Keegan Vaz
42
SECMPNB

Experiment 2

Aim: To implement addition and subtraction of 8 bit and 16 bit numbers, multiplication of 8 bit numbers and division of a 16 bit number by an 8 bit number

Prerequisite: TASM assembler

TASM(Turbo Assembler):

TASM itself is a 16-bit program.It runs on and produces code for 16- or 32-bit x86 MS-DOS and compatibles or Microsoft Windows. It will run on 16- and 32-bit versions of Windows, and produce code for the same versions, but it does not generate 64-bit x86 code. TASM is an assembler for x86 architecture, the job of assembler is to convert the assembly language program to machine language.

Theory:

To get the digits of an 8-bit number, we can use the masking operation. At first we will mask the upper nibble, and then the lower nibble. After masking the upper nibble, we have to rotate it to the right to make it least significant nibble. Then we can simply add it to the stored nibble to get the sum.

1A)Addition of two 8 bit numbers

Algorithm:

Step I : Initialize the data segment.

Step II : Get the first number in AX register.

Step III : Get the second number in BX register.

Step IV : Add the two numbers.

Step V : Display the result.

Step VI : Stop

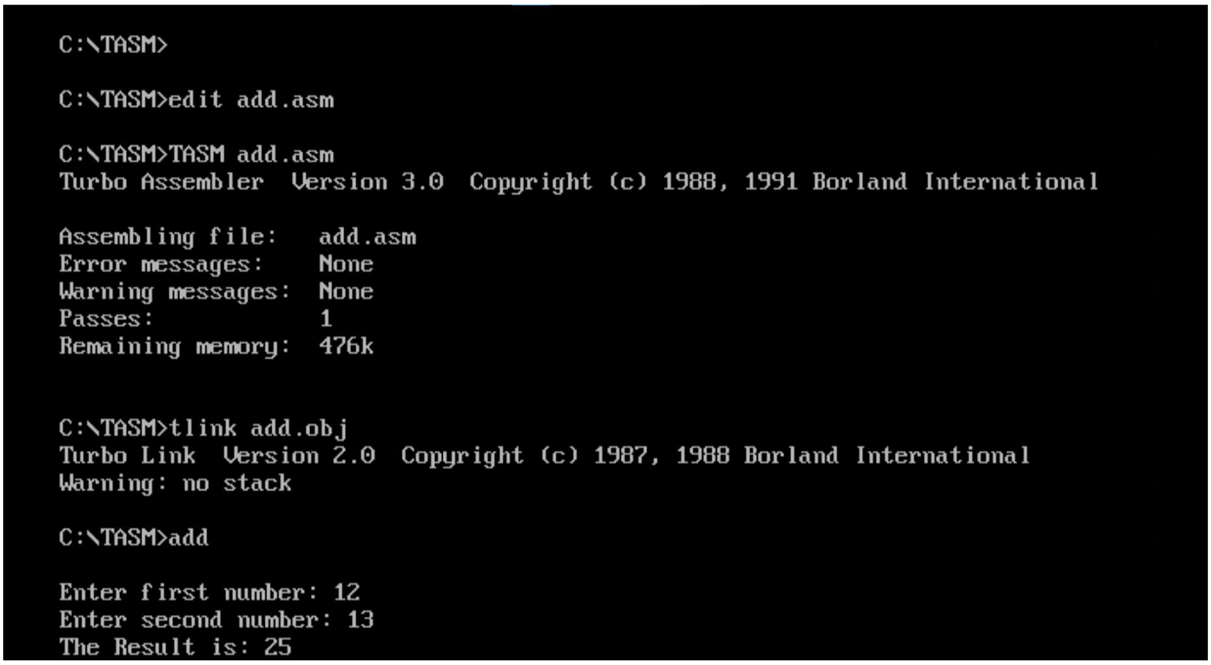
Code:

```
Data segment

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

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

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

Data ends

Code segment

 assume CS:Code,DS:Data

start:
```

```
mov ax,Data ; Move Data to Data Segment add8
mov DS,ax
mov dx,offset msg ; Display contents of variable msg
mov ah,09h
int 21h
mov ah,01h ; To accept input and store ASCII value into al(eg: 31h)
int 21h
sub al,30h ; Accept 10's place of the Number (31h becomes 01h)
mov bl,al
;mov cl,4
;rol bl,cl
rol bl,1
rol bl,1
rol bl,1
rol bl,1 ;(after rol cmds 01h becomes 10h)
mov ah,01h ; To accept input and store ASCII value into al (eg: 35h)
int 21h
sub al,30h ; Accept unit's place of Number (35h becomes 05h)
add bl,al ; Get the number by adding 10's and unit's place (10h + 05h = 15h)
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 (32h)
int 21h
sub al,30h ; Accept 10's place of the Number(02h)
mov cl,al
;rol cl,4
rol cl,1
rol cl,1
rol cl,1
rol cl,1 ;(20h)
```

```
mov ah,01h ; To accept input and store ASCII value into al (33h)
int 21h
sub al,30h ; Accept unit's place of Number (03h)
add cl,al ; Get the number by adding 10's and unit's place (20h+03h=23h)
add bl,cl ; Add the two accepted Number's(15h + 23h = 38h)
mov dx,offset result ; Display contents of string result
mov ah,09h
int 21h
mov cl,bl ; Store the value of the Result
and bl,0f0h ; Isolate 10's place of Result(30h)
;mov cl,4
;ror bl,cl
ror bl,1
ror bl,1
ror bl,1
ror bl,1
call AsciiConv ; Convert to ASCII to display
mov dl,bl ; Display a Number/Alphabet
mov ah,02h
int 21h
mov bl,cl ; Retrieve original Result
and bl,0fh ; Isolate unit's place of Result(08h)
call AsciiConv ; Convert to ASCII to display
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

Code ends

end start
```

