***2022***



**硬件综合训练 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： | CS2003 |
| 学 号： | U202015375 |
| 姓 名： | 汪宇飞 |
| 电 话： | 15926575495 |
| 邮 件： | [2453253921@qq.com](mailto:2453253921@qq.com) |

目 录

[1 课程设计概述 3](#_Toc117106723)

[1.1 课设目的 3](#_Toc117106724)

[1.2 设计任务 3](#_Toc117106725)

[1.3 设计要求 3](#_Toc117106726)

[1.4 技术指标 4](#_Toc117106727)

[2 总体方案设计 6](#_Toc117106728)

[2.1 单周期CPU设计 6](#_Toc117106729)

[2.2 中断机制设计 10](#_Toc117106730)

[2.3 流水CPU设计 11](#_Toc117106731)

[2.4 气泡式流水线设计 12](#_Toc117106732)

[2.5 重定向流水线设计 13](#_Toc117106733)

[2.6 动态分支预测机制 14](#_Toc117106734)

[3 详细设计与实现 16](#_Toc117106735)

[3.1 单周期CPU 实现 16](#_Toc117106736)

[3.2 中断机制实现 20](#_Toc117106737)

[3.3 流水CPU实现 24](#_Toc117106738)

[3.4 气泡式流水线实现 26](#_Toc117106739)

[3.5 重定向流水线实现 27](#_Toc117106740)

[3.6 动态分支预测机制实现 29](#_Toc117106741)

[4 实验过程与调试 30](#_Toc117106742)

[4.1 测试用例和功能测试 30](#_Toc117106743)

[4.2 性能分析 33](#_Toc117106744)

[4.3 主要故障与调试 33](#_Toc117106745)

[4.4 实验进度 35](#_Toc117106746)

[5 设计总结与心得 37](#_Toc117106747)

[5.1 课设总结 37](#_Toc117106748)

[5.2 课设心得 37](#_Toc117106749)

[参考文献 39](#_Toc117106750)

# 课程设计概述

## 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。

## 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。

## 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。

## 技术指标

1. 支持表 1.1前24条基本32位RISC-V指令；
2. 支持教师指定的4条扩展指令；
3. 支持多级嵌套中断，利用中断触发扩展指令集测试程序；
4. 支持5段流水机制，可处理数据冒险，结构冒险，分支冒险；
5. 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。
6. 能运行教师提供的标准测试程序，并自动统计执行周期数
7. 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RISC-V32指令集，最终功能以RARS模拟器为准。 |
| 2 | ADDI | 立即数加 |
| 3 | AND | 与 |
| 4 | ANDI | 立即数与 |
| 5 | SLLI | 逻辑左移 |
| 6 | SRAI | 算数右移 |
| 7 | SRLI | 逻辑右移 |
| 8 | SUB | 减 |
| 9 | OR | 或 |
| 10 | ORI | 立即数或 |
| 11 | XORI | 或非 |
| 12 | LW | 加载字 |
| 13 | SW | 存字 |
| 14 | BEQ | 相等跳转 |
| 15 | BNE | 不相等跳转 |
| 16 | SLT | 小于置数 |
| 17 | SLTI | 小于立即数置数 |
| 18 | SLTU | 小于无符号数置数 |
| 19 | JAL | 转移并链接 |
| 20 | JALR | 转移到指定寄存器 |
| 21 | ECALL | 系统调用 | if ($a7==34) LED 输出$a0 的值else暂停等待 Go 按键 |
| 22 | CSRRSI | 访问CP0 | 中断相关,可简化为开中断 |
| 23 | CSRRCI | 访问CP0 | 中断相关,可简化为关中断 |
| 24 | URET | 中断返回 | 清中断，mEPC 送 PC，  开中断 |
| 25 | MUL | 乘法 | C |
| 26 | REMU | 求无符号数的余数 | C |
| 27 | LB | 字节加载 | A |
| 28 | BLTU | 无符号小于时分支 | B |

# 总体方案设计

## 单周期CPU设计

单周期处理器是指所有指令均在一个时钟周期内完成的处理器。尽管不同指令执行时间不同，但对单周期处理器而言，时钟周期必须设计成对所有指令都等长。本次对于单周期CPU的设计我们采用的方案是硬布线控制，且由于取指令和执行指令阶段均需要使用存储器，因此使用指令存储器和数据存储器分开的结构，即采用哈佛结构，实现主存储器（MM）和微程序控制存储器（CM）不共用一个存储器的方式完成方案的设计。同时在实施的过程中，采用Logisim软件来实现。

总体结构图如图 2.1所示。

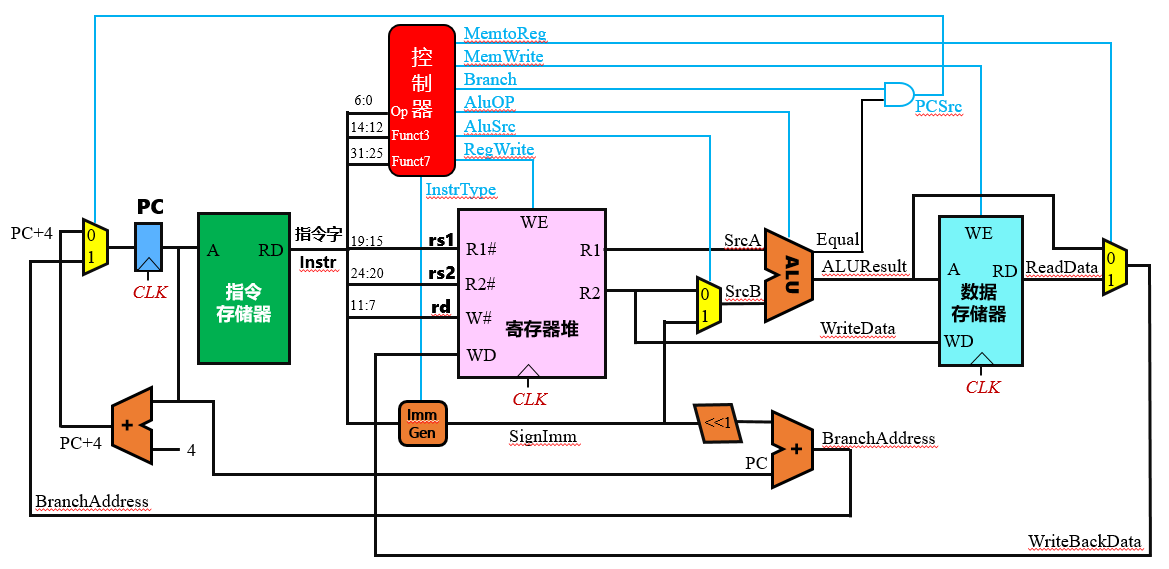


图 2.1总体结构图

### 主要功能部件

运算器部分，具体组成部分具体如下所示，分别为程序计数器PC、指令寄存器IM、运算器ALU和寄存器堆RF。

#### 程序计数器PC

程序计数器的作用为存放下一条要执行的指令在指令存储器中的地址。PC寄存器输入端口为下一条指令的地址、时钟信号CLK、使能信号以及复位信号，输出端口为指令的地址，采用上升沿触发。在单周期CPU中，由于存在跳转指令，需要分别计算跳转指令所需要跳转的地址，将这些地址与当前的PC+4一同送入多路选择器，根据跳转信号来进行选择作为PC寄存器的输入。

#### 指令存储器IM

指令存储器的作用为存储所要执行的RISC-V指令，为hex文件。只读寄存器ROM的输入端口为指令的存储地址，输出端口为指令字IR。由于一条RISC-V指令为四个字节，故应选取PC的2-11号为作为ROM的输入。

#### 运算器ALU

运算器的作用为进行数字计算并产生相应的逻辑信号，引脚与功能如表2.1所示。

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 |
| --- | --- | --- | --- |
| A | 输入 | 32 | 操作数A |
| B | 输入 | 32 | 操作数B |
| ALU\_OP | 输入 | 4 | 运算器功能码，具体功能见表2.2 |
| 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 | A≥B时为1，否则为0 |
| < | 输出 | 1 | A<B时为1，否则为0 |
| = | 输出 | 1 | A=B时为1，否则为0 |

运算器功能码ALU\_OP如表2.2所示。

表 2.2算术逻辑运算单元引脚与功能描述

| 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 |

#### 寄存器堆RF

寄存器堆的作用为接受信号进行数据的输入与输出。输入端口为读寄存器序号R1#和R2#、写寄存器序号W#、写入数据Din、写使能WE以及时钟信号CLK，输出端口为读出的寄存器数据R1和R2。本次课程设计已经给出设计好的上升沿触发寄存器，又由于不同电路的差异，在实现电路的过程中加入了另一个完全一样的但是是下降沿触发寄存器堆电路。

### 数据通路的设计

表 2.3指令系统数据通路框架

| 指令 | PC | IM | RF | | | | ALU | | | DM | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| R1# | R2# | W# | Din | A | B | OP | Addr | Din |
| R型指令 | PC+4 | PC | Rs1 | Rs2 | Rd | Result | R1 | R2 | 参考表2.2 |  |  |
| I型指令 | PC+4 | PC | Rs1 |  | Rd | Result /DM | R1 | IM | 参考表2.2 | Result | R2 |
| S型指令 | PC+4 | PC | Rs1 | Rs2 |  |  | R1 | R2 | 参考表2.2 | Result | R2 |
| B型指令 | PC+4/IMM | PC | Rs1 | Rs2 |  |  | R1 | R2 | 参考表2.2 |  |  |
| J型指令 | IM/IMM | PC |  |  | Rd |  |  |  | 参考表2.2 |  |  |

### 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 2.4。

表 2.4主控制器控制信号的作用说明

| 控制信号 | 取值 | 说明 |
| --- | --- | --- |
| RegWrite | 0/1 | 寄存器写使能 |
| MemWrite | 0/1 | 写内存控制信号 |
| AluOP | 0~12 | 运算器操作控制符（4位） |
| MemToReg | 0/1 | 寄存器写入数据来自存储器 |
| S\_Type | 0/1 | S型指令译码信号 |
| AluSrcB | 0/1 | 运算器B输入选择 |
| JALR | 0/1 | JALR指令译码信号 |
| JAL | 0/1 | JAL指令译码信号 |
| Beq | 0/1 | Beq指令译码信号 |
| Bne | 0/1 | Bne指令译码信号 |
| Bltu | 0/1 | Bltu指令译码信号 |
| LB | 0/1 | LB指令译码信号 |
| REMU | 0/1 | REMU指令译码信号 |
| URET | 0/1 | URET指令译码信号 |
| ecall | 0/1 | ecall指令译码信号 |

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图2.2所示。

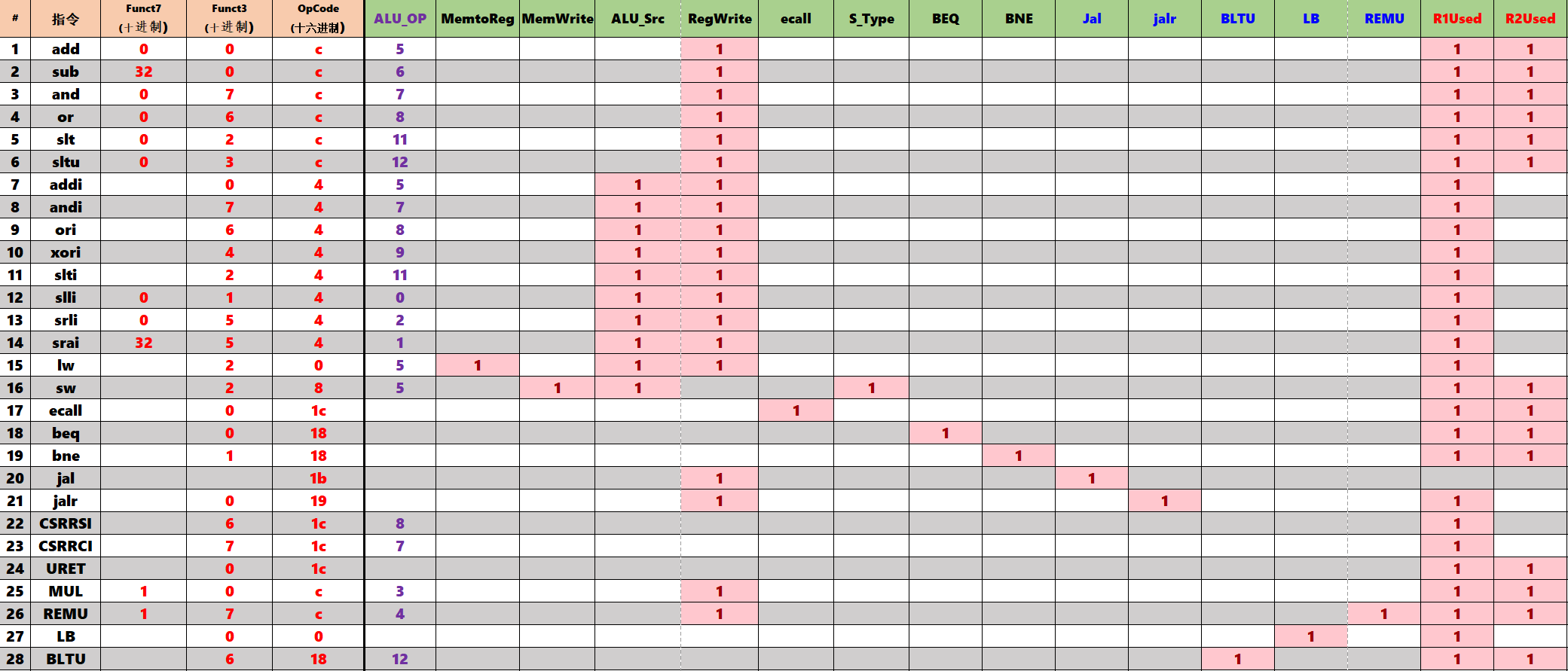


图2.2控制信号表

## 中断机制设计

### 总体设计

按照任务书要求，需要为所设计的单周期CPU添加中断机制。要求能够支持3个Logisim按钮触发的外部按键中断源，编号分别为1、2、3且中断优先级1<2<3。

对于外部中断有硬件实现的中断与软件实现的中断两种，在这里采用的是硬件实现的方法。按照要求，需要设计单级中断与多级中断两种中断。

首先对于单级中断的硬件实现，需要在进入中断时保存CPU的状态即保护现场，将中断返回地址即当前的PC值保存在EPC中，以便在中断部分执行完毕后退出时可以继续之前的进程。除此之外还需要设置相应的中断识别逻辑和中断相关指令数据通路等。另外对于流水单级中断而言，需要保存的中断地址为MEM段的PC。

对于多级中断的实现，需要在单级中断的基础上另外考虑中断优先级的问题，对于正在执行中的优先级较高的中断，优先级较低的中断需要排队等待，而优先级更高的中断可以直接执行。另外在多级中断中需要设计CSRRCI和CSRRSI指令的数据通路。

### 硬件设计

对于单级中断，将所提供的中断按键参考电路进行封装后产生中断信号，使用优先编码器获得需要执行的中断号。对于所要求的三个优先级不同的中断，使用所提供的rars工具对汇编文件进行操作从而获得三个对应的中断服务程序地址，使用多路选择器并用所获得的需要执行的中断号来获得所需要的对应地址。使用中断使能寄存器IE和异常程序计数器mEPC来保护现场也即CPU中断时的状态，使用多路选择器，通过URET信号和INT信号来控制多路选择器从而获得下一地址。存放返回地址的寄存器mEPC需要在INT信号为1即发生了中断时才能更新。

对于多级中断，在单级中断的基础上，由于有三个不同优先级的中断产生，需要使用三个各自存放这些中断的返回地址的寄存器并使用优先编码器来确定返回地址。

## 流水CPU设计

### 总体设计

根据所提供的指令流水线说明文档，流水线分为五个阶段：取指令（IF）、指令译码（ID）、指令执行（EX）、访存（MEM）以及写回（WB）五个阶段，每一条指令依次经过这五个阶段进行执行。为锁存本段所处理完成的所有数据或结果，需要在每个阶段中间增加一个流水寄存器，从而达到形成流水线、各阶段各自执行对应的指令并处理对应的数据的效果。

### 流水接口部件设计

流水接口部件的功能为：根据使能信号，在一个时钟周期内控制所接受的各个信号的存储与传递；根据清零信号，将本部件中所存储的信号全部清零。由于每一个流水线阶段所需要传递与接收的信号各不相同，故每一个流水接口部件需要各自单独设计。

每一个流水线接口部件所共有的信号接口为：CLK、PC、PC+4、CLR（清零信号）以及ABL（使能信号）。

结合理想流水线、气泡流水线、重定向流水线以及动态分支预测所各需要实现的功能，根据不同流水线阶段所需要传递的信号，不同流水线接口部件所需要实现的信号接口如表2.5所示。

表 2.5流水线接口部件所保存传递的信号

| 部件 | 信号 |
| --- | --- |
| IF/ID | 指令字,PJ |
| ID/EX | HALT,PJ,ECALL,REMU,LB,URET,JALR,JAL,BEQ,BLTU,BNE,RegWrite,MemToReg,MemWrite,ALUop,R1F,R2F,ASB,R1Data,R2Data,ALUB,RW,BA,JALA,JALRA,LBA |
| EX/MEM | HALT,REMU,LB,URET,JALR,JAL,RegWrite,MemToReg,MemWrite,R2Data,R2,ALUR,RW,LBA |
| MEM/WB | HALT,REMU,LB,URET,JALR,JAL,RegWrite,MemData,R2,LB01,MData,RW |

### 理想流水线设计

根据所设计的流水线接口部件对单周期CPU进行划分，根据各部件的功能将其划分至各自的流水线阶段，中间放置所设计的接口部件进行连接，实现信号与数据的储存与传递。理想流水线顶层试图如图2.3所示。

对于信号的传递，根据各信号所作用的不同流水线阶段，增加对应的隧道，使信号能够正确地实现相应的功能。对于数据的传递，不同阶段所使用的数据应使用该阶段的上一个接口部件中所传出的数据。

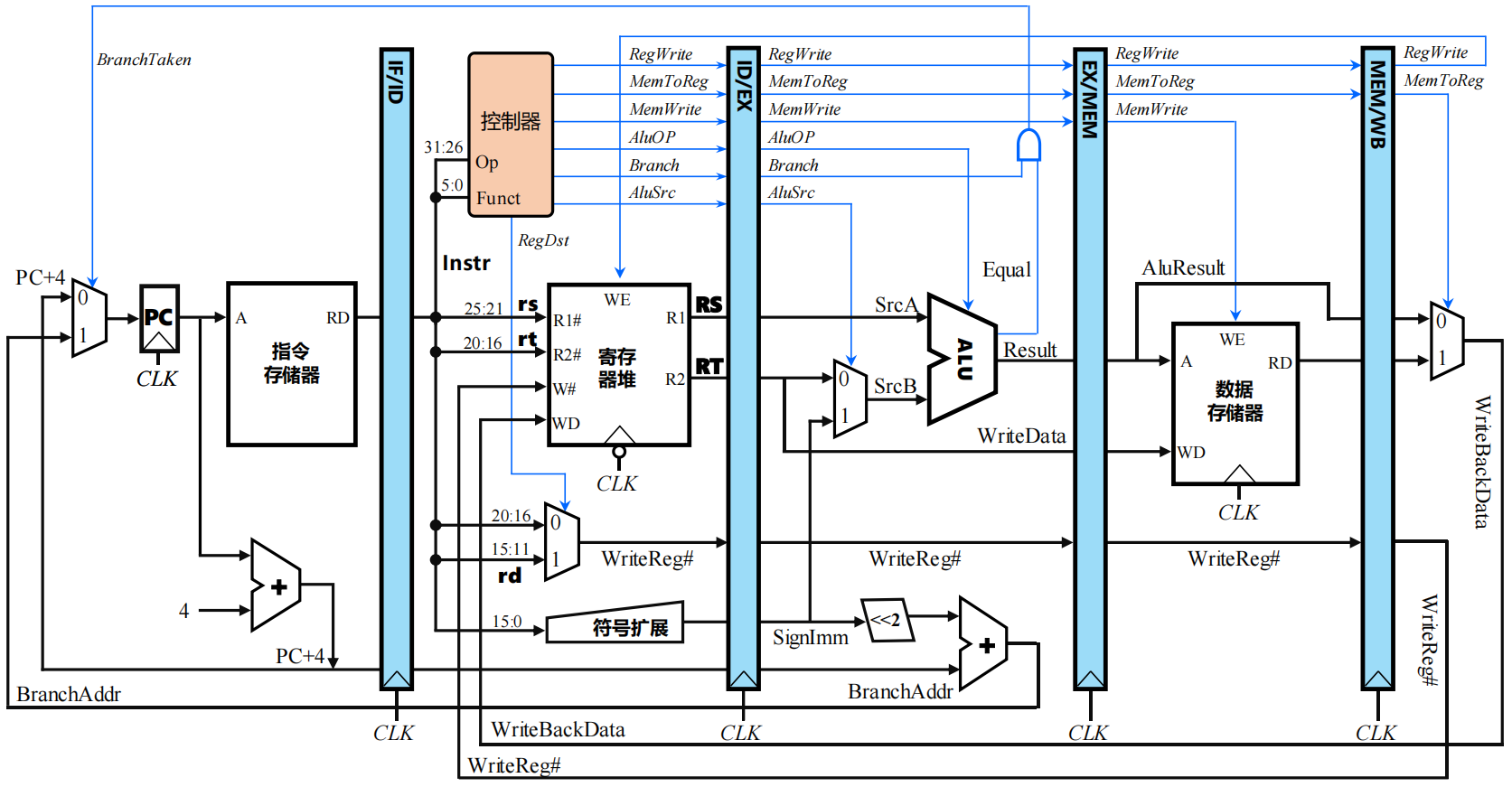


图2.3理想流水线顶层视图

## 气泡式流水线设计

由于存在控制冲突的问题，在理想流水线的基础上增加气泡机制。顾名思义，气泡机制即在发生了控制冲突的时候，在冲突的两条指令之间插入空白指令产生一个“气泡”，从而解决控制冲突问题。这一机制在解决控制冲突问题的同时也会降低流水线的性能、增加周期数。

