



```
1 // Verilog test fixture created from schematic C:\Users\Kyle\Xilinx\RISC\TOP.sch - Sun No
29 16:42:36 2009
2
3 `timescale 1ns / 1ps
4
5 module TOP_TOP_sch_tb();
6
7 // Inputs
8 reg CLOCK;
9 reg RESET;
10
11 // Output
12 wire [31:0] PCin;
13 wire [31:0] PC_;
14 wire [31:0] PC__;
15 wire [31:0] BranchAddress;
16 wire [31:0] ConstantUnit_;
17 wire [4:0] ain;
18 wire [4:0] bin;
19 wire [31:0] a;
20 wire [31:0] b;
21 wire [31:0] f;
22 wire [31:0] f_;
23 wire [31:0] WB;
24 wire [4:0] sh;
25 wire [4:0] fs;
26
27 wire rw;
28 wire [4:0] da;
29 wire [1:0] md;
30 wire rw2;
31 wire [4:0] da_;
32
33 wire [31:0] data;
34 wire [1:0] md_;
35
36 wire mw;
37 wire ps;
38
39 wire [31:0] IR_;
40 wire [1:0] CS;
41 wire bs1;
42 wire bs0;
43
44 wire [31:0] v;
45
46 wire z;
47
48
49
50 // Bidirs
51
52 // Instantiate the UUT
53 TOP UUT (
54     .PCin(PCin),
55     .PC_(PC_),
56     .ConstantUnit_(ConstantUnit_),
57     .a(a),
58     .sh(sh),
59     .fs(fs),
60     .f(f),
```

```
61         .rw(rw),
62         .da(da),
63         .md(md),
64         .rw2(rw2),
65         .da_(da_),
66         .f_(f_),
67         .data(data),
68         .md_(md_),
69         .WB(WB),
70         .mw(mw),
71         .ps(ps),
72         .BranchAddress(BranchAddress),
73         .IR_(IR_),
74         .CS(CS),
75         .bs1(bs1),
76         .bs0(bs0),
77         .CLOCK(CLOCK),
78         .RESET(RESET),
79         .b(b),
80         .v(v),
81         .PC__(PC__),
82         .z(z),
83         .ain(ain),
84         .bin(bin)
85
86     );
87 initial begin
88     // Initialize Inputs
89     CLOCK = 0;
90     RESET = 0;
91     // Add stimulus here
92
93 end
94
95 always
96     begin
97         CLOCK = ~CLOCK;
98         #50;
99     end
100
101 endmodule
102
```

```
1      `timescale 1ns / 1ps
2      //////////////////////////////////////////////////
3      // Company:
4      // Engineer:
5      //
6      // Create Date:      16:10:54 11/15/2009
7      // Design Name:
8      // Module Name:      AddressInstructionMemoryInstruction
9      // Project Name:
10     // Target Devices:
11     // Tool versions:
12     // Description:
13     //
14     // Dependencies:
15     //
16     // Revision:
17     // Revision 0.01 - File Created
18     // Additional Comments:
19     //
20     //////////////////////////////////////////////////
21     module InstructionMemory(
22     //Read ports
23         input  [31:0] PCin,
24         output [31:0] IRout
25
26     );
27
28     `include "Instructions.txt"
29     `include "Registers.txt"
30
31     reg [31:0] IR;
32
33     assign IRout = IR;
34
35
36     reg [31:0] InstructionMemory [31:0];    // 64 Registers of 32 bits
37
38     reg [31:0] MemLoc;
39
40
41     initial
42     begin
43
44         //////////////////////////////////////////////////
45         //MOV R5, R0
46         //Opcode
47         MemLoc = 0;
48         InstructionMemory[MemLoc][31:25] = MOV;
49         //DR
50         InstructionMemory[MemLoc][24:20] = R5;
51         //SA
52         InstructionMemory[MemLoc][19:15] = R0;
53         //IM
54         InstructionMemory[MemLoc][14:0] = 0;
55         MemLoc = MemLoc + 1;
56         //////////////////////////////////////////////////
57         //NOP
58         //Opcode
59         InstructionMemory[MemLoc][31:25] = NOP;
60         MemLoc = MemLoc + 1;
61         //////////////////////////////////////////////////
```

```

62      //AIU R5, 0||1000000000000000
63      //Opcode
64      InstructionMemory[MemLoc][31:25] = AIU;
65      //DR
66      InstructionMemory[MemLoc][24:20] = R5;
67      //SA
68      InstructionMemory[MemLoc][19:15] = R5;
69      //IM
70      InstructionMemory[MemLoc][14:0] = 15'b1000000000000000;
71      MemLoc = MemLoc + 1;
72      //////////////////////////////////////
73      //NOP
74      //Opcode
75      InstructionMemory[MemLoc][31:25] = NOP;
76      MemLoc = MemLoc + 1;
77      //////////////////////////////////////
78      //LSL R5, 1
79      //Opcode
80      InstructionMemory[MemLoc][31:25] = LSL;
81      //DR
82      InstructionMemory[MemLoc][24:20] = R5;
83      //SA
84      InstructionMemory[MemLoc][19:15] = R5;
85      //IM
86      InstructionMemory[MemLoc][14:0] = 1;
87      MemLoc = MemLoc + 1;
88      //////////////////////////////////////
89      //MOV R10, R0
90      //Opcode
91      InstructionMemory[MemLoc][31:25] = MOV;
92      //DR
93      InstructionMemory[MemLoc][24:20] = R10;
94      //SA
95      InstructionMemory[MemLoc][19:15] = R0;
96      //IM
97      InstructionMemory[MemLoc][14:0] = 0;
98      MemLoc = MemLoc + 1;
99      //////////////////////////////////////
100     //NOP
101     //Opcode
102     InstructionMemory[MemLoc][31:25] = NOP;
