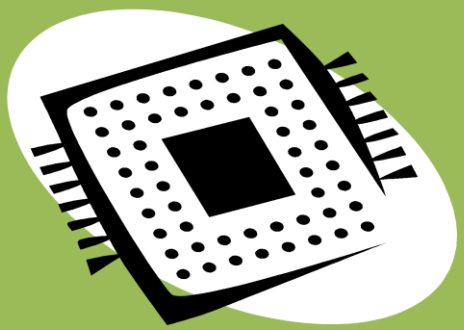
华中科技大学

2022

计算机组成原理 ·实验报告·

专　　业：	计算机科学与技术
班　　级：	计卓 2001 班
学　　号：	U201914858
姓　　名：	王怡彬
电　　话：	17389560072
邮　　件：	wyb896409234@gmail.com
完成日期：	2022-07-06



计算机科学与技术学院

1 CPU 设计实验

1.1 设计要求

利用 logisim 平台中的现有部件构建一个支持中断的 MIPS 现代时序单总线 CPU，为采用现代时序单总线结构的 MIPS CPU 增加中断处理机制，分别为采用微程序控制器和硬布线控制器的 CPU 实现，可实现多个外部按键中断事件的随机处理，需要增加硬件数据通路，增加中断返回指令 eret 的支持，需要中断服务程序配合。微程序控制器和硬布线控制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的控制器。

表 1-1 微程序控制器片引脚与功能描述

引脚	输入/输出	位宽	功能描述
指令字	输入	32	输入机器指令
EQUAL	输入	1	beq 指令时的比较结果（是否相等）
IntR	输入	1	输入中断请求
CLK	输入	1	时钟脉冲信号
mAddr	输出	5	微地址
IntSignals	输出	5	中断控制信号
ControlBus	输出	22	微指令控制字段

表 1-2 硬布线控制器片引脚与功能描述

引脚	输入/输出	位宽	功能描述
指令字	输入	32	输入机器指令
EQUAL	输入	1	beq 指令时的比较结果（是否相等）
IntR	输入	1	输入中断请求
CLK	输入	1	时钟脉冲信号
statu	输出	5	当前状态
IntSignals	输出	5	中断控制信号

ControlBus	输出	22	微指令控制字段

1.2 方案设计

1.2.1 MIPS 指令译码器设计

利用比较器等功能模块将 32 位 MIPS 指令字译码生成 LW、SW、BEQ、SLT、ADDI、OtherInstr 信号。将输入指令字拆分为 OP 与 FUNCT，参考 MIPS 英文指令手册查询相关指令的规格，对 OP 和 FUNCT 与对应的各个常量进行逻辑比较判断，即可生成目标信号。具体电路图如下。

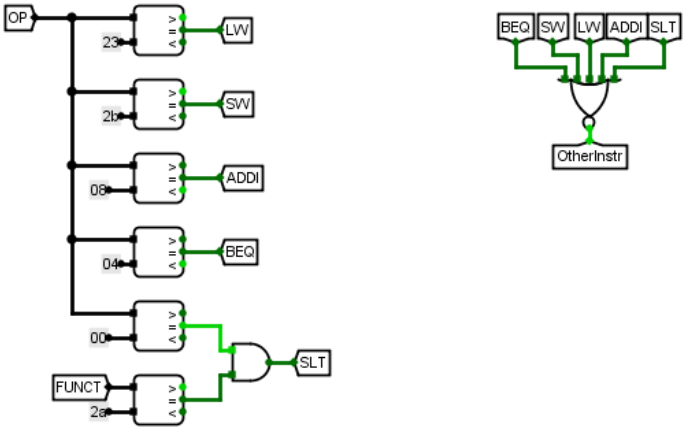


图 1-1 MIPS 指令译码器电路

1.2.2 支持中断的微程序入口查找逻辑

根据地址转移逻辑状态机图（如图 1-2 所示），每个状态对应一个时钟周期，将状态编号转换为微指令地址，即使用纯组合逻辑电路，将指令译码器输出的信号直接转换为对应的微程序入口地址。

通过写出微指令入口地址表（如图 1-3 所示），使用实验给出的自动逻辑表达式生成工具，生成对应的 S1-S4 的逻辑表达式，将其输入进 Logisim，通过逻辑表达式自动生成组合逻辑电路即可，具体电路图如图 1-4 所示。

