****

**计算机组成原理**

**课程设计报告**

学 号 ： 20191001171

组员姓名 ： 滕德淋

指导老师： 罗忠文

**中国地质大学地理与信息工程学院**

**2020年 7 月**

目录

目录

[1. 实习概述 2](#_Toc45111051)

[1.1 实习背景 3](#_Toc45111052)

[1.2 实习目标 3](#_Toc45111053)

[1.3 开发环境与工具 3](#_Toc45111054)

[2. 实习过程 2](#_Toc45111055)

[2.1 每日小结 2](#_Toc45111056)

[2.2 解决困难的方法 19](#_Toc45111057)

[3. 系统成果展示 20](#_Toc45111058)

[3.1 Alu 20](#_Toc45111059)

[3.2 Refile 21](#_Toc45111060)

[3.3 CPU 22](#_Toc45111061)

[3.4 测试 23](#_Toc45111062)

[4. 实习体会 23](#_Toc45111063)

# 实习概述

## 实习背景

在完成《计算机组成原理》课程之后进行计算机组成原理课程设计，对所学的知识进行实践操作，并对riscv的cpu架构有所了解并设计。本实习需要完成一个以risc-v为架构的CPU模型并且实现二阶段的流水线结构，最后通过提供的测试用例。编写自己的测试用例并完成测试。

## 实习目标

1. 完成alu；
2. 完成refile；
3. 完成CPU的addi指令测试，要求对需要的数据通道进行连接和需要的控制信号进行设置；
4. 增加数据通道，满足要求的32条指令的实现；
5. 设计控制通道，写出真值表进行设计；
6. 完成PC跳转结构，同时加上二阶段流水线结构；
7. 完成整个CPU模型设计，完成测试；

## 1.3 开发环境与工具

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 类别 | 具体需求 | 备注 |
| 1 | 硬件 | Linux虚拟机 | 操作系统 |
| 2 | 软件 | Logisim | 软件 |

# 实习过程

## 每日小结

**第0天**

安装虚拟机和调试好logisim软件，下载risc-v设计压缩包并对整体CPU设计的架构有所了解。浏览腾讯在线文档，了解这次实习的任务。

**第1天**

1. **完成ALU模块**

通过打开框架电路，根据说明文档所要求的15个指令，一步一步的完成整个ALU模块。（除mulh指令不确定对错）

**Add：**运用logisim内置的加法器adder可以实现；

**And：**运用logisim内置的and门电路可以实现；

**Or：**运用logisim内置的or门电路可以实现；

**Xor：**运用logsim内置的xor复合门电路可以实现；

**Srl：**运用内置的移位运算器可以实现；

**Sra：**运用内置的移位运算器可以实现；

**Sll：**运用内置的移位运算器可以实现；

**需要注意的是srl、sra、sll这三个指令需要运用的运算器相同但是属性需要相应的修改。另外，这三个指令需要用到分裂器将32位的移动位选择出低5位当做输入B，再将其他的27位通过or判断是否全为0，再通过一个选择器判断结果该为什么。如果其他27位不全为0，则表示移位大于32位，输出应该为0；反之；**

