**COA ASSIGNMENT NO. :- 5**

**Aim:-**To write a 64-bit Assembly Level Program to convert HEX 4-digit input to BCD 5-digit output.

**Objective:-** To input a 4-digit HEX number from user, convert it into a 5-digit BCD number and display it on screen.

**Theory:-** The Hexadecimal or simply Hex numbering system uses the Base of 16 system and are a popular choice for representing long binary values because their format is quite compact and much easier to understand compared to the long binary strings of 1’s and 0’s.

Binary-coded decimal (BCD) is a class of binary encodings of decimal numbers where each decimal digit is represented by a fixed number of bits, usually four or eight. Special bit patterns are sometimes used for a sign or for other indications.

Divide **FFFF** by **10** --- this FFFF is as decimal **65535** so Division

65535 / 10 Quotient = 6553 Reminder = 5

6553 / 10 Quotient = 655 Reminder = 3

655 / 10 Quotient = 65 Reminder = 5

65 / 10 Quotient = 6 Reminder = 5

6 / 10 Quotient = 0 Reminder = 6

and then we are pushing Reminder on stack and then printing it in reverse order.

**Source Code:-**

%macro scall 4

mov rax,%1

mov rdi,%2

mov rsi,%3

mov rdx,%4

syscall

%endmacro

section .data

m1 DB &quot;Enter the four digit hex input&quot;,10d,13d

l1 equ $-m1

m2 DB &quot;Equivalent bcd number is&quot;,10d,13d

l2 equ $-m2

section .bss

buf resb 6

digitcount resb 1

char\_ans resb 4

num resb 6

section .text

global \_start:

\_start:

scall 1,1,m1,l1

call accept\_proc

mov ax,bx

mov rbx,0Ah

back:

xor rdx,rdx

div rbx

push dx

inc byte[digitcount]

cmp rax,0H

jne back

print\_bcd:

pop dx

add dl,30H

mov [char\_ans],dl

scall 1,1,char\_ans,1

dec byte[digitcount]

jnz print\_bcd

mov rax,60

mov rdi,0

syscall

accept\_proc:

scall 0,0,buf,5

xor bx,bx

xor ax,ax

mov rcx,4

mov rsi,buf

next\_digit:

rol bx,04h

mov al,[rsi]

cmp al,39h

jbe L1

sub al,07h

L1: sub al,30h

add bx,ax

inc rsi

dec rcx

jnz next\_digit

;loop next\_digit

ret

**Output:-**

Enter the four digit hex input

FFFF

65535