气泡机制的实现需要：

1. 设计RegUseCon子电路，用来判断R1和R2被使用的情况。
2. 使用该子电路结合所给出的冲突检测逻辑设计Bubble子电路，用以检测是否需要产生气泡。
3. 修改指令寄存器，增加气泡产生信号对其的控制。
4. 根据气泡产生的流水线阶段，修改清零信号和使能信号的控制。

气泡流水线顶层视图如图2.4所示。

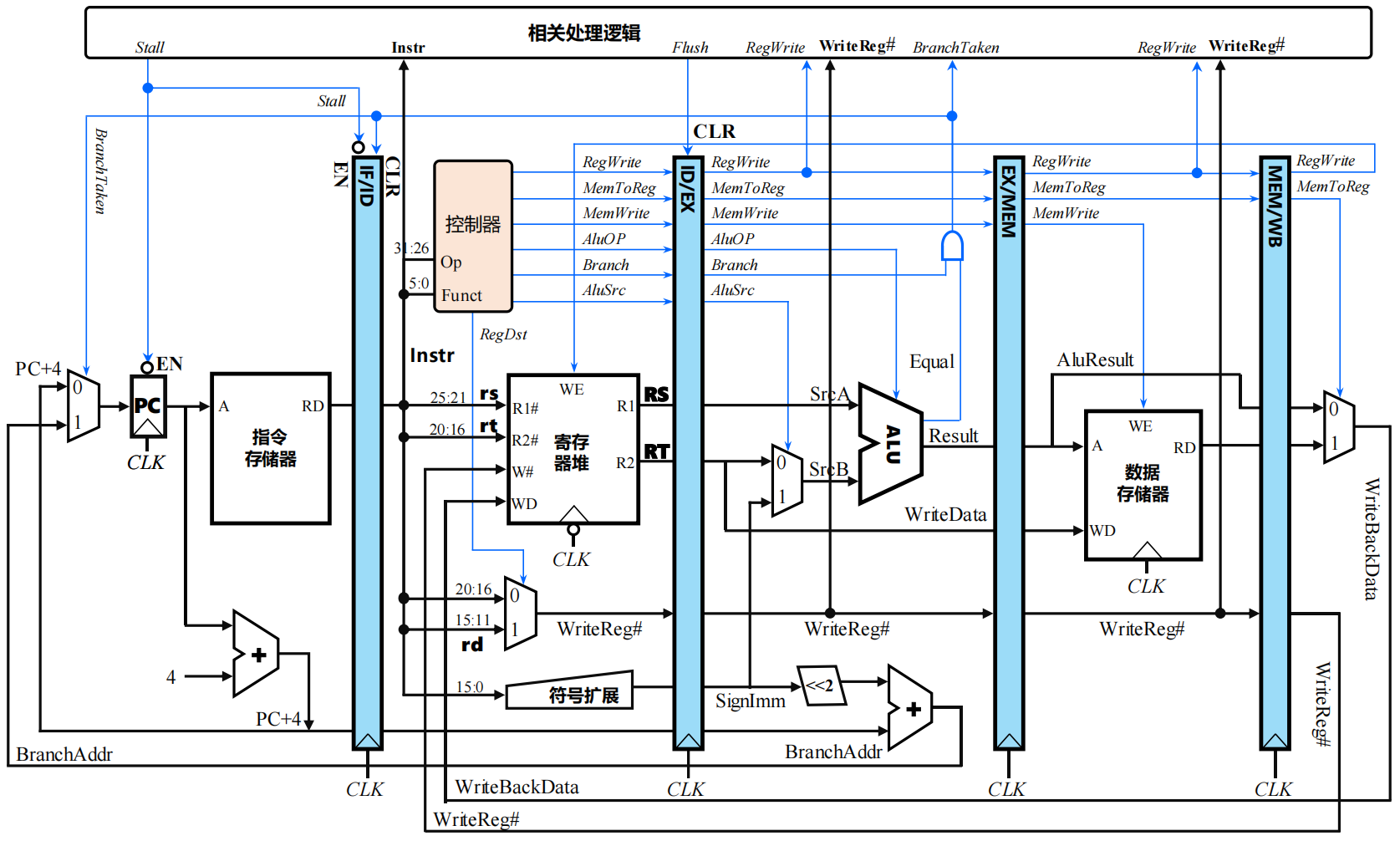


图2.4气泡流水线顶层视图

## 重定向流水线设计

由于气泡流水线虽然解决了数据冲突的问题却降低了流水线性能，在其基础上进行修改为重定向流水线提升性能。重定向机制为在发生数据冲突时将下一时钟周期需要写回寄存器堆的数据重定向至EX段，从而解决数据冲突问题。

重定向流水线会出现Load-Use问题，即load指令的后一条指令需要使用相关寄存器，此时不能使用重定向机制，仍然需要插入气泡来解决。

重定向机制的实现需要：

1. 使用已设计的RegUseCon子电路，结合所给的重定向逻辑设计Redirect子电路，生成相应的重定向信号。
2. 设计EX段重定向电路，使相应数据得到正确传输。
3. 修改指令寄存器，增加LoadUse气泡产生信号对其的控制。
4. 根据LoadUse气泡产生的流水线阶段，修改清零信号和使能信号的控制。

重定向流水线顶层视图如图2.5所示。

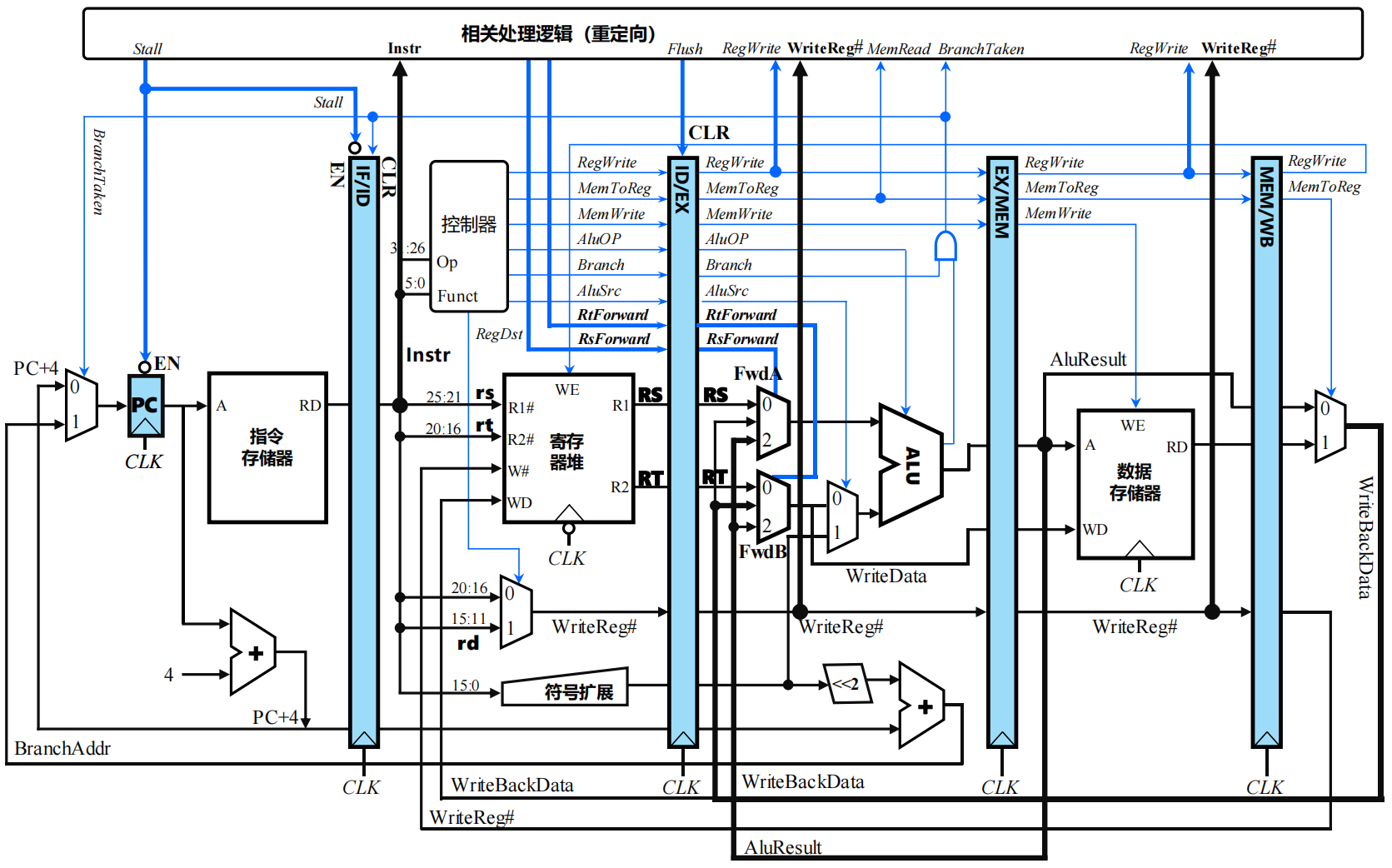


图2.5重定向流水线顶层视图

## 动态分支预测机制

在使用重定向机制优化了流水线的性能的基础上，再次使用动态分支预测机制进一步优化性能。动态分支预测即根据分支指令的分支跳转历史，不断地对预测策略进行动态调整。按照要求，应设计分支预测缓冲器BTB进行实现。BTB表为一个全相联的cache，每个表项包括valid 位、分支指令地址，分支目标地址，分支预测历史位、置换标记。其中分支预测历史位为2比特，状态转换图如图2.6所示。使用LRU算法对分支地址进行淘汰。

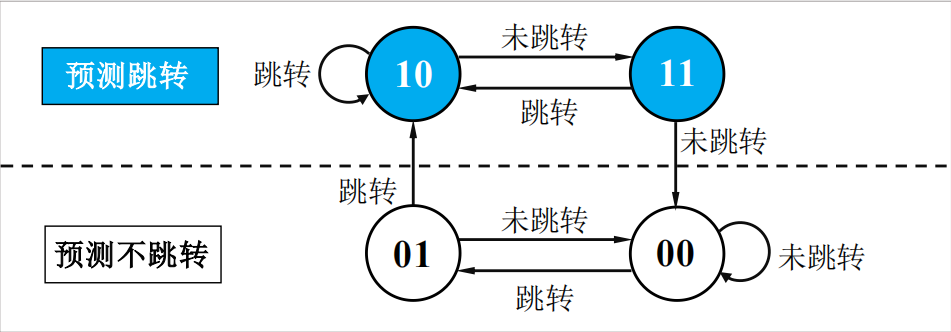


图2.6双预测位状态转换图

采用了动态分支预测机制后流水线运行情况如图2.7所示。



图2.7流水线运行情况

动态分支预测机制的实现需要：

1. 根据全相联映射cache机制设计BTB表。
2. 设计分支指令检测电路，生成分支指令检测信号。
3. 设计地址选择电路，将预测地址与原地址进行选择。
4. 修改对应阶段的清零信号的控制。

# 详细设计与实现

## 单周期CPU 实现

### 主要功能部件实现

1. 程序计数器（PC）

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。Halt为停机信号，将此控制信号通过非门取反之后和时钟相与，当需要进行停机时，Halt控制信号为1，经过非门之后为0，与时钟信号相与，屏蔽时钟信号，使整个电路停机。

由于存在分支跳转指令，需要使用对应信号控制多路选择器进行对下一地址的选择，将选择后的地址作为寄存器的输入。

如图 3.1所示。

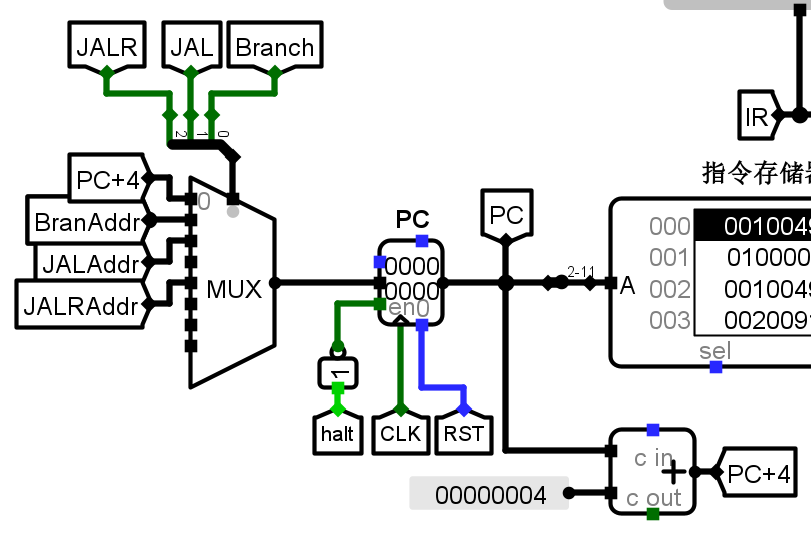


图 3.1程序计数器（PC）

1. 指令存储器（IM）

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 3.2所示。

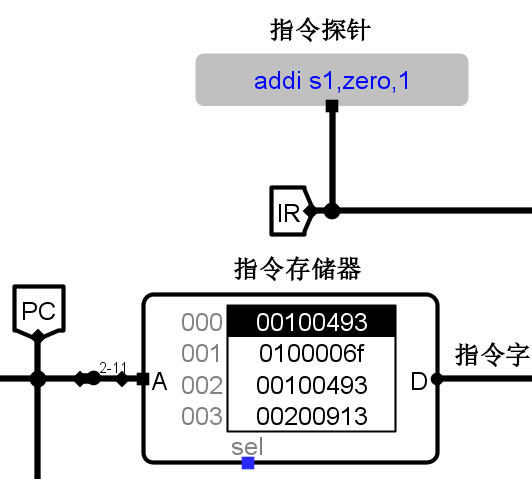


图 3.2指令存储器（IM）

1. 运算器（ALU）

使用所提供的运算器ALU子电路。该子电路根据4比特的AluOP信号进行对于运算功能的选择，对输入的X和Y进行相对应的运算操作，输出运算结果和比较结果信号。子电路如图3.3所示。

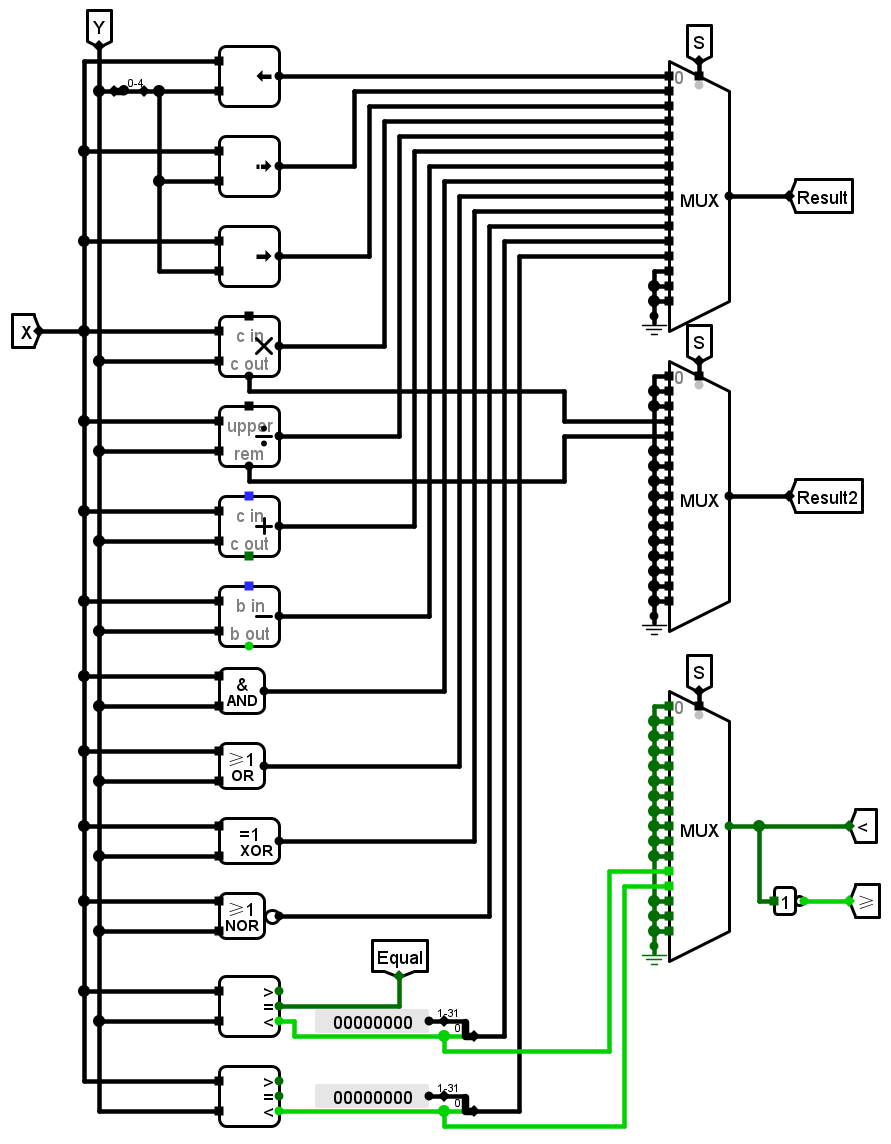


图 3.3运算器ALU

1. 寄存器堆（RF）

使用所提供的CS3410 Components包中的Register File部件设计寄存器堆子电路。该子电路由写使能信号WE控制数据写入，将数据Din写入对应的寄存器W#中。并按照所输入的寄存器序号输出相应的寄存器数据。子电路如图3.4所示。

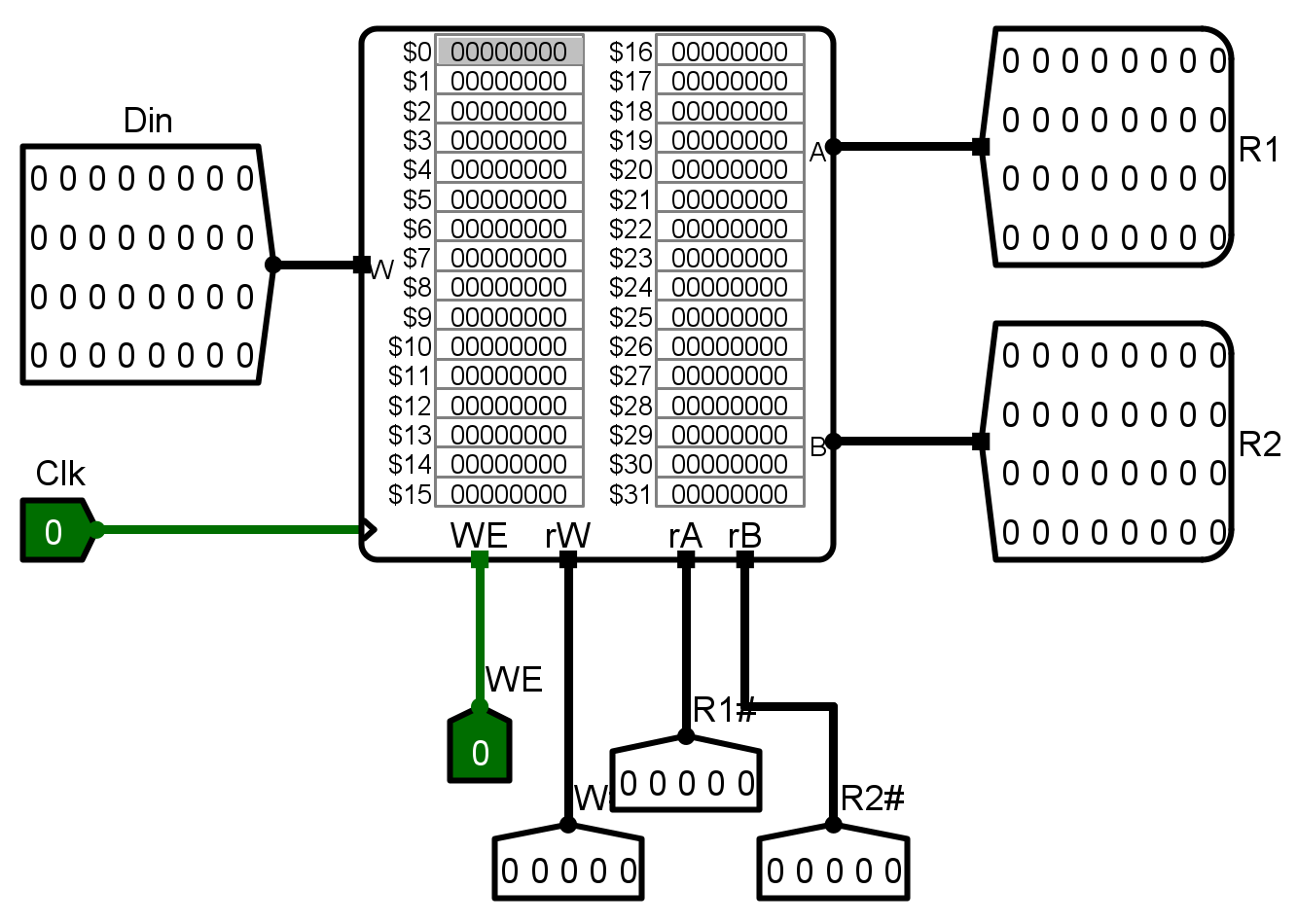


图3.4寄存器堆RF

### 数据通路的实现

本次课程设计采用的工程化的设计模式，一次性构建所有的数据通路。主要实现方法为，对于每一条指令，将其改写成RTL（Register Transfer Level），忽略控制类信号，仅保留数据类信号，根据RTL功能填写对应指令的数据通路表，描述五大部件之间的连接关系，记录各部件输入端数据来源。

