Name:	BARI ANKIT VINOD
Roll No:	65
Class/Sem:	SE/IV
Experiment No.:	1
Title:	To perform basic arithmetic operations on 16-bit data.
Date of Performance:	24/01/24
Date of Submission:	24/01/24
Marks:	
Sign of Faculty:	

Aim: Assembly Language Program to perform basic arithmetic operations (addition, subtraction, multiplication, and division) on 16-bit data.

Theory:

MOV: MOV Destination, Source.

The MOV instruction copies data from a specified destination. word or byte of data from a specified destination.

Source: Register, Memory Location, Immediate Number

Destination: Register, Memory Location

MOV CX, 037AH; Put immediate number 037AH to CX.

ADD: ADD Destination, Source.

These instructions add a number source to a number from some destination and put the result in the specified destination.

Source: Register, Memory Location, Immediate Number

Destination: Register, Memory Location

The source and the destination in an instruction cannot both be memory locations.

ADD AL, 74H; add the immediate number to 74H to the content of AL. Result in AL.

SUB: SUB Destination, Source.

These instructions subtract the number in some source from the number in some destination and put the result in the destination.

Source: Immediate Number, Register, or Memory Location.

Destination: Register or a Memory Location.

The source and the destination in an instruction cannot both be memory locations.

SUB AX, 3427H; Subtract immediate number 3427H from AX.

MUL: MUL Source.

This instruction multiplies an unsigned byte from some source times an unsigned byte in the AL register or an unsigned word from some source times an unsigned word in the AX register.

Source: Register, Memory Location.

MUL CX; Multiply AX with CX; result in high word in DX, low word in AX.

DIV: DIV Source.

This instruction is used to divide an unsigned word by a byte or to divide an unsigned double word (32 bits) by a word.

Source: Register, Memory Location.

If the divisor is 8-bit, then the dividend is in AX register. After division, the quotient is in AL and the remainder in AH.

If the divisor is 16-bit, then the dividend is in DX-AX register. After division, the quotient is in AX and the remainder in DX.

DIV CX; divide double word in DX and AX by word in CX; Quotient in AX; and remainder in DX.

Algorithm to add two 16-bit numbers

1. Load the first number in AX
2. Load the second number in BX

3 Add the second number to AX

4. Store the result in AX.

Algorithm to subtract two 16-bit numbers

1. Load the first number in AX.
2. Load the second number. in BX 3. Subtract the second number to AX

4. Store the result in AX.

Algorithm to multiply a 16-bit number by an 8-bit number

1. Load the first number in AX.
2. Load the second number. in BL
3. Multiply DX and AX.
4. The result is in DX and AX.

Algorithm to divide a 16-bit number by an 8-bit number

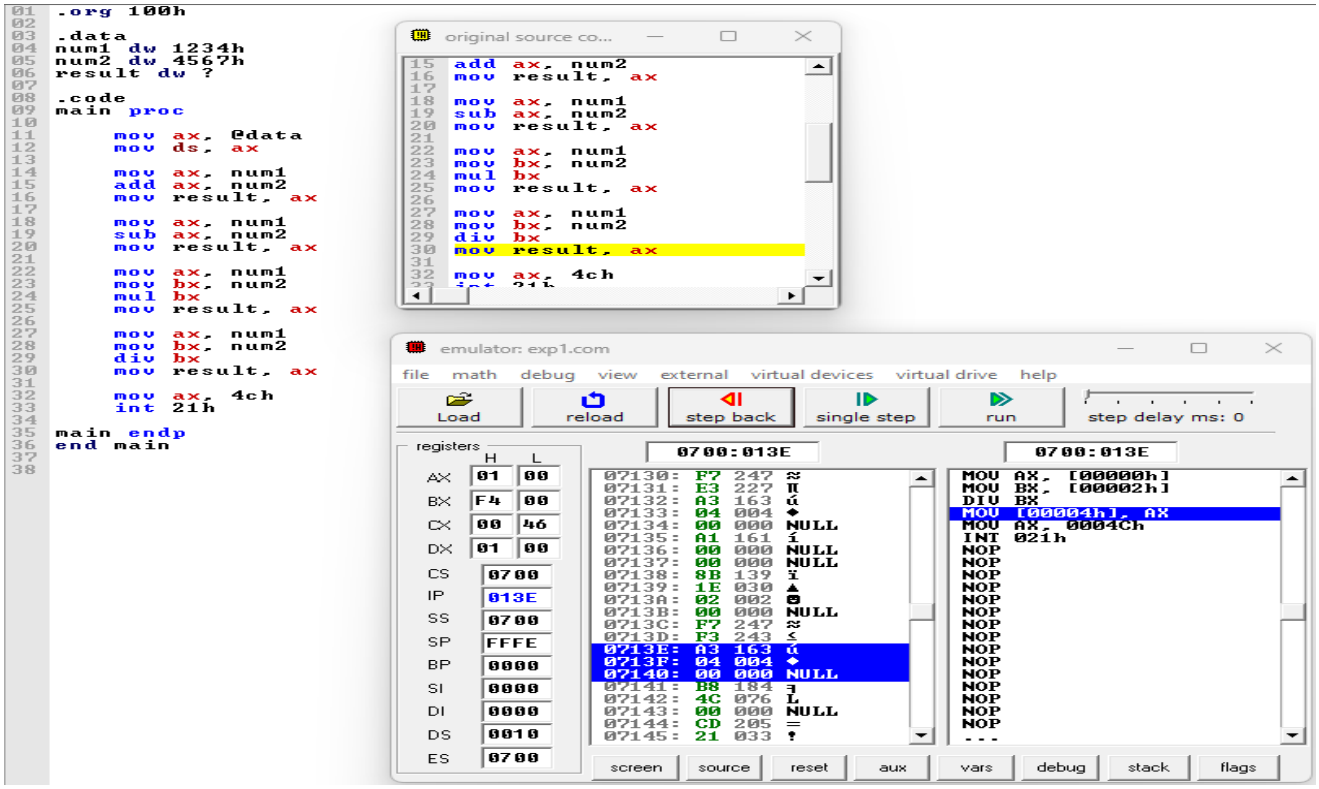
1. Load the first number in AX.
2. Load the second number. in BL
3. Divide AX by BL.
4. After division, the quotient is in AL and the remainder is in AH.

