How OFFSET Works in Assembly Language

The OFFSET operator in **8086 Assembly Language** is used to obtain the **memory address** of a variable or label. It is commonly used with MOV and LEA (Load Effective Address) instructions.

Basic Syntax

MOV DX, OFFSET msg

- msg is a **label** (usually a string or variable).
- OFFSET msg retrieves the **starting address** of msg in memory.
- MOV DX, OFFSET msg stores this address into the DX register.

Why Use OFFSET?

In **8086 segmented memory**, variables (like strings) are stored in the **Data Segment (DS)**. When printing a string using INT 21H, DOS expects the **memory address** of the string in DX. Since msg is stored in the .DATA section, we need to load its **offset (memory address)** into DX before calling the interrupt.

Example: Printing a String

```
.MODEL SMALL

.STACK 100H

.DATA

   msg DB 'Hello, World!$', 0  ; Define a string ending with '$'

.CODE

MAIN PROC

   ; Initialize Data Segment

   MOV AX, @DATA

   MOV DS, AX

   ; Load address of msg into DX

   MOV DX, OFFSET msg

   ; Print the string

   MOV AH, 09H                 ; DOS function to display a string

   INT 21H                     ; Call DOS interrupt

   ; Exit program

   MOV AH, 4CH

   INT 21H

MAIN ENDP

END MAIN
```

Step-by-Step Execution

1. MOV AX, @DATA → Loads the **data segment address** into AX.
2. MOV DS, AX → Moves the address to the **DS register**.
3. MOV DX, OFFSET msg → Loads the **memory address** of msg into DX.
4. MOV AH, 09H → DOS function to display a string.
5. INT 21H → Prints the string from the memory address in DX.

Key Points

- OFFSET retrieves the **starting address** of a variable.
- Used with MOV DX, OFFSET msg to **pass strings to DOS interrupts**.
- Works with variables in the .DATA section.
- Commonly used for **printing messages, accessing arrays, and pointers**.

Using LEA Instead of OFFSET in Assembly Language

The **LEA (Load Effective Address)** instruction is an alternative to OFFSET. It loads the **memory address** of a variable into a register.

Difference Between LEA and OFFSET

Instruction	Usage	Works With
MOV DX, OFFSET msg	Loads the address at **assembly time**	Constants, static variables
LEA DX, msg	Loads the address at **runtime**	Registers, memory operands

Both can be used to get the address of a variable, but LEA works dynamically at runtime.

Example: Printing a String Using LEA

```
.MODEL SMALL
.STACK 100H
.DATA
   msg DB 'Hello, World!$', 0  ; Define a string ending with '$'
.CODE
MAIN PROC
   ; Initialize Data Segment
   MOV AX, @DATA
   MOV DS, AX
   ; Load address of msg using LEA
   LEA DX, msg
   ; Print the string
   MOV AH, 09H                 ; DOS function to display a string
   INT 21H                     ; Call DOS interrupt
   ; Exit program
   MOV AH, 4CH
   INT 21H
MAIN ENDP
END MAIN
```

Why Use LEA Instead of OFFSET?

1. LEA is more **flexible** because it works at **runtime**.
2. It allows **register-based addressing**, unlike OFFSET.
3. LEA is useful for **computing addresses dynamically** (e.g., accessing array elements).

Example: Using LEA for Array Access

```
.DATA
  array DB 10, 20, 30, 40
.CODE
  LEA SI, array        ; SI now holds the address of array[0]
  MOV AL, [SI]         ; AL = array[0] (10)
```

- Here, LEA SI, array loads the **starting address** of array into SI.
- We can then access elements using **register-based addressing**.

Summary

Instruction	**Purpose**
MOV DX, OFFSET msg	Loads the **address** at **assembly time**
LEA DX, msg	Loads the **address dynamically** at **runtime**

For simple **string printing**, both work the same. But LEA is **better for arrays and memory operations**.