**NAME:Manav Jain**

**SAPID:60004180050**

**BATCH:A/3**

**Aim**: Case Study on Assembler Directives

**Theory**: Assembler directives help the assembler to correctly understand the assembly language programs to prepare the codes. They are instructions that direct the assembler to do something**.**

1. **DB: DEFINE BYTE** The DB directive is used to declare a byte type variable, or a set aside one or more storage locations of type byte in memory.

**Example:** PRICE DB 49h, 98h, 29h ;Declare an array of 3 bytes, named as PRICE and initialize

1. **DW :DEFINE WORD** The DW directive is used to tell the assembler to define a variable of type word or to reserve storage locations of type word in memory. The statement MULTIPLIER DW 437AH, for example, declares a variable of type word named MULTIPLIER, and initialized with the value 437AH when the program is loaded into memory to be run.

**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.

1. **DQ :DEFINE QUADWORD** The DQ directive is used to tell the assembler to declare a variable 4 words in length or to reserve 4 words of storage in memory.

**Example:**  
BIG\_NUMBER DQ 243598740192A92BH  
This will declare a variable named BIG\_NUMBER and initialize the 4 words set aside with the specified number when the program is loaded into memory to be run.

1. **DT :DEFINE TEN BYTES** The DT directive is used to tell the assembler to declare a variable, which is 10 bytes in length or to reserve 10 bytes of storage in memory.

**Example:**  
PACKED\_BCD DT 11223344556677889900

This will declare an array named PACKED\_BCD, which is 10 bytes in length. It will initialize the 10 bytes with the values 11, 22, 33, 44, 55, 66, 77, 88, 99, and 00 when the program is loaded into memory to be run. The statement RESULT DT 20H DUP (0) will declare an array of 20H blocks of 10 bytes each and initialize all 320 bytes to 00 when the program is loaded into memory to be run.

1. **ASSUME** The ASSUME directive is used tell the assembler the name of the logical segment it should use for a specified segment. The statement ASSUME CS: CODE, for example, tells the assembler that the instructions for a program are in a logical segment named CODE.

**Example**

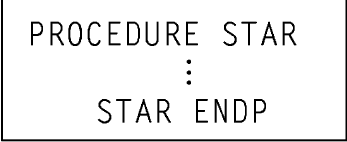
* **ASSUME CS:** Name of code segment
* **ASSUME DS:** Name of the data segment
* **ASSUME CS:** Code1, DS: Data1

1. **END : END PROGRAM** The END directive is put 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 statements after an END directive, so you should make sure to use only one END directive at the very end of your program module.

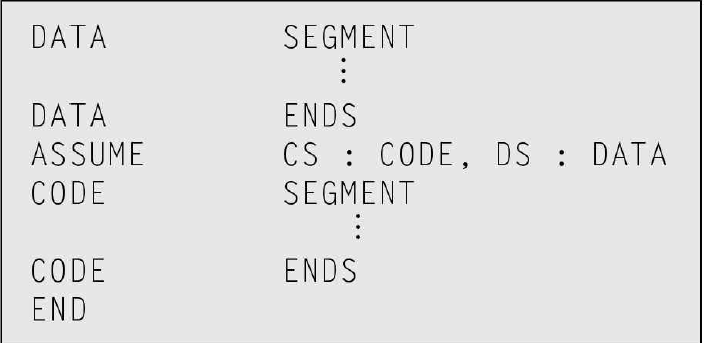
**Example**: CODE SEGMENT //Hear it Start the logic ;segment containing code ; Some instructions statements to perform the logical operation

CODE ENDS //End of segment named as CODE

1. **ENDP: END of Procedure** In assembly language programming, the subroutines are called procedures. The ENDP directive is used to indicate the end of a procedure.



1. **ENDS : END SEGMENT** This directive is used with the name of a segment to indicate the end of that logical segment.



1. **EVEN : ALIGN ON EVEN MEMORY ADDRESS** The assembler, while starting the assembler procedure of any program, initializes a location counter and goes on updating it, as assembly proceeds. The EVEN directive tells the assembler to increment the location counter to the next even address, if it is not already at an even address.

**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

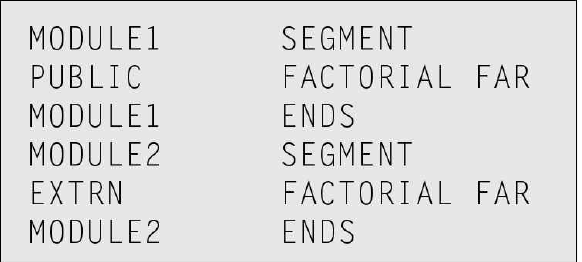
1. **EQU : EQUATE** EQU is used to give a name to some value or symbol. Each time the assembler finds the given name in the program, it replaces the name with the value or symbol you equated with that name. Suppose, for example, you write the statement FACTOR EQU 03H at the start of your program, and later in the program you write the instruction statement ADD AL, FACTOR. When the assembler codes this instruction statement, it will code it as if you had written the instruction ADD AL, 03H.

