***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： | CS17xx |
| 学 号： | U2017xxxxx |
| 姓 名： | Xxx |
| 电 话： | 1xxxxxxxxxx |
| 邮 件： | [xxxxxxxxx@qq.com](mailto:xxxxxxxxx@qq.com) |
| 完成日期： | 2019-11-27 |



目 录

[1 CPU设计实验 2](#_Toc25653663)

[1.1 设计要求 2](#_Toc25653664)

[1.2 方案设计 3](#_Toc25653665)

[1.3 实验步骤 4](#_Toc25653666)

[1.4 故障与调试 4](#_Toc25653667)

[1.5 测试与分析 5](#_Toc25653668)

[2 总结与心得 7](#_Toc25653669)

[2.1 实验总结 7](#_Toc25653670)

[2.2 实验心得 7](#_Toc25653671)

[参考文献 8](#_Toc25653672)

# CPU设计实验

## 设计要求

1. 在logisim平台利用已给出的组件构建一个32位MIPS 单周期CPU，该CPU支持如表1. 1列出的核心指令集中的8条指令。要求绘制单周期MIPS CPU数据通路、实现单周期硬布线控制器，并最终能在完成的CPU上运行冒泡排序的测试程序sort.hex且获得正确的输出结果。
2. 在logisim平台利用已给出的组件构建一个32位MIPS 多周期CPU，该CPU支持如表1. 1列出的核心指令集中的8条指令。要求绘制多周期MIPS CPU数据通路，以两种方式实现控制器，即分别实现微程序控制器和硬布线控制器。并最终能在完成的CPU上运行冒泡排序的测试程序sort.hex且获得正确的输出结果。

表1. 8条核心指令集

|  |  |  |
| --- | --- | --- |
| # | MIPS指令 | 格式 |
| 1 | **add** $rd,$rs,$rt | R[$rd]←R[$rs]+R[$rt] 溢出时产生异常，且不修改R[$rd] |
| 2 | **slt** $rd,$rs,$rt | R[$rd]←R[$rs]<R[$rt] 小于置1，有符号比较 |
| 3 | **addi** $rt,$rs,imm | R[$rt]←R[$rs]+SignExt16b(imm) 溢出产生异常 |
| 4 | **lw** $rt,imm($rs) | R[$rt]←Mem4B(R[$rs]+SignExt16b(imm)) |
| 5 | **sw** $rt,imm($rs) | Mem4B(R[$rs]+SignExt16b(imm))←R[$rt] |
| 6 | **beq** $rs,$rt,imm | if(R[$rs] = R[$rt]) PC ← PC + SignExt18b({imm, 00}) |
| 7 | **bne** $rs,$rt,imm | if(R[$rs] != R[$rt]) PC ← PC + SignExt18b({imm, 00}) |
| 8 | **syscall** | 系统调用，这里用于停机 |

## 方案设计

### 设计单周期MIPS CPU数据通路

待实现的8条指令，syscall停机指令可直接通过连接PC寄存器的使能端将其禁用达到停机的目的，而针对其他7条指令则另需设计数据通路。取指令阶段也需设计数据通路。

