**2010** - 2011

# **P**rogramare **O**rientata **spre O**biecte
# (*Object-Oriented Programming*)

a.k.a. Programare Obiect-Orientata

Titular curs: Eduard-Cristian Popovici

Suport curs: ***http://electronica08.curs.ncit.pub.ro/****course/view.php?id=113*

Suport curs vechi: *http://discipline.elcom.pub.ro/POO-Java/* si
*http://electronica07.curs.ncit.pub.ro/course/view.php?id=132*

## Continut POO in Java

**1. Introducere in abordarea orientata spre obiecte (OO)**
- 1.1. Obiectul cursului si relatia cu alte cursuri
- **1.2. Evolutia catre abordarea OO**
- 1.3. Caracteristicile si principiile abordarii OO
- 1.4. Scurta recapitulare a programarii procedurale/structurate (introducere in limbajul Java)

**2. Orientarea spre obiecte in limbajul Java**
- 2.1. Obiecte si clase. Metode (operatii) si campuri (atribute)
- 2.2. Particularitati Java. Clase de biblioteca Java (de uz general)
- 2.3. Clase si relatii intre clase. Asociere, delegare, agregare, compunere
- 2.4. Generalizare, specializare si mostenire
- 2.5. Clase abstracte si interfete Java
- 2.6. Polimorfismul metodelor
- 2.7. Clase pentru interfete grafice (GUI) din biblioteca Java Swing

**3. Programarea la nivel socket cu Java (pe platforma Java SE)**
- 3.1. Clase pentru fluxuri de intrare-iesire (IO)
- 3.2. Introducere in Protocolul Internet (IP) si stiva de protocoale IP
- 3.3. Socketuri flux (TCP) Java.
- 3.4. Clase Java pentru programe multifilare. Servere TCP multifilare
- 3.5. Socketuri datagrama (UDP) Java

## Obiectul cursului si relatia cu alte cursuri

- **Programarea**
  - **Masinile programabile** (suportul programarii) – curs **CID** si **AMP** (sem II)
  - **Programele** de calcul (tinta programarii) – curs **PC** si **SDA** (anul 1)
  - Programarea ca **rezolvare de probleme –** curs **Inginerie *Software* (?)**
- **Orientarea spre Obiecte (OO)**
  - **Evolutia catre OO**
    - Masina programabila si **codul masina** – curs **CID** si **AMP**
    - Limbajele de **asamblare** – curs **AMP**
    - Limbajele de **nivel inalt** (pre-OO) – curs **PC** si **SDA**
    - Tipurile de date abstracte (**ADT**) – curs **SDA**
  - **Orientarea** spre **modelarea realitatii**, entitati bazate pe **responsabilitati** (roluri), **incapsulare duala**, **interfete** (specificare), componente ***black-box***, **servicii**, etc.

1. **Introducere in abordarea orientata spre obiecte (OO)**

   1.2. Evolutia catre abordarea OO

## Evolutia catre abordarea Orientata spre Obiecte (OO)

- **Evolutia catre OO**
  - Masina programabila si **codul masina** – curs **CID** si **AMP**
  - Limbajele de **asamblare** – curs **AMP**
  - Limbajele de **nivel inalt** (pre-OO) – curs **PC** si **SDA**
    - Modelare si **abstractizare** (I)
    - **Incapsulare**, modularizare si **ascunderea detaliilor**
    - Programare **structurata**
    - Programare **procedurala**
  - Tipurile de date abstracte (**ADT**) – curs **SDA**
    - Incapsularea **duala**
    - Modelare si **abstractizare** (I)

## Orientarea spre Obiecte (OO)

- **Evolutia catre OO**
  - Masina programabila si **codul masina**
    - curs **CID** si **AMP** (sem II)

## Masina programabila si codul masina

**Arhitectura masinilor de calcul**

– deriva din cea creata de matematicianul **von Neumann** (1903 – 1957)

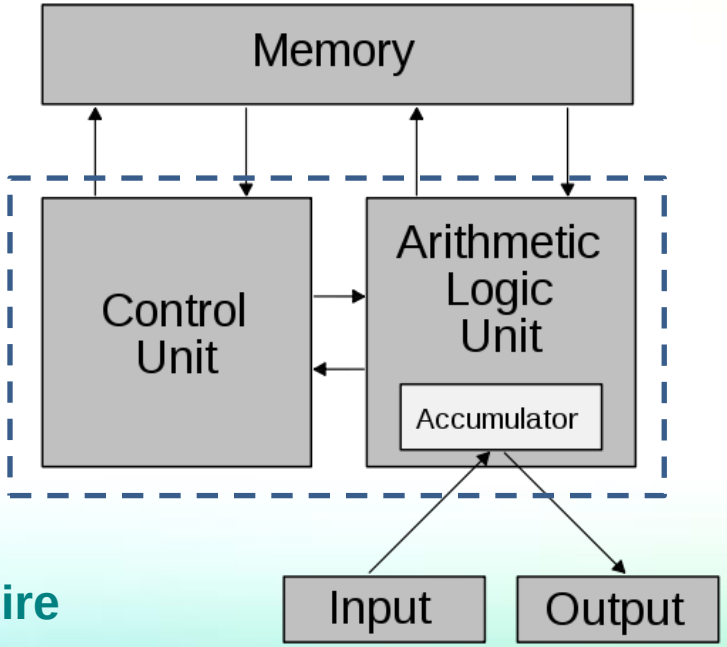
– este un **model de proiectare** pentru

– **masini de calcul digital** cu **programe stocate** in **memorie**

**Memorie principala** — **PM** — **stocare**

**Procesor** — **CPU** — **prelucrare**

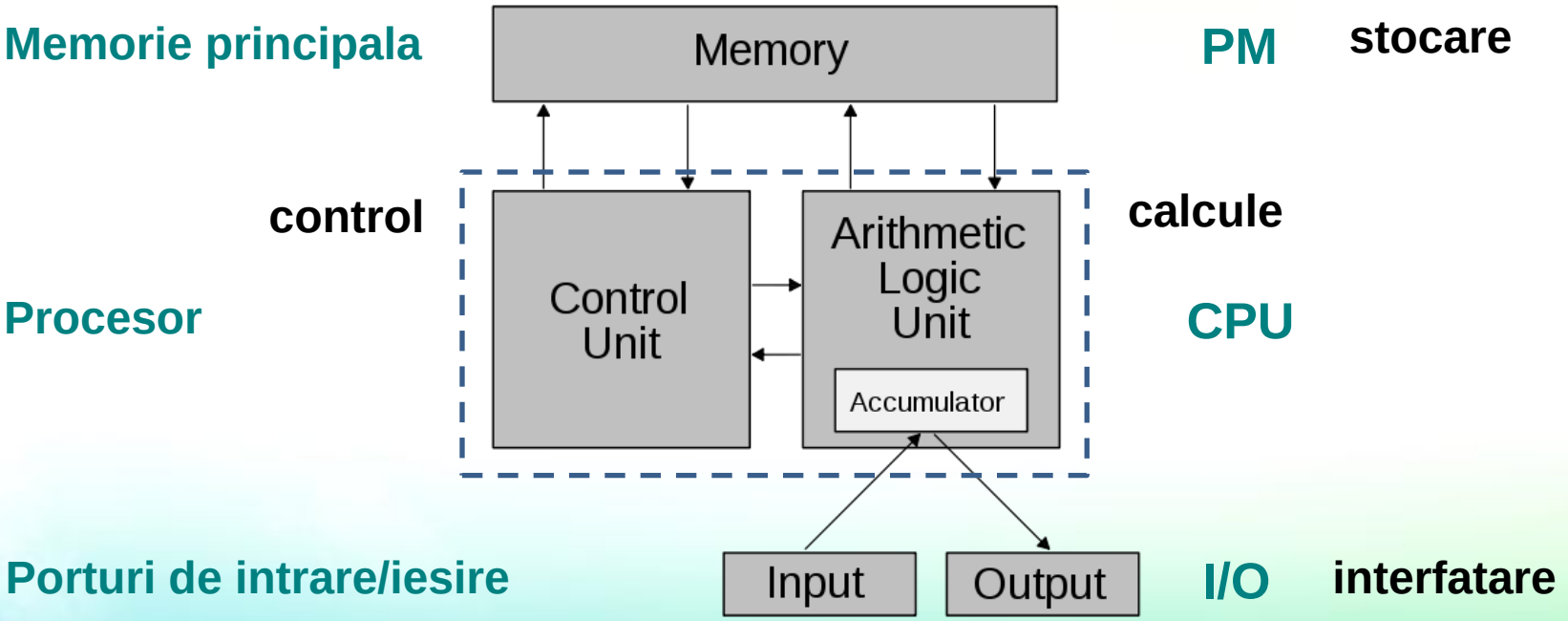
**Porturi de intrare/iesire** — **I/O** — **intrari/iesiri**



