**ASSIGNMENT 1**

**ASSIGNMENT 2**

;Assignment No.: 2

;Assignment Name:- **Write x86/64 ALP to accept a string and to display its length**

;----------------------------------------------------------------------------------

section .data

msg db 10,10, "Enter the string: "

msg\_len equ $-msg

sms db 10,10,"The length of string is:"

sms\_len equ $-sms

;----------------------------------------------------------------------------------

section .bss

string resb 50

stringl equ $-string

count resb 1

char\_ans resb 2

;----------------------------------------------------------------------------------

%macro print 2

Mov RAX, 1

Mov RDX, 1

Mov RSI, %1

Mov RDX, %2

syscall

%endmacro

%macro Read 2

Mov RAX ,0

Mov RDI, 0

Mov RSI, %1

Mov RDX, %2

syscall

%endmacro

%macro exit 0

Mov RAX, 60

Mov RDX, 0

Syscall

%endmacro

;-------------------------------------------------------------------------------------

section .text

global \_start

\_start:

print msg , msg\_len

Read string, stringl

Mov[count],RAX

print sms,sms\_len

Mov RAX,[count]

DEC RAX

call Display

exit

;-------------------------------------------------------------------------------------

Display:

Mov RBX, 10

Mov RCX, 2

Mov RSI ,char\_ans+1

cnt:

Mov RDX,0

div RBX

CMP dl, 09h

JBE add30

add dl,07h

add30:

add dl, 30h

Mov [rsi],dl

DEC RSI

DEC RCX

JNZ cnt

print char\_ans,2

ret

;-----------------------------------------------------------------------------------

**ASSIGNMENT 3**

section .data

arr dq 123H, -120H, 12H, -20H, -20H, 66H, -918737H, 9922H, -293H, -987H

n equ 10

msg db"Assignment no.3 : Calculate the count of positive and negative numbers in array",10

msg\_len equ $-msg

msg1 db"The positive numbers are: "

msg1\_len equ $-msg1

msg2 db 10,"The negative numbers are: "

msg2\_len equ $-msg2

msg3 db 10," "

msg3\_len equ $-msg3

%macro read 2

mov rax,0

mov rdi,0

mov rsi,%1

mov rdx,%2

syscall

%endmacro

%macro print 2

mov rax,1

mov rdi,1

mov rsi,%1

mov rdx,%2

syscall

%endmacro

section .bss

charans resb 4

result resq 1

p\_count resb 8

n\_count resb 8

section .text

global \_start

\_start:

print msg,msg\_len

mov rsi,arr

mov rcx,n

mov rbx,0;

mov rdx,0;

next\_num:

mov rax,[rsi]

Rol rax,1

jc negative

positive:

inc rbx

jmp next1

negative:

inc rdx

next1:

add rsi,8

dec rcx

jnz next\_num

mov [p\_count],rbx

mov [n\_count],rdx

print msg1,msg1\_len

mov rax,[p\_count]

call Display

print msg2,msg2\_len

mov rax,[n\_count]

call Display

print msg3,msg3\_len

mov rax,60

mov rdi,0

syscall

Display:

mov rbx,16

mov rcx,2

mov rsi,charans+1

next:

mov rdx,0

div rbx

cmp dl,09h

jbe add30

add dl,07h

add30:

add dl,30h

mov [rsi], dl

dec rsi

dec rcx

jnz next

print charans,2

ret

**ASSIGNMENT 4**

section .data

nline db 10,10

nline\_len equ $-nline

ano db 10,"Assign. No. 4"

db 10,"---------------------------"

ano\_len equ $-ano

hmsg db 10,"Enter 4 digit HEX No.:"

hmsg\_len equ $-hmsg

ebmsg db 10,"The Equivalent BCD No. is:"

ebmsg\_len equ $-ebmsg

emsg db 10,"Invalid Number Input",10

emsg\_len equ $-emsg

section .bss

buf resb 6

char\_ans resb 4

ans resw 1

%macro Print 2

mov rax,1

mov rdi,1

mov rsi,%1

mov rdx,%2

syscall

%endmacro

%macro Read 2

mov rax,0

mov rdi,0

mov rsi, %1

mov rdx, %2

syscall

%endmacro

%macro Exit 0

Print nline,nline\_len

mov rax,60

mov rdi,0

syscall

%endmacro

section .text

global \_start

\_start:

Print ano,ano\_len

hex\_bcd:

Print hmsg,hmsg\_len

call Accept\_16

mov ax,bx

mov bx,10

xor bp,bp

back:

xor dx,dx

div bx

push dx

inc bp

cmp ax,0

jne back

Print ebmsg,ebmsg\_len

back1:

pop dx

add dl,30h

mov [char\_ans],dl

Print char\_ans,1

dec bp

jnz back1

Exit

Accept\_16:

Read buf,5

mov rcx,4

mov rsi,buf

xor bx,bx

next\_byte:

shl bx,4

mov al,[rsi]

cmp al,'0'

jb error

cmp al,'9'

jbe sub30

cmp al,'A'

jb error

cmp al,'F'

jbe sub37

cmp al,'a'

jb error

cmp al,'f'

jbe sub57

error:

Print emsg,emsg\_len

Exit

sub57:

sub al,30h

sub37:

sub al,07h

sub30:

sub al,30h

add bx,ax

inc rsi

dec rcx

jnz next\_byte

RET

**ASSIGNMENT 5**

section .data

nline db 10,10

nline\_len equ $-nline

ano db 10,"ASSIGNMENT NO: 5", 10

db "----------------------------------------------------------------", 10

db " Conversion from BCD to HEX ", 10

db "----------------------------------------------------------------", 10

ano\_len equ $-ano

bmsg db 10,"Enter 5 digit BCD number::"

bmsg\_len equ $-bmsg

ehmsg db 10,"The Equivalent HEX number is::"

ehmsg\_len equ $-ehmsg

section .bss

buf resb 6

char\_ans resb 5

ans resw 1

%macro Print 2

mov rax, 1

mov rdi, 1

mov rsi, %1

mov rdx, %2

syscall

%endmacro

%macro Read 2

mov rax, 0

mov rdi, 0

mov rsi, %1

mov rdx, %2

syscall

%endmacro

%macro Exit 0

mov rax, 60

xor rdi, rdi

syscall

%endmacro

section .text

global \_start

\_start:

Print ano, ano\_len

BCD\_HEX:

Print bmsg, bmsg\_len

Read buf, 6

mov rsi, buf

xor ax, ax

mov rbp, 5

mov rbx, 10

next:

xor cx, cx

mul bx

mov cl, [rsi]

sub cl, 30h

add ax, cx

inc rsi

dec rbp

jnz next

mov [ans], ax

Print ehmsg, ehmsg\_len

mov ax, [ans]

call Disp\_16

Exit

Disp\_16:

mov rsi, char\_ans + 4

mov rcx, 4

mov rbx, 16

next\_digit:

xor rdx, rdx

div rbx

cmp dl, 9

jbe add30

add dl, 07h

add30:

add dl, 30h

dec rsi

mov [rsi], dl

dec rcx

jnz next\_digit

Print char\_ans, 4

ret

**OR :**

section .data

nline db 10,10

nline\_len equ $-nline

ano db 10," Assignment no :4",

db 10,"------------------------------------------------------------",

db 10," Assignment Name:Conversion From HEX to BCD and BCD to HEX Number.",

db 10,"----------------------------------------------------------",10

ano\_len equ $-ano

menu db 10,"1.Hex To BCD.",

db 10,"2.BCD To Hex.",

db 10,"3.Exit."

db 10,"Enter Your Choice::"

menu\_len equ $-menu

hmsg db 10,"Enter 4 digit Hex Number::"

hmsg\_len equ $-hmsg

bmsg db 10,"Enter 5 digit BCD Number::"

bmsg\_len equ $-bmsg

ebmsg db 10,"The Equivalent BCD Number is::"

ebmsg\_len equ $-ebmsg

ehmsg db 10,"The Equivalent Hex Number is::"

ehmsg\_len equ $-ehmsg

emsg db 10,"INVALID NUMBER INPUT",10

emsg\_len equ $-emsg

;------------------------------------------------------------------------------

section .bss

buf resb 6 ;a variable which will store the accepted number from the user. 6 bytes are reserved for it. 5 digits of BCD+enter key

char\_ans resb 4

ans resw 1

;-----------------------------------------------------------------------------

%macro Print 2

MOV RAX,1

MOV RDI,1

MOV RSI,%1

MOV RDX,%2

syscall

%endmacro

%macro Read 2

MOV RAX,0

MOV RDI,0

MOV RSI,%1

MOV RDX,%2

syscall

%endmacro

%macro Exit 0

Print nline,nline\_len

MOV RAX,60

MOV RDI,0

syscall

%endmacro

;---------------------------------------------------------------

section .text

global \_start

\_start:

Print ano,ano\_len

MENU: Print menu,menu\_len

Read buf,2 ;accept choice i.e 1 digit+enter

mov al,[buf] ;contains only digit character

c1: cmp al,'1'

jne c2

call HEX\_BCD

jmp MENU

c2: cmp al,'2'

jne c3

call BCD\_HEX

jmp MENU

c3: cmp al,'3'

jne invalid

Exit

invalid:

Print emsg,emsg\_len

jmp MENU

;---------------------------------------------------------------

HEX\_BCD: ; procedure to convert Hex to BCD

Print hmsg,hmsg\_len

call Accept\_16 ;accept 4 digit hex number

mov ax,bx ;mov hex number in ax

mov bx,10 ;for divide hex number by 10

xor bp,bp ;counter

back: xor dx,dx ;as dx each time contains remainder

div bx ;divide ax by 10 ax=Q,dx=R

push dx ;push dx on stack as it is bcd

inc bp ;inc counter by 1. It counts the no of digits pushed on the stack

cmp ax,0 ;compare whether Q is 0 if 0 means number get over

jne back ;mov to conversion of quotient

Print ebmsg,ebmsg\_len

back1: pop dx ;pop last digit pushed on stack

add dl,30h ;add 30 to digit to make them decimal

mov [char\_ans],dl ;print individual digit

Print char\_ans,1

dec bp ;counter decrement . it checks whehter all contents of stack are popped out

jnz back1 ;mov to next digit

RET

;---------------------------------------------------------------

BCD\_HEX:

Print bmsg,bmsg\_len

Read buf,6 ;5 digit + 1 enter

mov rsi,buf ;Points at the start of buffer

xor ax,ax ;Previous digit =0

mov rbp,5 ;counter

mov rbx,10 ;multiplier

next: xor cx,cx ;contains next digit each time

mul bx ;(ax\*bx)+cl

mov cl,[rsi]

sub cl,30h

add ax,cx

inc rsi ;Point at the next digit

dec rbp

jnz next

mov [ans],ax ;store ax in ans because ax get change in Print macro

Print ehmsg,ehmsg\_len

mov ax,[ans]

call Disp\_16 ;Print hex number

RET

;---------------------------------------------------------------

Disp\_16: ;Hex to Ascii(character) display

MOV RSI,char\_ans+3

MOV RCX,4 ;counter

MOV RBX,16 ;Hex no

next\_digit:

XOR RDX,RDX

DIV RBX

CMP DL,9

JBE add30

ADD DL,07H

add30 :

ADD DL,30H

MOV [RSI],DL

DEC RSI

DEC RCX

JNZ next\_digit

Print char\_ans,4

ret

;-------------------------------------------------------------------

Accept\_16: ;Ascii(character) to hex number input

Read buf,5 ;4 digits of Hex and one enter key

MOV RCX,4 ; rcx is a counter for 4 digits of Hex

MOV RSI,buf ; base address

XOR BX,BX ;making the contents of bx =0

next\_byte:

SHL BX,4 ; the earlier group of 4 bits will get discarded and new group of 4 bits will get added in on right side

MOV AL,[RSI] ; content of hex no goes to AL

CMP AL,'0' ; the hex no gets compared with 0

JB error ; jump if below i.e. if a negative no go to error

CMP AL,'9' ; hex no is compared with 9, if its less than or equal to it, then its a valid hex no.

JBE sub30 ; hence subtract 30

CMP AL,'A' ; hex no gets compared with 'A'

JB error

CMP AL,'F'

JBE sub37 ; subtract 37 if below F, sice its a valid hex no.

