section .data

welmsg db 10,'Welcome to Assembly Language Program',10

welmsg\_len equ $-welmsg

accmsg db 10,'Please enter the string:: '

accmsg\_len equ $-accmsg

lenmsg db 10,'Length of the string is:: '

lenmsg\_len equ $-lenmsg

revmsg db 10,'Reverse string is:: '

revmsg\_len equ $-revmsg

palimsg db 10,'String is Palindrome....'

palimsg\_len equ $-palimsg

npalimsg db 10,'String is not Palindrome....'

npalimsg\_len equ $-npalimsg

thankmsg db 10,10,'Thank you for using Program',10

thankmsg\_len equ $-thankmsg

section .bss

accbuff resb 50

accbuff\_len equ $-accbuff

revbuff resb 50

actlen resq 1

dnumbuff resb 16

%macro linuxsyscall 4

mov rax,%1

mov rdi,%2

mov rsi,%3

mov rdx,%4

syscall

%endmacro

section .text

global \_start

\_start:

linuxsyscall 01,01,welmsg,welmsg\_len ;display welcome message

linuxsyscall 01,01,accmsg,accmsg\_len ;display enter stringmessage

linuxsyscall 0,0,accbuff,accbuff\_len ;accept string

dec rax ;calculate actual length ofstring entered by decrementing rax as it includes ENTER char

mov [actlen],rax;Following lines displays length of string

linuxsyscall 01,01,lenmsg,lenmsg\_len ;display lengthmessage

mov rbx,[actlen] ;load count in rbx for display

call disp64\_proc ;call display procedure todisplay RBX contents

;Following lines will reverse the string and display it

mov rsi,accbuff ;Point rsi to first byte of original string

mov rdi,revbuff ;Point rdi to first byte of reverse string buffer

mov rcx,[actlen] ;Load count in RCX

add rsi,rcx ;Add count in rsi to point to next byte after original string

dec rsi ;Decrement rsi to point to last char of original string

again:

mov al,[rsi] ;Copy char from originalstring and

mov [rdi],al ;Copy to reverse string buffer

dec rsi ;decrement pointer to reverse

inc rdi ;increment pointer

loop again

linuxsyscall 01,01,revmsg,revmsg\_len ;display reversestring message

linuxsyscall 01,01,revbuff,[actlen] ;display reversestring

;Following lines will check for palindrome

mov rsi,accbuff ;Point rsi to first byte oforiginal string

mov rdi,revbuff ;Point rdi to first byte ofreverse string buffer

mov rcx,[actlen] ;Load count in RCX

shr rcx,1 ;Divide count by 2

repe cmpsb ;Compare original and reversestring char by char

jne nopali ;If last comparison is notequal, string is not palindrome

linuxsyscall 01,01,palimsg,palimsg\_len ;display stringpalindrome message

jmp exit

nopali:

linuxsyscall 01,01,npalimsg,npalimsg\_len ;display string notpalindrome message

exit:

linuxsyscall 01,01,thankmsg,thankmsg\_len

mov rax,60 ;Exit

mov rbx,00

syscall

disp64\_proc:

mov rdi,dnumbuff ;point esi to buffer

mov rcx,16 ;load number of digits to display

dispup1:

rol rbx,4 ;rotate number left by four bits

mov dl,bl ;move lower byte in dl

and dl,0fh ;mask upper digit of byte in dl

cmp dl,09h ;compare with 9

jbe dispskip1 ;if less than or equal to 9 skip adding 07more

add dl,07h

dispskip1:

add dl,30h ;add 30h to calculate ASCII code

mov [rdi],dl ;store ASCII code in buffer

inc rdi ;point to next byte

loop dispup1 ;decrement the count of digits to display

;if not zero jump to repeat

linuxsyscall 01,01,dnumbuff,16

ret