Systems I
Lab 8
A RISC-V processor — The Final Project!

1 Opcodes and instruction types to implement

Here is a listing of the instruction opcodes that should (and could) be implemented in your processor. For example uses of any of these, search (e.g., `risc-v slli example`).

1.1 Required opcodes

There are instructions that we have discussed at length, and whose implementation are the core of our datapath design. All of these should be implemented.

- **R-type:**
  - `add`: Addition
  - `sub`: Subtraction
  - `and`: Bitwise AND
  - `or`: Bitwise OR
  - `xor`: Bitwise XOR
- **I-type:**
  - `addi`: Addition w/ immediate
  - `andi`: Bitwise AND w/ immediate
  - `ori`: Bitwise OR w/ immediate
  - `xori`: Bitwise XOR w/ immediate
- **B-type:**
  - `beq`: Branch if equal
  - `bne`: Branch if not equal
  - `blt`: Branch if less than
  - `bge`: Branch if greater than or equal

1.2 Optional opcodes

You may implement these opcodes as part of your final project, you are not required to do so. However, you should be able to describe how each of these opcodes *could* be implemented, detailing the changes that would be required to the datapath & control. They are presented in the order in which I recommend you consider them. Specifically, the first two groups (*main memory access* and *register loading*) are likely sources of exam questions.

- **Main memory access:**
  - `lw`: Load word
  - `sw`: Store word
- **Register loading:**
  - `lui`: Load upper immediate
  - `auipc`: Add upper immediate to program counter
- **Bit-shifting:**
  - `sll`: Shift left logical
  - `srl`: Shift right logical
  - `slli`: Shift left logical immediate
  - `srli`: Shift right logical immediate
- **Unconditional branching:**
  - `jal`: Jump and link
  - `jalr`: Jump and link register

2 How to submit your work

2.1 Test cases

You should create one or more test programs that demonstrate the various opcodes being correctly decoded and executed. For each such test program, you should have a written listing of what values should appear (and in which registers or at what memory locations) with each step of the program. Suffice it to say, these test cases should be no longer than necessary.

2.2 Making a video

Create a narrated video that walks the viewer through the test program(s), highlighting what happens at each step and the evidence that the program is being executed correctly.

2.3 How to submit it all

There are a few steps for each group to submit its work:

1. **Add me as a collaborator on CircuitVerse:** Be sure to take the final processor circuit, with all its parts and test-program EEPROM(s), and add `sfkaplan@amherst.edu` as a collaborator.

2. **Copy all relevant files into a shared Google Drive folder:** Choose one member of your group who will make, in their shared Google Drive folder for this class, a subfolder named `final-project`. In that folder, place copies of:

   - The JSON file(s) for the EEPROM(s) created from your test program(s).
   - The assembly of your test program(s), annotated with the expected changes/behaviors that occur with each step.
   - The video of your walkthrough, running the test program(s) on your processor.

This assignment is due on May-05, 11:59 pm.