

## **Lab 4**

### **ALU Operations**

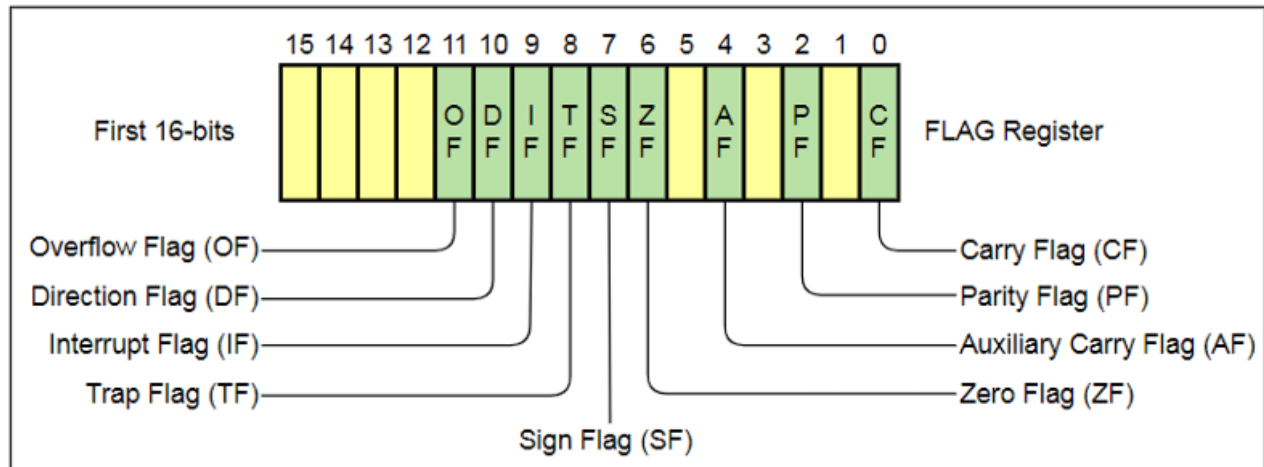
#### **Objectives**

After completing this lab

- Students will learn about various flags and their purposes.
- Students will learn various arithmetic and logic operations.
- Students will be able to perform arithmetic operations and logic operations on n-bit numbers.
- Students will learn what instructions affect which flags.

## Flag Register

The 8086 processor contains a 16-bit flag register where nine bits are used as flags as shown in the following figure. Out of which, 6 flags indicate the status of recently executed instruction, while the remaining 3 are control flags.



Flag registers of 8086 microprocessor

### Status Flag Registers:

#### 1. Overflow Flag (OF):

The overflow Flag is set to **1** when there is a **signed overflow**. For example, when you add bytes **100 + 50** (result is not in range -128...127).

#### Example:

On adding bytes 100 + 50 (result is not in range -128...127), so overflow flag will set.

```
MOV AL, 50 (50 is 01010000 which is positive)
MOV BL, 32 (32 is 00110010 which is positive)
ADD AL, BL (82 is 10000010 which is negative)
```

Overflow flag became set as we added 2 +ve numbers and we got a -ve number.

#### 2. Sign Flag (SF):

The sign Flag is set to **1** when the result is **negative**. When the result is **positive** it is set to **0**. This flag takes the value of the most significant bit.

### 3. Zero Flag (ZF):

Zero Flag (ZF) is set to **1** when the result is **zero**. For non-zero results, this flag is set to **0**.

### 4. Auxiliary Flag (AF):

Auxiliary Flag is set to **1** when there is an **unsigned overflow** for low nibble (4 bits).

### 5. Parity Flag (PF):

The parity Flag is set to **1** when there is an even number of ones in the result and set to **0** when there is an odd number of ones.

### 6. Carry Flag (CF);

Carry Flag is set to **1** when there is an **unsigned overflow**.

### Example

When you add bytes **255 + 1** (result is not in range 0...255). When there is no overflow, this flag is set to **0**.

### Control Flag Registers:

#### 1. Direction Flag (DF):

Direction Flag is used by some instructions to process data chains, when this flag is set to **0** – the processing is done forward, when this flag is set to **1** the processing is done backward.

