National University of Computer and Emerging Sciences

Laboratory Manual

for

Computer Organization and Assembly Language Programming

Course Instructor	Ms. Samin Iftikhar
Lab Instructor(s)	Amna Sehar Rida Mahmood
Semester	Fall 2023

Department of Computer Science

FAST-NU, Lahore, Pakistan

Objectives

After performing this lab, students shall be able to:

- ✔ Learn different types of addressing modes
- ✔ Differentiate between code data segment
- ✔ Declare and use variables in assembly language

- ✔ Segmented Memory Models

Addressing Modes

1. Register Addressing Mode:
Register operands are the easiest to understand. Consider the following forms of the MOV instruction:
MOV AX, AX
MOV AX, BX
MOV AL , BL

2.*Immediate Addressing Mode*:
Constants are also pretty easy to deal with. Consider the following instructions:
MOV AX, 0x25
MOV DX, 1000

3.*Direct Addressing Mode*:
There are three addressing modes which deal with accessing data in memory. These addressing modes take the following forms:
MOV AX, [1000]
The first instruction above uses the direct addressing mode to load AX with the 16 bit value stored in memory starting at location 1000.

4.*Indirect Addressing Mode*:
MOV AX, [BX]
The MOV AX, [BX] instruction loads AX from the memory location specified by the contents of the BX register. This is an *indirect* addressing mode. Rather than using the value in BX, this instruction accesses to the memory location whose address appears in BX. Note that the following two instructions:
MOV BX, 1000
MOV AX, [BX]
are equivalent to the following single instruction:
MOV AX, [1000]
Of course, the second sequence is preferable. However, there are many cases where the use of indirection is faster, shorter, and better.

5. ***Indexed Addressing Mode*:** The last memory addressing mode is the *indexed* addressing mode with the base register. An example of this memory addressing mode is

MOV AX, [1000+BX]
This instruction adds the contents of BX with 1000 to produce the address of the memory value to fetch. This instruction is useful for accessing elements of arrays, records, and other data structures.

Data Types

Variables are declared in memory.

DB	Define Byte	allocates 1 byte (0 – (2^8 – 1))
DW	Define Word	allocates 2 bytes (0 – (2^{16} – 1))
DD	Define Doubleword	allocates 4 bytes (0 – (2^{32} – 1))
DQ	Define Quadword	allocates 8 bytes (0 – (2^{64} – 1))
DT	Define Ten Bytes	allocates 10 bytes (0 – (2^{80} – 1))

Example:

```
; a program to add three numbers using memory variables
[org 0x0100]

mov ax, [num1]                ;load first number in ax
mov bx, [num2]               ; load second number in bx
add ax, bx                    ; accumulate sum in ax
mov [result1], ax            ; store result in result1 variable, 15

mov ax, 0x4c00               ; terminate program
int 0x21

num1: dw 5            ;variables
num2: dw 10

result1: dw
```

