实验报告二　CPU 与简单模型机设计实验

姓名	王磊	学号	202231060435
专业	计算机科学与技术	年级	2022 级

一、实验目的

1. 掌握一个简单 CPU 的组成原理。

2. 在掌握部件单元电路的基础上，进一步将其构造一台基本模型计算机。

3. 定义机器指令，编写相应的微程序，并上机调试掌握整机概念。

二、实验设备

PC 机一台，Logism 实验系统一套。

三、实验原理及内容

1. 模型计算机的构成

本实验要实现一个简单的 CPU，并利用该 CPU 构建一个简单的模型计算机。CPU 由运算器（ALU）、微程序控制器（操作控制器）、寄存器堆（Regs），指令寄存器（IR）、程序计数器（PC）和存储器地址寄存器（AR）、存储器数据寄存器 DR 组成，如图 1 所示。这个 CPU 在写入相应的微指令后，就具备了执行机器指令的功能，但是机器指令一般存放在主存当中，CPU 必须和主存挂接后，才有实际的意义，所以在该 CPU 的基础上增加了主存，以构成一个简单的模型计算机。

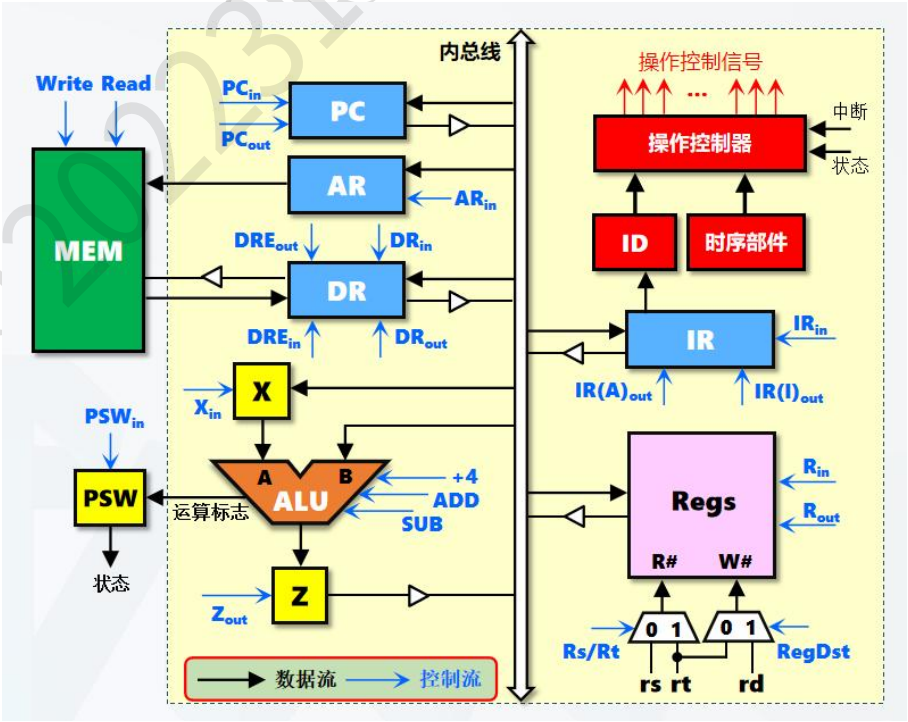

图 1 基本 CPU 构成原理图

2. 机器指令

本模型机共有五条机器指令：lw（存储器读）、sw（存储器写）、beq（比较相等跳转）、slt（比较小于置 1），addi（立即数加），其机器指令功能如图 2 所示。

#	MIPS指令	RTL功能描述
1	slt rd,rs,rt	R[rd]←R[rs]<R[rt]　小于置1，有符号比较
2	addi rt,rs,imm	R[rt]←R[rs]+SignExt(imm)　不考虑溢出
3	lw rt,imm(rs)	R[rt]←M[R[rs]+SignExt(imm)]
4	sw rt,imm(rs)	M[R[rs]+SignExt(imm)]←R[rt]
5	beq rs,rt,imm	if(R[rs] = R[rt]) PC ← PC + SignExt(imm)<<2

