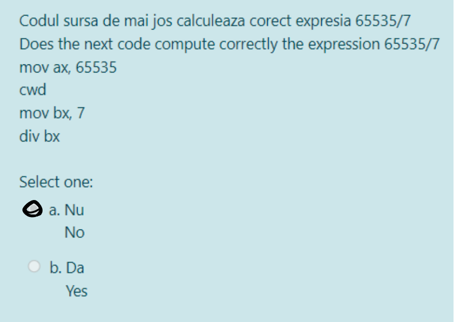
Grile

Saturday, December 4, 2021 3:39 PM

1



Codul sursa de mai jos calculeaza corect expresia 65535/7
Does the next code compute correctly the expression 65535/7

```
mov ax, 65535
cwd
mov bx, 7
div bx
```

Select one:

[✓ a. Nu
No]

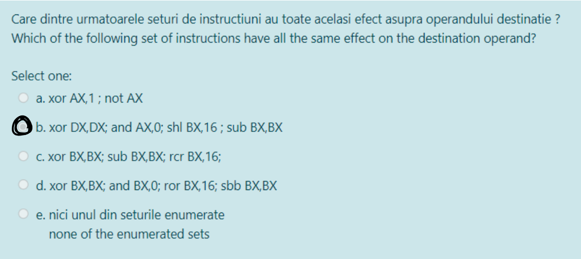
○ b. Da
Yes

Dacă AX e convertit la doubleword și e super mare față BX care e 7 și e word, se produce overflow.

DX:AX: 0000:FFFF
BX: 0007

Soluția: Nu există

2



Care dintre urmatoarele seturi de instructiuni au toate acelasi efect asupra operandului destinatie ?
Which of the following set of instructions have all the same effect on the destination operand?

Select one:

○ a. xor AX,1 ; not AX

[✓ b. xor DX,DX; and AX,0; shl BX,16 ; sub BX,BX]

○ c. xor BX,BX; sub BX,BX; rcr BX,16;

○ d. xor BX,BX; and BX,0; ror BX,16; sbb BX,BX

○ e. nici unul din seturile enumerate
none of the enumerated sets

$\boxed{x \& 0 = 0}$ $\boxed{x \hat{} 1 = \sim x}$

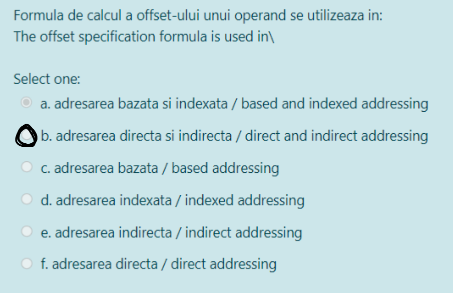
$\boxed{x \mid 1 = 1}$ $\boxed{x \& \sim x = 0}$

$\boxed{x \mid \sim x = 1}$ $\boxed{x \hat{} \sim x = 1}$

$\boxed{x \hat{} x = 0}$

- sbb, rcr: trebuie avut grijă la CF;
- ror: rotește nu shiftează, deci poate să nu fie 0;
- shl=sal: bagă zerouri, iar '— bit o să 'e în CF;
- neg face 2'sc, not face ~x ; not dotine 0 doar daca x=11...1.

3



Formula de calcul a offset-ului unui operand se utilizeaza in:
The offset specification formula is used in\

Select one:

○ a. adresarea bazata si indexata / based and indexed addressing

[✓ b. adresarea directa si indirecta / direct and indirect addressing]

○ c. adresarea bazata / based addressing

○ d. adresarea indexata / indexed addressing

○ e. adresarea indirecta / indirect addressing

○ f. adresarea directa / direct addressing

offset = [base] + ([index] * [scale]) + [displacement]

(2AM formula ♡) ADDRESS COMPUTATION FORMULA

$$\begin{bmatrix} EAX \\ EBX \\ ECX \\ EDX \\ ESP \\ EBP \\ ESI \\ EDI \end{bmatrix} + \left(\begin{bmatrix} EAX \\ EBX \\ ECX \\ EDX \\ EBP \\ ESI \\ EDI \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 4 \\ 8 \end{bmatrix} \right) + \begin{bmatrix} \text{None} \\ 8\text{-bit} \\ 16\text{-bit} \\ 32\text{-bit} \end{bmatrix}$$

fără
ESP
+
RSP

base = sursa
DAR: [EBX + ESP] ⇒ ESP baza
index = destinația
DAR: ' — '' —

[ ] = optional

displacement = constantă

Modes of addressing memory (ACF se folosește peste tot)

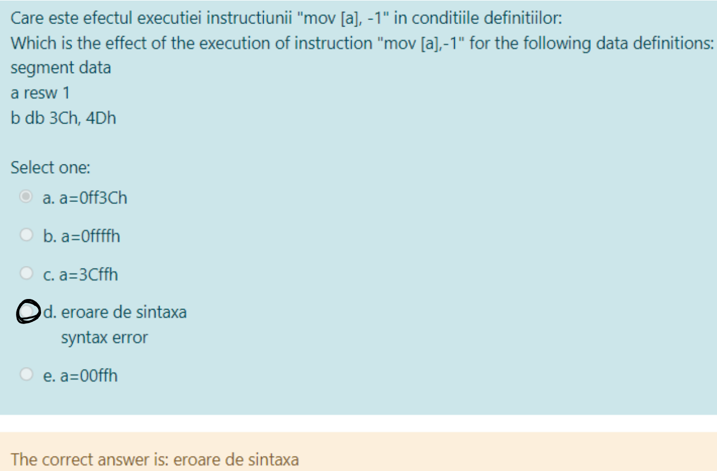
1) Adresare directă - displacement only

2) Adresare indirectă → adresare bazată - base only
→ adresare indexată - index only

Ways of expressing operands

- register mode: ex. mov EAX, 17
  mov [a], AX
- immediate mode: ex. mov EAX, 17
- memory address mode: ex. mov [a], AX
  AICI ADDRESS COMPUTATION FORMULA

(4)



Care este efectul executiei instructiunii "mov [a], -1" in conditiile definitiilor:
Which is the effect of the execution of instruction "mov [a],-1" for the following data definitions:
segment data
a resw 1
b db 3Ch, 4Dh

Select one:

● a. a=0ff3Ch

○ b. a=0ffffh

○ c. a=3Cffh

[✓ ○ d. eroare de sintaxa
syntax error]

