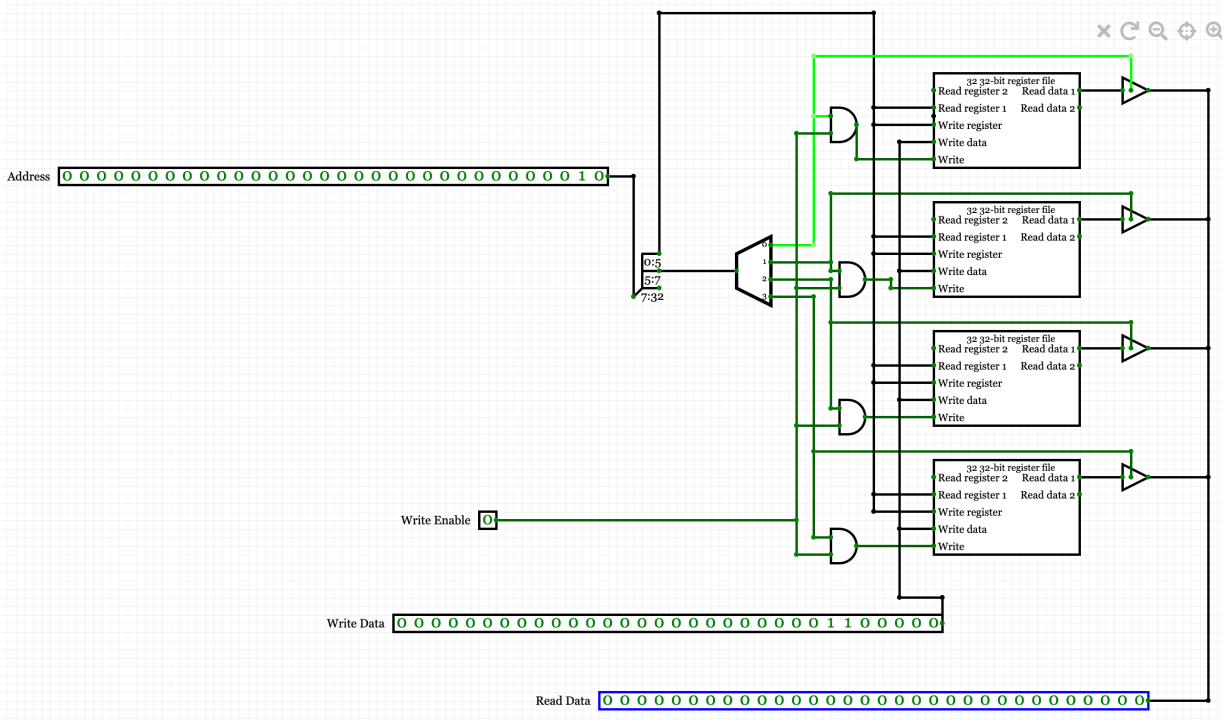
Name: _____

Lab for ITSC 3181, Introduction to Computer Architecture, Summer 2021

Lab #10: Logic design using Digital (https://github.com/hneemann/Digital): Memory (SRAM) created using register files. Due 06/17 and count for 1% of the final accumulated percentage grade.

From Lab 07 to Lab 12, we will build a RISC-V CPU to support an essential set of instructions including at least add, and, or, sub, lw, sw, and beq instructions. While you are encouraged to design a 32-bit CPU (each register stores a 32-bit word, and the ALU operates on 32-bit data), you are ok to design 8-bit or 16-bit CPU (each register stores a 8-/16-bit data, and the ALU operates on 8/16 bits data). Designing 8-bit/16-bit is much easier comparing with designing 32-bit, but with same design principle. But regardless of the data size we use, each instruction is still encoded as a 32-bit word and the address for accessing data from memory are also 32-bit address. In this lab, we will create a simple 256 x 32/16/8-bit memory module using the register file we create in Lab09. It is actually a DRAM.

