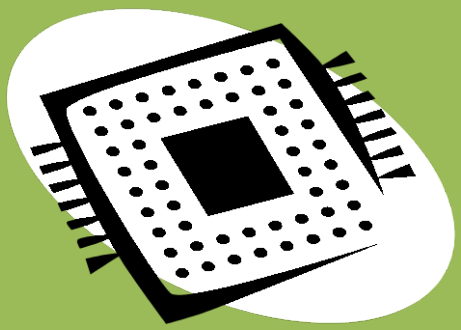
华中科技大学

2021

计算机组成原理 ·实验报告·

专　　业：	计算机科学与技术
班　　级：	CSIE1902
学　　号：	U201912633
姓　　名：	张睿
电　　话：	13544024941
邮　　件：	rui.zhangchn@qq.com
完成日期：	2021-12-15



计算机科学与技术学院

目　录

1 变长指令周期 CPU 设计实验

1.1 设计要求

本次实验需要设计变长周期指令的 CPU，具体而言，可拆分为如下设计目标：

1. 利用比较器等功能模块将 32 位 RISC-V 指令字译码生成 LW、SW、BEQ、SLT、ADDI、OtherInstr 等指令译码信号。

2. 利用数字逻辑电路相关知识设计变长指令周期的三级时序系统，时序发生器包括状态机和输出函数两部分，要求设计状态机。

3. 利用数字逻辑电路相关知识设计变长指令周期的三级时序系统，时序发生器包括状态机和输出函数两部分，上一关已经实现了 FSM 状态机逻辑，本实验要求设计实现输出函数组合逻辑。

4. 在实现了指令译码逻辑、时序发生器主要功能部件后，进一步设计实现控制器核心模块硬布线控制器组合逻辑单元。

5. 在实现指令译码、时序发生器关键功能部件、硬布线控制器等功能模块后，最终实现硬布线控制器的集成。

1.2 方案设计与实验步骤

1.2.1 指令译码设计

这一器件主要事通过对指令进行译码，得出对应的指令，起到一个“翻译”的作用，既然事翻译，就有“源”和“目的”，“源”，即为 riscv 指令机器码，“目的”即为对应指令。只需要通过 riscv 码表找对应指令的机器码，进行一一对应即可，可画出图 1

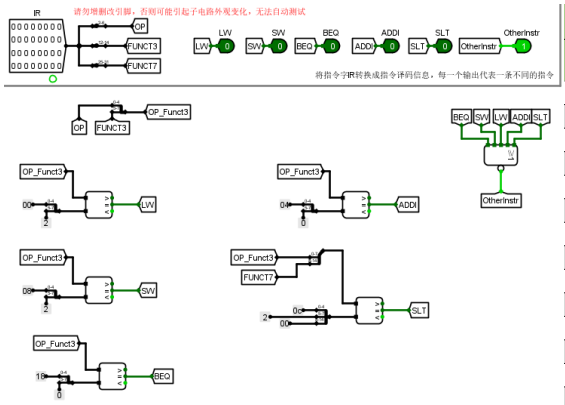


图 1.1 指令译码电路

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	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					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 1.3 状态转换表

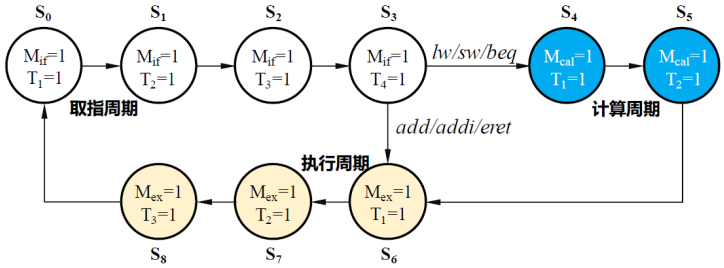


图 1.2 状态转换图

1.2.2 时序发生器状态机设计和输出函数组合逻辑设计

时序发生器状态机主要是根据输入进行状态转化，注意这个状态机是采用纯组合逻辑实现的，根据图 1.2 可以看到一共由 9 个状态，没有对应某个状态的周期的某个节拍信号，由此很快得出状态转换表图 1.3，并利用 excel 公式自动生成相应的逻辑表达式。