Output:



```
C:\TASM>

C:\TASM>edit add.asm

C:\TASM>TASM add.asm
Turbo Assembler  Version 3.0  Copyright (c) 1988, 1991 Borland International

Assembling file:   add.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  476k


C:\TASM>tlink add.obj
Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International
Warning: no stack

C:\TASM>add

Enter first number: 12
Enter second number: 13
The Result is: 25
```

Conclusion:

From the user we are able to get two 8 bit numbers, then store the numbers we then need to add the two 8 bit numbers and store it in result. Atlast we can compile the assembly program in tasm.

Aim: Write an assembly language program for 8 bit subtraction.

Prerequisite: TASM assembler

Theory:

Consider that a byte of data is present in the AL register and a second byte of data is present in the BL register. We have to subtract the byte in BL from the byte in AL. Using sub instruction subtract the contents of two registers. Results will be stored in the AL register. Display the result using display routine.

1B) Subtraction of two 8 bit numbers Algorithm:

Step I : Initialize the data segment.

Step II : Get the first number in AX register.

Step III : Get the second number in BX register.

Step IV : Subtract the two numbers.

Step V : Display the result.

Step VI : Stop

Code:

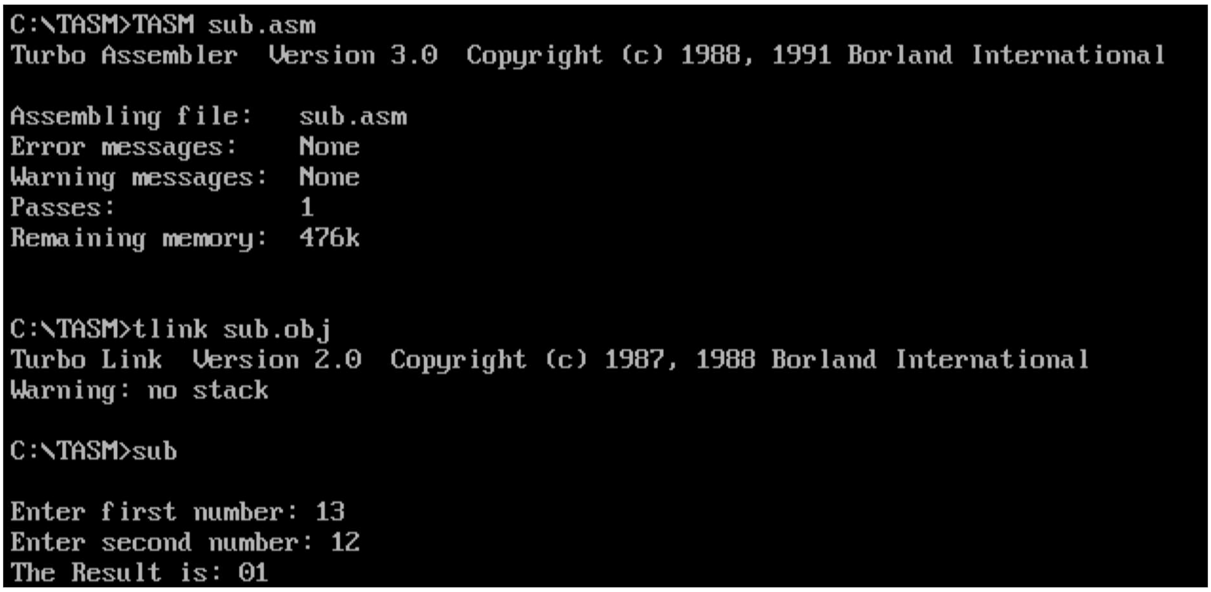
```
Data segment
 msg db 0dh,0ah,"Enter first number: $"
 msg1 db 0dh,0ah,"Enter second number: $"
 result db 0dh,0ah,"The Result is: $"
Data ends
Code segment
 assume CS:Code,DS:Data
start:
 mov ax,Data ; Move Data to Data Segment add8
 mov DS,ax

 mov dx,offset msg ; Display contents of variable msg
 mov ah,09h
 int 21h
 mov ah,01h ; To accept input and store ASCII value into al
```

```
int 21h
sub al,30h ; Accept 10's place of the Number
mov bl,al
;mov cl,4
;rol bl,cl
rol bl,1
rol bl,1
rol bl,1
rol bl,1
mov ah,01h ; To accept input and store ASCII value into al
int 21h
sub al,30h ; 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
