

华中科技大学

2023

计算机组成原理

· 实验报告 ·

专 业： 计算机科学与技术

班 级： CS2106

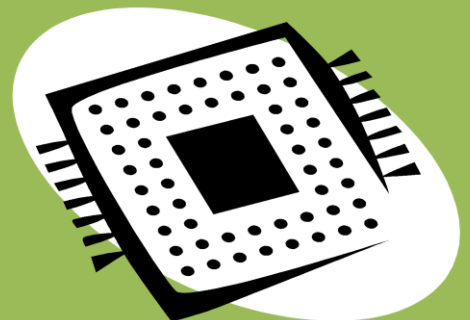
学 号： U202115514

姓 名： 杨明欣

电 话： 13390396012

邮 件： ymx@hust.edu.cn

完成日期： 2023-05-30



计算机科学与技术学院

1 CPU 设计实验

1.1 设计要求

本实验要求利用 logisim 平台，完成一个支持中断的单总线 CPU 设计，可支持指令译码，实现使用硬布线控制器和微程序控制器实现对译码指令相应的控制信号输出，同时可实现多个外部按键中断事件的随机处理。

本实验需要在现代时序微程序控制器的基础上完成对于中断服务程序的支持。需要增加硬件数据通路，升级控制器，增加中断返回指令 `eret` 的支持，需要中断服务程序配合，需要在微程序控制器、硬布线控制器和硬布线状态机中做部分修改。

总结一下，需要设计支持中断的单总线 CPU，使得其能够通过头歌平台上的七个关卡，同时在自己的平台上保证 MIPS 程序能正常运行，能够通过简单的冒泡排序 `sort-5-int.hex` 和中断逻辑操作，即为完成 CPU 设计实验。

最终需要实现得 MIPS 单总线 CPU 如图 1-1 所示。

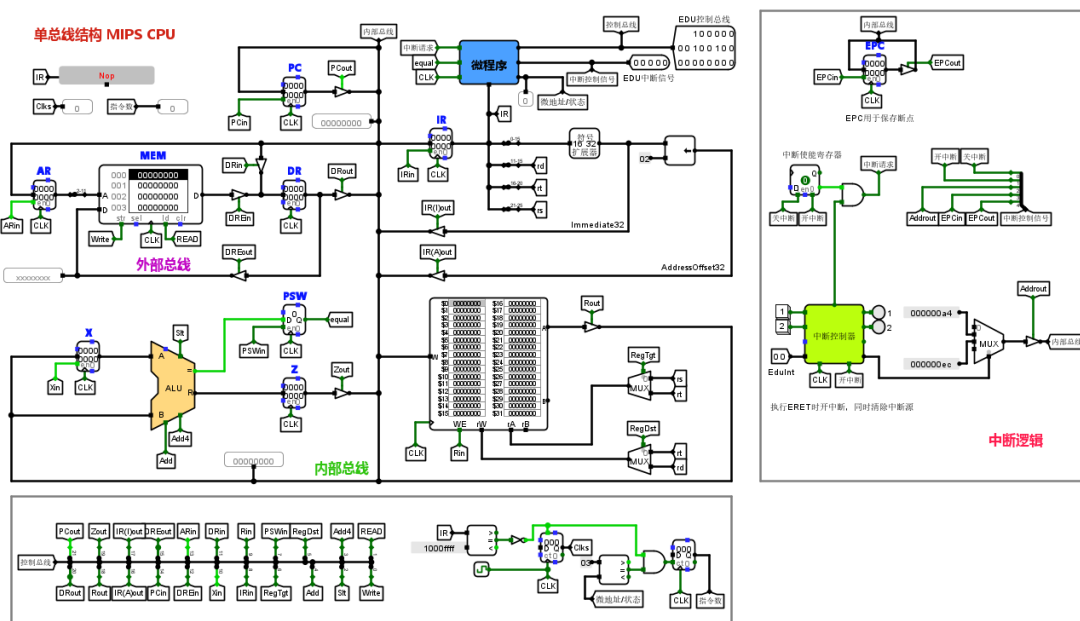


图 1-1 MIPS 单总线 CPU 设计概览

1.2 方案设计

1.2.1 MIPS 指令译码器

设计原理：通过 MIPS 指令手册中对应指令的指令字进行匹配判断，使得对应指令的输出为高电平。

设计思路：利用比较器模块和与门、或非门电路实现 32 位 MIPS 指令字生成 LW, SW, BEQ, SLT, ADDI 信号，如果 MIPS 指令无法匹配上述五个信号，则 OtherInstr 输出高电平。

设计过程：利用比较器，对应比较不同指令的指令字，根据比较器 equal 输出端口的值，使得匹配到信号输出高电平，其中 SLT 约束更为严格，需要两个比较器同时判断 OP 字段部分和 FUNCT 是否匹配，其他指令只需一个比较器匹配 OP，OtherInstr 则是在其他信号均为低电平时通过或非门逻辑输出高电平。电路图和封装如图 1-2 所示。

