Arhitectura Sistemelor de Calcul

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

Facultatea de Matematică și Informatică
Universitatea Babeș-Bolyai

Arhitectura microprocesoarelor X86 (IA-32)

Arhitectura Microprocesoarelor x86 (IA-32)

Cu ce categorii de informație ați lucrat la nivel de limbaj de programare?

Arhitectura Microprocesoarelor x86 (IA-32)

- INSTRUCȚIUNI
- DATE (tip de date)
- ADRESĂ (calcul de adrese)

Arhitectura Microprocesoarelor x86 (IA-32)

- Limbaj mașină:
  - Un limbaj numeric înțeles în mod specific de procesorul unui computer (CPU)
- Toate procesoarele x86 înțeleg un limbaj comun al mașinii

- Limbajul mașină – set de numere
- Limbajul de asamblare – instrucțiuni inteligibile

- Fiecare instrucțiune în limbajul de asamblare corespunde unei instrucțiuni unice în limbajul mașină

Arhitectura Microprocesoarelor x86 (IA-32)

- programarea microprocesoarelor compatibile cu procesoarele Intel IA-32
- procesor x86 => procesor Intel 80386 (1985) => registrii pe 32 de biți
- nu este portabil

Arhitectura Microprocesoarelor x86 (IA-32)

- Avantaje:
  - libertate de declarare sau transfer de date
  - gamă largă de operatori și expresii de adrese
- Dezavantaje:
  - Un număr mare de detalii necesare a fi stabilite înainte de a scrie efectiv cod semnificativ

Arhitectura Microprocesoarelor x86 (IA-32)

- Microprocesorul
  - Are viteză
  - Combină comenzi de bază (mai multe instrucțiuni de bază pe care procesorul le poate executa)
  - Orice comandă în orice limbaj de programare este ***tradusă*** într-un set de instrucțiuni pe care procesorul le poate executa
  - Orice program executabil, indiferent de unde provine, se poate vedea în ce se traduce

Astfel putem vedea limitele limbajului de programare

De ce trebuie să declar variabile?

- Orice operație sau instrucțiune este MAXIM binară

Structura microprocesorului

Care este unitatea primară de reprezentare a informației?

Care este unitatea primară de acces la informație?

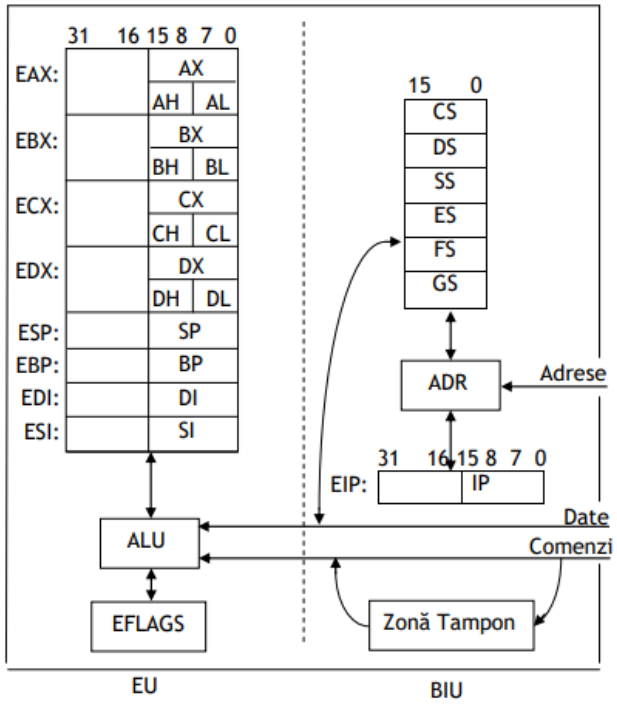
Care este diferența dintre REPREZENTARE și INTERPRETARE?

Structura microprocesorului

Microprocesor x86

EU (Executive Unit)
- execută instr. maşină prin intermediul componentei ALU

BIU (Bus Interface Unit)
- pregăteşte execuţia fiecărei instrucţiuni maşină
- citeşte o instrucţiune din memorie, o decodifică şi calculează adresa din memorie a unui eventual operand
- configuraţia rezultată este depusă într-o zonă tampon cu dimensiunea de 15 octeţi, de unde va fi preluată de EU

