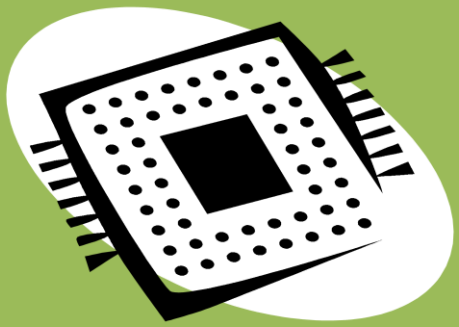
华中科技大学

2021

计算机组成原理 ·实验报告·

专　　业：	计算机科学与技术
班　　级：	CS1903
学　　号：	U201914995
姓　　名：	罗虞阳
电　　话：	13713561390
邮　　件：	1375081937@qq.com
完成日期：	2021-12-13



计算机科学与技术学院

目　录

1 CPU 设计实验_变长指令周期三级时序

1.1 设计要求

理解变长指令周期三级时序系统的设计，能利用该时序构造硬布线控制器，支持 5 条典型 RISC-V 指令在单总线 CPU 上运行，最终 CPU 能运行内存冒泡排序。使用指令集 RISC-V，通过对三级时序硬布线控制器的设计实现，最终组合成具有完整功能的单总线结构 CPU，如图 1.1 所示。

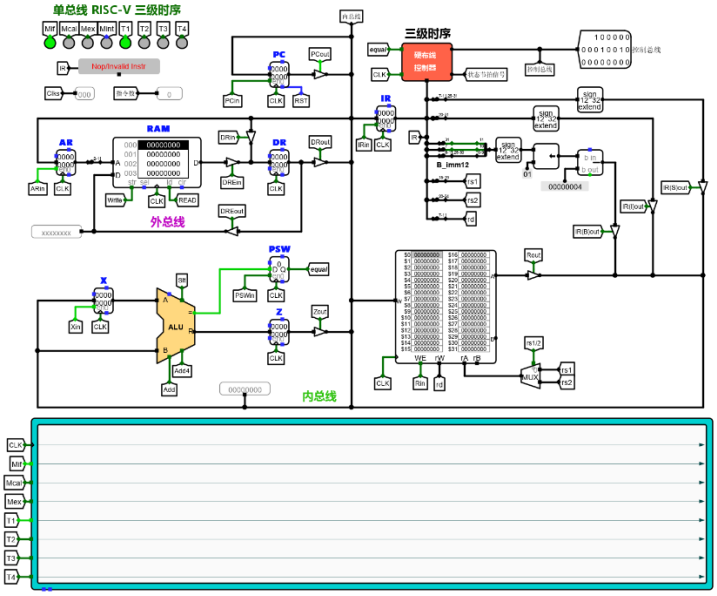


图 1.1 三级时序单总线 CPU

在三级时序硬布线控制器的设计实现，需要依次设计完成：

（1）RISC-V 指令译码器

（2）时序发生器

（3）时许发生器的输出函数

（4）硬布线控制器组合逻辑单元

（5）三级时序硬布线控制器

最终实现三级时序的硬布线控制器，如图 1.2 所示。

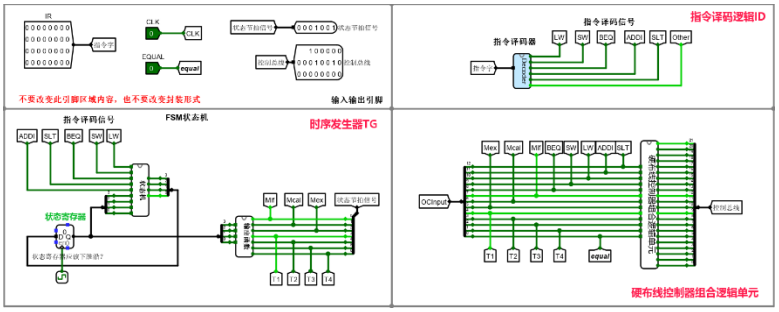


图 1.2 三级时序硬布线控制器

1.2 方案设计

1.2.1 RISC-V 指令译码器

指令译码器的电路实现如图 1.3 所示，输入为 32 位 RISC-V 指令集的指令，输出为 LW、SW、BEQ、ADDI、SLT 五个常用指令和判别出来的不属于其中指令的 OtherIInstr 的标识。

