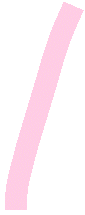
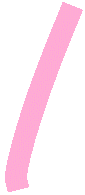
**Department of Electrical Engineering**



|  |  |
| --- | --- |
| **Faculty Member: Ma’am Qurat-ul-ain** | **Dated: November 12, 2020** |
|  |  |
| **Course/Section: BSCS-9B** | **Semester: 3rd** |
|  |  |

**Computer Organization and**



**Assembly Language (CS235)**



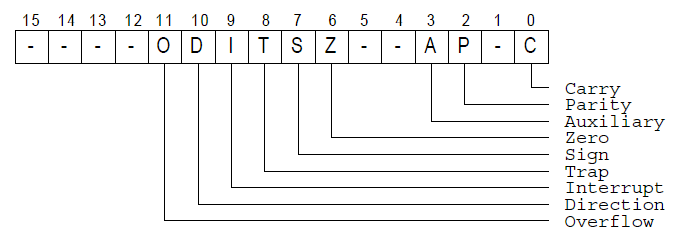
**Lab #5 Effect of instructions on Flag Register**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **PLO4** | | **PLO5** | **PLO8** | **PLO9** |  |
| **Name** | **Roll number** | **Viva /Quiz/ Lab performance**  **5 marks** | **Analysis of data in lab report**  **5 marks** | **Modern tool Usage**  **5 marks** | **Ethics and Safety**  **5 marks** | **Individual and team work**  **5 marks** | **Total** |
| **Fatima Seemab** | **291310** |  |  |  |  |  |  |
| **Mahum Samar** | **290647** |  |  |  |  |  |  |
| **Maryam Fatima** | **290479** |  |  |  |  |  |  |

**Objective:** The aim of this lab is to understand how value of Flag register is affected by different instructions and what the purpose of each flag is.

**Status register Flags**:

1. 16-bit flag to indicate the status of final arithmetic / logical result.



**Control flags (TF, IF, DF)**

Determine how the processor responds to certain situations

**Status flags (CF, PF, AF, ZF, SF, OF)**

Set to represent the result of certain operations

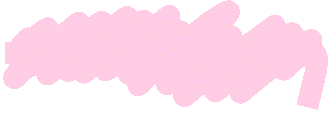
Used to control conditional jump instructions

Status flags are updated to indicate certain properties of the result, for example if the result is zero, zero flag is set. Once a flag is set, it remains in that state until another instruction that affects the flags is executed. Not all instructions affect all status flags.

∗ add and sub affect all six flags



∗ inc and dec affect all but not the carry flag



∗ mov, push, and pop do not affect any flags

**Zero Flag:**

The zero flag will be set whenever the result is zero.

If the result is zero, ZF = 1

Otherwise, ZF = 0



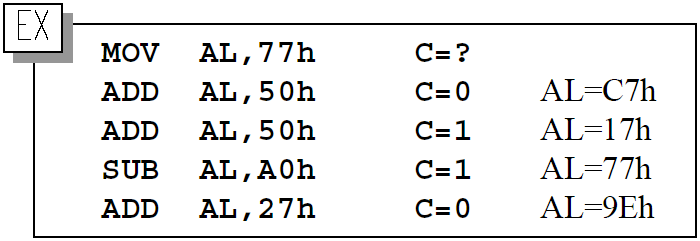
∗Zero can result in several ways (e.g. overflow)

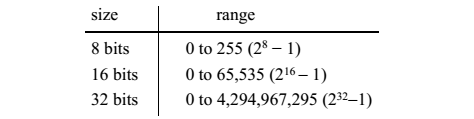


**Carry Flag:**

It Records the fact that the result of an arithmetic operation on **unsigned numbers is out of range**. Carry flag will be set whenever there is a carry or borrow. The carry flag is set in the following examples







Carry flag is not set by inc and dec instructions



**Overflow flag:**

It indicates **out-of-range** **result on signed numbers**. Signed number are counterpart of the carry flag. The following code sets the overflow flag but not the carry flag.



mov AL,72H; 72H = 114D

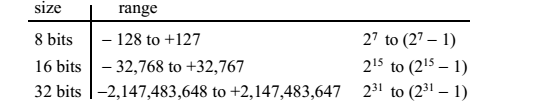


add AL,0EH; 0EH = 14D



sum is +128, since it is out of range of 8-bit reg, so overflow flag become 1.







