Programarea in Limbaj de Asamblare

Pe procesoare de la Intel cu arhitectura pe 32 de biti (IA-32)

```
;program.asm
.386
.model flat,stdcall;

includelib msvcrt.lib
extern exit:proc
public start

.data
V1 DB 1,2,3
CAR DB "123"
V2 DD 0A234Bh
V3 DW 1,12h,0123h
V4 DQ 1.2
```

```
.code
start:
LEA ESI, V1
MOV ECX, 3
etloop:
     MOV AL,[ESI]
     INC AL
     MOV [ESI],AL
LOOP etloop
push 0
call exit
end start
```

Mai multe tipuri de asambloare: MASM, TASM, NASM, etc.

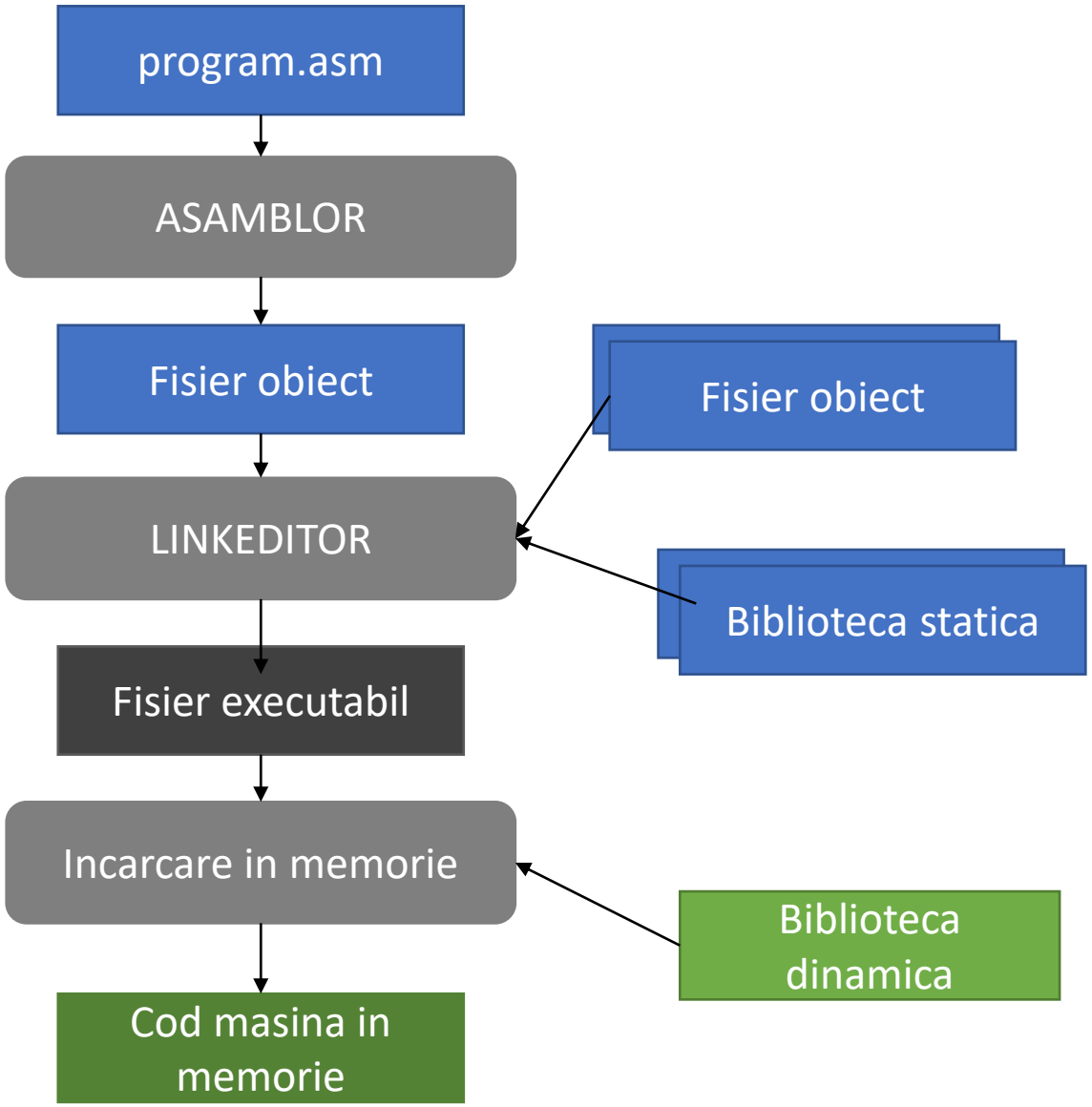
Sintaxa difera in functie de asamblorul folosit

