长沙理工大学

《系统能力综合实训》报告

MIPS 多周期处理器设计

学　　院 计算机与通信工程　专　　业 计算机科学与技术

班　　级 计算机 21-3 班　学　　号 202108061012

学生姓名 梁梓木　指导教师 罗永红

课程成绩 _____　完成日期 2023 年 1 月 4 日

《系统能力综合实训》成绩评定标准

毕业要求	考核与评价方式及成绩比例（%）			成绩比例（%）
	系统演示	项目答辩	实训报告	
实训目标 1：**问题分析**	15	10	17	42
实训目标 2：**使用现代工具**	15	10	3	28
实训目标 3：**环境和可持续发展**	/	/	15	15
实训目标 4：**个人和团队**	/	/	10	10
实训目标 5：**项目管理**	/	/	5	5
合计	30	20	50	100

《系统能力综合实训》成绩评定

毕业要求	考核与评价方式及成绩比例（%）			成绩比例（%）
	系统演示	项目答辩	实训报告	
实训目标 1：**问题分析**				
实训目标 2：**使用现代工具**				
实训目标 3：**环境和可持续发展**	/	/		
实训目标 4：**个人和团队**	/	/		
实训目标 5：**项目管理**	/	/		
合计				

指导教师对系统能力综合实训的评定意见

综合成绩 _____指导教师签字_____ 年 月 日

MIPS 多周期处理器设计

学生姓名：梁梓木　　指导老师：罗永红

摘　要： 运用 Logisim 实现了模拟多周期 MIPS 硬布线处理器，理解并运用各寄存器、存储器、运算器、控制器。涉及到的指令有存数、取数、加减法等。并使用汇编语言的冒泡排序代码在 CPU 中运行，成功实现从大到小排序的功能。主要工作在于设计硬布线控制器与学习数据通路与指令。最后实现了数字从大到小的冒牌排序排序。

关键词： MIPS；处理器；指令；冒泡排序

The Design of MIPS Multi-cycle Processor

Student name: Liang ZiMu Advisor: Luo YongHong

Abstract: Using Logisim, I have implemented a simulated multi-cycle MIPS hardwired processor, fully understanding and utilizing various registers, memories, arithmetic units, and controllers. The instructions involved are those of storing numbers, fetching numbers, addition, subtraction, etc. I have also used the bubble sorting code in assembly language to run on the CPU, successfully achieving the sorting function from largest to smallest. My main work lies in the design of the hardwired controller and learning about the data path and instructions. Finally, I have achieved the bubble sorting function of numbers from largest to smallest.

Keywords: MIPS; Processor; Instruction; Bubble Sort

目　录

1 引言

1.1 背景

随着计算机技术的不断发展，处理器作为计算机的核心部件，其性能和功能也在不断提升。多周期硬布线处理器是一种常见的处理器类型，其通过多个时钟周期完成指令执行，具有较高的执行效率和稳定性。在数字电路设计中，逻辑门是构成处理器等数字系统的基础元件，通过逻辑门的组合和配置可以实现各种复杂的逻辑功能。因此，利用逻辑仿真软件对处理器进行模拟和设计具有重要的实际意义和应用价值。

1.2 项目介绍

本次实训项目旨在通过逻辑仿真软件 Logisim 实现对多周期硬布线处理器的模拟和设计。通过学习和实践，我们将掌握逻辑门的基本原理和配置方法，了解多周期硬布线处理器的设计和实现过程。通过亲手实践，提高我们分析和解决实际问题的能力，培养我们的工程素养和创新精神。同时，通过本项目的实践，为后续深入学习处理器设计和实现打下坚实的基础。

1.3 系统任务

设计多周期 mips 硬布线处理器：

（1）实现对处理器各部件的连线组装

（2）实现各部件的内部逻辑

（3）设计指令，实现处理器的功能

（4）用汇编代码验证系统可行性

2 总体方案设计

2.1 设计核心部分

数据通路主要包括有一个存储器，寄存器堆，ALU 运算器和硬布线控制器。详情请见第四部分。除此之外主要设计还有状态机，ALU 运算器等。

2.2 功能分析

通过存数，取数，加法，减法，带条件的跳转等，同时还要对寄存器组进行操作。使得系统可以实现简单的用 mips 汇编语言编写的算法。并将结果于存储器中显示。

2.3 测试方案

用 mips 汇编语言编写的冒泡排序转换成机器代码，写入存储器中，使处理器运行，可以看见在存储器中显示排序的结果。

3 系统软件环境

本系统开发于本机环境中。用到的软件有基于 java 的 Logisim 仿真软件，Mars 汇编编译器。

3.1 Logisim

Logisim 是一种用于设计和模拟数字逻辑电路的教育工具。它可以帮助用户创建和测试各种类型的数字电路，包括组合逻辑电路和时序逻辑电路。在 Logisim 中，用户可以使用各种不同的组件（如逻辑门、触发器、寄存器等）来构建电路