#### 2. Interrupt Enable Flag (IF):

When the Interrupt Enable Flag is set to **1** CPU reacts to interrupts from external devices.



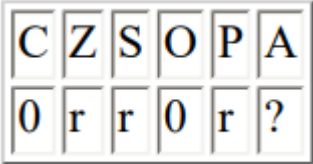

#### 3. Trap Flag (TF):

Trap Flag is used for on-chip debugging.

## Arithmetic and Logic Operations

The following table lists various arithmetic and logical operations, their descriptions, and their effects on flags.

Instructions D – Destination Operand S – Source Operand	Description	Flag status: 0 – Clear 1 – Set ? – Unknown r – Depends on the Result												
ADD D,S	$D \leftarrow D + S$	<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr></table>	C	Z	S	O	P	A	r	r	r	r	r	r
C	Z	S	O	P	A									
r	r	r	r	r	r									
ADC D,S	$D \leftarrow D + S + CF$	<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr></table>	C	Z	S	O	P	A	r	r	r	r	r	r
C	Z	S	O	P	A									
r	r	r	r	r	r									
SUB D,S	$D \leftarrow D - S$	<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr></table>	C	Z	S	O	P	A	r	r	r	r	r	r
C	Z	S	O	P	A									
r	r	r	r	r	r									
SBB D,S	$D \leftarrow D - S - CF$	<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td><td>r</td></tr></table>	C	Z	S	O	P	A	r	r	r	r	r	r
C	Z	S	O	P	A									
r	r	r	r	r	r									
AND D,S	$D \leftarrow D \text{ AND } S$	<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td></tr><tr><td>0</td><td>r</td><td>r</td><td>0</td><td>r</td></tr></table>	C	Z	S	O	P	0	r	r	0	r		
C	Z	S	O	P										
0	r	r	0	r										
OR D,S	$D \leftarrow D \text{ OR } S$	<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td>0</td><td>r</td><td>r</td><td>0</td><td>r</td><td>?</td></tr></table>	C	Z	S	O	P	A	0	r	r	0	r	?
C	Z	S	O	P	A									
0	r	r	0	r	?									
NOT D	$D \leftarrow \text{NOT } D$	<table><tr><td>C</td><td>Z</td><td>S</td><td>O</td><td>P</td><td>A</td></tr><tr><td colspan="6">unchanged</td></tr></table>	C	Z	S	O	P	A	unchanged					
C	Z	S	O	P	A									
unchanged														

INC D	$D \leftarrow D + 1$	 <p>CF - unchanged!</p>
DEC D	$D \leftarrow D - 1$	 <p>CF - unchanged!</p>
XOR D,S	$D \leftarrow D \text{ XOR } 1$	
NEG D	$D \leftarrow (\text{NOT } D) + 1$	 <p>CF - unchanged!</p>

**Program to add 100 and 50 to check status of various flags.**

```
.model small  
  
.data  
  
.code  
  
mov al, 100  
add al, 50  
  
.exit
```

The description of the above program is shown in the following table.

Instruction	Description
mov al,100	Moving value:100 to the AL register
add al, 50	<p>Adding value: 50 to the contents of the AL register and storing the result back to the AL register</p> <p>The sum of 150 exceeds the range of 8-bit signed numbers and will set the overflow (OF) flag.</p> <p>The sum of 150 will not affect the carry flag (CF) as the range for 8-bit unsigned numbers is 0–255.</p> <p>The number of ones in 150 is even, which will set the parity flag (PF).</p> <p>The sign flag (SF) will also be set as the most significant bit of the result is 1.</p>

