University of Rochester

Ece 200 - Computer Organization

Lab 4 Report

Adam Stenson

April 29, 2016

Abstract

In this lab I simulated a single-cycle 16 bit RISC processor in a mostly structural fashion. In total 26 instructions were implemented.

1 Single-Cycle Implementation

The single cycle processor was implemented using 21 modules. Each module, what they do, and how they were implemented (behaviorally or structurally) is listed below.

1. full_adder - Single bit full adder for use in the ALU and the adder units. Implemented structurally.
2. multiply - Multiplies two 16-bit numbers and outputs to high and low registers. Implemented behaviorally.
3. division - Divides two 16-bit numbers and outputs to high and low registers. Implemented behaviorally.
4. sll - Shifts the input number to the left a given number of bits. Implemented behaviorally.
5. srl - Shifts the input number to the left a given number of bits. Implemented behaviorally.
6. clock - Behavior implementation that has a frequency of 100ns and for the purpose of the general testbench only runs for 26 cycles.
7. mux - Single bit 2-to-1 multiplexer used in the ALU. Implemented structurally.
8. adder32bit - Adds two 32-bit numbers together. Used in the datapath that increments the PC. Implemented structurally.
9. ALU - Single bit ALU unit that performs six operations: and, or, nor, slt, add, sub. Implemented structurally.
10. ALU16bit - Uses the 1-bit ALU units to structurally implement a 16-bit ALU for the six operations listed above. Includes a behavioral implementation for the zero line needed in branch operations.
11. dataMem - Reads and write data for store and load commands. Implemented behaviorally.
12. instructMem - Reads 32-bit instructions from file in 8-bit chunks. Implemented behaviorally.
13. regfile - Reads and writes to the 16 16-bit registers. Implemented behaviorally.
14. shiftLeft - Shifts a 32-bit number left two bits, essentially multiplying it by four. Used in the PC incrementation datapath. Implemented structurally.
15. mux16bit - 16-bit 2-to-1 multiplexer implemented behaviorally.
16. mux6to1_16bit - 16-bit 6-to-1 multiplexer implemented behaviorally.

17. mux3to1_1ibt - Single bit 3-to-1 multiplexer implemented structurally.
18. mux3to1 - Uses the mux3ti1_1bit unit to structurally implement a 32-bit 3-to-1 multiplexer used in the PC incrementation datapath.
19. padding16to32 - Adds bit padding to 16-bit numbers to make them 32-bit. Implemented structurally.
20. padding26to32 - Adds bit padding to 26-bit numbers to make them 32-bit. Implemented structurally.
21. control - Connects all the components together in a structural fashion. Assigns the controls lines behaviorally.

2 Additional Instructions

The 8 additional instructions from extra credit part 1 were implemented with the single cycle processor. Unfortunately, the multiply module does not function properly. No matter the input, there is not output from that behavioral module.

3 Test Bench

All 26 instructions were tested in the same test bench code. At the end of every clock cycle the verilog code will output the PC, instruction, and the writeRegData line. The writeRegData line is the 16-bit line from the 6-to-1 mux at the end of the datapath to the register file. For the R-type and some I-type instructions this shows the result of the calculation. For load commands this shows the data being pulled from memory. For store, branch, jump, multiply, and divide commands this lines doesn't represent the actually command result and will not be written to a register.

4 Factorial Code

Assembly Code		Machine Code (Hex)
	lui $t0, 8	3C 08 00 08
	lui $t1, 1	3C 09 00 01
	lui $t2, 1	3C 0A 00 01
fact:	beq $t0, $t2, done	11 0A 00 05
	mult $t1, $t0	04 A8 00 18
	mflo $t1	00 00 48 12
	addi $t0, $t0, -1	42 10 FF FF
	j fact	40 00 00 04