计算机组成原理　　课程设计报告

题　　目：	5 段流水 CPU 设计
专　　业：	计算机科学与技术
班　　级：	CS1706
学　　号：	U21714762
姓　　名：	梁一飞
电　　话：	15387221573
邮　　件：	349062884@qq.com

目　录

1 课程设计概述

1.1 课设目的

计算机组成原理是计算机专业的核心基础课。该课程力图以“培养学生现代计算机系统设计能力”为目标，贯彻“强调软/硬件关联与协同、以 CPU 设计为核心/层次化系统设计的组织思路，有效地增强对学生的计算机系统设计与实现能力的培养”。课程设计是完成该课程并进行了多个单元实验后，综合利用所学的理论知识，并结合在单元实验中所积累的计算机部件设计和调试方法，设计出一台具有一定规模的指令系统的简单计算机系统。所设计的系统能在 LOGISIM 仿真平台和 FPGA 实验平台上正确运行，通过检查程序结果的正确性来判断所设计计算机系统正确性。

课程设计属于设计型实验，不仅锻炼学生简单计算机系统的设计能力，而且通过进行中央处理器底层电路的实现、故障分析与定位、系统调试等环节的综合锻炼，进一步提高学生分析和解决问题的能力。

1.2 设计任务

本课程设计的总体目标是利用 FPGA 以及相关外围器件，设计五段流水 CPU，要求所设计的流水 CPU 系统能支持自动和单步运行方式，能正确地执行存放在主存中的程序的功能，对主要的数据流和控制流通过 LED、数码管等适时的进行显示，方便监控和调试。尽可能利用 EDA 软件或仿真软件对模型机系统中各部件进行仿真分析和功能验证。在学有余力的前提下，可进一步扩展相关功能。

1.3 设计要求

（1） 根据课程设计指导书的要求，制定出设计方案；

（2） 分析指令系统格式，指令系统功能。

（3） 根据指令系统构建基本功能部件，主要数据通路。

（4） 根据功能部件及数据通路连接，分析所需要的控制信号以及这些控制信号的有效形式；

（5） 设计出实现指令功能的硬布线控制器；

（6） 调试、数据分析、验收检查；

（7） 课程设计报告和总结。

1.4 技术指标

（8） 支持表 1.1 前 31 条基本 32 位 MIPS 指令；

（9） 支持教师指定的 4 条扩展指令；

（10）支持多级嵌套中断，利用中断触发扩展指令集测试程序；

（11）支持 5 段流水机制，可处理数据冒险，结构冒险，分支冒险；

（12）能运行由自己所设计的指令系统构成的一段测试程序，测试程序应能涵盖所有指令，程序执行功能正确。

（13）能运行教师提供的标准测试程序，并自动统计执行周期数

（14）能自动统计各类分支指令数目，如不同种类指令的条数、冒险冲突次数、插入气泡数目、load-use 冲突次数、动态分支预测流水线能自动统计预测成功与失败次数。

表 1.1 指令集

#	指令助记符	简单功能描述	备注
1	ADD	加法	指令格式参考 MIPS32 指令集，最终功能以 MARS 模拟器为准。
2	ADDI	立即数加	
3	ADDIU	无符号立即数加	
4	ADDU	无符号数加	
5	AND	与	
6	ANDI	立即数与	
7	SLL	逻辑左移	
8	SRA	算数右移	
9	SRL	逻辑右移	
10	SUb	减	
11	OR	或	
12	ORI	立即数或	

#	指令助记符	简单功能描述	备注
13	NOR	或非	
14	LW	加载字	
15	SW	存字	
16	BEQ	相等跳转	
17	BNE	不相等跳转	
18	SLT	小于置数	
19	STI	小于立即数置数	
20	SLTU	小于无符号数置数	
21	J	无条件转移	
22	JAL	转移并链接	
23	JR	转移到指定寄存器	If $v0==10 halt(停机指令)
24	SYSCALL	系统调用	else 数码管显示$a0 值
25	MFC0	访问 CP0	中断相关，可简化，选做
26	MTC0	访问 CP0	中断相关，可简化，选做
27	ERET	中断返回	异常返回，选做
28	SRAV	算术可变右移	
29	SLTIU	小于立即数置 1(无符号)	
30	SB	存储字节	
31	BLEZ	小于等于 0 转移	

2 总体方案设计

2.1 单周期 CPU 设计

采用硬布线控制方式。

总体结构图如图 2.1 所示。

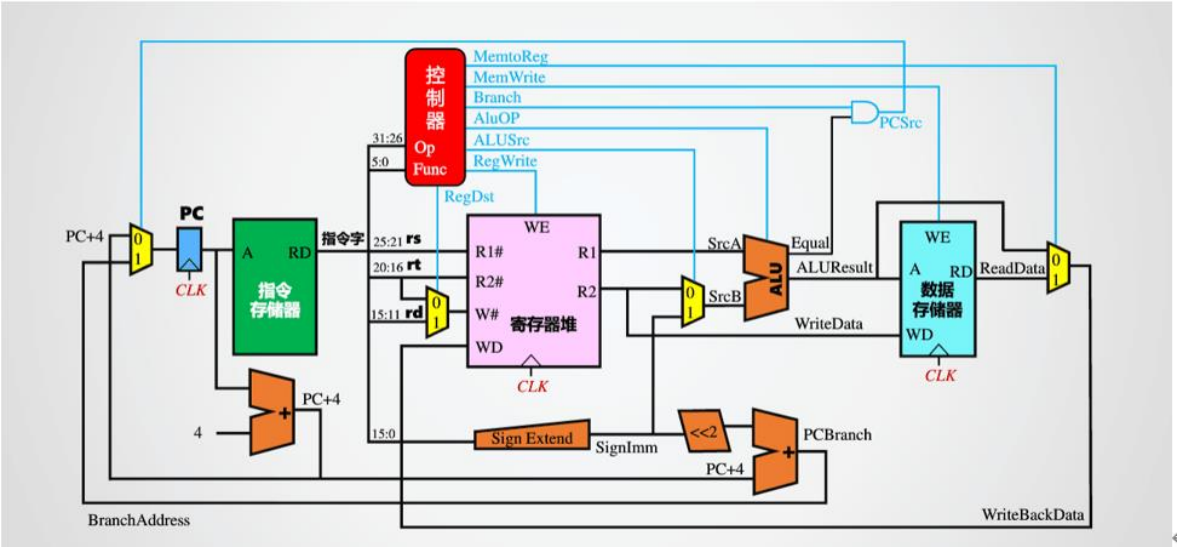


图 2.1 总体结构图

2.1.1 主要功能部件

运算器部分，具体设计思路如下

1. 程序计数器 PC

采用一个 32 位寄存器实现

2. 指令存储器 IM

采用 32 位只读存储器（ROM）实现，地址位宽为 10 位

3. 运算器

表 2.1 算术逻辑运算单元引脚与功能描述

引脚	输入/输出	位宽	功能描述
X	输入	32	操作数 X
Y	输入	32	操作数 Y
ALU_OP	输入	4	运算器功能码，具体功能见下表
shamt	输出	5	移位数
Result	输出	32	ALU 运算结果
Result2	输出	32	ALU 结果第二部分，用于乘法指令结果高位或除法指令的余数位，其他操作为零
Equal	输出	1	Equal=(x==y)?1:0, 对所有操作有效

表 2.2 运算器功能码

ALU_OP	十进制	运算功能
0000	0	Result = X << Y　逻辑左移 （Y 取低五位） Result2=0
0001	1	Result = X >>>Y　算术右移 （Y 取低五位） Result2=0
0010	2	Result = X >> Y　逻辑右移 （Y 取低五位） Result2=0
0011	3	$Result = (X * Y)_{[31:0]}$; $Result2 = (X * Y)_{[63:32]}$ 无符号乘法
0100	4	Result = X/Y;　Result2 = X%Y　无符号除法
0101	5	Result = X + Y　(Set OF/UOF)
0110	6	Result = X - Y　(Set OF/UOF)
0111	7	Result = X & Y　按位与
1000	8	Result = X | Y　按位或
1001	9	Result = X⊕Y　按位异或
1010	10	Result = ~(X |Y)　按位或非
1011	11	Result = (X < Y) ? 1 : 0　符号比较
1100	12	Result = (X < Y) ? 1 : 0　无符号比较

