**16-Bit Addition**

**Program :**

Data segment

    msg db 0dh,0ah,"Enter first number: $"

    msg1 db 0dh,0ah,"Enter second number: $"

    result db 0dh,0ah,"The Result of addition is: $"

Data ends

Code segment

    assume CS:Code,DS:Data

start:

    mov ax, Data

    mov DS, ax

    mov dx,offset msg ;----- Display contents of variable msg

    mov ah,09h

    int 21h

    mov ah,01h ; To accept 4bits 10s input of higher nibble

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input of higher nibble

    int 21h

    call AsciiToHex

    add bl,al

    mov ch,bl

    mov ah,01h ; To accept 4bits 10s input of lower nibble

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input of lower nibble

    int 21h

    call AsciiToHex

    add bl,al

    mov cl,bl

    mov dx,offset msg1 ;----- Display contents of variable msg1

    mov ah,09h

    int 21h

    mov ah,01h ; To accept 4bits 10s input

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input

    int 21h

    call AsciiToHex

    add bl,al

    mov bh,bl

    mov ah,01h ; To accept 4bits 10s input of lower nibble

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input of lower nibble

    int 21h

    call AsciiToHex

    add bl,al

    add bx,cx;-------------------main addn

    mov dx,offset result ; Display contents of string result

    mov ah,09h

    int 21h

    mov cl,bh

    and bh,0f0h

    rol bh,4h; interchange nibbles

    mov  al,bh

    call HexToAscii

    mov al,cl

    and al,0fh

    call HexToAscii

    mov cl,bl

    and bl,0f0h

    rol bl,4h; interchange nibbles

    mov  al,bl

    call HexToAscii

    mov al,cl

    and al,0fh

    call HexToAscii

    mov ah,4ch ; Terminate the program

    int 21h

        AsciiToHex proc

        cmp al,41h ; If it is greater than or equal to 41 then we also need to sub 8h along with 30

        jc skip

        sub al, 07h

        skip: sub al, 30h

        ret

        endp

        HexToAscii proc

        cmp al,0ah ; If it is greater than or equal to 0a then we also need to add 07 along with 30

        jc skip1

        add al,07h

        skip1: add al,30h

        mov dl, al; display

        mov ah,02h

        int 21h

        ret

        endp

Code ends

end start

**16-Bit Subtraction**

**Program :**

Data segment

    msg db 0dh,0ah,"Enter first number: $"

    msg1 db 0dh,0ah,"Enter second number: $"

    result db 0dh,0ah,"The Result of subtraction is: $"

Data ends

Code segment

    assume CS:Code,DS:Data

start:

    mov ax, Data

    mov DS, ax

    mov dx,offset msg ;----- Display contents of variable msg

    mov ah,09h

    int 21h

    mov ah,01h ; To accept 4bits 10s input of higher nibble

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input of higher nibble

    int 21h

    call AsciiToHex

    add bl,al

    mov ch,bl

    mov ah,01h ; To accept 4bits 10s input of lower nibble

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input of lower nibble

    int 21h

    call AsciiToHex

    add bl,al

    mov cl,bl

    mov dx,offset msg1 ;----- Display contents of variable msg1

    mov ah,09h

    int 21h

    mov ah,01h ; To accept 4bits 10s input

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input

    int 21h

    call AsciiToHex

    add bl,al

    mov bh,bl

    mov ah,01h ; To accept 4bits 10s input of lower nibble

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input of lower nibble

    int 21h

    call AsciiToHex

    add bl,al

    sub bx,cx;-------------------main subn

    mov dx,offset result ; Display contents of string result

    mov ah,09h

    int 21h

    mov cl,bh

    and bh,0f0h

    rol bh,4h; interchange nibbles

    mov  al,bh

    call HexToAscii

    mov al,cl

    and al,0fh

    call HexToAscii

    mov cl,bl

    and bl,0f0h

    rol bl,4h; interchange nibbles

    mov  al,bl

    call HexToAscii

    mov al,cl

    and al,0fh

    call HexToAscii

    mov ah,4ch ; Terminate the program

    int 21h

        AsciiToHex proc

        cmp al,41h ; If it is greater than or equal to 41 then we also need to sub 8h along with 30

        jc skip

        sub al, 07h

        skip: sub al, 30h

        ret

        endp

        HexToAscii proc

        cmp al,0ah ; If it is greater than or equal to 0a then we also need to add 07 along with 30

        jc skip1

        add al,07h

        skip1: add al,30h

        mov dl, al; display

        mov ah,02h

        int 21h

        ret

        endp

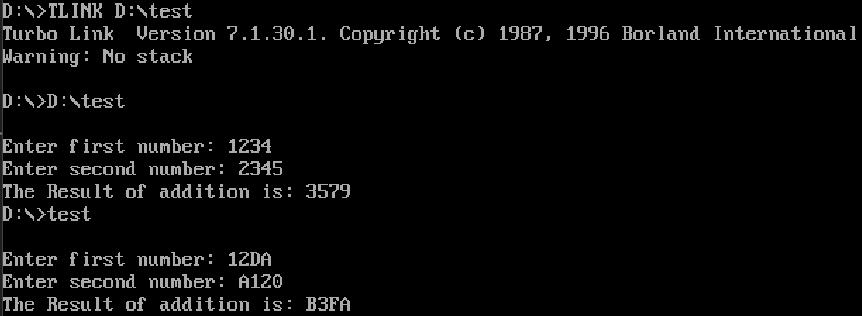
Code ends

end start

**Output :**



**Output :**



**16-Bit Multiplication**

**Program :**

Data segment

    msg db 0dh,0ah,"Enter first number: $"

    msg1 db 0dh,0ah,"Enter second number: $"

    result db 0dh,0ah,"The Result of multiplication is: $"

