CSCI 250, Homework 3
Spring 2011
100 Points Total

Name: _____

1 Integer Arithmetic Using Hexadecimal Numbers

For each of the following questions, write the result in hexadecimal. You **must** show your work for credit. The following table shows pairs of hexadecimal numbers:

	A	**B**
a.	0D34	DD17
b.	BA1D	3617

1. What is $A + B$ if the numbers represent unsigned 16-bit hexadecimal numbers? (2 points)
2. What is $A+B$ if the numbers represent unsigned 16-bit hexadecimal numbers stored in sign-magnitude format? (2 points)
3. Convert A into a decimal number, assuming it is unsigned. Repeat assuming it is stored in sign-magnitude format. (2 points)
4. What is $A - B$ if the numbers represent unsigned 16-bit hexadecimal numbers? (2 points)
5. What is $A - B$ if the numbers represent signed 16-bit hexadecimal numbers stored in sign-magnitude format? (2 points)

2 Integer Multiplication

1. Using a table similar to that shown in Figure 3.7 in the textbook, calculate the product of the hexadecimal unsigned 8-bit integers A and B using the hardware described in Figure 3.4. You should show the contents of each register on each step. (5 points)

3 Optimized Integer Multipliers

We would like to design multipliers that require less time. Many different approaches have been taken to accomplish this goal. In the following table, A represents the bit width of an integer, and B represents the number of time units (tu) taken to perform a step of an operation:

	A (bit width)	B (time units)
a.	4	3 tu
b.	32	7 tu

1. Calculate the time necessary to perform a multiply using the approach given in Figures 3.4 and 3.5 if an integer is A bits wide and each step of the operation takes B time units. Assume that in step 1a an addition is always performed — either the multiplicand will be added, or a 0 will be. Also assume that the registers have already been initialized (you are just counting how long it takes to do the multiplication loop itself). If this is being done in hardware, the shifts of the multiplicand and multiplier can be done simultaneously. If this is being done in software, they will have to be done one after the other. Solve for each case. (5 points)

2. Calculate the time necessary to perform a multiply using the approach described in the text (31 adders stacked vertically) in an integer is A bits wide and an adder takes B time units. (5 points)

3. Calculate the time necessary to perform a multiply using the approach given in Figure 3.8, if an integer is A bits wide and an adder takes B time units. (5 points)

4 IEEE 754 Floating Point

In the IEEE 754 floating point standard the exponent is stored in "bias" (also known as "excess-N") format. This approach was selected because we want an all-zero pattern to be as close to zero as possible. Because of the use of a hidden 1, if we were to represent the exponent in two's-complement format, an all-zero pattern would actually be the number 1! (Remember, anything raised to the zeroth power is 1, so $1.0^0 = 1$.) There are many other aspects of the IEEE 754 standard that exist in order to help hardware floating-point units work more quickly. However, in many older machines floating-point calculations where handled in software, and therefore other formats were used. The following table shows decimals numbers:

a.	5.00736125 x 10^5
b.	-2.691650390625 x 10^{-2}

1. NVIDIA has a "half" format, which is similar to the IEEE 754 except that it is only 16 bits wide. The leftmost bit is still the sign bit, the exponent is 5 bits wide and stored in excess-16 format, and the mantissa is 10 bits long. A hidden 1 is assumed. Write down the bit pattern assuming this format. Comment on how the range and accuracy of this 16-bit pattern compares to the single precision IEEE 754 standard. (5 points)

2. Calculate the sum of A and B by hand, assuming A and B are stored in the 16-bit NVIDIA format described above. Assume one guard, one round bit and one sticky bit, and round to the nearest even. You **must** show all the steps. (5 points)

5 Commutative, Associative, and Distributive Laws of Floating Point

Operations performed on fixed-point integers behave the way one expects — the commutative, associative, and distributive laws all hold. This is not always the case when working with floating-point numbers. Let's first look at the associative law. The following table shows sets of decimal numbers:

	A	B	C
a.	-1.6360 x 10^4	1.6360 x 10^4	1.0 x 10^0
b.	2.865625 x 10^1	4.140625 x 10^{-1}	1.2140625 x 10^1

1. Calculate $(A + B) + C$ by hand, assuming A, B, and C are stored in the 16-bit NVIDIA format described in above. Assume one guard, one round bit, and one sticky bit, and round to the nearest even. Show all the steps, and write your answer both in 16-bit floating point format and in decimal. (10 points)

2. Calculate $A + (B + C)$ by hand, assuming A, B, and C are stored in the 16-bit NVIDIA format described in above. Assume one guard, one round bit, and one sticky bit, and round to the nearest even. Show all the steps, and write your answer both in 16-bit floating point format and in decimal. (10 points)

3. Based on your answers to the above two exercises, does $(A + B) + C$ equal $A + (B + C)$? (5 points)

4. Calculate $(A \times B) \times C$ by hand, assuming A, B, and C are stored in the 16-bit NVIDIA format described in above. Assume one guard, one round bit, and one sticky bit, and round to the nearest even. Show all the steps, and write your answer both in 16-bit floating point format and in decimal. (15 points)

5. Calculate $A \times (B \times C)$ by hand, assuming A, B, and C are stored in the 16-bit NVIDIA format described in above. Assume one guard, one round bit, and one sticky bit, and round to the nearest even. Show all the steps, and write your answer both in 16-bit floating point format and in decimal. (15 points)

6. Based on your answers to the above two exercises, does $(A \times B) \times C$ equal $A \times (B \times C)$? (5 points)