***2024***



**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： | CS2208 |
| 学 号： | U202215642 |
| 姓 名： | 田清林 |
| 电 话： | 15603480922 |
| 邮 件： | [1240067633@qq.com](mailto:1240067633@qq.com) |
| 完成日期： | 2024-06-07 |



目录

[1 CPU设计实验 2](#_Toc169885460)

[1.1 设计要求 2](#_Toc169885461)

[1.2 方案设计 2](#_Toc169885462)

[1.2.1 MIPS指令译码器设计 2](#_Toc169885463)

[1.2.2 支持中断的微程序入口查找逻辑 3](#_Toc169885464)

[1.2.3 支持中断的微程序条件判别测试逻辑 4](#_Toc169885465)

[1.2.4 支持中断的微程序控制器设计 5](#_Toc169885466)

[1.2.5 支持中断的微程序单总线CPU设计 6](#_Toc169885467)

[1.2.6 支持中断的现代时序硬布线控制器状态机设计 8](#_Toc169885468)

[1.2.7 支持中断的现代时序硬布线控制器设计 9](#_Toc169885469)

[1.3 实验步骤 10](#_Toc169885470)

[1.4 故障与调试 10](#_Toc169885471)

[1.4.1 微程序条件判别测试无法通过 10](#_Toc169885472)

[1.4.2 寄存器时钟上升沿/下降沿的选择 11](#_Toc169885473)

[1.5 测试与分析 12](#_Toc169885474)

[1.6 实验总结 13](#_Toc169885475)

[1.7 实验心得 14](#_Toc169885476)

# CPU设计实验

## 设计要求

利用logisim平台中现有运算部件以及框架设计一个支持中断的现代时序MIPS单总线CPU，并执行简单的sort\_int.asm程序。CPU支持的指令及其功能如表1‑1所示。

表1‑1 CPU支持的指令及其功能

|  |  |  |  |
| --- | --- | --- | --- |
| 指令名 | 类型 | 汇编代码表示 | 指令功能 |
| lw | I型 | lw rd,imm(rs1) | R[rd] ← M[R[rs1] + SignExt(imm)] |
| sw | I型 | sw rs2,imm(rs1) | M[R[rs1] + SignExt(imm)] ← R[rs2] |
| beq | I型 | beq rs1,rs2,imm | if(R[rs1] == R[rs2]) PC ← PC+4 + SignExt(imm) << 1 |
| slt | R型 | slt rd,rs1,rs2 | If (rs1 < rs2) R[rd] ← 1 else R[rd] ← 0 |
| addi | I型 | addi rd,rs1,imm | R[rd] ← R[rs1] + SignExt(imm) |
| eret | -- | eret | PC←EPC，IE←1 |

## 方案设计

### MIPS指令译码器设计

指令译码器是控制器核心功能部件，负责将指令字翻译成一根根的指令译码信号，每一根指令译码信号代表一条具体的指令。MIPS常见指令格式如图 1.1所示。



图 1.1 MIPS常见指令格式

根据表1‑1可知，LW、SW、BEQ、ADDI是I型指令，需要按照高六位的OP字段值进行判断，SLT为R型指令，需要OP = 0且FUNCT字段等于0x2a才可判断。各指令OP、FUNCT字段值如表 1‑2所示。

表 1‑2各指令OP、FUNCT字段值(无eret)

|  |  |  |
| --- | --- | --- |
| 指令名 | OP | FUNCT |
| lw | 100011 (0x23) | xxxxxx |
| sw | 101011(0x2b) | xxxxxx |
| beq | 000100(0x04) | xxxxxx |
| slt | 000000(0x00) | 101010(0x2a) |
| addi | 001000(0x08) | xxxxxx |

