***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： | CS2301 |
| 学 号： | U202311111 |
| 姓 名： | losyi |
| 电 话： | 11111111111 |
| 邮 件： | u202311111@hust.edu.cn |
| 完成日期： | 2025-06-13 |



# 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基于单总线结构，支持五条MIPS指令，支持中断的现代时序处理器，实验需要分别实现现代时序的微程序控制器，硬布线控制器，并最终实现单总线MIPS CPU的联调。实验要求在给定的单总线架构中实现相应控制器，并支持表1-1所示的五条指令

表1-1 指令功能描述

|  |  |  |
| --- | --- | --- |
| # | 指令 | 指令功能 (RTL描述) |
| 1 | lw rt,imm(rs) | R[rt] ← M[R[rs] + SignExt(imm)] |
| 2 | sw rt,imm(rs) | M[R[rs] + SignExt(imm)]← R[rt] |
| 3 | beq rs,rt,imm | if(R[rs]==R[rt]) PC←PC+4+SignExt(imm)<< 2 |
| 4 | slt rd,rs,rt | R[rd] ← R[rs] < R[rt] 有符号比较 |
| 5 | addi rt,rs,imm | R[rt] ← R[rs] + SignExt(imm) |

图 1-1给出了现代时序硬布线控制器模型，现代时序采用有限状态机来描述指令的执行过长，将不同指令的执行的每个时钟周期均对应一个状态，每个状态对应特定的微操作控制信号，由硬布线控制器组合逻辑生成，次态则与指令的译码信号、反馈信息和现态有关。控制器的核心模块是有限状态机，由有状态寄存器和FSM状态机组合逻辑控制单元构成。FSM状态机组合逻辑控制单元的输入包括现态，指令的译码信号和反馈信号，输出为次态，送入状态寄存器输入端，在时钟信号的作用下输入状态寄存器中，作为下一时刻的现态。

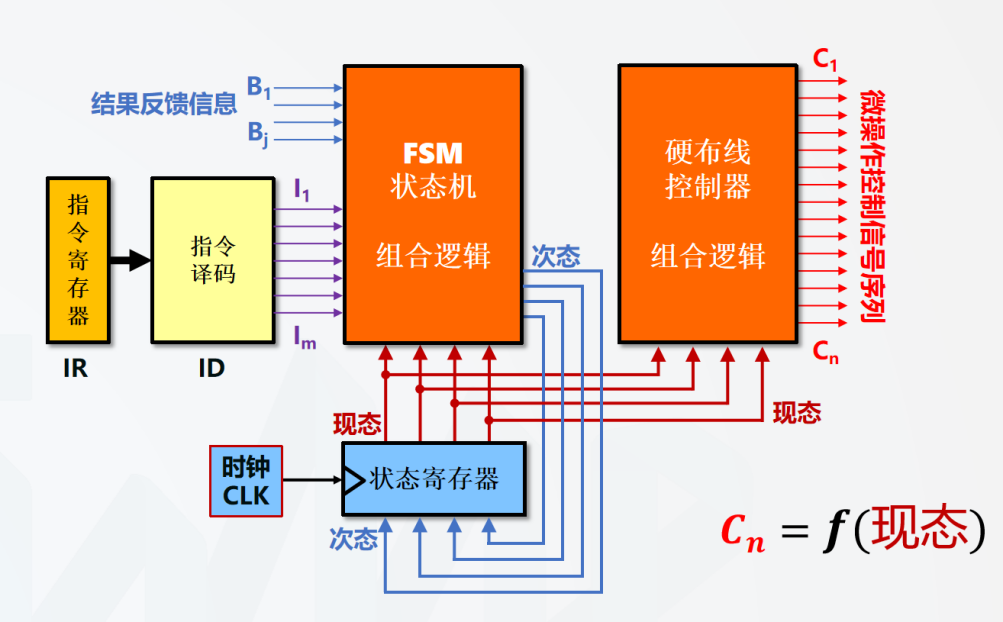


图 1-1现代时序硬布线控制器

实验首先需要实现微程序控制器，图 1-2给出了微程序控制器的组成框图。我们需要根据这个框图完成logisim中的微程序控制器。它主要由控制存储器、地址转移逻辑、微地址寄存器三部分组成。我们需要依次实现各功能部件，完善微程序控制框架，最终实现完整的微程序控制器，并进行实际的联调。注意，这里我们采用计数器法，且支持中断机制。

