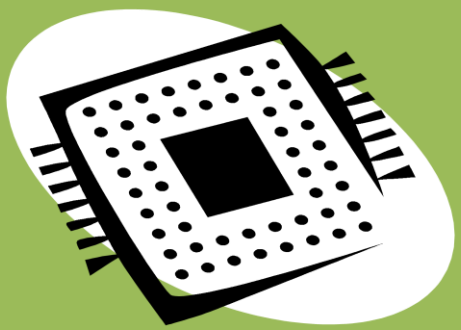
华中科技大学

2021

计算机组成原理 ·实验报告·

专　　业：	计算机科学与技术
班　　级：	CE1901
学　　号：	U201914870
姓　　名：	刘红阳
电　　话：	17307228012
邮　　件：	3184035501@qq.com
完成日期：	2021-12-18



计算机科学与技术学院

目　录

1 单总线 RISC-V CPU 设计（变长指令周期 3 级时序）实验

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。5 条指令的格式、类型和功能说明如表 1.1 所示。

表 1.1 核心指令集

#	指令	格式	类型	功能
1	lw	lw rd,imm(rs1)	I 型	R[rd] ← M[R[rs1] + SignExt(imm)]
2	sw	sw rs2,imm(rs1)	S 型	M[R[rs1] + SignExt(imm)] ← R[rs2]
3	beq	beq rs1,rs2,imm	B 型	if(R[rs1] == R[rs2]) PC ← PC + SignExt(imm) << 1
4	addi	addi rd,rs1,imm	I 型	R[rd] ← R[rs1] + SignExt(imm)
5	slt	slt rd,rs1,rs2	R 型	If (rs1 < rs2) R[rd] ← 1 else R[rd] ← 0

1.2 方案设计

1.2.1 设计思路

分别设计 cpu 的内部组成部件，最后完成各部件的连接组装，实现一个简单的 cpu。其中算术运算单元 ALU、主存 RAM、和各种寄存器 PC、AR、DR、IR、通用寄存器组的实现与连接，实验已经给出。我们主要设计的功能部件是硬布线控制器，它具有指令译码，和产生各种微控制信号的功能。为此我们需要设计指令译码器、时序发生器、和硬布线控制组合逻辑单元。

1.2.2 设计原理

如图 1.1 所示，硬布线控制器由指令译码器、时序发生器、硬布线控制组合逻辑单元三部分组成。

指令译码器对来自指令寄存器 IR 的指令进行译码，产生译码信号。译码信号送入时序发生器，在时钟信号的控制下，时序发生器输出状态周期和节拍信号。最后硬布线控制器逻辑单元根据译码信号、状态周期和节拍信号通过组合逻辑输出微操作控制信号序列，由微操作控制信号序列，控制 CPU 中内部各类寄存器的输入和输出。指令译码器、时序发生器、硬布线控制组合逻辑单元三个部件内部具体设计原理见实验步骤部分。

