***2019***



**计算机组成原理 ·实验报告·**

j0242087[1]

|  |  |
| --- | --- |
| 专 业： | 计算机科学与技术 |
| 班 级： | 计卓1701 |
| 学 号： | U201714468 |
| 姓 名： | 李晓宇 |
| 电 话： | 15271939616 |
| 邮 件： | [1960588640@qq.com](mailto:13456@qq.com) |
| 完成日期： | 2019-12-19 |



目 录

[1](#_Toc499846024) **[CPU实验](#_Toc499846024)** [3](#_Toc499846024)

[1.1 设计要求 3](#_Toc499846025)

[1.2 方案设计 4](#_Toc499846026)

[1.3 实验步骤 1](#_Toc499846027)9

[1.4 故障与调试 2](#_Toc499846028)0

[1.5 测试与分析 2](#_Toc499846029)0

[2 总结与心得 2](#_Toc499846048)2

[2.1 实验总结 2](#_Toc499846049)2

[2.2 实验心得 2](#_Toc499846050)2

[参考文献 2](#_Toc499846051)3

# CPU设计实验

## 设计要求

利用logisim平台中现有运算部件构建单周期MIPS硬布线CPU、多周期MIPS微程序和硬布线CPU，支持8条核心指令，并能够运行基本的冒泡排序程序。

8条核心指令集及其功能描述和控制信号描述如表 1-1和表 1-2所示：

|  |  |
| --- | --- |
| 指令 | 功能描述 |
| add $rd,$rs,$rt | R[$rd]←R[$rs]+R[$rt] 溢出时产生异常，且不修改R[$rd] |
| slt $rd,$rs,$rt | R[$rd]←R[$rs]<R[$rt] 小于置1，有符号比较 |
| addi $rt,$rs,imm | R[$rt]←R[$rs]+SignExt16b(imm) 溢出产生异常 |
| lw $rt,imm($rs) | R[$rt]←Mem4B(R[$rs]+SignExt16b(imm)) |
| sw $rt,imm($rs) | Mem4B(R[$rs]+SignExt16b(imm))←R[$rt] |
| beq $rs,$rt,imm | if(R[$rs] = R[$rt])  PC ← PC + SignExt18b({imm, 00}) |
| bne $rs,$rt,imm | if(R[$rs] != R[$rt]) PC ← PC + SignExt18b({imm, 00}) |
| syscall | 系统调用，这里用于停机 |

表 1-1 核心功能及功能描述

|  |  |  |
| --- | --- | --- |
| 控制信号 | 信号说明 | 产生条件 |
| MemToReg | 写入寄存器的数据来自存储器 | lw指令 |
| MemWrite | 写内存控制信号 | sw指令 未单独设置MemRead信号 |
| Beq | Beq指令译码信号 | Beq指令 |
| Bne | Bne指令译码信号 | Bne指令 |
| AluOP | 运算器操作控制符 | 加法，比较两种运算 |
| AluSrcB | 运算器第二输入选择 | Lw指令，sw指令，addi |
| RegWrite | 寄存器写使能控制信号 | 寄存器写回信号 |
| RegDst | 写入寄存器选择控制信号 | R型指令 |
| Halt | 停机信号，取反后控制PC使能端 | syscall指令 |

表 1-2 控制器控制信号

## 方案设计

### 单周期MIPS硬布线CPU设计

（1）整体数据通路

单周期MIPS CPU采用定长指令周期，所有指令均在一个时钟周期内完成，因此性能取决于运行最慢的指令。本实验中单周期内实现所有指令，因此不设置AR、DR、IR等寄存器，同时采用程序和数据分开存放的哈佛结构。下面提供各种指令所需要的数据通路。

所有要求实现的指令可分为三种格式：R型指令、I型指令和J型指令。指令的具体格式如下图 1-1 三种指令格式所示。

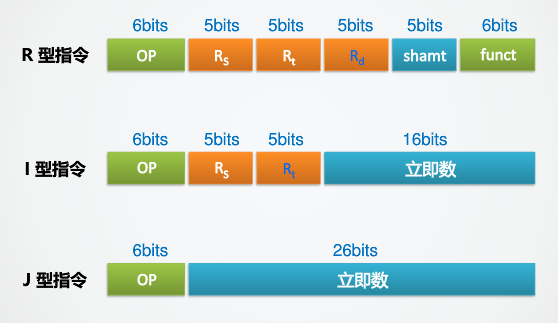


图 1-1 三种指令格式

取指令数据通路如图 1-2 取指令数据通路红色部分所示：

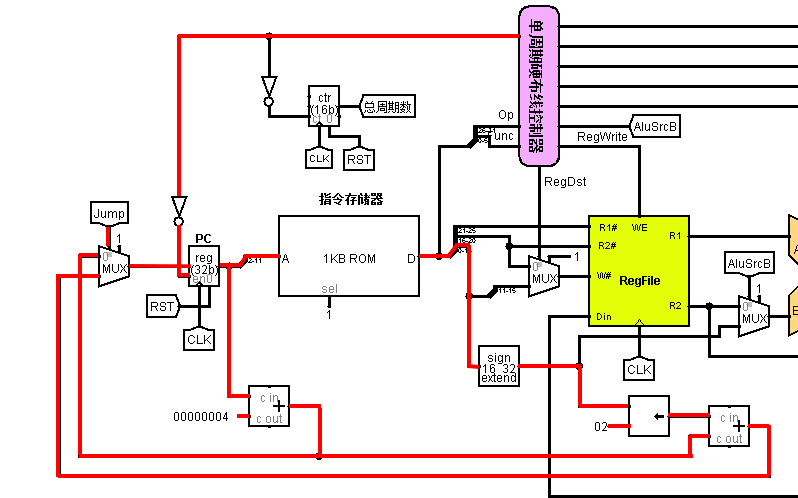


图 1-2 取指令数据通路

取指令主要涉及到对不同操作PC变换的处理。在我们所涉及到的指令中，PC的变换只有两种形式，一种是不涉及到更改PC值的指令，只需将PC+4，使其指向内存中下一条指令的地址即可；另一种是涉及到更改当前PC的跳转指令，这类指令属于I型指令，因此取出当前指令的后16位作为立即数，左移两位与PC+4相加得到跳转地址。两种地址通过二路选择器利用Jump信号进行选择，从而在时钟到来时写入PC寄存器，完成取指令操作。Jump信号是用来判断当前指令是否需要跳转地址，只有Bne和Beq指令需要此功能。相关求解Jump信号的数据通路见图 1-3 Jump信号数据通路.

