Number Systems (Byte Representation)

How to represent integer and float values in memory.
A memory byte represents 2 nibbles or 8 bits. Each bit is a transitor that can have a high voltage or low voltage. We can represent the high state as a boolean true and the low state as a boolean false. We can also represent the high state as binary 1 and the low state as binary 0.

Let us start by defining data types

Primitive C Types	#Bytes	Bits	Range Base 2 Low	Range Base 2 High	Range Base 10 Low	Range Base 10 High	Notes
signed char	1	8	-2^7	2^7-1	-128	127	1st Bit is sign bit
unsigned char	1	8	0	2^8-1	0	255	
signed short	2	16	-2^15	2^15-1	-32768	32767	1st Bit is sign bit
unsigned short	2	16	0	2^16-1	0	65535	
signed int	4	32	-2^31	2^31-1	-2147483648	2147483647	1st Bit is sign bit
unsigned int	4	32	0	2^32-1	0	4294967295	

Range is directly related to the number of bits

float	4	32	3 Bytes, 24 bits represents the mantissa, 1 Byte, 8 bits represents the characteristic
double	8	64	53 bits represent the mantissa, 11 bits represents the characteristic

Note: Think of scientific notation. Mantissa means the Base 2 fraction
Characteristic means the Base 2 exponent

The floating data types require a little bit different definition and calculation with respect to range and accuracy.
Integers are exact, the floating types will always be in error by fractional differences in representing Base 10 and Base 2 numbers.

Example 1/3 = 0.3333..... requires an infinite number of digits to represent exactly in Base 10 but 1/3 is 1 digit in Base 3 → 0.1 Base 3!

float accuracy — 2^24 = 2^10 * 2^10 * 2^4 ~ 3 SD + 3 SD +1 SD ~ 7 SD where
SD is significant Base 10 digits, i.e. 3 SD represents 0 to 999

float range — 8 bits ~ +- 127 → (10^(+-X) = 2^(+-127) → X=+-38

So, a 4 byte, 32 bit float represents approximate 7 significant digits of accuracy with a range of 10^(+-38)

double — Same calculations can be done with a double providing 16 significant digits of accuracy with a range of 10^(+-308)
Just use 2^53 for the accuracy and 11 bits or +-2^10 for the range

Negation in Bits and Bytes

The theory for negation was developed by Kurt Hensel in 1897 and are referenced as *p-adic* numbers.

In computer science we refer to it as 2's complement!

Whereas, 1/3 has an infinite sequnce to the right of the decimal point 0.333333..... out to infinity.
P-adic numbers are an infinite sequence to the left of the decimal using as many bits/bytes for the width of the data type.

A simple example should suffice.

89 Base 10 = 59 Base 16 = 01011001 Base 2 = 131 Base 8

	Base 2 Representation bit for bit				# Bytes	Hex Representation
89 Base 10 =				01011001	1	59
89 Base 10 =			00000000	01011001	2	0059
89 Base 10 =	00000000	00000000	00000000	01011001	4	00000059

How do we represent -89 ?

89 Base 10 =	01011001		
1's Complement =	10100110		Just flip the bit 1's to 0 and 0's to 1
	+1		Add 1 to get the 2's complement.
2's Complement =	10100111	=	-89 Base 10

Therefore, using the 2's Complement for any number of bits gives

-89 Base 10 =				10100111	1	A7
-89 Base 10 =			11111111	10100111	2	FFA7
-89 Base 10 =	11111111	11111111	11111111	10100111	4	FFFFFFA7

Note: The sign bit, which is the left most bit is always 1 for a negative number when the datatype is a signed number representation.
No matter how large the data type, the left most digits will be 1 extending to infinity, no matter how small the number.
Also, if you add 89 + -89 in Base 10 or Base 2 you will get 0. In the Base 2 bit representation, the last carry 1 bit drops off.

Representing a floating point value

A definition for a floating point 4 byte, 32 bit number using Base 2 → 1's and 0's

SMMMMMMM MMMMMMMM MMMMMMMM SCCCCCCC

Let S be the sign, positive if 0 and negative if 1
Let M be the mantissa, or decimal representation in Base 2 scientific notation
Let C be the characteristic, or exponent of the power of 2 in scientific notation
The same number in Scientific notation Base 2 would look like this

S .MMMMMMMMMMMMMMMMMMMMMMM x 2^(S CCCCCCC)

Let us use a previous example

1023.60546875_{10} = $3FF.9B_{16}$ = 1111111111.10011011_{2} = 1777.466_{8}

+ .102360546875 x 10^(+4) = + .3FF9B x 16^(+3) = + .111111111110011011 x 2^(+10) = +.1777466 x 8^(+4)

Then, to place this into our above definition we have,

	← Mantissa →			Characteristic
Binary	01111111	11110011	01100000	00001010
Hex	7F	F3	60	0A

Note: This is a very unusual case. Normally a base 10 number like this having 12 significant digits can't be represented in a 4 Byte float like this. However, this is really a Base 2 number which takes 18 bits easily fitting into the 23 bits allowed for this type of float. If a number can't be represented exactly, which is the rule not the exception, we truncate and accept the error.