Exista diferente in felul in care scrii codul pentru diferite SO (Linux, Windows, etc)

In functie de SO, difera functiile sistem si modul in care se apeleaza

Trebuie sa iei in considerare modul de operare al procesorului (32 bit, 64 bit)

In functie de procesor/arhitectura, instructiunile difera



```
;program.asm
.386
.model flat,stdcall;

includelib msvcrt.lib
extern exit:proc
public start

.data
V1 DB 1,2,3
CAR DB "123"
V2 DD 0A234Bh
V3 DW 1,12h,0123h
V4 DQ 1.2
```

```
.code
start:
LEA ESI, V1
MOV ECX, 3
etloop:
     MOV AL,[ESI]
     INC AL
     MOV [ESI],AL
LOOP etloop
push 0
call exit
end start
```

Arhitecturi Intel - scurt istoric

- **Arhitectura pe 16 biti (1978): 8086**, 8088
  - Adresare pe 20 de biti, magistrala de date de 16 biti
  - Memorie segmentata (1MB, segmente de 64KB), reistri segment de 16 biti + pointer de 16 biti pentru offset
  - Operare in mod real
- Introducerea modului protejat (1982): Intel 286
  - Adresare pe 24 de biti, memorie de 16MB, segmentata
  - Moduri de operare: protejat, real, virtual
- **Arhitectura pe 32 de biti IA-32 (1985): Intel 386**
  - Adresare si date pe 32 de biti
  - Memorie de 4GB cu segmentare, paginare
  - Compatibilitate cu arhitectura pe 16 biti
  - Moduri de operare: protejat, real, virtual
  - Pipelining
- Intel 486 ('89): imbunatatiri pipeline, FPU x87 integrat
- Pentium ('93): branch prediction, tehnologia MMX
- **Familia P6** ('95-'99): microarhitectura superscalara, tehnologia SSE, support multiprocesor
- **Pentium 4** (2000-2006): microarhitectura Netburst, tehnologia Hyperthreading, support pentru virtualizare
- 2005->
  - **Arhitectura Intel 64, Microarhitectura Intel Core, Tehnologia Multi-Core**, Smart Cache, Dynamic Power Coordination, etc.
- 2019->
  - **Intel Core Generatia 10**

Mediul de executie IA-32

- **Set de resurse** asigurat de processor pentru executia unui program
  - Executia instructiunilor
  - Stocarea codului
  - Stocarea datelor
  - Stocarea informatiilor legate de stare

- Mediul de executie contine
  - **Spatiu de adresare: 4GB**
  - **Registri (UCP, FPU x87, MMX, XMM, YMM)**
  - **Stiva**
  - **Porturi I/E**
  - Registri de control
  - Registri pentru managementul memoriei
  - Registri de depanare
  - Memory type range registers (MTRRs), Machine specific registers (MSRs), Machine check registers, Performance monitoring counters

Moduri de operare

- **Modul protejat**
  - Modul nativ al procesoarelor cu arhitectura IA-32
  - Mecanisme de protectie a memoriei
  - Mediu multi-tasking
  - Modul virtual 8086
- **Modul real**
  - Mediul de programare Intel 8086 cu extensii
  - Dupa pornire sau reset
- **Modul Management Sistem (SMM)**
  - Mecanism transparent care permite SO sa implementeze functii de power management si Securitate
  - Spatiu separat de adrese
  - Incepand cu Intel 386 SL

```
;program.asm
.386
.model flat,stdcall;

includelib msvcrt.lib
extern exit:proc
public start

.data
V1 DB 1,2,3
CAR DB "123"
V2 DD 0A234Bh
V3 DW 1,12h,0123h
V4 DQ 1.2
```