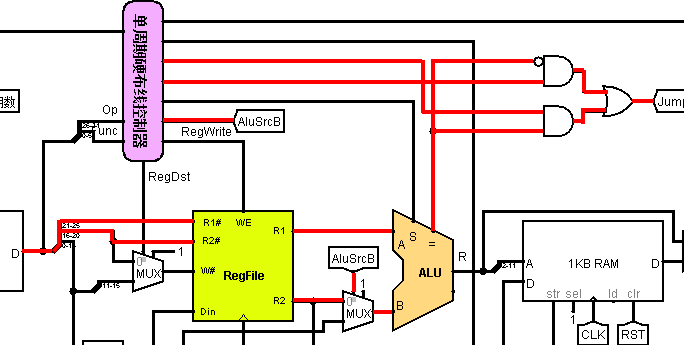


图 1-3 Jump信号数据通路

因为Bne和Beq指令均是I型指令，因此取出指令的两个操作数R1和R2编号，送入寄存器中取出对应寄存器的值，送入ALU中判断是否相等。将相等信号和控制器提供的Bne和Beq信号放在一起通过逻辑门最终生成Jump信号。

下图 1-4 寄存器和ALU数据通路：

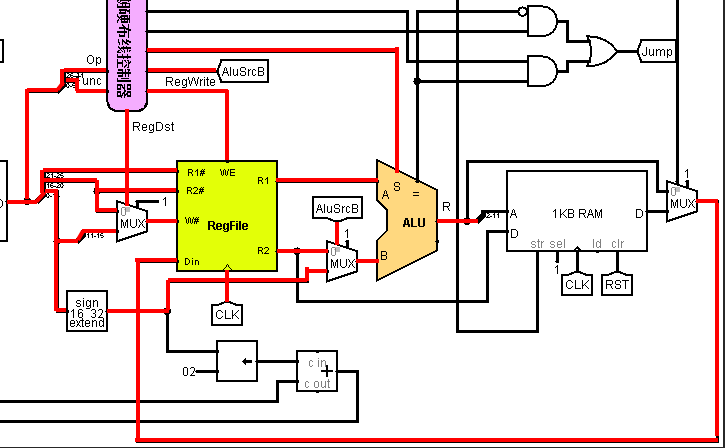


图 1-4 寄存器和ALU数据通路

所有用到R1寄存器和R2寄存器的指令序号所在位置是固定的，因此可以直接将指令的16到20位送入R2寄存器，21到25位送入R1寄存器取出寄存器的值。而需要写入寄存器值的指令有LW指令和R型指令。其中LW指令需要将内存中的数据读到R2寄存器中，因此选择16到20位送入W#接口，而R型指令需要写入R3，所有选择11到15位送入W#接口，通过控制器的RegDst信号进行数据的选择，RegWrite信号判断数据是否要写入寄存器。Din接口放入写入的数据，在后面的通路中加以定义。

对ALU的操作通过控制器的AluOP信号进行控制，Alu的操作类型也有两种，一种是寄存器R1与立即数操作，如LW指令。另一种是寄存器R1与寄存器R2操作，如R型指令。因此通过二路选择器和控制器AluSrc信号控制Alu第二个操作数的输入即可。

最后图 1-5 数据存储器数据通路。

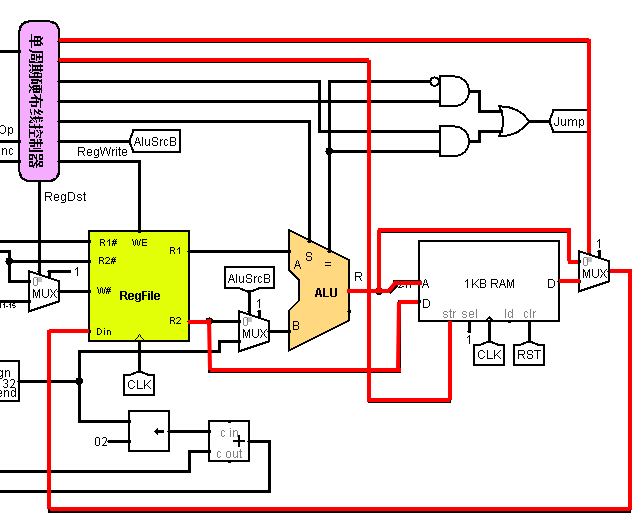


图 1-5 数据存储器数据通路

数据写入存储器只可能是R2寄存器的内容，因此直接将R2寄存器取出的值送入到D接口即可。而地址则是将Alu计算的结果送入A接口即可。存储器是否更新通过控制器的MemWrite信号判断。最后写回寄存器的值有两种可能，分别是R型指令对R1寄存器和R2寄存器操作的结果和LW型指令从存储器中读取的值，因此这两个值通过二路选择器和控制器的MemToReg信号选择后送回寄存器的Din接口。

