

ELEC 374 – Digital Systems Engineering

Phase 3 Report

Group 18

Divaydeep Singh	20143859
------------------------	-----------------

Lucas Austin	20061953
---------------------	-----------------

Code

Ctrl_unit.v

```
1 `timescale 1ns/10ps
2 module ctrl_unit (
3     output reg Gra, Grb, Grc, Rin, Rout, MDRin, MDRout, BAout, Cout, ZLowout, PCin, IRin,
4     output reg HIout, LOout, InPortout, OPin, LOin, ZHighout, HIin, Yin, Zin, PCout,
5     output reg IncPC, MARin, read, wren, clr, conf_in, Run,
6     output reg [3:0] ALUselect,
7     input [31:0] IR,
8     input clk, reset, stp,
9     output reg conf_out, R15ctrl
10 );
11
12 parameter
13     reset_state = 'b0000000, fetch0 = 'b0000001, fetch1 = 'b0000010, fetch2 = 'b0000011, ld = 'b0000100, ld2 = 'b0000101, ld3 = 'b0000110, ld4 = 'b0000111, ld5 = 'b0001000,
14     ldi = 'b0001001, ldi2 = 'b0001010, ldi3 = 'b0001011, st = 'b0001100, st2 = 'b0001101, st3 = 'b0001110, st4 = 'b0001111, st5 = 'b0010000, addi = 'b0010001,
15     addi2 = 'b0010010, addi3 = 'b0010011, ori = 'b0010100, ori2 = 'b0010101, ori3 = 'b0010110, andi = 'b0010111, andi2 = 'b0011000, andi3 = 'b0011001,
16     br = 'b0011010, br2 = 'b0011011, br3 = 'b0011100, br4 = 'b0011101, jr = 'b0011110, jal = 'b0011111, jal2 = 'b0100000, mghi = 'b0100001, mghi2 = 'b0100010,
17     in = 'b0100011, out = 'b0100100, add = 'b0100101, add2 = 'b0100110, add3 = 'b0100111, add4 = 'b0101000, sub = 'b0101001, sub2 = 'b0101010, sub3 = 'b0101011,
18     sub4 = 'b0101100, shr = 'b0101101, shr2 = 'b0101110, shr3 = 'b0101111, shr4 = 'b0110000, shr5 = 'b0110001, shl = 'b0110010, shl2 = 'b0110011, shl3 = 'b0110100,
19     shl4 = 'b0110101, shl5 = 'b0110110, ror = 'b0110111, ror2 = 'b0111000, ror3 = 'b0111001, ror4 = 'b0111010, ror5 = 'b0111011, ror6 = 'b0111100, rol = 'b0111101,
20     rol2 = 'b0111110, rol3 = 'b0111111, rol4 = 'b1000001, rol5 = 'b1000010, rol6 = 'b1000011, orl = 'b1000100, or2 = 'b1000101, or3 = 'b1000110, andl = 'b1000111,
21     and2 = 'b1001000, and3 = 'b1001001, mul = 'b1001010, mul2 = 'b1001011, mul3 = 'b1001100, mul4 = 'b1001101, mul5 = 'b1001110, div = 'b1001111, div2 = 'b1010000,
22     div3 = 'b1010001, div4 = 'b1010010, neg = 'b1010011, neg2 = 'b1010100, neg3 = 'b1010101, not1 = 'b1010110, not2 = 'b1010111, not3 = 'b1011000,
23     nop = 'b1011001, nop2 = 'b1011010, halt = 'b1011011, halt2 = 'b1011100;
24
25 reg[6:0] Present_state = reset_state;
26
27 always @(posedge clk, posedge reset)
28 begin
29     if(reset) #40 Present_state = reset_state;
30     else if(stp) #40 Present_state = halt;
31     else case(Present_state)
32         reset_state : #40 Present_state = fetch0;
33         fetch0 : #40 Present_state = fetch1;
34         fetch1 : #40 Present_state = fetch2;
35         fetch2 : #40
36
37     begin
38         case(IR[31:27])
39             5'b000000 : Present_state = ld;
40             5'b000001 : Present_state = ldi;
41             5'b000010 : Present_state = st;
42             5'b000011 : Present_state = add;
43             5'b000100 : Present_state = sub;
44             5'b000101 : Present_state = shr;
45             5'b000110 : Present_state = shl;
46             5'b000111 : Present_state = ror;
47             5'b001000 : Present_state = rol;
48             5'b001001 : Present_state = andl;
49             5'b001010 : Present_state = orl;
50             5'b001011 : Present_state = addi;
51             5'b001100 : Present_state = andi;
52             5'b001101 : Present_state = ori;
53             5'b001110 : Present_state = mul;
54             5'b001111 : Present_state = div;
55             5'b100000 : Present_state = neg;
56             5'b100001 : Present_state = not1;
57             5'b100010 : Present_state = br;
58             5'b100011 : Present_state = jr;
59             5'b100100 : Present_state = jal;
60             5'b100101 : Present_state = in;
61             5'b100110 : Present_state = out;
62             5'b100111 : Present_state = mghi;
63             5'b101000 : Present_state = mghi2;
64             5'b101001 : Present_state = mghi3;
65             5'b101010 : Present_state = halt;
66         endcase
67     end
68
69     ld : #40 Present_state = ld2;
70     ld2 : #40 Present_state = ld3;
71     ld3 : #40 Present_state = ld4;
72     ld4 : #40 Present_state = ld5;
73     ld5 : #40 Present_state = fetch0;
74
75     //ldi
76     ldi : #40 Present_state = ldi2;
77     ldi2 : #40 Present_state = ldi3;
78     ldi3 : #40 Present_state = fetch0;
79
80     //st
81     st : #40 Present_state = st2;
82     st2 : #40 Present_state = st3;
83     st3 : #40 Present_state = st4;
84     st4 : #40 Present_state = st5;
85     st5 : #40 Present_state = fetch0;
86
87     //addi
88     addi : #40 Present_state = addi2;
89     addi2 : #40 Present_state = addi3;
90     addi3 : #40 Present_state = fetch0;
91
92     //ori
93     ori : #40 Present_state = ori2;
94     ori2 : #40 Present_state = ori3;
95     ori3 : #40 Present_state = fetch0;
96
97     //andi
98     andi : #40 Present_state = andi2;
99     andi2 : #40 Present_state = andi3;
100     andi3 : #40 Present_state = fetch0;
101
```

```

102 //br
103 br : #40 Present_state = br2;
104 br2 : #40 Present_state = br3;
105 br3 : #40 Present_state = br4;
106 br4 : #40 Present_state = fetch0;
107
108 //jal
109 jal : #40 Present_state = jal2;
110 jal2 : #40 Present_state = fetch0;
111
112 //add
113 add : #40 Present_state = add2;
114 add2 : #40 Present_state = add3;
115 add3 : #40 Present_state = add4;
116 add4 : #40 Present_state = fetch0;
117
118 //sub
119 sub : #40 Present_state = sub2;
120 sub2 : #40 Present_state = sub3;
121 sub3 : #40 Present_state = sub4;
122 sub4 : #40 Present_state = fetch0;
123
124 //shr
125 shr : #40 Present_state=shr2;
126 shr2 : #40 Present_state=shr3;
127 shr3 : #40 Present_state=shr4;
128 shr4 : #40 Present_state=shr5;
129 shr5 : #40 Present_state=fetch0;
130
131 //shl
132 shl : #40 Present_state=shl2;
133 shl2 : #40 Present_state=shl3;
134 shl3 : #40 Present_state=shl4;
135 shl4 : #40 Present_state=shl5;
136 shl5 : #40 Present_state=fetch0;
137
138 //ror
139 ror : #40 Present_state=ror2;
140 ror2 : #40 Present_state=ror3;
141 ror3 : #40 Present_state=ror4;
142 ror4 : #40 Present_state=ror5;
143 ror5 : #40 Present_state=fetch0;
144
145 //rol
146 rol : #40 Present_state=rol2;
147 rol2 : #40 Present_state=rol3;
148 rol3 : #40 Present_state=rol4;
149 rol4 : #40 Present_state=rol5;
150 rol5 : #40 Present_state=rol6;
151 rol6 : #40 Present_state=fetch0;
152
153 //and
154 and1 : #40 Present_state=and2;
155 and2 : #40 Present_state=and3;
156 and3 : #40 Present_state=fetch0;
157
158 //or
159 or1 : #40 Present_state=or2;
160 or2 : #40 Present_state=or3;
161 or3 : #40 Present_state=fetch0;
162
163 //mul
164 mul : #40 Present_state=mul2;
165 mul2 : #40 Present_state=mul3;
166 mul3 : #40 Present_state=mul4;
167 mul4 : #40 Present_state=mul5;
168 mul5 : #40 Present_state=fetch0;
169
170 //div
171 div : #40 Present_state=div2;
172 div2 : #40 Present_state=div3;
173 div3 : #40 Present_state=div4;
174 div4 : #40 Present_state=fetch0;
175
176 //neg
177 neg : #40 Present_state=neg2;
178 neg2 : #40 Present_state=fetch0;
179
180 //not
181 not1 : #40 Present_state=not2;
182 not2 : #40 Present_state=fetch0;
183
184 //nop
185 nop : #40 Present_state=fetch0;
186
187 //halt
188 halt : #40 Present_state=halt;
189
190 //mfhi
191 mfhi : #40 Present_state=fetch0;
192
193 //mflo
194 mflo : #40 Present_state=fetch0;
195
196 //jr
197 jr : #40 Present_state=fetch0;
198 endcase
199 end

