CSC 270 Final Project – Build a CPU

Due Friday, March 14, 2025 at midnight

Commitment form due Monday, March 3, 2025 at 9:15am

The goal of this project is to apply your knowledge of computer architecture to the task of designing and implementing an 8-bit (or larger) CPU architecture in Logisim.

You may use any of the built-in Logisim components listed below:

- anything in the Wiring library, which includes probes, tunnels, and the sign extender
- any gate in the Gates library
- MUXes and DEMUXes in the Plexers library
- Full Adders, comparators, and shifters from the Arithmetic library
- Registers and RAM from the Memory library

This still means you need to build your own register file and your own ALU. Remember, an 8-bit computer means RAM addresses are now 8 bits and registers can hold 8-bit numbers.

Project evaluation will be based upon how many of the following features you successfully add to your CPU. Correctly implementing the basic project with no "extra" features will net you a maximum 70% passing grade.

Required: (**70 points)** An 8-bit single cycle CPU with all of the components we built in lab:

- A "basic" ALU capable of addition, subtraction, and logical operations AND & OR.
- Support for both R-type and immediate instructions.
- A register file with a minimum of 8 registers.
- Instruction memory, program counter, and data memory.
- Control Logic Unit to decipher opcodes
- An assembler/linker, written in Python, which converts your assembly code into hexadecimal machine language which runs on your CPU, and which can be loaded into the instruction memory of your CPU in Logisim.
- 5-10 page writeup described at the end of this document, along with a README file describing what your assembler does.

Note: I will be providing basic "test" assembly code that you will have to assemble and then run on your processor, providing proof in your writeup.

Pick-and-choose upgrades: (30 points expected; 45 points maximum)

(*) means that the "basic" assembly test code should work with this feature
(**) means that I will provide separate test code to test this feature.
(***) means that you must provide your own test code, explaining what the state of registers should look like if your instruction passes/fails the test.

- (*) A 16-bit CPU **(5 points)**
- (**) Branch-if-equal (beq) **(5 points)**
- (**) Branch-if-not-equal (bne) **(5 points)**
- (**) Jump (j) Instruction **(10 points)**
- (**) Support for SLT **(5 points)**
- (*) Carry Look-Ahead Adder **(10 points)**
- (**) Support for function calls (Jump-and-link, and jump-register) **(20 points)**
- (**) Support for a stack (via $sp) (**10 points**)
    - requires support for JAL and JR
    - the test code for this will involve recursion
- (***) Fancy Assembler/Linker (in Python):
    - uses labels (**5 points**)
    - support for pseudo-instructions (move, blt, ble, bgt, bge) (**1 point each, 5 pts max**)
    - can assemble and then link multiple assembly files (with cross-referenced labels) into a single hex file (**10 points**)
    - can assemble instruction (.text) and data (.data) into separate hex files for Logisim. (**5 points**)

You can get a maximum of **115** (i.e. 15% extra credit) on this assignment. You should be certain you've earned 100% before going for more. You can ONLY claim up to 115 possible points in your grade sheet! Any features beyond the 115 point limit will be ignored.

Additional Details:

1. You will be asked to make design decisions (such as how many registers to provide) using a Commitment form available on Nexus. Due date for that is at the top of this document.

2. You will have two .circ files. The first will be your "basic" 70-point processor which will have NO upgrades, unless you are going for the 16-bit processor. Then the "basic" version can be the same number of bits, which will save you some effort. The second .circ file will be your fully-upgraded processor. Both versions should be modular, where you create and use subcomponent circuits in the same .circ file. As always, subcircuits should be labeled with what they are and what each of their pins are.

3. **Create the 70-point version first!** Do not try to make the enhanced version first and then "erase some stuff" to "easily" make the basic version. That's a bad design decision. You won't find it easy nor just a matter of erasing some stuff.

4. Your writeup must contain a "grade sheet" describing, in a simple table, each feature you implemented successfully, and how many points you are requesting for it. In the event that a feature is partially implemented, be modest in requesting points. The template you must fill out is on Nexus.

How to turn in this project

The following files should be placed into a folder, which should then be zipped up and uploaded to Nexus.

- the two .circ files
- test files, both .asm and .hex, that YOU have written to test your features. Don't just rely on the assembly tests I've written -- write your own too!
- the "grade sheet" mentioned above, where you describe each feature you've partially/successfully implemented, and request a certain number of points for each.

- A 5-10 page description of your processor design. It should include

    - proof that implemented features work. This means:
        - screenshots that my test code behaves as expected for each feature for which you are expecting credit.
        - ditto for your own test code.
    - description of design decisions you made along the way
    - description of your instruction word format
    - a mapping of assembly code instructions to corresponding machine code (like the tables in Appendix B in your text).
    - directions for translating assembly code into hexadecimal machine code
    - anything else that a programmer would need to build a compiler for your processor

Grading

This is worth 100 points and counts for 18% of your grade!

- 10 points for each of the bullet points under the **Required: (70 points)** section above
- 30+ points for the upgraded features. Point distribution varies. See the **Pick-and-choose upgrades** section for details.

Gentle Reminder

Unless otherwise specified, lab projects and micro-homework assignments are ***individual*** projects. I encourage you to talk to others about the general nature of the project and ideas about how to pursue it. However, the technical work, the writing, and the inspiration behind these must be substantially your own. If any person besides you contributes in any way to the project, you must credit their work on your project. Similarly, if you include information that you have gleaned from other published sources, you must cite them as references. Looking at, and/or copying, other people's programs or written work is inappropriate, and will be reported to the Honor Council chair.