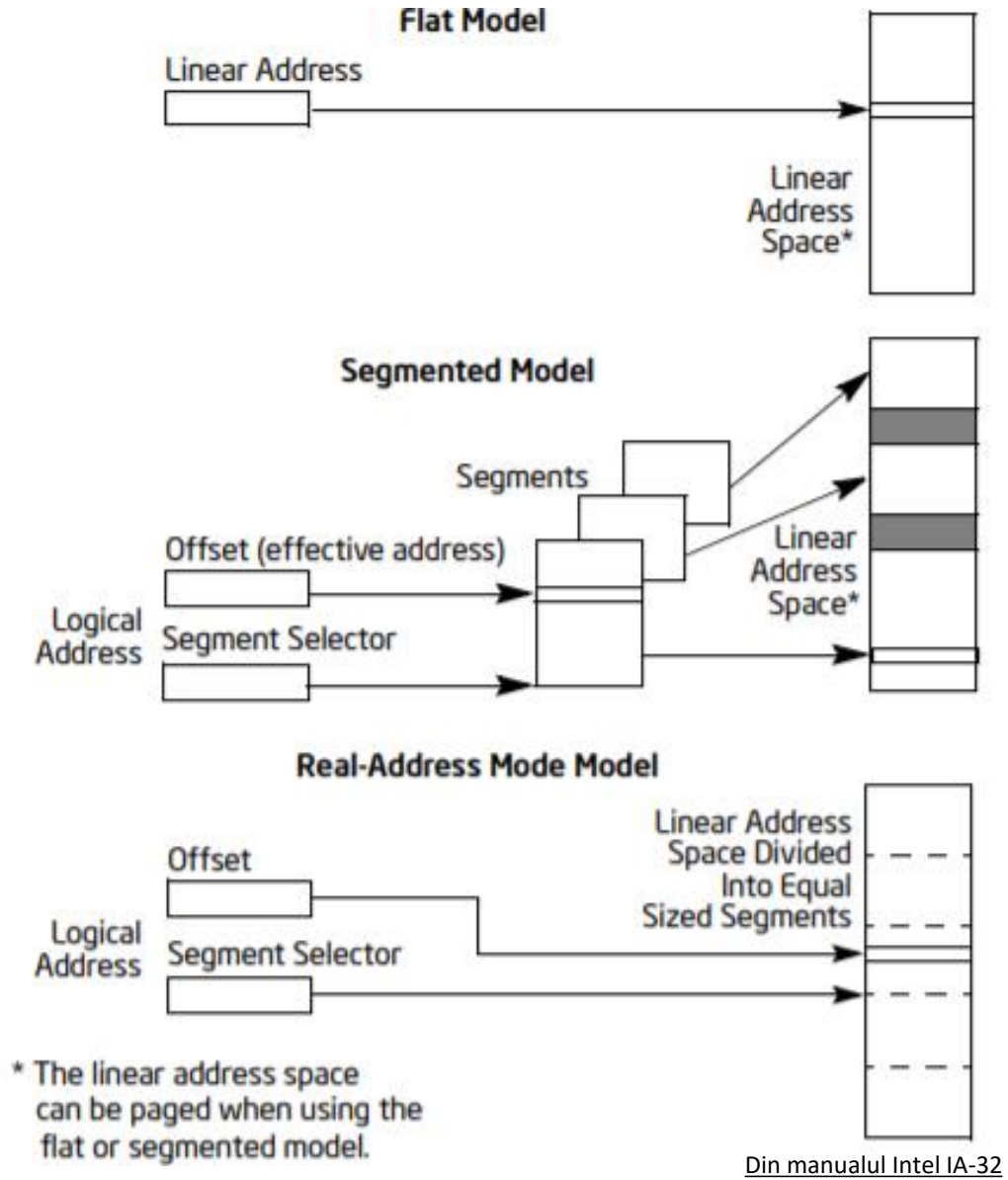
```
.code
start:
LEA ESI, V1
MOV ECX, 3
etloop:
     MOV AL,[ESI]
     INC AL
     MOV [ESI],AL
LOOP etloop
push 0
call exit
end start
```

Organizarea memoriei

- Memoria fizica
  - Accesata de processor pe magistrala
  - Organizata ca o secventa de octeti (8 biti)
  - Fiecare octet are asignata o adresa unica - adresa fizica
- Spatiul fizic de adresare
  - 2^{32}-1 (4GB)
  - Extins: 2^{36}-1 (64GB)
- Facilitati de management a memoriei
  - Segmentare
  - Paginare

Modele de memorie IA-32

- Modelul flat
  - Spatiu de adrese liniar (spatiu continuu de adrese)
  - Adresa liniara
- Modelul segmentat
  - Grup de spatii de adesare independente (segmente)
  - Segmente de: cod, date, stiva
  - Adresa logica: formata din selector si adresa de offset (SELECTOR:OFFSET)
- Modelul de adresare in mod real
  - Compatibilitate cu 8086



Din manualul Intel IA-32

Registri generali

- EAX - accumulator pentru operanzi si rezultat
- EBX - pointer de date in segmental de date DS
- ECX - numarator/contor pentru instructiuni pe siruri si loop
- EDX - pointer I/O
- ESI - pointer sursa pentru operatii cu siruri (source index)
- EDI - pointer destinatie pentru operatii cu siruri (destination index)
- EBP - pointer pentru date de pe stiva
- ESP - pointer de stiva (segmentul SS)