4. 寄存器堆 RF

如图所示：

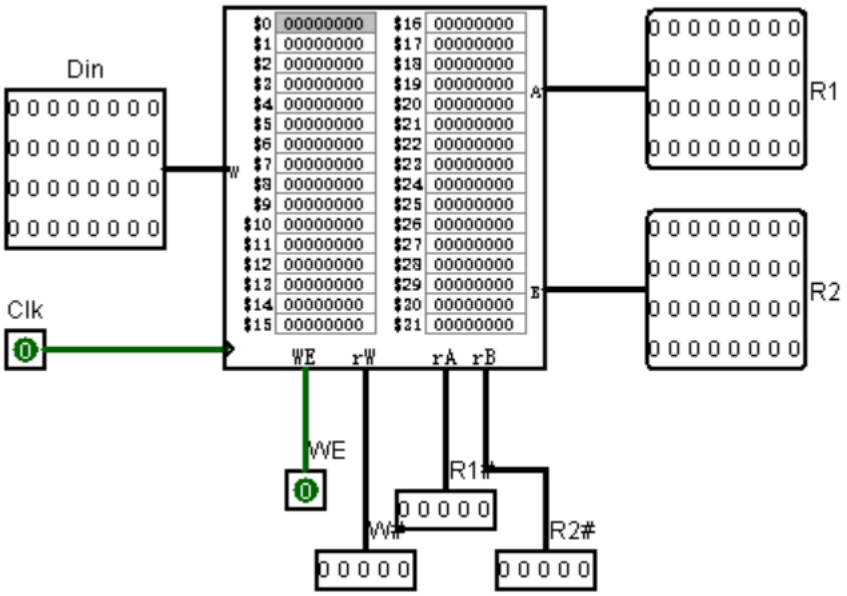


图 2.2 寄存器堆图

2.1.2 数据通路的设计

见 2.1.3 真值表

2.1.3 控制器的设计

首先对于控制信号进行统计，包括各个主要部件所需要输入的控制信号，以及数据通路合并表中所示的具有多输入的主要部件需要进行输入选择的控制信号，并且对各个统计信号的各种取值情况进行定义，统计得到的控制信号以及说明如表 2.2。

表 2.2 主控制器控制信号的作用说明

#	控制信号	信号说明	产生条件（信号为1）
1	RegWrite	寄存器写使能	寄存器写回信号
2	MemWrite	写内存控制信号	sw指令　未单独设置MemRead信号
3	AluOP	运算器操作控制符（4位）	R型指令根据Func选择
4	MemToReg	寄存器写入数据来自存储器	lw指令
5	RegDst	写入寄存器编号rt/rd选择	R型指令
6	AluSrcB	运算器B输入选择	lw指令，sw指令，立即数运算类指令
7	SignedExt	立即数符号扩展	ADDI、ADDIU、SLTI指令
8	JR	寄存器跳转指令译码信号	JR指令
9	JAL	JAL指令译码信号	JAL指令，选择寄存器写回编号，写回值
10	JMP	无条件分支控制信号	J、JAL、JR指令，选择无条件分支地址
11	Beq	Beq指令译码信号	Beq指令，用于有条件分支控制
12	Bne	Bne指令译码信号	Bne指令，用于有条件分支控制
13	Syscall	Syscall指令译码信号	根据$V0寄存器的值，决定是停机还是输出
14	SRAV	SRAV指令译码信号	SRAV指令，算术可变右移
15	SB	SB指令译码信号	SB指令，存储字节
16	BLEZ	BLEZ译码信号	BLEZ指令，小于等于 0 转移

对照所有控制信号，依次分析各条指令，分析该指令执行过程中需要哪些控制信号，对于与本条指令无关的控制信号，控制信号的取值一律为 0，以简化控制器电路的设计。该控制信号表的框架如表 2.3 所示。

表 2.3 主控制器控制信号框架

#	指令	OpCode (十进制)	FUNCT (十进制)	ALU_OP	MemtoReg	MemWrite	ALU_SRC	RegWrite	SYSCALL	SignedExt	RegDst	BEQ	BNE	JR	JMP	JAL	SRAV	SB	BLEZ
1	SLL	0	0	0				1			1								
2	SRA	0	3	1				1			1								
3	SRL	0	2	2				1			1								
4	ADD	0	32	5				1			1								
5	ADDU	0	33	5				1			1								
6	SUB	0	34	6				1			1								
7	AND	0	36	7				1			1								
8	OR	0	37	8				1			1								
9	NOR	0	39	10				1			1								
10	SLT	0	42	11				1			1								
11	SLTU	0	43	12				1			1								
12	JR	0	8	X										1	1				
13	SYSCALL	0	12	X					1										
14	J	2	X	X											1				
15	JAL	3	X	X				1							1	1			
16	BEQ	4	X	X								1							
17	BNE	5	X	X									1						
18	ADDI	8	X	5			1	1		1									
19	ANDI	12	X	7			1	1											
20	ADDIU	9	X	5			1	1		1									
21	SLTI	10	X	11			1	1		1									
22	ORI	13	X	8			1	1											
23	LW	35	X	5	1		1	1		1									
24	SW	43	X	5		1	1			1									
25	SRAV	0	7	1				1			1						1		
26	SLTIU	11	X	12			1	1		1									
27	SB	40	X	5		1	1			1								1	
28	BLEZ	6	X	11															1

2.2 中断机制设计

2.2.1 总体设计

如下图所示：

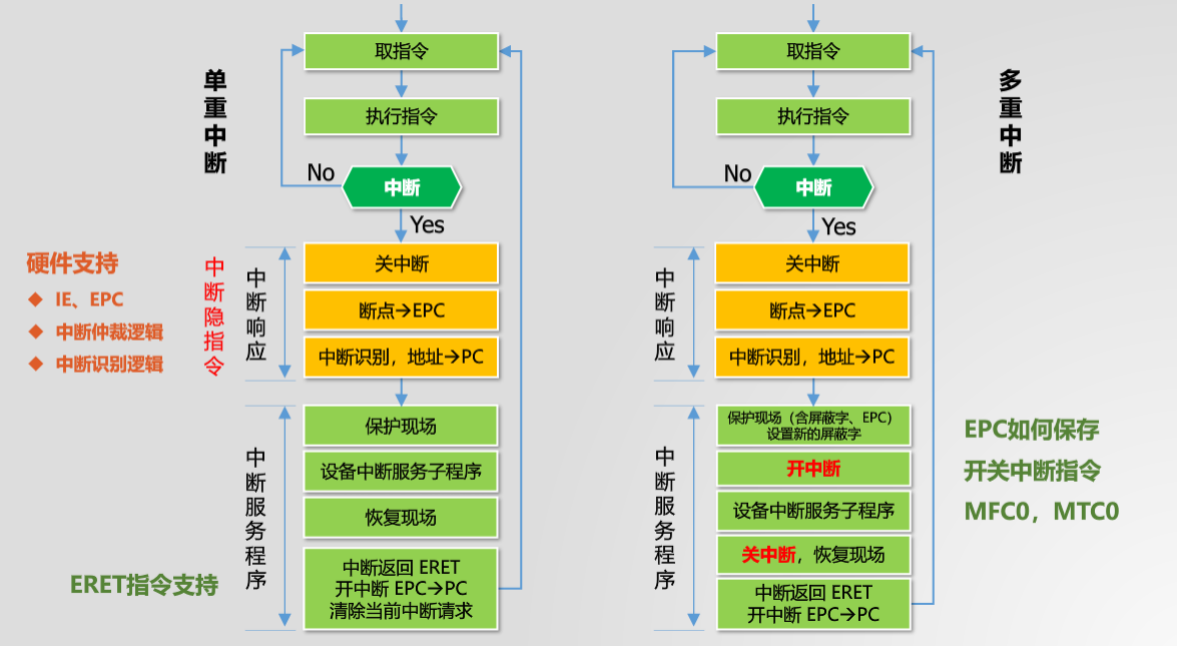


图 2.3 中断机制总体设计

2.2.2 硬件设计

主要需要实现：1.按键中断外围电路，功能：产生中断请求，锁存中断请求，清除中断请求。2.中断仲裁与中断识别电路，功能：保存断点，开关中断，中断仲裁，中断识别。设计如下图：

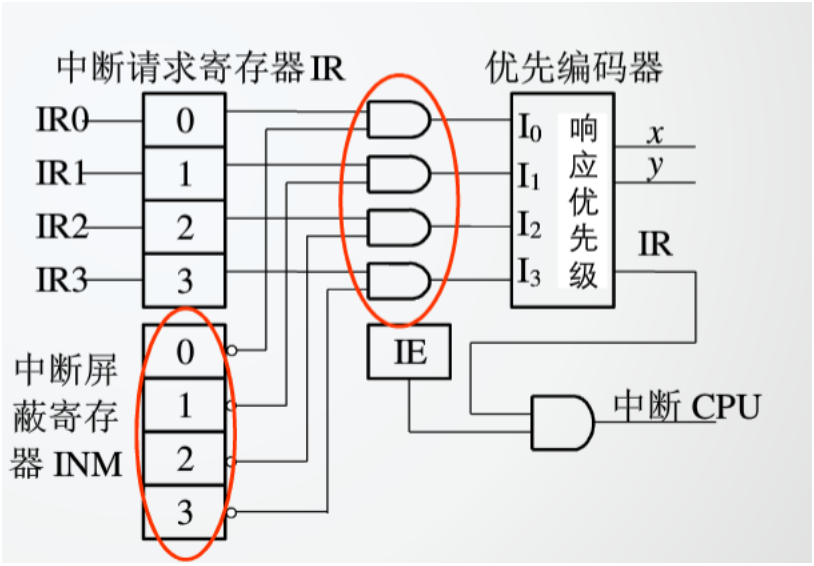


图 2.4 中断机制硬件设计

2.3 流水 CPU 设计

2.3.1 总体设计

将单周期 CPU 细分成五个阶段：IF,ID,EX,MEM,WB。在段间设置流水接口部件，接口本质是寄存器，锁存前端加工完成的数据。通过接口传递与指令相关的数据、控制、反馈信息。后段对数据加工处理依赖于前段接口传递过来的信息。整体如下图所示：

