***2016***



**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： | CS1307 |
| 学 号： | U201314975 |
| 姓 名： | 王映焜 |
| 电 话： | 15527366893 |
| 邮 件： | [316661644@qq.com](mailto:316661644@qq.com) |
| 完成日期： | 2016-02-05 周五下午 |

**课程设计任务书**

|  |
| --- |
| 一、设计题目 |
| **基于Logisim软件仿真平台的5段流水CPU设计实现** |
| 二、设计内容 |
| 设计模型机系统的总体结构、指令系统和时序信号。在对该模型机系统中的部件功能利用EDA软件的仿真功能进行仿真分析和功能验证的基础上，将部分电路下载到FPGA，并与适当的外围器件相配合，实现模型机的整机系统。要求所设计的整机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适时显示信息。 |
| 1. 设计要求 |
| 1. 支持28条基本指令，具体见表1和表2； 2. 支持5段流水机制，可处理数据冒险，结构冒险，分支冒险； 3. 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。 4. 能运行教师提供的标准测试程序，并统计执行时间。 |
| 四、设计流程 |
| 1. 根据课程设计指导书的要求，制定出设计方案； 2. 画出自己所设计计算机系统的原理框图，分析所需要的控制信号以及这些控制信号的有效形式； 3. 画出各指令的指令周期流程图和所需要的控制信号； 4. 设计出实现指令功能的控制器； 5. 调试、数据分析、验收检查； 6. 课程设计报告和总结。 |
| 五、成绩评定 |
| 成绩评定根据考勤、课程设计的过程、课程设计的效果、课程设计报告质量等进行综合评定；其中设计过程和结果占70%，课程设计报告占30%；课程设计的成绩评定等级为不及格、及格、中、良好、优秀五级；对基本功能进行扩展或设计具有非常鲜明的特征和一定程度的创新，可根据实际情况加分。 |
| 六、设计报告要求 |
| 课程设计报告主要内容包括：设计题目、设计目的、设备器材、设计原理及内容、设计步骤、遇到的问题及解决方法、设计总结、参考文献等。要求在适当位置配合相应的实验原理图、数据通路图、实验接线图等图表进行说明。总结部分主要写设计工作简介以及设计体会。应做到文理通顺，内容正确完整，书写工整，装订整齐。课程设计报告采用《计算机组织与结构》专用设计报告模板，A4纸双面打印。 |
| 七、时间安排 |
| 课程设计的总体时间为2周，具体安排如下:   1. 第1天：到实验室布置任务和集中讲解。 2. 第1~3天：学生查阅资料，开始方案设计。 3. 第4天：中期进度检查，单周期CPU验收检查。 4. 第6天：中期进度检查，理想流水线多周期CPU验收检查。 5. 第10天：最终结果验收。 |
| 八、主要参考文献 |
| 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. 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. 5. 张晨曦，王志英. 计算机系统结构. 高等教育出版社，2008年. |

目 录

[1 课程设计概述 3](#_Toc14602)

[1.1 课设目的 3](#_Toc13971)

[1.2 设计任务 3](#_Toc9628)

[1.3 设计要求 3](#_Toc20039)

[2 实验原理与环境 4](#_Toc11492)

[2.1 实验原理 4](#_Toc6496)

[2.2 实验环境 4](#_Toc17507)

[3 总体方案设计 5](#_Toc5546)

[3.1 需求分析 5](#_Toc32722)

[3.2 硬件设计 5](#_Toc26766)

[3.3 软件设计 7](#_Toc13613)

[4 详细设计与实现 9](#_Toc1415)

[4.1 选用芯片 9](#_Toc20933)

[4.2 硬件实现 9](#_Toc21851)

[4.3 软件实现 12](#_Toc28949)

[5 实验过程与调试 17](#_Toc19489)

[5.1 仿真XXX 17](#_Toc2879)

[5.2 可自行安排章节 17](#_Toc3152)

[5.3 主要故障与调试 17](#_Toc78)

[5.4 功能测试 17](#_Toc26477)

[5.5 实验流程图 18](#_Toc11959)

[6 设计总结与心得 19](#_Toc20759)

[6.1 课设总结 19](#_Toc30381)

[6.2 课设心得 19](#_Toc16254)

[参考文献 20](#_Toc22847)

# 课程设计概述

**请仔细阅读所有的批注，阅读理解后删除批注**

**模板各个标题下面的内容仅是举例，作者应依照自己思想重写该部分内容**

## 课设目的

计算机组成原理是计算机专业的核心专业基础课。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，而且通过进行设计及实现，进一步提高分析和解决问题的能力。（自己写）

……（其它内容）

## 设计任务

计算机系统设计的总体目标是设计模型机系统的总体结构、指令系统和时序信号。所设计的主机系统能支持自动和单步运行方式……（自己写）

具体设计任务如下：……（其它内容）

## 设计要求

根据理论课程所学的至少，设计出简单计算机系统的总体方案，结合各单元实验积累和课堂上所学知识，选择适当芯片，设计简单的计算机系统，具体要求如下：

1. 根据课设指导书的要求，制定设计方案。（自己写）
2. 画出自己所涉及计算机系统的原理图和器件连接头……
3. ……（其它内容）

# 实验原理与环境

## 实验原理

计算机组成原理，数字逻辑， FPGA(Field Programmable Gate Array)是……。等其他知识…….尽量做到图文并茂。

## 实验环境

本次试验环境是在自己的Laptop上做的，在Java下使用Logisim平台进行试验的仿真，使用Mars进行汇编程序的编写与测试。具体测试环境见表 2.1。

表 . 实验环境配置

|  |  |
| --- | --- |
| Configuration | Laptop |
| CPU | Intel Core i5-3230M @2.60GHz |
| Memory | 16GB |
| OS  Platform | Windows 10 Pro  Java 1.8.0\_37 |

# 总体方案设计

## 构建单周期CPU

本次试验的单周期CPU要支持27条指令，可以在之前组成原理的第三次试验的基础上进行扩展。由于之前的实验已经对运算器ALU进行了封装，也对寄存器组RegFile进行了封装，这些功能部件可以直接使用已封装的。首先根据MIPS手册查找每一条指令的指令格式及其实现的功能，分析其操作的数据通路，并绘制数据通路表（如图 3‑1）。



图 3‑1 数据通路表

之后根据数据通路表分析归纳控点信号。先统计出每个控点信号需要进行的选择通路的个数，以便对控点的选择信号进行编码，整理出控点信号表，如图 3‑2。进行了选择信号的编码，方便之后使用真值表生成电路。

