**题 目： CPU实验**

**学 院：**

**专 业：**

**学 号：**

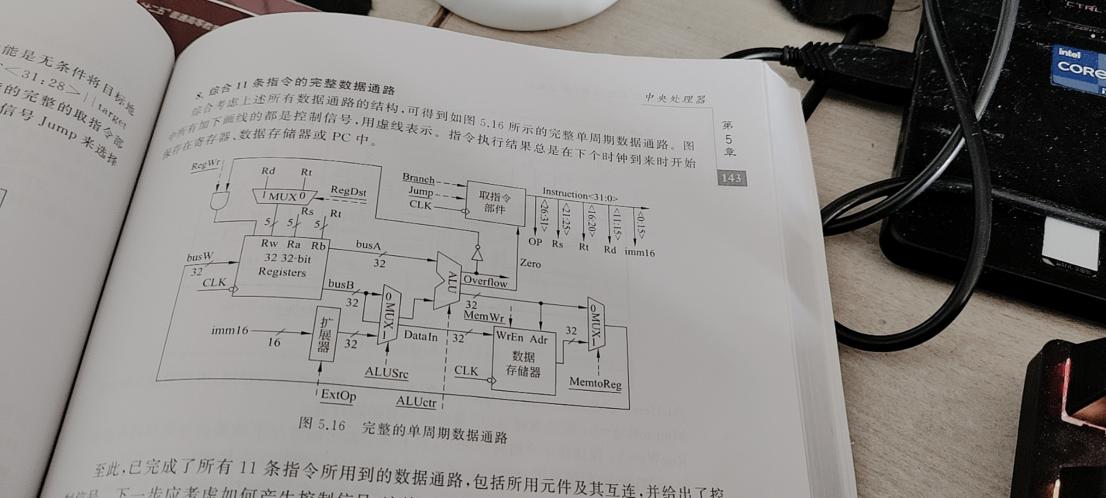
**姓 名：**

**指导教师：**

1. 实验目的和原理

实验目的：实现单周期CPU的组装

实验原理：



1. 代码展示

module CPU\_1(

 input wire Reset,

 input wire CLK,

 output wire [31:0] ALUresult,

 output wire [31:0] PCC,

 output wire [31:0] busA,

 output wire [31:0] busB,

 output wire [31:0] busW,

 output wire [31:0] busB\_dealed,

 output wire [5:0] OP,

 output wire [10:0] control,

 output wire [31:0] MemData

);

*// 控制信号*

    wire Branch, Jump, RegDst, ALUsrc, MemtoReg, RegWr, MemWr, ExtOp;

    wire [2:0] ALUctr;

*// 指令相关信号*

    wire [15:0] imm16;            *// 立即数扩展值*

    wire [31:0] imm32;

    wire [5:0] func;                *// 操作码*

    wire [4:0] Rs, Rt, Rd, Rw;       *// 源寄存器、目标寄存器等*

*// wire [5:0] OP;*

    wire Overflow, Zero;

*// 程序计数器信号*

*// wire [31:0] PC;               // 程序计数器）*

*// 存储信号*

    wire [31:0] MemDataIn;        *// 待写入的数据（数据存储）*

    assign control = {Branch, Jump, RegDst, ALUsrc, ALUctr, MemtoReg, RegWr, MemWr, ExtOp};

    InstructionFetch IF (

        .CLK(CLK),

        .Branch(Branch),

        .Jump(Jump),

        .Zero(Zero),              *// 由ALU控制*

        .OP(OP),                  *// 操作码*

        .Rs(Rs),                  *// 源寄存器1*

        .Rt(Rt),                  *// 源寄存器2*

        .Rd(Rd),                  *// 目标寄存器*

        .imm16(imm16),    *// 输出的原始立即数*

  .func(func),

  .reset(reset),

  .PCC(PCC)

    );

    Decode decode (

        .OP(OP),

        .func(func),

        .Branch(Branch),

        .Jump(Jump),

        .RegDst(RegDst),

        .ALUsrc(ALUsrc),

        .ALUctr(ALUctr),

        .MemtoReg(MemtoReg),

        .RegWr(RegWr),

        .MemWr(MemWr),

        .ExtOp(ExtOp)

    );

    assign RegWr\_dealed = RegWr && (~Overflow);

    MUX2to1 mux1(Rt, Rd, RegDst, Rw);

    Registers regs (

        .clk(CLK),

        .RegWr(RegWr\_dealed),

        .Rw(Rw),

        .Ra(Rs),

        .Rb(Rt),

        .busW(busW),       *// 写入寄存器的数据来自 ALU 或其他*

        .busA(busA),                 *// 从寄存器中读取的数据A*

        .busB(busB),                  *// 从寄存器中读取的数据B*

  .reset(reset)

    );

    Ext32 ext32(imm16, ExtOp, imm32);

    MUX2to1 mux2(busB, imm32, ALUsrc, busB\_dealed);

    ALU alu(

  .A(busA),

  .B(busB\_dealed),

  .ALUctr(ALUctr),

  .Result(ALUresult),

  .Zero(Zero),

  .Overflow(Overflow)

  );

  Store store (

        .clk(CLK),

        .WrEn(MemWr),

        .Adr(ALUresult),        *// ALU 结果作为存储的地址*

        .DataIn(busB),              *// 寄存器中的数据写入存储器*

        .DataOut(MemData)           *// 输出存储的数据*

    );