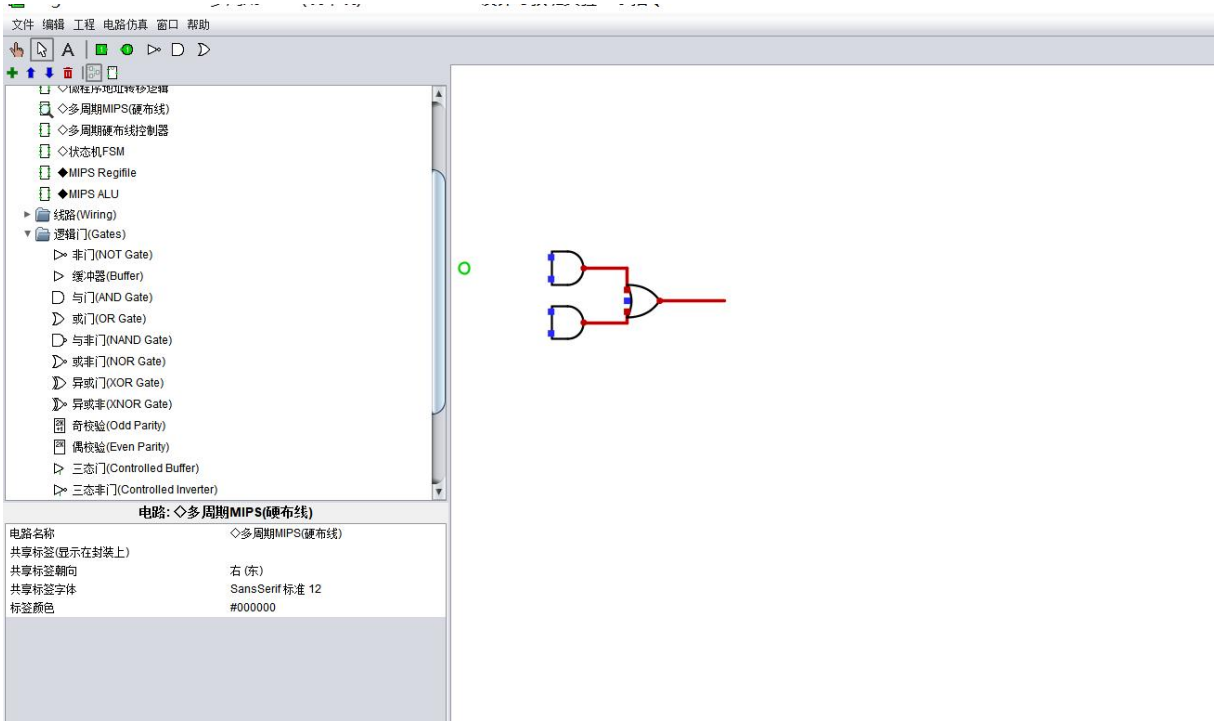


图 3.1 Logisim

3.2 Mars

Mars 是一种编程框架，可以构建分布式计算系统，包括基于 Mars 的模块化编程框架和基于 Mars 的分布式计算环境。它提供了一种基于消息传递的编程范式，允许开发者定义异步、并发和分布式系统。Mars 提供了许多工具和库，可以帮助开发者快速构建分布式系统，包括通信协议、数据序列化、分布式状态管理、任务调度等方面的支持。

