**Department of Electrical Engineering**

|  |  |
| --- | --- |
| **Faculty Member: Ma’am Qurat-ul-ain** | **Dated: November 5, 2020** |
|  |  |
| **Course/Section: BSCS-9B** | **Semester: 3rd** |
|  |  |

**Computer Organization and**

**Assembly Language (CS235)**

**Lab #4 Loops and Arrays in Assembly Language using MASM**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **PLO4 / CLO4** | | **PLO5/ CLO5** | **PLO8/ CLO 6** | **PLO9/ CLO 7** |  |
| **Name** | **Roll number** | **Viva /Quiz/ Lab performance**  **5 marks** | **Analysis of data in lab report**  **5 marks** | **Modern tool Usage**  **5 marks** | **Ethics and Safety**  **5 marks** | **Individual and team work**  **5 marks** | **Total**  **25 marks** |
| Fatima Seemab | 291310 |  |  |  |  |  |  |
| Mahum Samar | 290647 |  |  |  |  |  |  |
| Maryam Fatima | 290479 |  |  |  |  |  |  |

**Objective:** The aim of this lab is to use loops and arrays for data manipulation.

**Implementing a loop**:

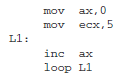
Mov ecx,10 //Exceutes loop 10 times

mylabel: // loop lab, You can change laber name

calldoSomething //loop body

loop mylabel //loop

Example



The ecx register is automatically decremented and tested by the loop instruction. The loop

terminates when ecx becomes 0. To make this a true for loop, another test is required before

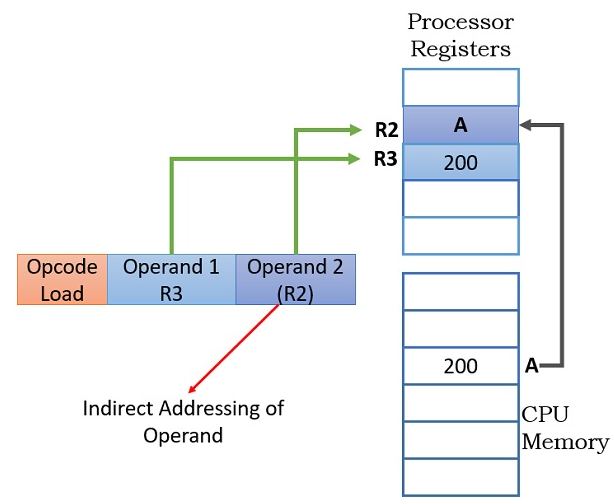
entering the loop. Be sure that the loop body doesn't mess up ecx.

**Register indirect addressing**

Access memory through address in a register. A processor register is used to hold the address of a memory location where the operand is placed.

**Examples:**

Load R3, [R2]



mov [esi],25

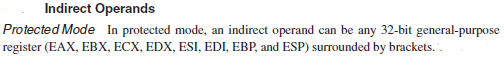
add [eax],ebx

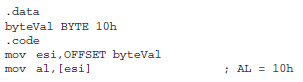
mov eax,[edi]

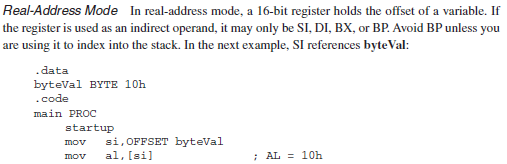
Brackets [ ] mean "memory referenced by the address in".

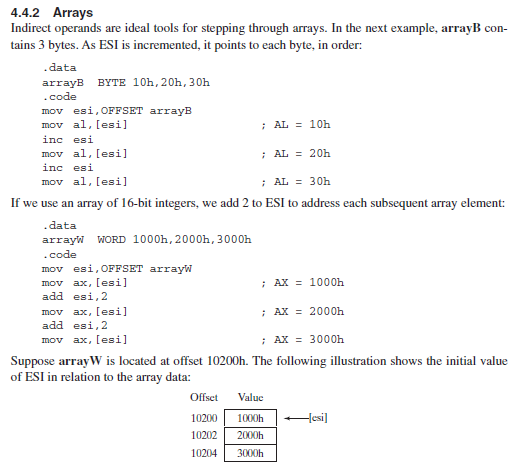
Note that the following instruction is invalid because it attempts to add memory to memory:

add [eax],[edi]

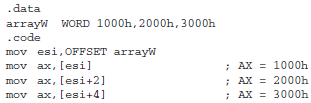
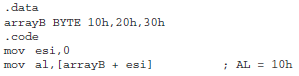








OR other ways for arrays use in code

* 
* 
* 

**Exercise 1:** Define three arrays to hold four one-word integers each. Initialize the first two with the numbers: 2,3,4,5 and 3,4,5,6. Keep 3rd array uninitialized.

