***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： | ACM1701 |
| 学 号： | U201714780 |
| 姓 名： | 刘晨彦 |
| 电 话： | 15927172332 |
| 邮 件： | [Chenyanliu712@qq.com](mailto:Chenyanliu712@qq.com) |
| 完成日期： | 2019-12-12 |



目 录

[1 CPU设计实验 1](#_Toc27232215)

[1.1 设计要求 1](#_Toc27232216)

[1.2 方案设计 6](#_Toc27232217)

[1.3 实验步骤 16](#_Toc27232218)

[1.4 故障与调试 17](#_Toc27232219)

[1.5 测试与分析 18](#_Toc27232220)

[2 总结与心得 21](#_Toc27232221)

[2.1 实验总结 21](#_Toc27232222)

[2.2 实验心得 21](#_Toc27232223)

[参考文献 22](#_Toc27232224)

# CPU设计实验

## 设计要求

### 单周期MIPS硬布线CPU的设计要求

#### 单周期MIPS硬布线CPU通路的设计要求

利用Logisim平台中现有运算部件构建一个单周期硬布线的CPU，支持八种基本的MIPS指令，能够实现内存区域的冒泡排序。单周期MIPS硬布线CPU通路的输入输出引脚见下表。

表 1.1单周期MIPS硬布线电路引脚与功能描述

|  |  |  |  |
| --- | --- | --- | --- |
| 引脚 | 输入/输出 | 位宽 | 功能描述 |
| RST | 输入 | 1 | 复位信号 |
| CLK | 输入 | 1 | 时钟信号 |
| PC | 输出 | 32 | 地址 |
| IR | 输出 | 32 | 指令 |
| RegWrite | 输出 | 1 | 寄存器写使能控制信号 |
| RDin | 输出 | 32 | 写入寄存器的内容 |
| MemWrite | 输出 | 1 | 写内存控制信号 |
| MDin | 输出 | 32 | 写入随机存储器的内容 |

#### 单周期硬布线控制器的设计要求

为了实现电路的控制，需要利用Logisim平台设计一单周期硬布线控制器，能够根据输入的指令信号输出对应的控制信号。单周期硬布线控制器电路的输入输出引脚见下表。

表 1.2单周期硬布线控制器电路引脚与功能描述

|  |  |  |  |
| --- | --- | --- | --- |
| 引脚 | 输入/输出 | 位宽 | 功能描述 |
| OP | 输入 | 6 | 操作码 |
| Func | 输入 | 6 | 功能码 |
| ALU\_OP | 输出 | 4 | 运算器操作控制符，5H为加法，BH为比较 |
| RegDst | 输出 | 1 | 写入寄存器选择控制信号 |
| RegWrite | 输出 | 1 | 寄存器写使能控制信号 |
| MenToReg | 输出 | 1 | 写入寄存器的数据来自存储器 |
| MemWrite | 输出 | 1 | 写内存控制信号 |
| AluSrc | 输出 | 1 | 运算器第二输入选择 |
| Beq | 输出 | 1 | Beq指令译码信号 |
| Bne | 输出 | 1 | Bne指令译码信号 |
| Halt | 输出 | 1 | 停机信号 |

### 多周期MIPS微程序CPU的设计要求

#### 多周期MIPS微程序CPU通路的设计要求

利用Logisim平台中现有运算部件构建一个多周期微程序CPU，支持八种MIPS指令，能够实现简单的冒泡排序。多周期微程序CPU电路的输入输出引脚见下表。

表 1.3多周期微程序CPU电路引脚与功能描述

|  |  |  |  |
| --- | --- | --- | --- |
| 引脚 | 输入/输出 | 位宽 | 功能描述 |
| RST | 输入 | 1 | 复位信号 |
| CLK | 输入 | 1 | 时钟信号 |
| PC | 输出 | 32 | 地址 |
| IR | 输出 | 32 | 指令 |
| RegWrite | 输出 | 1 | 寄存器写使能控制信号 |
| RDin | 输出 | 32 | 写入寄存器的内容 |
| MemWrite | 输出 | 1 | 写内存控制信号 |
| MDin | 输出 | 32 | 写入随机存储器的内容 |