## Masina programabila si codul masina

**Arhitectura masinilor de calcul** cuprinde

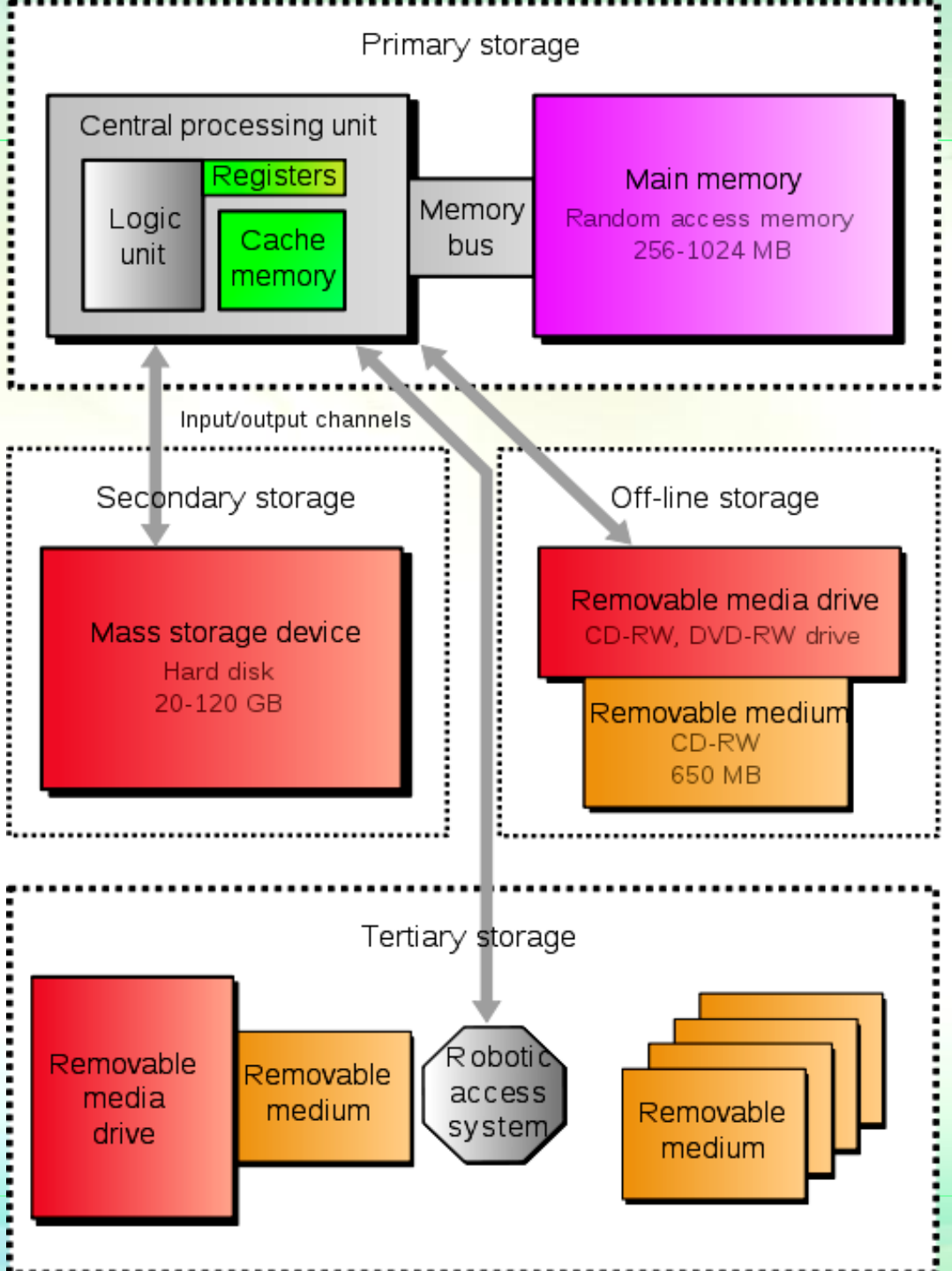
- **procesor** – **CPU** (*central processing unit*)
  - care interpreteaza/executa **instructiuni** in cod binar (**cod masina**)
- **memorie** principala – **PM** (*primary memory*) sau **MM** (*main memory*)
  - structura de stocare separata atat pentru **instructiuni** cat si pentru **date**



# Masina programabila si codul masina

**Niveluri de memorie**

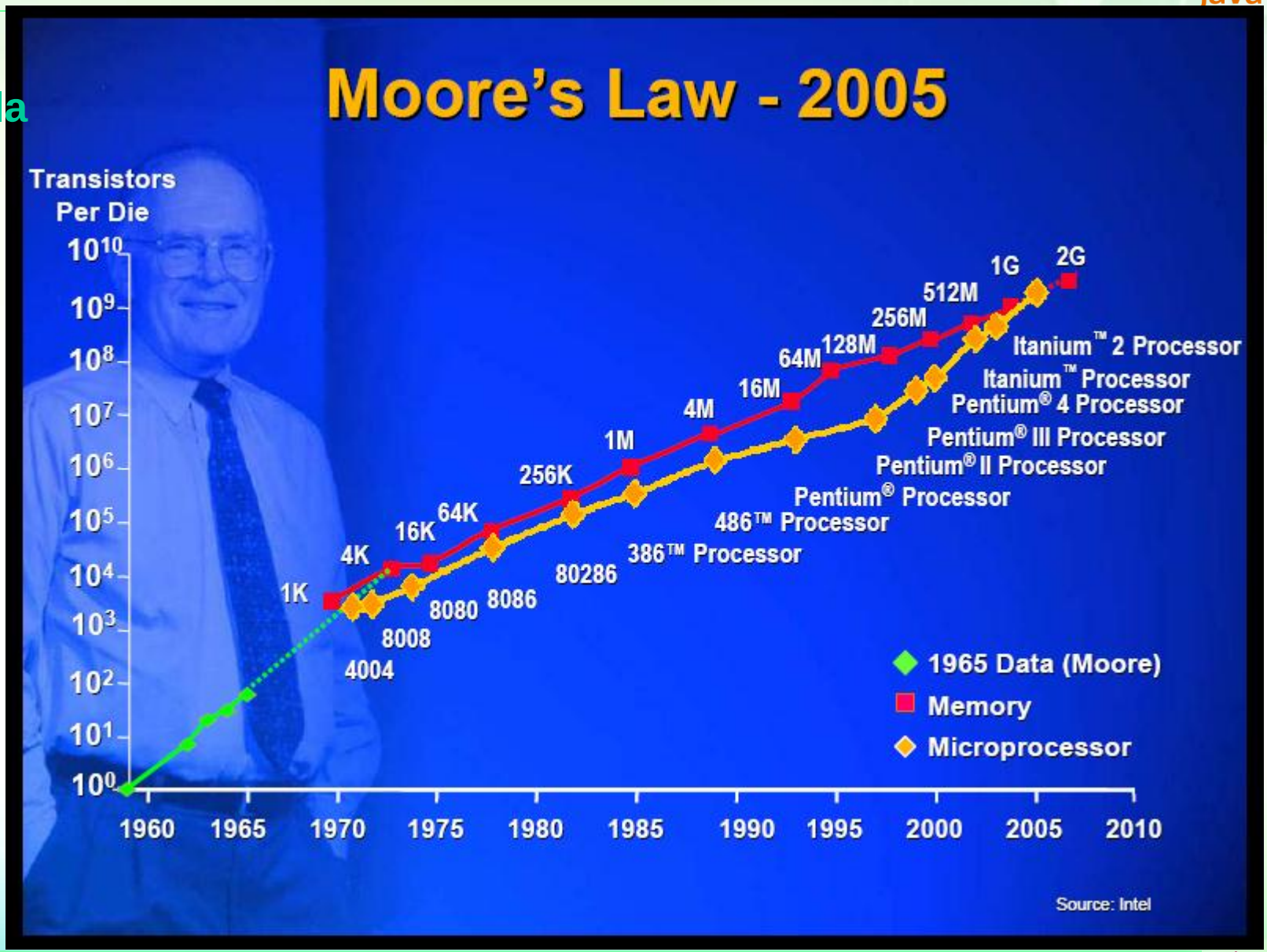
- **registre interne** ale **CPU**
- *cache* intern al CPU (mai nou)
- **memorie principala** (ROM + RAM)
- memorie secundara conectata prin **porturi I/O** (*harddisk*, etc.)



# 1.2. Evolutia catre abordarea OO

**Masina programabila si codul masina**

**Performante ale CPU si ale memoriei principale** (situatia in 2005, conform **legii lui Moore**)

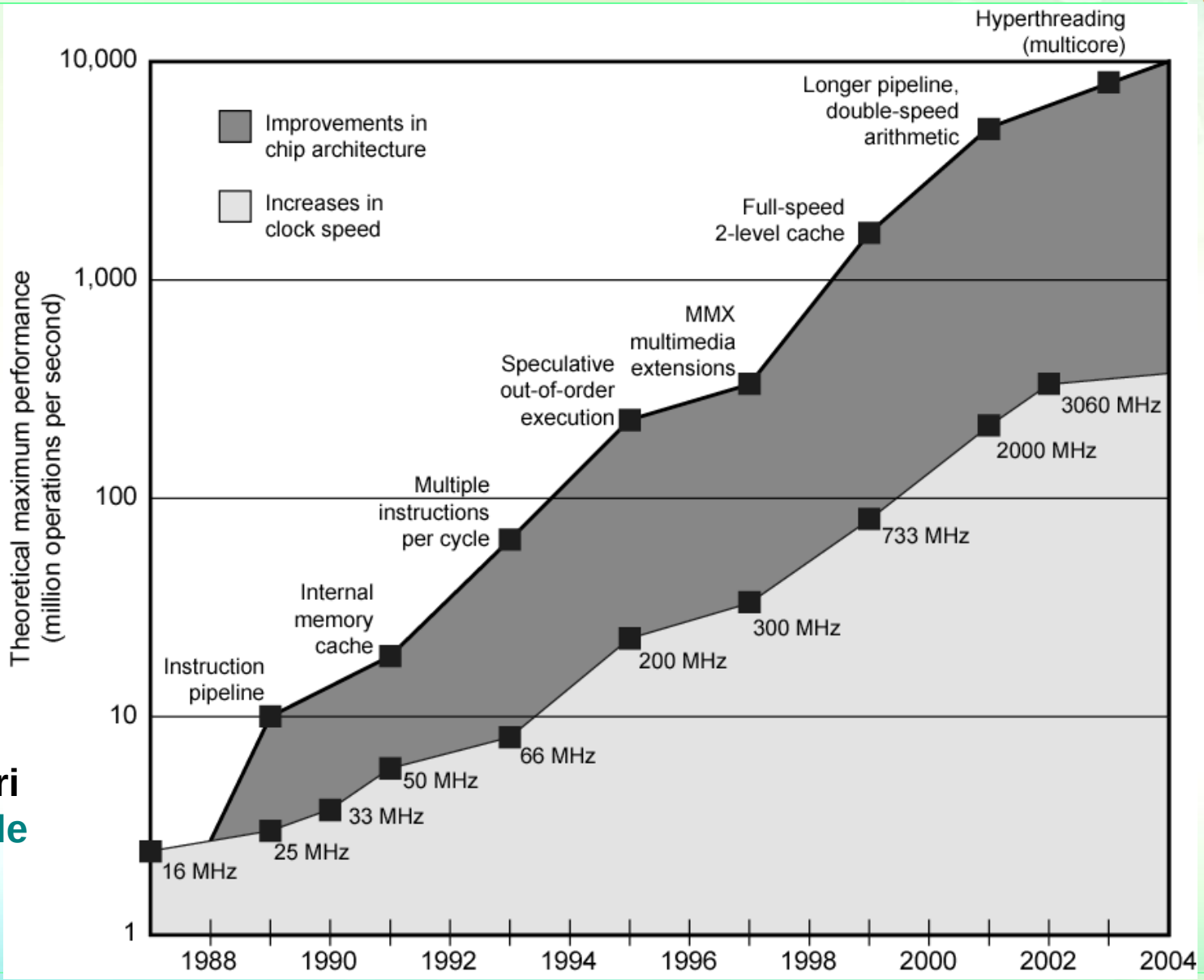


# 1.2. Evolutia catre abordarea OO

## Masina programabila si codul masina

**Sursele sporirii performantei CPU**

- **viteza ceasului** (ajunsa la **saturatie** din motive “termice”)
- **imbunatatiri arhitecturale**



