Due:  Friday, September 16 at 5:30 p.m.

Submission:  Please hand me a paper copy or slide it under my office door (SCI 3809).

Problems

1. What is the decimal notation for the following two’s complement number?
   \[ \begin{array}{ccccccccccc}
   1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
   \end{array} \]

2. What is the 16-bit two’s complement representation for the decimal number -6200?

3. What are the minimum and maximum integers that can be represented with a 64-bit two’s complement integer?

4. (Patterson & Hennessy C.24) The ALU we built in class supported the \texttt{slt} operation using just the sign bit of the adder for the most significant bit. Let’s try a \texttt{slt} operation using the values $-7_{\text{ten}}$ and $0_{\text{ten}}$. To make it simpler to follow the example, let’s limit the binary representations to 4 bits: $1001_{\text{two}}$ and $0110_{\text{two}}$.

   \[ 1001 - 0110 = 1001 + 1010 = 0011 \]

   This result would indicate that $-7 > 6$, which is clearly wrong. Hence, we must factor in overflow in the decision. Modify the 1-bit ALU in Figure C.5.10 on page C-33 to handle \texttt{slt} correctly when there is overflow.

5. Consider the following binary addition problem that will be computed using a carry-lookahead adder.

   \[ \begin{array}{cccccccc}
   1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
   + & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
   \hline
   1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
   \end{array} \]

   Following the format given in a similar example on p. C-44 of Patterson & Hennessy:
   - Find $g_i$ and $p_i$ for each bit.
   - Find $P_0$ through $P_3$, and $G_0$ through $G_3$, the propagate and generate functions for each 4-bit nibble.
   - Finally, find $C_1$ through $C_4$, the carry-out values from each 4-bit nibble.

6. Suppose you had worked out the various $P_i$, $G_i$, and $C_i$ values corresponding to a particular carry-lookahead sum, but then your roommate spilled a super caffiene drink on your paper, rendering much of it unreadable. All that is salvagable are the following values.

   \[ \begin{array}{cccc}
   P_0 & = & G_0 & = & C_1 = \\
   P_1 & = & G_1 & = 0 & C_2 = 1 \\
   P_2 & = & G_2 & = & C_3 = 0 \\
   P_3 & = & G_3 & = 1 & C_4 = \\
   \end{array} \]

   (a) Fill in as many of the missing values as possible. If not enough information is available to be sure about the correct value for one of the missing values, then leave it blank.
   (b) Give a pair of 16-bit binary numbers that could have resulted in the table given above.

Turn over for more problems...
7. Consider a 32-bit adder that is constructed by “cascading” two 16-bit carry-lookahead adders. In other words, the final carry-out of the first 16-bit adder is fed directly to the second 16-bit adder as its initial carry-in signal.

(a) How many gate delays are required before the final carry-out from the full 32-bit adder becomes correct?

(b) How many gate delays are required before the final sum bit from the 32-bit adder becomes correct?

Please justify your answers to both parts of this question.

Optional extra problems

1. Design a logic block (using logic gates) to implement overflow checking for 1-bit ALU for the most significant bit shown in Figure C.5.10.

2. (Patterson & Hennessy) A simple check for overflow during addition is to see if the CarryIn to the MSB is not the same as the CarryOut of the most significant bit. Prove that this check is the same as that shown in Figure 3.2 on page 226.

3. Work out the cost (in total number of gates for computing all carryouts) for a two-level 16-bit carry lookahead adder, versus a one-level 16-bit carry lookahead adder, versus a 16-bit adder with “infinite” hardware. Show your work.