Page No.
Date

Name: Manthan Jonawane
Panel: E Batch: E1 Roll No: 17

MAIOT LAB Assignment - 8

Problem Statement: Perform BCD to HEX and HEX to BCD using X86/64 ALP.

Theory:

1) Write algorithm to convert BCD to HEX number.

→ 1. Initialize product to 0.
2. Accept the BCD number.
3. Extract the first digit.
4. Convert it to HEX
5. Multiply the product by 0Ah.
6. Add the extracted digit to it
7. Increment pointer
8. Repeat step 3 to 5 until you read "0a".
9. Display the product

2) Write algorithm to convert HEX to BCD number.

→ 1. Accept the HEX no.
2. Pack and save it to ax
3. Initialize dx = 0
4. Divide by 0Ah
5. Push remainder on stack
6. compare quotient with zero
7. If not, go to step 3, else go to 8.
8. Pop digits from stack and display.

Page No.
Date

3) Explain MUL and DIV instructions as well PUSH & POP

→ MUL :- There are 2 instructions for multiplying binary data. MUL (multiply) instruction handles unsigned data and the IMUL (Integer multiply) handles signed data. Both instructions affect carry and overflow flag.

DIV - DIV operator is used to carry out division where a quotient and remainder is generated. In division, when overflow occurs, the processor generates an interrupt. There are DIV and IDIV instructions.

PUSH - It stores a constant or 64-bit register out onto the stack. eg. of register is "rax" or "r8".

POP - It retrieves the last values pushed from the stack eg. POP ebx

Conclusion : Hence we have learned to perform BCD to HEX and HEX to BCD conversion in ALP

FAQ's

1) What are packed and unpacked numbers?

→ - Packed numbers are numbers in which 2 BCD digits are stored in a single ~~eg~~ 8-bit register.
eg - 98 - 10011000

- Unpacked numbers are in which each 4 digit BCD group corresponding to a decimal digit is stored in separate register
eg. 98 ⇒ 09 [00001001] and 08 [00001000]

Page No.
Date

2) What is the necessity to convert unpacked to packed

→ - while accepting an array of numbers from the user, all numbers are stored in unpacked form and need to be packed for further arithmetic operations.
- Packing is required for displaying a number taken from user or displaying addition result of the initialized 2-digit hex numbers.

3) What are assembler directives? ~~Sol~~ Give examples.

→ Assembler directives - They supply data to the program and control assembly process. They are effective only during the assembly of a program but they do not generate any code that is executable

eg: DB - define byte - used to define 8 bit data
EQU - used to define a constant without storing information in the memory
SET - it allows redifination of a symbol at a later stage.

BCD TO HEX

```
%macro rw 4
mov rax,%1
mov rdi,%2
mov rsi,%3
mov rdx,%4
syscall
%endmacro


section .data
msg1 db "Enter 5 digit bcd no:",10
msg1len equ $-msg1
msg2 db "HEX equivalnet is:",10
msg2len equ $-msg2
product dw 0
msg3 db 10d
msg3len equ $-msg3


section .bss
Result resw 1
temp resb 1
num resb 6


section .text
global _start:
_start:
rw 1,1,msg1,msg1len
rw 0,0,num,6
rw 1,1,msg2,msg2len
mov ax,0
mov bp,0ah
mov rsi,num
up: mov bx,0
mov bl,byte[rsi]
cmp bl,0Ah
jz display
```

```
sub bl,30h
mul bp
add ax,bx
inc rsi
jmp up



display:
mov bp,4
up1: rol ax,4
mov bx,ax
and ax,0Fh
cmp al,09
jbe down
add al,07h
down:Add al,30h
mov byte[temp],al
rw 1,1,temp,1
mov ax,bx
dec bp
jnz up1
rw 1,1,msg3,msg3len
rw 60,0,0,0
```

