华中科技大学

2021

计算机组成原理 ·实验报告·

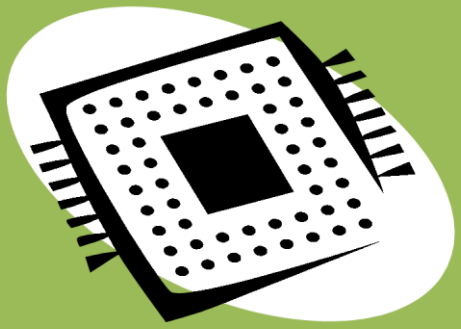
专业：计算机科学与技术

班级：CS1906

学号：U201915116

姓名：廖翔

电话：15027278369



邮　　件：liao1120x@gmail.com

完　成　日　　期：2021-12-14

计算机科学与技术学院

目　录

1 CPU 设计实验

1.1 设计要求

1.1.1 单总线 RISC-V CPU 设计(变长指令周期 3 级时序)

利用变长指令周期三级时序系统构造硬布线控制器，包括如下几项：

- RISC-V 指令译码器设计
- 时序发生器 FSM 设计
- 时序发生器输出函数设计
- 硬布线控制器组合逻辑单元
- 变长指令周期--硬布线控制器设计
- 变长指令周期—单总线 CPU 设计

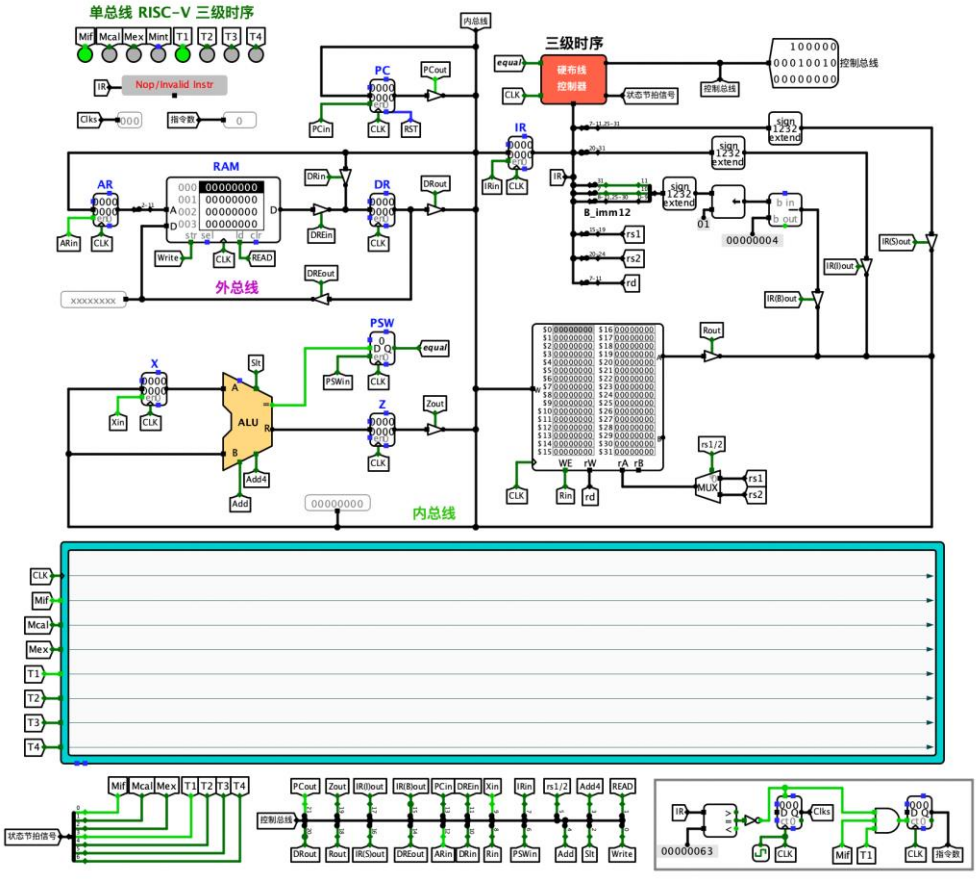


图 1-1 单总线 RISC-V 三级时序 CPU 结构图

1.1.2 RISC-V 现代时序中断机制实现

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，包括以下几项：

- RISC-V 指令译码器设计
- 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设计
- 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设计
- 支持中断的现代时序硬布线控制器设计