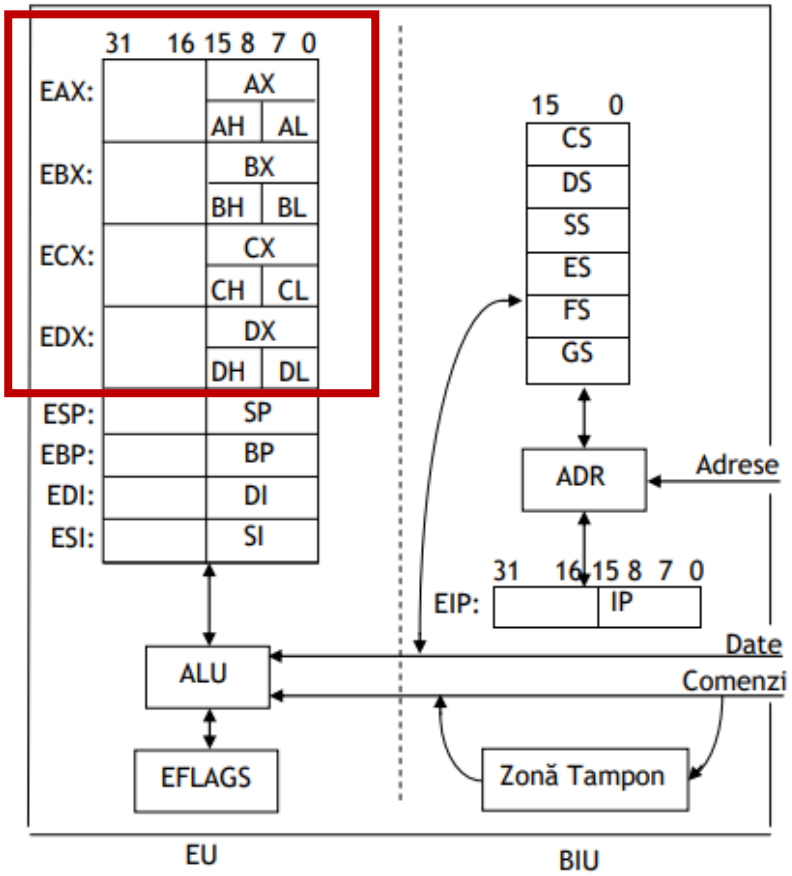


EU si BIU lucrează **în paralel**

În timp ce EU execută instrucţiunea **curentă**, BIU pregăteşte instrucţiunea **următoare**.

Cele două acţiuni sunt **sincronizate** - cea care termină prima aşteaptă după cealaltă.

Regiştrii generali EU



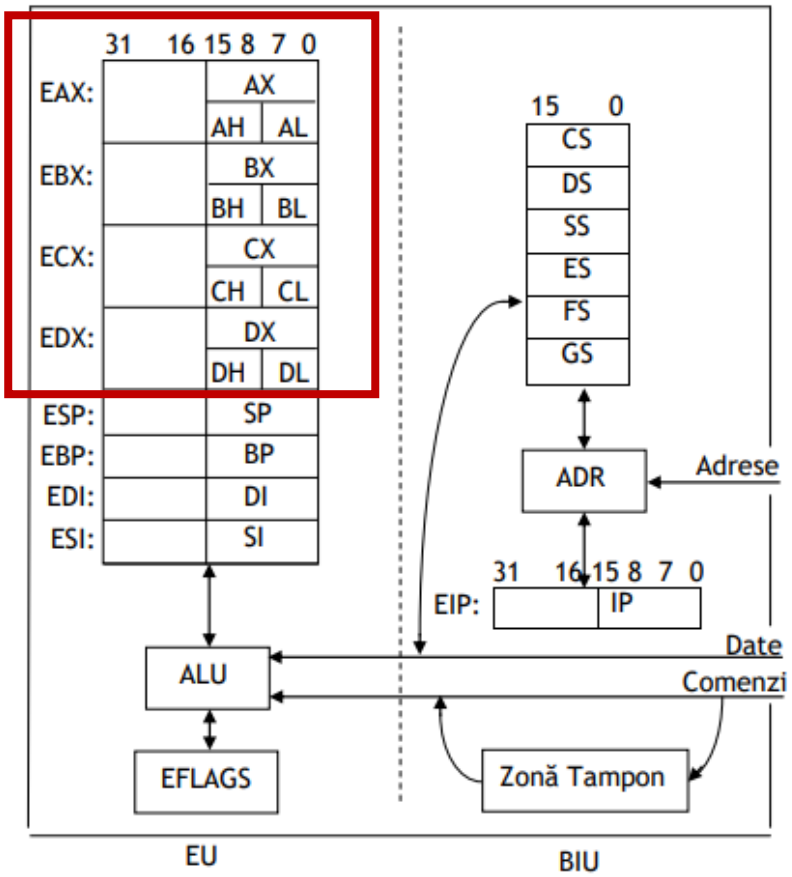
Cum stie procesorul sa distingă între date (variabile / constante), coduri de comenzi sau adrese?

Canale de transmisie (magistrale)

- **A – BUS** – magistrala de adrese
- **C – BUS** – magistrala de comenzi
- **D – BUS** – magistrala de date

Ce înseamnă că un calculator / procesor e pe N biți ?