103     MemLoc = MemLoc + 1;
104     //////////////////////////////////////
105     //AIU R10, R10, 0||000000000100000
106     //Opcode
107     InstructionMemory[MemLoc][31:25] = AIU;
108     //DR
109     InstructionMemory[MemLoc][24:20] = R10;
110     //SA
111     InstructionMemory[MemLoc][19:15] = R10;
112     //IM
113     InstructionMemory[MemLoc][14:0] = 15'b000000000100000;
114     MemLoc = MemLoc + 1;
115     //////////////////////////////////////
116     //MOV R3, R0
117     //Opcode
118     InstructionMemory[MemLoc][31:25] = MOV;
119     //DR
120     InstructionMemory[MemLoc][24:20] = R3;
121     //SA
122     InstructionMemory[MemLoc][19:15] = R0;

```

```
123 //IM
124 InstructionMemory[MemLoc][14:0] = 0;
125 MemLoc = MemLoc + 1;
126 ///////////////////////////////////////////////////
127 //ANI R8, R2, 0|0000000000000001
128 //Opcode
129 InstructionMemory[MemLoc][31:25] = ANI;
130 //DR
131 InstructionMemory[MemLoc][24:20] = R8;
132 //SA
133 InstructionMemory[MemLoc][19:15] = R2;
134 //IM
135 InstructionMemory[MemLoc][14:0] = 15'b0000000000000001;
136 MemLoc = MemLoc + 1;
137 ///////////////////////////////////////////////////
138 //NOP
139 //Opcode
140 InstructionMemory[MemLoc][31:25] = NOP;
141 MemLoc = MemLoc + 1;
142 ///////////////////////////////////////////////////
143 //BZ R8, 6
144 //Opcode
145 InstructionMemory[MemLoc][31:25] = BZ;
146 //DR
147 InstructionMemory[MemLoc][24:20] = R0;
148 //SA
149 InstructionMemory[MemLoc][19:15] = R8;
150 //Offset
151 InstructionMemory[MemLoc][14:0] = 6;
152 MemLoc = MemLoc + 1;
153 ///////////////////////////////////////////////////
154 //NOP
155 //Opcode
156 InstructionMemory[MemLoc][31:25] = NOP;
157 MemLoc = MemLoc + 1;
158 ///////////////////////////////////////////////////
159 //NOP
160 //Opcode
161 InstructionMemory[MemLoc][31:25] = NOP;
162 MemLoc = MemLoc + 1;
163 ///////////////////////////////////////////////////
164 //AND R7, R3, R5
165 //Opcode
166 InstructionMemory[MemLoc][31:25] = AND;
167 //DR
168 InstructionMemory[MemLoc][24:20] = R7;
169 //SA
170 InstructionMemory[MemLoc][19:15] = R3;
171 //IM
172 InstructionMemory[MemLoc][14:10] = R5;
173 MemLoc = MemLoc + 1;
174 ///////////////////////////////////////////////////
175 //ADD R3, R3, R1
176 //Opcode
177 InstructionMemory[MemLoc][31:25] = ADD;
178 //DR
179 InstructionMemory[MemLoc][24:20] = R3;
180 //SA
181 InstructionMemory[MemLoc][19:15] = R3;
182 //IM
183 InstructionMemory[MemLoc][14:10] = R1;
```

```
184      MemLoc = MemLoc + 1;
185      //////////////////////////////////////////
186      //NOP
187      //Opcode
188      InstructionMemory[MemLoc][31:25] = NOP;
189      MemLoc = MemLoc + 1;
190      //////////////////////////////////////////
191      //ANI R8, R3, 0||0000000000000001
192      //Opcode
193      InstructionMemory[MemLoc][31:25] = ANI;
194      //DR
195      InstructionMemory[MemLoc][24:20] = R8;
196      //SA
197      InstructionMemory[MemLoc][19:15] = R3;
198      //IM
199      InstructionMemory[MemLoc][14:0] = 15'b0000000000000001;
200      MemLoc = MemLoc + 1;
201      //////////////////////////////////////////
202      //NOP
203      //Opcode
204      InstructionMemory[MemLoc][31:25] = NOP;
205      MemLoc = MemLoc + 1;
206      //////////////////////////////////////////
207      //LSR R3, R3, 0||0000000000000001
208      //Opcode
209      InstructionMemory[MemLoc][31:25] = LSR;
210      //DR
211      InstructionMemory[MemLoc][24:20] = R3;
212      //SA
213      InstructionMemory[MemLoc][19:15] = R3;
214      //IM
215      InstructionMemory[MemLoc][14:0] = 15'b0000000000000001;
216      MemLoc = MemLoc + 1;
217      //////////////////////////////////////////
218      //NOP
219      //Opcode
220      InstructionMemory[MemLoc][31:25] = NOP;
221      MemLoc = MemLoc + 1;
222      //////////////////////////////////////////
223      //ADD R3, R3, R7
224      //Opcode
225      InstructionMemory[MemLoc][31:25] = ADD;
226      //DR
227      InstructionMemory[MemLoc][24:20] = R3;
228      //SA
229      InstructionMemory[MemLoc][19:15] = R3;
230      //IM
231      InstructionMemory[MemLoc][14:10] = R7;
232      MemLoc = MemLoc + 1;
233      //////////////////////////////////////////
234      //NOP
235      //Opcode
236      InstructionMemory[MemLoc][31:25] = NOP;
237      MemLoc = MemLoc + 1;
238      //////////////////////////////////////////
239      //LSR R2, R2, 0||0000000000000001
240      //Opcode
241      InstructionMemory[MemLoc][31:25] = LSR;
242      //DR
243      InstructionMemory[MemLoc][24:20] = R2;
244      //SA
```

```
245     InstructionMemory[MemLoc][19:15] = R2;
246     //IM
247     InstructionMemory[MemLoc][14:0] = 15'b0000000000000001;
248     MemLoc = MemLoc + 1;
