***2025***



**计算机组成原理 ·实验报告·**

j0242087[1]

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： | CS2305 |
| 学 号： | U202315677 |
| 姓 名： | 岳皓 |
| 电 话： | 15115920806 |
| 邮 件： | [240733165@qq.com](mailto:13456@qq.com) |
| 完成日期： | 2025-06-30 |



# CPU设计实验

## 设计要求

利用logisim平台中现有运算部件构建一个32位运算器，可支持算数加、减、乘、除，逻辑与、或、非、异或运算、逻辑左移、逻辑右移，算术右移运算，支持常用程序状态标志（有符号溢出OF、无符号溢出CF，结果相等Equal），运算器功能以及输入输出引脚见下表，在主电路中详细测试自己封装的运算器。

表1-1 片引脚与功能描述

|  |  |  |  |
| --- | --- | --- | --- |
| 引脚 | 输入/输出 | 位宽 | 功能描述 |
| X | 输入 | 32 | 操作数X |
| Y | 输入 | 32 | 操作数Y |
| ALU\_OP | 输入 | 4 | 运算器功能码，具体功能见下表 |
| Result | 输出 | 32 | ALU运算结果 |
| Result2 | 输出 | 32 | ALU结果第二部分，用于乘法指令结果高位或除法指令的余数位，其他操作为零 |

表1-2 运算符功能

|  |  |  |
| --- | --- | --- |
| ALU OP | 十进制 | 运算功能 |
| 0000 | 0 | Result = X << Y 逻辑左移 （Y取低五位） Result2=0 |
| 0001 | 1 | Result = X >>>Y 逻辑右移 （Y取低五位） Result2=0 |
| 0010 | 2 | Result = X >> Y 算术右移 （Y取低五位） Result2=0 |
| 0011 | 3 | Result = (X \* Y)[31:0]; Result2 = (X \* Y)[63:32] 有符号 |
| 0100 | 4 | Result = X/Y; Result2 = X%Y 无符号 |
| 0101 | 5 | Result = X + Y Result2=0 (Set OF/CF) |
| 0110 | 6 | Result = X - Y Result2=0 (Set OF/CF) |
| 0111 | 7 | Result = X & Y Result2=0 |
| 1000 | 8 | Result = X | Y Result2=0 |
| 1001 | 9 | Result = X⊕Y Result2=0 |
| 1010 | 10 | Result = ~(X |Y) Result2=0 |
| 1011 | 11 | Result = (X < Y) ? 1 : 0 Signed Result2=0 |
| 1100 | 12 | Result = (X < Y) ? 1 : 0 Unsigned Result2=0 |
| 1101 | 13 | Result = Result2=0 |
| 1110 | 14 | Result = Result2=0 |
| 1111 | 15 | Result = Result2=0 |

## 方案设计

### 指令译码器

设计思路：围绕32位MIPS指令字，依据不同指令的编码特征（op、funct等字段），用比较器等逻辑模块，将指令译码为对应控制信号，区分指令类型，为CPU后续执行阶段提供控制依据，让CPU识别并执行不同指令操作。

设计原理：MIPS指令字的特定字段（如op）区分指令类型，不同指令有唯一编码，其中lw、sw、beq、addi为I型指令，由op段区分，slt为R型指令，其op段为0，由funct段区分，若指令不为上述五种指令则输出为otherinstr。利用组合逻辑，提取指令字中op等关键字段，与预设指令编码常量进行比较，匹配则输出对应指令译码信号。

设计过程：查阅MIPS指令集，明确各指令的op等字段的值，指令lw、sw、beq、addi的op段的十进制值分别为35、43、04、08，对应的十六进制编码常量分别为23、2b、04、08，指令slt的op段值为0，funct段十进制值为42，对应的十六进制编码常量为2a。在指令字段中，op段和funct段分别对应指令字IR的后六字和前六字，在logisim中分别取出，对于前四种指令，直接将取出的op与相应的编码常量在自带的比较器中比较，相等则输出，对于指令slt，除了将op进行比较，还要将funct与相应的编码字段进行比较，再通过与门进行输出，如果指令不为上述五种，通过或非门输出为otherinstr指令。实现电路如图 1-1所示。

