IEEE754 Format

A way to represent very large or very small numbers precisely using scientific notation in binary form

A common standardized format to do this is called the "Institute of Electrical and Electronic Engineers Standard 754" or **IEEE754** format.

This comes in three forms, all of which are very similar in procedure:

- single precision (32-bit)
- double precision (64-bit)
- extended precision (80-bit)

Our example will use single precision to represent the number -0.09375 in IEEE754 Format.

Single precision formatted numbers contain 32 bits, where

- the first bit is use to designate the sign of the number – 0=positive, 1=negative (the "sign bit")
- the next 8 bits are designated for the exponent to represent very large or very small numbers. (the "exponent part")
- the final 23 bits are for the *mantissa* – that is, the digits found to the right side of the decimal point.

What about the digits found to the left side of the decimal point?

This will always be nonzero (and therefore equal to one) when a binary number is in scientific notation. Since this is always known to be a 1, it can be left out of the binary representation.

A. Is your number negative or positive? If positive, the first bit will be a 0; if negative, the first bit will be a 1.

B. Exponents are stored in "**EXCESS 127 FORM**"
   1. Count the number of places the binary point needs to be moved until a single digit of 1 sits by itself on the left side of the binary point. If the number you are representing in a large one, this count will be positive; if the number you are representing is a small one, this count will be negative.
   2. Add 127 to your result above.
   3. Note that, since 8-bit binary numbers can range from 0 to 255, exponents in single precision format can range from -126 to +127, that is from 2^{-126} to 2^{127} or, approximately, 10^{-38} to 10^{38} in size. In "*excess 127 form*" negative exponents range from 0 to 126, and positive exponents range from 128 to 255. The missing exponent, 127, is the one right in the middle and represents a power of zero.

4. Now translate your sum (which will always be a positive value after adding 127) into binary form.
5. This should be represented using 8 bits, so zeros may need to be adjoined to the left side to ensure a string length of 8 bits.
6. This 8-bit string represents the exponent.

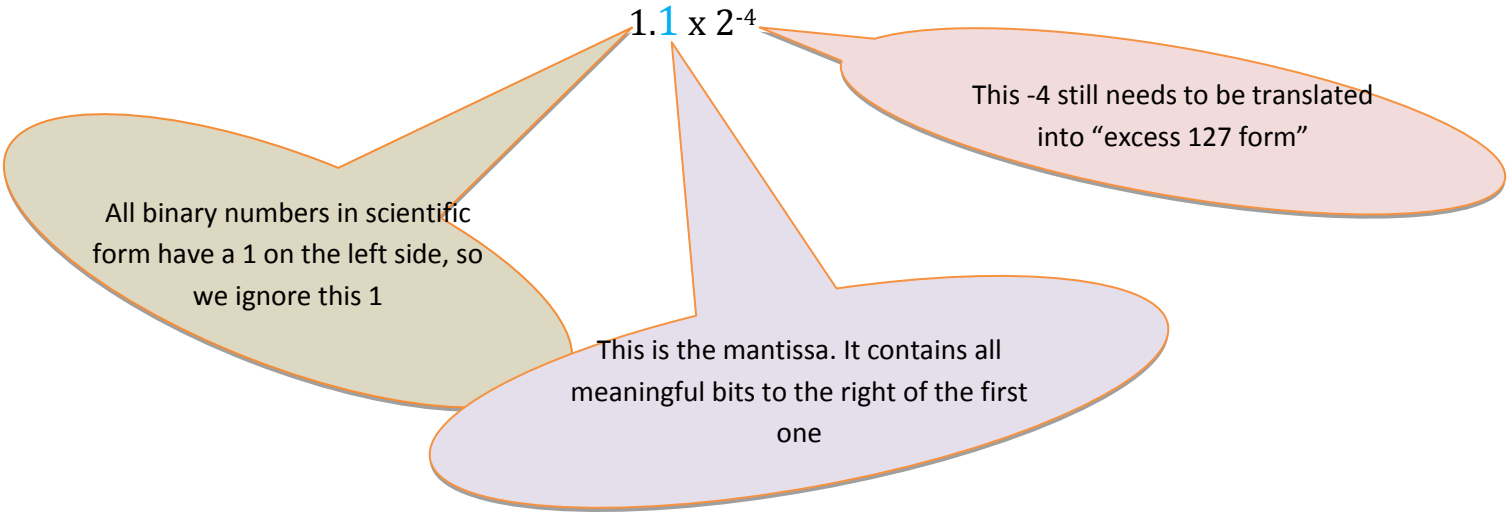
An Example: Represent -0.09375 in IEEE754 format.

The sign is negative, therefore our number will start with a 1.

Now, ignoring the sign, convert 0.09375 to binary:

	Integer Part	
0.09375 x 2 = 0.1875	0	↓ (remember, read downwards)
0.1875 x 2 = 0.375	0	
0.375 x 2 = 0.75	0	
0.75 x 2 = 1.50	1	
0.50 x 2 = 1.00	1	Therefore, $0.09375_{10} = 0.00011_2$

Next, rewrite this binary number in scientific form by moving the binary point to the right four places. Since the number we are formatting, –0.09375, is small, this will be a negative 4.



Now rewrite -4 in “excess 127” form: -4 + 127 = 123
and $123_{10} = 1111011_2$, but since this contains only 7 bits, we rewrite it with 8:

01111011

Hence,
-0.09375 will be represented in IEEE754 format as

1 01111011 10000000000000000000000

Sometimes this is written as a hexadecimal number, hence

1011	1101	1100	0000	0000	0000	0000	0000
B	D	C	0	0	0	0	0

or $BDC00000_{16}$.

Now you try some:

Represent each of the given base ten numbers in IEEE754 format.

a. 5.1
b. -14.25
c. 6.425

Now you try some: Answers

a. $0|10000001|01000110011001100110011_2$
   $40A33333_{16}$
b. $1|10000010|11001000000000000000000_2$
   $C1640000_{16}$
c. $0|10000001|10011011001100110011001_2$
   $40CD9999_{16}$