Digital Computer Design Laboratory

#7 INSTRUCTION EXECUTION

1. In this laboratory course, you will be creating a structural instruction separation and execution. An instruction has two-bit size instruction, four-bit size write register address, four-bit size read register address and another four-bit size second read register address as an input.

2 bit	4 bit	4 bit	4 bit
Ins.	REG_{WRT}	REG_1	REG_2

Your design should separate each instruction into the instruction and addresses. These values will be used as input to registerFile and ALU modules. The output of the ALU should be written into the register addressed by REG_{WRT}

Create *instruction.sv* with structural design and test your design with *register_instruction.sv* .

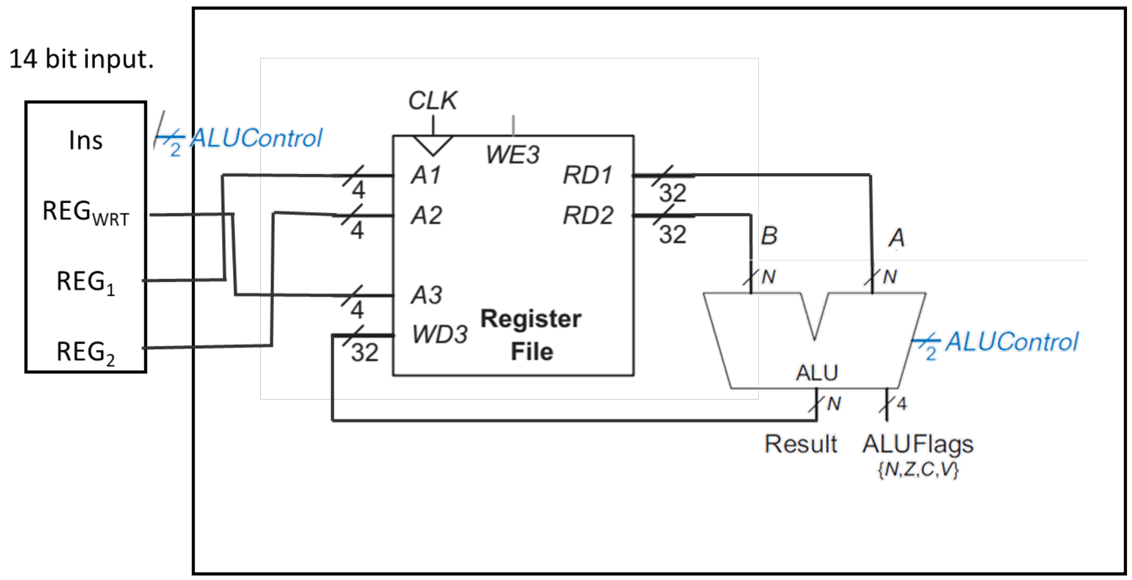

Figure-2

```
module registerFile(input logic [1:0] A1, A2,A3,
            input logic WE,
            input logic clk,
            input logic [3:0] WD3,
            output reg [3:0] RD1,RD2 );
            reg [3:0] rf [3:0];

            always@(posedge clk )
            if(WE)
               rf[A3]<=WD3;
            always@(posedge clk )
            RD1 <= rf[A1];
            always@(posedge clk )
            RD2 <= rf[A2];

endmodule

module ALU(input [31:0] A, B,
input [1:0] ALUControl,
output [31:0]result,
output V, C, N, Z
    );
    wire [31:0] n1,n2,n3,n4,nb;
    wire n5,n6,cout;
    assign nb=~B;
    assign n3=A&B;
    assign n4=A|B;
    assign C= (~ALUControl[1])&cout;
    assign n5=n2[31]^A[31];
    assign n6=~(ALUControl[0] ^ A[31] ^ B[31]);
    assign V= (~ALUControl[1]) & n5 & n6;
    assign Z= &(~result);
    assign N=result[31];
    mux2 first_mux (B,nb,ALUControl[0],n1);
    NbitFulladder firs_adder (A,n1,ALUControl[0],n2,cout);
    mux4 secound_mux (n2,n2,n3,n4,ALUControl,result);
endmodule

module mux2(input logic [31:0] d0, d1,
input logic s,
output logic [31:0] y);

            assign y = s ? d1 : d0;

endmodule
```

```
module NbitFulladder(input logic [31:0] a, b,
input logic cin,
output logic [31:0] s,
output logic cout  );
wire [32:0] c;

assign c=a+b;
assign cout=c[32];
assign s=c[31:0];

endmodule


module mux4(input logic [31:0] d0, d1, d2, d3,
input logic [1:0] s,
output logic [31:0] y);

        assign y = s[1] ?
        (s[0] ? d3 : d2):
        (s[0] ? d1 : d0);

endmodule
```