取指令通路实验报告

2021 年 4 月 26 日　　成绩：_____

姓名	周俊佐	学号	19071337	班级	19185312
姓名	颜伟鹏	学号	19071334	班级	19185312
姓名	高炜哲	学号	18081811	班级	18181411
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	章复嘉	指导老师	章复嘉	机位号	95
实验序号	7	实验名称	取指令数据通路设计实验		

一、实验目的和实验要求

1. 学习指令存储器的设计方法;

2. 掌握 ARM V7 CPU 取指令操作和判断指令条件执行的方法;

3. 用 Memory IP 核生成一个只读的指令存储器，并关联一个 test.coe 文件;

4. 编程实现取指令模块，调用指令存储器 IP 核;

5. 编程实现条件判断，条件符合时将取出来的指令写入 IR 中，否则不改写 IR 寄存器。同时完成 PC 自增;完成仿真调试和板级调试;

6. 撰写实验报告。

二、实验程序源代码

顶层模块

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
module top(
    // input
    input clk,
    input Rst,
    input Write_PC,
    input Write_IR,
    input [3:0] NZCV, // from CSPR[32:28]
    // output
    output flag,
    output [7:2] Inst_Addr,
    output  [31:28] Inst_condition,
    output [27:0] Inst_left,
    output [31:0] IR
);

wire [31:0] PC;
wire [31:0] Inst;
assign Inst_Addr = PC[7:2];
assign Inst_condition = Inst[31:28];
assign Inst_left = Inst[27:0];

// PC
PC PC_Instance(
    // input
    .clk(clk),
    .Rst(Rst),
    .Write_PC(Write_PC),
    // output
    .PC(PC)
);

// Inst_Rom
Inst_Rom Inst_Rom_Instance(
    // input
    .clka(clk),
    .addra(Inst_Addr),
    // output
    .douta(Inst)
);
```

```
// Inst_Reg
Inst_Reg Inst_Reg_Instance(
    // input
    .clk(clk),
    .Rst(Rst),
    .NZCV(NZCV),
    .Write_IR(Write_IR),
    .Inst(Inst),
    // output
    .flag(flag),
    .IR(IR)
);
endmodule
```

PC 模块

```
module PC(
    input clk,
    input Rst,
    input Write_PC,
    output reg [31:0] PC
);
    wire [31:0] PC_New;
    assign PC_New = PC + 4;
    always@(negedge clk or posedge Rst ) begin
        if(Rst) begin
            PC <= 0;
        end
        else if(Write_PC) begin
            PC <= PC_New;
        end
    end
Endmodule
```

指令寄存器模块

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
module Inst_Reg(
    // input
    input clk,
    input Rst,
```

```
    input [3:0] NZCV, // from CSPR
    input [31:0] Inst, // Instruction from Inst_Rom
    input Write_IR,
    //output
    output reg flag,
    output reg [31:0] IR
    );

    assign VF = NZCV[0];
    assign CF = NZCV[1];
    assign ZF = NZCV[2];
    assign NF = NZCV[3];

    always@(negedge clk or posedge Rst) begin
        // Clear
        if(Rst) begin
            IR <= 32'h00000000;
        end
        // Write
        else begin
            if(Write_IR) begin
                case(Inst[31:28])
                    4'b0000: begin //EQ
                        if(ZF == 1) begin IR <= Inst;
                        flag <= 1;end
                        else flag <= 0;
                    end
                    4'b0001: begin //NE
                        if(ZF == 0) begin IR <= Inst;
                        flag <= 1;end
                        else flag <= 0;
                    end
                    4'b0010: begin //CS
                        if(CF == 1) begin IR <= Inst;
                        flag <= 1;end
                        else flag <= 0;
                    end
                    4'b0011: begin //CC
                        if(CF == 0) begin IR <= Inst;
                        flag <= 1;end
                        else flag <= 0;
                    end
                    4'b0100: begin //MI
                        if(NF == 1) begin IR <= Inst;
```

```
                flag <= 1;end
                else flag <= 0;
            end
            4'b0101: begin //PL
                if(NF == 0) begin IR <= Inst;
                flag <= 1;end
                else flag <= 0;
            end
            4'b0110: begin //VS
                if(VF == 1) begin IR <= Inst;
                flag <= 1;end
                else flag <= 0;
            end
            4'b0111: begin //VC
                if(VF == 0) begin IR <= Inst;
                flag <= 1;end
                else flag <= 0;
            end
            4'b1000: begin //HI
                if(CF == 1 && ZF == 0) begin IR <= Inst;
                flag <= 1;end
                else flag <= 0;
            end
            4'b1001: begin //LS
                if(CF == 0 && ZF == 1) begin IR <= Inst;
                flag <= 1;end
                else flag <= 0;
            end
            4'b1010: begin //GE
                if(NF == VF) begin IR <= Inst;
                flag <= 1;end
                else flag <= 0;
            end
            4'b1011: begin //LT
                if(NF != VF) begin IR <= Inst;
                flag <= 1;end
                else flag <= 0;
            end
            4'b1100: begin //GT
                if(ZF == 0 && NF == VF) begin IR <= Inst;
                flag <= 1;end
                else flag <= 0;
            end
            4'b1101: begin //LE
```

```
                        if(ZF == 1 && NF != VF) begin IR <= Inst;
                        flag <= 1;end
                        else flag <= 0;
                    end
                    4'b1110: begin //AL
                        IR <= Inst;
                        flag <= 1;
                    end
                    default: begin
                        flag <= 0;
                    end
                endcase
            end
        end
    end
endmodule
```

