

```
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
20 //ram storage
21 logic [7:0] memory_array [3:0];
22
23 //allows for writing into the ram, and also
24 //makes sure it reads at the correct address
25 always_ff @(posedge clk) begin
26     if(wen) begin
27         memory_array[write_addr] <= data_in;
28     end
29     data_out <= memory_array[read_addr];
30 end
31
32 endmodule
33
34 //testbench
35 module RAM_16x8_testbench();
36     //reset logic variables
37     logic clk, wen;
38     logic [7:0] data_in;
39     logic [3:0] read_addr;
40     logic [3:0] write_addr;
41     logic [7:0] data_out;
42     //instantiate ram module
43     RAM_16x8 dut (.clk, .wen, .data_in, .read_addr, .write_addr, .data_out);
44
45     // clock setup
46     parameter clock_period = 100;
47
48     initial begin
49         clk <= 0;
50         forever #(clock_period / 2) clk = ~clk;
51     end // of clock setup
52
53     //tests an instance of writing two pieces of data, reading
54     //the first, then writing another in a third address, and
55     //then reading the second data address
56
57     initial begin
58         wen <= 1; data_in <= 8'b01110000; write_addr <= 4'b0;
59         read_addr <= 4'b0; @(posedge clk);
60         wen <= 1; data_in <= 8'b00000001; write_addr <= 4'b1;
61         read_addr <= 4'b0; @(posedge clk);
62         wen <= 0; data_in <= 8'b01110000; write_addr <= 4'b0111;
63         read_addr <= 4'b0; @(posedge clk);
64         wen <= 0; data_in <= 8'b01110000; write_addr <= 4'b0111;
65         read_addr <= 4'b1; @(posedge clk);
66         wen <= 1; data_in <= 8'b01110000; write_addr <= 4'b0111;
67         read_addr <= 4'b1; @(posedge clk);
68         wen <= 1; data_in <= 8'b01110000; write_addr <= 4'b0111;
69         read_addr <= 4'b1; @(posedge clk);
70
71         $stop;
72     end
73 endmodule
```