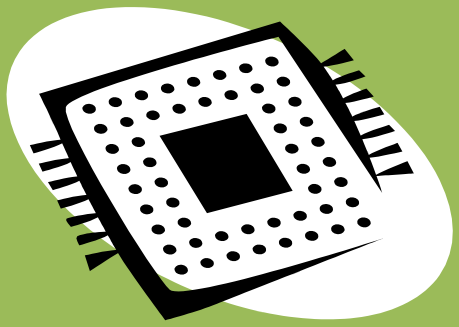
华中科技大学

2020

计算机组成原理 ·实验报告·

专　　业：	计算机科学与技术
班　　级：	CS1806
学　　号：	U201814655
姓　　名：	杨雨鑫
电　　话：	18162318665
邮　　件：	2056404413@qq.com
完成日期：	2020-12-7



计算机科学与技术学院

目　录

1　CPU 设计实验.....2
1.1　设计要求.....2
1.2　方案设计.....2
1.3　实验步骤.....10
1.4　故障与调试.....11
1.5　测试与分析.....11
2　总结与心得.....13
2.1　实验总结.....13
2.2　实验心得.....13
参考文献.....15

1 CPU 设计实验

1.1 设计要求

帮助学生理解单总线结构 CPU 基本原理，能设计定长指令周期的三级时序系统，是的 MIPS 程序能在单总线结构上运行，最终能运行简单的排序程序 sort-5.hex。在 RAM 中加载 sort-5.hex 程序，ctrl+k 自动运行，程序应该运行至 0xbbb 节拍停下，指令计数为 251，注意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，是死循环。

1.2 方案设计

1.2.1 单总线 CPU 设计（定长指令周期三级时序）

1. MIPS 指令译码器设计

设计思路：通过 IR 输入分离出 OP 字段和 FUNCT 字段，通过比较器对 OP 字段和对应常量进行比较，可以得出 BEQ,LW,SW,ADDI 指令。通过对 OP 字段和 FUNCT 字段与相应常量进行比较可以得出是否为 SLT 指令。设计出的电路如图 1-1 所示：

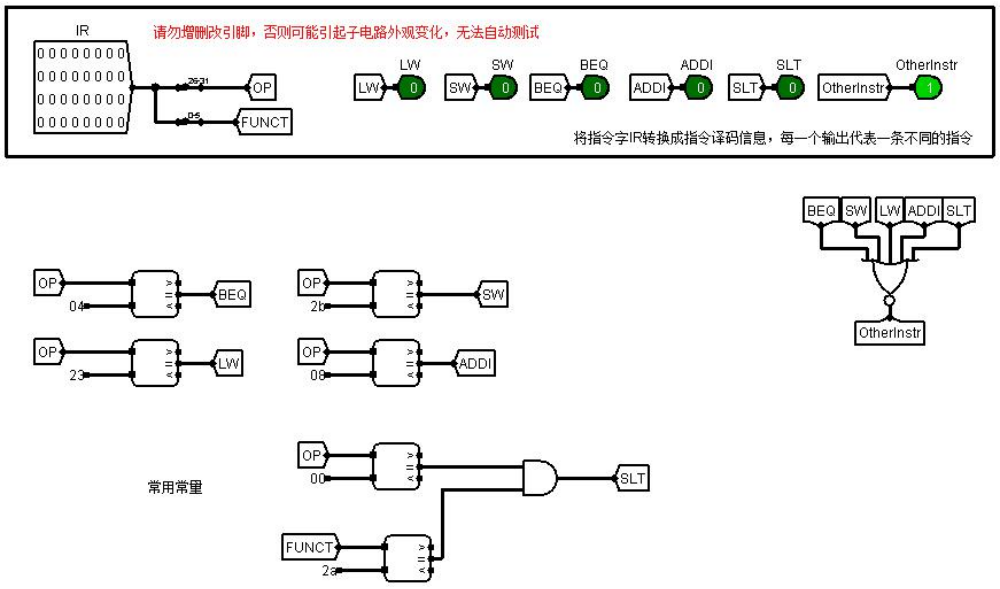


图 1-1 指令译码器

2. 时序发生器 FSM

设计思路：由于所有的 MIPS 指令都需要三个机器周期，每个机器周期四个时钟节拍，一共需要十二个状态，通过对着十二个状态按照顺序切换即可。这里我们使用 excel 表格生成逻辑表达式之后使用 logisim 中的逻辑表达式生成电路的选项生成电路，生成逻辑表达式的表格如图 1-2：

