

Digital Design and Computer Organization Laboratory

UE21CS251A

3rd Semester, Academic Year 2022-23

Date: 29/11/2022

Name : Sujal.S	SRN : PES2UG21CS548	Section : I
----------------	---------------------	-------------

Title of the Program:

Microprocessor Control Logic-2

Aim of the Program:

The task in this assignment the intent of this assignment is to enhance the control logic to implement a load and a jump instruction

```
1 module fa (input wire i0, i1, cin, output wire sum, cout);
2   wire t0, t1, t2;
3   xor3 _i0 (i0, i1, cin, sum);
4   and2 _i1 (i0, i1, t0);
5   and2 _i2 (i1, cin, t1);
6   and2 _i3 (cin, i0, t2);
7   or3 _i4 (t0, t1, t2, cout);
8 endmodule
9
10 module addsub (input wire addsub, i0, i1, cin, output wire sumdiff, cout);
11   wire t;
12   fa _i0 (i0, t, cin, sumdiff, cout);
13   xor2 _i1 (i1, addsub, t);
14 endmodule
15
16 module alu_slice (input wire [1:0] op, input wire i0, i1, cin, output wire o, cout);
17   wire t_sumdiff, t_and, t_or, t_andor;
18   addsub _i0 (op[0], i0, i1, cin, t_sumdiff, cout);
19   and2 _i1 (i0, i1, t_and);
20   or2 _i2 (i0, i1, t_or);
21   mux2 _i3 (t_and, t_or, op[0], t_andor);
22   mux2 _i4 (t_sumdiff, t_andor, op[1], o);
23 endmodule
```

```

25 module alu (input wire [1:0] op, input wire [15:0] i0, i1,
26     output wire [15:0] o, output wire cout);
27     wire [14:0] c;
28     alu_slice _i0 (op, i0[0], i1[0], op[0], o[0], c[0]);
29     alu_slice _i1 (op, i0[1], i1[1], c[0], o[1], c[1]);
30     alu_slice _i2 (op, i0[2], i1[2], c[1], o[2], c[2]);
31     alu_slice _i3 (op, i0[3], i1[3], c[2], o[3], c[3]);
32     alu_slice _i4 (op, i0[4], i1[4], c[3], o[4], c[4]);
33     alu_slice _i5 (op, i0[5], i1[5], c[4], o[5], c[5]);
34     alu_slice _i6 (op, i0[6], i1[6], c[5], o[6], c[6]);
35     alu_slice _i7 (op, i0[7], i1[7], c[6], o[7], c[7]);
36     alu_slice _i8 (op, i0[8], i1[8], c[7], o[8], c[8]);
37     alu_slice _i9 (op, i0[9], i1[9], c[8], o[9], c[9]);
38     alu_slice _i10 (op, i0[10], i1[10], c[9], o[10], c[10]);
39     alu_slice _i11 (op, i0[11], i1[11], c[10], o[11], c[11]);
40     alu_slice _i12 (op, i0[12], i1[12], c[11], o[12], c[12]);
41     alu_slice _i13 (op, i0[13], i1[13], c[12], o[13], c[13]);
42     alu_slice _i14 (op, i0[14], i1[14], c[13], o[14], c[14]);
43     alu_slice _i15 (op, i0[15], i1[15], c[14], o[15], cout);
44 endmodule

```

```

1 module invert (input wire i, output wire o);
2     assign o = !i;
3 endmodule
4
5 module and2 (input wire i0, i1, output wire o);
6     assign o = i0 & i1;
7 endmodule
8
9 module or2 (input wire i0, i1, output wire o);
10    assign o = i0 | i1;
11 endmodule
12
13 module xor2 (input wire i0, i1, output wire o);
14    assign o = i0 ^ i1;
15 endmodule
16
17 module nand2 (input wire i0, i1, output wire o);
18    wire t;
19    and2 and2_0 (i0, i1, t);
20    invert invert_0 (t, o);
21 endmodule
22
23 module nor2 (input wire i0, i1, output wire o);
24    wire t;
25    or2 or2_0 (i0, i1, t);
26    invert invert_0 (t, o);
27 endmodule
28
29 module xnor2 (input wire i0, i1, output wire o);
30    wire t;
31    xor2 xor2_0 (i0, i1, t);
32    invert invert_0 (t, o);
33 endmodule
34

