```
/*
   Ben Davis
   2/13/24
   EE 371
   Lab 4, Task 1

   This is a module that controls a datapath for finding
   how many 1's are in a byte. It has an input clock and reset.
   It also has a start user input and a done input from the
   datapath module. It has an output s to start the counting
   and an output shr to shift the data to the right.

*/

module lab4task1_FSM (
               input logic clk, //clock
               input logic reset, //reset
               input logic start, //start (from user)
               input logic done, //the datapath is exhausted
               output logic s, //start the counting
               output logic shr); //shift the data

   enum {s1, s2, s3} ps, ns; //states

   //if reset, go to s1. otherwise move to the next state
   always_ff @(posedge clk) begin
      if(reset) begin
         ps <= s1;
      end else begin
         ps <= ns;
      end
   end

   always_comb begin
      case(ps)

         s1: begin  //if start is pressed, move off this idle state
               if(start) ns <= s2;
               else ns <= s1;
             end

         s2: begin //if done is true, this working state no longer
               if(done)ns <= s3; //is needed
               else ns <= s2;
             end

         s3: begin //this state holds all values until reset
               if(start) ns <= s3;
               else ns <= s1;
             end
      endcase
   end

   assign s = (ps==s2); //start the counting
   assign shr = ((ps==s2)&&(~done)); //shift to the right

endmodule
//testbench
module lab4task1_FSM_tb ();

   //recall variables
   logic clk, reset, start, done, s, shr;

   //recall module
   lab4task1_FSM dut (.clk, .reset, .start, .done, .s, .shr);

   //clock setup
   parameter clk_pd = 100;
   initial begin
      clk <= 0;
      forever #(clk_pd /2) clk <= ~clk;
   end //of clock setup

```

```
74      //tests an instance where start is pressed after one cycle
75      //and takes five cycles to complete
76      initial begin
77         reset <= 1'b1; start <= 1'b0; done <= 1'b0; @(posedge clk);
78         reset <= 1'b0; start <= 1'b1; done <= 1'b0; @(posedge clk);
79         reset <= 1'b0; start <= 1'b1; done <= 1'b0; @(posedge clk);
80         reset <= 1'b0; start <= 1'b1; done <= 1'b0; @(posedge clk);
81         reset <= 1'b0; start <= 1'b1; done <= 1'b0; @(posedge clk);
82         reset <= 1'b0; start <= 1'b1; done <= 1'b0; @(posedge clk);
83         reset <= 1'b0; start <= 1'b1; done <= 1'b1; @(posedge clk);
84         reset <= 1'b0; start <= 1'b1; done <= 1'b1; @(posedge clk);
85         $stop;
86      end
87   endmodule //testbench
```