**Slt：**运用内置的运算器实现A和B的大小比较，比较结构选择0扩展；

**Divu：**运用内置的除法器可以实现；

**Remu：**运用内置的除法器可以实现，但是输出结果不为商，而为余数；

**Mul：**运用内置的乘法器可以实现，将低32位作为结果；**（因为32\*32=64位）**

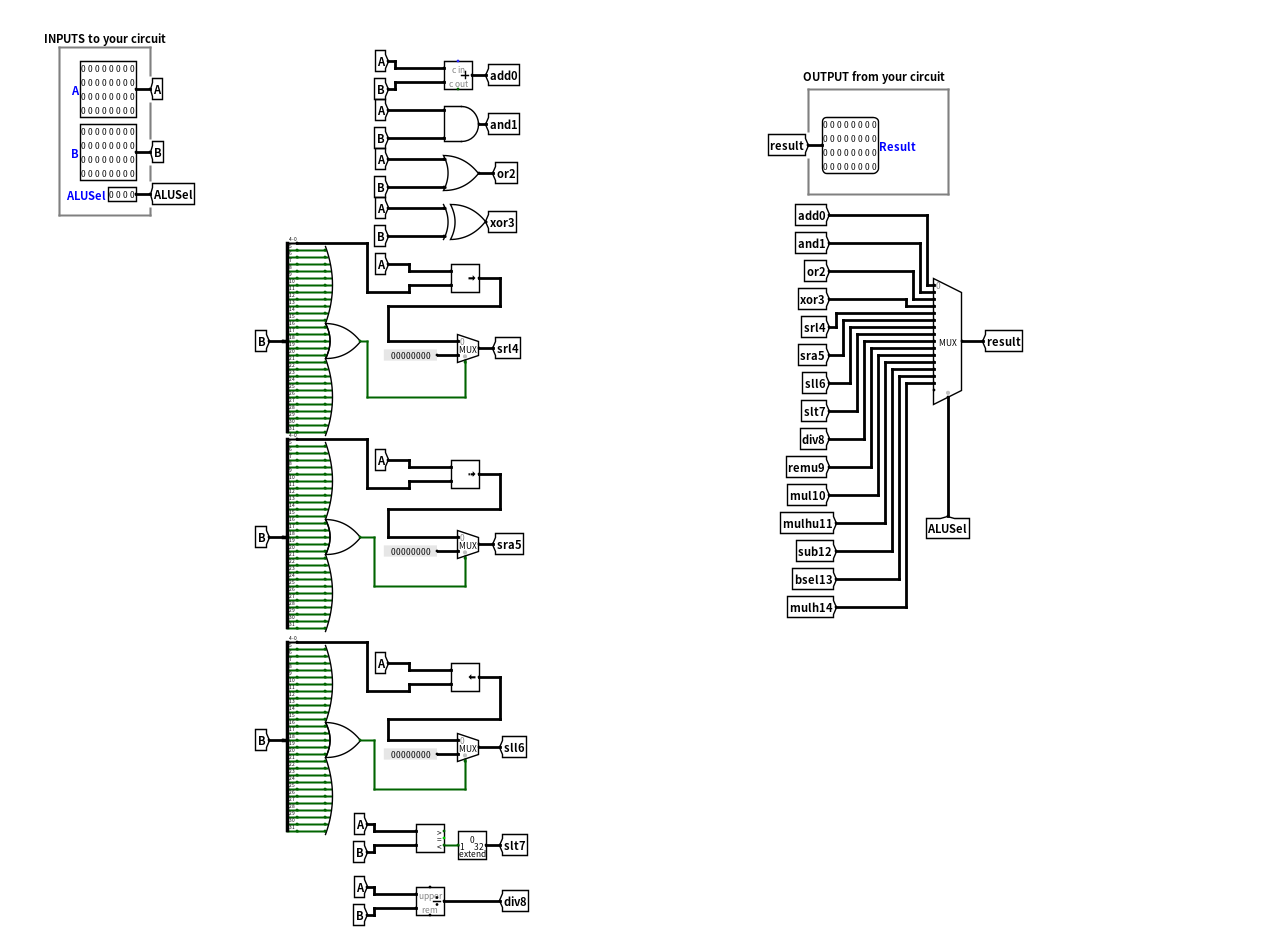
**Mulhu：**运用内置的乘法器实现，不过需要将高32位作为结果；

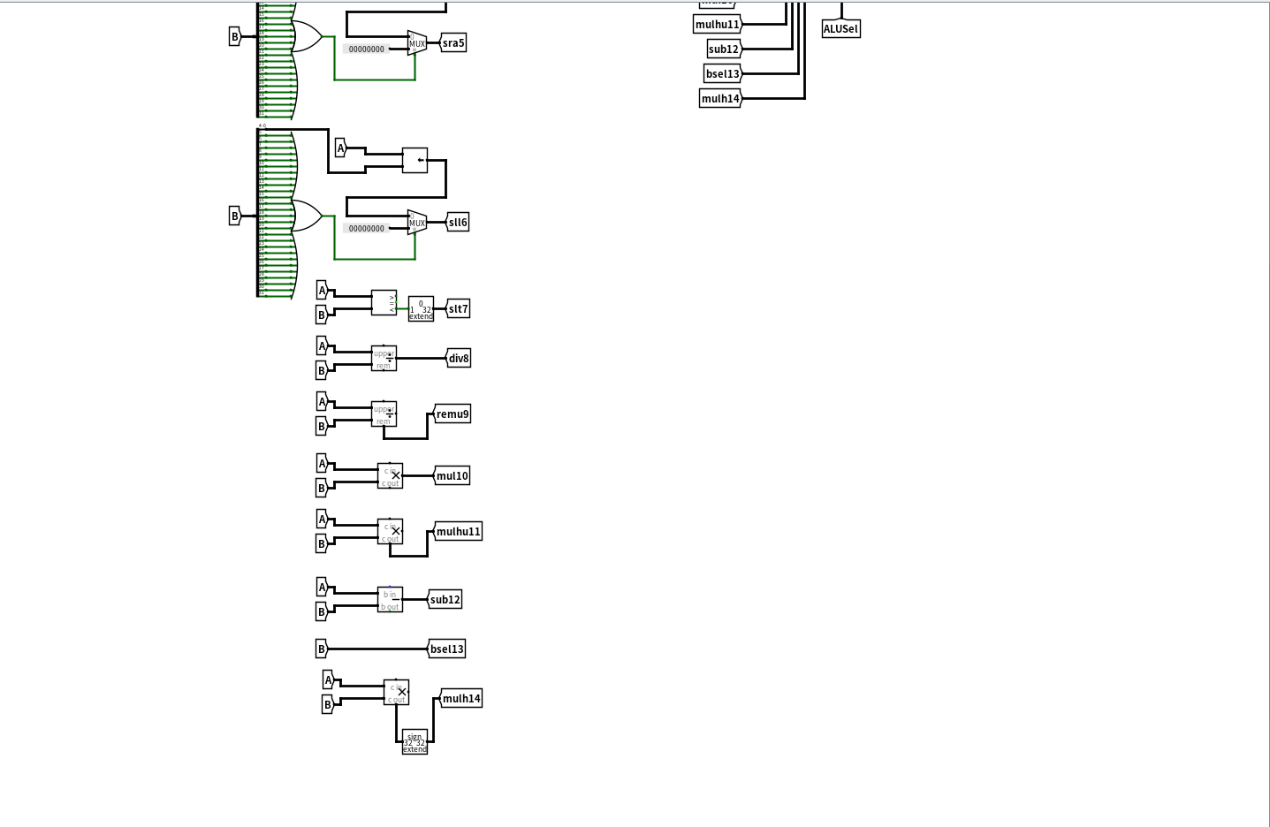
**Sub：**直接运用内置运算器可以实现；

**Bsel：**直接输出结果为B；

**Mulh：**我对这个指令的理解应该是有符号数的乘法运算，所以这就涉及到负数，这就涉及到补码；我是将得到的高32位进行一次符号扩展完成的。

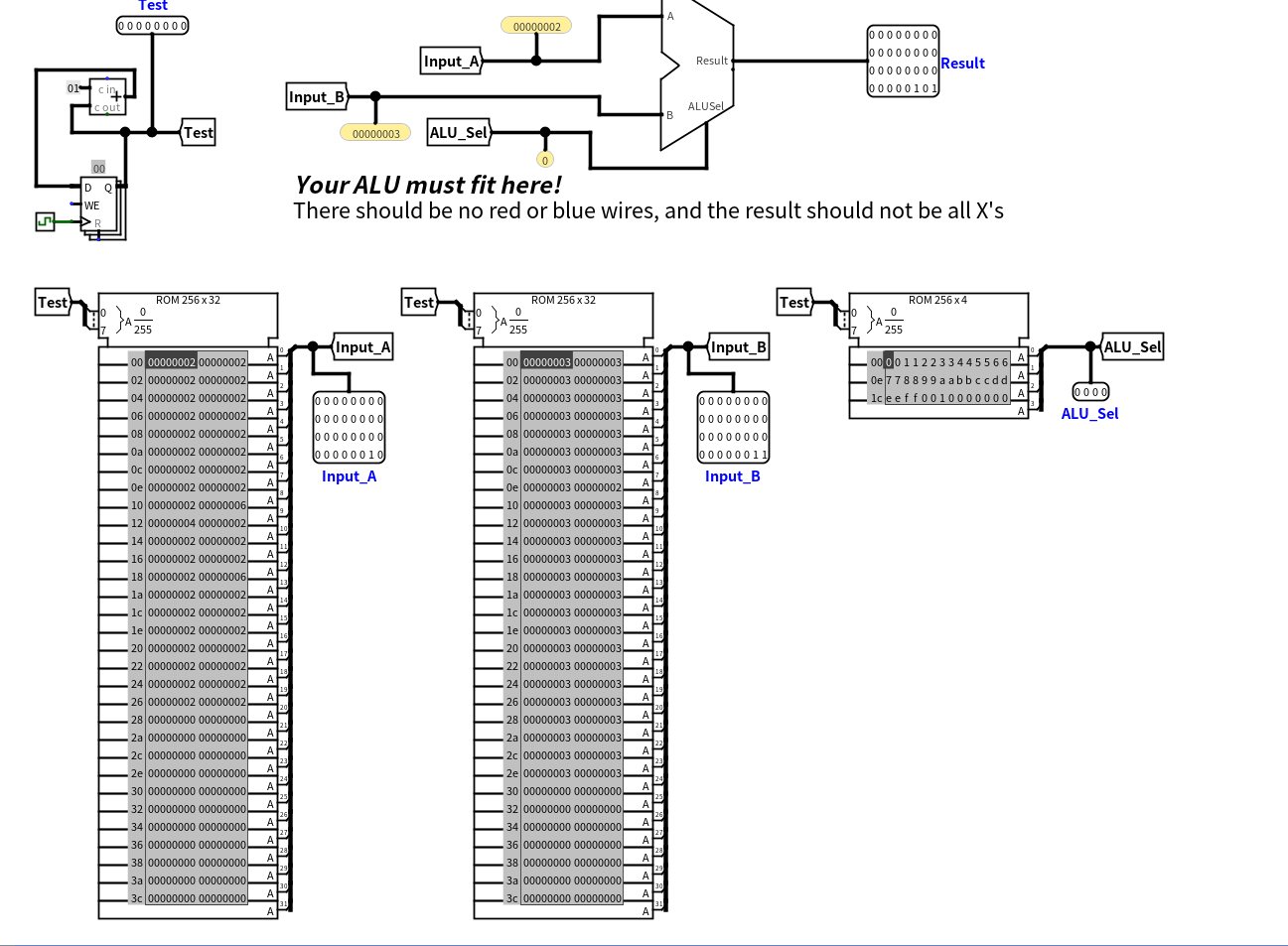
完成了运算的核心部分之后，进行输入和输出的组合，使用一个**4位的多路选择器**；选择哪个指令运算之后的结果作为最后的结果。运用隧道连接会是使得电路看起来更加的简洁。





1. **ALU手动测试**

打开测试电路，将alu文件重新置入该电路中，确保alu计算模块是更新了之后的，然后在ROM输入A中设置合适的值，在ROM输入B中输入合适的值，在ROM输入ALU\_sel中分别输入从0到15,（确保每个指令都得到测试）；之后看输出的结果是否符合我们想要的到的结果；



**第2天**

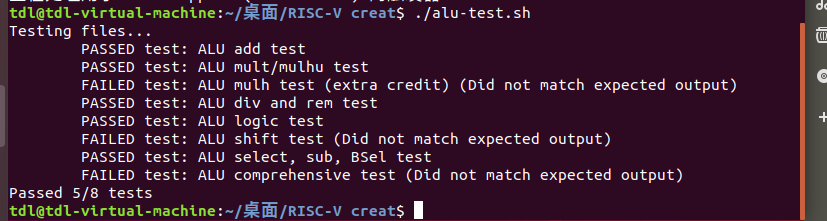
1. **ALU自动测试**

**输入安装Python：$ sudo apt install python**

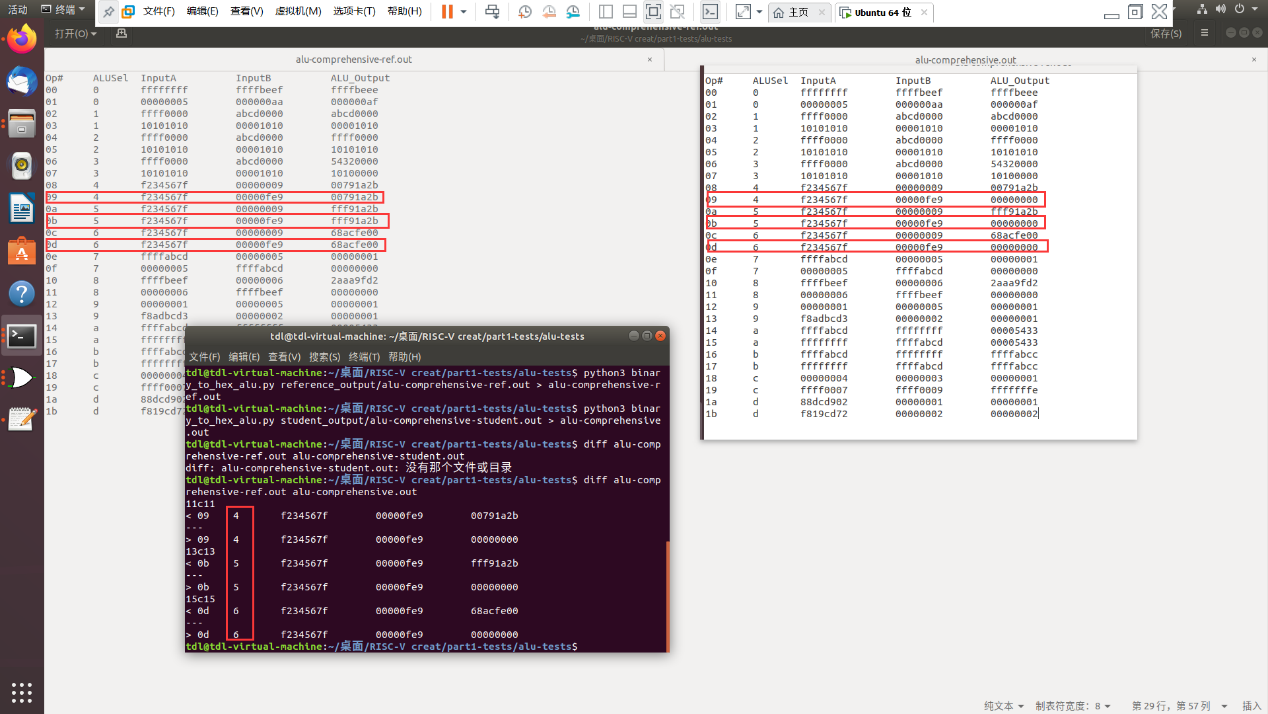
**修改文件权限：$ chmod 777 ./alu-test.sh**

**启动自动测试脚本：$ ./alu-test.sh**

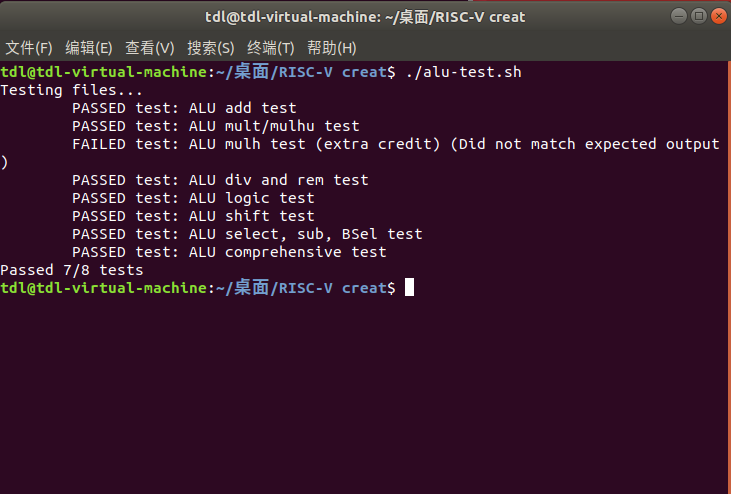
**第一次测试结果：**



**通过将二进制转换为十六进制数，对比正确结果和自己的结果。**



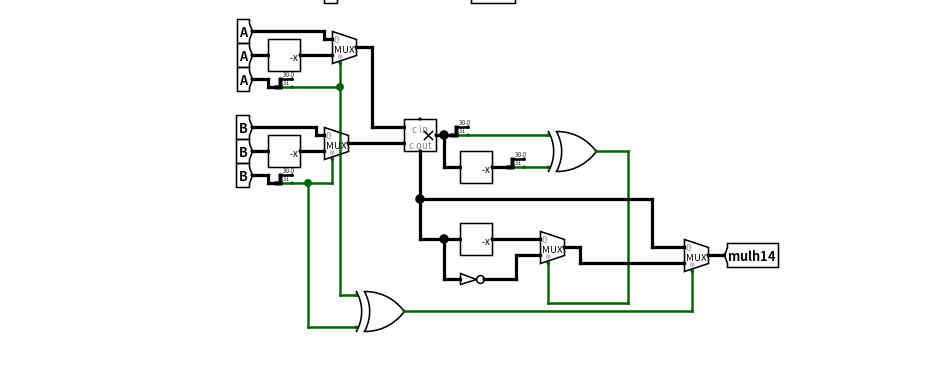
**发现错误出现在shift指令处，并且是由于移位的规则不清楚导致。经过删除判断超过32位的选择器，输出结果与正确结果相符；**



