计算机组成原理 实验报告

姓名：魏钊　学号：PB18111699　实验日期：2020-5-16

一、实验题目：

实验四 多周期 CPU

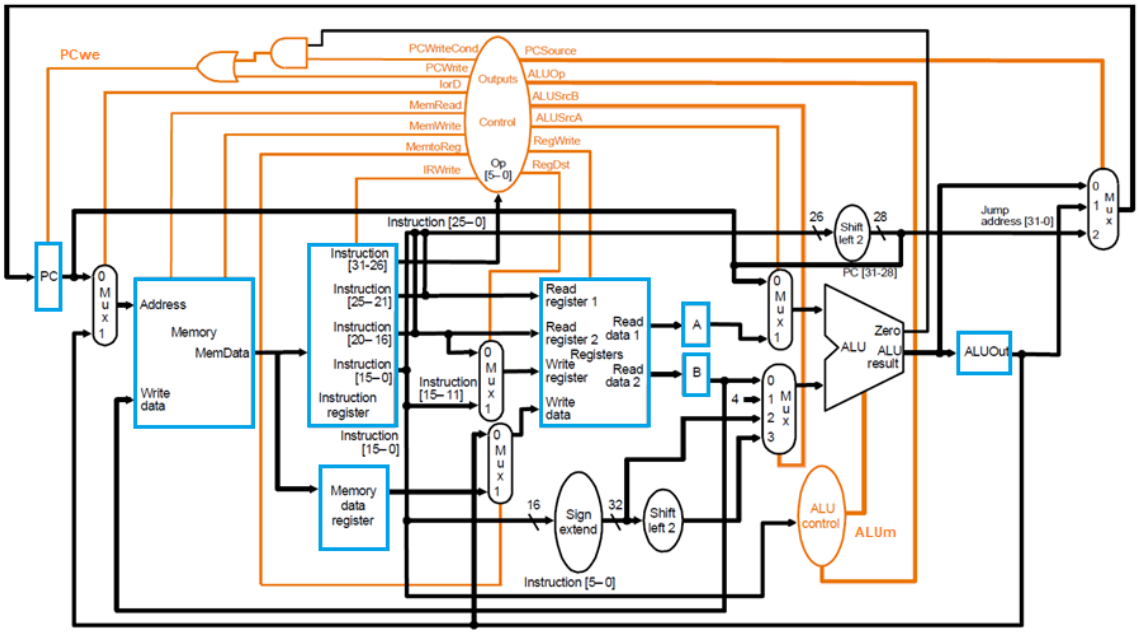
二、实验目的：

1. 理解计算机硬件的基本组成、结构和工作原理；
2. 掌握数字系统的设计和调试方法；
3. 熟练掌握数据通路和控制器的设计和描述方法。

三、实验平台：

ISE / Vivado（暂不支持其他 Verilog HDL 开发环境的检查）

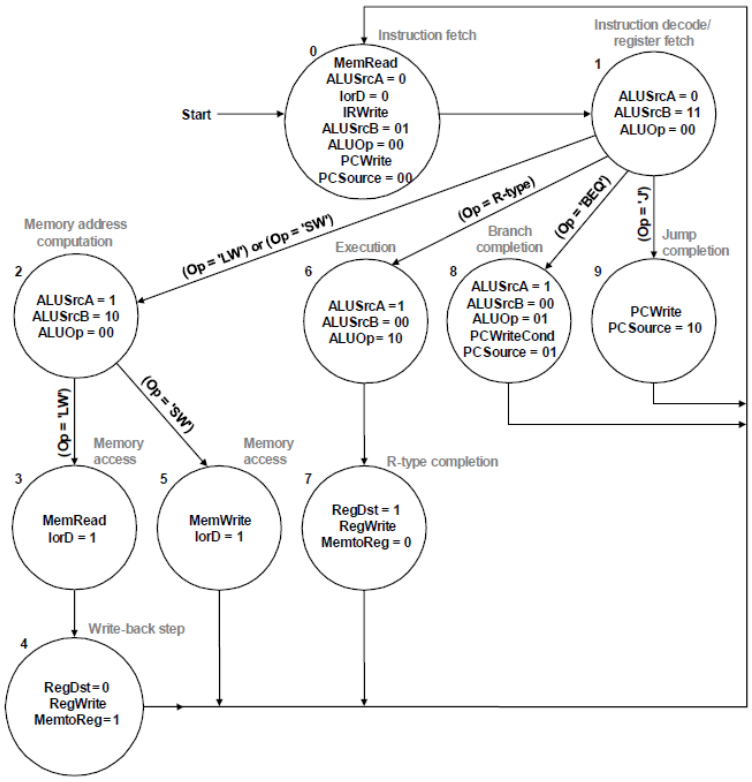
四、实验过程：



根据上图设计多周期 CPU 相关部件。

控制器：

Step	R-Type	lw/sw	beq/bne	j
IF	IR = Mem[PC] PC = PC + 4			
ID	A = Reg[IR[25-21]] B = Reg[IR[20-16]] ALUOut = PC + (SE(IR[15-0]) << 2)			
EX	ALUOut = A op B	ALUOut = A +SE(IR[15-0])	If (A==B) then PC = ALUOut	PC = PC[31-28] || (IR[25-0]<<2
MEM	Reg[IR[15-11]] = ALUOut	MDR=Mem[ALUOut] Mem[ALUOut] = B		
WB		Reg[IR[20-16]] = MDR		



