Discussion #6: Floating Point

Written by Justin Hsia (7/8/2011)

Floating Point Representation

Written like scientific notation: $(-\mathbf{1})^{\mathbf{S}} \times (\mathbf{1} + \mathbf{Significand}) \times \mathbf{2}^{(\mathbf{Exponent-bias})}$

`float -`	`S`	`exponent (8)`	`significand (23)`
`double -`	`S`	`exponent (11)`	`significand (20)`
	`significand continued(32)`		

The Sign Bit

The same as the sign bit in sign and magnitude. `0` means positive, `1` means negative.

The Exponent

The exponent is encoded in biased notation. What this means in practice is that you read the exponent as an unsigned integer and then subtract the bias value (127 for `float`, 1023 for `double`). The reasoning behind this is that we want easy comparisons of the exponents without having to load the exponent into a separate place and decode it as a two's complement (want comparisons to work "in place").

It is useful to notice that the biases are represented exactly by a zero followed by all ones (so `0b0111 1111` for `float` and `0b011 1111 1111` for `double`). This way we can represent about the same number of positive and negative exponents.

For example, if we want $8 = 1 \times 2^3$ in single precision, we need to set exponent to 3+127 = 130 = `0b1000 0010`. For double precision, we need exponent = 3+1023 = 1026 = `0b100 0000 0010`.

As we will see, the exponent can be thought of as a shift operation on the Significand.

The Significand and Normalized Numbers

This tends to be the most confusing part of floating point representation. So let's go back to what we do know in the world of decimals. We have what is known as the decimal point ('.'), which marks the place where we cross into negative exponents of our base.

- How do we write out the following number in base notation?

  $1.234_{10} = 1 \times 10^0 + 2 \times 10^{-1} + 3 \times 10^{-2} + 4 \times 10^{-3}$

- How do we convert the following number into scientific notation?

  $314.159 = 3.14159 \times 10^2$ (notice that the decimal point shifted left by the exponent value)

When it comes to integers, we read from right to left with unspecified digits to the left assumed to be leading zeros. For the significand, the opposite is true – we read from left to right with unspecified digits to the far right assumed to be trailing zeros.

Now let's apply these thoughts to binary (base two). Now the period is called the binary point:

- Convert the following from binary point to decimal point:

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

- Convert the following into scientific notation:

  $0.001101_2 = 1.101_2 \times 2^{-3}$ (notice again the binary point shifted left by the exponent value)

Normed numbers (short for "normalized") are the main form of representation in floating point. A normed number essentially means that it is in proper scientific format – there is only a single non-zero digit immediately to the left of the point. In binary, the only possible digits are 0 or 1, which means there must be a one there. So the significand portion of the floating point number is often written $(1.\mathrm{xxx}\ldots)$, where the x's are the bits of the significant from left to right. The leading 1 is implicit and not found in the floating point representation.

Special Numbers

Floating point allows for special numbers. These are obtained using specific combinations of the exponent and significand as follows (the table is for single precision, for double precision substitute 2047 for 255):

Exponent	Significand	Meaning
0	0	± 0
0	non-zero	Denorm number
1-254	anything	Normed number
255	0	$\pm\infty$
255	non-zero	NaN

As previously mentioned, the vast majority of floating point numbers fall under normed numbers. A denormalized number does not have the implicit leading 1. This allows for the attainment of smaller numbers (closer to 0). NaN stands for Not-a-Number.

Note: The exponent for denorms is actually not $0 - 127 = -127$ as you might expect from the exponent field. The smallest normal number you can represent is 2^{-126} (`0x00800000`) and a denorm number is of the form $2^{\mathrm{e}} \times (0.\mathrm{xxxx}\ldots)$, so to get 2^{-127}, you need $\mathrm{e} = -126$ for denorm numbers.

Truncation

Because the significand has a limited width, floating point operations sometimes cause truncation, leading to rounding that can sometimes significantly affect operations. Looking at just the significand, the bits represent, from left to right, the following numbers:

bit position:	22	21	20	…	2	1	0
number:	2^{-1}	2^{-2}	2^{-3}	…	2^{-21}	2^{-22}	2^{-23}

Assuming your exponent $\mathrm{e} = (\mathrm{Exponent} - \mathrm{bias})$ has been chosen, then the bits of the significand actually represent the following numbers:

bit position:	22	21	20	…	2	1	0
number:	2^{e-1}	2^{e-2}	2^{e-3}	…	2^{e-21}	2^{e-22}	2^{e-23}

This means that any number smaller than 2^{e-23} cannot be represented within the significand and will be truncated.

Check: Let `num = 0x4C000000` be a floating point number. What positive integers will get truncated (that is, find all `x>0` such that `num + x = num` in floating point arithmetic)?

Does your answer change if `num = 0x4C011111`? (same exponent, different significand)

Can truncation occur with subtraction? With numbers of different signs (1 pos, 1 neg)?

Number Conversion Practice

Converting from floating point to decimal is pretty straight-forward. Copy the bits of the significand after "1." for normed numbers and after "0." for denormed numbers. Calculate the exponent and then shift the binary point appropriately. Then convert the number using standard base notation rules.

Converting from decimal to floating point involves the opposite procedure. First you need to split the number into the sum of different powers of two and write the number in binary point notation. Find the exponent value necessary to shift your number into normed form and then copy the trailing digits into the significand. An example is shown below, followed by some extra practice problems:

- Convert 354.375 to floating point notation.
  1) Here $354.375 = 256 + 64 + 32 + 2 + 0.25 + 0.125 = 2^8 + 2^6 + 2^5 + 2^1 + 2^{-2} + 2^{-3}$ $= 101100010.011_2$
  2) To normalize, we get $101100010.011_2 = 1.01100010011_2 \times 2^8$, so $\mathrm{Exponent} = 135 = \mathrm{0b10000111}$.
  3) Now using $\mathrm{Sign} = 0$, and $\mathrm{Significand} = \mathrm{0b011000100110} \ldots 0$, we get:
  4) $354.375 = \mathrm{0b0|100\ 0011\ 1|011\ 0001\ 0011\ 0000\ 0000\ 0000} = \mathbf{0x43B13000}$

Check: Convert from float to decimal: `0xBFD00000.`

Convert from decimal to floating point: 8.25

Convert to float (use denorm): -2^{-130}