长沙理工大学

《系统能力综合实训》报告

MIPS 单周期处理器设计

学　　院	计算机与通信工程	专　　业	计算机科学与技术
班　　级	计科 2106 班	学　　号	202108061020
学生姓名	梅轩	指导教师	罗永红
课程成绩		完成日期	2024 年 1 月 5 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**	/	20	20	40
实训目标 2：**使用现代工具**	15	/	15	30
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：梅轩　　指导老师：罗永红

摘　要：在 Logisim 平台上，我们倾力打造了一款单周期硬布线的 24 条指令 CPU 处理器。此处理器能够准确执行 R 型、J 型和 I 型的全部 24 条指令。为了确保系统稳定运行，我构建了信号控制逻辑和地址转移逻辑。在设计过程中，我对系统的 24 条指令进行了详尽的测试。该系统已经实现了简单的冒泡排序功能。这一功能的实现证明了该处理器的运算能力。在阐述 CPU 的整体设计及关键模块的详细描述时，我深入研究了该 CPU 的工作原理和性能特点。这款处理器运行高效且稳定，通过实验验证，该 CPU 具有良好的运行能力，能够应对许多复杂的计算任务。无论是在速度还是稳定性方面，该处理器都展现出了良好的性能。

关键词：MIPS；多周期；处理器

The Design of Multi-Cycles Processor

Student name: Mei Xuan Advisor：Luo YongHong

Abstract：On the Logisim platform, we built a single-cycle hard-wired, 24-instruction CPU processor. The processor is capable of accurately executing all 24 instructions for R, J, and I. To ensure the stable operation of the system, I built the signal control logic and the address transfer logic. During the design process, I exhaustively tested the 24 instructions of the system. The system has implemented a simple bubble sorting function. The implementation of this feature is a testament to the computing power of the processor. When explaining the overall design of the CPU and the detailed description of the key modules, I delved into the working principle and performance characteristics of the CPU. This processor runs efficiently and stably, and has been experimentally proven to be capable of handling many complex computing tasks. Both in terms of speed and stability, the processor shows good performance.

Keywords：MIPS；multiple cycles；processors

目　录

1 引言

1.1 实训目的

本实训针对具体具体的 MIPS 指令系统，进行进行时序划分、构建主要功能部件、控制器、地址转移逻辑等，部件连接、控点连接，调试获得一个多周期 CPU，最后提交规范、有合理参考图书和文献资料的设计实训报告文档。

具体目标包括：

（1）查阅有关设计 CPU 的资料，并设计 CPU，培养学生通过研究分析文献来寻求解决复杂工程问题的途径；

（2）搭建并熟练使用 LogiSim 仿真平台，在平台上实现复杂的 CPU；

（3）了解 Logisim 仿真平台对计算机设计技术的影响，能缩短设计周期，降低设计成本，对节能环保具有积极意义；

（4）在设计过程中要对学生进行分组，每个组员都有不同的定位与责任，明确个人承担的任务；

（5）在设计过程中，需要平衡技术与经济、环境、人员等之间的关系。

1.2 选用平台

本次实训选用的软件平台为 Logisim，Logisim 是一种用于设计和模拟数字逻辑电路的教育工具。凭借其简单的工具栏界面和构建它们时的电路仿真，它非常简单，有助于学习与逻辑电路相关的最基本概念。由于能够从较小的子电路构建更大的电路，并通过鼠标拖动来绘制电线束，因此可以使用 Logisim 来设计和仿真本项目 MIPS 多周期处理器。其可以根据真值表或逻辑表达式，自动方便，避免手绘，生成电路相对美观。在 logisim 仿真平台上完成整个设计、测试流程，极大的降低了设计前期的检错成本，使用该仿真平台，极大程度上缩短了设计周期，降低了设计成本，对环境友好，减少碳排放，为碳中和目标贡献了一份力量[3]

1.3 需求分析

设计实现包含 24 条指令的单周期硬布线和地址转移逻辑，完成控制信号的转移，使 CPU 能够正常执行全部指令。使用排序程序进行测试，程序结束后在对应内存区域可以看到排序后的结果，并可以通过数码管使排序结果可视化，并进行单步调试测试不同指令。

为了实现这个目标，我们首先需要设计一个单周期硬布线的地址转移逻辑。这个逻辑需要包含 24 条指令，并且能够完成控制信号的转移，使 CPU 能够正常执行全部指令。

在设计过程中，我们需要考虑如何将这 24 条指令合理地分配到硬布线的各个部分，以保证 CPU 在单周期内能够正确地执行所有的指令。同时，我们还需要考虑到地址转移的问题，以确保 CPU 在执行指令时能够正确地访问到所需的内存地址[1]。

为了测试这个设计，我们可以使用一个排序程序进行测试。这个程序需要将一组数据按照从小到大的顺序进行排序，并在程序结束后在对应的内存区域输出排序后的结果。这样，我们就可以通过数码管将排序结果可视化，以便更好地观察程序的执行情况。

最后，我们还需要进行单步调试测试不同的指令。通过单步调试，我们可以逐条执行指令，并观察每条指令执行后的结果。这样，我们就可以发现并解决任何可能存在的问题，保证 CPU 能够正常地执行所有的指令。

2 指令周期设计分析

2.1 单周期 CPU

2.1.1 单周期说明

一个时钟周期完成一条指令，如果一个程序有多条指令，则时钟周期的时间根据执行时间最长的那条指令为主。执行一条指令就需要一个时钟周期则 CPI 为 1。通常指令的执行分为 5 个步骤：

