

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： ACM1901

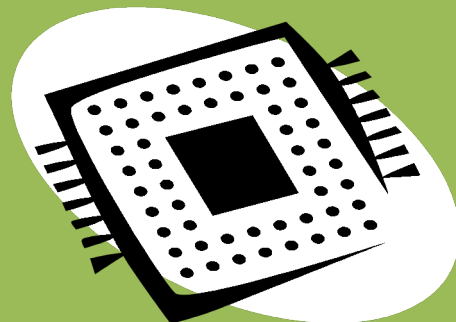
学 号： U201915035

姓 名： 邹雅

电 话： 15058667378

邮 件： 1542527211@qq.com

完成日期： 2021-12-16



计算机科学与技术学院

华中科技大学课程实验报告

目 录

1	CPU 设计实验	2
1.1	设计要求	2
1.2	方案设计	3
1.3	实验步骤	10
1.4	故障与调试	10
1.5	测试与分析	10
2	总结与心得	11
2.1	实验总结	11
2.2	实验心得	11
	参考文献	13

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)

1. RISC-V 指令译码器设计

根据输入的指令字 IR 翻译出 LW、SW、BEQ、ADDI、SLT 等五条指令信号并输出。其中 5 条指令对应指令字如下所示：

指令类型	指令	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
R	指令集	func7							rs2				rs1				funct3				rd				opcode				1	1					
R	RAV32A	add	0	0	0	0	0	0	rs2				rs1				0				rd				c				1	1					
R	RAV32A	sll	0	0	0	0	0	0	rs2				rs1				2				rd				c				1	1					
B	RAV32A	beq	11	imm[9:4]							rs2				rs1				funct3				imm[3:0]				10	opcode				1	1		
B	RAV32A	bneq	11	imm[9:4]							rs2				rs1				0				imm[3:0]				10	0x18				1	1		
I	RAV32A	addi	imm12												rs1				funct3				rd				opcode				1	1			
I	RAV32A	slli	imm12												rs1				0				rd				4				1	1			
I	RAV32A	slli	imm12							rs1				2				rd				0				1	1								
S	RAV32A	slli	imm[11:5]							rs2				rs1				funct3				imm[4:0]				opcode				1	1				
S	RAV32A	sllw	imm[11:5]							rs2				rs1				2				imm[4:0]				0x08				1	1				

图 1.2.1 指令字

那么 op、funct3、funct7 字段即可区别出各条指令并输出。绘制出电路图如下所示

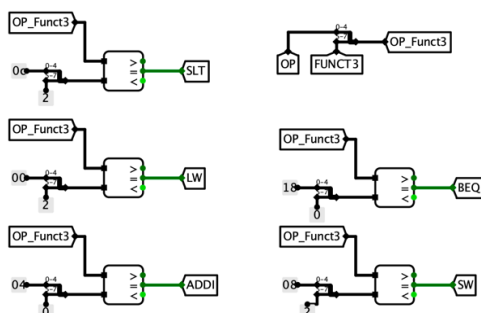


图 1.2.2 指令译码电路逻辑

2. 变长指令周期时序发生器设计

单总线结构采用变长指令周期，不同指令机器周期数不同，每个机器周期的节拍数也是可变的，其中具体状态图如下所示：

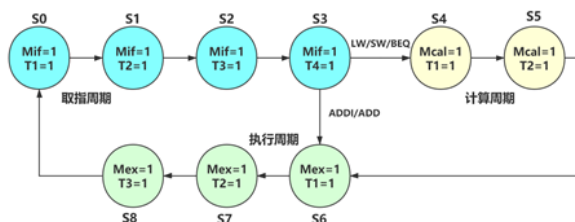


图 1.2.3 变长指令周期时序状态图

华中科技大学课程实验报告

其中我们可以看到最复杂的部分是 S3 状态的转移，在 LW/SW/BEQ 存在信号为 1 时转移到 S4 状态，在 ADDI/SLT 存在指令为 1 时转移到 S6 状态。由此得表如下所示：

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	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	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

图 1.2.4 时序状态转换表格

由此表格自动生成的逻辑表达式输入 logisim 自动生成电路即可。

3. 时序发生器输出函数设计

输出函数是指每个时序状态对应的机器周期和节拍电位的输出。取指周期 Mif 需要 4 个节拍电位，分别对应 S0、S1、S2 和 S3.计算周期 Mcal 需要 2 个节拍电位，LW/SW/BEQ 指令才有计算周期，分别对应 S4 和 S5.执行周期 Mex 需要 3 个节拍电位，分别对应 S6、S7 和 S8.由此我们可以得出下面输出函数的表格：