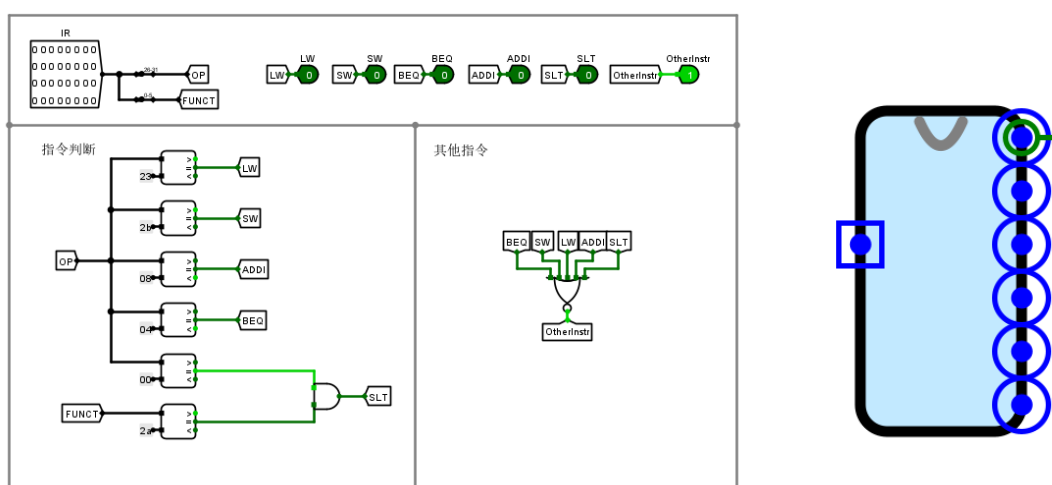


图 1-2 MIP 指令译码器设计及封装

1.2.2 支持中断的微程序入口查找逻辑

设计原理：指令译码信号和微程序入口地址对应，每个信号都对应着一个微程序入口地址，根据指令译码信号可以生成不同的微程序入口地址。

设计思路：根据输出的指令译码信号生成五位的微程序入口地址，地址转移逻辑如图 1-3 所示，通过地址转移逻辑判断出微程序的十进制入口地址，构造出微程序入口查找逻辑。

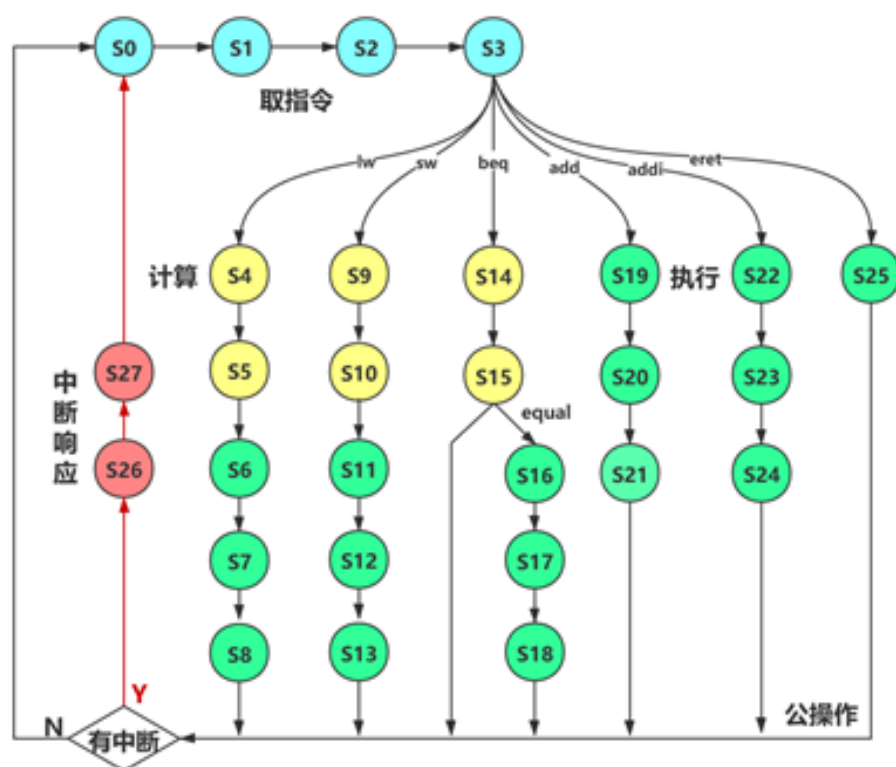


图 1-3 支持中断的现代时序状态机

设计过程：将入口地址的查找逻辑在 excel 中完成之后，excel 会自动输出口地址为二进制的信号和 S0~S4 生成表达式，具体 excel 中的微程序入口地址表格如图 1-4 所示。

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	0	0	0	0	0	4	0	0	1	0	0
0	1	0	0	0	0	9	0	1	0	0	1
0	0	1	0	0	0	14	0	1	1	1	0
0	0	0	1	0	0	19	1	0	0	1	1
0	0	0	0	1	0	22	1	0	1	1	0
0	0	0	0	0	1	25	1	1	0	0	1

图 1-4 微程序入口地址表

在输入相应的机器指令译码信号和微程序入口后，会在微程序查找逻辑自动生成表中自动生成 S0~S4 的逻辑表达式，将逻辑表达式输入 logisim 的分析组合逻辑电路中，即可自动生成相应的逻辑电路。电路图和封装如图 1-5 所示。