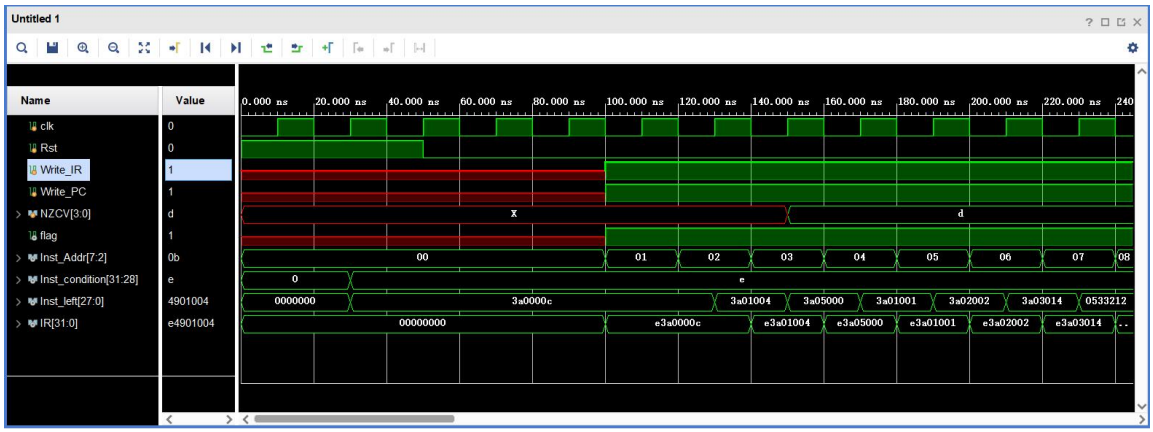
仿真代码

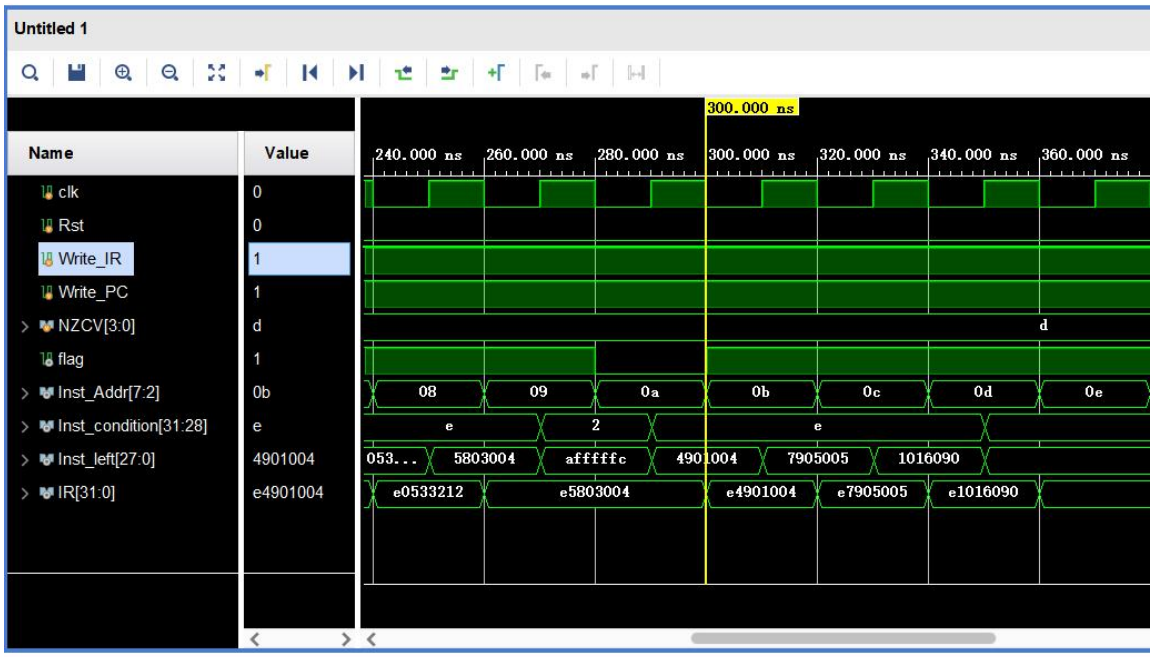
```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
module tb_top_normal();
    // top input
    reg clk = 0;
    reg Rst;
    reg Write_IR;
    reg Write_PC;
    reg [3:0] NZCV;
    // top output
    wire flag;
    wire [7:2] Inst_Addr;
    wire [31:28] Inst_condition;
    wire [27:0] Inst_left;

    integer i;

    always #10 clk = ~clk;
    initial begin
    Rst = 1;
    #50
    Rst = 0;
    #50;
```

```
    Write_IR = 1;
    Write_PC = 1;
    for(i=0; i<15; i=i+1) begin
        NZCV = i;
        #50;
    end
    end

top top_Instance(
    .clk(clk),
    .Rst(Rst),
    .Write_IR(Write_IR),
    .Write_PC(Write_PC),
    .NZCV(NZCV),
    .flag(flag),
    .Inst_Addr(Inst_Addr),
    .Inst_condition(Inst_condition),
    .Inst_left(Inst_left)
);
endmodule



`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
module tb_top_implementation();
    // top input
    reg clk = 0;
    reg Rst;
    reg Write_IR;
    reg Write_PC;
    reg [3:0] NZCV;
    // top output
    wire flag;
    wire [7:2] Inst_Addr;
    wire [31:28] Inst_condition;
    wire [27:0] Inst_left;
    wire [31:0] IR;

    always #10 clk = ~clk;
    initial begin
    Rst = 1;
    #50
```

```
    Rst = 0;
    #50;
    Write_IR = 1;
    Write_PC = 1;
    #50;
    NZCV = 4'b1101;
    end

top top_Instance(
    .clk(clk),
    .Rst(Rst),
    .Write_IR(Write_IR),
    .Write_PC(Write_PC),
    .NZCV(NZCV),
    .flag(flag),
    .Inst_Addr(Inst_Addr),
    .Inst_condition(Inst_condition),
    .Inst_left(Inst_left),
    .IR(IR)
);
endmodule
```