Regiştrii generali EU



- EAX - **registru acumulator**
  - este folosit de către majoritatea instrucţiunilor ca unul dintre operanzi.
- EBX - **registru de bază**
  - provine ca denumire de la *Base Register* folosit in aceasta acceptiune la programarea pe 16 biti
- ECX - **registru contor**
  - pentru instructiuni care au nevoie de indicaţii numerice
- EDX - **registru de date**
  - împreună cu EAX se foloseşte în calculele ale căror rezultate depăşesc un dublucuvânt (32 biţi).

Tipuri de date

- BYTE (8 biți)
- WORD (2 octeți, 16 biți)
- DWORD (4 octeți, 32 biți)
- QWORD (8 octeți, 64 biți)

De ce există QUADWORD?

De ce există QUADWORD?

- La adunare:

  op1 (m cifre) +
  op2 (n cifre)
  ---
  suma lor are **max(m,n) + 1 cifre**

- La inmulțire:

  op1 (m cifre) *
  op2 (n cifre)
  ---
  produsul lor are **m+n cifre**

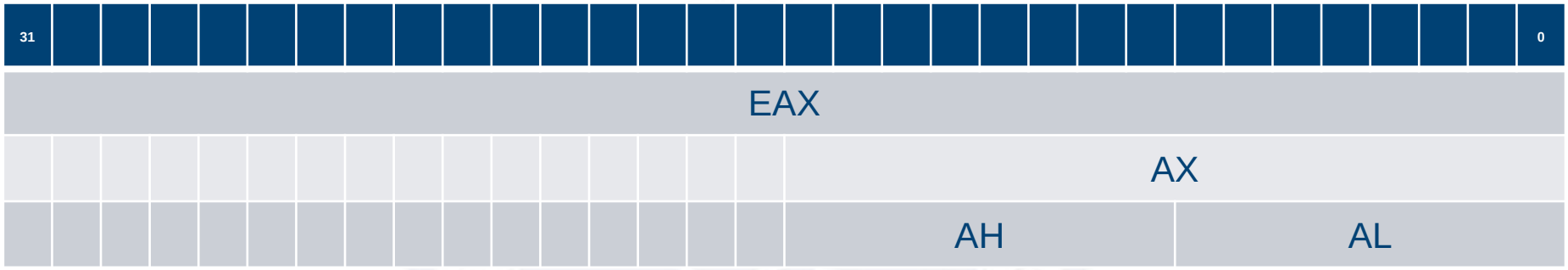
Consecințe la nivelul arhitecturii

byte + byte => byte
word + word => word
dword + dword => dword

byte * byte => word
word * word => dword
dword * dword => **qword**

EDX : EAX (qword)	
EDX	EAX
Jumătatea superioară (32 biți)	Jumătatea inferioară (32 biți)

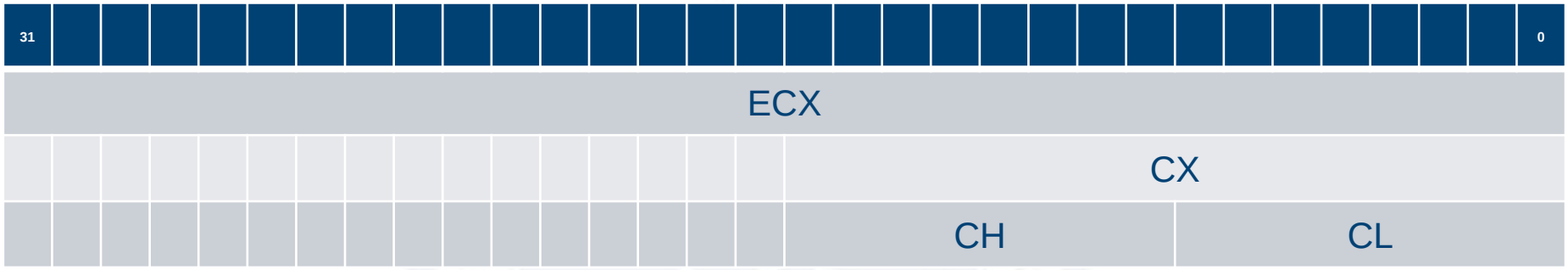
Regiştrii generali EU



De ce începe numerotarea biților de la 0?

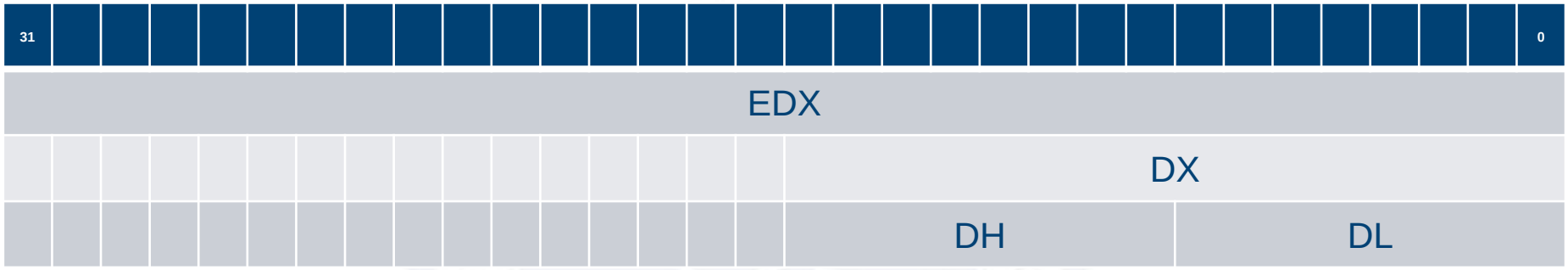
De ce se face numerotarea de la dreapta la stânga?