Code :

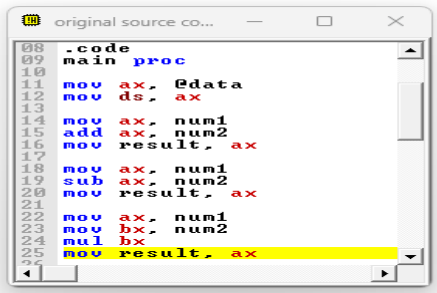
```
.org 100h

.data
num1 dw 1234h
num2 dw 4567h
result dw ?

.code
main proc

   mov ax, @data
   mov ds, ax
```

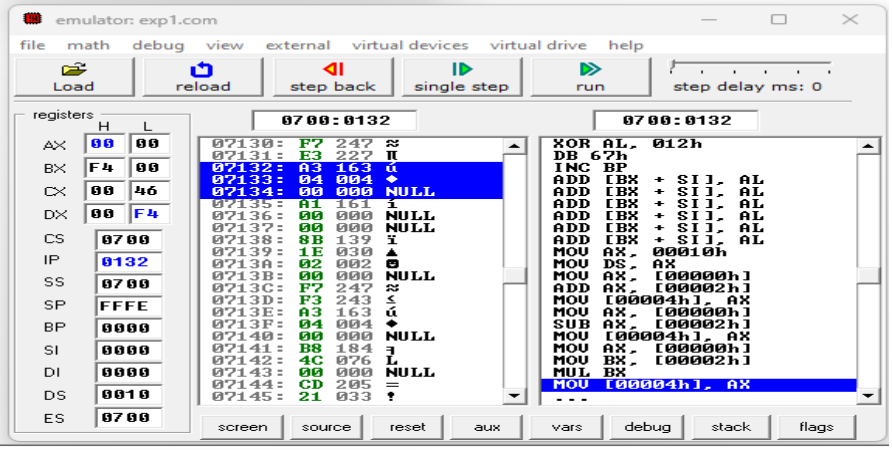
```
mov ax, num1
add ax, num2
mov result, ax

mov ax, num1
sub ax, num2
mov result, ax

mov ax, num1
mov bx, num2
mul bx
mov result, ax

mov ax, num1
mov bx, num2
div bx
mov result, ax

mov ax, 4ch
int 21h

main endp
end main
```

Output :

