Title: Write 64-bit ALP to convert HEX 4-digit input to BCD 5-digit output.

Aim :

Write 64 bit ALP to accept number and display it on screen.

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

Software/Hardware Required:

Core 2 duo/i3/i5/i7 - 64bit processor

Operating System – ubuntu/Fedora 64bit OS

Assembler: NASM

Editor Used – gedit

Theory:

Explanation:

1. Hexadecimal to BCD conversion:

Conversion of a hexadecimal number can be carried out by dividing number by 10 or 0Ah and

displaying quotient in reverse way.

Example: Number in Hexadecimal form: 7BH

÷

7BH Remainder

0AH 3

-------

Quotient 0CH

÷

0AH

2

--------

Quotient 01H

÷

0AH

1

-------

Quotient 00H

Result: 123 is the equivalent BCD2. BCD to Hexadecimal number:

Conversion of BCD number to Hexadecimal number can be carried out by multiplying the

previous result by 10 0r 0AH and adding new digit to it.

e.g. 123

1 x 0AH = 0AH

+ 2

------------

0CH

X 0AH

------------

78H

+

3

-----------

7BH is the hex equivalent of 123 BCD

Algorithm for HEX to BCD conversion procedure

1. Start
2. Display ‘Input 4 digit hex number’ message using Display macro
3. Accept 4 digit HEX number from user using accept macro and store it
4. in num variable.
5. Call Ascii\_to\_Hex procedure to convert accepted ascii value of num
6. digit into hexadecimal number.
7. Load result of step iv in RAX
8. Initialise RCX=0005 i.e. number of times to divide the number by 0Ah
9. Load RDX=0000
10. Load RBX=000Ah,
11. Divide the number using DIV RBX instruction, which produces
12. Quotient in RAX and remainder in RDX
13. Push DX on stack
14. Decrement RCX, If not zero jump to step vii, else continue
15. Display the result message using Display macro
16. Load RCX=0005, number of digits to display
17. Pop the last pushed remainder in DX for display
18. Add 30H in DL to produce ASCII code of the digit, and display digit
19. display macro.
20. Decrement RCX, If not zero jump to step xiv
21. Return

Ascii\_to\_Hex

1.Purpose : Convert 4 Ascii character into 4 digit hex number

2. I/P : Number accepted from user

3.Algorithm for Procedures

a. Start

b. ESI point to the ascii\_num

c. Initialize ecx by 4

d. Initialize bx with 0

e. Rotate left bx by 4 bits

f. Mov value pointed by ESI in al

g. Compare al with 39H

i. If al is below or equal to 39H then sub 30H in al

ii. Else sub 37H in al

h. Add bl with al

i. Move value of al in bl

j. Repeat step e-i until ecx becomes 0

k. Return

Hex\_to\_Ascii

1.Purpose : Convert 4 digit hex number into 4 Ascii character to display it on Standard

output

2.I/P : 4 digit HEX number

3. O/P: 4 Ascii Character

4. Algorithm for Proceduresa.

1. Start
2. ESI point to the temp
3. Initialize ecx by 4
4. Rotate left bx by 4 bits
5. Mov bl in al
6. Compare al with 39H
7. i. If al is below or equal to 39H then add 30H in al
8. ii. Else add 37H in al
9. Move value of al in temp pointed by ESI
10. Increment ESI
11. Repeat step d-i until ecx becomes 0
12. Call Display macro to print result.
13. Return

Program:

section .data

menumsg db 10,"\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*"

db 10,"1.HEX to BCD"

db 10,"2.EXIT"

db 10,"Enter U r Choice:"

menul equ $-menumsg

h2b db 10,"HEX to BCD"

db 10,"Enter 4 digit Hex Number:"

h2bl equ $-h2b

b1msg db 10,13,"Equivalent BCD number is:"

b1msgl equ $-b1msg

section .bss

choice resb 2

buf resb 6

bufl equ $-buf

digitcount resb 1

ans resw 1

char\_ans resb 4

fact resw 1

%macro scall 4

mov rax,%1

mov rdi,%2

mov rsi,%3

mov rdx,%4

syscall

%endmacro

%macro exit 0

mov rax,60

xor rdi,rdi

syscall

%endm

section .text

global \_start

\_start:

scall 01,01,h2b,h2bl

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

scall 01,01,b1msg,b1msgl

print\_bcd:

