Name: **Tushar Nankani** Roll No: **1902112** Batch: **C23**

**Microprocessor: Experiment 3**

**Aim:** Write a menu driven assembly language program to accept ,display the string, find the length of string, reverse the string, check the string is palindrome or not and scan the string to check whether a accepted letter is present or not

**Theory:**

A Macro is a set of instructions grouped under a single unit. It is another method for implementing modular programming in the 8086 microprocessors

A Macro can be defined in a program using the following assembler directives: MACRO (used after the name of Macro before starting the body of the Macro) and ENDM (at the end of the Macro). All the instructions that belong to the Macro lie within these two assembler directives

LEA (Load Effective Address) is a shift-and-add instruction. It was added to 8086 because hardware is there to decode and calculate adressing modes

string storage in memory,how is string declared,procedure near and far concept.

**String Instructions**

Each string instruction may require a source operand, a destination operand or both. For 32-bit segments, string instructions use ESI and EDI registers to point to the source and destination operands, respectively.

For 16-bit segments, however, the SI and the DI registers are used to point to the source and destination, respectively.

There are five basic instructions for processing strings. They are −

● MOVS − This instruction moves 1 Byte, Word or Doubleword of data from memory location to another.

● LODS − This instruction loads from memory. If theoperand is of one byte, it is loaded into the AL register, if the operandis one word, it is loaded into the AX register and a doubleword is loaded into the EAX register.

● STOS − This instruction stores data from register (AL, AX, or EAX) to memory.

● CMPS − This instruction compares two data items in memory. Data could be of a byte size, word or doubleword.

● SCAS − This instruction compares the contents of aregister (AL, AX or EAX) with the contents of an item in memory.

Each of the above instruction has a byte, word, and doubleword version, and string instructions can be repeated by using a repetition prefix.

These instructions use the ES:DI and DS:SI pair of registers, where DI and SI registers contain valid offset addresses that refers to bytes stored in memory. SI is normally associated with DS (data segment) and DI is always associated with ES

(extra segment).

The DS:SI (or ESI) and ES:DI (or EDI) registers point to the source and destination operands, respectively. The source operand is assumed to be at DS:SI (or ESI) and the destination operand at ES:DI (or EDI) in memory.

For 16-bit addresses, the SI and DI registers are used, and for 32-bit addresses, the ESI and EDI registers are used.

procedure : A procedure is group of instructions thatusually performs one task. It is a reusable section of a software program whichis stored in memory once but can be used as often as necessary. A procedure can be of two types. 1) Near Procedure 2) Far Procedure

Near Procedure: A procedure is known as NEAR procedure if is written(defined) in the same code segment which is calling that procedure. Only Instruction Pointer(IP register) contents will be changed in NEAR procedure.

FAR procedure : A procedure is known as FAR procedure if it is written (defined) in the different code segment than the calling segment. In this case both Instruction Pointer(IP) and the Code Segment(CS) register content will be changed.

Directives used for procedure:

PROC directive: The PROC directive is used to identify the start of a procedure. The PROC directive follows a name given to the procedure.After that the term FAR and NEAR is used to specify the type of the procedure.

ENDP Directive: This directive is used along with the name of the procedure to indicate the end of a procedure to the assembler.The PROC and ENDP directive are used to bracket a procedure.

CALL instruction and RET instruction:

The CALL instruction is used whenever we need to make a call to some procedure or a subprogram. Whenever a CALL is made, the following process takes place inside the microprocessor:

• The address of the next instruction that exists in the caller program (after the program CALL instruction) is stored in the stack.

• The instruction queue is emptied for accommodating the instructions of the procedure.

• Then, the contents of the instruction pointer (IP) is changed with the address of the first instruction of the procedure.

• The subsequent instructions of the procedure are stored in the instruction queue for execution.

The Syntax for the CALL instruction is as follows:

CALL subprogram\_name

CALL instruction : The CALL instruction is used to transfer execution to a procedure.It performs two operation.When it executes,first it stores the address of instruction after the CALL instruction on the stack.Second it changes the content of IP register in case of Near call and changes thecontent of IP register and cs register in case of FAR call.

