# **组成原理课程第 四 次实验报告**

# **实验名称：ALU**

学号： 2312141 姓名： 张德民 班次： 李涛老师班

## 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。

## 实验内容说明

请根据实验指导手册中完成实验四ALU实验的代码功能验证，并做以下改进：

1、原有的操作码为12位，请压缩到操作码控制型号位宽为4位。

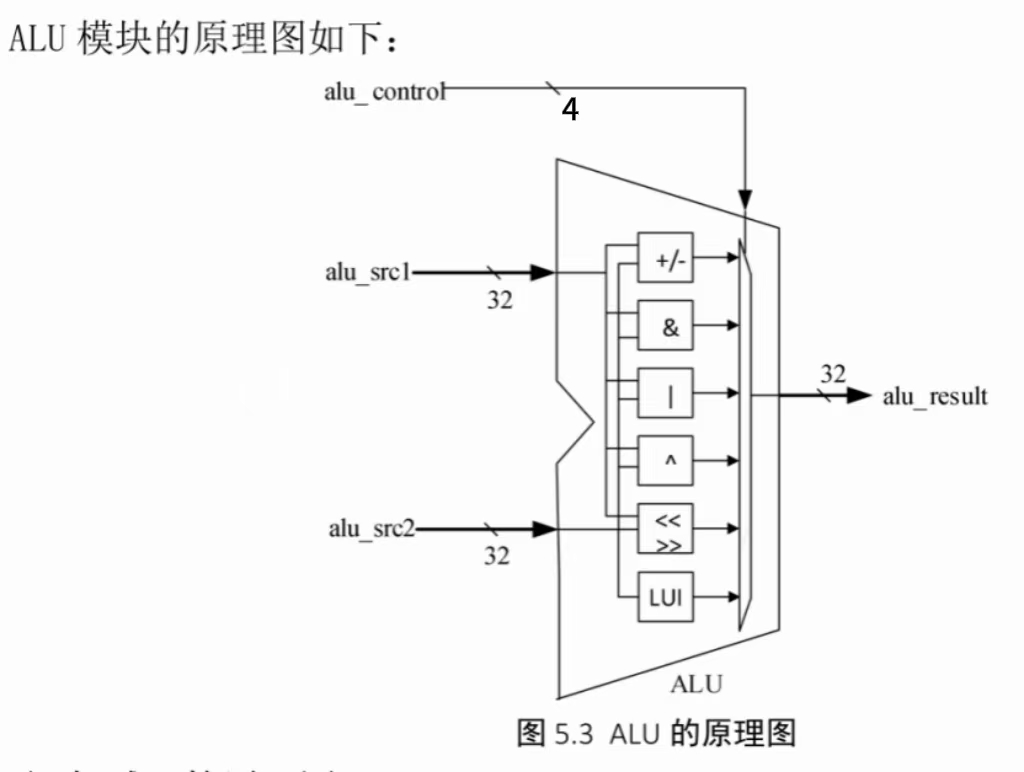
2、在操作码调整到4位之后，应该能支持15种不同运算，请添加至少三种运算功能（有 符号数比较和无符号数比较的大于置位运算、一种未实现的位运算），然后上实验箱验 证（可以不用仿真）。

3、实验报告中的原理图就用图5.3即可，不再是顶层模块图。

4、添加的三种运算上箱需要有验证照片。

提交时文件名格式为“学号\_姓名\_组成原理第四次实验.pdf”。注意实验报告中要有介 绍分析的内容，针对实验箱照片，要解释图中信息，是否验证成功。

## 实验原理图

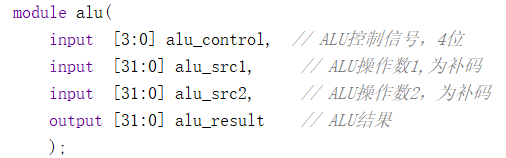


此原理图和实验指导手册上的基本一样，但是alu\_control的位数修改了。我们需要传入一个4位的控制信号，两个32位的操作数，经过alu运算以后，就可以得到一个32位的结果。

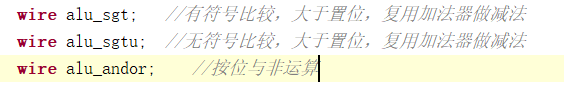
## 实验步骤

这里我添加了三个功能，分别是有符号数比较的大于置位和无符号数比较的大于置位，以及按位与非运算，并把控制信号由12位压缩为4位，这里我主要介绍我修改的地方。

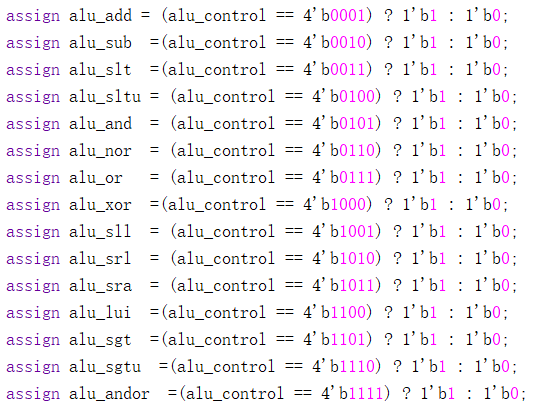
### 修改alu.v代码



首先，需要把alu控制信号由原来的12位修改为4位。



然后添加三个wire类型的变量，alu\_sgt,alu\_sgtu和alu\_sla,分别对应我添加的三种新的运算：有符号比较，大于置位，无符号比较，大于置位以及按位与非。