(1) 取指令(IF)：根据程序计数器 pc 中的指令地址，从存储器中取出一条指令，同时，pc 根据指令字长度自动递增产生下一条指令所需要的指令地址，但遇到“地址转移”指令时，则控制器把“转移地址”送入 pc，当然得到的“地址”需要做些变换才送入 pc。

(2) 指令译码(ID)：对取指令操作中得到的指令进行分析并译码，确定这条指令需要完成的操作，从而产生相应的操作控制信号，用于驱动执行状态中的各种操作。

(3) 指令执行(EXE)：根据指令译码得到的操作控制信号，具体地执行指令动作，然后转移到结果写回状态。

(4) 存储器访问(MEM)：所有需要访问存储器的操作都将在这个步骤中执行，该步骤给出存储器的数据地址，把数据写入到存储器中数据地址所指定的存储单元或者从存储器中得到数据地址单元中的数据。

(5) 结果写回(WB)：指令执行的结果或者访问存储器中得到的数据写回相应的目的寄存器中。

2.1.2 单周期 CPU 特点

（1）所有指令均在一个时钟周期内完成，CPI=1

（2）系统性能取决于最慢的指令，时钟周期过长

（3）设计简单

2.1.3 单周期数据通路设计

因为处理器在一个周期内只能操作每个部件一次，而在一个周期内不可能对一个单端口存储区进行两次存取，所以数据寄存器和指令寄存器需要分开，采用**哈佛结构**。同时需要注意，单周期 CPU 不需**要** IR、DR 等寄存器锁存数据，运算器和 PC 累加器分开。

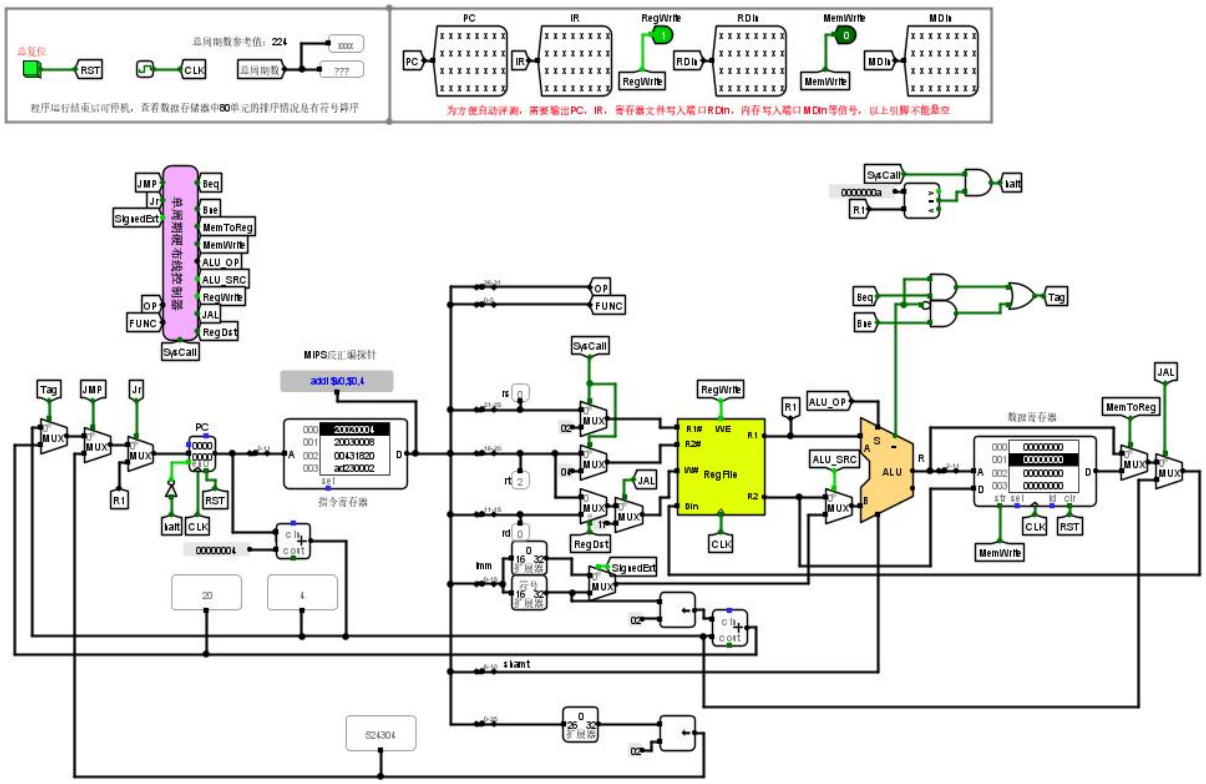

图 2.1 单周期数据通路

3 控制器设计分析

3.1 硬布线控制器

3.1.1 硬布线控制器的基本思想

硬布线控制器的基本思想是将指令的执行控制逻辑固定在硬件电路中，通过预先设计和连接电路来实现指令的解码和控制信号的生成。

硬布线控制器的设计包括以下几个基本步骤：

指令解码：硬布线控制器需要解析指令，并识别出指令中的操作码以确定所需执行的操作。这通常涉及到对指令进行位分解和位操作。

控制信号生成：根据指令解码的结果，硬布线控制器需要生成相应的控制信号，以控制 CPU 中的各个功能模块的操作。这些控制信号可能包括数据通路中寄存器读写使能信号、ALU 操作控制信号、内存访问控制信号等。

时序控制：硬布线控制器需要确保控制信号按照正确的时序顺序生成和传递。这通常通过时钟信号和时序逻辑电路来实现。

状态更新：硬布线控制器还需要根据指令执行的结果来更新 CPU 的状态。例如，更新程序计数器 PC 以指向下一条指令的地址，或者更新标志寄存器以反映运算结果的状态。