CMP AL,'a'

JB error

CMP AL,'f'

JBE sub57 ; 57 will get subtracted if the hex no. is between 'a' to'f'

error:

Print emsg,emsg\_len

Exit

sub57: SUB AL,20H ; this and the following 2 instructions will get executed serially. Hence split up of 57 as 20, 7 and 30.

sub37: SUB AL,07H

sub30: SUB AL,30H

ADD BX,AX ; this now valid hex no is added to contents of bx. after 4 such iterations we will get a binary equivalent of the hex no

INC RSI ; increment rsi so that it points to the next hex digit

DEC RCX ; decrement the counter rcx since its a counter for the no of digits of hex

JNZ next\_byte ; keep going in a loop till the counter=0

RET

;-------------------------------------------------------------------------------------

**ASSIGNMENT 6**

section .data

sblock db 10h,20h, 30h, 40h,50h

dblock db 0h,0h,0h,0h,0h

msg db "The source array is: "

msg\_len equ $-msg

msg1 db 10, "The destination array is: "

msg1\_len equ $-msg1

msg2 db "",10

msg2\_len equ $-msg2

msg3 db "After block transfer, the arrays are :",10

msg3\_len equ $-msg3

space db " "

%macro read 2

mov rax,0

mov rdi,0

mov rsi,%1

mov rdx,%2

syscall

%endmacro

%macro print 2

mov rax,1

mov rdi,1

mov rsi,%1

mov rdx,%2

syscall

%endmacro

section .bss

charans resb 4

result resq 1

section .text

global \_start

\_start:

print msg,msg\_len

mov rsi,sblock

call Display\_block

print msg1, msg1\_len

mov rsi,dblock

call Display\_block

print msg2,msg2\_len

call Block\_transfer

print msg3,msg3\_len

print msg,msg\_len

mov rsi,sblock

call Display\_block

print msg1, msg1\_len

mov rsi,dblock

call Display\_block

print msg2,msg2\_len

mov rax,60

mov rdi,0

syscall

Display:

mov rbx,16

mov rcx,2

mov rsi,charans+1

next:

mov rdx,0

div rbx

cmp dl,09h

jbe add30

add dl,07h

add30:

add dl,30h

mov [rsi], dl

dec rsi

dec rcx

jnz next

print charans,2

ret

Display\_block :

mov rbp,5

next\_num:

mov al,[rsi]

push rsi

call Display

print space,1

pop rsi

inc rsi

dec rbp

jnz next\_num

ret

Block\_transfer:

mov rsi,sblock

mov rdi,dblock

mov rcx,5

nextb:

mov al,[rsi]

mov [rdi],al

inc rsi

inc rdi

dec rcx

jnz nextb

ret

**ASSIGNMENT 7**

; Assignment No.07

; **Overlapped Block Transfer With String Instructions**

section .data

nline db 10,10

nline\_len equ $-nline

space db " "

ano db 10,"Assignment No.07",

db 10,"Block Transfer-Overlapped With String Instructions",

ano\_len equ $-ano

bmsg db 10,"Before Transfer :: "

bmsg\_len equ $-bmsg

amsg db 10,"After Transfer :: "

amsg\_len equ $-amsg

smsg db 10,"Source Block : "

smsg\_len equ $-smsg

dmsg db 10,"Destination Block : "

dmsg\_len equ $-dmsg

sblock db 11h,22h,33h,44h,55h

dblock times 5 db 0

section .bss

char\_ans resB 2

%macro Print 2

mov rax,1

mov rdi,1

mov rsi,%1

mov rdx,%2

syscall

%endmacro

%macro Read 2

mov rax,0

mov rdi,0

mov rsi,%1

mov rdx,%2

syscall

%endmacro

%macro Exit 0

Print nline,nline\_len

mov rax,60

mov rdi,0

syscall

%endmacro

section .text

global \_start

\_start:

Print ano,ano\_len

Print bmsg,bmsg\_len

Print smsg,smsg\_len

mov rsi,sblock

call disp\_block

Print dmsg,dmsg\_len

mov rsi,dblock-2

call BT\_OS

Print amsg,amsg\_len

Print smsg,smsg\_len

mov rsi,sblock

call disp\_block

Print dmsg,dmsg\_len

mov rsi,dblock-2

call disp\_block

Exit

BT\_OS:

mov rsi,sblock+4

mov rdi,dblock+2

mov rcx,5

STD ; STD(Set Direction Flag) / dec rsi,dec rdi

REP MOVSB ; MOVSB(Move String Bytes) / mov al,[rsi], mov [rdi],al

; REP(Repeat) / dec rcx,jnz back

ret

disp\_block:

mov rbp,5

next\_num:

mov al,[rsi]

