

Lab 3. 8-bit CPU – Fully functional! CSE 141L

In this assignment, you will finally design a single-cycle implementation of a processor to execute your 8-bit ISA. At a minimum, your design will have a program counter (PC), a PC incrementor, a register file, and an ALU, memory. The ALU and register file should not be significantly changed from the last lab. **YOU WILL BE REQUIRED TO DEMO/TEST YOUR DESIGN FOR A TA. PLEASE SIGN UP IN LAB MEETING OR TAT OFFICE HOURS FOR A DEMO TIME!**

Some things to keep in mind for the lab:

- Use hierarchical design (subcircuits) to make your design easier to understand and think about. The highest-level schematic should mostly be functional elements (register file, ALU, memory, etc.) and wires/buses. The lowest levels should be in terms of basic Xilinx components. Turn in diagrams of the ALU and register file again.
- Isolate control. Follow the text's lead by generating control signals in one place from the opcode. This should make the design easier conceptually, at least. For your single cycle processor, you only need to use combinational logic to set up the control lines.
- Keep in mind that you will have to debug your design! Think about how to make your life easier before it happens.
- Remember that it is your responsibility to convince us that your CPU works, not our responsibility to figure that out ourselves. Providing sufficient and clear results, information, and commented diagrams and code (your assembly language file for example) is crucial.

Memory and other new things to build:

- You will create a ROM to hold instruction memory and a RAM to hold data memory. Both will be initialized – the ROM(s) to hold your programs, the RAM to hold your input data. Instructions to create memory parts in Xilinx will be on the web page.
- You can actually create multiple RAMs to test various input sets. On the web site, find two input “sets” that I expect you to test. Remember to set these up in the appropriate “places” in your RAM. See Lab 1 for directions about exactly where certain values should go in memory.
- I want you to have an init signal that sets the PC to some 0, possibly initializes some registers to 0, etc. I also want a “done signal” that goes high at the *halt* instruction. This “done signal” should be shown on your waveform output.
- I also want you to have a **cycle counter**, which you will use to reliably determine dynamic instruction counts. It should be initialized to 0 at the start with your init signal (as part of the stuff that happens on the init signal).
- You should CHECK that your RAM has the correct output at the end of each test program (more details below). But you don’t need to provide any paper output.

You need to turn in the following items:

- Print out schematics of all circuits, hierarchically organized. The highest-level design needs to have all of the signals necessary to demonstrate correct program execution via the timing diagram.
- The description of your ISA, your assembly language program (commented and documented), and the data structures (registers and memory locations) used for the program.
- **Two timing diagrams** for your program **demonstrating correct operation on each input set (run it 2 times with the different RAMs)** and other important data. Once again, the timing diagrams should be annotated heavily. The test inputs for this are posted on the lab webpage.
 - It should at a minimum show execution of the program through at least the first few iterations of each loop structure that you have, the cycle counter, the PC. It need not show the whole execution of the program (particularly if it takes over 100 cycles or so! – again, cut out some of the loop iterations. It, once again, should be heavily annotated so we can figure out what is going on.
 - **You MUST point out any particular instructions or overall execution that is not functioning.** Give your explanation of what you think is going wrong.

Required Lab Format:

1. Introduction
2. ISA Review (remember, we haven't memorized your ISA!)
 - a. Assembly code
3. Data Structure Overview (subsection for each structure, you may have more to explain now – PC increment functionality, memories and how they connect, etc.)
 - a. Descriptions
 - b. What instructions use this and in what ways
 - c. Schematics, annotated (like Lab 2)
 - d. NO timing diagrams needed (we assume you got them working in Lab 2)
4. Branches
 - a. Branches are “new” in this lab, describe your branch functionality in detail.
 - b. Show a timing diagram of a branch taking (modifying the PC) and one falling through (where $PC = PC + 1$)
5. Two Timing Diagrams heavily annotated (showing behavior on the required input sets)
 - a. See details above (entire execution does not have to be shown)
6. Other details that you need to note.
7. Questions (numbered please, see below)

In addition, you need to answer the following questions:

In the questions for the lab report, I am no longer looking for you to convince me that your design is wonderful, but rather **I am looking at how effectively you critique your own design.**

Please number these questions in a “questions” section of your lab report.

1. If you made any changes to your ISA -- what were they? Why did you make them? (we hope you didn't, but if you did, this is the place).
2. What was the dynamic instruction count for your program using the hardware cycle counter on input A and input B? Please comment on this value for each one and explain where you get differences.
3. What could have been done differently to better optimize for dynamic instruction count? Give examples.
4. How successful were you at optimizing for ease of design and what was particularly difficult to design? Give examples.
5. What could you have done differently to better optimize for ease of design?
6. How easy/difficult would it be to extend your design to a multicycle implementation? A pipelined implementation? Give examples.
7. What might you have done differently if the priority was ease of assembly programming/compiler writing? Give examples.
8. What instruction takes the longest on your machine? This instruction would determine the cycle time of the processor. Use rough estimates.
9. What might you have done differently if the priority was short cycle time? Give examples.
10. State any known problems/bugs with the design in executing the programs. This will facilitate grading if we know problems ahead of time, and can allow you to receive more partial credit if something isn't working.
11. Reflect on this lab experience.
 - What did you learn from this lab?
 - What was the best thing about it?
 - What was the worst thing about it?
 - What advice would you give to someone taking this lab next quarter?
 - What would you do as the professor to make this lab a better experience?
 - How would you describe (in 3 sentences) the value of this lab experience in a job interview?