**Practical - 15**

**Aim:** Write an assembly program to print string in reverse order.

**Description of instructions used:**

**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.

**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.

**ASSUME:** The ASSUME directive tells the assembler to assume, that a certain register contains the base of some structure (in your case: segments). In your case, CS and DS point to the code segment and the data segmentrespectively, both the one and only of their respective kind.

**LEA (Load Effective Address):** LEA and MOV both are same but in that there are quite difference between both of them.

* LEA means Load Effective Address
* MOV means Load Value

In short, LEA loads a pointer to the item you're addressing whereas MOV loads the actual value at that address. The purpose of LEA is to allow one to perform a non-trivial address calculation and store the result.

**LEA AX, [BP+SI+5]; Compute address of value**

**MOV AX, [BP+SI+5]; Load value at that address**

**ADD:** Used to add the provided byte to byte/word to word.

**DEC:** Used to decrement the provided byte/word by 1.

**INC:** Used to increment the provided byte/word by 1.

**LOOP:** Used to loop a group of instructions until the condition satisfies, i.e., CX = 0

**INT:** Used to interrupt the program during execution and calling service specified.

**Code:**

DATA **SEGMENT**

STR1 **DB** 'Rajiv\_Gupta','$'

STRLEN1 **DW** $-STR1

STRREV **DB** 20 DUP(' ')

DATA **ENDS**

CODE **SEGMENT**

ASSUME **CS**:CODE, **DS**:DATA

BEGIN:

**MOV** AX, DATA

**MOV** DS, AX

**MOV** ES, AX

**MOV** CX, STRLEN1

**ADD** CX, -2

**LEA** SI, STR1

**LEA** DI, STRREV

**ADD** SI, STRLEN1

**ADD** SI, -2

L1:

**MOV** AL, [SI]

**MOV** [DI], AL

**DEC** SI

**INC** DI

**LOOP** L1

**MOV** AL, [SI]

**MOV** [DI], AL

**INC** DI

**MOV** DL, '$'

**MOV** [DI], DL

PRINT:

**MOV** AH, 09H

**LEA** DX, STRREV

**INT** 21H

CODE **ENDS**

**END** BEGIN

**Output:**