○ e. a=00ffh

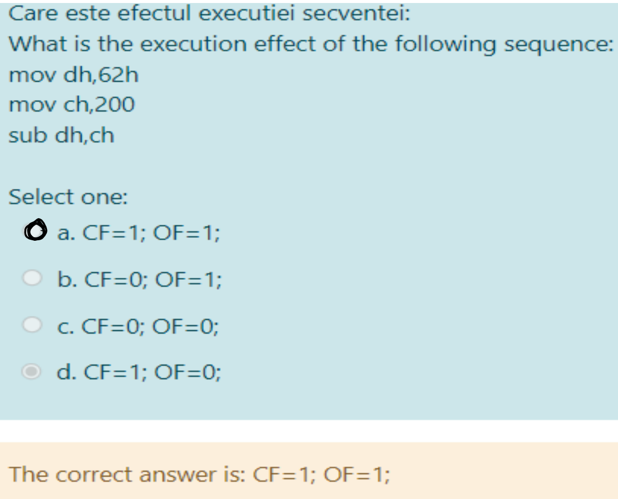
The correct answer is: eroare de sintaxa

Trebuie specificat tipul unei constante sau variabilă de memorie.

Obs. La MOV nu trebuie să fie neapărat de același tip

Soluția: MOV [a], word/byte -1

(5)



Care este efectul executiei secventei:
What is the execution effect of the following sequence:
mov dh,62h
mov ch,200
sub dh,ch

Select one:

[✓ ○ a. CF=1; OF=1;]

○ b. CF=0; OF=1;

○ c. CF=0; OF=0;

● d. CF=1; OF=0;

The correct answer is: CF=1; OF=1;

62h = 98d (+) => 0 sign bit
-200 d (-) => 1 sign bit

-102d(-) => 1 sign bit

Overflow rule: 0-1 = 1 => OF = 1

$$\begin{array}{r} 62_{(16)} - \\ C8_{(16)} \\ \hline FF9A_{(16)} \end{array} \qquad \begin{array}{r} (-1)\,{-1}\;0 \\ 62_{(16)} - \\ C8_{(16)} \\ \hline 9A \end{array}$$

FF = -1 => CF = 1
(we have borrow)

Pentru CF: faci operația pe calculator și vezi că are mai multe cifre

Pentru OF: Rules: 0+0=1, 1+1=0, 0-1=1, 1-0=0

Add + Sub: CF dacă e unsigned, OF dacă e signed

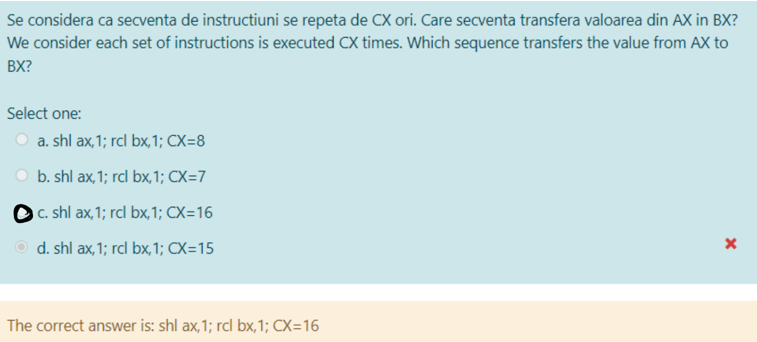
Mul: OF (NEVER), dar { OF=CF=0 dacă size(res) = size(ops)
OF=CF=1 altfel }

Div: OF = FATAL (Runtime Error), OF/CF undefined

OF=1, SF=1 => CF=0

OF=1, SF=1 ⇒ CF=0

6



Se considera ca secventa de instructiuni se repeta de CX ori. Care secventa transfera valoarea din AX in BX?
We consider each set of instructions is executed CX times. Which sequence transfers the value from AX to BX?

Select one:

○ a. shl ax,1; rcl bx,1; CX=8

○ b. shl ax,1; rcl bx,1; CX=7

[✓ c. shl ax,1; rcl bx,1; CX=16]

● d. shl ax,1; rcl bx,1; CX=15 ✗

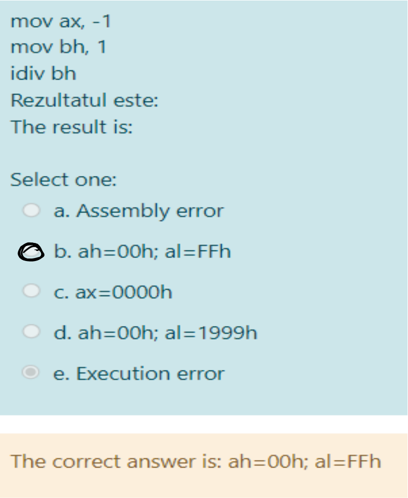
The correct answer is: shl ax,1; rcl bx,1; CX=16

shl bagă ultimul în CF
rcl îl ia
AX are 16 biți
⇒ CX=16

~~shl ax, 16~~
~~rcl bx, 16~~ (bagă ULTIMUL bit în CF)

=)

7



mov ax, -1
mov bh, 1
idiv bh
Rezultatul este:
The result is:

Select one:

○ a. Assembly error

[✓ b. ah=00h; al=FFh]

○ c. ax=0000h

○ d. ah=00h; al=1999h

● e. Execution error

The correct answer is: ah=00h; al=FFh

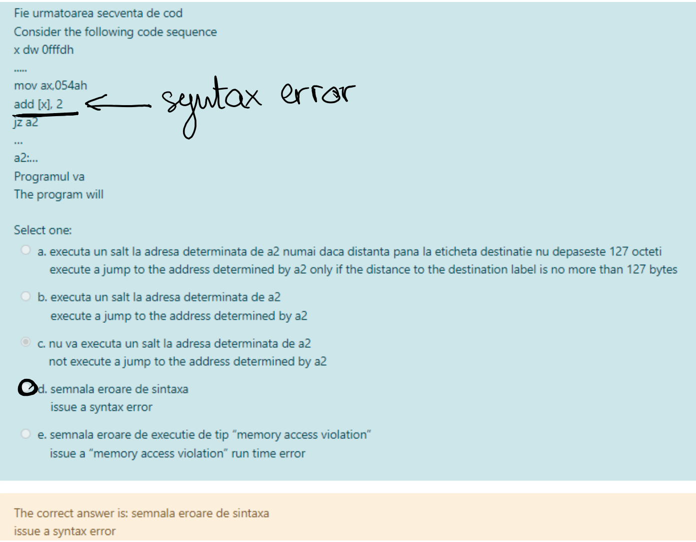
AX ... FFFF

$BX \ldots \underbrace{01}_{BH}\underbrace{00}_{BL}$

$AX : BH = FFFF : 1 \Rightarrow \underset{rest}{AH} : \underset{cât}{AL} = 00 : FF$