General-Purpose Registers

31 16	15 8	7 0	16-bit	32-bit
	AH	AL	AX	EAX
	BH	BL	BX	EBX
	CH	CL	CX	ECX
	DH	DL	DX	EDX
	BP			EBP
	SI			ESI
	DI			EDI
	SP			ESP

Din manualul Intel IA-32

Registri segment

Registri pe 16 biti: CS, DS, SS, ES, FS, GS

Contin un selector de segment (pointer special care identifica segmentul)

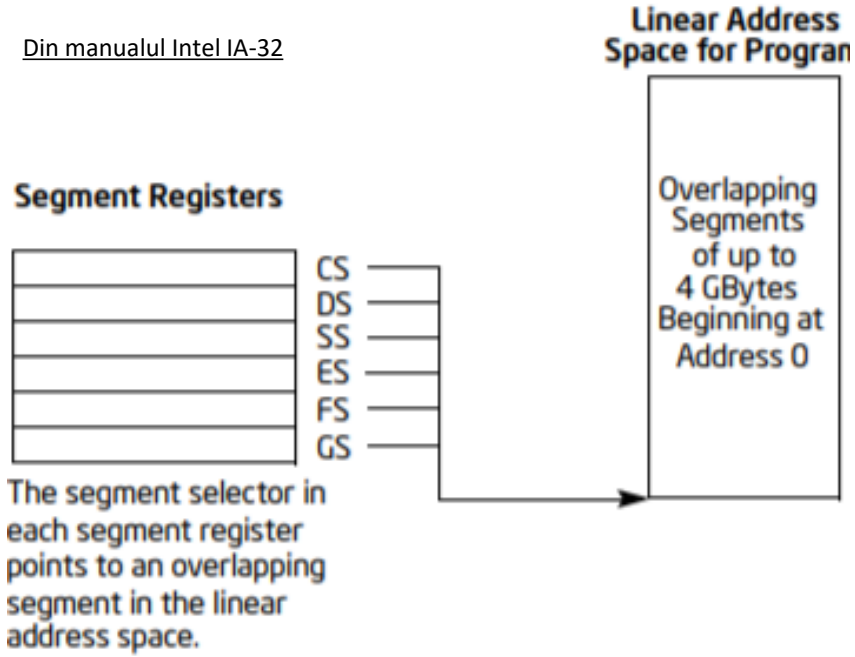
CS - segmentul de cod, nu poate fi incarcat explicit de o aplicatie

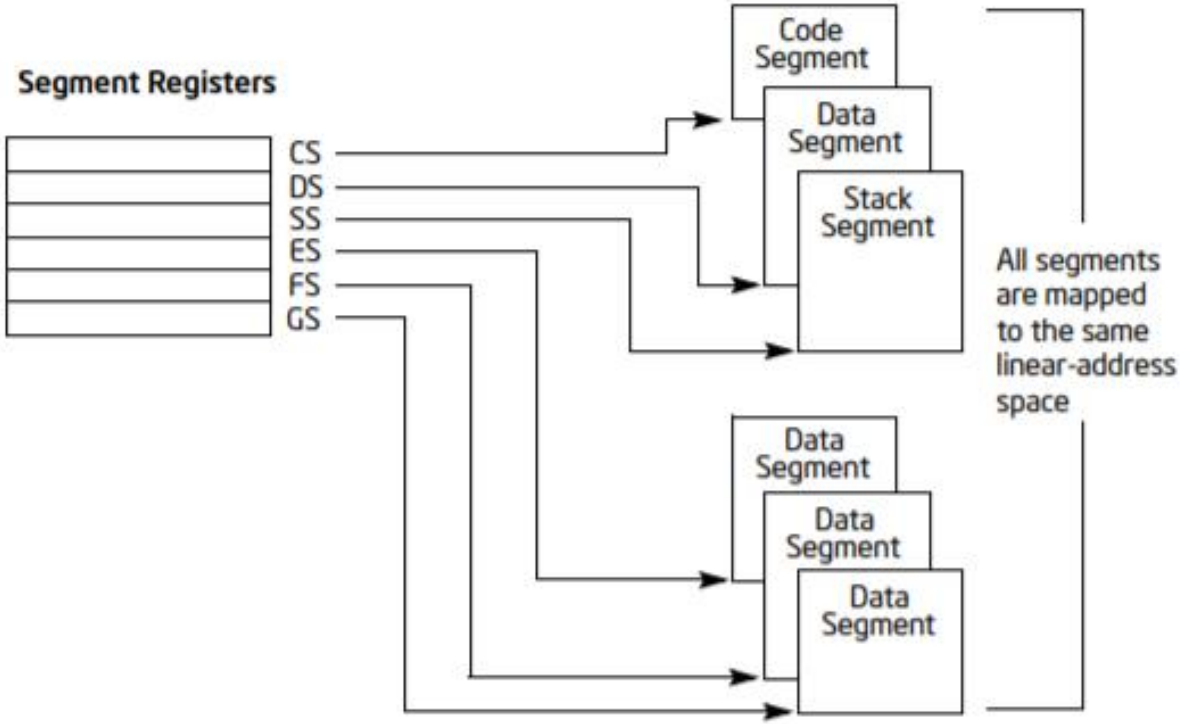
DS, ES, FS, GS - segmente de date, segmente de date aditionale incarcate de aplicatie