图 2 机器指令功能

各机器指令格式如图 3 所示。

		6bits	5bits	5bits	5bits	5bits	6bits
指令	格式	OP	rs	rt	rd	shamt	funct
slt	R	000000	Reg	Reg	Reg	0	101010
lw	I	100011	Reg	Reg	16bits 立即数		
sw	I	101011	Reg	Reg	16bits 立即数		
andi	I	001000	Reg	Reg	16bits 立即数		
beq	I	000100	Reg	Reg	16bits 立即数（相对寻址）		

图 3 机器指令格式

现有一简单的排序程序 sort-5.hex，设计基于微程序控制器的单总线 CPU，使得 sort-5.hex 能在单总线结构上运行，最终实现数据排序。

3. 微指令

微指令字长共 29 位，其微指令格式如图 4 所示。其中，PIR 为操作码判别测试字段，Pequal 为比较相等测试字段。

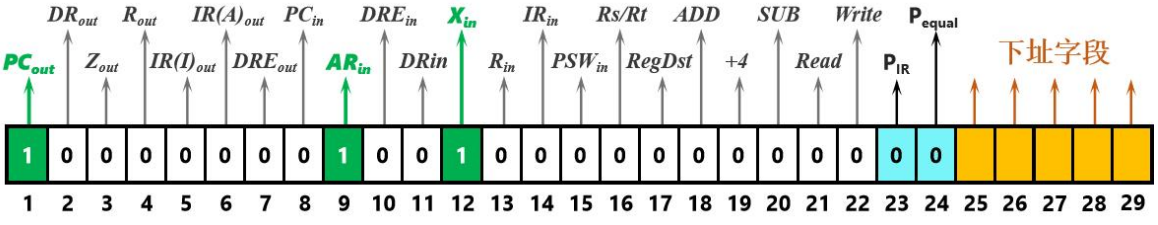

图 4 微指令格式

4.微程序

分析指令功能，本模型机所使用的微程序流程图如图 5 所示。本机共 5 条指令，共 5 路分支，占用 5 个固定起始地址。

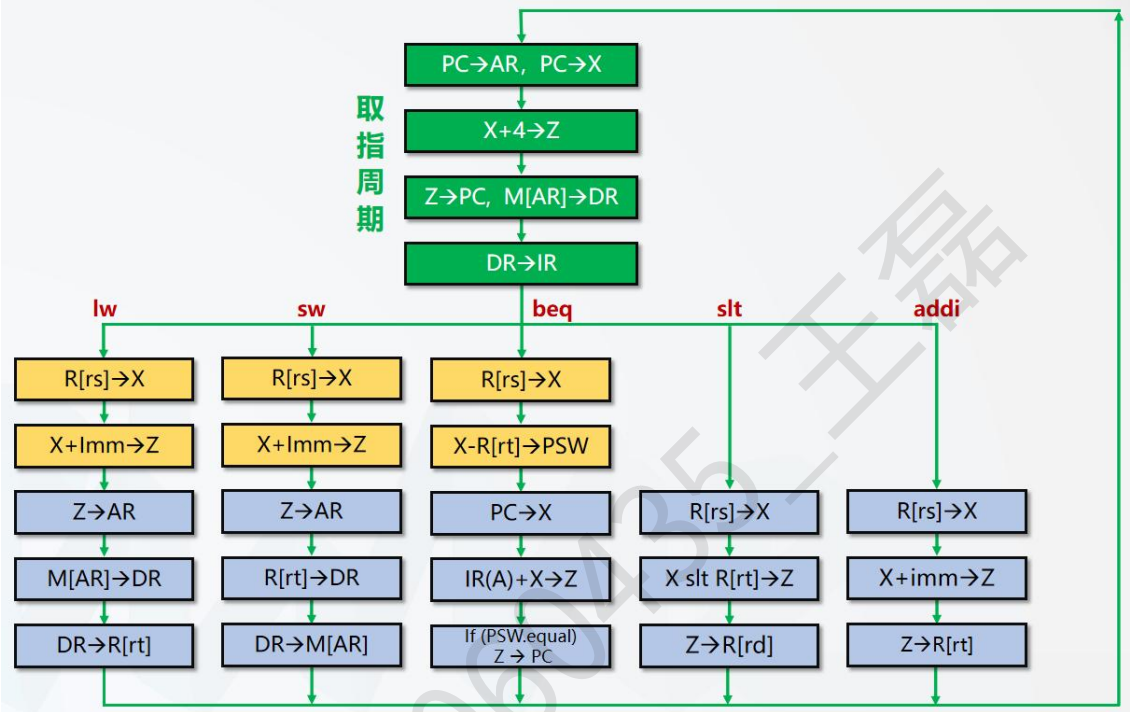

