# ASSEMBLER DIRECTIVES / PSEUDO OPCODES

Assembly language has 2 types of statements:

1. **Executable**: Instructions that are translated into Machine Code by the assembler.

2. **Assembler Directives**:
   Statements that direct the assembler to do some special task.
   No M/C language code is produced for these statements.
   Their main task is to inform the assembler about the start/end of a segment, procedure or program, to reserve appropriate space for data storage etc.
   Some of the assembler directives are listed below

**1. DB** (Define Byte) ; Used to define a Byte type variable.
Eg: SUM DB 0 ; Assembler reserves 1 Byte of memory for the variable
; named SUM and initialize it to 0.

**2. DW** (Define Word) ; Used to define a Word type variable (2 Bytes).

**3. DD** (Double Word) ; Used to define a Double Word type variable (4 Bytes).

**4. DQ** (Quad Word) ; Used to define a Quad Word type variable (8 Bytes).

**5. DT** (Ten Bytes) ; Used to define 10 Bytes to a variable (10 Bytes).

**6. DUP()** ; Copies the contents of the bracket followed by this
; keyword into the memory location specified before it.
Eg: LIST DB 10 DUP (0) ; Stores LIST as a series of 10 bytes initialized to Zero.

**7. SEGMENT** ; Used to indicate the beginning of a segment.

**8. ENDS** ; Used to indicate the end of a segment.

**9. ASSUME** ; Associates a logical segment with a processor segment.
Eg: Assume CS:Code ; Makes the segment "Code" the actual Code Segment.

**10. PROC** ; Used to indicate the beginning of a procedure.

**11. ENDP** ; Used to indicate the end of a procedure.

**12. END** ; Used to indicate the end of a program.

**13. EQU** ; Defines a constant
E.g.:: AREA EQU 25H ; Creates a constant by the name AREA with a value 25H
Do remember, in the class, you have been clearly made to understand the difference between using a variable and using a constant.

**14. EVEN / ALIGN** ; Ensures that the data will be stored by the assembler in the memory in an aligned form. Aligned data works faster as it can be accessed in One cycle. Misaligned data, though is valid, requires two cycles to be accessed hence works slower.

**15. OFFSET** ; Can be used to tell the assembler to simply substitute the offset address of any variable.
E.g.:: MOV Si, OFFSET String1 ; SI gets the offset address of String1

**16. Start** ; It's the label from where the microprocessor to start executing the program

**17. Model Directives**

**.MODEL SMALL** **;** All Data Fits in one 64 KB segment.
All Code fits in one 64 KB Segment

**.MODEL MEDIUM** **;** All Data Fits in one 64 KB segment.
Code may be greater than 64 KB

**.MODEL LARGE** **;** Both Data and Code may be greater than 64 KB

Combined Example:

```
Data    SEGMENT
        LIST DB 10 DUP (0); Stores LIST as a series of 10 bytes initialized to zero    …
Data    ENDS

Code    SEGMENT
        Assume CS: Code, DS: Data       ; Makes Code → Code Segment
                                        ; and Data → Data Segment.
Start:      …
            …
            …

Code    ENDS
        END Start
```

There are many more assembler directives.