具体实现方式为通过将指令 IR 的 2-6 位 opcode 和 12-14 位 func3 提取出来进行对比，用 5 个比较器来判断该指令的 opcode 和 func3 是否与标准指令相匹配。

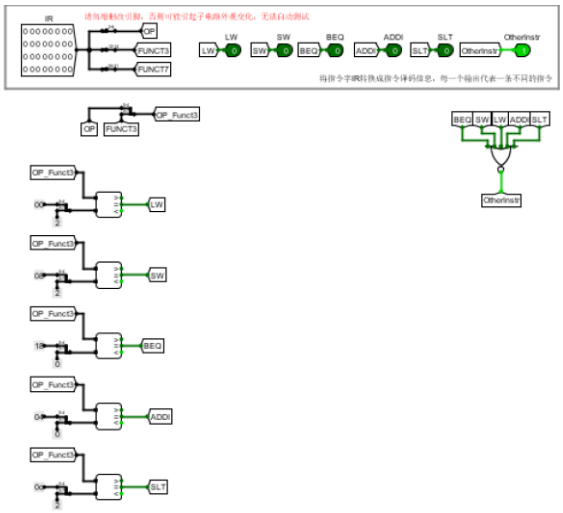


图 1.3 指令译码器

1.2.2 时序发生器

通过输入现态编码 S3~S0 和五个常用指令标识得到次态编码，具体实现逻辑为利用 Excel 表格，输入现态编号、次态编号和输入信号，得到组合逻辑，在 logism 组合

逻辑生成器生成电路。

具体状态图如图 1-4 所示。

当前状态(现态)					输入信号							下一状态（次态）				
S3	S2	S1	S0	现态10进制	LW	SW	BEQ	SLT	ADDI	ERET	IntR	次态10进制	N3	N2	N1	N0
0	0	0	0	0								1	0	0	0	1
0	0	0	1	1								2	0	0	1	0
0	0	1	0	2								3	0	0	1	1
0	0	1	1	3	1							4	0	1	0	0
0	0	1	1	3		1						4	0	1	0	0
0	0	1	1	3			1					4	0	1	0	0
0	0	1	1	3				1				6	0	1	1	0
0	0	1	1	3					1			6	0	1	1	0
0	1	0	0	4								5	0	1	0	1
0	1	0	1	5								6	0	1	1	0
0	1	1	0	6								7	0	1	1	1
0	1	1	1	7								8	1	0	0	0
1	0	0	0	8								0	0	0	0	0

图 1.4 三级时序的时序发生器

1.2.3 时序发生器的输出函数

该模块输入为时序发生器的现态编码 S3~S0，输出为相应状态的具体指令周期和时钟周期编码。

具体实现方法为在 Excel 表格中实现不同状态与输出的一一映射关系，如图 1.5 所示，将 Excel 得到的七个输出相应组合逻辑输入到 logism 中自动生成电路。

当前状态(现态)					输出							
S3	S2	S1	S0	现态10进制	Mif	Mcal	Mex	Mint	T1	T2	T3	T4
0	0	0	0	0	1				1			
0	0	0	1	1	1					1		
0	0	1	0	2	1						1	
0	0	1	1	3	1							1
0	1	0	0	4		1			1			
0	1	0	1	5		1				1		
0	1	1	0	6			1		1			
0	1	1	1	7			1			1		
1	0	0	0	8			1				1	

