

Carleton University
Department of Systems and Computer Engineering
SYSC 3006 Fall 2016
Computer Organization
Lab #3 (Version 2)

Prelab: 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).


Note: missing or forgetting to bring the Prelab can result in an up to **40% mark loss**. The Prelab is described at the end of this document.

In this lab you will:

- Design a datapath like the one discussed in the lectures
- Implement and test the design using Logisim.

Read this entire document carefully before deciding what to do.

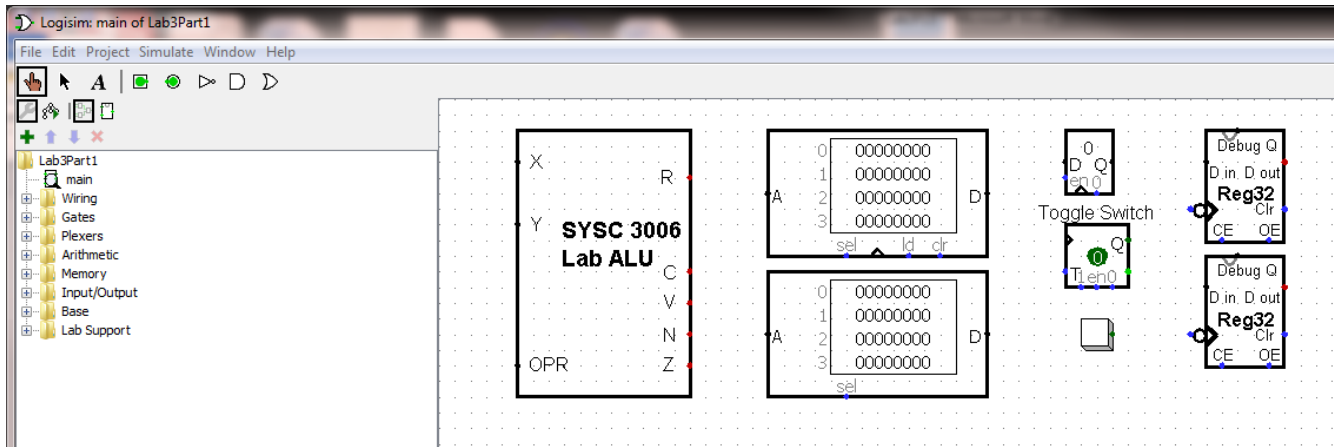
A folder containing 2 Logisim circuits has been included (not required for the Prelab). **You MUST UNZIP the folder** (i.e. extract all files) before trying to load these circuit files into Logisim. **If you do not unzip the files, they may load into Logisim, but will not simulate properly!!!**

In this lab, all page and figure numbers refer to the  [Towards Hardware](#) notes on culearn.

The Simple Processing Circuit (shown in Figure 21) performs ALU operations on memory words and stores the result in a memory word. This circuit requires a unique Control FSM for each operation. An encoded operation latch can be added to the circuit to generalize the (Mealy) Control FSM, as shown in slide 95. The introduction of the latch enables a single Control FSM to execute every operation. When further developing the circuit into the microarchitecture, the memory words and encoded operation latch are renamed to registers and the Instruction Register (IR), respectively. This lab uses the microarchitecture terminology (i.e. registers and Instruction Register).

In the Lab [Part 1]

Load the supplied Lab3Part1 circuit into Logisim. After opening the file, Logisim should look like this:



Implement the Simple Processing System circuit (Figure 21 in Towards Hardware) using the Logisim components included in the original Lab3Part1 circuit (i.e. those shown on the canvas above). You must not add or exchange any components EXCEPT for a few simple gates to support interfacing to the SEL and LD pins (discussed below). I.E. You may add Wires, Tunnels, Splitters, Probes, and Constants from the Logisim Wiring Library, and the permitted gates (for SEL and LD interfacing), **but nothing else!** The ALU and Reg32 components are from the (included) Lab Support Library.

The ALU implements operations according to the OPR encoding presented in class (recall Figure 20).

The Reg32 component implements a 32-bit instance of the n-bit Data Latch abstraction shown in Figure 10. The names on the Reg32 pins are the same as those used in class, with the exception of “Debug Q”. The Reg32 data outputs are controlled by the OE signal (as expected), and therefore the value stored in the component is only visible externally when OE is asserted. Debug Q outputs the currently latched value at all times, and is intended to allow you to monitor the contents with a probe while testing/debugging. Do not use the Debug Q output for any other purpose.