硬布线控制器的优点是具有较高的执行效率和速度，因为控制逻辑直接实现在硬件电路中，无需额外的指令解释和执行时间。然而，硬布线控制器的缺点是其设计复杂且不灵活，不容易进行修改和扩展。

3.1.2 硬布线控制器的特点

高效性：硬布线控制器将指令的执行控制逻辑固定在硬件电路中，无需额外的指令解释和执行时间，因此具有较高的执行效率和速度。

稳定性：由于硬布线控制器的控制逻辑直接实现在硬件电路中，因此不会受到软件错误或异常情况的影响，具有较高的稳定性。

可靠性：硬布线控制器的电路结构比较简单，不容易出现故障，因此具有较高的可靠性。

难以修改：由于硬布线控制器的设计是预先固定的，不容易进行修改和扩展，因此对于需要频繁修改或升级的应用场景，不太适合采用硬布线控制器。

硬件成本高：硬布线控制器需要预先设计和连接电路，因此其硬件成本相对较高。

4 公有器件设计

4.1 寄存器堆

MIPS 有 32 个通用寄存器（0-31），寄存器堆有一个写端口，两个读端口。各寄存器的功能及汇编程序中使用约定如下：

下表描述 32 个通用寄存器的别名和用途：

表 4-1 32 个寄存器

REGISTER	NAME	USAGE
$0	$zero	常量0(constant value 0)
$1	$at	保留给汇编器(Reserved for assembler)
$2-$3	$v0-$v1	函数调用返回值(values for results and expression evaluation)
$4-$7	$a0-$a3	函数调用参数(arguments)
$8-$15	$t0-$t7	暂时的(或随便用的)
$16-$23	$s0-$s7	保存的(或如果用，需要SAVE/RESTORE的)(saved)
$24-$25	$t8-$t9	暂时的(或随便用的)
$28	$gp	全局指针(Global Pointer)
$29	$sp	堆栈指针(Stack Pointer)
$30	$fp	帧指针(Frame Pointer)
$31	$ra	返回地址(return address)