根据状态图及相关信号使能设计控制器：

```
module Control(
    input   clk,
    input   rst,
    input   [5:0]   opcode,
    output  reg PCWriteCond,
    output  reg PCWrite,
    output  reg MemWrite,
    output  reg IRWrite,
    output  reg RegWrite,
    output  reg IorD,
    output  reg MemRead,
    output  reg MemtoReg,
    output  reg [1:0]   PCSource,
    output  reg ALUOp,
    output  reg [1:0]   ALUSrcB,
    output  reg ALUSrcA,
    output  reg RegDst
    );
reg [3:0]   state,next_state;

always@(posedge clk,posedge rst)
begin
    if(rst)
        state<=4'b0;
    else
        state<=next_state;
end

always@(*)
```

```
begin
    case(state)
        4'b0000:
            begin
                MemRead=1'b1;
                lorD=1'b0;
                ALUSrcA=1'b0;
                ALUSrcB=2'b1;
                ALUOp=0;
                PCSource=2'b0;
                PCWrite=1'b1;
                PCWriteCond=1'b0;
                MemWrite=1'b0;
                IRWrite=1'b1;
                RegWrite=1'b0;
                next_state=4'b0001;
            end
        4'b0001:
            begin
                IRWrite=1'b0;
                PCWrite=1'b0;
                ALUSrcA=1'b0;
                ALUSrcB=2'b11;
                ALUOp=0;
                if((opcode==6'b100011)||(opcode==6'b101011))//lw或sw
                    next_state=4'b0010;
                else    if((opcode==6'b000000)||(opcode==6'b001000))//R指令
                            next_state=4'b0110;
                else    if(opcode==6'b000100)//beq
                            next_state=4'b1000;
                else    if(opcode==6'b000010)//J
                            next_state=4'b1001;
            end
        4'b0010:
            begin
                ALUSrcA=1'b1;
                ALUSrcB=2'b10;
                ALUOp=0;
                if(opcode==6'b100011)
                    next_state=4'b0011;
                else
                    next_state=4'b0101;
            end
        4'b0011:
            begin
                MemRead=1'b1;
                lorD=1'b1;
                next_state=4'b0100;
            end
        4'b0100:
            begin
                RegDst=1'b0;
                RegWrite=1'b1;
                MemtoReg=1'b1;
                next_state=4'b0000;
            end
        4'b0101:
            begin
```

```
                MemWrite=1'b1;
                lorD=1'b1;
                next_state=4'b0000;
            end
        4'b0110:
            begin
                ALUSrcA=1'b1;
                if(opcode==6'b000000)
                    ALUSrcB=2'b0;
                else
                    ALUSrcB=2'b10;
                ALUOp=0;
                next_state=4'b0111;
            end
        4'b0111:
            begin
                if(opcode==6'b000000)
                    RegDst=1'b1;
                else
                    RegDst=1'b0;
                RegWrite=1'b1;
                MemtoReg=1'b0;
                next_state=4'b0000;
            end
        4'b1000:
            begin
                ALUSrcA=1'b1;
                ALUSrcB=2'b0;
                ALUOp=1;
                 PCWriteCond=1'b1;
                 PCSource=2'b01;
                 next_state=4'b0000;
             end
         4'b1001:
             begin
                 PCWrite=1'b1;
                 PCSource=2'b10;
                 next_state=4'b0000;
             end
         default:next_state=4'b0000;
    endcase
end
endmodule
```

ALUControl：

```
module ALUControl(
    input   ALUOp,
    output[2:0] ALUm//ALU控制信号
    );
assign ALUm=(ALUOp==1'b1) ? 3'b001:3'b000;//ALUop为1时beq判断两数是否为0，为减法，否则为加法
endmodule
```

ALU：

```
module ALU
    #(parameter WIDTH = 32)
    (
    output reg [WIDTH-1:0] y,        //运算结果
    output reg zf,                   //零标志
    output reg cf,                   //进位/借位标志
    output reg of,                   //溢出标志
    output reg sf,                   //最高位是否为1
    input [WIDTH-1:0] a, b,      //两操作数
    input [2:0] m                          //操作类型
    );
    always@(*)
        case (m)
            3'b000://加
            begin
                {cf,y}=a+b;
                of=(~a[WIDTH-1]&~b[WIDTH-1]&y[WIDTH-1])|(a[WIDTH-1]&b[WIDTH-1]&~y[WIDTH-1]);//溢出标志
                sf=y[WIDTH-1];//符号标志
                zf=~|y;//零标志
            end

            3'b001://减
            begin
                {cf,y}=a-b;
                of=(~a[WIDTH-1]&b[WIDTH-1]&y[WIDTH-1])|(a[WIDTH-1]&~b[WIDTH-1]&~y[WIDTH-1]);//溢出标志
                sf=y[WIDTH-1];//符号标志
                zf=~|y;//零标志
            end

             3'b010://与
             begin
                 cf=1'bx;
                 of=1'bx;
                 y=a&b;
                 sf=y[WIDTH-1];//符号标志
                 zf=~|y;//零标志
             end

             3'b011://或
             begin
                 cf=1'bx;
                 of=1'bx;
                 y=a|b;
                 sf=y[WIDTH-1];//符号标志
                 zf=~|y;//零标志
             end

             3'b100://异或
             begin
                 cf=1'bx;
                 of=1'bx;
                 y=a^b;
                 sf=y[WIDTH-1];//符号标志
                 zf=~|y;//零标志
             end

             default:
             begin
```

```
                cf=1'bx;
                of=1'bx;
                y=32'bx;
                sf=1'bx;
                zf=1'bx;
            end
        endcase
endmodule
```

各种 MUX：

```
module MUX_2
#(parameter WIDTH = 32)
(
    input   m,
    input   [WIDTH-1:0] a,b,
    output  [WIDTH-1:0] y
    );
assign y=(m==1)?b:a;
endmodule

module MUX_3
#(parameter WIDTH = 32)
(
    input   [1:0]    m,
    input   [WIDTH-1:0] a,b,c,
    output  [WIDTH-1:0] y
    );
assign y=(m==2'b0)?a:(m==2'b1)?b:c;
endmodule

module MUX_4
#(parameter WIDTH = 32)
(
    input   [1:0]    m,
    input   [WIDTH-1:0] a,b,c,d,
    output  [WIDTH-1:0] y
    );
assign y=(m==2'b0)?a:(m==2'b1)?b:(m==2'b10)?c:d;
endmodule
```

Mem:

指令和数据存储共用一个 RAM 存储器，采用 IP 例化实现，容量为 512 x 32 位的分布式存储器。

寄存器堆：

```
module RF
#(parameter WIDTH = 32)
(
    input clk,                  //时钟（上升沿有效）
    input [4:0] ra0,                //读端口0地址
    output [WIDTH-1:0] rd0,     //读端口0数据
    input [4:0] ra1,                //读端口1地址
    output [WIDTH-1:0] rd1,     //读端口1数据
    input [4:0] wa,                 //写端口地址
    input we,                   //写使能，高电平有效
    input [WIDTH-1:0] wd        //写端口数据
    );
reg [4:0] addr_reg;
reg [WIDTH-1:0] mem[0:31];

assign rd0=mem[ra0];//异步读
assign rd1=mem[ra1];

always@(*)
    mem[0]=0;

always@(posedge clk)
begin
    if(we)
        if(~(wa==5'b0))
            mem[wa]<=wd;//同步写
    mem[0]=0;
end
endmodule
```