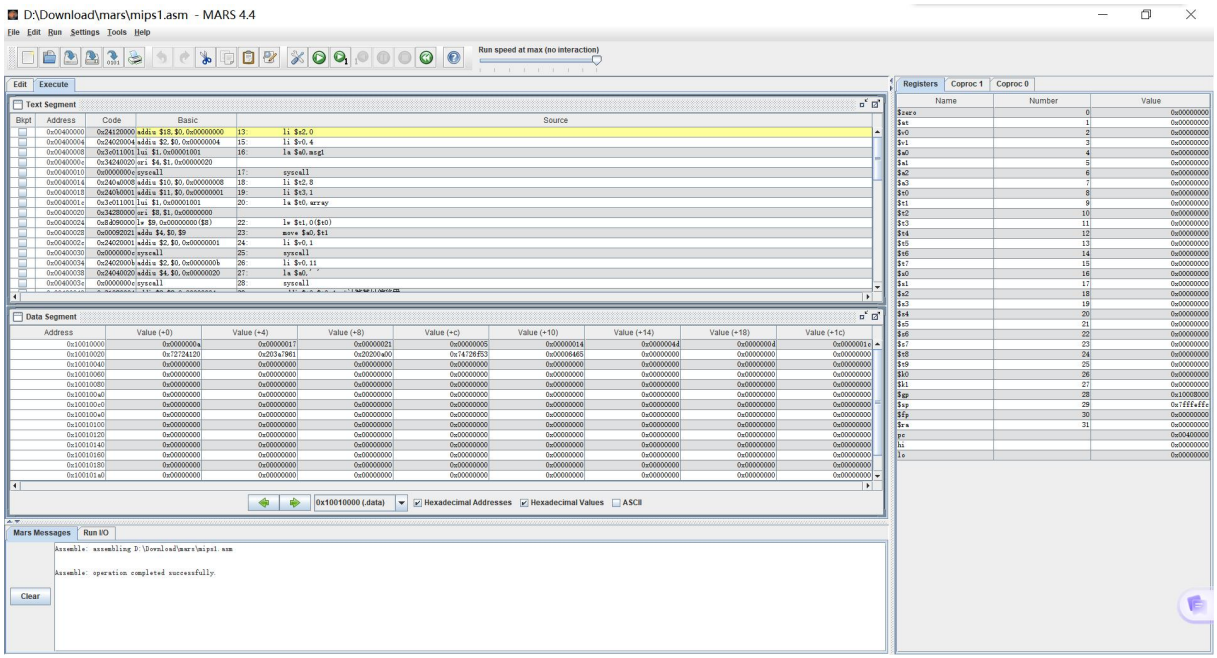


图 3.2 Mars

4 MIPS 多周期 CPU 设计

4.1 数据通路设计

数据通路设计见图 4.1 所示：

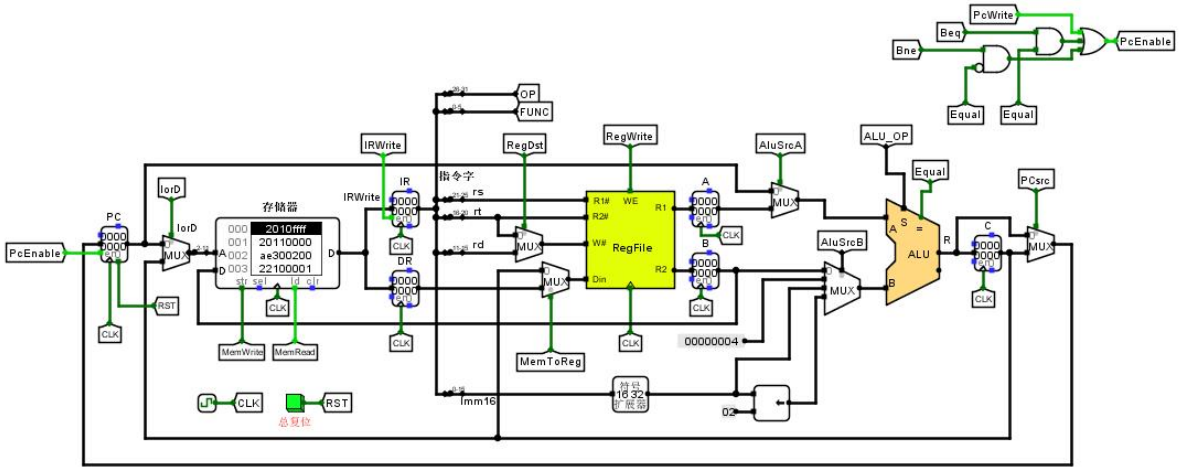


图 4.1 数据通路设计

数据通路包含一个存储器用于存储指令和数据，还有寄存器堆，ALU 运算器。多周期 CPU 必须要有锁存器来保存上一次周期的临时数据[1]，因此还设计了 IR，DR，A，B，C 五个寄存器。IR 寄存器用于保存取指令周期后的指令数据。DR 寄存器用于保存取数据周期后的数据。A 寄存器用于存储 rs 选中的寄存器的数据。B 寄存器用于存储 rt 选中的寄存器的数据。C 寄存器用于存储 ALU 运算过后的值。

4.2 硬布线控制器设计

硬布线控制器设计见图 4.2 所示：

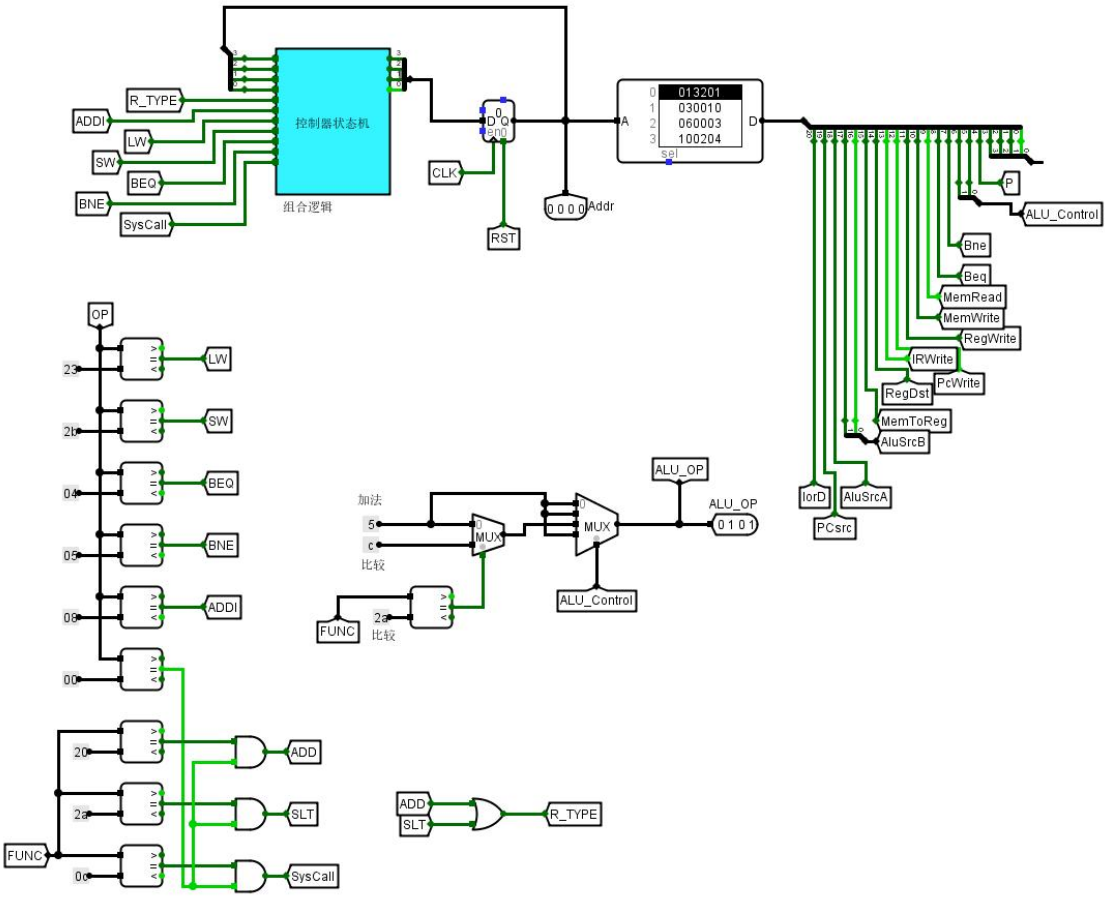


图 4.2 硬布线控制器设计

OP，Func 为输入信号，输出为其他引脚信号，引脚功能见图 4.3 所示：

信号	功能
RST	复位信号
CLK	时钟信号
IorD	控制这条指令是指令取值还是数据取值
MemWrite	存储器写信号
MemRead	存储器读信号
IRWrite	该信号控制从存储器读出的信号是否写入 IR 寄存器
OP	指令的 OP 字段，是用于辨识指令类型的条件之一
Func	指令的 Func 字段，是用于辨识指令类型的条件之一
RegDst	选择寄存器堆的 W#输入源是 rt 或 rd
PcWrite	控制 PC 寄存器的写开关
Beq、Bne	根据 ALU 的条件，控制条件转移所需的 PC 寄存器写开关
PcSrc	控制 PC 寄存器的写入值
AluOP	控制 ALU 的运算方式
AluSrcB	控制 ALU 的 B 端的输入值
AluSrcA	控制 ALU 的 A 端的输入值
RegWrite	控制寄存器堆可写，写入值由 Din 提供
MemToReg	选择打开存储器到寄存器的通路或 C 寄存器到寄存器堆的通路