注意：0 号寄存器里面恒 0

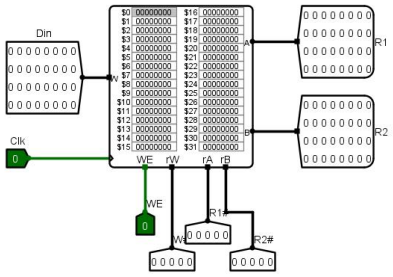

图 4.1 寄存器堆

封装以后的寄存器堆：

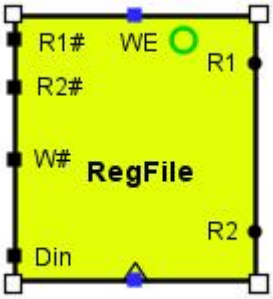

图 4.2 封装的寄存器堆

4.2 ALU 设计

4.2.1 ALU 设计

ALU 包含以下运算：

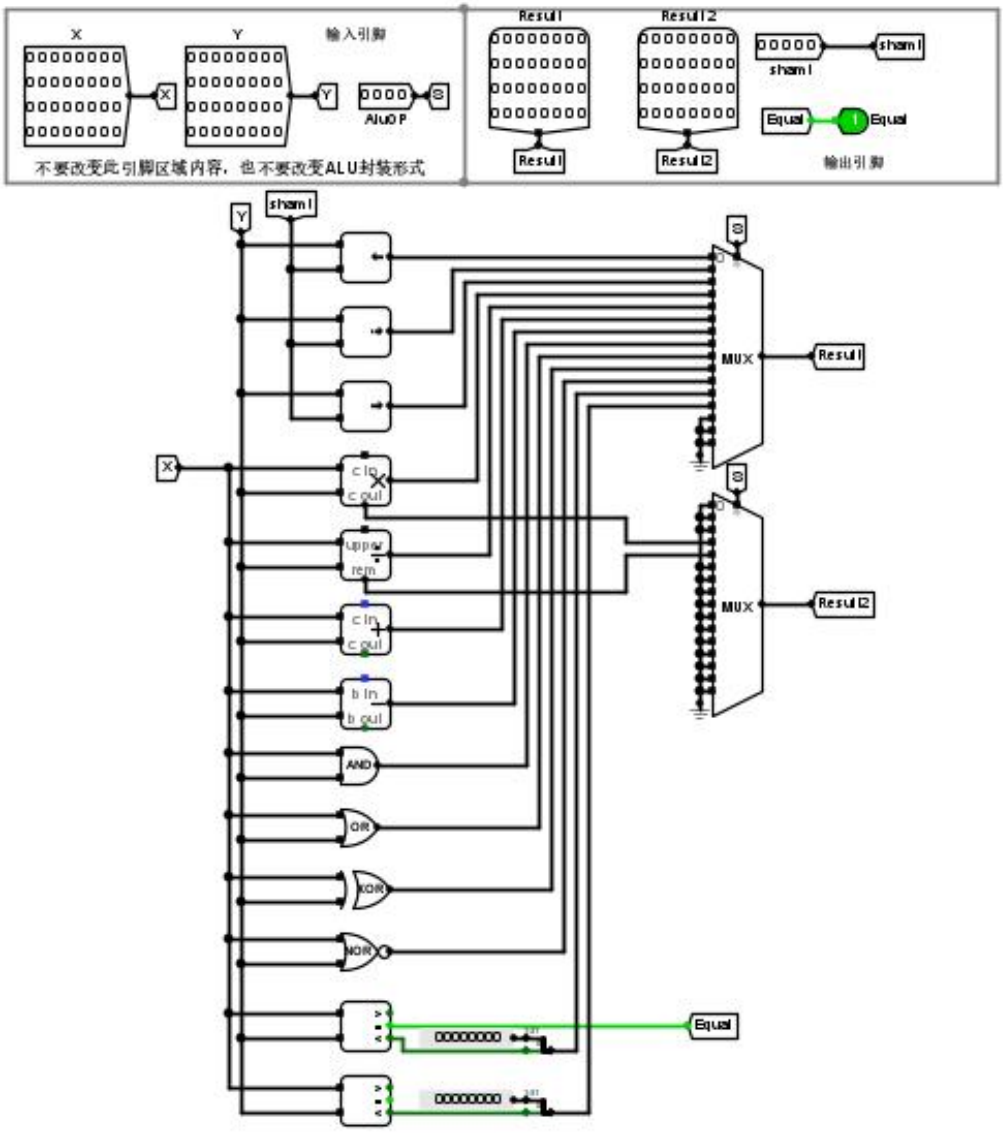

图 4.3 ALU 设计

4.2.2 ALU 的指令设计

根据编码解析出运算信号[5]：

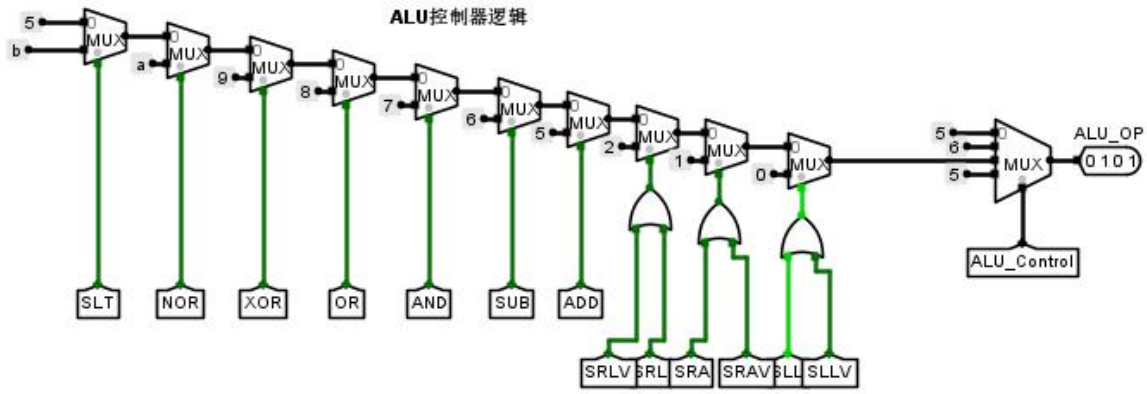

图 4.4 ALU 指令设计

4.3 指令设计

包括一定的新指令的设计：

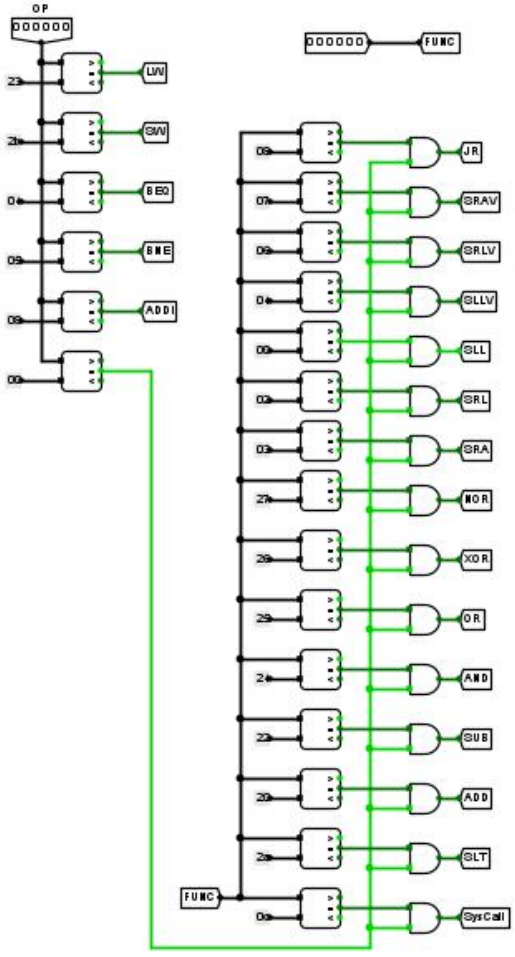

图 4.5 指令设计编码

5 单周期 CPU 设计

5.1 单周期硬布线 MIPS 数据通路

因为是单周期 CPU，所有指令均在一个时钟周期内完成操作，所以指令与数据分别采用 ROM 及 RAM 存储[2]。

PC 的来源有四种：①.顺序执行 PC=PC+4

②.beq 或者 bne 指令有效时，PC = PC + 4 + (SignExtImm << 2)

③.J 或者 Jal 指令有效时，取 PC+4 的高四位再加 address<<2

④.JR 指令有效时，PC=Reg[rs]

