Assembler directives

- **DB: Define Byte The DB directive is used to reserve byte or bytes of memory** locations in the available memory.
- Example: VALUE DB 50H

  LIST DB 0lH, 02H, 03H, 04H
- This statement directs the assembler to reserve four memory locations for a list named LIST and initialise them with the above specified four values.
- MESSAGE DB 'GOOD MORNING'

Assembler directives

- **DW: Define Word. The DW directive serves the same purposes as the DB** directive, but it now makes the assembler reserve the number of memory words (16-bit) instead of bytes.

EX: WORDS DW 1234H

- Another option of the DW directive is explainedwith the DUP operator.

EX: WDATA DW 5 DUP (6666H)

- This statement reserves five words, i.e. 10-bytes of memory for a word lable WDATA and initialises all the word locations with 6666H.

DQ: Define Quad word This directive is usedto direct the assembler to reserve 4 words (8 bytes) of memory for the specified variable and may initialise it with the specified values.

DT: Define Ten Bytes. The DT directive directs the assembler to define the specified variable requiring la-bytes for its storage and initialise the 10bytes with the specified values.

ASSUME: Assume Logical Segment Name The ASSUME directive is used to inform the assemble, the names of the logical segments to be assumed for different segments used in the program.**END: END of Program The END directive marks the end of an assembly** language program. END statement should be the last statement in the file

- **SEGMENT: Logical Segment The SEGMENT directive marks the starting of a** logical segment. The started segment is also assigned a name, i.e. label, by this statement.
- **ENDS: END of Segment This directive marks the end of a logical segment.** Any statement appearing after ENDS will be neglected from the segment.

EX:
```
DATA SEGMENT
.
.
.
DATA ENDS
ASSUME CS: CODE, DS:DATA
CODE SEGMENT.
.
.
.
CODE ENDS
END
```

- 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.

- **EQU: Equate The directive EQU is used to assign a label with a value or a** symbol. The use of this directive is just to reduce the recurrence of the numerical values or constants in a program code. The recurring value is assigned with a label, and that label is used in place of that numerical value, throughout the program.
- Example

```
LABEL EQU 0500H
ADDITION EQU ADD
```

Assembling ,linking &loading

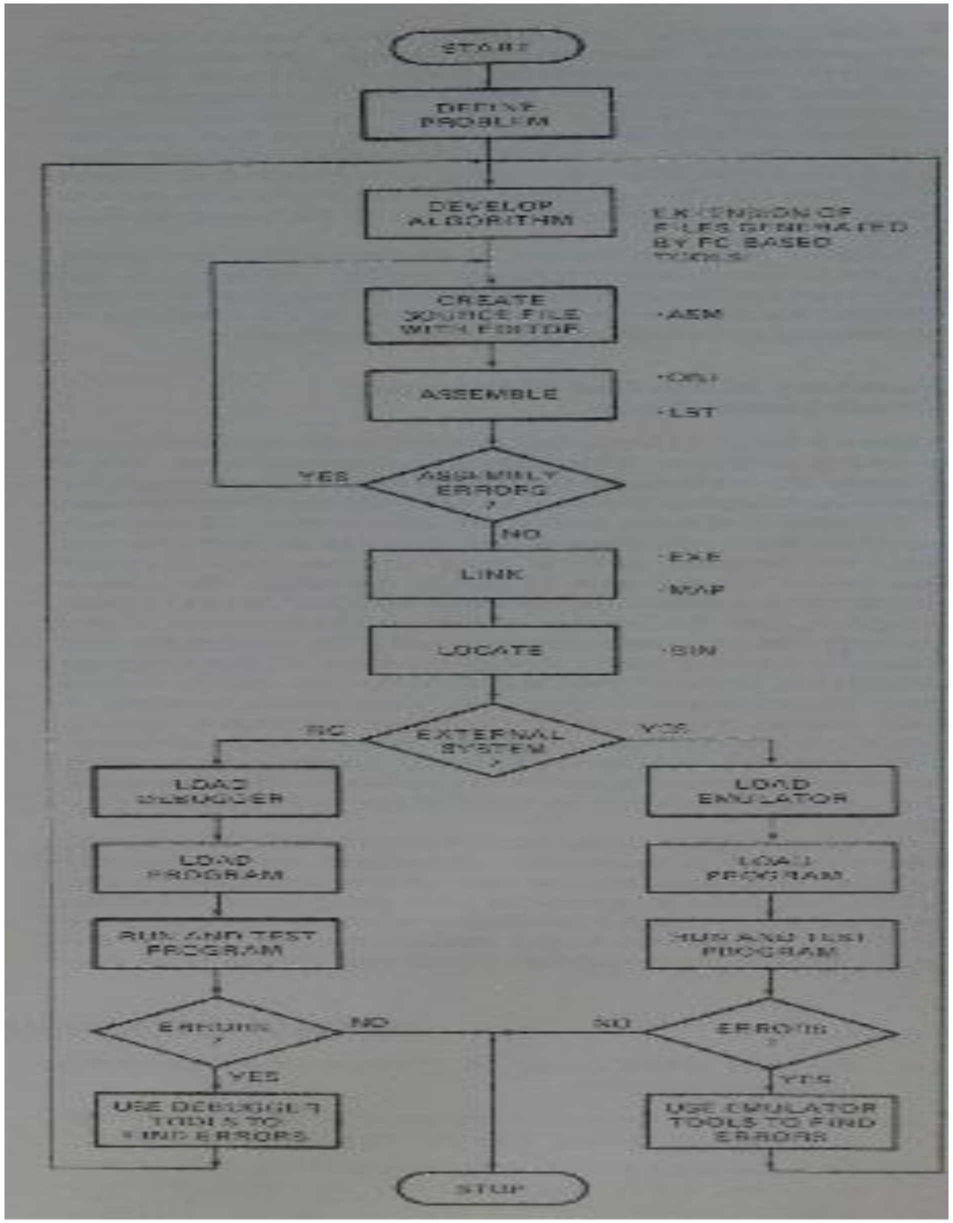
- Editor: an editor is a program which allows you to create a file containing the assembly language statements for your programs.
- The actual position of each field on a line is not important. After typing the program we have to save the file on the disk or floppy.
- This file is called **source file.** It contains the extension **.ASM** for turbo assembler.