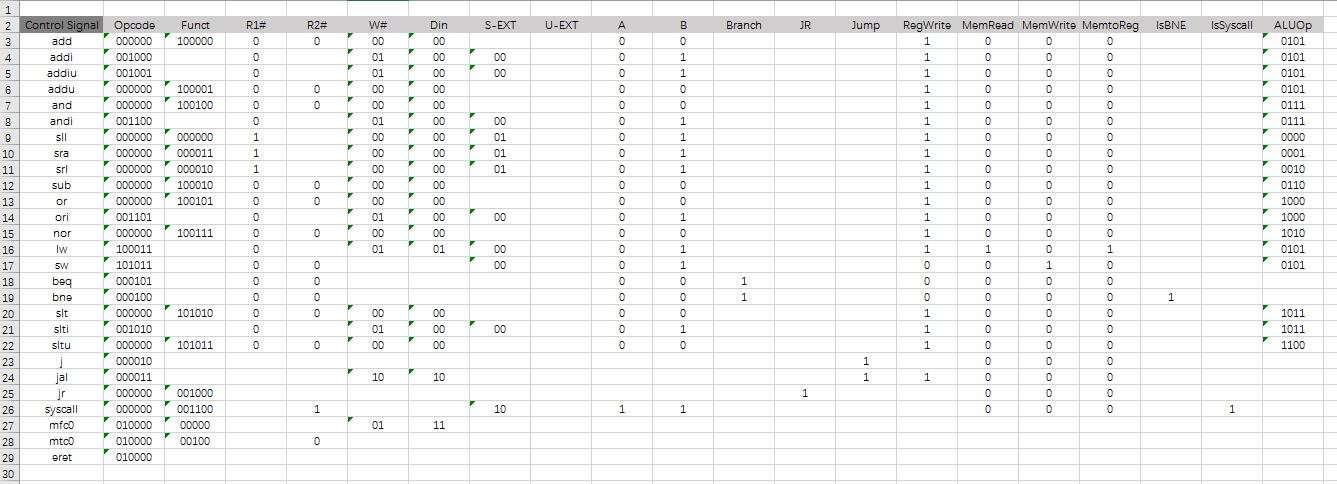


图 3‑2 控点信号表

根据控点信号表构建Control模块，再根据数据通路高层透视图，如图 3‑3，将主要功能部件之间的数据流向进行初步连接。

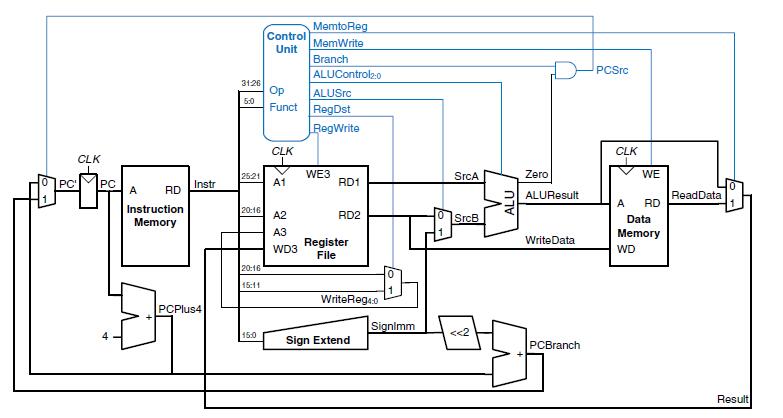


图 3‑3数据通路高层透视图

之后再根据每一条指令的特点以及每个部件的特点进行相应的改动和调整，完善该CPU的数据路线及相应的控点选择器的输入和输出信号。

最后参照任务文件夹下的测试程序进行汇编，得到十六进制的机器码文件，装填到CPU分别进行J指令测试、B指令测试、排序测试、移位测试、走马灯测试之后，再通过基准测试程序，即完成单周期CPU的构建。

### 总体设计

本次CPU的设计是采用硬布线设计，主要部件包括PC计数器、指令存储器IM、寄存器组RF、运算器ALU、数据存储器DM；主要逻辑部件包括立即数扩展器、地址转移逻辑NPC，控制器Control。

控制器Control模块对每条指令的解析全部是由逻辑门组合实现的；PC计数器即logisim寄存器，其和一个加法器构成PC+4的控制逻辑；指令存储器IM使用了一个logisim内置的元器件ROM，其地址位宽为10，数据位宽为32；寄存器组RF使用了上学期课程实验二中封装好的寄存器组RegFile，并对其封装进行了稍微的改善，去掉了最外层封装的文字部分，后面调用该部件时，便于连线；运算器ALU使用了上学期课程实验一中封装好的支持13种运算模式的运算器；数据存储器DM使用了一个logisim的元器件RAM，其地址位宽为16，数据位宽为32；立即数扩展直接使用了logisim的元器件有符号数16位到32位的扩展器；地址转移逻辑NPC在原来实验三的电路的基础上进行的改进，增加了J型指令的地址转移控制；控制器Control模块的设计是根据控点信号表转换来的真值表，利用logisim内置的真值表生成电路的功能，进行了电路的生成和改善，并最终封装成模块形式。

之后按照数据通路高层透视图（如图 3‑3）， 将各个部件进行布局，之后在各个控点信号处按照数据来源的个数放置不同控制位宽的数据选择器，再按照数据通路表的数据来源和去向进行数据通路线路的连接。然后按照控点信号表中给出的真值表将各个控点数据选择器的控制端的信号进行电路生成，并封装在Control模块中，最后再使用隧道将各个控点的控制信号和Control模块的输出进行匹配连接。最后通过测试程序进行测试即可。

### 主要功能部件

由于本实验是在logisim平台下进行的，logisim内置库中给出了很多功能完善的部件，以便在构建各个功能部件的时候进行相应的调用。每个功能部件的实现均是根据其功能和特点，使用适当的逻辑部件或已封装好的组件，即可构建各个功能部件。其中运算器是使用上学期课程实验一中已经封装好的ALU，其支持13种运算操作；寄存器组使用上学期课程实验二中已经封装好的RegFile，其为一个32个寄存器组成的寄存器组，符合MIPS通用寄存器组的特点及操作。下面是其他几个关键功能部件：

#### 程序计数器PC

程序计数器PC即一个指向当前指令的指针，由于在单周期CPU的运行过程中，每个时钟到来时均要对其值进行更新，故使用寄存器组件，在每个时钟上升沿的时候将其数据端的值送入寄存器中，其输出一直为其当前的值，符合对程序计数器功能的要求，由于该CPU为32bit的，故其数据位宽为32。