由于指令存储器存储字长为 4B，而 PC 为指令字节地址，非指令字地址，故舍弃最低两位，将 2~11 位送入存储器地址端。

指令 IR 分线后送入单周期硬布线控制器译码，同时下一指令的地址通过多路选择器送入程序计数器的数据端口，等待时钟上升沿锁存[4]。

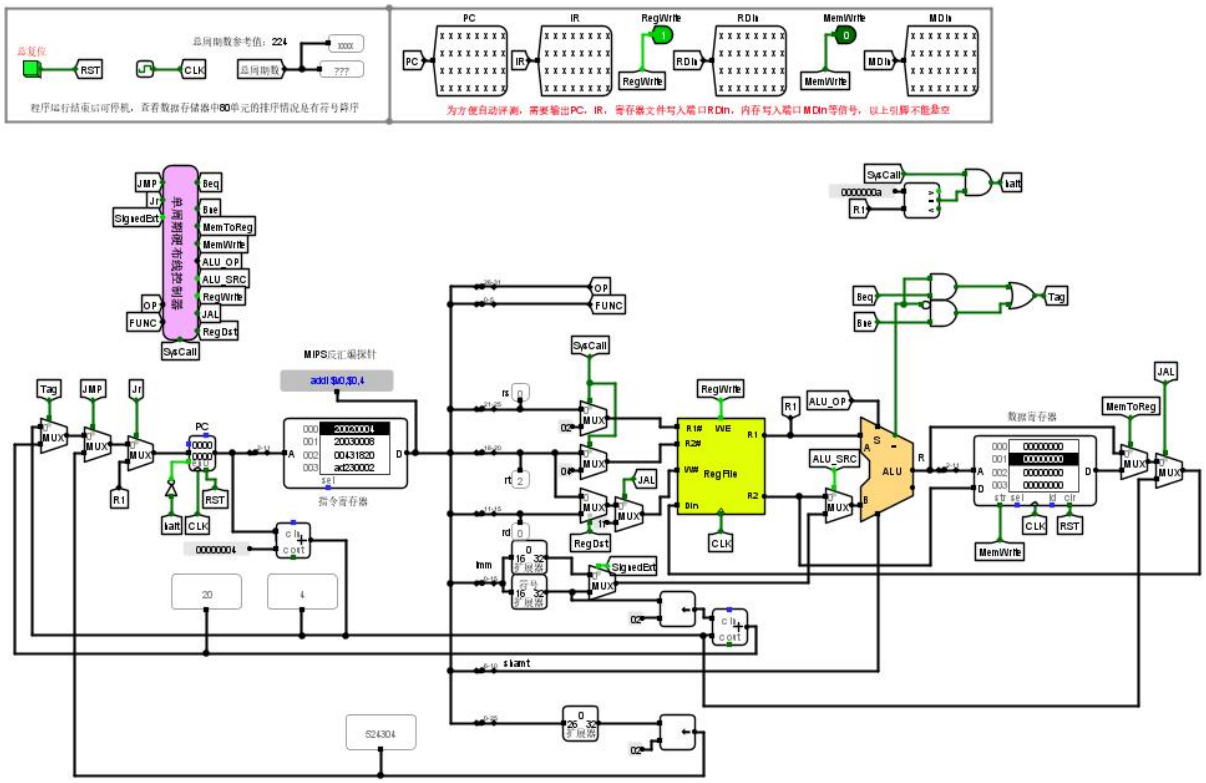

图 5.1 单周期硬布线数据通路

5.2 单周期硬布线控制器

根据 OP 字段和 FUNC 字段去分析指令，进而使该指令对应的控制信号有效。

表 5-1 控制信号

控制信号	说明	产生条件（=1）
RegWrite	寄存器写使能	寄存器写入
MemWrite	内存写使能	sw
AluOP	运算器操作控制符（4位）	R型指令依据 funct 字段
MenToReg	寄存器写入数据来自主存	lw
RegDst	写入寄存器编号选择	R型指令
AluSrcB	运算器B端输入选择	lw、sw、立即数运算指令
SignedExt	立即数符号扩展	addi、addiu、slti
JR	寄存器跳转指令信号	jr
JAL	JAL指令信号	jal
JMP	无条件转移控制信号	j、jal、jr
Beq	Beq指令信号	beq
Bne	Bne指令信号	bne
Syscall	系统调用指令信号	syscall

