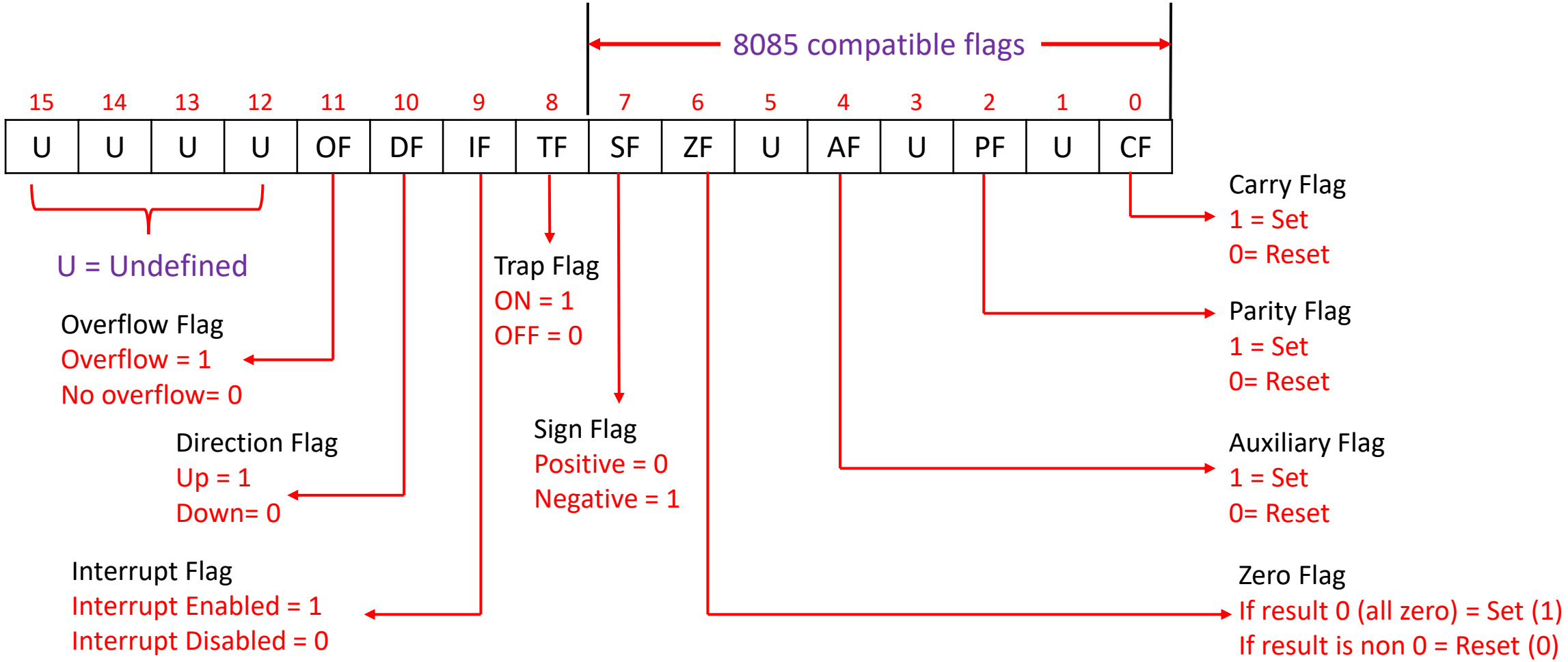
Flag Register of 8086

Flag Register of 8086

- Flag register is a part of Execution Unit.
- It is a 16-bit register with each bit corresponding to a flip-flop.
- Flag register is used to give status of operation performed by processor.
- A flag is flip-flop.
- It indicates some condition produced by the execution of an instruction.



