0.4 + 0.3 = 0.70000005? [Andy Chong Sam]

Floating point numbers are not exact representations of decimals containing fractional amounts. For instance, in Java the code 0.3f + 0.4f produces the unusual result 0.70000005. On this article we'll try and derive this result without necessarily taking a deep dive into hardware or Java implementation.

(II) A rounding system utilizing a (G)uard, (R)ound, and (S)ticky bit will be used. In this system, we'll assume that the least significant bit in the mantissa will be the guard bit. The round bit is the one after guard, and the sticky bit is the OR result of all remaining bits. Here are the rules:

GRS	Round?
000	no
100	no
010	no
110	yes
011	yes
111	yes

(III) In binary, 0.4 is:
0 01111101 10011001100110011001101

The first cluster is 0, it's positive. For the exponent,$(01111101)_2 = (125)_{10}$.The mantissa to 27 bits, with bit 23 (G) in bold is:

$$10011001100110011001\mathbf{0}1100$$

Using the rules from section II we round the mantissa to 23 bits:

$$10011001100110011001101$$

In scientific notation, the stored float is:

$$1.10011001100110011001101 \times 2^{125}$$

(IV) In binary, 0.3 is:
0 01111101 00110011001100110011010

The exponent is 125, the mantissa shown up to 27 bits is, with the (G) bit in bold:

$$00110011001100110011001\mathbf{1}1001$$

Using the established rules we round this to:

$$00110011001100110011010$$

Written in scientific notation:

$$1.00110011001100110011010 \times 2^{125}$$

(V) The next step is to add the mantissas from steps III and IV together (along with the assumed 1), which is the result of 0.3f + 0.4f:

$$\begin{aligned} &1.10011001100110011001101 \\ +&1.00110011001100110011010 \\ \hline &10.11001100110011001100111 \end{aligned}$$

In scientific notation this result is:

$$10.11001100110011001100111 \times 2^{125}$$

After normalizing...

$$1.01100110011001100110011 \times 2^{126}$$

Let's see how it's stored. The exponent is now 01111110, the binary representation of 126. We round based on the bolded (G) bit:

0 01111110 01100110011001100110011
0 01111110 01100110011001100110010

(VI) We use the formula $d = (-1)^s(1+m)\ 2^e$ to verify the stored value. The calculation is shown in full on the next page but we find that this gives us approximately **0.700000048**... close enough!

For the formula $d = (-1)^s(1 + m)\ 2^e$...

- $s = 0$ since the number is positive.
- m is the decimal representation of the mantissa 01100110011001100110 11010
- $e = -1$. The value of e is derived by subtracting 127, the bias, from the exponent 126

Finally the calculation itself:

$$(-1)^0(1 + (2^{-2} + 2^{-3} + 2^{-6} + 2^{-7} + 2^{-10} + 2^{-11} + 2^{-14} + 2^{-15} + 2^{-18} + 2^{-19} + 2^{-21}))(2^{-1}) \approx 0.700000048$$

When rounded to include only six places after the decimal we get 0.70000005.