```
 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 [16:0] 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, cond; // 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 [16:0] G;
18      wire [15:0] IR;
19      reg pc_inc, W_D;
20      wire IMM, z, c;
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      assign cond = IR[11:9];
27      assign z = ~|G[15:0];
28      assign c = G[16];
29      dec3to8 decX (rX, Xreg);
30
31      parameter T0 = 3'b000, T1 = 3'b001, T2 = 3'b010, T3 = 3'b011, T4 =
    3'b100, T5 = 3'b101;
32
33      // Control FSM state table
34      always @(Tstep_Q, Run, Done)
35          case (Tstep_Q)
36              T0: // instruction fetch
37                  if (~Run) Tstep_D = T0;
38                  else Tstep_D = T1;
39              T1: // wait cycle for synchronous memory
40                  Tstep_D = T2;
41              T2: // this time step stores the instruction word in IR
42                  Tstep_D = T3;
43              T3: // some instructions end after this time step
44                  if (Done) Tstep_D = T0;
45                  else Tstep_D = T4;
46              T4: // always go to T5 after this
47                  Tstep_D = T5;
48              T5: // instructions end after this time step
49                  Tstep_D = T0;
50              default: Tstep_D = 3'bxxx;
51          endcase
52
53      /* OPCODE format: III M XXX DDDDDDDDD, where
54      *     III = instruction, M = Immediate, XXX = rX. If M = 0,
    DDDDDDDDD = 000000YYY = rY
55      *     If M = 1, DDDDDDDDD = #D is the immediate operand
56      *
57      *  III M  Instruction   Description
58      *  --- -  -----------   -----------
59      *  000 0: mv    rX,rY    rX <- rY
```

```
60          *  000 1: mv   rX,#D    rX <- D (0 extended)
61          *  001 1: mvt  rX,#D    rX <- D << 8
62          *  010 0: add  rX,rY    rX <- rX + rY
63          *  010 1: add  rX,#D    rX <- rX + D
64          *  011 0: sub  rX,rY    rX <- rX - rY
65          *  011 1: sub  rX,#D    rX <- rX - D
66          *  100 0: ld   rX,[rY]  rX <- [rY]
67          *  101 0: st   rX,[rY]  [rY] <- rX
68          *  110 0: and  rX,rY    rX <- rX & rY
69          *  110 1: and  rX,#D    rX <- rX & D
70          *  111 1: b{cond}                  */
71          parameter mv = 3'b000, mvt = 3'b001, add = 3'b010, sub = 3'b011, ld
        = 3'b100, st = 3'b101,
72                    and_ = 3'b110, bxx = 3'b111;
73          // {cond} parameters
74          parameter none = 3'b000, eq = 3'b001, ne = 3'b010, cc = 3'b011, cs =
         3'b100,
75                    PCreg = 8'b1;
76          // selectors for the BusWires multiplexer
77          parameter Sel_R0 = 4'b0000, Sel_R1 = 4'b0001, Sel_R2 = 4'b0010,
        Sel_R3 = 4'b0011,
78                    Sel_R4 = 4'b0100, Sel_R5 = 4'b0101, Sel_R6 = 4'b0110, Sel_PC =
        4'b0111, Sel_G = 4'b1000,
79                    Sel_D /* immediate data */  = 4'b1001, Sel_D8 /* immediate data
        << 8 */ = 4'b1010,
80                    Sel_DIN /* data-in from memory */  = 4'b1011;
81          // Control FSM outputs
82          always @(*) begin
83              // default values for control signals
84              Done = 1'b0; Ain = 1'b0; Gin = 1'b0; AddSub = 1'b0; IRin = 1'b0;
         Sel = 4'bxxxx;
85              DOUTin = 1'b0; ADDRin = 1'b0; W_D = 1'b0; Rin = 8'b0; pc_inc =
        1'b0; ALUand = 1'b0;
86              case (Tstep_Q)
87                  T0: begin // fetch the instruction
88                      Sel = Sel_PC; // put pc onto the internal bus
89                      ADDRin = 1'b1;
90                      pc_inc = Run; // to increment pc
91                  end
92                  T1: // wait cycle for synchronous memory
93                      ;
94                  T2: // store instruction on DIN in IR
95                      IRin = 1'b1;
96                  T3: // define signals in T1
97                      case (III)
98                          mv: begin
99                              if (!IMM) Sel = rY;    // mv rX, rY
100                             else Sel = Sel_D;      // mv rX, #D
101                             Rin = Xreg;
102                             Done = 1'b1;
103                         end
104                         mvt: begin
105                             Sel = Sel_D8;
106                             Rin = Xreg;
107                             Done = 1'b1;
108                         end
109                         add, sub, and_: begin
110                             Sel = rX;
111                             Ain = 1'b1;
112                         end
113                         ld, st: begin
114                             Sel = rY;
```

```
115                         ADDRin = 1'b1;
116                     end
117                     bxx: begin
118                         case (cond)
119                             none: begin
120                                 Sel = Sel_D; // Selects branch addr
121                                 Rin = PCreg;
122                                 Done = 1'b1;
123                             end
124                             eq: begin // equal to zero
125                                 if (z) begin
126                                     Sel = Sel_D;
127                                     Rin = PCreg;
128                                 end
129                                 Done = 1'b1;
130                             end
131                             ne: begin // not equal to zero
132                                 if (!z) begin
133                                     Sel = Sel_D;
134                                     Rin = PCreg;
135                                 end
136                                 Done = 1'b1;
137                             end
138                             cc: begin // carry clear