**Defining and Initializing:**

.data

arrayA WORD 2,3,4,5

arrayB WORD 3,4,5,6

arrayC WORD 4 DUP(?) ; uninitialized array

**Exercise 2:** Write code, assemble and execute it, to display the sum of each of the first two arrays in registers EAX and EBX. Make use of loops for addition. Print the following message before the results are displayed:

“1. The sums of the arrays are shown in the registers EAX and EBX:”

**Program:**

TITLE initialize three arrays

INCLUDE Irvine32.inc

.data

arrayA WORD 2,3,4,5

arrayB WORD 3,4,5,6

arrayC WORD 4 DUP(?) ; uninitialized array

string BYTE 'The sums of the arrays are shown in the registers EAX and EBX',0

.code

main proc

mov edx,OFFSET string

call writestring

call crlf

mov ecx,4 ; execute loop 4 times

mov eax,0 ; getting rid of garbage values

mov ebx,0

mov esi,0 ; initialize esi

L1:

add ax,arrayA[si] ; add elements of arrayA in eax

add bx,arrayB[si] ; add elements of arrayB in ebx

add si,2

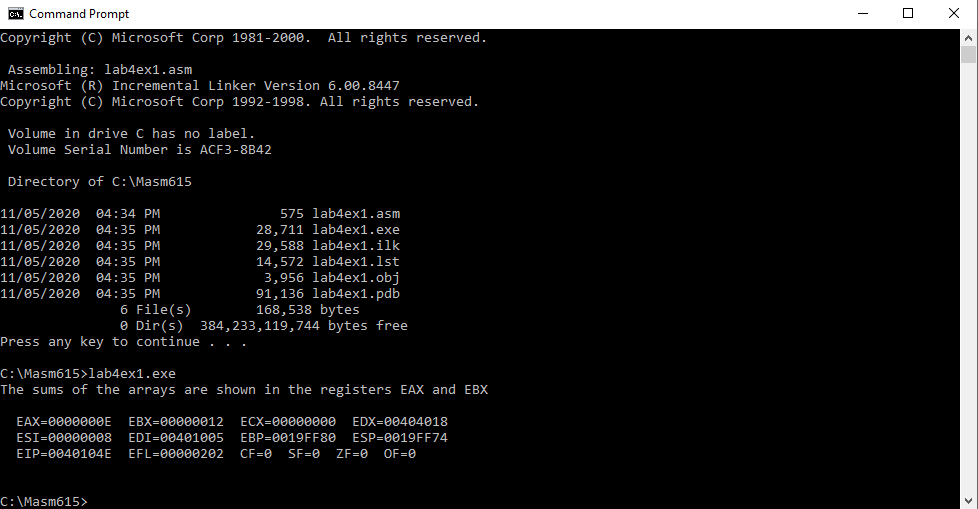
loop L1

call dumpregs

exit

main endp

end main



EAX = 0000000E = 14\* 16^0 = 14

EBX = 00000012 = 1\* 16^1 + 2 \* 16^0 = 18

**Exercise 3:** Write code, assemble and execute it, to store the sums of the corresponding elements of the first two arrays in the corresponding elements of the third array that was defined in Exercise 1. Make use of loops for addition. Display the sums stored in the array on the screen, reading directly from the memory. Print the following message before the results are displayed:

“2. The sums of the elements of the arrays are shown below:”

**How to see memory contents**

Mov esi, offset variablename // ESI,EDI,EBX can use here.

Mov ecx ,lengthof variablename

Mov ebx , type variablename

Call dumpmem

**Writing strings to display**:

Declare/define the string: mystring BYTE “How are you?”, 0

Get the offset of string into EDX: mov edx, offset mystring

Call the procedure to display: call **writestring**

(**Note** that writestring only works with EDX holding the offset)

**Program:**

INCLUDE Irvine32.inc

.data

arrayA WORD 2,3,4,5

arrayB WORD 3,4,5,6

arrayC WORD 4 DUP(?) ; uninitialized array

string BYTE "2. The sums of the elements of the arrays are shown below:",0

.code

main proc

mov edx,OFFSET string

call writestring

call crlf

mov ecx,4 ; execute loop 4 times

mov eax,0 ; get rid of garbage

mov ebx,0

mov esi,0 ; initialize esi

L1:

mov ax,arrayA[si] ; move elements of arrayA in ax

add ax,arrayB[si] ; add elements of arrayB in ax

mov arrayC[si],ax ; move sum to arrayC corresponding element

add si,2

loop L1

Mov esi, offset arrayC

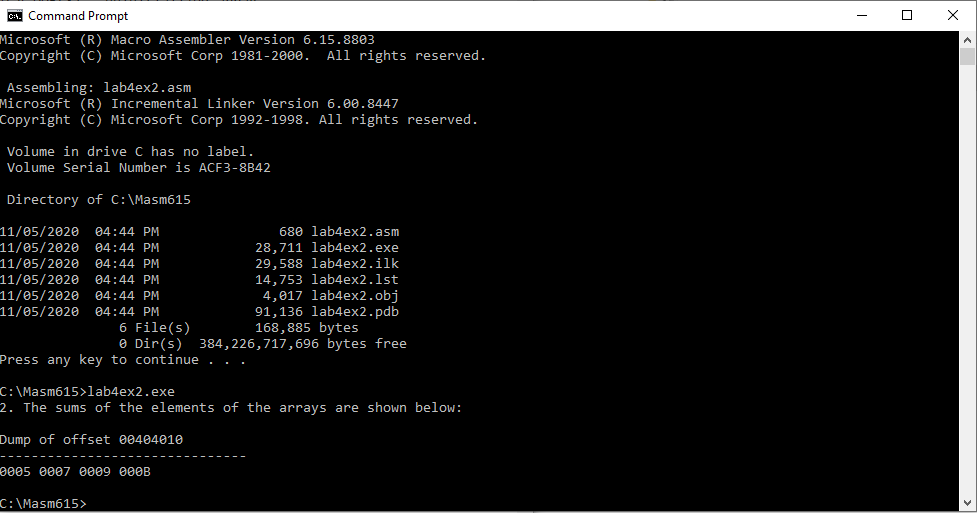
Mov ecx ,lengthof arrayC

Mov ebx , type arrayC

Call dumpmem

exit

main endp

end main