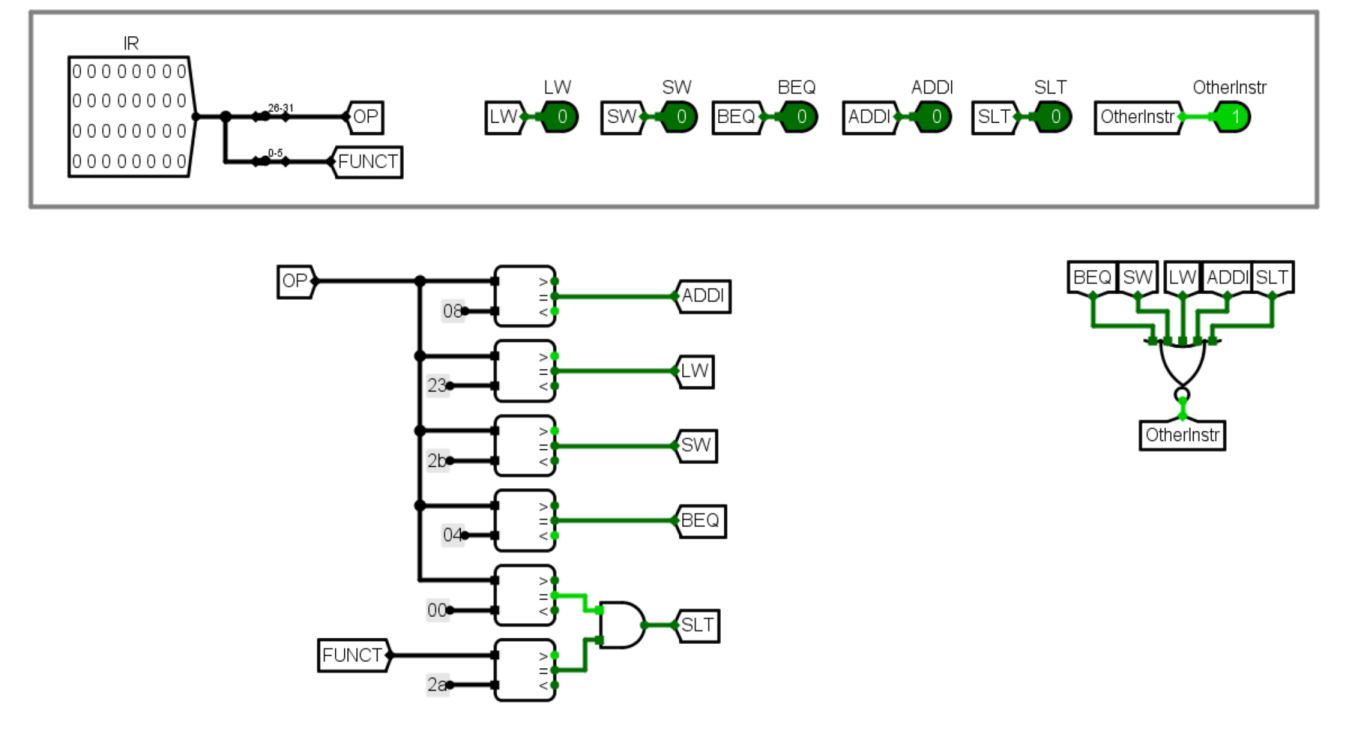


图 1-1指令译码器实现电路

### 支持中断的微程序入口查找

设计思路：以指令译码信号和中断请求为输入，构建分层状态机：先通过 “取指令” 等基础流程识别指令类型，再依据指令译码结果、中断响应条件，引导微程序进入不同功能阶段的入口，实现 “指令—中断” 双条件驱动的微程序流程跳转，让 CPU 能按指令类型和中断需求，有序执行对应微程序序列。

设计原理：用多级状态描述微程序流程，将每个指令的转移状态一一对应，通过指令译码器识别输入的指令后，再基于指令译码信号和中断请求，通过组合逻辑决定状态转移方向。

设计过程： 根据地址转移逻辑，lw、sw、beq、slt、addi、eret的微程序入口地址分别为4、9、14、19、25，填写6号EXCEL表格中的相应位置，如图 1-2所示。可以得到相对应的自动生成逻辑表达式，如图 1-3所示。根据得到的最终表达式，将其复制到logisim中，利用分析组合逻辑电路功能自动生成电路，得到最终的电路，如图 1-4所示。



图 1-2 地址转移逻辑表



图 1-3 自动生成逻辑表达式表

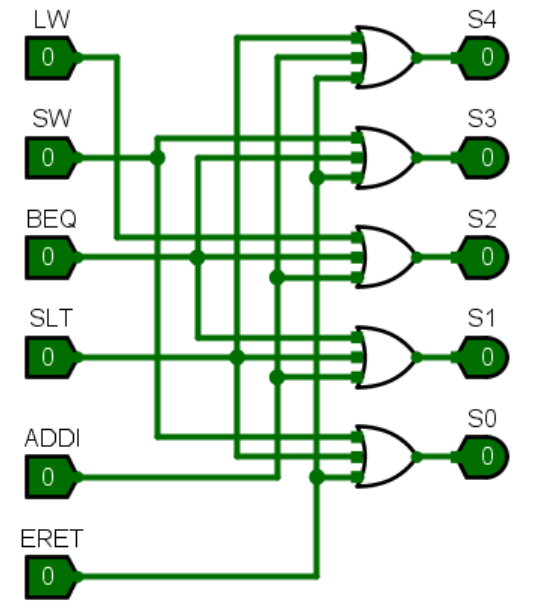


图 1-4 支持中断的微程序入口查找电路

### 支持中断的微程序条件判断

设计思路：根据输入条件（指令分支需求、运算状态、中断可能），动态决定微程序流向，优先响应中断，同时支持指令功能分支、相等判断分支，让微程序执行适配“正常指令流程+中断处理”的复杂逻辑。

设计原理：根据不同的输入进行相应的判断从而输出不同的值，其中输入的引脚及功能如表 1-3所示，其中优先级为中断>指令功能分支>相等分支。

表 1-3 支持中断的微程序条件判别测试逻辑电路引脚与功能描述

|  |  |  |  |
| --- | --- | --- | --- |
| 引脚 | 输入/输出 | 位宽 | 功能描述 |
| P0 | 输入 | 1 | 条件判断位，用于指令译码 |
| P1 | 输入 | 1 | 条件判断位，用于beq分支 |
| P2 | 输入 | 1 | 条件判断位，用于中断检测 |
| equal | 输入 | 1 | 运算操作数相等标志位 |
| IntR | 输入 | 1 | 中断请求信号 |

