***2020***



**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： | CS1704 |
| 学 号： | U201714641 |
| 姓 名： | 余天越 |
| 电 话： | 13826490126 |
| 邮 件： | 1418583869@qq.com |

目 录

[1 课程设计概述 3](#_Toc48472365)

[1.1 课设目的 3](#_Toc48472366)

[1.2 设计任务 3](#_Toc48472367)

[1.3 设计要求 3](#_Toc48472368)

[1.4 技术指标 4](#_Toc48472369)

[2 总体方案设计 6](#_Toc48472370)

[2.1 单周期CPU设计 6](#_Toc48472371)

[2.2 中断机制设计 11](#_Toc48472372)

[2.3 流水线设计 13](#_Toc48472373)

[2.4 气泡式流水线设计 14](#_Toc48472374)

[2.5 重定向流水线设计 14](#_Toc48472375)

[3 详细设计与实现 15](#_Toc48472376)

[3.1 单周期CPU 实现 15](#_Toc48472377)

[3.2 中断机制实现 19](#_Toc48472378)

[3.3 流水CPU实现 22](#_Toc48472379)

[3.4 气泡式流水线实现 23](#_Toc48472380)

[3.5 数据重定向流水线实现 24](#_Toc48472381)

[4 实验过程与调试 26](#_Toc48472382)

[4.1 测试用例和功能测试 26](#_Toc48472383)

[4.2 性能分析 31](#_Toc48472384)

[4.3 主要故障与调试 31](#_Toc48472385)

[4.4 实验进度 33](#_Toc48472386)

[5 设计总结与心得 34](#_Toc48472387)

[5.1 课设总结 34](#_Toc48472388)

[5.2 课设心得 34](#_Toc48472389)

[参考文献 36](#_Toc48472390)

# 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27条基本32位MIPS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 |
| 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 $v0==10 halt(停机指令)  else 数码管显示$a0值 |
| 24 | SYSCALL | 系统调用 |
| 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 |  |
| 29 | BLEZ | 小于或等于0跳转 |  |
| 30 | LUI | 把一个16位的立即数填入到寄存器的高16位，低16位补零 | rt 🡨 imm\*65536 |
| 31 | LH | 从存储器中读取半个字的数据到寄存器中 |  |

# 总体方案设计

## 单周期CPU设计

单周期CPU采用的是硬布线的方法，可以将CPU分为若干模块：指令寄存器、指令存储器、译码逻辑、硬布线控制器、寄存器文件、运算器、内存、地址转移逻辑等，在一个时钟周期内，五段功能模块各司其职，分别为：取指令IF、指令译码ID、指令执行EXE、读写存储器MEM和写回寄存器WB。在 24 单周期 CPU 的基础上添加 4 条扩展指令分别是:SUBU、LUI、LH、BLEZ。

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

#### 程序计数器PC

程序计数器是下一条指令在指令存储器中的地址，用寄存器存储，决定着整个程序的走向。PC的值可是正常的自增4，表示顺序执行，也可以是任意地址，表示跳转语句，多个可能的地址并行计算得到，最终通过使能信号控制的多路选择器决定最终的值。

#### 指令存储器IM

指令存储在ROM中，数据位宽为32位，以字寻址，取PC的第2到11位输入，将最终取出来的地址送到译码逻辑模块进行译码。

#### 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 |
| --- | --- | --- | --- |
| A | 输入 | 32 | 操作数A |
| B | 输入 | 32 | 操作数B |
| AluOP | 输入 | 4 | 运算器功能码，决定运算方式 |
| shamt | 输入 | 32 | 移位数目 |
| Result | 输出 | 32 | ALU计算结果 |
| Equal | 输出 | 1 | 两操作数是否相等 |

#### 寄存器堆RF

MIPS寄存器有32个，位宽为32，每个时钟周期可根据寄存器编号同时读两个寄存器的内容，并且修改一个寄存器，写使能控制写数据，输入输出的引脚见表2.2。

表 2.2寄存器堆引脚与功能描述

| 引脚 | 输入/输出 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| CLK | 输入 | 1 | 时钟信号 |
| R1# | 输入 | 5 | 读寄存器1的编号 |
| R2# | 输入 | 5 | 读寄存器2的编号 |
| W# | 输入 | 5 | 写寄存器的编号 |
| WE | 输入 | 1 | 写使能信号 |
| Din | 输入 | 32 | 输入数据 |
| A | 输出 | 32 | 寄存器1的数据 |
| B | 输出 | 32 | 寄存器2的数据 |