图 5 微程序流程图

思考题

1.图 6 中为何 IR 有两个 out 端 IR(I)out 和 IR(A)out 控制 IR 的输出？

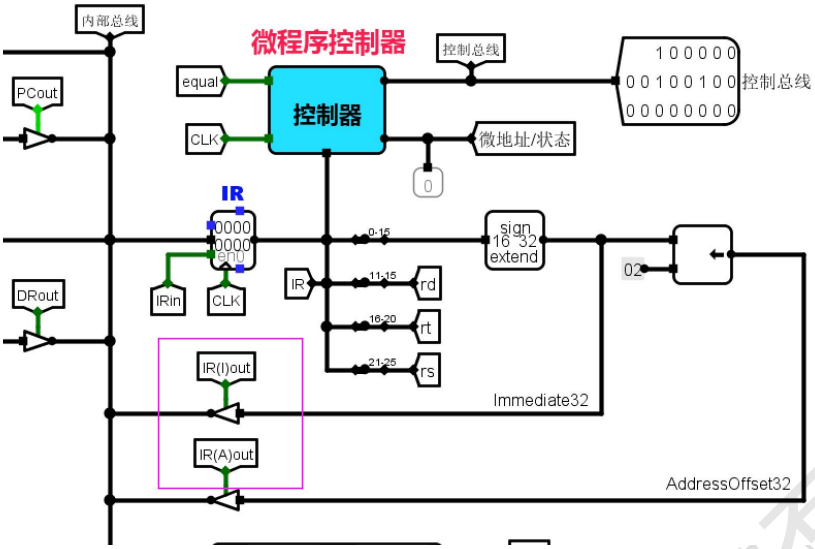

图 6 微程序控制器中 IR 的输出命令

答：指令寄存器(IR)用于存储当前正在执行的机器语言指令。这些指令通常由操作码和操作数两部分组成。将 IR 分为 IR(I)out 和 IR(A)out 两个输出端，分别用于输出操作码和操作数地址，可以更高效地将操作码发送到解码逻辑，同时将操作数地址发送给内存单元或其他部件进行相应的数据读写操作。这种设计使得指令的执行更加流畅高效，提高了 CPU 的整体性能。

2.结合图 7，简要说明该实验中，后续微指令的地址分几种情况？分别是如何形成的？

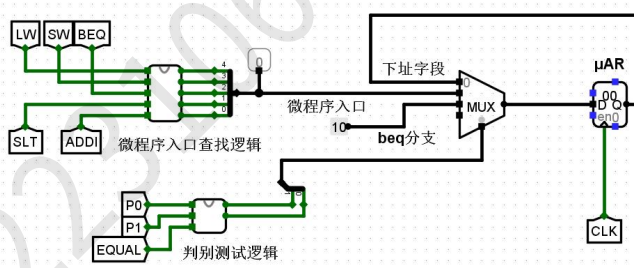

图 7 后续伪指令地址形成电路

答：两种情况。

情况一：顺序执行模式

在未涉及分支或跳转等特殊控制流程时，微指令按顺序执行。此时，后续微指令的地址通常是当前微指令地址递增 1。微程序入口查找逻辑会根据当前微指令地址自动计算出下一个微指令的地址。这个计算得出的地址随后通过多路选择器（MUX）传送至微地址寄存器（μAR）。这种机制确保了微程序在没有特殊控制需求时的连续、顺畅执行。

