Ministerul Educaţiei și Cercetării

al Republicii Moldova   
Universitatea Tehnică a Moldovei

Faculatea Calculatoare, Infromatică și Microelectronică   
  
  
  
  
  
  
  
RAPORT

# LUCRARE DE LABORATOR NR. 1

**la Arhitectura Calculatoarelor**

**Tema: Elaborare programelor în limbajul de asamblare** **MASM în Visual Studio**

A efectuat: Popa Cătălin

st. gr. TI-211

A verificat: asist. univ. Secrieru Adrian

UTM, Chișinău 2022

**Tema:**

Elaborare programelor în limbajul de asamblare MASM în Visual Studio.

**Scopul lucrării:**

Lucrarea urmărește familiarizarea studenților cu regiştrii microprocesoarelor pe 16, 32 şi 64 de biți, cu mediul de dezvoltare a programelor Visual Studio. Se prezintă setările mediului Visual Studio necesare pentru a elabora, rula şi depăna aplicaţiile elaborate în limbajul de asamblare MASM utilizând regiştri pe 16, 32 şi 64 de biţi.

**1. Codurile sursă**

**Program pe 16 biți**

;exemplu de program pe 16 biti

INCLUDE Irvine16.inc

.DATA

Promt DB 'Doresti sa devii programator?(da/nu)-[y/n]$'

Dad DB 13,10,'Vei deveni!',13,10,'$'

Nud DB 13,10,'Vei deveni filosof!',13,10,'$'

.CODE

main PROC

mov ax,@data

mov ds,ax

mov dx,OFFSET Promt

mov ah,9

Int 21h

mov ah,1

Int 21h

cmp al,'y'

jz IsDad

cmp al,'n'

jz IsNud

IsDad: mov dx,OFFSET Dad

Jmp SHORT Disp

IsNud: mov dx,OFFSET Nud

Disp: mov ah,9

Int 21h

mov ah,1

Int 21h

Exit

main ENDP

END main

Se remarca urmatoarele:

1. Fişierul **Irvine16.inc** conţine un set de proceduri, macro-uri ce completează codul sursă necesare pentru dezvoltarea programelor în Visual Studio;

2.Directiva *.DATA* marchează începutul segmentului de date. In cazul nostru acest segment contine 3 şiruri de caractere, fiecărui caracter i se atribuie câte un byte (**DB**) în memorie, cu offset Prompt, Dad și Nud, valorile13,10 sunt tratate ca comenzi –sfârșitul rândului (LF), din rând nou (CR);

3. **main PROC** indică începutul procedurii main în care este inclus tot codul programului, **main ENDP** indică sfârşitul procedurii. Directiva *END main* indică sfârşitul programului (numit si **punctul de intrare in program**), tot codul plasat după această directiva va fi ignorat de asamblor;

4.Primele doua instrucțiuni inițializează registrul *ds*. Simbolul *@data* reprezintă numele segmentului creat cu directiva *.DATA* (sau adresa segmentului, adresa primului octet din acest segment);

5.Continutul variabilelor Prompt, Dad și Nud de tip șir de caractere se afișează pe ecran utilizând funcția *MSDOS* cu codul *9*. Șirurile pot fi de asemenea afișate pe ecran cu funcția *MSDOS* cu codul *40h*;

6.Întreruperea Int 21h apelează serviciul *MSDOS* (DOS Services), iar *int 10h* la serviciul BIOS, ca funcţia apelată să fie executată, codul ei este necesar să fie încărcat în registrul **al**.

