# Name: Shrinivas Hatyalikar

# Div : CS-B

# Roll No: 24

**Title:** Write 8086 ALP for following operations on the string entered by the user:

a. String length

b. Reverse of the String

c. Palindrome

**Aim:-** Write 64 bit ALU program for non overlap block transfer without string specific instruction

**Apparatus:**

* Core 2 duo/i3/i5/i7 - 64bit processor
* OS – ubuntu 32bit/64bit OS
* Assembler used –nasm (the netwide assembler)
* Editor Used – gedit

**Theory:**

**Procedure:**

A procedure refers to a set of instructions that are grouped together and can be called by a program or another procedure. It is designed to perform a single task and operate independently of the rest of the program.

**Macros:**

A macro is a named sequence of instructions that can be used throughout a program. In assembly language programming, macros are defined using directives such as %macro and %endmacro. A macro starts with the %macro directive and ends with the %endmacro directive.

To perform operations on a string entered by the user in 8086 assembly language, we can use various instructions like MOV, ADD, CMP, Jxx, LOOP, etc. These instructions can be used to perform tasks like string length calculation, string reversal, and checking if the string is a palindrome.

**a. String Length Calculation:**

To calculate the length of a string, we can count the number of characters until we reach the end of the string. We can do this by comparing each character to a null character (ASCII code 0) until we find the end of the string.

**b. String Reversal:**

To reverse a string, we can swap the characters at the beginning and end of the string, then move inward until we reach the middle of the string. The steps to reverse a string are:

**c. Palindrome:**

A string is a palindrome if it reads the same backward as forward. To check if a string is a palindrome, we can compare the first and last characters of the string, then move inward until we reach the middle of the string. The steps to check if a string is a palindrome are:

**Algorithm:**

1. **String Length Calculation:**
2. Load the starting address of the string into a register (e.g. SI or BX).
3. Initialize a counter (e.g. CX) to 0.
4. Load the first character of the string into a register (e.g. AL).
5. Compare the character to a null character (ASCII code 0).
6. If the character is not null, increment the counter and move to the next character.
7. If the character is null, stop counting and the counter will contain the length of the string.

**b. String Reversal:**

1. Load the starting address of the string into a register (e.g. SI or BX).
2. Load the ending address of the string into another register (e.g. DI).
3. Move the first character of the string into a register (e.g. AL).
4. Swap the first and last characters of the string.
5. Move inward and repeat step 4 until the middle of the string is reached.

**c. Palindrome:**

1. Load the starting address of the string into a register (e.g. SI or BX).
2. Load the ending address of the string into another register (e.g. DI).
3. Move the first character of the string into a register (e.g. AL).
4. Compare the first and last characters of the string.
5. If the characters are not the same, the string is not a palindrome.
6. Move inward and repeat steps 4-5 until the middle of the string is reached. If all characters match, the string is a palindrome.

**Code:**

%macro IO 4

mov rax,%1

mov rdi,%2

mov rsi,%3

mov rdx,%4

syscall

%endmacro

section .data

m1 db "enter string",10

l1 equ $-m1

m2 db "Entered :"

l2 equ $-m2

m3 db "Length is "

l3 equ $-m3

m4 db "Assignment 6: accept a string and to display its length" ,10

l4 equ $-m4

section .bss

string resb 50

strl equ $-string

count resb 1

\_output resb 20

section .text

global \_start

\_start:

IO 1,1,m4,l4

IO 1,1,m1,l1

input:

IO 0,0,string,strl

mov [count],rax

output:

IO 1,1,m2,l2

IO 1,1,string,strl

IO 1,1,m3,l3

mov rax,[count]

call hex\_to\_dec

IO 1,1,\_output,16

exit:

mov rax, 60

mov rdi, 0

syscall

hex\_to\_dec:

mov rsi,\_output

mov rcx,2

letter2:

xor rdx,rdx

mov rbx,10

div rbx

cmp dl,09h

jbe add30

add30:

add dl,30h

mov [rsi],dl

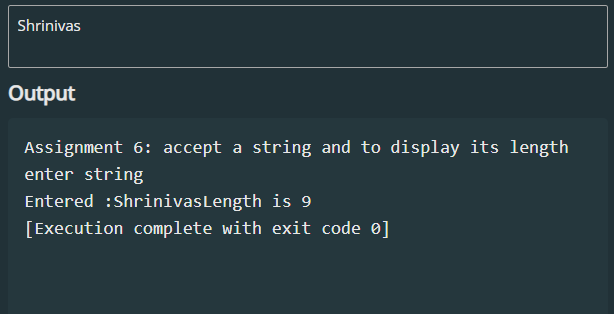
dec rsi

dec rcx

jnz letter2

ret

**Output:**

****

**2)Reverse the String:**

%macro IO 4

mov rax,%1

mov rdi,%2

mov rsi,%3

mov rdx,%4

syscall

%endmacro

section .data

m1 db "Assigment 6: Reverse The String",10

l1 equ $-m1

m3 db "Reversed String Is: ",10

l3 equ $-m3

section .bss

rev\_str resb 15

msg resb 50

msg\_len equ $-msg

%macro read 2

mov rax,0

mov rdi,0

mov rsi,%1

mov rdx,%2

syscall

%endmacro

section .text

global \_start

\_start:

IO 1,1,m1,l1

IO 1,1,m3,l3

read msg,msg\_len

mov rcx, 0

mov rsi, msg

count\_len:

cmp byte [rsi+rcx], 0

je reverse\_str

inc rcx

jmp count\_len

reverse\_str:

mov rdi, rev\_str

mov rdx, rcx

dec rdx

copy\_char:

mov al, [rsi+rdx]

mov [rdi], al

inc rdi

dec rdx

cmp rdx, 0

jge copy\_char

mov rax, 1

mov rdi, 1

mov rsi, rev\_str

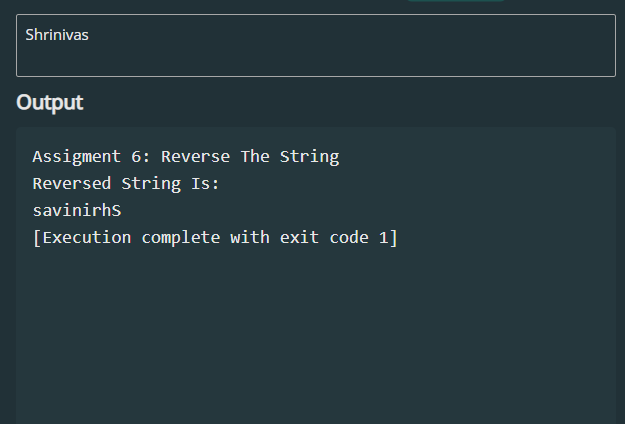
mov rdx, rcx

syscall

mov rax, 60

syscall

**Output:**

****

**3)String is palindrome or not.**

%macro IO 4

mov rax,%1

mov rdi,%2

mov rsi,%3

mov rdx,%4

syscall

%endmacro

section .data

msg1 db 'String is palindrome'

msg1\_len equ $-msg1

msg2 db 'String is not palindrome'

msg2\_len equ $-msg2

section .bss

msg resb 50

msg\_len equ 50

%macro read 2

mov rax,0

mov rdi,0

mov rsi,%1

mov rdx,%2

syscall

%endmacro

section .text

global \_start

\_start:

read msg, msg\_len

call Palindrome

mov rax, 60

syscall

Palindrome:

mov rsi, msg

loop1:

mov ax,[rsi]

cmp al, 0

je label1

inc rsi

jmp loop1

label1:

mov rdi, msg

dec rsi

loop2:

cmp rsi, rdi

jle output1

mov al, [rsi]

mov bl, [rdi]

cmp al, bl

jne output2

inc rdi

dec rsi

jmp loop2

output1:

mov rax, 1

mov rdi, 1

mov rsi, msg1

mov rdx, msg1\_len

syscall

ret

output2:

mov rax, 1

mov rdi, 1

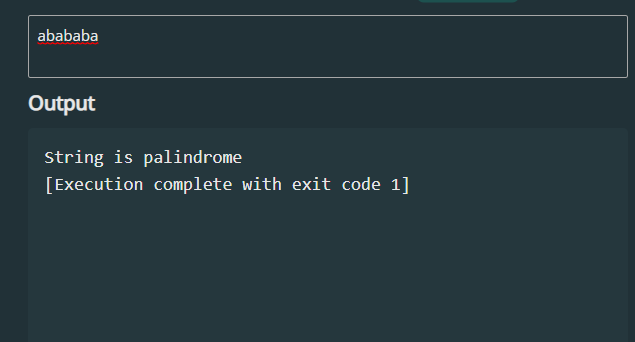
mov rsi, msg2

mov rdx, msg2\_len

syscall

ret

**Output:**



**Conclusion:**

In this assignment, we covered various string manipulation operations such as calculating the length of a string, reversing a string, and checking if a string is a palindrome. For reversing a string, we first took input from the user and then determined its length. We then reversed the string character by character using a loop and displayed the resulting reversed string using the IO macro. The Palindrome subroutine used two loops to compare the characters from the beginning and end of the string. It kept comparing the characters until it either found a mismatch or reached the middle of the string. Depending on whether the input string was a palindrome or not, the program displayed the appropriate message.