

华中科技大学

2022

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: CS1903

学 号: U201914995

姓 名: 罗虞阳

电 话: 13713561390

邮 件: 1375081937@qq.com

目 录

1	课程设计概述.....	3
1.1	课设目的	3
1.2	设计任务	3
1.3	设计要求	3
1.4	技术指标	4
2	总体方案设计.....	6
2.1	单周期 CPU 设计	6
2.2	单级中断机制设计.....	9
2.3	多级中断机制设计.....	10
2.4	理想流水 CPU 设计	11
2.5	气泡式流水线设计.....	12
2.6	重定向流水线设计.....	13
2.7	动态分支预测机制.....	13
2.8	重定向单级中断.....	13
2.9	团队任务设计.....	14
3	详细设计与实现	16
3.1	单周期 CPU 实现	16
3.2	中断机制实现.....	19
3.3	流水 CPU 实现.....	21
3.4	气泡式流水线实现.....	22
3.5	重定向流水线实现.....	24
3.6	动态分支预测机制实现	25
3.7	重定向单级中断.....	27
3.8	团队任务-硬件设计.....	28

华中科技大学课程设计报告

4	实验过程与调试.....	31
4.1	测试用例和功能测试.....	31
4.2	性能分析	35
4.3	主要故障与调试.....	35
4.4	实验进度	37
5	设计总结与心得.....	38
5.1	课设总结	38
5.2	课设心得	38
	参考文献.....	40

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及实现能力的培养”。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；

华中科技大学课程设计报告

- (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	

华中科技大学课程设计报告

#	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	
24	SYSCALL	系统调用	If \$v0==10 halt(停机指令) 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	SLL	逻辑左移	
29	SLTIU	无符号小于时置位	
30	LH	读半字	
31	BLT	小于则分支跳转	

2 总体方案设计

2.1 单周期 CPU 设计

单周期 CPU 本次我们采用的方案是硬布线控制方案，且实现了指令和数据分开存储的结构来完成方案的设计。在一个周期内，控制器根据读入的指令给出相应的控制信号来控制数据通路流通情况，同时其余各功能部件根据得到的控制信号完成相应功能的实现。控制信号通过 excel 表格工具完成控制器逻辑对指令译码的转换，在 logisim 平台中实现了各个部件的物理结构和数据通路连接。

总体结构图如图 2.1 所示。

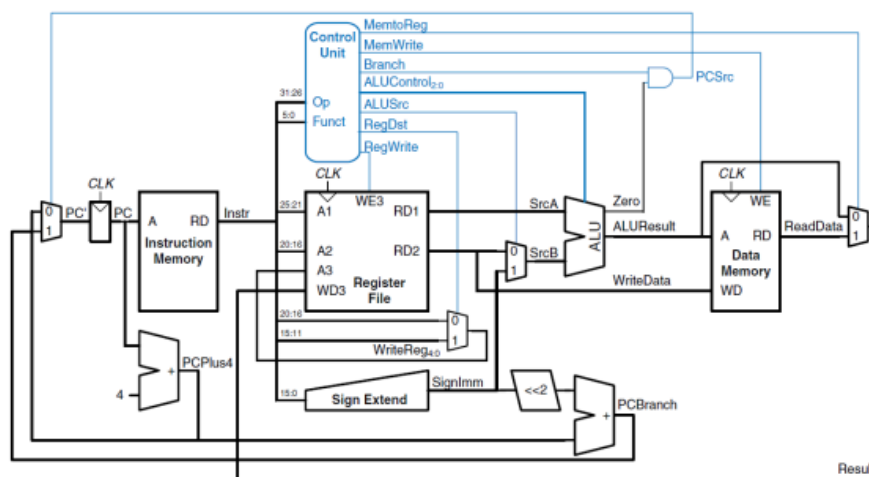


图 2.1 总体结构图

2.1.1 主要功能部件

主要使用到的功能部件包括程序计数器，指令存储器，数据存储器，运算器和寄存器堆部分，具体设计思路如下：

程序计数器 PC

程序计数器 PC 的设计思想为通过设置一个 32 位的寄存器用来存储下一条指令在指令存储器中的地址，使用时钟上升沿触发的方式进行 PC 指令的存取功能。程序计数器的输入为经过数据通路计算选择的下一条指令在指令存储器中的地址，控制

华中科技大学课程设计报告

PC 是否跳转到输入的使能信号，清零重置的复位信号和时钟信号，输出为取指令阶段的指令值。在 logisim 中使用时钟上升沿触发的寄存器作为程序计数器。

指令存储器 IM

由于设计的 CPU 为单周期模式，必须实现指令和数据的同时访问，所以需要将指令存储器和数据存储器分开。

指令寄存器 IM 用于存储二进制形式的 RISC-V 指令。输入为指令的存储地址；输出为二进制 RISC-V 指令。在 logisim 的具体实现中，使用 ROM 器件即可，注意，要进行指令的装载，需要首先手动将对应的指令集加载到该 ROM 中。在取指令时根据 PC 寄存器输出端口的值来进行指令的读取，一条指令占 4 个字节，所以需要将读取 PC 的值去掉末两位再取第 2-11 位作为地址进行读取。

运算器

运算器使用的是 CS3410 中已经封装好的寄存器，这里不做赘述。具体功能参考

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
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
<	输出	1	该输出的赋值为 $(x < y)?1:0$
>=	输出	1	该输出的赋值为 $(x \geq y)?1:0$
Equal	输出	1	$\text{Equal} = (x == y)?1:0$, 对所有操作有效

寄存器堆 RF

寄存器堆这里是直接使用了 CS3410 中已经封装好了的 Register File 部件。寄存器堆中含有 32 个 32 位寄存器，设置了两个并行的读入端口和一个写入端口，使

华中科技大学课程设计报告

用一个 5 位输入索引到寄存器中的相应位置。

2.1.2 数据通路的设计

在这次课设中，我设计的数据通路主要分为以下四部分，包括控制器通路，指令存储通路，ALU 计算通路和数据存储器通路。以下位这四个部分的设计思路。

在 RISC-V 中有六种指令格式，分别识用于寄存器-寄存器操作的 R 型指令，用于短立即数和访存 load 操作的 I 型指令，用于访存 store 操作的 S 型指令，用于条件跳转的 B 类型指令，用于长立即数的 U 型指令和用于无条件跳转的 J 型指令。这些指令执行需要用到公共的部件包括 PC、Alu 等。为了在单周期 CPU 中完成相应的功能，我们在 PC、寄存器、ALU 前增加了多路选择器来选择输入的信号，而这些控制信号通过控制器发出。特别的，对于条件跳转和无条件跳转指令，需要在已有通路的基础上新增加新 PC 地址计算和地址选择功能电路。

接下来说明自己的 CCAB 指令设计。我的 ccab 指令是 SLL、SLTIU、LH 和 BLT，前两者属于 R 型指令，不需要更改当前数据通路。LH 指令是需要从内存中指定的加载地址处读取一个半字，然后符号拓展到 32 位，因此，需要新增多路选择器选取半字然后使用符号拓展器拓展到 32 位，同时需要使用 LH 信号控制写入寄存器中的值。BLT 指令为小于跳转，需要结合 ALU 部件 '<' 信号进行控制。

2.1.3 控制器的设计

控制器采用了硬布线控制器，利用控制信号自动生成表格实现。控制器根据指令的 OP、FUNC 字段，通过组合逻辑输出该指令执行所需要的所有控制信号，这些信号分为三类：多路选择器的选择信号，器件的写信号和 ALU 的功能选择信号。大部分信号的功能和产生条件在指导书已有详细说明，本报告不赘述，这里只是介绍自己增加的几个信号：

Jal: 判断当前指令是否为 Jal 指令，如果是则为 1，否则为 0；

Jalr: 判断当前指令是否为 Jalr 指令，如果是则为 1，否则为 0；

Sll: 判断当前指令是否为 Sll 指令，如果是则为 1，否则为 0；

Sltiu: 判断当前指令是否为 Sltiu 指令，如果是则为 1，否则为 0；

Lh: 判断当前指令是否为 Lh 指令，如果是则为 1，否则为 0；

Bltn: 判断当前指令是否为 Bltn 指令，如果是则为 1，否则为 0；

2.2 单级中断机制设计

2.2.1 总体设计

为了实现单级中断，在单周期 CPU 的基础上需要增加一个中断使能信号 MIE 判断当前是否产生了中断和一个寄存器 mEPC 用来保存返回断点，且数据通路需要支持中断返回指令 uret 的处理，同时需要保存中断跳转的 PC 和返回处 PC 的值。

在一条指令的上升沿后，检测到中断请求，立即执行中断隐指令，将当前要执行的 PC 值保存到 retPC 中，置中断使能位 0，并且通过获取 intPC 值取下一条指令。

2.2.2 硬件设计

硬件设计中，当电路中产生了中断信号时，使用两个 D-触发器将信号锁存，直到下一次对应的中断清除信号到来时将触发器归零。由于在单级中断中存在中断优先级判断，仅仅需要处理当前中断中最高优先级中断，才取得方法是将所有锁存中断请求信号放入到优先编码器中，如果优先编码器中同时存在多个不同等级中断请求信号，优先编码器仅输出编号最大的输入端口，即仅处理优先级最大的中断请求信号。优先编码器输出选择应处理的中断编号和中断指示，而后中断指示与中断使能信号 MIE 进行逻辑与操作，产生最终的中断信号 INT。

使用 RARS 软件通过对 benchmark 汇编程序阅读，找到中断程序的入口地址，当电路中产生了 INT 信号时，说明此时电路应该响应当前中断。然后将程序中当前执行的下一条程序语句地址作为中断返回地址 retPC 送入 mEPC 寄存器中，首先关中断，将 MIE 信号置为 0，然后 INT 信号会让多路选择器将中断处理程序首地址 intPC 送入 PC 寄存器，作为下一条指令的地址，当下一个时钟周期到来，便可进入中断处理程序。