249     //////////////////////////////////////
250     //NOP
251     //Opcode
252     InstructionMemory[MemLoc][31:25] = NOP;
253     MemLoc = MemLoc + 1;
254     //////////////////////////////////////
255     //BZ R2, 6
256     //Opcode
257     InstructionMemory[MemLoc][31:25] = BZ;
258     //DR
259     InstructionMemory[MemLoc][24:20] = R0;
260     //SA
261     InstructionMemory[MemLoc][19:15] = R2;
262     //Offset
263     InstructionMemory[MemLoc][14:0] = 6;
264     MemLoc = MemLoc + 1;
265     //////////////////////////////////////
266     //NOP
267     //Opcode
268     InstructionMemory[MemLoc][31:25] = NOP;
269     MemLoc = MemLoc + 1;
270     //////////////////////////////////////
271     //NOP
272     //Opcode
273     InstructionMemory[MemLoc][31:25] = NOP;
274     MemLoc = MemLoc + 1;
275     //////////////////////////////////////
276     //ADD R2, R2, R5
277     //Opcode
278     InstructionMemory[MemLoc][31:25] = ADD;
279     //DR
280     InstructionMemory[MemLoc][24:20] = R2;
281     //SA
282     InstructionMemory[MemLoc][19:15] = R2;
283     //IM
284     InstructionMemory[MemLoc][14:10] = R5;
285     MemLoc = MemLoc + 1;
286     //////////////////////////////////////
287     //SIU R2, R2, R5
288     //Opcode
289     InstructionMemory[MemLoc][31:25] = SIU;
290     //DR
291     InstructionMemory[MemLoc][24:20] = R10;
292     //SA
293     InstructionMemory[MemLoc][19:15] = R10;
294     //IM
295     InstructionMemory[MemLoc][14:0] = 15'b0000000000000001;
296     MemLoc = MemLoc + 1;
297     //////////////////////////////////////
298     //NOP
299     //Opcode
300     InstructionMemory[MemLoc][31:25] = NOP;
301     MemLoc = MemLoc + 1;
302     //////////////////////////////////////
303     //BNZ R2, 6
304     //Opcode
305     InstructionMemory[MemLoc][31:25] = BZ;
```



```
306 //DR
307 InstructionMemory[MemLoc][24:20] = R0;
308 //SA
309 InstructionMemory[MemLoc][19:15] = R10;
310 //Offset
311 InstructionMemory[MemLoc][14:0] = 6;
312 MemLoc = MemLoc + 1;
313 //////////////////////////////////////
314 //NOP
315 //Opcode
316 InstructionMemory[MemLoc][31:25] = NOP;
317 MemLoc = MemLoc + 1;
318 //////////////////////////////////////
319 //NOP
320 //Opcode
321 InstructionMemory[MemLoc][31:25] = NOP;
322 MemLoc = MemLoc + 1;
323 //////////////////////////////////////
324 //MOV R3, R1
325 //Opcode
326 InstructionMemory[MemLoc][31:25] = MOV;
327 //DR
328 InstructionMemory[MemLoc][24:20] = R3;
329 //SA
330 InstructionMemory[MemLoc][19:15] = R1;
331 //IM
332 InstructionMemory[MemLoc][14:0] = 0;
333 MemLoc = MemLoc + 1;
334
335 //Start Program
336 IR = InstructionMemory[0];
337 end
338
339 //Read
340 always @ (PCin)
341 begin
342     begin
343         IR = InstructionMemory[PCin];
344     end
345
346 end
347
348 endmodule
349
```

```
1      `timescale 1ns / 1ps
2      ///////////////////////////////////////////////////
3      // Company:
4      // Engineer:
5      //
6      // Create Date:      16:33:39 11/15/2009
7      // Design Name:
8      // Module Name:      InstructionDecoder
9      // Project Name:
10     // Target Devices:
11     // Tool versions:
12     // Description:
13     //
14     // Dependencies:
15     //
16     // Revision:
17     // Revision 0.01 - File Created
18     // Additional Comments:
19     //
20     ///////////////////////////////////
21     module InstructionDecoder(
22         input [31:0] IRin,
23         input Reset,
24         output RWout,
25         output [4:0] DAout,
26         output [1:0] MDout,
27         output [1:0] BSout,
28         output PSout,
29         output MWout,
30         output [4:0] FSout,
31         output MAout,
32         output MBout,
33         output [4:0] AAout,
34         output [4:0] BAout ,
35         output CSout
36     );
37
38     `include "Instructions.txt"
39
40     reg RWreg;
41     assign RWout = RWreg;
42     reg [4:0] DArege;
43     assign DAout = DArege;
44     reg [1:0] MDreg;
45     assign MDout = MDreg;
46     reg [1:0] BSreg;
47     assign BSout = BSreg;
48     reg PSreg;
49     assign PSout = PSreg;
50     reg MWreg;
51     assign MWout = MWreg;
52     reg [4:0] FSreg;
53     assign FSout = FSreg;
54     reg MAreg;
55     assign MAout = MAreg;
56     reg MBreg;
57     assign MBout = MBreg;
58     reg [4:0] AAreg;
59     assign AAout = AAreg;
60     reg [4:0] BAreg;
61     assign BAout = BAreg;
```

```

62     reg CSreg;
63     assign CSout = CSreg;
64
65     //Clear outputs
66     initial
67     begin
68         RWreg = 0;
69         DArege = 0;
70         MDreg = 0;
71         BSreg = 0;
72         PSreg = 0;
73         MWreg = 0;
74         FSreg = 0;
75     end
76
77     //Hardware Reset Triggered
78     always @ (posedge Reset)