根据总体方案设计中数据通路设计那一小节的详细内容，具体分析每一条指令在执行过程中各个主要部件的输入和输出端口的连接，完成指令系统数据通路表的填写，如表 3.1所示。

表 3.1指令系统数据通路表

| 指令 | PC | IM | RF | | | | ALU | | | DM | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| R1# | R2# | W# | Din | A | B | OP | Addr | Din |
| ADD | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 5 |  |  |
| SUB | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 6 |  |  |
| AND | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 7 |  |  |
| OR | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 8 |  |  |
| SLT | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 11 |  |  |
| SLTU | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 12 |  |  |
| ADDI | PC+4 | PC | rs |  | rd | alu/DM | r1 | IMM | 5 | alu | rt |
| ANDI | PC+4 | PC | rs |  | rd | alu/DM | r1 | IMM | 7 | alu | rt |
| ORI | PC+4 | PC | rs |  | rd | alu/DM | r1 | IMM | 8 | alu | rt |
| XORI | PC+4 | PC | rs |  | rd | alu/DM | r1 | IMM | 9 | alu | rt |
| SLTI | PC+4 | PC | rs |  | rd | alu/DM | r1 | IMM | 11 | alu | rt |
| SLLI | PC+4 | PC | rs |  | rd | alu/DM | r1 | IMM | 0 | alu | rt |
| SRLI | PC+4 | PC | rs |  | rd | alu/DM | r1 | IMM | 2 | alu | rt |
| SRAI | PC+4 | PC | rs |  | rd | alu/DM | r1 | IMM | 1 | alu | rt |
| LW | PC+4 | PC | rs |  | rd | alu/DM | r1 | IMM | 5 | alu | rt |
| SW | PC+4 | PC | rs | rt |  |  | r1 | r2 | 5 | alu | rt |
| BEQ | PC+4/IMM | PC | rs | rt |  |  | r1 | r2 |  |  |  |
| BNE | PC+4/IMM | PC | rs | rt |  |  | r1 | r2 |  |  |  |
| JAL | PC+4/IMM | PC |  |  | rd |  |  |  |  |  |  |
| JALR | PC+4/IMM | PC |  |  | rd |  |  |  |  |  |  |
| MUL | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 3 |  |  |
| REMU | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 4 |  |  |
| LB | PC+4 | PC | rs |  | rd | alu/DM | r1 | IMM |  |  |  |
| BLTU | PC+4/IMM | PC | rs | rt |  |  | r1 | r2 | 12 |  |  |

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，如图3.5所示。

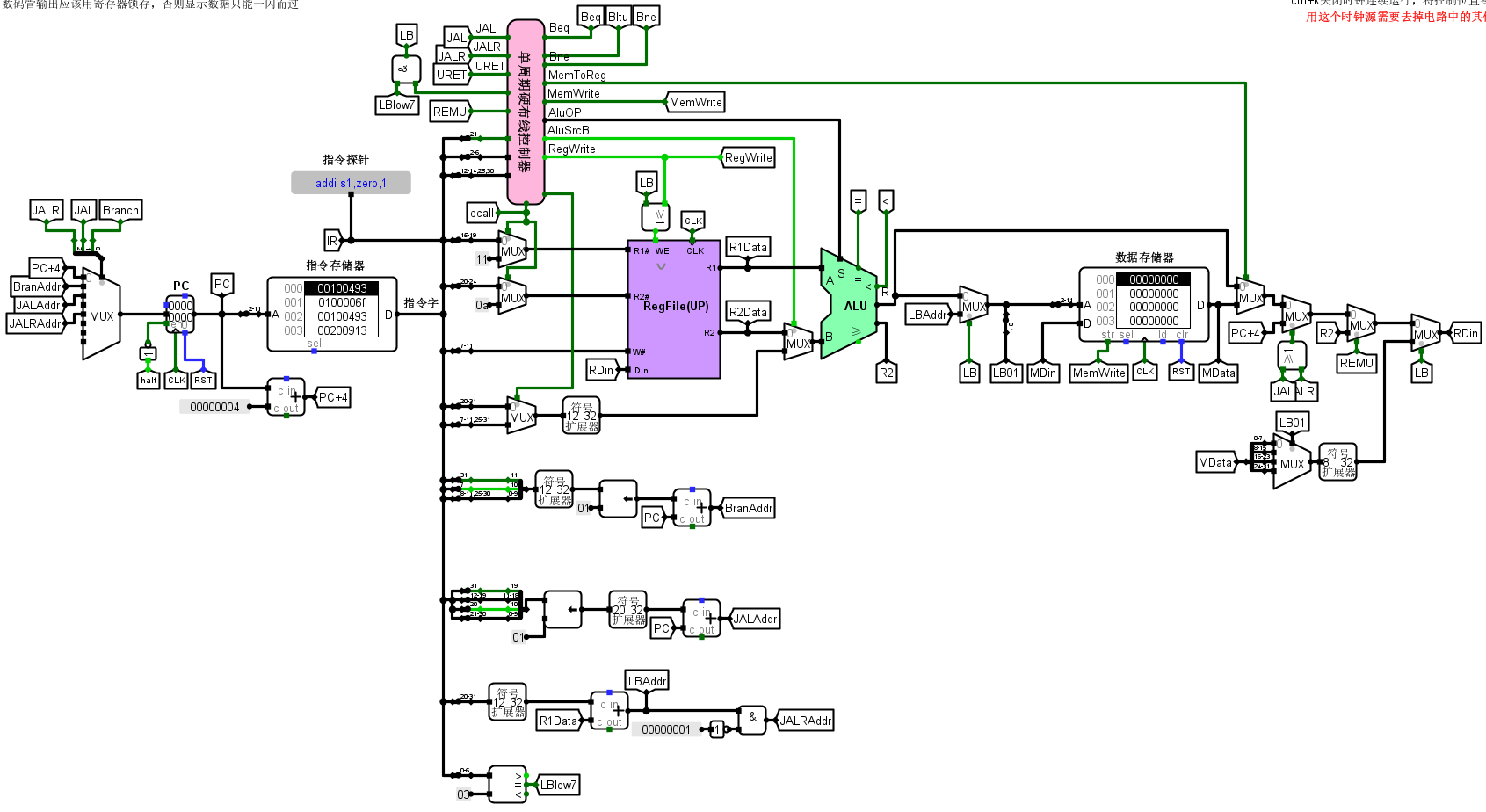


图 3.5单周期CPU数据通路

### 控制器的实现

根据总体方案设计中控制器的设计那一小节的相关内容，使用填写表格所生成的各信号的表达式，利用Logisim自带的组合逻辑生成电路功能分别生成运算控制器和控制信号生成两个子电路，在Logisim上进行单周期硬布线控制器的具体实现。如图3.6所示。

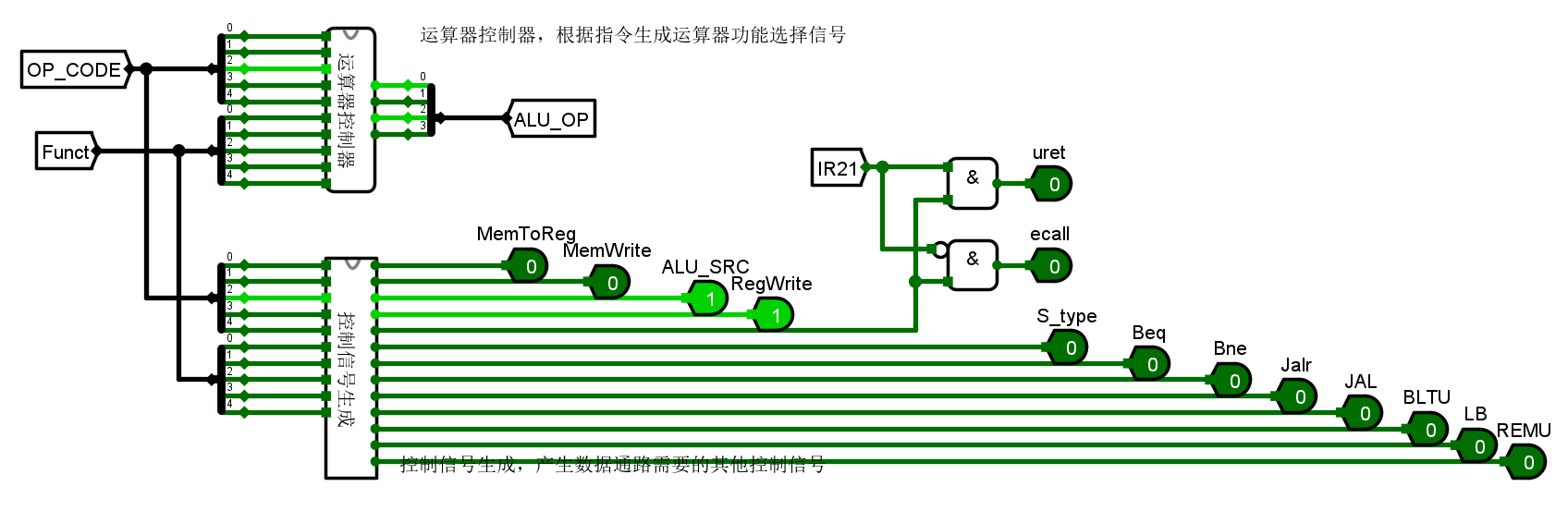


图3.6单周期硬布线控制器

## 中断机制实现

### 单级中断

使用所提供的中断信号采样电路进行封装作为中断信号产生部件，该子电路如图3.7所示。

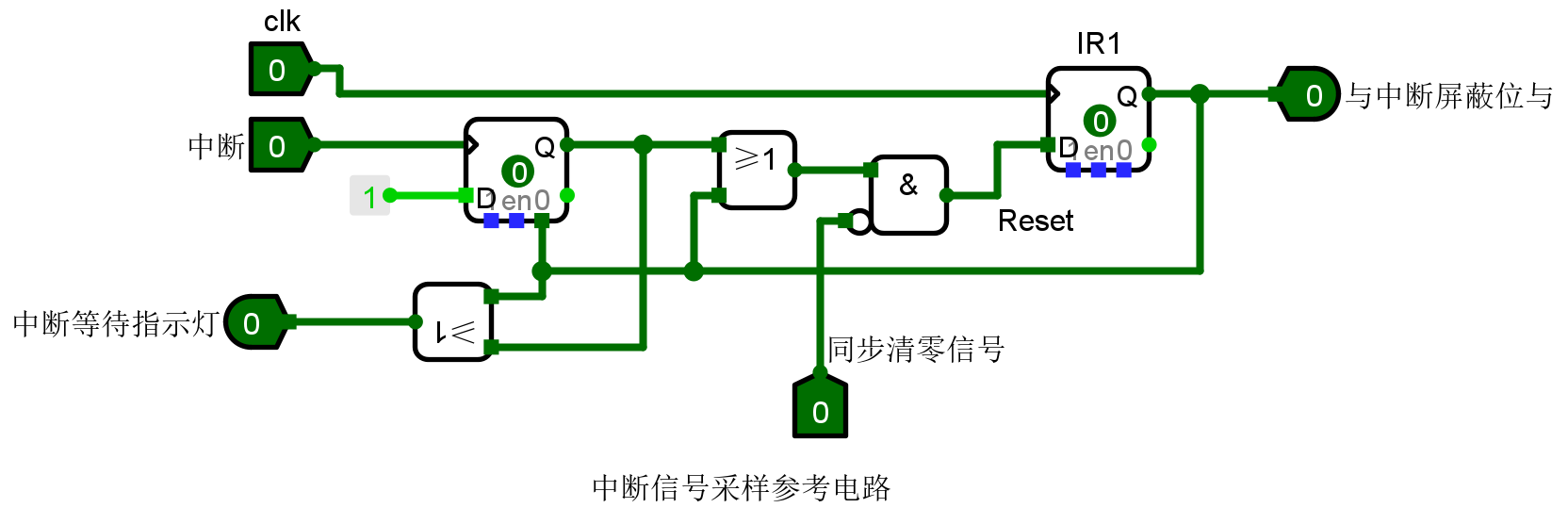


图3.7中断信号产生电路

使用优先编码器获取中断号，将该中断号用于控制多路选择器来选择对应的中断地址，并用于在中断完成后用于控制译码器以恢复现场。用所提供的rars工具对汇编文件进行操作从而获得三个对应的中断服务程序地址，分别为0x000030ac、0x00003150和0x000031f4。

