Experiment No 2

Aim : 8086 Assembly language programming for Addition and subtraction of two 16 bit numbers

Requirement : Emu8086 (Assembler and Microprocessor emulator)

Theory :

1. Data transfer instructions

Data transfer instructions are used to transfer data from source operand to destination operand

MOV instruction

MOV instruction moves data from one location to another. It also has the widest variety of parameters; so it the assembler programmer can use MOV effectively, the rest of the commands are easier to understand.

Format: `MOV destination,source`
Logically, destination = destination + source

Eg. MOV AX,6

2. Arithmetic instructions

a. ADD instruction

ADD adds the contents of the source to the destination. The source and destination may be either bytes or words but both operands must be the same type or the assembler will generate an error.

If the sum of the two numbers cannot fit in the destination, an extra bit is required and this is signalled by the ADD operation setting the carry flags (CF) to 1. If the sum fits without spillage, CF=0. Other registers can be affected by addition operations as well; ZF=0 if the sum is zero, SF=1 if the sum is negative, etc. The logic of the basic addition command is:

Format: **ADD** `destination,source`

Logically, destination = destination + source

Eg. ADD AX,BX

b. SUB instruction

SUB subtracts the source value from the destination. Operation is almost identical to addition, except that the CF flag is used as a borrow in the case of the SBB (subtract with borrow) instruction.

Format: **SUB** `destination,source`

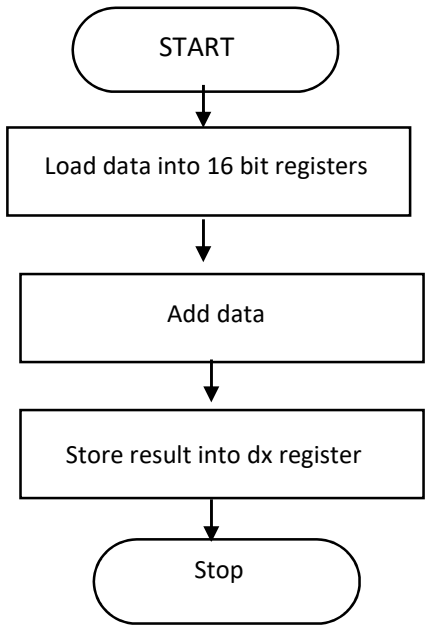
Logically, it is destination = destination - source

destination = destination - source - carry (if required)

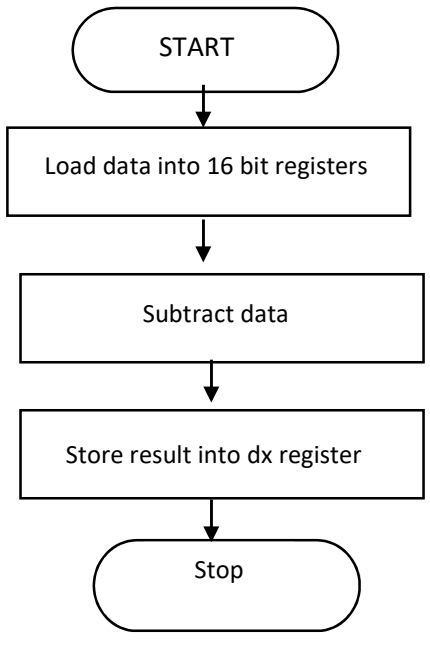
Eg. SUB AX,BX

Flowchart :

Flowchart for 16 bit numbers addition



Flowchart for 16 bit numbers subtraction



16 Bit Addition :

Program:

```
data segment
   a dw 4121h
   b dw 1742h
   c dw ?
data ends
code segment
    assume cs:code,ds:data
    start:
    mov ax,data
    mov ds,ax
    mov ax,a
    mov bx,b
    add ax,bx
    mov cx,ax
    mov c,cx
    int 3
    code ends
```

end start

MEMORY LOCATION	OP-CODE	LABEL	MNEMOIC
O7110h	**B8**		**mov ax,data**
07113H	8E		**mov ds,ax**
07115H	A1		**mov ax,a**
07118H	8B		**mov bx,b**
0711CH	03		**add ax,bx**
0711EH	8B		**mov cx,ax**
07120H	89		**mov c,cx**
07124H	CC		**int 3**