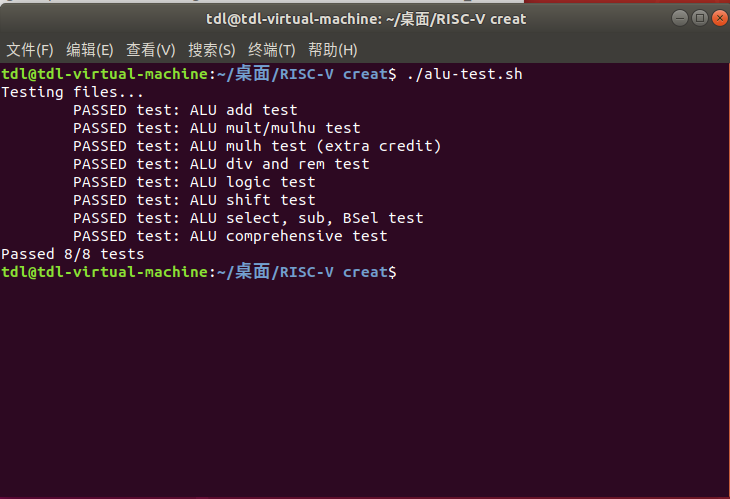
**最终完成测试。**

1. **继续完成mulh运算指令**

**有符号数的乘法，主要是对正数和负数的不同结果。需要运用补码进行运算。具体电路如下：**



**之后再进行自动测试**

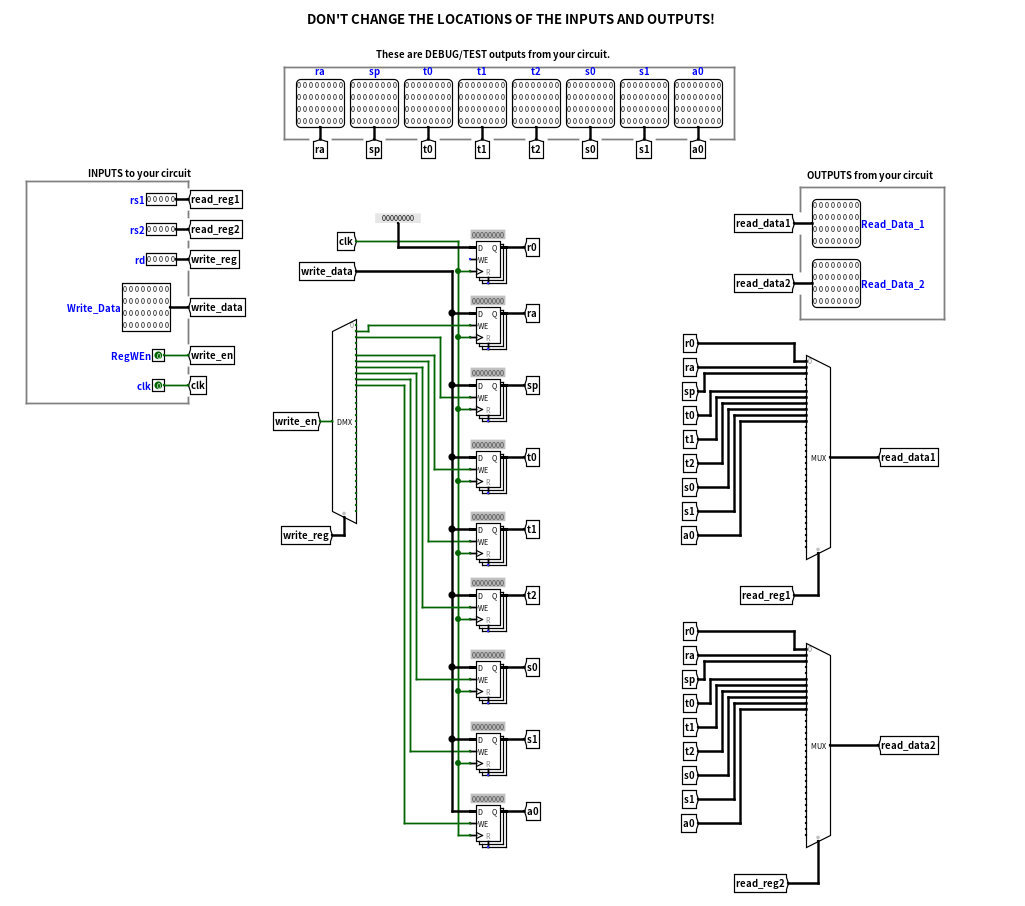


**测试完成。**

**第3天**

1. **完成refile电路**

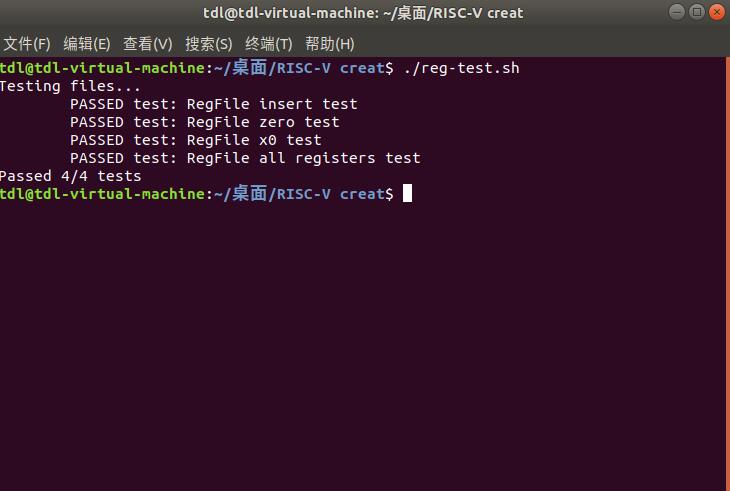
**熟悉要求之后，打开refile.cric电路。首先使用一个解复用器(Dex)并连接好write-en选择位，用于选择输入几号寄存器，然后根据寄存器号连接好9个寄存器，连接好clk和Write\_Data。同时运用两个多路复合选择器选择输出read\_data1和read\_data2。同时将测试用到的隧道连接到输入中。电路图如下**



1. **测试refile**

**首先手动测试refile，通过在****中更新refile之后，在其中的的内存中设置适合的值。观察输出结果是否和我们预想的结果符合。**

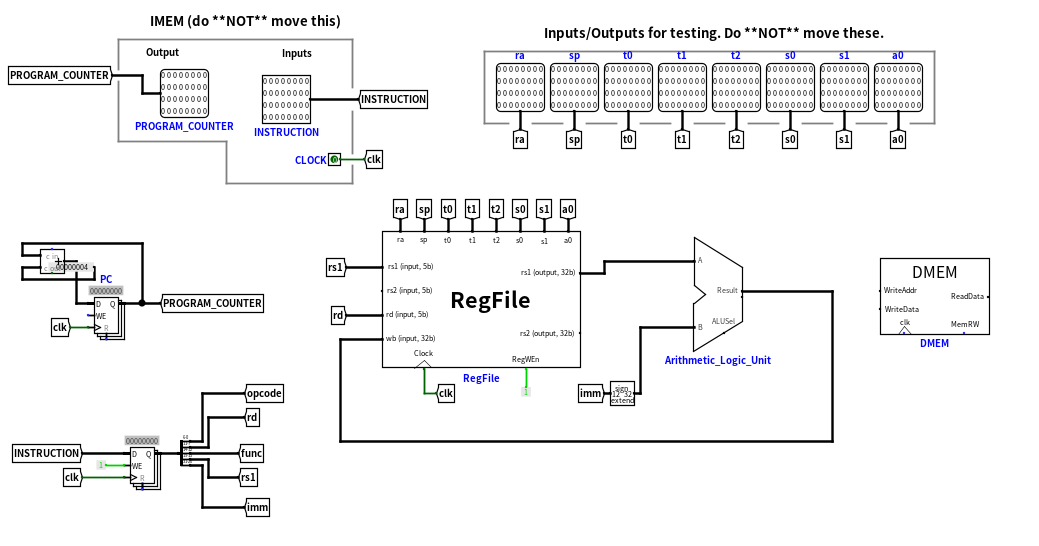
**手动测试没问题之后进行自动测试，在终端输入指令。**



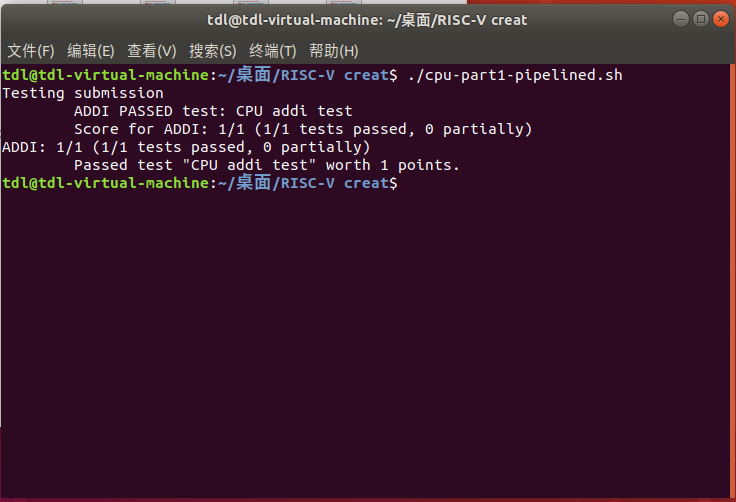
**Refile测试成功。（如果有错可以使用指令调用脚本将输出结果和正确结果从而进行分析错误）**

1. **完成addi指令运行**

**读懂说明文档之后，打开cpu.cric电路和run.cric电路。本练习只要求完成addi指令，所以只需要满足对应所需要的条件即可。首先了解rics-v的指令格式，我们知道了addi的指令格式与mips有所不同，所以在解码的时候就做出改变。其次需要设计一个二阶段的流水线结构，我们可以知道，二阶段流水线就只是延迟了一个指令周期的时间，所以我们在解码的前面添加一个寄存器来增加一个指令阶段，从而达到二阶段流水线的效果。具体电路如下**



