****

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Ingineria Software i Automatică**

**Raport**

**Lucrarea de laborator №2**

Disciplina: Arhitectura Calculatoarelor.

Tema: Elaborarea programelor în limbajul de asamblare MASM în Visual Studio

**Realizat:** Bunescu Gabriel, TI-207

**Profesor:** asist.univ.mag.Secrieru Adrian

Chișinău 2022

**Scopul lucrării:**

Se prezintă problemele principale legate de conversii de date, reprezentarea datelor întregi,

reprezentarea întregilor în format BCD, reprezentarea caracterelor și a sirurilor de caractere,

reprezentarea valorilor reale, elemente de memorie, tipuri de date utilizate și modurile de adresare a

operanzilor.

**Mersul lucrării:**

1. **Codul sursă programul 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

1. **Codul sursă programul 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 dublucuv?nt

; 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 c?tul ?n ax si restul ?n 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 ?n al si restul ?n ah

mov rez, al

exit

main ENDP

END

1. **Codul sursă programul 3.3**



INCLUDE Irvine32.inc

; Sa se calculeze expresia aritmetica: . z=(a+b+c+1) \*(a+b+c+1) /((a-b+d)\*(a-b+d))

.data

a dw 5

b dw 6

cd dw 10

d dw 9

interm dw ?

interm1 dw ?

rez db ?

.code

main proc

mov eax, 0

mov ax, a

add ax,b ; in ax avem a+b

add ax, cd ; ax=a+b+c

add ax, 1 ; ax=a+b+c+1

mov interm, ax; interm=(a+b+c+1)

imul ax, ax

mov interm, ax; interm=(a+b+c+1)\*(a+b+c+1)

mov eax, 0

mov ax, a

sub ax,b ; in ax avem a-b

add ax, d ; ax=a-b+d

mov interm1, ax; interm=(a-b+d)

imul ax, ax

mov interm1, ax; interm=(a-b+d)

xchg interm1, ax

xchg ax, interm

cwd

div interm

mov rez, al

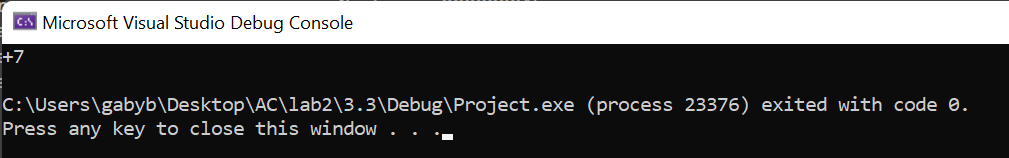
call WriteInt

call Crlf

exit

main ENDP

END



**Concluzie:**

În urma efectuării lucrării de laborator nr. 2 s-a realizat un program ce efectuează mai multe operații aritmetice pentru calcularea unei valori finale Z în limbajul assembler. S-au utilizat douătipuri de date –  db -define byte și dw - define word. Totodată s-au utilizat și instructiuni precum add, mov, imul, div, etc. și s-au pus în practică cunoștințele referitor la modurile de adresare a operanzilor.