设计过程：当p0=1时，s0=1，转移到微程序分支；p0=0时，当p1=1，若equal=1，则s1=1，转到equal分支，若equal=0，p2=0或p2=1且intR=0，则s2=1，转到取指令分支，若equal=0，p2=1且intR=1，s0=1且s1=1，转移到中断程序入口，若p0和p1均为0，p2=1，若intR=1，则s2=1，转移到中断程序入口，若intR=0，则s1~s2均为0，当p1~p2均为0，则s1~s2均为0，按照该思路将4号EXCEL表填写完整，如图 1-5所示，根据自动生成的逻辑表达式如图 1-6，在logisim中自动生成电路，如所示。

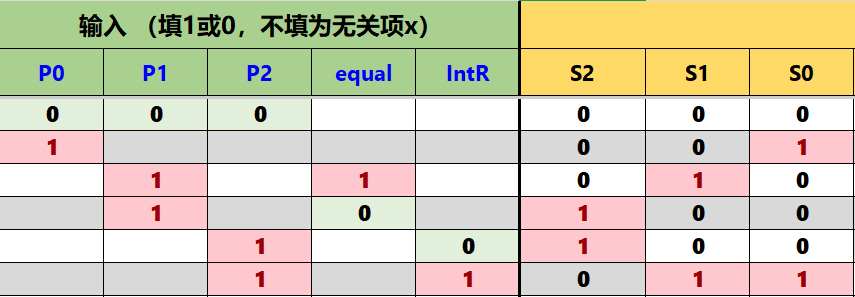


图 1-5 判断测试逻辑真值表



图 1-6 自动生成逻辑表

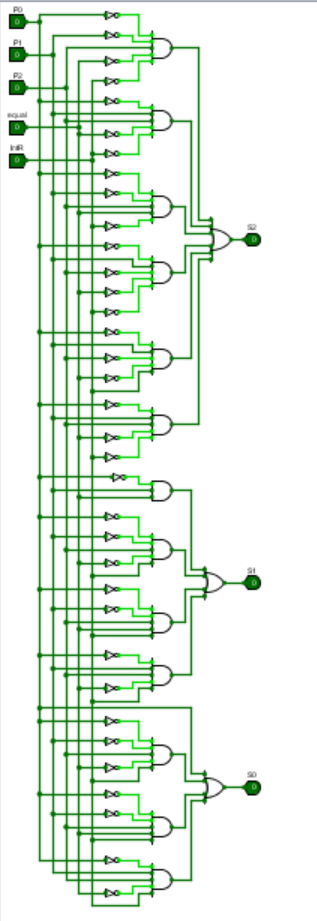


图 1-7 微程序条件判断电路实现

### 支持中断的微程序控制器

设计思路：构建“指令执行+中断响应”双流程的微程序控制器，让 CPU 既能按序执行指令（通过微程序驱动 ），又能在中断请求时，暂停当前流程、跳转至中断服务微程序，执行完后返回断点继续执行。

设计原理：根据微程序中的微指令、控制字段和下址字段，生成对应的微程序命令，将其加载到控制存储器中，再连接电路完成该微程序控制器的实现。

设计过程：查阅资料，根据每个指令的微程序指令填写EXCEL表，如所图 1-8示，将自动生成的十六进制的微指令复制到控制存储器中，根据前面的设计，在电路自带的多路选择器的相应的位置添加上编码常量，如beq分支地址和中断响应入口分别为16和26，对应的十六进制分别为10和1a。将微程序入口查找逻辑得到的值连接在入口地址，在将判别测试逻辑得到的值连接在多路选择器的选择口，具体电路如图 1-9所示。