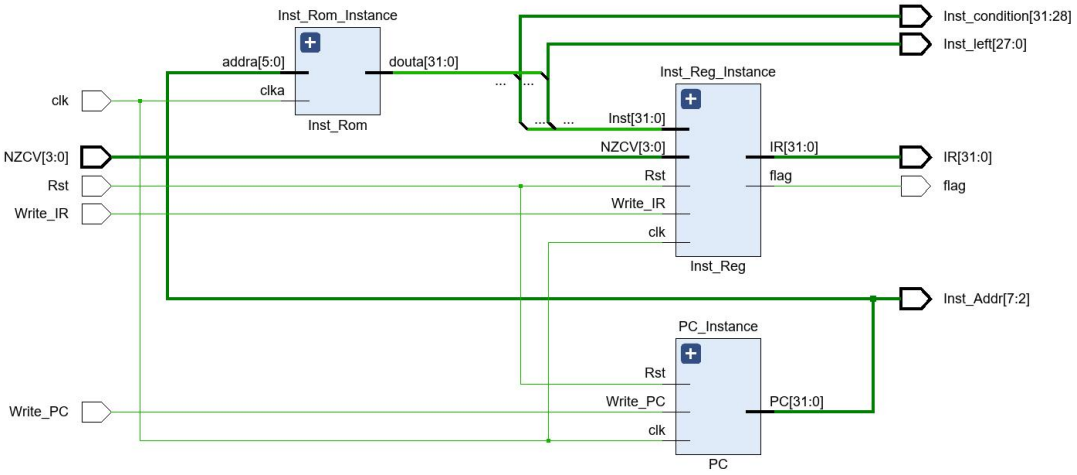
三、仿真文件和波形图（有就填，没有就不填）



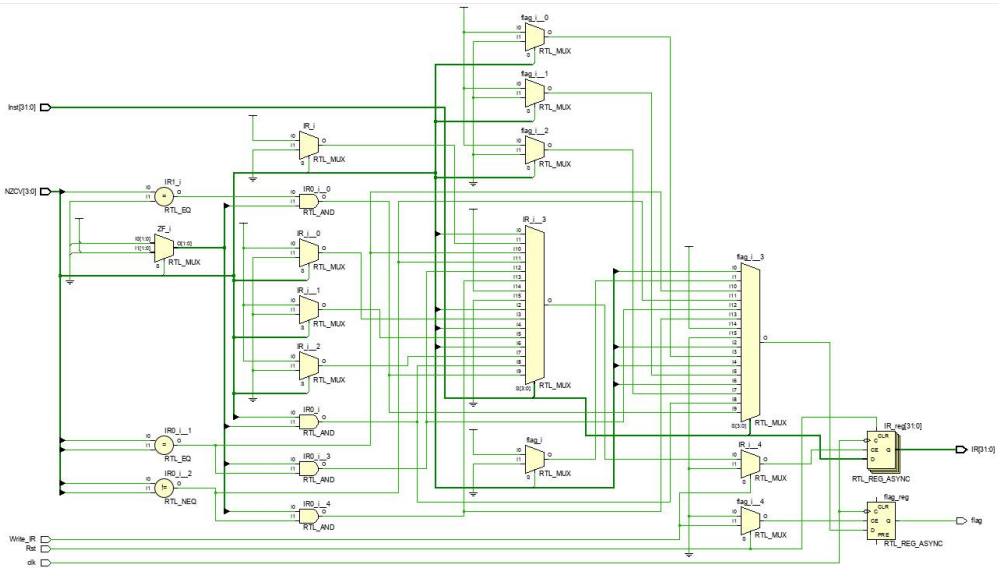


四、电路图（有就填，没有就不填）

该图是顶层模块，可清晰观察指令寄存器、PC 以及 ROM



该图是指令寄存器内部电路图



五、引脚配置（约束文件，有就填，没有就不填））

```
# .bit
set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]

# Switch
set_property PULLDOWN true [get_ports sw]
set_property IOSTANDARD LVCMOS18 [get_ports sw]
set_property PACKAGE_PIN T3  [get_ports {sw[1]}]
set_property PACKAGE_PIN U3  [get_ports {sw[2]}]
set_property PACKAGE_PIN T4  [get_ports {sw[3]}]
set_property PACKAGE_PIN V3  [get_ports {sw[4]}]
set_property PACKAGE_PIN V4  [get_ports {sw[5]}]
set_property PACKAGE_PIN W4  [get_ports {sw[6]}]
set_property PACKAGE_PIN Y4  [get_ports {sw[7]}]
set_property PACKAGE_PIN Y6  [get_ports {sw[8]}]
set_property PACKAGE_PIN W7  [get_ports {sw[9]}]
set_property PACKAGE_PIN Y8  [get_ports {sw[10]}]
set_property PACKAGE_PIN Y7  [get_ports {sw[11]}]
set_property PACKAGE_PIN T1  [get_ports {sw[12]}]
set_property PACKAGE_PIN U1  [get_ports {sw[13]}]
set_property PACKAGE_PIN U2  [get_ports {sw[14]}]
set_property PACKAGE_PIN W1  [get_ports {sw[15]}]
set_property PACKAGE_PIN W2  [get_ports {sw[16]}]
set_property PACKAGE_PIN Y1  [get_ports {sw[17]}]
set_property PACKAGE_PIN AA1 [get_ports {sw[18]}]
set_property PACKAGE_PIN V2  [get_ports {sw[19]}]
set_property PACKAGE_PIN Y2  [get_ports {sw[20]}]
set_property PACKAGE_PIN AB1 [get_ports {sw[21]}]
set_property PACKAGE_PIN AB2 [get_ports {sw[22]}]
set_property PACKAGE_PIN AB3 [get_ports {sw[23]}]
set_property PACKAGE_PIN AB5 [get_ports {sw[24]}]
set_property PACKAGE_PIN AA6 [get_ports {sw[25]}]
set_property PACKAGE_PIN R2  [get_ports {sw[26]}]
set_property PACKAGE_PIN R3  [get_ports {sw[27]}]
set_property PACKAGE_PIN T6  [get_ports {sw[28]}]
set_property PACKAGE_PIN R6  [get_ports {sw[29]}]
set_property PACKAGE_PIN U7  [get_ports {sw[30]}]
set_property PACKAGE_PIN AB7 [get_ports {sw[31]}]