输出函数组合逻辑主要是利用当前状态生成当前的状态周期和节拍信号，非常简单，不多赘述。

1.2.3 硬布线控制器组合逻辑单元设计

这里是一个难点，主要难在要分析每个指令在各个状态周期中各个节拍的运行过程，即指令的数据通路，才能成功分析出对应控制信号，但是由于这里书上基本都由现成的，所以大大降低了难度，只用稍微分析的一下 slt 指令。

对于 slt 指令，其为一条 R 型指令，功能为 If (rs1 < rs2) R[rd] ← 1 else R[rd] ← 0，由于控制指令有 slt 指令，所以数据通路跟 add 指令一样，唯一不同的是把 add 信

号变为 slt 信号。由此可得 slt 指令得控制信号。

Mif	Mcal	Mex	Mint	T1	T2	T3	T4	LW	SW	BEQ	SLT	ADDI	EQUAL	PCout	DRout	Zout	Rout	IR(0)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								

图 1.4 slt 指令的控制信号

1.2.4 硬布线控制器的集成

这一部分主要是把之前做的模块全部组合起来，难点在于对控制器模块即工作流程的理解，对于三级周期变长指令的控制器，首先其需要通过状态机生成当前状态，将当前状态送入状态寄存器和输出函数组合电路生成状态周期信号和节拍信号，再根据状态周期和节拍信号，并分析指令的数据通路，最终得出各个状态各个节拍的控制信号。对于状态寄存器，其主要存储上一状态，使得本来应为时序逻辑的状态机变为组合逻辑，见图 1.5。最后将该控制器放入 cpu 通路中，即可实现一个简易变长周期的 cpu。