Carry Flag (CF)

- It can also be called as a final carry.
- This flag is set whenever there has been a carry out of, or borrow into, the MSB of the result (8 bit / 16 bit).
- The flag is used by the instruction that add and subtract multibyte numbers.
- CF = 1 , if there is a carry out from the most significant bit (MSB)
- CF = 0 , if no carry out from MSB

Example 1 (8 bit) :

ADD BL, CL where BL = 02 H and CL= 51 H

```
  BL = 02 h =   0 0 0 0  0 0 1 0
+ CL = 51 h =   0 1 0 1  0 0 0 1
-------------------------------
               0 1 0 1  0 0 1 1
```

MSB LSB

Carry is not generated from MSB

U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

0

Example 2 (8 bit) :

ADD BL, CL where BL = 83 H and CL= 81 H

```
  BL = 83 h =   1 0 0 0  0 0 1 1
+ CL = 81 h =   1 0 0 0  0 0 0 1
    1                    1 1
-------------------------------
               0 0 0 0  0 1 0 0
```

MSB LSB

Carry is generated from MSB

U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

1

Example 1 (16 bit) :

ADD BX, CX where BX = 0212 H and CX= 1251 H

BX = 0212 h = 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0

CX = 1251 h = 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0 1

1 1

0 0 0 1 0 1 0 0 0 1 1 0 0 0 1 1

MSB LSB

Carry is not generated from MSB

U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

0

Parity Flag (PF)

- This flag is normally used to check data transmission errors.
- PF = 1 , when the result has even parity, an even number of 1's
- PF = 0 , when the result has odd parity, an odd number of 1's

Example 1 :

ADD BL, CL where BL = 02 H and CL= 51 H

```
  BL = 02 h =   0 0 0 0  0 0 1 0
+ CL = 51 h =   0 1 0 1  0 0 0 1
              ---------------------
              0 0 0 0  0 0 1 1
```

Result has even number of 1's

U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF
													1		

Example 2 :

ADD BL, CL where BL = 83 H and CL= 81 H

```
  BL = 83 h =   1 0 0 0  0 0 1 1
  CL = 81 h =   1 0 0 0  0 0 0 1
          1                 1 1
              ---------------------
              0 0 0 0  0 1 0 0
```

Result has odd number of 1's

U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF
													0		

Auxiliary Carry Flag (AF)

- It is a carry generated from lower nibble to upper nibble.
- AF = 1 , if carry or borrow generated from lower nibble to upper nibble.
- AF = 0 , if carry or borrow not generated from lower nibble to upper nibble.

8 bit data

02 h = 0 0 0 0 0 0 1 0

HB – Higher Nibble **LB – Lower Nibble**

16 bit data

0212 h = 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0

HB – Higher Nibble **LB – Lower Nibble**

ADD BL, CL where BL = 02 H and CL= 51 H

BL = 02 h = 0 0 0 0 0 0 1 0

+ CL = 51 h = 0 1 0 1 0 0 0 1

0 0 0 0 0 0 1 1

Carry is not generated from lower nibble to higher nibble.

U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

0

ADD BL, CL where BL = 08 H and CL= 58 H

Carry is generated from lower nibble to higher nibble.

```
   BL = 08 h =   0 0 0 0   1 0 0 0
+  CL = 58 h =   0 1 0 1   1 0 0 0
                       1
  ---------------------------------
                0 0 0 0   0 0 0 0
```

U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF
											1				

Example 1 (16 bit) :

ADD BX, CX where BX = 0082 H and CX= 1281 H

```
   BX = 0212 h =   0 0 0 0   0 0 0 0   1 0 0 0   0 0 1 0
   CX = 1251 h =   0 0 0 1   0 0 1 0   1 0 0 0   0 0 0 1
                           1
  -----------------------------------------------------
                   0 0 0 1   0 0 1 1   0 1 1 0   0 0 1 1
```

Carry is generated from lower nibble to higher nibble.

AF = 1

Zero Flag (ZF)

- This flag is normally used to check the result of operation is zero or non zero.
- This flag is monitor in Compare instruction.
- ZF = 1 , when the result consist all bits zero
- ZF = 0 , when the result is non zero which means at least one bit is 1.