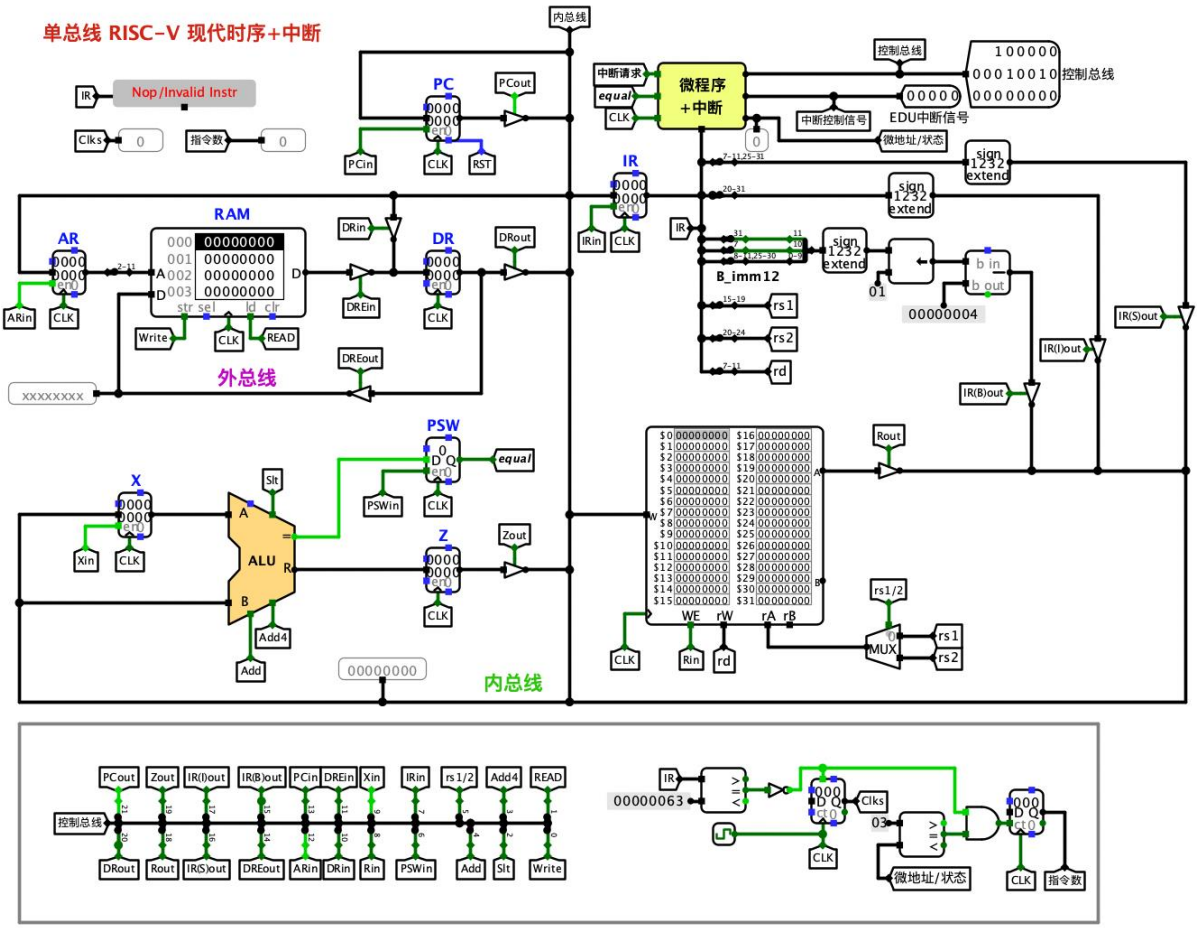


图 1-2 单总线 RISC-V 现代时序+中断 CPU 结构图

1.2 方案设计

1.2.1 单总线 RISC-V CPU 设计(变长指令周期 3 级时序)

- RISC-V 指令译码器设计

输入:32 位指令 IR

输出:指令译码信息。如 BEQ 代表 BEQ 指令。

实现方式:通过查阅具体指令，从取出的 FUNCT3 和 FUNCT7 中选择合适的进行比较，若匹配就说明是该指令。这里以 sw 指令为例，查阅 RISC-V 官方文档，可知 sw 指令如下：

SW

31-27	26-25	24-20	19-15	14-12	11-7	6-2	1-0
offset[11:5]		rs2	rs1	010	offset[4:0]	01000	11

图 1-3 RISC-V sw 指令

因此，将 OP 部分与 FUNCT_3 拼接为 OP_Funct3，然后前五位与图中 01000，即 8H，后三位与 010，即 2H 比较，若匹配，则说明是 SW 指令。

电路图:

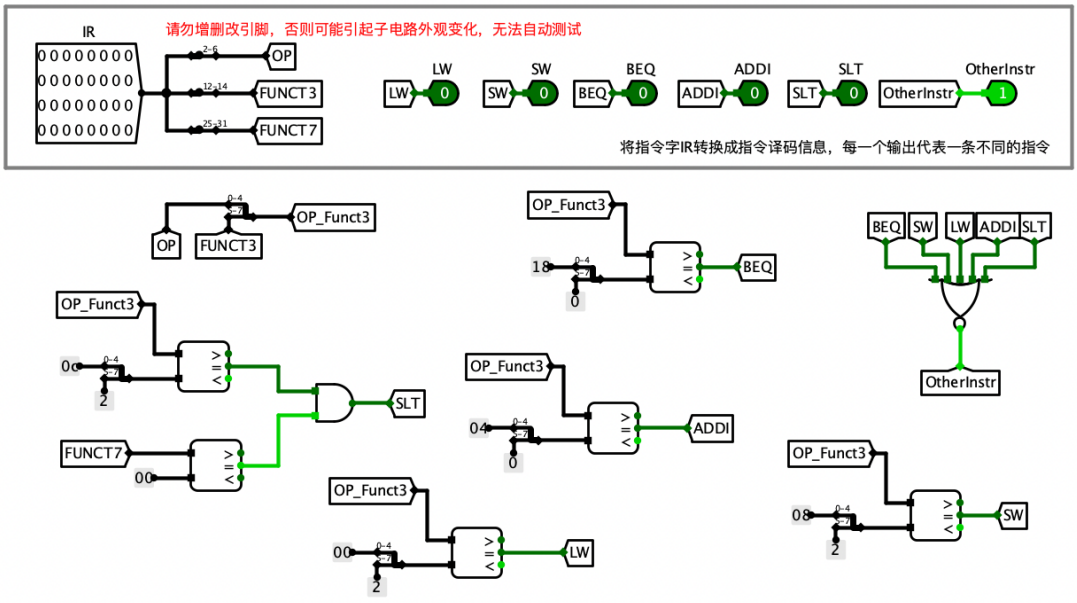


图 1-4 译码器电路图

- 时序发生器 FSM 设计

输入:当前状态

输出:次态

实现方式:根据课本上的三级时序状态转换图，填写 EXCEL 表格，再将最后结果复制进 logisim 中自动生成电路。具体状态图和 EXCEL 表格如下:

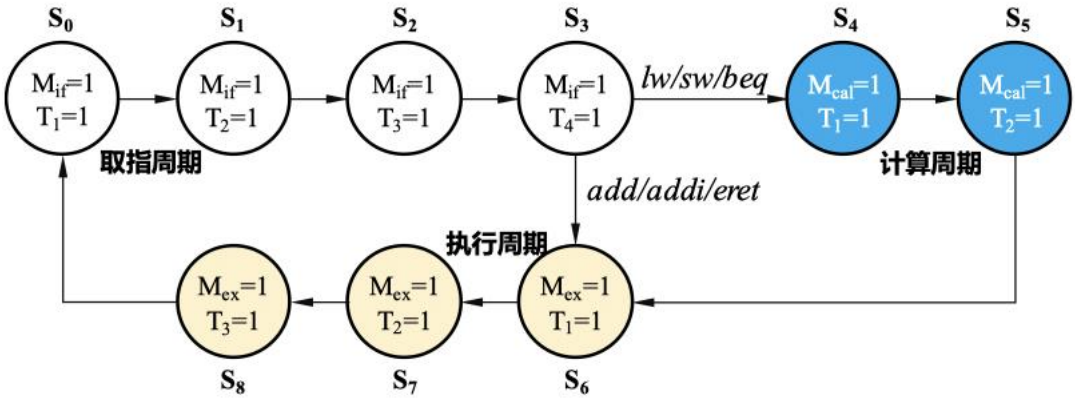


图 1-5 变长指令周期三级时序状态机

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	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
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	0	1	1	3	0	0	0	0	0			6	0	1	1	0

图 1-6 单总线 RISC-V 三级时序产生器组合逻辑自动生成表

- 时序发生器输出函数设计

输入:当前状态

输出:当前状态对应的机器周期和节拍电位

实现方式:同样的，根据课本上的时序状态转换表填写对应的 EXCEL 表，根据结果即可自动生成逻辑电路。具体状态表和 EXCEL 表格如下:

现态	输入/次态			周期、节拍电位输出
	任意指令	add/addi	lw/sw/beq	
S0 (0000)	S1 (0001)			M_{if}=T1=1
S1 (0001)	S2 (0010)			M_{if}=T2=1
S2 (0010)	S3 (0011)			M_{if}=T3=1
S3 (0011)		S6 (0110)	S4 (0100)	M_{if}=T4=1
S4 (0100)	S5 (0101)			M_{cal}=T1=1
S5 (0101)	S6 (0110)			M_{cal}=T2=1
S6 (0110)	S7 (0111)			M_{ex}=T1=1
S7 (0111)	S8 (1000)			M_{ex}=T2=1
S8 (1000)	S0 (0000)			M_{ex}=T3=1