当中断程序运行结束时，最后一条指令为 uret，当程序开始处理 uret 指令，通过多路选择器将 mEPC 中保存的中断返回地址 retPC 送入 PC 寄存器，返回中断处程序，开中断，将 MIE 信号置为 1，系统可以响应其他中断信号，使用中断清除信号将锁存的中断请求信号清空。

2.2.3 软件设计

软件设计中，重点是进入中断程序后的现场保护和运行结束即将退出时的现场恢

华中科技大学课程设计报告

复。现场的保护采用的方式是把原程序中重要的寄存器数据使用 SW 指令保存到数据寄存器中，恢复时使用 LW 指令从数据寄存器中读取之前保存的寄存器数据，重新写回寄存器堆。

2.3 多级中断机制设计

2.3.1 总体设计

在单级中断的基础上实现多级中断，需要考虑中断程序被优先级更高的中断信号所中断的情况。我们需要用电路比较新的中断和当前中断程序的优先级，以决定是否需要进入新的中断程序。我们需要设计硬件栈来保存返回地址，以支持嵌套中断。我们还需要考虑进入中断服务程序后保护现场和恢复现场的原子性，在进入中断后关中断，保护现场后用指令 CSRRCI 开中断；在恢复现场前用指令 CSRRSI 关中断，执行指令 ERET 时同步开中断。

2.3.2 硬件设计

单周期多级中断的硬件设计同单周期单级中断设计类似，在检测到中断请求信号后，将其锁存然后输入到优先编码器中，对应着输入的 123 位置。在发生下述情况是进行程序段的切换，首先锁存当前的优先编码器输出（无任何中断请求时锁存为 0），当 IE 为 1 时，每个时间周期都将所存的编码与当前的优先编码器输出相比较，若不同，有以下两种情况：第一种为在当前程序段中存在新的中断请求，第二种为中断程序处理完成返回上一级程序段。针对上述不同的情况进行相应的处理。

对于 IE 中断使能信号的切换，当 INT 信号来临或者中断程序中产生了 CSRRCI 指令时将 IE 置为 0，此时系统进行中断程序的现场保护和现场恢复的任务；当中断返回指令 uret 到来并且中断程序中存在 CSRRSI 指令时将 MIE 置为 1，此时可以响应优先级更高的中断，从而完成了高优先级中断总能被相应。

2.3.3 软件设计

在产生中断时，首先进行现场保护，将源程序中寄存器的数据使用 SW 存储指令保存到数据寄存器中，方便中断返回后的数据还原；当中断结束时，使用 LW 指令从数据寄存器中读入中断前的数据，将其诚信写回寄存器堆中。同时需要新增两条指令

华中科技大学课程设计报告

CSRRSI 和 CSRRCI 并实现对相应指令的响应,产生方式为通过 excel 自动生成逻辑,使用 CSRRSI 指令控制中断使能信号的打开,当保存完现场数据后,用其开中断;使用 CSRRCI 控制中断使能信号的关闭,当恢复数据前,使用它关中断。

2.4 理想流水 CPU 设计

2.4.1 总体设计

理想流水线将单周期 CPU 分成了五个阶段分别是: IF、ID、EX、MEM 和 WB,各阶段之间设立流水接口部件,其本质是寄存器,用于所存前一阶段加工完成的数据。理想流水线,通过接口传递和指令相关的数据、控制、反馈信息,所以需要设立相应的接口。

取指令阶段根据 PC 获取指令寄存器中的相应指令,然后将其向后传递;译码阶段需要根据指令产生需要的控制信号和取出寄存器操作数向后传递给执行阶段;执行阶段利用 ALU 进行运算,得到相应计算结果或者分支跳转地址;在访存阶段中需要向数据寄存器中写入或者读取相应数据;写回阶段是将 ALU 中计算得到的结果或者是从内存中读取的结果写回到对应的寄存器文件中。

需要注意的是,在这一部分需要考虑到后续气泡流水线和重定向流水线对流水接口特殊要求,如气泡插入和数据重定向判断等。在阶段与阶段的传递当中,需要重复传递相同名称的信息,所以也需要注意通过命名进行区分。

2.4.2 流水接口部件设计

流水接口部件设计了使能端和清零端。当使能端为高电平时,将从数据输入端读入数据并且保存到相应的寄存器中,数据输出端的值和寄存器中的值保持一致;当使能端为低电平时,不进行操作。当清零端为高电平时,进行各阶段中寄存器同步清零;若为低电平则正常进行。特别的,每一个接口部件都需要向后传递 PC 和 IR,方便流水线正常的运行。下表 2.2 为各阶段流水接口具体寄存的数据名称。

表 2.2 流水接口寄存数据名称表

流水部件名称	保存信号名称
IF/ID	IR, PC, PC+4, halt, clk

华中科技大学课程设计报告

ID/EX	jal,jalr,lh,RegWrite,MemtoReg,MemWrite,BLT,BEQ,BNE,AluOP,AluSrcB, halt,ecall,R1,R2,Rd,IR,imm,PC,PC+4,halt,clk
EX/MEM	jal,jalr,RegWrite,MemtoReg,MemWrite,halt,ecall,Alures,MDin,Rd,IR,PC,P C+4,halt,clk
MEM/WB	jal,jalr,lh,RegWrite,MemtoReg,halt,ecall,Alures,MRD,LHres,Rd,IR,PC,PC+ 4,halt,clk

2.4.3 理想流水线设计

在理想流水线中，需要考虑数据通路的实现，同时需要将对应数据信号连接到对应的模块端口。同时，所有这一阶段使用到的数据需要从本阶段的上一流水接口中取出。流水线中的跳转指令在 EX 阶段完成跳转地址计算，并且在同一阶段完成跳转。JAL 指令由于同时涉及到跳转和写寄存器，所以将其跳转功能放在 EX 阶段实现，但是将写寄存器地址安排在 WB 阶段。SYSCALL 指令在 EX 阶段计算出结果，在 WB 阶段进行相应信号的处理。

2.5 气泡式流水线设计

2.5.1 总体设计

在理想流水线中，会遇到分支相关和数据相关冲突问题，通过使用流水阻塞和插入气泡的方式构建气泡流水线来解决相关问题。通过使用哈弗结构解决取指令和数据的争用主存，然后另外在电路中添加计算 PC 和分支地址的加法器来解决 ALU 争用。

在 IE 阶段添加检测分支成功信号，当检测到时，需要在 IF/ID 和 ID/EX 段插入气泡，解决分支相关冲突。

对于数据相关冲突，需要考虑不同段间的情况。通过将寄存器堆触发方式改为下降沿触发解决 ID 取操作数和 WB 向寄存器写的数据相关冲突；若是 ID 和 MEM 阶段冲突，则需要暂停 PC 取指令和 IF/ID，并且在 ID/EX 接口中插入气泡；如果是 ID 和 EX 计算冲突，同样是暂停 PC 和 IF/ID 并且在 ID 和 EX 间的流水接口中插入气泡，将其转变成 ID 和 WB 相关。

2.6 重定向流水线设计

2.6.1 总体设计

重定向流水线在气泡流水线的基础上进行修改得到，通过加入 Load_Use 冲突检测和数据重定向路径将其改为重定向流水线。

在 EX 阶段，重新定向进入 EX 的 R1 和 R2 通路的数据，判断是将 R1 和 R2 作为 ALU 的操作数还是将 R2 向后传递到 MEM 作为写入存储器的值。

在 ID 阶段，将重定向选择信号传递到 EX 阶段，同时在检测相关时，还需要给出是否存在 Load_Use 相关以及具体给出 R1 和 R2 和 EX 和 MEM 哪一阶段相关，若相邻两条指令存在数据相关且前一条指令为访存指令时，产生 Load_Use 信号并且需要采用插入气泡的方式消除相关。

重定向流水线中需要添加 Forward 判断模块用来选择 ALU 的两个数据输入端时选择对应的原始数据输入还是指定的重定向数据进行输入。

2.7 动态分支预测机制

2.7.1 总体设计

动态分支预测流水线的基础为重定向流水线，通过添加 BHT 器件已获得动态分支预测功能。当在 IF 阶段识别到分支指令时，流水线会出现三种情况，包括未命中，预测失败和预测成功。若未命中，则流水线表现与未添加预测时一致；若是预测失败，则需要 EX 段对预测结果进行判断，向 IF 和 ID 段插入气泡消除相关同时向 PC 寄存器中传入正确的地址；若是预测成功，则流水线可以根据预测地址顺序执行。

BHT 中存放的时分支指令地址和分支目标地址，以及分支预测历史位。采用双预测位对 BHT 进行状态转换，高位用于预测，低位用于更新预测状态。使用八路全相联映射 cache 实现 BHT。

2.8 重定向单级中断

2.8.1 总体设计

重定向单级中断电路在重定向流水线和单级中断的基础上进行设计，需要分别考虑中断陷入和中断返回处理。

华中科技大学课程设计报告

首先介绍中断陷入设计。当主程序始终上升沿到来，会触发 INT 上升沿。在中断程序的第一个时钟周期内会产生高电平信号 INT_Level。当 INT_Level 为 1 时，需要将 ID/EX 和 EX/MEM 同步清零，PC 此时选择中断入口地址，并产生中断选择信号，并且在这时 IF 的原 PC 将失效，插入两个气泡让 ID 和 EX 段指令失效。特别需要注意的是，对于连续中断，需要保证恢复现场的原子性，若前一个中断的中断返回处于 ID 段，则不需要插入气泡，进入新的中断，中断返回地址不变。同时需要保存 EX 段的 PC 值，因为当 EX 段的 PC 送到寄存器终止后中断准备信号才触发寄存器，所以寄存器中保存的是正确的 PC 地址。当需要关中断时，中断信号将 IE 寄存器置零。