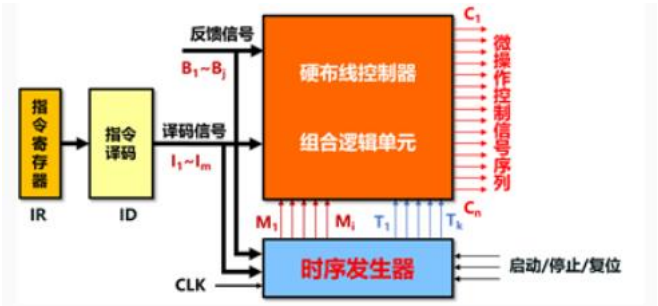


图 1.1 硬布线控制器结构图

1.2.3 单总线 CPU(3 级时序)总体设计电路图

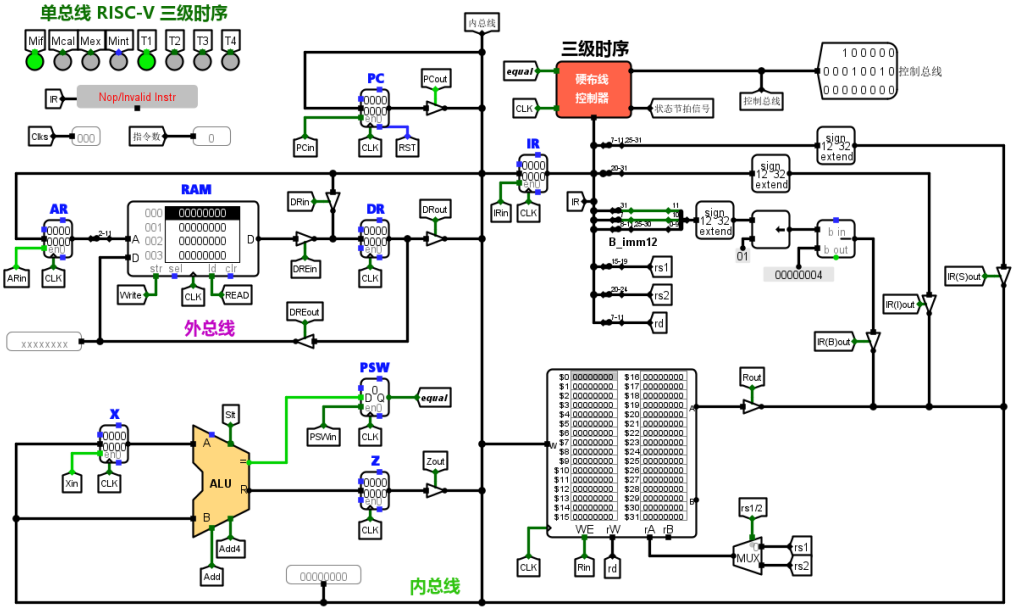


图 1.2 单总线 CPU(3 级时序)总体结构图

1.3 实验步骤

（1） 指令译码器的设计

从主存中取出指令后，根据指令的 OP 字段和 Funct3 字段的值利用比较器将指令译码为 LW、SW、BEQ、SLT、ADDI、OtherInstr 等指令译码信号。指令译码器结构图如图 1.3 所示。

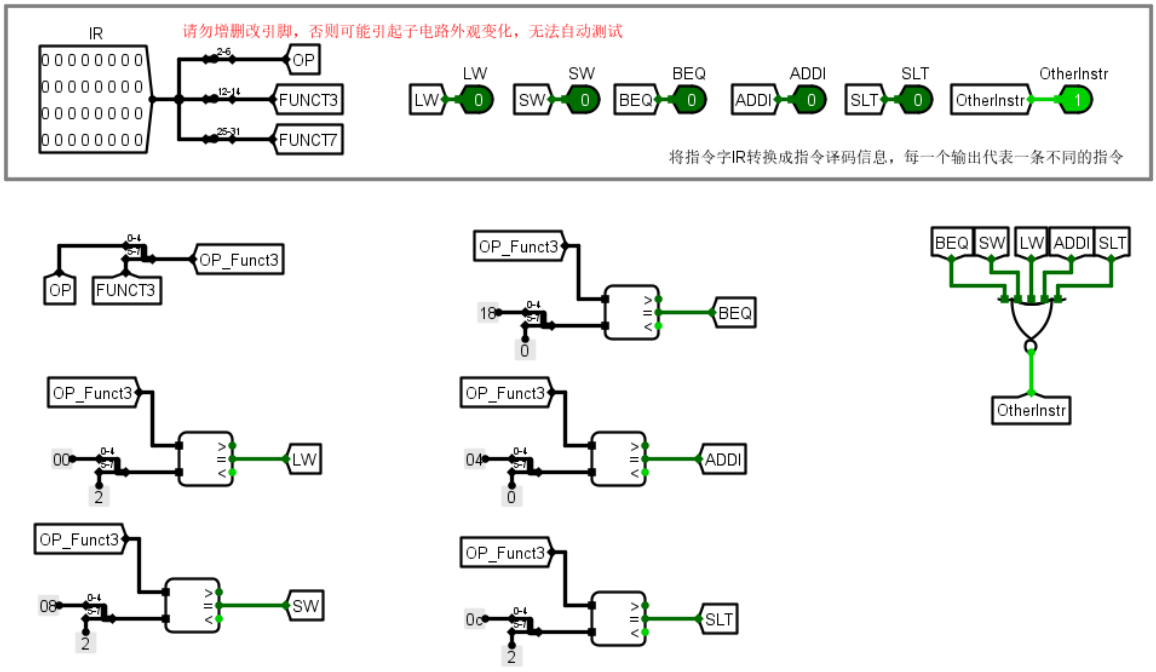


图 1.3 指令译码器结构图

（2） 时序发生器 FSM 的设计（变长指令周期）

时序发生器主要包括状态寄存器，状态机组合逻辑，输出函数组合逻辑三部分。

- ✧ 状态寄存器：时钟信号 clk 到来的时，实现现态和次态的转换。
- ✧ 状态机组合逻辑：根据现态和指令译码信号输出次态
- ✧ 输出函数组合逻辑：根据现态输出状态周期和节拍电位

