FLOATING POINT NUMBERS

We learnt about how binary numbers can be stored in a [ ] *representation. The magnitude of the numbers stored depends on the number of bits used. For example, 8 bits allowed a range of* [ ] *(using two's complement representation) whereas* [ ] *increased this range to −16 384 to* [ ]

However, this type of representation limits the range of numbers and does not allow for fractional values. To increase the range, and to allow for fractions, we can look to the method used in the [ ] *number system. For example:*

312 110 000 000 000 000 000 000 can be written as 3.1211 × [ ]

Using [ ] *notation. If we adopt this system in binary, we get:*

$$M \times 2^{E}$$

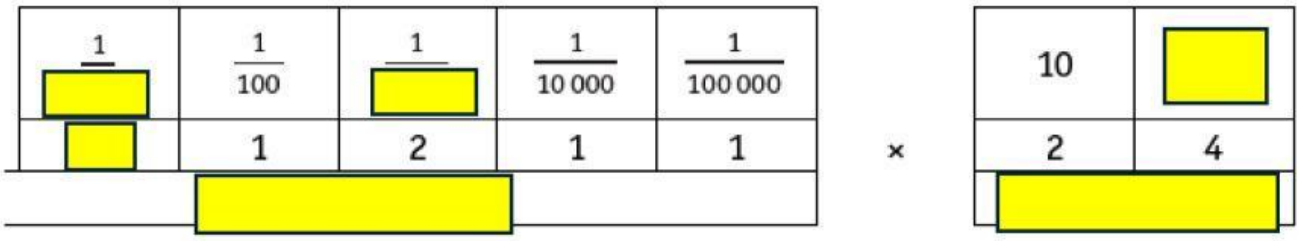
M is the [ ] *and E is the* [ ]

This is known as [ ] ***representation.***

In our examples, we will assume a computer is using 8 bits to represent the mantissa and 8 bits to store the exponent (a binary point is assumed to exist between the first and second bits of the mantissa). Again, using denary as our example, a number such as

$$0.31211 \times 1024$$

means.



Drag and Drop the answer from the box here:

Exponent	**1 0 2 3**	**scientific**	**Mantissa**
Fixed-point	**Binary floating-points**	**denary**	***+16 383***
1	**1000**	**3**	**10**
16 bits	**Mantissa value**	***−128 to +127***	**Exponent value**

Precision versus range

The following values relate to an 8-bit mantissa and an 8-bit exponent (using two's complement):

The [ ] **positive number** which can be stored is:

0	1	1	1	1	1	1	1

0	1	1	1	1	1	1	1

≡ [ ]

The **smallest** [ ] **number** which can be stored is:

0	1	0	0	0	0	0	0

1	0	0	0	0	0	0	0

≡ [ ]

The [ ] **magnitude negative number** which can be stored is:

1	0	1	1	1	1	1	1

1	0	0	0	0	0	0	0

≡ [ ]

The **largest magnitude** [ ] **number** which can be stored is:

1	0	0	0	0	0	0	0

0	1	1	1	1	1	1	1

≡ [ ]

Drag and Drop the answer from the box here:

largest	$\frac{127}{128} \times 2^{127}$	**positive**	$-\frac{65}{128} \times 2^{-128}$
smallest	-1×2^{127}	**negative**	$\frac{1}{2} \times 2^{-128}$

Floating-point problems

Problem 1:

The storage of certain numbers is an approximation, due to limitations in the size of the _____. This problem can be minimised when using programming languages that allow for _____ precision and _____ precision.

Problem 2:

If a calculation produces a number which exceeds the _____ possible value that can be stored in the mantissa and exponent, an _____ error will be produced. This could occur when trying to _____ by a very small number or even 0.

Problem 3:

When dividing by a very large number this can lead to a result which is less than the _____ number that can be stored. This would lead to an _____ error.

Problem 4:

One of the issues of using normalised binary floating-point numbers is the inability to store the number _____. This is because the mantissa must be _____ or 1.0 which does not allow for a zero value.

Drag and Drop the answer from the box here:

overflow	zero	underflow	mantissa	maximum
0.1	divide	double	smallest	quadruple