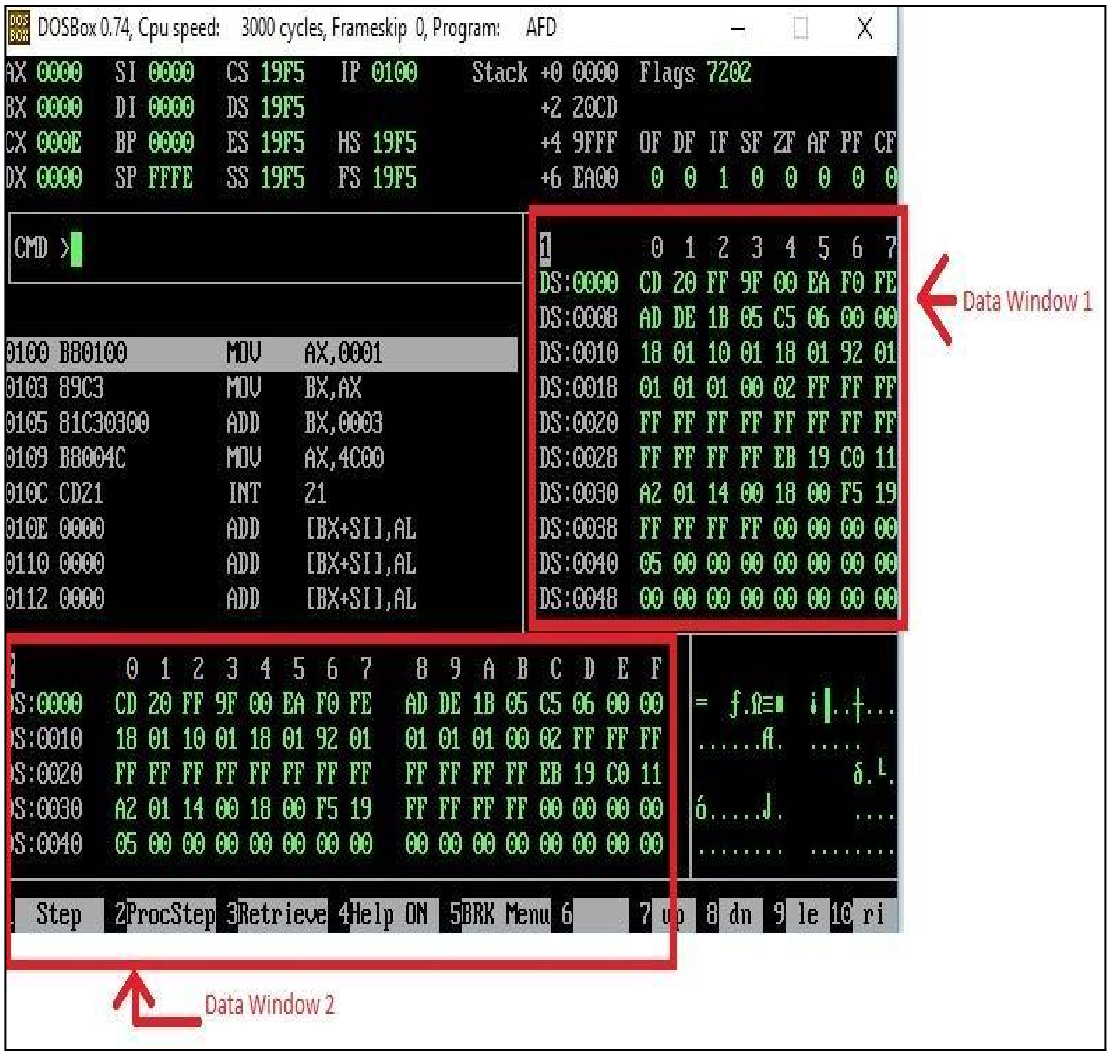
Code Segment

Pointed by CS

Data Segment

Pointed by DS or ES

How to View Memory in AFD


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD
Data Window 1
Data Window 2

```
AX 0000  SI 0000  CS 19F5  IP 0100   Stack +0 0000  Flags 7202
BX 0000  DI 0000  DS 19F5                  +2 20CD
CX 000E  BP 0000  ES 19F5  HS 19F5         +4 9FFF  OF DF IF SF ZF AF PF CF
DX 0000  SP FFFE  SS 19F5  FS 19F5         +6 EA00   0  0  1  0  0  0  0  0

CMD >                              1          0  1  2  3  4  5  6  7
                                   DS:0000   CD 20 FF 9F 00 EA F0 FE
                                   DS:0008   AD DE 1B 05 C5 06 00 00
0100 B80100       MOV   AX,0001    DS:0010   18 01 10 01 18 01 92 01
0103 89C3         MOV   BX,AX      DS:0018   01 01 01 00 02 FF FF FF
0105 81C30300     ADD   BX,0003    DS:0020   FF FF FF FF FF FF FF FF
0109 B8004C       MOV   AX,4C00    DS:0028   FF FF FF FF EB 19 C0 11
010C CD21         INT   21         DS:0030   A2 01 14 00 18 00 F5 19
010E 0000         ADD   [BX+SI],AL DS:0038   FF FF FF FF 00 00 00 00
0110 0000         ADD   [BX+SI],AL DS:0040   05 00 00 00 00 00 00 00
0112 0000         ADD   [BX+SI],AL DS:0048   00 00 00 00 00 00 00 00

2          0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
DS:0000   CD 20 FF 9F 00 EA F0 FE  AD DE 1B 05 C5 06 00 00   = ƒ.Ω=■  ↓|..┼...
DS:0010   18 01 10 01 18 01 92 01  01 01 01 00 02 FF FF FF   ......Æ.  .....
DS:0020   FF FF FF FF FF FF FF FF  FF FF FF FF EB 19 C0 11              δ.└.
DS:0030   A2 01 14 00 18 00 F5 19  FF FF FF FF 00 00 00 00   ó.....J.      ....
DS:0040   05 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   .......  .......

 Step  2ProcStep 3Retrieve 4Help ON 5BRK Menu 6      7 up  8 dn  9 le 10 ri
```

In above screenshot, there are two data windows, each window is showing the contents of Memory. Such as at Offset 0000, we can see that the data is CD and at Offset 0001, the data is 20.
If you want to see the data at offset 0040, simply write m1 DS:0040 or m2 DS:0040 on AFD console. (m1 is for window 1, and m2 is for window 2).

If you want to check your declared memory content, you have to create listing file of your program, then note offset of your data label from listing file, and then simply write m1 DS:offset, then you will see your data label content in m1 window. Offset value will be calculated after the addition of 0x0100 in the instruction address displayed by listing window since we ask assembler to start writing machine code from the address 0x0100 using the first line of code: [org 0x0100]