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								4	0	1	0	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								9	1	0	0	1
1	0	0	1	9								10	1	0	1	0
1	0	1	0	10								11	1	0	1	1
1	0	1	1	11								0	0	0	0	0

图 1-2 时序发生器状态表

3. 时序发生器输出函数

设计思路：通过设计的 12 个状态的切换条件，在 excel 中输入每一个状态的切换条件，生成逻辑表达式，并利用 logisim 中的逻辑表达式生成电路，具体表格见 ppt。

4. 硬布线控制器组合逻辑单元

设计思路：在前面设计好的时序发生器模块以及指令译码器的配合输入下，需要我们对每一条指令所需要执行的操作进行信号翻译，通过 MIPS 使用说明书了解了每一条指令在执行指令周期需要依次产生的信号然后填入 excel 表格，最后生成每一个输出信号的逻辑表达式并通过 logisim 的逻辑表达式生成功能生成组合逻辑电路，填入的表格如图 1-3：

输入 （填1或0，不填为无关项x）													输出 （只填写为1的情况）																							
Mif	Mcal	Mex	Mint	T1	T2	T3	T4	LW	SW	BEQ	SLT	ADDI	EQUAL	PCout	DRout	Zout	Rout	IRout	IRAout	DREout	PCin	ARin	DREin	DRin	Xin	Rin	IRin	PSWin	Rs/Rt	RegDst	Add	Add4	Slt	READ	WRITE	ControlBus （hex）
1				1										1								1			1											202400
1					1																											1				8
1						1										1					1		1											1		85002
1							1								1											1										100100
	1			1				1									1								1											40400
	1			1					1								1								1											40400
	1			1						1							1								1											40400
	1				1			1										1													1					20010
	1				1				1									1													1					20010
	1				1					1							1											1	1							400C0
		1		1				1								1						1														82000
		1		1					1							1						1														82000
		1		1						1				1											1											200400
		1		1							1						1								1											40400
		1		1								1					1								1											40400
																																				0
		1			1			1															1											1		1002
		1			1				1								1							1					1							40840
		1			1					1								1												1						10010
		1			1						1						1												1		1					40050
		1			1							1						1													1					20010
																																				0
		1				1		1							1											1										100200
		1				1			1											1														1	8001
		1				1				1			1			1					1															84000
		1				1					1					1									1					1						80220
		1				1						1				1									1											80200
																																				0
																																				0

图 1-3 硬布线组合逻辑单元输出逻辑表达式生成表

5. 硬布线控制器设计

设计思路：在实现了指令译码器，时序发生器之后，最后为了实现实现次态和现态的切换，需要使用状态寄存器来进行现态的维持并且通过时钟信号完成状态的切换。设计出的电路图如图 1-4：

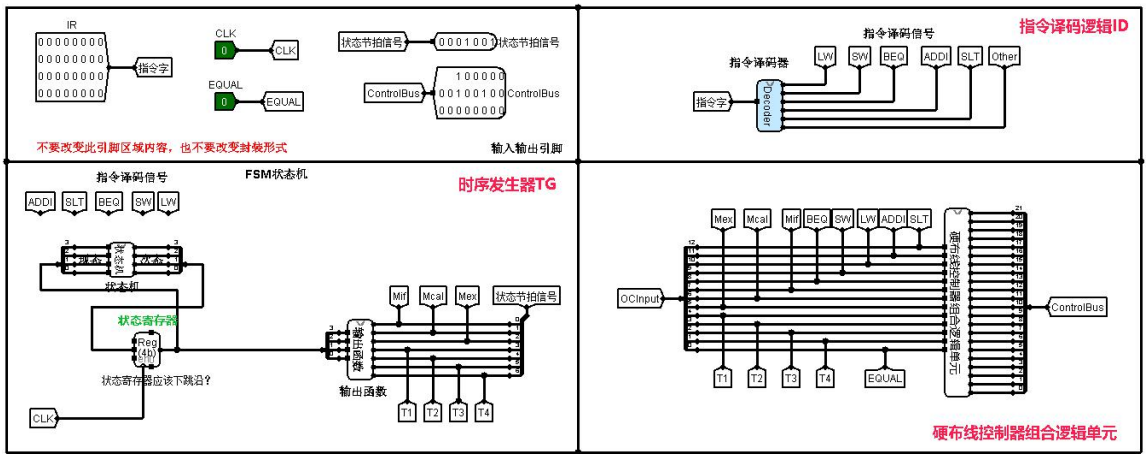


图 1-4 硬布线控制器设计电路

6. 单总线 CPU 设计

设计思路：在完成了前面的所有电路设计之后，对单总线架构的 CPU 进行调试，把 sort-5.hex 文件打开后复制进入内存，通过时钟周期连续来观察能否实现排序功能。生成的总体结构图如图 1-5：

