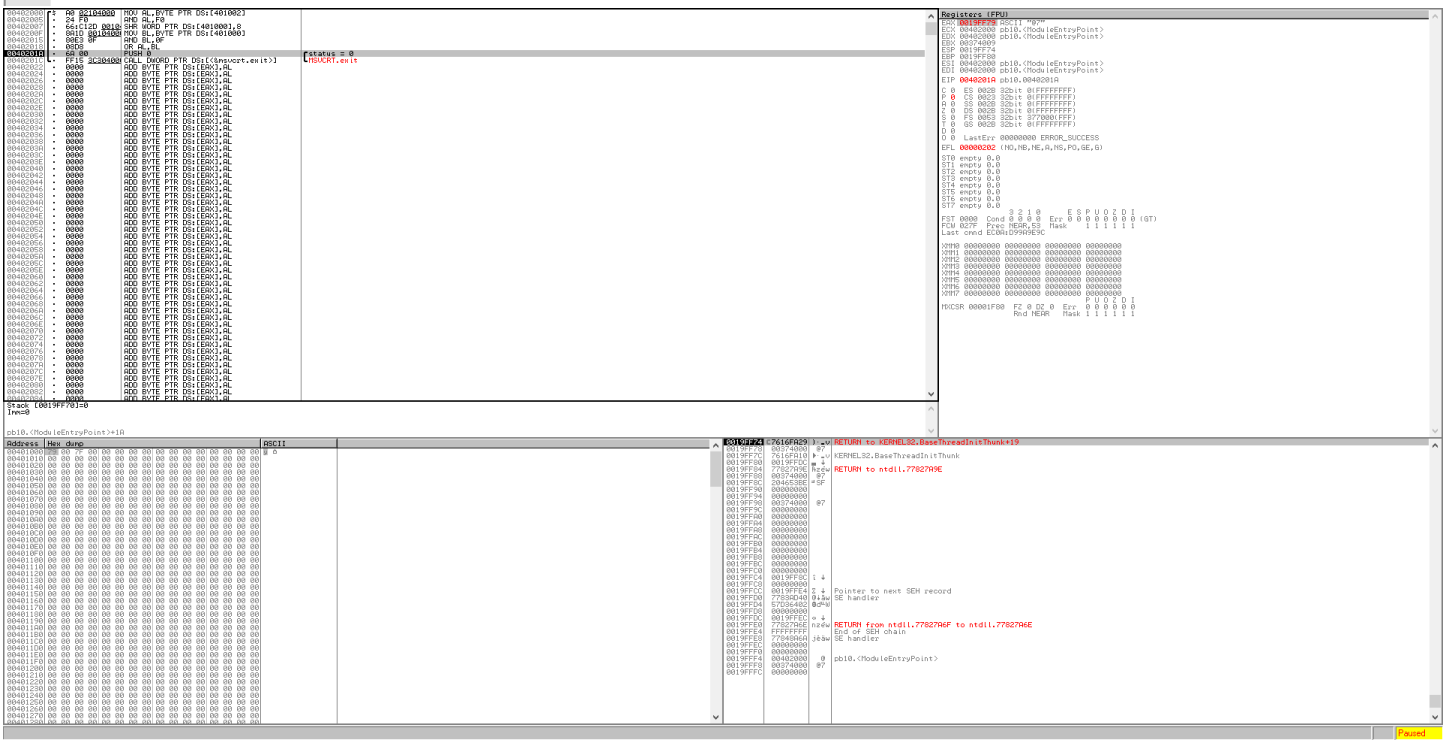
Pb10

10. Sa se inlocuiasca bitii 0-3 ai octetului B cu bitii 8-11 ai cuvantului A.

```
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               ; tell nasm that exit exists even if we won't be
defining it
import exit msvcrt.dll    ; exit is a function that ends the calling process.
It is defined in msvcrt.dll
                          ; msvcrt.dll contains exit, printf and all the other
important C-runtime specific functions

; our data is declared here (the variables needed by our program)
segment data use32 class=data
        ;FEDCBA9876543210
    a dw 0111 1001 0111 0011b ;=7973h 1001- biti 8-11
    b db 01111111b ;=7Fh
; [b]=0111 1001=79h
; our code starts here
segment code use32 class=code
    start:
        mov AL,[b];AL=[b]
        and AL,11110000b ;zerorizam ultimii 4 biti ai lui b, ceilalti raman
neschimbati AL=0111 0000b =70h

        shr word[a],8        ;shiftam spre drepata cu 8 pozitii continutul lui
a, astfel aliniem pozitiile ce trebuie schimbate [a]=0000 0000 0111 1001=
0079H
        mov BL,byte[a]   ;selectam primul octet din a BL=[a]=0111 1001
        and BL,00001111b ;zerorizam restul continutului astfel incat raman
doar cu bitii pe care vrem sa ii modificam BL=0000 1001 =09h

        or AL,BL         ;inlocuim bitii AL=0111 1001=79h

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