图 1-6 时序发生器状态转换表

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 1-7 单总线 RISC-V 三级时序产生器输出函数自动生成表

- 硬布线控制器组合逻辑单元

输入:机器周期和节拍电位

输出:控制信号

实现方式:按照课本，根据每条指令所处的机器周期和对应的节拍电位，填写 EXCEL 表格中的控制信号输出，然后将结果在 logisim 中自动生成电路。EXCEL 表格如下:

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(1)out	IR(5)out	IR(8)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				1			1										1													1				
	1				1				1										1												1				
	1				1					1							1													1	1				

图 1-8 单总线 RISC-V 控制信号表

- 变长指令周期--硬布线控制器设计

输入:指令字,时钟信号,EQUAL 信号

输出:控制总线输出

实现逻辑:当前状态和指令译码输出作为状态机的输入，输出的次态作为下一次的输入，根据状态即可控制各个节拍信号的输出。

电路图:

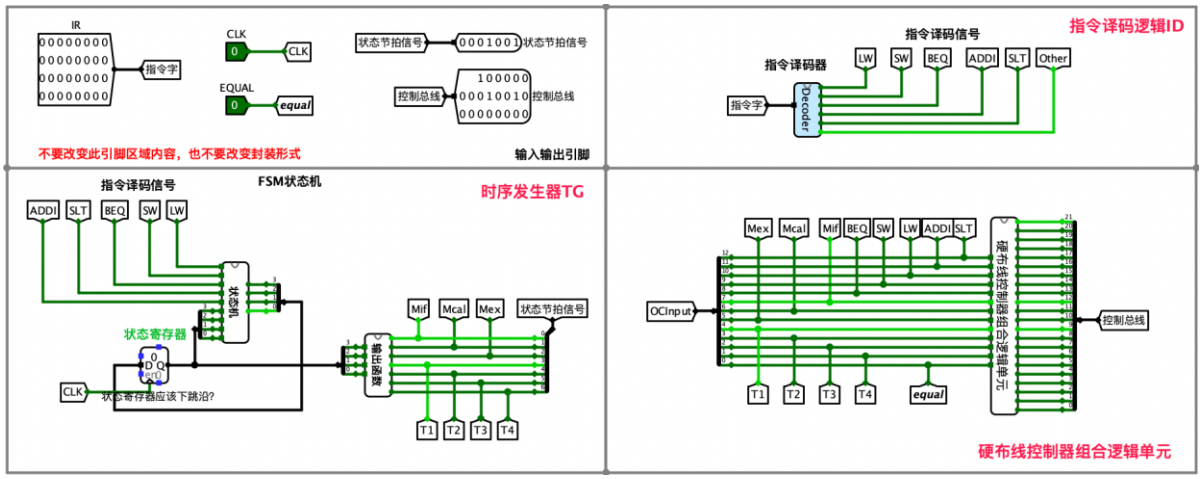


图 1-9 单总线 RISC-V 硬布线控制器电路图

- 变长指令周期—单总线 CPU 设计

这部分不需要我们做什么，只需要把 sort.hex 中内容导入 RAM 中，然后运行检查最后的结果即可，同时也能够更加直观地看到整个 RISC-V CPU 的组成和运行逻辑，图已在前面附上，这里不再贴。

1.2.2 RISC-V 现代时序中断机制实现

- RISC-V 指令译码器设计

这部分与 4.2.1 中的译码器设计相同。

- 支持中断的微程序入口查找逻辑

输入:机器指令译码信号

输出:微程序入口地址

实现方式:同样的，根据课本上各机器指令对应的入口填写 EXCEL 表格，将最后的结果到 logisim 生成电路即可。具体转换图和 EXCEL 表如下:

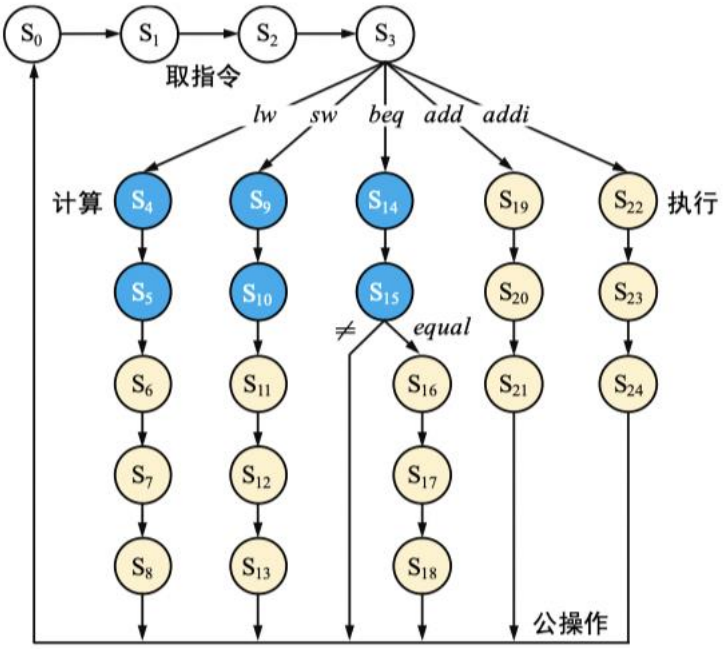


图 1-10 指令执行状态转换图

需要注意到这个转换图中没有加上中断入口。

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 1-11 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

输入:微程序流程控制信号,相等关系信号 EQUAL

输出:微程序地址转移控制信号

实现方式:首先要明确各个流程控制信号的含义。

译码测试位 P0:其值为 1 代表微指令后续地址应该根据指令译码情况选择指令对应的微程序入口地址，即微程序入口查找组合逻辑输出的地址。

判别测试位 P1:其值为 1 代表要根据 equal 标志进行微程序分支。

结束微指令判别测试位 P2:其值为 1 表示当前微指令是微程序的最后一条微指令，这一条微指令执行完毕表示当前机器指令执行结束。

如果 P2 为 1 切控制器收到中断请求，没有别的分支需要执行，就可以将判别测试位 P2 对应的微程序分支地址，也就是中断响应微程序的入口地址作为后续地址。

同时要理解微程序地址转移信号 S(S2S1S0)，S=0 代表是下一条地址，S=1 代表是取入口地址，S=2 代表取 BEQ 指令且当前运算是相等的跳转地址，S=3 代表是中断处理程序的第一个微程序地址，S=4 代表回到取指令的第一条微程序。

根据以上几点，再加上地址转移逻辑中的译码器设计，可以填写出 EXCEL 表格如下:

输入（填1或0，不填为无关项x）							
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 1-12 单总线结构现代时序(带中断)微程序条件判别逻辑自动生成表

- 支持中断的微程序控制器设计

输入:指令 IR，EQUAL 信号

输出:当前微程序地址和控制总线

实现方式:这个部分最重要也是最困难的地方就是解决如何根据微程序地址生成控制总线。首先，要根据微程序填写对应的控制信号输出表，这个部分在 RISC-V 版教材中就包含，可直接填写在 EXCEL 表中，下一步就是根据状态转换关系设计控制信号 P1P2P3，P1P2P3 的含义在之前已经介绍，那么设计的原则如下:

(1) 如果当前为取指令的最后一条微程序，则需要标记 P0 表示要根据入口查找逻辑得到入口地址

(2) 如果当前指令需要判断 EQUAL 分支，则需要标记 P1 表示需要跳转到处理 EQUAL 分支的微程序

(3) 如果为当前指令的最后一条微程序，则需要标记 P2 表示结束。

根据以上三条原则，可以填写出 EXCEL 表格如下:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AI	AJ
微指令功能		PCout	DRout	Zout	Rout					PCin	ARin	DREin	DRin	Xin	Rin	IRin		rs1/2	Add	Add4	Slt	READ	WRITE	EPCout	EPCin		STI	CLI	P0	P1	P2	微指令	微指令十六进制
取指令	0	1									1			1																		00000000010010000000000000000000	20120000
取指令	1																			1												00000000000000000001000000000000	800
取指令	2			1						1		1										1										00100000101000000000100000000000	8280200
取指令	3		1													1													1			01000000000000100000000000000100	10008004
LW	4				1									1																		00010000000010000000000000000000	4020000
LW	5					1													1													00001000000000001000000000000000	2001000
LW	6			1							1																					00100000010000000000000000000000	8100000
LW	7											1										1										00000000001000000000100000000000	80200
LW	8		1												1																1	01000000000001000000000000000001	10010001
SW	9				1									1																		00010000000010000000000000000000	4020000
SW	10						1												1													00000100000000000100000000000000	1001000
SW	11			1							1																					00100000010000000000000000000000	8100000
SW	12				1								1					1														00010000000100001000000000000000	4042000
SW	13							1																1							1	00000001000000000000001000000001	400101
BEQ	14				1									1																		00010000000010000000000000000000	4020000
BEQ	15				1												1	1												1	1	00010000000000011000000000000011	4006003
BEQ	16	1												1																		10000000000010000000000000000000	20020000
BEQ	17							1											1													00000010000000001000000000000000	801000
BEQ	18			1						1																					1	00100000100000000000000000000001	8200001
SLT	19				1									1																		00010000000010000000000000000000	4020000
SLT	20				1													1			1											00010000000000001001000000000000	4002400
SLT	21			1											1																1	00100000000001000000000000000001	8010001
ADDI	22				1									1																		00010000000010000000000000000000	4020000
ADDI	23					1													1													00001000000000001000000000000000	2001000
ADDI	24			1											1																1	00100000000010000000000000000001	8010001
ERET	25									1														1			1				1	00000000100000000000010010001	200091
中断响应1	26	1																							1			1				10000000000000000000001001000	20000048
中断响应2	27									1																1					1	00000000100000000000000100001	200021
																																	正确