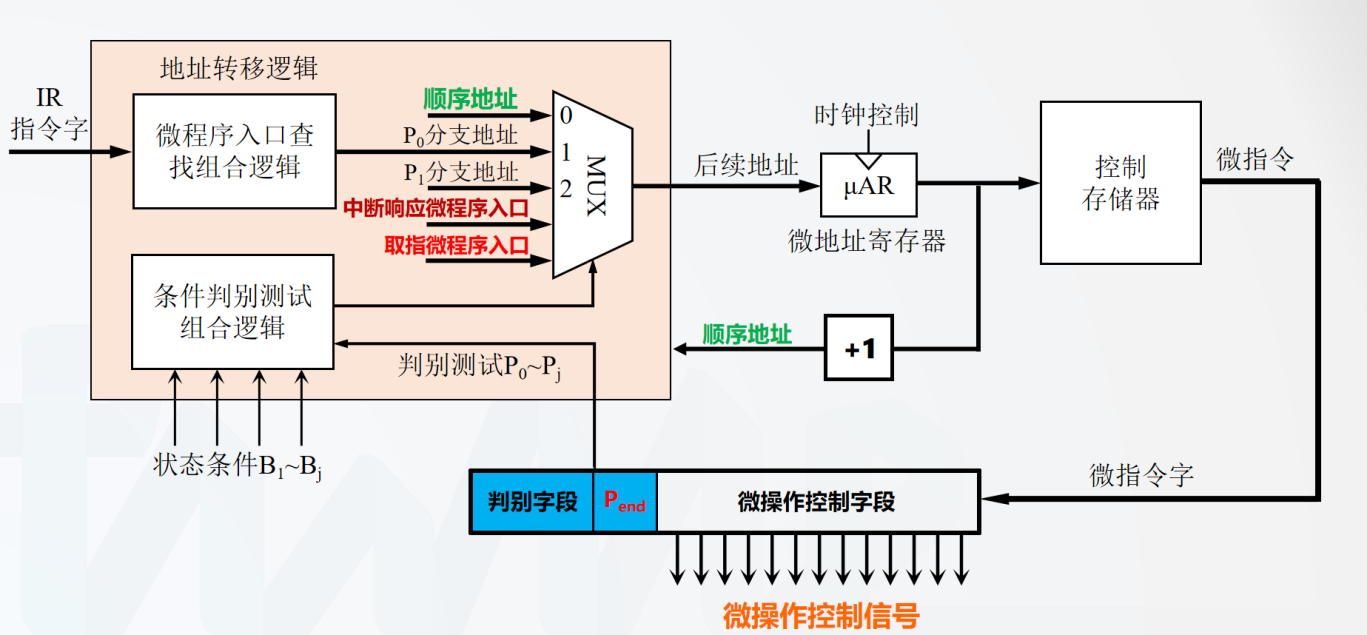


图 1-2微程序控制器组成框图

## 方案设计

本实验将依次实现指令译码器、入口查找逻辑、条件判别测试组合逻辑，FSM状态机，以逐步实现微程序控制器和硬布线控制器。

### 设计MIPS指令译码器

指令译码器的核心任务是将32位MIPS指令字解析为具体的控制信号。MIPS指令的格式决定了译码方式：每条指令的高6位是操作码（opcode），用于区分指令类型；对于R型指令，低6位是功能码（funct），用于进一步细分操作。例如，lw指令的操作码是100011，sw指令是101011，beq指令是000100，addi指令是001000，而slt指令需要同时检查操作码000000和功能码101010。其他指令则统一归为OtherInstr。译码过程本质上是将输入指令的特定字段与这些固定值进行比较，匹配时输出对应的高电平信号。由于指令之间互斥，这些信号不会同时有效，最后通过一个简单的逻辑组合就能生成OtherInstr信号。

具体实现上，需要用比较器模块来识别每条目标指令：每个比较器接收指令的opcode或funct字段，与预设的二进制值进行比对，匹配时输出有效信号。比如LW信号的比较器就是检查opcode是否等于100011。对于slt指令，需要同时比较opcode和funct两个字段，这可以通过与门连接两个比较器来实现。OtherInstr信号则是将其他所有输出信号通过或非门组合而成，确保没有其他指令匹配时才有效。最终电路如

图 1-3所示。

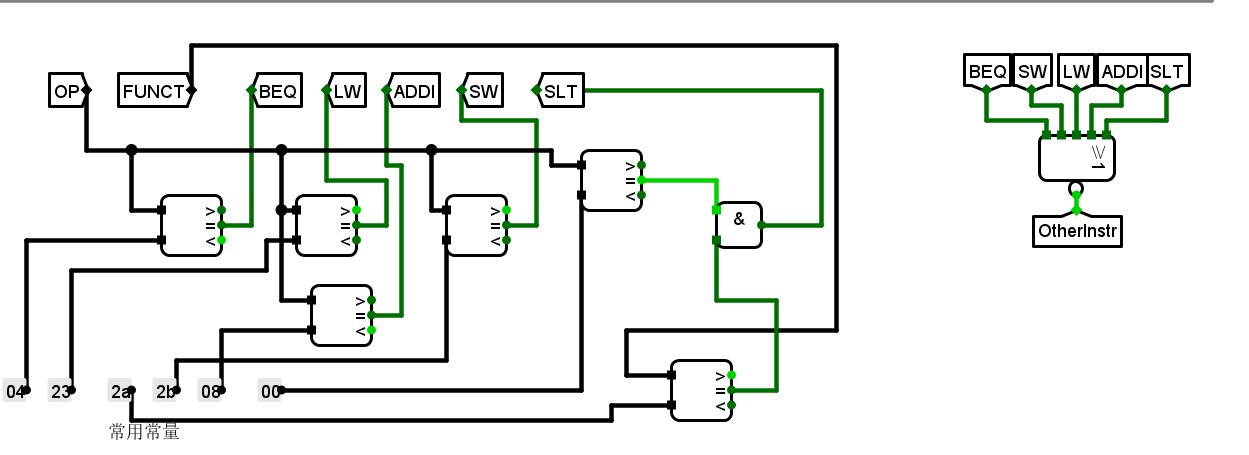


图 1-3指令译码器电路

### 微程序入口查找逻辑

通过现代时序分析5条MIPS指令的指令周期及数据通路，可以得到图 1-4所示的指令执行状态转换图，取指令阶段的最有一个状态S3要根据指令译码信号生成不同的微程序入口地址，需要构建图 1-2中的微程序入口查找组合逻辑，其引脚功能如表 1-2所示。