#### 多周期微程序控制器的设计要求

为了实现电路的控制，需要设计多周期微程序控制器，用于对应指令下提供对应的控制指令。控制器的框架如下图所示：

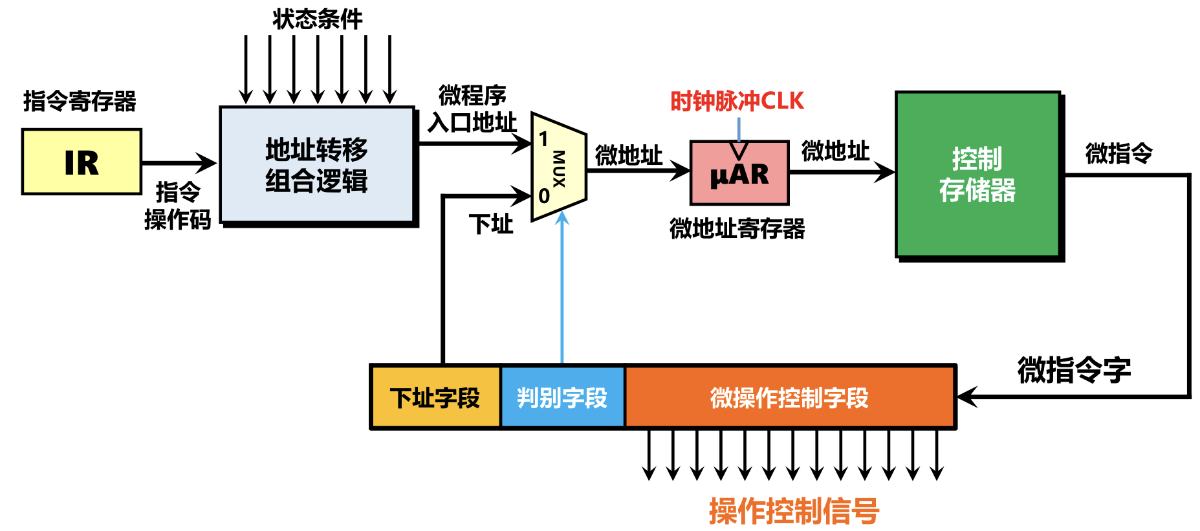


图 1.1 微程序控制器原理框架

当指令输入微程序控制器时，首先通过地址转移的组合逻辑电路给出该指令首个时钟周期内的控制指令所在的地址，利用该地址从微程序控制存储器中取得控制指令并输出控制信号，该指令若不是单周期的指令，则从控制指令中给出下一个时钟周期的控制指令所在地址，重复上述过程直至一条指令执行结束，回到微地址寄存器首行进行取指令和译码，处理下一条指令。

电路的输入输出引脚见下表：

表 1.4 多周期微程序CPU控制器电路引脚与功能描述

|  |  |  |  |
| --- | --- | --- | --- |
| 引脚 | 输入/输出 | 位宽 | 功能描述 |
| OP | 输入 | 6 | 操作码 |
| Func | 输入 | 6 | 功能码 |
| PCWrite | 输出 | 1 | PC写使能控制信号 |
| LorD | 输出 | 1 | 指令或数据信号：0表示指令，1表示数据 |
| IRwrite | 输出 | 1 | 指令寄存器写使能 |
| MemWrite | 输出 | 1 | 写内存控制信号 |
| MemRead | 输出 | 1 | 读内存控制信号 |
| Beq | 输出 | 1 | Beq指令译码信号 |
| Bne | 输出 | 1 | Bne指令译码信号 |
| PcSrc | 输出 | 1 | PC输入来源信号 |
| AluOP | 输出 | 4 | 运算器操作控制符 |
| AluSrcA | 输出 | 1 | 运算器第一输入选择信号 |
| AluSrcB | 输出 | 2 | 运算器第二输入选择信号 |
| RegWrite | 输出 | 1 | 寄存器写使能控制信号 |
| RegDst | 输出 | 1 | 写入寄存器选择控制信号 |
| MenToReg | 输出 | 1 | 写入寄存器的数据来自存储器 |

#### 微程序地址转移逻辑电路的设计要求

地址转移逻辑则需要根据输入的指令，通过组合逻辑电路产生对应的控制信号的首地址。微程序地址转移逻辑电路的输入输出引脚如下表所示：

表 1.5 多周期微程序CPU地址转移逻辑电路引脚与功能描述

|  |  |  |  |
| --- | --- | --- | --- |
| 引脚 | 输入/输出 | 位宽 | 功能描述 |
| RTYPE | 输入 | 1 | 1：指令为RTYPE，0：指令不是RTYPE |
| ADDI | 输入 | 1 | 1：指令为ADDI，0：指令不是ADDI |
| LW | 输入 | 1 | 1：指令为LW，0：指令不是LW |
| SW | 输入 | 1 | 1：指令为SW，0：指令不是SW |
| BEQ | 输入 | 1 | 1：指令为BEQ，0：指令不是BEQ |
| BNQ | 输入 | 1 | 1：指令为BNQ，0：指令不是BNQ |
| SYSCALL | 输入 | 1 | 1：指令为SYSCALL，0：指令不是SYSCALL |
| S1 | 输出 | 1 | 控制信号地址第0位 |
| S2 | 输出 | 1 | 控制信号地址第1位 |
| S3 | 输出 | 1 | 控制信号地址第2位 |
| S4 | 输出 | 1 | 控制信号地址第3位 |

### 多周期MIPS硬布线CPU的设计要求

#### 多周期MIPS硬布线CPU通路的设计要求

利用Logisim平台中现有运算部件构建一个多周期硬布线CPU，支持八种基本的MIPS指令，能够实现简单的冒泡排序。多周期硬布线CPU电路的输入输出引脚见下表。

表 1.6 多周期硬布线CPU电路引脚与功能描述

|  |  |  |  |
| --- | --- | --- | --- |
| 引脚 | 输入/输出 | 位宽 | 功能描述 |
| RST | 输入 | 1 | 复位信号 |
| CLK | 输入 | 1 | 时钟信号 |
| PC | 输出 | 32 | 地址 |
| IR | 输出 | 32 | 指令 |
| RegWrite | 输出 | 1 | 寄存器写使能控制信号 |
| RDin | 输出 | 32 | 写入寄存器的内容 |
| MemWrite | 输出 | 1 | 写内存控制信号 |
| MDin | 输出 | 32 | 写入随机存储器的内容 |

#### 多周期硬布线控制器的设计要求

为了实现电路的控制，需要设计多周期微程序控制器，根据上一条控制指令的地址和指令译码信号给出下一条控制指令的地址，根据下一条指令的地址在控制存储器中找到对应的控制指令并输出。电路的输入输出引脚与表1.4一致，此处不再复述。