以下是指令信号生成：

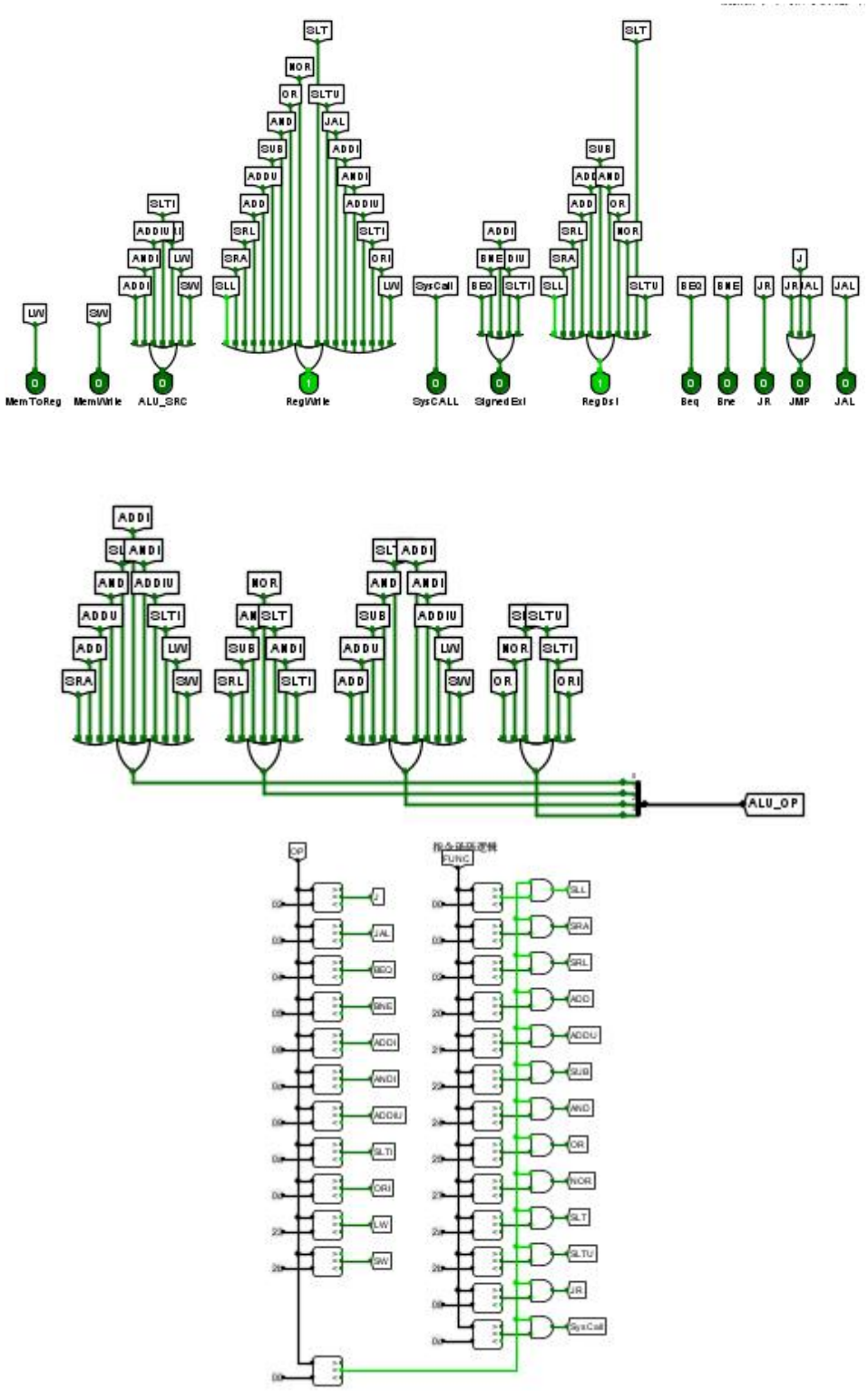

图 5.2 单周期硬布线控制器

硬布线控制器又由两部分组成，分为**运算控制器**和**控制信号生成**。

这两部分我都采用填写 EXCEL 表格，进而自动生成电路。

#	指令	OpCode (十进制)	FUNCT (十进制)	ALU_OP	MemtoReg	MemWrite	ALU_SRC	RegWrite	SYSCALL	SignedExt	RegDst	BEQ	BNE	JR	JMP	JAL
1	SLL	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
2	SRA	0	3	1	0	0	0	1	0	0	1	0	0	0	0	0
3	SRL	0	2	2	0	0	0	1	0	0	1	0	0	0	0	0
4	ADD	0	32	5	0	0	0	1	0	0	1	0	0	0	0	0
5	ADDU	0	33	5	0	0	0	1	0	0	1	0	0	0	0	0
6	SUB	0	34	6	0	0	0	1	0	0	1	0	0	0	0	0
7	AND	0	36	7	0	0	0	1	0	0	1	0	0	0	0	0
8	OR	0	37	8	0	0	0	1	0	0	1	0	0	0	0	0
9	NOR	0	39	10	0	0	0	1	0	0	1	0	0	0	0	0
10	SLT	0	42	11	0	0	0	1	0	0	1	0	0	0	0	0
11	SLTU	0	43	12	0	0	0	1	0	0	1	0	0	0	0	0
12	JR	0	8	x	0	0	0	0	0	0	0	0	0	1	1	0
13	SYSCALL	0	12	x	0	0	0	0	1	0	0	0	0	0	0	0
14	J	2	X	x	0	0	0	0	0	0	0	0	0	0	1	0
15	JAL	3	X	x	0	0	0	1	0	0	0	0	0	0	1	1
16	BEQ	4	X	x	0	0	0	0	0	0	0	1	0	0	0	0
17	BNE	5	X	x	0	0	0	0	0	0	0	0	1	0	0	0
18	ADDI	8	X	5	0	0	1	1	0	1	0	0	0	0	0	0
19	ANDI	12	X	7	0	0	1	1	0	0	0	0	0	0	0	0
20	ADDIU	9	X	5	0	0	1	1	0	1	0	0	0	0	0	0
21	SLTI	10	X	11	0	0	1	1	0	1	0	0	0	0	0	0
22	ORI	13	X	8	0	0	1	1	0	0	0	0	0	0	0	0
23	LW	35	X	5	1	0	1	1	0	0	0	0	0	0	0	0
24	SW	43	X	5	0	1	1	0	0	0	0	0	0	0	0	0

