***2021***



**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： |  |
| 学 号： |  |
| 姓 名： | [作者] |
| 电 话： |  |
| 邮 件： |  |
| 完成日期： | 2021-12-16 |



目 录

[1 单总线RISC-V CPU设计（变长指令周期3级时序） 2](#_Toc90893329)

[1.1 设计要求 2](#_Toc90893330)

[1.2 方案设计 2](#_Toc90893331)

[1.3 实验步骤 6](#_Toc90893332)

[1.4 故障与调试 6](#_Toc90893333)

[1.5 测试与分析 6](#_Toc90893334)

[2 RISC-V 现代时序中断机制实现 8](#_Toc90893335)

[2.1 设计要求 8](#_Toc90893336)

[2.2 方案设计 8](#_Toc90893337)

[2.3 实验步骤 14](#_Toc90893338)

[2.4 故障与调试 14](#_Toc90893339)

[2.5 测试与分析 15](#_Toc90893340)

[3 总结与心得 17](#_Toc90893341)

[3.1 实验总结 17](#_Toc90893342)

[3.2 实验心得 17](#_Toc90893343)

[参考文献 18](#_Toc90893344)

# 单总线RISC-V CPU设计（变长指令周期3级时序）

## 设计要求

利用logisim平台中现有的框架依次完成RISC-V指令译码器的设计、时序发生器FSM的设计、时序发生器输出函数的设计、硬布线控制器组合逻辑单元的设计、硬布线控制器的设计，并在最后完成联调，测试排序程序。在RAM中加载sort-5-riscv.hex程序。Ctrl+k自动运行，程序应该正常运行至0x81d节拍停下，指令计数为251（最后一条指令是一条beq分支指令，会跳回当前指令继续执行，将是死循环）。

## 方案设计

### RISC-V指令译码器设计

（1）输入信号：

IR：Riscv指令字。

（2）输出信号：

LW、SW、BEQ、ADDI、SLT：操作信号；

Otherinstr：不属于上述五个操作的其他信号。

（3）设计原理：指令码在部分位数的值是固定的，我们只要将IR值的识别部分与固定值的相对部分相比对，就能够区分不同的操作信号，利用logisim库中的比较器，我们能很轻易地实现这个区分功能。

（4）设计思路：由给出的RISC-V指令码表，我们能够发现各个指令的识别关键部分主要是在2-6位、12-14位和25-31位，所以将这三段提出保存为OP、FUNCT3、FUNCT7，其中OP和FUNCT3组合成OP\_Funct3。利用比较器，将OP\_Funct3和指令相对部分的值进行比对，“=”引脚输出即为指令；而五个指令经过或非门的输出即为Otherinstr。

### 变长指令周期——时序发生器FSM设计

（1）输入信号：

S3、S2、S1、S0：现态输入；

LW、SW、BEQ、ADDI、SLT：指令选择。

（2）输出信号：

N3、N2、N1、N0：次态输出。

（3）设计原理：虽然单总线结构中，使用变长指令周期时，不同指令的机器周期数不同、每个机器周期节拍数也是可变的，但是他们都是确定的，只要根据相应的状态图，就能够得知不同现态下不同指令输入后的次态了。

（4）设计思路：根据变长指令周期的状态图，状态图如下：

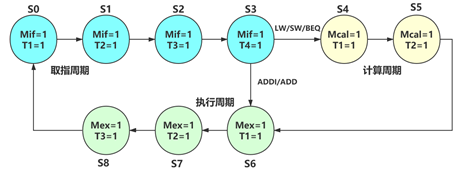


图 1.2.1 三级时序状态图

在提供的三级时序产生器逻辑自动生成excel表格中，只要按照状态图填写出现态以及输入信号以及对应的次态，就可以在excel中的触发器输入函数自动生成表格中获得次态表达式。在logisim分析组合逻辑电路中直接填写就能够生成电路。