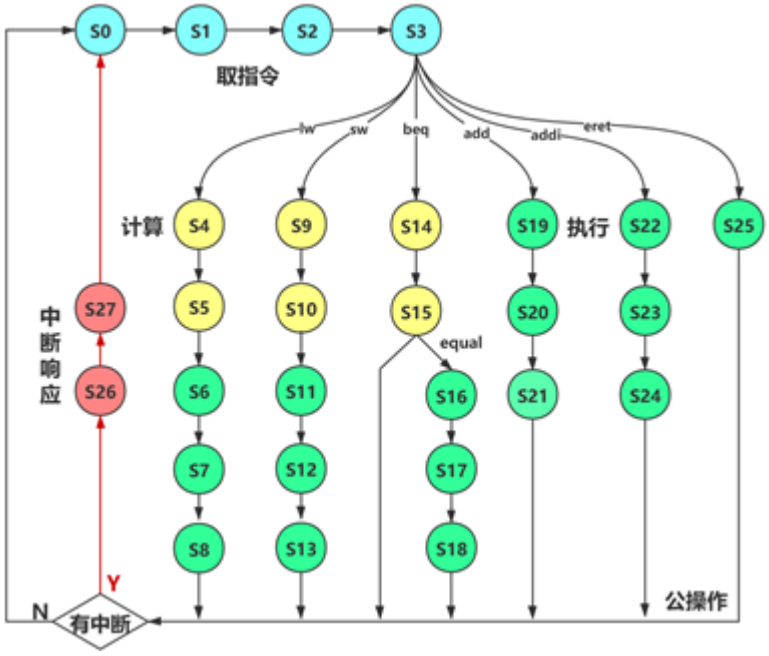


图 1-2 地址转移逻辑状态机图

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 1-3 微程序入口地址表

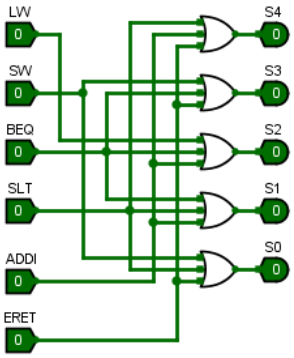


图 1-4 支持中断的微程序入口查找逻辑电路

1.2.3 支持中断的微程序条件判别测试逻辑

实验采用计数器法地址转移逻辑，通过 μAR+1 得到顺序地址并送入地址转移逻辑得到下一条微指令的地址。判别字段中的 P_0 判别测试位，为 1 表示要根据指令功能进行微程序分支。P_1 判别测试位，为 1 表示要根据 equal 标志进行微程序分支。

为支持中断实现，增加了判别测试位 P_2，P_2 为 1 表示当前微指令为微程序最后一条微指令，需要进行中断判断，如果存在中断请求（IntR），需要转中断响应微程序执行。

根据微指令字中的判别测试字段和条件反馈信息，按照判别测试逻辑，写出条件判别逻辑真值表（如图 1-4 所示），利用实验所给工具，自动生成输出信号逻辑表达式，再使用 Logisim，利用其自动生成组合逻辑电路，生成后续地址的多路选择信号，搭配多路选择器即可实现地址转移，具体电路图如图 1-5 所示。

输入 (填1或0，不填为无关项x)					输出 (只填写为1的情况)							
P0	P1	P2	equal	IntR	S2	S1	S0	Out4	Out5	Out6	Out7	
0	0	0			0	0	0					
1		0			0	0	1					
0	1	0	0		1	0	0					
0	1		1		0	1	0					
	0	1		1	0	1	1					
		1	0	1	0	1	1					
	0	1		0	1	0	0					
0	1	1	0	0	1	0	0					

图 1-4 条件判别逻辑真值表

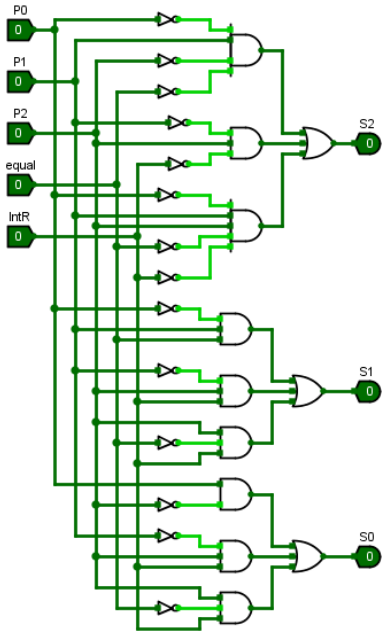


图 1-5 条件判别测试组合逻辑电路

1.2.4 支持中断的微程序控制器设计

为响应中断，需要新增中断响应周期，对应增加两条微指令实现中断响应操作，即中断响应微程序；为返回中断，在原始的微指令基础上增加一条 eret 指令，用于回归原来被中断的程序继续执行。