#### 数据存储器

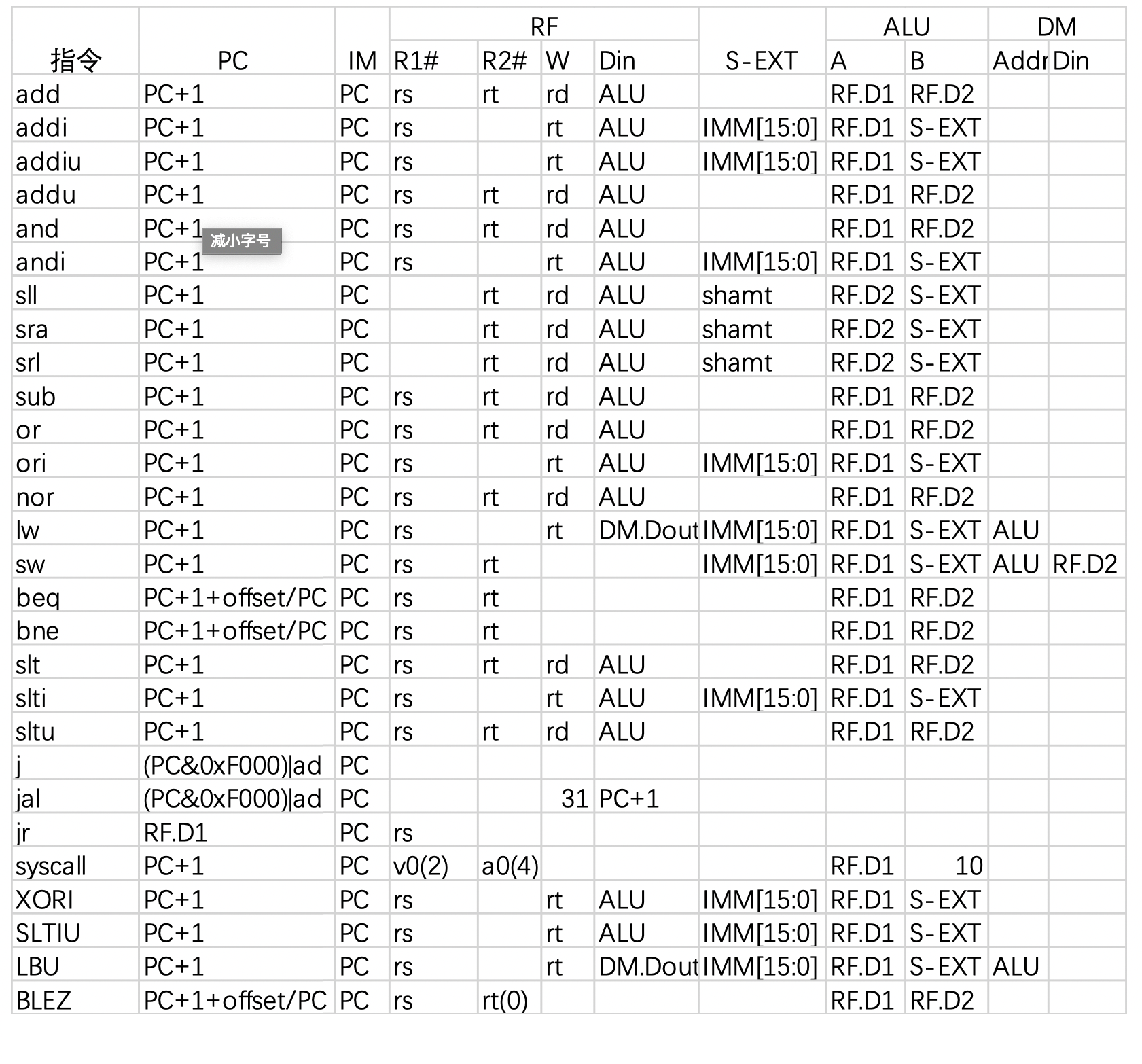
存储器用RAM实现，sel决定的访问模式在组成原理实验中有过实现，读写地址由其中2到11位决定，其输入输出引脚见表2.3。

表 2.3寄存器堆引脚与功能描述

| 引脚 | 输入/输出 | 位宽 | 功能描述 |
| --- | --- | --- | --- |
| CLK | 输入 | 1 | 时钟信号 |
| RST | 输入 | 1 | 清零信号 |
| sel | 输入 | 2 | 决定字节、半字或字访问 |
| str | 输入 | 1 | 写使能信号 |
| Din | 输入 | 32 | 输入数据 |
| Addr | 输入 | 10 | 读写地址 |
| Dout | 输出 | 32 | 读出的数据 |

### 数据通路的设计

表2.4 指令系统数据通路框架



一些文字和图案

描述已自动生成

### 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2.5。

表 2.5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 |
| MemToReg | 0 | 写数据来源为ALU结果 |
| 1 | 写数据来源为Mem |
| MemWrite | 0 | Mem读 |
| 1 | Mem写 |
| AluSrcB | 0 | ALU输入端B取值为R2 |
| 1 | ALU输入端B取值为imm |
| RegWrite | 0 | 寄存器组读 |
| 1 | 寄存器组写 |
| Syscall | 0/1 | 是否系统调用 |
| SignedExt | 0 | imm使用0扩展 |
| 1 | imm使用符号扩展 |
| RegDst | 0 | 写寄存器编号为rt |
| 1 | 写寄存器编号为rd |
| BEQ | 0/1 | 指令是否为BEQ |
| BNE | 0/1 | 指令是否为BNE |
| JR | 0/1 | 指令是否为JR |
| JMP | 0/1 | 指令是否为JMP |
| JAL | 0/1 | 指令是否为JAL |
| MFC0 | 0/1 | 指令是否为MFC0 |
| MTC0 | 0/1 | 指令是否为MTC0 |
| ERET | 0/1 | 指令是否为ERET |
| LUI | 0/1 | 指令是否为LUI |
| BLEZ | 0/1 | 指令是否为BLEZ |
| LH | 0/1 | 指令是否为LH |

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2.6所示。

表 2.6主控制器控制信号框架

| 指令 | ALU\_OP | MemToReg | MemWrite | ALU\_SRC | RegWrite | SignedExt | RegDst | JMP |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|
| SLL | 0 |  |  |  | 1 |  | 1 |  |
| SRA | 1 |  |  |  | 1 |  | 1 |  |
| SRL | 2 |  |  |  | 1 |  | 1 |  |
| ADD | 5 |  |  |  | 1 |  | 1 |  |
| ADDU | 5 |  |  |  | 1 |  | 1 |  |
| SUB | 6 |  |  |  | 1 |  | 1 |  |
| AND | 7 |  |  |  | 1 |  | 1 |  |
| OR | 8 |  |  |  | 1 |  | 1 |  |
| NOR | 10 |  |  |  | 1 |  | 1 |  |
| SLT | 11 |  |  |  | 1 |  | 1 |  |
| SLTU | 12 |  |  |  | 1 |  | 1 |  |
| JR |  |  |  |  |  |  |  | 1 |
| SYSCALL |  |  |  |  |  |  |  |  |
| J |  |  |  |  |  |  |  | 1 |
| JAL |  |  |  |  | 1 |  |  | 1 |
| BEQ |  |  |  |  |  | 1 |  |  |
| BNE |  |  |  |  |  | 1 |  |  |
| ADDI | 5 |  |  | 1 | 1 | 1 |  |  |
| ANDI | 7 |  |  | 1 | 1 |  |  |  |
| ADDIU | 5 |  |  | 1 | 1 | 1 |  |  |
| SLTI | 11 |  |  | 1 | 1 | 1 |  |  |
| ORI | 8 |  |  | 1 | 1 |  |  |  |
| LW | 5 | 1 |  | 1 | 1 | 1 |  |  |
| SW | 6 |  | 1 | 1 |  | 1 |  |  |
| SUBU | 5 |  |  |  | 1 |  | 1 |  |
| LUI |  |  |  |  | 1 |  |  |  |
| LH |  | 1 |  |  | 1 | 1 |  |  |
| BLEZ |  |  |  |  |  |  |  |  |