其中最需要注意的就是现态3到4的转换，不同的输入信号会到不同的次态，需要区分。

### 变长指令周期——时序发生器输出函数设计

（1）输入信号：

S3、S2、S1、S0：现态输入。

（2）输出信号：

Mif、Mcal、Mex：周期电位输出；

T1、T2、T3、T4：节拍电位输出。

（3）设计原理：根据不同的现态来确定不同的状态电位和节拍电位，而后两者的取值正如图1.2.1中所展示的，只要根据该状态图就可以知道每个现态值对应的输出了。

（4）设计思路：与1.2.2相似的。在三级时序产生器逻辑自动生成excel表格的输出函数真值表中填写现态的值，并根据三级时序状态图，填写对应的输出，就可以在输出函数自动生成表格中获得输出信号最终的表达式，复制到logisim中的分析组合逻辑电路中就能自动生成电路图了。

### 硬布线控制器组合逻辑单元

（1）输入信号：

Mif、Mcal、Mex：周期电位输入；

T1、T2、T3、T4：节拍电位信号输入；

LW、SW、BEQ、SLT、ADDI：指令译码信号输入；

equal：相等状态标志位，反馈信息。

（2）输出信号：

PCout…：控制器信号输出。

（3）设计原理：硬布线控制器组合逻辑单元就是将反馈信号、指令译码信号、状态周期电位、节拍电位转化为微操作控制信号序列。每一条指令序列在不同时钟周期产生的控制信号序列都是一定的，所以，我们只要根据输入，结合逻辑便能知道输出信号序列。

（4）设计思路：每一种输入都有相对的输出，所以在三级时序控制器控制信号逻辑自动生成excel表中填写将状态电位、节拍电位、指令译码信号、反馈信号逐个列出，并由此写出每个控制信号序列，如下图：



图 1.2.2 三级时序状态图

之后便可在输出信号表达式自动生成表格中获得每个输出控制信号的逻辑表达式，再复制到logisim中的分析组合逻辑电路中就能自动生成电路图了。

### 变长指令周期——硬布线控制器设计

（1）输入信号：

IR：Risc-V指令字。

（2）输出信号：

Mif、Mcal、Mex、T1、T2、T3、T4：状态节拍信号；

PCout…：控制总线。

（3）设计原理：硬布线控制器的组成部分主要包括指令译码逻辑、时序发生器、以及硬布线控制器组合逻辑单元：

指令译码逻辑就是将IR译码，得到指令信息作为时序发生器以及硬布线控制器的输入信号；

时序发生器根据时钟脉冲信号持续不断地产生状态周期电位以及节拍电位，主要由状态寄存器、FSM状态机以及输出函数构成。这是我们需要完成的部分，完成之后能够使得时序发生器正常运作；

硬布线控制器组合逻辑单元是根据时序发生器所给出的时序，结合译码信号、反馈信号得出控制总线的各信号。

（4）设计思路：实验文件中指令译码逻辑、硬布线控制器组合逻辑都已完成，我们只需要完成时序发生器的连线。时序发生器要能够输出现态并保存次态，因此我们利用寄存器完成这样的功能。寄存器的输出置为状态机和输出函数的输入，状态机的输出返回寄存器进行保存。最终实现的电路效果如下：