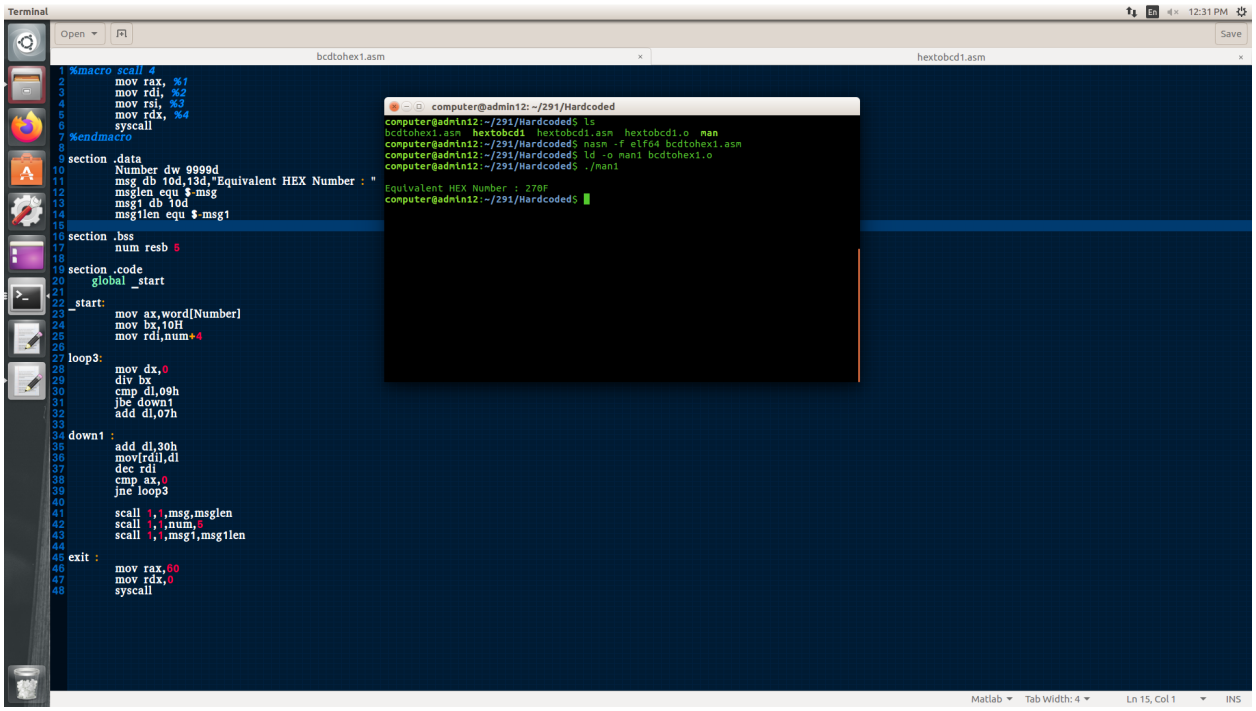
HEX TO BCD

```
section .data
                msg1 db "Enter HEX number:",10
                msg1len equ $-msg1
                msg2 db "BCD equivalent number:",10
                msg2len equ $-msg2

section .bss
                a resb 1
                num resb 5

%macro operat 4
                mov rax,%1
                mov rdi,%2
                mov rsi,%3
                mov rdx,%4
```

```
                syscall
%endmacro

section .text
        global _start
_start:
        operat 1,1,msg1,msg1len
        operat 0,0,num,5
        mov rsi,num
        mov rbp,00

        mov ax,00h
again:
        mov bl,byte[rsi]
        cmp bl,0ah
        je htob
        cmp bl,39h

        jbe sub30h
        sub bl,07h
        sub30h:
        sub bl,30h
        rol ax,4
        add al,bl
        inc rsi
        jmp again
htob:
        mov dx,00
        mov bx,0Ah
        div bx
        push dx
        inc rbp
        cmp eax,00
        jnz htob
        operat 1,1,msg2,msg2len

prnt:
         pop dx
nxt1:
         add dx,30h
        mov [a],dl
        operat 1,1,a,1
        dec rbp
        jnz prnt
```

```
        mov rax,60
        mov rdx,0
        syscall
```

OUTPUT



Terminal

bcdtohex1.asm | hextobcd1.asm

```
%macro rw 4
        mov rax,%1
        mov rdi,%2
        mov rsi,%3
        mov rdx,%4
        syscall
        %endmacro

section .data
        Number dw 0FFFFH
        msg db 10d,13d,"Equivalent BCD Number:"
        msglen equ $-msg
        msg1 db 10d
        msg1len equ $-msg1

section .bss
        num resb 5

section .text
    global _start:

_start:

        mov ax,word[Number]
        mov bx,0AH
        mov rdi,num+4
loop3:
        mov dx,0
        div bx
        add dl,30h
        mov [rdi],dl
        dec rdi
        cmp ax,0
        jne loop3

        rw 1,1,msg,msglen
        rw 1,1,num,5
        rw 1,1,msg1,msg1len
        rw 60,0,0,0
```

computer@admin12: ~/291/Hardcoded

```
computer@admin12:~/291/Hardcoded$ nasm -f elf64 hextobcd1.asm
computer@admin12:~/291/Hardcoded$ ld -o man hextobcd1.o
computer@admin12:~/291/Hardcoded$ ./man

Equivalent BCD Number:65535
computer@admin12:~/291/Hardcoded$
```

Matlab | Tab Width: 4 | Ln 10, Col 22 | INS