ASC - 30 ian. 2024

→ 01 x dw -129, 10+100h+1000b

-129 = C2(129) = C2(0000 0000 1000 0001) = 1111 1111 0111 1111 =

= FF 7F h

100h = 256

1000b = 8

10+100h+1000b = 10 + 256 + 8 = 266 + 8 = 274 = 112h

in memorie: 7F | FF | 12 | 01 ✓

→ 41 y dw 1001h >> 1001b, 128h & 128

1001h = 0001 0000 0000 0001b

1001b = 9

1001h >> 1001b = 0000 0000 0000 1000 b = 08h

128h = 0001 0010 1000b

128 = 80h = 1000 0000b

128h & 128 = 0001 0010 1000 & 0000 1000 0000 = 0h

08 | 00 | 00 | 00 ✓

→ 81 z dw z, $$-z

-8 = C2(8) = C2(0000 0000 0000 1000) = 1111 1111 1111 1000 =

= FF F8h

08 | 00 | F8 | FF ✓

?

→ 121 w dd x+y-z, w-y+x

x+y-z = (y-z)+x = 4-8+0 = -4 = C2(4) = C2(100) = 1100b = Fch

w-y+x = 12-4+x = x+8

FC | FF | FF | FF | 08 | 00 | 00 | 00

BRUNNEN

→ 20| h dw 101b, 101-h, 11h-11b, h-11

101-h va da eroare, deci îl ignorăm

0101b = 5h

11h-11b = 17-3 = 14 = 0Eh

h-11 = 20-11 = 9 = 09h

05|00|0E|00|09|00

→ 26|a db $$-$, h-11b

h-11b se va ignora, deoarece un offset nu poate fi pe 8 biți

$$-$ = 0-26 = -26 = C2(26) = C2(0 001 1010) = 1110 0110 = E6h

|E6|

→ 27| b dd a+b-0Ah+2, h-b+0Ah-0Bh

a+b-0Ah+2 se va ignora, deoarece adunarea de pointeri nu ~~este~~ se poate efectua.

h-b+0Ah-0Bh = 20-27+10-11 = -7+10-11 = 3-11 = -8 = C2(8) =

(20 - 27 scalar)

= C2(0000 1000) = 1111 1000 = F8h

în memorie: F8|FF|FF|FF

→ 31| c db 3-b, z-w

3-b se va ignora, întrucât rezultatul este pointer și nu poate fi reprezentat pe 8 biți

z-w = 8-12 = -4 = C2(4) = C2(0000 0100) = 1111 1100 = FCh

în memorie: |FC|

→ 32| d dw -513, 128^(~128) → în memorie: FF|FD|FF|FF

-513 = C2(513) = C2(0000 0010 0000 0001) = 1111 1101 1111 1111 =

= FDFFh

128^(~128) = 0000 0000 1000 0000 ^ 1111 1111 0111 1111 =

= 1111 1111 1111 1111 = FFFFh

→ 36 | e dd ABCDEFh, 'abcdefh'

ABCDEFh se va ignora, deoarece nu e o valoare în baza 16, ci un simbol nedefinit.

în memorie: 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'h' | 00

→ 44 | f dw w-1, [w-1]

[w-1] se va ignora, deoarece nu poate fi determinat la momentul asamblării.

w-1 = 12-1 = 11 ca pointer

în memorie: 0B | 00

→ 46 | g times 3 dw 'db'

Directiva "times" indică că șirul de caractere 'db' se generează de 3 ori în memorie:

'd' | 'b' | 'd' | 'b' | 'd' | 'b'

→ 52 | k dw 1+2b+3h+a, c+0ch

1+2b+3h+a se ignoră, deoarece este eroare de sintaxă (2b)

c+0ch = 31+12 = 43 ca pointer
(2Bh)

în mem: 2B | 00

→ 54 | m dd a+0Ah, a+ah

Al doilea operand se va ignora, deoarece 'ah' este considerată un simbol care nu este definit.

a+0Ah = 26+10 = 36 ca și pointer
(24h)

24 | 00 | 00 | 00

BRUNNEN

III. a) i) • mov eax, -2 ; EAX = FF FF FF FE
AH AL
AX

În interpretarea cu semn:

EAX = -2 = FF FF FF FE h = 1111 1111 1111 1111 1111 1111 1111 1110 b

În interpretarea fără semn:

EAX = $2^{32} - 2$

• mov ebx, -1 ; EBX = FF FF FF FF
BX

În interpretarea cu semn:

EBX = -1 = FFFF FFFF h = 1111 1111 1111 1111 1111 1111 1111 1111 b

În interpretarea fără semn:

EBX = $2^{32} - 1$

• div bl

Se va efectua împărțirea fără semn a lui AX la BL, iar rezultatele sunt reținute în AL (câtul), respectiv AH (restul).

