

# CENG 351: Computer Architecture I

## Lab 5: Control

Labs should be completed in pairs.

Processor design requires careful attention to detail. The most important thing about the single cycle processor control unit is the truth table that defines its operation. However, at some point of the processor design process, that truth table must be implemented in hardware.

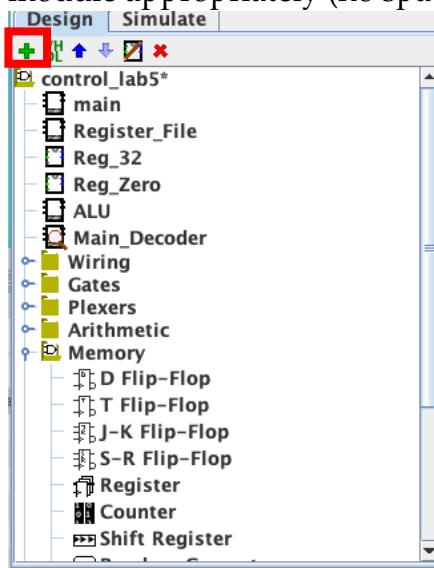
In this lab, you will build on your single-cycle datapath design in Logisim to include a functional control unit. If your datapath did not work properly, speak with Prof. Harrison about making sure any issues are fixed before you implement the control unit. Although there are many options in implementing a control unit, this lab will guide you through one potential solution. You are welcome to implement another hardware solution as long as it works properly. Once completed, you will test your complete processor design using the same basic set of assembly instructions as the previous lab.

### A note on Modularity:

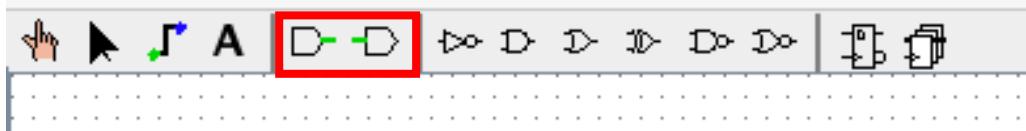
We will use a hierarchy of modules to simplify our design. We will add a “control unit” module that contains both the “main decoder” and “ALU decoder” as separate modules.

### ALU Decoder

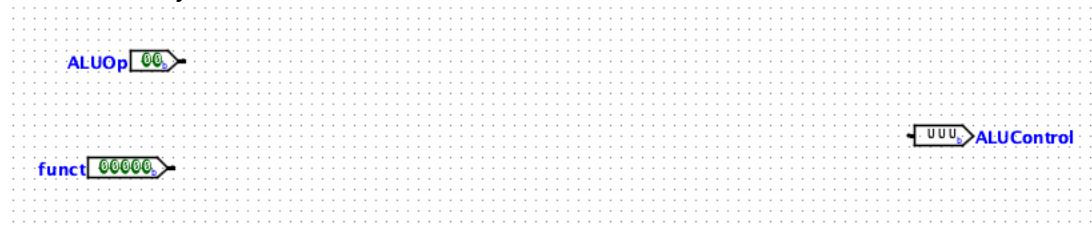
1. Click on the green plus button in the top left to add the ALU decoder. Name the module appropriately (no spaces allowed; you can use underscores).



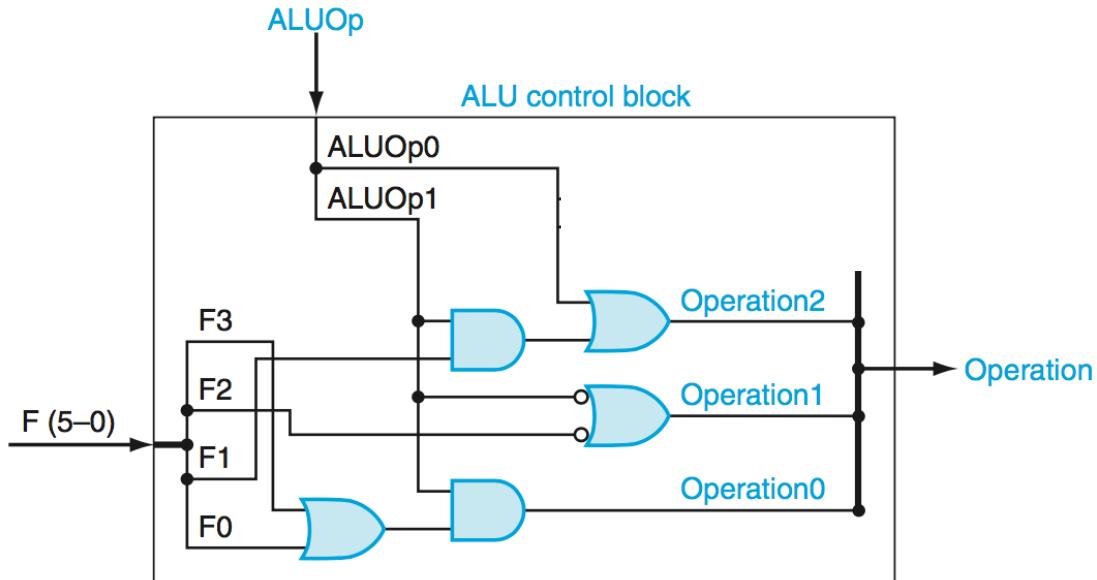
2. Recall that the ALU decoder has two inputs, the ALUOp signal, which is two bits wide, and the funct field, which is six bits wide. The output is the ALUControl signal, which is 3 bits wide and will connect to the ALU F input. You will need to add these controls using the input and output pin objects.



