

CSE 141L: Project in Computer Architecture

Fall 2003

Lab 4: Construct & Test a Complete 8-bit Processor

Reports and electronic submissions are due before 1 PM on Friday, December 5th.

In this lab, you will design complete logic in LogicWorks 4 for the 8-bit processor architecture you developed in Lab 1 and test its functionality.

You will implement all instructions in your architecture including the control transfer instructions (branch, jump, etc.) and memory load/store instructions. You will test the three programs for which your architecture has been optimized.

It is your responsibility to make an appointment with one of the TAs to show that your CPU design works before above deadline. You should test the programs using the data patterns given in Lab1. The TAs may test your design for correct functionality using their own data files that satisfy the constraints outlined in Lab 1.

What you will turn in for this Lab

- Summary of your ISA from Lab 1 and assembly code with machine code for 3 programs.
- Printed schematics for the top level CPU as well as all the lower level modules you designed in LogicWorks 4.
- All the LogicWorks 4 files you created (to be submitted via electronic submission).
- Answers to following questions.

Design of the CPU

You designed the datapath for your 8-bit architecture in Lab3. You need to provide additional blocks around this datapath to complete the design of the processor. Note that the processor you design will not have a pipeline. Instead, all operations required to execute an instruction will take place in a single cycle. The following is a list of additional blocks you will need. The details of the blocks will depend on your architecture and you may need additional blocks.

- **Reset logic:** A reset signal must be provided to initialize the PC (generally to zero) and set any appropriate internal state registers (e.g. condition code) to a known value. You should not reset the contents of the general-purpose register to a known value, since your program(s) should not assume any initialized values in these registers.
- **Program Counter (PC) logic:** You must provide logic to appropriately update the value of PC every clock cycle. At the end of every non-control transfer instruction, the PC must be incremented by 1. In case of a taken branch or jump, the PC should be updated to the target of the control transfer instruction. Furthermore, a HALT instruction must be implemented to halt the processor at the end of a program.
- **Instruction Memory:** Use the procedure in LogicWorks 4 to create a PROM using the raw format hex files (the .imi files you created in Lab 2). Create one PROM file for each one of the 3 programs you have developed.

- **Data Memory:** You should create a generic RAM module using the LogicWorks 4 which is 8-bits wide and has appropriate number of address bits. The contents of the RAM can be initialized to a .dmi file following procedure outlined in lecture.
- **Datapath:** You will have to enhance the datapath you designed in Lab 3 to implement the memory load/store instructions and control transfer instructions. You also may have to modify it to supply immediate data fields from an instruction.
- **Control Logic:** Your design will have control signals in various blocks. Examples are multiplexor control, register file and memory write enable signals, ALU opcode, etc. Typically, the control logic block will have as input appropriate bits from the 8-bit instruction and some state information for your processor, such as condition code flag(s). The output of the control block will be the above control signals. The control logic, in general, will have only combinational logic.
- **Instruction Counter:** You must provide an instruction counter, which is initialized to zero when the reset signal is asserted. After the reset signal is deasserted, it must count instructions until a halt instruction is executed, at which point it should freeze.

Testing your CPU

You will substitute appropriate versions of the instruction memory and data memory in your CPU design to test a particular program. After the execution of the program is complete, you can view the contents of the data memory by following the procedure outlined in the tutorial on LogicWorks 4. The TAs may also check the simulation waveforms to verify that your design is functioning correctly.

It is highly recommended that you write a simple program to test all the instructions in your architecture and simulate it on your CPU logic before testing the three programs. This will help you identify bugs in your design and fix them. You do not have to submit this program.

Questions

1. What changes did you make in your original ISA and why?
2. What is instruction count for each one of the three programs? How do the numbers compare with those for the ISS? If the numbers are different, why?
3. What are the strengths of your design?
4. What are the deficiencies of your design?
5. Which instruction is most responsible for the length of the critical path?
6. Which instruction is most expensive in terms of the number of gates required?
7. Having gone through a complete CPU design experience, what would you do differently with your ISA to:
 - Decrease static and dynamic instruction count
 - Simplify data path design
 - Simplify CPU design
8. If you were to pipeline the execution, what would the pipeline stages be? Give at least three issues that will complicate the design of your processor.

