计算机组成原理实验报告

实验题目：综合实验（Logisim）

学生名称：刘恒远

学生学号：PB20111642

完成日期：2022.5.26

实验目的：

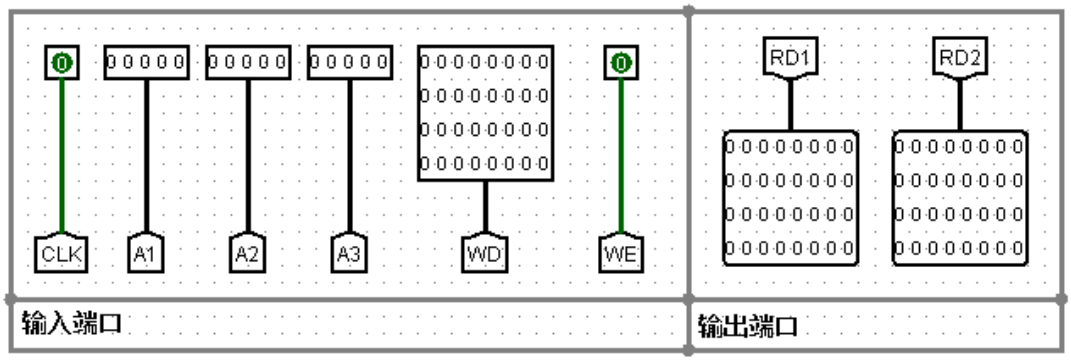
通过自己实际连接CPU的各个模块，加深对CPU的模块、单周期CPU、流水线CPU的理解

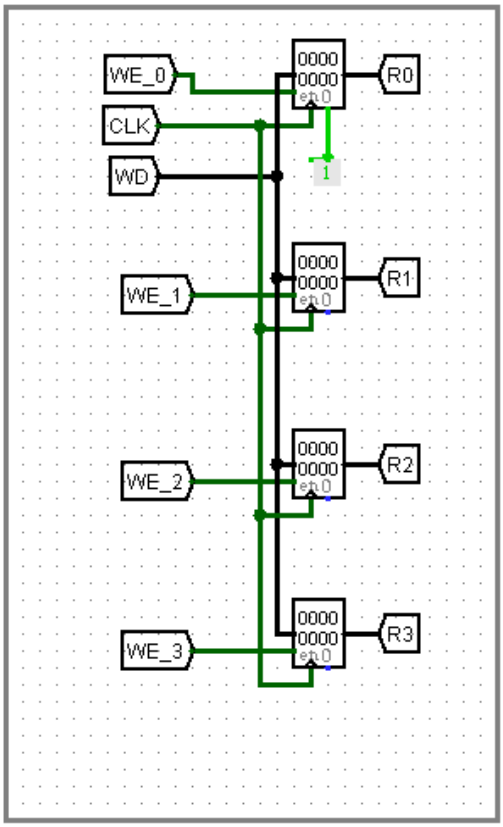
实验环境：

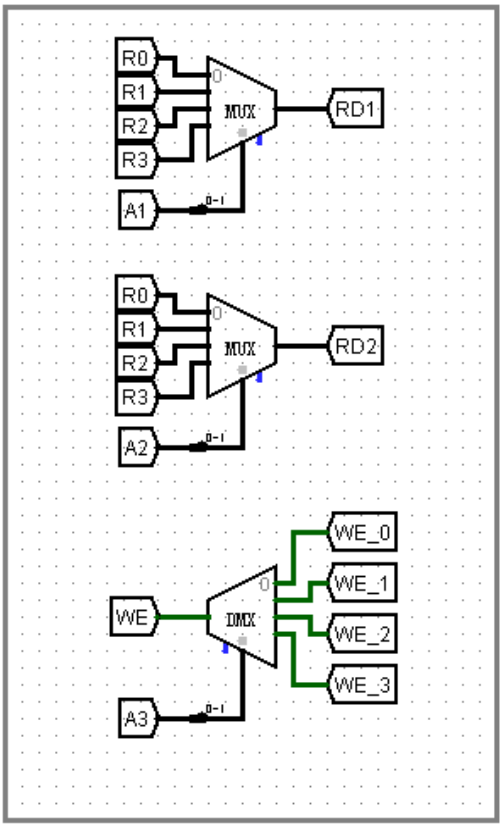
Logisim

实验过程：

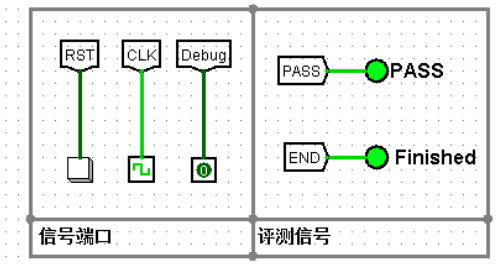
题目一：RegisterFiles

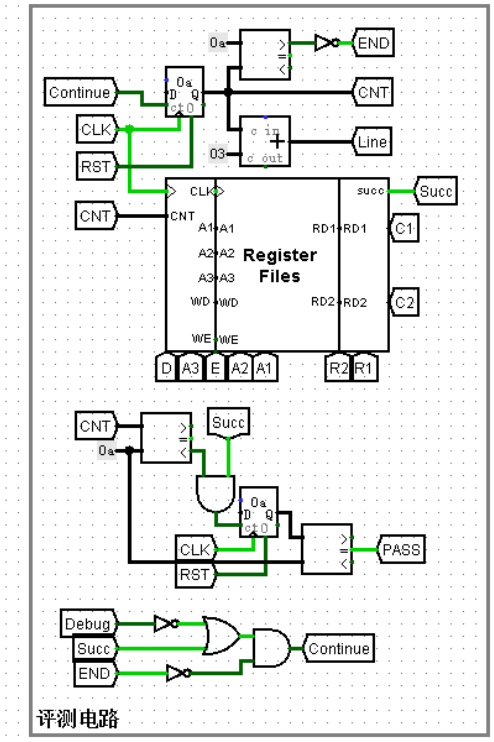

CLK
A1
A2
A3
WD
WE
RD1
RD2
输入端口
输出端口

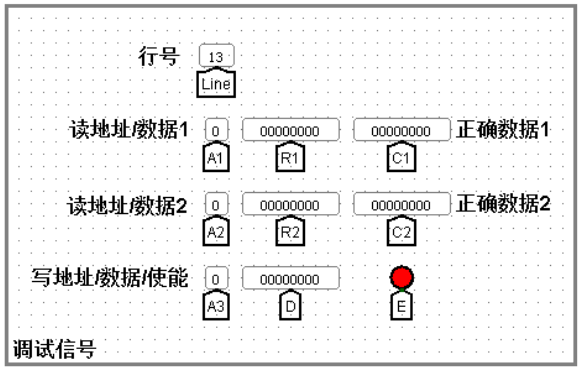

WE_0
CLK
WD
WE_1
WE_2
WE_3
R0
R1
R2
R3


R0
R1
R2
R3
A1
A2
A3
MUX
RD1
RD2
WE
DMX
WE_0
WE_1
WE_2
WE_3

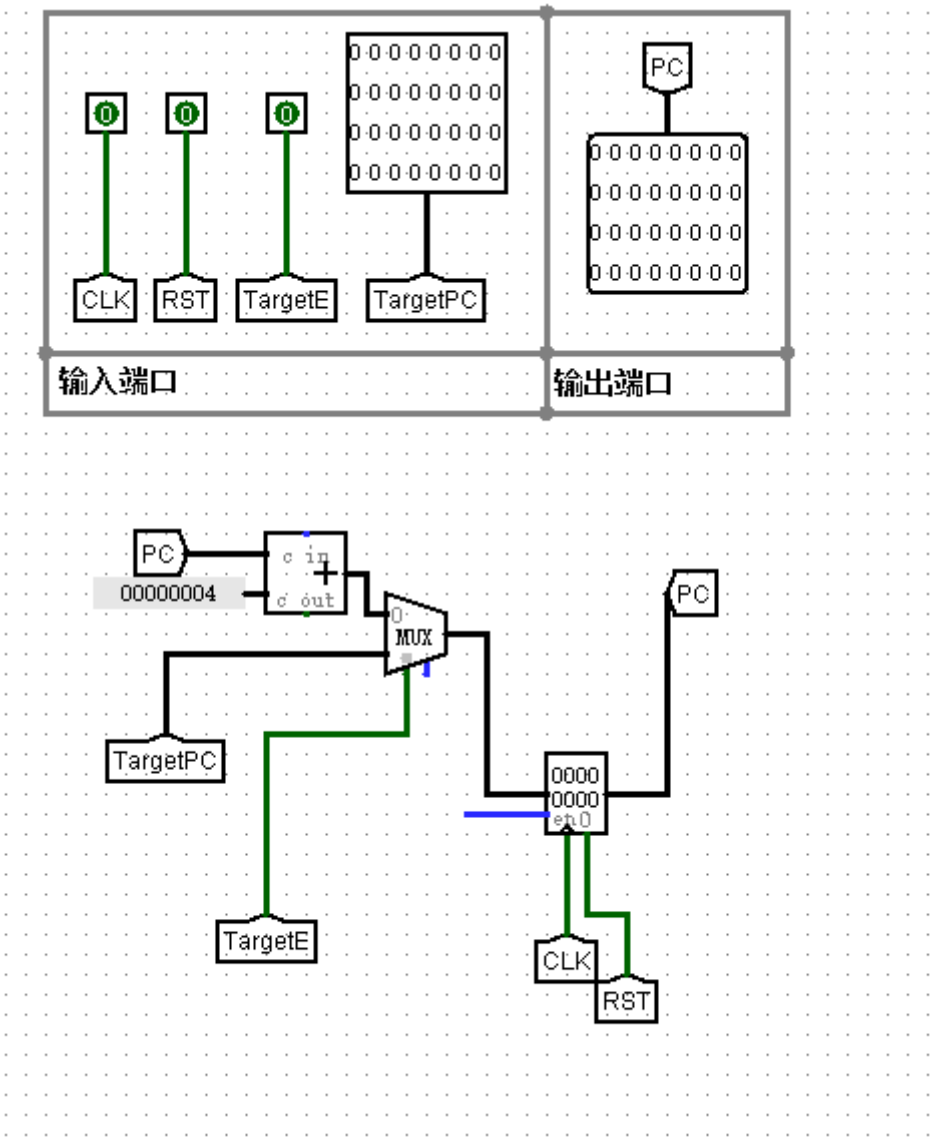
仿真如下：


RST
CLK
Debug
PASS
PASS
END
Finished
信号端口
评测信号

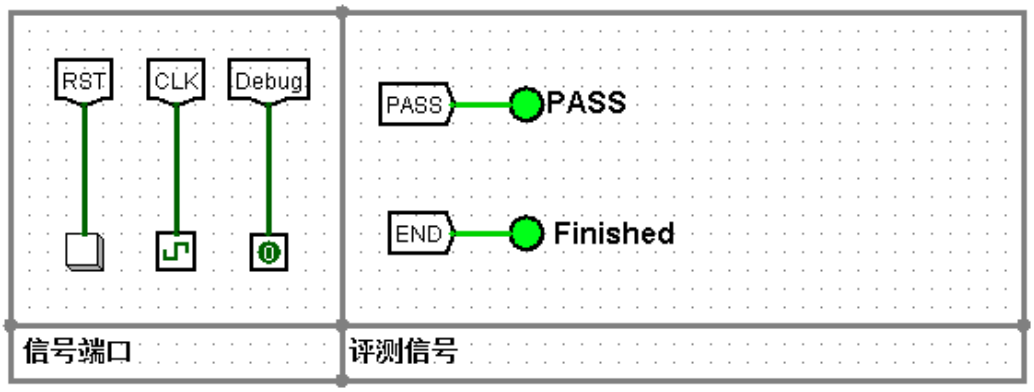

0a
END
Continue
CNT
CLK
RST
03
Line
CNT
CLK
succ
Succ
A1
RD1
C1
Register
Files
A2
A3
WD
RD2
C2
WE
D
A3
E
A2
A1
R2
R1
CNT
0a
Succ
CLK
RST
PASS
Debug
Succ
END
Continue
评测电路