There are two types of calls.

1)Near Call or Intra segment call.

2) Far call or Inter Segment call

Operation for Near Call : When 8086 executes a near CALL instruction, it decrements the stack pointer by 2 and copies the IP register contents on to the stack.Then it copies the address of the first instructionof the called procedure.

SP<--SP-2

IP<--stores onto stack

IP<--starting address of a procedure.

Operation of FAR CALL:When 8086 executes a far call, it decrements the stack pointer by 2 and copies the contents of CS registerto the stack. It the decrements the stack pointer by 2 again and copies the content of IP register to the stack.Finally it loads cs register with base address of segment having procedure and IP with address of first instruction in procedure.

SP<--sp-2 cs contents stored on stack

SP<--sp-2

IP<--contents stored on stack

CS<--Base address of segment having procedure IP<--address of first instruction in procedure

The RET instruction stands for return. This instruction is used at the end of the procedures or the subprograms. This instruction transfers the execution to the caller program. Whenever the RET instruction is called, the following process takes place inside the microprocessor:

• The address of the next instruction in the mainline program which was previously stored inside the stack is now again fetched and is placed inside the instruction pointer (IP).

• The instruction queue will now again be filled with the subsequent instructions of the mainline program.

The Syntax for the RET instruction is as follows:

RET

**Algorithm:**

1. Start

2. Initialize data segment through AX register in the DS register.

3. If choice =1 then goto step no.4 (accept the string)elsegoto the step no.14

4. Display the message “Enter the string”

5. Initialize the SI with 1000h(source index)

6. Initialize the DI with 2000h(destination index)

7. Accept the character through keyboard(AL=52h i.e ASCII hex value of ‘m’)

8. Move the AL contents to a location pointed by SI and DI.

9. Increment the CL register contents by 1

10. Increment SI and DI by 1

11. Repeat the step 7 to 10 till Enter key get pressed(i.e AL=0Dh ASCII hex value for enter key used to detect the end of the string.)

12. Decrement the CL by 1 ( to avoid the enter key as a part of the string to obtain correct length value)

13. Preserve the length in temporary variable say count from CL

14. If choice = 2 then goto step no.15 (display thestring)else goto step no. 21

15. Initialize SI again with 1000h (string source)

16. Move the content of location pointed by SI toDL

17. Display the character on the screen

18. Increment SI by 1and decrement the CL by 1

19. Repeat the step from 15 to 18 till Zero flag will come set (i.e CL reaches tozero).

20. Load the String length from count to CL register back. 21. If choice=3 (display the length of string) then goto step no. 22 else goto step no. 24

22. Move the contents of CL to AL and add 30h to AL

23. Move the AL contents to DL and display the length of string. 24. If choice=4 (Reverse the string) then goto step no. 24 else goto step no.30

25. Add CX (e.g CX=0005)to SI to make SI to point to the last letter of String.

26. Decrement to SI by 1 as SI will start from 0 index (e.x1000 to 10004 are the locations if string is of 5 letters)

27. Move the letter pointed by SI to DL and display it on the screen

28. Decrement the SI by 1 and decrement the CL by 1

29. Repeat step 25 to 29 till CL reaches to zero if zero flag get set.

30. If choice = 5 then goto step no. 31 else gotothe step no.43

31. Initialize SI with 1000h again

32. Initialize DI with 2000h again

33. Load CL with original length of string from count

34. Add DI with CX (e.x CX=0005)so DI will point to the last letter of string

35. Move the letter from location pointed by SI to AL

36. Move the letter from location pointed by DI to BL

37. Compare the AL and BL if zero flag is not setthen goto step no. 38 else goto step no 39

38. Display the string is not palindrome.

39. Increment the SI by 1 and Decrement DI by 1

40. Decrement CL by 1.

41. Repeat step no. 34 to 38 till CL reaches to zero

42. If zero flag is set then display the string ispalindrome.

43. If choice=6 then goto step no. 44 else goto stepno 49

44. Accept the letter to be searched in AL.

45. Load CL with original length of string from count

46. Initialize the DI with 2000h

47. Use REPNE SCASB instruction this instruction scans the string for a letter stops when it finds the first occurrence of a letter.It decrements CX also by 1 for every scan. When this instruction stops CX will contain value as position-1 value.