时序发生器结构图如图 1.4 所示

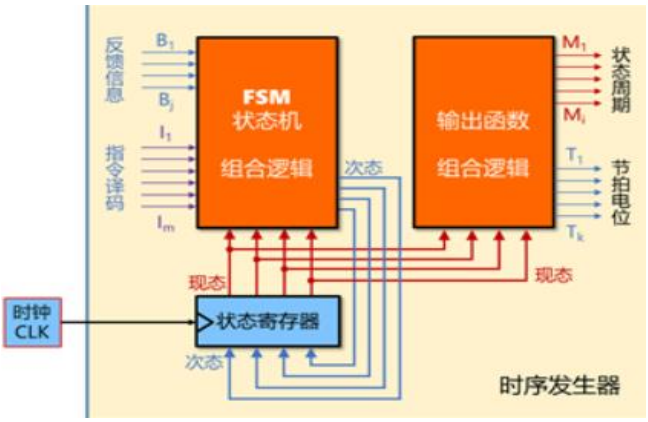


图 1.4 时序发生器结构图

因为单总线结构中采用变长指令周期，故不同指令机器周期数不同，每个机器周期节拍数也是可变化的，具体状态图如图 1.5 所示。

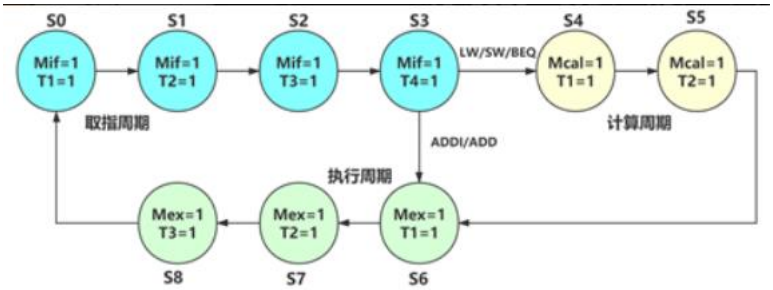


图 1.5 状态图转移图

根据图 1.4 和 1.5，状态机组合逻辑和输出函数组合逻辑部件可以根据他们对应的输入和输出设计组合逻辑电路。

（3） 硬布线控制器设计

硬布线控制器为组合逻辑电路，它的输入为状态周期、节拍电位、和指令译码信号，输出为微操作控制信号，通过填写表 1.2 输入和输出的对应关系，得到输出函数的逻辑表达式，生成相应的逻辑电路。

表 1.2 三级时序控制器控制信号产生表

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	IR(I)out	IR(S)out	IR(B)out	DREout	PCin	ARin	DREin	DRin	Xin	Rin	IRin	PSWin	rs1/rs2	Add	Add4	Slt	READ	WRITE
1				1										1									1			1									
1					1																											1			
1						1										1						1		1										1	
1							1								1													1							
	1			1				1									1									1									
	1				1			1										1													1				
		1		1				1								1							1												

1.4 故障与调试

1.4.1 状态寄存器上升下降沿问题

故障现象： 输出的状态节拍信号与预期的不一致。

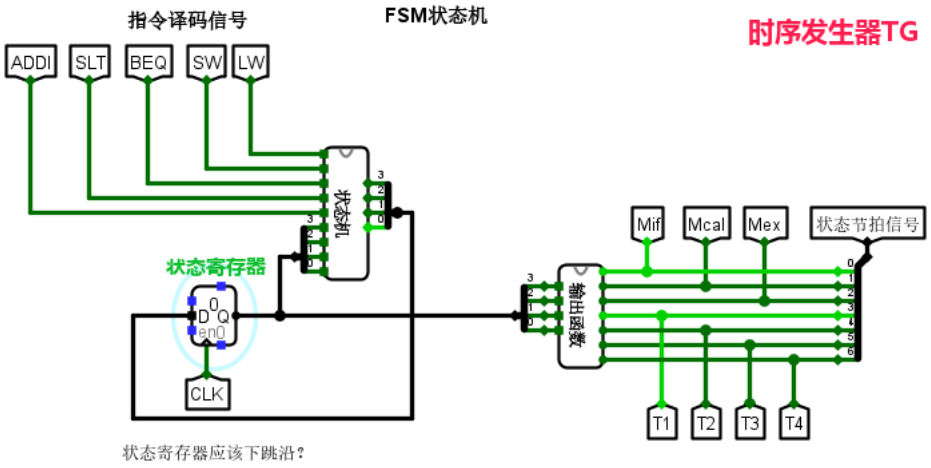


图 1.6 时序发生器结构图

原因分析： 硬布线控制器内部寄存器和外部寄存器的时钟触发信号应该不同。如果它们都在上升沿触发，当时钟信号 clk 上升沿到来时，硬布线控制器外的寄存器会根据硬布线控制器输出的微控制信号发生状态变化，而与此同时，硬布线内部的寄存器需要根据外部寄存器变化后的值来确定下一次的状态。而寄存器是有建立时间的，即在时钟上升沿，应该需要有一个稳定的输入信号，故硬布线控制器内部寄存器应采用下降沿有效，否则会出现信号混乱。