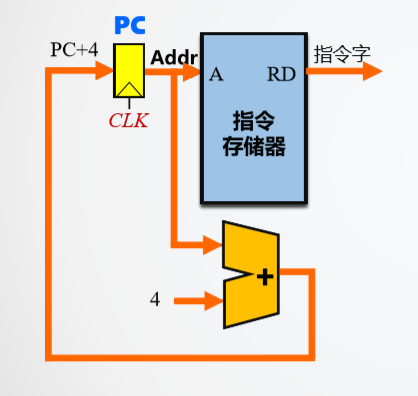


图1. 1取指令数据通路

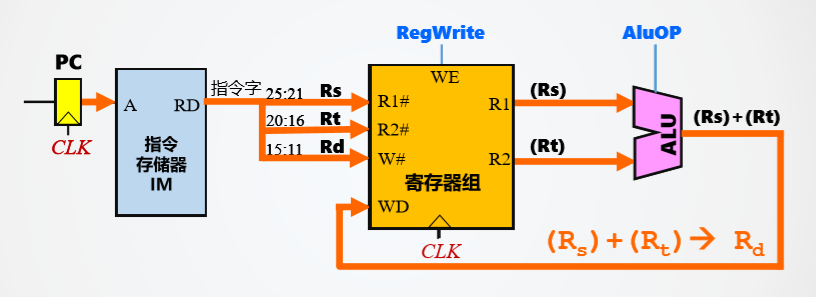


图1. 2 R型运算指令（add和slt）数据通路

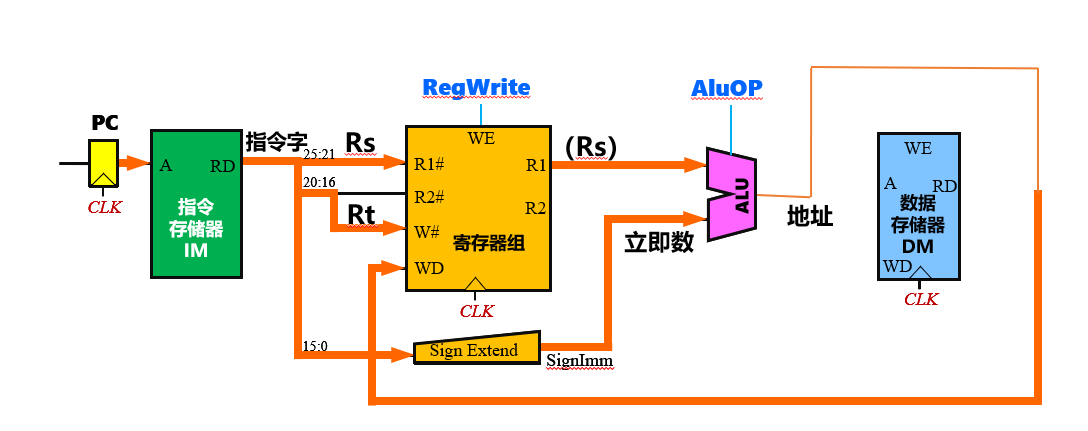


图1. 3 addi指令数据通路

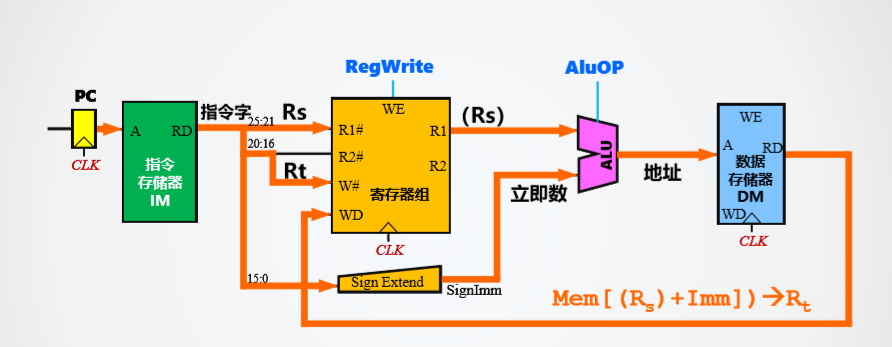


图1. 4 lw指令数据通路

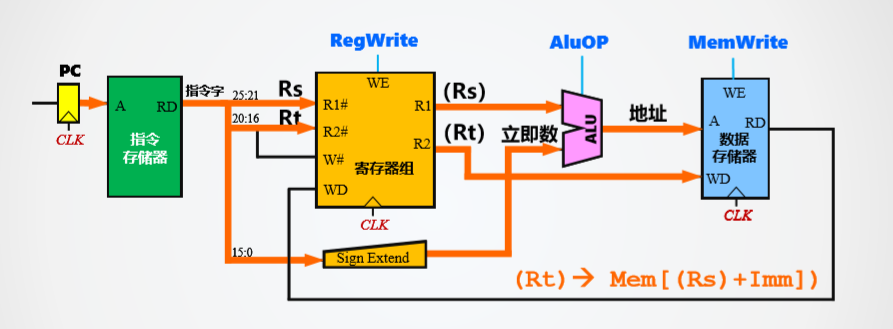


图1. 5 sw指令数据通路

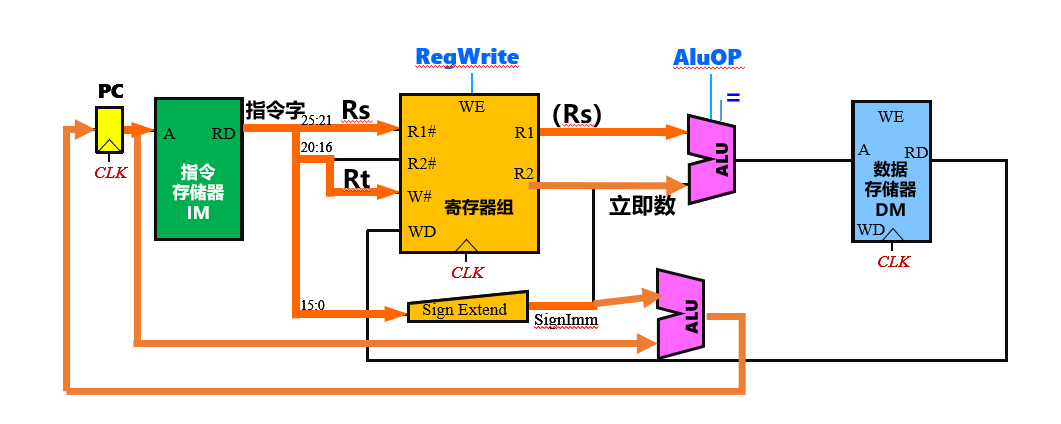


图1. 6 beq和bne指令数据通路

将已设计的不同指令的数据通路进行综合，在有多个输入来源的地方增加数据选择器MUX，且将控制信号作为数据选择器的选择端，构建综合数据通路（如图1. 7所示）。

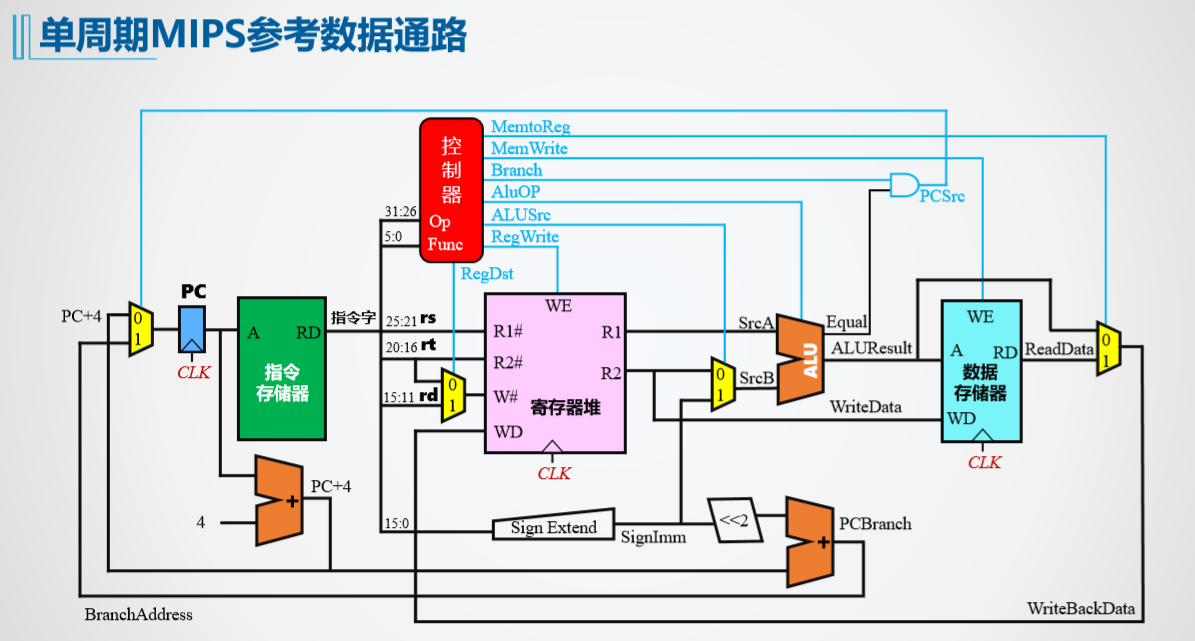


图1. 单周期MIPS参考数据通路示意图

### 设计单周期MIPS CPU硬布线控制器

单周期MIPS CPU硬布线控制器输入信号功能为提供CPU运行过程中的各种控制信号。已封装好的控制器如图1. 8所示。输入为指令的Op字段和Func字段 ，输出为各种控制信号。分析各指令各自执行时的数据通路，得到各指令分别需要何种控制信号。完成后绘制控制信号功能表如表1. 2所示。

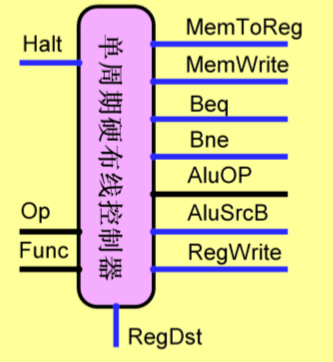


图1. 8 单周期CPU硬布线控制器封装图

表1. 2 控制信号功能说明表

|  |  |  |  |
| --- | --- | --- | --- |
| # | 控制信号 | 信号说明 | 产生条件 |
| 1 | MemToReg | 写入寄存器的数据来自寄存器 | lw指令 |
| 2 | MemWrite | 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指令，sw指令，addi |
| 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 | syscall指令 |

1. 指令译码逻辑