中断返回通过 uret 信号控制选择中断返回地址作为 IF 段的 PC 值，同时开中断，为下一个中断的到来做准备。

2.9 团队任务设计

2.9.1 团队总体任务

设计实现一个走迷宫的小游戏。

硬件方面使用 logisim 平台，通过键盘 keyboard 输入上下左右及选择关卡，通过 LED 点阵输出显示初始界面、迷宫地图和完成界面。

软件方面使用 rars，编写写入界面信息、写入地图信息、上下左右中断等汇编程序。最后 assemble 无错误后，将 hex 文件导入指令存储器，与硬件部分进行联调。

2.9.2 团队成员及分工

郑欣觉（组长）负责总体设计和软硬件联调，罗虞阳负责硬件部分，黄绍恒和兰浩负责软件部分。具体分工如表 2.3 团队成员及分工所示。

表 2.3 团队成员及分工

成员	分工
郑欣觉（组长）	实现走迷宫的基础软硬件模型，进行联调
罗虞阳	完成 logisim 平台中硬件的设计
黄绍恒	完成软件部分汇编代码的编写
兰浩	设计初始界面、迷宫、通关界面

2.9.3 团队任务总体设计

(1) 硬件方面

在 logisim 平台上，以 RISC-V 单周期 CPU+单级中断为基础实现。

CPU 有六个主要输入，分别为上下左右和两个关卡选择。输入均通过单级中断实现相应功能。另设有 CLK 时钟输入和 RST 清零置位。

CPU 输出为 8 行点阵信息。通过 LED 点阵显示迷宫界面信息。

通过 logisim 平台提供的 keyboard 组件实现键盘的输入逻辑。

(2) 软件方面

在 rars 上编写 RISC-V 架构的汇编代码。

需要实现初始界面的数据写入、迷宫地图的数据写入、通关界面的数据写入、上下左右移动的中断程序。

整体流程图如下图 2.2 所示。

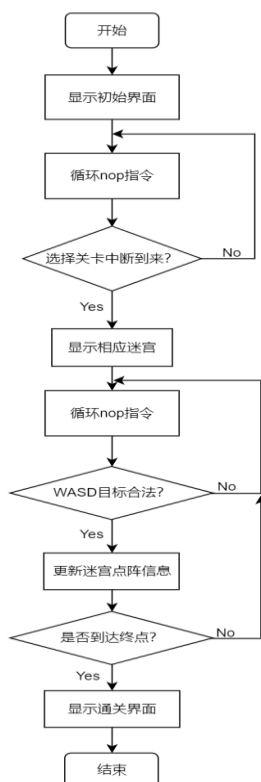


图 2.2 走迷宫流程图

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)

程序计数器使用 32 位寄存器实现，触发方式位上升沿触发，输入为下一条指令的地址，输出为当前需要执行的指令的地址。使用 halt 信号控制电路是否停机，以及使用 RST 信号将寄存器置零复位。如图 3.1 所示。

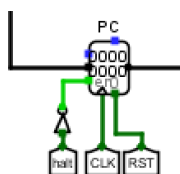


图 3.1 程序计数器 (PC)

2) 指令存储器 (IM)

指令寄存器使用只读存储器实现。指令存储器中存储位宽为 10 位，所以只取指令地址的 2-11 位作为指令存储器的输入地址而屏蔽掉了高位部分和字节偏移部分。如图 3.2 所示。



图 3.2 指令存储器 (IM)

3) 数据存储器 (DM)

数据存储器使用随机 RAM 实现，地址位宽为 10 位，存储数据位宽为 32 位。但是由于 RAM 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，只取指令地址的 2-11 位作为输入地址。如图 3.3 所示。

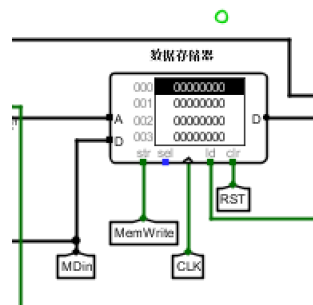


图 3.3 数据存储器 (DM)

4) 寄存器堆 (RF)

寄存器堆直接使用 cs3410.jar 中预先实现好的寄存器组，只需要将信号传递通路连接好就能完成相应功能。如图 3.4 所示。

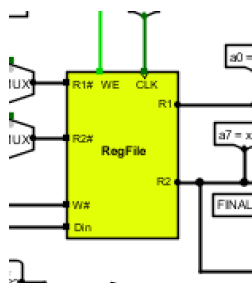


图 3.4 寄存器堆 (RF)

5) 运算器 (ALU)

使用 cs3410.jar 中的 ALU 部件，输入为两个操作数和 AluOp，输出为计算的值或结果信号。如图 3.5 所示。

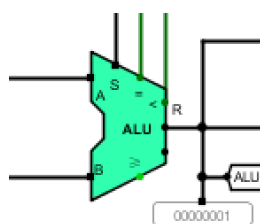


图 3.5 运算器 (ALU)

3.1.2 数据通路的实现

将 3.1.1 中的各个部件和控制器实现后，将信号正确连接到输入端口和输出端口，添加部分额外的计算和判断电路，完成整体数据通路的实现。如图 3.6 所示。

华中科技大学课程设计报告

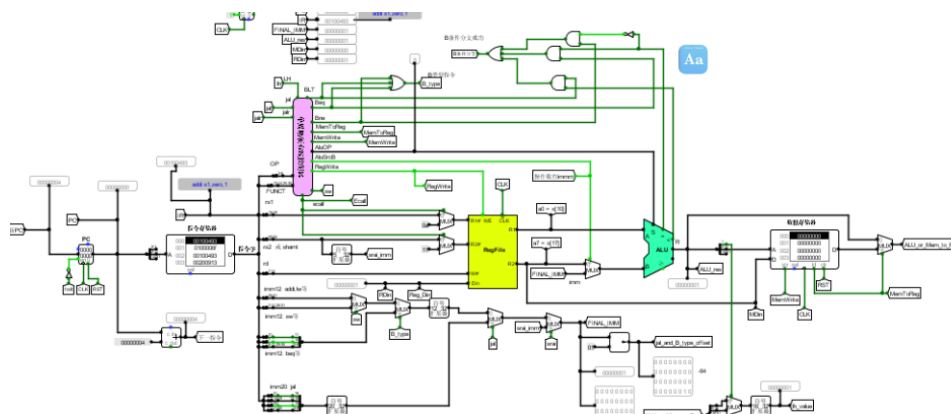


图 3.6 数据通路实现

3.1.3 控制器的实现

首先填写单周期硬布线自动生成 Excel 表格中的相应数据，然后使用 Excel 中的自动生成功能生成每个信号的响应表达式，输入到 logisim 中的自动生成电路中，构建出相应控制器。Excel 表数据如图 3.7 所示，封装后的控制器如图 3.8 所示。

#	指令	Func7 (1 位 0)	Func3 (1 位 0)	OpCode (1 位 0)	ALU_OP	MemToRegMemWrite	ALU_Src	RegWrite	ecall	S_Type	Beq	Bne	Jal	Jalr	SLL	SLTIU	LH	BLT	rs1_used	rs2_used	CSRRI	CSRRCI
1	add	0	0	c	5			1											1	1		
2	sub	32	0	c	6			1											1	1		
3	and	0	7	c	7			1											1	1		
4	or	0	6	c	8			1											1	1		
5	sll	0	2	c	11			1											1	1		
6	sllr	0	3	c	12			1											1	1		
7	addi	0	4	5			1	1											1	1		
8	andi	7	4	7			1	1											1	1		
9	ori	6	4	8			1	1											1	1		
10	xori	4	4	9			1	1											1	1		
11	slli	2	4	11			1	1											1	1		
12	sllr	0	1	4	9		1	1											1	1		
13	srlr	0	5	4	2		1	1											1	1		
14	srai	32	5	4	1		1	1											1	1		
15	lw	2	0			1	1	1											1	1		
16	sw	2	8			1	1	1		1									1	1		
17	ecall	0	0	1c					1										1	1		
18	bneq	0	10	6							1								1	1		
19	bne	1	10	6								1							1	1		
20	jal	1b					1	1					1						1	1		
21	jalr	0	19	6			1	1						1					1	1		
22	CSRRI	6	1c																		1	
23	CSRRCI	7	1c																			1
24	URET	2	0	1c																		
25	SLL	0	1	c	0			1							1				1	1		
26	SLTIU	3	4	12			1	1								1			1	1		
27	LH	1	0	5		1		1									1		1	1		
28	BLT	4	10	11				1										1	1	1		

图 3.7 硬布线 Excel 表格

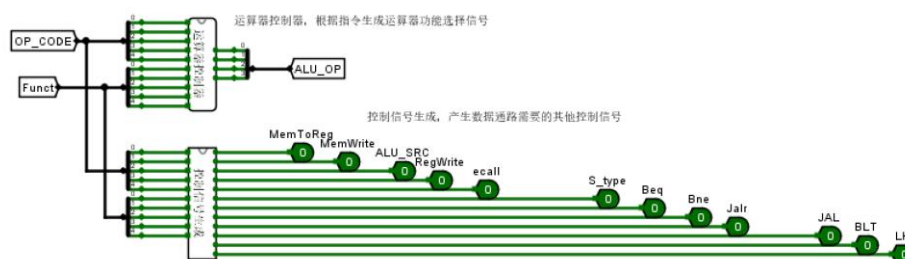


图 3.8 硬布线控制器实现

3.2 中断机制实现

3.2.1 单级中断

中断请求信号产生电路如图 3.9 所示。当中断请求 IR 信号到来，首先使用两个 D-触发器锁存并标记为 no_，代表第几级中断请求，直到相应中断清除信号 clr_到来，将其归零。

