计算机系统结构实验 Lab3 实验报告

- 姓名：夏鸿驰
- 学号：520021910965

- 日期：2022年4月26日

目录

1. 实验概述

1.1 实验名称

- 简单的类 MIPS 单周期处理器功能部件的设计与实现

1.2 实验目的

1. 理解主控制部件或单元、ALU 控制单元、ALU 单元的原理
2. 熟悉所需的 Mips 指令集
3. 使用 Verilog HD 设计与实现主控制器部件（Ctr）
4. 使用 Verilog 设计与实现 ALU 控制器部件（ALUCtr）
5. ALU 功能部件的实现
6. 使用 Vivado 进行功能部件的行为仿真

2. 实验步骤

2.1 新建工程

2.1.1 用与前两个lab一样的步骤创建一个名为lab03的工程即可

2.2 构建主控制单元模块

2.2.1 模块描述

主控制单元（Ctr）的输入为指令的 opCode 字段，操作码经过 Ctr 的译码，给 ALUCtr，Data Memory，Registers，Muxs 等功能单元输出正确的控制信号。

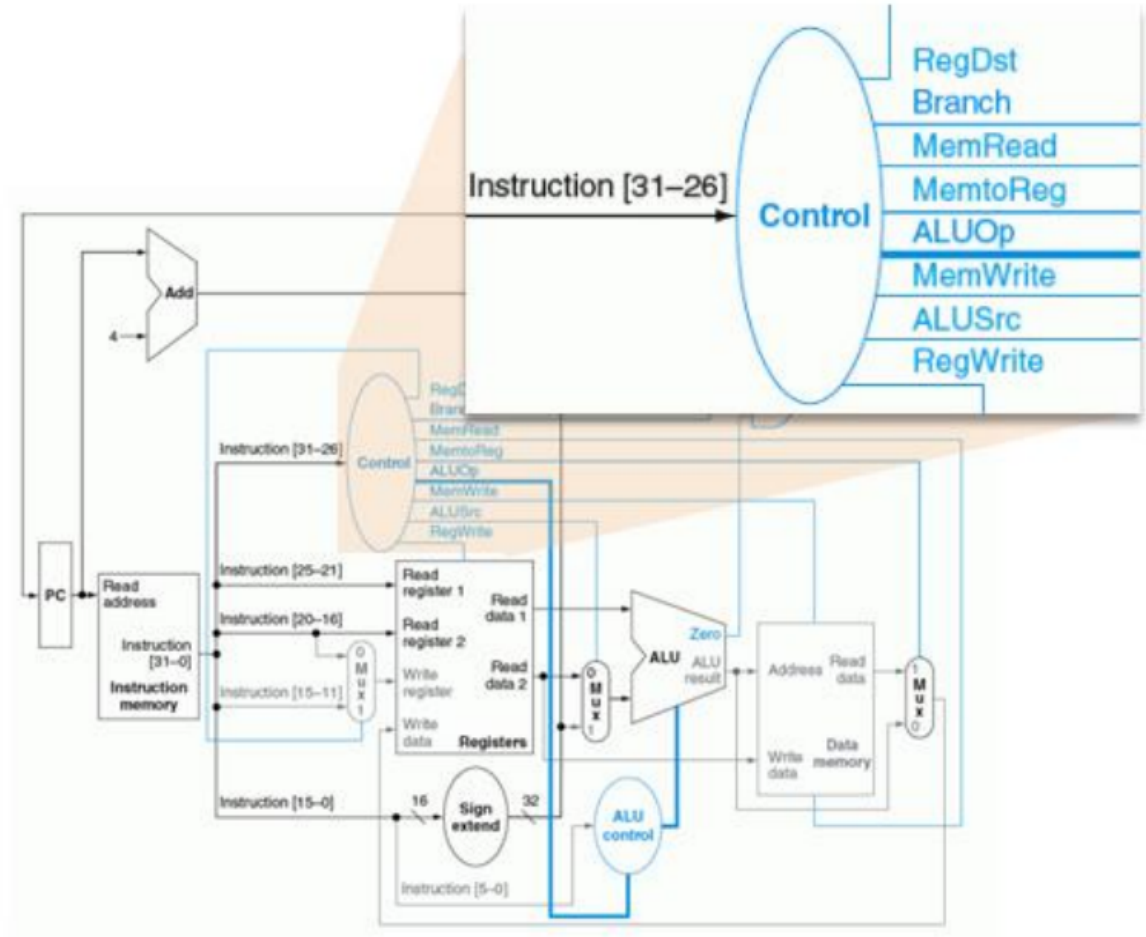


图 2. 简单的 Mips 处理器主控制器单元模块的 IO 定义

相应的译码规则为：

Input or output	Signal name	R-format	lw	sw	beq
Inputs	Op5	0	1	1	0
	Op4	0	0	0	0
	Op3	0	0	1	0
	Op2	0	0	0	1
	Op1	0	1	1	0
	Op0	0	1	1	0
Outputs	RegDst	1	0	X	X
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

