Introduction to Computer Engineering I (ECSE-221)
Assignment 1
Available: Sep 22, 2006
Due Date: Oct 6, 2006

Please submit this assignment by 5pm, Oct 6, 2006 on WebCT. Your assignment must consist of the answers to Q1-Q5 in a Microsoft Word (.doc) file, the source code for Q6 as a "C" (.c) file, and output from Q6 as an ASCII text (.txt) file.

This assignment will be marked out of 100 points.

Assignments received up to 24 hours late will be penalized by 10%; assignments received up to 48 hours late will be penalized by 20%, and assignments received more than 48 hours late will not be marked.

Q1. Convert the following integers into their numerical equivalents in the indicated bases (X refers to your answer, the subscript indicates the base). Be sure to use the correct number of significant figures for each case and show how the number of significant figures was obtained. Show all your work. (10 points)

(a) 122_{10} (2 points)
- (i) X_2
- (ii) X_{16}

(b) $1100\ 1010\ 0010_2$ (3 points)
- (i) X_{16}
- (ii) X_{10}
- (iii) X_3

(c) $C21.23F_{16}$ (5 points)
- (i) X_2
- (ii) X_{10}
- (iii) X_5

Q2. What is the maximum and minimum number in the range of a 5-digit number for each of the following cases? Express all numerical answers in both the original base and its decimal (base-10) equivalent. For each case, also express the total number (in base-10) of distinct numbers that can be represented by the number scheme. Show all your work. (10 points)

(a) binary (2's complement) (3 points)

(b) base 21 (positive integers) (3 points)

(c) base 6 (6's complement) (4 points)

Q3. Perform the following calculations assuming that all numbers are stored in 16-bit registers as 2's complement binary numbers with no overflow provision. Convert each of the numbers to their binary equivalent, and do all your calculations in binary to produce a binary result. Convert the result to hexadecimal and decimal (base-10) formats. If a computation produces incorrect results, explain why. Show all of your work. (10 points)

(a) $33B7_{16} - 219C_{16}$ (5 points)

(b) $4321_{16} + 531F_{16}$ (5 points)

Q4. Perform the following calculations using long arithmetic (long multiplication and long division) in unsigned binary. Give the quotient to 3 significant figures after the decimal place. Show all your work. (10 points)

(a) $00010011_2 * 00001011_2$ (5 points)

(b) $00010011_2 / 00001011_2$ (5 points)

Q5. IEEE754 (10 points)

(a) Convert the following base-10 number to single precision IEEE 754 based on the procedure described in class and in the notes. Express the resulting 32-bit binary number in hexadecimal format. Show all your work.

$-5.77359 * 10^{35}$ (5 points)

(b) Convert the following IEEE 754 single precision number to its decimal (base-10) equivalent. Show all your work.

$078DB219_{16}$ (4 points)

(c) Given a 40-bit IEEE 754 representation, where 11 bits are used for the exponent, what is the range of exponents that can be represented? In other words, what are the minimum and maximum powers of 2 that can be represented? Explain your reasoning (don't just apply the formula given in class). (1 point)

Q6. C Program (50 points)

Simple processors do not always support multiplication and division with native machine instructions. Here one has no choice but to implement these operators in software. Write a "C" function to perform *unsigned* binary division according to the following prototype, using the subtract and shift algorithm as demonstrated in class.

```
int div32(unsigned long dividend, unsigned long divisor,
          unsigned long *quotient, unsigned long *remainder);
```

The function should return a status code, 0, for successful completion and –1 for an error. The function must not use the multiplication or division operators.

Write a main() procedure which will repeatedly prompt the user for two numbers and then call your div32() function. After each call of the div32() function, print out the numbers entered, and the quotient and remainder returned by your function, in the following format (for the example of 12 divided by 5):

12 divided by 5 is 2 remainder 2

When the user enters two zeros, the program should stop.

Validate your code using an appropriate set of test cases. You'll need to devise a testing strategy, i.e., a set of test cases, which ensure that the function produces correct results over the permissible range of inputs. The problem boils down to choosing an appropriate set of test cases out of the 2^{64} possible input combinations.

In general, one of the most difficult and challenging aspects of software development is that of actually ensuring that the software is free from error and meets specifications.

Hint: Use "%u" for the input and output of unsigned integers ("%d" is used for input and output of *signed* integers).