由于本实验中指令译码器只会接收到六个指令中的任何一条，因此指令不属于上述五条的指示引脚OtherInstr就可作为eret的指令引脚。

电路图如图 1.2所示。

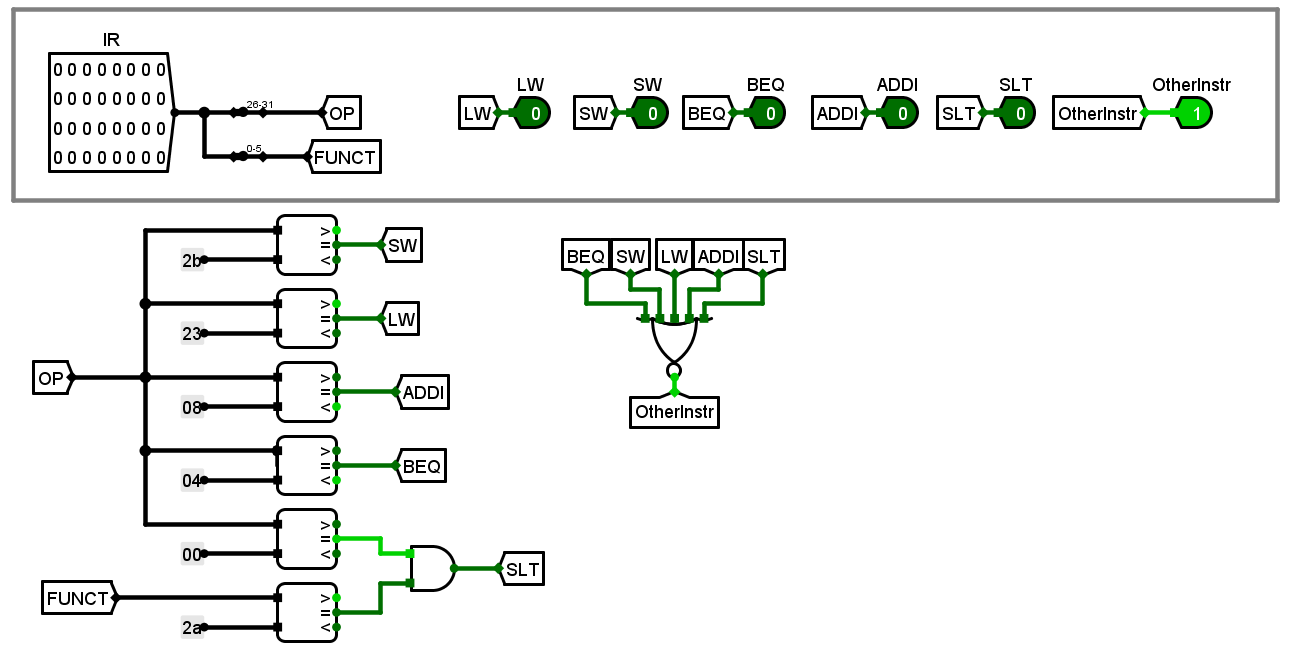


图 1.2MIPS指令译码器电路图

### 支持中断的微程序入口查找逻辑

根据地址转移逻辑（如图 1.3所示），在取值周期结束后需要根据译码信号进入合适的微程序继续执行，其微程序入口地址值与状态图中状态编号一致，填写相应excel表，如图 1.4所示。填表后自动生成逻辑表达式，将得到的逻辑表达式输入logsim的分析组合逻辑电路功能中即可自动生成逻辑电路。

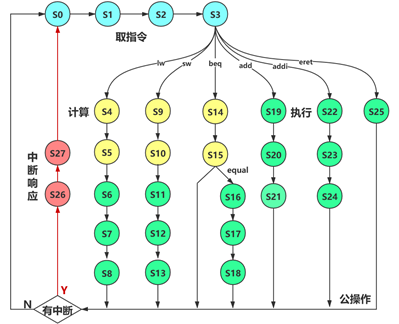


图 . MIPS指令执行状态转换图

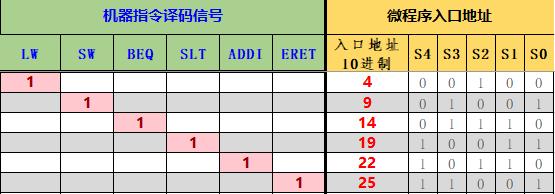


图 1.4微程序入口查找逻辑设计表

### 支持中断的微程序条件判别测试逻辑

条件判别测试主要是用来控制指令执行完毕后续地址的选择，分为计数器计算出的顺序地址、微程序入口地址、BEQ分支、中断响应程序地址、取指微程序地址，各后续地址对应状态编码如表 1‑3所示。

表 1‑3后续地址与状态编码对应表

|  |  |
| --- | --- |
| 后续地址 | 状态编码S |
| 顺序地址 | 000 |
| 微程序入口地址 | 001 |
| BEQ分支 | 010 |
| 中断响应程序地址 | 011 |
| 取指微程序地址 | 100 |

当判别测试字段信号全部为0时后续地址为计数器计算出的顺序地址，当P0 为1时跳转到微程序入口地址；在P1测试信号为1时，若equal为1则跳转到beq分支，否则跳转到取址微程序地址。