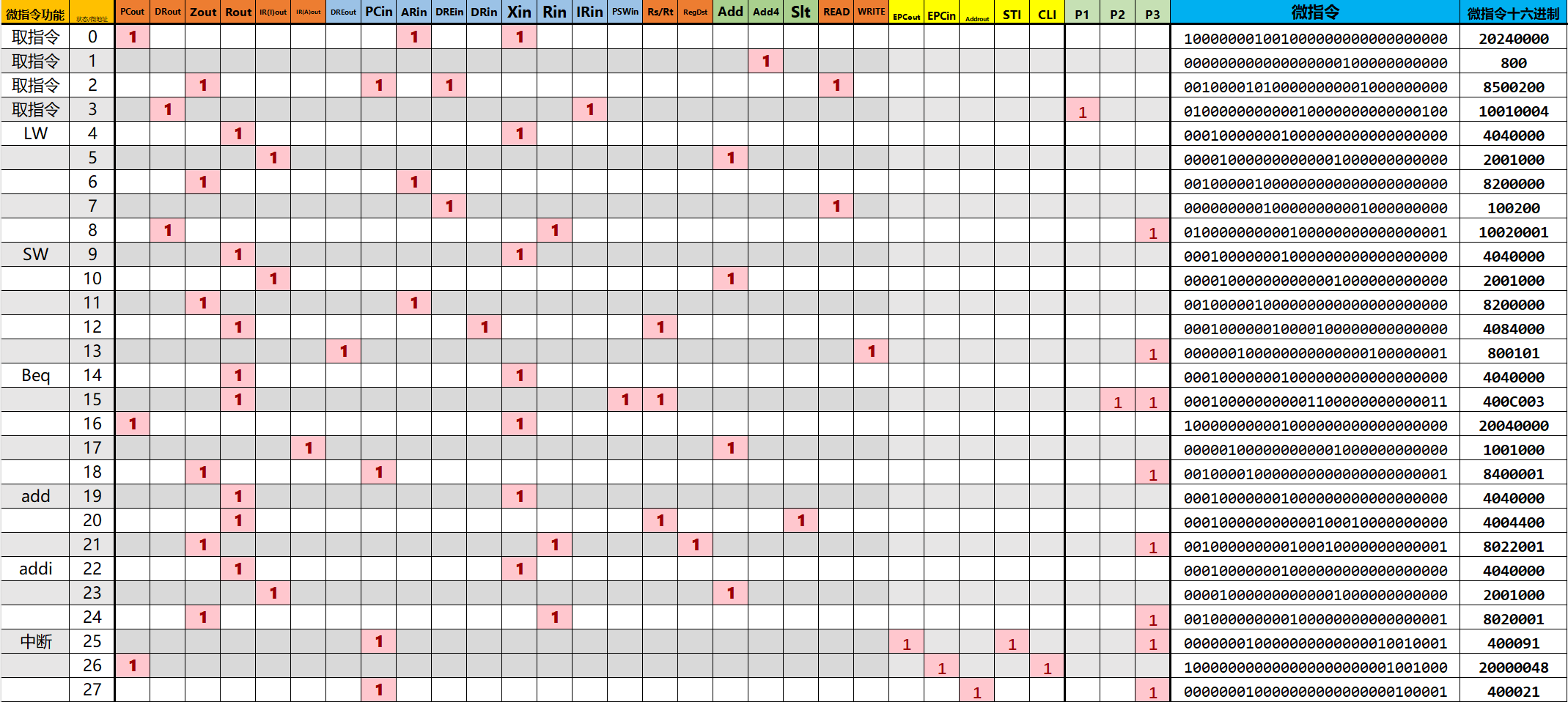


图 1-8 微程序自动生成表

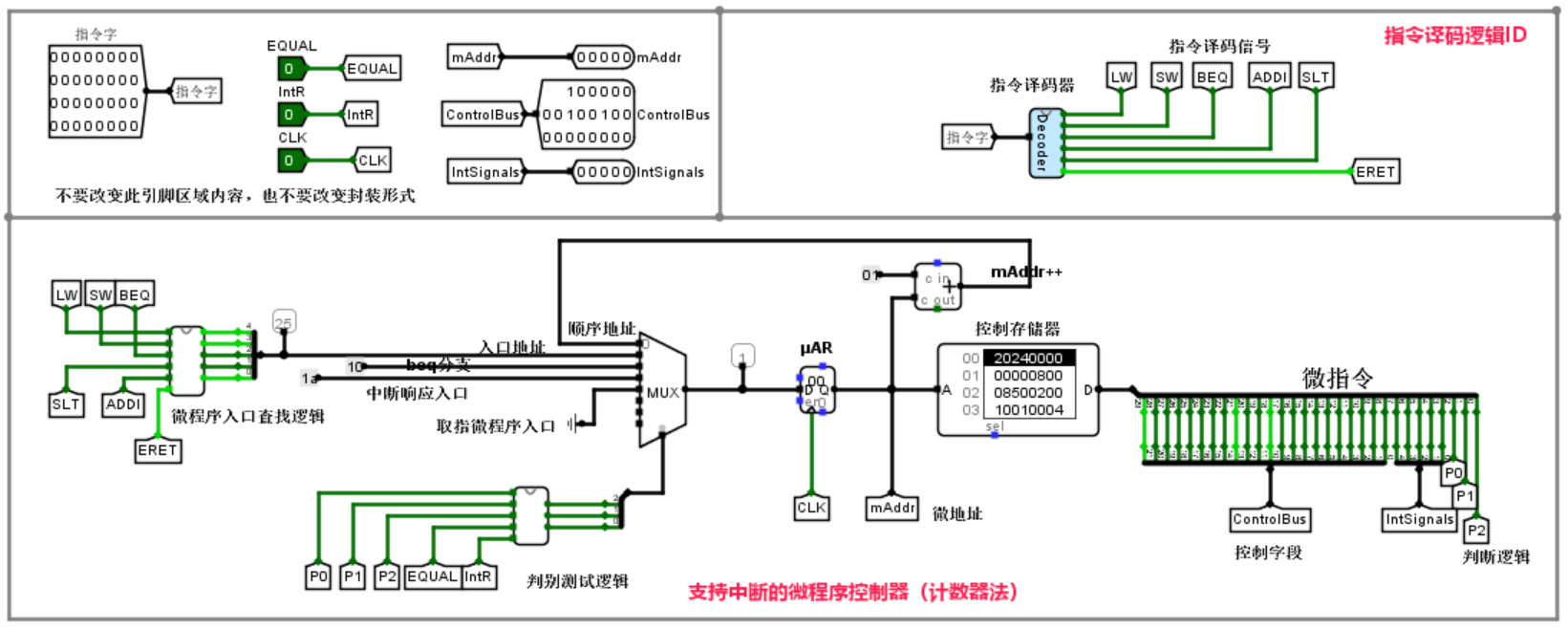


图 1-9 支持中断的微程序控制器实现电路

### 支持中断的单总线CPU

设计思路：通过硬件电路（EPC、中断使能寄存器、中断控制器）的协同，让单总线CPU具备“检测中断请求→保存断点→响应中断→恢复断点”的完整能力，实现按键中断的响应与服务程序执行。