单周期MIPS CPU整体的数据通路如图 1-6 单周期MIPS CPU整体数据通路：

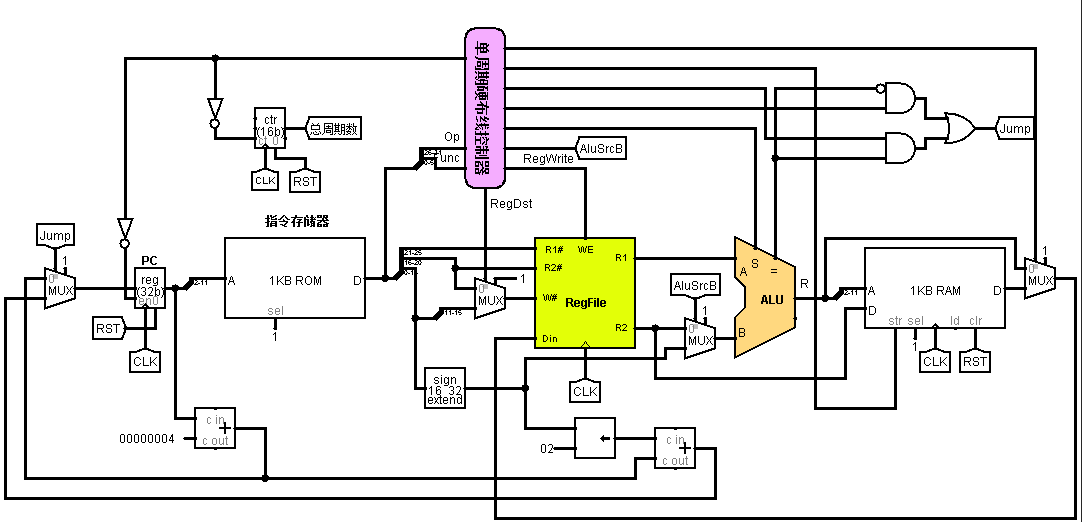


图 1-6 单周期MIPS CPU整体数据通路

（2）单周期硬布线控制器

指令译码信号见表 1-3

|  |  |  |
| --- | --- | --- |
| 指令译码信号 | OP | FUNC |
| SLT | 0 | 2AH |
| ADD | 0 | 20H |
| SysCall | 0 | 0CH |
| R\_TYPE | 0 | 2AH || 20H |
| LW | 23H | 无 |
| SW | 2BH | 无 |
| BEQ | 04H | 无 |
| BNE | 05H | 无 |
| ADDI | 08H | 无 |

表 1-3 指令译码信号解析

因此可以得到相应的指令译码逻辑如图 1-7 指令译码逻辑

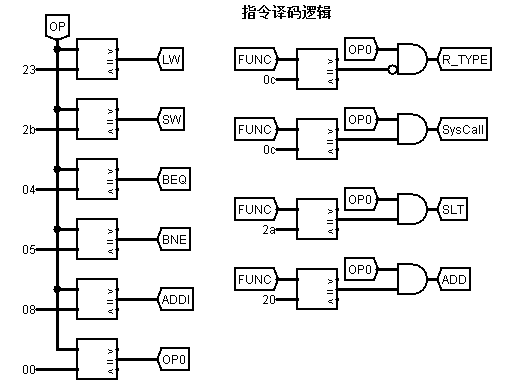


图 1-7 指令译码逻辑

下图 1-8 ALU逻辑表示ALU控制逻辑，除了SLT指令以外，ALU\_OP均是5，表示加法操作。而SLT指令时ALU\_OP为b，表示有符号比较。

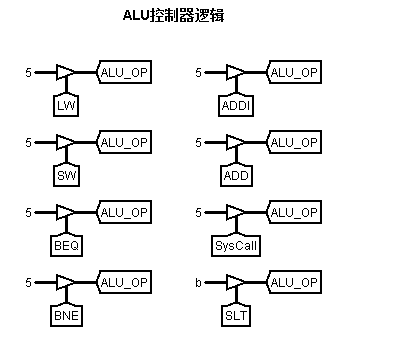


图 1-8 ALU逻辑

控制信号逻辑如图 1-9 控制信号逻辑所示。只有R型指令需要选择R3寄存器，所以赋值给RegDst信号。只有ADDI、R型指令和LW指令需要写入寄存器，因此赋值RegWrite。ADDI、LW和SW指令均需要立即数操作，因此赋值ALUSRC。存储器写入MemWrite对应LW指令，MemToReg对应SW指令。剩下的Beq、Bne和SysCall指令分别输出相应的信号即可。

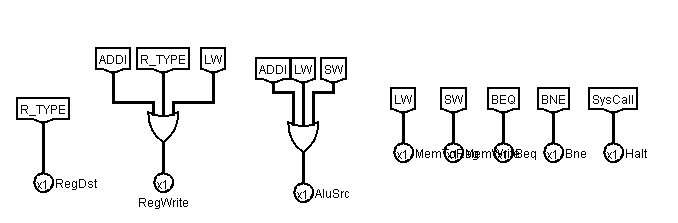


图 1-9 控制信号逻辑

### 多周期MIPS微程序CPU设计

1. 整体数据通路

多周期MIPS CPU设计不再区分指令和数据存储器，分时使用功能部件。同时采用变长指令周期，使得整体时钟周期变小，传输通路变短。为了适应变长指令周期，功能部件输出端增加寄存器索存数据。