#### 多周期硬布线有限状态机的设计要求

设计一个组合逻辑电路，以现态和指令译码信号为输入，下一条控制指令地址为次态和输出。电路的输入输出引脚如下表。

表 1.7 多周期硬布线有限状态机电路引脚与功能描述

|  |  |  |  |
| --- | --- | --- | --- |
| 引脚 | 输入/输出 | 位宽 | 功能描述 |
| S0 | 输入 | 1 | 现态控制指令地址第0位 |
| S1 | 输入 | 1 | 现态控制指令地址第1位 |
| S2 | 输入 | 1 | 现态控制指令地址第2位 |
| S3 | 输入 | 1 | 现态控制指令地址第3位 |
| RTYPE | 输入 | 1 | 1：指令为RTYPE，0：指令不为RTYPE |
| ADDI | 输入 | 1 | 1：指令为ADDI，0：指令不为ADDI |
| LW | 输入 | 1 | 1：指令为LW，0：指令不为LW |
| SW | 输入 | 1 | 1：指令为SW，0：指令不为SW |
| BEQ | 输入 | 1 | 1：指令为BEQ，0：指令不为BEQ |
| BNE | 输入 | 1 | 1：指令为BNE，0：指令不为BNE |
| SYSCALL | 输入 | 1 | 1：指令为SYSCALL，0：指令不为SYSCALL |
| N0 | 输出 | 1 | 次态控制指令地址第0位 |
| N1 | 输出 | 1 | 次态控制指令地址第1位 |
| N2 | 输出 | 1 | 次态控制指令地址第2位 |
| N3 | 输出 | 1 | 次态控制指令地址第3位 |

## 方案设计

### 单周期MIPS硬布线CPU的方案设计

#### 单周期MIPS硬布线CPU通路设计

首先根据提供的单周期CPU的参考通路，绘制单周期MIPS硬布线CPU电路。如下图所示。

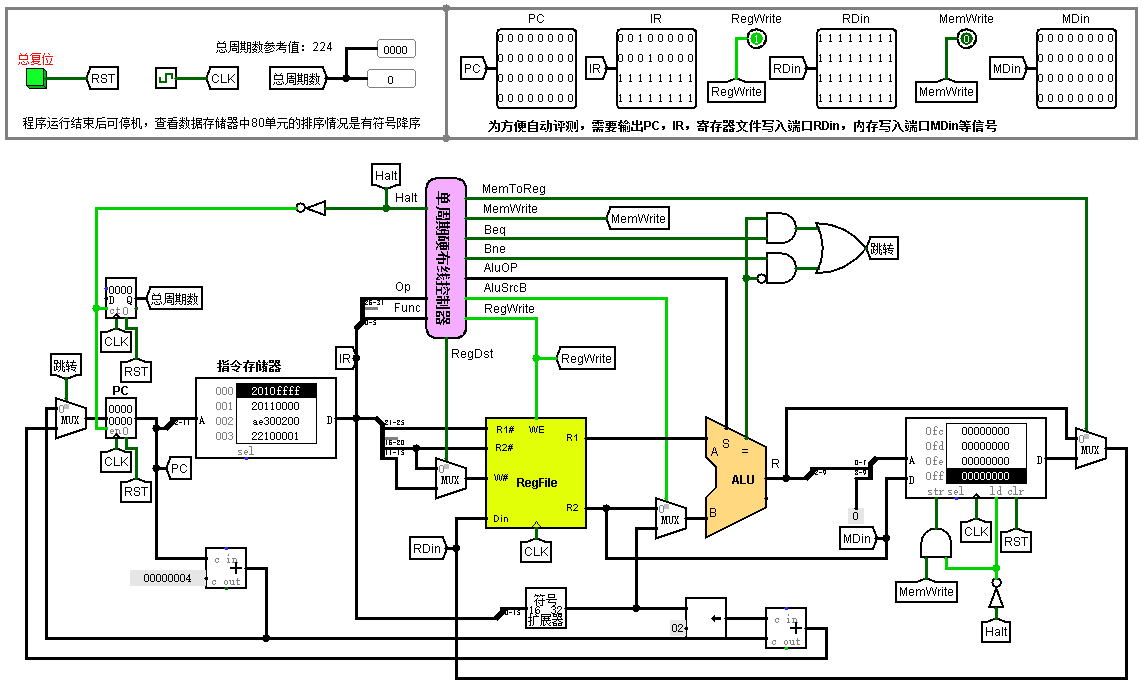


图 1.2 单周期MIPS硬布线CPU电路图

#### 单周期硬布线控制器设计

在指令译码逻辑电路中，可以查找MIPS32指令手册可知指令对应的操作码和功能码，如下表所示。其中指令ADD和SLT为R型指令。

表 1.8 指令与对应操作码和功能码表

|  |  |  |
| --- | --- | --- |
| 指令 | 操作码 | 功能码 |
| LW | 23H | NULL |
| SW | 2BH | NULL |
| BEQ | 04H | NULL |
| BNE | 05H | NULL |
| ADDI | 08H | NULL |
| ADD | 00H | 20h |
| SLT | 00H | 2AH |
| SYSCALL | 00H | 0CH |

在ALU控制逻辑中，当且仅当指令为SLT时，ALU控制器输出比较信号0BH，其余情况均输出加法信号05H。

在控制信号电路中，根据各指令的功能，输出对应的控制信号。LW指令需要写入寄存器、将存储器的数据写入寄存器、将立即数位扩展后和寄存器R1的输出相加。故需要RegWrite、MenToReg、AluSrc信号为1。SW指令需要写入存储器、将立即数位扩展后和寄存器R1的输出相加，故需要MemWrite、AluSrc信号为1。BEQ指令只需要Beq信号为1，BNE指令只需要Bne信号为1。ADDI指令需要写入寄存器、将立即数位扩展后和寄存器R1的输出相加，故需要RegWrite、AluSrc信号为1。R型指令则需要指令的第11~15位作为寄存器的写入地址、写入寄存器，故需要RegDst、RegWrite信号为1。最后SYSCALL指令则只需要停机信号Halt为1即可。