指令译码逻辑将指令的操作码字段转换为对应的运算信号，如addi信号，lw信号，sw信号等。输入为Op字段和Func字段,输出共6个，R\_TYPE、ADDI、LW、SW、BEQ、BNE、SysCall。

查询实验包内的《MIPS32指令手册》得到LW、SW、BEQ、BNE、ADDI指令的OP字段如表1. 3所示。

表1. 3 部分指令OP字段值

|  |  |
| --- | --- |
| 指令 | OP字段 |
| LW | 0x23 |
| SW | 0x2b |
| BEQ | 0x04 |
| BNE | 0x05 |
| ADDI | 0x08 |

只需使用比较器将OP与各指令的OP字段值进行比较便可得到以上指令对应的控制信号。

而对于信号R\_TYPE,它是R型运算指令的控制信号，对于本次设计而言，共3条R型指令，分别为ADD、SLT、SysCall。其中ADD和SLT指令为R型运算指令。故R\_TYPE在SysCall对应的指令无效，而ADD对应的控制信号或SLT对应的控制信号有效时有效。即R\_TYPE = ( ADD + SLT ) & ~SysCall 。故需先得到ADD、SLT和SysCall信号。R型指令的OP字段为0，具体功能需看Func字段，故需再次查阅实验包内的《MIPS32指令手册》得到ADD、SLT、SysCall指令的Func字段值，如表1. 4所示。

表1. 4 部分R型指令Func字段值

|  |  |
| --- | --- |
| 指令 | Func字段 |
| ADD | 0x20 |
| SLT | 0x2a |
| SysCall | 0x0c |

只需将FUNC和各指令对应字段比较相等的信号与OP和0比较相等的信号作与运算即可得到相应的控制信号ADD、SLT和SysCall。

这样，我们就得到了R\_TYPE、ADDI、LW、SW、BEQ、BNE、SysCall这6个指令译码信号。

1. ALU控制逻辑

ALU控制逻辑负责控制运算器的功能，输出信号ALU\_OP送入运算器ALU的功能选择端，输入仍为指令的OP字段和FUNC字段。由于运算器只进行加法和比较大小两种运算，且只有在运行SLT指令时才需要进行比较大小运算，故使用SLT指令时使ALU\_OP控制信号控制ALU进行比较大小运算其余时候都作加法运算即可。查看ALU内部逻辑（如图1. 9所示）可知加法运算操作码为0x5，比较大小运算操作码为0xb。将0x5和0xb作为二路选择器的输入端，SLT信号作为选择端即可。

