National University of Computer and Emerging Sciences

Lab Manual

Computer Organization and Assembly Language

Lab 02

Fast School of Computing

FAST-NU, Lahore, Pakistan

Objectives:

- How to modify the memory contents
- How to use various addressing modes
- How to use the registers associated with memory
- How to use segmentation

Contents

Objectives
....2
Addressing Modes [Theory]... 2
ACTIVITY 1:.. 3
ACTIVITY 2:.. 3
ACTIVITY 3... 4
ACTIVITY 4:.. 4
REFERENCES.. 4

Addressing Modes [Theory]

The x86 processors support the register addressing mode, the immediate addressing mode, the indirect addressing mode, the indexed addressing mode, and the direct addressing mode. The following paragraphs explain each of these modes:

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 Ax, Cx
MOV Ax, Dx
```

The first instruction accomplishes absolutely nothing. It copies the value from the Ax register back into the Ax register. The remaining three instructions copy the value of Bx, Cx and Dx into Ax. Note that the original values of Bx, Cx, and Dx remain the same. The first operand (the destination) is not limited to Ax; you can move values to any of these registers.

Immediate Addressing Mode: Constants are also easy to deal with. Consider the following instructions:

```
MOV Ax, 25H
MOV Bx, 195H
MOV Cx, 2056H
MOV Dx, 1000H
```

These instructions are all straightforward; they load their respective registers with the specified hexadecimal constant.

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 hex.

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 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.

Register + Offset Addressing:

A base register is used with a constant offset in this addressing mode. The value contained in the base register is added with the constant offset to get the effective address. For example, "mov [Bx+300], Ax" stores the word contained in Ax at the offset attained by adding 300 to Bx in the current data segment.

Indexed Addressing Mode: The last memory addressing mode is the indexed addressing mode.

An example of this memory addressing mode is

MOV Ax, [1000+SI]

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.

ACTIVITY 1:

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 Cx.

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

ACTIVITY 2:

a) Write a program which adds five numbers using memory variables n1, n2, n3, n4, n5. Initialize the memory variables to 10, 20, 30, 40, 50. Make a new variable, sum, which stores the sum of all.

b) Modify the program in part a using a single memory label, n, for inputs and result.

ACTIVITY 3

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.

ACTIVITY 4:

Write a program which adds the contents of the following array using register + offset addressing. **Array** *dw* **111. 999, 888, 888, 11, 99, 88, 88, 1, 9, 8, 8**

REFERENCES

- "http://www.dosbox.com/download.php?main=1
- http://sourceforge.net/projects/nasm
- http://www.nasm.us/
- http://www.programmersheaven.com/download/21643/download.aspx (AFD)