使用D除法器控制中断信号INT，用以控制触发器的使能端。对于原PC值用mEPC寄存器进行保存，使能端接入INT信号以控制。最后使用多路选择器来选择下一地址。

单级中断的实现部分如图3.8所示。

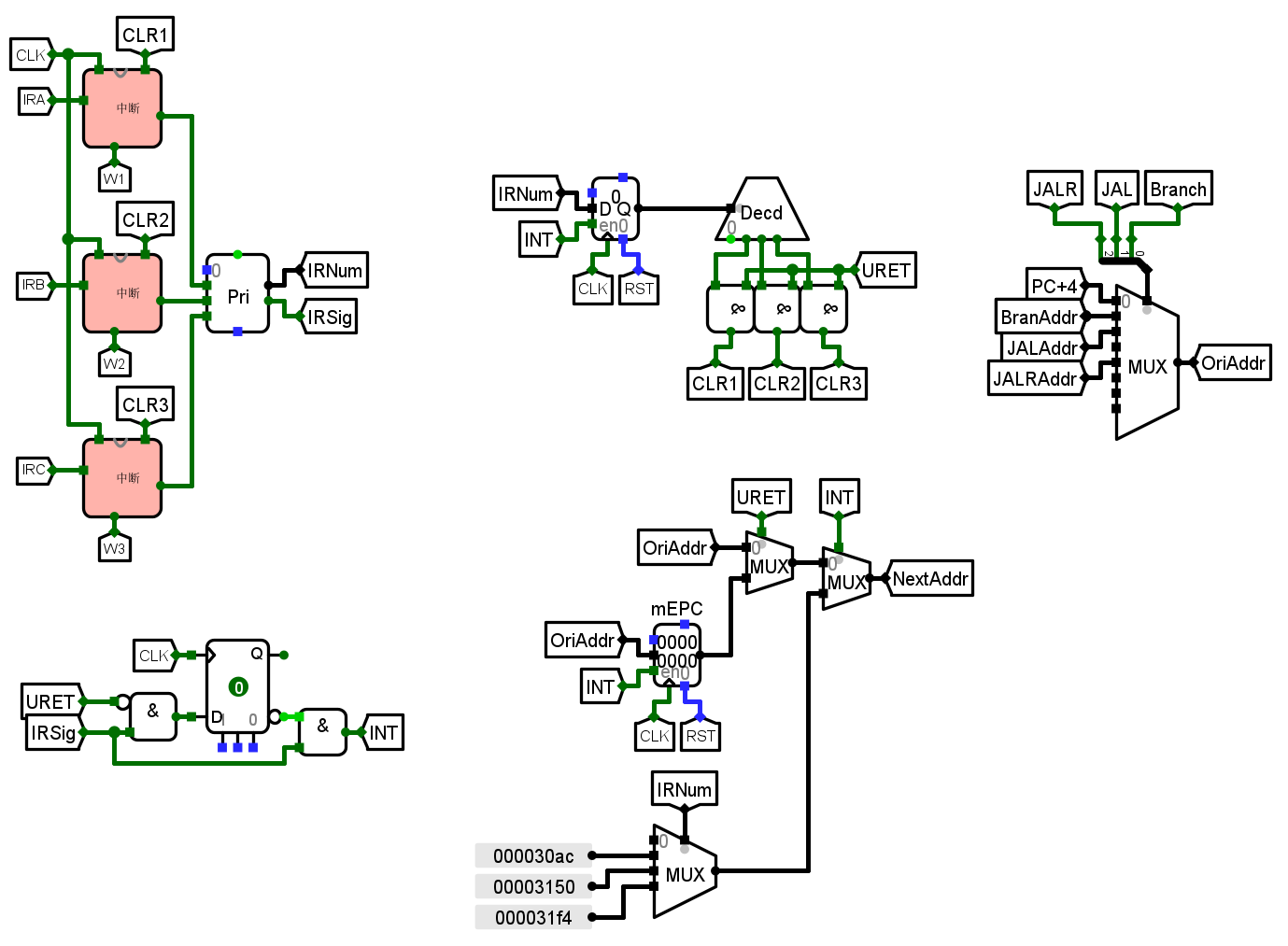


图3.8单级中断实现部分

单级中断的总体如图3.9所示。

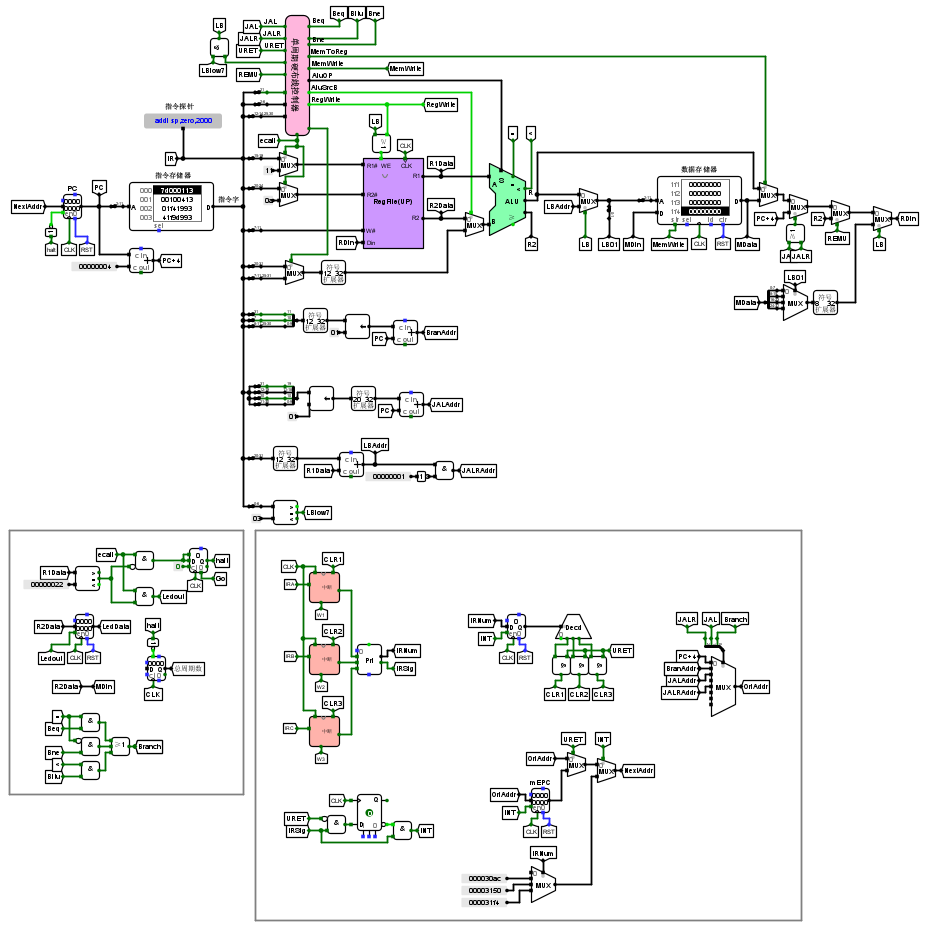


图3.9 单级中断

### 多级中断

多级中断电路的中断信号产生电路与单级中断相同。

在单级中断的基础上，需要增加多级嵌套功能以及CSRRCI指令和CSRRSI指令的数据通路。

CSRRCI指令和CSRRSI指令的数据通路如图3.10所示。

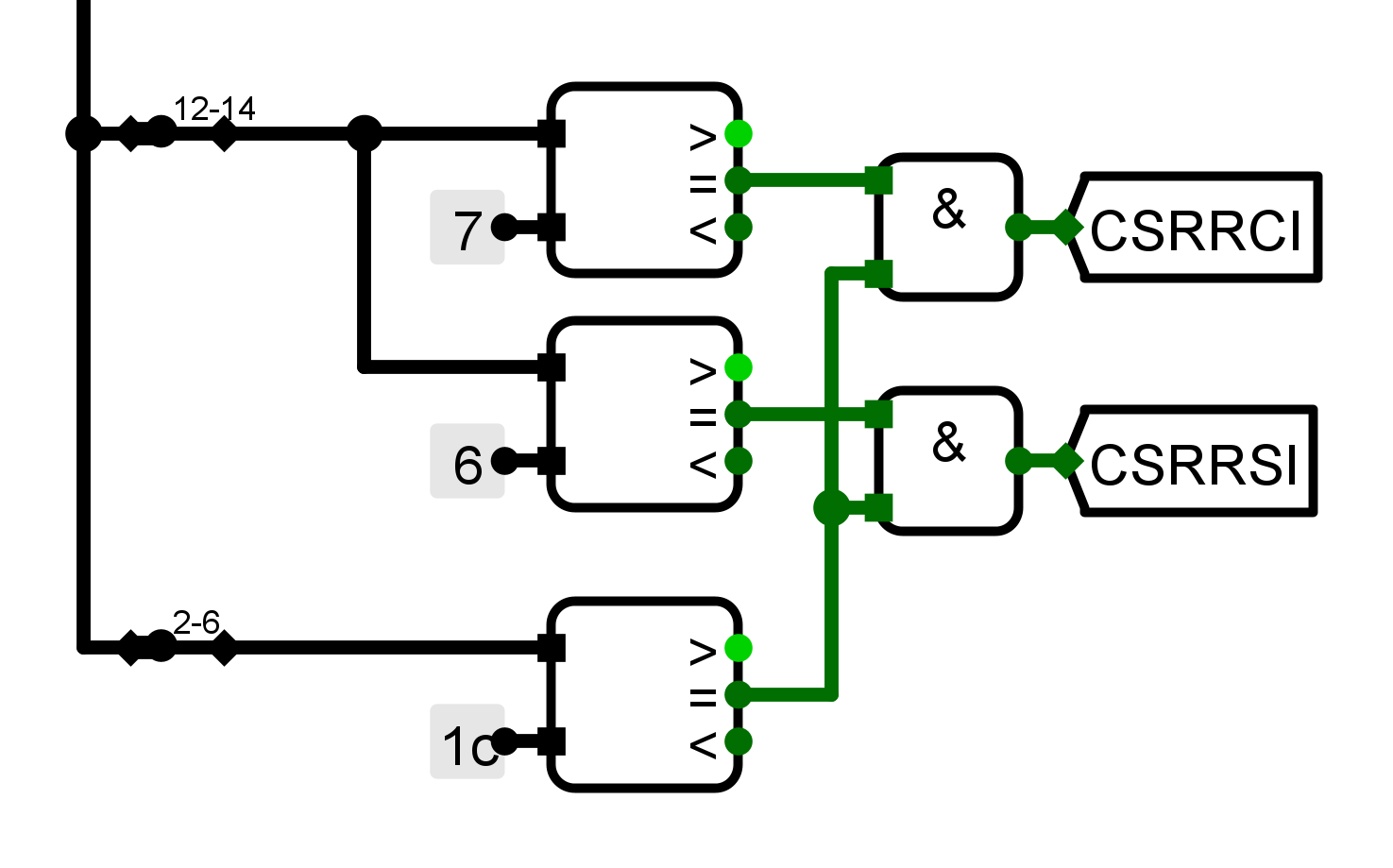


图3.10 CSRRCI和CSRRSI指令的数据通路

在多级嵌套中，需要设计中断号的优先选择、现场的保护与恢复以及开关中断。使用优先编码器实现中断号的优先选择；使用三组多路选择器和寄存器的组合形成硬件堆栈保护现场，实现现场的保护与恢复功能；使用对应组合逻辑实现开关中断。该部分如图3.11所示。

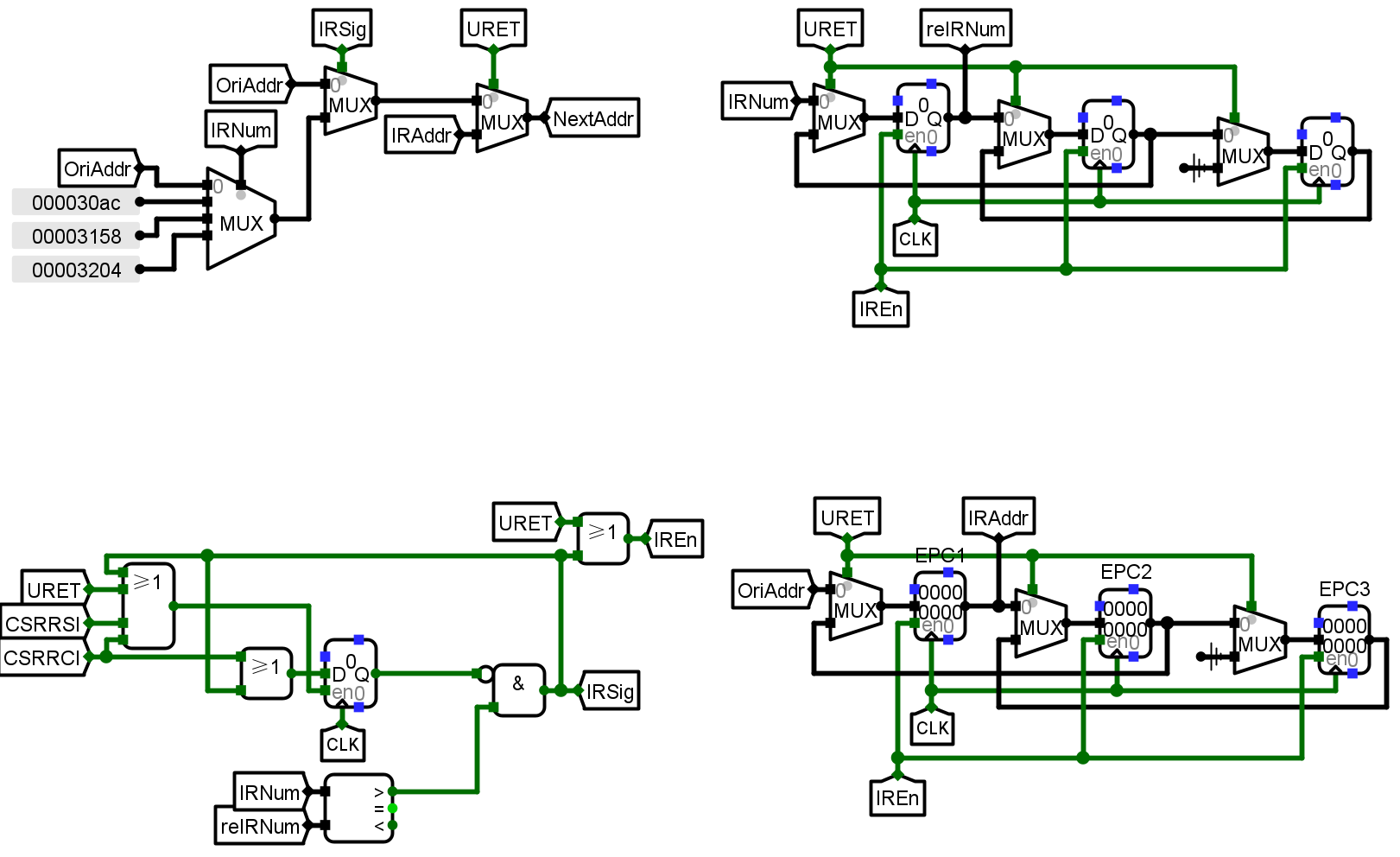


图3.11多级中断实现部分

多级中断的总体如图3.12所示。

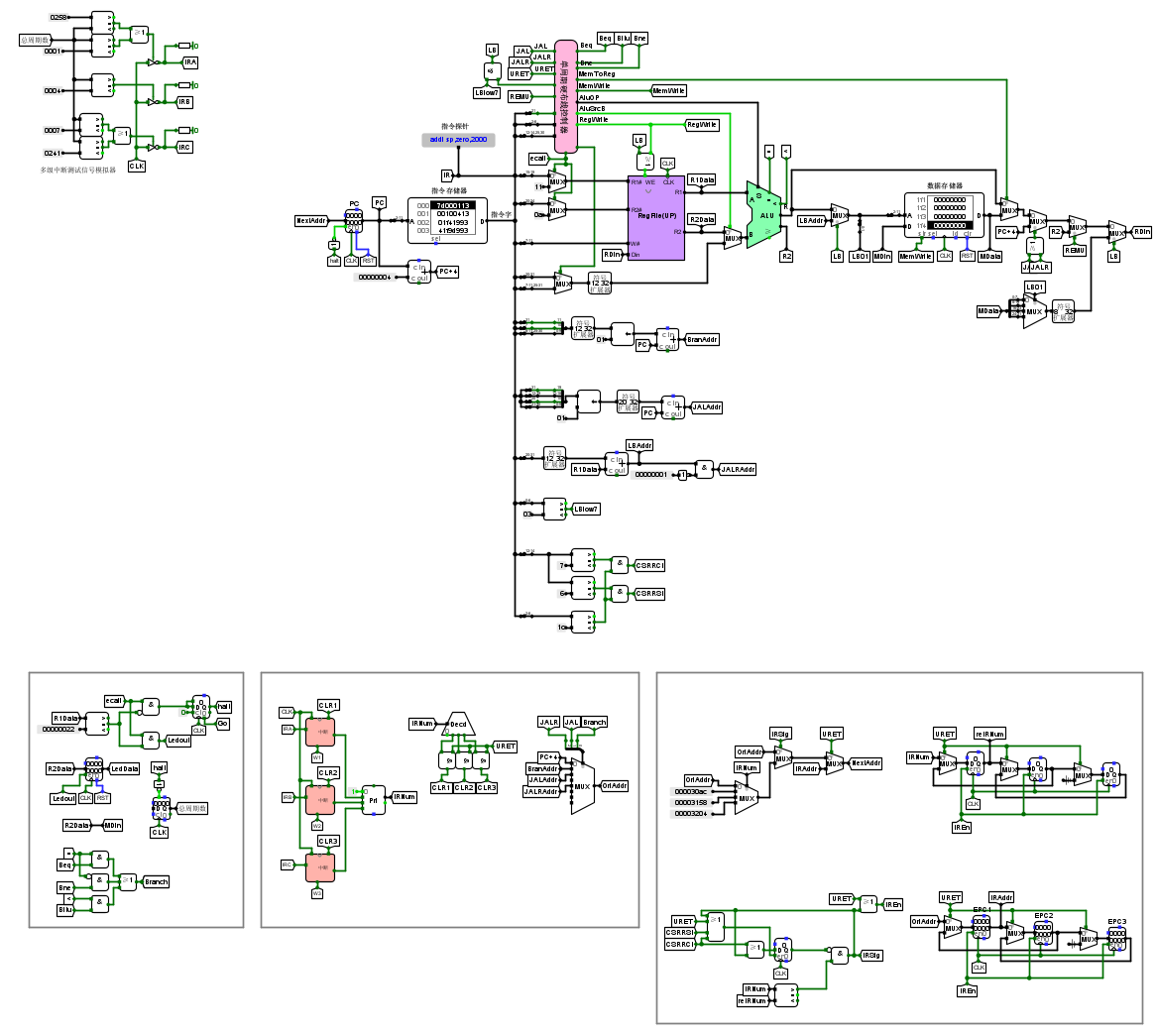


图3.12多级中断

### 流水中断

在重定向流水线的基础上，结合单级中断的电路即可得出流水中断电路。使用单级中断实现部分的电路进行中断控制，修改流水接口部件，增加中断信号INT对清零端的控制。流水中断的总体如图3.13所示。

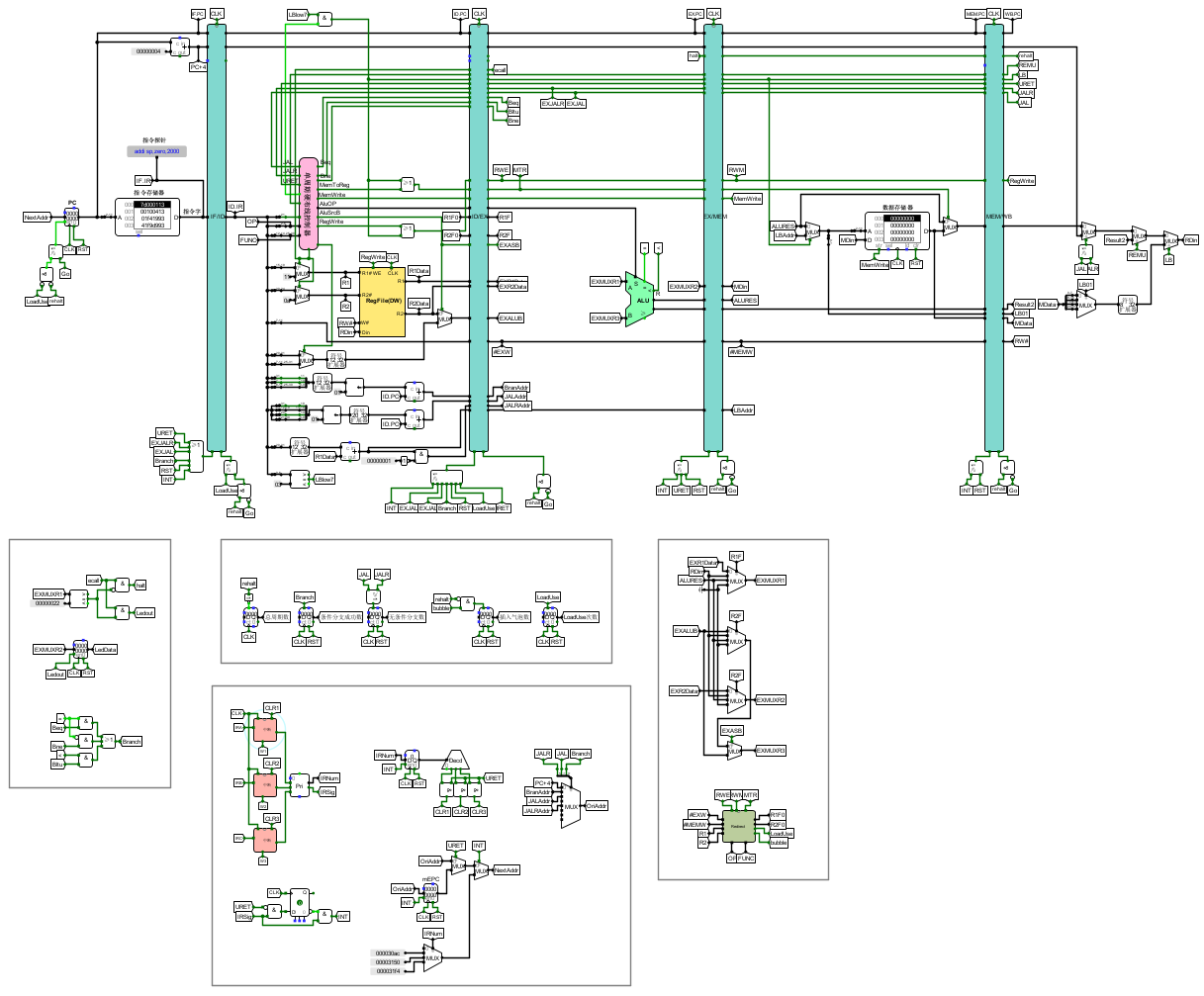


图3.13流水中断

## 流水CPU实现

### 流水接口部件实现

流水接口部件使用多路选择器和寄存器实现。CLR清零信号控制多路选择器选择对应数据和零，将选择结果送至寄存器输入，ABL使能信号控制寄存器的使能端，寄存器实现数据的存储与输出。将上述结构多次重复并设置对应的位宽即可实现每一阶段对应的数据与信号的储存与传递。

由于四个流水线接口部件均为该结构的重复，故选择其中之一进行说明，IF/ID部件如图3.14所示。

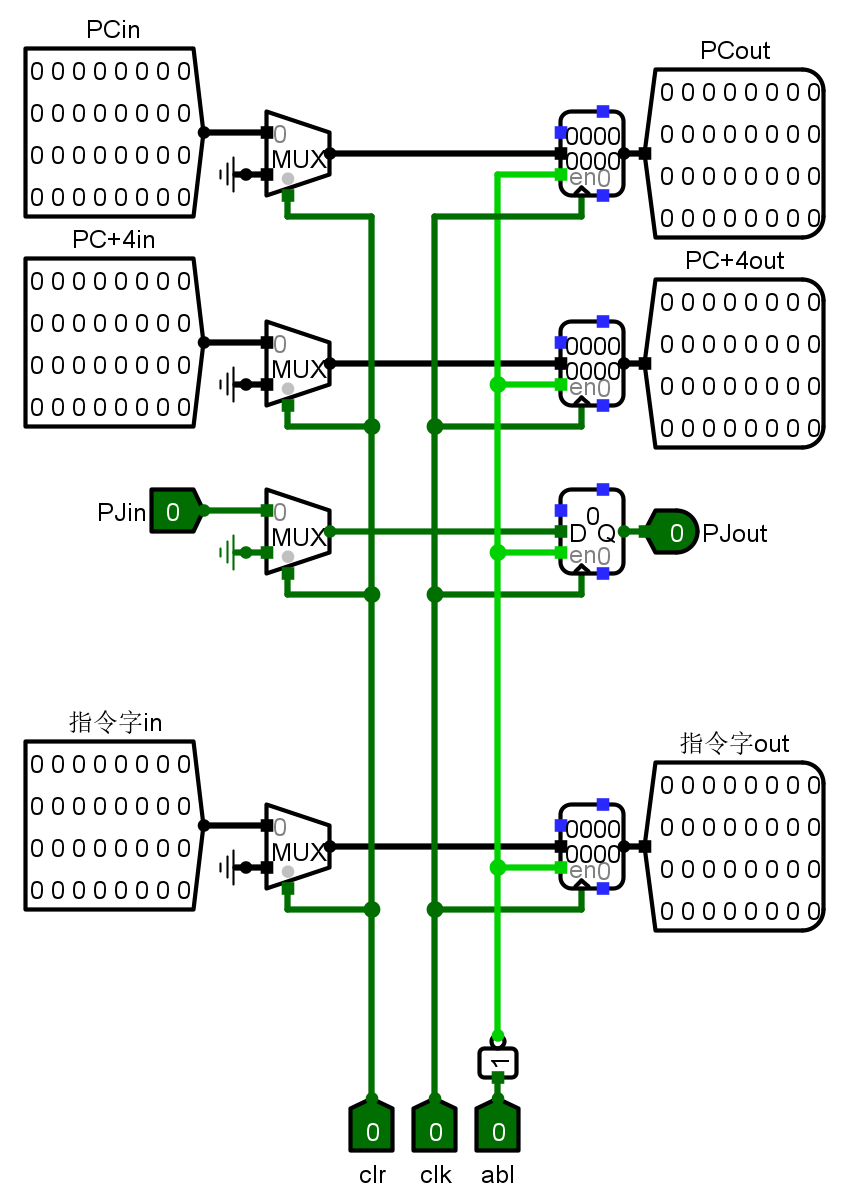


图3.14IF/ID端流水接口部件

### 理想流水线实现

根据2.3.3部分对于理想流水线的分析与设计，结合所设计的对应的流水线接口部件，对单周期CPU进行划分和修改，增加对应线路与隧道，完成理想流水线的实现，如图3.15所示。

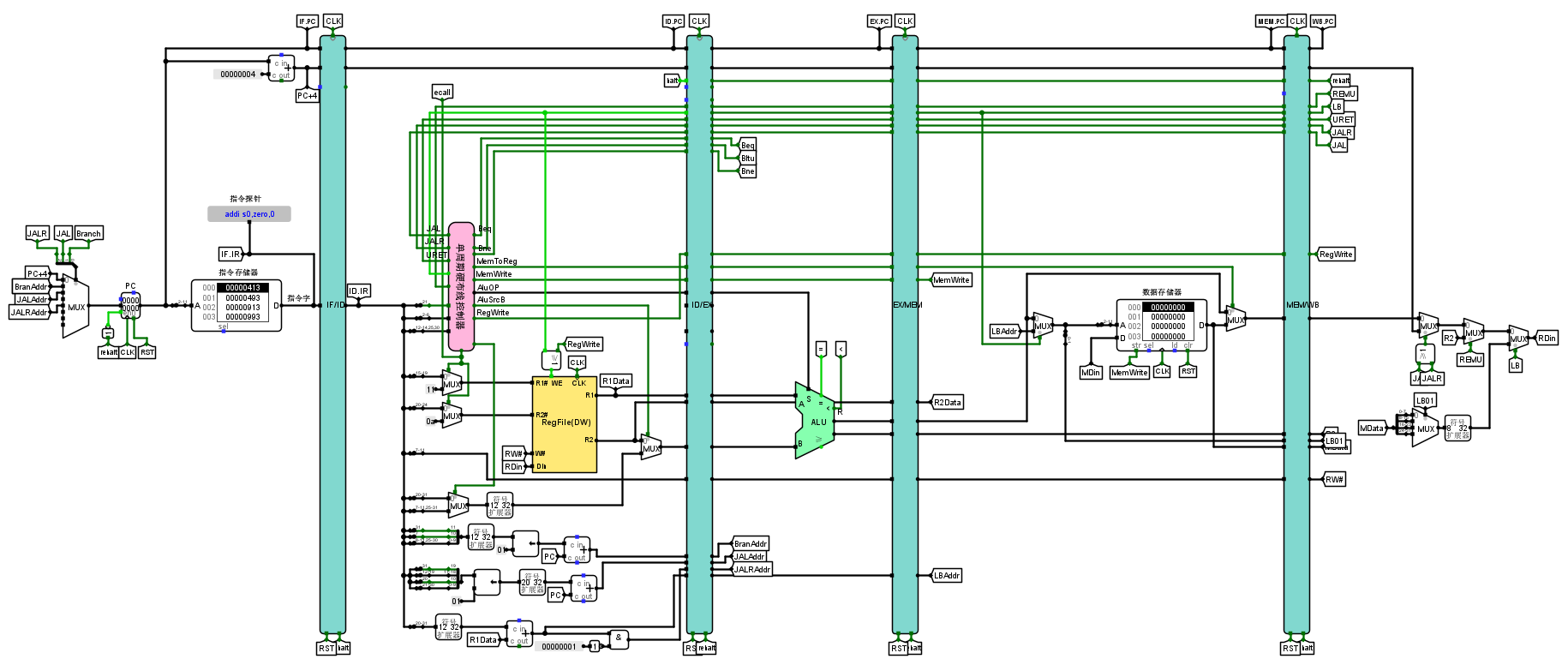


图3.15理想流水线

## 气泡式流水线实现

根据2.4部分对于气泡流水线的分析与设计，基于已经完成的理想流水线进行修改与优化。按照所给出逻辑设计的Bubble子电路如图3.16所示，其中黄色部件为使用Logisim组合逻辑自动生成的电路，用以检测R1和R2寄存器的使用情况。