Obs. Când se trunchiază câtul, se ia low part (când încape)

8



Fie urmatoarea secventa de cod
Consider the following code sequence

```
x dw 0fffdh
.....
mov ax,054ah
add [x], 2
jz a2
...
a2:...
```

← syntax error

Programul va
The program will

Select one:

○ a. executa un salt la adresa determinata de a2 numai daca distanta pana la eticheta destinatie nu depaseste 127 octeti
execute a jump to the address determined by a2 only if the distance to the destination label is no more than 127 bytes

○ b. executa un salt la adresa determinata de a2
execute a jump to the address determined by a2

● c. nu va executa un salt la adresa determinata de a2
not execute a jump to the address determined by a2

[✓ d. semnala eroare de sintaxa
issue a syntax error]

○ e. semnala eroare de executie de tip “memory access violation”
issue a “memory access violation” run time error

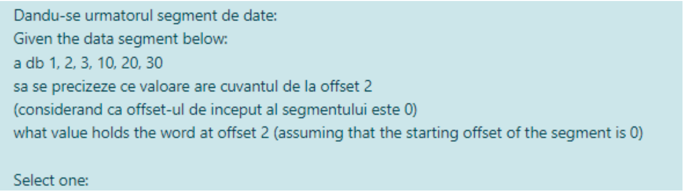
The correct answer is: semnala eroare de sintaxa
issue a syntax error

Când băgăm o constantă sau o variabilă în memorie trebuie specificată mărimea/tipul.
Dacă era corect, jz verifică ZF=1.
ZF în general, se modifică după CMP/TEST. Altfel, dacă rezultatul ultimei operații e 0, ZF=1. Altfel, ZF=0.

Distance between defining and calling a label must be < 127 bytes.

1 instr = x bytes. S-ar fi rezolvat cu un jump FAR (jz e short).

9



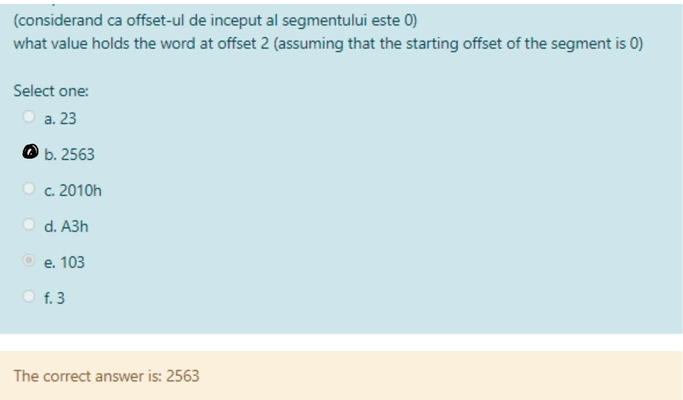
Dandu-se urmatorul segment de date:
Given the data segment below:
a db 1, 2, 3, 10, 20, 30
sa se precizeze ce valoare are cuvantul de la offset 2
(considerand ca offset-ul de inceput al segmentului este 0)
what value holds the word at offset 2 (assuming that the starting offset of the segment is 0)

Select one:

a db 01, 02, 03, 0A, 14, 1E

0	1	2	3	4	5
01	02	03	0A	14	1E

word-ul


(considerand ca offset-ul de inceput al segmentului este 0)
what value holds the word at offset 2 (assuming that the starting offset of the segment is 0)

Select one:

○ a. 23

● b. 2563

○ c. 2010h

○ d. A3h

○ e. 103

○ f. 3

The correct answer is: 2563

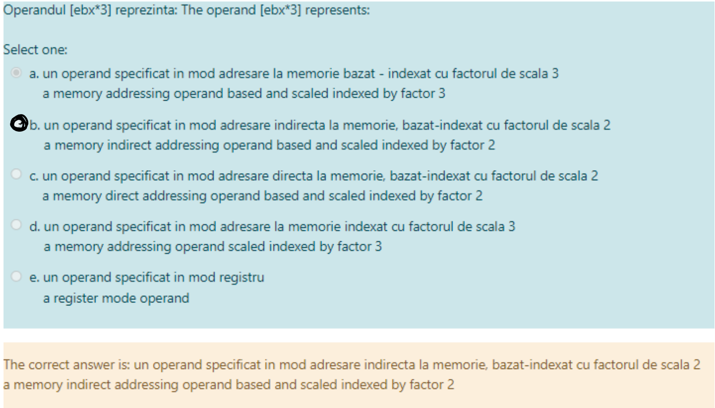
01	02	03	0A	14	1E

a

word-ul de la offset 2

Pentru că Little Endian => 0A03h =
= 2563d

(10)


Operandul [ebx*3] reprezinta: The operand [ebx*3] represents:

Select one:

○ a. un operand specificat in mod adresare la memorie bazat - indexat cu factorul de scala 3
a memory addressing operand based and scaled indexed by factor 3

● b. un operand specificat in mod adresare indirecta la memorie, bazat-indexat cu factorul de scala 2
a memory indirect addressing operand based and scaled indexed by factor 2

○ c. un operand specificat in mod adresare directa la memorie, bazat-indexat cu factorul de scala 2
a memory direct addressing operand based and scaled indexed by factor 2

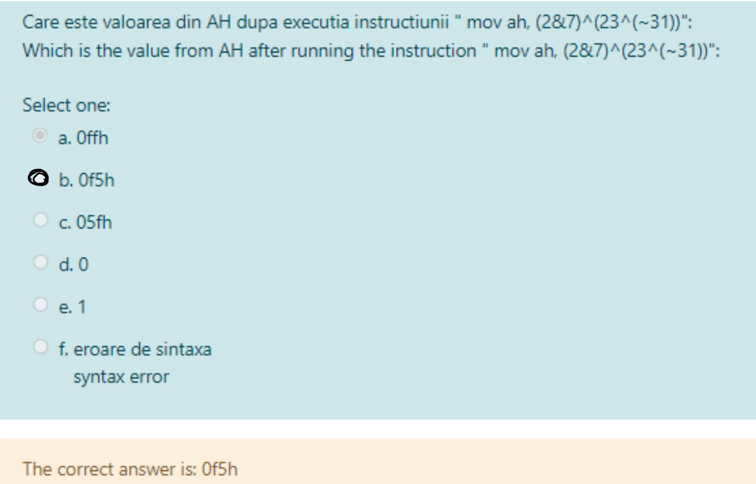
○ d. un operand specificat in mod adresare la memorie indexat cu factorul de scala 3
a memory addressing operand scaled indexed by factor 3

○ e. un operand specificat in mod registru
a register mode operand

The correct answer is: un operand specificat in mod adresare indirecta la memorie, bazat-indexat cu factorul de scala 2
a memory indirect addressing operand based and scaled indexed by factor 2

ebx*3 = ebx + ebx*2
(pentru că scale e doar 1, 2, 4, 8)
Deci e și base, și index, și scale
E indirectă, pentru că nu e displacement only.

