华中科技大学

2021

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：	CS1807
学　　号：	U201814745
姓　　名：	朱槐志
电　　话：	15623851632
邮　　件：	2607840393@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 27 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	
13	NOR	或非	

#	指令助记符	简单功能描述	备注
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	SUBU	无符号减	差异化指令 1
29	XORI	异或立即数	差异化指令 2
30	LBU	加载字节(无符号)	差异化指令 3
31	BLEZ	小于等于 0 转移	差异化指令

2 总体方案设计

2.1 单周期 CPU 设计

本阶段实现基于硬布线控制器的 MIPS 单周期 CPU，支持基本 24 条指令和 4 条扩 展指令（中断相关指令在后续阶段完成），总体结构图如图 2.1 所示。

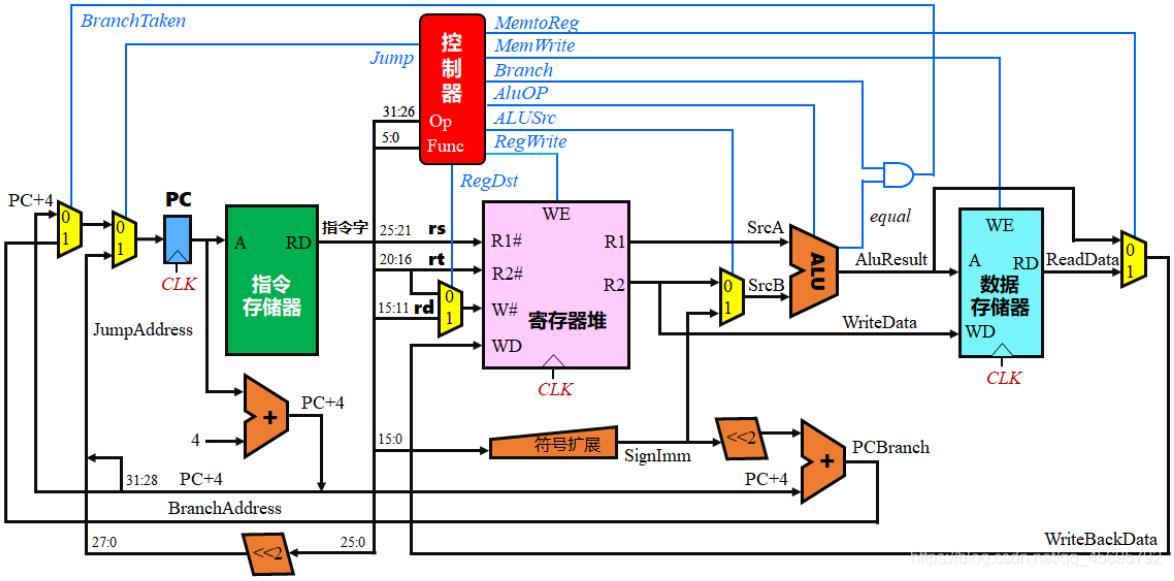

图 2.1 总体结构图

2.1.1 主要功能部件

主要用到的功能部件有：程序计数器 PC、指令存储器 IM、运算器、寄存器堆 RF、数据存储器 DM 以及符号扩展模块及数据选择器等具体设计思路如下：

1. 程序计数器 PC

PC 里面存储着下一条指令在指令存储器中的地址。输入为：下一条指令在 IM 中的地址、控制 PC 是否跳转的使能信号、时钟和复位信号；输出为取址阶段的指令地址，有两种情况，一是正常指令执行顺序的地址，二是跳转指令的目标地址，取决于选择器的输入信号。PC 采用寄存器实现，时钟触发为上升沿触发。

2. 指令存储器 IM

用于存储二进制形式的 MIPS 指令。输入为指令的地址；输出为 32 位二进制 MIPS 指令。使用 ROM 器件实现指令存储器，指令存储内容自行设定。取指令时根据 PC 寄存器的输出来进行指令的读取，一条指令占 4 个字节，需要将读取的 PC 的值去掉末两位再取第 2-11 位作为地址进行读取。

3. 运算器

运算器能根据 OP 字段的值对引脚数据进行不同的运算，包括位运算，移位运算，证书算术运算。

表 2.2 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
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
OF	输出	1	有符号加减溢出标记，其他操作为零
UOF	输出	1	无符号加减溢出标记，其他操作为零
Equal	输出	1	Equal=(x==y)?1:0, 对所有操作有效

4. 寄存器堆 RF

寄存器堆采用 MIPS 自带的寄存器实现，包含 32 个通用寄存器，该寄存器堆的输入输出引脚以及相应功能的描述如表 2.3 所示。

表 2.3 算术逻辑运算单元引脚与功能描述

引脚	功能描述
R1#	读寄存器 1 的编号
R2#	读寄存器 2 的编号

W#	写寄存器的编号
Din	写入的数据
WE	写使能信号
CLK	时钟信号
R1	寄存器 R1 中的数据
R2	寄存器 R2 中的数据

5. 控制器

采用硬布线控制，通过解析指令得到的控制信号，再通过组合逻辑生成相应的控制信号。

2.1.2 数据通路的设计

数据通路主要分为控制器通路、指令存储器通路、ALU 计算通路和数据存储器通路四个部分。下面从四个方面介绍具体的设计思路。

1. 三种指令的数据通路

MIPS 有 R、I、J 型三种指令，这些指令的执行需要用到公共的器件如 PC、ALU 等，一般我们更关注其执行过程的不同的地方。对于不同之处，为了完成各自的功能，我们需要设计专门的数据通路，比如为指令跳转生成地址计算的数据通路等。由于我们实现的是单周期 CPU，所以需要在关键器件前加多路选择器以满足不同的数据来源需求。具体实现就是在 PC、寄存器文件、ALU 等部件前加入多路选择器，选择信号由控制器发出。

2. 跳转指令

跳转指令分为条件跳转和无条件跳转。前者为 I 型指令，包括 BEQ、BNE 和 BLEZ7，后者为 J 型指令，包括 JMP、JR 和 JAL。他们的数据通路只需在已有通路的基础上加入地址计算与地址选择的部分即可。

3. 四条拓展指令

拓展指令分别为 SUBU、XORI、BLEZ 和 LBU。前两种属于 R 型指令，R 型数据通路是通用的；为实现 BLEZ 指令，我们用额外的比较器比较 RS 和 0 的大小以确定是否跳转。至于 XORI 指令我们只需要给出相应的控制信号，ALU 本身有异或运算的功能，只要给出正确的信号，指令即可正确工作。

4. 关于 Syscall 指令

该指令能完成简易系统调用的功能，需要设计专门的通路完成 LED 显示、停机等功能。

2.1.3 控制器的设计

首先需要对于查询 mips 指令集，获取指令详细信息，分析指令执行过程中需要的控制信号，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 2.5。

表 2.4 主控制器控制信号的作用说明

控制信号	取值	说明
AluOP	0-12	ALU 功能选择信号，0 为逻辑左移，1 为算术右移，…
MemToReg	0/1	控制寄存器写数据来源
MemWrite	0/1	Mem 写使能
AluSrc	0/1	控制 ALU 的输入端 B 的来源
RegWrite	0/1	寄存器组写使能
Syscall	0/1	Syscall 指令的判断
Signedext	0/1	符号扩展
RegDst	0/1	寄存器写控制
BEQ	0/1	BEQ 指令信号
BNE	0/1	BNE 指令信号
JR	0/1	JR 指令信号
JMP	0/1	JMP 指令信号
JAL	0/1	JAL 指令信号
BLEZ	0/1	BLEZ 指令信号
LBU	0/1	LBU 指令信号

