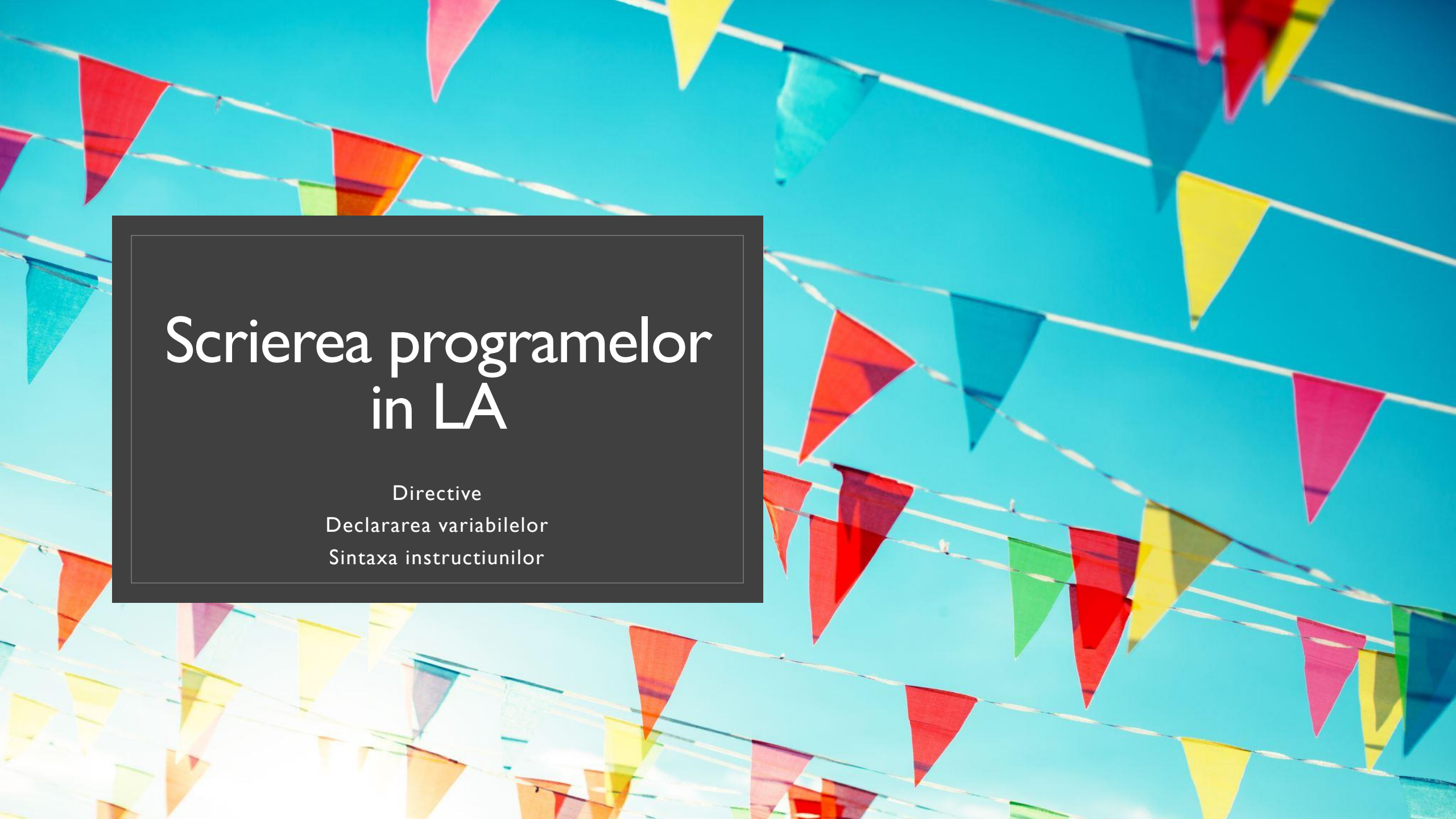


Scrierea programelor in LA

Directive

Declararea variabilelor

Sintaxa instructiunilor

```
;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
co EQU 25
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
```

