

## 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:

```

≡ File View Run Breakpoints Data Options Window Help
[ ]=CPU 80486 1=[↑][↓]
cs:0000 B8AD44 mov ax,44AD ax 000A c=0
cs:0003 8ED8 mov ds,ax bx 0002 z=0
cs:0005 B80000 mov ax,0000 cx 0000 s=0
cs:0008 B80000 mov bx,0000 dx 0000 o=0
cs:000B A00000 mov al,[0000] si 0000 p=0
cs:000E 8A1E0100 mov bl,[0001] di 0000 a=0
cs:0012 F6260100 mul byte ptr [000] bp 0000 i=1
cs:0016 A30200 mov [0002],ax sp 0000 d=0
cs:0019 CC int 03 ds 44AD
cs:001A 0000 add [bx+si],al es 449D
cs:001C 0000 add [bx+si],al ss 44AC
cs:001E 0000 add [bx+si],al cs 44AE
cs:0020 0000 add [bx+si],al ip 0019
es:0000 CD 20 FF 9F 00 EA FF FF = f Ω
es:0008 AD DE E5 01 00 15 AF 01 ; 0 8 >
es:0010 00 15 7D 02 1C 0F 92 01 8 0 1
es:0018 01 01 01 00 02 FF FF FF 0 0 0
ss:0002 6568
ss:0000 6474
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```

**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:

File View Run Breakpoints Data Options Window Help

[ ]=CPU 80486 ds:5678 = 00 1=[↑][↓]

cs:0000	B8AD44	mov	ax,44AD	ax	0060	c=1
cs:0003	8ED8	mov	ds,ax	bx	5678	z=0
cs:0005	A10000	mov	ax,[0000]	cx	0000	s=0
cs:0008	8B1E0200	mov	bx,[0002]	dx	0626	o=1
cs:000C	F7E3	mul	bx	si	0000	p=0
cs:000E	A30400	mov	[0004],ax	di	0000	a=0
cs:0011	89160600	mov	[0006],dx	bp	0000	i=1
cs:0015	CC	int	03	sp	0000	d=0
cs:0016	0000	add	[bx+si],al	ds	44AD	
cs:0018	0000	add	[bx+si],al	es	449D	
cs:001A	0000	add	[bx+si],al	ss	44AC	
cs:001C	0000	add	[bx+si],al	cs	44AE	
cs:001E	0000	add	[bx+si],al	ip	0016	

es:0000 CD 20 FF 9F 00 EA FF FF = f Ω

es:0008 AD DE E5 01 00 15 AF 01 ; 0 0 0 0 0 0 0 0

es:0010 00 15 7D 02 1C 0F 92 01 0 0 0 0 0 0 0 0

es:0018 01 01 01 00 02 FF FF FF 0 0 0 0 0 0 0 0

ss:0002 6568

ss:0000 6474

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

### **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

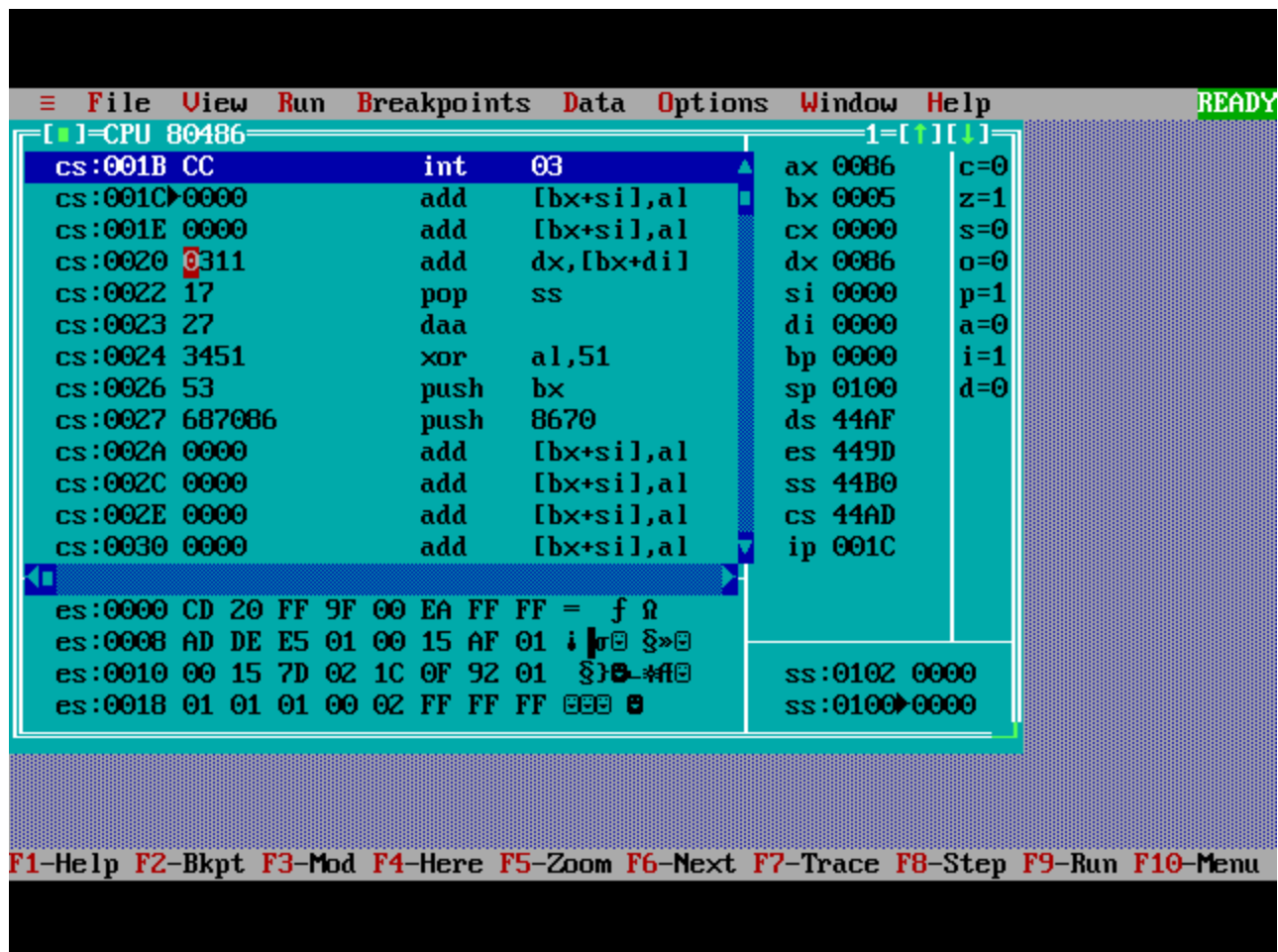
    mov sum,al
    mov dl,sum
    int 3
main endp
end main

```

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.