Two things to tackle in this lab. First, become familiar with Logisim, create some circuits, and simulate others. Logisim is free; you can download it to your computer, and spend as much time as you like hacking. Second, we’ll fire up the Arduino, and download a simple program to it.

**LOGISIM**

1) Download 3bit-adder.circ from the CS120 web page, and open it with Logisim. Try the following addition problems with the circuit:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>101</td>
<td>111</td>
</tr>
<tr>
<td>+001</td>
<td>+011</td>
<td>+111</td>
</tr>
</tbody>
</table>

Do the results make sense if the numbers are unsigned? Do they make sense if the numbers are 2’s complement (you should be able to figure out the base-10 equivalent of these easily).

2) Build a 2-input, 4-output decoder, and a 2-input, 1-output MUX, using the basic Boolean gates. Simulate them – make sure you know how they work, and what they’re supposed to do.

3) Download traffic.circ, which is the traffic light example from the book. Simulate that, and think about what it’s doing in terms of being a finite state machine. Getting an idea of how this works will help a LOT with the next step.

4) Build a finite state machine that has four states, and no inputs. The state machine will have two outputs: light A, and light B. As the clock ticks on your state machine, the lights should light up A, B, both lights on, both lights off, and then back to the first state. The state diagram is below.

Steps to follow to do this:

a) Build a truth table for the state machine.
b) Build a 2-input decoder
3) Add OR gates for each output (one for the A light, one for the B light), and OR gates for each bit of the next state you need to move to (there are two bits).
4) Add in back-to-back latches and a clock (you can copy this from the traffic circuit)
5) Hook up the lights
6) Toggle the clock, and see your FSM run!
Make sure you understand how to build the finite state machine circuit. You'll get questions very much like this during the exam.

**ARDUINO TIME**

Open up the Arduino development environment, and hook your Arduino to the lab computer using a USB cable (there should be one that goes into the monitor – just grab that if you don’t have a USB cable of your own).

1) Open “file/examples/basics/blink” – this is a small sample program.
2) Click the “upload” button – this is an arrow to the right
3) If everything is set up correctly, you’ll see the LEDs on the Arduino flicker for a moment, and then there will be a single LED that blinks on and off. This LED is connected wires THAT THE SOFTWARE CAN CONTROL DIRECTLY! This is an important thing to note – software can affect hardware (and hardware can affect software). In many places, hardware and software are interchangeable.
4) The Arduino language is very much like C (not exactly the same, but close). If you look in source code, you can find where the light turns on and off – and also a delay of 1000ms. Change the code so that the light blinks on and off more slowly (for example 2000ms), or faster (200ms).

Now would be a good time to think for a bit. The blink example connects to pin 13 (one of the wires coming out of the processor chip); the chip has a number of other wires that you could control and could connect LEDs to.

A mux and a decoder, just in case you need them. You’ll need to draw them on the exam, and know how they work...