cs281: Introduction to Computer Systems

Y86 CPU Project

1 Overview

In this lab, we build our own logisim model of a fully functioning CPU to execute Y86 instructions. You are given components of the machine to use. You will combine these with the appropriate control logic to make your CPU function correctly.

You will work with a partner for this project. It is important that you and your partner divide the work appropriately and both contribute actively to the project. You should not receive assistance or share information with any other students or people besides your partner or the course assistant/professor.

The project is due on Monday, April 15, at 11:59pm You and your partner will demonstrate the correct operation of your circuit to the course professor by the end of day (5pm) on Friday, April 19th. You will also submit your circ files for evaluation by the Monday deadline.

2 Components

You are given the following components. You should not modify them in any way. Use them in your CPU circuit and design the appropriate logic around them.

- **y86.circ**. This is your main circuit file. Obviously you will modify this file as you will add a substantial number of other pieces. Please keep this file named as `y86.circ`.
- **alu.circ**. This is your main ALU file.
- **imem.circ**. This is your instruction memory. It should already be placed on your y86 circuit. There are four banks of memory in this circuit. Though Y86 is a 64 bit architecture, this memory device is addressed with the lowest 32 bits of addresses. Look inside the file to see how it works.
- **dmem.circ**. This is your data memory. It is divided into 2 banks. It is addressed with a 64 bit address and reads/writes 64 bits of data.
- **regs.circ**. This is your register file. It is a complex circuit with many inputs and outputs. Be sure to study the circuit and your textbook to fully understand its operation.

You are also given the following files:

- `yo2dmem.c`: Converts `*.yo` files into two banks of data memory.
- `yo2imem.c`: Converts `*.yo` files into four banks of instruction memory.

These two programs will be helpful for loading y86 programs into memory for testing purposes. Read the source code and follow the in-class demonstrations.

3 Design

- Use "tunnels" extensively to organize your circuit and keep the number of spaghetti wires to a minimum.
- Put a debugging section at the bottom of your circuit so that you can see all the important values quickly while debugging with programs.
- Keep things neat and organized. Position smartly. Wire smartly.
- Use the following values for your stat registers:

Value	Status
00	Normal operation
11	Halt (overrides all other errors)
01	Invalid instruction (overrides all memory errors)
10	Invalid address for instruction or data memory.

Complete the Instruction Design spreadsheet so that you have, in one place, the control information needed for all of the instructions.

For Spring 19, the spreadsheet can be found at:

`https://docs.google.com/spreadsheets/d/1Gu5ojtUyyw9OHyzlcg5g3r7fUn_LO0QoLSgjFp6Vp_`

This link is also available on Notebowl.

4 Debugging

You will want to reserve time for debugging and testing (see suggested schedule in next section). I recommend that you write a series of small programs that add increasing layers of complexity. Load each program into memory and test that new feature. Expect that you will discover errors in the circuit design and you will need time to fix them. Start with the simplest instructions and move towards the more complex ones. Be sure you test ALL kinds of instructions. After you have tested each one, then write a much longer, extensive test program that tests everything fully. Decide what programs you will use to demonstrate your circuit after April 15.

5 Project Management

This is a significantly large project, hence you and your partner have more than two lab times and additional time beyond to complete it. Even then, it will go much smoother if you set some parameters ahead of time. The temptation will be to jump right in and get started, but it would probably be helpful to agree on some timelines and strategies first. Here are some suggestions.

1. You and your partner will have to agree upon an equitable workload distribution. I suggest the following:
   - Person 1 Fetch, Decode and Writeback. These stages work closely together and involve the complex register file.
   - Person 2 Execute, Memory, PCupdate, Stat. These stages work together well too.

   Agree on deadlines before you start, not after you think your partner is late with their work. Hold each other accountable to the deadlines. If your partner is not keeping to the schedule, have your group consult with the course professor. I recommend that you write your timeline down. One possible timeline:

Stage	Date
Fetch	April 7
ALU	April 7
integrate	April 7
Decode, WriteBack	April 9
Memory, PCupdate	April 9
Justify Instruction Design Table and Demo irmovq to Hayley	April 10
** integrate **	April 11
Group Debug	April 13
Group Testing	April 14-15
Final Demonstration	April 16+

   The **integrate** sessions are agreed upon date/times for getting together and combining the separate circuits into one single diagram.

2. Agree upon a set of names for global signals. You will use ”tunnels” extensively in your circuit, but you will need to share signals across different components. Having an agreed upon set of names will make things go more smoothly when you integrate.

3. Check in regularly with each other. Nothing is worse than going off by yourself, doing a whole bunch of work, and then finding your partner had a different idea of your design and your work isn't compatible. If you check in a couple of times per week, and use the integrate sessions appropriately, you should avoid much of the conflicts from incompatible assumptions.

6 Deliverables

You will submit your y86.circ file and any subordinate files (beyond the ones given to you) to the notebowl assignment. You will not submit a lab report. You and your partner will demonstrate the correct operation of your circuit to the course professor at a time scheduled the week of April 16 to 19. Be sure to have your circuit ready with a loaded program. You will have only about 15 minutes, so carefully create one or two programs that fully demonstrate the operation of your CPU. Failure to demonstrate a particular instruction will imply that instruction is not operating correctly and you will not receive credit for it. Practice your demonstration program before your scheduled time. Print out the .yo files for your programs. Know what register and memory values to look for in order to verify correct operation.

7 Bonus

If you are so inclined, as an optional bonus you can add to your Y86 circuit by completing the `iaddq` instruction as outlined in Practice Problem 4.3 (page 369). You will earn credit if you can demonstrate the correct operation of this instruction in your CPU. This will add 5 percentage worth of points on to your project score.