79     begin
80         RWreg = 0;
81         DArege = 0;
82         MDreg = 0;
83         BSreg = 0;
84         PSreg = 0;
85         MWreg = 0;
86         FSreg = 0;
87     end
88
89     //At each Instruction Change
90     always @ (IRin)
91     begin
92         case (IRin[31:25])
93             ///////////////////////////////////
94             NOP:
95                 begin
96                     RWreg = 0;
97                     //MDreg = 2'bxx;
98                     BSreg = 2'b00;
99                     //PSreg = 1'bx;
100                    MWreg = 0;
101                    //FSreg = 5'bxxxxxx;
102                    //MBreg = 1'bx;
103                    //MAreg = 1'bx;
104                    //CSreg = 1'bx;
105
106                    DArege = IRin[24:20];
107                    AArege = IRin[19:15];
108                    BArege = IRin[14:10];
109
110
111                end
112            ///////////////////////////////////
113            ADD:
114                begin
115                    RWreg = 1;
116                    MDreg = 2'b00;
117                    BSreg = 2'b00;
118                    //PSreg = 1'bx;
119                    MWreg = 0;
120                    FSreg = 5'b000010;
121                    MBreg = 0;
122                    MAreg = 0;

```

```

123          //CSreg = 1'bx;
124
125          DAre = IRin[24:20];
126          AAre = IRin[19:15];
127          BAre = IRin[14:10];
128      end
129      //////////////////////////////////////
130  SUB:
131      begin
132          RWreg = 1;
133          MDreg = 2'b00;
134          BSreg = 2'b00;
135          //PSreg = 1'bx;
136          MWreg = 0;
137          FSreg = 5'b00100;
138          MBreg = 0;
139          MAreg = 0;
140          //CSreg = 1'bx;
141
142          DAre = IRin[24:20];
143          AAre = IRin[19:15];
144          BAre = IRin[14:10];
145      end
146      //////////////////////////////////////
147  SLT:
148      begin
149          RWreg = 1;
150          MDreg = 2'b10;
151          BSreg = 2'b00;
152          //PSreg = 1'bx;
153          MWreg = 0;
154          FSreg = 5'b00101;
155          MBreg = 0;
156          MAreg = 0;
157          //CSreg = 1'bx;
158
159          DAre = IRin[24:20];
160          AAre = IRin[19:15];
161          BAre = IRin[14:10];
162      end
163      //////////////////////////////////////
164  AND:
165      begin
166          RWreg = 1;
167          MDreg = 2'b00;
168          BSreg = 2'b00;
169          //PSreg = 1'bx;
170          MWreg = 0;
171          FSreg = 5'b01000;
172          MBreg = 0;
173          MAreg = 0;
174          //CSreg = 1'bx;
175
176          DAre = IRin[24:20];
177          AAre = IRin[19:15];
178          BAre = IRin[14:10];
179
180          DAre = IRin[24:20];
181          AAre = IRin[19:15];
182          BAre = IRin[14:10];
183      end

```

```

184          //////////////////////////////////////////
185      OR:
186          begin
187              RWreg = 1;
188              MDreg = 2'b00;
189              BSreg = 2'b00;
190              //PSreg = 1'bx;
191              MWreg = 0;
192              FSreg = 5'b01010;
193              MBreg = 0;
194              MAreg = 0;
195              //CSreg = 1'bx;
196
197              DAreg = IRin[24:20];
198              AAreg = IRin[19:15];
199              BAreg = IRin[14:10];
200          end
201          //////////////////////////////////////////
202      XOR:
203          begin
204              RWreg = 1;
205              MDreg = 2'b00;
206              BSreg = 2'b00;
207              //PSreg = 1'bx;
208              MWreg = 0;
209              FSreg = 5'b01100;
210              MBreg = 0;
211              MAreg = 0;
212              //CSreg = 1'bx;
213
214              DAreg = IRin[24:20];
215              AAreg = IRin[19:15];
216              BAreg = IRin[14:10];
217          end
218          //////////////////////////////////////////
219      ST:
220          begin
221              RWreg = 0;
222              MDreg = 2'b00;
223              BSreg = 2'b00;
224              //PSreg = 1'bx;
225              MWreg = 1;
226              //FSreg = 5'bxxxxxx;
227              MBreg = 0;
228              MAreg = 0;
229              //CSreg = 1'bx;
230
231              DAreg = IRin[24:20];
232              AAreg = IRin[19:15];
233              BAreg = IRin[14:10];
234          end
235          //////////////////////////////////////////
236      LD:
237          begin
238              RWreg = 1;
239              MDreg = 2'b01;
240              BSreg = 2'b00;
241              //PSreg = 1'bx;
242              MWreg = 0;
243              //FSreg = 5'bxxxxxx;
244              //MBreg = 1'bx;

```

```