根据以上控制器的设计思路，绘制电路图如下所示：

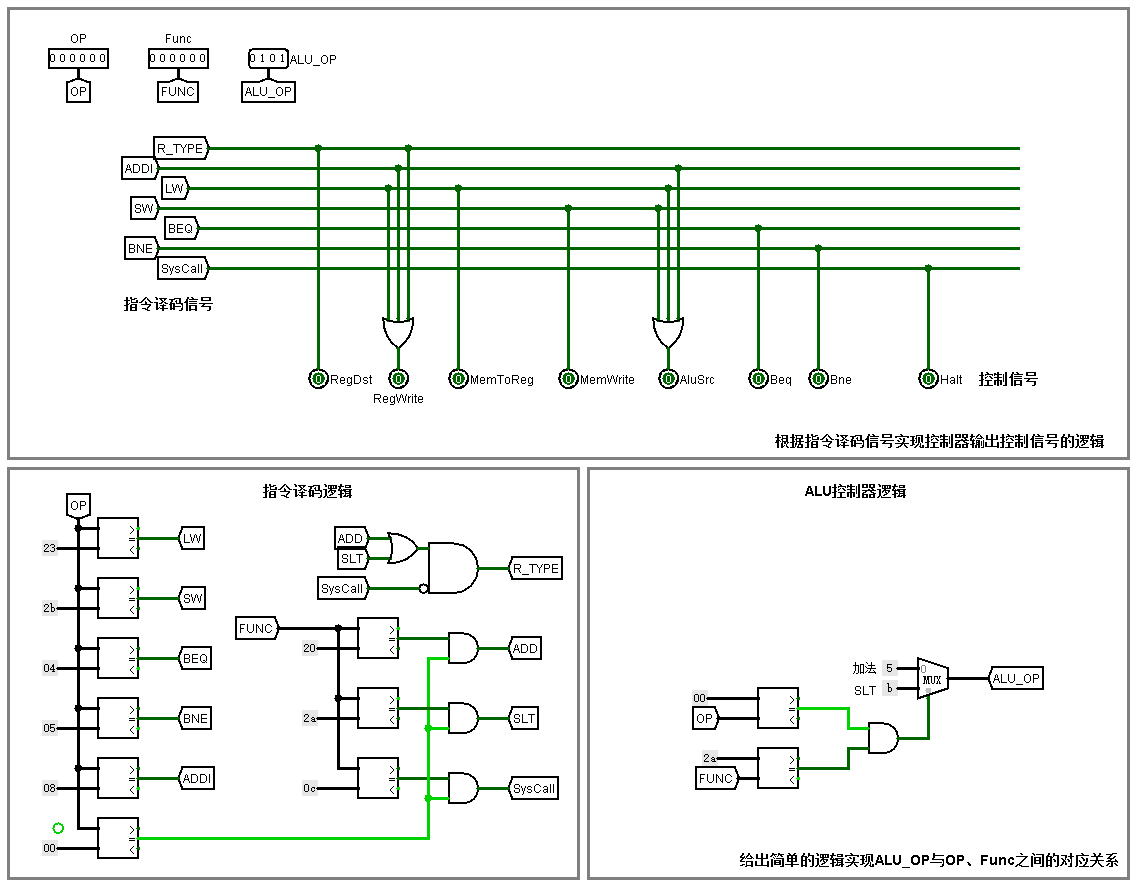


图 1.3 单周期硬布线控制器电路

### 多周期MIPS微程序CPU的方案设计

#### 多周期MIPS微程序CPU通路的设计

首先根据提供的多周期CPU的参考通路，绘制多周期微程序CPU电路。如下图所示。

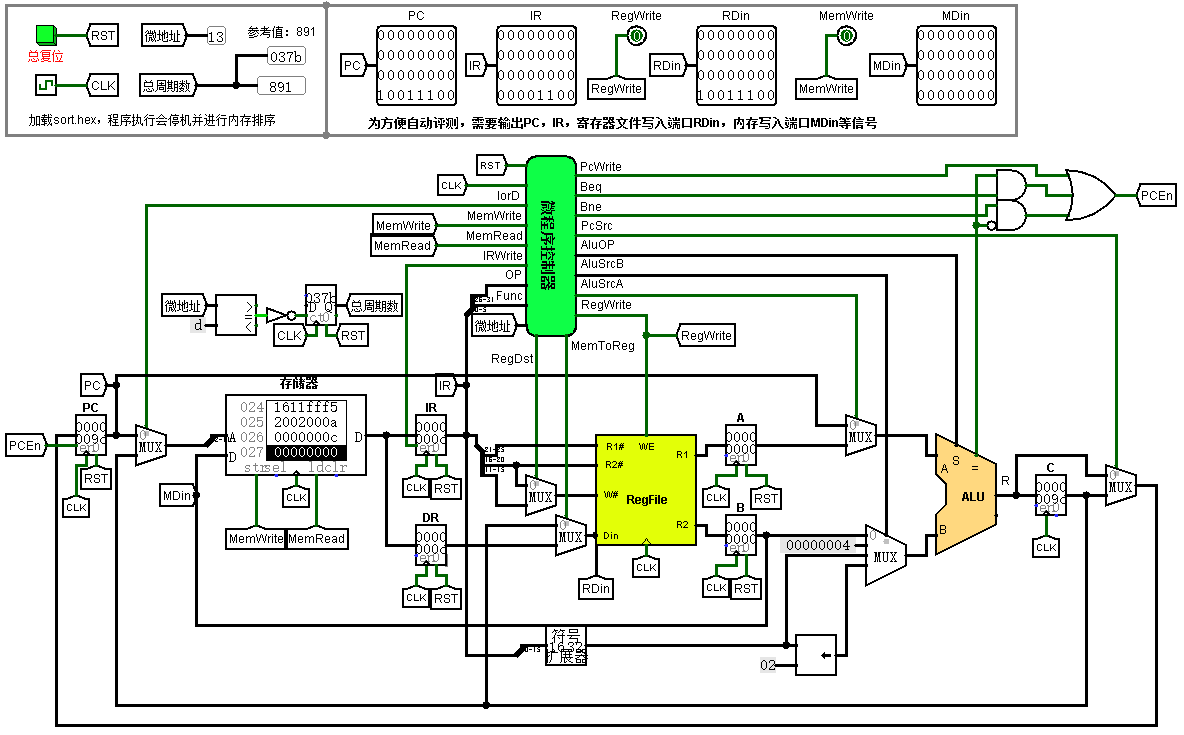


图 1.4 多周期微程序CPU电路

#### 微程序控制器电路的设计

指令译码逻辑电路和单周期硬布线控制器中的一致，此处不再赘述。

在ALU控制器逻辑中，当ALU\_Control为00时运算器做加法，为01时做减法，为10时由FUNC决定，此时仅当FUNC为2AH时执行比较操作。

在控制存储器中需要填写对应的控制信号，指令各周期的操作如下表：

表 1.9 指令各周期对应动作表

|  |  |
| --- | --- |
| 指令阶段 | 操作流程 |
| 取指令 | (MEM[PC])-> IR  (PC)+4->PC |
| 译码 | (R[IR[25:21]])->A  (R[IR[20:16]])->B  (PC)+(S-EXT(IR[15:0])<<2) ->C |
| LW1 | (A)+ S-EXT(IR[15:0]) ->C |
| LW2 | (MEM[PC]) ->DR |
| LW3 | (DR)-> R[IR[20:16]] |
| SW1 | (A)+ S-EXT(IR[15:0]) ->C |
| SW2 | (MEM[PC])->DR |
| RTYPE1 | (A)+(B)| ((A)<(B))->C |
| RTYPE2 | (C)-> R[IR[15:11]] |
| BEQ | If(A==B) (C) -> PC |
| BNE | If(A!=B) (C)-> PC |
| ADDI1 | (A)+ S-EXT(IR[15:0]) ->C |
| ADDI2 | (C) -> R[IR[20:16]] |
| SYSCALL | 空操作 |

