**MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**RAPORT**

Lucrare de laborator nr. 2

la cursul ***„Arhitectura calculatoarelor”***

***Varianta 4***

**A efectuat:**  **St. gr. CR-221FR Serba Cristina**

**A verificat: lect. univ. Postovan Dumitru**

**Chișinău 2024**

# 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.

# Codurile 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 |

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 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 |

Programul 4:

Să se calculeze expresia :

*4. z=(a\*3+b\*b\*5)/(a\*a+2\*a\*b)-a-b*

|  |
| --- |
| INCLUDE Irvine32.inc  ; Sa se calculeze ((a\*3+b\*b\*5)/(a\*a+2\*a\*b))-a-b  .data  a dw 1  b db 8  interm dw ?  rez db ?  .code  main proc  ; Calculam numaratorul  mov ax, a ;ax = 1  imul ax, 3 ;ax\*3  mov cx, ax ;salvam valoarea obtinuta in cx  movzx ax, b ;ax =8  imul ax, ax ;ax\*ax  imul ax, 5 ;ax\*5  add ax, cx ;ax + cx (valoarea salvata anterior)  mov interm, ax ;salvam valoarea numaratorului  ; Calculam numitorul  mov ax, a ;ax = a  imul ax, a ;ax \* a  mov cx, ax ; salvam valoarea obtinuta in cx  movzx dx, b ;dx = b  imul dx, 2 ;dx \* 2  imul ax, dx ;ax \* dx  add ax, cx ;ax + cx (valoarea salvata anterior)  mov cx, ax ; Salvam valuarea obtinuta in cx  mov ax, interm ;Restabilim numaratorul  cwd ;Extindem valoarea numaratorului pentru a putea imparti  idiv cx ;Impartim valoarea extinsa la cx si salvam in ax  sub ax, a ;Rezultatul obtinut la impartire - a  movzx dx, b ;dx = b (pe 16 biti)  sub ax, dx ;Rezultatul precedent - dx  mov rez, al ;Salvam rezultatul final in rez  exit  main ENDP  END main |

# Fișierele listing:

Listing-ul 3.1:

|  |
| --- |
| Microsoft (R) Macro Assembler Version 14.00.24210.0 11/23/24 16:25:22  program2.asm Page 1 - 1  INCLUDE Irvine32.inc  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 00000003 [ alfa DW 3 DUP(?)  0000  ]  00000000 .code  00000000 main proc  00000000 66| B8 0011 mov ax,17 ; adresare indirecta a operandului sursa  00000004 66| B8 0015 mov ax,10101b ;  00000008 66| B8 0003 mov ax,11b ;  0000000C 66| B8 0011 mov ax,21o ;  00000010 66| A3 mov alfa,ax ; Adresare directa a operandului destinatie  00000000 R  00000016 66| 8B C8 mov cx,ax ; Interschimba registrele ax si bx  00000019 66| 8B C3 mov ax,bx ; Folosind registrul cx  0000001C 66| 8B C1 mov ax,cx ;  0000001F 66| 93 xchg ax,bx ; Interschimba direct cele 2 registre.  00000021 66| BE 0002 mov si,2  00000025 66| 89 04 35 mov alfa[si],ax ; Adresare relativa cu registrul si  00000000 R  0000002D BE 00000002 mov esi,2  00000032 BB 00000000 R mov ebx,offset alfa ; Adresare imediata a operandului sursa  00000037 8D 1D 00000000 R lea ebx,alfa ; Acelasi efect  0000003D 8B 0C 1E mov ecx,[ebx][esi] ; Adresare bazata indexata a sursei  00000040 66| 8B 0D mov cx,alfa[2] ; Acelasi efect.  00000002 R  00000047 66| 8B 0D mov cx,[alfa+2] ; Acelasi efect  00000002 R  0000004E 66| BF 0004 mov di,4  00000052 C6 04 1F 55 mov byte ptr [ebx][edi],55h ;  00000056 BE 00000002 mov esi,2  0000005B BB 00000003 mov ebx,3  00000060 66| C7 84 1E mov alfa[ebx][esi],33h ; Adresare bazata indexata relativa a  00000000 R  0033  ; destinatiei  0000006A 66| C7 84 1E mov alfa[ebx+esi],33h ; Notatii echivalente  00000000 R  0033  00000074 66| C7 84 1E mov [alfa+ebx+esi],33h  00000000 R  0033  0000007E 66| C7 84 1E mov [ebx][esi]+alfa,33h  00000000 R  0033  exit  0000008F main ENDP  END main  Microsoft (R) Macro Assembler Version 14.00.24210.0 11/23/24 16:25:22  program2.asm Symbols 2 - 1 |

