

Discussion notes on floating point

Bert Maher
February 26, 2009

1 Binary fractions

1.1 Binary to decimal

Interpret the n^{th} location past the binary point as 2^{-n} :

$$11.101_2 = 2^1 + 2^0 + 2^{-1} + 2^{-3} = 3.625_{10}$$

1.2 Decimal to binary

Repeatedly multiply by 2 and “shift off” the leading ones and zeros that result, and reading top to bottom:

Example: Convert 0.1875_{10} to base 2.

0.1875	0.
0.375	0
0.75	0
1.5	1
1.0	1

$$0.1875_{10} = 0.0011_2$$

Example: Convert 0.2_{10} to base 2.

0.2	0.
0.4	0
0.8	0
1.6	1
1.2	1

$$0.2_{10} = 0.\overline{0011}$$

1.3 Binary scientific notation

In decimal scientific notation, numbers are normalized such that there is a single leading non-zero digit, e.g., 6.02×10^{23} .

In binary scientific notation, we normalize such that there is a single leading non-zero bit, e.g., 11.0101×2^{37} . Notice that, for non-zero numbers, the leading bit will *always* be a 1, which is important for our floating point format.

2 IEEE floating point formats

2.1 Single precision, 32-bit

Single precision floating point uses:

- 1 bit sign
- 8 bit exponent, encoded in excess-127 format
- 23 bit fractional component of mantissa, where a leading 1 is implicit

The bits are arranged as follows:

<i>sign</i> ₁	<i>exp</i> ₈	<i>mant</i> ₂₃
--------------------------	-------------------------	---------------------------

2.2 Double precision, 64-bit

Double precision floating point uses:

- 1 bit sign
- 11 bit exponent, encoded in excess-1023 format
- 52 bit mantissa (with implicit 1)

The bits are arranged as follows:

<i>sign</i> ₁	<i>exp</i> ₁₁	<i>mant</i> ₅₂
--------------------------	--------------------------	---------------------------

2.3 Floating point to decimal

To find the value encoded by a single precision floating point representation, apply the formula:

$$(-1)^{sign} \times 2^{exp-127} \times 1.mant$$

Again, note the implicit one before the mantissa! This is done because (as noted above) numbers encoded in binary scientific notation always have a leading one, as a result of normalization.

Example: 0 1000001 01010000000000000000

$$\begin{aligned} sign &= 0 \\ exp &= 129 - 127 = 2 \\ mant &= 1.0101 \end{aligned}$$

$$(-1)^0 \times 2^2 \times 1.0101 = 101.01_2 = 5.25_{10}$$

2.4 Decimal to floating point

To convert decimal to single precision floating point, e.g., 3.5:

1. Convert to binary: $3.5_{10} = 11.1$
2. Normalize/convert to scientific notation: $11.1 = 1.11 \times 2^1$
3. Convert exponent to excess-127: $1 \rightarrow 1 + 127 = 128_{10} = 10000000_2$
4. Drop the leading one from the mantissa: $1.11 \rightarrow 11$
5. Concatenate the bits: 010000000110...
6. (Optional) For a more compact representation, write the binary string as hexadecimal: `0x40600000`

2.5 Denormalized numbers

In order to represent 0, we must, in some cases, eliminate the implicit 1 in the mantissa, thus allowing *denormalized* numbers. This case occurs when the exponent is all zeros. In this case we (1) do not add an implicit 1, and (2) treat the exponent as -126 (single precision) or -1022 (double precision)¹. So denormalized numbers are interpreted as:

$$v = (-1)^{sign} \times 2^{-126} \times 0.mant$$

Note that because floating point numbers use sign-magnitude format, there are both positive and negative zeros; for most cases these are considered equal.

2.6 Special cases

There are a few special cases that have “non-numeric” interpretations. These examples are for single precision, but can be extrapolated to double.

- $+\infty$: $sign = 0$, $exp = 0xFF$, $mant = 0$
- $-\infty$: $sign = 1$, $exp = 0xFF$, $mant = 0$
- NaN: $exp = 0xFF$, $mant \neq 0$

¹The exponent is one greater than its excess-N value so that the denormalized numbers are evenly spaced between 0 and the smallest normalized number (2^{-126} in single precision)