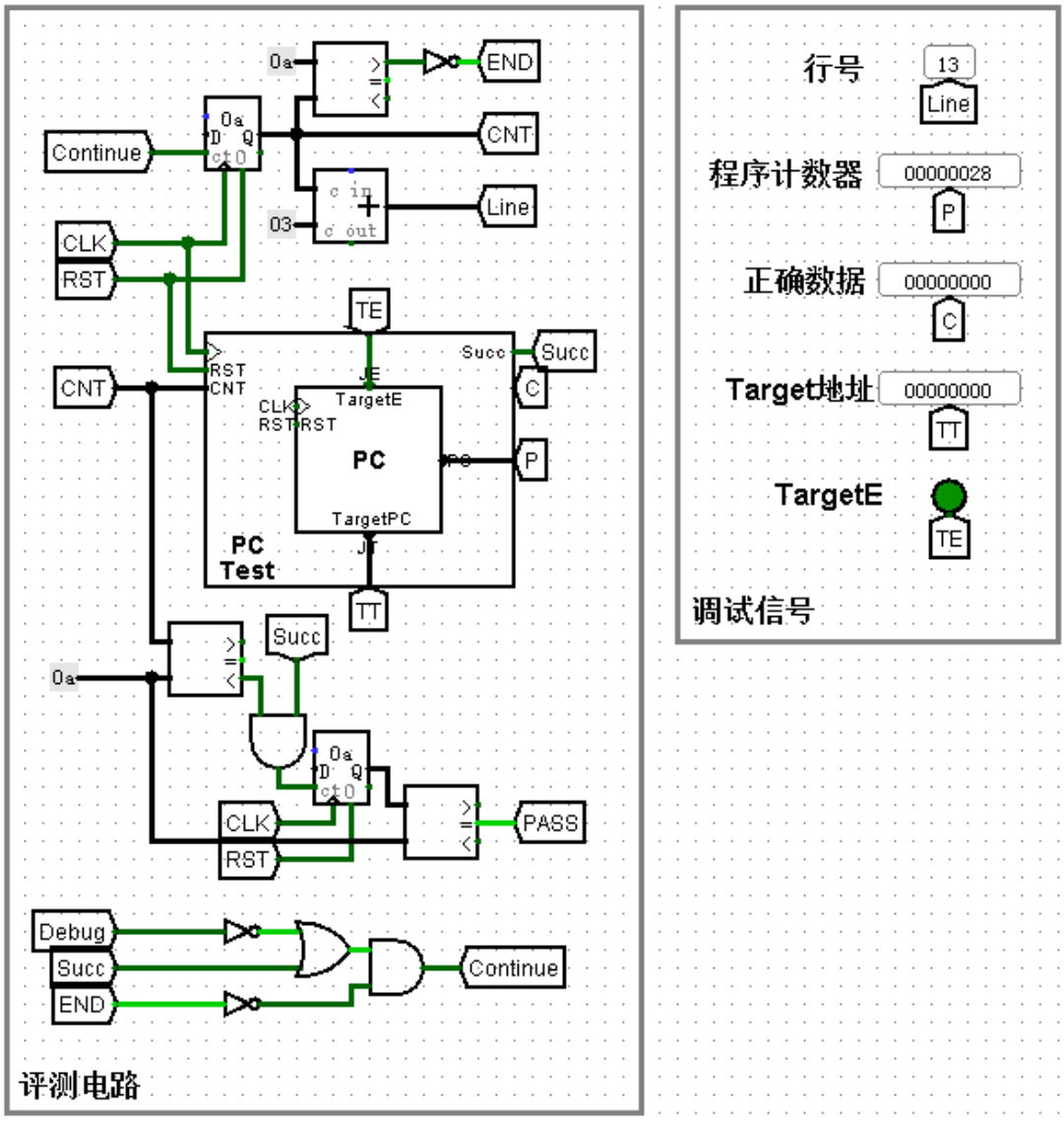

行号
13
Line
读地址/数据1
0
A1
00000000
R1
00000000
C1
正确数据1
读地址/数据2
0
A2
00000000
R2
00000000
C2
正确数据2
写地址/数据/使能
0
A3
00000000
D
E
调试信号

题目二：PC

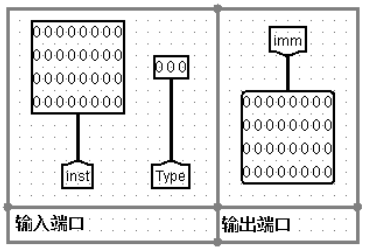

CLK
RST
TargetE
TargetPC
PC
输入端口
输出端口
PC
00000004
c in
c out
0
MUX
PC
TargetPC
0000
0000
en 0
TargetE
CLK
RST