(11)


Care este valoarea din AH dupa executia instructiunii " mov ah, (2&7)^(23^(~31))":
Which is the value from AH after running the instruction " mov ah, (2&7)^(23^(~31))":

Select one:

○ a. 0ffh

● b. 0f5h

○ c. 05fh

○ d. 0

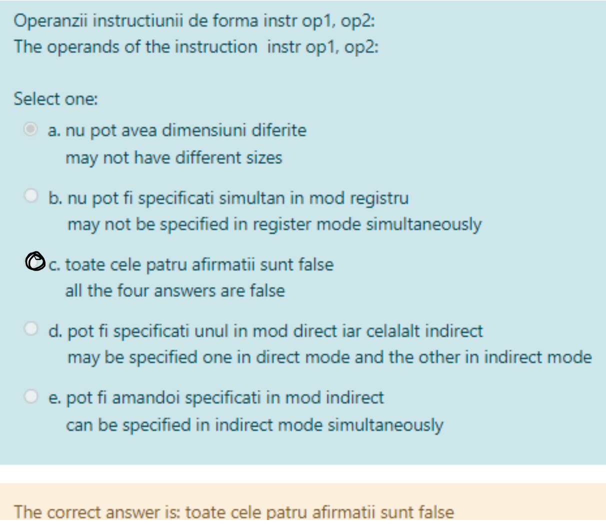
○ e. 1

○ f. eroare de sintaxa
syntax error

The correct answer is: 0f5h

(2&7)^(23^(~31)) = -11 = FFF..F5
AH = byte => AH = F5
(se pot face calculele în binar și de mână)

(12)


Operanzii instructiunii de forma instr op1, op2:
The operands of the instruction instr op1, op2:

Select one:

○ a. nu pot avea dimensiuni diferite
may not have different sizes

○ b. nu pot fi specificati simultan in mod registru
may not be specified in register mode simultaneously

● c. toate cele patru afirmatii sunt false
all the four answers are false

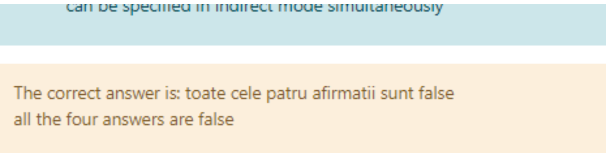
○ d. pot fi specificati unul in mod direct iar celalalt indirect
may be specified one in direct mode and the other in indirect mode

○ e. pot fi amandoi specificati in mod indirect
can be specified in indirect mode simultaneously

The correct answer is: toate cele patru afirmatii sunt false

a. contraexemplu: MOV
b. — ‖ — : sub al, bl

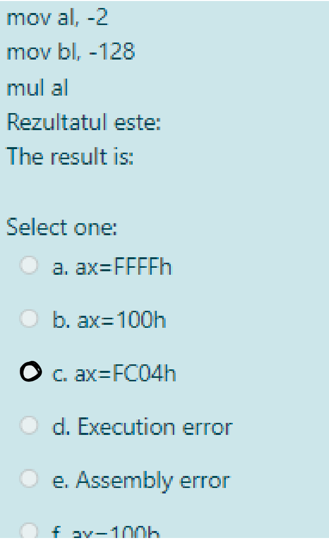
e.
d.



can be specified in indirect mode simultaneously

The correct answer is: toate cele patru afirmatii sunt false
all the four answers are false

13



mov al, -2
mov bl, -128
mul al
Rezultatul este:
The result is:

Select one:

○ a. ax=FFFFh

○ b. ax=100h

[✓ c. ax=FC04h]

○ d. Execution error

○ e. Assembly error

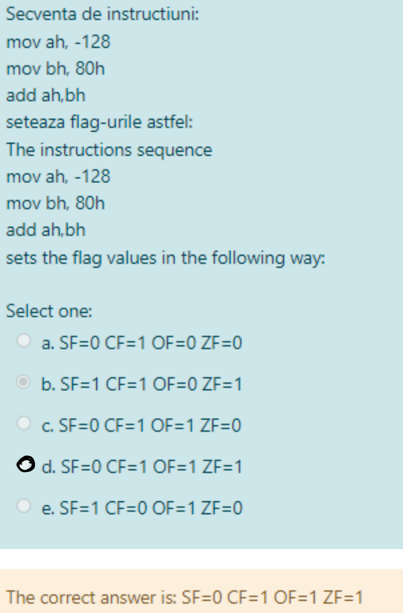
○ f. ax=100h

0-2 = FF...FFE

AL = byte => AL = FE

mul AL => AX = AL*AL = FC04

14



Secventa de instructiuni:
mov ah, -128
mov bh, 80h
add ah,bh
seteaza flag-urile astfel:
The instructions sequence
mov ah, -128
mov bh, 80h
add ah,bh
sets the flag values in the following way:

Select one:

○ a. SF=0 CF=1 OF=0 ZF=0

● b. SF=1 CF=1 OF=0 ZF=1

○ c. SF=0 CF=1 OF=1 ZF=0

[✓ d. SF=0 CF=1 OF=1 ZF=1]

○ e. SF=1 CF=0 OF=1 ZF=0

The correct answer is: SF=0 CF=1 OF=1 ZF=1

AH = 80h

BH = 80h

ADD 80h, 80h => AH = 00h

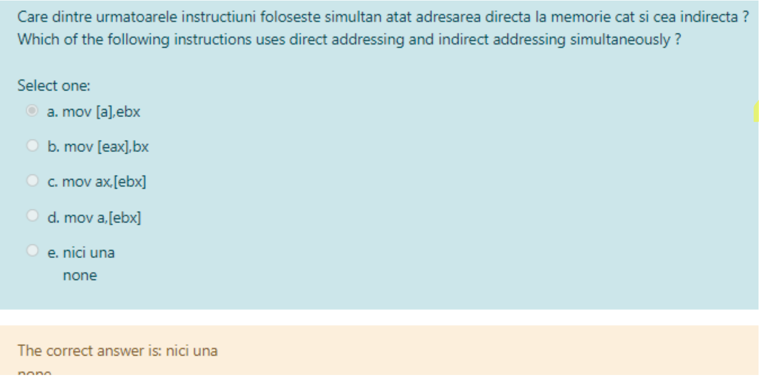
80h + 80h = $\underset{c}{1}$00h => CF = 1

Overflow rule: 1 + 1 = 0 => OF = 1

AH = 00 => ZF = 1

AH = pos (0) => SF = 0

15



Care dintre urmatoarele instructiuni foloseste simultan atat adresarea directa la memorie cat si cea indirecta ?
Which of the following instructions uses direct addressing and indirect addressing simultaneously ?