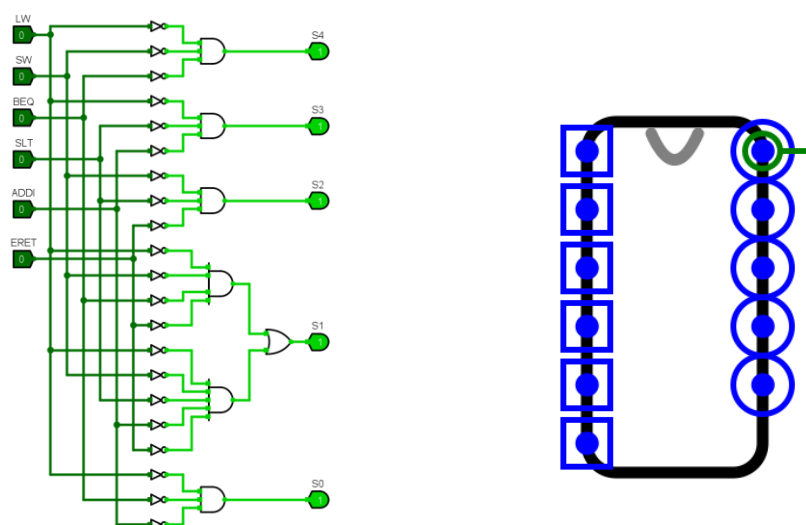


图 1-5 支持中断的微程序入口查找逻辑电路设计及封装

1.2.3 支持中断的微程序条件判别测试逻辑

设计原理：微指令执行过程中，判别测试字段和状态条件决定着接下来所选择的微指令地址。根据条件判别测试逻辑判别结果，即多路选择信号，决定微程序控制器获得下一步微指令的地址。

设计思路：通过指令执行的状态转化，根据不同测试位的具体功能和起作用的时机，一一列举出条件判别测试逻辑对应输出的多路选择信号。

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	Out8	Out9
0	0	0			0	0	0						
1					0	0	1						
0	1		1		0	1	0						
0		1	0	0	1	0	0						
0		1		1	0	1	1						
0	1	0	0		1	0	0						
0	0	1		0	1	0	0						

图 1-6 判别测试逻辑关系 excel 图表

设计过程：通过以下逻辑进行判别测试逻辑关系 excel 图表的填写，填写的结果请见图 1-6 所示，对应的电路图及封装如图 1-7 所示：

- (1) $P0P1P2 = 000$ 时，继续执行下一条微指令， $MulSex = 0$;
- (2) $P0 = 1$ 时，微程序根据指令译码情况选择指令对应的微程序入口地址，因此 $MulSex = 1$;
- (3) $P1 = 1$ 且 $equal = 1$ 时，执行 beq 分支， $MulSel = 2$;

- (4) $P2 = 1$ 且 $IntR = 1$ 时, 执行中断分支, $MulSel = 3$;
- (5) $P0$ 的优先级高于 $P1$, $P1$ 的优先级高于 $P2$;
- (6) 其余情况下应当返回取指微程序入口, 即 $MulSel = 4$ 。

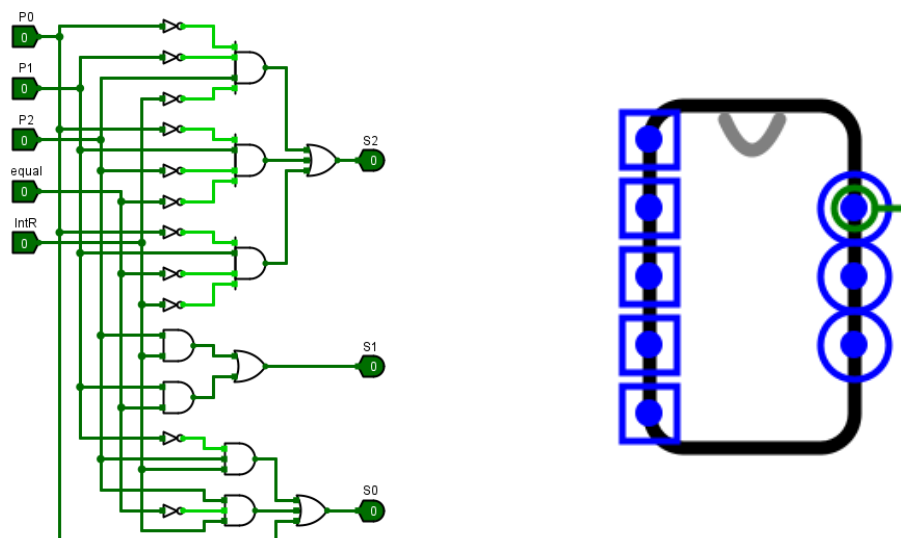


图 1-7 判别测试逻辑电路图及其封装

1.2.4 支持中断的微程序控制器设计

设计原理：根据微程序控制器的结构完成连线，实现微程序控制器的主要数据通路，设计微程序并加载到控制存储器中。

设计思路：利用前面实验实现的指令译码器，微程序入口查询地址和判别测试逻辑，再根据 Excel 设计不同指令对应的一系列微指令，存入微程序存储器。

设计过程：首先将对应的地址接入多路选择器的输入端，通过已经实现的指令译码器进行指令的译码，通过微程序入口查询地址查找指令对应的微指令的地址，通过判别测试逻辑生成多路选择信号决定下一微指令的入口地址，不断获取每一时钟周期对应的微指令。微程序控制器中对应的控制字段则通过 excel 填表自动生成表达式逻辑来实现。