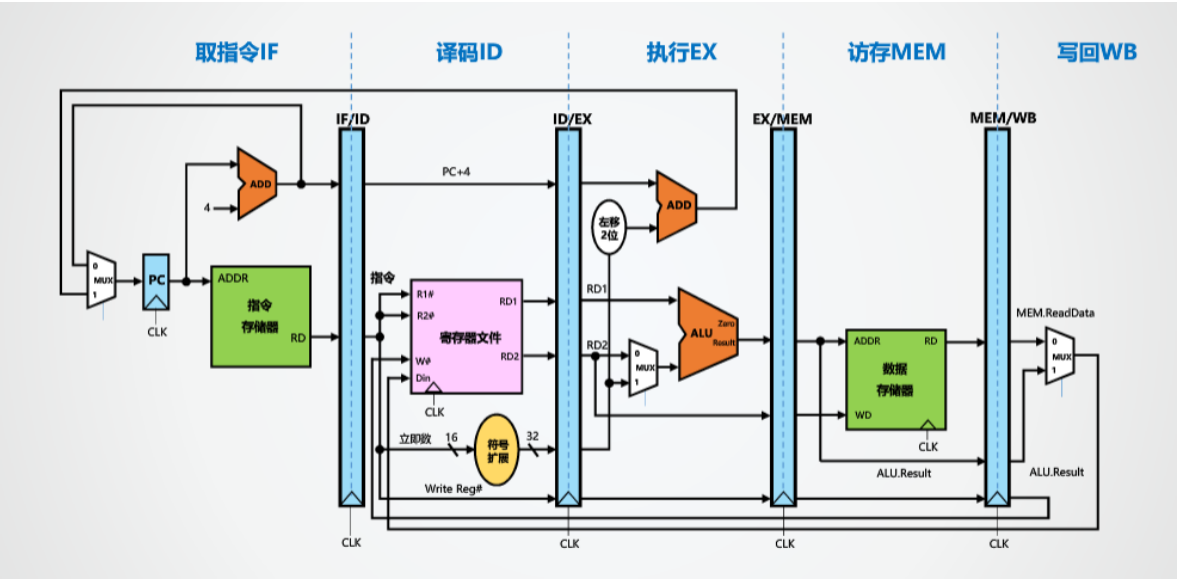


图 2.5 流水总体设计

2.3.2 流水接口部件设计

部件内部是若干寄存器，用于锁存数据与信号，控制端包括同步清零（插入气泡），使能端（stall 信号）

2.3.3 理想流水线设计

理想流水线不需要考虑太多，实现五段流水即可，具体设计见 2.3.1

2.4 气泡式流水线设计

出现指令相关时插入气泡，IF 段重新获取新的指令，IF/ID,ID/EX 段给出同步清零信号。ID 段和 WB 段有数据相关时，才用先写后读的形式消除，寄存器文件下跳沿读入，流水接口上跳沿有效。ID 段和 MEM 段有数据相关时，IF 段，ID 段暂停等待数据写回，EX 段插入气泡，则在下一时刻数据相关变为与 WB 段有关，处理逻辑同上。ID 段和 EX 段有数据相关时，IF 段，ID 段暂停等待数据写回，EX 段插入气泡，则在下一时刻变为与 MEM 段有关，处理逻辑同上。同时，还要设计数据相关检测逻辑：构建源寄存器使用情况子电路，构建数据相关检测子电路。IF 段，ID 段暂停逻辑的实现：利用数据相关信号控制对应部件写使能，低电平有效。EX 段插入气泡逻辑的实现：利用数据相关信号控制 ID/EX 接口的同步清零信号。

2.5 重定向流水线设计

构建重定向通路：1.在第一次使用寄存器的位置增加多路选择器 2.连接可能的重定向通路。构建重定向逻辑：在 ID 段根据数据相关情况产生对应的重定向控制信号。构建 Load-Use 检测器：在 ID 段增加 Load-Use 插入气泡逻辑。

3 详细设计与实现

3.1 单周期 CPU 实现

3.1.1 主要功能部件实现

1) 程序计数器（PC）

① Logism 实现：

使用一个 32 位寄存器实现程序计数器 PC，触发方式为下降沿触发，输入为下一条将要执行的指令的地址，输出为当前执行指令的地址。使能端由 syscall 指令以及相应的判断逻辑相与控制

2) 指令存储器（IM）

① Logism 实现：

使用一个只读存储器 ROM 实现指令存储器（IM）。设置该只读存储器的地址位宽为 10 位，数据位宽为 32 位。因为 PC 中存储的指令地址有 32 位，而 ROM 地址线宽度有限，仅为 10 位，故将 32 位指令地址高位部分和字节偏移部分直接屏蔽，使用分线器只取 32 位指令地址的 2-11 位作为指令存储器的输入地址。如图 3.所示。

图 3.1 指令存储器（IM）

3.1.2 数据通路的实现

在完成指令系统数据通路表的填写之后，根据列出的数据通路表，进行多指令数据通路的合并输入数，表，将各个主要功能部件进行连接，根据数据通路合并表的最终结果，对于所有的多输入部件使用多路选择器进行输入选择。最终便可以完成数据通路的搭建。

