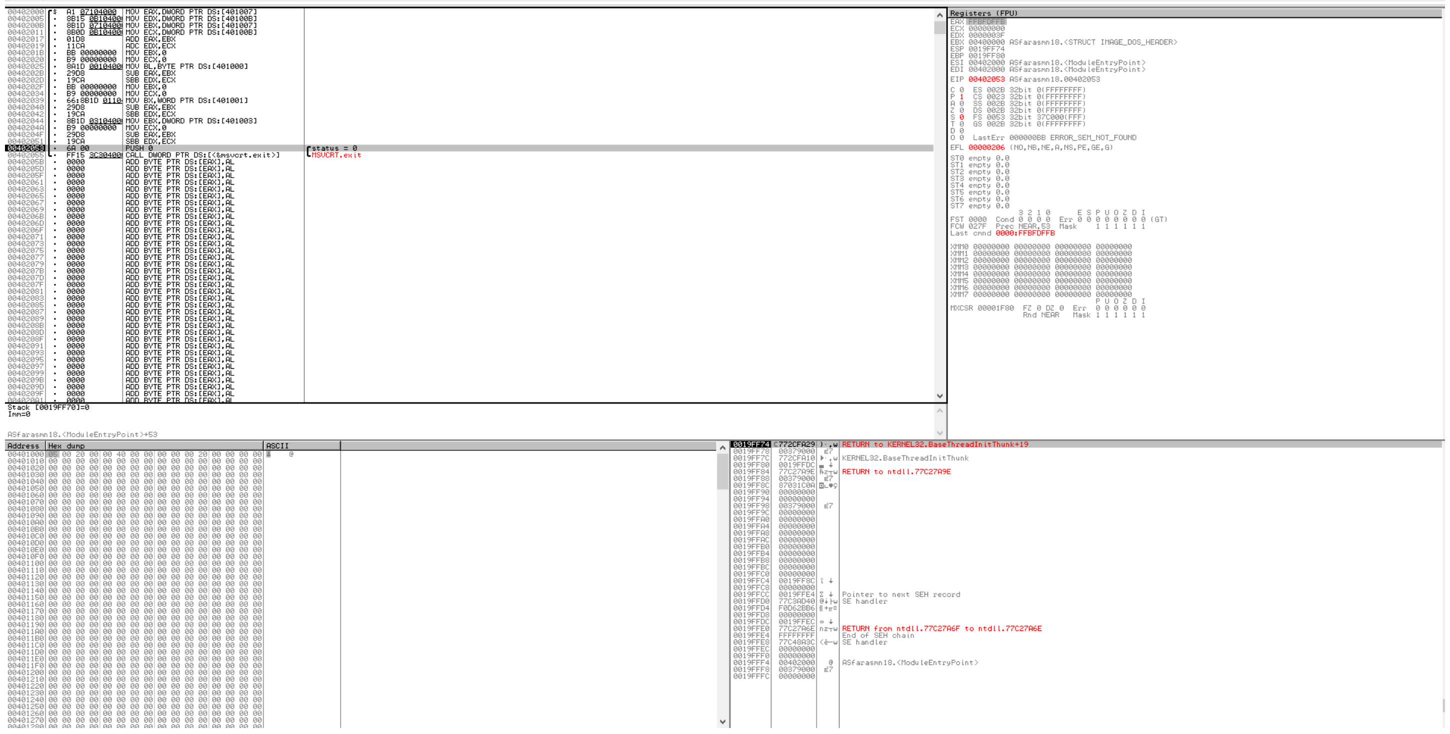
Adunari,scaderi fara semn Pb18 - (d+d)-a-b-c

```
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 db 5
    b dw 8192
    c dd 4194304
    d dq 137438953472

;(d+d)-a-b-c=(137438953472+137438953472)-5-8192-4194304=274877906944-5-8192-
4194304=274873704443=3FFFBFDFFB
; our code starts here
segment code use32 class=code
    start:
        ;EDX:EAX=d
        mov EAX,dword[d]   ;EAX=0h
        mov EDX,dword[d+4] ;EDX:EAX=137438953472=00000020 00000000h

        mov EBX,dword[d]   ;EBX=0h
        mov ECX,dword[d+4] ;ECX:EBX=137438953472=00000020 00000000h

        add EAX,EBX        ;EAX=EAX+EBX=0h
        adc EDX,ECX        ;EDX=EDX+ECX+CF=40h
                           ;EDX:EAX=d+d=00000040 00000000h
        ;convertim byte[a] la quadword
        mov EBX,0
        mov ECX,0
        mov BL,[a]    ;ECX:EBX=00000000 00000005h

        sub EAX,EBX   ;EAX=EAX-EBX
        sbb EDX,ECX   ;EDX=EDX-ECX-CF,  EDX:EAX=(d+d)-a=274877906939=0000003F FFFFFFFBh

        ;convertim word[b] la quadword
        mov EBX,0
        mov ECX,0
        mov BX,[b]    ;ECX:EBX=8192=00000000 00002000h

        sub EAX,EBX   ;EAX=EAX-EBX
        sbb EDX,ECX   ;EDX=EDX-ECX-CF,  EDX:EAX=(d+d)-a-b=274877898747=0000003F FFFFDFFBh

        ;convertim dword[c] la quadword
        mov EBX,[c]
        mov ECX,0     ;ECX:EBX=000000 00400000h

        sub EAX,EBX   ;EAX=EAX-EBX
        sbb EDX,ECX   ;EDX=EDX-ECX-CF,  EDX:EAX=(d+d)-a-b-c=274869510139=0000003F FFBFDFFBh

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

```
; our data is declared here (the variables needed by our program)
segment data use32 class=data
    a db 5
    b dw 8192
    c dd 4194304
    d dq 137438953472

;(d+d)-a-b-c=(137438953472+137438953472)-5-8192-4194304=274877906944-5-8192-4194304=274873704443=3FFFBFDFFB
; our code starts here
segment code use32 class=code
    start:
        ;EDX:EAX=d
        mov EAX,dword[d]    ;EAX=0h
        mov EDX,dword[d+4]  ;EDX:EAX=137438953472=00000020 00000000h


        mov EBX,dword[d]    ;EBX=0h
        mov ECX,dword[d+4]  ;ECX:EBX=137438953472=00000020 00000000h

        add EAX,EBX         ;EAX=EAX+EBX=0h
        adc EDX,ECX         ;EDX=EDX+ECX+CF=40h
                            ;EDX:EAX=d+d=00000040 00000000h

        ;convertim byte[a] la quadword
        mov EBX,0
        mov ECX,0
        mov BL,[a]     ;ECX:EBX=00000000 00000005h

        sub EAX,EBX    ;EAX=EAX-EBX
        sbb EDX,ECX    ;EDX=EDX-ECX-CF,   EDX:EAX=(d+d)-a=274877906939=0000003F FFFFFFFBh

        ;convertim word[b] la quadword
        mov EBX,0
        mov ECX,0
        mov BX,[b]     ;ECX:EBX=8192=00000000 00002000h

        sub EAX,EBX    ;EAX=EAX-EBX
        sbb EDX,ECX    ;EDX=EDX-ECX-CF,   EDX:EAX=(d+d)-a-b=274877898747=0000003F FFFFDFFBh

        ;convertim dword[c] la quadword
        mov EBX,[c]
        mov ECX,0      ;ECX:EBX=000000 00400000h

        sub EAX,EBX    ;EAX=EAX-EBX
        sbb EDX,ECX    ;EDX=EDX-ECX-CF,   EDX:EAX=(d+d)-a-b-c=274869510139=0000003F FFBFDFFBh
```