3. Add these inputs and outputs and name them appropriately (you can double-click the pin to add a name). Naming them will help when we go up a step in the hierarchy.



4. Implement the ALU Decoder according to the design below. Recall that this design is a combinational logic implementation of the truth table for the ALU Decoder covered in class. Recall that two of the funct field bits are unused. In the design below, the wires labeled "Operation" are the bits of the ALUControl signal for output. If you wish, you can implement another design with the same functionality.

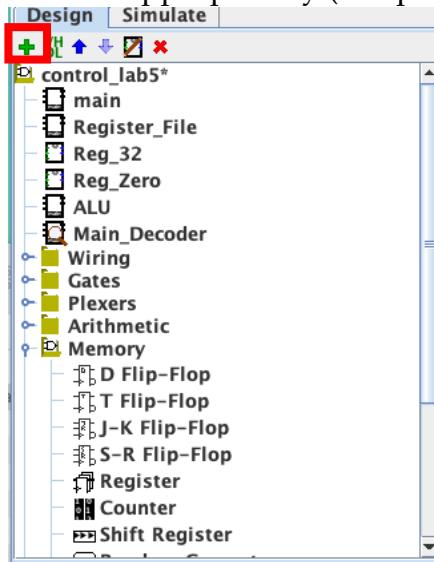


- When working on your design, be careful with what bits you are using from each of your splitters. You can negate the inputs (add bubbles) of logic gates using the properties tab after you select the gate.

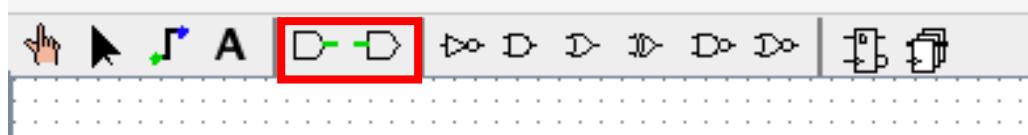
Properties		State
Selection: OR Gate (730,380)		
FPGA supported:	Supported	
Facing	East	
Data Bits	1	
Gate Size	Medium	
Number Of Inputs	2	
Output Value	0/1	
Label		
Label Font	SansSerif Bold 16	
Negate 1 (Top)	Yes	
Negate 2 (Bottom)	Yes	

## Main Decoder

- Click on the green plus button in the top left to add the main decoder. Name the module appropriately (no spaces allowed; you can use underscores).

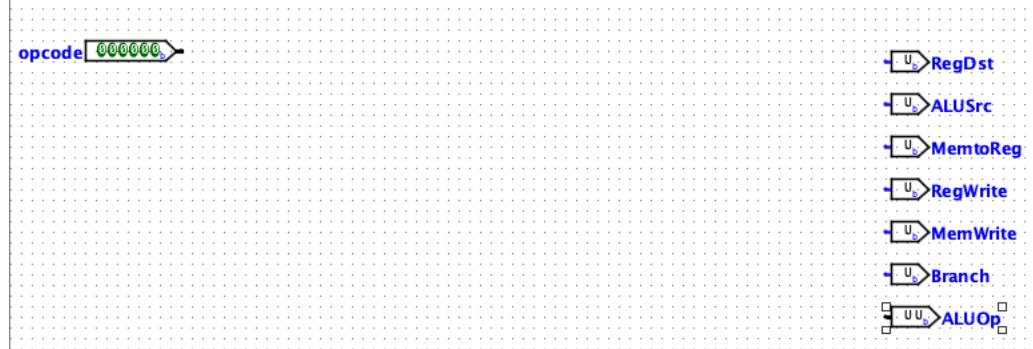


- Recall that the main decoder has one input, the six-bit opcode. The outputs are all the control signals, including ALUOp. You will need to add these controls using the input and output pin objects.