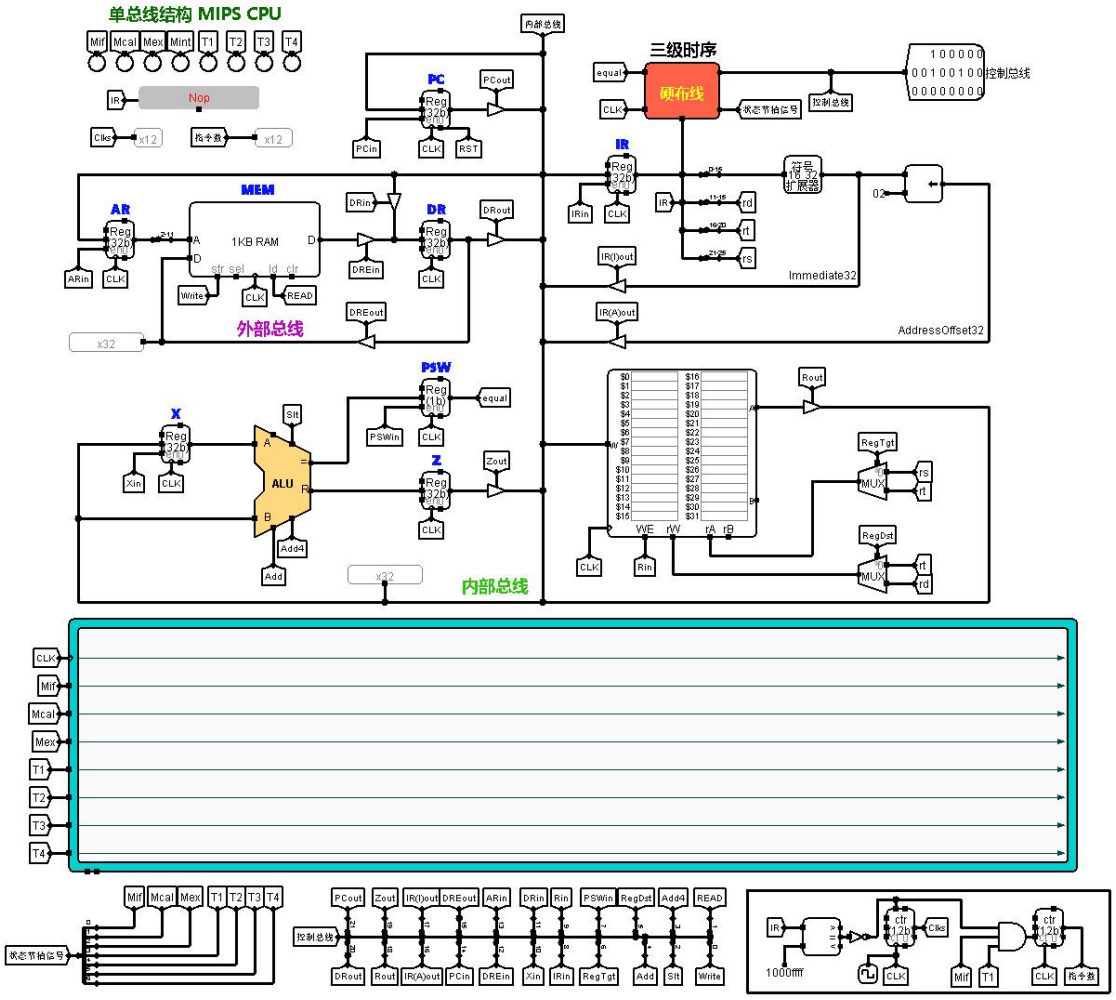


图 1-5 单总线 CPU 总体结构图

1.2.2 单总线 CPU 设计（现代时序）

1. MIPS 指令译码器的设计

设计思路：整体设计思路和 1.2.1 节的译码器设计思路一致。

2. 单总线 CPU 微程序入口查找逻辑

设计思路：为了在微指令周期的取指令周期结束后，根据输入的指令信号实现指令的状态跳转，从而进入相应的指令计算周期和执行周期，这里同样是使用 excel 表格先生成跳转分支地址的选择信号 s4s3s2s1s0 的逻辑表达式，之后使用 logisim 的逻辑表达式生成电路功能，生成组合逻辑电路。