245         MReg = 0;
246         //CSreg = 1'bx;
247
248         DReg = IRin[24:20];
249         AReg = IRin[19:15];
250         BReg = IRin[14:10];
251     end
252     //////////////////////////////////////
253     ADI:
254     begin
255         RReg = 1;
256         MReg = 2'b00;
257         BReg = 2'b00;
258         //PSreg = 1'bx;
259         WReg = 0;
260         FReg = 5'b00010;
261         MReg = 1;
262         MReg = 0;
263         CSreg = 1;
264
265         DReg = IRin[24:20];
266         AReg = IRin[19:15];
267
268     end
269     //////////////////////////////////////
270     SBI:
271     begin
272         RReg = 1;
273         MReg = 2'b00;
274         BReg = 2'b00;
275         //PSreg = 1'bx;
276         WReg = 0;
277         FReg = 5'b00100;
278         MReg = 1;
279         MReg = 0;
280         CSreg = 1;
281
282         DReg = IRin[24:20];
283         AReg = IRin[19:15];
284     end
285     //////////////////////////////////////
286     NOT:
287     begin
288         RReg = 1;
289         MReg = 2'b00;
290         BReg = 2'b00;
291         //PSreg = 1'bx;
292         WReg = 0;
293         FReg = 5'b01110;
294         //MReg = 1'bx;
295         MReg = 0;
296         //CSreg = 1'bx;
297
298         DReg = IRin[24:20];
299         AReg = IRin[19:15];
300     end
301     //////////////////////////////////////
302     ANI:
303     begin
304         RReg = 1;
305         MReg = 2'b00;

```

```

306         BSreg = 2'b00;
307         //PSreg = 1'bx;
308         MWreg = 0;
309         FSreg = 5'b01000;
310         MBreg = 1;
311         MAreg = 0;
312         CSreg = 0;
313
314         DAreg = IRin[24:20];
315         AAreg = IRin[19:15];
316     end
317     //////////////////////////////////
318 ORI:
319     begin
320         RWreg = 1;
321         MDreg = 2'b00;
322         BSreg = 2'b00;
323         //PSreg = 1'bx;
324         MWreg = 0;
325         FSreg = 5'b01010;
326         MBreg = 1;
327         MAreg = 0;
328         CSreg = 0;
329
330         DAreg = IRin[24:20];
331         AAreg = IRin[19:15];
332     end
333     //////////////////////////////////
334 XRI:
335     begin
336         RWreg = 1;
337         MDreg = 2'b00;
338         BSreg = 2'b00;
339         //PSreg = 1'bx;
340         MWreg = 0;
341         FSreg = 5'b01100;
342         MBreg = 1;
343         MAreg = 0;
344         CSreg = 0;
345
346         DAreg = IRin[24:20];
347         AAreg = IRin[19:15];
348     end
349     //////////////////////////////////
350 AIU:
351     begin
352         RWreg = 1;
353         MDreg = 2'b00;
354         BSreg = 2'b00;
355         //PSreg = 1'bx;
356         MWreg = 0;
357         FSreg = 5'b00010;
358         MBreg = 1;
359         MAreg = 0;
360         CSreg = 0;
361
362         DAreg = IRin[24:20];
363         AAreg = IRin[19:15];
364     end
365     //////////////////////////////////
366 SIU:

```

```

367         begin
368             RWreg = 1;
369             MDreg = 2'b00;
370             BSreg = 2'b00;
371             //PSreg = 1'bx;
372             MWreg = 0;
373             FSreg = 5'b00101;
374             MBreg = 1;
375             MAreg = 0;
376             CSreg = 0;
377
378             DAre = IRin[24:20];
379             AAre = IRin[19:15];
380         end
381         //////////////////////////////////////
382     MOV:
383         begin
384             RWreg = 1;
385             MDreg = 2'b00;
386             BSreg = 2'b00;
387             //PSreg = 1'bx;
388             MWreg = 0;
389             FSreg = 5'b00000;
390             //MBreg = 1'bx;
391             MAreg = 0;
392             //CSreg = 1'bx;
393
394             DAre = IRin[24:20];
395             AAre = IRin[19:15];
396         end
397         //////////////////////////////////////
398     LSL:
399         begin
400             RWreg = 1;
401             MDreg = 2'b00;
402             BSreg = 2'b00;
403             //PSreg = 1'bx;
404             MWreg = 0;
405             FSreg = 5'b10100;
406             //MBreg = 1'bx;
407             MAreg = 0;
408             //CSreg = 1'bx;
409
410             DAre = IRin[24:20];
411             AAre = IRin[19:15];
412         end
413         //////////////////////////////////////
414     LSR:
415         begin
416             RWreg = 1;
417             MDreg = 2'b00;
418             BSreg = 2'b00;
419             //PSreg = 1'bx;
420             MWreg = 0;
421             FSreg = 5'b11000;
422             //MBreg = 1'bx;
423             MAreg = 0;
424             //CSreg = 1'bx;
425
426             DAre = IRin[24:20];
427             AAre = IRin[19:15];

```



```
428         end
429         //////////////////////////////////////
430     JMR:
431         begin