```

```
35 module and3 (input wire i0, i1, i2, output wire o);
36     wire t;
37     and2 and2_0 (i0, i1, t);
38     and2 and2_1 (i2, t, o);
39 endmodule
40
41 module or3 (input wire i0, i1, i2, output wire o);
42     wire t;
43     or2 or2_0 (i0, i1, t);
44     or2 or2_1 (i2, t, o);
45 endmodule
46
47 module nor3 (input wire i0, i1, i2, output wire o);
48     wire t;
49     or2 or2_0 (i0, i1, t);
50     nor2 nor2_0 (i2, t, o);
51 endmodule
52
53 module nand3 (input wire i0, i1, i2, output wire o);
54     wire t;
55     and2 and2_0 (i0, i1, t);
56     nand2 nand2_1 (i2, t, o);
57 endmodule
58
59 module xor3 (input wire i0, i1, i2, output wire o);
60     wire t;
61     xor2 xor2_0 (i0, i1, t);
62     xor2 xor2_1 (i2, t, o);
63 endmodule
64
65 module xnor3 (input wire i0, i1, i2, output wire o);
66     wire t;
67     xor2 xor2_0 (i0, i1, t);
68     xnor2 xnor2_0 (i2, t, o);
69 endmodule
```

```

70
71 module mux2 (input wire i0, i1, j, output wire o);
72     assign o = (j==0)?i0:i1;
73 endmodule
74
75 module mux4 (input wire [0:3] i, input wire j1, j0, output wire o);
76     wire t0, t1;
77     mux2 mux2_0 (i[0], i[1], j1, t0);
78     mux2 mux2_1 (i[2], i[3], j1, t1);
79     mux2 mux2_2 (t0, t1, j0, o);
80 endmodule
81
82 module mux8 (input wire [0:7] i, input wire j2, j1, j0, output wire o);
83     wire t0, t1;
84     mux4 mux4_0 (i[0:3], j2, j1, t0);
85     mux4 mux4_1 (i[4:7], j2, j1, t1);
86     mux2 mux2_0 (t0, t1, j0, o);
87 endmodule
88
89 module demux2 (input wire i, j, output wire o0, o1);
90     assign o0 = (j==0)?i:1'b0;
91     assign o1 = (j==1)?i:1'b0;
92 endmodule
93
94 module demux4 (input wire i, j1, j0, output wire [0:3] o);
95     wire t0, t1;
96     demux2 demux2_0 (i, j1, t0, t1);
97     demux2 demux2_1 (t0, j0, o[0], o[1]);
98     demux2 demux2_2 (t1, j0, o[2], o[3]);
99 endmodule
100
101 module demux8 (input wire i, j2, j1, j0, output wire [0:7] o);
102     wire t0, t1;
103     demux2 demux2_0 (i, j2, t0, t1);
104     demux4 demux4_0 (t0, j1, j0, o[0:3]);
105     demux4 demux4_1 (t1, j1, j0, o[4:7]);
106 endmodule

```



```

107
108 module df (input wire clk, in, output wire out);
109     reg df_out;
110     always@(posedge clk) df_out ≤ in;
111     assign out = df_out;
112 endmodule
113
114 module dfr (input wire clk, reset, in, output wire out);
115     wire reset_, df_in;
116     invert invert_0 (reset, reset_);
117     and2 and2_0 (in, reset_, df_in);
118     df df_0 (clk, df_in, out);
119 endmodule
120
121 module dfrl (input wire clk, reset, load, in, output wire out);
122     wire _in;
123     mux2 mux2_0(out, in, load, _in);
124     dfr dfr_1(clk, reset, _in, out);
125 endmodule
126
127 module dfs (input wire clk, set, in, output wire out);
128     wire dfr_in, dfr_out;
129     invert invert_0(in, dfr_in);
130     invert invert_1(dfr_out, out);
131     dfr dfr_2(clk, set, dfr_in, dfr_out);
132 endmodule
133
134 module dfs1 (input wire clk, set, load, in, output wire out);
135     wire _in;
136     mux2 mux2_0(out, in, load, _in);
137     dfs dfs_1(clk, set, _in, out);
138 endmodule
139