Regiştrii generali EU



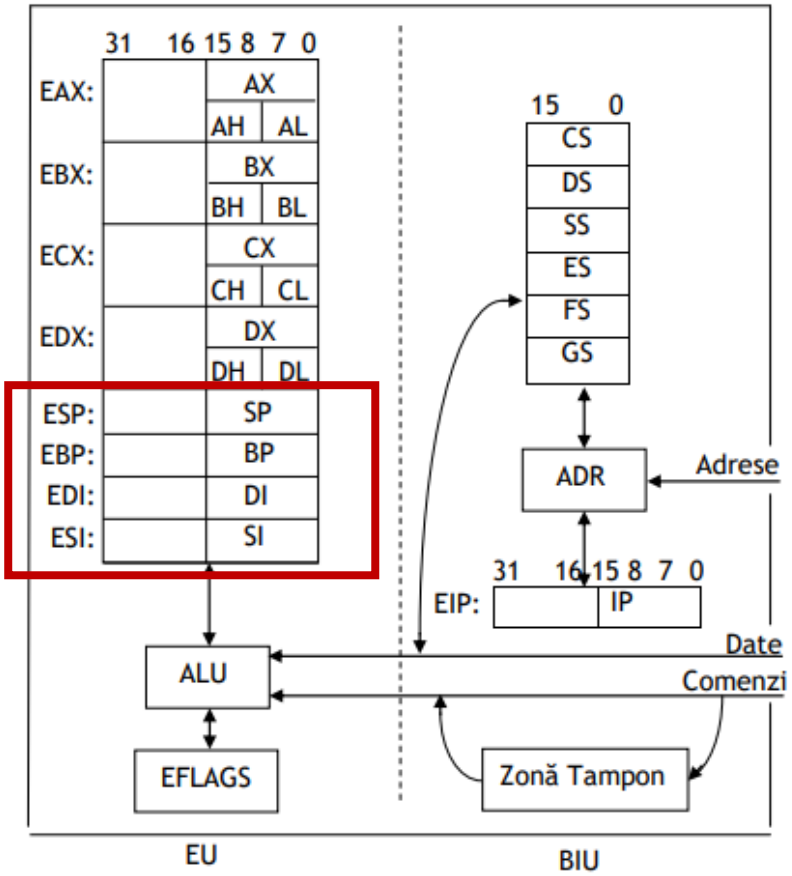
Ce numărăm în cadrul limbajelor de programare?

Regiştrii generali EU



Cum putem accesa partea superioară a regiștrilor EAX, EBX, ECX EDX?