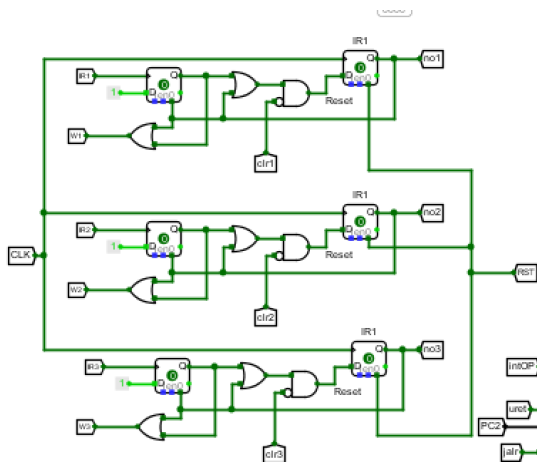


图 3.9 中断信号生成电路

将锁存的中断请求信号送入优先编码器输出编号最大的输入端口，代表系统仅处理优先级最大的中断请求信号。优先编码器会输出待处理的中断编号以及给出一个中断指示信号表示准备进入中断，该信号和中断使能信号 mie 做与运算得到最终的可以打断程序运行转到中断程序的中断信号 INT。如下图 3.10 所示。当中断程序执行完毕运行到最后一条指令 uret 时，通过硬布线控制器给出 uret 信号为 1，结合 INT 信号和 chOP 信号选择当前中断编号，并生成对应的中断清零信号 clr_，clr 会将中断产生电路中的对应信号清零。

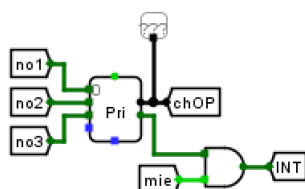


图 3.10 中断控制信号产生电路

在准备进入中断时，需要首先保存即将执行的下一条指令的地址作为中断返回地址，将改地址送入寄存器保存为 retPC，并将 mie 信号置零，然后将 intPC 送入 PC 寄存器中，作为下一条指令的地址，在下一个时钟周期进入中断处理程序。在中断程序

华中科技大学课程设计报告

运行到结束指令 `uret` 时，程序识别 `uret` 指令，并生成对应信号 `uret`，通过多路选择器选择 `retPC` 作为返回地址送入 `PC` 寄存器，同时将 `mie` 信号置 1，此时表明可以继续相应其他中断信号。整体中断电路如图 3.11 所示。

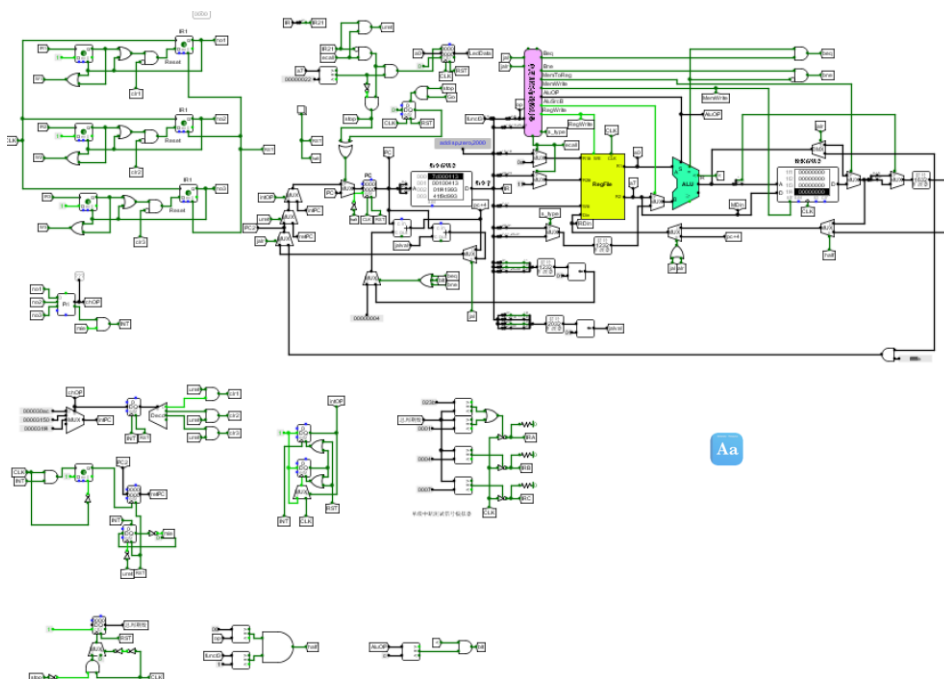


图 3.11 单级中断电路

3.2.2 多级中断

多级中断的大体思路和单级中断相类似，不同点在于多级中断中可能会出现中断嵌套。在中断请求信号产生后，首先输入到优先编码器中，然后输出当前最大编号并使用多路选择器选择需要跳转到的中断入口地址。在每个时钟周期，若 `mie` 为 1，则将当前正在执行的中断编号和新产生的中断编号作比较，若相同或者小于，则代表优先级低于当前正在执行的中断程序，不对当前程序进行中断，反之则产生中断信号 `INT`，并输出中断准备信号。如图 3.12 所示。

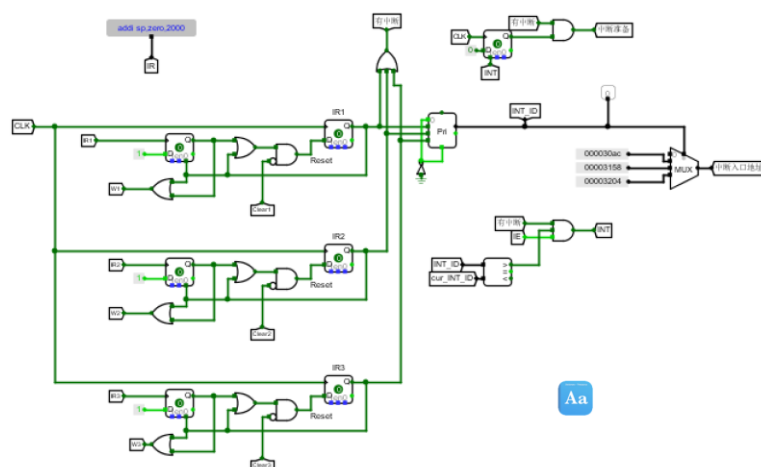


图 3.12 多级中断信号产生电路

由于最多有三级中断，所以使用三个寄存器进行保存断点。根据中断编号将中断返回地址保存到对应寄存器之下，在相应中断完成后返回到对应的中断返回地址处继续执行后续指令。如图 3.13 所示。

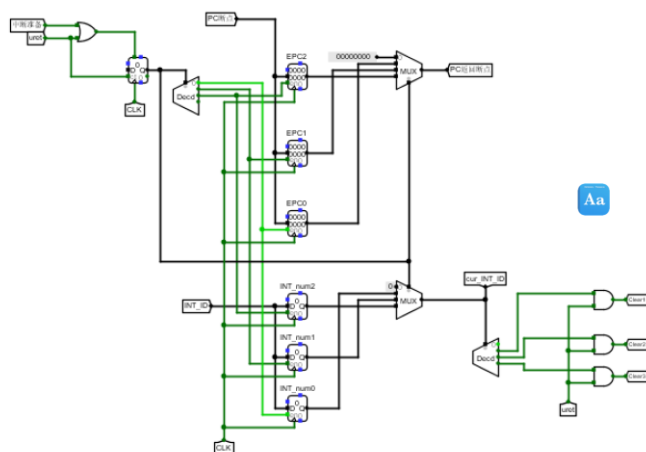


图 3.13 中断断点保存电路

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进行锁存，寄存器使用上升沿触发，另外设置清零端和使能端分别对寄存器进行置零操作和实现连通。因为四个流水接口实现原理完全相同并且只有信号数目上的差异，所以只展示其中一个，如图 3.14 所示。

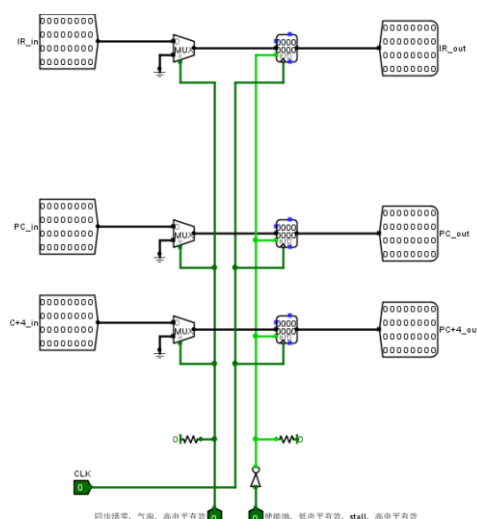


图 3.14 流水接口电路

3.3.2 理想流水线实现

理想流水线因为不涉及分支指令和对相关的处理，所以只需要正确连接控制信号和对应的控制部件即可，整体数据通路如图 3.15 所示。

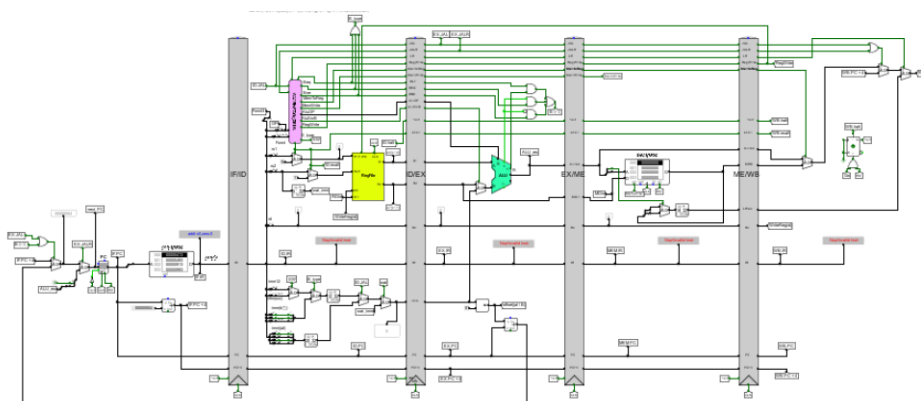


图 3.15 理想流水线

3.4 气泡式流水线实现

气泡流水线需要首先实现数据相关的检测判断逻辑。ID 和 WB 的数据冲突只需要将寄存器触发条件改为下降沿触发即可，对于 ID 段和 EX、MEM 段的冲突，通过检查这两个段的寄存器写入信号是否为 1 即可，同时写寄存器编号与源寄存器编号是否相同，最后生成出数据冲突信号。具体判断电路如图 3.16 所示。