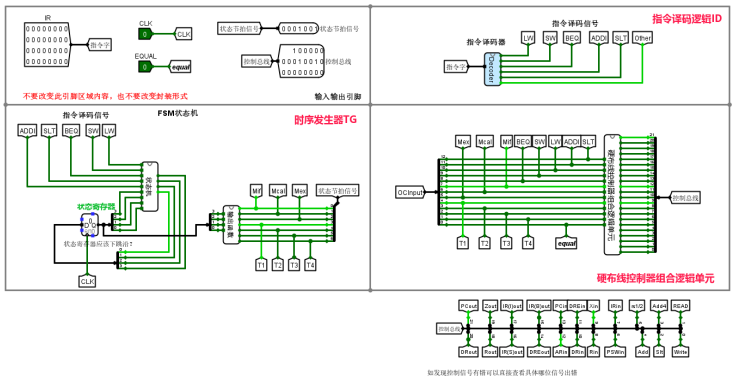


图 1.5 控制器集成

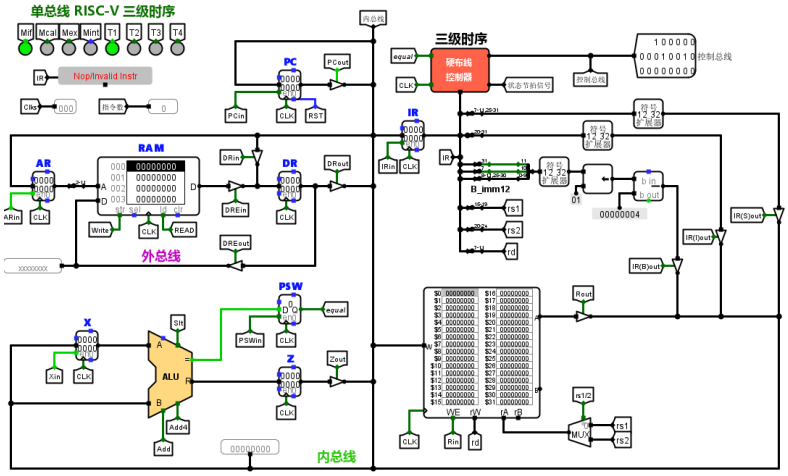


图 1.6 CPU 通路图

1.3 故障与调试

1.3.1 状态机无输入问题

故障现象：对于状态机无输入的情形，无法正确跳转对应状态。

原因分析：如图 1.3，可以看到在取指最后一个节拍，有一种可能是没有任何输入。

解决方案：在取指最后一个节拍时，在所有输入为 0 的情形下，跳转至执行状态。

1.3.2 状态寄存器上下沿问题

故障现象：输出重复。

原因分析： 由于状态寄存器的时钟和 cpu 其他部件的时钟相同，导致状态改变的时候其余部件也同时改变，由于延迟问题，此时其余部件并未得到当前的控制信号而是上一指令的控制信号。

解决方案：将状态寄存器和其它部件时钟的沿向错开。

1.4 测试与分析

图 1.7 测试结果

由图 1.7 可以看到，cpu 成功的运行了冒泡排序，并输出了正确结果。

2 CPU 中断设计实验

2.1 设计要求

本次实验需要设计现代时序带中断的 CPU，分为微程序控制和硬布线控制，具体而言，可拆分为如下设计目标：

1. 利用比较器等功能模块将 32 位 RiscV 指令字译码生成 LW、SW、BEQ、SLT、ADDI、OtherInstr 等指令译码信号

2. 了解微程序控制器中微程序分支的基本原理，为已经实现的微程序入口查找逻辑增加 eret 对应微程序入口查找逻辑，要求重新设计微程序入口查找逻辑。

3. 了解微程序控制器中微程序分支的基本原理，要求能设计判别测试逻辑，注意这里判别测试逻辑增加了 P2 位，P2=1 表示当前微指令为微程序最后一条微指令，需要进行中断判断，如果存在中断请求，需要转中断响应微程序执行。

4. 完善微程序框架，将微程序入口查找逻辑,判别测试逻辑，控制存储器等部件进行适当连接，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。

5. 完成前面任务以后，还需要在单总线数据通路中增加与中断相关的硬件模块，主要包括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，需要在主电路中将这些模块进行有效连接，并进行最终的联调，测试 CPU 是否能正常响应 2 个按键对应的中断服务程序。（注意不同按键的中断服务程序入口地址可以利用 RARS 汇编器汇编源程序查看 lable 地址得到）

6. 理解现代时序系统中硬布线控制器中中断机制的实现原理，设计支持中断处理的硬布线控制器核心部件状态机模块。

7. 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，在下图中完成硬布线控制器框架连接，注意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 cpu 中的微程序控制器进行程序测试。

2.2 方案设计与实验步骤

2.2.1 指令译码设计

这个部分和前面一样，只需要再加上中断功能的指令即可，不做过多叙述。

2.2.2 微程序入口查找逻辑

这一部分和之前的硬布线不同，由于是采用微程序，所以需要找到微程序的入口地址，这里为了方便，采用 10 进制表示，即每一个指令都需要对应一个微程序的入口地址，然后采用 excel 逻辑自动生成，即可用组合逻辑实现，可得图 2.1.

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 2.1 微程序入口地址

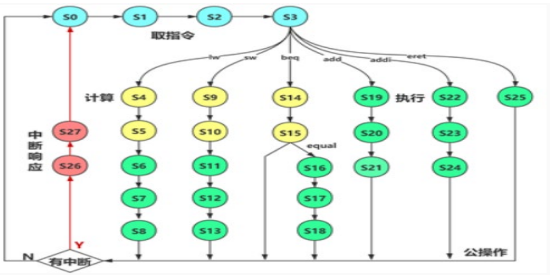


图 2.2 转移逻辑图

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

图 2.3 条件判别测试逻辑

可以看到，入口地址被翻译成 5 位状态码，利用 5 位状态码，即可完成入口地址的表示，之所以是 5 位状态码，是因为一共有 28 个状态需要表示，可见图 2.2。