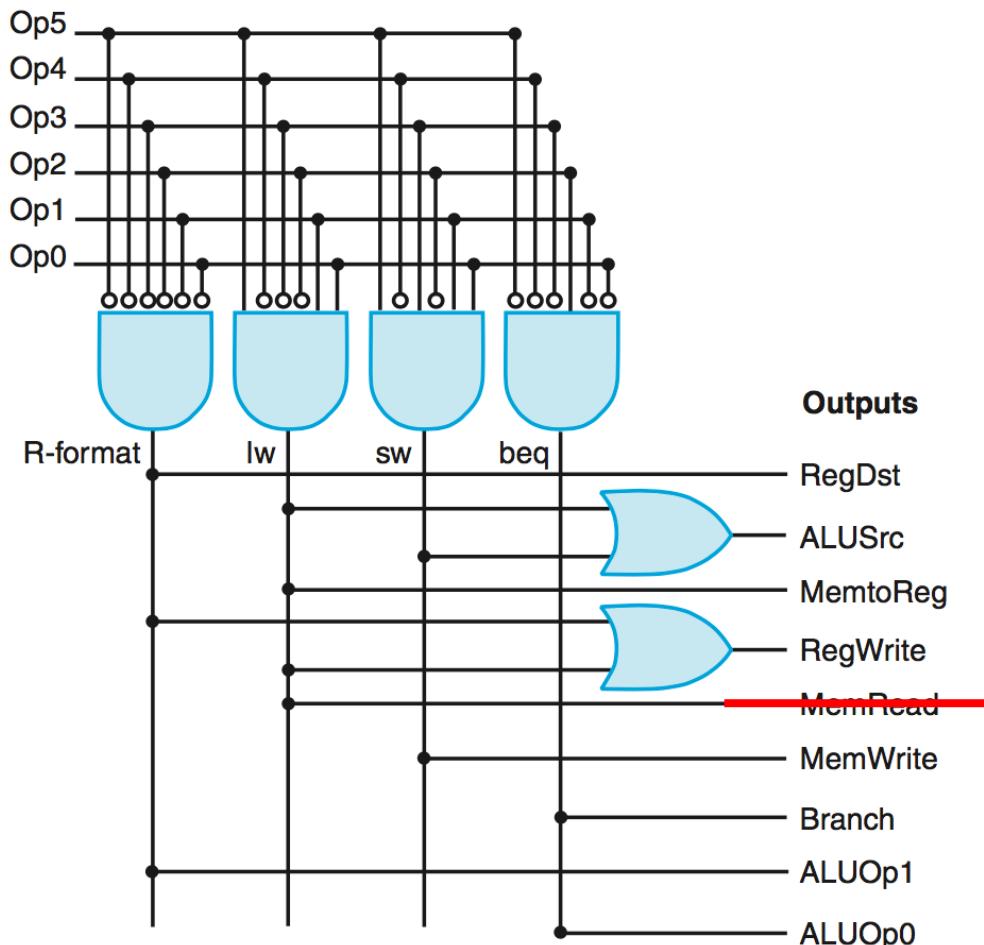
- Add these inputs and outputs and name them appropriately (you can double-click the pin to add a name). Naming them will help when we go up a step in the hierarchy. *Note: A diagram showed in class includes a MemRead signal, but we did*

not implement this. You do not need to include it as an output for this lab.



4. Implement the Main Decoder according to the design below. Recall that this design is a combinational logic implementation (basically just a decoder) of the truth table for the Main Decoder covered in class. In the design below, the ALUOp bits are separated, but you will want them combined into a single 2-bit output. If you wish, you can implement another design with the same functionality.