解决方案： 将状态寄存器的时钟信号设置为下降沿有效。

1.5 测试与分析

将排序程序 sort-5-riscv.hex 输入内存中，ctrl+K 运行程序后结果如图 1.7 和图 1.8 可以看到排序成功，符合预期。其他部件测试结果见 educoder 平台。

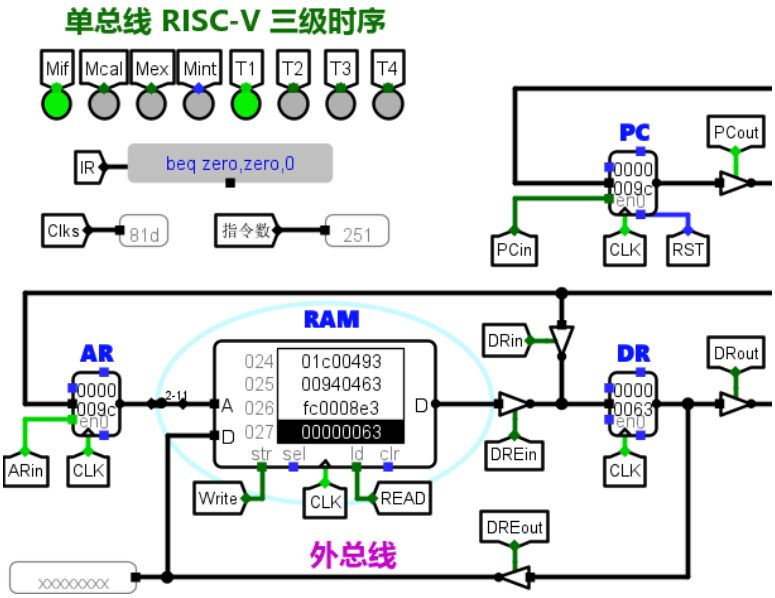


图 1.7 排序程序指令执行完毕后的结果

000	fff00413	00000493	2084a023	00140413	00448493	2084a023	00140413	00448493
008	2084a023	00140413	00448493	2084a023	00140413	00448493	2084a023	00140413
010	00448493	2084a023	00140413	00448493	2084a023	00140413	00448493	2084a023
018	00000413	01c00493	20042983	2004aa03	0149a2b3	00028663	2134a023	21442023
020	ffc48493	00940463	fe0000e3	00440413	01c00493	00940463	fc0008e3	00000063
028	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
030	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
038	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
040	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
048	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
050	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
058	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
060	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
068	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
070	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
078	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
080	00000006	00000005	00000004	00000003	00000002	00000001	00000000	ffffffff
088	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

图 1.8 排序后 RAM 中数据

2 RISC-V 现代时序中断机制实现

2.1 设计要求

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。

5 条指令的格式、类型和功能说明如表 1.1 所示。

2.2 方案设计

2.2.1 设计思路

本实验主要需要我们设计的功能就是根据指令，选择恰当的微指令入口地址，访问微指令控制存储器，输出相应的控制字段和判别字段。为此我们需要设计指令译码器、地址转移逻辑、判别测试逻辑、控制存储器等四个部件。

2.2.2 设计原理

如图 2.1 所示，微程序控制器由指令译码器、地址转移逻辑、判别测试逻辑、控制存储器四个部件组成。

指令译码器对来自指令寄存器 IR 的指令进行译码，产生译码信号。译码信号送入地址转移逻辑部件，输出入口地址。判别测试逻辑为组合电路，它的输入为微指令的判别字段和状态条件，产生选择信号，通过多路选择器选择不同的微地址到微地址寄存器 uAR 中。最后控制存储器存储着我们设计好的二进制微指令，通过 uAR 寄存器中的微地址访问存储器中微指令。每条微指令分为控制字段、中断控制信号和判断逻辑字段三部分，中断控制信号用来控制开中断、关中断和保存断点。指令译码器、地址转移逻辑、判别测试逻辑、控制存储器四个部件的具体设计原理见实验步骤部分。