**Şablonul (template) unui program pe 16 de biţi este următorul:**

; This program

; Last update:

INCLUDE Irvine16.inc

.data

; aici definiti datele necesare

.code

main PROC

mov ax,@data

mov ds,ax

;aici plasati codul program

exit

main ENDP

END main

**Program pe 32 biți**

INCLUDE Irvine32.inc

.data

Promt DB 'Doresti sa devii programator?(da/nu)-[y/n]',0

Dad DB 13,10,'Vei deveni!',13,10,0

Nud DB 13,10,'Vei deveni filosof!',13,10,0

.code

main PROC

mov edx,OFFSET Promt

call WriteString

call ReadChar

cmp al,'y'

jz IsDad

cmp al,'n'

jz IsNud

IsDad: mov edx,OFFSET Dad

call WriteString

jmp ex

IsNud: mov edx,OFFSET Nud

call WriteString

ex:

exit

main ENDP

END main

Se remarca urmatoarele:

1. Se observă, că este acelaşi program dar elaborat ca o aplicație pe 32 de biţi, în care se utilizaeză procedurile din **Irvine32.inc,** pentru dezvoltarea programelor în Visual Studio;
2. Definirea şirurilor de caractere se finalizează cu zero, offset-ul şirurilor se încarcă în registrul **edx** (pe 32 de biţi).
3. Cu instrucţiunea **call** se apelează două proceduri WriteString – afişarea şirului şi ReadChar- introducerea unui caracter de la tastatură. Pentru afişarea unui şir de caractere, offset-ul şirului este necesar să fie încărcat în registrul **edx** şi apoi apelată procedura (funcţia). După apelarea procedurii ReadChar, în registrul **al** se va introduce codul tastei.

**Program pe 64 biți**

; AddTwoSum\_64.asm - Ch3 example.