**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.

1. **EXTRN** **and PUBLIC** The EXTRN directive is used to tell the assembler that the name or labels following the directive are in some other assembly module. Name or labels referred to as external in one module must be declared public with the PUBLIC directive in the module in which they are defined.



1. **GROUP: Group the Related Segments** This directive is used to form logical groups of segments with similar purpose or type . The assembler passes an information to the linker/loader to form the code such that the group declared segments or operands must lie within 64K byte memory segment.



1. **LABEL** The Label directive is used to assign a name to the current content of the location counter. When an assembly process starts, the assembler initializes a location counter to keep track of the memory locations assigned to the program. As the program proceeds, the contents of the location counter are updated. During the assembly process, whenever the assembler comes across the LABEL directive, it assigns the declared label with the current contents of the location counter. The LABEL directive must be followed by a term that specifics the type you want to associate with that name. If the label is going to be used as the destination for a jump or a call, then the label must be specified as type near or type far. If the label is going to be used to reference a data item, then the label must be specified as type byte, type word, or type double word. Here’s how we use the LABEL directive for a jump address.



1. **LENGTH** LENGTH is an operator, which tells the assembler to determine the number of elements in some named data item, such as a string or an array. When the assembler reads the statement MOV CX, LENGTH STRING1, for example, will determine the number of elements in STRING1 and load it into CX.

Example: MOV CX, LENGTH STRING1

1. **LOCAL** The labels, variables , constants or procedures declared LOCAL in a module are to be used only by that particular module.



1. **NAME: Logical Name of a Module** The NAME directive is used to give a specific name to each assembly module when programs consisting of several modules are written.

NAME DB ‘ABCDEF’

1. **OFFSET** OFFSET is an operator, which tells the assembler to determine the offset or displacement of a named data item (variable), a procedure from the start of the segment, which contains it. When the assembler reads the statement MOV BX, OFFSET PRICES, for example, it will determine the offset of the variable PRICES from the start of the segment in which PRICES is defined and will load this value into BX.

**Example**: when the assembler read MOV BX.OFFSET PRICES, it will determine the offset of the prices.

1. **ORG : ORIGIN** As an assembler assembles a section of a data declarations or instruction statements, it uses a location counter to keep track of how many bytes it is from the start of a segment at any time. The location counter is automatically set to 0000 when assembler starts reading a segment. The ORG directive allows you to set the location counter to a desired value at any point in the program. The statement ORG 2000H tells the assembler to set the location counter to 2000H, for example.

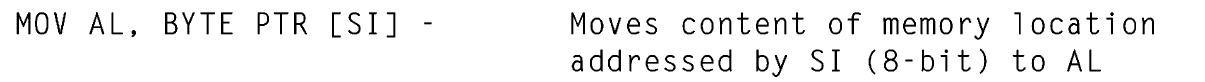
**Example:**  ORG 2000H tells the assembler to set the location counter to 2000H.

1. **PROC : PROCEDURE** 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 the term far is used to specify the type of the procedure. The statement DIVIDE PROC FAR, for example, identifies the start of a procedure named DIVIDE and tells the assembler that the procedure is far . The PROC directive is used with the ENDP directive to “bracket” a procedure.

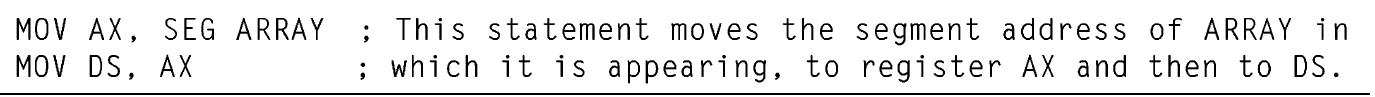
**Example**:

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

1. **PTR : POINTER** The PTR operator is used to declare the type of a label, variable or memory 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 quantity.



1. **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.



1. **SEGMENT** The SEGMENT directive is used to indicate the start of a logical segment. Preceding the SEGMENT directive is the name you want to give the segment. For example, the statement CODE SEGMENT indicates to the assembler the start of a logical segment called CODE. The SEGMENT and ENDS directive are used to “bracket” a logical segment containing code of data.

**Example**: the code segment is used to indicate to the assembler the start of logical segment.

1. **SHORT** The SHORT operator is used to tell the assembler that only a 1 byte displacement is needed to code a jump instruction in the program. The destination must in the range of –128 bytes to +127 bytes from the address of the instruction after the jump. The statement JMP SHORT LABEL is an example of the use of SHORT.

JMP SHORT NEARBY\_LABEL

1. **TYPE** The TYPE operator tells the assembler to determine the data type of a specified variable. The assembler actually determines the number of bytes in the type of the variable. For a byte-type variable, the assembler will give a value of 1, for a word-type variable, the assembler will give a value of 2, and for a double word-type variable, it will give a value of 4.

**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.

**Conclusion:** Hence we studied Assembler Directives