```
// Eugene Ngo
// 4/20/2023
// CSE 469
// Lab 2, Task 1 and 2

/* top is a structurally made toplevel module. It consists of 3 instantiations, as well as
the signals that link them.
** It is almost totally self-contained, with no outputs and two system inputs: clk and rst.
clk represents the clock
** the system runs on, with one instruction being read and executed every cycle. rst is the
system reset and should
** be run for at least a cycle when simulating the system.
*/

// clk - system clock
// rst - system reset. Technically unnecessary
module top(
    input logic clk, rst
);

    // processor io signals
    logic [31:0] Instr;
    logic [31:0] ReadData;
    logic [31:0] WriteData;
    logic [31:0] PC, ALUResult;
    logic        MemWrite;

    // our single cycle arm processor
    arm processor (
        .clk        (clk        ),
        .rst        (rst        ),
        .Instr      (Instr      ),
        .ReadData   (ReadData   ),
        .WriteData  (WriteData  ),
        .PC         (PC         ),
        .ALUResult  (ALUResult  ),
        .MemWrite   (MemWrite   )
    );

    // instruction memory
    // contained machine code instructions which instruct processor on which operations to
make
    // effectively a rom because our processor cannot write to it
    imem imemory (
        .addr   (PC     ),
        .instr  (Instr  )
    );

    // data memory
    // containes data accessible by the processor through ldr and str commands
    dmem dmemory (
        .clk     (clk       ),
        .wr_en   (MemWrite  ),
        .addr    (ALUResult ),
        .wr_data (WriteData ),
        .rd_data (ReadData  )
    );


endmodule


// testbench tests the behaviors of the ALU by running through an instance of the
// top module. The top module instantiates the imeme module which calls on
// the memfile.dat and memfile2.dat files that send in the instruction inputs
// for the testbench module to use. The results are compared to actual expected
// values to determine if the functionality of the overall CPU is correct.
module testbench();

    // system signals
    logic clk, rst;

    // generate clock with 100ps clk period
    initial begin
        clk = '1;
        forever #50 clk = ~clk;
    end

    // processor instantion. Within is the processor as well as imem and dmem
    top cpu (.clk(clk), .rst(rst));

     initial begin
        // start with a basic reset
        rst = 1; @(posedge clk);
        rst <= 0; @(posedge clk);

        // repeat for 50 cycles. Not all 50 are necessary, however a loop at the end of the program will keep anything weird from happening
        repeat(50) @(posedge clk);

        // basic checking to ensure the right final answer is achieved. These DO NOT prove your system works. A more careful look at your
        // simulation and code will be made.

        // task 1:
//        assert(cpu.processor.u_reg_file.memory[8] == 32'd11) $display("Task 1 Passed");
//        else                                                 $display("Task 1 Failed");

        // task 2:
        assert(cpu.processor.u_reg_file.memory[8] == 32'd1)  $display("Task 2 Passed");
        else                                                 $display("Task 2 Failed");

        $stop;
    end

endmodule
```