push rsi

call Disp\_8

Print space,1

pop rsi

inc rsi

dec rbp

jnz next\_num

ret

Disp\_8:

mov rsi,char\_ans+1

mov rcx,2 ; counter

mov rbx,16 ; Hexadecimal Number

next\_digit:

xor rdx,rdx

div rbx

cmp dl,9

jbe add30

add dl,07h

add30:

add dl,30h

mov [rsi],dl

dec rsi

dec rcx

jnz next\_digit

Print char\_ans,2

ret

**OR**

%macro print 2

mov rax, 1

mov rdi, 1

mov rsi, %1

mov rdx, %2

syscall

%endmacro

%macro exit 0

mov rax, 60

mov rdi, 0

syscall

%endmacro

section .data

bmsg db 10, "Before transfer: "

bmsg\_len equ $-bmsg

amsg db 10, 10, "After transfer: "

amsg\_len equ $-amsg

sblock db 11h, 22h, 33h, 44h, 55h

dblock times 5 db 0

smsg db 10, "Source block: "

smsg\_len equ $-smsg

dmsg db 10, "Destination block: "

dmsg\_len equ $-dmsg

space db " "

space\_len equ $-space

section .bss

char\_ans resb 2

section .text

global \_start

\_start:

print bmsg, bmsg\_len

print smsg, smsg\_len

mov rsi, sblock

call block\_display

print dmsg, dmsg\_len

mov rsi, dblock - 2

call block\_display

call block\_transfer

print amsg, amsg\_len

print smsg, smsg\_len

mov rsi, sblock

call block\_display

print dmsg, dmsg\_len

mov rsi, dblock - 2

call block\_display

exit

block\_transfer:

mov rsi, sblock + 4

mov rdi, dblock + 2

mov rcx, 5

back:

std

rep movsb

ret

block\_display:

mov rbp, 5

next\_num:

mov al, [rsi]

push rsi

call Disp\_8

print space, space\_len

pop rsi

inc rsi

dec rbp

jnz next\_num

ret

Disp\_8:

mov rsi, char\_ans + 1

mov rcx, 2

mov rbx, 16

next\_digit:

xor rdx, rdx

div rbx

cmp dl, 9

jbe add30

add dl, 07H

add30:

add dl, 30H

mov [rsi], dl

dec rsi

dec rcx

jnz next\_digit

print char\_ans, 2

ret

**ASSIGNMENT 8**

section .data

msg1 db "Processor is in protected mode",10

msg1\_len equ $-msg1

msg2 db "Processor is not in protected mode",10

msg2\_len equ $-msg2

mgdt db "Value of GDTR: "

mgdt\_len equ $-mgdt

mline db "",10

mline\_len equ $-mline

mldt db "Value of LDTR: "

mldt\_len equ $-mldt

midt db "Value of IDTR: "

midt\_len equ $-midt

mmsw db "Value of MSW: "

mmsw\_len equ $-mmsw

%macro read 2

mov rax,0

mov rdi,0

mov rsi,%1

mov rdx,%2

syscall

%endmacro

%macro print 2

mov rax,1

mov rdi,1

mov rsi,%1

mov rdx,%2

syscall

%endmacro

%macro exit 0

mov rax,60

mov rdi,0

syscall

%endmacro

section .bss

LDTR resw 1

GDTR resw 3

MSW resw 1

IDTR resw 3

TR resw 1

charans resb 4

section .text

global \_start

\_start:

SMSW [MSW]

mov ax,[MSW]

shr ax,1

jc pmode

print msg2,msg2\_len

pmode: print msg1,msg1\_len

jmp next

next:

SGDT [GDTR]

SLDT [LDTR]

STR [TR]

SIDT [IDTR]

SMSW [MSW]

print mgdt,mgdt\_len

mov ax,[GDTR+4]

call display

mov ax,[GDTR+2]

call display

mov ax,[GDTR]

call display

print mline,mline\_len

print mldt,mldt\_len

mov ax,[LDTR]

call display

print mline,mline\_len

print mmsw,mmsw\_len

mov ax,[MSW]

call display

exit

display:

mov rbx,16

mov rcx,2

mov rsi,charans+1

next1:

mov rdx,0

div rbx

cmp dl,09h

jbe add30

add dl,07h

add30:

add dl,30h

mov [rsi], dl

dec rsi

dec rcx

jnz next1

print charans,2

ret