    MUX2to1 mux3(ALUresult, MemData, MemtoReg, busW);

 endmodule

module MUX2to1(

 input [31:0] A,

 input [31:0] B,

 input ctr,

 output [31:0] Result);

 assign Result = (ctr == 1'b0) ? A : B;

endmodule

module Ext32(

 input [15:0] imm16,

 input ExtOp,

 output [31:0] imm32

);

 assign imm32 = (ExtOp == 0) ? {16'b0, imm16} : {{16{imm16[15]}}, imm16};

endmodule

1. 仿真实验

这里制造了一个程序

#include <stdio.h>

int main() {

int sum = 0; // 存储和，初始值为 0

int i = 1; // 迭代变量 i，初始值为 1

int limit = 10; // 循环上限

// 循环计算 sum = 1 + 2 + ... + limit

while (i <= limit) {

sum += i; // 将当前 i 加入 sum

i++; // i 自增 1

}

// 打印结果

printf("Sum = %d\n", sum);

return 0;

}

其理想输出应当是45.

对应指令为：

addiu $t0, $zero, 0

addiu $t1, $zero, 1

addiu $t2, $zero, 10

Loop：

beq $t1, $t2, end

add $t0, $t0, $t1

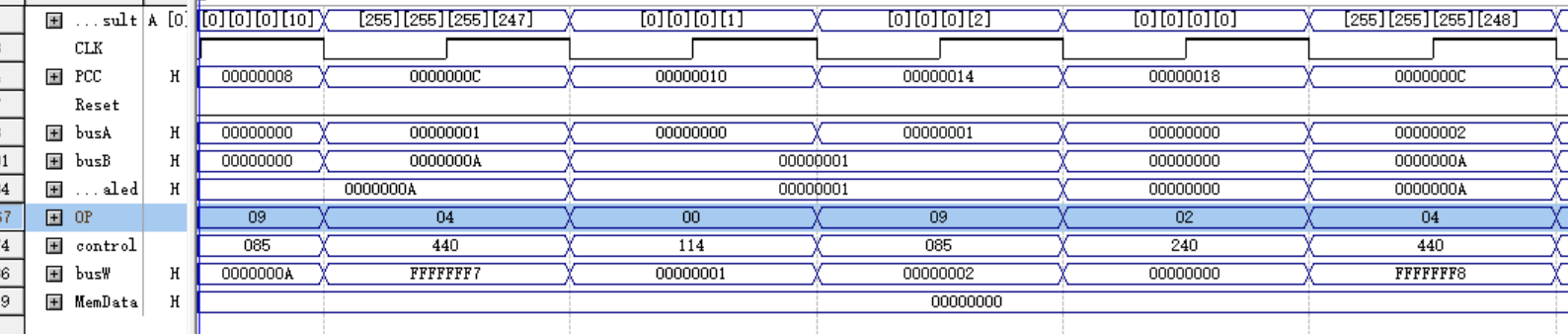
addiu $t1, $t1, 1

j loop

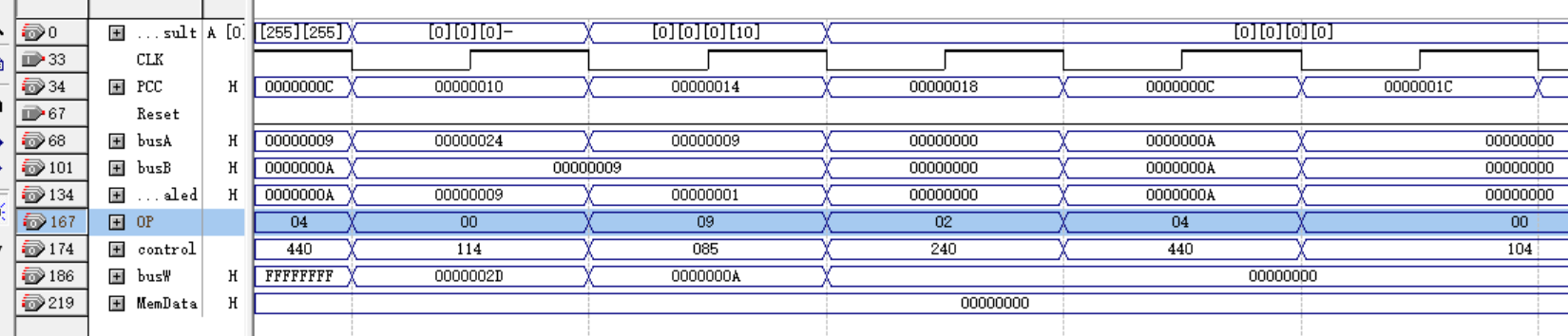
End:

Nop

对应到设计的仿真结果，这里着重关注PC的自增和跳转



000C处对应的是beq指令，可以看见当未达到条件的时候。beq指令不会进行跳转。



看busW的值，busW的值即为在加法结束之后的值。当busW为002DH，即45时，跳转条件也符合了，此时PC从0C跳转到1C，即程序的结束位置。

**题 目： ALU实验**

**学 院：**

**专 业：**

**学 号：**

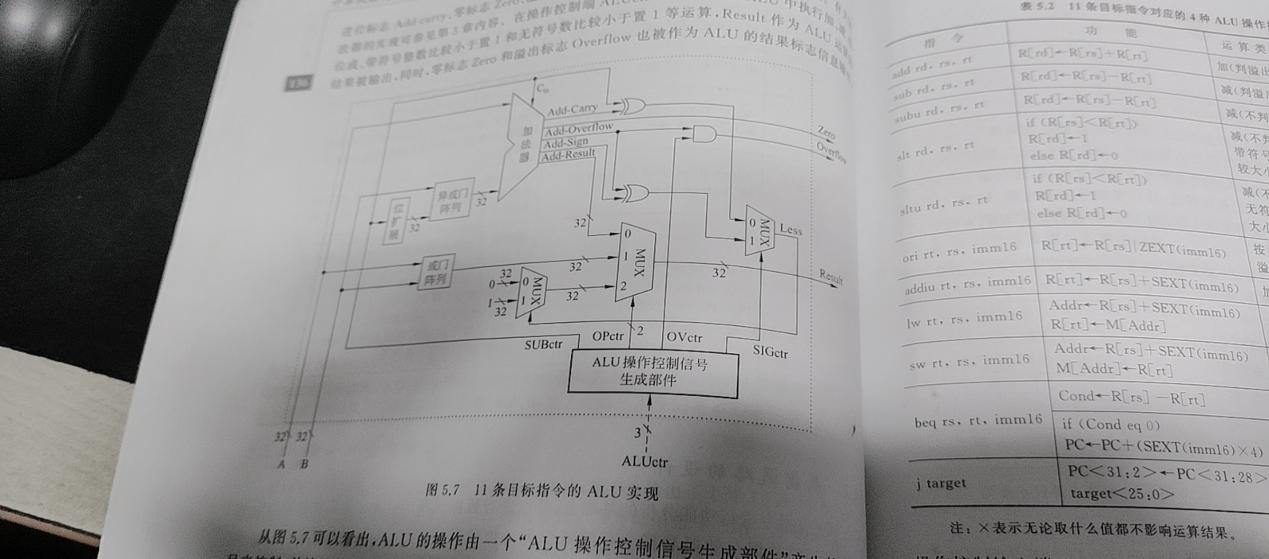
**姓 名：**

**指导教师：**

一、实验目的和原理

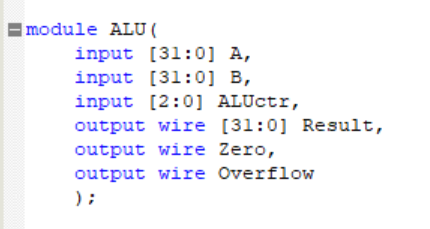
实验目的：搭建一个ALU用于单周期CPU计算

实验原理：

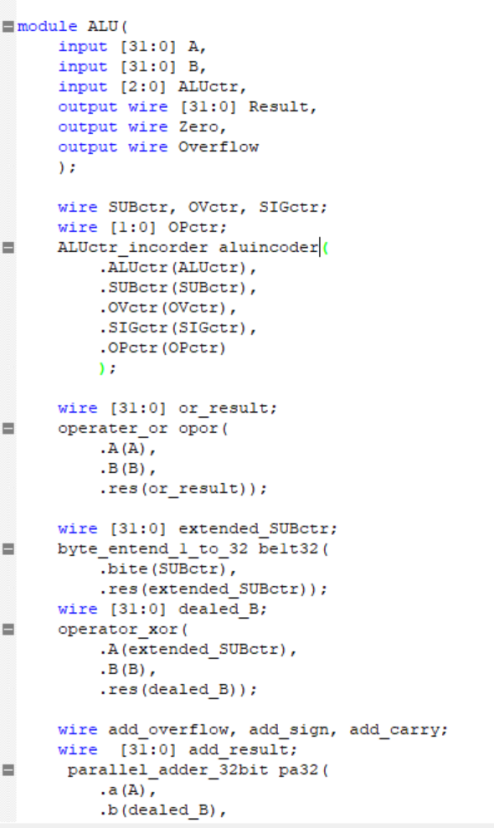


