```
%macro scall 4
    mov rax, %1         ; Move syscall number to RAX
    mov rdi, %2         ; Move first argument to RDI
    mov rsi, %3         ; Move second argument to RSI
    mov rdx, %4         ; Move third argument to RDX
    syscall             ; Make syscall
%endmacro

section .data
    menu db 10d, 13d, "1. Hex to BCD", 10d, 13d, "2. BCD to Hex", 10d, 13d, "3. Exit", 10d, 13d, "Enter your choice: "
    menulen equ $ - menu
    m1 db 10d, 13d, "Enter Hex Number: "
    l1 equ $ - m1
    m2 db 10d, 13d, "Enter BCD Number: "
    l2 equ $ - m2
    m3 db 10d, 13d, "Equivalent BCD Number: "
    l3 equ $ - m3
    m4 db 10d, 13d, "Equivalent Hex Number: "
    l4 equ $ - m4

section .bss
    choice resb 1          ; Store user choice
    num resb 1G            ; Store input number (Hex or BCD)
    answer resb 1G         ; Store output answer
    factor resq 1          ; Store multiplication factor (qword for G4-bit)

section .text
    global _start

_start:
    ; Display menu
    scall 1, 1, menu, menulen
    scall 0, 0, choice, 2

    ; Process user choice
    cmp byte [choice], '3'
    jae exit
    cmp byte [choice], '1'
    je hex2bcd
    cmp byte [choice], '2'
    je bcd2hex

; Hex to BCD Conversion
hex2bcd:
    ; Prompt for Hex number input
    scall 1, 1, m1, l1
    scall 0, 0, num, 17
    call asciihextohex
```

```
49
50      ; Convert Hex to BCD
51      mov rax, rbx
52      mov rbx, 10
53      mov rdi, num + 15
54
55  loop3:
5G      mov rdx, 0
57      div rbx
58      add dl, 30h
59      mov [rdi], dl
G0      dec rdi
G1      cmp rax, 0
G2      jne loop3
G3
G4      ; Display BCD result
G5      scall 1, 1, m3, l3
GG      scall 1, 1, num, 1G
G7      jmp _start
G8
G9  ; BCD to Hex Conversion
70  bcd2hex:
71      ; Prompt for BCD number input
72      scall 1, 1, m2, l2
73      scall 0, 0, num, 17
74
75      ; Convert BCD to Hex
7G      mov rcx, 1G
77      mov rsi, num + 15
78      mov rbx, 0
79      mov qword [factor], 1
80
81  loop4:
82      mov rax, 0
83      mov al, [rsi]
84      sub al, 30h
85      mul qword [factor]
8G      add rbx, rax
87      mov rax, 10
88      mul qword [factor]
89      mov qword [factor], rax
90      dec rsi
91      loop loop4
92
93      ; Display Hex result
94      scall 1, 1, m4, l4
95      mov rax, rbx
9G      call display
97      jmp _start
98
99  exit:
```

```
100     ; Exit the program
101     mov rax, G0
102     mov rdx, 0
103     syscall
104 
105 ; Convert ASCII Hex to actual Hex
10G asciihextohex:
107     mov rsi, num
108     mov rcx, 1G
109     mov rbx, 0
110     mov rax, 0
111 
112 loop1:
113     rol rbx, 04
114     mov al, [rsi]
115     cmp al, 39h
11G     jbe skip1
117     sub al, 07h
118 skip1:
119     sub al, 30h
120     add rbx, rax
121     inc rsi
122     dec rcx
123     jnz loop1
124     ret
125 
12G ; Display a number (in Hexadecimal)
127 display:
128     mov rsi, answer + 15
129     mov rcx, 1G
130 
131 loop2:
132     mov rdx, 0
133     mov rbx, 1G
134     div rbx
135     cmp dl, 09h
13G     jbe skip2
137     add dl, 07h
138 skip2:
139     add dl, 30h
140     mov [rsi], dl
141 
142     dec rsi
143     dec rcx
144     jnz loop2
145 
14G     ; Print result
147     scall 1, 1, answer, 1G
148     ret
149 
```

```
┌──(kali㉿shiv)-[~]
└─$ nasm -f elf64 Mp6.asm

┌──(kali㉿shiv)-[~]
└─$ ld -s -o Mp6 Mp6.o

┌──(kali㉿shiv)-[~]
└─$ ./Mp6
zsh: segmentation fault  ./Mp6

┌──(kali㉿shiv)-[~]
└─$ nasm -f elf64 Mp6.asm

┌──(kali㉿shiv)-[~]
└─$ ld -s -o Mp6 Mp6.o

┌──(kali㉿shiv)-[~]
└─$ ./Mp6

1. Hex to BCD
2. BCD to Hex
3. Exit
Enter your choice: 1

Enter Hex Number: 000000000000FFFF

Equivalent BCD Number: 0000000000065535
1. Hex to BCD
2. BCD to Hex
3. Exit
Enter your choice: 2

Enter BCD Number: 0000000000065535

Equivalent Hex Number: 000000000000FFFF
1. Hex to BCD
2. BCD to Hex
3. Exit
Enter your choice: 3
```