sub al,30h ; Accept 10's place of the Number
mov cl,al
;rol cl,4
rol cl,1
rol cl,1
rol cl,1
rol cl,1

mov ah,01h ; To accept input and store ASCII value into al
int 21h
sub al,30h ; Accept unit's place of Number
add cl,al ; Get the number by adding 10's and unit's place
sub bl,cl ; Add the two accepted Number's
```

```
mov dx,offset result ; Display contents of string result

mov ah,09h

int 21h

mov cl,bl ; Store the value of the Result

and bl,0f0h ; Isolate 10's place of Result

;mov cl,4

;ror bl,cl

ror bl,1

ror bl,1

ror bl,1

ror bl,1

call AsciiConv ; Convert to ASCII to display

mov dl,bl ; Display a Number/Alphabet

mov ah,02h

int 21h

mov bl,cl ; Retrieve original Result

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

call AsciiConv ; Convert to ASCII to display

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

Code ends
```

end start

Output:



```
C:\TASM>TASM sub.asm
Turbo Assembler  Version 3.0  Copyright (c) 1988, 1991 Borland International

Assembling file:   sub.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  476k


C:\TASM>tlink sub.obj
Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International
Warning: no stack

C:\TASM>sub

Enter first number: 13
Enter second number: 12
The Result is: 01
```

Conclusion:

From the user we are able to get two 8 bit numbers, then store the numbers we then need to subtract the two 8 bit numbers and store it in result. Atlast we can compile the assembly program in tasm.

Aim: Write an assembly language program for 16 bit addition.

Prerequisite: TASM assembler

Theory:

We use registers AX and BX to take the first and second number to find the sum of two numbers. Consider that a word of data is present in the AX register and a 2nd word of data is present in the BX register. We have to add words in AX with the word in BX. Using ADD instruction, add the contents. Results will be stored in the AX register. Display the result using display routine.