图 5.3 控制器表达式自动生成图

因为有 13 种 ALU 运算，根据六位的 OP 字段和六位的 FUNC 字段，可以进行区分，例如：ALU_OP=0000,表示逻辑左移。

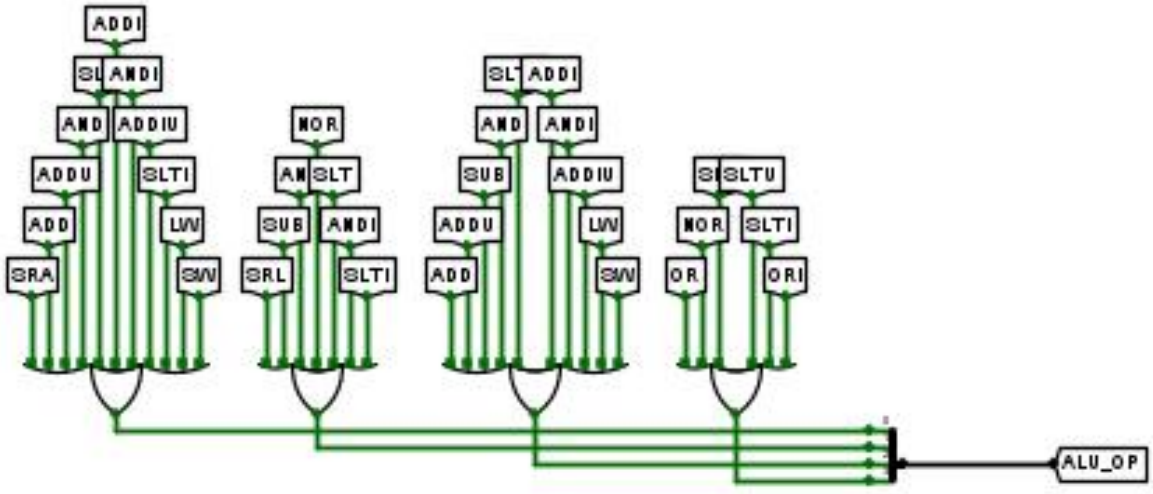

图 5.4 运算控制器

根据 OP 字段和 FUNC 字段区分指令，进而产生对应的控制信号，去控制数据通路中的控点或者器件的使能信号。

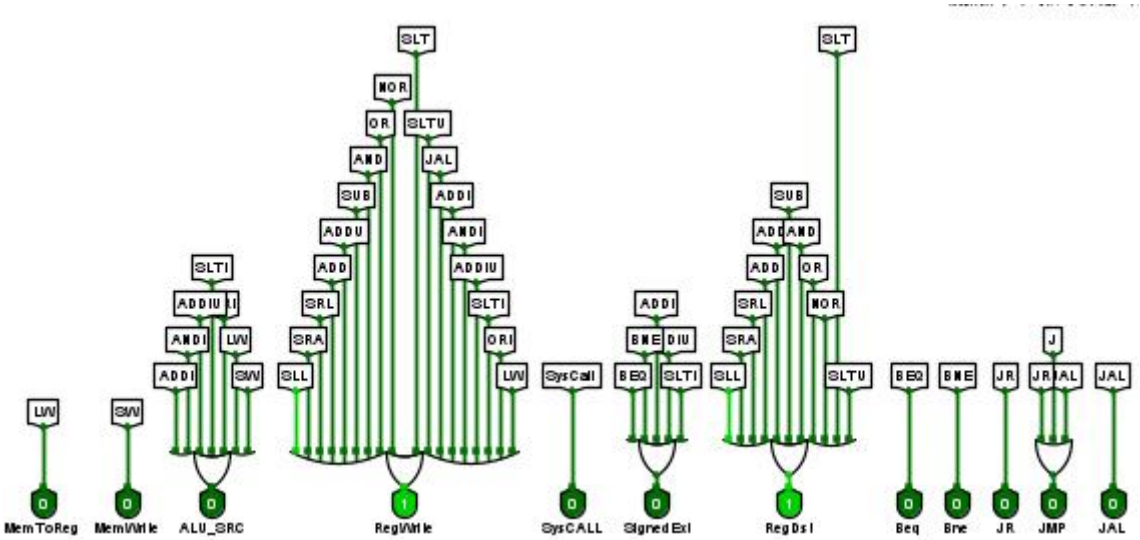

图 5.5 控制信号生成电路

6 测试程序

6.1 个人测试

6.1.1 syscall 功能

用于停机，指令 16 进制：0x0000000c，现象如下图所示：

图 6.1 停机指令图

6.1.2 单步调试

在 Mars 里面编写指令，包含 R、I、J 三种类型

```
xitong.asm
1  addi $v0,$0,4
2  addi $v1,$0,8
3  tag:
4  add $v1,$v0,$v1
5  sw $v1,2($t1)
6  jal tag
```