## Masina programabila si codul masina

### Programele de calcul

- sunt **specificatii** ale **comportamentelor viitoare** ale unui **calculator**
  - descriu **raspunsurile** la **cereri/evenimente** venite **din exterior/de la utilizator**
  - sub forma de **instructiuni** intr-un "**limbaj**" pe care il poate intelege calculatorul
    - **cod masina**
      - direct **executabil** de catre **masina programabila** sau
    - **cod de octeti** interpretabil de catre un program **interpretor**
  - obtinute eventual prin **conversie**
    - din instructiuni scrise intr-un **limbaj** pe care il poate intelege mai bine **programatorul** (limbaj de programare)

## Masina programabila si codul masina

**Codul (limbajul) masina**

– este folosit **in masinile** de calcul **programabile** pentru a

– **specifica pasii** (instructiuni, actiuni) realizati de masina de calcul

– este initial **stocat** in **memorie** (PM)

– este apoi **incarcat** (*fetch*), **decodat** si **executat** in **unitatea de control** a procesorului (**CPU**), care este un fel de "***creier***" al intregii masini programabile

– **CPU** in acest caz este "***capul***" masinii programabile

– are **format numeric** digital (**binar**), in general **multiplu** de **8 biti** (**octeti** = *Bytes*) care face parte din **specificatiile tehnice ale procesorului**

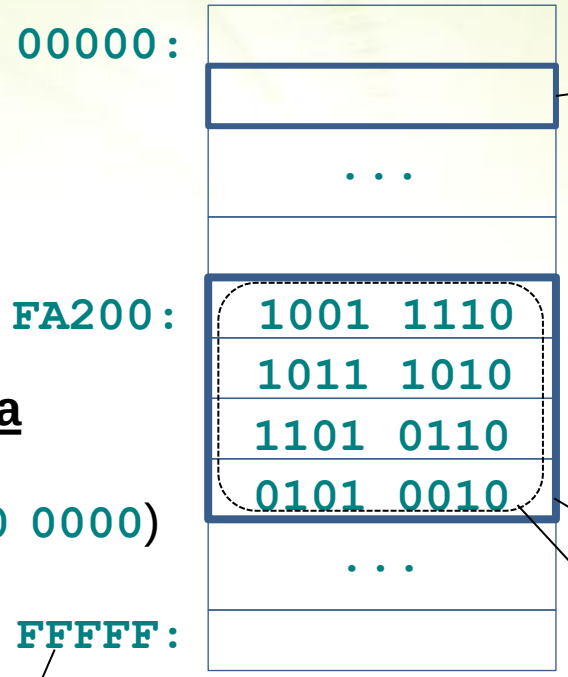
– **distinct** pentru fiecare **procesor** si

– **distinct** pentru fiecare **familie de procesoare**

## Masina programabila si codul masina

**Modul de reprezentare a informatiilor** digitale (binare) in **masinile programabile**

**Toate informatiile sunt reprezentate complet numeric!**




**8b** = **1B** (octet)

**Dimensiunea** spatiului alocat **(4B = 32b)**

**Adresa numerica** (echivalenta cu 1111 1010 0010 0000 0000)

5 cifre hexazecimal = 5 x 4 cifre binare = 20 biti

**Locatie** memorie (**spatiu** alocat)

**Valoare** stocata (**continut** locatie) – in acest caz 9E BA D6 52

## Masina programabila si codul masina

**Codul (limbajul) masina** este in general format din **actiuni** cum ar fi

- **copierea** din **memorie** (PM) in **registru** intern
  - = **citire** din PM
- **copierea** din **registru** intern in **memorie** (PM)
  - = **scriere** in PM
- **salturi conditionate la eticheta**
  - cod distinct pentru fiecare conditie
- **iteratii la eticheta**
  - salturi la eticheta cu incrementarea / decrementarea unui contor
- **operatii aritmetice** sau **logice**
  - realizate de unitatea aritmetica/logica (**ALU**)

  asupra valorilor din **registre** interne si/sau **memorie** (PM)

## Orientarea spre Obiecte (OO)

- **Evolutia catre OO**
  - Limbajele de **asamblare**
    - curs **CID** si **AMP** (sem II)

## Limbajele de asamblare

### Limbajul de asamblare

– este folosit de **programatorii** **masinilor** de calcul pentru a

– **specifica pasii** (instructiunile) realizati de masina de calcul

– are **format alfa-numeric**, **fiecare instructiune** in limbaj de **asamblare**

– fiind **echivalenta ”unu-la-unu”** cu o instructiune **cod masina** binar asociata unui procesor (CPU)

– ofera

– modalitati de **structurare slaba** a programelor, bazate pe **etichete**

– salturi (ne)conditionate catre **instructiuni etichetate**

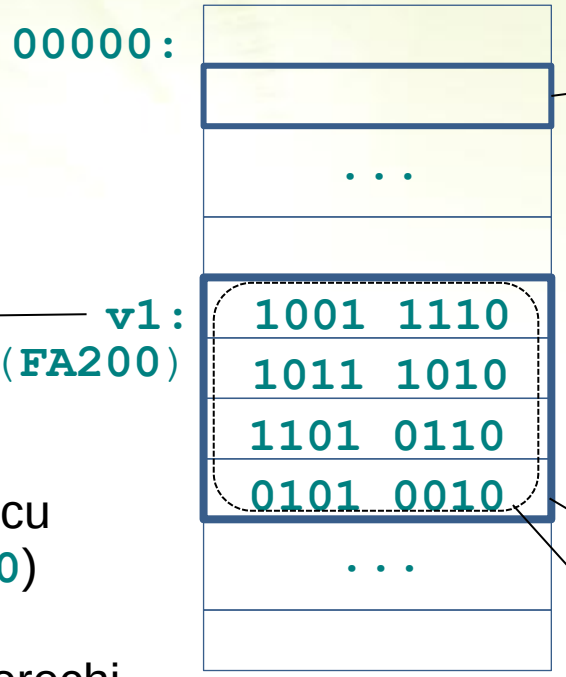
– bucle (salturi catre **instructiuni etichetate** cu actualizarea unui contor)

– operatii aritmetice/logice la nivel de **registre interne** / **locatii memorie**

## Limbajele de asamblare

**Modul de reprezentare a informatiilor** digitale (binare) la **nivel de asamblare**

**O parte dintre informatii** capata **nume / mnemonici** (abstracte pt. masina)




**8b** = **1B** (octet)

**Nume variabila**

**Adresa numerica** corespunzatoare (echivalenta cu `1111 1010 0010 0000 0000`)

Accesibila, de exemplu, prin perechi de **registre din procesor (I8086)**:

| | | | |
|---|---|---|---|
| Reg. **DS** | `1111 1010` 0000 0000 | 0000 + | `FA000+` |
| Reg. **SI** | 0000 `0010 0000 0000` | | `0200` |

**Dimensiunea** spatiului alocat **(4B = 32b)**

**Variabila** (**locatie** memorie)

**Valoare** stocata (**continut** variabila)
`9E BA D6 52`
(in zecimal: `2663044690`)

## Limbajele de asamblare

**Limbajul de asamblare** este in general format din **actiuni** cum ar fi

– **copierea** din **memorie** (PM) in **registru** intern = **citire** din PM

```
instr1:   MOV  AX, pi          ; copierea variabilei pi in registrul AX
```

– **copierea** din **registru** intern in **memorie** (PM) = **scriere** in PM

```
          MOV  suma, AX        ; copierea registrului AX in variabila suma
```

– **operatii** realizate de unitatea aritmetica/logica (**ALU**) asupra valorilor din registre interne si/sau memorie (PM)

```
          ADD  suma, CX        ; adunarea la variabila suma a registrului AX
```

– **salturi conditionate** (cod distinct pentru fiecare conditie) **la eticheta**

```
          JNC  instr1          ; salt conditionat de CF==0 la eticheta instr1
```

– **iteratii la eticheta** (salturi la eticheta cu decrementarea unui contor)

```
          LOOP  instr1         ; salt cu decrementare contor CX la instr1
```

## Limbajele de asamblare

**Programul** in **limbaj de asamblare**

– este mai intai **editat**

– in **fisiere sursa** (ex. `equation.asm`)

– este apoi **convertit** (**asamblat**) in **cod masina** al **CPU** pe care va fi executat

– si eventual “legat” cu alte coduri masina (biblioteci, alte surse, etc.)

– rezultand in final **fisiere executabile** (de ex. `equation.exe`)

– si este in final **executat** **de catre CPU** prin intermediul ajutorul sistemului de executie/operare

– devenind **proces in executie**

## Orientarea spre Obiecte (OO)

- **Evolutia catre OO**
  - Limbajele de **nivel inalt** (pre-OO)
    - curs **PC** si **SDA** (anul 1)

## Limbajele de nivel inalt (pre-OO)

**Limbajul de nivel inalt**

– este folosit de **programatorii** **masinilor** de calcul pentru a

– **specifica pasii** (instructiunile) realizati de masina de calcul

– are **format alfa-numeric**, iar **fiecare instructiune la nivel inalt**

– este in general **echivalenta cu o secventa** de instructiuni in limbaj de **asamblare** (sau de **cod masina** binar) al unui procesor (CPU)

– ofera o **abstractizare** **puternica** a **detaliilor** **masinii programabile**

– spre deosebire de **limbajele de asamblare** (de **nivel redus**)

