```
 1  module proc(DIN, Resetn, Clock, Run, DOUT, ADDR, W);
 2      input [15:0] DIN;
 3      input Resetn, Clock, Run;
 4      output wire [15:0] DOUT;
 5      output wire [15:0] ADDR;
 6      output wire W;
 7
 8      reg [15:0] BusWires;
 9      reg [3:0] Sel; // BusWires selector
10      reg [0:7] Rin;
11      reg [15:0] Sum;
12      reg IRin, ADDRin, Done, DOUTin, Ain, Gin, AddSub, ALUand;
13      reg [2:0] Tstep_Q, Tstep_D;
14      wire [2:0] III, rX, rY; // instruction opcode and register operands
15      wire [0:7] Xreg;
16      wire [15:0] R0, R1, R2, R3, R4, R5, R6, PC, A;
17      wire [15:0] G;
18      wire [15:0] IR;
19      reg pc_inc, W_D;
20      wire IMM;
21
22      assign III = IR[15:13];
23      assign IMM = IR[12];
24      assign rX = IR[11:9];
25      assign rY = IR[2:0];
26      dec3to8 decX (rX, Xreg);
27
28      parameter T0 = 3'b000, T1 = 3'b001, T2 = 3'b010, T3 = 3'b011, T4 =
    3'b100, T5 = 3'b101;
29
30      // Control FSM state table
31      always @(Tstep_Q, Run, Done)
32          case (Tstep_Q)
33              T0: // instruction fetch
34                  if (~Run) Tstep_D = T0;
35                  else Tstep_D = T1;
36              T1: // wait cycle for synchronous memory
37                  Tstep_D = T2;
38              T2: // this time step stores the instruction word in IR
39                  Tstep_D = T3;
40              T3: // some instructions end after this time step
41                  if (Done) Tstep_D = T0;
42                  else Tstep_D = T4;
43              T4: // always go to T5 after this
44                  Tstep_D = T5;
45              T5: // instructions end after this time step
46                  Tstep_D = T0;
47              default: Tstep_D = 3'bxxx;
48          endcase
49
50      /* OPCODE format: III M XXX DDDDDDDDD, where
51      *     III = instruction, M = Immediate, XXX = rX. If M = 0,
    DDDDDDDDD = 000000YYY = rY
52      *     If M = 1, DDDDDDDDD = #D is the immediate operand
53      *
54      *  III M  Instruction   Description
55      *  --- -  -----------   -----------
56      *  000 0: mv    rX,rY    rX <- rY
57      *  000 1: mv    rX,#D    rX <- D (0 extended)
58      *  001 1: mvt   rX,#D    rX <- D << 8
59      *  010 0: add   rX,rY    rX <- rX + rY
60      *  010 1: add   rX,#D    rX <- rX + D
```

```
61          *  011 0: sub  rX,rY    rX <- rX - rY
62          *  011 1: sub  rX,#D    rX <- rX - D
63          *  100 0: ld   rX,[rY]  rX <- [rY]
64          *  101 0: st   rX,[rY]  [rY] <- rX
65          *  110 0: and  rX,rY    rX <- rX & rY
66          *  110 1: and  rX,#D    rX <- rX & D */
67          parameter mv = 3'b000, mvt = 3'b001, add = 3'b010, sub = 3'b011, ld
    = 3'b100, st = 3'b101,
68                    and_ = 3'b110;
69          // selectors for the BusWires multiplexer
70          parameter Sel_R0 = 4'b0000, Sel_R1 = 4'b0001, Sel_R2 = 4'b0010,
    Sel_R3 = 4'b0011,
71                    Sel_R4 = 4'b0100, Sel_R5 = 4'b0101, Sel_R6 = 4'b0110, Sel_PC =
    4'b0111, Sel_G = 4'b1000,
72                    Sel_D /* immediate data */  = 4'b1001, Sel_D8 /* immediate data
    << 8 */ = 4'b1010,
73                    Sel_DIN /* data-in from memory */  = 4'b1011;
74          // Control FSM outputs
75          always @(*) begin
76              // default values for control signals
77              Done = 1'b0; Ain = 1'b0; Gin = 1'b0; AddSub = 1'b0; IRin = 1'b0;
     Sel = 4'bxxxx;
78              DOUTin = 1'b0; ADDRin = 1'b0; W_D = 1'b0; Rin = 8'b0; pc_inc =
    1'b0; ALUand = 1'b0;
79              case (Tstep_Q)
80                  T0: begin // fetch the instruction
81                      Sel = Sel_PC; // put pc onto the internal bus
82                      ADDRin = 1'b1;
83                      pc_inc = Run; // to increment pc
84                  end
85                  T1: // wait cycle for synchronous memory
86                      ;
87                  T2: // store instruction on DIN in IR
88                      IRin = 1'b1;
89                  T3: // define signals in T1
90                      case (III)
91                          mv: begin
92                              if (!IMM) Sel = rY;    // mv rX, rY
93                              else Sel = Sel_D;      // mv rX, #D
94                              Rin = Xreg;
95                              Done = 1'b1;
96                          end
97                          mvt: begin
98                              // ... your code goes here
99                              Sel = Sel_D8;
100                             Rin = Xreg;
101                             Done = 1'b1;
102                         end
103                         add, sub, and_: begin
104                             // ... your code goes here
105                             Sel = rX;
106                             Ain = 1'b1;
107                         end
108                         ld, st: begin
109                             // ... your code goes here
110                             Sel = rY;
111                             ADDRin = 1'b1;
112                         end
113                         default: ;
114                     endcase
115                 T4: // define signals T2
116                     case (III)
```

```
117                         add: begin
118                             // ... your code goes here
119                             if (!IMM) Sel = rY;
120                             else Sel = Sel_D;