SS - segmentul de stiva, folosit implicit pentru operatii cu stiva, poate fi incarcat explicit, stive multiple

Modelul Flat (sus), Modelul segmentat (jos)

Din manualul Intel IA-32



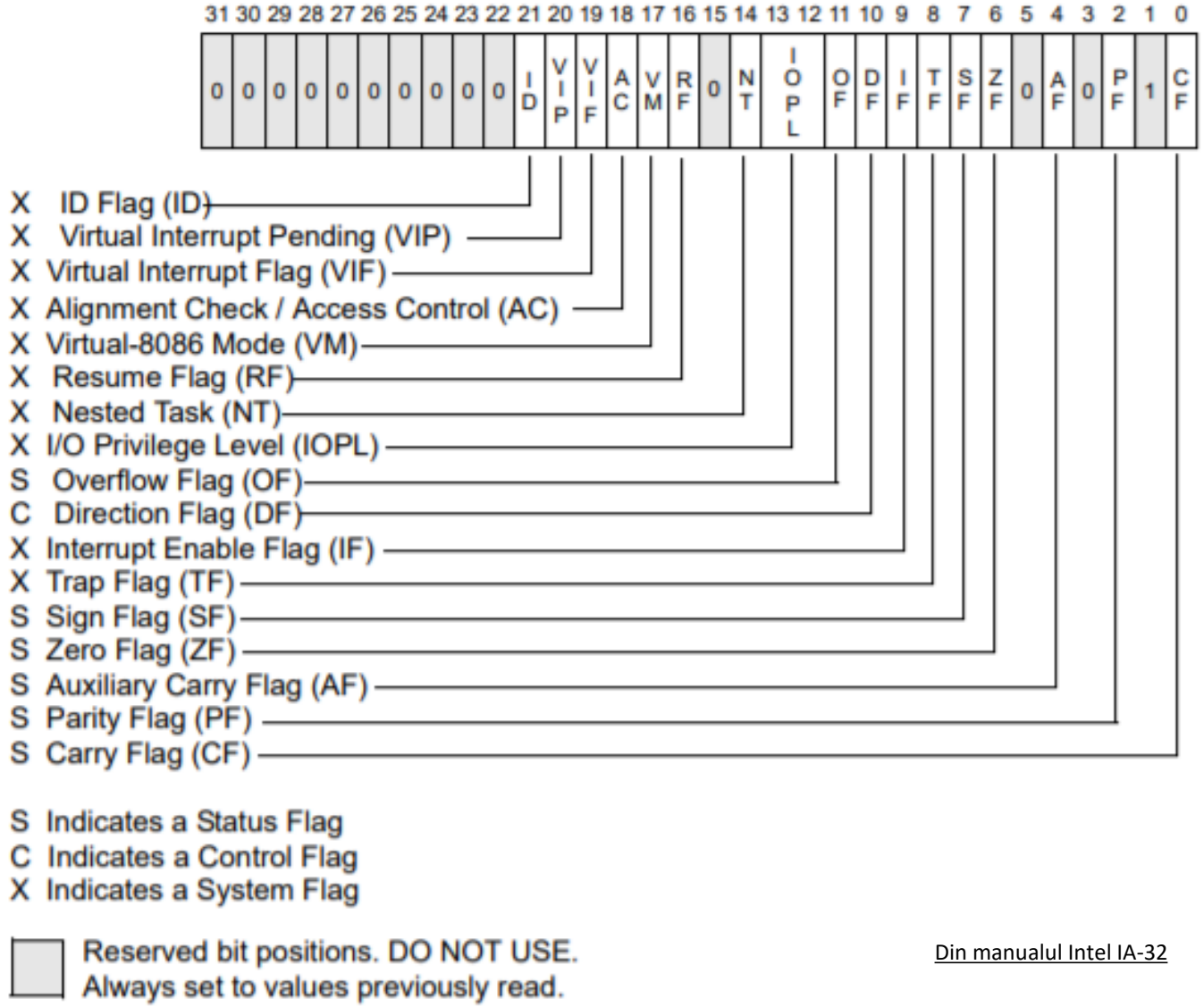


```
;program.asm
.386
.model flat,stdcall;

includelib msvcrt.lib
extern exit:proc
public start

.data
V1 DB 1,2,3
CAR DB "123"
V2 DD 0A234Bh
V3 DW 1,12h,0123h
V4 DQ 1.2
```

```
.code
start:
LEA ESI, V1
MOV ECX, 3
etloop:
     MOV AL,[ESI]
     INC AL
     MOV [ESI],AL
LOOP etloop
push 0
call exit
end start
```