ExitProcess proto

.data

sum qword 0

.code

main proc

mov rax,5

add rax,6

mov sum,rax

mov ecx,0

call ExitProcess

main endp

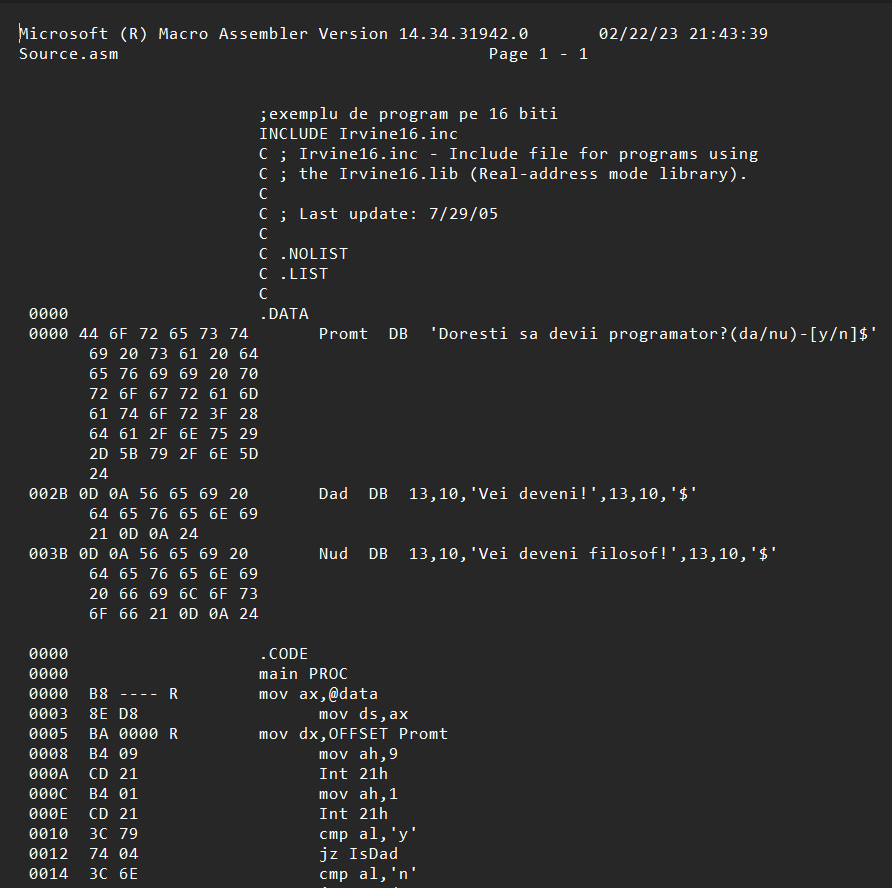
end

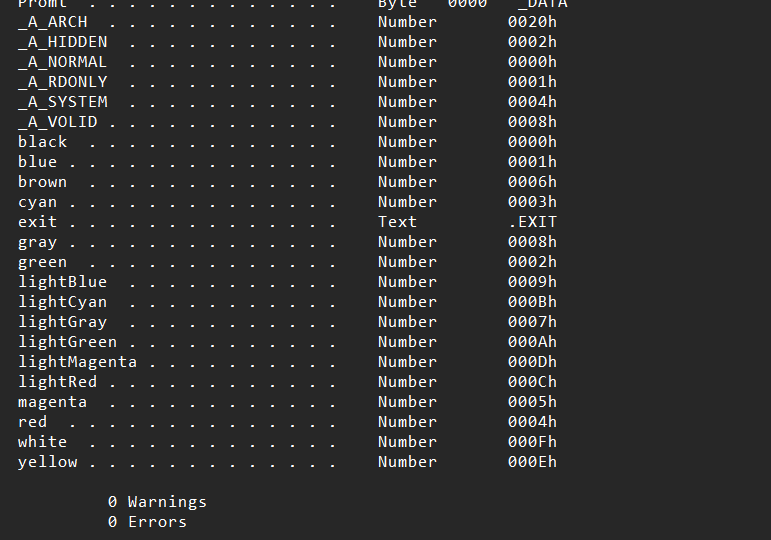
Se remarca urmatoarele:

1. Este un program ce adună două numere întregi şi rezultatul este salvat în variabila **sum** pe 64 biţi (qword).
2. Procedura de ieşire din program ExitProcess, şi alte proceduri utilizate, este necesar să le declarăm la începutul programului, ExitProcess proto.