1. **测试addi指令运行**

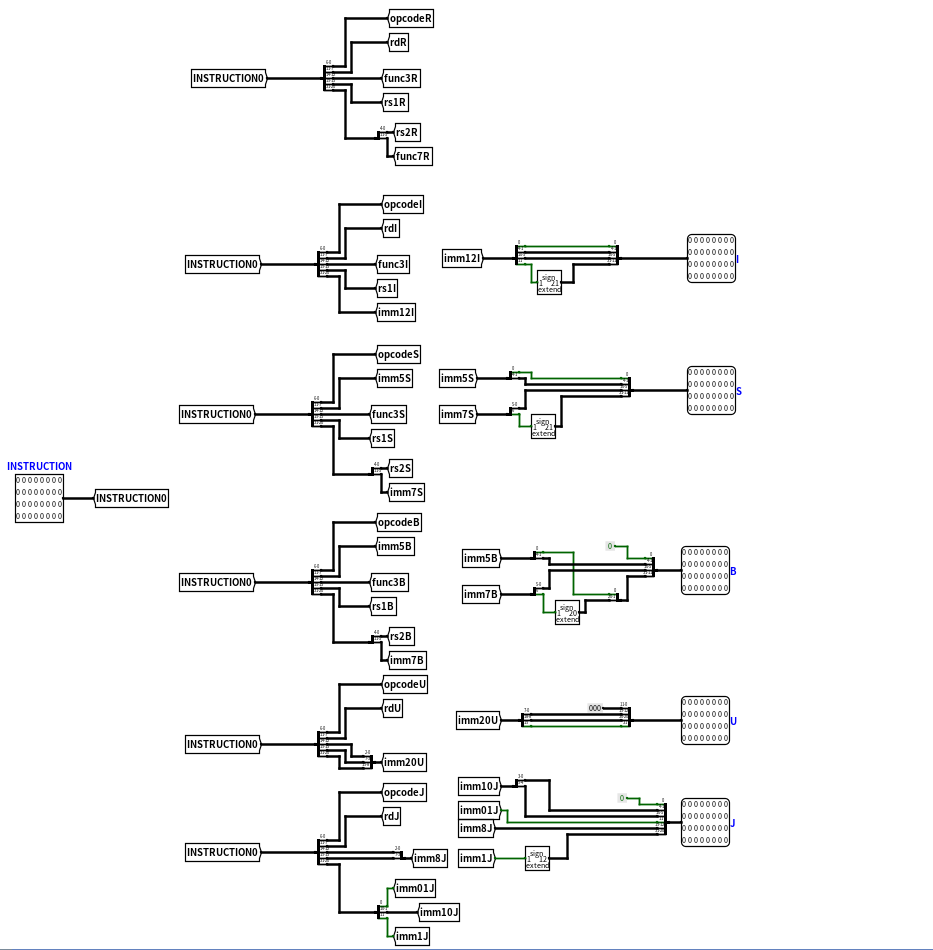


**完成测试。**

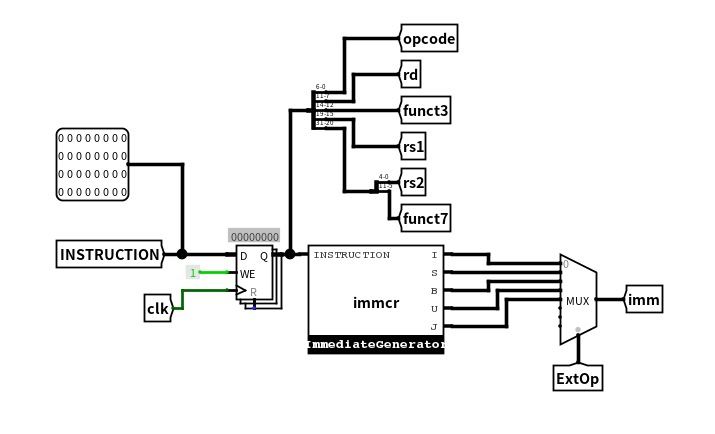
**第4天**

1. **完成解码部分**

**根据几种指令结构和对不同的指令立即数扩展方式进行分析，做一个立即数选择器子电路，电路图如下**

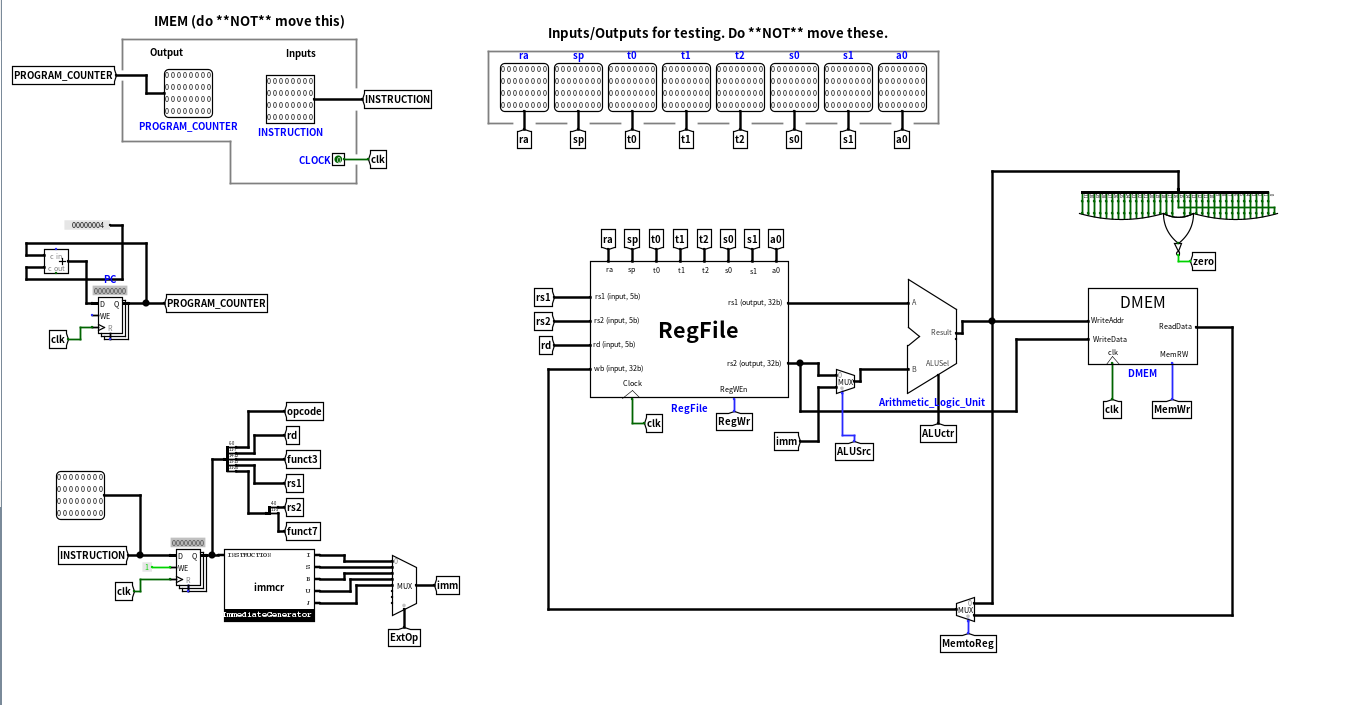


**完成立即数选择器之后进行解码，完成解码部分，电路图如下**



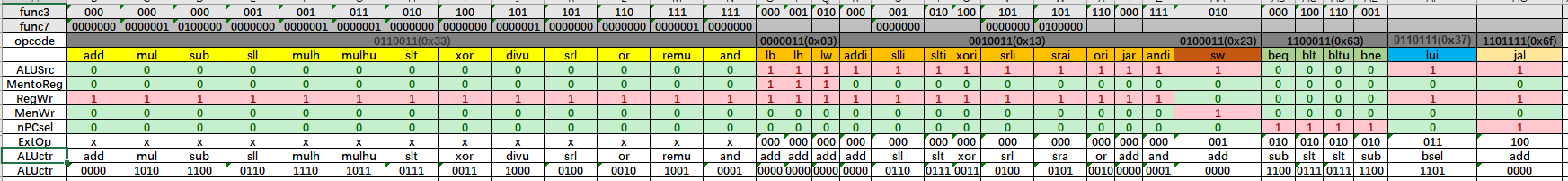
1. **完成数据通道连接**

**按照每一个指令所需要的硬件条件进行设计电路，如下图所示（跳转和分支未实现对应电路）**



1. **准备进行控制通道设计**
2. **写出真值表**

**根据每一个指令进行分析，在控制信号处找到合适的值。**

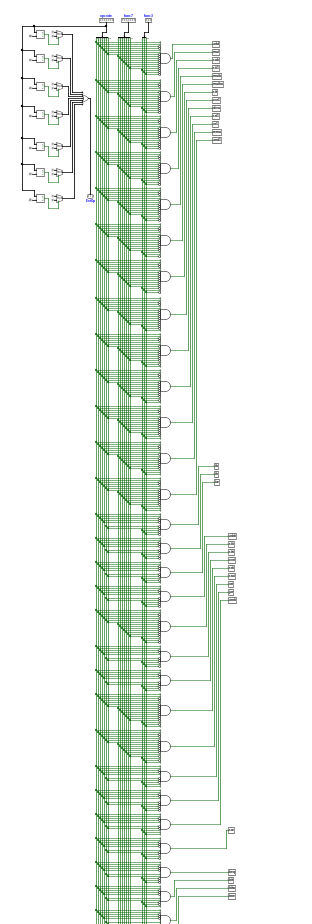


**经过整理得到真值表。**

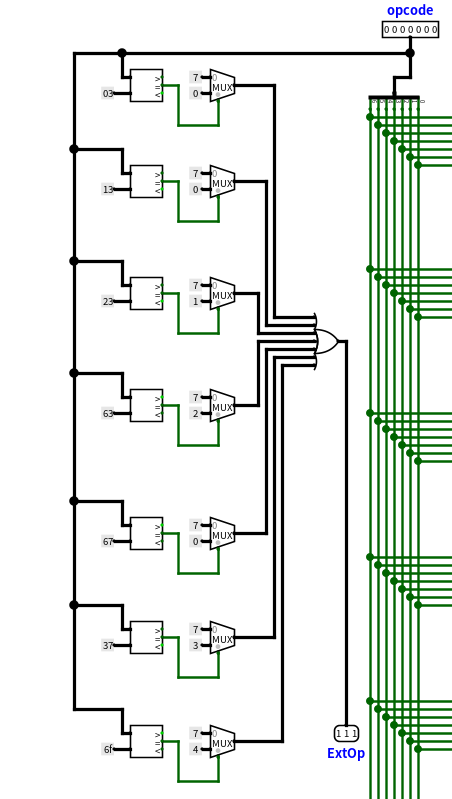
**第5天**

1. **根据真值表进行布尔运算**

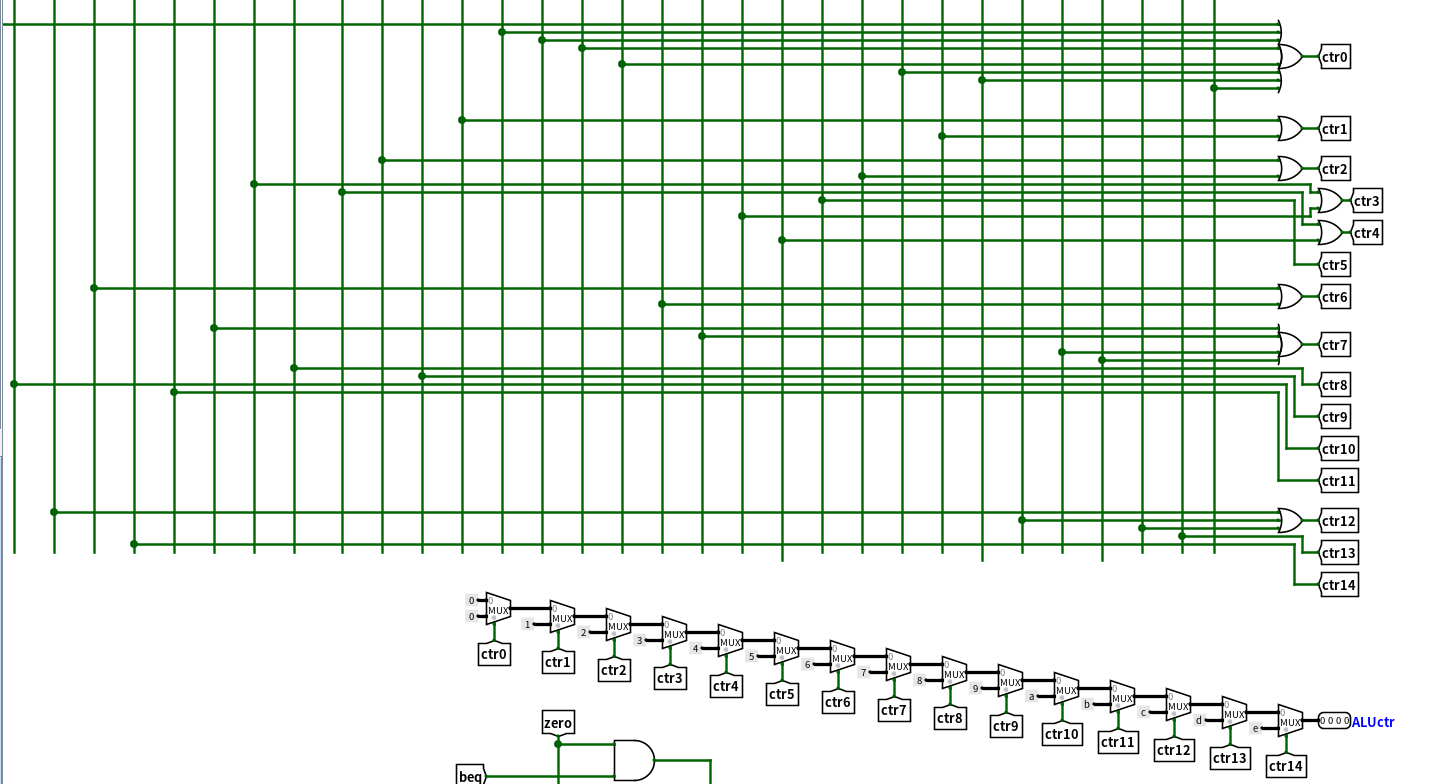
**首先根据opcode和func7和func3字段选择出各种指令信号，用and电路连接，电路图如下（可能有点大）**



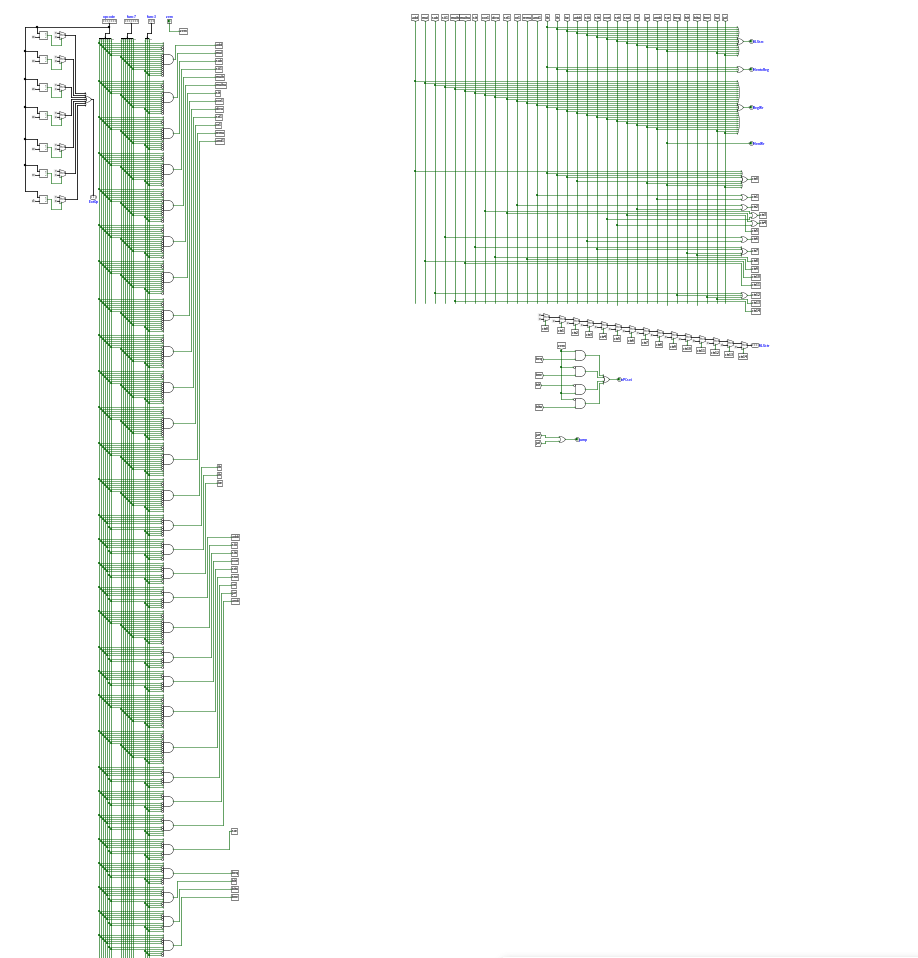
