实验报告

__2014__年__05__月__28__日　　成绩:

姓名	卢魏旭	学号	12051521	班级	12052313
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	
实验序号	9	实验名称	实现 R-I 型指令的CPU设计实验		
实验时间	5 月28日	实验地点	1 教 237	实验设备号	

一、实验程序源代码

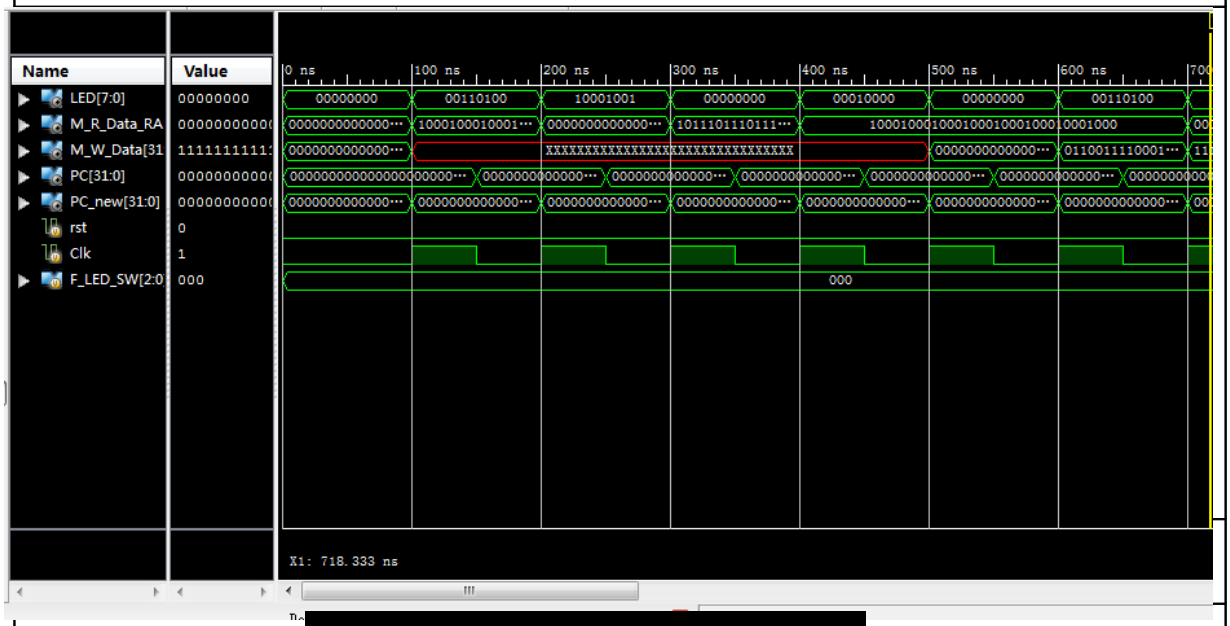
```
module work8(rst,LED,Clk,F_LED_SW,M_R_Data_RAM,M_W_Data,PC,PC_new);
        input wire Clk,rst;
        input wire[2:0] F_LED_SW;
       output wire[7:0] LED;
       wire [31:0] R_Data_A,R_Data_B,M_R_Data,Mem_addr,W_Data,imm_data;
       output wire [31:0] M_R_Data_RAM,M_W_Data;
        wire [31:0] Result,B;
       reg [5:0] ALU_OP;
       wire [4:0] W_Addr;
        output reg [31:0] PC,PC_new;
       reg  WorR,WorR_RAM,rd_rt_s,rt_imm_s,imm_s,alu_mem_s;

work4 RAM (
    .addr_A(M_R_Data[25:21]), // 数据 1 地址
    .addr_B(M_R_Data[20:16]), // 数据 2 地址
   .Data_A(R_Data_A),
   .Data_B(R_Data_B),
    .Clk(Clk),
   .Reset(rst),
    .W_Data(W_Data),
    .W_addr(W_Addr),   // 写地址
   .WorR(WorR_RAM)
   );

      Inst_ROM myROM (
   .clka(Clk), // input clka
   .addra(PC[7:2]), // input [5 : 0] addra
```