Example 1 :

SUB BL, CL where BL = 02 H and CL= 02 H

BL = 02 h = 0 0 0 0 0 0 1 0

\- CL = 02 h = 0 0 0 0 0 0 1 0

0 0 0 0 0 0 0 0

Result has all zero

U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

1

Example 2 :

ADD BL, CL where BL = 83 H and CL= 81 H

BL = 83 h = 1 0 0 0 0 0 1 1

CL = 81 h = 1 0 0 0 0 0 0 1

1 1 1

0 0 0 0 0 1 0 0

Result is non zero

U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

0

Sign Flag (SF)

- MSB of the result is used to indicate whether the result is positive or negative.
- SF = 0 , result is positive number
- SF = 1 , result is negative number

S	
0	+ ve
1	- ve

ADD BL, CL where BL = 02 H and CL= 51 H

U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF
								↓							
								0							

BL = 02 h = 0000 0010

+ CL = 51 h = 0101 0001

0000 0011

-23 h 1101 1101

+ -31 h 1100 1111

- 54 h 1010 1100

U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF
								↓							
								1							

Trap Flag (TF)

- Setting TF puts the processor into single step mode for debugging.
- In single stepping, microprocessor executes a instruction and enters into single step ISR.
- After that user can check registers or memory contents, if found ok, he/she will proceed the further, else necessary action will be taken.
- This utility is called as debug the program.
- if TF=1 , the CPU automatically generates an internal interrupt after each instruction, allowing a program to be inspected as it execute instruction by instruction.
- TF = 1 , Trap on (single instruction execution)
- TF = 0 , trap off (all instructions execution)

TF = 0 , trap off

Registers	Memory
AL =	MOV BL, 02h
BL = 04	MOV CL, 02h
CL = 02	ADD BL, CL
DL =	

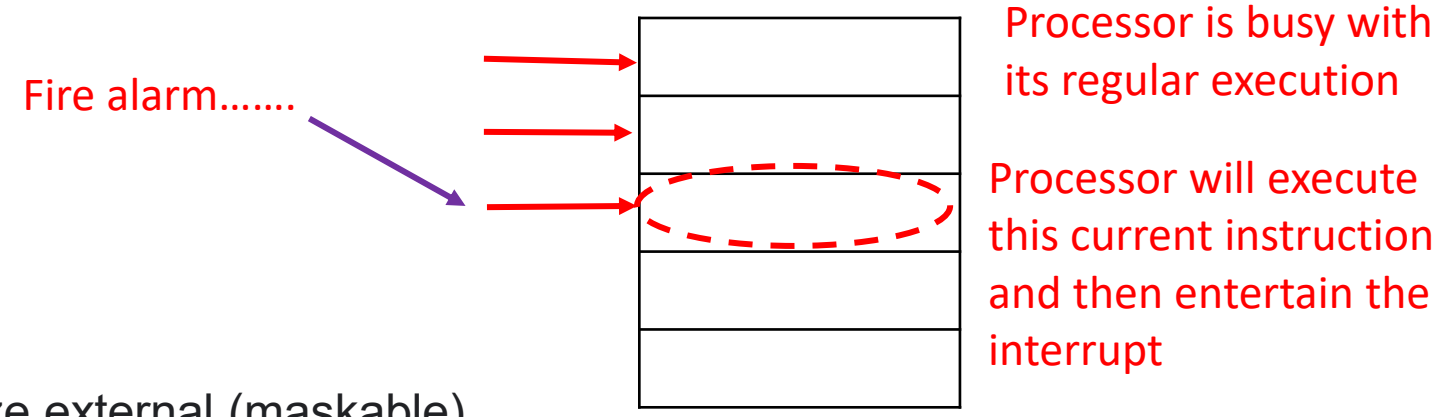
Processor execute all instructions and displays output

TF = 1 , trap ON

Registers	Memory
AL =	MOV BL, 02h ←
BL = ~~02~~ 04	MOV CL, 02h ←
CL = 02	ADD BL, CL ←
DL =	