```

```

201 always@(Present_state)
202 begin
203   case(Present_state)
204     reset_state:begin
205       Gra <= 0; Grb <= 0; Grc <= 0; Rin <= 0; Rout <= 0; MDRin <= 0; MDRout <= 0; BAout <= 0; Cout <= 0; ZLowout <= 0;
206       PCin <= 0; IRin <= 0; HIout <= 0; LOout <= 0; InPortout <= 0; OPin <= 0; LOin <= 0; ZHighout <= 0; HIin <= 0;
207       Yin <= 0; Zin <= 0; PCout <= 0; IncPC <= 0; MARin <= 0; R15ctrl <= 0;
208       read <= 0; wren <= 0; clr <= 0; ALUselect <= 4'b0000;
209       conff_in <= 0; conff_out<=0; Run <=1;
210     end
211     fetch0: begin
212       Gra <= 0; Grb <= 0; Grc <= 0; Rin <= 0; Rout <= 0; MDRin <= 0; MDRout <= 0; BAout <= 0; Cout <= 0; ZLowout <= 0;
213       PCin <= 0; IRin <= 0; HIout <= 0; LOout <= 0; InPortout <= 0; OPin <= 0; LOin <= 0; ZHighout <= 0; HIin <= 0;
214       Yin <= 0; IncPC <= 0; read <= 0; wren <= 0; clr <= 0; conff_in <= 0; conff_out <= 0; R15ctrl <= 0;
215
216       PCout <= 1; MARin <= 1; ALUselect <= 4'b1001; IncPC <= 1; Zin <= 1;
217     end
218     fetch1: begin
219       MARin <= 0; PCout <= 0; ALUselect <= 4'b0000; IncPC <= 0; Zin <= 0;
220       ZLowout <= 1; PCin <= 1; read <= 1; MDRin <= 1;
221     end
222     fetch2: begin
223       ZLowout <= 0; MDRin <= 0; PCin <= 0; read <= 0;
224       MDRout <= 1; IRin <= 1;
225     end
226     //ld
227     ld:
228       begin
229         IRin <= 0;
230         MDRout <= 0;
231
232         Yin <= 1;
233         Grb <= 1;
234         BAout <= 1;
235       end
236
237     ld2:
238       begin
239         Grb <= 0;
240         Yin <= 0;
241         BAout <= 0;
242
243
244         Cout <= 1;
245         Zin <= 1;
246
247         #20
248         ALUselect <= 4'b0001;
249       end
250
251     ld3:
252       begin
253         Cout <= 0;
254         Zin <= 0;
255
256
257         ZLowout <= 1;
258         MARin <= 1;
259       end
260
261     ld4: begin
262       ZLowout <= 0;
263       MARin <= 0;
264
265       MDRin <= 1;
266       read <= 1;
267     end
268
269     ld5: begin
270       MDRin <= 0;
271       read <= 0;
272
273       MDRout <= 1;
274       Gra <= 1;
275       Rin <= 1;
276     end
277
278     //ldi
279     ldi: begin
280       MDRout <= 0; IRin <= 0;
281       Grb <= 1; BAout<=1; Yin <=1;
282     end
283     ldi2: begin
284       Grb <=0; BAout<=0; Yin <=0;
285       Cout <=1; Zin<=1;
286       #20
287       ALUselect <=4'b0001;
288     end
289     ldi3: begin
290       Cout <=0; Zin<=0;
291       Gra<=1; ZLowout <=1; Rin<=1;
292     end
293
294     //st
295     st: begin
296       MDRout <= 0; IRin <= 0;
297       Grb <=1; BAout<=1; Yin <=1;
298     end
299     st2: begin
300       Grb <=0; BAout<=0; Yin <=0;
301       Cout <=1; Zin<=1; ALUselect <=4'b0001;
302     end
303     st3: begin
304       Cout <=0; Zin<=0;
305       ZLowout <=1; MARin<=1;
306     end
307     st4: begin
308       ZLowout <=0; MARin<=0;
309       Rout <= 1; Gra <=1;
310       MDRin<=1;
311     end
312     st5: begin
313       MDRin<=0;
314       Rout <= 0; Gra <=0;
315       MDRout<=1; wren<=1;
316     end
317