图 3. 主控制模块的真值表

注意：Jump 指令编码是 000010，Jump 信号输出 1，其余输出 0

2.2.2 编写译码功能

新建source文件Ctr.v，并输入以下verilog代码：

```
module Ctr(
    input [5:0] OpCode,
    output RegDst,
    output ALUSrc,
    output MemToReg,
    output RegWrite,
    output MemRead,
    output MemWrite,
    output Branch,
    output [1:0] ALUOp,
    output Jump
    );

    reg regDst;
    reg aluSrc;
    reg memtoReg;
    reg regWrite;
    reg memRead;
    reg memWrite;
    reg branch;
    reg [1:0] aluOp;
    reg jump;

    always @ (OpCode)
    begin
        case(OpCode)
```

```
            6'b000000: //R type
            begin
                regDst = 1;
                aluSrc = 0;
                memtoReg = 0;
                regWrite = 1;
                memRead = 0;
                memWrite = 0;
                branch = 0;
                aluOp = 2'b10;
                jump = 0;
            end

            6'b100011: //lw
            begin
                regDst = 0;
                aluSrc = 1;
                memtoReg = 1;
                regWrite = 1;
                memRead = 1;
                memWrite = 0;
                branch = 0;
                aluOp = 2'b00;
                jump = 0;
            end

            6'b101011: //sw
            begin
                regDst = 0;
                aluSrc = 1;
                memtoReg = 0;
                regWrite = 0;
                memRead = 0;
                memWrite = 1;
                branch = 0;
                aluOp = 2'b00;
                jump = 0;
            end

            6'b000100: //beq
            begin
                regDst = 0;
                aluSrc = 0;
                memtoReg = 0;
                regWrite = 0;
                memRead = 0;
                memWrite = 0;
                branch = 1;
                aluOp = 2'b01;
                jump = 0;
            end

            6'b000010: //J
            begin
                regDst = 0;
                aluSrc = 0;
                memtoReg = 0;
                regWrite = 0;
```

```
                memRead = 0;
                memWrite = 0;
                branch = 0;
                aluOp = 2'b00;
                jump = 1;
            end

            default:
            begin
                regDst = 0;
                aluSrc = 0;
                memtoReg = 0;
                regWrite = 0;
                memRead = 0;
                memWrite = 0;
                branch = 0;
                aluOp = 2'b00;
                jump = 0;
            end
        endcase
    end

    assign RegDst = regDst;
    assign ALUSrc = aluSrc;
    assign MemToReg = memtoReg;
    assign RegWrite = regWrite;
    assign MemRead = memRead;
    assign MemWrite = memWrite;
    assign Branch = branch;
    assign ALUOp = aluOp;
    assign Jump = jump;

endmodule
```