Excel 填表结果如图 1-8 所示。

华中科技大学课程实验报告

	指令字编号	PCin	PCout	Zero	Out	Min	Max	PCin	Rin	DRin	Xin	Rin	IRin	P3in	Rz/Rt	Negate	Add	AddH	Slt	Read	Mult	CPIn	Mem	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										00000000000000000000000000000000	800
取指令	2		1					1		1										1									00100001010000000000000000000000	8500200
取指令	3			1									1													1			01000000000010000000000000000100	1001000A
lw	4				1							1																	00010000000010000000000000000000	4040000
lw	5					1												1											00001000000000000000000000000000	2001000
lw	6			1					1																				00100000100000000000000000000000	8200000
lw	7									1											1								00000000010000000000000000000000	100200
lw	8		1										1														1		11000000000100000000000000000001	10020001
sw	9				1							1																	00010000000100000000000000000000	4040000
sw	10					1												1											00001000000000000000000000000000	2001000
sw	11			1					1																				00100000100000000000000000000000	8200000
sw	12				1							1					1												00010000010000100000000000000000	4084000
sw	13						1																			1			00100000100000000000000000000001	800101
beq	14					1						1															1		00010000010000000000000000000001	4040000
beq	15						1									1	1									1			00010000000000010000000000000010	40C002
beq	16		1									1																	10000000000100000000000000000000	20040000
beq	17						1											1											00000100000000000100000000000000	1001000
beq	18			1					1																			1	00100001000000000000000000000001	8400001
slt	19				1							1																	00010000000100000000000000000000	4040000
slt	20					1															1								00010000000000010001000000000000	4004400
slt	21						1						1					1									1		00100000000001000100000000000000	8022001
addi	22					1						1																	00010000000100000000000000000000	4040000
addi	23						1											1											00001000000000000100000000000000	2001000
addi	24				1								1														1		00100000000100000000000000000001	8020001
eret	25								1													1				1			10000001000000000000000001001001	400091
中断	26	1																											1000000000000000000000000001001000	20000048
中断	27								1																	1		1	10000001000000000000000000000001	400021

图 1-8 微程序表达式自动生成表格

最终生成电路图如图 1-9 所示。

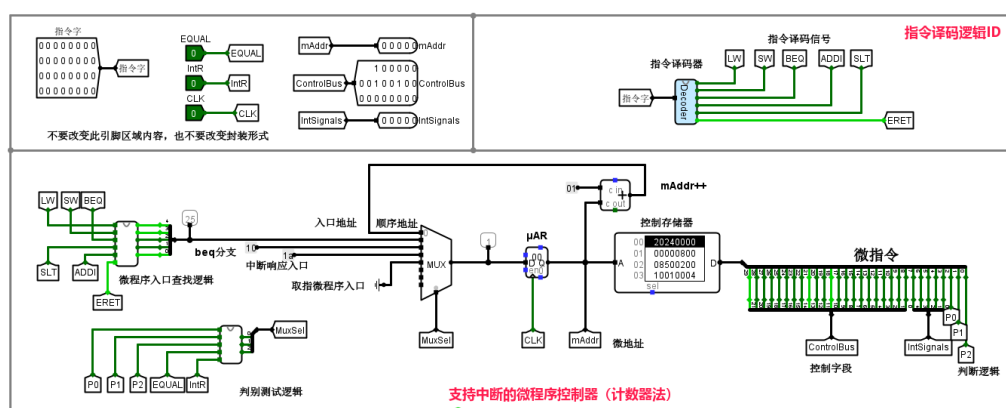


图 1-9 支持中断的微程序控制器电路图

1.2.5 支持中断的微程序单总线 CPU 设计

设计原理: 升级数据通路, 使得其支持中断请求。完善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, 同时在电路中将这此模块进行联通, 实现支持中断的微程序单总线 CPU 设计。

设计思路：中断发生时保存断点，利用内部总线和三态门将断点地址存入 EPC 中用于后续恢复现场。中断程序入口地址进入内部总线，则需要中断使能寄存器输入的是开中断，同时中断控制器有中断指令输出。

设计过程:

- (1) 完善 EPC, 通过三态门控制内部总线中对应地址 (断点) 的保存和输出。

华中科技大学课程实验报告

- (2) 完善中断使能寄存器, 通过当前中断寄存器的状态和中断信号, 判断中断请求。
- (3) 完善中断地址的查找, 利用 MARS 汇编器汇编源程序查看 label 地址, 获取中断服务的入口地址。具体如图 1-10 所示。

