Page No.
Date: / /

Directives in 8086 :-

The directives are the number of statements that enable us to control the way in which the source program assembles and lists.

These statements called directives act only during the assembly of program and generate no. machine-executable code.

Different types of directives are:-

1. The page and title listing directives

The page and title directives help to control the format of listing of an assembled program.

The page directives defines the maximum no. of lines to list as a page and the maximum number of character as a line.

PAGE [length] [width]
DEFAULT : Page [50] [80]

Title gives title and place the title on second line of each page of the program.

TITLE:- Text [comment]

2. Segment directives

It gives the start of a segment for stack, data and the code.

Page No.
Date: / /

Seg -name Segment [align] [combine] ['class]
Seg - name ENDS
The segment name must be present, must be unique and must follow assembly language naming convention.

ENDS - ENDS statement indicates the END of the segment.
ALIGN :- Align option indicates the boundary on which the segment is to begin; para is used to align the segment on paragraph boundary.
- COMBINE option indicates whether to combine the segment with other segment when they are linked after assembly.
- Class options is used to group related segments when linking
- The class code for code segment, stack for stack segment and data for data segment.

3) PROC directives

The code segment contains the executable code for a program, which consist of one or more procedures, defined initially with the PROC directives and ended with the END P directives.

PROC -name
PROC [FAR/NEAR]
- - - - - -
- - - - - -
- - - - - -
PROC -name END P

FAR is used for that first executing procedures and rest procedures call will be near.

4. END directives

END directives end the entire program and appears as the last statement.

- ENDS directive ends a segment and
- ENDP directives ends a procedure.

5) ASSUME Directives:-

An .EXE program uses the SS register to address the stack, DS to address the data segment and CS to address the code segment.

- Used in conventional full segment directives only

6) DN directive:- (Defining data types):-

Assembly language has directives to define data.

[name] Expression

The Dn directive can be any one of the following:-

DB:-	Define Byte	1 byte
DW:-	Define Word	2 byte
DD:-	Define Double	4 byte
DF:-	Define forward	6 bytes

Page No.
Date: / /

DQ :- Define quadword 8 bytes
DT :- Define 10 bytes 10 bytes

Eg :-

```
VAL1    DB  15
ARR     DB  21, 23, 27, 53
MOV     AL, ARR[2]
MOV     AL, ARR+2
```

EQU directive

It can be used to assign a name to constant.

```
Eg :- FACTOR  EQU 12
      PIE     EQU 3.14
```

DUP directive

It can be used to initialize several locations to zero.

```
eg :- sum   DW   4 DUP(0)
```

Reserves four word starting at the offset sum in DS and initialize them to 0.

Page No.
Date: / /

Page 60, 132

Program 1:

TITLE sum program to add two numbers

MODEL SMALL
STACK 64
DATA
NUM 1 DW 3241
NUM 2 DW 572
SUM DW ?

CODE
MAIN PROC FAR
MOV AX, @DATA
MOV DS, AX
MOV AX, NUM1
ADD AX, NUM2
MOV SUM, AX
MOV AX, 4C00H
INT 21H
MAIN ENDP
END MAIN

Program 2:

TITLE Subtraction program to subtract two numbers.

MODEL SMALL
STACK 64
DATA
NUM 1 DW 3241
NUM2 DW 572

Page No.
Date: / /

DIFF DW ?

- CODE

```
MAIN PROC FAR
MOV AX, @DATA
MOV DS, AX
MOV AX, NUM1
MOV 1
SUB AX, NUM2
SUB Diff AX
MOV AX, 4C00H
INT 21H
MAIN ENDP
ENDP MAIN
```

Addressing mode

Addressing mode describes types of operand and the way in which they are accessed for executing an instruction. An operand address provides source of data for an instruction to process.

An instruction may have 0 to 2 operands

1. Register Addressing Mode

For this mode, a register may contain source operand, destination operand or both.

