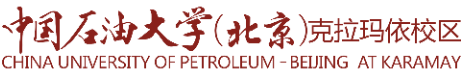
实 验 报 告





课 程 ：计算机组成原理

实验名称：单总线现代时序CPU设计实验

学 院 ：石油学院

专 业 ：数据科学与大数据技术

学 号 ：2023015509

姓 名 ：胡林森

2024年秋季学期

**目 录**

[一、实验目的 3](#_Toc183771882)

[二、实验内容 3](#_Toc183771883)

[三、实验步骤 3](#_Toc183771884)

[ 1.设计MIPS指令译码器 3](#_Toc183771885)

[ 2.设计微程序控制器 4](#_Toc183771886)

[四、实验结果 5](#_Toc183771887)

[ 1.设计MIPS指令译码器 5](#_Toc183771888)

[连接译码器 5](#_Toc183771889)

[单总线结构 MIPS CPU 结构的微程序入口地址产生逻辑 6](#_Toc183771890)

[ 2.设计微程序控制器 7](#_Toc183771891)

[MIPS 指令微程序编写（结合表格完成） 7](#_Toc183771892)

[连接电路并将生成的十六进制微指令复制到控制存储器中。 7](#_Toc183771893)

[微程序条件判断测试产生逻辑 8](#_Toc183771894)

[ 3.单总线CPU（微程序）电路及验证 9](#_Toc183771895)

[整体电路 9](#_Toc183771896)

[验证 10](#_Toc183771897)

[1.向主存中载入数据sort-5.hex，便于进行排序算法验证 10](#_Toc183771898)

[2.微程序导入 10](#_Toc183771899)

[3.执行 10](#_Toc183771900)

[4.结果 11](#_Toc183771901)

[五、问题和感想 11](#_Toc183771902)

# 一、实验目的

掌握现代时序硬布线控制器、微程序控制器设计的基本原理，能在Logisim平台中基于单总线结构实现支持5条MIPS指令的现代时序处理器。

# 二、实验内容

本实验将依次设计：

① 指令译码器

② 地址转移逻辑

③ 微程序控制存储器

④ FSM状态机

等电路，以逐步实现：

⑤ 微程序控制器

⑥ 硬布线控制器

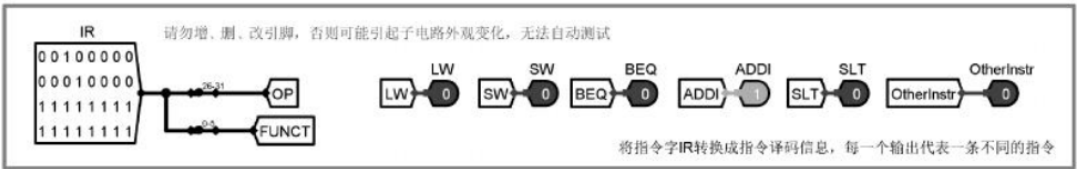
# 三、实验步骤

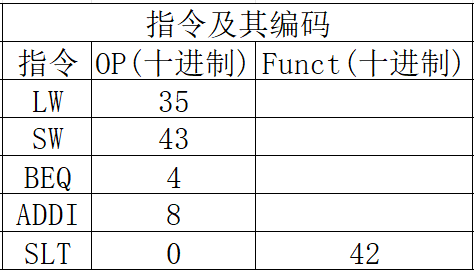
## 1.设计MIPS指令译码器

• MIPS指令译码器负责将定长的32位MIPS指令字翻译为指令译码信号，以此识别指令功能，具体电路封装与引脚功能描述如表5所示。



• 指令译码器电路框架如下图所示。



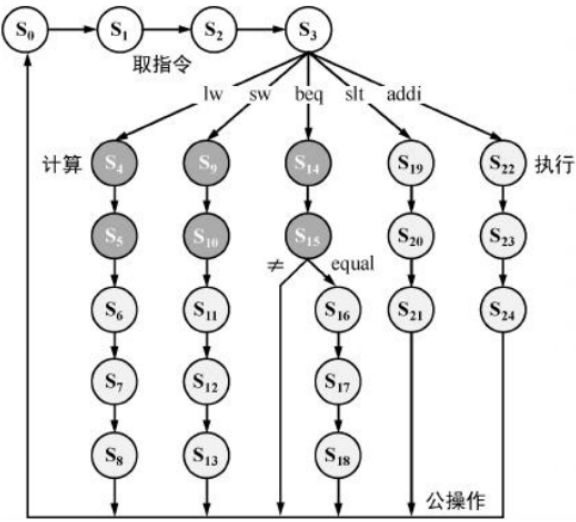
• 五种指令（LW、SW、BEQ、ADDI、SLT）及其操作编码的方式如右图所示。根据对应指令的OP字段和Funct字段的值，利用Logisim中的比较器组件和基本逻辑门电路输出各指令译码信号。

