23 ian 2019

Present the corresponding memory layout of the following segment (it's structure and corresponding memory values, assuming that its starting offset is 0), justifying in context the reason for obtaining each of the stored values:

data segment

```
a1 db '256'
a2 dw 256, 256h
a3 dw $+a2          adding 2 pointers
a4 equ -256/4
a5 db 256 >>1, 256 << 1
a6 dw a5-a2, !(a5-a2)
a7 dw [a2], ~a2     the contents are not accessable at asm time
a8 dd 256h^256, 256256h
a9 dd $-a9
a10 db 256, -255
a11 dw 256h-256
a12 dw 256-256h
a13 dw -256
a14 dw -256h
a15 db 2,5,6,25,6,2,56
data ends
```

If there are data elements or source lines which you consider to be syntactically incorrect, motivate and explain your decision and ignore further that values or lines in providing the requested memory layout.

offset :

a_1 0	1	2	a_2 3	4	5	6	a_5 7	8	a_6 9
'2'	'5'	'6'	00	01	56	02	80	00	04
10	11	12	a_8 13	14	15	16	17	18	19
00	00	00	56	03	56	00	00	56	62
20	21	a_9 22	23	24	25	a_{10} 26	27	a_{11} 28	29
25	00	00	00	00	00	00	01	56	01
a_{12} 30	31	a_{13} 32	33	a_{14} 34	35	a_{15} 36	37	38	39
AA	FE	00	FF	AA	FD	02	05	06	19
40	41	42							
06	02	38							

* >> 1 = împărțire la 2

256 : 2 = 128 = 80h

<< 1 = înmulțire cu 2

256 * 2 = 512 = 200h

!(a5-a2) = !4 = 0

0000 0010 0101 0110 ^ = 256h

0000 0001 0000 0000

0000 0011 0101 0110 = 0356h

-255 ∈ $[-2^{15}, 2^{15}-1]$

100 00 00 00 −
FF

FF FF FF 01 = −255

10 0101 0110 −
1 0000 0000

00 0101 0110

00 01 0000 0000 −
00 02 0101 0110

FFFF 1110 1010 1010

FFFE A8

0001 0000 0000 −
10 0101 0110

1110 1010 1010

−256h

100 00 −
256

FDAA

0010 0101 0110 −
0001 0000 0000

0101 0110

III. a). Which will be the values from CF, AH, SF, OF and ZF after running each of the following sequ

a1).
```
mov ah,129
mov bh,9Fh
add ah,bh
```

a2).
```
mov ax, 128
sar al, 7
imul ah
```
leftmost bit is filled with the sign bit
last bit kept in CF

a3).
```
mov ax, 256
mov bx, -1
add ah,bh
```

a4).
```
mov ah, 128 | 2
mov bh, 90h >> 3
sub ah,bh
```

Justify your answer and explain in detail the effects and the output of each source line.

b1). Give an example of an asm instruction for which both explicit operands to be of different size. Explain.
b2). Give an example of an asm instruction for which both implicit operands to be of different size. Explain.
b3). Give an example of an asm instruction having both explicit operands from memory. Explain.
b4). Give an example of an asm instruction having both implicit operands from memory. Explain.
b5). Give an example of an asm instruction for which both implicit operands to be of the same size. Explain

a1)

AH = 81h
BH = 9Fh
AH ← AH + BH

```
  1000 0001 +
  1001 1111
-------------
1 0010 0000
```

-124 + -97

```
  100h -
   9F
-------
   61h
```

CF = 1 ~~OF = 0~~ ZF = 0 SF = 0 AF = 1

OF = 1 !!

a2)

AX = 0080h

AL = 1000 0000h

sar AL, 4 ⇒ AL = FFh

imul AH ⇒ AX = AH * AL = 0 * (-1) = 0

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

a3)

AX = 0100h
BX = FFFFh
AH ← AH + BH = 100h

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

```
  0000 0001 +
  1111 1111
-------------
1 0000 0000
```

a4)