判断是哪一种运算这一步，由于我把12位压缩成了4位，所以之前的赋值方法就不能用了，需要改为我上边的赋值代码,刚好使用4位二进制数:

1：加法

2：减法

3：有符号比较，小于置位

4：无符号比较，小于置位

5：与运算

6：或非运算

7：或运算

8：异或运算

9：逻辑左移

A：逻辑右移

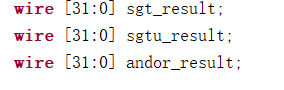
B：算术右移

C：高位加载

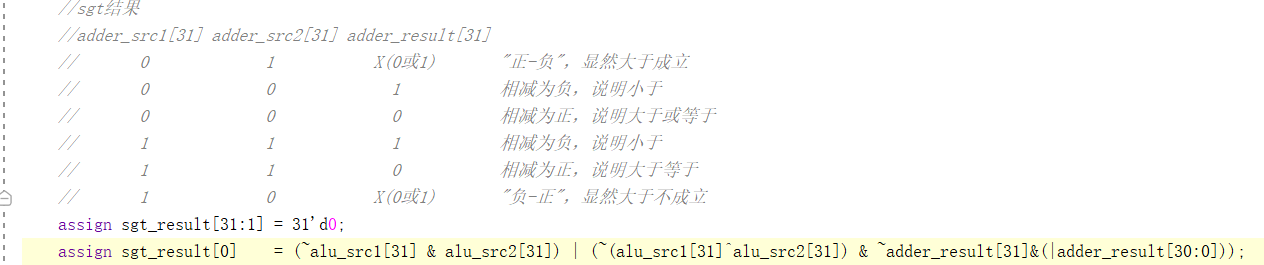
D：有符号比较，大于置位

E：无符号比较，大于置位

F：与非运算

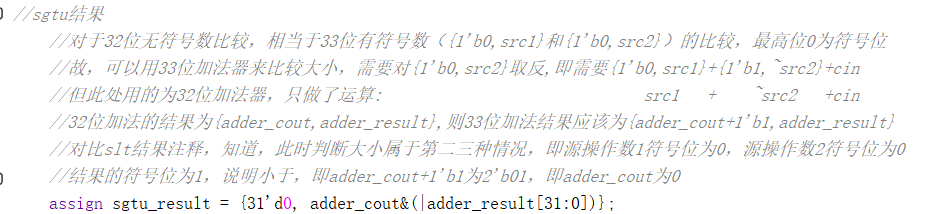


然后添加3个wire类型数据来存新的三种运算的结果。



此处实现了sgt的功能，这里我仿照小于置位的逻辑，然后把结果取反即可，但是有一点是需要注意的，小于置位不需要考虑相等的情况，因为符号位为1的一定是负数，但是反过来，符号位为0的数不一定是正数，还有可能是0，因此我需要排除相等这个情况。

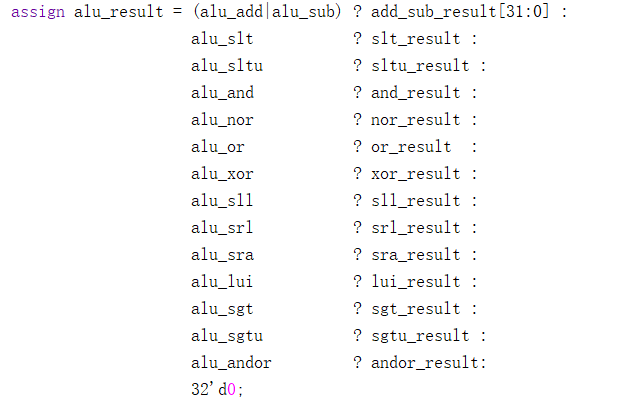
我把最后结果与(|adder\_result[30:0])进行与运算，这样的目的是保证源数据1减去源数据2的结果不是0。最后就实现了有符号数比较，大于置位的功能。



上图是sgtu的功能实现，这里我依旧是仿照了无符号数比较，小于置位的逻辑，并进行取反。然后和上边的处理一样，最后结果与(|adder\_result[31:0])进行与运算，避免出现等于的情况。