## 2.设计微程序控制器

• 对于图5.1所示的单总线数据通路，采用现代时序分析5条MIPS指令的指令周期及数据通路，可以得到图5.18所示的指令执行状态转换图。

• 图5.18中一个状态对应一个时钟周期，微操作控制信号的值仅与现态有关。控制存储器中的微指令可以与状态转换图中的状态一一对应，状态的编号值可以转换成微指令地址；而某一状态需要给出的微操作控制信号可以映射到对应微指令操作控制字段中的控制信号位；状态之间的切换关系可以对应微指令之间的执行顺序，用于设置判别测试字段和下址字段

结合图5.18，参考教材第6章的原理设计微指令格式，构建5条MIPS指令的微程序。本实验只需要用slt指令替换原有的add指令



32个寄存器；按照字节编址。32个寄存器集成一个小容量的高速存储器，它们的编号和基本用途如下表所示：

|  |  |
| --- | --- |
| 寄存器编号 | 用途 |
| $0 | 总为0 |
| $1 | 留作汇编器生成一些合成指令 |
| $2、$3 | 存放子程序返回值 |
| $4-$ 7 | 调用子程序时，传输前4个非浮点参数 |
| 寄存器编号 | 用途 |
| $8-$15 | 临时寄存器，使子程序使用时不用存储和恢复 |
| $16-$23 | 存储临时子程序寄存器变量 |
| $24、$25 | 临时寄存器，使子程序使用时不用存储和恢复 |
| $26、$27 | 异常处理时使用 |
| $28 | 存储全局指针 |
| $29 | 存储堆栈指针 |
| $30 | 子程序用来做堆栈帧指针 |
| $31 | 存储子程序返回地址 |

• 完成“单总线MIPS微程序地址转移逻辑自动动生成.xlsx”电子表格文件。

• 完成“单总线MIPS现代时序微程序控制器设计.x1sx”

