Ministerul Educaţiei și Cercetării

al Republicii Moldova   
Universitatea Tehnică a Moldovei

Faculatea Calculatoare, Infromatică și Microelectronică   
  
  
  
  
  
  
  
RAPORT

# LUCRARE DE LABORATOR NR. 2

**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:**

Se prezinta problemele principale legate de conversii de date, reprezentarea datelor întregi, reprezentarea întregilor in format BCD, reprezentarea caracterelor si a șirurilor de caractere, reprezentarea valorilor reale, elemente de memorie, tipuri de date utilizate si modurile de adresare a operanzilor.

**1. Codurile sursă**

**p.3.1**

INCLUDE Irvine32.inc

.data

alfa DW 3 DUP(?)

.code

main proc

mov ax,17 ; adresare indirecta a operandului sursa

mov ax,10101b ;

mov ax,11b ;

mov ax,21o ;

mov alfa,ax ; Adresare directa a operandului destinatie

mov cx,ax ; Interschimba registrele ax si bx

mov ax,bx ; Folosind registrul cx

mov ax,cx ;

xchg ax,bx ; Interschimba direct cele 2 registre.

mov si,2

mov alfa[si],ax ; Adresare relativa cu registrul si

mov esi,2

mov ebx,offset alfa ; Adresare imediata a operandului sursa

lea ebx,alfa ; Acelasi efect

mov ecx,[ebx][esi] ; Adresare bazata indexata a sursei

mov cx,alfa[2] ; Acelasi efect.

mov cx,[alfa+2] ; Acelasi efect

mov di,4

mov byte ptr [ebx][edi],55h ;

mov esi,2

mov ebx,3

mov alfa[ebx][esi],33h ; Adresare bazata indexata relativa a

; destinatiei

mov alfa[ebx+esi],33h ; Notatii echivalente

mov [alfa+ebx+esi],33h

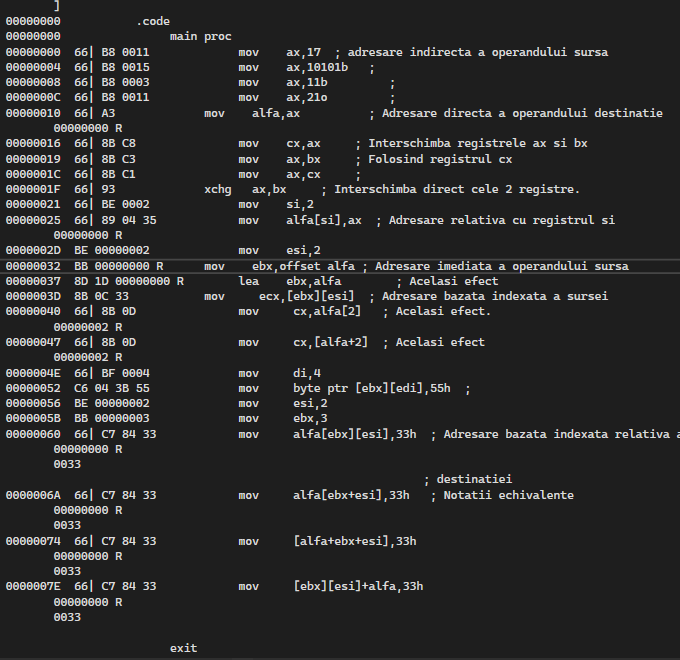
mov [ebx][esi]+alfa,33h

exit

main ENDP

END main

**Listing code**



**p.3.2**

INCLUDE Irvine32.inc

; Sa se calculeze expresia aritmetica: e=((a+b\*c-d)/f+g\*h)/i

; se considera a, d, f – cuvant b, c, g, h, i –byte

; ca sa putem executa impartirea cu f convertim impartitorul la dublucuvânt

; ne vor interesa doar caturile impartirilor, rezultatul va fi de tip octet

.data

a dw 5

b db 6

cd db 10

d dw 5

f dw 6

g db 10

h db 11

i db 10

interm dw ?

rez db ?

.code

main proc

mov eax,0

mov al, b

imul cd ; in ax avem b\*c

add ax, a ; ax=b\*c+a

sub ax, d ; ax=b\*c+a-d

cwd ; am convertit cuvantul din ax, in dublu cuvantul , retinut in dx:ax

idiv f ; obtinem câtul în ax si restul în dx ax=(a+b\*c-d)/f

mov interm, ax ; interm=(a+b\*c-d)/f

mov al, g

imul h ; ax=g\*h

add ax, interm ; ax=(a+b\*c-d)/f+g\*h

idiv i ; se obtine catul în al si restul în ah

mov rez, al

exit

main ENDP

END main

**2. Probelma propusa**

**Varianta 11**

*z=(5\*a-b/7)/(3/b+a\*a)*

INCLUDE Irvine32.inc ; include biblioteca Irvine32

.DATA

a DWORD 1 ; valoarea lui a

b DWORD 2 ; valoarea lui b

z DWORD ? ; valoarea lui z

intreg DWORD 0 ;

rest DWORD 0

alMsg db "Intreg: ", 0

ahMsg db "Rest: ", 0

.CODE

main PROC

; calculeaza z=(5\*a-b/7)/(3/b+a\*a)