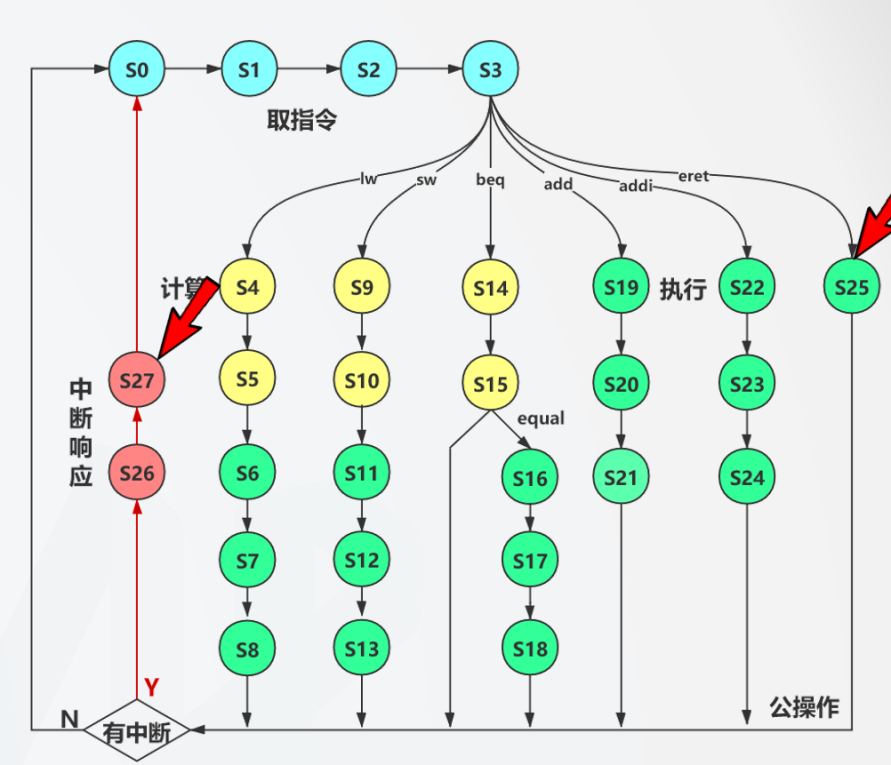


图 1-4指令执行状态转换图

表 1-2微程序入口查找电路引脚功能

|  |  |  |  |
| --- | --- | --- | --- |
| 信号 | 输入/输出 | 位宽 | 功能描述 |
| LW | 输入 | 1 位 | LW 指令指令译码信号 |
| SW | 输入 | 1 位 | SW 指令译码信号 |
| BEQ | 输入 | 1 位 | BEQ 指令译码信号 |
| ADDI | 输入 | 1位 | ADDI指令译码信号 |
| SLT | 输入 | 1 位 | SLT指令译码信号 |
| ERET | 输入 | 1 位 | ERET指令译码信号 |
| S4~S0 | 输出 | 1 位 | 微程序地址入口地址 |

为方便实现相应逻辑，实验包提供了相应的电子表格文件，通过填写真值表，可以自动生成微程序入口地址状态的逻辑表达式，最终得到的组合逻辑电路。

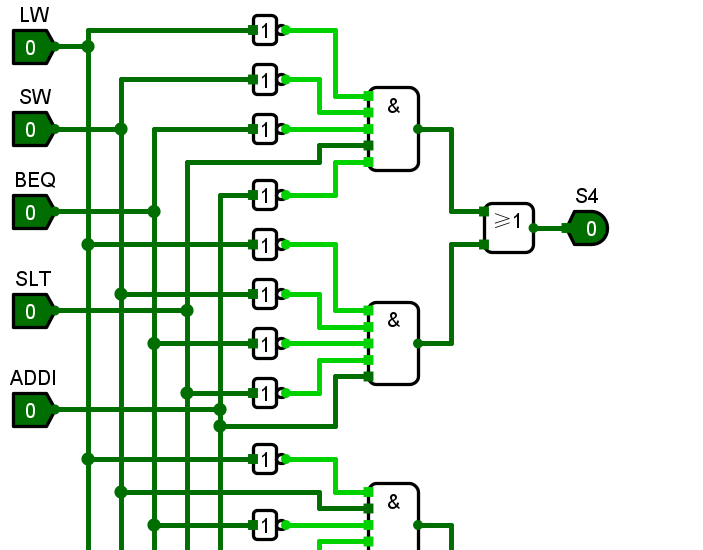


图 1-5微程序入口查找逻辑电路

### 条件判别测试组合逻辑

根据图 1-2中多路选择器的连接情况及具体条件判别测试位的定义，设计条件判别测试组合逻辑。利用实验包中提供的Excel表生成逻辑表达式可以自动生成电路。该电路引脚功能表述如表 1-3所示。

表 1-3条件判别测试组合逻辑电路引脚功能

|  |  |  |  |
| --- | --- | --- | --- |
| 信号 | 输入/输出 | 位宽 | 功能描述 |
| P0 | 输入 | 1 | 判别测试位，为1表示要根据指令功能进行微程序分支 |
| P1 | 输入 | 1 | 判别测试位，为1表示要根据equal标志进行微程序分支 |
| P2 | 输入 | 1 | 判别测试位，为1表示是微程序的最后一条微指令，可能需要进行中断响应 |
| equal | 输入 | 1 | 条件状态位，表示运算相等 |
| S4~S0 | 输入 | 1 | 现态输出第0位 |

具体的输入和输出(MuxSel)的关系为：P0为1时，MuxSel为1，选择对应指令的入口地址；P1为1且equal为1时，进行beq分支，MuxSel为2，选择beq入口地址；

P2为1且IntR为1时，进行中断分支，MuxSel为3，选择中断入口地址；P1比P2的优先级更高。由此可填写真值表(Excle表)如下。

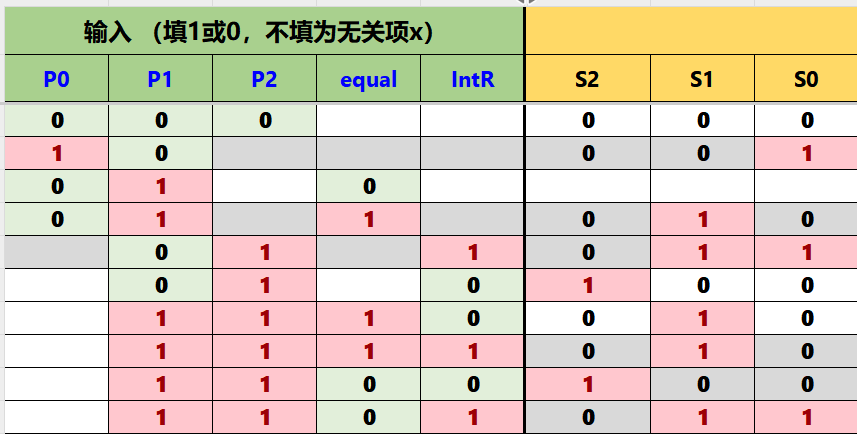


图 1-6判别测试逻辑自动生成表

### 微程序控制器

打开微程序控制器子电路，其电路引脚功能描述如表 1-4所示，参考图 1-2给出的组成框图和前面已经实现的指令译码器、微程序入口查找逻辑和条件判别测试组合逻辑和其他Logisim电路组件，进行、正确连线，实现微程序控制器的主要数据通路，如图 1-7所示。