```
00402000 ┌$  A1 07104000   MOV EAX,DWORD PTR DS:[401007]
00402005 │.  8B15 0B10400  MOV EDX,DWORD PTR DS:[40100B]
0040200B │.  8B1D 0710400  MOV EBX,DWORD PTR DS:[401007]
00402011 │.  8B0D 0B10400  MOV ECX,DWORD PTR DS:[40100B]
00402017 │.  01D8          ADD EAX,EBX
00402019 │.  11CA          ADC EDX,ECX
0040201B │.  BB 00000000   MOV EBX,0
00402020 │.  B9 00000000   MOV ECX,0
00402025 │.  8A1D 0010400  MOV BL,BYTE PTR DS:[401000]
0040202B │.  29D8          SUB EAX,EBX
0040202D │.  19CA          SBB EDX,ECX
0040202F │.  BB 00000000   MOV EBX,0
00402034 │.  B9 00000000   MOV ECX,0
00402039 │.  66:8B1D 0110  MOV BX,WORD PTR DS:[401001]
00402040 │.  29D8          SUB EAX,EBX
00402042 │.  19CA          SBB EDX,ECX
00402044 │.  8B1D 0310400  MOV EBX,DWORD PTR DS:[401003]
0040204A │.  B9 00000000   MOV ECX,0
0040204F │.  29D8          SUB EAX,EBX
00402051 │.  19CA          SBB EDX,ECX
00402053 │.  6A 00         PUSH 0                                   ┌status = 0
00402055 └.  FF15 3C30400  CALL DWORD PTR DS:[<&msvcrt.exit>]       └MSVCRT.exit
0040205B  .  0000          ADD BYTE PTR DS:[EAX],AL
0040205D  .  0000          ADD BYTE PTR DS:[EAX],AL
0040205F  .  0000          ADD BYTE PTR DS:[EAX],AL
00402061  .  0000          ADD BYTE PTR DS:[EAX],AL
00402063  .  0000          ADD BYTE PTR DS:[EAX],AL
00402065  .  0000          ADD BYTE PTR DS:[EAX],AL
00402067  .  0000          ADD BYTE PTR DS:[EAX],AL
00402069  .  0000          ADD BYTE PTR DS:[EAX],AL
0040206B  .  0000          ADD BYTE PTR DS:[EAX],AL
0040206D  .  0000          ADD BYTE PTR DS:[EAX],AL
0040206F  .  0000          ADD BYTE PTR DS:[EAX],AL
00402071  .  0000          ADD BYTE PTR DS:[EAX],AL
00402073  .  0000          ADD BYTE PTR DS:[EAX],AL
00402075  .  0000          ADD BYTE PTR DS:[EAX],AL
00402077  .  0000          ADD BYTE PTR DS:[EAX],AL
00402079  .  0000          ADD BYTE PTR DS:[EAX],AL
0040207B  .  0000          ADD BYTE PTR DS:[EAX],AL
0040207D  .  0000          ADD BYTE PTR DS:[EAX],AL
0040207F  .  0000          ADD BYTE PTR DS:[EAX],AL
00402081  .  0000          ADD BYTE PTR DS:[EAX],AL
00402083  .  0000          ADD BYTE PTR DS:[EAX],AL
00402085  .  0000          ADD BYTE PTR DS:[EAX],AL
00402087  .  0000          ADD BYTE PTR DS:[EAX],AL
00402089  .  0000          ADD BYTE PTR DS:[EAX],AL
0040208B  .  0000          ADD BYTE PTR DS:[EAX],AL
0040208D  .  0000          ADD BYTE PTR DS:[EAX],AL
0040208F  .  0000          ADD BYTE PTR DS:[EAX],AL
00402091  .  0000          ADD BYTE PTR DS:[EAX],AL
00402093  .  0000          ADD BYTE PTR DS:[EAX],AL
00402095  .  0000          ADD BYTE PTR DS:[EAX],AL
00402097  .  0000          ADD BYTE PTR DS:[EAX],AL
00402099  .  0000          ADD BYTE PTR DS:[EAX],AL
0040209B  .  0000          ADD BYTE PTR DS:[EAX],AL
0040209D  .  0000          ADD BYTE PTR DS:[EAX],AL
0040209F  .  0000          ADD BYTE PTR DS:[EAX],AL
004020A1  .  0000          ADD BYTE PTR DS:[EAX],AL
Stack [0019FF70]=0
Imm=0

ASfarasmn18.<ModuleEntryPoint>+53
```

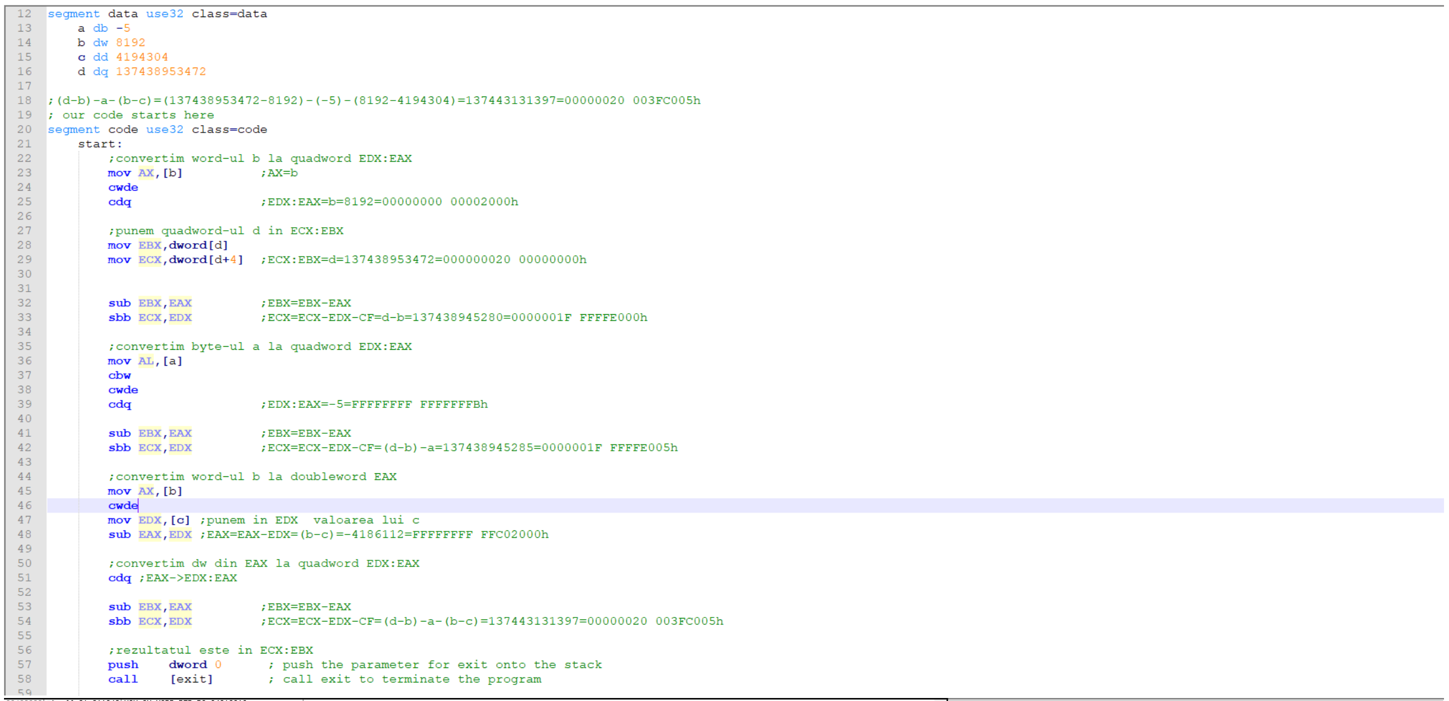
```
Registers (FPU)
EAX FFBFDFFB
ECX 00000000
EDX 0000003F
EBX 00400000 ASfarasmn18.<STRUCT IMAGE_DOS_HEADER>
ESP 0019FF74
EBP 0019FF80
ESI 00402000 ASfarasmn18.<ModuleEntryPoint>
EDI 00402000 ASfarasmn18.<ModuleEntryPoint>

EIP 00402053 ASfarasmn18.00402053

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 37C000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr 000000BB ERROR_SEM_NOT_FOUND

EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
Last cmnd 0000:FFBFDFFB

XMM0 00000000 00000000 00000000 00000000
XMM1 00000000 00000000 00000000 00000000
XMM2 00000000 00000000 00000000 00000000
XMM3 00000000 00000000 00000000 00000000
XMM4 00000000 00000000 00000000 00000000
XMM5 00000000 00000000 00000000 00000000
XMM6 00000000 00000000 00000000 00000000
XMM7 00000000 00000000 00000000 00000000
                                P U O Z D I
MXCSR 00001F80  FZ 0 DZ 0  Err  0 0 0 0 0 0
                Rnd NEAR   Mask 1 1 1 1 1 1
```

```
Address  Hex dump                                         ASCII
00401000 05 00 20 00 00 40 00 00 00 00 00 20 00 00 00 00 ♣     @
00401010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401250 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401280 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

