**计算机组成原理**

**课内大作业报告**

**学 号\_\_\_\_\_\_\_\_\_20074411\_\_\_\_\_\_\_\_\_\_\_**

**姓 名\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**指导教师\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**提交日期\_\_\_\_\_\_\_2022.6.2\_\_\_\_\_\_\_\_\_\_\_\_\_**

**成绩评价表**

|  |  |  |
| --- | --- | --- |
| **报告内容** | **报告结构** | **报告最终成绩** |
| **□丰富正确**  **□基本正确**  **□有一些问题**  **□问题很大** | **□完全符合要求**  **□基本符合要求**  **□有比较多的缺陷**  **□完全不符合要求** |  |
| **报告与Project功能一致性** | **报告图表** | **总体评价** |
| **□完全一致**  **□基本一致**  **□基本不一致** | **□符合规范**  **□基本符合规范**  **□有一些错误**  **□完全不正确** |  |

**教师签字:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **sya**目录

# 总体数据通路结构设计图p3

# 二、数据通路中模块的详细描述p3

# 三、 机器指令描述p12

# 四、编写的测试程序p12

# 五、与测试程序相对应的测试结果截图p13

# 六、设计过程的收获、体会及总结p20

## 一、总体数据通路结构设计图

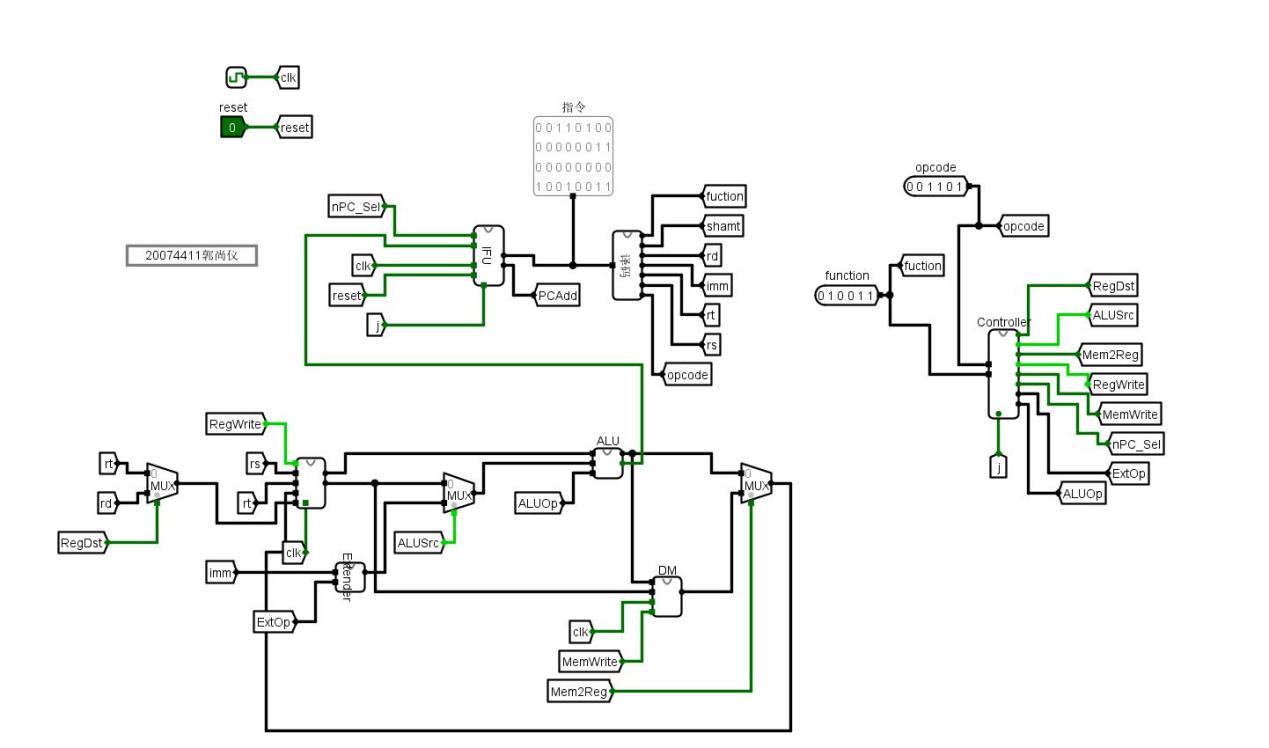


Figure1顶层设计视图

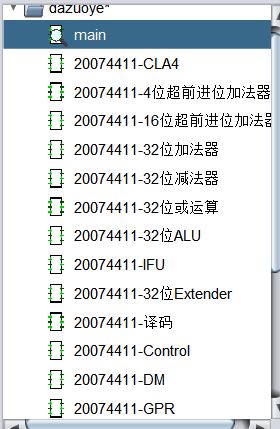


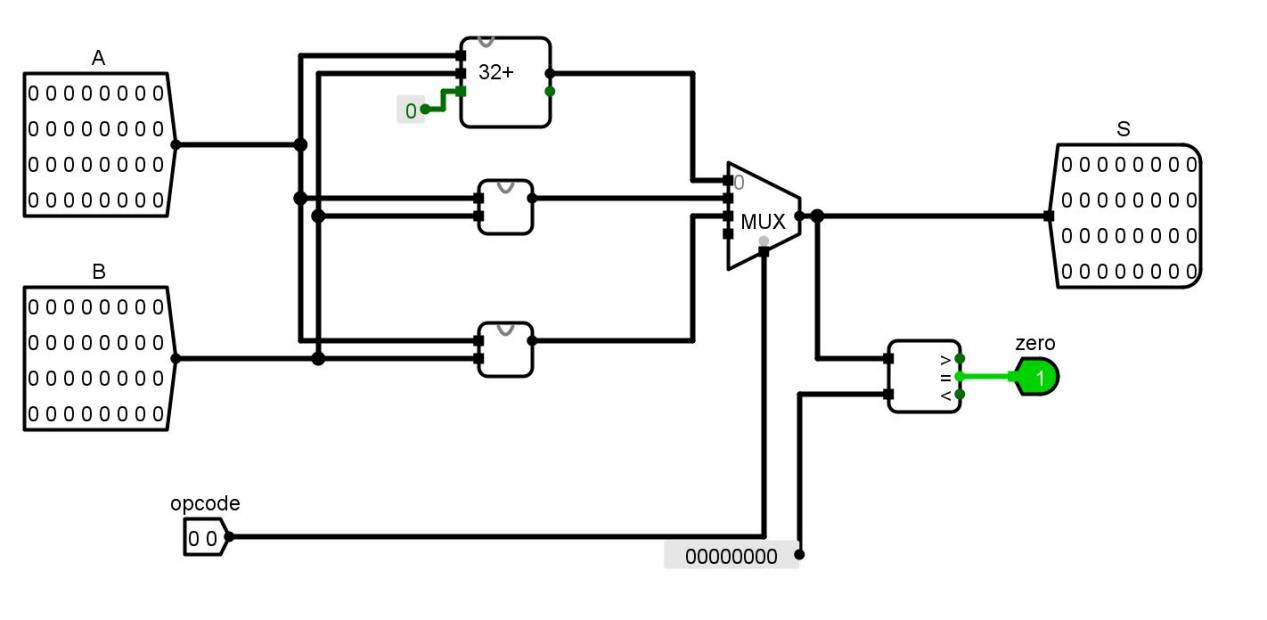
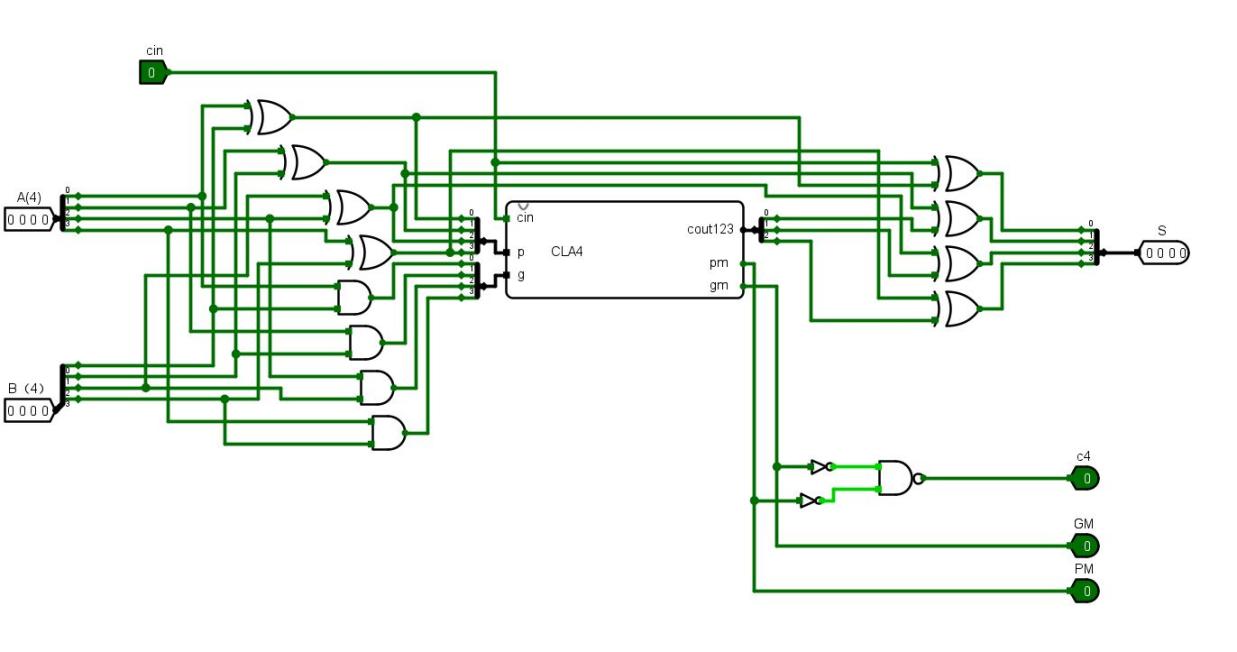
Figure2设计层次