图 4.3 引脚功能

4.2.1 指令译码逻辑

译码逻辑中通过 OP、Func 用比较器与常量进行比较确定是哪个类型的指令。例如若 OP 为 0 则该指令为 R 型指令，再由 FUNC 字段确定该指令功能具体是什么。指令译码逻辑设计见图 4.4 所示：

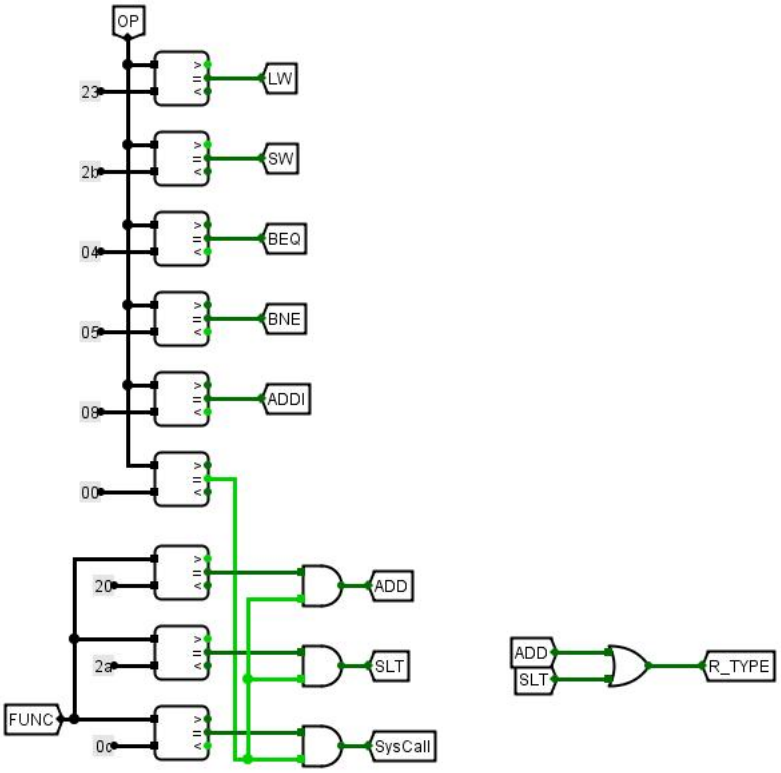


图 4.4 指令译码逻辑设计

4.2.1 ALU 控制器逻辑

控制器逻辑设计见图 4.5 所示：

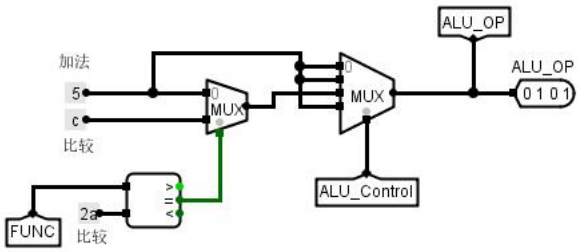


图 4.5 ALU 控制器逻辑设计

当 ALU_Control 为 00、01、11 时，运算器做加法，为 10 时运算器将根据 FUNC 决定。

4.3 寄存器堆

寄存器堆由多个寄存器组成，每个寄存器可以存储一个数据项。这些寄存器通常按照一定的顺序排列，以便于访问和操作。在处理器中，寄存器堆可以用于存储操作数、临时结果、指令地址等信息。寄存器堆见图示 4.6：

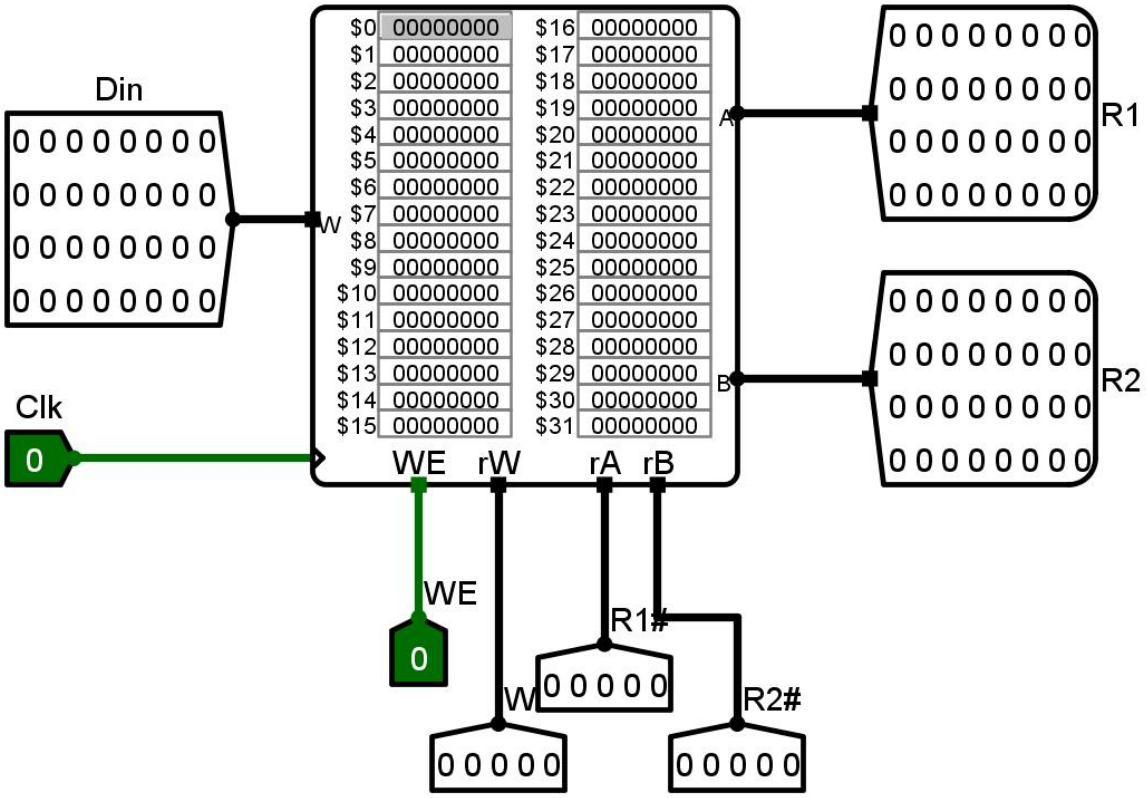
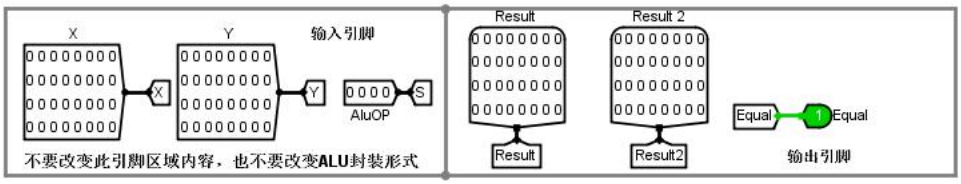