图 6.3 指令编写图

图 6.4 指令寄存器内容

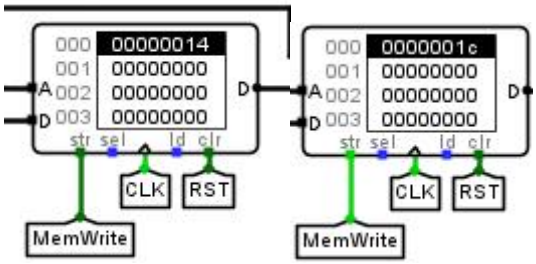

图 6.5 数据寄存器结果

6.2 互相联调测试

本次实训，为了更好的测试项目，还接收了龙琪同学的联调测试，编写程序进行了冒泡排序测试，测试的结果无误。

指令寄存器：

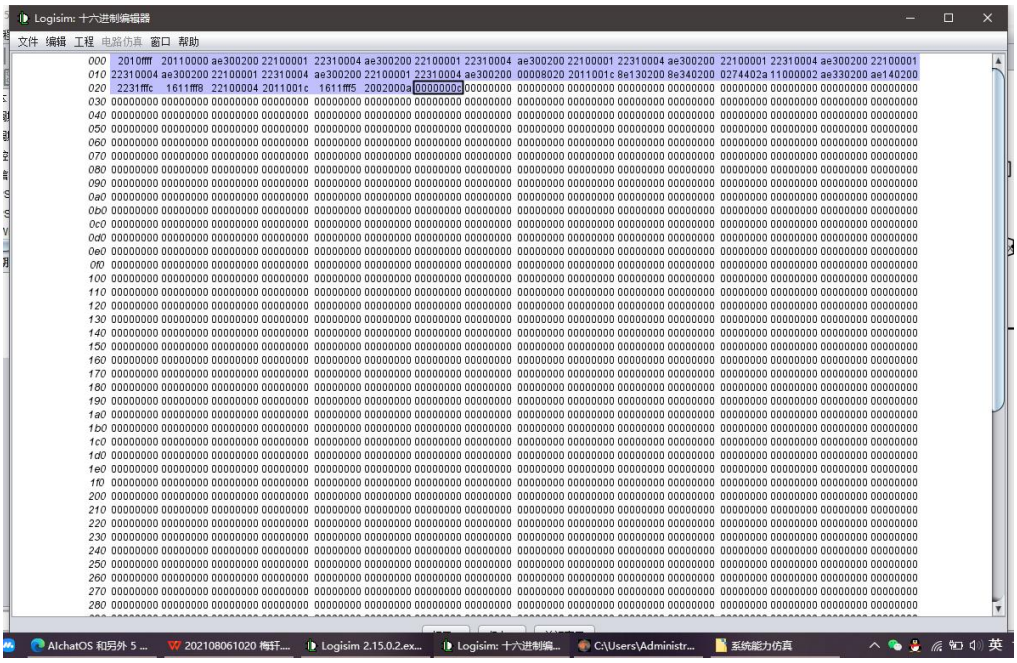

图 6.8 指令寄存器

6.2.1 单周期硬布线测试

冒泡排序：在 0x80~0x87 号存储单元放入数据，并排序，排序结果如下图所示：

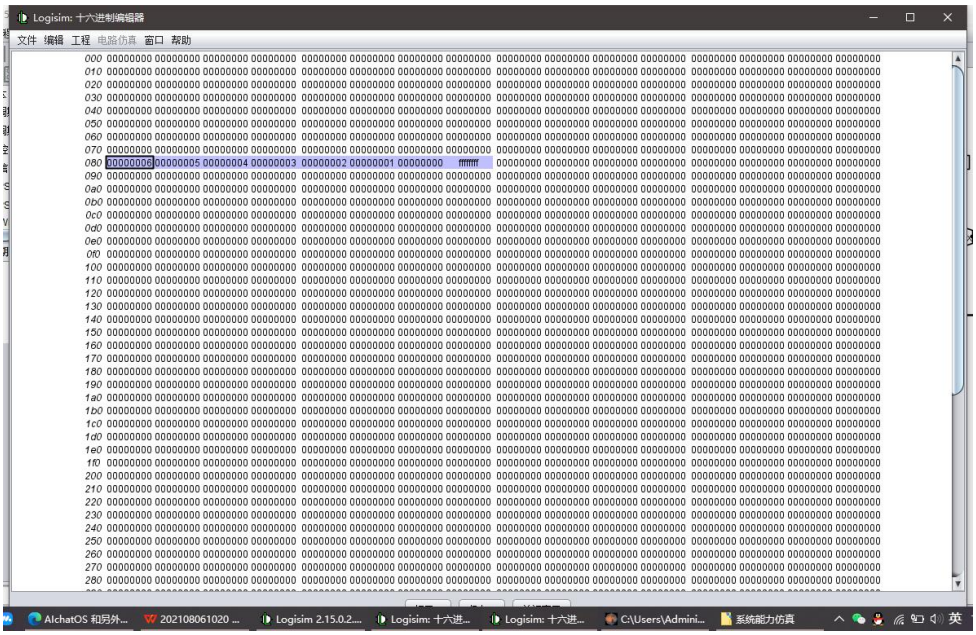

图 6.9 单周期硬布线测试

7 总结

在初次接触实训课题——自主设计一个 CPU 时，由于缺乏教师的指导，我感到困惑和不知所措，不清楚应该如何展开工作。为了逐步入门，我决定从最简单的 CPU——8 指令单周期硬布线开始学习。我通过在线课程、查阅 CSDN 博客等途径获取资料，并亲自动手进行实践操作。

在完成 8 指令单周期 CPU 并成功运行排序程序后，我将其扩展为 24 指令单周期硬布线 CPU。然而，这一过程中我意识到，扩展并非易事。首先，需要对新增的指令数据通路有深入的了解；其次，需要对原有结构进行调整，增加控制点以满足更多指令的需求。完成设计工作后，测试环节同样重要。在测试过程中，我发现了一些细微的错误。通过单步测试与单条指令测试，我最终修正了所有的指令问题。

通过此次实训，我对单周期、多周期 MIPS 的设计有了更深入的了解。越深入研究底层原理，我越能感受到计算机技术的神奇之处。尽管底层由最基本的电路组合而成，但其呈现出的智能化特性令人叹为观止。我坚信，我应该继续努力学习，不断提升自己的技术水平。

参考文献

[1]吴继明,曾碧卿. 一种高效的 CPU 设计方法及其在计算机组成原理课程中应用[J]. 实验室研究与探索,2018,37(09):147-153.

[2]方灿.基于 MIPS 指令集的流水线 CPU 设计与验证[D].武汉：华中科技大学,2021.

[3]何杰,张磊,齐悦,姚琳,单柳昊. 流水线 CPU 设计关键技术虚拟仿真实验教学平台构建[J]. 中国现代教育装备,2022,2022(15):49-52.

[4]蔡晓燕，袁春风，张泽生. MIPS 体系结构单周期 CPU 的设计 [D]. 天津：天津科技大学

[5]柳成，荣静. 基于 MIPS 架构的多周期 CPU 设计[D]. 吉林：北华大学