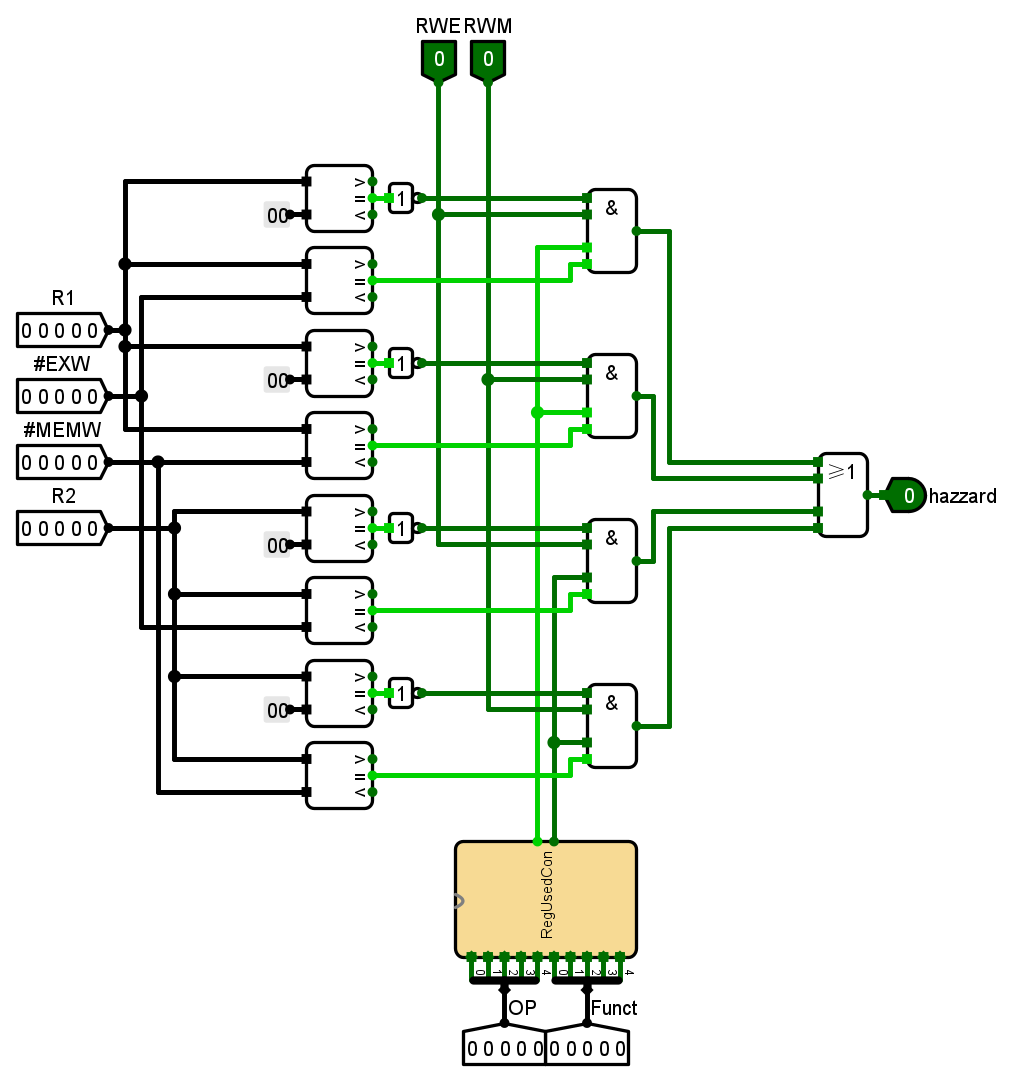


图3.16 Bubble子电路

使用所生成的hazzard信号控制气泡的插入，如图3.17所示。

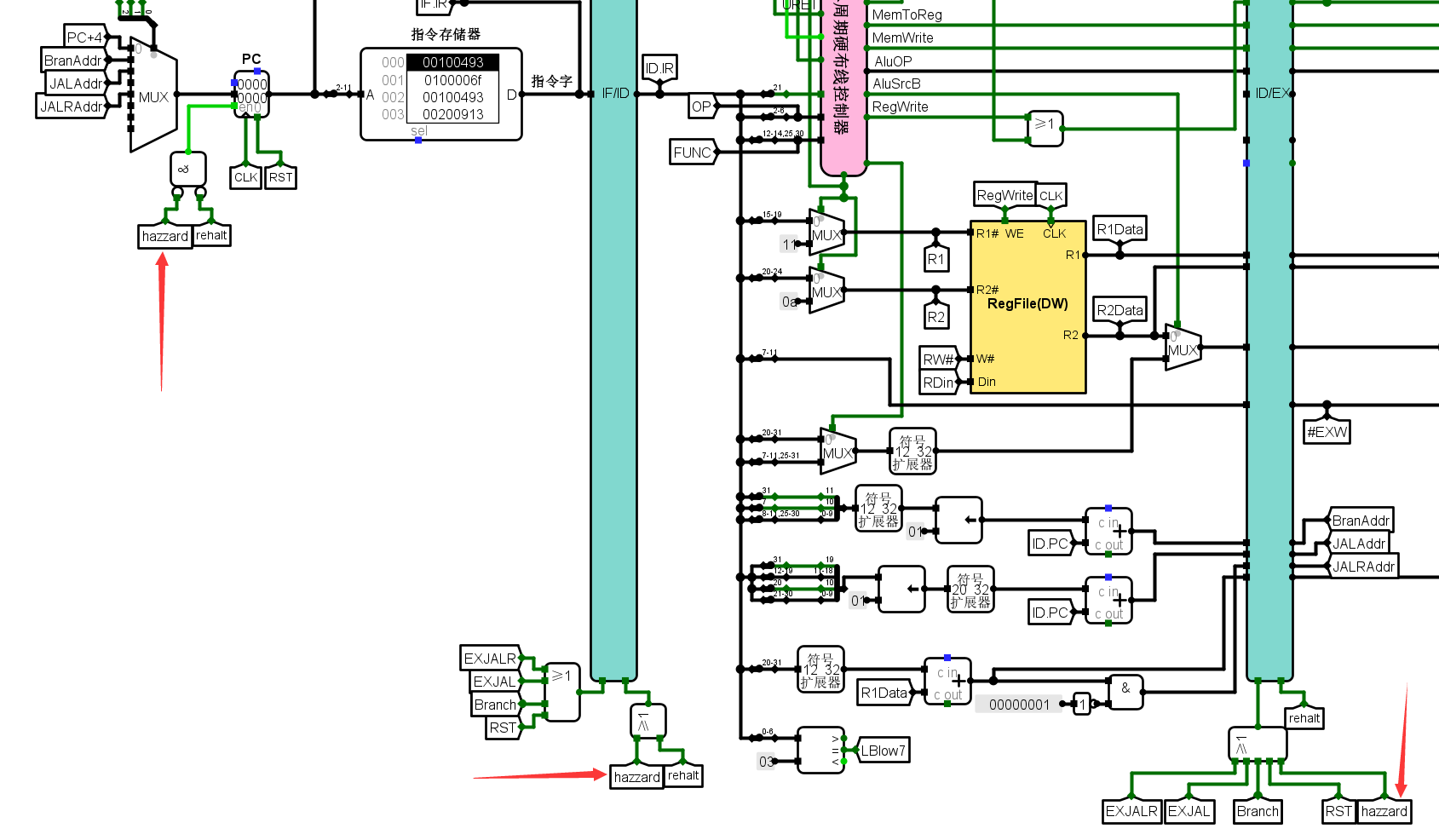


图3.17 hazzard信号对气泡的控制

从而完成气泡流水线的实现，如图3.18所示。

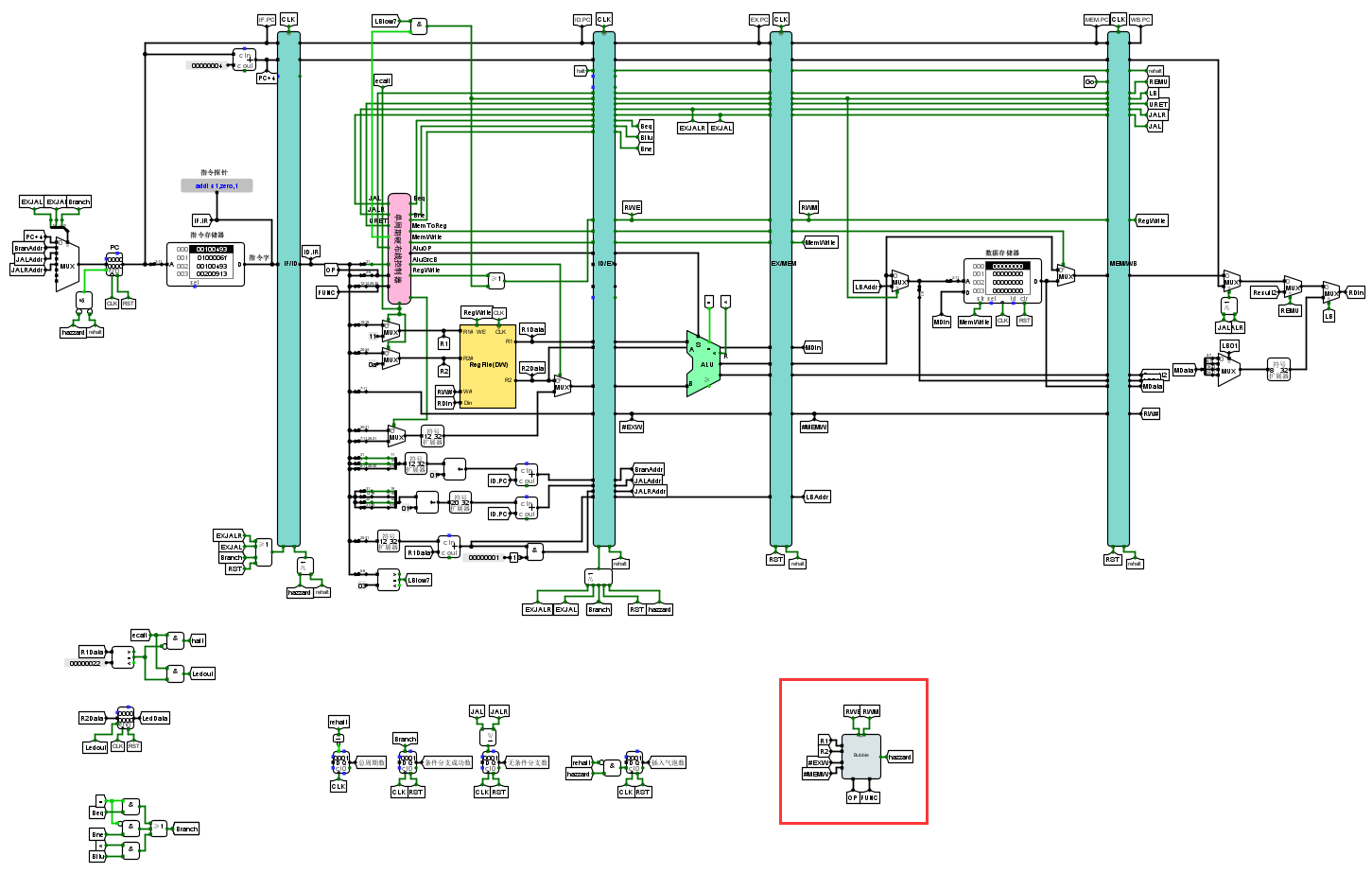


图3.18 气泡流水线

## 重定向流水线实现

根据2.5部分对于重定向流水线的分析与设计，基于已经完成的气泡流水线进行修改与优化。按照所给出逻辑设计的Redirect子电路如图3.19所示，黄色部件为RegUseCon子电路。