表 1-4微程序控制器电路引脚功能

|  |  |  |  |
| --- | --- | --- | --- |
| 信号 | 输入/输出 | 位宽 | 功能描述 |
| IR | 输入 | 32 | MIPS指令字 |
| equal | 输入 | 1 | 条件反馈信号，表示运算相等 |
| IntR | 输入 | 1 | 中断请求信号 |
| mAddr | 输出 | 5 | 当前微指令地址 |
| ControlBus | 输出 | 5 | 微指令字的控制字段 |

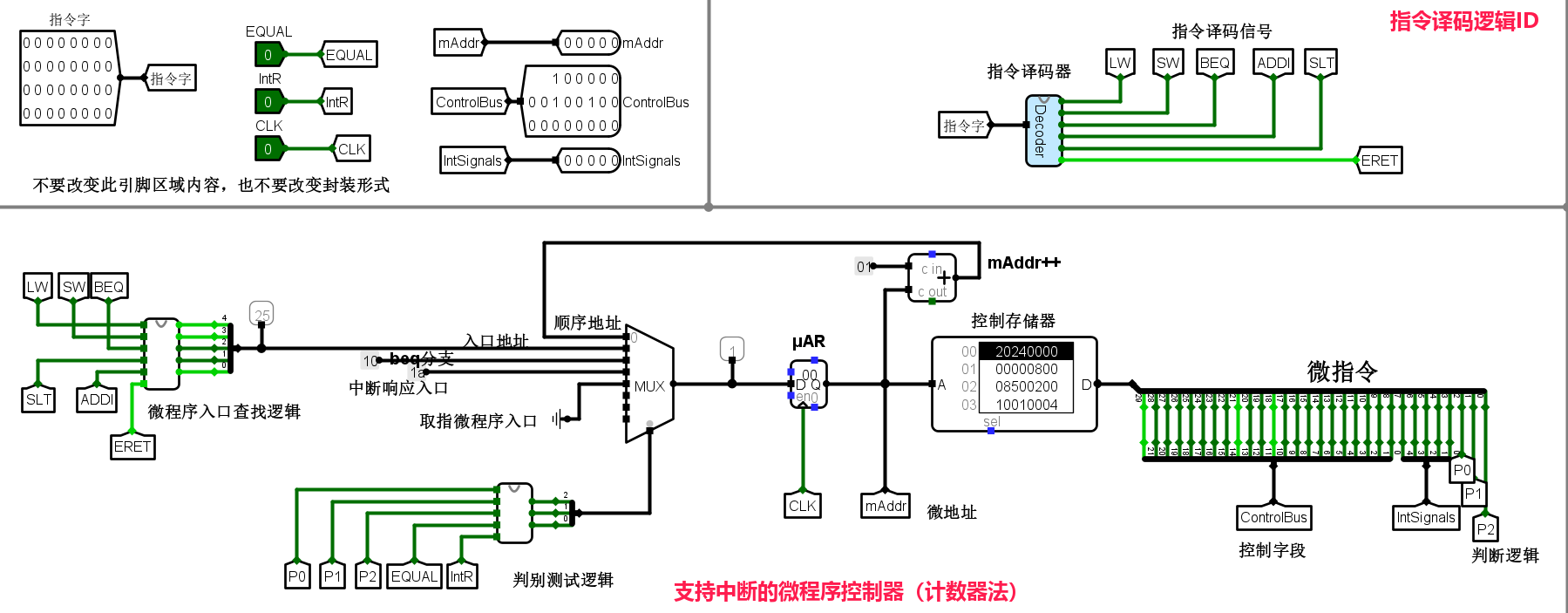


图 1-7微程序控制器电路图

接

着，我们需要利用“6.单总线MIPS支持中断微程序逻辑自动生成(2020-4-1)”表自动生成微指令，加载到控制存储器中，我们需要明白S0-S27每个状态对应的微指令所对应的控制字段信号，例如Eret中断返回指令，需要将EPC->PC,并且开中断(STI),