1. 实验代码

如图为ALU的输入输出：

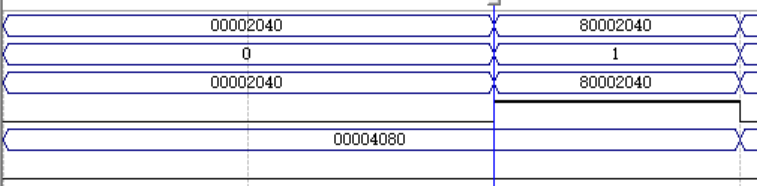


然后根据书上的结构搭建即可



1. 仿真展示

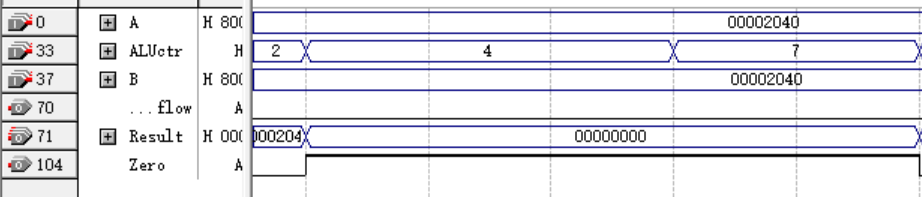
这里制造了多个数据，以实验每个接口是否可以正常工作。



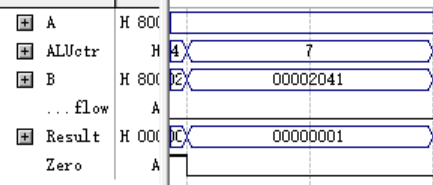
当2040H+20240H时，结果为4080H，加法器可以正常工作。

在ALUctr = 1 时制造了一个会出现溢出的数据，此处的溢出标志位正常工作。

这是2040H - 2040H= 0 时的结果，表明零标志位可以正常工作



当比较100100和100100100的大小时，即使用slt指令时，会返回less