2.2.3 主控制单元模块功能仿真

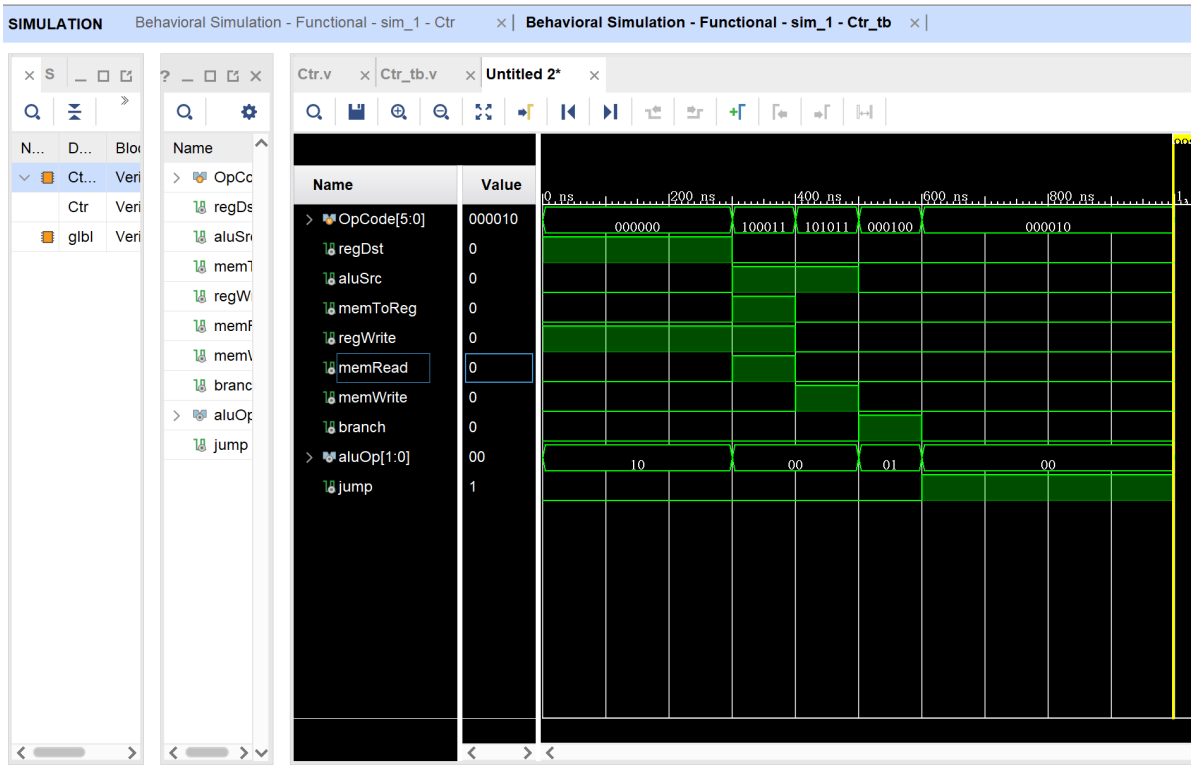
新建激励文件Ctr_tb.v，并输入以下verilog代码：

```
module Ctr_tb(

    );
    reg [5:0] OpCode;
    wire regDst;
    wire aluSrc;
    wire memToReg;
    wire regWrite;
    wire memRead;
    wire memWrite;
    wire branch;
    wire [1:0] aluOp;
    wire jump;

    Ctr c0 (
        .OpCode(OpCode),
        .RegDst(regDst),
        .ALUSrc(aluSrc),
        .MemToReg(memToReg),
        .RegWrite(regWrite),
```

```
        .MemRead(memRead),
        .MemWrite(memWrite),
        .Branch(branch),
        .ALUOp(aluOp),
        .Jump(jump)
        );


    initial begin
        OpCode = 0;

        #100;

        #100 OpCode = 6'b000000;//R type

        #100 OpCode = 6'b100011;//lw

        #100 OpCode = 6'b101011;//sw

        #100 OpCode = 6'b000100;//beq

        #100 OpCode = 6'b000010;//J
    end

endmodule
```

2.2.4 获取仿真图像

经过仿真，得到以下图像：



2.2.5 仿真结果分析

观察图像，可知仿真结果与逻辑相一致，目标模块达到了预想的结果，模块构建成功。

2.3 构建ALU控制单元模块

2.3.1 模块描述

算数逻辑单元 ALU 的控制单元（ALUCtr）是根据主控制器的 ALUOp 控制信号来判断指令类型，并依据指令的后 6 位区分 R 型指令。综合这两种输入，以控制 ALU 做正确操作。