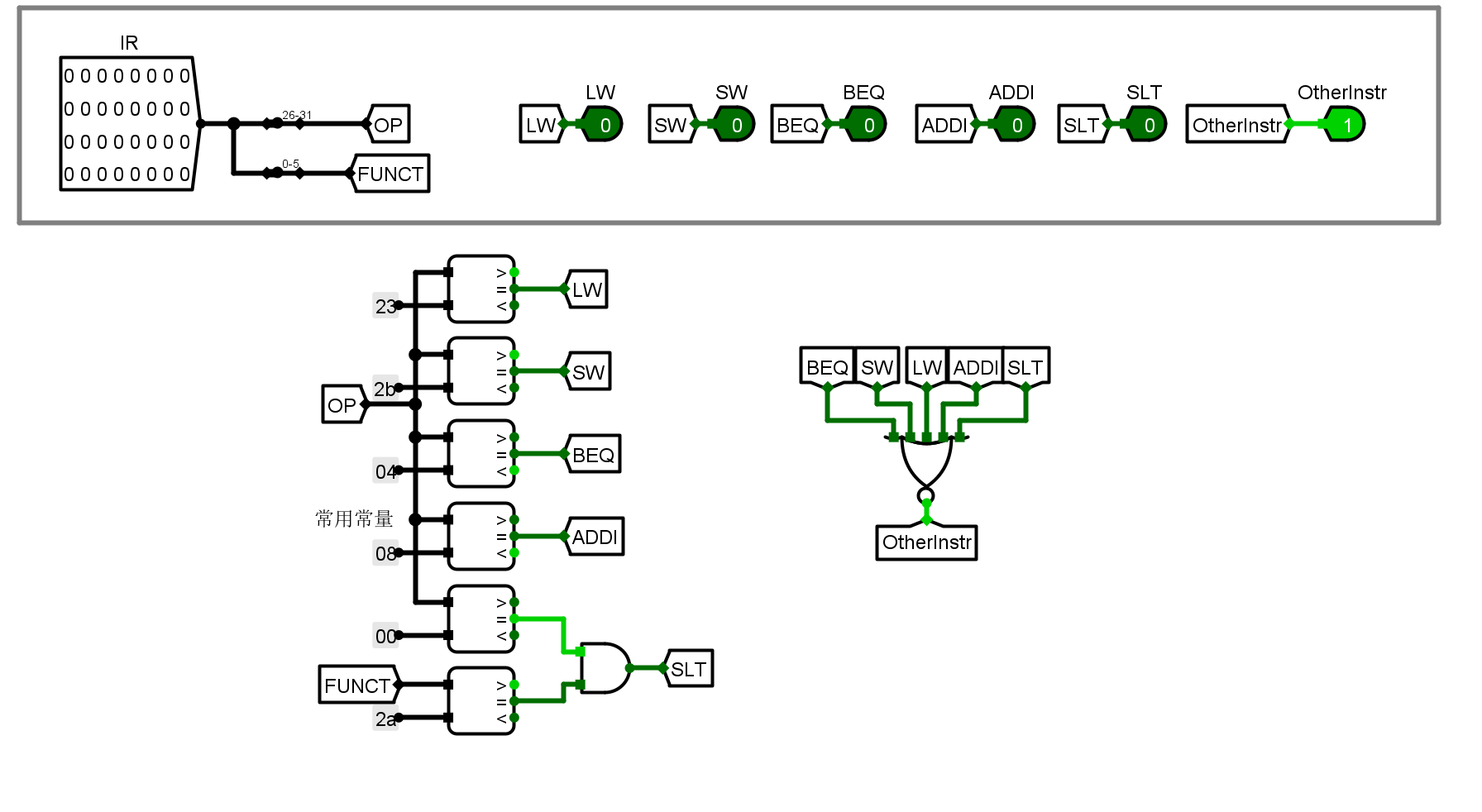
• 连接其他电路

• 调试

# 四、实验结果

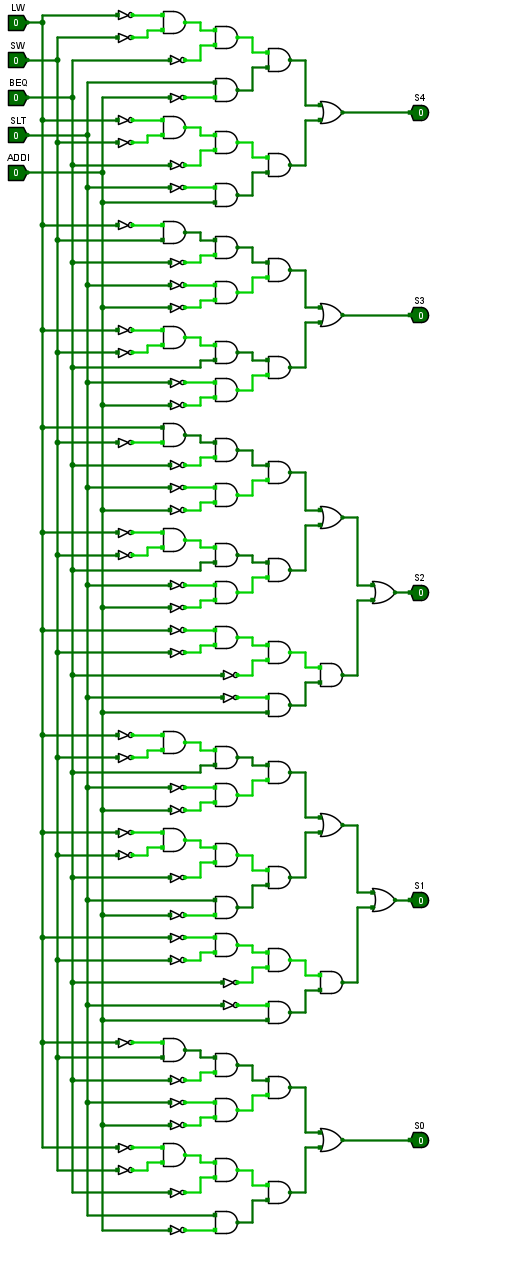
## 1.设计MIPS指令译码器

### 连接译码器



### 单总线结构 MIPS CPU 结构的微程序入口地址产生逻辑