CPU：

```
module CPU(
    input   rst,
    input   clk,
    input   read,
    input   [8:0]   m_rf_addr,
    output  [1:0]   PCSource,
    output  PCwe,
    output  lorD,
    output  MemWrite,
    output  IRWrite,
    output  RegDst,
    output  MemtoReg,
    output  RegWrite,
    output  [2:0]   ALUm,
    output  ALUSrcA,
    output  [1:0]   ALUSrcB,
    output  zero,
    output  reg     [31:0]  Instruction,
    output  reg     [31:0]  Memdata_r,
    output  reg     [31:0]  A,
    output  reg     [31:0]  B,
    output  reg     [31:0]  ALUOut,
    output  [31:0]  rf_data,//寄存器数据
    output  [31:0]  m_data//mem数据
    );
reg     [31:0]  PC;
wire    [31:0]  PC_next;
//reg     [31:0]  Instruction,Memdata_r;//指令和数据寄存器
wire    [31:0]  MemData,ALU_result,Read_data1,Read_data2,Write_data,ALU_OP1,ALU_OP2;
```

```
wire    [4:0]   Write_register;
//reg     [31:0]  A,B;
//reg     [31:0]  ALUOut;
wire    [8:0]   Address,addr;
wire    [4:0]   rf_addr;
//控制信号
//wire    PCwe;
wire    PCWriteCond;
wire    PCWrite;
//wire    lorD;
wire    MemRead;
//wire    MemWrite;
//wire    MemtoReg;
//wire    IRWrite;
//wire    [1:0]   PCSource;
wire    ALUOp;
//wire    [2:0]   ALUm;
//wire    [1:0]   ALUSrcB;
//wire    ALUSrcA;
//wire    RegWrite;
//wire    RegDst;
//wire    zero;
wire    zf,cf,of;

assign addr=(read==1)?m_rf_addr:Address;
assign  m_data=MemData;
assign  rf_data=Read_data1;
assign  rf_addr=(read==1)?m_rf_addr[4:0]:Instruction[25:21];

Mem mem(.a(addr),.d(B),.clk(clk),.we(MemWrite),.spo(MemData));

RF  Registers(.clk(clk),.ra0(rf_addr),.rd0(Read_data1),
              .ra1(Instruction[20:16]),.rd1(Read_data2),
              .wa(Write_register),.we(RegWrite),.wd(Write_data));

ALU alu(.y(ALU_result),.zf(zf),.cf(cf),.of(of),.a(ALU_OP1),.b(ALU_OP2),.m(ALUm));

Control control(.clk(clk),.rst(rst),.opcode(Instruction[31:26]),
              .PCWriteCond(PCWriteCond),.PCWrite(PCWrite),.MemWrite(MemWrite),
              .IRWrite(IRWrite),.RegWrite(RegWrite),.lorD(lorD),.MemRead(MemRead),
              .MemtoReg(MemtoReg),.PCSource(PCSource),.ALUOp(ALUOp),.ALUSrcB(ALUSrcB),
              .ALUSrcA(ALUSrcA),.RegDst(RegDst));

ALUControl  alu_c(.ALUOp(ALUOp),.ALUm(ALUm));

MUX_2 mux0(.m(lorD),.a(PC[10:2]),.b(ALUOut[10:2]),.y(Address));
MUX_2 mux1(.m(RegDst),.a(Instruction[20:16]),.b(Instruction[15:11]),.y(Write_register));
MUX_2 mux2(.m(MemtoReg),.a(ALUOut),.b(Memdata_r),.y(Write_data));
MUX_2 mux3(.m(ALUSrcA),.a(PC),.b(A),.y(ALU_OP1));
MUX_4 mux4(.m(ALUSrcB),.a(B),.b(32'b100),.c({16'b0,Instruction[15:0]}),.d({14'b0,Instruction[15:0],2'b0}),.y(ALU_OP2));
MUX_3 mux5(.m(PCSource),.a(ALU_result),.b(ALUOut),.c({PC[31:28],Instruction[25:0],2'b00}),.y(PC_next));

assign  PCwe=((PCWrite==1)||((PCWriteCond==1)&&(zero==1)));
assign  zero=(zf==1&&cf==0&&of==0);

always@(posedge clk,posedge rst)
begin
    Memdata_r<=MemData;
```

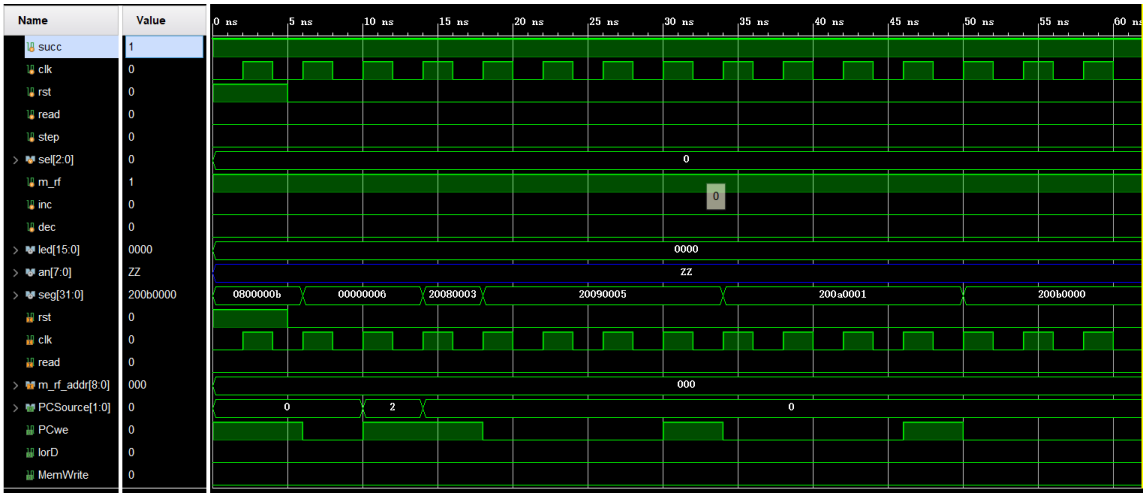
```
    ALUOut<=ALU_result;
    A<=Read_data1;
    B<=Read_data2;
    if(rst)
        PC<=32'b0;
    else
        if(PCwe)
           PC<=PC_next;
    if(IRWrite)
        Instruction<=MemData;
end
endmodule
```