set_property PACKAGE_PIN AB8 [get_ports {sw[32]}]
```

```
# Switch Buttonť
set_property IOSTANDARD LVCMOS18 [get_ports swb]
set_property PACKAGE_PIN R4  [get_ports {swb[1]}]
set_property PACKAGE_PIN AA4 [get_ports {swb[2]}]
```

```
set_property PACKAGE_PIN AB6 [get_ports {swb[3]}]
set_property PACKAGE_PIN T5  [get_ports {swb[4]}]
set_property PACKAGE_PIN V8  [get_ports {swb[5]}]
set_property PACKAGE_PIN AA8 [get_ports {swb[6]}]
```

```
# LED
set_property PULLDOWN true [get_ports led]
set_property IOSTANDARD LVCMOS18 [get_ports led]
set_property PACKAGE_PIN R1 [get_ports {led[1]}]
set_property PACKAGE_PIN P2 [get_ports {led[2]}]
set_property PACKAGE_PIN P1 [get_ports {led[3]}]
set_property PACKAGE_PIN N2 [get_ports {led[4]}]
set_property PACKAGE_PIN M1 [get_ports {led[5]}]
set_property PACKAGE_PIN M2 [get_ports {led[6]}]
set_property PACKAGE_PIN L1 [get_ports {led[7]}]
set_property PACKAGE_PIN J2 [get_ports {led[8]}]
set_property PACKAGE_PIN G1 [get_ports {led[9]}]
set_property PACKAGE_PIN E1 [get_ports {led[10]}]
set_property PACKAGE_PIN D2 [get_ports {led[11]}]
set_property PACKAGE_PIN A1 [get_ports {led[12]}]
set_property PACKAGE_PIN L3 [get_ports {led[13]}]
set_property PACKAGE_PIN G3 [get_ports {led[14]}]
set_property PACKAGE_PIN K4 [get_ports {led[15]}]
set_property PACKAGE_PIN G4 [get_ports {led[16]}]
set_property PACKAGE_PIN K1 [get_ports {led[17]}]
set_property PACKAGE_PIN J1 [get_ports {led[18]}]
set_property PACKAGE_PIN H2 [get_ports {led[19]}]
set_property PACKAGE_PIN G2 [get_ports {led[20]}]
set_property PACKAGE_PIN F1 [get_ports {led[21]}]
set_property PACKAGE_PIN E2 [get_ports {led[22]}]
set_property PACKAGE_PIN D1 [get_ports {led[23]}]
set_property PACKAGE_PIN B1 [get_ports {led[24]}]
set_property PACKAGE_PIN B2 [get_ports {led[25]}]
set_property PACKAGE_PIN N3 [get_ports {led[26]}]
set_property PACKAGE_PIN M3 [get_ports {led[27]}]
