```
module RegFile (RdestRegLoc, RsrcRegLoc, Clk, En, Rst, Load, RdestOut, RsrcOut);
   input [3:0] RdestRegLoc, RsrcRegLoc;
   input Clk, En, Rst;
   input [15:0] Load;

   output [15:0] RdestOut, RsrcOut;


   parameter reg00 = 4'b0000;
   parameter reg01 = 4'b0001;
   parameter reg02 = 4'b0010;
   parameter reg03 = 4'b0011;
   parameter reg04 = 4'b0100;
   parameter reg05 = 4'b0101;
   parameter reg06 = 4'b0110;
   parameter reg07 = 4'b0111;
   parameter reg08 = 4'b1000;
   parameter reg09 = 4'b1001;
   parameter reg10 = 4'b1010;
   parameter reg11 = 4'b1011;
   parameter reg12 = 4'b1100;
   parameter reg13 = 4'b1101;
   parameter reg14 = 4'b1110;
   parameter reg15 = 4'b1111;

   wire [15:0] Out[15:0], enable;

   Dec4to16 decoder(
      .in(RdestRegLoc),
      .E(En),
      .en(enable)
   );

   genvar i;
   generate
   for (i = 0; i < 16; i = i + 1)
      begin:registerForLoop
         Register register (
            .in(Load),
            .clk(Clk),
            .en(enable[i]),
            .rst(Rst),
            .out(Out[i])
         );
      end
   endgenerate

   MUX RdestMux (
      .in00(Out[0]),
      .in01(Out[1]),
      .in02(Out[2]),
      .in03(Out[3]),
      .in04(Out[4]),
      .in05(Out[5]),
      .in06(Out[6]),
      .in07(Out[7]),
      .in08(Out[8]),
      .in09(Out[9]),
      .in10(Out[10]),
      .in11(Out[11]),
      .in12(Out[12]),
      .in13(Out[13]),
      .in14(Out[14]),
      .in15(Out[15]),
      .loc(RdestRegLoc),
      .out(RdestOut)
   );

   MUX RsrcMux (
      .in00(Out[0]),
      .in01(Out[1]),
      .in02(Out[2]),
      .in03(Out[3]),
```

```
74          .in04(Out[4]),
75          .in05(Out[5]),
76          .in06(Out[6]),
77          .in07(Out[7]),
78          .in08(Out[8]),
79          .in09(Out[9]),
80          .in10(Out[10]),
81          .in11(Out[11]),
82          .in12(Out[12]),
83          .in13(Out[13]),
84          .in14(Out[14]),
85          .in15(Out[15]),
86          .loc(RsrcRegLoc),
87          .out(RsrcOut)
88       );
89
90    endmodule
91
92
93    module Register(in, clk, en, rst, out);
94       input [15:0] in;
95       input clk, en, rst;
96
97       output reg [15:0] out;
98
99       always @(negedge rst, posedge clk) begin
100         if (rst == 0)
101            out <= 16'b0;
102
103         else begin
104            if (en)
105               out <= in;
106            else
107               out <= out;
108         end
109      end
110   endmodule
111
112
113   module Dec4to16(in, E, en);
114      input [3:0] in;
115      input E;
116      output [15:0] en;
117
118      assign en[0]  = ~in[3] & ~in[2] & ~in[1] & ~in[0] & E;
119      assign en[1]  = ~in[3] & ~in[2] & ~in[1] &  in[0] & E;
120      assign en[2]  = ~in[3] & ~in[2] &  in[1] & ~in[0] & E;
121      assign en[3]  = ~in[3] & ~in[2] &  in[1] &  in[0] & E;
122      assign en[4]  = ~in[3] &  in[2] & ~in[1] & ~in[0] & E;
123      assign en[5]  = ~in[3] &  in[2] & ~in[1] &  in[0] & E;
124      assign en[6]  = ~in[3] &  in[2] &  in[1] & ~in[0] & E;
125      assign en[7]  = ~in[3] &  in[2] &  in[1] &  in[0] & E;
126      assign en[8]  =  in[3] & ~in[2] & ~in[1] & ~in[0] & E;
127      assign en[9]  =  in[3] & ~in[2] & ~in[1] &  in[0] & E;
128      assign en[10] =  in[3] & ~in[2] &  in[1] & ~in[0] & E;
129      assign en[11] =  in[3] & ~in[2] &  in[1] &  in[0] & E;
130      assign en[12] =  in[3] &  in[2] & ~in[1] & ~in[0] & E;
131      assign en[13] =  in[3] &  in[2] & ~in[1] &  in[0] & E;
132      assign en[14] =  in[3] &  in[2] &  in[1] & ~in[0] & E;
133      assign en[15] =  in[3] &  in[2] &  in[1] &  in[0] & E;
134   endmodule
135
136   module MUX(in00, in01, in02, in03, in04, in05, in06, in07, in08, in09, in10, in11, in12,
      in13, in14, in15, loc, out);
137      input [15:0] in00, in01, in02, in03, in04, in05, in06, in07, in08, in09, in10, in11, in12
      , in13, in14, in15;
138      input [3:0] loc;
139
140      output [15:0] out;
141
142      assign out = (loc == 4'b0000) ? in00 :
143                   (loc == 4'b0001) ? in01 :
144                   (loc == 4'b0010) ? in02 :
```

```
145                     (loc == 4'b0011) ? in03 :
146                     (loc == 4'b0100) ? in04 :
147                     (loc == 4'b0101) ? in05 :
148                     (loc == 4'b0110) ? in06 :
149                     (loc == 4'b0111) ? in07 :
150                     (loc == 4'b1000) ? in08 :
151                     (loc == 4'b1001) ? in09 :
152                     (loc == 4'b1010) ? in10 :
153                     (loc == 4'b1011) ? in11 :
154                     (loc == 4'b1100) ? in12 :
155                     (loc == 4'b1101) ? in13 :
156                     (loc == 4'b1110) ? in14 :
157                     (loc == 4'b1111) ? in15 : 16'bx;
158
159   endmodule
160
```