# Arhitectura Sistemelor de Calcul

Lect. Dr. Șotropa Diana
diana.sotropa@ubbcluj.ro



Facultatea de Matematică și Informatică
Universitatea Babeș-Bolyai


# CURS 1 – INTRODUCERE

LECT. Dr. Șotropa Diana-Florina

# CE ESTE ARHITECTURA CALCULATOARELOR?

- Sistem de calcul = dispozitiv care lucrează automat, sub controlul unui program memorat, prelucrând date în vederea producerii unor rezultate ca efect al procesării

- Arhitectura unui Sistem de calcul poate fi analizată la nivel structural (care sunt componentele fizice și căile de comunicare între acestea) sau la nivel logic (funcțiile fiecărei componente în cadrul structurii)

# CE ESTE ARHITECTURA CALCULATOARELOR?

- **Arhitectura calculatoarelor** studiază modul în care sunt proiectate și organizate componentele hardware ale unui calculator și cum interacționează acestea cu software-ul.

- Proiectarea logică a unui sistem de calcul: cum sunt structurate CPU-ul, memoria, magistralele și dispozitivele de intrare/ieșire.
- Interfața hardware–software: cum scrie un programator cod care ajunge să fie executat de un procesor.
- Modelarea internă a procesorului: registre, unități funcționale, circuite de control, moduri de adresare etc.
- Legătura cu performanța aplicațiilor: ce se întâmplă când rulăm un program, cum influențează arhitectura viteza de execuție.

# Structura CURS

1. Reprezentarea datelor: Tipuri de date elementare, reprezentari binare si ordini de plasare, organizarea si memorarea datelor

2. Codificarea caracterelor, codificarea numerelor întregi, convenție cu semn si fara semn, bitul de semn, cod complementar, operatii aritmetice, conceptul de depasire, conversia la o locație de alte dimensiuni

3. Performantele unui SC, dimensiunea unui microprocesor, arhitectura microprocesorului 80x86 – structura, registrii, calculul de adresa, moduri de adresare, adrese FAR si NEAR

4. Unitatea executiva (EU) a microprocesorului 80x86: rolul si functiile registrilor si al flagurilor. Clasificare (Registrii si Flaguri) si studii de caz.

5. Unitatea BIU a microprocesorului 80x86: registrii de adresa, registrii de segment, reprezentarea instrucțiunilor. Formula despecificare offset pe 32 biți si formula de specificare offset pe 16 biți. Aritmetica de pointeri

6. Elementele limbajului de asamblare: formatul unei linii sursa, expresii, tipuri de accesare a operanzilor, operatori aritmetici, operatori pe biti, operatori de tip si tipuri de date asociate operanzilor, contorul de locații. Conversii nondistructive (si operatorii specifici)

7. Directive standard pentru definirea segmentelor. Directive pentru definirea datelor. Directive de generare a datelor. Directivele EQU și INCLUDE

# Structura CURS

8. Instructiuni ale limbajului de asamblare: instructiuni de transfer, conversii, operatii aritmetice cu semn si fara semn, operatii de deplasare si rotire de biti, operatii logice pe biti

9. Impactul reprezentării little endian asupra accesării datelor. Constante de tip string. Reprezentare în memorie și utilizare în cadrul unor instrucțiuni de transfer.

10. Instructiuni de salt conditionat si neconditionat, instructiuni de ciclare, instructiuni pe siruri. Conceptul de depășire și modul în care arhitectura 80x86 reactioneaza la aparitia acestei situatii

11. Reprezentarea instrucțiunilor mașină. Formatul intern al unei instrucțiuni. Prefixe de instrucțiuni.

12. Programarea multimodul ASM-ASM: directivele de import-export GLOBAL si EXTERN. Legarea de module scrise in limbaj de asamblare si modul de comunicare intre acestea

13. Implementarea apelului de subprograme. Convenții de apel: CDECL și STDCALL, cod de apel, cod de intrare, cod de iesire la nivelul limbajelor de nivel inalt vs. limbaj de asamblare

14. Legarea de module NASM cu module scrise în limbaje de nivel înalt (studiu de caz – programarea C). Exemple și discuții pentru apeluri recursive