## 中断机制设计

### 总体设计

中断分为单级中断和多级中断，其中单级中断无需考虑被打断的情况，在遭遇中断时现将断点存入EPC寄存器，再在指令层面保护现场，根据中断仲裁逻辑执行优先度最高的中断服务子程序，假如在执行的过程中遇到其他中断，仍继续执行至结束，之后恢复现场，最后将EPC中存储的地址信息传递给PC，以恢复中断前的指令执行；多级中断在保护现场后需要开中断以接收其他中断请求，对EPC、IE和中断屏蔽寄存器的操作需要增加ERET、MFC0和MTC0指令并更新控制器和数据通路。

中断流程图见图2.2。

截图里有图片

描述已自动生成

图 2.2 中断流程图

### 硬件设计

#### 中断请求电路

用三个按钮来代替中断请求信号的产生，分别为IR1#, IR2#和IR3#。为了方便测试，可以加入信号模拟电路，自发地产生中断信号。

#### 中断仲裁电路

由于三个中断信号可能同时出现，所以需要选择其中优先度最高的信号，使用优先编码器对中断请求信号进行编码。

#### 中断使能寄存器

用D触发器实现，输出1时允许中断，为0时屏蔽所有中断。

#### 操作控制器修改

修改控制器的逻辑，使其能识别eret, mfc0和mtc0指令，mfc0和mtc0指令需要读写通用寄存器和IE, EPC寄存器，需要将mfc和mtc指令作为寄存器的使能信号。

### 软件设计

软件层面需要与硬件协同，首先是需要使得指令支持开关中断，分别将中断使能信号IE置为1和0，决定CPU能否接收到外部中断信号，而ERET对应中断返回指令，开中断并恢复现场，从EPC中取出保存的指令地址到PC，继续执行之前被打断的指令。现场是可能被随机出现的中断请求破坏的寄存器值，保护现场和恢复现场分别指的是保存寄存器中的值到RAM中，以及从RAM中恢复寄存器的值。

## 流水线设计

### 总体设计

将CPU的数据通路划分为五段流水：取指令IF、指令译码ID、指令执行EXE、读写存储器MEM和写回寄存器WB，弥补CPU在所有指令执行时间相同的时候，执行时间取决于最慢的指令而效率低下的缺点，能够使得各组件的利用率更高，性能更好，指令流水线逻辑框架见图2.3。

图片包含 游戏机, 钟表, 物体

描述已自动生成

图 2.3 指令流水线逻辑框架

### 流水接口部件设计

流水接口部件的作用是将上一段中的数据和控制信号锁存给下一段使用，可以用寄存器组将功能段分开并将结果存储一个时钟周期。所有寄存器共用同一个时钟信号，上升沿触发，使能端收接停机信号，低电平有效；将多路选择器的1端信号接地，正常情况下选择输入信号，清零时选择常数0。

### 理想流水线设计

所谓理想流水线指的是没有任何数据相关和控制冲突指令的流水线，只需要保证数据正确传递即可。对于单周期MIPS CPU的改进，需要把数据通路用寄存器组划分为五个段，值得注意的是，写回段WB为了防止数据冲突，需要将RegFile设置为下降沿写入，保证下一个上升沿到来在译码段ID也能得到正确的值；其次，停机指令产生的信号Halt要放到WB段，否则会因为在执行段EX停机导致MEM和WB段没有执行完整。

## 气泡式流水线设计

由于理想流水线不能解决数据读写冲突，所以需要检测数据相关检测逻辑，输入为ID段和EX段的指令信号，以及ID段和MEM段的指令信号，假设任意一个检测到了数据相关，则暂停ID段并插入气泡解决冲突，直到相关的数据成功地写回寄存器内才能继续执行ID段的指令。假设在EX段遇到了分支指令，意味着IF和ID段的指令是无效的，故将其以气泡替换，解决控制冲突。

## 重定向流水线设计

气泡流水线虽然能解决冲突，但是会带来过多的气泡导致效率的下降，假设可以判断指令的冲突情况将MEM或WB阶段的数据交给EX段，就可以不用产生气泡而占用额外的时钟周期了。在ID段检测数据相关分为load-use相关和其他相关，ID段的某一源寄存器是EX段的目标寄存器，且EX段是一条load指令则产生load-use相关，需要在ID/EX接口插入一个气泡，否则流水线的关键时延会变长。修改控制器产生2位宽的输出信号f1和f2，决定ALU的输入端选择何处的数值。

# 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）

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。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。如图 3.1所示。

图片包含 游戏机, 钟表, 标志

描述已自动生成

图 3.1程序计数器（PC）

1. 指令存储器（IM）

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 3.2所示。

手机屏幕的截图

描述已自动生成

图 3.2指令存储器（IM）

1. 寄存器文件（RegFile）

Logism实现：

由32个32位寄存器组合而成，不赘述，如图3.3所示。

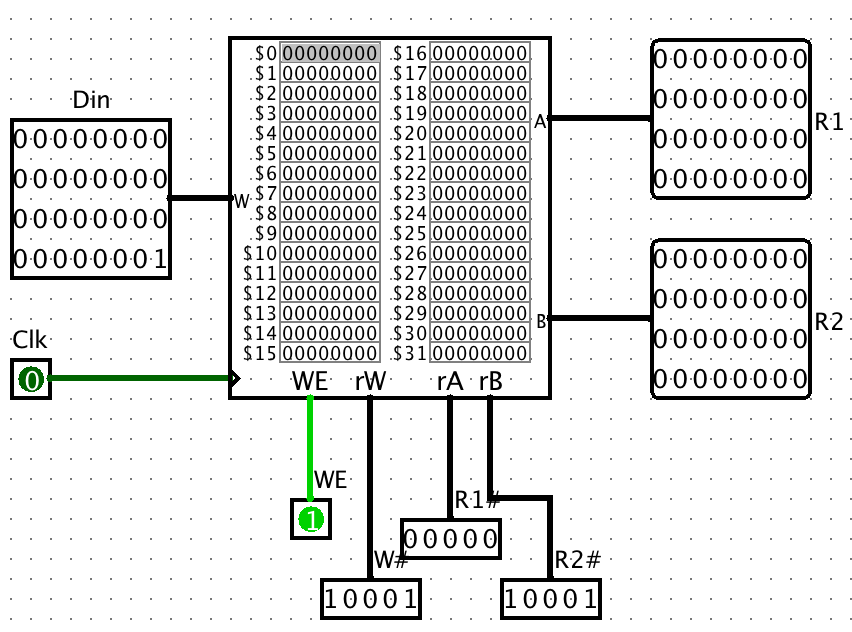


图 3.3寄存器文件（RegFile）

1. 数据存储器(DM)