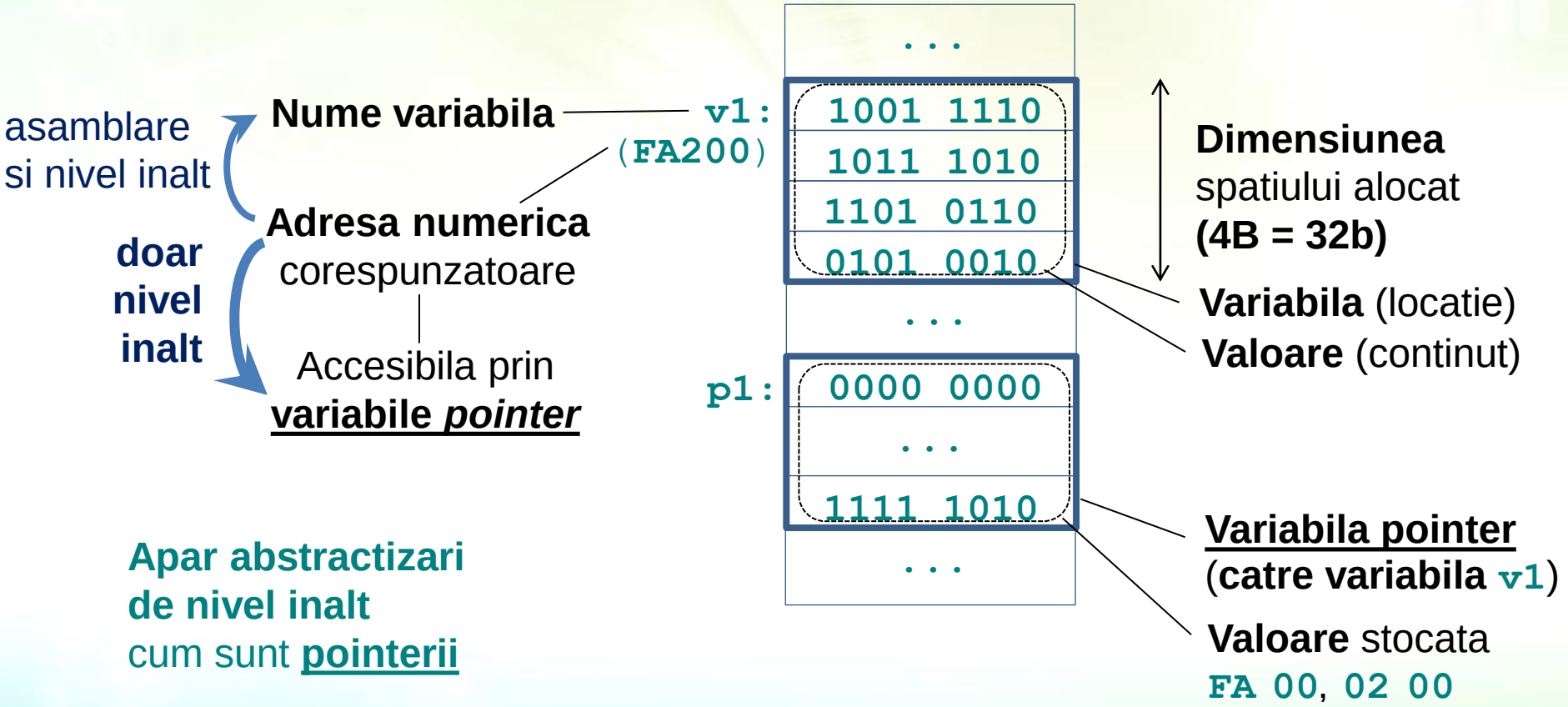
– care folosesc **echivalente directe** ale **codurilor masina**

– ceea ce le face mai **usor de folosit**

– si mai **portabile** pe **diverse masini programabile**

## Limbajele de nivel inalt (pre-OO)

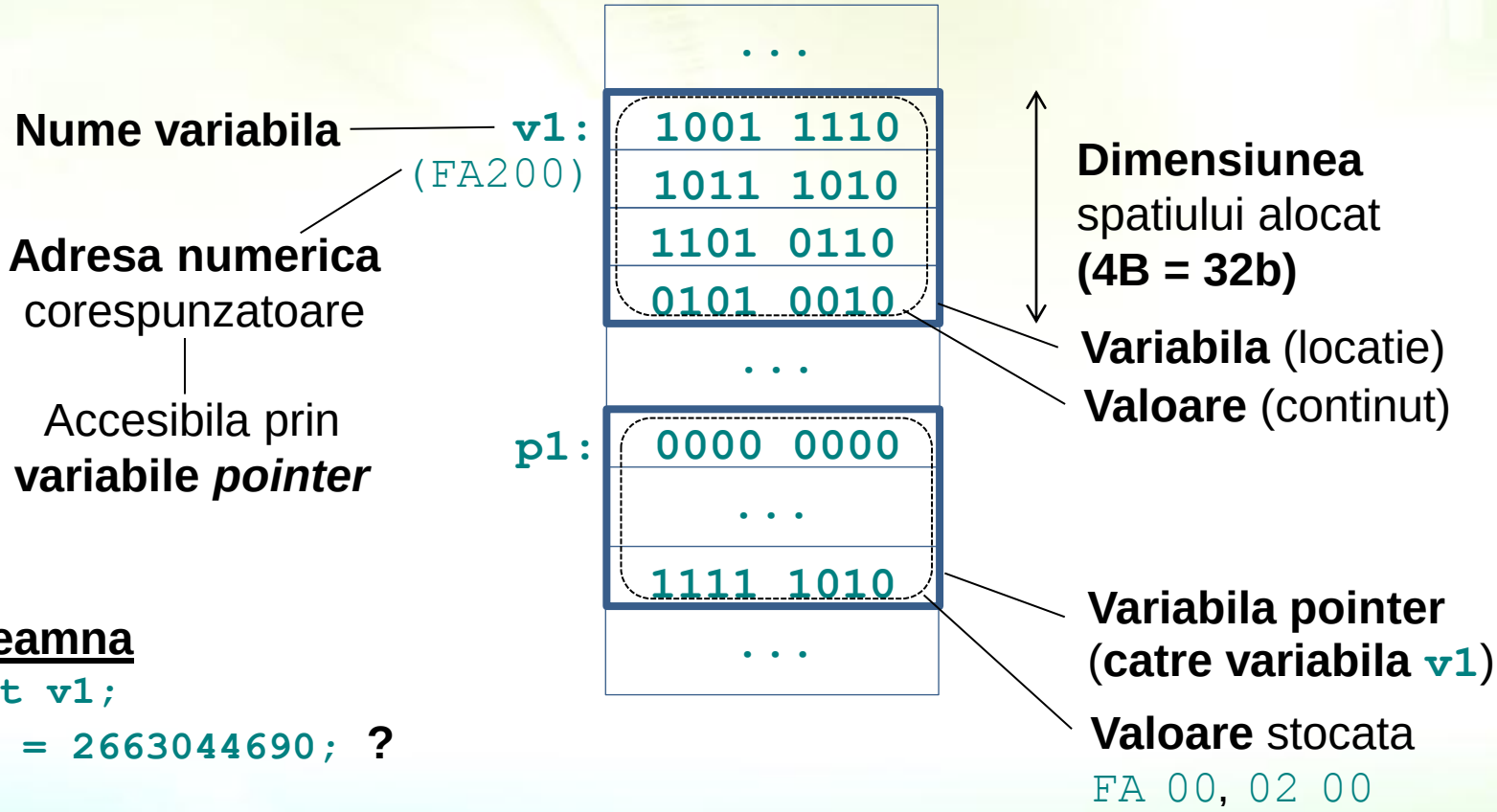
**Modul de <u>reprezentare</u> a informatiilor** digitale (binare) la **nivel inalt**