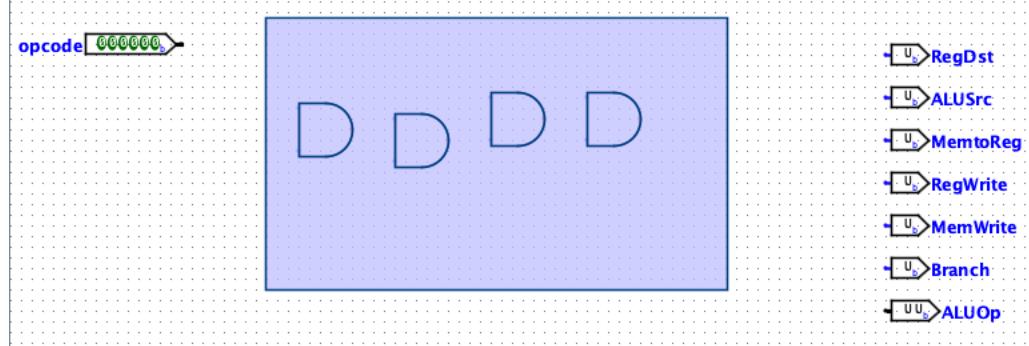
### Inputs



5. The design above is missing the functionality to execute the *addi* instruction. You need to add this instruction to the main decoder. The control information (including opcode) for *addi* is given below.

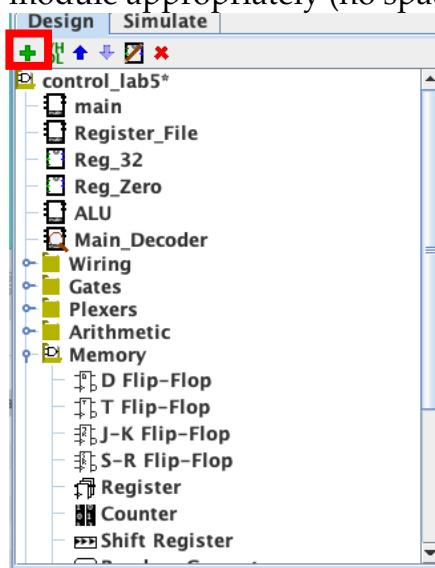
Instruction	Op <sub>5:0</sub>	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	ALUOp <sub>1:0</sub>
addi	001000	1	0	1	0	0	0	00

6. Note: You can drag and select multiple objects and set some of their properties all at once. For example, you can add 4 AND gates, then select all four to change their orientation and the number of input bits at the same time. Recall that the inputs to the AND gates match the opcode of the instruction. I suggest you label the and gates with the instruction they represent.