同时可能是最后一个状态(P2),因此最终为1(有效)的字段为EPCout，PCin，STL，P2。

其他微指令在《计算机组成原理（微课版）》和谭老师课件中均有讲解，此处不再赘述，完成后的表格如图 1-8所示。

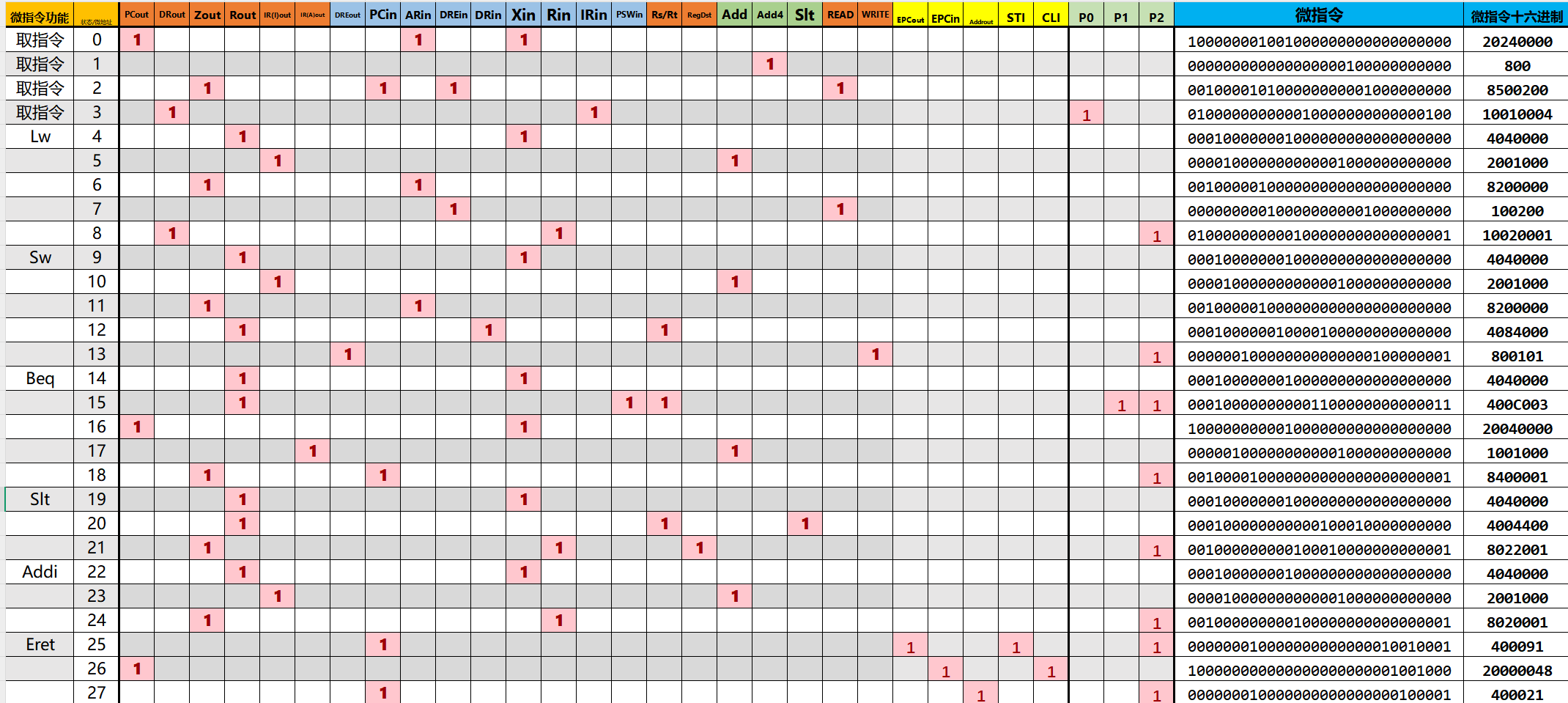


图 1-8支持中断的微程序自动生成表

### 微程序单总线CPU设计

本阶段的电路框架实验包已经给出，我们只需要实现中断逻辑部分，中断异常处理数据通路如图 1-9所示。本实验中，中断控制器的输出（编号）对应多路选择器的选择端口，两个类型中断1，2对应多路选择器的1，2端口，我们需要将中断入口地址作为常量连接到相应端口，利用三态门根据中断控制信号进行内总线的读取数据和输出数据，添加一个D触发器作为中断使能寄存器管理中断请求。EPC设计方面，使用32位寄存器存储断点地址，输入端（EPCin）和输出端（EPCout）均连接总线，输出端通过三态门控制，仅当EPCout=1时将值输出到总线。完成连线后的电路如所示。

