The UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Comp 541 Digital Logic and Computer Design
Fall 2014

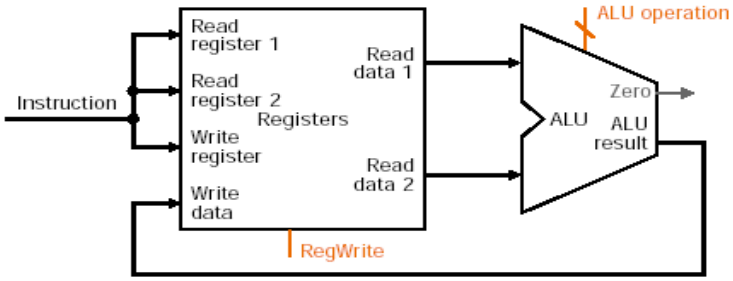
Lab #9: A Basic Datapath
Issued Mon 10/20/14; Due Wed 10/29/14 (11:59pm)

This lab assignment consists of several steps, each building upon the previous. Detailed instructions are provided. Verilog code is provided for almost all of the designs, but some portions of the code have been erased; in those cases, it is your task to complete and test your code carefully. Submission instructions are at the end.

You will learn the following:

- Designing a multi-ported memory (3-port register file)
- Integrating ALU, registers, memory, etc., to form a full datapath
- More practice with test fixtures

Today you begin to implement the MIPS subset that we will use for this class. You will implement the portion of the CPU datapath shown in the following diagram, and test it. However, since we do not yet have a source for instructions, and to aid in testing, we will slightly modify this part of the datapath to provide more controllability and visibility. In particular, we will cut the feedback from the ALU to the write port of the register file, and instead allow *Write data* to be directly supplied by a test fixture. A modified picture is shown on the next page.



Part 1: Register File

First, you will design a 3-port register file. We call it 3-port because three different addresses can be specified at any time: *Read Address 1, Read Address 2,* and *Write Addr.* These are required to access (up to) two source operands and one destination operand required for MIPS instructions.

Do the following:

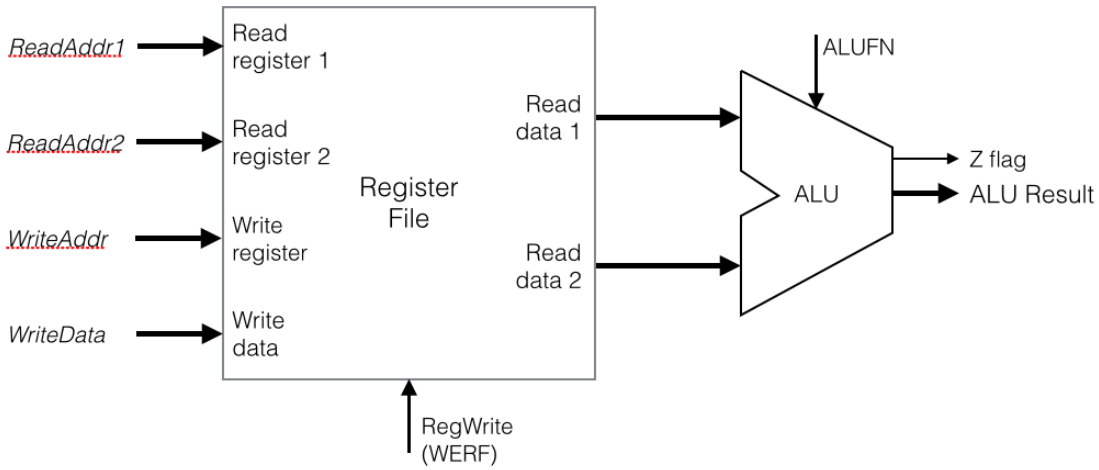
- Start with the **`ram_module`** from Lab 8. Modify it to create a new module called **`register_file`**, which has the following enhancements:
  - three address inputs instead of just one (e.g., *ReadAddr1, ReadAddr2* and *WriteAddr*).
  - two data outputs instead of just one (e.g., *ReadData1* and *ReadData2*).
  - the write enable and clock stay the same.

- when writing, *WriteAddr* is used to determine the memory location to be written.
- when reading register 0, the value read should always be 0 (it does not matter what value is written to it, or whether you write any values into it).
- use parameters for address width, data width, and number of memory locations, just as in Lab 8.

- While in the final CPU design, the three addresses will come from the register fields in the instruction being executed, for now you will need to write a Verilog test fixture to provide these addresses, the data to be written, and the *RegWrite* signal. The test fixture should do a few different reads and writes so you can see via simulation that your register file is working.
- For testing purposes, first try a register file with the default parameters (4 address bits, 4 data bits, and 16 locations). When everything is working okay, increase the parameters to the actual MIPS size (5 address bits, 32 data bits, and 32 locations).
- Note: Without any display, you cannot test this design by implementing it on the board. Simulation would suffice.
- Name your Verilog source `regfile.v`.

Part 3: Putting it together

Design a top-level module that contains the register file and your ALU (from Lab 3). Name the Verilog source `datapath.v`. This module must exactly correspond to the block diagram below.



Note the following:

- To aid in testing your design, send "*Read data 1", "Read data 2"* and *"ALU Result"* to the output of the top-level module so they can be easily observed during simulation. The Zero flag (*Z*) must also be generated as an output from the top-level module (because branch instructions will need it).
- For now, do not feed the ALU result back to the register file. Instead, the data to be written into the register file should come in directly from the test fixture as an input to the top-level module. This will allow you to initialize any register with a value you choose. Otherwise, all registers will be uninitialized and you will not be able to test the functionality of your design.
- The inputs to the top-level module are: clock, *RegWrite*, the three addresses, the ALU operation to be performed (*ALUFN*), and the data to be written into the register file (*WriteData*).
- Create a Verilog test fixture to simulate and test your design.

What to submit:

- **Your Verilog source for the register file (regfile.v) and the top-level datapath (datapath.v).**

How to submit: **Please submit your work by email by 11:59pm, October 29, 2014 (Wed) as follows:**

- **Send email to:** `comp541submit-cs@cs.unc.edu`
- **Use subject line: Lab 9**
- **In the body of the email, mention whether the datapath worked as expected**
- **Include the two attachments as specified above**