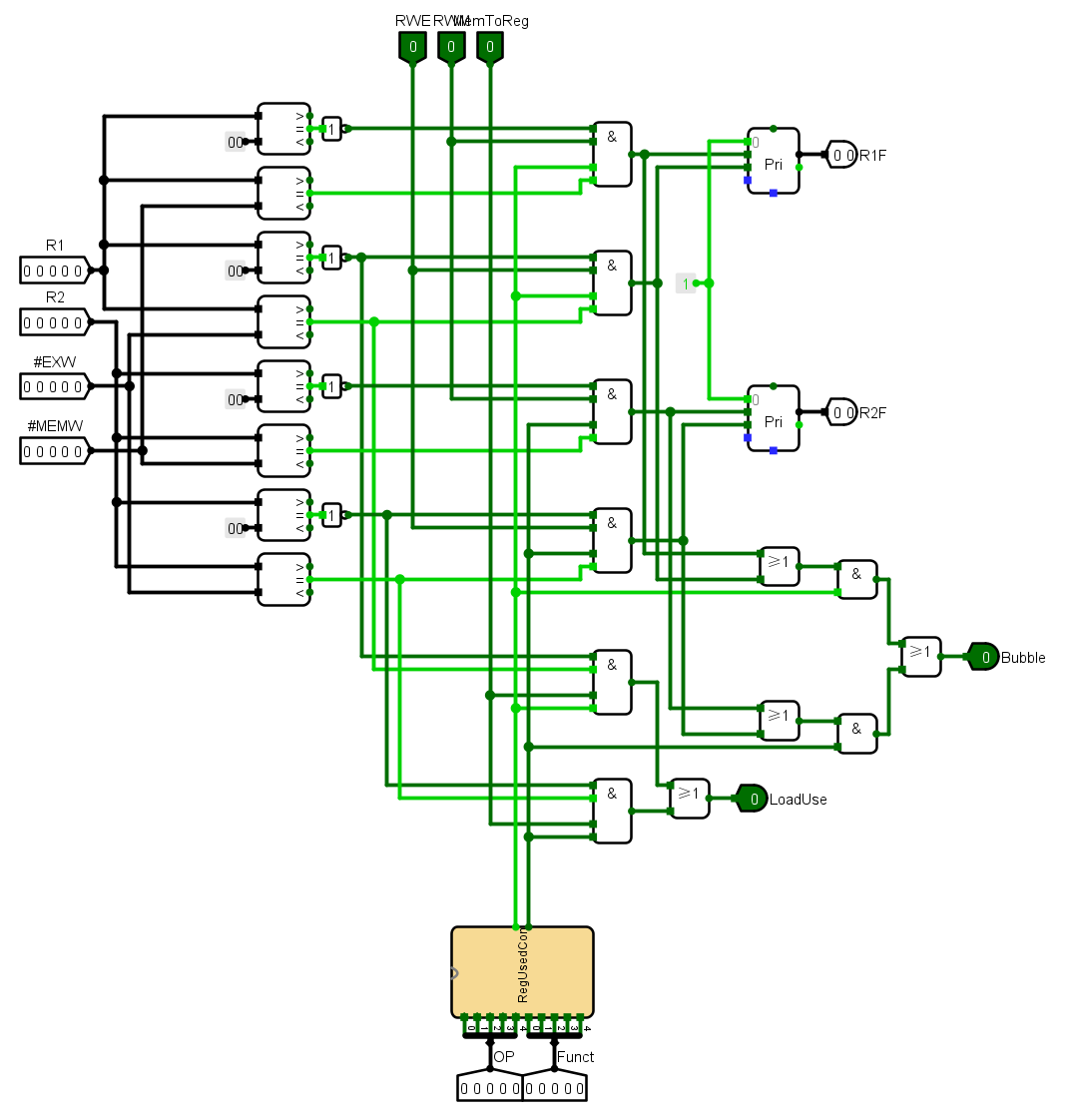


图3.19 Redirect子电路

EX端数据重定向部分的设计如图3.20所示。

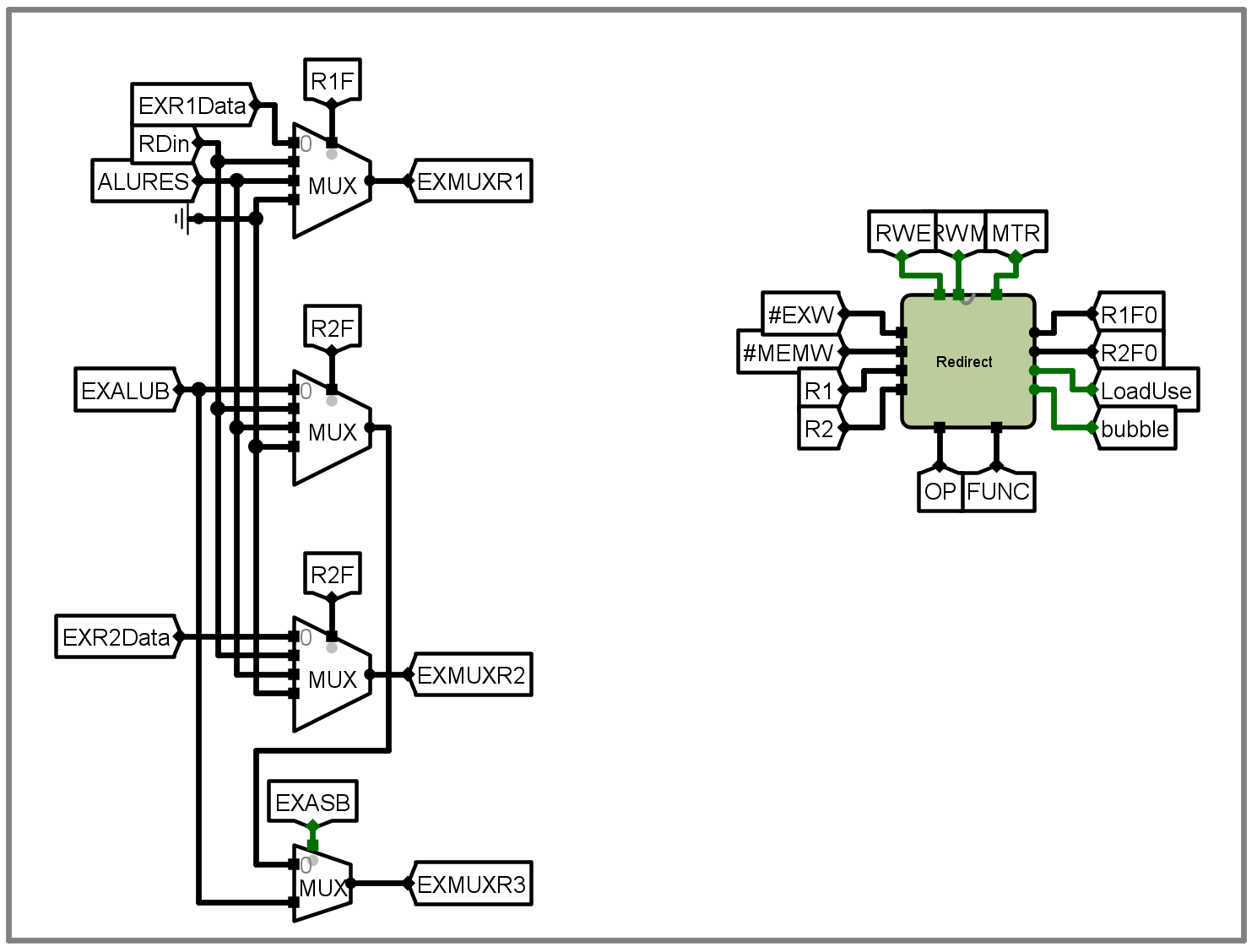


图3.20 EX端数据重定向部分

从而完成重定向流水线的实现，如图3.21所示。

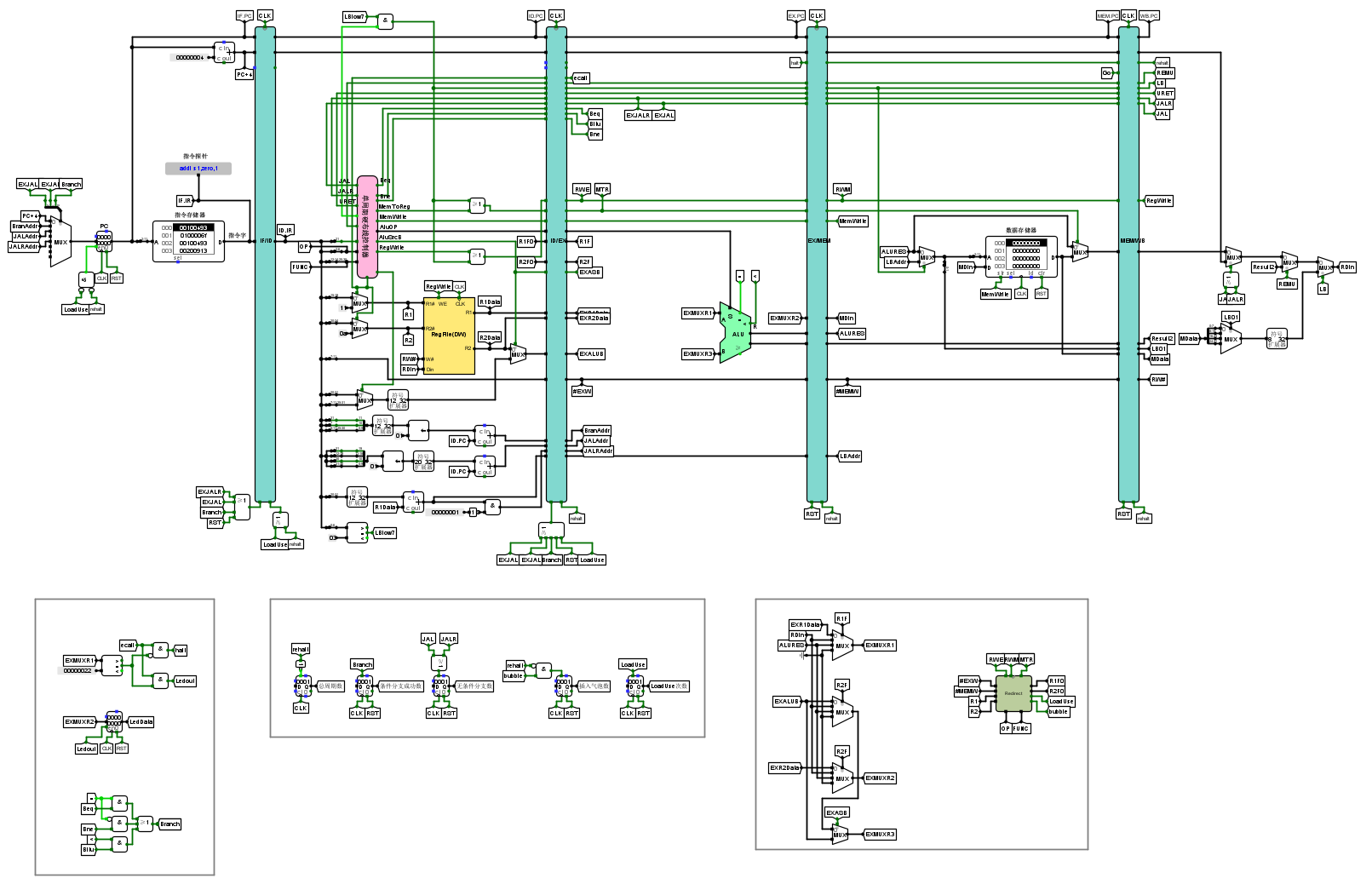


图3.21重定向流水线

## 动态分支预测机制实现

根据2.6部分对于动态分支预测的分析与设计，基于已经完成的重定向流水线进行修改与优化。对于BTB表的设计，参考全相联cache，如图3.22所示。

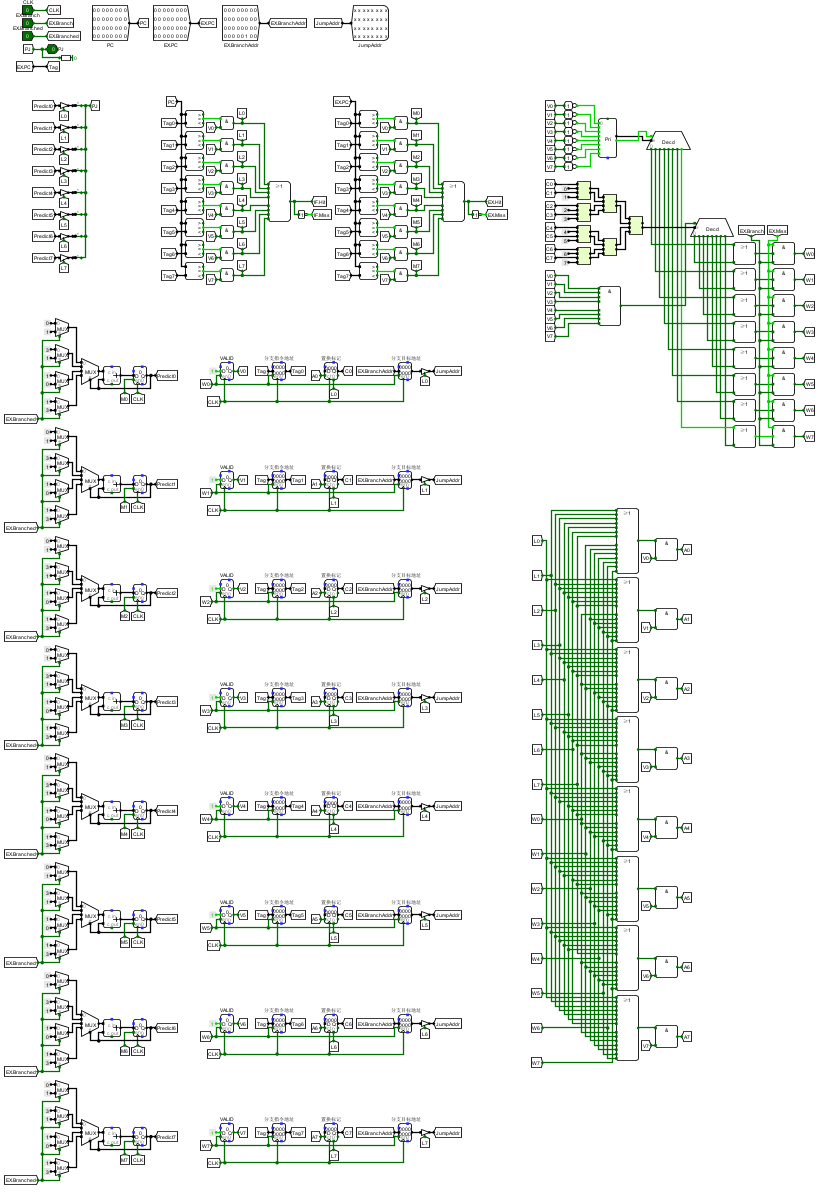


图3.22 BTB表

从而完成动态分支预测的实现，如图3.23所示。

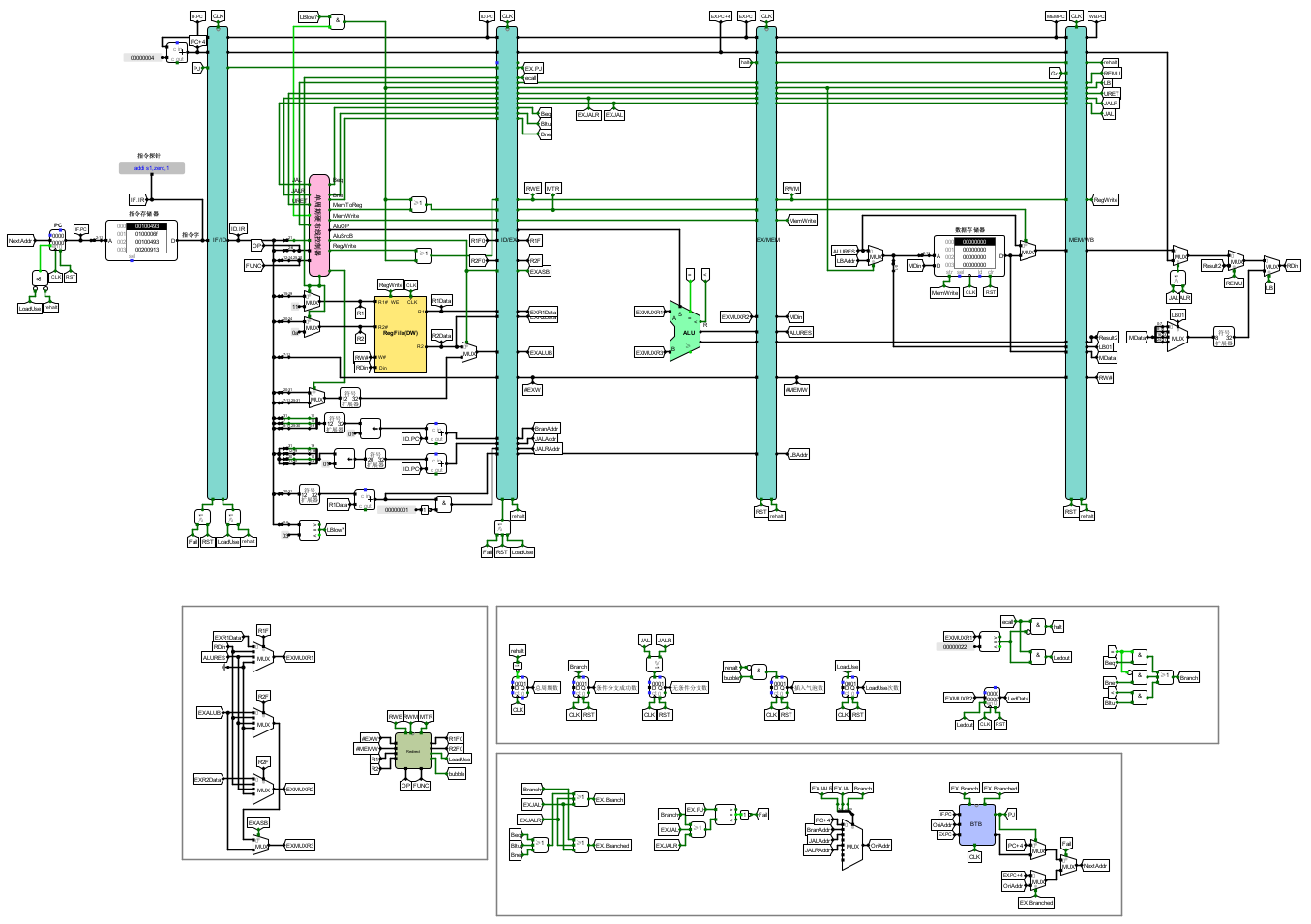


图3.22动态分支预测

# 实验过程与调试

## 测试用例和功能测试

### 单周期CPU测试

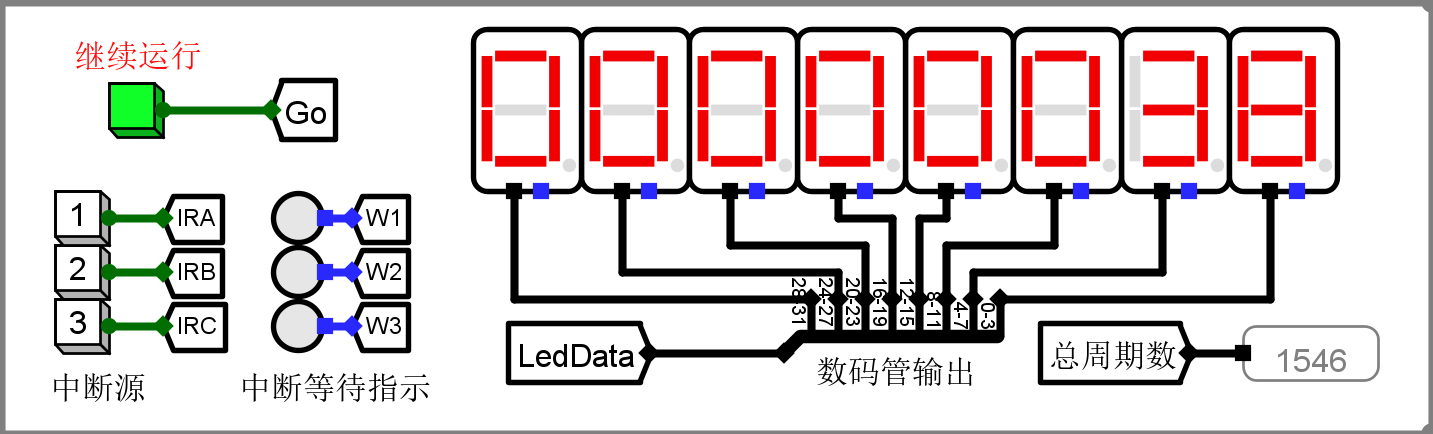


图4.1单周期CPU测试

### 差异化个人CCAB指令测试

根据差异化CCAB指令分配，个人指令为：MUL、REMU、LB和BLTU。

1. MUL指令测试

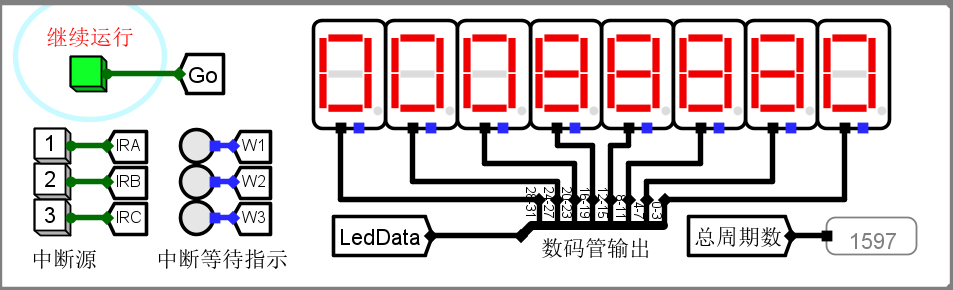


图4.2 MUL指令测试

1. REMU指令测试

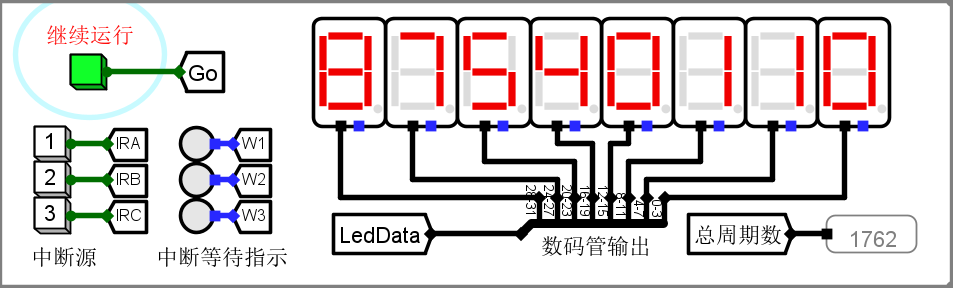


图4.3 REMU指令测试

1. LB指令测试

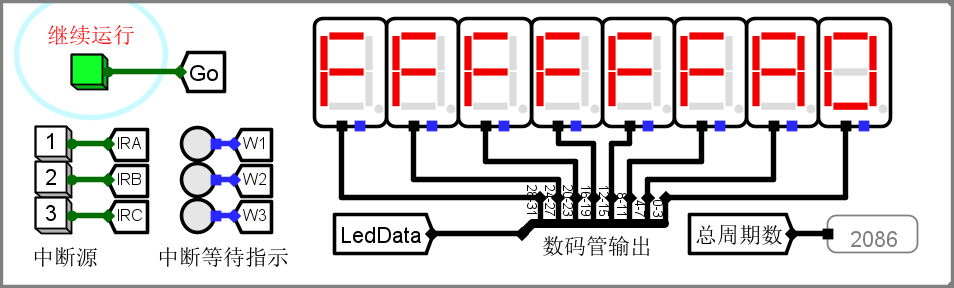


图4.4 LB指令测试

1. BLTU指令测试