3. 单总线 CPU 微程序条件判别测试逻辑

设计思路：通过微指令操作的判别字段，输入进入条件判别测试逻辑，从而获取

下一分支地址的选择信号 S2S1S0，这里使用 excel 生成逻辑表达式之后，利用 logisim 的逻辑表达式生成电路的功能生成条件判别测试逻辑。

4. 单总线 CPU 微程序控制器

设计思路：这里使用的是微程序设计的程序控制器，与硬布线控制器类似，需要我们根据状态图设计出 25 条指令用来表示 25 个程序状态，这里我通过查阅 MIPS 使用说明书找到每一个指令所对应的每一个状态时需要的控制信号，填入 excel 表格，通过 excel 表格生成表示每一个状态的微程序的十六进制的表达式。最后把生成的 25 条微指令程序填入控制存储器中，实现类似于硬布线控制器相同的功能，填入后的 excel 表格如图 1-6：

	PCout	Offset	Zout	Rout	IRout	RAout	RWout	PCin	ARin	DRin	DRin	Xin	Rin	IRin	PSWin	RsRt	Imdin	Add	Add4	Slt	READ	WRITE	P0	P1	P2	下址DEC	微指令	微指令十六进制
0	1								1			1														1	10000000100100000000000000000001	20240001
1																			1							2	00000000000000000010000000010	802
2			1					1		1											1					3	00100001010000000001000000011	8500203
3		1											1										1			0	01000000000001000000010000000	10010080
4				1								1														5	00010000001000000000000000101	4040005
5					1													1								6	00001000000000001000000000110	2001006
6			1						1																	7	00100000100000000000000000111	8200007
7										1											1					8	00000000010000000001000001000	100208
8		1											1													0	01000000000010000000000000000	10020000
9				1								1														10	00010000000100000000000001010	404000A
10					1													1								11	00001000000000000100000001011	200100B
11			1						1																	12	00100000100000000000000001100	820000C
12				1							1					1										13	00010000001000010000000001101	408400D
13						1																1				0	00000010000000000000100000000	800100
14				1								1														15	00010000000100000000000001111	404000F
15				1											1	1								1		0	00010000000000110000001000000	400C040
16	1											1														17	10000000000100000000000010001	20040011
17						1												1								18	00000100000000001000000010010	1001012
18			1					1																		0	00100001000000000000000000000	8400000
19				1								1														20	00010000001000000000000010100	4040014
20				1												1				1						21	00010000000000010001000010101	4004415
21			1										1				1									0	00100000000010001000000000000	8022000
22				1								1														23	00010000001000000000000010111	4040017
23					1													1								24	00001000000000001000000011000	2001018
24			1										1													0	00100000000010000000000000000	8020000
																												正确