# METODOLOGIE

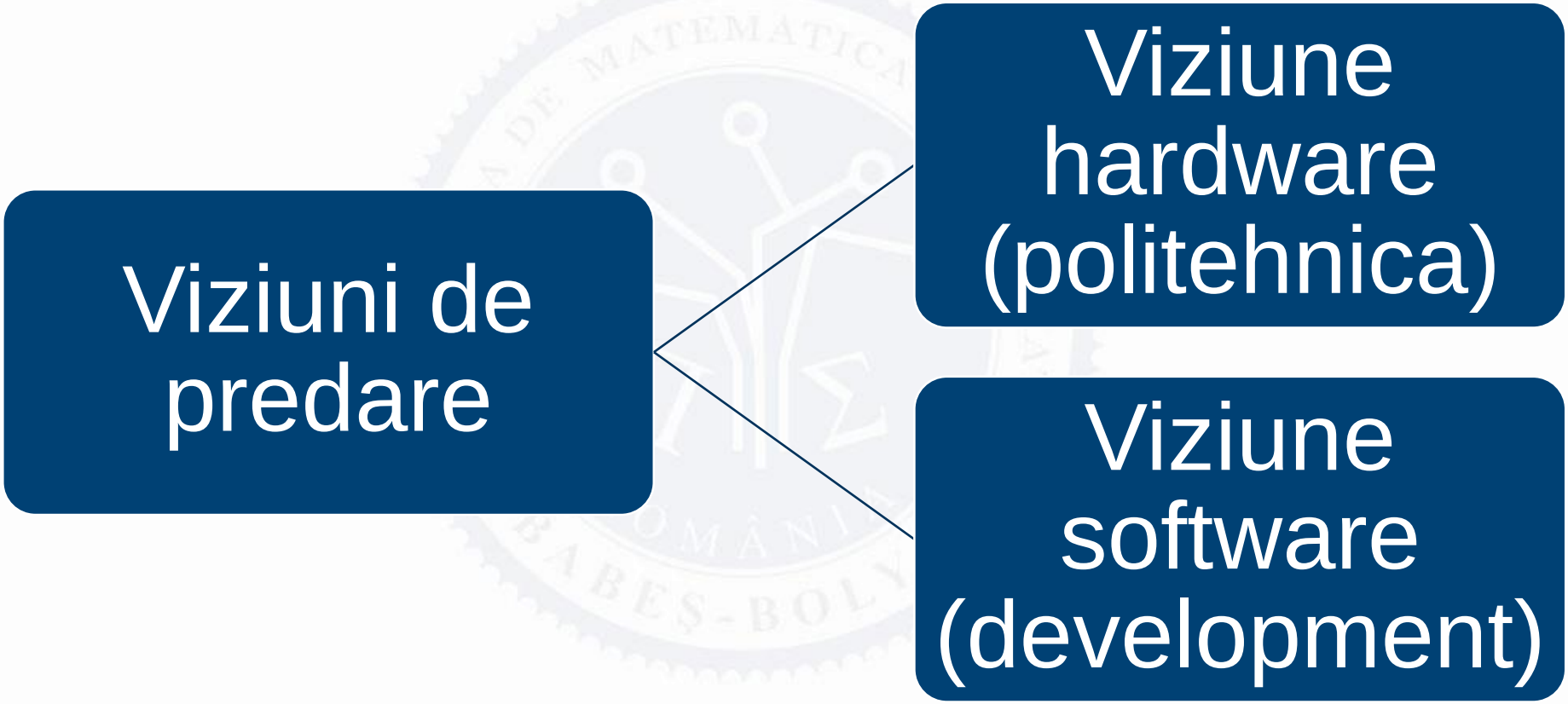
- Software utilizat: NASM
- Debugger: OLLYDBG – deoarece limbajul de asamblare nu are instrucțiuni de intrare sau de ieșire
  - asm16 – are, dar nu se mai folosesc
  - asm32 – folosește apeluri sistem C

# EVALUARE

Mai multe detalii pe site:
https://www.cs.ubbcluj.ro/~diana.sotropa/teaching/asc/

- EVALUARE:
  - 15% nota activitate laborator (minim nota 5)
  - 15% nota lucrare de control
    La CURS, 8 decembrie 2025
  - 15% nota probă practică (minim nota 5)
    5 - 18 ianuarie 2026
  - 55% nota examen scris (minim nota 5)
    19 ianuarie – 8 februarie 2026,
    21 ianuarie 2026
    a doua data de examen preferabil în perioada 26-30.01.2026
    16 - 22 februarie 2026
    16 februarie 2026
- SEMINAR:
  - se admit maximum 2 absențe NEMOTIVATE
- LABORATOR:
  - se admit maximum 2 absențe NEMOTIVATE
  - teste practice în săptămânile 5 și 9

# DE CE STUDIEM ASC?



# DE CE ESTE IMPORTANT ACEST CURS?

- De ce limbajele de programare la nivel general (C, C++) au limitarile pe care le au?
- Fundamentul tehnic pentru orice programator serios

– Nu poți optimiza un program dacă nu înțelegi cum funcționează procesorul și memoria.

- Baza pentru programarea de sistem (C, ASM, OS development)

– Înveți cum interacționează codul cu hardware-ul la nivelul cel mai de jos.

- Înțelegerea performanței aplicațiilor

– De ce unele bucle sunt lente? Cum afectează accesul la memorie timpul de execuție?

- Control total asupra execuției codului

– Limbajul de asamblare îți oferă o precizie imposibilă în limbajele de nivel înalt.

- Introducere în domenii precum securitate cibernetică, reverse engineering, exploatări

– Mulți hackeri etici, analiști de malware și ingineri de securitate încep de aici.

- Aplicabil în embedded systems și IoT

– Dispozitivele mici (ex: Arduino, ESP32) au nevoie de cod eficient scris într-un limbaj de programare low-level.

- Te ajută să înțelegi limitările hardware-ului

– Ce înseamnă „buffer overflow”? Ce face un „stack frame”? Acestea nu mai sunt mistere.

- Punte între hardware și software

– Poți comunica eficient atât cu dezvoltatorii hardware, cât și cu cei software

- Pregătire pentru laboratoare și proiecte complexe

– Fără această bază, laboratoarele din anii următori (ex: Sisteme de Operare) vor părea mult mai dificile.


FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
UNIVERSITATEA BABEȘ-BOLYAI

Str. Mihail Kogălniceanu nr. 1
Cluj-Napoca, Cluj, România

**www.cs.ubbcluj.ro**