CSE 351 Section 3 – Integers and Floating Point

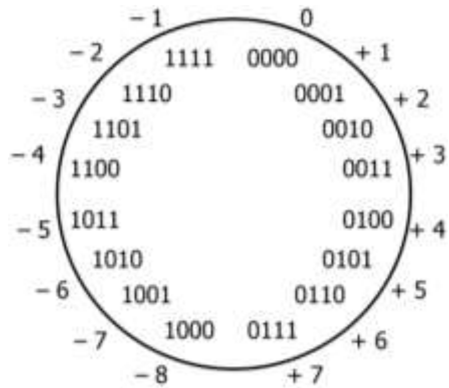
Welcome back to section, we're happy that you're here ☺

Signed Integers with Two's Complement

Two's complement is the standard for representing signed integers:

- The most significant bit (MSB) has a negative value; all others have positive values (same as unsigned)
- Binary addition is performed the same way for signed and unsigned
- The bit representation for the negative (additive inverse) of a two's complement number can be found by:

  <u>flipping all the bits and adding 1</u> (i.e. $-x = \sim x + 1$).



The "number wheel" showing the relationship between 4-bit numerals and their Two's Complement interpretations is shown on the right:

- The largest number is 7 whereas the smallest number is -8
- There is a nice symmetry between numbers and their negative counterparts except for -8

<u>Exercises</u>: (assume 8-bit integers)

1) What is the **largest integer**? The **largest integer + 1**?

<u>Unsigned</u>:	<u>Two's Complement</u>:

2) How do you represent (if possible) the following numbers: **39**, **-39**, **127**?

<u>Unsigned</u>:	<u>Two's Complement</u>:
`39:`	`39:`
`-39:`	`-39:`
`127:`	`127:`

3) Compute the following sums in binary using your Two's Complement answers from above. *Answer in hex.*

a. `39 -> 0b _ _ _ _ _ _ _ _`
`+(-39) -> 0b _ _ _ _ _ _ _ _`
`0x _ _ <- 0b _ _ _ _ _ _ _ _`
	b. `127 -> 0b _ _ _ _ _ _ _ _`
	`+ (-39)-> 0b _ _ _ _ _ _ _ _`
	`0x _ _ <- 0b _ _ _ _ _ _ _ _`

c. `39 -> 0b _ _ _ _ _ _ _ _`
`- 127 -> 0b _ _ _ _ _ _ _ _`
`0x _ _ <- 0b _ _ _ _ _ _ _ _`
	d. `127 -> 0b _ _ _ _ _ _ _ _`
	`+ 39 -> 0b _ _ _ _ _ _ _ _`
	`0x _ _ <- 0b _ _ _ _ _ _ _ _`

4) Interpret each of your answers above and indicate whether or not overflow has occurred.

a. `39+(-39)` Unsigned: Two's Complement:	**b.** `127+(-39)` Unsigned: Two's Complement:
c. `39-127` Unsigned: Two's Complement:	**d.** `127+39` Unsigned: Two's Complement:

Goals of Floating Point

Representation should include: [1] a large range of values (both very small and very large numbers), [2] a high amount of precision, and [3] real arithmetic results (*e.g.* ∞ and NaN).

IEEE 754 Floating Point Standard

The <u>value</u> of a real number can be represented in scientific binary notation as:

$$\textbf{Value} = (-1)^{\text{sign}} \times \text{Mantissa}_2 \times 2^{\text{Exponent}} = (-1)^S \times 1.M_2 \times 2^{E-\text{bias}}$$

The <u>binary representation</u> for floating point values uses three fields:

- **S**: encodes the *sign* of the number (0 for positive, 1 for negative)
- **E**: encodes the *exponent* in **biased notation** with a bias of $2^{w-1}-1$
- **M**: encodes the *mantissa* (or *significand*, or *fraction*) – stores the fractional portion, but does not include the implicit leading 1.

	S	E	M
`float`	1 bit	8 bits	23 bits
`double`	1 bit	11 bits	52 bits

How a `float` is interpreted depends on the values in the exponent and mantissa fields:

E	M	Meaning
0	anything	denormalized number (denorm)
1-254	anything	normalized number
255	zero	infinity (∞)
255	nonzero	not-a-number (NaN)

<u>Exercises</u>:

5) What is the largest, finite, positive value that can be stored using a `float`?

6) What is the smallest, positive, normalized value that can be stored using `float`?

7) Convert the decimal number 1.25 into single precision floating point representation:

8) What are the decimal values of the following `float`s?

0x80000000 0xFF94BEEF 0x41180000

Floating Point Mathematical Properties

- Not <u>associative</u>: $(2 + 2^{50}) - 2^{50}$!= $2 + (2^{50} - 2^{50})$
- Not <u>distributive</u>: $100 \times (0.1 + 0.2)$!= $100 \times 0.1 + 100 \times 0.2$
- Not <u>cumulative</u>: $2^{25} + 1 + 1 + 1 + 1$!= $2^{25} + 4$

<u>Exercises</u>:

9) Based on floating point representation, explain why each of the three statements above occurs.

10) If `x` and `y` are variable type `float`, give two *different* reasons why `(x+2*y)-y==x+y` might evaluate to false.

1EEE 754 Float (32 bit) Flowchart