图 1.5 三级时序时序发生器的输出函数

1.2.4 硬布线控制器组合逻辑单元

该模块输入为指令周期、时钟周期以及相应指令信号，输出为具体指令在不同时钟周期下的控制信号。

具体实现方法为作出每条指令在每个机器周期中每个时钟周期的数据通路，继而根据数据通路判断出相应的控制信号并填入表中，利用 Excel 生成组合逻辑表达式，将组合逻辑表达式输入到 logisim 自动生成电路。具体的输入和输出如图 1.6 所示。

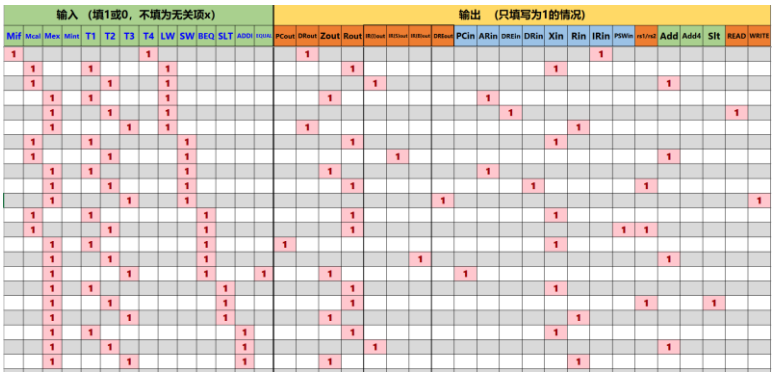


图 1.6 三级时序控制命令的产生

1.2.5 三级时序硬布线控制器

该模块实现方法为在 logisim 中实现硬布线控制器的连线任务，硬布线控制器如图 1.2 所示。整个控制器包括 FSM 状态机和控制信号逻辑生成这两个部分。FSM 状态机部分需要使用 1.2.1-1.2.3 中的电路封装起来，得到输出相应时序的指示。然后将该指示再与指令结合输入 1.2.4 的控制信号生成相应时钟周期的控制信号。

1.2.1~1.2.5 完成后，将其与 CPU 中其他组件相组合，即可实现三级时序的单总线 CPU。前面已经展示过，这里就不在赘述。

1.3 实验步骤

（1） 在 Excel 中得到组合逻辑表达式

（2） 将组合逻辑表达式输入 logism 中自动生成电路；其他部分自己实现硬件连接

（3） 在 educoder 平台提交实验电路.circ 文件

1.4 故障与调试

1.4.1 上升沿下降沿问题

故障现象：在程序逻辑都正确的情况下输出出错误的结果。

原因分析：使能端使用的时上升沿触发方式，但是上升沿会使得在电路接通的瞬间

状态机寄存器就会发生变化，而下降沿触发则是在断开之前一直保持电路接通状态不变。

解决方案：将该部分电路改为时钟下降沿有效，修改后将该电路.circ 文件输入educoder，结果正确通过。

1.4.2 拓展段译码设计问题

故障现象：SLT 指令在不满足出现条件的情况下产生，并且覆盖了超出范围的情况。

原因分析：错误地简化了 SLT 指令的构成条件，没有注意到它与其他指令之间的区别，SLT 指令相较于其他指令是一个扩展指令，不仅需要 OP 字段，还需要 FUNCT 字段参与构成，如下图所示。

解决方案：将 FUNCT 参与对 SLT 指令的构成，形成扩展指令，如图 1.7。

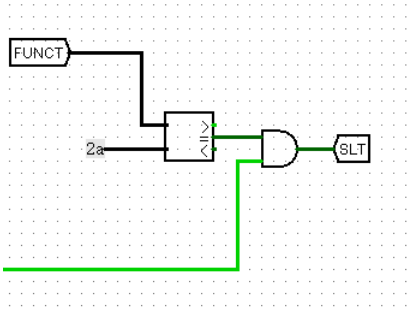


图 1.7 拓展段译码设计电路图

1.5 测试与分析

Educoder 上的所有模块均已经通过测试。

在 logisim 中，执行冒泡排序后 RAM 中相应数据如下图 1.9 所示，结果正确。

080 00000006 00000005 00000004 00000002 00000003 00000001 00000000 ffffffff

图 1.9 CPU 操作后降序排序结果

2 CPU 设计实验_带中断机制的现代时序

2.1 设计要求

本实训项目帮助学生理解现代时序控制器中断机制的实现原理，能为采用现代时序单总线结构的 RISC-V CPU 增加中断处理机制，可实现多个外部按键中断事件的随机处理，本实验需要完成现代时序微程序控制器的基础上完成，需要增加硬件数据通路，增加中断返回指令 meret 的支持，需要中断服务程序配合。