139                                 if (!c) begin
140                                     Sel = Sel_D;
141                                     Rin = PCreg;
142                                 end
143                                 Done = 1'b1;
144                             end
145                             cs: begin // carry set
146                                 if (c) begin
147                                     Sel = Sel_D;
148                                     Rin = PCreg;
149                                 end
150                                 Done = 1'b1;
151                             end
152                         endcase
153                     end
154                     default: ;
155                 endcase
156             T4: // define signals T2
157                 case (III)
158                     add: begin
159                         if (!IMM) Sel = rY;
160                         else Sel = Sel_D;
161                         AddSub = 1'b0;
162                         Gin = 1'b1;
163                     end
164                     sub: begin
165                         if (!IMM) Sel = rY;
166                         else Sel = Sel_D;
167                         AddSub = 1'b1;
168                         Gin = 1'b1;
169                     end
170                     and_: begin
171                         if (!IMM) Sel = rY;
172                         else Sel = Sel_D;
173                         ALUand = 1'b1;
174                         Gin = 1'b1;
175                     end
176                     ld: // wait cycle for synchronous memory
```

```
177                         ;
178                     st: begin
179                         Sel = rX;
180                         DOUTin = 1'b1;
181                         W_D = 1'b1;
182                     end
183                     default: ;
184                 endcase
185             T5: // define T3
186                 case (III)
187                     add, sub, and_: begin
188                         Sel = Sel_G;
189                         Rin = Xreg;
190                         Done = 1'b1;
191                     end
192                     ld: begin
193                         Sel = Sel_DIN;
194                         Rin = Xreg;
195                         Done = 1'b1;
196                     end
197                     st: // wait cycle for synhronous memory
198                         Done = 1'b1;
199                     default: ;
200                 endcase
201             default: ;
202         endcase
203     end
204
205     // Control FSM flip-flops
206     always @(posedge Clock)
207         if (!Resetn)
208             Tstep_Q <= T0;
209         else
210             Tstep_Q <= Tstep_D;
211
212     regn reg_0 (BusWires, Rin[0], Clock, R0);
213     regn reg_1 (BusWires, Rin[1], Clock, R1);
214     regn reg_2 (BusWires, Rin[2], Clock, R2);
215     regn reg_3 (BusWires, Rin[3], Clock, R3);
216     regn reg_4 (BusWires, Rin[4], Clock, R4);
217     regn reg_5 (BusWires, Rin[5], Clock, R5);
218     regn reg_6 (BusWires, Rin[6], Clock, R6);
219
220     // R7 is program counter
221     // module pc_count(R, Resetn, Clock, E, L, Q);
222     pc_count pc (BusWires, Resetn, Clock, pc_inc, Rin[7], PC);
223
224     regn reg_A (BusWires, Ain, Clock, A);
225     regn reg_DOUT (BusWires, DOUTin, Clock, DOUT);
226     regn reg_ADDR (BusWires, ADDRin, Clock, ADDR);
227     regn reg_IR (DIN, IRin, Clock, IR);
228
229     flipflop reg_W (W_D, Resetn, Clock, W);
230
231     // alu
232     always @(*)
233         if (!ALUand)
234             if (!AddSub)
235                 Sum = A + BusWires;
236             else
237                 Sum = A - BusWires;
238         else
```

```
239                 Sum = A & BusWires;
240         regn #(.n(17)) reg_G (Sum, Gin, Clock, G);
241
242         // define the internal processor bus
243         always @(*)
244             case (Sel)
245                 Sel_R0: BusWires = R0;
246                 Sel_R1: BusWires = R1;
247                 Sel_R2: BusWires = R2;
248                 Sel_R3: BusWires = R3;
249                 Sel_R4: BusWires = R4;
250                 Sel_R5: BusWires = R5;
251                 Sel_R6: BusWires = R6;
252                 Sel_PC: BusWires = PC;
253                 Sel_G: BusWires = G[15:0];
254                 Sel_D: BusWires = {7'b0000000, IR[8:0]};
255                 Sel_D8: BusWires = {IR[7:0], 8'b00000000};
256                 default: BusWires = DIN;
257             endcase
258     endmodule
259
260     module pc_count(R, Resetn, Clock, E, L, Q);
261         input [15:0] R;
262         input Resetn, Clock, E, L;
263         output [15:0] Q;
264         reg [15:0] Q;
265
266         always @(posedge Clock)
267             if (!Resetn)
268                 Q <= 9'b0;
269             else if (L)
270                 Q <= R;
271             else if (E)
272                 Q <= Q + 1'b1;
273     endmodule
274
275     module dec3to8(W, Y);
276         input [2:0] W;
277         output [0:7] Y;
278         reg [0:7] Y;
279
280         always @(*)
281             case (W)
282                 3'b000: Y = 8'b10000000;
283                 3'b001: Y = 8'b01000000;
284                 3'b010: Y = 8'b00100000;
285                 3'b011: Y = 8'b00010000;
286                 3'b100: Y = 8'b00001000;
287                 3'b101: Y = 8'b00000100;
288                 3'b110: Y = 8'b00000010;
289                 3'b111: Y = 8'b00000001;
290             endcase
291     endmodule
292
293     module regn(R, Rin, Clock, Q);
294         parameter n = 16;
295         input [n-1:0] R;
296         input Rin, Clock;
297         output [n-1:0] Q;
298         reg [n-1:0] Q;
299
300         always @(posedge Clock)
```

```
301             if (Rin)
302                 Q <= R;
303     endmodule
304
```