**Apar abstractizari de nivel inalt**
cum sunt **<u>pointerii</u>**

## Limbajele de nivel inalt (pre-OO)

**Modul de reprezentare a informatiilor** digitale (binare) la **nivel inalt**




**Ce inseamna**

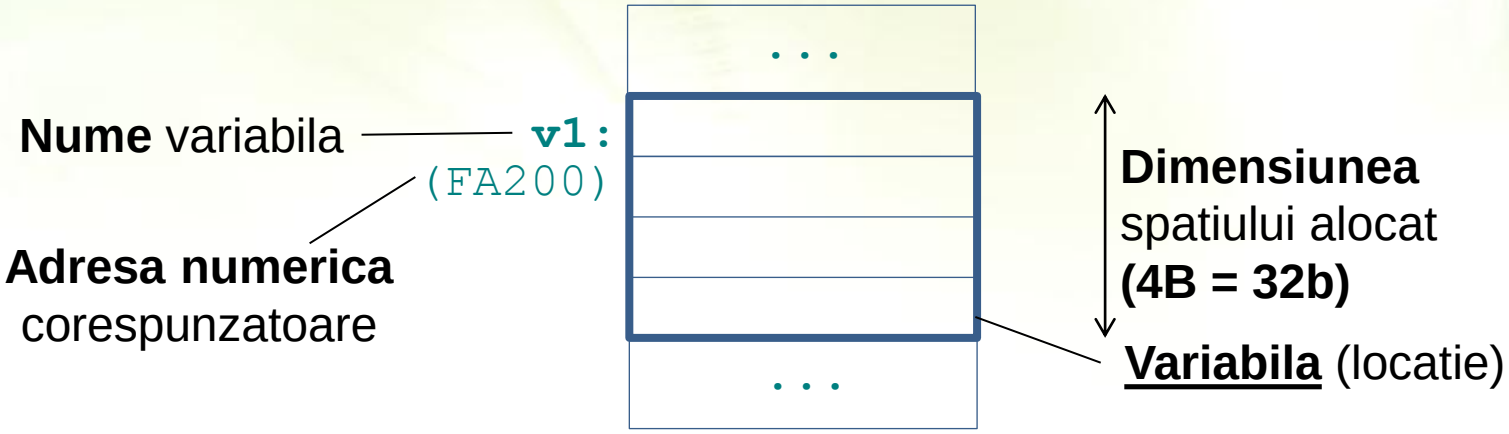
```
int v1;
v1 = 2663044690;
```
**?**

**Dar**

```
int *p1;
p1 = &v1;
```
**?**

## Limbajele de nivel inalt (pre-OO)

**Modul de reprezentare a informatiilor** digitale (binare) la **nivel inalt**



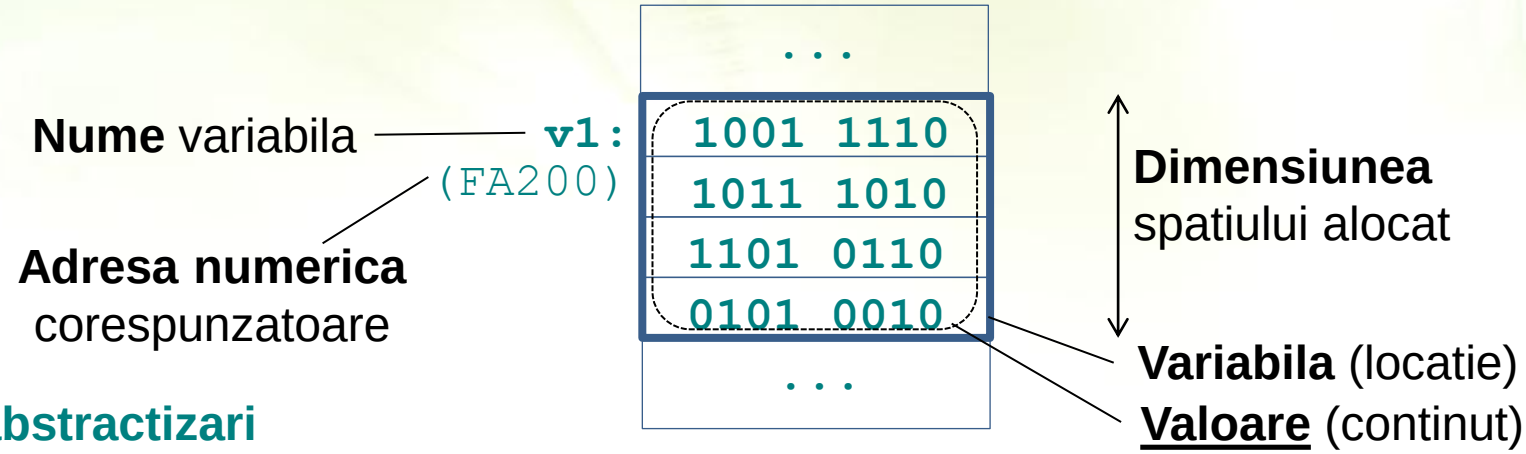

```
int v1; // declara variabila v1 de tip "int"
```

**Tipul de date** este

- **descriere** a unei **multimi de entitati de date asemanatoare** numite **variabile**
  - care **specifica structura** variabilelor (**dimensiunea** spatiului alocat)
  - **domeniul de definitie** al **valorilor** lor (**gama** valorilor, **formatul** lor literal)
  - **conversiile** catre alte tipuri, **operatorii** asociati / permisi

## Limbajele de nivel inalt (pre-OO)

**Modul de reprezentare a informatiilor** digitale (binare) la **nivel inalt**

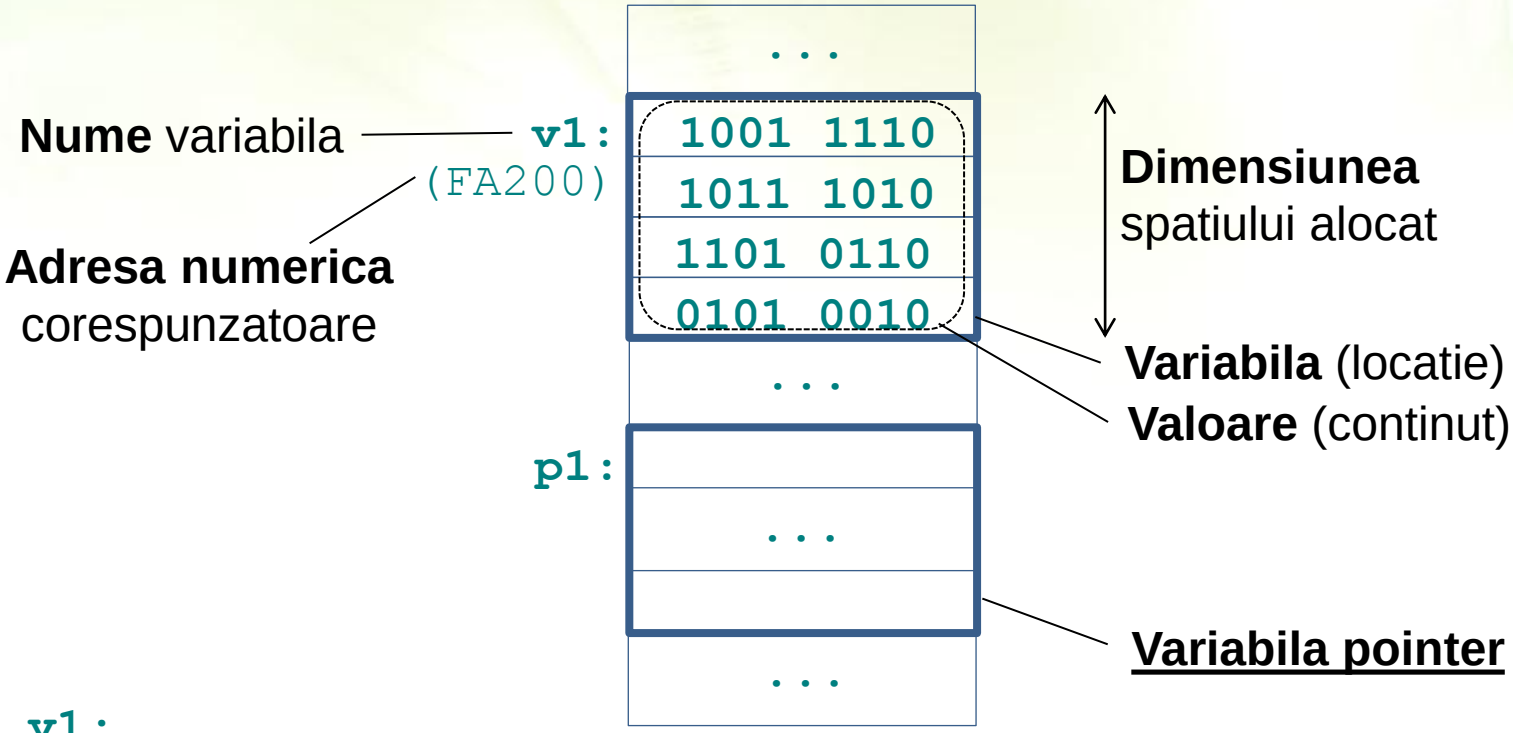



**Apar abstractizari**
**de nivel inalt** cum sunt
**tipurile de date**

```
int v1;            // declara variabila v1 de tip "int"
v1 = 2663044690; // atribuie o valoare variabilei v1
```

## Limbajele de nivel inalt (pre-OO)

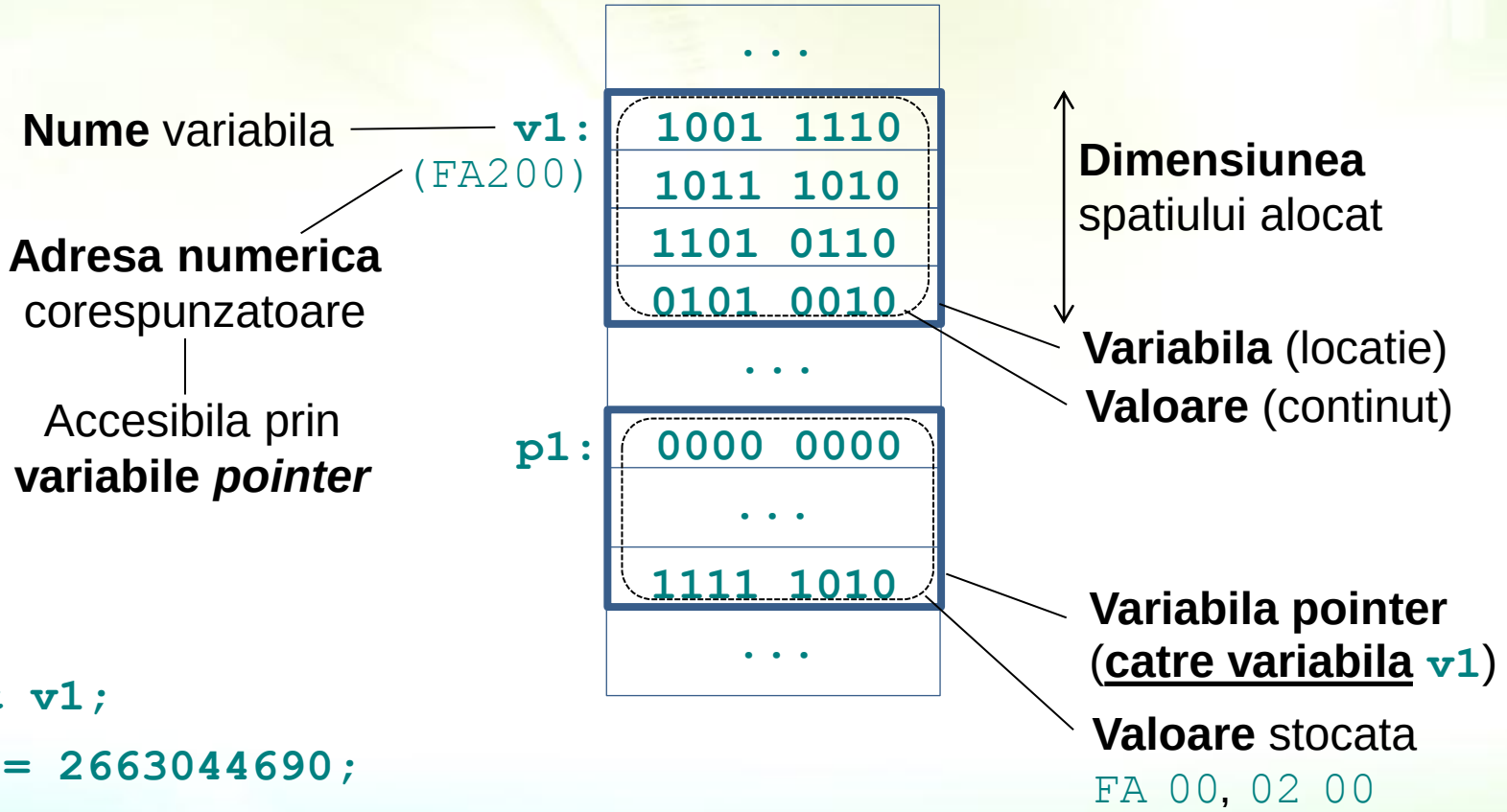
**Modul de reprezentare a informatiilor** digitale (binare) la **nivel inalt**




```
int v1;
v1 = 2663044690;
int *p1;  // declara p1 de tip "pointer catre"
          // variabile de tip "int"
```

## Limbajele de nivel inalt (pre-OO)

**Modul de reprezentare a informatiilor** digitale (binare) la **nivel inalt**




```
int v1;
v1 = 2663044690;
int *p1;  // "pointer catre" variabile tip "int"
p1 = &v1; // atribuie variabilei p1 valoarea adresei v1
```