eFlags

- **Stare**
  - **CF - carry flag** (=1 daca a existat imprumut sau transport la CMSB al rezultatului)
  - **PF - parity flag** (=1 daca rezultatul are un numar par de biti de 1)
  - **AF - auxiliary cary flag** (=1 daca a existat imprumut sau transport de la bitul 3 al rezultatului; folosit la aritmetica BCD)
  - **ZF - zero flag** (=1 daca rezultatul este 0)
  - **SF - sign flag** (= cu bitul de semn al rezultatului)
  - **OF - overflow flag** (=1 daca nr pozitiv e prea mare sau nr negative e prea mic ca sa incapa in destinatie; C2 cu semn)
- Control
  - **DF - direction flag** (=1 auto-decrement; =0 auto-increment; instructiuni pe siruri)
- Sistem
  - **IF - interrupt enable flag** (=1 procesorul raspunde la cereri de intrerupere)



Din manualul Intel IA-32

Pointerul de instructiuni EIP

- Contine adresa de offset, in segment curent de cod, a urmatoarei instructiuni de executat
- Se modifica
  - Automat pentru a trece la instructiunea imediat urmatoare(prin adunare) EIP=EIP+lungine instr. curenta
  - Prin instructiuni de salt

Instructiuni in cod masina

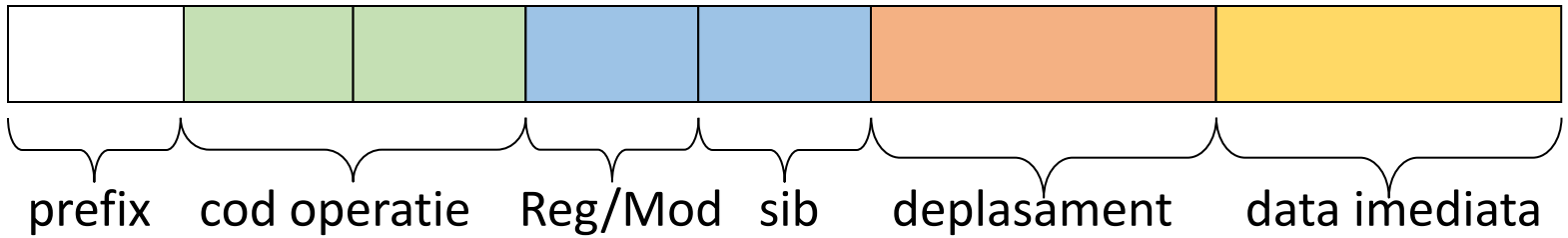
- **Varianta 1**: fara operanzi in codul instructiunii

  - operatia nu necesita operanzi sau operanzii sunt specificati implicit prin codul de operatie
- **Varianta 2**: un singur camp de operand in codul instructiunii

  - operatia este unara (un operand) sau al doilea operand este implicit
- **Varianta 3:** doua campuri de operanzi (sursa, destinatie)

- **Varianta 4:** trei campuri de operand (operand 1, operand 2, rezultat)