2A) Addition of two 16 bit numbers

Algorithm:

Step I : Initialize the data segment.

Step II : Get the first number in AX register.

Step III : Get the second number in BX register.

Step IV : Add the two numbers.

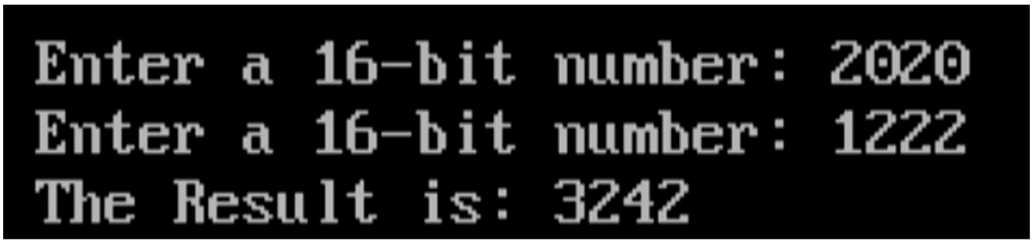
Step V : Display the result.

Step VI : Stop

Code:

```
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
 rol bl,1
 rol bl,1
 rol bl,1
 rol bl,1
 mov ah,01h
 int 21h
 call HexAccept
 add bl,al
 ret
endp
DispNum proc
 mov al,bl
 and al,0f0h
 ;ror al,4
 ror al,1
 ror al,1
 ror al,1
 ror al,1
 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:



```
Enter a 16-bit number: 2020
Enter a 16-bit number: 1222
The Result is: 3242
```

Conclusion:

From the user we are able to get two 16 bit numbers, then store the numbers we then need to add the two 16 bit numbers and store it in result. Atlast we can compile the assembly program in tasm.

Aim: Write an assembly language program for 16 bit subtraction**.**

Prerequisite: TASM assembler

Theory:

Consider that a word of data is present in the AX register and a second word of data is present in the BX register. We have to subtract the word in BX with the word AX. Using subtract (SUB) instruction, subtract the contents of two registers. Results will be stored in the AX register. Display the result using display routine.

2B) Subtraction of two 16 bit numbers

Algorithm: Step I : Initialize the data segment.

Step II : Get the first number in AX register.

Step III : Get the second number in BX register.

Step IV : Subtract the two numbers.

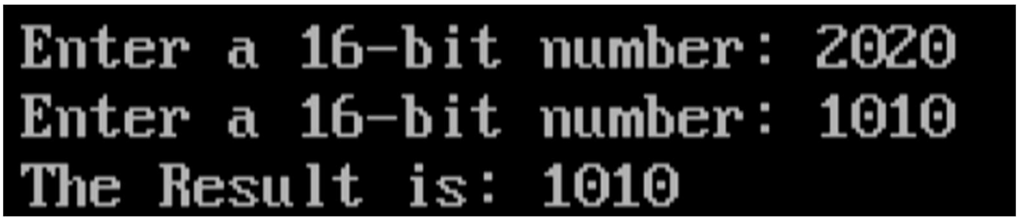
Step V : Display the result.

Step VI : Stop

Code:

```
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

 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
 rol bl,1
 rol bl,1
 rol bl,1
 rol bl,1
 mov ah,01h
 int 21h
 call HexAccept
 add bl,al
 ret
endp
DispNum proc
 mov al,bl
 and al,0f0h
 ;ror al,4
 ror al,1
 ror al,1
 ror al,1
 ror al,1
 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:



```
Enter a 16-bit number: 2020
Enter a 16-bit number: 1010
The Result is: 1010
```

Conclusion:

From the user we are able to get two 16 bit numbers, then store the numbers we then need to subtract the two 16 bit numbers and store it in result. Atlast we can compile the assembly program in tasm.

Aim: Write an assembly language program for 8 bit multiplication.