#	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	BLEZ	LBU	LUI	R1_used	R2_used	X
1	SLL	0	0	0				1			1										1	
2	SRA	0	3	1				1			1										1	
3	SRL	0	2	2				1			1										1	
4	ADD	0	32	5				1			1									1	1	
5	ADDU	0	33	5				1			1									1	1	
6	SUB	0	34	6				1			1									1	1	
7	AND	0	36	7				1			1									1	1	
8	OR	0	37	8				1			1									1	1	
9	NOR	0	39	10				1			1									1	1	
10	SLT	0	42	11				1			1									1	1	
11	SLTU	0	43	12				1			1									1	1	
12	JR	0	8											1						1		
13	SYSCALL	0	12						1											1	1	
14	J	2	X												1							
15	JAL	3	X					1								1						
16	BEQ	4	X							1		1								1	1	
17	BNE	5	X							1			1							1	1	
18	ADDI	8	X	5			1	1		1										1		
19	ANDI	12	X	7			1	1												1		
20	ADDIU	9	X	5			1	1		1										1		
21	SLTI	10	X	11			1	1		1										1		
22	ORI	13	X	8			1	1												1		
23	LW	35	X	5	1		1	1		1										1		
24	SW	43	X	5		1	1			1										1	1	
25	SUBU	0	35	6				1			1									1	1	
26	XORI	14	X	9			1	1												1		
27	LBU	36	X	5	1		1	1		1								1		1		
28	BLEZ	6	X							1							1			1		
29																						

图 2.5 指令信号表

2.2 中断机制设计

2.2.1 总体设计

对于单级中断，采用一个寄存器 EPC 保留返回地址，然后用一个 D 触发器实现中断使能信号 IE。在指令方面需要增加对 ERET 指令的支持。在指令的上升沿后，检测到中断请求，立刻执行中断隐指令：将 PC 的值保存到 RET_PC 中，将中断使能 IE 置为 0，并通过该中断号对应的入口地址 INT_PC8 取得该中断程序的第一条指令，这样下一个时钟上升沿就开始执行中断服务程序了。执行到中断程序的 ERET 指令时，把返回地址 RET_PC 置于 PC 中，然后清空该中断的请求信号，将 IE 的值置为 1，这些都是同步 在 ERET 的上升沿的。所有这些完成后，CPU 开始执行被中断的主程序。

对于多级中断，需要考虑中断程序响应优先级更高的中断信号的情况。我们需要增加电路比较新的中断和当前中断程序的优先级，根据比较的结果来决定是否需要进入新的中断程序。我们还需要设计硬件栈来保存返回地址，来支持嵌套中断。还需要考虑进入中断服务程序后保护现场和恢复现场，在进入中断后关中断，保护现场后用指令 MFC0 开中断；在恢复现场前用指令 MTC0 关中断，执行指令 ERET 时同步开中断。

2.2.2 硬件设计

需要设计电路来检测和保存所有的中断请求，判断并执行优先级最高的中断并保留其中断号，这样就能在返回时清空对应的中断信息，这些功能可以利用优先编码、优先解码器和寄存器来实现；还需要设计电路让 CPU 能够在主程序和中断程序之间来回切换，这些功能依赖中断请求信号 INT、中断使能信号 IE 和返回指令 ERET，采用寄存器等器件来完成。

为比较新的中断请求和当前运行的中断程序的优先级，需要设计电路保存正在运行程序的中断程序号，可以用三个寄存器来保存三个中断程序的运行状态。用硬件栈保存返回地址，在响应新的中断前保存当前程序的返回地址，在中断程序执行 ERET

时同步写入返回地址到寄存器。

2.2.3 软件设计

对于单级中断，软件方面只需增加对 ERET 指令的支持，控制器如果检测到该指令，会给出“ERET” 的信号，其他中断相关电路能根据给出的信号完成中断返回的工作。

对于多级中断，需要新增加对指令 MFC0 和 MTC0 指令的支持，这是软件开关中断的关键。实现在控制器检测到这两条指令的时候，能够给出“MFC0”和“MTC0”的信号，以便多级中断电路能及时开关中断。编写测试程序时也应注意，保护现场后执行 MFC0 开中断，恢复现场之前执行 MTC0 指令关中断。

2.3 流水 CPU 设计

2.3.1 总体设计

MIPS 指令执行的过程分为 5 个阶段：取指（IF）、译码（ID）、

执行（EX）、访存（MEM）、写回（WB）。在其中插入流水接口部件完成相应信号的锁存和传递，从而形成一定程度的隔离，达到使不同阶段能够处理不同的指令的目的。要考虑到后续气泡流水线和重定向流水线对流水接口的要求，预留一定的空间以实现较好的可扩展性。在 EX 段针对跳转指令需要新增电路实现输入 PC 端的地址的判断和选择。

2.3.2 理想流水线设计

在此之前需要设计好流水接口部件，需要它能完成以下的功能：当使能信号为高电平时，从所有的数据输入端读入数据然后将其缓存在寄存器中，数据输出端的值和寄存器保持一致；当使能为低电平则不操作。当清零信号为高电平则进行同步清零；清零信号为低电平则不操作。除去每个接口都要传递的 PC 和 IM 值外，各个流水接口中需要保存传递的具体数据在表 2.6 中进行了说明。

表 2.6 各流水部件保存的信号

流水部	保存的信号

件	
IF/ID	IR,PC,In,Out,Clk,Stall
ID/EX	IR,PC,A,B,Imm,Address,RtForward,RsForward,WB,W#,MemW,EX,AluOp
EX/MEM	IR,PC,B,Imm,ALU,LedData,WB,W#,MemW,Show
MEM/WB	IR,PC,B,MD,ALU,LedData,WB,W#

完成流水接口的设计后，理想流水线的设计就比较清楚了，只需要考虑数据通路的实现，再将信号连接到对应的模块端口就行了。不过，为了体现流水设计的特点，数据应该从本阶段的上一流水接口中取出，而不能够从其它地方取出。比较方便的地方在于，单周期 CPU 中几乎所有的部件都可以在流水线中复用。对于 syscall 指令而言，既可以在 EX 阶段处理也可以在 WB 阶段处理。在本次设计中，在 EX 阶段计算出结果之后，需要将信号传递到 WB 段再进行停机信号的处理。

2.4 气泡式流水线设计

2.4.1 总体设计

气泡流水线用流水阻塞和插入气泡的方式解决分支相关和数据相关。对于结构相关，采用哈佛结构的解决方案，这样能解决取指令和取数据的争用主存问题，另外使用增加 PC、分支地址计算的加法器的方案，来解决 ALU 争用的问题。 对于分支相关，因为分支在 IE 阶段执行，只需我们在 IE 阶段检测到成功的分支信号后在 IF/ID 和 ID/EX 同步插入气泡即可。对于数据相关，如果是 ID 阶段与 WB 阶段相关，则将寄存器设置为下降沿触发来解决 ID 阶段取操作数和 WB 阶段写回的相关；如果是 ID 阶段与 MEM 阶段 相关，则暂停 PC 和 IF/ID，在 ID/EX 插入一个气泡即可解决；如果是 ID 阶段与 EX 阶段 相关，则采取与 ID 与 MEM 相关时相同的解决方法，最后会变为 ID 阶段与 WB 阶段相关的问题。

2.4.2 硬件设计

需要在 EX 阶段增加跳转检测逻辑；在 ID 阶段增加数据相关检测逻辑，包括 ID 阶段当前使用的源寄存器和 EX、MEM 阶段使用的写寄存器的检测。

2.5 数据转发流水线设计

数据重定向流水线的设计是在气泡流水线的基础上完成的，主要需要增加 Load_Use 冲突的检测、数据重定向通路的设计，数据重定向是将数据可能的输入路径全部汇总，然后根据指令的执行情况决定从哪条路径上获取数据；如果相邻两条指令之间存在数据相关，并且前一条指令是访存指令的时候（称为 Load-Use 相关），此时不能采用重定向方式进行处理，即此时仍然需要使用插入气泡的方式来解决。

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）

使用 32 位寄存器实现程序计数器 PC，采用时钟上升沿触发，输入为下一条指令的地址，输出为当前执行指令的地址。Enable 信号为控制运行信号，根据此控制信号和时钟信号与操作的结果来判断是否停机。PC 输入端的选择有多种情况，需要根据不同的指令来进行选择，因此设计了多个多路选择器，PC 设计如图 3.1 所示。

