CSE4372/5392 (Spring 2025)
Lab #4

This lab is due by March 6, with a 20% penalty per week day for being late.

In this lab, you will start using the dual-ported memory that will be accessed by the IF and MEM stages of the pipeline.

The following steps will guide you through this process:

1. Create a module for a 32k x 32b true dual port RAM with byte write enable and save in a separate file.

```
module dual_port_ram
    (
    // Clock
    input clk,

    // Instruction port (RO)
    input [31:2] i_addr,
    output reg [31:0] i_rdata,

    // Data port (RW)
    input [31:2] d_addr,
    output reg [31:0] d_rdata,
    input d_we,
    input [3:0] d_be,
    input [31:0] d_wdata);
…
```

2. Create a top level module that instantiates the dual port RAM module, which provides a port for fetching the instruction and another port for reading and writing the data values.

3. Use the 100 MHz clock (CLK100) as your system clock (we can use a slower clock later to make it easier to close timing).

4. Modify the design to read an initialization file (this will contain the program words and constant values for your design at a later time).

5. Drive the i_addr port with a series of addresses (one per clock) and verify that i_rdata values are the same as those in the hex file. Use ILA to verify the values are correct on the next clock after the address is provided.

6. Drive the d_addr port as in step 6 and verify that d_rdata is correct. Use ILA to verify the operation is correct.

7. Drive the d_wdata with a test value d_be = 1111 for one clock to cause the memory to be overwritten. In a subsequent clock, verify that the d_rdata at this address reflects the written value.

8. Add a test case counter that drives a test block.

9. Use a test block to generate a series of five test values (one per clock for each test counter value) as excitation for your design:
- 32-bit address for the r/w operation (alu_out)
- 32-bit data value for write operations (rs2)
(D31-0 for words, D15-0 for half words, or D7-0 for bytes)
- 2-bit width signal (width)
(0=byte, 1=half word, 2 = word)
(note this will later be the 2 LSb of funct3)
- 1-bit signed/unsigned flag (unsigned)
(1=unsigned, 0=signed)
(note this will later be the MSb of funct3)
- 1-bit write enable signal (we)
(1=write, 0=read)

10. Build a shifter to shift the data value from test block (rs2) to the memory write data input (d_wdata) based on the address (alu_out). This makes sure that the data bits are on the correct data bus lines.

11. Build a function to generate the byte enable signals (w_be) for the memory based on the address (pc) and width (width). This makes sure that only the selected bytes are used.

12. Connect the memory so that d_addr, d_wdata, d_be, and d_we are driven from the test block.

13. Build a function to convert the output of the memory (d_rdata) to a 32-bit value that will be written to a register in future labs. This block will provide shifting and optionally zero stuffing or sign extension based on the address, width, and unsigned control signals.

14. Add signal taps to the excitation signals (pc, rs2, width, unsigned, and we), the memory data write input (d_wdata), memory byte enable (d_be), and memory data read output (d_rdata).

15. Verify that the operation is correct for all 5 load and all 3 store instructions.

16. Demonstrate the register read and write operations to the TA and send your Verilog source .v or .sv files (don't send the entire project) to the TA for credit.