Practical – 6

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).

Program:

```
section .data

nline db 10,10

nline_len: equ $-nline

menu db 10,"-----------Menu----------"

db 10,"1. Hex to BCD "

db 10,"2. BCD to Hex"

db 10,"3. Exit "

db 10

db 10,"Enter your choice: "

menu_len: equ $-menu

h2bmsg db 10,"Hex to BCD "

db 10,"Enter 4-digit Hex number: "

h2bmsg_len: equ $-h2bmsg

b2hmsg db 10,"BCD to Hex "

db 10,"enter 5-digit BCD number: "

b2hmsg_len: equ $-b2hmsg

hmsg db 10,13,"Equivalent Hex number is: "

hmsg_len: equ $-hmsg

bmsg db 10,13,"Equivalent BCD number is: "
```

```
bmsg_len: equ $-bmsg
emsg db 10,"You entered Invalid Data!!!",10
emsg_len: equ $-emsg
section .bss
buf resb 6
buf_len: equ $-buf
digitcount resb 1
ans resw 1
char_ans resb 4
;macros as per 64-bit convensions
%macro print 2
mov rax,1 ; Function 1 - write
mov rdi,1 ; To stdout
mov rsi,%1 ; String address
mov rdx,%2 ; String size
syscall ; invoke operating system to WRITE
%endmacro
%macro read 2
mov rax,0 ; Function 0 - Read
mov rdi,0 ; from stdin
mov rsi,%1 ; buffer address
mov rdx,%2 ; buffer size
syscall ; invoke operating system to READ
%endmacro
%macro exit 0
print nline, nline_len
```

```
mov rax, 60 ; system call 60 is exit
xor rdi, rdi ; we want return code 0
syscall ; invoke operating system to exit
%endmacro
section .text
global _start
_start:
print menu, menu_len
read buf,2 ; choice + enter
mov al,[buf]
c1: cmp al,'1'
jne c2
call hex_bcd
jmp _start
c2: cmp al,'2'
jne c3
call bcd_hex
jmp _start
c3: cmp al,'3'
jne err
exit
err: print emsg,emsg_len
jmp _start
hex_bcd:
print h2bmsg, h2bmsg_len
call accept_16
```

```
mov ax,bx
mov rbx,10
back:
xor rdx,rdx
div rbx
push dx
inc byte[digitcount]
cmp rax,0h
jne back
print bmsg, bmsg_len
print_bcd:
pop dx
add dl,30h ; possible digits are 0-9 so add 30H only
mov [char_ans],dl ; store character in char_ans
print char_ans,1 ; print on screen in reverse order
dec byte[digitcount]
jnz print_bcd
ret
bcd_hex:
print b2hmsg, b2hmsg_len
read buf,buf_len ; buflen = 5 + 1
mov rsi,buf ; load bcd pointer
xor rax,rax ; sum
mov rbx,10
mov rcx,05 ; digit_count
back1: xor rdx,rdx
```

```
mul ebx ; previous digit * 10 = ans (rax*rbx = rdx:rax)
xor rdx,rdx
mov dl,[rsi] ; Take current digit
sub dl,30h ; accepted digit is Decimal, so Sub 30H only
add rax,rdx
inc rsi
dec rcx
jnz back1
mov [ans],ax
print bmsg, bmsg_len
mov ax,[ans]
call display_16
ret
;---------------------------------------------------------------
accept_16:
read buf,5 ; buflen = 4 + 1
xor bx,bx
mov rcx,4
mov rsi,buf
next_digit:
shl bx,04
mov al,[rsi]
cmp al,"0" ; "0" = 30h or 48d
jb error ; jump if below "0" to error
cmp al,"9"
jbe sub30 ; subtract 30h if no is in the range "0"-"9"
```

```
cmp al,"A" ; "A" = 41h or 65d
jb error ; jump if below "A" to error
cmp al,"F"
jbe sub37 ; subtract 37h if no is in the range "A"-"F"
cmp al,"a" ; "a" = 61h or 97d
jb error ; jump if below "a" to error
cmp al,"f"
jbe sub57 ; subtract 57h if no is in the range "a"-"f"
error: print emsg,emsg_len ; "You entered Invalid Data!!!"
exit
sub57: sub al,20h ; subtract 57h if no is in the range "a"-"f"
sub37: sub al,07h ; subtract 37h if no is in the range "a"-"f"
sub30: sub al,30h ; subtract 30h if no is in the range "0"-"9"
add bx,ax ; prepare number
inc rsi ; point to next digit
loop next_digit
ret
;---------------------------------------------------------------
display_16:
mov rsi,char_ans+3 ; load last byte address of char_ans in rsi
mov rcx,4 ; number of digits
cnt: mov rdx,0 ; make rdx=0 (as in div instruction rdx:rax/rbx)
mov rbx,16 ; divisor=16 for hex
div rbx
cmp dl, 09h ; check for remainder in RDX
jbe add30
```

```
add dl, 07h
add30:
add dl,30h ; calculate ASCII code
mov [rsi],dl ; store it in buffer
dec rsi ; point to one byte back
dec rcx ; decrement count
jnz cnt ; if not zero repeat
print char_ans,4 ; display result on screen
ret
;---------------------------------------------------------------
```

Output:

```
atharva@atharva:~$ gedit lab6.asm
atharva@atharva:~$ nasm -f elf64 lab6.asm
atharva@atharva:~$ ld -o lab6 lab6.o
atharva@atharva:~$ ./lab6
----------Menu----------
1. Hex to BCD
2. BCD to Hex
3. Exit
Enter your choice: 1
Hex to BCD
Enter 4-digit Hex number: 8A9F
Equivalent BCD number is: 35487
----------Menu----------
1. Hex to BCD
2. BCD to Hex
```

3. Exit

Enter your choice: 2

BCD to Hex

enter 5-digit BCD number: 35487

Equivalent BCD number is: 8A9F

----------Menu----------

1. Hex to BCD

2. BCD to Hex

3. Exit

Enter your choice: 3