BRAC UNIVERSITY
Inspiring Excellence

Department of Computer Science and Engineering

Course Code: CSE341	Credits: 1.5
Course Name: Microprocessors	Semester: Fall 21

Lab 02

Registers, Arithmetic Operations, Instruction: MOV

I. Topic Overview:

The lab is designed to introduce the students to get the basic idea on the registers involved in an Intel 8086 processor. In other words, students will be taught about how the registers can be used as variables and be introduced to some of the registers' special features. Furthermore, on successfully understanding how a register works, they will learn to perform basic arithmetic operations using those registers. Lastly, they will be introduced to an Intel 8086 instructions which is called 'MOV'.

II. Lesson Fit:

In order to do the lab with ease, the student must have completed Lab 1 of the CSE341 Course, which was 'Introduction'. The lab was designed to introduce the students with the simulator and its functions.

III. Learning Outcome:

After this lecture, the students will be able to:

a. know about the registers and their functions in an Intel 8086 processor.
b. perform arithmetic operations using the registers.
c. use the 'MOV' instruction in their assembly language coding.

IV. Anticipated Challenges and Possible Solutions

a. Use of 8 bit and 16 bit registers together during arithmetic operations

Solutions:

i. Mention of keeping the registers bit size fixed.

ii. Give them examples

V. Acceptance and Evaluation

If a task is a continuing task and one couldn't finish within time limit, then he will continue from there in the next Lab, and if it is a one Lab task then it will be given as a home work and in the next Lab you have to submit the code and have to face a short viva. A deduction of 30% marks is applicable for late submission. The marks distribution is as follows:

Code: 50%

Viva: 50%

VI. Activity Detail

a. **Hour: 1**

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 1DX = 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 memory of 20 bits. Now to store a value of10 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. [The combination of 2 16-bit registers to form a 20-bit location will be covered in theory class].

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

b. **Hour: 2**

Discussion: Instructions (mov, add, sub, inc, dec, neg)

i. **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

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**

ii. **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**

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

inc ax; // this would increase the value of ax by 1.

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

dec ax; // this would decrease the value of ax by 1.

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

NEG destination

c. **Hour: 3**

Discussion: Instructions (mul. div)

i. **MUL:** multiplication between 2 numbers. You must be very careful while carrying out multiplication. This is because 2 n bit numbers will result a 2n bit number. Therefore, multiplication is divided into2branches. byte multiplication where the operands are 8-bit numbers and word multiplication where the numbers are 16-bit numbers.

For byte multiplication one number is contained in the source and the other number has to be in AL. The result is saved in AX. The source can be a register or a memory location but not a constant.

mul BL; the 8-bit number is AL is multiplied by the 8-bit number in BL. The result is in AX.

For word multiplication one number is contained in the source and the other number has to be in AX. The result will be a 32-bit number. The higher 16 bits will be in DX and the lower 16-bit will be in AX. The source can be a register or a memory location but not a constant.

ii. **DIV:** When division is performed there are 2 results- quotient and remainder. Division is also divided into byte and word form. In the byte form the divisor is an 8 bit register or memory location. The dividend is 16 bit and it must be in AX. After division the 8-bitquotient is in AL and the
8-bit remainder is in AH. The divisor can't be a constant.
div BL; the 16-bit number is AX is divided by the 8-bit number I n BL. The quotient is in AL and the 8-bit remainder is in AH.
In the word form, the divisor is a 16-bit register or memory location. The dividend is 32 bit and it must be in DX:AX. After division the 16-bit quotient is in AX and the 16-bit remainder is in DX. The divisor can't be a constant.