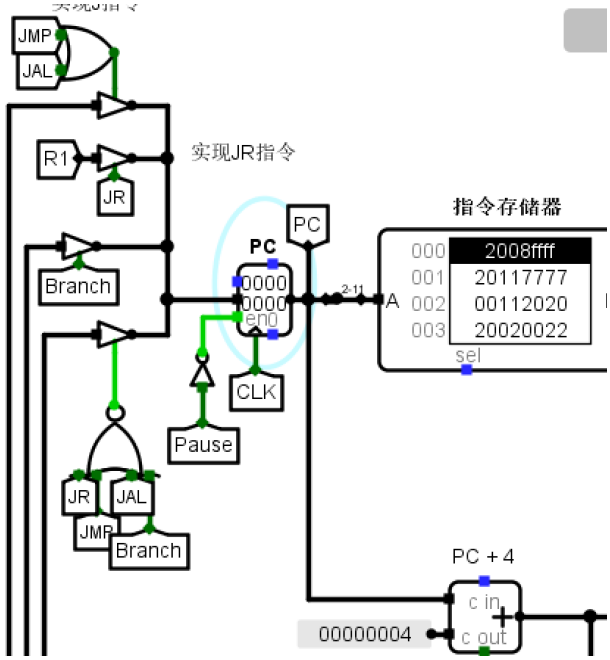

图 3.1 程序计数器（PC）

2) 指令存储器（IM）

使用只读存储器 ROM 实现指令存储器（IM）。将只读存储器的地址位宽设置为 10 ，数据位宽设为 32。因为 PC 中存储的指令地址宽度为 32 位，而 ROM 地址线宽度有限，仅为 10 位，所以需要将 32 位指令地址高位部分和字节偏移部分屏蔽掉，屏蔽方法为采用分线器取 32 位指令地址的 2-11 位作为指令存储器的输入。如图 3.2 所示。

图 3.2 指令存储器（IM）

3) 数据存储器（DM）

与指令存储器 IM 类似，数据寄存器使用 RAM 实现。地址位宽设置为 10 位，数据位宽设置为 32 位。由于计算出的地址为 32 位，而 RAM 地址线宽度有限，仅为 10 位，所以需要将 32 位指令地址高位部分和字节偏移部分屏蔽掉，同上，还是使用分线器，只取 32 位指令地址的 2-11 位作为指令存储器的输入。在完成 LB 指令的时候，还需要将数据进行划分并实现符号扩展。如图 3.3 所示。

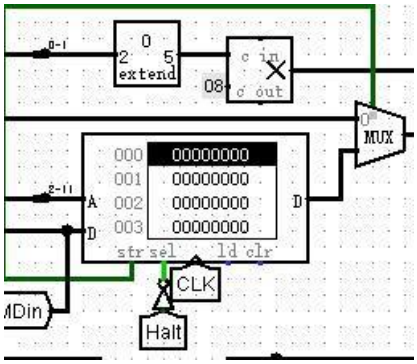

图 3.3 数据存储器（DM）

4) 寄存器堆（RF）

使用所给库 cs3410.jar 文件中实现好的 MIPS 寄存器组，把相应的端口连接好即可。不过需要注意的是，在连接 W#和 Din 信号时，需要考虑 JAL,RegDst,LB 等多种控制信号，采用多个多路选择器实现。如图 3.4 所示。

