CURS 4 – 16 mai. 2022

10 Dec: în ce mod se adresează op. direct/indir./etc.

- Orice fel de conținut al registrilor sunt modificabile la momentul asamblării.
- accesarea a 2 pointeri → val. scalară.

Utilizarea operatorilor

Operator = pt. combinarea, compunerea, modificarea și analiza operanzilor

→ operatorii efectuează calcule cu valori constante SCALARE determinabile la momentul asamblării.

ex: mov eax, ebx + ecx. → SyntaxError.
operator

excepție [ebx + ecx] → formula de calcul a offsetului în operand.

Ce este aici în curs nu trebuie știut.

Operatori logici pe biți

u << 4 → deplasare biți către stânga
u >> 4 → deplasare biți către dreapta.
u & 4 → și bit cu bit
u | 4 , u ^ 4 → sau bit cu bit.

Examples

X AND 0 = 0

X AND 1 = X.

X AND X = X.

X AND ~X = 0

Operația AND este utilă pt. forțarea valorii anumitor biți la val. 0.

X or 0 = X

X or 1 = 1

X or X = X

X or ~X = 1

Operația OR este utilă pt. forțarea valorii anumitor biți la val 1.

X xor 0 = X

X xor 1 = ~X

X xor X = 0

X xor ~X = 1.

Operația XOR este utilă pt. complementarea anumitor biți.

! → negare logică

!x = 0 dacă x ≠ 0 altfel 1.

~ → complementare

Dată să pună
12 module de inițializare
a lui constructor
registru cu 0
(întrebare examen)

example 2

a db ...

b dw ...

...

mov eax, ![a] ⇒ Syntax Error (nu e const la mom. asm).

mov eax, [!a] ⇒ Syntax Error (not a scalar value)

mov eax, !a ⇒ ——— " ———

mov eax, !(a+7) ⇒ ——— " ———

mov eax, !(b-a) ✓ (pointer aritmetich + scalar values)

mov eax, ![a+7] ⇒ Syntax Error (nu e const. la mom. asm).

mov eax, !7. ; eax = 0

example 3

AA equ 2.

mov ah, !AA ✓.

example 4

mov ah, 17 ^ (~17) = ffh.

mov ax, value ^ (~value) 0ffffh.

example 5

```
v db 10, -101, 'x'
a dw 0ffh, -1, 42
b dd 12345678h, -1.
```

Explicați efectele reprezentării little endian ~~la registru~~ la nivelul arhitecturii x86 ~~op~~ / mv se aplică la registru

push v ; stack ← offset (v)

push [v] ⇒ ~~Syntax~~ Error, push word [v] / dword [v]

mov eax, v ⇒ eax ← offset dav pe 32 byți

mov eax, [v]

push eax ← ia conținutul lui eax și îl pune în stivă

push [eax] ⇒ Syntax Error.

push 15 ⇒ DWORD 15 (declară 4 octeți)

push byte [v] ⇒ Syntax Error (push nu are operand pe 1 byte)

pop [v] ⇒ Syntax Error ; pop word [v] / dword [v]

pop v ; 2:23 v is not a L-value ! ⇒ Syntax Err.

pop dword v ⇒ Syntax Error.

pop [eax] ⇒ Syntax Error (size off).

pop 15 ⇒ Syntax Error. (not a L-value)

pop [v] ⇒ Syntax Error (lipsa dim.)

Syntax Error

```
mov [v], 0
    byte, word, dword
mov v, 0
```

```
mov [v], byte 0.v
div [v] ⇒ Syntax Error
  ↑
  word, byte, dword.
```

Se va testa pe 10 dec ↓

Exemple 6

a db ...
b dw ...

mov a,b → S.E. (syntax error) not L-value

mov [a], b → S.E. ~~mov byte [a], b. X~~

mov a, [b] → S.E. not L-value — mov | word / dword | [a], b

mov [a], [b] → nu putem avea 2 operanzi din mem.

mul v → S.E. → nu poate fi const.

mul [v] → S.E. → sintaxa lui mul nu e respectată

| byte / word / dword |

mul [eax] → S.E. → nu se specifică nivelul

mul 15 → S.E. → nu respectă sintaxa mul

pop byte [v] → S.E.

pop dword [v] → instruction not supported in 32 bit. S.E.

Clasificarea erorilor

- eroare de sintaxă → diagnosticată de compilator / asamblor
  - eroare de asamblare
- run-time error (eroare de execuție) → → programul „crapă" – crash.
- eroare logică → programul rulează, dar nu primește rezultatele așteptate

Pe 10 dec: formularea erorilor