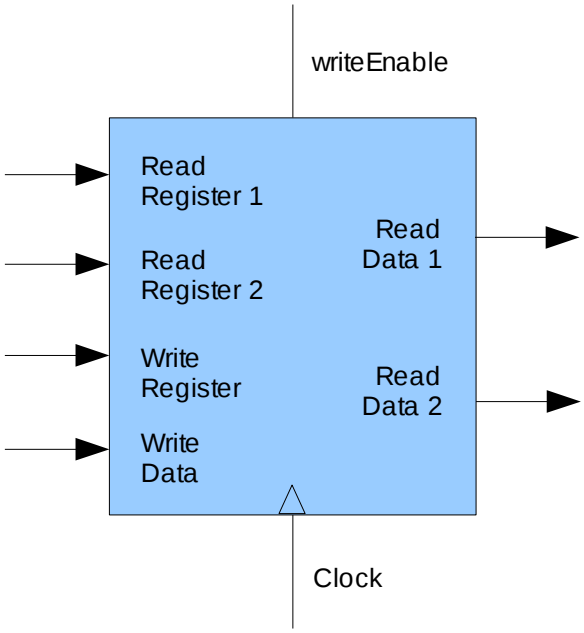
CS/COE 0447 Fall 2009
Lab 11: Register Files and Memory
Due Date: December 3, 2009

To get started on this lab, attend recitation on 11/20. Each of you should submit your own solution, according to these instructions: http://www.cs.pitt.edu/~sab104/teaching/cs447/submission.html. You may collaborate with your partner, but each person must turn in their own copy of the lab, with the name of their partner. The lab is due on 12/03 at 11:59pm.

For this lab, we will use a tool for designing and simulating digital circuits. The tool is called *Logisim* and is available at http://ozark.hendrix.edu/~burch/logisim/.

1) Register Files

Consider a register file with two read ports and one write port. A register file always outputs the contents of the registers corresponding to the Read Register inputs. In addition, the value of the register corresponding to the Write Register input is updated with the value of the Write Data input when the Write Enable signal is activated and there is a rising clock edge.



A register file can be implemented with a multiplexer for each read port, a decoder for the write port and an array of registers built from D flip-flops.

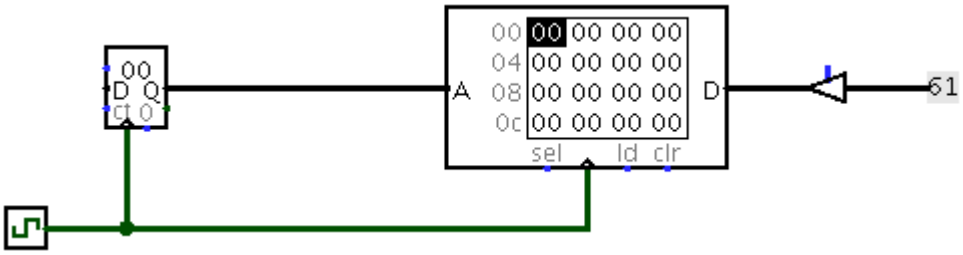
Implement a register file in *Logisim*. The register file should have four 16-bit registers, two read ports and one write port.

2) Memory

Logisim provides a RAM component, which can store up to 2^{24} values, each up to 32 bits wide. To store and load data from RAM, you have to specify an address, which is up to 24 bits wide. The RAM component provides either two separate ports for loads and stores or one single load/store port that is shared by reads and writes. For this problem, we will be using a RAM memory with 256 8-bit values and a single load/store port.

Logisim also provides a counter component, which counts from 0 to a given number each time the clock ticks.

Consider the problem of setting every byte of the RAM to a specified value. We can do this in a circuit by having a counter generate every address of the memory and storing the specified value at each memory location. The component that looks like a triangle is called a Controlled Buffer. Its purpose is to put the value of its input at its output only when the controlling signal is high. When it is not high, the output is held floating, which means that another component can drive the signal.



Build a circuit in *Logisim* that writes the value 0x61 to every memory location. Your circuit should allow the user to reset the counter anytime. In addition, the circuit should stop writing values to memory after it has already written all memory locations.

Hint: build a FSM to control the inputs of the memory, the counter and the controlled buffer based on the user input and the state of the counter.