图 1-6 微程序控制器状态表示图

5. 采用微程序的单总线 CPU

设计思路：与硬布线 CPU 类似，这里需要我们能够把 sort-5.hex 程序复制进入内存，最终实现排序程序的执行，测试的电路如图 1-7：

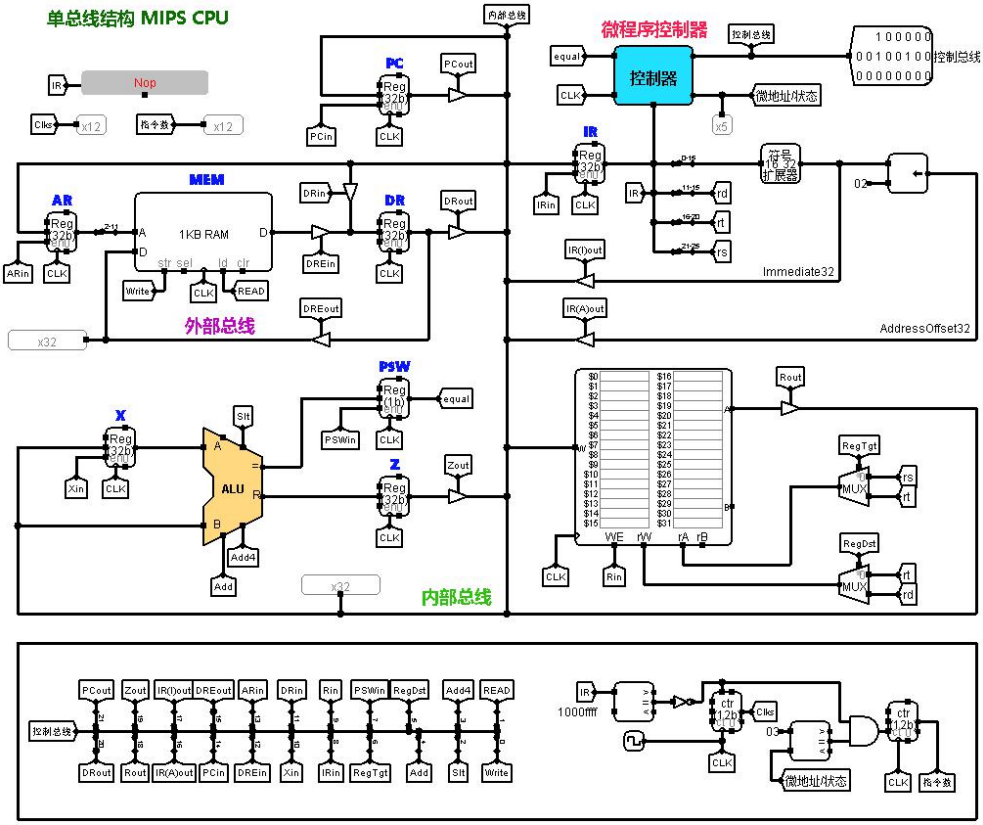


图 1-7 微程序设计的 CPU

6. 现代时序硬布线控制器状态机

设计思路：类似于传统三级时序逻辑电路，首先画出 25 个状态，做出每一个状态之间的转移关系，在表格中输入当前状态以及输入的信号所得到的次态，得到次态 N4N3N2N1N0，完成次态和现态的转换。

7. 现代时序硬布线控制器

设计思路：这里的设计和传统时序逻辑电路比较类似，唯一的不同就是输出的是微指令字段，高 8~29 位直接代表的是输出的控制信号（比如 ARin，IR(A)out 等信号）。生成的电路图如图 1-8：