```

```

318 //addi
319 addi: begin
320   MDRout <= 0; IRin <= 0;
321   Grb <=1; Rout<=1; Yin <=1;
322 end
323 addi2: begin
324   Grb <=0; Rout<=0; Yin <=0;
325   Cout <=1; Zin<=1;
326   #20
327   ALUselect <= 4'b0001;
328 end
329 addi3: begin
330   Cout <=0; Zin<=0;
331   ZLowout <=1; Gra<=1; Rin<=1;
332 end
333
334 //ori
335 ori: begin
336   MDRout <= 0; IRin <= 0;
337   Grb <=1; Rout<=1; Yin <=1;
338 end
339 ori2: begin
340   Grb <=0; Rout<=0; Yin <=0;
341   Cout <=1; Zin<=1;
342   #20
343   ALUselect <= 4'b0111;
344 end
345 ori3: begin
346   Cout <=0; Zin<=0;
347   ZLowout <=1; Gra<=1; Rin<=1;
348 end
349
350 //andi
351 andi: begin
352   MDRout <= 0; IRin <= 0;
353   Grb <=1; Rout<=1; Yin <=1;
354 end
355 andi2: begin
356   Grb <=0; Rout<=0; Yin <=0;
357   Cout <=1; Zin<=1;
358   #20
359   ALUselect <= 4'b0110;
360 end
361 andi3: begin
362   Cout <=0; Zin<=0;
363   ZLowout <=1; Gra<=1; Rin<=1;
364 end
365
366 //br
367 br: begin
368   MDRout <= 0; IRin <= 0;
369   Gra <=1; Rout<=1; conf_in <=1;
370 end
371 br2: begin
372   Gra <=0; Rout<=0; conf_in <=0;
373   PCout<=1; Yin<=1;
374 end
375 br3: begin
376   PCout<=0; Yin<=0;
377   Cout<=1; Zin <=1;
378   ALUselect <=4'b0001;
379 end
380 br4: begin
381   Cout<=0; Zin <=0;
382   ZLowout<=1; conf_out<=1;
383 end
384
385 //jr
386 jr: begin
387   MDRout <= 0; IRin <= 0;
388   Gra <=1; Rout<=1; PCin <=1;
389 end
390
391 //jal
392 jal: begin
393   MDRout <= 0; IRin <= 0;
394   Ri5ctrl <=1; PCout <=1;
395 end
396
397 jal2: begin
398   Ri5ctrl <=0; PCout <=0;
399   Gra <=1; Rout<=1; PCin<=1;
400 end
401
402 //mfhi
403 mfhi: begin
404   MDRout <= 0; IRin <= 0;
405   Gra <=1; Rin<=1; Hiout <=1;
406 end
407
408 //mflo
409 mflo: begin
410   MDRout <= 0; IRin <= 0;
411   Gra <=1; Rin<=1; Loout <=1;
412 end
413
414 //in
415 in: begin
416   MDRout <= 0; IRin <= 0;
417   Gra <=1; Rin<=1; InPortout <=1;
418 end
419