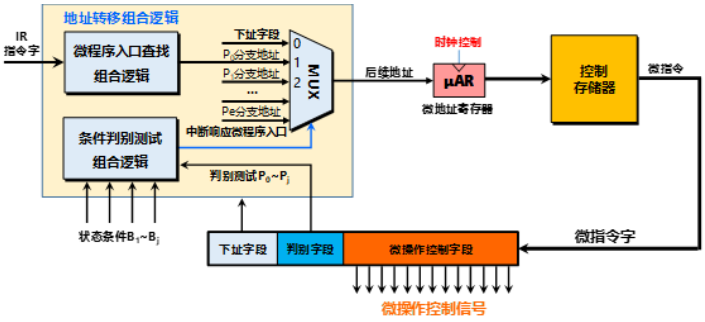


图 2.1 电路框架图

上图 2.1 通路在电路框架中已经给出，同学们主要任务是设计中断控制器内部内部逻辑，并最终调试运行标准测试程序 sort-5-int-riscv.hex。

在现代时序微程序控制器的设计实现过程中，需要依次设计完成：

（1）RISC-V 指令译码器

（2）支持中断的微程序入口查找逻辑

（3）支持中断的微程序条件判别测试逻辑

（4）支持中断的微程序控制器

（5）CPU 中的中断控制部分

最终实现使用计数器法的支持中断的微程序控制器。

2.2 方案设计

2.2.1 RISC-V 指令译码器

RISC-V 指令译码器已在 1.2.1 中做了详细的说明，这里就不再做赘述。

2.2.2 支持中断的微程序入口查找逻辑

该模块输入为六个指令选择信号，输出为微程序的入口地址。具体实现逻辑为使用老师给的 Excel 表构造组合逻辑，在 logism 中使用组合逻辑生成器生成电路。

Excel 表中内容如图 2.2，生成电路如图 2.3。

机器指令译码信号						微程序入口地址					
LW	SW	BEQ	SLT	ADDI	ERET	入口地址10进制	S4	S3	S2	S1	S0
1						4	0	0	1	0	0
	1					9	0	1	0	0	1
		1				14	0	1	1	1	0
			1			19	1	0	0	1	1
				1		22	1	0	1	1	0
					1	25	1	1	0	0	1

图 2.2 Excel 微程序入口查找逻辑内容

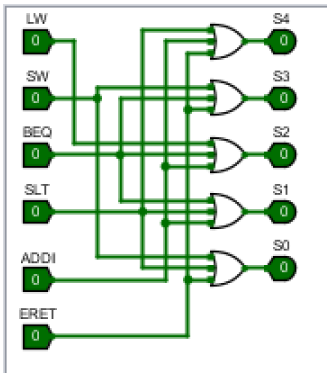


图 2.3 logism 微程序入口查找电路图

2.2.3 支持中断的微程序条件判别逻辑测试

输入为 P 字段流程控制信号和表示相等的 equal 信号，输出为微程序转移地址，具体实现逻辑为使用老师给的 Excel 表构造组合逻辑，如图 2.4 所示，在 logism 中使用组合逻辑生成器自动生成电路。

P0	P1	P2	equal	IntR	S2	S1	S0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	1
1	0	0	1	1	0	0	1
0	1	0	1	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	1	0	1	0
0	1	1	0	1	0	1	1
0	1	1	0	0	1	0	0
0	0	1	0	1	0	1	1
0	0	1	1	1	0	1	1
0	0	1	0	0	1	0	0
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	0
0	1	0	0	1	1	0	0