Regiştrii generali EU – regiștrii de stivă

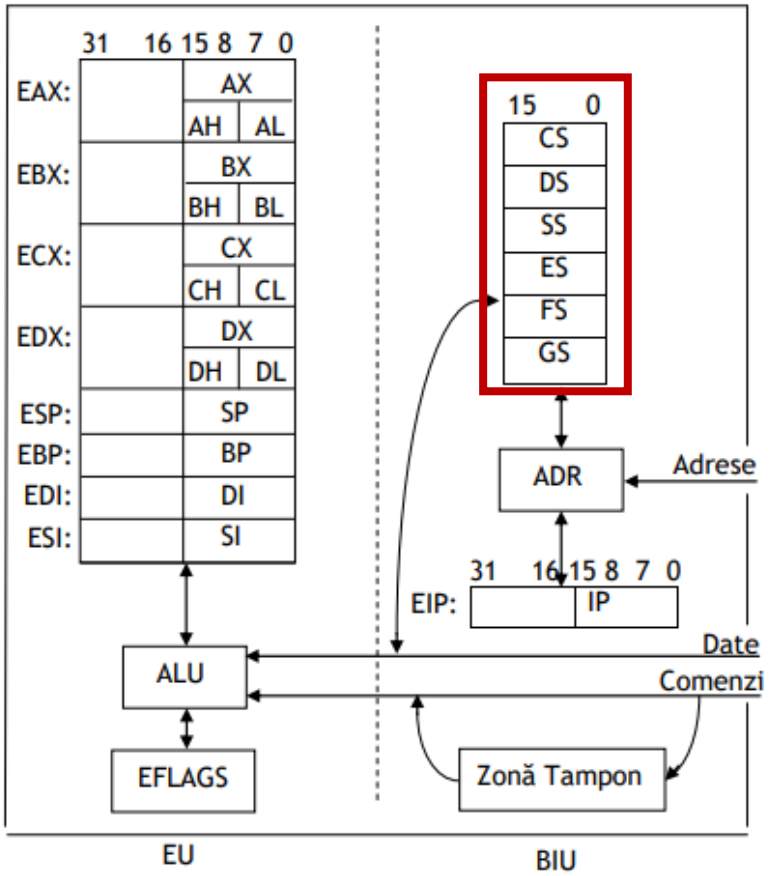


- ESP – **registru de stiva** *(Stack Pointer)*
  - punctează spre elementul ultim introdus în stivă (elementul din vârful stivei).
- EBP – **registru de stiva** *(Base pointer)*
  - punctează spre primul element introdus în stivă (indică baza stivei).

Ce este stiva? Prin ce diferă stiva de coadă?

Ce tipuri de date avem?

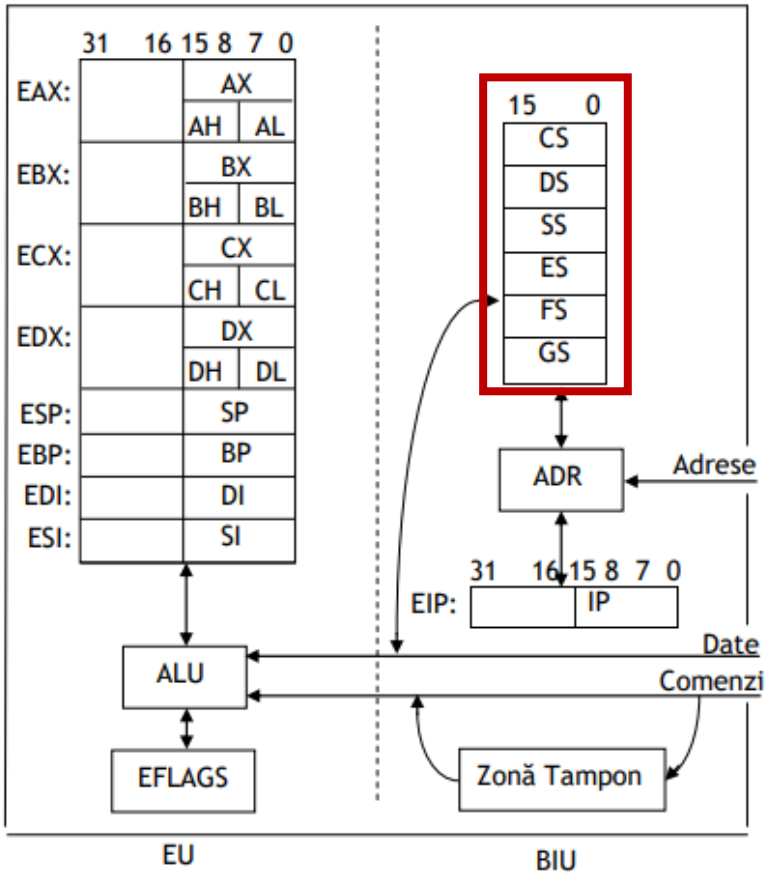
Regiştrii de segment



- **CS** - *Code segment*
- **DS** - *Data Segment*
- **SS** - *Stack Segment*
- **ES** - *Extra Segment*
- **FS**
- **GS**

De ce procesorul se ocupă de stivă cu 3 regiștrii și nici unul nu face referire la coadă, heap etc.?

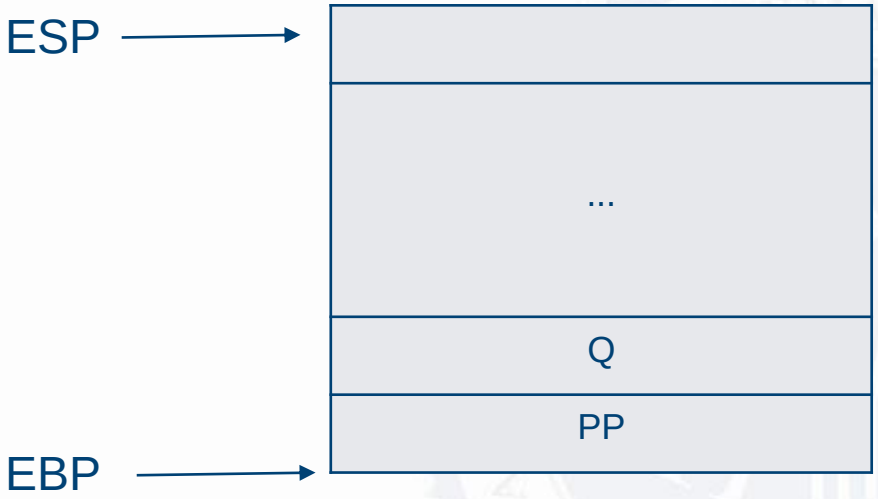
Regiştrii de segment



- **CS** - *Code segment*
- **DS** - *Data Segment*
- **SS** - *Stack Segment*
- **ES** - *Extra Segment*
- **FS**
- **GS**

De ce procesorul știe de LIFO și nu e interesat de FIFO?