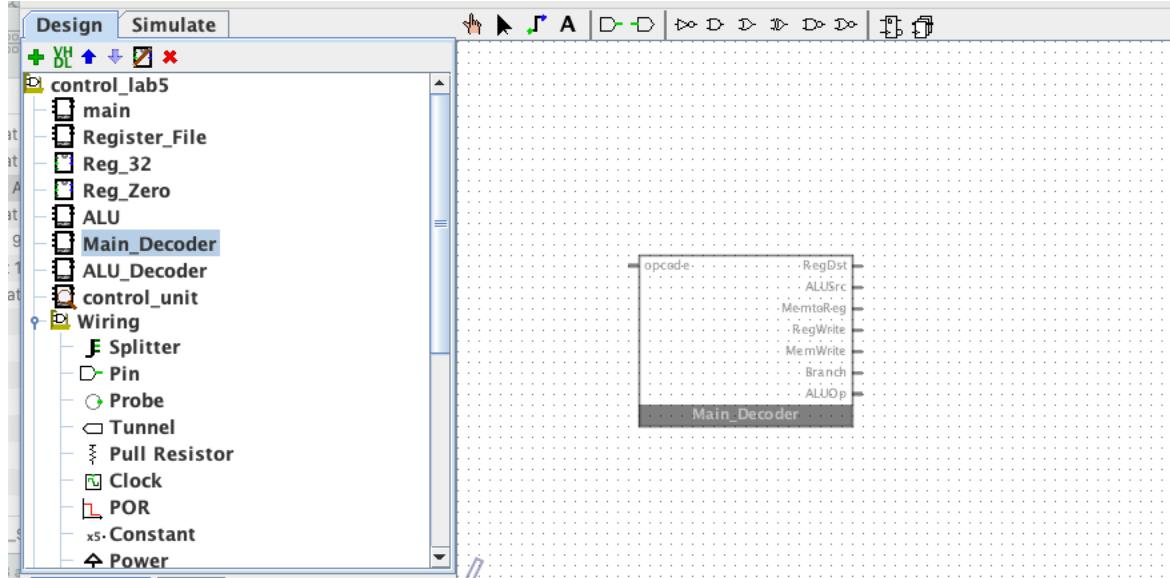


## Constructing the Control Unit

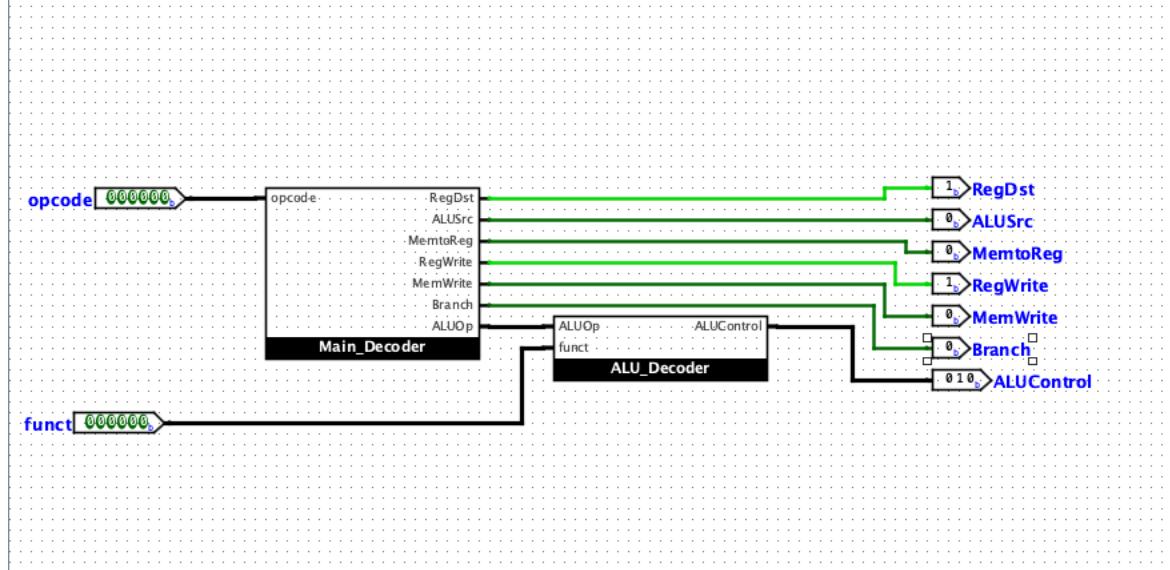
1. Click on the green plus button in the top left to add the control unit. Name the module appropriately (no spaces allowed; you can use underscores).



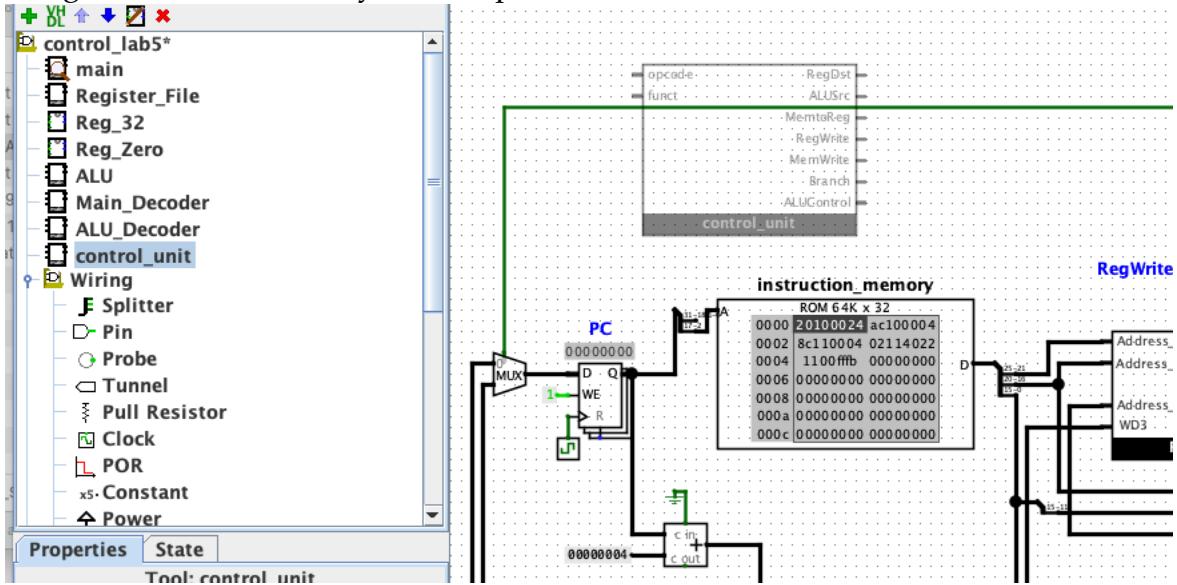
2. Use the design tree on the left-hand side to select your Main Decoder and ALU decoder and add them to your design. When you click on the name in the design tree, the unit will be highlighted. You can then click on the design area to add it as a module.