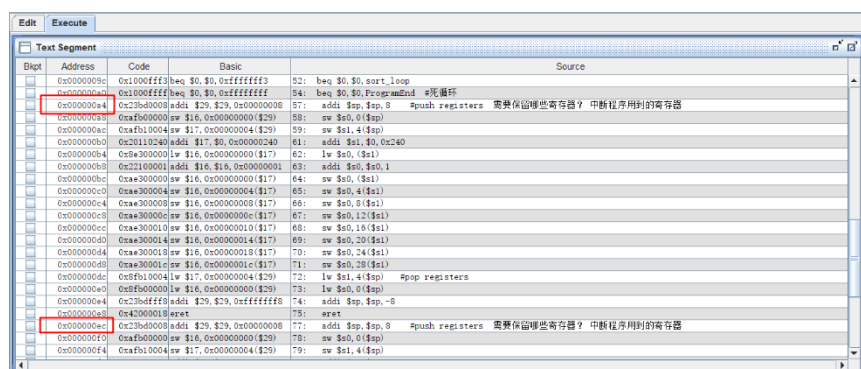


图 1-10 MARS 汇编器汇编源程序

最终实现的中断逻辑如图 1-11 所示。

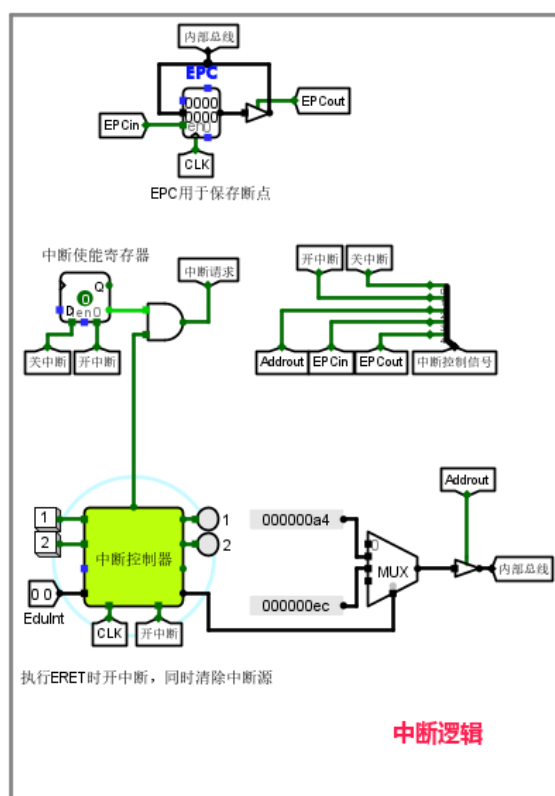


图 1-11 MIPS CPU 中断逻辑

华中科技大学课程实验报告

1.2.6 支持中断的现代时序硬布线控制器状态机设计

设计原理：硬布线控制器状态机的功能是根据反馈信号以及指令译码信号，现态信号作为输入，输出次态信号。

设计思路：构造出所有现态信号时候会发生的输入信号的不同情况，根据上述图 1.3 支持中断的现代时序状态机以及状态转移图表 excel，生成对应的逻辑表达式，输入 logisim 生成组合逻辑电路图。状态转移图表如图 1-12 所示。

当前状态(现态)						输入信号								下一状态(次态)					
S4	S3	S2	S1	S0	现态 10进制	LW	SW	BEQ	SLT	ADDI	URET	IR	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	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
0	1	0	0	0	8									0	0	0	0	0	0
0	1	0	0	1	9									10	0	1	0	1	0
0	1	0	1	0	10									11	0	1	0	1	1
0	1	0	1	1	11									12	0	1	1	0	0
0	1	1	0	0	12									13	0	1	1	0	1
0	1	1	1	0	14									15	0	1	1	1	1
0	1	1	1	1	15									0	0	0	0	0	0
0	1	1	1	1	15								1	16	1	0	0	0	0
1	0	0	0	0	16									17	1	0	0	0	1
1	0	0	0	1	17									18	1	0	0	1	0
1	0	0	1	0	18									0	0	0	0	0	0
1	0	0	1	1	19									20	1	0	1	0	0
1	0	1	0	0	20									21	1	0	1	0	1
1	0	1	0	1	21									0	0	0	0	0	0
1	0	1	1	0	22									23	1	0	1	1	1
1	0	1	1	1	23									24	1	1	0	0	0
1	1	0	0	0	24									0	0	0	0	0	0
0	0	0	1	1	3						1			25	1	1	0	0	1
1	1	0	0	1	25									0	0	0	0	0	0
1	1	0	0	1	25							1	0	26	1	1	0	1	0
0	1	0	0	0	8							1	0	26	1	1	0	1	0
0	1	1	0	1	13							1	0	26	1	1	0	1	0
0	1	1	1	1	15							1	0	26	1	1	0	1	0
1	0	0	1	0	18							1	0	26	1	1	0	1	0
1	0	1	0	1	21							1	0	26	1	1	0	1	0
1	1	0	0	0	24							1	0	26	1	1	0	1	0
1	1	0	1	0	26									27	1	1	0	1	1
1	1	0	1	1	27									0	0	0	0	0	0
1	0	0	1	0	18							1	1	26	1	1	0	1	0

图 1-12 硬布线状态转移图表

设计过程：首先进行取指令的状态转移，直到判断出现态为 3 的时候有六种不同的输入信号，导致不同的次态发生，其中 beq 分支在微程序执行到状态 15 的时候，要

判断输入信号 `equal` 是否为 1, 据此来分别决定次态为 16 还是直接进入判断中断状态。8, 13, 15, 18, 21, 24, 25 状态需要判断是否有中断发生, 发生则进入中断响应, 次态为 26, 进行相应的中断响应操作; 若是没有中断则次态直接返回取值指令的第一条微指令, 即状态 0 处。其他的情况下, 次态为现态递增。据此完成 Excel 表格的填写, 获取次态的逻辑表达式, 利用 logisim 根据表达式构造组合逻辑电路的功能直接画出响应电路, 电路部分截图如图 1-13 所示。

