Written Homework 01

Assigned: Th 7 Sep 2017
Due: Wed 20 Sep 2017

Instructions:

- The assignment has to be uploaded to Blackboard by the due date. NO assignment will be accepted after 11:59pm on that day.

- We expect that you will study with friends and often work out problem solutions together, but you must write up your own solutions, in your own words. Cheating will not be tolerated. Professors, TAs, and peer tutors will be available to answer questions but will not do your homework for you. One of our course goals is to teach you how to think on your own.

- You may turn in work to Blackboard that is either handwritten and scanned, written in a word processor such as Word, or typeset in LaTeX. In the case of handwritten work, we may deduct points if the scan is upside down or the work is illegible.

- To get full credit, show INTERMEDIATE steps leading to your answers, throughout.

Problem 1 [25 pts (3,3,2,3,5,4,5)]: Base conversions

i. By hand, convert the binary number 01010101 from binary to decimal, showing the powers of two that must be added to produce the correct solution.

ii. Convert the number from the previous part to hexadecimal, and show mathematically how each digit multiplied by a power of 16 produces the same decimal number that you found in part (i).

iii. The binary number above can be written as the binary number 0101 written twice in a row. What is the decimal version of this binary number? Explain how this fact could be used to convert the binary 01010101 directly into hex.

iv. Convert (by hand) the hexadecimal number $\text{C0DE}_{16}$ to sixteen digits of binary, using whatever method you like. Show and explain your work.

v. Use a method similar to the shortcut for converting hexadecimal to binary to convert the base 3 number $(2211002121202020)_3$ to base 9.

vi. Convert $(607080)_9$ to base 3 using a similar trick.

vii. Write the binary number 10110000 in base 32. Assume digits beyond 15 continue to use the normal alphabet (g=16, h=17, etc.).
Problem 2 [20 pts (4 pts each)]: Two’s Complement

i. Convert the decimal numbers -36 and -14 to two’s complement, assuming an eight bit two’s complement representation. Show your work, including how you get from the binary for positive integers 36 and 14 to their negative counterparts.

ii. Sum the numbers you produced in the preceding question using binary addition. (You must show where you carry for full credit.) Verify that the result is equal to the binary for -50 by converting to a positive integer and summing the values.

iii. If you are using 64-bit signed numbers (two’s complement), what is the sum of the number represented by 64 1 bits (1111...1) and the number represented by 56 0 bits followed by 8 1 bits (000...01111111)? (Give the answer in decimal.)

iv. Suppose I am using an ancient computer architecture that uses 12-bit numbers with a two’s complement representation of negative numbers. What is the smallest positive number that, when added to itself, produces overflow (a result with a negative interpretation)? Give the answer in both binary and decimal.

v. Generalize your answer to the preceding question to the $n$-bit case: with $n$ bits, what number is the smallest possible positive number that, when added to itself, overflows (produces a number with a negative interpretation)? Give your answer in decimal, in terms of $n$.

Problem 3 [16 pts (9,3,4)]: More Two’s Complement

i. For each binary number, give both its interpretation as a positive unsigned number and its interpretation as a two’s complement negative number. (Both should be in decimal.) For each, assume that the number of bits being used is equal to the number of bits shown.

   1. 10001000 (8 bits)
   2. 1110 (4 bits)
   3. 100000000001 (12 bits)

ii. For each pair of numbers you generated in the preceding problem, sum their absolute values.

iii. For an $n$-bit two’s complement number, what is the sum of these two absolute values in terms of $n$?

Problem 4 [23 pts (4,6,6,4,3)]: Logical Equivalence

i. Draw a logic circuit for $\neg p \lor q$.

ii. Draw a logic circuit for $\neg((p \lor q) \land \neg q)$. (Do not simplify.)

iii. Write out a truth table for the second formula. Is it equivalent to the first?

iv. Rewrite the formula from part (a) using De Morgan’s Law so that it uses no OR operators. (Use $\neg$ for negation and $\land$ for AND.)
We say “$p$ implies $q$” if, whenever $p$ is true, $q$ is also true. (If $p$ is false, $q$ can be true or false.)

Briefly explain in your own words why the logical formula you just produced for the previous question is true only if the inputs $p$ and $q$ are consistent with the statement “$p$ implies $q$.”

**Problem 5** [16 pts (6, 4, 6)]: The Satisfiability Problem

A formula is *satisfiable* if there exists some variable assignment that makes it true; that is, there is some row of its truth table that comes out to true instead of false. Determining whether an arbitrary boolean formula is satisfiable is called the *satisfiability problem*. This is a famously hard problem in computer science – nobody knows a good way of doing this that is much better than trying assignments until one succeeds, and fabulous prizes await anybody who comes up with a better method.

i. Make a truth table for the formula $(a \lor b) \land (b \lor \neg c) \land (a \lor c)$. The formula is satisfiable if at least one row evaluates to true. Is it satisfiable? (Complete the truth table even if you find a satisfying assignment.)

ii. Suppose a truth table must be created to know for sure whether a formula is satisfiable. If the formula has $n$ variables, how many rows will the truth table have, as a function of $n$?

iii. The satisfiability problem is not hard for all formulas; some formulas are easy, even if they have a lot of variables, because they have a structure that makes it easy to infer what the values of the variables must be. Consider, for example, the 100-clause formula:

$$x_1 \land (\neg x_1 \lor x_2) \land (\neg x_2 \lor x_3) \land \ldots \land (\neg x_{99} \lor x_{100})$$

How many satisfying assignments does this formula have? How do you know?