AH = 128 | 2
= 82h

```
  1000 0000 |
  0000 0010
-------------
  1000 0010
```

BH = 12h

```
1001 0000 >> 3
```

AH ← AH - BH = 70h

```
  82h -
   12
-------
   70h
```

```
  0001 0010
  1000 0010 -
  0001 0010
-------------
  0111 0000
```

SF = 0
CF = 0
OF = 1

AF = 0
ZF = 0

b1) movsx d, s moves from s the contents into d with sign extension

b2) CBW converts from AL (byte) to AX (word)

b3) Mov d, s (eax, [esi])

b4) MOVSB (mov [edi], byte [esi])

b5) XCHG byte [var], AL (exchanges the contents of 2 op)

27 ian 2023

8. Present the corresponding memory layout of the following segment (its structure and corresponding memory values, assuming that its starting offset is 0), justifying in context the reason for obtaining each of the stored values:

segment data
x dw -128, 128h
y dw 256 >> 2, 256h-256
z db z, $-z
p dd ($-y) * (x-$), y-x

a dw $-$$, $$-a
h dw 11b, 11-h, 11h, 11-h
b dw x+y-z+18, x-y+z-18
c db 2-b, b-2
d dw 257, -255
e dd 12345h, '12345'

f dw x+1, [x+1]
g times 4 db 'dw'
h db 11b, 11h, -11, -11b
k dw 1+2b+3h+a, a+0ah
m dd 0ah+0bh, a+b
times 2 dd 1234h, 5678h

If there are data elements or source lines which you consider to be syntactically incorrect, motivate and explain your decision and ignore further that values or lines in providing the requested memory layout.

offset:	x 0	1	2	3	y 4	5	6	7	z 8	p 9
	80	FF	28	01	h0	00	56	01	00	05
	10	11	12	13	14	15	16	a 17	18	19
	00	00	00	04	00	00	00	11	00	EF
	20	h 21	22	23	24	b 25	26	27	28	c 29
	FF	03	00	11	00	0D	00	F2	FF	14
	d 30	31	32	33	e 34	35	36	37	38	39
	01	01	01	FF	45	23	01	00	'1'	'2'
	h0	h1	h2	h3	h4	h5	h6	h7	h8	h9
	'3'	'4'	'5'	00	00	00	01	00	'd'	'w'
	50	51	52	53	54	55	56	57	58	59
	'd'	'w'	'd'	'w'	'd'	'w'	03	11	F5	FD
	k 60	61	m 62	63	64	65	66	67	68	69
	1B	00	1C	00	00	00	2A	00	00	00
	70	71	72	73	74	75	76	77	78	79
	34	12	00	00	78	56	00	00	34	12
	80	81	82	83	84	85				
	00	00	F8	56	00	00				

17 = 11h + 0Ah = 1Bh

10 + 11 = 21 ⇒ 15h

17 + 25 = 42

00101010

-128 = 256 - 128 = 128 FF 1000 0000

$2^8 : 2^2 = 2^6$ 0100 0000

256 = 100h ⇒ 256h - 100 = 156

FF - a = -14 = EF

256 - 14 = 239 = EF

1110 1111

0+4-8+18 = -4+18 = 14

0-4+8-18 = -14

256 - 14 = 212 ⇒ 1111 0010 ⇒ F2

III. a). The following 5 ASM code sequences are given :

i). mov eax,-2 / mov ebx,-1 / div bl

ii). mov eax, 65409 / idiv ah / add al,al

iii). mov eax, 255h&255 / mov ebx, 256^256h / mul bh

iv). mov ax, 128|2 / mov bh, 4ah>>2 / sub ah, bh

v). xor eax, eax / lea ebx,[esi] / xlat

Which is the result and effect of every one of the 5 above sequences ? How will be the OF and CF flags set in the above i)-v) cases and why ? Detail the complete effect of every source line providing every involved value in each of the 10 and 16 nummeration bases, signed and unsigned. Explain. In v) sequence replace the last instruction with another (also with no explicit operands!) such that the output effect on the AL register to be the same. Justify and explain every line.

b). The following 6 instructions ASM sequence is given:

```
push ebp
push esp
mov ebx, [esp+4]
sub ebx, 4
add esp, 8
push dword [ebx]
```