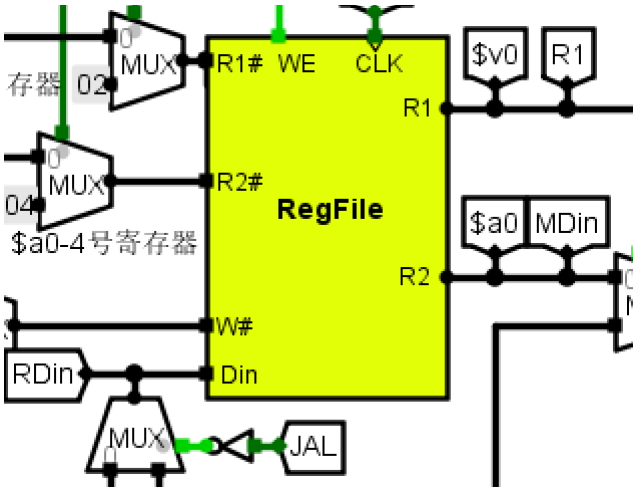

图 3.4 寄存器堆

3.1.2 数据通路的实现

在主要功能部件的设计和实现完成后，各个部件组合连接之后得到的结果如图 3.5 所示。

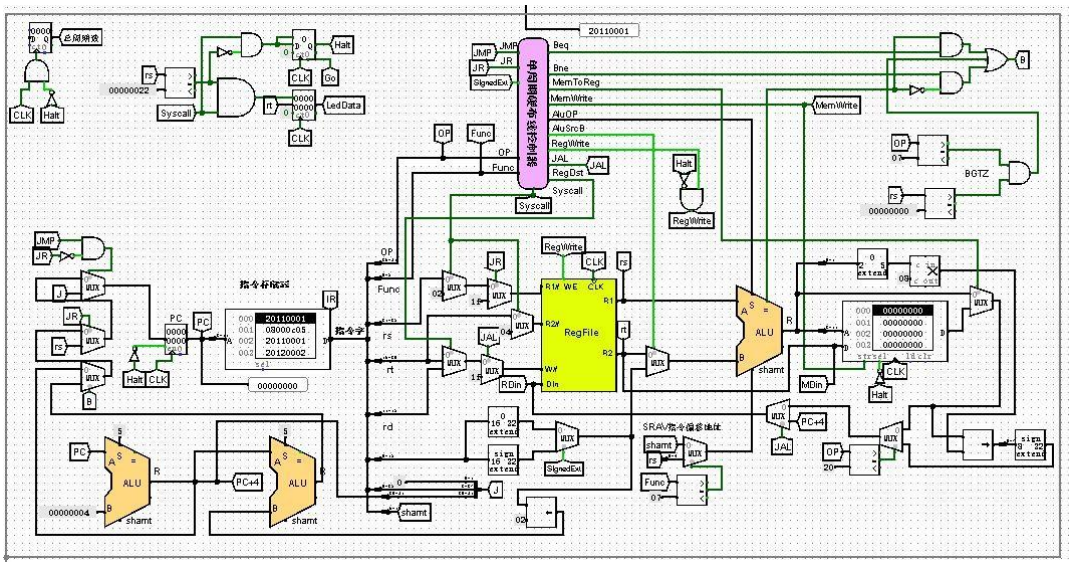

表 3.5 指令系统数据通路表

3.1.3 控制器的实现

根据填写好的控制信号信息表，得到相应指令的信号逻辑表达式，使用 Logisim 的逻辑分析功能完成控制器的实现，得到的控制器电路如图 3.6 所示。

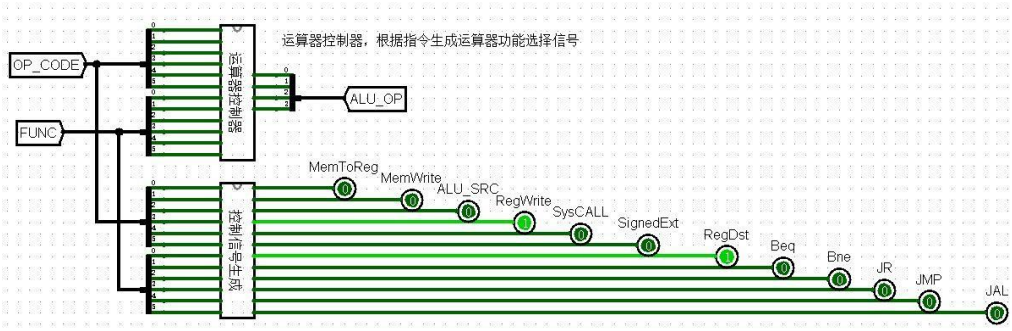

表 3.6 控制器实现

3.2 中断机制实现

3.2.1 单级中断

中断信号产生电路的设计如图 3.7 所示。

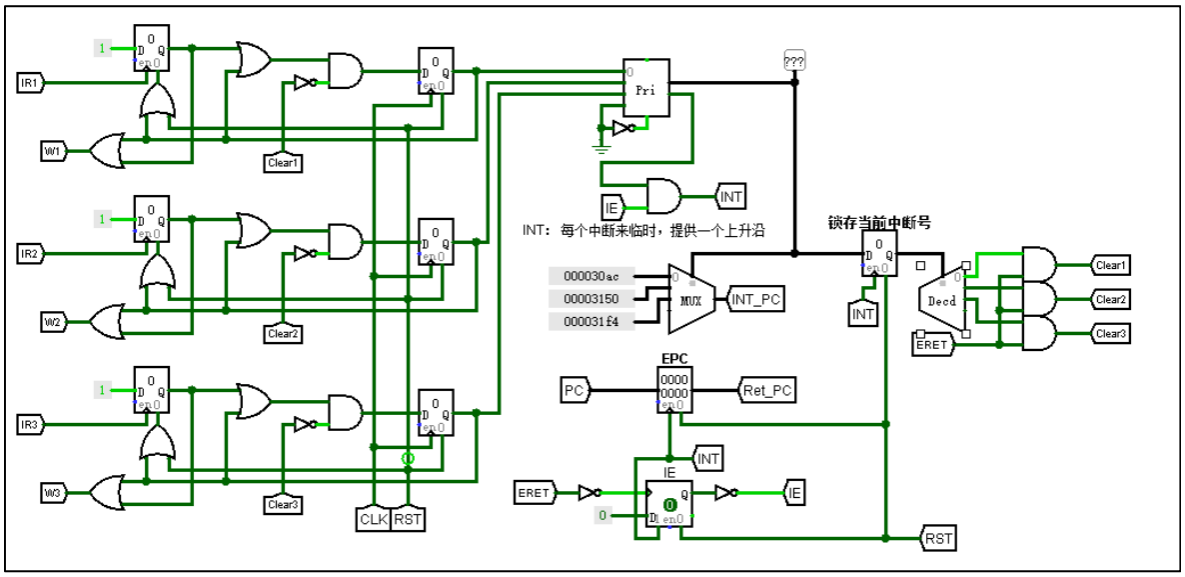

表 3.7 中断信号产生电路

中断开关、断点保存以及中断服务程序选择电路的实现如图 3.8 所示。