**题 目： 存储器实验**

**学 院：**

**专 业：**

**学 号：**

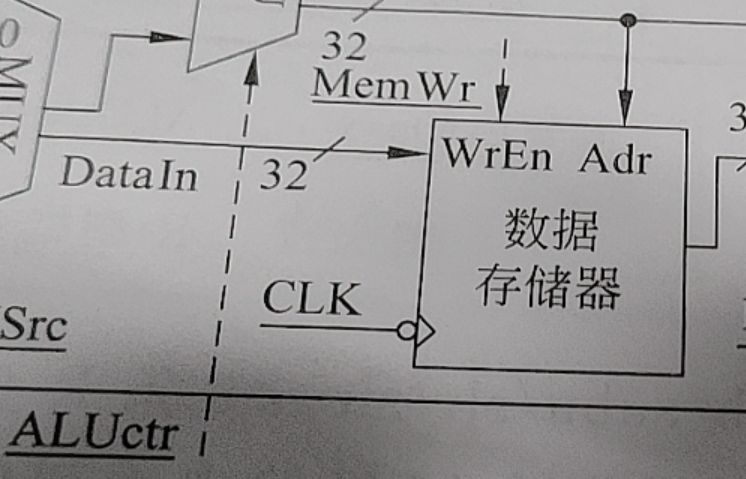
**姓 名：**

**指导教师：**

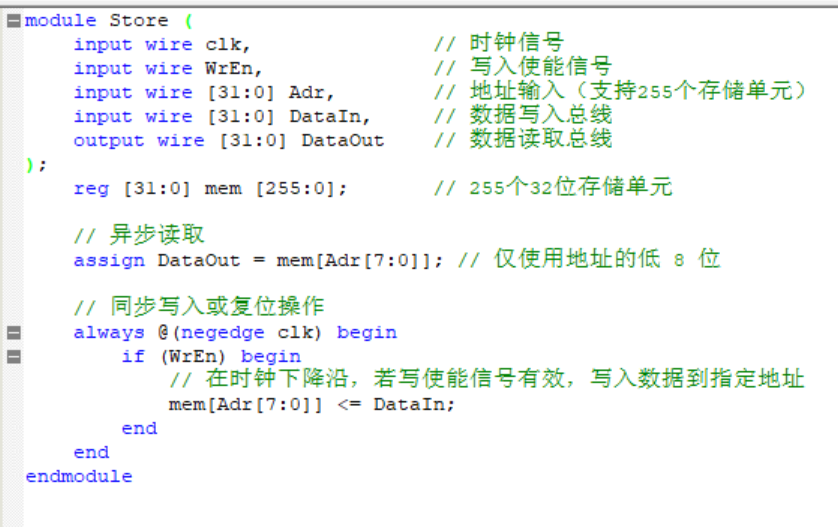
一、实验目的和原理

实验目的：建立起单周期CPU的数据存储器。

实验原理：

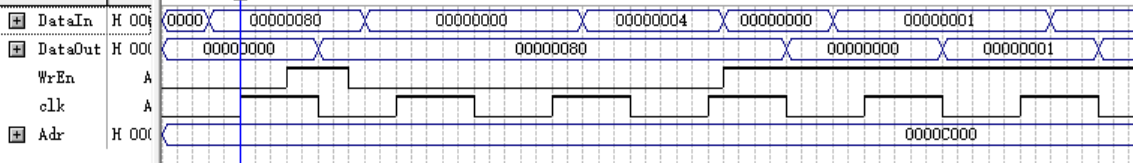


1. 代码展示



这个程序当中不需要有reset，这是因为存储程序应该是断线时内存不受影响的、

1. 仿真调试



由图可知，在时钟下降沿且写使能端是高电平的时候，会对指定的存储器进行注入。如果使能端为低电平，则不会进行注入。