设计原理：对于EPC，其作用为保存中断断点，时钟上升沿时，若EPCin=1，将内部总线数据锁存到EPC，若EPCout=1，将EPC数据输出到内部总线。对于IE，其作用为控制中断相应权限，用一位D触发器，开中断信号置1，关中断信号置0，输出参与中断请求判断，对于中断控制器，其作用为汇集中断源，输出最高优先级中断请求。当有中断请求且IE为1时，触发中断，然后中断响应信号触发EPCin=1，PC=中断向量地址，最后ERET指令触发EPCout=1、开中断等。

设计过程：将一个32位寄存器用作EPC的寄存器，使能端连接EPCin，输出端为一个三态门，当EPCout为1时将寄存器的值输入到内总线。对于中断使能寄存器，用一个D触发器实现，置位端和清空端分别连接关中断和开中断。对于中断控制逻辑，核心是找到两个对应的地址，在实验中实现的时候需要用到mars工具去寻找对应的地址，多路选择器的输出端口连接一个三态门，当Addrout的值为1时将值输入到内总线上。将对应电路连接完成，中断逻辑的电路便完成了，具体的实现电路如图 1-10所示。

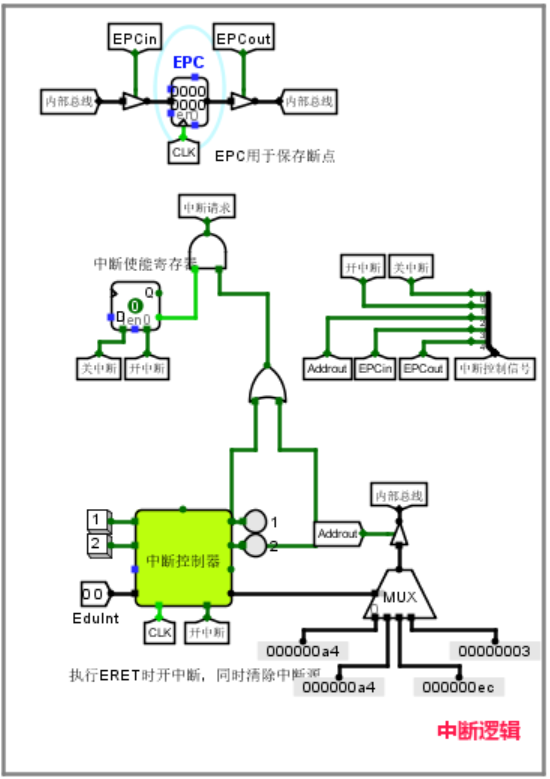


图 1-10 支持中断的单总线CPU实现

### 支持中断的现代时序硬布线控制器状态机

设计思路：用硬布线逻辑实现“指令执行状态流+中断响应分支”，让CPU按固定状态转移执行指令，同时在中断请求时，快速切入中断响应状态，完成服务后返回原流程。

设计原理：根据状态机的原理，找到现态和次态之间的逻辑关系，每当一个时间脉冲到来时，若没有出现指令则自动进入下一个次态，若有指令或者判断条件，则根据对应的指令或者判断条件进入与之对应的状态。每当一条微程序完成时，还需判断是否需要进行中断操作。

设计过程：根据状态机得出逻辑关系，并将相对应的逻辑转变在EXCEL表中给出，分别填入相应的现态、输入信号与次态，如图 1-11所示。根据得到的自动生成的逻辑表达式如图 1-12所示，在logisim中自动生成相应的电路，如图 1-13所示。

