**EXPERIMENT 3**

**AIM:** Write an assembly program to count the number of characters in a string.

**PREREQUISITE**: TASM assembler

**THEORY**:

First we need to initialize count and set the count to 0. Then we need to just need to point it towards the string. If the string is null it then returns the count value i.e. the length of the string is 0. If the string is not null, we just have to increment the count. Then it moves to the next char till it encounters a ‘$’ symbol. In assembly language, to end the string we just have to insert a ‘$’ symbol. When the ‘$’ symbol is encountered, the value in count is the total length of the string.

**ALGORITHM:**

1. Start
2. Lets initialize the string in the data segment to find the length.
3. It should end with a “$” sign.
4. Assuming cs as code and ds as data.
5. In the code, check till the the ‘$’ sign is encountered.
6. If the ‘$’ sign is encountered, it will go to exit1.
7. Then increment the value of si and store it.
8. Also increase the value of len.
9. In exit1, it displays all the contents.
10. Stop.

**CODE:**

data segment

str1 db "hello"

len db ?

data ends

code segment

assume cs :code, ds :data

start:

mov ax , data

mov ds ,ax

LEA si ,str1

up: mov al, [si]

cmp al , "$"

jz exit1

inc si

inc len

jmp up

exit1:

add len,30

mov dl,len

mov ah,02h

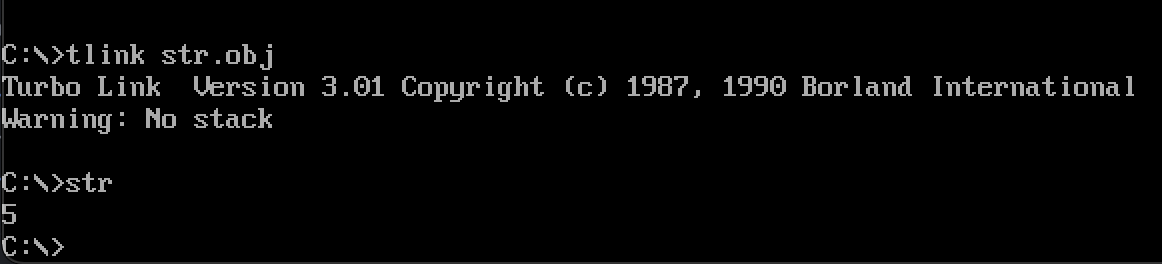
int 21h

mov ah,4ch

int 21h

code ends

end start

**OUTPUT:**

**CONCLUSION:**

From the experiment above we are able to know how to find the length of the string that has been initialized in the assembly program. And also we are able to learn how the strings end.

**Aim: To write an assembly program to reverse a string**

**Algorithm:**

1. Create a string

2. Traverse through the string

3. Push the characters in the stack

4. Count the number of characters

5. Load the starting address of the string

6. POP the top character of the stack until count is not equal to zero

7. Put the character and reduce the count and increase the address

8. Continue until the count is greater than zero

9. Load the effective address of the string in dx using LEA command

10.Print the sting by calling the interrupt with 9H in AH

11.The string must be terminated by the ‘$’ sign.

**Code:**

.MODEL SMALL

.STACK 100H

.CODE

Data Segment

msg db 0dh,0ah,"Enter a string: $"

result db 0dh,0ah,"The Reverse is: $"

newl db 0dh,0ah," $"

Data ends

MAIN PROC

assume DS:Data

Start:

mov ax,Data

mov DS,ax

mov dx,offset msg;add16

mov ah,09h

int 21h

MOV CH,0

MOV CL,0

INPUT:

MOV AH,1

INT 21H

CMP AL,13D

JE LINE

AND DX,0

MOV DL,AL

PUSH DX

INC CL

JMP INPUT

LINE:

MOV AH,2

MOV DL,0AH

INT 21H

MOV DL,0DH

INT 21H

mov dx,offset result;display result

mov ah,09h

int 21h

PRINT:

AND DX,0

POP DX

MOV AH,2

INT 21H

LOOP PRINT

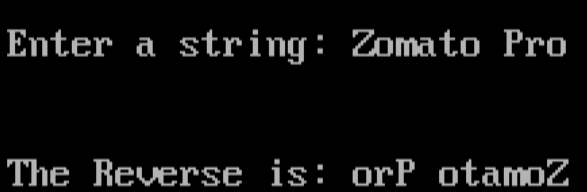
mov ax, 4c00h ;to come back to dos box

int 21h

MAIN ENDP

END MAIN

**Output:**

****

**CONCLUSION:**

By typing the ‘td block.exe’ command we are able to see how the transfer of block takes place from one location to another. The right side of the output shows all the values present in the registers. The top left-side just shows how the program works on each and every step. The down left part is the dump of the program. A message can also be generated after running the code that the block transfer was successful.