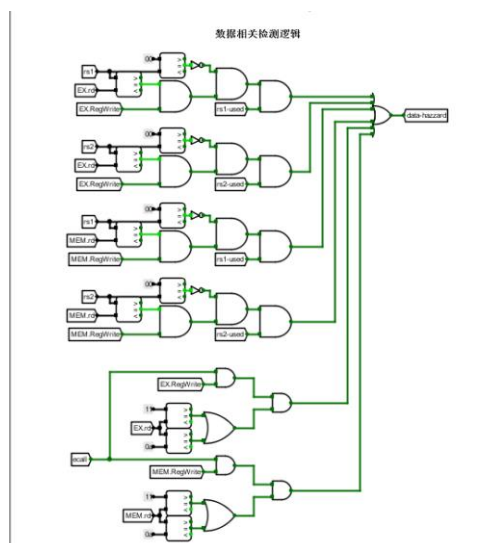


图 3.16 数据相关检测

在判断完之后需要根据判断的结果结合分支跳转给出是否需要插入气泡，分支跳转判断在主电路中完成，同时检测无条件跳转指令和条件跳转指令，通过将条件跳转信号同 ALU 的结果信号相判断并与无条件跳转指令做或运算生成 BranchTaken 信号，在主电路中的实现如图 3.17 所示。

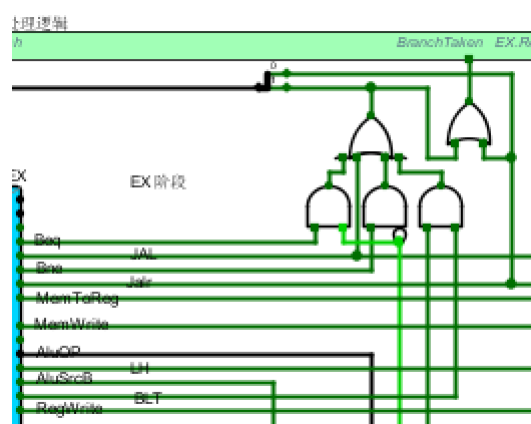


图 3.17 分支相关检测

当 BranchTaken 信号为 1 时，表示当前需要执行分支跳转，此时 IF 和 ID 段的指令为错误指令，需要给其流水寄存器清空信号，然后通过控制寄存器的使能端让流水寄存器的值保持不变来暂停 IF 和 ID 段的执行。发生数据冲突，则需要使用暂停信号 stall 时 PC 和 IF/ID 寄存器值不变即可。整体相关判断和气泡判断具体实现如图 3.18 所示。

数据相关处理逻辑 & 气泡处理输出(Output)

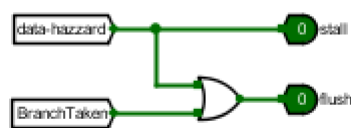


图 3.18 相关处理和气泡产生逻辑

气泡流水线的整体实现如图 3.19 所示。

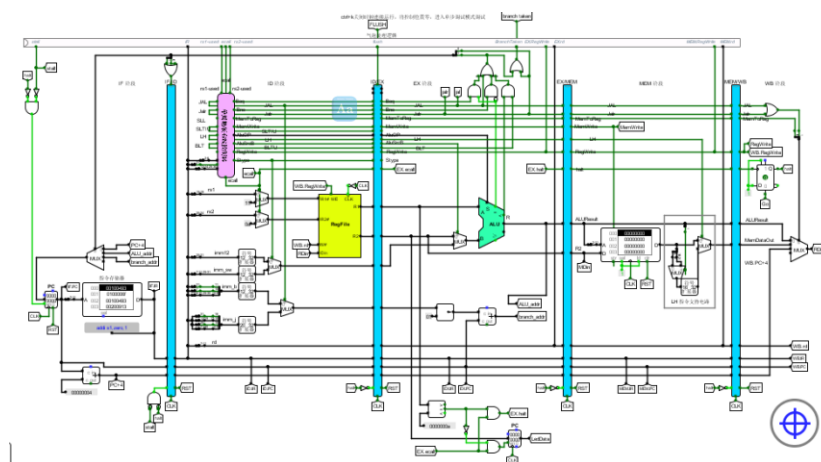


图 3.19 气泡流水线

3.5 重定向流水线实现

重定向流水线需要在气泡流水线的基础之上添加 Load_Use 冲突的检测，通过检测各个段的信号判断是否有 Load_Use 冲突发生。具体实现如图 3.20 所示。

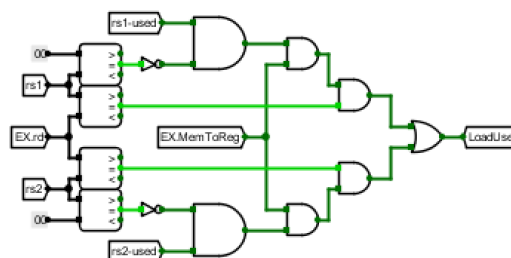


图 3.20 Load_Use 冲突判断逻辑

重定向流水线还需要产生多路选择信号 Forward 使得 ALU 使用正确的数据进行计算，其中 rs1-used 代表指令的第一个寄存器被使用，EX.RegWrite 和 MEM.RegWrite 表示对应段存在被写的寄存器，从上到下分别检测 rs1 和 EX 数据相关，rs1 和 MEM

华中科技大学课程设计报告

数据相关，rs2 和 EX 数据相关，rs2 和 MEM 数据相关。具体实现如图 3.21 所示。

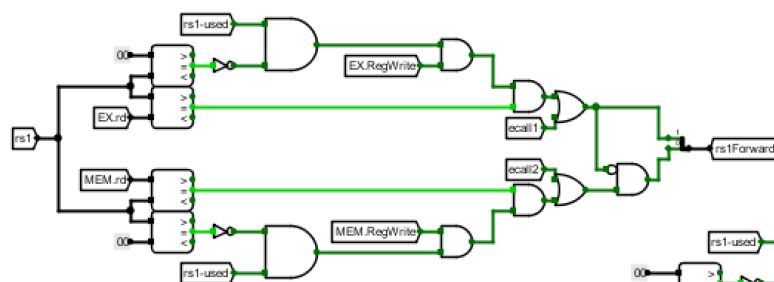


图 3.21 forward 信号产生逻辑

发生 Load-Use 相关，需要暂停 IF 和 ID 段当前指令的执行，然后在 EX 段插入气泡；在 EX 段处理分支跳转指令时，需要清空 IF 和 ID 段的错误指令。最后，重定向流水线的整体数据通路如图 3.22 所示。

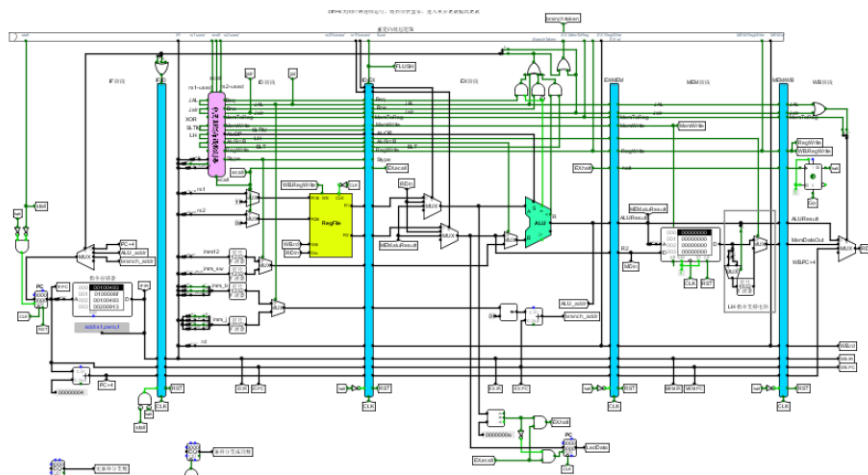


图 3.22 重定向流水线

3.6 动态分支预测机制实现

3.6.1 BTB

BTB 将分支成功的分支指令的地址和它的分支目标地址都放到一个缓冲区中保存起来，缓冲区以分支指令的地址作为标识，使用八路全相连 cache 实现，使用四个寄存器保存 valid、分支指令地址、分支目标地址和分支预测历史位。在需要读取时，若命中则置 read 信号为 1，同时将分支预测位值传给 p 信号，分支目标地址传给 JumpAddr 信号。在写数据时，valid 位置 1，然后将当前 EX.PC 写入分支指令地址，将跳转地址写入分支目标地址，然后将分支预测历史位置 0，表示重新开始计数。整体全相连 cache 如下图 3.23 所示。

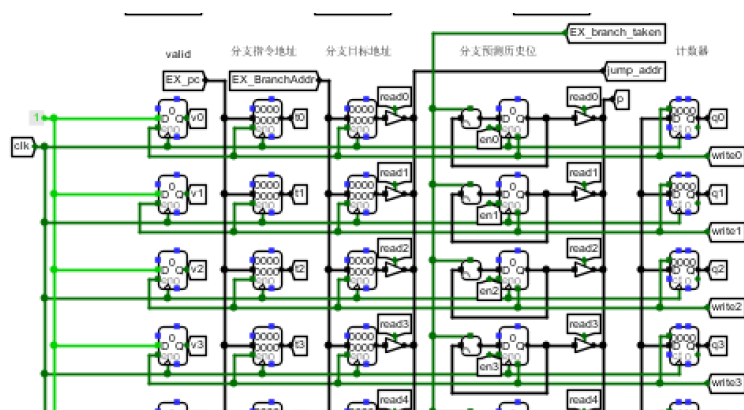


图 3.23 全相连 cache

分支预测历史位高位为将分支预测为成功或者失败，低位为此次预测是否成功，转换规律在此不做赘述，更新方法为通过当前状态和是否跳转使用下图中的组合逻辑计算下一跳跳转指令来临时的状态，此次更新会在下一时钟周期到来时更新。分支预测电路如图 3.24 所示。