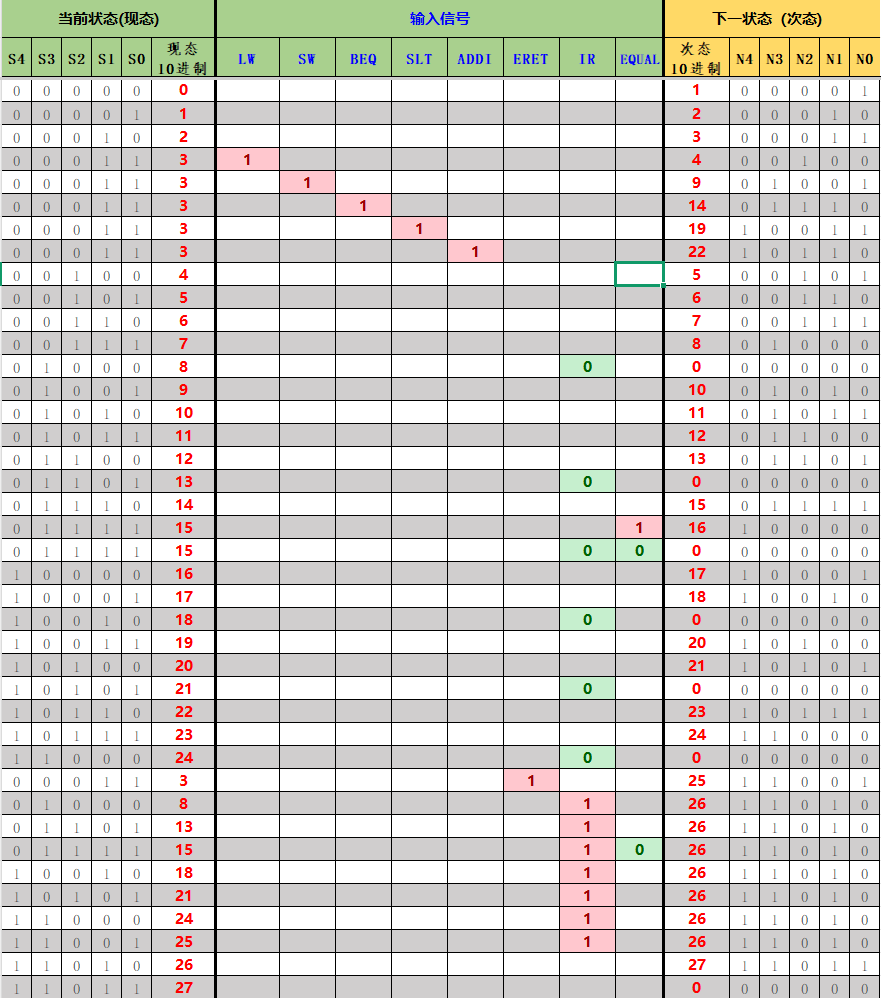


图 1-11 状态机逻辑生成表

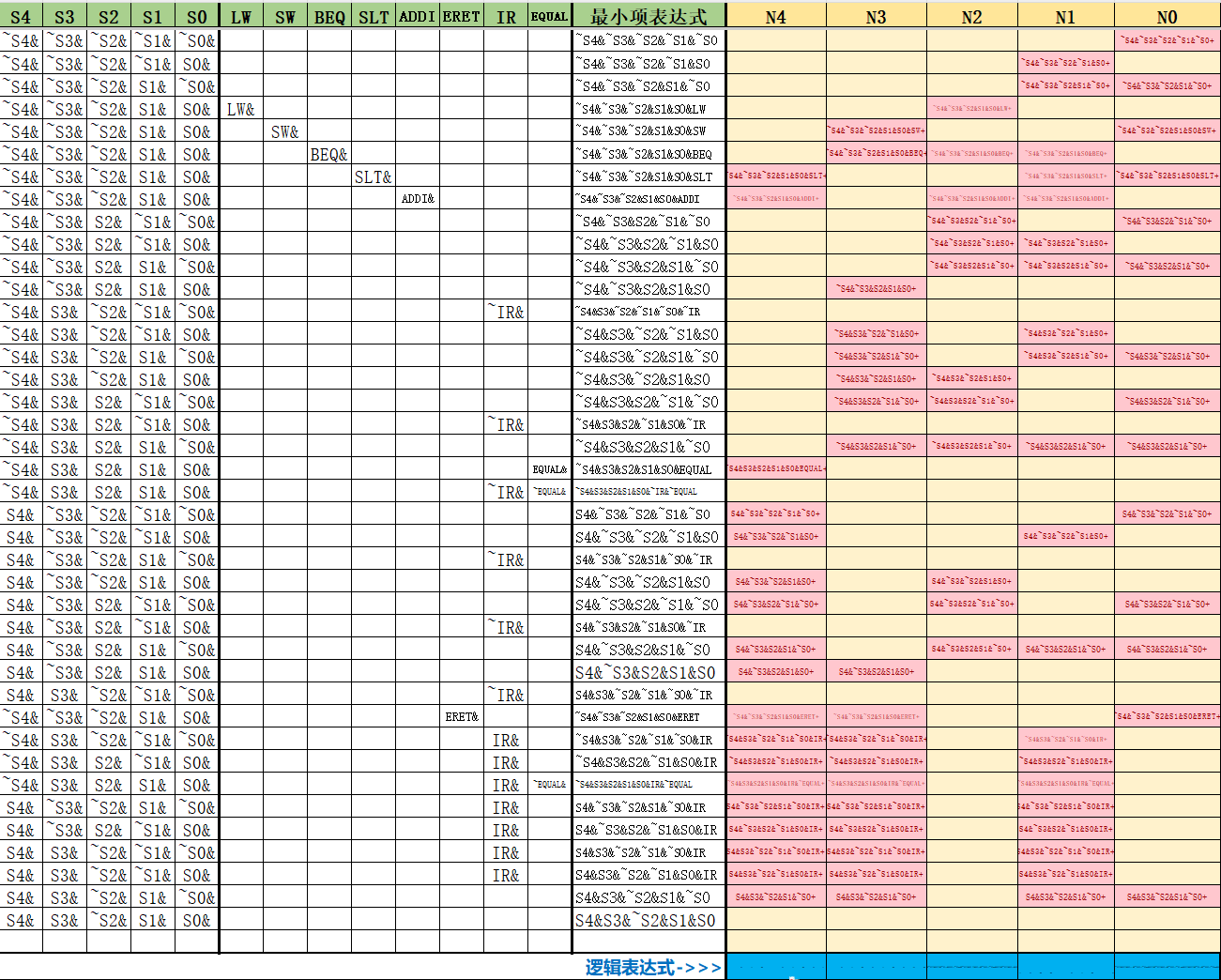


图 1-12 自动生成逻辑表

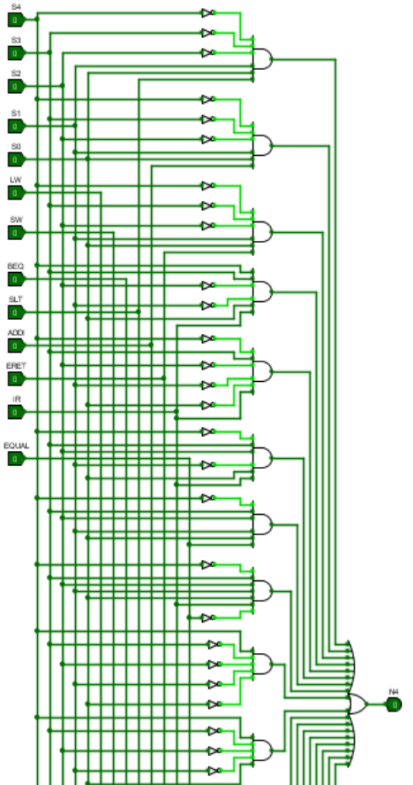


图 1-13 支持中断的硬布线控制器状态机实现电路

### 支持中断的现代时序硬布线控制器

设计思路：用硬布线逻辑替代微程序控制器，构建“指令执行+中断响应”双流程的时序控制单元，让CPU按固定时序驱动数据通路，同时支持中断请求的快速响应与恢复。

设计原理：将微程序控制机的微程序命令加载到现代时序硬布线控制器的控制存储起中，再连接电路完成现代时序硬布线控制器的实现。

设计过程：将之前的微指令命令加载到控制存储器，运用设计的硬布线状态机，将向对应的指令连接到相对应的位置，在将状态用分线器将四位分别连接到相应位置，具体的实现电路如图 1-14所示。

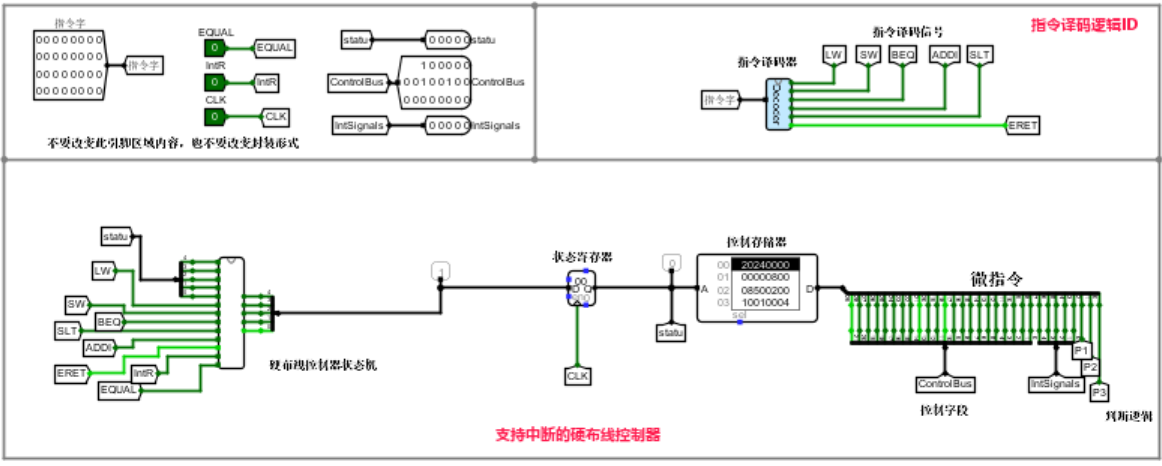


图 1-14 支持中断的现代时序硬布线控制器实现电路

## 实验步骤

1. 仔细阅读头歌平台各关卡中的实验提示，并了解各部分子电路的功能。
2. 根据提示及上课所学知识完成子电路的设计，对于部分关卡需要填写实验包里面提供的Excel文件来自动生成表达式，再使用Logisim的自动生成功能生成组合逻辑电路。
3. 提交代码文件到头歌平台进行评测。

