***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： |  |
| 学 号： |  |
| 姓 名： |  |
| 电 话： |  |
| 邮 件： |  |
| 完成日期： | 2019-12-21 |



目 录

[1 单周期MIPS(硬布线) 2](#_Toc27843503)

[1.1 设计要求 2](#_Toc27843504)

[1.2 方案设计 2](#_Toc27843505)

[1.3 实验步骤 6](#_Toc27843506)

[1.4 故障与调试 8](#_Toc27843507)

[1.5 测试与分析 9](#_Toc27843508)

[2 多周期MIPS(微程序) 11](#_Toc27843509)

[2.1 设计要求 11](#_Toc27843510)

[2.2 方案设计 11](#_Toc27843511)

[2.3 实验步骤 14](#_Toc27843512)

[2.4 故障与调试 15](#_Toc27843513)

[2.5 测试与分析 15](#_Toc27843514)

[3 多周期MIPS(硬布线) 16](#_Toc27843515)

[3.1 设计要求 16](#_Toc27843516)

[3.2 方案设计 16](#_Toc27843517)

[3.3 实验步骤 17](#_Toc27843518)

[3.4 测试与分析 17](#_Toc27843519)

[4 总结与心得 19](#_Toc27843520)

[4.1 实验总结 19](#_Toc27843521)

[4.2 实验心得 19](#_Toc27843522)

[参考文献 21](#_Toc27843523)

# 单周期MIPS(硬布线)

## 设计要求

利用logisim平台中现有运算部件构建MIPS数据通路，实现七条MIPS指令的执行，包括：LW，SW，BEQ，BNE，SYSCALL，ADDI，R\_Type。

表 1.1 MIPS七条指令及其功能说明

|  |  |
| --- | --- |
| 指令 | RTL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] 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 |

## 方案设计

### 设计思路：

实验需要建立完整的指令数据通路，首先将数据通路的实现细化为多个较小的子问题模块，包括取指令部件，寄存器堆，算数逻辑单元，数据存储器，不同指令的执行通路。下面主要分析取值部件和不同类型指令的数据通路，包括R型指令，LW/SW型指令，BNE/BEQ型指令的数据通路，并将其组合以构建整体指令的数据通路。

### MIPS指令格式分析

　所有MIPS指令的长度相同，都是32位。为了让指令的格式刚好合适，设计者做了折中：将所有指令定长，但是不同的指令有不同的格式。在MIPS架构中，指令的最高6位均为Op码，剩下的26位可以将指令分为3种类型，分别为R型、I型和J型。

R型指令用连续3个5位二进制码表示3个寄存器的地址，然后用1个5位二进制码表示移位的位数（如果未使用移位操作，则全为0），最后是6位的Function码（它与Op码共同决定R型指令的具体操作方式）。

