Institut for Matematik og Datalogi September 16, 2015
Syddansk Universitet JFB

Introduction to Computer Science E15 – Study Groups – Week 39

The first questions in this set of exercises are concerned with integers represented in the two's complement notation (see textbook pages 58–61).

For concreteness, assume we are working with 8-bit two's complement notation, which can represent integers between -128 and 127. See Figure 1.19 in the textbook for two's complement notation using 3 and 4 bits. All of the nonnegative values (from 0 up to the largest value) are expressed using standard binary notation. The negative numbers all have a 1 in the high order bit. The most negative number has all zeros, except for the high order bit. The remaining negative numbers are placed in order from smallest (the most negative number plus 1) to largest (-1) with the binary representation going from 1 to the largest positive number, if you ignore the sign bit. As examples, in this notation 00000101 represents the integer 5 and 10000101 represents the integer $-(2^7) + 5 = -128 + 5 = -123$.

1. Find the two's complement representations of the following integers: 0, 7, 27, 127, -128, -121, -101, -1.

2. Argue that in general, a bit sequence $b_7b_6b_5b_4b_3b_2b_1b_0$ represents the integer $b_7 \cdot (-(2^7)) + b_6 \cdot 2^6 + b_5 \cdot 2^5 + b_4 \cdot 2^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$.

3. The algorithm for addition of two numbers in two's complement is described as follows on pages 60–61: Add the two numbers position by position (including any carries), just as for positive integers in standard binary notation. If any carry comes out of the leftmost position (position of b_7), just discard it. If the input numbers are both positive (i.e., both have 0 in their leftmost position), but the result is negative (i.e., has 1 in its leftmost position), or the input numbers are both negative, but the result is positive, report an overflow error. Else return the result.

   Argue that this algorithm is correct, i.e., reports an overflow error when the result cannot be represented in 8-bit two's complement, and else returns the correct result.

Hint: Start by using the algorithm on various pairs of numbers whose representation you found in question 1. Then argue what happens in general in all eight cases of the possible signs of the two inputs and the single output, while using the expression from question 2.

4. Choose a number which cannot be expressed exactly in the floating point format we use (see the slides on floating point notation from September 8 or the notes associated with that lecture), but could be expressed exactly if there were more bits. How many more bits do you need?

5. If there is time, work on the following: Recall that any Boolean function can be computed using only AND, OR and NOT gates, since any Boolean function (with one output) can be be written in Disjunctive Normal Form.

   - Discuss the ease or difficulty of going from a formula to a circuit, vs. the ease or difficulty of going from a circuit to a formula. Create a circuit where the corresponding formula would naturally be larger than the circuit.
   - Show that for any circuit containing AND, OR and NOT gates, we can create an equivalent circuit (producing the same output for any set of inputs) which only has AND and NOT gates. Hint: For each gate in the original circuit, you can replace it by a subcircuit containing only AND and NOT gates. Consider the truth tables.
   - Show that for any circuit containing AND, OR and NOT gates, we can create an equivalent circuit (producing the same output for any set of inputs) which only has NAND gates. Hint: For each gate in the original circuit, you can replace it by a subcircuit containing only NAND gates. Consider the truth tables.