```
/*
   Ben Davis
   1/22/24
   EE 371
   Lab 2, Task 3

   This module implements a 16 word x 8 bit 2 port RAM. It has a read
   and a write enable.

*/

module RAM_16x8 (
                  input logic clk, wen,
                  input logic [7:0] data_in,
                  input logic [3:0] read_addr,
                  input logic [3:0] write_addr,
                  output logic [7:0] data_out
                  );
   //ram storage
   logic [7:0] memory_array [3:0];

   //allows for writing into the ram, and also
   //makes sure it reads at the correct address
   always_ff @(posedge clk) begin
      if(wen) begin
         memory_array[write_addr] <= data_in;
      end
      data_out <= memory_array[read_addr];
   end

endmodule
//testbench
module RAM_16x8_testbench();
   //reset logic variables
   logic clk, wen;
   logic [7:0] data_in;
   logic [3:0] read_addr;
   logic [3:0] write_addr;
   logic [7:0] data_out;
   //instatiate ram module
   RAM_16x8 dut (.clk, .wen, .data_in, .read_addr, .write_addr, .data_out);

   // clock setup
   parameter clock_period = 100;

   initial begin
      clk <= 0;
      forever #(clock_period /2) clk = ~clk;
   end // of clock setup

   //tests an instance of writing two pieces of data, reading
   //the first, then writing another in a third address, and
   //then reading the second data address

   initial begin

      wen <= 1; data_in <= 8'b01110000; write_addr <= 4'b0;
                                    read_addr <= 4'b0; @(posedge clk);
      wen <= 1; data_in <= 8'b00000001; write_addr <= 4'b1;
                                    read_addr <= 4'b0; @(posedge clk);
      wen <= 0; data_in <= 8'b01110000; write_addr <= 4'b01111;
                                    read_addr <= 4'b0; @(posedge clk);
      wen <= 0; data_in <= 8'b01110000; write_addr <= 4'b01111;
                                    read_addr <= 4'b1; @(posedge clk);
      wen <= 1; data_in <= 8'b01110000; write_addr <= 4'b01111;
                                    read_addr <= 4'b1; @(posedge clk);
      wen <= 1; data_in <= 8'b01110000; write_addr <= 4'b01111;
                                    read_addr <= 4'b1; @(posedge clk);

      $stop;
   end
endmodule
```