I型指令则用连续2个5位二进制码表示2个寄存器的地址，然后是由1个16位二进制码表示1个立即数二进制码。

J型指令用26位二进制码表示跳转目标的指令地址（实际的指令地址应为32位，其中最低2位为“00”，最高4位由PC当前地址决定）。



图1.1 MIPS指令格式

Op: 指令基本操作，称为操作码。

Rs: 第一个源操作数寄存器。

Rt: 第二个源操作数寄存器。

Rd: 存放操作结果的目的操作数。

Shamt: 位移量。

Funct: 函数，这个字段选择Op操作某个特定变体。

### 取指令数据通路

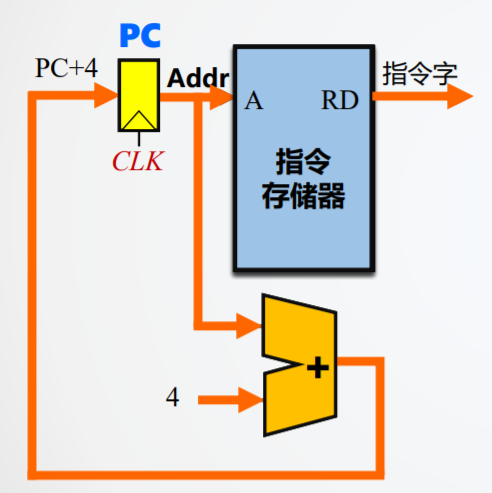


图1.2 取指令数据通路

### R型指令数据通路

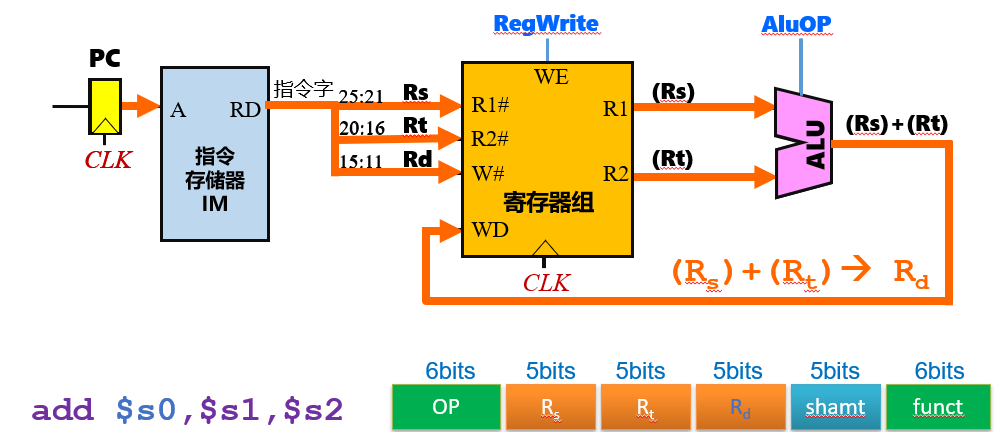


图1.3 R型数据通路

### Lw指令数据通路

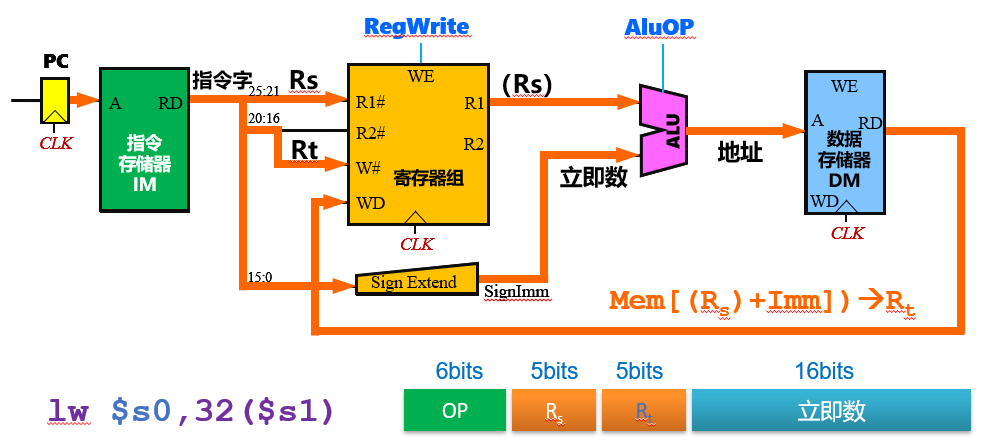


图1.4 LW型数据通路

### SW型指令数据通路

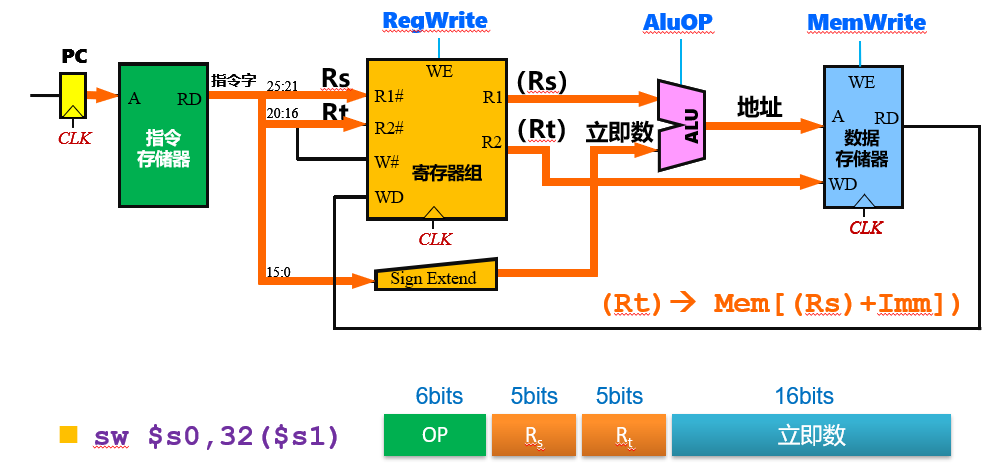


图1.5 SW型数据通路

### 整体数据通路

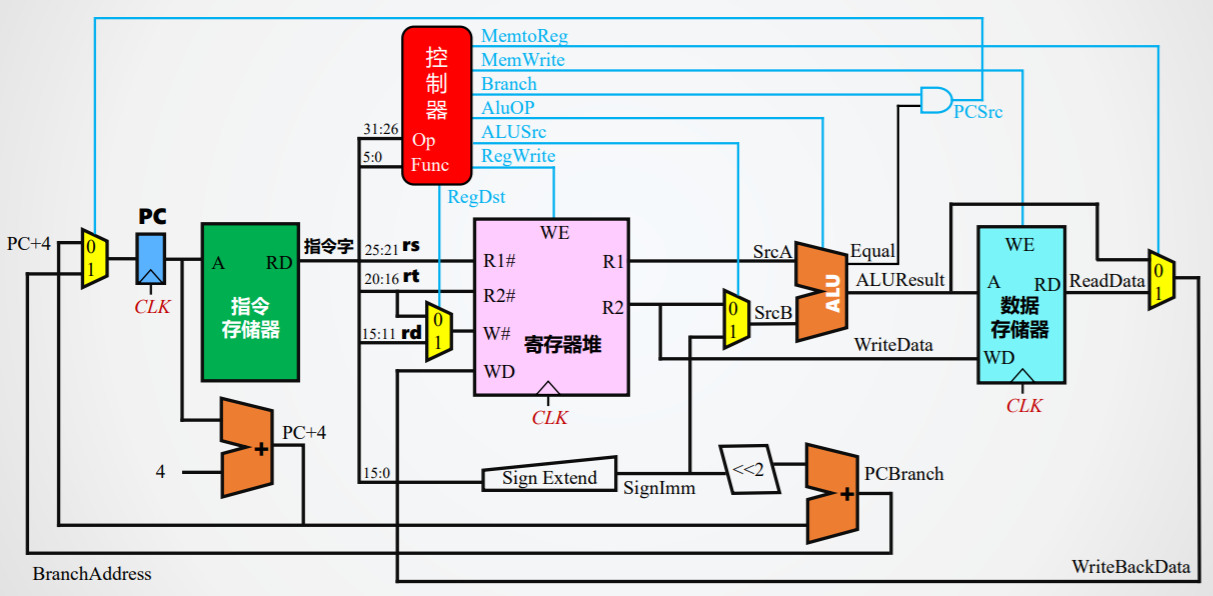


图1.6 单周期MIPS数据通路

单周期MIPS数据通路分析：单周期CPU指的是一条指令的执行在一个时钟周期内完成，然后开始下一条指令的执行，即一条指令用一个时钟周期完成。在处理指令时，一般需要经过以下几个步骤：

