```
// An Nguyen, Darren Do
// 3/8/2024
// EE 371
// Lab #6, Task #2

// HEX_RAM module takes a 4-bit input 'in' and produces a 7-bit output 'out'.
// The primary purpose of this module is to convert the 4-bit input into its
// hexadecimal equivalent and represent it in a 7-segment display format.
module HEX_RAM(in, out);
   input logic [3:0] in;
   output logic [6:0] out;


   // always_comb assigns the HEXADECIMAL 4 bit inputs to a 7-segment display
   // format output
   always_comb
      case(in)
      4'h0: out = 7'b1000000;
      4'h1: out = 7'b1111001;
      4'h2: out = 7'b0100100;
      4'h3: out = 7'b0110000;
      4'h4: out = 7'b0011001;
      4'h5: out = 7'b0010010;
      4'h6: out = 7'b0000010;
      4'h7: out = 7'b1111000;
      4'h8: out = 7'b0000000;
      4'h9: out = 7'b0011000;
      4'hA: out = 7'b0001000;
      4'hB: out = 7'b0000011;
      4'hC: out = 7'b1000110;
      4'hD: out = 7'b0100001;
      4'hE: out = 7'b0000110;
      4'hF: out = 7'b0001110;
      default: out = 7'bx;
      endcase

endmodule

// HEX_RAM_testbench examines the functionality of the HEX_RAM module.
// As the value of the 4-bit inputs increases, the 7 segment HEX display
// should also change as well. Likewise, when the 4-bit inputs decreases,
// the HEX display should also reflect that change.
module HEX_RAM_testbench();
   logic clk;
   logic [3:0] in;
   logic [6:0] out;

   HEX_RAM dut (.in, .out);
   parameter clock_period = 100;
   initial begin
      clk <= 0;
      forever #(clock_period /2) clk <= ~clk;

   end

   initial begin
      in <= 4'b0000;
      for (int i = 0; i < 16; i++) begin
         in += 4'b0001; @(posedge clk);
      end
      $stop;
   end
endmodule

```