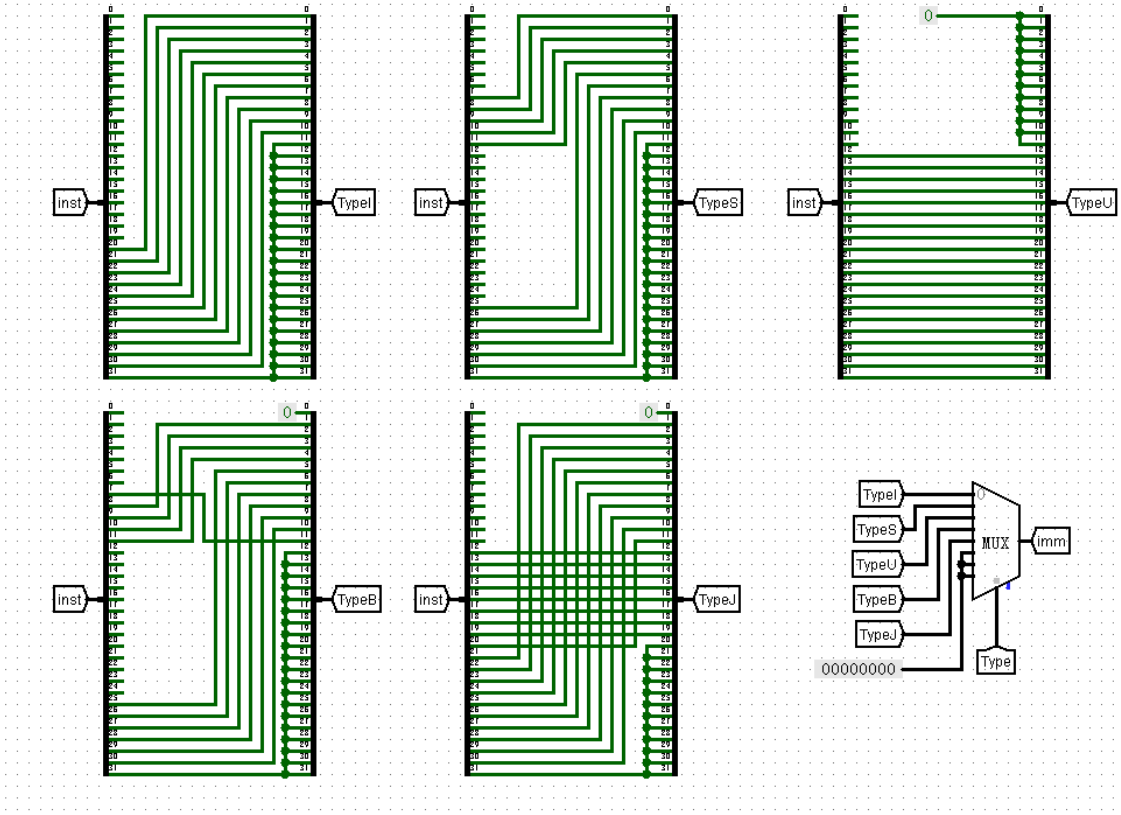
仿真如下：



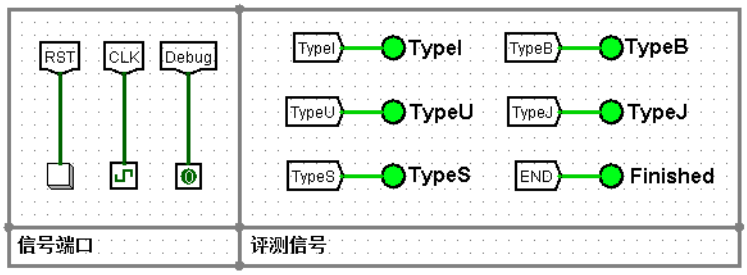


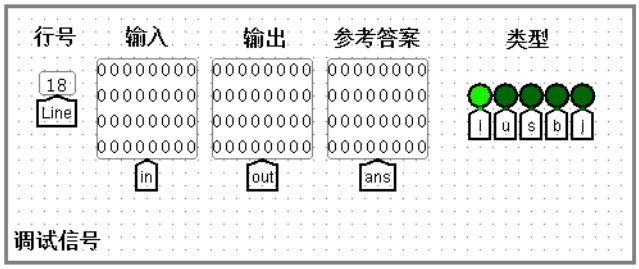
题目三：Immediate

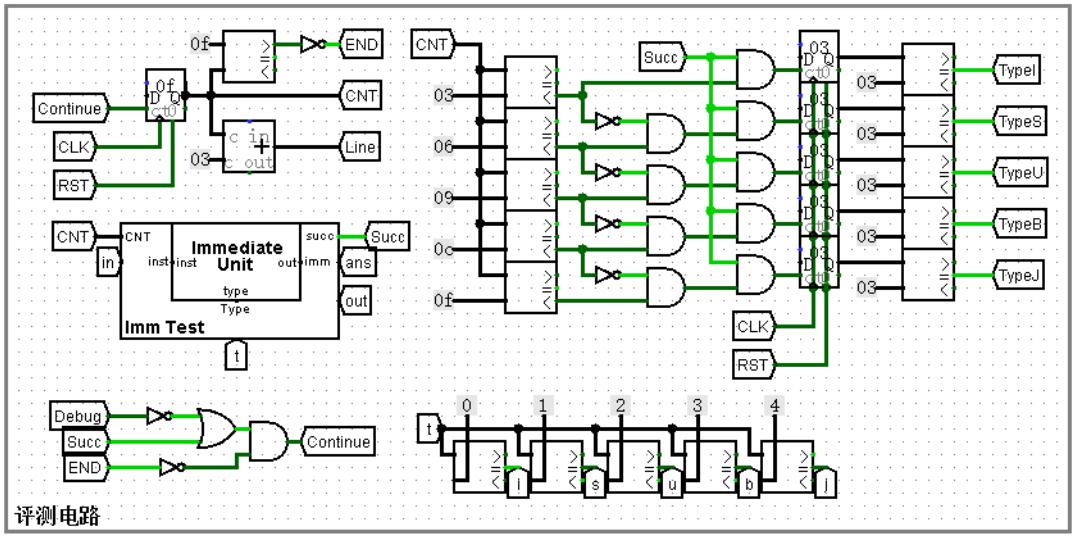

00000000
00000000
00000000
00000000
inst
000
Type
imm
00000000
00000000
00000000
00000000
输入端口
输出端口


inst
TypeI
inst
TypeS
0
inst
TypeU
0
inst
TypeB
0
inst
TypeJ
TypeI
TypeS
TypeU
TypeB
TypeJ
00000000
MUX
imm
Type

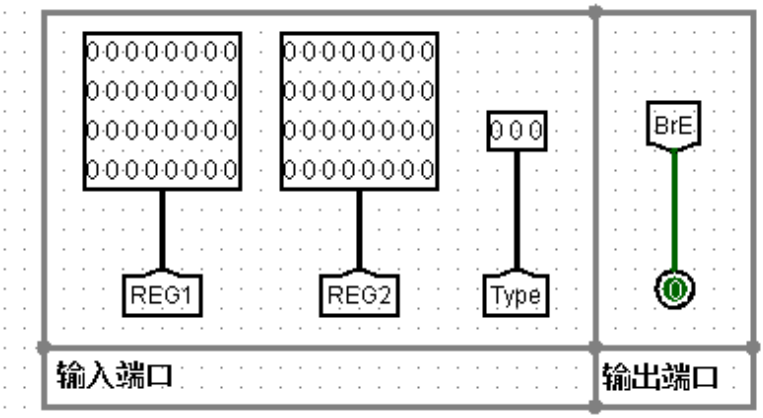
仿真如下：

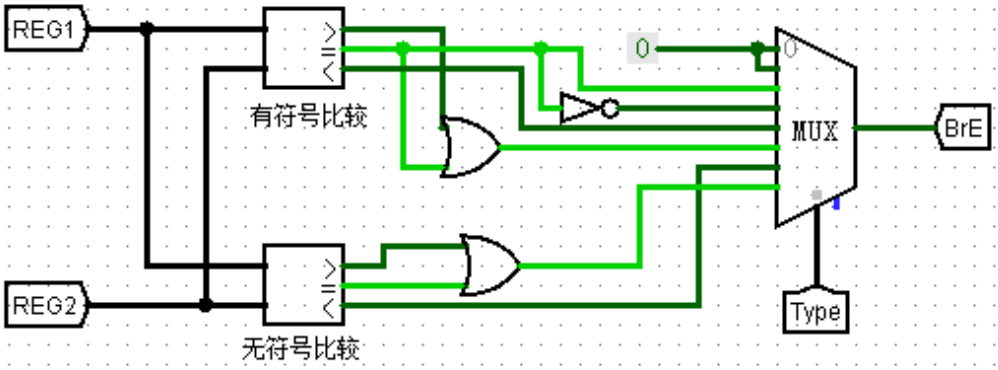




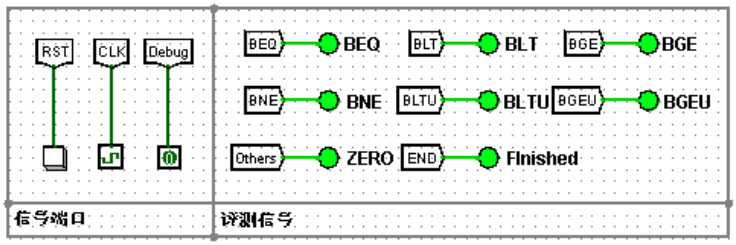


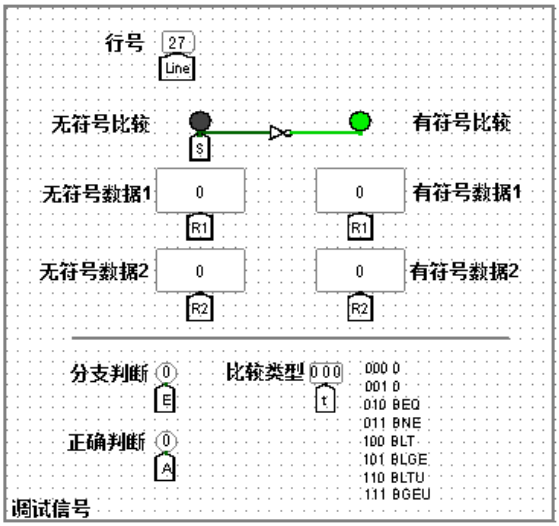
题目四：Branch

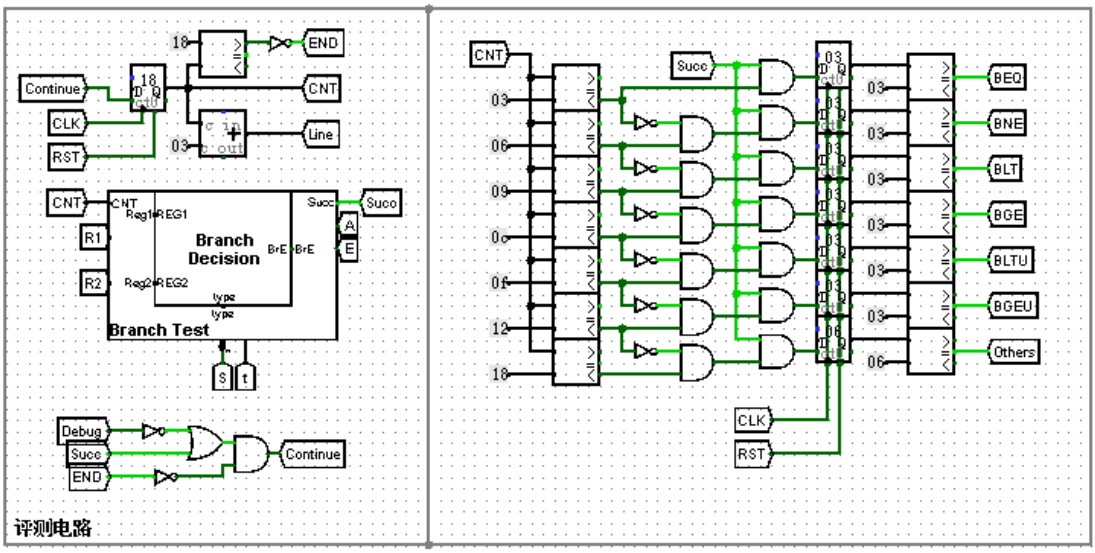

00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
000
BrE
0
REG1
REG2
Type
输入端口
输出端口


REG1
REG2
有符号比较
无符号比较
0
0
MUX
BrE
Type

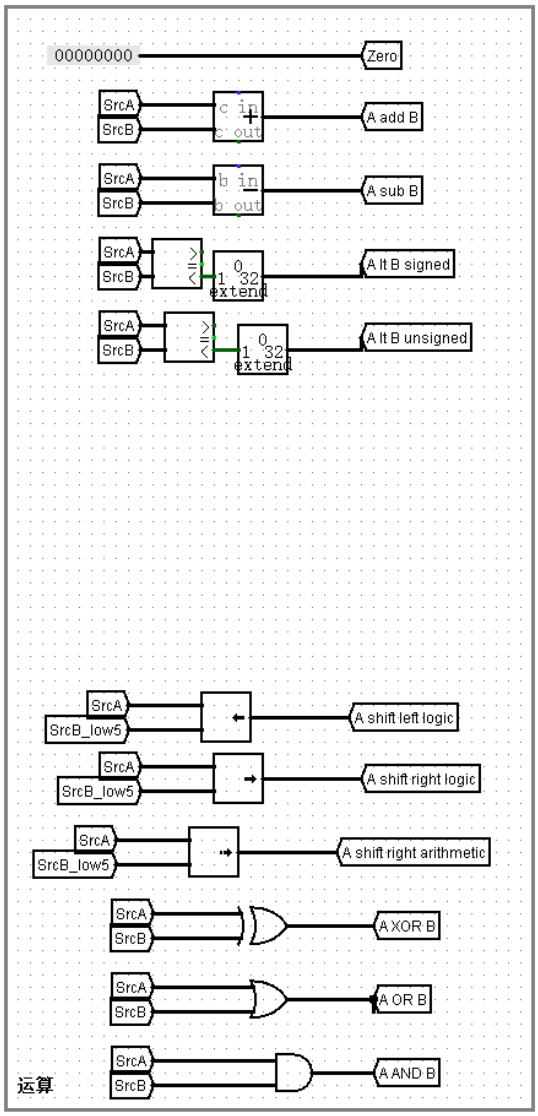
仿真如下：

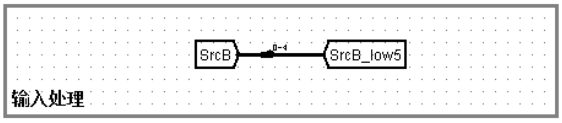

RST
CLK
Debug
BEQ
BEQ
BLT
BLT
BGE
BGE
BNE
BNE
BLTU
BLTU
BGEU
BGEU
Others
ZERO
END
FInished
信号端口
评测信号

