Assignment No.

Program:

```
;Assignment number
Aim : Write 64 bit ALP to convert 4-digit Hex number into its
equivalent BCD number and 5-digit BCD number into its equivalent HEX
number. Make
;Student Name - Aviraj Popat Kale
;PRN number : 122B1B123
;Perform Date :
;----------------------------------------------------------------------
--
;-----------------------------------------Section Data-----------------
--
section .data
        nline           db      10,10; Define newline characters
        nline_len       equ     $-nline          ; Calculate length of
newline characters

        ano             db      10,"    Assignment no    :3",  ;
Define assignment information
                            db      10,"-------------------------------
-----------------------------",
                            db      10,"    Assignment Name:Conversion
From HEX to BCD and BCD to HEX Number.",
                            db      10,"-------------------------------
----------------------------",10
        ano_len         equ     $-ano            ; Calculate length of
assignment information

        menu            db      10,"1.Hex To BCD.",   ; Define menu
options
                            db      10,"2.BCD To Hex.",
                            db      10,"3.Exit."
                            db      10,"Enter Your Choice::"
        menu_len        equ     $-menu          ; Calculate length of
menu options

        hmsg            db      10,"Enter 4 digit Hex Number::"  ;
Define messages for input prompts
        hmsg_len        equ     $-hmsg          ; Calculate length of
hex input message

        bmsg            db      10,"Enter 5 digit BCD Number::"  ;
Define messages for input prompts
        bmsg_len        equ     $-bmsg          ; Calculate length of
BCD input message
```

```
        ebmsg           db      10,"The Equivalent BCD Number is::"  ;
Define messages for output prompts
        ebmsg_len       equ     $-ebmsg         ; Calculate length of
equivalent BCD output message

        ehmsg           db      10,"The Equivalent Hex Number is::"  ;
Define messages for output prompts
        ehmsg_len       equ     $-ehmsg         ; Calculate length of
equivalent hex output message

        emsg            db      10,"INVALID NUMBER INPUT",10  ; Define
error message
        emsg_len        equ     $-emsg          ; Calculate length of
error message
;--------------------------------------------------------------------
---------
section .bss
        buf             resB    6           ; Define buffer for input
        char_ans        resB    4           ; Define buffer for
character answer
        ans             resW    1           ; Define buffer for
answer
;--------------------------------------------------------------------
--------
%macro  Print   2                    ; Define macro for printing
         MOV    RAX,1
         MOV    RDI,1
         MOV    RSI,%1
         MOV    RDX,%2
    syscall
%endmacro
%macro  Read    2                    ; Define macro for reading input
         MOV    RAX,0
         MOV    RDI,0
         MOV    RSI,%1
         MOV    RDX,%2
    syscall
%endmacro
%macro Exit 0                        ; Define macro for exiting
program
    Print   nline,nline_len
    MOV    RAX,60
        MOV    RDI,0
    syscall
%endmacro
;--------------------------------------------------------------

section .text
        global _start
_start:
        Print   ano,ano_len          ; Print assignment information
        jmp     MENU                    ; Jump directly to the menu
loop

MENU:
        Print   menu,menu_len          ; Print menu options
        Read    buf,2                   ; Accept choice i.e 1
digit+enter
        mov     al, [buf]               ; Contains only digit character

        cmp     al, '1'                 ; Compare input with option 1
        je      HEX_BCD                 ; If equal, jump to HEX_BCD
conversion
        cmp     al, '2'                 ; Compare input with option 2
        je      BCD_HEX                 ; If equal, jump to BCD_HEX
conversion
        cmp     al, '3'                 ; Compare input with option 3
        je      Exit                    ; If equal, exit program

        ; If input is invalid, print error message and repeat menu
        Print   emsg,emsg_len
        jmp     MENU

HEX_BCD:                                ; Convert HEX to BCD
        Print   hmsg,hmsg_len          ; Print input message for hex
number
        call    Accept_16              ; Accept 4 digit hex number
        mov     ax, bx                  ; Move hex number to ax

        mov     bx, 10                  ; Set bx for division by 10
        xor     bp, bp                  ; Initialize counter to 0

back:   xor     dx, dx                  ; Clear dx (remainder)
        div     bx                      ; Divide ax by 10; ax=Q, dx=R
        push    dx                      ; Push dx (remainder) onto
stack (BCD)