将对应值填入excel表格，如图 1.5所示。之后按照逻辑表达式生成电路。

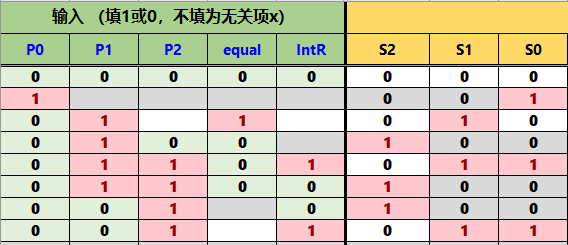


图 1.5条件判别逻辑组合逻辑设计表

### 支持中断的微程序控制器设计

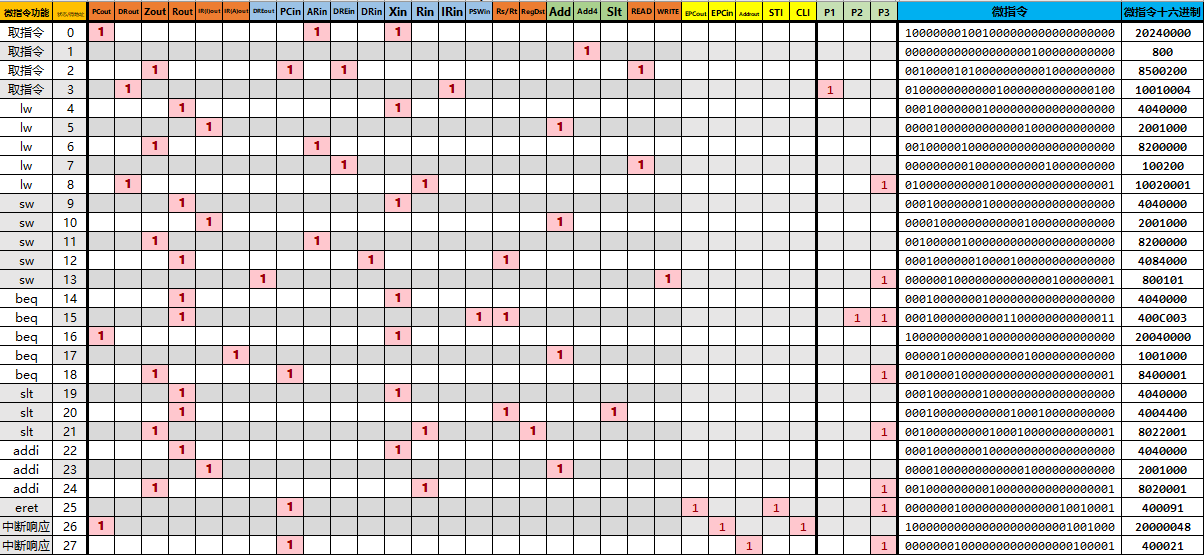


图 1.6微指令设计表

首先需要进行微程序的设计。本次实验使用直接表示法和计数器法对微指令进行编码。根据表1‑1给出的6个指令功能，分析每个微程序在每个时钟周期中需给出的操作信号以及顺序控制字段的值，填写excel表如图 1.6所示，生成微指令后，将其填入控制存储器。