432             RWreg = 0;
433             //MDreg = 2'bxx;
434             BSreg = 2'b10;
435             //PSreg = 1'bx;
436             MWreg = 0;
437             FSreg = 5'b00000;
438             //MBreg = 1'bx;
439             //MAreg = 1'bx;
440             //CSreg = 1'bx;
441
442             DAreg = IRin[24:20];
443             AAreg = IRin[19:15];
444         end
445         //////////////////////////////////////
446     BZ:
447         begin
448             RWreg = 0;
449             //MDreg = 2'bxx;
450             BSreg = 2'b01;
451             PSreg = 0;
452             MWreg = 0;
453             FSreg = 5'b00000;
454             MBreg = 1;
455             MAreg = 0;
456             CSreg = 1;
457
458             DAreg = IRin[24:20];
459             AAreg = IRin[19:15];
460         end
461         //////////////////////////////////////
462     BNZ:
463         begin
464             RWreg = 0;
465             //MDreg = 2'bxx;
466             BSreg = 2'b01;
467             PSreg = 1;
468             MWreg = 0;
469             FSreg = 5'b00000;
470             MBreg = 1;
471             MAreg = 0;
472             CSreg = 1;
473
474             DAreg = IRin[24:20];
475             AAreg = IRin[19:15];
476         end
477         //////////////////////////////////////
478     JMP:
479         begin
480             RWreg = 0;
481             //MDreg = 2'bxx;
482             BSreg = 2'b11;
483             //PSreg = 1'bx;
484             MWreg = 0;
485             //FSreg = 5'bxxxxxx;
486             MBreg = 1;
487             //MAreg = 1'bx;
488             CSreg = 1;
```

```
489
490         DAreg = IRin[24:20];
491         AAreg = IRin[19:15];
492     end
493     //////////////////////////////////////////
494     JML:
495     begin
496         RWreg = 1;
497         MDreg = 2'b00;
498         BSreg = 2'b11;
499         //PSreg = 1'bx;
500         MWreg = 0;
501         FSreg = 5'b00111;
502         MBreg = 1;
503         MAreg = 1;
504         CSreg = 1;
505
506         DAreg = IRin[24:20];
507         AAreg = IRin[19:15];
508     end
509 endcase
510 end
511
512 endmodule
513
```

```
1      `timescale 1ns / 1ps
2      ///////////////////////////////////////////////////
3      // Company:
4      // Engineer:
5      //
6      // Create Date:      16:05:37 11/15/2009
7      // Design Name:
8      // Module Name:      FunctionUnit
9      // Project Name:
10     // Target Devices:
11     // Tool versions:
12     // Description:
13     //
14     // Dependencies:
15     //
16     // Revision:
17     // Revision 0.01 - File Created
18     // Additional Comments:
19     //
20     ///////////////////////////////////
21
22
23     module FunctionUnit(
24         input signed [31:0] Ains,
25         input signed [31:0] Bins,
26         input [31:0] Ainu,
27         input [31:0] Binu,
28         input [4:0] SHin,
29         input [4:0] FSin,
30         input Reset,
31         output Zout,
32         output Vout,
33         output Nout,
34         output Cout,
35         output [31:0] Fout
36     );
37
38     `include "FSCodes.txt"
39
40     reg Zreg, Vreg, Nreg, Creg;
41     reg [32:0] Freg;
42     reg signed [32:0] Fregs;
43     assign Fout = Freg[31:0];
44
45     reg [31:0] ShiftBuf;
46     reg i;
47
48     assign Zout = Zreg;
49     assign Vout = Vreg;
50     assign Nout = Nreg;
51     assign Cout = Creg;
52
53     reg k;
54
55
56     initial
57         begin
58             ShiftBuf = 0;
59             Zreg = 0;
60             Vreg = 0;
61             Nreg = 0;
```

```

62         Creg = 0;
63         Freg = 0;
64
65     end
66
67     always @ (negedge Reset)
68     begin
69         ShiftBuf = 0;
70         Zreg = 0;
71         Vreg = 0;
72         Nreg = 0;
73         Creg = 0;
74         Freg = 0;
75     end
76
77     always @ *
78     begin
79         case (FSin)
80             //////////////////////////////////
81             ADD:
82                 begin
83                     Freg = Ains + Bins;
84                     Fregs = Freg;
85
86                     Zreg = Freg==0;
87                     Nreg = Freg[31];
88
89                     if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
90                     else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
91                     else Vreg = 0;
92
93                     Creg = Freg[32];
94                 end
95             //////////////////////////////////
96             SUB:
97                 begin
98                     Freg = Ains - Bins;
99                     Fregs = Freg;
100                    Zreg = Freg==0;
101                    Nreg = Freg[31];
102                    if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