```

```

420 //out
421 out: begin
422     MDRout <= 0; IRin <= 0;
423     Gra <=1; Rout<=1; OPin <=1;
424
425 end
426
427
428 //OR
429 or1: begin
430     MDRout <= 0; IRin <= 0;
431     Grb <=1; Rout <=1; Yin <=1;
432 end
433 or2: begin
434     Grb <=0; Yin <=0;
435     Grc <= 1; Rout <= 1; Zinc=1;
436     #20
437     ALUselect <= 4'b0111;
438 end
439 or3: begin
440     Grc <= 0; Rout <= 0; Zin<=0;
441     Gra<=1; ZLowout<=1; Rin <=1;
442 end
443
444 //AND
445 and1: begin
446     MDRout <= 0; IRin <= 0;
447     Grb <=1; Rout <=1; Yin <=1;
448 end
449 and2: begin
450     Grb <=0; Yin <=0;
451     Grc <= 1; Rout <= 1; Zin<=1;
452     #20
453     ALUselect <= 4'b0110;
454 end
455 and3: begin
456     Grc <= 0; Rout <= 0; Zin<=0;
457     Gra<=1; ZLowout<=1; Rin <=1;
458 end
459
460 //ADD
461 add: begin
462     MDRout <= 0; IRin <= 0;
463     Grb <=1; Rout <=1; Yin <=1;
464 end
465 add2: begin
466     Grb <=0; Yin <=0;
467     Grc <= 1; Rout <= 1; Zin<=1;
468     #20
469     ALUselect <= 4'b0001;
470 end
471 add3: begin
472     Grc <= 0; Rout <= 0; Zin<=0;
473     Gra<=1; ZLowout<=1; Rin <=1;
474 end
475
476 //SUB
477 sub: begin
478     MDRout <= 0; IRin <= 0;
479     Grb <=1; Rout <=1; Yin <=1;
480 end
481 sub2: begin
482     Grb <=0; Yin <=0;
483     Grc <= 1; Rout <= 1; Zin<=1;
484     #20
485     ALUselect <= 4'b0010;
486 end
487 sub3: begin
488     Grc <= 0; Rout <= 0; Zin<=0;
489     Gra<=1; ZLowout<=1; Rin <=1;
490 end
491
492 //MUL
493 mul: begin
494     MDRout <= 0; IRin <= 0;
495     Gra <=1; Rout <=1; Yin <=1;
496 end
497 mul2: begin
498     Gra <=0; Yin <=0;
499     Grb <= 1; Rout <= 1; Zin<=1;
500     #20
501     ALUselect <= 4'b0011;
502 end
503 mul3: begin
504     Grb <= 0; Rout <= 0; Zin<=0;
505     ZLowout<=1; LOin <=1;
506 end
507 mul4: begin
508     LOin <=0; ZLowout<=0; ZHighout <=1; HIn <= 1;
509 end
510
511 //DIV
512 div: begin
513     MDRout <= 0; IRin <= 0;
514     Gra <=1; Rout <=1; Yin <=1;
515 end
516 div2: begin
517     Gra <=0; Yin <=0;
518     Grb <= 1; Rout <= 1; Zin<=1;
519     #20
520     ALUselect <= 4'b0101;
521 end
522 div3: begin
523     Grb <= 0; Rout <= 0; Zin<=0;
524     ZLowout<=1; LOin <=1;
525 end
526 div4: begin
527     LOin <=0; ZLowout<=0; ZHighout <=1; HIn <= 1;
528 end

```

```

530 //SHR
531 shr: begin
532     MDRout <= 0; IRin <= 0;
533     Grb <=1; Rout <=1; Yin <=1;
534 end
535 shr2: begin
536     Grb <=0; Yin <=0;
537     Grc <= 1; Rout <= 1; Zin<=1;
538     #20
539     ALUselect <= 4'b1101;
540 end
541 shr3: begin
542     Grc <= 0; Rout <= 0; Zin<=0;
543     Gra<=1; ZLowout<=1; Rin <=1;
544 end
545
546 //SHL
547 shl: begin
548     MDRout <= 0; IRin <= 0;
549     Grb <=1; Rout <=1; Yin <=1;
550 end
551 shl2: begin
552     Grb <=0; Yin <=0;
553     Grc <= 1; Rout <= 1; Zin<=1;
554     #20
555     ALUselect <= 4'b1100;
556 end
557 shl3: begin
558     Grc <= 0; Rout <= 0; Zin<=0;
559     Gra<=1; ZLowout<=1; Rin <=1;
560 end
561
562 //ROR
563 ror: begin
564     MDRout <= 0; IRin <= 0;
565     Grb <=1; Rout <=1; Yin <=1;
566 end
567 ror2: begin
568     Grb <=0; Yin <=0;
569     Grc <= 1; Rout <= 1; Zin<=1;
570     #20
571     ALUselect <= 4'b1111;
572 end
573 ror3: begin
574     Grc <= 0; Rout <= 0; Zin<=0;
575     Gra<=1; ZLowout<=1; Rin <=1;
576 end
577
578 //ROL
579 rol: begin
580     MDRout <= 0; IRin <= 0;
581     Grb <=1; Rout <=1; Yin <=1;
582 end
583 rol2: begin
584     Grb <=0; Yin <=0;
585     Grc <= 1; Rout <= 1; Zin<=1;
586     #20
587     ALUselect <= 4'b1110;
588 end
589 rol3: begin
590     Grc <= 0; Rout <= 0; Zin<=0;
591     Gra<=1; ZLowout<=1; Rin <=1;
592 end
593
594 //NEG
595 neg: begin
596     MDRout <= 0; IRin <= 0;
597     Grb<=1;Rout <=1;ALUselect <= 4'b1000;Zin <=1;
598 end
599 neg2: begin
600     Grb<=0; Rout <=0; Zin <=0;
601     ZLowout <=1; Gra <=1; Rin <=1;
602 end
603
604 //NOT
605 not1: begin
606     MDRout <= 0; IRin <= 0;
607     Grb<=1;Rout <=1;ALUselect <= 4'b1010;Zin <=1;
608 end
609 not2: begin
610     Grb<=0; Rout <=0; Zin <=0;
611     ZLowout <=1; Gra <=1; Rin <=1;
612 end
613
614 //NOP
615 nop:begin
616
617 end
618
619 nop2:begin
620
621 end
622
623 // HALT
624 halt:begin
625     Run <= 0;
626 end
627
628
629 endcase
630 end
631 endmodule