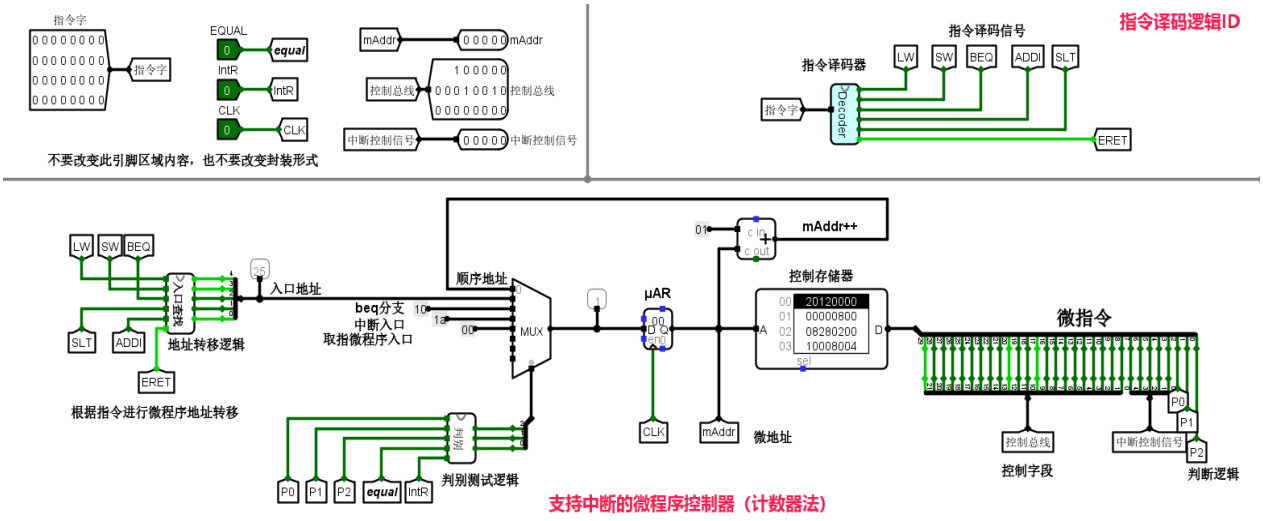


图 2.1 微程序控制器结构图

2.2.3 支持中断的微程序单总线 CPU 设计电路图

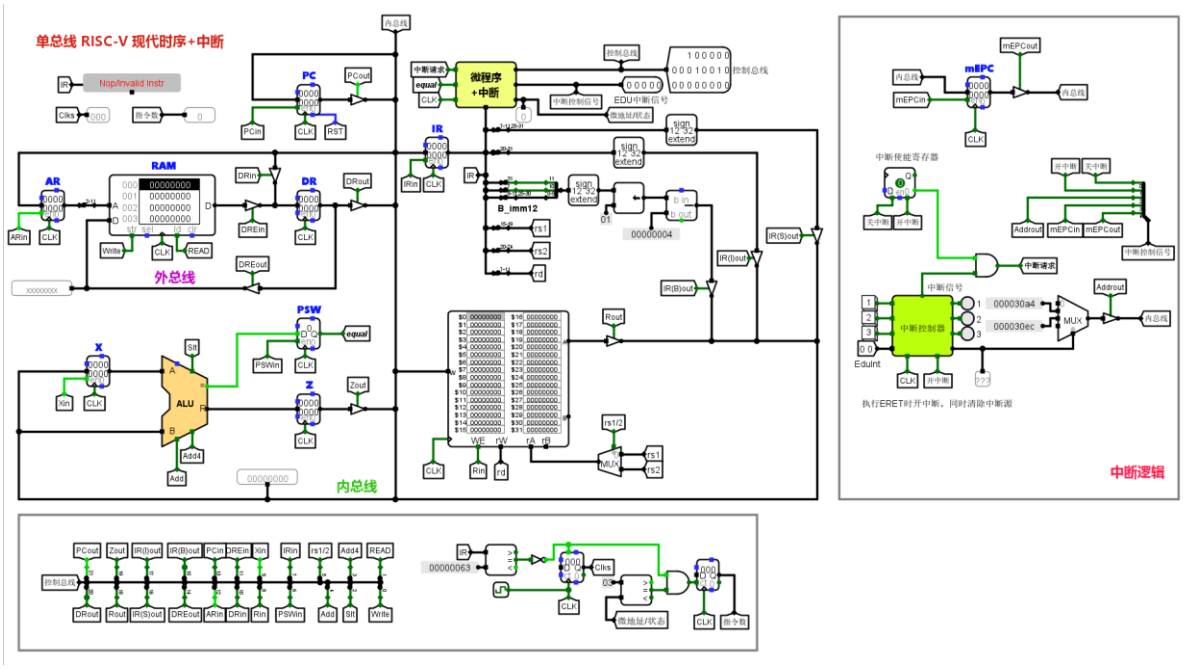


图 2.2 总体结构图

2.3 实验步骤

（1） 指令译码器的设计

同上 1.3 实验步骤指令译码器设计。

（2） 支持中断的微程序入口查找逻辑

CPU 取完指令并译码后，根据不的指令，需要执行不同的微程序。这时需要根据指令来查找微程序入口地址。根据图 2.3，可以填写如图 2.4 的表格，输入为指令译码信号，输出为微程序入口地址。根据输入输出关系，生成组合逻辑电路。

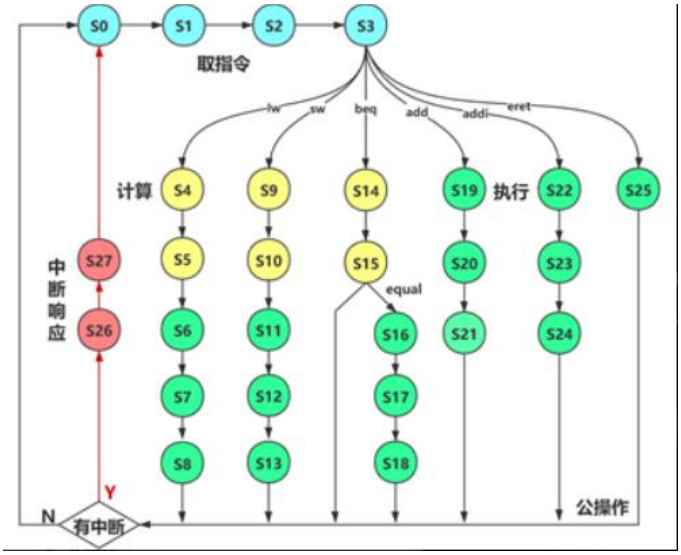


图 2.3 地址转移逻辑图

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 2.4 地址转移逻辑表

（3） 支持中断的微程序条件判别测试逻辑

由图 2.3 可知，在一次指令周期中，有多次分支，故需要判别字段$P_0P_1P_2$，

P_0 为 1 表示要根据指令功能进行微程序分支

P_0 为 1 表示要根据 equal 标志进行微程序分支

P_2 为 1 表示是微程序的最后一条微指令，可能需要进行中断响应

equal 为相等标记，IntR 为中断请求信号

$S_2S_1S_0$为输出信号，用来做多路选择器的选择信号，选择恰当的分支分析可知，当$P_0P_1P_2$为 000 时，表示微程序顺序执行，$S_2S_1S_0$为 000。当$P_0P_1P_2$为 100 时，需要根据指令进行分支，$S_2S_1S_0$为 001。