Select one:

● a. mov [a],ebx

○ b. mov [eax],bx

○ c. mov ax,[ebx]

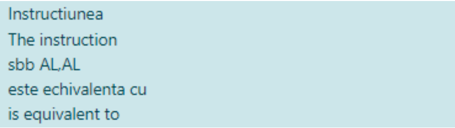
○ d. mov a,[ebx]

○ e. nici una
none

The correct answer is: nici una
none

Ambii operanzi trebuie să fie adresați în mod direct

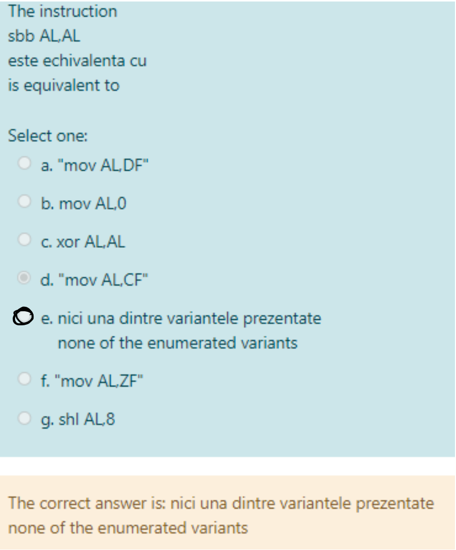
16



Instructiunea
The instruction
sbb AL,AL
este echivalenta cu
is equivalent to

a. nu știi cât e DF

b. dacă carry = 1, nu



The instruction
sbb AL,AL
este echivalenta cu
is equivalent to

Select one:

○ a. "mov AL,DF"

○ b. mov AL,0

○ c. xor AL,AL

○ d. "mov AL,CF"

○ e. nici una dintre variantele prezentate
none of the enumerated variants

○ f. "mov AL,ZF"

○ g. shl AL,8

The correct answer is: nici una dintre variantele prezentate
none of the enumerated variants

[illegible]

b. dacă carry = 1, nu

c. — || —

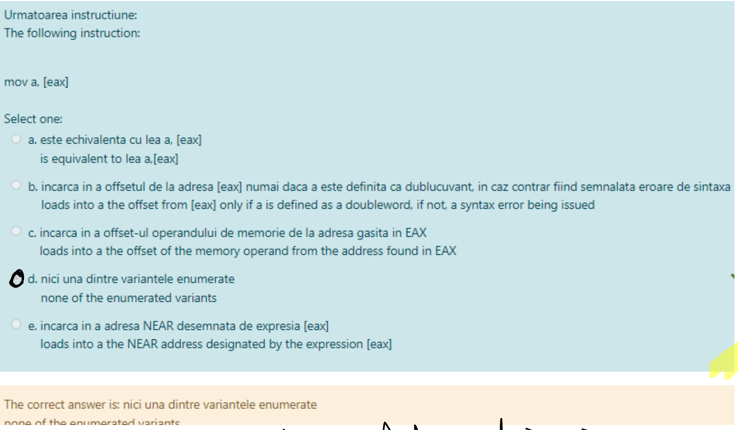
d. SBB ⇒ dest = dest - source - CF (nu plus)

minus

f. ai carry ⇒ ZF=0 (nu se sincronizează)

g. dacă carry = 1, nu

(17)



Urmatoarea instructiune:
The following instruction:

mov a, [eax]

Select one:

○ a. este echivalenta cu lea a, [eax]
is equivalent to lea a,[eax]

○ b. incarca in a offsetul de la adresa [eax] numai daca a este definita ca dublucuvant, in caz contrar fiind semnalata eroare de sintaxa
loads into a the offset from [eax] only if a is defined as a doubleword, if not, a syntax error being issued

○ c. incarca in a offset-ul operandului de memorie de la adresa gasita in EAX
loads into a the offset of the memory operand from the address found in EAX

○ d. nici una dintre variantele enumerate
none of the enumerated variants

○ e. incarca in a adresa NEAR desemnata de expresia [eax]
loads into a the NEAR address designated by the expression [eax]

The correct answer is: nici una dintre variantele enumerate
none of the enumerated variants

lea = load effective address - ia adresa, nu valoarea (ca MOV)

$\text{lea eax, } [\underbrace{\text{ebx} + \text{v}}_{} - \underline{6}]$

2 arithmetic operations at the same time!

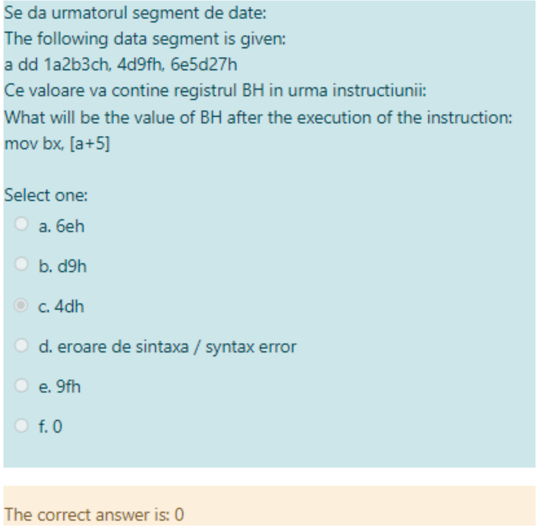
$\boxed{\text{mov a, eax} \approx \text{lea a, [eax]}}$

b. nu e S.E., doar il trunchiază

c. — || —

e. NEAR address = offset
FAR address = offset + segment_selector
Dar cu segment_selector se lucrează pe 16-bit, pe 32-bit nu e la manipulabil.

