Experiment 09

Experiment 9 : 9.1 Write an ALP program to multiply 8 bit and 16 bit numbers
9.2 Write an ALP program to add 10 numbers using a loop.

Roll No.	17
Name	Manav Jawrani
Class	D10-A
Subject	Microprocessor Lab
LO Mapped	LO3: Build a program on a microprocessor using arithmetic & logical instruction set of 8086. LO4 : Develop the assembly level programming using 8086 loop instruction set.

Aim: Write an ALP program

1. Program to multiply two 8-bit numbers.
2. Program to multiply two 16-bit numbers.
3. Program to add 10 numbers using loop.

Introduction:

1. Multiplication

The MUL/IMUL Instruction

There are two instructions for multiplying binary data. The MUL (Multiply) instruction handles unsigned data and the IMUL (Integer Multiply) handles signed data. Both instructions affect the Carry and Overflow flag.

2. Loop

A **loop** is a block of statements that are repeatedly executed until a condition is satisfied.The assembly language uses **JMP** instruction to implement loops. However, the processor set can use the **LOOP** instruction to implement loops conveniently.
Syntax and explanation:
The following code snippet illustrates how a loop is implemented through JMP instruction:

```
mov AL, 5     ; store the number of iteration in AL
L1: (loop code)
DEC AL        ; decrement AL
JNZ L1
```

- The number of iterations of the loop is stored in AL.
- At the end of each iteration, the code decrements AL, then takes a conditional jump if AL is not zero.

Theory:

A. Program to multiply two 8-bit numbers.

Algorithm

Step 1 - Initialize the data segment with input numbers and memory location for the answer.

Step 2 - Start the program by loading the first data into the Accumulator.

Step 3 - Move the first number to register ‘al’ and move the second number to register ‘bl’.

Step 4 - Multiply the two register contents.

Step 5 - Store the content of ‘ax’ register to ‘c’

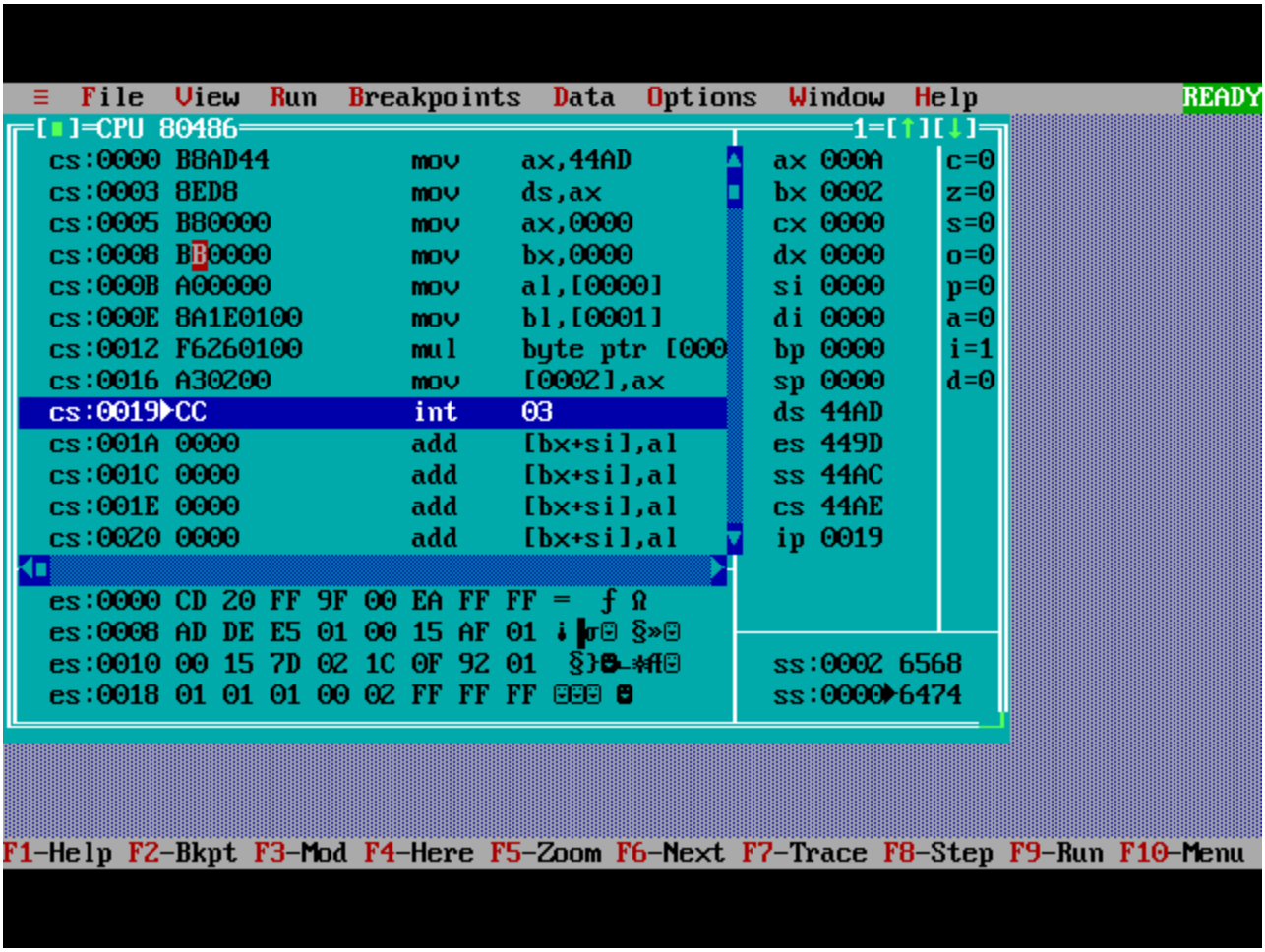
Step 6 - Terminate the program.