#### 指令存储器IM

指令存储器IM相当于存储着程序的固件，其内容应该是不会轻易改变的，故使用只读存储器ROM，由于该实验中的程序都比较小，不需要过大的存储空间，以免存在资源浪费，故其地址位宽为10。

#### 数据存储器DM

数据存储器DM，即为计算机系统中的主存，其中存储的内容在程序运行的过程中会根据程序内容改变，故使用随机存储器RAM，其地址位宽为16，理由同指令寄存器。

### 构造R型指令

R型指令的指令格式如图 3‑4：



图 3‑4 R型指令指令格式

对于R型指令，其OP位均为000000，通过OP位即可判断是否为R型指令，对于R型指令之间的区分需要用funct字段，不同R型指令拥有不同的funct字段。若判断为R型指令，只需通过funct字段得到该指令为何，解析其Rs、Rt、Rd段位，并根据这条指令的数据通路和数据流向即可得到其要表达的功能。

### 构造I型指令

I型指令的指令格式如图 3‑5：



图 3‑5 I型指令指令格式

对于I型指令，其只需参照OP位即可，根据OP位的不同，可以比较得到各个I型指令，根据不同I型指令的功能，解析Rs、Rt和立即数段位，并根据这条指令的数据通路和数据流向即可得到其要表达的功能。

### 构造J型指令

J型指令的指令格式如图 3‑6：



图 3‑6 J型指令指令格式

对于J型指令，由于需要实现的指令只有J指令和JAL指令，可以只针对这两条指令进行J型指令的解析，只需参照OP位即可，根据OP位的不同，可以比较得到这两个J型指令，根据不同J型指令的功能，解析立即数段位，并根据这条指令的数据通路和数据流向即可得到其要表达的功能。

### DISP指令

DISP指令的实现是通过在解析到syscall指令的时候，比较v0寄存器的值是否等于10来判断的，当v0寄存器的值不等于10时，DISP信号即有效，此时在数码管上显示a0寄存器的值。

### halt指令

halt指令的实现是和DISP指令的实现同步的，都是在解析到syscall指令的时候，比较v0寄存器的值，当等于10的之后，halt信号有效，即断掉时钟。

### 控制器封装

利用logisim根据控点信号表中的真值表来生成电路，根据输入的指令的OP段和funct段进行解析并输出一些列控点信号。由于logisim生成电路的输出个数最大限制为12，所以需要把所有的控点信号的输出拆分成两个子电路，并进行合并，最终封装为Control模块。

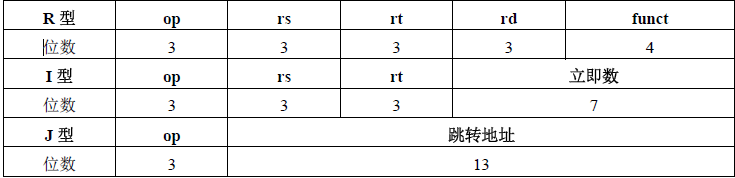
## 可支持理想流水线的多周期CPU设计

### 总体设计

指令的格式设计是……指令的寻址方式设计是……(请自行扩展修改)指令的格式设计是……指令的寻址方式设计是……(请自行扩展修改)指令的格式设计是……指令的寻址方式设计是……(请自行扩展修改)指令的格式设计是……指令的寻址方式设计是……(请自行扩展修改)指令的格式设计是……指令的寻址方式设计是……(请自行扩展修改)

采用定长指令每条指令的长度为16位，总共有3种类型，如表3.1所示。(请自行扩展修改)

表 . 各类型指令的划分



表中op字段决定是什么指令，rs、rt、rd为三个寄存器编号，funct字段为算术指令的功能选择字段，决定进行什么样的运算。(请自行扩展修改)

对于寄存器的编号，由于我们的系统总共有8个寄存器，故使用3位二进制码将它们编号（如1号寄存器编号为001xxxxxxx…等等(请自行扩展修改)

存储工业已经开始对接口提出更多要求，希望接口能做更多的事情。自从1956年第一块磁盘驱动器诞生以来，磁盘密度和性能已经分别提高了6个、4个数量级，然而存储接口没有根本性的改变——仍然是基于数据块（block）。

### 接口部件设计

微指令的设计思想是……

微指令的设计方案是……

设计的微指令流程……

……（根据自己的实际设计情况进行撰写）

### 流水冲突检测器

### 插入气泡的流水冲突处理

### 数据重定向的流水冲突处理

### 中断机制实现

# 详细设计与实现

指令周期流程图要在此部分出现、微程序流程图、微指令代码表、实验接线图等均需要在适当的位置和模块中表达出来。本章具体实现细节尽量多用图表方式展示，但要做到图文并茂，不能全文都是图。

## 构建单周期CPU

### 主要功能部件实现

#### 程序计数器PC

**选用的元器件**：寄存器（数据位宽： 32， 触发方式：上升沿，标签： PC），时钟隧道，输入引脚（默认参数值）。

**输入**： 时钟信号（1bit），PC将要指向的下一条指令的地址数据（32bits）。时钟信号是主电路的时钟输入，所以设计为隧道方式引入到 PC 部分。 时钟隧道（朝向：北，数据位宽： 1， 标签：CLK）。

**输出**： 寄存器的输出端（32bits）。

**具体实现**： 寄存器的时钟端连线时钟隧道，寄存器的使能端连接高电平，寄存器的数据端连接的是下一次PC将要更新的值的数据，其输出连接到后面的指令存储器，作为指向当前指令的指针，具体电路见图 4‑1。

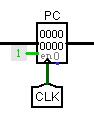


图 4‑1 程序计数器PC电路图

#### 指令存储器IM

**选用的元器件**：只读存储器ROM（地址位宽： 10， 数据位宽： 32），分线器，输入引脚（默认参数值）。

**输入**： 来自PC输出端的寄存器值，经过分线器，将寄存器值得2-11位分离出来送入ROM的地址端。

**输出**： ROM的输出端（数据端）。

**具体实现**： PC寄存器的输出地址连接到分线器，分离得到地址数据的2-11位，再将分离得到的数据接到ROM的地址端即可。由于模拟实际计算机中指令地址的分布，每条指令为32bits，需要占用4个字节，所以每次PC都是加4再送回，而该ROM的输入地址单位为字（4个字节），故将PC的低两位屏蔽掉，得到的值即为ROM中每条指令的地址值。具体电路见图 4‑2。