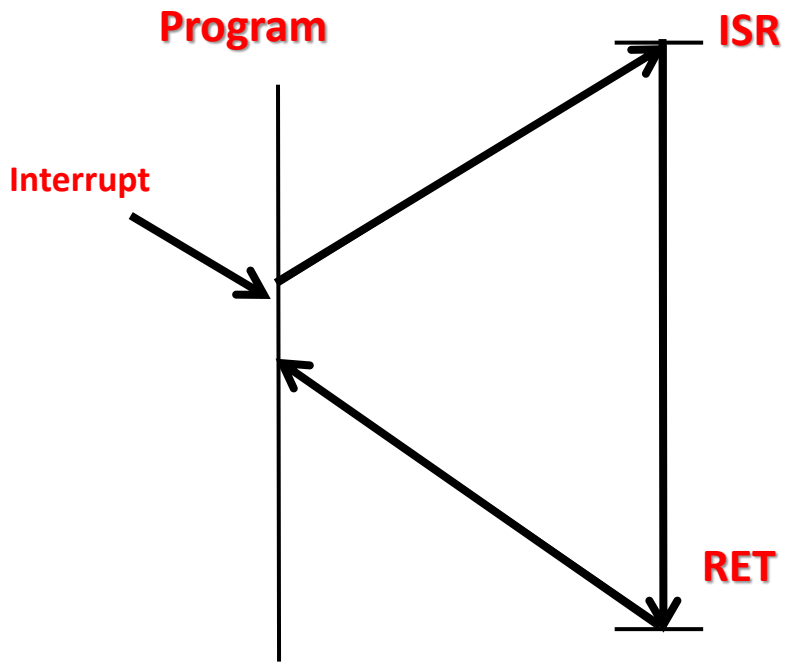
Processor instruction one by one and wait

What is an interrupt??? Ans : To disturb regular flow of processor execution



Interrupt Flag (IF)

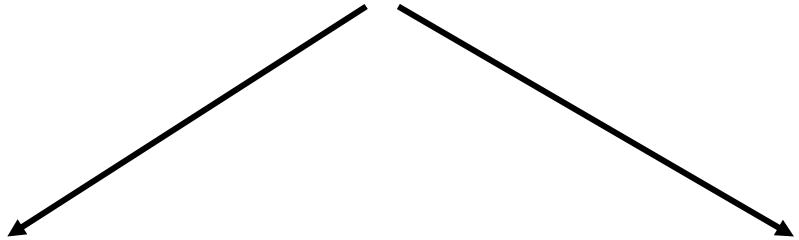
- If user sets IF flag, the CPU will recognize external (maskable) interrupt requests.
- Clearing IF disables these interrupts.
- IF = 1 , interrupt enabled.
- IF = 0 , interrupt disabled.



Overflow Flag (OF)

- It indicates an overflow from the magnitude to the sign bit of result.
- If OF is set, an arithmetic overflow has occurred, that is a significant bit has been lost because the size of the result exceeded the capacity of its destination location.
- In 8086 interrupt on overflow instruction is available that will generate an interrupt in this situation.
- OF = 1 , signed overflow occur
- OF = 0 , no overflow

