CS61C Spring 2015 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:

For normalized floats:

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

For denormalized floats:

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

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

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?

   0x7F7FFFFF = $(2 - 2^{-23}) \times 2^{127}$

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

-∞ = 0xFF800000

AMAT

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

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

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% x (15 + 25% x (H + 30% x 100)) ≤ 8
   Solving for H, we find that H ≤ 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.