Date:

**Homework for Chapter 3 Arithmetic for Computers** 3.10

Name: Leonard Blam

1. This problem covers 4-bit binary multiplication. Fill in the table for the Product, Multiplier and Multiplicand for each step. You need to provide the DESCRIPTION of the step being performed (shift left, shift right, add, no add). The value of M (Multiplicand) is 1011, Q (Multiplier) is initially 1010.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Product** | **Multiplicand** | **Multiplier** | **Description** | **Step** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0000 0000 | 0000 1011 | 1010 | Initial Values | Step 0 |
| 0000 0000 | 0000 1011 | 0 | Multyply by 0 (first right place of multiplier) | Step 1 |
| 0001 0110 | 0000 1011 | 1 | Multiply by 1 (second right place of multiplier) | Step 2 |
| 0001 0110 | 0000 1011 | 0 | Multyply by 0 (third right place of multiplier) | Step 3 |
| 0101 1000 | 0000 1011 | 1 | Multiply by 1 (last right place of multiplier) | Step 4 |
| 0110 1110 |  |  | Sum all product together | Step 5 |
|  |  | etc … |  |  |

1. This problem covers floating-point IEEE format.
2. List four floating-point operations that cause NaN to be created? + ,x, /, square-root
3. Assuming single precision IEEE 754 format, what decimal number is represent by this word:

1 01111101 00100000000000000000000

(Hint: remember to use the biased form of the exponent.)

Decimal: -0.28125

Binary: 10111110100100000000000000000000

3. The floating-point format to be used in this problem is an 8-bit IEEE 754 normalized format with 1 sign bit, 4 exponent bits, and 3 mantissa bits. It is identical to the 32-bit and 64-bit formats in terms of the meaning of fields and special encodings. The exponent field employs an excess-7 coding. The bit fields in a number are (sign, exponent, mantissa). Assume that we use *unbiased rounding to the nearest even* specified in the IEEE floating point standard. (a) Encode the following numbers the 8-bit IEEE format:

1. 0.0011011binary
2. 16.0decimal =
3. Decode the following 8-bit IEEE number into its decimal value: 1 1010 101

Sign value: -1

Exponent: 10 - 15

Fraction: 2^-1 + 2^-3

-13/256 or -0.0507812

=Decide which number in the following pairs are greater in value (the numbers are in 8-bit IEEE 754 format):

* 1. 0 0100 100 and 0 0100 111 3/256 and 15/16384
  2. 0 1100 100 and 1 1100 101 3/16 and -13/64

1. In the 32-bit IEEE format, what is the encoding for negative zero? -1 \* 2^-127 \* .00000000000000000
2. In the 32-bit IEEE format, what is the encoding for positive infinity? -1^0 \* 2^128 \* 1.0000000000000

1. The floating-point format to be used in this problem is a normalized format with 1 sign bit, 3 exponent bits, and 4 mantissa bits. The exponent field employs an excess-4 coding. The bit fields in a number are (sign, exponent, mantissa). Assume that we use *unbiased rounding to the nearest even* specified in the IEEE floating point standard. (a) Encode the following numbers in the above format:
   1. 1.0binary = 0 011 0000
   2. 0.0011011binary
2. Using 32-bit IEEE 754 single precision floating point with one(1) sign bit, eight (8) exponent bits and twenty three (23) mantissa bits, show the representation of -11/16 (-0.6875).

1 01111110 01100000000000000000000

-1 \* 2^-1 \* 1.011000000000

1. What is the smallest positive (not including +0) representable number in 32-bit IEEE 754 single precision floating point? Show the bit encoding and the value in base 10 (fraction or decimal OK).

-1^0 \* 2^-127 \* .00000000000000000000000001 = 1.4E-45

1. We’re going to look at some ways in which binary arithmetic can be unexpectedly useful. For this problem, all numbers will be 8-bit, signed, and in 2’s complement.
2. For x = 8, compute x & (−x). (& here refers to bitwise-and, and − refers to arithmetic negation.)

0000 1000

1111 1000

& 0000 1000 = 8

1. For x = 36, compute x & (−x).

0010 0100

1101 1100

& 0000 0100 = 4

1. Explain what the operation x & (−x) does.

Takes the x value in 8 bit code and “and” it with the 2’ compliment of its own x value