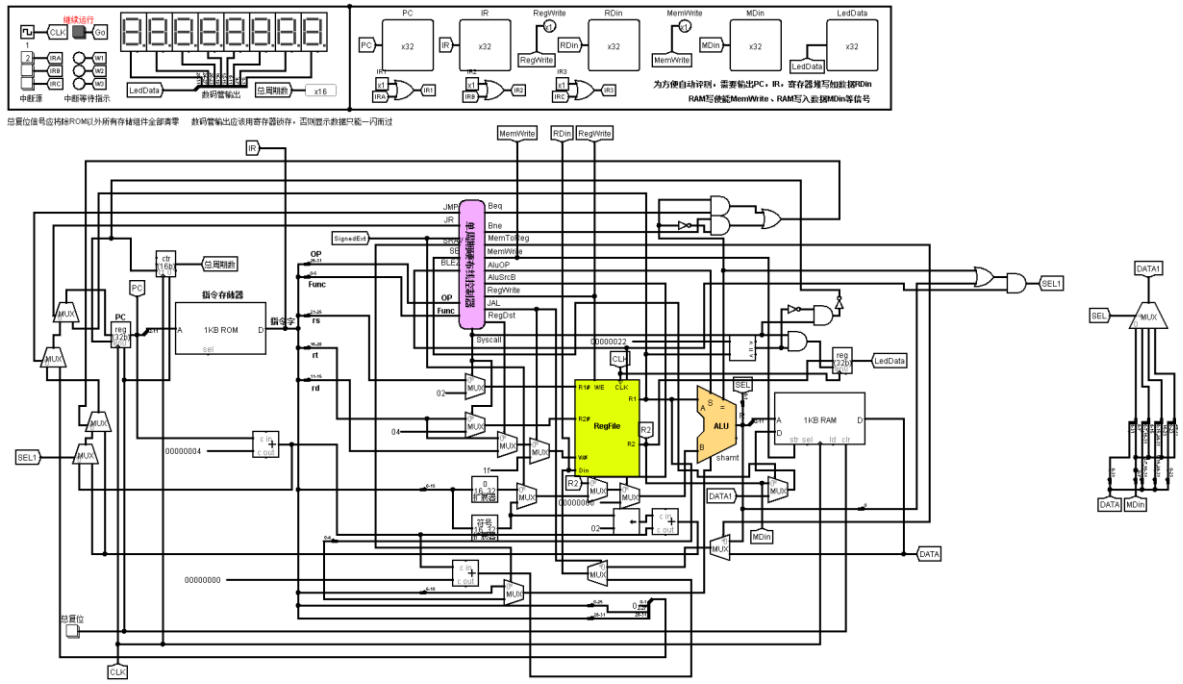


图 3.2 单周期 CPU 数据通路（Logism）

3.1.3 控制器的实现

根据数据通路填写真值表，在 logism 里自动生成控制电路，表见 2.1.3.最终生成的控制器电路如图所示：

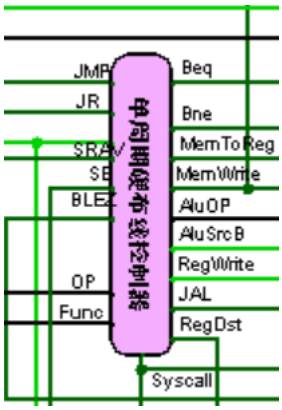


图 3.3 单周期 CPU 硬布线控制器（Logism）

3.2 中断机制实现

3.2.1 单极中断实现

根据编写的中断程序，当执行特定指令时，发出中断请求，电路如图：

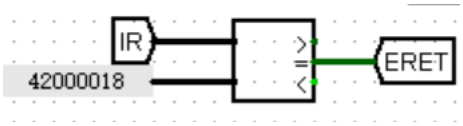
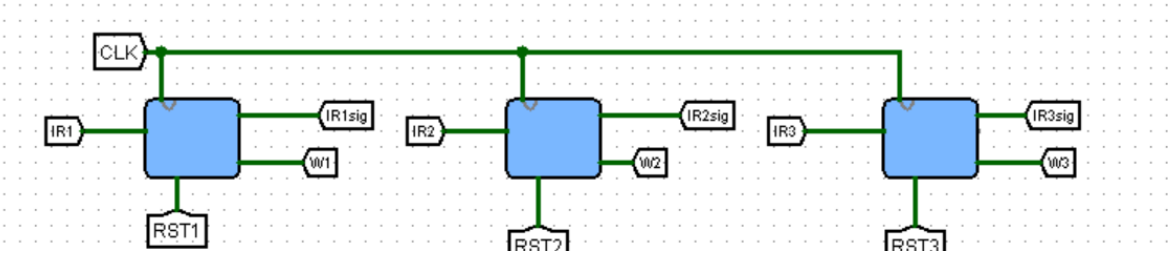
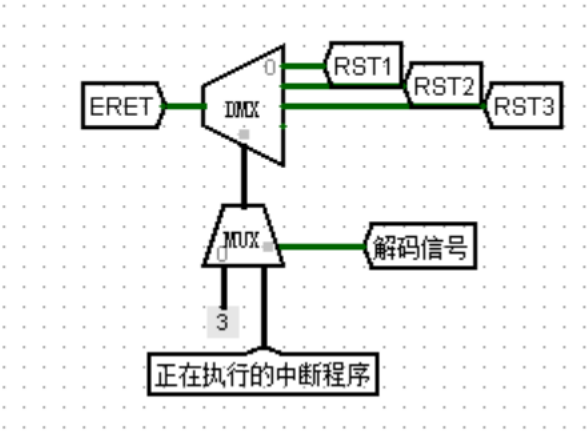


图 3.4 中断请求

中断仲裁电路与中断识别电路如下图所示：





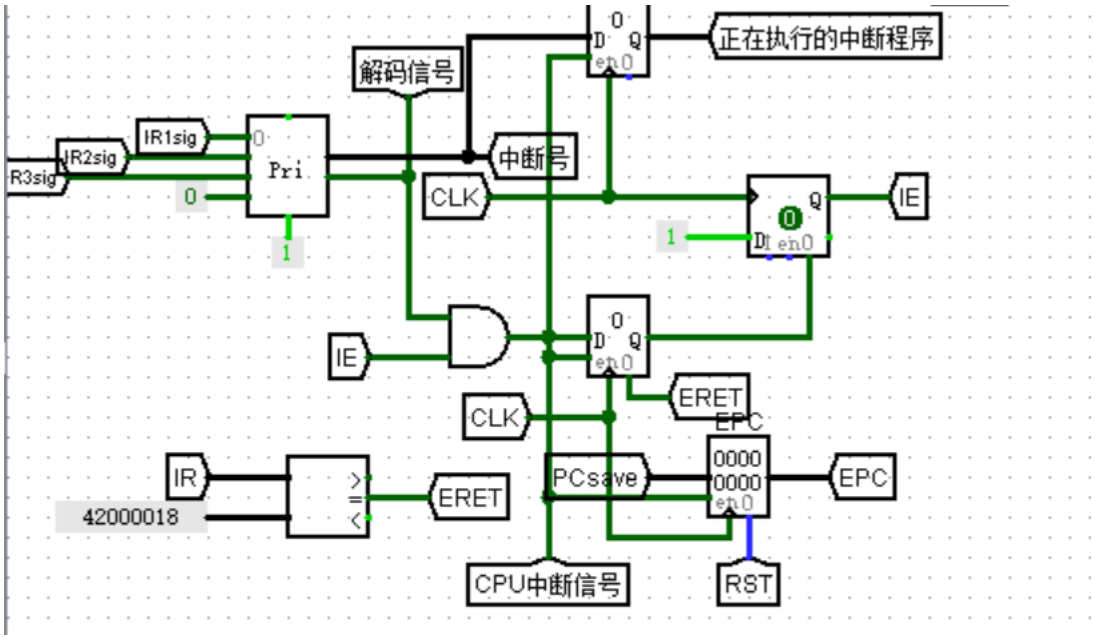


图 3.5 中断仲裁及中断识别电路

运用多路选择器实现根据中断号选择中断地址，如下图所示：

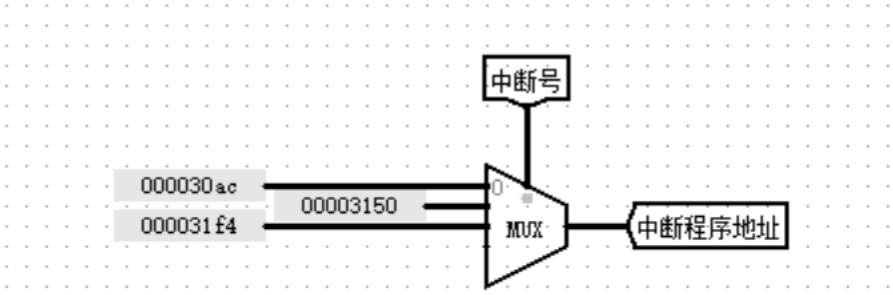


图 3.6 选择中断程序地址

中断信号处理电路如图所示：

增加一个多路选择器保护现场：

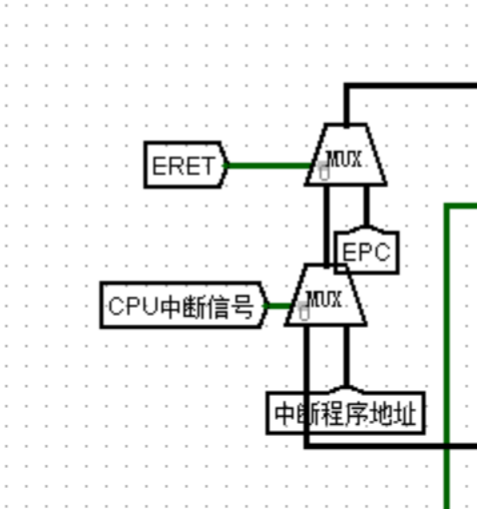
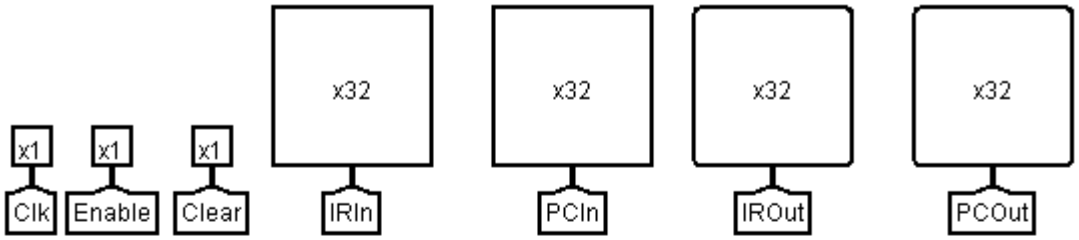