根据上表，可在提供的微指令自动生成表中填写相对应的控制信号，填写情况如图所示：

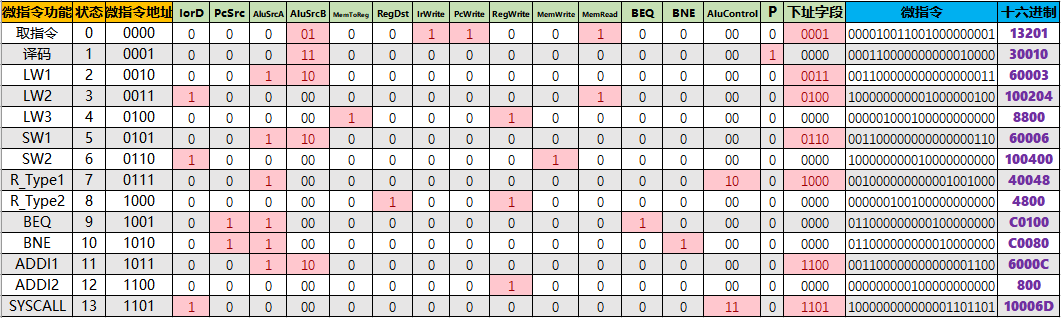


图 1.5 微指令自动生成表

将十六进制结果复制黏贴至微程序控制器即可。根据以上设计，得到多周期微指令控制器如下图所示：

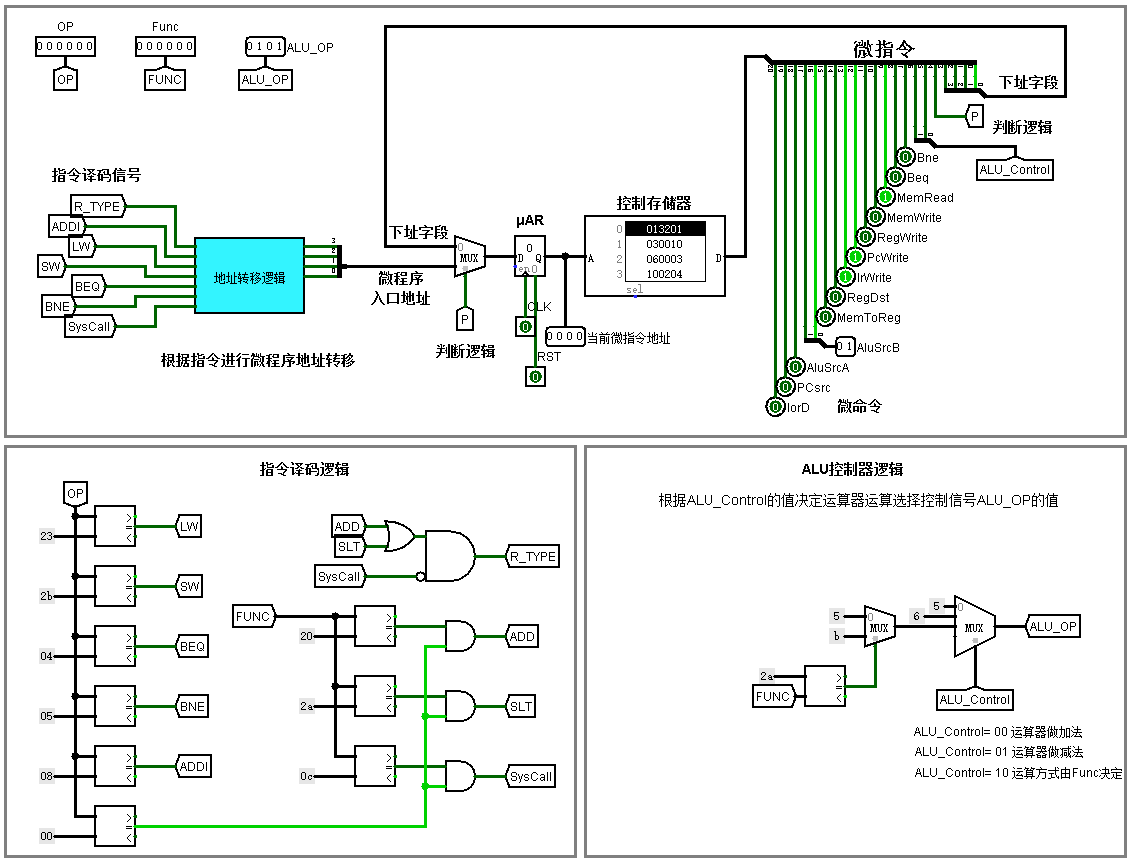


图 1.6 微程序控制器电路

#### 微程序地址转移逻辑电路设计

根据微指令自动生成表，可以得到每条指令的首地址。将每条指令首地址以十进制形式输入微地址转移逻辑自动生成表，可以得到转移逻辑电路的组合逻辑。如下图和下表所示：



图 1.7 微程序地址入口表截图

表 1.10 位号与对应地址组合逻辑表

|  |  |
| --- | --- |
| 位号 | 组合逻辑 |
| S0 | R\_Type + ADDI + SW + BEQ + SYSCALL |
| S1 | R\_Type + ADDI + LW + BNE |
| S2 | R\_Type + SW + SYSCALL |
| S3 | ADDI + BEQ + BNE + SYSCALL |