Stiva de execuție (runtime stack)



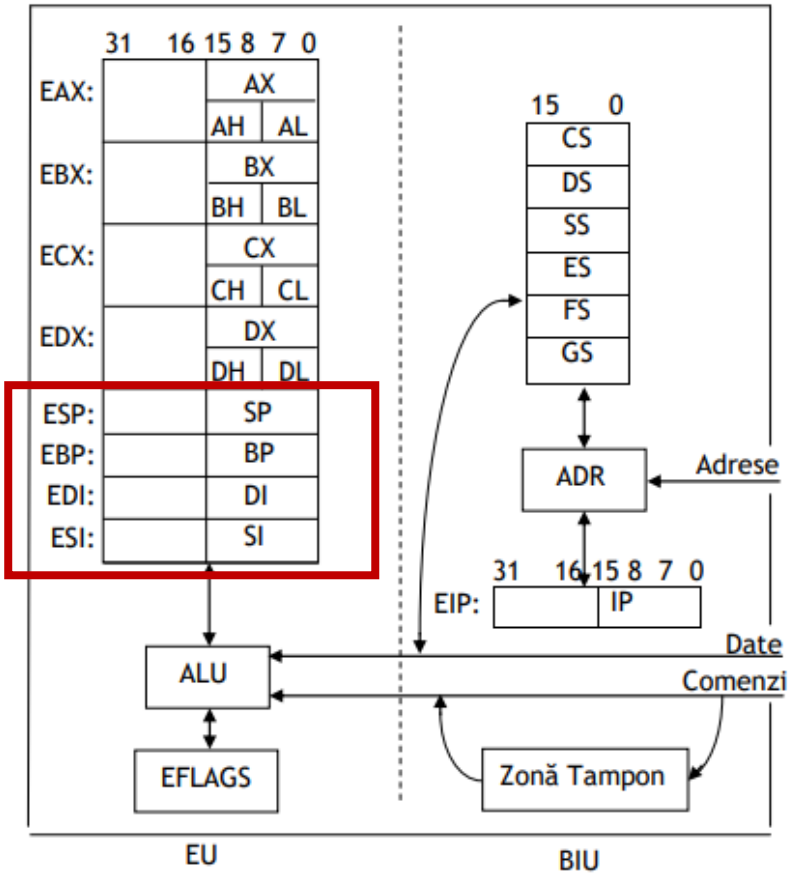
Se suspendă PP și se activează funcția Q

Se activează programul principal

Necesar pentru a ști unde a început programul

Rostul limbajelor de asamblare este legătura cu limbajele de programare de nivel înalt (ex. Programare C + ASM)

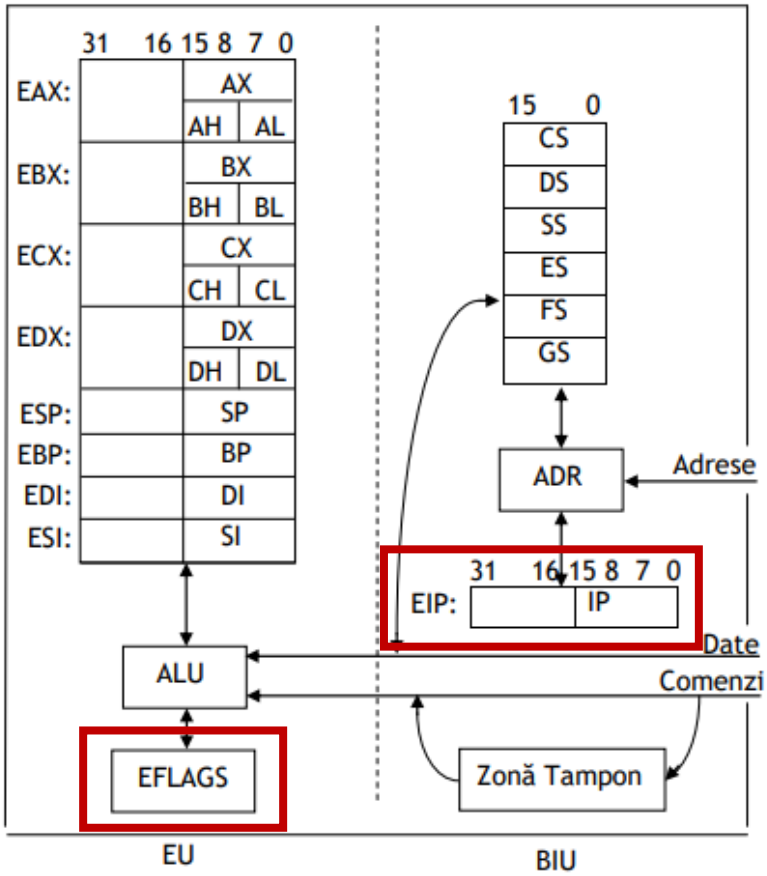
Regiştrii generali EU – regiștrii de index



- EDI şi ESI – **regiştrii de index** *(Destination Index şi Source Index)*
  - utilizaţi de obicei pentru accesarea elementelor din şiruri de octeţi sau de cuvinte.

a [5432] – cine e indexul?

