

# IFT 307

# Computer Organization and Architecture

Mr. Ibrahim Lawal

# Data Representation

- How numbers are stored on your computer.
- Write them as a string and get them back either as 0 or 1 which is a binary number and we call them bits.
- Bytes are represented as storage.
- If you have a 0 or 1 you can't use a decimal such as 1,2,3...9,10 = base 10.

Base 2 include

0,1,10,11,100,101,110,111.... Counting system.     . (binary point).

In the ten's digit it is the ten's digit, in the two's digit it is the two's digit i.e.  $b^2$  is 100 in base 10 and 4 in base 2.

# Integer

- ❑ Positive integers they can be represented in base 2. There are numbers of bits dedicated to them and if any is not in use just contains a zero.
- ❑ How many bits are in an integer? It depends on the computer you are using. Either a 32bit or a 64bit.

- ❑ Integers that are positive are also called unsigned
- ❑ Negative integers are signed since all integers are not signed. Another way it is done is signed/magnitude.
- ❑ Sign/magnitude e.g  $+10$ (sign and magnitude)

# Signed Magnitude

- ❑ The signed magnitude (also referred to as sign and magnitude) representation is most familiar to us as the base 10 number system.
- ❑ A plus or minus sign to the left of a number indicates whether the number is positive or negative as in  $+12_{10}$  or  $-12_{10}$ .

- ❑ In the binary signed magnitude representation, the leftmost bit is used for the sign, which takes on a value of 0 or 1 for ‘+’ or ‘-’, respectively.
- ❑ The remaining bits contain the absolute magnitude.

□ Consider representing  $(+12)_{10}$  and  $(-12)_{10}$  in an eight-bit format:

$$(+12)_{10} = (\textcolor{red}{0}0001100)_2$$

$$(-12)_{10} = (\textcolor{red}{1}0001100)_2$$



- ❑ The negative number is formed by simply changing the sign bit in the positive number from 0 to 1.
- ❑ Notice that there are both positive and negative representations for zero:  
**0**0000000 and **1**0000000.

- There are eight bits in this example format, and all bit patterns represent valid numbers, so there are  $2^8 = 256$  possible patterns.
- Only  $2^8 - 1 = 255$  different numbers can be represented, however, since  $+0$  and  $-0$  represent the same number.

# One's Complement

The one's complement operation is trivial to perform: convert all of the 1's in the number to 0's, and all of the 0's to 1's.

We can observe from the table that in the **one's complement** representation the leftmost bit is 0 for positive numbers and 1 for negative numbers, as it is for the signed magnitude representation.

<u>Decimal</u>	<u>Unsigned</u>	<u>Sign-Mag.</u>	<u>1's Comp.</u>	<u>2's Comp.</u>	<u>Excess 4</u>
7	111	–	–	–	–
6	110	–	–	–	–
5	101	–	–	–	–
4	100	–	–	–	–
3	011	011	011	011	111
2	010	010	010	010	110
1	001	001	001	001	101
+0	000	000	000	000	100
-0	–	100	111	000	100
-1	–	101	110	111	011
-2	–	110	101	110	010
-3	–	111	100	101	001
-4	–	–	–	100	000

3-bit Integer Representations

- ❑ This negation, changing 1's to 0's and changing 0's to 1's, is known as **complementing** the bits.
- ❑ Consider again representing  $(+12)_{10}$  and  $(-12)_{10}$  in an eight-bit format, now using the one's complement representation:
- ❑  $(+12)_{10} = (00001100)_2$
- ❑  $(-12)_{10} = (11110011)_2$

- Note again that there are representations for both  $+0$  and  $-0$ , which are  $00000000$  and  $11111111$ , respectively.
- As a result, there are only  $2^8 - 1 = 255$  different numbers that can be represented even though there are  $2^8$  different bit patterns.

- ❑ The one's complement representation is not commonly used.
- ❑ This is at least partly due to the difficulty in making comparisons when there are **two representations for 0**.
- ❑ There is also additional complexity involved in adding numbers.



## *Two's Complement*

- ❑ The two's complement is formed in a way similar to forming the one's complement: complement all of the bits in the number,
- ❑ but then add 1, and if that addition results in a carry-out from the most significant bit of the number, discard the carry-out.

The example below shows that in the **two's complement** representation, the leftmost bit is again 0 for positive numbers and is 1 for negative numbers.

❑ However, this number format does not have the unfortunate characteristic of signed-magnitude and one's complement representations: it has only one representation for zero.

□ To see that this is true, consider forming the negative of  $(+0)_{10}$ , which has the bit pattern:

$$(+0)_{10} = (00000000)_2$$

Forming the one's complement of  $(00000000)_2$  produces  $(11111111)_2$  and adding 1 to it yields  $(00000000)_2$ , thus  $(-0)_{10} = (00000000)_2$ .

- ❑ The carry out of the leftmost position is discarded in two's complement addition (except when detecting an overflow condition).
- ❑ Since there is only one representation for 0, and since all bit patterns are valid, there are  $2^8 = 256$  different numbers that can be represented.

- ❑ Consider again representing  $(+12)_{10}$  and  $(-12)_{10}$  in an eight-bit format, this time using the two's complement representation.
- ❑ Starting with  $(+12)_{10} = (00001100)_2$ , complement, or negate the number, producing  $(11110011)_2$ .

Now add 1, producing  $(11110100)_2$ , and  
thus  $(-12)_{10} = (11110100)_2$ :

$(+12)_{10} = (00001100)_2$	$(-12)_{10}$
$= (11110100)_2$	

□ There is an equal number of positive and negative numbers provided zero is considered to be a positive number, which is reasonable because its sign bit is 0.



□ The positive numbers start at 0, but the negative numbers start at -1, and so the magnitude of the most negative number is one greater than the magnitude of the most positive number.

- ❑ The positive number with the largest magnitude is  $+127$ , and the negative number with the largest magnitude is  $-128$ .
- ❑ There is thus no positive number that can be represented that corresponds to the negative of  $-128$ .

If we try to form the two's complement negative of -128, then we will arrive at a negative number, as shown below:

$$(-128)_{10} = (100000000)_2$$



$$01111111$$

$$\underline{+} \quad \underline{1}$$

$$(100000000)_2$$

THANK

YOU