**题 目： 寄存器实验**

**学 院：**

**专 业：**

**学 号：**

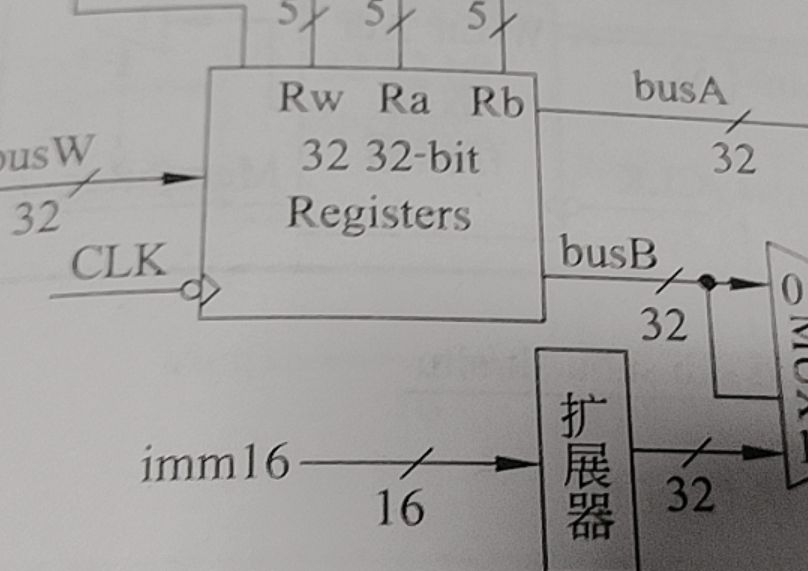
**姓 名：**

**指导教师：**

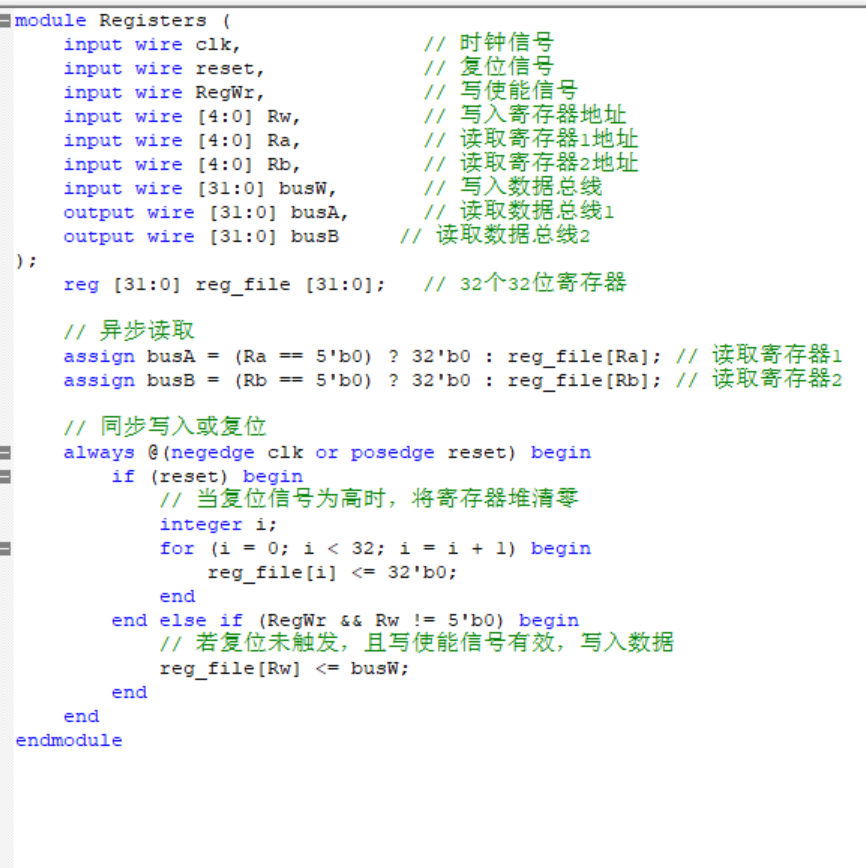
1. 实验目的和原理

实验目的：实现单周期CPU当中的寄存器。

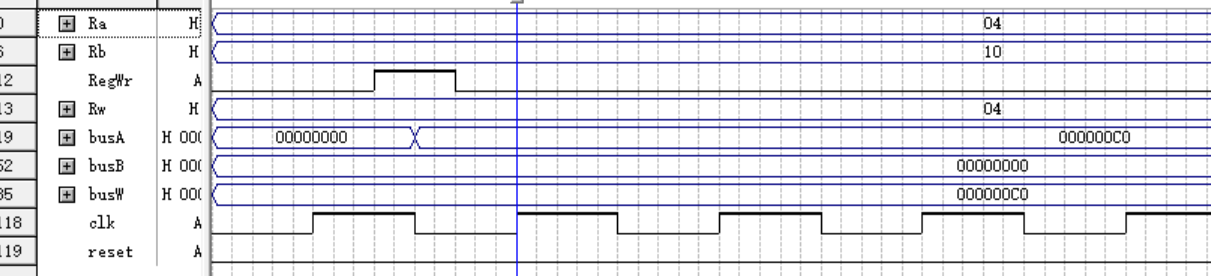
实验原理：



1. 代码展示



1. 仿真实验



这里对4号寄存器进行了同步注入之后，对busA和busB进行了异步的提取。

寄存器的运转时正常的。

**题 目： 取指令部件实验**

**学 院：**

**专 业：**

**学 号：**

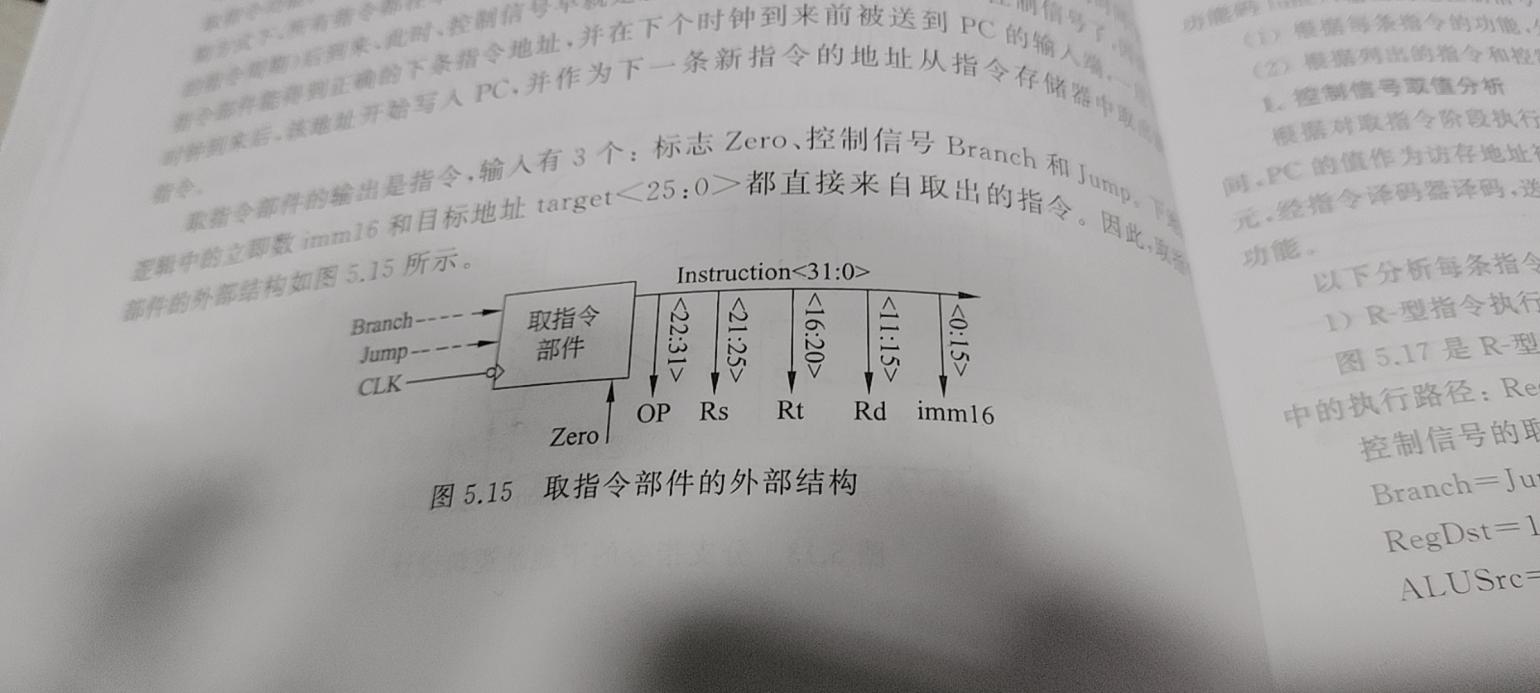