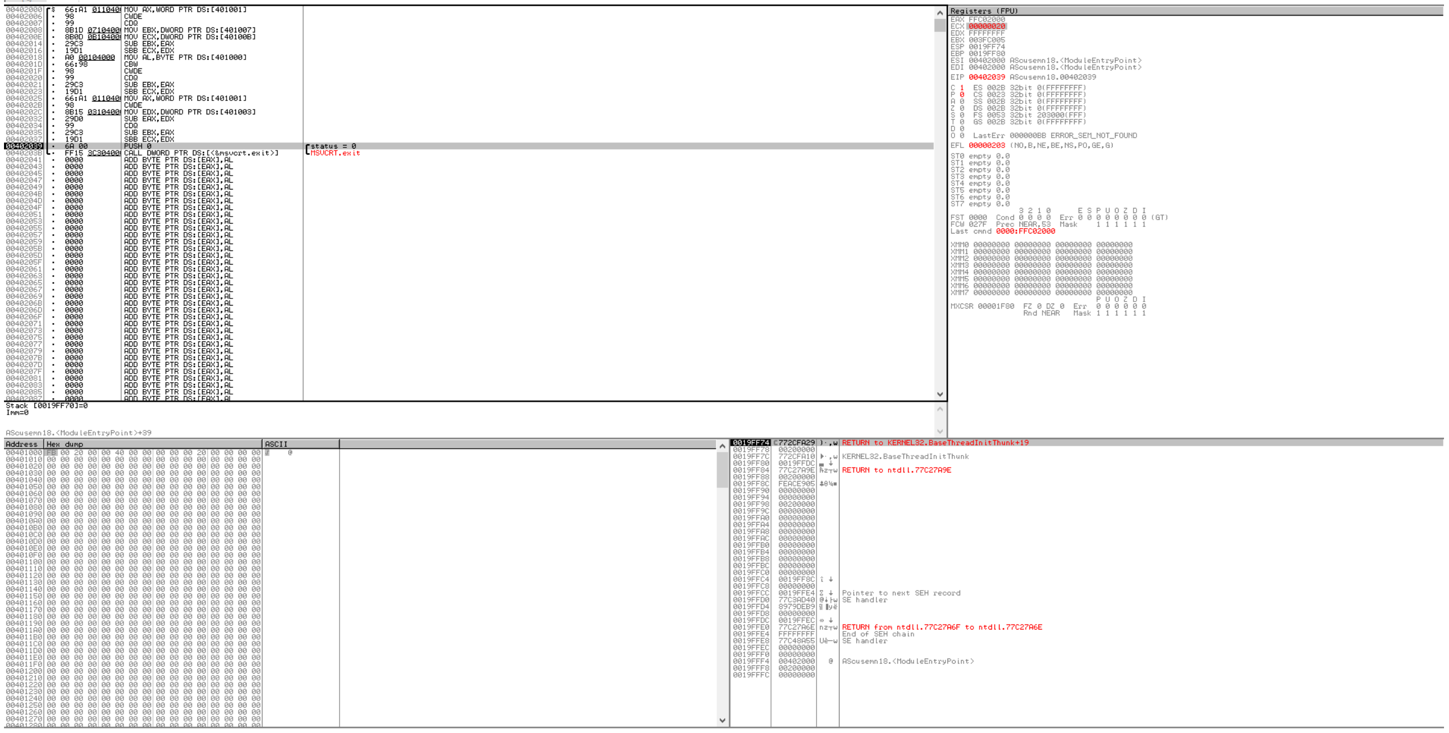
```
0019FF74  772CFA29  )·,w  RETURN to KERNEL32.BaseThreadInitThunk+19
0019FF78  00379000   ė7
0019FF7C  772CFA10  ▶·,w  KERNEL32.BaseThreadInitThunk
0019FF80  0019FFDC  ▬ ↓
0019FF84  77C27A9E  Ä≥┬w  RETURN to ntdll.77C27A9E
0019FF88  00379000   ė7
0019FF8C  87031C0A  ◙∟♥ç
0019FF90  00000000
0019FF94  00000000
0019FF98  00379000   ė7
0019FF9C  00000000
0019FFA0  00000000
0019FFA4  00000000
0019FFA8  00000000
0019FFAC  00000000
0019FFB0  00000000
0019FFB4  00000000
0019FFB8  00000000
0019FFBC  00000000
0019FFC0  00000000
0019FFC4  0019FF8C  î ↓
0019FFC8  00000000
0019FFCC  0019FFE4  Σ ↓   Pointer to next SEH record
0019FFD0  77C3AD40  @¡├w  SE handler
0019FFD4  F0D62BB6  ╢+╓≡
0019FFD8  00000000
0019FFDC  0019FFEC  ∞ ↓
0019FFE0  77C27A6E  n≥┬w  RETURN from ntdll.77C27A6F to ntdll.77C27A6E
0019FFE4  FFFFFFFF        End of SEH chain
0019FFE8  77C48A3C  <è─w  SE handler
0019FFEC  00000000
0019FFF0  00000000
0019FFF4  00402000    @   ASfarasmn18.<ModuleEntryPoint>
0019FFF8  00379000   ė7
0019FFFC  00000000
```