(18)



Se da urmatorul segment de date:
The following data segment is given:
a dd 1a2b3ch, 4d9fh, 6e5d27h
Ce valoare va contine registrul BH in urma instructiunii:
What will be the value of BH after the execution of the instruction:
mov bx, [a+5]

Select one:

○ a. 6eh

○ b. d9h

○ c. 4dh

○ d. eroare de sintaxa / syntax error

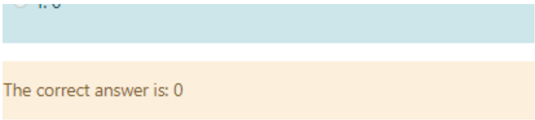
○ e. 9fh

○ f. 0

The correct answer is: 0

0	1	2	3	4	5	6	7	8	9	10	11
3c	2b	1a	00	9f	4d	00	00	27	5d	6e	00

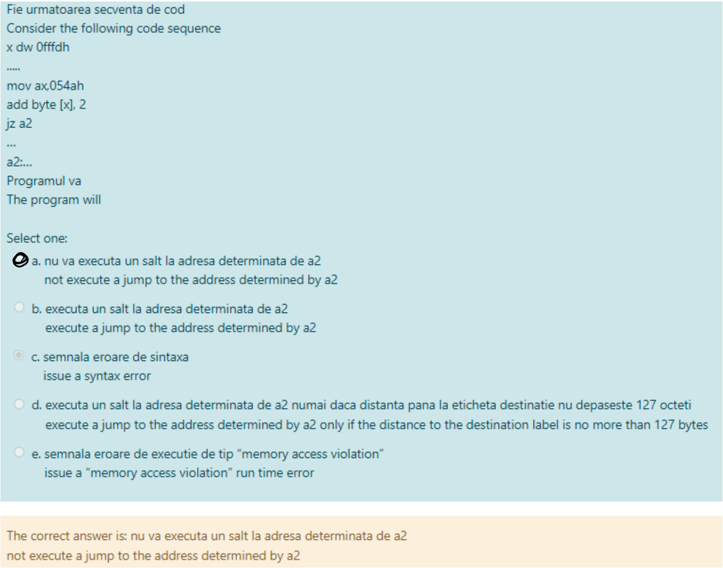
BX = 4dh ⇒ BH = 0



○ f. 0

The correct answer is: 0

19



Fie urmatoarea secventa de cod
Consider the following code sequence
x dw 0fffdh
.....
mov ax,054ah
add byte [x], 2
jz a2
...
a2:...
Programul va
The program will

Select one:

● a. nu va executa un salt la adresa determinata de a2
not execute a jump to the address determined by a2

○ b. executa un salt la adresa determinata de a2
execute a jump to the address determined by a2

◉ c. semnala eroare de sintaxa
issue a syntax error

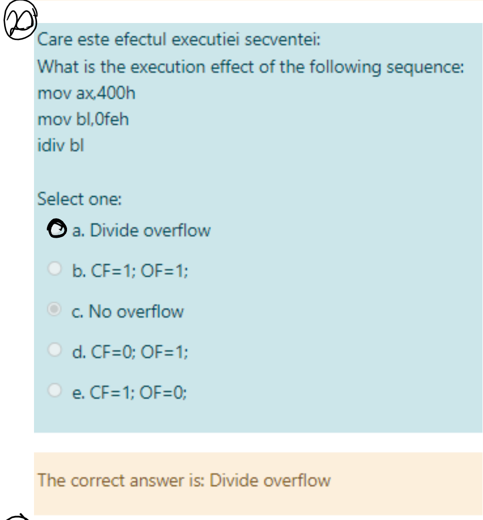
○ d. executa un salt la adresa determinata de a2 numai daca distanta pana la eticheta destinatie nu depaseste 127 octeti
execute a jump to the address determined by a2 only if the distance to the destination label is no more than 127 bytes

○ e. semnala eroare de executie de tip "memory access violation"
issue a "memory access violation" run time error

The correct answer is: nu va executa un salt la adresa determinata de a2
not execute a jump to the address determined by a2

x = FFFDh = -3
add byte [x], 2 => x = -1
=> ZF = 0 => nu se execută jz
- byte e necesar doar la un operand.

20



Care este efectul executiei secventei:
What is the execution effect of the following sequence:
mov ax,400h
mov bl,0feh
idiv bl

Select one:

● a. Divide overflow

○ b. CF=1; OF=1;

◉ c. No overflow

○ d. CF=0; OF=1;

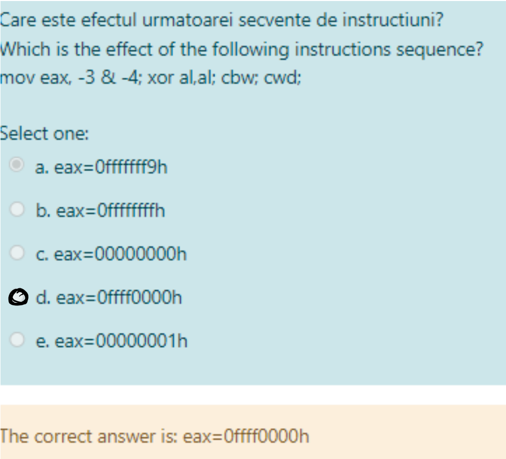
○ e. CF=1; OF=0;

The correct answer is: Divide overflow

400h signed = 1024
0feh signed = -2
———————— (÷)
-512, care nu încape în AL

=> Divide overflow

21



Care este efectul urmatoarei secvente de instructiuni?
Which is the effect of the following instructions sequence?
mov eax, -3 & -4; xor al,al; cbw; cwd;

Select one:

◉ a. eax=0fffffff9h

○ b. eax=0ffffffffh

○ c. eax=00000000h

● d. eax=0ffff0000h

○ e. eax=00000001h

The correct answer is: eax=0ffff0000h

eax = -4d = FFFF FFFC
al = 0

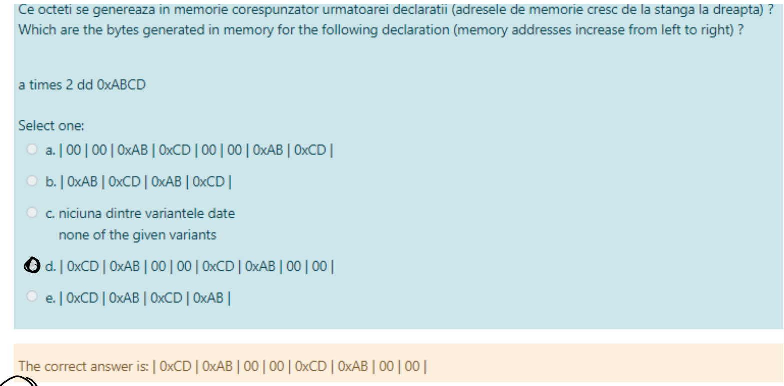
$$EAX = FFFF\ \underbrace{00}_{AH}\ \underbrace{00}_{AL}$$

cBW => AL = AH:AL

} => EAX = ffff0000

cWD => AX = DX:AX

22



Ce octeti se genereaza in memorie corespunzator urmatoarei declaratii (adresele de memorie cresc de la stanga la dreapta) ?
Which are the bytes generated in memory for the following declaration (memory addresses increase from left to right) ?