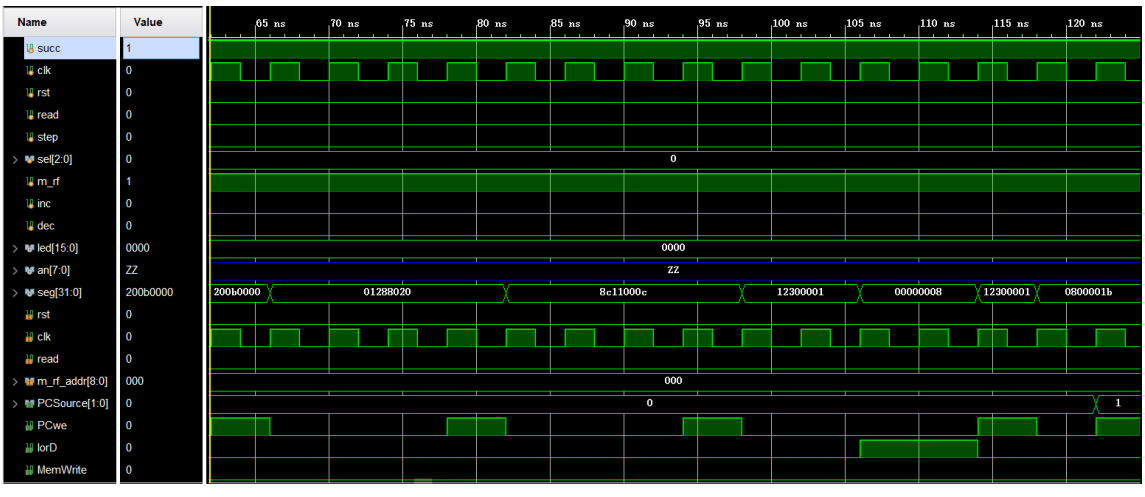
DBU:

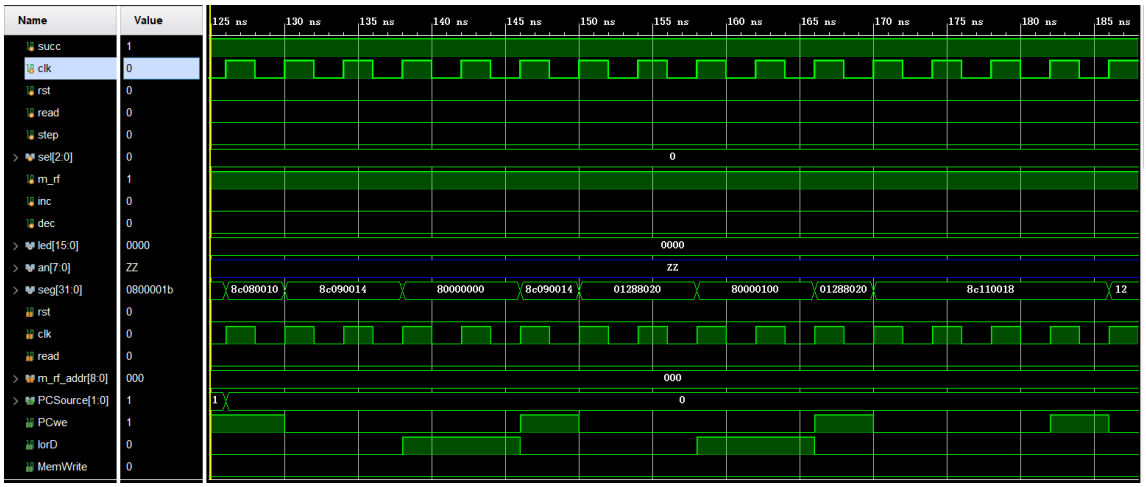
```
module DBU(
    input   succ,
    input   clk,
    input   rst,
    input   read,//读使能
    input   step,
    input   [2:0] sel,
    input   m_rf,
    input   inc,
    input   dec,
    output  [15:0] led,//16位led
    output  [7:0]  an,//数码管片选
    output  reg [31:0]  seg//8个八段数码管
    );
wire    step_clean,inc_clean,dec_clean;
wire    step_fin,inc_fin,dec_fin;
wire    clock;
wire    [31:0] PC;
reg     [8:0]   m_rf_addr;
wire    [31:0]  rf_data;
wire    [31:0]  m_data;
wire     [31:0]  Instruction,Memdata_r,A,B,ALUOut;

wire    [1:0]   PCSource;
wire    PCwe;
wire    lorD;
wire    MemWrite;
wire    IRWrite;
wire    RegDst;

wire    MemtoReg;
wire    RegWrite;
wire    [2:0]   ALUm;
wire    ALUSrcA;
wire    [1:0]   ALUSrcB;
wire    zero;
//相关按钮去抖动
Jitter_clr jitter_clr1(.clk(clk),.button(step),.button_clean(step_clean));
Jitter_clr jitter_clr2(.clk(clk),.button(inc),.button_clean(inc_clean));
Jitter_clr jitter_clr3(.clk(clk),.button(dec),.button_clean(dec_clean));
//相关按钮取边沿
EDG edg1(.clk(clk),.rst(rst),.y(step_clean),.p(step_fin));
EDG edg2(.clk(clk),.rst(rst),.y(inc),.p(inc_fin));
EDG edg3(.clk(clk),.rst(rst),.y(dec_clean),.p(dec_fin));

CPU cpu(rst,clock,read,m_rf_addr,PCSource,PCwe,lorD,MemWrite,IRWrite,RegDst,
        MemtoReg,RegWrite,ALUm,ALUSrcA,ALUSrcB,zero,Instruction,Memdata_r,A,B,
        ALUOut,rf_data,m_data);
assign  clock=(read==1)?1'b0:((succ==1)?clk:step_fin);
assign  led=(sel==0)?{7'b0,m_rf_addr}:{PCSource,PCwe,lorD,MemWrite,IRWrite,RegDst,MemtoReg,RegWrite,ALUm,ALUSrcA,ALUSrcB,zero};

always@(posedge clk or posedge rst)
begin
    if(rst)
        m_rf_addr<=9'b0;
    else if((inc_fin==1'b1)&(dec_fin==1'b0))
    m_rf_addr<=m_rf_addr+9'b1;
    else if((dec_fin==1'b1)&(inc_fin==1'b0))
    m_rf_addr<=m_rf_addr-9'b1;
```