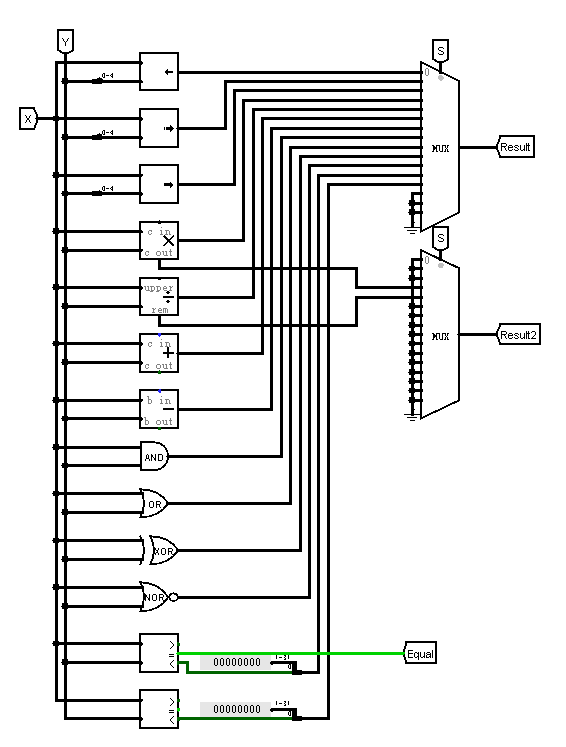


图1. 9 ALU内部逻辑

1. 完善控制信号逻辑

根据刚刚得到的6个指令译码信号，R\_TYPE、ADDI、LW、SW、BEQ、BNE、SysCall，进一步实现所有控制信号逻辑。

根据表1. 2 控制信号功能说明表可知哪些信号在执行哪些指令时起作用，将对应的指令译码信号进行逻辑或即可得到对应控制信号。

需额外指出的是，寄存器写回操作包括R\_TYPE指令、ADDI指令和LW指令。

### 设计多周期MIPS CPU数据通路

在多周期MIPS CPU中，不再区分指令存储器与数据存储器，各功能部件也只保留一个（如存储器、ALU），分时使用各功能部件，在时间上错开。且在功能部件的输出端增加相应寄存器锁存数据方便时序控制。将指令执行的过程细分成若干段，每条指令的执行分为“取指->译码->执行”三个阶段，取指和译码阶段是所有指令的必经阶段，执行阶段具体不同的指令占用的时钟周期不尽相同。不同的时钟周期下也会产生相应的控制信号。多周期CPU的整体架构与单周期CPU的不同之处具体如下

1. 不再区分指令存储器与数据存储器，指令和数据保存在同一个存储器中，取指令与存数据操作分时使用存储器；
2. 部分功能部件，如存储器、ALU等可以在一条指令执行过程的不同时钟周期内多次分时使用；
3. 主要功能部件输出端都增加了寄存器并引入时序控制，在后续时钟周期中要用到的所有数据必须锁存在相应的寄存器中。具体增加了数据寄存器DR，用于存放从存储器读取的数据；增加了指令存储器IR，用于存放从存储器取出的指令，增加了三个存储器A、B、C分别用于保存RegiFile和ALU的输出；
4. PC作为指令计数器，由于不同指令时钟周期数不同，因此PC不再仅由时钟周期控制，而是增加了专门的写操作控制信号；
5. 由于取消了单周期CPU中加法器的使用，PC+4的操作和地址转移指令中PC地址加上立即数的操作都由ALU来完成，所以ALU的B输入端又增加了两个输入，共4个输入，应采用4路选择器选择输出。A端也多了一个PC值的输入，增加一个2路选择器选择输出。
6. 由于存储器复用，除了原有的PC值外，存储器的输入端口还需将经过ALU计算得到的地址送入存储器地址输入端。
7. 由于新引入了一些寄存器，需引入相应的控制信号。
8. 多周期CPU的停机功能无需halt信号，借由状态的变迁来实现（一直循环在某个状态达到停机的目的）。

根据上述设计思想，逐条指令分析其执行过程，绘制单条指令数据通路，最后对数据通路进行综合，绘制多周期CPU数据通路示意图，如图1. 10所示。

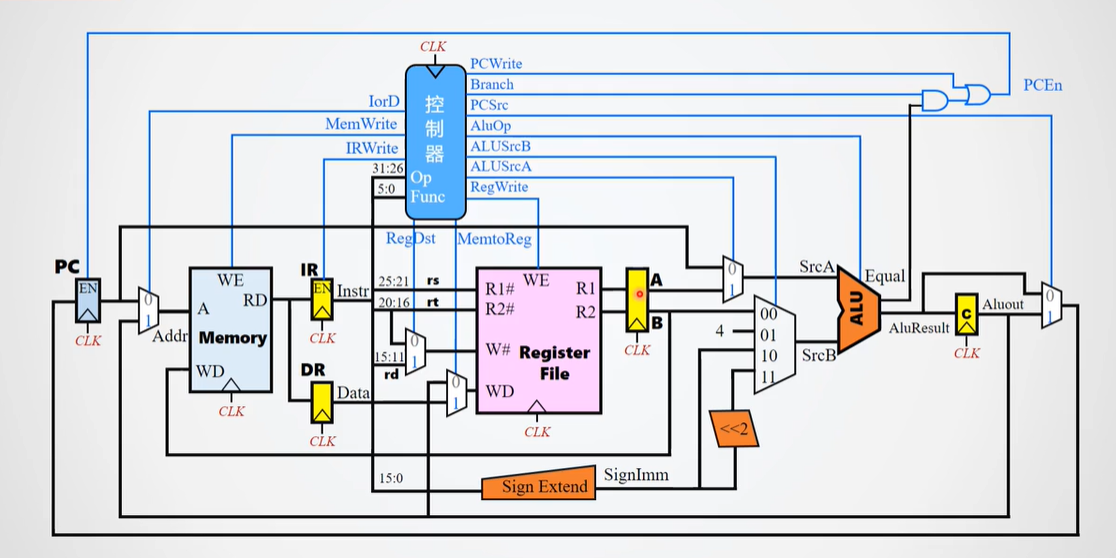


图1. 10 多周期CPU数据通路示意图

### 设计多周期MIPS CPU微程序控制器

多周期CPU控制下，不同指令对应不同的时钟周期数，每一条指令的执行分为“取指->译码->执行”三个阶段，其中取指和译码两个阶段所有指令对应的数据通路相同控制信号也相同，执行阶段由于不同指令所进行的微操作不同所需时间也不同。

对于微程序控制器，每条指令由若干条微指令组成，每条微指令对应一个微操作（如IR<-(MEM[PC])），一个微操作的顺利执行需要一系列可以并发产生的控制信号进行控制。且每条微指令对应一个时钟周期，在每一个时钟周期内这些并发的控制信号控制完成此条微指令。一系列微指令共同完成一条指令的全部操作。故接下来分析取指、译码阶段以及每条指令执行阶段的具体执行过程，每条微指令执行过程（即每个时钟周期）进行何种微操作。并根据每一微指令划分时钟周期（每一时钟周期对应状态图中的每一个状态），根据该微指令执行过程的数据通路给出所需要的控制信号，进而通过控制信号生成微指令。