Logism实现：使用4个位宽为8的随机存储器RAM实现，输入地址32位，取2-11位作为DM的地址，如图3.4所示。

手机屏幕的截图

描述已自动生成

图 3.4数据存储器(DM)

### 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 |  |  |
| ADDI | PC+4 | PC | rs |  | rt | alu | r1 | 立即数 | 5 |  |  |
| ADDIU | PC+4 | PC | rs |  | rt | alu | r1 | 立即数 | 5 |  |  |
| ADDU | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 5 |  |  |
| AND | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 7 |  |  |
| ANDI | PC+4 | PC | rs |  | rt | alu | r1 | 立即数 | 7 |  |  |
| SLL | PC+4 | PC |  | rt | rd | alu | r2 | 立即数 | 0 |  |  |
| SRA | PC+4 | PC |  | rt | rd | alu | r2 | 立即数 | 1 |  |  |
| SRL | PC+4 | PC |  | rt | rd | alu | r2 | 立即数 | 2 |  |  |
| SUB | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 6 |  |  |
| SUBU | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 6 |  |  |
| OR | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 8 |  |  |
| ORI | PC+4 | PC | rs |  | rt | alu | r1 | 立即数 | 8 |  |  |
| NOR | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 10 |  |  |
| SLT | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 1 |  |  |
| SLTU | PC+4 | PC | rs | rt | rd | alu | r1 | r2 | 1 |  |  |
| JR | PC+4 | PC | rs |  |  |  |  |  | X |  |  |
| SYSCALL | r1 | PC | v0 |  |  |  | r1 | 10 | X |  |  |
| J | (PC&0XF000) | PC |  |  |  |  |  |  | X |  |  |
| JAL | (PC&0XF000) | PC |  |  | 31 | PC+4 |  |  | X |  |  |
| BEQ | PC+4+offset | PC |  |  |  |  |  |  | X |  |  |
| BNE | PC+4+offset | PC |  |  | 31 | PC+4 |  |  | X |  |  |
| LW | PC+4 | PC | rs |  | rt | DM.Dout | r1 | 立即数 | 5 | ALU |  |
| SW | PC+4 | PC | ra | rt |  |  | r1 | 立即数 | 5 | ALU | RF.D |

数据通路见图3.5。

图片包含 游戏机

描述已自动生成

图 3.5 单周期CPU数据通路（Logism）

### 控制器的实现

根据表2.6的主控制器控制信号框架，将对应值填入自动生成真值表达式的execel文件中，将表达式填入logisim的自动生成电路功能，生成运算器控制器、控制信号生成的电路，扩展指令单独建立一个模块，步骤同上，最后将3个模块的输出信号引到输出引脚，将控制器封装，控制器实现见图3.6。

手机屏幕截图

描述已自动生成

图 3.6 控制器实现

## 中断机制实现

### 中断信号的产生

中断信号作为D触发器的时钟端输入，假如按下相当于触发器经历了一个上升沿，被锁存在其中，下一个时钟周期D触发器IR会变为1，持续产生中断信号直到同步清零信号将触发器清零。如图3.7。

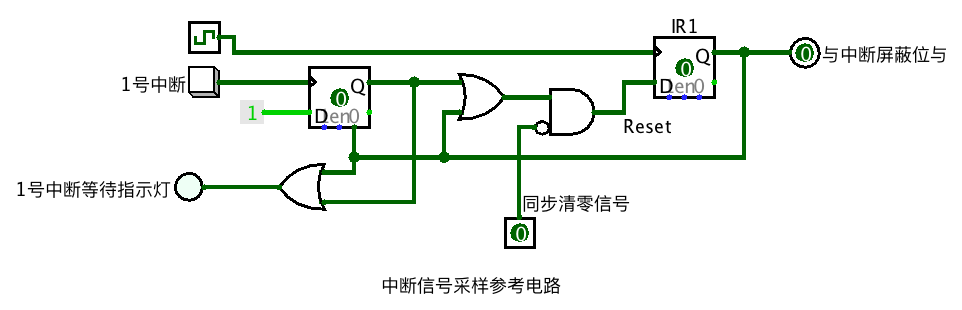


图 3.7 中断信号产生电路

### 中断入口地址选择

单级中断中，将3个中断请求信号连上优先编码器，输出为当前优先级最高的中断信号，即3>2>1，根据2位宽的中断编号决定最终选用的中断程序入口地址。中断入口地址选择电路如图3.8。

图片包含 游戏机, 钟表

描述已自动生成

图3.8 单级中断入口地址选择电路

多级中断的入口选择会更加复杂，因为要记住被打断时当前的中断号，而进入新的中断子程序时可能又被更高优先度的子程序中断，需要记住这一重的中断号，那么最开始记住的中断号就会丢失。为了解决这个问题，我设计了一个容量为3的硬件栈，可以将中断时的当前中断号压栈，在遇到ERET后令栈顶元素（上一个中断号）出栈，从而实现中断号的记忆。

输入端ir是当前中断号，sig是中断信号，高电平时计数器正向计数，表示栈元素增加，同时通过解码器令代写寄存器的写使能为1；eret高电平时计数器反向计数，表示有元素出栈，计数器的值是多路选择器的选择输入端，决定栈顶元素是哪个D寄存器的值。电路图见图3.9。

图片包含 游戏机, 文字, 钟表

描述已自动生成

图3.9 中断号硬件栈电路

多级中断决定入口地址同单级中断，看优先选择器的输出，如图3.10。

图片包含 游戏机, 钟表

描述已自动生成

图3.10 多级中断入口地址选择电路

### 中断信号清零

单级中断，当ERET指令到来时，ERET信号与自身中断信号通过与门，求反后作为IR的输入信号，上升沿到来时，将中断信号清零，见图3.11。

手机屏幕的截图

描述已自动生成

图3.11 单级中断信号清零

多级中断因为用到了一个2位宽的D寄存器保存当前中断号，因此需要用一个解复用器将ERET的高电平信号输入到对应IR前的与非门里，清空当前控制中断的寄存器，如图3.10。