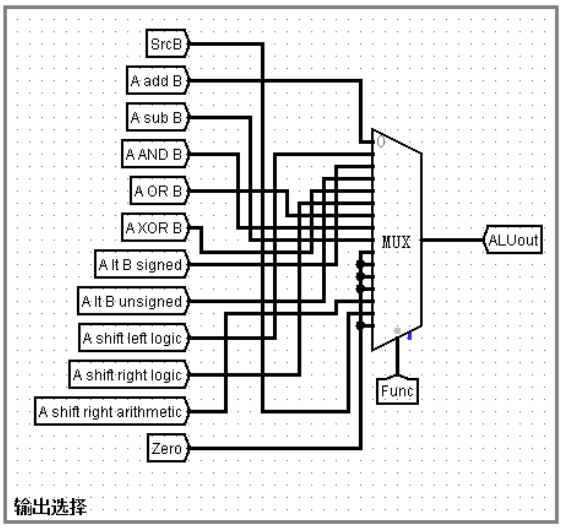

行号
27
Line
无符号比较
S
有符号比较
无符号数据1
0
R1
0
R1
有符号数据1
无符号数据2
0
R2
0
R2
有符号数据2
分支判断
0
E
比较类型
000
t
000 0
001 0
010 BEQ
011 BNE
100 BLT
101 BLGE
110 BLTU
111 BGEU
正确判断
0
A
调试信号


18
END
Continue
CNT
CLK
Line
RST
03
CNT
Reg1
REG1
Branch
Decision
Reg2
REG2
type
Branch Test
Succ
Succ
A
E
R1
R2
BrE
BrE
S
t
Debug
Succ
END
Continue
评测电路
CNT
03
06
09
0c
0f
12
18
Succ
CLK
RST
BEQ
BNE
BLT
BGE
BLTU
BGEU
Others

题目五：ALU

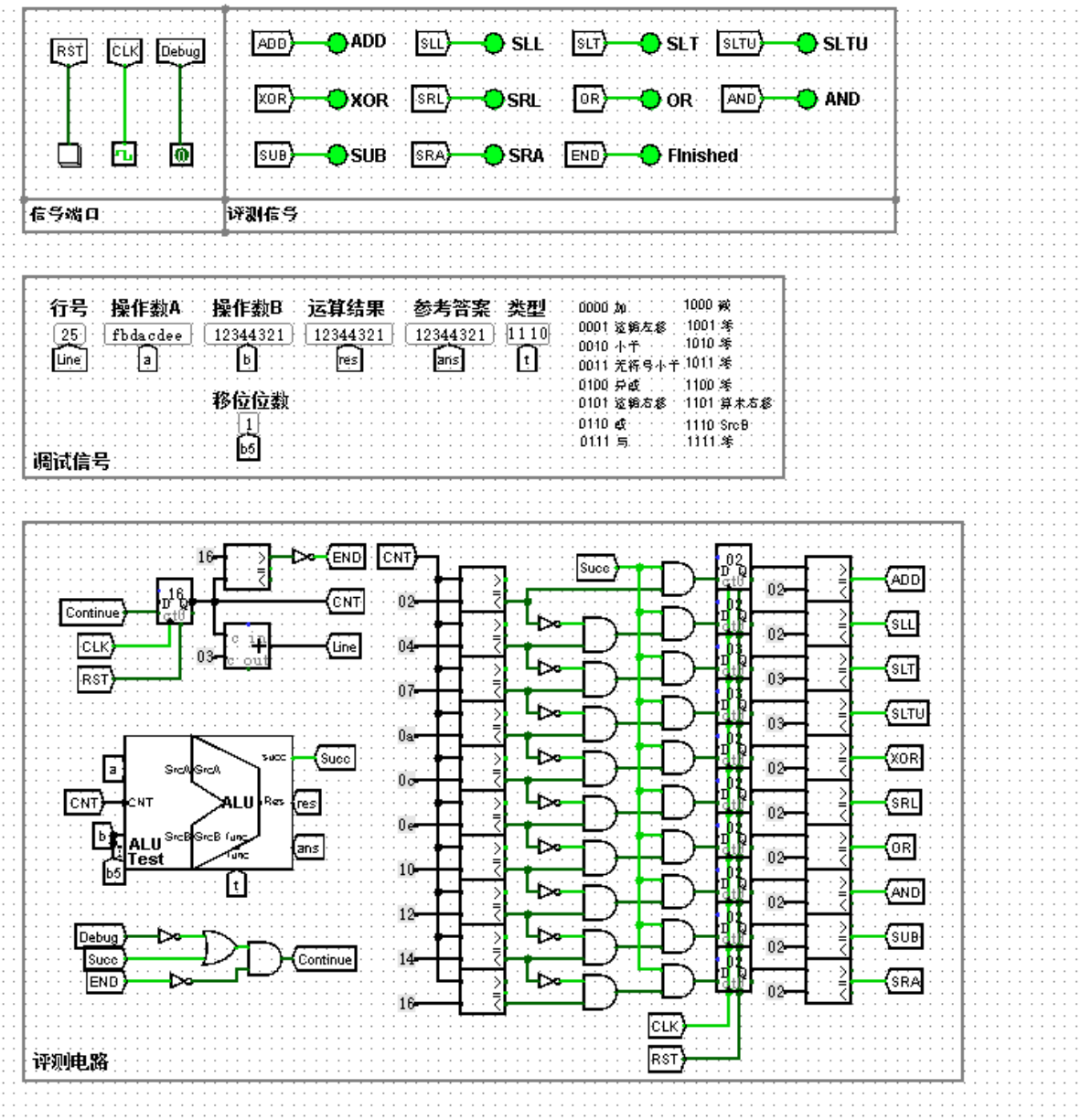






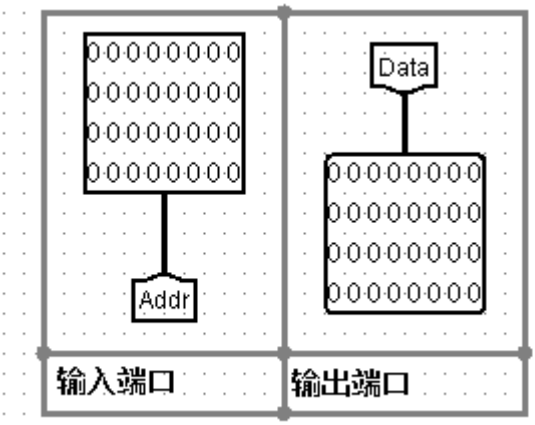
0000 加	0100 异或	1000 减	1100 零
0001 逻辑左移	0101 逻辑右移	1001 零	1101 算术右移
0010 小于	0110 或	1010 零	1110 SrcB
0011 无符号小于	0111 与	1011 零	1111 零

