**Assignment number: 3**

**Subject: MICROPROCESSOR LAB**

Name: ***RIA MITTAL***

Class: ***SECOND YEAR ENGINEERING***

Division: ***B***

Roll no: ***222008***

Batch: ***B1***

**PROBLEM STATEMENT:**

Write X86/64 ALP to convert 4-digit Hex number into its equivalent BCD number and 5-digit BCD number into its equivalent HEX number. Make your program user friendly to accept the choice from user for:

(a) HEX to BCD b) BCD to HEX (c) EXIT.

Display proper strings to prompt the user while accepting the input and displaying the result. (wherever necessary, use 64-bit registers)

**Code:**

%macro scall 4

mov rax,%1

mov rdi,%2

mov rsi,%3

mov rdx,%4

syscall

%endmacro

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

section .data

m1 db "enter choice ",10,13

db "1.)HEX TO BCD",10,13

db "2.)BCD TO HEX",10,13

db "3.)EXIT",10

m1l equ $ -m1

m2 db "enter 4 digit hex number",10

m2l equ $ -m2

m3 db "enter 5 digit bcd number",10

m3l equ $ -m3

m4 db "equivalent bcd number is ",10

m4l equ $ -m4

m5 db "equivalent hex number is ",10

m5l equ $-m5

new db " ",10

newl equ $ -new

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

section .bss

choice resb 2

buf resb 6

bufl equ $ -buf

digit resb 1

ans resw 16

fact resw 16

chans resb 5

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

section .text

accept:

mov rsi,buf

back: rol rbx,04

mov al,[rsi]

cmp al,39h

jbe next

sub al,07h

next: sub al,30h

add rbx,rax

inc rsi

dec rcx

jnz back

ret

display:

ab:

rol bx,04h

mov dl,bl

and dl,0Fh

cmp dl,09h

jbe nex

add dl,07h

nex: add dl,30h

mov [rsi],dl

inc rsi

dec rcx

jnz ab

ret

global \_start

\_start:

scall 1,1,m1,m1l

scall 0,0,choice,2

mov al,[choice]

cmp al,'1'

je case1

cmp al,'2'

je case2

cmp al,'3'

je case3

case1:

scall 1,1,m2,m2l

scall 0,0,buf,bufl

mov rax,0

mov rcx,4

call accept

mov ax,bx

mov rbx,10

bk1: mov rdx,0

div rbx

push dx

inc byte[digit]

cmp rax,0

jne bk1

scall 1,1,m4,m4l

top: pop dx

add dl,30h

mov [chans],dl

scall 1,1,chans,1

dec byte[digit]

jnz top

scall 1,1,new,newl

jmp exit

case2:

scall 1,1,m3,m3l

scall 0,0,buf,6

mov rsi,buf+4

mov word[fact],1

mov rcx,5

mov rbx,0

up:

mov rax,0

mov al,[rsi]

sub al,30h

mul word[fact]

add bx,ax

mov ax,10

mul word[fact]

mov [fact],ax

dec rsi

dec rcx

jnz up

scall 1,1,m5,m5l

mov rsi,ans

mov rcx,4

call display

scall 1,1,ans,4

scall 1,1,new,newl

jmp exit

case3:

jmp exit

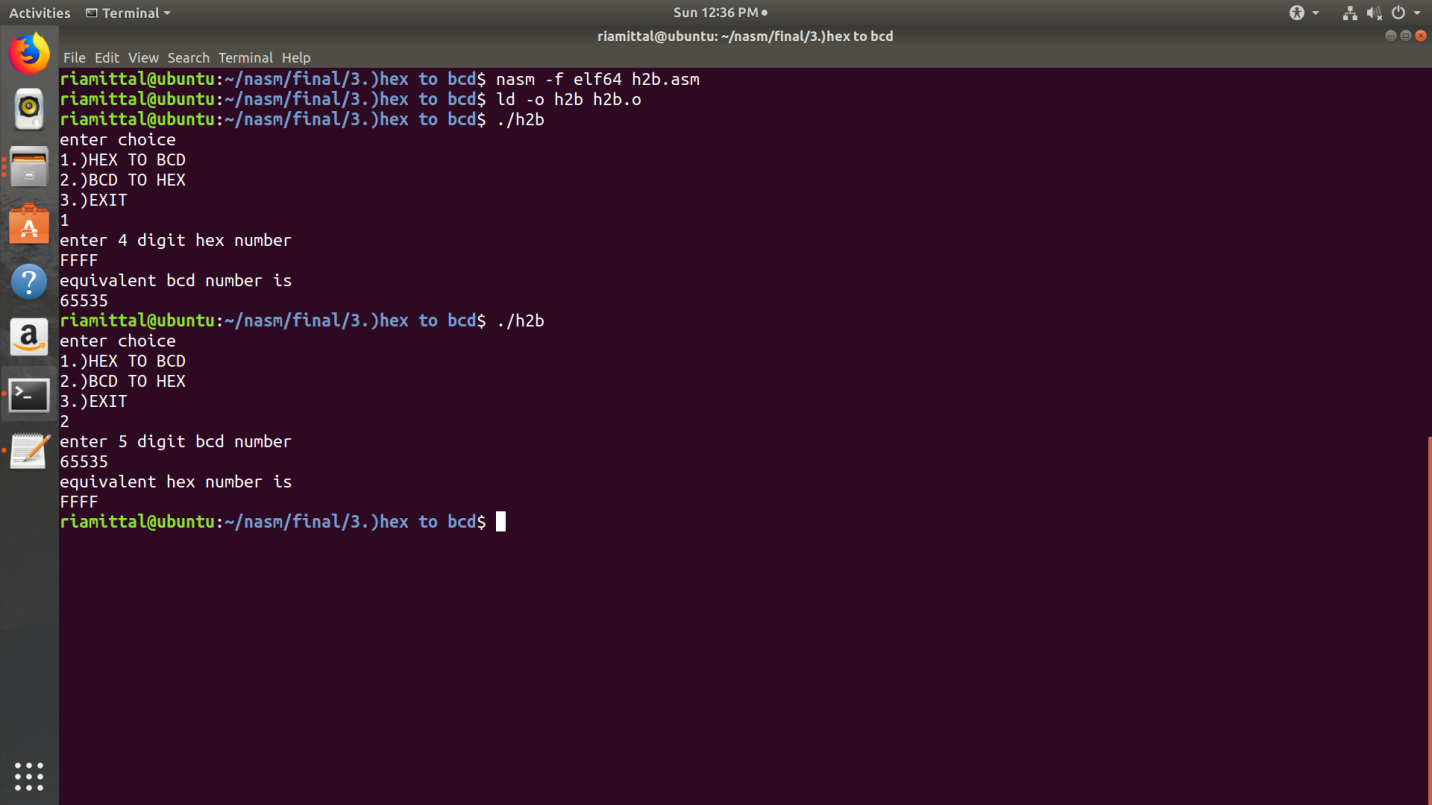
exit:

mov rax,60

mov rdi,0

syscall

;------------------------------------END----------------------------------

****