```
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               ; tell nasm that exit exists even if we won't be defining it
import exit msvcrt.dll    ; exit is a function that ends the calling process. It is defined in msvcrt.dll
                          ; msvcrt.dll contains exit, printf and all the other important C-runtime specific functions

; our data is declared here (the variables needed by our program)
segment data use32 class=data
        ;FEDCBA9876543210
    a dw 0111 1001 0111 0011b ;=7973h 1001- biti 8-11
    b db 01111111b ;=7Fh
; [b]=0111 1001=79h
; our code starts here
segment code use32 class=code
    start:
        mov AL,[b];AL=[b]
        and AL,11110000b ;zerorizam ultimii 4 biti ai lui b, ceilalti raman neschimbati AL=0111 0000b =70h

        shr word[a],8        ;shiftam spre drepata cu 8 pozitii continutul lui a, astfel aliniem pozitiile ce trebuie schimbate [a]=0000 0000 0111 1001= 0079H
        mov BL,byte[a]   ;selectam primul octet din a BL=[a]=0111 1001
        and BL,00001111b ;zerorizam restul continutului astfel incat raman doar cu bitii pe care vrem sa ii modificam BL=0000 1001 =09h

        or AL,BL         ;inlocuim bitii AL=0111 1001=79h

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



Pb18

18. Se da un cuvant A. Sa se obtina dublucuvantul B astfel:

bitii 0-3 ai lui B sunt 0;

bitii 4-7 ai lui B sunt bitii 8-11 ai lui A

bitii 8-9 si 10-11 ai lui B sunt bitii 0-1 inversati ca valoare ai lui A (deci de 2 ori) ;

bitii 12-15 ai lui B sunt biti de 1

bitii 16-31 ai lui B sunt identici cu bitii 0-15 ai lui B.

```
; our data is declared here (the variables needed by our program)
segment data use32 class=data
        ;FEDCBA9876543210
    a dw 1010011011101101b ;=A6ED
    b resw 2

;0-3 = 0
;4-7 = 8-11 a
;8-9,10-11 = 0-1 a inversati
;12-15 = 1
;[b]=1111 1010 0110 0000 1111 1010 0110 0000=FA60FA60h
; our code starts here
segment code use32 class=code
    start:
        ;0-3 = 0
        ;initializam EAX(unde vom stoca si vom lucra cu variabila [b]) astfel indeplinind conditia ca primii 4 biti sa fie 0 EAX=00000000h
        mov EAX,0

        ;4-7 = 8-11 a
        mov EBX,0 ;initializam EBX cu 0 pentru a facilita interactiunea cu EAX
        mov BX,[a] ; folosim BX ca o copie pentru [a] BX=[a]
        and BX,0000111100000000b ; izolam biti 8-11 BX=0000 0110 0000 0000b=0600h
        shr BX,4 ;rotim la dreapta cu 4 pozitii astfel incat sa aliniem continutul bitilor
        or EAX,EBX

        ;8-9,10-11 = 0-1 a , inversati
        mov EBX,0 ;initializam EBX cu 0 pentru a facilita interactiunea cu EAX
        mov BX,[a] ; folosim BX ca o copie pentru [a] BX=[a]
        neg BX ; negam toti biti din BX
        sub BX,1 ; neg ne returneaza complementul fata de 2, noi avem de nevoie de cel fata de 1
        and BX,0000000000000011b ; izolam biti 0-1 BX=0000000000000010b=0002h astfel avem pe pozitie ultimii 2 biti, inversati
        shl BX,8 ;shiftam la stanga cu 8 pozitii, astfel aliniind biti cu 8-9
        or EAX,EBX ; punem bitii pe pozitia 8-9    EAX=0000 0000 0000 0000 0000 0010 0110 0000b = 00000260h
        shl BX,2 ;shiftam la stanga cu inca 2 pozitii astfel sunt aliniati cu pozitiile 10-11
        or EAX,EBX ; punem bitii pe pozitia 10-11  EAX=0000 0000 0000 0000 0000 1010 0110 0000b = 00000A60h

        ;12-15 = 1
        mov EBX,0 ;initializam EBX cu 0 pentru a facilita interactiunea cu EAX
        mov BX,1111000000000000b ; modificam ultimii 4 biti ai registrului BX facand-ui 1
        or EAX,EBX ; astfel bitii 12-15 din registrul EAX vor fi facuti 1, ceilalti nefiind afectati EAX=1111 1010 0110 0000b=FA60h

        ;16-31 = 0-15 tot ai lui B
        mov EBX,EAX ;punem in EBX, valoarea lui EAX
        shl EBX,16  ;shiftam la stanga cu 16 pozitii aliniind astfel bitii ce trebuie modificati EBX=1111 1010 0110 0000 0000 0000 0000 0000b=FA600000h
        or EAX,EBX  ;punem bitii din EBX(cel obtinut prin shiftarea celor din EAX ) astfel bitii 16-31 vor fi egali 0-15
        mov [b],EAX ;[b]=1111 1010 0110 0000 1111 1010 0110 0000=FA60FA60h