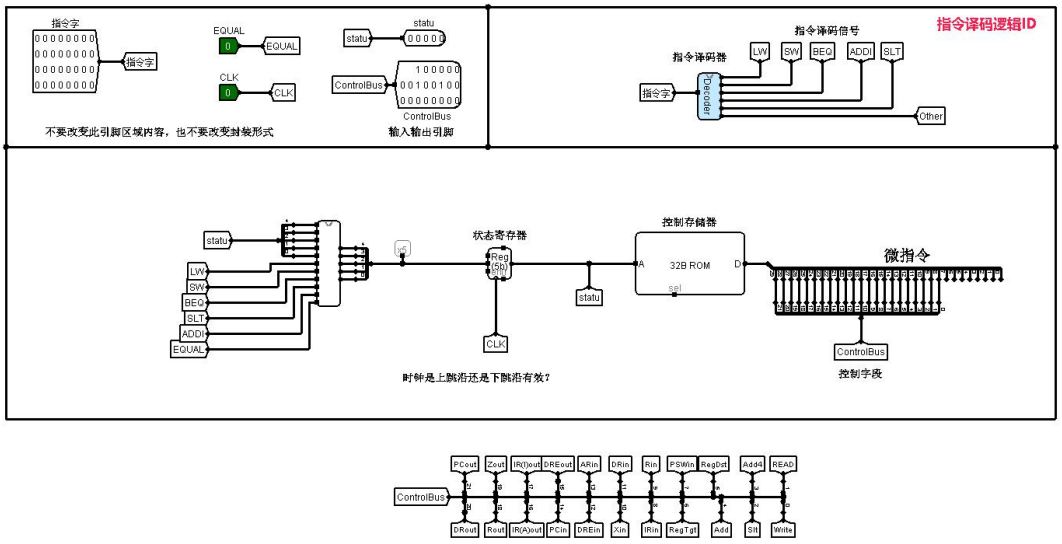


图 1-8 现代时许硬布线设计电路

1.2.3 现代时序中断机制实现

1. MIPS 指令译码器设计

设计思路：与之前两个设计一致。

2. 支持中断的微程序入口查找逻辑

设计思路：首先完成加入中断之后地址转移逻辑状态图的填写，接着填入 excel 表格，明确的指出微程序的入口地址，从而实现机器周期之间的跳转。

3. 支持中断的微程序条件判别测试逻辑

设计思路：这里的设计和微程序设计大致一样，唯一不同的就是加入了 P2 判别位配合 INTR 中断信号用来识别中断信号。生成了逻辑表达式，最后使用 logisim 中的逻辑表达式生成功能生成电路。

4. 支持中断的微程序控制器设计

设计思路：这里需要填写的表格原理和微程序的控制器设计是一致的，但是由于新添加了 P2 标志位和中断隐指令，因此需要多加上几行，此外，由于加上了中断判断，需要在每个指令的执行指令周期后跳入中断判断，如果存在中断信号，则进入中断微程序。最后填写完毕的 excel 表格如图 1-9：

	PCout	DRout	Zout	Rout	PCin	ARin	DRin	Xin	Rin	IRin	Add	Add4	Slt	READ	WRITE	EPCout	EPCin	ST1	CLI	P1	P2	P3	微指令	微指令十六进制
0	1					1		1															10000000100100000000000000000000	20240000
1												1											00000000000000000100000000000	800
2			1		1		1							1									00100001010000000000100000000	8500200
3		1								1										1			01000000000010000000000000100	10010004
4				1				1															00010000001000000000000000000	4040000
5				1							1												00001000000000001000000000000	2001000
6			1			1																	00100000100000000000000000000	8200000
7							1							1									00000000010000000001000000000	100200
8		1							1													1	01000000000010000000000000000001	10020001
9				1				1															00010000001000000000000000000	4040000
10				1							1												00001000000000001000000000000	2001000
11			1			1																	00100000100000000000000000000	8200000
12				1			1				1												00010000001000010000000000000	4084000
13					1										1							1	00000010000000000000100000001	800101
14				1				1															00010000001000000000000000000	4040000
15				1							1	1									1	1	00010000000000110000000000011	400C003
16	1							1															10000000000100000000000000000	20040000
17					1																		00000100000000001000000000000	1001000
18			1				1															1	00100001000000000000000000001	8400001
19				1				1															00010000001000000000000000000	4040000
20				1								1			1								00010000000000010001000000000	4004400
21			1						1				1									1	00100000000010001000000000001	8022001
22				1				1															00010000001000000000000000000	4040000
23				1																			00001000000000001000000000000	2001000
24			1						1													1	00100000000010000000000000001	8020001
25							1									1			1			1	00000001000000000000010010001	400091
26	1																1						10000000000000000000000001001000	20000048
27							1											1				1	00000001000000000000000100001	400021

图 1-9 支持中断的微程序控制器微指令

5. 支持中断的微程序单总线 CPU 设计

设计思路：这个设计的难点就在于设计中断硬件模块的设计，首先需要添加 EPC 模块用来存储发生中断的时候的 PC 地址，接着需要添加一个多路选择器把终端微程序的入口地址送入总线中，此外还需要添加中断控制器，在执行中断指令的时候关中断，在退出中断微程序的时候开中断。设计出来的中断模块电路图如图 1-10：

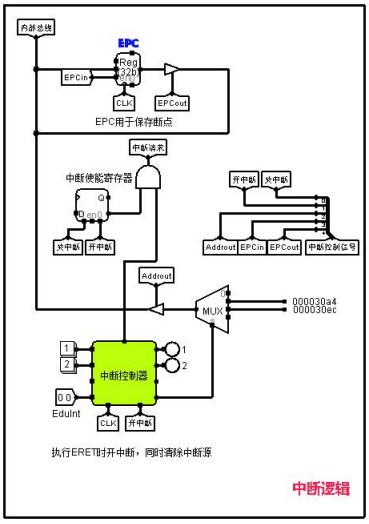


图 1-10 中断硬件模块

6. 支持中断的现代时序硬布线控制器状态机设计

设计思路：同之前不加中断的硬布线逻辑设计一样，这里只是多加了三个状态用来表示判断中断的状态。

7. 支持中断的现代时序硬布线控制器设计

设计思路：同之前的硬布线控制器设计，这里的变化是标志位增加了一位，以及多了五位中断信号用来表示中断隐周期的状态。

1.3 实验步骤

1.3.1 单总线 CPU 设计（定长指令周期三级时序）

根据设计思路，对指令译码器进行连线操作，之后依次设计时序发生器以及输出函数电路。在完成时序发生器之后，完成硬布线控制器电路设计并把时序发生器和硬布线控制器联合起来最后在完整的 CPU 电路对所有模块进行联调，验证 CPU 是否能够按照要求工作。

