Name:

SSN(6):

CMSC 311 Computer Organization

Answers to Nasty Numbers Worksheet or Fondly Floating Point

–NO CALCULATORS ON EXAMS–

Dr. Hugue

Please note: In some of the following problems, a calculator would be helpful for speed in conversion. However, no calculators will be used for the exams, and the problems will be written to reflect that restriction.

Fixed Point Problems

1. Give three different *base 10* interpretations of the bit string:

   1101 1001 1000 0001

   Answers: Here are four, and a recipe for more.

   And for a little added value: excess/bias: really anything one wants, given the right bias

   - two's complement: -9855
   - one's complement: -9854
   - signed magnitude: -55681
   - unsigned binary :

   Note: You could certainly interpret the bit string as other *integral* or *fixed point* numbers by moving the assumed position of the binary point from the current right most position to left most position Or, you could define a bias and use that

2. Convert the following "decimal fractions" to 24-bit fixed point equivalents, where the "binary fraction" is the lower 12 bits, and the "binary integer" part is the upper 12 bits. The "binary integer" part is stored as a sign magnitude number. You may give your answer in hexadecimal for convenience. Also indicate which representations are exact, which are approximations because of truncations, and which cannot be expressed because of overflow. Is underflow present anywhere?

   (a) 256.75
   (b) 0.908
   (c) -4099.125
   (d) 0.0000 0004 (hint: multiply by 16 and accumulate integers)
   (e) -64.0000 0004

   Answers:

   first, in sign magnitude

   - 100c00 exact
   - 000e87 approximation because of truncation
   - cannot be expressed because of overflow

- 000000 - underflow
- 840000 - underflow

3. Express both operands in 32-bit fixed point, with 16 bits for "binary integer" part (including sign), and 16 bits for the "binary fraction" part. Then, perform the indicated operations. Remember, subtraction is implemented by adding the 2's complement of the number being subtracted. (Hint: some of the numbers to be converted were used in previous problems.)

   (a) $-12,598 - (0.908)$

   (b) $-4099.125 + 4097.375$

   (c) $1 - 0.0000 \quad 0004$

   (d) $256.75 - 264.00$

in 2's complement

- cec9178d
- fffe4000
- 00010000
- fff8c000

Floating Point Problems

An IEEE single precious floating point number, **z**, is stored in 32 bits, with parts **s**, the sign bit; **e**, the 8-bit excess-127 exponent; and, **f**, the 23 bit fractional part as shown below.

$$z = (-1)^s \quad 1.f \quad \times 2^{e-127}$$

1. Express the fixed point numbers below using 32-bit IEEE floating-point notation. (Hint: use your fixed point base 2 representation as a starting point.)

   (a) 256.75

   (b) 0.908

   (c) -4099.125

   (d) 0.0000 0004 (hint: multiply by 16 and accumulate integers)

   (e) $-64.0000 \quad 0004$

2. Express both operands in 32-bit IEEE floating point and perform the indicated operations. Remember, addition and subtraction require both operands to have the same exponent so that the binary points are aligned. Furthermore, subtraction is implemented by adding the 2's complement of the number being subtracted. (Hint: some of the numbers to be converted were used in previous problems.)

   (a) $-4099.125 + 4097.375$

   (b) $1 - 0.0000 \quad 0004$

   (c) $256.75 - 264.00$

Extreme Issues

As mentioned previously, and repeated for your convenience, an IEEE single precision floating point number, **z**, is stored in 32 bits, with parts **s**, the sign bit; **e**, the 8-bit excess-127 exponent; and, **f**, the 23 bit fractional part as shown below.

$$z = (-1)^s \quad 1.f \quad \times 2^{e-127}$$

Since all of the 2^{32} bit patterns are accounted for above, we must set aside several exceptional or *signal values* which are interpreted by the rules below, instead of the simple-minded formula above.

s	e	f	Meaning
0/1	00_H	00_H	Zero
0/1	00_H	non-zero	DeNormalized
0	FF_H	00_H	$+\infty$
1	FF_H	00_H	$-\infty$
0/1	FF_H	non-zero	Not a Number (NaN)

1. Why must zero be a "signal" value?
2. What is the largest exponent that can be represented for a normalized floating point number? What value of **e** is stored corresponding to this exponent?
3. What is the smallest (most negative) exponent that can be represented, and what value of **e** is stored for this exponent?
4. Give an example of a denormalized floating point number and indicate how it would be stored.