**x**

**Department of Electrical Engineering**

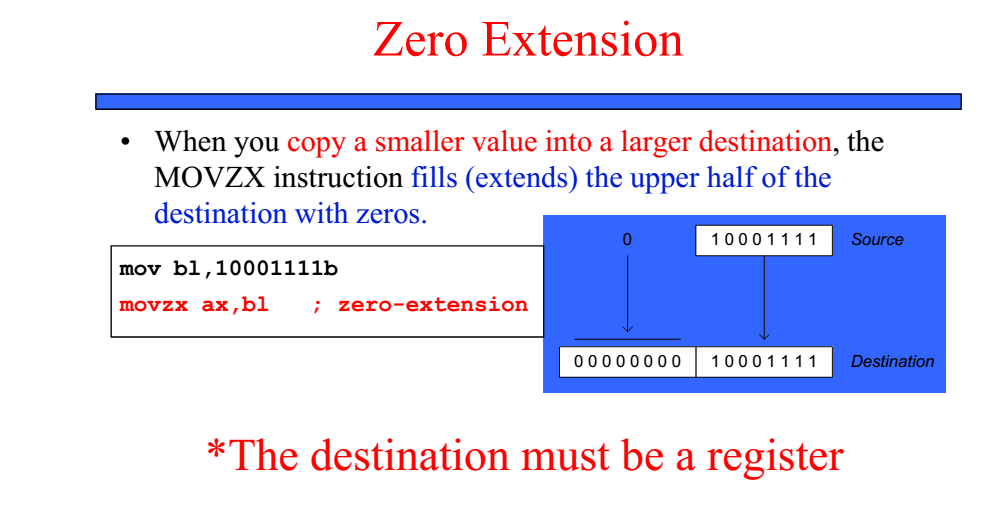
|  |  |
| --- | --- |
| **Faculty Member: Ma’am Qurat-ul-ain** | **Dated: October 29, 2020** |
|  |  |
| **Course/Section: BSCS-9B** | **Semester: 3rd** |
|  |  |

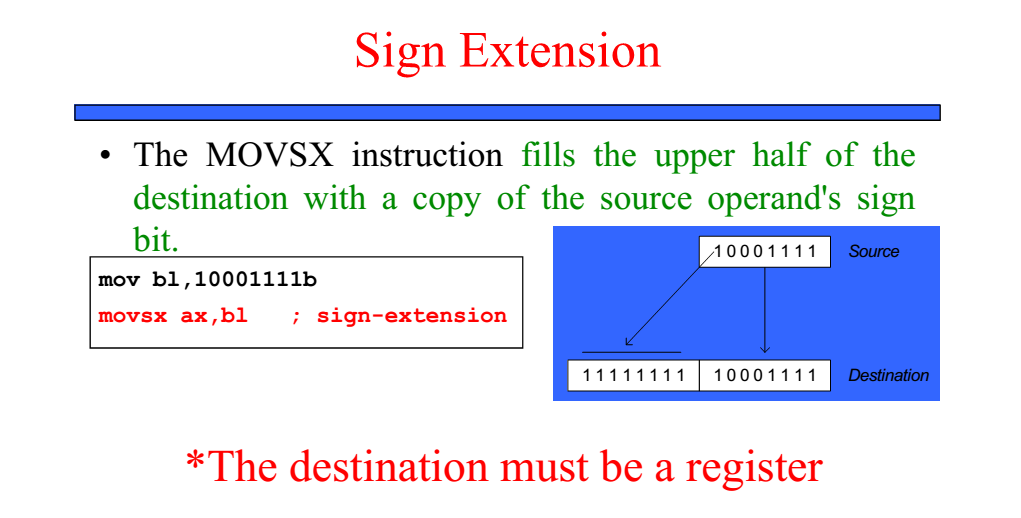
**CS-235 Computer Organization and Assembly Language**

**Lab #3Memory Access in Assembly Language**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Reg. no.** | **Report Marks / 8** | **Viva Marks / 7** | **Total/15** |
| **Fatima Seemab** | **291310** |  |  |  |
| **Mahum Samar** | **290647** |  |  |  |
| **Maryam Fatima** | **290479** |  |  |  |

**Objective:** The aim of this lab is to use some of the data transfer and manipulation instructions, and to use some assembler operators.





**Exercise 1:** In the memory list shown, insert the values of the variables as declared below, in hexadecimal format: (can be obtained from .data part of .lst file)

**Program**

|  |  |  |  |
| --- | --- | --- | --- |
| **offset** | **Content** | **Offset** | **Content** |
| 00 | 05 | 10 | 68 |
| 01 | 0C | 11 | 65 |
| 02 | 64 | 12 | 72 |
| 03 |  | 13 | 65 |
| 04 | 50 | 14 | 00 |
| 05 | 00 | 15 | 48 |
| 06 | 60 | 16 | 69 |
| 07 | 00 | 17 | 2C |
| 08 | 70 | 18 | 20 |
| 09 | 80 | 19 | 54 |
| 0A | 90 | 1A | 68 |
| 0B | A0 | 1B | 61 |
| 0C | 48 | 1C | 6E |
| 0D | 69 | 1D | 6B |
| 0E | 20 | 1E | 73 |
| 0F | 54 | 1F | 2E |
|  |  | 20 |  |