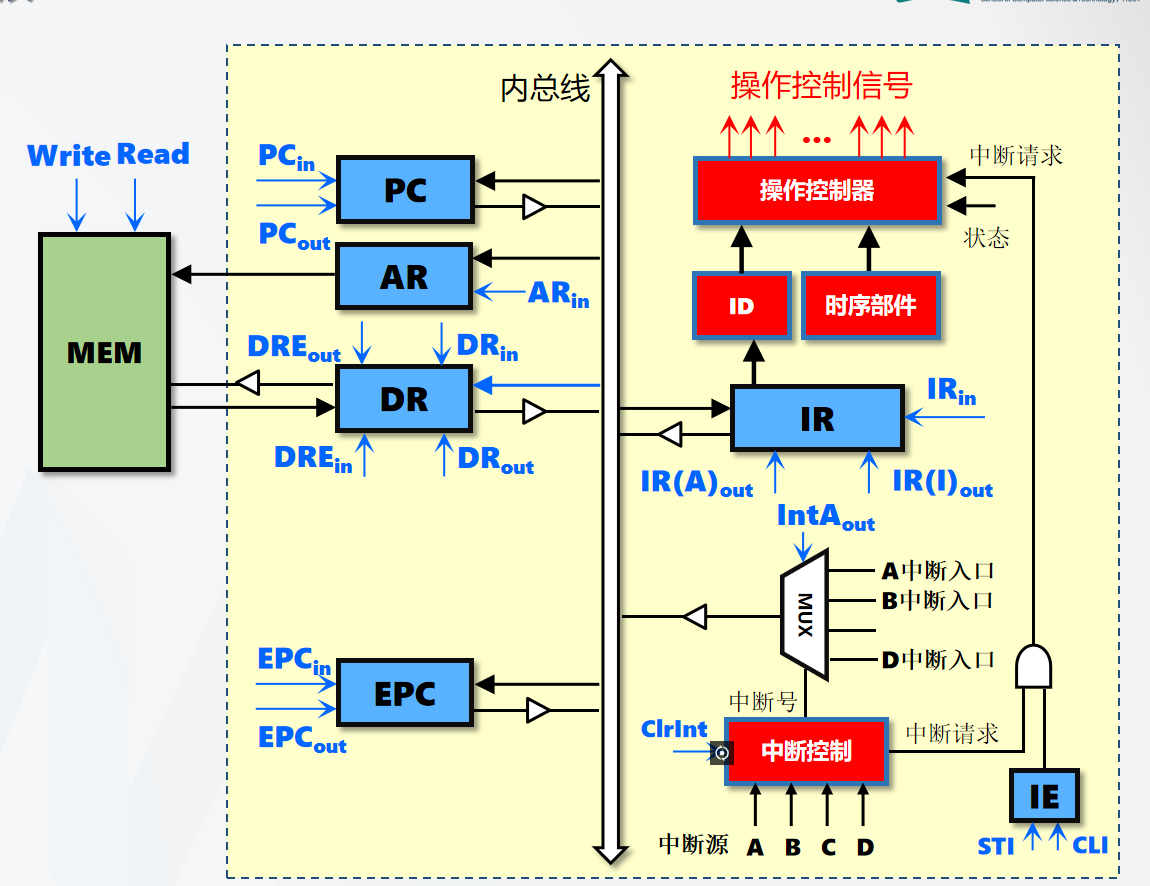


图 1-9 支持中断的单总线CPU数据通路示意图

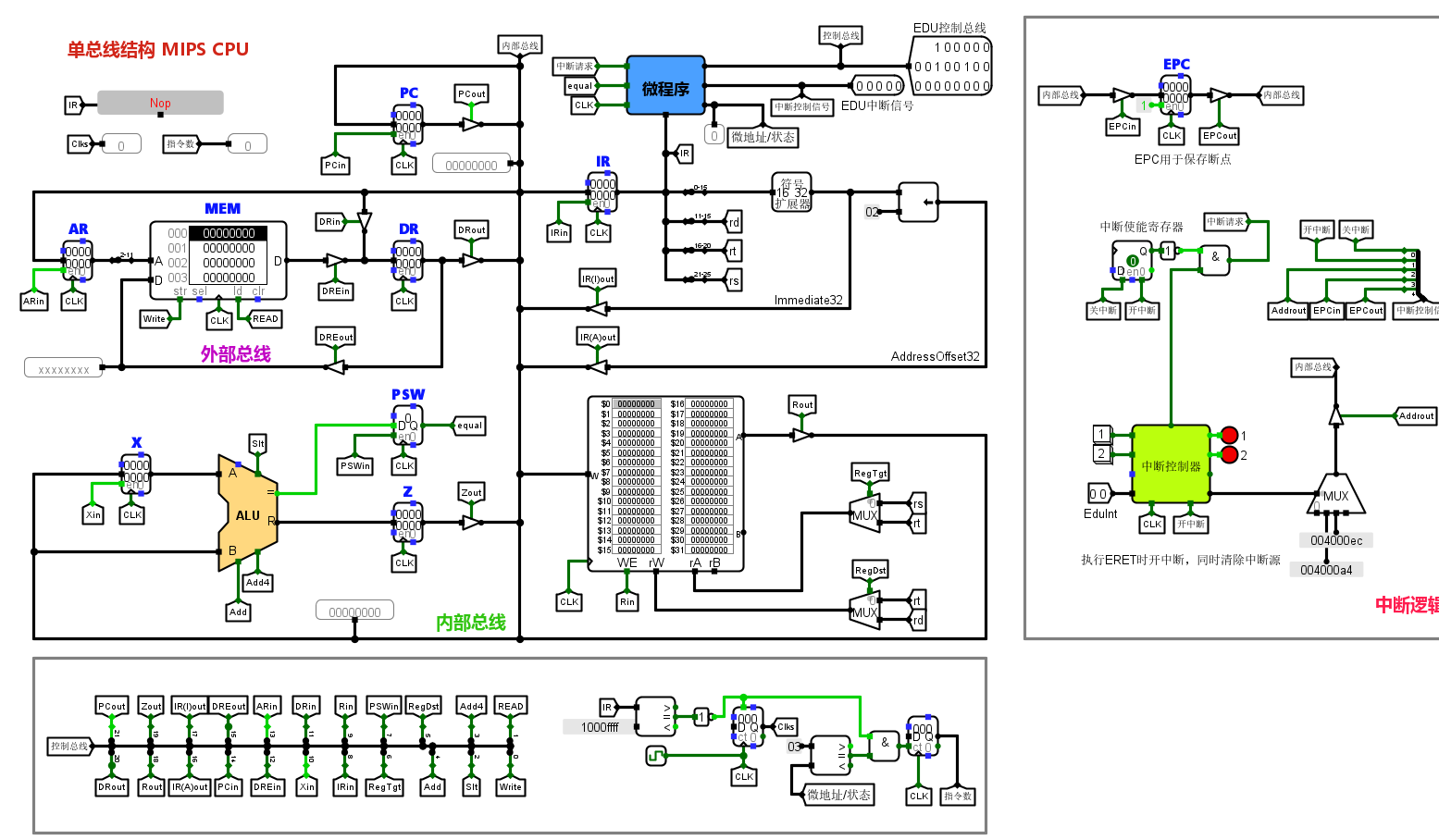


图 1-10支持中断的单总线MIPS结构CPU

### 硬布线状态机

打开子电路，其电路的引脚功能描述如表 1-5所示，由于大部分指令的前几个状态都是顺序执行，不涉及跳转，所以可以直接省略不填，比如S0的次态一定是S1，因此所有的指令译码信号列都不用填，而状态3需要根据指令功能进入不同状态，需要填写多行，所有指令的最后一个状态需判断是否有中断，从而决定是否进入中断处理状态S26,最终完成的表格如图 1-9所示。

表 1-5硬布线状态机电路的封装与引脚功能描述

|  |  |  |  |
| --- | --- | --- | --- |
| 信号 | 输入/输出 | 位宽 | 功能描述 |
| LW | 输入 | 1 | 当前指令为lw指令 |
| SW | 输入 | 1 | 当前指令为sw指令 |
| BEQ | 输入 | 1 | 当前指令为beq指令 |
| ADDI | 输入 | 1 | 当前指令为addi指令 |
| SLT | 输入 | 1 | 当前指令为slt指令 |
| equal | 输入 | 1 | 条件反馈信号 |
| IntR | 输入 | 1 | 中断请求信号 |
| S4-S0 | 输入 | 1 | FSM状态机现态 |
| N4-N0 | 输出 | 1 | FSM状态机次态 |