(1) 取指令：根据程序计数器PC中的指令地址，从存储器中取出一条指令，同时，PC根据指令字长度自动递增产生下一条指令所需要的指令地址，但遇到“地址转移”指令时，则控制器把“转移地址”送入PC，当然得到的“地址”需要做些变换才送入PC。

(2) 指令译码：对取指令操作中得到的指令进行分析并译码，确定这条指令需要完成的操作，从而产生相应的操作控制信号，用于驱动执行状态中的各种操作。

(3) 指令执行：根据指令译码得到的操作控制信号，具体地执行指令动作，然后转移到结果写回状态。

(4) 存储器访问：所有需要访问存储器的操作都将在这个步骤中执行，该步骤给出存储器的数据地址，把数据写入到存储器中数据地址所指定的存储单元或者从存储器中得到数据地址单元中的数据。

(5) 结果写回：指令执行的结果或者访问存储器中得到的数据写回相应的目的寄存器中。

单周期CPU，需要在一个时钟周期内完成这五个阶段的处理。

## 实验步骤

依据各个组件构建整体数据通路

单周期MIPS控制器设计，控制器各个引脚封装如图1.7

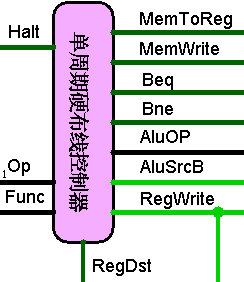


图1.7 控制器引脚封装

表1.2 控制器各引脚功能

|  |  |  |  |
| --- | --- | --- | --- |
| # | 控制信号 | 信号说明 | 产生条件 |
| 1 | MemToReg | 写入寄存器的数据来自存储器 | Lw指令 |
| 2 | MemWrute | 写内存控制信号 | Sw指令 |
| 3 | Beq | Beq指令译码信号 | Beq指令 |
| 4 | Bne | Bne指令译码信号 | Bne指令 |
| 5 | AluOP | 运算器操作控制符 | 加法，比较两种运算 |
| 6 | AluSrcB | 运算器第二输入选择 | Lw，sw，addi指令 |
| 7 | RegWrite | 寄存器写使能控制信号 | 寄存器写回信号 |
| 8 | RegDst | 写入寄存器选择控制信号 | R型指令 |
| 9 | Halt | 停机信号，取反后控制PC使能端 | Syacall指令 |

控制器具体实现

首先根据FUNCT字段和OpCode生成对应的ALUOP信号，如表1.3

表1.3 ALU\_OP控制信号

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | OPCODE(十进制) | FUNCT(十进制) | ALU\_OP |
| ADD | 0 | 32 | 5 |
| SLT | 0 | 42 | 11 |
| ADDI | 8 | X | 5 |
| LW | 35 | X | 5 |
| SW | 43 | X | 5 |
| BEQ | 4 | X | 5 |
| BNE | 5 | X | 5 |
| SYACALL | 0 | 12 | X |

