1. The following circuit is used as a synchronizer for the traffic sensor used on the traffic controller designed in class. The traffic controller is clocked on a 10 Hz clock. We expect the sensor to be activated about once a minute during the day.

   ![Circuit Diagram]

   a. How fast should we clock this synchronizer?

   b. Shouldn’t we have a reset on these flip-flops? Why or why not?

   c. Assuming we clock this synchronizer at 10 Hz, can we assume that since the input, SENSOR_IN, only changes once a minute (1/60 Hz) that it will never fail (i.e. store a metastable state)?

   d. Assuming we cannot change the clock frequency or rate of input changes, is there any way to reduce the probability that we will have a failure of the synchronizer?

2. Design a 3-register machine that has 8-bit registers on a single data bus. RA will have its input tied to memory (IN), and the other two registers (RB and RC) will have their inputs and outputs on the data bus. Draw a block diagram and also draw a timing diagram showing the control signals, and clock, to perform the following order of transfers: RA ← IN, RB ← RC, RC ← RA in as few clock cycles as possible. IN is an 8-bit input (i.e. not the contents of another register.). Show all control signals changing on the falling edge of the clock. The registers are clocked on the rising edge of the clock.

3. Build a state machine for your circuit in Problem 2 that, when the input GO is asserted, performs the transfers listed, in order.
4. Think about the set of registers in Problem 2 as a simple computer. Do not assume anything else (like resets on the registers).
   a. What are some possible instructions you would be able to execute given those three registers and the shown control lines. Think about the instructions we have for the simple computer and use this to start thinking about what you could do with this one. Keep in mind, this one is very simple and very limited!
   
   b. Assuming that you have an address space of 128 words, how many bits would you need to encode your instructions and operands? That is, how large is an instruction word?
   
   c. Using the instructions you came up with in Part A, write a program to load RB with a value from memory location 0, and RC with a value from memory location 1 (using the IN lines as an interface to memory).

5. Using the CPU instructions from class, hand-assemble a program that computes the square of the value in memory location 10h and places this result in memory location 20h. Don’t worry about any overflow conditions. Also, you can assume the number will be non-negative. Using the VHDL code for the processor designed in class, simulate your program and show the result when memory location is initialized with the value 4. Your program should start at memory location 000 so it runs after reset.

Turn in the end of your simulation that clearly shows the final result being written into location 20h and then halts.