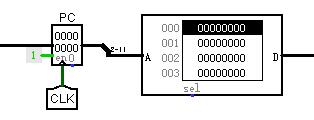


图 4‑2 指令寄存器IM电路图

#### 寄存器组RF

**选用的元器件**：寄存器（数据位宽： 32， 触发方式：上升沿，标签： 各个寄存器编号），隧道，数据选择器，输入引脚（默认参数值）。

**输入**： 读寄存器1的编号R1#（5bits），读寄存器2的编号R2#（5bits），写入寄存器编号W#（5bits），写入数据Din（32bits），写入使能信号WE（1bit），时钟信号CLK（1bit）。

**输出**： R1#寄存器的值R1（32bits），R2#寄存器的值R2（32bits），编号为16的寄存器的值$s0（32bits），编号为17的寄存器的值$s1（32bits），编号为18的寄存器的值$s2（32bits），编号为31的寄存器的值$ra（32bits）。

**具体实现**：首先需要把32个寄存器放置到适当位置，建立数据通路。在写入数据的输入隧道处接上一个数据选择器，其中选择信号为写入寄存器编号，即把写入数据送入到各个寄存器的数据端，其中0号寄存器送入的为常数0。再使用数据选择器进行寄存器使能端信号的选择，将写入寄存器编号的使能端置为WE。在输出时使用两个个数据选择器，其中选择信号分别为R1#和R2#，即将两个读寄存器编号对应的寄存器的值输出。具体电路见图 4‑3。

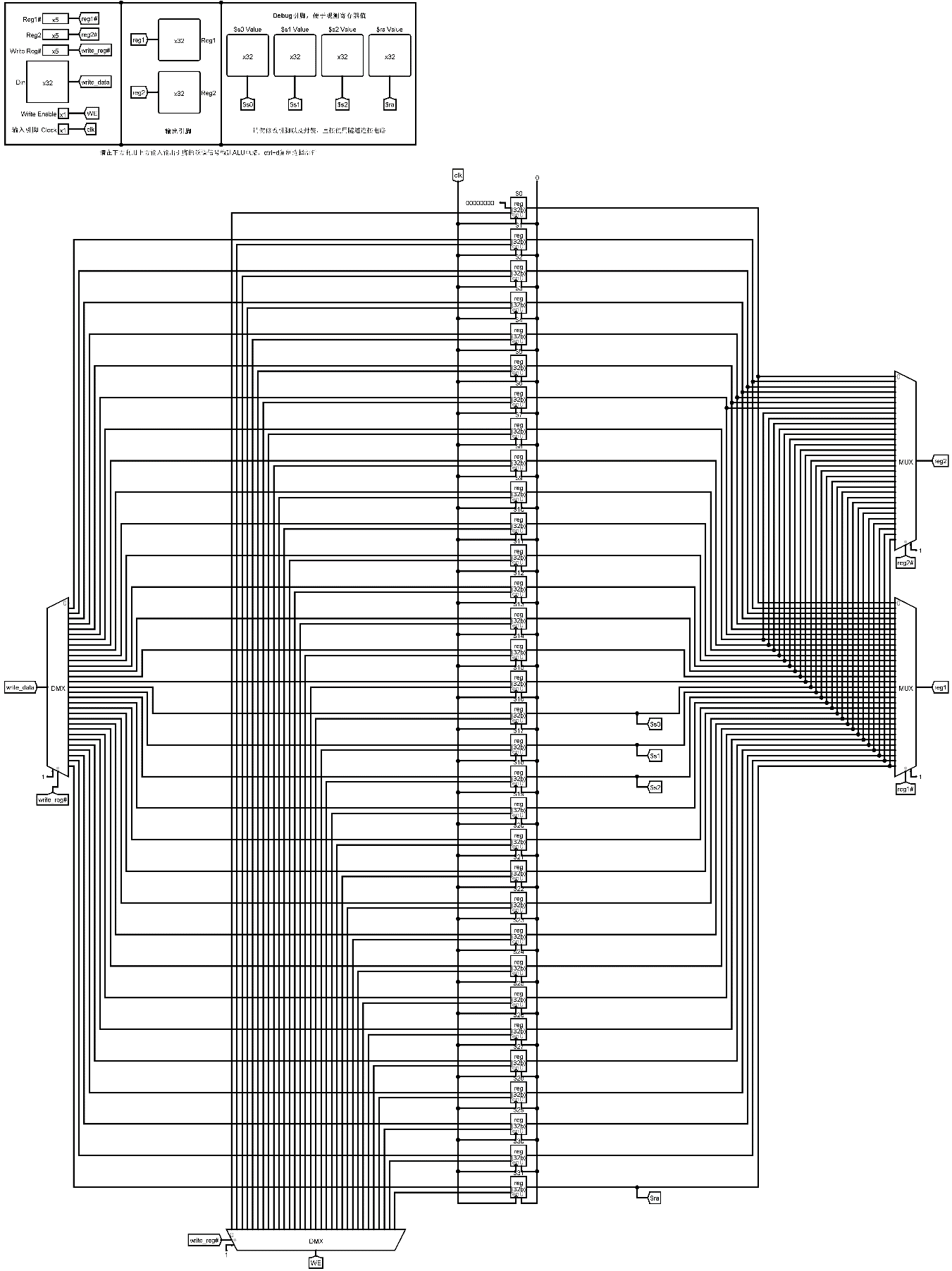


图 4‑3 寄存器组RF电路图

#### 运算器ALU

**选用的元器件**： 移位器（数据位宽： 32），乘法器（数据位宽： 32），除法器（数据位宽： 32），加法器（数据位宽为31和1的各一个），减法器（数据位宽为31和1的各一个），比较器（关于2的补码的和无符号的各一个，位宽均为32），数据选择器（数据位宽： 32，选择数据位宽 ：1），隧道，分线器，时钟（默认参数值），输入引脚（默认参数值）。

**输入**： 操作数X（32bits），操作数Y（32bits），运算器功能码ALU\_OP（4bits）。

**输出**： ALU运算结果Result（32bits），ALU运算结果第二部分Result2（32bits），有符号加减溢出标记OF（1bit），无符号加减溢出标记CF（1bit），两操作数是否相等Equal（1bit）。

**具体实现**： 该运算器要实现13种运算功能，故将每种运算功能按模块进行了封装。大部分模块都是直接调用logisim内置库中的元器件，其中加法和减法模块中，需要使用分线器将X和Y值进行最高位和其余31位的分离，之后将低31位的数相加（减）后，再将最高位进行加（减），最高位加（减）时加法器的进（借）位即低31位进行加（减）时的进（借）位输出，此时最高位进（借）位输出即为CF，CF和低31位的进（借）位进行异或即可得到OF；比较器模块的输出不为比较器直接的输出，需要使用数据选择器，其输入端为32位的数字0和1，选择信号为比较器小于输出。封装之后的具体电路见图 4‑4。