INCLUDE Irvine32.inc

.data

mbyte BYTE 05,12,100

Align 2

mword WORD 50h, 60h

mdouble DWORD 0A0908070h

greetings BYTE "Hi There",0

Response TEXTEQU <'Hi, Thanks.'>

Reply BYTE Response

.code

Main proc

; Use move instructions as per requirement

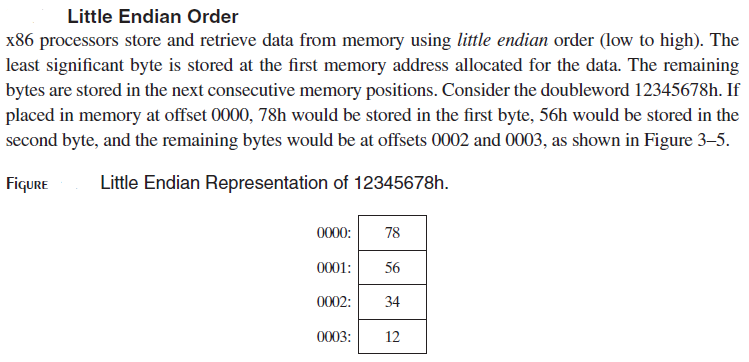
; Use call instructions as per requirement

exit

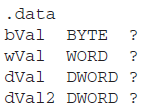
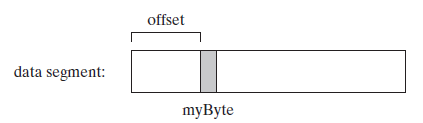
main endp

END main

Note: No ASCII table is to be used, wait to fill the text character codes after the exercise where the textstrings have been used, and the .lst file can be used to see these codes).**DO NOT FORGET**

****

         The **OFFSET** represents the distance, in bytes, from the beginning of the data segment. It tells about the location of a variable





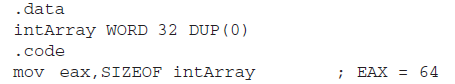
         The **TYPE** operator returns the size (in bytes) of an operand or of each element in an array. For example, the TYPE of a byte equals 1, the TYPE of a word equals 2

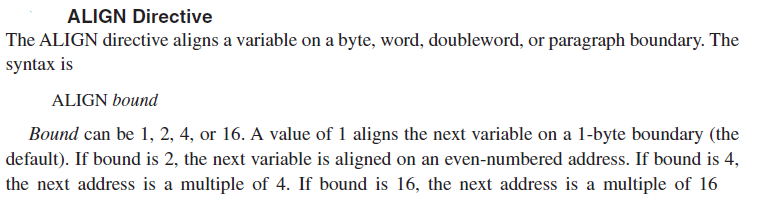
         The **LENGTHOF** operator returns the number of elements in an array.





         The **SIZEOF** operator returns the number of bytes used by an array initializer. The SIZEOF operator returns a value that is equivalent to multiplying LENGTHOF by TYPE



****

**Exercise 2:** Without writing any code, write down the expected contents of the register after the instruction is executed:

1. Mov al, mbyte-1 ; AL = 00
2. Movsx ax, mbyte+1 ; AX = 000C

**Exercise 3:** In this exercise we will learn about and use some new procedures that can be called to display register or memory contents.

Writing strings to display:

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

Get the offset of string into EDX: movedx, offset mystring

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

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

Writing register contents to display:

Get the contents to display in AL, AX, or EAX.

Ensure the bits not used are set/reset so as to improve readability.

Call the procedure to display: a. call **writeint** to print in decimal format

b. call **writehex** to print in hexadecimal format

c. call **writebin** to print in binary format

d. call **writechar** to print a character, the LSD of EAX

(**Note** that all these writexxx display EAX contents)

A call to **crlf** adds carriage return followed by a linefeed, e.g. call crlf.

**Part1:** Write code to get Byte No 2 of mbyte into AL and byte No 1 of mbyte into AH. Ensure that the higher order bits of EAX are cleared. Display EAX to verify that the correct bytes are in the locations specified. Use all four write procedures to see the various output formats, with a call to crlf after each writexxx to make the output easy to read.

