lab6 logisim综合实验

姓名：宋玮　学号：PB20151793　实验日期：2022.5.26

实验题目

lab6 logisim综合实验

实验平台

Rars, logisim

实验结果

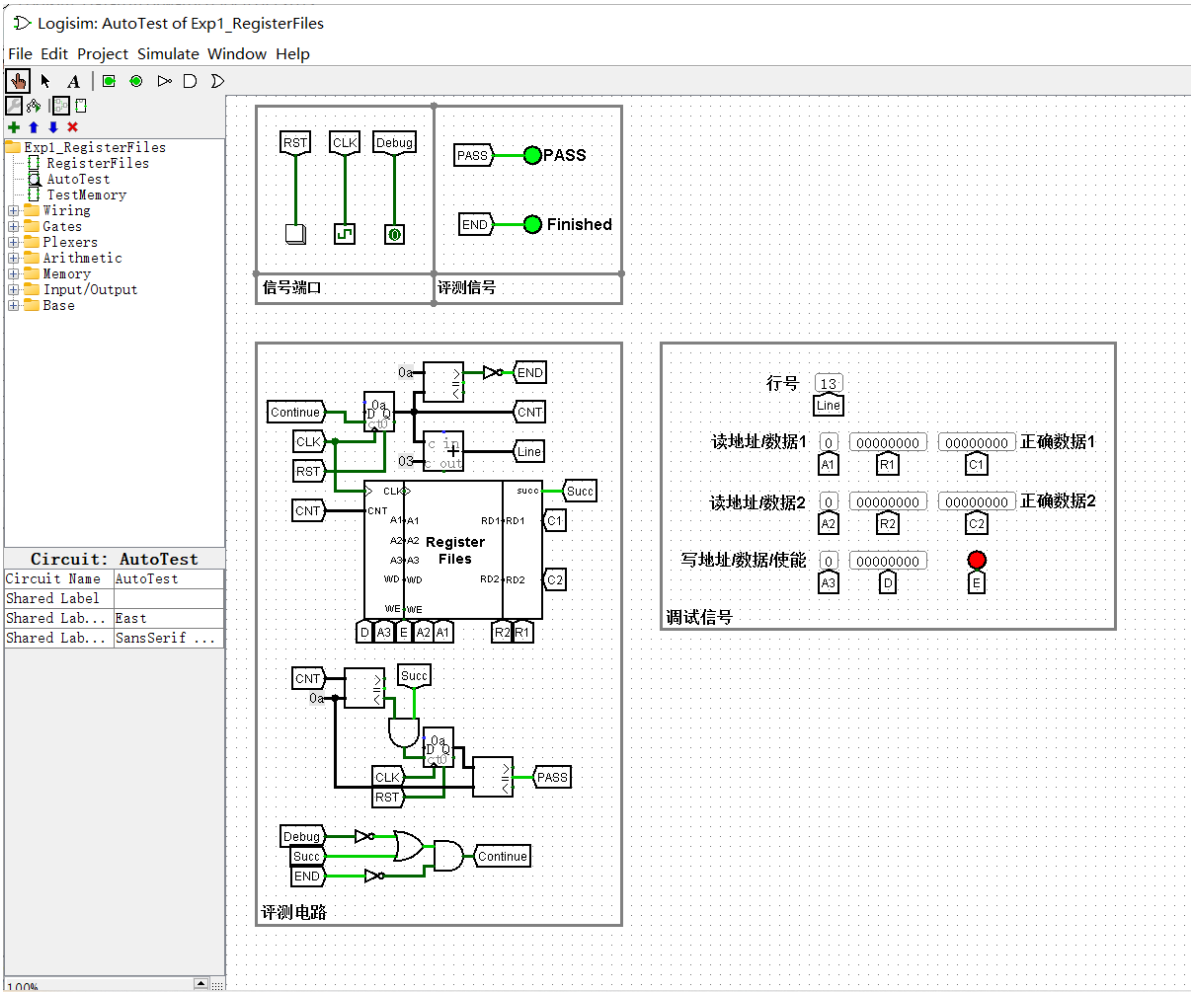
在logisim平台上实现了以下内容：

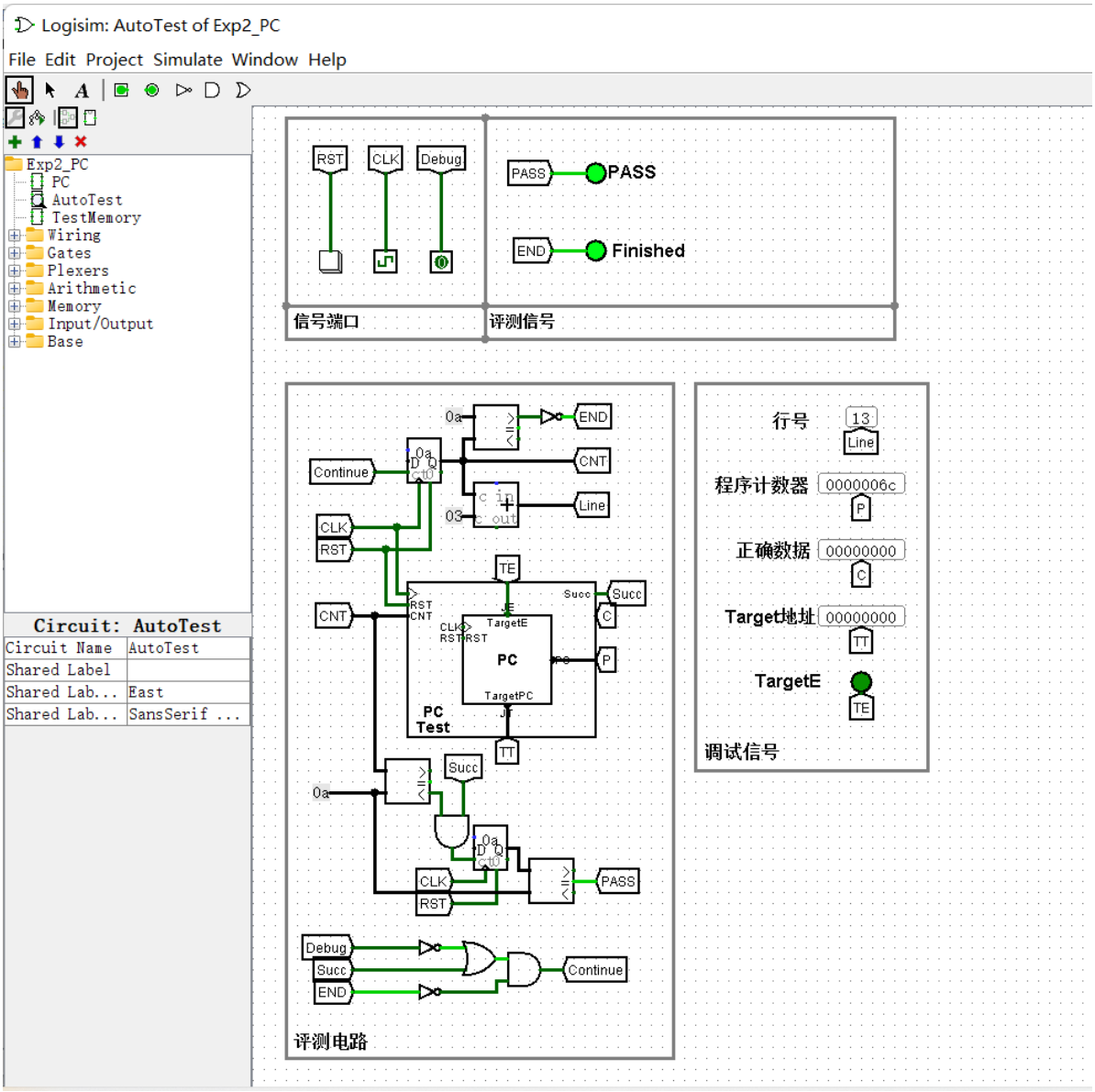
> 1. **单周期CPU**
> 2. **流水线CPU**
> 3. **能实现ecall指令以及外设输入的流水线CPU**