图 4.6 寄存器堆

4.4 运算器 ALU

运算器是计算机中执行各种算术和逻辑运算操作的部件。它的基本操作包括加、减、乘、除四则运算，与、或、非、异或等逻辑操作，以及移位、比较和传送等操作。运算器的处理对象是数据，所以数据长度和计算机数据表示方法对运算器的性能影响极大。运算器 ALU 见图示 4.7：



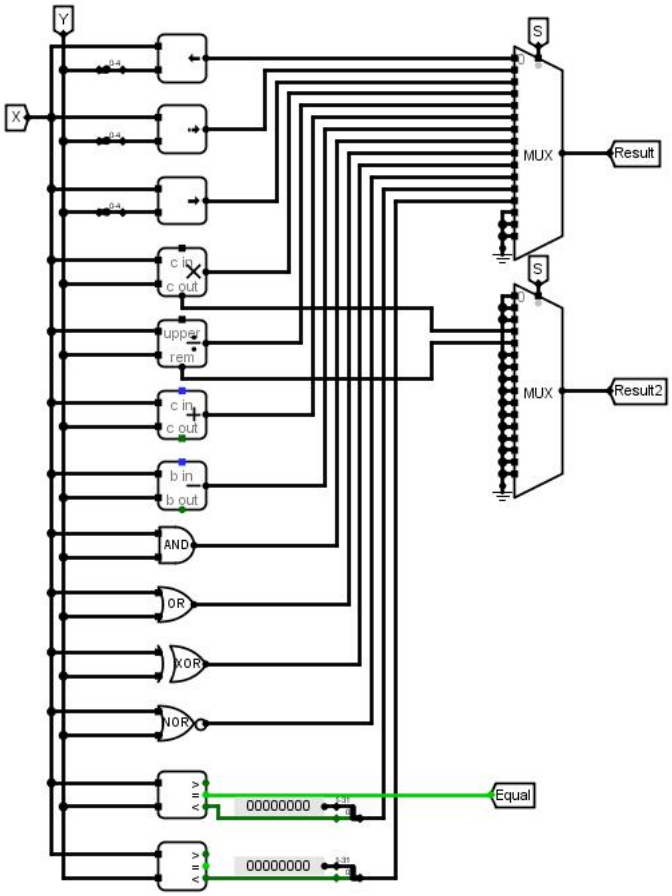


图 4.7 运算器 ALU

4.5 状态机

状态机设计方法：根据状态转换图填写 Excel。如图 4.8：

当前状态(现态)					输入信号							下一状态（次态）				
S3	S2	S1	S0	现态10进制	R_Type	LW	SW	BEQ	BNE	SYSCALL	ADDI	次态10进制	N3	N2	N1	N0
0	0	0	0	0								1	0	0	0	1
0	0	0	1	1	1							7	0	1	1	1
0	0	0	1	1		1						2	0	0	1	0
0	0	0	1	1			1					5	0	1	0	1
0	0	0	1	1				1				9	1	0	0	1
0	0	0	1	1					1			10	1	0	1	0
0	0	0	1	1						1		13	1	1	0	1
0	0	0	1	1							1	11	1	0	1	1

