Computer Organization LAB3 Report

109550168 林慧旻

1. **Implementation**
   a. **Decoder.v**

```
/* Write your code HERE */
assign opcode = instr_i[6:0];
assign funct3 = instr_i[14:12];
assign ALUSrc = 1'b1;
assign RegWrite = 1'b1;
assign Branch = 1'b0;
assign ALUOp = 2'b10;
endmodule
```

Because in this homework we only need to build r-type instruction, I only need to assign ALUSrc, RegWrite to 1, and assign Branch and ALUOp to 0. If there are other instruction like beq, then I need to assign these wires according to different conditions. As for opcode and funct3, they only need to be assigned to the value within the bit range in the 32-bit instruction.

   b. **Adder.v**

```
/* Write your code HERE */
    assign sum_o = src1_i + src2_i;
```

Assign the output wire sum_o to be src1_i plus src2_i. This is a simple adder.

   c. **alu.v**

This one is made exactly as the one in HW2, so I am not going to describe the same thing here. The only different part is as follows:

```
always@(*)begin
    if(rst_n==1'b1) begin
        if(ALU_control == 4'b0100)begin
            result = src1 ^ src2;
            zero = 1'b0;
            cout = 1'b0;//?
            overflow = 1'b0;//?
        end
        else if(ALU_control == 4'b0101)begin
            result = src1 << src2;
            zero = 1'b0;
            cout = 1'b0;//?
            overflow = 1'b0;//?
        end
        else if(ALU_control == 4'b0011)begin
            result = src1 >>> src2;
            zero = 1'b0;
            cout = 1'b0;//?
            overflow = 1'b0;//?
        end
        else begin
            result = ans;
            zero = ~(|result);
            cout = Cin[31] & (ope == 2'd2);
            overflow = (Cin[31] ^ Cin[30]);
        end
    end
    else begin
        result = 32'b0;
        zero = 1'b0;
        cout = 1'b0;
        overflow = 1'b0;
    end
end
```

When it confronts to the shift left logically, shift right arithmetically, and exclusive-or instructions, it will do the instructions by simple operator, so that it won't compute the value incorrectly from the loop above.

To build the alu.v, I also put alu_1bit.v, MUX4to1.v and MUX2to1.v into Lab3_Code file.

d. ALU_Ctrl.v

```
/* Write your code HERE */
always@(*)begin
    if(ALUOp == 2'b10 && instr == 4'b0000)//add//2
        ALU_Ctrl_o = 4'b0010;
    else if(ALUOp == 2'b10 && instr == 4'b1000)//sub//6
        ALU_Ctrl_o = 4'b0110;
    else if(ALUOp == 2'b10 && instr == 4'b0111)//and//0
        ALU_Ctrl_o = 4'b0000;
    else if(ALUOp == 2'b10 && instr == 4'b0110)//or//1
        ALU_Ctrl_o = 4'b0001;
    else if(ALUOp == 2'b10 && instr == 4'b0100)//exclusive_or//4
        ALU_Ctrl_o = 4'b0100;
    else if(ALUOp == 2'b10 && instr == 4'b0010)//less//7
        ALU_Ctrl_o = 4'b0111;
    else if(ALUOp == 2'b10 && instr == 4'b0001)//shift_left//5
        ALU_Ctrl_o = 4'b0101;
    else if(ALUOp == 2'b10 && instr == 4'b1101)//shift_right//3
        ALU_Ctrl_o = 4'b0011;
end
```

In here I need to decide the value of ALU_Ctrl in different instructions.

I let it first recognize what type of operation is it from ALUOp and instr. The instr is assigneed as follows in Simple_Single_cycle.v:

```
wire [3:0]alu_instr;
assign alu_instr[2:0] = instr[14:12];
assign alu_instr[3] = instr[30];
```

The input is made up by the [14:12] and [30] bit in the 32-bit instruction.

The add, subtract, and, or and less operations have the ALU_Ctrl_o following alu.v's design in HW2. As for the other three operations, I set them to the random values that doesn't repeat the values above.

e. Simple_Single_CPU.v

```
wire [31:0] pc_i;
wire [31:0] pc_o;
wire [31:0] instr;
wire [31:0] ALUresult;
wire RegWrite;
wire [31:0] RSdata_o;
wire [31:0] RTdata_o;
wire ALUSrc;
wire [1:0] ALUOp;
wire [3:0]ALU_control;
wire zero,cout,overflow;
wire [31:0]imm_4 = 4;
wire branch;

wire [3:0]alu_instr;
assign alu_instr[2:0] = instr[14:12];
assign alu_instr[3] = instr[30];
```

Here I have all the wires ready to connect the different parts off this CPU.

```
ProgramCounter PC(
        .clk_i(clk_i),
        .rst_i(rst_i),
        .pc_i(pc_i),
        .pc_o(pc_o)
        );
```

In program counter, let clock, the negative reset, pc input be the input wires. Let pc output be the output, so it can change update pc's value in every clock cycle.

```
Instr_Memory IM(
        .addr_i(pc_o),
        .instr_o(instr)
        );
```

Here it recognizes what is the pc's value now, and acquires instructions' value.

```
Reg_File RF(
        .clk_i(clk_i),
        .rst_i(rst_i),
        .RSaddr_i(instr[19:15]),
        .RTaddr_i(instr[24:20]),
        .RDaddr_i(instr[11:7]),
        .RDdata_i(ALUresult),
        .RegWrite_i(RegWrite),
        .RSdata_o(RSdata_o),
        .RTdata_o(RTdata_o)
        );
```

Here I give the address of the two sources and the rd,so that in Reg_File, it can get the values in the addresses for the later alu's computation.

```
Decoder Decoder(
        .instr_i(instr),
        .ALUSrc(ALUSrc),
        .RegWrite(RegWrite),
        .Branch(branch),
        .ALUOp(ALUOp)
        );
```

Let instr be the input here, so it can output the ALUSrc, RegWrite, Branch, and ALUOp's values. They decide the later operations.

```
Adder PC_plus_4_Adder(
        .src1_i(pc_o),
        .src2_i(32'd4),
        .sum_o(pc_i)
        );
```

Here I let pc_o plus 4 every cycle and assign it to pc_i.

```
ALU_Ctrl ALU_Ctrl(
        .instr(alu_instr),
        .ALUOp(ALUOp),
        .ALU_Ctrl_o(ALU_control)
        );
```

ALU_Ctrl reads instr and ALUOp, and then outputs the ALU_Ctrl_o to alu.

```
alu alu(
        .rst_n(rst_i),
        .src1(RSdata_o),
        .src2(RTdata_o),
        .ALU_control(ALU_control),
        .result(ALUresult),
        .zero(zero),
        .cout(cout),
        .overflow(overflow)
        );
```

Finally, alu is incharge of dealing with sources and alu control which are some of the outputs in the modules above. It then output the result value.

2. Problem and Solution

The homework is not difficult for me, so I have not many questions about it. The only two one that may cause a little confusion is as follows:

(a) I am not familiar to Ubuntu, and I don't have one. When checking the correctness of my homework, I send the .v files to one of my classmates who use Linux operating system. I really want to figure out how to run .sh files in window. I tried several ways but none of them work out.

(b) I was confused by Rt and Rs, I wasn't sure which one is rs1 and which one is rs2 before asking ta on Hackmd.