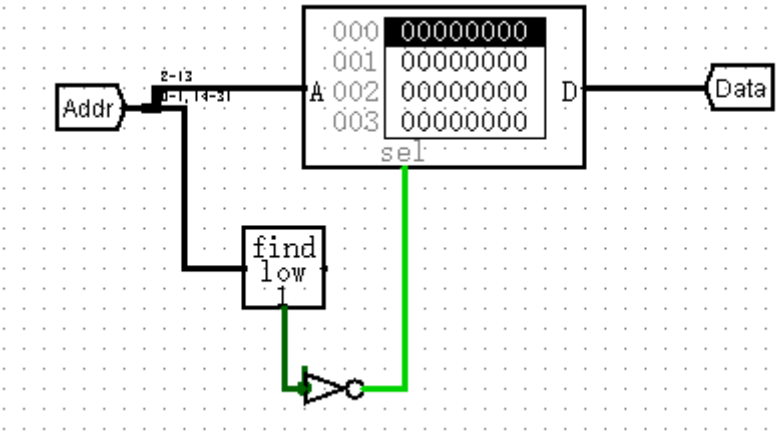
仿真如下：



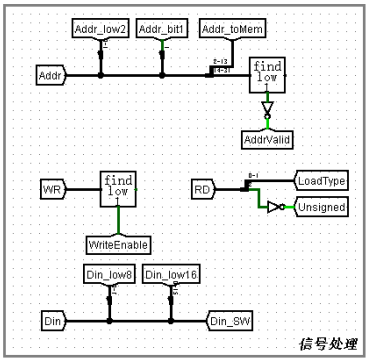
题目六：Memory

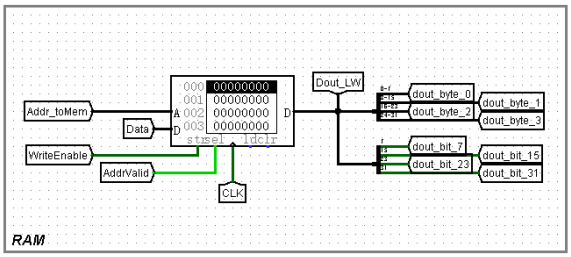
指令寄存器：

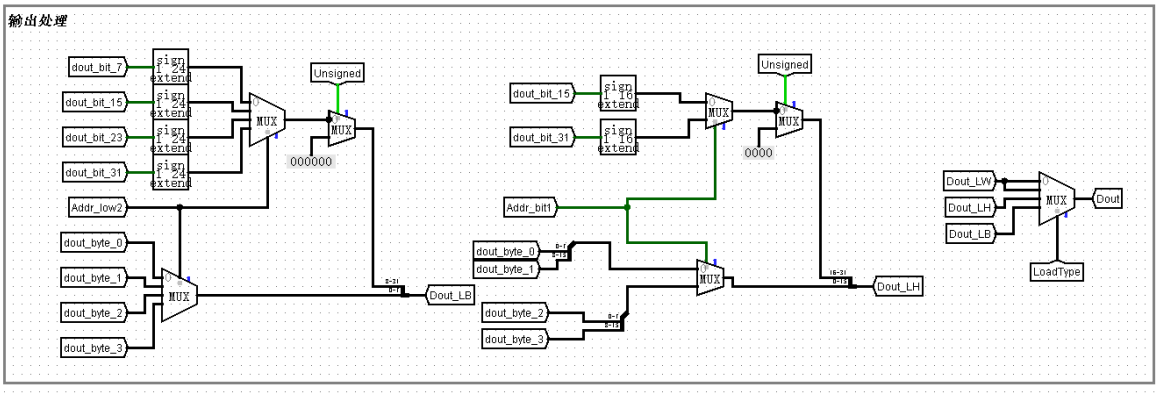

Data
Addr
输入端口
输出端口

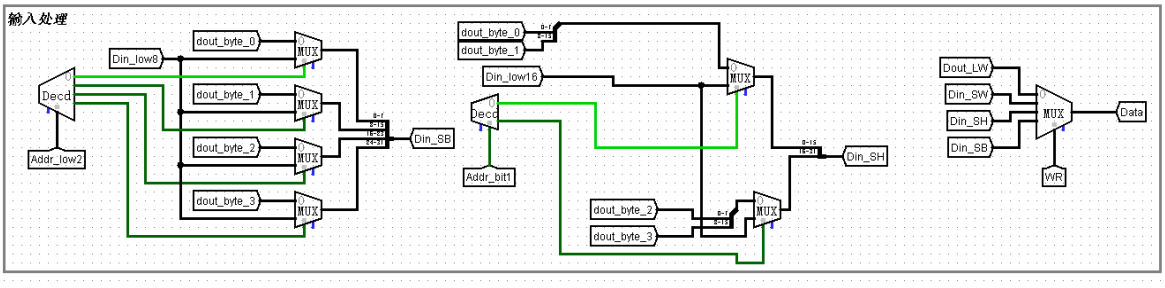

000 00000000
001 00000000
002 00000000
003 00000000
A
D
sel
Addr
Data
find low

数据寄存器：

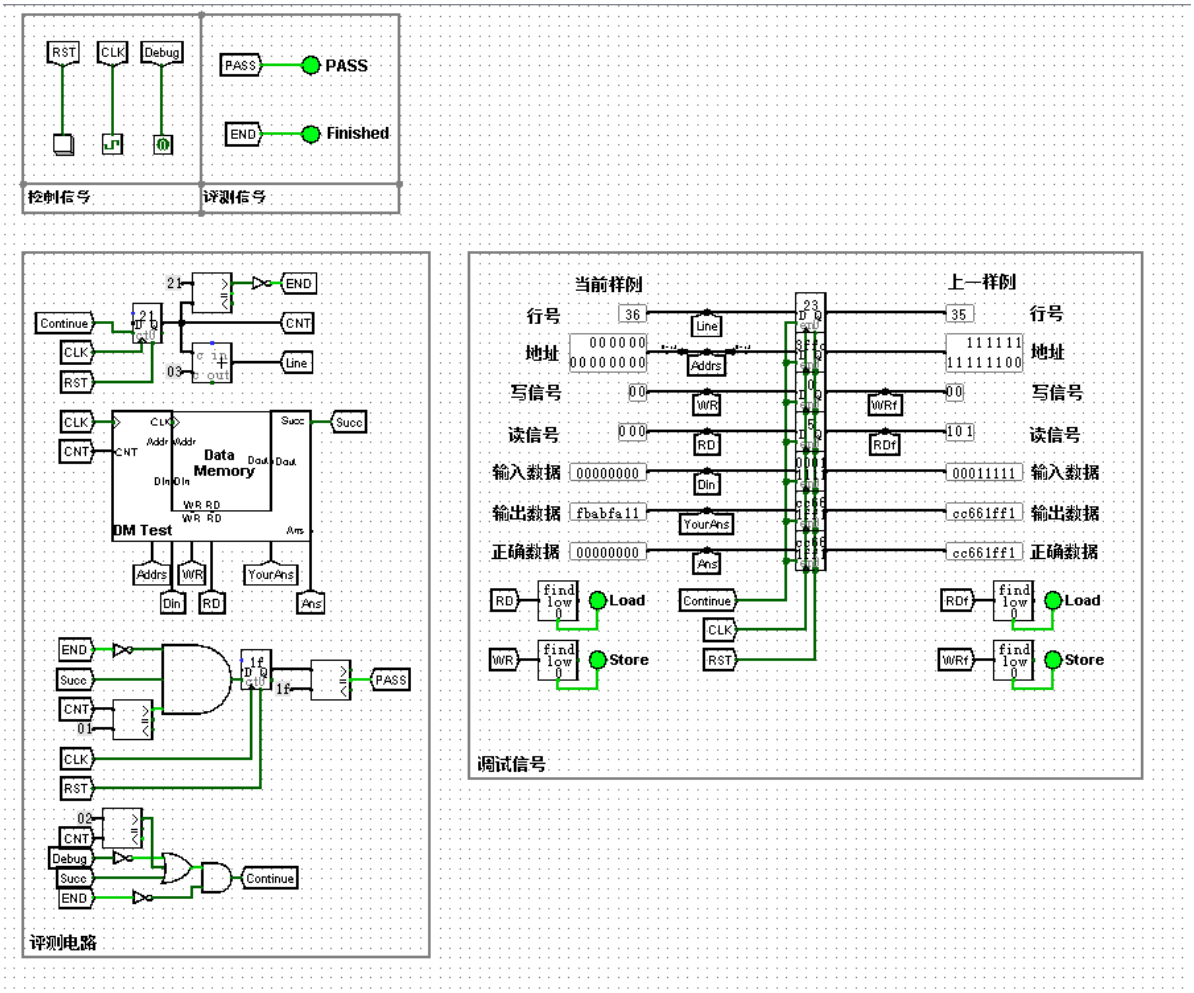

Addr_low2
Addr_bit1
Addr_toMem
Addr
find low
AddrValid
WR
RD
LoadType
Unsigned
WriteEnable
Din_low8
Din_low16
Din
Din_SW
信号处理


Addr_toMem
Data
WriteEnable
AddrValid
CLK
Dout_LW
dout_byte_0
dout_byte_1
dout_byte_2
dout_byte_3
dout_bit_7
dout_bit_15
dout_bit_23
dout_bit_31
RAM

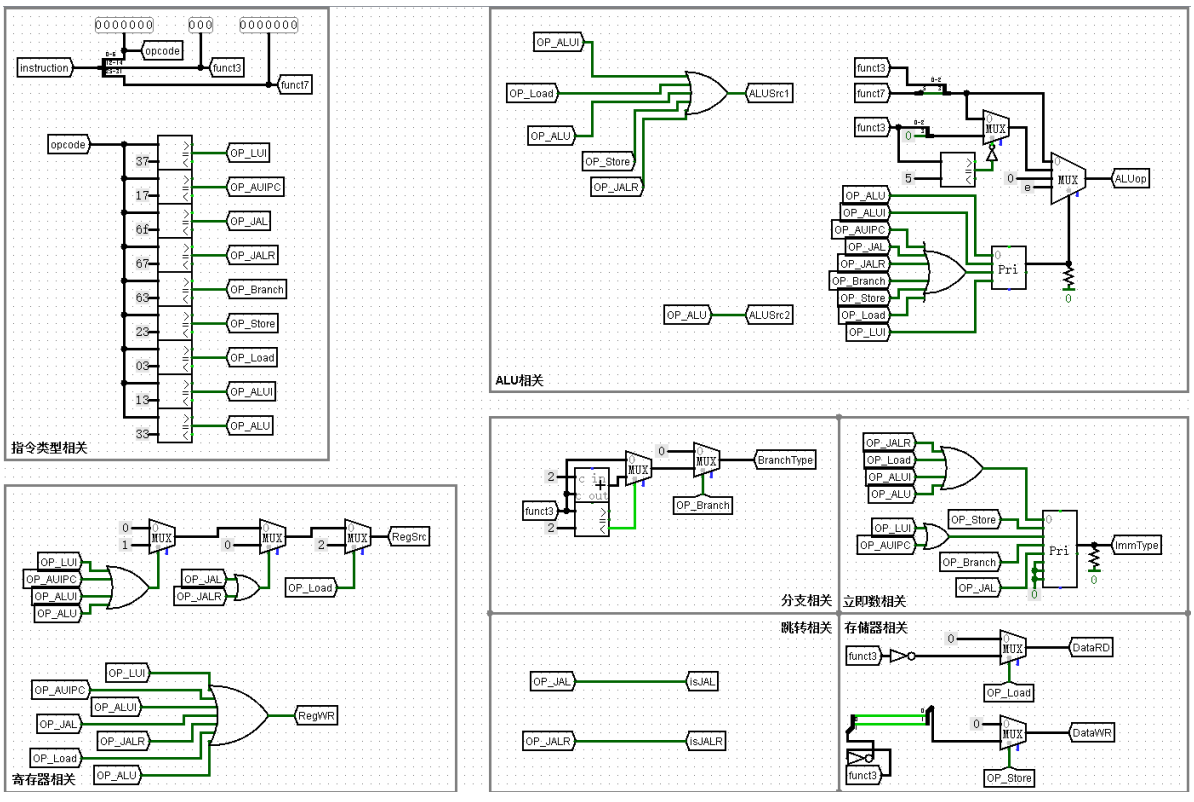

输出处理
dout_bit_7
dout_bit_15
dout_bit_23
dout_bit_31
Unsigned
000000
Addr_low2
dout_byte_0
dout_byte_1
dout_byte_2
dout_byte_3
Dout_LB
Addr_bit1
0000
Dout_LW
Dout_LH
Dout_LB
Dout
LoadType


输入处理
Decd
Addr_low2
Din_low8
dout_byte_0
dout_byte_1
dout_byte_2
dout_byte_3
Din_SB
Din_low16
Addr_bit1
Din_SH
Dout_LW
Din_SW
Din_SH
Din_SB
Data
WR