**姓 名：**

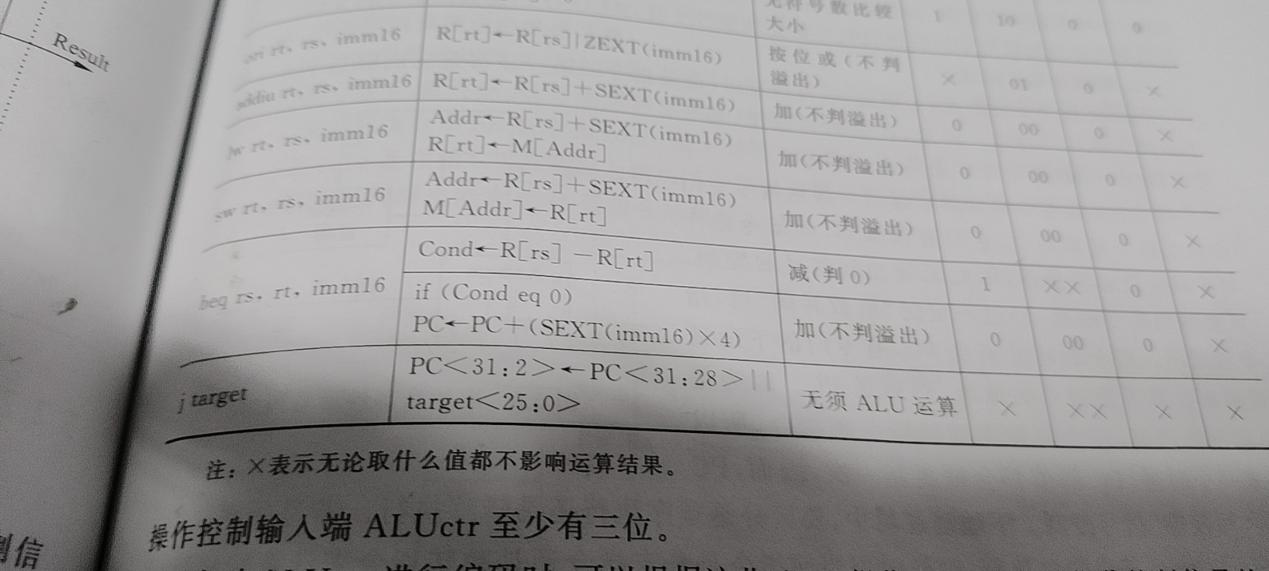
**指导教师：**

1. 实验目的和原理

实验目的：实现单周期CPU当中的取指令部件。

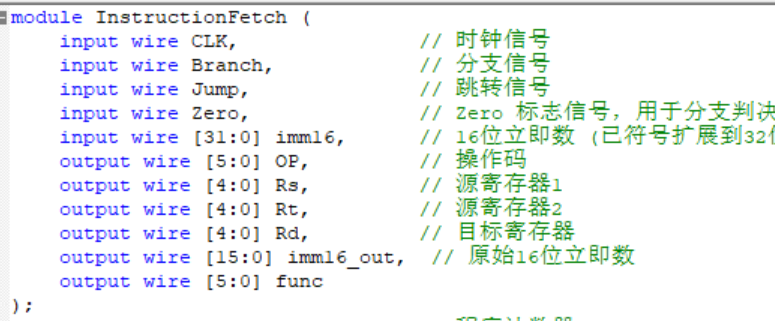
实验原理：



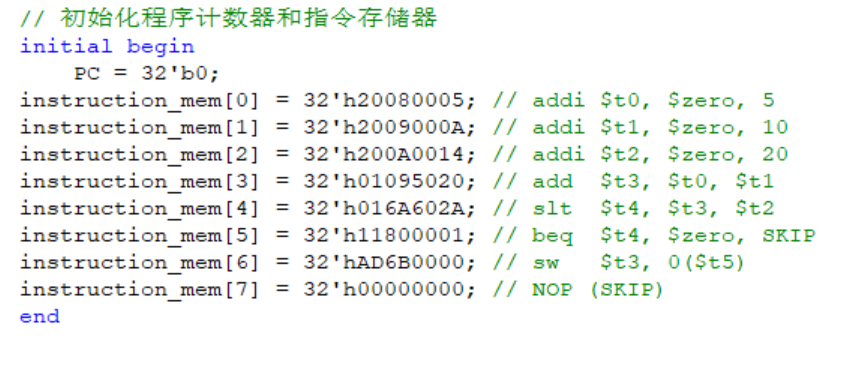


转移表达式也与这两个指令相关。

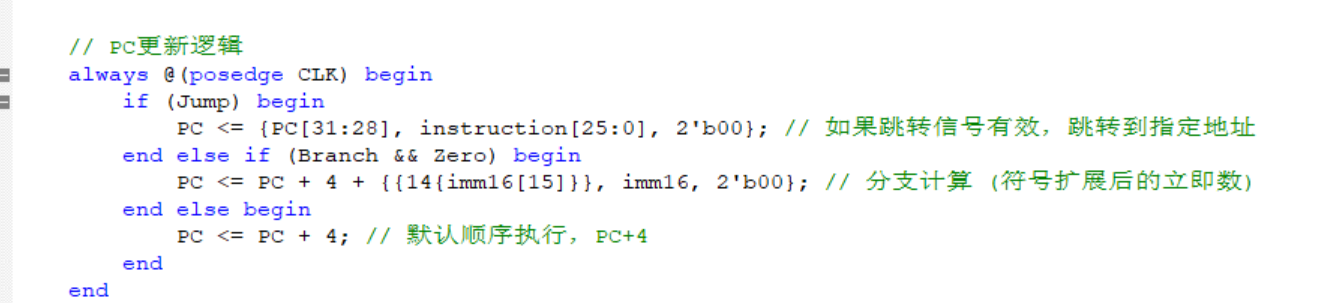
1. 代码展示
2. 取指令部件的输入和输出



1. 设置PC和导入指令

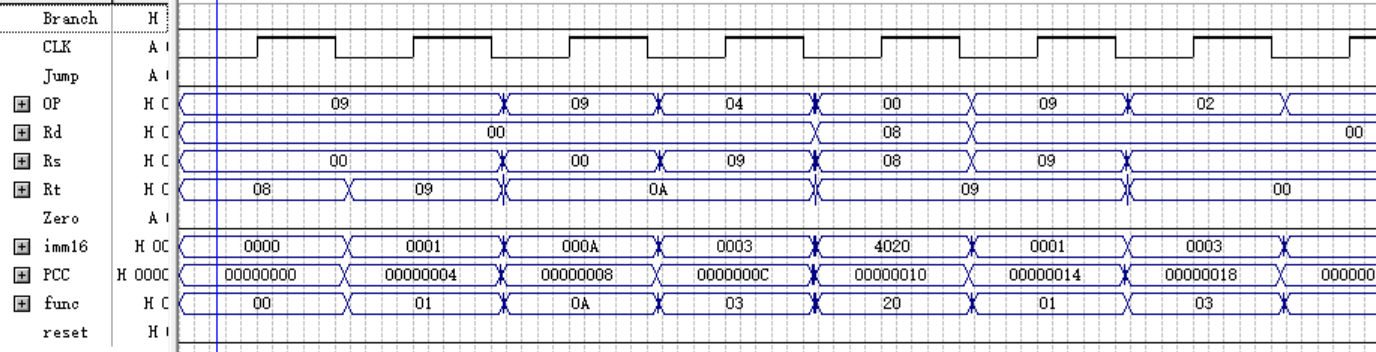


1. 对于不同的指令,PC的跳转有所区别



1. 仿真实验

通过检测PC的值可知，PC的自增是正常的。



同时其他控制信号也处在在时钟下降沿更新的状态。