根据EXCEL表生成ALU\_OP的表达式，在logisim中借助分析电路可得到ALU\_OP的生成电路。

接着需要实现指令译码，如图1.8所示：

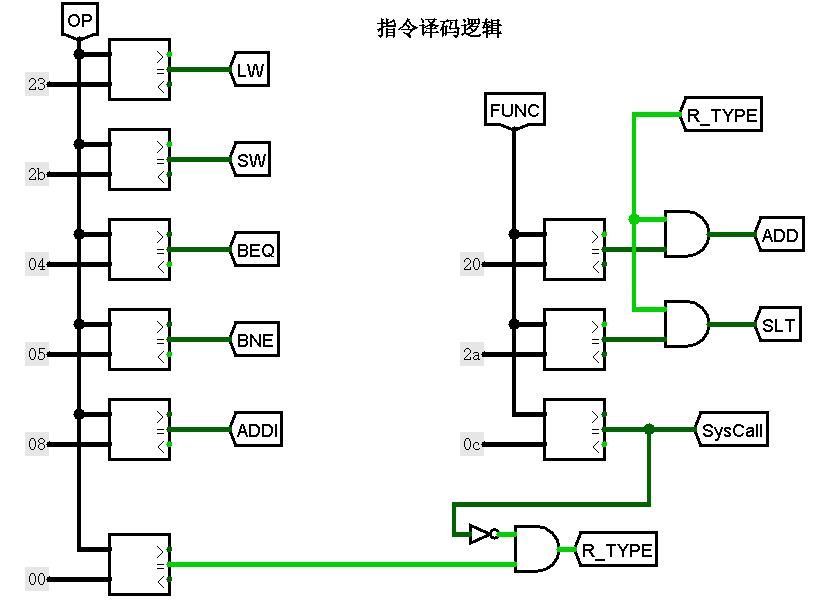


图1.8 指令译码

注意到SYSCALL指令属于特殊的R型指令，不与R\_TYPE型指令放在同一类型。

根据指令译码信号生成对应的引脚信号如图1.9所示



图1.9 引脚信号生成

## 故障与调试

### 单挑指令执行均通过，冒泡排序执行出错

**故障现象：**执行Syscall指令后，Reg中的数据发生变化，此时控制器的输出信号不对，定位到指令译码信号，发型R\_TYPE指令也有输出信号，检查指令译码逻辑。

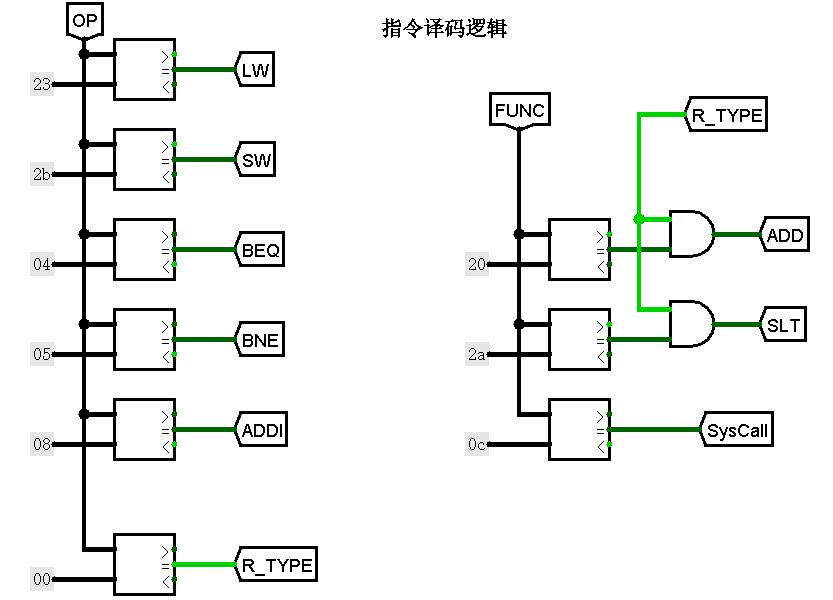


图1.10 syscall指令错误分析

**原因分析：**如图1.10，当OP为0，FUNC为0c时，R\_TYPE和Syscall指令会同时输出信号，导致RegDst和RegWrite有输出信号，带来寄存器的写操作。

**解决方案：**将R\_tyepe和Syscall指令分开，不把Syscall作为R\_type指令。

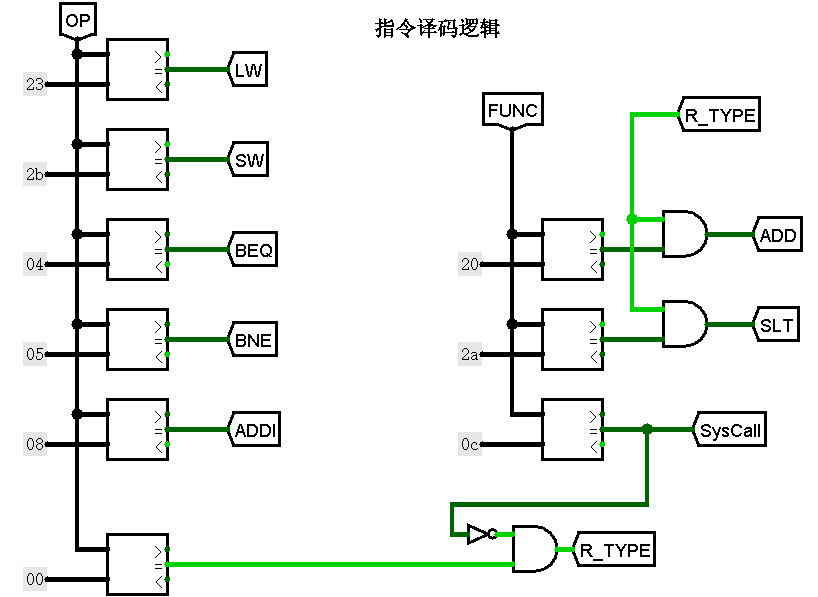


图1.11 指令译码逻辑纠错

## 测试与分析

在指令存储器中载入排序程序sort.hex

时钟自动仿真，Ctrl+k

程序停机后，查看数据存储器中排序情况，有符号降序排列

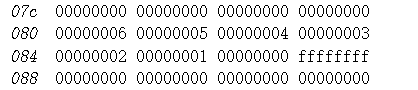


图1.12 冒泡排序运行结果

符合预期结果。

单周期MIPS\_CPU分析：单周期MIPS的每一条指令均在一个周期内完成，时钟周期取决于执行最慢的指令，时钟周期过长，效率较低，下面进行单条指令多个周期的设计。

