```
/*
   Non-restoring 32-bit division with quotient placed in Z[31:0] and remainder places in
Z[63:32].
   Operations takes 32 cycles to complete.
*/

module DIV(

    input [31:0] Q, // Dividend
    input [31:0] M, // Divisor
    input clk, resetn,

    output [31:0] quotient,
    output [31:0] remainder

);
   reg [63:0] AQ_reg;
   integer count;
   wire [31:0] M_signed = (M[31]) ? -M : M;

   always @(posedge clk) begin

      if (~resetn) begin

         AQ_reg = 64'b0;
         count = 0;

      end

      else if (count == 0) begin

         count = count + 1;

         AQ_reg = {32'b0, Q};

      end

      else if (count >= 1 && count <= 32) begin

         count = count + 1;

         AQ_reg = AQ_reg << 1;

         if (AQ_reg[63] == 1'b0) begin

            AQ_reg[63:32] = AQ_reg[63:32] - M_signed;

         end

         else begin

            AQ_reg[63:32] = AQ_reg[63:32] + M_signed;

         end

         AQ_reg[0] = (AQ_reg[63] == 1'b0) ? 1'b1 : 1'b0;

      end

      else if (AQ_reg[63] == 1) begin

         AQ_reg[63:32] = AQ_reg[63:32] + M_signed;

      end

   end


   assign quotient =  (M[31]) ? -AQ_reg[31:0] : AQ_reg[31:0];

   assign remainder =   AQ_reg[63:32];

endmodule
```

```
module DIV_tb;

    // Declare inputs as reg type
    reg [31:0] Q;
    reg [31:0] M;
    reg clk, resetn;

    // Declare outputs as wire type
    wire [31:0] quotient;
    wire [31:0] remainder;

    // Instantiate the DIV module
    DIV uut (
        .Q(Q),
        .M(M),
        .clk(clk),
        .resetn(resetn),
        .quotient(quotient),
        .remainder(remainder)
    );

    // Clock generation
    always begin
        clk = 0;
        forever #5 clk = ~clk;
    end

   initial begin

      resetn = 0;

      @ (posedge clk)

      resetn = 1;
      Q = 32'd38;
      M = 32'd6;

      #340;

      @ (posedge clk)

      resetn = 0;

      @ (posedge clk)

      resetn = 1;
      Q = 32'd100;
      M = 32'd25;
      #340;

      @ (posedge clk)

      resetn = 0;

      @ (posedge clk)

      resetn = 1;
      Q = {0,{31{1'b1}}};
      M = {0,{31{1'b1}}};
      #340;

      @ (posedge clk)

      resetn = 0;

      @ (posedge clk)

      resetn = 1;
      Q = {0,{31{1'b1}}};
      M = 32'b1;
      #340;

      @ (posedge clk)
```

```

        resetn = 0;

        @ (posedge clk)

        resetn = 1;
        Q = 32'b1;
        M = 32'd50;
        #340;

        @ (posedge clk)

        resetn = 0;

        @ (posedge clk)

        $stop;
    end

endmodule
```