Eg:- MOV AH, BL

MOV DX, CX

DW

Page No.
Date: /

Program 8: ADD
TITLE Program to^Ten Number:-

```
.MODEL SMALL
.STACK 64H
.DATA
    ARR DB 73,91,12,15,79,94,55,86
SUM DW ?
    .CODE
    MAIN PROC FAR
    MOV AX,@DATA
    MOV DX,AX
    MOV CX,10
    MOV AX,0
    LEA BX,ARR
L2: ADD AL,[BX]
    JNC L1
    INC AH
L1: INC BX
    LOOP L2
    MOV SUM,AX
    MOV AX,4C00H
    INT 21H
    MAIN ENDP
    END MAIN
```

Page No.
Date: / /

Program 4

```
TITLE  To display a string
.MODEL SMALL
.STACK 64
.DATA
    STR DB 'programming is fun', '$'
.CODE
 MAIN PROC FAR
      MOV  AX, @DATA
      MOV  DS, AX
      MOV  AH, 09H          (using string)
      LEA  DX, STR
      INT  21H
      MOV  AX, 4C00H
      INT  21H
MAIN ENDP
END MAIN
```

Program 5: 8086 to multiply two 8-bit no.

Page No.
Date: / /

TITLE: program to multiply 8-bit number

```
.MODEL SMALL
.STACK 64H
.DATA
        ARR  DW 7812H, 4592H
        PRO  DW ?
.CODE
MAIN  PROC FAR
MOV   AX  @DATA
MOV   DS, AX
LEA   BX, ARR        move array to BX
MOV   CX, BX         move one data to CX
INC   BX             Increase BX
MOV   AX, BX         Increased value to AX
MUL   CX             multiply val. of AX w
MOV   PRO, AY
MOV   AX, 4C00H
INT   21H
MAIN ENDP
END MAIN
```

Program 6:

Write a program to find largest block of data, length of block is 0A, store the maximum in location result.

```
TITLE Maximum in given series.
.MODEL SMALL
Stack 100h
Data
    List db 80,81,78,65,45,89,90,10,99,85
```

```
        Result db ?
.   CODE
        Main PROC
            MOV   AX  @DATA
            MOV     DS, AX
            MOV  SI  offset list
            MOV   AL, 00H
            MOV   CX, 0AH
Back:   CMP  AL, [SI]
            JNC  Ahead
            MOV  AL, [SI]
Ahead:  INC  SI
            LOOP  Back
            MOV  Result, AL
            MOV  AH, 4CH
            INT  21H
            Main ENDP
            END MAIN
```

2. Linking

3) Loading and executing:- It is loads the program in memory for execution it resolves remaining address.
This process creates the program segment prefix before loading.
It executes to generate the result.

Source program → (assembling) → Object program → (linking) → executable program

Program 7:
VIMP Write a program to reverse the given string for 8086.

Title reverse the string

```
DO sseg
. Model small
. stack 100h
. data
    string 1 db 'assembly language program', $
    length dw $ - string 1 - 1
. CODE
    Main PROC
    MOV AX, @data
    MOV DS, AX
    MOV SI, offset string1
    MOV CX, length
    ADD SI, CX

Back: MOV DL, [SI]
    MOV AH, 02H
    INT 21H
```

Page No.
Date: / /

```
DEC SI
Loop Back
MOV AH, 4CH
INT 21H
@Main ENDP
END Main
```

Old Qstn

Program 8:

Write a program to generate the multiplication tabb.

Title generate the multiplication table

```
MODEL SMALL
STACK 32 H
DATA
        NUM1 DB 5
        NUM2 DB1
        TAB DB 10 DUP (?)
CODE
    MAIN PROC FAR
    MOV  AX, @DATA
    MOV  DS, AX
    MOV  BX, 0
    MOV  CX, 10
L1: MOV AL, NUM1
    MUL NUM2
    MOV TAB[BX], AL
    INC NUM2
    LOOP L1
                MOV AX, 4C00H
                INT 21H
                MAIN ENDP
```