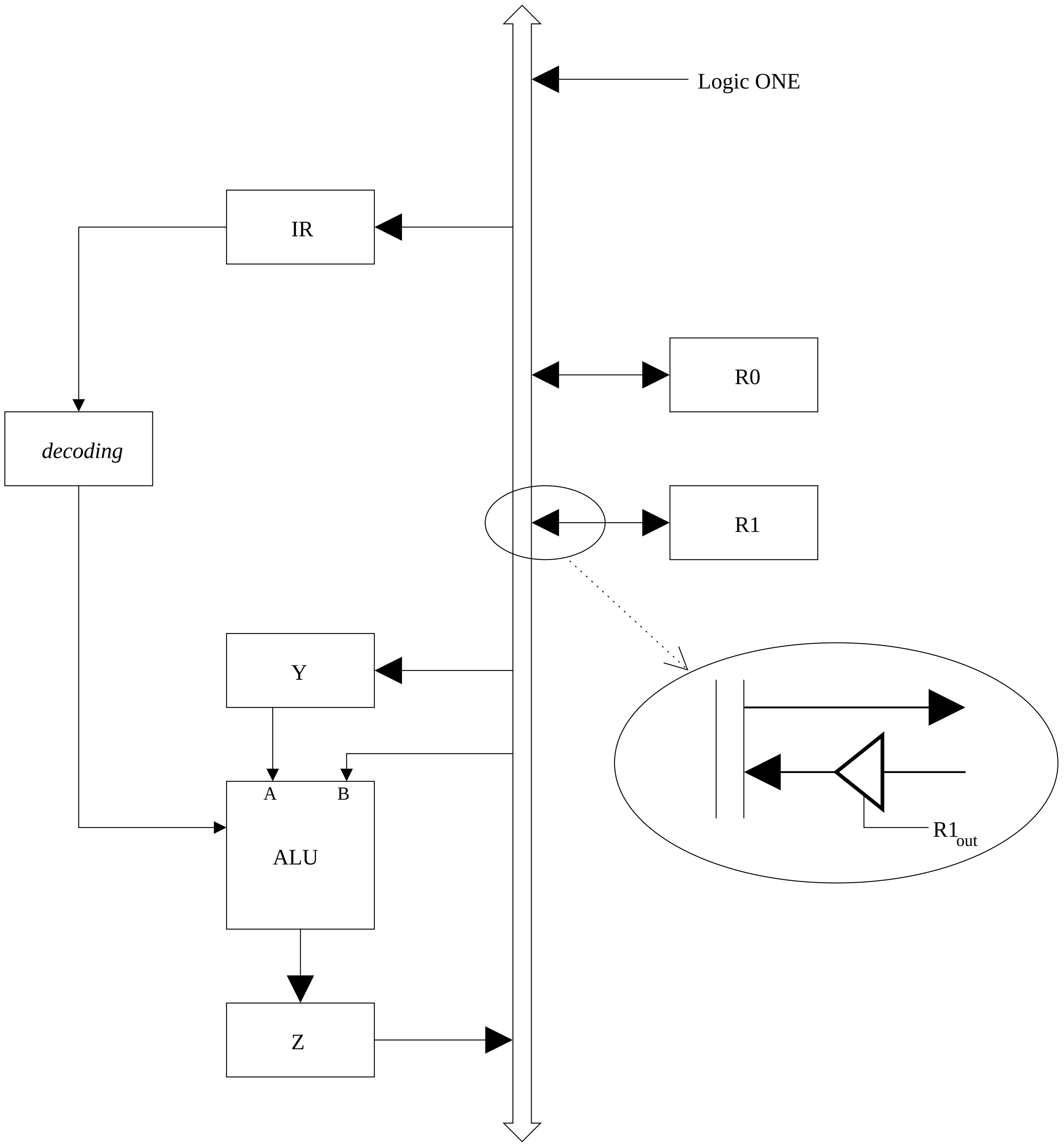
CSC 258 Lab 3

The figure below shows a very simple single-bus CPU.
In this lab you are to construct and test a 1-bit slice of this CPU.



This figure omits a lot of detail, including all of the control signals (indicated by large arrowheads). It is your task to include these control signals as well as the data lines shown. In a real CPU, instruction decoding circuitry would generate the control signals; in this lab, the contents of the IR will determine which function is to be performed by the ALU but all other control signals will be generated by hand, using the input switches.

(over)

You will need nine control signals: ONE_{out}, IR_{in}, $R0_{in}$, $R0_{out}$, $R1_{in}$, $R1_{out}$, Y_{in}, Z_{in}, and Z_{out}. You will not need a clock signal (the LOAD line for the particular register(s) will suffice for manual control).

Since there are only eight normal input switches, you will have to use the pulse input switch for one of these nine inputs. It will have to be an ‘‘in’’ control line. I suggest IR_{in}.

The registers R0, R1, Y, Z, and IR are one bit wide and should be implemented using D flip-flops. The Clock line is the ‘‘LOAD’’ line. Connect the outputs of all of the registers to LEDs so that you can see their contents. For the ALU, implement only the boolean AND and XOR operations (since we have only a single bit with which to select the ALU function). Since 1 XOR 1 is 0, you can use the logic one input to set up any register with its desired initial value. And there is nothing to do for ‘‘decoding’’, for this very simple CPU.

Note that the control input of the tri-state bus drivers used in the lab is active-low (i.e. the control input means ‘‘output disable’’ instead of ‘‘output enable’’), so that the output is driven when the control line is 0. Also, some of the tri-state drivers used in the lab are designed for either 2 bits or 4 bits of data per control input. You will want to use only 1 data line per control.

Your writeup should describe the manner in which the equipment is actually used, of course.

DEMONSTRATION

Run through the sequence of steps required to perform the following operations. For each step, always begin by switching on the appropriate ‘‘out’’ control line, then turn the ‘‘in’’ control line(s) on then off.

$R0 \leftarrow 1$	(‘‘MOVE #1, R0’’)
$R0 \leftarrow [R1]$	(‘‘MOVE R1, R0’’)
$R0 \leftarrow 0$	(‘‘MOVE #0, R0’’ — use $1 \oplus 1$ to do this — three steps)
$R0 \leftarrow [R0] \cdot [R1]$	*(logical and)*
$R0 \leftarrow [R0] \oplus [R1]$	*(exclusive or)*

Execute both ALU operations with a variety of values in R0 and R1 to show that your circuit works correctly.

Hint: If you have difficulty debugging your circuit, first test the ALU as a separate unit, then concentrate on data transfers on the bus. (Actually, as suggested in the lab notes, you should have tested your components separately before interconnection wherever possible.)

WRITEUP

- Give a complete schematic (logic) diagram of your CPU, including all control and data lines. Explain your design decisions and give a truth table of the ALU’s operation.
- Write out the sequence of control steps needed to perform each of the above five operations in the style we’ve been using. Each step will be analogous to a single cycle of the system clock.
- Draw a timing diagram showing the control signals used in the $R0 \leftarrow [R0] \oplus [R1]$ operation.