CS61C Spring 2018 Discussion 9

Floating Point

The IEEE 754 standard defines a binary representation for floating point values using three fields:

- The *sign* determines the sign of the number (0 for positive, 1 for negative)
- The *exponent* is in **biased notation** with a bias of 127
- The *significand* is akin to unsigned, but used to store a fraction instead of an integer.

The below table shows the bit breakdown for the single precision (32-bit) representation:

Sign	Exponent	Significand
1 bit	8 bits	23 bits

There is also a double precision encoding format that uses 64 bits. This behaves the same as the single precision but uses 11 bits for the exponent (and thus a bias of 1023) and 52 bits for the significand.

How a float is interpreted depends on the values in the exponent and significand fields:

Exponent	**Significand**	**Meaning**
0	Anything	Denorm
1-254	Anything	Normal
255	0	Infinity
255	Nonzero	NaN

For normalized floats:

$$\text{Value} = (-1)^{\text{Sign}} \times 2^{(\text{Exponent} - \text{Bias})} \times 1.\text{significand}_2$$

For denormalized floats:

$$\text{Value} = (-1)^{\text{Sign}} \times 2^{(\text{Exponent} - \text{Bias} + 1)} \times 0.\text{significand}_2$$

Exercises

1. How many zeroes can be represented using a float? 2

2. What is the largest finite positive value that can be stored using a single precision float?
   incr0x7F7FFFFF = $(2 - 2^{-23}) \times 2^{127}$

3. What is the smallest positive value that can be stored using a single precision float?
   0x00000001 = $2^{-23} \times 2^{-126}$

4. What is the smallest positive normalized value that can be stored using a single precision float?
   0x00800000 = 2^{-126}

5. Convert the following numbers from binary to decimal or from decimal to binary:
   0x00000000 8.25 0x00000F00 39.5625 0xFF94BEEF -∞

0x00000000 = 0
8.25 = 0x41040000

$0x000000F0 = (2^{-12} + 2^{-13} + 2^{-14} + 2^{-15}) \times 2^{-126}$
$39.5625 = 0x421E4000$
$0xFF94BEEF = NaN$
$-\infty = 0xFF800000$

Representation Exercises

Not every number can be represented perfectly using floating point. For example ⅓ can only be approximated and thus must be rounded in any attempt to represent it. Sometimes even numbers we may expect to be representable may not be. We will look at analysing those numbers in this next question. For this question we will only look at positive numbers.

1. What is next smallest number larger than 2 that can be represented completely? Next smallest number larger than 4 that can be represented completely? Next smallest number larger than $4 - 2^{-22}$ that can be represented completely?
For each of these questions you increment the number by the smallest amount possible. This is the same as incrementing the signficand by 1 at the rightmost location.

$2 + 2^{-23} * 2 = 2 + 2^{-22}$

$4 + 2^{-23} * 4 = 4 + 2^{-21}$

$4 - 2^{-22} + 2^{-23} * 2 = 4 - 2^{-22} + 2^{-22} = 4$

2. Define the stepsize to be the distance between some value x and the smallest value larger than x that can be completely represented. What is the step size for 2? 4? $4 - 2^{-22}$?
This is just the amount added in part 1.

2^{-22}
2^{-21}
2^{-22}

Notice that those with the same exponent value had the same value.

3. Now let's see if we can generalize the stepsize for normalized numbers (we can do so for denormalized numbers as well but we won't in this question). If we are given a normalized number that is not the largest representable normalized number with exponent value x and with significand value y what is the stepsize at that value? Hint: There are 23 signifcand bits.

Here we need to generalize the solution we got in 1 and 2. However this is the same approach. Just increment the significand by the 1.

$curr\_number = 2^{x-127} + 2^{x} * 2^{y}$

$next\_number = 2^{x-127} + 2^{x-127} * 2^{y} + 2^{x-127} * 2^{-23}$

step size = next_number - curr_number = 2^{x-150}

4. Now let's apply this technique. What is the largest odd number that we can represent. Part 3 should be very useful in finding this answer.

To find the largest odd number we can represent we want to find when odd numbers will stop appearing. This will be with step size of 2.

As a result plugging into 3:

$2 = 2^{x-150}$, $x = 151$

This means the number before $2^{151-127}$ was a distance of 1 (it is the first value whose stepsize is 2) and no number after will be odd. Thus the odd number is simply subtracting the previous step size of 1.

$2^{24} - 1$

AMAT

AMAT is the average (expected) time it takes for memory access. It can be calculated using this formula:

$$\text{AMAT} = \text{hit time} + \text{miss rate} \times \text{miss penalty}$$

Miss rates can be given in terms of either local miss rates or global miss rates. The *local miss rate* of a cache is the percentage of accesses into the particular cache that miss at the cache, while the *global miss rate* is the percentage of all accesses that miss at the cache.

Exercises

Suppose your system consists of:

- A L1$ that hits in 2 cycles and has a local miss rate of 20%
- A L2$ that hits in 15 cycles and has a global miss rate of 5%
- Main memory hits in 100 cycles

1. What is the local miss rate of L2$?
   Local miss rate = 5% / 20% = 0.25 = 25%

2. What is the AMAT of the system?
   AMAT = 2 + 20% x 15 + 5% x 100 = 10 (using global miss rates)
   Alternatively, AMAT = 2 + 20% x (15 + 25% x 100) = 10

3. Suppose we want to reduce the AMAT of the system to 8 or lower by adding in a L3$. If the L3$ has a local miss rate of 30%, what is the largest hit time that the L3$ can have?
   Let H = hit time of the cache. Using the AMAT equation, we can write:
   $2 + 20\% \times (15 + 25\% \times (H + 30\% \times 100)) \leq 8$
   Solving for H, we find that $H \leq 30$. So the largest hit time is 30 cycles.

Flynn Taxonomy

1. Explain SISD and give an example if available.

Single Instruction Single Data; each instruction is executed in order, acting on a single stream of data. For example, traditional computer programs.

2. Explain SIMD and give an example if available.

Single Instruction Multiple Data; each instruction is executed in order, acting on multiple streams of data. For example, the SSE Intrinsics.

3. Explain MISD and give an example if available.

Multiple Instruction Single Data; multiple instructions are executed simultaneously, acting on a single stream of data. There are no good modern examples.

4. Explain MIMD and give an example if available.

Multiple Instruction Multiple Data; multiple instructions are executed simultaneously, acting on multiple streams of data. For example, map reduce or multithreaded programs.