## Emu8086 Tutorial Step-by-Step

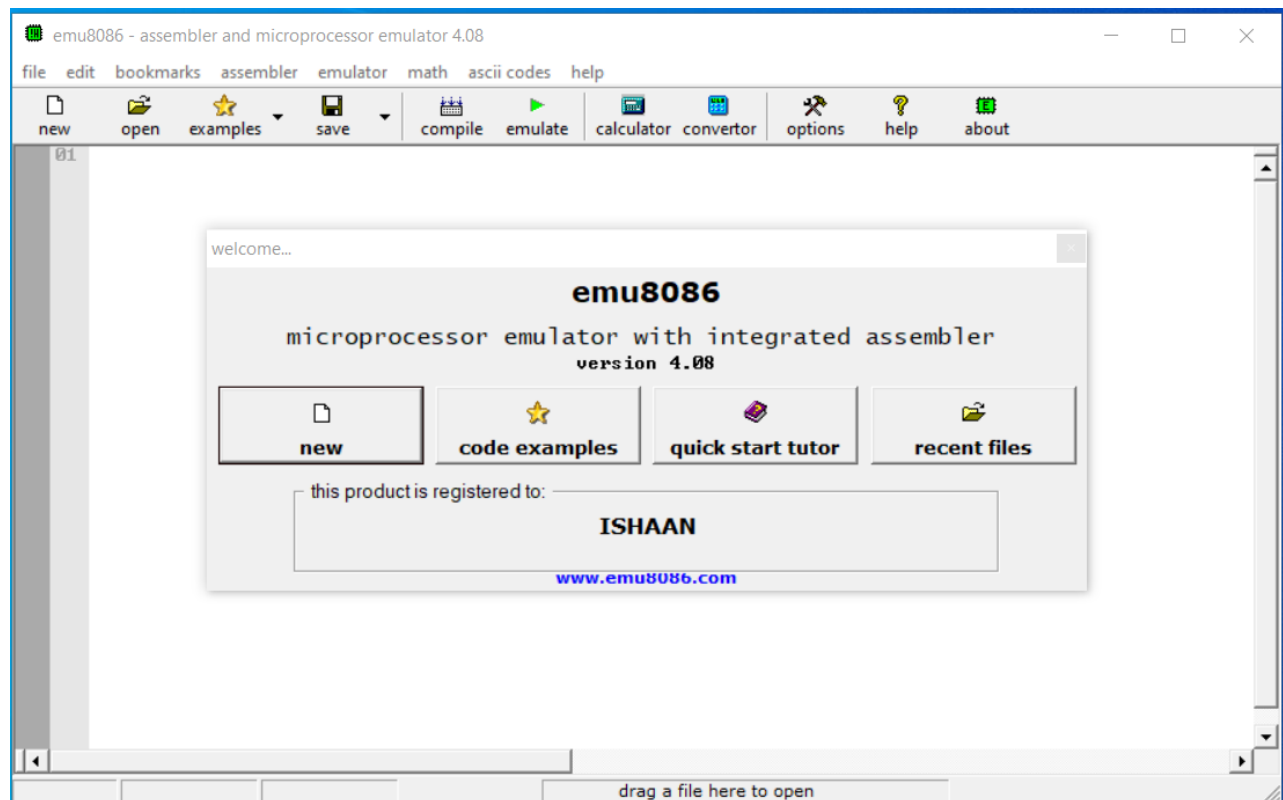
### Step-1



Double-click on the icon on the desktop

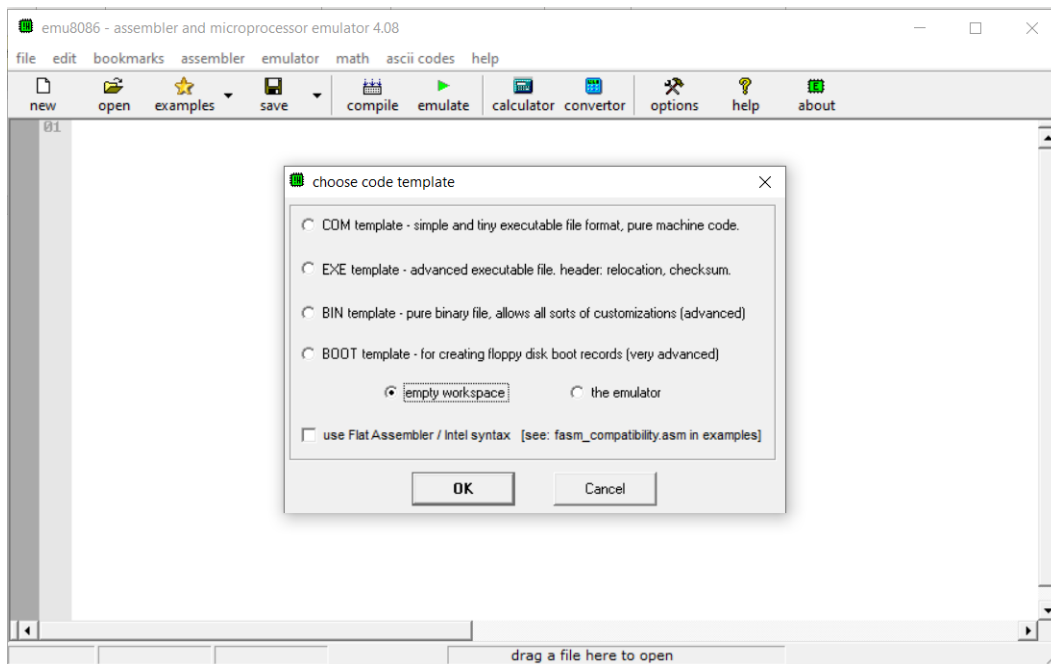