1. Review lecture slides (145 – 146 of https://passlab.github.io/ITSC3181/notes/AppendixA_LogicDesignBasics.pdf) for the design of a memory module that uses tristate buffer and a decoder to select memory cell to read/write.
2. You will use the register file you designed in Lab09 to create a 256 x 32/16/8-bit memory module (256 rows and 32/16/8-bit width), which would require 8 register files you created before (256 = 8 * 32). Below is the design of a 128 x 32bit memory module that uses 4 register files. Your design should be exactly like this, but have 8 register files for 256 rows. The input and output of the 256 x 32/16/8-bit module should be exactly the same as the following 128 x 32bit memory module. The inputs are 32-bit “Address”, 1-bit “Write Enable”, and 32/16/8-bit “Write Data”. The output is 32/16/8-bit “Read Data”.
3. It is important for you to understand the address of the memory module and how it is used to access a row of the memory. Given a 32-bit address, the least-significant 5 bits (bit 0-4) are used to select a row, which is a specific register within a register file, and next two bits (bit 5-6, and since we have four register files) is used to select a register file from the four files using a 4-1 decoder. The rest of the 32-bit address are ignored. The splitter in the following design shows how bits are split and fed into the right component of the design.



Tasks:

1. (10%) Create a new circuit and save it as “256 x <bitwidth>-bit Memory”. Arrange 8 register files for the 256 x 32/16/8-bit memory module by reusing the 32 32/16/8-bit RegisterFile you have designed before. Leave space between register files for wires.
2. (10%) Add and label the inputs (32-bit “Address”, 1-bit “Write Enable”, and 32/16/8-bit “Write Data”), and the output (32/16/8-bit “Read Data”). Make sure the bitwidth and label are correctly set for them. Organize them in the appropriate position for the overall layout of the circuit. Connect the “Write Data” input to the “Write Data” input of all register files.
3. (10%) Since we have 8 register files, we need an 8-1 decoder to select one of the register files for read/write. Add a decoder in your design and set it to have the correct bitwidth for an 8-1 decoder.
4. (20%) Add a splitter to split the 32-bit address input into three parts, the least 5 bits go to “Read register 1” and “Write Register” of all registers, the next 3 bits go to the input of the decoder and ignore the rest.
5. (20%) Add an AND gate for each of the register files, and connect the two inputs of the AND gate to the corresponding decoder output and the “Write Enable” input. Connect the output of the AND gate to the “Write” input of the register file.
6. (20%) Add a trigate buffer (Wire→Driver) for each of the register file; and connect “Read data 1” output of the register file to the input of the trigate, connect the corresponding output of the decoder to the “Enable” input of the trigate, and connect the output of the trigate to the “Read Data” output of the memory module.
7. (10%) Test the “256 x 32/16/8-bit Memory” you designed: Simulate and test your memory design by setting address, write signal, read data and write data to make sure it works properly.

Important: when designing the circuit, please make sure you follow these rules for adding, changing components, input/output and wires. These rules are applied to all the design lab tasks of the course.

1) **Each input and output of a design MUST be properly and meaningfully labeled.**
2) **Each component, input and output should be correctly configured in terms of its bitwidth and signal control width.**

3) **Keep wires and components organized and layed out according to the circuit schematics rules, a) Inputs on the left (or top), b) Outputs on right (or bottom), c) gates flow from left to right, d) Straight (not angled) wires are best, e) Wires always connect at a T junction (only 90-degree turn or connection), f) A dot where wires cross indicates a connection between the wires, and g) Wires crossing *without* a dot make no connection. While you may not need to follow all those rules for creating correct and small circuit, it is very important when for creating complex circuit. So, make sure you properly organize the components and wires, make them structured and look good. That will help reduce errors.**
4) **For drawing straight wires using mouse, make one turn per each draw. You should not use one draw to make a connection that needs to have two or more 90-degree turns, which would create angled wires. For those wires, you have to make a wire that has a 90-degree turn, and then connect it with another wire that also make 90 turn, and so on. Try to minimize the turns as much as possible. If two wires have to be crossed, but should not be connected, make sure no dot is marked on where the wire is crossed.**

Submission: take screen shoot of your design of each task and save it to a single file of PDF format. Then submit the file on Canvas.

Your grade will be based on both the correctness of your design and the organization of the design, 60% and 40% respectively.

Task	1 (10%)	2 (10%)	3 (10%)	4 (20%)	5 (20%)	6 (20%)	7 (10%)	Total
Correctness (components, input/output, bitwidth, connection, width of control, etc), 60%								
Organization (I/O labeled and positioned correctly, all straight wires and T junction, dot used correctly, readability etc), 40%								
Total								