```

main1.v

```
1 module main1(
2     input Rin,
3     input Rout,
4     input HIin, LOin, PCin, IRin, Yin, InPortout, Zin, conIn, outPortin, R15ctrl,
5     input HIout, LOout, PCout, MDRout, MDRin, MARin, MDRread, memWrite, Cout, clk, IncPC, ZLowout, ZHighout, conOut, BAout, Gra, Grb, Grc,
6     input [3:0] ALUselect,
7     input [31:0] MDatain,
8     output [31:0] IRdata
9 );
10
11 wire[63:0] ZReg;
12 wire[31:0] bus, PCtemp;
13 wire clr;
14 wire IRout, R15in;
15 wire [31:0] YData, XData;
16 wire [31:0] ZLowData, ZHighData;
17
18 wire [31:0] R0temp, busInR0, busInR1, busInR2, busInR3, busInR4, busInR5, busInR6, busInR7, busInR8, busInR9, busInR10, busInR11, busInR12, busInR13, busInR14, busInR15,
19     busInPC, busInMAR, busInMDR, busInHI, busInLO, busInZ, busInInPort, busInC;
20
21 wire [15:0] genRegIn, genRegOut;
22
23 ctrl_unit ctrl(GraI, GrbI, GrcI, RinI, RoutI, MDRinI, MDRoutI, BAoutI, CoutI, ZLowoutI, PCinI, IRinI, HIoutI, LOoutI,
24     InPortoutI, outPortinI, LOinI, ZHighoutI, HIinI, YinI, ZinI, PCoutI, IncPCI, MARinI, readI, wrenI, clrI, conInI, RunI,
25     ALUselectI, IRdataI, clkI, resetI, stpl, conOutI);
26
27
28
29 gen_reg r0(R0temp, bus, genRegIn[0], clr, clk);
30
31 assign busInR0 = BAout ? 32'b0 : R0temp;
32
33 gen_reg r1(busInR1, bus, genRegIn[1], clr, clk);
34 gen_reg r2(busInR2, bus, genRegIn[2], clr, clk);
35 gen_reg r3(busInR3, bus, genRegIn[3], clr, clk);
36 gen_reg r4(busInR4, bus, genRegIn[4], clr, clk);
37 gen_reg r5(busInR5, bus, genRegIn[5], clr, clk);
38 gen_reg r6(busInR6, bus, genRegIn[6], clr, clk);
39 gen_reg r7(busInR7, bus, genRegIn[7], clr, clk);
40 gen_reg r8(busInR8, bus, genRegIn[8], clr, clk);
41 gen_reg r9(busInR9, bus, genRegIn[9], clr, clk);
42 gen_reg r10(busInR10, bus, genRegIn[10], clr, clk);
43 gen_reg r11(busInR11, bus, genRegIn[11], clr, clk);
44 gen_reg r12(busInR12, bus, genRegIn[12], clr, clk);
45 gen_reg r13(busInR13, bus, genRegIn[13], clr, clk);
46 gen_reg r14(busInR14, bus, genRegIn[14], clr, clk);
47 gen_reg r15(busInR15, bus, R15ctrl, clr, clk);
48
49 assign R15in = genRegIn[15] | R15ctrl;
50
51 sel_enc selectEncodeLogic(IRdata, Rin, Rout, BAout, Cout, Gra, Grb, Grc, genRegIn, genRegOut, busInC);
52
53 con_ff conFF(IRdata, bus, conIn, clk, conFFOut);
54 inoutport inOutPort(outPortin, clr, clk, inPortout, busInInPort, bus);
55
56
57
58 gen_reg ir(IRdata, bus, IRin, clr, clk);
59 pc_reg pc(busInPC, bus, PCin, conOut, conFFOut, IncPC, clr, clk);
60
61 memSubsys memSys(MARin, busInMDR, MDatain, bus, MDRin, MDRread, memWrite, clr, clk);
62
63 gen_reg hi(busInHI, bus, HIin, clr, clk);
64 gen_reg lo(busInLO, bus, LOin, clr, clk);
65 gen_reg y(YData, bus, Yin, clr, clk);
66 z_reg_64 z(busInZ, ZReg, Zin, ZLowout, ZHighout, clr, clk);
67
68 //ALU
69 alu alu(ALUselect, clk, YData, bus, ZReg, carry);
70
71 // Bus
72 bus bus_inst(busInR0, busInR1, busInR2, busInR3, busInR4, busInR5, busInR6, busInR7, busInR8, busInR9, busInR10, busInR11,
73     busInR12, busInR13, busInR14, busInR15, busInHI, busInLO, busInZ, busInPC, busInMDR, busInInPort, busInC,
74     genRegOut[0], genRegOut[1], genRegOut[2], genRegOut[3], genRegOut[4], genRegOut[5], genRegOut[6], genRegOut[7], genRegOut[8],
75     genRegOut[9], genRegOut[10], genRegOut[11], genRegOut[12], genRegOut[13], genRegOut[14], genRegOut[15], HIout,
76     LOout, ZHighout, ZLowout, PCout, MDRout, InPortout, Cout, bus, clk);
77 endmodule
78
79
```


