```
1   // Eugene Ngo
2   // 4/7/2023
3   // CSE 469
4   // Lab 1 Task 2
5
6   // This is the top level module
7   // This register file is 16x32, has two read ports, one write port, and is asynchronous
8   // as specified
9   module reg_file (input logic clk, wr_en, input logic [31:0] write_data,
10                   input logic [3:0] write_addr,
11                   input logic [3:0] read_addr1, read_addr2,
12                   output logic [31:0] read_data1, read_data2);
13
14      logic [15:0][31:0] memory;
15
16      always_ff @ (posedge clk) begin
17         if (wr_en) begin
18            memory[write_addr] <= write_data;
19         end
20      end
21
22      always_comb begin
23         read_data1 = memory[read_addr1];
24         read_data2 = memory[read_addr2];
25      end
26
27  endmodule
28
29  // reg_file_testbench tests all expected, unexpected, and edgecase behaviors
30  module reg_file_testbench();
31      logic clk, wr_en;
32      logic [31:0] write_data, read_data1, read_data2;
33      logic [3:0] write_addr, read_addr1, read_addr2;
34
35      reg_file dut (.clk, .wr_en, .write_data, .write_addr, .read_addr1, .read_addr2, .
    read_data1, .read_data2);
36
37      parameter clock_period = 10000;
38
39      integer i;
40
41      initial begin // Set up the clock
42         clk <= 0;
43         for (i=0; i<1000; i++) begin: clockCount
44            forever #(clock_period /2) clk <= ~clk;
45         end
46
47      end
48
49      initial begin
50         $display("%t Behavior check", $time);
51
52               // Testing functionality of
53               // 1 cycle delay of writes.
54         wr_en = 1'b1;                  @(posedge clk);
55         write_data = 32'b0;            @(posedge clk);
56         write_addr = 4'b0010;          @(posedge clk);
57                                        @(posedge clk);
58
59         write_data = 32'b1;            @(posedge clk);
60         write_addr = 4'b0011;          @(posedge clk);
61                                        @(posedge clk);
62               // Testing functionality of
63               // same cycle reads.
64         read_addr1 = 4'b0010;          @(posedge clk);
65         read_addr2 = 4'b0011;          @(posedge clk);
66
67               // Testing the functionality of
68               // 1 cycle delayed reads
69               // after an updated write
70         write_addr = 4'b0010;          @(posedge clk);
71         read_addr1 = 4'b0010;          @(posedge clk);
72                                        @(posedge clk);
```

```
73          // read_data1 should update to 1 now.
74
75          write_data = 32'b0;             @(posedge clk);
76          write_addr = 4'b0011;           @(posedge clk);
77
78          read_addr2 = 4'b0011;           @(posedge clk);
79                                          @(posedge clk);
80          // read_data2 should update to 0 now.
81
82
83
84
85      end
86
87
88
89
90  endmodule
91
```