set_property PACKAGE_PIN K3 [get_ports {led[28]}]
set_property PACKAGE_PIN H3 [get_ports {led[29]}]
set_property PACKAGE_PIN N4 [get_ports {led[30]}]
set_property PACKAGE_PIN L4 [get_ports {led[31]}]
set_property PACKAGE_PIN J4 [get_ports {led[32]}]
```

```
set_property IOSTANDARD LVCMOS18 [get_ports seg]
set_property PACKAGE_PIN H19 [get_ports {seg[7]}]
set_property PACKAGE_PIN G20 [get_ports {seg[6]}]
```

```
set_property PACKAGE_PIN J22 [get_ports {seg[5]}]
set_property PACKAGE_PIN K22 [get_ports {seg[4]}]
set_property PACKAGE_PIN K21 [get_ports {seg[3]}]
set_property PACKAGE_PIN H20 [get_ports {seg[2]}]
set_property PACKAGE_PIN H22 [get_ports {seg[1]}]
set_property PACKAGE_PIN J21 [get_ports {seg[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports which]
set_property PACKAGE_PIN N22 [get_ports {which[0]}]
set_property PACKAGE_PIN M21 [get_ports {which[1]}]
set_property PACKAGE_PIN M22 [get_ports {which[2]}]
set_property -dict {IOSTANDARD LVCMOS18 PACKAGE_PIN L21} [get_ports enable]
set_property -dict {IOSTANDARD LVCMOS18 PACKAGE_PIN H4} [get_ports clk]
```

```
# [Place 30-574] Poor placement for routing between an IO pin and BUFG.If this
# sub optimal condition is acceptable for this design, you may use the
# CLOCK_DEDICATED_ROUTE constraint in the .xdc file to demote this message to a
# WARNING. However, the use of this override is highly discouraged.
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clk_IBUF]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets swb_IBUF[1]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets swb_IBUF[2]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets swb_IBUF[3]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets swb_IBUF[4]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets swb_IBUF[5]]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets swb_IBUF[6]]
```

