**16 Bit Addition:**

Data Segment

msg db 0dh,0ah,"Enter a 16-bit number: $"

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

newl db 0dh,0ah," $"

Data ends

Code Segment

assume CS:Code,DS:Data

Start:

mov ax,Data

mov DS,ax

mov dx,offset msg;add16

mov ah,09h

int 21h

call AcceptNum

mov bh,bl

call AcceptNum

mov cx,bx

mov dx,offset msg

mov ah,09h

int 21h

call AcceptNum

mov bh,bl

call AcceptNum

add cx,bx

mov dx,offset result

mov ah,09h

int 21h

mov bl,ch

call DispNum

mov bl,cl

call DispNum

mov ah,4ch

int 21h

AcceptNum proc

mov ah,01h

int 21h

call HexAccept

mov bl,al

rol bl,4

mov ah,01h

int 21h

call HexAccept

add bl,al

ret

endp

DispNum proc

mov al,bl

and al,0f0h

ror al,4

mov dl,al

call HexDisp

mov ah,02h

int 21h

mov al,bl

and al,0fh

mov dl,al

call HexDisp

mov ah,02h

int 21h

endp

HexAccept proc

cmp al,41h

jc norm

sub al,07h

norm: sub al,30h

ret

endp

HexDisp proc

cmp dl,0ah

jc nothex

add dl,07h

nothex: add dl,30h

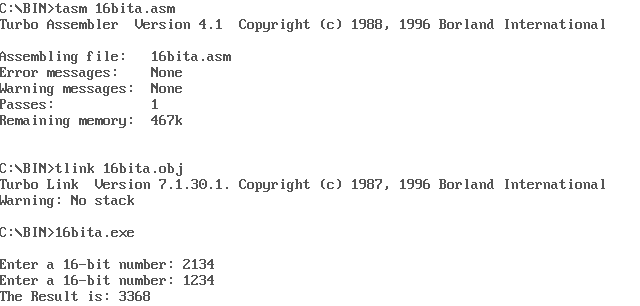
ret

endp

Code ends

end Start

**OUTPUT:**



**16 Bit Subtraction:**

Data Segment

msg db 0dh,0ah,"Enter a 16-bit number: $"

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

newl db 0dh,0ah," $"

Data ends

Code Segment

assume CS:Code,DS:Data

Start:

mov ax,Data

mov DS,ax

mov dx,offset msg;sub16

mov ah,09h

int 21h

call AcceptNum

mov bh,bl

call AcceptNum

mov cx,bx

mov dx,offset msg

mov ah,09h

int 21h

call AcceptNum

mov bh,bl

call AcceptNum

sub cx,bx

mov dx,offset result

mov ah,09h

int 21h

mov bl,ch

call DispNum

mov bl,cl

call DispNum

mov ah,4ch

int 21h

AcceptNum proc

mov ah,01h

int 21h

call HexAccept

mov bl,al

rol bl,4

mov ah,01h

int 21h

call HexAccept

add bl,al

ret

endp

DispNum proc

mov al,bl

and al,0f0h

ror al,4

mov dl,al

call HexDisp

mov ah,02h

int 21h

mov al,bl

and al,0fh

mov dl,al

call HexDisp

mov ah,02h

int 21h

endp

HexAccept proc

cmp al,41h

jc norm

sub al,07h

norm: sub al,30h

ret

endp

HexDisp proc

cmp dl,0ah

jc nothex

add dl,07h

nothex: add dl,30h

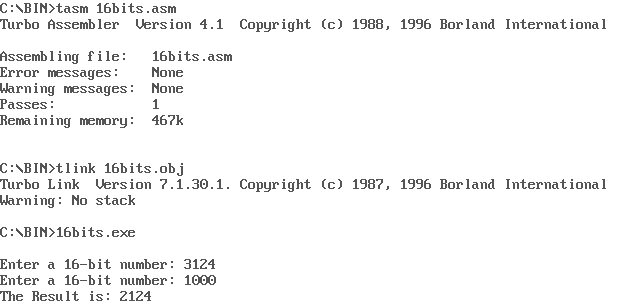
ret

endp

Code ends

end Start

**OUTPUT:**



**16 Bit Multiplication:**

Data Segment

msg db 0dh,0ah,"Enter a 16-bit number: $"

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

newl db 0dh,0ah," $"

mul\_res\_h dw ?

mul\_res\_l dw ?