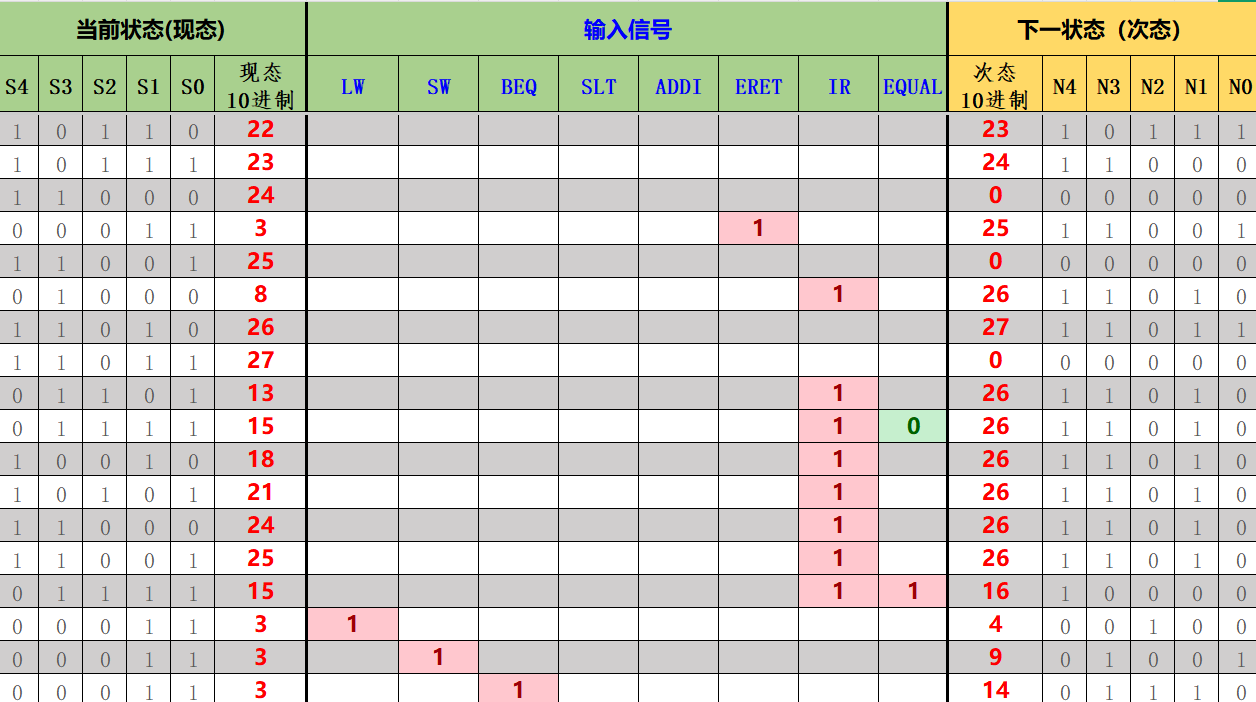


图 1-11支持中断的状态转换表

### 硬布线控制器

在实现指令译码、现代时序状态机模块后，最终需要实现硬布线控制器的集成，硬布线控制器组合逻辑不需要实现直接采用微程序控制器的控制存储器代替即可，只需要将输入信号和前面实现的状态机之间进行正确连线，输出连接到状态寄存器即可。最终电路见图 1-10。