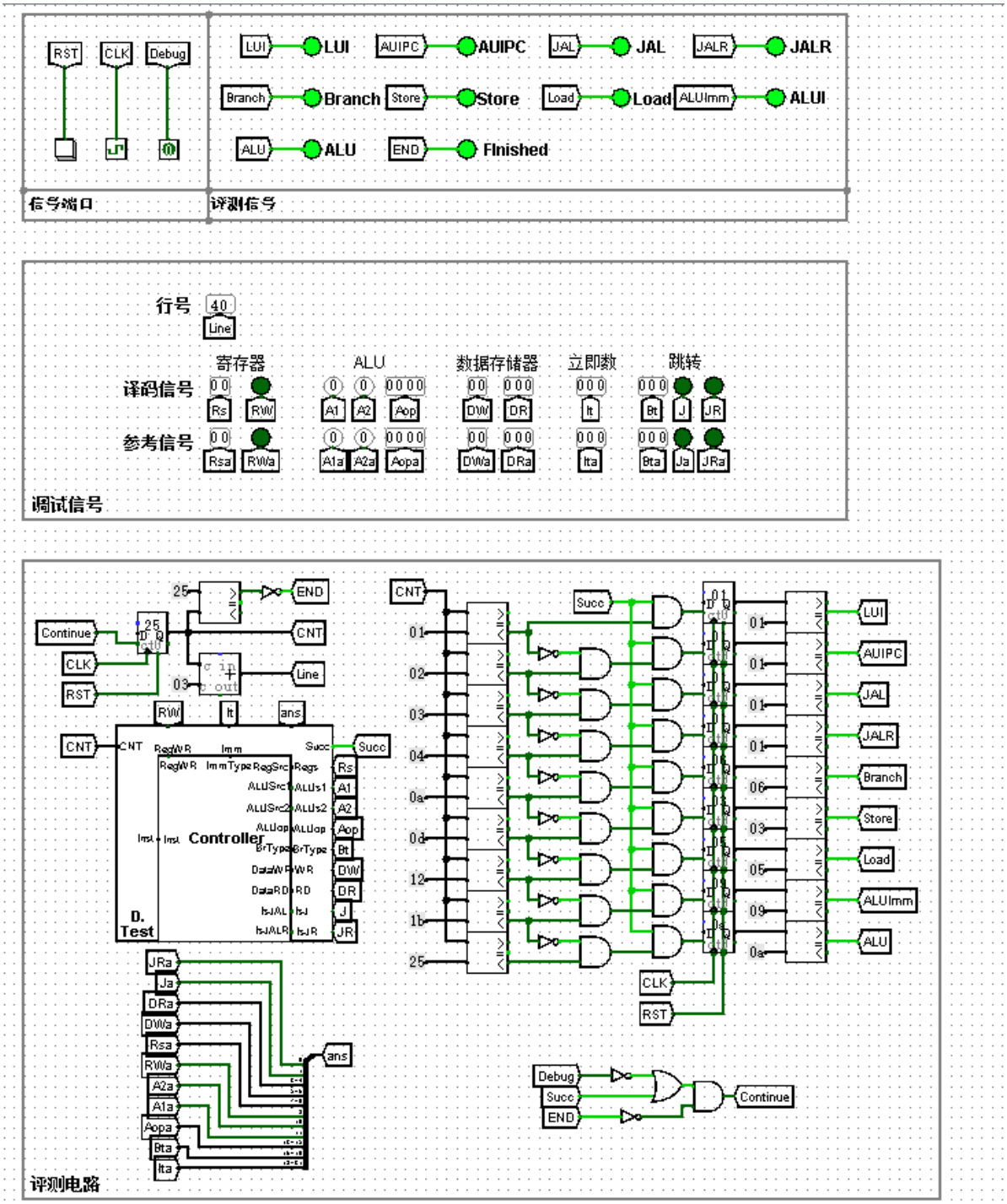
仿真如下：



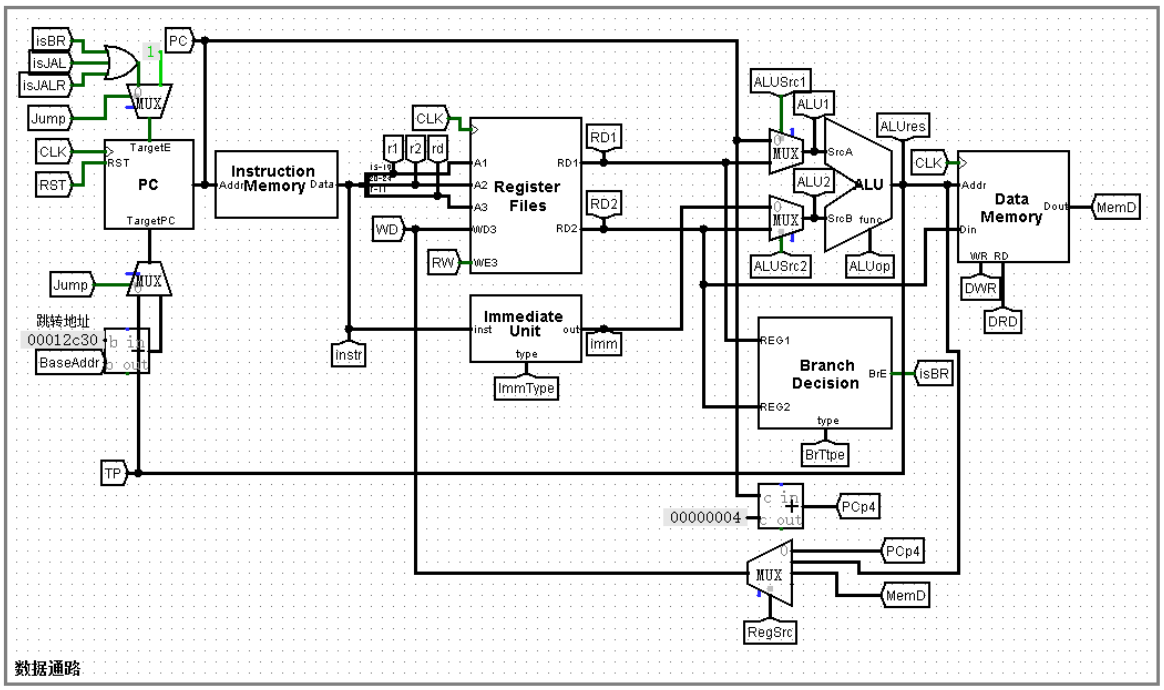
实验七：Controller



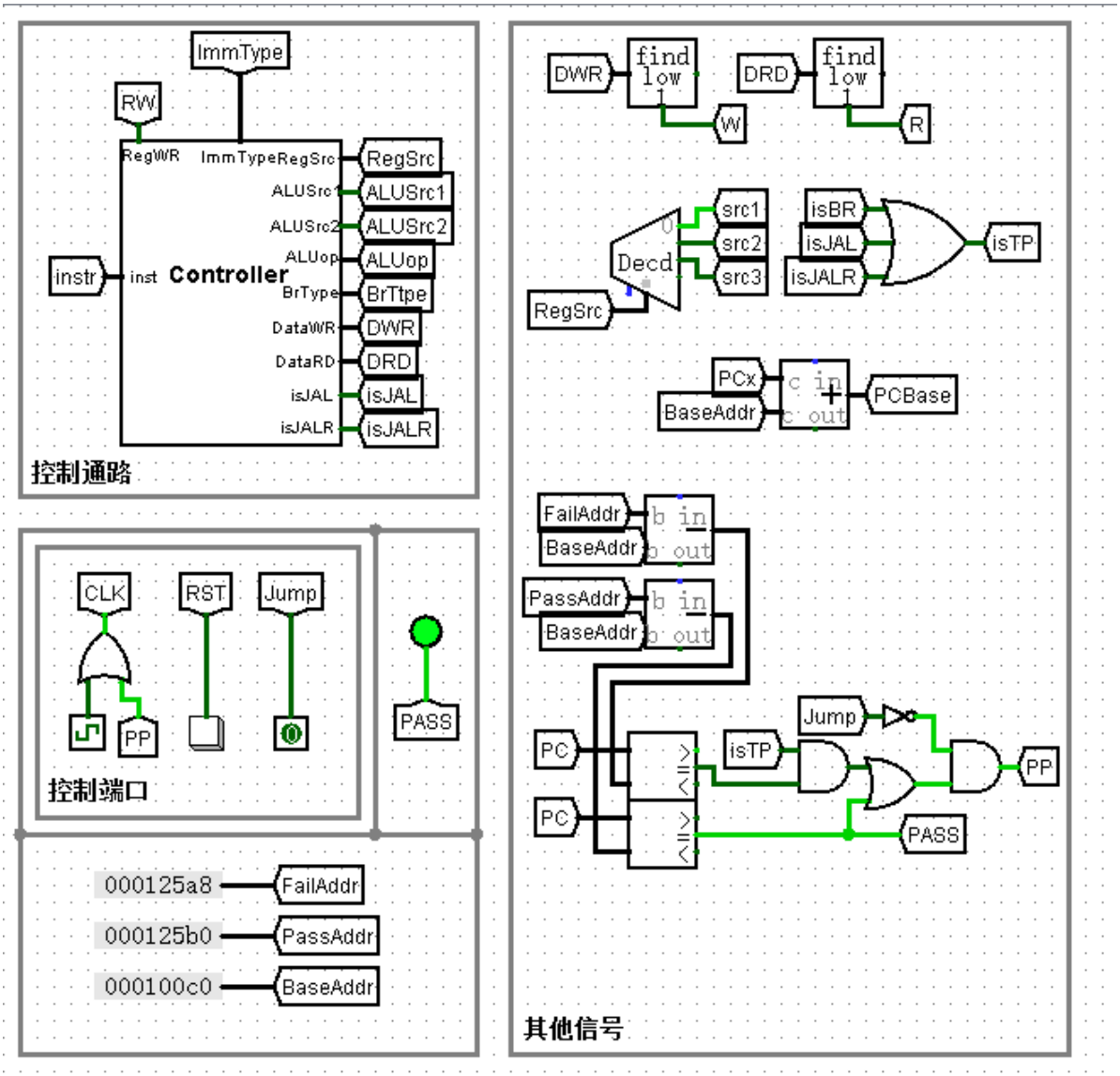
仿真如下：


RST
CLK
Debug
信号端口
LUI
AUIPC
JAL
JALR
Branch
Store
Load
ALUImm
ALU
END
Finished
评测信号
行号
40
Line
寄存器
ALU
数据存储器
立即数
跳转
译码信号
参考信号
Rs
RW
A1
A2
Aop
DW
DR
It
Bt
J
JR
Rsa
RWa
A1a
A2a
Aopa
DWa
DRa
Ita
Bta
Ja
JRa
调试信号
Continue
CNT
Line
END
Succ
Controller
D.
Test
ans
评测电路