```

```
bits 32 ; assembling for the 32 bits architecture
global start
extern exit             ; tell nasm that exit exists even if we won't be defining it
import exit msvcrt.dll    ; exit is a function that ends the calling process. It is defined in msvcrt.dll
segment data use32 class=data
   a dw 1010011011101101b ;=A6ED
   b resw 2
;0-3 = 0
;4-7 = 8-11 a
;8-9,10-11 = 0-1 a inversati
;12-15 = 1
;[b]=1111 1010 0110 0000 1111 1010 0110 0000=FA60FA60h
; our code starts here
segment code use32 class=code
   start:
      ;0-3 = 0
      ;initializam EAX(unde vom stoca si vom lucra cu variabila [b]) astfel indeplinind conditia ca primii 4 biti sa fie 0
EAX=00000000h
      mov EAX,0

      ;4-7 = 8-11 a
      mov EBX,0 ;initializam EBX cu 0 pentru a facilita interactiunea cu EAX
      mov BX,[a] ; folosim BX ca o copie pentru [a] BX=[a]
      and BX,0000111100000000b ; izolam biti 8-11 BX=0000 0110 0000 0000b=0600h
      shr BX,4 ;rotim la dreapta cu 4 pozitii astfel incat sa aliniem continutul bitilor
      or EAX,EBX

      ;8-9,10-11 = 0-1 a , inversati
      mov EBX,0 ;initializam EBX cu 0 pentru a facilita interactiunea cu EAX
      mov BX,[a] ; folosim BX ca o copie pentru [a] BX=[a]
      neg BX ; negam toti biti din BX
      sub BX,1 ; neg ne returneaza complementul fata de 2, noi avem de nevoie de cel fata de 1
      and BX,0000000000000011b ; izolam biti 0-1 BX=0000000000000010b=0002h astfel avem pe pozitie ultimii 2 biti, inversati