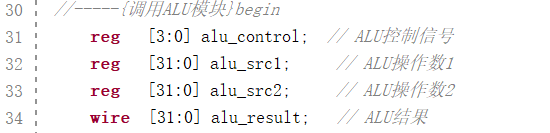


然后是一个新的位运算，我这里选择实现的是按位与非运算，实现的方法比较简单，直接用vivado自带的运算就实现了。

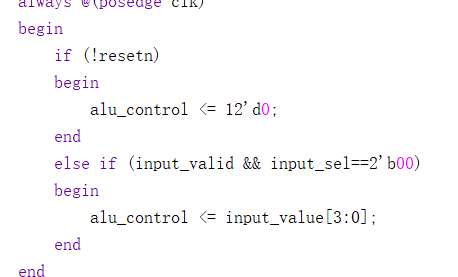


最后根据控制位来决定要输出哪一种运算的结果。

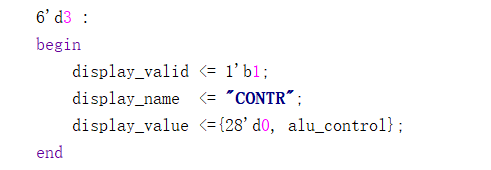
### alu\_display.v的修改



首先，把alu控制信号由12位改为4位。



在always语句里边也要改为4位。

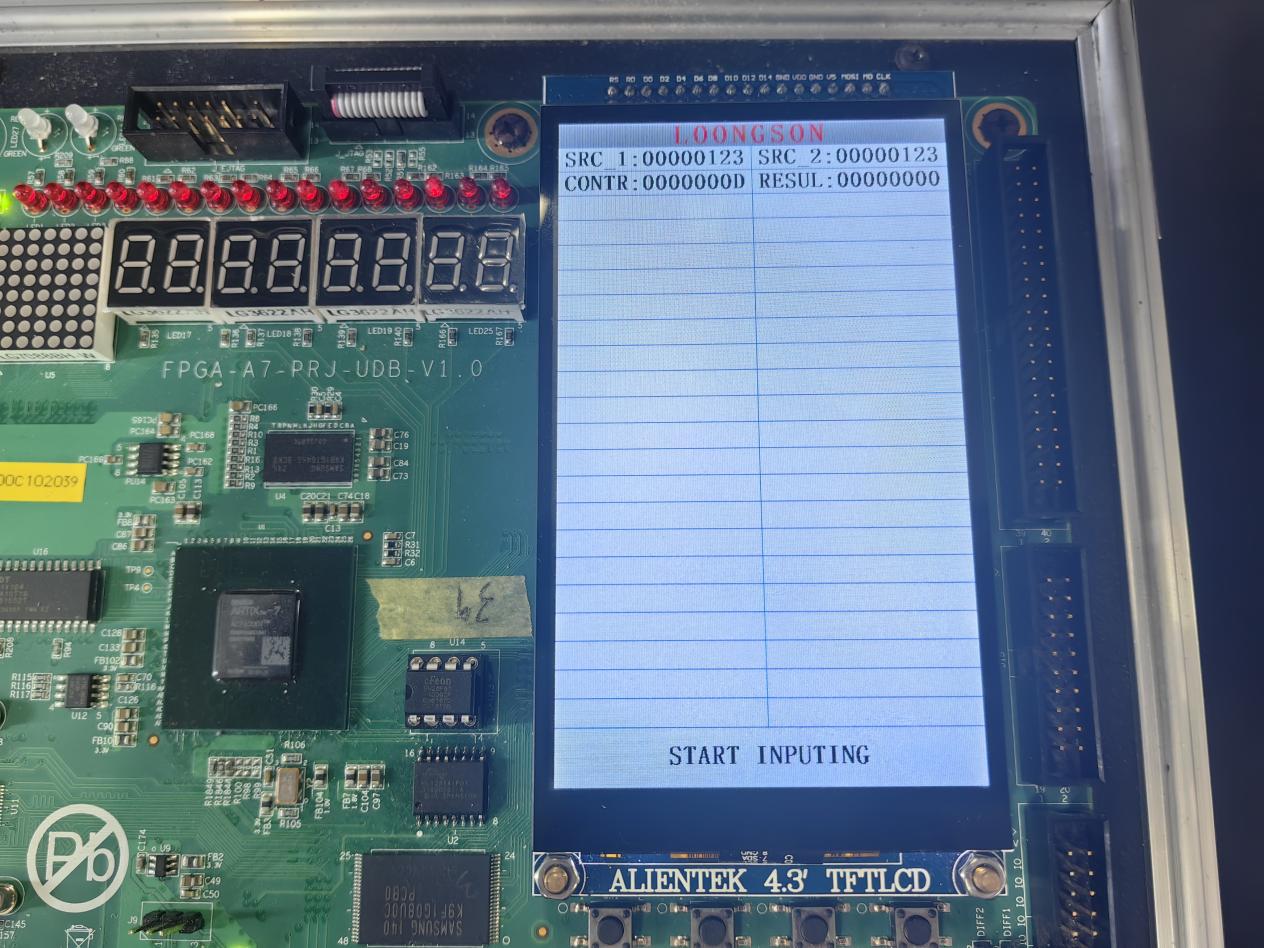


最后，在这个显示屏这里，由于压缩为4位，所以需要把前边的20个0增加到28个。

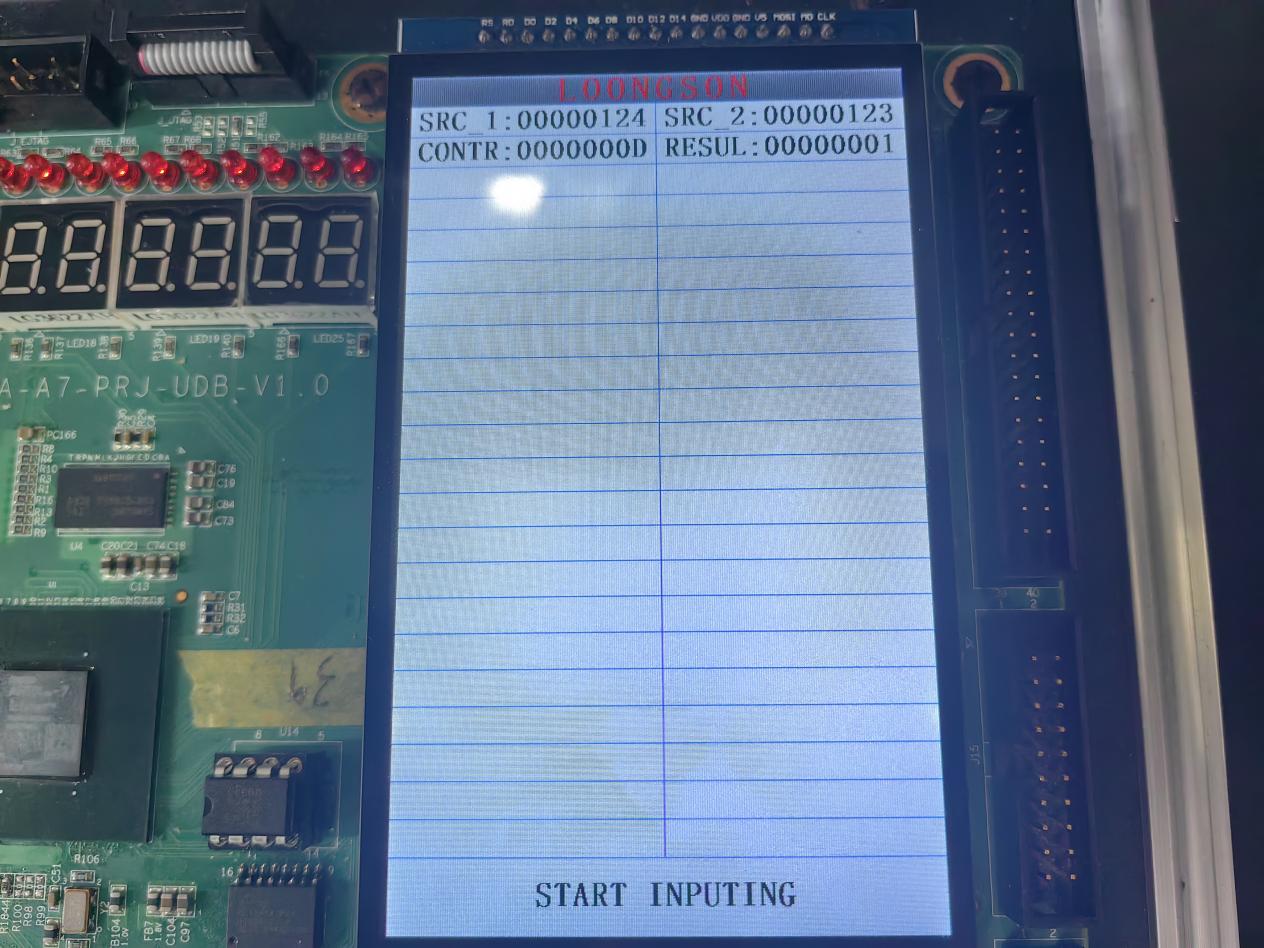