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 1.2.5 输出函数表格

4. 硬布线控制器组合逻辑单元

对于每一个微命令而言，都是机器周期、节拍电位以及指令信号的组合逻辑，我们依据 5 条 RISC-V 指令的意义以及其对应的微命令可以得到表格。

取指令周期

节拍	控制信号(4 cycles)	
T1	S_0	PC _{out} , AR _{low} , X _{in}
T2	S_1	+4
T3	S_2	Z _{out} , PC _{inv} , DRE _{in} , Read
T4	S_3	DR _{out} , IR _{in}

计算周期

节拍	lw (5 cycles)	sw (5 cycles)	beq (5 cycles)	add (3 cycles)	addi (3 cycles)
T5	R _{out} , X _{in}	R _{out} , X _{in}	R _{out} , X _{in}		
T6	IR(I) _{out} , ADD	IR(I) _{out} , ADD	R _{out} , Rs/Rt, SUB, PSW _{in}		
T7	Z _{out} , AR _{in}	Z _{out} , AR _{in}	PC _{out} , X _{in}	R _{out} , X _{in}	R _{out} , X _{in}
T8	DRE _{in} , Read	R _{out} , Rs/Rt, DR _{in}	IR(A) _{out} , ADD	Rs/Rt, R _{out} , ADD	IR(I) _{out} , ADD
T9	DR _{out} , R _{in}	DRE _{out} , Write	Z _{out} , C _{in} = PSW _{equal}	Z _{out} , R _{out} , RegDst	Z _{out} , R _{in} , RegDst

执行周期

图 1.2.6 指令对应的微命令

[illegible]

图 1.2.7 微命令组合逻辑真值表

5. 硬布线控制器设计

硬布线控制器设计中需要将指令译码信号、状态机、状态寄存器以及输出函数模块连接在一起。在本小节实验中，我们需要完善时序发生器模块。

每次读取状态时从状态寄存器中取出现态输入状态机，同时将指令译码信号也输入状态机，从状态机输出次态返回输入状态寄存器在下一个时钟输入。从状态寄存器取出的现态也同时传递给输出函数模块，由输出函数模块得到状态节拍信号，也即机器周期和节拍电位信号。由上述思路构建时序发生器模块如下所示：

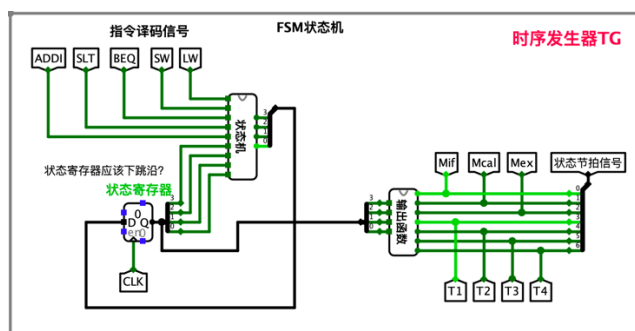


图 1.2.8 时序发生器模块

6. 单总线 CPU 设计

本小节是一个综合设计的实验。基于以上所有实验的基础上，调试使得最终的单总线 CPU 能运行 sort-5-riscv 程序。在 RAM 中加载该程序（输入十六进制的程序），并通过 ctrl+k 时钟连续进行运行。运行时可以看到下方时钟信号 CLK 和时序信息在数字示波器上的显示跳变，并从中体会 CPU 运行是受时钟同步控制的，理解时序的变化。

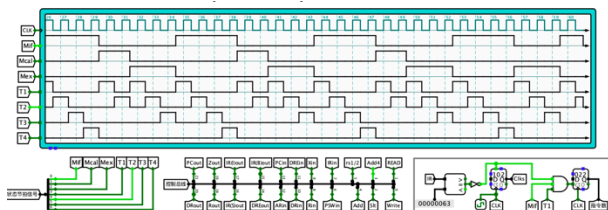


图 1.2.9 单总线 CPU 运行时序输出

1.2.2 RISC-V 现代时序中断机制实现

1. RISC-V 指令译码器设计

同上 1.2.1.1，因为指令字以及对应翻译指令译码信号是一致的。

2. 支持中断的微程序入口查找逻辑

微程序入口查找逻辑是指根据指令译码信号，输出微程序对应的入口地址。在支持中断的微程序控制器中，多支持了一个指令译码信号是 ERET，对应的地址是 25。其他地址转移逻辑都如下所示：