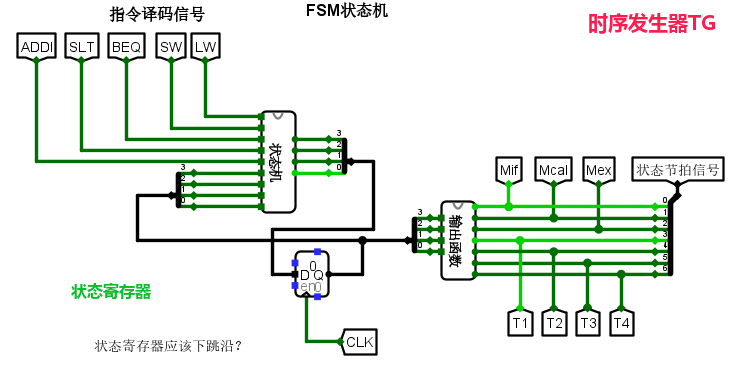


图 1.2.3 时序发生器TG连接图

### 变长指令周期——单总线CPU设计

（1）输入信号：无

（2）输出信号：

ControlBus：控制总线，用于EduCoder系统自动测试。

（3）设计原理：这只是将前面的5个实验联调得到的一个完整的CPU，需要完成的只是在RAM中加载指令即可。

（4）设计思路：在RAM中加载sort-5-riscv.hex程序，自动运行，检查是否执行正确，之后放入educoder即可验证。

## 实验步骤

1. 学习实验，阅读文档、观看视频，初步了解实验的目的以及过程；
2. 根据educoder的试验任务，从简到难逐个完成各个部件的设计，通过平台的测验；
3. 将各个部件组装为总控制器。在RAM中加载排序程序sort-5-riscv.hex，运行CPU检查有无异常；
4. 将内容传至educoder对最终实验进行测试。

## 故障与调试

### 时序发生器输出异常

**故障现象：logisim上的现象正常，但是在educoder中检验时发现有两次重复的输出**。

**原因分析：**最初以为是寄存器的问题，但是在logisim上经过检查，好像并没有问题；于是考虑是否是educoder或者logisim本身存在的问题。

**解决方案：**通过重启educoder和logisim，再将logisim文件传至educoder上进行检查，便通过了。

## 测试与分析

在RAM中加载sort-5-riscv.hex文件，加载内容后如下：

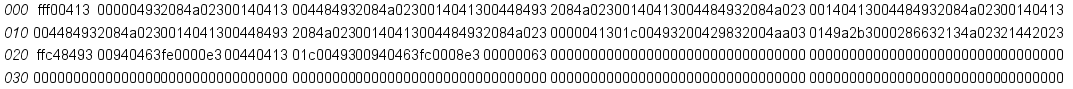


图 1.5.1 RAM内容最初显示

启动时钟进行测试，直到CPU运行结束：

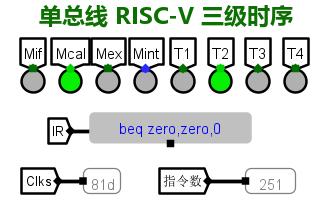


图 1.5.2 运行结束时时钟周期数和指令数显示

而最终完成排序后，RAM中的内容就变成了：

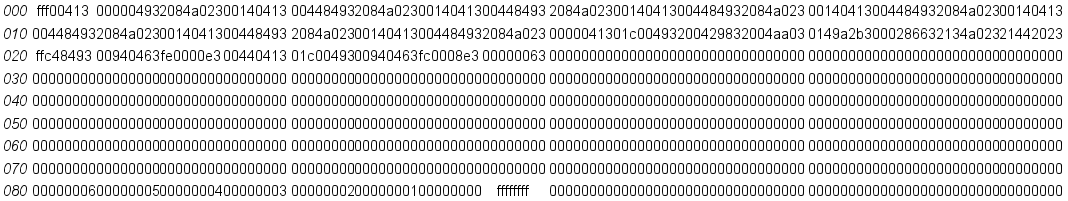


图 1.5.3 运行结束后RAM内容

我们可以发现在080排，有着6到-1的递减排序，那么这样我们就能够知道我们的整个功能是正常完成的。再将文件放入educoder平台，测试成功。



图 1.5.2 运行结束时时钟周期数和指令数显示

# RISC-V 现代时序中断机制实现

## 设计要求

本实验需要完成现代时序微程序控制器，在其基础上增加硬件数据通路、中断返回指令meret的支持，需要中断服务程序配合。

利用logisim平台中现有的框架依次完成RISC-V指令译码器设计、支持中断的微程序入口查找逻辑、支持中断的微程序条件判别测试逻辑、支持中断的微程序控制器设计、支持中断的微程序单总线CPU设计，在RAM中加载sort-5-riscv.hex程序，Ctrl+k自动运行，CPU能够完成简单的排序程序；接着完成支持中断的现代时序硬布线控制器状态机设计、支持中断的现代时序硬布线控制器设计，对CPU中的微程序控制器进行替换，完成联调后测试，同样应该能够完成简单的排序程序。