Data ends

Code Segment

assume CS:Code,DS:Data

Start:

mov ax,Data

mov DS,ax

mov dx,offset msg ;mul16

mov ah,09h

int 21h

call AcceptNum

mov ch,bl

call AcceptNum

mov cl,bl

mov dx,offset msg

mov ah,09h

int 21h

call AcceptNum

mov dh,bl

call AcceptNum

mov dl,bl

mov ax,cx

mov bx,dx

mul bx

mov mul\_res\_l,ax

mov mul\_res\_h,dx

mov dx,offset result

mov ah,09h

int 21h

mov ax,mul\_res\_h

mov bl,ah

call DispNum

mov ax,mul\_res\_h

mov bl,al

call DispNum

mov ax,mul\_res\_l

mov bl,ah

call DispNum

mov ax,mul\_res\_l

mov bl,al

call DispNum

mov ah,4ch

int 21h

AcceptNum proc

mov ah,01h

int 21h

call HexAccept

mov bl,al

rol bl,4

mov ah,01h

int 21h

call HexAccept

add bl,al

ret

endp

DispNum proc

mov al,bl

and al,0f0h

ror al,4

mov dl,al

call HexDisp

mov ah,02h

int 21h

mov al,bl

and al,0fh

mov dl,al

call HexDisp

mov ah,02h

int 21h

endp

HexAccept proc

cmp al,41h

jc norm

sub al,07h

norm: sub al,30h

ret

endp

HexDisp proc

cmp dl,0ah

jc nothex

add dl,07h

nothex: add dl,30h

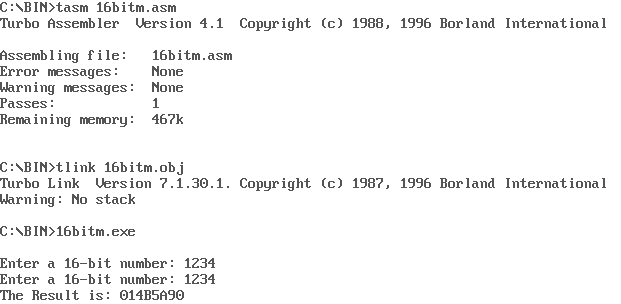
ret

endp

Code ends

end Start

**OUTPUT:**



**16 Bit Division:**

Data segment

msg db 0dh,0ah,"Enter Dividend: $"

msg1 db 0dh,0ah,"Enter Divisor: $"

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

resr db 0dh,0ah,"The Remainder is: $"

Data ends

Code segment

assume CS:Code,DS:Data

start:

mov ax,Data ; Move Data to Data Segment

mov DS,ax

mov dx,offset msg ; Display contents of variable msg div16

mov ah,09h

int 21h

mov ah,01h ; To accept input and store ASCII value into al

int 21h

call AsciitoHex

mov bh,al ; Accept 1000's place of the Number

rol bh,4

mov ah,01h ; To accept input and store ASCII value into al

int 21h

call AsciitoHex

add bh,al

mov ah,01h ; To accept input and store ASCII value into al

int 21h

call AsciitoHex

mov bl,al ; Accept 10's place of the Number

rol bl,4

mov ah,01h ; To accept input and store ASCII value into al

int 21h

call AsciitoHex ; Accept unit's place of Number

add bl,al ; Get the number by adding 10's and unit's place

mov dx,offset msg1 ; Display contents of variable msg1

mov ah,09h

int 21h

mov ah,01h ; To accept input and store ASCII value into al

int 21h

call AsciitoHex ; Accept 10's place of the Number

mov cl,al

rol cl,4

mov ah,01h ; To accept input and store ASCII value into al

int 21h

call AsciitoHex ; Accept unit's place of Number

add cl,al ; Get the number by adding 10's and unit's place

mov ax,bx ; Divide the two accepted Number's

mov bl,cl

div bl

mov cx,ax ; Store the value of the Result

mov dx,offset resq ; Display contents of string resq

mov ah,09h

int 21h

mov ax,cx ; Retreive Result

and al,0f0h ; Isolate 10's place of the Quotient

ror al,4

mov bl,al ; Convert to ASCII to display

call AsciiConv

mov dl,bl ; Display a Number/Alphabet

mov ah,02h

int 21h

mov ax,cx ; Retrieve original Result

and al,0fh ; Isolate unit's place of Result

mov bl,al ; Convert to ASCII to display

call AsciiConv

mov dl,bl ; Display a Number/Alphabet

mov ah,02h

int 21h

mov dx,offset resr ; Display contents of string resr

mov ah,09h

int 21h

mov ax,cx ; Retrieve original Result

and ah,0f0h ; Isolate 10's place of the Quotient

ror ah,4

mov bl,ah ; Convert to ASCII to display

call AsciiConv

mov dl,bl ; Display a Number/Alphabet

mov ah,02h

int 21h

mov ax,cx ; Retrieve original Result

and ah,0fh ; Isolate unit's place of Result

mov bl,ah ; Convert to ASCII to display

call AsciiConv

mov dl,bl ; Display a Number/Alphabet

mov ah,02h

int 21h

mov ah,4ch ; Terminate the program

int 21h

AsciiConv proc ; Compare to 0a if it is less than A then we need to add only 30

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

jc skip

add bl,07h

skip: add bl,30h

ret

endp

AsciitoHex proc ; Compare to 41 if it is less than A then we need to sub only 30

cmp al,41h ; If it is greater than or equal to 41 then we also need to sub 07

jc skippy

sub al,07h

skippy: sub al,30h

ret

endp

Code ends

end start

**OUTPUT:**

