Computer Architecture
Floating Points 1

Cain Susko

February 4, 2022

Queen's University
School of Computing

Fractional Binary Numbers

We know we must be able to represent numbers that are not just integers for things like physics or calculus that heavily rely on computers to do useful computations. We can represent a decimal number like so:

$$10110.1001$$

Using a binary point '.'
if the number k is the digits position relative to the ones position (right of binary point) then the position represents the value 2^k. The actual number N represented by a solely Fractional number is:

$x =$ the number represented by the binary string if it were not Fractional
$k =$ the length of x
$N = \frac{x}{2^k}$

value	*representation*
5. 3/4	101.11
2. 7/8	10.111
1. 7/16	1.0111

the limitations of this form are that

1. One can only enter numbers of the exact form $x/2^k$
2. If we use this binary point in a set number of bits w then we will limit thenumbers we can represent as we are essentially cutting our range proportionally to the position of the binary point

IEEE Floating Point standard 754

This standard was established in 1999 to create a uniform representation for floating point numbers and their arithmetic. Now adays most modern CPUs use this standard. It is driven by numerical concers and is thus very good at handling rounding, overflow, and underflow.
Unfourtunately it is hard to make fast hardware using this standard, although the hardware can be very comprehensive. an IEEE float is represented as:

$$(-1)^s M * 2^E$$

such that
s determines whether the number is negative or positive
M is normally a fractional value from $[1.0, 2.0)$
E weights the value M by a power of 2

This form can then be represented in 3 different bit formats, or precisions. These differ by the number of bits allocated to each part of the float representation

single precision	$s = 1 \;\; M = 23 \;\; E = 8$
double precision	$s = 1 \;\; M = 52 \;\; E = 11$
extended precision	$s = 1 \;\; M = 63, 64 \;\; E = 15$

Where all numbers are measured in **bits**

Furthermore there are 3 types of values that we can represent with the IEEE format. They are Normalized, De Normalized, and Special Values.

Normalized

The value is normalized if the exp $\neq$ 000...0 and exp $\neq$ 111...1. $2^{k-1} - 1$ where k is the number of exp bits
A normalized value also has an E such that $E = exp - bias$
A normalized value also has a M value such that M has a leading one for example. $M = 1.xxxxx...x_2$ note M is binary
For example, the float $F = 25213.0$

$$\begin{aligned} 15213.0 &= 11101101101101_2 \\ &= 1.1101101101101_2 \times 2^{13} \end{aligned}$$

The significand M equals

$$M = 1.1101101101101_2$$
$$frac = 11011011011010000000000_2$$

The exponent E equals

$$\begin{aligned} E &= 13 \\ bias &= 127 \\ exp &= 140 = 1000100_2 \end{aligned}$$

Thus the result is:

$$0\ 1000100\ 11011011011010000000000$$

where s is the fisrt portion, exp is the second, and $frac$ is the third

Denormalized

a float is this type of value if:

$$\begin{aligned} \text{exp} &= 000\ldots00 \\ E &= 1 - bias \\ M &= 0.xxx\ldots \\ frac &= 000\ldots00 \wedge \neq 000\ldots00. \end{aligned}$$

When $frac = 000\ldots$ the number being represented is a zero value, $+0, -0$ When $frac \neq 000\ldots$ the number being represented are the numbers closest to 0. The latter values are equispaced.

Special Values

when $\text{exp} = 111\ldots11$, this value is known as a special value. There are 2 cases:
When $frac = 000\ldots00$ this represents the value ∞, both negative and positive. This operation is one that overflowed. For example:

$$\frac{1.0}{0.0} = \frac{1.0}{-0.0} = \infty$$

When $frac \neq 000\ldots00$ this represents the case where no numeric value can be determined. For example:

$$sqrt{-1}, \infty - \infty, \infty \times 0$$

From negative to positive these 3 types of float values are arranged like so:

1. NaN
2. $-\infty$
3. -Normalized
4. -Denorm
5. -0
6. +0
7. Denorm
8. Normalized
9. ∞
10. NaN

Examples

we shall take the toy example to show these concepts.
consider a tiny floating point with 8 bits where s is 1 bit, exp is 4 bits and $frac$ is 3 bits.

$$s\ eeee\ fff$$

this tiny float has the same general form as the IEEE formats which means it can represent normalized, denormalized, 0, NaN, and ∞
for non-negative numbers the ranges of these values are

denormalized	0 0000 000 $\rightarrow$ 0 0000 111
normalized	0 0001 000 $\rightarrow$ 0 1110 111
special	0 1111 000

Within the distribution of thes numbers one can observe that the numbers are most dense at 0, meaning that this 4-bit representation of the IEEE standard for floating point numbers can represent more values closer to 0 and less as one moves farther away from 0.

Properties

a special property of this representation is that we can represent positive and negative values of 0,
Furthermore, the representation for 0 is all 0s in binary. We can also compare signed and unsigned integers as long as NaN is not involved and one must compare the sign bits first.