Regiștrii EFLAGS și EIP



- EFLAGS
  - O configuraţie a registrului de flaguri indică un rezumat sintetic a execuţiei fiecărei instrucţiuni.
- EIP
  - conţine adresa următoarei instrucțiuni de executat

Arhitectura Microprocesoarelor x86 (IA-32)

- Reprezentarea datelor:
  - Cea mai mică unitate de stocare a informației este BIT-ul (0 sau 1)
  - **byte** (octet) = 8 biți
  - **word** (cuvânt) = 2 octeți = 16 biți
  - **doubleword** (dublucuvânt) = 2 cuvinte = 4 octeți = 32 biți
  - **quadword** (tetracuvânt) = 2 dublucuvinte = 4 cuvinte = 8 octeți = 64 biți

Arhitectura Microprocesoarelor x86 (IA-32)

- Cuvinte rezervate:
  - Numele de instrucțiuni: MOV, ADD, MUL, DIV
  - Numele regiștrilor
  - Numele directivelor
  - Operatori
- Identificatori (variabile, constante, proceduri, etichete):
  - pot contine maxim 247 caractere, litere, cifre și caracterele _, $, $$, #, @, ~, . și ?
  - în NASM sunt case sensitive
  - primul caracter trebuie sa fie o litera (A..Z, a..z), _ și ?
  - nu poate fi un cuvant rezervat

  - Pentru denumirile implicit parte a limbajului, cum ar fi cuvintele cheie, mnemonicile și numele regiștrilor, nu se diferențiază literele mari de cele mici (acestea sunt case insensitive).

Arhitectura Microprocesoarelor x86 (IA-32)

- Directive:
  - sunt indicaţii date asamblorului în scopul generarii corecte a octetilor

    Ex: relaţii între modulele obiect, definirea unor segmente, indicaţii de asamblare condiţionată, directive de generare a datelor *(pot defini variabile, macro-uri, proceduri sau pot atribui nume segmentelor de memorie)*

Directive pentru definirea variabilelor:
`DB, DW, DD, DQ`

Directive pentru definirea constantelor:
`EQU`

Exemplu: `zece EQU 10`

Arhitectura Microprocesoarelor x86 (IA-32)

- Etichete:
  - Un identificator care reprezintă adresa unor instrucțiuni sau date
    - Etichete de date: identifică locația unei variabile

```
count DW 100
array DW 1024, 2048
         4096, 8192
```

