# LAB 07: Processor Status & Flag Registers

## **Objectives:**

Introduction to Flags Register

## **Pre-Lab**

### **Introduction to Flags Register**

The circuits in the CPU can perform simple decision making based on the current state of the processor. For the 8086 processor, processor state is implemented as nine individuals; bits called flags. Each decision made by the 8086 is based on the values of these flags. The flags are based in the Flags register and they are classified as either status flags or control flags. The status flags reflect the result of a computation. The control flags are used to enable or disable certain operations of the processor.

### **The Status Flags**

Figure 5.1 below shows the Flags register. The status flags are located in bits 0, 2,4,6,7 and 11 and the control flags are located in the bits 8, 9 and 10. The other bits have no significance.

**Figure 5.1: The Flags Register**

### **Carry Flag (CF)**

CF = 1 if there is a carry out from the Most Significant Bit (MSB) on addition, or there is a borrow into the MSB on subtraction; otherwise it is 0.

### **Parity Flag (PF)**

PF = 1 if the low byte of a result has an even number of one bit (even parity). It is 0 if the low byte has odd parity. For example, if the result of a word addition is FFFEH, then the low byte contains 7 one bits, so PF = 0.

### **Auxiliary Cary Flag (AF)**

AF = 1 if there is a carry out from bit 3 on addition, or a borrow into bit 3 on subtraction.

### **Zero Flag (ZF)**

ZF = 1 for a zero result, and ZF = 0 for a nonzero result.

### **Sign Flag (SF)**

SF = 1 if the MSB of a result is 1; it means the result is negative if you are giving a signed interpretation. SF = 0 if the MSB is 0.

### **Overflow Flag (OF)**

OF = 1 if signed overflow occurred, otherwise it is 0.

### **How the Processor determines the Overflow Occurred**

Many instructions can cause overflow; for simplicity, we’ll limit the discussion to addition and subtraction.

### **Unsigned Overflow**

On addition, unsigned overflow occurs when there is a carry out of the MSB. This means that the correct answer is larger than the biggest unsigned number; that is, FFFFH for a word and FFH for a byte. On subtraction, unsigned overflow occurs when there is a borrow into the MSB.

### **Signed Overflow**

On addition of numbers with the same sign, signed overflow occurs when the sum has a different sign.

Subtraction of numbers with different signs is like adding number of the same sign. For example, A - (-B) = A + B and –A – (+B) = -A + -B. Signed overflow occurs if the result has a different sign than expected.

### **Instructions Affect the Flags**

The instructions that affect the flags as follows:

|  |  |
| --- | --- |
| **INSTRUCTION** | **AFFECTS FLAGS** |
| **MOV/XCHG** | None |
| **ADD/SUB** | All |
| **INC/DEC** | All except CF |
| **NEG** | All (CF = 1 unless result is 0,  OF = 1 if word operand is 8000H,  Or byte operand is 80H) |

## **In-Lab Task:**

**1. Add AX, BX where AX contains FFFFH, BX contains FFFFH.**

Solution: FFFFH + FFFFH = 1 FFFEH

The result stored in AX is FFFEH = 1111 1111 1111 1110

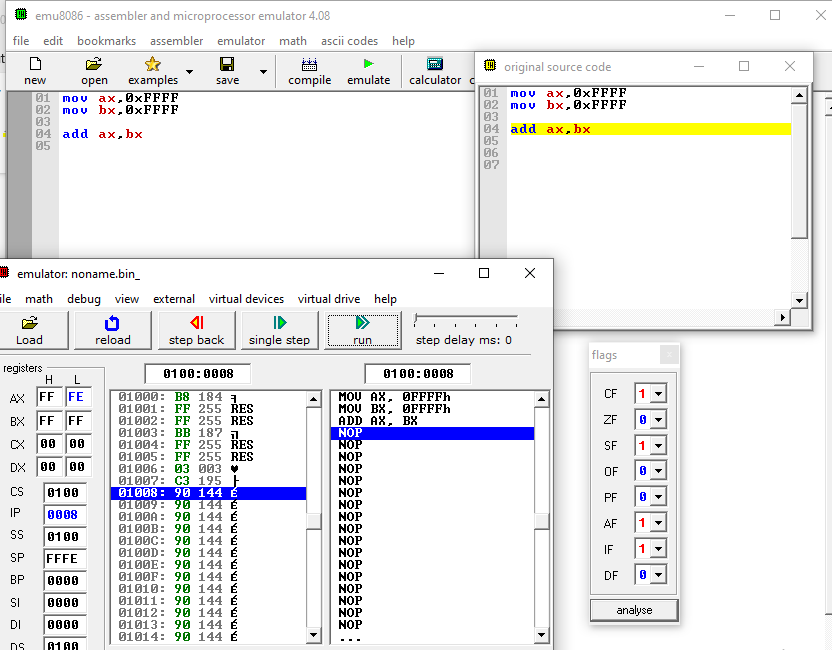
**SF** = 1 because the MSB is 1.