```

```
1 module ir (input wire clk, reset, load, input wire [15:0] din, output wire [15:0] dout);
2   dfrl dfrl_0 (clk, reset, load, din['h0], dout['h0]);
3
4   dfrl dfrl_1 (clk, reset, load, din['h1], dout['h1]);
5
6   dfrl dfrl_2 (clk, reset, load, din['h2], dout['h2]);
7
8   dfrl dfrl_3 (clk, reset, load, din['h3], dout['h3]);
9
10  dfrl dfrl_4 (clk, reset, load, din['h4], dout['h4]);
11
12  dfrl dfrl_5 (clk, reset, load, din['h5], dout['h5]);
13
14  dfrl dfrl_6 (clk, reset, load, din['h6], dout['h6]);
15
16  dfrl dfrl_7 (clk, reset, load, din['h7], dout['h7]);
17
18  dfrl dfrl_8 (clk, reset, load, din['h8], dout['h8]);
19
20  dfrl dfrl_9 (clk, reset, load, din['h9], dout['h9]);
21
22  dfrl dfrl_a (clk, reset, load, din['ha], dout['ha]);
23
24  dfrl dfrl_b (clk, reset, load, din['hb], dout['hb]);
25
26  dfrl dfrl_c (clk, reset, load, din['hc], dout['hc]);
27
28  dfrl dfrl_d (clk, reset, load, din['hd], dout['hd]);
29
30  dfrl dfrl_e (clk, reset, load, din['he], dout['he]);
31
32  dfrl dfrl_f (clk, reset, load, din['hf], dout['hf]);
33
34 endmodule
35
```

```

58
59 module nor5 (input wire [0:4] i, output wire o);
60
61 wire t;
62
63 or3 or3_0 (i[0], i[1], i[2], t);
64
65 nor3 nor3_0 (t, i[3], i[4], o);
66 endmodule
67 module control_logic (input wire clk, reset, input wire [15:0] cur_ins, output wire [2:0] rd_addr_a, rd_addr_b,
68 wr_addr, output wire [1:0] op, output wire pc_inc, load_ir, wr_reg);
69
70 wire t, alu_ins;
71
72 dfsl fetch (clk, reset, 1'b1, wr_reg, pc_inc);
73 assign load_ir=pc_inc;
74 dfnl dec_exec (clk, reset, 1'b1, load_ir, t);
75 nor5 nor5_0 (cur_ins[15:11], alu_ins);
76 and2 and2_0 (t, alu_ins, wr_reg);
77 assign rd_addr_a = cur_ins[2:0];
78 assign rd_addr_b = cur_ins[5:3];
79 assign wr_addr = cur_ins[8:6];
80 assign op = cur_ins[10:9];
81 endmodule
82 module mproc (input wire clk, reset, input wire [15:0] ins, output wire [15:0] addr);
83 wire pc_inc, cout; wire [2:0] rd_addr_a, rd_addr_b, wr_addr; wire [1:0] op; wire [15:0] cur_ins, d_out_a, d_out_b;
84
85 pc pc_0 (clk, reset, pc_inc, 1'b0, 1'b0, 16'b0, addr);
86 ir ir_0 (clk, reset, load_ir, ins, cur_ins);
87 control_logic control_logic_0 (clk, reset, cur_ins, rd_addr_a, rd_addr_b, wr_addr, op, pc_inc, load_ir, wr_reg);
88 reg_alu reg_alu_0 (clk, reset, 1'b1, wr_reg, op, rd_addr_a, rd_addr_b, wr_addr, 16'b0, d_out_a, d_out_b, cout);
89 endmodule