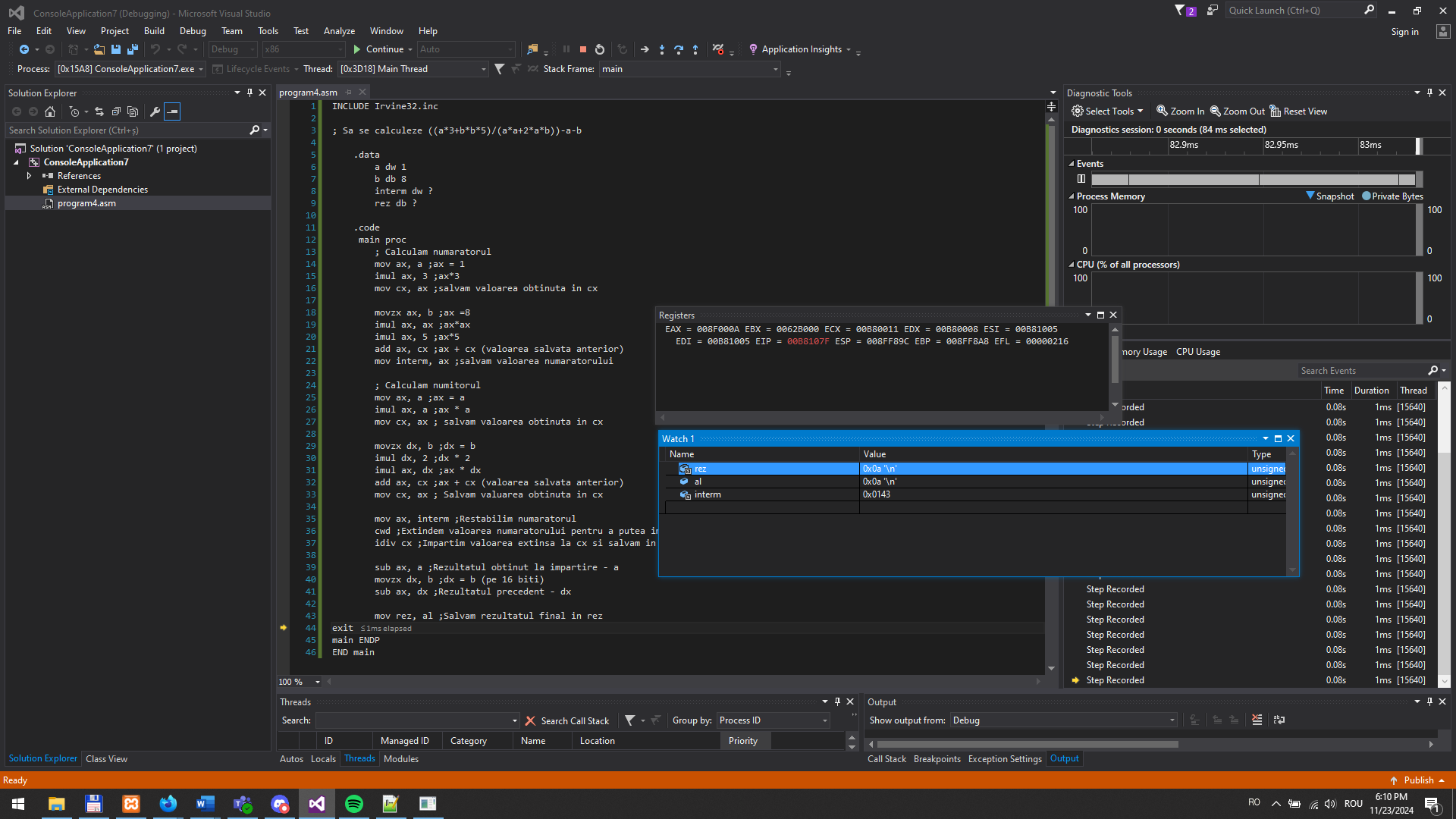
Listing-ul 3.2:

|  |
| --- |
| Microsoft (R) Macro Assembler Version 14.00.24210.0 11/23/24 16:37:44  program3.asm Page 1 - 1  INCLUDE Irvine32.inc  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  ; 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    00000000 .data  00000000 0005 a dw 5  00000002 06 b db 6  00000003 0A cd db 10  00000004 0005 d dw 5  00000006 0006 f dw 6  00000008 0A g db 10  00000009 0B h db 11  0000000A 0A i db 10  0000000B 0000 interm dw ?  0000000D 00 rez db ?  00000000 .code  00000000 main proc  00000000 B8 00000000 mov eax,0  00000005 A0 00000002 R mov al, b  0000000A F6 2D 00000003 R imul cd ; in ax avem b\*c  00000010 66| 03 05 add ax, a ; ax=b\*c+a  00000000 R  00000017 66| 2B 05 sub ax, d ; ax=b\*c+a-d  00000004 R  0000001E 66| 99 cwd ; am convertit cuvantul din ax, in dublu cuvantul , retinut in dx:ax  00000020 66| F7 3D idiv f ; obtinem câtul în ax si restul în dx ax=(a+b\*c-d)/f  00000006 R  00000027 66| A3 mov interm, ax ; interm=(a+b\*c-d)/f  0000000B R  0000002D A0 00000008 R mov al, g  00000032 F6 2D 00000009 R imul h ; ax=g\*h  00000038 66| 03 05 add ax, interm ; ax=(a+b\*c-d)/f+g\*h  0000000B R  0000003F F6 3D 0000000A R idiv i ; se obtine catul în al si restul în ah  00000045 A2 0000000D R mov rez, al  exit  00000051 main ENDP  END main  Microsoft (R) Macro Assembler Version 14.00.24210.0 11/23/24 16:37:44  program3.asm Symbols 2 - 1 |

Listing-ul programului 4:

|  |
| --- |
| Microsoft (R) Macro Assembler Version 14.00.24210.0 11/23/24 18:06:40  program4.asm Page 1 - 1  INCLUDE Irvine32.inc  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  ; Sa se calculeze ((a\*3+b\*b\*5)/(a\*a+2\*a\*b))-a-b  00000000 .data  00000000 0001 a dw 1  00000002 08 b db 8  00000003 0000 interm dw ?  00000005 00 rez db ?  00000000 .code  00000000 main proc  ; Calculam numaratorul  00000000 66| A1 mov ax, a ;ax = 1  00000000 R  00000006 66| 6B C0 03 imul ax, 3 ;ax\*3  0000000A 66| 8B C8 mov cx, ax ;salvam valoarea obtinuta in cx  0000000D 66| 0F B6 05 movzx ax, b ;ax =8  00000002 R  00000015 66| 0F AF C0 imul ax, ax ;ax\*ax  00000019 66| 6B C0 05 imul ax, 5 ;ax\*5  0000001D 66| 03 C1 add ax, cx ;ax + cx (valoarea salvata anterior)  00000020 66| A3 mov interm, ax ;salvam valoarea numaratorului  00000003 R  ; Calculam numitorul  00000026 66| A1 mov ax, a ;ax = a  00000000 R  0000002C 66| 0F AF 05 imul ax, a ;ax \* a  00000000 R  00000034 66| 8B C8 mov cx, ax ; salvam valoarea obtinuta in cx  00000037 66| 0F B6 15 movzx dx, b ;dx = b  00000002 R  0000003F 66| 6B D2 02 imul dx, 2 ;dx \* 2  00000043 66| 0F AF C2 imul ax, dx ;ax \* dx  00000047 66| 03 C1 add ax, cx ;ax + cx (valoarea salvata anterior)  0000004A 66| 8B C8 mov cx, ax ; Salvam valuarea obtinuta in cx  0000004D 66| A1 mov ax, interm ;Restabilim numaratorul  00000003 R  00000053 66| 99 cwd ;Extindem valoarea numaratorului pentru a putea imparti  00000055 66| F7 F9 idiv cx ;Impartim valoarea extinsa la cx si salvam in ax  00000058 66| 2B 05 sub ax, a ;Rezultatul obtinut la impartire - a  00000000 R  0000005F 66| 0F B6 15 movzx dx, b ;dx = b (pe 16 biti)  00000002 R  00000067 66| 2B C2 sub ax, dx ;Rezultatul precedent - dx  0000006A A2 00000005 R mov rez, al ;Salvam rezultatul final in rez  exit  00000076 main ENDP  END main  Microsoft (R) Macro Assembler Version 14.00.24210.0 11/23/24 18:06:40  program4.asm Symbols 2 - 1 |

Rezultatul executării programului 4:



# Concluzii:

În urma efectuării lucrării de laborator, am obținut cunoștințe în rularea programelor assembly și manipularea datelor cu ajutorul regiștrilor. Astfel, am aplicat cunoștințele despre lucrul cu comenzi precum add, mov, imul, idiv, sub. Întrucât am lucrat cu regiștri de dimensiuni diferite și împărțiri de 32 de biți, a fost esențial să am grijă la manipularea valorilor înregistrate în ax, dx, și cx pentru a nu le suprascrie sau utiliza incorect. Programarea în assembly necesită o înțelegere detaliată a arhitecturii procesorului și a modului în care datele sunt stocate și manipulate în registre.