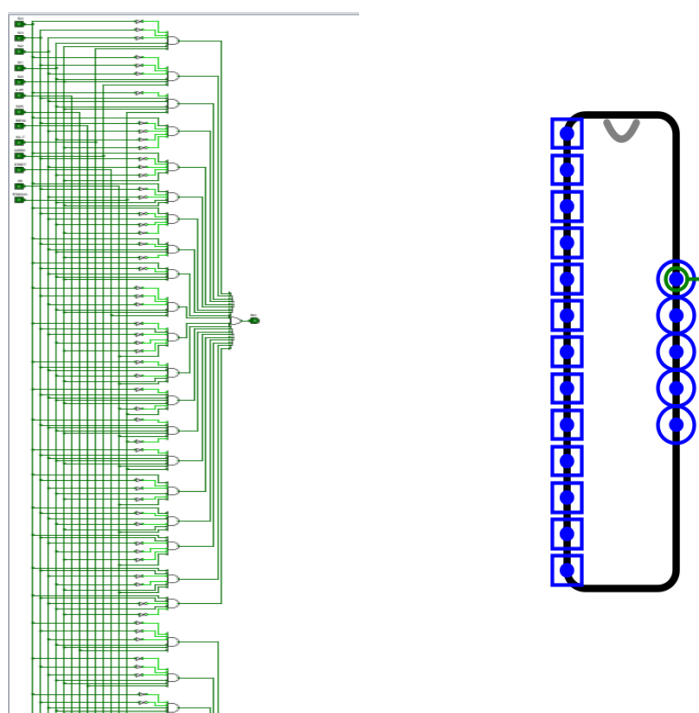


图 1-13 硬布线控制器状态机部分电路图及封装

1.2.7 支持中断的现代时序硬布线控制器设计

设计原理: 在实现指令译码、现代时序状态机模块后, 最终实现硬布线控制器的集成。

设计说明: 利用状态机接收输入的状态信号和指令译码信号确定次态信号, 同时利用状态寄存器来控制状态随着时钟跳变而改变, 此处的硬布线控制器组控制存储器无需再次实现, 直接使用之前构造的微程序控制器的控制存储器替代即可。

设计过程: 完成硬布线控制器框架连接, 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, 完成测试后用硬布线控制器替换 MIPS 中的

微程序控制器进行系统联调。

1.3 实验步骤

- (1) 首先查询指令对应的特征指令码，然后将指令码作为比较器的输入端进行比较，将指令字转换为对应的译码指令信号；
- (2) 构建微程序控制器：
 - a) 实现微程序的入口查找逻辑。微指令入口查找逻辑根据状态图，将不同的指令移码信号映射到唯一的微程序入口地址处，构建电路时利用 excel 填表自动构建表达式实现。
 - b) 实现微程序的条件判别测试逻辑。根据测试字段和状态信号决定微程序下一入口地址的类型，即通过多路选择信号选择下一微指令的地址，通过 excel 填表实现。
 - c) 完善微程序控制器的逻辑。入口查找逻辑的输出即为入口地址分支，判别测试逻辑的输出则为后续地址多路选择器选择端，在多路选择器按照设计将不同类型地址依次放到多路选择器的输入端，最后把输出的后续地址输入控制存储器。控制存储器中储存的微指令字通过 excel 表格自动生成对应指令字，加载到控制器中实现。控制存储器根据输入的地址输出相应的微指令。
- (3) 构建硬布线控制器：
 - a) 设计状态机：通过对于支持中断的现代时序状态机进行状态机的设计，最终通过填充 excel 表格完成状态机的逻辑电路设计。
 - b) 完善硬布线控制器：思路与完善微程序控制器类似。
- (4) 最终构造完整的单总线结构 MIPS CPU，升级数据通路，使得其支持中断服务程序。需要实现中断逻辑部分，根据 EPC 保存断点地址和开中断时候才能接收到中断信号进行中断逻辑的构建，中断程序的入口地址通过查询汇编程序获取。借用中断控制器模拟不同种类的中断以及中断结束后开中断清除中断源，恢复现场等功能。