多周期MIPS取指令阶段T1和T2数据通路如图 1-10和图 1-11所示

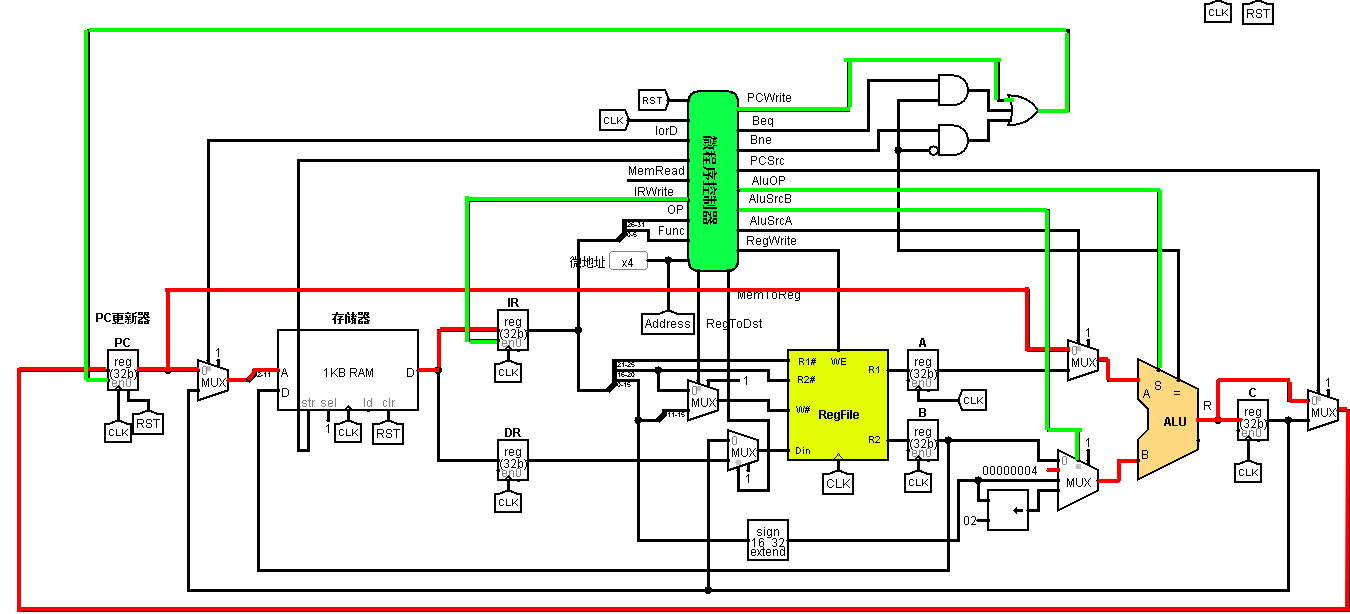


图 1-10 多周期取指令阶段T1数据通路

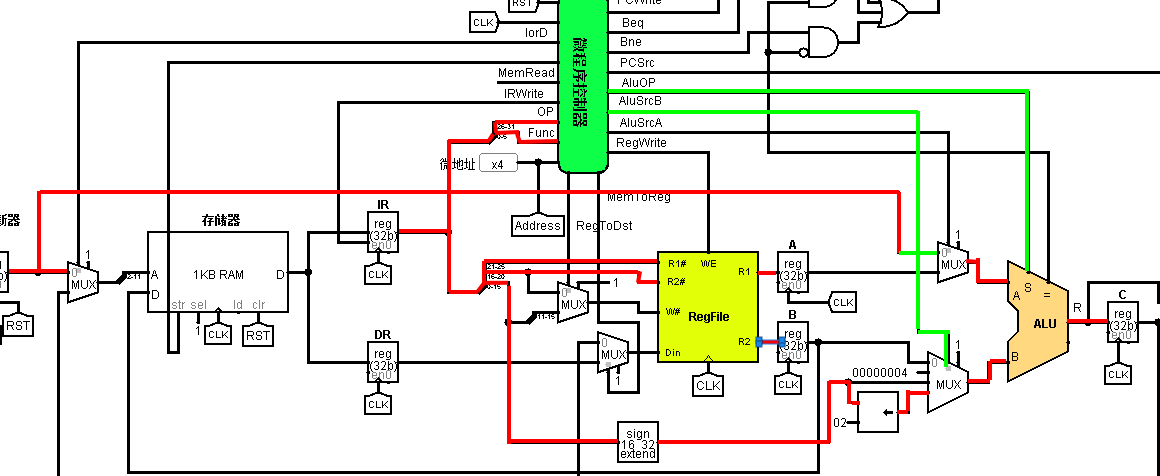


图 1-11 多周期取指令阶段T2数据通路

取指令T1时，用PC从存储器中读出下一条指令。将PC+4的值送回PC寄存器中。取指令T2时，将T1阶段取出的指令进行译码，并利用相应字段取出寄存器R1和R2的值送入寄存器中进行保存。同时取出立即数和当前PC寄存器中的值相加，即和PC+4相加得到跳转地址，为后续执行Bne和Beq指令提供准备。

下图 1-12是R型指令执行状态周期T3和T4。其中绿色线代表T3，红色线代表T4

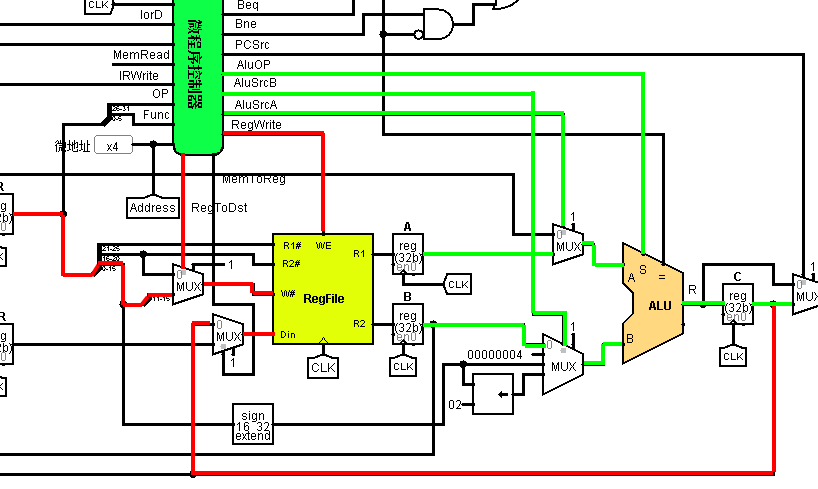


图 1-12 R型指令执行状态T3和T4

R型指令T3时寄存器R1和R2的值进行对应ALU运算并将结果送往C寄存器，在时钟到来后锁存在寄存器内部。T4时，将寄存器C中保存的计算结果送往Din接口，并且提供R3寄存器的序号和寄存器写入信号RegWrite进行相应的寄存器写入操作，将运算出的结果保存到寄存器中。