; Write one single ASM instruction to have the same effect on the stack ; configuration as the given sequence and explain/justify why the same effect ;is obtained, explaining in detail the effects of every given line.

a) 1) EAX = FF FF FF FE

EBX = FF FF FF FF

div bl ⇒ AX : BL ⇒ FFFF : FF = 65535 : 254

⇒ Error integer overflow

65535 | 254
508 | 258 ∉ [0,255]
1473
1270
=2033
2032
= = = 1

2) EAX = - 127 = FF81

FF = -1

-127 : (-1) = |127| = AL

AH = 0

AL+AL = 127 + 127 = 254 = FE

CF = 0 OF = 1

3) EAX = 00 00 00 55

EBX = 00 00 03 56

mul bh = 03 * 55

= 3 * 85 = 255 = (FF)

00 00 02 55 &
00 00 00 FF
00 00 00 55

00 00 01 00 ^
00 00 02 56
00 00 03 56

4) AX = 0082h

BH = 12h

4Ah = 01001010

>>1 00010010

1 000 0000 |
0 000 0010
1 000 0010

ch - bh = 00 - 12 = EE

5) EAX = 0

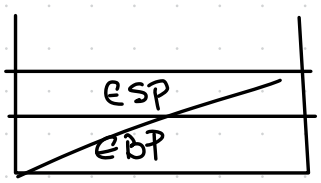
LEA EBX, [esi] EBX = esi address

XLAT ⟶ LODSB !

CF = 1

OF = 0

b)



EBX = EBP

ESP + 8 ⇒ clear stack

⇒ push dword [ebx]

idem ! push [ebp+4]

31 ian 2020

Prezentați și justificați structura din memorie a următorului segment:

```
segment data use32 class=data
    x  dw -256, 256h
    y  dw 256|-256, 256h & 256
    z  db $-z, y-x
       db 'y'-'x', 'y-x'
    a  db  512 >> 2, -512 << 2
    b  dw z-a, !(z-a)
    c  dd ($-b) + (d-$), $-2*y+3
    d  db -128, 128^(~128)
    e  times 2 resw 6
       times 2 dd 1234h, 5678h
```

Dacă identificați date sau linii sursă pe care le considerați incorecte sintactic, justificați motivul și ignorați apoi acele valori sau linii în construirea modului de reprezentare în memorie a segmentului de date.

x 0	1	2	3	y 4	5	6	7	z 8	9
00	FF	56	02	00	FF	00	00	00	04
10	11	12	13	a 14	15	b 16	17	18	19
01	'y'	'_'	'x'	80	00	FA	FF	00	00
c 20	21	22	23	24	25	26	d 27	28	e 29
0B	00	00	00	13	00	00	80	FF	—
30	31	32	33	34	35	36	37	38	39
—	—	—	—	—	—	—	—	—	—
40	41	42	43	44	45	46	47	48	49
—	—	—	—	—	—	—	—	—	—
50	51	52	53	54	55	56	57	58	59
—	—	—	34	12	00	00	48	56	00
60	61	62	63	64	65	66	67	68	
00	34	12	00	00	48	56	00	00	

1 0000 0000 0000 0000 − 0000 0001 0000 0000 = 1111 1111 0000 0000

FF00

'y' − 'x' = 01

100h | FF00h = FF00h

1 0000 0000 0000 0000 − 10 0000 0000 = 1111 1110 0000 0000

256 & 100 = 000

24 − 8 + 3 = 16 + 3

4 + 7 = 11

128 = 1000 0000 ^ 0111 1111

FE00 << 2 = 1111 1000 0000 0000

8 − 14 = −4

!(FF FA) = 00 0 0110

a). Explicați functionarea și efectul fiecăreia dintre următoarele instrucțiuni :

