```
1   /*
2      Ben Davis
3      1/22/24
4      EE 371
5      Lab 2, Task 3
6
7      This module implements a 16 word x 8 bit 2 port RAM. It has a read
8      and a write enable.
9
10  */
11
12  module RAM_16x8 (
13                      input logic clk, wen,
14                      input logic [7:0] data_in,
15                      input logic [3:0] read_addr,
16                      input logic [3:0] write_addr,
17                      output logic [7:0] data_out
18                      );
19     //ram storage
20     logic [7:0] memory_array [3:0];
21
22     //allows for writing into the ram, and also
23     //makes sure it reads at the correct address
24     always_ff @(posedge clk) begin
25        if(wen) begin
26           memory_array[write_addr] <= data_in;
27        end
28        data_out <= memory_array[read_addr];
29     end
30
31  endmodule
32  //testbench
33  module RAM_16x8_testbench();
34     //reset logic variables
35     logic clk, wen;
36     logic [7:0] data_in;
37     logic [3:0] read_addr;
38     logic [3:0] write_addr;
39     logic [7:0] data_out;
40     //instatiate ram module
41     RAM_16x8 dut (.clk, .wen, .data_in, .read_addr, .write_addr, .data_out);
42
43     // clock setup
44     parameter clock_period = 100;
45
46     initial begin
47        clk <= 0;
48        forever #(clock_period /2) clk = ~clk;
49     end // of clock setup
50
51     //tests an instance of writing two pieces of data, reading
52     //the first, then writing another in a third address, and
53     //then reading the second data address
54
55     initial begin
56
57        wen <= 1; data_in <= 8'b01110000; write_addr <= 4'b0;
58                                          read_addr <= 4'b0; @(posedge clk);
59        wen <= 1; data_in <= 8'b00000001; write_addr <= 4'b1;
60                                          read_addr <= 4'b0; @(posedge clk);
61        wen <= 0; data_in <= 8'b01110000; write_addr <= 4'b01111;
62                                          read_addr <= 4'b0; @(posedge clk);
63        wen <= 0; data_in <= 8'b01110000; write_addr <= 4'b01111;
64                                          read_addr <= 4'b1; @(posedge clk);
65        wen <= 1; data_in <= 8'b01110000; write_addr <= 4'b01111;
66                                          read_addr <= 4'b1; @(posedge clk);
67        wen <= 1; data_in <= 8'b01110000; write_addr <= 4'b01111;
68                                          read_addr <= 4'b1; @(posedge clk);
69
70        $stop;
71     end
72  endmodule
```