实验过程

一.基础模块

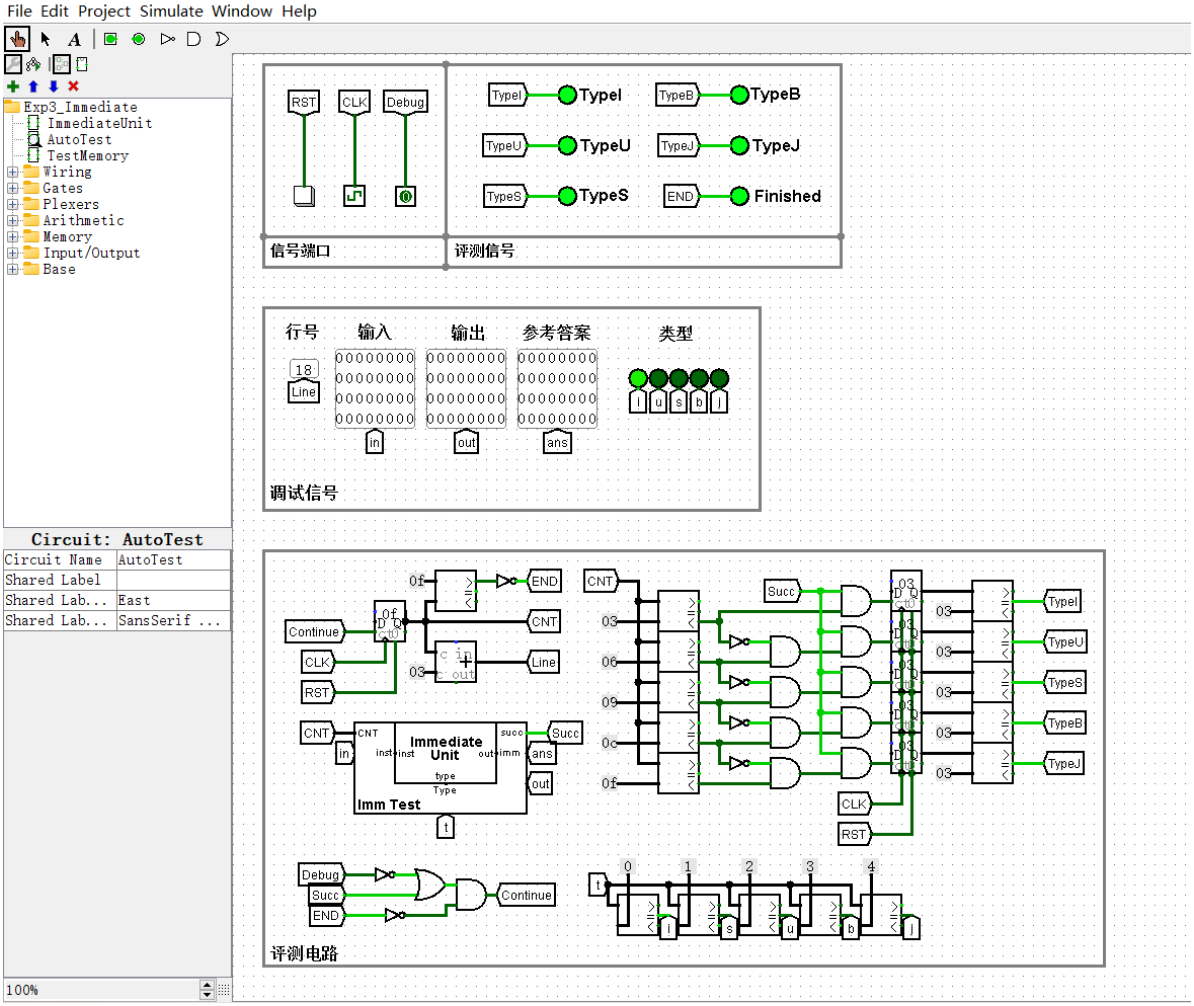
基础模块包括从Exp1~Exp7的Registerfiles，PC，Immediate，Branch，ALU，Memory，Controller模块. 全部测试如下：




Logisim: AutoTest of Exp2_PC
File Edit Project Simulate Window Help
Exp2_PC
PC
AutoTest
TestMemory
Wiring
Gates
Plexers
Arithmetic
Memory
Input/Output
Base
Circuit: AutoTest
Circuit Name AutoTest
Shared Label
Shared Lab... East
Shared Lab... SansSerif ...
RST
CLK
Debug
PASS
PASS
END
Finished
信号端口
评测信号
0a
END
Continue
CNT
03
Line
CLK
RST
TE
Succ
CNT
TargetE
PC
TargetPC
PC Test
TT
0a
Succ
CLK
RST
PASS
Debug
Succ
END
Continue
评测电路
行号
13
Line
程序计数器
0000006c
P
正确数据
00000000
C
Target地址
00000000
TT
TargetE
TE
调试信号

