1.4 ASSEMBLER DIRECTIVES

Assembler directives help the assembler to correctly understand the assembly language programs to prepare the codes. Another type of hint which helps the assembler to assign a particular constant with a label or initialize particular memory locations or labels with constants is called an operator. Rather, the operators perform the arithmetic and logical tasks unlike directives that just direct the assembler to correctly interpret the program to code it appropriately. The following directives are commonly used in the assembly language programming practice using Microsoft Macro Assembler (MASM) or Turbo Assembler (TASM).

DB: Define Byte

The DB directive is used to reserve byte or bytes of memorylocations in the available memory.

LIST DB 0lH, 02H, 03H, 04H

This statement directs the assembler to reserve four memory locations for a list named LIST andinitialize them with the above specified four values.

DW: Define Word

It makes the assembler reserve the number of memory words (16-bit) instead of bytes. Some examples are given to explain this directive.

Examples WORDS DW 1234H, 4567H, 78ABH, 045CH

DQ: Define Quad word

This directive is used to direct the assembler to reset 4words (8 bytes) of memory for the specified variable and may initialize it with the specified values.

DT: Define Ten Bytes

The DT directive directs the assembler to define the specified variable requiring l -bytes for its storage and initialize the 10bytes with the specified values.

ASSUME: Assume Logical Segment Name

The ASSUME directive is used to inform the assembler, the names of the logical segments to be assumed for different segments used in the program. In the assembly language program, each segment is given aname.

For example, the code segment may be given the name CODE, data segment may be given the name DATA etc.

ASSUME CS:

CODE directs the assembler that the machine codes are available in a segment named CODE, and hence the CS register is to be loaded with the Address(segment) allotted by the operating system for the label CODE, while loading.

ASSUME DS:

DATA indicates to the assembler that the data items related to theprogram, are available in a logical segment named DATA, and the DS register is to be initialized by the segment Address value decided by the operating system for thedata segment, while loading.

END: END of Program

The END directive marks the end of an assembly language program. When the assembler comes across this END directive, it ignores the source lines available later on. Hence, it should be ensured that the END statement should be the last statement in the file and should not appear in between.

ENDP: END of Procedure.

In assembly language programming, the subroutines arecalled procedures. Thus, procedures may be independent program modules which return particular results or values to the calling programs. The ENDP directive is used toindicate the end of a procedure.

```
PROCEDURE STAR
    .
    .
    STAR ENDP
```

ENDS: END of Segment

This directive marks the end of a logical segment.

```
DATA SEGMENT
.
.
.
DATA ENDS
ASSUME CS: CODE, DS:DATA CODE SEGMENT.
.
.
.
C
O
D
E
E
N
D
S
E
N
D
```

The above structure represents a simple program containing two segments named DATA and CODE. The data related to the program must lie between the DATA SEGMENT and DATA ENDS statements. Similarly, all the executable instructions must lie between CODE SEGMENT and CODE ENDS statements.

EVEN: Align on Even Memory Address

The EVEN directive updates the location counterto the next even Address if the current location counter contents are not even, and assigns the following routine or variable or constant to that Address.

EQU: Equate

The directive EQU is used to assign a label with a value or a symbol. Theuse of this directive is just to reduce the recurrence of the numerical values or constants in a program code.

Example

LABEL

EQU

0500H

ADDITI

ON EQU

ADD

The first statement assigns the constant 500H with the label LABEL, while the second statement assigns another label ADDITION with mnemonic ADD.

EXTRN: External and Public

The directive EXTRN informs the assembler that the names, procedures and labels declared after this directive have already been defined in some other assembly language modules.

GROUP: Group the Related segment

The directive is used to form logical groups of segments with similar purpose or type. This directive is used to inform the assembler to form a logical group of the following segment names.

PROGRAM GROUP CODE, DATA, STACK

The above statement directs the loader/linker to prepare an EXE file such that CODE, DATA and STACK segment must lie within a 64kbyte memory segment that is named as PROGRAM. Now, for the ASSUME statement, one can use the label PROGRAM rather than CODE, DATA and STACK as shown.

ASSUME CS: PROGRAM, DS: PROGRAM, SS: PROGRAM.

LABEL: Label

The Label directive is used to assign a name to the current content of the location counter. At the start of the assembly process, the assembler initializes a location counter to keep track of memory locations assigned to the program.

LENGTH: Byte Length of a Label

This directive is not available in MASM. This isused to refer to the length of a data array or a string.

MOV CX, LENGTH ARRAY

LOCAL:

The labels, variables, constants or procedures declared LOCAL in a module areto be used only by thatmodule.

NAME: Logical Name of a Module

The NAME directive is used to assign a name to an assembly language program module. The module may now be referred to by its declared name.

OFFSET: Offset of a Label

When the assembler comes across the OFFSET operator along with a label, it first computes the 16-bit displacement (also called as offset interchangeably) of the particular label, and replaces the string 'OFFSET LABEL' by thecomputed displacement.

ORG: Origin

The ORG directive directs the assembler to start the memory allotment for the particular segment, block or code from the declared Addressing. The ORG statement while starting the assembly process for a module, the assembler initializes a location counter to keep track of the allotted addresses for the module. If the ORG statement is not written in the program, the location counter is initialized to 0000. If an ORG 200H statement is present at the starting of the code segment of that module, then the code will start from 200H Addressing code segment.

PROC: Procedure

The PROC directive marks the start of a named procedure in the statement.

PTR: Pointer

The pointer operator is used to declare the type of a label, variable ormemory operand. The operator PTR is prefixed by either BYTE or WORD. If the prefix is BYTE, then the particular label, variable or memory operand is treated as an 8-bit quantitywhile if WORD is the prefix, then it is treated as a 16- bit quantity.

Example:

MOV AL, BYTE PTR [SI];

Moves content of memory location addressed by SI (8-bit) toAL

SEG: Segment of a Label

The SEG operator is used to decide the segment Address of the label, variable, or procedure and substitutes the segment base Address in place of ‘SEG label’. The example given below explains the use of SEG operator.

Example MOV AX, SEG ARRAY;

This statement moves the segment address

SEGMENT: Logical Segment

The SEGMENT directive marks the starting of alogical segment. The started segment is also assigned a name, i.e. label, by this statement. The SEGMENT and ENDS directive must bracket each logical segment of a program.

TYPE :

The TYPE operator directs the assembler to decide the data type of the specified label and replaces the 'TYPE label' by the decided data type. For the word type variable, the data type is 2, for double word type, it is 4, and for byte type, it is 1.

GLOBAL:

The labels, variables, constants or procedures declared GLOBAL may be used by other modules of the program. Once a variable is declared GLOBAL, it can be used by any module in the program. The following statement declares the procedure ROUTINE as a global label.