## Limbajele de nivel inalt (pre-OO)

**Modul de <u>prelucrare</u> a informatiilor** digitale (binare) la **nivel inalt**

**Exemplu de <u>specificatie</u> pseudocod pentru "*calculul modulului sumei a doua valori*"**

***calculul sumei a doua valori***

**daca [*suma este negativa*]**

***negarea sumei***

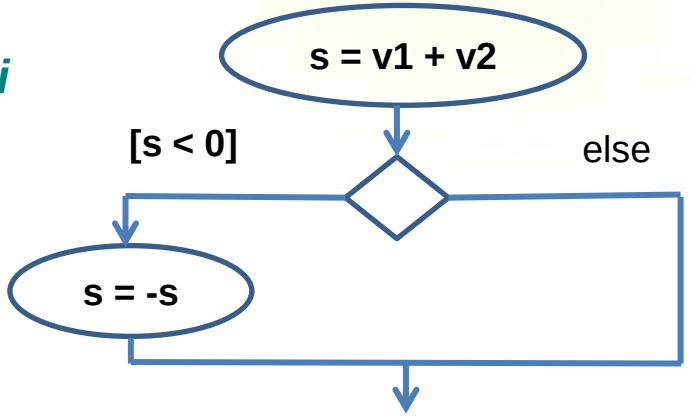


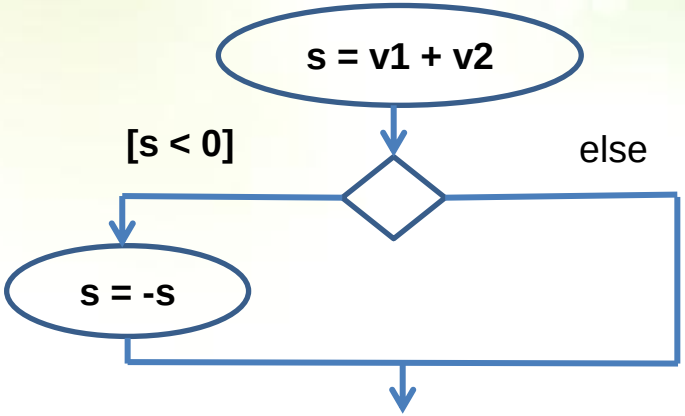

**Diagrama UML de activitati** (diagrama de **flux al programului**) echivalenta

<u>Cum ar putea arata o portiune de **program** scris in **limbajul de nivel inalt** C care sa realizeze **sarcina specificata?**</u>

## Limbajele de nivel inalt (pre-OO)

**Posibila portiune de program C** pentru **"*calculul modulului sumei a doua valori*"**

```
s = v1 + v2;
if (s<0)
    s = -s;
```




Exista / cunoasteti si **forme mai simple**?

Ce sunt **s**, **v1** si **v2**?

**Ce lipseste din program** in ceea ce le priveste?

Cum ar putea arata un **program C complet**?

## Limbajele de nivel inalt (pre-OO)

**Posibil program C** complet pentru **"*calculul modulului sumei a doua valori*"**

```
#include <stdio.h>
int v1 = 3;  // variabile globale
int v2 = 5;
int main(void) { // functie principala, punct intrare progam
    int s;       // variabila locala
    s = v1 + v2;
    if (s<0)
        s = -s;
    printf("modulul sumei %d cu %d este %d\n", v1, v2, s);
    return 0;
}
```

Cum ar putea arata o portiune de **program** scris in **limbajul de asamblare** care sa realizeze **sarcina specificata?**

## Limbajele de nivel inalt (pre-OO)

**Posibila portiune de program** scris in **limbajul de asamblare al procesorului Intel8086** pentru **"*calculul modulului sumei a doua valori*"**

```
            MOV AX, v1      ; copiere din memorie in registru
            ADD AX, v2      ; adunare la valoarea din registru
            JGE eticheta1   ; salt conditionat de [AX >= 0]
            NEG AX          ; negare aritmetica (schimbare semn)
eticheta1:  MOV s, AX       ; copiere din registru in memorie
```

Cat de usor sunt de **inteles intuitiv mnemonicele** instructiunilor?

Ce stiti despre **salturile la etichete** (instructiunile de tip "**go to**")?

Cum este **echivalata** instructiunea **if**?

Cum ar putea arata un program care calculeaza **doar suma**?

## Limbajele de nivel inalt (pre-OO)

**Programul** in **limbaj de nivel inalt**

– este mai intai **editat**

– in **fisiere sursa** (ex. `equation.cpp`)

– este apoi **convertit** (**compilat**) in **cod masina** al **CPU** pe care va fi executat

– si eventual “legat” cu alte coduri masina (biblioteci, alte surse, etc.)

– rezultand in final **fisiere executabile** (de ex. `equation.exe`)

– si este in final **executat** **de catre CPU** prin intermediul ajutorul sistemului de executie/operare

– devenind **proces in executie**

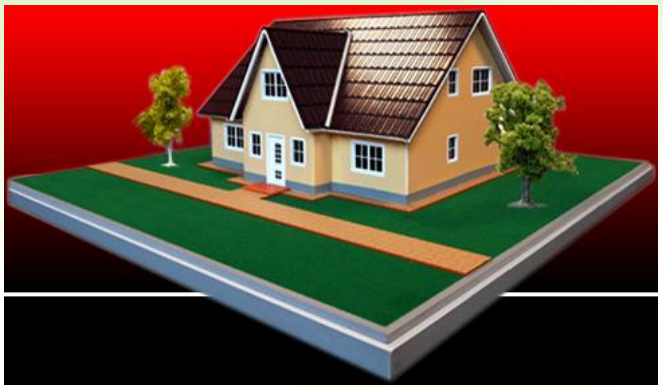
## Orientarea spre Obiecte (OO)

- **Evolutia catre OO**
  - Modelare si **abstractizare (I)**

## Modelare si abstractizare

**Model** (definitii)

♦ **machetă** = obiect cu **dimensiuni reduse** care **reprezintă** un **obiect real**



♦ **tipar, sablon, tip** = obiect determinat **după care se reproduc** obiecte **similare**

♦ **mostră, exemplu** = obiect întrunind **însușirile tipice** ale unei **categorii**, destinat pentru a fi **reprodus**

## Modelare si abstractizare

**Model** (sensul informatic)

- **reprezentare simplificată** a unui proces sau a unui sistem complex, care
  - ofera o **analogie** cu procesul/sistemul complex
  - cuprinde **elementele esentiale** ale procesului/sistemului complex
- **usureaza** astfel **accesul** la **esenta** (pt. analiza, exploatare, etc.)
  - printr-o forma de **acces indirect** la procesul/sistemul complex

## Modelare si abstractizare

**Modelul**

Un model expandeaza o portiune


in detrimentul celorlalte

## Modelare si abstractizare

**Modelare**

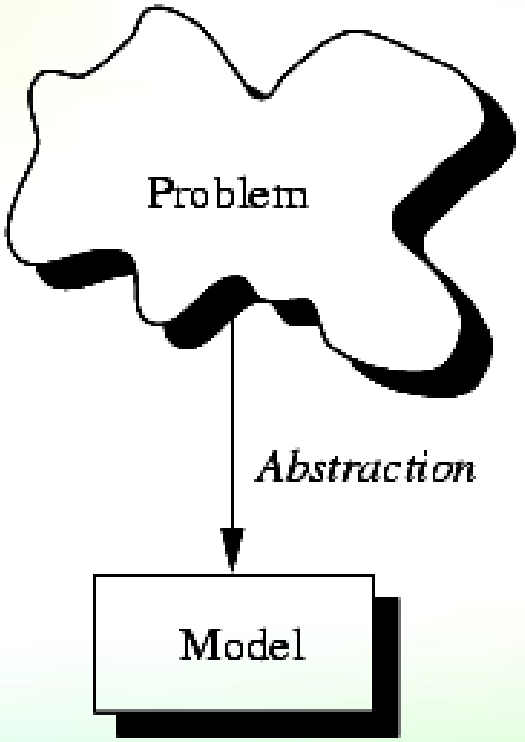
– construire de modele

– **metodă** care consta în reproducerea / **reprezentarea simplificată** a unui proces sau sistem complex

– sub forma unui **proces/sistem** **similar** sau **analog**

– care ofera **accesul** la **esenta** procesului/sistemului complex

– ceea ce insemna **accesul indirect** la procesul/sistemul complex




De ce **constructia unui model** este **ilustrata** ca “**abstractizare**”?

## Modelare si abstractizare

**Abstractizare** (abstractie)

– **operație a gândirii** prin care

– **se desprind** și **se reține** **unele elemente** (caracteristici și relații) considerate **esențiale** (fundamentale, generale)

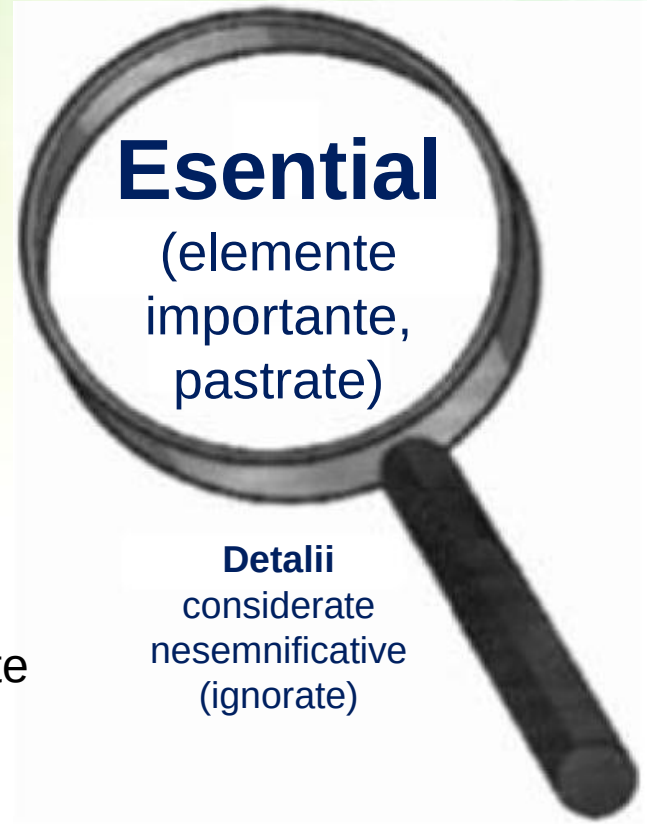
– ale unei **entitati analizate**

– sau **comune unei multimi de entitati**

– si se **ignora** (vremelnic) elementele considerate **neesențiale**

– **elementele** considerate **esentiale**

– **diferentiaza** entitatea sau multimea analizata **de alte** entitati sau multimi de entitati