**Sign flag:**



Indicates the **sign of the result** and useful only when dealing with signed numbers. It is simply a copy of the most significant bit of the result. It is 1 for negative numbers, 0 for positive number.



**Auxiliary flag:**



Indicates whether an operation produced **a carry or borrows in the low-order 4 bits** (nibble) of 8-, 16-, or 32-bit operands (i.e. operand size doesn’t matter)



**Parity flag:**



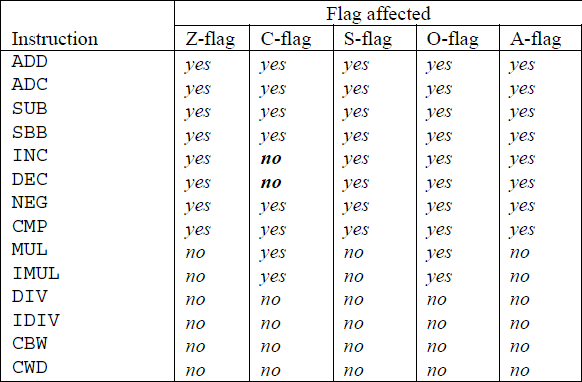
Indicates even parity of the low 8 bits of the result, PF is set, if **the lower 8 bits contain even number 1 bits**. For 16- and 32-bit values, only the least significant 8 bits are considered for computing parity value.

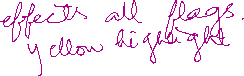


26 (11010) — PF= 0

102 (1100110) — PF= 1

**Following is the table of some instruction and flags effected by those instruction**





**NOTE:** mov, push, and pop do not affect any flags

**Signed or unsigned: How does the system know?**

∗The processor does not know the interpretation

∗It sets carry and overflow under each interpretation

**Exercise # 1** Explain the status of carry flag after each pair of instruction. Insert the screenshot of result also (2 marks- teamwork)

1. mov AL,0FH

add AL,0F1H



CF=?





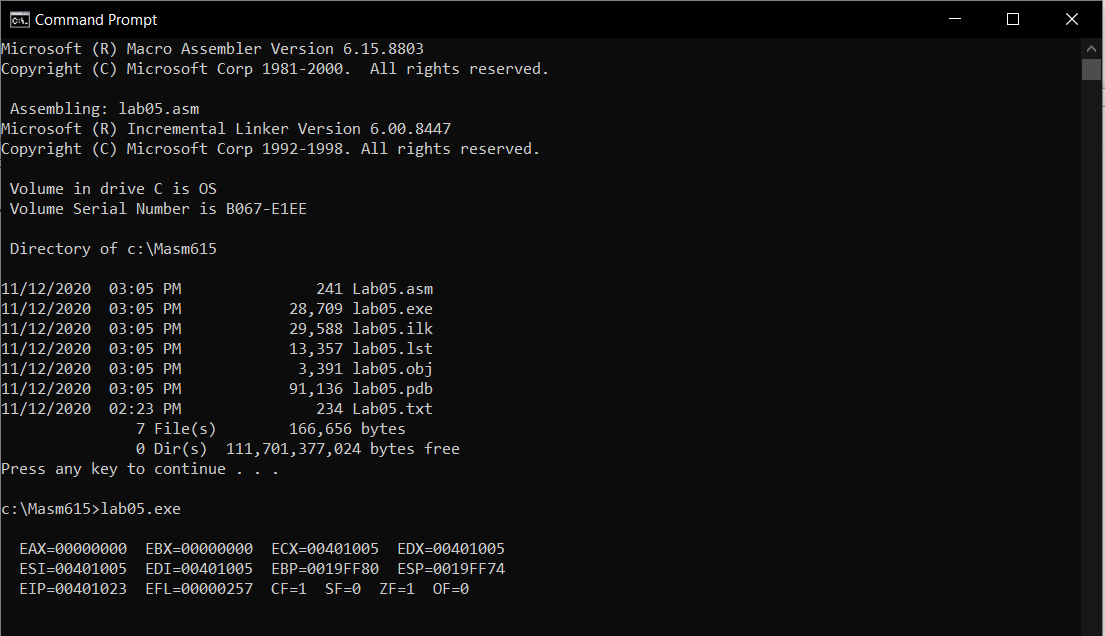
**Reason**:

Its calculation is 100H, which is out of the range of the AL register.