当P_1为 1 且 equal 为 1 时，需要走 beq 分支，$S_2S_1S_0$为 010。当P_2为 1 且 IntR 为 1 时，需要走中断分支，$S_2S_1S_0$为 011。当P_2为 1 且 IntR 为 0 时，即没发生中断，需要走取微指令分支，$S_2S_1S_0$为 100。除此之外我们需要注意输入信号不能有二义性，故可以得到如图 2.5 所示的表格。

输入 （填1或0，不填为无关项x）							
P0	P1	P2	equal	IntR	S2	S1	S0
0	0	0	X	X	0	0	0
1	0	0	X	X	0	0	1
0	1	X	1	X	0	1	0
0	X	1	0	1	0	1	1
0	0	1	X	1	0	1	1
0	X	1	0	0	1	0	0
0	0	1	X	0	1	0	0
0	1	0	0	X	1	0	0

图 2.5 条件判别逻辑输入输出表

（4） 支持中断的微程序控制器设计

根据微程序每个节拍的功能，设置控制字段。当取指令完成后，应该将P_0置为 1。当指令执行完毕，需要将P_2置为 1。在地址 15 处，微程序需要根据 equal 值选择分支，故要将 P1 置为 1。执行 ERET 指令时，表示中断微程序处理完毕要中断返回，此时要开中断，STI 置为 1，同时需要将中断返回地址送入 PC 寄存器。需要将EPC_{out}和PC_{in}置为 1。在中断响应周期的第一个节拍，需要将保存断点地址和关中断，需要将PC_{out}、EPC_{in}和 CLI 置为 1。在中断响应周期的第二个节拍,需要将中断处理程序入口送入 PC,需要将$Addr_{out}$和PC_{in}置为 1。分析可得，能够得到如图 2.6 所示的表格。最后将二进制微指令存入微指令控制存储器就行。

