# Name: Shrinivas Hatyalikar

# DIv: CS-B

# Roll No: 24

**Aim:** Write 64 bit ALP to convert 5-digit BCD number into its equivalent HEX number. Make your program user friendly to accept the choice from user for:

Display proper strings to prompt the user while accepting the input and displaying the result. (Use of 64-bit registers is expected)

**Apparatus:**

* Core 2 duo/i3/i5/i7 - 64bit processor
* OS – ubuntu 32bit/64bit OS
* Assembler used –nasm (the netwide assembler)
* Editor Used – gedit

**Theory :**

**STACK Operations:**

The stack follows the LIFO (Last In First Out) principle. The stack can be used, for example, to pass parameters to functions. T here are three instructions that can be used for interaction with the stack: Pop, Push and Exch.

**PUSH:-**

The PUSH instruction increments the stack pointer and stores the value of the specified byte operand at the internal RAM address indirectly referenced by the stack pointer. No flags are affected by this instruction.

**Operation**

**PUSH**

**SP = SP + 1**

**SP) = (direct)**

**POP:-**

The POP instruction reads a byte from the address indirectly referenced by the SP register. The value read is stored at the specified address and the stack pointer is decremented. No flags are affected by this instruction.

**Operation**

**POP**

**(direct) = (SP)**

**SP = SP - 1**

**ALGORITHM:- BCD to Hexadecimal conversion**

1. Declare

Section .data of proper string messages.

Section .bss of proper variables.

Declare various macros e.g. print, read & exit macros.

Section .text as starting point of code segment.

2. Prompt messages on screen & accept choice from user.

3. According to choice, accept 5 digit BCD no. from user using read macro. The number is

accepted in character format.

4. Point at the start of number by RSI.

5. Clear contents of RAX. Final answer will be stored in RAX.

6. Load RBX as 10 (BCD number i.e. base 10, so multiply with 10)

7. Load counter CX as 5. (5 digit no.)

8. Clear contents of RDX (for multiplication it is used. So clear previous answer)

9. Multiply RAX by 10.

i.e. RAX \* RBX = RDX:RAX

previous digit \* 10 = ans

10. Clear contents of RDX (for multiplication it is used. So clear previous answer)

11. Load current digit in DL from RSI.

12. Convert the contents of DL from character to digit by subtracting 30h.

13. Add the current digit in final answer.

14. Increment source pointer.

15. Repeat steps 8 to 14 till CX becomes zero.

16. Store answer from AX to ans variable.

17. Print BCD to HEX ConversiIon message on screen.

18. Store answer from ans variable to AX again.

19. Call display\_16 procedure to display final 16-bit hex no. on screen.

20. Stop

**Program:**

section .data

nline db 10,10

nline\_len equ $-nline

ano db 10," Assignment no :5",

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

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

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

ano\_len equ $-ano

hmsg db 10,"BCD to 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

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

call BCD\_HEX

Exit

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

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

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,20H

sub37: SUB AL,07H

sub30: SUB AL,30H

ADD BX,AX

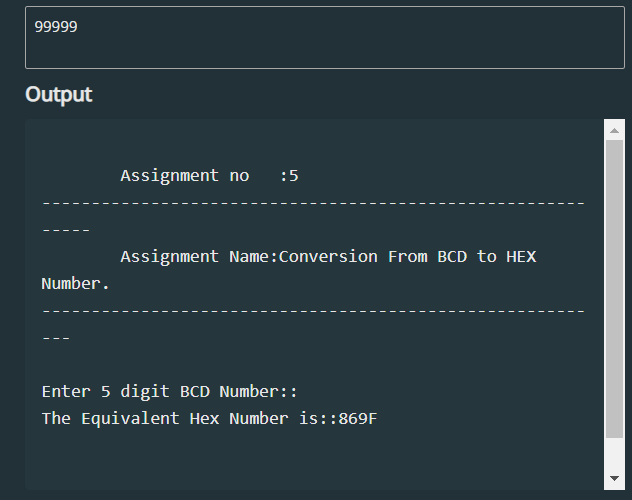
INC RSI

DEC RCX

JNZ next\_byte

RET

**Output:**

****

**Conclusion:**

To summarize, the implementation of an assembly program for converting decimal to hexadecimal was successful. The program accepts a decimal number as input, and using a loop, converts each digit to its hexadecimal equivalent and saves it in memory. The final output is displayed as a hexadecimal number. This program showcases the power and versatility of assembly language, which allows for efficient conversion of numbers between different systems. It also emphasizes the significance of understanding various number systems and their conversions, particularly when working with low-level programming. Overall, this program is a valuable tool for programmers and computer scientists who work with decimal and hexadecimal numbers. Additionally, it can be modified to support other number systems like binary or octal. With its clear demonstration of converting between decimal and hexadecimal, this program can improve understanding of number systems and assembly programming for interested individuals.