pc_reg.v

```
1 module pc_reg(  
2     output reg [31:0] Q,  
3     input [31:0] D,  
4     input wr, conFFwr, conFFen, inc, clr, clk  
5 );  
6     initial Q=0;  
7     reg incHelp, wrHelp;  
8     initial incHelp=0;  
9     initial wrHelp=0;  
10    always @(posedge clk)  
11    begin  
12        if(inc)  
13            incHelp <= 1;  
14        if(wr)  
15            begin  
16                Q <= D;  
17                wrHelp <= 1;  
18            end  
19        if(!wr && wrHelp)  
20            begin  
21                wrHelp <= 0;  
22                if(incHelp)  
23                    begin  
24                        incHelp <= 0;  
25                        Q <= Q+1;  
26                    end  
27                end  
28            if(conFFen && conFFwr)  
29                Q <= D;  
30            if(clr)  
31                Q <= 0;  
32        end  
33    endmodule  
34
```

memSubsys.v

```
1 module memSubsys(input MARin, output reg[31:0] busInMDR, input[31:0] MDatain, input[31:0] bus, input MDRin, input read, input write,  
2     input clr, input clk);  
3     wire[8:0] address;  
4     wire[31:0] memOut, qMDR;  
5     gen_reg mdr(address, bus, MARin, clr, clk);  
6     mdr_reg mdr(qMDR, memOut, bus, MDRin, read, clr, clk);  
7     ram_memory(address, clk, qMDR, read, write, memOut);  
8     always @(posedge clk)  
9     begin  
10        if(read) busInMDR=qMDR;  
11        else if(write) busInMDR=memOut;  
12    end  
13 endmodule
```

datapath_tb.v

```

1  `timescale 1ns/10ps
2  module datapath_tb;
3
4      wire Rin;
5
6      wire Rout;
7
8      wire HIin, LOin, PCin, IRin, Yin, InPortout, Zin, conIn, outPortin, R15ctrl;
9
10     wire HIout, LOout, PCout, MDRout, MDRin, MARin, MDRread, Cout, IncPC, ZLowout, ZHighout;
11
12     reg clk;
13
14     wire conOut, BAout, Gra, Grb, Grc;
15
16     wire [3:0] ALUselect;
17     reg [31:0] MDatain;
18     wire Read, Write;
19
20     wire [31:0] IRdata;
21
22     parameter Default = 4'b0000, Reg_load1a = 4'b0001, Reg_load1b = 4'b0010, Reg_load2a = 4'b0011, Reg_load2b = 4'b0100,
23           Reg_load3a = 4'b0101, Reg_load3b = 4'b0110, T0 = 4'b0111, T1 = 4'b1000, T2 = 4'b1001, T3 = 4'b1010, T4 = 4'b1011,
24           T5 = 4'b1100, T6 = 4'b1101, T7 = 4'b1110;
25
26     reg [3:0] Present_state = Default;
27
28     main1 DUT (.Rin(Rin), .Rout(Rout), .HIin(HIin), .LOin(LOin), .PCin(PCin), .IRin(IRin), .Yin(Yin), .InPortout(InPortout),
29             .Zin(Zin), .conIn(conIn), .outPortin(outPortin), .R15ctrl(R15ctrl), .HIout(HIout), .LOout(LOout), .PCout(PCout), .MDRout(MDRout), .MDRin(MDRin),
30             .MARin(MARin), .MDRread(Read), .memWrite(Write), .Cout(Cout), .clk(clk), .IncPC(IncPC), .ZLowout(ZLowout), .ZHighout(ZHighout),
31             .conOut(conOut), .BAout(BAout), .Gra(Gra), .Grb(Grb), .Grc(Grc), .ALUselect(ALUselect), .MDatain(MDatain), .IRdata(IRdata));
32
33
34
35     //control signals seem to work, but IR in ctrl unit isn't recieving data in current form
36     ctrl_unit ctrl(Gra, Grb, Grc, Rin, Rout, MDRin, MDRout, BAout, Cout, ZLowout, PCin, IRin, HIout, LOout,
37             InPortout, outPortin, LOin, ZHighout, HIin, Yin, Zin, PCout, IncPC, MARin, Read, Write, clr, conIn, Run,
38             ALUselect, IRdata, clk, reset, stp, conOut, R15ctrl);
39
40
41
42     initial
43     begin
44         clk = 0;
45         forever #5 clk = ~clk;
46     end
47
48
49     always @(posedge clk)
50     begin
51         case(Present_state)
52             Default      : #40 Present_state = T0;
53             T0           : #40 Present_state = T1;
54             T1           : #40 Present_state = T2;
55             T2           : #40 Present_state = T3;
56             T3           : #40 Present_state = T4;
57             T4           : #40 Present_state = T5;
58             T5           : #40 Present_state = T6;
59             T6           : #40 Present_state = T7;
60         endcase
61     end
62
63     //always @(Present_state)
64     // begin
65     //     case(Present_state)
66     //         Default:begin
67     //             Rout <=0; Rin <=0;
68     //             HIout <=0; LOout <=0; InPortout <=0; Cout <=0;
69     //             LOin <=0; HIin <=0;
70     //             PCout <=0; ZLowout <=0; ZHighout <=0; MDRout <=0;
71     //             MARin <=0; Zin <=0;
72     //             PCin <=0; MDRin <=0; IRin <=0; Yin <=0;
73     //             IncPC <=0; Read <=0; Write <=0; ALUselect <=0;
74     //             Gra <=0; Grb <=0; Grc <=0; BAout <=0; conOut <=0;
75     //             MDatain<=32'h00000000;
76     //         end
77     //     end