```

```

1 module ram_128_16 (input wire clk, reset, wr, input wire [6:0] addr,
2 input wire [15:0] din, output wire [15:0] dout);
3   reg [0:127] ram [15:0];
4
5   initial begin
6     ram[0]=16'o000100;
7     ram[1]=16'o001201;
8     ram[2]=16'o002321;
9     ram[3]=16'o003432;
10  end
11  always @(wr) ram[addr]=din;
12  assign dout=ram[addr];
13 endmodule
14
15 module mproc_mem (input wire clk, reset);
16   wire [15:0] addr; wire [15:0] ins;
17
18   ram_128_16 ram_128_16_0 (clk, reset, 1'b0, addr[6:0], 16'b0, ins);
19   mproc mproc_0 (clk, reset, ins, addr);
20 endmodule

```

```

1 module pc_slice (input wire clk, reset, cin, load, inc, sub, offset,
2 output wire cout, pc);
3   wire in, inc_;
4   invert invert_0 (inc, inc_);
5   and2 and2_0 (offset, inc_, t);
6   addsub addsub_0 (sub, pc, t, cin, in, cout);
7   dfrl dfrl_0 (clk, reset, load, in, pc);
8 endmodule
9 module pc_slice0 (input wire clk, reset, cin, load, inc, sub, offset,
10 output wire cout, pc);
11   wire in;
12   or2 or2_0 (offset, inc, t);
13   addsub addsub_0 (sub, pc, t, cin, in, cout);
14   dfrl dfrl_0 (clk, reset, load, in, pc);
15 endmodule
16 module pc (input wire clk, reset, inc, add, sub, input wire [15:0] offset,
17 output wire [15:0] pc);
18   input wire load; input wire [15:0] c;
19   or3 or3_0 (inc, add, sub, load);
20   pc_slice0 pc_slice_0 (clk, reset, c[0], load, inc, sub, offset[0], c[0], pc[0]);
21   pc_slice pc_slice_1 (clk, reset, c[0], load, inc, sub, offset[1], c[1], pc[1]);
22   pc_slice pc_slice_2 (clk, reset, c[1], load, inc, sub, offset[2], c[2], pc[2]);
23   pc_slice pc_slice_3 (clk, reset, c[2], load, inc, sub, offset[3], c[3], pc[3]);
24   pc_slice pc_slice_4 (clk, reset, c[3], load, inc, sub, offset[4], c[4], pc[4]);
25   pc_slice pc_slice_5 (clk, reset, c[4], load, inc, sub, offset[5], c[5], pc[5]);
26   pc_slice pc_slice_6 (clk, reset, c[5], load, inc, sub, offset[6], c[6], pc[6]);
27   pc_slice pc_slice_7 (clk, reset, c[6], load, inc, sub, offset[7], c[7], pc[7]);
28   pc_slice pc_slice_8 (clk, reset, c[7], load, inc, sub, offset[8], c[8], pc[8]);
29   pc_slice pc_slice_9 (clk, reset, c[8], load, inc, sub, offset[9], c[9], pc[9]);
30   pc_slice pc_slice_10 (clk, reset, c[9], load, inc, sub, offset[10], c[10], pc[10]);
31   pc_slice pc_slice_11 (clk, reset, c[10], load, inc, sub, offset[11], c[11], pc[11]);
32   pc_slice pc_slice_12 (clk, reset, c[11], load, inc, sub, offset[12], c[12], pc[12]);
33   pc_slice pc_slice_13 (clk, reset, c[12], load, inc, sub, offset[13], c[13], pc[13]);
34   pc_slice pc_slice_14 (clk, reset, c[13], load, inc, sub, offset[14], c[14], pc[14]);
35   pc_slice pc_slice_15 (clk, reset, c[14], load, inc, sub, offset[15], c[15], pc[15]);
36 endmodule