## 方案设计

### RISC-V指令译码器设计

（1）输入信号：

IR：Riscv指令字。

（2）输出信号：

LW、SW、BEQ、ADDI、SLT：操作信号；

Otherinstr：不属于上述五个操作的其他信号。

（3）设计原理：其实与1.2.1中的设计是相同的，不再分析。

（4）设计思路：与1.2.1中的设计思路相同，只是在本次实验中，Otherinstr这个输出会成为中断信号MERET。

### 支持中断的微程序入口查找逻辑

（1）输入信号：

LW、SW、BEQ、ADDI、SLT：指令译码信号；

ERET：MERET指令译码信号。

（2）输出信号：

S4、S3、S2、S1、S0：微程序地址入口地址。

（3）设计原理：微程序入口查找逻辑中，输出信号是在P0=1时，多路选择器从中选出下次跳转的微程序地址。而P0=1只有在取指令最后一条微指令中才会出现，所以，查找逻辑的结果就是取指微程序执行之后下一个微程序的入口地址，也因此我们便能够知道应该如何设计了。

（4）设计思路： 只要根据不同德指令信号，给出微程序入口地址就行，入口地址我们可以根据下面的状态转化图得出：

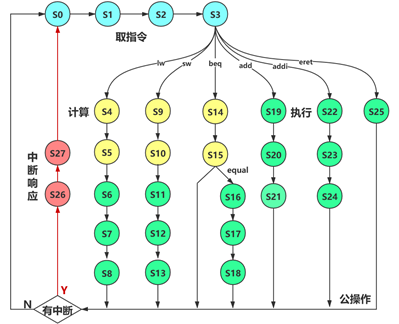


图 2.2.1 支持中断的现代时序状态转化图

将译码信号和对应的入口地址填入微程序地址转移逻辑自动生成excel中，在微程序入口查找逻辑自动生成表格中就可以得到逻辑表达式，填入logisim就可以自动生成电路了。

### 支持中断的微程序条件判别测试逻辑

（1）输入信号：

P0：判别测试位，为1表示根据指令功能进行微程序分支；

P1：判别测试位，为1表示根据equal标志进行微程序分支；

P2：判别测试位，为1表示这是微程序的最后一条微指令，可能需要进行中断响应；

equal：条件状态位，表示运算相等；

IntR：中断请求信号。

（2）输出信号：

S2、S1、S0：微程序地址。

（3）设计原理：条件判别逻辑的输出是多路选择器的选择信号，我们可以根据知识得到不同选择条件下多路选择器应该选择的信号，并由此得到不同输入条件下条件判别逻辑应有的输出。

（4）设计思路：

当P0=1时，应该选择微程序入口逻辑的输出信号；

当P1=1时，如果equal=1，需要进行跳转，否则进行公操作判断；

当P2=1时，表明微程序执行结束，接下来进行公操作，当存在IntR信号时，进行中断响应，否则进行下一次取值。由此填入微程序判别逻辑自动生成表达式excel表中，如下图：