**将32个指令的信号选择出来之后，再通过or门电路将相同运算的指令并在一起，选择出控制信号。其中有两个比较麻烦的控制信号：extop和ALUctr，由于这两个是多位的输出，所以需要单独设计：extop信号由于是指令的立即数扩展方式，这个信号是由opcode就决定了的，所以可以直接运用opcode进行选择，电路图如下：**



**ALUctr信号选择，由于各个运算是按照顺序来的，所以可以运用多个选择器进行常量的输入，最后将选择的结果进行输出，电路图如下：**



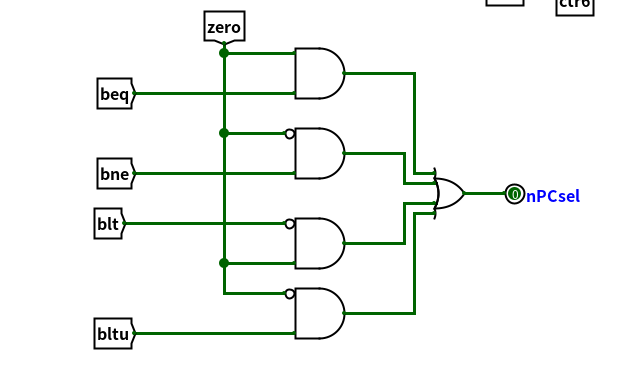
**其他信号就比较好选择出来。最后，总体的控制通道电路图如下：**



**第6天**

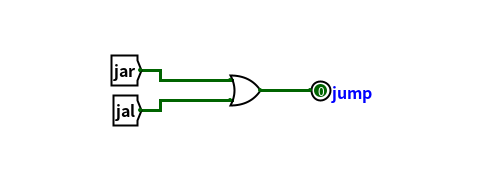
**完成一些特殊的指令的硬件结构（设计PC）**

**Beq、bne、blt、bltu：需要考虑到npcsel信号和zero的值一起决定是否需要跳转，所以需要重新设计一个npcsel信号，电路图如下：**



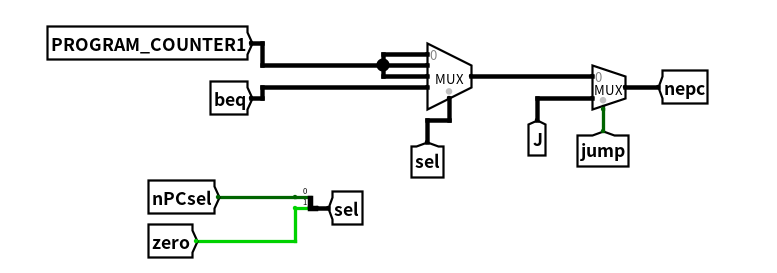
**如果需要跳转，那么就用当前的PC值加上imm段的值；如果不跳转，那么就直接PC+4；所以，需要设计一个PC子电路。电路图如下（暂时还未完成）**

**之后需要增加一个jump信号来实现jal和jarl指令，电路图如下：**

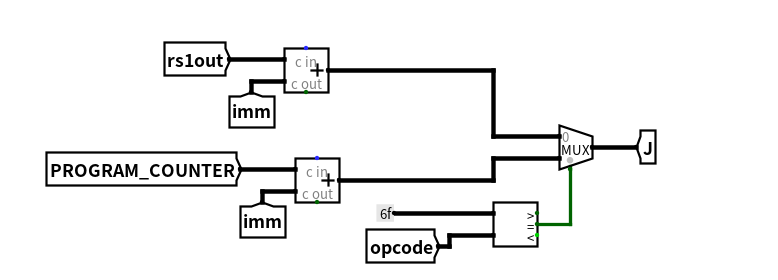


**第7天**

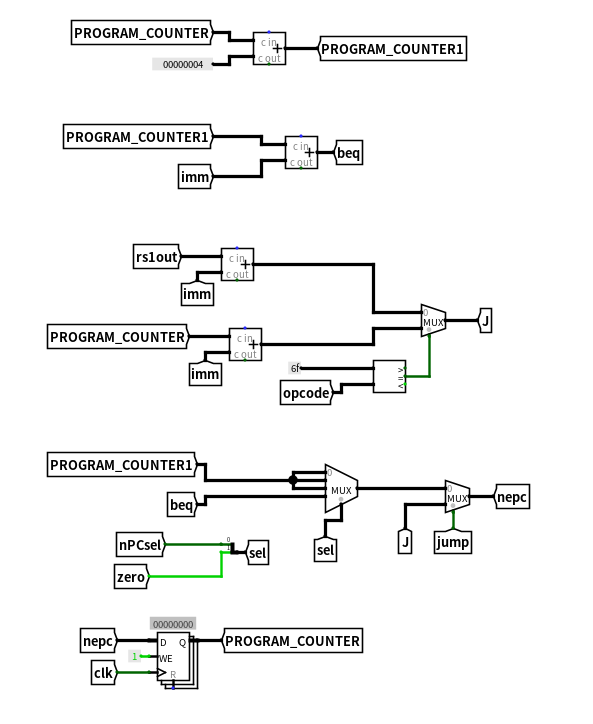
**1、完成PC部分电路，实现跳转，需要注意的是控制信号的选择。修改了昨天设计的npcsel信号。现在设计的选择是否跳转的信号sel由npcsel和zero组成。具体电路图如下：**