```

```

77 //
78 //
79 // //ld R1, $85 : 8388693
80 // //ld R0, $35(R1) : 524323
81 // //ldi R1, $85 : 142606421
82 // //ldi R0, $35(R1) : 134742051
83 // //st $90, R1 : 276824154
84 // //st $90(R1), R1 : 277348442
85 // //addi R2, R1, -5 : 1494220795
86 // //andi R2, R1, $26 : 1627914266
87 // //ori R2, R1, $26 : 1762131994
88 // //brzr R2, 35 : 2432696355
89 // //brnz R2, 35 : 2433220643
90 // //brpl R2, 35 : 2433744931
91 // //brmi R2, 35 : 2434269219
92 // //jr R1 : 2558525440
93 // //jal R1 : 2692743168
94 // //mfhi R2 : 3103784960
95 // //mflo R2 : 3238002688
96 // //out R1 : 2961178624
97 // //in R1 : 2826960896
98 //
99 //
100 //
101 //
102 // T0: begin #5 PCout <= 1; MARin <= 1; IncPC <= 1; Zin <= 1; ALUselect <= 4'b1001;
103 // #30 PCout <= 0; MARin <= 0; IncPC <= 0; Zin <= 0; ALUselect <= 4'b0000; end
104 //
105 // T1: begin #5 ZLowout <= 1; PCin <= 1; Read <=1; MDRin <=1;
106 // #30 ZLowout <= 0; PCin <= 0; Read <=0; MDRin <=0; end
107 //
108 // T2: begin #5 MDRout <= 1; IRin <=1;
109 // #30 MDRout <= 0; IRin <=0; end
110 //
111 //ldi
112 // T3: begin #5 Grb <= 1; BAout <= 1; Yin <= 1;
113 // #30 Grb <= 0; BAout <= 0; Yin <= 0; end
114 //
115 // T4: begin #5 Cout <= 1; ALUselect <= 4'b0001; Zin <= 1;
116 // #30 Cout <= 0; ALUselect <= 4'b0000; Zin <= 0; end
117 //
118 // T5: begin #5 ZLowout <= 1; MARin <= 1;
119 // #30 ZLowout <= 0; MARin <= 0; end
120 //
121 // T6: begin #5 Read <= 1; MDRin <= 1;
122 // #30 Read <= 0; MDRin <= 0; end
123 //
124 // T7: begin #5 MDRout <= 1; Gra <= 1; Rin <= 1;
125 // #30 MDRout <= 0; Gra <= 0; Rin <= 0; end
126 //
127 //ldi
128 // T3: begin #5 Grb <= 1; BAout <= 1; Yin <= 1;
129 // #30 Grb <= 0; BAout <= 0; Yin <= 0; end
130 //
131 // T4: begin #5 Cout <= 1; ALUselect <= 4'b0001; Zin <= 1;
132 // #30 Cout <= 0; ALUselect <= 4'b0000; Zin <= 0; end
133 //
134 // T5: begin #5 ZLowout <= 1; Gra <= 1; Rin <= 1;
135 // #30 ZLowout <= 0; Gra <= 0; Rin <= 0; end
136 //
137 //st
138 // T3: begin #5 Grb <= 1; BAout <= 1; Yin <= 1;
139 // #30 Grb <= 0; BAout <= 0; Yin <= 0; end
140 //
141 // T4: begin #5 Cout <= 1; ALUselect <= 4'b0001; Zin <= 1;
142 // #30 Cout <= 0; ALUselect <= 4'b0000; Zin <= 0; end
143 //
144 // T5: begin #5 ZLowout <= 1; MARin <= 1;
145 // #30 ZLowout <= 0; MARin <= 0; end
146 //
147 // T6: begin #5 MDRin <= 1; Gra <= 1; Rout <= 1;
148 // #30 MDRin <= 0; Gra <= 0; Rout <= 0; end
149 //
150 // T7: begin #5 MDRout <= 1; Write <= 1;
151 // #30 MDRout <= 0; Write <= 0; end
152 //
153 //addi, andi, ori
154 //0001, 0110, 0111
155 // T3: begin #5 Grb <= 1; Rout <= 1; Yin <= 1;
156 // #30 Grb <= 0; Rout <= 0; Yin <= 0; end
157 //