2.2.3 支持中断的微程序条件判别测试逻辑

这是一个难点，花了比较久的时间，主要难点是要理解中断的响应时机及其对应的测试位以及 beq 指令的两种分支。由图 2.2，可以看到，中断响应时机是每条指令执行完的时候，为此，我们应对每条微指令增加一个测试位 Pend，来表示当前指令是不是最后一条指令，如果当前指令为最后一条指令，需要把 Pend 置为 1，此时，需要判

断是否由中断到来，如果中断到来则应该对其进行响应，如果没有，则转换到取指微程序入口。

与此同时，对于 beq 指令，其也比较麻烦，beq 指令有两个分支，再加上中断是否到来，所以会产生 4 种情况。

- P0 为 1 则无条件跳转至当前指令的入口地址
- P0 为 0
  - P1 为 0，equal 为无关项
    - P2 为 0，IntR 为无关项，此时对应于中间的微指令，直接跳转至下一地址（计数器法则是地址+1）
    - P2 为 1，需要看 IntR
      - IntR 为 1，此时对应于最后一条微指令且有中断到来，应跳转至中断处理程序
      - IntR 为 0，此时对应于最后一条微指令且没有中断，应跳转至取指微程序。
  - P1 为 1，需要讨论 equal。
    - P2 为 0，不需要讨论 IntR
      - Equal 为 0，对应于 beq 指令的错误判断，此时应该跳转至取指微程序（这种情况应该不可能出现，因为 beq 判断时，P2 也会置为 1，因为其有可能是最后一条指令）
      - Equal 为 1，对应于 beq 指令的正确判断，此时应跳转至 beq 微程序的下一条微指令。
    - P2 为 1，需要讨论 IntR，此时会出现 4 种情况
      - Equal&IntR，这种情况应该不可能出现，因为当 equal 成立时，此时不会是最后一条指令。
      - Equal&~IntR，同上
      - ~Equal&IntR，此时对应于 beq 指令错误判断，且有中断到来，此时应该跳转至中断处理程序
      - ~Equal&IntR，此时对应于 beq 指令错误判断，且无中断到来，此时应该跳转至取指微程序。