103                    else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
104                    else Vreg = 0;
105                    Creg = Freg[32];
106                end
107            //////////////////////////////////
108            SLT:
109                begin
110                    if(Ains < Bins)Freg = 1;
111                    else Freg = 0;
112                    Fregs = Freg;
113                    Zreg = Freg==0;
114                    Nreg = Freg[31];
115                    if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
116                    else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
117                    else Vreg = 0;
118                    Creg = Freg[32];
119                end
120            //////////////////////////////////
121            AND:
122                begin

```

```

123         Freg = Ains & Bins;
124         Fregs = Freg;
125         Zreg = Freg==0;
126         Nreg = Freg[31];
127         if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
128         else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
129         else Vreg = 0;
130         Creg = Freg[32];
131
132     end
133     //////////////////////////////////////////
134 OR:
135     begin
136
137         Freg = Ains | Bins;
138         Fregs = Freg;
139         Zreg = Freg==0;
140         Nreg = Freg[31];
141         if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
142         else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
143         else Vreg = 0;
144         Creg = Freg[32];
145
146     end
147     //////////////////////////////////////////
148 XOR:
149     begin
150
151         Freg = Ains ^ Bins;
152         Fregs = Freg;
153         Zreg = Freg==0;
154         Nreg = Freg[31];
155         if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
156         else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
157         else Vreg = 0;
158         Creg = Freg[32];
159
160     end
161     //////////////////////////////////////////
162 NOT:
163     begin
164
165         Freg = ~Ains;
166         Fregs = Freg;
167         Zreg = Freg==0;
168         Nreg = Freg[31];
169         if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
170         else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
171         else Vreg = 0;
172         Creg = Freg[32];
173
174     end
175     //////////////////////////////////////////
176 AIU:
177     begin
178
179         Freg = Ainu + Binu;
180
181         Zreg = Freg==0;
182         Nreg = Freg[31];
183

```

```

184             Vreg = Freg[32];
185
186             Creg = Freg[32];
187
188         end
189         //////////////////////////////////
190     SIU:
191         begin
192
193             Freg = Ainu - Binu;
194             Zreg = Freg==0;
195             Nreg = Freg[31];
196             Vreg = Freg[32];
197             Creg = Freg[32];
198
199         end
200         //////////////////////////////////
201     MOV:
202         begin
203
204             Freg = Ainu;
205             Zreg = Freg==0;
206             Nreg = Freg[31];
207             Vreg = Freg[32];
208             Creg = Freg[32];
209
210         end
211         //////////////////////////////////
212     LSL:
213         begin
214
215             Freg = Ains << SHin;
216             Fregs = Freg;
217             Zreg = Freg==0;
218             Nreg = Freg[31];
219             if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
220             else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
221             else Vreg = 0;
222             Creg = Freg[32];
223         end
224         //////////////////////////////////
225     LSR:
226         begin
227
228             Freg = Ains >> SHin;
229             Fregs = Freg;
230             Zreg = Freg==0;
231             Nreg = Freg[31];
232             if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
233             else if (Fregs < 32'b10000000000000000000000000000000)Vreg=1;
234             else Vreg = 0;
235             Creg = Freg[32];
236
237         end
238
239
240     JML:
241         begin
242             Freg = Ains;
243             Zreg = Freg==0;
244             Nreg = Freg[31];

```

```
245         if(Fregs > 32'b01111111111111111111111111111111)Vreg=1;
246         else if (Fregs < 32'b100000000000000000000000000000)Vreg=1;
247         else Vreg = 0;
248         Creg = Freg[32];
249
250     end
251
252 endcase
253
254 end
255
256 endmodule
257
```

```
1      `timescale 1ns / 1ps
2      ///////////////////////////////////////////////////////////////////
3      // Company:
4      // Engineer:
5      //
6      // Create Date:      16:12:33 11/15/2009