#### 数据存储器DM

**选用的元器件**：随机存储器RAM（地址位宽： 16，数据位宽： 32），分线器，隧道，输入引脚（默认参数值）。

**输入**： ALU的输出经过分线器得到其值得2-17位的数据（16bits），写入RAM的数据（32bits），存储控制端（1bit），芯片选择信号（1bit），时钟信号（1bit），加载信号（1bit），清空信号（1bit）。

**输出**： 读取的对应输入地址在RAM中的数值。

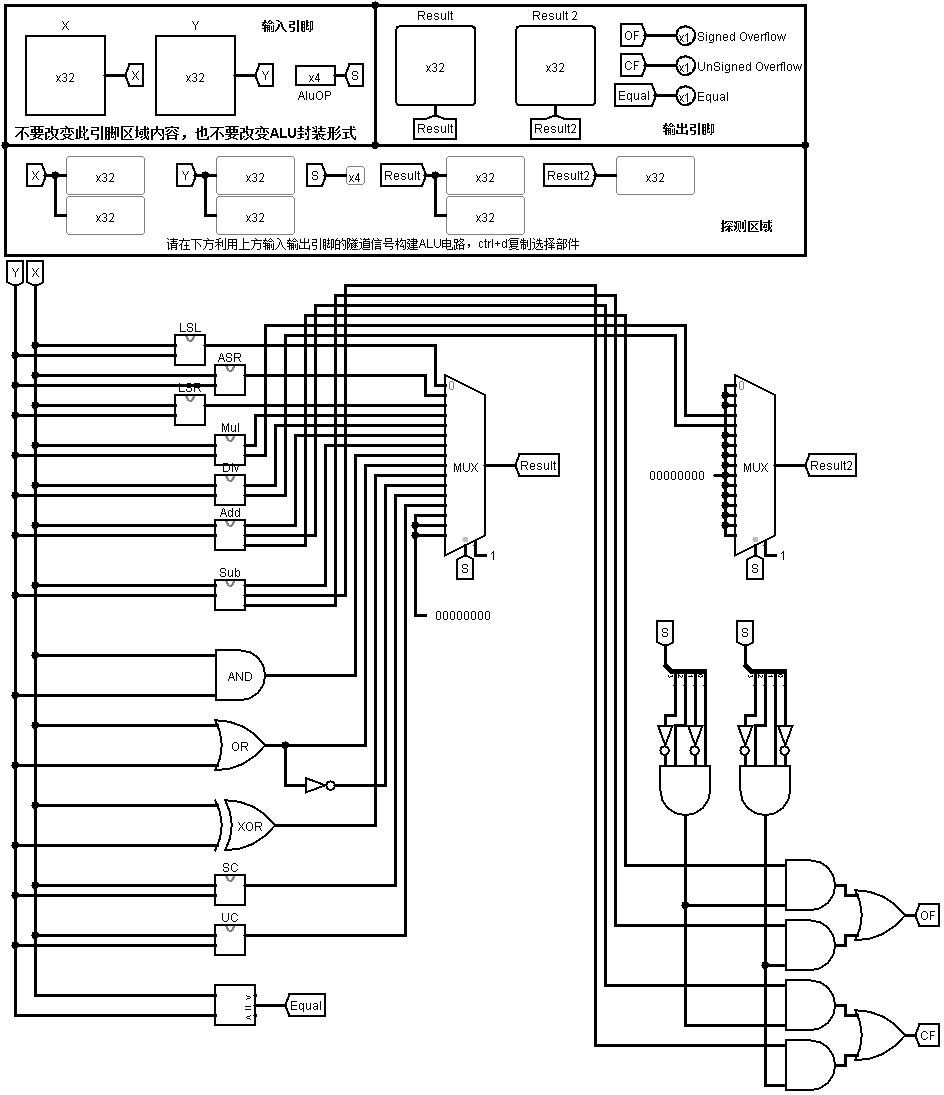


图 4‑4 运算器ALU电路图

**具体实现**： RAM的输入地址端接入的是经过分线器后的ALU的输出结果，由于只有LW指令和SW指令需要对RAM进行读写，而这两条指令送给RAM地址端的值都是从ALU来的。由SW指令中将要送入RAM中值，RAM的输入数据端连接在RegFile的R2输出端。MemWrite和MemRead是由Control模块进行指令解析后针对LW和SW指令给出的控点信号，用来控制RAM的读和写，由于在时钟下降沿写存，故接入时钟端的隧道为CLK的非运算后的结果，片选信号一直为高电平，清零信号一直为低电平。具体电路见图 4‑5。

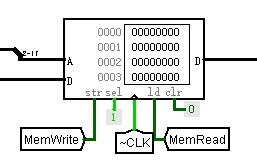


图 4‑5 数据存储器DM电路图

### 封装Control模块

**选用的元器件**： 与门，或门，非门（logisim生成与或非门的组合逻辑电路），分线器，数据选择器，隧道，输入引脚（默认参数值）。

**输入**： 指令前6位段OP（6bits），指令后6位段Funct（6bits）。

**输出**： 读寄存器1输入编号选择信号R1#（1bit），读寄存器2输入编号选择信号R2#（1bit），写寄存器编号选择信号W#（2bits），运算器功能码AluOP（4bits），写寄存器数据选择信号Din（2bits），扩展器输入数据选择信号S-EXT（2bits），运算器操作数1输入选择信号A（1bit），运算器操作数2输入选择信号B（1bit），JR指令（1bit），Branch分支信号（1bit），Jump信号（即J指令和JAL指令）（1bit），寄存器写信号RegWrite（1bit），读数据存储器信号MemRead（1bit），写数据寄存器信号MemWrite（1bit），是否为BNE指令IsBNE（1bit），是否为R型指令IsSpecial（1bit），是否为返回指令IsERET（1bit）。

**具体实现**： 该运算器要实现13种运算功能，故将每种运算功能按模块进行了封装。大部分模块都是直接调用logisim内置库中的元器件，其中加法和减法模块中，需要使用分线器将X和Y值进行最高位和其余31位的分离，之后将低31位的数相加（减）后，再将最高位进行加（减），最高位加（减）时加法器的进（借）位即低31位进行加（减）时的进（借）位输出，此时最高位进（借）位输出即为CF，CF和低31位的进（借）位进行异或即可得到OF；比较器模块的输出不为比较器直接的输出，需要使用数据选择器，其输入端为32位的数字0和1，选择信号为比较器小于输出。封装之后的具体电路见图 4‑4。

