组成原理实验课程第_4_次实报告

实验名称	ALU 模块实现			班级	李涛老师
学生姓名	曹瑜	学号	2212794	指导老师	董前琨
实验地点	实验楼 A 区 306		实验时间	2024.05.16	

1、实验目的

1.熟悉 MIPS 指令集中的运算指令，学会对这些指令进行归纳分类；

2.了解 MIPS 指令结构；

3.熟悉并掌握 ALU 的原理、功能和设计；

4.进一步加强运用 verilog 语言进行电路设计的能力；

5.为后续设计 cpu 的实验打下基础；

2、实验内容说明

1.学习 MIPS 指令集，熟知指令类型，了解指令功能和编码，归纳基础的 ALU 运算指令。

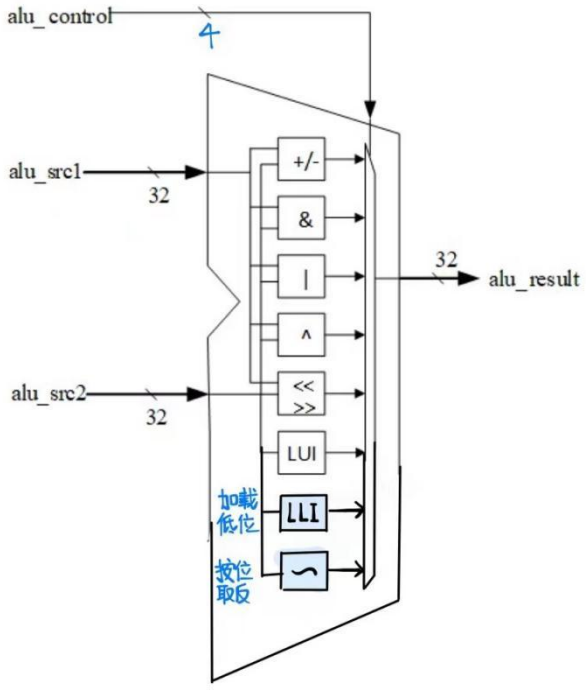
2.归纳确定自己本次实验中准备实现的 ALU 运算，除表 5.1 中的 11 种运算，另外补充至少三种不同类型运算(比较运算、位运算、数据加载运算等)；

3.自行设计本次实验的方案，画出结构框图，大致结构框图如图 5.1。图 5.1 中的操作码位数和类型请自行设计，可以设计为独热码（一位有效编码）或二进制编码；

4.完成上箱验证和实验报告撰写；

3、实验原理图

加减运算为调用 adder 加法器直接运算的，有符号比较的小于/大于置位，是利用减法结果比较的；



对于有符号比较的大于置位，是利用减法结果比较的，其比较的真值表如下：

有符号比较大于置位的真值表

源操作数 1 符号位		源操作数 2 符号位		结果符号位		判断	slt 结果
alu_src1[31]		alu_src2[31]		adder_result[31]			
0	正数	0	正数	0	正数	正>正	1
0	正数	0	正数	1	负数	正<正	0
0	正数	1	负数	X	无关	正>负	1
1	负数	0	正数	X	无关	负<正	0
1	负数	1	负数	0	正数	负>负	1
1	负数	1	负数	1	负数	负<负	0

由此可得有符号 32 位比较小于置位运算结果表达式:

~((alu_src1[31] & ~alu_src2[31]) | (~(alu_src1[31]^alu_src2[31]) & adder_result[31]))&(|(alu_src1-alu_src2))

not 操作为按位取反操作，对源操作数 2 按位取反，即~ alu_src2；

Lli 操作为低位加载数操作，即把源操作数 2 的低 16 位装载到 result 的低 16 位；

4、实验步骤

[alu]模块：

在 alu 模块里新增三个 ALU 控制信号，对应三个新增操作：

有符号大于置位操作（sgt）、按位取反（not）、低位加载数（lli）

```
wire alu_sgt;  //有符号大于置位
wire alu_not;  //按位取反
wire alu_lli;  //低位加载
```

此时共有 15 个操作，故将控制信号改为 4 位

```
module alu(
    input  [3:0] alu_control,  // ALU控制信号
    input  [31:0] alu_src1,    // ALU操作数1,为补码
    input  [31:0] alu_src2,    // ALU操作数2，为补码
    output [31:0] alu_result   // ALU结果
    );
```

并修改 4 位控制信号不同值和指令的对应关系

```
assign alu_add  = alu_control==4'b0001;
assign alu_sub  = alu_control==4'b0010;
assign alu_slt  = alu_control==4'b0011;
assign alu_sltu = alu_control==4'b0100;
assign alu_and  = alu_control==4'b0101;
assign alu_nor  = alu_control==4'b0110;
assign alu_or   = alu_control==4'b0111;
assign alu_xor  = alu_control==4'b1000;
assign alu_sll  = alu_control==4'b1001;
assign alu_srl  = alu_control==4'b1010;
assign alu_sra  = alu_control==4'b1011;
assign alu_lui  = alu_control==4'b1100;

assign alu_sgt  = alu_control==4'b1101;
assign alu_not  = alu_control==4'b1110;
assign alu_lli  = alu_control==4'b1111;
```