**PF** = 0 because there are 7 (odd number) of 1 Bits in the low byte of the result.

**ZF** = 0 because the result is nonzero

**CF** = 1 because there is a carry out of the MSB on addition.

**OF** = 0 because the sign of the stored result is the same as that of the numbers being added.



**2. ADD AL, BL where AL contains 80H, BL contains 80H.**

Solution: **80H + 80H = 100H**

The result stored in AL is 00H.

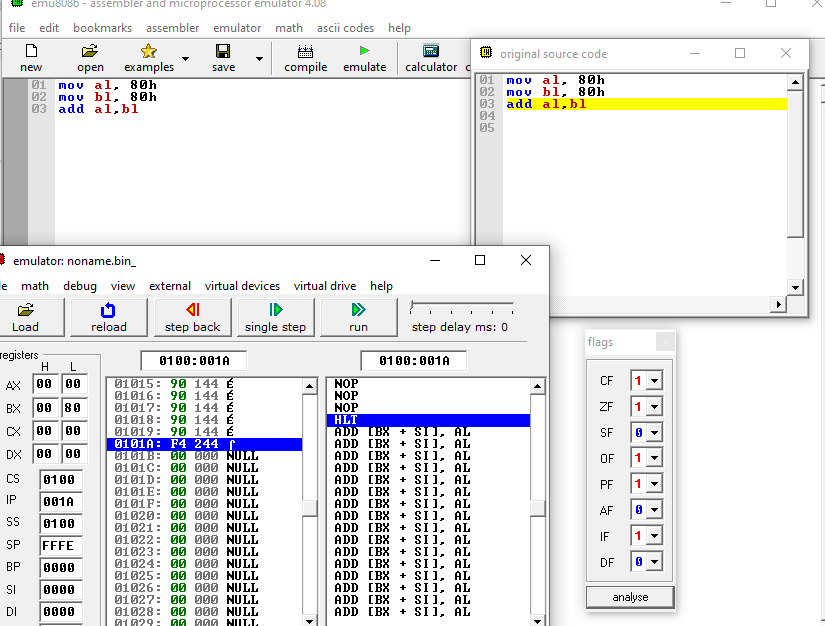
**SF** = 0 because the MSB is 0.

**PF** = 1 because all the bits in the result are 0

**ZF** = 1 because the result is 0

**CF** = 1 because there is a carry out of the MSB on addition

**OF** = 1 because the numbers being added are both negative



**3. SUB AX, BX where AX contains 8000H and BX contains 0001H.**

Solution: 8000H – 0001H

The result stored in AX = 7FFFH

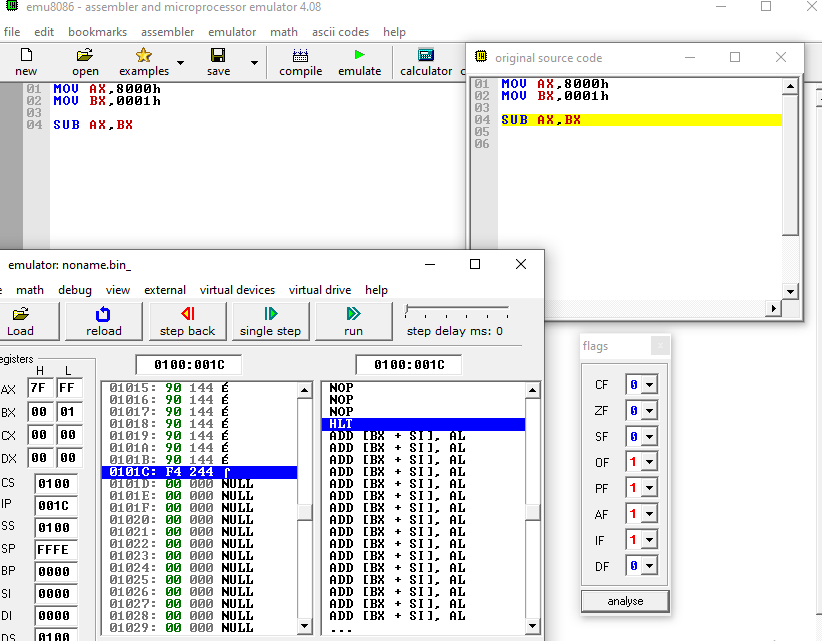
**SF** = 0 because the MSB is 0.