### Step-2

The following window will appear. Click on new.



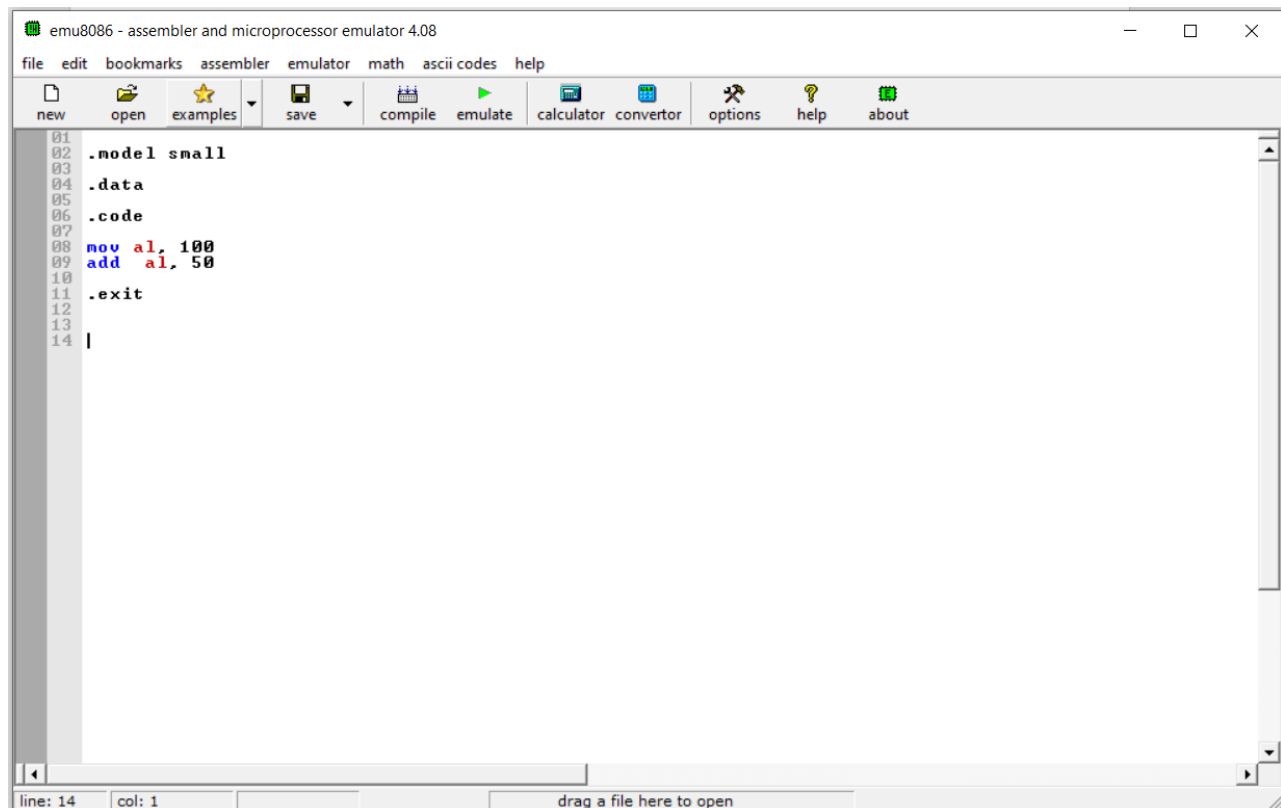
### Step-3

Click on the empty workspace and press OK.



