```
1   /*
2   * Ferhat Can ATAMAN 2030112
3   * 17/04/2018 Tuesday
4   */
5
6   module controller
7      (  input wire  clk,reset,
8         input wire  [31:0] Instr,
9         input wire  CO, N, Z, OVF,
10        output wire WriteEnable,
11        output reg  PCsrc, shiftLR, RegWrite, ImmSrc,
12        output wire [1:0] MemtoReg,
13        output wire ALUSrcB, RegSrc,
14        output reg  [2:0] ALUControl,
15        output wire [3:0] Flags);
16
17
18     reg  NoWrite, FlagWrite;
19     wire RegW, ALUOp;
20     reg  [6:0] controls;
21     wire [5:0] Funct;
22     wire [1:0] Op;
23     wire [3:0] Rd;
24     wire [3:0] ALUFlags;
25
26     assign ALUFlags   = {N,Z,CO,OVF};
27     assign Funct      = Instr[25:20];
28     assign Op         = Instr[27:26];
29     assign Rd         = Instr[15:12];
30
31     simpleregWE #(2) flagreg1(clk, reset, ALUFlags[3:2], FlagWrite, Flags[3:2]);
32     simpleregWE #(2) flagreg0(clk, reset, ALUFlags[1:0], FlagWrite, Flags[1:0]);
33
34     //Main Decoder
35     always @(*)
36     begin
37        case(Op)
38           2'b00:
39              begin
40                 if(Instr[25] == 1'b1) // LSL,LSR
41                    controls = 8'b10110101; //Data processing instructions with no immediate
   source
42                 else //ADD,SUB,AND,ORR,CMP
43                    controls = 8'bx0001101;
44
45              end
46           2'b01:
47              begin
48                 if(Funct[0] == 1'b1)
49                    controls = 8'b0x100100; //LDR instruction
50                 else
51                    controls = 8'b011xx010; //STR instruction
52              end
53           default: controls = 8'bx;
54
55        endcase
56     end
57
58     assign {ImmSrc, RegSrc, ALUSrcB, MemtoReg, RegW, WriteEnable, ALUOp} = controls;
59     assign RegWrite = RegW & (~NoWrite);
60
61     //ALU decoder
62     always @(*)
63     begin
64        if(ALUOp == 1'b1)
65        begin
66           case(Funct[4:1])
67              4'b0100: begin //ADD
68                 ALUControl  = 3'b000;
69                 NoWrite     = 0;
70                 FlagWrite   = 1;
71                 end
72              4'b0010: begin //SUB
73                 ALUControl  = 3'b001;
74                 NoWrite     = 0;
75                 FlagWrite   = 1;
```

```
76                end
77                4'b0000: begin //AND
78                   ALUControl  = 3'b100;
79                   NoWrite     = 0;
80                   FlagWrite   = 1;
81                end
82                4'b1100: begin //ORR
83                   ALUControl  = 3'b101;
84                   NoWrite     = 0;
85                   FlagWrite   = 1;
86                end
87                4'b1010: begin //CMP
88                   ALUControl  = 3'b001;
89                   NoWrite     = 1;
90                   FlagWrite   = 1;
91                end
92                4'b1101: begin //LSL,LSR
93                   ALUControl  = 3'b000;
94                   NoWrite     = 0;
95                   FlagWrite   = 0;
96                   if(Instr[6:5] == 2'b00) //LSL
97                      shiftLR = 0;
98                   else if(Instr[6:5] == 2'b01) //LSR
99                      shiftLR = 1'b1;
100               end
101               default: begin
102                  ALUControl = 3'bx;
103                  NoWrite    = 1'bx;
104                  FlagWrite  = 1'bx;
105               end
106            endcase
107         end else begin
108            ALUControl  = 3'b000; //Non DP Instuctions
109            FlagWrite   = 0; // Don't Update Flags
110            NoWrite     = 0;
111         end
112      end
113
114      //PC Logic
115      always @* begin
116         PCsrc = (Rd == 4'b1111) & RegW;
117      end
118
119   endmodule
```