将对应组合逻辑复制粘贴至微程序地址转移逻辑电路中自动生成电路，如下图所示：

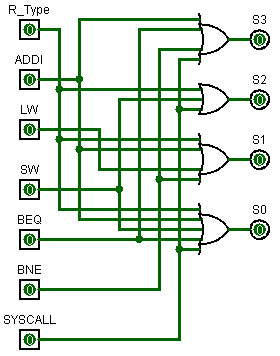


图 1.8 微程序地址转移逻辑电路

### 多周期MIPS硬布线CPU的方案设计

#### 多周期MIPS硬布线CPU通路的方案设计

首先根据提供的多周期CPU的参考通路，绘制多周期硬布线CPU电路。如下图所示。

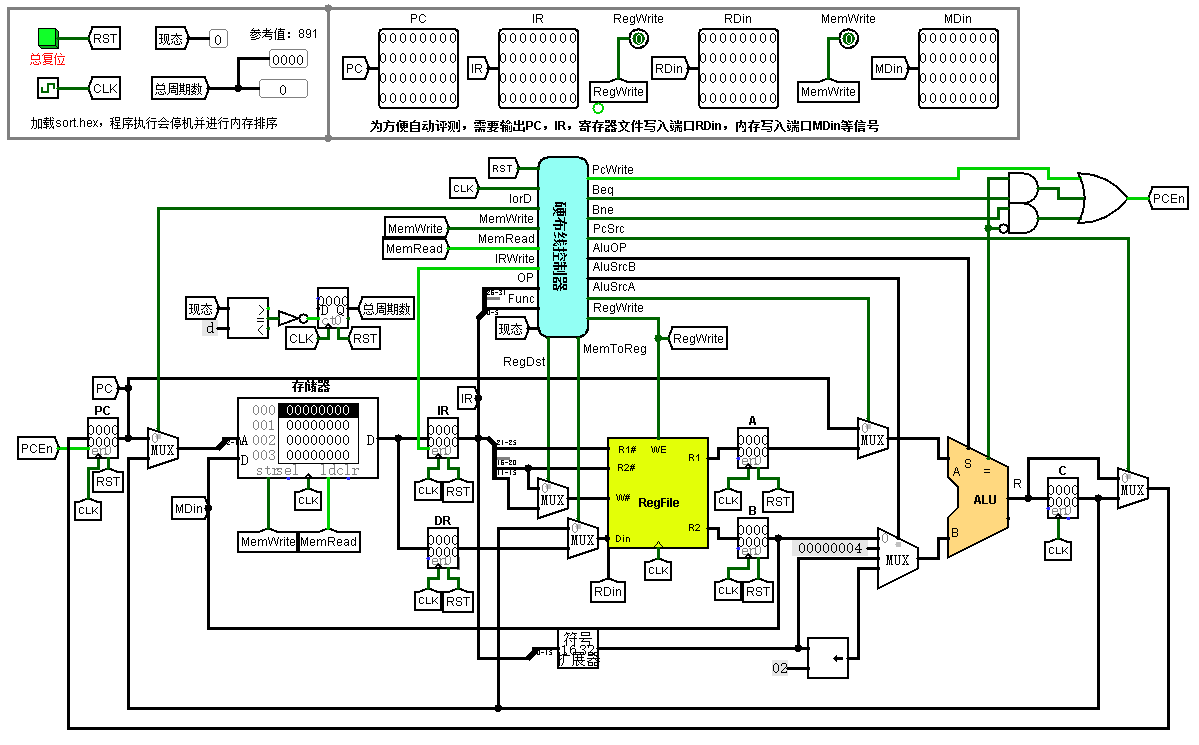


图 1.9 多周期硬布线CPU通路

#### 多周期硬布线控制器的方案设计

指令译码逻辑电路和单周期硬布线控制器中的一致，此处不再赘述。

ALU控制器逻辑中和多周期微程序控制器中的一致，此处同样不再赘述。

故多周期硬布线控制器的设计如下图。

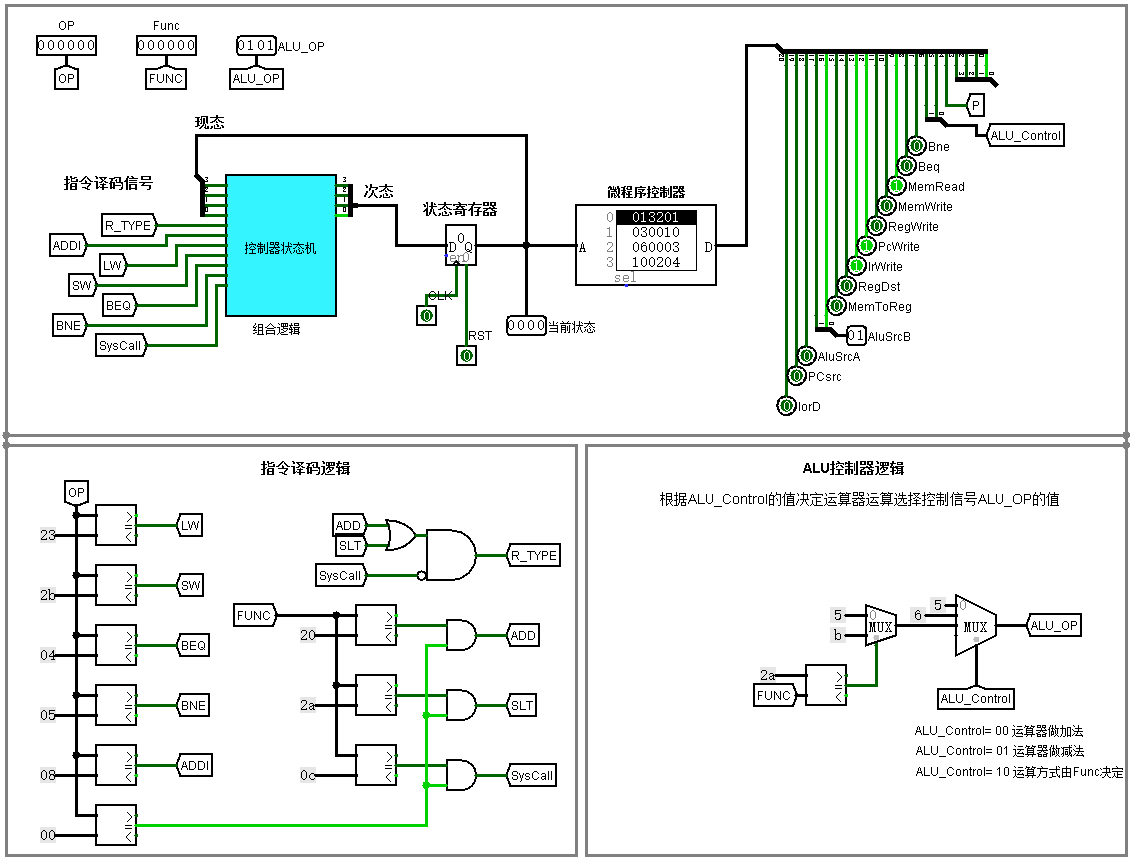


图 1.10 多周期硬布线控制器

#### 多周期硬布线有限状态机的方案设计

根据图1.5的微指令自动生成表可知各控制指令所在的地址，将现态、指令译码信号和对应次态控制指令的地址输入状态转移表，如下图所示。



图 1.11 硬布线状态转移表截图

则自动生成的各地址位的逻辑表达式见下表。

表 1.11 地址位号与对应组合逻辑表