Adunari si scaderi cu semn Pb18 - `(d-b)-a-(b-c)`

```
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
                          ; msvcrt.dll contains exit, printf and all the other important C

; our data is declared here (the variables needed by our program)
segment data use32 class=data
    a db -5
    b dw 8192
    c dd 4194304
    d dq 137438953472

;(d-b)-a-(b-c)=(137438953472-8192)-(-5)-(8192-4194304)=137443131397=00000020 003FC005h
; our code starts here
segment code use32 class=code
    start:
        ;convertim word-ul b la quadword EDX:EAX
        mov AX,[b]          ;AX=b
        cwde
        cdq                 ;EDX:EAX=b=8192=00000000 00002000h

        ;punem quadword-ul d in ECX:EBX
        mov EBX,dword[d]
        mov ECX,dword[d+4]  ;ECX:EBX=d=137438953472=000000020 00000000h

        sub EBX,EAX         ;EBX=EBX-EAX
        sbb ECX,EDX         ;ECX=ECX-EDX-CF=d-b=137438945280=0000001F FFFFE000h

        ;convertim byte-ul a la quadword EDX:EAX
        mov AL,[a]
        cbw
        cwde
        cdq                 ;EDX:EAX=-5=FFFFFFFF FFFFFFFBh

        sub EBX,EAX         ;EBX=EBX-EAX
        sbb ECX,EDX         ;ECX=ECX-EDX-CF=(d-b)-a=137438945285=0000001F FFFFE005h

        ;convertim word-ul b la doubleword EAX
        mov AX,[b]
        cwde
        mov EDX,[c] ;punem in EDX  valoarea lui c
        sub EAX,EDX ;EAX=EAX-EDX=(b-c)=-4186112=FFFFFFFF FFC02000h

        ;convertim dw din EAX la quadword EDX:EAX
        cdq ;EAX->EDX:EAX

        sub EBX,EAX         ;EBX=EBX-EAX
        sbb ECX,EDX         ;ECX=ECX-EDX-CF=(d-b)-a-(b-c)=137443131397=00000020 003FC005h
        ;rezultatul este in ECX:EBX
        push    dword 0      ; push the parameter for exit onto the stack
        call    [exit]       ; call exit to terminate the program
```



```
segment data use32 class=data
    a db -5
    b dw 8192
    c dd 4194304
    d dq 137438953472

;(d-b)-a-(b-c)=(137438953472-8192)-(-5)-(8192-4194304)=137443131397=00000020 003FC005h
; our code starts here
segment code use32 class=code
    start:
        ;convertim word-ul b la quadword EDX:EAX
        mov AX,[b]          ;AX=b
        cwde
        cdq                 ;EDX:EAX=b=8192=00000000 00002000h

        ;punem quadword-ul d in ECX:EBX
        mov EBX,dword[d]
        mov ECX,dword[d+4]  ;ECX:EBX=d=137438953472=000000020 00000000h


        sub EBX,EAX         ;EBX=EBX-EAX
        sbb ECX,EDX         ;ECX=ECX-EDX-CF=d-b=137438945280=0000001F FFFFE000h

        ;convertim byte-ul a la quadword EDX:EAX
        mov AL,[a]
        cbw
        cwde
        cdq                 ;EDX:EAX=-5=FFFFFFFF FFFFFFFBh

        sub EBX,EAX         ;EBX=EBX-EAX
        sbb ECX,EDX         ;ECX=ECX-EDX-CF=(d-b)-a=137438945285=0000001F FFFFE005h

        ;convertim word-ul b la doubleword EAX
        mov AX,[b]
        cwde
        mov EDX,[c] ;punem in EDX  valoarea lui c
        sub EAX,EDX ;EAX=EAX-EDX=(b-c)=-4186112=FFFFFFFF FFC02000h

        ;convertim dw din EAX la quadword EDX:EAX
        cdq ;EAX->EDX:EAX

        sub EBX,EAX         ;EBX=EBX-EAX
        sbb ECX,EDX         ;ECX=ECX-EDX-CF=(d-b)-a-(b-c)=137443131397=00000020 003FC005h

        ;rezultatul este in ECX:EBX
        push    dword 0      ; push the parameter for exit onto the stack
        call    [exit]       ; call exit to terminate the program
```



```
00402000 ┌$  66:A1 0110400 MOV AX,WORD PTR DS:[401001]
00402006 │.  98            CWDE
00402007 │.  99            CDQ
00402008 │.  8B1D 0710400  MOV EBX,DWORD PTR DS:[401007]
0040200E │.  8B0D 0B10400  MOV ECX,DWORD PTR DS:[40100B]
00402014 │.  29C3          SUB EBX,EAX
00402016 │.  19D1          SBB ECX,EDX
00402018 │.  A0 00104000   MOV AL,BYTE PTR DS:[401000]
0040201D │.  66:98         CBW
0040201F │.  98            CWDE
00402020 │.  99            CDQ
00402021 │.  29C3          SUB EBX,EAX
00402023 │.  19D1          SBB ECX,EDX
00402025 │.  66:A1 0110400 MOV AX,WORD PTR DS:[401001]
0040202B │.  98            CWDE
0040202C │.  8B15 0310400  MOV EDX,DWORD PTR DS:[401003]
00402032 │.  29D0          SUB EAX,EDX
00402034 │.  99            CDQ
00402035 │.  29C3          SUB EBX,EAX
00402037 │.  19D1          SBB ECX,EDX
00402039 │.  6A 00         PUSH 0                                        ┌status = 0
0040203B └.  FF15 3C30400  CALL DWORD PTR DS:[<&msvcrt.exit>]            └MSVCRT.exit
00402041  .  0000          ADD BYTE PTR DS:[EAX],AL
00402043  .  0000          ADD BYTE PTR DS:[EAX],AL
00402045  .  0000          ADD BYTE PTR DS:[EAX],AL
00402047  .  0000          ADD BYTE PTR DS:[EAX],AL
00402049  .  0000          ADD BYTE PTR DS:[EAX],AL
0040204B  .  0000          ADD BYTE PTR DS:[EAX],AL
0040204D  .  0000          ADD BYTE PTR DS:[EAX],AL
0040204F  .  0000          ADD BYTE PTR DS:[EAX],AL
00402051  .  0000          ADD BYTE PTR DS:[EAX],AL
00402053  .  0000          ADD BYTE PTR DS:[EAX],AL
00402055  .  0000          ADD BYTE PTR DS:[EAX],AL
00402057  .  0000          ADD BYTE PTR DS:[EAX],AL
00402059  .  0000          ADD BYTE PTR DS:[EAX],AL
0040205B  .  0000          ADD BYTE PTR DS:[EAX],AL
0040205D  .  0000          ADD BYTE PTR DS:[EAX],AL
0040205F  .  0000          ADD BYTE PTR DS:[EAX],AL
00402061  .  0000          ADD BYTE PTR DS:[EAX],AL
00402063  .  0000          ADD BYTE PTR DS:[EAX],AL
00402065  .  0000          ADD BYTE PTR DS:[EAX],AL
00402067  .  0000          ADD BYTE PTR DS:[EAX],AL
00402069  .  0000          ADD BYTE PTR DS:[EAX],AL
0040206B  .  0000          ADD BYTE PTR DS:[EAX],AL
0040206D  .  0000          ADD BYTE PTR DS:[EAX],AL
0040206F  .  0000          ADD BYTE PTR DS:[EAX],AL
00402071  .  0000          ADD BYTE PTR DS:[EAX],AL
00402073  .  0000          ADD BYTE PTR DS:[EAX],AL
00402075  .  0000          ADD BYTE PTR DS:[EAX],AL
00402077  .  0000          ADD BYTE PTR DS:[EAX],AL
00402079  .  0000          ADD BYTE PTR DS:[EAX],AL
0040207B  .  0000          ADD BYTE PTR DS:[EAX],AL
0040207D  .  0000          ADD BYTE PTR DS:[EAX],AL
0040207F  .  0000          ADD BYTE PTR DS:[EAX],AL
00402081  .  0000          ADD BYTE PTR DS:[EAX],AL
00402083  .  0000          ADD BYTE PTR DS:[EAX],AL
00402085  .  0000          ADD BYTE PTR DS:[EAX],AL
00402087  .  0000          ADD BYTE PTR DS:[EAX],AL
Stack [0019FF70]=0
Imm=0

AScusemn18.<ModuleEntryPoint>+39
```

```
Registers (FPU)
EAX FFC02000
ECX 00000020
EDX FFFFFFFF
EBX 003FC005
ESP 0019FF74
EBP 0019FF80
ESI 00402000 AScusemn18.<ModuleEntryPoint>
EDI 00402000 AScusemn18.<ModuleEntryPoint>

EIP 00402039 AScusemn18.00402039

C 1  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 203000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr 000000BB ERROR_SEM_NOT_FOUND

EFL 00000203 (NO,B,NE,BE,NS,PO,GE,G)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
Last cmnd 0000:FFC02000

XMM0 00000000 00000000 00000000 00000000
XMM1 00000000 00000000 00000000 00000000
XMM2 00000000 00000000 00000000 00000000
XMM3 00000000 00000000 00000000 00000000
XMM4 00000000 00000000 00000000 00000000
XMM5 00000000 00000000 00000000 00000000
XMM6 00000000 00000000 00000000 00000000
XMM7 00000000 00000000 00000000 00000000
                                P U O Z D I
MXCSR 00001F80  FZ 0 DZ 0  Err  0 0 0 0 0 0
                Rnd NEAR   Mask 1 1 1 1 1 1
```

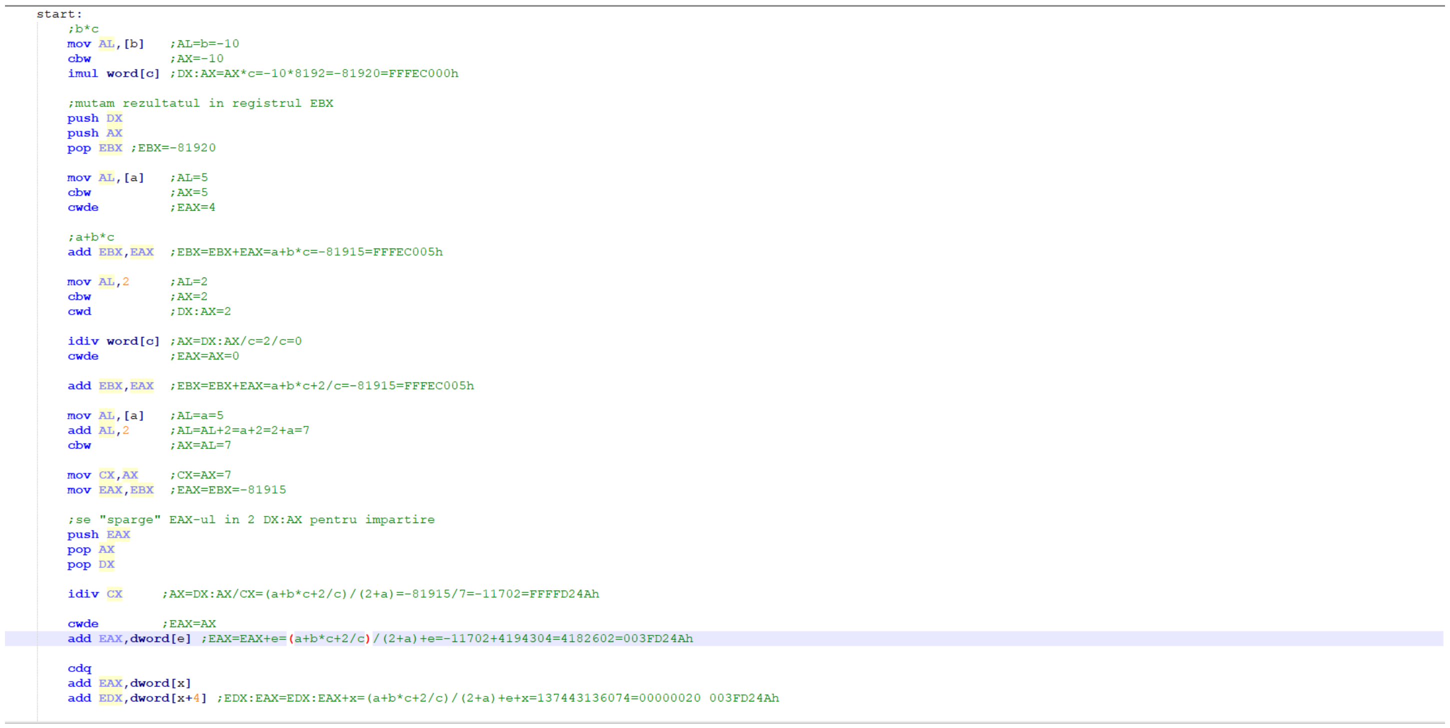
```
Address  Hex dump                                         ASCII
00401000 FB 00 20 00 00 40 00 00 00 00 00 20 00 00 00 00 √    @
00401010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401250 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
0019FF74  772CFA29  )·,w  RETURN to KERNEL32.BaseThreadInitThunk+19
0019FF78  00200000
0019FF7C  772CFA10  ▸·,w  KERNEL32.BaseThreadInitThunk
0019FF80  0019FFDC   ↓
0019FF84  77C27A9E        RETURN to ntdll.77C27A9E
0019FF88  00200000
0019FF8C  FEACE905
0019FF90  00000000
0019FF94  00000000
0019FF98  00200000
0019FF9C  00000000
0019FFA0  00000000
0019FFA4  00000000
0019FFA8  00000000
0019FFAC  00000000
0019FFB0  00000000
0019FFB4  00000000
0019FFB8  00000000
0019FFBC  00000000
0019FFC0  00000000
0019FFC4  0019FF8C   ↓
0019FFC8  00000000
0019FFCC  0019FFE4  Σ ↓   Pointer to next SEH record
0019FFD0  77C3AD40        SE handler
0019FFD4  8979DEB9
0019FFD8  00000000
0019FFDC  0019FFEC   ↓
0019FFE0  77C27A6E        RETURN from ntdll.77C27A6F to ntdll.77C27A6E
0019FFE4  FFFFFFFF        End of SEH chain
0019FFE8  77C48A55        SE handler
0019FFEC  00000000
0019FFF0  00000000
0019FFF4  00402000    @   AScusemn18.<ModuleEntryPoint>
0019FFF8  00200000
0019FFFC  00000000
```