### Step-4

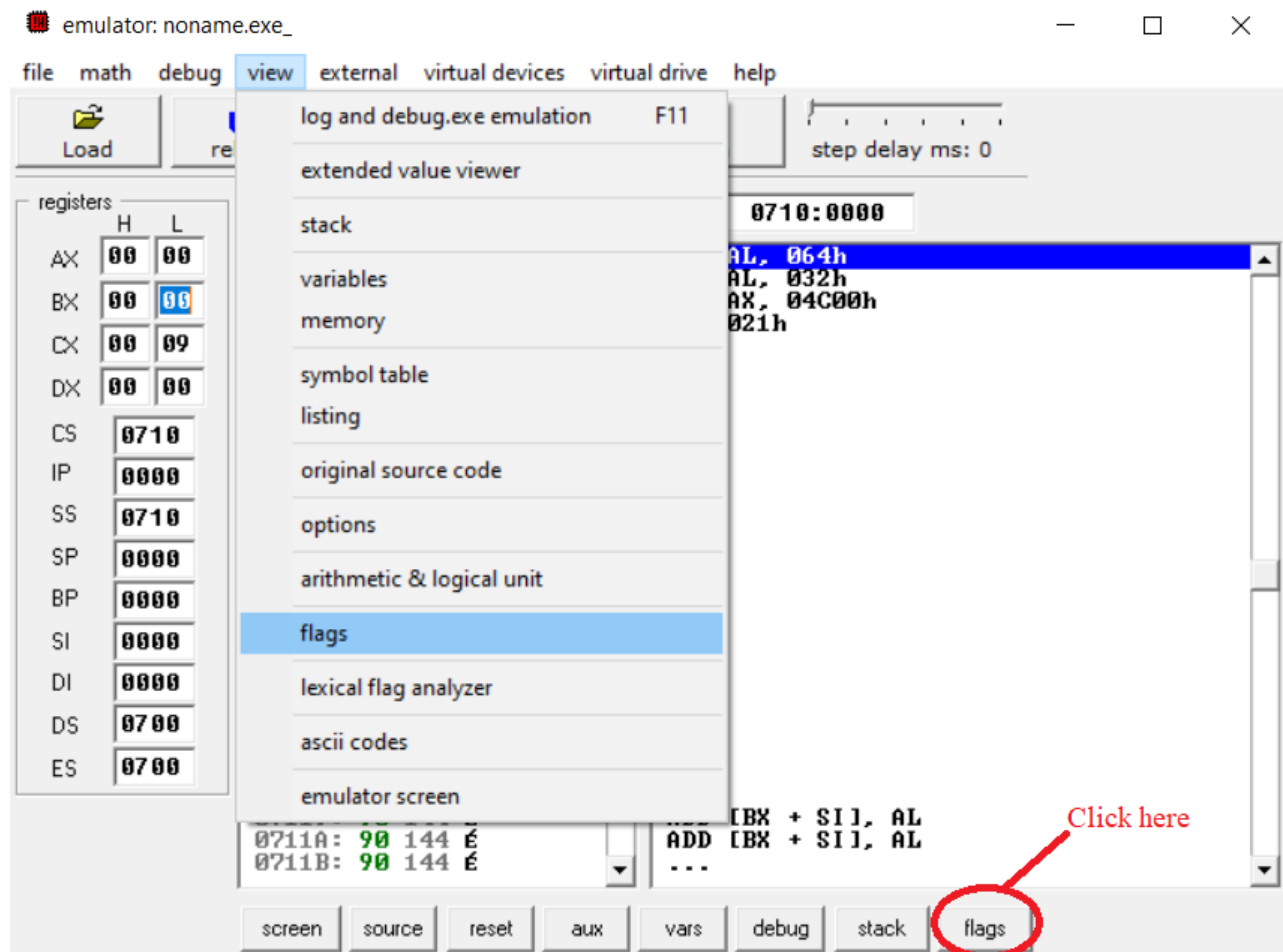
Type the code given above and click on emulate.





