National University of Computer and Emerging Sciences

Lab Experiment 02
Computer Organization and Assembly Language Lab

Department of Computer Science

FAST-NU, Lahore, Pakistan

MANUAL-2: ADDRESSING MODES

OBJECTIVES:

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

Addressing Modes

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 table explain each of these modes:

Addressing Mode	Definition	Example
Register Addressing	Uses a register as the operand, either source or destination. The operation is performed directly on the data in the register.	MOV AX, BX MOV CX, DX
Immediate Addressing	Uses a constant value or immediate data as the operand. The operand is directly specified in the instruction.	MOV AX, 25H MOV BX, 195H
Direct Addressing	Specifies the exact memory address to access data. The operand is the memory address directly given in the instruction.	MOV AX, [1000H]
Indirect Addressing	Uses a register to specify the memory address where the data is located. The operand is accessed indirectly through the register's value.	MOV AX, [BX] MOV AX, [1000H + BX]
Indexed Addressing	Combines a base address and an index to access memory. Often used for accessing elements of arrays or data structures.	MOV AX, [1000H + BX]

Question #1:

a) Load 25h to Ax register and 10h to Bx using immediate addressing mode.
b) Swap the contents of Ax and Bx using register addressing.
c) Load 1234h as the contents of memory location 0x270 using indirect addressing mode.
d) Define an array of **Num: dw** 12,25,38,44,105 , Only using the given command MOV Ax, [100 +Bx] , load the contents of Num to AX one after another.

Question #2:

The following contents are to be placed at their respective memory locations and registers as shown in the table below:

AX	**BX**	**200h**	**250h**
200h	250h	25h	15h

a) Load AX with contents of memory location 200 using indirect addressing with offset.
b) Load CX with contents of memory location 250 using direct addressing.
c) Switch the Contents of memory location 200 and 250 using appropriate addressing method.

Question #3:**A)** Run the following code and see the changes in registers write the values of ax, al and ah after each line.

```
[org 0x0100]

;code

Mov ax, [num1] ;ax=?

Mov ax, [num2] ;ax=?

Mov ax, [num2+2] ;ax=?

Mov ax, [num2+1] ;ax=?

Mov al, [num2+3] ;ax=?

Mov ah, [num1] ;ax=?

Mov ax, [array1] ;ax=?

Mov ax, [array1+2] ;ax=?
Mov al, [array2] ;ax=?

Mov al, [array2+1] ;ax=?

Mov ax, [array2] ;ax=?

mov ax, 0x4c00 ; termination statements

int 21h

; data

Num1: dw 0A0Bh

Num2: dd 0C0D0E0Dh

Array1: dw 0102h , 0304h

Array2: db 05h , 06h, 07h
```

B)Run the following code and see the changes in memory (in labels you declared)

```
[org 0x0100]

;code

Mov ax, 9876h

Mov bx, 5432h

Mov [num1], ax

Mov [num2], bx

Mov [num2+2], bx

Mov [array1], ax

Mov [array2], bl

Mov [array2], ax

Mov word [num1], 0000h

Mov byte [num1], 01h

Mov byte [num2+1], 11h

Mov word [array1+2], 3870h

mov ax, 0x4c00 ; termination statements

int 21h

; data
```

```
Num1: dw 0A0Bh

Num2: dd 0C0D0E0Dh

Array1: dw 0102h , 0304h

Array2: db 05h , 06h, 07h
```

Question #4:

Write a program that reads array1 and saves it in array2 in reverse order. Sample run is given below.

Array1: 1, 2, 3, 4, 5, 6

Array2: 0,0,0,0,0,0

After Program Execution:

Array1: 1, 2, 3, 4, 5, 6

Array2: 6, 5, 4, 3, 2, 1

Help: You can and cannot use JNZ and Loop. You may need SI and DI (Source Index and Destination Index) registers to save two different indices.