综上，我们可以获得图 2.3。

2.2.4 支持中断的微程序控制器设计

这一部分有两个任务，一是把之前做的模块统一起来，主要是要理解微程序控制的处理流程，其次还要填写微程序的控制逻辑。

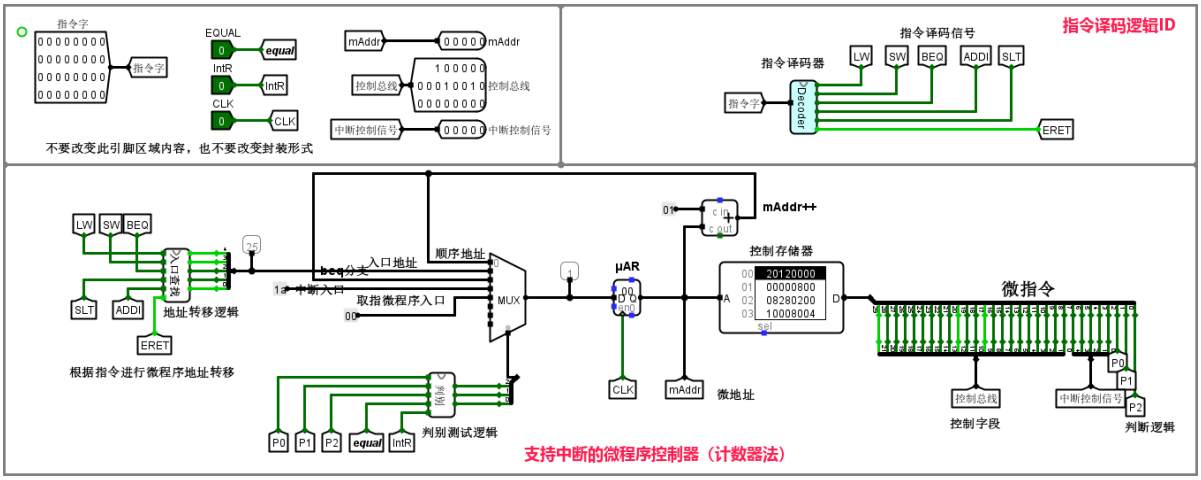


图 2.4 微程序控制器集成

主要流程为，利用多路选择器选择下一条微指令的入口地址，利用判别逻辑对其进行选择，可见图 2.4。

另一个部分就是控制信号的生成，这一部分和前面类似，只需要加上判别位的信息即可，不多赘述。

2.2.5 支持中断的微程序单总线 CPU 设计

这一部分主要是对中断逻辑的通路图实现，这里主要有两个部分，一个是 mEPC 的使用和中断入口的查找。

对于 mEPC 的使用，其主要是是保存当前断点的地址即 PC 值，只需要将内总线连入即可，加上控制信号作为使能信号。

对于中断入口的查找，其主要是根据程序文件找到中断入口地址，由于有两个中断，需要一个多路选择器，选择对应中断程序的入口地址并将其输出到内总线上，可得图 2.5.

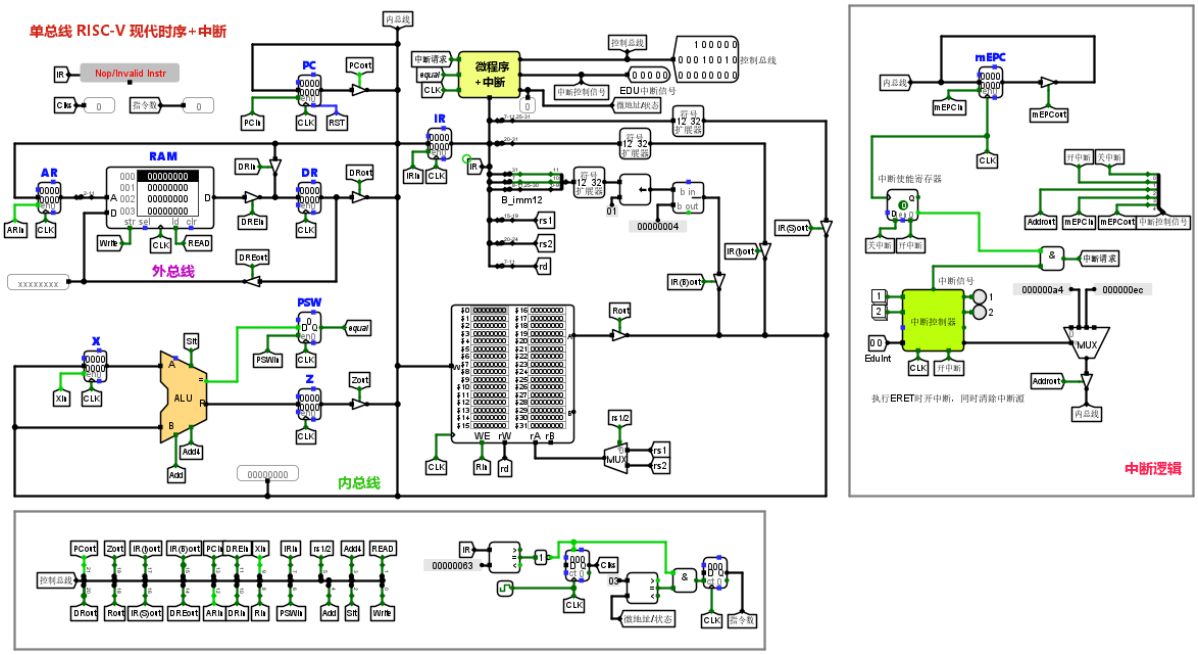


图 2.5 CPU 通路图

2.2.6 支持中断的现代时序硬布线控制器状态机设计

这一部分主要是硬布线控制器的设计，利用组合逻辑完成对入口地址的输出，主要是利用当前状态、指令类型以及测试相关的信号即可确定入口地址，即表示图 2.2 的状态转移逻辑即可，较为简单，不多叙述。

2.2.7 支持中断的现代时序硬布线控制器设计

这一部分主要是组合硬布线控制器而已，即利用微指令地址取出相应控制信号，可见图 2.6.

