ROLL NO-2

MAIOT ASSIGNMENT 8

AARON PHILIP
1032210163

AIM:
Write x86/64 ALP to perform BCD to HEX and HEX to BCD conversion.

INPUT: unsigned hex / BCD number
OUTPUT: equivalent BCD or Hex number

PLATFORM:
Linux

THEORY:

Write an algorithm to convert BCD number to HEX

make a macro scall for input & output
in section .data
declare the BCD number & messages
In section .bss reserve 5 bytes to store hex
_start:
mov ax, word [Number]
mov bx, 10h
mov rdi, num+4
loop3:
mov dx, 0 ; clear contents of d-register
div bx
cmp dl, 09h
jbe down1
add dl, 07h

```
down1:
add dl, 30h
mov [rdi], dl
dec rdi
cmp ax, 0
jne loop3
```

→ print the equivalent Hex number

4) Write an algorithm to convert HEX number to BCD number

→ make a macro scall for input & output

→ in section .data
declare the hex number & messages

→ In section .bss reserve 5 bytes to store BCD

```
_start:
mov ax, word [Number]
mov bx, 0Ah
mov rdi, num + 4
loop3:
mov dx, 0 ; clear contents of d-register
dw bx
cmp al, 09h
jbe down1
add dl, 07h
down1:
add dl, 30h
mov [rdi], dl
dec rdi
cmp ax, 0 ; check if 0 so we can terminate
jne loop 3
```

→ display & exit syscall

Q3) Explain MUL and DIV instructions as well as PUSH & POP instructions.

A3) "MUL" and "DIV" are x86 Assembly Language Programming (ALP) instructions for performing multiplication and division operations, respectively. They take operands from the registers or memory locations and return the result to the specified order.

- "PUSH" and "POP" are x86 ALP instructions used for stack manipulation. "PUSH" pushes a value onto the stack, while "POP" retrieves the top value from the stack & stores it in a register or memory location.

CONCLUSION

Through the above implementation, we learnt conversion of BCD to Hex and Hex to BCD using x86/64 ALP instructions.

FAQs

Q1) What are packed & unpacked numbers ?

A1) Packed & unpacked numbers in byte-oriented systems are representations of numbers. The term packed BCD implies a full byte for each digit (often including a sign) whereas packed BCD typically encodes two digits within a single byte by taking advantage of the fact that 4 bits are enough to represent the range 0 to 9.

Q2) What is the necessity to convert from unpacked to packed?

A2) Packing of hexadecimal numbers is required in computing to save memory space. Computers store & process data in binary form, which can be efficiently represented using hexadecimal numbers.

Q3) What are assembler directives? Give examples.

A3) Assembler directives are instructions for the assembler, not the machine. They are used to provide information about the program and control the assembly process.

1) ① 'DB' - Define Byte - Defines a constant value of one or more bytes

② 'EQU' - Equate - Defines a symbolic constant.

CODE :

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

section .data
Number dw 9999d
msg db 10d, "Equivalent Hex number is:" ; 10d is newline and 13d is left-align
msglen equ $-msg

section .bss
num resb 5 ; reserved 5 bytes to store hex (need only 4 but use 1 byte as buffer)

section .text
global _start
_start:
mov ax,word[Number]
mov bx,10h
mov rdi,num+4

loop3:
mov dx,0 ; clear contents of d-register
div bx
cmp dl,09h
jbe down1
add dl,07h

down1:
add dl,30h
mov [rdi],dl
dec rdi
cmp ax,0 ; check if it is 0 so we can terminate
jne loop3

scall 1,1,msg,msglen
scall 1,1,num,5

exit:
mov rax,60
```

mov rdx,0
syscall

OUTPUT :