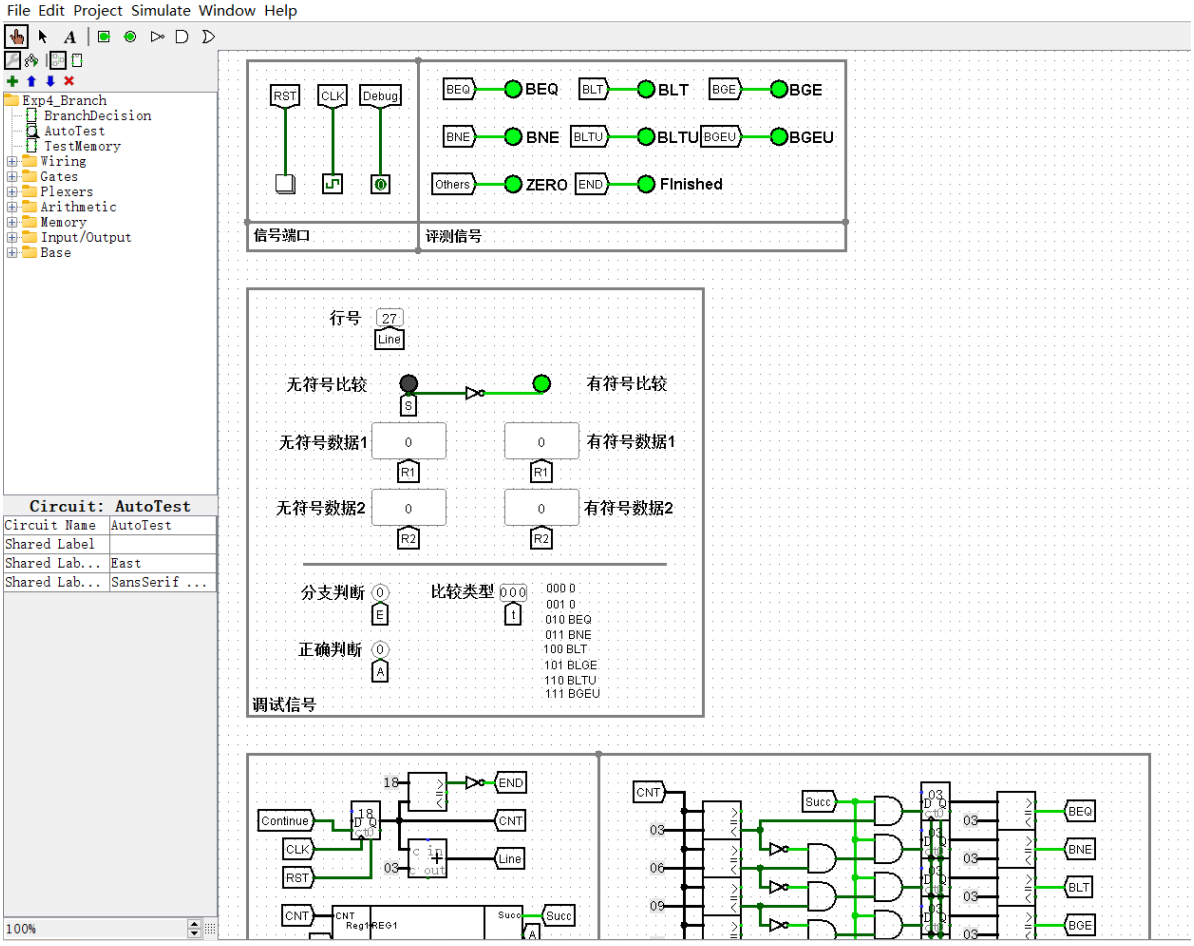
Logisim: AutoTest of Exp3_Immediate

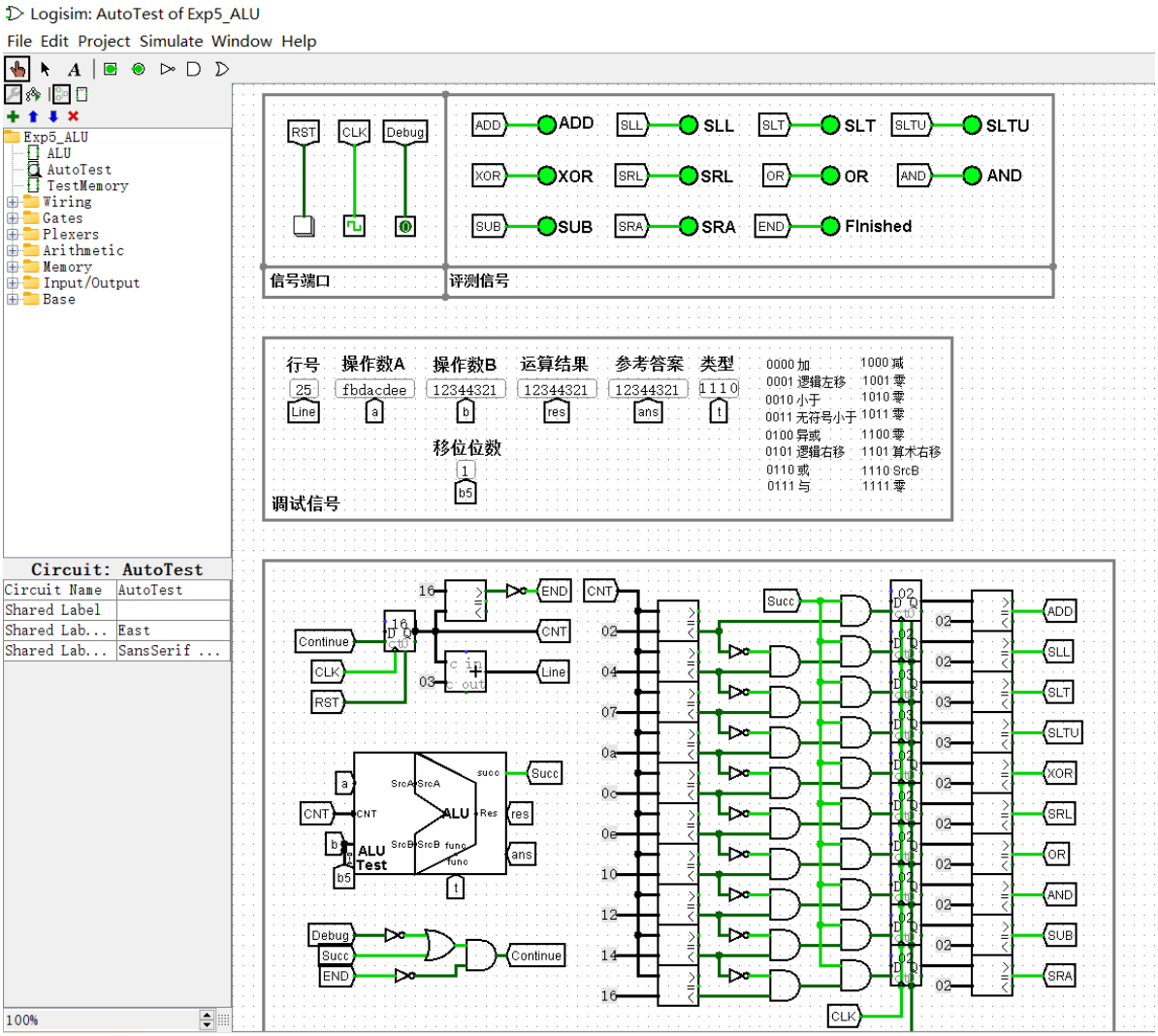
File Edit Project Simulate Window Help

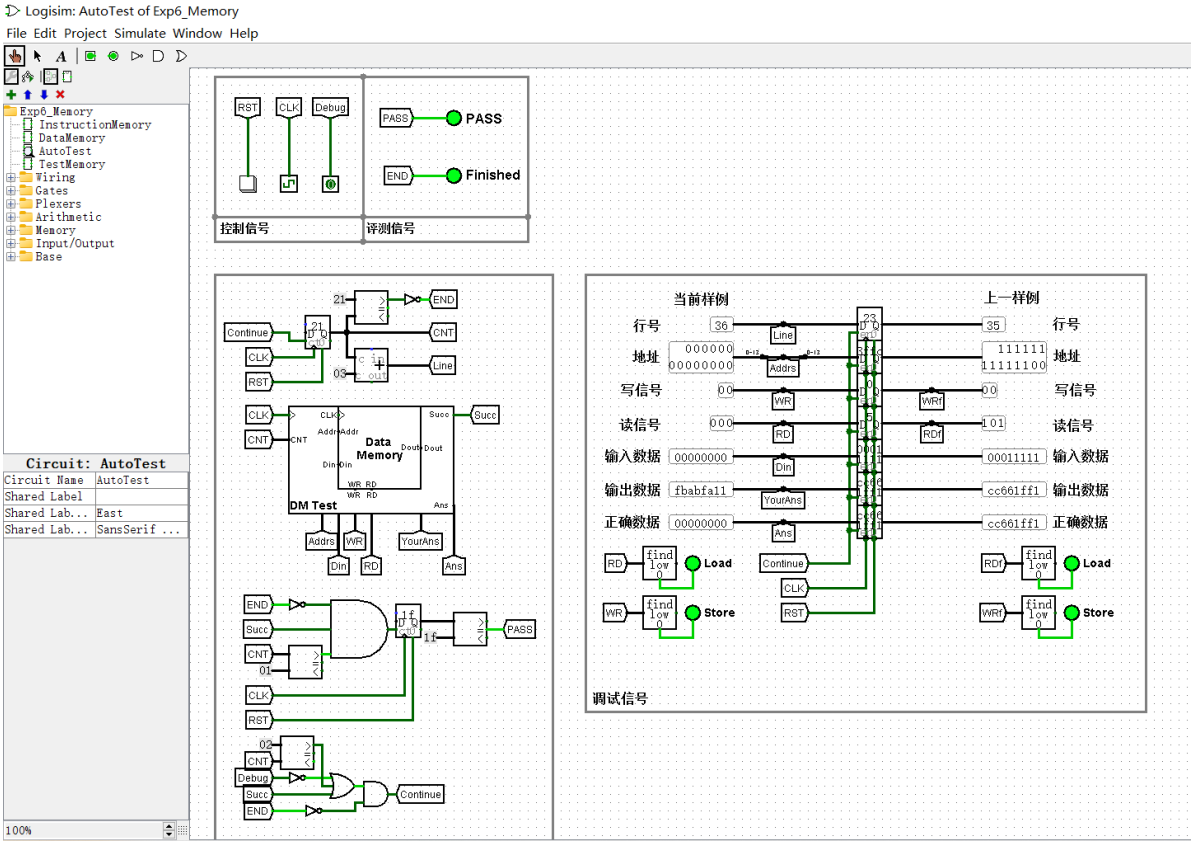


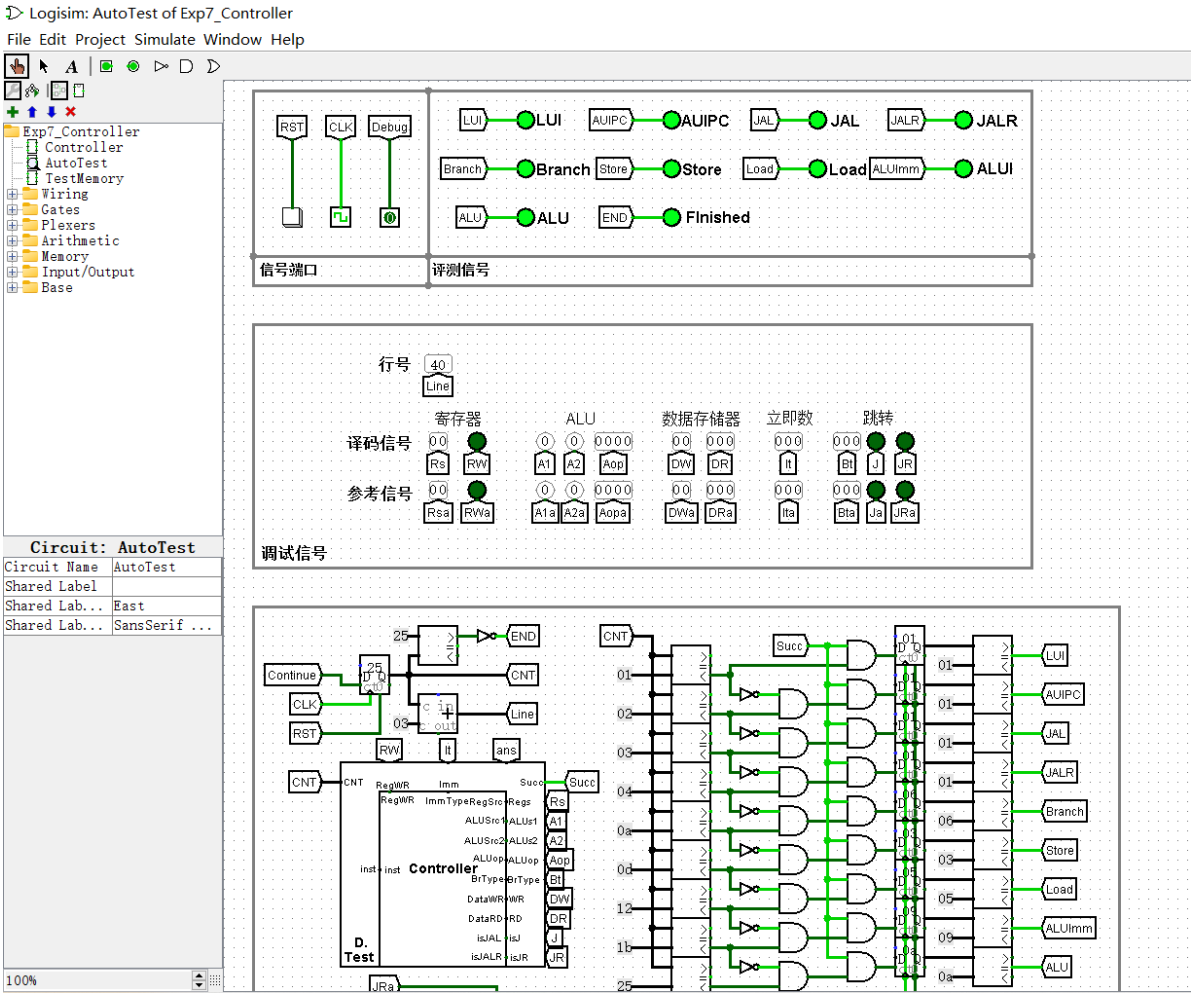
Logisim: AutoTest of Exp4_Branch

File Edit Project Simulate Window Help



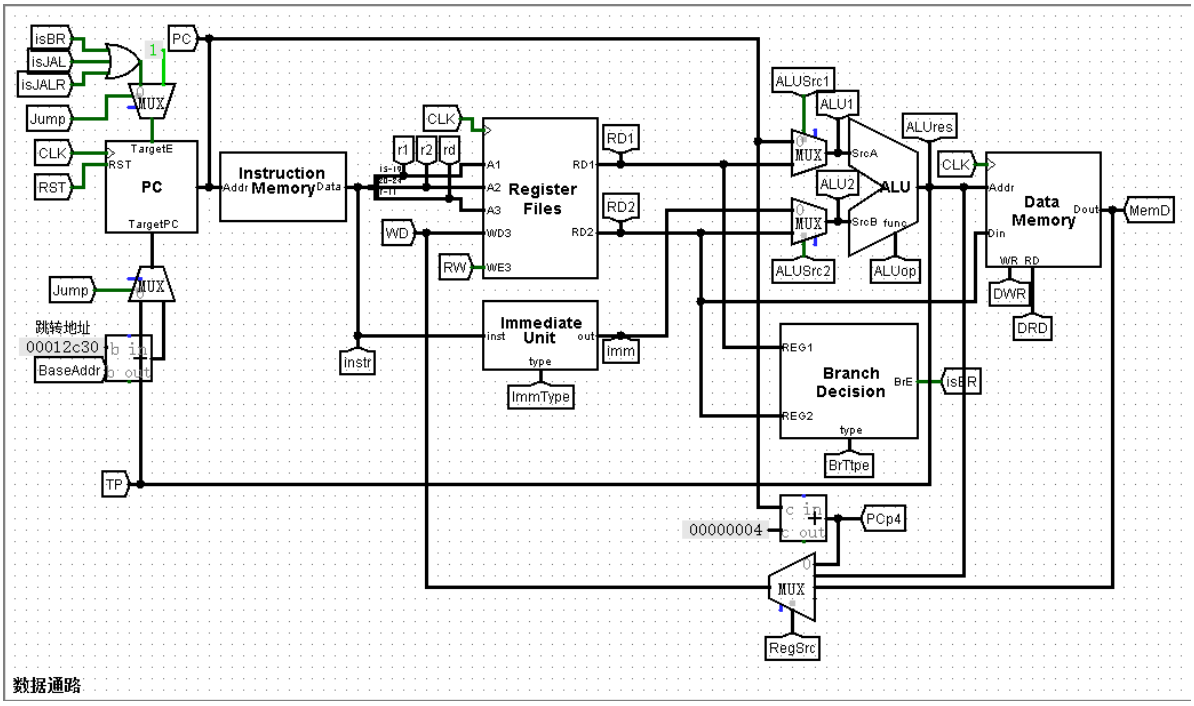

Logisim: AutoTest of Exp5_ALU
File Edit Project Simulate Window Help
Exp5_ALU
ALU
AutoTest
TestMemory
Wiring
Gates
Plexers
Arithmetic
Memory
Input/Output
Base
RST
CLK
Debug
ADD
SLL
SLT
SLTU
XOR
SRL
OR
AND
SUB
SRA
END
FInished
信号端口
评测信号
行号
操作数A
操作数B
运算结果
参考答案
类型
25
fbdacdee
12344321
12344321
12344321
1110
Line
a
b
res
ans
t
移位位数
1
b5
调试信号
0000 加
0001 逻辑左移
0010 小于
0011 无符号小于
0100 异或
0101 逻辑右移
0110 或
0111 与
1000 减
1001 零
1010 零
1011 零
1100 零
1101 算术右移
1110 SrcB
1111 零
Circuit: AutoTest
Circuit Name
AutoTest
Shared Label
Shared Lab...
East
Shared Lab...
SansSerif ...
Continue
CLK
RST
CNT
Line
END
ALU
ALU Test
Succ
Debug
100%


Logisim: AutoTest of Exp6_Memory
File Edit Project Simulate Window Help
Exp6_Memory
InstructionMemory
DataMemory
AutoTest
TestMemory
Wiring
Gates
Plexers
Arithmetic
Memory
Input/Output
Base
RST
CLK
Debug
PASS
END
Finished
控制信号
评测信号
Data Memory
DM Test
Addrs
WR
RD
YourAns
Din
Ans
Succ
CNT
Continue
当前样例
上一样例
行号
地址
写信号
读信号
输入数据
输出数据
正确数据
36
35
00000000
00000000
fbabfa11
cc661ff1
Load
Store
调试信号
Circuit: AutoTest
Circuit Name
AutoTest
Shared Label
Shared Lab...
East
Shared Lab...
SansSerif ...
100%



