Homework 6

Building the MARIE Architecture: Part 2

Due Date: 11:59 p.m., Tuesday, March 15th.

Step 0: Setup

Start with your Logisim circuit from the previous assignment (or you can use my solution). Make a copy and name it `part_2.circ`. Make sure that you can transfer values between the registers by manipulating the `Write Control` and `Read Control` values. If this part isn't working, or doesn't make sense, then you should come to my office hours ASAP.

Step 1: ALU

Create the MARIE ALU as a subcircuit in Logisim. Build it using the built-in `adder` and `subtractor` parts, and a multiplexer.

- Create a subcircuit named `alu`.
- The `alu` circuit should have two 16-bit inputs `A` and `B` for data operands; you will eventually connect the MBR and ACC registers to these inputs.
- The `alu` circuit should have a two-bit input `F` to select the ALU function, which is described in Table 4.8 on p. 237.
- The `alu` circuit should have a 16-bit output `Q`, with the result of the calculation specified by `F`.

You can test your ALU using the `alu_vector.txt` file as a test vector.

Step 2: Memory

Add a memory to your main circuit. Use the built-in `RAM` part. As specified in MARIE.pdf, it should use 12 address bits and 16 data bits. Choose "Seperate Load and Store Ports" for the data interface.

Hook the RAM to the bus using a `busif`. Make sure to specify the correct bus ID for the memory. The data ports connect to the interface much like the `D` and `Q` ports of the registers. You can use the `Read Enable` pin of the `busif` to drive the `str` input of the memory, and ignore `sel`, `ld`, and `clr`.

Temporarily connect an input pin to the RAM's address input and verify that you can transfer data between memory and the registers using bus commands (i.e., manipulating the `write control` and `read control` inputs and the clock.) Then delete that input pin—you'll replace it with input from the MAR in the next step.

Step 3: Memory Address Register

1. Add a 12-bit register and label it "MAR".
2. Connect it to the bus using a `busif`. While the registers in the previous assignment were 16 bits, this is a 12-bit register, so if you try hooking the MAR up to the `busif` directly, you will get a "bit width mismatch" error. You can solve this by using a `bit extender` part.
3. *In addition* to the connection to the `busif`, you should connect the output of the MAR to the address input of the RAM.

You should now see that changing values in the MAR changes the output of the memory.

Step 4: AC and MBR

1. Add two more registers, both of them 16-bit, and label them AC and MBR.
2. Connect them to the bus using two more `busif` parts.
3. Add an `alu` circuit—the one you just created.
4. Connect the `A` input of the ALU to the ouput of the AC, and connect the `B` input to the output of the MBR.
5. Add a two-bit "ALU Function" input to your circuit. Add a 16-bit "ALU Ouput" output to your circuit.

You should be able to add or subtract the contents of the AC and MBR by changing the "ALU Function" input. You should also be able to transfer values between the MAR, MBR, AC, and memory by changing the "Read Control" and "Write Control" values.

Turning it in

Make a folder labeled "Homework 6" in your turn-in folder and copy in your `part2_circ`.