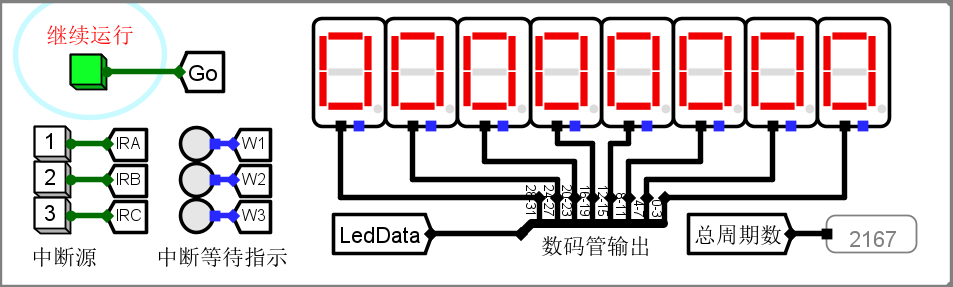


图4.5 BLTU指令测试

### 流水线测试

1. 理想流水线测试

运行完毕后数据存储器如图4.6所示。

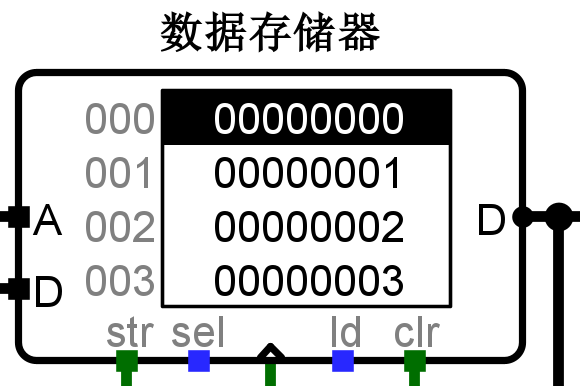


图4.6理想流水线测试

1. 气泡流水线测试

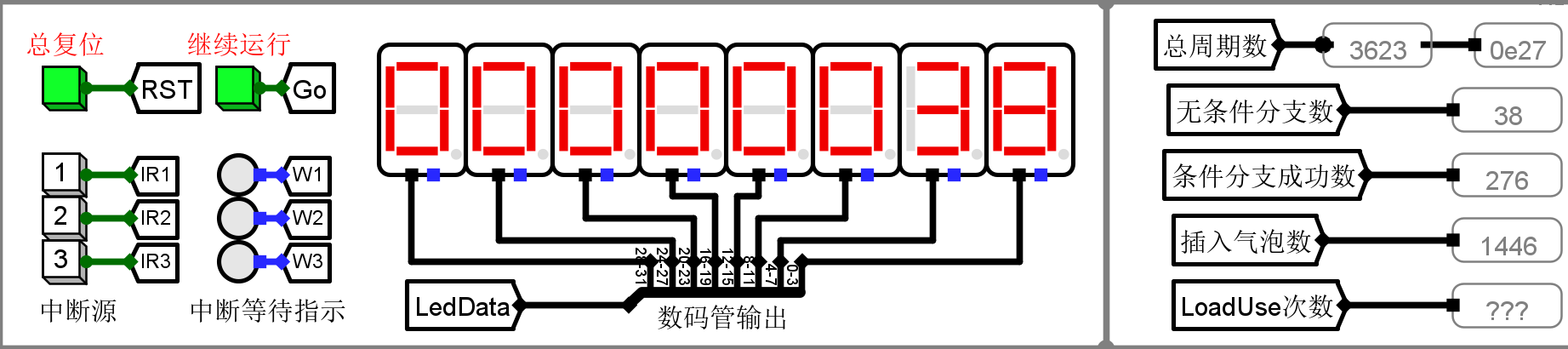


图4.7气泡流水线测试

1. 重定向流水线测试

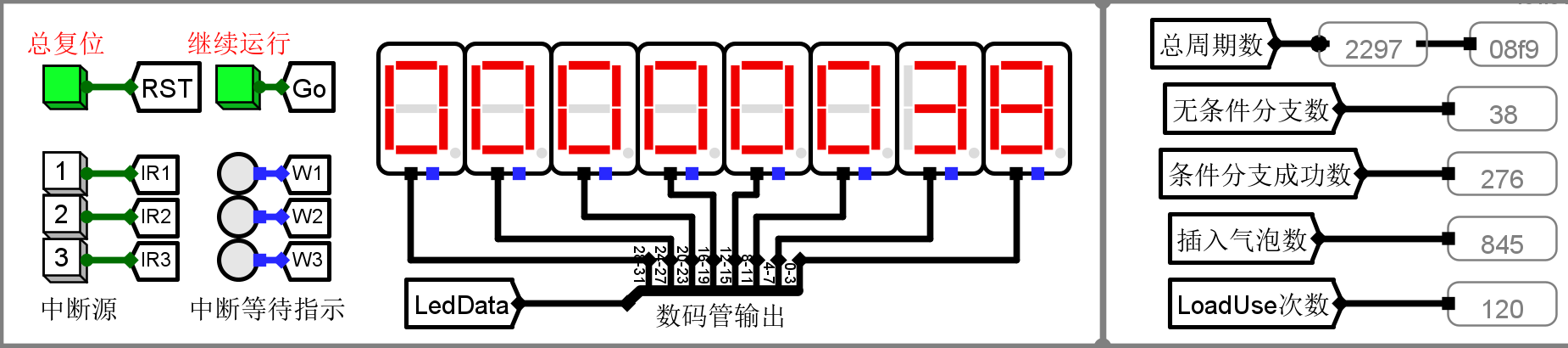


图4.8重定向流水线测试

1. 动态分支预测测试

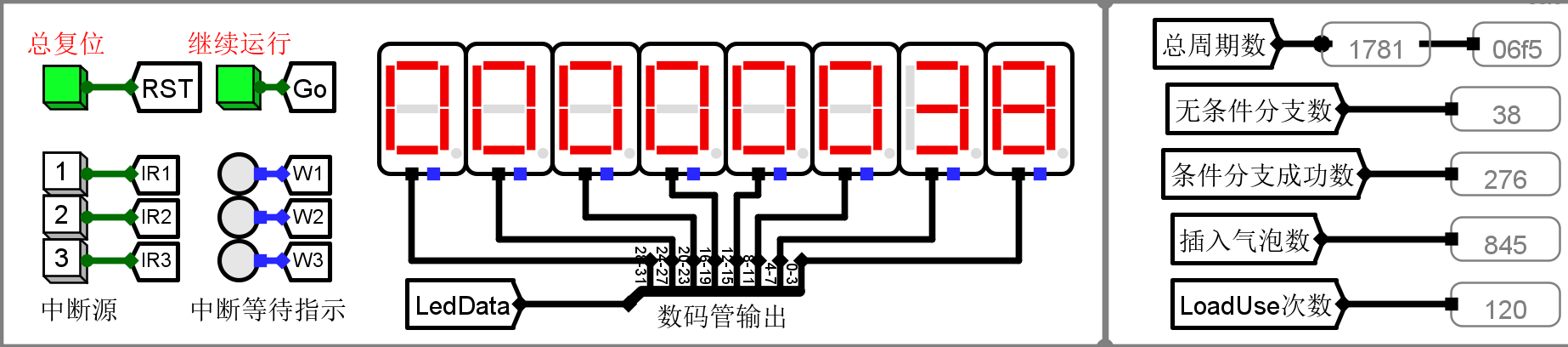
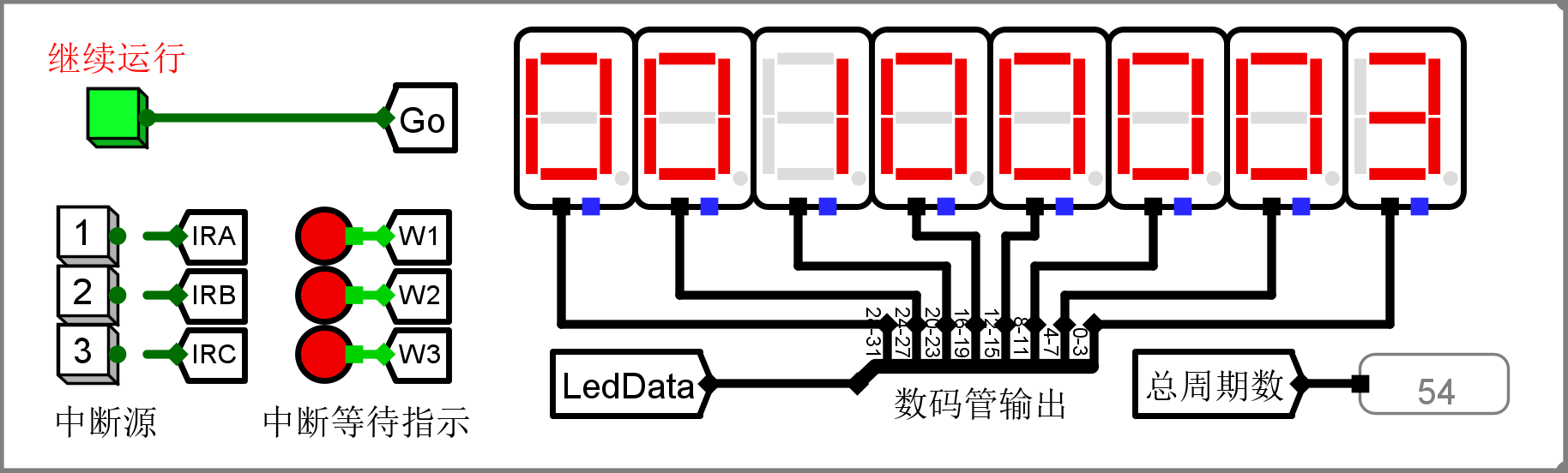
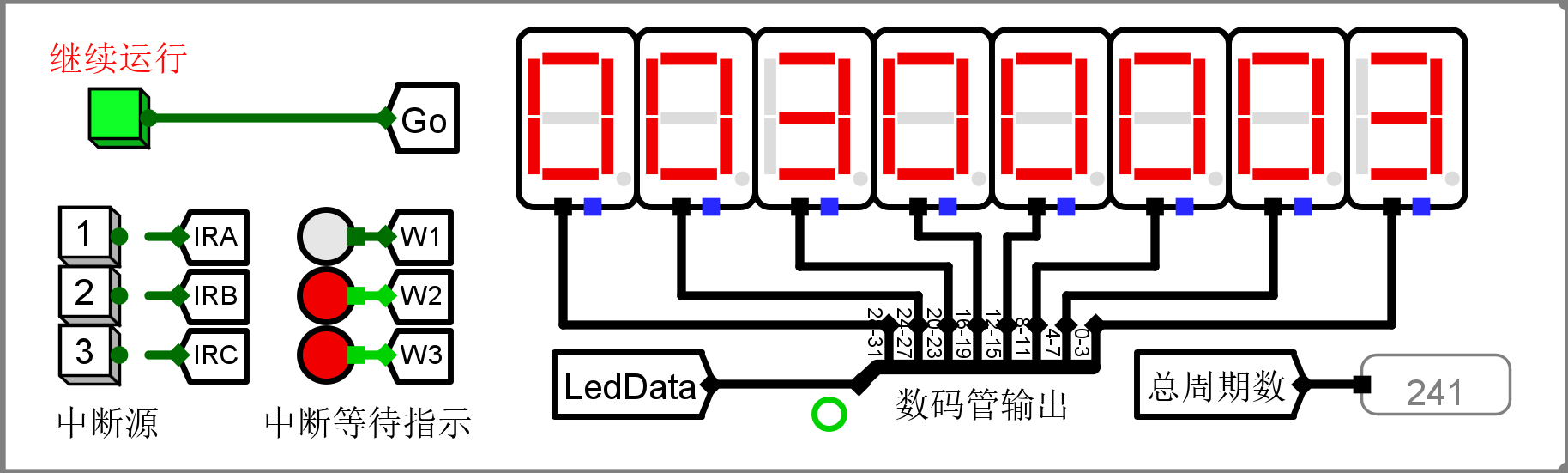


图4.9动态分支预测测试

### 中断测试

1. 单级中断





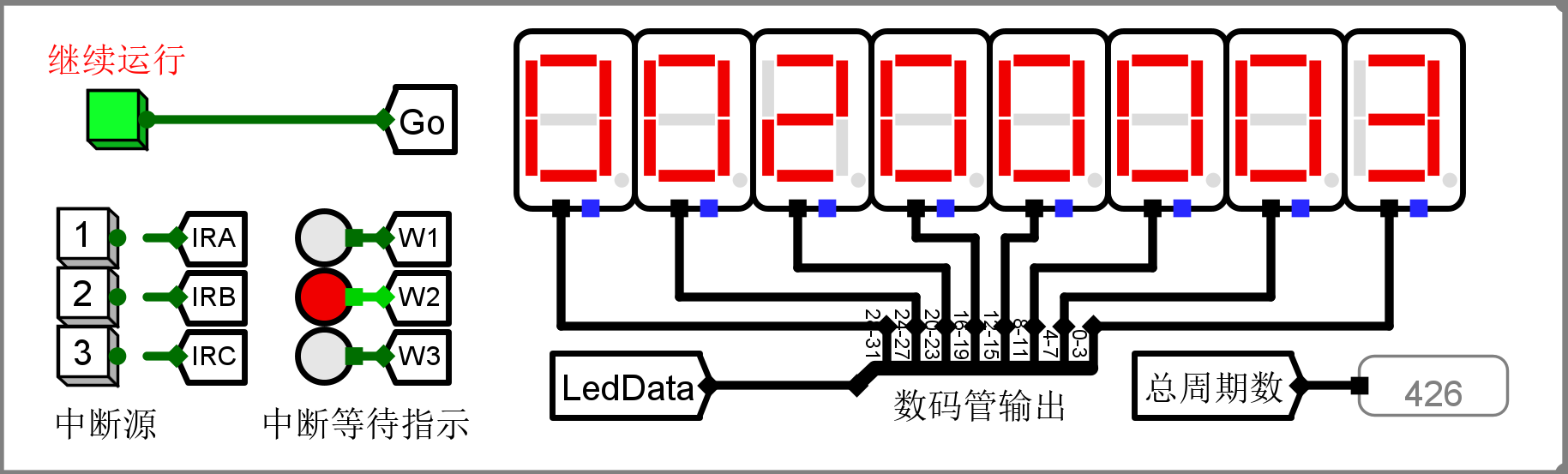
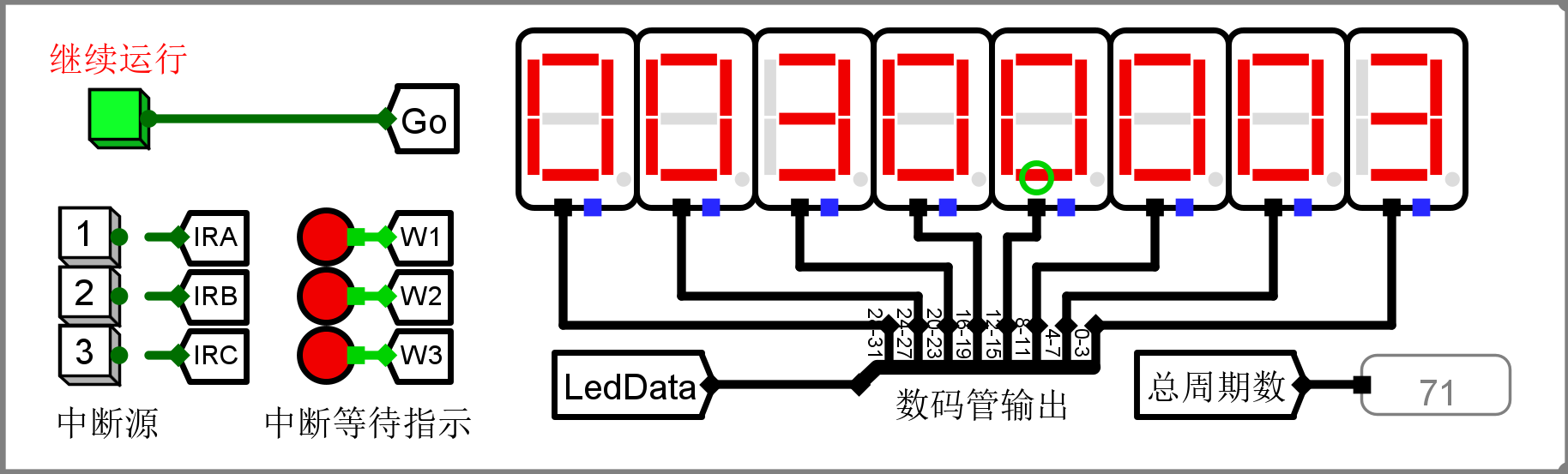
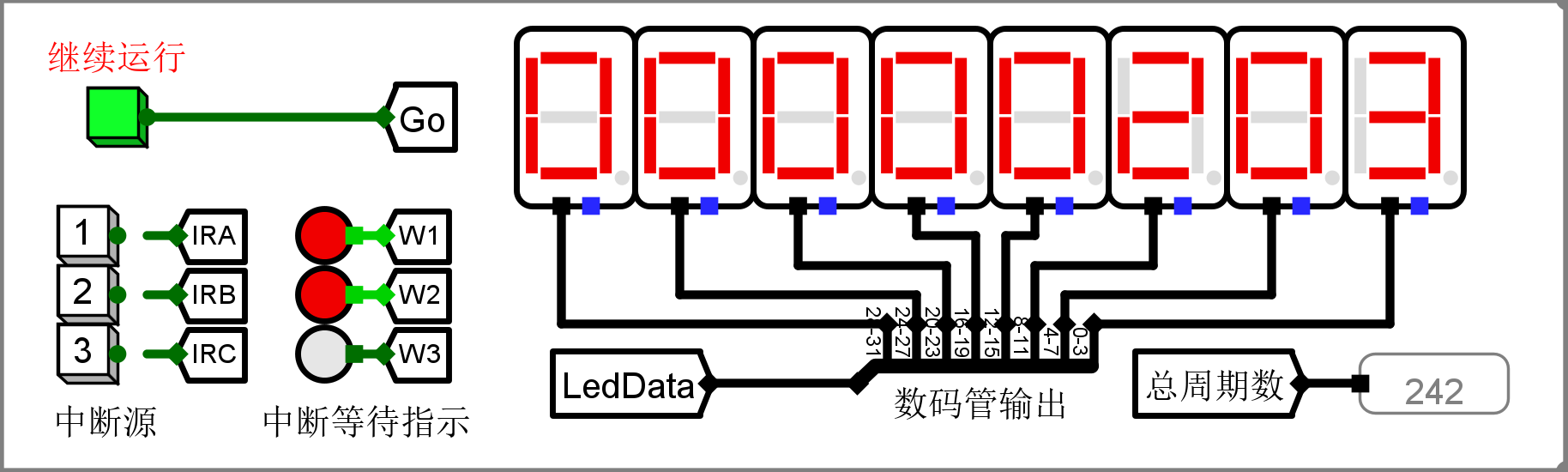
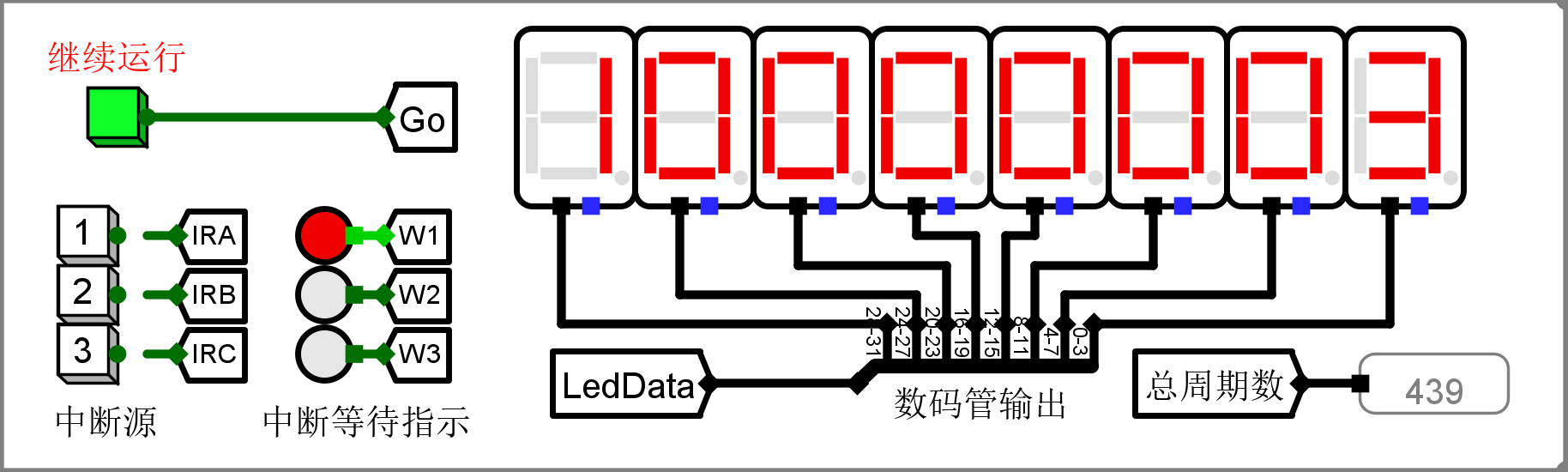


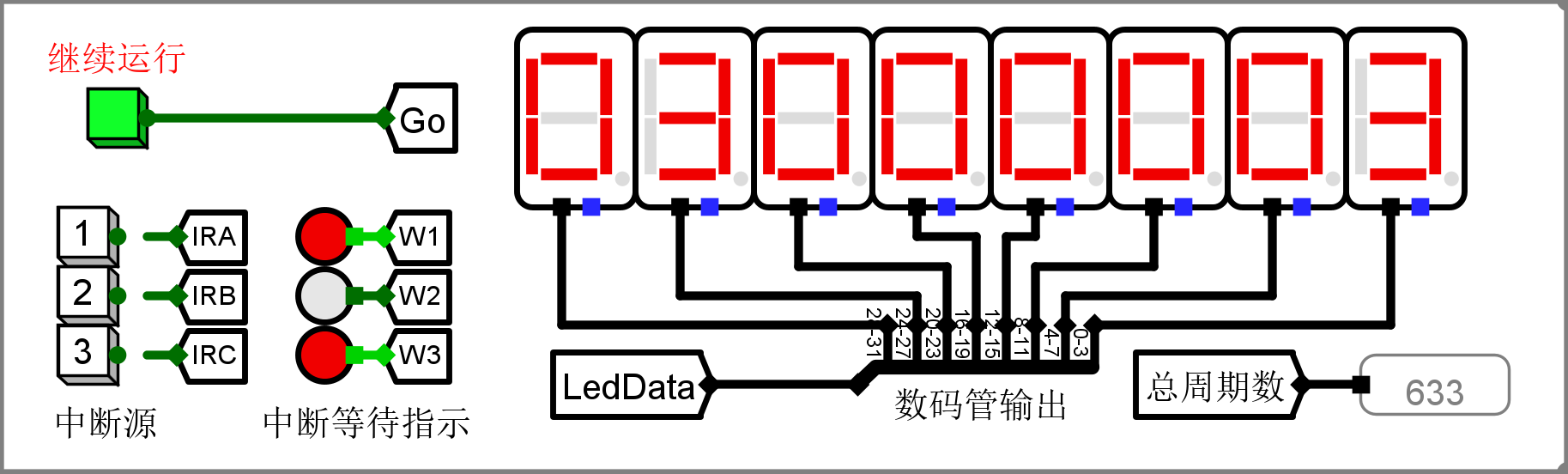
图4.10单级中断测试

1. 多级中断









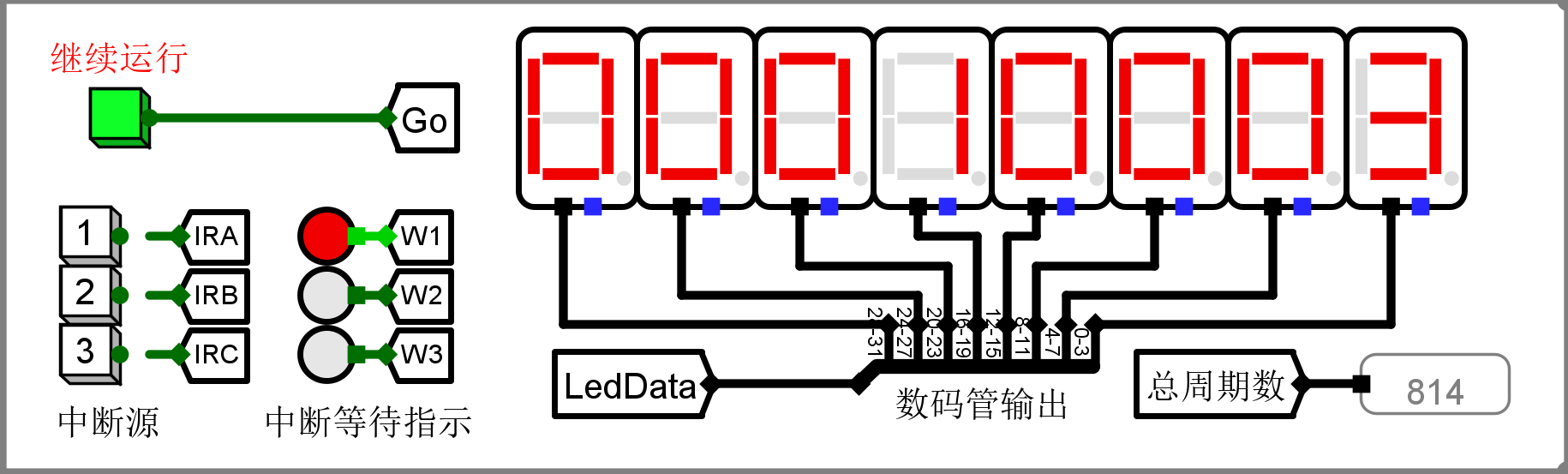
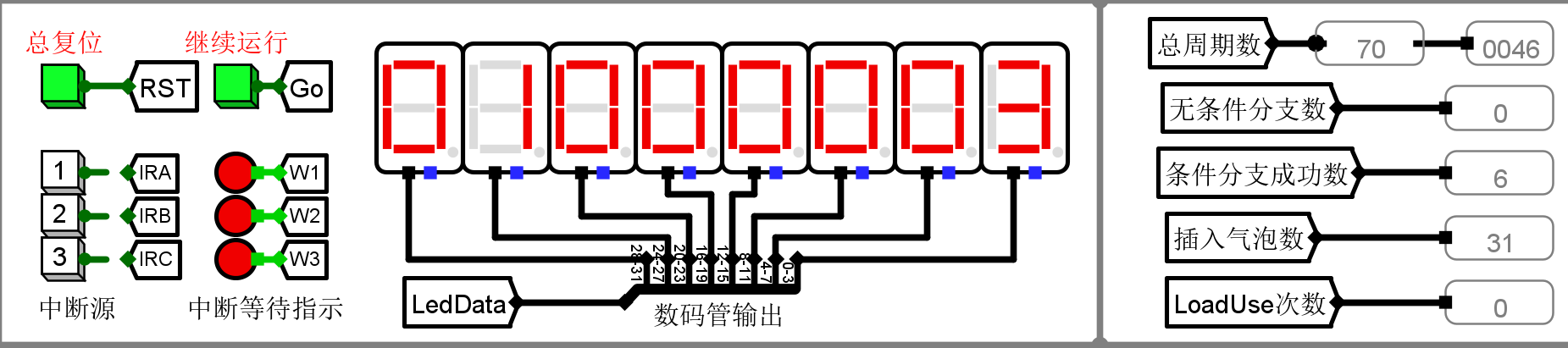
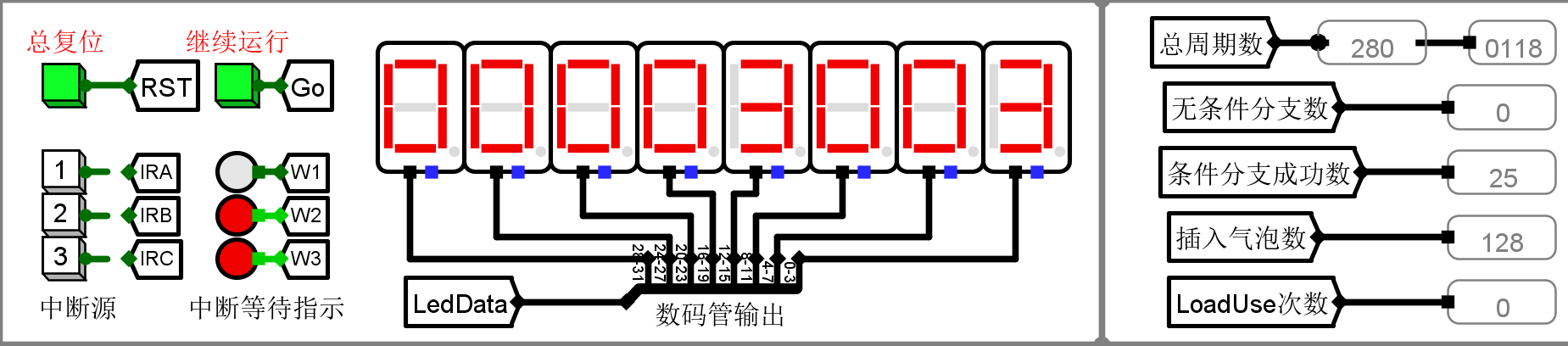