Number System



Signed Numbers

Positive & Negative
+ ve & - ve

Unsigned Numbers

they assume to be positive

No sign i.e. no + ve nor - ve

Unsigned Numbers

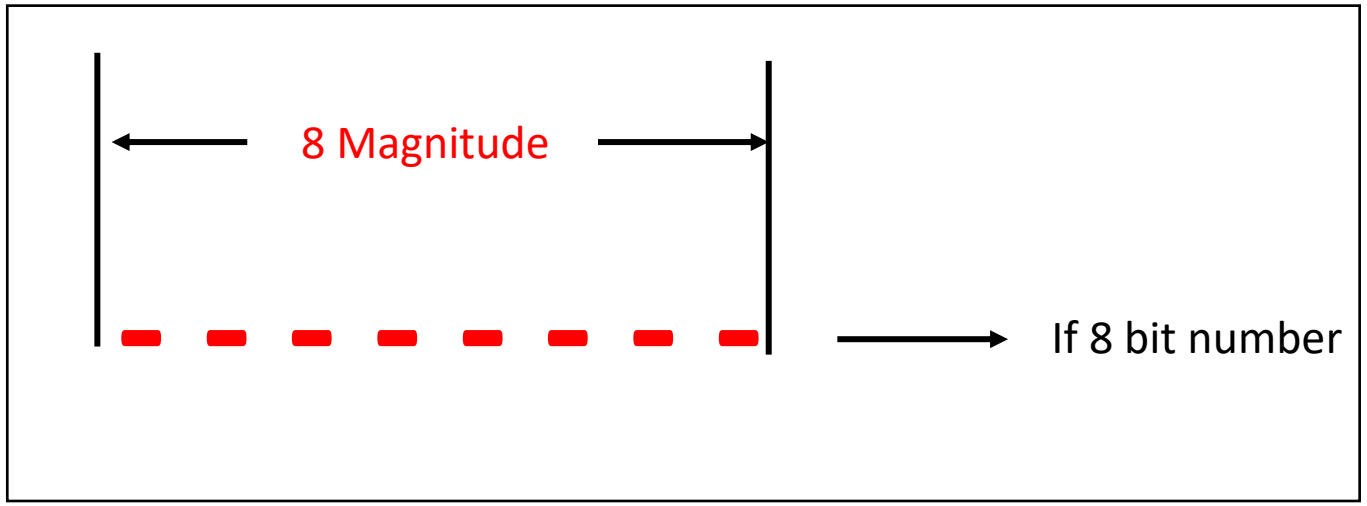
All Positive numbers

Example : Roll Number

For Unsigned numbers if magnitude is 8 then,

$2^8 = 256$

i.e. total 256 + ve numbers are used



Range for unsigned numbers

00	0000 0000
01	0000 0001
FF	1111 1111

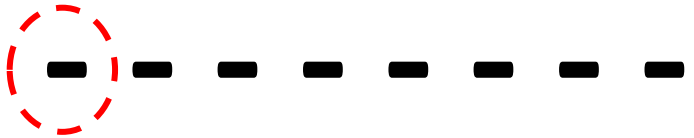
BACK

Signed Numbers

Positive & Negative
+ ve & - ve

How to find whether number is + ve or - ve?

Answers :

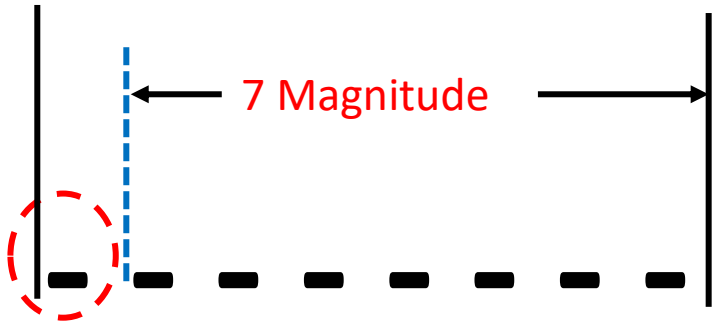


MSB LSB

If MSB of number is 0 then + ve

If MSB of number is 1 then - ve

If number is 8 bit



S	
0	+ ve
1	- ve

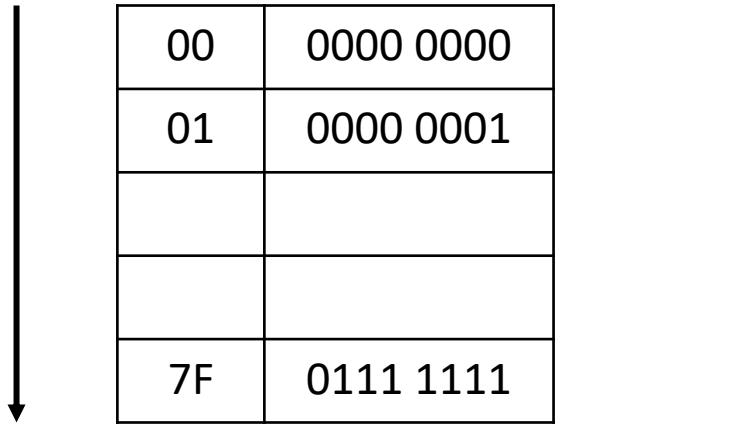
For signed numbers if magnitude is then 7 ,
$2^7 = 128$
i.e. 128 **- ve** numbers & 128 **+ ve** numbers