# 多周期MIPS(微程序)

## 设计要求

将单周期MIPS的数据通路进行改进，同时设计微程序控制器，实现多个周期执行一条MIPS指令。

## 方案设计

### 设计思路

微程序控制的数据通路中需要实现的内容有：数据通路的重构，指令状态字的设计，微程序控制的实现。

### 数据通路的重构

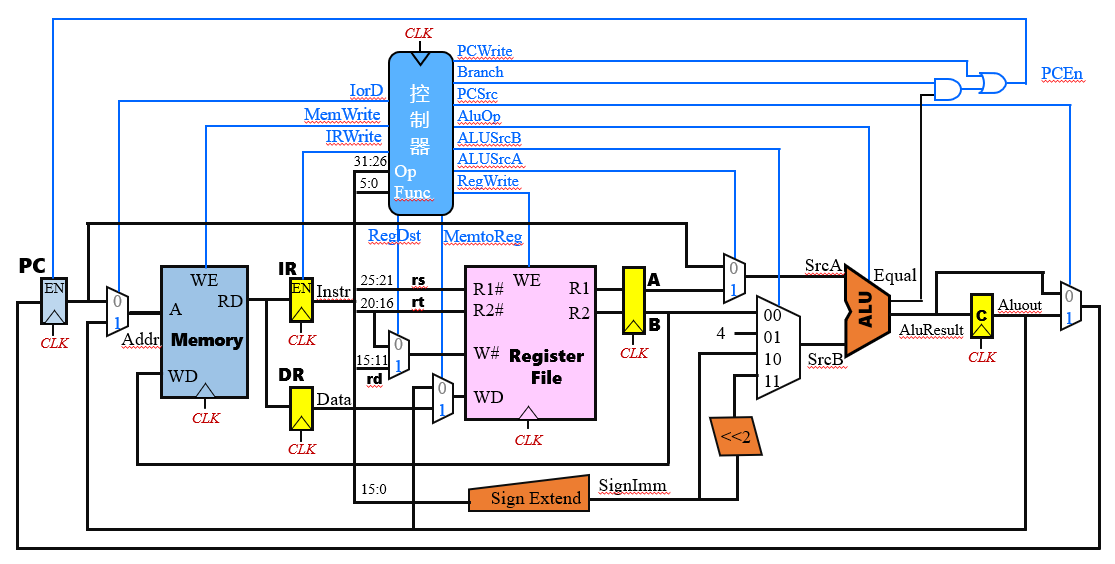


图2.1 多周期数据通路

图2.1为多周期CPU的顶层视图，与单周期CPU的区别有：不再区分指令存储器和数据存储器，分时使用部分功能部件；主要寄存器增加了输出端寄存器锁存数据；传输通路的延迟减小，时钟周期变短。