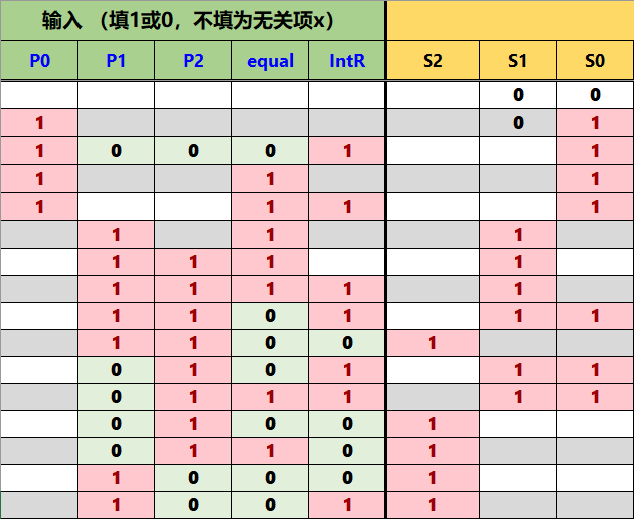


图 2.2.2 组合逻辑真值表

将输出信号表达式自动生成表格中的信号逻辑表达式填入logisim，便可自动生成电路。

### 支持中断的微程序控制器设计

（1）输入信号：

IR：RISC-V指令字；

equal：条件反馈信号，表示运算相等；

IntR：中断请求信号。

（2）输出信号：

mAddr：当前微指令地址；

ControlBus：微指令字的控制字段。

（3）设计原理：只需要知晓每一条微程序之后需要跳转到哪即可。然后就是对于各部件进行线路连接，通过对计数法寻址的学习，我们便能比较轻松地完成连线的操作。

（4）设计思路：只要根据指令及有限状态机填写支持中断微程序逻辑自动生成excel表中的微程序地址入口表，便能获得微指令十六进制编码，如下：

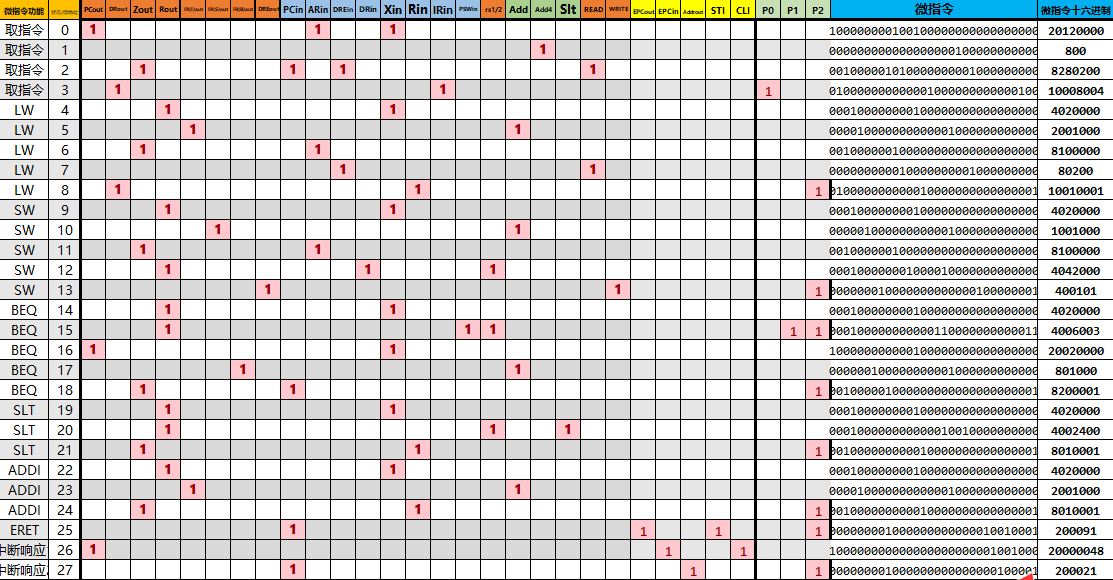


图 2.2.3 微程序逻辑自动生成表

保存到相应的控制存储器中。再进行电路连接，使微程序控制器能够正常工作，连接图如下：

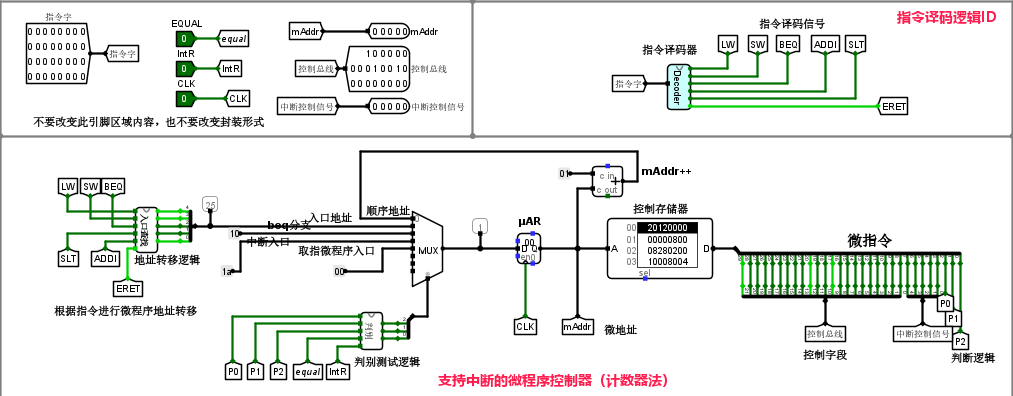


图 2.2.4 微程序控制器+中断实现图

### 支持中断的微程序单总线CPU设计

（1）输入信号：无。

（2）输出信号：无。