二.单周期CPU

完整数据通路如下：



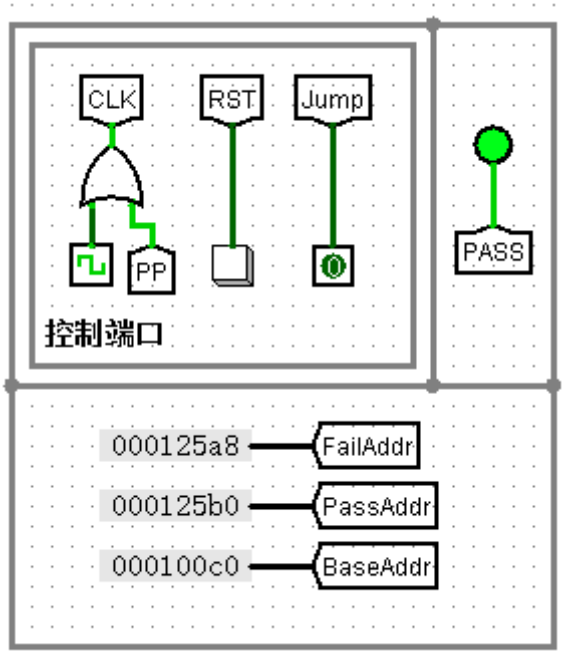
完成的部分包括：

- ALU的输入信号
- 分支判断器的输入信号

- 寄存器文件的写回数据`WD`的选择
- 数据存储器的输入信号

向指令存储器和数据存储器导入相应文件

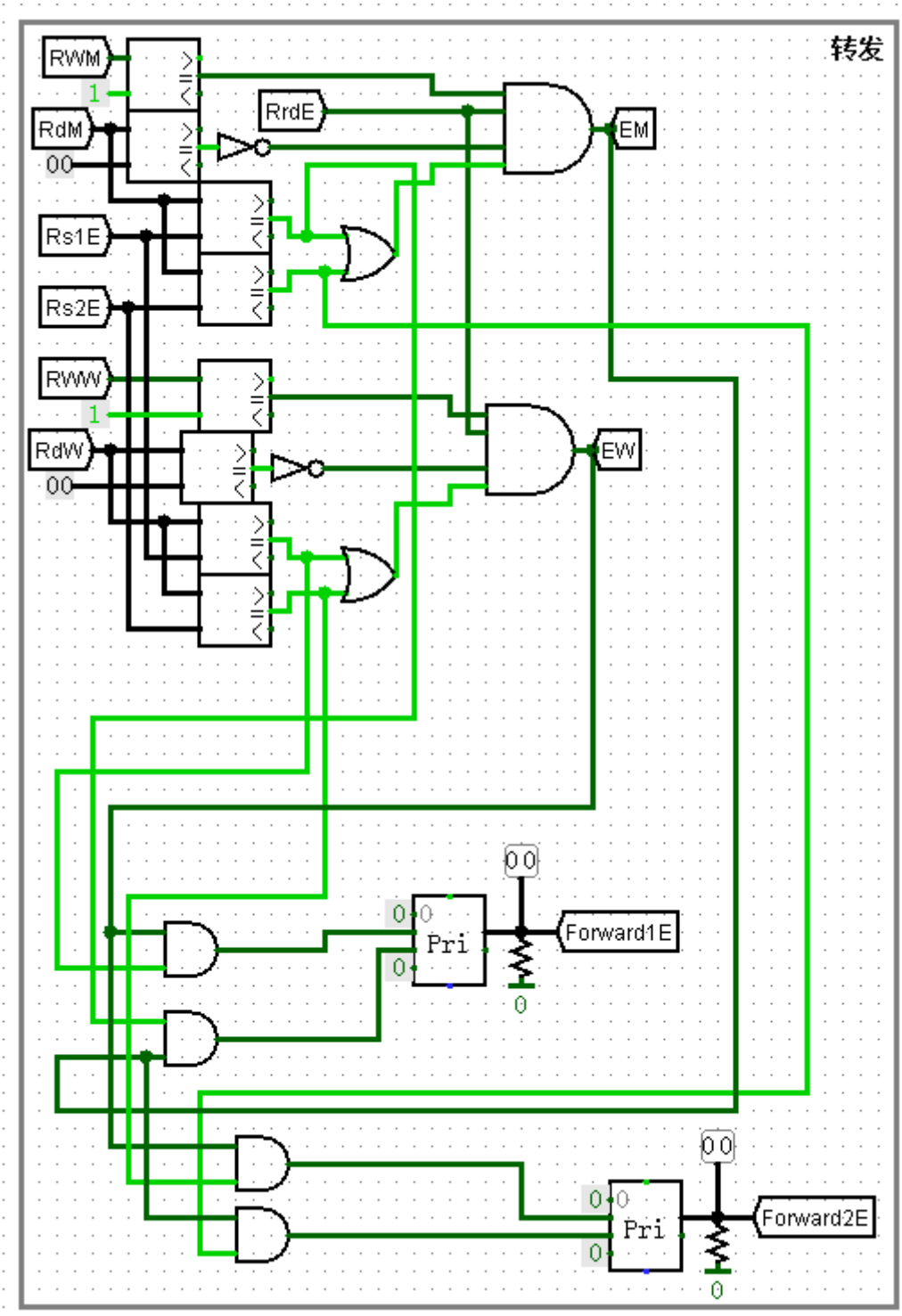
最终测试结果如下：



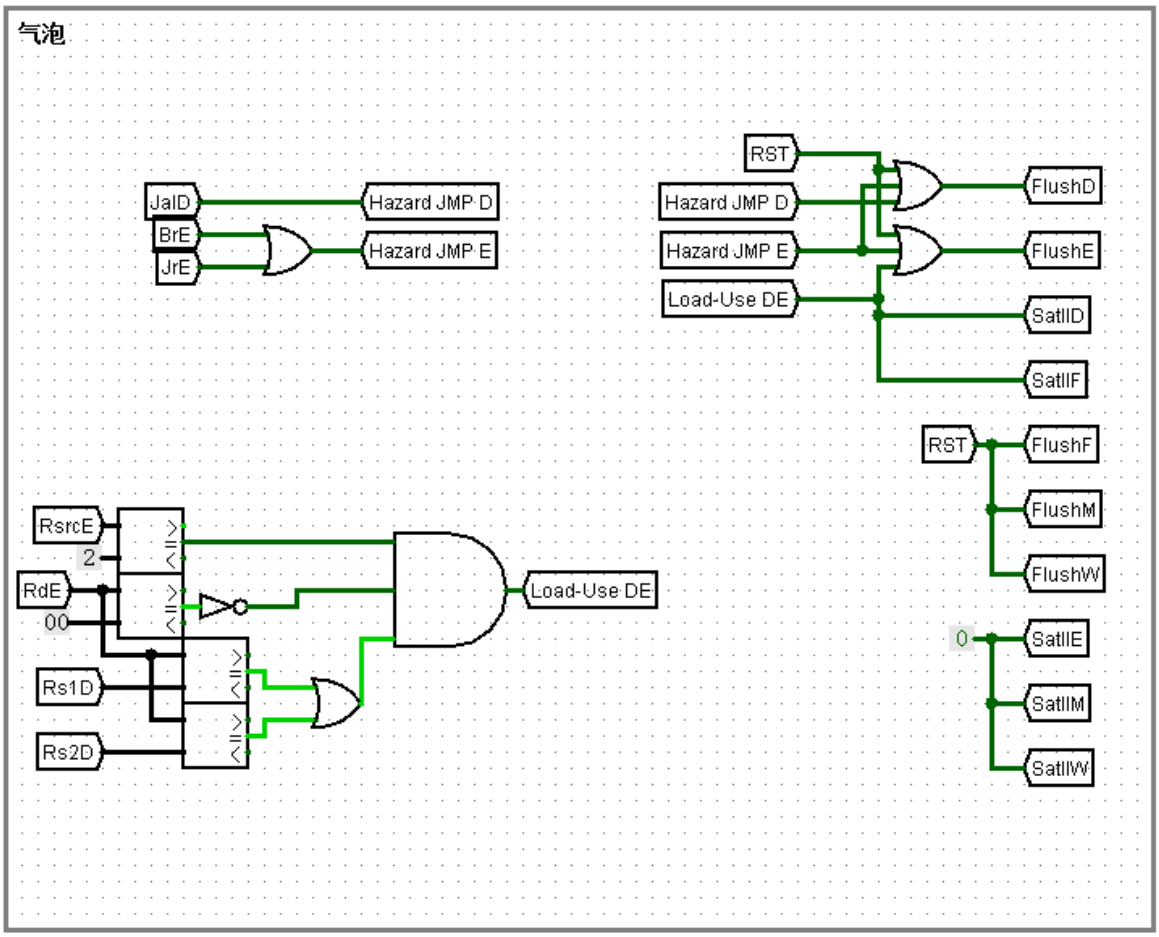
三.流水线CPU

主要完成的模块：

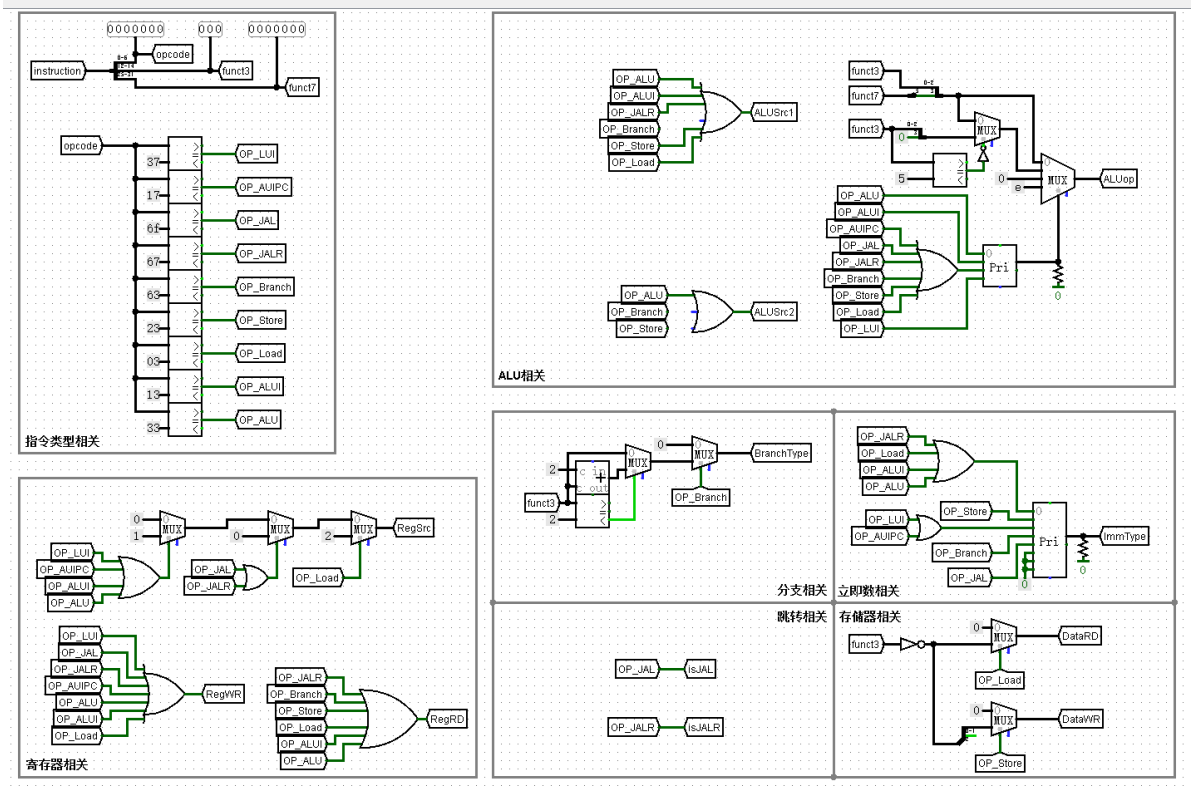
1. 前递模块



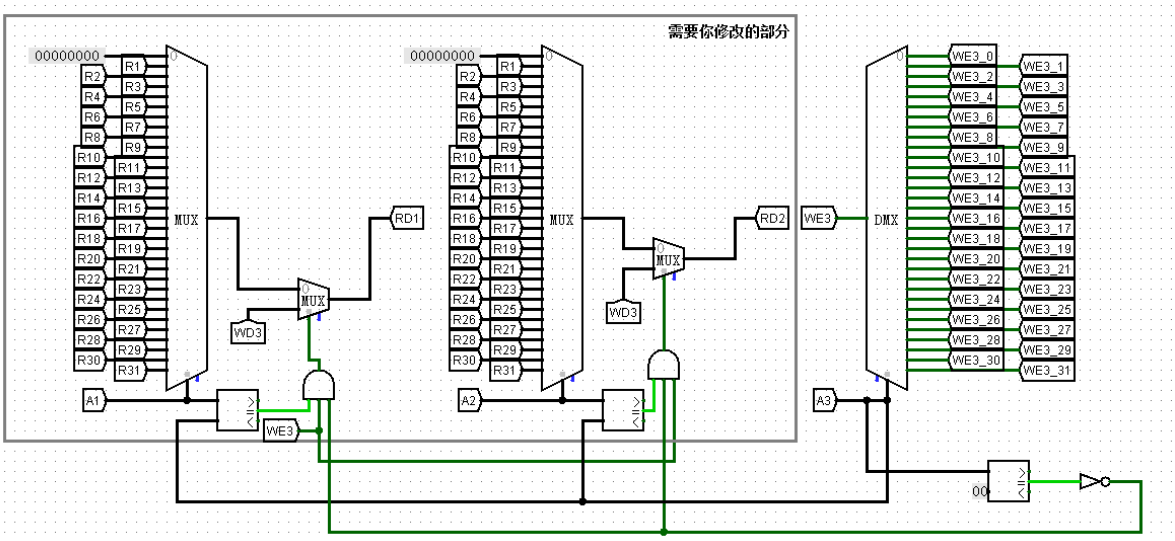
2. load-use和冒险模块



