Assembler Directives

Dr. Manju Khurana
Assistant Professor, CSED
TIET, Patiala
manju.khurana@thapar.edu

Assembler Directives

- **The instructions that will give the information to assembler for performing assembling are called Assembler Directives.**
- **These assembler directives are not executed by microprocessor, so they are also called dummy or pseudo instructions.**

When Assembler performs assembling:

- **Instructions are translated from assembly language to the machine language.**
- **Each byte of the instruction code or data is allotted effective address of one memory location.**

1. DB (Define Byte)

- **The assembler directive creates storage for a byte or a group of bytes and optionally assigns starting values.**

Examples are:

- **DB 12H, 23H, 45H, 56H**
- **NEXT DB 12, 34, 45**
- **DB 100 DUP (0)**
- **DB 100 DUP (?)**
- **DB 'DURG'**

2. DW (Define Word)

- **It is used for memory location allotment of data word (16 bits), i.e. it is allotted two memory locations for each word.**

Examples are:

- **DW 93H, 0ABCDH, 6745H**
- **NEXT DW 12H, 0AB34H, 4567H**
- **DW 200 DUP (0ABCDH)**
- **DW 100 DUP (?)**

3. DD (Define Double Word)

- **It is used for memory location allotment of data double word (32 bits), it creates storage for a 32-bit double word variables, with the option of giving it a starting value.**

Example is:

- **NEXT DD 12345678H, 7890H, 0H**

4. DQ (Define Quad Word)

- **It is used for memory location allotment of data quad word (64 bits), it creates storage for a 64-bit double word variables, with the option of giving it a starting value.**

Example is:

- **DURG DQ 123456789ABCDEF0H, 8988H, 515H**

5. DT (Define Ten Bytes)

This directive is used to tell the assembler to define a variable, which is 10 bytes in length, or to reserve 10 bytes of storage memory.

Format:

[name] DT initial value.....

For example:

1. BHILAIDURG DT 123456789A

   The above statement will declare an array named BHILAIDURG, which are 10 bytes in length. It will initialize the 10 bytes with the value 123456789A when the program is loaded into memory to the run.

2. DURG DT 100 DUP (?)

   1000 memory locations are reserved for 100 numbers of 10 bytes each. Let us assume that starting memory location is 5000h, so the effective address allotted to DURG = 5000h

6. ASSUME

It is used to inform the assembler about the names, which are assumes for different memory segments.

For example:

1. ASSUME CS: CODE
   Above statement tells the assembler that the instructions for a program are in logical segment named CODE.
2. ASSUME CS: CODE, DS: DATA, ES: EXTRA, SS: STACK

7. SEGMENT and ENDS

Segment assembler directives indicate the start of the SEGMENT and ENDS indicate that end of the segment.

Format:

```
Name                SEGMENT
(31 Alphanumeric)
        :
        :
Name                ENDS
```

For example:

If the name assumes for code memory segment is CODE, for data memory segment is DATA, for extra-memory segment is EXTRA, and for stack memory segment is STACK, then start and end of each segment can be indicated as given below:

```
CODE        SEGMENT
              :
              :
CODE        ENDS
DATA        SEGMENT
              :
              :
DATA        ENDS
EXTRA       SEGMENT
              :
              :
EXTRA       ENDS
STACK       SEGMENT
              :
              :
STACK       ENDS
```

8. ORG (Originate)

This assembler directive is used to assign the starting memory location to the program. If ORG is not given, then start the program from effective address 0000h by default.

Format:

ORG address/$

For example:

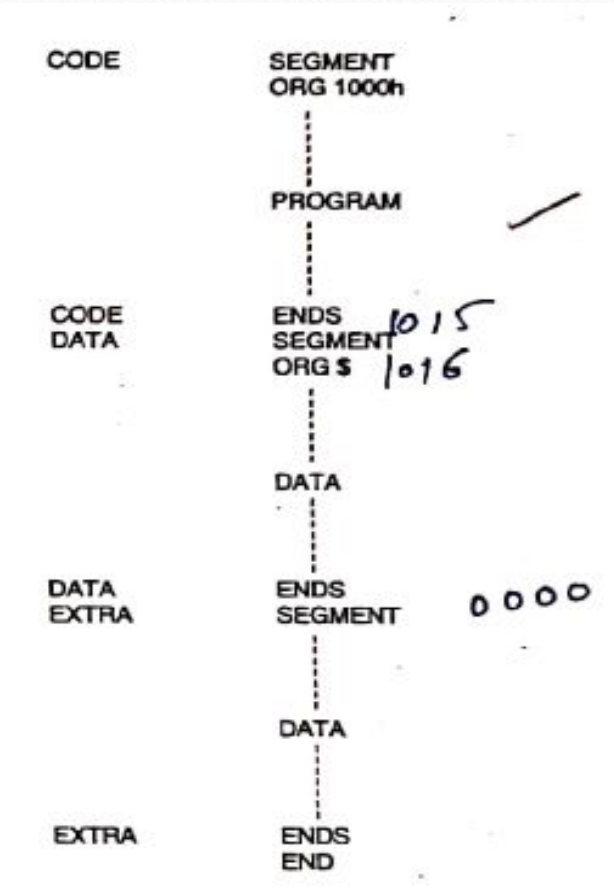
ASSUME CS: CODE, DS: DATA, SS: STACK

(a) The name of code memory segment, data segment and extra-segment are CODE, DATA and EXTRA.