-128 to -1 0 to 127

Range for +ve and –ve numbers

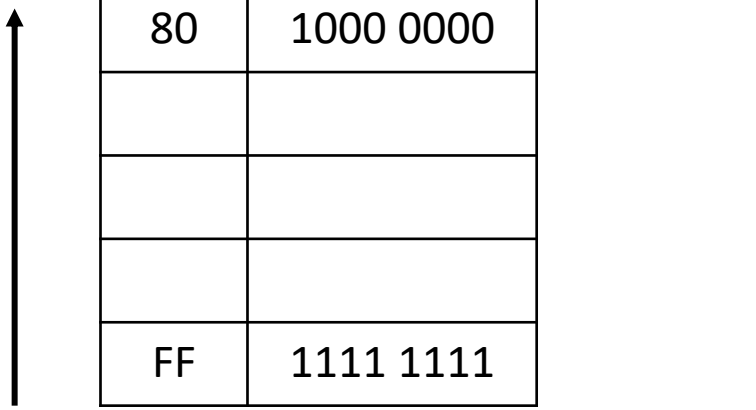
Range for +ve and –ve numbers

Range for Positive numbers (0 to 127)



00	0000 0000
01	0000 0001
7F	0111 1111

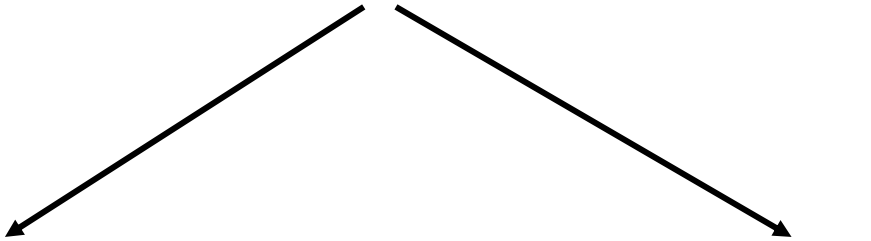
Range for Negative numbers (- 80 to -01)



80	1000 0000
FF	1111 1111

What is unsigned and signed number following binary ?

1 0 0 0 0 0 1 1



Unsigned

83 H

Signed

03 H 7D H

- ve number means 2's complement of given number

Shortcut for 2's complement : copy number as it is from right side till gets first 1 after 1 complement all numbers

example 1 :

+ 24 h	**0010 0100**
2's complement of 24 i.e. -24	**1101 1100**

example 1 :

+ 5 h	**0101**
2's complement of 05 i.e. -5	**1011**

NEXT

Overflow Flag :

Overflow flag matters only for signed numbers

Range for Positive numbers (0 to 127)

00	0000 0000
01	0000 0001
7F	0111 1111

Range for Negative numbers (- 128 to -01)

80	1000 0000
FF	1111 1111

01	0000 0001
-01	1111 1111
80	1000 0000
-80	1000 0000

Range for Positive numbers : 00 to 7F

Range for Negative numbers : FF to 80

After addition if result is going beyond above ranges then overflow flag is set i.e. OF = 1

Example for overflow flag:

Overflow flag matters only for signed numbers

23 h	0010 0011
+ 31 h	0011 0001
54 h	0101 0100

CY	AC	OF	P
0	0	0	0

-23 h	1101 1101
+ -31 h	1100 1111
- 54	1010 1100

CY	AC	OF	P
1	1	0	1

27 h	0010 0111
+ 39 h	0011 1001
60 h	0110 0000

CY	AC	OF	P
0	1	0	0

-27 h	1101 1001
+ - 39 h	1100 0111
- 60 h	1010 0000

CY	AC	OF	P
1	1	0	1

42 h	0100 0010
+ 43 h	0100 0011
85 h	1000 0101

CY	AC	OF	P
0	0	1	0

-42 h	1011 1110
+ -43 h	1011 1101
-85 h	0111 1011

CY	AC	OF	P
1	1	1	1

1. Using MSB bit we can identify whether number is +ve or –ve
2. If MSB is 1 it means number is – ve
3. But sometimes it will give wrong sign bit
4. In such cases checking only MSB is not sufficient
5. We have to check range of both numbers
6. If number cross range it means there is overflow problem

Range for Positive numbers : 00 to 7F (0 to 127)

Range for Negative numbers : FF to 80 (-80 to -01)

Example :

7F h	0111 1111
+ 01 h	0000 0001
80 h	1000 0000

↑

Result is positive and answer gives sign bit negative.