1、单总线结构 MIPS CPU 结构的微程序入口地址产生逻辑图



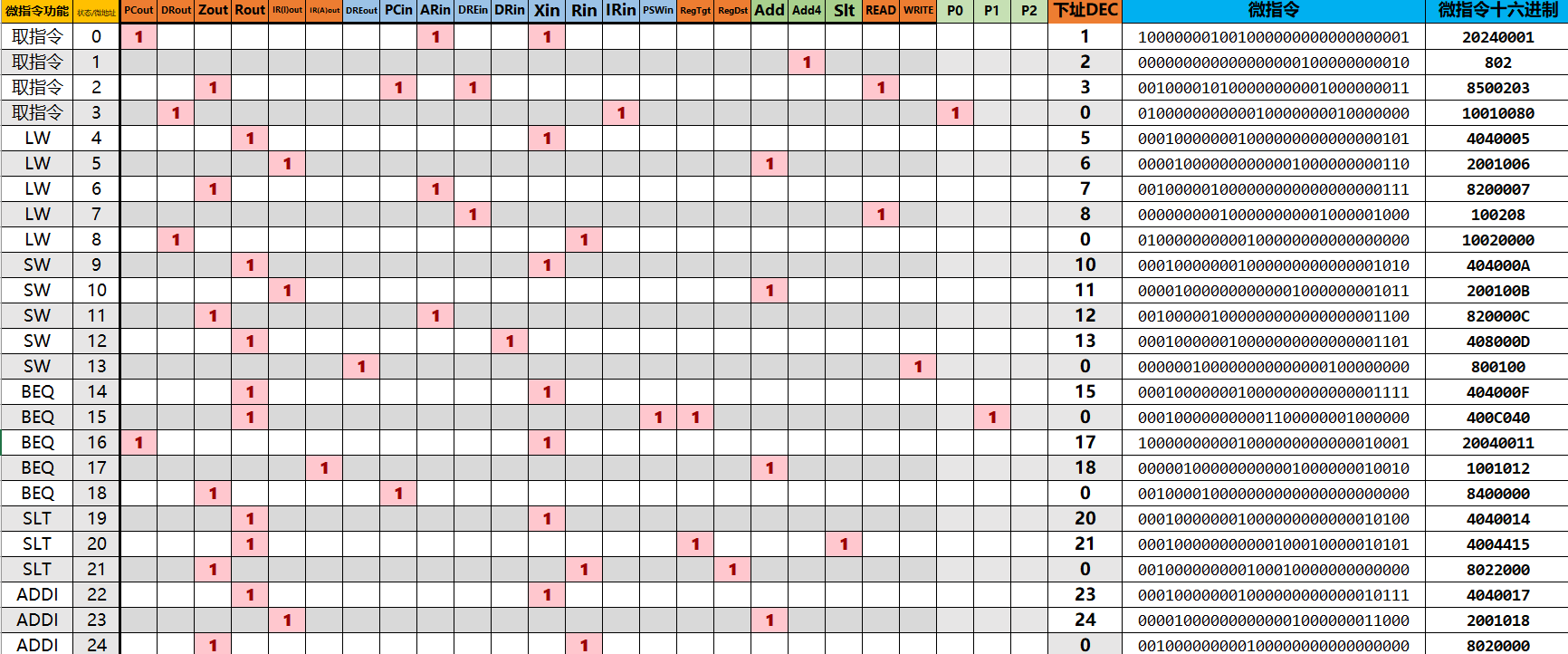
2、单总线结构 MIPS CPU 结构的微程序入口地址表格

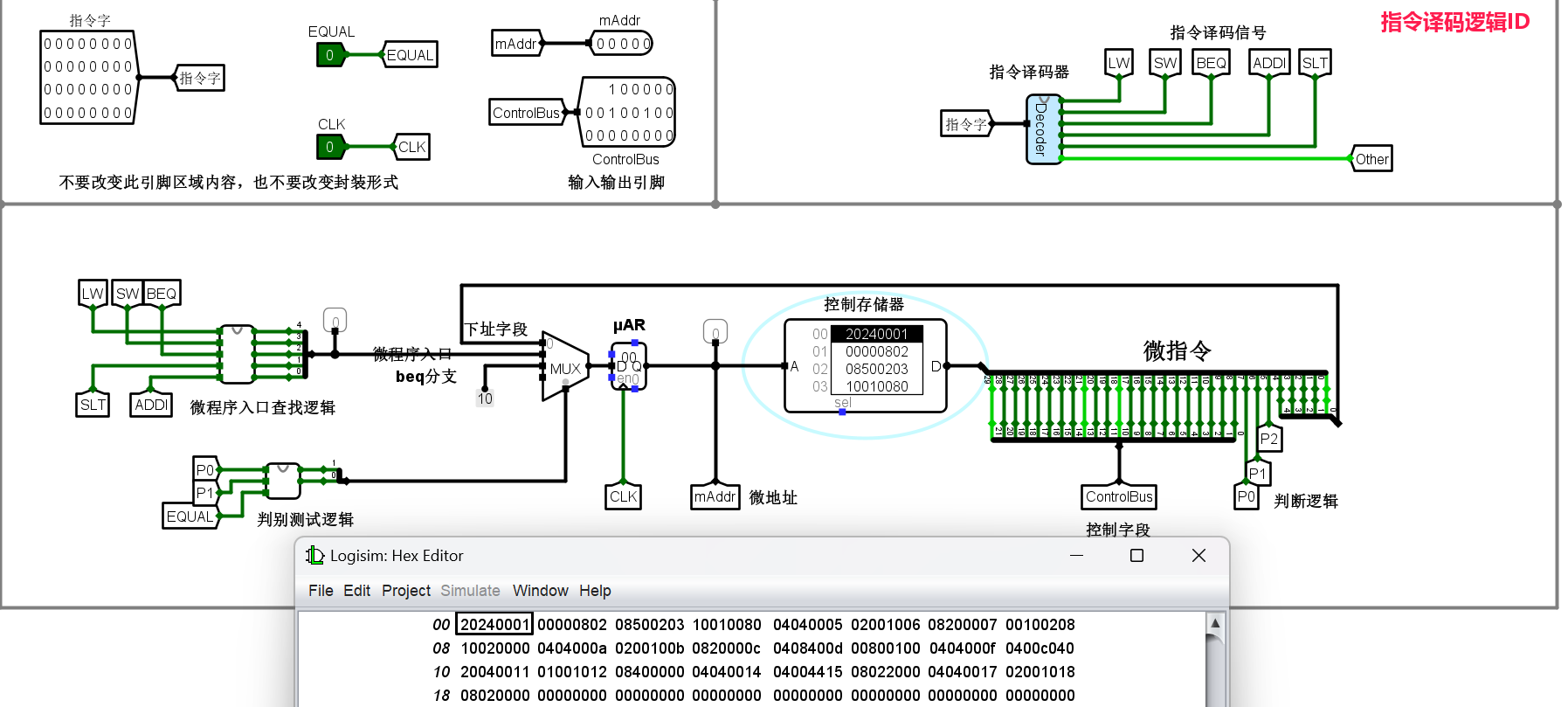
将公式复制logsim中，可自动生成电路。

