

# COA – Lab Assignment 6

Name: Bhavin Patil

Roll No- 78

GR No : 12120056

Div : CS- D

Batch : B3

## **Problem Statement :**

### **Instructions –**

- **MOV:** This instruction is used to move data from one location to another.  
Syntax – mov destination, source
- **LEA (Load Effective Address):** It loads the specified register with the offset of a memory location.
- **JNZ: (conditional jump)** The program sequence is transferred to a particular level or a 16-bit address if Z=0 (or zero flag is 0)
- **MACRO:** A macro is a series of commands and instructions that you group together as a single command to accomplish a task automatically.
- **DUP:** The DUP directive tells the assembler to duplicate an expression a given number of times
- **SI:** SI is called source index and DI is destination index. As the name follows, SI is always pointed to the source array and DI is always pointed to the destination. This is usually used to move a block of data, such as records (or structures) and arrays.
- **DI:** DI stands for destination index, used as a pointer to the current character being written or compared in a string instruction. It is also available as an offset just like SI
- **CLD:** Description. The cld instruction clears the direction flag: DF = 0. The direction flag (DF) is used to influence the direction in which some of the instructions work when used with the REP\* prefix.
- **DEC:** The DEC instruction is used for decrementing an operand by one. It works on a single operand that can be either in a register or in memory.

- **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.
- **MOVSb:** The MOVSb instruction tells the assembler to move data as bytes; the MOVsw implies the string is to be moved as words.
- **RET:** Return from Procedure (ret).
- **JNC:** The JNC instruction transfers program control to the specified address if the carry flag is 0. Otherwise, execution continues with the next instruction. No flags are affected by this instruction.
- **DIV:** The DIV (unsigned divide) instruction performs 8-bit, 16-bit, and 32-bit division on unsigned integers.
- **CALL:** CALL instruction is used to call a subroutine. Subroutines are often used to perform tasks that need to be performed frequently.
- **JLE:** It performs a signed comparison jump after a cmp if the destination operand is less than or equal to the source operand.

## Commands –

1. **01h** : It is used to read character from standard input, with echo, result is stored in AL.
2. **02h** : It is used to display single character
3. **09h**: Displays the string until “\$” is reached.
4. **Int 21h**: Interrupt used to exit the program.
5. **.data**: This Command is used only when we want to store in Data Segment, basically, it is the memory access of the Data Segment. Whatever we want to print must be written here. Also, the variables are declared here.
6. **10, 13**: They work as Escape Sequence Character
7. **\$**: It states the end of a Statement
8. **Db (Define Byte)**: It acts as an Assembler Directive
9. **.code**: Full Logical Program is written here
10. **Tasm** – Used for Compilation
11. **tlink**– Perform linking operation

## Screenshots of Source Code and Output:

### Source Code –

```
File Edit Search View Options Help
C:\TASM\EXP6.ASM

.model small
.data

disp1 macro msg
mov ah, 09h
lea dx, msg
int 21h
endm

p1 label byte
maxlen1 db 20
len1 db 0
arr1 db 20 dup ('$')

p2 label byte
maxlen2 db 20
actlen2 db 0
arr2 db 20 dup ('$')

arr3 db 30 dup('$')
arr4 db 20 dup('$')
arr5 db 20 dup('$')

F1=Help                                     Line:1   Col:1
```

```
File Edit Search View Options Help
C:\TASM\EXP6.ASM

arr3 db 30 dup('$')
arr4 db 20 dup('$')
arr5 db 20 dup('$')

msg10 db 10,13, "Enter first string::$"
msg11 db 10,13, "Enter second string::$"
msg12 db 10,13, "Length is ::$"
msg13 db 10,13, "Concatenated String is ::$"
msg14 db 10,13, "Copied String is ::$"
msg15 db 10,13, "Reversed String is ::$"
msg16 db 10,13, "Not Palindrome$"
msg17 db 10,13, "Yes! It is Palindrome$"

.code
mov ax, 0data
mov ds, ax
mov es, ax

;accept string1
disp1 msg10
mov ah, 0ah      ; 0ah is used to accept the string
lea dx, p1

F1=Help                                     Line:41  Col:1
```

```
File Edit Search View Options Help
C:\TASM\EXP6.ASM

    lea dx, p1
    int 21h

:length
    disp1 msg12
    mov al, len1
    call print

:accept string2
    disp1 msg11
    mov ah, 0ah
    lea dx, p2
    int 21h

:length
    disp1 msg12
    mov al, actlen2
    call print

:concatenation
    lea si, arr1
    lea di, arr3

```

F1=Help | Line:62 Col:1

```
File Edit Search View Options Help
C:\TASM\EXP6.ASM

:concatenation
    lea si, arr1
    lea di, arr3
    mov cl, len1
    cld
m1:movsb
    dec cl
    jnz m1
    lea si, arr2
    mov cl, actlen2
    cld
m2:movsb
    dec cl
    jnz m2
    disp1 msg13
    disp1 arr3

:copying
    lea si, arr1
    lea di, arr4
    mov cl, len1
    cld

```

F1=Help | Line:81 Col:1

```
File Edit Search View Options Help
C:\TASM\EXP6.ASM

    mov cl, len1
    cld
m3: movsb
    dec cl
    jnz m3
    disp1 msg14
    disp1 arr4
;reverse
    lea si, arr1
    lea di, arr5

    mov cl, len1
    mov ch, 00
    add si, cx
    dec si
m4: std
    lodsb
    cld
    stosb
    dec cl
    jnz m4

F1=Help | Line:101 Col:1
```

```
File Edit Search View Options Help
C:\TASM\EXP6.ASM

    disp1 msg15
    disp1 arr5
;palindrome
    lea si, arr1
    lea di, arr5
    mov cl, len1
    repe cmpsb
    jnz np
    disp1 msg17
    jmp m6
np: disp1 msg16

m6: mov ah, 4ch
    int 21h

print proc near
    add al, 30h
    cmp al, 39h
    jle l1

F1=Help | Line:122 Col:1
```

```
File Edit Search View Options Help
C:\TASM\EXP6.ASM

    jnz np
    disp1 msg17
    jmp m6
np:disp1 msg16

m6:mov ah, 4ch
    int 21h

print proc near
    add al, 30h
    cmp al, 39h
    jle l1
    add al, 07h
l1:mov dl, al

    mov ah, 02h
    int 21h
    ret
endp
end
```

F1=Help | Line:131 Col:1

## Output:

```
C:\TASM>exp6.exe

Enter first string::eveneve
Length is ::7
Enter second string::neven
Length is ::5
Concatenated String is ::eveneveneven
Copied String is ::eveneve
Reversed String is ::eveneve
Yes! It is Palindrome
C:\TASM>exp6.exe

Enter first string::bhavin
Length is ::6
Enter second string::patil
Length is ::5
Concatenated String is ::bhavinpatil
Copied String is ::bhavin
Reversed String is ::nivahb
Not Palindrome
C:\TASM>
```