如图 1-6 所示，分别实现取指微程序、lw 指令微程序、sw 指令微程序、beq 指令微程序、slt 指令微程序、addi 指令微程序、eret 指令微程序以及中断响应微程序的对应微指令的设计，将其加载入控制存储器。并在前述 MIPS 指令译码器、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基础上将其适当连接，即可实现支持中断的微程序控制器设计，具体电路图如图 1-7 所示。

微指令功能		PCout	DRout	Zout	Rout	RAMout	RAMwe	DRSout	PCin	ARin	DRSin	DRin	Xin	Rin	IRin	PSWin	Rs/Rt	Rs_Rt	Add	Add4	Slt	READ	WRITE	EPCout	EPCin	RAMread	STI	CLI	P1	P2	P3	微指令	微指令十六进制
取指令	0	1								1			1																			10000000100100000000000000000000	20240000
取指令	1																			1												00000000000000000001000000000000	800
取指令	2			1					1		1											1										00100001010000000000100000000000	8500200
取指令	3		1												1															1		01000000000001000000000000000100	10010004
lw	4				1								1																			00010000000100000000000000000000	4040000
lw	5					1														1												00001000000000001000000000000000	2001000
lw	6			1						1																						00100000100000000000000000000000	8200000
lw	7										1											1										00000000001000000000001000000000	100200
lw	8		1											1																	1	01000000000010000000000000000001	10020001
sw	9				1								1																			00010000001000000000000000000000	4040000
sw	10					1													1													00001000000000001000000000000000	2001000
sw	11			1						1																						00100000100000000000000000000000	8200000
sw	12				1						1						1															00010000001000001000000000000000	4084000
sw	13						1																1								1	00000010000000000000001000000001	800101
beq	14				1								1																			00010000001000000000000000000000	4040000
beq	15				1											1	1														1	1	00010000000001100000000000000011	400C003
beq	16	1											1																			10000000000010000000000000000000	20040000
beq	17						1												1													00000100000000001000000000000000	1001000
beq	18			1					1																						1	01000001000000000000000000000001	8400001
slt	19				1								1																			00010000001000000000000000000000	4040000
slt	20				1												1				1											00010000000000001000100000000000	4004400
slt	21			1										1				1													1	01000000000001000100000000000001	8022001
addi	22				1								1																			00010000001000000000000000000000	4040000
addi	23					1													1													00001000000000001000000000000000	2001000
addi	24			1										1																	1	01000000000010000000000000000001	8020001
eret	25								1																		1		1		1	00000001000000000000000100010001	400091
	26	1																								1		1				10000000000000000000000001001000	20000048
	27								1																		1				1	00000001000000000000000000100001	400021