## 2.设计微程序控制器

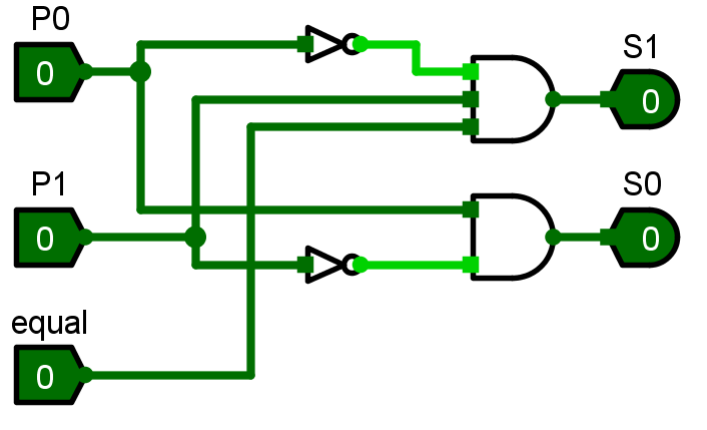
### MIPS 指令微程序编写（结合表格完成）

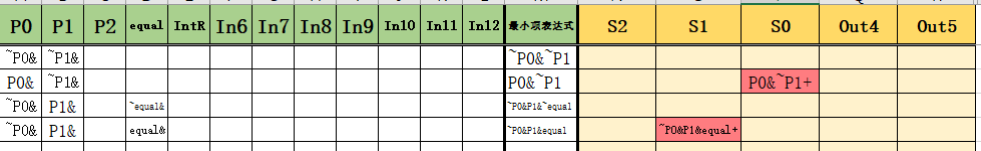
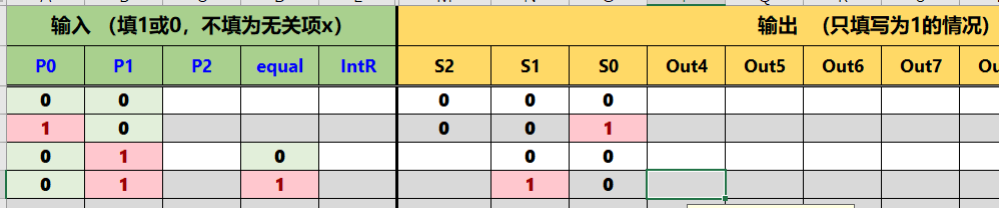