（1）首先给出取指、译码阶段以及每条指令的执行流程表，如下各表所示：

表1. 取指、译码阶段操作流程

|  |  |
| --- | --- |
| 指令阶段 | 操作流程 |
| 取指令 | IR <- (Mem [PC])  PC <- (PC)+4 |
| 译码及取操作数 | A <- (R[IR[25:21]])  B <- (R[IR[20:16]])  C <- (PC) + (S-EXT(IR[15:0])<<2) |

表1. add指令执行操作流程

|  |  |
| --- | --- |
| 指令阶段 | 操作流程 |
| 加运算 | C <- (A) + (B) |
| 写回 | R[IR[15:11]] <- (C) |

表1. slt指令执行操作流程

|  |  |
| --- | --- |
| 指令阶段 | 操作流程 |
| 比较运算 | C <- ((A)<(B)) |
| 写回 | R[IR[15:11]] <- (C) |

表1. lw指令执行操作流程

|  |  |
| --- | --- |
| 指令阶段 | 操作流程 |
| 计算地址 | C <- (A) + S-EXT(IR[15:0]) |
| 访存 | DR <- (Mem[PC]) |
| 写回 | R[IR[20:16]] <- (DR) |

表1. sw指令执行操作流程

|  |  |
| --- | --- |
| 指令阶段 | 操作流程 |
| 计算地址 | C <- (A) + S-EXT(IR[15:0]) |
| 访存 | DR <- (Mem[PC]) |

表1. beq指令执行操作流程

|  |  |
| --- | --- |
| 指令阶段 | 操作流程 |
| 送目标地址 | if(A==B) PC <- (C) |

表1. bne指令执行操作流程

|  |  |
| --- | --- |
| 指令阶段 | 操作流程 |
| 送目标地址 | if(A!=B) PC <- (C) |

表1. addi指令执行操作流程

|  |  |
| --- | --- |
| 指令阶段 | 操作流程 |
| 加运算 | C <- (A) + S-EXT(IR[15:0]) |
| 写回 | R[IR[20:16]] <- (C) |