**Aim:** To write an assembly program to transfer a block of 4 bytes by using string instructions

**Prerequisite**: TASM assembler

**Theory:**

First take the string that we want to reverse it from the user. Then we need to copy the string from one memory location to another in reverse order and display it. We first copy the first two bytes of the string array as it is in the new string,since they remain the same for the reversed string. Then we position the SI pointer to the end of the given string and copy character by character in the new string in reverse order. Finally the new string is displayed,which is the reverse of the original string.

**Algorithm:**

1. Assign value 500 in SI and 600 in DI
2. Assign the value 0000 H to AX
3. Move the content of AX in DS
4. Move the content of AX in ES
5. Assign the value 0004 H to CX
6. Clear the directional flag
7. Repeat until CX=0, Move string block
8. Halt of the program

**CODE:**

DATA SEGMENT

BLK1 DB 10 DUP(?)

BLK2 DB 10 DUP(?)

DATA ENDS

CODE SEGMENT

START: ASSUME CS: CODE, DS: DATA

MOV AX, DATA

MOV DS, AX

LEA SI, BLK1

LEA DI, BLK2

MOV CL, 0AH

UP:

MOV AL, [SI]

MOV [DI], AL

INC DI

INC SI

DEC CL

JNZ UP

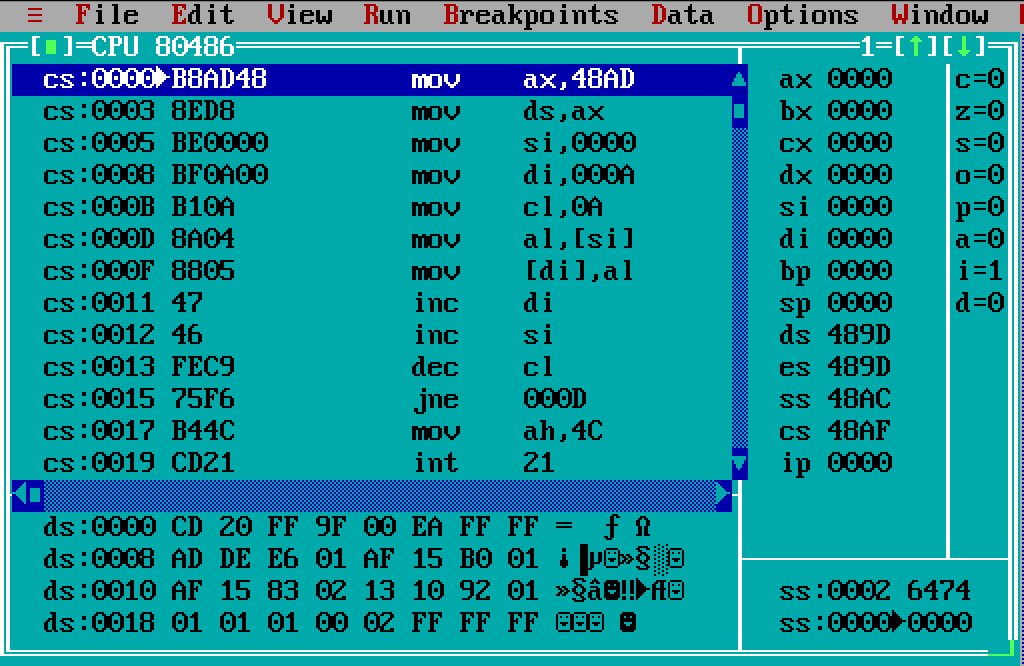
MOV AH, 04CH

INT 21H

CODE ENDS

END START

**OUTPUT:**



**Conclusion:**

To reverse a string in assembly language is really very easy. Just we need to move the contents from one location to another and then just order it in a reverse manner. By using the above logic we can easily reverse any string.