新增 3 个新操作的定义：

```
wire [31:0] sgt_result;
wire [31:0] not_result;
wire [31:0] lli_result;
```

sgt 为有符号大于置位操作，先写出真值表，可得最后结果为对 slt 的判断按位取反后再去除两个操作数相等的情况

最后的表达式为：

~((alu_src1[31] & ~alu_src2[31]) | (~(alu_src1[31]^alu_src2[31]) & adder_result[31]))& (|(alu_src1-alu_src2))

```
//sgt结果
//adder_src1[31] adder_src2[31] adder_result[31]
//      0              1           X(0或1)       "正-负"，显然大于成立
//      0              0             1           相减为负，说明不大于
//      0              0             0           相减为正，说明大于
//      1              1             1           相减为负，说明不大于
//      1              1             0           相减为正，说明大于
//      1              0           X(0或1)       "负-正"，显然大于不成立

assign sgt_result[31:1] = 31'd0;
assign sgt_result[0]    = ~((alu_src1[31] & ~alu_src2[31]) | (~(alu_src1[31]^alu_src2[31]) & adder_result[31]))
                          & (|(alu_src1-alu_src2)); //对slt的判断按位取反后再去除两个操作数相等的情况
```

not 为按位取反操作，实际为对源操作数 2 进行按位取反，即 ~ alu_src2；

lli 为低位加载数操作，即把源操作数 2 的低 16 位装载到结果的低 16 位；

```
assign not_result = ~ alu_src2;                 //按位取反操作对源操作数2进行按位取反
assign lli_result = {16'd0,alu_src2[15:0]};   // 装载到低16位
```

补充新增 3 种运算的结果选择输出

```
                    alu_sgt          ? sgt_result :
                    alu_not          ? not_result :
                    alu_lli          ? lli_result :
                    32'd0;
```

[alu_display]模块：

将 alu 控制信号更新为 4 位

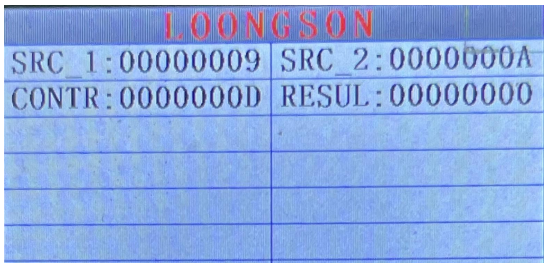
```
                //-----{调用ALU模块}begin
                    reg   [3:0] alu_control;  // ALU控制信号
```

修改 alwys 块中对 input_value 的截取位数，此时只需截取低四位

```
//-----{从触摸屏获取输入}begin
//根据实际需要输入的数修改此小节，
//建议对每一个数的输入，编写单独一个always块
    //当input_sel为00时，表示输入数控制信号，即alu_control
    always @(posedge clk)
    begin
        if (!resetn)
        begin
            alu_control <= 4'd0;
        end
        else if (input_valid && input_sel==2'b00)
        begin
            alu_control <= input_value[3:0];
        end
    end
```

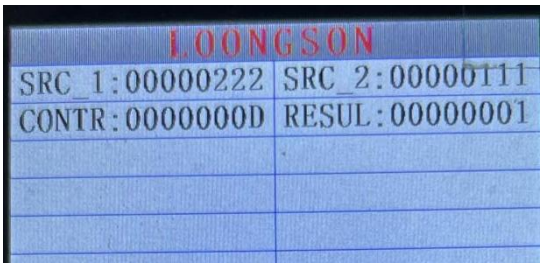
5、实验结果分析

新增操作 1：sgt（有符号置位大于）



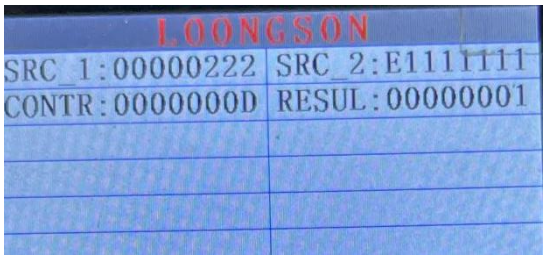
正数<正数

操作数 1：　9 (16)
操作数 2：A (16)
操作：D (16)　对应 1101 操作 sgt
9 < A
Result：0



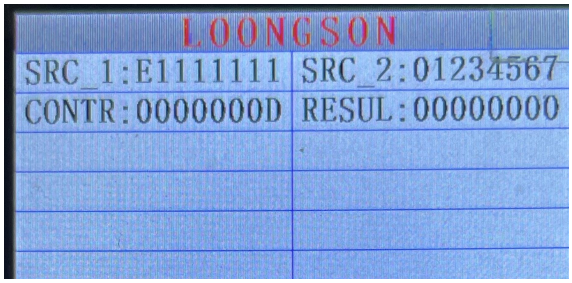
正数>正数

操作数 1：　222 (16)
操作数 2：111 (16)
操作：D (16)　对应 1101 操作 sgt
222 > 111
Result：1



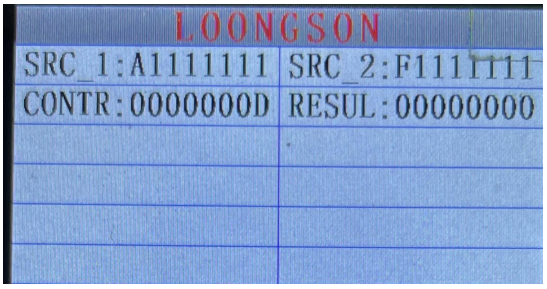
正数>负数

操作数 1：　222 (16)
操作数 2：E1111111 (16)
操作：D (16)　对应 1101 操作 sgt
222　>　E1111111
Result：1



