**Lab1**  
Assembly language program deals directly with internal structure of processor, so a AL programmer must know about the internal structure of a processor or in other words knows about the processor architecture or the component of the processor that a programmer deals with them.

**Processor architecture:**

1. Register: Memory inside processor. Programmer must knows about name, size their formats

Like AX, BX, CX etc..

1. Instruction set: A programmer must know name, format and their working. Like ADD, SUB etc. Complex task A= B + 5\*C is performed by the basic instruction ADD, SUB etc,

**Categories of instructions:**

1. Data movement (mov or load)
2. Arithmetic and logical instruction(ADD, SUB, AND)
3. Flow control instruction(CALL, RET, JMP,JNZ)
4. Special instruction(STI,CLI)

**General format:**

1. Inst dst, src…..eg ADD AX, BX
2. Inst dst……….INC AX, POP BX
3. Inst src…… PUSH BX
4. Inst…… STD, CLD

**Program:**

Prove:

1. 1 to 1 correspondence bw AL and ML… see it in debugger
2. Sequence in program is maintained by IP …see it in debugger
3. Show Machine code

Assembly language is short hand of English.

Write a AL program to add there number like 5+10+15 by using variables

**C code:**

Ax = 5

BX = 10

AX = AX + BX

BX = 15

AX = AX + BX

**Pseducode:**

MOV 5 to AX

Mov 10 to BX

Add bx to ax

Mov 15 to bx

Add bx to ax

**AL CODE**

.MODEL SMALL

.STACK 100H

.CODE

MAIN PROC

MOV AX, 5

MOV BX, 10

ADD AX, BX

MOV BX, 15

ADD AX, BX

MOV AH, 4CH

INT 21H

MAIN END

END MAIN

**Program life cycle: See… chapter 4**

1. EDITOR -> .ASM FILE eg. Prog1.asm by using notepad
2. ASSEMBLER -> Prog1.obj by using masm p1.asm
3. LINKER -> Prog1.exe by by using link p1

**Commands** to edit, assembly and run a program

C:\> edit p1.asm

C:\> masm p1.asm

C:\> link p1.asm

C:\> p1

**Commands** to debug a program

C:\> debug p1.exe

-R (Show registers)

-t(trace a program)….show the role of IP

**PATH SETTING** ….

SYSTEM PROPERTIES…. ENIRONMENT VARIABLE

PATH : C:\MASM\BIN; C:\MASM\BINR;