图 2.4 支持中断的微程序条件判别逻辑表

2.2.4 硬布线时序产生器

硬布线时序产生器输入为当前状态，结合输入信号获得输出的次态，具体实现逻辑为使用老师提供的 Excel 表构造组合逻辑，然后在 logisim 中使用组合逻辑生成器自动生成电路。Excel 表内容如图 2.5 所示。

S4	S3	S2	S1	S0	现态10进制	LW	SW	BEQ	SLT	ADD1	ERET	EQUAL	次态10进制	N4	N3	N2	N1	N0
0	0	0	0	0	0								1	0	0	0	0	1
0	0	0	0	1	1								2	0	0	0	1	0
0	0	0	1	0	2								3	0	0	0	1	1
0	0	0	1	1	3	1							4	0	0	1	0	0
0	0	0	1	1	3		1						9	0	1	0	0	1
0	0	0	1	1	3			1					14	0	1	1	1	0
0	0	0	1	1	3				1				19	1	0	0	1	1
0	0	0	1	1	3					1			22	1	0	1	1	0
0	0	0	1	1	3						1		25	1	1	0	0	1
0	0	1	0	0	4								5	0	0	1	0	1
0	0	1	0	1	5								6	0	0	1	1	0
0	0	1	1	0	6								7	0	0	1	1	1
0	0	1	1	1	7								8	0	1	0	0	0

图 2.5 硬布线时序产生器组合逻辑表

2.2.5 支持中断的微程序控制器

该模块输入为微程序地址，输出为 30 位微程序，具体实现逻辑为对照硬布线的控制信号输出表填写控制信号，然后再根据 FSM 中状态之间的跳转关系设计流程控制信号 P。微程序控制寄存器内容如图 2.5 所示。

微指令功能		PCout	DRout	Zout	Rout	R0out	R1out	R2out	R3out	PCin	ARin	DRin	DRin	Xin	Rin	IRin	PSWin	rs1/2	Add	Add4	Slt	READ	WRITE	EPCout	EPCin	Address	STI	CLI	P0	P1	P2	微指令
取指令	0	1									1			1																		10000000010010000000000000000
取指令	1																			1												00000000000000000010000000000
取指令	2			1						1		1										1										00100000101000000000100000000
取指令	3		1												1															1		01000000000001000000000000010
LW	4				1									1																		00010000000100000000000000000
LW	5					1													1													00001000000000001000000000000
LW	6			1							1																					00100000010000000000000000000
LW	7											1										1										00000000001000000001000000000
LW	8		1												1															1		10000000000001000000000000000
SW	9				1									1																		00010000000100000000000000000
SW	10					1													1													00001000000000001000000000000
SW	11			1							1																					00100000010000000000000000000
SW	12				1								1						1													00010000001000010000000000000
SW	13							1														1									1	00000010000000000001000000000
BEQ	14				1									1																		00010000000100000000000000000
BEQ	15				1											1	1													1	1	00100000000000110000000000001
BEQ	16	1												1																		10000000000010000000000000000
BEQ	17						1												1													00000010000000001000000000000
BEQ	18			1						1																					1	01000001000000000000000000000
SLT	19				1									1																		00010000000100000000000000000
SLT	20				1													1			1											00010000000000001001000000000
SLT	21			1											1																1	01000000000010000000000000000
ADDI	22				1									1																		00010000000010000000000000000
ADDI	23					1													1													00001000000000001000000000000
ADDI	24			1											1																1	01000000000010000000000000000
ERET	25									1															1		1				1	00000001000000000001001000
中断响应	26	1																								1		1				10000000000000000000000100100
中断响应	27									1																	1				1	00000001000000000000000010000

图 2.5 微程序控制寄存器内容

当表格完成后，将微指令使用十六进制的方式存入微程序控制器的控制存储器中，这样就可以按照正确的逻辑通过多路选择器选取下一条微程序的入口地址。如图 2.6 所示。

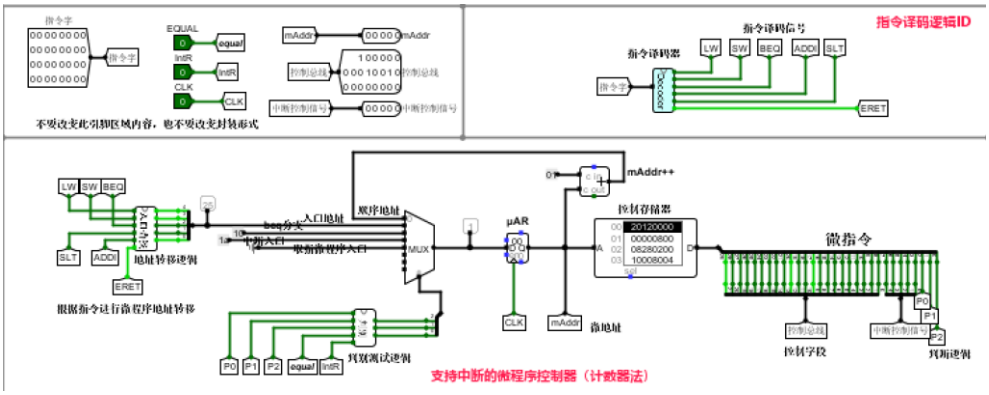


图 2.6 微程序控制器总体结构图

2.2.6 CPU 中的中断控制部分

这部分为 CPU 中的终端逻辑，具体实现方法为在 logisim 中直接连接中断逻辑的各个组成部件。连接方式如图 2.7 所示。

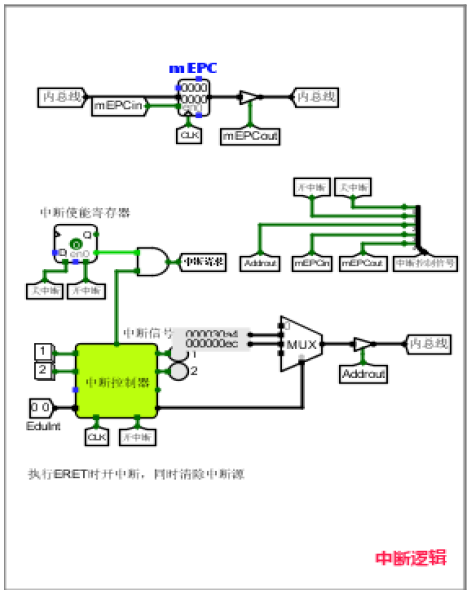


图 2.7 中断逻辑连接图

如图 2.7 所示，中断逻辑可分为三个部分，分别是中断控制器、mEPC 寄存器和中断地址查找。

中断控制器使能信号为是否开关中断。若有中断申请且当前正处于开中断状态，则根据中断优先级判断逻辑输出中断请求和相应的中断入口地址选择逻辑；若当前正处于关中断状态，则不会有任何的输出。

mEPC 寄存器在写入信号为 0 时，不会把输入端内容保存下来，锁存器输出连接三态门，只有在 EPC 输出信号有效的情况下才能进行输出，不能输出时三态门为阻挡状态。

中断地址查找则是根据中断控制器输出的中断类型使用多路选择器进行分支判断。若输出类型为 0 和 3 的就不予处理；若为分支 1 或 2，则接上代表 1 或 2 号中断的中断入口地址常量。