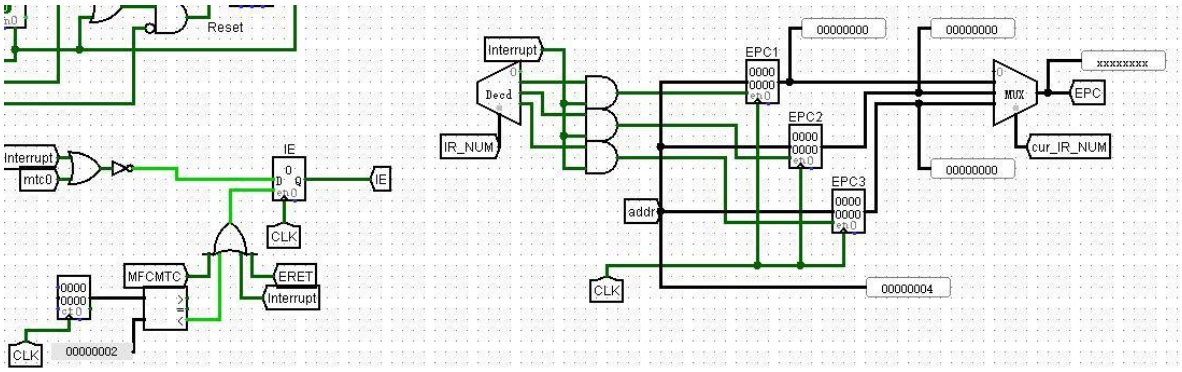

表 3.8 中断开关和断点保存

还需要对 PC 输入的选择进行调整，包括 ERET 信号进行选择，如图 3.9 所示

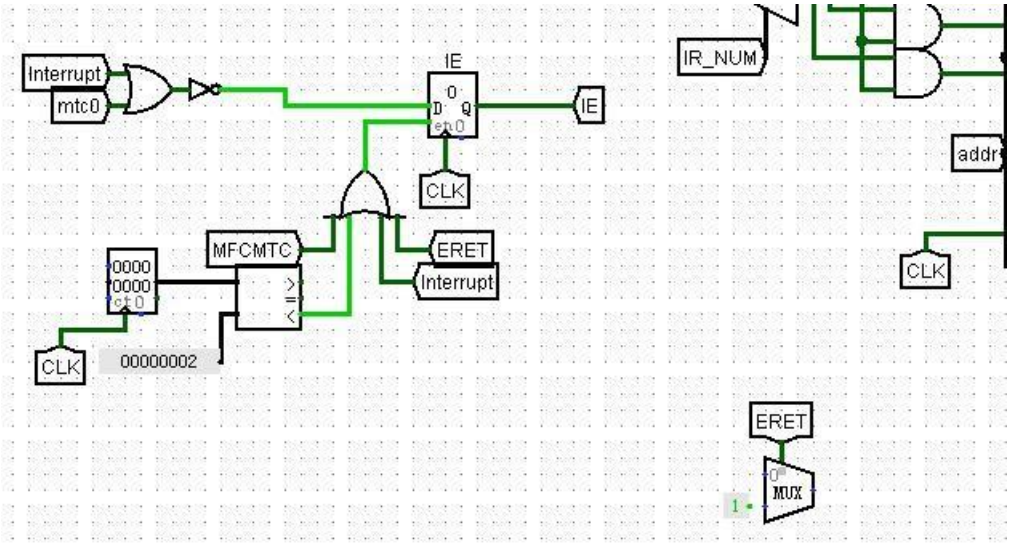

表 3.9 中断选择

3.2.2 多级中断

多级中断中断信号产生电路的设计与单级中断没有改变，COP0 寄存器的模拟如图 3.10 所示。

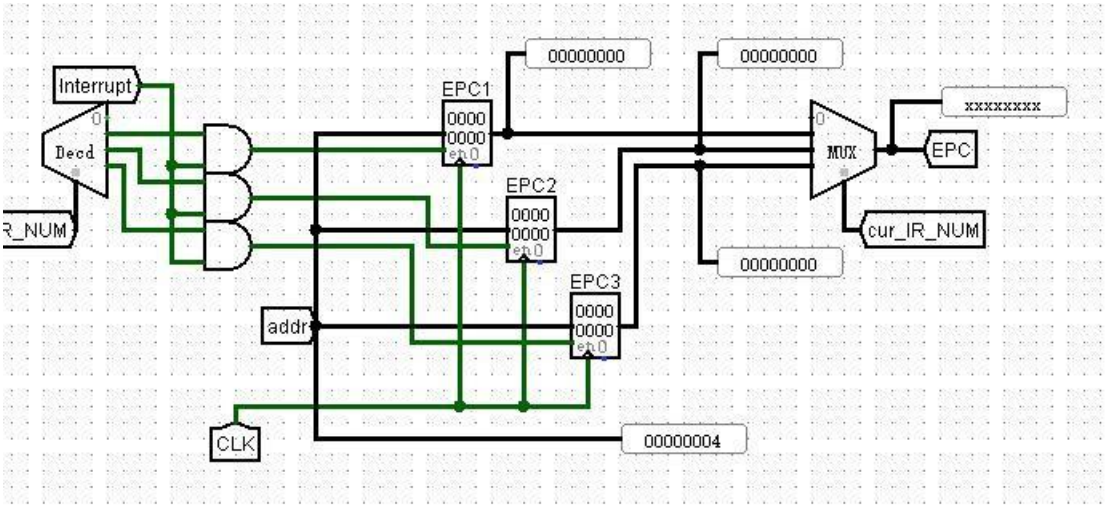

图 3.10 COP0 寄存器模拟

增加中断屏蔽字后的中断服务程序的入口地址处理如图 3.11 所示。

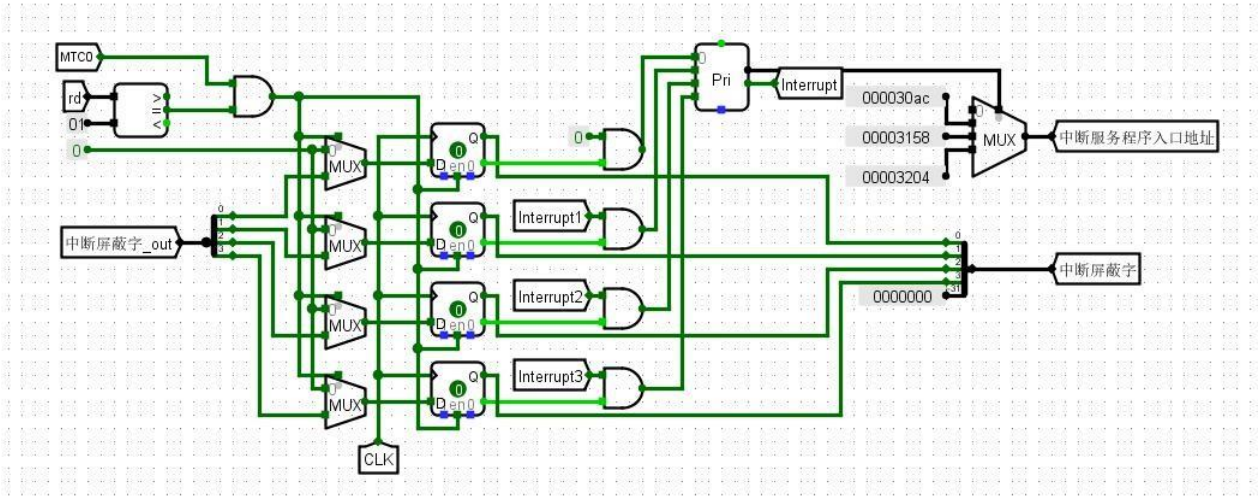

图 3.11 中断屏蔽字

此时需要根据 MFC0 指令和 MTC0 指令对寄存器堆进行相应数据的处理，要求能够根据指令的要求进行存取。具体实现见图 3.12.