      shl BX,8 ;shiftam la stanga cu 8 pozitii, astfel aliniind biti cu 8-9
      or EAX,EBX ; punem bitii pe pozitia 8-9    EAX=0000 0000 0000 0000 0000 0010 0110 0000b = 00000260h
      shl BX,2 ;shiftam la stanga cu inca 2 pozitii astfel sunt aliniati cu pozitiile 10-11
      or EAX,EBX ; punem bitii pe pozitia 10-11  EAX=0000 0000 0000 0000 0000 1010 0110 0000b = 00000A60h

      ;12-15 = 1
      mov EBX,0 ;initializam EBX cu 0 pentru a facilita interactiunea cu EAX
      mov BX,1111000000000000b ; modificam ultimii 4 biti ai registrului BX facand-ui 1
      or EAX,EBX ; astfel bitii 12-15 din registrul EAX vor fi facuti 1, ceilalti nefiind afectati EAX=1111 1010 0110 0000b=FA60h

      ;16-31 = 0-15 tot ai lui B
      mov EBX,EAX ;punem in EBX, valoarea lui EAX
      shl EBX,16  ;shiftam la stanga cu 16 pozitii aliniind astfel bitii ce trebuie modificati EBX=1111 1010 0110 0000 0000 0000
0000 0000b=FA600000h
      or EAX,EBX  ;punem bitii din EBX(cel obtinut prin shiftarea celor din EAX ) astfel bitii 16-31 vor fi egali 0-15
      mov [b],EAX ;[b]=1111 1010 0110 0000 1111 1010 0110 0000=FA60FA60h

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

Pb32

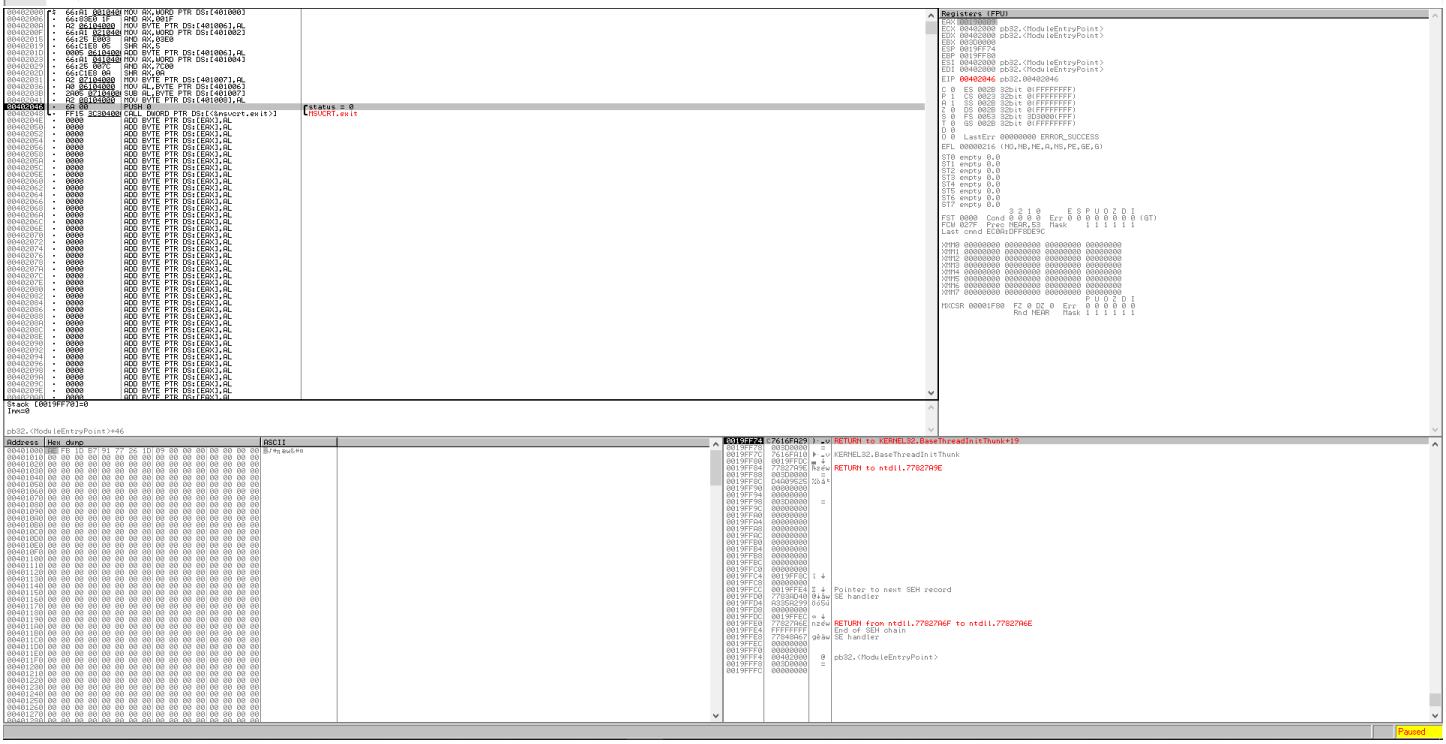
32. Se dau cuvintele A, B si C. Sa se obtina octetul D ca suma a numerelor reprezentate de:

biții de pe pozițiile 0-4 ai lui A

biții de pe pozițiile 5-9 ai lui B

Octetul E este numarul reprezentat de bitii 10-14 ai lui C. Sa se obtina octetul F ca rezultatul scaderii D-E.