图 1-6 微程序自动生成表

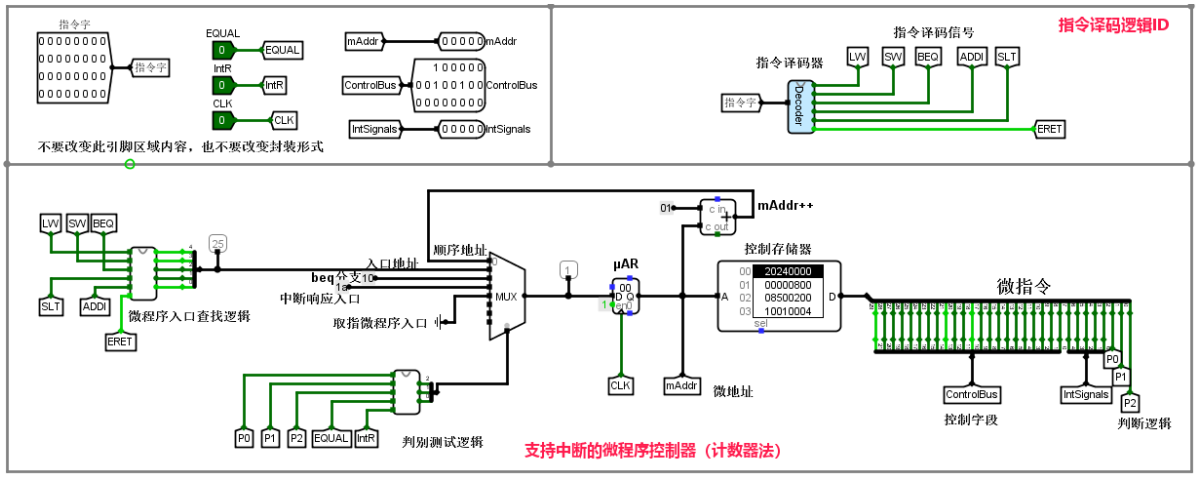


图 1-7 微程序控制器电路图

1.2.5 支持中断的微程序单总线 CPU 设计

在实现上述模块后，为使得微程序单总线 CPU 支持中断，需为其增加中断逻辑，在单总线数据通路中增加与中断相关的硬件模块，主要包括异常程序地址计数器 EPC，中断使能寄存器 IE，中断控制器等模块。在主电路中将这些模块进行有效连接，即可完成支持中断的微程序单总线 CPU 设计，中断逻辑具体电路图如下。

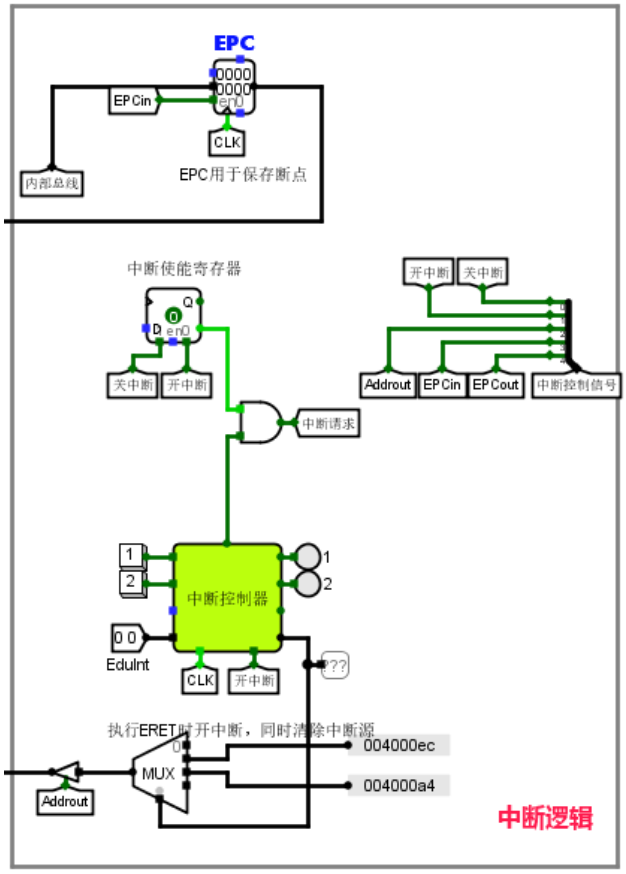


图 1-8 支持中断的微程序单总线 CPU 中断逻辑图

1.2.6 支持中断的现代时序硬布线控制器状态机设计

根据前述现代时序控制器状态图（如图 1-2 所示），每个状态对应一个时钟周期。根据状态图得到状态转换表，使用状态转换表生成次态逻辑表达式，使用次态逻辑表达式完成组合逻辑设计。由于硬布线控制器状态机组合逻辑较为复杂，电路图内容过多，此处图略。

1.2.7 支持中断的现代时序硬布线控制器设计

利用指令译码器和现代时序状态机模块，实现硬布线控制器的集成。这里直接采用微程序控制器的控制存储器代替硬布线控制器组合逻辑，将现态和微指令建立映射关系，即可实现输出现态决定的当前的微操作信号。具体电路图如下。