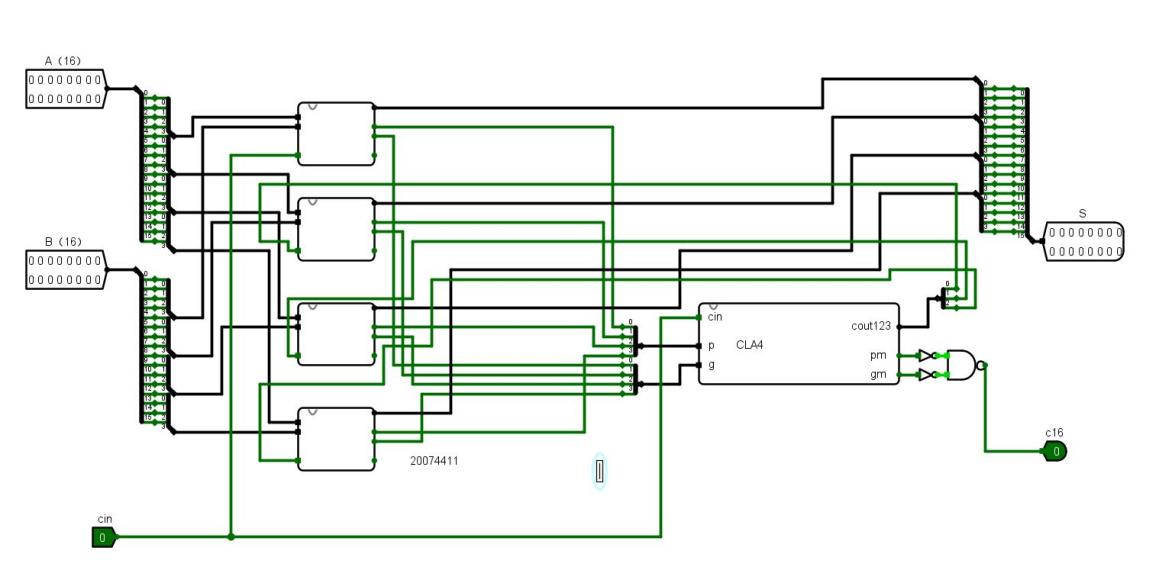
## 二、数据通路中模块的详细描述

ALU模块：

1. 基本描述

ALU主要功能是完成计算功能。根据输入的opcode指令作为控制信号选择使用加法器、减法器、进行或运算，分别对应addu、subu、ori指令。

其中32位加法器由4位超前进位加法器组间组内并行组成16位超前进位加法器再并行组成，并行算出所有进位，因此节约了大量时间



减法器使用了加法器，通过补码减法加法运算转换实现A-B=A+（-B）

或使用了逻辑门按位或

1. 模块接口

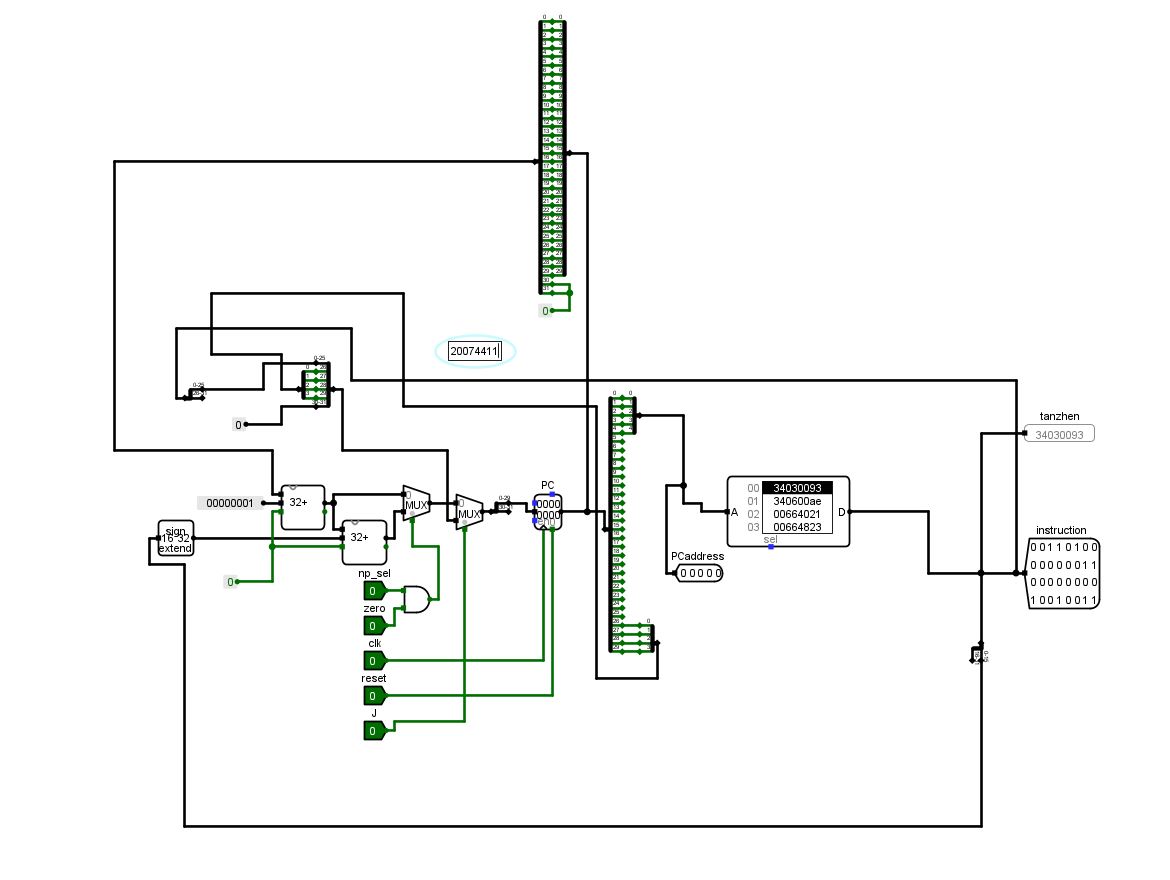
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A | I | 被加/减/或数 |
| B | I | 加/减/或数 |
| opcode | I | 2位选择信号 |
| zero | O | ALU计算结果为0标志。  1：计算结果为0  0：计算结果非0 |
| S[31:0] | O | 32位MIPS指令 |

1. 功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加 | 如果opcode为00，输出A+B |
| 2 | 减 | 如果opcode为01，输出A-B |
| 3 | 或 | 如果opcode为10，输出A|| B |

IFU模块：

（1）基本描述

IFU主要功能是完成取指令功能。IFU内部包括PC、IM(指令存储器)以及其他相关逻辑。IFU除了能执行顺序取指令外，还能根据BEQ指令的执行情况决定顺序取指令还是转移取指令。tanzhen的probe显示当前执行指令的十六进制代码。