Inmultiri si impartiri Pb.18 - (a+b*c+2/c)/(2+a)+e+x

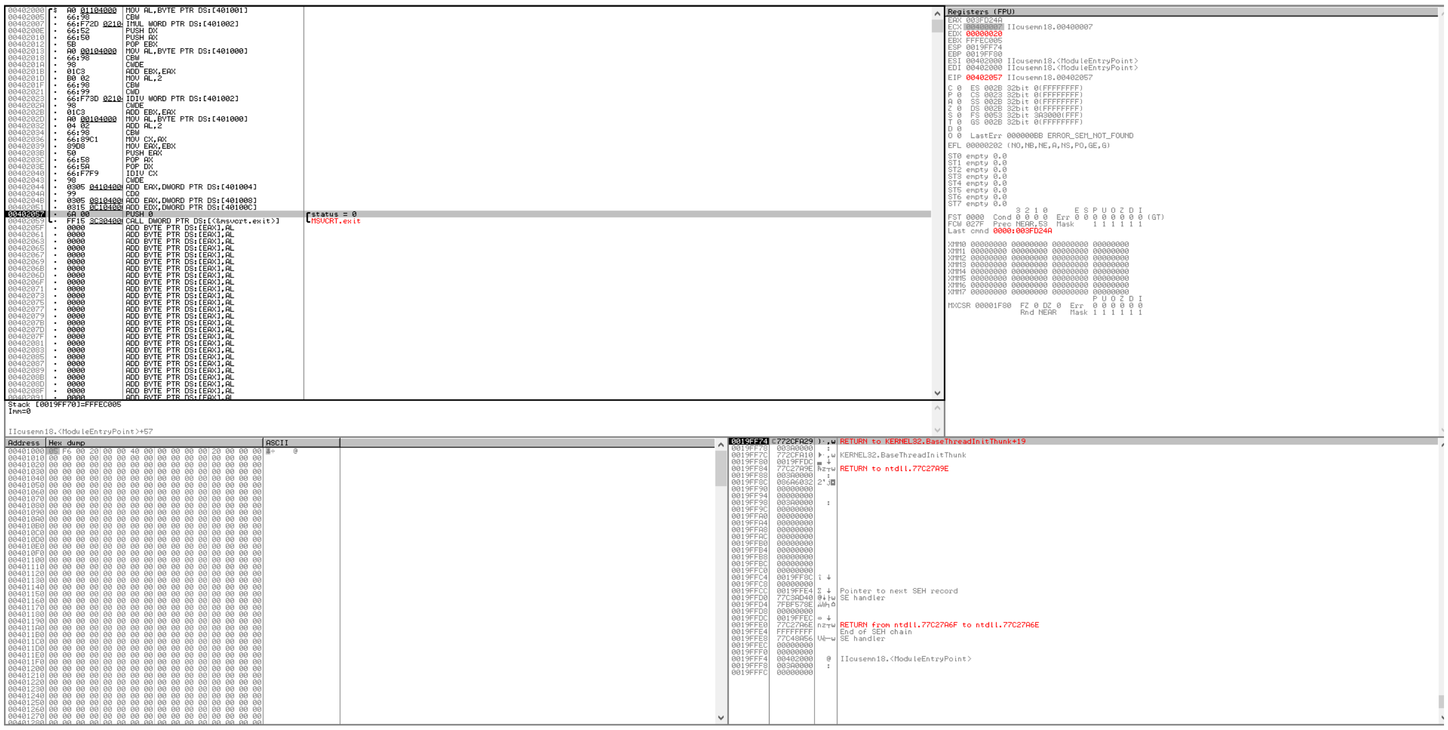
```
bits 32 ; assembling for the 32 bits architecture

; 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 contains exit, printf and all the other important C

; our data is declared here (the variables needed by our program)
segment data use32 class=data
    a db 5
    b db -10
    c dw 8192
    e dd 4194304
    x dq 137438953472

;(a+b*c+2/c)/(2+a)+e+x=(5+(-10)*8192+2/8192)/(2+5)+4194304+137438953472=137443136074=00000020
003FD24Ah
; our code starts here
segment code use32 class=code
    start:
        ;b*c
        mov AL,[b]    ;AL=b=-10
        cbw           ;AX=-10
        imul word[c] ;DX:AX=AX*c=-10*8192=-81920=FFFEC000h

        ;mutam rezultatul in registrul EBX
        push DX
        push AX
        pop EBX ;EBX=-81920

        mov AL,[a]    ;AL=5
        cbw           ;AX=5
        cwde          ;EAX=4

        ;a+b*c
        add EBX,EAX   ;EBX=EBX+EAX=a+b*c=-81915=FFFEC005h

        mov AL,2      ;AL=2
        cbw           ;AX=2
        cwd           ;DX:AX=2

        idiv word[c] ;AX=DX:AX/c=2/c=0
        cwde          ;EAX=AX=0

        add EBX,EAX   ;EBX=EBX+EAX=a+b*c+2/c=-81915=FFFEC005h

        mov AL,[a]    ;AL=a=5
        add AL,2      ;AL=AL+2=a+2=2+a=7
        cbw           ;AX=AL=7

        mov CX,AX     ;CX=AX=7
        mov EAX,EBX   ;EAX=EBX=-81915
```

```
;se "sparge" EAX-ul in 2 DX:AX pentru impartire

        push EAX

        pop AX

        pop DX


        idiv CX     ;AX=DX:AX/CX=(a+b*c+2/c)/(2+a)=-81915/7=-11702=FFFFD24Ah


        cwde        ;EAX=AX

        add EAX,dword[e] ;EAX=EAX+e=(a+b*c+2/c)/(2+a)+e=-11702+4194304=4182602=003FD24Ah

        cdq

        add EAX,dword[x]

        add EDX,dword[x+4] ;EDX:EAX=EDX:EAX+x=(a+b*c+2/c)/(2+a)+e+x=137443136074=00000020
003FD24Ah

        push    dword 0      ; push the parameter for exit onto the stack

        call    [exit]       ; call exit to terminate the program
```



```
start:
    ;b*c
    mov AL,[b]    ;AL=b=-10
    cbw           ;AX=-10
    imul word[c] ;DX:AX=AX*c=-10*8192=-81920=FFFEC000h

    ;mutam rezultatul in registrul EBX
    push DX
    push AX
    pop EBX ;EBX=-81920

    mov AL,[a]    ;AL=5
    cbw           ;AX=5
    cwde          ;EAX=4

    ;a+b*c
    add EBX,EAX   ;EBX=EBX+EAX=a+b*c=-81915=FFFEC005h

    mov AL,2      ;AL=2
    cbw           ;AX=2
    cwd           ;DX:AX=2

    idiv word[c] ;AX=DX:AX/c=2/c=0
    cwde          ;EAX=AX=0

    add EBX,EAX   ;EBX=EBX+EAX=a+b*c+2/c=-81915=FFFEC005h

    mov AL,[a]    ;AL=a=5
    add AL,2      ;AL=AL+2=a+2=2+a=7
    cbw           ;AX=AL=7

    mov CX,AX     ;CX=AX=7
    mov EAX,EBX   ;EAX=EBX=-81915

    ;se "sparge" EAX-ul in 2 DX:AX pentru impartire
    push EAX
    pop AX
    pop DX

    idiv CX      ;AX=DX:AX/CX=(a+b*c+2/c)/(2+a)=-81915/7=-11702=FFFFD24Ah

    cwde         ;EAX=AX
    add EAX,dword[e] ;EAX=EAX+e=(a+b*c+2/c)/(2+a)+e=-11702+4194304=4182602=003FD24Ah

    cdq
    add EAX,dword[x]
    add EDX,dword[x+4] ;EDX:EAX=EDX:EAX+x=(a+b*c+2/c)/(2+a)+e+x=137443136074=00000020 003FD24Ah
```

```
; our data is declared here (the variables needed by our program)
segment data use32 class=data
    a db 5
    b db -10
    c dw 8192
    e dd 4194304
    x dq 137438953472

;(a+b*c+2/c)/(2+a)+e+x=(5+(-10)*8192+2/8192)/(2+5)+4194304+137438953472=137443136074=00000020 003FD24Ah
; our code starts here
```



```
00402000  $ A0 01104000   MOV AL,BYTE PTR DS:[401001]
00402005  . 66:98         CBW
00402007  . 66:F72D 0210  IMUL WORD PTR DS:[401002]
0040200E  . 66:52         PUSH DX
00402010  . 66:50         PUSH AX
00402012  . 5B            POP EBX
00402013  . A0 00104000   MOV AL,BYTE PTR DS:[401000]
00402018  . 66:98         CBW
0040201A  . 98            CWDE
0040201B  . 01C3          ADD EBX,EAX
0040201D  . B0 02         MOV AL,2
0040201F  . 66:98         CBW
00402021  . 66:99         CWD
00402023  . 66:F73D 0210  IDIV WORD PTR DS:[401002]
0040202A  . 98            CWDE
0040202B  . 01C3          ADD EBX,EAX
0040202D  . A0 00104000   MOV AL,BYTE PTR DS:[401000]
00402032  . 04 02         ADD AL,2
00402034  . 66:98         CBW
00402036  . 66:89C1       MOV CX,AX
00402039  . 89D8          MOV EAX,EBX
0040203B  . 50            PUSH EAX
0040203C  . 66:58         POP AX
0040203E  . 66:5A         POP DX
00402040  . 66:F7F9       IDIV CX
00402043  . 98            CWDE
00402044  . 0305 0410400  ADD EAX,DWORD PTR DS:[401004]
0040204A  . 99            CDQ
0040204B  . 0305 0810400  ADD EAX,DWORD PTR DS:[401008]
00402051  . 0315 0C10400  ADD EDX,DWORD PTR DS:[40100C]
00402057  . 6A 00         PUSH 0                                  status = 0
00402059  . FF15 3C30400  CALL DWORD PTR DS:[<&msvcrt.exit>]      MSVCRT.exit
0040205F  . 0000          ADD BYTE PTR DS:[EAX],AL
00402061  . 0000          ADD BYTE PTR DS:[EAX],AL
00402063  . 0000          ADD BYTE PTR DS:[EAX],AL
00402065  . 0000          ADD BYTE PTR DS:[EAX],AL
00402067  . 0000          ADD BYTE PTR DS:[EAX],AL
00402069  . 0000          ADD BYTE PTR DS:[EAX],AL
0040206B  . 0000          ADD BYTE PTR DS:[EAX],AL
0040206D  . 0000          ADD BYTE PTR DS:[EAX],AL
0040206F  . 0000          ADD BYTE PTR DS:[EAX],AL
00402071  . 0000          ADD BYTE PTR DS:[EAX],AL
00402073  . 0000          ADD BYTE PTR DS:[EAX],AL
00402075  . 0000          ADD BYTE PTR DS:[EAX],AL
00402077  . 0000          ADD BYTE PTR DS:[EAX],AL
00402079  . 0000          ADD BYTE PTR DS:[EAX],AL
0040207B  . 0000          ADD BYTE PTR DS:[EAX],AL
0040207D  . 0000          ADD BYTE PTR DS:[EAX],AL
0040207F  . 0000          ADD BYTE PTR DS:[EAX],AL
00402081  . 0000          ADD BYTE PTR DS:[EAX],AL
00402083  . 0000          ADD BYTE PTR DS:[EAX],AL
00402085  . 0000          ADD BYTE PTR DS:[EAX],AL
00402087  . 0000          ADD BYTE PTR DS:[EAX],AL
00402089  . 0000          ADD BYTE PTR DS:[EAX],AL
0040208B  . 0000          ADD BYTE PTR DS:[EAX],AL
0040208D  . 0000          ADD BYTE PTR DS:[EAX],AL
0040208F  . 0000          ADD BYTE PTR DS:[EAX],AL
00402091  . 0000          ADD BYTE PTR DS:[EAX],AL

Stack [0019FF70]=FFFEC005
Imm=0

IIcusemn18.<ModuleEntryPoint>+57
```