Practice Exercise:

Write a program which fulfills the following requirements: AX = 200h BX=150h Memory location 250 =50h Memory Location 200= 25h Array = {1,2,7,5,10}

a) Load AX with contents of memory location 200 using indirect addressing

b) Load CX with contents of memory location 250 using direct addressing.

For further understanding of registers addressing, follow Bilal Hashmi Book Chapter 2.

QUESTIONS

Exercise 1: (5) Write an assembly language program which fulfills the following:

a) Load 25h to Ax register

b) Swap contents of Ax and Bx

c) Load the contents of memory location [0x270] in Bx

d) Define an array of num = [12,25,10] and load the contents of array in Ax using offset addressing. [hint: array is defined as: num: db 1, 2 at the end of code]

Exercise 2: (10) Make two Byte type arrays named array1 and array2 and one Byte Type array named array3, each containing only 5 elements such that following rules are satisfied for each element of all the arrays:

Array1 [i+1] = Array1 [i] + Array3 [i]

Array2 [i+1] = Array3 [i] - Array2 [i]

Array3 [i] = Array1 [i] + Array2 [i]

Here "i" denotes the element number in each array and it will range from 0 to 4 for five elements. Use BX as your Offset Register. Don't use loops for this question

You have to use two Initial Values: Array1 [0]= 0 and Array2[0]=1

The final arrays should look like this:

Array1: 0, 1, 2, 5, 12

Array2: 1, 0, 1, 2, 5

Array3: 1, 1, 3, 7, 17

Exercise 3:(5) Write instructions to do the following. Visualize the memory contents using memory windows to see if instruction is executed correctly. (Use m2 DS:offset to visualize the memory contents at the specified offset)

a. Copy contents of memory location with offset 0025 into AX.

b. Copy AX into memory location with offset 0FFF.

c. Move contents of memory location with offset 0010 to memory location with offset 002F.

Exercise 4:(5)

Give the value of the zero flag, the carry flag, the sign flag, and the overflow flag after each of the following instructions:

	ZF	CF	SF	OF

mov ax, 0x1254				
mov bx, 0x0FFF				
add ax, 0xEDAB				
add ax, bx				
add bx, 0xF001				