### 中断使能寄存器IE和EPC寄存器的实现

多级中断中需要实现开关中断的指令，MFC0和MTC0分别对应开中断和关中断。ERET为1时，将IE寄存器清零，意味着开中断，关中断信号产生时IE使能端为0，如图3.12。

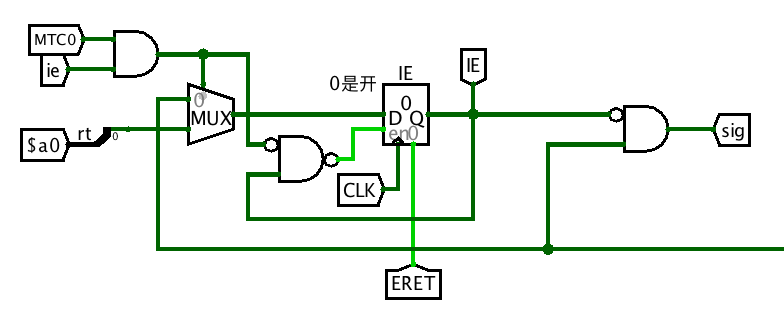


图3.12 中断使能寄存器IE

## 流水CPU实现

### 流水接口部件实现

程序计数器和所有锁存器采用同一个时钟信号，用寄存器组存放需要的数据和控制信号，将清零端设为常量0，高电平同步清零，所有输入信号采用对应位宽的寄存器输出，按时钟上升沿触发，使能端低电平有效，高电平忽略时钟输入。如图3.13。

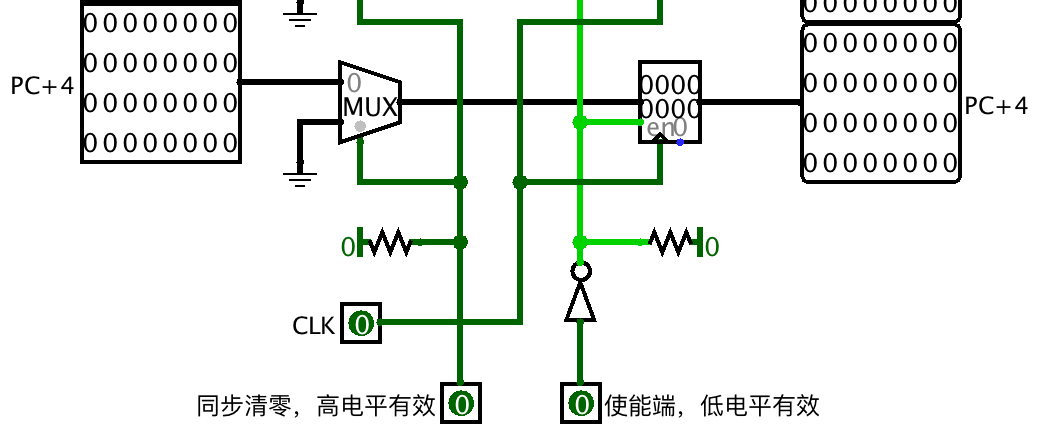


图3.13 流水接口

### 理想流水线实现

数据通路被4段寄存器组划分为五个部分：IF, ID, EX, MEM和WB，寄存器根据左右两部分的名称命名，分为IF/ID, ID/EX, EX/MEM和MEM/WB。每个寄存器组都由图3.13的基本单元构成，可以锁存各种位宽的数值，并在流水线中传递信息，五段流水的理论加速比为5:1。