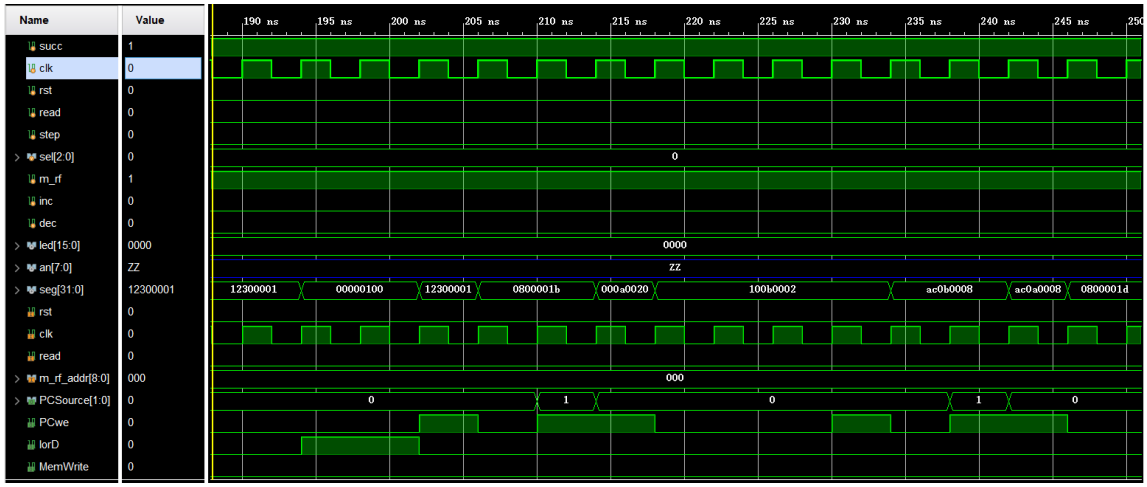
```
    else
    m_rf_addr<=m_rf_addr;
end

always@(*)
begin
    if(sel==3'b0)
    begin
        if(m_rf==1)
            seg=m_data;
        else
            seg=rf_data;
    end
    else
    begin
    if(sel==3'b001)
    seg=PC;
    else if(sel==3'b010)
        seg=Instruction;
    else if(sel==3'b011)
        seg=Memdata_r;
    else if(sel==3'b100)
        seg=A;
    else if(sel==3'b101)
        seg=B;
    else if(sel==3'b110)
        seg=ALUOut;
    else if(sel==3'b111)
        seg=0;

    end
end
endmodule
```

