CS 310 - Fall 2020 - Assignment 4: Single-cycle implementation of the Larc ISA

Total points: 100

Due: Thursday 10/22/20 BY 8:00 AM (NOT the beginning of class)

You must work in pairs on this and the following assignments and submit only one electronic copy per team.

For this assignment, you have to implement in our HDL the single-cycle implementation of the Larc computer that we discussed in class. For full credit, each chip must work properly in all cases and the corresponding HDL file must be clearly organized.

Preparation

1. Study slide sets #7 and #8 carefully.
2. Download the `a4.zip` file from Canvas.
3. Unzip this file and change to the newly created `a4` directory, in which you will do all of your work, namely complete the two files `Larc.hdl` and `CU.hdl`. In addition to the files needed by the simulator, the code handout contains four test files that you should load into the simulator and trace step by step. I will test your submission on these and additional test programs.
4. Copy your file `RegisterFile.hdl` from assignment 3 into the `a4` directory and change the names of the registers in it to `RegisterR0`, `RegisterR1`, ..., `RegisterR15` so that you take advantage of these built-in registers and their associated widgets in the simulator window.
5. Do NOT copy your other memory chips from the previous assignment since you want to use the built-in ones and their associated widgets in the simulator window.
6. Copy your `ALU.hdl` and `ALU1bit.hdl` files from assignment 2 into the `a4` directory.
7. Finally, you will also need to copy a small number of multiplexers and other 16-way chips that you implemented in assignment 2. Most of the chips you need are built into the simulator code and you should NOT copy them into the `a4` directory, even if you re-implemented them in a previous assignment. In fact, the best way to proceed is to use the chips you need and only copy them into the `a4` directory if the simulator displays an error message about not finding the chip.

Make sure to test each new chip fully. It is your responsibility to make sure that each chip works in all cases according to its intended function, as described in class, even if the provided test files do not cover all possible cases. **For full credit, your Larc chip must follow exactly the schematics included in the relevant slide sets (unless explicitly superseded by instructions in this handout) and all of your chips must use the smallest number of chips needed to meet all of the requirements of the chip's API**. For example, if one of the components in the implementation of a chip behaves like a 4-way 16-bit-wide multiplexer, you must use a single chip (namely, Mux4Way16) for this component, instead of re-implementing it with basic gates.

Here is a list of the chips that you must implement, some built-in chips that you must use and the other digital circuits you must implement for this assignment:

1. `Larc.hdl`: Main file to be completed to meet all of the requirements described in slide sets #7 and #8. Furthermore:
   - This chip must include a digital circuit with the smallest number of chips that enforces the constraint of the Larc ISA that the first register (i.e., register 0) always contain the value 0, even if the user program attempts to modify this value. For full credit, your implementation file must contain a short but precise comment that explains how you designed this circuit.
   - This chip, in combination with the following one, must contain a digital circuit that implements the HALT instruction with opcode `1111`.
2. `CU.hdl`: The other chip to be implemented.
3. `RegisterFile.hdl`: Your chip from assignment 3, with the register chips renamed (NO other changes are allowed in the register file).
4. `PC`: A built-in chip you must use for the Program Counter register. This chip is associated with a GUI widget that displays the value stored in the register.

5. `RAM4K`: A built-in chip you must use for the instruction memory (IM). This chip is associated with a GUI widget that displays the value stored in each memory location.

6. `RAM16K`: A built-in chip you must use for the data memory (DM). This chip is associated with a GUI widget that displays the value stored in each memory location.

Submission

When you have implemented and fully tested all of the chips above, follow the submission procedure:

1. Pick the drop box of one of the team members.

2. In their local directory, create a separate "submission directory" called `a4`.

3. Copy all of the `.hdl` files needed to complete this assignment. This directory should only contain the two `.hdl` files I provided, and one `.hdl` file for each non-built-in chip needed by at least one submitted chip.

4. Submit a SINGLE copy of your work in the drop box chosen at step 1: move to the parent of your "submission directory" `a4` and zip it up to produce a file called `a4.zip`.

5. For full credit, the `a4` directory resulting from unzipping the submitted zip file must contain only the files specified in step 3 above. Note that all of these files are `.hdl` files. **Make sure to delete ALL extraneous files before submitting your `a4` directory**. Furthermore, each `.hdl` files must load AS IS in the simulator and pass each and every one of my tests. If I have to copy extra files that you did not submit, you will lose points.

6. For full credit, each and every HDL file must be formatted as follows:

   - No line may be longer than 80 characters.
   - Use the full power of HDL to minimize the length of your implementation (e.g., use bus notation rather than single-bit notation whenever possible).
   - If a line is still too long, split it after a comma and indent the continuation lines as follows:

     ```
     Chip1( a=input1, b=input2, c=input3,
            d=..., e=..., ...
            z=...);
     Chip2( ... )
     ```

     Note that all chips must start in the same column. Incorrect indentation makes your files hard to debug and grade and will thus cost you a significant number of points.
   - In the two new files you have to implement, you must precede each and every chip with a concise, one-line comment identifying the chip on the schematics provided in the slides, e.g., `// Mux at WD`.

As always, the late-submission policy will apply. Start early and ask questions if needed.

Good luck and have fun!