Directive

- Pseudo-instructiune
- Directivele nu se executa
- Se folosesc pentru
  - Declararea variabilelor
  - Declararea segmentelor si procedurilor
  - Controlul procesului de asamblare, linkeditare si lansare a programului

Declararea variabilelor

- Utilizarea de nume simbolice in locul adreselor
- Rezervarea de spatiu in memorie si initializarea variabilelor
- Verificarea de tip a variabilelor (utilizarea lor corecta)
- Modul de declarare: prin directive

Tipuri de date

- **Tipuri de date fundamentale**: Byte (8b), Word (16b), Doubleword (32b), Quadword (64b), Double Quadword (128b)
- **Tipuri de date numerice**
  - Intregi cu si fara semn: intreg fara semn pe byte, intreg cu semn pe byte, etc.
  - Numere in virgula mobil: half precision (16b), simple precision (32b), double precision (64b), double extended precision (80)
- **Pointeri**
  - Near pointer (32b): numai offset
  - Far pointer (16+32b): selector:offset
- **Bit field**: secventa continua de biti (max 32b)
- **String:** secventa de biti, bytes, words (max 2^{32}-1 biti)
- **Date SIMD impachetate** (MMX,SSE,…)
- **BCD, BCD impachetat**

Variabile simple

`<NUME> TIP <valoare>|?`

NUME - eticheta ce simbolizeaza adresa variabilei

valoare - valoare numerica cu care este initializata locatia

? - locatia ramane neinitializata (se pastreaza ce era ininte in memorie la acea adresa)

Variabila poate pastra o valoare numerica intreaga, in virgula mobile sau o adresa

TIP - unul din tipurile fundamentale

- Byte: `DB|Byte|SByte`
- Word: `DW|Word|SWord`
- Doubleword: `DD|DWord|SDWord`
- Quadword:`DQ|Qword`
- FP: `Real4, Real8, Real10`

Interval de reprezentare

Byte: intreg fara semn [0,255], intreg cu semn [-128, 127], 1 cod ASCII

Word: intreg fara semn $[0,2^{16}]$, intreg cu semn $[-2^{15}, 2^{15}-1]$, 2 coduri ASCII

DWord: intreg fara semn $[0,2^{32}]$, intreg cu semn $[-2^{31}, 2^{31}-1]$, 4 coduri ASCII

QWord: intreg fara semn $[0,2^{64}]$, intreg cu semn $[-2^{64}, 2^{64}-1]$, 8 coduri ASCII

Exemple de declaratii

```
m db ?
i db 6
l byte 260
j byte -7
al word 23
l byte 255
dcuv dword 12345678h
```

```
tt byte -130
k sbyte -23
bits byte 10101111b
car byte 'A'
cuv dw 1234h
var word 0FFFFh
Fvalue dd 1.2
V2 Real8 13.12
```

Exemple de declaratii

```
m db ?
i db 6
l byte 260
j byte -7
al word 23
l byte 255
dcuv dword 12345678h
```

```
tt byte -130
k sbyte -23
bits byte 10101111b
car byte 'A'
cuv dw 1234h
var word 0FFFFh
Fvalue dd 1.2
V2 Real8 13.12
```

Definirea propriului tip

nume **TYPEDEF** *tip*

Exemplu:

```
integer TYPEDEF SWord
char TYPEDEF Byte
val integer -5
chr char 'A'
```

- Directiva TYPEDEF este o macrodefinitie
- Nume diferite pentru tipuri predefinite

Inregistrari

nume **STRUCT**
lista de variabile
nume **ENDS**

numeInregistrare nume {[val1 [… valn]]}

```
student STRUCT
        FirstName char 12 dup (?)
        LastName char 12 dup (?)
        BirthYear integer ?
        Mark byte ?
student ENDS
.............
popescu student {Ion, Popescu, 1989,10}
...........
```