### 指令状态字设计

需要将不同的指令进行状态的划分

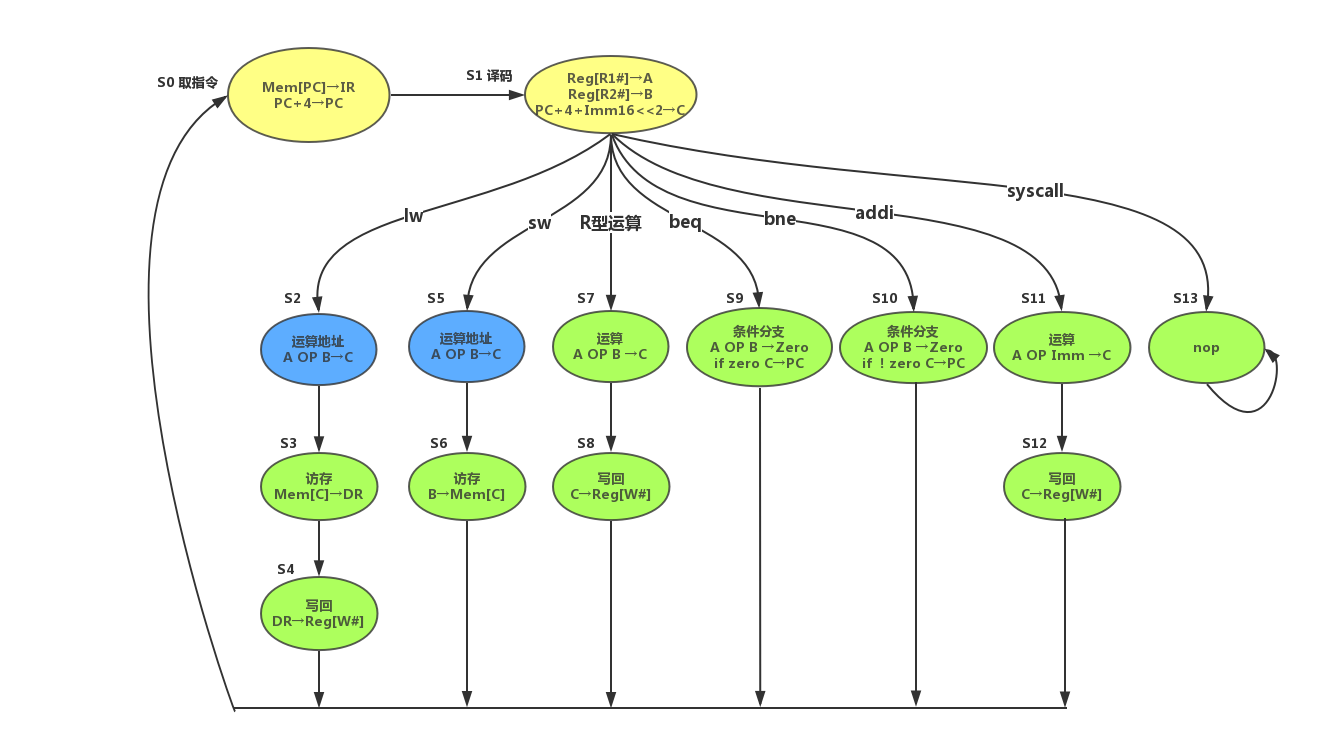


图2.2 指令状态字设计

将七种指令根据执行过程划分为13个不同的状态字，每个状态有各自不同的控制信号输出，从而控制数据通路的整体流通，后续根据状态字的输出信号进行微程序的设计。

### 微程序控制器实现

首先需要根据状态转换图设计微程序，

表2.1 微程序设计

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 微指令功能 | 状态 | 微指令地址 | IorD | PcSrc | AluSrcA | AluSrcB | MemToReg | RegDst | IrWrite | PcWrite |
| 取指令 | 0 | 0000 | 0 | 0 | 0 | 01 | 0 | 0 | 1 | 1 |
| 译码 | 1 | 0001 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 |
| LW1 | 2 | 0010 | 0 | 0 | 1 | 10 | 0 | 0 | 0 | 0 |
| LW2 | 3 | 0011 | 1 | 0 | 0 | 00 | 0 | 0 | 0 | 0 |
| LW3 | 4 | 0100 | 0 | 0 | 0 | 00 | 1 | 0 | 0 | 0 |
| SW1 | 5 | 0101 | 0 | 0 | 1 | 10 | 0 | 0 | 0 | 0 |
| SW2 | 6 | 0110 | 1 | 0 | 0 | 00 | 0 | 0 | 0 | 0 |
| R1 | 7 | 0111 | 0 | 0 | 1 | 00 | 0 | 0 | 0 | 0 |
| R2 | 8 | 1000 | 0 | 0 | 0 | 00 | 0 | 1 | 0 | 0 |
| BEQ | 9 | 1001 | 0 | 1 | 1 | 00 | 0 | 0 | 0 | 0 |
| BNE | 10 | 1010 | 0 | 1 | 1 | 00 | 0 | 0 | 0 | 0 |
| ADDI1 | 11 | 1011 | 0 | 0 | 1 | 10 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 微指令功能 | 状态 | RegWrite | MemWrite | MemRead | BEQ | BNE | AluControl | P | 下址字段 | 十六进制 |
| 取指令 | 0 | 0 | 0 | 1 | 0 | 0 | 00 | 0 | 0001 | 13201 |
| 译码 | 1 | 0 | 0 | 0 | 0 | 0 | 00 | 1 | 0000 | 30010 |
| LW1 | 2 | 0 | 0 | 0 | 0 | 0 | 00 | 0 | 0011 | 60003 |
| LW2 | 3 | 0 | 0 | 1 | 0 | 0 | 00 | 0 | 0100 | 100204 |
| LW3 | 4 | 1 | 0 | 0 | 0 | 0 | 00 | 0 | 0000 | 8800 |
| SW1 | 5 | 0 | 0 | 0 | 0 | 0 | 00 | 0 | 0110 | 60006 |
| SW2 | 6 | 0 | 1 | 0 | 0 | 0 | 00 | 0 | 0000 | 100400 |
| R1 | 7 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 1000 | 40048 |
| R2 | 8 | 1 | 0 | 0 | 0 | 0 | 00 | 0 | 0000 | 4800 |
| BEQ | 9 | 0 | 0 | 0 | 1 | 0 | 00 | 0 | 0000 | C0100 |
| BNE | 10 | 0 | 0 | 0 | 0 | 1 | 00 | 0 | 0000 | C0080 |
| ADDI1 | 11 | 0 | 0 | 0 | 0 | 0 | 00 | 0 | 1100 | 6000C |

下面按微程序入口地址设计地址转移逻辑：

表2.2 状态转移逻辑

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **机器指令译码信号** | | | | | | | **微程序入口地址** | | | | |
| **R\_Type** | **ADDI** | **LW** | **SW** | **BEQ** | **BNE** | **SYSCALL** | **入口地址 10进制** | **S3** | **S2** | **S1** | **S0** |
| 1 |  |  |  |  |  |  | 7 | 0 | 1 | 1 | 1 |
|  | 1 |  |  |  |  |  | 11 | 1 | 0 | 1 | 1 |
|  |  | 1 |  |  |  |  | 2 | 0 | 0 | 1 | 0 |
|  |  |  | 1 |  |  |  | 5 | 0 | 1 | 0 | 1 |
|  |  |  |  | 1 |  |  | 9 | 1 | 0 | 0 | 1 |
|  |  |  |  |  | 1 |  | 10 | 1 | 0 | 1 | 0 |
|  |  |  |  |  |  | 1 | 13 | 1 | 1 | 0 | 1 |