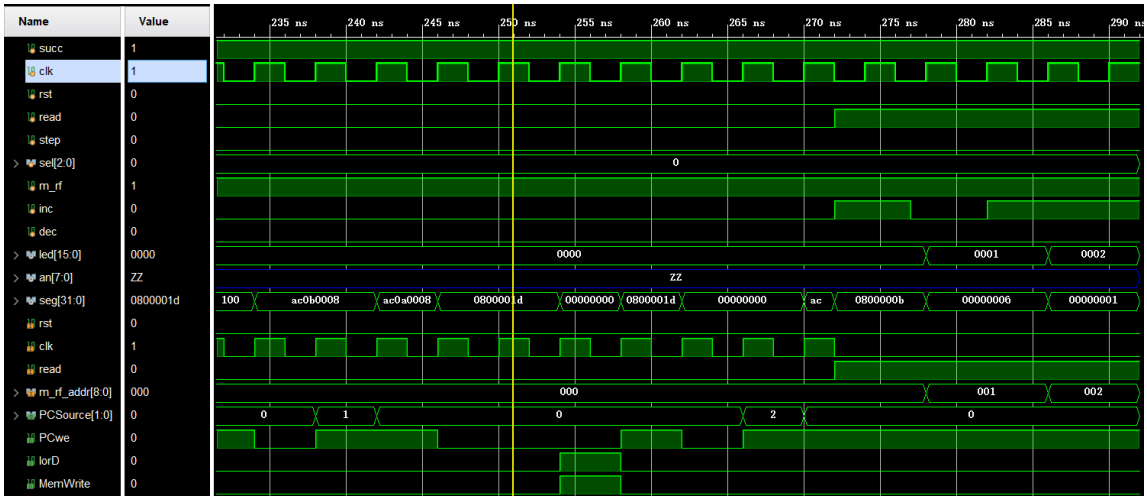
五、实验结果：

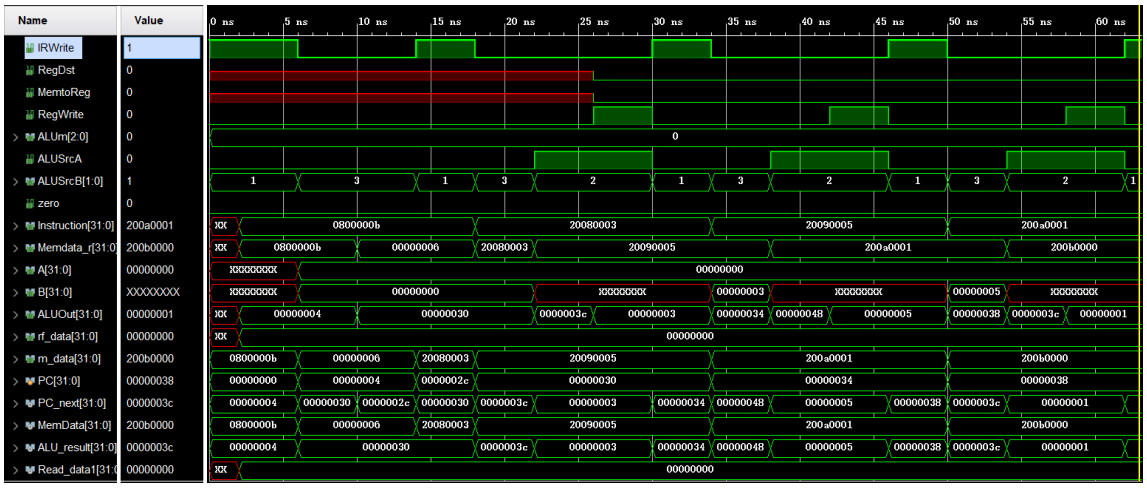


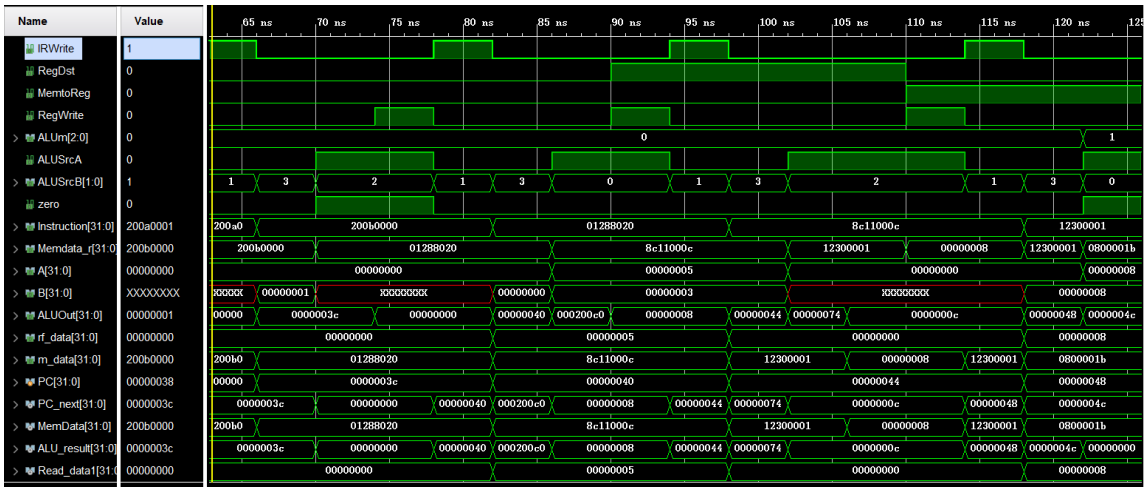


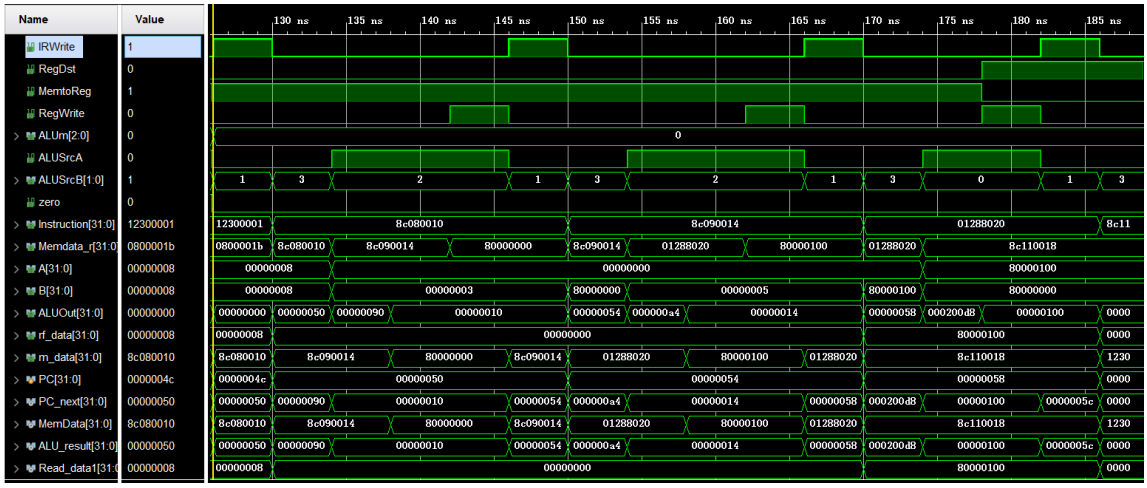

Name
Value
succ 1
clk 0
rst 0
read 0
step 0
sel[2:0] 0
m_rf 1
inc 0
dec 0
led[15:0] 0000
an[7:0] ZZ
seg[31:0] 0800001b
rst 0
clk 0
read 0
m_rf_addr[8:0] 000
PCSource[1:0] 1
PCwe 1
IorD 0
MemWrite 0
125 ns 130 ns 135 ns 140 ns 145 ns 150 ns 155 ns 160 ns 165 ns 170 ns 175 ns 180 ns 185 ns
8c080010 8c090014 80000000 8c090014 01288020 80000100 01288020 8c110018 12

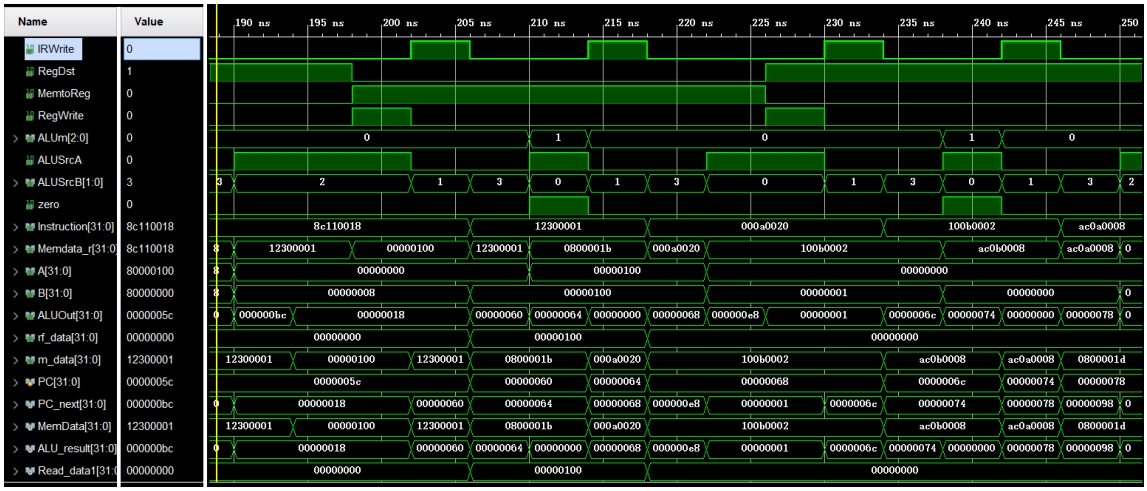

Name
Value
succ 1
clk 0
rst 0
read 0
step 0
sel[2:0] 0
m_rf 1
inc 0
dec 0
led[15:0] 0000
an[7:0] ZZ
seg[31:0] 12300001
rst 0
clk 0
read 0
m_rf_addr[8:0] 000
PCSource[1:0] 0
PCwe 0
IorD 0
MemWrite 0
190 ns 195 ns 200 ns 205 ns 210 ns 215 ns 220 ns 225 ns 230 ns 235 ns 240 ns 245 ns
12300001 00000100 12300001 0800001b 000a0020 100b0002 ac0b0008 ac0a0008 0800001d