a times 2 dd 0xABCD

Select one:

○ a. | 00 | 00 | 0xAB | 0xCD | 00 | 00 | 0xAB | 0xCD |

○ b. | 0xAB | 0xCD | 0xAB | 0xCD |

○ c. niciuna dintre variantele date
none of the given variants

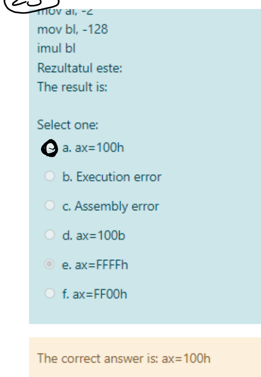
● d. | 0xCD | 0xAB | 00 | 00 | 0xCD | 0xAB | 00 | 00 |

○ e. | 0xCD | 0xAB | 0xCD | 0xAB |

The correct answer is: | 0xCD | 0xAB | 00 | 00 | 0xCD | 0xAB | 00 | 00 |

CD	AB	00	00	CD	AB	00	00

23



mov al, -2
mov bl, -128
imul bl
Rezultatul este:
The result is:

Select one:

● a. ax=100h

○ b. Execution error

○ c. Assembly error

○ d. ax=100b

○ e. ax=FFFFh

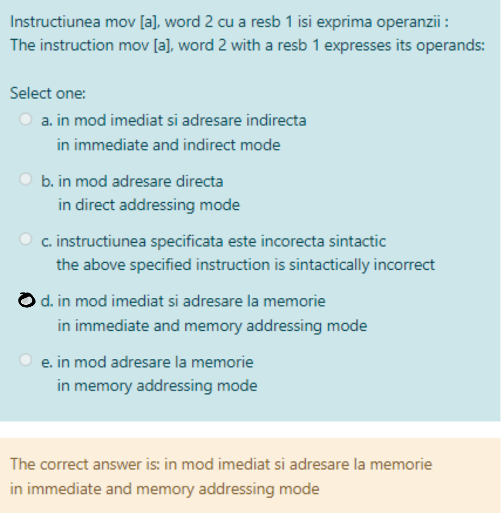
○ f. ax=FF00h

The correct answer is: ax=100h

AL=FE
BL=80

AX=100h

24



Instructiunea mov [a], word 2 cu a resb 1 isi exprima operanzii :
The instruction mov [a], word 2 with a resb 1 expresses its operands:

Select one:

○ a. in mod imediat si adresare indirecta
in immediate and indirect mode

○ b. in mod adresare directa
in direct addressing mode

○ c. instructiunea specificata este incorecta sintactic
the above specified instruction is sintactically incorrect

● d. in mod imediat si adresare la memorie
in immediate and memory addressing mode

○ e. in mod adresare la memorie
in memory addressing mode

The correct answer is: in mod imediat si adresare la memorie
in immediate and memory addressing mode

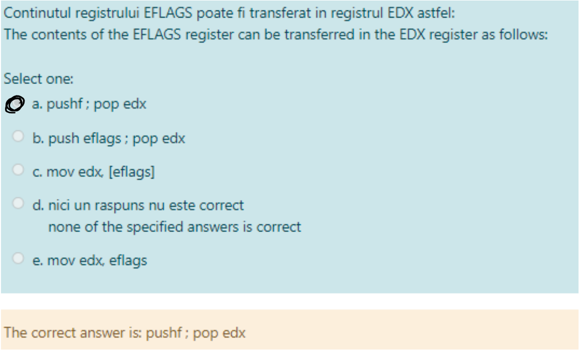
a resb 1

mov [a], word 2

adresare la memorie

adresare imediata

25



Continutul registrului EFLAGS poate fi transferat in registrul EDX astfel:
The contents of the EFLAGS register can be transferred in the EDX register as follows:

Select one:

● a. pushf ; pop edx

○ b. push eflags ; pop edx

○ c. mov edx, [eflags]

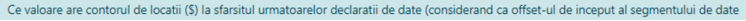
○ d. nici un raspuns nu este correct
none of the specified answers is correct

○ e. mov edx, eflags

The correct answer is: pushf ; pop edx

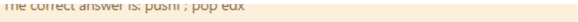
PUSHF e inherited din 16-bit => poate fi folosit și pe 32-bit ca PUSHFD

26



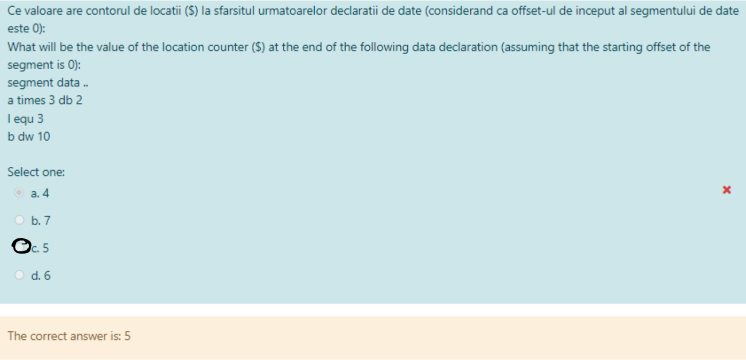
Ce valoare are contorul de locatii ($) la sfarsitul urmatoarelor declaratii de date (considerand ca offset-ul de inceput al segmentului de date

0 1 2 3 4 5



The correct answer is: push ; pop eax

26



Ce valoare are contorul de locatii ($) la sfarsitul urmatoarelor declaratii de date (considerand ca offset-ul de inceput al segmentului de date este 0):
What will be the value of the location counter ($) at the end of the following data declaration (assuming that the starting offset of the segment is 0):

```
segment data ..
a times 3 db 2
l equ 3
b dw 10
```

Select one:

● a. 4 ✗

○ b. 7

[✓ ○ c. 5]