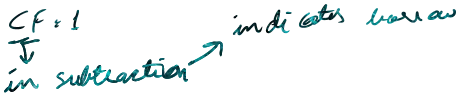
So, CF = 1.





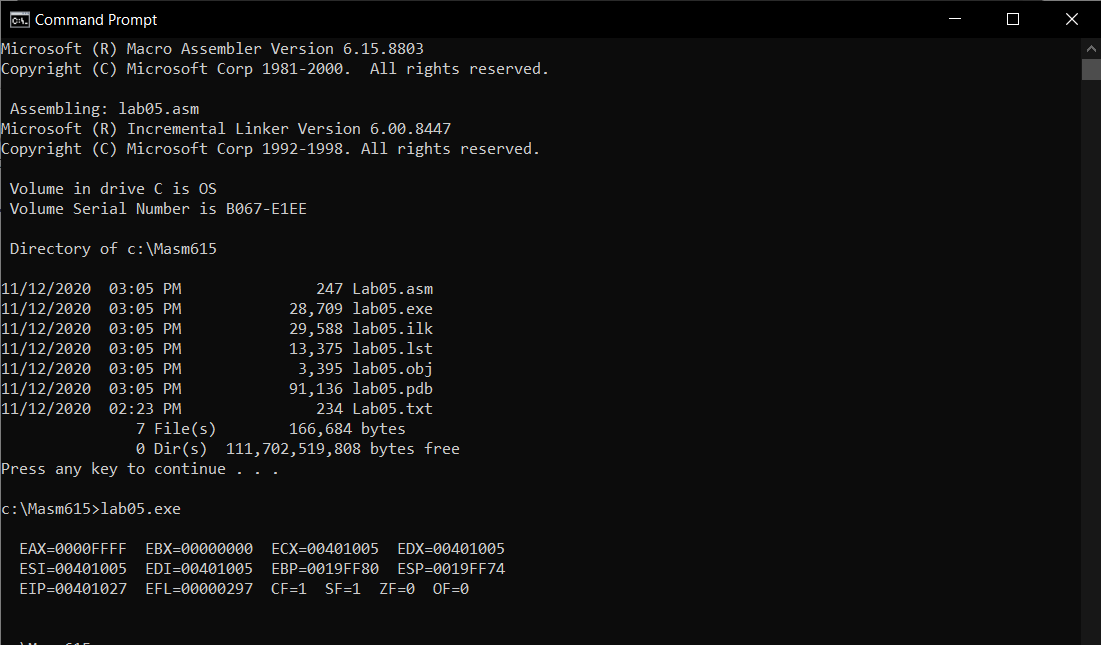


1. mov AX,12AEH

sub AX,12AFH

CF=?

**Reason**:

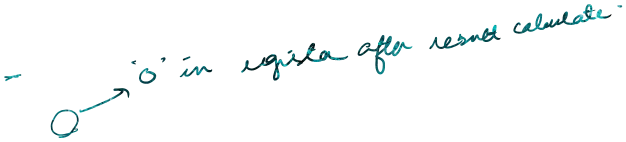
The value in AX is FFFF H, as larger value is being subtracted from small value so CF = 1 as it is indication of borrow.