图 3.7 保护现场

3.3 流水 CPU 实现

3.3.1 流水接口部件实现

IF/ID 段接口部件实现如下图所示：



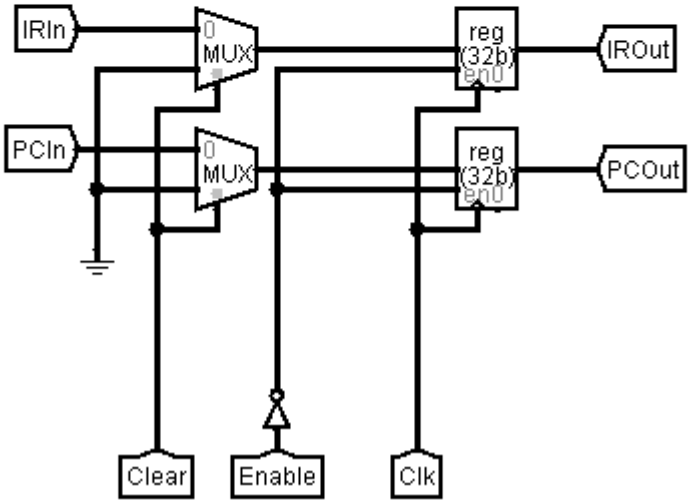


图 3.8 IF/ID 段

ID/EX 段接口部件实现如下图所示：

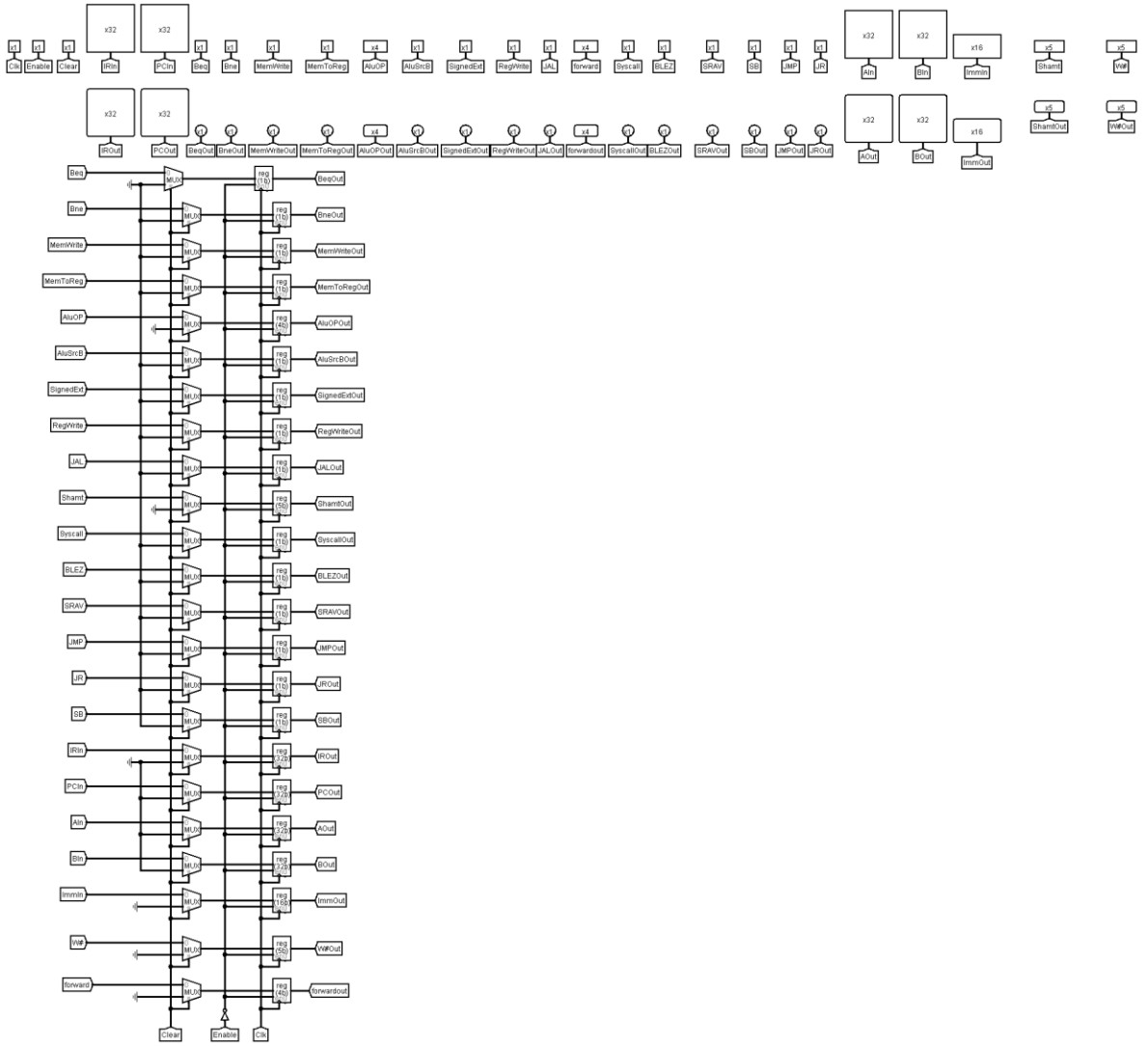
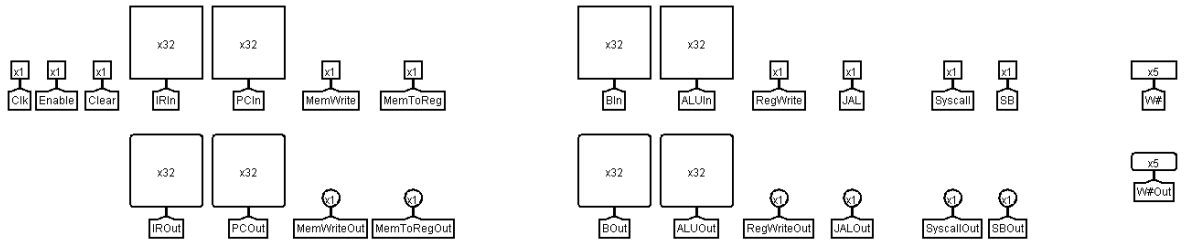