## 实验结果分析

### 有符号数比较，大于置位：

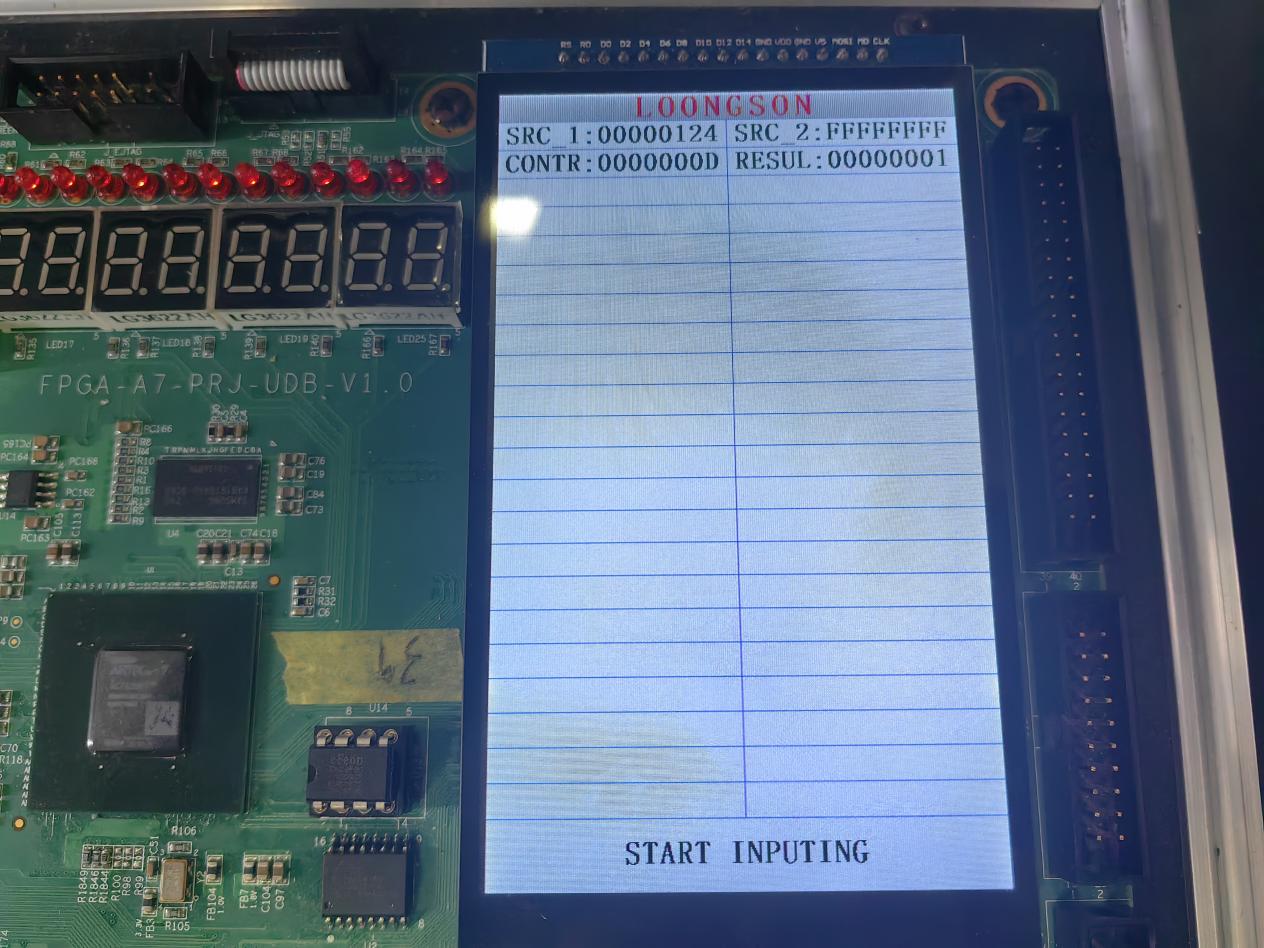
此时的控制信号是1101，也就是D。



如图，操作数1和2都是123，此时二者相等，所以结果为0，没有置位，正确。



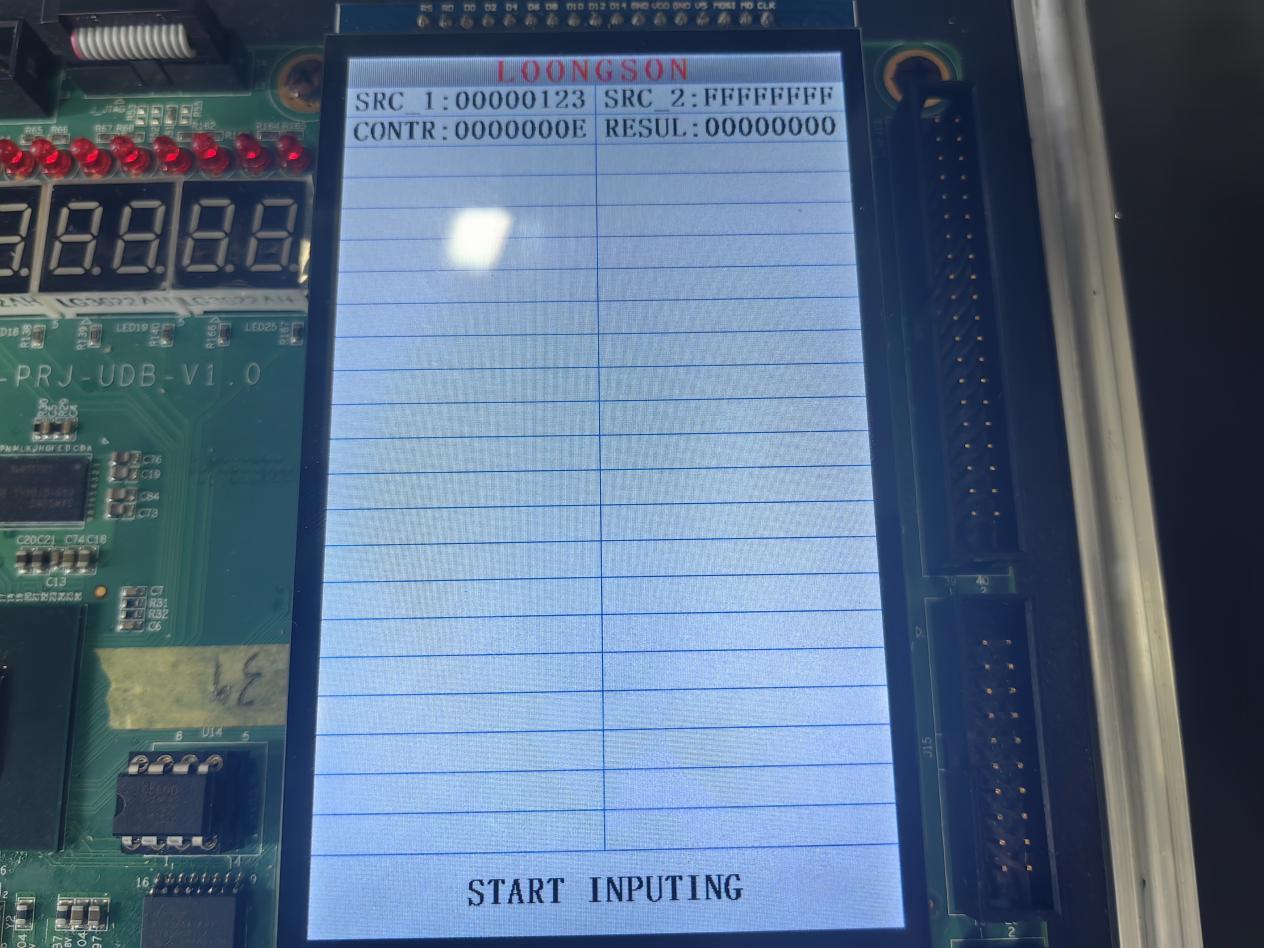
此时的操作数1是124，操作数2是123，操作数1大于2，则结果是1，置位了，答案正确。