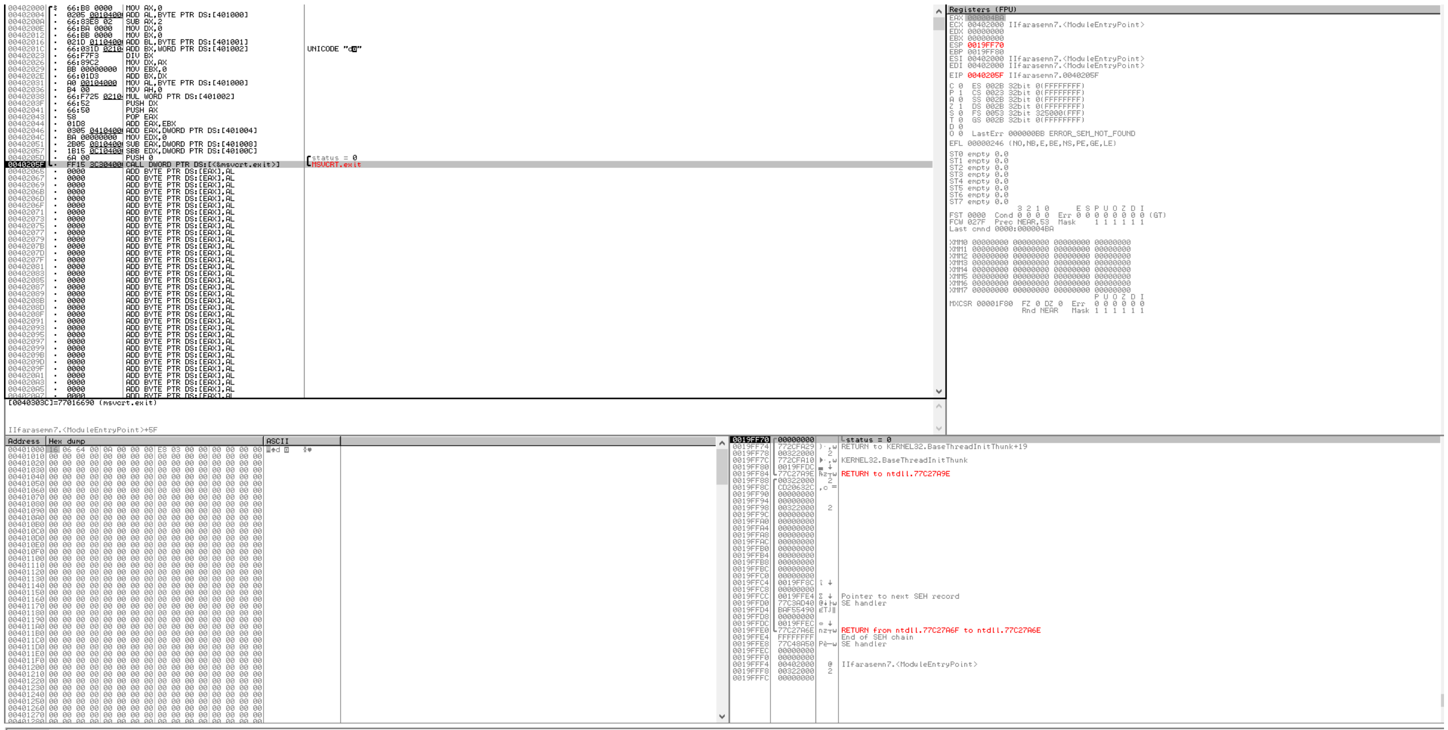
```
Registers (FPU)
EAX 003FD24A
ECX 00400007 IIcusemn18.00400007
EDX 00000020
EBX FFFEC005
ESP 0019FF74
EBP 0019FF80
ESI 00402000 IIcusemn18.<ModuleEntryPoint>
EDI 00402000 IIcusemn18.<ModuleEntryPoint>

EIP 00402057 IIcusemn18.00402057

C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 3A3000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr 000000BB ERROR_SEM_NOT_FOUND

EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
Last cmnd 0000:003FD24A

XMM0 00000000 00000000 00000000 00000000
XMM1 00000000 00000000 00000000 00000000
XMM2 00000000 00000000 00000000 00000000
XMM3 00000000 00000000 00000000 00000000
XMM4 00000000 00000000 00000000 00000000
XMM5 00000000 00000000 00000000 00000000
XMM6 00000000 00000000 00000000 00000000
XMM7 00000000 00000000 00000000 00000000
                                P U O Z D I
MXCSR 00001F80  FZ 0 DZ 0  Err  0 0 0 0 0 0
                Rnd NEAR   Mask 1 1 1 1 1 1
```

```
Address   Hex dump                                          ASCII
00401000  05 F6 00 20 00 00 40 00 00 00 00 00 20 00 00 00  ‡÷      @
00401010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401200  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401210  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401220  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401230  00 F6 00 00 00 00 40 00 00 00 00 00 20 00 00 00
00401240  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401250  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401260  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401280  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
0019FF74  772CFA29  ).,w  RETURN to KERNEL32.BaseThreadInitThunk+19
0019FF78  003A0000    :
0019FF7C  772CFA10  ▶.,w  KERNEL32.BaseThreadInitThunk
0019FF80  0019FFDC  ▬ ↓
0019FF84  77C27A9E  Ŗz┬w  RETURN to ntdll.77C27A9E
0019FF88  003A0000    :
0019FF8C  086A6032  2'j■
0019FF90  00000000
0019FF94  00000000
0019FF98  003A0000    :
0019FF9C  00000000
0019FFA0  00000000
0019FFA4  00000000
0019FFA8  00000000
0019FFAC  00000000
0019FFB0  00000000
0019FFB4  00000000
0019FFB8  00000000
0019FFBC  00000000
0019FFC0  00000000
0019FFC4  0019FF8C  î ↓
0019FFC8  00000000
0019FFCC  0019FFE4  Σ ↓   Pointer to next SEH record
0019FFD0  77C3AD40  @↓├w  SE handler
0019FFD4  7FBF578E  ÄW┐⌂
0019FFD8  00000000
0019FFDC  0019FFEC  ∞ ↓
0019FFE0  77C27A6E  nz┬w  RETURN from ntdll.77C27A6F to ntdll.77C27A6E
0019FFE4  FFFFFFFF        End of SEH chain
0019FFE8  77C48A56  Vè─w  SE handler
0019FFEC  00000000
0019FFF0  00000000
0019FFF4  00402000    @   IIcusemn18.<ModuleEntryPoint>
0019FFF8  003A0000    :
0019FFFC  00000000
```