○ d. 6

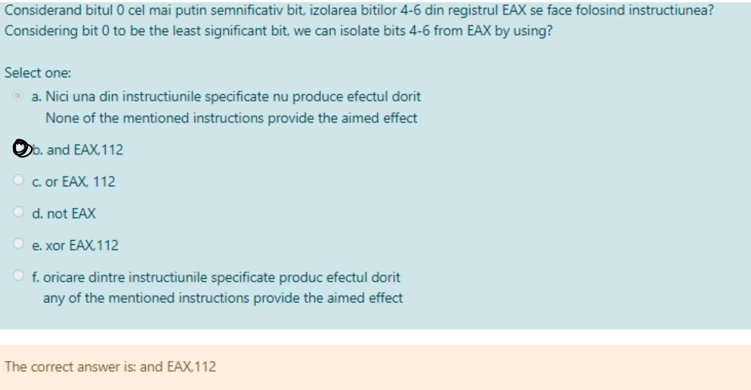
The correct answer is: 5

0	1	2	3	4	5
02	02	02	0A	00	$

-equ = constantă => nu e în memorie!

$ (offset-ul) se pune pe căsuța următoare!! (se indexează de la 0 aici)

27



Considerand bitul 0 cel mai putin semnificativ bit, izolarea bitilor 4-6 din registrul EAX se face folosind instructiunea?
Considering bit 0 to be the least significant bit, we can isolate bits 4-6 from EAX by using?

Select one:

● a. Nici una din instructiunile specificate nu produce efectul dorit
None of the mentioned instructions provide the aimed effect

[✓ ○ b. and EAX,112]

○ c. or EAX, 112

○ d. not EAX

○ e. xor EAX,112

○ f. oricare dintre instructiunile specificate produc efectul dorit
any of the mentioned instructions provide the aimed effect

The correct answer is: and EAX,112

0000 0000

EAX

112 = 0111 0000

$\underbrace{7\,6\,5\,4}$ 3 2 1 0

AND cu 1 pe biți.

Isolating bits 2-4

```
mov al, [a]
and al, 00011100b
```

Setting bits 2-4 to 0

```
mov al, [a]
and al, 11100011b
```

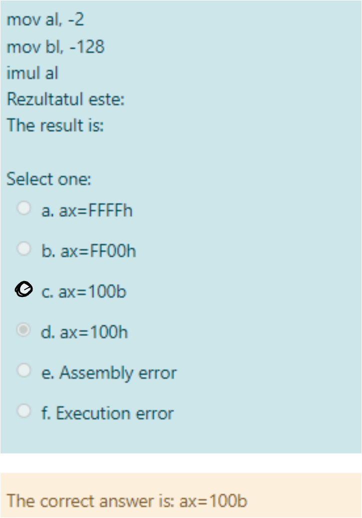
Setting bits 2-4 to 1

```
mov al, [a]
or al, 00011100b
```

Move from x_1-y_1 to x_2-y_2

- compute the difference and rotate properly.

28



```
mov al, -2
mov bl, -128
imul al
```

Rezultatul este:
The result is:

Select one:

○ a. ax=FFFFh

○ b. ax=FF00h

[✓ ○ c. ax=100b]

● d. ax=100h

○ e. Assembly error

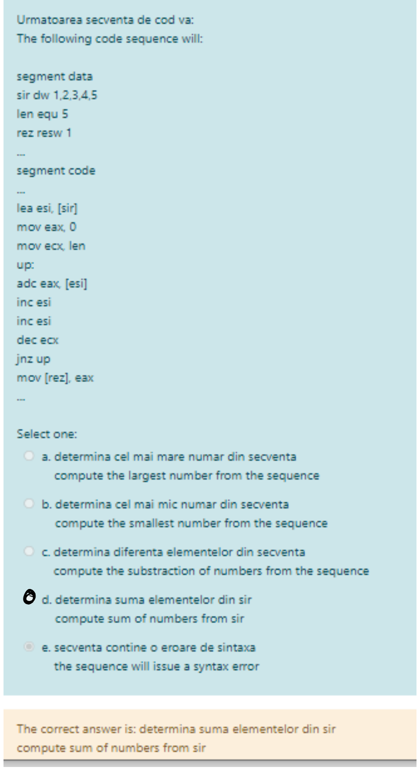
○ f. Execution error

The correct answer is: ax=100b

$-2 * -2 = 4 = 100b$

- înmulțirea se face normal

(29)



Urmatoarea secventa de cod va:
The following code sequence will:

```
segment data
sir dw 1,2,3,4,5
len equ 5
rez resw 1
...
segment code
...
lea esi, [sir]
mov eax, 0
mov ecx, len
up:
adc eax, [esi]
inc esi
inc esi
dec ecx
jnz up
mov [rez], eax
...
```

Select one:

○ a. determina cel mai mare numar din secventa
compute the largest number from the sequence

○ b. determina cel mai mic numar din secventa
compute the smallest number from the sequence

○ c. determina diferenta elementelor din secventa
compute the substraction of numbers from the sequence

● d. determina suma elementelor din sir
compute sum of numbers from sir

◉ e. secventa contine o eroare de sintaxa
the sequence will issue a syntax error

The correct answer is: determina suma elementelor din sir
compute sum of numbers from sir

- inc esi de două ori, pentru că avem word.

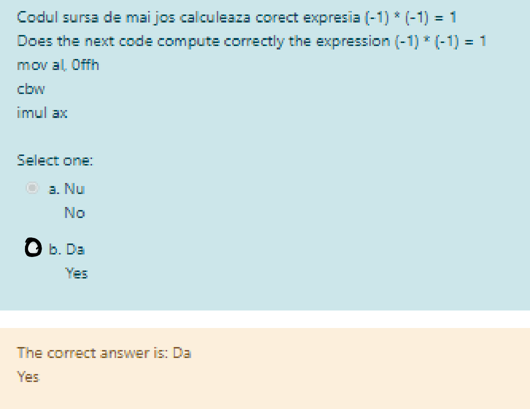
byte: ESI = ESI+1   DF=0

word: ESI = ESI+2

dword: ESI = ESI+4

(decrementare pentru DF=1)

(30)



Codul sursa de mai jos calculeaza corect expresia (-1) * (-1) = 1
Does the next code compute correctly the expression (-1) * (-1) = 1

```
mov al, 0ffh
cbw
imul ax
```

Select one:

◉ a. Nu
No

● b. Da
Yes

The correct answer is: Da
Yes

AL = -1

AL → AH:AL = 00:FF