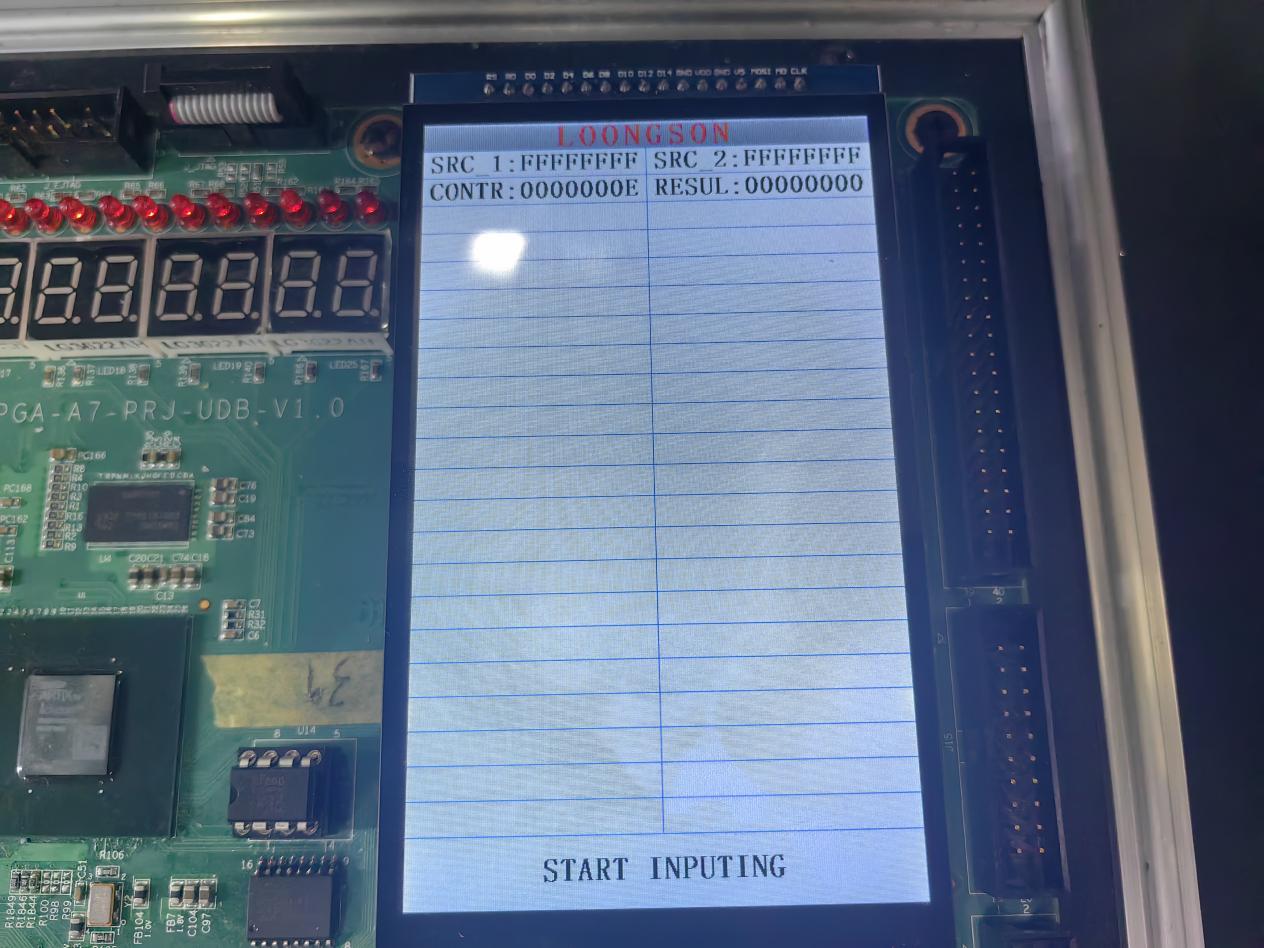
此时的操作数1是124，操作数2是ffffffff(-1)，很显然操作数1大于2，则结果是1，置位了，答案正确。