下图 1-13是LW指令执行状态周期T3、T4和T5。其中绿色代表T3操作，红色代表T4操作，紫色代表T5操作。

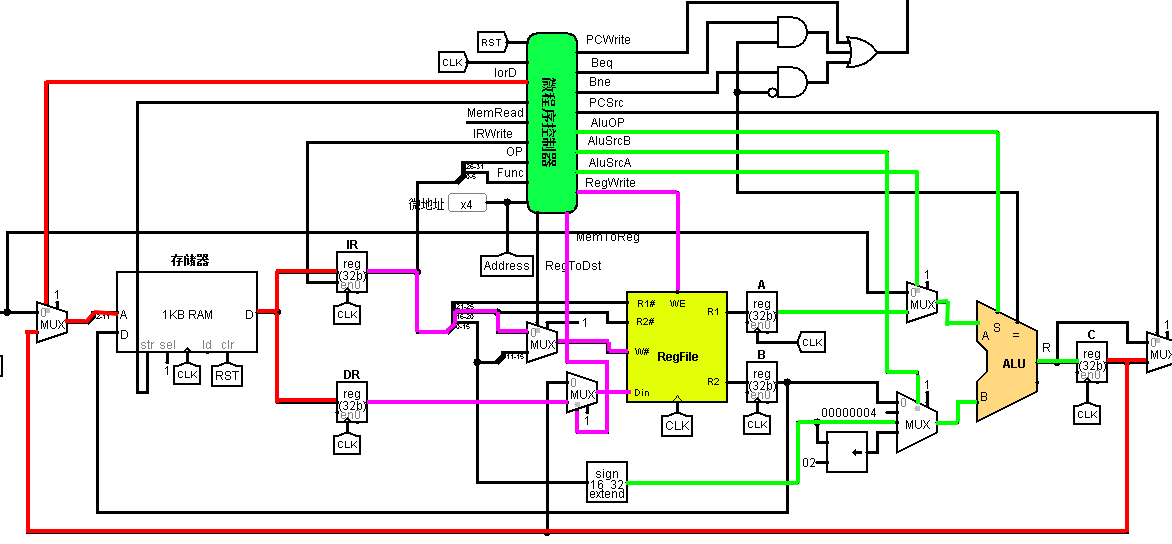


图 1-13 LW指令执行状态周期T3、T4、T5

LW指令T3阶段通过将寄存器A中存放的R1寄存器值和立即数的值相加计算出数据存放地址并存放在C寄存器中。T4阶段将C寄存器的值传入存储器的A接口取出对应地址的数据并送到DR寄存器中。利用IRWrite信号控制数据不会存入IR寄存器中。最后的T5阶段将指令的R2寄存器序号传入W#接口，将数据寄存器存放的数据传入WD接口，时钟到来后完成对寄存器的写入操作。SW指令同LW指令的执行状态大体相同，这里不做过多的赘述。

下图 1-14是Beq和Bne指令执行状态周期T3

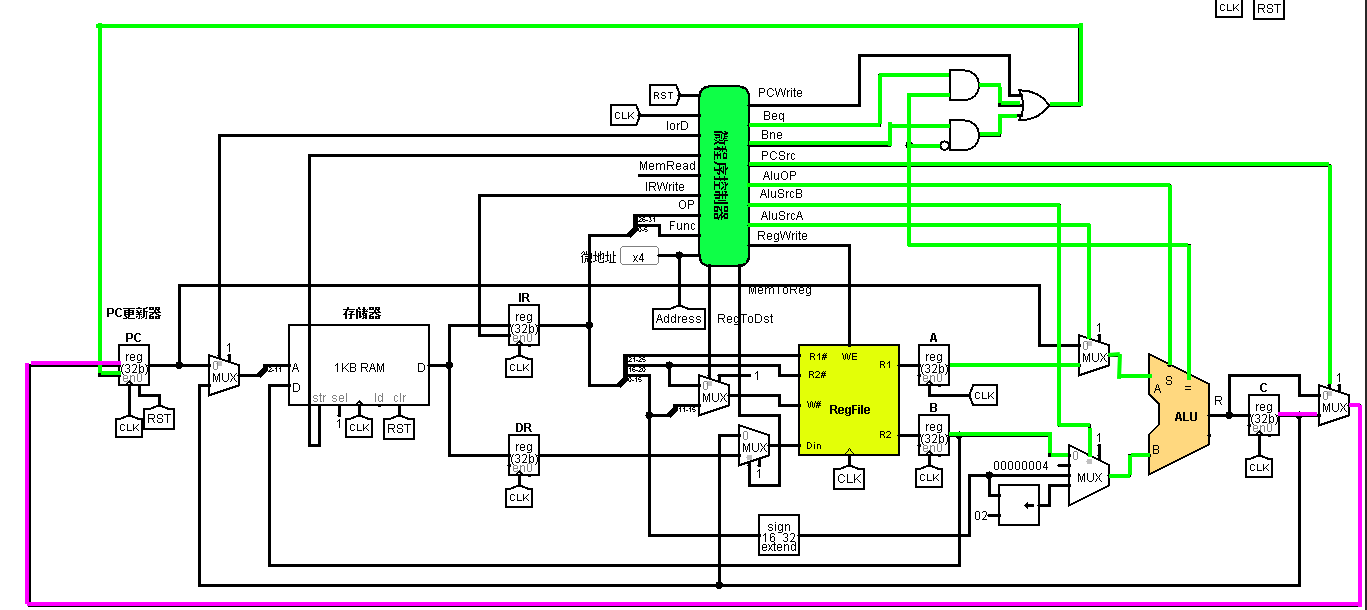


图 1-14 Beq和Bne指令执行状态周期T3

Beq和Bne指令T3阶段时操作分为并行的两个部分。第一个部分通过ALU比较R1寄存器和R2寄存器的值产生相等型号Equal，并结合Bne和Beq指令生成Jump型号来判断是否需要进行跳转。第二个部分通过将T2阶段计算到的跳转地址传入PC寄存器的写入接口，等待时钟到来时根据Jump指令判断是否刷新PC寄存器的值。