1.4 故障与调试

1.4.1 中断请求响应问题

故障现象：中断请求没有被响应，没有执行正常的中断服务程序。异常请求对应的中断逻辑电路图如图 1-14 所示。

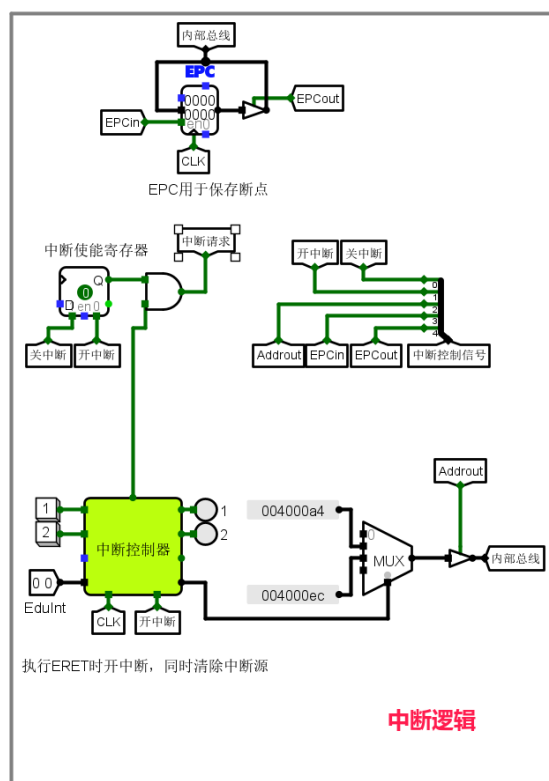


图 1-14 异常请求对应中断逻辑电路图

原因分析：中断使能寄存器中开中断对应的是异步重置触发器状态为 0，所以判断的时候接入的不应该是触发器的 Q 端，而是表示触发器当前状态的非值的端口。

解决方案：把与门判断中断请求的输入端 Q 改为 Q 的非值作为输入端。

1.4.2 判别测试逻辑问题

故障现象：判别测试逻辑不能输出正确的多路选择信号。

原因分析：对于 P1、P2、P3 之间的优先级关系不够明确，情况考虑的不够全面和彻底。

解决方案：按照优先级依次列出需要处理的逻辑，然后填入 excel 表中多次检查。

1.5 测试与分析

具体测试方式分为以下几个步骤：

- (1) 加载相应程序后判断是否能够完成排序；
- (2) 是否支持中断服务程序 1；
- (3) 是否支持中断服务程序 2；

1.5.1 支持中断的微程序单总线 CPU 测试

首先进行第一步测试，检查其是否能够完成排序操作，检查结果如图 1-15 所示，可以看出其正确通过了排序测试。

```
000 23bd0400 2010fff 20110000ae300200 2210000122310004ae30020022100001 22310004ae3002002210000122310004 ae3002002210000122310004ae300200
010 2210000122310004ae30020022100001 22310004ae3002002210000122310004 ae300200201000002011001c8e130200 8e3402000274402a11000002ae330200
020 ae140200 2231ffc 12110001 1000fff7 221000042011001c12110001 1000fff3 1000fff 23bd0008afb00000 afb10004 203102408e30000022100001ae300000
030 ae300004ae300008ae30000cae300010 ae300014ae300018ae30001c8fb10004 8fb00000 23bdfff8 4200001823bd0008 afb00000 afb10004 203102808e300000
040 2210fff ae300000ae300004ae300008 ae30000cae300010ae300014ae300018 ae30001c8fb10004 8fb00000 23bdfff8 4200001800000000000000000000000000
050 000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
060 000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
070 000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
080 00000006000000050000000400000003 000000020000000100000000 ffffff 00000000000000000000000000000000 00000000000000000000000000000000
```

图 1-15 排序测试结果示意图

然后进行第二步操作，通过按键 1 进行中断服务程序 1 的调用，同时连续两次进行调用，测试结果如图 1-16 和图 1-17 所示。

```
000 23bd0400 2010fff 20110000ae300200 2210000122310004ae30020022100001 22310004ae3002002210000122310004 ae3002002210000122310004ae300200
010 2210000122310004ae30020022100001 22310004ae3002002210000122310004 ae300200201000002011001c8e130200 8e3402000274402a11000002ae330200
020 ae140200 2231ffc 12110001 1000fff7 221000042011001c12110001 1000fff3 1000fff 23bd0008afb00000 afb10004 203102408e30000022100001ae300000
030 ae300004ae300008ae30000cae300010 ae300014ae300018ae30001c8fb10004 8fb00000 23bdfff8 4200001823bd0008 afb00000 afb10004 203102808e300000
040 2210fff ae300000ae300004ae300008 ae30000cae300010ae300014ae300018 ae30001c8fb10004 8fb00000 23bdfff8 4200001800000000000000000000000000
050 000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
060 000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
070 000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
080 00000006000000050000000400000003 000000020000000100000000 ffffff 00000000000000000000000000000000 00000000000000000000000000000000
090 00000001000000010000000100000001 000000010000000100000000 00000000 00000000000000000000000000000000 00000000000000000000000000000000
```

图 1-16 中断服务程序 1 调用第一次

```
000 23bd0400 2010fff 20110000ae300200 2210000122310004ae30020022100001 22310004ae3002002210000122310004 ae3002002210000122310004ae300200
010 2210000122310004ae30020022100001 22310004ae3002002210000122310004 ae300200201000002011001c8e130200 8e3402000274402a11000002ae330200
020 ae140200 2231ffc 12110001 1000fff7 221000042011001c12110001 1000fff3 1000fff 23bd0008afb00000 afb10004 203102408e30000022100001ae300000
030 ae300004ae300008ae30000cae300010 ae300014ae300018ae30001c8fb10004 8fb00000 23bdfff8 4200001823bd0008 afb00000 afb10004 203102808e300000
040 2210fff ae300000ae300004ae300008 ae30000cae300010ae300014ae300018 ae30001c8fb10004 8fb00000 23bdfff8 4200001800000000000000000000000000
050 000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
060 000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
070 000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
080 00000006000000050000000400000003 000000020000000100000000 ffffff 00000000000000000000000000000000 00000000000000000000000000000000
090 00000002000000020000000200000002 000000020000000200000002 00000002 00000000000000000000000000000000 00000000000000000000000000000000
```