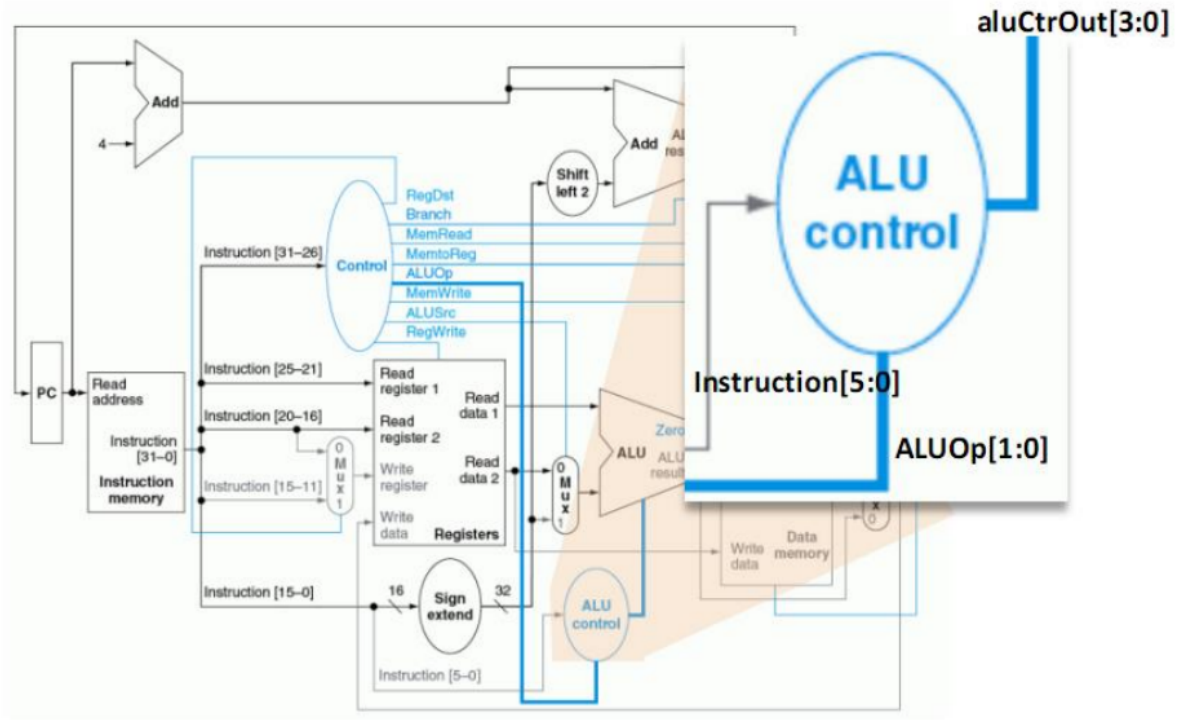


图 2. ALU 控制器模块的 IO 定义

相应的译码规则为：

ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	0110
1	X	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	X	X	X	0	1	0	0	0000
1	X	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

图 5. ALU 控制单元输入输出真值表

2.3.2 编写译码功能

新建source文件ALUCtr.v，并输入以下verilog代码：

```
`timescale 1ns / 1ps
module ALUCtr(
    input [1:0] ALUOp,
    input [5:0] Funct,
    output [3:0] ALUCtr
    );
    reg [3:0] aluCtr;
    always @ (ALUOp or Funct)
    begin
        casex ({ALUOp, Funct})
            8'b00xxxxxx: aluCtr = 4'b0010;
            8'b01xxxxxx: aluCtr = 4'b0110;
            8'b1xxx0000: aluCtr = 4'b0010;
```

```
            8'b1xxx0010: aluCtr = 4'b0110;
            8'b1xxx0100: aluCtr = 4'b0000;
            8'b1xxx0101: aluCtr = 4'b0001;
            8'b1xxx1010: aluCtr = 4'b0111;
            default: aluCtr = 4'b0000;
        endcase
    end
    assign ALUCtr = aluCtr;
endmodule
```