微指令功能		PCout	DRout	Zout	Rout	IR(1)out	IR(0)out	IRout	DRout	PCin	ARin	DREin	DRin	Xin	Rin	IRin	PSWin	rs1/2	Add	Add4	Slt	READ	WRITE	EPCout	EPCin	Addrout	STI	CLI	P0	P1	P2	微指令	微指令十六进制
取指令	0	1									1			1																		10000000010010000000000000000	20120000
取指令	1																			1												00000000000000000010000000000	800
取指令	2			1						1		1										1										00100000101000000000100000000	8280200
取指令	3		1													1														1		01000000000001000000000000100	10008004
LW	4				1									1																		00010000000100000000000000000	4020000
LW	5					1													1													00001000000000001000000000000	2001000
LW	6			1							1																					00100000010000000000000000000	8100000
LW	7											1										1										00000000001000000000100000000	80200
LW	8		1												1																1	01000000000001000000000000001	10010001
SW	9				1									1																		00010000000100000000000000000	4020000
SW	10					1													1													00000100000000001000000000000	1001000
SW	11			1							1																					00100000010000000000000000000	8100000
SW	12				1								1					1														00010000000100001000000000000	4042000
SW	13							1															1								1	00000001000000000000010000001	400101
BEQ	14				1									1																		00010000000100000000000000000	4020000
BEQ	15				1												1	1												1	1	00010000000000011000000000011	4006003
BEQ	16	1												1																		10000000000010000000000000000	20020000
BEQ	17						1												1													00000010000000001000000000000	801000
BEQ	18			1						1																					1	00100000100000000000000000001	8200001
SLT	19				1									1																		00010000000100000000000000000	4020000
SLT	20				1													1			1											00010000000000001001000000000	4002400
SLT	21			1											1																1	00100000000001000000000000001	8010001
ADDI	22				1									1																		00010000000100000000000000000	4020000
ADDI	23					1													1													00001000000000001000000000000	2001000
ADDI	24			1											1																1	00100000000001000000000000001	8010001
ERET	25									1														1			1				1	00000001000000000000010010001	200091
中断响应	26	1																							1			1				10000000000000000000001001000	20000048
中断响应	27									1																1				1	00000001000000000000000100001	200021

图 2.6 微程序控制存储器数据

（5） 中断逻辑的设计

如图 2.7 所示，mEPCout 信号通过一个三态门控制 mEPC 寄存器中中断地址输出到内总线，开中断和关中断信号控制中断使能寄存器值与中断信号相与，输出中断请求。中断控制器根据不同的中断源选择不同的中断处理程序入口地址，在 Addrout 信号的控制下，输出到内总线。

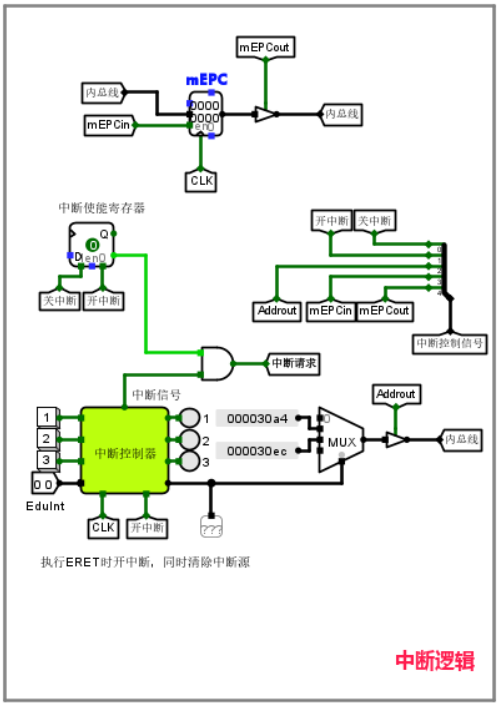


图 2.7 中断逻辑的设计图

2.4 故障与调试

2.4.1 接口处数据传输问题

故障现象：educoder 中断判别逻辑关卡一直报错。

原因分析：填写图 2.5 条件判别逻辑输入输出表时，存在歧义性，特别是 educoder 上当 P1 为 1 且 P0、P2 和 equal 均为 0 时，输出应为 100。