实验八：CPU

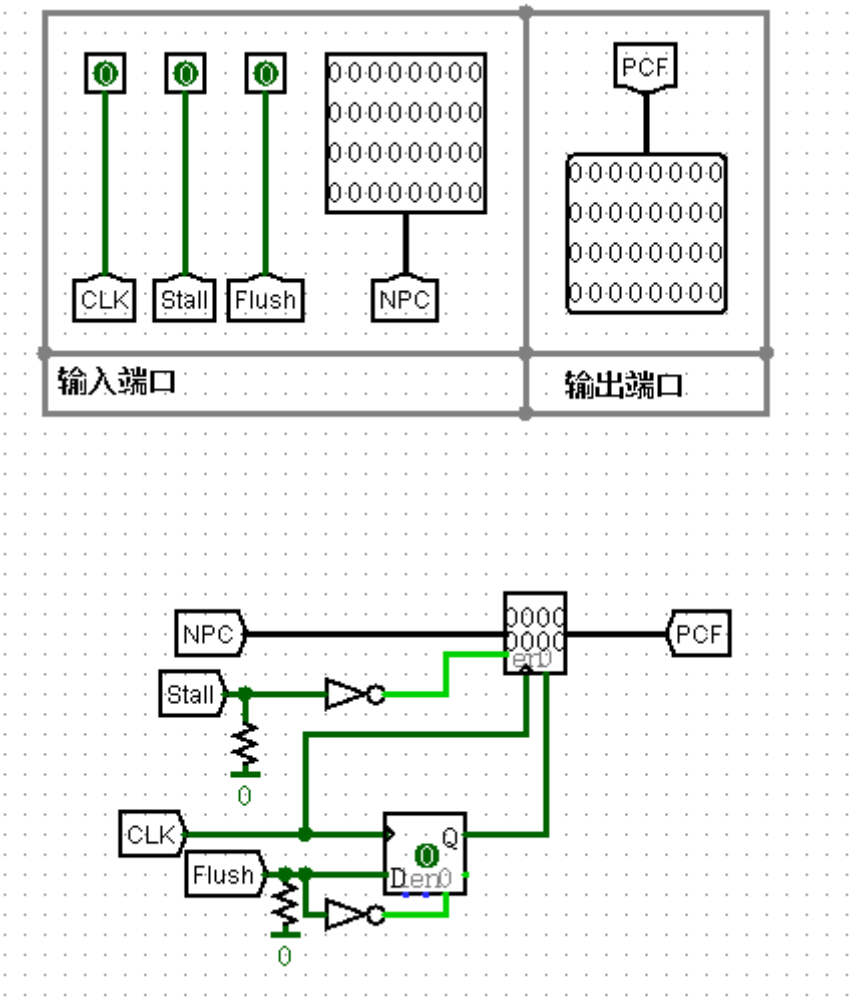


仿真如下：

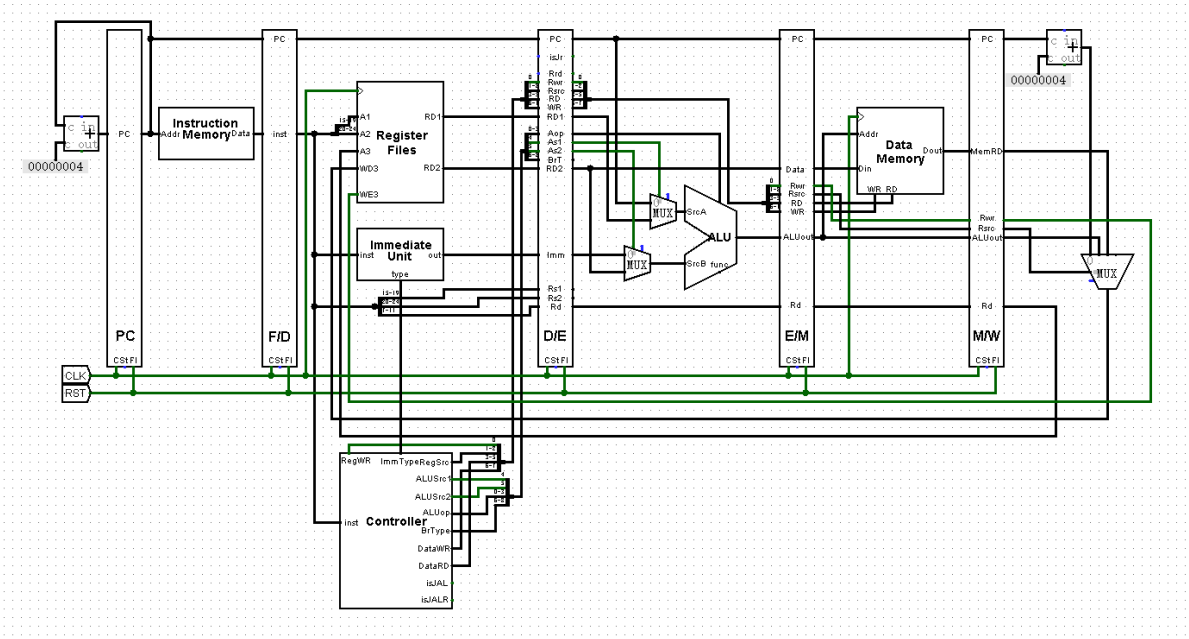


实验九：Pipeline 1

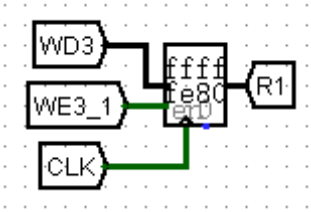
PC：


CLK
Stall
Flush
NPC
PCF
输入端口
输出端口
NPC
PCF
Stall
0
CLK
Flush
Q
0
0

CPU：

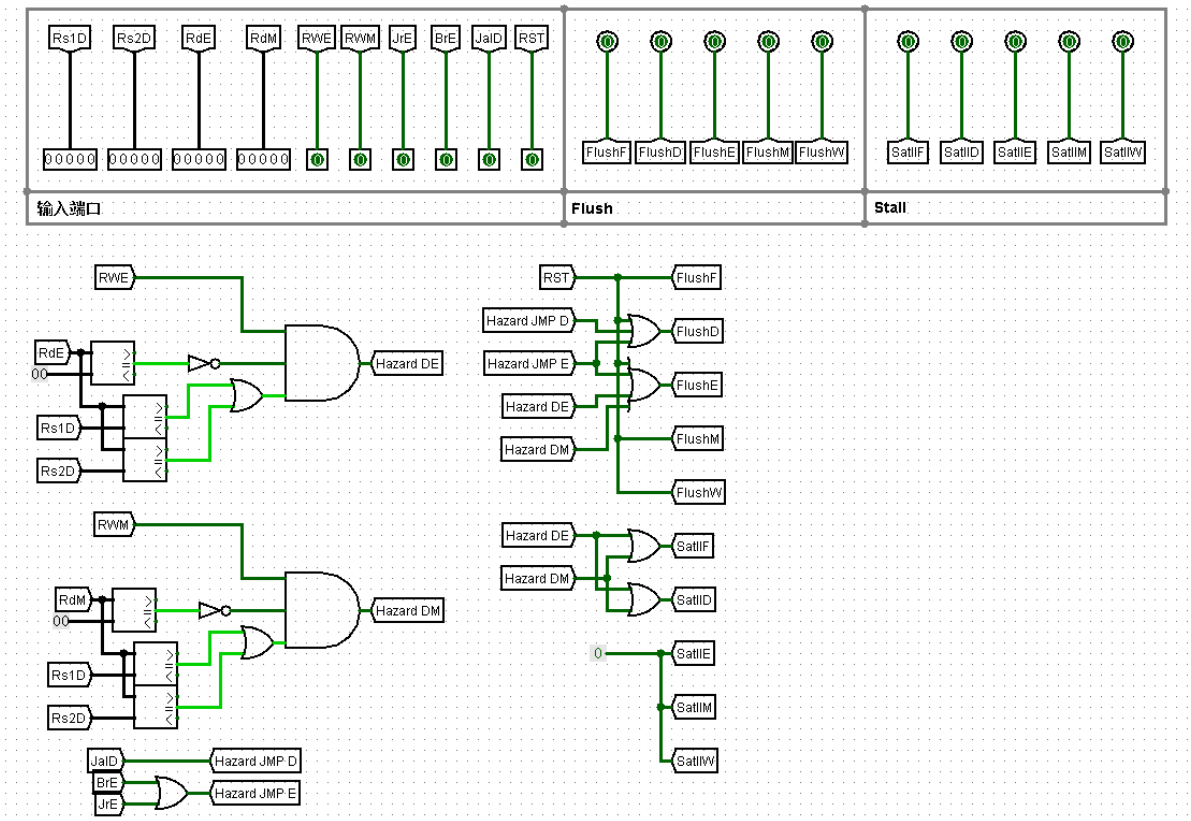

00000004
Instruction Memory
Register Files
Immediate Unit
Controller
ALU
Data Memory
MUX
PC
F/D
D/E
E/M
M/W
CLK
RST

仿真结果：

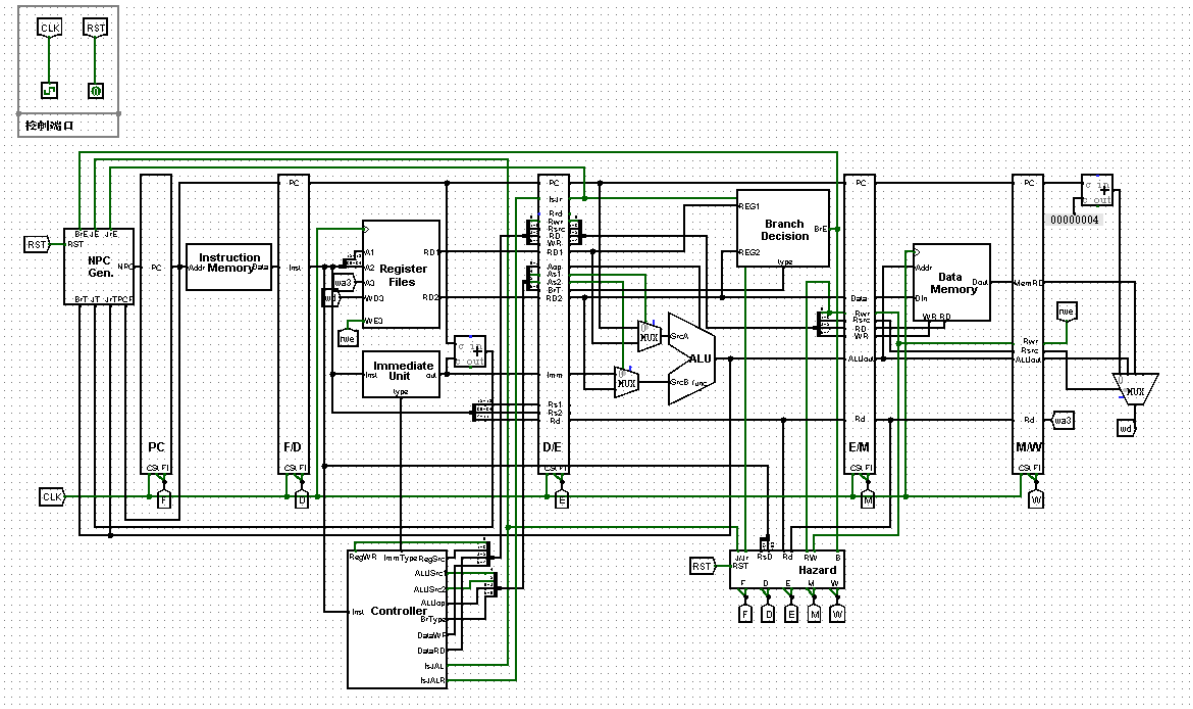

WD3
WE3_1
CLK
ffff
fe80
R1

实验十：Pipeline 2

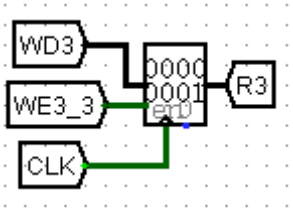
Hazard：



CPU：

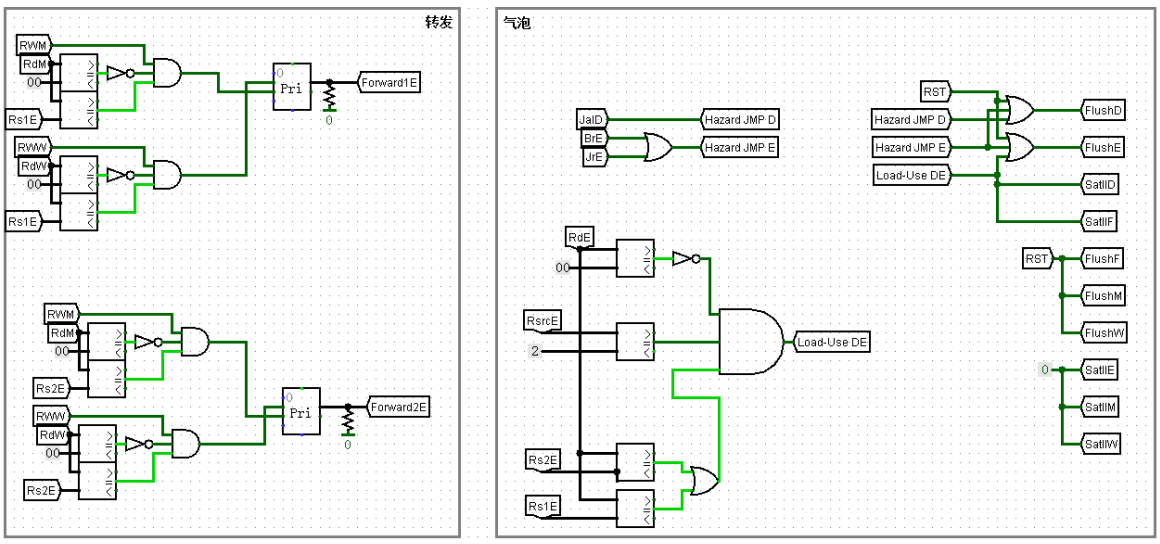


仿真如下：

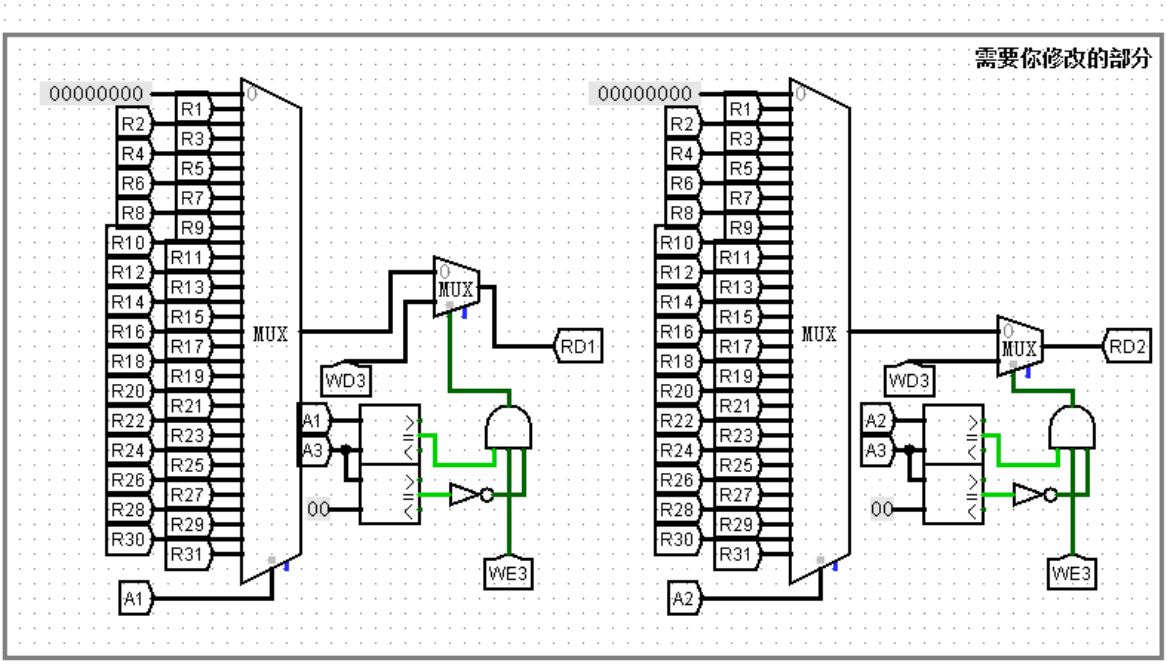


实验十一：Pipeline 3

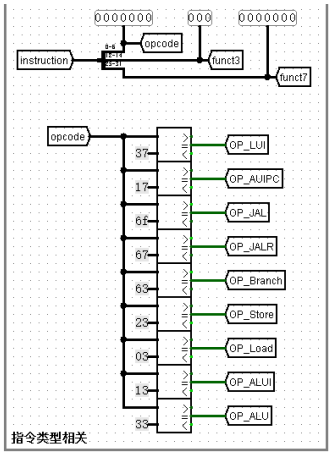
Hazard/Forward：

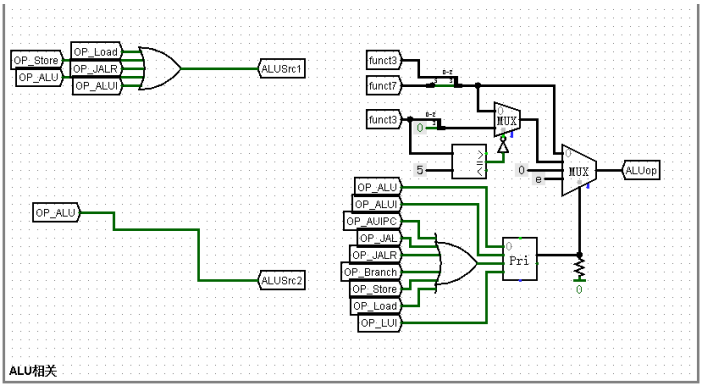