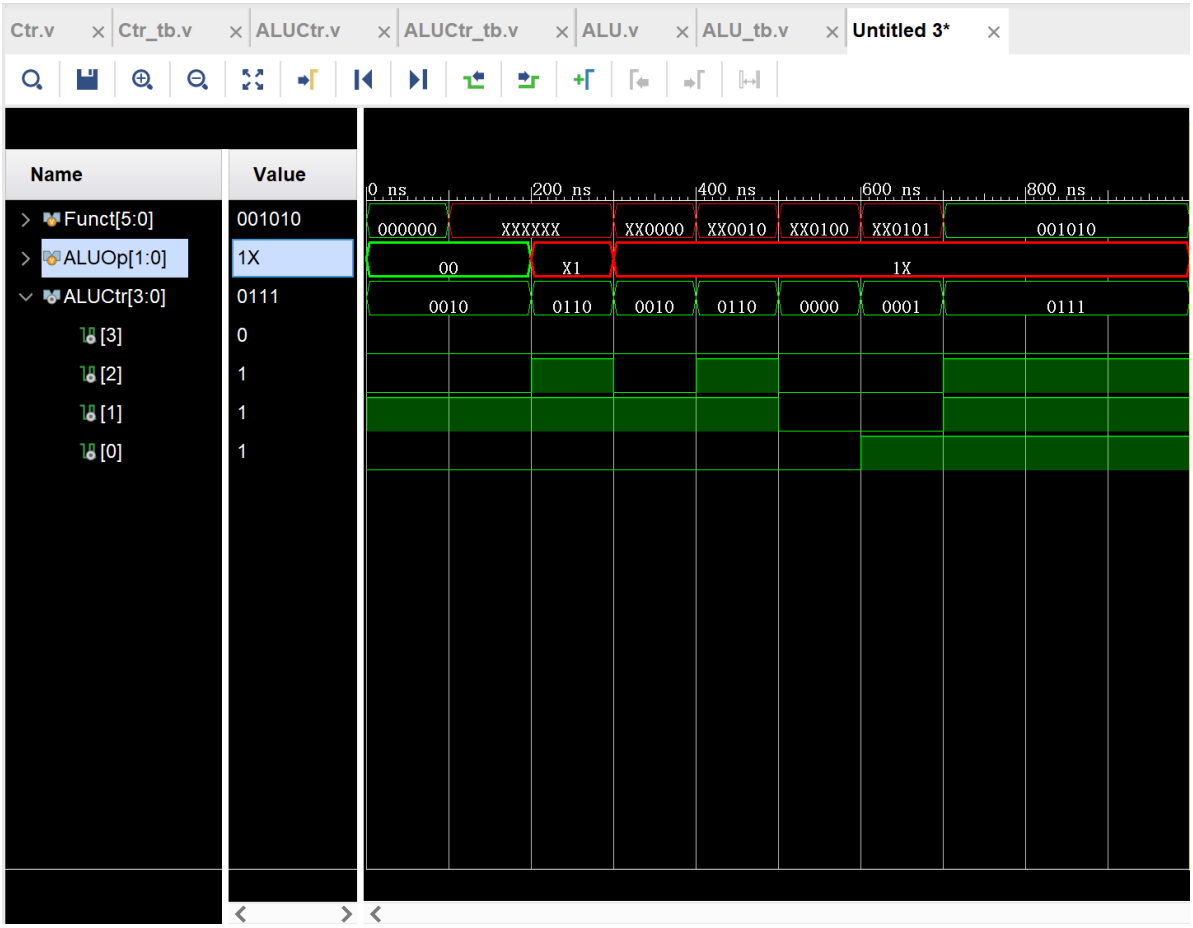
2.3.3 主控制单元模块功能仿真

新建激励文件ALUCtr_tb.v，并输入以下verilog代码：

```
`timescale 1ns / 1ps
module ALUCtr_tb(

    );
    reg [5:0] Funct;
    reg [1:0] ALUOp;
    wire [3:0] ALUCtr;

    ALUCtr aluctr(
        .Funct(Funct),
        .ALUOp(ALUOp),
        .ALUCtr(ALUCtr)
    );

    initial begin
        Funct = 6'b000000;
        ALUOp = 2'b00;
        #100;
        Funct = 6'bxxxxxx;
        ALUOp = 2'b00;
        #100;
        Funct = 6'bxxxxxx;
        ALUOp = 2'bx1;
        #100;
        Funct = 6'bxx0000;
        ALUOp = 2'b1x;
        #100;
        Funct = 6'bxx0010;
        ALUOp = 2'b1x;
        #100;
        Funct = 6'bxx0100;
        ALUOp = 2'b1x;
        #100;
        Funct = 6'bxx0101;
        ALUOp = 2'b1x;
        #100;
        Funct = 6'b1010;
        ALUOp = 2'b1x;
    end
endmodule
```

2.3.4 获取仿真图像

经过仿真，得到以下图像：



2.3.5 仿真结果分析

观察图像，可知仿真结果与逻辑相一致，目标模块达到了预想的结果，模块构建成功。

2.4 构建ALU模块

2.4.1 模块描述

算术逻辑单元 ALU 根据 ALUCtr 的控制信号将两个输入执行与之对应的操作。ALURes 为输出结果。若减法操作 ALURes 的结果为 0 时，则 Zero 输出置为 1。