理想流水见图3.14。

图片包含 游戏机

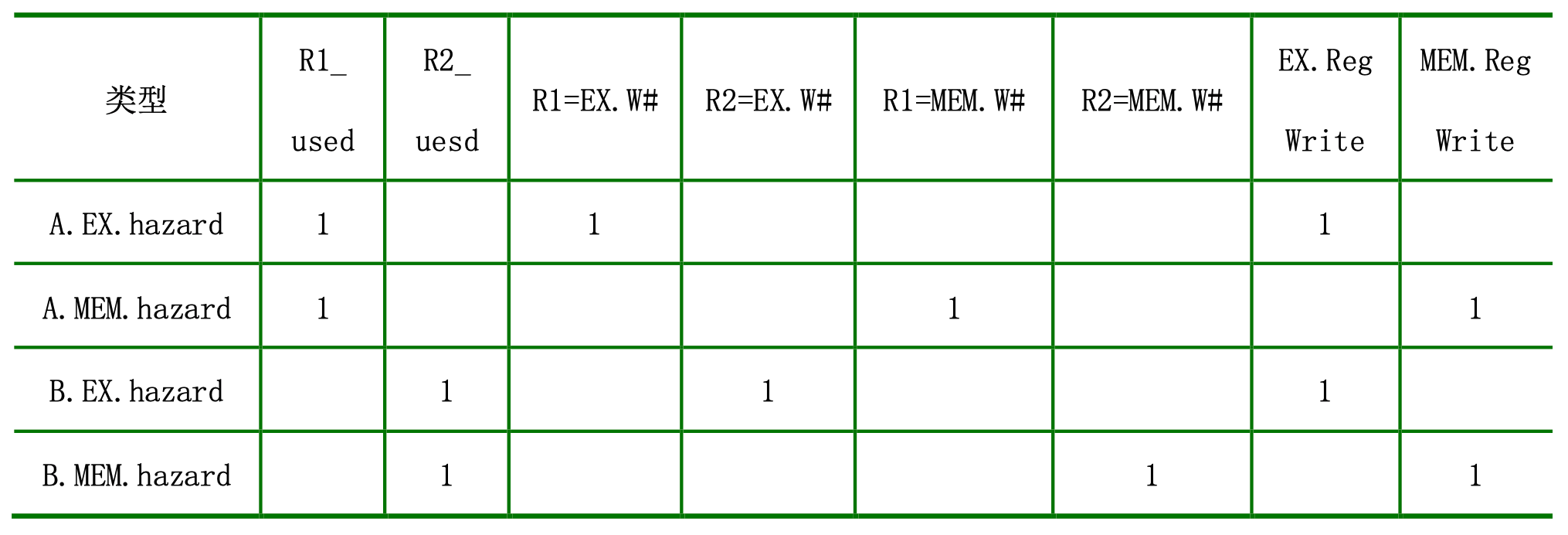
描述已自动生成

图3.14 理想流水

## 气泡式流水线实现

在ID段需要对指令进行数据相关检测，对于四种需要检测的数据相关，发生的条件见表3.1。

图3.1 数据相关产生条件



绘制数据相关检测电路如图3.15。



图3.15 数据相关检测电路

## 数据重定向流水线实现

在气泡流水线的基础上增加重定向数据通路，客服执行效率底下的缺点，还需检测load-use数据相关，解决流水线的关键时延会变长的缺点。

重定向控制器改造电路见图3.16。

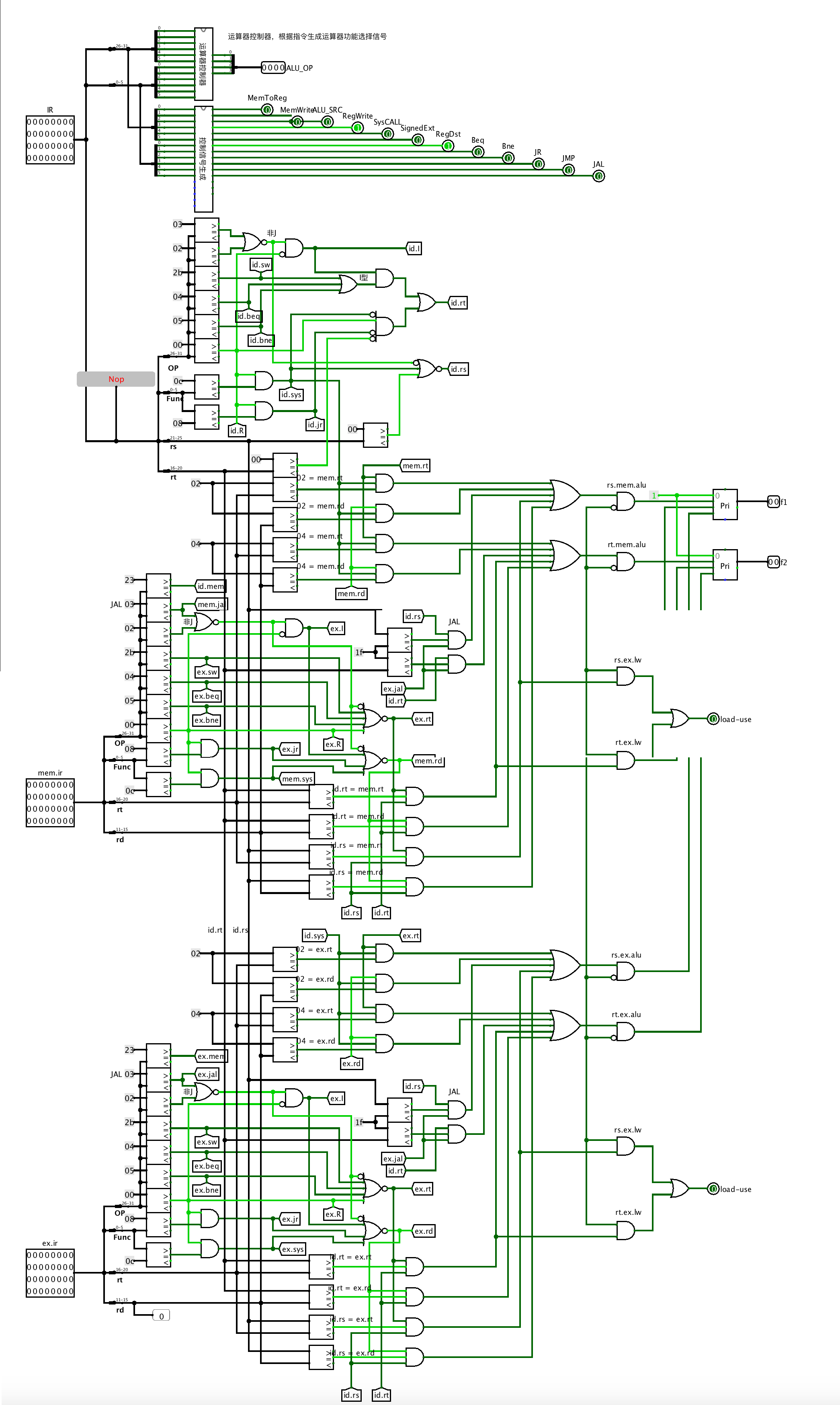


图3.16 重定向控制信号生成电路

# 实验过程与调试

## 测试用例和功能测试

对各个任务用测试文件分别单独测试，除了多级中断执行中断服务程序，理想流水线执行简化后的测试程序，其他任务都用benchmark标准程序进行测试。最后还要测试自己的扩展指令：SUBU, LUI, LH和BLEZ。

### 单周期CPU

加载benchmark.hex，按cmd+k启用信号模拟，结果见图4.1。

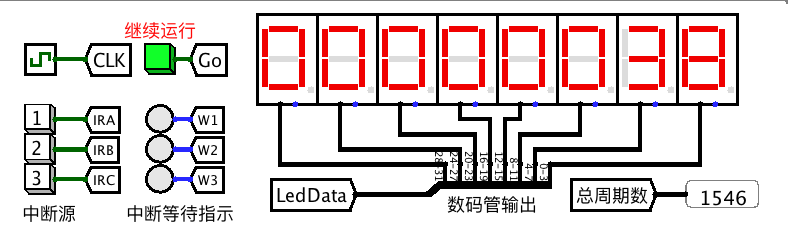


图4.1 单周期CPU测试结果

### 扩展指令

1. SUBU

将SUBU的asm测试文件用Mars打开，并dump成hex文件，最后加载到指令存储器中并运行，依此输出：

0x00000010 0x0000000f 0x0000000e 0x0000000d 0x0000000c 0x0000000b 0x0000000a 0x00000009 0x00000008 0x00000007 0x00000006 0x00000005 0x00000004 0x00000003 0x00000002 0x00000001 0x00000000 0xffffffff 0xfffffffe 0xfffffffd 0xfffffffc 0xfffffffb 0xfffffffa 0xfffffff9 0xfffffff8 0xfffffff7 0xfffffff6 0xfffffff5 0xfffffff4 0xfffffff3 0xfffffff2 0xfffffff1 0xfffffff0