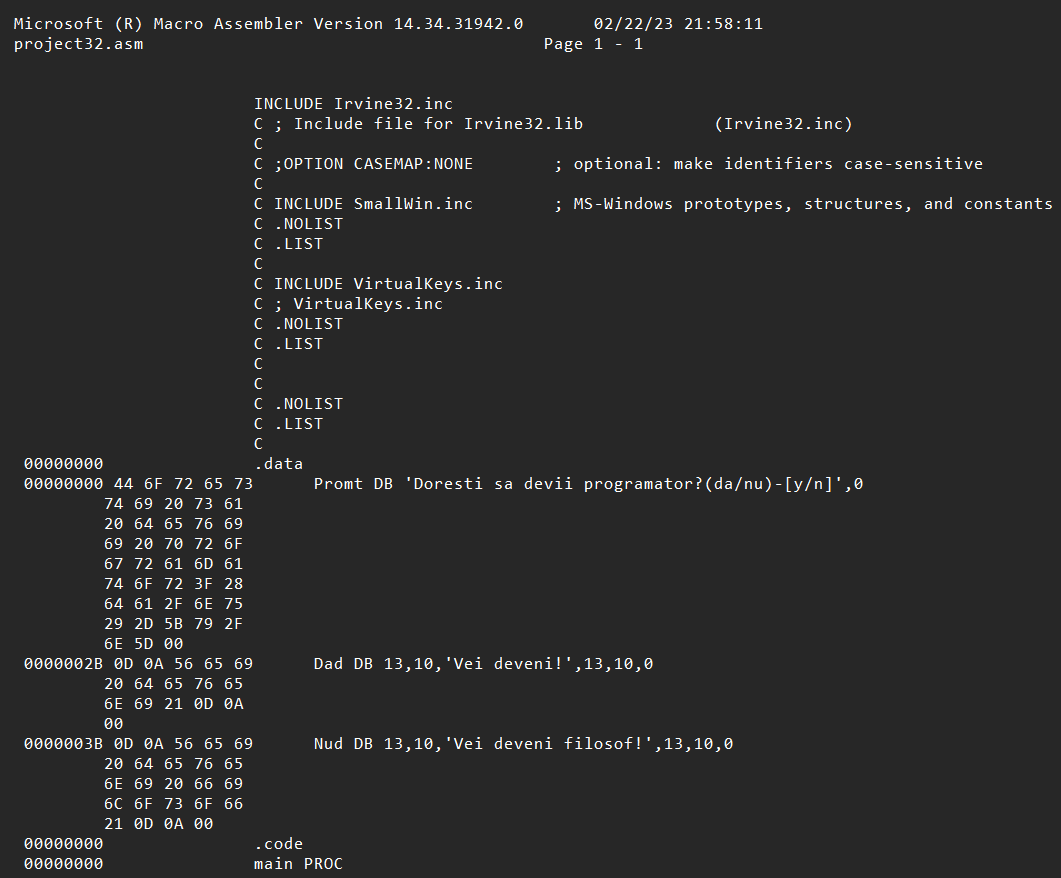
**2. Fișierele-listing**

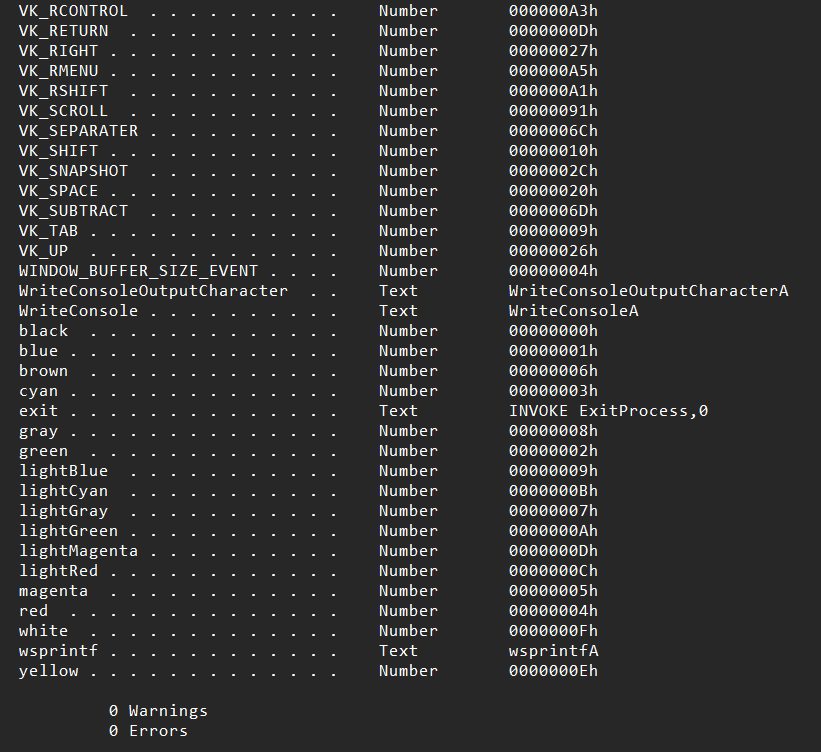
**Program pe 16 biți**

****

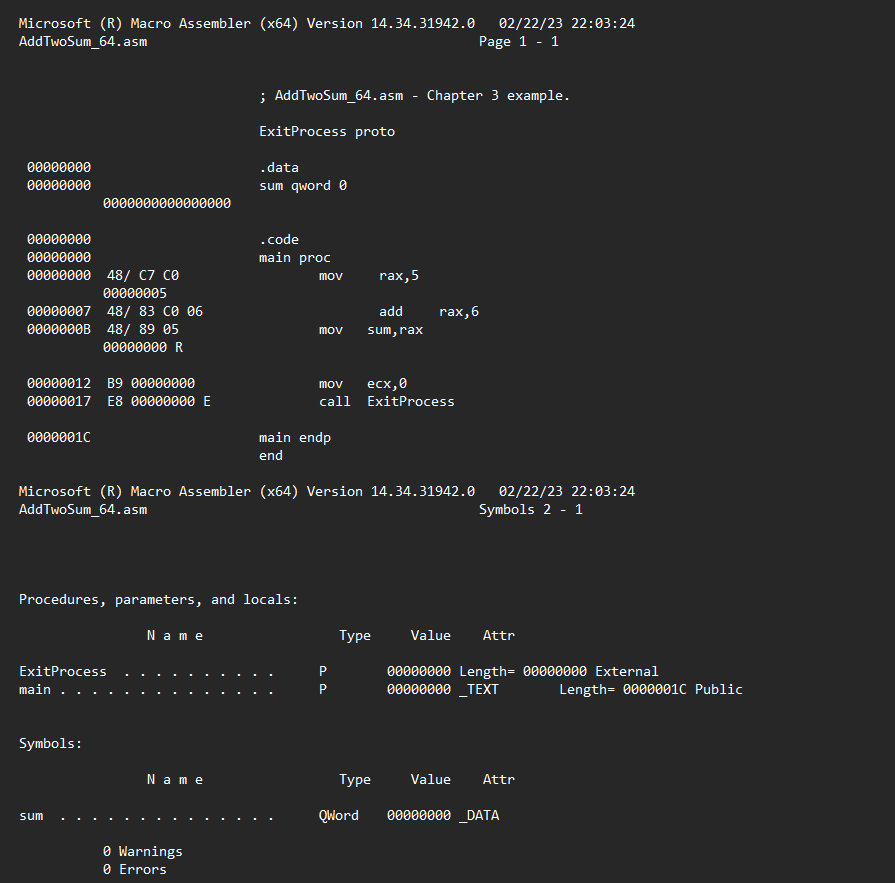
