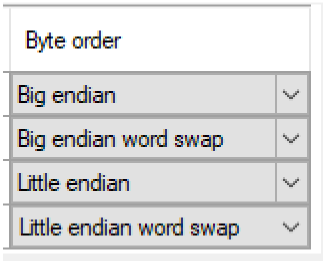
Understanding Floating point – big & little endian

Big endian vs. little endian

(source: https://whatis.techtarget.com/definition/)

Big-endian is an order in which the "big end" (most significant value in the sequence) is stored first (at the lowest storage address). Little-endian is an order in which the "little end" (least significant value in the sequence) is stored first. For example, in a big-endian computer, the two bytes required for the hexadecimal number 4F52 would be stored as 4F52 in storage (if 4F is stored at storage address 1000, for example, 52 will be at address 1001). In a little-endian system, it would be stored as 524F (52 at address 1000, 4F at 1001).
For people who use languages that read left-to-right, big endian seems like the natural way to think of a storing a string of characters or numbers - in the same order you expect to see it presented to you. Many of us would thus think of big-endian as storing something in *forward* fashion, just as we read.

Byte order



	value	0 x 40866665		
Address	big endian memory	big endian word swap memory	little endian memory	little endian word swap memory
0x1	40	66	65	86
0x2	86	65	66	40
0x3	66	40	86	65
0x4	65	86	40	66
0x5				

Bit, Byte and Word

A byte is a data measurement unit that contains eight bit
Word is a combination of bits and can be 8,16,32 but also 25 bit, actually this will be determined by the computer architecture. Often a word contain 16 bit and a longword refer to 32 bit notation

Floating point calculation (32 bit)

Floating point notation is introduced to store any number in a predefined format. In our case we focus only on 32 bit.

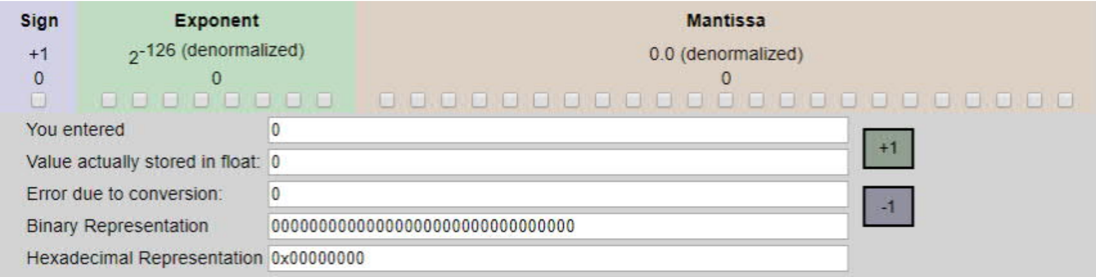
The value of a IEEE-754 number is computed as:

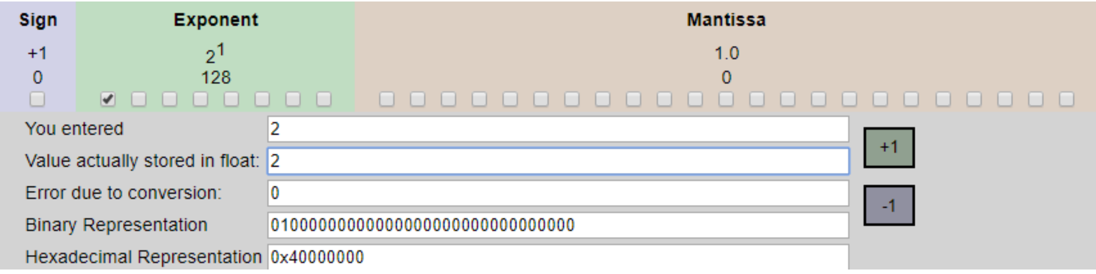
sign $\mathbf{2}^{\mathbf{exponent}}$ mantissa

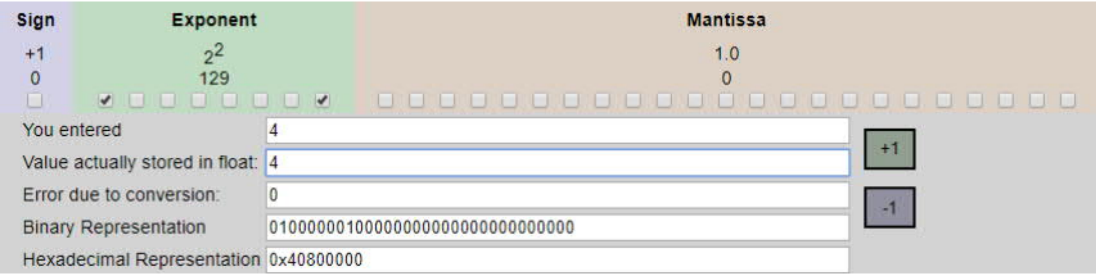
The sign is stored in bit 32. The exponent can be computed from bits 24-31 by subtracting 127. The mantissa (also known as significand or fraction) is stored in bits 1-23. An invisible leading bit (i.e. it is not actually stored) with value 1.0 is placed in front, then bit 23 has a value of 1/2, bit 22 has value 1/4 etc. As a result, the mantissa has a value between 1.0 and 2.

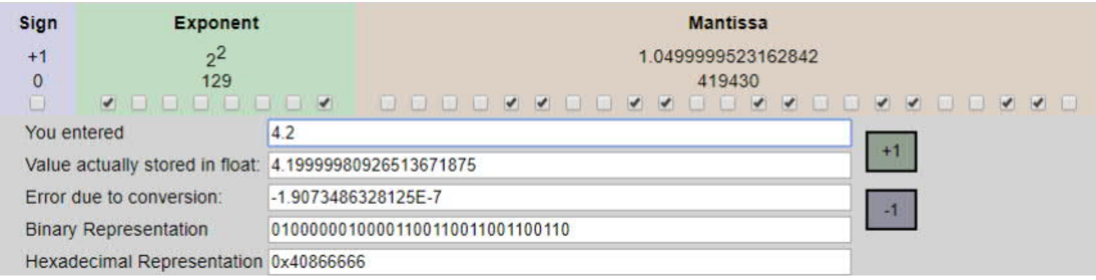
Some examples

Source: https://www.h-schmidt.net/FloatConverter/IEEE754.html









Value = $2^2 \times 1.0499 \approx 4.2$

Explanation how to convert 4.2

Register	Decimal value	Hexadecimal	Hexadecimal combined	Float 32 bit
1	26214	6666	40866666	4.199999809
2	16518	4086		

Hexadecimal 0x40866666 is in binary format:
01000000100000110011001100110

DEC

big endian

Register	Decimal value	Hexadecimal		Hexadecimal combined	Float 32 bit
1	26214	6666	6666	40866666	4.199999809
2	16518	4086	4086		

0 10000001 00001100110011001100110

Sign	0	Pos
Exponent	10000001	2
Mantissa	00001100110011001100110	1.0500

$2^{(2)}$ ✖ 1.0500 = 4.199999809