II fara semn PB7 – (a-2)/(b+c)+a*c+e-x

```
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 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 db 22
    b db 6
    c dw 100
    e dd 10
    x dq 1000

;(a-2)/(b+c)+a*c+e-x=20/106+2200+10-1000=0+2200+10-1000=1210
; our code starts here
segment code use32 class=code
    start:
        ;convertim a la dw in perecea DX:AX pentru impartire
        mov AX,0
        add AL,[a]
        sub AX,2;AX=(a-2)
        mov DX,0

        mov BX,0
        add BL,[b]
        add BX,[c];BX=(b+c)
        div BX    ;AX=DX:AX/BX=(a-2)/(b+c)=0

        mov DX,AX ;DX=AX=0
        mov EBX,0
        add BX,DX ;BX=DX=AX=0

        mov AL,[a];AL=a=22=16h
        mov AH,0  ;AX=a=22=16h

        mul word[c];DX:AX=a*c=2200

        push DX
        push AX
        pop EAX    ;EAX=DX:AX=a*c=2200

        add EAX,EBX;EAX=(a-2)/(b+c)+a*c=2200
        add EAX,[e];EAX=(a-2)/(b+c)+a*c+e=2210

        mov EDX,0

        sub EAX,dword[x]   ;EAX=EAX-[x]
        sbb EDX,dword[x+4] ;EDX=EAX-[x]-CF=(a-2)/(b+c)+a*c+e-x=1210=4BAh

        ;rezultat in EDX:EAX
        push    dword 0      ; push the parameter for exit onto the stack
        call    [exit]       ; call exit to terminate the program
```