之后需要正确连接控制器电路。判别测试逻辑的输出作为入口地址多路选择器的选择端。电路图如图 1.7所示。

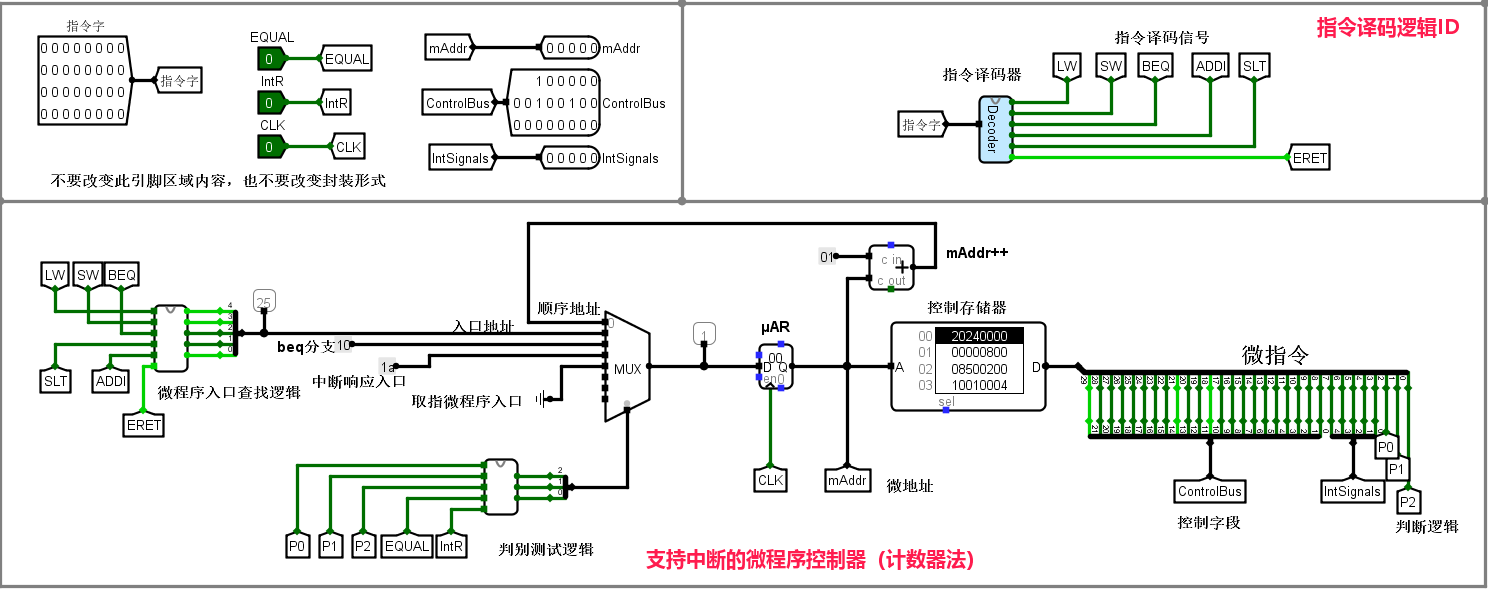


图 1.7现代时序带中断微程序控制器电路图

多路选择器的各输入中，0号代表顺序地址，是当前微地址mAddr加1后的值；1号是微程序入口地址，由微程序入口查找逻辑给出各指令微程序入口地址；2号是beq分支，是状态转换图（如图 1.3所示）中S16对应微程序地址，当处于S15状态时且equal等于1时，会进入S16状态；3号是中断响应入口地址，是状态转换图（如图 1.3所示）中S26对应微程序地址，当指令执行完毕且有中断时会跳转到的位置；4号是取址微程序入口地址，是状态S0的对应微程序地址。

### 支持中断的微程序单总线CPU设计

单总线CPU的其他部件已在框架中给出，这里只需完成中断逻辑相关电路（如图 1.8所示）。在不支持中断的单总线CPU中加入中断，需要加入开/关中断、保存断点和中断识别功能。

1. 开/关中断

开/关中断的操作需增加IE寄存器。图 1.8中的D触发器相当于IE寄存器，当输出值为1时才可处理中断，开中断、关中断时输出分别为1和0，接异步置0和置1端（触发器输出是~Q）。将IE寄存器输出与中断控制器的中断信号接入与门的输入，输出即是中断请求信号。