（2）模块接口

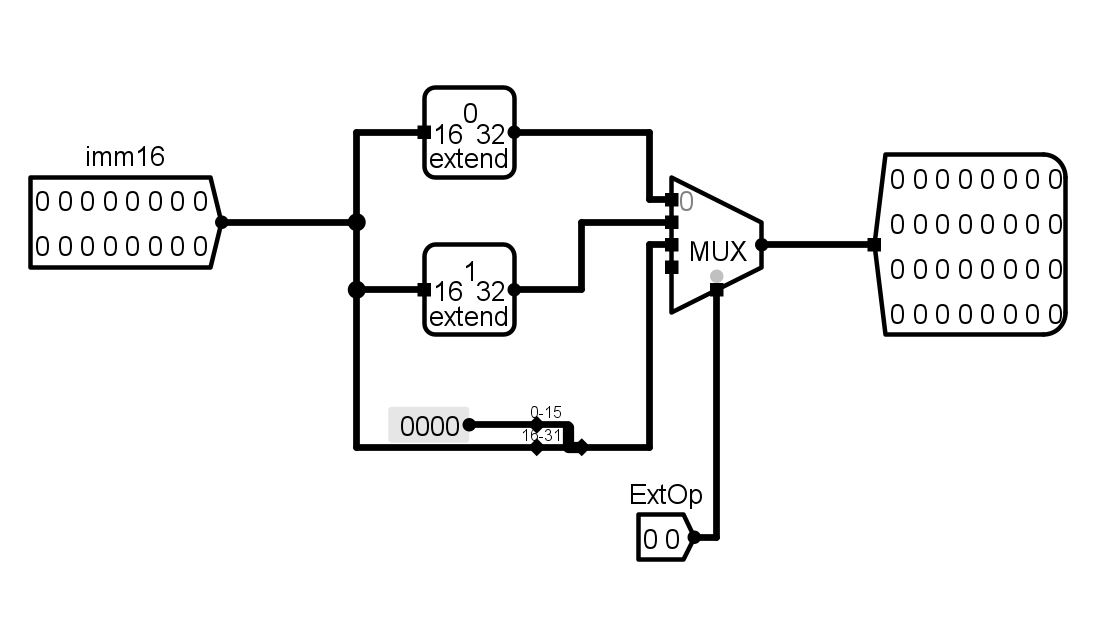
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| npc\_sel | I | 当前指令是否为beq指令标志。  1：当前指令为beq  0：当前指令非beq |
| zero | I | ALU计算结果为0标志。  1：计算结果为0  0：计算结果非0 |
| clk | I | 时钟信号 |
| reset | I | 复位信号。  1：复位  0：无效 |
| j | I | 标记是否为j指令 |
| instruction[31:0] | O | 32位MIPS指令 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当复位信号reset有效时，PC被设置为0x00000000。 |
| 2 | 取指令 | 根据PC从IM中取出指令。 |
| 3 | 计算下一条指令地址 | 如果当前指令不是beq指令，则PC🡨PC+4  如果当前指令是beq指令，并且zero为0，则PC🡨PC+4  如果当前指令是beq指令，并且zero为1，则PC🡨PC+4+(sign\_ext(ins[15:0])<<2)  如果当前指令为j指令，则pc跳转至相应地址 |

EXT模块：

（1）基本描述

EXT符号拓展单元模块通过extender和多路选择器将lw、sw、beq等指令中的立即数字段的二进制补码表示的16位数实现16-bit to 32-bit拓展，包括补0补1和低位补0。

有符号数，将原有的16位数赋到32位新数的低16位，复制其符号位填满新数的高16位，实现符号拓展。

无符号数，将原有的16位数赋到32位新数的低16位，用0填满新数的高16位。

为了实现lui指令的功能，要将原有的16位数赋到32位新数的高16位，再讲低16位用0填充。

（2）模块接口

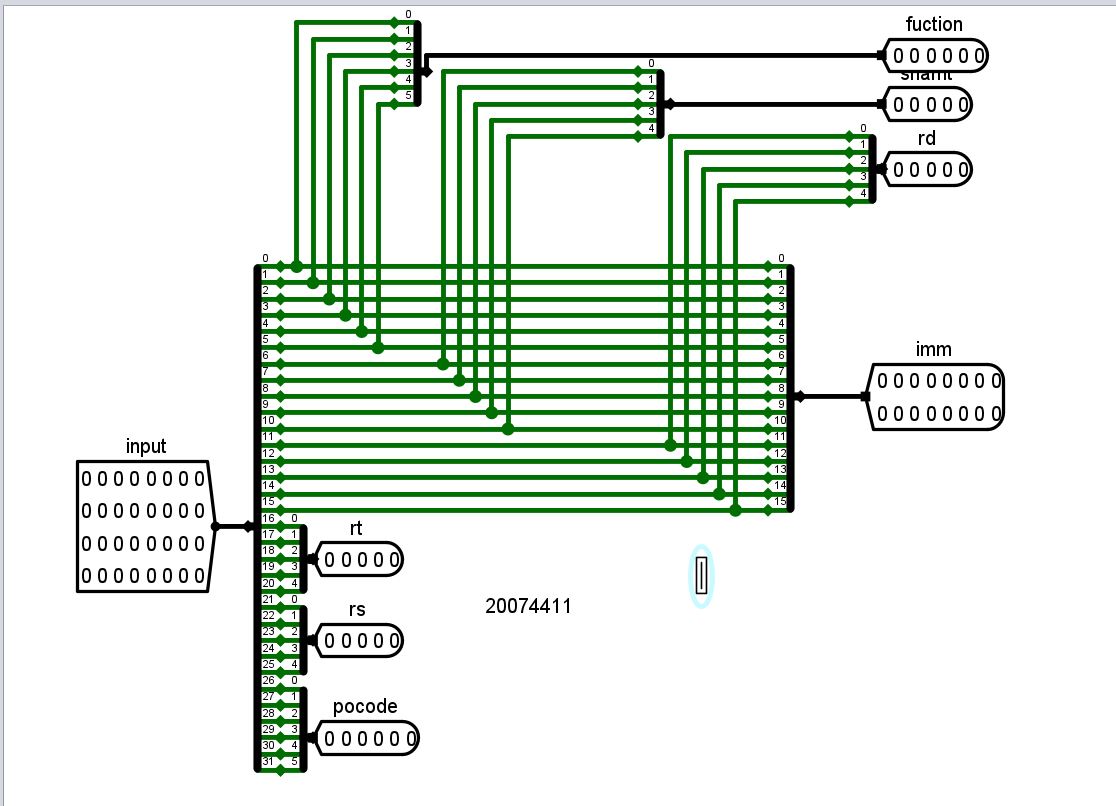
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| ExtOp | I | 立即数拓展模式选择。  00：高16位补0  01：高16位补1  10：低16位补0 |
| Imm16 | I | 16位输入 |
| Output[31:0] | O | 32位数 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 高16位补0 | ExtOp为00时，输出高16位0低16位imm组成的32位数 |
| 2 | 高16位补1 | ExtOp为01时，输出高16位1低16位imm组成的32位数 |
| 3 | 低16位补0 | ExtOp为10时，输出高16位imm低16位0组成的32位数 |

译码模块：

（1）基本描述

译码主要功能是从32位指令中取出不同类型指令各个部分的二进制代码。内部用一个分线器分出各个指令的长度的数据，用tunnel连接输出。

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Input[31:0] | I | 32位指令 |
| opcode[5:0]（图上打错了） | O | 32位指令的后六位，指令类型 |
| function[5:0] | O | 相应位置，和opcode一起确认指令类型 |
| shamt[4:0] | O | 相应位置，指定移位指令的位数 |
| rs[4:0] | O | 相应位置，source寄存器 |
| rt[4:0] | O | 相应位置，target寄存器 |
| rd[4:0] | O | 相应位置，destination寄存器 |
| imm[15:0] | O | 相应位置，i型指令用到的，输出为16位立即数 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 译码 | 将32位指令拆分输出 |