图 3.9 ID/EX 段

EX/MEM 段接口部件实现如下图所示：



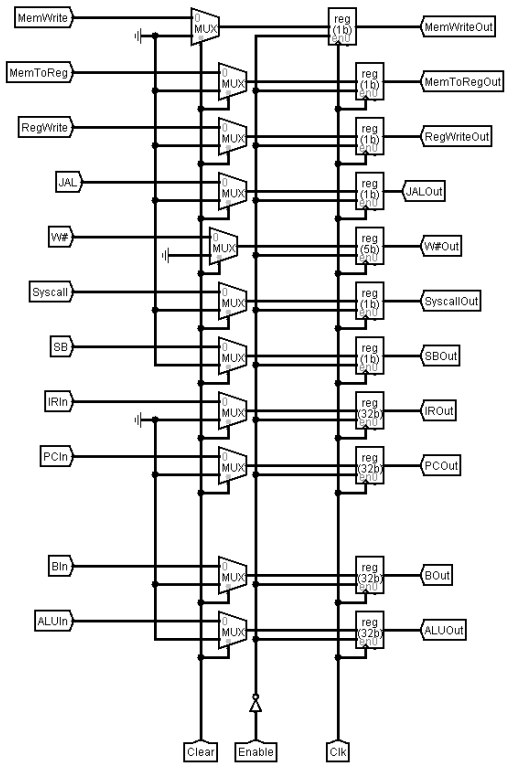


图 3.10 EX/MEM 段

MEM/WB 段接口部件实现如下图所示：

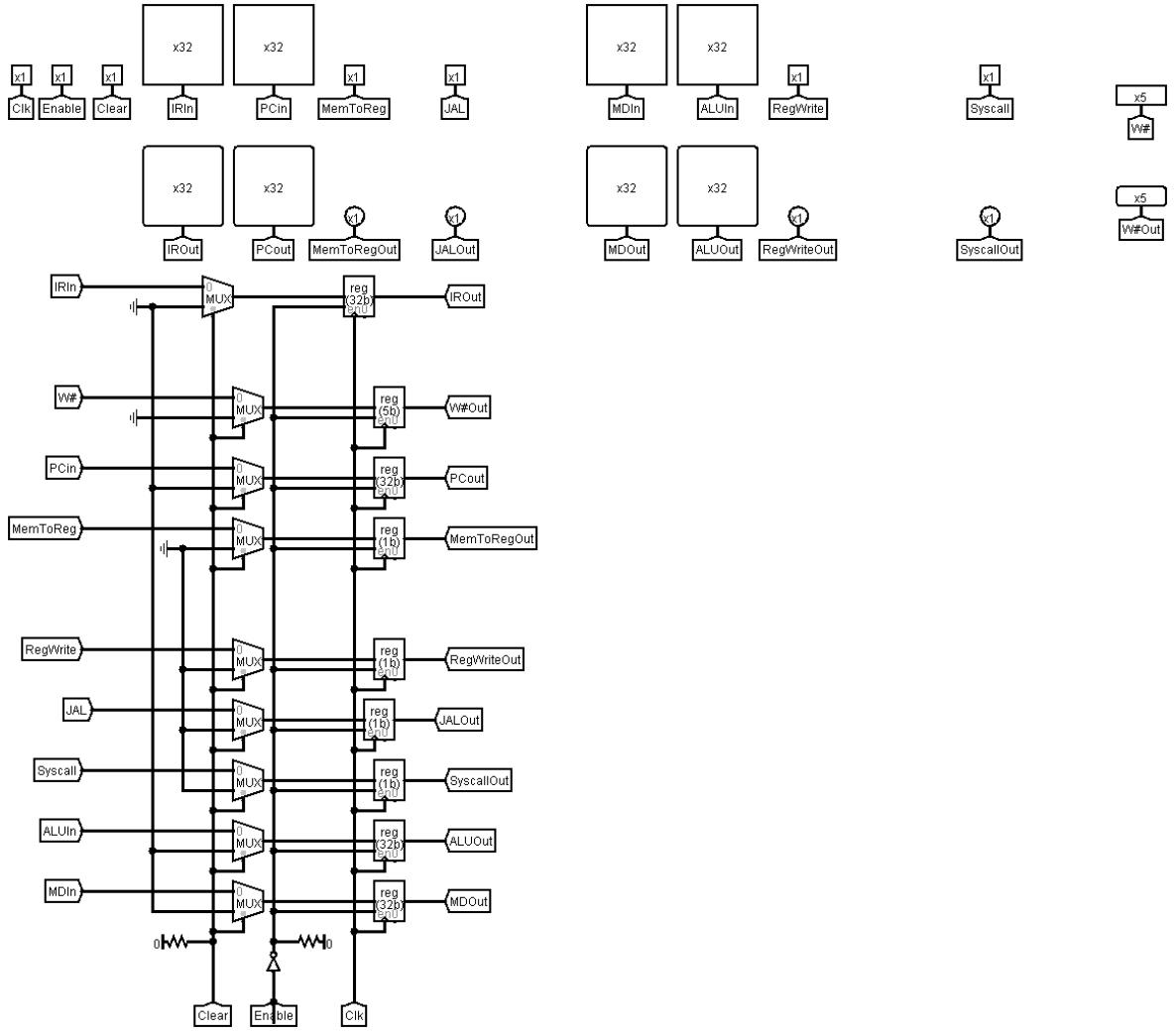


图 3.11 MEM/WB 段

3.3.2 理想流水线实现

主要是将单周期 CPU 进行分段，利用流水接口部件连接，实现如下图所示：

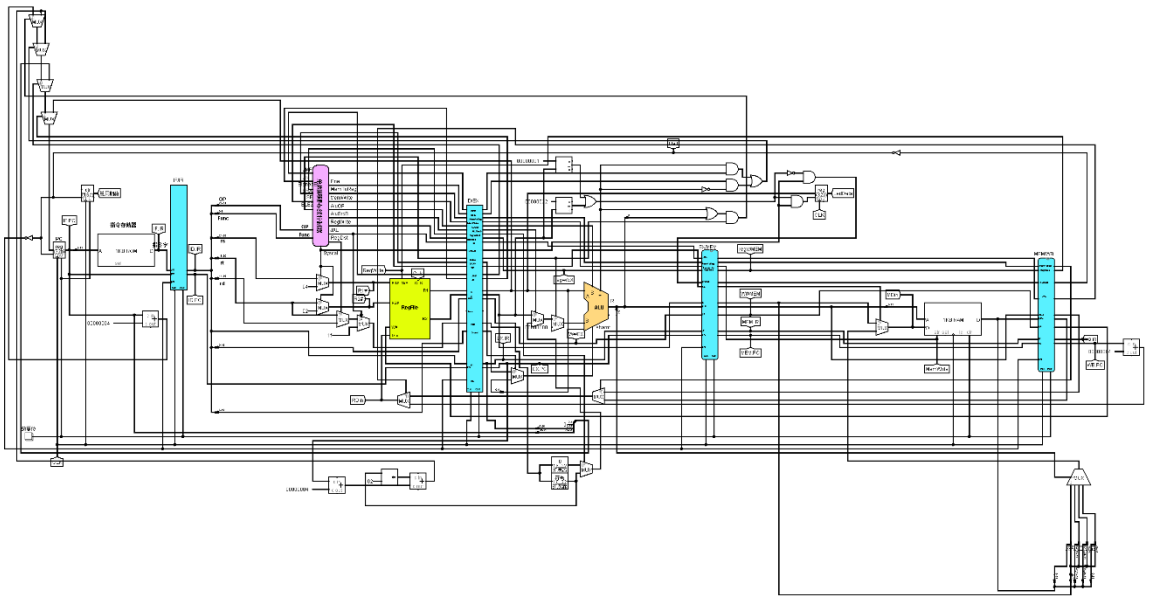


图 3.12 理想流水线实现

3.4 气泡式流水线实现

在理想流水线的基础上，引入指令相关处理和数据相关处理。数据相关检测电路如下图所示：

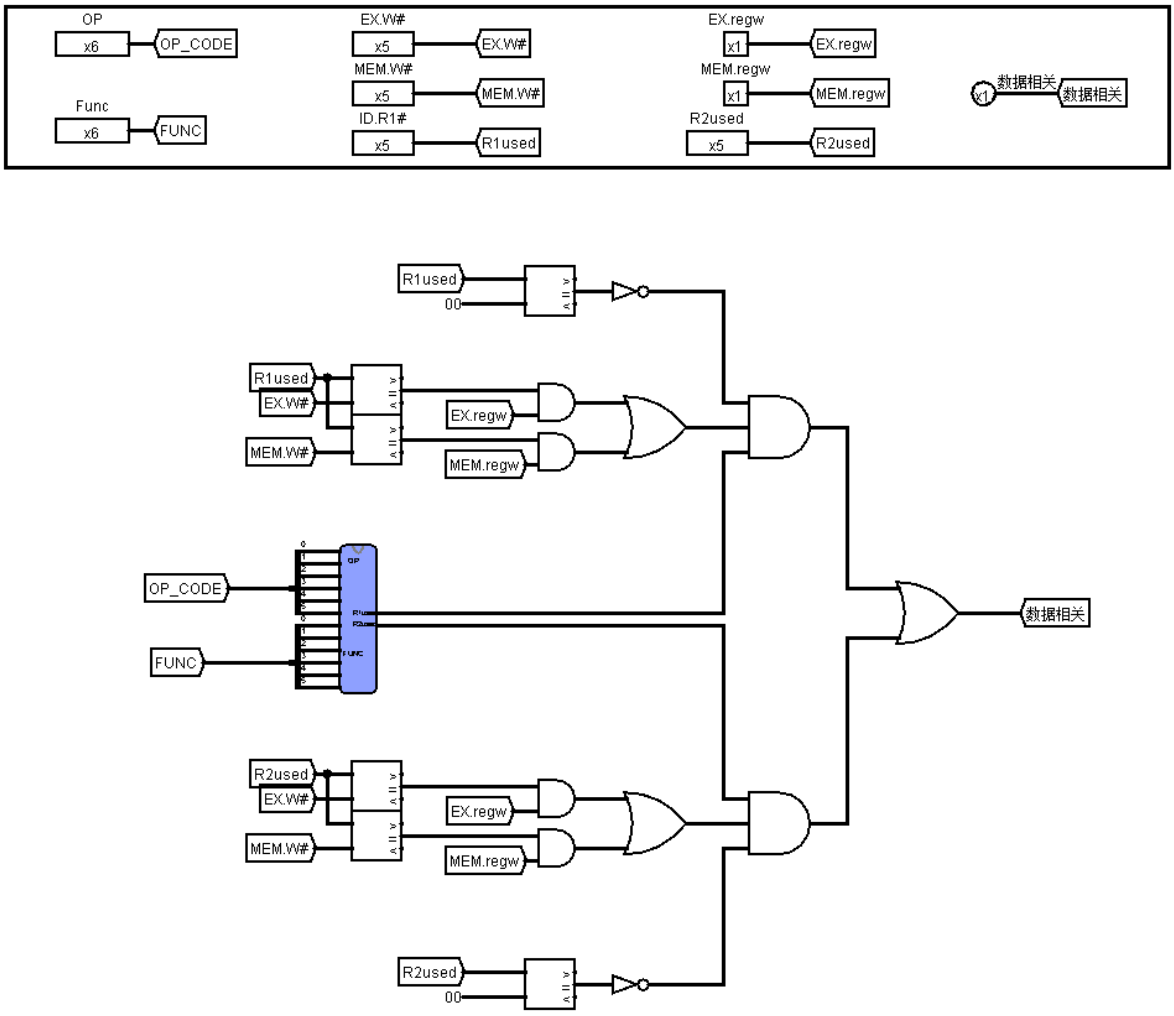


图 3.13 数据相关检测

气泡流水线如下图所示：

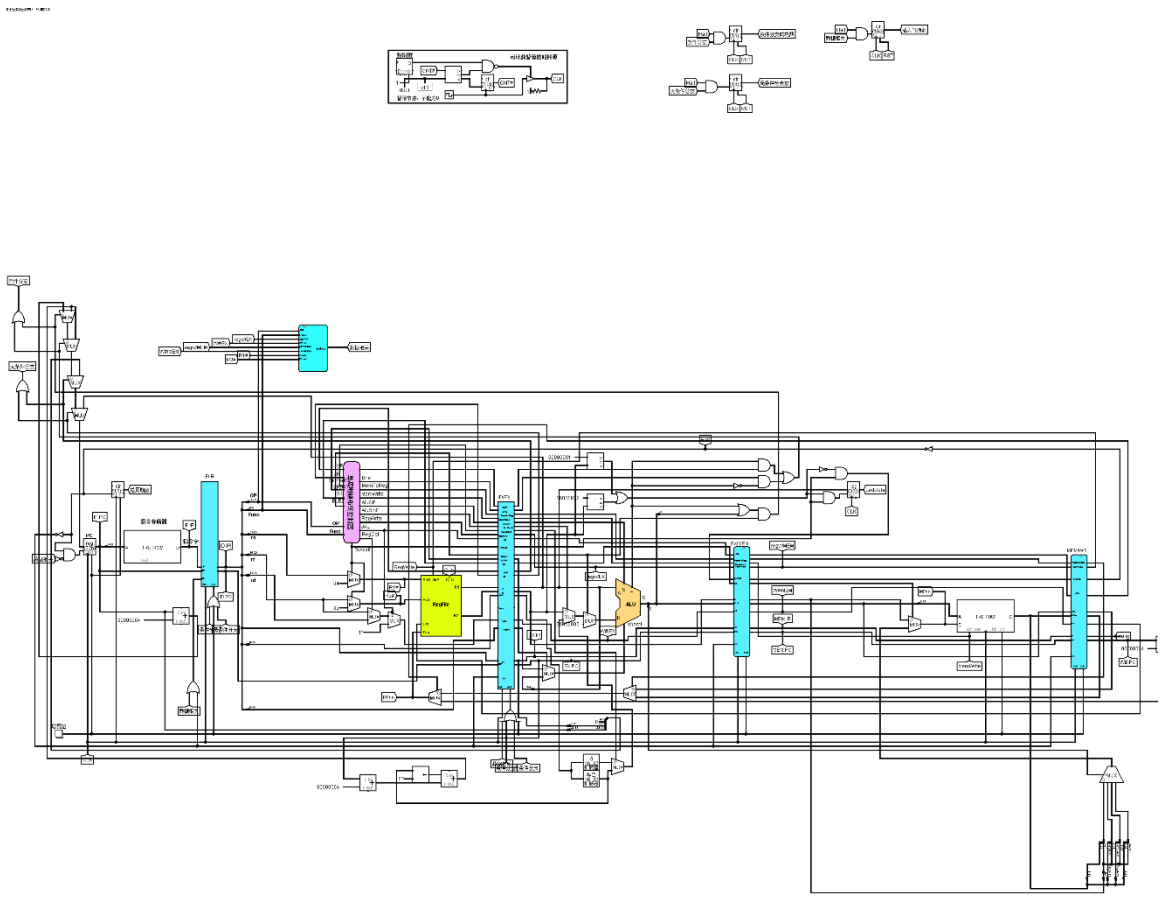


图 3.14 气泡流水线

3.5 重定向流水线实现

在气泡流水线的基础上，引入重定向检测电路以及源寄存器相关电路，将气泡流水线的数据相关替换为 LoadUse 相关。源寄存器相关电路由真值表自动生成，真值表如下图所示是：