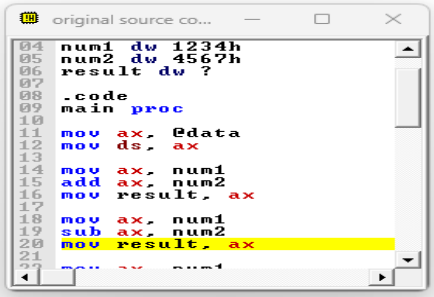


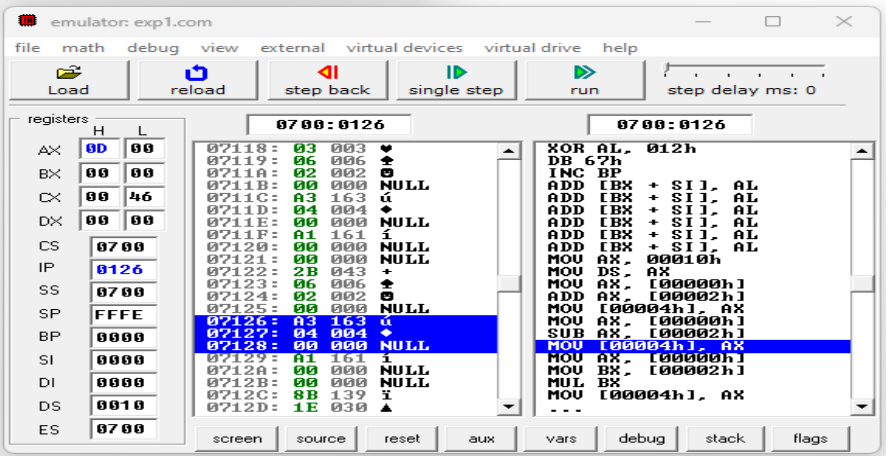
```
.org 100h

.data
num1 dw 1234h
num2 dw 4567h
result dw ?

.code
main proc

      mov ax, @data
      mov ds, ax

      mov ax, num1
      add ax, num2
      mov result, ax

      mov ax, num1
      sub ax, num2
      mov result, ax

      mov ax, num1
      mov bx, num2
      mul bx
      mov result, ax

      mov ax, num1
      mov bx, num2
      div bx
      mov result, ax

      mov ax, 4ch
      int 21h

main endp
end main
```



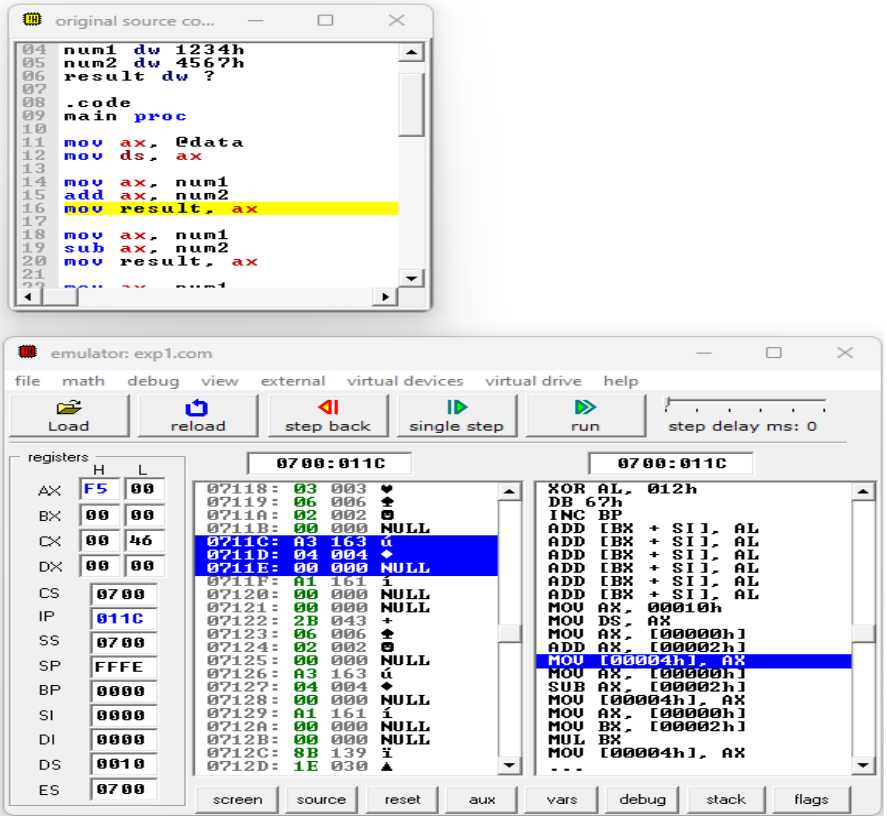



```
.org 100h

.data
num1 dw 1234h
num2 dw 4567h
result dw ?

.code
main proc

      mov ax, @data
      mov ds, ax

      mov ax, num1
      add ax, num2
      mov result, ax

      mov ax, num1
      sub ax, num2
      mov result, ax

      mov ax, num1
      mov bx, num2
      mul bx
      mov result, ax

      mov ax, num1
      mov bx, num2
      div bx
      mov result, ax

      mov ax, 4ch
      int 21h

main endp
end main
```





```
.org 100h

.data
num1 dw 1234h
num2 dw 4567h
result dw ?

.code
main proc

     mov ax, @data
     mov ds, ax

     mov ax, num1
     add ax, num2
     mov result, ax

     mov ax, num1
     sub ax, num2
     mov result, ax

     mov ax, num1
     mov bx, num2
     mul bx
     mov result, ax

     mov ax, num1
     mov bx, num2
     div bx
     mov result, ax

     mov ax, 4ch
     int 21h

main endp
end main
```



Conclusion:

In conclusion, the ability to perform basic arithmetic operations on 16-bit data is fundamental in various fields such as computer science, engineering, and mathematics. By efficiently manipulating 16-bit data, we can solve complex problems, process large datasets, and design intricate algorithms. Whether it's addition, subtraction, multiplication, or division, mastering these operations enables us to build robust systems, develop advanced technologies, and push the boundaries of innovation. As we continue to advance in the digital age, a solid understanding of arithmetic operations on 16-bit data remains an indispensable skill for professionals across diverse disciplines.