|  |  |
| --- | --- |
| 位号 | 组合逻辑 |
| N0 | ~S3&~S2&~S1&~S0+~S3&~S2&~S1&S0&R\_Type+~S3&~S2&~S1&S0&SW+~S3&~S2&~S1&S0&BEQ+~S3&~S2&~S1&S0&SYSCALL+~S3&~S2&~S1&S0&ADDI+~S3&~S2&S1&~S0+S3&S2&~S1&S0 |
| N1 | ~S3&~S2&~S1&S0&R\_Type+~S3&~S2&~S1&S0&LW+~S3&~S2&~S1&S0&BNE+~S3&~S2&~S1&S0&ADDI+~S3&~S2&S1&~S0+~S3&S2&~S1&S0 |
| N2 | ~S3&~S2&~S1&S0&R\_Type+~S3&~S2&~S1&S0&SW+~S3&~S2&~S1&S0&SYSCALL+~S3&~S2&S1&S0+~S3&S2&~S1&S0+S3&~S2&S1&S0+S3&S2&~S1&S0 |
| N3 | ~S3&~S2&~S1&S0&BEQ+~S3&~S2&~S1&S0&BNE+~S3&~S2&~S1&S0&SYSCALL+~S3&~S2&~S1&S0&ADDI+~S3&S2&S1&S0+S3&~S2&S1&S0+S3&S2&~S1&S0 |