指令	OpCode (十进制)	FUNCT (十进制)	ALU_OP	MemtoReg	MemWrite	ALU_SRC	RegWrite	SYSCALL	SignedExt	RegDst	BEQ	BNE	JR	JMP	JAL	SRAV	SB	BLEZ	R1_used	R2_used
SLL	0	0	0				1			1									1	1
SRA	0	3	1				1			1									1	1
SRL	0	2	2				1			1									1	1
ADD	0	32	5				1			1									1	1
ADDU	0	33	5				1			1									1	1
SUB	0	34	6				1			1									1	1
AND	0	36	7				1			1									1	1
OR	0	37	8				1			1									1	1
NOR	0	39	10				1			1									1	1
SLT	0	42	11				1			1									1	1
SLTU	0	43	12				1			1									1	1
JR	0	8	X										1	1					1	
SYSCALL	0	12	X					1											1	1
J	2	X	X											1						
JAL	3	X	X				1							1	1					
BEQ	4	X	X								1								1	1
BNE	5	X	X									1							1	1
ADDI	8	X	5			1	1		1										1	
ANDI	12	X	7			1	1												1	
ADDIU	9	X	5			1	1		1										1	
SLTI	10	X	11			1	1		1										1	
ORI	13	X	8			1	1												1	
LW	35	X	5	1		1	1		1										1	
SW	43	X	5		1	1			1										1	1
SRAV	0	7	1				1			1						1			1	1
SLTIU	11	X	12			1	1		1										1	
SB	40	X	5		1	1			1								1		1	
BLEZ	6	X	11															1	1	

图 3.15 源寄存器相关真值表

重定向检测电路如下图所示：

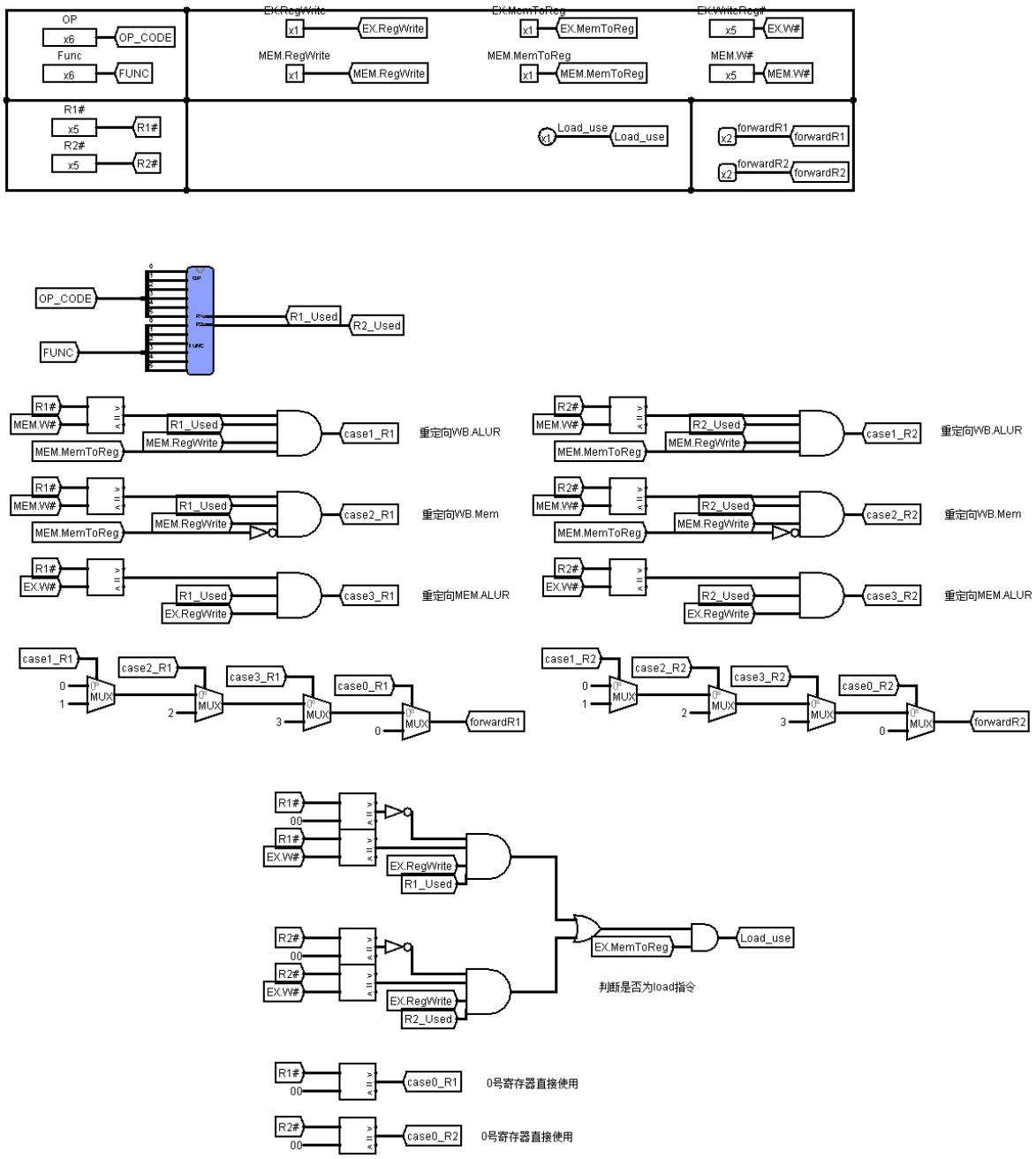


图 3.15 重定向检测电路

由此构建的重定向流水线如下图所示：

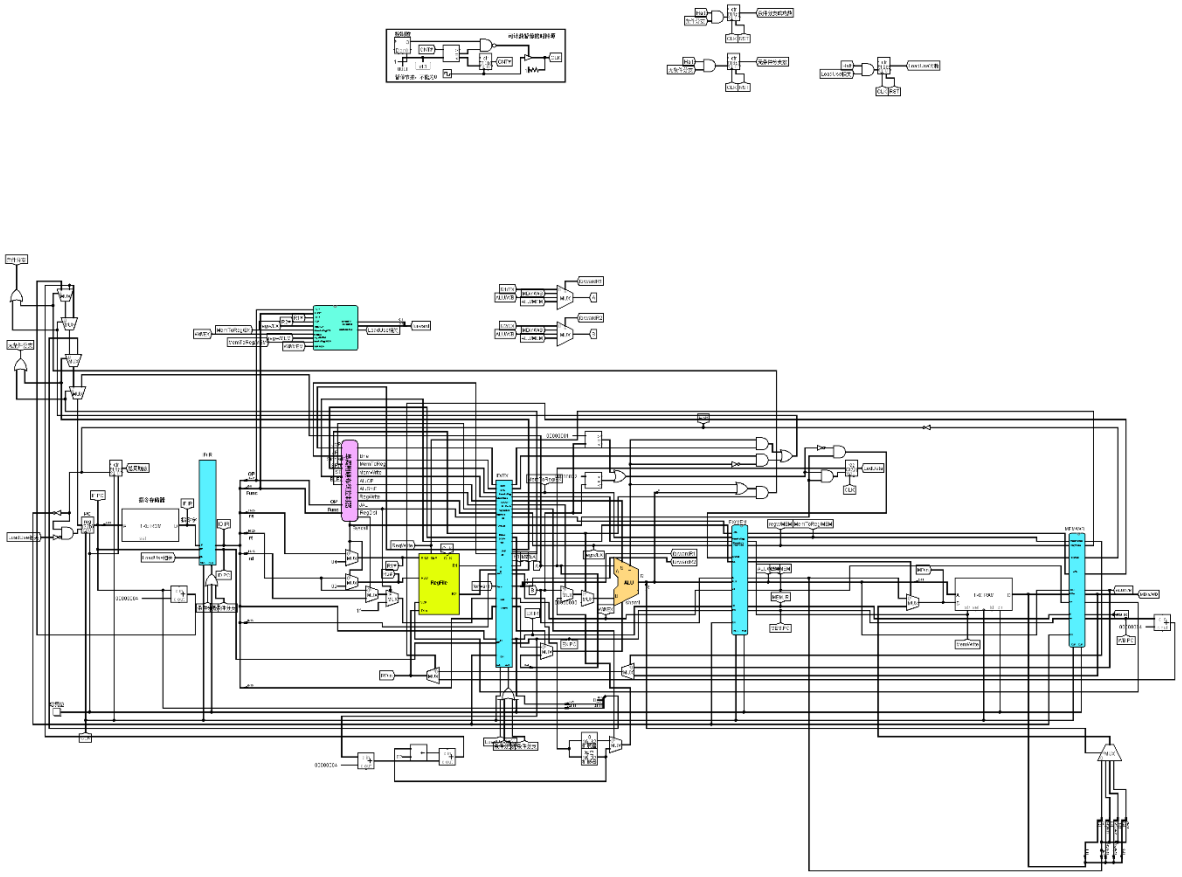


图 3.16 重定向流水线

4 实验过程与调试

4.1 测试用例和功能测试

本次实现采用 educoder 平台进行测试，测试用例为组原群里提供的用例，测试结果如下图所示：

图 4.1 测试结果图

4.2 主要故障与调试

4.2.1 SB 指令错误

单周期 CPU：SB 存储半字长指令错误

故障现象：无法通过测试

原因分析：经和同学讨论发现是自己想得太简单，直接取数据的低 8 位扩展成 32 位，由选择信号选择原始数据或者数据的低 8 位。实际上要根据 ALU 运算结果的高两位确定存储的是 0~7 位还是 8~15、16~22、23~31 位。