图4.11多级中断测试

1. 流水中断





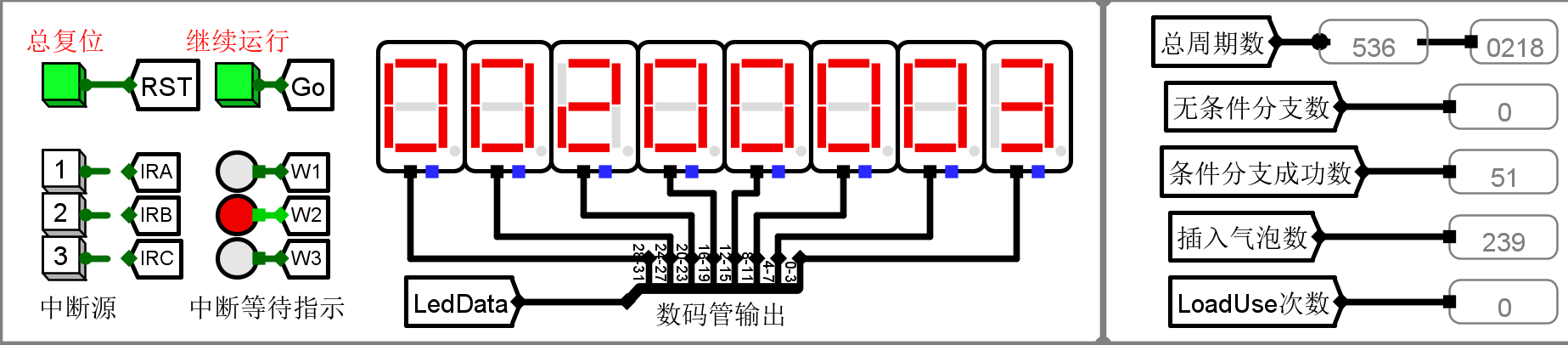


图4.12流水中断测试

## 性能分析

由上述测试结果可看出单周期硬布线CPU、气泡流水线、重定向流水线以及动态分支预测的周期数分别为1546、3623、2297和1781。

1. 单周期硬布线CPU周期数最少，也最为简单，但是由于缺少并发性，单个指令执行时间较长，故效率较低。
2. 气泡流水线周期数最多，虽然实现了流水线并保证了运行过程和结果的正确性，但是因为插入较多气泡导致性能较低。
3. 重定向流水线在气泡流水线的基础上进行了优化，很好地解决了插入气泡过多的问题，大幅减少了周期数，表现较好。
4. 动态分支预测通过使用BTB表进行预测，在重定向流水线的基础上更进一步地进行了优化，周期数仅仅比单周期硬布线CPU多了两百左右，兼顾了性能和正确性。

## 主要故障与调试

### 个人LB指令实现故障

对于个人LB指令的实现先后出现了两次问题，且均花费了较多的时间去寻找问题并进行相关处理，因此这里将这两次问题均进行说明。

1. 单周期CPU：LB指令和全零指令区分问题。

**故障现象：**LB指令无法和全零指令进行区分，无法通过educoder测试。

**原因分析：**在填写单周期硬布线控制器表达式自动生成表时，由于LB指令与表格本身的原因，无法将其与全零指令区分开。在单周期CPU运行时遇到全零指令时则会产生LB信号和RegWrite信号，导致出错。

**解决方案：**查询RISC-V指令表时发现LB指令低7位为0000011，因此可以通过判断这七位来确定是否为LB指令。此外通过表格生成RegWrite信号时将LB一格填为0，在电路中通过或门实现LB的RegWrite操作，如图4.13所示。

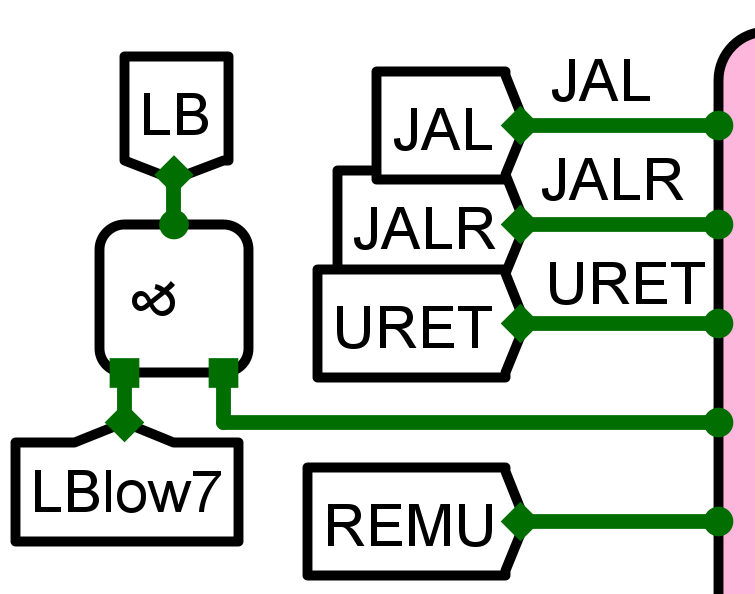
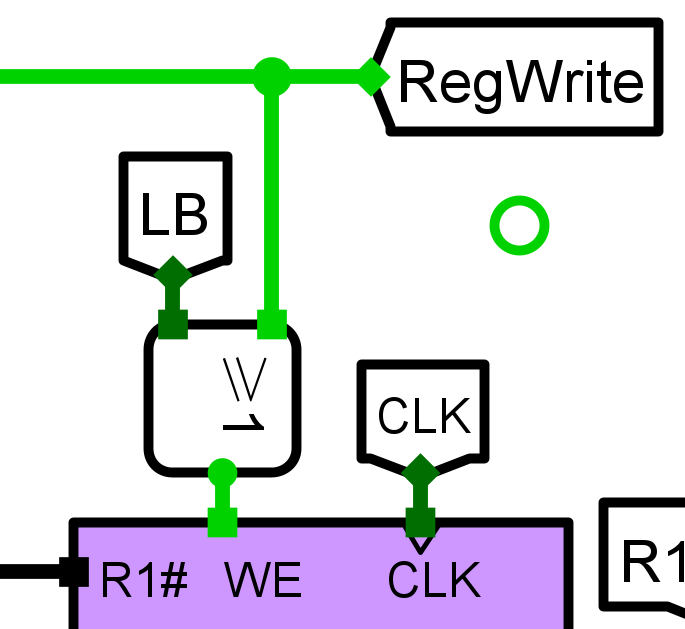
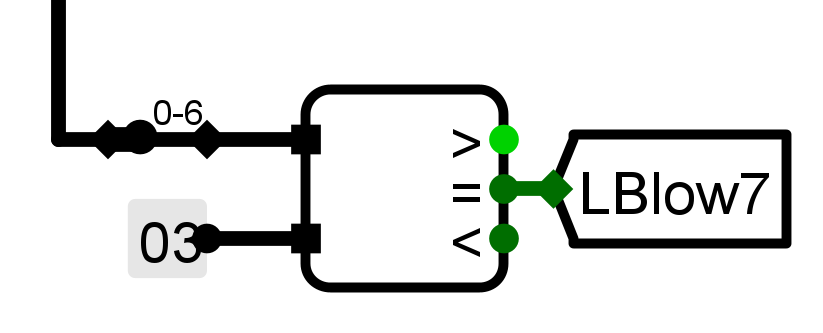
  

图4.13 LB指令与全零指令的区分

1. 重定向流水线：LB指令无法完全按照预期运行。

**故障现象：**重定向流水线运行LB指令时重定向相关信号产生错误，导致运行未按预期进行。

**原因分析：**在单周期硬布线控制器表达式自动生成表中，MemToReg信号是在寄存器写入数据来自存储器时产生，由于LB指令本身的操作与电路的设计，LB指令无需产生该信号。而该信号对于重定向逻辑有着一定程度的影响，而在LB指令运行时按照重定向的逻辑是需要该信号向重定向部件进行输入的。

**解决方案：**将LB信号与MemToReg信号进行或操作后送入流水接口部件，总体4.14所示。。

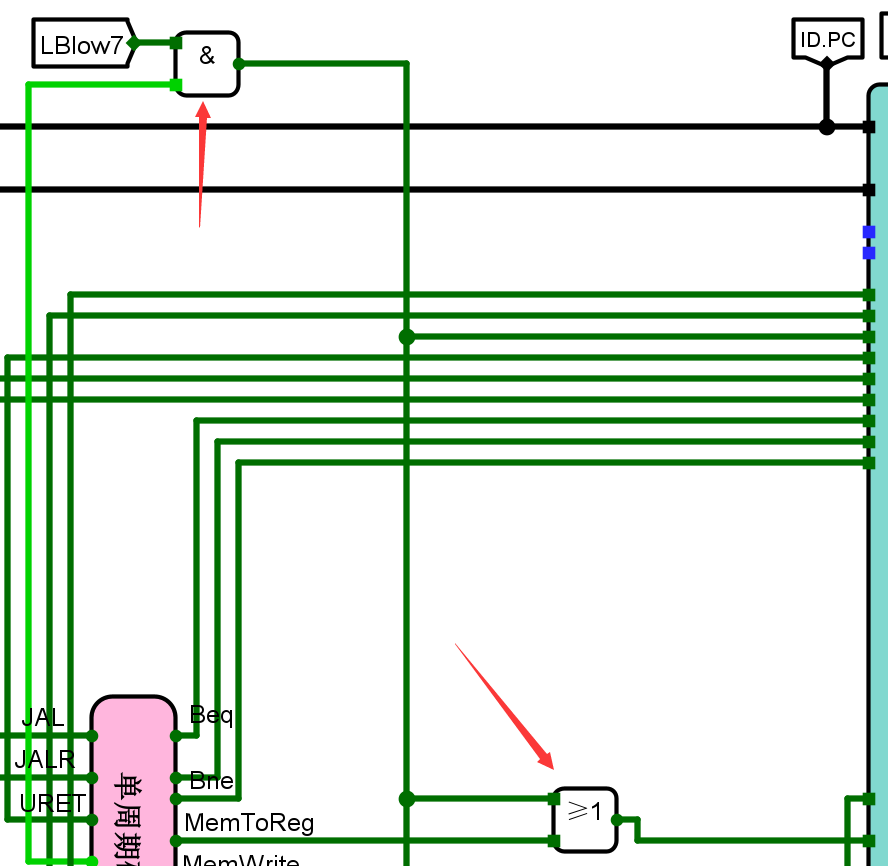


图4.14 LB信号与MemToReg信号或操作

### 流水线隧道处理故障

该故障包括多个流水线信号的隧道的处理的问题，这里使用halt信号进行说明。

**故障现象：**流水线中halt信号的隧道产生和作用位置发生错误，导致流水线在错误的时候因为halt信号而停止。

**原因分析：**流水线对与halt信号的处理是在ID段，读取了寄存器值时即判断是否需要产生halt信号，而各流水线接口部件的使能端也接入了该halt信号进行控制，导致halt信号的作用提取产生，也即提前停止了。

**解决方案：**给ID段后各流水线接口部件增加halt信号接口，将halt信号进行传递，传递至WB段，作为rehalt信号对各流水线接口部件的使能端进行控制，如图4.15所示。

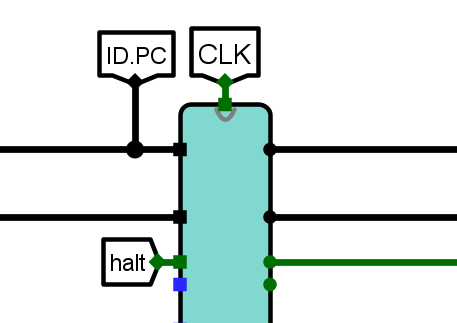
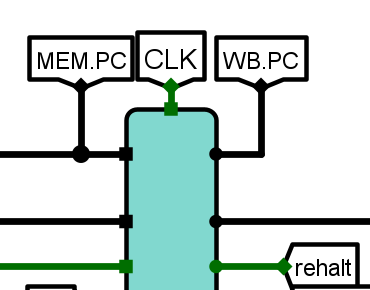
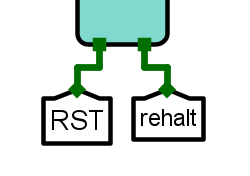
  

图4.15 halt信号隧道的处理

另外由于重定向流水线的重定向机制，halt信号应该使用重定向后的寄存器数据进行产生，因此在重定向流水线以及动态分支预测中将halt信号的产生改为了EX段。

halt信号隧道的处理问题只是一个举例，对于其它信号位置的处理也有一些对应的问题，各种程度上影响了流水线的运行，均按照其各自的作用位置进行了修改。

## 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RISC-V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。根据差异化任务分配，了解所分配的CCAB指令功能，并实现了CCAB指令功能并完成了测试。阅读流水线相关文件。 |
| 第三天 | 了解理想流水线的要求，在Logisim上完成理想流水线并通过educoder测试。 |
| 第四天 | 阅读气泡流水线相关文件，了解相关要求，设计了相关子电路，在Logisim上完成了气泡流水线的实现并通过educoder测试。 |
| 第五天 | 阅读重定向流水线相关文件，在气泡流水线的基础上，设计重定向相关子电路，在Logisim上完成了重定向流水线的实现并通过educoder测试。阅读动态分支预测相关文件，初步规划动态分支预测的实现。 |
| 第六天 | 复习全相联cache的结构，设计BTB子电路。 |
| 第七天 | 结合所设计的BTB子电路，设计动态分支预测机制，并进行调试。 |
| 第八天 | 对动态分支预测子电路进行调试和修改，成功实现动态分支预测机制。阅读了中断相关文件，在单周期CPU电路基础上设计了单级中断机制并进行调试。 |
| 第九天 | 对单级中断电路进行调试和修改，成功完成单级中断机制并通过educoder测试。在单级中断的基础上，设计多级中断的机制并进行调试，成功完成多级中断机制并通过educoder测试。。 |
| 第十天 | 在重定向流水线电路的基础上，结合单级中断机制，设计并调试流水线中断电路，完成流水线中断电路的实现。 |
| 第十一天 | 对所设计的各个电路进行最后的调试与修改，并进行整合。 |

# 设计总结与心得

## 课设总结

本次课程设计为期两个星期，根据任务书要求，作了如下几点工作：

1. 完成了单周期CPU、理想流水线、气泡流水线、重定向流水线、动态分支预测、单级中断、多级嵌套中断以及流水中断的实现。
2. 实现了24条通用指令和4条差异化个人CCAB指令在各个电路上的运行。完成了流水线的划分与连接，实现了指令并行执行的机制。完成了气泡机制、重定向机制以及动态分支预测机制在流水线上的实现，解决了数据冲突问题，不同程度上调高了流水线性能。
3. 完成了单级中断、多级嵌套中断在单周期CPU上的实现以及单级流水中断在重定向流水线上的实现。
4. 完成了对于课设各个部分的调试与修改，通过了educoder的线上测试与线下测试。

## 课设心得

本次课设按照安排为期两周，但在暑假中期就以及公布了课设相关的文件，以便我们提前开始熟悉并上手进行设计与实现，总体来说时间还是很宽裕的。课设接着大二下学期的计算机组成原理课程之后进行，对应的课程实验的基础和经验对课设的进行有了不小的帮助，课设的任务书与相关文件诸如流水线的讲解文件都十分详细，老师也在整个课设进行阶段很有耐心地在课程群内分析解答我们的问题与困惑，因此整体来说本次课设的进行大体上还是比较顺利的。

俗话说万事开头难，最开始的单周期CPU的设计也确实花费了较多精力，在开始设计之前需要对文件包内各个文件进行了解，尤其是任务书需要仔细阅读。在设计时也因为单周期CPU是后面所有电路实现的基础，而在设计每一个数据通路的时候都在考虑是否会给后续改动带来问题、或者是否会给后续电路的实现带来便利，再加上前前后后诸多小问题，尤其是4.3.1部分中所提到的LB指令的问题，单周期CPU的设计用了两天左右的时间。

但是有了这一基础的电路，后续所有的电路只需考虑如何加上各自的机制即可，流水线和中断都是如此。理想流水线由于是对单周期CPU的划分和组合，因此实现得较为顺利。但在后续更为复杂的流水线的设计中，正如4.3.2部分中所提到的，在流水线接口部件对于数据与信号的保存的基础上，这些数据与信号的使用便成为了一个重要问题，流水线各个阶段的不同指令对于不同数据和信号的使用与影响均需要纳入考虑范围，尤其是气泡机制、重定向机制和动态分支预测机制均受到对应阶段的影响，因此需要慎重考虑每一个信号应该在哪个阶段发挥作用，每一个数据应该在哪个阶段被使用，以及每一个隧道应该连接哪些部件。

而对于气泡机制、重定向机制和动态分支预测机制的实现，由于所提供的流水线讲解的文件给出了细致的描述，均实现得较为顺利，只是在动态分支预测的BTB表实现中，由于对于全相联cache的具体实现有所遗忘，需要查阅复习上个学期的组原知识，这也进一步加深了我对cache机制的印象与理解。

在中断机制的实现中，由于之前对于中断的理解与认识均停留在理论阶段，因此使用硬件来实现中断还是花费了较多精力的，尤其是多级中断，其设计与实现更为复杂，好在通过仔细阅读任务书并查阅相关资料以及向同学老师咨询之后成功完成了对于中断的实现，也让我更深入地理解了中断的硬件实现而不再是停留在理论。

整体上来说这次课设较为成功，而且是目前所做过的课设中最为偏向硬件的一次，给我带来了很深的印象，同学们和老师们的帮助也让我十分感激。

至于对于课设的建议，就我个人而言，本次课设的时间十分充裕，从任务的公布到检查再到报告的提交，前前后后总共给了将近两三个月，因此可以不用因为时间不足而着急。老师们也在课程群中有耐心地为我们答疑，解决了我们很多的问题。

而对于课设指引而言，我认为所给的参考资料虽然多而详细，但却因为太多而有些让人难以上手，产生畏难情绪，因此我个人觉得可以在资料包中添加一个对于各个文件的内容和作用给出解释的readme文件或者txt文件，方便学生们了解所给的各个文件，以便快速上手。

# 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.

|  |
| --- |
| 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: 汪宇飞** |