**在jump判断的时候需要将J格式的指令的具体指令选择出来，从而实现J格式指令的跳转。电路图如下：**

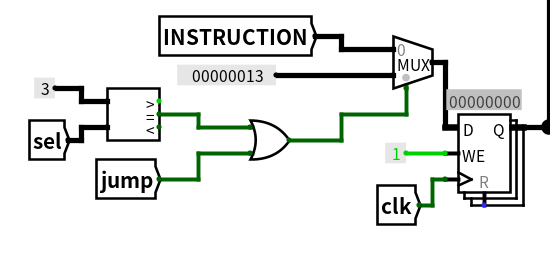


**按照PC应该的逻辑连接好电路，总电路图如下：**



**2、完成增加nop指令的流水线电路**

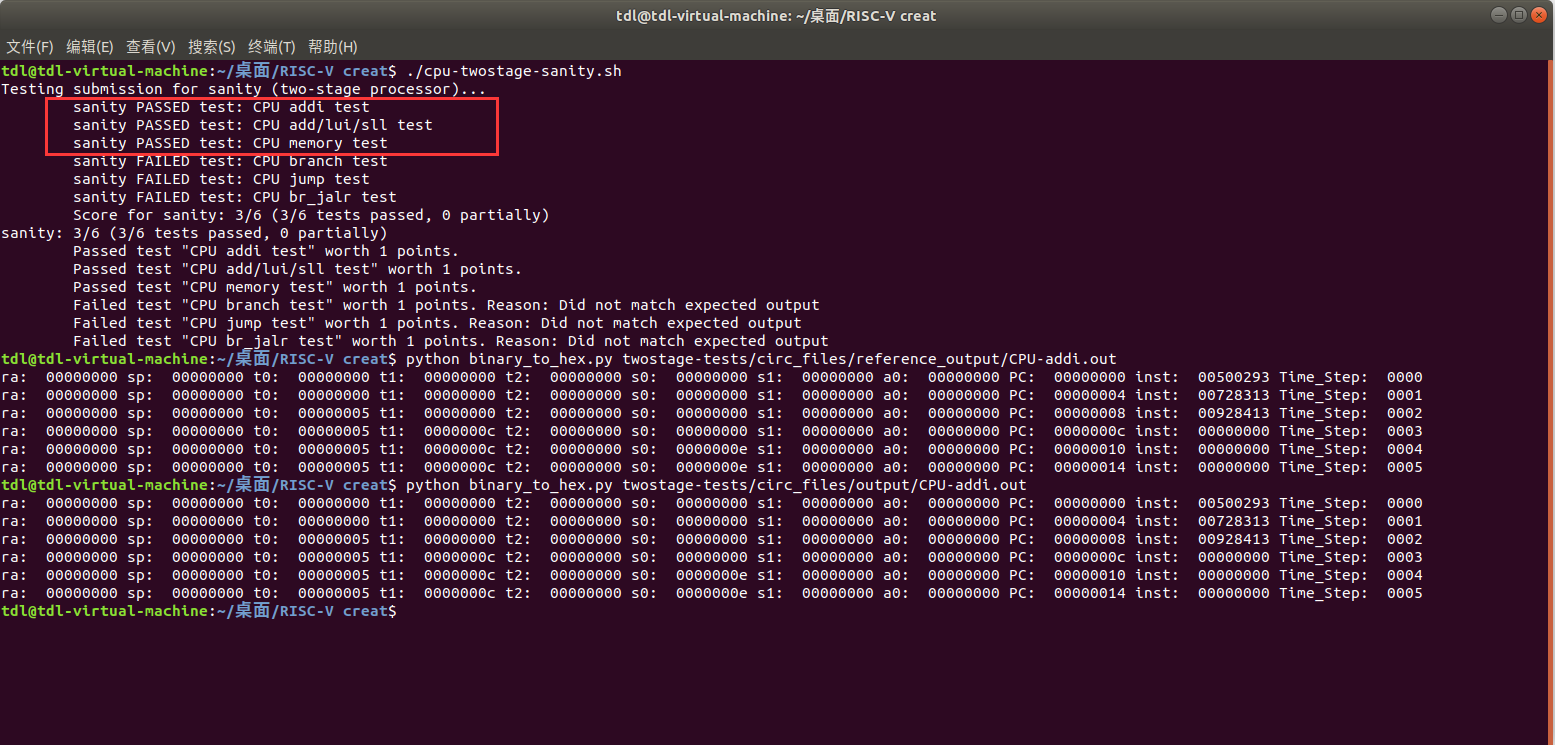
**现在需要的是，当遇到上一条指令是需要跳转的就增加一条nop指令，通过反汇编可知add $0 $0 $0的编码为00000013；所以在将指令输入解码阶段前，需要使用一个选择器。这里的选择信号就必须结合sel信号和jump信号；电路图如下：**



**3、开始测试**

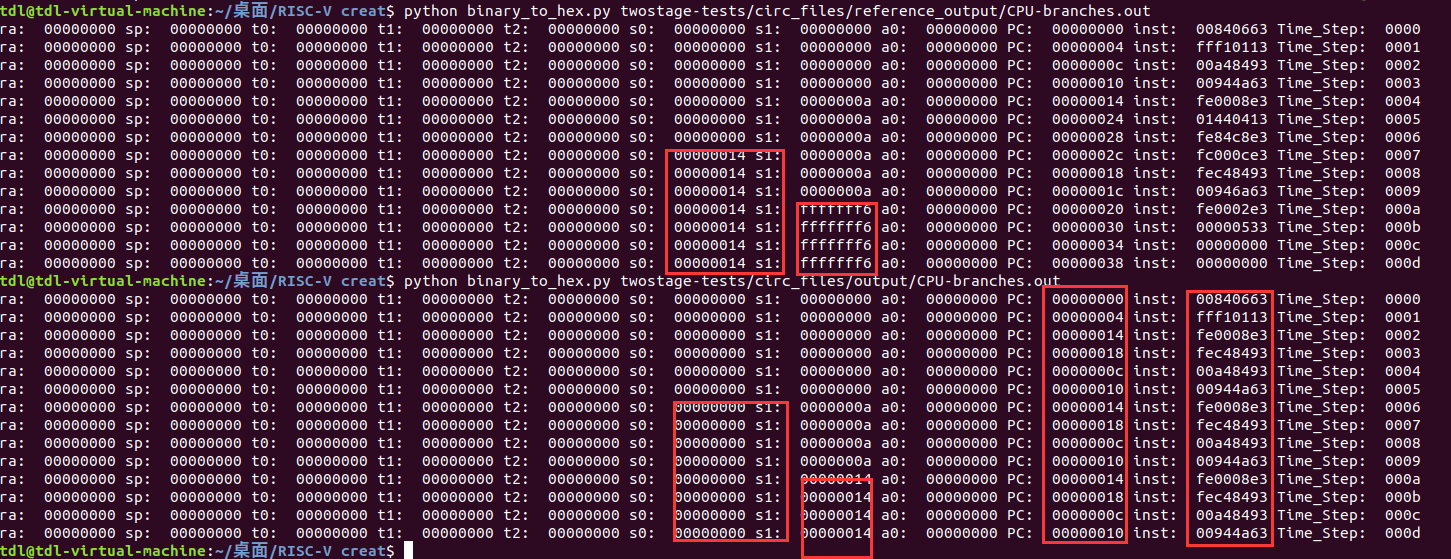
**终端中输入**./cpu-twostage-sanity.sh **测试结果出乎意料，全错；（预料之中）**

**之后开始找问题，首先对addi进行手动调试；先对比正确的输出和我的输出的差别在哪里，经过对比之后发现，我的PC没问题，指令没问题，出问题在寄存器的值不一样。所以可以判断出，问题应该出现在控制信号的选择上面。在控制通道子电路中手动输入opcode的值，去判断所有信号的输出。最后发现问题出现在opcode和func7和func3这几个字段的位取反连接有错误，之后经过长时间修改之后，再去测试。测试结果如下：**



**可以知道对于addi和add/lui/sll和memory类的指令可以通过测试。但是分支和跳转还存在问题。**

**之后再去查看正确值和我的值之间的差别：**



**先分析branch类的问题，可以看出是PC的问题；**

**第8天**

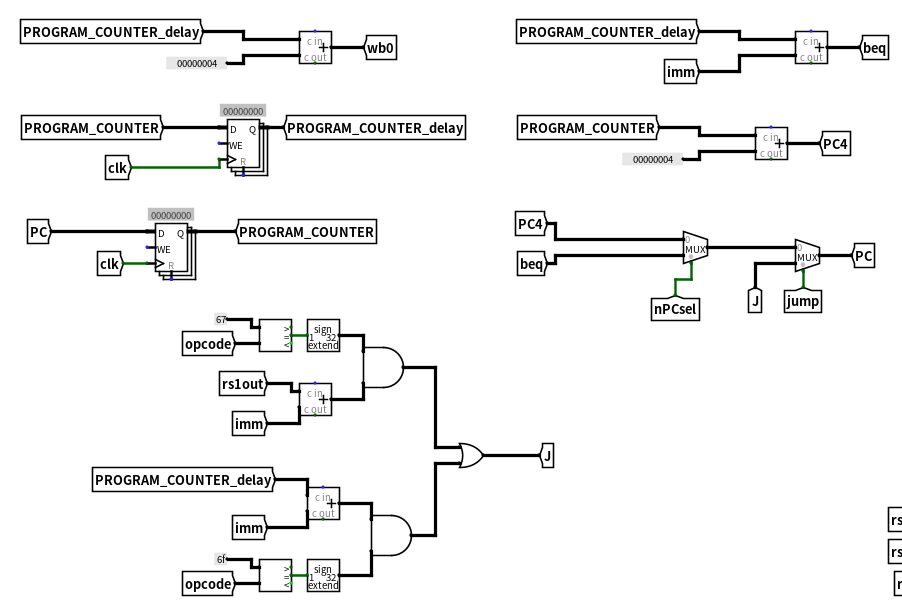
**继续分析为什么通不过，通过仔细分析和对比正确输出和我的输出。尝试着去理解正确输出的含义，从而去修改自己的PC电路；**