## Modelare si abstractizare

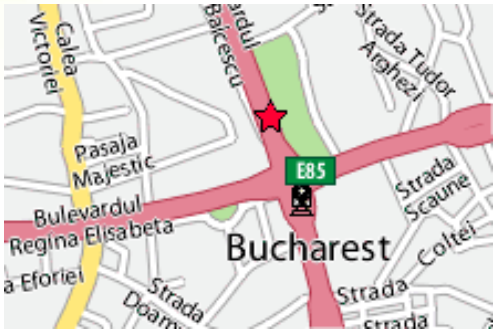
**Exemplu de abstractizare** (*teritoriul si harta*)

| *Entitate modelata* | *Model abstract* |
|---|---|
| **Teritoriul** | O **harta detaliata** a teritoriului |
|  |  |
| **Portiune din realitate,** imposibil de atins in totalitate, de cuprins | **Simplificare, reducere a realitatii** (mai usor de gestionat si modificat) |

**CONCRET**

## Modelare si abstractizare

**Exemplu de abstractizare** (*teritoriul si harta*)

| *Entitate modelata* | *Model abstract* | *Model si mai abstract* |
|---|---|---|
| **Teritoriul** | O **harta detaliata** a teritoriului | O **harta** a teritoriului **mai putin detaliata** |
|  |  |  |
| **Portiune din realitate,** imposibil de atins in totalitate, de cuprins | **Simplificare, reducere a realitatii** (mai usor de gestionat si modificat) | **Simplificare, reducere suplimentara** |

**CONCRET (complex)** ⟵ ***Ierarhie de abstractizari*** ⟶ **ABSTRACT (simplificat)**

## Modelare si abstractizare

**Abstractizare** (abstractie)

– **importanta** in **rezolvarea problemelor** (*problem solving*) deoarece

– le permite **celor ce rezolva** problemele sa se **concentreze** pe detaliile **esentiale**

– in timp ce sunt **ignorate celelalte**, avand ca efecte

– **simplificarea problemei** si

– **concentrarea atentiei** pe **aspectele** problemei care sunt implicate in **solutia** sa

## Modelare si abstractizare

**Abstractizarea**

- este o **forma de <u>management</u>**
  - al **<u>complexitatii</u>**
    - deoarece are ca scop **concentrarea pe esential**
    - si ca efect implicit **simplificarea**
  - al **<u>schimbarii</u>**
    - deoarece poate fi folosita pentru
      - **identificarea** si **transformarea in element al modelului** a **ceea ce este stabil in timp / permanent**
      - si pentru **separarea** a ceea ce este **stabil in timp / permanent** de ceea ce este **variabil in timp / temporar**
    - ceea ce poate avea ca efect pozitiv **inlesnirea schimbarii**

## Modelare si abstractizare

**Abstractizarea**

– forma de **modelare** aplicabila fie unei **entitati** fie unei **multimi** de entitati

– inseamna **crearea** unui **model simplificat**

– format din **detalii** considerate **esentiale** ale **entitatii/multimii modelate**

– **neglijand celelalte detalii** ale entitatii/multimii modelate

– ***"concentrarea pe esential"*** face **abstractizarea** un **concept relativ**

– deoarece **depinde de factorii care pot dicta** ceea **ce este esential**

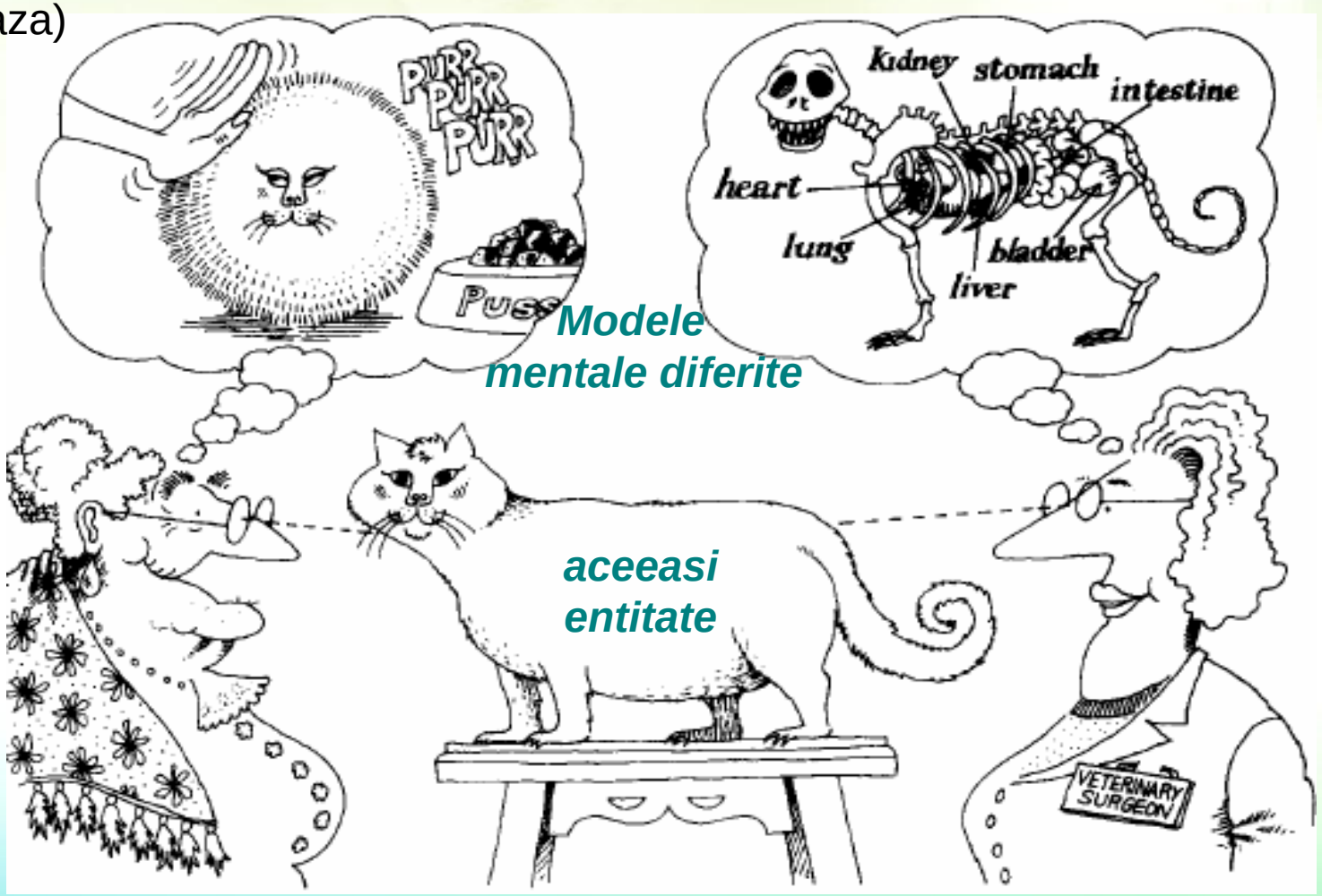
– **contextul** abstractizarii si

– **interesul celui care abstractizeaza**!

– "***neglijarea unor detalii***" inseamna o forma de **aproximare a realitatii**

## Modelare si abstractizare

**Abstractizarea e relativa** – depinde de **cine decide ce e esential** (persoana care abstractizeaza)




**Diferite perspective (puncte de vedere)**

## Modelare si abstractizare

**Modelare vs Abstractizare**

- **modelarea** insista pe
  - usurarea **accesului indirect** la procesul/sistemul tinta
    - prin crearea unui model **analog**, **similar**, simplificat si esential
- **abstractizarea** insista pe
  - **simplificare** prin **identificarea** si **pastrarea** elementelor considerate esential
  - si **neglijarea** detaliilor **considerate neesentiale**
    - conducand la crearea unui model **simplificat** si **esential**

**Altfel spus**

- **modelarea** insista pe **analogie** si **similitudine**
- **abstractizarea** insista pe **simplificare** si **esential**

## Modelare si abstractizare

**Exemplu de abstractizare** (*masina de calcul si limbajele*)

***Entitate abstractizata***

**Masina de calcul**
condusa de
**coduri masina**
**+**
**Limbajul masina**

(**limbajul masinii** de calcul)

⇨ **Cum este abstractizata masina de calcul?**

**CONCRET**

## Modelare si abstractizare

**Exemplu de abstractizare** (*masina de calcul si limbajele*)

| ***Entitate abstractizata*** | ***Model abstract*** | ***Model si mai abstract*** | ***Modelul cel mai abstract*** |
|---|---|---|---|
| **Masina de calcul** condusa de **coduri masina** + **Limbajul masina** | **Masina de calcul** programata la **nivel de asamblare** + **Limbajul de asamblare** | **Masina de calcul** programata la **nivel inalt** + **Un limbaj de nivel inalt** | Proiectarea **programarii masinii de calcul** + **Pseudocod**, etc. |
| (**limbajul masinii** de calcul) | (**echivalent uman** al limbajului masinii) | (**forma concentrata** a limbajului masinii, **apropiata de cea umana**) | (**forma intuitiva din punct de vedere uman**) |