16 Bit Subtraction :

Program:

```
data segment
a dw 21A6h
b dw 1022h
c dw ?
data ends
code segment
   assume cs:code,ds:data
   start:
   mov ax,data
   mov ds,ax
   mov ax,a
   mov bx,b
   sub ax,bx
   mov cx,ax
   mov c,cx
   int 3
   code ends
end start
```

MEMORY LOCATION	OP-CODE	LABEL	MNEMOIC
7110H	**B8**		**mov ax,data**
07113H			**mov ds,ax**
07115H	A1		**mov ax,a**
07118H	8B		**mov bx,b**
0711CH	2B		**sub ax,bx**
0711EH	8B		**mov cx,ax**
07120H	89		**mov c,cx**
07124H	CC		**int 3**

Result :

Output OF 16 BIT ADDITION :

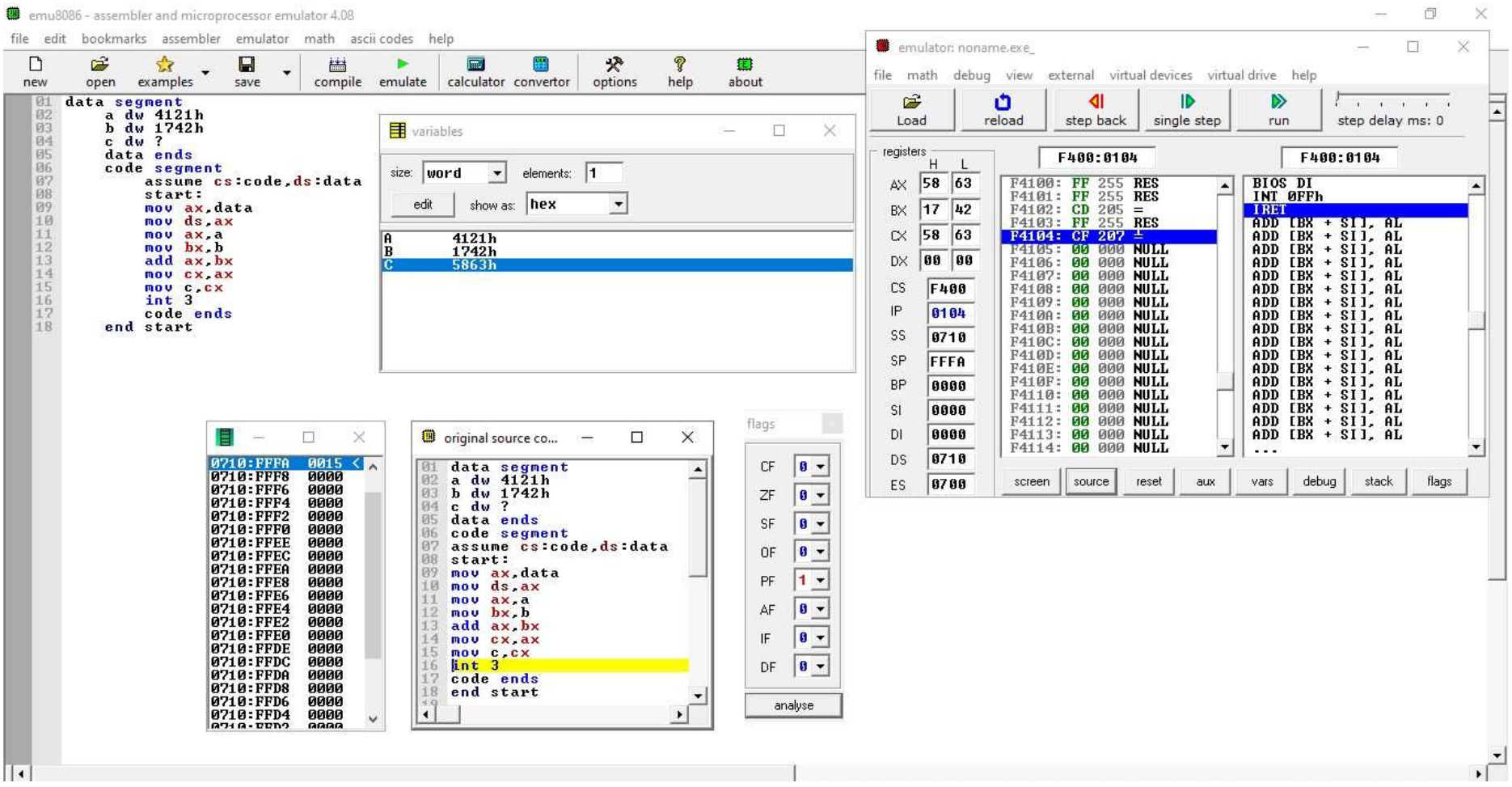
INPUT		OUTPUT	
REGISTER	DATA	REGISTER	DATA
AX	4121	AX	5863
BX	1742		