下图 1-15为多周期状态转换图，图 1-16为多周期MIPS CPU的整体数据通路

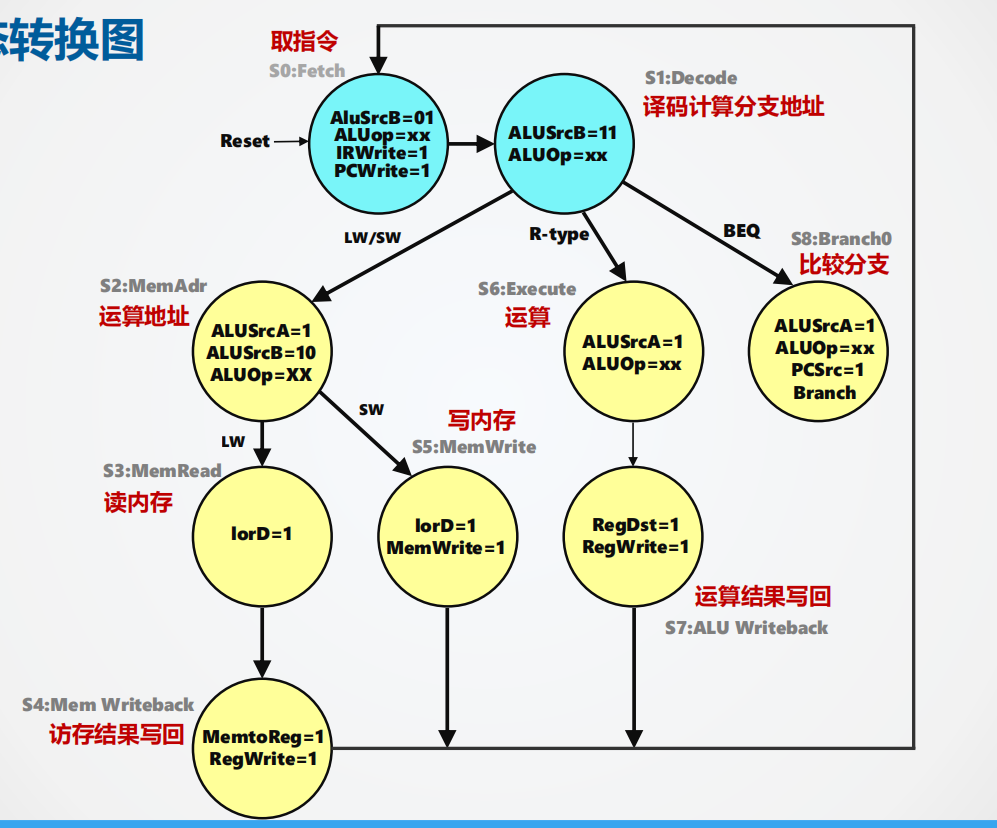


图 1-15 多周期状态转换图

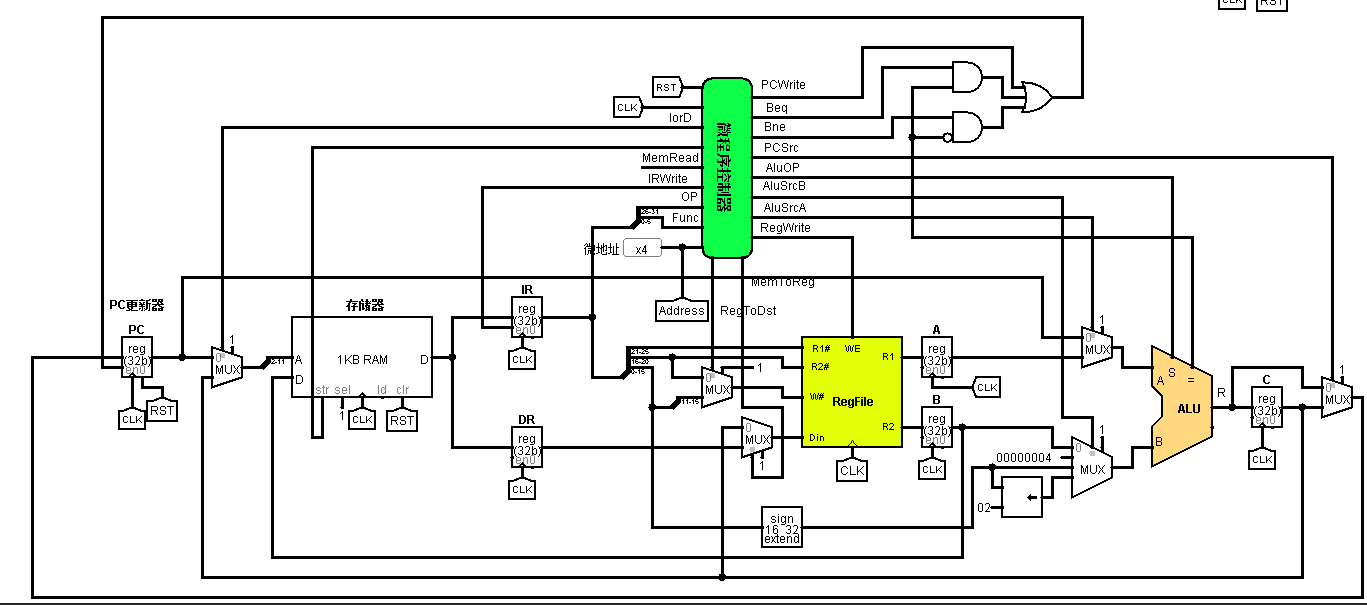


图 1-16 多周期MIPS CPU整体数据通路

1. 指令译码器逻辑和ALU逻辑

同单周期一致，如图1.7，ALU逻辑如图 1-17，在单周期的ALU基础上添加了通过ALU\_Control信号来选择对应的ALU操作码的功能。

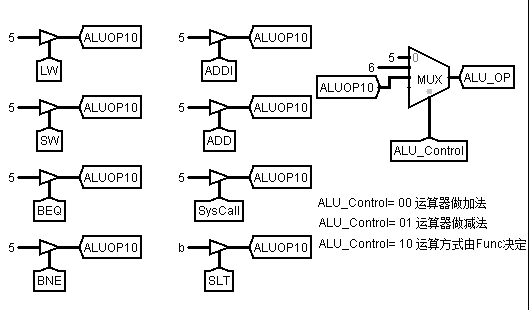


图 1-17 多周期MIPS ALU设计

1. 微程序地址转移逻辑

|  |  |  |
| --- | --- | --- |
| 微指令 | 状态 | 地址 |
| 取指令 | S0 | 0 |
| 译码 | S1 | 1 |
| LW1 | S2 | 2 |
| LW2 | S3 | 3 |
| LW3 | S4 | 4 |
| SW1 | S5 | 5 |
| SW2 | S6 | 6 |
| R型运算 | S7 | 7 |
| R型运算 | S8 | 8 |
| Beq | S9 | 9 |
| Bne | S10 | 10 |
| ADDI1 | S11 | 11 |
| ADDI2 | S12 | 12 |
| SYSCALL | S13 | 13 |