3. Note: the input and output names you chose when building the modules will be reflected here, with inputs on the left and outputs on the right. If you wish to modify the appearance of the module, right-click the name in the design tree and select "Edit Circuit Appearance"
4. Once the two decoders have been added, connect them together appropriately. Add the appropriate inputs (opcode and funct fields) and outputs (control signals). Make sure to label your outputs appropriately.



5. If you wish to simulate the control unit manually, you may do so here by adjusting the opcode and funct fields. However, we will test it with instructions in the next section.
6. Add the control unit to your datapath. Double click on the “main” module in the design tree to return to the datapath. Select your control unit module from the design tree and add it to your datapath.



7. Hookup the control unit appropriately. It will need parts of the instruction (opcode and funct field) as inputs, and the outputs will be the control signals. You should remove the input pins you previously used for the control signals and connect the wires directly to the control unit.

## Simulation

1. You will test the processor design using the same assembly as in the last lab. You can re-load some sample instructions in the instruction memory or simply reset the simulation (ctrl-R or cmd-R).
2. An assembly program which will loop forever is given below.

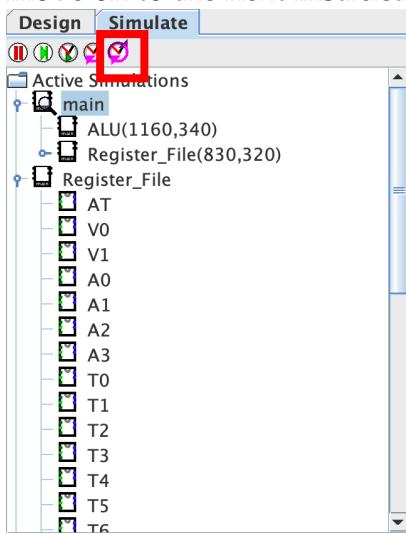
```
top: addi $s0, $0, 36
      sw    $s0, 4($0)
      lw    $s1, 4($0)
      addi $s1, $s1, -1
      sub  $t0, $s0, $s1
      and  $t1, $t0, $s1
      beq  $t0, $t1, top
```

machine language:

```
20100024
ac100004
8c110004
2231ffff
```

02114022  
01114824  
1109ffff9

3. Once you have loaded the simulation, you will step through the assembly one clock cycle at a time. Before you step forward, make sure that the control unit is setting appropriate control signals for your instruction.
4. After each instruction is verified, click on the “Tick clock one full cycle” button. You should see the program counter increment and the instruction memory move on to the next instruction.



5. Repeat this process for all instructions, taking screenshots to verify the operation of your control unit. You will need to click on the ALUControl signal with the hand pointer tool to see its contents. At the end, the assembly program should repeat. You should also be able to see a value stored in the data memory.

Compile a short report that contains screenshots of the control unit showing the correct operation for each instruction. There should be at least 5 screenshots for instruction operation, but you can include more if breaking up the screenshot makes your report easier to read. Also include a screenshot of each of your underlying schematics. Describe any problems you had during the lab and what you did to troubleshoot. Describe each partner’s contribution to the lab. Include answers to the lab questions below

#### Lab questions:

1. Aside from using the decoder combinational logic shown in this lab, what is another way you could implement the main decoder? You can just describe another implementation; you do not need to design it.

2. What are the advantages of taking a hierarchical, modular approach to implementing the control unit? Would you consider implementing the design without using this hierarchy/modularity?

### **Submission**

Please submit a digital copy of your lab report. If taking screenshots, please save them into the lab report document instead of as separate files. The lab should be submitted through canvas and only one report needs to be submitted per group. In the lab report, please indicate all lab partners and their contributions.

### **Due Date**

This assignment is due on the date shown on Canvas by 11:59pm. Submit via Canvas in PDF format.

### **Grading**

Grading will be based on correctness and completeness of the solution (i.e. show all your work).