Output OF 16 BIT SUNTRACTION :

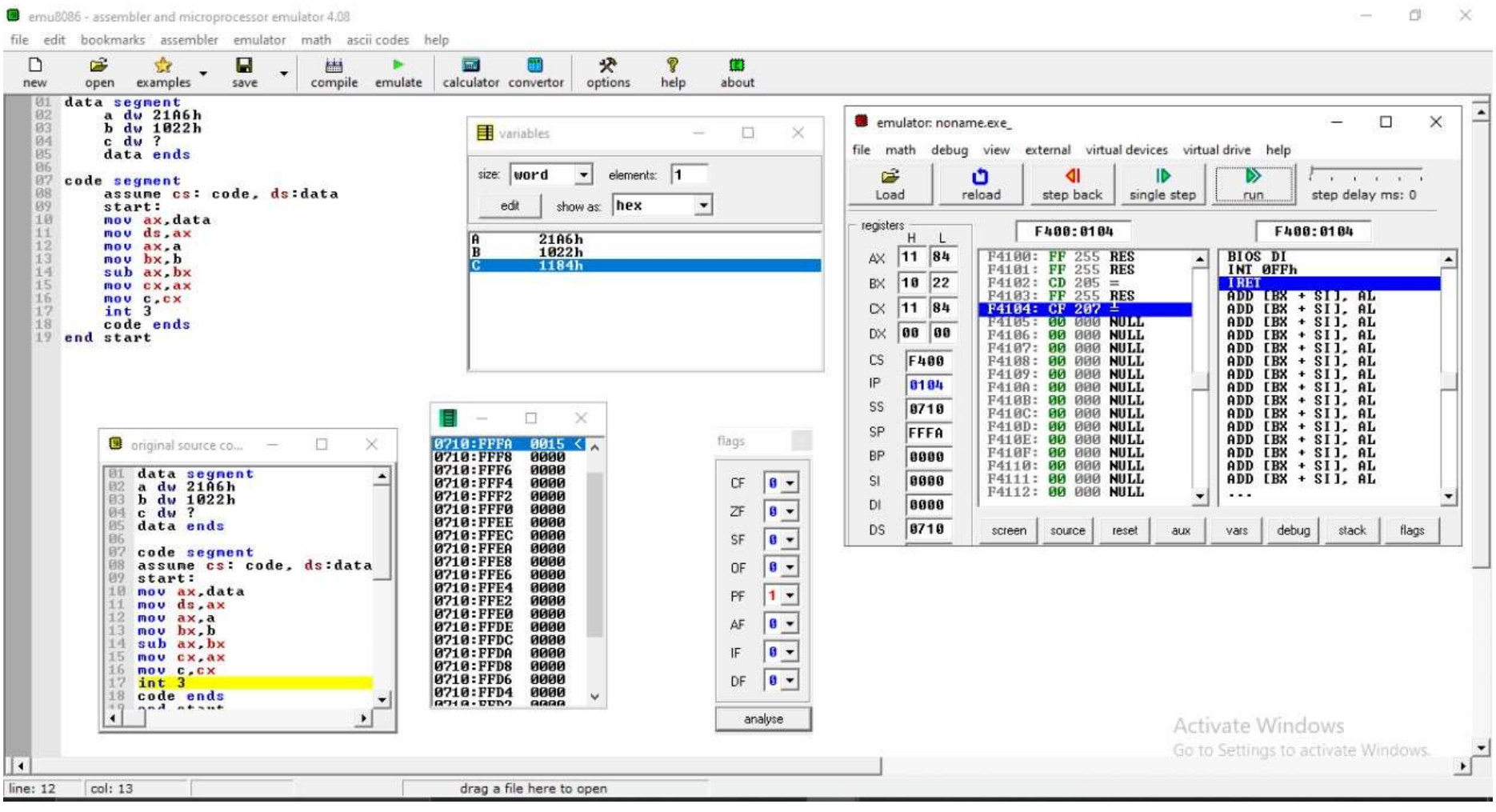
INPUT		OUTPUT	
REGISTER	DATA	REGISTER	DATA
AX	21A6	AX	1184
BX	1022		