pop dx

add dl,30h

mov [char\_ans],dl

scall 01,01,char\_ans,1

dec byte[digitcount]

jnz print\_bcd

exit

accept\_16:

scall 0,0,buf,5

xor bx,bx

mov rcx,4

mov rsi,buf

next\_digit:

shl bx,04

mov al,[rsi]

cmp al,39h

jbe l1

sub al,07h

l1: sub al,30h

add bx,ax

inc rsi

loop next\_digit

ret

display\_16:

scall 01,01, b1msg,b1msgl

mov rcx,4

mov rsi,char\_ans

back2:

rol bx,04

mov dl,bl

and dl,0fh

cmp dl,09h

jbe l2

add dl,07

l2:add dl,30h

mov [rsi],dl

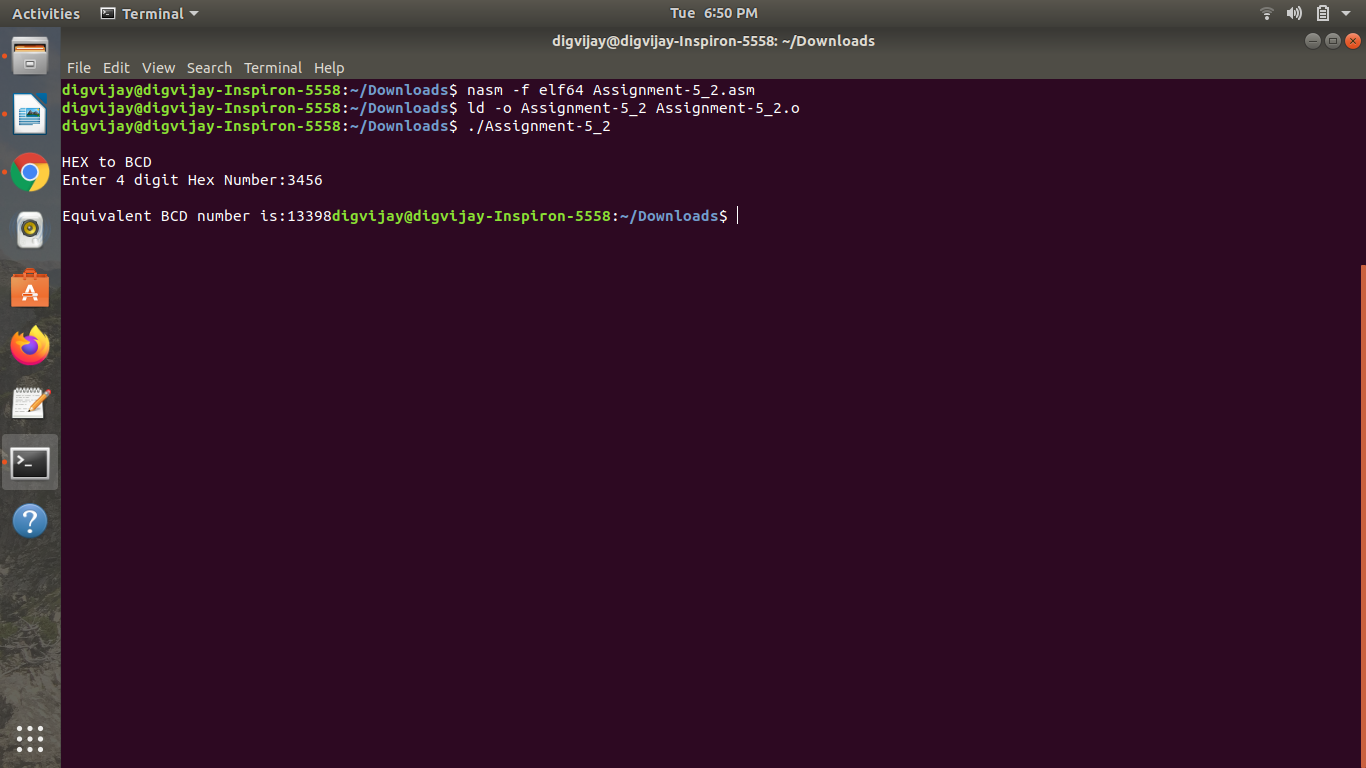
inc rsi

loop back2

scall 01,01,char\_ans,4

ret

Output:



Conclusions:

Assembly Level Program for HEX to BCD and BCD to HEX conversion is assembled and

executed successfully.