23.01.2019

Subject I

**Explain and exemplify how negative numbers are represented and operated at the level of the 80x86 architecture.**

**Detail the representation mechanism and after that give 2 adequate representation examples (ex: discussion and analysis on the representation of -32 or -912 etc.).**

**How can we obtain the corresponding signed value in base 10 for a given representation in base 2?**

**Which are in this sense the possible practical methods/techniques/mechanisms to be applied?**

**Present how the programmer operates with negative numbers at the level of the assembly language in the case of each of the 4 basic arithmetic operations: addition, subtraction, multiplication and division (1 adequate source code example for each basic operation – explained and justified).**

**Present and exemplify (by adequate short source code sequences) every assembly language instruction (or category of instructions if they are very similar) capable to operate with such numbers (1 example for each instruction/category – explained and justified).**

**How is the value 0 (zero) considered and why?**

Subject II

a1 db ‘256’

**a1: ‘2’ ‘5’ ‘6’**

a2 dw 256, 256h

**a2: 00 01 | 56 02**

256 = 0001 0000 0000b = 100h

a3 dw $+a2

**syntax error, sum of pointers**

a4 equ -256/4

**doesn’t take any space in the memory, it’s a constant so it’s stored separately**

a5 db 256>>1, 256<<1

**a5: 80 | 00**

256 = 0001 0000 0000

256>>1 = 1000 0000b = 80h

256<<1 = 0010 0000 0000b = 200h

a6 dw a5-a2, !(a5-a2)

**a6: 04 00 | 00 00**

a5-a2 = 7 – 3 = 4

a7 dw [a2], ~a2

**syntax error**

a8 dd 256h^256, 256256h

**a8: 56 03 00 00 | 56 62 25 00**

256h = 0010 0101 0110b

256 = 0001 0000 0000b

256h^256 = 0011 0101 0110b = 356h

a9 dd $-a9

**a9: 00 00 00 00**

a10 db 256, -255

**a10: 00 | 01**

256 = 0001 0000 0000b = 100h

255 = 1111 1111b

-255 = 0001b = 01h

a11 dw 256h-256

**a11: 56 01**

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

a12 dw 256-256h

**a12: AA FE**

256-256h = 256-598 = -342 = 1111 1110 1010 1010b = FEAAh

342 = 0001 0101 0110b

a13 dw -256

**a13: 00 FF**

256 = 0001 0000 0000b

-256 = 1111 0000 0000b = F … F00h

a14 dw -256h

**a14: AA FD**

256h = 0010 0101 0110b = 598

-598 = 1101 1010 1010b = FDAAh

a15 db 2, 5, 6, 25, 6, 2, 56

**a15: 02 | 05 | 06 | 19 | 06 | 02 | 38**

25 = 0001 1001b = 19h

56 = 0011 1000b = 38h

Subject III

a) CF (carry), AH (auxiliary), SF(sign), OF(overflow), ZF(zero)

1.

mov ah, 129 AH = 81h

mov bh, 9Fh BH = 9Fh

add ah, bh AH = 8\*16 + 1 + 9\*16 + 15 = 288 = 120h => AH = 20h

**CF = 1, AH = 1, SF = 0, OF = 1, ZF = 0**

2.

mov ax, 128 AX = 0080h AL = 80h = 1000 0000b

sar al, 7 AL = FFh AX = 00FFh (sar inlocieste cu 1)

imul ah AX = AL\*AH = 0\*(-1) = 0000h

**CF = 0, AF = 0, SF = 0, OF = 0, ZF = 0?**

3.

mov ax, 256 AX = 0100h AH = 01h

mov bx, -1 BX = FFFFh BX = FFh

add ah, bh AH = AH + BH = 1 + (-1) = 0

**CF = 1, AF = 1, SF = 0, OF = 0, ZF = 1**

4.

mov ah, 128|2 AH = 1000 0000b | 0000 0010b = 1000 0010b = 82h

mov bh, 90h>>3 BH = 1001 0000b >> 3 = 0001 0010b = 12h

sub ah, bh AH = 82h – 12h = 130 – 18 = 112 = 70h

**CF = 0, AF = 0, SF = 0, OF = 1, ZF = 0**

b) instructions

1.

2. cbw

3.

4. lodsb (both strings need to be saved in memory first?)

5. lodsw (the word (16 bits) from the address <DS:ESI> is loaded in AX (16 bits))