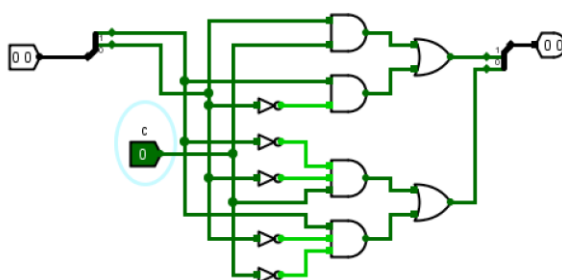


图 3.24 分支预测电路

在获得分支指令地址后，首先对该指令进行查找，使用 8 个比较器进行八路并发计算，若命中则产生 read 信号，并将预测跳转信号置 1，反之置 0；若需要对 BTB 表进行更新，则使用 EX 段处的 PC 进行比较，使用 LRU 的替换策略，若命中，则只需要更新分支预测历史位即可。如图 3.25 所示

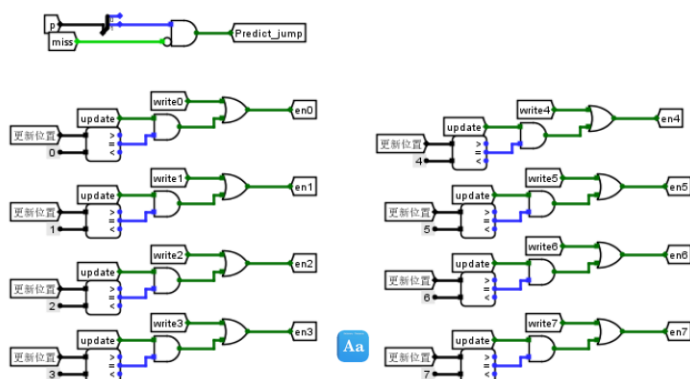


图 3.25 分支预测历史位更新电路

3.6.2 主体电路

根据 BTB 输出的 pj 信号选择 $PC+4$ 或是分支目标地址送入 PC 寄存器，然后通过计算 BranchTaken 信号和 pj 信号生成 PErr 信号，当该信号为 1 时代表预测错误，根据 EX 段的分支选择信号使用多路选择器重新选取指令。预测错误是，需要清空 IF/ID 和 ID/EX 流水寄存器。整体电路如下图 3.26 所示。

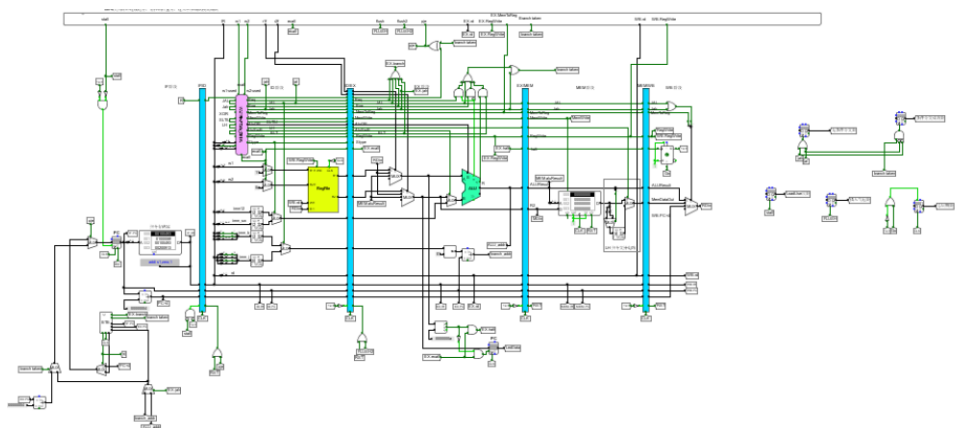


图 3.26 动态分支预测电路

3.7 重定向单级中断

重定向单级中断使用重定向电路和单级中断电路组合设计，中断信号的产生，选择，消除同单级中断相同。当中断到来，首先保存 PC 断点，然后根据中断准备信号选择中断入口地址，在执行到最后一条 `uret` 语句时，根据保存的 PC 断点返回源程序继续执行。整体电路如图 3.27 所示。

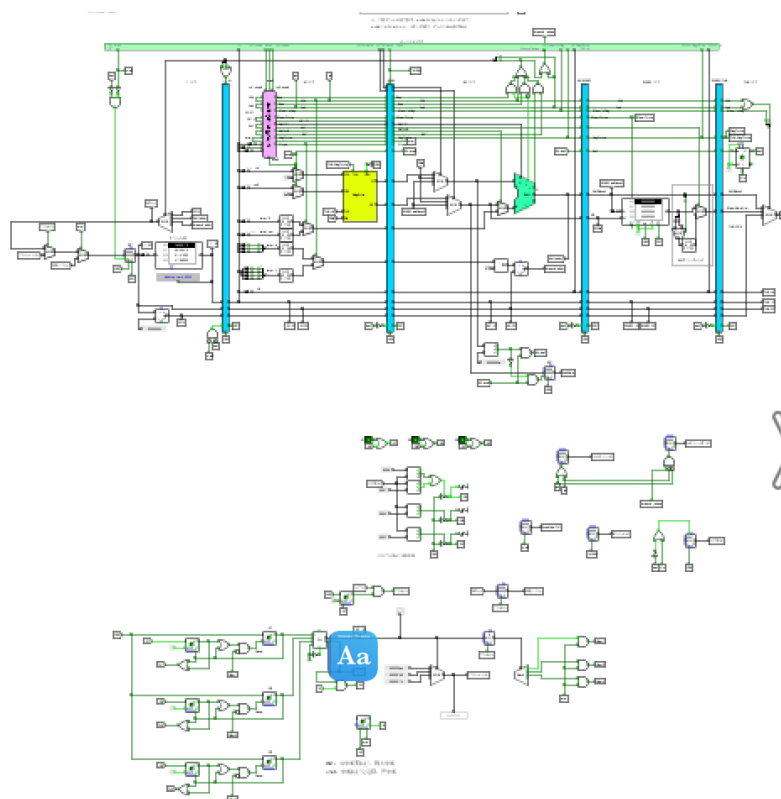


图 3.27 重定向单级中断电路

3.8 团队任务-硬件设计

我们这次的团队任务是设计一个走迷宫电路，具体实现的电路如下图 3.28 所示。

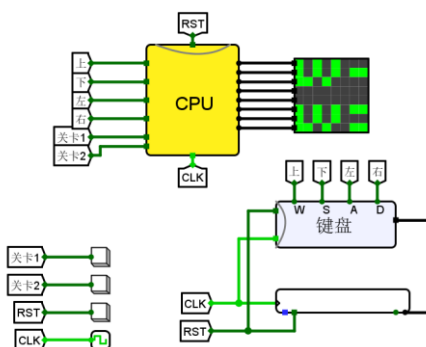


图 3.28 走迷宫电路

接下来介绍硬件实现部分。

首先是上图中黄色的 CPU 部分，实现电路如下图所示。使用六个中断分别控制上下左右和两个关卡选择。当电路接收到中断信号后，根据预存好的中断入口地址进入相应的中断地址处继续执行，然后产生对应的中断清除信号进行复原。如图 3.29 所示。

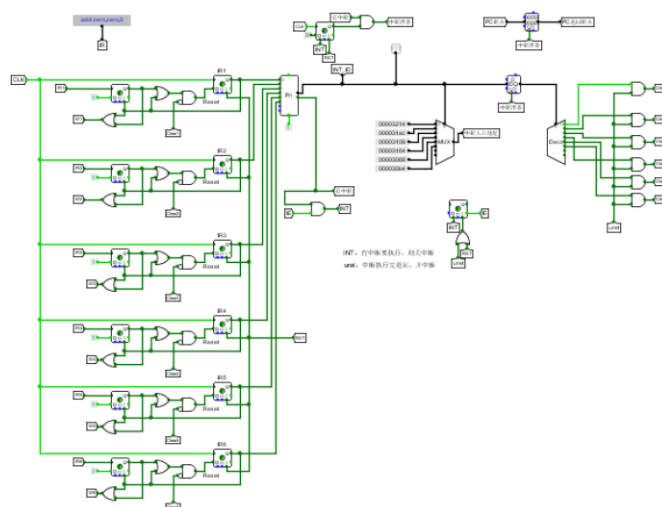


图 3.29 中断处理电路

对于图 3.28 所示电路的输出时点阵地图，实现的电路如下图所示。地图和点阵分别使用寄存器进行保存，当地址信号和寄存器写信号到来时将地图寄存器中存储的数据输入到点阵电路中使用寄存器保存，然后将对应的点阵信息输出到 LED 点阵中完成显示。如图 3.30 所示。