（3）设计原理：CPU的执行，其实与现代时序没有什么大差别，主要关注中断逻辑。先要判断是否为开中断，如果是开中断，需要保存现场，再将PC值送入EPC，之后根据不同的中断号选择不同的中断地址程序进行执行；而如果为关中断，则不必发送中断请求。

整个CPU是在单总线数据通路中增加了异常程序地址计数器、中断使能寄存器、中断控制器。注意EPC，进入中断响应阶段时，mEPCin=1时，使断点值保持不变；中断响应结束时，mEPCout=1，将断点地址传输到PC中。而IE在关中断时置1，开中断时置0，输出与寄存器保存内容相反的值。

（4）设计思路：只要理解了中断的逻辑，便可以根据逻辑连接电路，得到最终需要的效果，连线如下：

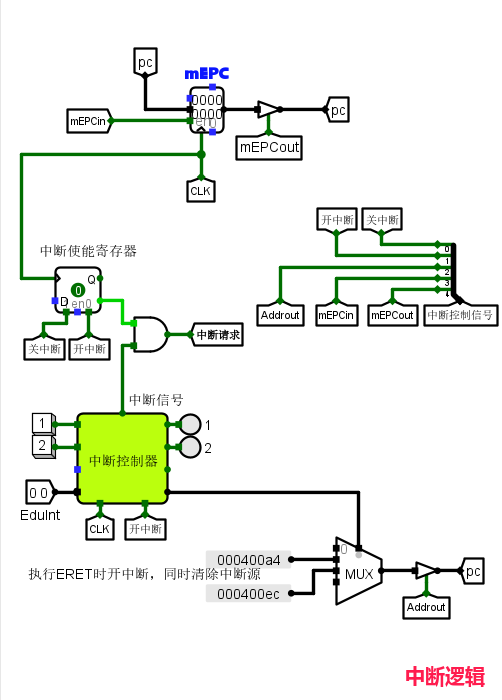


图 2.2.5 中断逻辑实现图

### 支持中断的现代时序硬布线控制器状态机设计

（1）输入信号：

S4、S3、S2、S1、S0：现态输入；

LW、SW、BEQ、ADDI、SLT：译码指令；

equal：条件反馈信号；

IntR：中断请求信号。

（2）输出信号：

N4、N3、N2、N1、N0：次态输出。

（3）设计原理：原理其实与三级时序的设计原理相似，不过多介绍。只要注意增加的URET执行状态、中断响应状态，不同的分支需要注意。

（4）设计思路：在硬布线控制器状态机逻辑自动生成excel表格中逐个填写状态转换表，便可以在表达式自动生成表格中获得五个次态输出的逻辑表达式，填入logisim即可自动生成电路。