表1. syscall指令执行操作流程

|  |  |
| --- | --- |
| 指令阶段 | 操作流程 |
| 空操作，停机 | 锁住PC |

（2）根据以上操作流程表对每条指令划分执行阶段，每个阶段对应一个时钟周期（一个状态），每个时钟周期对应一条微指令。构建指令状态变换图，如图1. 11所示。

并根据状态图的操作和指令执行时的数据通路分析给出不同状态所需的控制信号。其中微指令地址即为状态编号。微指令及对应控制信号表如表1. 14所示。

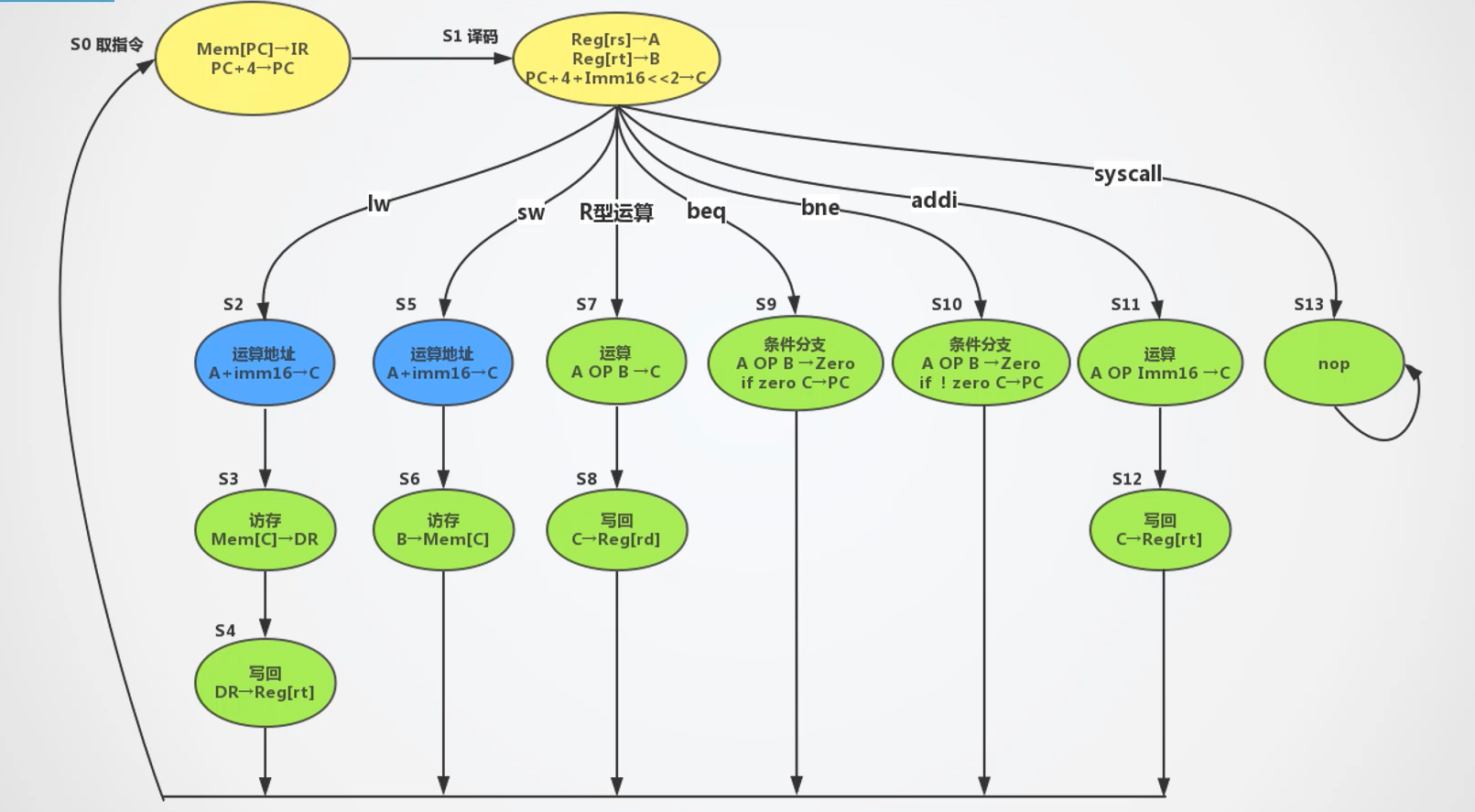


图1. 11 指令状态变换图

表1. 14 微指令及对应控制信号表



### 设计多周期MIPS CPU硬布线控制器

设计硬布线控制器的关键是实现一输入为现态和指令译码信号、输出为次态有限状态机。而控制信号则只与该有限状态机的次态有关。首先需分析各指令画出所有指令的指令周期流程图，明确各节拍控制信号，构建指令状态变换图。而这些工作已在1.2.4完成。不同的状态需要不同的控制信号，形成相应的数据通路，而不同的控制信号仅与状态有关（输出仅与状态有关，为Moore型电路）。

根据1.2.4中的指令状态变换图，共13个状态，首先需要一4位的状态寄存器记录状态编号。状态寄存器的输出为FSM的现态，输入为FSM的次态。当时钟到来时，现态更新为上一个次态。状态的转换由状态机组合逻辑实现。

状态机的输入为指令译码信号、反馈信号和现态，输出为次态。状态机为纯组合逻辑，可利用真值表自动生成。

状态机生成后可根据状态机的现态生成相应的控制信号（硬布线控制器组合逻辑）。正如前所述，不同的状态对应相应的控制信号。这一部分也为组合逻辑。

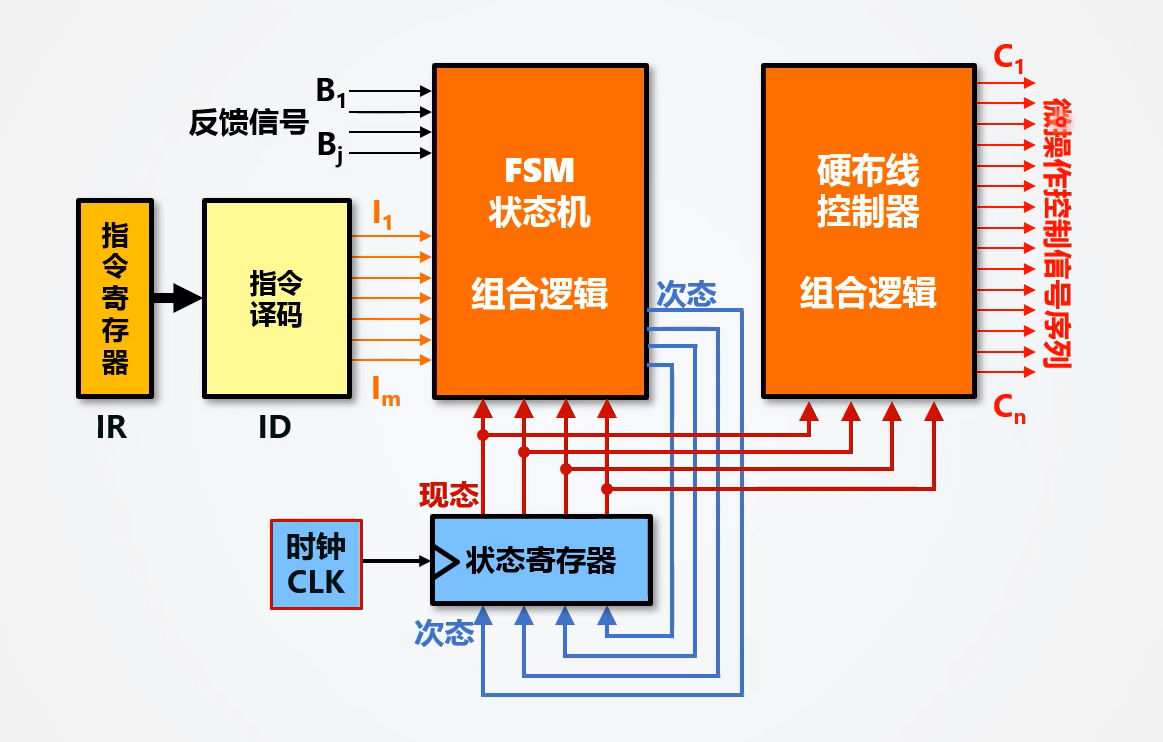


图1. 硬布线控制器示意图

## 实验步骤

### 构建单周期CPU数据通路

所给组件有PC、IMEM（指令寄存器）、Regfile（通用寄存器组）、ALU（运算器）、DMEM（数据寄存器）、Controller（控制器），其中控制器只是给出封装，待我们自己去实现，其他组件直接使用即可。 按照图1. 7中的数据通路示意图，根据上述已给出的器件进行布线。具体布线时，除按照示意图连接外，还需实现以下逻辑。

1. 示意图中Sign Extend部分使用16位to 32位扩展器进行有符号数扩展。
2. 利用ALU的equal信号，生成branch=beq\*equal + bne\*~equal分支信号，此branch信号即对应示意图中的Branch信号，将其连接至PC寄存器输入端的数据选择器使能端。
3. 将halt停机信号送至PC寄存器使能端，达到需要停机时使PC寄存器禁言的目的，以实现停机操作。
4. 将复位信号连接至数据寄存器和指令寄存器。
5. PC信号为32位，指令存储器地址输入端为10位，又因为PC为字节地址，而指令存储器输入为字地址，故需用分离器取PC的第2-11位送入指令存储器地址输入端。数据寄存器也有类似的问题。