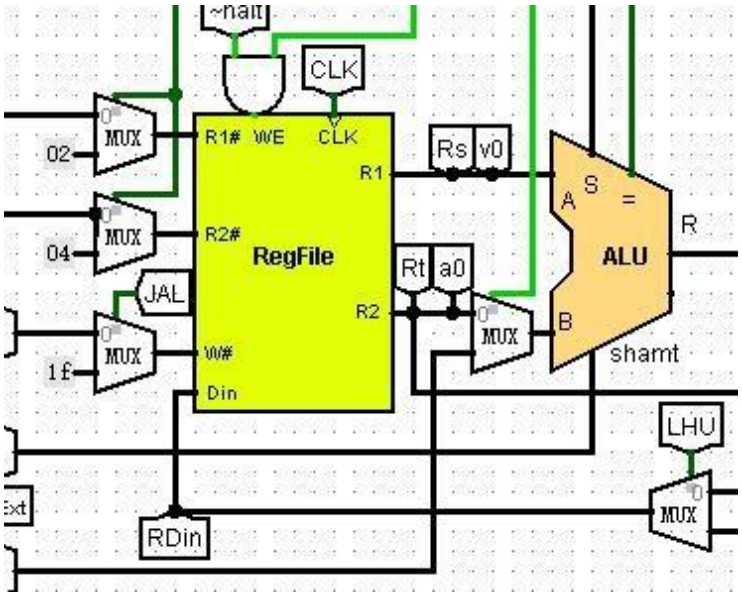

图 3.12 MFC0 和 MTC0

3.3 流水 CPU 实现

3.3.1 流水接口部件实现

流水接口部件使用寄存器实现信号的锁存，寄存器均采用上升沿触发，需要增加插入气泡的接口，即能够实现同步清零，电路见图 3.13 所示。

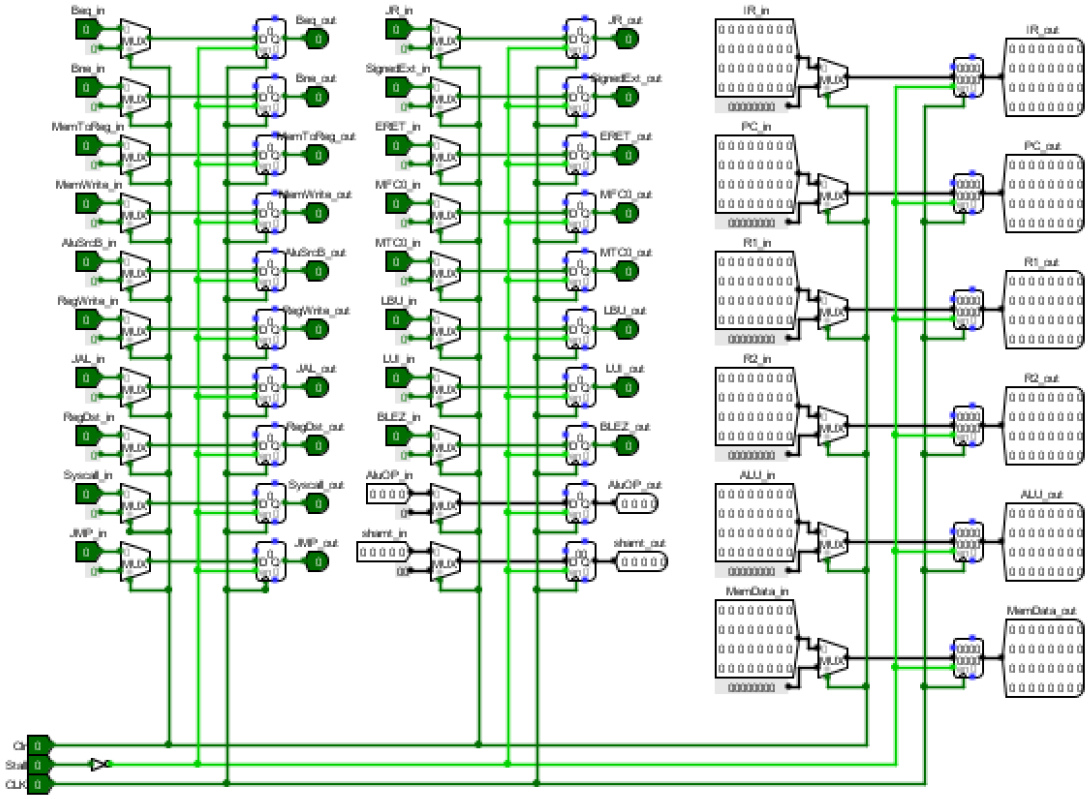

图 3.13 MEM/WB

3.3.2 理想流水线实现

理想流水线因为不存在分支指令和各种冲突，因此只需要完成控制信号的传递，不过注意的是需要将某些信号进行合并传递，如图 3.14 所示。

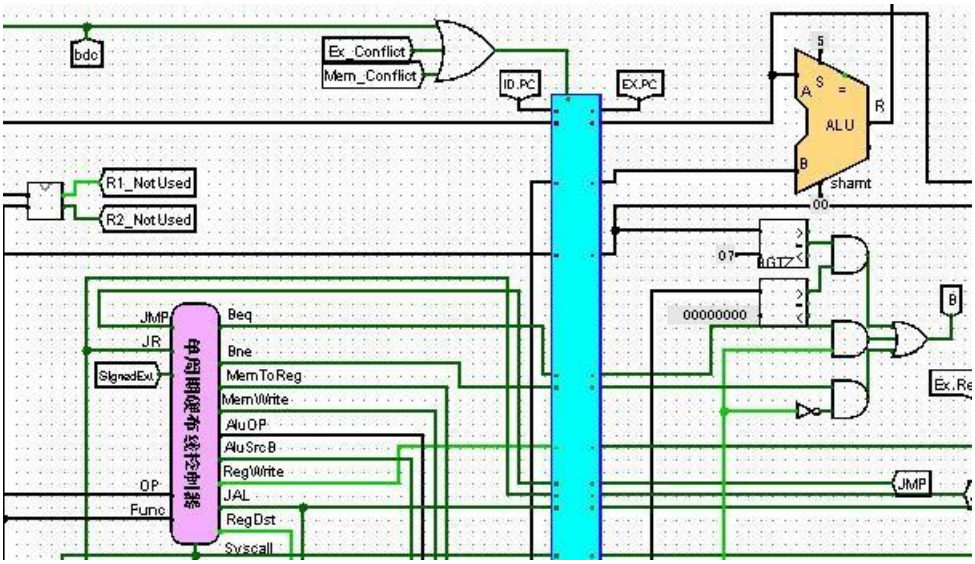

图 3.14 信号的合并传递

随后在需要时进行分线处理。数据通路如图 3.15 所示。

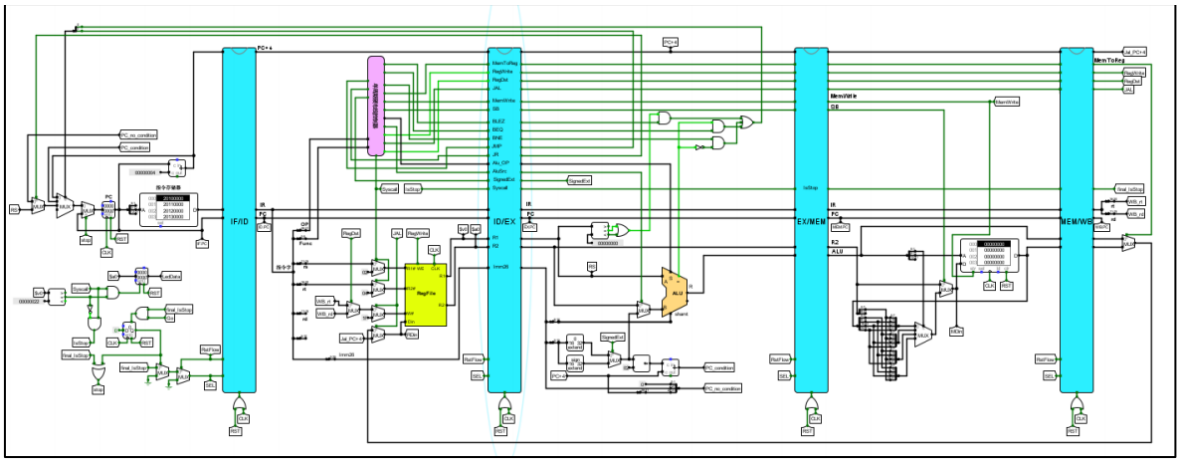

图 3.15 理想流水线

3.4 气泡式流水线实现

气泡流水线首先要实现数据相关的检测逻辑，具体的数据通路如图 3.16 所示。

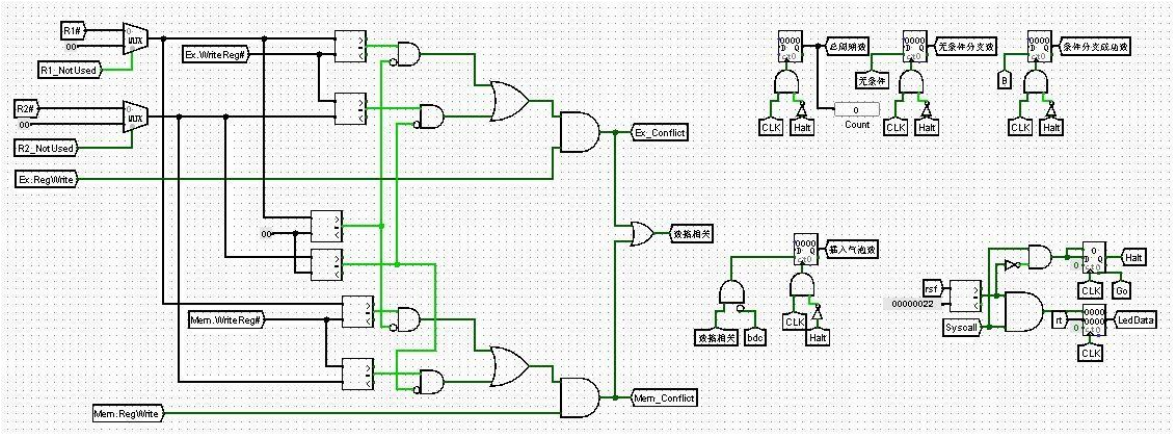

图 3.16 数据相关检测

如果出现数据相关或分支指令的情况，需要插入气泡，气泡插入逻辑电路如图 3.17 所示。

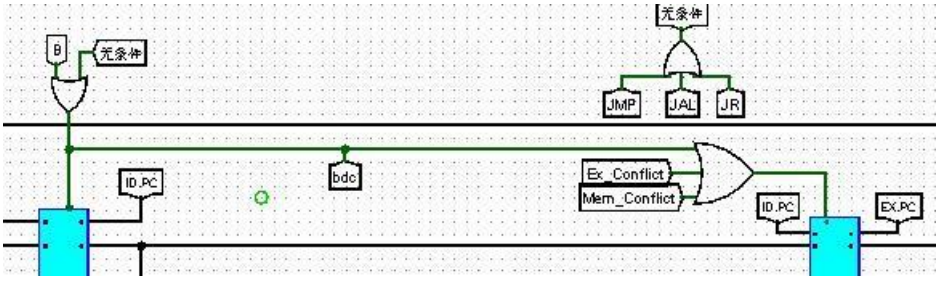

图 3.17 气泡插入

另外如果出现 Load_Use 相关，还应该让 PC 暂停取指令，电路实现如图 3.18 所示。

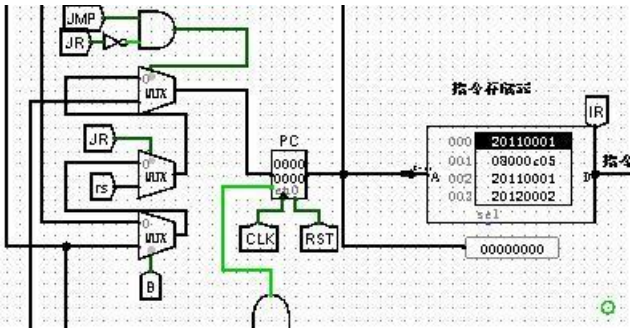

图 3.18 暂停 PC

整个气泡流水线的数据通路如图 3.19 所示。

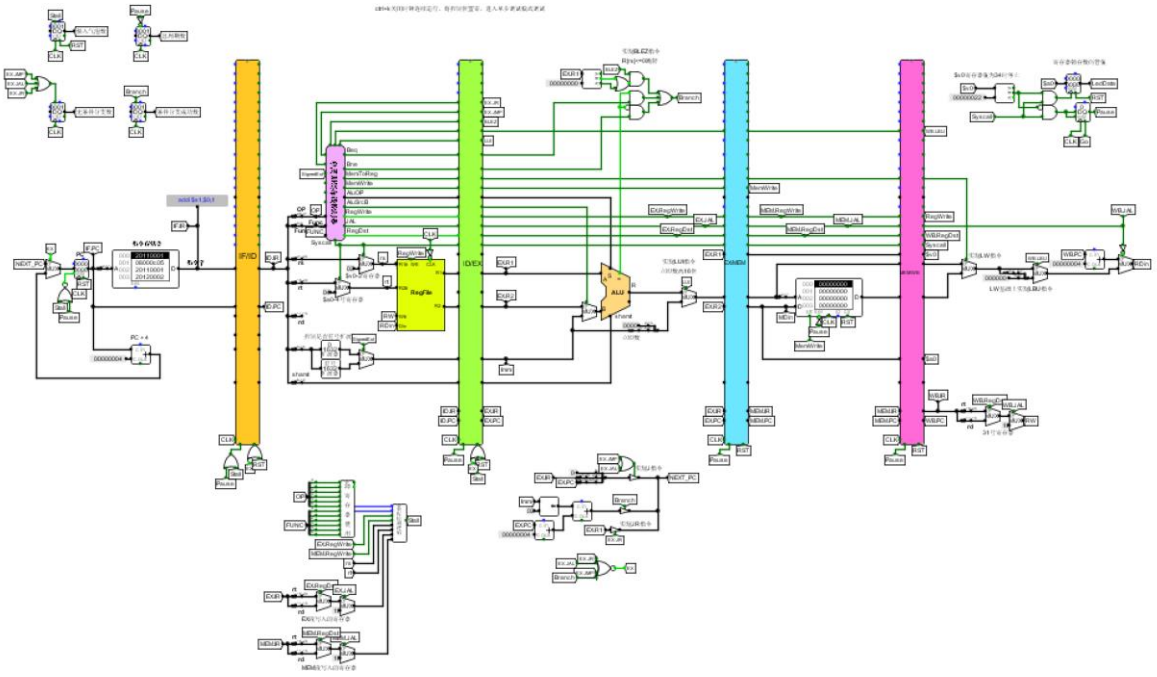