7      // Design Name:
8      // Module Name:      RegisterFile
9      // Project Name:
10     // Target Devices:
11     // Tool versions:
12     // Description:
13     //
14     // Dependencies:
15     //
16     // Revision:
17     // Revision 0.01 - File Created
18     // Additional Comments:
19     //
20     ///////////////////////////////////////////////////////////////////
21     module RegisterFile(
22         //Read ports
23         input [4:0] AAin,
24         input [4:0] BAin,
25         output [31:0] Aout,
26         output [31:0] Bout,
27
28         //Write Ports
29         input [31:0] Din,
30         input RWin,
31         input [4:0] DAin,
32
33         //Control Signals
34         input Clock,
35         input Reset
36
37     );
38
39     `include "Instructions.txt"
40     `include "Registers.txt"
41
42     reg [31:0] Areg;
43     reg [31:0] Breg;
44
45     assign Aout = Areg;
46     assign Bout = Breg;
47
48
49     reg [31:0] RegisterMemory [31:0];    // 32 Registers of 32 bits
50
51     //Loop counter
52     reg k;
53
54     always @ (posedge Reset)
55     begin
56         Areg = 0;
57         Breg = 0;
58
59     end
60
61     initial
```



```
62     begin
63         Areg = 0;
64         Breg = 0;
65
66         RegisterMemory[R1] = 3;
67         RegisterMemory[R2] = R2;
68         RegisterMemory[R3] = R3;
69         RegisterMemory[R4] = R4;
70         RegisterMemory[R5] = R5;
71
72
73     end
74
75
76
77     //Read
78     always @ (negedge Clock)
79     begin
80         if(AAin == 0)
81             begin
82                 Areg = 0;
83             end
84         else
85             begin
86                 Areg = RegisterMemory[AAin];
87             end
88         if(BAin == 0)
89             begin
90                 Breg = 0;
91             end
92         else
93             begin
94                 Breg = RegisterMemory[BAin];
95             end
96     end
97
98     //Write
99     always @ (posedge Clock)
100    begin
101        //If write is enabled
102        if(RWin == 1)
103            begin
104                //If you are sending data to R0, throw data away
105                if(DAin == 0)
106                    begin
107                        RegisterMemory[0] = 0;
108                    end
109                //Else if there is a valid Register address
110                else
111                    begin
112                        //Store the data in Register[DAin]
113                        RegisterMemory[DAin] = Din;
114                    end
115            end
116    end
117 endmodule
118
```

```
1      `timescale 1ns / 1ps
2      ///////////////////////////////////////////////////////////////////
3      // Company:
4      // Engineer:
5      //
6      // Create Date:      12:47:21 11/18/2009
7      // Design Name:
8      // Module Name:      PC
9      // Project Name:
10     // Target Devices:
11     // Tool versions:
12     // Description:
13     //
14     // Dependencies:
15     //
16     // Revision:
17     // Revision 0.01 - File Created
18     // Additional Comments:
19     //
20     ///////////////////////////////////////////////////////////////////
21     module PC(
22         input  Clock,
23         input  Reset,
24         input  [31:0] Inpin,
25         output [31:0] Outpin
26     );
27
28     reg [31:0] Buffer;
29     assign Outpin = Buffer;
30
31     initial
32     begin
33         Buffer = 0;
34     end
35
36     always @ (Reset)
37     begin
38         Buffer = 0;
39     end
40
41     always @ (posedge Clock)
42     begin
43         Buffer = Inpin;
44     end
45
46
47     endmodule
48
```

```
1      `timescale 1ns / 1ps
2      //////////////////////////////////////////////////
3      // Company:
4      // Engineer:
5      //
6      // Create Date:      12:33:06 11/18/2009
7      // Design Name:
8      // Module Name:      MuxA
9      // Project Name:
10     // Target Devices:
11     // Tool versions:
12     // Description:
13     //
14     // Dependencies:
15     //
16     // Revision:
17     // Revision 0.01 - File Created
18     // Additional Comments:
19     //
20     //////////////////////////////////
21     module MuxA(
22         input [31:0] Zero,
23         input [31:0] One,
24         output [31:0] BusA,
25         input MA
26     );
27
28         reg [31:0] Selected;
29         assign BusA = Selected;
30
31     initial
32     begin
33         Selected = 0;
34     end
35
36     always @ *
37     begin
38         if(MA == 0)
39             begin
40                 Selected = Zero;
41             end
42         else if(MA == 1)
43             begin
44                 Selected = One;
45             end
46         end
47     end
48 endmodule
49
```



