```
`timescale 1ns / 1ps

/****************************** C E C S  4 4 0 ******************************
 *
 * File Name:  regfile32_TB.v
 * Project:    Lab_Assignment_2
 * Designer:   Rosswell Tiongco
 * Email:      rosswelltiongco@gmail.com
 * Rev. No.:   Version 1.0
 * Rev. Date:  Current Rev. Date
 *
 * Purpose: To verify the validity of the register file, three requirements
 *          are to be met
 *          1) Display initial contents of all 32 registers $readmemh
 *          2) Write new values to reg file based on customer's pattern
 *          3) Display updated contents of all 32 registers
 *
 * Notes:
 *
 ****************************************************************************/

module regfile32_TB;

   // Inputs
   reg         clk,    reset,  D_En;
   reg  [ 4:0] S_Addr, D_Addr, T_Addr;
   reg  [31:0] D;

   // Outputs
   wire [31:0] S, T;

   // Instantiate the Unit Under Test (UUT)
   regfile32 uut (
      .clk(clk),
      .reset(reset),
      .S_Addr(S_Addr),
      .D(D),
      .D_En(D_En),
      .D_Addr(D_Addr),
      .T_Addr(T_Addr),
      .S(S),
      .T(T)
   );

   // Variable declaration
   integer i = 0;
   reg [31:0] memory [31:0];
   always #5 clk = ~clk;

   initial begin
      $display("\n\nCECS440 Register File Testbench");
      clk = 0;
      reset = 0;
      S_Addr = 0;
      D = 0;
      D_En = 0;
      D_Addr = 0;
      T_Addr = 0;

      // Initializing registers
      $display("\n\nINITIALIZING REGISTERS");
      $readmemh("IntReg_Lab2.dat",memory);
      D_En = 1;
      for (i = 0; i < 32; i = i + 1) begin
         D = memory[D_Addr];
         @ (posedge clk) D_Addr = D_Addr + 1;
      end
      D_En = 0;

      //One time reset assertion
      @ (negedge clk) reset = 1'b1;
      @ (negedge clk) reset = 1'b0;

      // Formatting time
      // $timeformat [ ( units_number , precision_number ,
      //                 suffix_string , minimum_field_width ) ] ;
      $timeformat(-9,3,"ns",5);

      // 1) Display initial contents of all 32 registers $readmemh
      reg_dump();
      // 2) Write new values to reg file based on customer's pattern
      reg_write();
      // 3) Display updated contents of all 32 registers
      reg_dump();

   end

// The register dump task displays the address and its contents
task reg_dump(); begin
   #1 $display("\n\nREADING REGISTERS");
    for (i = 0; i < 16; i = i + 1)begin
        // Assigning ALL INPUTS on the negative edge of clk
        @ (negedge clk) begin
            reset = 0;
            D_En = 0;
            D = 0;
            D_Addr = 0;
            S_Addr = i;
            T_Addr = S_Addr + 16;
        end
        // Reading values on the positive edge of clk
        @ (posedge clk)
            #1 $display("Time %t || S_Addr: %h  S:%h || T_Addr:%h  T:%h",
                        $time, S_Addr, S, T_Addr, T);
    end
end
endtask

// The register write task writes the customer's required pattern
task reg_write(); begin
   #1 $display("\n\nWRITING TO REGISTERS");
    // Writing pattern to the register file
    // Start at 1, since register 0 is reserved and not written to
    for (i = 1; i < 32; i = i + 1) begin
```

```verilog
    // Assigning ALL INPUTS on the negative edge of clk
    @ (negedge clk)
        D_Addr = i;
        D = ((~i) << 8 ) + (-65536 * i ) + i;
        S_Addr = 0;  T_Addr = 0;
        D_En = 1; ; reset = 0;
    end
end
endtask


endmodule
```