GPR模块：

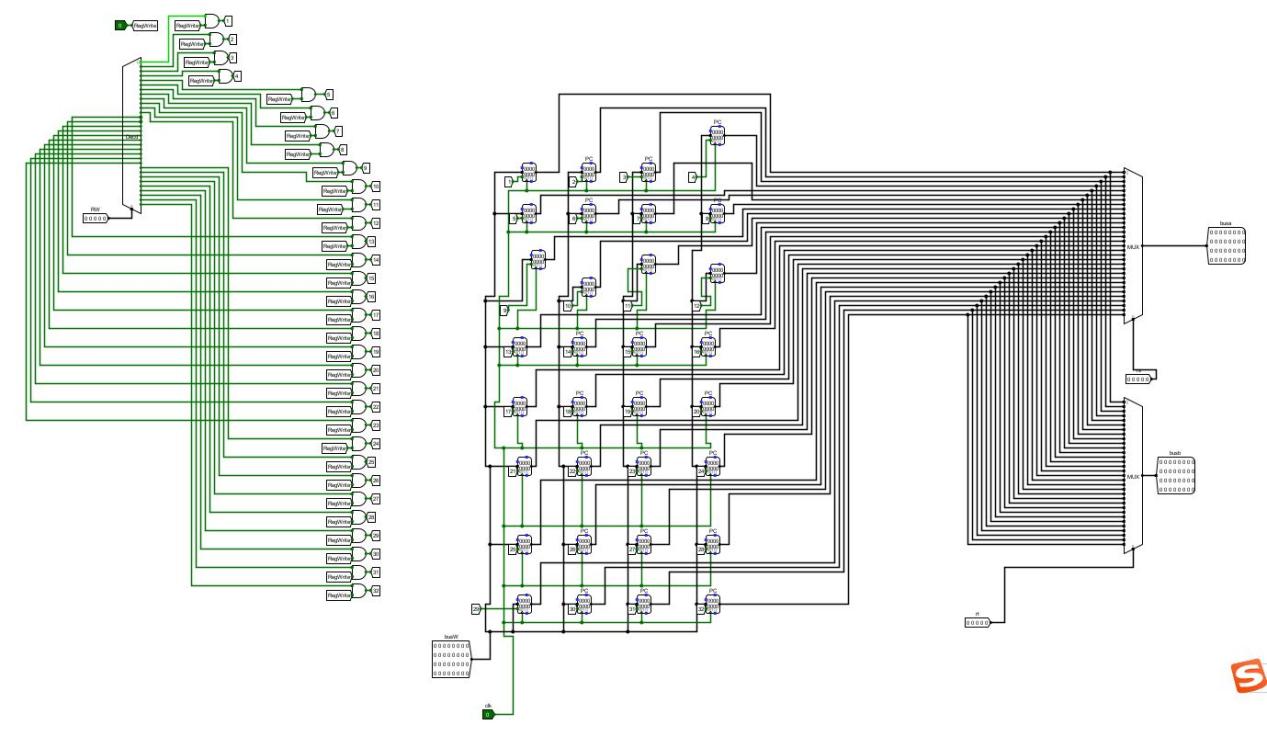
（1）基本描述

寄存器模块，两个读端口一个写端口。

寄存器写入需要写使能信号控制。写操作的输入必须在时钟边沿有效。

读操作读出以前写入的内容，而写入的内容在下一时钟周期才可读。

寄存器号的输入为5位，数据线为32位。整个模块由32个32位寄存器、decoder和多路选择器构成，每个寄存器有标号，分别对应mips标准的寄存器。



（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| RegWrite | I | 写入使能端，0不可写，1可写 |
| RW[4:0] | I | 写入选择端，控制写入哪个寄存器,5位控制32个 |
| clk | I | 时钟信号 |
| busW[31:0] | I | 要写入的数据 |
| rt | I | 选择读的地址1 |
| rs | I | 选择读的地址2 |
| busA | O | 读出的数据1 |
| busB | O | 读出的数据2 |

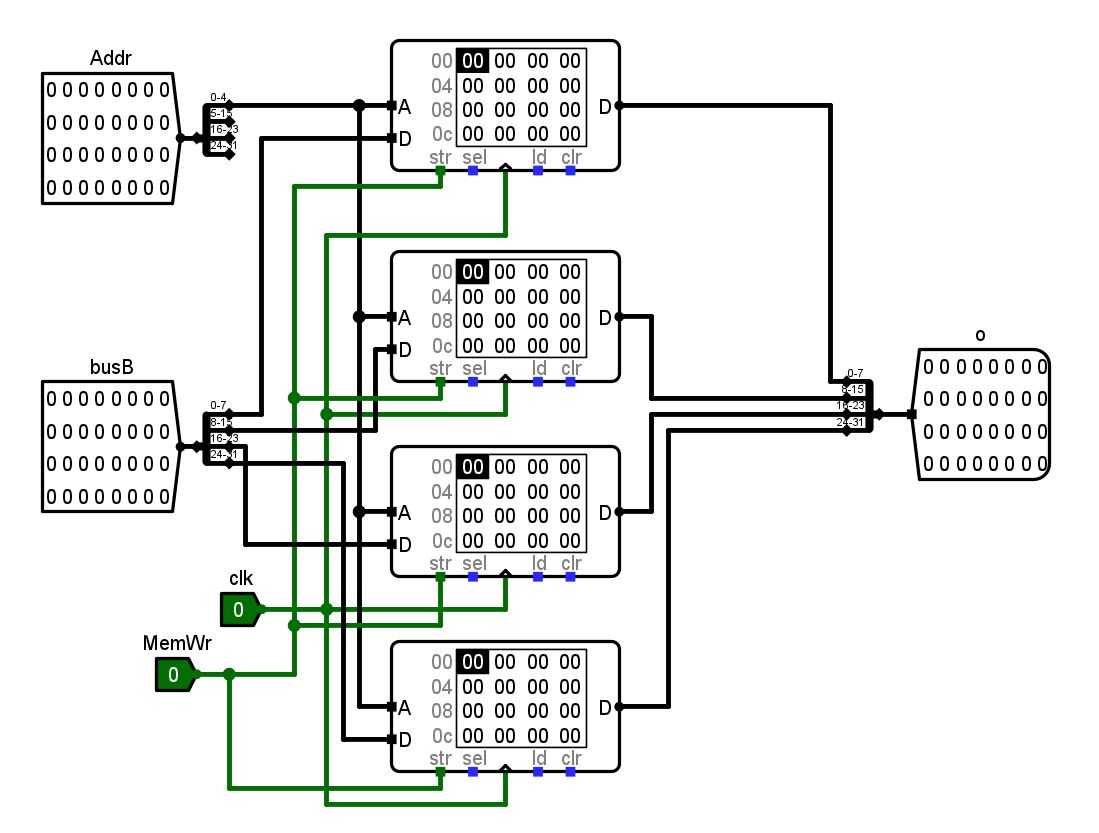
（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 写入 | 使能信号有效时根据选择端选择寄存器写入值 |
| 2 | 读出 | 根据选择端输入读出某寄存器的值 |

DM模块：

（1）基本描述

数据存储器，四个存储体并行小端序储存方式。输入为写入使能、地址（5位 2^5）和所写数据，一个输出为读出的数据。



（2）模块接口

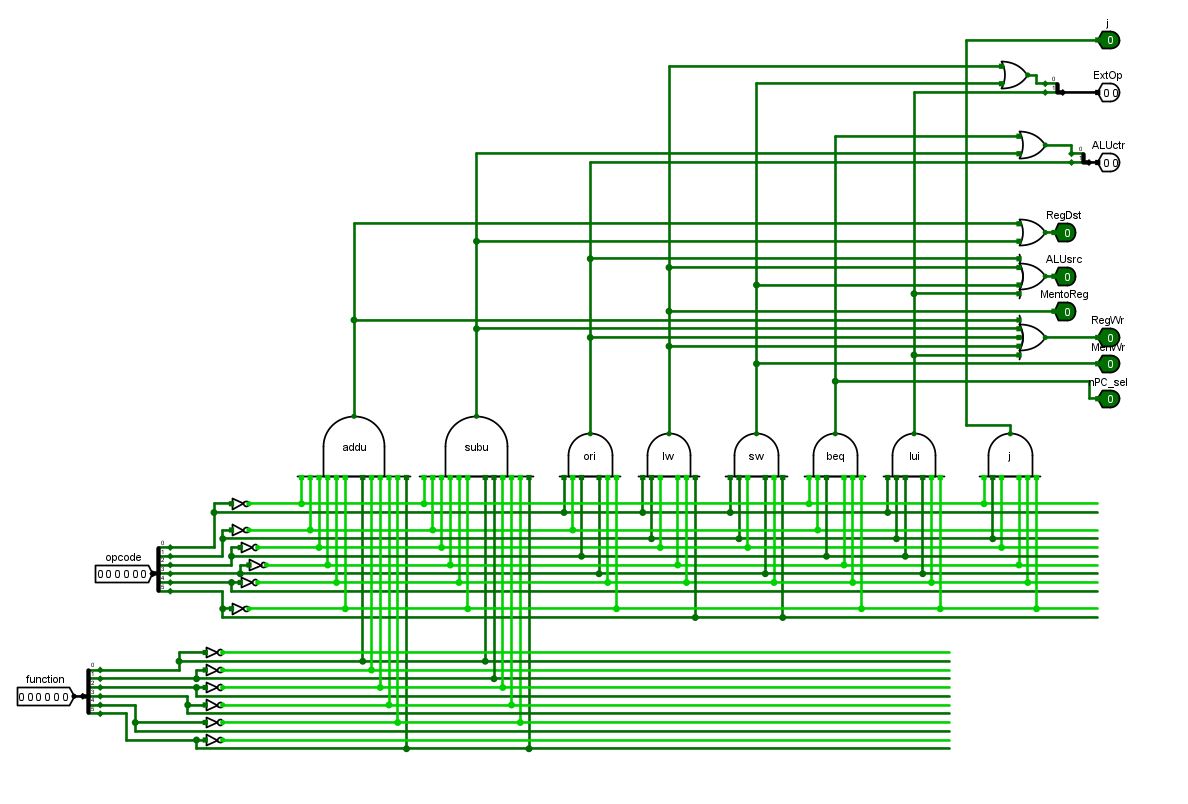
|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Addr[4:0] | I | 5位地址（RAM寻址2^5） |
| busB[31:0] | I | 要存入的数据 |
| MemWr | I | 数据存储器写入使能 |
| clk | I | 时钟信号 |
| o | O | 读出的数据 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 写入 | 根据使能信号和选择端将输入值写入RAM |
| 2 | 读出 | 根据选择端选择读出值 |

Control模块：

（1）基本描述

控制单元。根据译码器译出并输入的opcode与function指令，产生一系列使能信号、选择信号、控制信号

（2）模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Function[5:0] | I | 32位指令的function段 |
| opcode[5:0] | I | 32位指令opcode段 |
| J | O | 判断PC是否进行J指令的寻址方式 |
| ExtOp[2:1] | O | Extender的拓展方式 |
| ALUctr[2:1] | O | 控制ALU执行的运算 |
| nPC\_sel | O | 控制PC是否跳转 |
| MemWr | O | DM写使能 |
| RegWr | O | GPR写使能 |
| MemToReg | O | 选择DM写入寄存器的数据 |
| ALUSrc | O | ALU的选择端 |
| RegDst | O | GPR写入寄存器的目标寄存器 |