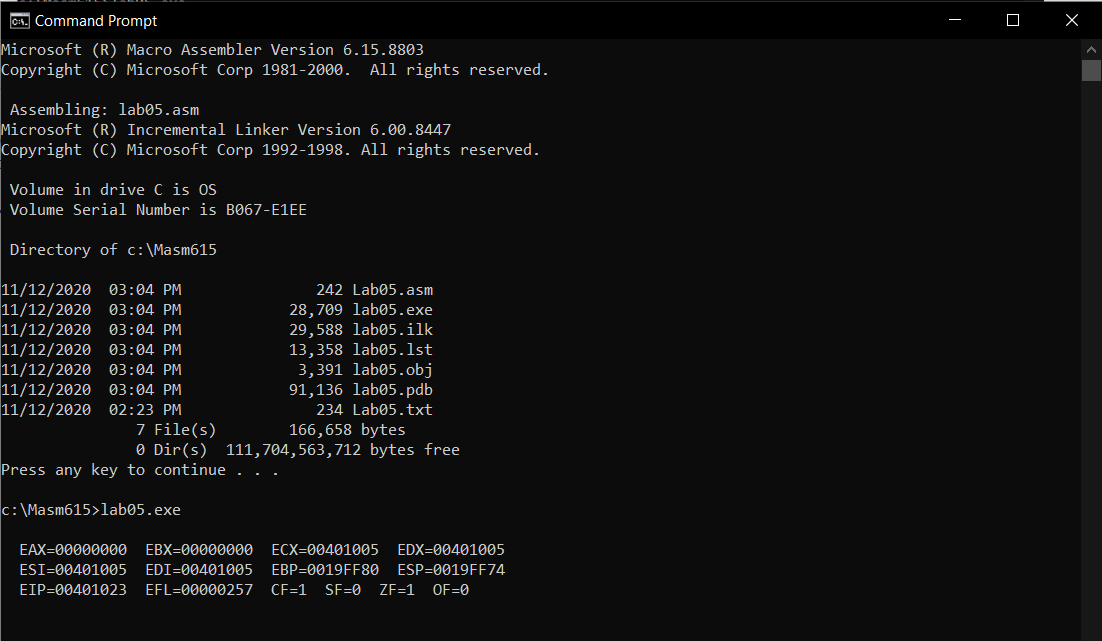
**Exercise # 2** Explain the status of zero flag after each pair of instruction. Insert the screenshot of result also (3 marks- teamwork)

1. mov AL,0FH

add AL,0F1H

ZF=?

**Reason**:

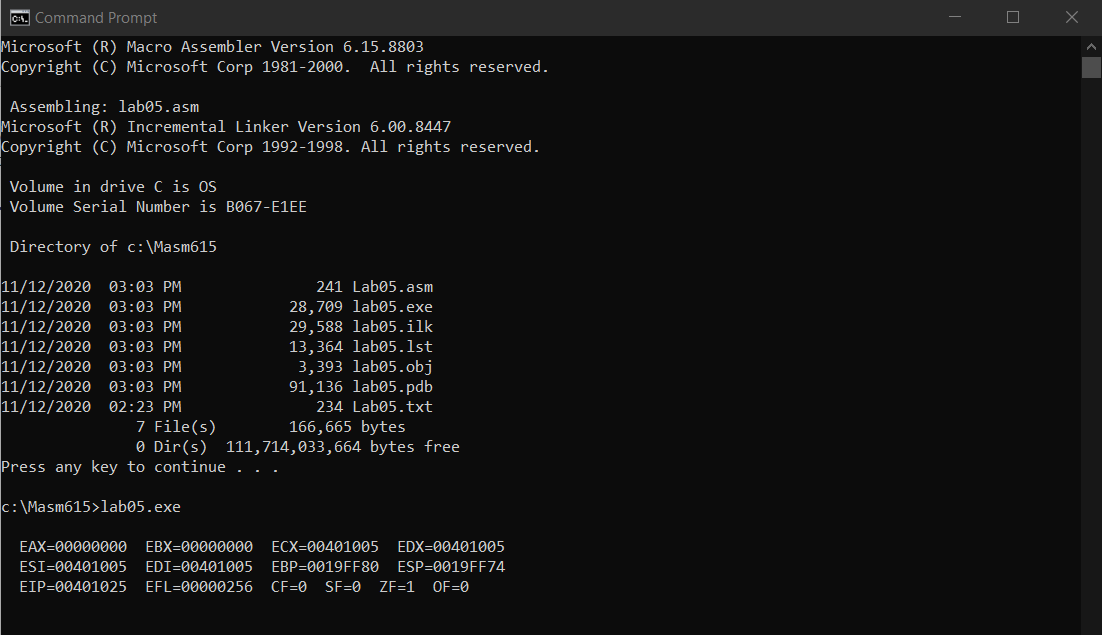
ZF = 1 because the AL register value becomes 00 after the calculation as the result of calculation was 100H which was out of range of AL.

1. mov AX,0FFFFH

inc AX

ZF=?