表 1-4 微指令状态及地址分配

以指令译码信号为输入，根据该表得到微程序入口地址真值表如图 1-18所示，生成的表达式为

S3= ADDI + BEQ + BNE + SYSCALL

S2= R\_Type + SW + SYSCALL

S1= R\_Type + ADDI + LW + BNE

S0= R\_Type + ADDI + SW + BEQ + SYSCALL

对应的微程序地址转移逻辑电路如图 1-19所示

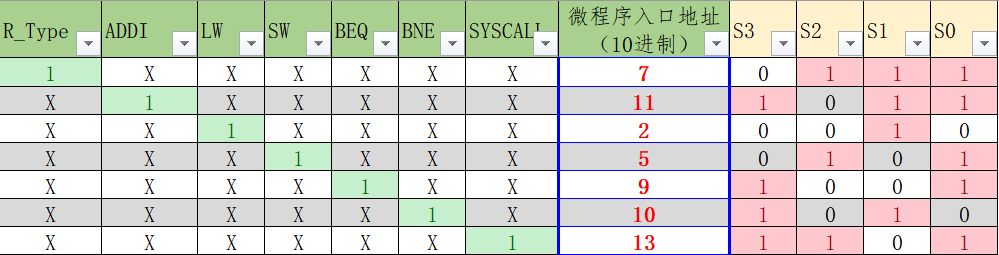


图 1-18 微程序入口地址真值表

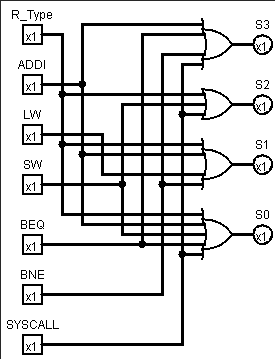


图 1-19 微程序转移地址逻辑电路

1. 微程序控制器指令载入

根据状态图设置微指令的地址为4位，根据相应的信号要求得出下列表 1-5 微指令

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 微指令功能 | 状态 | 微指令地址 | 微指令 | 十六进制 |
| 取指令 | 0 | 0000 | 000010011001000000001 | 13201 |
| 译码 | 1 | 0001 | 000110000000000010000 | 30010 |
| LW1 | 2 | 0010 | 001100000000000000011 | 60003 |
| LW2 | 3 | 0011 | 100000000001000000100 | 100204 |
| LW3 | 4 | 0100 | 000001000100000000000 | 8800 |
| SW1 | 5 | 0101 | 001100000000000000110 | 60006 |
| SW2 | 6 | 0110 | 100000000010000000000 | 100400 |
| R型运算1 | 7 | 0111 | 001000000000001001000 | 40048 |
| R型运算2 | 8 | 1000 | 000000100100000000000 | 4800 |
| Beq | 9 | 1001 | 011000000000100100000 | C0120 |
| Bne | 10 | 1010 | 011000000000010100000 | C00A0 |
| ADDI1 | 11 | 1011 | 001100000000000001100 | 6000C |
| ADDI2 | 12 | 1100 | 000000000100000000000 | 800 |
| SYSCALL | 13 | 1101 | 000000000000000001101 | D |

表 1-5 微指令

将这些微指令载入到微程序控制器中，最终完成的多周期微程序控制器电路图如图 1-20所示。

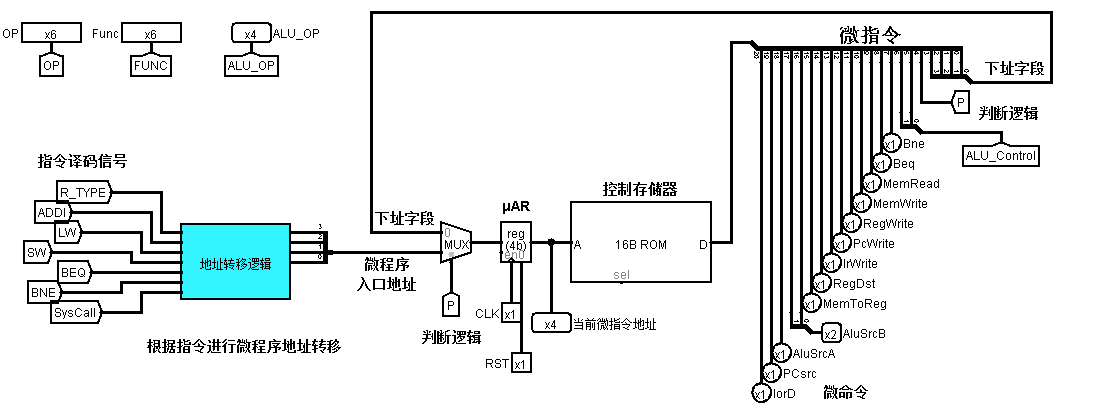


图 1-20 多周期微程序控制器电路

### 多周期MIPS硬布线CPU设计

（1）整体数据通路、指令译码逻辑和ALU控制逻辑均与上述的多周期MIPS微程序CPU相同，因此这里不再赘述。

（2）控制器状态机

根据现态和指令译码信号为输入，做出微程序次态真值表如表 1-6所示

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S3 | S2 | S1 | S0 | R\_Type | LW | SW | BEQ | BNE | SYSCALL | ADDI | N3 | N2 | N1 | N0 |
| 0 | 0 | 0 | 0 | X | X | X | X | X | X | X | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | X | X | X | X | X | X | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | X | 1 | X | X | X | X | X | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | X | X | 1 | X | X | X | X | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | X | X | X | 1 | X | X | X | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | X | X | X | X | 1 | X | X | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | X | X | X | X | X | 1 | X | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | X | X | X | X | X | X | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | X | X | X | X | X | X | X | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | X | X | X | X | X | X | X | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | X | X | X | X | X | X | X | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | X | X | X | X | X | X | X | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | X | X | X | X | X | X | X | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | X | X | X | X | X | X | X | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | X | X | X | X | X | X | X | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | X | X | X | X | X | X | X | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | X | X | X | X | X | X | X | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | X | X | X | X | X | X | X | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | X | X | X | X | X | X | X | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | X | X | X | X | X | X | X | 1 | 1 | 0 | 1 |