Tablouri

- Structura de date care contine elemente de acelasi tip

Exemplu

```
V1   DB 1,2,3
CAR  DB "123"
V3   DW 1,12h,0123h
A    DD 5 DUP(21)
M    DB 1,2,3
     DB 4,5,6
     DB 8,9,10
M2   DD 10 DUP(6 DUP(3))
```

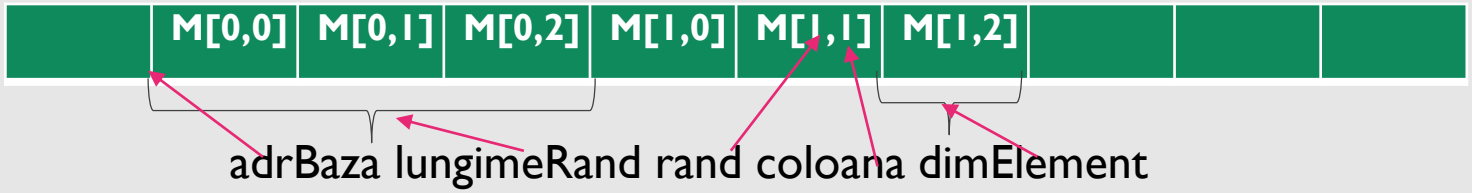
- adrBaza Index dimElement

- Tablou 1D

adrElement = adrBaza+Index*dimElement

- Tablou 2D

adrElement = adrBaza+rand*lungimeRand+coloana*dimElement



Declararea constantelor

- Constanta = nume symbolic dat unei valori des utilizate
- Directive EQU, =, TEXTEQU
- Sintaxa: *nume* EQU *expresie*
- La compilare numele constantei se inlocuieste cu expresia; este o constructie de tip MACRO
- Sintaxa se verifica la inlocuire
- Expresia e una aritmetica sau logica, evaluata la asamblare

```
trei equ 3
true equ 0
false = 0ffh
text byte 'acesta este un text'
lung_text equ $-text
```

$ - reprezinta valoarea contorului curent de adrese

Declararea structurilor de program

- Directivele .CODE, .DATA, .STACK
  - Declara inceputul unui segment de cod, date, stiva
  - Pentru ca un segment sa poata fi accesibil, adresa trebuie incarcata intr-un registru segment
  - un program contine in mod obisnuit un segment de date, un segment de cod si un segment de stiva
- Directiva PUBLIC face numele specificat disponibil tuturor modulelor program

- Directiva END
  - directiva de incheiere a programului
  - tot ce urmeaza dupa directiva se ignora la compilare

  Sintaxa: END [*eticheta*]
  - *eticheta* este punctul de start al programului

- Directiva EXTERN declara variabile, etichete sau simboluri externe

Sintaxa instructiunilor

- **<linie_program>**:= [<eticheta>:] [<prefix>] [<instructiune>|<directiva>] [;<comentariu>]
- **<eticheta>** - sir de litere si cifre, care incepe cu o litera
- **<prefix>** - cuvant cheie care modifica regimul de executie al instructiunii care urmeaza (ex: REP – repeta instructiunea urmatoare de un nr. de ori)
- **<instructiune>**:= <mnemonica> [<operand1> [,<operand2>[,<operand3>]]]
- **<mnemonica>** - grup de litere care simbolizeaza o instructiune
- **<operand1>**:= <registru>|<variabila>
- **<operand2>**:= <val_imediata>|<operand1>
- **<operand3>**:= <val_imediata>

Sintaxa instructiunilor (continuare)

- **<registru>** := EAX|EBX|...AX|BX| .. |AH|BH|.. |AL|BL| ...|CS|DS, ...|GS => nume de registru
- **<variabila>**:= <nume_var>|<adresa_var>
- **<adresa_var>**:=[<nume_var>]'['[reg_index][+<reg_baza>] [+<deplasament>]']'
- **<val_imediata>**:=<numar>|<expresie>
  - un numar sau o expresie aritmetico-logica care se poate evalua in momentul compilarii; se poate exprima in zecimal, hexazecimal (indicativul H) sau binar (indicativul B)
- **<deplasament>** := <val_pe_16biti>|<val_pe_32biti>
  - valoare exprimabila pe 16 sau 32 biti
- **<comentariu>** - text (cu caracter explicativ) ignorat de compilator