情况二：条件分支模式

当遇到条件分支指令（如"beq"）时，地址形成过程变得更为复杂。微程序入口查找逻辑会

依据当前微指令中的下址字段来判断是否需要执行分支。具体而言：

- 若分支条件满足，系统会选择微指令中指定的下址字段作为后续微指令的地址。

- 若分支条件不满足，系统则继续按照顺序执行模式进行。

2.结合微程序流程图和微指令格式，设计出该模型机的微程序如图 8 所示：

指令：取指令

微指令分析：取指令周期由 4 条微指令组成，存放在控制存储器的 0 号单元，这样系统上电后就能从这里开始执行。

第一条微指令（地址 0）：

- 功能：将 PC 的内容通过总线送到 AR 和 X 寄存器
- 控制信号：PCout、ARin、Xin 这三位置 1，其他置 0
- 判别测试位 P1、P0 置 0（表示顺序执行）
- 下址字段设为 1（指向下一条指令）

第二条微指令（地址 1）：

- 功能：X 寄存器值加 4 存入到 Z 寄存器，为下一条指令做准备
- 控制信号：仅 add4 位置 1
- 判别测试位置 0
- 下址字段设为 2

第三条微指令（地址 2）：

- 功能：将 Z 中的内容送入 PC，AR 中内容送至 DR
- 控制信号：PCout，PCin，DREin，Read 位置 1，其他置 0
- 判别测试位置 0
- 下址字段设为 3

