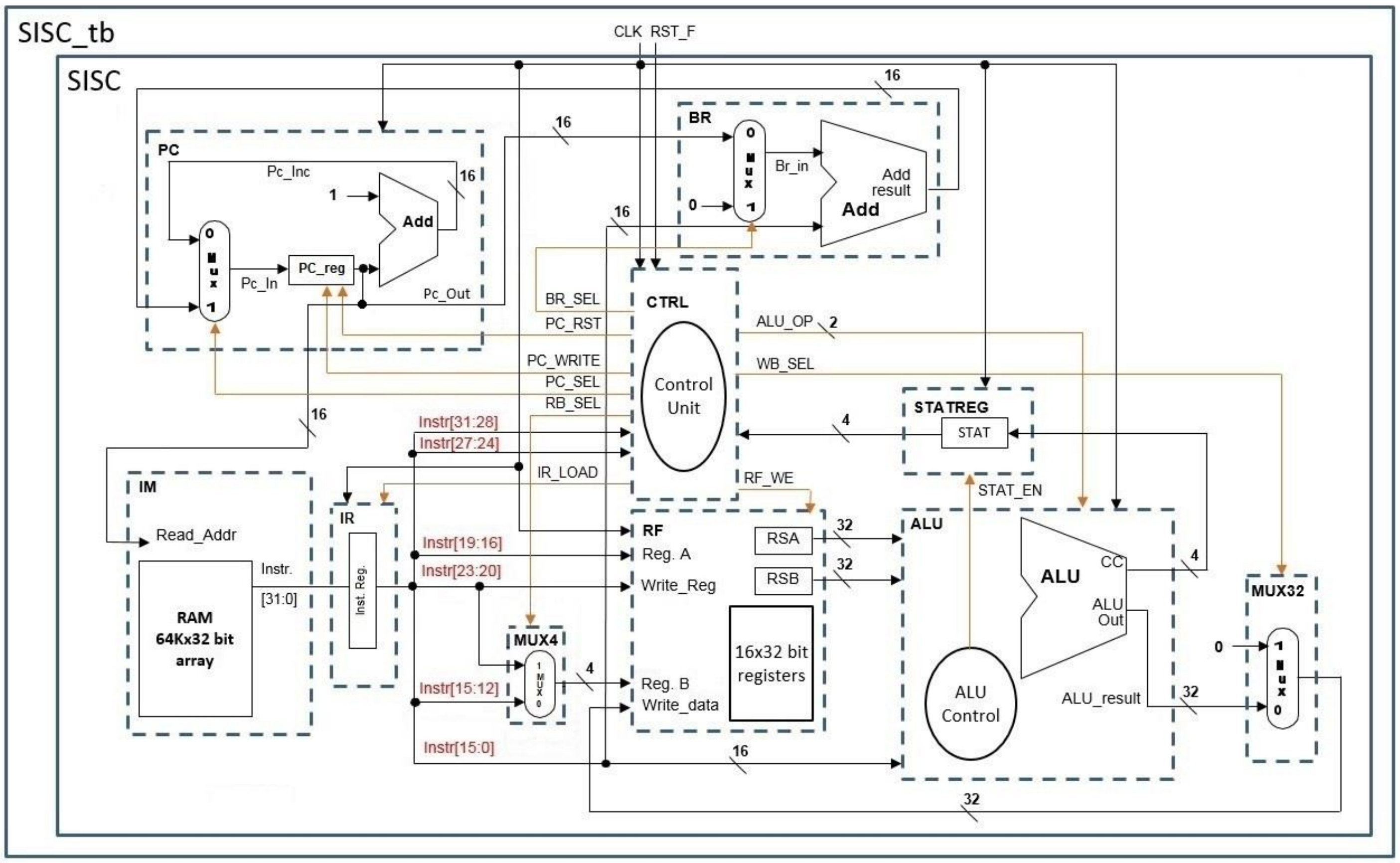
ECE:3350 Spring 2019 -
Computer Architecture and Organization
Simple Instruction Set Computer (SISC) Project

Part 2: Due Monday, March 11, @ 8:30 am

For this part of the project, you are to include in the datapath the modules for program counter control, instruction memory, and branch execution, as shown below.



You are given:

- Your own solution from Part 1. We will not provide a working solution to build off of, so **it is imperative that you finish Part 1 before moving on to Part 2.**
- An imem.data file that contains all of the instructions from Part 1. You are provided this so that you may be certain that your modifications from Part 1 to Part 2 have not broken the original functionality. (See the note about imem.data below for more information.)
- The Verilog files pc.v, br.v, ir.v, and im.v (with descriptions provided), which you are not to modify.

You are required to:

- Modify the sisc module to instantiate and connect the pc, br, ir, and im modules.
- Modify the control unit to generate the new RD_SEL, BR_SEL, PC_RST, PC_WRITE, PC_SEL, and IR_LOAD signals, while not disrupting the control signals from Part 1.
- In the ctrl.v module, implement instruction fetch, and the branch instructions.
- Update the FSM state machine you created for Part 1 to reflect your new ctrl.v.
- Compress your project folder and submit it to the "Verilog Project – Part 2" ICON dropbox.

Part 2 Notes:

- You are given a new testbench file that only generates the clock and reset signals.
- **About imem.data:** When the simulation of your design begins, the im module reads the data in the file imem.data and stores it. The data is written in 32-bit hexadecimal strings (8 hexadecimal digits each), sequentially from address 00 onward. In-line comments can be included if they are preceded by two forward slash characters (//Just like in C). You may want to keep the original contents of imem.data as a reference of how to format the instructions. The imem.data file also contains new instructions to test that the branch instructions work correctly.
- You will no longer have to control the timing of the instructions with delays. If your control unit correctly generates the program counter and branch signals, the instructions will be read during the instruction fetch stage by the im module.
- **Be careful when implementing the PC update and branch control signals!** The PC must be incremented as part of the Fetch state. This ensures that the PC has the correct value at the end of instruction execution for all instructions that do not modify the PC. In addition, this ensures that the incremented value of the PC is available to the BR module at the beginning of the Decode state. For the branch instructions where the branch is taken, the PC is again updated in the Decode state with the new PC value originating in the BR module
- Again, you may use any $monitor(…) statements you wish to test your design, but your submitted sisc.v should monitor the following signals: IR, PC, R1 through R5, ALU_OP, BR_SEL, PC_WRITE, and PC_SEL.
- After simulation, save your transcript file!

Submission Overview:

- Your .zip file should be named "Project_p2.zip" and contain the following:
    - A description of your part 2 design and the names of your project partners.
    - Alu.v, br.v, im.v, mux4.v, mux32.v, pc.v, ir.v, rf.v, and statreg.v, exactly as they were provided.
    - Ctrl.v, and sisc.v completed by your group.
        - Again, sisc.v should contain the $monitor statement described above.
    - The 'work' directory.
    - The transcript information from your simulation.
    - ModelSim screen captures.

- Your updated FSM state diagram.

Grading Rubric:

Revised FSM Diagram	10 pts
Correct Execution of Pt. 1 Instructions	15 pts
Correct Execution of Pt. 2 Instructions	30 pts
Ctrl.v Implementation	15 pts
Total	**70 pts**