Arhitectura sistemelor de calcul

Operații și operatori pe biți

Operatori vs instrucțiuni

mov AH, 01110111 << 3 vs mov AH, 01110111

SHL AH, 3

- operatorii pot fi folosiți doar cu valori constante la momentul asamblării
- instrucțiunile pot fi folosite pe regiștri care sunt cunoscuți doar la momentul execuției

Operatori bit pe bit

& - operatorul x and 0 = 0 x and x = x

AND - instrucțiune x and 1 = x x and ~x = 0

Operație and utilă pt forțarea valorii anumiților biți la 0

| - operatorul x or 0 = x x or x = x

OR - instrucțiune x or 1 = 1 x or ~x = 1

Operația or utilă pt forțarea anumiților biți la 1

^ - operatorul XOR - instrucțiunea

x xor 0 = x x xor x = 0

x xor 1 = ~x x xor ~x = 1

Operația xor utilă pt complementarea anumitor biți

Exemple:

mov eax, ![a] syntax error deoarece valoarea variabilei nu este cunoscută la asamblare

mov eax, [!a] syntax error deoarece singurelor operatori permiși pe adresă sunt + și -

mov eax, !a syntax error ——— " ———

mov eax, !(a+7) syntax error a+7 este adresă

mov eax, !(b-a) ✓ b-a nu dă scalar, adică constantă de tip imediat, nu de tip adresă

mov eax, ![a+7] syntax error deoarece valoarea de la adresa a+7 nu este știută la momentul asamblării.

mov eax, !7 ✓ eax = 0

mov eax, !0 ✓ eax = 1

mov eax, ~7 ✓ eax = ~ 0000 0000 0000 0000 0000 0000 0000 0111 =
= FF FF FF FF FF F8

mov eax, !ebx syntax error deoarece ebx nu este o valoare constantă la momentul asamblării

mov eax, ![ebx] syntax error ——— " ———

aa egu 2

mov ah, !aa ✓ ah = 0

mov ah, 17^(~17) ✓ ah = FF

Operatori de tip și tipuri de date asociate operanzilor

v d?...

a d?...

b d?...

push v ✓ stack se va pune offset v pe stivă

push [v] syntax error nu s-a specificat size-ul care să se pună pe stivă (16 sau 32)

push dword [v] ✓

push word [v] ✓ stiva e organizată pe 32 de biți dar pentru backwards compatibility, adică programarea pe 16 biți trebuie să

putem folosi stiva și cu val pe 16 biți

push byte [v] syntax error

push [eax] syntax error nu s-a specificat size-ul care să se pună pe stivă

push 15 ✓ îl pune pe dword 15

pop [v] syntax error nu s-a specificat sizeul care se ia din stivă (16 sau 32)

pop word [v] ✓

pop dword [v] ✓

pop v syntax error nu se poate modifica o adresă

pop dword b syntax error ——"——

pop [eax] syntax error nu s-a specificat sizeul care se ia din stivă (16 sau 32)

pop 15 syntax error nu se poate modifica o constantă

pop [15] syntax error nu s-a specificat sizeul care se ia din stivă (16 sau 32)

pop dword [15] ✓ cel mai probabil se va obține memory violation error

Clasificarea erorilor

- syntax error : sintaxa este greșită
- run-time error : treci peste faza de asamblare dar programul dă crash (memory violation)
- logic error : programul funcționează însă greșit după logic
- linking error : ex: avem mai multe declarări ale aceleiași variabile când lucrăm cu mai multe module

mov [v], 0 syntax error size not specified

mov byte/word/dword [v], 0 ✓

mov [v], byte/word/dword 0 ✓

div [v] syntax error size not specified

imul [v+2] syntax error size not specified

mov a, b syntax error nu putem atribui unei adrese valoarea altei adrese

mov [a], b syntax error size not specified

mov word/dword [a], b ✓ 16 bit compatibility

mov byte [a], b syntax error nu poți stoca într-un octet o adresă

mov a, [b] syntax error nu putem atribui unei adrese o valoare

mov [a], [b] syntax error nu pot să existe în aceași operație doi operanzi din memorie deoarece procesorul are nu poate calcula simultan două adrese de memorie

mul v syntax error sintaxa este mul reg/mem

mul [v] syntax error operation size not specified

mul eax ✓

mul [eax] syntax error operation size not specified

mul 15 syntax error sintaxa este mul reg/mem

pop byte [v] syntax error invalid combination doar 16 / 32 biți

pop qword [v] syntax error invalid combination doar 16 / 32 biți

Steps followed by a program:

1. Syntadic checking
2. obj files generated
3. linking phase (if more modules it links variables / procedures) and if there is no linking error, exe file is generated
4. when opening exe file firstly there is loader and doing relocation (FAR compilation) from RAM
5. loader gives control to processor by specifying the programs entry point (start)