**CONCRET** ⟵ ***Ierarhie de abstractizari*** ⟶ **ABSTRACT**

## Modelare si abstractizare

**Exemplu de abstractizare** (*masina de calcul si limbajele*)

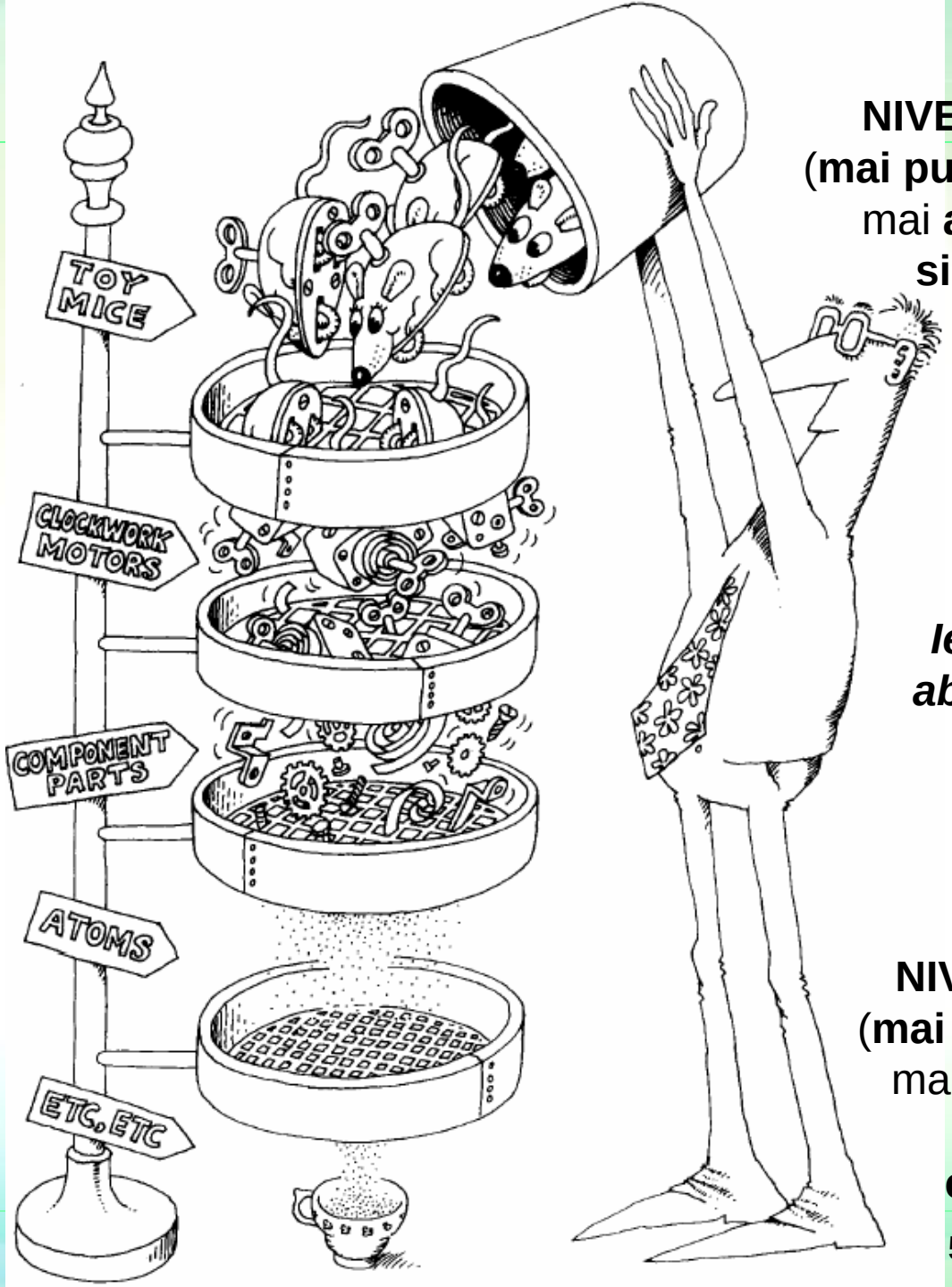
| ***Entitate abstractizata*** | ***Model abstract*** | ***Model si mai abstract*** | ***Modelul cel mai abstract*** |
|---|---|---|---|
| **Masina de calcul** condusa de **coduri masina** + **Limbajul masina** | **Masina de calcul** programata la **nivel de asamblare** + **Limbajul de asamblare** | **Masina de calcul** programata la **nivel inalt** + **Un limbaj de nivel inalt** | Proiectarea **programarii masinii de calcul** + **Pseudocod**, etc. |
| **Ex. de concepte:**<br>– coduri instructiuni<br>– registre<br>– locatii memorie<br>– operanzi<br>– adrese | **Ex. de concepte:**<br>– mnemonici instructiuni<br>– registre<br>– variabile<br>– salturi<br>– operanzi<br>– call procedura | **Ex. de concepte:**<br>– decizii<br>– variabile<br>– structuri de date<br>– expresii<br>– apeluri functii<br>– blocuri de cod<br>– pointeri | **Ex. de concepte:**<br>– decizii<br>– date<br>– informatii<br>– actiuni<br>– operatii<br>– repetitii |

## Modelare si abstractizare

**Abstractizarile formeaza ierarhii**

**Ierarhiile** de abstractizari

– **grupeaza abstractizarile** in **niveluri de abstractizare** (**complexitate**)




**NIVEL INALT** (**mai putine detalii**, mai **abstract**, **simplu**)

***Ierarhie de abstractizari***

**NIVEL REDUS** (**mai multe detalii**, mai apropiat de **concret**, **complex**)

## Modelare si abstractizare

**Exemplu de abstractizare** (*entitatile lumii reale, programele si limbajele*)

***Entitate abstractizata***

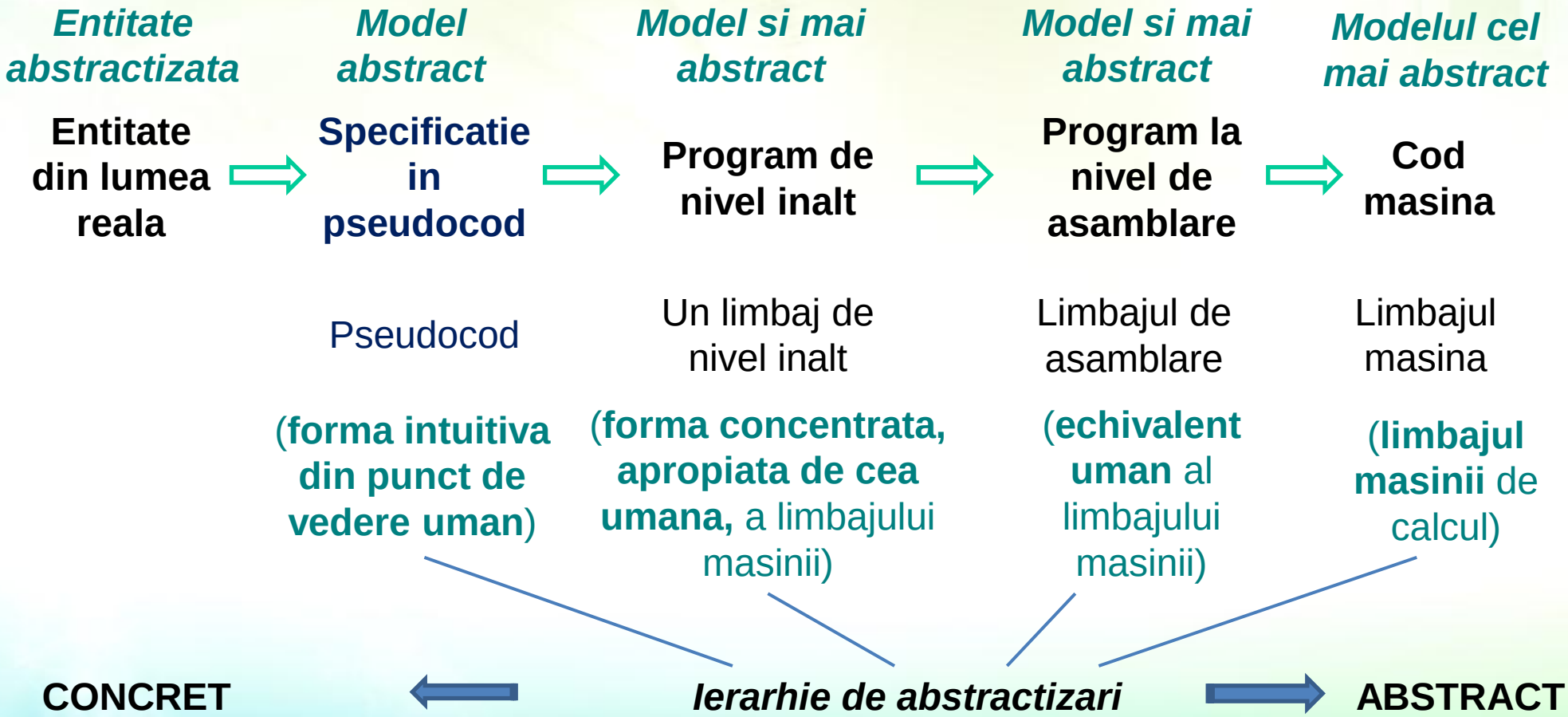
**Entitate din lumea reala** ⇨

**Cum poate fi abstractizata informatic?**

**CONCRET**

## Modelare si abstractizare

**Exemplu de abstractizare** (*entitatile lumii reale, programele si limbajele*)



## Modelare si abstractizare

**Exemplu de abstractizare** (*entitatile lumii reale, programele si limbajele*)

| *Entitate abstractizata* | *Model abstract* | *Model si mai abstract* | *Model si mai abstract* | *Modelul cel mai abstract* |
|---|---|---|---|---|
| **Entitate din lumea reala** | **Specificatie in pseudocod** | **Program de nivel inalt** | **Program la nivel de asamblare** | **Cod masina** |
| | Pseudocod | Un limbaj de nivel inalt | Limbajul de asamblare | Limbajul masina |

***"calculul modulului sumei a doua valori"***

***calculul sumei a 2 valori***

**daca [*suma e negativa*]**

***negarea sumei***

```
s = v1 + v2;
if (s<0)
    s = -s;
```

```
MOV AX, v1
ADD AX, v2
JGE et1
NEG AX
et1: MOV s, AX
```

```
01000101
11000010
01000101
01110010
01010001
01010101
01111010
10010111
```

## Modelare si abstractizare

**Recapitulare a exemplelor de abstractizare care reflecta relativitatea conceptului**

| Limbaj de programare | Format | Controlul executiei realizat prin | Folosit in | Cat de abstract este | |
|---|---|---|---|---|---|
| | | | | pentru masina | pentru om |
| cod masina | numeric binar | **salturi** conditionate, iteratii catre etichete | masina de calcul | deloc | **extrem de mult** |
| asamblare | alfa-numeric | **salturi** conditionate, iteratii simplecatre etichete | programarea la nivel asamblare | destul de mult | foarte mult |
| procedural (nivel inalt) | alfa-numeric | **decizii** (simple+multiple), iteratii complexe (mai multe tipuri) | programarea la nivel inalt | mult | mult |
| pseudocod | textual, informal | **decizii** (simple+multiple), iteratii | proiectarea programelor | **foarte mult** | **destul de putin** |