**PF** = 1 Because there are 8 (even number) one bits in the low byte of the result.

**ZF** = 0 Because the result is non-zero.

**CF** = 0 Because smaller unsigned number is being subtracted from a larger one.

**OF** = 1 In a signed sense, we are subtracting a positive number from a negative one, which is like adding two negatives. Because the result is positive (the wrong sign).

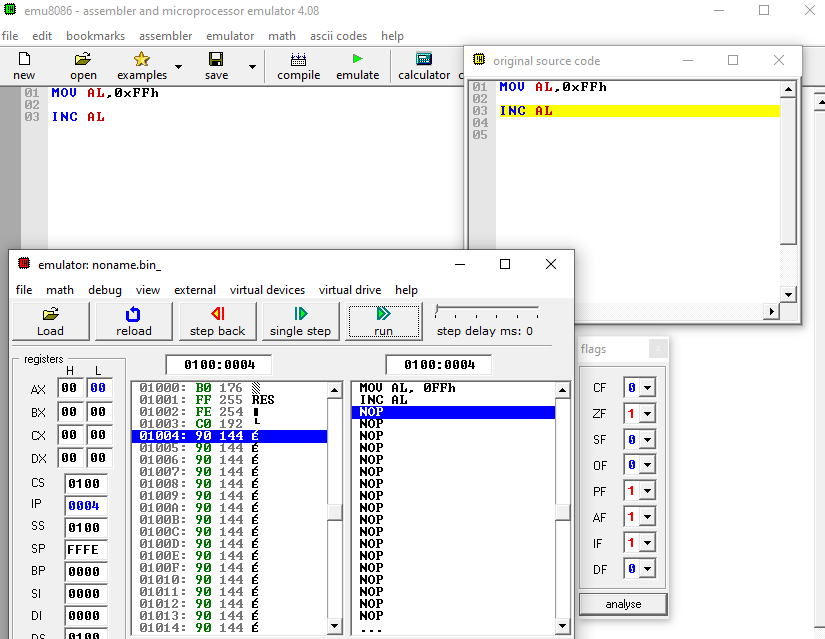


**4. INC AL, where AL contains FFH**

Solution: FFH + 1H = 1 00H

**SF = 0, PF = 1, ZF = 1**, Even though there is a carry out, CF is unaffected by INC.

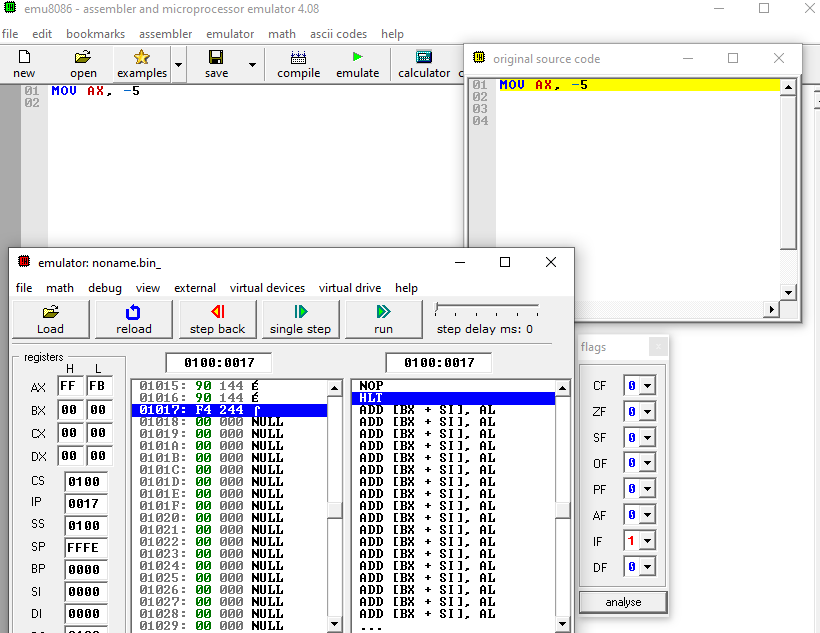
OF = 0 because numbers of unlike sign are being added.