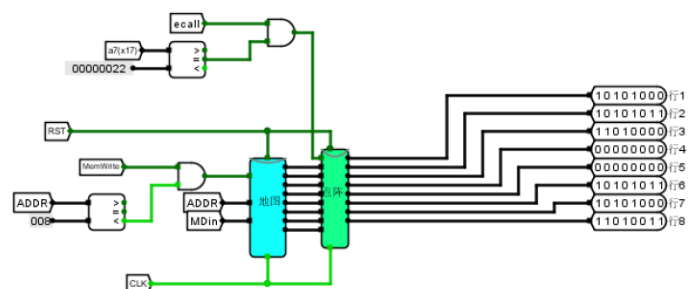


图 3.30 地图显示电路

地图电路在每一次时钟周期信号到来更新每一行的新数据，实现电路如下图 3.31 所示。

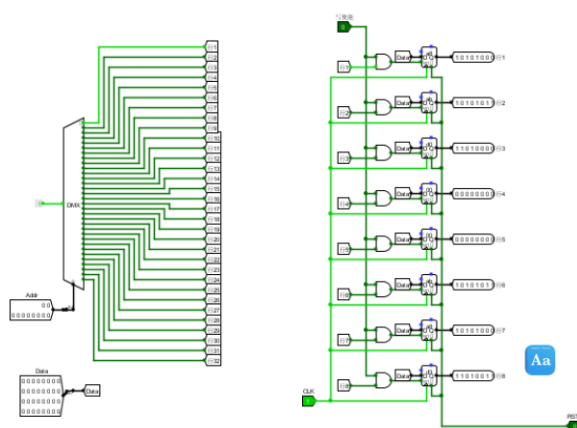


图 3.31 地图电路

点阵电路使用寄存器堆相应显示数据进行锁存，维持在 LED 显示器上的显示。如图 3.32 所示。

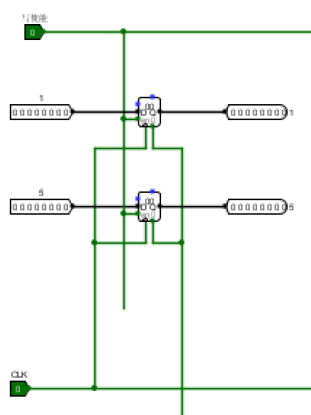


图 3.32 点阵显示电路

键盘电路根据键盘中的输入，通过比较得出相应的按键输入，然后产生对应的信号输出到中断中。实现电路如下图 3.33 所示。

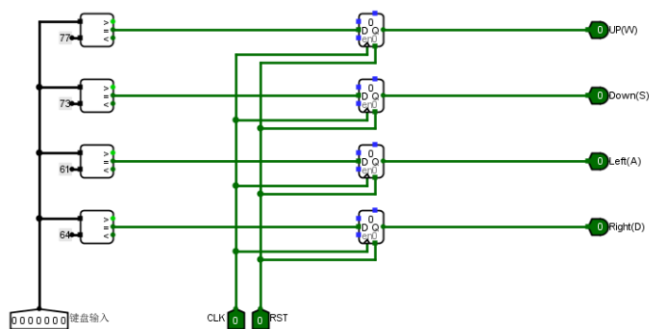


图 3.33 键盘按键中断电路

4 实验过程与调试

4.1 测试用例和功能测试

4.1.1 CPU24+4 测试

首先修改 risc-v-benchmark_ccab.asm 程序，在程序末尾添加上自己的 ccab 指令，然后生成 benchmark.hex 文件。在单周期指令寄存器中加载 benchmark.hex 文件，运行结果和执行四条新增 ccab 指令结果如下表 4.1 所示。

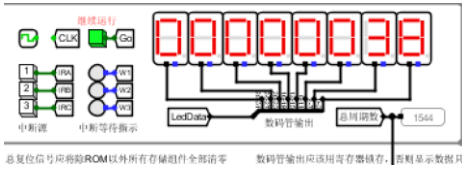
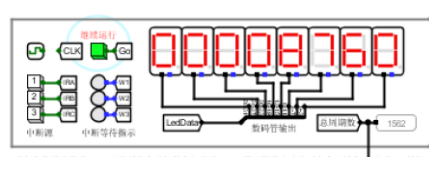
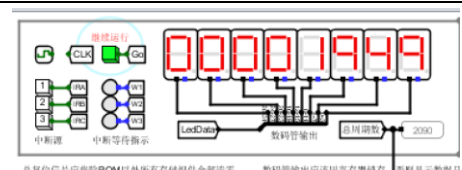
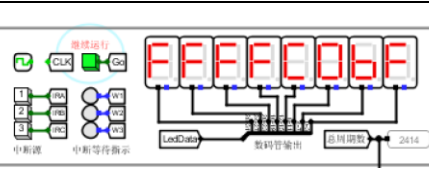
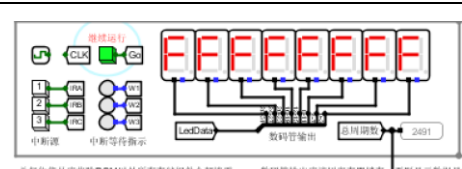
	
CPU24+4 测试结果	CPU24+4 测试 SLL 指令过程
	
CPU24+4 测试 SLTIU 指令结果	CPU24+4 测试 LH 指令结果
	
CPU24+4 测试 BLT 指令结果	

表 4.1 CPU24+4 测试结果

下图 4.1 为内存排序结果。

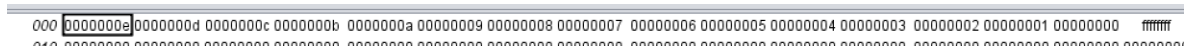


图 4.1 CPU24+4 测试内存结果

从上表和上图中可以看出，测试结果和预期结果相同。

4.1.2 单级中断测试

加载 risc-v-单级中断测试程序.hex，顺序执行程序，执行顺序为 1、3、2，最后返回主程序执行。下表 4.2 为运行时截图。

华中科技大学课程设计报告

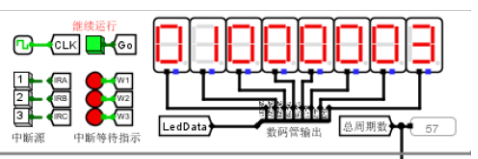
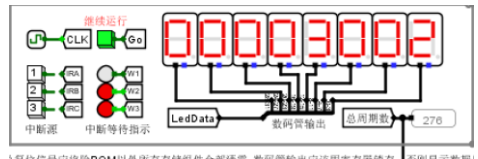
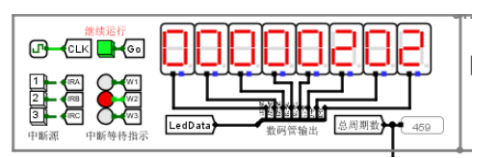
	
执行中断 1	执行中断 3
	
执行中断 2	

表 4.2 单级中断测试结果

4.1.3 多级中断测试

加载 risc-v-多级中断测试程序.hex，顺序按下中断 1、2、3，执行顺序为 1、2、3、2、1。下表 4.3 为执行过程截图。

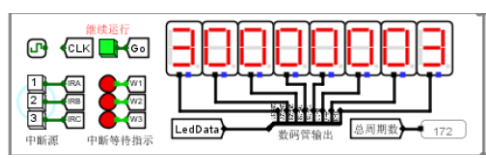
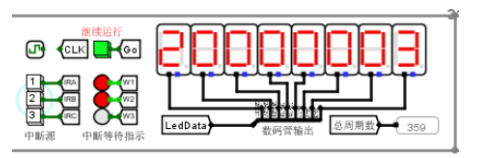
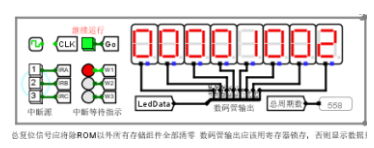
	
执行中断 3	执行中断 2
	
执行中断 1	

表 4.3 多级中断测试结果

4.1.4 理想流水线测试

加载 risc-v-理想流水线测试程序.hex，执行结果总周期数为 20。结果如下图 4.2 所示。

总周期数	20	0014
无条件分支数	???	
条件分支成功数	???	
插入气泡数	???	
LoadUse次数	???	

华中科技大学课程设计报告

图 4.2 理想流水线测试结果

内存保存内容如下图 4.3 所示。

```
000 00000000 00000001 00000002 00000003 01
010 00000000 00000000 00000000 00000000 01
```