详细描述最终的微指令格式

……（根据自己的实际设计情况进行撰写）

微指令各位的功能如表 4.2所示。

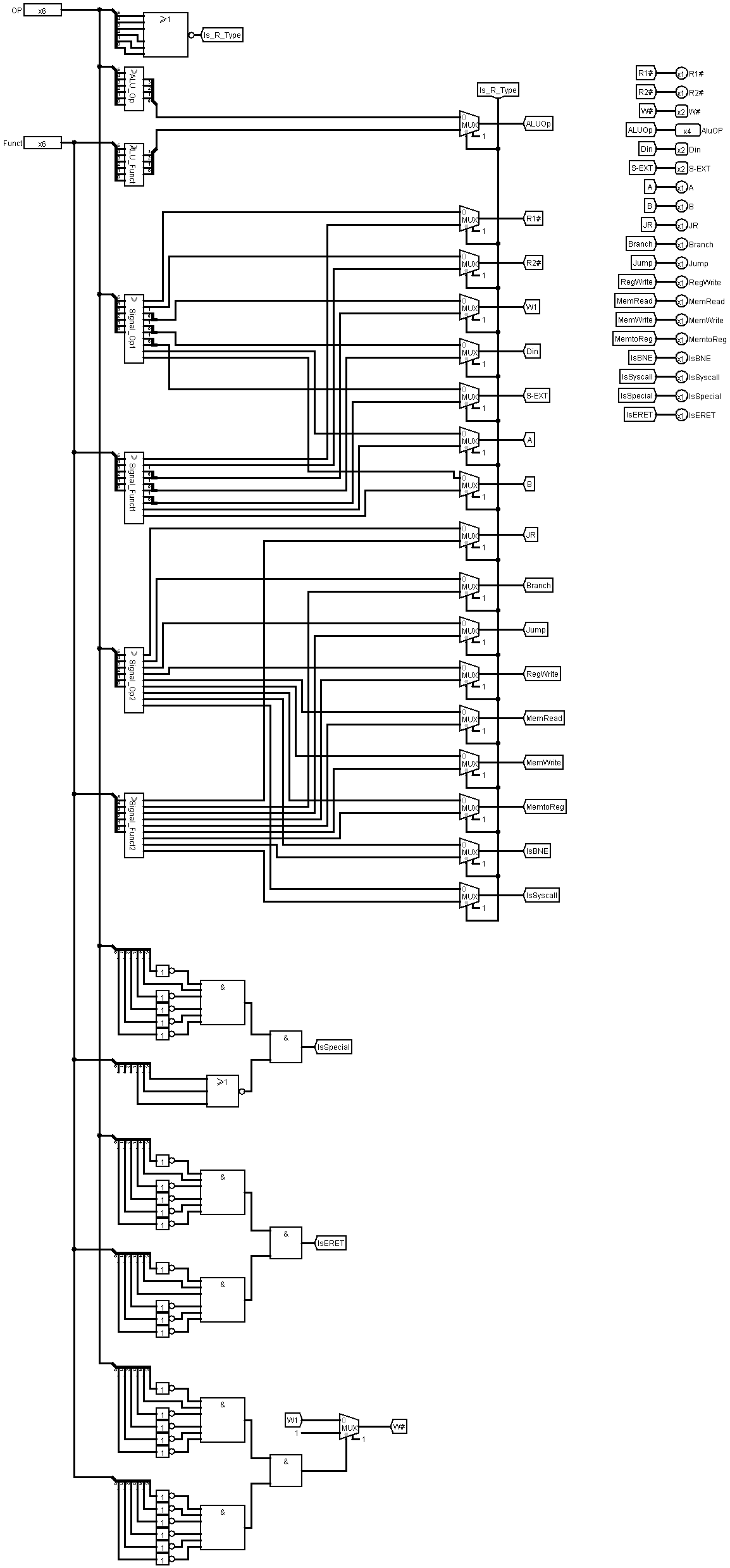
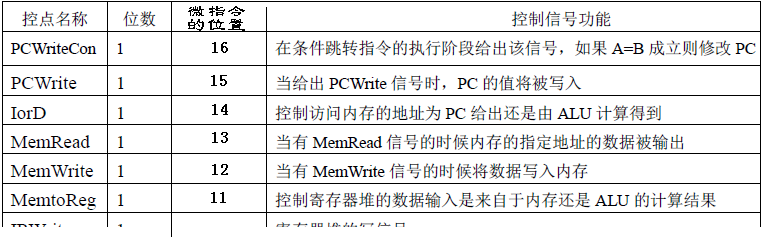


图 4‑6

表 . 微指令各位的功能描述



### 构造XX指令

所有机器指令对应的微程序，微指令集的存放和读取，下一条微指令的执行过程等......根据自己不同的实现方案具体说明）

ypedef struct SmartDiskInfo\_s{

unsigned char attributename[64]; //属性名字

unsigned char current; //当前值

unsigned char worst; //最大出错值

unsigned char threshold; //阈值

unsigned char rawstring[64];

unsigned char status[12]; //状态描述

}SmartDiskInfo;

最终测试程序以及说明：（根据自己不同的实现方案具体说明）代码如下

typedef struct SmartDiskInfo\_s{

unsigned char attributename[64]; //属性名字

unsigned char current; //当前值

unsigned char worst; //最大出错值

unsigned char threshold; //阈值

unsigned char rawstring[64];

unsigned char status[12]; //状态描述

}SmartDiskInfo;

## 可支持理想流水线多周期CPU

## 流水线冲突检测器

## 插入气泡的流水冲突处理

。。。

# 实验过程与调试

## 测试用例和功能测试

### 测试用例1

## 可自行安排章节

## 性能分析

分析不同方案时钟周期数差异

## 主要故障与调试

### 故障1

单周期： Control模块不能正确解析指令。

**故障现象：**Control模块输出时的信号不能按照事先分析得到的。

**原因分析：**用表达式生成控制电路的时候，输入的顺序是按照opcode的从低位到高位编址的，而在输入接入的opcode的时候，分线器按照从高位到低位编址的，导致由于opcode的接入错误产生指令解析错误。

**解决方案：**将opcode的分线器改为从低位到高位编址即可（如图 5.4.1）。

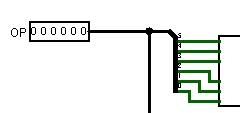


图 5.4.‑ 修改后opcode的分线器

### 故障2

单周期： jal指令不能正确执行。

**故障现象：**执行到jal指令的时候虽然正常跳转，但是没有把PC+4的值保存到RegFile的$ra（$31）寄存器中，导致后面若使用jr指令的时候不能正确将$ra寄存器的值送给PC的后续错误。

**原因分析：**检测发现最初编写的Control signal表中，误将jal的W#和Din的控点信号写到了jr指令的W#和Din中了，这样既使jal指令不能正确执行，也使jr指令不能正确执行，还有可能引起其他后续错误，这是起初设计控制信号时的失误。