### 支持中断的现代时序硬布线控制器设计

（1）输入信号：

IR：Risc-V指令字；

CLK：时钟输入；

equal：条件状态信号，相等标志。

（2）输出信号：

控制总线：控制信号汇总而成的控制总线；

当前状态值：状态字的值。

（3）设计原理：略。

（4）设计思路：只需要选择正确的组件并相连即可。时序发生器要能够输出现态并保存次态，因此我们利用寄存器完成这样的功能。寄存器的输出置为状态机和输出函数的输入，状态机的输出返回寄存器进行保存。最后的电路**错误!未找到引用源。**如下：

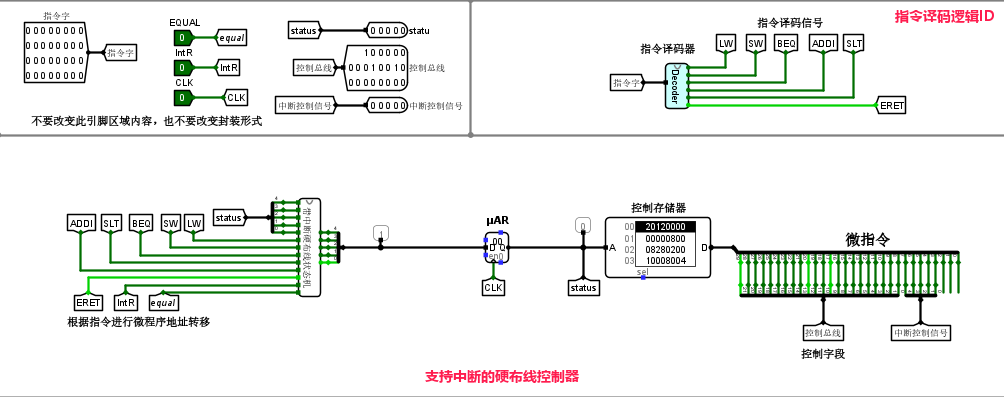


图 2.2.6 中断硬布线控制器实现图

## 实验步骤

1. 学习实验，阅读文档、观看视频，初步了解实验的目的以及过程；
2. 根据educoder的试验任务，从简到难逐个完成各个部件的设计，通过平台的测验；
3. 将各个部件组装为总控制器。加载排序程序sort-5-int-riscv.hex，运行CPU检查有无异常；
4. 将内容传至educoder对最终实验进行测试。

