section .data

menumsg db 10,10,'###### Menu for Code Conversion ######'

db 10,'1: Hex to BCD'

db 10,'2: BCD to Hex'

db 10,'3: Exit'

db 10,10,'Please Enter Choice::'

menumsg\_len equ $-menumsg

hexinmsg db 10,10,'Please enter 4 digit hex number::'

hexinmsg\_len equ $-hexinmsg

bcdopmsg db 10,10,'BCD Equivalent::'

bcdopmsg\_len equ $-bcdopmsg

bcdinmsg db 10,10,'Please enter 5 digit BCD number::'

bcdinmsg\_len equ $-bcdinmsg

hexopmsg db 10,10,'Hex Equivalent::'

hexopmsg\_len equ $-hexopmsg

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

section .bss

numascii resb 06 ;common buffer for choice, hex and bcd input

outputbuff resb 02

dispbuff resb 08

%macro display 2

mov rax,01

mov rdi,01

mov rsi,%1

mov rdx,%2

syscall

%endmacro

%macro accept 2

mov rax,0

mov rdi,0

mov rsi,%1

mov rdx,%2

syscall

%endmacro

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

section .text

global \_start

\_start:

menu:

display menumsg,menumsg\_len

accept numascii,2

cmp byte [numascii],'1'

je hex2bcd\_proc

cmp byte [numascii],'2'

je bcd2hex\_proc

cmp byte [numascii],'3'

je exit

jmp \_start

exit:

mov rax,60

mov rbx,0

syscall

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

hex2bcd\_proc:

display hexinmsg,hexinmsg\_len

accept numascii,5

call packnum ;packnum converts hexadecimal number to binary and stores result in bx register

mov ax,bx ;we store value of bx in ax

mov rcx,0 ;initialise the count to 0

mov bx,10 ;Base of Decimal No. system

h2bup1: ;procedure responsible for repeatedly dividing number by 10 until quo becomes zero

mov dx,0 ; Since the **div** instruction considers both **dx** and **ax** as part of the dividend, it's essential to ensure that **dx** is initialized to 0 before each division operation to prevent any unwanted values from affecting the result.

div bx ;earlier bx contained 10 and ax contains our packed bcd num

push rdx

inc rcx

cmp ax,0 ;we check if ax (register that contains the quotient ) has turned to 0

jne h2bup1

mov rdi,outputbuff ; sets the value of the register **rdi** to the memory address where the variable **outputbuff** is located.

By moving the memory address of **outputbuff** into the **rdi** register, the instruction prepares **rdi** to point to the beginning of the memory block reserved for **outputbuff**.

h2bup2:

pop rdx

add dl,30h ; remainder is stored in dx register lower bits of dx is dl

1. **Usage of dl**:
   * After the **div** instruction, the remainder is stored in the least significant byte of the **dx** register, which is **dl**.
   * Since we're only interested in the remainder (which is a single byte), we operate on **dl** directly.
   * Using **dl** instead of **dx** ensures that we only modify the byte containing the remainder, leaving the most significant byte (**dh**) unaffected.

mov [rdi],dl

* **[rdi]** represents the memory address pointed to by the **rdi** register.
* In this case, **rdi** is set to point to the beginning of the **outputbuff** buffer where the ASCII representation of the BCD digits is to be stored.
* The **mov [rdi], dl** instruction copies the ASCII character in **dl** to the memory location pointed to by **rdi**, which is the next available location in the **outputbuff** buffer.
* This effectively stores the ASCII representation of the BCD digit at the appropriate position in **outputbuff**.

inc rdi

loop h2bup2

display bcdopmsg,bcdopmsg\_len

display outputbuff,5

jmp menu

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bcd2hex\_proc:

display bcdinmsg,bcdinmsg\_len

accept numascii,6

display hexopmsg,hexopmsg\_len

mov rsi,numascii ;let rsi point to input buffer

mov rcx,05 ;let rcx=5 for processing 5 bcd digits

mov rax,0 ;clearing rax to store result

mov ebx,0ah ;set ebx to 10(base of hex)

b2hup1:

mov rdx,0 ;clear rdx for correct multiplication reason same as previous operation

mul ebx ;content in ebx and rax will get multiplied

mov dl,[rsi] ;to fetch next bcd digit from input buffer

sub dl,30h` ; o convert an ASCII character representing a digit to its corresponding numeric value, we need to subtract the ASCII value of '0' (30h)

add rax,rdx ; used to accumulate the numeric value of each BCD digit into **rax**

inc rsi

loop b2hup1

mov ebx,eax

call disp32\_num

jmp menu

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

packnum: ;to convert ascii seq of hexadecimal digits into 16-bit binary coded decimal

;value is stored in bx register

mov bx,0 ;bx will store packed bcd decimal so we clear it

mov ecx,04 ;we will process 4 ascii chars

mov esi,numascii ;let esi pointer point to numascii (our input buffer) that stores ascii chars

up1:

rol bx,04 we rotate the contents of bx to the left by 4 digits

mov al,[esi] ;we move the next character pointed by esi to al register

cmp al,39h

jbe skip1

sub al,07h

skip1:

sub al,30h

add bl,al

inc esi

loop up1

ret

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

disp32\_num:

mov rdi,dispbuff ;point esi to buffer

mov rcx,08 ;load number of digits to display

dispup1:

rol ebx,4 ;rotate number left by four bits

mov dl,bl ;move lower byte in dl

and dl,0fh ;mask upper digit of byte in dl

add dl,30h ;add 30h to calculate ASCII code

cmp dl,39h ;compare with 39h

jbe dispskip1 ;if less than 39h akip adding 07 more

add dl,07h ;else add 07

dispskip1:

mov [rdi],dl ;store ASCII code in buffer

inc rdi ;point to next byte

loop dispup1 ;decrement the count of digits to display

;if not zero jump to repeat

display dispbuff+3,5 ;Dispays only lower 5 digits as upper three are '0'

ret