图 5.4.5‑2.1 错误的jal和jr指令控点信号

**解决方案：**修改相应控点值后成功解决。



图 5.4.5‑3.2 修改后的jal和jr指令控点信号

### 故障3

单周期： 测试程序benchmark不能正确执行。

**故障现象：**运行结束时的指令计数器的值比正确值少1。

**原因分析：**没有将最后的halt指令计入（如图 5.4.3.1）。最初将clk的上升沿个数作为执行的指令数，但执行halt指令时clk不可能出现上升沿，故使得计数结果少1。

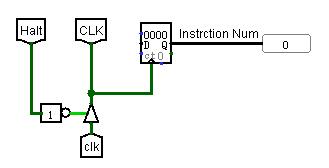


图 5.4.3.1 修改前的指令计数器

**解决方案：**修改显示结果部分（如图 5.4.3.2），执行halt指令时Halt为1，用Halt作为一个选择器的选择信号，选择计数器加一后的数值作为显示即可获得正确的计数。

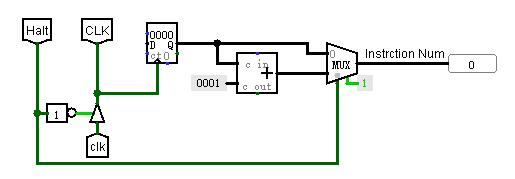


图 5.4.3.2 修改后的指令计数器

### 故障4

单周期： R型指令统计计数器不能正常显示。

**故障现象：**当程序结束的时候，R型指令统计计数器仍然继续计数，没有停止。

**原因分析：**在程序结束的时候，由于时钟端一直连接在计数器上，没有断开，导致计数器仍然在计数，由于运行的指令的最后一条是R型指令，所以只有R型指令统计计数器仍在计数。

**解决方案：**在计数器的计数端与上Halt的非信号，当程序终止时，Halt的非信号为0，使技术端为0，从而使计数器停止计数。

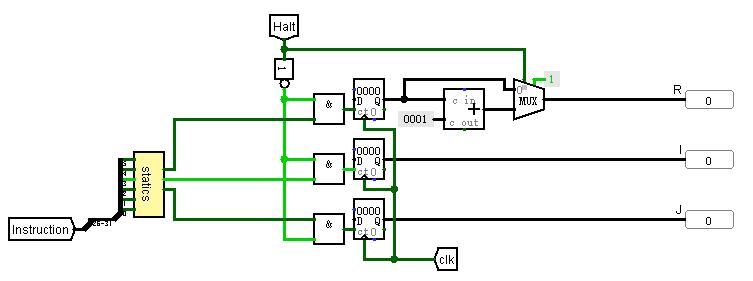


图 5.4.4 修改后的指令计数器

### 故障5

单周期： 中断处理不够合理。

**故障现象：**主程序结束后时钟仍能够相应中断信号并开始中断。

**原因分析：**在按下中断信号按钮时，会使PC值立刻发生改变（中断硬件实现的），从而即使主程序已经执行完，时钟已经停止，PC值的改变会导致Halt信号改变，导致程序继续执行。

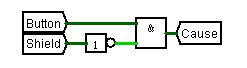


图 5.4.5.1 出现问题的中断响应信号产生器

**解决方案：**将Halt的非信号和中断信号按钮进行与操作，从而使得在时钟结束的时候（此时Halt为1），中断信号按钮失效。

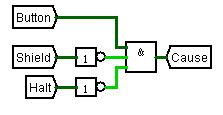


图 5.4.5.2 修改后的中断响应信号产生器

### 故障6

理想流水线： Halt控制信号问题。

**故障现象：**程序执行完统计的指令条数不正确。

**原因分析：**理想流水线中的停机（Halt）指令需要在最后WB阶段执行，结果设计时没有将Halt信号向后传递，造成在执行阶段EX执行，结果最后两条执行没有执行。

**解决方案：**在每一个阶段都将Halt信号向后传递，直到最后WB阶段执行Halt停机。

### 故障7

插气泡流水线：插入气泡的清零问题。

**故障现象：**在IF/ID阶段、ID/EX阶段和EX/MEM阶段的插入气泡的清零操作会使流水线中的指令发生缺失现象。

**原因分析：**在这几个阶段（举例为IF/ID）中的清零信号为1时，采用的是寄存器组件自带的异步清零信号，当清零信号为1是，异步将寄存器的值清零，这样就会导致插入气泡的时候会将要传递的指令信号清空，从而导致指令缺失的现象。

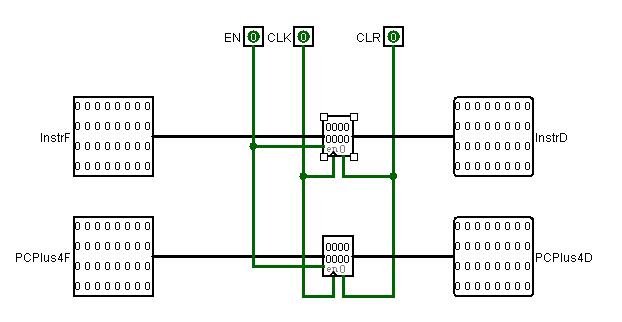


图 5.4.7.1 出现清零问题的IF/ID模块

**解决方案：**为了使寄存器的清零后不影响后面指令的传递，将清零改为和时钟同步的清零信号即可。具体修改方法为：将CLR清零信号当做数据选择器的选择信号，数据选择的数据为原本的输入和相同位数的0，当CLK上升沿来临时，就会把数据选择器的输出结果写入寄存器，此时如果CLR信号为1，则送入寄存器的值就为0，达到了同步清零的目的。

