Practice 1: Control

Code

```
module Controller(
input [6:0] Inst,
output reg Branch,
output reg [1:0] ALUOp,
output reg ALUSrc,
output reg MemRead,
output reg MemWrite,
output reg MemtoReg,
output reg RegWrite
    );

always @*
begin
    case(Inst)
        7'b0110011: // R-type
        begin
            Branch = 1'b0;
            ALUOp = 2'b10;
            ALUSrc = 1'b0;
            MemRead = 1'b0;
            MemWrite = 1'b0;
            MemtoReg = 1'b0;
            RegWrite = 1'b1;
        end
        7'b0010011: // I-type
        begin
            Branch = 1'b0;
            ALUOp = 2'b00;
            ALUSrc = 1'b1;
            MemRead = 1'b0;
            MemWrite = 1'b0;
            MemtoReg = 1'b0;
            RegWrite = 1'b1;
        end
        7'b0000011: // I-type
        begin
            Branch = 1'b0;
            ALUOp = 2'b00;
            ALUSrc = 1'b1;
            MemRead = 1'b1;
            MemWrite = 1'b0;
            MemtoReg = 1'b1;
            RegWrite = 1'b1;
        end
        7'b0100011: // S-type
```

```
        begin
            Branch = 1'b0;
            ALUOp = 2'b00;
            ALUSrc = 1'b1;
            MemRead = 1'b0;
            MemWrite = 1'b1;
            MemtoReg = 1'b0;
            RegWrite = 1'b0;
        end
        7'b1100011: // B-type
        begin
            Branch = 1'b1;
            ALUOp = 2'b01;
            ALUSrc = 1'b0;
            MemRead = 1'b0;
            MemWrite = 1'b0;
            MemtoReg = 1'b0;
            RegWrite = 1'b0;
        end
        default:
        begin
            Branch = 1'b0;
            ALUOp = 2'b00;
            ALUSrc = 1'b0;
            MemRead = 1'b0;
            MemWrite = 1'b0;
            MemtoReg = 1'b0;
            RegWrite = 1'b0;
        end
    endcase
end

endmodule
```

Testbench

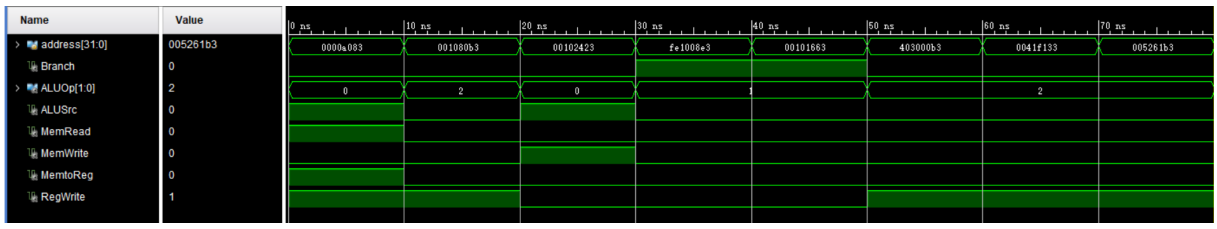

Figure 1: Controller Testbench

The instructions are from lab slides: `lw`, `add`, `sw`, `beq`, `bne`, `sub`, `and`, `or`.

ALU

Code

```
module ALU(
input [31:0] Inst,
input [31:0] Read_data_1,
input [31:0] Read_data_2,
input [31:0] Imme,
input [1:0] ALUOp,
input ALUSrc,
output reg [31:0] ALU_result,
output Zero
    );

wire [3:0] ALUControl;
wire [31:0] Operand2;

ALU_Control ALU_Control(
.ALUOp(ALUOp),
.Inst(Inst),
.ALUControl(ALUControl)
);

assign Operand2 = ALUSrc ? Imme : Read_data_2;
assign Zero = ALU_result == 0;

always @* begin
    case(ALUControl)
        4'b0000:
        begin
            ALU_result = Read_data_1 & Operand2;
        end
        4'b0001:
        begin
            ALU_result = Read_data_1 | Operand2;
        end
        4'b0010:
        begin
            ALU_result = Read_data_1 + Operand2;
        end
        4'b0110:
        begin
            ALU_result = Read_data_1 - Operand2;
        end
    endcase
end
endmodule
```

```
module ALU_Control(
input [1:0] ALUOp,
input [31:0] Inst,
output reg [3:0] ALUControl
    );

wire [6:0] funct7;
wire [2:0] funct3;
assign funct7 = Inst[31:25];
assign funct3 = Inst[14:12];

always @* begin
    case(ALUOp)
        2'b00:
        begin
            ALUControl = 4'b0010;
        end
        2'b01:
        begin
            ALUControl = 4'b0110;
        end
        2'b10:
        begin
            case(funct7)
                7'b0000000:
                begin
                    case(funct3)
                        3'b000:
                        begin
                            ALUControl = 4'b0010;
                        end
                        3'b111:
                        begin
                            ALUControl = 4'b0000;
                        end
                        3'b110:
                        begin
                            ALUControl = 4'b0001;
                        end
                    endcase
                end
                7'b0100000:
                begin
                    ALUControl = 4'b0110;
                end
            endcase
        end
    endcase
```

```
end
endmodule
```

Testbench

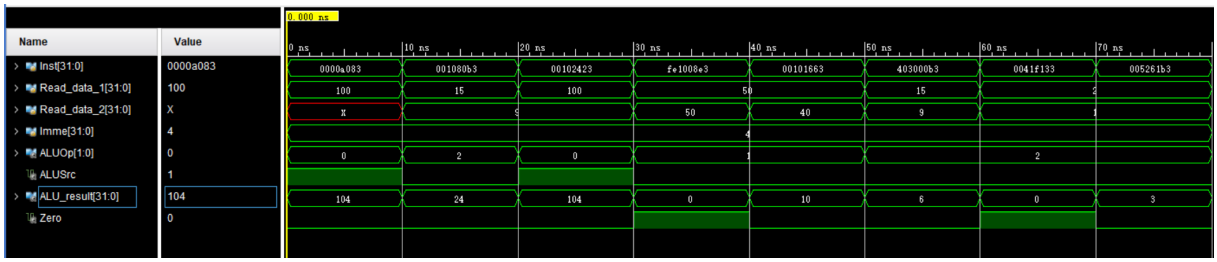

Figure 2: ALU Testbench

The instructions are the same as the previous testbench.