1). lea eax,[6+esp]
2). mov eax, 6+esp
3). movsx ax,[6+esp]
4). mov ebp, [6+ebp*2]
5). mov [6+ebp*2], 12
6). mov ebp, [ebx+esp]
7). movsx [6+esp],eax
8). mov [6+esp*2],eax
9) mov [6+ebp*2], [6+esp]
10). movzx eax, [6+ebp*2]

b). Se da urmatoarea secventa de 12 instructiuni ASM:

push edx / push eax / pop edx / xor dh,dh / shl edx, 16 / clc
rcr edx,16 / add edx, ebx / push edx / pop esi / lodsb / pop edx

Scrieti o singura instructiune ASM care sa aiba acelasi efect ca si secventa data si explicati/justificati de ce se obtine acelasi efect. Detaliati efectul fiecarei linii din seventa data.

a) 1) eax = address esp +6

2) invalid operand type, that's why lea exists

3) moves a word from the address [6+esp] into ax

4) ——— 11 ———

5) syntax error

6) ebx index, esp base

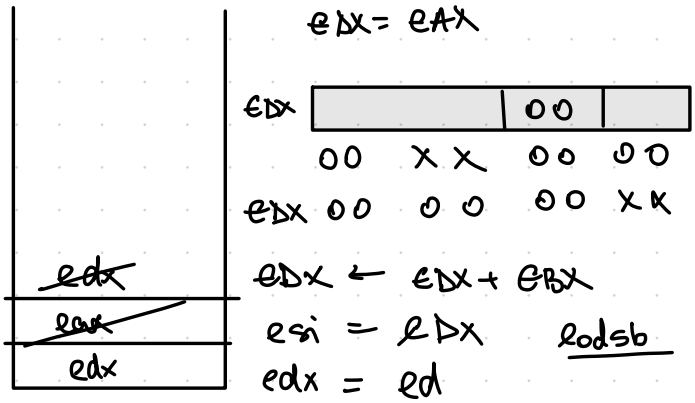
7) valoarea din eax o pune la [6+esp]

8) esp NU poate fi index ⇒ syntax error

9) adresă memorie ⇒ error

10) dimensiune a [6+ebp*2] nu e specificată (poate fi byte sau word)

b)



edx = eax

edx 00

00 xx 00 00

edx 00 00 00 xx

edx ← edx + ebx

esi = edx lodsb

edx = ed

AL = [ebx+AL] = XLAT

8 feb 2018

II. a). Which is the MINIMUM number of bits necessary for representing: i). 61 ii). -62 iii). 130 iv). -129 Justify and explain your answer by detailing the representation mechanisms of these values (Examples: the minimum number of bytes for representing number 5 is 3: 101b; the minimum number of bytes for representing number 16 is 5: 10000b). For each of the above numbers give the representation in base 2 and then in base 16.

b). The following ASM sequence is given:

```
xor ah, ah
cwde          ; Write one single ASM instruction having the same effect as the given
add ebx, eax  ; sequence and explain/justify why the same effect is accomplished.
mov al, [ebx] ; Detail the exact effects of every source line from the given sequence
```

a) i) $61 < 64 = 2^6 \Rightarrow$ can be rep. on 6 bits

$111101 \Rightarrow 3Dh$

ii) -62

$\underbrace{1000}_{8}\underbrace{0010}_{2h} \Rightarrow$ 4 bits

iii) $130 \in [0, 255] \Rightarrow$ 8 bits

$82h$

iv) $-129 \notin [-128, 127] \Rightarrow$ 9 bits

$17Fh$

b) $AH = 0$

cwde $\Rightarrow$ AX $\rightarrow$ EAX keeping the sign 0 $\Rightarrow$

$EAX = 00\ 00\ 00\ AL$

add ebx, eax $\Rightarrow$ EBX = EBX + AL

mov al, [ebx] $\Rightarrow$ XLAT

III. The following ASM sequences are given:

a). mov ax, 1000h
mov bl, 1000b+10b
div bl

b). mov ah, 0bch
mov al, 0deh
add ah,al

c). mov ax, 1001h
mov bx, 1111b
imul bl

d). mov dh, 62h
mov ch, 200
sub dh,ch

Which is the result of the each of the above operations and instructions ? Detail the effect of every source line providing every involved value in each of the 10 and 16 nummeration bases, signed and unsigned. Define and explain the overflow concept, the way in which the x86 architecture reacts at such an event in the case of each of the basic arithmetic operations and specify if such a situation appears in one or more of the above cases, justifying its cause and effect.