### 连接电路并将生成的十六进制微指令复制到控制存储器中。



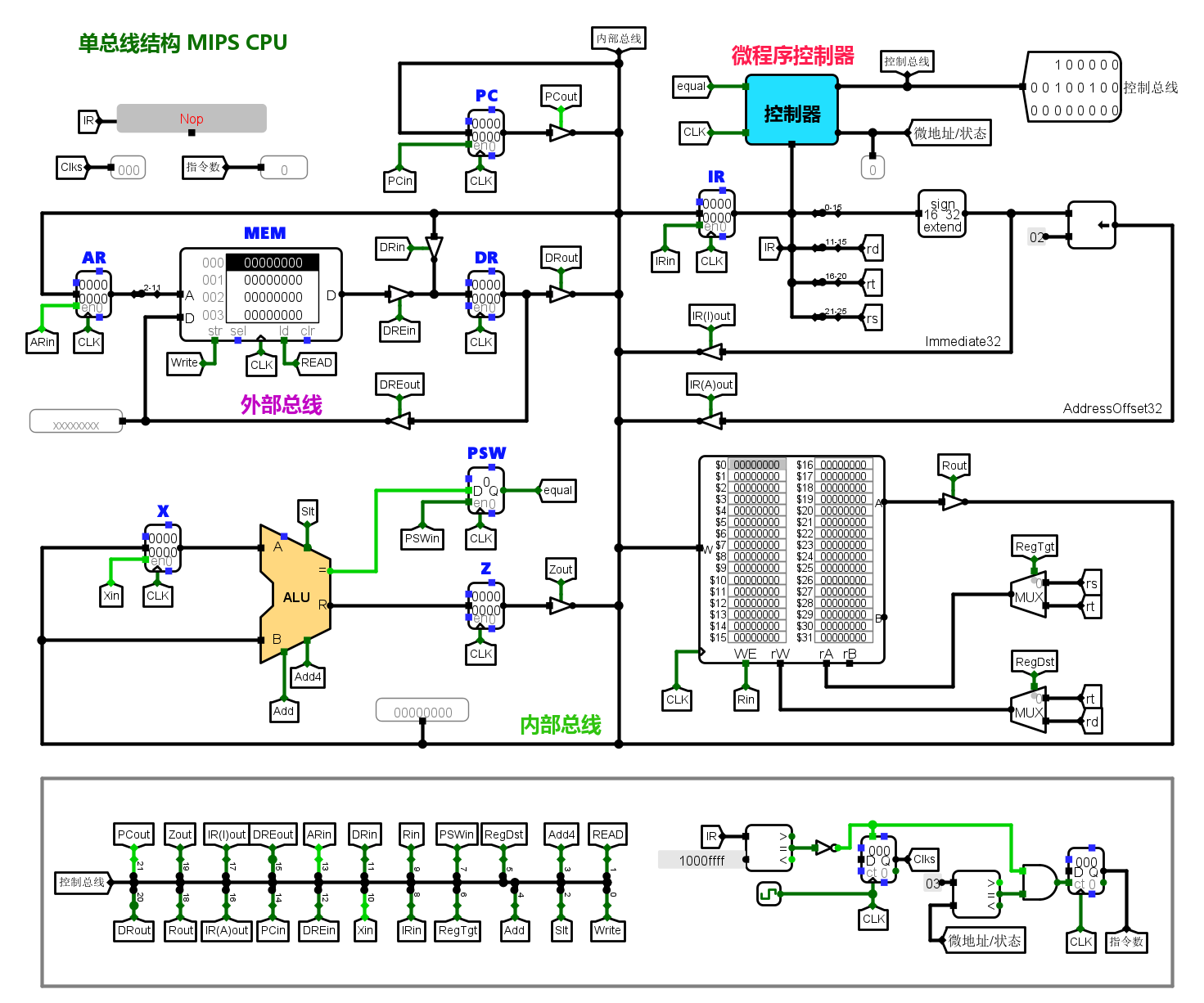
### 微程序条件判断测试产生逻辑





## 3.单总线CPU（微程序）电路及验证

### 整体电路



以上各个部件：

1. ALU：

设置暂存器X和 Z，分别用于暂存内部总线输入的值和输出结果，其中Z 仅受时钟信号

的控制，ALU上的控制信号+4、ADD、SUB 不能同时有效，只能给出一种运算。ALU还包

括一个状态寄存器，PSW保存运算标志。

2. Regs：

通用寄存器堆，用于控制输入和输出，有一个读端口和一个写端口，读写控制信号分别

为Rin和Rout。

3. 其他寄存器：

PC、AR（地址寄存器）、DR（缓冲寄存器），IR(指令寄存器)