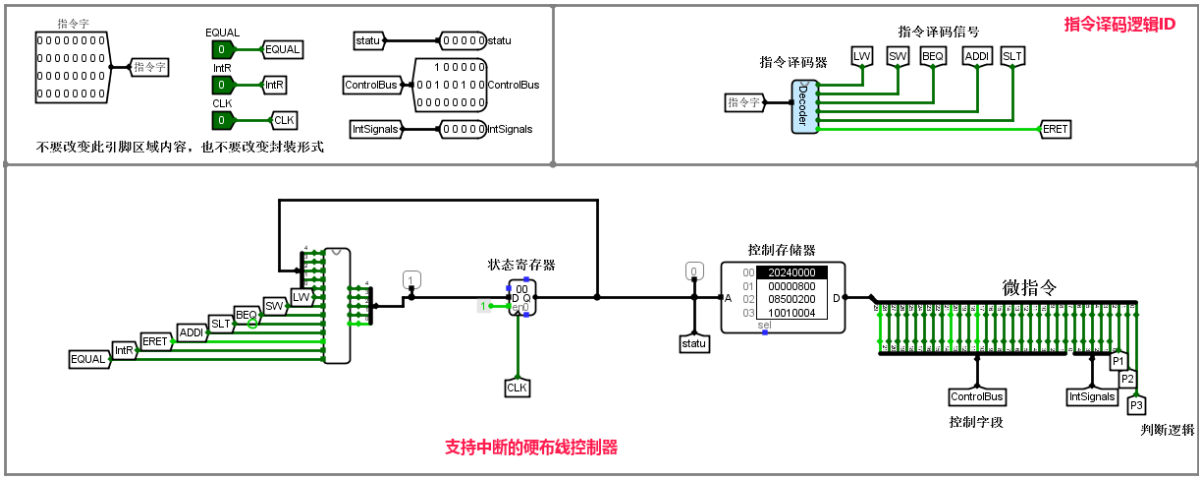


图 1-9 支持中断的硬布线控制器电路图

1.3 实验步骤

（1） 按照上述方案设计依序完成各个模块的设计与连接。

（2） 调试各个模块，验证其单一子功能是否正确，修正出现的各种故障。最后提交至 educoder 平台完成最终评测验证。

（3） 在支持中断的微程序单总线 CPU 中加载测试程序，完成中断程序功能的测试。

1.4 故障与调试

1.4.1 微指令重复输出的问题

故障现象：对设计完毕的 CPU 进行测试时，每条微指令都会额外重复输出一次。

原因分析：微程序控制器中的 μAR 寄存器和硬布线控制器中的状态寄存器都使用系统提供的唯一时钟信号来控制状态更新，并且都采用上升沿触发，但实验中 CPU 的

时序控制信号也是上升沿有效，导致在每个时钟周期的开始就控制寄存器更新状态，寄存器在上一条微指令刚结束后就直接更新到了下一条微指令，故导致了每条微指令都会额外重复输出一次的故障。

解决方案：将两种控制器中的状态寄存器都改为下降沿触发，即可避免寄存器状态更新时的冲突。

1.4.2 微程序条件判别测试逻辑出错问题

故障现象：判别逻辑在 P2、P3 均为 1，相等标记与中断请求信号同时出现的情况下会出现判别错误。

原因分析：在 P2、P3 均为 1，相等信号与中断请求信号同时出现时，编写的判别逻辑有误。

解决方案：重新修改判别测试逻辑，使其能够正确输出对应信号。

1.5 测试与分析

对 CPU 能否正常响应 2 个按键对应的中断服务程序进行测试。在 RAM 中加载冒泡排序程序 sort-5-int.hex，ctrl+k 自动运行，如没有按键行为，程序运行至 0x7c8 节拍停下，指令计数为 252。内存数据如下图所示，蓝色区域的数值为有符号降序排序后的数据，而红色区域数值为全 0。

```
000 23bd0400 2010ffff 20110000 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200
010 22100001 22310004 ae300200 22100001 22310004 ae300200 22100001 22310004 ae300200 20100000 2011001c 8e130200 8e340200 0274402a 11000002 ae330200
020 ae140200 2231fffc 12110001 1000fff7 22100004 2011001c 12110001 1000fff3 1000ffff 23bd0008 afb00000 afb10004 20310240 8e300000 22100001 ae300000
030 ae300004 ae300008 ae30000c ae300010 ae300014 ae300018 ae30001c 8fb10004 8fb00000 23bdfff8 42000018 23bd0008 afb00000 afb10004 20310280 8e300000
040 2210ffff ae300000 ae300004 ae300008 ae30000c ae300010 ae300014 ae300018 ae30001c 8fb10004 8fb00000 23bdfff8 42000018 00000000 00000000 00000000
050 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
070 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
080 00000006 00000005 00000004 00000003 00000002 00000001 00000000 ffffffff 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
090 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0a0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0b0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

图 1-10 无中断时内存数据

按下按键 1，触发 1 号中断，程序在 90 开始的 8 个字单元全部加 1，如下图所示

