```
1   `timescale 1ns / 1ns // `timescale time_unit/time_precision
2
3   /* top-level entity for 32 x 4 embedded memory */
4   module embedded_Memory (input [9:0] SW, input [0:0] KEY, output [6:0] HEX0, HEX2, HEX4,
5                           HEX5, output [9:0] LEDR);
6
7               assign LEDR[9:0] = 10'd0; // all LEDs on the board should be off
8
9               /*
10               * KEY[0] is the clock
11               * SW[3:0] is the dataIn
12               * HEX2 displays dataIn
13               * SW[8:4] is the address
14               * HEX4 and HEX5 display address
15               * HEX0 displays dataOut
16               * SW[9] is writeEnable
17              */
18
19              wire clock, writeEn;
20              wire [3:0] dataIn, dataOut;
21              wire [4:0] address;
22
23              assign clock = KEY[0];
24              assign writeEn = SW[9];
25              assign dataIn = SW[3:0];
26              assign address = SW[8:4];
27
28              hex_decoder D1 (.hex_digit(dataIn), .segments(HEX2));
29              hex_decoder D2 (.hex_digit(dataOut), .segments(HEX0));
30              hex_decoder D3 (.hex_digit(address[3:0]), .segments(HEX4));
31              hex_decoder D4 (.hex_digit({3'd0, address[4]}), .segments(HEX5));
32
33              ram32x4 M1 (.address(address), .clock(clock), .data(dataIn), .wren(writeEn), .q(
    dataOut));
34
35  endmodule // embedded_Memory
36
37  /* bcd to hex for seven-segment display */
38  module hex_decoder (input [3:0] hex_digit, output reg [6:0] segments);
39
40              always @(*)
41              begin
42
43                 case (hex_digit)
44                    4'h0: segments = 7'b100_0000;
45                    4'h1: segments = 7'b111_1001;
46                    4'h2: segments = 7'b010_0100;
47                    4'h3: segments = 7'b011_0000;
48                    4'h4: segments = 7'b001_1001;
49                    4'h5: segments = 7'b001_0010;
50                    4'h6: segments = 7'b000_0010;
51                    4'h7: segments = 7'b111_1000;
52                    4'h8: segments = 7'b000_0000;
53                    4'h9: segments = 7'b001_1000;
54                    4'hA: segments = 7'b000_1000;
55                    4'hB: segments = 7'b000_0011;
56                    4'hC: segments = 7'b100_0110;
57                    4'hD: segments = 7'b010_0001;
58                    4'hE: segments = 7'b000_0110;
59                    4'hF: segments = 7'b000_1110;
60                    default: segments = 7'h7f;
61                 endcase
62
63              end
64
65  endmodule // hex_decoder
```