ASC - 23 ian 2019

1 → 01 a1 db '256'

Se consideră un șir de caractere:

în memorie: '2' | '5' | '6'

→ 3 | a2 dw 256, 256h

256 = 100h = 0000 0001 0000 0000b

în memorie: 00 | 01 | 56 | 02

→ a3 dw $+a2

Eroare de sintaxă, deoarece aritmetica de pointeri nu se poate efectua

Se ignoră linia

→ 7 | a4 equ -256/4

-256/4 = -64

a4 este o constantă, deci nu se stochează în memorie, iar offsetul nu crește.

→ 7 | a5 db 256>>1, 256<<1

256>>1 = 0000 0001 0000 0000>>1 = 0000 0000 1000 0000 = 128 =

= 0080h (80h pe byte)

256<<1 = 0000 0010 0000 0000 = 0200h (00h pe byte)

în memorie: 80 | 00 |

→ 9 | a6 dw a5-a2, !(a5-a2)

a5 - a2 = 7 - 3 = 4 = 04h (scalar)

!(a5-a2) = !4 = 0

în memorie: 04 | 00 | 00 | 00

→ 13/a7 dw [a2], ~a2

[a2] se va ignora, deoarece nu poate fi determinat la momentul asamblării, eroare de sintaxă

~a2 se va ignora, deoarece operatorul ~ nu se poate aplica pe un pointer.

→ 15/a8 dd 256h ^ 256, 256256h

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

= 0000 0000 0000 0000 0000 0011 0101 0110 b (dword)

= 00 00 03 56h (pe dword)

în memorie: 56 | 03 | 00 | 00 | 56 | 62 | 25 | 00

→ 21/a9 dd $-a9

$-a9 = 21-21 = 0 (scalar)

în memorie: 00 | 00 | 00 | 00

→ 25/a10 db 256, -255

256 = 100h (se trunchiază și se ia doar 00h)

-255 = C2(255) = C2(1111 1111) = 0000 0001 = 01h

în memorie: 00 | 01

→ 27/a11 dw 256h-256

$256h = 6 \cdot 1 + 16 \cdot 5 + 16^2 \cdot 2 = 6 + 80 + 256 \cdot 2 = 86 + 512 = 598$

256h - 256 = 598 - 256 = 342 = 156h

în memorie: 56 | 01

→ 29/a12 dw 256 - 256h

256 - 256h = 256 - 598 = -342 = C2(342) = C2(0001 0101 0110) =

= 1110 1010 1010 = EAAh

în memorie: AA | FE

→31/ a13 dw-256

-256 = C2(256) = C2(0000 0001 0000 0000) =

= 1111 1111 0000 0000 = FF00h

în memorie: 00|FF

→38| a14 dw -256h

-256h = C2(256h) = C2(0000 0010 0101 0110) =

= 1111 1101 1010 1010 = FDAAh

în memorie AA|FD

→35| a.15 db 2, 5, 6, 25, 6, 2, 56

2=02h 5=05h 6=06h 25=19h 56=38h

în memorie: 02|05|06|19|06|02|38

III. a) CF, OF, ZF, SF, AH=?

a1) • mov ah, 129 ; AH = 81h = 1000 0001b

interpretare
- cu semn: AH = 129 - 256 = -127 ∈ [-128, 127]
- fără semn: AH = 129 ∈ [0, 256]

• mov bh, 9Fh ; BH = 9Fh = 1001 1111b

9Fh = 15·16^0 + 9·16 = 15 + 144 = 159

interpretare
- cu semn: 159 ∉ [-128, 127], deci BH = 159 - 256 = -97
- fără semn: 159 ∈ [0, 255], deci BH = 159

• add ah, bh ;

Se va efectua adunarea dintre AH și BH, iar rezultatul va fi în AH.

AH = 1000 0001

BH = 1001 1111

1 0010 0000 = 20h = 32 (în orice interpretare)

CF = 1

Se observă că rezultatul nu a putut fi reprezentat pe 8 biți, deci CF=1. Mai mult, se observă că negativ - negativ = pozitiv (cf. bitului de semn), deci OF=1. Rezultatul este nenul, deci ZF=0. Bitul de semn al rezultatului este 0, deci SF=0.

a2) • mov ax, 128 ; AX = 0080h = 0000 0000 1000 0000b
AH AL

AX are valoarea 128 în orice interpretare

• sar al, 7 ; AL = 1111 1111 b = FF h

AL are valoarea 128, care, în interpretarea cu semn, 128 ∉ [-128, 127], deci AL = 128 - 256 = -128. Bitul de semn fiind 1, se va pune 1 pe cei 7 biți ~~de~~ shiftați la dreapta

• imul ah

Se face înmulțirea cu semn dintre AL și AH. Rezultatul este în AX.

AL * AH = -1 * 0 = 0 = 00h

CF = OF = 0, întrucât rezultatul încape pe un octet

ZF nu este definit la înmulțire și împărțire

SF = 0 (0 pozitiv)

analog a3, a4

b) b1) movsx EAX, AL
↓
move with sign extend, copiază nr. întreg din AL și îl mută pe 32 de biți, extinzând dimensiunea cu bitul de semn.

b2) cbw, cwd, cwde
↳ convertește cu semn byte-ul din AL în wordul AX

b3) nu există, deoarece BIU poate calcula o singură adresă

b4) movsb / movsw / movsw

b5) movsb / movsw / movsw / cmpsb / cmpsw / cmpsd