最终结果见图4.2。

图片包含 游戏机, 钢琴, 画

描述已自动生成

图4.2 SUBU测试结果

1. LUI

依次输出：

0xfedcffff 0xba980000 0x76540000 0x32100000 0xfedcffff 0xba980000 0x76540000 0x32100000 0xfedcffff 0xba980000 0x76540000 0x32100000 0xfedcffff 0xba980000 0x76540000 0x32100000 0xfedcffff 0xba980000 0x76540000 0x32100000 0xfedcffff 0xba980000 0x76540000 0x32100000 0xfedcffff 0xba980000 0x76540000 0x32100000 0xfedcffff 0xba980000 0x76540000 0x32100000

最终结果见图4.3。

卡通人物

描述已自动生成

图4.3 LUI测试结果

1. LH

依次输出：

0xffff8281 0xffff8483 0xffff8685 0xffff8887 0xffff8a89 0xffff8c8b 0xffff8e8d 0xffff908f 0xffff9291 0xffff9493 0xffff9695 0xffff9897 0xffff9a99 0xffff9c9b 0xffff9e9d 0xffffa09f 0xffffa2a1 0xffffa4a3 0xffffa6a5 0xffffa8a7 0xffffaaa9 0xffffacab 0xffffaead 0xffffb0af 0xffffb2b1 0xffffb4b3 0xffffb6b5 0xffffb8b7 0xffffbab9 0xffffbcbb 0xffffbebd 0xffffc0bf

最终结果见图4.4。

图片包含 游戏机, 钢琴, 画

描述已自动生成

图4.4 LH测试结果

1. BLEZ

依次输出：

0xfffffff1 0xfffffff2 0xfffffff3 0xfffffff4 0xfffffff5 0xfffffff6 0xfffffff7 0xfffffff8 0xfffffff9 0xfffffffa 0xfffffffb 0xfffffffc 0xfffffffd 0xfffffffe 0xffffffff 0x00000000

最终结果见图4.5。

图片包含 游戏机, 画, 钟表

描述已自动生成

图4.5 BLEZ测试结果

### 单级中断

主程序的输出内容为1-F左右循环的跑马灯，第一号中断的输出内容为倒数1…3, 1…2, 1…1，第二号中断的输出内容为2…3, 2…2, 2…1，以此类推。

按cmd+k开始仿真，一开始就依此产生1~3号中断，于是先进入1号中断子程序，执行完成后进入优先度更高的3号中断，见图4.6。

手机屏幕截图

描述已自动生成 手机屏幕截图

描述已自动生成

图4.6 进入1号中断后进入3号中断

3号中断子程序运行完毕后进入2号中断程序，结束后回到主程序，接着遇到了3号中断，见图4.7。

图片包含 游戏机, 画

描述已自动生成 图片包含 游戏机, 钢琴, 画

描述已自动生成

图4.7 进入2号中断后回到主程序

遇到3号中断后进入第3号中断子程序，在运行过程中遇到1号中断，不被打断直到3号中断子程序执行完毕，之后运行1号中断子程序，见图4.8。

手机屏幕截图

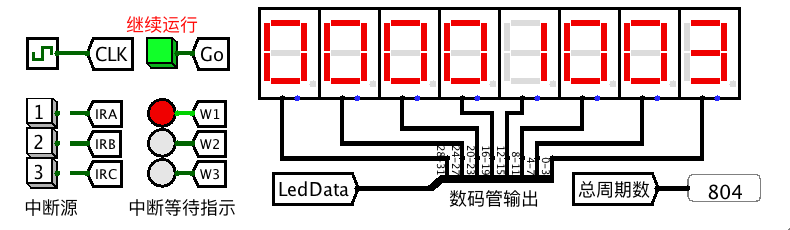
描述已自动生成 

图4.8 进入3号中断后进入1号中断

剩余的执行过程类似，结果正确，不演示。

### 多级中断

多级中断在第3.2.2章有过分析，测试结果如下：

在开始时也是顺序接收1, 2, 3号中断，但是多级中断允许打断，故进入优先度最高的3号中断子程序，执行完后进入2号中断子程序，见图4.9。

图片包含 游戏机, 画

描述已自动生成 图片包含 游戏机, 画

描述已自动生成

图4.9进入3号中断后进入2号中断

执行完2号中断后进入优先度最低而被搁置的1号中断子程序，执行完后接着进入3号中断子程序，见图4.10。

手机屏幕截图

描述已自动生成 手机屏幕截图

描述已自动生成

图4.10进入1号中断后进入3号中断

剩余的执行过程类似，结果正确，不演示。

### 理想流水线

加载理想流水线的测试程序，结果见图4.11。

手机屏幕的截图

描述已自动生成

图4.11 理想流水线运行情况

查看存储器中的内容，见图4.12。

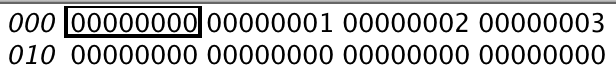


图4.12 理想流水线存储器情况

### 气泡流水线

加载benchmark，运行结果见图4.13。

手机屏幕的截图

描述已自动生成

图4.13 气泡流水线运行情况

### 气泡流水线单级中断

气泡流水单级中断是在单级中断电路的基础上增加了数据相关检测逻辑，并及时向流水线中插入气泡，执行情况与单级中断无异，一开始遇到1，2号中断而先来后到，结束后进入2号中断子程序，见图4.14。

图片包含 游戏机, 画

描述已自动生成手机屏幕截图

描述已自动生成

图4.14 气泡流水线单级中断运行情况

### 重定向流水线

由于采用EX段的重定向，总周期数为2298，见图4.15。

手机屏幕截图

描述已自动生成

图4.15 重定向流水线运行情况

## 性能分析

(1) 虽然单周期CPU的总周期数最少，CPI为1，而流水线CPU的CPI高很多，但实际上单周期CPU的关键路径时延高得多，时钟频率无法提的很高，导致执行时间严重被关键路径长的指令约束。