第四条微指令（地址 3）：

- 功能：将 DR 中的内容送入 IR
- 控制信号：DR out 和 IR in
- P0 位置 1（用于指令译码）
- 下址字段置 0（回到 0 号单元）

执行完最后一条微指令后，系统会根据指令译码信号，通过地址转移逻辑找到对应的微程序入口地址，开始执行具体指令的微程序。

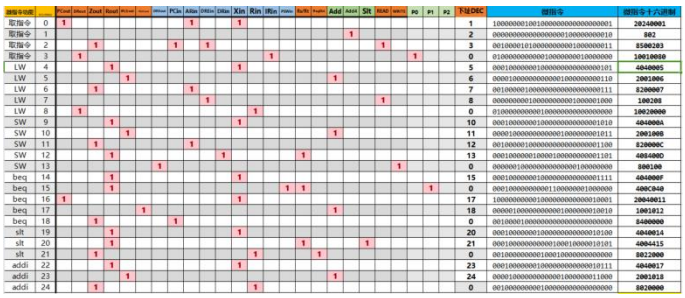

图 8 模型机微程序生成

3.运行 sort-5.hex 并对实验结果进行分析

将图 8 的微指令十六进制存储到微程序控制器的控制存储器中：

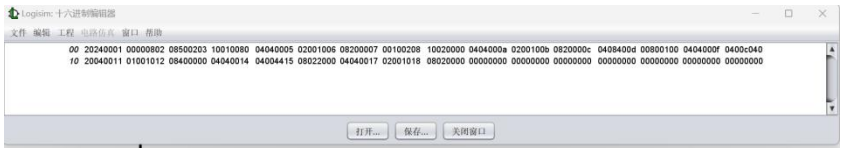

图 7 控制存储器存储指令

然后在单总线 CPU(微程序)的外部总线中的 MEM 中导入排序程序 sort-5.hex 并让时钟连续运行进行排序：

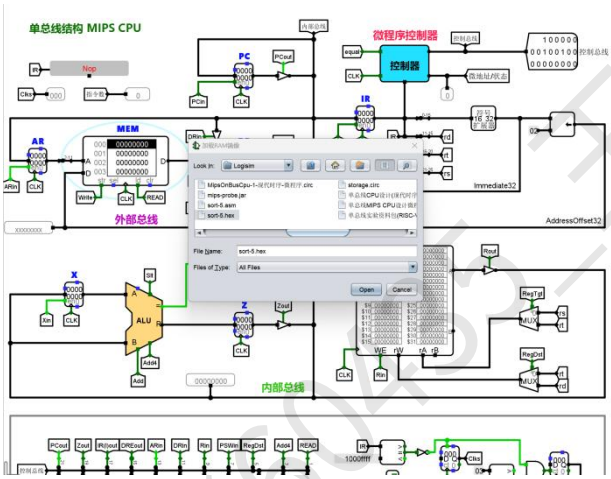

图 8 向 MEM 中导入排序程序 sort-5.hex

运行后观察排序结果，可以发现 080 处的数据按照降序排列

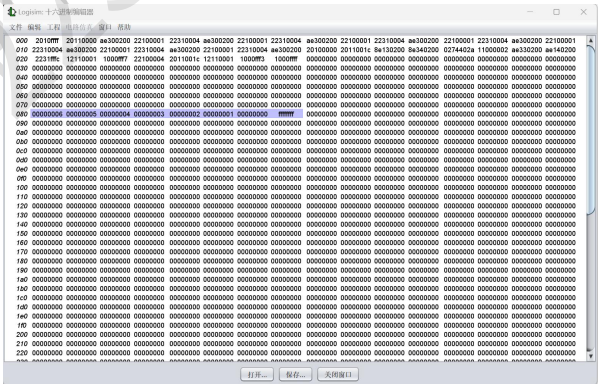

图 9 排序后的结果

4.请在基于微程序控制器的模型里增加一条指令，并编写一段示例程序，以证明该指令添加成功。

一、指令分析：

1. 分析 XOR 指令

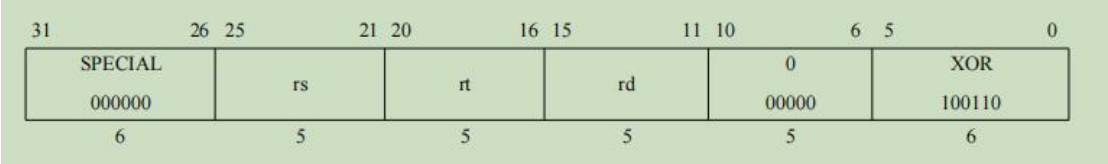

图 10 MIPS 中的 XOR 指令格式

XOR 对应的微指令如下：

R[rs] xor R[rt] -> R[rd]