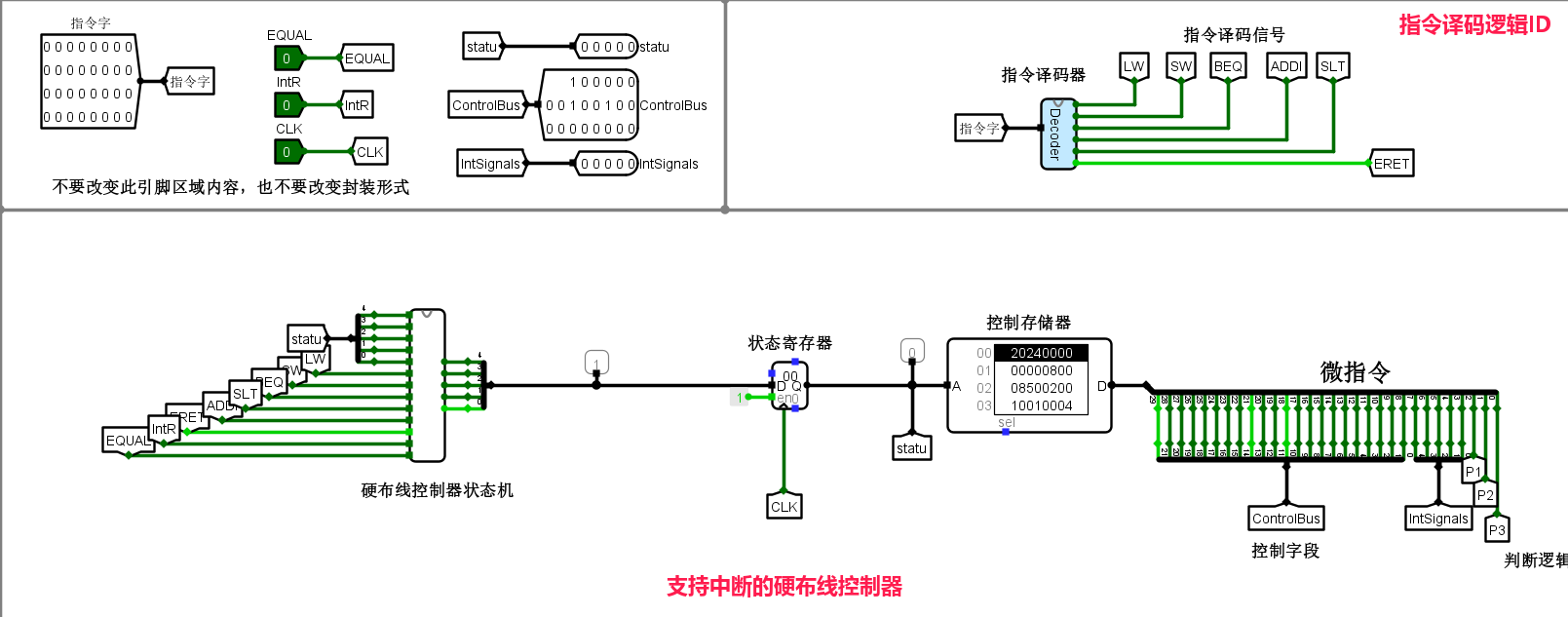


图 1-12硬布线控制器电路

## 实验步骤

1. 根据上一小节的方案设计，完成实验包中给出的各类自动生成表，得到各类输出信号的逻辑表达式和微指令，在Logisim中自动生成相应电路并进行连先，数据载入等操作。
2. 在1.2.5环节，还需要利用实验包中的预装软件Mars4\_5.jar，查看中断入口地址，具体方法为：命令行输入java -jar Mars4\_5.jar命令打开软件，加载sort-5-int.asm文件，在展示的标签窗口中查看相应地址。如图 1-13所示。将对应地址和多路选择器连接后，将asm文件载入RAM，打开时钟连续进行测试。
3. 完成各部分电路后，将文件circ文件内容复制到头歌平台进行自动测试。

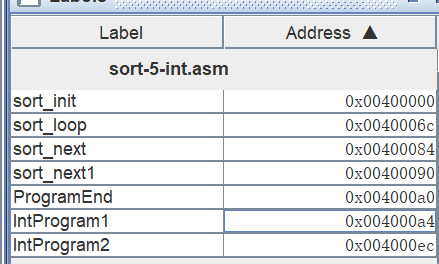


图 1-13 利用Mars4\_5查看中断入口地址

## 故障与调试

### 线路连接问题

**故障现象：**测试硬布线控制器时，发现头歌大量报错，在Logisim中测试，发现时钟到来后多条线路标红（出错），如图 1-14所示。

**原因分析：**在检查状态机逻辑和控制器连线后，发现输入端口5，6之间多了一条垂直的连线，导致出错。

**解决方案：**调整状态机位置，撤销分线器和输入信号和状态机的连线，发现输入端口之间多了一条连线，将其进行删除。

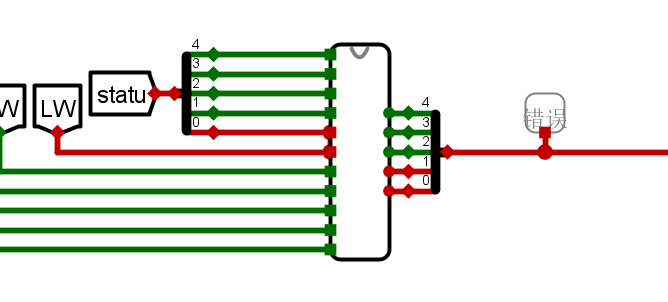


图 1-14硬布线控制器出错示意图

### 组合逻辑出错

**故障现象：**测试条件判别测试组合逻辑电路时，发现头歌多行用例报错，输出信号与预期不符。

**原因分析：**通过对照测试用例和真值表填写情况，发现对Equal和IntR的几种出现情况没有考虑周全，也没搞清楚其优先级关系

**解决方案：**重新填写真值表，减少留白单元格，对各种情况进行分析和填写，得到正确的逻辑表达式。

## 测试与分析

通过头歌平台测试，程序通过了头歌平台的所有测试用例，如图 1-15所示。同时，完成前面所有实验关卡以后，在1.2.5的电路中进行最终联调，测试CPU是否能正常响应2个按键对应的中断服务程序，在RAM中加载sort-5-int.hex程序，ctrl+k自动运行，运行至0x7c8节拍停下，指令计数为252，RAM内容如所示，可见第九行数据（080开头）已按照降序排序。若按下按键1，程序将在90开始的8个字单元全部加1，如按下按键2，程序将在a0开始的8个字单元全部减1。



图 1-15头歌测试完成图

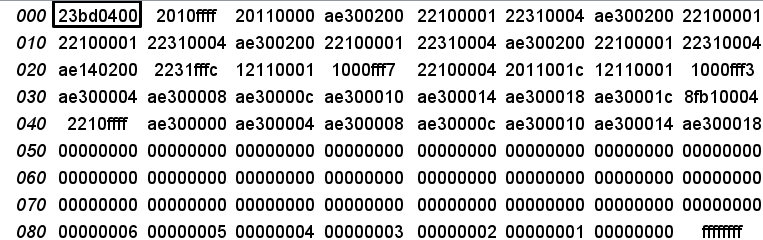


图 1-16运行排序程序后的RAM内容

## 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：本实验依次实现了指令译码器、微程序入口查找逻辑、条件判别测试组合逻辑、硬布线状态机等模块，并最终完成了支持中断的支持五条MIPS指令的现代时序处理器。其中，指令译码器通过组合逻辑实现MIPS指令的解析，微程序入口查找逻辑新增了ERET指令支持，条件判别测试逻辑整合了中断判别位P2

2）功能总结：实验电路顺利通过了头歌的测试验证。基础指令执行测试中，排序程序sort-5-int.hex能够正确运行至预期位置；中断功能测试验证了按键1和按键2的中断响应，包括数据修改、断点保存等功能。

## 实验心得

1. 实验收获：本次实验大大帮助了我对CPU这一章节理论知识的学习，虽然经过理论课的学习，但实验之前我其实对CPU的组成框架，数据通路，各种微指令的数据流都不是很清楚，实验引导我们依次完成各个模块，让我真正明白了指令执行流程、控制信号生成和异常处理等关键环节的实现原理。
2. 实验建议：实验某些模块的测试用例不是很全面，容易在后续引发错误，也容易让同学们面向测试用例编程，或许可以改进。

|  |
| --- |
| 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: losyi** |
| 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:** |