a) $AX = 1000h = 4096$

$BL = 1010b = 0Ah = 10$

div bl $\Rightarrow$ AX : BL $\Rightarrow$ $4096 : 10 = 409,6 \notin [0, 255] \Rightarrow$ division overflow

b) $AH = BCh = 188$ or $1011\ 1100 = -68$

$AL = DCh = 222$ or -34

add AH, AL $\Rightarrow$ $AH = BCh + DEh$ $\Rightarrow$ OF = 0, CF = 1

$$\begin{array}{r} BC+ \\ DE \\ \hline 18A \end{array}$$

c) $AX = 1001h$

$BX = 00FFh$

$BL * AL = FF * 01 = FF \Rightarrow CF = OF = 0$

d) $DH = 62h$

$CH = 11001000_2 = C8h$

$DH - CH$: $62h - C8h = 9A$

CF = 1

OF = 1

9 feb 2018

1. Present the corresponding memory layout of the following segment (its structure and corresponding memory values), justifying in context the reason for obtaining each of the stored values:

```
data segment
a1 db '256,-256'
a2 dw 256, 256h
a3 dw $-a2
a4 equ -256/4   nu ocupă spațiu în memorie
a5 db 128>>1, -128<<1
a6 dw a2-a5, ~(a2-a5)
a7 dd [a2], [a2]   - not determinable at asm time
a8 dd 256h*256, 256256h
a9 dd ($-a8) + (a10-$)
a10 dw -255, 256
a11 resb 6
a12 times 4 dw 256
a13 dw times 4 -128   syntax error
times 2 resw 2
times 2 dd 12345678h
data ends
```

syntax error

If there are data elements or source lines which you consider to be syntactically incorrect, motivate and explain your decision and ignore further that values or lines in providing the requested memory layout.

offset :

a_1 0	1	2	3	4	5	6	7	a_2 8	9
'2'	'5'	'6'	','	'-'	'2'	'5'	'6'	00	01
10	11	a_3 12	13	a_5 14	15	a_6 16	17	18	19
56	02	04	00	40	00	FA	FF	05	00
a_8 20	21	22	23	24	25	26	27	a_9 28	29
56	03	00	00	56	62	25	00	0C	00
30	31	a_{10} 32	33	34	35	a_4 36	37	38	39
00	00	01	FF	00	01	00	00	00	00
40	41	a_{12} 42	43	44	45	46	47	48	49
00	00	00	01	00	01	00	01	00	01
50	51	52	53	54	55	56	57	58	59
00	00	00	00	00	00	00	00	78	56
60	61	62	63	64	65	66	67	68	69
34	12	48	56	34	12				

a_5 : 128 >>1 : 1000 0000 >> 1 ⇒ 0100 0000 = 40

-128 << 1 : -256

256 − 256 = 0 ⇒ 00

a_6 : $a_2 - a_5 = -6$ ⇒ FA

$\sim(a_2 - a_5) = \sim FA$: 0000 0101 = 05h

a_8 :

```
  0010 0101 0110 ^
  1 0000 0000
  ---------------
  0011 0101 0110
     3    5   6h
```

a_9 : 8 + 4 = 12 = 0Ch

III a). Present a classification of the following 14 instructions/sequences in a number of categories based on the **"identical effect on the EBX register"** criterium (which must be explained and justified):

```
1). lea ebx, [ebx+6]
2). lea ebx,[bx+6]
3). lea bx,[bx+6]
4). lea bx, [ebx+6]   takes only bx+6 from it
5). mov ebx, ebx+6
6). mov ebx,[ebx+6]
7). movzx ebx,[ebx+6]
8). movzx ebx,[bx+6]
9). add bx,6 ; movzx ebx,bx
10). mov [ebx], dword [bx+6]
11). add ebx,6
12). add bx,6
13). push [ebx+6]; pop ebx
14). xchg ebx,[ebx+6]
```