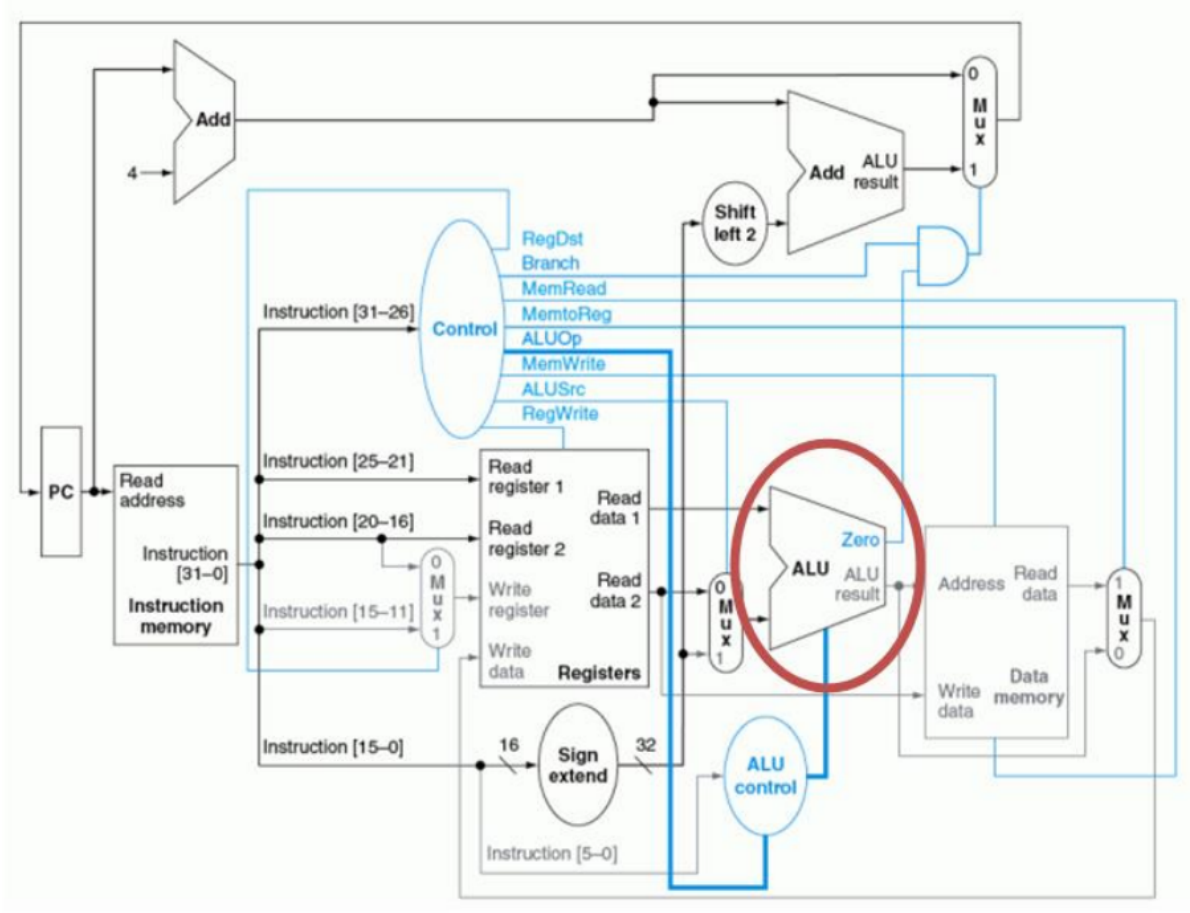


图 6. ALU 模块的 IO 定义

相应的对应关系规则为：

aluCtrOut[3:0]的值与 ALU 操作的对应关系如下：

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

注意：beq 时 实际是个减法操作

2.4.2 编写ALU功能

新建source文件ALU.v，并输入以下verilog代码：

注意slt判断时需要先转换成带符号数。

```
`timescale 1ns / 1ps


module ALU(
    input [31:0] input1,
```

```
    input [31:0] input2,
    input [3:0] ALUCtr,
    output zero,
    output [31:0] ALURes
    );

    reg Zero;
    reg [31:0] alures;
    reg signed [31:0] slt_input1;
    reg signed [31:0] slt_input2;

    always @ (input1 or input2 or ALUCtr)
    begin
        case(ALUCtr)
            4'b0000: //and
            begin
                alures = input1 & input2;
                if(alures == 0)
                    Zero = 1;
                else
                    Zero = 0;
            end

            4'b0001: //or
            begin
                alures = input1 | input2;
                if(alures == 0)
                    Zero = 1;
                else
                    Zero = 0;
            end

            4'b0010: //add
            begin
                alures = input1 + input2;
                if(alures == 0)
                    Zero = 1;
                else
                    Zero = 0;
            end

            4'b0110: //sub
            begin
                alures = input1 - input2;
                if(alures == 0)
                    Zero = 1;
                else
                    Zero = 0;
            end

            4'b0111: //slt
            begin
                slt_input1 = input1;
                slt_input2 = input2;
                if(slt_input1 < slt_input2)
                    alures = 1;
                else
                    alures = 0;
```

```
                if(alures == 0)
                    Zero = 1;
                else
                    Zero = 0;
            end

            4'b1100: //nor