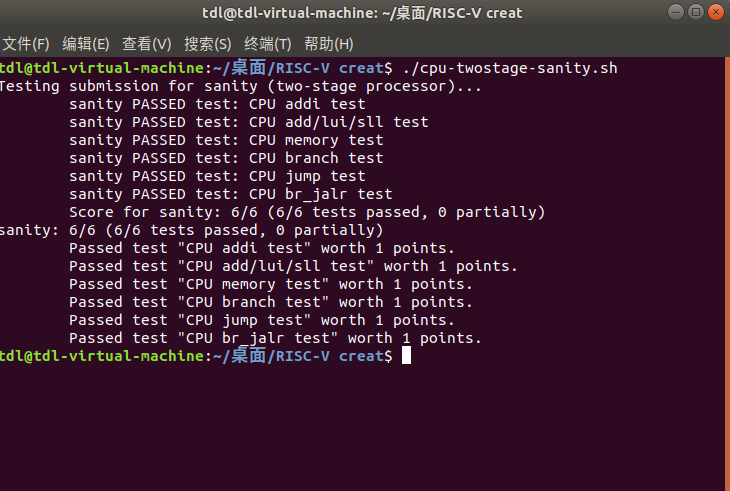
**通过案例指令的解码，可以知道，问题出现在时间延迟上面，PC跳转的值不对，导致指令和计算也不对。最终未修改对；同时开始安装RISCV-GCC，但是有点漫长。**

**星期六：早上起来下载RISCV-GCC，千辛万苦下载好了；**

**星期天：对branch和jump类型的指令进行仔细的分析。最后后发现对于branch的错误在于PC器的跳转地址不正确，由于设计了流水线，对于指令的解码有延迟，对于信号的选择有延迟，同时zero信号也进行了延迟。所以我们的跳转指令的下下条指令才开始跳转，所以需要维持这个延迟周期。最后修改之后，成功通过branch类型的指令。**

**继续观察jump类型的指令，发现有的寄存器的值与标准输出不一样；首先怀疑的是是否有延迟有关，通过仔细分析发现没问题之后。将问题移到是否是jump信号的选择出了问题。通过对控制通道的再一次检查，发现没有问题。又将问题移动到jump跳转的地址是否有问题，最后通过手动测试发现问题出现在选择jump跳转的地址那里。修改之后完成测试。电路图和测试结果如下：**





**CPU健全性测试完成。**

**第9天**

**开始编写自己的测试用例**

1. **单元测试**

**通过说明文档了解到单元测试就是对我们写的32条指令一一对应的进行测试。通过手动编译的方式，保存为.s文件。之后运用**

$ python3 create\_test.py <test 1 name here>.s <test 2 name here>.s ...

**指令将自己的测试用例装进自己的CPU里面input；**

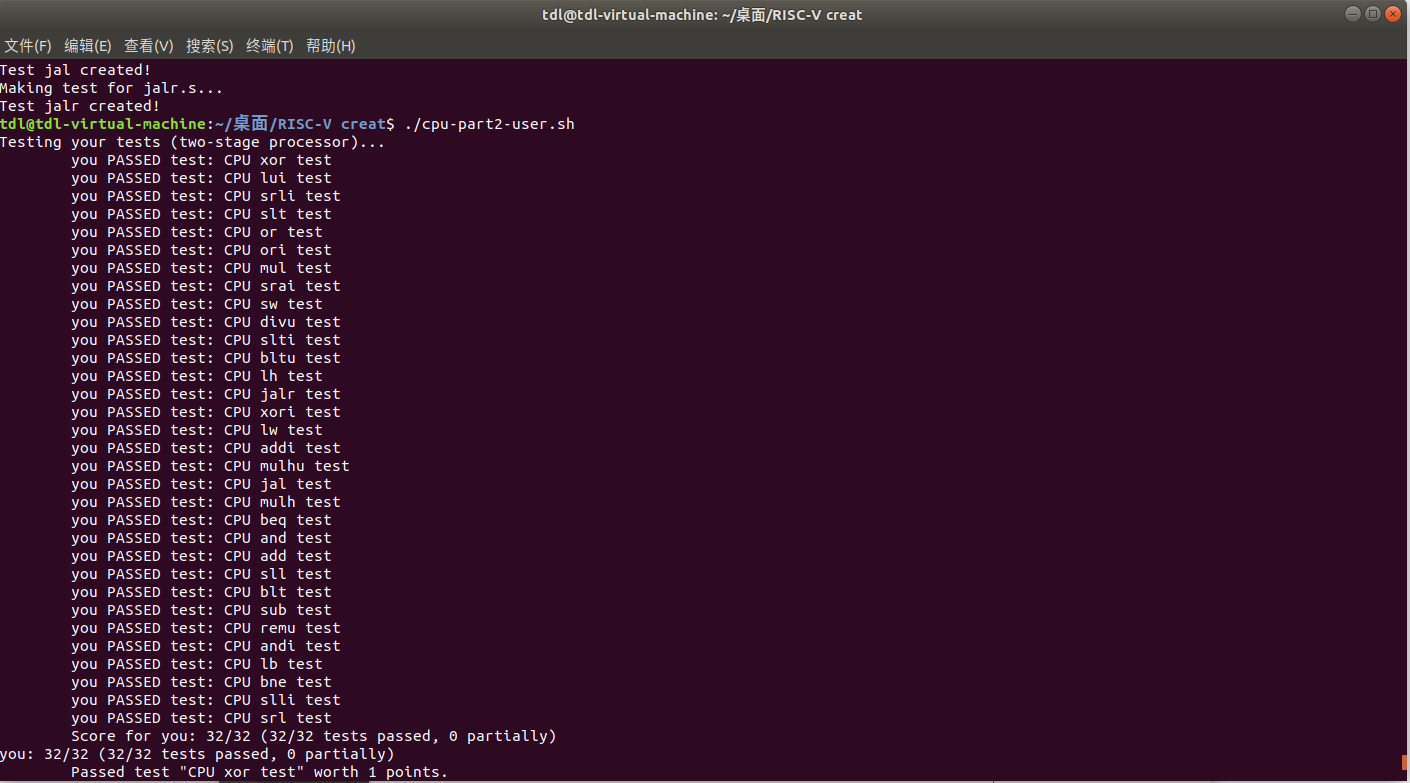
**之后再用**

./cpu-part2-user.sh

**进行测试。**

**其中在写测试用例当中，发现一些问题：1、通过单个测试，可以发现我的原来slt指令运用的比较器之后是进行的符号扩展，正确的做法是进行0扩展；2、发现我的xor指令也不能通过，通过手动调试，发现是我的控制信号连线连错了一根线；3、在分支和跳转的地方参考了提供的健全性测试用例；**

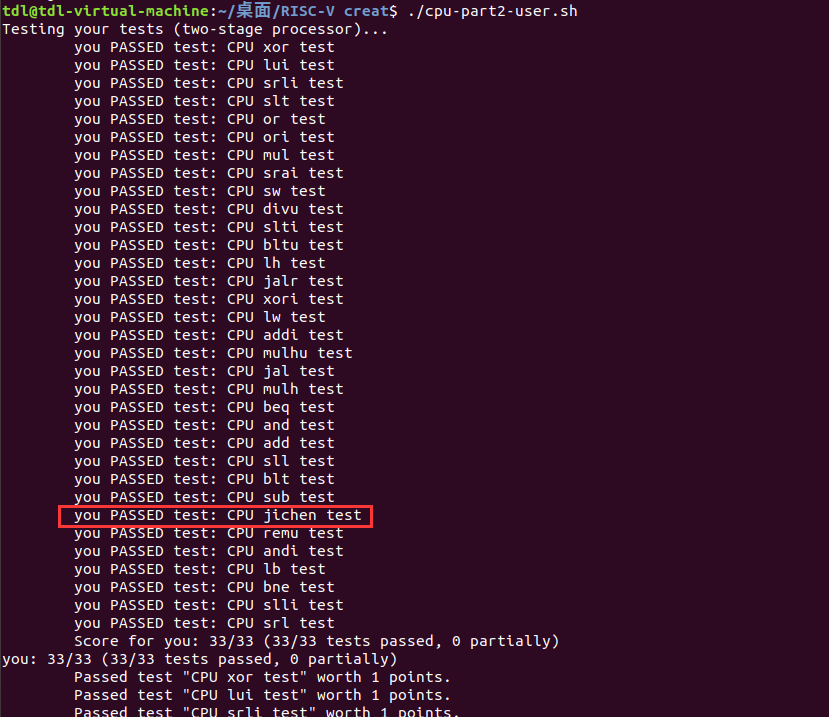
**测试结果如下：**



**第10天**

1. **集成测试**

**在一个.s文件中手动编译一个小程序，用到我们写的32条指令。其中值得注意的地方是在调用函数的时候地址的保存。另外通过分支和跳转指令的测试发现自己的cpu设计还存在延迟不正确的问题。又重新回到电路图中，仔细手动调试，最后发现是bne这一个指令的问题，由于当初这个指令和beq设计的一样，只是取反了，所以通过调试之后改正。最后测试通过，结果如下：**