mov eax, a ; incarca valoarea lui a in registrul eax

imul eax, 5 ; inmulteste a cu 5 si stocheaza rezultatul in eax

mov ebx, eax ; eax -> ebx

mov eax, b ; incarca valoarea lui b in registrul ebx

mov ecx, 7 ; incarca 7 in registrul ecx

cdq ; semn-extinde valoarea lui b la edx:eax

idiv ecx ; imparte la 7 si stocheaza rezultatul in eax

add intreg, eax ; adaugam partea intreaga

add rest, edx ; adaugam restul

sub ebx, eax ; scade b/7 din 5\*a si stocheaza rezultatul in eax

add intreg, ebx

mov eax, 3 ; incarca 3 in registrul ecx

mov ebx, b ; stocam valoarea lui b in ebx

cdq ; semn-extinde valoarea

idiv ebx ; imparte ebx:eax si stocheaza rezultatul in eax

add rest, edx ; adaugam ebx in rest

mov ebx, eax ; mutam eax->ebx

mov eax, a ; incarca valoarea lui a in registrul eax

imul eax ; inmulteste a cu a si stocheaza rezultatul in eax

add ebx, eax ; aduna eax cu ebx

mov eax, intreg ; intreg -> eax

cdq ; semn-extinde valoarea

idiv ebx ; imparte pentru a afla rezultul final

add rest, edx ; aflam restul total

mov edx, offset alMsg

call writestring

call writeint ; afisam partea intreaga

call crlf

mov edx, offset ahMsg

call writestring

mov eax, rest ; mutam restul in eax

call writeint ; afisam restul

exit ; iese din program

main ENDP

END main

Listing code

INCLUDE Irvine32.inc ; include biblioteca Irvine32

C ; Include file for Irvine32.lib (Irvine32.inc)

C

C ;OPTION CASEMAP:NONE ; optional: make identifiers case-sensitive

C

C INCLUDE SmallWin.inc ; MS-Windows prototypes, structures, and constants

C .NOLIST

C .LIST

C

C INCLUDE VirtualKeys.inc

C ; VirtualKeys.inc

C .NOLIST

C .LIST

C

C

C .NOLIST

C .LIST

C

00000000 .DATA

00000000 00000001 a DWORD 1 ; valoarea lui a

00000004 00000002 b DWORD 2 ; valoarea lui b

00000008 00000000 z DWORD ? ; valoarea lui z

0000000C 00000000 intreg DWORD 0 ;

00000010 00000000 rest DWORD 0

00000014 49 6E 74 72 65 alMsg db "Intreg: ", 0

67 3A 20 00

0000001D 52 65 73 74 3A ahMsg db "Rest: ", 0

20 00

00000000 .CODE

00000000 main PROC

; calculeaza z=(5\*a-b/7)/(3/b+a\*a)

00000000 A1 00000000 R mov eax, a ; incarca valoarea lui a in registrul eax

00000005 6B C0 05 imul eax, 5 ; inmulteste a cu 5 si stocheaza rezultatul in eax

00000008 8B D8 mov ebx, eax ; eax -> ebx

0000000A A1 00000004 R mov eax, b ; incarca valoarea lui b in registrul ebx

0000000F B9 00000007 mov ecx, 7 ; incarca 7 in registrul ecx

00000014 99 cdq ; semn-extinde valoarea lui b la edx:eax

00000015 F7 F9 idiv ecx ; imparte la 7 si stocheaza rezultatul in eax

00000017 01 05 0000000C R add intreg, eax ; adaugam partea intreaga

0000001D 01 15 00000010 R add rest, edx ; adaugam restul

00000023 2B D8 sub ebx, eax ; scade b/7 din 5\*a si stocheaza rezultatul in eax

00000025 01 1D 0000000C R add intreg, ebx

0000002B B8 00000003 mov eax, 3 ; incarca 3 in registrul ecx

00000030 8B 1D 00000004 R mov ebx, b ; stocam valoarea lui b in ebx

00000036 99 cdq ; semn-extinde valoarea

00000037 F7 FB idiv ebx ; imparte ebx:eax si stocheaza rezultatul in eax

00000039 01 15 00000010 R add rest, edx ; adaugam ebx in rest

0000003F 8B D8 mov ebx, eax ; mutam eax->ebx

00000041 A1 00000000 R mov eax, a ; incarca valoarea lui a in registrul eax

00000046 F7 E8 imul eax ; inmulteste a cu a si stocheaza rezultatul in eax

00000048 03 D8 add ebx, eax ; aduna eax cu ebx

0000004A A1 0000000C R mov eax, intreg ; intreg -> eax

0000004F 99 cdq ; semn-extinde valoarea

00000050 F7 FB idiv ebx ; imparte pentru a afla rezultul final

00000052 01 15 00000010 R add rest, edx ; aflam restul total

00000058 BA 00000014 R mov edx, offset alMsg

0000005D E8 00000000 E call writestring

00000062 E8 00000000 E call writeint ; afisam partea intreaga

00000067 E8 00000000 E call crlf

0000006C BA 0000001D R mov edx, offset ahMsg

00000071 E8 00000000 E call writestring

00000076 A1 00000010 R mov eax, rest ; mutam restul in eax

0000007B E8 00000000 E call writeint ; afisam restul

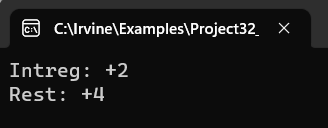
exit ; iese din program

00000080 6A 00 \* push +000000000h

00000082 E8 00000000 E \* call ExitProcess

00000087 main ENDP

END main



**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.