মেট্রোপলিটন ইউনিভার্সিটি
Metropolitan University

Department of Computer Science and Engineering

Course Code: CSE238	Credits: 1.5
Course Name: Microprocessor & Interfacing Lab	Faculty: FRS

Lab 01

Introduction to Registers, Arithmetic Operations, Instruction: MOV

Activity Detail:

Discussion: Registers 16-bit storage for holding data temporarily during operations. Do not confuse with RAM. Registers can be divided into 4 types.

i. *Data Registers:* AX, BX, CX, DX These 4 registers are used to hold data for carrying out mathematical and logical operations. These 4 registers are byte accessible. This means each 16-bit register can be used as two separate8-bit registers. AX = AH and AL, same goes for others. These four registers also perform other special functions. AX = During multiplication and division, one of the numbers must be inside AX or AL. BX = Used as address registers. CX = used in loop. CX increases automatically by 1. DX = Used in multiplication/division and i/o operations.

ii. *Segment Registers:* CS, DS, SS, ES To understand the these registers we must understand what memory segments are. The 8086 is a 16-bit Microprocessor and contains a physical memory of 20 bits. Now to store a value of 10 bits the registers had to be of 20 bits each but they are not. To overcome this problem the whole memory is divided into a 3 segments data segment, stack segment and code segment. The segment registers will contain the address of the segments. Each segment register will be combined with index registers (16 bit) to form

a 20-bit storage to store memory locations. [Page 94 - Microprocessors & Interfacing].

iii. *Index and Pointer Registers:* SP, BP, SI, DI They contain the offset addresses mentioned in the previous section. There are restrictions of using a segment register and index registers.

SP used with SS
BP used with DS, SS
SI used with DS
DI used with DS.

iv. *Data Types:*

1. Numbers: Must specify the base of the number at the end. If not specified the numbers will be considered as decimal.
   For binary, add a 'b' at the end of the bit-string.
   e.g. 1110b
   For decimal, just type the number without any suffix or prefix. e.g. 1101
   For hexadecimal, start with a 0 and end with 'h'.
   e.g. A06 = invalid, 0A06h = valid
2. Characters: Enclosed by single quotes. For example, 'a'. The assembler converts every character into its ASCII value while storing
3. Strings: Array of character ends with a $ and enclosed by double quotes.
   e.g. "hjshj$"
4. Data types: DB = define byte, DW = define word
5. Variables: In java the syntax for defining a variable is,
   data_typevar_name; /= value;
   In assembly it is,
   var_name data_type value.
   e.g.: m db 4
   Variables are saved in the data segment of the memory.

v. *MOV:* Used to transfer data between registers; to a register from a memory location, from a memory location to register. Moving constants to register or memory locations.

Syntax: MOV Destination, Source
Example: MOV AL , 5

The decimal value 5 will be stored in AL.

	Destination			
Source	**General registers**	**Segment Register**	**Memory Location**	**Constants**
General registers	**valid**	**valid**	**valid**	**invalid**
Segment Registers	**valid**	**invalid**	**valid**	**invalid**
Memory Location	**valid**	**valid**	**invalid**	**invalid**
Constant	**valid**	**invalid**	**valid**	**invalid**

vi. *ADD/SUB:* Addition/Subtraction between numbers. These operations occur between 2 registers, register/memory location and a number, register and memory location.

Syntax: ADD/SUB Destination, Source
i.e. Destination = Destination +/ - Source

	Destination	
Source	**General registers**	**Memory Location**
General registers	**valid**	**valid**
Memory Location	**valid**	**invalid**
Constant	**valid**	**valid**

vii. *INC:* This is used to increment a value of a register by 1.

Syntex: INC AX; // this would increase the value of AX by 1

viii. *DEC:* This is used to decrement a value of a register by 1.

Syntex: DEC AX; // this would decrease the value of AX by 1

ix. *NEG:* This is used to negate the contents of the destination. NEG does this by replacing the contents by its two's. complement.

Syntex: NEG Destination; // this would increase the value of AX by 1

Problem Solving

Task 01

Take input in the register AX, and then move it to BX using the MOV instruction.

Task 02

Swap two numbers, using a maximum of 3 registers.

Hint: Use the MOV instruction.

Task 03

Add two numbers using two registers.

Task 04

Subtract two numbers using two registers. Do you always get the correct answer? What happens when you subtract larger number from the smaller one?

Task 05

Swap two numbers using ADD/SUB instructions only.

Task 06

Perform the following arithmetic instructions. A, B, C are three variables to be declared beforehand.

1. A = B - A
2. A = -(A + 1)
3. C = A + (B + 1); Use INC
4. A = B – (A – 1); Use DEC

Code Template

```
.MODEL SMALL

.STACK 100H

.DATA


.CODE
MAIN PROC

;iniitialize DS

MOV AX,@DATA
MOV DS,AX

;enter your code here









;exit to DOS

MOV AX,4C00H
INT 21H

MAIN ENDP
END MAIN
```