****

**Program pe 32 biți**

****

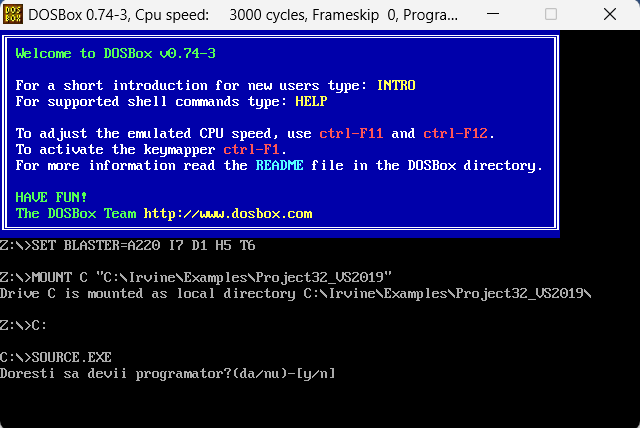
****

**Program pe 64 biți**

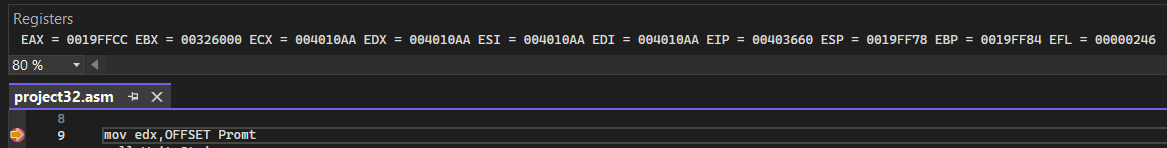
****

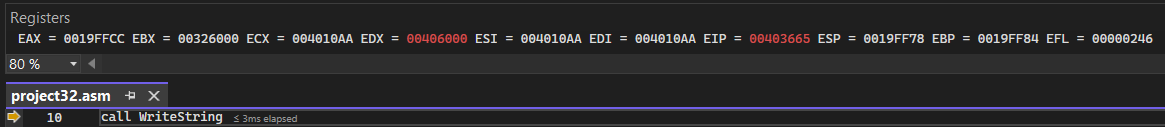
**3. Pașii rulării programelor în Debug**