**5. Mov AX, -5**

Solution: The result stored in AX is -5 = FFFBH

None of the flags are affected by MOV.

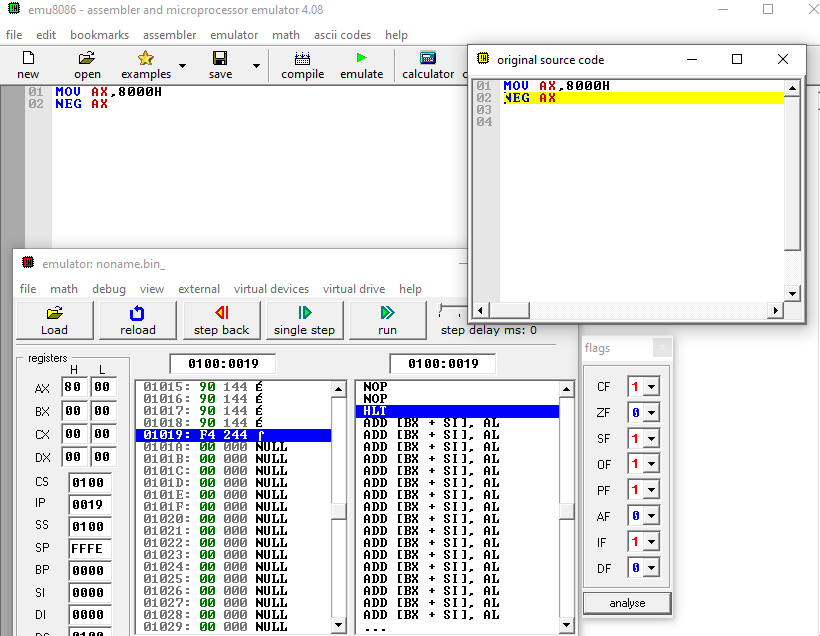


**6. NEG AX, where AX contains 8000H**

Solution: The result stored in AX is 8000H.

**SF = 1, PF = 1, ZF = 0, CF = 1** because for NEG CF is always 1 unless the result is 0.

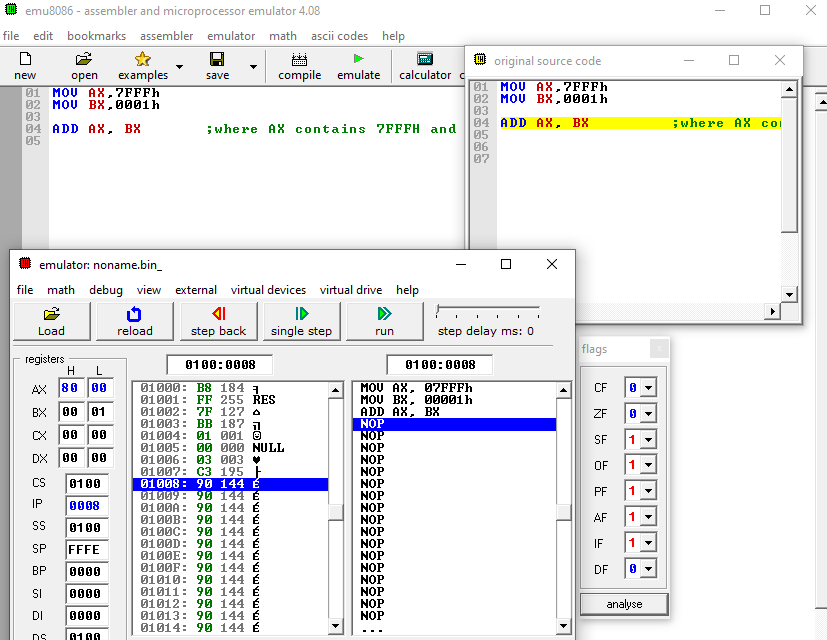
OF = 1 because the result is 8000H, when a number is negated we would expect the sign change but because 8000h is its own two’s complement, there is no sign change.



**Exercise 2:**

1. For each of the following instructions, give the new destination contents and new settings of CF, SF, ZF, PF, OF. Suppose flags are zero initially in each part of this question.

**ADD AX, BX** where AX contains 7FFFH and BX contains 0001H

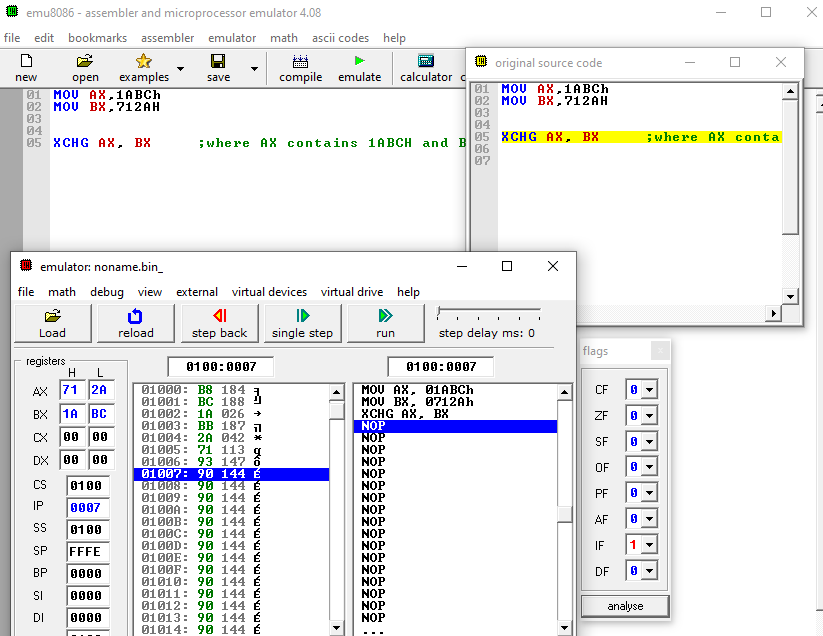


**DEC AL** where AL contains 00H

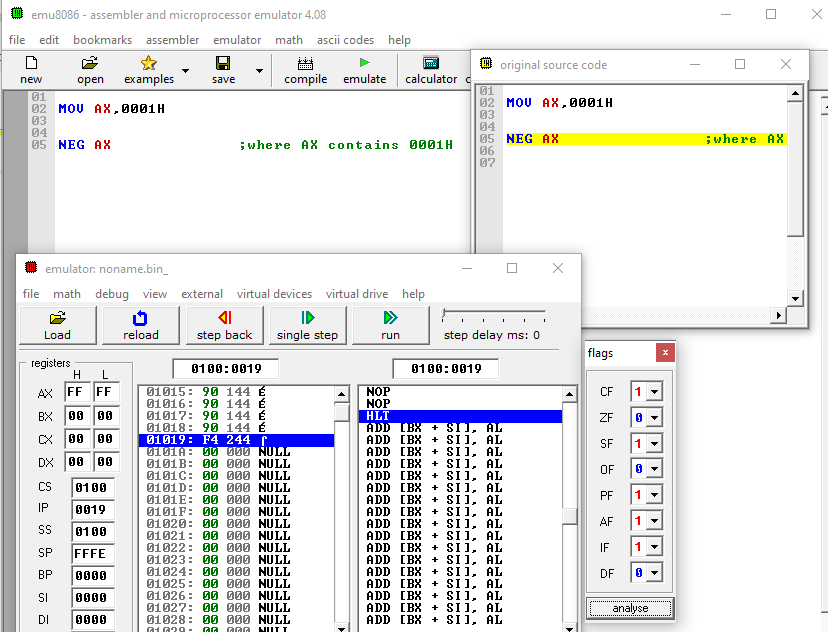
A screenshot of a computer

Description automatically generated

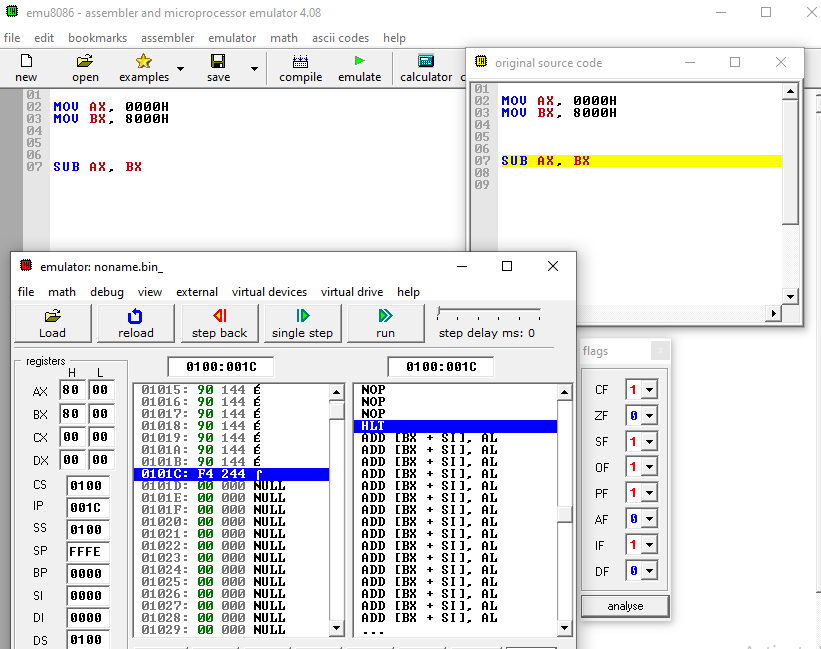
**XCHG AX, BX** where AX contains 1ABCH and BX contains 712AH



**NEG AX** where AX contains 0001H



**SUB AX, BX** where AX contains 0000H and BX contains 8000H

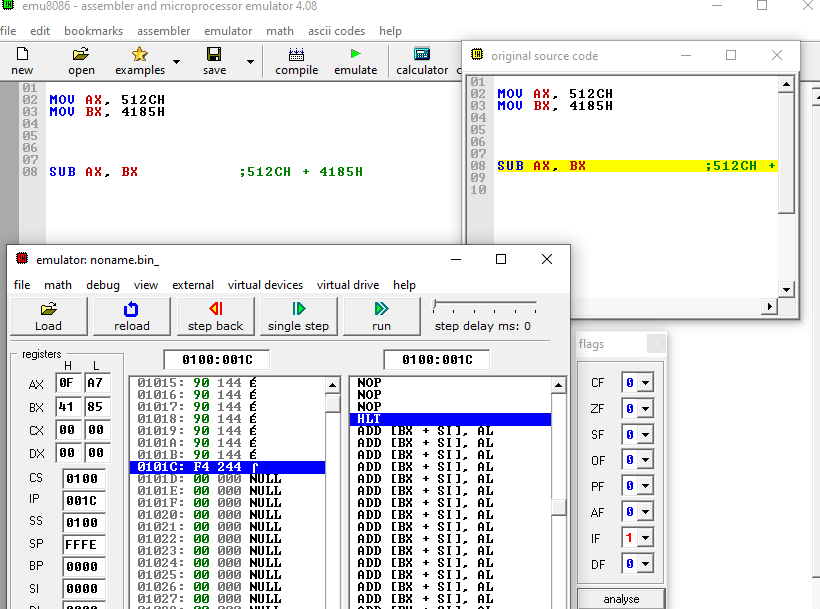


## **Post-Lab**

1. Suppose ADD AX, BX is executed. In each of the following parts, the first number being added is the contents of AX and the second numbers is contents of BX. Give the resulting value of AX and tell whether signed or unsigned overflow occurred.

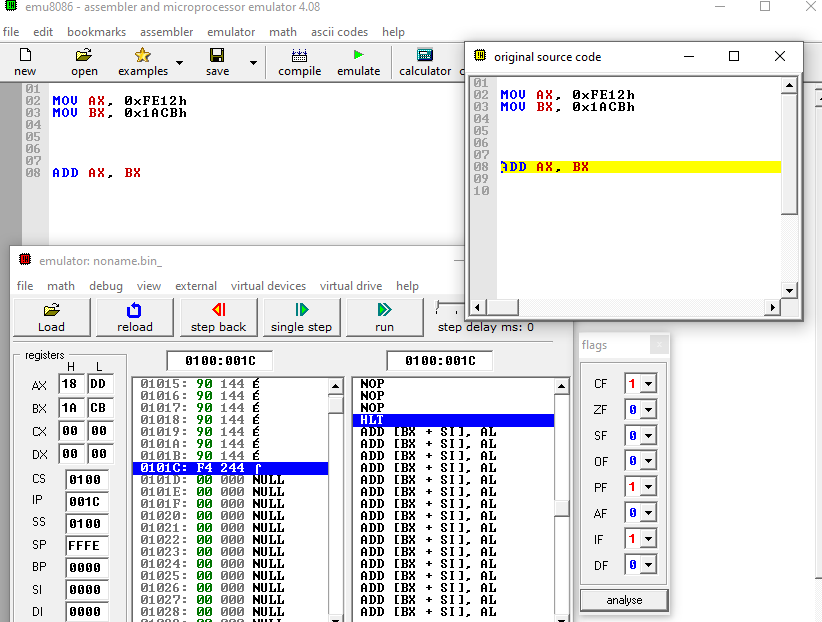
**512CH + 4185H**

**There was no signed or unsigned overflow**

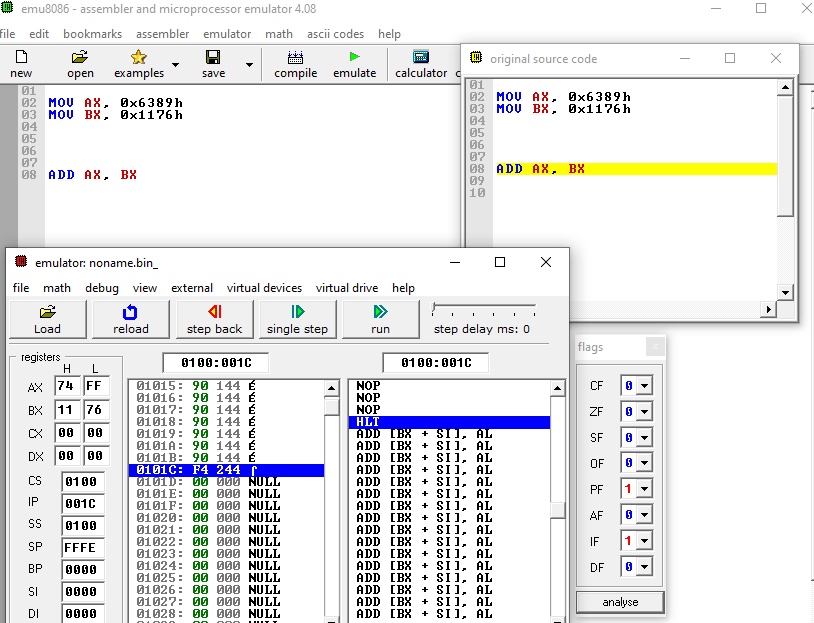
****

**FE12H + 1ACBH**

**Unsigned Overflow Occurred**

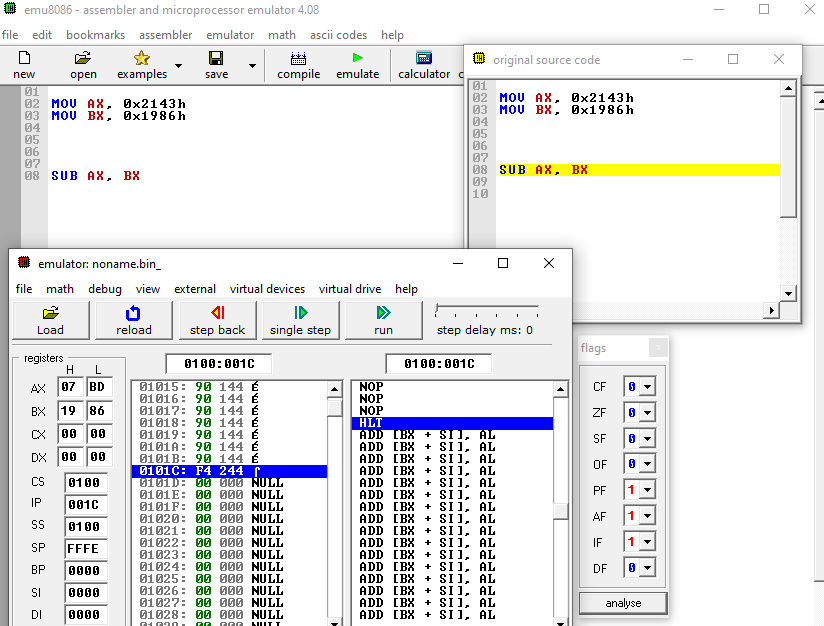
****

**6389H + 1176H**

****

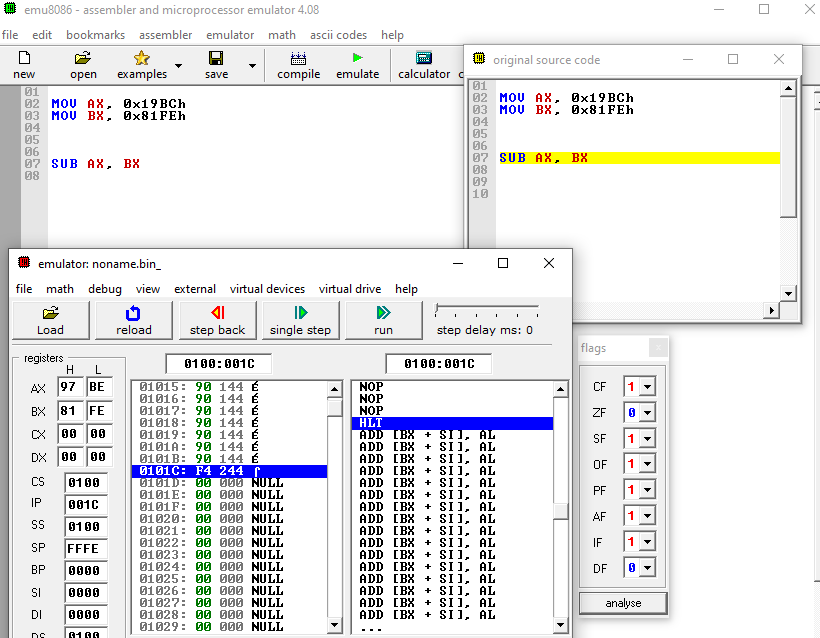
2. Suppose ADD AX, BX is executed. In each of the following parts, the first number being added is the contents of AX and the second numbers is contents of BX. Give the resulting value of AX and tell whether signed or unsigned overflow occurred.

**2143H – 1986H**

****

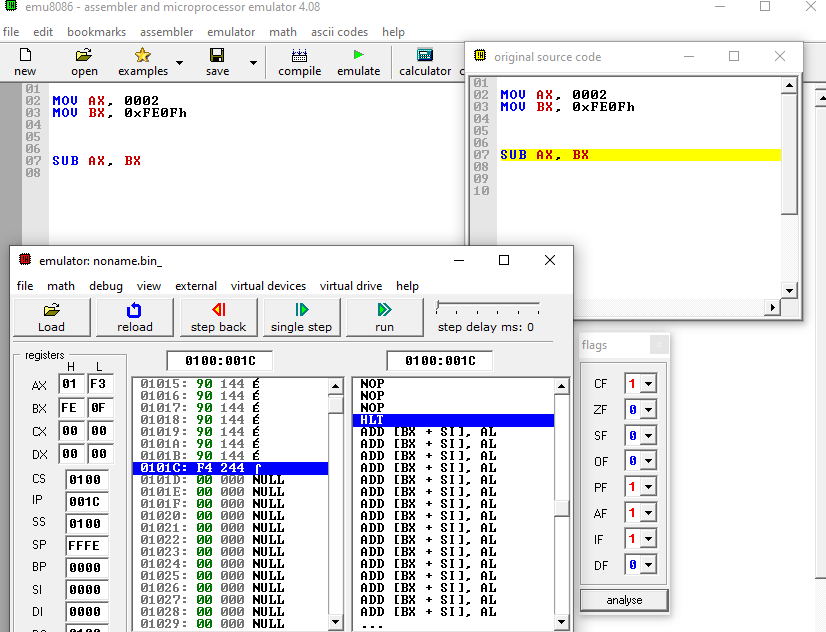
**19BCH – 81FEH**

**Signed Overflow Occurred**

****

**0002 + FE0FH**

**Unsigned Overflow Occurred**

****

## **Review Questions**

State the name of flags placed in flags register:

Carry, Parity, Auxiliary, Zero, Sign, Trap, Interrupt, Direction, Overflow.

Is it possible to get signed overflow on additions of two numbers with different sign?:

No, it's not possible to get signed overflow on additions of two numbers with different signs.

For unsigned and signed overflow indicate the status of CF and OF:

* **Unsigned Overflow**: CF = 1 indicates overflow.
* **Signed Overflow**: OF = 1 indicates overflow.

## **Rubric for Lab Assessment**

|  |  |  |  |
| --- | --- | --- | --- |
| The student performance for the assigned task during the lab session was: | | | |
| Excellent | The student completed assigned tasks without any help from the instructor and showed the results appropriately. | 4 |  |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 |  |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 |  |
| Worst | The student did not complete assigned tasks. | 1 |  |

Instructor Signature