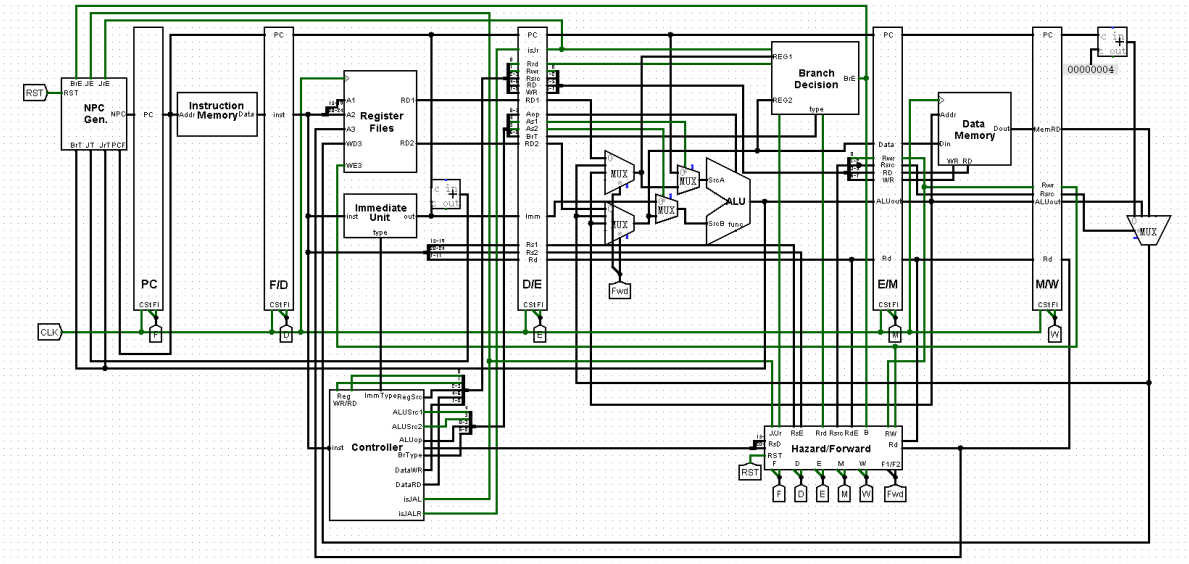
3. 控制模块



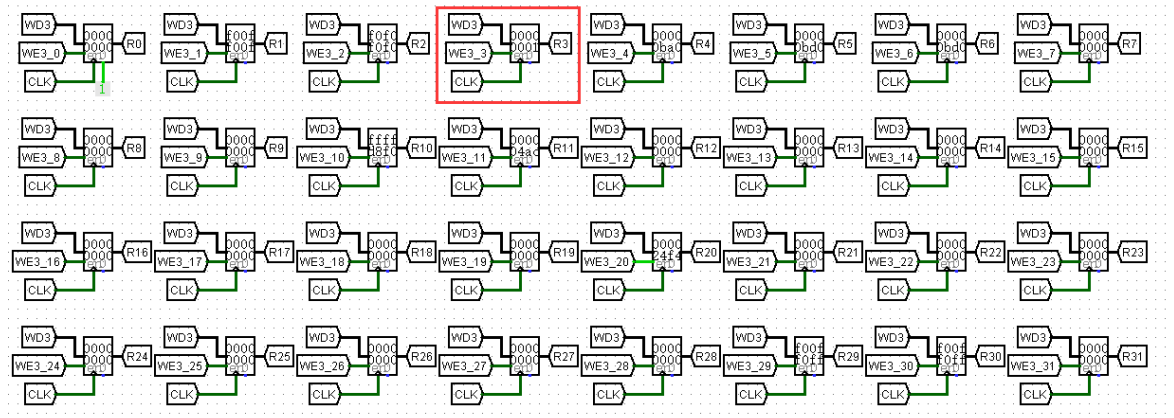
4. 寄存器模块



CPU连线



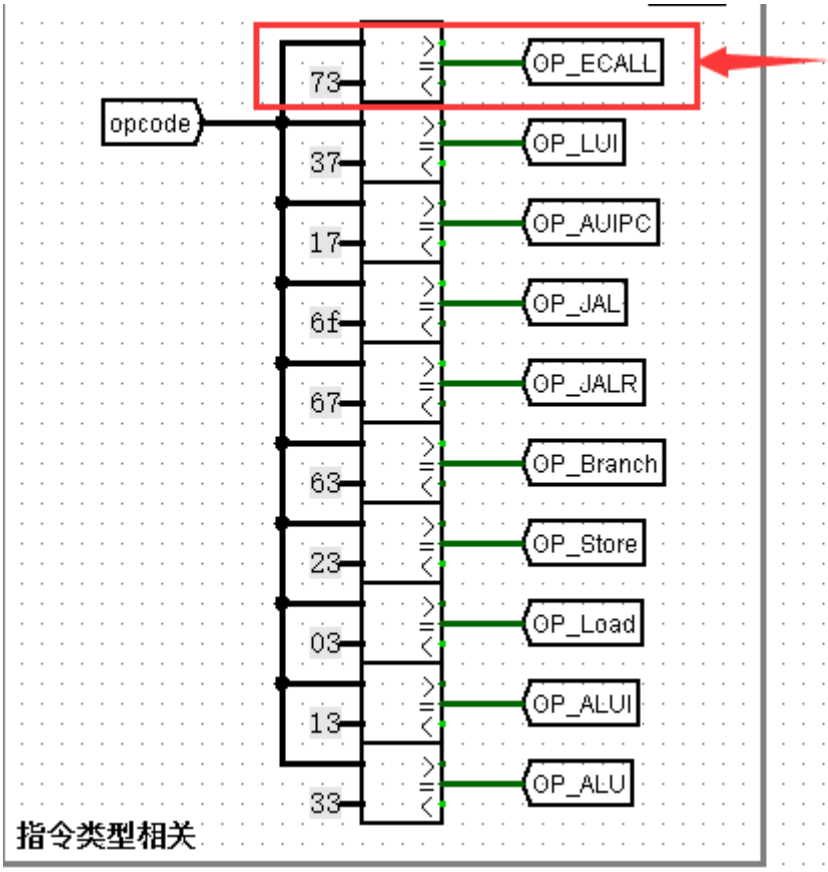
最终运行结果：

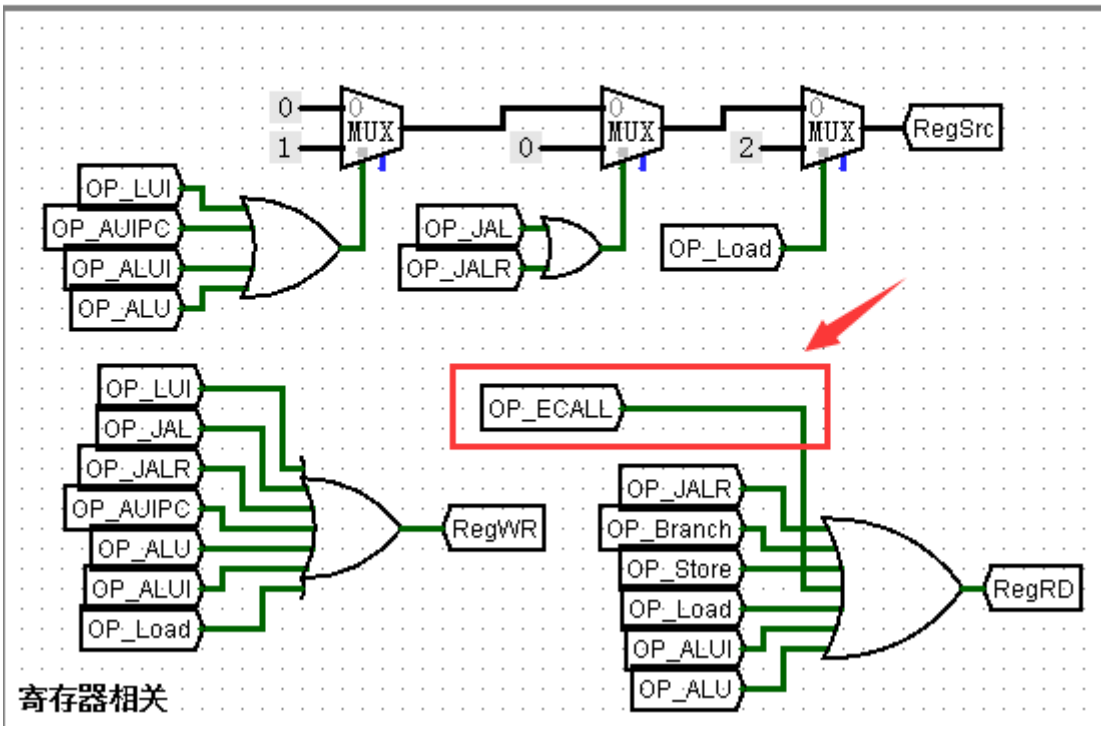


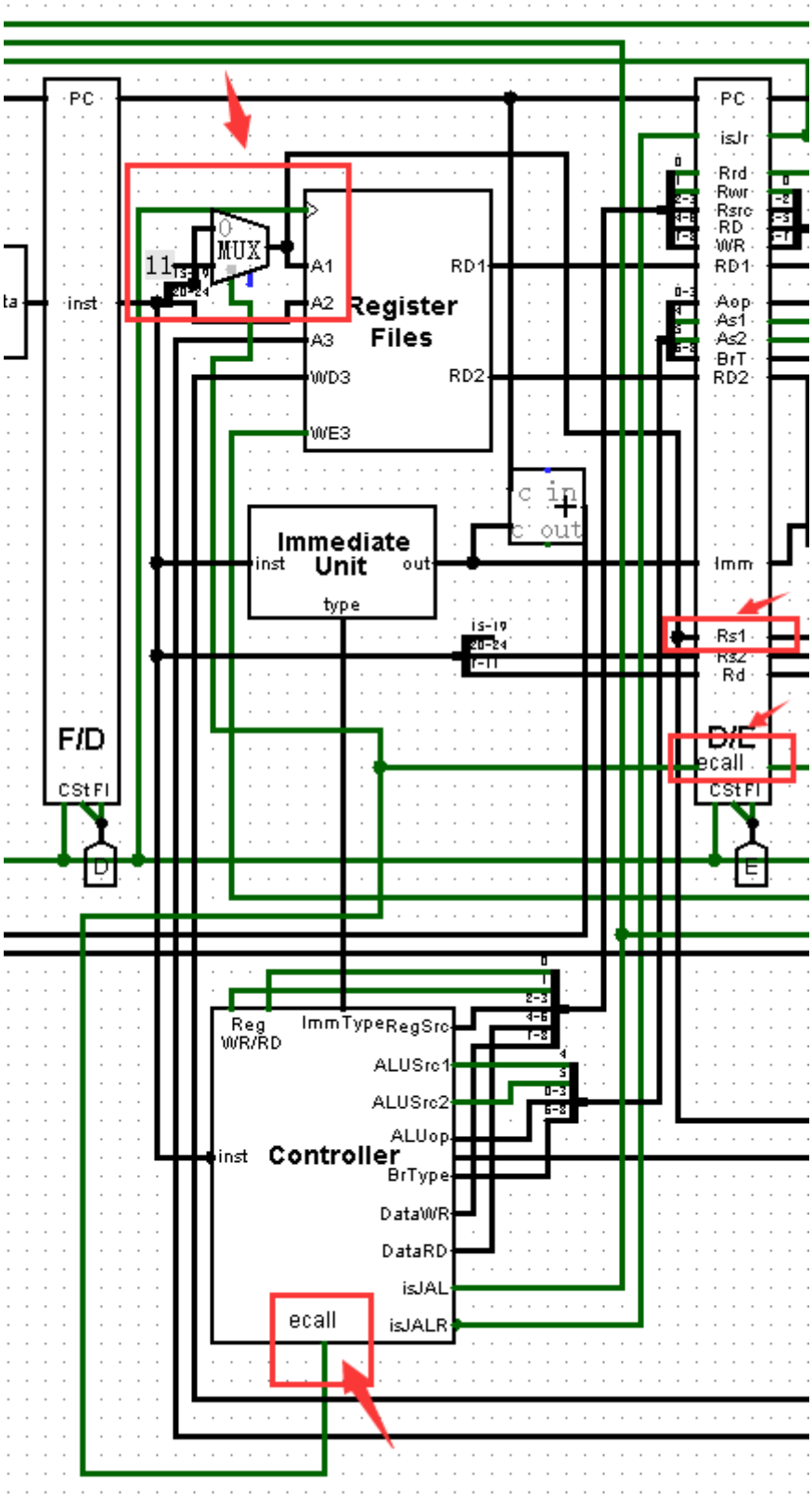
四.能实现ecall指令以及外设输入的流水线CPU（扩展实验）

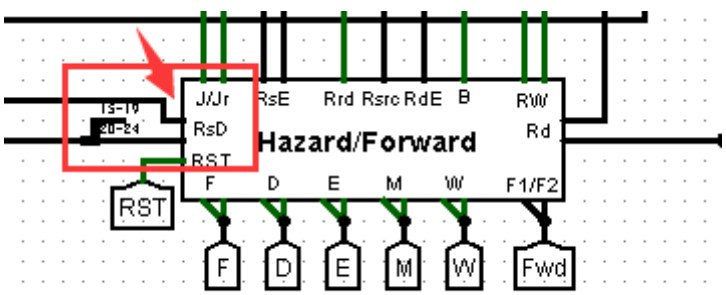
在（三）流水线CPU的基础上，通过修改controller模块，前递冒险模块，向D/E寄存器添加ecall信号，修改A1，RS1的输入值，以及增加输入外设来完成一个能实现ecall指令以及外设输入的流水线CPU.