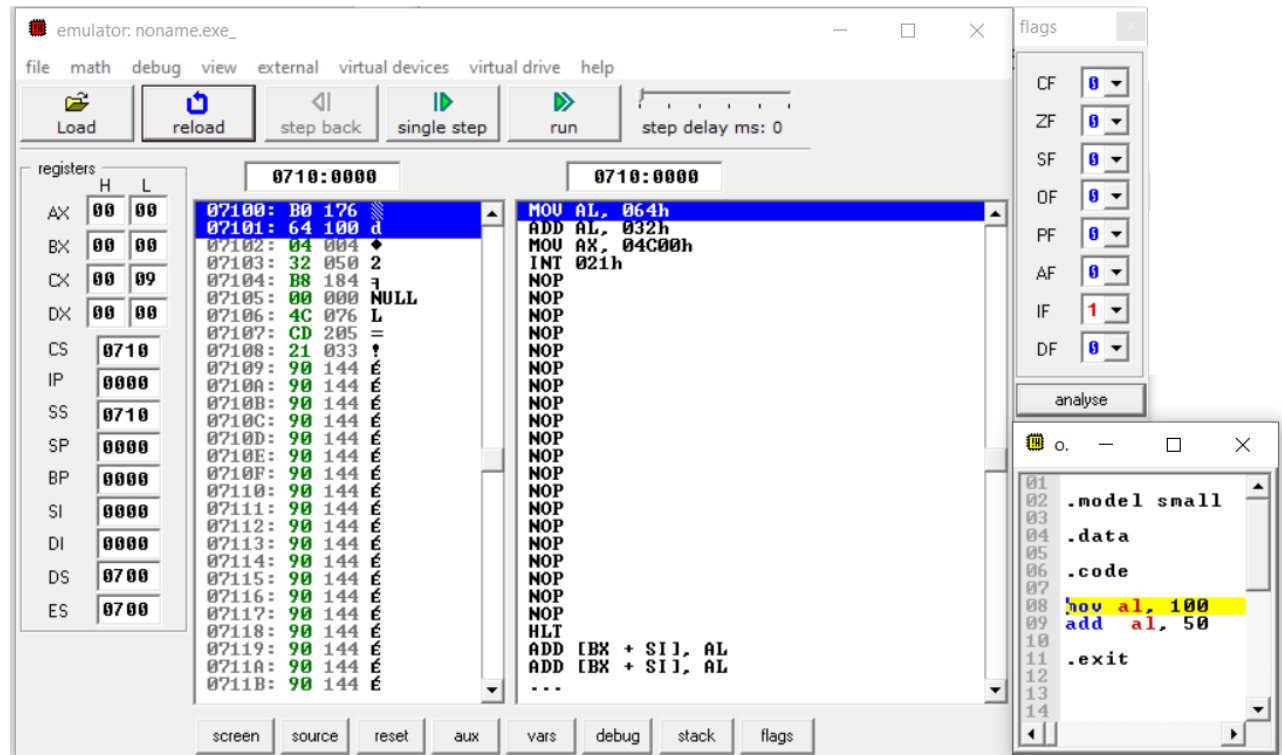
### Step-5

Click on “flags” from the view menu OR click on the button “flags” at the bottom.



**Step-6**

Keep clicking on “Single step” to execute program instructions one by one. Stop clicking “Single step” just after the “Add al, 50” instruction to observe various flags.





## Practice Exercise

### Task-1

Write a program that stores the given two 32-bit numbers into the current data segment at offset: 0x1000 and 0x1008, respectively. The program then calculates the sum of these numbers and stores it at the offset: 0x1010.

**Numbers:** 0x1F540398, 0xC0A1F02E

### Task-2

Write a program to implement the following equation.

$$X = \sim 0xFF12 \wedge \{0xABFF \& (0x2113 \mid 0x2340)\}$$

~	Invert all bits
^	Bitwise XOR
&	Bitwise AND
	Bitwise OR

### Task-3

Perform any ALU operation that sets CF and OF at the same time.