完成布线后如图1. 13所示。

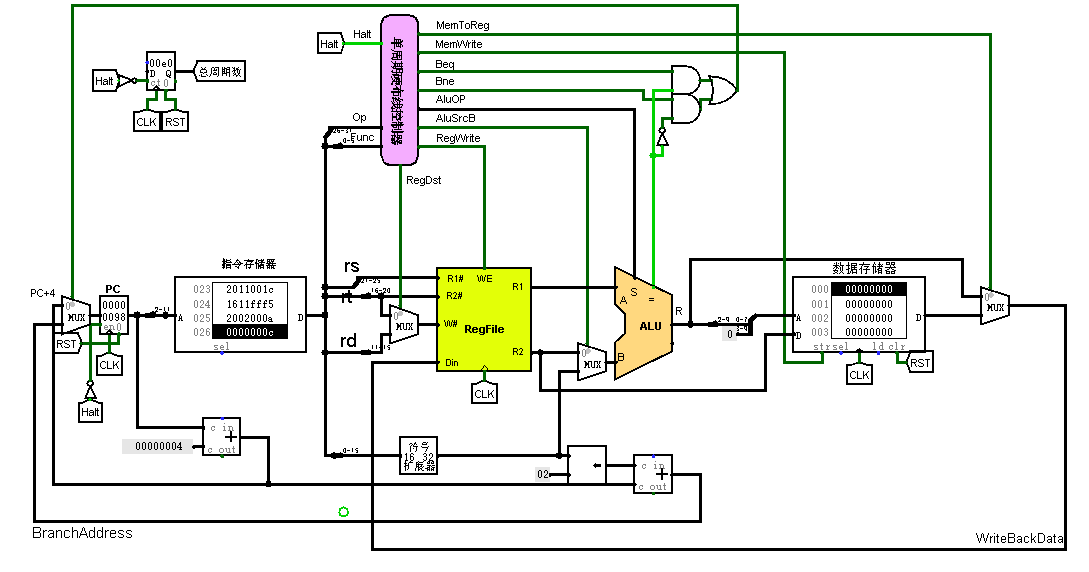


图1. 13 单周期MIPS CPU数据通路布线图

### 构建单周期CPU硬布线控制器

按照1.2.2的设计方案，先得到指令译码信号和ALU控制信号，再根据指令译码信号生成最终的控制信号。

完成布线后如图1. 14图1. 15和图1. 16所示。

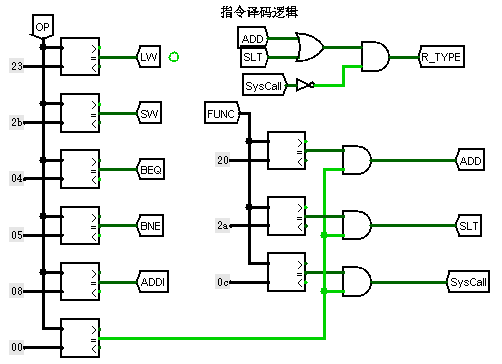


图1. 14 指令译码逻辑布线图

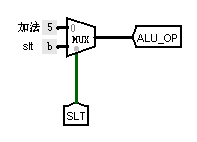


图1. 15 ALU控制器逻辑布线图

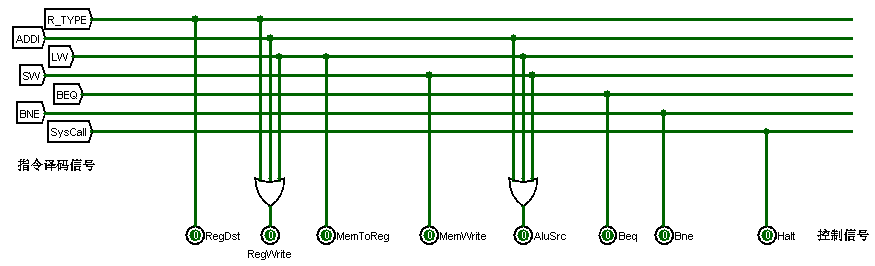


图1. 16 控制信号逻辑布线图

### 构建多周期CPU数据通路

根据多周期CPU数据通路示意图，利用如所示组件，在logisim中完成多周期CPU数据通路布线图。

需额外注意之处：

1. 示意图中Sign Extend部分使用16位to 32位扩展器进行有符号数扩展。
2. 利用ALU的equal信号，生成branch=beq\*equal + bne\*~equal分支信号，此branch信号即对应示意图中的Branch信号，将其连接至PC寄存器输入端的数据选择器使能端。
3. PC信号为32位，存储器地址输入端为10位，又因为PC为字节地址，而存储器输入为字地址，故需用分离器取PC的第2-11位送入存储器地址输入端。
4. 将复位信号连接至各寄存器。