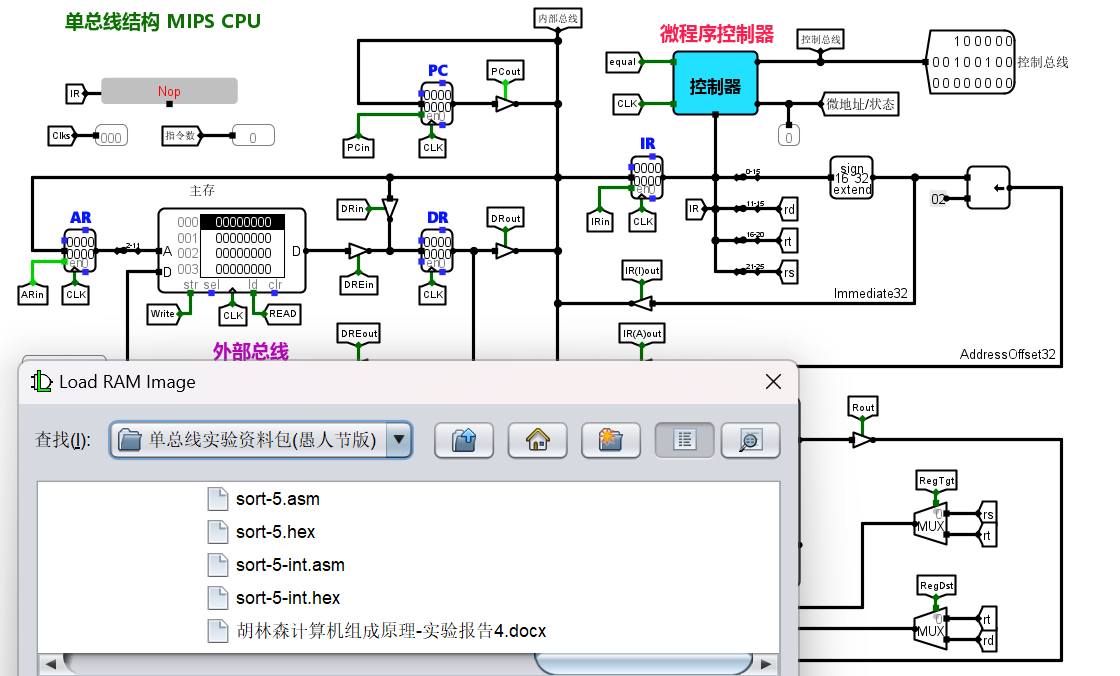
4. 控制器:

产生操作控制信号，控制流的产生会引起数据的流动，从而形成相应的数据通路，完成

指令的功能。

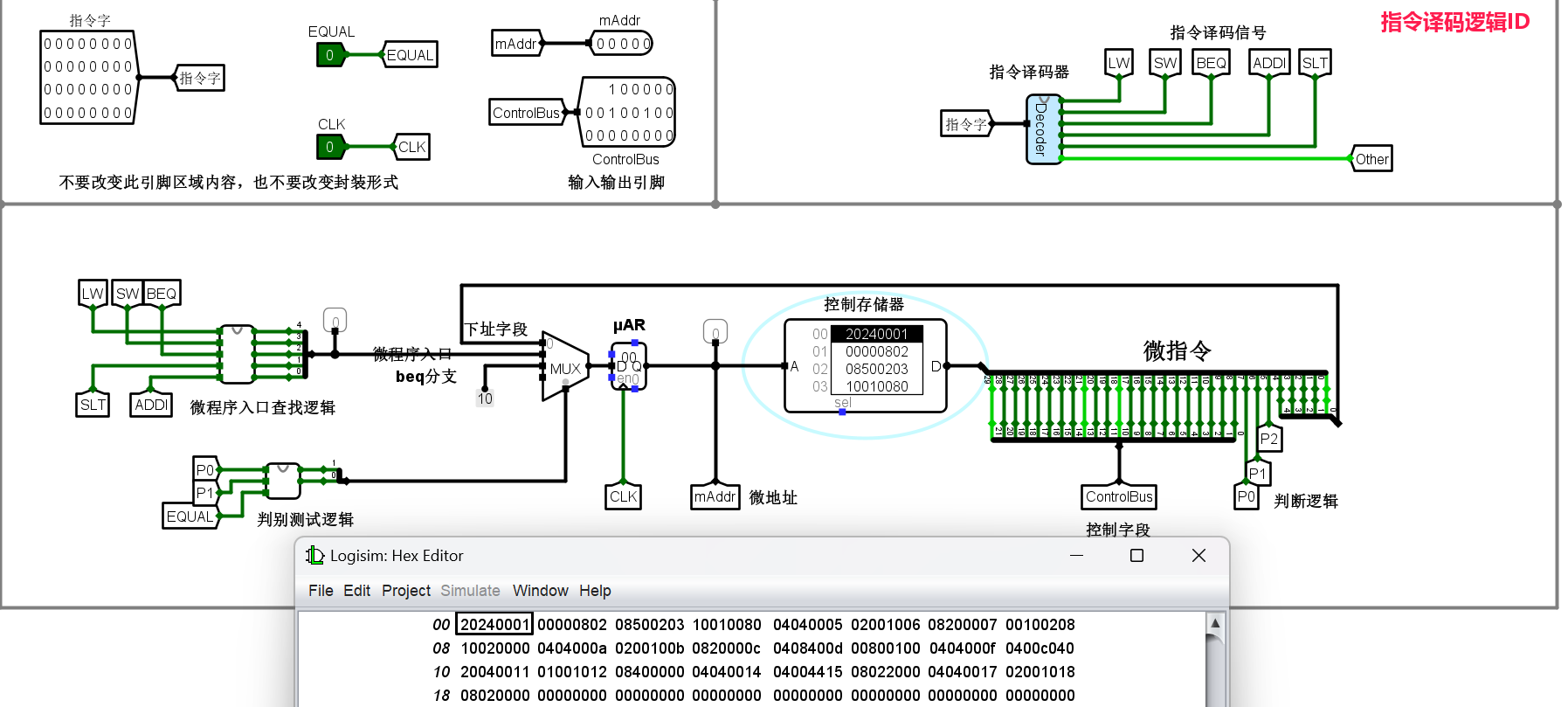
### 验证

1.向主存中载入数据sort-5.hex，便于进行排序算法验证



2.微程序导入

将表格“3.单总线MIPS微程序地址转移逻辑自动生成”产生的微指令载入控制存储器

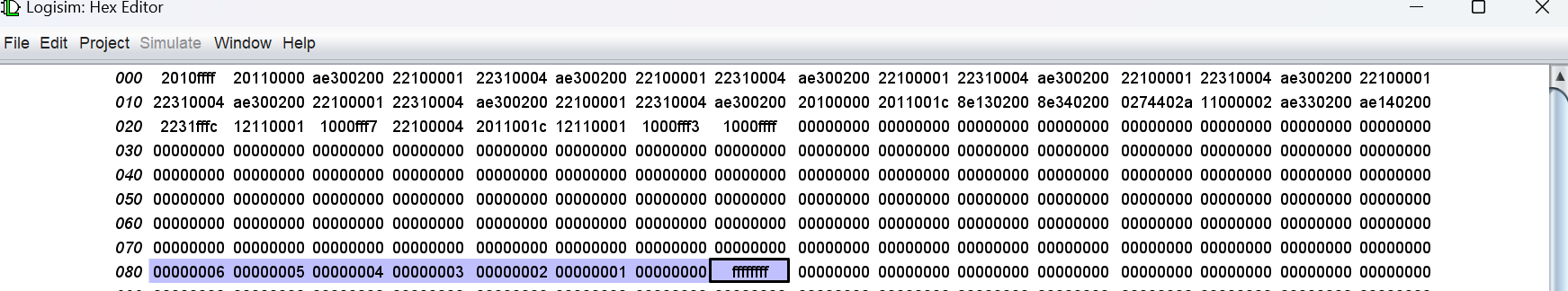


3.执行

点击 ctrl+k自动执行，因为冒泡排序结束的标志是进入死循环，此时时钟周期数和指令数不再增加，按 ctrl+k 暂停。

4.结果

查看外部存储器发现，数据降序排列。



# 五、问题和感想

**问题：**

1.在设计MIPS指令译码器时，我们遇到了一些挑战，比如如何准确地识别不同的指令类型并生成相应的译码信号。这需要对MIPS指令集有深入的理解，并熟练运用Logisim中的比较器组件和基本逻辑门电路。

2.在构建微程序控制器的过程中，我们发现状态转换图的理解和微指令格式的设计是关键。我们需要确保每个状态对应的微操作控制信号正确无误，并且状态之间的切换关系能够正确反映在微指令的执行顺序中。

3.在调试过程中，我们遇到了一些电路连接错误和逻辑错误，这些问题导致电路无法正常工作。

4.在验证排序算法时，我们发现微程序的执行结果并不总是符合预期。这可能与微程序的编写、控制存储器的微指令加载或者数据加载有关。通过反复检查和测试，发现是微程序表格填写出现逻辑错误导致操作地址错误，最终通过参考老师的程序编写确保了排序算法的正确执行。

**感想：**

1.让我感受到实践操作相结合的重要性。

在课堂上，我们学习了MIPS指令集、CPU结构和微程序设计等方面的理论。本次实验特定的MIPS指令，并测试这些微程序以确保它们的正确性，包括将微指令加载到控制存储器中，并验证它们的执行结果。从理论到实践的过程，不仅加深了我的理解，也让我意识到了理论与实践之间的差距。