将EXCL生成的表达式输入到logisim中生成地址转换逻辑电路，接着进行微程序控制器的构造：

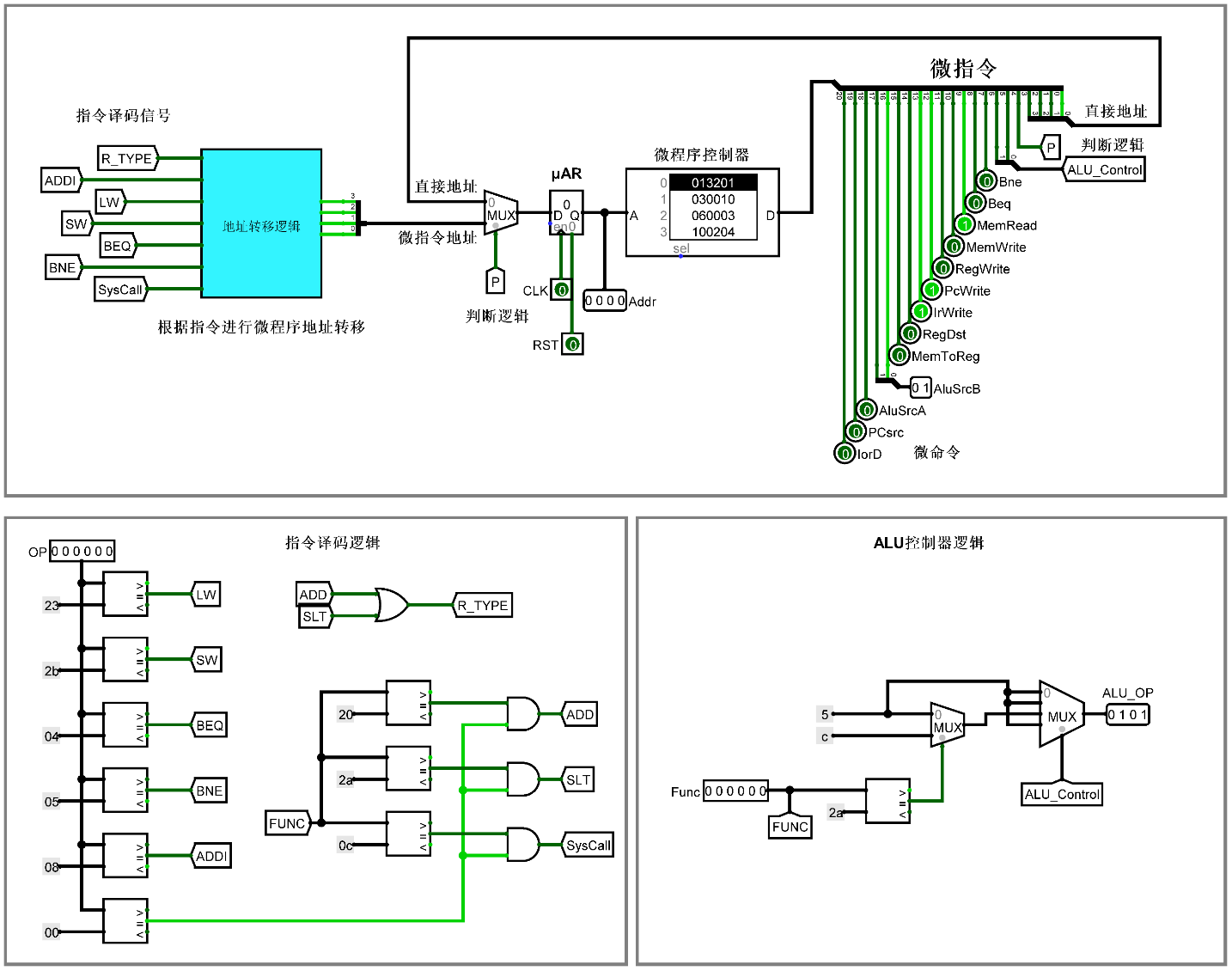


图2.3 微程序控制器

微程序控制器分析：首先指令译码信号经过地址转移逻辑会找到相应的微指令地址，根据P是否为1判断是否执行倒下一条指令，当P==0时，判断逻辑会从当前位置了的下址字段读出下一条位置令的地址，从而进入到当前执行指令的下一个状态，知道P==1时，才会执行下一条指令。在每一条微指令中，会分解出对应的输出信号来控制整个数据通路的数据流动。

## 实验步骤

1. 重构数据通路
2. 设计每条指令的状态字
3. 设计微程序和地址转换逻辑，构建微程序控制器电路

## 故障与调试

### syscall状态变化不符合预期

**故障现象：**执行syscall指令后，下一个运行状态不符合预期，未停机。

**原因分析：** 经检查数据通路并无明显问题，将问题定位到微程序的设计上，发现状态转移存在部分问题，syscall状态之后又回到了取指令状态。

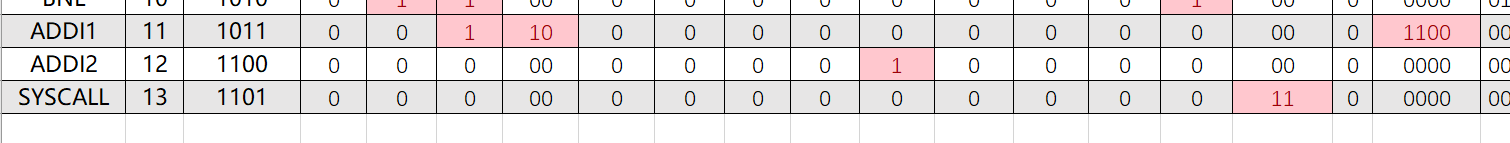


图2.4syscall微程序设计

**解决方案：** syscall的下一个状态为1101，即保持syscall状态不变。

## 测试与分析

在指令存储器中载入排序程序sort.hex

时钟自动仿真，Ctrl+k

程序停机后，查看数据存储器中排序情况，有符号降序排列



图2.5 冒泡排序运行结果

符合预期结果

# 多周期MIPS(硬布线)

## 设计要求

设计多周期硬布线控制器代替微程序，实现对数据通路的控制

## 方案设计

### 设计思路

复用微程序控制器中的微程序单元，利用组合逻辑重写控制单元，设计状态机控制器，在相同的数据通路中替换掉微程序控制器即可。

### FSM控制器设计

FSM的输入为现态和指令译码信号，输出为次态。根据微程序控制器中的状态字完成状态转换即可。具体状态转换如下表：

表3.1 FSM状态转移设计

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 现态 | 输入信号 | | | | | | | 次态 |
| 10进制 | R\_Type | LW | SW | BEQ | BNE | SYSCALL | ADDI | 10进制 |
| 2 |  | 1 |  |  |  |  |  | 3 |
| 3 |  | 1 |  |  |  |  |  | 4 |
| 4 |  |  |  |  |  |  |  | 0 |
| 1 |  |  | 1 |  |  |  |  | 5 |
| 5 |  |  | 1 |  |  |  |  | 6 |
| 6 |  |  |  |  |  |  |  | 0 |
| 7 | 1 |  |  |  |  |  |  | 8 |
| 8 |  |  |  |  |  |  |  | 0 |
| 1 |  |  |  | 1 |  |  |  | 9 |
| 9 |  |  |  |  |  |  |  | 0 |
| 1 |  |  |  |  | 1 |  |  | 10 |
| 10 |  |  |  |  |  |  |  | 0 |
| 1 |  |  |  |  |  | 1 |  | 13 |
| 13 |  |  |  |  |  |  |  | 13 |
| 2 |  | 1 |  |  |  |  |  | 3 |
| 3 |  | 1 |  |  |  |  |  | 4 |
| 4 |  |  |  |  |  |  |  | 0 |
| 1 |  |  | 1 |  |  |  |  | 5 |