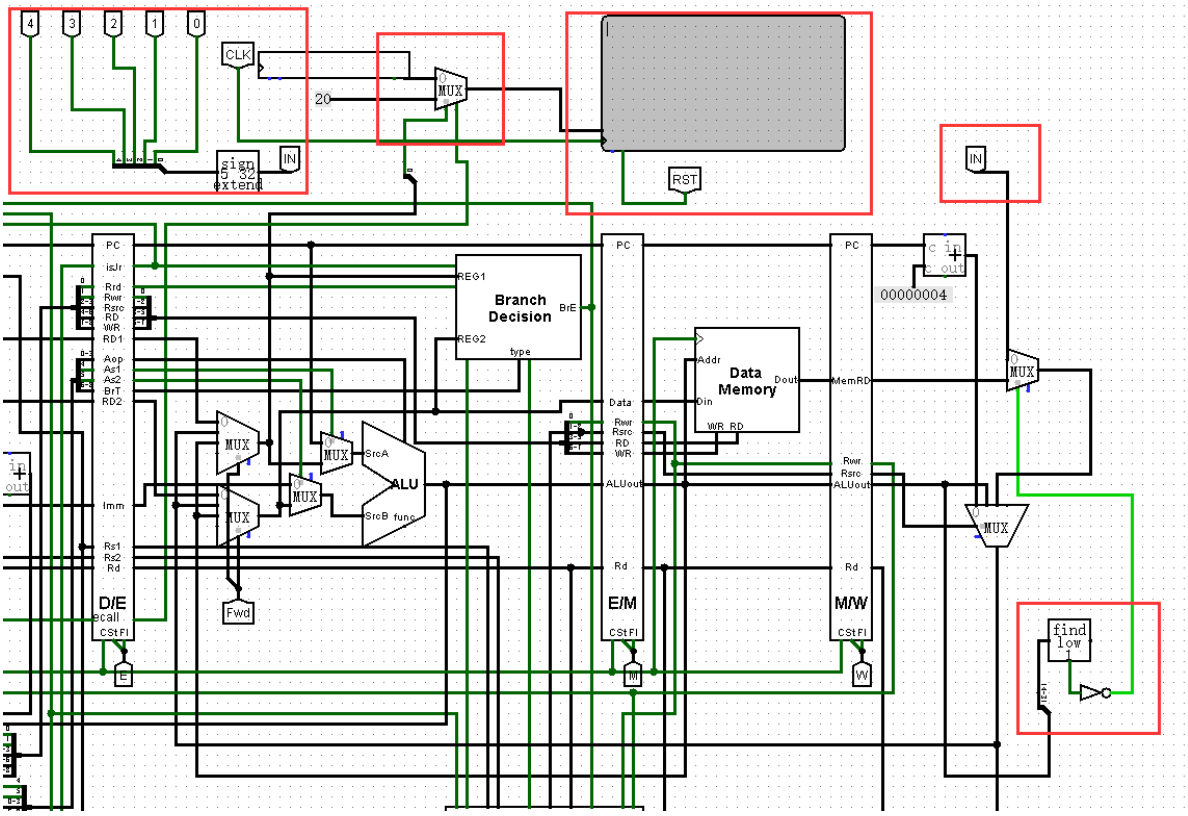
修改如下：






PC
inst
F/D
CStFl
D
MUX
A1
A2
A3
WD3
WE3
RD1
RD2
Register Files
Immediate Unit
inst
type
out
c in
c out
PC
isJr
Rrd
Rwr
Rsrc
RD
WR
RD1
Aop
As1
As2
BrT
RD2
Imm
Rs1
Rs2
Rd
D/E
ecall
CStFl
E
Reg WR/RD
ImmType
RegSrc
ALUSrc1
ALUSrc2
ALUop
inst
Controller
BrType
DataWR
DataRD
isJAL
isJALR
ecall