1. 保存断点

EPC用来保存断点地址，是一个寄存器，输入与输出都接到内部总线，并接入三态门，由EPCin和EPCout信号分别控制写入和输出，如图 1.8所示

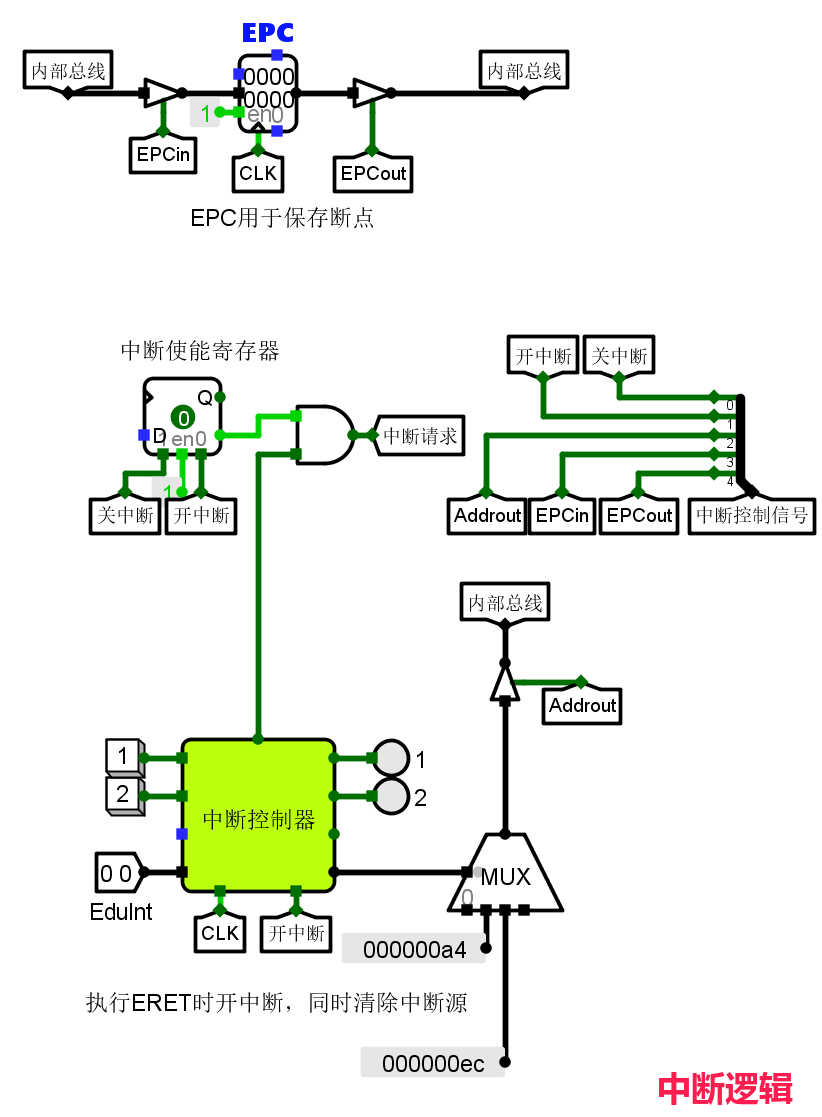


图 1.8 CPU中断逻辑电路图

1. 中断识别

中断控制器会检测中断并输出中断号，将中断号接入多路选择器的选择端，中断1和2地址分别接入多路选择器的1、2端口，中断入口地址需要用Mars4\_5.jar程序查看。

使用Mars4\_5.jar打开Sort-5-int-riscv.asm，在Settings勾选对应选项,如图 1.9所示。

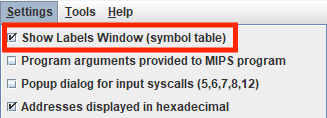


图 1.9 Settings中勾选对应选项示意图

在Run菜单中选择Assemble，即可查看对应中断程序的入口地址，如图 1.10所示。注意这里给出的IntProgram1和IntProgram2的地址需要减去sort\_init值，结果分别为0xa4和0xec。将对应地址用常量输入多路选择器的端口1和2。多路选择器的输出接入一个三态门连接到内部总线，由Addout信号控制数据的输出。

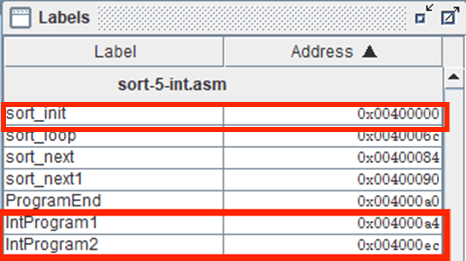


图 1.10对应中断程序的入口地址

### 支持中断的现代时序硬布线控制器状态机设计

根据状态机的状态转换条件（如图 1.3所示），填写excel表，如图 1.11所示。

对于取址指令的S3状态，之后需要根据输入的LW、SW、BEQ、SLT、ADDI、ERET等译码信号进入对应的微程序入口。

对于指令的最后一个状态，根据IR值来判断是否处理中断，若IR为1则进入中断处理周期，IR为0则回到取指程序。

对于BEQ指令的S15状态，若EQUAL为1则进入S16状态，若EQUAL为0该状态为指令的最后一个状态，需要根据IR值判断处理中断还是回到取指程序。

填写完后，将自动生成的逻辑表达式输入logisim自动生成电路即可。

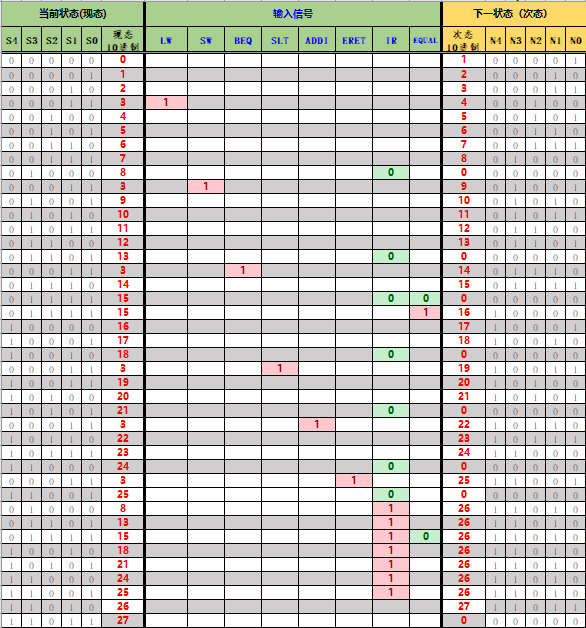


图 1.11 硬布线控制器状态机状态转换表

### 支持中断的现代时序硬布线控制器设计

将当前状态信息statu与各译码信号和控制信号作为硬布线控制器状态机的输入，将它的输出（次态）作为状态寄存器的输入，状态寄存器的输出为现态。由于没有编写各控制信号的组合逻辑表示，在控制存储器中放入先前的微程序，来控制各控制信号和顺序控制字段的值。电路图如图 1.12所示。