For example (types of possible answers): i).all the instructions have the same effect on the EBX register (so we have only 1 category); ii). Each instruction has a different effect on EBX (so in this case we have though 10 categories); iii). instructions 2 and 5 represents one category (because they accomplish.... etc) and 1, 3 and 7 another category (having the effect...) etc.... giving all the necessary reasons. Explain in detail which is the effect of each given instruction and based on these explanations provide and explain the required classification.

b). The following ASM sequence is given:

```
cld
push ax
mov ax, [esp+2]
stosw
lea edi, [edi-2]
add esp,4
```

notify → pop ax

; Write one single ASM instruction having the same effect as the given sequence
; (except the value from AX) and explain/justify why the same effect is accomplished

a)

1) ebx ← ebx+6
2) ebx ← bx+6
3) bx ← bx+6
4) bx ← bx+6
5) syntax error (not offset-specification)
6) ebx ← [ebx+6] conținutul de la adresa ebx+6
7) syntax error (size not specified)
8) syntax error (size)
9) bx ← bx+6
   ebx ← bx
10) syntax error (ambii din memorie)
11) ebx ← ebx+6
12) bx ← bx+6
13) syntax error (word / dword not specified)
14) ebx ← [ebx+6]

b)

```
cld     ; DF = 0
push ax                 | push ax
mov ax, [esp-2]         | pop ax
stosw   ; mov word[edi], AX   => [EDI] = AX
          edi+=2
lea edi, [edi-2]   => EDI -= 2
add esp, 4 ; returnez esp.

        mov word[edi], ax
```

14 feb 2022

II. Explicați funcționarea și efectul fiecăreia dintre următoarele linii de cod sursă, presupunând că offsetul de început al segmentului de date este 0:

```
a1 dd -103, a6
a2 dw 1&2, 3|4, 5^6, 1|2, 3^4, 5&6
   dw 1^2, 3&4, 5|6
a3 dw $-$$, $$-a3
a4 db ~((-1)^0bbh), 1^0bbh
a5 dd -129<<1Fh, 81h>>0111b
a6 dd 'a1a2a3a4a5',(a6-a5)<<(a5-a4)
a7 times 4 dw a2, a2+1
a8 dw !(a2-a1), !(a2-1)
.....
movsx ebx,[ebx+ebp]   syntax error
xchg ebx,[ebx+esp]
lea eax, [ebx+esp]
lea eax, [ebx+ebp]
mov ax, a+b ;unde a db -1 și b dw 80h
                      syntax error
```

Scrieți o instrucțiune/o linie cod sursa ASM cu efect identic asupra structurii stivei ca și "push eax"

push eax

offset :

a_1 0	1	2	3	4	5	6	7	a_2 8	9
99	FF	FF	FF	1E	00	00	00	00	00
10	11	12	13	14	15	16	17	18	19
07	00	03	00	03	00	07	00	04	00
20	21	22	23	24	25	a_3 26	27	28	29
03	00	00	00	07	00	1A	00	1A	00
a_4 30	31	a_5 32	33	34	35	36	37	38	39
BB	BA	00	00	00	80	01	00	00	00
a_7 40	41	42	43	44	45	46	47	48	49
08	00	09	00	08	00	09	00	08	00
50	51	a_8 52	53	54	55	56	57	58	59
09	00	00	00	00	00				
60	61	62	63	64	65	66	67	68	69

$|-103) = 103 = 2^6 + 2^5 + 2^2 + 2^1 + 2^0 = 01100111_b$

$C_2(103) = 10011001 = 99h$

103 − 96 = 7

$01_b \;\; 11\;11 \;\; 101\;^$
$10 \;\; 100 \;\; 110$
$00 \;\; 111 \;\; 11$

$!(a_2 - a_1) \;\; !(a_2 - 1)$
$!8 = 0 \;\; !4 = 0$

$\sim((-1)\,^\wedge\, 066h) \;\; 1\,^\wedge\, 066h$
$FF\,^\wedge\,BB$

$\Rightarrow$ 1111 1111 ^ / 1011 1011 / 0100 0100 $= 44h$

$\sim 44h = BBh$

0000 0001 ^ / 1011 1011 / 1011 1010 → B A

1000 0001
FF 0111 1111
FF 7F h

$|-129) = 129 = 2^7 + 2^0 =$

$129 * 2^{31}$

B1h >> 7 ~~1011 0001~~ = 00000001

xchg ebx, [ebx + esp] ebx ← [ebx + esp]

eax ← [ebx + ebp]

eax ← [ebx + esp]

push dword [ebx + ebp]