48. Obtain the position of a letter by subtractingtotal length-CX value and display it on the screen.

49. Stop.

**Programs and output**

;macro for printing a string

print macro m

mov ah,09h

mov dx,offset m

int 21h

endm

.model small

.data

menu db 10,13, "Menu: "

db 10,13, "1.Enter the string to find length "

db 10,13, "2.Check for palindrome "

db 10,13, "3.Display Entered String"

db 10,13, "4.Exit "

db 10,13, " "

db 10,13, "Your choice: $"

msg1 db 10,13, "Your choice is: $"

mc1 db 10,13, "case 1 $"

mc2 db 10,13, "case 2 $"

mc3 db 10,13, "case 3 $"

mc4 db 10,13, "Exiting the program $"

mc6 db 10,13, "Invalid choice....try again $"

mc7 db 10,13, "Exiting the program $"

empty db 10,13, " $"

str1 db 25,?,25 dup('$')

str2 db 25,?,25 dup('$')

len db ?

str3 db 0ah,0dh,"enter name:$"

;str4 db 25,?,25 dup('$')

mstring db 10,13, "Enter the string: $"

notpalin db 10,13, "String is not a palindrome. $"

palin db 10,13, "String is a palindrome. $"

mstring2 db 10,13, "Enter second string: $"

mlength db 10,13, "Length is: $"

scount db ?

;.code

start:

mov ax,@data

mov ds,ax

again:

print menu

call accept ;accept user choice

mov bl,al

case1:

cmp bl,"1" ;compare user choice with '1'

jne case2 ;if not equal,check for case 2

print mc1

print empty

print mstring

call accept\_string ;function call to accept a string

mov cl,str1+1 ;storing length in cl

mov bl,cl

print mlength

call display1 ;printing the length

print empty

jmp again ;printing the menu again

case2: cmp bl,"2" ;checking for case 2

jne case3 ;if not equal jump to case 3

print mc2

print empty

print mstring

call accept\_string

mov si,offset str1

mov cl,str1+1 ;store the length

mov ch,00h

mov len,cl

inc si

add si,cx ;si points to last

mov di,offset str1 ;di to start of string

add di,0002h ;di to actual start of string;

;in string 0th byte->size

;1st byte->length of string

;from 2nd byte->actual string

mov cl,len ;load counter

cmpagain: mov al,[si]

mov ah,[di]

inc di

dec si

cmp al,ah

jne nopalin

dec cl

jnz cmpagain

print palin

print empty

jmp again

nopalin: print notpalin

print empty

jmp again

case3:

cmp bl,"3" ;check for case 4

jne case4

print mc3

lea dx,str3

mov ah,09h

int 21h

call accept\_string

print empty

lea dx,str1+2

mov ah,09h

int 21h

case4: cmp bl,"4" ;check for case 6

jne case5 ;if not equal,default to case 7 and print the error message

print mc6

jmp exit

case5: print mc7 ;print error message

jmp again ;display the menu again

exit:

mov ah,4ch ;exit the program

int 21h

;accept procedure

accept proc near

mov ah,01

int 21h

ret

accept endp

display1 proc near

mov al,bl

mov bl,al

and al,0f0h

mov cl,04

rol al,cl

cmp al,09

jbe number

add al,07

number: add al,30h

mov dl,al

mov ah,02

int 21h

mov al,bl

and al,00fh

cmp al,09

jbe number2

add al,07

number2: add al,30h

mov dl,al

mov ah,02

int 21h

ret

display1 endp

accept\_string proc near

mov ah,0ah ;accept string from user function

mov dx,offset str1 ; store the string in memory pointed by "DX"

int 21h

ret

accept\_string endp

end start

end