图 1-13 单总线结构现代时序(带中断)微程序控制器微指令

得到控制存储器的内容后，下一步将各个入口地址连接即可得到最后的电路图如下:

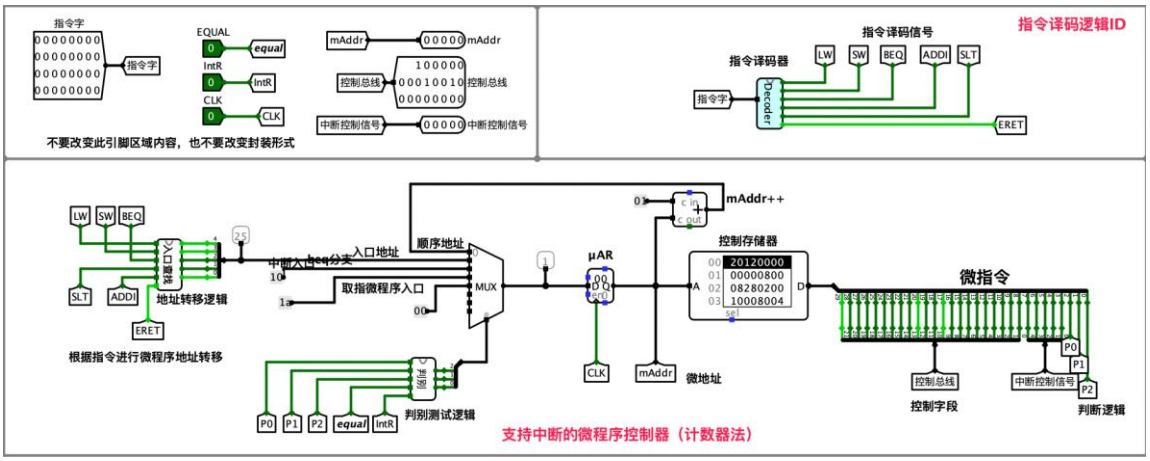


图 1-14 单总线结构现代时序(带中断)微程序控制器电路图

- 支持中断的微程序单总线 CPU 设计

这部分大体结构与变长指令周期单总线 CPU 相同，这里主要介绍中断逻辑的连接。

查找中断地址:根据中断控制器的中断类型输出进行分支判断，使用多路选择器。对于分支 1 则接上 1 号中断的中断入口地址的常量，对于分支 2 则接上 2 号中断的中断入口地址的常量。

IE 寄存器:开中断的时候异步置为 0，关中断的时候异步置为 1，输出为与寄存器保存内容相反的部分。

mEPC 寄存器:与其他锁存器是同一种构造，具体而言，在写入信号为 0 时，通过使能端控制寄存器忽略时钟信号，不寄存输入端内容，输出接一个三态门，只有在 EPC 输出信号有效的情况下才进行输出，不能进行输出的时候无法输出到内总线。

观察这个不带中断的程序，可以发现在 0x63 处停止，第 42 行是最后一条正常程序，因此有一个中断入口指令为第 41 条。同样的， 可以看到中断的第一条指令为 00810113，那么接着往下寻找该指令，可以看到第 61 行也有该指令，于是可以得到另一条中断入口指令为第 59 条。因为每一条指令由 4 个字节组成，所以第一个中断指令的地址为 0XA4,第二个中断指令的地址为 0XEC。

根据上述内容，可以连出中断逻辑如下:

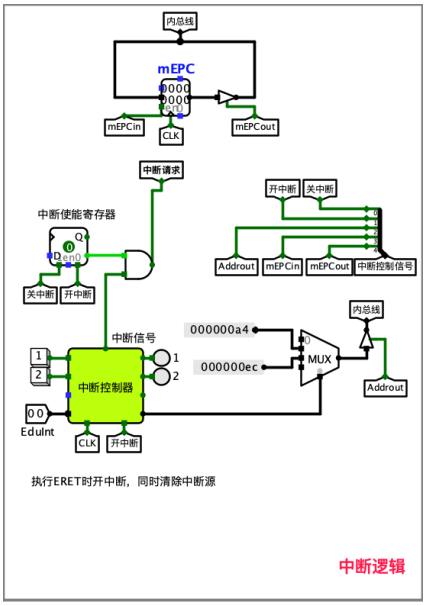


图 1-15 现代时序中断逻辑结构

- 支持中断的现代时序硬布线控制器状态机设计

输入:当前状态

输出:次态

实现方式:该部分需要在原来的现代时序硬布线控制器状态机上加上对中断的处理，需要注意到 EXCEL 表中 IR 一列需要专门拖出来。其实没什么需要太多注意的地方，就是在每一个有中断的地方跳转至中断执行入口，具体的 EXCEL 表如下:

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	ERET	IR	EQUAL	次态10进制	N4	N3	N2	N1	N0
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	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
1	1	0	0	1	25							0		0	0	0	0	0	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
0	1	0	0	0	8							1		26	1	1	0	1	0
0	1	1	0	1	13							1		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		26	1	1	0	1	0
1	0	1	0	1	21							1		26	1	1	0	1	0
1	1	0	0	0	24							1		26	1	1	0	1	0
1	1	0	0	1	25							1		26	1	1	0	1	0

图 1-16 支持中断的现代时序硬布线控制器状态机逻辑自动生成表

最后将结果导入 logisim 中自动生成电路即可。

- 支持中断的现代时序硬布线控制器设计

输入:指令字，EQUAL，时钟信号

输出:控制总线信号

实现方式:这个部分比较简单，直接将之前已经得到的微指令导入即可，主要思想与前面相同，这里不再赘述。电路图如下:

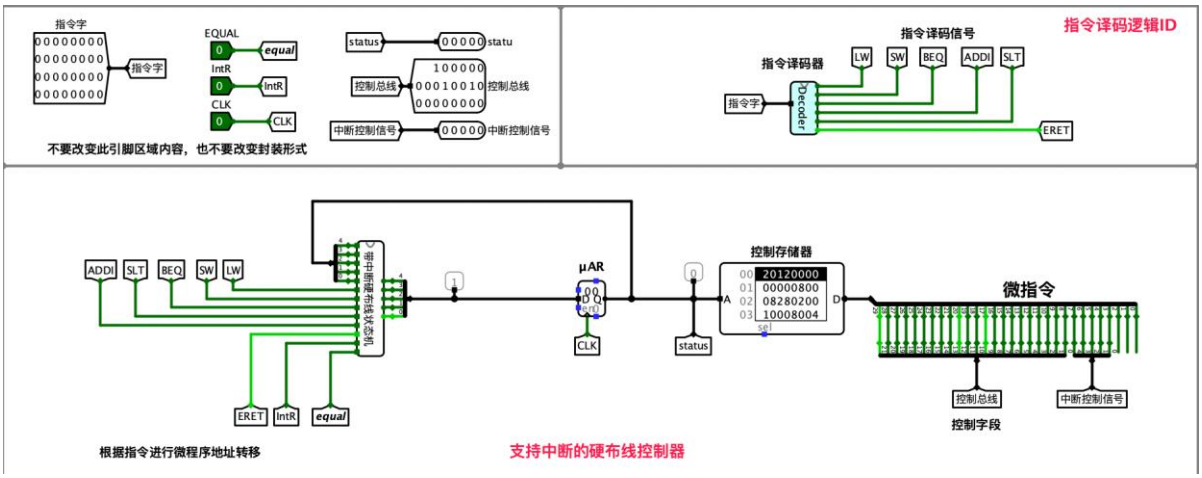


图 1-17 支持中断的现代时序硬布线控制器结构图

1.3 实验步骤

(1) 按照方案设计中的内容在 logisim 中连接电路图

(2) 提交 circ 文件到 Educoder 平台进行评测，修改

1.4 故障与调试

1.4.1 中断实验中无法触发中断

故障现象：中断始终无法触发

原因分析：IE 寄存器连线有误

解决方案：IE 寄存器应该连接触发器当前状态的非值

1.4.2 支持中断的现代时序硬布线状态机

故障现象：将表格结果导入 logisim 仍然有误

原因分析：EXCEL 表格中还有一栏，只是被隐藏较深

解决方案：EXCEL 表格需要把 IR 一栏拖出，之前忽略了

1.5 测试与分析

1.5.1 Educoder 平台测评

测试样例全部通过。

1.5.2 CPU 运行结果

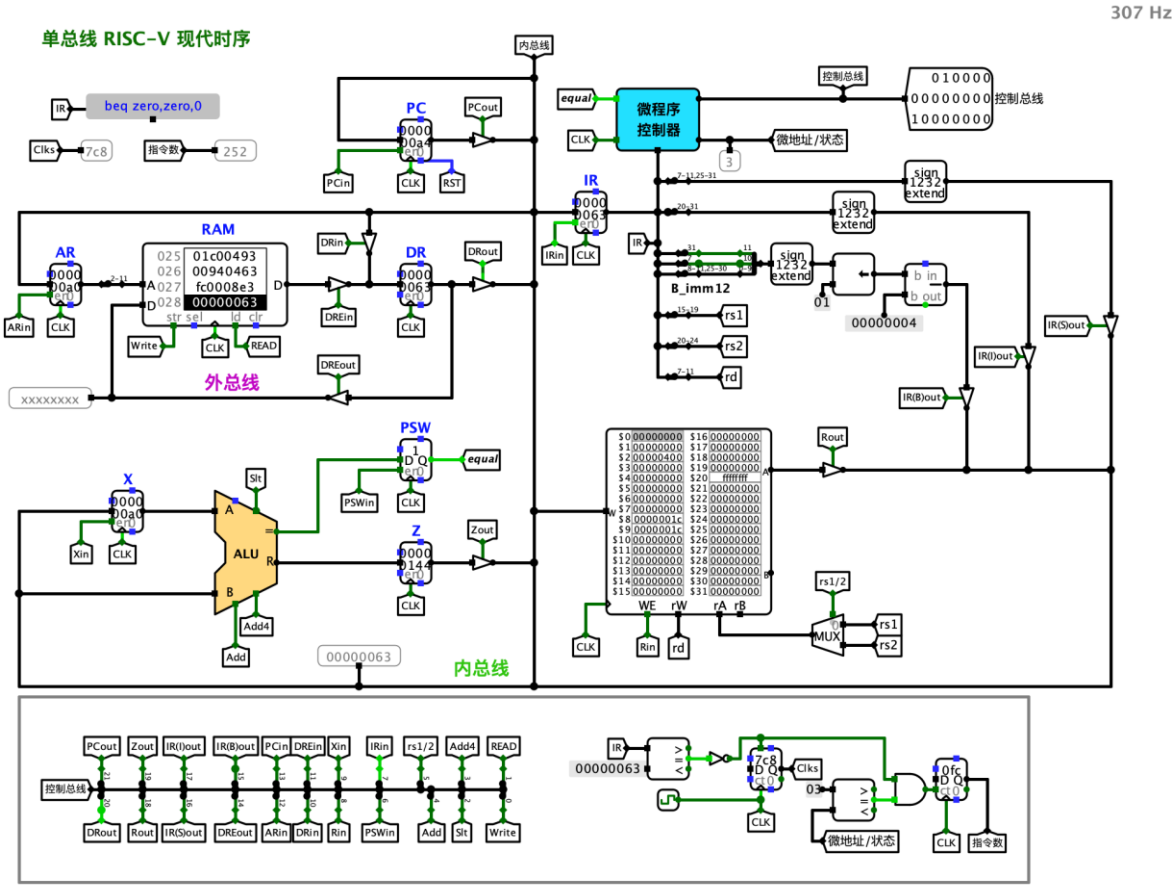


图 1-18　CPU 运行结果

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） 通关 logisim 的所有测试样例，通过动手实践的方式从数据表示，运算器，存储系统，CPU 几个维度来深刻认识到计算机的组成原理。

2） 实现完各个部件后，最后组装成 CPU，并能运行一个简单的冒泡排序算法。

2.2 实验心得

- 在数电实验的基础上再来进行组员实验，不需要再像数电实验是关心内部组件的实现逻辑，更重要的是能将各个组件组装起来，满足我们所需要的功能。
- 在理论上了解一些部件的原理后，通过实验的方式能够更加加深我们的印象，比如先行加法器，虽然书上的讲解已经非常详细，但是在 logisim 上将异或门，时钟信号等连接起来，将器件接口接出，让我们的理解更加深刻。
- 今年我们使用的是 RISC-V 版本的新 CPU 实验，虽然与 MIPS 区别不太大，但也能看到大萝卜老师在制作 EXCEL 表，电路的辛苦，建议就是可以在 circ 文件中把一些东西讲的更清晰点，然后及时地更新 Educoder 吧，其余的我觉得都做的很好。

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.

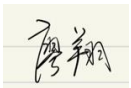
[4] 谭志虎,秦磊华,胡迪青.计算机组成原理实践教程.北京:清华大学出版社，2018 年.

· 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: 廖翔 嵌入签名图片

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: _____

· 指导教师评定意见·