```
// Eugene Ngo
// 5/3/23
// EE 469
// Lab 3

// reg file is a 16x32 register file that has 1
// write port and 2 asynchronous read ports.
// When wr_en is true, we write the write data into the
// write address.
module reg_file(
   input  logic        clk, wr_en,
   input  logic [31:0] write_data,
   input  logic [3:0]  write_addr,
   input  logic [3:0]  read_addr1, read_addr2,
   output logic [31:0] read_data1, read_data2);

  logic [15:0][31:0] memory;

  always_ff @(posedge clk) begin
    if (wr_en) begin
      memory[write_addr] <= write_data;
    end


  end
  assign read_data1 = memory[read_addr1];
  assign read_data2 = memory[read_addr2];
endmodule


// reg_file_testbench is a testing fiel for reg_file
// that tests  that write data is written
// into register file a cycle after wr_en is true,
// checks if read data updates the register data
// the same cycle as the address asserted,
// and checks if read data is updated to write data
// the cycle after address is provided if
// write address is the same and wr_en is true
module reg_file_testbench();
   logic CLOCK_50;
   logic clk;
   logic wr_en;
   logic [31:0] write_data;
   logic [3:0]  write_addr;
   logic [3:0]  read_addr1, read_addr2;
   logic [31:0] read_data1, read_data2;
   assign CLOCK_50 = clk;

   reg_file dut (.clk, .wr_en, .write_data, .write_addr, .read_addr1,. read_addr2, .
read_data1, .read_data2);
   parameter CLOCK_PERIOD=100;
   initial begin
      clk <= 0;
      forever #(CLOCK_PERIOD/2) clk <= ~clk;
   end


   initial begin
      wr_en <= 0; write_data = 5; write_addr = 3; read_addr1 = 2; read_addr2 = 3; @(posedge
clk);
      wr_en <= 1;  repeat(1) @(posedge clk);
      write_data = 10; write_addr = 4;@(posedge clk);
      write_data = 11; write_addr = 5;@(posedge clk);
      read_addr1 = 4; @(posedge clk);
      read_addr2 = 5; @(posedge clk);
      write_data = 12; write_addr = 4; @(posedge clk);
      write_data = 13; write_addr = 5; @(posedge clk);

   end
endmodule
```