;Write 16/64 bit 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 the 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. (use of 64-bit registers is expected).

```
;***************define macro **************
%macro input 4
	mov rax,%1
	mov rdi,%2
	mov rsi,%3
	mov rdx,%4
	syscall
%endmacro

;***************.data section **************
section .data
	menu db 10d,13d,"***MENU***"
	    db 10d,"1. Hex to BCD"
	    db 10d,"2. BCD to Hex"
	    db 10d,"3. Exit"
	    db 10d,"Enter your choice: "
	menulen equ $-menu

	m1 db 10d,13d,"Enter 4-digit Hex number: "
	l1 equ $-m1

	m2 db 10d,13d,"Enter 5-digit BCD number: "
	l2 equ $-m2
```

```
        m3 db 10d,13d,"Equivalent 5-digit BCD number: "
        l3 equ $-m3

        m4 db 10d,13d,"Equivalent 4-digit Hex number: "
        l4 equ $-m4


section .bss
        choice resb 1
        num resb 5
        answer resb 16
      ;  factor resb 4

section .code
        global _start
_start:

        input 1,1,menu,menulen
        input 0,0,choice,2

        cmp byte[choice],'3'
        jae exit
        cmp byte[choice],'1'
        je hex2bcd
        cmp byte[choice],'2'
        je bcd2hex

;**********Hex to BCD Conversion*****************
hex2bcd:
        input 1,1,m1,l1
        input 0,0,num,5
        call asciihextohex
```

```
       mov rax,rbx
        mov rbx,10
        mov rdi,num+15
loop3:
        mov rdx,0
        div rbx
        add dl,30h
        mov [rdi],dl
        dec rdi
        cmp rax,0
        jne loop3

      input 1,1,m3,l3
      input 1,1,num,16
jmp _start

;**********BCD to Hex Conversion************************
bcd2hex:
      input 1,1,m2,l2
      input 0,0,num,6

       mov rbp,5
       mov rsi,num
        mov rbx,10

nxt4: xor rcx,rcx
        mul rbx
        mov cx,[rsi]
       sub cx,30h
        add rax,rcx

        inc rsi
```

```
dec rbp
          jnz nxt4
       mov [ answer],rax

          input 1,1,m4,l4
          mov rax,[ answer]
          call display
         jmp _start

exit:
       mov rax,60
       mov rdx,0
       syscall


;*******************PROCEDURES**********************

asciihextohex:
          mov rsi,num
          mov rcx,4
          xor rbx,rbx
          mov rax,0

loop1: rol rbx,04
          mov al,[rsi]
          cmp al,39h
          jbe skip1
          sub al,07h
skip1:sub al,30h
          add rbx,rax
          inc rsi
          dec rcx
          jnz loop1
```

```
    ret

display:
        mov rsi,answer+3
        mov rcx,4

loop2:mov rdx,0
        mov rbx,16
        div rbx
        cmp dl,09h
        jbe skip2

        add dl,07h
skip2: add dl,30h
        mov [rsi],dl

        dec rsi
        dec rcx
        jnz loop2
        input 1,1,answer,16
ret
```

****

;Output Of 64 Bit NASM Code:

```
$ nasm -f elf64 mil3.asm
$ ld -o a mil3.o
$ ./a

MENU
1. Hex to BCD
2. BCD to Hex
3. Exit
```

```
Enter your choice: 1

Enter 4-digit Hex number: FFFF

Equivalent 5-digit BCD number: 65535
MENU
1. Hex to BCD
2. BCD to Hex
3. Exit
Enter your choice: 2

Enter BCD Number:65535

Equivalent Hex Number:FFFF
MENU
1. Hex to BCD
2. BCD to Hex
3. Exit
Enter your choice: 3
$
```