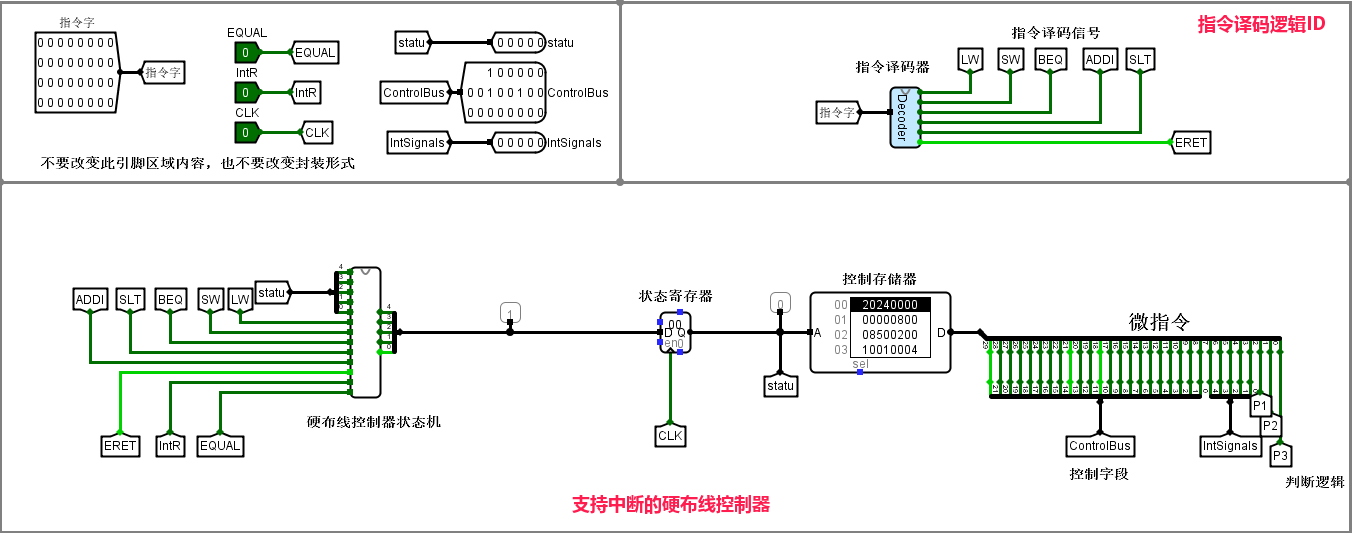


图 1.12支持中断的现代时序硬布线控制器电路图

## 实验步骤

1. 分析实验各子模块的作用及功能，理解支持中断的单总线CPU的结构和运行原理。
2. 完成各个子模块的实现，可以借助实验附带的表格生成逻辑表达式，根据自动生成的表达式自动生成电路。
3. 将各个子电路进行组装联调，构建最终的CPU电路。
4. 将logisim代码复制到Educoder进行测试，若未通过则寻找故障并调试。

## 故障与调试

### 微程序条件判别测试无法通过

**故障现象：** 在Educoder测试后，当输入依次为01101时，输出状态为4，错误。

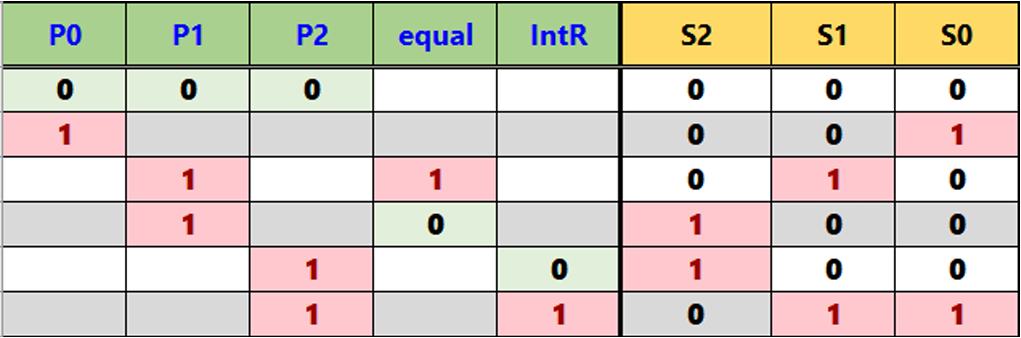


图 1.13条件判别逻辑组合逻辑设计表（错误）

**原因分析：** 首先，该excel表格空格处为无关项x，当编码出现二义性时，以靠前的行结果为准。判别逻辑组合逻辑设计表填写错误，没有考虑到P1和P2都为1时的优先级问题。以输入依次为01101为例，此时P1和P2都为1，说明既要参考equal也要参考IntR， equal为0而IntR为1，说明不会进入beq分支而是进入中断响应周期（状态编码为011），但按照填写的设计表，会进入取指微程序（状态编码为100）。

