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

   This module implements a 32x4 ram and uses the hexadecimal
   module. It takes in CLOCK_50, KEY0 as a reset, SW9 as a write
   enable, SW3-0 as the input data, and SW8-4 as the data address.
   It also outputs the data output on hex 0, the input data on
   hex 2, and the data address on hex 4 and 5. All hex data is
   displayed in hexadecimal format.

*/

module DE1_SoC_RAM_32x4 (SW, KEY, HEX0, HEX2, HEX4, HEX5);
   //setup logic variables for FPGA
   input logic [9:0] SW;
   input logic [3:0] KEY;
   output logic [6:0] HEX0;
   output logic [6:0] HEX2;
   output logic [6:0] HEX4;
   output logic [6:0] HEX5;
   //intermediate logic
   logic [3:0] out;
   //instantiate the ram
   RAM_32x4 task1_ram (.clk(KEY[0]), .wen(SW[9]), .data_in(SW[3:0]),
                       .data_addr(SW[8:4]), .data_out(out));
   //conversion of binary data to hexadecimal format
   //on 7 segment active low hex displays
   //also use of more intermediate data
   logic [6:0] z_out;
   hexadecimal zero (.in(out), .out(z_out));

   logic [6:0] two_out;
   hexadecimal two (.in(SW[3:0]), .out(two_out));

   logic [6:0] four_out;
   hexadecimal four (.in(SW[7:4]), .out(four_out));

   logic [6:0] five_out;
   hexadecimal five (.in(SW[8]), .out(five_out));
   //assignment of hex displays
   assign HEX0 = z_out;
   assign HEX2 = two_out;
   assign HEX4 = four_out;
   assign HEX5 = five_out;

endmodule
//testbench
module DE1_SoC_RAM_32x4_testbench();
   //reset logic variables
   logic [9:0] SW;
   logic [3:0] KEY;
   logic [6:0] HEX0;
   logic [6:0] HEX2;
   logic [6:0] HEX4;
   logic [6:0] HEX5;
   //instantiate a case of this module
   DE1_SoC_RAM_32x4 dut (.SW, .KEY, .HEX0, .HEX2, .HEX4, .HEX5);

   // clock setup with KEY 0
   parameter clock_period = 100;

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

   //this instance tests two pieces of data being written into the
   //ram and then being read immediately afterwards
   initial begin
      SW[8:4] <= 5'b01000; SW[3:0] <= 4'b0101; SW[9] <= 1; @(posedge KEY[0]);
```

```
74          SW[8:4] <= 5'b01000; SW[3:0] <= 4'b0101; SW[9] <= 0; @(posedge KEY[0]);
75          SW[8:4] <= 5'b01000; SW[3:0] <= 4'b0101; SW[9] <= 0; @(posedge KEY[0]);
76          SW[8:4] <= 5'b00010; SW[3:0] <= 4'b0011; SW[9] <= 1; @(posedge KEY[0]);
77          SW[8:4] <= 5'b00010; SW[3:0] <= 4'b0011; SW[9] <= 0; @(posedge KEY[0]);
78          SW[8:4] <= 5'b00010; SW[3:0] <= 4'b0011; SW[9] <= 0; @(posedge KEY[0]);
79          SW[8:4] <= 5'b01000; SW[3:0] <= 4'b0101; SW[9] <= 0; @(posedge KEY[0]);
80          SW[8:4] <= 5'b01000; SW[3:0] <= 4'b0101; SW[9] <= 0; @(posedge KEY[0]);
81          $stop;
82      end
83  endmodule
```