图 3.19 气泡流水线

3.5 数据转发流水线实现

3.5.1 总体电路

重定向流水线总体电路如下图 3.20 所示。

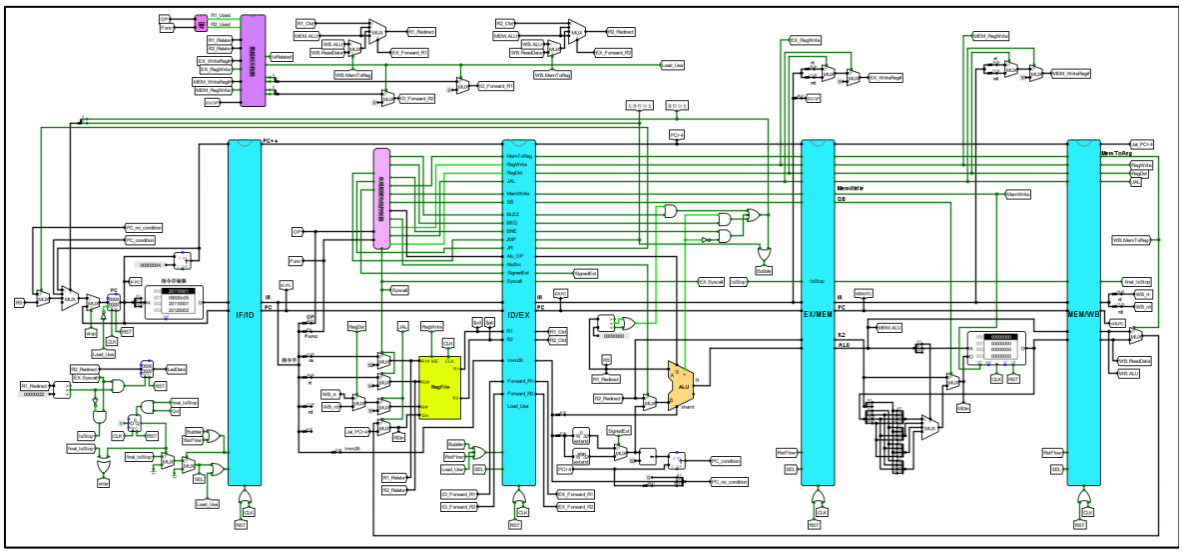

图 3. 20 重定向流水总体电路

3.5.2 EX 重定向电路

在 EX 阶段完成所有的重定向工作，相关电路如下图 3.21 所示。

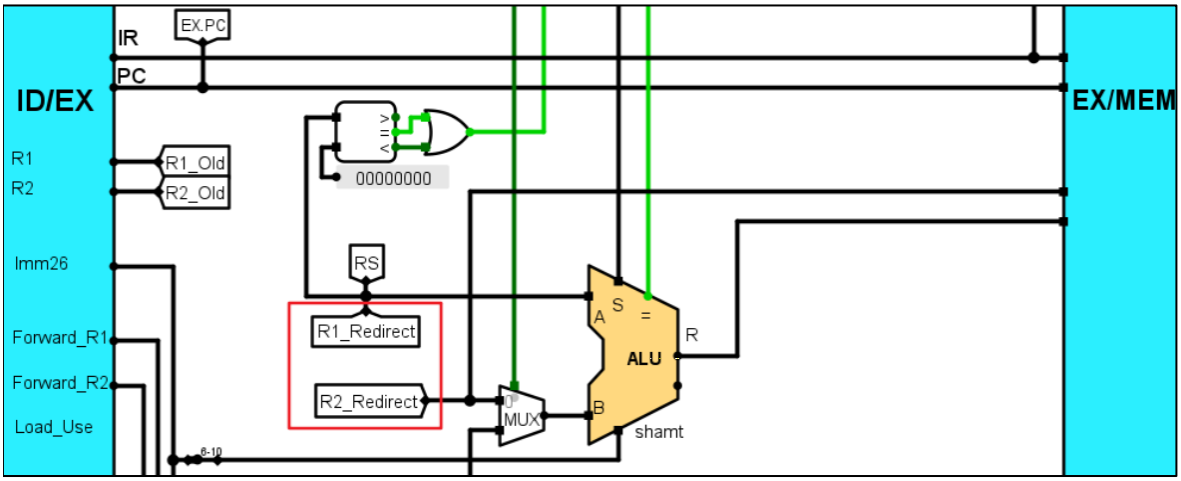

图 3.21 EX 重定向电路

3.5.3 重定向选择信号的传递

将 ID 阶段的重定向选择信号传递到 EX 阶段，如下图所示。

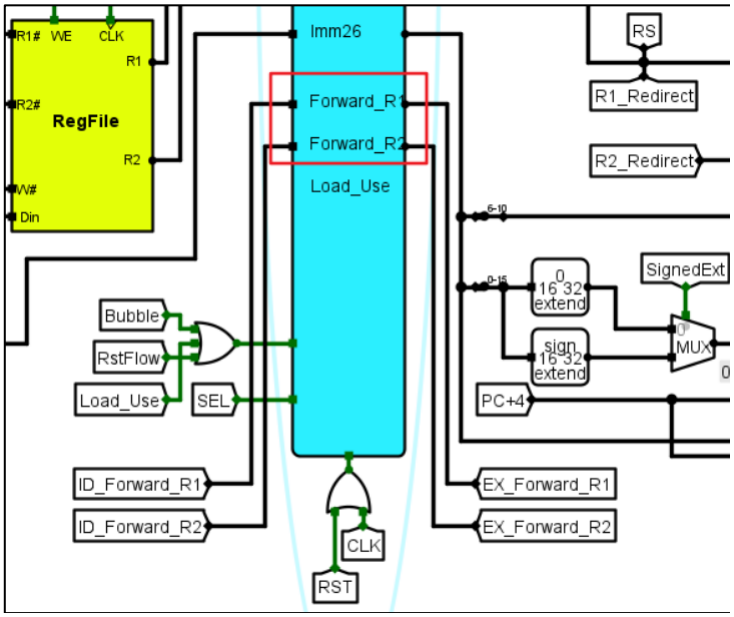

图 3.22 重定向选择信号的传递

ID_Forward_R1 是 ID 阶段为 R1 生成的重定向选择信号，通过流水接口传递到 EX 段。R2 同理。

3.6 团队任务

3.6.1 工作量与团队成员分工

- 顾时禹　框架设计，关卡中断
- 吕毅东　地图设计及代码实现
- 朱槐志　上下左右中断
- 王祺霖　CPU 新增中断
- 师宇哲　显示逻辑

3.6.2 项目内容

小组实现一个迷宫游戏。在一个 32*32 的二维网格空间中（即上帝视角），玩家要操纵一个 Agent 从$(0,0)$走到$(31,31)$。在这个网格平面上，有颜色的方块代表墙壁，没有颜色的方块代表通路。迷宫游戏是一个考验参与者空间感、空间记忆力和搜索启发策略等多方面认知能力的综合测试，因此，我们的项目具有相当好的实用价值。在我们的实现中，玩家启动界面后会首先进入欢迎界面，有两个关卡可供选择。选择关卡后需要通过控制计算机的上、下、左、右键将 Agent 从右上角移动到左下角，若成功，则屏幕会显示 Success 这一字样。