## 故障与调试

### 微程序控制器无法正常运行

**故障现象：**在educoder平台上运行时出现错误。

**原因分析：**在一开始，寄存器设置为上升沿刷新，但当输入端有输入且clk变化一个周期后寄存器中的数据仍然没有改变，在下一个时钟脉冲时才有改变，导致测试时多出了一倍的结果，通过检查发现了该错误。

**解决方案：**将寄存器设置为下降沿刷新。

### 状态转移存在问题

**故障现象：**在“支持中断的微程序控制器设计”关卡中，后续近一半指令全部错误。

**原因分析：**在Excel文件中设计时，在SW微程序最后一条指令P3=1未设置，导致SW微程序执行后无法返回取指微程序。

### **解决方案：**修改对应Excel文件中出错的跳转逻辑。

### 单总线CPU的中断无法正常运行

**故障现象：**在educoder平台上运行时出现错误。

**原因分析：**在一开始，没有正确的理解EPC的逻辑，将寄存器的输出没有输出到内总线上，导致错误。在对于多路选择器的两个地址的获取时，一开始获取的有误，导致出现故障。

**解决方案：**将epc输出端为一个三态门，当EPCout为1时将寄存器的值输入到内总线，通过mars工具去寻找对应的中断地址。

## 测试与分析

单周期CPU的测试用例如图 1-15所示。

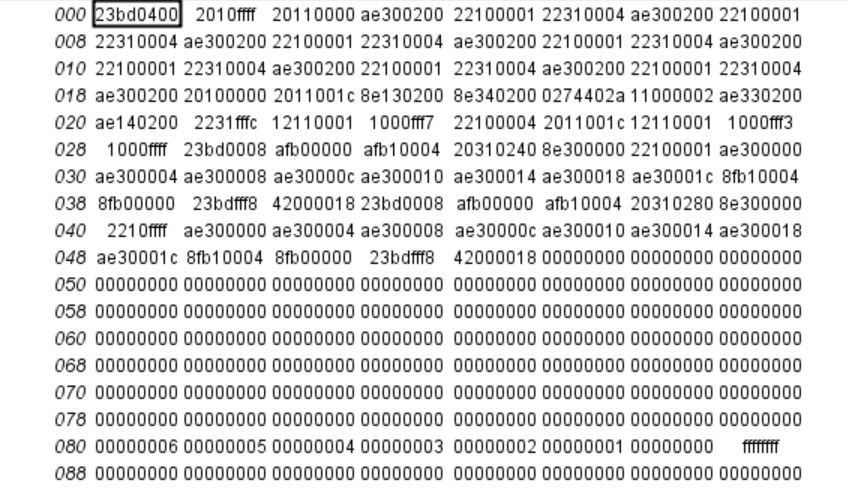


图 1-15 测试用例

测试过程中，加载sort-5-int.hex程序并运行后，结果如图 1-16所示，按下按键1，其中的内存数据如图 1-17所示，再次按下按键1，执行的结果如图 1-18所示，如按下按键2，执行结果如图 1-19所示，此时再按下按键2，执行结果如图 1-20所示。