图 4.8 状态转换图填写的 Excel

然后利用自动生成的表达式生成状态机电路，如图 4.9 所示：

S3	S2	S1	S0	R_Type	LW	SW	BEQ	BNE	SYSCALL	ADDI	最小项表达式	N3	N2	N1	N0
~S3&	~S2&	~S1&	~S0&								~S3&~S2&~S1&~S0				~S3&~S2&~S1&~S0+
~S3&	~S2&	~S1&	S0&	R_Type&							~S3&~S2&~S1&S0&R_Type		~S3&~S2&~S1&S0&R_Type+	~S3&~S2&~S1&S0&R_Type+	~S3&~S2&~S1&S0&R_Type+
~S3&	~S2&	~S1&	S0&		LW&						~S3&~S2&~S1&S0&LW			~S3&~S2&~S1&S0&LW+	
~S3&	~S2&	~S1&	S0&			SW&					~S3&~S2&~S1&S0&SW		~S3&~S2&~S1&S0&SW+		~S3&~S2&~S1&S0&SW+
~S3&	~S2&	~S1&	S0&				BEQ&				~S3&~S2&~S1&S0&BEQ	~S3&~S2&~S1&S0&BEQ+			~S3&~S2&~S1&S0&BEQ+
~S3&	~S2&	~S1&	S0&					BNE&			~S3&~S2&~S1&S0&BNE	~S3&~S2&~S1&S0&BNE+		~S3&~S2&~S1&S0&BNE+	
~S3&	~S2&	~S1&	S0&						SYSCALL&		~S3&~S2&~S1&S0&SYSCALL	~S3&~S2&~S1&S0&SYSCALL+	~S3&~S2&~S1&S0&SYSCALL+		~S3&~S2&~S1&S0&SYSCALL+
~S3&	~S2&	~S1&	S0&							ADDI&	~S3&~S2&~S1&S0&ADDI	~S3&~S2&~S1&S0&ADDI+		~S3&~S2&~S1&S0&ADDI+	~S3&~S2&~S1&S0&ADDI+
~S3&	~S2&	S1&	~S0&								~S3&~S2&S1&~S0			~S3&~S2&S1&~S0+	~S3&~S2&S1&~S0+
~S3&	~S2&	S1&	S0&								~S3&~S2&S1&S0		~S3&~S2&S1&S0+		
~S3&	S2&	~S1&	~S0&								~S3&S2&~S1&~S0				
~S3&	S2&	~S1&	S0&								~S3&S2&~S1&S0		~S3&S2&~S1&S0+	~S3&S2&~S1&S0+	
~S3&	S2&	S1&	~S0&								~S3&S2&S1&~S0				
~S3&	S2&	S1&	S0&								~S3&S2&S1&S0	~S3&S2&S1&S0+			
S3&	~S2&	~S1&	~S0&								S3&~S2&~S1&~S0				
S3&	~S2&	~S1&	S0&								S3&~S2&~S1&S0				
S3&	~S2&	S1&	~S0&								S3&~S2&S1&~S0				
S3&	~S2&	S1&	S0&								S3&~S2&S1&S0	S3&~S2&S1&S0+	S3&~S2&S1&S0+		
S3&	S2&	~S1&	~S0&								S3&S2&~S1&~S0				
											逻辑表达式->>>				

图 4.9 表达式自动生成

状态机部分电路如图 4.10 所示：

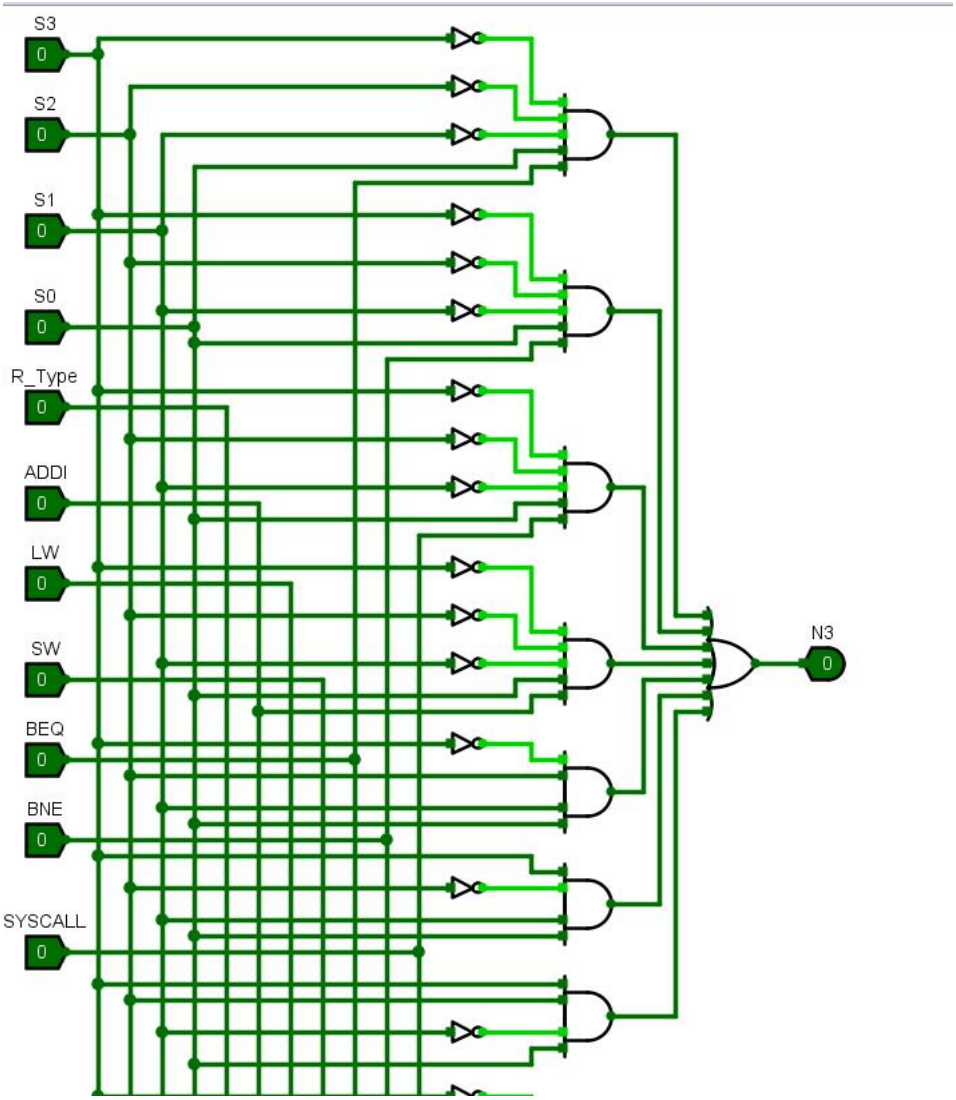


图 4.10 状态机部分电路

5 数据通路建立过程及指令介绍

5.1 取指令阶段

送指令字到 IR 同时 PC 累加，PC 与 ALU 之间存在直接的数据通路，ALUScrB 控制的多路选择器选择 01 的控制信号，选中 4 和 PC 做加法，算出结果送回 PC 端实现 PC+4。在这同时，PC 也在进行指令存储器的访问（指令存储器和数据存储器为同一个存储器），PC 给出地址送到指令存储器，取出指令送到 IR 寄存器。这就是取指令的阶段，需要一个时钟周期[2]。

5.2 译码阶段

第二阶段会进行译码，将 func 和 op 送到控制器进行译码产生译码控制信号，然后要进行取操作数，R1 和 R2 会读对应的寄存器进行读出，相应立即数也会进行扩展，这是取指令的部分。然后将 PC+4 的值与立即数左移两位进行相加，时钟来临时会将 R1 和 R2 的值传入 a、b 寄存器，运算的分支地址由 ALU 传入到 c 寄存器。在这之后便是各指令的执行阶段了。

5.3 R 型指令

5.3.1 R 型指令结构

R 格式指令为纯寄存器指令，所有的操作数（除移位外）均保存在寄存器中。Op 字段均为 0，使用 funct 字段区分指令。结构如图 5.1 所示：

图 5.1 R 型指令结构

5.3.2 R 型算术逻辑运算指令执行阶段的数据通路

如图 5.2 所示：

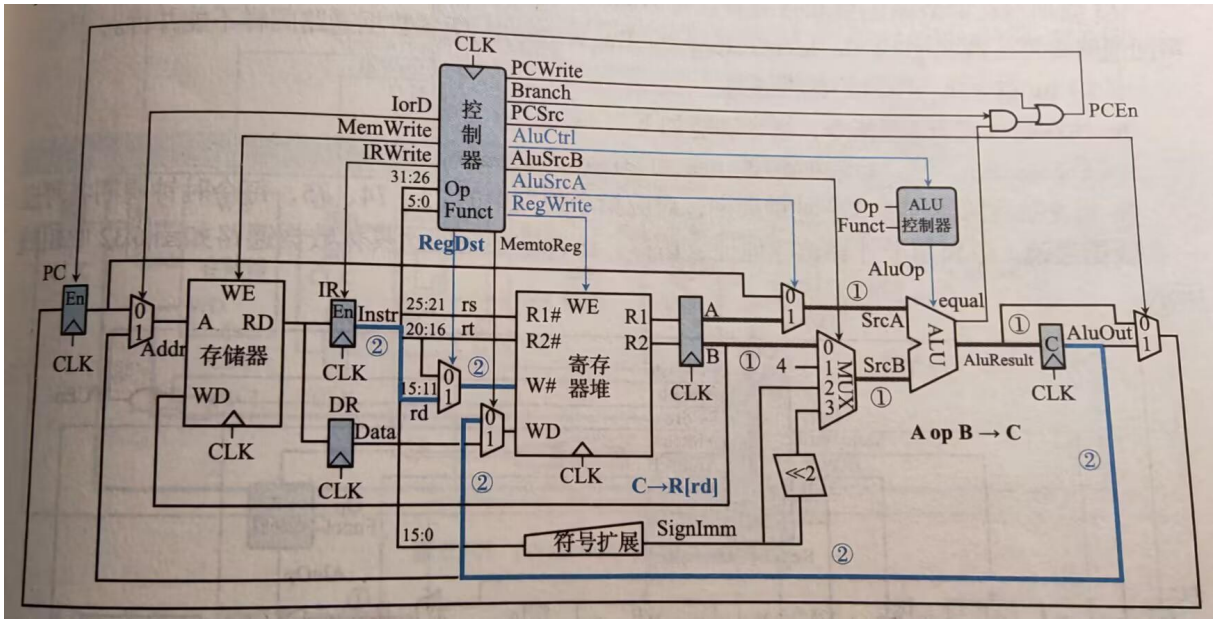


图 5.2 R 型算术逻辑运算指令执行阶段的数据通路

5.4 I 型指令

5.4.1 I 型指令结构

I 格式指令为带立即数的指令，最多使用两个寄存器，同时包括了 load/store 指令。使用 Op 字段区分指令[2]。结构如图 5.3 所示：

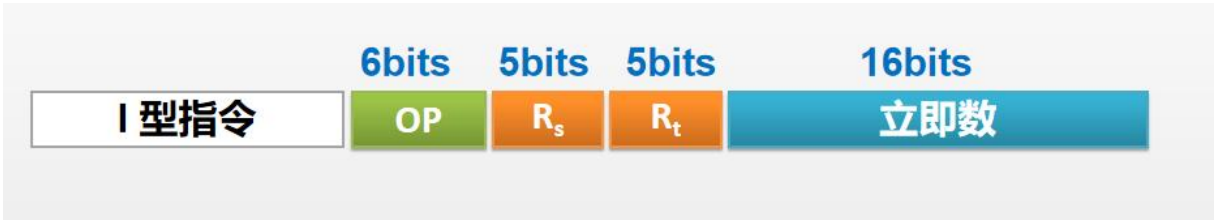


图 5.3 I 型指令结构

5.4.2 I 型算术逻辑运算指令执行阶段的数据通路

如图 5.4 所示：

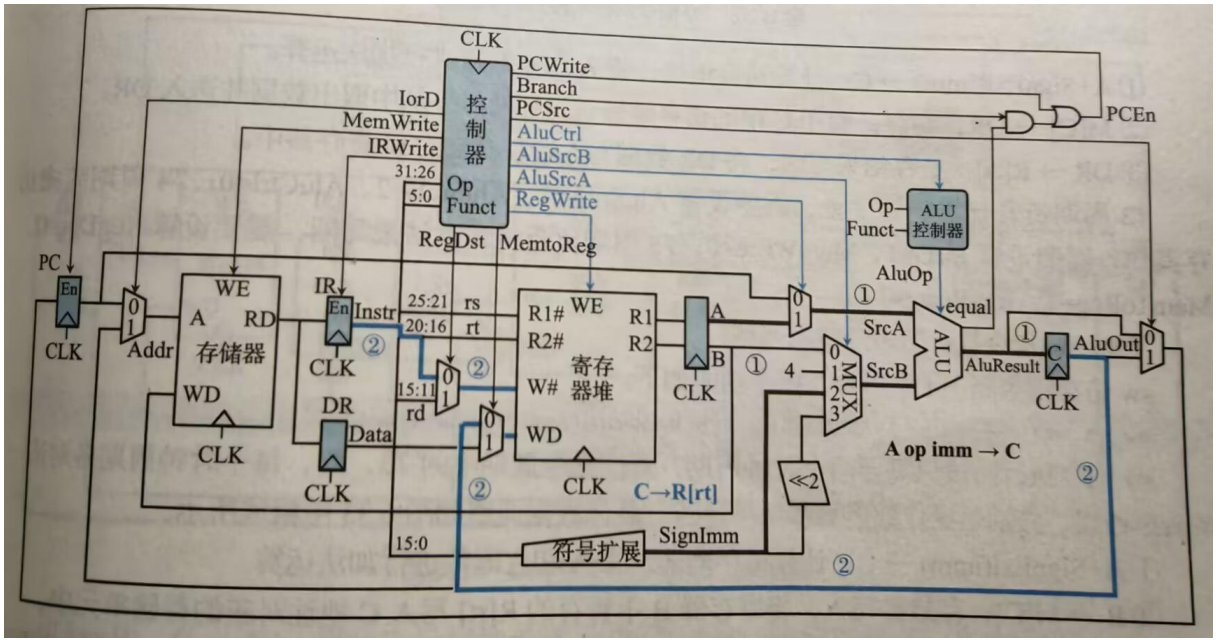


图 5.4 I 型算术逻辑运算指令执行阶段的数据通路

5.5 J 型指令

5.5.1 R 型指令结构

J 格式指令为长跳转指令，仅有一个立即数操作数。使用 Op 字段区分指令[4]。结构如图 5.5 所示：

图 5.5 J 型指令结构

该项目中并未设计 J 型指令。

6 功能测试

测试方法为使用一段 mips 汇编语言编写的冒泡排序代码转换成机器代码，写入存储器中，使处理器运行，运行完后可以看见在存储器中显示数据从大到小排序的结果。机器代码如图 6.1 所示：

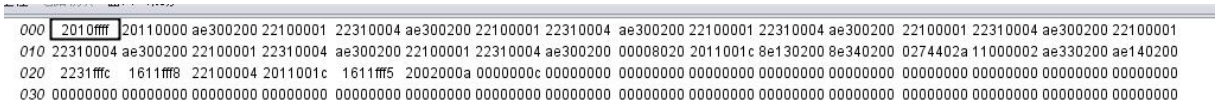


图 6.1 机器代码

排序完成后结果如图 6.2 所示：

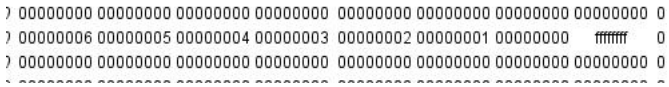


图 6.2 排序结果

汇编代码如图 6.3 所示：

```
sort_init:
 addi $s0,$0,-1
 addi $s1,$0,0
 sw $s0,128($s1)
 addi $s0,$s0,1
 addi $s1,$s1,4
 sw $s0,128($s1)
 addi $s0,$s0,1
 addi $s1,$s1,4
 sw $s0,128($s1)
 addi $s0,$s0,1
 addi $s1,$s1,4
 sw $s0,128($s1)
 addi $s0,$s0,1
 addi $s1,$s1,4
 sw $s0,128($s1)
 addi $s0,$s0,1
addi $s1,$s1,4
 sw $s0,128($s1)
 addi $s0,$s0,1
 addi $s1,$s1,4
 sw $s0,128($s1)
 addi $s0,$s0,1
 addi $s1,$s1,4
 sw $s0,128($s1)

 add $s0,$zero,$zero
 addi $s1,$zero,28
sort_loop:
 lw $s3,128($s0)
 lw $s4,128($s1)
 slt $t0,$s3,$s4
 beq $t0,$0,sort_next
 sw $s3, 128($s1)
 sw $s4, 128($s0)
sort_next:
 addi $s1, $s1, -4
 bne $s0, $s1, sort_loop


 addi $s0,$s0,4
 addi $s1,$zero,28
 bne $s0, $s1, sort_loop
 addi   $v0,$zero,10
 syscall
```

图 6.3 汇编代码

7 实训心得

本次实训的目标是使用 Logisim 平台设计一个处理器，我选择设计实现八条指令的多周期硬布线处理器，实训过程中还是遇到了不少困难。我将计算机组成原理的教材上有关部分反复看了好几遍，在网上查阅资料，观看教学视频并经常向同学请教，与同学交流。好不容易看懂了一些理论，动起手来拼装的时候仍然出了很多错误。虽然过程很曲折，但终于还是实现了。在这个过程中，我不仅增加了对 Logisim 平台使用的熟练度，也提升了自己查阅资料学习的能力。当然也加深了对 CPU 的认识等。我知道我暂时还只是学得了一些皮毛，我还没有更深入地去了解，但这次实训也激发了我的兴趣，我会更加努力，更加深入地去学习相关知识。这次的实训给我带来了很多收获。

参考文献

[1] 唐朔飞.计算机组成原理（第二版）[M].北京:高等教育出版社,2008：21-45，71.

[2] 谭志虎，秦磊华，胡迪青. 计算机组成原理实践教程[M]. 北京：清华大学出版社，2018：78-90.

[3] 谭志虎,秦磊华,胡迪青.计算机组成原理实践教程——从逻辑门到 CPU[M].北京:清华大学出版社,2018：45-67，78-90.

[4]王爱英.计算机组成原理与系统结构（第五版）[M].北京:清华大学出版社出版社,2001：45-51.