Data ends

Code segment

    assume CS:Code,DS:Data

start:

    mov ax, Data

    mov DS, ax

    mov dx,offset msg ;----- Display contents of variable msg

    mov ah,09h

    int 21h

    mov ah,01h ; To accept 4bits 10s input of higher nibble

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input of higher nibble

    int 21h

    call AsciiToHex

    add bl,al

    mov ch,bl

    mov ah,01h ; To accept 4bits 10s input of lower nibble

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input of lower nibble

    int 21h

    call AsciiToHex

    add bl,al

    mov cl,bl

    mov dx,offset msg1 ;----- Display contents of variable msg1

    mov ah,09h

    int 21h

    mov ah,01h ; To accept 4bits 10s input of higher nibble

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input of higher nibble

    int 21h

    call AsciiToHex

    add bl,al

    mov bh,bl

    mov ah,01h ; To accept 4bits 10s input of lower nibble

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input of lower nibble

    int 21h

    call AsciiToHex

    add bl,al

    mov ax,bx

    mul cx;-------------------main multip

    mov bx,dx

    mov ch,ah

    mov cl,al

    mov dx,offset result ; Display contents of string result

    mov ah,09h

    int 21h

    mov dh,cl

    mov cl,bh

    and bh,0f0h

    rol bh,4h; interchange nibbles

    mov  al,bh

    call HexToAscii

    mov al,cl

    and al,0fh

    call HexToAscii

    mov cl,bl

    and bl,0f0h

    rol bl,4h; interchange nibbles

    mov  al,bl

    call HexToAscii

    mov al,cl

    and al,0fh

    call HexToAscii ; displayed dx

    mov cl,ch

    and ch,0f0h

    rol ch,4h; interchange nibbles

    mov  al,ch

    call HexToAscii

    mov al,cl

    and al,0fh

    call HexToAscii

    mov cl,dh

    and dh,0f0h

    rol dh,4h; interchange nibbles

    mov  al,dh

    call HexToAscii

    mov al,cl

    and al,0fh

    call HexToAscii; displayed ax

    mov ah,4ch ; Terminate the program

    int 21h

        AsciiToHex proc

        cmp al,41h ; If it is greater than or equal to 41 then we also need to sub 8h along with 30

        jc skip

        sub al, 07h

        skip: sub al, 30h

        ret

        endp

        HexToAscii proc

        cmp al,0ah ; If it is greater than or equal to 0a then we also need to add 07 along with 30

        jc skip1

        add al,07h

        skip1: add al,30h

        mov dl, al; display

        mov ah,02h

        int 21h

        ret

        endp

Code ends

end start

**Output :**



**16-Bit Division**

**Program :**

Data segment

    msg db 0dh,0ah,"Enter the denominator: $"

    msg1 db 0dh,0ah,"Enter the numerator: $"

    result db 0dh,0ah,"The Quotient is: $"

    result1 db 0dh,0ah,"The Result of Remainder is: $"

Data ends

Code segment

    assume CS:Code,DS:Data

start:

    mov ax, Data

    mov DS, ax

    mov dx,offset msg ;----- Display contents of variable msg

    mov ah,09h

    int 21h

    mov ah,01h ; To accept 4bits 10s input

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input

    int 21h

    call AsciiToHex

    add bl,al

    mov cl,bl

    mov dx,offset msg1 ;----- Display contents of variable msg1

    mov ah,09h

    int 21h

    mov ah,01h ; To accept 4bits 10s input of higher nibble

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input of higher nibble

    int 21h

    call AsciiToHex

    add bl,al

    mov bh,bl

    mov ah,01h ; To accept 4bits 10s input of lower nibble

    int 21h

    call AsciiToHex

    mov bl, al

    rol bl,4

    mov ah,01h ; To accept 4bits units input of lower nibble

    int 21h

    call AsciiToHex

    mov ah,00h

    div cl;--------------main divn --al(quotient), ah(remainder)

    mov bx,ax

    mov dx,offset result ; Display contents of string result

    mov ah,09h

    int 21h

    mov cl,bl

    and bl,0f0h

    rol bl,4h; interchange nibbles

    mov  al,bl

    call HexToAsciiDisp

    mov al,cl

    and al,0fh

    call HexToAsciiDisp

    mov dx,offset result1 ; Display contents of string result

    mov ah,09h

    mov cl,bl

    and bl,0f0h

    rol bl,4h; interchange nibbles

    mov  al,bl

    call HexToAsciiDisp

    mov al,cl

    and al,0fh

    call HexToAsciiDisp

    mov ah,4ch ; Terminate the program

    int 21h

        AsciiToHex proc

        cmp al,41h ; If it is greater than or equal to 41 then we also need to sub 8h along with 30

        jc skip

        sub al, 07h

        skip: sub al, 30h

        ret

        endp

        HexToAsciiDisp proc

        cmp al,0ah ; If it is greater than or equal to 0a then we also need to add 07 along with 30

        jc skip1

        add al,07h

        skip1: add al,30h

        mov dl, al

        mov ah,02h

        int 21h

        ret

        endp

Code ends

end start

**Output :** 