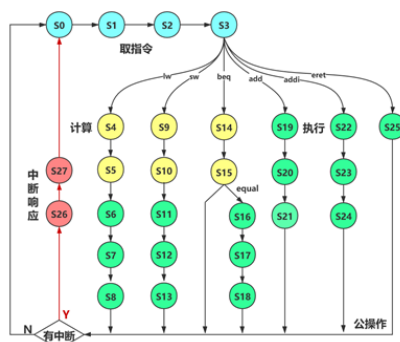


图 1.2.10 微程序地址转移/状态转移逻辑

根据该图填微程序入口地址表如下：

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 1.2.11 微程序入口地址表

3. 支持中断的微程序条件判别测试逻辑

在该条件判别测试逻辑中，有 3 个判别位 P0(PIR)、P1(Pequal)、P2(Pend)和 2 个信号 equal 和 IntR。P0 是在取指周期最后一条指令中指示微程序地址为依照指令译码信号生成的入口逻辑地址，此时对应的是输出 s 为 1。P1 是指在 equal 信号为 1 时，两个输入寄存器中值相等需要进行跳转，那么将跳转到的是 beq 分支地址，此时对应的是输出 s 为 2。P2 表示当前微程序运行到指令的最后一行，需要去判断是否存在中断，如果当前同时 IntR 信号也为 1，那么就微指令地址就应该为中断响应的微程序地址，此时对应的是输出 s 为 3；而如果此时 IntR 信号为 0，也就是当前没有中断时，则直接重新开始取指令，对应微程序地址为取指微程序的地址，此时对应的是输出 s 为 4。

由上述逻辑，可以得到支持中断的微程序条件判别测试逻辑，对应表格如下所示：

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
1	0	0				0	1
0	1	0	0		1	0	0
0	1	0	1			1	0
		1	0	1		1	1
	0	1		0	1		
	1	1	1	0		1	
	1	1	0	0	1		
	1	1	1	1		1	
	0	1	1	1		1	1

图 1.2.12 微程序条件判别测试逻辑

4. 支持中断的微程序控制器设计

在该图中，微地址转移逻辑部分如上一小节所述，可以构建输入的微地址。得到下图的电路图连接：

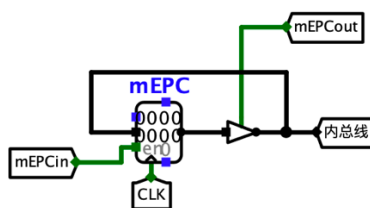


图 1.2.15 EPC 寄存器电路

中断请求信号是中断使能寄存器的输出和中断控制器的中断信号输出相与，也就是中断信号要在开中断的情况下才会起作用。此外，我们还需要送出中断服务程序的地址，这是中断控制器的地址输出，由 AddrOut 信号控制的。具体电路图如下所示：

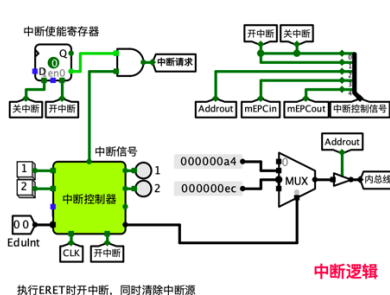


图 1.2.16 中断逻辑电路图

最后在 RAM 中加载对应的排序程序，即可得到结果。

6. 支持中断的现代时序硬布线控制器状态机设计

硬布线控制器状态转移逻辑如图 1.2.10 所示，由该图可以得到次态对应现态和指令译码信号之间的关系。在现代时序的基础之上，增加了中断的逻辑。比如说在状态 3，信号 ERET 为 1 时需要转移到状态 25。在此省略表格的截图。