Sintaxa instructiunilor x86

- Exemple:
  - instructiuni fara operand

    NOP

    MOVSB
  - instructiuni cu un operand

    PUSH AX

    MUL CL
  - instructiuni cu doi operanzi

    MOV AX, BX
  - linie cu eticheta instructiune si comentariu

    START: MOV AX,BX ;muta cont. AX in BX
  - linie de comentariu

    ; aceasta este o linie de comentariu
  - linie cu eticheta

    ETICHETA:

Reguli sintactice

- o linie de program poate contine maxim o instructiune (mnemonica + operanzi) sau o directiva
- o linie poate contine:
  - nici o entitate de instructiune (camp) – linie goala
  - numai eticheta
  - numai comentariu
  - combinatii de eticheta, instructiune, directiva si comentariu
- un comentariu incepe cu ';' si se incheie la sfarsitul liniei
- o instructiune x86 poate contine maxim 3 campuri de operanzi

Reguli sintactice - Exemple

NOP
- instructiune fara operanzi

MOVSB
- instructiune cu operanzi impliciti

MUL CL
- instructiune cu primul operand implicit (AX=AL*CL)

MOV AX, BX
- AX – destinatia , BX sursa transferului

INC SI
- SI – termenul incrementat si destinatia rezultatului

ADD CX,DX
- CX – primul termen al sumei si destinatia rezultatului, DX – al doilea termen

ADD AX,BX,CX
- instructiune incorecta, prea multi operanzi

Reguli sintactice

- pt. scrierea programului pot fi folosite **litere mici si mari**, insa asamblorul nu face distinctie intre literele mici si mari
- separarea campurilor dintr-o instructiune se poate face cu un numar arbitrar de caractere **<spatiu> si <tab>**
- pt. lizibilitate se recomanda aranjarea campurilor pe **coloane distincte**, separate prin <tab>:
  - eticheta: mnemonica operanzi ;comentariu
- se recomanda utilizarea de **nume simbolice** in locul unor valori numerice

  ex: adrese de variabila => nume_variabila,
  - adrese de instructiune => eticheta,
  - valori de constante numerice=>nume_constanta

Reguli sintactice- simboluri

- secventa de litere, cifre si unele caractere speciale (ex: _, $, @), ?), care nu incepe cu o cifra
- lungimea simbolului este arbitrara, dar se considera primele 31 de caractere
- exista simboluri rezervate, predefinite in limbaj (cuvinte cheie pt. instructiuni, directive, macrodefinitii)
- exemple:
  - `L1  Bletch  RightHere  Right_Here  Item1 __Special`
  - `$1234  @Home  $_1  Dollar$  WhereAmI?  @1234`
  - `erori:`
    - `1TooMany – incepe cu o cifra`
    - `Hello.There – contine punct`
    - `$ - $ sau ? nu poate sa apara singur`
    - `LABEL – cuvant rezervat.`

Reguli sintactice - constante

- intregi: 12, 21d, 123h, 0fffh, 1011b
- reale (flotant): 1.5, 1.0e10, 23.1e-12
- sir de caractere: "text", 'Text', 'TEXT"TEXT'
- constante simbolice: - nume simbolic dat pentru o secventa de caractere (text); ex:

```
unu       equ    1
numar     =      26
var       textequ    "5[bx]"    ; 5[bx] – constanta textuala
```

Reguli sintactice - operanzi

- **operanzii** unei instructiuni trebuie sa fie de **aceeasi lungime**: octet, cuvant, dublu-cuvant (exceptii: operatii de inmultire si impartire)
- o instructiune poate contine **cel mult un operand** de tip **locatie de memorie**
  - formatul instructiunilor x86 permite exprimarea adresei unei singure locatii de memorie
  - pentru o operatie aritmetica sau logica intre doua variabile (locatii de memorie) unul dintre operanzi trebuie sa se transfere temporar intr-un registru intern
  - aceasta restrictie favorizeaza operatiile pe registre – pt. cresterea eficientei de executie
- **instructiunile sunt echivalente** ca nivel de structurare si sunt independente intre ele
  - nu exista forme de programare structurata
  - structurarea programului se poate face la nivel logic (formal) prin directive

