```
1   /*
2      Non-restoring 32-bit division with quotient placed in Z[31:0] and remainder places in
    Z[63:32].
3      Operations takes 32 cycles to complete.
4   */
5
6   module DIV(
7
8       input [31:0] Q, // Dividend
9       input [31:0] M, // Divisor
10      input clk, resetn,
11
12      output [31:0] quotient,
13      output [31:0] remainder
14
15  );
16     reg [63:0] AQ_reg;
17     integer count;
18     wire [31:0] M_signed = (M[31]) ? -M : M;
19     wire [31:0] Q_signed = (Q[31]) ? -Q : Q;
20
21     always @(posedge clk) begin
22
23        if (~resetn) begin
24
25           AQ_reg = 64'b0;
26           count = 0;
27
28        end
29
30        else if (count == 0) begin
31
32           count = count + 1;
33
34           AQ_reg = {32'b0, Q_signed};
35
36        end
37
38        else if (count >= 1 && count <= 32) begin
39
40           count = count + 1;
41
42           AQ_reg = AQ_reg << 1;
43
44           if (AQ_reg[63] == 1'b0) begin
45
46              AQ_reg[63:32] = AQ_reg[63:32] - M_signed;
47
48           end
49
50           else begin
51
52              AQ_reg[63:32] = AQ_reg[63:32] + M_signed;
53
54           end
55
56           AQ_reg[0] = (AQ_reg[63] == 1'b0) ? 1'b1 : 1'b0;
57
58        end
59
60        else if (AQ_reg[63] == 1) begin
61
62           AQ_reg[63:32] = AQ_reg[63:32] + M_signed;
63
64        end
65
66     end
67
68
69
70     assign quotient =  (M[31] ^ Q[31]) ? -AQ_reg[31:0] : AQ_reg[31:0];
71
72     assign remainder =   AQ_reg[63:32];
73
74  endmodule
```

```
 75
 76  module DIV_tb;
 77
 78      // Declare inputs as reg type
 79      reg [31:0] Q;
 80      reg [31:0] M;
 81      reg clk, resetn;
 82
 83      // Declare outputs as wire type
 84      wire [31:0] quotient;
 85      wire [31:0] remainder;
 86
 87      // Instantiate the DIV module
 88      DIV uut (
 89          .Q(Q),
 90          .M(M),
 91          .clk(clk),
 92          .resetn(resetn),
 93          .quotient(quotient),
 94          .remainder(remainder)
 95      );
 96
 97      // Clock generation
 98      always begin
 99          clk = 0;
100          forever #5 clk = ~clk;
101      end
102
103     initial begin
104
105        resetn = 0;
106
107        @ (posedge clk)
108
109        resetn = 1;
110        Q = 32'd38;
111        M = 32'd6;
112
113        #340;
114
115        @ (posedge clk)
116
117        resetn = 0;
118
119        @ (posedge clk)
120
121        resetn = 1;
122        Q = 32'd100;
123        M = 32'd25;
124        #340;
125
126        @ (posedge clk)
127
128        resetn = 0;
129
130        @ (posedge clk)
131
132        resetn = 1;
133        Q = {0,{31{1'b1}}};
134        M = {0,{31{1'b1}}};
135        #340;
136
137        @ (posedge clk)
138
139        resetn = 0;
140
141        @ (posedge clk)
142
143        resetn = 1;
144        Q = {0,{31{1'b1}}};
145        M = 32'b1;
146        #340;
147
148        @ (posedge clk)
149
```

```
150         resetn = 0;
151
152         @ (posedge clk)
153
154         resetn = 1;
155         Q = 32'b1;
156         M = 32'd50;
157         #340;
158
159         @ (posedge clk)
160
161         resetn = 0;
162
163         @ (posedge clk)
164
165         $stop;
166      end
167
168   endmodule
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
```