        inc     bp                      ; Increment counter

        cmp     ax, 0                   ; Check if Q is 0 (end of
number)
        jne     back                    ; If not 0, continue conversion

        Print   ebmsg,ebmsg_len        ; Print output message for
equivalent BCD

back1:  pop     dx                      ; Pop last digit from stack
        add     dl, 30h                 ; Convert digit to decimal
ASCII
        mov     [char_ans], dl          ; Store digit in char_ans for
printing
        Print   char_ans, 1             ; Print individual digit

        dec     bp                       ; Decrement counter
        jnz     back1                   ; Continue with next digit if
counter is not zero

        jmp     MENU                    ; Jump back to menu

BCD_HEX:                               ; Convert BCD to HEX
        Print   bmsg, bmsg_len          ; Print input message for BCD
number
        Read    buf, 6                  ; Read 5 digit BCD number

        mov     rsi, buf                ; Point at the start of buffer
        xor     ax, ax                  ; Initialize ax to 0 (previous
digit)
        mov     rbp, 5                  ; Set counter to 5
        mov     rbx, 10                 ; Set multiplier for addition

next:   xor     cx, cx                  ; Initialize cx (next digit)
        mul     bx                      ; Multiply ax by 10 and add cx
        mov     cl, [rsi]               ; Load next character from
input
        sub     cl, 30h                 ; Convert ASCII to number
        add     ax, cx                  ; Add to ax

        inc     rsi                     ; Point to next digit
        dec     rbp                     ; Decrement counter
        jnz     next                    ; Repeat for next digit if
counter is not zero

        mov     [ans], ax               ; Store ax in ans for printing
        Print   ehmsg, ehmsg_len        ; Print output message for
equivalent hex
        mov     ax, [ans]
        call    Disp_16                 ; Print hex number

        jmp     MENU                    ; Jump back to menu

; Convert hex to ASCII and display
Disp_16:
        MOV     RSI, char_ans+3         ; Point to last character in
char_ans
        MOV     RCX, 4                   ; Set counter to 4 (4 digits
in hex)
        MOV     RBX, 16                 ; Set divisor to 16
(hexadecimal)

next_digit:
        XOR     RDX, RDX                ; Clear remainder
        DIV     RBX                     ; Divide by 16
        CMP     DL, 9                   ; Check if remainder is less
than or equal to 9
        JBE     add30                   ; If so, add 30h to convert to
ASCII
```

```
        ADD     DL, 07H                 ; Otherwise, add 07h for A-F

add30:
        ADD     DL, 30H                 ; Add 30h to convert to ASCII
        MOV     [RSI], DL               ; Store ASCII digit
        DEC     RSI                     ; Move to previous position
        DEC     RCX                     ; Decrement counter
        JNZ     next_digit              ; Repeat for next digit if
counter is not zero

        Print   char_ans, 4             ; Print 4 characters
        ret

; Accept 4-digit hexadecimal number
Accept_16:
        Read    buf, 5                  ; Read 4 digits + enter

        MOV     RCX, 4                  ; Set counter to 4
        MOV     RSI, buf                ; Point to buffer
        XOR     BX, BX                  ; Clear BX

next_byte:
        SHL     BX, 4                   ; Shift BX left by 4 bits
        MOV     AL, [RSI]               ; Load next character
        CMP     AL, '0'                 ; Check if it's a valid hex
digit
        JB      error                   ; Jump to error if not
        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          ; Print error message
        Exit
sub57:  SUB     AL, 20H                 ; Convert lowercase to
uppercase
sub37:  SUB     AL, 07H                 ; Convert A-F to 0-9
sub30:  SUB     AL, 30H                 ; Convert ASCII to number
        ADD     BX, AX                  ; Add to BX
        INC     RSI                     ; Move to next character
        DEC     RCX                     ; Decrement counter
        JNZ     next_byte               ; Repeat for next byte if
counter is not zero

        RET
```

Output :

```
    Assignment no    :3
------------------------------------------------------------
    Assignment Name:Conversion From HEX to BCD and BCD to HEX Number.
----------------------------------------------------------

1.Hex To BCD.
2.BCD To Hex.
3.Exit.
Enter Your Choice::
Enter 4 digit Hex Number::
The Equivalent BCD Number is::4369
1.Hex To BCD.
2.BCD To Hex.
3.Exit.
Enter Your Choice::
Enter 5 digit BCD Number::
The Equivalent Hex Number is::2B67
1.Hex To BCD.
2.BCD To Hex.
3.Exit.
Enter Your Choice::


;-----------------------------------END---------------------------------
--
```