(b) In CODE memory segment, the effective address allotment is started from 1000h (ORG 1000h)

(c) If the last effective address allotted in code memory segment is 1015h, then in DATA memory segment the effective address allotment will start in continuation, i.e. from address 1016h. (ORG $)

(d) In EXTRA memory segment ORG is not given, so effective address allotment will start from 0000h.



9. END

This directive as the name implies informs the assembler that assembling should stop and is given only once and in the very last. The assembler will ignore any instructions or statements after END directive.

10. EQU

This assembler directive is used to tell the assembler about the values assigns for different types of labels used in the program.

Format:

Label EQU 8/16-bit number/arithmetic expression

For example:

1. MOV CX, COUNT
   COUNT EQU 1000h
   The value of the COUNT will be used as 1000h.
2. MOV DI, ADDRESS1
   ADDRESS1 EQU ADDRESS+1000h
   If the value of ADDRESS is already defined as 2000h, then ADDRESS1 = ADDRESS + 1000 = 2000 + 1000 = 3000h
3. DATA1 EQU 23h
   DATA2 EQU 24h
   MOV AL, DATA1
   ADD AL, DATA2
   Here DATA1 i.e. 23h and DATA2 i.e. 24 is added and the result is copied into register AL.

11. LABEL

LABEL: The label directive is used to give a name to the current value in the location counter. The label directive must be followed by a term which specifies the type you want associated with that name.

```
STK SEGMENT
        S DW 100 DUP(0)
        S_TOP LABEL WORD
STK ENDS
```

12. LENGTH

- **LENGTH:** The directive length informs the assembler about the number of elements in a data item such as an array. If an array is defined with DB then it returns the number of bytes allocated to the variable. If an array is defined with DW then it returns the number of words allocated to the array variable.

Ex : MOV AX, LENGTH ITEMS

```
data segment
        items db 08h,78h,56h,78h,98h
data ends
```

- In the above example the number of elements assigned as array are 5. So five is the length of items which will be stored in AX.

13. PROC and ENDP

PROC: The PROC directive is used to identify the start of a procedure. The PROC directive follows a name you give the procedure. After the PROC directive the term NEAR or FAR is used to specify the type of the procedure.

```
Factorial  PROC  Near
---------
---------
RET
Factorial  ENDP
```

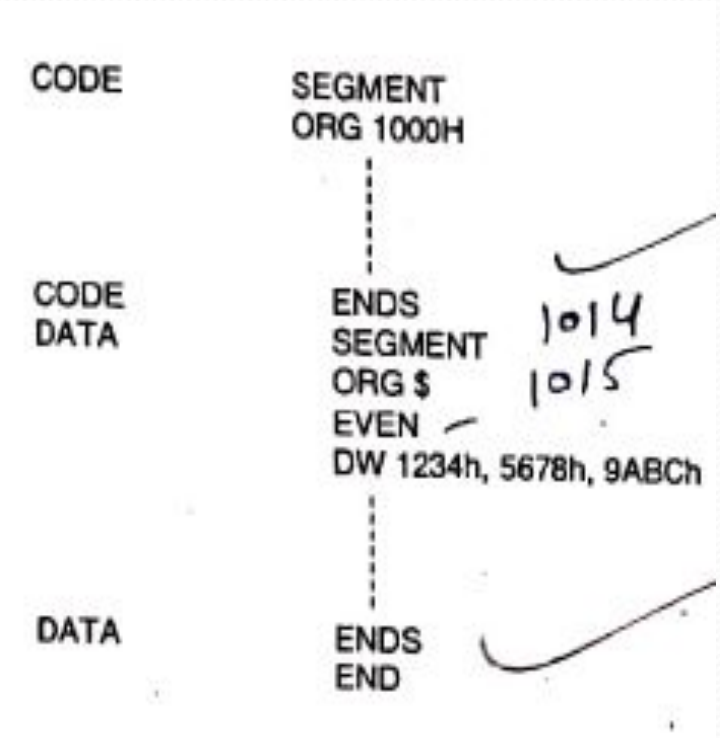
13. PROC and ENDP

ENDP [end procedure]: This directive is used along with the name of the procedure to indicate the end of a procedure to the assembler.

```
SQUARE_ROOT   PROC   NEAR
----------
----------
SQUARE_ROOT   ENDP
```

14. EVEN

✓**EVEN :** Align as even memory address. The directive EVEN is used to inform the assembler to increment the location counter to the next even memory address if it is not pointing to even memory location already.



15. OFFSET

OFFSET : The directive OFFSET informs the assembler to determine the displacement of the specified variable with respect to the base of data segment.

Offset variable_ name

Ex: MOV DX, OFFSET MSG

```
Data segment
-----
-----
MSG DB 'suresh'
----
Data ends
```


Thank you!