Code

```
data segment
a db 05h
b db 02h
c dw ?
data ends

code segment
assume cs:code, ds:data
start:
mov ax,data
mov ds,ax
mov ax,0000h
mov bx,0000h
mov al,a
mov bl,b
mul b
mov c,ax
int 3
code ends
```

end start

Input:
Num1 - 05H
Num 2 - 02H
Num1*Num2 = 10 = A
Output:



B. Program to multiply two 16-bit numbers.

Algorithm

Step 1 - Initialize the data segment with input numbers and memory location for the answer.

Step 2 - Start the program by loading the first data into the Accumulator.

Step 3 - Move the first number to register ‘ax’ and move the second number to register ‘bx’.

Step 4 - Multiply the two register contents.

Step 5 - Store the final ans in ‘ax’ and ‘dx’ registers.

Step 6 - Terminate the program.

Code

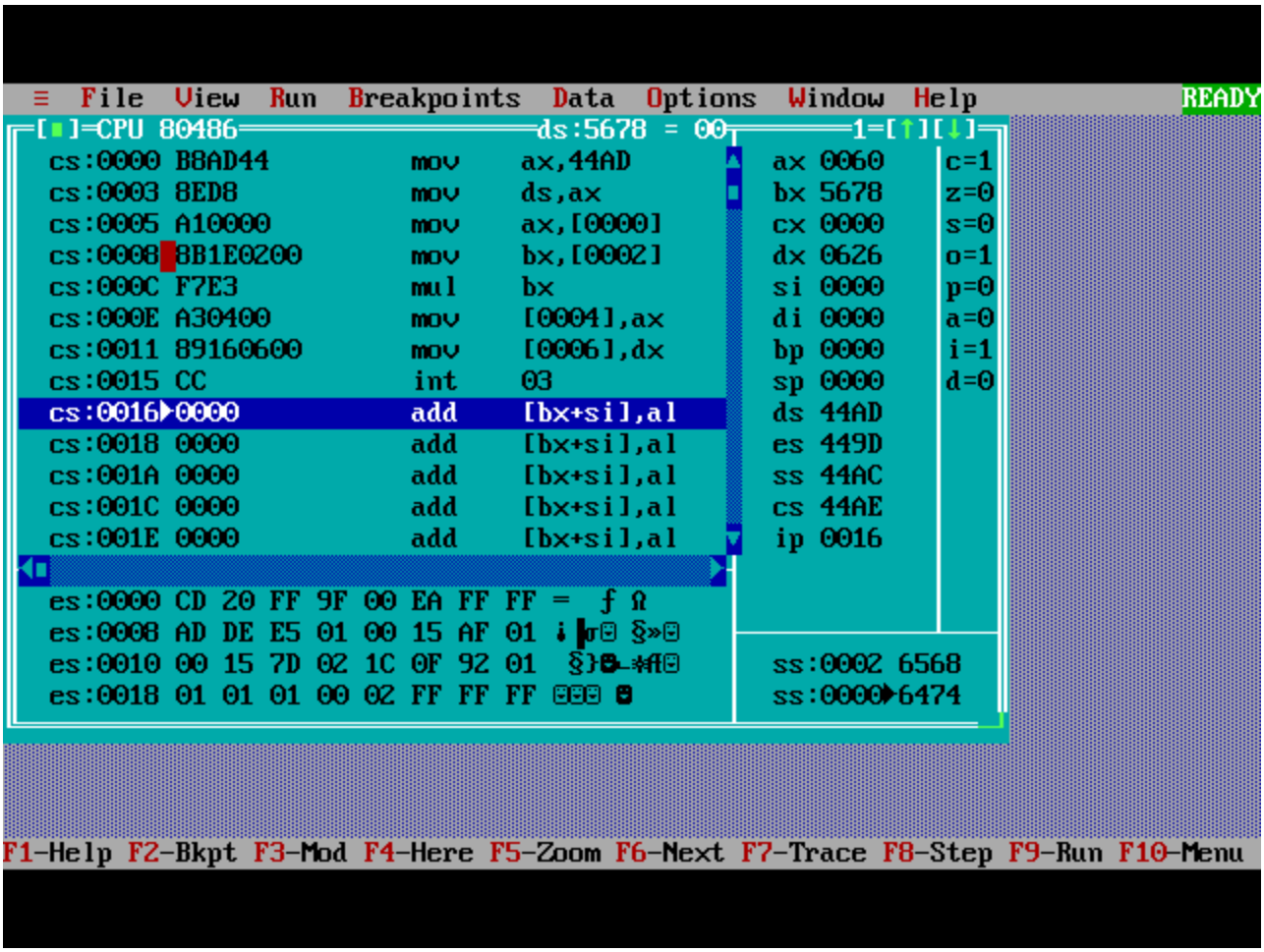
```
data segment
a dw 1234h
b dw 5678h
c dd ?
data ends

code segment
assume ds:data, cs:code
start:
mov ax,data
mov ds,ax
mov ax,a
mov bx,b
mul bx
mov word ptr c,ax
mov word ptr c+2,dx
int 3
code ends
end start
```

Input:

Num1 - 1234H

Num2 - 5678H

Output:



C. Program to add 10 numbers using loop.

Algorithm

Step 1 - Initialize the data segment and create an array of 10 numbers whose sum is to be found.

Step 2 - Create a procedure.

Step 3 - In the procedure , move content of DATA to register ‘ax’ and move content of register ‘ax’ to ‘ds’.

Step 4 - Offset the array and store in the ‘bx’ register.

Step 5 - Create a loop in the procedure , for adding the numbers in the array.

Step 6 - Increase the index of bx and decrease the index of cx.

Step 7 - Jump and repeat the steps until all the elements are added.

Step 8 - Move the contents of the ‘al’ to sum and the contents of sum to ‘dl’.