AX = [cu semn: FFFEh = -2 = 1111 1111 1111 1110 b
fără semn: FFFEh = 65536 - 2 = 65534

BL = [cu semn: FF = -1 = 1111 1111
fără semn: FF = 256 - 1 = 255

AX : BL = 65534 : 255 = 256 rest 254

510
1453
1275
1784
1530
254

Programul se oprește, deoarece s-a produs depășirea la împărțire, adică rezultatul nu încape în spațiul rezervat acestuia (AL ar trebui să primească valoarea 256, însă 256 nu aparține domeniului de reprezentare admis pe un octet, $[0, 255]$).

Flagurile nu sunt definite la împărțire și nu le sunt modificate valorile inițiale.

ii) • mov eax, 65409

interpretarea
- cu semn: EAX = 65409 = FF81h = 1111 1111 1000 0001b
- fără semn: EAX = 65409 = FF81h = 1111 1111 1000 0001b

• idiv ah

Se efectuează împărțirea cu semn dintre AX și AH, iar câtul și restul vor fi stocate în AL respectiv AH

AX = FF81h = 65409

În interpretarea fără semn, 65409 încape în AX, întrucât $65409 \in [0, 65535]$, însă la interpretarea cu semn, valoarea din AX este $65409 - 65536 = -127 = C2(127) = C2(0111\ 1111) = 1000\ 0001 = 81h$

AH = FFh = 255

În interpretarea fără semn, 255 încape în AH, întrucât $255 \in [0, 255]$, însă la interpretarea cu semn, valoarea din AH este $255 - 256 = -1 = FFh$

AX : AH = -127 : (-1) = 127 rest 0

Deci AL = 127 = 7Fh = 0111 1111b și AH = 0 = 0h = 0000 0000b în ambele interpretări.

• add al, al

AL ← AL + AL = 127 + 127 = 254

În interpretarea fără semn, 254 încape în AL, întrucât 254 ∈ [0, 255], deci CF = 0, însă în interpretarea cu semn se produce depășire, deoarece 254 ∉ [-128, 127], deci OF = 1. Rezultatul din AL în interpretarea cu semn este 254 - 256 = -2 = FEh = 1111 1110b.

iii) • mov eax, 255h & 255 ; EAX = 055h (AL)

255 = 1111 1111

255h = 0010 0101 0101

255h & 255 = 0010 0101 0101 & 0000 1111 1111 =

= 0000 0101 0101 = $0 \cdot 55h$ (2 1 0) =

= $16^0 \cdot 5 + 16^1 \cdot 5 + 16^2 \cdot 0 =$

= 5 + 80 = 85

EAX are valoarea 85 atât în interpretarea cu semn, cât și fără semn, deci OF = 0, CF = 0

• mov ebx, 256 ^ 256h ; EBX = 0356h (BH BL)

256 = 0001 0000 0000b

256h = 0010 0101 0110b

256 ^ 256h = 0001 0000 0000 ^ 0010 0101 0110 = 0011 0101 0110 =

= $356h$ (2 1 0) = $6 \cdot 1 + 5 \cdot 16 + 16^2 \cdot 3 = 6 + 80 + 768 = 854$

EBX are valoarea 854 atât în interpretarea cu semn, cât și fără semn, deci OF = 0 și CF = 0

• mul bh fără semn

Se va efectua înmulțirea dintre AL și BH, iar rezultatul va fi în AX.

AL = 55h = 85 (în ambele interpretări)

BH = 03h = 3 (în ambele interpretări)

AX = AL * BH = 85 * 3 = 255 = FFh = 1111 1111b (în ambele interpretări)

Întrucât rezultatul în interpretarea fără semn ar fi putut fi reprezentat pe un octet, OF = CF = 1.

iv) • mov ax, 12812 ; AX = 82h

12812 = 1000 0000 1000 0010 = 1000 0010 = $82^{10}h$ = 2·1 + 8·16 = 128+2 = 130

AX are valoarea 130 în ambele interpretări

• mov bh, 4Ah >>2 ; BH = 12h

4Ah >>2 = 0100 1010 >>2 = 0001 0010b = $12^{10}h$ = 2·1 + 16·1 = 18

BH are valoarea 18 în ambele interpretări.

• sub ah, bh

Se va efectua scăderea dintre AH și BH. Rezultatul va fi în AH.

AH = AH - BH = 0 - 18 = -18 (în interpretarea cu semn)

= C2(18) = 1110 1110 = $EE^{10}h$ = 14·1 + 14·16 =

= 14 + 224 = 238 (în interpr. fără semn)

0000 0000 -
0001 0010
1110 1110

CF = 1
OF = 0

v) • xor eax, eax

EAX = 0

• lea ebx, [esi]

Încarcă în EBX adresa lui ESI

vi) instrucțiuni cu un operand explicit și unul implicit de dimensiuni diferite: div, idiv

vii) instrucțiuni cu un operand explicit și unul implicit de dimensiuni identice: mul, imul

b)
```
lea esi, [esp+4]    ; în ESI se pune adresa de la ESP+4
lea edi, [esi-8]    ; în EDI se pune adresa lui ESI-8, adică ESP-4
push esp            ; vârful stivei se pune pe stivă
ss lodsb
ss stosb
```

→ mov eax, ebx (?)