最后将 2.2.1-2.2.6 分别完成之后，再将各个已经实现好的模块与 CPU 中的其他组件相结合，即可实现带中断的单总线 CPU。

2.3 实验步骤

（1）按要求填写表格并在 logisim 实现硬件连接。

（2）在 CPU 设计中载入相应汇编程序的十六进制代码运行冒泡程序。

（3）在 educoder 平台提交通过实验。

2.4 故障与调试

2.4.1 故障 1：带中断 CPU 无法触发中断

错误原因：IE 寄存器连线错误。原先是连接到了当前触发器的状态值，但是应该是连接当前触发器状态取非后的值。

解决方案：连接当前触发器状态的非值。

2.4.2 故障 2：支持中断微程序逻辑自动生成报错

错误原因：控制信号输入错误，导致数据通路出现问题。

解决方案：仔细看书，挨个对照检查，确保控制信号和通路正确。

2.5 测试与分析

Educoder 上已通关所有关卡。在 logisim 中，RAM 执行冒泡排序后存储内容如图 2.8 所示

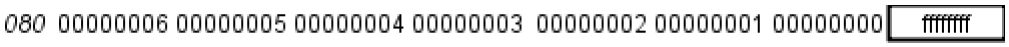


图 2.8 带中断的冒泡程序结果

若按下按键 1，程序将在 90 开始的 8 个字单元全部加 1，如图 2.9 所示：

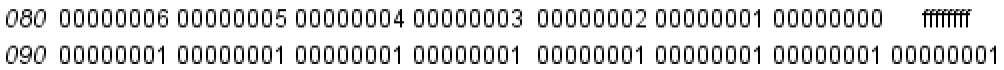


图 2.9 第一次按下 1 号中断结果

再次按下按键 1，执行结果如图 2.10 所示：

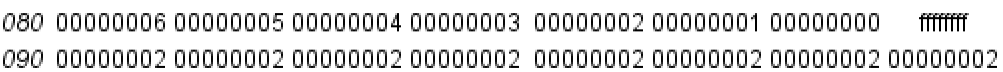


图 2.10 第二次按下 1 号中断结果

若按下按键 2，程序将在 a0 开始的 8 个字单元全部减 1，如图 2.11 所示：

```
080 00000006 00000005 00000004 00000003  00000002 00000001 00000000    ffffffff
090 00000002 00000002 00000002 00000002  00000002 00000002 00000002 00000002
0a0    ffffffff       ffffffff       ffffffff       ffffffff       ffffffff       ffffffff       ffffffff       ffffffff
```

图 2.11 第一次按下 2 号中断结果

再次按下按键 2，执行结果如图 2.12 所示：

```
080 00000006 00000005 00000004 00000003  00000002 00000001 00000000    ffffffff
090 00000002 00000002 00000002 00000002  00000002 00000002 00000002 00000002
0a0    fffffffe       fffffffe       fffffffe       fffffffe       fffffffe       fffffffe       fffffffe       fffffffe
```

图 2.12 第二次按下 2 号中断结果

综上可见，冒泡排序正确且各中断程序输出正确，实验成功。

3 总结与心得

3.1 实验总结

本次实验主要完成了如下几点工作：

（1） 完成基于 RISC-V 指令集的现代时序单总线 CPU 设计、变长指令周期三级时序 CPU 设计和现代时序中断机制的实现。

（2） 提交设计电路 circ 文件到 educoder 并通过全部测试。

（3） 测试冒泡排序并取得正确结果。

3.2 实验心得

（1） 编写 CPU 需要优质的工具，在组合逻辑部分可以使用 Excel 表格获得逻辑表达式，再由 logisim 的电路生成功能直接获得组合逻辑电路图，简化了电路连线操作。

（2） 大型的复杂电路设计往往需要拆分为较为简单的基础原件设计，再将调试好的寄出原件自底向上组装形成复杂电路。

（3） 需要熟练掌握 RISC-V 指令集，实验对使用 RISC-V 核心指令的结构和条件有着很高的要求，通过这次实验加深了我对该指令集的了解。

（4） 关于建议方面，我认为可以适当屏蔽测试集，或者只提供部分测试案例以方便同学进行 debug，减少面向测试集编写 excel 表的情况。同时部分实验测试集针对边界值情况还有待完善，存在通过前面关卡但是拼接后不能通过测试的情况。

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.

• 指导教师评定意见 •

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：罗虞阳 

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： _____