**Practical - 16**

**Aim:** Write an assembly program which converts upper case to lower and vice versa for a given string.

**Description of instructions used:**

**MARCO:** A **Macro** is a set of instructions grouped under a single unit. It is another method for implementing modular programming in the **8086** microprocessors (The first one was using Procedures)

**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.

**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**

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

**CMP:** The CMP instruction compares two operands. It is generally used in conditional execution. This instruction basically subtracts one operand from the other for comparing whether the operands are equal or not. It does not disturb the destination or source operands.

**JL:** The **JL instruction** is a conditional jump that follows a test. It performs a signed comparison jump after a **CMP** if the destination operand is less than the source operand.

**JG:** Jumps to the destination label mentioned in the instruction if the result of previous instruction (generally compare) causes ZF to have value equal to 0 and CF and OF to have same values, else no action is taken.

**JMP:** the JMP instruction performs an unconditional jump. Such an instruction transfers the flow of execution by changing the instruction pointer register.

**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

**Code:**

DIS **MACRO** STR

**MOV** AH,09H

**LEA** DX,STR

**INT** 21H

**ENDM**

DATA **SEGMENT**

MSG1 **DB** "ENTER YOUR STRING : $"

MSG2 **DB** "CONVERTED STRING IS : $"

STR1 **DB** 20 DUP('$')

LINE **DB** 10,13,'$'

DATA **ENDS**

CODE **SEGMENT**

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

START:

**MOV** AX,**DATA**

**MOV** DS,AX

**DIS** MSG1

**MOV** AH,0AH

**LEA** DX,STR1

**INT** 21H

**DIS** LINE

**MOV** CH,00

**MOV** CL,BYTE PTR[STR1+1]

**LEA** SI,STR1+2

L1: **MOV** AH,BYTE PTR[SI]

**CMP** AH,'A'

**JL** L4

**CMP** AH,'Z'

**JG** L2

**ADD** BYTE PTR[SI],32

**JMP** L3

L2: **CMP** AH,'a'

**JL** L4

**CMP** AH,'z'

**JG** L4

**SUB** BYTE PTR[SI],32

L3: **INC** SI

**LOOP** L1

**DIS** MSG2

**DIS** STR1+2

L4: **MOV** AH,4CH

**INT** 21H

CODE **ENDS**

**END** START

**Output:**





