Translating between Decimal and binary floating point

Hunter Pitelka

January 26, 2009

From Decimal To Binary

Fill in the table below with the binary number

s	exponent								fraction																						

1. Fill in the following values:
   - number of exponent bits: $k = \underline{\qquad\qquad}$
   - unsigned value of the exponent: $exp = \underline{\qquad\qquad}$
   - Bias value for exponent: $B = 2^{k-1} - 1 = \underline{\qquad\qquad\quad}$
2. Is the exponent (exp):
   - All 1's ?
     - If the fraction is all 0's the value is $\pm\infty$
     - if the fraction is nonzero, the value is NaN
   - All 0's? ... **Denormalized**
     - Exponent is $e = 1-$ Bias $= \underline{\qquad\qquad}$
     - Fraction is $frac = 0.\{\textit{fractional Component from above}\}$
     - **Your answer is:** $0.frac \times 2^e$
   - Something else: **Normalized**
     - $e = exp-$Bias $= \underline{\qquad\qquad\quad}$
     - Fraction has implied leading one: $1.\{\textit{fractional component from above}\}$
     - **Your answer is:** $1.frac \times 2^e$

From Decimal to Binary

Write down the binary equilivalent of your number

Now shift the binary decimal point as necessary to create a leading 1:

1.

$\times 2$________

- Now fill in the following values:
  - Number of exponent bits $k =$____________
  - Now calculate your exponent:
    * The actual exponent as recorded above is $e =$____________
    * Your *Bias* is $B = 2^{k-1} - 1 =$ ____________
    * The encoded exponent is $exp = e + b =$____________
    * Is your encoded exponent $exp \leq 0$? You are generating a **Denormalized value!**
      · Set your actual exponent to $e = 1 - B = 1-$____________
      · Set your encoded exponent $exp = 0$ to signify denormalization.
      · Shift over your binary point to be 2^{1-B}
      · Denormalization simply means that you do not have a leading zero infront of your fraction.
    * Write down your final exponent bits:

  - Number of fraction bits $f =$____________
    * Is your fraction larger than the max bits? You need to **round!**:
    * Mark off the last bit that can be kept, this is your *guard* bit.
    * The bit less significant from your guard bit is your *sticky* bit.
    * The logical OR of everything less signifcant than your sticky bit is your *round* bit.
    * If $(G||S)\&\&R$ then round up (add one). And post-normalize to get the leading one. (possibly dealing with denormalization as shown above)
    * Write down your final fractional bits:

You're all set! Now just write out the binary number as:
Sign bit - exponent bits - fractional bits