Floating Point Numbers

Fixed-point methods of storing numbers will always require a trade-off between magnitude and precision.

There is also the problem of very large or very small numbers, like:

```
1110110101000000000000000000000000000000000000000000000000000000000000000.0
```

We might need to use several consecutive memory locations to hold this number, and most of the space would be wasted, containing nothing but zeroes.

It would be much better to store the significant part of the number, the 1110110101 at the left hand end, in a memory location, and then store a count of the number of bits in another memory location. There are 73 digits, to the left of the binary point. So we could store the number above like this:

0	**0.1110110101**	**01001001**
Sign Bit	**Significand**	**Exponent**

So in binary, a floating point number is expressed in three parts.

Sign bit This is normally set to 0 for a positive number, and 1 for a negative number.

Significand or **Mantissa** The significant digits. This is usually stored as a fraction between 1/2 and 1. With the binary point on the left of the most significant 1, this allows for maximum precision. This is called Normalised form.

Exponent This is a count of the number of places the binary point has been shifted in order to get it to the left hand end of the significand. I'm using an 8 bit two's complement number in the example above.

Similarly, the number:

```
0.0000000000000000000000000000000000000000000000000000000000000110101011101
```

could be stored as follows:

0	**0.110101011101**	**11000011**
Sign Bit	**Significand**	**Exponent**

Note that the 8 bit 2's complement exponent is the code for negative 61, which indicates that the binary point has been shifted to the right in order to get it to the left hand end of the significand.

There are many ways of representing floating point numbers. Some may use two's complement for both significand and exponent, some may use a sign bit for significand, but two's complement for the exponent.

Also, the number of bits allocated to the significand and the exponent may vary from processor to processor.

Generally speaking, there tend to be two or three times as many bits allocated to the mantissa as to the exponent.