```

```

159      ///      T4: begin #5 Cout <= 1; ALUselect <= 4'b0111; Zin <= 1;
160      ///      #30 Cout <= 0; ALUselect <= 4'b0000; Zin <= 0; end
161      ///
162      ///      T5: begin #5 ZLowout <= 1; Gra <= 1; Rin <= 1;
163      ///      #30 ZLowout <= 0; Gra <= 0; Rin <= 0; end
164      ///
165      ///brzr, brnz, brpl, brmi
166      ///      T3: begin #5 Gra <= 1; Rout <= 1; conIn <= 1;
167      ///      #30 Gra <= 0; Rout <= 0; conIn <= 0; end
168      ///
169      ///      T4: begin #5 PCout <= 1; Yin <= 1;
170      ///      #30 PCout <= 0; Yin <= 0; end
171      ///
172      ///      T5: begin #5 Cout <= 1; ALUselect <= 4'b0001; Zin <= 1;
173      ///      #30 Cout <= 0; ALUselect <= 4'b0000; Zin <= 0; end
174      ///
175      ///      T6: begin #5 ZLowout <= 1; conOut <= 1;
176      ///      #30 ZLowout <= 0; conOut <= 0; end
177      ///jzr
178      ///      T3: begin #5 Gra <= 1; Rout <= 1; PCin <= 1;
179      ///      #30 Gra <= 0; Rout <= 0; PCin <= 0; end
180      ///
181      ///jal
182      ///      T3: begin #5 R15ctrl <= 1; PCout <= 1;
183      ///      #30 R15ctrl <= 0; PCout <= 0; end
184      ///
185      ///      T4: begin #5 Gra <= 1; Rout <= 1; PCin <= 1;
186      ///      #30 Gra <= 0; Rout <= 0; PCin <= 0; end
187      ///
188      ///mfhi
189      ///      T3: begin #5 HIout <= 1; Gra <= 1; Rin <= 1;
190      ///      #30 HIout <= 0; Gra <= 0; Rin <= 0; end
191      ///
192      ///mflo
193      ///      T3: begin #5 LOout <= 1; Gra <= 1; Rin <= 1;
194      ///      #30 LOout <= 0; Gra <= 0; Rin <= 0; end
195      ///
196      ///out
197      ///      T3: begin #5 Gra <= 1; Rout <= 1; outPortIn <= 1;
198      ///      #30 Gra <= 0; Rout <= 0; outPortIn <= 0; end
199      ///
200      ///in
201      ///      T3: begin #5 Gra <= 1; Rin <= 1; InPortout <= 1;
202      ///      #30 Gra <= 0; Rin <= 0; InPortout <= 0; end
203      ///
204      ///
205      ///      endcase
206      ///      end
207  endmodule

```

Functional Simulations

Instructions 0-15

	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16
/datapath_tb/DUT/#C/Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
/datapath_tb/DUT/#1/Q	00000000	000...09800087	09980001	01000075	0917ffff	00900004	08000001	09800073	91980003	09980005	039ffffd	4800...	93900002	119918000	9868		
/datapath_tb/DUT/#2/Q	00000000	00...09800087	09980001	01000075	00...0917ffff	00900004	00...08000001	09800073	91980003	09980005	039ffffd	00...	4800...	93900002	119918000	9868	
/datapath_tb/DUT/#3/Q	00000000	00000000	00000001	00000002	00...	00000003	00000004	00...	00000005	00000006	00000007	00000008	00000009	00...	0000...	0000000a	0000000b
/datapath_tb/DUT/#4/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#5/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#6/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#7/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#8/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#9/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#10/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#11/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#12/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#13/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#14/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#15/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#16/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
/datapath_tb/DUT/#17/Q	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Instructions 16-31

[illegible]

Instructions 32-end

[illegible]

State of memory

Memory before run

[illegible]

Memory after run

00000000	00001001100000000000000010000111	00001001100110000000000000000001	0000000100000000000000000110101	000010010001011111111111111110
00000004	000000000100100000000000000000100	00001000000000000000000000000001	0000100110000000000000000000011001	100100011001100000000000000000011
00000008	000010011001100000000000000000101	0000011100111111111111111111101	11001000000000000000000000000000	10010011100100000000000000000010
00000012	000010100000100000000000000000110	00001001101000000000000000000010	00011001100100011000000000000000	0101101101100000000000000000011
00000016	10000011011100000000000000000000	00010111011100000000000000000000	01100011011100000000000000000111	01101011000100000000000000000001
00000020	00101001000110000000000000000000	00010001000000000000000001011000	00111000100010000000000000000000	01000010001000000000000000000000
00000024	01010000010000000000000000000000	01001000100100001000000000000000	0001000100001000000000000000011011	00100001100110000000000000000000
00000028	00100000100100000000000000000000	00001010000000000000000000000101	0000101010000000000000000000011011	01110010100100000000000000000000
00000032	10111011100000000000000000000000	11000011000000000000000000000000	01111010101000000000000000000000	00001101001000000000000000000000
00000036	00000110101010000000000000000010	00001110001100000000000000000000	00001110101110000000000000000000	10100110000000000000000000000000
00000040	11010000000000000000000000000000	00000011000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000044	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000048	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000052	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000056	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000060	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000064	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000068	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000072	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000076	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000080	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000084	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000088	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000092	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000
00000096	00000000000000000000000000000000	00000000000000000000000000000000	00000000000000000000000000000000	0

Memory before run (Hex)

[illegible]

Memory after run (Hex)

[illegible]