121                             AddSub = 1'b0;
122                             Gin = 1'b1;
123                         end
124                         sub: begin
125                             // ... your code goes here
126                             if (!IMM) Sel = rY;
127                             else Sel = Sel_D;
128                             AddSub = 1'b1;
129                             Gin = 1'b1;
130                         end
131                         and_: begin
132                             // ... your code goes here
133                             if (!IMM) Sel = rY;
134                             else Sel = Sel_D;
135                             ALUand = 1'b1;
136                             Gin = 1'b1;
137                         end
138                         ld: // wait cycle for synchronous memory
139                             ;
140                         st: begin
141                             // ... your code goes here
142                             Sel = rX;
143                             DOUTin = 1'b1;
144                             W_D = 1'b1;
145                         end
146                         default: ;
147                     endcase
148                 T5: // define T3
149                     case (III)
150                         add, sub, and_: begin
151                             // ... your code goes here
152                             Sel = Sel_G;
153                             Rin = Xreg;
154                             Done = 1'b1;
155                         end
156                         ld: begin
157                             // ... your code goes here
158                             Sel = Sel_DIN;
159                             Rin = Xreg;
160                             Done = 1'b1;
161                         end
162                         st: // wait cycle for synhronous memory
163                             // ... your code goes here
164                             Done = 1'b1;
165                         default: ;
166                     endcase
167                 default: ;
168             endcase
169         end
170 
171         // Control FSM flip-flops
172         always @(posedge Clock)
173             if (!Resetn)
174                 Tstep_Q <= T0;
175             else
176                 Tstep_Q <= Tstep_D;
177 
178         regn reg_0 (BusWires, Rin[0], Clock, R0);
```

```
179         regn reg_1 (BusWires, Rin[1], Clock, R1);
180         regn reg_2 (BusWires, Rin[2], Clock, R2);
181         regn reg_3 (BusWires, Rin[3], Clock, R3);
182         regn reg_4 (BusWires, Rin[4], Clock, R4);
183         regn reg_5 (BusWires, Rin[5], Clock, R5);
184         regn reg_6 (BusWires, Rin[6], Clock, R6);
185 
186         // R7 is program counter
187         // module pc_count(R, Resetn, Clock, E, L, Q);
188         pc_count pc (BusWires, Resetn, Clock, pc_inc, Rin[7], PC);
189 
190         regn reg_A (BusWires, Ain, Clock, A);
191         regn reg_DOUT (BusWires, DOUTin, Clock, DOUT);
192         regn reg_ADDR (BusWires, ADDRin, Clock, ADDR);
193         regn reg_IR (DIN, IRin, Clock, IR);
194 
195         flipflop reg_W (W_D, Resetn, Clock, W);
196 
197         // alu
198         always @(*)
199             if (!ALUand)
200                 if (!AddSub)
201                     Sum = A + BusWires;
202                 else
203                     Sum = A - BusWires;
204             else
205                 Sum = A & BusWires;
206         regn reg_G (Sum, Gin, Clock, G);
207 
208         // define the internal processor bus
209         always @(*)
210             case (Sel)
211                 Sel_R0: BusWires = R0;
212                 Sel_R1: BusWires = R1;
213                 Sel_R2: BusWires = R2;
214                 Sel_R3: BusWires = R3;
215                 Sel_R4: BusWires = R4;
216                 Sel_R5: BusWires = R5;
217                 Sel_R6: BusWires = R6;
218                 Sel_PC: BusWires = PC;
219                 Sel_G: BusWires = G;
220                 Sel_D: BusWires = {7'b0000000, IR[8:0]};
221                 Sel_D8: BusWires = {IR[7:0], 8'b00000000};
222                 default: BusWires = DIN;
223             endcase
224     endmodule
225 
226     module pc_count(R, Resetn, Clock, E, L, Q);
227         input [15:0] R;
228         input Resetn, Clock, E, L;
229         output [15:0] Q;
230         reg [15:0] Q;
231 
232         always @(posedge Clock)
233             if (!Resetn)
234                 Q <= 9'b0;
235             else if (L)
236                 Q <= R;
237             else if (E)
238                 Q <= Q + 1'b1;
239     endmodule
240 
```

```
241   module dec3to8(W, Y);
242       input [2:0] W;
243       output [0:7] Y;
244       reg [0:7] Y;
245
246       always @(*)
247           case (W)
248               3'b000: Y = 8'b10000000;
249               3'b001: Y = 8'b01000000;
250               3'b010: Y = 8'b00100000;
251               3'b011: Y = 8'b00010000;
252               3'b100: Y = 8'b00001000;
253               3'b101: Y = 8'b00000100;
254               3'b110: Y = 8'b00000010;
255               3'b111: Y = 8'b00000001;
256           endcase
257   endmodule
258
259   module regn(R, Rin, Clock, Q);
260       parameter n = 16;
261       input [n-1:0] R;
262       input Rin, Clock;
263       output [n-1:0] Q;
264       reg [n-1:0] Q;
265
266       always @(posedge Clock)
267           if (Rin)
268               Q <= R;
269   endmodule
270
```