Step 9- Terminate the program

Code

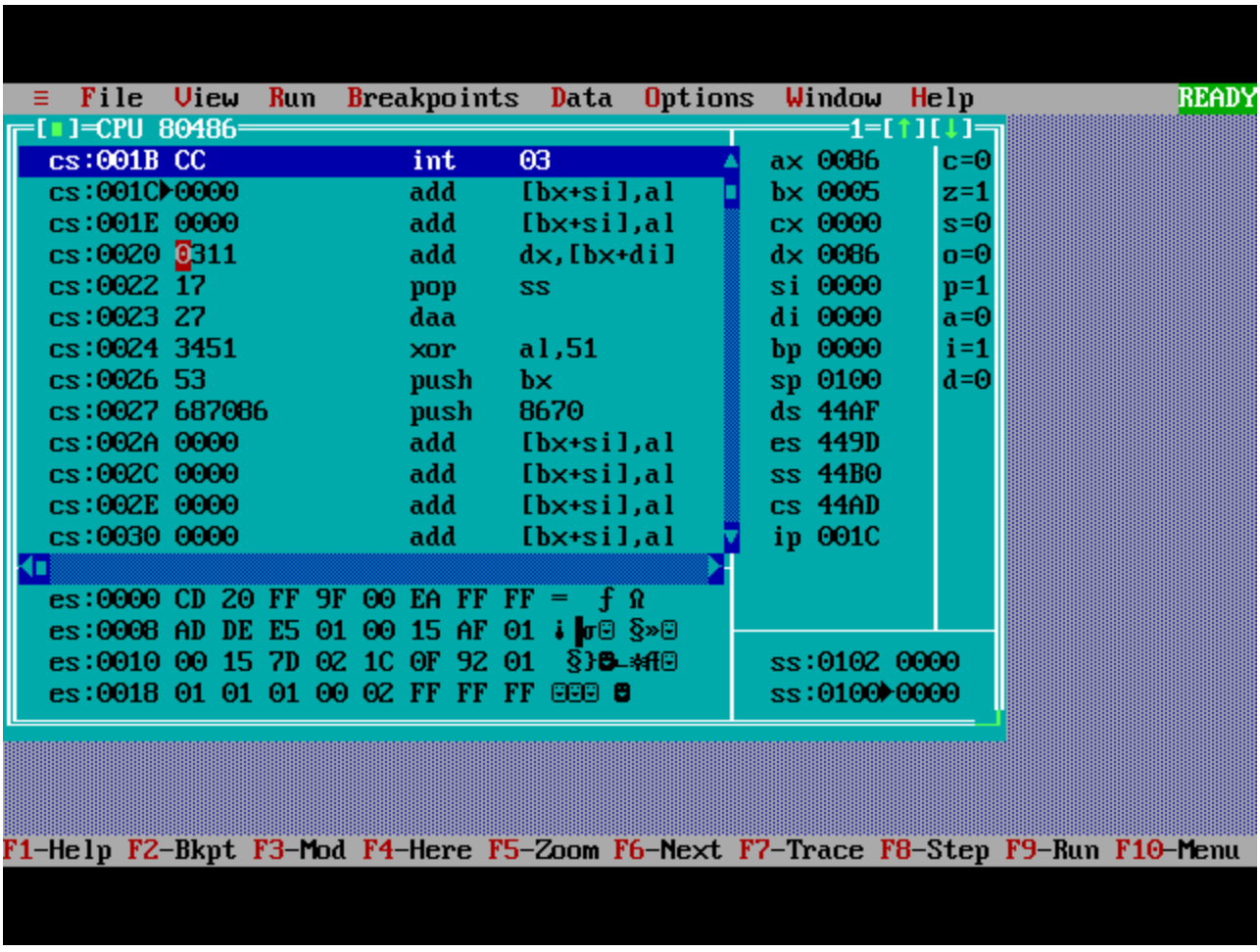
```
.MODEL small
.STACK 100h
.DATA
   ARR db 03H,11H,17H,27H,34H,51H,53H,68H,70H
   sum db 0

.CODE
main proc
   mov ax, @data
   mov ds, ax

   mov cx,5
   mov ax,0
   mov bx, offset arr

   repeat:
   add al, [bx]
```

```
    inc bx
        dec cx
    jnz repeat

        mov sum,al
        mov dl,sum
    int 3
main endp
end main
```

Input:

The array is - 03H,11H,17H,27H,34H,51H,53H,68H,70H

Output:



Conclusion:

Thus , we have performed the aim of the given experiment successfully.