**Reason**:

ZF = 1 because the result of calculation after the increment of 1, is 10000H which is out of range of AX, so value of AX is 0000H (zero).

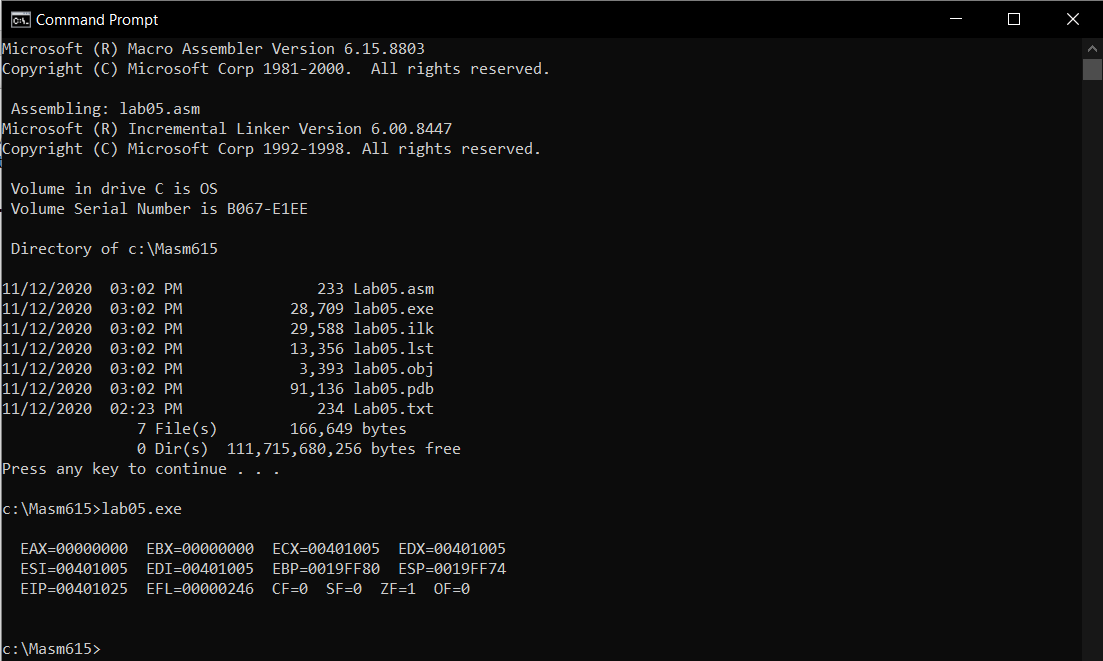


1. mov AX,1

dec AX

ZF=?

**Reason**:

ZF = 1 because the value of AX is 0000H after the decrement.

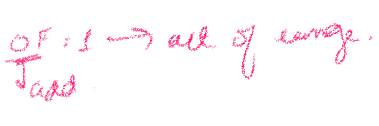


**Exercise # 3** Explain the status of Overflow flag after instruction. Insert the screenshot of result also (2 marks- tool usage)