解释：把 rs 寄存器的值与 rt 寄存器的值进行异或，然后将结果存入 rd 寄存器中

2. 分析 SLL 指令

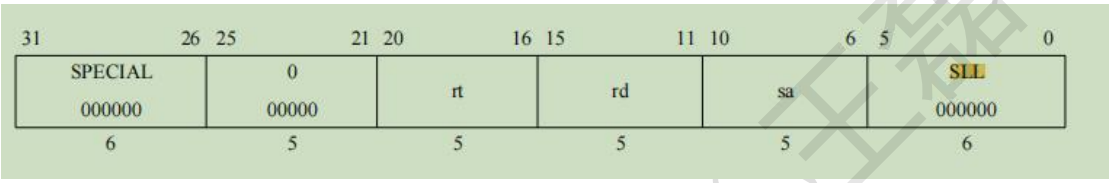

图 11 MIPS 中的 SLL 指令格式

SLL 对应的指令如下：

rd ← rt << sa

微操作：

s ← sa

temp ← GPR[rt](31-s)..0 || 0s

GPR[rd]← temp

解释：将 rt 寄存器的值左移 sa 位，结果存储在 rd 寄存器中。

二、电路修改

（1）修改指令译码器：

操作码皆为 0：

FUNCT=26（二进制 100110）时，是 XOR 指令

FUNCT=00（二进制 000000）时，是 SLL 指令

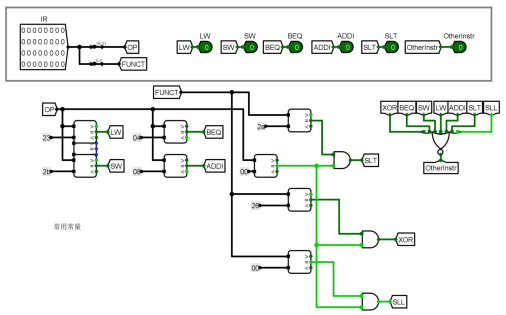

图 12 指令译码器

（2）修改微程序入口查找逻辑：

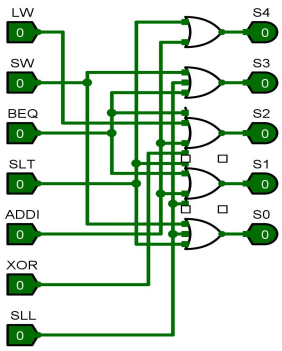

图 13 微程序入口查找逻辑

（3）XOR，SLL 的微地址分别是 4，11

修改 ALU 电路：修改多路选择器的 0，2 号路分别为异或运算，逻辑左移运算

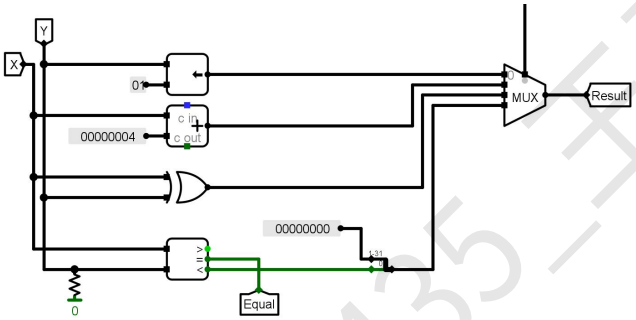

图 14 ALU 修改

（4）在 mem 中写入两条指令：

00011026：执行 0 号和 1 号寄存器中数值的异或运算，并将运算结果存入 2 号寄存器中

00011840：对 1 号寄存器内的数值执行 1 位的逻辑左移操作，然后将操作后的结果存入 3 号寄存器

结果：0 号和 1 号寄存器的数值分别是 0000H 和 0110H，2 号和 3 号寄存器中的数值分别变为 0110H 和 0220H

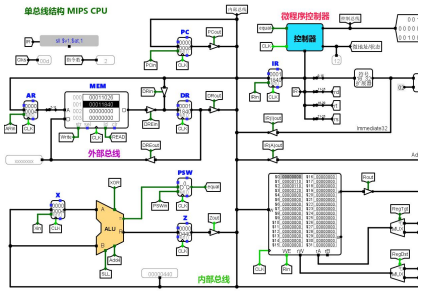

图 15 运算结果

实验结果与理论结果一致，电路设计正确

4. 请在基于组合逻辑控制器的模型机上增加一条指令。该模型机为课堂上讲过的模型机，

见学习通 6.3。编写一段示例程序，以证明该指令添加成功。

1. 分析 XORI 指令

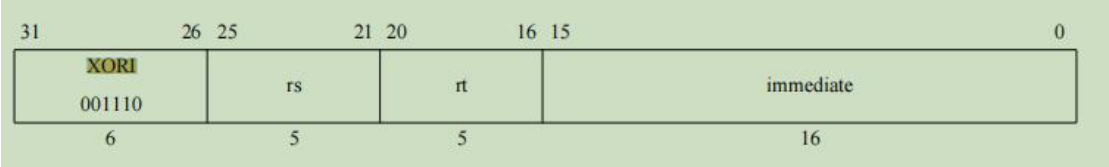

图 16 MIPS 中的 XORI 指令格式

（1）前六位为操作码（OP）字段

（2）指令功能：rt = rs ^ Immediate