将对应组合逻辑复制粘贴至硬布线地址转移逻辑电路中自动生成电路，如下图所示：

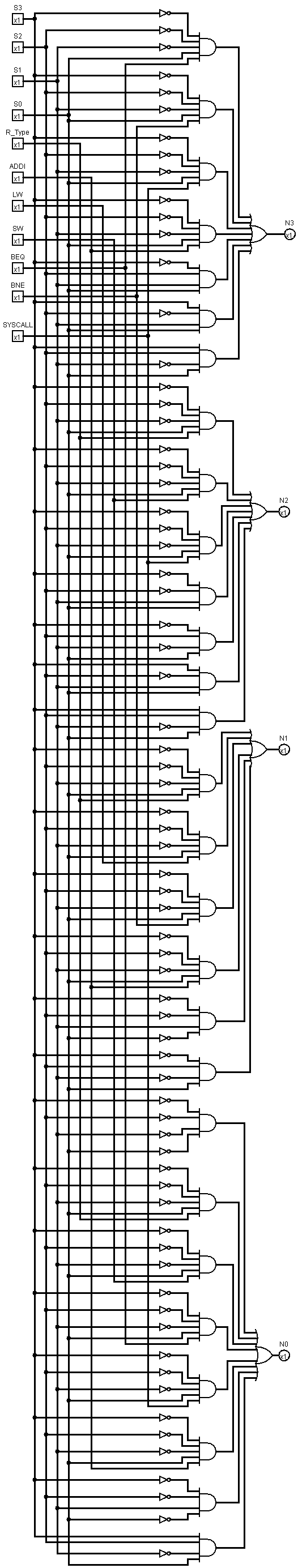


图 1.12 硬布线状态机电路

## 实验步骤

1. 观看MOOC，了解实验的操作步骤
2. 按照给出的CPU通路绘制单周期硬布线CPU通路
3. 根据对应指令译码信号设计单周期硬布线控制器
4. 按照给出的CPU通路绘制多周期微程序CPU通路
5. 在微指令自动生成表中填入指令对应的控制信号，将控制指令输入控制存储器，完成控制器的绘制
6. 将对应控制指令入口地址填入微程序地址转移逻辑自动生成表，得到地址位的逻辑表达式，在Logisim中自动生成电路
7. 按照给出的CPU通路绘制多周期硬布线CPU通路
8. 完成控制器的绘制
9. 将对应现态、指令译码信号和次态填入微程序地址转移逻辑自动生成表，得到地址位的逻辑表达式，在Logisim中自动生成电路
10. 测试三个CPU的结果，修改错误直至正确输出

## 故障与调试

### 控制器指令译码逻辑问题

**故障现象：** 单周期MIPS硬布线CPU运行无法结束，在地址01AH~021H之前循环但不结束。

**原因分析：** 由于PC一直在地址01AH~021H中循环而没有结束，问题可能存在于数据通路和指令译码逻辑电路中。

**解决方案：**检查CPU通路发现与参考通路一致，没有问题。检查指令译码逻辑电路，发现RTYPE类型的比较器的值为04H，与MIPS手册上的不符，修改为00H 后测试通过，故障产生的原因是在复制比较常量时忘记修改常数。

### 单周期数据存储器地址错误

**故障现象：**冒泡排序sort.hex程序执行完毕后，没有在指定的80号存储单元形成降序排列的数据，而是在200号存储单元形成降序排列数据，而且两个数据地址之差为4。

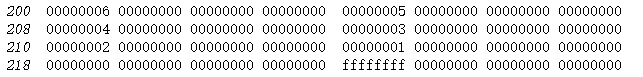


图 1.13 存储地址错误

**原因分析：**Logisim中RAM只支持一种访问模式，一次访问读出32位数据，直接给出字地址使得内存布局错误，如下图：

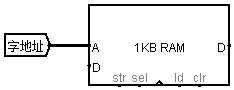


图 1.14 直接字地址访问

**解决方案：**字地址除以4即可得到正确的内存布局，即高两位补零作为字节地址送入存储器地址即可，如下图：

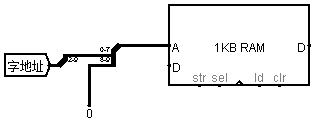


图 1.15 字节地址访问

## 测试与分析

### 单周期硬布线CPU执行

加载镜像sort.hex，启动时钟模拟。运行周期数如下图所示：

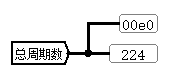


图 1.16 单周期硬布线CPU运行周期数

与参考周期数一致。查看内存，内存情况如下图所示：



图 1.17 内存第80号单元开始的存储情况

可知排序的结果正确。将文件上载至Educoder平台运行，运行结果如下图所示：



图 1.18 平台测试结果

综上所述，单周期硬布线CPU设计正确

### 多周期微程序CPU执行

加载镜像sort.hex，启动时钟模拟。运行周期数如下图所示：

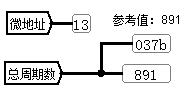


图 1.19 多周期微程序CPU运行周期数

与参考周期数一致。查看存储器，存储器情况如下图所示：



图 1.20 存储器第80号单元开始的存储情况

可知排序的结果正确。将文件上载至Educoder平台运行，运行结果如下图所示：



图 1.21 平台测试结果

\*注：输出结果过长的问题出自最后一个时钟周期，MDin的输出00000000之后出现一连串“ ”，如下图所示，由于MDin的最大输出仅为32位，不存在出现 的情况，应当认为是平台错误而非实验错误。



图 1.22 平台测试警告截图

综上所述，多周期微程序CPU设计正确。

### 多周期硬布线CPU执行

加载镜像sort.hex，启动时钟模拟。运行周期数如下图所示：

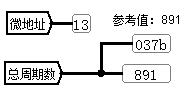


图 1.23 多周期硬布线CPU运行周期数

与参考周期数一致。查看存储器，存储器情况如下图所示：



图 1.24 存储器第80号单元开始的存储情况

可知排序的结果正确。将文件上载至Educoder平台运行，运行结果如下图所示：



图 1.25 平台测试结果

\*注：输出结果过长的问题出自最后一个时钟周期，MDin的输出00000000之后出现一连串“ ”，如下图所示，由于MDin的最大输出仅为32位，不存在出现 的情况，应当认为是平台错误而非实验错误。



图 1.26 平台测试警告截图

综上所述，多周期硬布线CPU设计正确。

# 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. 实现了对MIPS指令的解析
2. 实现了寄存器、存储器的读写
3. 实现了MIPS单周期、多周期硬布线和微程序CPU通路的绘制
4. 实现了MIPS单周期硬布线控制器的设计
5. 实现了MIPS多周期微程序控制器的设计
6. 实现了MIPS多周期微程序地址转移逻辑的的设计
7. 实现了MIPS多周期硬布线控制器的设计
8. 实现了MIPS多周期硬布线有限状态机的设计

## 实验心得

1. 巩固并熟练掌握了Logisim工具的使用方法。
2. 具体实践了CPU的工作原理，首次具体的了解到计算机底层的工作方式。
3. 对MIPS三类指令有了较为清晰的认识
4. 在MOOC的帮助下实验变得容易理解和上手。

# 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:** |