
10. Sa se inlocuiasca bitii 0-3 ai octetului B cu bitii 8-11 ai cuvintului 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 dreapta cu 8 pozitii continutul lui
a, astfel aliniam 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
```


18. Se da un cuvânt 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.

```

11 ; our data is declared here (the variables needed by our program)
12 segment data use32 class=data
13     ;FEDCBA9876543210
14     a dw 10100110111010101b ;:=A6ED
15     b resw 2
16
17 ;0-3 = 0
18 ;4-7 = 8-11 a
19 ;8-9,10-11 = 0-1 a inversati
20 ;12-15 = 1
21 ;[b]=1111 1010 0110 0000 1111 1010 0110 0000=FA60FA60h
22 ; our code starts here
23 segment code use32 class=code
24     start:
25     ;0-3 = 0
26     ;initializam EAX(unde vom stoca si vom lucra cu variabila [b]) astfel indeplinind conditia ca primii 4 biti sa fie 0 EAX=00000000h
27     mov EAX,0
28
29     ;4-7 = 8-11 a
30     mov EBX,0 ;initializam EBX cu 0 pentru a facilita interactiunea cu EAX
31     mov BX,[a] ; folosim BX ca o copie pentru [a] BX=[a]
32     and BX,000011100000000b ; izolam biti 8-11 BX=0000 0110 0000 0000b=0600h
33     shr BX,4 ;rotim la dreapta cu 4 pozitii astfel incat sa aliniam continutul bitilor
34     or EAX,EBX
35
36     ;8-9,10-11 = 0-1 a , inversati
37     mov EBX,0 ;initializam EBX cu 0 pentru a facilita interactiunea cu EAX
38     mov BX,[a] ; folosim BX ca o copie pentru [a] BX=[a]
39     neg BX ; negam toti biti din BX
40     sub BX,1 ; neg ne returneaza complementul fata de 2, noi avem de nevoie de cel fata de 1
41     and BX,0000000000000011b ; izolam biti 0-1 BX=0000000000000010b=0002h astfel avem pe pozitie ultimii 2 biti, inversati
42     shl BX,8 ;shiftam la stanga cu 8 pozitii, astfel aliniind biti cu 8-9
43     or EAX,EBX ; punem bitii pe pozitia 8-9 EAX=0000 0000 0000 0000 0000 0010 0110 0000b = 00000260h
44     shl BX,2 ;shiftam la stanga cu inca 2 pozitii astfel sunt aliniasi cu pozitiile 10-11
45     or EAX,EBX ; punem bitii pe pozitia 10-11 EAX=0000 0000 0000 0000 0000 1010 0110 0000b = 00000A60h
46
47     ;12-15 = 1
48     mov EBX,0 ;initializam EBX cu 0 pentru a facilita interactiunea cu EAX
49     mov BX,1111000000000000b ; modificam ultimii 4 biti ai registrului BX facand-ui 1
50     or EAX,EBX ; astfel bitii 12-15 din registrul EAX vor fi facuti 1, ceilalti nefiind afectati EAX=1111 1010 0110 0000b=FA60h
51
52     ;16-31 = 0-15 tot ai lui B
53     mov EBX,EAX ;punem in EBX, valoarea lui EAX
54     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
55     or EAX,EBX ;punem bitii din EBX(cei obtinuti prin shiftarea celor din EAX ) astfel bitii 16-31 vor fi egali 0-15
56     mov [b],EAX ; [b]=1111 1010 0110 0000 1111 1010 0110 0000=FA60FA60h
57
58

```

Address	Hex dump	Disassembly	Registers (FPU)
00401000	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401001	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401002	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401003	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401004	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401005	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401006	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401007	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401008	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401009	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040100A	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040100B	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040100C	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040100D	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040100E	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040100F	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401010	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401011	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401012	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401013	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401014	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401015	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401016	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401017	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401018	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401019	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040101A	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040101B	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040101C	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040101D	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040101E	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040101F	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401020	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401021	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401022	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401023	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401024	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401025	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401026	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401027	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401028	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401029	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040102A	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040102B	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040102C	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040102D	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040102E	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040102F	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401030	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401031	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401032	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401033	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401034	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401035	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401036	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401037	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401038	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401039	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040103A	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040103B	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040103C	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040103D	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040103E	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040103F	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401040	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401041	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401042	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401043	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401044	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401045	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401046	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401047	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401048	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401049	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040104A	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040104B	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040104C	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040104D	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040104E	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040104F	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401050	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401051	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401052	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401053	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401054	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401055	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401056	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401057	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401058	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401059	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040105A	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040105B	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040105C	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040105D	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040105E	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
0040105F	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401060	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401061	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401062	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401063	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401064	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401065	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401066	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401067	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000
00401068	41 55 8B 00 00 00 00 00 00 00 00 00 00 00 00 00	CALL EBX,0	EAX=00000000