**题 目： 译码器实验**

**学 院：**

**专 业：**

**学 号：**

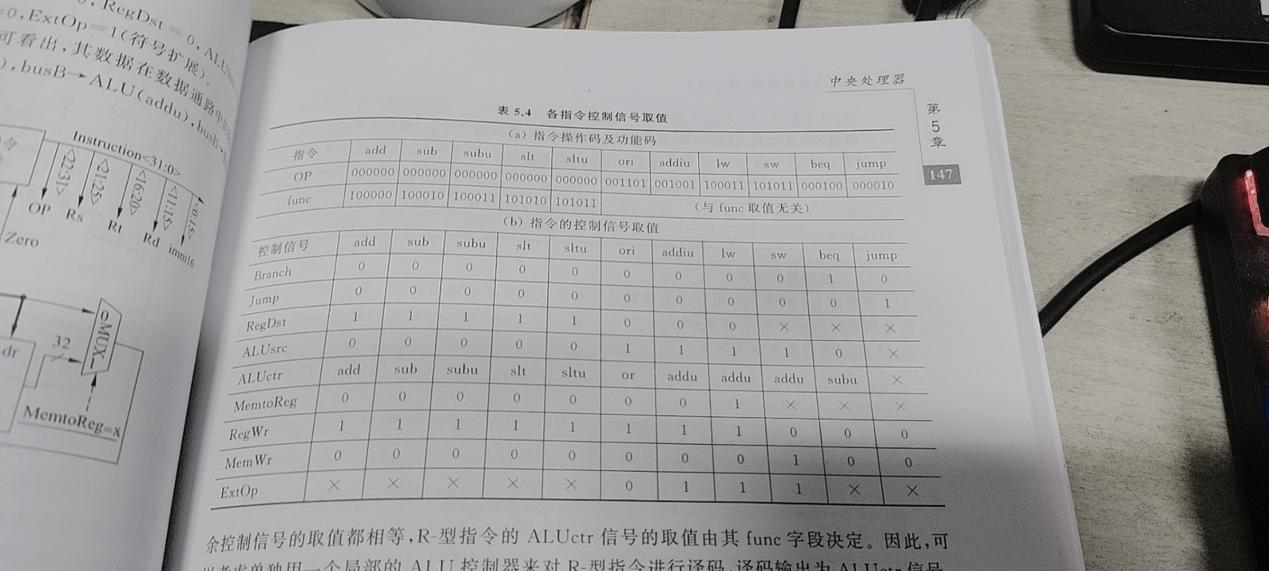
**姓 名：**

**指导教师：**

一、实验目的和原理

实验目的：实现单周期CPU当中对指令的译码。

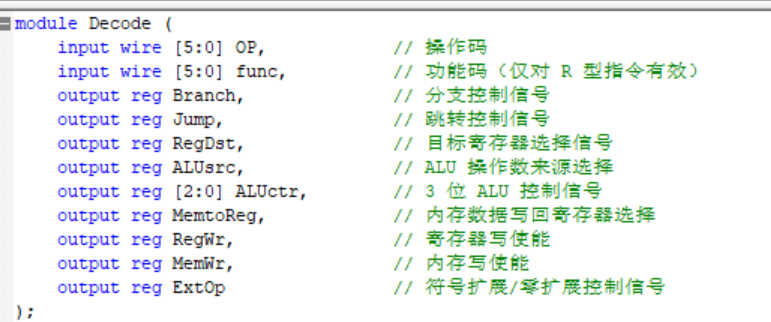
实验原理：



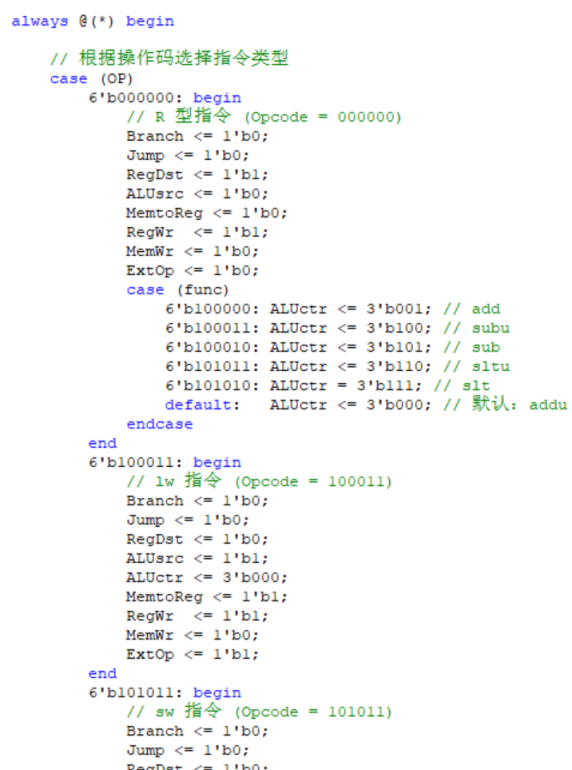
该表归纳了需要设计的指令当中各的控制信息。

二、代码展示

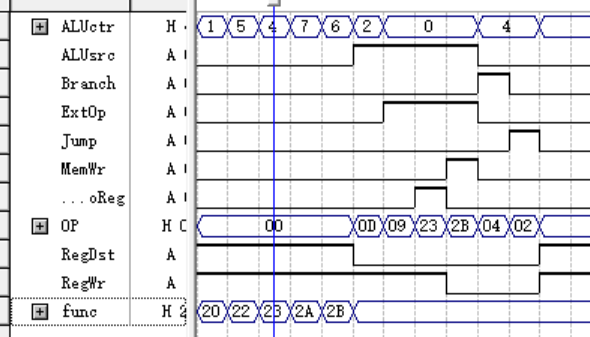
1、输入输出如下



1. 使用case语句即可



三、仿真实验



此处针对不同指令制造了若干数据，对比上面的表可知正确。