Prerequisite: TASM assembler

Theory:

Program should load the first number and second number in registers AL and BL registers . Then it should implement some logic for multiplication of two numbers . Consider that a byte of data is present in the AL register and a second byte of data is present in the BL register. We have to multiply the byte in AL with the byte in BL. Using MUL instruction, multiply the contents of two registers. The multiplication of two 8 bit numbers may result in a 16 bit number. So the result is stored in the AX register. The MSB is stored in AH and LSB in AL.

3) Multiplication of two 8 bit numbers

Algorithm:

Step I : Initialize the data segment.

Step II : Get the first number in AL register.

Step III : Get the second number in BL register.

Step IV : Multiply the two numbers.

Step V : Display the result.

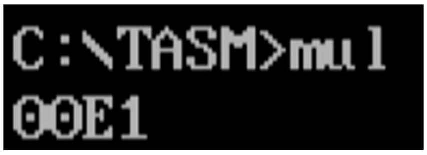
Step VI : Stop

Code:

```
.model small
.data
a db 09H
b db 02H
.code
mov ax, @data ; Initialize data section
mov ds, ax
mov ah, 0
mov al, a ; Load number1 in al
mov bl, b ; Load number2 in bl
mul bl ; multiply numbers and result in ax
mov ch, 04h ; Count of digits to be displayed
mov cl, 04h ; Count to roll by 4 bits
mov bx, ax ; Result in reg bx
l2: rol bx, cl ; roll bl so that msb comes to lsb
```

```
mov dl, bl ; load dl with data to be displayed

and dl, 0fH ; get only lsb

cmp dl, 09 ; check if digit is 0-9 or letter A-F

jbe l4

add dl, 07 ; if letter add 37H else only add 30H

l4: add dl, 30H

mov ah, 02 ; Function 2 under INT 21H (Display character)

int 21H

dec ch ; Decrement Count

jnz l2

mov ah, 4cH ; Terminate Program

int 21H

end
```

Output:



```
C:\TASM>mul
00E1
```

Conclusion:

We had set two 8 bit numbers, then store the numbers we then need to multiply the two 8 bit numbers and store it in a 16-bit register. Atlast we can compile the assembly program written in tasm and see the final output.

Aim: Write an assembly language program for Division of 16 bit number with 8 bit number.

Prerequisite: TASM assembler

Theory:

8086 has DIV instruction to perform division. Take the 8-bit number into BL, and 16-bit number into AX. Now divide AX by BL. The result will be stored at AX. We are taking two numbers, a 16 bit dividend and 8 bit divisor. Then we display the remainder as well as the quotient. The contents of quotient are moved in DI.

4) Division of a 16 bit number by an 8 bit number

Algorithm:

1. Assign values in SI and DI
2. Move the contents of [SI] in BL and increment SI by 1
3. Move the contents of [SI] and [SI + 1] in AX
4. Use **DIV** instruction to divide AX by BL
5. Move the contents of AX in [DI].
6. Halt the program.