(**Note** Use the data given in Exercise No. 1)

**Program**

INCLUDE Irvine32.inc

.data

mbyte BYTE 05,12,100

Align 2

mword WORD 50h, 60h

mdouble DWORD 0A0908070h

greetings BYTE "Hi There",0

Response TEXTEQU <'Hi, Thanks.'>

Reply BYTE Response

.code

main proc

Movzx eax,mbyte+1 ; moving the value of 2nd byte into AL

Mov ah,mbyte ; moving the value of 1st byte into AH

call dumpregs

call writeint ; To print in decimal format

call crlf

call writehex ; To print in hexadecimal format

call crlf

call writebin ; To print in binary format

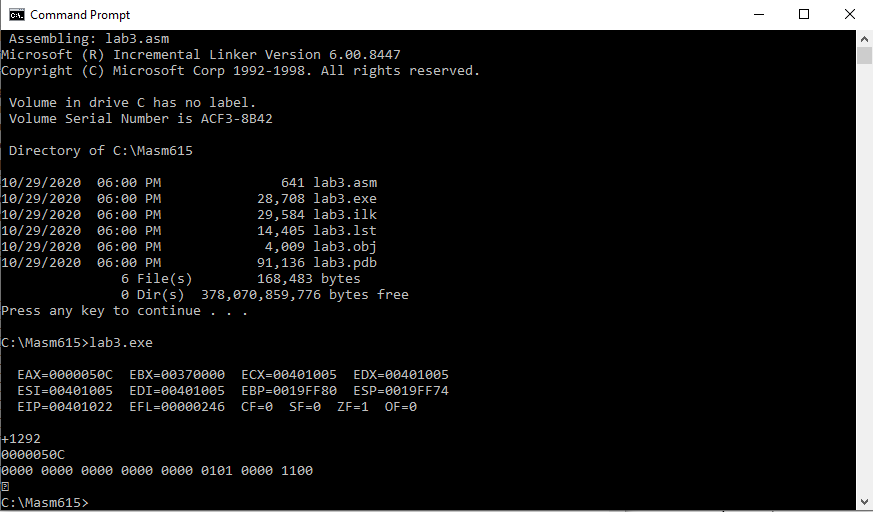
call crlf

call writechar ; To print a character, the LSD of EAX

exit

main endp

END main



Note down the outputs below:

1. EAX= +1292 ( int )
2. EAX= 0000050C ( hexadecimal )
3. EAX= 0000 0000 0000 0000 0000 0101 0000 1100 ( binary )
4. ------=

**Find the Explanation, what is (d):**

**writechar** method prints a character, the LSD of EAX i.e. 0C here, which corresponds to **NP Form Feed, new page** in ASCII table. Since no symbol is associated to new page command so it was presented in output as a question mark.

**part2:** Extend program to display the length and size of the string variable “greetings” in decimal format, and then print the first string

Hint: Use keyword sizeof, lengthof to display the length and size in register

**Program**

INCLUDE Irvine32.inc

.data

mbyte BYTE 05,12,100

Align 2

mword WORD 50h, 60h

mdouble DWORD 0A0908070h

greetings BYTE "Hi There",0

Response TEXTEQU <'Hi, Thanks.'>

Reply BYTE Response

.code

main proc

mov eax, LENGTHOF greetings ; move length to eax

mov ebx, SIZEOF greetings ; move size to ebx

Call dumpregs

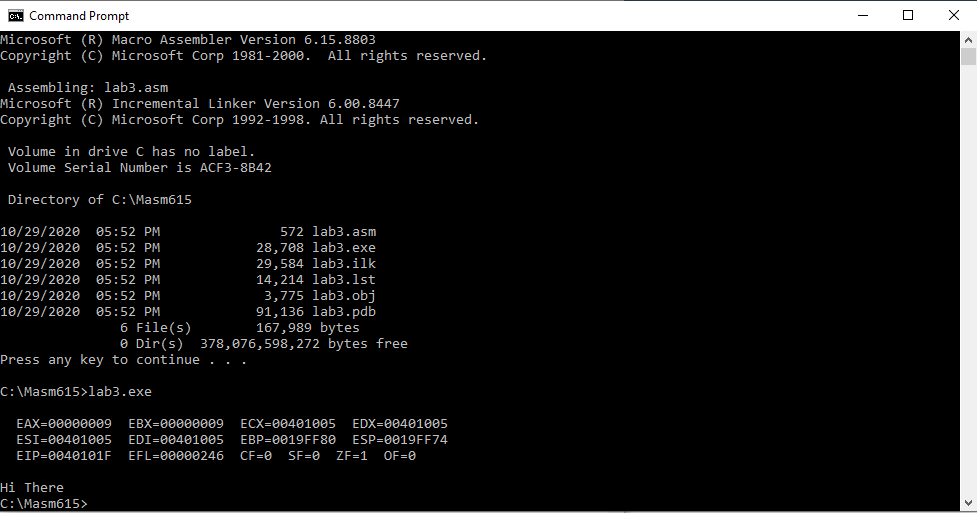
mov edx, offset greetings ; Get the offset of string into EDX

Call writestring ; Call the procedure to display

exit

main endp

END main



**part3:** Display the second string defined by the TEXTEQU operator

**Program**

INCLUDE Irvine32.inc

.data

mbyte BYTE 05,12,100

Align 2

mword WORD 50h, 60h

mdouble DWORD 0A0908070h

greetings BYTE "Hi There",0

Response TEXTEQU <'Hi, Thanks.'>

Reply BYTE Response

.code

main proc

Call crlf

mov edx, offset Reply ; Get the offset of string into EDX

Call writestring ; Call the procedure to display

exit

main endp

END main