图 1-17 中断服务程序 1 调用第二次

然后进行第三步操作，通过按键 2 进行中断服务程序 2 的调用，同时连续两次进行调用，测试结果如图 1-18 和图 1-19 所示。

华中科技大学课程实验报告

```
000 23bd0400 2010fff 20110000ae300200 2210000122310004ae30020022100001 22310004ae3002002210000122310004 ae3002002210000122310004ae300200
010 2210000122310004ae30020022100001 22310004ae3002002210000122310004 ae300200221000002011001c8e130200 8e3402000274402a11000002ae330200
020 ae140200 2231fff 12110001 1000fff7 221000042011001c12110001 1000fff3 1000fff 23bd0008afb00000 afb10004 203102408e30000022100001ae300000
030 ae300004ae300008ae30000cae300010 ae300014ae300018ae30001c8fb10004 8fb00000 23bdfff8 4200001823bd0008 afb00000 afb10004 203102808e300000
040 2210fff ae300000ae300004ae300008 ae30000cae300010ae300014ae300018 ae30001c8fb10004 8fb00000 23bdfff8 42000018000000000000000000000000
050 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
060 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
070 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
080 000000060000000050000000400000003 000000020000000100000000 fffffff 00000000000000000000000000000000 00000000000000000000000000000000
090 00000002000000020000000200000002 00000002000000020000000200000002 00000000000000000000000000000000 00000000000000000000000000000000
0a0 fffffff fffffff fffffff fffffff fffffff fffffff fffffff fffffff 00000000000000000000000000000000 00000000000000000000000000000000
```

图 1-18 中断服务程序 2 调用第一次

```
000 23bd0400 2010fff 20110000ae300200 2210000122310004ae30020022100001 22310004ae3002002210000122310004 ae3002002210000122310004ae300200
010 2210000122310004ae30020022100001 22310004ae3002002210000122310004 ae300200221000002011001c8e130200 8e3402000274402a11000002ae330200
020 ae140200 2231fff 12110001 1000fff7 221000042011001c12110001 1000fff3 1000fff 23bd0008afb00000 afb10004 203102408e30000022100001ae300000
030 ae300004ae300008ae30000cae300010 ae300014ae300018ae30001c8fb10004 8fb00000 23bdfff8 4200001823bd0008 afb00000 afb10004 203102808e300000
040 2210fff ae300000ae300004ae300008 ae30000cae300010ae300014ae300018 ae30001c8fb10004 8fb00000 23bdfff8 42000018000000000000000000000000
050 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
060 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
070 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000
080 000000060000000050000000400000003 000000020000000100000000 fffffff 00000000000000000000000000000000 00000000000000000000000000000000
090 00000002000000020000000200000002 00000002000000020000000200000002 00000000000000000000000000000000 00000000000000000000000000000000
0a0 fffffff fffffff fffffff fffffff fffffff fffffff fffffff fffffff 00000000000000000000000000000000 00000000000000000000000000000000
```

图 1-19 中断服务程序 2 调用第二次

1.5.2 支持中断的硬布线单总线 CPU 测试

测试逻辑与支持中断的微程序单总线 CPU 测试基本相同，测试结果均正确。

1.5.3 头歌平台单总线 CPU 测试

头歌平台测试均已通过，平台测试如图 1-20 所示。



图 1-20 头歌平台通过测试

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
 - a) 根据 MIPS 指令手册设计并完成指令译码器；
 - b) 设计并完成微程序入口查找逻辑和条件测试逻辑模块，实现了支持中断的微程序控制器；
 - c) 设计并完成支持中断的硬布线控制器的状态机，构造了硬布线控制器；
 - d) 升级数据通路，完善单总线 MIPS CPU 的中断逻辑。

2) 功能总结

- a) 实现了基于微程序的支持中断的单总线 CPU, 能够完成五条简单指令的执行和简单程序的处理;
- b) 实现了基于硬布线的支持中断的单总线 CPU, 能够完成五条简单指令的执行和简单程序的处理。

1.7 实验心得

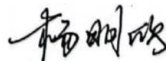
- 1) 本课程实验, 通过 logisim 连接器件, 从最基础的逻辑门开始构建一个完整的 CPU, 使得我更加熟悉了 logisim 的各个部件, 通过使用 logisim 输入逻辑表达式自动生成组合逻辑电路, 极大加速了电路设计的速度。
- 2) 通过比较三级时序和现代时序的构建方法, 使得我对于不同 CPU 处理器设计时的共通之处和不同之处更加理解, 同时也更加突出展现了不同的时序方案的优劣之处, 同时增进了对这两种时序的执行状态变化情况。
- 3) 虽然我们并没有完整实现整个 CPU、总线、ALU 和数据通路等, 但是通过老师绘制的整个单总线 CPU, 我对于课程中所学习的数据的传输以及其他的部件也有了更加深刻的了解。

• 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：杨明欣 

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：_____