### 无符号数比较，大于置位

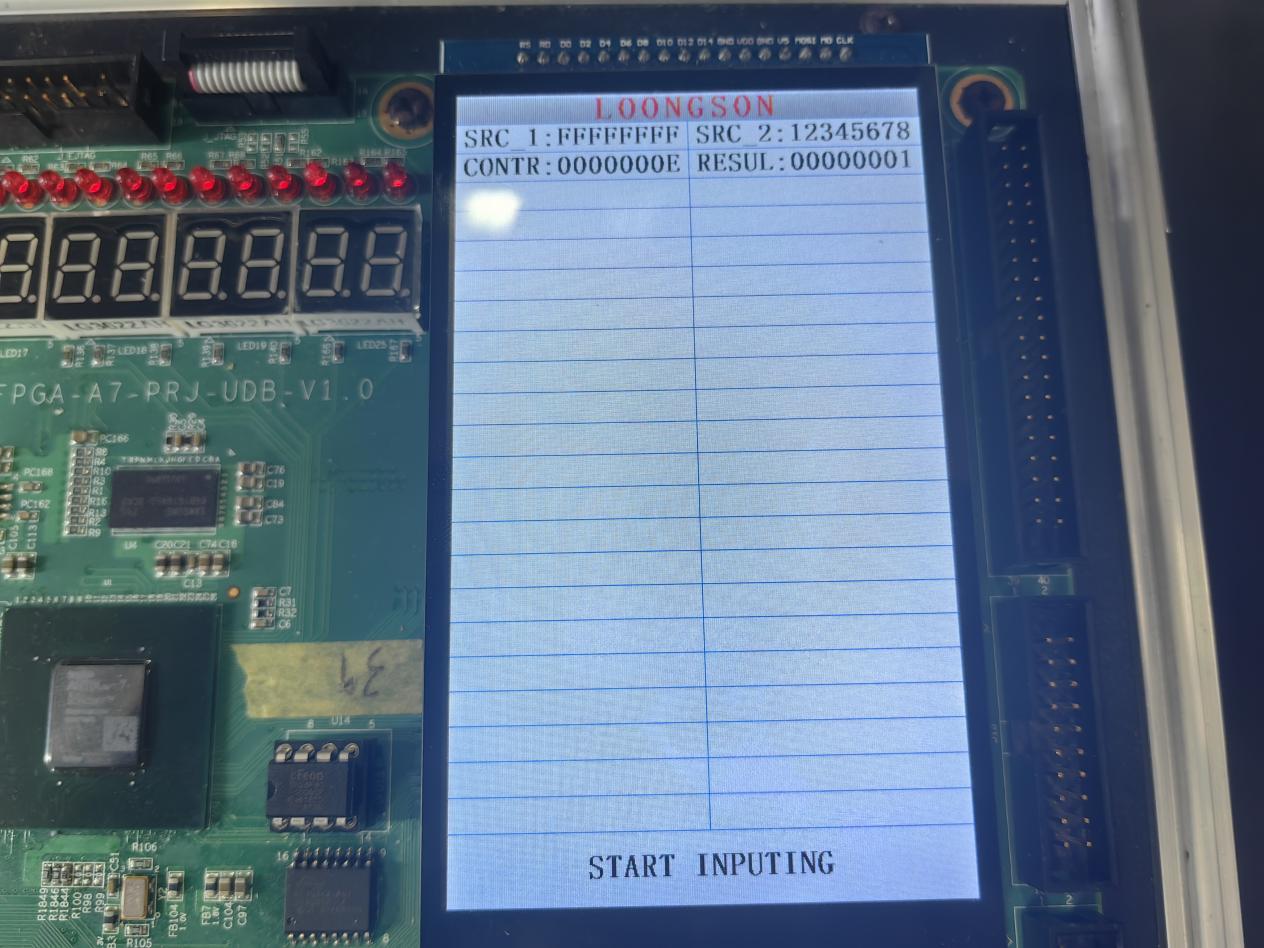
此时的控制信号是1110，也就是E



此时的操作数1是124，操作数2是ffffffff，由于2是无符号数，则2应该是2^32-1，很显然操作数1小于2，则结果为0，没有置位，答案正确。



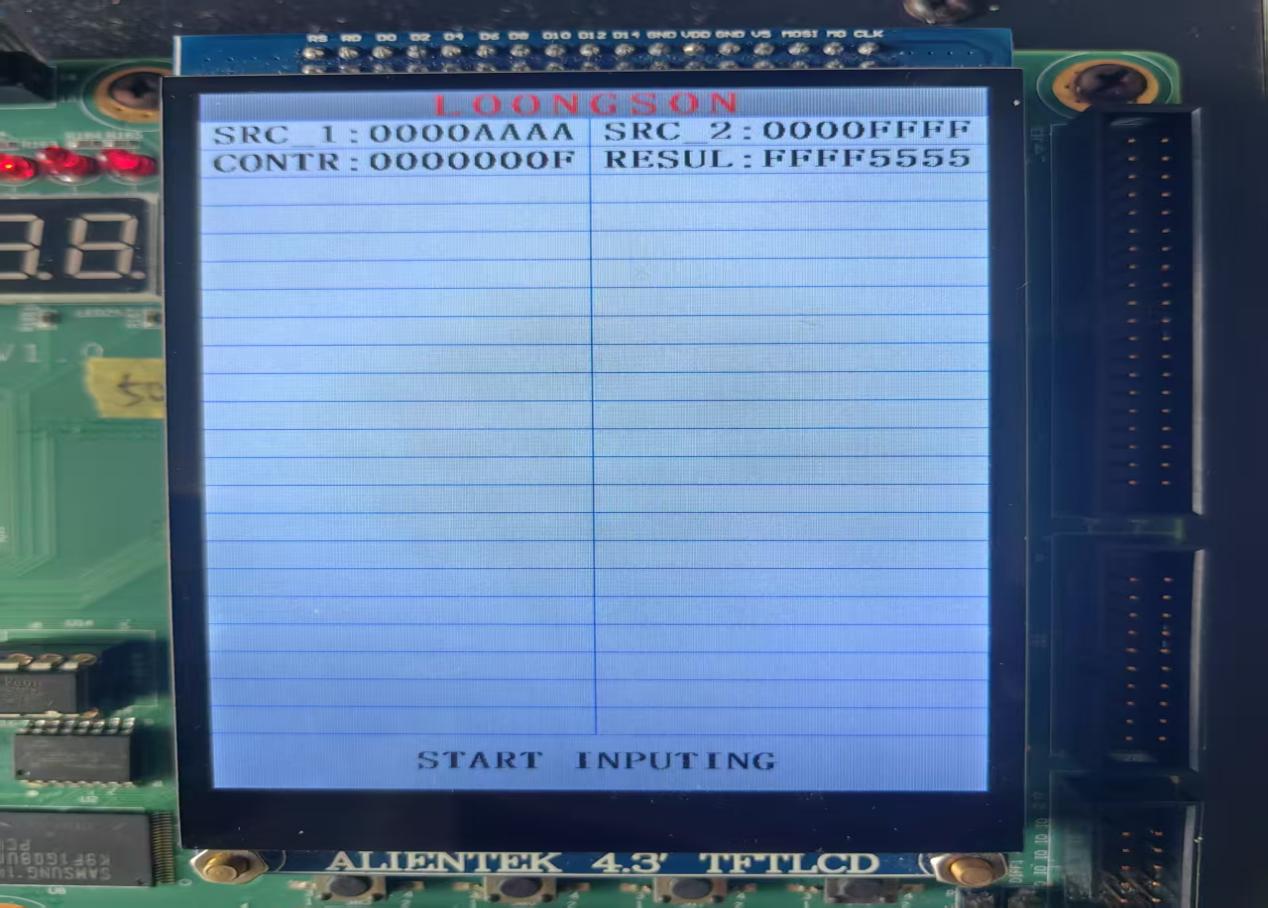
此时的操作数1和操作数2都是ffffffff，由于是无符号数，则应该是2^32-1，很显然操作数1等于2，则结果为0，没有置位，答案正确。



此时的操作数1是ffffffff，操作数2是12345678，由于二者是无符号数，很显然操作数1大于2，则结果为1，置位，答案正确。

### 按位与非运算

此时的控制位是1111，即f。



如图，操作数1是0000aaaa，操作数2是0000ffff，首先二者与运算之后，结果应该是0000aaaa，然后进行非操作，得到的结果应该是ffff5555，与结果一致，答案正确。

## 总结感想

本次实验是基于实验指导手册中的 ALU 功能模块的扩展与优化实践，基础代码来源于老师提供的代码，我需要通过对原有 12 位操作码的压缩处理，将控制信号有效位宽优化为 4 位，在修改的过程里我对于控制逻辑的认识更深入了，学到了新的控制策略。

同时，我需要添加了三种新的运算功能，包括：有符号数大于比较置位，无符号数大于比较置位和按位与非运算。在修改代码的过程中，我进一步巩固了对 ALU 模块结构的理解，特别是操作码控制的使用。

通过上箱测试这一过程，我加深了对整个流程的认知。尽管没有进行仿真验证，但实际的板上结果说明了代码逻辑的正确性。

我熟悉了 MIPS 指令集中的运算指令，学会对这些指令进行归纳分类。了解了 MIPS 指令结构。熟悉并掌握了 ALU 的原理、功能和设计，并进一步加强运用 verilog 语言进行电路设计的能力。

最后，为后续设计 cpu 的实验打下更好的基础。