## 故障与调试

### 支持中断的微程序条件判别测试逻辑出错

**故障现象：**educoder评测时SLT指令报错。

**原因分析：**显然微程序条件判别测试逻辑整个设计，只有在填表的过程中会出现复杂的过程可能有错误。

**解决方案：**对excel表中涉及SLT指令的输入与输出检查，很快发现有一个S0的逻辑错误，修改后再传至educoder进行检查，通过。

### 支持中断的微程序单总线CPU设计出错

**故障现象：**logisim上测试时发现CPU触发中断时有误。

**原因分析：**因为之前组件的测试均通过了，那么触发有误应该是连线的问题，经过筛查，发现是选择器输出有误。

**解决方案：**对按照逻辑更改连线，添加了三态门组件，最终经过反复调试得到需要效果。传至educoder进行检查，通过。

## 测试与分析

将数据镜像文件sort-5-int-riscv.hex载入RAM，此时存储内容如下：

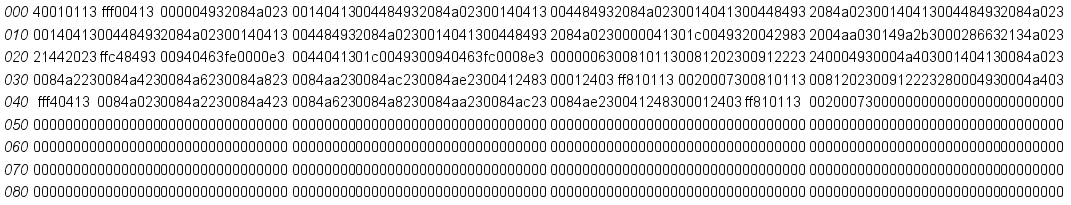


图 2.5.1 CPU运行前RAM存储

使用Ctrl+k自动测试，直到CPU将命令运行结束进入死循环。此时应执行到beq指令，指令数应为252，图如下：

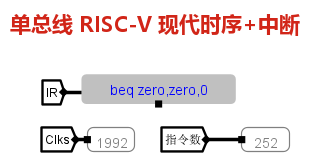


图 2.5.2 CPU运行后IR和指令数

执行完指令后RAM中存储部分为如下：

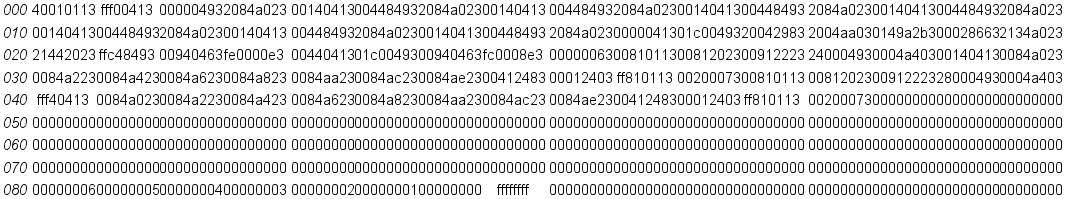


图 2.5.3 CPU运行后RAM存储

执行一次1号中断信号之后，指令数如下：

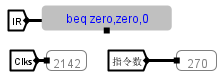


图 2.5.4 一次1号中断号后IR和指令数

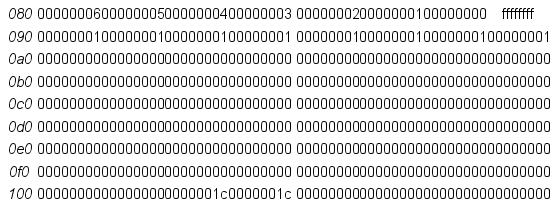


图 2.5.5 执行一次1号中断后的RAM存储

# 总结与心得

## 实验总结

本次CPU设计实验主要工作如下：

1. 完成了RISC-V指令译码器设计、时序发生器设计、时序发生器输出函数设计、硬布线控制器组合逻辑单元设计；
2. 通过连线实现了硬布线控制器设计的逻辑；
3. 联调实现变长指令周期3级时序单总线CPU设计；
4. 实现了支持中断的微程序入口查找逻辑、支持中断的微程序条件判别测试逻辑；
5. 通过连线实现了支持中断的微程序控制器设计；
6. 联调实现支持中断的微程序单总线CPU设计；
7. 实现了支持中断的现代时序硬布线控制器状态机设计，并逻辑实现支持中断的现代时序硬布线控制器设计，最终使得CPU能够顺利运行简单排序程序。

## 实验心得

这次实验，在给出的基础框架上进行CPU的功能实现，通过从简到难的逐步实现，对CPU的整个工作过程、控制器的不同部件以及其功能实现、三级时序和现代时序的异同、中断的实现流程、计数法的实现方法的理解更加地深入。

实验过程中有很多问题、很多最初不理解的地方，但是随着逐步地排错和资料查询，结合书本的知识，问题一个一个解决，我对整个CPU也更加理解。

总体来说实验的难度虽然较高，但编排的很好，能够让同学们逐步理解每个部件的实现逻辑，与课程内容相辅相成，也锻炼了学生的动手能力。

# 参考文献

1. DAVID A.PATTERSON(美).计算机组成与设计硬件/软件接口(原书第5版).北京:机械工业出版社.
2. David Money Harris(美).数字设计和计算机体系结构（第二版）. 机械工业出版社
3. 谭志虎,秦磊华,吴非，肖亮.计算机组成原理.北京:人民邮电出版社，2021年.
4. 谭志虎,秦磊华,胡迪青.计算机组成原理实践教程.北京:清华大学出版社，2018年.

|  |
| --- |
| 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:** [作者] |
| 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:** |