Assembly Lab - 01

DIRECTIVES AND SAMPLE PROGRAMS

- ***Directives*** are statements that give directions to the assembler about how it should translate the assembly language instructions into machine code.

- An assembly language instruction consists of four fields,

  [**label:**] **mnemonic** [**operands**] [**;comments**]

  Brackets indicate that the field is optional. Brackets are not typed.

1. The label field allows the program to refer to a line of code by name.
2. In a line of assembly language program there can be mnemonic (instruction) and operand(s).

```
Ex:     ADD  AL,BL
        MOV  AX,6764H
```

3. Alternatively, instead of these two fields there can be directives. Directives are used by the assembler to organize the program as well as other output files.
4. The comment field begins with a “;”

MODELS

- **SMALL MODEL** (.MODEL SMALL): The model uses maximum of ***64K bytes for Code*** and ***64K bytes for Data*** ***(Code<=64K and Data <=64K)***. This model is the most widely used memory model and is sufficient for all the programs to be used in this course.

- **MEDIUM MODEL**, (.MODEL MEDIUM): The model uses maximum of ***64K bytes for Data*** and Code can exceed ***64K bytes*** ***(Code>64K and Data <=64K)***.

- **COMPACT MODEL**, (.MODEL COMPACT): The model uses maximum of ***64K bytes for Code*** and Data can exceed ***64K bytes*** ***(Code<=64K and Data >64K)***.

- **LARGE MODEL**, (.MODEL LARGE): Both Code and Data can exceed ***64K bytes. However no single data set (i.e. array) can exceed 64K bytes*** ***(Code>64K and Data >64K)***.

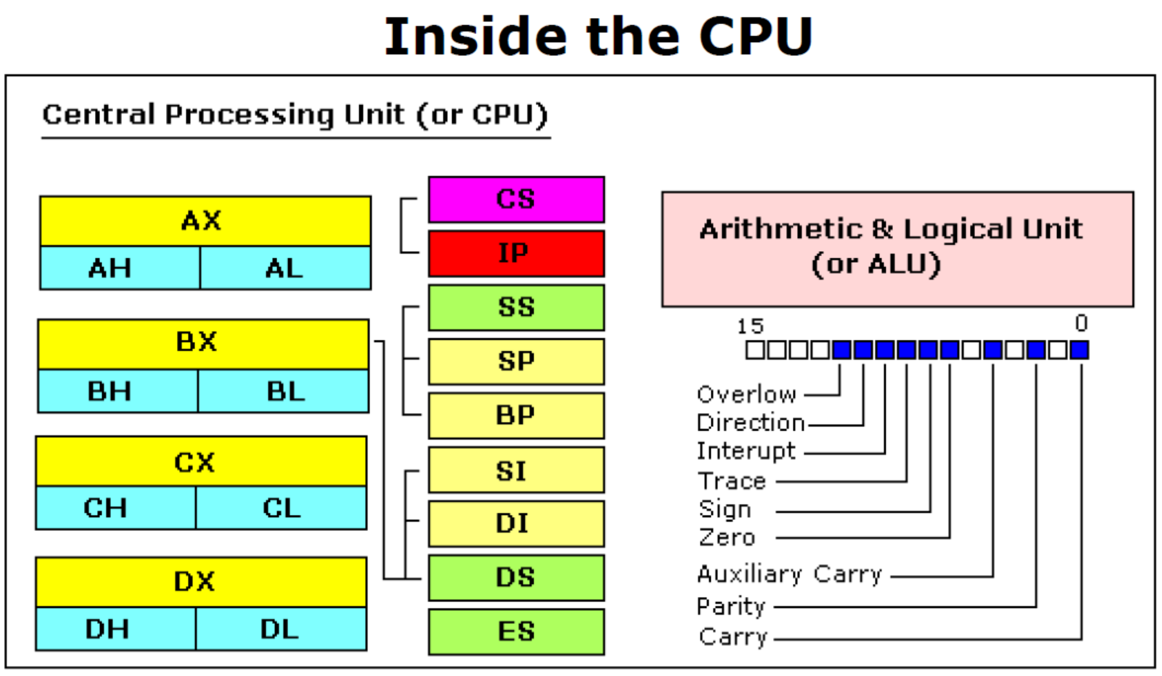
- **HUGE MODEL**, (.MODEL HUGE): Both Code and Data can exceed ***64K bytes. Additionally, a single data set (i.e. array) can exceed 64K bytes*** ***(Code>64K and Data >64K)***

A Sample program to Add two number

```
.MODEL SMALL                    ;Gives the memory model to be used by the program
.STACK 64
;---------------------------------------------------
.DATA
        DATA1        DB     52H
        DATA2        DB     29H
        SUM          DB     ?
;---------------------------------------------------
.CODE
MAIN PROC
        MOV AX,@DATA
        MOV DS,AX               ;assign value to DS
        MOV AL,DATA1            ;get the first operand
        MOV BL,DATA2            ;get the second operand
        ADD AL,BL               ;add the operands
        MOV SUM,AL              ;store result in location SUM
        MOV AH,4CH              ;set up to
        INT 21H                 ;return to the Operating System (DOS)
        MAIN ENDP
        END        MAIN         ;this is the program exit point
```

Theory:

GENERAL PURPOSE REGISTERS



- **AX** - the accumulator register (divided into **AH / AL**).
- **BX** - the base address register (divided into **BH / BL**).
- **CX** - the count register (divided into **CH / CL**).
- **DX** - the data register (divided into **DH / DL**).
- **SI** - source index register.
- **DI** - destination index register.
- **BP** - base pointer.
- **SP** - stack pointer.

SEGMENT REGISTERS

- **CS** - points at the segment containing the current program.
- **DS** - generally points at segment where variables are defined.
- **ES** - extra segment register, it's up to a coder to define its usage.
- **SS** - points at the segment containing the stack