```
`timescale 1ns / 100ps
/*************************************************************************
* Date: Aug. 16, 2006
* File: Test_Reg1.v (440 Examples)
*
* Testbench to generate some stimulus and display the results for the
* Reg1 module -- a 32 bit register with asynch reset.
*************************************************************************/
//*******************************************************
module Test_Reg1;
//*******************************************************
   wire [31:0] OutReg;
   reg  [31:0] Din;
   reg         clk, reset_;

   // Instantiate Reg32 (named DUT {device under test})
   Reg32 DUT(OutReg, Din, clk, reset_);

   initial begin
      $timeformat(-9, 1, " ns", 6);
      clk = 1'b0;
      reset_ = 1'b1;     // deassert reset  t=0
      #3 reset_ = 1'b0;  // assert reset    t=3
      #4 reset_ = 1'b1;  // deassert reset  t=7

      @(negedge clk) //will wait for next negative edge of the clock (t=20)
        Din = 32'hAAAA5555;
      @(negedge clk) //will wait for next negative edge of the clock (t=40)
        Din = 32'h5F5F0A0A;
      @(negedge clk) //will wait for next negative edge of the clock (t=60)
        Din = 32'hFEDCBA98;
      @(negedge clk) //will wait for next negative edge of the clock (t=80)
        reset_ = 1'b0;
      @(negedge clk) //will wait for next negative edge of the clock (t=100)
        reset_ = 1'b1;
      @(negedge clk) //will wait for next negative edge of the clock (t=120)
        Din = 32'h76543210;

      @(negedge clk) //will wait for next negative edge of the clock (t=140)
        $finish;     // to kill the simulation
   end

   // This block generates a clock pulse with a 20 ns period.
   // Rising edges at 10, 30, 50, 70 .... Falling edges at 20, 40, 60, 80 ....
   always
     #10 clk = ~ clk;

   // this block is sensitive to changes on clk or reset and will
   // then display both the inputs and corresponding output
   always @(posedge clk or reset_)
     #1 $display("At t=%t / Din=%h OutReg=%h" ,
                   $time,  Din,   OutReg);

endmodule
```

```
//*******************************************************
module Test_mux_4to1;
//*******************************************************

   wire [15:0] MuxOut;     //use wire data type for outputs from instantiated module
   reg  [15:0] A, B, C, D; //use reg data type for all inputs
   reg  [1:0]  sel;        //  to the instantiated module
   reg         clk;        //to be used for timing of WHEN to change input values

   // Instantiate the MUX (named DUT {device under test})
      mux_4to1 DUT(MuxOut, A, B, C, D, sel);

   //This block generates a clock pulse with a 20 ns period
   always
      #10  clk = ~clk;

   //This initial block will provide values for the inputs
   // of the mux so that both inputs/outputs can be displayed
   initial begin
      $timeformat(-9, 1, " ns", 6);
      clk = 1'b0;    // time = 0
      A = 16'hAAAA;  B = 16'h5555;  C = 16'h00FF;  D = 16'hFF00;  sel = 2'b00;

      @(negedge clk) //will wait for next negative edge of the clock (t=20)
        A = 16'h0000;
      @(negedge clk) //will wait for next negative edge of the clock (t=40)
        sel = 2'b01;
      @(negedge clk) //will wait for next negative edge of the clock (t=60)
        B = 16'hFFFF;
      @(negedge clk) //will wait for next negative edge of the clock (t=80)
        sel = 2'b10;
        A = 16'hA5A5;
      @(negedge clk) //will wait for next negative edge of the clock (t=100)
        sel = 2'b00;

      @(negedge clk) //will wait for next negative edge of the clock (t=120)
        $finish;     //  to shut down the simulation

   end  //initial


   // this block is sensitive to changes on ANY of the inputs and will
```

```
   // then display both the inputs and corresponding output
   always @(A or B or C or D or sel)
      #1 $display("At t=%t / sel=%b A=%h B=%h C=%h D=%h / MuxOut=%h",
                  $time, sel,   A,   B,   C,   D,     MuxOut);


endmodule
```