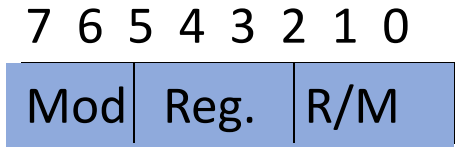
Formatul instructiunii - cod masina



- Formatul instr. Intel – variabil ca lungime si continut
- Semnificatia campurilor:
  - **Prefix**: unul sau mai multi octeti care preced instructiunea si modifica modul de executie al acesteia
    - tipuri: segment explicit, dim. adresa (16/32 biti), dim. operand (16/32 biti), repetare instr. , blocare acces (lock)
  - **Cod operatie**: 1..2 octeti, care codifica tipul de operatie
    - instr. setului de baza au codul operatiei pe 1 octet

Formatul instructiunii - cod masina

- **Reg/Mod :** specifica registrele folosite si modul de adresare adoptat



  - **Mod:** modul de adresare folosit (registru-registru, registru-memorie
  - **Reg:** codul registrului folosit
  - **R/M:** al doilea registru folosit sau modul de adresare a memoriei
- **sib**: scala-index-baza – detaliaza modul de adresare folosit
- **Deplasament:** contine o adresa pe 8, 16 sau 32 biti, folosita pentru calculul adresei unui operand
- **Data imediata:** contine o valoare constanta pe 8, 16 sau 32 de biti
- instructiunea cea mai scurta: 1 octet
- instructiunea cea mai lunga: ~ 16 octeti

Operanzi

- Dimensiuni **operanzi**: Byte (8 biti), Word (16 biti), DWord (32 biti), QWord (64 biti)
- Dimensiuni **adrese**: 16 biti (mod real), 32 biti (mod protejat)
- Operandul sursa poate fi
  - In codul instructiunii (data imediata)
  - Registru
  - Locatie de memorie
  - Port I/O
- Operandul destinatie (rezultatul) poate fi
  - Registru
  - Locatie de memorie
  - Port I/O

Locatia de memorie: specificarea adresei

SELECTOR: OFFSET(sau Adresa liniara)

(selector-16b;offset sau adr. Liniara 32b)

MOV ES:[EBX], EAX

Specificarea offsetului/adresei liniare:

Offset = Baza+Index*Scala+Deplasament

MOV EAX, [EBX+ESI*2+12h]

MOV EAX, [EBX][EDI]

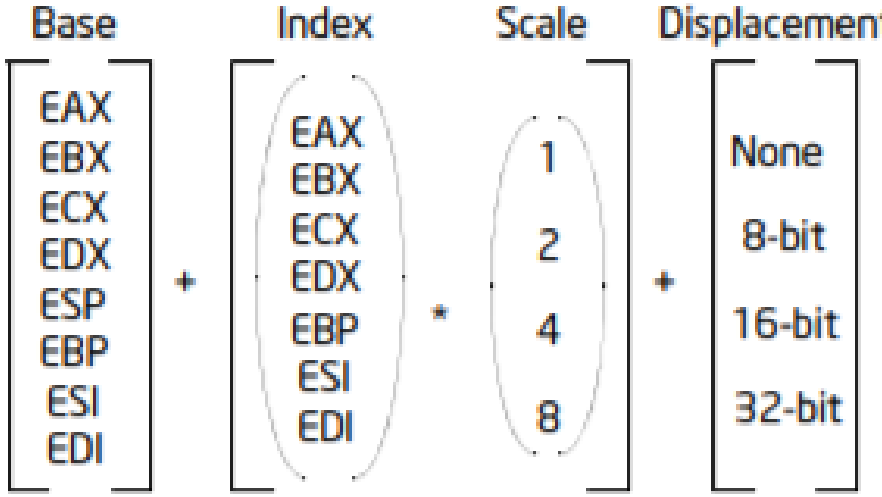
Baza - valoare intr-un registru general

Index - valoare intr-un registru general

Scala - valoare egala cu 2, 4 sau 8

Deplasament - valoare pe 8, 16 sau 32 de biti

Din manualul Intel IA-32



Deplasament

MOV EAX, [012345h]

Baza

MOV EAX, [ESI]

Baza+Deplasament

MOV EAX, [EBX+8]

Index*Scala+Deplasament

MOV EAX, [ESI*2+4]

Baza+Index+Deplasament

MOV EAX, [EBX+EDI+16]