图 4.3 理想流水线测试内存结果

4.1.5 气泡流水线测试

加载 benchmark.hex 文件，运行结果如下图 4.4 所示，其中总周期数位 3623。

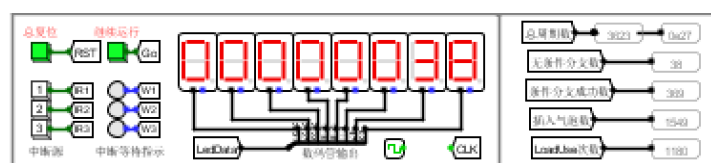


图 4.4 气泡流水线测试结果

内存存储内容如下图 4.5 所示。

```
000 00000009 0000000d 0000000c 0000000b 0000000a 00000009 00000008 00000007 00000006 00000005 00000004 00000003 00000002 00000001 00000000 ffffff
010 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

图 4.5 气泡流水线内存结果

4.1.6 重定向流水线测试

加载 benchmark.hex 文件，运行结果如下图 4.6 所示，其中总周期数位 2297。

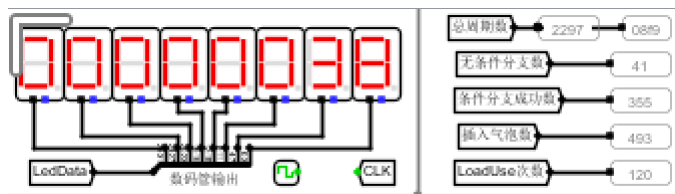


图 4.6 重定向流水线测试结果

4.1.7 动态分支预测测试

加载 benchmark.hex 文件，运行结果如下图 4.7 所示，其中总周期数位 1789。

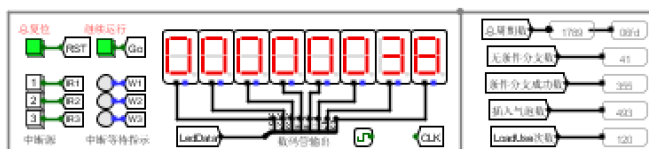
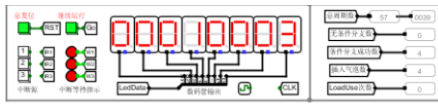
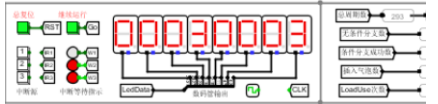
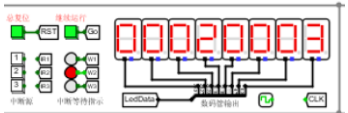


图 4.7 动态分支预测测试结果

4.1.8 重定向+中断流水线测试

risc-v-单级中断测试程序.hex，中断顺序为 1、2、3，执行顺序为 1、3、2，最后返回到主程序执行，符合预期。表 4.3 为运行时截图。

表 4.4 重定向单级中断测试结果

	
执行中断 1	执行中断 3
	
执行中断 2	

4.1.9 团队任务测试

在这里仅做简要的演示测试，详情可参考小组团队任务演示视频。
首先显示初始状态，LED 点阵中显示的是关卡选择提示，分别是 W1 和 W2。如图 4.8 所示。

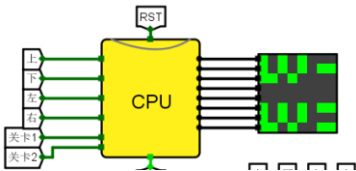


图 4.8 走迷宫初始界面

点击关卡一进入第一关，通过使用键盘的 WASD 四个按键控制上下左右移动。如图 4.9 所示。

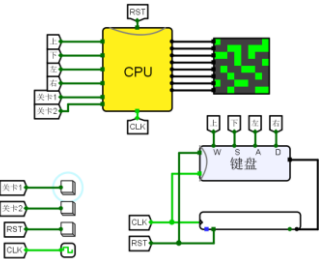


图 4.9 走迷宫第一关展示

当完成通关后，显示“√”表示通关成功。如下图 4.10 所示。



图 4.10 走迷宫通关提示

4.2 性能分析

下表列出了运行相同测试程序 `benchmark.hex`，不同电路的时钟周期数。

表 4.3 多级中断测试结果

方案名称	CPU24	气泡流水线	重定向流水线	动态分支预测
周期数	1544	3623	2297	1765

从上表中可以看出，气泡流水线的周期数是最多的，原因是因为电路在数据相关和分支相关部分插入了大量的气泡，所以导致总的周期数最多；重定向流水线通过使用旁路技术进行数据重定向，提高了流水线的速度，但是在分支相关部分和 `Load_Use` 相关还是需要插入气泡；动态分支预测建立在重定向流水线上，优化了分支处理相关的内容，使得周期数减少。

4.3 主要故障与调试

4.3.1 理想流水线故障

故障现象：在理想流水线中，执行 `ecall` 指令时指令运行错误，在 `ecall` 指令之前的指令无法正常完成执行，导致电路过早暂停。

原因分析：最开始 `halt` 指令在 EX 段生成，当下一个时钟周期到来，如果前一个时钟周期在 EX 段由于 `ecall` 指令产生了满足生成 `halt` 信号的条件，那么此时 `halt` 信号为 1，CPU 会暂停运行，这时 `ecall` 指令在 MEM 段执行，而 `ecall` 指令之前的指令在 WB 段执行，导致对应指令没有执行完，产生了相应的错误。

解决方案：对电路稍作修改，将 `ecall` 指令的判别部分放置到 EX 段进行比较，然后将比较的结果使用 EX/MEM 和 MEM/WB 两个流水寄存器传送到 WB 段，最后使用 WB 段的 `halt` 信号对 `halt` 寄存器进行设置，从而解决了上述问题。设计电路如图 4.11 所示。

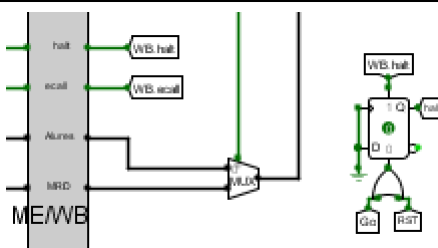


图 4.11 理想流水线解决方案

4.3.2 气泡流水线故障

故障现象：气泡流水线在前半段运行正常，但是到了后半段运行出现异常。

原因分析：在连接流水寄存器的过程中，由于操作失误导致出现了电路的短路，从而发生了工作异常的错误。如图 4.12 所示。

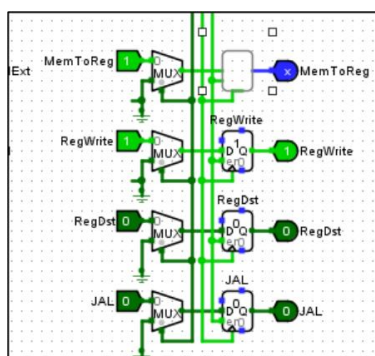


图 4.12 气泡流水线问题截图

解决方案：将短路电路重新正确连接即可。

4.3.3 单级中断故障

故障现象：uret 指令无法正确执行，中断程序在执行完成后不能返回断点处继续执行。

原因分析：uret 和 ecall 指令没有进行区分，另一个指令在 funct 和 opcode 字段都相同，导致中断程序将 uret 指令当做 ecall 指令进行处理。

解决方案：经查询，ecall 和 uret 指令在 imm12 字段不同，所以在单周期硬布线控制器的电路中新增一个输入位进行判断两个指令，该输入位为指令字的第 21 位。解决方案如图 4.13 所示。

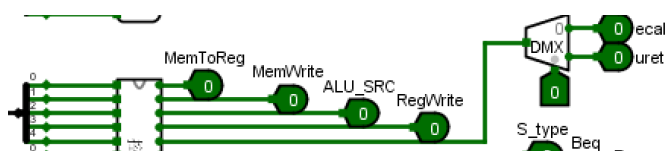


图 4.13 硬布线控制器内部实现

4.4 实验进度

表 4.1 课程设计进度表

时间	进度
第一天	单周期 risc-v 24+4 条指令，搭建整体框架
第二天	完成单周期 risc-v28 条指令，通过 educoder 测试
第三天	理想流水线，通过 educoder 测试
第四天	气泡流水线，设计并搭建流水接口
第五天	组装流水接口和主电路并调试，通过 educoder
第六天	设计搭建重定向流水线接口和旁路设计
第七天	整体组装，通过 educoder 测试
第八天	单周期 CPU 单级中断，查找资料，整理初步思路
第九天	开始上手，设计中断相关电路
第十天	调试中断电路，debug
第十一天	单级中断通过 educoder 测试；开始多级中断，查资料
第十二天	在单级中断的基础上进行修改完成多级中断设计并通过 educoder 测试
第十三天	动态分支预测，构建 BTB
第十四天	完成动态分支预测，
第十五天	完成重定向流水中断
后续	写组原报告

5 设计总结与心得

5.1 课设总结

该课程目标为掌握 5 段流水机制基本原理，能够利用多门课程的专业知识解决五段流水 CPU 设计的复杂工程问题，提升复杂工程问题分析解决能力。在该课设中主要做了一下工作：

- (1) 实现了单周期 CPU24+4 条 ccab 指令，构建了 CPU 的基本数据通路。这一部分是后续任务的基础；
- (2) 实现了单周期 CPU 单级中断功能，主要实现了中断的陷入和返回，并设计了相应控制信号；
- (3) 实现了单周期 CPU 多级中断功能，主要实现了嵌套中断的电路；
- (4) 实现了理想流水线，主要完成了流水接口的设计和流水线显示、清空、停机等功能；
- (5) 实现了气泡流水线，完成了流水阻塞和插入气泡功能，解决了控制冲突和数据冲突；
- (6) 实现了重定向流水线，在气泡流水线的基础上增加了旁路机制，将相关数据送到对应段的部件处，通过减少气泡数提高了流水线的运行效率；
- (7) 实现了重定向流水线的单机中单机制，主要完成了单周期和重定向流水线的合并；
- (8) 实现了流水线动态分支预测，主要设计了 BTB 部件和分支预测历史位变化电路，通过预测机制进一步提升了流水性能。

5.2 课设心得

这次的课设项目个人感觉难度非常大，依次实现了单周期 CPU，中断设计，流水线的相关设计以及最后的动态分支预测，每一个模块的完成都相当的考验耐心和学习能力，因此整个过程下来很有成就感，很充实。

从最简单的单周期 CPU 到实现单级中断，多级中断，再到以单周期 CPU 为基础建立理想流水线、气泡流水线和重定向流水线，再到最后的重定向单级中断流水线和

华中科技大学课程设计报告

动态分支预测功能的实现，难度是越来越大，辅助部件的实现也是越来越难，同时对出现的 bug 的调试也是越来越复杂和难以捉摸，有时调试电路让人头疼，完全不知道哪里出现了问题，有时又是在本地测试通过但是放到 educoder 上无法通过，结果发现是指令测试集没有更新到最新的版本，好几次是睡前闭上眼睛脑子里还都是电路图。有几次我也是请教了我们组内的队员，在他们的帮助下最终也是解决了问题，最终调试成功完成的喜悦也是让人难以忘却。

整个课程设计难度是由易到难，同时各个电路间又是相辅相成的关系，避免了一上来直接从入门到放弃的局面。把前面的基础任务完成对后续任务的顺利进行有很大的帮助，只要有耐心和毅力，出现的问题总是能够解决的。在这里我也是要感谢我们小组内的成员，没有他们在我找 bug 时的帮助，我也是没有办法顺利完成此次课设。这也让我意识到了团队合作的重要性。

此次课设是对我们基础知识的检测，将课本中的知识用自己的方式实现出来，加强巩固我们的学习能力和设计能力以及出现问题时的解决能力，团队项目则是很好的锻炼了我们的团队协作能力和沟通能力。

整理来讲，这次的课设体验苦中作乐，感谢组原老师们的辛勤付出，感谢小组成员的无私帮助。

最后我也对本次课设有一点点小小的建议，首先是团队设计部分，在一开始小组讨论的时候，大家都没什么思路，因为不知道能够实现一些什么功能，而且也不清楚定下来的题目实现难度如何。在查阅了大量的资料和讨论之后才确定了做走迷宫这个设计。所以我建议可以像软件工程那样预先设立好几个基础选题，同学们能够从这其中去进行选择，也可以在基础选题的基础之上进行优化拓展，或者进行创新，这样可以让同学们更有目标，同时也可以提高整个设计的上限。

最后再次感谢在我遇到困难时，帮助过我的群里的老师和同学们，以及我们小组内的成员。同时也希望组成原理这门课可以越来越好。

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] 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.

• 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：罗虞阳