Carleton University

Department of Systems and Computer Engineering

SYSC 3006 Fall 2016

Computer Organization

Lab #6

Prelab Due: Your Prelab should be done *before* the start of your scheduled lab session. Bring your Prelab with you in hard or soft copy form (e.g. print it or have a word/pdf document ready) for TA inspection during the lab session (no online submission).

In this lab you will:

- Program a given microarchitecture to execute additional instructions.
- Implement and test using Logisim.

Read this document carefully before deciding what to do.

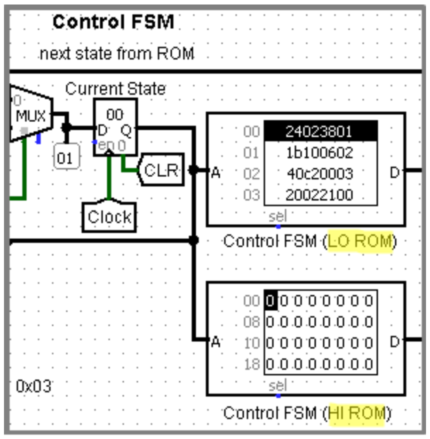
A folder including 2 Logisim circuits has been included. **You MUST UNZIP the folder** (i.e. extract all files) before trying to use the files. If you do not unzip the files, they may load, but may not work.

The distributed Lab6 circuit is a working implementation of a computer system that is very similar to the one in Lab 5. In this lab, a few circuits have been added to support immediate operands (similar to those discussed in class). Just as in Lab 5, you will add some new instructions to the processor's instruction set, and execute a small Test Program that uses them. In this lab, **you will not modify any of the supplied circuits** … you will only change the contents of the ROMS, registers and Main Memory as required.

By The Way: This is the last lab that requires filling in ROM Tables. ☺ The remaining labs will stay on the software side of the hardware/software interface (i.e. filling Main Memory with instruction encodings).

Adding the new "immediate operand" circuitry required more control bits (in total) than the 32-bit capacity of the Control FSM ROM. So, to compensate, the Control FSM was split into two ROMs:

- The LO ROM starts out with familiar signals from Lab 5 (excluding the Dead state; it will not part of Lab 6).
- A parallel HI ROM has been added for the new bits. The same address (i.e. current state) is fed to both ROMs at the same time, and therefore, the ROMs (together) provide more than 32 control bits at once.



00	24023801
01	1b100602
02	40c20003
03	20022100

00	0	0	0	0	0	0	0	0
08	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0

The Prelab support tables include a Test Program (the last table in the doc). For this Lab, you must design the Control FSM implementations of the instructions in the Test Program. You do not have to implement any other instructions … only the ones in the Test Program. Pay attention to the different forms of the same instruction (which will require different execution states).

[PreLab: Complete the provided Lab6F16_SupportTables.docx]

Design the Control FSM states for the instructions in address 0 through 9 of the Test Program (the variables X, Y, and Z don't count as instructions ☺). The remaining instructions (A-E) are **optional**.

Note: The "mystery instruction" is an intended twist. ☺ Do **not** try to build a Control FSM implementation for the instruction Opcode 0xAB, but be sure to include the "mystery instruction" encoding exactly as given in your Main Memory Test Program (expect the unexpected!).

There are several Prelab tables to be filled:

- The Control FSM Output tables (LO ROM and HI ROM): the first 6 states (F0-E2) will be identical to those in Lab 5, and will execute Form 1 instructions. State 7 will be used to execute a few Form 2 instructions. The remaining Form 2, Form 3, Form 4 and Form 5 instructions will need new execution states (as discussed in class; refer to the TowardsSoftware lectures or notes).
    - Design the execution states for the instructions in the Test Program only. You do not have to implement any other instructions. Instructions at addresses A to E are **optional**.
    - All instructions will use states 0 through 3 for their fetch and decode stages.
    - MOV and NOT (Form 2) require two execution states, yet notice how state 7 reuses state 6 (i.e. sets it as the next state) to finalize the execution. Feel free to reuse states in that way to reduce the number of redundant rows in your Control ROM. Remember to consider both the LO and HI tables to evaluate whether or not you have a redundant row.
- Complete the Decode ROM table for the instructions used in the Test Program. After fetching an instruction and decoding it, the Decode ROM decides the Control ROM's next state (i.e. the first execution state). As an example, see the entires which have already been filled for you.
    - Notice that the NOP instruction in this lab doesn't require any execution states (it simply returns to state 0 to fetch the next instruction, like in Lab 4). The "waste 2 cycles" version we did in Lab 5 was an exercise in idling the processor for a specified number of cycles (multiple uses: e.g. synchronizing within a multi-core processor, waiting on I/O, etc.).
- Complete the Main Memory Test Program to contain the encodings of the indicated instructions.
    - The Results column is only there for your own use and practice. It will **not** be marked.
    - An **Instruction Encoder Tool** has been provided in the zip file to help you with the task of encoding the instructions. Note that the tool opens in a browser. Double clicking on the file should open the tool automatically in your default browser. The human interface is reasonably intuitive, just pick the opcode and operands that you want and the tool will tell you the encoding. The tool includes more instructions than needed, including the instructions required for the lab. Know limitation as of version V0-3: the tool does not accept negative immediate decimal values in situations where it should.

[**In the Lab:** Validate your design and record a simulation log]

The working implementations will be demonstrated by loading the Test Program into Main Memory. Be sure to put the indicated variables (X, Y, Z) into Main Memory, too. Validate your prelab design by following these steps in the provided Logisim circuit:

1. Fill in the words of the Control FSM Output ROMs, Decode ROM, and Main Memory with the values from your completed prelab tables.
2. Make sure the System clock is high (in the 1 state).
3. Press the Clear button to put the system in a known initial state.
4. Cycle the System Clock through the execution of the Test Program (or use Simulate > Ticks Enabled to let the program automatically trigger the clock for you. Its toggle shortcut is Ctrl+K. The rate can be changed under Simulate > Ticks Frequency)

It is not necessary to demonstrate the step-by-step operation of the circuit to the TAs. Instead, show your completed simulation log to the TAs before you leave the lab.

Simulation Log: create a simulation log (under Simulate > Logging) that validates the execution of your test program. The log should show the hex values for the relevant parts of the Programmer's Model: i.e. the IR, any registers modified during the execution of the Test program (including the PC or R15), and any main memory locations that are modified during the execution of the Test Program. You may choose to show some register values in decimal (instead of hex), if desired.

Good Luck! ☺