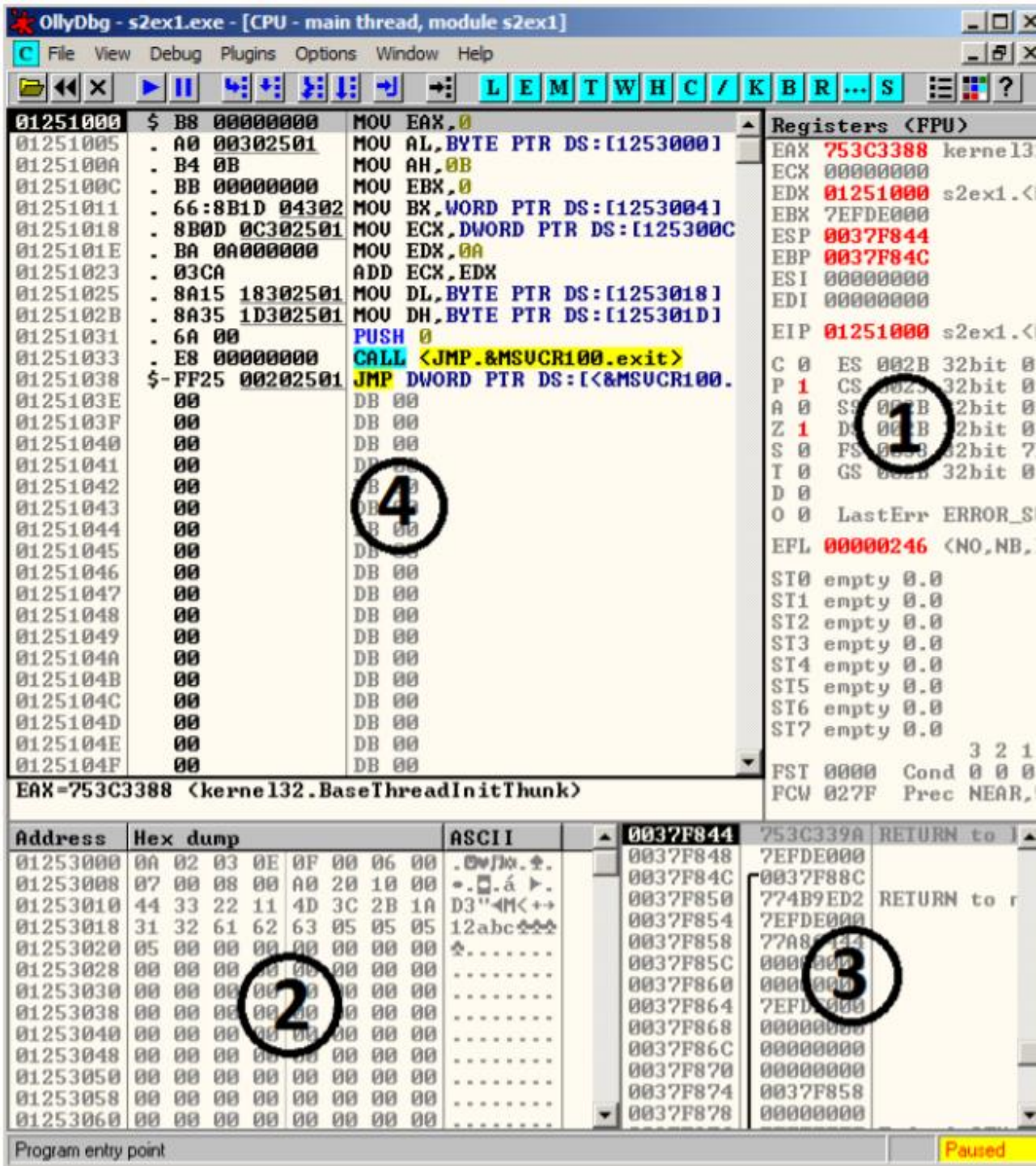
Access la date

- Date imediate
  - Procesorul citeste operandul din codul instructiunii
  - Timp de acces minim
- Registru
  - Procesorul citeste operandul din registru
  - Timp de acces minim
- Porturi I/O
  - Registre continute in interfetele de intrare/iesire HW
  - Access pe baza de adresa
  - Instructiuni speciale sau prin functii expuse de SO
- Memorie
  - Interna (memoria fizica)
    - Access pe baza de adresa
    - Ciclu de transfer pe magistrala
    - Timp de access mediu
  - Externa (ex: fisiere stocate pe HDD sau SSD)
    - Accesul se face prin functii expuse de sistemul de operare
    - Timp de access mare

Depanare

OllyDbg

- 1 - registri, eFlags, registri FPU, indicatori FPU
- 2 - Memoria de date
- 3 - Stiva
- 4 - Memoria de instructiuni



Exercitii

- Aratati cum apar valorile urmatoare, in memorie stiind ca sunt una dupa alta, incepand cu adresa 0: -5 (byte), '1' (ASCII), 16 (word), 1235h (dword), 1.2 (dword).