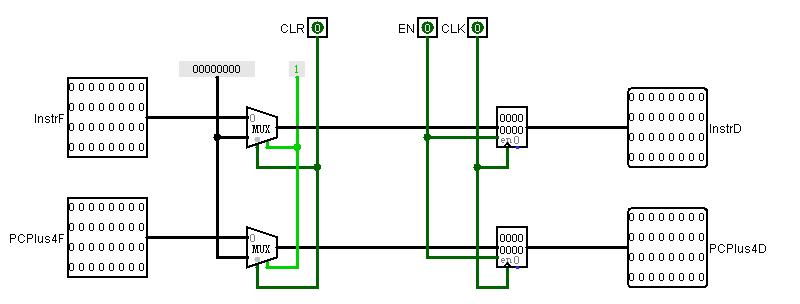


图 5.4.7.2 修改后的IF/ID模块

### 故障8

重定向流水线：访存阶段时RAM读出的结果不能正确重定向。

**故障现象：**在访存阶段，信号为LW指令时，重定向到译码阶段的值应该为从RAM读出的结果，但是实际上送回的值为访存阶段的ALU的结果。

**原因分析：**在Hazard模块中对重定向的数据选择器的选择信号解析的时候使用了优先编码器，而本来当MemReadM信号为1时，MemReadM的信号的优先级应该高于ALU结果的优先级，然而如下图，连接的时候经过优先编码器后ALU的优先级高于MemReadM信号的优先级，导致在访存阶段时LW指令送回的数据不是RAM读出的结果，而是ALU的结果。

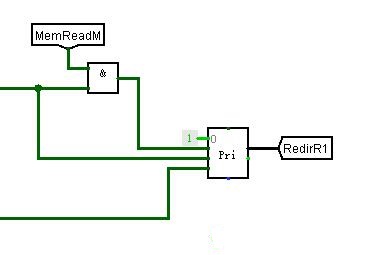


图 5.4.8.1 出现重定向错误问题的优先编码模块

**解决方案：**为了使经过优先编码器后MemReadM信号的优先级高于ALU信号，则调换接入优先编码器的MemReadM信号和ALU信号。

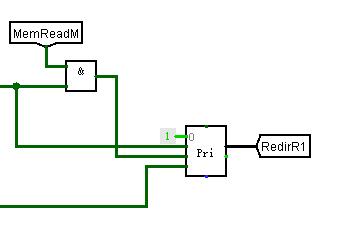


图 5.4.7.2 修改后的优先编码模块

## 实验流程

数据通路验收时间点

手动控制（单步）验收时间点

自动控制验收时间点

程序运行验收时间点

扩展功能验收时间点

表 . 课程设计进度表

|  |  |
| --- | --- |
| 时间 | 进度 |
| 第一天 |  |
| 第二天 |  |
| 第三天 |  |
| 第三天 |  |
| 第四天 |  |
| 第五天 |  |
| 第六天 |  |
| 第七天 |  |
| 第八天 |  |
| 第九天 |  |
| 第十天 |  |

# 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. 完成方案总结（自行修订扩充）（自行修订扩充）（自行修订扩充）（自行修订扩充）（自行修订扩充）。
2. 功能总结（自行修订扩充）（自行修订扩充）（自行修订扩充）（自行修订扩充）（自行修订扩充）。
3. 其他需要总结的内容，（自行修订扩充）。

## 课设心得

1. 自行修订扩充，自行修订扩充自行修订扩充自行修订扩充自行修订扩充自行修订扩充自行修订扩充自行修订扩充收OSD命令并进行处理。
2. 自行修订扩充自行修订扩充自行修订扩充自行修订扩充自行修订扩充。

# 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. 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.
5. 张晨曦，王志英. 计算机系统结构. 高等教育出版社，2008年.
6. 张志刚，FPGA 与SOPC 设计教程-DE2 实践. 西安：电子科技大学出版社，2007

参考文献属于论文的辅文部分。为了反映文稿的科学依据和作者尊重他人研究成果的严肃态度以及向读者提供有关信息的出处，论文中应列出参考文献。所列参考文献一般只限于作者亲自阅读过且发表在公开出版物上的文献，非此类文献一般不能作为参考文献，慎用互联网上的文献。

列示参考文献时，要求著录项目齐全，格式规范，其要点如下：

(1) 允许列入公开出版的图书、期刊的文章、专利、硕士和博士论文、科技报告等。未公开发表的文章和非正式出版物等请勿列入。

(2) 文献的著录项目及其次序，有以下几种情况：

A 图书：[序号] 作者.书名.版本(第×版).译者.出版地:出版者,出版年：起页-止页

B 期刊：[序号] 作者.文章名称.期刊名称,年号,卷号(期号)：起页-止页

C 会议论文集：[序号] 作者.文章名称. In(见):整本文集的编者姓名ed.(多编者用eds.).文集名.会址.开会年.出版地：出版者,出版年：起页-止页

D 专利：[序号] 专利申请者.专利题名.专利国别,专利文献种类,专利号,出版年：起页-止页

E 学位论文：[序号] 作者.题名:[博士或硕士学位论文].保存地点:保存单位(如华中科技大学),年份.

F 网页: [序号] URL: 网络地址，如URL:http://www.cbs.dtu.dk/services

(3) 所有参考文献均应在正文中予以引用，引用方式分以下两种情况：

A在正文中附注参考文献时，把所有文献的号码连方括号一并放在加注处的右上角，例如：“TCP/IP[3-6,9]是……”；

B所提及的文献作为叙述文中的直接说明语时，则其编号连方括号应与正文并排，例如：“…见文献[2,3-6]”。

方括号内可为单个文献的编号，如[2]；也可为若干参考文献编号的罗列，如[2,6,9]；也可为用“x-y”表示的序号区间，如[3-6]；或以上形式的组合，如[2,3-6]。

(4) 其它注意事项：

A 序号编制顺序：参考文献的序号依据引用先后编制，即第一篇被引用的参考文献编号为1，第二篇被引用的参考文献编号为2，依此类推；

B 文献若属第1版，则参考文献著录项目“版次”一项可略；

C 若为多作者的文献，则作者间用逗点“,”隔开。作者超过三个时，只著录前3个，其后加“等”(英文用“et al ”)。外文作者采用姓在前、名在后的书写形式。外国人的名可以缩写为首字母，缩写的名后不加圆点“.”；

D 无出版地者要注明“[出版地不详]”或与之相应文字（英文用[S.L.]），无出版者要注明“出版者不详”或与之相应文字（英文用[S.N.]）；

E 网页作为参考文献，除非必要，尽量少用。

F参考文献引用时标注位置不能在章、节的标题上，如“2.3 软件测试方法概述[1,3-6]”为不正确的标注，必须引用在正文的文字段落中；同时标注位置要在句末的标点符号以内，不能先写标点符号，再标参考文献，如“……网络协议TCP/IP等。[3-6,9]” 亦为不正确的标注。