- Assembler: this program is used to translate the assembly language mnemonics for instructions to the corresponding binary codes.
- It runs in two passes.
- on the first pass it generates a symbol table.
- On the second pass it generates the binary codes.
- The first file is called the object file and having the extension .OBJ
- The second file generated by the assembler is called the assembler list file and is having the extension .LST

- LINKER:it is a program used to join several object files into one object file.
- The linker produces link file which contains the binary codes for all combined modules.
- The linker also produces a link map file which contains the address information about the linked files.
- The produced link files have the extension .EXE and .MAP

- **LOCATOR:**it is a program used to assign the specific addressess of where the segments of object code are to be loaded into memory.
- A locator program converts the .EXE file to a .BIN file which has the physical addressess.
- **Debugger:**if your program does not require external hardware or requires only hardware accessible directly from your microcomputer then you can use a debugger to run and debug your program.
- It allows you to load your object code program and troubleshoot it.

- A debugger also allows you to set a breakpoint at any point in your program.
- The advantage of a breakpoint is we can debug the program upto the breakpoint and it is correct then we can further move after that point.
- **Emulator:** another way to run the program is using emulator. It is a mixture of hardware and software.
- It is used to teat and debug external systems.


START
DEFINE PROBLEM
DEVELOP ALGORITHM
EXTENSION OF FILES GENERATED BY PC BASED TOOLS
CREATE SOURCE FILE WITH EDITOR
.ASM
ASSEMBLE
.OBJ
.LST
YES
ASSEMBLY ERRORS ?
NO
LINK
.EXE
.MAP
LOCATE
.BIN
NO
EXTERNAL SYSTEM ?
YES
LOAD DEBUGGER
LOAD EMULATOR
LOAD PROGRAM
LOAD PROGRAM
RUN AND TEST PROGRAM
RUN AND TEST PROGRAM
ERRORS ?
NO
NO
ERRORS ?
YES
YES
USE DEBUGGER TOOLS TO FIND ERRORS
USE EMULATOR TOOLS TO FIND ERRORS
STOP

Timing and delays

- The rate at which 8086 instructions executed are determind by a clock.
- Each instruction takes a certain number of clock cycles to execute.
- Generally we have to introduce a delay between the execution of 2 instructions.
- For example we want to read the data values from port wait 1ms and then read again.
- The basic princple is to execute an instruction or series of instructions over and over until the desire time has elapsed.