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, November 18, at 11:59pm You will submit your circ files for evaluation by the Monday deadline. You and your partner will demonstrate the correct operation of your circuit to the course professor by the end of day (5pm) on Thursday, November 21.

2 Hardware Design 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`.
- **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.

3 Stages of Execution

I have provided you with slides that give basic templates for each stage of execution, but certain sub-circuits will require you to use what you have learned previously in the semester with truth tables and K-maps to reduce the total logic needed.

- **ALU**. In the previous lab, you had built your own 8-bit ALU. For this project, you will chain together 4 of these to build a 32-bit ALU. Two of these chained together will be necessary for you to perform operations on 64-bit values.

- **Fetch**. You will use a 32-bit pin to represent your Program Counter, which holds the address of the next instruction to execute. Use the design from the slides regarding the fetch as a layout guide when designing your subcircuits. Instruction memory is provided to you, but how will you Align? Decide if regs is being used for this instruction? Decide if V or D (value C) is required for this instruction? Lastly, you will need to use your 32-bit ALU to increment the Program Counter.
- **Decode**. If Fetch went well, you now have the values you need to set source A, source B, destination E, and destination M (Hint: E is a destination when writing back to a register from the ALU; M is a destination when writing back to a register from Memory). Use the instruction code (icode) to figure out which of (rA, rB) is being used to set these values.
- **Execute**. This is where your 64-bit ALU and overall condition codes will come into play. You will need to think carefully about how to design the condition circuit to handle the instruction function and result in a condition that will be used to move or jump.
- **Memory Access**. Memory may only be read or written in a single cycle. Use the instruction code to determine whether either is appropriate, and what the address or data might be. If an error results, this should feed back into the status of the CPU as a whole.
- **PC Update** What should the next value of the PC be? Based on the previous instruction code, the outcome from condition in the Execute cycle, and values (V or D), a value from Memory, or the previously calculated PC increment from fetch, you can design a subcircuit that updates the PC correctly.

4 Design Advice

- Use ”tunnels” extensively to organize your circuit and keep the number of spaghetti wires to a minimum.
- Focus on each stage of execution individually. Get one correct before you move onto the next.
- Put a debugging section at the bottom of your circuit so that you can see all the important values quickly while debugging with programs.
- 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.
- 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 Fall 19, the spreadsheet can be found at:

`https://docs.google.com/spreadsheets/d/17jcGPghyiyWdWWdGkzQo5v1Jp2Z9YWa2v_BKjLuobI`

This link is also available on Notebowl.

5 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.

You have been given the following files to help with loading programs into your memory banks:

- `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 to find out more.

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 November 18.

6 Project Management

This is a significantly large project, hence you and your partner have more than one lab time 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 deadlines and an equitable workload distribution. 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.
2. I suggest working this week on the ALU, Fetch, Decode, and Execute stages. A 32-bit ALU is necessary for Execute and Fetch, so I suggest one partner works on chaining these together while the other partner works on the Decode. Once the ALU is prepared, Fetch and Execute can be distributed between you and your partner.
3. That would leave Memory Access, PC Update, and debugging for week 2. While it is recommended to test each stage thoroughly when it is completed, likely some errors will not be found until all stages are complete.

4. 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.

7 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.