表 1-6 次态真值表

从而自动生成组合逻辑表达式，并导入到logisim中生成对应的组合逻辑电路，完成状态机的构建。

（3）硬布线控制器如图 1-21

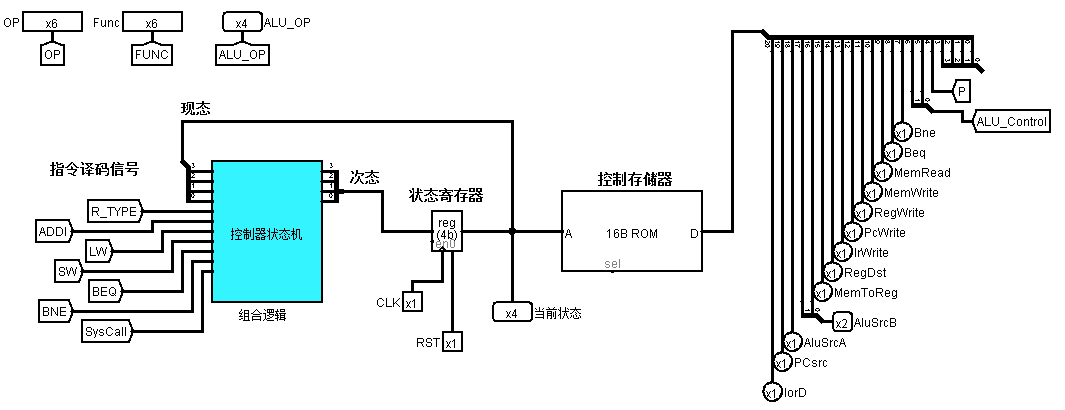


图 1-21 多周期硬布线控制器

## 实验步骤

### 单周期MIPS硬布线CPU

1. 连接单周期硬布线CPU的数据通路
2. 根据OP和FUNC的值设计指令译码逻辑和ALU控制逻辑，最终生成控制

器输出信号

1. 在指令存储器中加载冒泡程序，进行时钟模拟。当程序运行完毕后比对数据

存储器中的值与理论值是否相同，并且比较周期与参考周期是否相同从而判

断电路的正确性。

### 多周期MIPS微程序CPU

1. 连接多周期硬布线CPU的数据通路
2. 设计微程序的格式，为每一条微指令对应输出的控制信号赋值，根据状态图

设计次态真值表，从而将对应的微指令地址存储在微指令存储器中

1. 根据OP、FUNC设计指令译码和ALU控制器逻辑
2. 在指令存储器中加载冒泡程序，进行时钟模拟。当程序运行完毕后比对数据

存储器中的值与理论值是否相同，并且比较周期与参考周期是否相同从而判

断电路的正确性。

### 多周期MIPS硬布线CPU

（1）连接多周期硬布线CPU的数据通路

（2）设计微程序的状态，根据状态图设计次态真值表，从而构建状态机FSM，并

填充微指令存储器

（3）根据OP、FUNC设计指令译码和ALU控制器逻辑

（4）在指令存储器中加载冒泡程序，进行时钟模拟。当程序运行完毕后比对数据

存储器中的值与理论值是否相同，并且比较周期与参考周期是否相同从而判

断电路的正确性。

## 故障与调试

### 周期计算问题

**故障现象：**多周期MIPS微程序CPU最后运行结果正确但周期数目与参考值有差异。

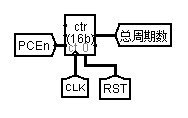


图 1-22 周期计数故障图

**原因分析：**如图 1-22，周期计数时不应该使用PCEn信号，而应该判断微程序的地址不等于D，即不是停机状态。

**解决方案：**修改成正确的计数模式。

## 测试与分析

### 单周期MIPS硬布线CPU测试

导入冒泡排序程序到指令存储器后，程序运行完的结果如图 1-23





图 1-23 单周期MIPS硬布线CPU程序运行结果

### 多周期MIPS微程序CPU测试

导入冒泡排序程序到存储器后，程序运行完的结果如图 1-24





图 1-24 多周期MIPS微程序CPU程序运行结果

### 多周期MIPS硬布线CPU测试

导入冒泡排序程序到存储器后，程序运行完的结果如图 1-25





图 1-25 多周期MIPS硬布线CPU程序运行结果

# 总结与心得

## 实验总结

本次实验主要完成了如下几点工作：

1. 完成单周期硬布线、多周期硬布线和多周期微程序MIPS CPU的设计和连接。
2. 完成单周期硬布线、多周期硬布线和多周期微程序控制器的设计和连接。
3. 在三种CPU电路上能够正确跑通所有给定的程序，并保持正确的运行周期数（包括冒泡排序程序）

## 实验心得

本次实验带我入了计算机硬件设计的门，让我从以前对计算机系统硬件实现一窍不通的状态到现在有了自己独特的理解。这次实验综合运用了以前学习过的各种数字电路知识，真正完成了从理论到实践有机结合的目标。我认为整个计算机系统的构建就是一门艺术，将最基本的逻辑门电路和寄存器设备一步步组合封装，最后可以形成如此复杂的系统，不可谓是一代又一代工程师智慧的结晶。

对于本次实验的设计我感觉非常满意，通过MOOC和上课的讲解，整个CPU的设计思路非常简洁易懂，并且组合方式清晰有条理，从而在连接数据通路时能够一步步接近目标，不会有挫败感。同时验证程序采用了冒泡排序程序，顿时有了亲切的感觉，让我们对自己设计的CPU感到自信，同时将组成原理的知识同上学期的汇编结合在了一起，真正打通了整个计算机自底向上的逻辑结构。

对于今后实验的建议：我希望以后的实验设计能够更加专注于对数据通路的构建，从而有助于加深对组成原理的理解。

# 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5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,胡迪青.计算机组成原理实践教程.北京:清华大学出版社，2018年.
4. 秦磊华，吴非，莫正坤.计算机组成原理. 北京:清华大学出版社，2011年.
5. 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:** |
| 二、对课程实验的学术评语（教师填写） |
|  |
| 三、对课程实验的评分（教师填写） |
| |  |  |  |  | | --- | --- | --- | --- | | 评分项目  （分值） | 报告撰写  （30分） | 课设过程  （70分） | 最终评定  （100分） | | 得分 |  |  |  | |
| **指导教师签字:** |