1. **边界测试**

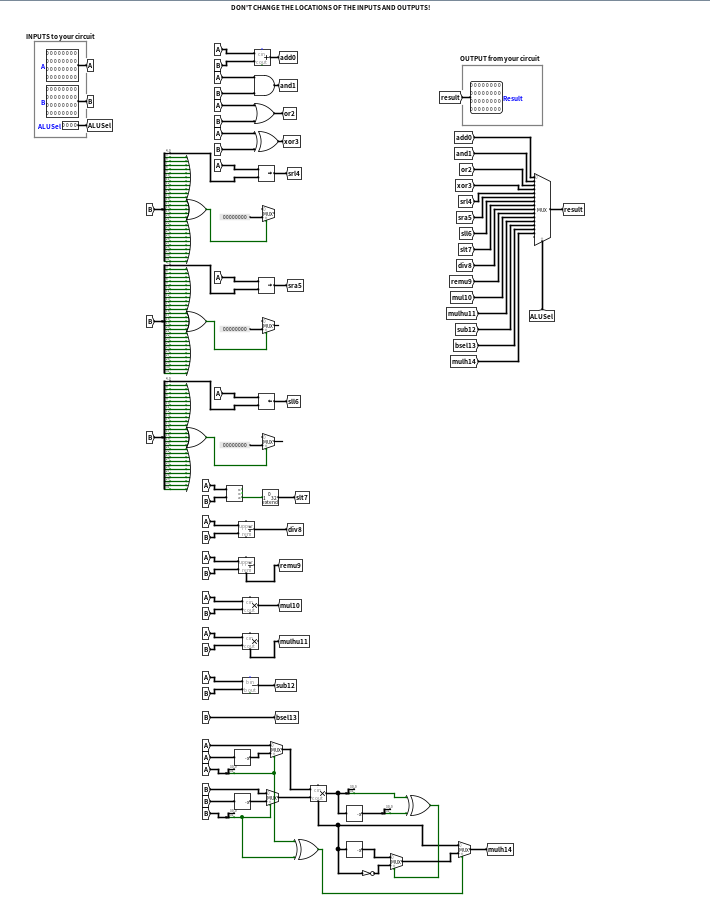
**仔细分析分支和跳转指令，发现跳转的大小不是都可以的，立即数正数最大范围是2047，代码行数在一定范围内（距离跳转指令发出位置）；所以需要注意在使用分支和跳转的时候注意跳转的距离。**

## 解决困难的方法

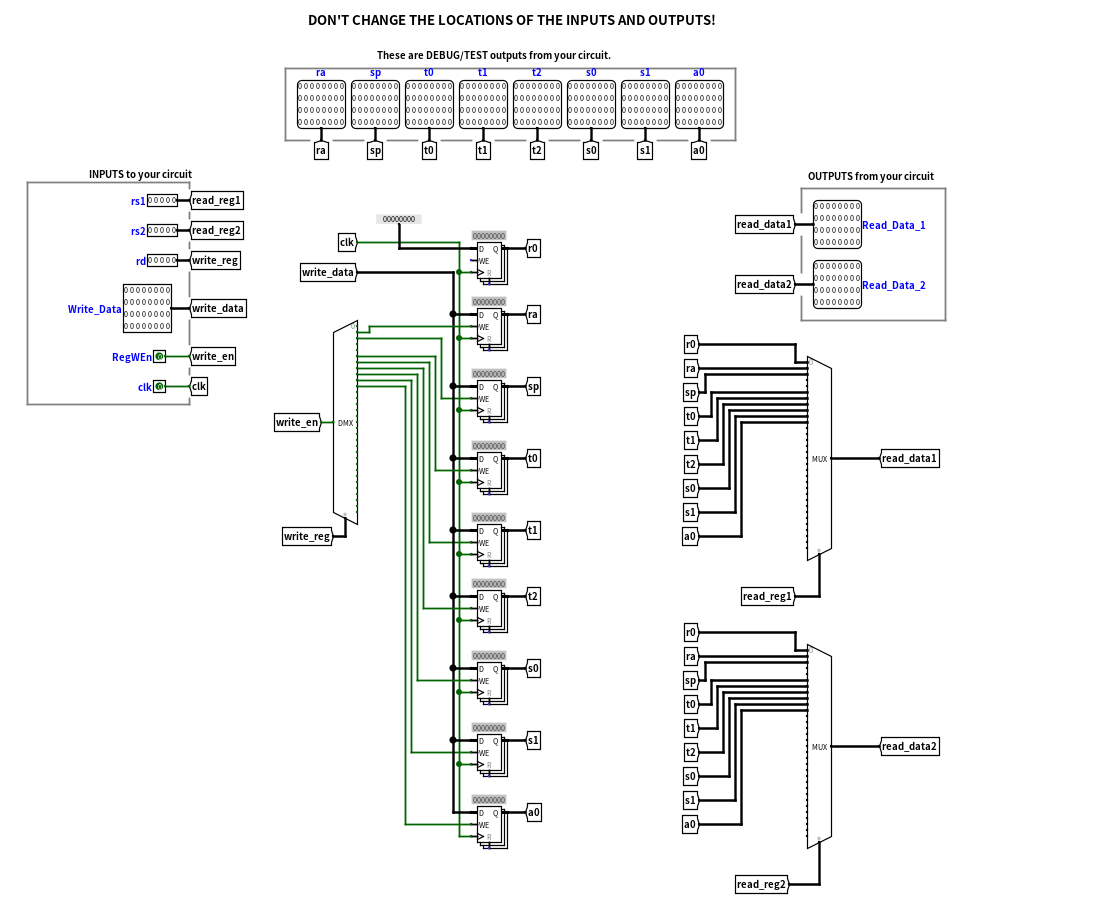
* **与同学交流沟通，一起探讨解决问题的方法；**
* **自行百度；**
* **复习网课内容寻找思路；**

# 系统成果展示

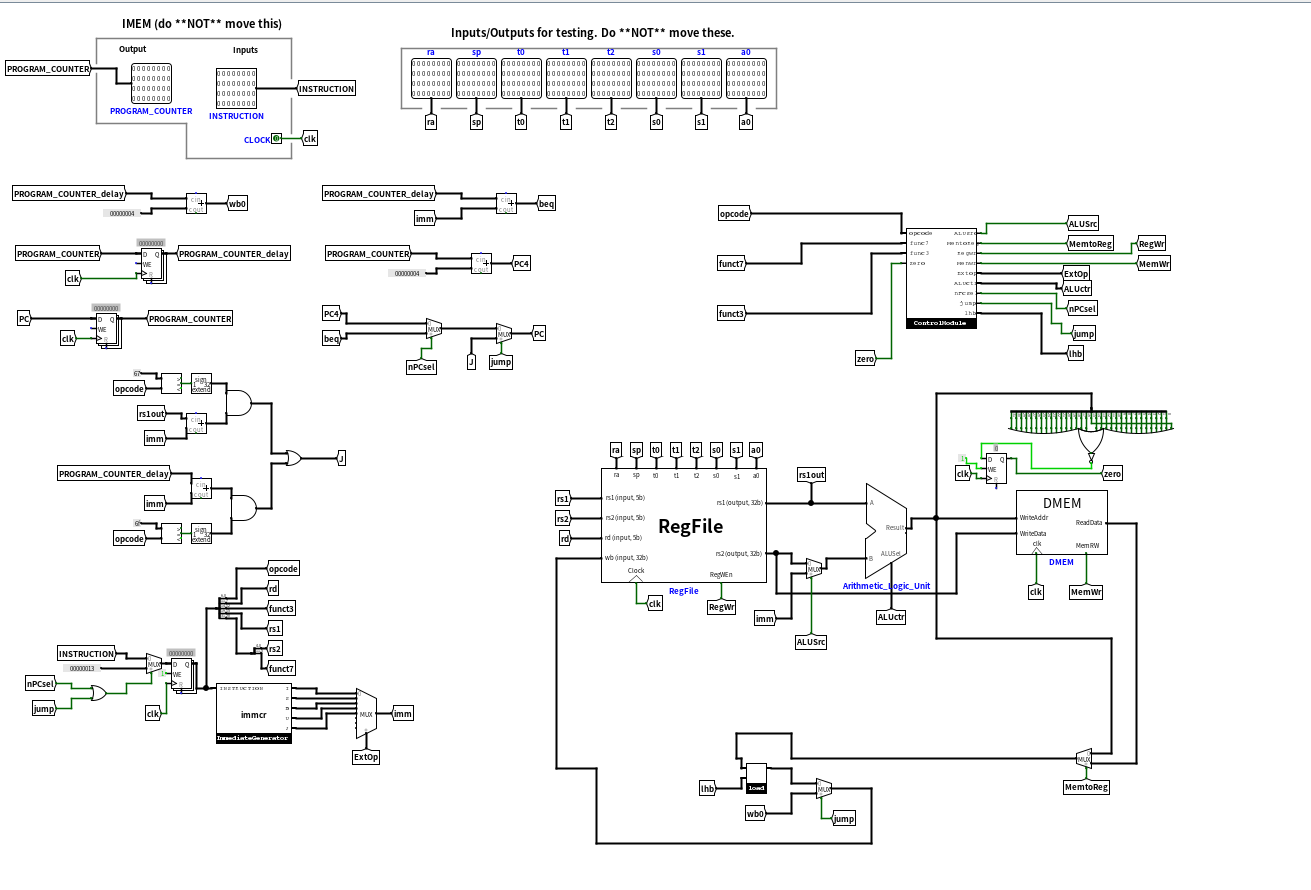
## Alu



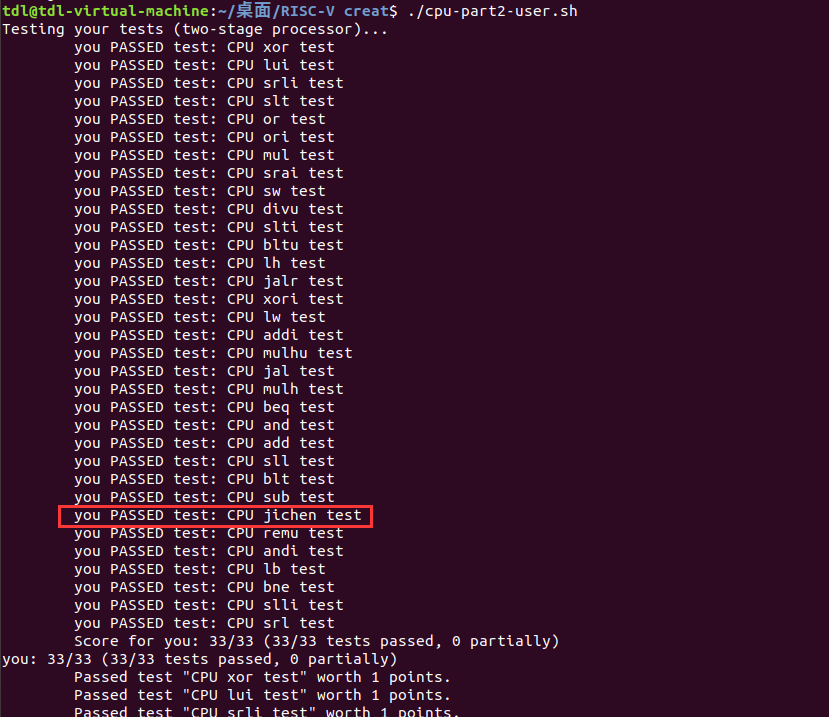
## Refile



## CPU



## 测试



# 实习体会

通过这次实习我了解到risc-v的架构，在实习过程中，我认为按时完成作业对课程进度和理解是有很大帮助的，所以在之后的课程中，不能拖，需要按照老师的进度完成。另外需要学会和同学一起讨论问题。在遇到困难的时候不要一开始就问老师，可以自行百度，增强自身的自学能力。