The Reg32 component has a Clear (CLR) input that should be 0 during normal operation. If CLR = 1, then the internal latches are reset to 0. You may find it useful to reset the latches while testing. The Logisim Button included with the Lab3Part1 components has been provided for this purpose. Do not use the Button for any other purpose.

The RAM component has address and data connections (as expected) and is controlled using the SEL and LD inputs. The control provided by these inputs is a bit different than bank of words developed in class! Design and implement some simple logic using only a few logic Gates to interface the WordR and WordW signals from the Control FSM to the SEL and LD signals to implement Read and Write operations.

Use the Toggle Switch flip flop as the system clock.

Your implementation should label the components and signals with the same names used in class (see Figure 21).

Use a ROM-based implementation of the Control FSM (as you did in Lab 1). The state machine should cycle back to its first state after completing each operation. A word in ROM can be used to generate all of the outputs for a given state (i.e. specific bits in each ROM word will correspond to a specific output signals from the FSM). Since each operation requires a unique FSM, the ROM must be loaded with specific values for each operation (hint: 2 operand operations will use 3 ROM words, while 1 operand operations will use 2 ... sound familiar?). Refer to pages 18-25 for various examples of the ROM design process.

A next state table (like the one below) can be used to design ROM memory values:

State	OPR (4 bits)	Addr (4 bits)	WordR	WordW	T1CE	T1OE	T2CE	T2OE	Next State
E0									
E1									
E2									

To test your circuit design, use the values from Part 1 in your Prelab (see end of this document). Start with RB set to 0 and execute the 3 operations in the sequence shown.

In the Lab [Part2]

The circuit in Part 1 requires a unique FSM (i.e. unique ROM contents) for each operation. In this part you will introduce a 16-bit Instruction Register (IR) to generalize the FSM. Introducing the IR allows the same ROM contents to execute all operations. Use a Logisim Register to implement the IR, and poke values into the register as needed. You will need some tristate buffers in this part.

The instruction will have the following fields:

OPR (4 bits) Destination Reg (4 bits) Source X Reg (4 bits) Source Y Reg (4 bits)

A next state table for the Control FSM that executes every instruction in this part will look like this:

State	AOP	ANOP	DR	SXR	SYR	WordR	WordW	T1CE	T1OE	T2CE	T2OE	Next State
E0												
E1												
E2												
E3												

As discussed in class (refer to pages 25-28 in Towards Hardware), your FSM in Part 2 will have an additional “Decode” state, and you will have to redesign the FSM state transition logic to take appropriate action.

Implement Part 2 in the lab. You may include additional Logisim components, **but only** to account for behaviour associated with the Decode state in the Control FSM and incorporating the IR. To execute an operation, poke a value into IR, and then poke the Toggle Switch to advance the FSM through the operation.

Set RB to 0 and execute the 3 operations from Part 2 in your Prelab.

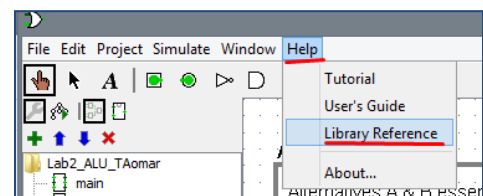
Before leaving your scheduled lab session, demonstrate what you have working to the TA.

NOTE: The possibility that you might not get the complete circuits working in the lab should influence how you build your circuit! Have an implementation plan that allows you to build and verify your design **incrementally**.

Prelab

This lab and its Prelab focus on the construction and operation of a Simple Processing System, as discussed in pages 18-28 in [Towards Hardware](#).

- In your own words, describe how the SEL and LD pins are used to control Read and Write operations on the RAM.
[hint: consult the Logisim docs]



- Complete the provided Lab3F16-Tables document.
 - Complete the Part 1 Output Tables for the following operations:
 $R9 \leftarrow \text{NOT}(RB)$
 $R8 \leftarrow RB - R9$
 $RA \leftarrow R8 \text{ XOR } R9$
 - Complete the Part 2 Output and Encoded Instruction tables for the following operations:
 $R9 \leftarrow \text{NOT}(RB)$
 $R8 \leftarrow RB - R9$
 $RA \leftarrow R8 \text{ XOR } R9$
 - [hint: pages 18-25 have examples relevant to Part 1; pages 25-28 for Part 2]

Good Luck! ☺