解决方案：在填写图 2.5 条件判别逻辑输入输出表时，适当的填写一些 0，解决二义性。

2.5 测试与分析

执行程序 sort-5-int-riscv.hex,没有按键行为，程序 在指令计数为 252 时停下，如图 2.8 所示：

图 2.8 支持中断的单总线 CPU 测试结果

查看内存内容，发现 0x80 地址开始的数降序排序，如图 2.9 所示：

40010113	fff00413	00000493	2084a023	00140413	00448493	2084a023	00140413	00448493	2084a023	00140413	00448493	2084a023	00140413	00448493	2084a023
00140413	00448493	2084a023	00140413	00448493	2084a023	00140413	00448493	2084a023	00000413	01c00493	20042983	2004aa03	0149a2b3	00028663	2134a023
21442023	ffc48493	00940463	fe0000e3	00440413	01c00493	00940463	fc0008e3	00000063	00810113	00812023	00912223	24000493	0004a403	00140413	0084a023
0084a223	0084a423	0084a623	0084a823	0084aa23	0084ac23	0084ae23	00412483	00012403	ff810113	00200073	00810113	00812023	00912223	28000493	0004a403
fff40413	0084a023	0084a223	0084a423	0084a623	0084a823	0084aa23	0084ac23	0084ae23	00412483	00012403	ff810113	00200073	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
00000006	00000005	00000004	00000003	00000002	00000001	00000000	ffffffff	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

图 2.9 内存数据

按下按键 1，程序将会在 0x90 开始的 8 个字单元全部加 1，如图 2.10

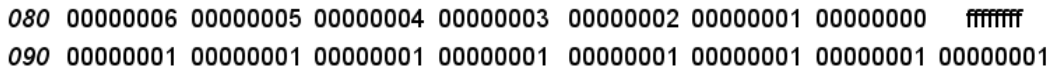


图 2.10 按下 1 次中断 1 后内存数据

再次按下按键 1，程序执行结果如图 2.11

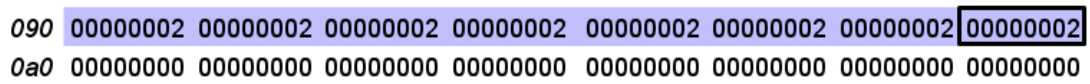


图 2.11 按下 2 次中断 1 后内存数据

按下按键 2，程序将在 0xa0 开始的 8 个子单元减 1，如图 2.12

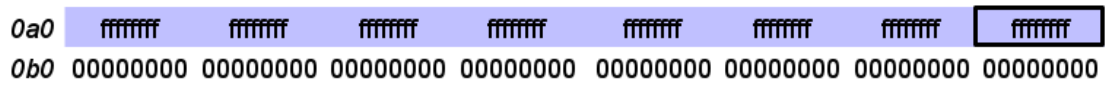


图 2.12 按下 1 次中断 2 后内存数据

再次按下按键 2，再次减 1，如图 2.13

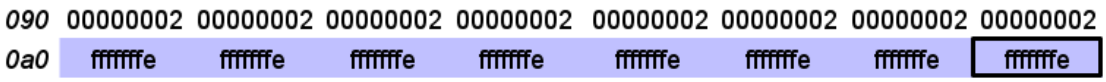


图 2.13 按下 2 次中断 2 后内存数据

3 总结与心得

3.1 实验总结

实验主要完成了如下几点工作：

1） 理解了变长指令周期三级时序 CPU 和支持中断的现代时序 CPU 的原理功能总结。

2） 设计了两种 CPU 的各部件以及数据通路组合。

3） 掌握了设计 CPU 的流程与思路。

4） 调试了设计的产品中出现的故障。

3.2 实验心得

1） 课程体会：计算机组成原理的实验给我带来了非常好，可以看出老师是在非常用心地设计实验。实验指导文档很详细，让我们非常清楚实验任务和目的，同时实验的设计都是组原理论课各章重要的知识点，很好的完成了理论与实际的相结合，同时为了学生的实验能够更好的进行，不拘泥于一些繁枝末节，更在乎主干知识和易错点的学习，老师设计了一些十分有用的工具（如 excel 文档），同时老师也设计了许多他功能部件，只留核心功能部件设计给学生，节省了我们许多时间。有的实验还设计了测试电路，方便我们找出错误。

2） 课程收获：：本学期组原实验真的让我学会许多东西，通过这学期的实验实践课，我对 logisim 的使用更加熟练。对 CPU 的内部结构有了更清楚的认识，此外，通过实验让我对理论课上模糊的概念理解的更加透彻。通过 CPU 的实验，让我明白了微指令和硬布线的区别。掌握了微程序和硬布线控制器的设计的方法，同时学会了中断的基本处理流程。本次实验课极大提高了我的动手能力和分析解决问题的能力。

3） 建议：中断实验缺乏视频解说，实验指导书不够详尽。

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：刘红阳 

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： _____