解决方案：独立出一个电路，单独选择要存储的数据，如下图所示：

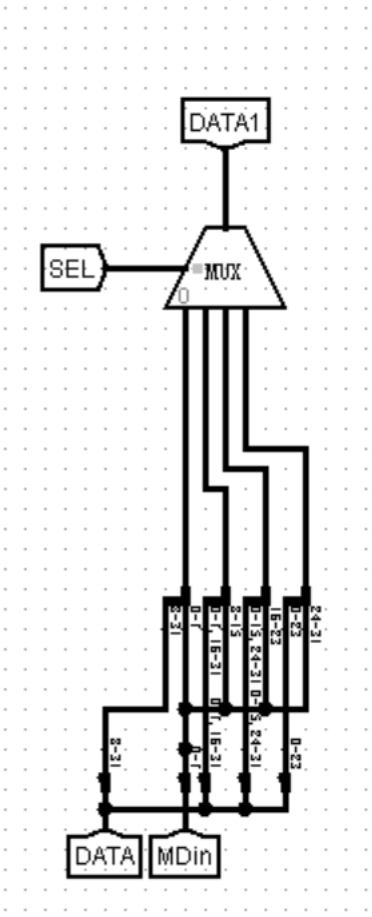


图 4.1 SB 指令解决方案

4.2.2 LedData 故障

气泡流水线：LedData 无法正常显示

故障现象： LedData 始终为 0

原因分析：经查看 R1，R2 的数据均正常，那么只可能是 syscall 指令出了问题，我的 syscall 指令是用的 ID 段的，早了一个时钟周期，故 R1，R2 的数据送过来后相与就成 0 了。

解决方案：将 ID 段 syscall 指令改为 EX 段的，问题成功解决。

4.2.3 气泡流水线测试程序不停止

气泡流水线故障

故障现象：气泡流水线运行测试程序时无法停止

原因分析： 和同学的正确程序对比，发现在 3000 多拍的时候出现不一致，于是回溯一拍，发现 R0 不同，继续回溯，找到问题在于存储器里存储的数值不同，就在那一拍导致 R0 的值不同。于是采用二分法对比拍数，寻找存储器里的值是何时不同的，最终找到是 16 拍出了问题，此处对应的指令是 JAL 指令，JAL 指令负责选择存储，而我用的是 EX 段的 JAL 指令，早了两拍。

解决方案：将选择存储的 JAL 指令由 EX 段改为 WB 段，问题解决。

4.3 实验进度

表 4.1 课程设计进度表

时间	进度
第一天	复习组成原理 CPU 相关理论知识，阅读课设任务书，阅读 MIPS 指令手册，并列出 CPU 各部件的数据通路表，并完成数据通路的基本构建。
第二天	完成单周期 CPU 的控制信号表，使用 Logisim 搭建控制器，实现了单周期 CPU 并且通过了测试。完成部分 Logism 单周期 CPU 故障报告。
第三天	完成 Logism 单周期 CPU 的故障报告，并且通过了 Logism 单周期 CPU 的检查。。
第四天	学习流水线相关知识
第五天	完成 Logism 理想流水线，并通过测试
第六天	初步 Logism 气泡流水线，出现故障
第七天	经调试改好气泡流水线的故障
第八天	完成重定向流水线的源寄存器相关电路以及重定向检测电路

时间	进度
第九天	完成重定向流水线
第十天	学习中断的知识
第十一天	完成单极中断，出现故障
第十二天	调试单极中断电路，修复故障，通过测试
第十三天	在流水线电路加入自己的四条指令
第十四天	讨论团队任务，确定完成目标，在随后的时间内完成。

5 设计总结与心得

5.1 课设总结

5.1.1 个人任务总结

在本次课程设计中，我完成了如下任务：

1.单周期 24 指令 CPU

2.在 24 指令 CPU 的基础上增加个人任务中的四条指令

3.完成理想流水线、气泡流水线、重定向流水线

4.完成单极中断

本次课程设计中，耗时最长的是气泡流水线。

本次课设还有诸多不足，例如未完成多级中断、流水中断、动态分支预测，以及未完成 FPGA 的设计。

5.1.2 团队任务总结

1. 项目内容，特色

本次团队任务主要运用 CPU，显示 bad apple（一首歌+视频），由于这个视频在二次元中很火，所以在自己的 CPU 上运行出来很有成就感。

2. 项目工作量

主要是四个方面，1.将视频转换成像素点阵，生成 txt 文件。2.根据 txt 文件，编写 JAVA 程序，生成 hex 文件。3.编写汇编程序，取数据。4.修改电路，使之可以显示 bad apple。

3. 遇到的问题

如何将视频转换成像素点阵，如何改造数据存储器使其有足够的存储空间来存放得到的视频文件，如何将内存中的文件逐步取出并显示在 32*32 显示器中

4. 选题原因

网上有这么一句话，有屏幕的地方就有 bad apple

5. 可行性分析

因为 bad apple 影绘为黑白视频，按照特定规则转换为像素点后即可显示在大部分显示器上

6. 用到的团队合作开发工具，特殊的开发工具

腾讯文档，git，logisim

7. 团队成员、分工

贺子杰：电路修改

梁一飞：编写汇编程序

胡晨风：编写 JAVA 程序

尹卯：视频文件分析提取

8. 个人总结

我在本次团队任务中，主要负责编写汇编程序取数据，使像素点阵可以正确显示，编写的汇编程序如下图所示：

```
1  .text
2  addi  $s1, $zero, 0        #初始值
3  addi  $s2, $zero, 8716
4  sll  $s2, $s2, 5
5  branch:
6  add $a0, $0, $s1
7  lw $a2, ($s1)
8  addi  $s1, $s1, 4
9  bne $s1, $s2, branch
10
11 addi    $v0, $zero, 10
12 syscall                    # 暂停或退出
13
```

图 5.1 汇编程序

数据一共 69728 行，除 4 后为 17432，由于数据太长不能用 addi 直接加载到寄存器里，故可以除 32 之后将寄存器里的内容左移 5 位。S2 里面存的是文件末尾地址，S1 存的是起始位置，通过对 S1 累加比较 S2，可以控制程序在正确的位置停止。由于一开始忽略了进制转换的问题，没有算好文件的末尾位置，导致视频放送的一半的时候突然停止，后续发现这个问题及时改正过来了。这次团队任务分工明确，安排合理，所以做起来难度不大，感谢同组成员的支持与配合。

5.2 课设心得

这次课程设计应该是除编译原理实验外最难的实验了，任务内容多，实现难度大。但好在指导充分，看过 MOOC 的课程后，我就有了大致的思路，剩下的查阅资料，多次试错也就能完成任务。

在本次课程设计中，我遇到的最大的困难是气泡流水线的实现，并不是因为分支检测以及数据相关检测难以实现，而是因为气泡流水线基于理想流水线，在测试中很多理想流水线中没有发现的隐疾在气泡流水线中一一体现出来，往往让人摸不着头脑，不清楚错在哪里。经过与同学反复讨论以及与正确结果逐拍对比，一些小问题很快被解决，但也有一些问题让人很难解决，比如在 4.2.3 中提到的，在与同学的对比中，发现在第 3000 多拍出现问题，这要是逐拍对比寻找问题是很难的，经排查发现是存储器里面存储的数据出现了问题，于是我盯住存储器，运用算法的二分法思想，先把拍数分为前 1500 拍以及后 1500 拍，这样逐步分下去，很快就找到是第 16 拍的时候出现的问题，进而找到是 JAL 指令的问题。这种实践中悟出来的寻找问题，解决问题的方法让我进步很大，以后在学习生活中一定可以用到。

本次课程设计我觉得组原群很有用，许多问题在群里提出来很快就能得到老师以及同学的解答，这节省了很多时间，同时我也觉得任务量有点大，可能是我比较笨的原因，在规定的时间内只能完成到单级中断，无法完成全部任务。

同时很感谢老师耐心的解答以及同学的帮助，虽然没有做到完美，但通过这次课程设计我也收获了很多，巩固了所学知识，以及掌握了许多排查问题，解决问题的方法。

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 4 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] 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.

[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.

[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.

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: