实验报告

2021 年 5 月 28 日　　成绩：_____

姓名	汪凌峰	学号	19270824	班级	19052321
专业	计算机科学与技术		课程名称	计算机组成原理课程设计	
任课老师	曾虹	指导老师	曾虹	机位号	31
实验序号	8	实验名称	实现 R 型指令的 CPU 设计实验		
实验时间	2021.5.28	实验地点	一教 225	实验设备号	31

一、实验程序源代码

```
`timescale 1ns / 1ps

module R_CPU(
    input clk,
    input rst,
    output [31:0]Inst_code,
    output reg [31:0]PC,
    output [5:0]opcode,
    output [4:0]rs,
    output [4:0]rt,
    output [4:0]rd,
    output [5:0]func,
    output [31:0] ALU_F,
    output ZF,
    output OF,
    output reg [2:0] ALU_OP
   );

   initial PC = 32'h0000_0000;

   ROM_B Inst (
     .clka(clk), // input clka
     .addra(PC[7:2]), // input [5 : 0] addra
     .douta(Inst_code) // output [31 : 0] douta
   );

   reg Write_Reg;
   wire [31:0]PC_new;
   wire [31:0]R_Data_A;
```

```
    wire [31:0]R_Data_B;

    assign PC_new = PC + 4;

    assign opcode =  Inst_code[31:26];
    assign rs =  Inst_code[25:21];
    assign rt =  Inst_code[20:16];
    assign rd=  Inst_code[15:11];
    assign func =  Inst_code[5:0];

    always@(negedge clk or posedge rst)
    begin
        if(rst)
            PC = 32'h0000_0000;
        else
            PC = PC_new;
    end

    always @(*)
    begin
        ALU_OP = 3'b000;
        Write_Reg = 1'b0;

        if (opcode==6'b000000)    //R 指令

        begin

            Write_Reg = 1'b1;   //结果送寄存器

            case (func)
                6'b100000:begin ALU_OP=3'b100; end
                6'b100010:begin ALU_OP=3'b101; end
                6'b100100:begin ALU_OP=3'b000; end
                6'b100101:begin ALU_OP=3'b001; end
                6'b100110:begin ALU_OP=3'b010; end
                6'b100111:begin ALU_OP=3'b011; end
                6'b101011:begin ALU_OP=3'b110; end
                6'b000100:begin ALU_OP=3'b111; end
            endcase
        end
    end

    ALU ALU_1(ALU_OP,R_Data_A,R_Data_B,ALU_F,ZF,OF);
    REGS REGS_1(~clk,rst,Write_Reg,rs,rt,rd,ALU_F,R_Data_A,R_Data_B);
Endmodule
```

```
`timescale 1ns / 1ps

module ALU(
    input [2:0] OP,
    input [31:0] A,
    input [31:0] B,
    output reg [31:0] F,
    output reg ZF, //零值

    output reg OF //溢出
    );
    reg CF;
    always@(*)
    begin
        CF = 0;
        case(OP)
            3'b000: begin F = A & B; end
            3'b001: begin F = A | B; end
            3'b010: begin F = A ^ B; end
            3'b011: begin F = ~(A | B); end
            3'b100: begin {CF, F} = A + B; end
            3'b101: begin {CF, F} = A - B; end
            3'b110: begin F = A < B; end
            3'b111: begin F = B << A; end
        endcase
        ZF = F == 0;
        OF = A[31] ^ B[31] ^ F[31] ^ CF;
    end
endmodule

`timescale 1ns / 1ps

module REGS(
     input CLK,
     input Reset,
     input Write_Reg,
     input [4:0] R_Addr_A,
     input [4:0] R_Addr_B,
     input [4:0] W_Addr,
     input [31:0] W_Data,
     output [31:0] R_Data_A,
     output [31:0] R_Data_B
    );
```

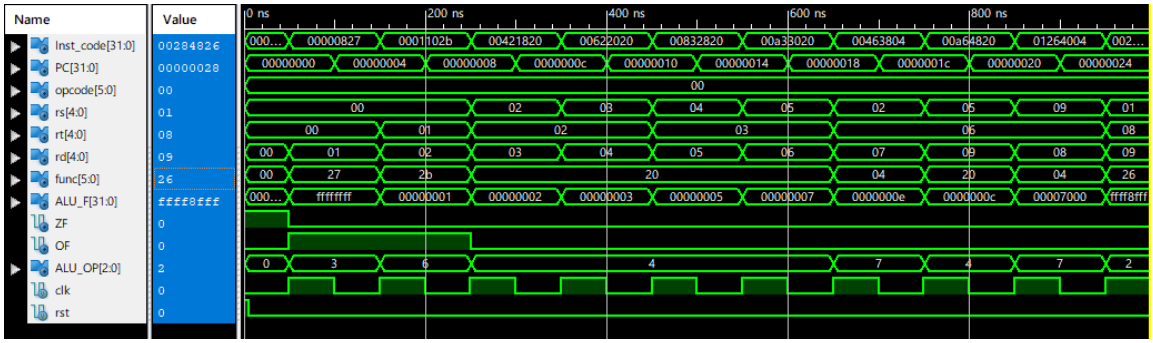
```
    integer i;
    reg [31:0]REG_Files[0:31];
    initial REG_Files[0]=32'h00114514;
    initial REG_Files[1]=32'h01919810;
    initial REG_Files[2]=32'h23333333;

    assign R_Data_A = REG_Files[R_Addr_A];
    assign R_Data_B = REG_Files[R_Addr_B];

    always@(posedge CLK or posedge Reset)
    begin
        if(Reset)
            for(i = 0; i < 32; i = i + 1)
                REG_Files[i] = 0;
        else
            if (Write_Reg)
                REG_Files[W_Addr] = W_Data;
    end
endmodule
```

二、仿真测试代码

```
`timescale 1ns / 1ps

module TestR_CPU;

    // Inputs
    reg clk;
    reg rst;

    // Outputs
    wire [31:0] Inst_code;
    wire [31:0] PC;
    wire [5:0] opcode;
    wire [4:0] rs;
    wire [4:0] rt;
    wire [4:0] rd;
    wire [5:0] func;
    wire [31:0] ALU_F;
    wire ZF;
    wire OF;
    wire [2:0] ALU_OP;

    // Instantiate the Unit Under Test (UUT)
    R_CPU uut (
```

```
        .clk(clk),
        .rst(rst),
        .Inst_code(Inst_code),
        .PC(PC),
        .opcode(opcode),
        .rs(rs),
        .rt(rt),
        .rd(rd),
        .func(func),
        .ALU_F(ALU_F),
        .ZF(ZF),
        .OF(OF),
        .ALU_OP(ALU_OP)
    );

    initial begin
        // Initialize Inputs
        clk = 0;
        rst = 1;
        #5;
        rst=0;
    end
    always #50 clk=~clk;

endmodule
```

三、仿真波形



四、思考与探索

有了 IP 核之后干什么都很方便，但是这次做 R 型的时候偷懒了，少写了很多控制信号，下次做 R-I 型的时候再补上去吧。这次主要是只实现了 R 型指令，主要是对 ALU 的操作。对于拓展，位移运算的移位位数需要依靠 shamt 那一项，我在这次实验中没有实现，在 RI 型设计实验中我再尝试实现它吧。