```

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(cei obtinuti 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

```

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

biți de pe pozițiile 0-4 ai lui A

biți de pe pozițiile 5-9 ai lui B

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

```

12 segment data use32 class=data
13     a dw 111110110101110b
14     b dw 101101100011101b
15     c dw 011101110010001b
16     d resb 1
17     e resb 1
18     f resb 1
19
20 ;D=01110b+11000b=100110b
21 ;E=11101b
22 ;F=D-E=1001b
23 ; our code starts here
24 segment code use32 class=code
25     start:
26         ;formam in AX nr format cu biti 0-4 din [a] apoi il adaugam la [d]
27         mov AX,[a]
28         and AX,0000000000001111b ;izolam biții 0-4 ai nr [a] AX=0000 0000 0000 1110b=000Dh
29         mov [d],AL ;[d]=AX=01110b=0Eh
30
31         ;formam in AX nr format cu biti 5-9 din [b] apoi il adaugam la [d]
32         mov AX,[b] ;AX=[b]
33         and AX,000000111100000b ;izolam biții 5-9 ai nr [b] AX=00000001100000000b
34         shr AX,5 ;shiftam spre dreapta astfel obtinand nr format din biții 5-9 ai lui [b]
35         add [d],AL ;adaugam valoarea lui AL in variabila [d] astfel obtinand valoarea dorita [d]=01110+11000=100110b=26h
36
37         ;formam in AX nr format cu biti 10-14 ai lui [c] acest nr reprezentand f-ul
38         mov AX,[c] ;AX=[c]
39         and AX,011110000000000b ;izolam biții 10-14 ai nr [c] AX=0111010000000000b
40         shr AX,10 ;shiftam spre dreapta astfel obtinand nr format din biții 10-14 ai lui [c]
41         mov [e],AL ;obtinem in [e] valoarea dorita [e]=11101b=1Dh
42
43         ;realizam scaderea dintre cele doua variabile anterior construite
44         mov AL,[d] ;AL=[d]
45         sub AL,[e] ;AL=[d]-[e]
46
47         mov [f],AL ; AL=[f]=[d]-[e]=1001b=9h
48
49         push dword 0 ; push the parameter for exit onto the stack
50         call [exit] ; call exit to terminate the program
51

```

The screenshot displays a debugger interface with three main panels:

- Assembly Window:** Shows the assembly code being executed. The code is in x86 assembly, using the 32-bit instruction set. It includes comments in Romanian and English. The code calculates the sum of the first 5 bits of A and the next 5 bits of B, then subtracts the value of C from the result. The final result is stored in the AL register.
- Registers (CPU) Window:** Shows the state of the CPU registers. The EAX register contains the value 00000009, which is the decimal representation of the hexadecimal value 09h. Other registers like ECX, EDI, and ESI are shown with their respective values.
- Memory Window:** Shows the stack frame for the program. The return address at 00401000 points to 00401000. The stack is growing downwards, and the program is running at address 00401000.

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,0000000000001111b ;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=0000000110000000b

shr AX,5 ;shiftam spre dreapta astfel obtinand nr format din bitii 5-9 ai lui [b]

add [d],AL ;adaugam valoarea lui 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