            begin
                alures = ~(input1 | input2);
                if(alures == 0)
                    Zero = 1;
                else
                    Zero = 0;
            end

            default:
            begin
                alures = 0;
                Zero = 0;
            end
        endcase
    end
    assign ALURes = alures;
    assign zero = Zero;
endmodule

```

2.4.3 主控制单元模块功能仿真

新建激励文件ALU_tb.v，并输入以下verilog代码：

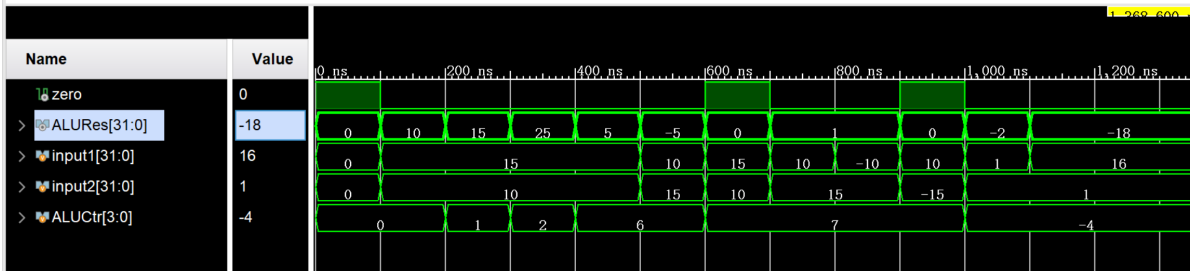
对指导书上的示例进行了一点修改，加入了slt判断负数的样例。

```
`timescale 1ns / 1ps

module ALU_tb(

    );
    reg [31:0] input1;
    reg [31:0] input2;
    reg [3:0] ALUCtr;
    wire zero;
    wire [31:0] ALURes;

    ALU alu(
        .input1(input1),
        .input2(input2),
        .ALUCtr(ALUCtr),
        .zero(zero),
        .ALURes(ALURes)
    );

    initial begin
        input1 = 0;
        input2 = 0;
        ALUCtr = 4'b0000;
        #100;
        input1 = 15;

```

```
        input2 = 10;
        #100;
        ALUCtr = 4'b0001;
        #100;
        ALUCtr = 4'b0010;
        #100;
        ALUCtr = 4'b0110;
        #100;
        input1 = 10;
        input2 = 15;
        #100;
        input1 = 15;
        input2 = 10;
        ALUCtr = 4'b0111;
        #100;
        input1 = 10;
        input2 = 15;
        #100;
        input1 = -10;
        input2 = 15;
        #100;
        input1 = 10;
        input2 = -15;
        #100;
        input1 = 1;
        input2 = 1;
        ALUCtr = 4'b1100;
        #100;
        input1 = 16;
        #100;
    end
endmodule

```

2.4.4 获取仿真图像

经过仿真，得到以下图像：



2.4.5 仿真结果分析

观察图像，可知仿真结果与逻辑相一致，目标模块达到了预想的结果，模块构建成功。

3. 实验心得

3.1 实验重难点

- 本实验的重难点通过理解实验指导书样例中的主控制单元模块的构建，来构建ALU控制单元模块和ALU模块，和独立完成相应激励代码的编写和调试。

3.2 实验感想

- 在本次实验过程中，从构建主控制单元模块到构建ALU控制单元模块，再到构建ALU模块让我感到十分的新奇，也给了我一种聚沙成塔和完成任务的成就感。此外，在实验中遇到的困难都会在微信群里得到老师及时的反馈，所以要在这里感谢老师的付出。

4. 参考资料

[1] 2022计算机系统结构实验指导书lab3