Semnificatia entitatilor unei linii de program

- **Eticheta**
  - nume simbolic dat unei adrese de memorie unde incepe instructiunea care urmeaza dupa eticheta
  - util pentru instructiuni de salt si apel de rutine
- **Mnemonica (numele) instructiuni**
  - nume simbolic dat unui cod de instructiune (2, 3, 4 sau 5 litere)
  - semnifica un anumit tip de operatie elementara direct executabila de UCP
  - aceeasi mnemonica poate simboliza mai multe coduri cu semnificatie apropiata (ex: MOV, ADD, ...)
  - acelasi cod de instructiune se poate exprima prin mnemonici diferite (ex: JZ si JE)
  - fiecarei instructiuni in L.A. ii corespunde strict o instructiune in cod masina (relatie biunivoca)

Semnificatia entitatilor unei linii de program

- **Operand**
  - camp care exprima un termen al operatiei elementare exprimata prin mnemonica
  - indica locul si modul de regasire al operandului (**modul de adresare** folosit)
- Tipuri de operanzi:
  - registre interne ale UCP:
  - date imediate (constante numerice)
  - locatii de memorie (variabile)
  - porturi de intrare sau de iesire (registre de I/E)

Operand

- **Registre interne:**
  - registre generale:
    - (8 biti) AH,AL,BH,BL,CH,CL,DH,DL
    - (16 biti) AX, BX,CX,DX, SI,,DI,SP, BP
    - (32 biti) EAX, EBX,ECX,EDX, ESI,EDI,ESP, EBP
  - registre speciale: CS,DS, SS, ES, FS,GS, GDTR, LDTR , CR0,..CR4, eFlags
- **Date imediate (constante):**
  - numar sau expresie aritmetico-logica evaluabila la un numar => expresia trebuie sa contina numai constante
  - valoarea este continuta in codul instructiunii
  - lungimea constantei – in acord cu lungimea celui de al doilea operand (octet, cuvant sau dublu-cuvant)
  - ex: 0, -5, 1234**h, 0**ABCD**h**, 11001010**b, 1b, 8* 4 - 3**

Operand

- **Locatii de memorie (variabile):**
  - expresie care exprima adresa unei locatii de memorie de o anumita lungime
  - lungimea variabilei:
    - in acord cu al doilea operand (daca exista)
    - se deduce din declaratia numelui de variabila
    - se indica in mod explicit ('byte', 'word', 'dword')
  - adresa variabilei:
    - selector:
      - specificat in mod implicit – continutul registrului DS
      - exprimat in mod explicit: <reg_segment>:<variabila>

      ex: CS:Var1, ES:[100h]
    - adresa de offset - adresa relativa in cadrul segmentului

Operand

- **Porturi de Intrare/Iesire**
  - registre continute in interfetele de intrare/iesire
  - spatiul de adresare maxim: 64Ko (adr. maxima 0FFFFH)
  - la PC-uri spatiul este limitat la 1ko (adr. maxima 3FFH)
  - pe aceeasi adresa pot fi 2 registre:
    - un reg. de intrare si unul de iesire
  - porturile apar doar in instructiunile IN si OUT
  - specificare:
    - direct, prin adresa fizica (daca adresa este exprimabila pe un octet) sau nume simbolic

    ex: IN AL, 12h
    OUT 33h, AL
    - indirect, prin adresa continuta in registrul DX

    ex: IN AL,DX
    OUT DX,AL

```
;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
co EQU 25
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
```

Exercitii

- Aratati cum apar datele din program in memorie.
- Calculati adresa de offset a lui **V3**.
- Calculati adresa de offset a lui **co**.