitatea de a găsi probabilitatea corectă.

- Perioada seriei ajunge să fie impara

Dacă apare oricare dintre aceste cazuri, ne întoarcem la început și alegem un nou x .

CONCLUZIE

In 2001, o masina cu 7 qubiți a fost construita si programata pentru a executa algoritmul Shor, factorizand cu success numarul 15.

