Assembler Directives (cont..)

- **ASSUME**
- **DB** - Defined Byte.
- **DD** - Defined Double Word
- **DQ** - Defined Quad Word
- **DT** - Define Ten Bytes
- **DW** - Define Word

Assembler Directives (cont..)

- **ASSUME Directive** - The ASSUME directive is used to tell the assembler that the name of the logical segment should be used for a specified segment. The 8086 works directly with only 4 physical segments: a Code segment, a data segment, a stack segment, and an extra segment.
- **Example:**

  ASUME CS:CODE ;This tells the assembler that the logical segment named CODE contains the instruction statements for the program and should be treated as a code segment.

  ASUME DS:DATA ;This tells the assembler that for any instruction which refers to a data in the data segment, data will found in the logical segment DATA.

Assembler Directives (cont..)

- **DB** - DB directive is used to declare a byte-type variable or to store a byte in memory location.
- **Example:**

1. **PRICE DB 49h, 98h, 29h** ;Declare an array of 3 bytes, named as PRICE and initialize.
2. **NAME DB 'ABCDEF'** ;Declare an array of 6 bytes and initialize with ASCII code for letters
3. **TEMP DB 100 DUP(?)** ;Set 100 bytes of storage in memory and give it the name as TEMP, but leave the 100 bytes uninitialized. Program instructions will load values into these locations.

Assembler Directives (cont..)

- **DW - The DW directive is used to define a variable of type word or to reserve storage location of type word in memory.**
- **Example:**

  MULTIPLIER DW 437Ah ; this declares a variable of type word and named it as MULTIPLIER. This variable is initialized with the value 437Ah when it is loaded into memory to run.

  EXP1 DW 1234h, 3456h, 5678h ; this declares an array of 3 words and initialized with specified values.

  STOR1 DW 100 DUP(0); Reserve an array of 100 words of memory and initialize all words with 0000.Array is named as STOR1.

Assembler Directives (cont..)

- **END** - END directive is placed after the last statement of a program to tell the assembler that this is the end of the program module. The assembler will ignore any statement after an END directive. Carriage return is required after the END directive.
- **ENDP** - ENDP directive is used along with the name of the procedure to indicate the end of a procedure to the assembler
- **Example:**

  SQUARE_NUM PROCE ; It start the procedure

  ;Some steps to find the square root of a number

  SQUARE_NUM ENDP ;Hear it is the End for the procedure

Assembler Directives (cont..)

- **END** - End Program
- **ENDP** - End Procedure
- **ENDS** - End Segment
- **EQU** - Equate
- **EVEN** - Align on Even Memory Address
- **EXTRN**

Assembler Directives (cont..)

- **ENDS** - This ENDS directive is used with name of the segment to indicate the end of that logic segment.
- **Example:**

```
CODE    SEGMENT    ;Hear it Start the logic
                   ;segment  containing code
; Some instructions statements to perform the logical
;operation
CODE    ENDS       ;End of segment named as
                   ;CODE
```

Assembler Directives (cont..)

- **EQU** - This EQU directive is used to give a name to some value or to a symbol. Each time the assembler finds the name in the program, it will replace the name with the value or symbol you given to that name.
- **Example:**

  FACTOR EQU 03H ; you has to write this statement at the starting of your program and later in the program you can use this as follows

  ADD AL, FACTOR ; When it codes this instruction the assembler will code it as ADDAL, 03H

  ;The advantage of using EQU in this manner is, if FACTOR is used many no of times in a program and you want to change the value, all you had to do is change the EQU statement at beginning, it will changes the rest of all.

Assembler Directives (cont..)

- **EVEN** - This **EVEN** directive instructs the assembler to increment the location of the counter to the next even address if it is not already in the even address. If the word is at even address 8086 can read a memory in 1 bus cycle.

  If the word starts at an odd address, the 8086 will take 2 bus cycles to get the data. A series of words can be read much more quickly if they are at even address. When EVEN is used the location counter will simply incremented to next address and NOP instruction is inserted in that incremented location.

Assembler Directives (cont..)

- **Example**:

```
DATA1     SEGMENT
; Location counter will point to 0009 after assembler reads
;next statement
SALES  DB  9  DUP(?)    ;declare an array of 9  bytes
EVEN                    ; increment  location counter to 000AH
RECORD  DW  100  DUP( 0 )  ;Array of 100 words will start
                        ;from an even address for quicker read
DATA1     ENDS
```

Assembler Directives (cont..)

- **GROUP** - Group Related Segments
- **LABLE**
- **NAME**
- **OFFSET**
- **ORG** - Originate

Assembler Directives (cont..)

- **GROUP** - The **GROUP** directive is used to group the logical segments named after the directive into one logical group segment.
- **INCLUDE** - This **INCLUDE** directive is used to insert a block of source code from the named file into the current source module.

Assembler Directives (cont..)

- **PROC** - Procedure
- **PTR** - Pointer
- **PUBLC**
- **SEGMENT**
- **SHORT**
- **TYPE**

Assembler Directives (cont..)

- **PROC** - The **PROC** directive is used to identify the start of a procedure. The term near or far is used to specify the type of the procedure.
- Example:

  SMART PROC FAR ; This identifies that the start of a procedure named as SMART and instructs the assembler that the procedure is far .

  SMART ENDP

  This PROC is used with ENDP to indicate the break of the procedure.

Assembler Directives (cont..)

- **PTR** - This **PTR** operator is used to assign a specific type of a variable or to a label.
- Example:

INC [BX] ; This instruction will not know whether to increment the byte pointed to by BX or a word pointed to by BX.

```
INC   BYTE  PTR  [BX]   ;increment the byte
                        ;pointed to by BX
```

This PTR operator can also be used to override the declared type of variable . If we want to access the a byte in an array **WORDS DW 437Ah, 0B97h,**

```
MOV   AL, BYTE  PTR  WORDS
```

Assembler Directives (cont..)

- **PUBLIC** - The **PUBLIC** directive is used to instruct the assembler that a specified name or label will be accessed from other modules.
- Example:

  PUBLIC DIVISOR, DIVIDEND ;these two variables are public so these are available to all modules.

  If an instruction in a module refers to a variable in another assembly module, we can access that module by declaring as **EXTRN** directive.

Assembler Directives

- **TYPE** - **TYPE** operator instructs the assembler to determine the type of a variable and determines the number of bytes specified to that variable.
- **Example:**

  Byte type variable – assembler will give a value 1

  Word type variable – assembler will give a value 2

  Double word type variable – assembler will give a value 4

  ADD BX, TYPE WORD_ ARRAY ; hear we want to increment BX to point to next word in an array of words.

DOS Function Calls

- **AH 00H** : Terminate a Program
- **AH 01H** : Read the Keyboard
- **AH 02H** : Write to a Standard Output Device
- **AH 08H** : Read a Standard Input without Echo
- **AH 09H** : Display a Character String
- **AH 0AH** : Buffered keyboard Input
- **INT 21H** : Call DOS Function