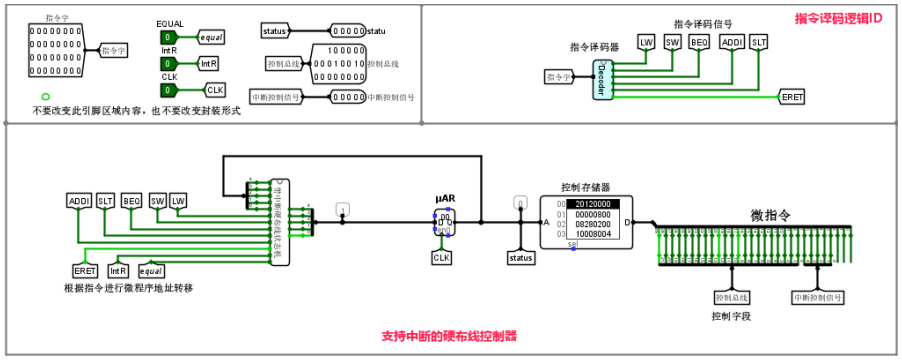


图 2.6 现代时序硬布线控制器

2.3 测试与分析

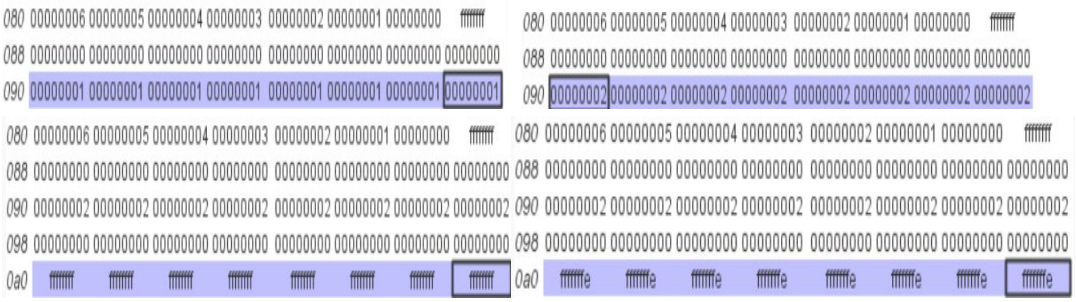


图 2.7 微程序控制器测试结果

图 2.8 硬布线控制器测试结果

由图 2.7 可以看到，采用微程序控制器的 cpu 成功的运行了冒泡排序，并成功响应了两种中断。由图 2.8 可以看到，采用硬布线控制器的 cpu 成功的运行了冒泡排序，并成功响应了两种中断，测试正确。

3 总结与心得

3.1 实验总结

本次实验主要完成了如下几点工作：

1） 完成了三级时序变长指令 CPU 的设计和现代时序支持中断的设计。

2） 理解了 CPU 的设计原理和具体实现。

3） 粗略了解了中断的处理流程。

4） 理解了硬布线和微程序两种不同控制方式的差异和实现方式。

3.2 实验心得

本次实验最大的收获是深入了解了 CPU 的结构和设计，还有之前存储系统、运算器、数据表示等一系列内容，把课本上学到的知识都用了起来，从这个角度来说，实验设计的是很到位的，基本上书上讲的都在实验中实践到了。就我个人的体验来说，很满意实验的安排，甚至对整个组原从理论课到实践课都很满意，是目前大学以来最满意的课程。不过有一点困惑的是，为什么不用 verilog 来进行实验，听清华和浙大的同学，他们都是用 verilog 进行的相关实验，不过我也不太了解 verilog 的好坏，所以不做评价，只是提出困惑罢了，最后衷心祝愿，华科的课程越来越好，都像组原甚至比组原更好。

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：张睿 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： _____