(2) 虽然看似气泡流水线是一种较为方便的解决冲突的妥协方式，但是实际上数据相关的指令和跳转指令在整个程序指令中的占比高达30%，因此会造成流水线资源极大地浪费，而同时高能耗与低功效不平衡导致最终能效比的低下。

(3) 优秀的分支预测策略能很好地解决数据相关和控制冲突，将其损失降低。

## 主要故障与调试

### 流水线指令执行故障

理想流水线：总周期数不正确。

**故障现象：**如图4.16所示，正确的总周期数为20，虽然之前的执行结果都是正确的，但是最终少一个周期。

图片包含 游戏机, 键盘, 钟表

描述已自动生成

图 4.16 程序故障图

**原因分析：**停机信号Halt要放到WB段，否则会因为在执行段EX停机导致MEM和WB段没有执行完整。

**解决方案：**如图4.17。

**手机屏幕截图

描述已自动生成**

图 4.17 解决方案

### 气泡流水线故障

气泡流水线：停在了第31个周期。

**故障现象：**如图4.18，流水线过早地接收到了停机信号。

**原因分析：**遇到syscall指令时，首先应该看$v0寄存器的值是否为34，是表示打印$a0的内容到led灯，否则执行停机，观察到syscall的前一条指令是addi $v0, $0, 34，说明$v0的写入有问题才发生的误将打印变成停机。最后观察气泡流水线的特点，发生数据相关的时候，清除之前段相关的指令，发现是结构冲突，寄存器组同时读写会有先后问题。

图片包含 游戏机, 钟表, 画

描述已自动生成

图 4.18 程序故障图

**解决方案：**打开RegFile，改上升沿的触发方式为下降沿，使得在读写寄存器前半个周期时数据就被成功写入了寄存器**。**

### 中断跑马灯显示异常

单级中断：跑马灯效果不正确。

**故障现象：**中断回来主程序的时候没有衔接上之前的断点，可能出现断层的现象。

**原因分析：**跑马灯错误只可能是寄存器读写的错误，在进入中断子程序前可能是进早了一步或者晚一步，回来的时候可能也是没有掌握好时机，导致中断子程序的寄存器值影响到了主程序的运行，所谓毫厘之差谬以千里。

**解决方案：**这是最麻烦的一个bug，只能单步调试，甚至跟着标准的运行结果去分析**。**

## 实验进度

表 4.1 课程设计进度表

| 时间 | 进度 |
| --- | --- |
| 第一天 | 复习组原CPU相关理论知识，阅读任务书，看慕课视频，与组员制定计划。 |
| 第二天 | 完成单周期CPU的控制信号表，使用Logisim搭建控制器。 |
| 第三天 | 继续画单周期CPU的数据通路，不断调试直到通过测试。 |
| 第四天 | 看理想流水线CPU的慕课，研究如何划分数据通路。 |
| 第五天 | 画4个流水线寄存器组，修修补补。 |
| 第六天 | 完成理想流水线CPU，不断调试直到通过测试，看气泡流水线的慕课。 |
| 第七天 | 画数据相关检测逻辑，通过测试，在气泡流水线的基础上重构控制器，不断调试。 |
| 第八天 | 通过了重定向流水线的测试。画单级中断。 |
| 第九天 | 通过单级中断测试，在此基础上画多级中断，通过测试。 |
| 第十天 | 在单级中断基础上加上气泡流水线的相关逻辑，通过了测试。 |

# 设计总结与心得

## 课设总结

基于对象的存储是为了克服当前基于块的存储存在的诸多难题，在存储接口和结构层次的重要发展。可以根据应用负载选择优化的存储策略。作了如下几点工作：

1. 单周期MIPS CPU，一共24+4条指令，扩展指令为：SUBU, LUI, LH和BLEZ，并通过了测试。
2. 完成了理想流水线、气泡流水线和重定向流水线，并通过了测试。
3. 完成了单周期单级和多级中断，气泡流水线单级中断，并通过了测试。

## 课设心得

受疫情影响，整个课设都是在家完成，和老师同学沟通都是通过网络，极大地增加了实验的钻研成本，尽管最终没有加入大二以来就令人“魂牵梦萦”的Verilog上板项目，本次课程设计也可以说是迄今为止所有实验以及课程设计中难度最大的一门。两个星期从早到晚的不懈努力以及国庆节假期的辛苦加班才终于完成了整个课程设计的设计任务。现在再来回顾整个课程设计的整个过程，满满的成就感自是不用说，但是其中也有不少的细节值得我去深思与体会。

课设刚开始的时候，看到了老师发的任务计划着实被吓得不轻，感觉根本无从下手，但是还是咬咬牙每天端坐在电脑前研究电路，一次次地调试，直到通过测试。万事开头难，一旦进入状态就要好好保持这份热忱，专心致志地去钻研，不要三心二意，这样才可以事半功倍。没有什么是克服不了的，取其上者得其中,取其中者得其下,取其下者,无所取焉，关键就是要有那个必胜的决心。

在整个课设过程中，有两个比较难过的坎，一个是流水线，一个是中断。由于我一开始把电路写得非常密集，要改成理想流水线时就犯了难，满世界拖组件，经常冒红，而调整寄存器组的时候又常要挪动周围的组件避免线打架，因此流水线是在磕磕碰碰中完成的。中断相对来说更棘手，首先是视频资料非常贫瘠，导致我不得不常去请教老师和同学，以及查阅课本和相关资料，然后是中断经常会遇到非常多诡异的bug，需要像福尔摩斯一样去耐心地侦破各种bug产生的原因，过程非常枯燥，以至于最后通过测试的一刻是每天的快乐之源。

然而对于本次课程设计，我还有一些小小的建议和改进。课设的时间周期跨度太大了，开学进行，8月结束交报告，有点虎头蛇尾的感觉，而我不善于写文档记录因此遗忘了非常多细节，导致报告相对空洞，希望以后能更好地安排课设的时间表。

最后在这里也感谢三位老师对于我在本次课程设计中无数问题的耐心解答，也感谢本组所有成员在课程设计中对于我的帮助和建议。我相信组成原理课程设计必将成为我整个大学生涯中一段无比难忘的回忆。

# 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:** |