（3）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 产生控制信号 | 根据输入指令产生状态单元和ALU的控制信号、多路选择器的选择信号 |

## 机器指令描述

|  |  |  |
| --- | --- | --- |
| 指令操作码助记符 | 操作码的机器指令代码 | 指令功能的对应关系表 |
| addu | 000000+rs+rt+rd+0000010001 | $rd = $rs op $ rt |
| subu | 000000+rs+rt+rd+00000100011 | $rd = $rs op $ rt |
| ori | 001101+rs+rt+imm | $rt = $rs op E(imm) |
| lui | 001111+00000+rt+imm | $rt=imm<<16 |
| beq | 000100+rs+rt+imm | If($rs==$rt)PC=PC+E(imm)<<2 |
| bne | 000101+rs+rt+imm | If($rs!=$rt)PC=PC+E(imm)<<2 |
| j | 000010+address | PC <- (PC+4)[31..28],address,0,0 |
| lw | 100011+rs+rt+imm | $rt=memory[$rs+(sign-extend)] |
| sw | 101011+rs+rt+imm | memory[$rs+(sign-extend)]=$rt |

## 编写的测试程序

这里采用老师给出的p1-test测试文件

34030093

340600ae

00664021

00664823

012a0021

ac090010

8c0a0010

112a0002

3c0bcdcd

08000c0d

340befef

3c094567

08000c07

以下为mips汇编代码

ori $3,$0,0x93 ;$3 = $0||0x93

ori $6,$0,0xae ;$6 = $0||0xae

addu $8,$3,$6 ;$8=$3+$6(无符号加法)

subu $9,$3,$6 ;$9=$3-$6(无符号减法)

addu $0,$9,$10 ;$0=$9+$10(无符号加法)

sw $9,16($0) ;将$9内容存入储存器基址为0偏移量为16的内存单元

lw $10,16($0) ;将储存器基址为0偏移量为16的内存单元的内容存入$10寄存器

l3: beq $9,$10,l1 ;$9和$10内容相等则跳转到l1

lui $11,0xcdcd ;$11低16位补0高16位填入cdcd

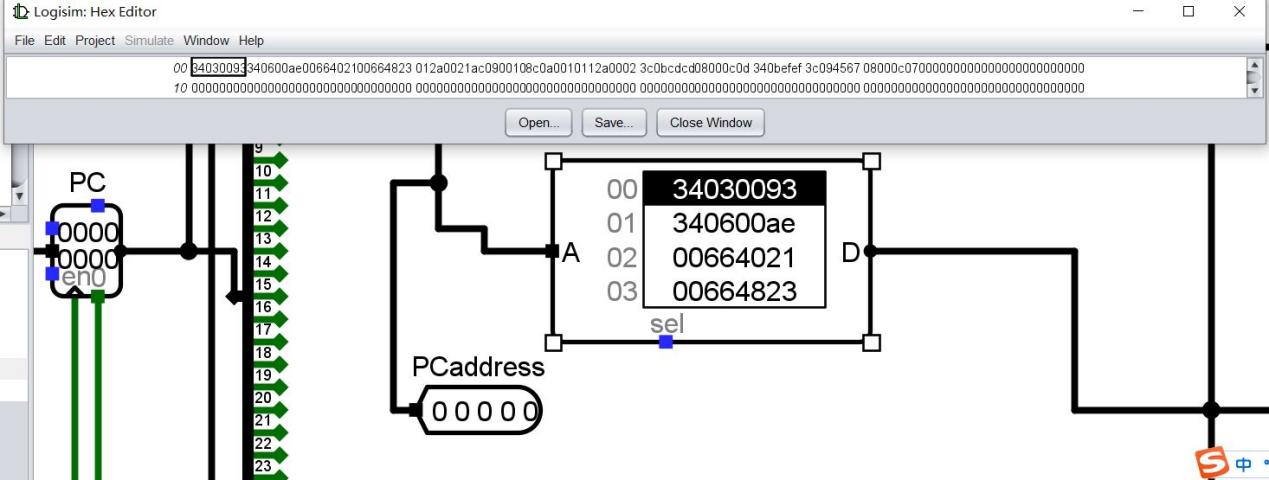
j end ;跳转到end

l1: ori $11,$0,0xefef ;$11 = $0||0xefef

lui $9,0x4567 ;$9低16位补0高16位填入4567

j l3 ;跳转到l3

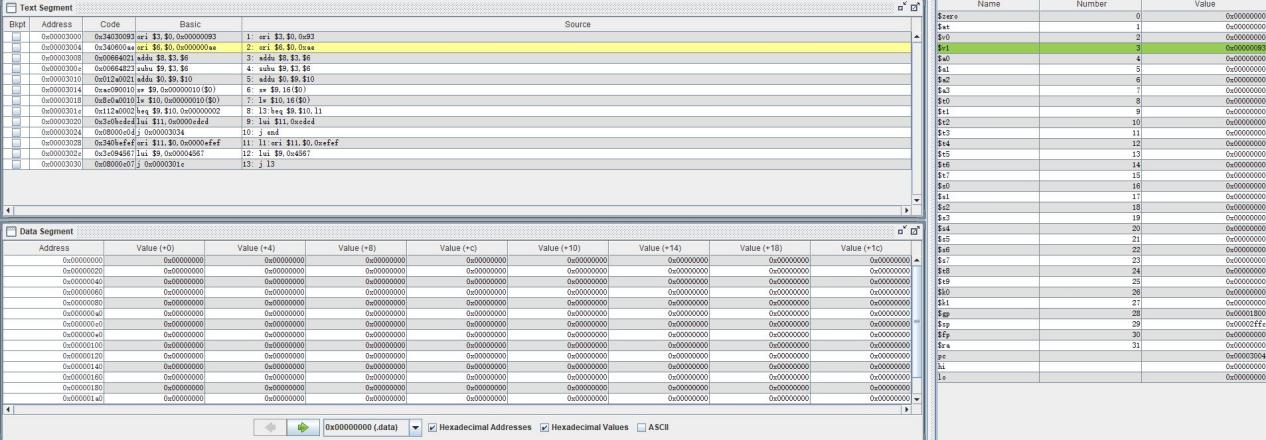
end:

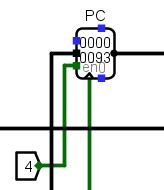


上图为IFU中提前写入的指令

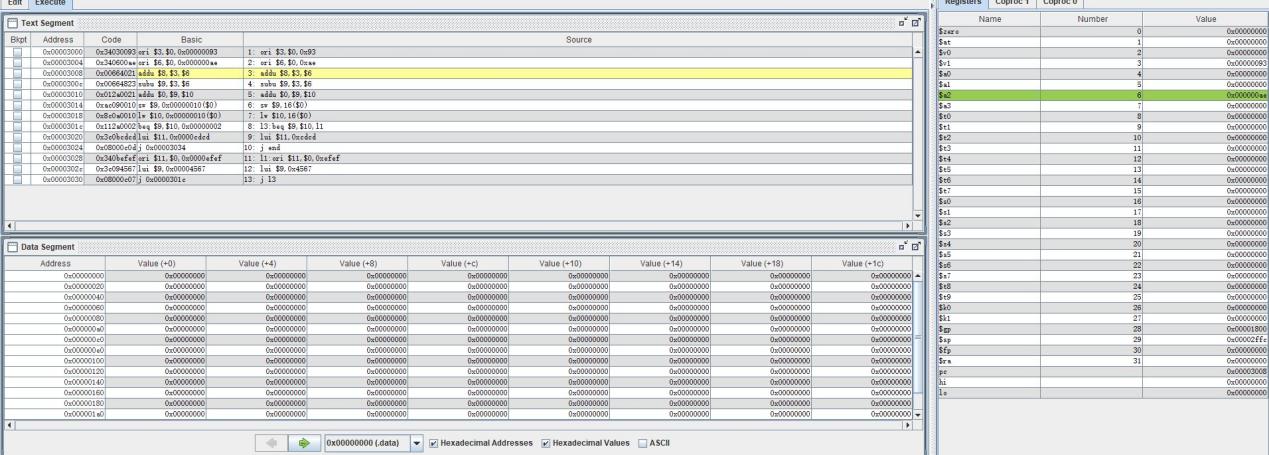
## 与测试程序相对应的测试结果截图

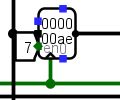
**1**



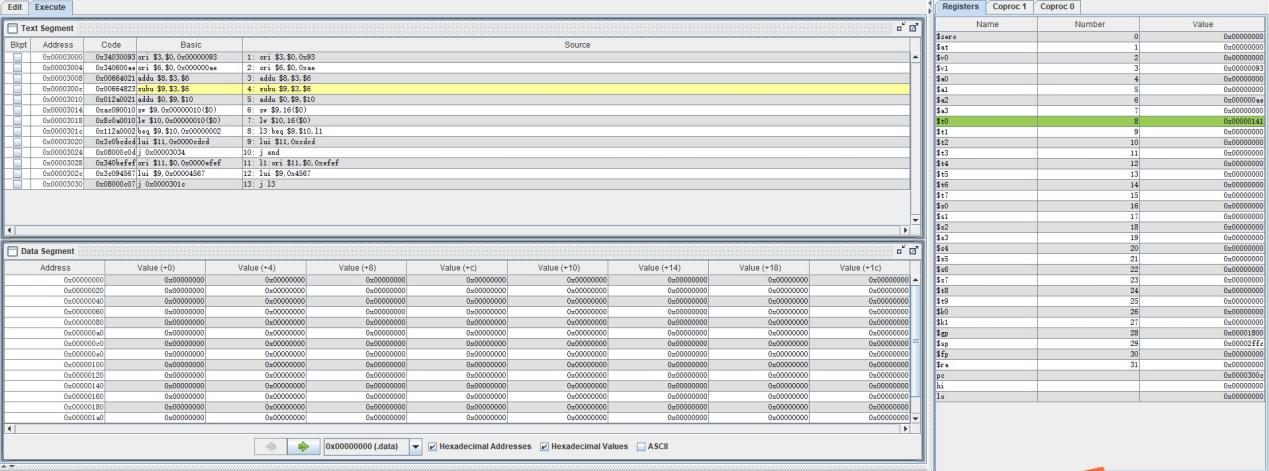
第4个寄存器$v1（编号为3）的值被写为00000093

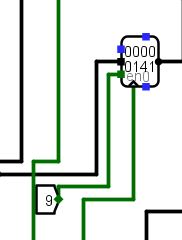
**2**



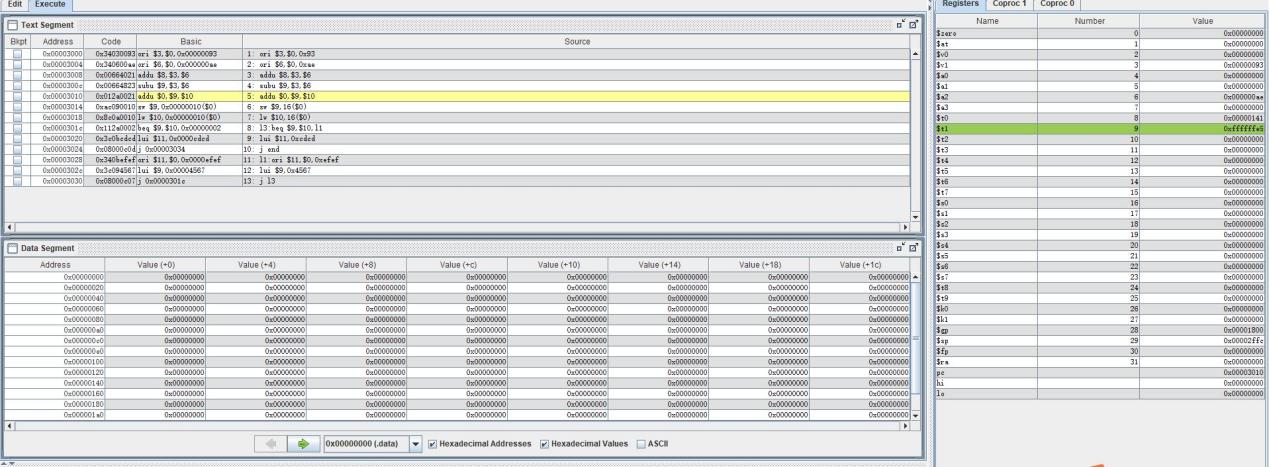
第7个寄存器$a2（编号为6）的值被写为000000ae

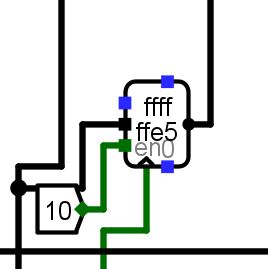
**3**



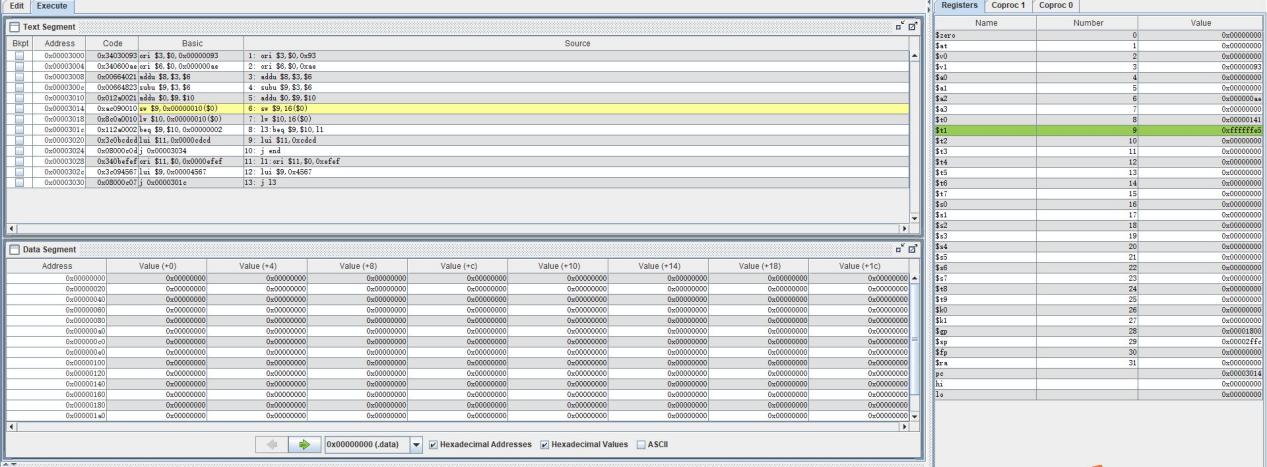