```
segment data use32 class=data
    a dw 1111011101011110b
    b dw 1011011100011101b
    c dw 0111011110010001b
    d resb 1
    e resb 1
    f resb 1

;D=01110b+11000b=100110b
;E=11101b
;F=D-E=1001b
; our code starts here
segment code use32 class=code
    start:
        ;formam in AX nr format cu biti 0-4 din [a] apoi il adaugam la [d]
        mov AX,[a]
        and AX,0000000000011111b ;izolam bitii 0-4 ai nr [a] AX=0000 0000 0000 1110b=000Dh
        mov [d],AL ;[d]=AX=01110b=0Eh

        ;formam in AX nr format cu biti 5-9 din [b] apoi il adaugam la [d]
        mov AX,[b] ;AX=[b]
        and AX,0000001111100000b ;izolam biti 5-9 ai nr [b]   AX=0000001100000000b
        shr AX,5 ;shiftam spre dreapta astfel obtinand nr format din bitii 5-9 ai lui [b]
        add [d],AL ;adaugam valoarealui AL in variabila [d] astfel obtinand valoarea dorita  [d]=01110+11000=100110b=26h

        ;formam in AX nr format cu biti 10-14 ai lui [c] acest nr reprezentand f-ul
        mov AX,[c] ; AX=[c]
        and AX,0111110000000000b ;izolam biti 10-14 ai nr [c] AX=0111010000000000b
        shr AX,10;shiftam spre dreapta astfel obtinand nr format din bitii 10-14 ai lui [c]
        mov [e],AL ;obtinem in [e] valoarea dorita   [e]=11101b=1Dh

        ;realizam scaderea dintre cele doua variabile anterior construite
        mov AL,[d] ;AL=[d]
        sub AL,[e] ;AL=[d]-[e]

        mov [f],AL ; AL=[f]=[d]-[e]=1001b=9h

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



```
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             ; tell nasm that exit exists even if we won't be defining it
import exit msvcrt.dll    ; exit is a function that ends the calling process. It is defined in msvcrt.dll
                        ; msvcrt.dll contains exit, printf and all the other important C-runtime specific functions

; our data is declared here (the variables needed by our program)
segment data use32 class=data
   a dw 1111101110101110b
   b dw 1011011100011101b
   c dw 0111011110010001b
   d resb 1
   e resb 1
   f resb 1

;D=01110b+11000b=100110b
;E=11101b
;F=D-E=1001b
; our code starts here
segment code use32 class=code
   start:
      ;formam in AX nr format cu biti 0-4 din [a] apoi il adaugam la [d]
      mov AX,[a]
      and AX,0000000000011111b ;izolam bitii 0-4 ai nr [a] AX=0000 0000 0000 1110b=000Dh
      mov [d],AL ;[d]=AX=01110b=0Eh

      ;formam in AX nr format cu biti 5-9 din [b] apoi il adaugam la [d]
      mov AX,[b] ;AX=[b]
      and AX,0000001111100000b ;izolam biti 5-9 ai nr [b]   AX=00000001100000000b
      shr AX,5 ;shiftam spre dreapta astfel obtinand nr format din bitii 5-9 ai lui [b]
      add [d],AL ;adaugam valoarealui AL in variabila [d] astfel obtinand valoarea dorita  [d]=01110+11000=100110b=26h

      ;formam in AX nr format cu biti 10-14 ai lui [c] acest nr reprezentand f-ul
      mov AX,[c] ; AX=[c]
      and AX,0111110000000000b ;izolam biti 10-14 ai nr [c] AX=0111010000000000b
      shr AX,10;shiftam spre dreapta astfel obtinand nr format din bitii 10-14 ai lui [c]
      mov [e],AL ;obtinem in [e] valoarea dorita   [e]=11101b=1Dh

      ;realizam scaderea dintre cele doua variabile anterior construite
      mov AL,[d] ;AL=[d]
      sub AL,[e] ;AL=[d]-[e]

      mov [f],AL ; AL=[f]=[d]-[e]=1001b=9h

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