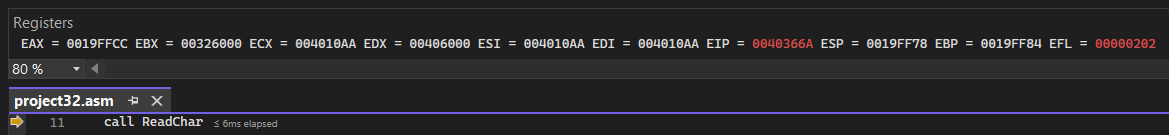
**Program pe 16 biți**

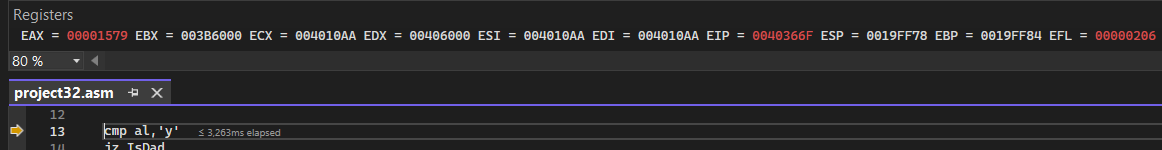
****

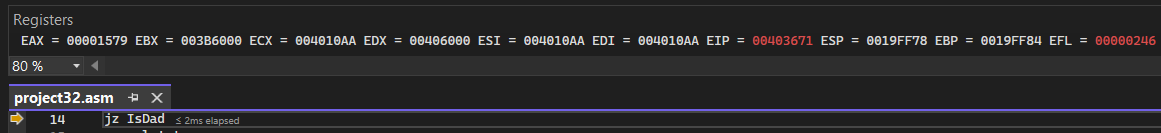
**Program pe 32 biți**

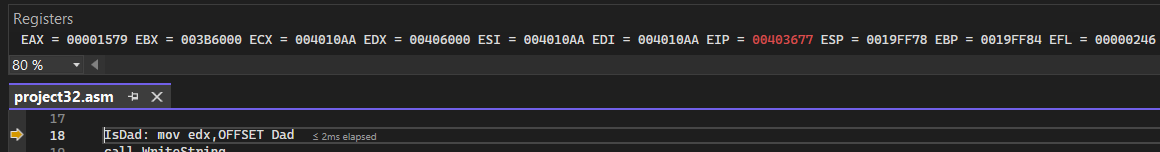


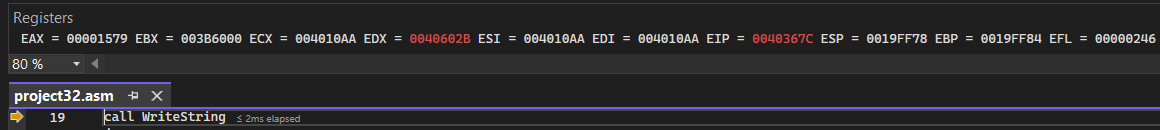


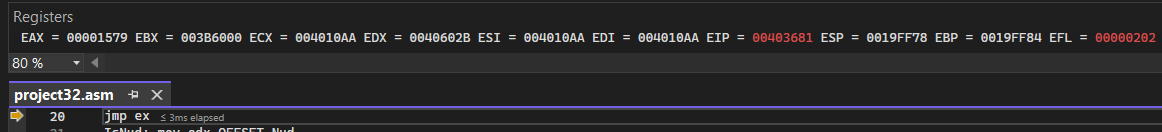


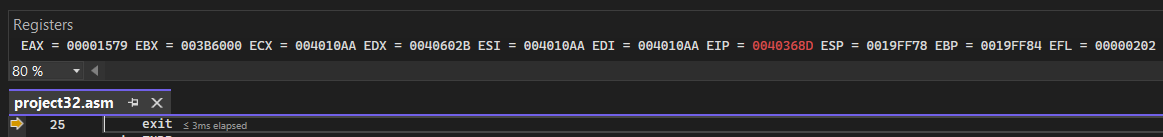




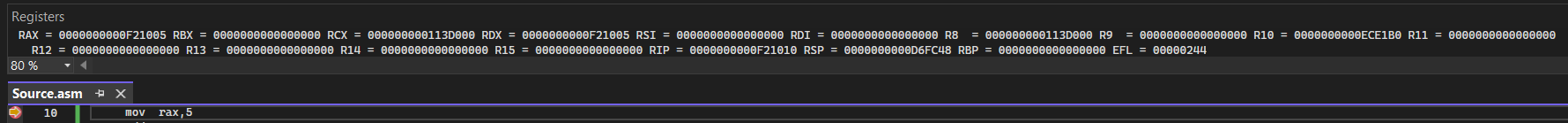


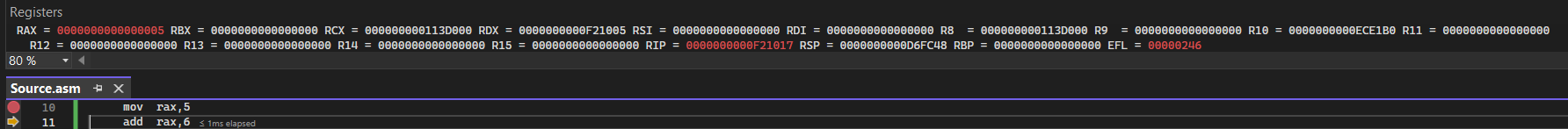


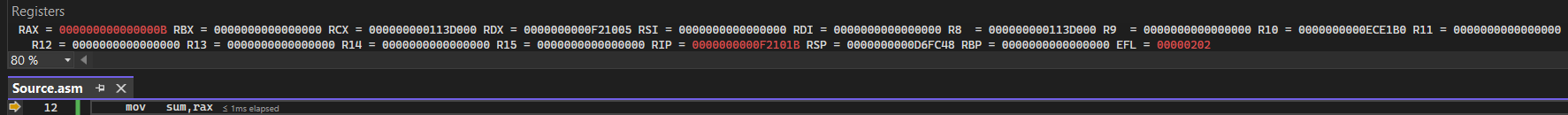


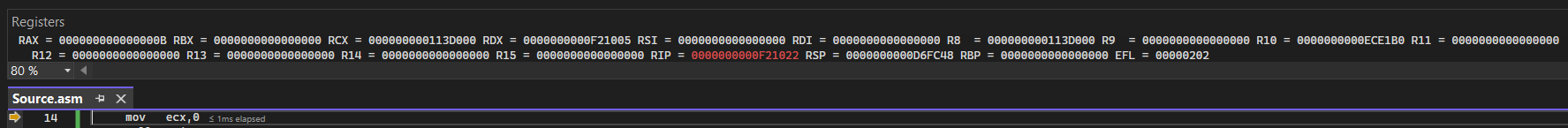


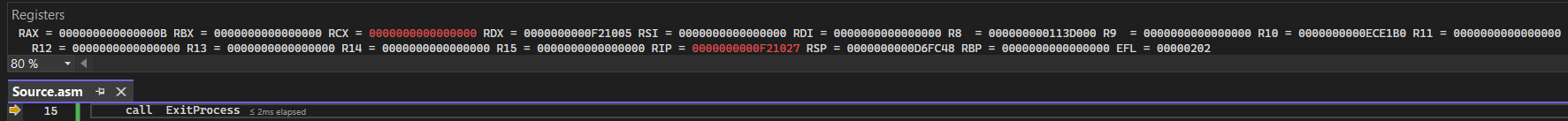
**Program pe 64 biți**











**4. Concluzii**

În lucrarea dată, m-am familiarizat cu regiștrii microprocesoarelor pe 16, 32 și 64 de biți, cu mediul de dezvoltare a programelor Visual Studio. S-a prezentat stările mediului Visual Studio pentru a elabora, rula și depăna aplicațiile elaborate în limbajul de asamblare MASM utilizând regiștri pe 16, 32 și 64 de biți. Am obținut fișiere .exe, care se află în Project, cu numele Project.exe. De asemenea, am obținut și am analizat fișierele listing ale programelor Project.lst. Am rulat sub Debag pas cu pas exemplele de programe actualizând Registers și Watch1(pentru variabile). În final am introdus datele obținute și afișările în raport.