[ebx] + [ebp]

mov [esp-4], eax

III. Se dau următoarele următoarele secvențe de program ASM:

a). mov ax, 0100h
mov bx, 1000+10b
idiv bl

b). mov ah, 0cdh
mov al, 0ebh
add ah,al

c). mov ax, 1010h
mov bx, 1111b
mul bl

d). mov dh, 200
mov ch, 62h
sub dh,ch

Care este rezultatul efectuării operațiilor ? Detaliați efectul fiecărei linii sursă exprimând fiecare dintre valorile implicate în fiecare din bazele 10 și 16, cu semn și fără semn. Definiți și explicați conceptul de depășire (*overflow*), modul în care reactioneaza arhitectura x86 la aparitia unui astfel de eveniment in cazul fiecarei operatii aritmetice de baza in parte și precizați dacă apare în vreunul din cazurile de mai sus, justificând cauza și efectul său.

a) $AX = 0100_h$

$BX = 03EA_h$

$AX : BL = 256 : (-22)$

$\Rightarrow AH = 14 \quad AL = -11$

$100_2 = 0111110.1010$

$1110\ 1010$

$0001\ 0110 = 22$

$16\ \ 42$

b) $AH = CD_h$

$AL = EB_h$

$AH + AL =$ $\begin{array}{r} {\scriptstyle 1} \\ CD+ \\ EB \\ \hline (1)\ B8 \end{array}$

$CF = 1$

$OF = 0$

c) $AX = 1010_h$

$BX = 000F_h$

$AL * BL = 10_h * 0F_h = 16 * 15 = 240 \in [0, 255] \Rightarrow CF = OF = 0$

d) $DH = 11001000 = C8_h$

$CH = 62_h$

$DH - CH =$ $\begin{array}{r} C8_h \\ 62 \\ \hline 66_h \end{array}$

$OF = 1$

$CF = 0$

18 feb 2019

```
bits 32 ; assembling for the 32 bits architecture

; declare the EntryPoint (a label defining the very first instruction of the program)
global start

; declare external functions needed by our program
extern exit, printf
import exit msvcrt.dll
import printf msvcrt.dll

; our data is declared here (the variables needed by our program)
segment data use32 class=data
    sir dd 1234A678h, 12785634h, 1A4D3C26h
    sir_len equ ($-sir)/4 ; number of dwords we have in 'sir'
    new_sir times sir_len dw 0 ; set the exact space for the new 'sir'
    format db "%d", 0
    cnt dd 0

; our code starts here
segment code use32 class=code
    start:
        mov ecx, sir_len
        mov esi, sir
        mov edi, new_sir
        jecxz end_sir
            parcurge_sir:
                lodsb ; low part of the low byte, esi += 2
                lodsb ; high part of the low byte, esi += 2
                stosb ; [edi] = AX, edi+=2

                lodsb ; low part of the high byte
                lodsb ; high part of the high byte
                stosb ; [edi] = AX, edi+=2

            loop parcurge_sir
        end_sir:
        mov ecx, sir_len
        mov esi, new_sir
        mov edx, 0
        jecxz endd
            count_bits:
                lodsw
                mov ebx, 0
                    bitss:
                        cmp ebx, 16
                        je finish
                        shl ax, 1
                        adc dx, 0
                        inc ebx
```

```
            jmp bitss
        finish:
    loop count_bits
endd:
mov dword[cnt], edx
push dword[cnt]
push dword format
call [printf]
add esp, 4*2

push    dword 0      ; push the parameter for exit onto the stack
call    [exit]       ; call exit to terminate the program
```