```
070 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
080 00000006 00000005 00000004 00000003 00000002 00000001 00000000 ffffffff
090 00000001 00000001 00000001 00000001 00000001 00000001 00000001 00000001
```

图 1-11 第一次 1 号按键

再次按下按键 1，触发 1 号中断，程序在 90 开始的 8 个字单元再次全部加 1，执行结果如下图所示

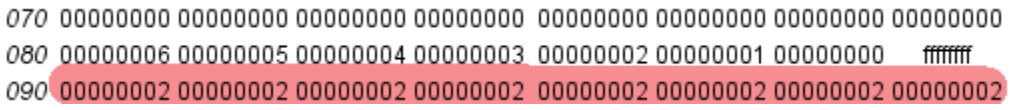


图 1-12 第二次 1 号按键

按下按键 2，触发 2 号中断，程序将在 a0 开始的 8 个字单元全部减 1，这里补码表示为全 1，如下图所示

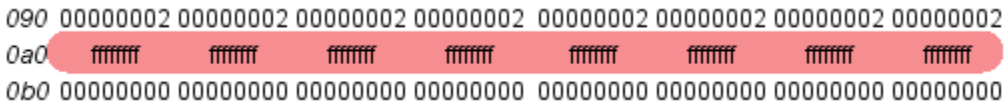


图 1-13 第一次 2 号按键

再次按下按键 2，触发 2 号中断，程序将在 a0 开始的 8 个字单元再次全部减 1，如下图所示

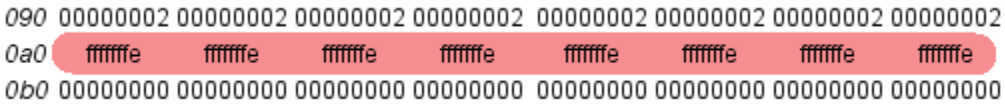


图 1-14 第二次 2 号按键

综上所述，CPU 能够正常响应 2 个按键对应的中断服务程序，并正确执行对应程序功能，测试成功。

1.6 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

设计实现了 MIPS 指令译码器、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、支持中断的微程序单总线 CPU 设计、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。

2） 功能总结

实现了微程序控制器和硬布线控制器的功能、并为其各自增加了中断功能，能够通过简单冒泡排序程序的运行和按键中断的测试。

3） 调试与测试

调试与修正了实验中出现的各种故障，进行功能测试，并最终完成了测试要求。

1.7 实验心得

1） 实验收获

在本次实验中，从最简单的指令译码器的设计开始，逐步完成了一个个组件的设计，最终亲手完成了支持中断的控制器设计，再将其与其他现成部件进行连接，最终实现了一个基本的支持中断的单总线 CPU 设计，并通过了功能测试。在这个过程中，我重新复习并巩固了理论课中学习到的理论知识，通过亲手实践进一步加深了对于中断实现逻辑与微程序控制器和硬布线控制器的认识，并且重新学习到了很多在理论学习中忽略的重点知识，从底层的角度重新认识了 CPU 控制器的工作原理与中断的运行逻辑。

2） 实验体会

本次实验的实验量看似虽然很大，但老师已经为我们提供了很多便捷的功能，例如通过状态转换表自动生成次态逻辑表达式等，减少了很多冗余的组合逻辑设计工作。同时，实验将整个 CPU 的设计拆解为一个个小组件的设计与连接，让我从小至大、由浅入深地逐步学习、理解、调试并最终完成了支持中断的单总线 CPU 设计，收获颇丰。非常感谢老师与课程组对整个实验的付出，也非常感谢同学与老师在我实验遇到困难时提供的各种帮助与支持。

3） 实验建议

个人建议可以在实验评测时隐藏测试样例答案，防止对照答案硬性修改电路逻辑。

• 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：王怡彬 王怡彬

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： _____