寄存器 rs 的值与低 16 的值进行亦或，并将结果存入 rt 寄存器

2. 电路修改

（1）修改控制器

（2）在控制器的 op 部分电路进行更新，添加一个比较器，将其与高位常量 001110（0xe）连接以及输出端连接一个控制信号 XORI。

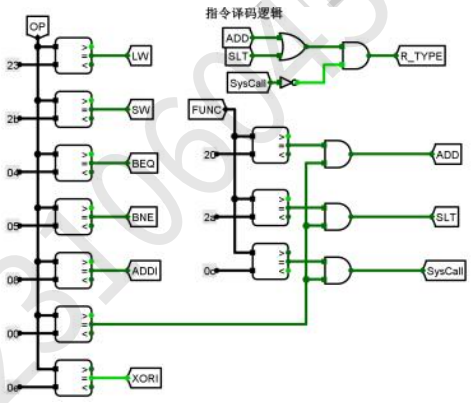

图 17 op 部分修改

（3）控制器中添加控制信号 XORI

由于存在寄存器写入操作，且低 16 位的选择与否跟此多选器和 AlrSrc 信号有关，所以我们要连接两个控制信号

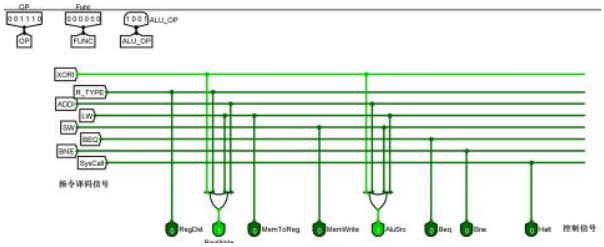

图 18 控制信号添加

（4）ALU 控制器逻辑修改

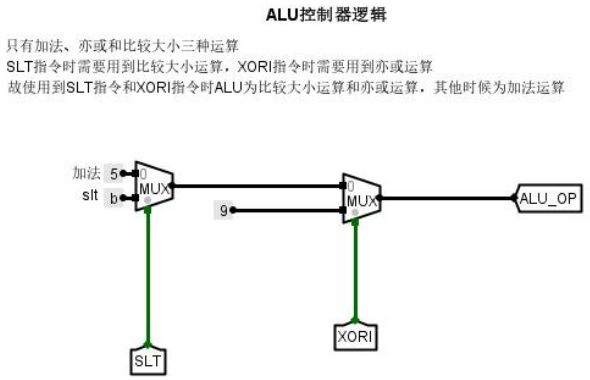

图 12　ALU 控制器逻辑修改

3. 编写指令验证

（1）因为 XORI 指令功能为：rt = rs ^ Immediate，即 rs 的值与低 16 的值进行亦或，所以我们需要先写入一个值到 rs 寄存器中，这个我们采用 addi 指令完成

（2）以 ADDI 指令"20100010"为例,根据 MIPS 指令集手册,该指令的功能是将 rs 寄存器的内容与指令低 16 位表示的有符号立即数相加,并将结果写入到 rt 寄存器中。在这条指令中,rt 字段为 16,因此执行结果会被写入到编号为 16 的寄存器中

（3）设计 XORI 指令如 3a11000F,换算为二进制为

001110 10000 10001 0000 0000 0000 1111

即为将 10000（即 16 号地址）的内容与低 16 位（扩展 32）的内容异或后存入 10001（即 17 号）寄存器中

（4）16 号寄存器的值应为 00000010，10 与 XORI 指令中的 1111（其他位置全 0）异或后的值为 0001 1111（其他位置全 0），转换为 16 进制后的理论值为 0000001F

（5）在 Logisim 中的电路上进行实践验证结果

（6）将以上两条指令存入指令存储器

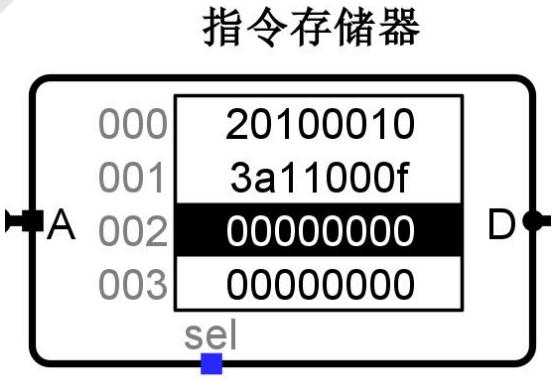

图 13 将指令存入寄存器

（7）运行电路检查结果

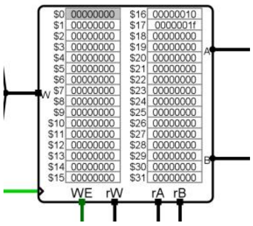

图 14 寄存器结果显示

（8）实验结果与理论结果一致，电路设计正确