```



```

1 module reg_file_2_1 (input wire clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr, d_in, output wire d_out_a, d_out_b);
2   wire l0, l1, o0, o1;
3   dfrl dfrl_0 (clk, reset, l0, d_in, o0);
4   dfrl dfrl_1 (clk, reset, l1, d_in, o1);
5   mux2 mux2_a (o0, o1, rd_addr_a, d_out_a);
6   mux2 mux2_b (o0, o1, rd_addr_b, d_out_b);
7   demux2 demux2_0 (wr, wr_addr, l0, l1);
8 endmodule
9
10 module _reg_file_2_1 (input wire clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr, d_in, output wire d_out_a, d_out_b);
11   wire l0, l1, o0, o1;
12   dfrl dfrl_0 (clk, reset, l0, d_in, o0);
13   dfrl dfrl_1 (clk, reset, l1, d_in, o1);
14   mux2 mux2_a (o0, o1, rd_addr_a, d_out_a);
15   mux2 mux2_b (o0, o1, rd_addr_b, d_out_b);
16   demux2 demux2_0 (wr, wr_addr, l0, l1);
17 endmodule
18
19 module _reg_file_4_1 (input wire clk, reset, wr, input wire [1:0] rd_addr_a, rd_addr_b, wr_addr, input wire d_in, output wire d_out_a, d_out_b);
20   wire wr0, wr1, o0_a, o0_b, o1_a, o1_b;
21   _reg_file_2_1 reg_file_2_1_0 (clk, reset, wr0, rd_addr_a[0], rd_addr_b[0], wr_addr[0],
22     d_in, o0_a, o0_b);
23   reg_file_2_1 reg_file_2_1_1 (clk, reset, wr1, rd_addr_a[0], rd_addr_b[0], wr_addr[0],
24     d_in, o1_a, o1_b);
25   mux2 mux2_a (o0_a, o1_a, rd_addr_a[1], d_out_a);
26   mux2 mux2_b (o0_b, o1_b, rd_addr_b[1], d_out_b);
27   demux2 demux2_0 (wr, wr_addr[1], wr0, wr1);
28 endmodule
29

```

```

30 module reg_file_4_1 (input wire clk, reset, wr, input wire [1:0] rd_addr_a, rd_addr_b, wr_addr, input wire d_in, output wire d_out_a, d_out_b);
31   wire wr0, wr1, o0_a, o0_b, o1_a, o1_b;
32   reg_file_2_1 reg_file_2_1_0 (clk, reset, wr0, rd_addr_a[0], rd_addr_b[0], wr_addr[0],
33     d_in, o0_a, o0_b);
34   reg_file_2_1 reg_file_2_1_1 (clk, reset, wr1, rd_addr_a[0], rd_addr_b[0], wr_addr[0],
35     d_in, o1_a, o1_b);
36   mux2 mux2_a (o0_a, o1_a, rd_addr_a[1], d_out_a);
37   mux2 mux2_b (o0_b, o1_b, rd_addr_b[1], d_out_b);
38   demux2 demux2_0 (wr, wr_addr[1], wr0, wr1);
39 endmodule
40
41 module reg_file_8_1 (input wire clk, reset, wr, input wire [2:0] rd_addr_a, rd_addr_b, wr_addr, input wire d_in, output wire d_out_a, d_out_b);
42   wire wr0, wr1, o0_a, o0_b, o1_a, o1_b;
43   _reg_file_4_1 reg_file_4_1_0 (clk, reset, wr0, rd_addr_a[1:0], rd_addr_b[1:0], wr_addr[1:0],
44     d_in, o0_a, o0_b);
45   reg_file_4_1 reg_file_4_1_1 (clk, reset, wr1, rd_addr_a[1:0], rd_addr_b[1:0], wr_addr[1:0],
46     d_in, o1_a, o1_b);
47   mux2 mux2_a (o0_a, o1_a, rd_addr_a[2], d_out_a);
48   mux2 mux2_b (o0_b, o1_b, rd_addr_b[2], d_out_b);
49   demux2 demux2_0 (wr, wr_addr[2], wr0, wr1);
50 endmodule
51
52 module reg_file_8_4 (input wire clk, reset, wr, input wire [2:0] rd_addr_a, rd_addr_b, wr_addr, input wire [3:0] d_in, output wire [3:0] d_out_a, d_out_b);
53   reg_file_8_1 reg_file_8_1_0 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
54     d_in[0], d_out_a[0], d_out_b[0]);
55   reg_file_8_1 reg_file_8_1_1 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
56     d_in[1], d_out_a[1], d_out_b[1]);
57   reg_file_8_1 reg_file_8_1_2 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
58     d_in[2], d_out_a[2], d_out_b[2]);
59   reg_file_8_1 reg_file_8_1_3 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
60     d_in[3], d_out_a[3], d_out_b[3]);
61 endmodule
62