**解决方案：** 改正条件判别逻辑组合逻辑设计表。如图 1.5所示。

### 寄存器时钟上升沿/下降沿的选择

**故障现象：**微程序控制器在Educoder测评时输出结果大面积混乱，cnt值每个值都会出现两次。电路图如图 1.14所示，Educoder测试故障如图 1.15所示。

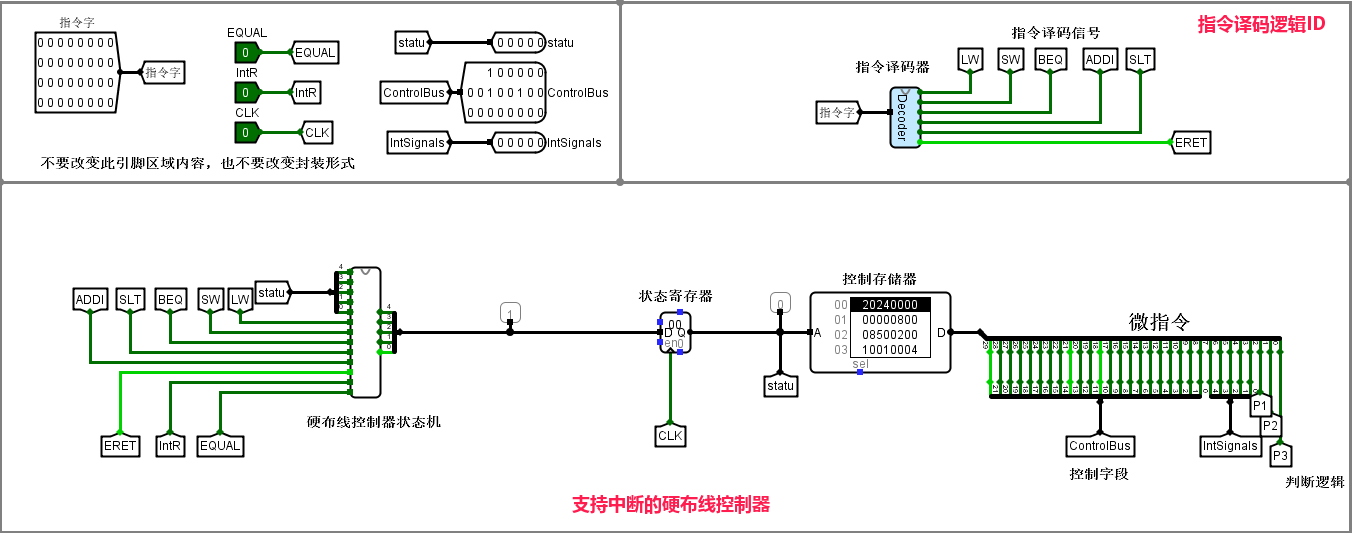


图 1.14微程序控制器电路图



图 1.15 微程序控制器Educoder测试结果

**原因分析：**微指令寄存器 μAR设置成了上升沿触发。在时钟的上升沿，CPU进行各个寄存器的锁存操作，在触发器翻转前后这段时间内，触发器的输入应该保持稳定。然而，一旦微指令寄存器设置为上升沿触发，微指令也在时钟上升沿发生变化；而微指令中的控制信号直接管制着各锁存器的输入，因而使得触发器不能可靠翻转。

**解决方案：**将微指令寄存器设置为下降沿触发，从而错开数据锁存和状态切换的过程，保证触发器可靠翻转。

## 测试与分析

Educoder的评测关全部通过。对于支持中断的微程序单总线CPU，将sort-5-int.hex放入内存并运行，无中断情况下运行结果如图 1.16所示，可以看到从内存0x80开始的降序序列。