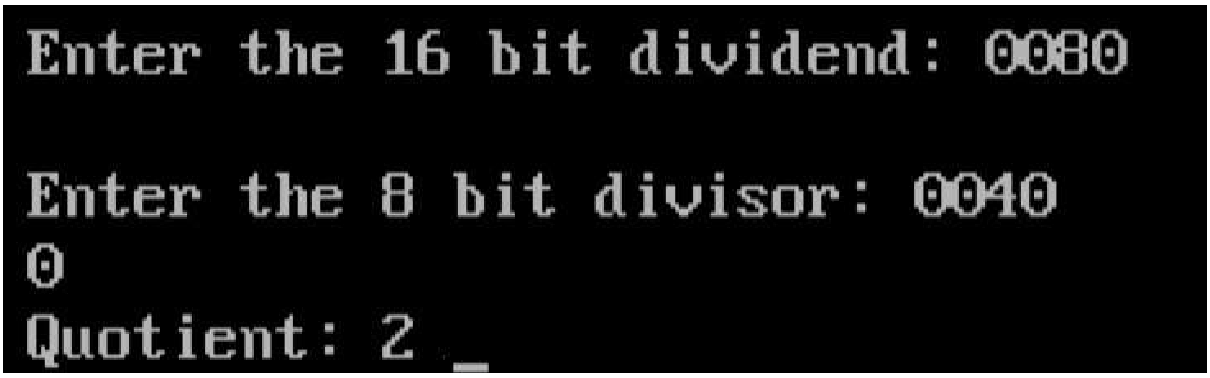
Code:

```
.model small
.data
a dw 0000
c db 0
msg1 db 10,13,"Enter the 16 bit dividend: $"
msg3 db 10,13,"Enter the 8 bit divisor: $"
msgr db 10,13,"Remainder: $"
msgq db 10,13,"Quotient: $"
.stack 100
.code
.startup
mov ax, @data ; Initialize data section
mov ds, ax
lea dx,msg1
```

```
mov ah,09h
int 21h
numh1: mov ah,01h
int 21h
cmp al,'0'
jb ENTER_DIVIDENDH
CMP AL,'9'
JA NUMH1
SUB AL,48
MOV BH,0
MOV BL,AL
MOV AX,[A]
MOV CX,10
MUL CX
ADD AX,BX
MOV [A],AX
JMP NUMH1
ENTER_DIVIDENDH:
CMP AL,13
JNE NUMH1
LEA DX,MSG3
MOV AH,09H
INT 21H
NUML2:
MOV AH,01H
INT 21H
CMP AL,'0'
JB ENTER_DIVIDENDL2
CMP AL,'9'
JA NUML2
SUB AL,48
MOV BL,AL
MOV AL,[C]
MOV CL,10
MUL CL
ADD AL,BL
MOV [C],AL
JMP NUML2
ENTER_DIVIDENDL2:
CMP AL,13
JNE NUML2
MOV DX,0
MOV AX,[A]
MOV BH,0
MOV BL,[C]
DIV BX
MOV CX,DX
PUSH CX
MOV [A],AX
POP AX
MOV BP,SP
DIVR: MOV CL,10
DIV CL
MOV BH,AH
MOV BL,0
PUSH BX
MOV AH,0
CMP AL,0
JNE DIVR
DISP1: CMP BP,SP
JE DIVE
POP DX
MOV DL,DH
ADD DL,48
```

```
MOV AH,02H
INT 21H
JMP DISP1
DIVE: LEA DX,MSGQ
MOV AH,09H
INT 21H
MOV BP,SP
DIVQ:
MOV AX,[A]
MOV DX,0
MOV CX,10
DIV CX
PUSH DX
MOV [A],AX
CMP AX,0
JNE DIVQ
DISP2:
CMP BP,SP
JE DONE1
POP DX
ADD DX,48
MOV AH,02H
INT 21H
JMP DISP2
DONE1:
;.EXIT
END
mov ax, a ; Load number1 in ax
mov bl, b ; Load number2 in bl
div bl ; divide numbers. Quotient in al and Rem in ah
mov ch, 04h ; Count of digits to be displayed
mov cl, 04h ; Count to roll by 4 bits
mov bx, ax ; Result in reg bh
rol bx, cl ; roll bl so that msb comes to lsb
mov dl, bl ; load dl with data to be displayed
and dl, 0fH ; get only lsb
cmp dl, 09 ; check if digit is 0-9 or letter A-F
jbe l4
add dl, 07 ; if letter add 37H else only add 30H
add dl, 30H
mov ah, 02 ; Function 2 under INT 21H (Display character)
int 21H
dec ch ; Decrement Count
jnz l2
mov ah, 4cH ; Terminate Program
int 21H
end:
```

Output:



```
Enter the 16 bit dividend: 0080

Enter the 8 bit divisor: 0040
0
Quotient: 2 _
```

Conclusion:

To divide a 16 bit number by a 8 bit number we just have to take a dividend and divisor from the user and by using div we finally get the quotient and the remainder printed using tasm.