Problem 1. (12 points) Sketch a block diagram for a magnitude comparator bit-slice circuit. Create K-maps to define the bit-slice circuit, and use them to find optimal logic equations. Sketch the bit-slice circuit.
Problem 2. (12 points) Modify the bit-slice block of problem 1 by removing the logic gates and signals that form the EQ output. Sketch a "block" circuit diagram for a 4-bit comparator that uses the modified bit slice blocks, and add a single gate to form the EQ output from the LT and GT outputs from the MSB (most significant bit).

(4 points) Could you make the bit-slice modules even more efficient by leaving in the EQ logic and removing some other logic? Explain.

Problem 3. (10 points) Complete truth tables and K-maps for HA and FA circuits, using XOR patterns where appropriate. Loop minimum SOP equations, and sketch the circuits (assume all inputs and outputs are active high).

<table>
<thead>
<tr>
<th>Half Adder</th>
<th>A</th>
<th>B</th>
<th>S</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

S = \quad B

Cout = \quad B

A

0 0

A

0 1
Problem 4. (10 points) Sketch a block diagram for a full adder using two half-adder blocks and an OR gate.

Problem 5. (8 points) Sketch a Carry-Propagate-Generate circuit that can form the carry-out for a 4-bit CLA.
Problem 6. (20 points) Design a full-subtractor bit-slice circuit (Borrow-Ripple Subtractor). Label the inputs A, B, and Bin, and label the outputs D and Bout. Start by completing the subtraction examples, then complete the truth table and K-maps, and then sketch the circuit.

\[
\begin{array}{c c c c}
\begin{array}{c}
0010 \\
-1000 \\
\end{array} & \begin{array}{c}
0010 \\
-1100 \\
\end{array} & \begin{array}{c}
0110 \\
-1000 \\
\end{array} & \begin{array}{c}
0110 \\
-1100 \\
\end{array}
\end{array}
\]

\[
\begin{array}{c c c c}
\begin{array}{c}
0010 \\
-1010 \\
\end{array} & \begin{array}{c}
0010 \\
-1100 \\
\end{array} & \begin{array}{c}
0110 \\
-1010 \\
\end{array} & \begin{array}{c}
0110 \\
-1110 \\
\end{array}
\end{array}
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Bin</th>
<th>D</th>
<th>Bout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c c c c}
\begin{array}{c}
001110 \\
\end{array} & \begin{array}{c}
001110 \\
\end{array} & \begin{array}{c}
01110 \\
\end{array} & \begin{array}{c}
01110 \\
\end{array}
\end{array}
\]

\[
\begin{array}{c c c c}
\begin{array}{c}
001110 \\
\end{array} & \begin{array}{c}
001110 \\
\end{array} & \begin{array}{c}
01110 \\
\end{array} & \begin{array}{c}
01110 \\
\end{array}
\end{array}
\]

\[
\begin{array}{c c c c}
\begin{array}{c}
001110 \\
\end{array} & \begin{array}{c}
001110 \\
\end{array} & \begin{array}{c}
01110 \\
\end{array} & \begin{array}{c}
01110 \\
\end{array}
\end{array}
\]

\[
\begin{array}{c c c c}
\begin{array}{c}
001110 \\
\end{array} & \begin{array}{c}
001110 \\
\end{array} & \begin{array}{c}
01110 \\
\end{array} & \begin{array}{c}
01110 \\
\end{array}
\end{array}
\]
Exercise 6: Arithmetic Circuits

Problem 7. (8 points) Complete the number conversions indicated. Note that all binary numbers are 8-bit two’s complement representations.

-19 = ________________ 10011010 = ________________

10000000 = _______ -101 = ________________

Problem 8. (22 points) Complete all the 8-bit 2’s compliment arithmetic problems below, showing both the decimal and binary numbers in each case.

17 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 1 0 1 1 +) -11 1 1 1 1 0 1 0 1 +) 6 + ________________

-35 -) 42 - ________________ -) -7 - ________________

Is the answer to the equation on the left correct in 8 bits? Explain.

Problem 9. (10 points) Sketch a circuit to convert a 4-bit binary number to its 2’s complement representation using only 3 XOR/XNOR gates and 2 AND or OR gates.
**Problem 10:** (8 points) Examine several examples of addition overflow and subtraction underflow, and sketch a circuit below that can output a ‘1’ whenever an addition or subtraction result is incorrect due to underflow or overflow. Assume that both operands and result of the addition and subtraction are N-bits. (Hint: compare the carry in and carry out signals of the most-significant bit).

**Problem 11.** (16 points) Fill in the squares below to show all signal values when “1101” and “1010” are multiplied.
Problem 12. (8 points) Sketch a block diagram for a 4-bit ALU built from bit-slice ALU circuits that can implement the functions shown in the table. Label all signals, and recall that inputs to the bit slices must come from the A and B input busses as well as from neighboring bit slices (and outputs must drive the F output bus as well as neighboring bit slices). To design the signals that communicate information between slices, you must understand the ALU operations and the implications for information transfer (e.g., does the operation A PLUS B require that information be transferred between slices? If so, what? Does the operation A OR B require that information be transferred?).

### ALU Function Table

<table>
<thead>
<tr>
<th>Operation Code</th>
<th>ALU function</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>A PLUS B</td>
</tr>
<tr>
<td>001</td>
<td>A PLUS 1</td>
</tr>
<tr>
<td>010</td>
<td>A MINUS B</td>
</tr>
<tr>
<td>011</td>
<td>A MINUS 1</td>
</tr>
<tr>
<td>100</td>
<td>A XOR B</td>
</tr>
<tr>
<td>101</td>
<td>A’</td>
</tr>
<tr>
<td>110</td>
<td>A OR B</td>
</tr>
<tr>
<td>111</td>
<td>A AND B</td>
</tr>
</tbody>
</table>

(8 points) The ALU operation table from the module has been reproduced below, but opcode 3 has been redefined as “decrement”. Complete the F and Cout table entries to define the decrement logic functions.

### ALU Operation Table

<table>
<thead>
<tr>
<th>Op Code</th>
<th>Function</th>
<th>F</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>A PLUS B</td>
<td>A xor B xor Cin</td>
<td>(A and B) or (Cin and (A xor B))</td>
</tr>
<tr>
<td>001</td>
<td>A PLUS 1</td>
<td>A xor Cin</td>
<td>A and Cin</td>
</tr>
<tr>
<td>010</td>
<td>A MINUS B</td>
<td>A xor B xor Cin</td>
<td>(A’ and B) or (Cin and (A xor B)’)</td>
</tr>
<tr>
<td>011</td>
<td>A MINUS 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>A XOR B</td>
<td>A xor B</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>A’</td>
<td>A’</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>A OR B</td>
<td>A or B</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>A AND B</td>
<td>A and B</td>
<td>0</td>
</tr>
</tbody>
</table>