六、实验分工（需写清楚每个人负责的工作，以及个人贡献在小组总工作量中的占比）

总计：高炜哲负责 40%的实验，周俊佐和颜伟鹏各负责 30%的实验。
具体分工如下：
高炜哲：代码编写，代码调试，板卡调试，报告修改
周俊佐：错误分析，测试数据，资料分析
颜伟鹏：实验报告，代码仿真测试。

七、实验收获（心得体会等，请以个人为单位分别写）

思考题：

(1) 在复位后，第一次按动 clk 按键，你的程序读出的是哪条指令?

A.按下按钮，在下跳沿 PC 输出第一个地址给 ROM，松开按钮后，在上升沿此时指令寄存器为空，并**未读出指令**。而要在下一个时钟的下降沿，才会读出第一条指令。

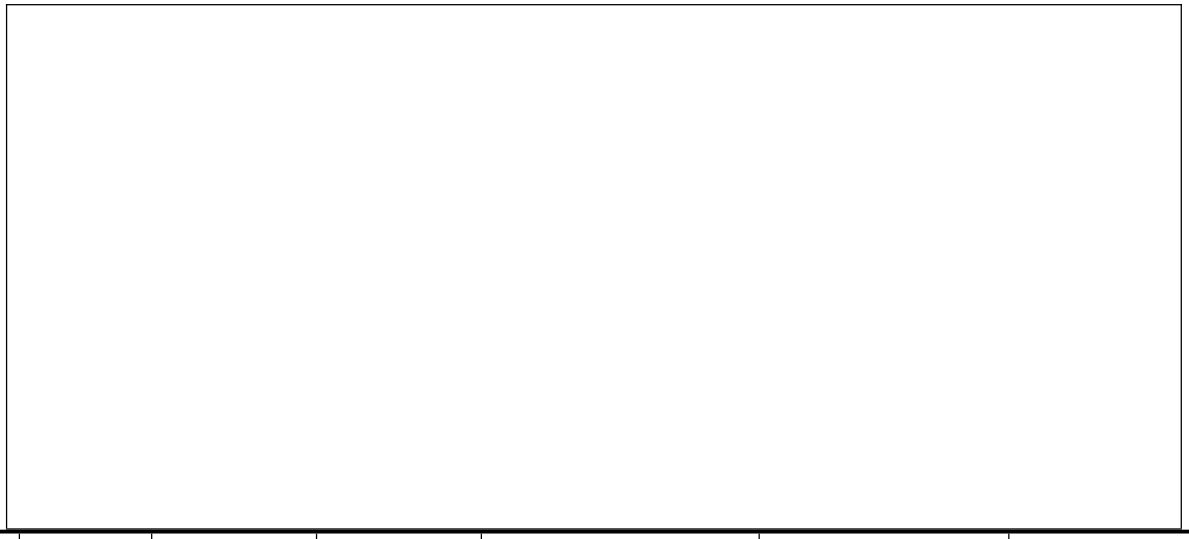
(2) 计算机系统总是保证当前正在执行的机器指令执行完毕才会停机。当指令执行一半时，按动复位信号 Rst，要想保证当前指令执行完毕才停机，应该如何修改 Verilog HDL 程序?请修改程序并调试

将复位按键的执行放进程序的循环体中，则只有当前循环完成后才能开始下一个循环，此时如果由复位标志位为 1，则复位。

(3)假设当复位信号 Rst 有效时，系统在 clk 的下跳沿同步停机。那么，在 clk 低电平时按动复位信号 Rst，将多执行一条指令，你认为应该如何解决这个问题?请编程并调试。
将复位操作编入循环体中，当复位信号为 1 时，clk 上升沿不进行操作；则若 rst 在低电平输入，已经判断复位信号为 1 时，与 clk 上升沿无操作，下降沿执行停机操作。

感悟：

本次实验实际上主要是锻炼模块拼接和线路调通的能力。主要分为 3 个模块，分别是指令 ROM 用于存放指令，PC 程序计数器用于提供指令的地址，以及指令寄存器用于将指令移交给未来可能出现的部件译码。整体线路过程可大致分为 3 步：1.从 PC 读取指令所在的地址 2.根据 PC 所提供的地址从 ROM 中读取地址 3.经过条件判断后将指令送入 IR 指令寄存器中。中途在板级调试中出现了按钮按下并放开，连续读入两条指令的情况发生，经检查后发现是时钟紊乱，导致上升沿和下降沿都读入了指令。其次是一开始没有输出 IR 寄存器中的指令，不便于判断送入 IR 的条件判断是否成功，后经修改并保证了数据通路和各个部件的正常运行。



PC[7:2]	N Z C V	Write_PC	Write_IR	是否符合执行条件	读出的指令代码	关联文件中指令代码
000001	1110	1	1	是	e3a0000c	e3a0000c
000010	1110	1	1	是	e3a01004	e3a01004
000011	1101	1	1	是	e3a05000	e3a05000
000100	1101	1	1	是	e3a01001	e3a01001
000101	1101	1	1	是	e3a02002	e3a02002
000110	1101	1	1	是	e3a03014	e3a03014
000111	1101	1	1	是	e0533212	e0533212
001000	1101	1	1	是	e5803004	e5803004
001001	1101	1	0	否	e5803004	2a ff ff f c
001010	1101	1	1	是	e4901004	e4901004
001011	1101	1	1	是	e7905005	e7905005
001100	1101	1	1	是	e1016090	e1016090
001101	1101	1	1	否	00000000	————