1 Floating Point

1. We can have +0 or -0. (The mantissa and exponent need to be 0, so we are free to change our sign bit).

\[0x8000 0000 \quad 0x0000 0000\]

-0 +0

2. Our exponent cannot be 255 since that would yield ±Inf or NaN. Thus, we need our exponent field to be 254. Subtracting our bias, we will be multiplying our mantissa by \(2^{254-127} = 2^{127}\). To maximize this we need our mantissa to be all 1's. Thus our value will be \(1.11\ldots1\cdot2^{127}\). Now to express this as a closed for value, we use a little math.

\[
0.11\ldots1 = \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^{23}} = \sum_{i=1}^{23} 2^{-i} = \frac{\frac{1}{2}(1-\left(\frac{1}{2}\right)^{23})}{1-\frac{1}{2}}
\]

\[
= \frac{1}{2} \cdot 2 \cdot (1 - \left(\frac{1}{2}\right)^{23}) = 1 - 2^{-23} \Rightarrow 1.11\ldots1 = 1 + 1 - 2^{-23} = 2 - 2^{-23}
\]

Thus, our closed form answer is \((2 - 2^{-23}) \cdot 2^{127}\). In hexadecimal,

\[
\begin{array}{c|c|c|c|c|c|c}
S & E & M & 1 & 1 & 1 & 1 \Rightarrow 0x7F7FFFFF
\end{array}
\]

Keep the above trick in mind, it is pretty useful.

3. We want to take advantage of denormalized values, since with them, we can obtain values < \(2^{-126}\). Our sign bit is 0 and our exponent field is also 0. We want a non-zero value and we know for denormalized values we don’t have an implicit leading 1. Thus, to make our mantissa as small as possible, we should set the 23\textsuperscript{rd} bit to 1 ⇒ our number will be \(2^{-23} \cdot 2^{-126} = 2^{-149}\). In binary,

\[
\begin{array}{c|c|c|c|c|c|c|c}
S & E & M & 0 & 0 & 0 & 0 \ldots 01 & \Rightarrow 0x00000001
\end{array}
\]

4. If we want the smallest normalized value, we need our exponent field to be non-zero; let’s set it to 1. One actual power of 2 is now \(2^{1-127} = 2^{126}\). We now have an implicit leading 1 since we have normalized values so our mantissa can now be all 0’s. Since that will yield a value of 1.0…0. Thus, our value becomes

\[
\begin{array}{c|c|c|c|c|c|c|c}
S & E & M & 0 & 000\ldots0000 & \Rightarrow 0x00800000
\end{array}
\]

Thus, our number is

\[
0.0000000000011100000000 \cdot 2^{-126} \Rightarrow (2^{-12} + 2^{-13} + 2^{-14} + 2^{-15}) \cdot 2^{-126}
\]
5. $39.5625$

$0.5625 = 0.5 + 0.0625 = \frac{1}{2} + \frac{1}{16} = 0b0.1001$

$39 = 0b100111$

$⇒ 39.5625 = 0b100111.1001$ In scientific notation, $1.001111001 \cdot 2^5$

$39.5625$ is clearly a normalized number.

$+1.001111001 \cdot 2^5 = +1.001111001 \cdot 2^{132-127}$

$S = 0$

$E = 132_{10} = 0b10000100$

$M = 0b00111100100\ldots0$

$0 \quad 00|001110|001|1110|00001|00000000$

$⇒ 0x421E4000$

$0xFF94BEEF$

$S = 0x1 \quad E = 0b11111111 = 255_{10} \quad M ≠ 0$

When $E = 255$, $M ≠ 0$, we have NaN.

$0x0000 0000$

$\left\{ \begin{array}{l}
 S = 0x0 \\
 E = 0x00 \\
 M = 0x0 \\
\end{array} \right\} \Rightarrow \text{This is } +0$

$8.25$

$0.25 = \frac{1}{4} = 0b0.01$

$2^{-1} \uparrow \uparrow 2^{-2}$

$8 = 0b1000$

$⇒ 8.25 = 1000.01$ Converting to scientific notation:

$1000.01 = 1.00001 \cdot 2^3$ (note: this is base 2, so we multiply by $2^k$, $k ∈ Z$) $8.25$ is clearly a normalized number

$⇒ +1.00001 \cdot 2^3 = +1.00001 \cdot 2^{130-127}$

$S = 0$

$M = 000010\ldots0$ (remember, there’s an implicit 1 so we only need everything after the decimal point)!

$E = 130_{10} = 10000010$

$⇒ \frac{0 \quad 00|0001|000000001|00\ldots0100|0\ldots0}{S \quad E \quad M} \Rightarrow 0x4104 0000$

$0x0000 0F00$

$S = 0x0$

$E = 0x0$

$M = 0b00000000 0000 1111 0000 0000 \rightarrow \text{E} = 0, \text{M} ≠ 0 \Rightarrow \text{a denormalized number.}$

$-∞$

$S = 0b1 \quad E = 0b1111 1111 \quad M = 0x0$
1. Let’s visualize how this system looks (simplified):

```
CPU 100% → L1 $ 20% → L2 $ 5% → MEM *
```

The L1 $ receives all accesses from the CPU, so its local miss/hit rate is also its global hit/miss rate.

If we have a global miss rate of 5% on L2 $, and the L1 $ misses 20% of its accesses, that means the L2 $ receives 20% of accesses globally, of which means its local miss rate is

\[ M = \frac{\text{the amount of accesses it misses}}{\text{all accesses that reach the L2 $}} = \frac{0.05}{0.20} = 0.25 \]

2. Following the AMAT formula,

\[
\text{AMAT} = (\text{L1 $ hit time}) + (\text{Local L1 $ miss rate})(\text{L1 $ miss penalty})
\]

\[
= 2 + 0.20 ((\text{L2 $ hit time}) + (\text{Local L2 $ miss rate})(\text{L2 $ miss penalty}))
\]

\[
= 2 + 0.20 (15 + 0.25 \cdot 100) = 10
\]

Alternatively, we could consider how often we hit each cache globally; from the diagram in part 1:

\[
\text{AMAT} = (100\%)(\text{L1 $ hit time}) + (20\%)(\text{L2 $ hit time}) + (5\%)(\text{main memory hit time})
\]

\[
= 2 + 0.2 \cdot 15 + 0.5 \cdot 100 = 10
\]

Both answers yield the same answer (this is because \( (\text{L1 $ local miss rate})(\text{L2 $ local miss rate}) = (\text{L2 $ global miss rate}) \))

3. Now our diagram looks like this:

```
CPU 100% → L1 $ 20% → L2 $ 25% → L3 $ 30% → Main Memory *
```

Following the AMAT equation:

\[
\text{AMAT} = (\text{L1 $ hit time}) + (\text{Local L1 $ miss rate})(\text{L1 $ miss penalty})
\]

\[
= 2 + 0.2 ((\text{L2 $ hit time}) + (\text{Local L2 $ miss rate})(\text{L2 $ miss penalty}))
\]
\[
= 2 + 0.2 \left( 15 + 0.25 \left( \text{(L3 $ hit time)} + \left( \text{Local L3 $ miss rate)\text{(L3 $ miss penalty)}} \right) \right) \right)
\]

\[
= 2 + 0.2(15 + 0.25(H + 0.30 \cdot 100 )) \leq 8
\]

can’t miss main mem

⇒ H ≤ 30, so the largest hit time we can have for our L3 $ while still having an 8 cycle AMAT is 30 cycles.