第9个寄存器$t0（编号为8）的值被写为00000141

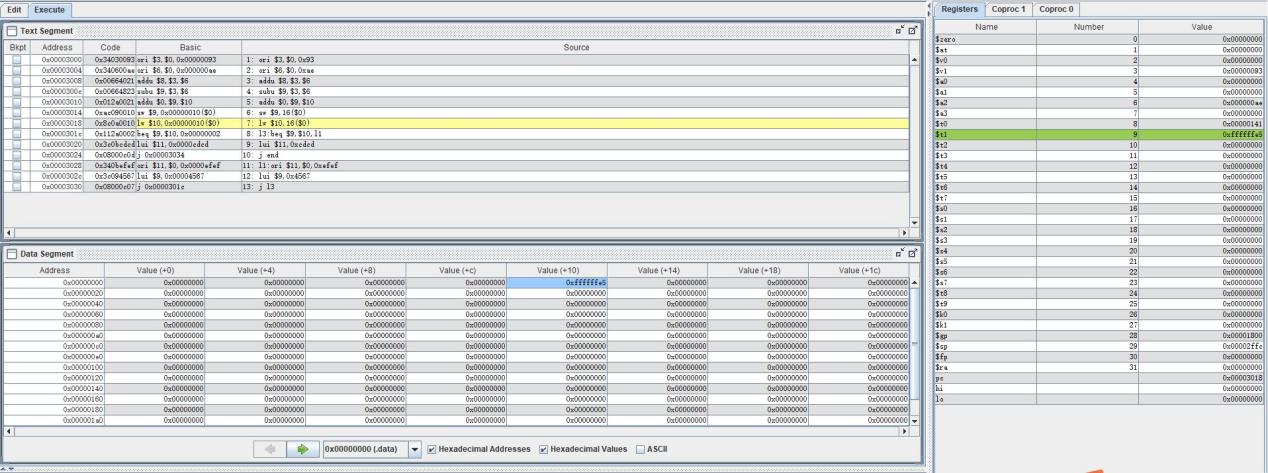
**4**



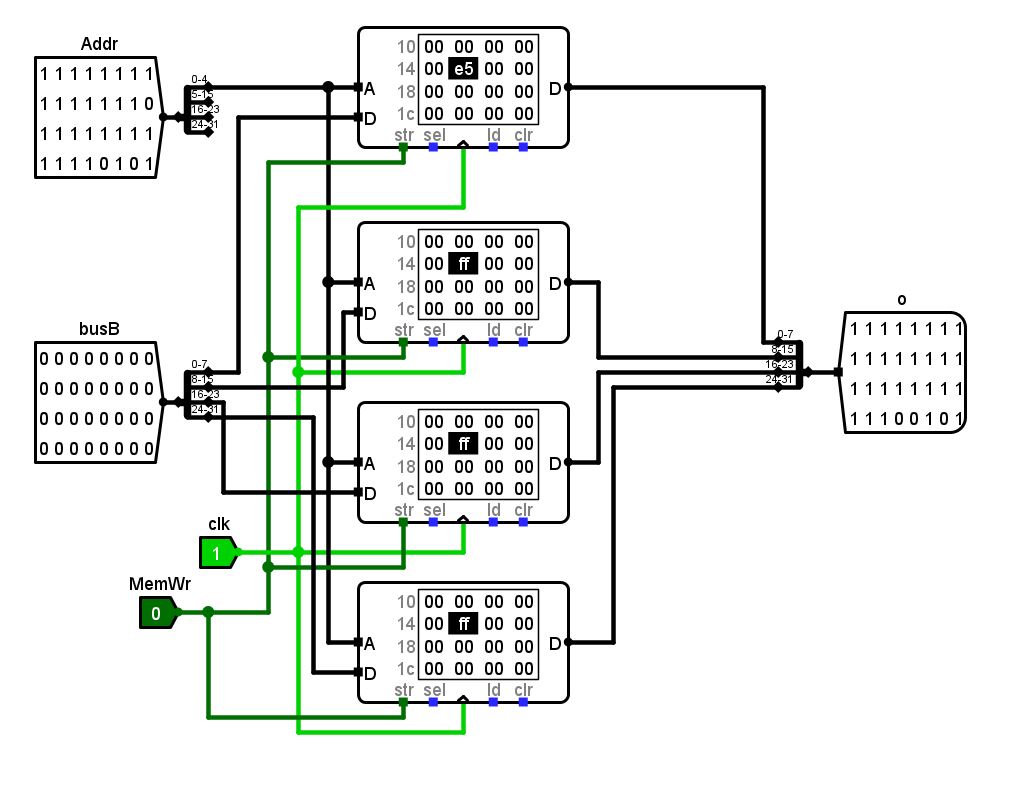
第10个寄存器$t1（编号为9）的值被写为ffffffe5

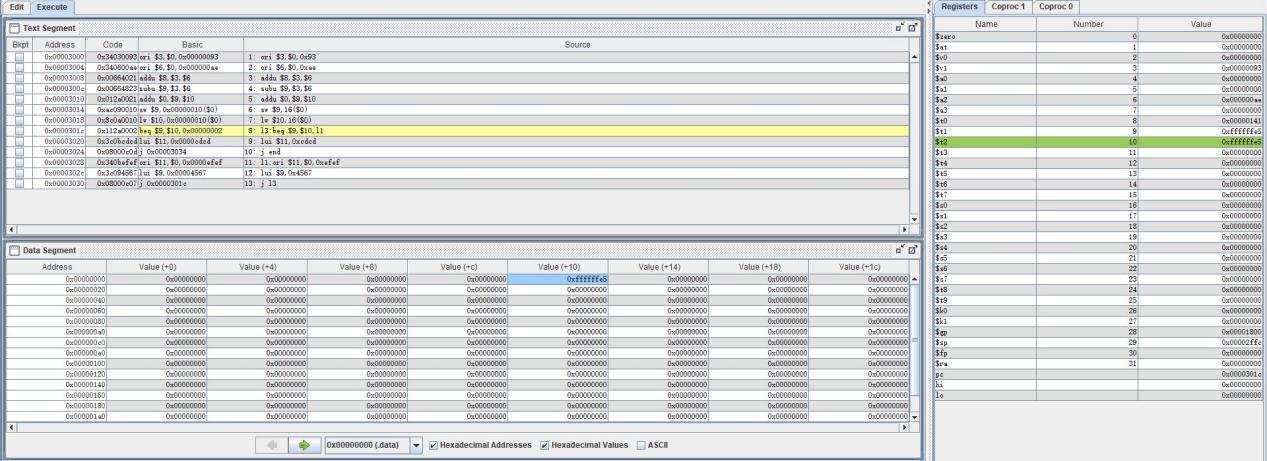
**5、6**



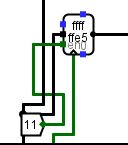


sw执行，$t1的内容被写入DM相应位置

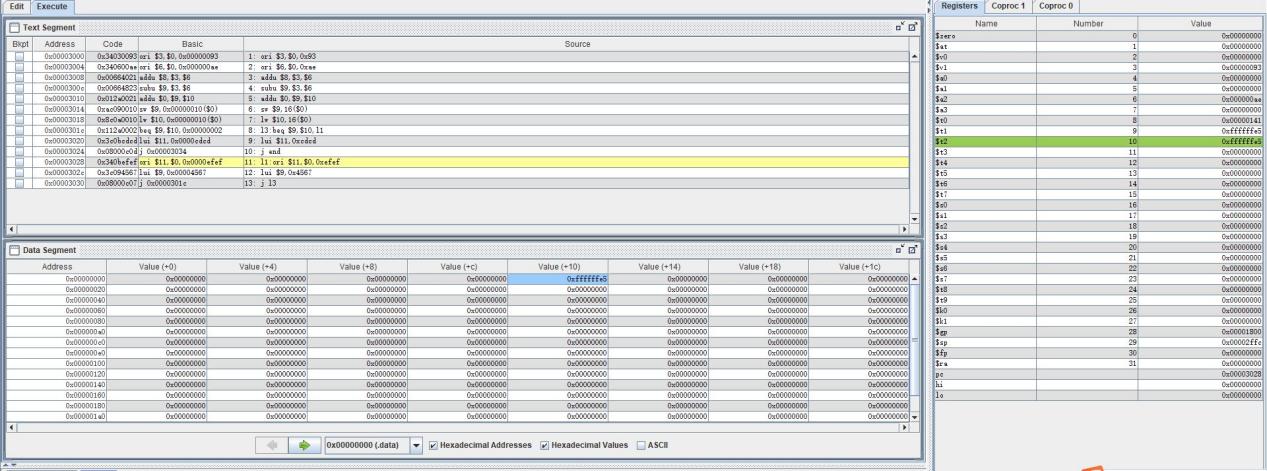
**7**



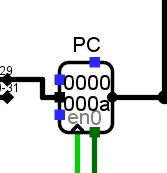
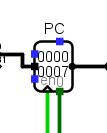
Lw执行，DM中的值被写入第11个寄存器$t2（编号为10）