在logisim中生成对应的FSM控制电路即可，构建的FSM控制器如图

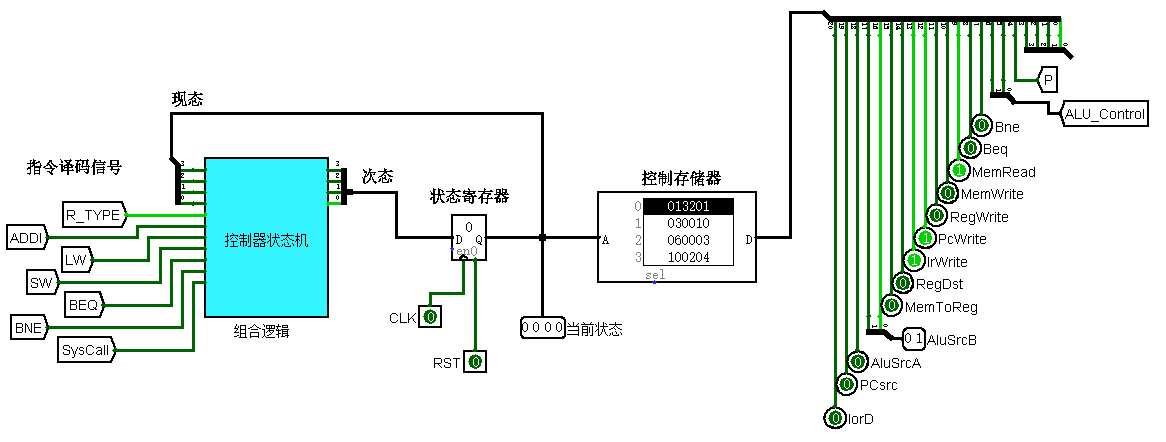


图 3.1 FSM控制器

## 实验步骤

1. 构建与实验2相同的数据通路
2. 设计FSM状态转移电路
3. 设计FSM控制器

## 测试与分析

在指令存储器中载入排序程序sort.hex

时钟自动仿真，Ctrl+k

程序停机后，查看数据存储器中排序情况，有符号降序排列

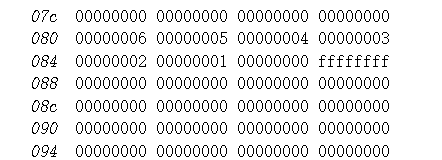


图3.2 冒泡排序运行结果

符合预期结果。

# 总结与心得

## 实验总结

实验一设计数据通路的过程中，将整个数据通路的构建划分成了几个小的问题，取指令部件，寄存器堆，算数逻辑单元，数据存储器等等，很多部件是在前几次实验中设计过的。从开始设计小的组件，到最终完成整个cpu的构建，很好的体现出了计算机结构的层次性。

实验二设计多周期CPU，最初的工作是重构数据通路，需要控制每条指令的各个状态，以及各个状态下的控制信号的改变，从而达到不同状态执行不同操作的目的。有了单周期CPU的基础，所以在实现的时候对于每条指令的理解更加清晰。后续进行微程序的设计，在这个过程中，多次使用了EXCEL和logisim的自动生成功能，减轻了很多的工作量。微程序的工作方式，将众多的输出信号进行集合，在设计完成所有的状态字后，针对每一种状态设定特点的信号集，在完成信号转变后再对信号集进行解析，从而将信号输出和指令跳转进行了分离。

实验三将微程序替换为硬布线的方式，在我看来像是替换了微程序的控制电路，其利用的思想还是将输出信号与控制电路分开，以FSM状态转换的方式来区分不同的输出信号集。实现方式直接里用EXCEL设计状态字转换，实现较为简单。

## 实验心得

通过本次CPU的实验，收获良多，对于计算机的层次结构和cpu内部的工作原理有了较为清晰的认识。

实验是对于课本内容的实践，通过实验我系统的了解了MIPS指令执行的流程，每条指令的执行过程都有了一定程度上的理解。在排查指令问题的时候，都是先构想一遍指令的执行过程，再分析可能出错的地方，大大提高了效率。

本次实验中所用到的微程序以及状态机是两种不同的控制状态转移的方式，巧妙的设计模式在其他系统的设计中具有借鉴意义。

实验中在设计组合逻辑的时候多次使用了logisim的自动生成电路的功能，学会利用现有的工具解决问题，去替代重复性的工作是一种很重要的能力。

总的来说，组成原理实验和课程的内容联系非常紧密，对于我们掌握课程内容，了解原理有很大的帮助，实验的难度贴切，希望能将组原实验的设计理念向其他课程实验上反馈一些，有些课程实验内容过于偏离课程，花费大量的时间效果也不是特别好。

# 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: 嵌入签名图片** |
| 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:** |