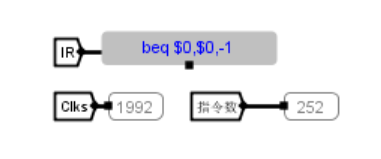


图 1-16 测试结果1

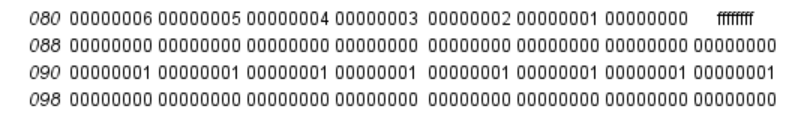


图 1-17 按下按键1后内存数据

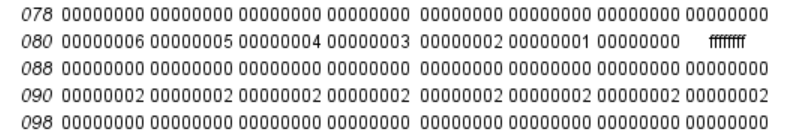


图 1-18 再次按下按键1后内存数据

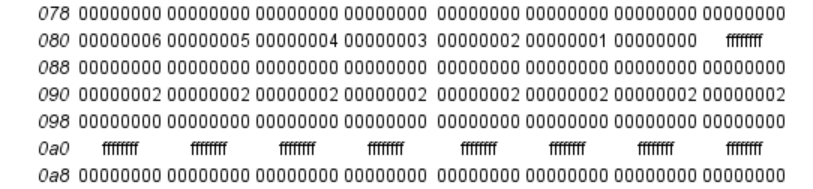


图 1-19 按下按键2后内存数据

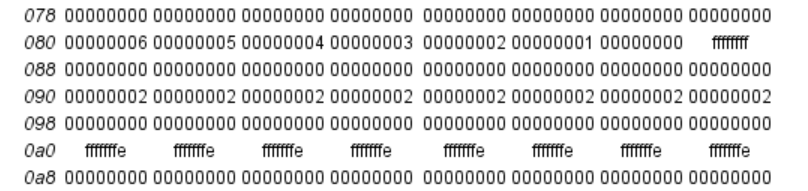


图 1-20 再次按下按键2后内存数据

分析：值得注意的是，测试程序的预期功能是对数据进行有符号比较的降序排序。根据在educoder平台的测试结果显示，测试结果良好，且自己运行结果与实验所给运行结果一致，说明实验基本通过。

## 实验总结

本次实验主要完成了如下几点工作：

1. 完成方案总结：在本次实验主要完成了支持中断的现代时序的MIPS的CPU，其中包括指令译码器、支持中断的微程序入口查找、支持中断的微程序条件判断、支持中断的微程序控制器、支持中断的现代时序硬布线控制器状态机、支持中断的现代时序硬布线控制器和支持中断的单总线CPU总共七个部分。本次试验是在上一个实验的基础上完成的，相较于上个实验添加了中断机制。
2. 功能总结：成功的完成了支持中断的现代时序的MIPS的CPU的逻辑设计与电路实现，能够在educoder平台上完成所有的测试任务。

## 实验心得

1. 通过本次实验，收获颇丰。首先是对支持中断的现代时序的MIPS的CPU的逻辑结构与设计有了更加深刻的认识。在本次实验中，有不少电路都是需要根据逻辑关系或者现态与次态的关系填写对应的逻辑表后根据逻辑表达式来自动生成，这与前几章的实验题目不同，因此更加考验大家对逻辑关系的把握。在做实验的时候有一个比较困难的点就是当时填写微指令生成对应的微程序的时候，指令的数目较多且比较繁杂，如果仅仅只是照着书本上的内容填写显然是不够的，还应该深入理解每个指令的步骤和功能，才能够填写完整并填写准确，同时也能够对CPU的设计以及中断的过程有更加清晰的认识，为之后的课程设计打下基础，收获良多。
2. 通过本次实验，我对课本知识的理解更加深刻透彻，对CPU的结构和运行原理更加清晰，并且对硬布线和微程序两者的优势、不足和不同之处认识更加充分。总的来说本次实验帮助我巩固了课本知识，激发了我对计算机硬件设计的兴趣，为日后的学习打下基础。

|  |
| --- |
| 一、原创性声明 |
| 本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。  特此声明！  **作者签字: 岳皓 IMG_20250630_130903_edit_44540041134518** |
| 二、对课程实验的学术评语（教师填写） |
|  |
| 三、对课程实验的评分（教师填写） |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | 评分项目  （分值） | 课程目标1  工具应用  （10分） | 课程目标2  设计实现  （70分） | 课程目标3  验收与报告  （20分） | 最终评定  （100分） | | 得分 |  |  |  |  | |
| **指导教师签字:** |