    - Etichete de cod

```
L1: mov AX, BX
L2:
mov AX, BX
...
jmp L2
```

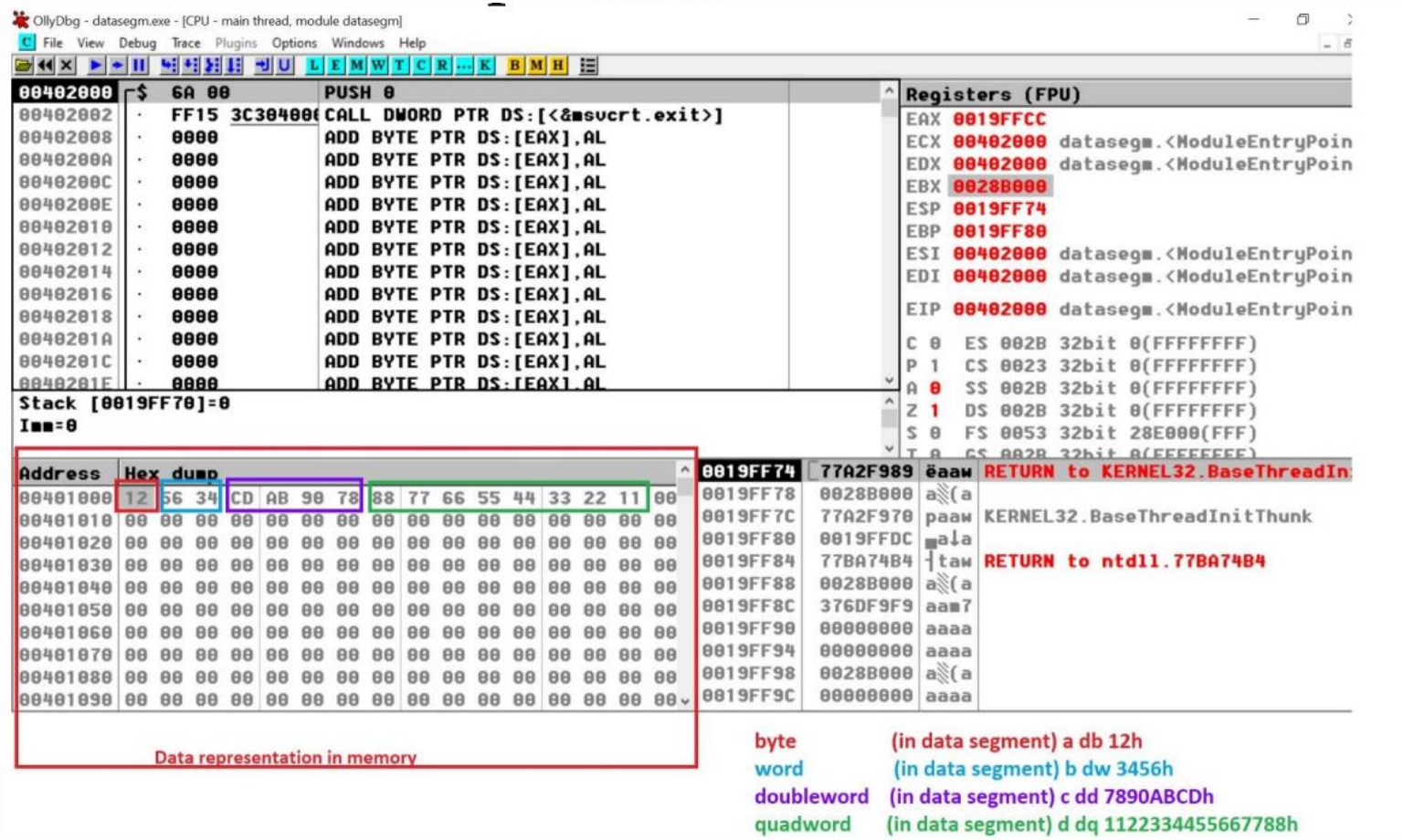
Arhitectura Microprocesoarelor x86 (IA-32)

- Reprezentarea datelor în memorie: little endian
  - O adresă referă o locație de memorie
  - Procesorul x86:
    - permite ca fiecare octet să fie accesibil prin adresă
    - stochează și furnizează datele din memorie folosind reprezentarea little endian
      - Cel mai puțin semnificativ octet este stocat în memorie la ce mai mica adresă alocată
      - Octeții rămași sunt stocați în memorie în octeții următori

Exemplu:

```
a DB 12h                    ;|12h|
b DW 3456h                  ;|56h|34h|
c DD 7890ABCDh              ;|CDh|Abh|90h|78h|
d DQ 112233445566778h       ;|88h|77h|66h|55h|44h|33h|22h|11h|
```

Arhitectura Microprocesoarelor x86 (IA-32)



Arhitectura Microprocesoarelor x86 (IA-32)

- Instrucțiuni:
  - Sunt translatate de către asamblor în limbaj mașină și apoi sunt încărcate și executate de către CPU la runtime
  - Formatul unei linii sursă

`[eticheta[:]][prefixe][mnemonică][operanzi][;comentariu]`

```
NUME_INSTRUCTIUNE destinatie, sursa
```

destinatie = primul operand
sursa = al doilea operand

Alte formate de instructiuni:

```
NUME_INSTRUCTIUNE operand1, openrand2
NUME_INSTRUCTIUNE operand
NUME_INSTRUCTIUNE
```

Arhitectura Microprocesoarelor x86 (IA-32)

- Un program în limbaj de asamblare este format din segmente logice:
  - Segmentul de cod: instrucțiuni
  - Segmentul de date: variabile
  - Stiva: parametrii procedurilor, variabile locale, adrese de revenire

- Instrumente necesare:
  - **Notepad++**: scrierea fișierului sursă în limbaj de asamblare (.asm)
  - **NASM**: asamblor, adică programul care citește fișierul sursă și produce fișierul obiect (.obj)
  - **ALINK**: link-editor, adică programul care citește mai multe fișiere obiect și produce fișierul executabil (.exe)
  - **OllyDbg**: debugger

Abrevieri uzuale

Abreviere	Semnificatie
`reg`	Un registru pe 8, 16 sau 32 de biti AH, AL, BH, BL, CH, CL, DH, DL, AX, BX, CX, DX, SI, DI, BP, SP, EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
`reg8, reg16, reg32`	Un registru identificat prin numarul de biti reg8: AH, AL, BH, BL, CH, CL, DH, DL reg16: AX, BX, CX, DX, SI, DI, BP, SP reg32: EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
`mem`	Un operand din memorie (o variabila)
`mem8, mem16, mem32`	Un operand din memorie identificat prin numarul de biti
`con`	Un operand imediat (constanta)
`con8, con16, con32`	Un operand imediat identificat prin numarul de biti

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
UNIVERSITATEA BABEȘ-BOLYAI

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

www.cs.ubbcluj.ro