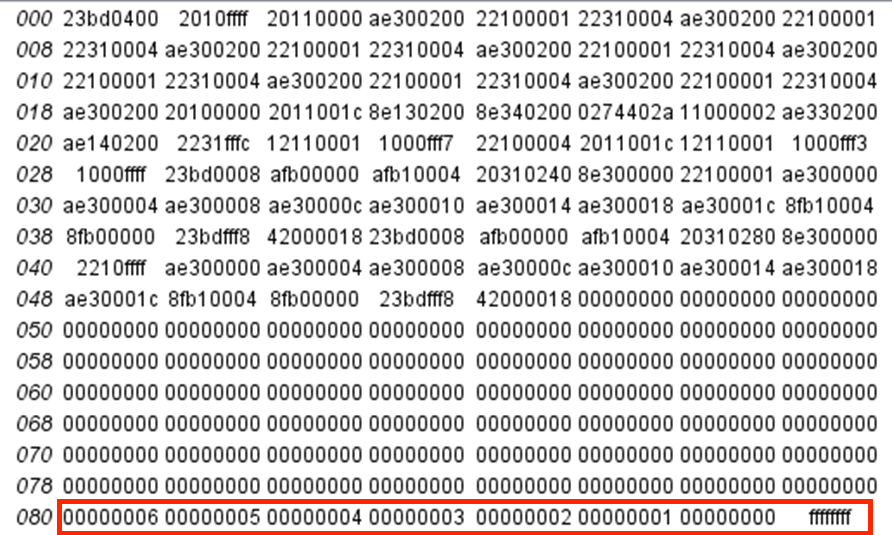


图 1.16现代时序CPU无中断运行结果

支持中断的现代时序CPU在一次按键1中断的情况下的测试结果如图 1.17所示，可以看到 0x90 开始的八个字节内容全部加 1。

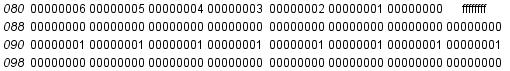


图 1.17现代时序CPU运行1次1号中断结果

两次按键1号中断和两次2号中断的情况下测试结果如图 1.18所示，可以看到0x90 开始的八个字节内容全部加 2，从0xa0开始的八个字节全部减2（补码表示为0xfffffffe）。可见该CPU可正确支持中断响应。

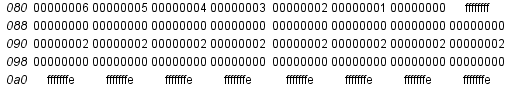


图 1.18现代时序CPU运行2次1号中断和2次2号中断结果

注意到中断请求2发出后，中断控制器右侧对应中断号2的红灯亮了一段时间，如图 1.19所示。说明中断1正在被处理。此时应有，这个单级中断系统在这段时间内将屏蔽任何中断请求；因此在中断请求未响应完成前再次触发中断1或2都不会处理新的中断请求。

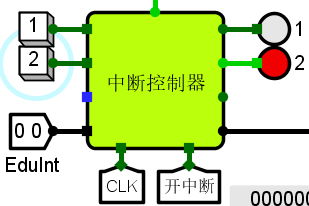


图 1.19 处理请求时的中断控制器

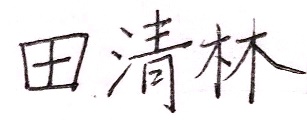
## 实验总结

本次实验主要完成了如下几点工作：

1. 设计了支持6条简单指令的MIPS定长指令集的指令译码器、微程序入口查找逻辑电路、条件判别测试逻辑电路以及保存微程序的控制存储器。
2. 设计了微程序的编码。
3. 设计了基于现代时序的硬布线状态机。
4. 完成了现代时序硬布线控制器、微程序控制器的设计。
5. 实现了基于硬布线控制器、微程序控制器的单总线CPU。
6. 实现了CPU对中断的处理。
7. 成功运行了一个简易的冒泡排序程序。

## 实验心得

1. 通过本次实验，我了解并逐渐熟练应用logisim中各个部件，也熟悉了之前从未使用过的陌生器件，如多路选择器、译码器、三态门等，并复习了上个学期中数字电路和逻辑设计的知识。
2. 加深了对单总线CPU的结构的理解。CPU是进行数据加工、程序控制、操作控制、时序控制、异常控制的部件，其中所有的控制操作都由控制器完成。当课上讲执行指令过程中CPU的数据流和控制流时，我一边感叹CPU控制器的精妙和复杂，一边苦恼如何实现能执行那么多指令的CPU。本次实验虽然只实现了支持6条指令的CPU，但引导我找到了合适的设计方法和策略，对于各部件的作用和他们整体的结构也更加得心应手了。
3. 加深了对单总线CPU程序运行方式的理解。指令执行一般包括取指令、指令译码、操作数地址计算、取操作数、执行指令、操作数地址计算、存操作数，对于本次实验支持中断的CPU来说，还要根据是否有中断决定是否需要进入中断响应周期。这一部分在课程学习过程中一直觉得理解不到位，直到在实验中通过实践才加深了理解。
4. 计算机组成原理作为计算机科学与技术专业核心课程之一，在学习过程中我对计算机有了更深的认识与理解，每每看到精妙的计算机设计不禁被前人的智慧折服。平时的理论学习为我的实验打基础，实验更是在实践中对理论知识的强化，也暴露了我很多理解不到位的地方。“知者行之始，行者知之成。”大概说的就是这样吧。
5. 本次实验引导较为细致，大大减少了实验所需的时间，但也具有一定的局限性。实验中体力劳动内容不少而思考内容不够多，希望多添加一些思考问题，辅助理解；内容上只涉及了单总线的CPU，而多总线CPU在如今计算机结构中也具有广泛的应用场景和较高的地位，希望可以加入部分相关内容。

****

|  |
| --- |
| 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: 田清林** |
| 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:** |