```

```

63 module reg_file (input wire clk, reset, wr, input wire [2:0] rd_addr_a, rd_addr_b, wr_addr, input wire [15:0] d_in, output wire [15:0] d_out_a, d_out_b);
64   reg_file_8_4 reg_file_8_4_0 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
65     d_in[3:0], d_out_a[3:0], d_out_b[3:0]);
66   reg_file_8_4 reg_file_8_4_1 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
67     d_in[7:4], d_out_a[7:4], d_out_b[7:4]);
68   reg_file_8_4 reg_file_8_4_2 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
69     d_in[11:8], d_out_a[11:8], d_out_b[11:8]);
70   reg_file_8_4 reg_file_8_4_3 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
71     d_in[15:12], d_out_a[15:12], d_out_b[15:12]);
72 endmodule
73
74 module mux2_4 (input wire [3:0] i0, i1, input wire j, output wire [3:0] o);
75   mux2 mux2_0 (i0[0], i1[0], j, o[0]);
76   mux2 mux2_1 (i0[1], i1[1], j, o[1]);
77   mux2 mux2_2 (i0[2], i1[2], j, o[2]);
78   mux2 mux2_3 (i0[3], i1[3], j, o[3]);
79 endmodule
80
81 module mux2_16 (input wire [15:0] i0, i1, input wire j, output wire [15:0] o);
82   mux2_4 mux2_4_0 (i0[3:0], i1[3:0], j, o[3:0]);
83   mux2_4 mux2_4_1 (i0[7:4], i1[7:4], j, o[7:4]);
84   mux2_4 mux2_4_2 (i0[11:8], i1[11:8], j, o[11:8]);
85   mux2_4 mux2_4_3 (i0[15:12], i1[15:12], j, o[15:12]);
86 endmodule
87
88 module reg_alu (input wire clk, reset, sel, wr, input wire [1:0] op, input wire [2:0] rd_addr_a,
89   rd_addr_b, wr_addr, input wire [15:0] d_in, output wire [15:0] d_out_a, d_out_b, output wire cout);
90   wire [15:0] d_in_alu, d_in_reg; wire cout_0;
91   alu alu_0 (op, d_out_a, d_out_b, d_in_alu, cout_0);
92   reg_file reg_file_0 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr, d_in_reg, d_out_a, d_out_b);
93   mux2_16 mux2_16_0 (d_in, d_in_alu, sel, d_in_reg);
94   dfr dfr_0 (clk, reset, cout_0, cout);
95 endmodule

```

```

1| timescale 1 ns / 100 ps
2
3`define TESTVECS 4
4
5
6module tb;
7
8    reg clk, reset;
9
10    integer i;
11
12    initial begin
13        $dumpfile("tb_mproc_mem.vcd");
14        $dumpvars(0,tb);
15    end
16
17    initial
18    begin
19        reset = 1'b1;
20        #12.5 reset = 1'b0;
21    end
22
23    initial clk = 1'b0;
24    always #5 clk =~ clk;
25
26    mproc_mem mproc_mem_0 (clk, reset);
27
28    initial begin
29
30        #6 for(i=0;i<`TESTVECS;i=i+1)
31
32            begin #10; end
33
34        #100 $finish;
35
36    end
37endmodule

```

Output waveform