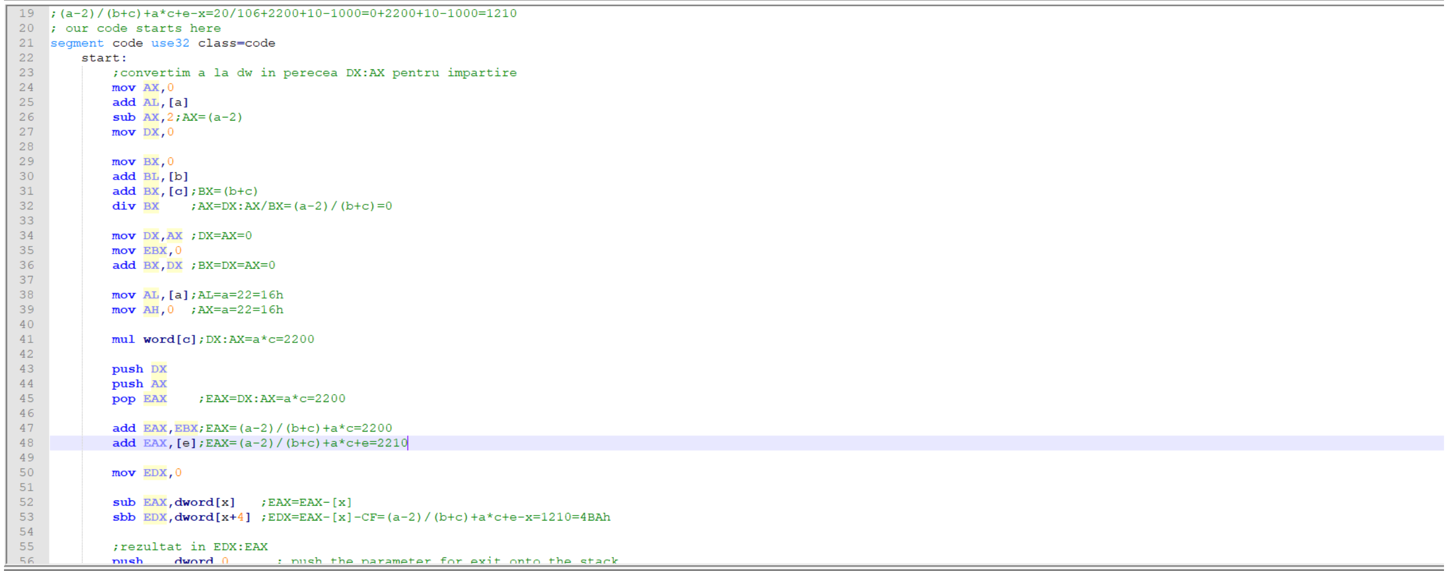


```
00402000 ┌$  66:B8 0000     MOV AX,0
00402004 │.  0205 00104000  ADD AL,BYTE PTR DS:[401000]
0040200A │.  66:83E8 02     SUB AX,2
0040200E │.  66:BA 0000     MOV DX,0
00402012 │.  66:BB 0000     MOV BX,0
00402016 │.  021D 01104000  ADD BL,BYTE PTR DS:[401001]
0040201C │.  66:031D 02104  ADD BX,WORD PTR DS:[401002]         UNICODE "d▣"
00402023 │.  66:F7F3        DIV BX
00402026 │.  66:89C2        MOV DX,AX
00402029 │.  BB 00000000    MOV EBX,0
0040202E │.  66:01D3        ADD BX,DX
00402031 │.  A0 00104000    MOV AL,BYTE PTR DS:[401000]
00402036 │.  B4 00          MOV AH,0
00402038 │.  66:F725 02104  MUL WORD PTR DS:[401002]
0040203F │.  66:52          PUSH DX
00402041 │.  66:50          PUSH AX
00402043 │.  58             POP EAX
00402044 │.  01D8           ADD EAX,EBX
00402046 │.  0305 04104000  ADD EAX,DWORD PTR DS:[401004]
0040204C │.  BA 00000000    MOV EDX,0
00402051 │.  2B05 08104000  SUB EAX,DWORD PTR DS:[401008]
00402057 │.  1B15 0C104000  SBB EDX,DWORD PTR DS:[40100C]
0040205D │.  6A 00          PUSH 0                              ┌status = 0
0040205F └.  FF15 3C304000  CALL DWORD PTR DS:[<&msvcrt.exit>]  └MSVCRT.exit
00402065  .  0000           ADD BYTE PTR DS:[EAX],AL
00402067  .  0000           ADD BYTE PTR DS:[EAX],AL
00402069  .  0000           ADD BYTE PTR DS:[EAX],AL
0040206B  .  0000           ADD BYTE PTR DS:[EAX],AL
0040206D  .  0000           ADD BYTE PTR DS:[EAX],AL
0040206F  .  0000           ADD BYTE PTR DS:[EAX],AL
00402071  .  0000           ADD BYTE PTR DS:[EAX],AL
00402073  .  0000           ADD BYTE PTR DS:[EAX],AL
00402075  .  0000           ADD BYTE PTR DS:[EAX],AL
00402077  .  0000           ADD BYTE PTR DS:[EAX],AL
00402079  .  0000           ADD BYTE PTR DS:[EAX],AL
0040207B  .  0000           ADD BYTE PTR DS:[EAX],AL
0040207D  .  0000           ADD BYTE PTR DS:[EAX],AL
0040207F  .  0000           ADD BYTE PTR DS:[EAX],AL
00402081  .  0000           ADD BYTE PTR DS:[EAX],AL
00402083  .  0000           ADD BYTE PTR DS:[EAX],AL
00402085  .  0000           ADD BYTE PTR DS:[EAX],AL
00402087  .  0000           ADD BYTE PTR DS:[EAX],AL
00402089  .  0000           ADD BYTE PTR DS:[EAX],AL
0040208B  .  0000           ADD BYTE PTR DS:[EAX],AL
0040208D  .  0000           ADD BYTE PTR DS:[EAX],AL
0040208F  .  0000           ADD BYTE PTR DS:[EAX],AL
00402091  .  0000           ADD BYTE PTR DS:[EAX],AL
00402093  .  0000           ADD BYTE PTR DS:[EAX],AL
00402095  .  0000           ADD BYTE PTR DS:[EAX],AL
00402097  .  0000           ADD BYTE PTR DS:[EAX],AL
00402099  .  0000           ADD BYTE PTR DS:[EAX],AL
0040209B  .  0000           ADD BYTE PTR DS:[EAX],AL
0040209D  .  0000           ADD BYTE PTR DS:[EAX],AL
0040209F  .  0000           ADD BYTE PTR DS:[EAX],AL
004020A1  .  0000           ADD BYTE PTR DS:[EAX],AL
004020A3  .  0000           ADD BYTE PTR DS:[EAX],AL
004020A5  .  0000           ADD BYTE PTR DS:[EAX],AL
004020A7  .  0000           ADD BYTE PTR DS:[EAX],AL
[0040303C]=77016690 (msvcrt.exit)

IIfarasemn7.<ModuleEntryPoint>+5F
```

```
Registers (FPU)
EAX 000004BA
ECX 00402000 IIfarasemn7.<ModuleEntryPoint>
EDX 00000000
EBX 00000000
ESP 0019FF70
EBP 0019FF80
ESI 00402000 IIfarasemn7.<ModuleEntryPoint>
EDI 00402000 IIfarasemn7.<ModuleEntryPoint>

EIP 0040205F IIfarasemn7.0040205F

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 325000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr 000000BB ERROR_SEM_NOT_FOUND

EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
Last cmnd 0000:000004BA

XMM0 00000000 00000000 00000000 00000000
XMM1 00000000 00000000 00000000 00000000
XMM2 00000000 00000000 00000000 00000000
XMM3 00000000 00000000 00000000 00000000
XMM4 00000000 00000000 00000000 00000000
XMM5 00000000 00000000 00000000 00000000
XMM6 00000000 00000000 00000000 00000000
XMM7 00000000 00000000 00000000 00000000
                                P U O Z D I
MXCSR 00001F80  FZ 0 DZ 0  Err  0 0 0 0 0 0
                Rnd NEAR   Mask 1 1 1 1 1 1
```

```
Address  Hex dump                                          ASCII
00401000 16 06 64 00 0A 00 00 00 E8 03 00 00 00 00 00 00  ▬♠d ◙    ↨♥
00401010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004010F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
004011F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401250 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401260 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401270 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00401280 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
0019FF70 ┌00000000         └status = 0
0019FF74 │772CFA29 )·,w    RETURN to KERNEL32.BaseThreadInitThunk+19
0019FF78 │00322000   2
0019FF7C │772CFA10 ▶·,w    KERNEL32.BaseThreadInitThunk
0019FF80 │0019FFDC ▬ ↓
0019FF84 └77C27A9E ÑzΤw    RETURN to ntdll.77C27A9E
0019FF88 ┌00322000   2
0019FF8C │CD20632C ,c =
0019FF90 │00000000
0019FF94 │00000000
0019FF98 │00322000   2
0019FF9C │00000000
0019FFA0 │00000000
0019FFA4 │00000000
0019FFA8 │00000000
0019FFAC │00000000
0019FFB0 │00000000
0019FFB4 │00000000
0019FFB8 │00000000
0019FFBC │00000000
0019FFC0 │00000000
0019FFC4 │0019FF8C î ↓
0019FFC8 │00000000
0019FFCC │0019FFE4 Σ ↓     Pointer to next SEH record
0019FFD0 │77C3AD40 @↓├w    SE handler
0019FFD4 │BAF55490 ÉTJ║
0019FFD8 │00000000
0019FFDC │0019FFEC ∞ ↓
0019FFE0 └77C27A6E nzΤw    RETURN from ntdll.77C27A6F to ntdll.77C27A6E
0019FFE4  FFFFFFFF         End of SEH chain
0019FFE8  77C48A50 Pè─w    SE handler
0019FFEC  00000000
0019FFF0  00000000
0019FFF4  00402000   @     IIfarasemn7.<ModuleEntryPoint>
0019FFF8  00322000   2
0019FFFC  00000000
```



```
;(a-2)/(b+c)+a*c+e-x=20/106+2200+10-1000=0+2200+10-1000=1210
; our code starts here
segment code use32 class=code
    start:
        ;convertim a la dw in perecea DX:AX pentru impartire
        mov AX,0
        add AL,[a]
        sub AX,2;AX=(a-2)
        mov DX,0

        mov BX,0
        add BL,[b]
        add BX,[c];BX=(b+c)
        div BX     ;AX=DX:AX/BX=(a-2)/(b+c)=0

        mov DX,AX ;DX=AX=0
        mov EBX,0
        add BX,DX ;BX=DX=AX=0

        mov AL,[a];AL=a=22=16h
        mov AH,0   ;AX=a=22=16h

        mul word[c];DX:AX=a*c=2200

        push DX
        push AX
        pop EAX     ;EAX=DX:AX=a*c=2200

        add EAX,EBX;EAX=(a-2)/(b+c)+a*c=2200
        add EAX,[e];EAX=(a-2)/(b+c)+a*c+e=2210

        mov EDX,0

        sub EAX,dword[x]    ;EAX=EAX-[x]
        sbb EDX,dword[x+4] ;EDX=EAX-[x]-CF=(a-2)/(b+c)+a*c+e-x=1210=4BAh

        ;rezultat in EDX:EAX
        push    dword 0      ; push the parameter for exit onto the stack
```