```
1   // Eugene Ngo
2   // 5/3/23
3   // EE 469
4   // Lab 3
5
6   /* top is a structurally made toplevel module. It consists of 3 instantiations, as well as
    the signals that link them.
7   ** It is almost totally self-contained, with no outputs and two system inputs: clk and rst.
    clk represents the clock
8   ** the system runs on, with one instruction being read and executed every cycle. rst is the
    system reset and should
9   ** be run for at least a cycle when simulating the system.
10  */
11
12  // clk - system clock
13  // rst - system reset. Technically unnecessary
14  `timescale 1ns/10ps
15  module top(
16      input logic clk, rst
17  );
18
19      // processor io signals
20      logic [31:0] Instr;
21      logic [31:0] ReadData;
22      logic [31:0] WriteData;
23      logic [31:0] PC, ALUResult;
24      logic        MemWrite;
25
26      // our single cycle arm processor
27      arm processor (
28          .clk        (clk        ),
29          .rst        (rst        ),
30          .InstrF      (Instr      ),
31          .ReadData   (ReadData   ),
32          .WriteDataM  (WriteData  ),
33          .PC         (PC         ),
34          .ALUResultM  (ALUResult  ),
35          .MemWrite   (MemWrite   )
36      );
37
38      // instruction memory
39      // contained machine code instructions which instruct processor on which operations to
    make
40      // effectively a rom because our processor cannot write to it
41      imem imemory (
42          .addr   (PC     ),
43          .instr  (Instr  )
44      );
45
46      // data memory
47      // containes data accessible by the processor through ldr and str commands
48      dmem dmemory (
49          .clk     (clk       ),
50          .wr_en   (MemWrite  ),
51          .addr    (ALUResult ),
52          .wr_data (WriteData ),
53          .rd_data (ReadData  )
54      );
55
56
57  endmodule
58
59  /* testbench is a simulation module which simply instantiates the processor system and runs
    50 cycles
60  ** of instructions before terminating. At termination, specific register file values are
    checked to
61  ** verify the processors' ability to execute the implemented instructions.
62  */
63  module testbench();
64
65      // system signals
66      logic clk, rst;
67
```

```
68        // generate clock with 100ps clk period
69        initial begin
70            clk = '1;
71            forever #50 clk = ~clk;
72        end
73
74        // processor instantion. Within is the processor as well as imem and dmem
75        top cpu (.clk(clk), .rst(rst));
76
77        initial begin
78            // start with a basic reset
79            rst = 1; @(posedge clk);
80            rst <= 0; @(posedge clk);
81
82            // repeat for 50 cycles. Not all 50 are necessary, however a loop at the end of the
      program will keep anything weird from happening
83            repeat(50) @(posedge clk);
84
85            // basic checking to ensure the right final answer is achieved. These DO NOT prove
      your system works. A more careful look at your
86            // simulation and code will be made.
87
88            // task 1:
89            assert(cpu.processor.u_reg_file.memory[8] == 32'd11) $display("Task 1 Passed");
90            else                                                 $display("Task 1 Failed");
91
92            // task 2:
93    //        assert(cpu.processor.u_reg_file.memory[8] == 32'd1)  $display("Task 2 Passed");
94    //        else                                                 $display("Task 2 Failed");
95
96            $stop;
97        end
98
99    endmodule
100
```