J/Jr
RsD
RST
RsE
Rrd
Rsrc
RdE
B
RW
Rd
Hazard/Forward
F
D
E
M
W
F1/F2
RST
F
D
E
M
W
Fwd



说明：

1. ecall指令为00000073

   通过寄存器x17的值，ecall指令执行相应功能：

   x17 = 0；输出框中输出键盘（keyboard）输入的值；

   x17 = 1；输出框中输出space（空格）；

   其余情况，输出框中随clk输出？

   由于ecall指令需要用到x17的值，因此该指令同样存在前递情况。

2. 在外设输入端，可以通过5个输入口选择要输入的数。

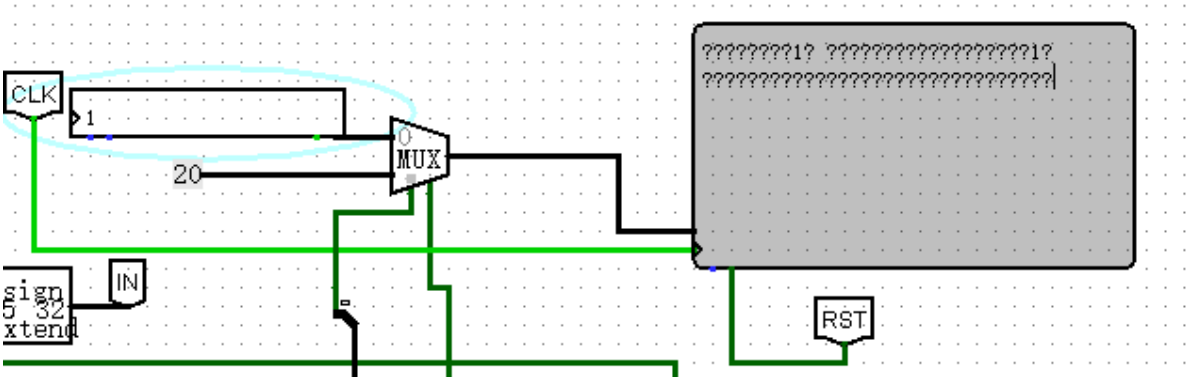
   在汇编代码中，该外设的地址为除data_memory的有效地址外的任意地址。即地址的14~31位含1即可。

3. 测试汇编如下：

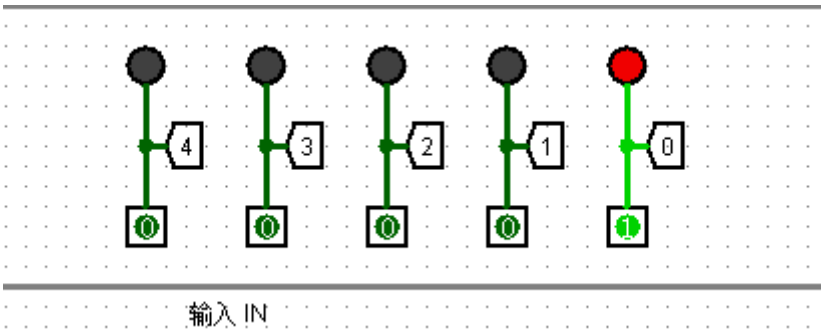
```
start:
#test data hazards
addi x1, x0, 1  #x1=1
addi x2, x1, 1  #x2=2
add x3, x1, x2  #x3=3
add x4, x1, x3  #x4=4
add x5, x1, x4  #x5=5
addi x17, x0, 0
ecall
addi x17, x0, 1
ecall
add x6, x1, x2  #x6=3
add x6, x6, x3  #x6=6
```

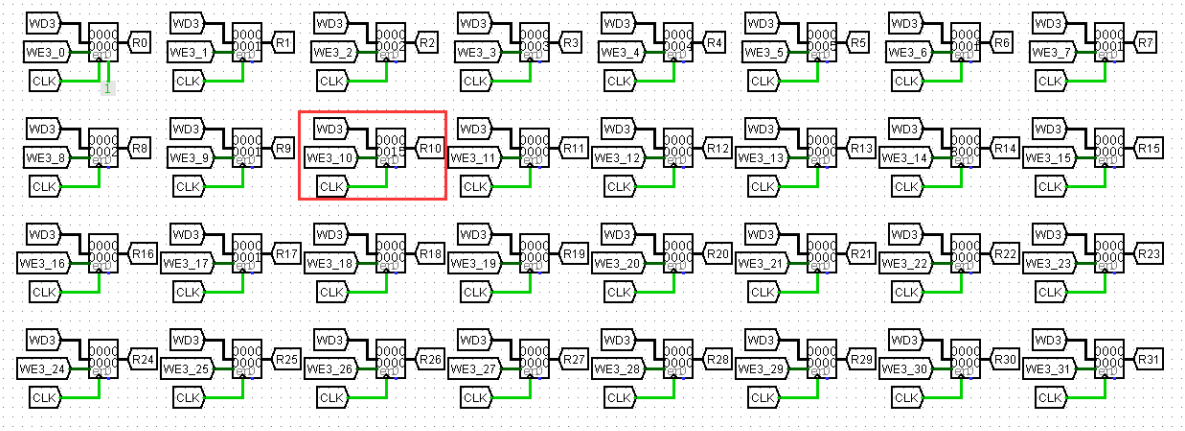
```
add x6, x6, x4  #x6=10
add x6, x6, x5  #x6=15

#test load-use hazard
lui x14 ,8
lw x7, 0x4(x14) #x7=in
addi x8, x7, 1   #x8=in+1
addi x9, x8, -1  #x9=in

#test control hazard
beq x9, x0, start  #if (in==0) start
add x10, x9, x5
add x10, x10, x6
stop: jal x0, stop

#do not execute
add x11, x9, x10
add x12, x10, x11
add x13, x11, x12

```

测试结果：



输入1





思考题

2. 在原本流水线结构中，`Branch` 的优先级比 `Jal` 高。现若在流水线中加入分支预测，`Branch` 指令在 `IF` 阶段就可以跳转（假设预测跳转）。假设现在有一条 `Jal` 指令在 `ID` 阶段，有一条 `Branch` 指令在 `IF` 阶段，此时会导致执行顺序位于前面的 `Jal` 指令跳转被忽略，如何解决这个问题?

答：方法1：将 `Jal` 的跳转也提前到 `IF` 阶段；

方法2：将跳转的优先级设置成越往后流水线越优先，`EX` 跳转优先于 `ID` 跳转优先于 `IF` 跳转。

3. 我们在 `ID` 阶段处理 `Jal` 跳转，在 `EX` 阶段处理 `Jalr` 和 `Br` 跳转以及 `Load-use` 冒险。为什么他们有处理阶段的差别？是否可以将某些处理放在更前的流水段进行？如果可以，应该怎么做，`Forwarding` 模块应该如何修改，效率会不会提升?

答： `Jalr` 与 `Br` 需要从寄存器堆中读数，需要立即数生成的模块的结果，并且 `Jalr` 需要 `ALU` 而 `Br` 需要 `Branch Decision` 模 块，因此 `Jalr` 与 `Br` 在 `EX` 阶段；`Jal` 需要立即数生成的模块的结果，因此在 `ID` 阶段；

可把 `Jal` 提前到 `IF`(方法：利用单独的立即数生成器)，`Jalr` 提前到 `ID`(方法：利用单独的 `rs1` 与立即数加法器)，`Br` 提前到 `ID`（方法：`Forwarding` 检测 `A1`,`A2` 与 `EX`/`Mem`/`WB` 阶段是否相同，并前递到 ID 处以供判断）；

由于减少了停顿，因此效率会提升。

5. 在流水线 CPU 中，当 `ID` 和 `WB` 阶段需要读写同一个寄存器时会存在数据冒险，给出两种解决该数据冒险的方法。

答：方法1：使寄存器堆在时钟下降沿也有效，前半周期写入，后半周期读出；

方法2：将寄存器堆改为写优先；

实验总结

通过本次实验，巩固了对CPU各结构的知识，通过连线增加了对各功能模块的了解与认识，扩展实验也让自己开阔了思路，收获不少。