Please answer question 1 on this paper and put all other answers in the blue book.

1. True/False. Please circle the correct response.

   a. T  Exactly one output of a decoder will be asserted, for any input.
   b. F  A 4-input multiplexer, where each input is 32-bits, can be constructed from
         32 multiplexers, where each multiplexer has four 4-bit inputs.
   c. T  If the clock rate of a processor is too high, the memory elements (registers,
         etc.) will capture incorrect values emanating from the combinational logic
         (e.g. the ALU).
   d. F  In a rising-edge triggered flip-flop, the data will enter the first latch when
         the clock rises and then enter the second latch when the clock falls.
   e. T  A “narrow” SRAM (many locations, each location containing a small
         number of bits) will require fewer input pins than a “wide” SRAM (fewer
         locations, each location containing more bits) with equivalent memory
         capacity.
   f. T  DRAM cells need to be refreshed because 1) reading changes the voltage
         level of the capacitor in a cell and 2) capacitors leak current.
   g. F  In an ALU, subtraction is performed by changing the sign (i.e. leftmost) bit
         of the second operand and then adding the operands.
   h. F  The fastest way to load a 32-bit constant on a MIPS 32-bit processor is to
         put the constant within a single instruction.
   i. T  In 8-bit two’s complement numbers, the negative of 10101101 (binary) is
         01010011.
   j. T  In IEEE 32-bit floating point numbers, a bias is used in the representation
         of exponents, rather than two’s complement, so that two floating point
         numbers can be easily compared using integer comparison on the entire 32
         bits of both numbers.
2. Build a decoder with three input lines but with only six output lines. If the value of the input corresponds to 6 or 7, then all output lines should be asserted to signal an error.

![Decoder Diagram]

3. Build a 3-bit ALU that performs addition, subtraction, bitwise-AND, and bitwise-OR. The inputs should be two 3-bit values (A and B), a carry-in, and a 2-bit op code (where 0 indicates addition, 1 indicates subtraction, 2 indicates AND, and 3 indicates OR). There are no additional inputs. The outputs should be a 3-bit result, a carry-out, and a 1-bit “neg” output whose value is 1 if and only if the result is negative. The addition does not have to be performed in a lookahead fashion. You can use and/or/not gates, as well as multiplexers, decoders, and flip-flops, as needed – without building them from scratch (I’m not suggesting you need all of these).

A three-bit ALU is created using three 1-bit ALUs and a 1-bit ALU is created using a 1-bit adder (exactly as shown in the class notes). See the figures below.
Here is a 1-bit adder:

Here is a 1-bit ALU:
Below is the three-bit adder. The only thing notable about it (i.e. different from the class notes) is that the Binvert line for every ALU and the Cin line for ALU0 are asserted when the value of the 2-bit OP line is 01, indicating subtraction.
4.

a. Convert (algebraically, not by using a truth table) the following boolean formula to “sum of products” form:

\[(a + (b \cdot c)) \cdot (d + (a \cdot e))\]

show your steps, justifying each step.

Since

\[(a + (b \cdot c)) = \overline{(a \cdot (b \cdot c))} = \overline{(a \cdot (b + c))} = \overline{(a \cdot b) + (a \cdot c)}\]

by deMorgan’s law (twice) and distributivity, and since

\[(d + (a \cdot e)) = (d + \overline{(a + e)}) = (d + \overline{a + e})\]

by deMorgan’s law and associativity,

\[\overline{(a + (b \cdot c))} \cdot (d + (a \cdot e)) = ((a \cdot b) + (a \cdot c)) \cdot (d + \overline{a + e})\]

\[= ((a \cdot b \cdot d) + (a \cdot c \cdot d) + (a \cdot b) + (a \cdot c) + (a \cdot b \cdot e) + (a \cdot c \cdot e))\]

\[= (a \cdot b) + (a \cdot c)\]

by distributivity and then simplification (since, e.g., \((a \cdot b \cdot d) + (a \cdot c) = (a \cdot b)\)).

There’s probably a simpler way to arrive at the result though...

b. Write a single truth table corresponding to the boolean equations

\[x = (a + (\overline{b \cdot c}))\]
\[y = \overline{c} \cdot (a + b)\]

\[
\begin{array}{ccc|cc}
 a & b & c & x & y \\
 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 1 & 1 & 0 \\
 0 & 1 & 0 & 0 & 1 \\
 0 & 1 & 1 & 0 & 0 \\
 1 & 0 & 0 & 1 & 1 \\
 1 & 0 & 1 & 1 & 0 \\
 1 & 1 & 0 & 1 & 1 \\
 1 & 1 & 1 & 1 & 0 \\
\end{array}
\]