



1.4.1 Floating Point Binary

Floating Point: Normalisation

As the binary point in floating point form is able to move left and right depending on the exponent, there are multiple different representations of the same decimal number.

The decimal 6.25 represented as a 12-bit floating point number with an 8-bit mantissa and a 4-bit exponent both in two's complement in three different formats.

Mantissa	Exponent
0 1 1 0 • 0 1 0 0	0 0 0 0

Mantissa	Exponent
0 1 • 1 0 0 1 0 0	0 0 1 0

Mantissa	Exponent
0 1 1 0 0 0 1 • 0	0 1 0 1

It is not obvious at first that these different floating point numbers all represent the same decimal value. It is therefore helpful to have a normalised format to store the floating point numbers so that equivalence comparisons are easier.

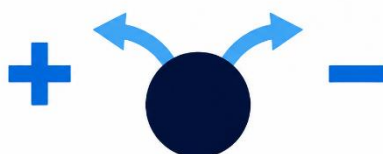
Advantages of Normalisation

- Allows for **more precision** for the given number of bits.
- Ensure each binary value has a **unique representation**.

Moving the decimal point and exponents

To move the point **left**:
Add to the exponent

To move the point **right**:
Subtract from the exponent





Normalised floating point numbers always start with the first change between 0 and 1. Either 01 or 10.

Example

Below is a 12-bit floating point number with an 8-bit mantissa and a 4-bit exponent both in two's complement. It is **not** in normalised form.

Mantissa	Exponent
0 0 1 0 0 0 ● 0 0	0 0 1 0

The binary point needs to be moved to the first change between 0 and 1.

Mantissa	Exponent
0 0 ● 1 0 0 0 0 0	0 1 1 0

The point needs to be moved 4 places to the left. To do this we add 4 to the current value of the exponent.

A positive normalised mantissa must start with the 01.

Any extra 0s to the left are discarded. Any extra spaces on the right of the mantissa are filled with 0s.

Mantissa	Exponent
0 ● 1 0 0 1 0 0	0 1 1 0





Floating Point: Addition

In order to add to floating point numbers:

- Convert the exponent of both numbers zero by moving each binary point.
- Line up the binary points.
- Add the mantissas together.
- Normalise the resulting number.

Example

Add the following floating point numbers, each with an 8-bit mantissa and 4-bit exponent which are both in two's complement.

0.0000110 0011

1.0010000 1110

The exponent of the first number is 3. To make it 0 we need to subtract 3 so we move the decimal 3 to the right.

The exponent of the second number is -2. To make it 0 we need to add 2 so we move the decimal 2 to the left.

0000.1100 0000

1.110010000 0000

Now line up the binary points and add the two numbers together:

$$\begin{array}{r} 0 . 0 1 1 0 0 \\ 1 . 0 1 1 0 1 + \\ \hline 1 . 1 1 0 0 1 \\ 1 \quad 1 \end{array}$$

Lastly we need to normalise this result into normalised form in the same format with 8-bit mantissa and 4-bit exponent.

The point needs to move 2 places to the right, so we add 2 to the exponent. Then remove any extra digits at the start of the binary number and pad out the end with 0s so that it is in the required format.

1.0010000 0010