1.3.2 单总线 CPU 设计（现代时序）

根据设计思路，首先对译码器进行连线操作，之后根据状态图设计完成 CPU 微程序入口查找逻辑以及条件判别测试逻辑电路。接着通过 excel 表格生成微指令，并放入微程序控制器寄存器中，完成和之前硬布线控制器类似的功能，最后对整个微程序 CPU 电路进行联调，验证 CPU 是否能够按照要求工作。

1.3.3 现代时序中断机制实现

根据设计思路，首先对译码器进行连线操作，之后类似于微程序 CPU 设计流程，依次对微程序入口查找逻辑电路进行设计。之后在之前的微程序设计的 excel 表格中加入新的标记位以及中断隐周期的对应标记位并完成填写，使新设计的微指令具有中断功能，最后两个硬布线控制器设计电路同理，也只需要加上新增的几个中断状态就可以通过 excel 在原来的基础上完成电路设计。

1.4 故障与调试

1.4.1 微程序控制器跳转错误

故障现象：微程序控制器不能成功的按照设定的要求跳转。

图 1-11 故障一

原因分析：如图 1-11，在 educoder 上面一直显示我的下址跳转地址错误，后来才发现我这里的判别逻辑模块的封装是高位在上，低位在下的，这会导致产生的两位多路选择信号错误从而无法正确的选择下址地址。

解决方案：交换判别测试逻辑的两条输出线的连线即可。

1.4.2 中断微程序入口地址的寻找

故障现象：按照之前设计的的中断电路，需要我输入微程序的入口地址，从而让系统成功切换进入微程序。

原因分析：通过在 sort5-int.asm 文件中寻找到了中断微程序的两个入口，但是由于对地址定位不准确导致没有成功切换。

解决方案：直接删除掉所有的注释行以及把段标识符全部都去掉之后得到的微程序入口地址才是真正的地址，然后还需要加上 0X3000 才能够得到真正的物理地址。最终得到的地址为 0x000030a4 和 0x000030ec。

1.5 测试与分析

1.5.1 三级时序 CPU 设计排序结果存储位置错误

现象：传统三级时序逻辑电路在运行了排序程序之后，发现并没有按照之前设计的把排序结果存储在 0x80 的位置，而是存储在了 0x00 的位置并且把之前的程序指令

给覆盖了。

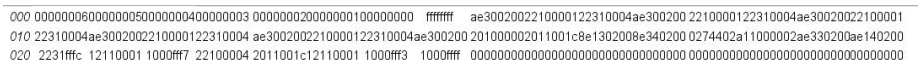


图 1-12 传统三级时序 CPU 排序结果截图

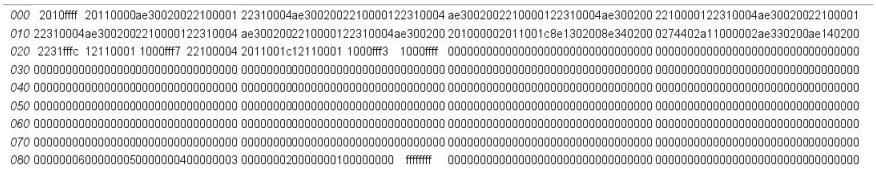


图 1-13 微程序 CPU 排序结果截图

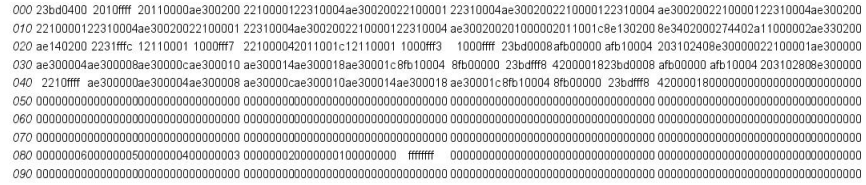


图 1-14 包含中断逻辑的微程序 CPU 排序结果截图

原因分析：通过对 CPU 的单步调试我发现，对于传统三级时序逻辑电路的有些指令，由于在执行指令周期并没有让每一个节拍都占满，这个时候的总线默认值为 0，电路会把 0x00 这个地址送入寄存器 Z 中，这会把原本设置为地址+0x80 的操作中的 0x80 没有成功的加上去，导致在之后的放入排序结果的地址发生了改变，最终造成了上面的这种情况。但是基于微程序的另外两个排序结果都成功的把排序结果放在了 0x80 的位置处。