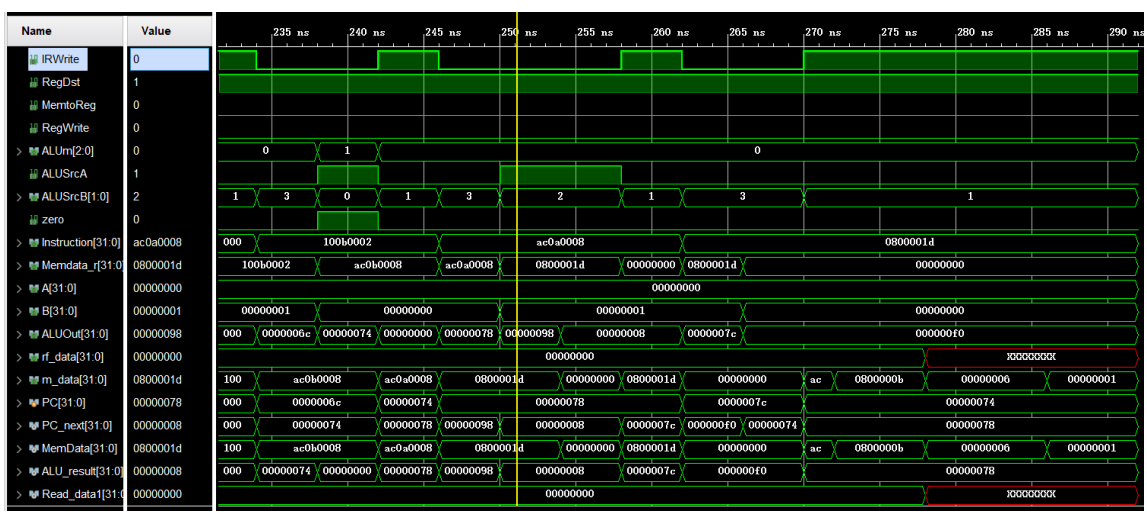

Name
Value
succ 1
clk 1
rst 0
read 0
step 0
sel[2:0] 0
m_rf 1
inc 0
dec 0
led[15:0] 0000
an[7:0] ZZ
seg[31:0] 0800001d
rst 0
clk 1
read 0
m_rf_addr[8:0] 000
PCSource[1:0] 0
PCwe 0
IorD 0
MemWrite 0
235 ns 240 ns 245 ns 250 ns 255 ns 260 ns 265 ns 270 ns 275 ns 280 ns 285 ns 290 ns
100 ac0b0008 ac0a0008 0800001d 00000000 0800001d 00000000 ac 0800000b 00000006 00000001

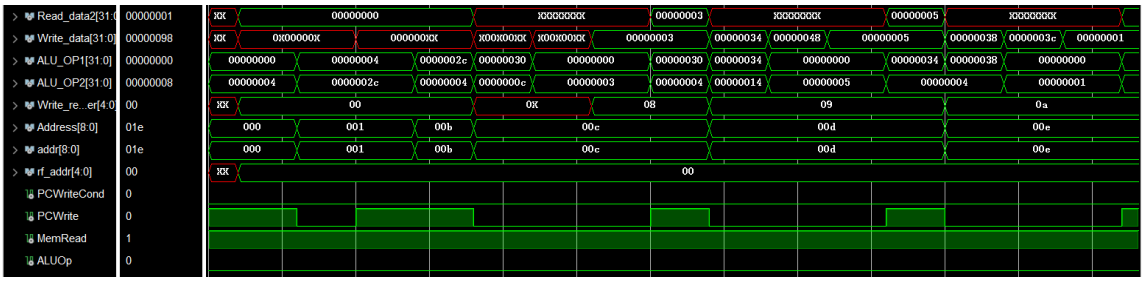

Name
Value
IRWrite 1
RegDst 0
MemtoReg 0
RegWrite 0
ALUm[2:0] 0
ALUSrcA 0
ALUSrcB[1:0] 1
zero 0
Instruction[31:0] 200a0001
Memdata_r[31:0] 200b0000
A[31:0] 00000000
B[31:0] XXXXXXXX
ALUOut[31:0] 00000001
rf_data[31:0] 00000000
m_data[31:0] 200b0000
PC[31:0] 00000038
PC_next[31:0] 0000003c
MemData[31:0] 200b0000
ALU_result[31:0] 0000003c
Read_data1[31:0] 00000000
0 ns 5 ns 10 ns 15 ns 20 ns 25 ns 30 ns 35 ns 40 ns 45 ns 50 ns 55 ns 60 ns

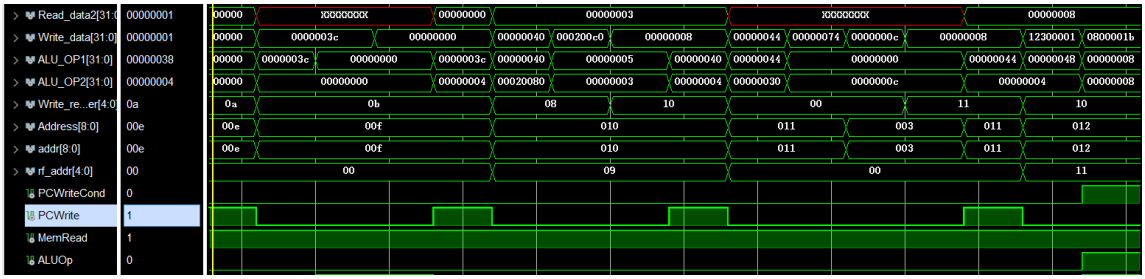

Name
Value
IRWrite 1
RegDst 0
MemtoReg 0
RegWrite 0
ALUm[2:0] 0
ALUSrcA 0
ALUSrcB[1:0] 1
zero 0
Instruction[31:0] 200a0001
Memdata_r[31:0] 200b0000
A[31:0] 00000000
B[31:0] XXXXXXXX
ALUOut[31:0] 00000001
rf_data[31:0] 00000000
m_data[31:0] 200b0000
PC[31:0] 00000038
PC_next[31:0] 0000003c
MemData[31:0] 200b0000
ALU_result[31:0] 0000003c
Read_data1[31:0] 00000000
65 ns
70 ns
75 ns
80 ns
85 ns
90 ns
95 ns
100 ns
105 ns
110 ns
115 ns
120 ns