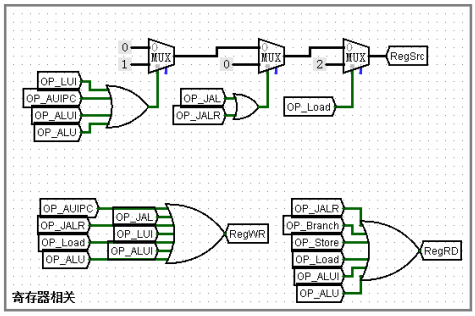
RegisterFiles：

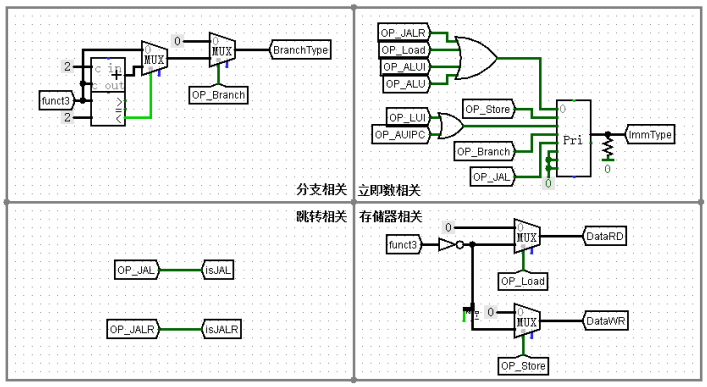


Controller：

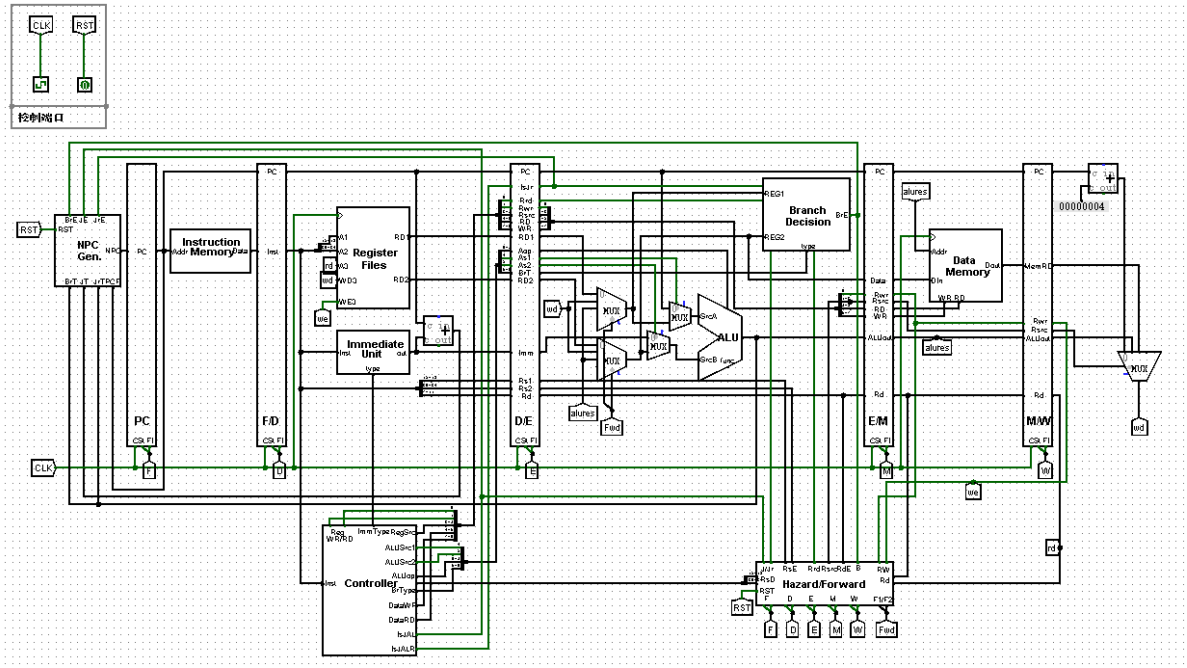




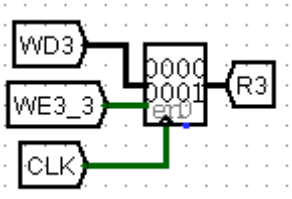




CPU：



仿真如下：



思考部分：

1. 怎样减少分支延迟槽，减少后 `Hazard Detection` 和 `Forwarding` 有何变化?

减少分支延迟槽：提前分支的判断时间。例如将Branch判断提前到 ID，此时发生的Hazard与Jal指令相同；而如果Branch与Load发生Hazard需要多暂停流水线一次，Forwarding也需要检测A1,A2与EX/Mem/WB阶段是否相同，并前递到ID处以供判断。

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?

如果允许提前Jal指令的跳转，由于此时在IF阶段就可以跳转，那么可以直接将Jal的跳转提前到IF阶段。

如果不允许提前Jal指令的跳转，则调整跳转的优先级为流水线部分越往后越优先，即EX跳转优先于ID跳转优先于IF跳转。

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。

1）将寄存器堆改为写优先。

2）让寄存器堆在时钟下降沿有效，实现前半周期写、后半周期读。

总结:

Logisim实验通过组合不同的模块再进行拼接，让我更好地理解了CPU的实现原理，加深了我的印象，对我学习组成原理很有帮助。