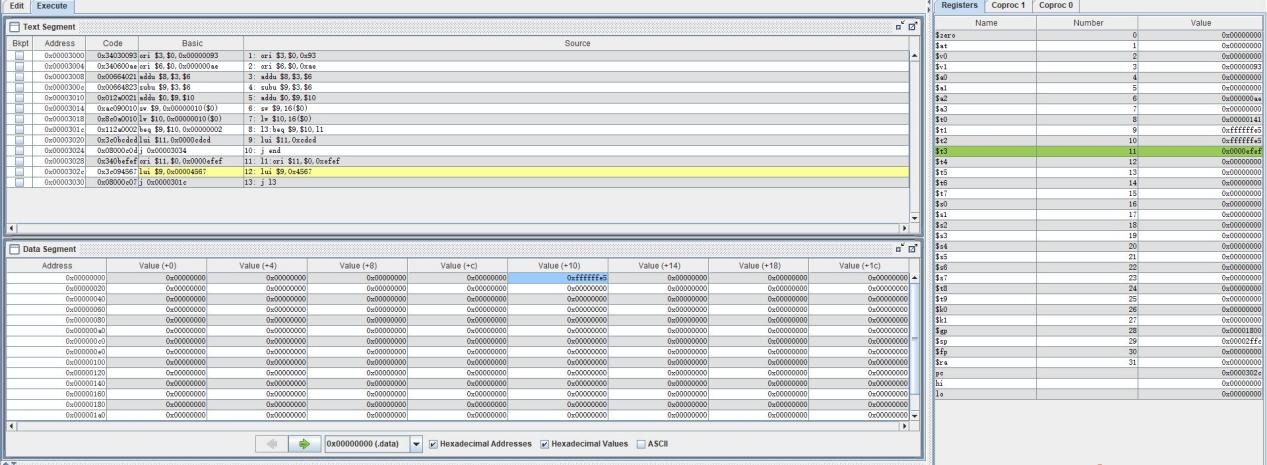
**8**



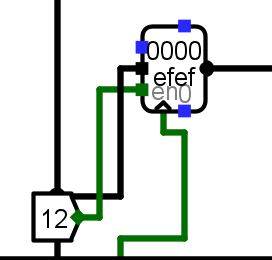
beq判断后PC跳转



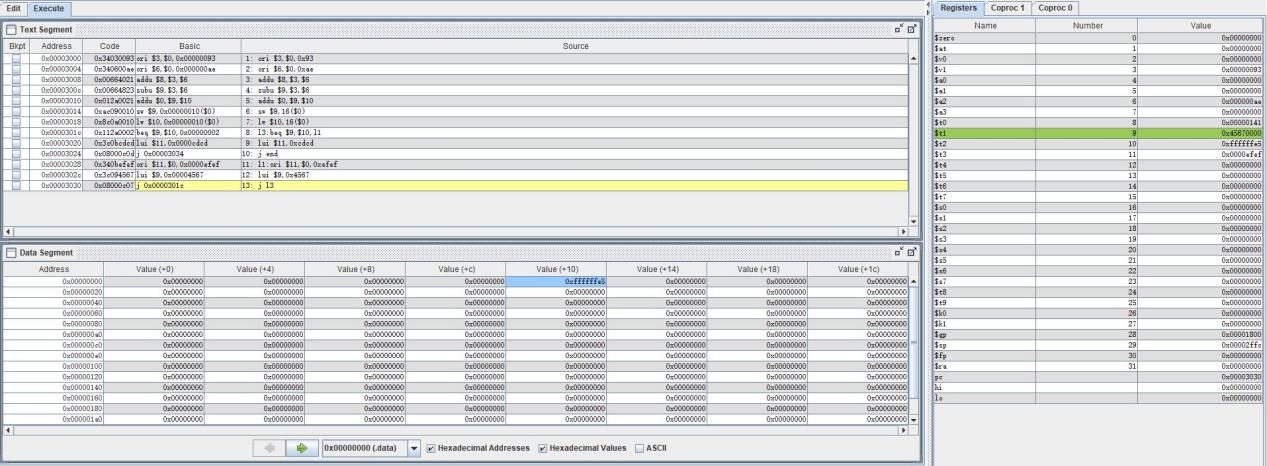
**9**

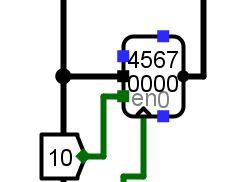


Ori指令执行，第12个寄存器$t3（编号为11）的值被写为0000efef

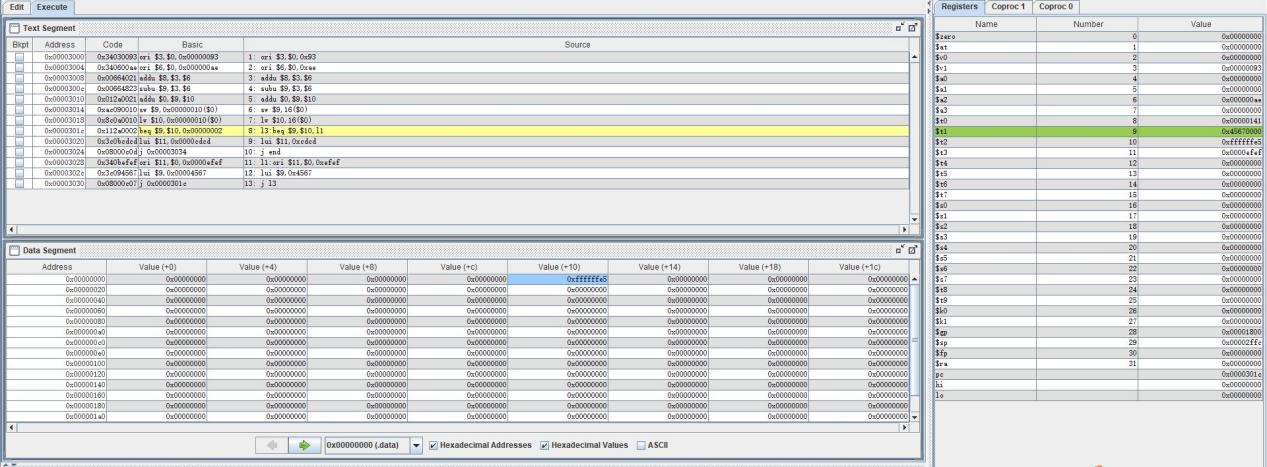


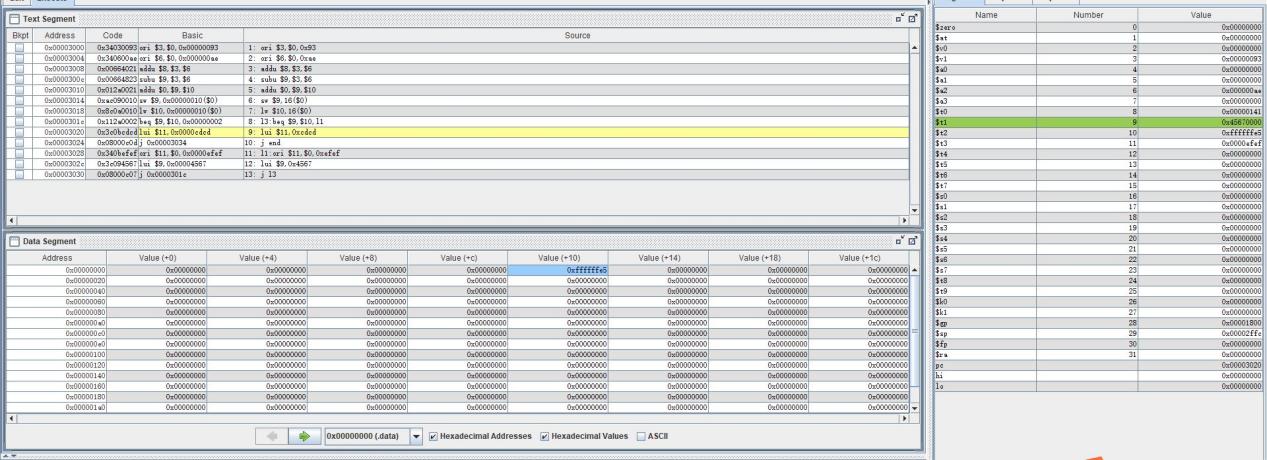
**10**



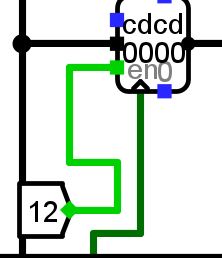
lui指令执行，第10个寄存器$t1（编号为9）的值被写为45670000

**11、12**

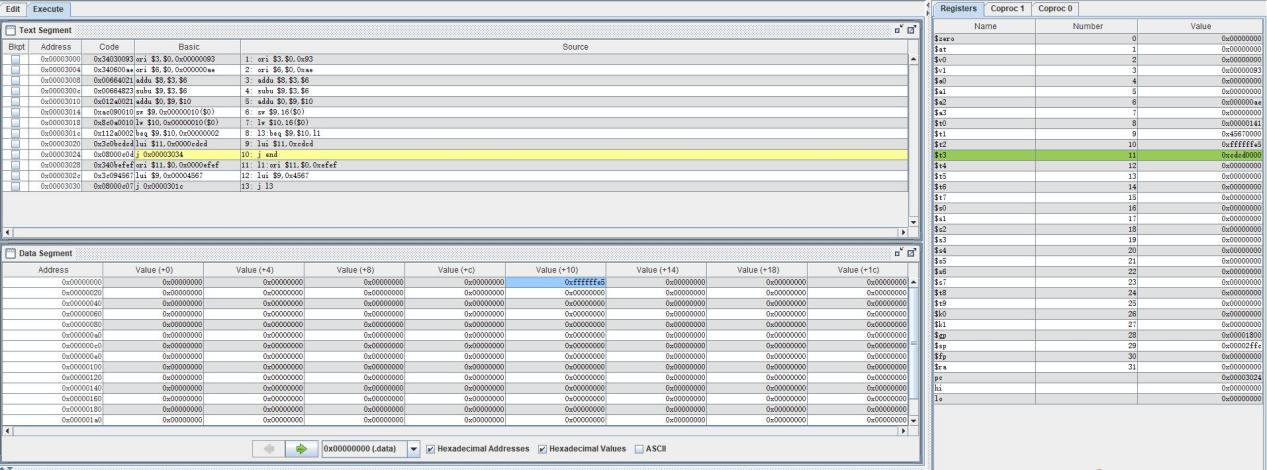




返回l3进行beq判断，这次不跳转并运行lui指令，第12个寄存器$t3（编号为11）的值被写为cdcd0000



**13**



跳转到end结束程序

## 设计过程的收获、体会及总结

做完这次大作业，加上之前在数逻课上用logisim的经历，首先我对logisim的使用已经非常熟练了：我对各类逻辑门还有比如splitter、constant、tunnel、multiplexer等等各种元件的使用以及它们能够实现的功能更加熟悉了。我从加法器开始，自己动手设计了运算模块、译码器等等，一直到最终做出了能够运行mips lite指令的一个简陋的cpu，这让我对计算机的原理架构等等都有了十分深入的了解和体会，也对相关知识比如各种进制正负数原码补码加减、计算机数据的储存读写等等理解更深刻了。

这次大作业让我能够集合运用学到的知识制作出可以使用的器件，虽然初期有一些困难有一点懵，但是随着逐渐熟悉操作、理解整体，我在大作业的完成过程中收获了十足的成就感，有时候想通了连线会连得废寝忘食，如果连好了直接运行还通过了那就更高兴了，整个人心情都会特别好。

总之大作业让我对计算机相关硬件、软件，相关职业，甚至类似的学科和实验课都有了更多了解，收获颇丰，也让我明白了面对困难一定要坚持下去，这样才能成功。之前数字逻辑课做大作业整个人是比较懵的，在计算机组成原理的大作业里我才有了融会贯通学到的知识，清楚地知道每个模块是干什么的感觉，这次大作业也让我深刻地体会到计算机是人类智慧的结晶，尤其是cpu这样一个小小的元件竟然能够如此精巧，在做这个单周期处理器之前我也去各种视频平台上了解了流水线处理器乱序执行、分支预测、超标量处理器以及多核处理器等等更加深入的知识，我深深地被相关科研人员和工程师们的智慧折服，并且我也因为满满的收获感深深地爱上了我所学习的专业。总之，这次大作业给我留下了非常深刻的印象。