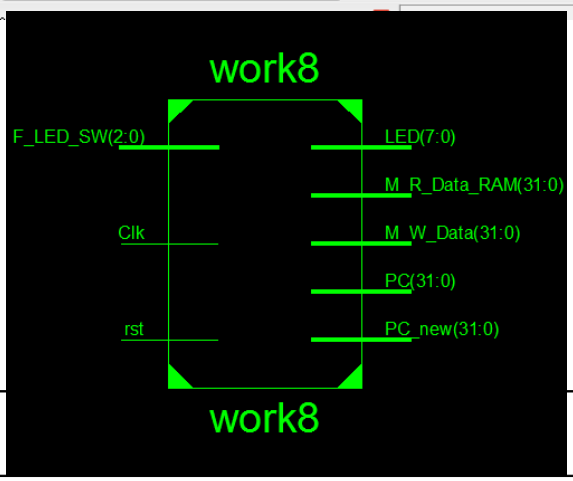
```
    .douta(M_R_Data) // output [31 : 0] douta
);
      RAM_9 RAM2 (
    .clka(Clk), // input clka
   .wea(WorR), // input [0 : 0] wea
    .addra(Mem_addr), // input [5 : 0] addra
   .dina(M_W_Data), // input [31 : 0] dina
    .douta(M_R_Data_RAM) // output [31 : 0] douta
);
work3 ALU (
    .A(R_Data_A),
   .B(B),
    .ALU_OP(ALU_OP),
   .F_LED_SW(F_LED_SW),
      .LED(LED),
    .F(Result)
   );

       initial
       begin
       PC=32'h0000_0000;
       PC_new=32'h0000_0000;
       end

        assign W_Addr=(rd_rt_s)?M_R_Data[20:16]:M_R_Data[15:11];
       assign imm_data=(imm_s)?{{16{M_R_Data[15]}},M_R_Data[15:0]}:
{{16{1'b0}},M_R_Data[15:0]};
       assign B=(rt_imm_s)?imm_data:R_Data_B;
       assign Mem_addr=Result;
       assign W_Data=(alu_mem_s)?M_R_Data_RAM:Result;
       assign M_W_Data=R_Data_B;

       always@(posedge Clk or posedge rst)  // 取指令操作
       begin
       if(rst)
            PC_new[31:0]=32'h0000_0000;
       else
             PC_new=PC+4;
       end
       always@(negedge Clk or posedge rst)  // 取指令操作
       begin
       if(rst)
            PC[31:0]=32'h0000_0000;
```

```
        else
              PC=PC_new;
        end


       always@(*)
        begin
       if(Clk)
             if(M_R_Data[31:26]==6'b000000)
              begin
                          rd_rt_s=0;
                          rt_imm_s=0;
                         alu_mem_s=0;
                          imm_s=0;
                         WorR=0;
                         WorR_RAM=1;
                         case(M_R_Data[5:0])
                               6'b100000:ALU_OP=3'b000;
                               6'b100010:ALU_OP=3'b001;
                               6'b100100:ALU_OP=3'b010;
                               6'b100101:ALU_OP=3'b011;
                               6'b100110:ALU_OP=3'b100;
                               6'b100111:ALU_OP=3'b101;
                               6'b101011:ALU_OP=3'b110;
                               6'b000100:ALU_OP=3'b111;
                          endcase
              end
              else
              begin
                    rd_rt_s=1;
                    rt_imm_s=1;
                   case(M_R_Data[31:26])
                         6'b001000:begin
  ALU_OP=3'b000;imm_s=1;WorR=0;WorR_RAM=1;alu_mem_s=0;end
                         6'b001100:begin
  ALU_OP=3'b010;imm_s=0;WorR=0;WorR_RAM=1;alu_mem_s=0;end
                         6'b001110:begin
  ALU_OP=3'b100;imm_s=0;WorR=0;WorR_RAM=1;alu_mem_s=0;end
                         6'b001011:begin
 ALU_OP=3'b110;imm_s=0;WorR=0;WorR_RAM=1;alu_mem_s=0;end
                         6'b100011:begin
  ALU_OP=3'b000;imm_s=1;WorR=0;WorR_RAM=1;alu_mem_s=1;end
                         6'b101011:begin
```

```
ALU_OP=3'b000;imm_s=1;WorR=1;WorR_RAM=0;alu_mem_s=1;end
                endcase
            end
        end
    endmodule
```

二、仿真波形





由于本实验没有在板卡上实现，故没有编写其约束文件。

五、思考与探索

本实验是在实验 8 基础上添加了 4 条立即数寻址和两条跳转指令的 I 型指令，有了实验 8 的学习基础后，做实验 9 就显得要轻松一些了，实验 9 主要是对 OP 和Func字段进行更多的译码选择，这里出现了一些问题就是没有使用过 assign 赋值语句，之前的实验中没有用过这样的语句，不知道它和always 语句块怎么配合使用的，所以之前写的代码尤其是选择控制字段都是用always 语句来检测控制，但是这样有了一个时间先后顺序的问题，容易造成类似数据还没有读入就已经在做运算了，或者 2 选 1 标志还没有赋值就已经开始写数据了，后来向同学询问了一下 assign 语句的使用方法，发现其很简单，因为它是持续运行的语句，所以不用检测时钟跳

变，随时都有数据流出的，这样就可以避免了时间先后的问题了。

总的来说实验 9 相对于实验 8 要轻松一些，只需要添加一些 I 型指令的语句，按照实验书上的方法写入就可以实现了。