SCREENSHOT :

16 Bit Addition :



16 bit subtraction :



CONCLUSION :

Thus ,we performed 16 bit addition and 16 bit subtraction using AX ,BX and CX register. We used MOV , ADD and SUB instructions during the procedure.

Experiment No 3

Aim : 8086 Assembly language programming for Addition of series of 16 bit numbers

Requirement : Emu8086 (Assembler and Microprocessor emulator)

Theory :

1. Data transfer instructions

Data transfer instructions are used to transfer data from source operand to destination operand

a. MOV instruction

MOV instruction moves data from one location to another. It also has the widest variety of parameters; so it the assembler programmer can use MOV effectively, the rest of the commands are easier to understand.

Format: `MOV destination,source`
Logically, destination = destination + source

Eg. MOV AX,6

b. LEA instruction

LEA instruction determines the offset of the variable or memory location named as the source and puts this offset in the indicated 16-bit register. LEA does not affect any flag.

Syntax : LEA Register, Source

Example : LEA AX,14

2. Arithmetic instructions

a. ADD instruction

ADD adds the contents of the source to the destination. The source and destination may be either bytes or words but both operands must be the same type or the assembler will generate an error.

If the sum of the two numbers cannot fit in the destination, an extra bit is required and this is signalled by the ADD operation setting the carry flags (CF) to 1. If the sum fits without spillage, CF=0. Other registers can be affected by addition operations as well; ZF=0 if the sum is zero, SF=1 if the sum is negative, etc. The logic of the basic addition command is:

Format: **ADD** `destination,source`

Logically, destination = destination + source

Eg. ADD AX,BX

b. DEC instruction

DEC instruction subtracts 1 from the destination word or byte. The destination can be a register or a memory location. AF, OF, SF, PF, and ZF are updated, but CF is not affected. This means that if an 8-bit destination containing 00H or a 16-bit destination containing 0000H is decremented, the result will be FFH or FFFFH with no carry (borrow).

Syntax : DEC Destination

Example : DEC AX

3. Logical instructions

CMP instruction

This instruction comes under **Logical Instruction.** This instruction compares a byte / word in the specified source with a byte / word in the specified destination. The source can be an immediate number, a register, or a memory location. The destination
can be a register or a memory location. However, the source and the destination cannot both be memory

locations. The comparison is actually done by subtracting the source byte or word from the destination byte or word. The source and the destination are not changed, but the flags are set to indicate the results of the comparison. AF, OF, SF, ZF, PF, and CF are updated by the CMP instruction. For the instruction CMP CX, BX, the values of CF, ZF, and SF will be as follows:

Syntax : CMP Destination, Source

Example : CMP AL, 01H

4. Transfer Of Control Instruction

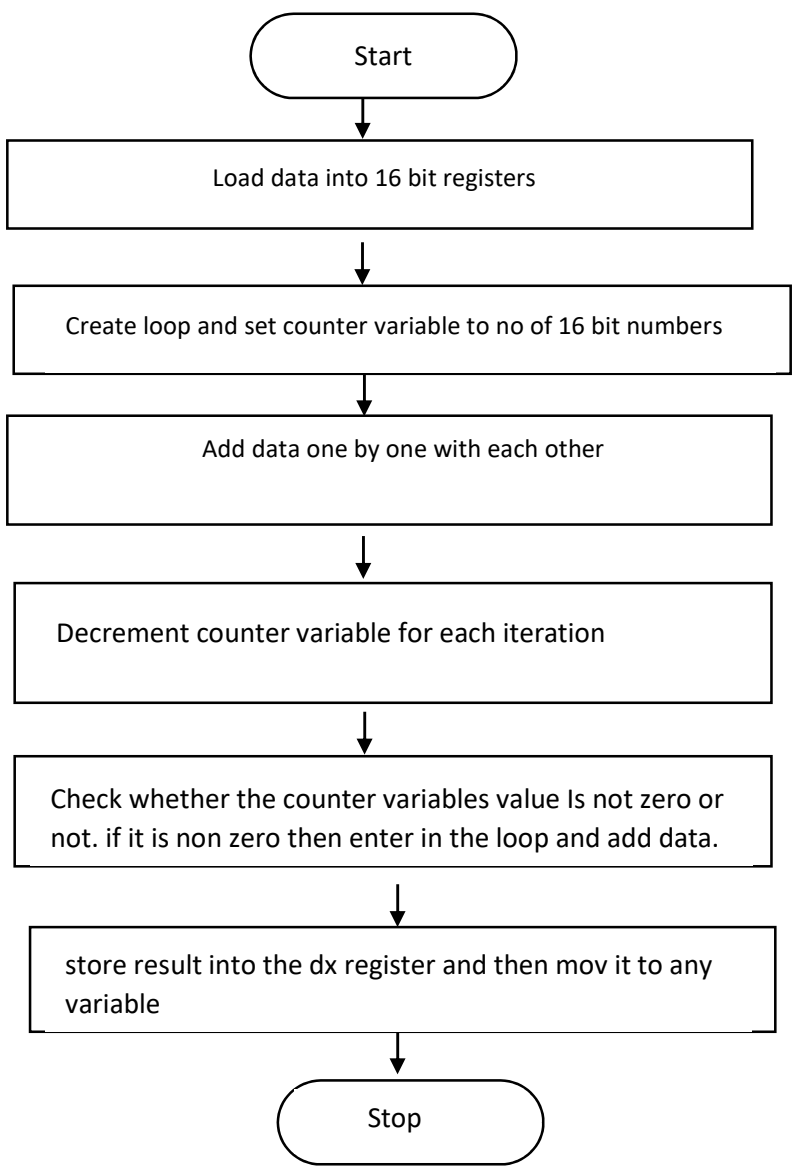
JNZ instruction

This instruction comes under **Transfer Of Control Instruction.** This instruction is usually used after a Compare instruction. If the zero flag is 0, then this instruction will cause a jump to the label given in the instruction.

Example :

```
ADD AX, 0002H
DEC BX
JNZ NEXT
```

Flowchart :



Program :

```
data segment
   a dw 4200h, 5300h, 1600h, 8000h, 1900h
   b dw ?
   data ends
code segment
      assume cs:code,ds:data
      start:
      mov ax,data
      mov ds,ax
      mov cl,5
      lea bx,a
      mov ax,00
      OP: add ax,word ptr[bx]
         add bx,02
         dec cl
         cmp cl,00
         jnz OP

      mov b,ax
      int 3
      code ends
end start
```

MEMORY LOCATION	OP-CODE	LABEL	MNEMOIC
7110h	**B8**		**mov ax,data**
7113h	**8E**		**mov ds,ax**
7115h	**B1**		**mov cl,5**
7117h	**BB**		**lea bx,a**
711Ah	**B8**		**mov ax,00**
711Dh	**03**		**OP: add ax,word ptr[bx]**
711Fh	**83**		**add bx,02**
7122h	**FE**		**dec cl**
7124h	**80**		**cmp cl,00**
7127h	**75**		**jnz OP**
7129h	**A3**		**mov b,ax**
712Ch	**CC**		**int 3**

Result :

Output OF 16 BIT ADDITION :

INPUT		OUTPUT	
REGISTER	**DATA**	**REGISTER**	**DATA**
AX	4200h, 5300h, 1600h, 8000h, 1900h	**AX**	**4400h**

SCREENSHOT :