3.6.3 项目总体设计

实现“走迷宫”的项目需要从硬件和软件两个方面进行，即对已经实现的单周期单级中断 CPU 加以适当的改造，编写相关程序。

3.6.4 项目硬件设计

主控 CPU——负责游戏的主体逻辑。六路中断逻辑——四个中断是上下左右，两个中断是关卡选择。显示逻辑——包括 32 个 32 位寄存器，对应更新线路和 LED 点阵。实现的时候只 需要在将地图及人物数据写入内存的时候同时写入到备份的 32 个 32 位寄存器中（以便能在一个时钟周期内全部读出），需要更新 LED 点阵时调用 SYSCALL 指令，将备份寄存器中的数据更新到控制点阵输出的寄存器中即可。

3.6.5 项目软件设计

主控程序的总体逻辑：首先，载入欢迎界面，并提示 2 个关卡；然后进入空操作死循环等待关卡选择；当用户选择关卡（按 1 或 2 按钮）后，视情况进入2 个关卡中断服务程序中的一个，该程序会初始化屏幕界面，再次进入空操作死循环等待上下左右操作；如果有上下左右按键对应的中断，那就去执行相应的中断服务程序；上下左右中断服务程序的最后会检查主人公的位置是否是终点，如果是，就跳到游戏结束程序段游戏结束程序会在屏幕上显示成功提示。

4 实验过程与调试

4.1 主要故障与调试

4.1.1 多级中断中断信息显示故障

故障现象：多级中断电路无法正确保存历史中断号。

原因分析：在实现多级中断电路时，新中断产生时应该有相应的中断优先级判断逻辑，并且将当前中断号按顺序保存，简单地根据中断号的大小判断是无法满足要求的，需要根据 IE 状态进行使能处理，更新当前中断号，并且将历史中断号顺序保存。

解决方案：使用三个寄存器，将三级中断保存下来，将 IE 寄存器的状态和是否产生新中断信号作为使能端，更新历史记录，根据最近和当前中断号进行判断决定是否发生嵌套中断。如图 4.1 所示。

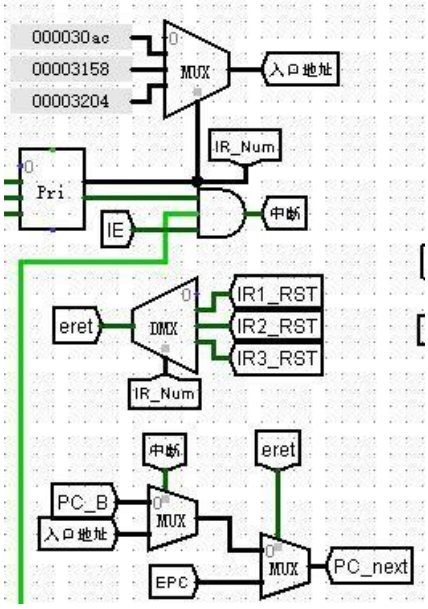

图 4.1 中断号历史记录实现

4.1.2 异或立即数指令

在做差异化指令的 XORI 指令的时候，因为运算器里已经有了异或运算的操作，所以我只需要把对应的信号传入正确，异或指令就会正确执行，不需要对该指令额外增加数据通路。

故障现象：将 XORI 指令的逻辑信号表达式复制到 logisim 里去之后。执行测试程序 XORI.hex，显示器却一直显示 7，不是期望的结果。

解决方案： 发现原来是因为修改表达式的值的时候没有点输入的按键，导致信号逻辑表达式没改变。

4.1.3 重定向流水线故障

故障现象：重定向流水处理 Load_Use 相关时出现错误。

原因分析：重定向选择信号使用了 ID 阶段新产生的 EX_Load_Use，而不是传递到 EX 阶段的 EX_Load_Use，如下图所示。图 4.2 重定向流水 Load_Use 处理错误

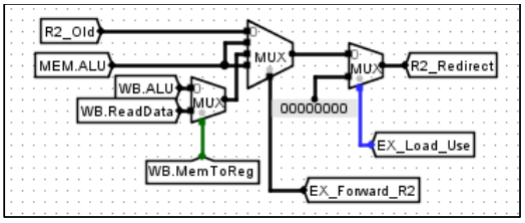

图 4.2 重定向流水线故障

解决方案：需要减少流水接口，在 ID 阶段优先考虑 Load_Use 信号，如果信号为 1，则直接将重定向选择信号置为 0，即不实行重定向，下一周期直接在 ID/EX 阶段插入气泡。

4.2 实验进度

表 4.3 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，完成了控制器
第三天	实现了单周期 CPU 并且通过了测试。并复习中断知识
第四天	Educoder 上通过了单级中断
第五天	继续完成多级中断电路
第六天	Educoder 上通过了多级中断关卡
第七天	复习关于指令流水线的知识点，修改单周期 mips 电路
第八天	完成的理想流水线电路顺利通过了 Educoder 关卡
第九天	完成气泡流水线电路并通过 Educoder 测试
第十天	完成数据重定向电路
第十一天	给控制信号表增加差异化指令部分的信号，重新生成硬布线控制器电路
第十二天	在重定向电路基础上给差异化指令增加电路
第十三天	修改完成的电路成功跑出预期的结果

5 设计总结与心得

5.1 课设总结

1）完成方案总结：实现了单周期 CPU、理想流水线 CPU、气泡流水线 CPU、重定向流水线 CPU、支持单级中断和多级中断的单周期 CPU、以及扩展设计了 CCAB 四条指令的重定向流水线 CPU。

2） 功能总结：单周期和流水线 CPU 都支持包括 CCMB 指令在内的 31 条指令，理想流水线主要工作是流水接口的设计、流水线的分割和停机、显示、清空流水等功能的实现，气泡流水线用流水阻塞和插入气泡等方法解决各种冲突，重定向流水线在气泡流水线的基础上加入旁路机制，关键是搞清重定向的数据来源和重定向选择信号的产生，单周期实现了单级中断和三级可嵌套外部中断。

3） 团队任务总结：与同组同学一起完成了团队任务的选题、设计、具体构建，自己实现了迷宫移动的中断实现，一起完成了团队任务。

5.2 课设心得

这次课设虽然在寒假期间就布置了，但是自己放假没有做，不过还好开学有一个操作系统课设的缓冲期。刚开始的时候因为上学期的组原知识基本都忘光了，所以花了一段时间看书复习 CPU 部分的知识。单周期 CPU 开始还是没什么问题。做到流水线时碰到了一些问题，主要还是流水部件的设计以及流水线各阶段数据传递的问题，包括后续的重定向流水线中，都需要注意进行数据相关检测的输入指令是哪个阶段的指令，这都需要对流水线的充分理解。做中断的时候其实看完 mooc 思路还很混乱，没有头绪，后来请教了一下同学，之后才断断续续的完成，其中多级中断难度相对较大，不过最终也顺利完成了。

这次课程设计要求我们能很好地掌握 CPU 各个部件运行原理并且用 logisim 来实现。很考验我们的动手能力。感觉组原课设正是组原这门课的精髓，它让我们能把学到的知识投以运用，极大锻炼了我们的设计能力和解决问题能力。经历了本次课程设计，感觉自己的对组成原理的理解有了进一步提升。

最后由衷感谢课程组老师在组原群对我们的耐心指导和包容！

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] 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] 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.

[5] 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.

[6] 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.

[7] 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：朱槐志