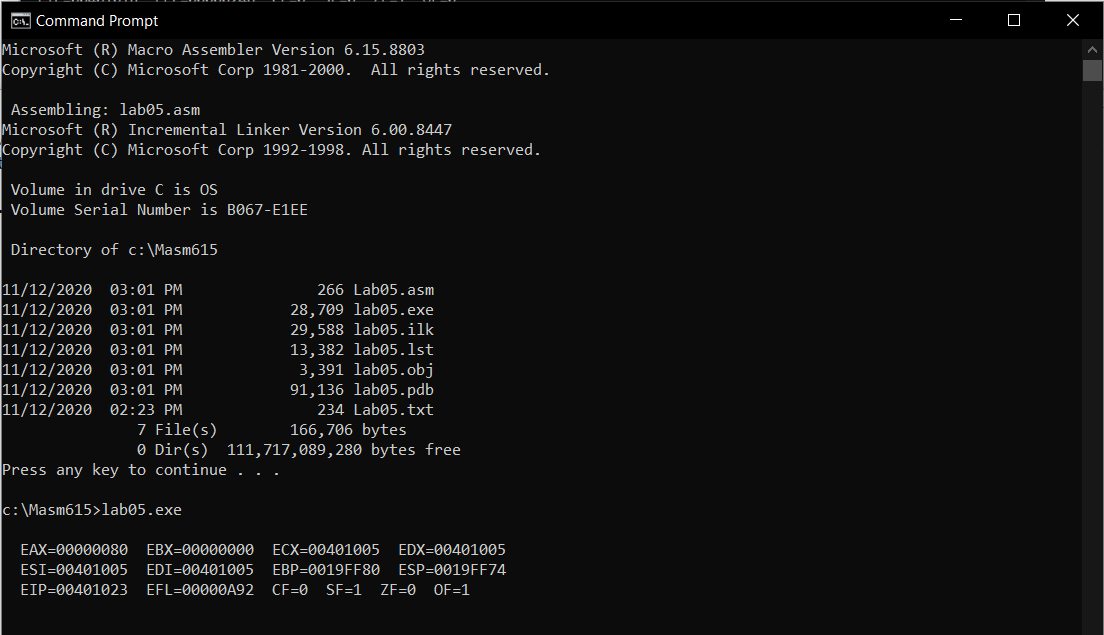
1. mov AL,72H; 72H = 114D

add AL,0EH; 0EH = 14D

OF=?

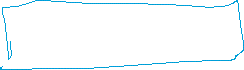
**Reason**:

Sum = +128D. It is out of range of 8-bit register so overflow occurs and OF = 1.



**Exercise :4** Explain the status of Sign flag after each pair of instruction? Insert the screenshot of result also (3 marks- tool usage)





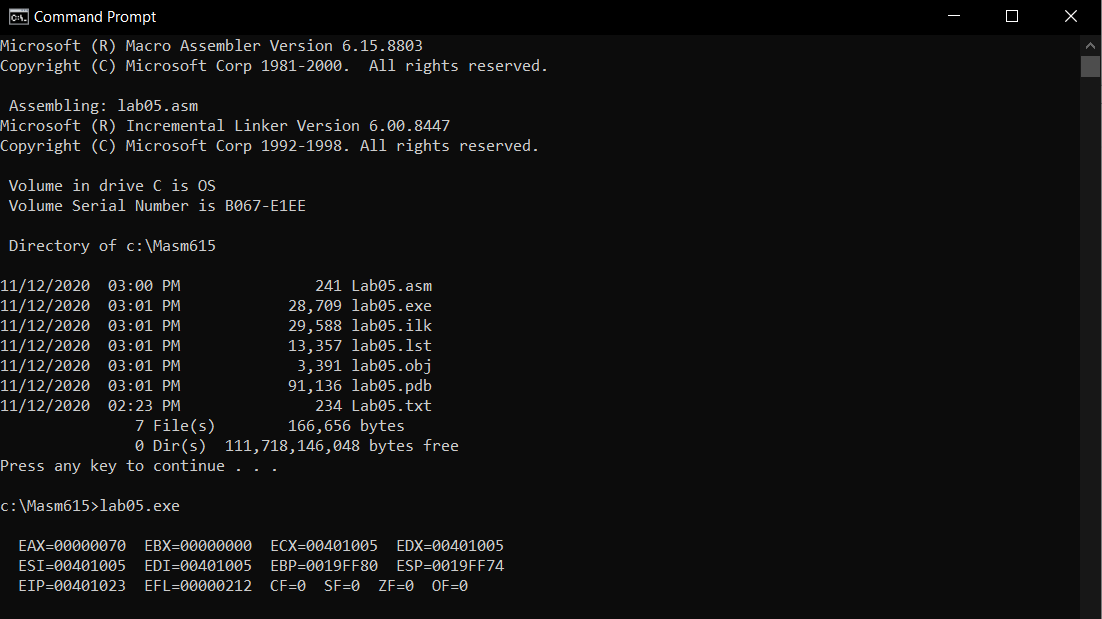
1. mov AL,15

add AL,97

SF=?

**Reason**:

Sum = 112D which is 70H, it is a positive number and there is no overflow.

So, SF = 0.



1. mov AL,15

sub AL,97

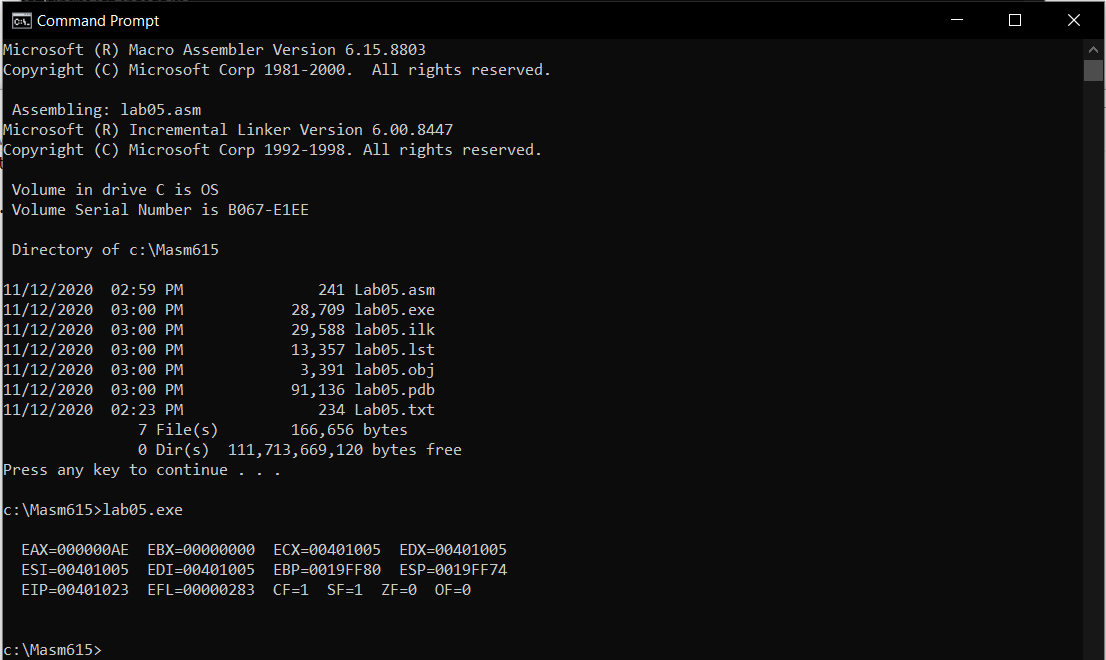
SF=?



**Reason**:

Difference = -82D, which is equal to AEH. It’s a negative number, so SF = 1.





**Exercise:5** Explain the status of flag which is set or reset after execution of each instruction. Insert the screenshot of result also (4 marks- report analysis)

Mov ax,7100h



**Mov bx,4000h ; ZF=1 CF=0 OF=0 SF=0**



**Reason:**

* The mov instruction does not affect any status flag i.e. the status of flags after the mov instruction remains the same as it was before executing mov. And indicate the properties of the result of last operation executed.



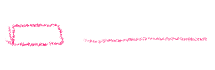
**Add ax, bx ; ZF=0 CF=0 OF=1 SF=1 AX = B100**

**Reason:**



* **ZF**: = 0 because the result is non-zero.



* **CF**: = 0 because result is in range of unsigned numbers.



* **OF**: result is 45312D out of range of signed numbers as two positive numbers are added but the result is negative which should not be the case, so OF = 1



* **SF**: = 1 because the most significant bit is B in hexadecimal. In hex msb between 8 and F represents a negative number.



**Add ax, 7700h ; ZF=0 CF=1 OF=0 SF=0 AX = 2800**



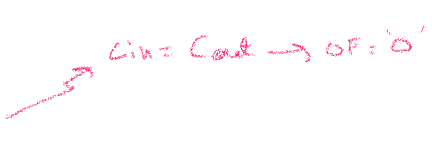
**Reason:**

* **ZF**: = 0 because the result is non-zero.



* **CF**: 75776D out of range of the unsigned number of 16-bit register. So, CF = 1.



* **OF**: The Cin of msb is equal to the Cout of the msb so overflow is zero. Also, because a positive number is added to the negative number(B100H).



* **SF**: = 0 because the most significant bit of AX is 2. In hex msb between 0 and 7 represents a positive number.



**Sub ax,2000h ; ZF=0 CF=0 OF=0 SF=0 AX = 0800**

**Reason:**

* **ZF**: = 0 because the result is non-zero.
* **CF**: 0800H is in range of the unsigned number of 16-bit register. So, CF = 0.



* **OF**: = 0 the Cin of msb is equal to the Cout of the msb so overflow is zero. Also, because a positive number is subtracted from the positive number.



* **SF**: = 0 because the most significant bit is 0 in hexadecimal. In hex msb between 0 and 7 represents a positive number.



**Sub ax,1000h ; ZF=0 CF=1 OF=0 SF=1 AX = F800**

**Reason:**

* **ZF**: = 0 because the result is non-zero.
* **CF**: The result is F800 in hexadecimal, as larger value is being subtracted from small value so CF = 1 as it is indication of borrow.



* **OF**: Two positive numbers are subtracted so OF = 0.



* **SF**: = 1 because the most significant bit is F in hexadecimal. In hex msb between 8 and F represents a negative number.



**Add ax,0800h ; ZF=1 CF=1 OF=0 SF=0 AX = 0000**

**Reason:**

* **ZF**: = 1, the AX register has all the zeros.



* **CF**: The result is 10000H which is out of range of AX register so CF = 1.



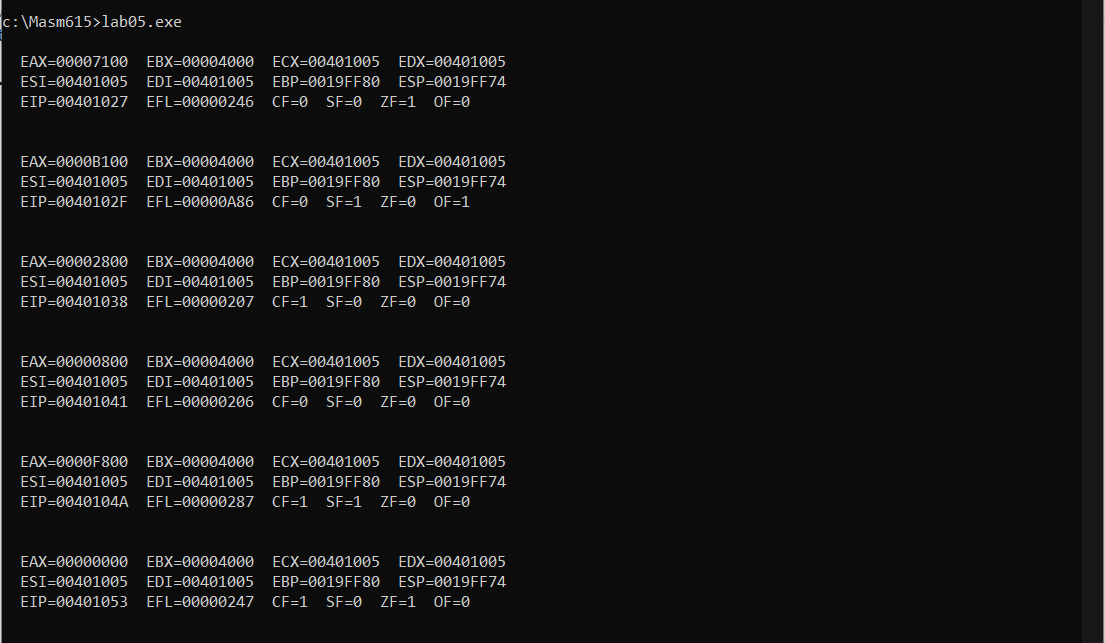
* **SF**: = 0 because the most significant bit of AX is 0. In hex msb between 0 and 7 represents a positive number.



* **OF**: = 0 the Cin of msb is equal to the Cout of the msb so overflow is zero. Also, because a positive number is added to the negative number (F800H).



**Output:**



Also Write **conclusion** of a lab. (1 mark - report analysis)

**Conclusion of the Lab:**



Status flags indicates whether the result is correctly represented or not. Whether it is:

* **Too Large:** when the CF = 1 in unsigned and OF = 1 in signed number.



* **Positive, Negative:** Indicated by SF, the msb in the register; may indicate the wrong sign in case of too large result.
* **Zero:** Indicated by the ZF = 1. ZF can be misleading if the result is too large.



Moreover, the mov, push, pop, div, idiv, cbw and cwd instruction does not affect any status flags i.e. the status of flags after these instructions remains the same as it was before execution.