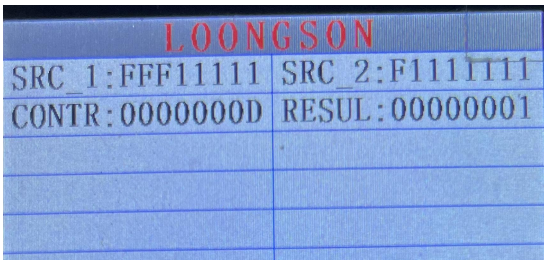
负数<正数

操作数 1：　E1111111 (16)
操作数 2：　1234567 (16)
操作：D (16)　对应 1101 操作 sgt
E1111111 < 1234567
Result：0



负数<负数

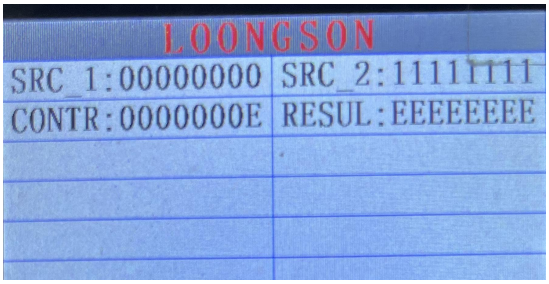
操作数 1: A1111111 (16)
操作数 2: F1111111 (16)
操作：D (16)　对应 1101 操作 sgt
A111111　<　E1111111
Result：0



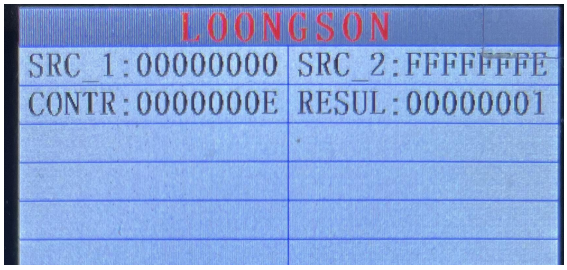
负数>负数

操作数 1: FFF11111 (16)
操作数 2: F1111111 (16)
操作：D (16)　对应 1101 操作 sgt
A111111　>　E1111111
Result：1

新增操作 2：not（按位取反）

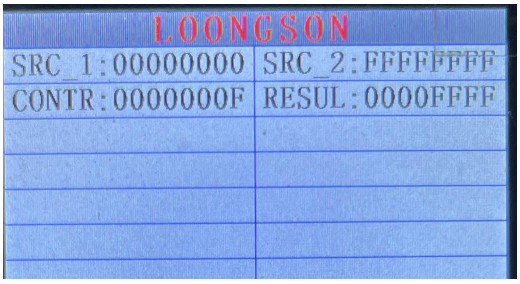


操作数 2: 11111111 (16)
0001 0001 0001 0001 0001 0001 0001 0001
操作：E (16)　对应 1110 操作 not
按位取反
Result：EEEEEEEE (16)
1110 1110 1110 1110 1110 1110 1110 1110



操作数 2: FFFFFFFE (16)
1111 1111 1111 1111 1111 1111 1111 1110
操作：E (16)　对应 1110 操作 not
按位取反
Result：00000001 (16)
0000 0000 0000 0000 0000 0000 0000 0001

新增操作 3：lli（低位加载数）

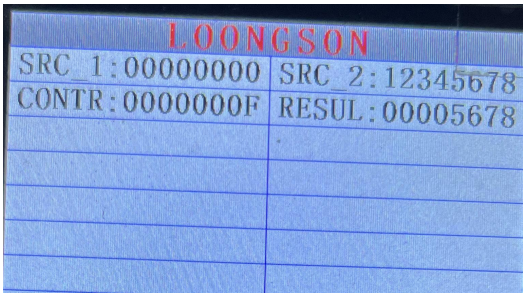


操作数 2: FFFFFFFF (16)

操作：F (16)　对应 1111 操作 lli

低位加载数

Result：0000FFFF (16)



操作数 2: 12345678 (16)

操作：F (16)　对应 1111 操作 lli

低位加载数

Result：00005678 (16)

6、总结感想

本次实验中学习 MIPS 指令集，归纳基础的 ALU 运算指令类型，熟悉了指令编码和功能，在理解了已有的几种指令后，自行新增了 3 种不同类型的指令并成功上箱验证，对 MIPS 指令有了更深入的认知和理解。