解决方案：可以使用变长指令周期来设计 CPU，针对每一种指令都有相应的执行周期的长度，使得每两个机器周期之间都是没有空节拍的，保证了数据的正确性。

2 总结与心得

2.1 实验总结

本次实验主要完成了如下几点工作：

1） 完成了单总线 CPU 设计（定长指令周期三级时序），完成了单总线 CPU 设计（现代时序）设计，完成了现代时序中断机制的实现设计。

2） 完成了译码器模块设计，完成了入口查找逻辑设计，完成了条件判别测试逻辑设计，完成了微程序控制器设计，完成了硬布线控制器设计，完成了对硬布线控制器和微程序控制器添加中断操作的设计。

3） 学会了使用 excel 生成逻辑表达式后通过 logisim 生成电路图，学会了通过时钟单步调试找出错误跳转位置并做出分析，学会了通过查阅 MIPS 使用说明书来分析每一个指令在每个机器周期的每个节拍引发的操作，学会了传统三级时序 CPU 和现代时序 CPU 的原理，并能分析两者的优缺点，学会了中断的具体操作流程，学会了分析底层汇编代码并在 CPU 电路中进行实时调试。

2.2 实验心得

1） 连续做了三个 CPU 实验，对于三个 CPU 实验的完成模块就可以发现设计流程基本相同，尤其是前面两个传统时序逻辑电路和现代时序逻辑电路的区别，主要是前者使用了硬布线设计，后者使用了微程序设计，前者一但设计完成之后就不能再修改了，但是后者可以直接修改微指令来完成修改，相比之下，微程序控制器更具有健壮性。最困难的一部分就是给微程序添加中断功能了，需要重新设计出 FSM 再修改微程序控制器等电路，这一部分修改起来很快，也体现出了微程序 CPU 易修改的优势。但是为了实现中断逻辑，除了在软件层次上作出修改，还需要添加寄存器用来保存断点并标记中断开关状态，这也说明了为了实现中断逻辑，需要软件和硬件的同时做出修改才能支持。在调试过程中基本上没有遇到特别多的问题，唯一的难点就是对于排序结果的分析，其实只需要我使用时钟单步执行到 512+s0 对应的那一行指令就可

以发现 Z 寄存器的值并没有按照之前设计的保存 512，而是被空节拍周期重新赋值为 0 了。整个实验做完，对于原理上的问题都完全弄懂了，但是对于一些指令上的细节还需要查阅 MIPS 指令说明书。

2） 对于这次使用的 educoder 评测平台个人感觉十分满意，能够很快的把我与测试答案不符的位置定位出来，帮助我在单步调试的时候快速定位出我是哪里出了问题，但是对于 logisim 中稍微一移动引脚就会导致封装出现问题，这个问题我在之前的几个实验中都遇到了，希望能够在 logisim 中添加锁死按钮，把设计好的封装直接锁死，防止我们误操作导致封装改变从而评测失败，此外希望额能够在使用逻辑表达式生成电路的选项中，把最大 12 个输入引脚的限制调整为 16 位或者是 32 位，以免出现超出 12 个输入引脚就不能生成电路的现象。希望 logisim 软件的越做越好，也感谢老师为了实验辛辛苦苦准备的实验学习资料！

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,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.

[4] 秦磊华，吴非，莫正坤.计算机组成原理. 北京:清华大学出版社，2011 年.

[5] 袁春风编著. 计算机组成与系统结构. 北京:清华大学出版社，2011 年.

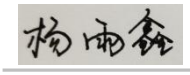
[6] 张晨曦，王志英. 计算机系统结构. 高等教育出版社，2008 年.

• 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： 

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）	报告撰写 （30 分）	课设过程 （70 分）	最终评定 （100 分）
得分			

指导教师签字：_____