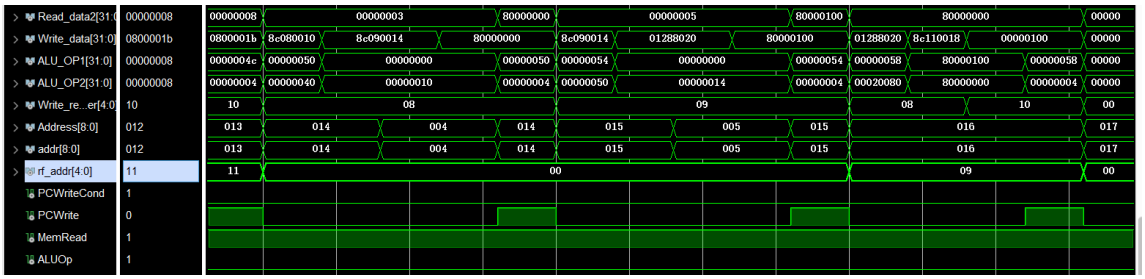

Name
Value
IRWrite 1
RegDst 0
MemtoReg 1
RegWrite 0
ALUm[2:0] 0
ALUSrcA 0
ALUSrcB[1:0] 1
zero 0
Instruction[31:0] 12300001
Memdata_r[31:0] 0800001b
A[31:0] 00000008
B[31:0] 00000008
ALUOut[31:0] 00000000
rf_data[31:0] 00000008
m_data[31:0] 8c080010
PC[31:0] 0000004c
PC_next[31:0] 00000050
MemData[31:0] 8c080010
ALU_result[31:0] 00000050
Read_data1[31:0] 00000008
130 ns
135 ns
140 ns
145 ns
150 ns
155 ns
160 ns
165 ns
170 ns
175 ns
180 ns
185 ns

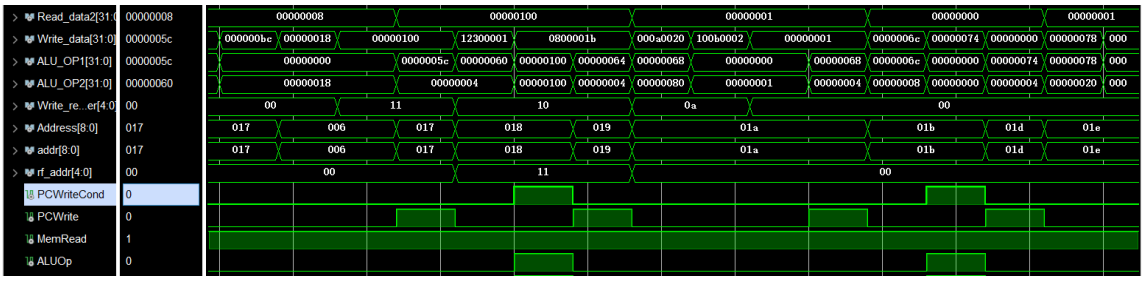

Name
Value
IRWrite 0
RegDst 1
MemtoReg 0
RegWrite 0
ALUm[2:0] 0
ALUSrcA 0
ALUSrcB[1:0] 3
zero 0
Instruction[31:0] 8c110018
Memdata_r[31:0] 8c110018
A[31:0] 80000100
B[31:0] 80000000
ALUOut[31:0] 0000005c
rf_data[31:0] 00000000
m_data[31:0] 12300001
PC[31:0] 0000005c
PC_next[31:0] 000000bc
MemData[31:0] 12300001
ALU_result[31:0] 000000bc
Read_data1[31:0] 00000000
190 ns
195 ns
200 ns
205 ns
210 ns
215 ns
220 ns
225 ns
230 ns
235 ns
240 ns
245 ns
250 ns

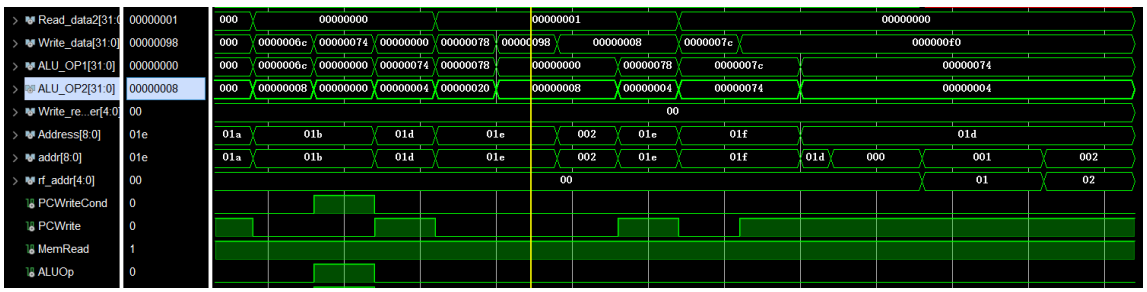

Name
Value
IRWrite 0
RegDst 1
MemtoReg 0
RegWrite 0
ALUm[2:0] 0
ALUSrcA 1
ALUSrcB[1:0] 2
zero 0
Instruction[31:0] ac0a0008
Memdata_r[31:0] 0800001d
A[31:0] 00000000
B[31:0] 00000001
ALUOut[31:0] 00000098
rf_data[31:0] 00000000
m_data[31:0] 0800001d
PC[31:0] 00000078
PC_next[31:0] 00000008
MemData[31:0] 0800001d
ALU_result[31:0] 00000008
Read_data1[31:0] 00000000
235 ns
240 ns
245 ns
250 ns
255 ns
260 ns
265 ns
270 ns
275 ns
280 ns
285 ns
290 ns











六、心得体会：

通过本次实验理解了计算机硬件的基本组成、结构和工作原理；掌握了数字系统的设计和调试方法；熟练掌握了数据通路和控制器的设计和描述方法。