7. 支持中断的现代时序硬布线控制器设计

将指令译码信号和状态连接到带中断的硬布线状态机上，对应输出连接微地址寄存器即可完成本小节实验，对应电路图如下所示：

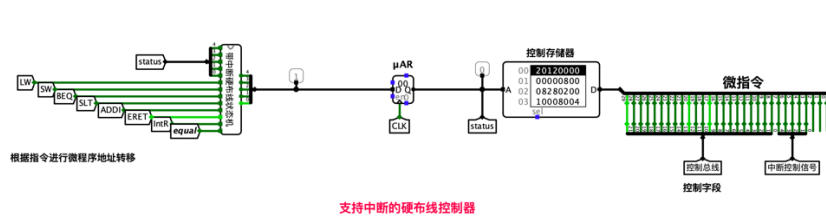


图 1.2.17 硬布线控制器电路图

1.3 实验步骤

- (1) 观看 educoder 上首页的实验相关知识视频以及其中的实验指导视频。
- (2) 在 logisim 中打开实验文件完成对应电路图的逻辑实现。
- (3) 按照 educoder 上的实验要求进行提交或运行冒泡排序程序进行检验。

1.4 故障与调试

1.4.1 微程序跳转不对

故障现象:在带中断的 RSIC-V 现代时序 CPU 的微程序控制器设计中,当执行 BEQ 分支跳转指令时,本应跳转至微地址为 16 的微程序进行执行,但是跳转至了 26 对应的中断响应微程序。

原因分析:可以看到其他的微程序都能自动正常跳转,只是在这个地方输入的微程序地址有问题,检查对应微程序判别测试逻辑表格。发现我把当 P2 和 IntR 同时为 1 时的所有情况都让其输出 3。但其实是不对的。因为当 P1 和 equal 同时为 1 时,表示程序需要执行 BEQ 的分支跳转,这个时候这整个指令还没有完成,不能执行中断中断响应。

解决方案:重新填入该微程序判别测试逻辑表格,当 P1、equal、P2 和 IntR 同时为 1 时,应该输出为 2。而其他的 P2、IntR 同时为 1 的情况下,应该都跳转到 3。再重新填入判别测试逻辑的组合逻辑表达式,即可解决本问题。

1.5 测试与分析

1.5.1 Educoder 测评

全部测试样例通过。

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) 实现单总线变长指令周期 3 级时序 RISC-V CPU 设计，以及实现 RISC-V 现代时序中断机制。
- 2) 进一步理解了 RISC-V CPU 微程序控制器以及硬布线控制器的原理。
- 3) 深入理解了微程序的指令，理解取指周期对应的微指令以及 5 条经典 RISC-V 指令分别对应计算周期、执行周期的微指令，并完成了微程序表格。
- 4) 对于变长指令周期 3 级时序 CPU，先分别完成指令译码器、时序发生器的状态状态机、时序发生器的输出函数、保存有微程序的控制存储器，再将这些器件组合成硬布线控制器。
- 5) 对于带中断的现代时序 CPU，在现代时序 CPU 的基础上，适当增加指令和器件并更改逻辑，即可完成。

2.2 实验心得

- 1) 在做本实验的过程中，加深了对单总线 CPU 设计的理解。在课上学习理论性知识的时候，对知识也都理解了，但是没有形成结构性的认识，各个内容知识点对我来说是分散的。但是在做实验的过程中，我清晰地把这些知识串联在了一起。比如说对于微程序控制器，需要先用指令字产生指令译码信号，利用判别控制模块的输出控制选择输入微程序的地址，而各个指令对应的入口地址由入口查找逻辑生成，在控制存储器中填入微程序后即可得到完整的微程序控制器。这整个逻辑都在我的脑海中形成了知识体系。
- 2) 在完成带中断的 CPU 实验设计后，我彻底知道了中断的实现方式，因为之前别的课程都是在谈中断的作用，给人一种系统不知道怎么做就自动完成了的感觉。但是自己亲自动手设计了 CPU 中的中断模块后，就能明白了。
- 3) 在实验的过程中遇到了不少的问题，不过在组内实践群里面的大萝卜老师和群里面很多同学都在热情地回答我们的问题，而且我们想要去提的问题可能

华中科技大学课程实验报告

之前有些同学也刚好遇到过，在组原群里就能学到许多，确实组原群也成了这门课程的一个特色。

- 4) 组成原理的实验是我在大学中除了 CSAPP 的实验以外，做过觉得最好的实验了。其他课程的实验不是年代流传久远，就是和课程、考试联系不大，做了也没有很大的意义。但是组原实验在我做完了之后，是能实实在在感觉学到了东西的。我认为只有这种实验才能真正吸引学生做下去，而不是靠所谓的防止抄袭检测，需要依靠内容来吸引我们去学。希望组成原理这门课越做越好，让大家都能真正理解组成原理。

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：邹雅 

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 •