完成布线图如图1. 17所示。

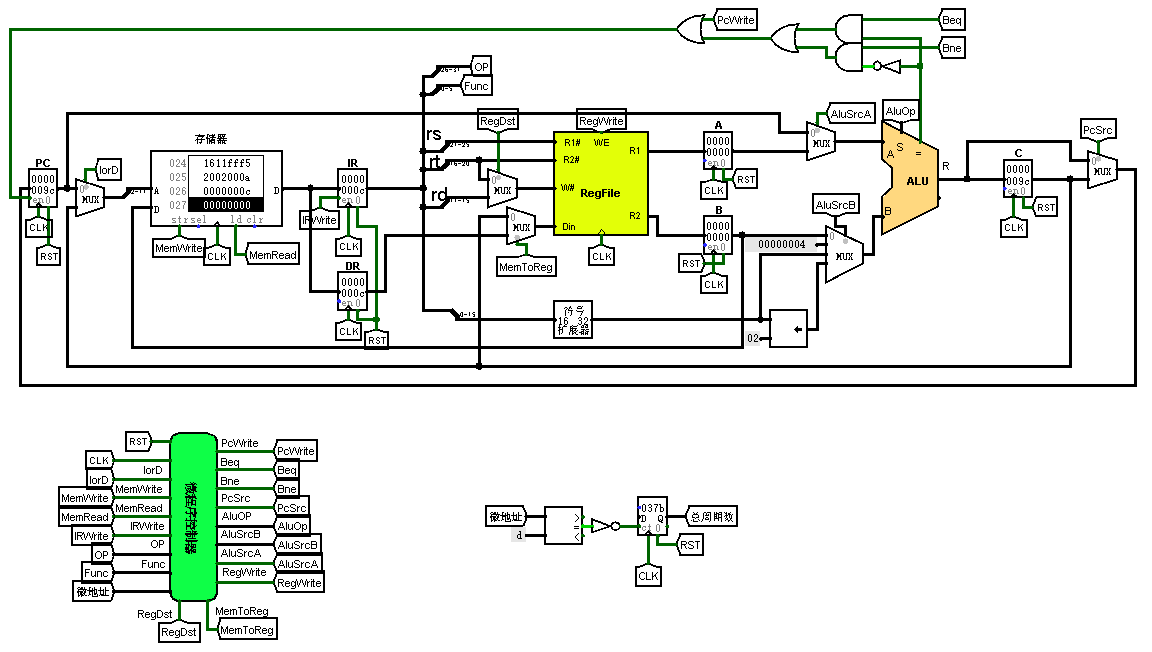


图1. 17 多周期CPU数据通路布线图

### 构建多周期CPU微程序控制器

（1）设计地址转移逻辑

地址转移逻辑，输入为7个指令译码信号，输出为微程序入口地址。该地址转移逻辑会根据不同的指令译码信号生成相应的微程序入口地址。

根据1.2.4中的微指令控制信号表填写excel表格微程序地址转移逻辑自动生成表，自动生成微程序地址逻辑表达式，如下所示。



表1. 15 微程序地址转移逻辑自动生成表



表1. 16 微程序地址转移逻辑自动生成表达式

将生成的逻辑表达式填入Logisim中，利用Logisim自动生成电路功能得到微程序地址转移逻辑，如图1. 18所示。

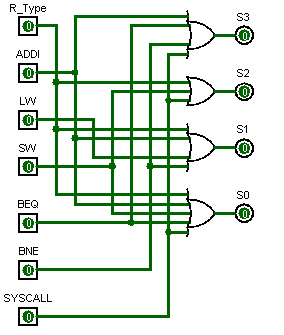


图1. 18 微程序地址转移逻辑

（2）设计指令译码信号与ALU控制信号

地址转移逻辑的输入为7个指令译码信号，指令译码信号的得到与单周期MIPS CPU中一致。但ALU控制信号发生了一些变化，需重新设计。根据ALU\_Control的值决定运算器运算选择控制信号ALU\_OP的值，ALU\_Control为00时做加法，为01时做减法，为10时由Func决定。故采用4路选择器。第一个输入端为ALU的加法操作码，第二个输入端为ALU的减法操作码。为了不使有输入端存在悬空，第四个输入置为加法的操作码5（默认为加法操作）。第三个输入取决于FUNC字段，故将FUNC字段与SLT的FUNC字段值2a用比较器进行比较后控制一二路选择器输出5（作加法运算）和b（作比较大小运算）。当比较器输出相等时，二路选择器输出b给第三个输入，表示此时应作比较大小运算。其他时候默认作加法运算。布线如图1. 19所示。

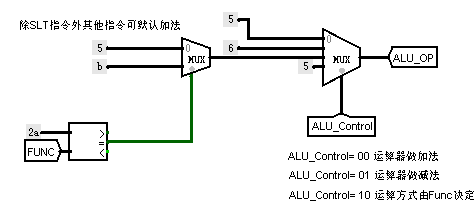


图1. 19 ALU控制信号布线图

（3）构建微程序

根据1.2.4中的指令状态变换图和微指令控制信号表填写Excel表格微指令自动生成表。如表1. 17所示。

每条指令执行阶段的最后一条微指令的下地址字段为0，非最后一条微指令则为下一条微指令的地址。取指微指令的下地址字段为译码微指令的地址，译码微指令的则为0（表示需要根据当前状态和控制信号选择进行跳转的下一状态）。syscall微指令的下地址字段为自身，一直在该状态循环，即实现停机操作。

表1. 17 微指令自动生成表



将得到的微指令复制生成一个txt文本（如图1. 20所示），将其加载到微程序控制器的控制存储器组件。即完成微程序控制器的设计。如图1. 21所示。

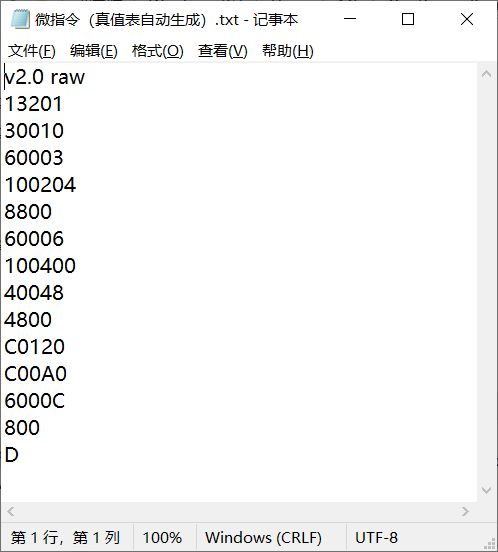


图1. 导出的微指令

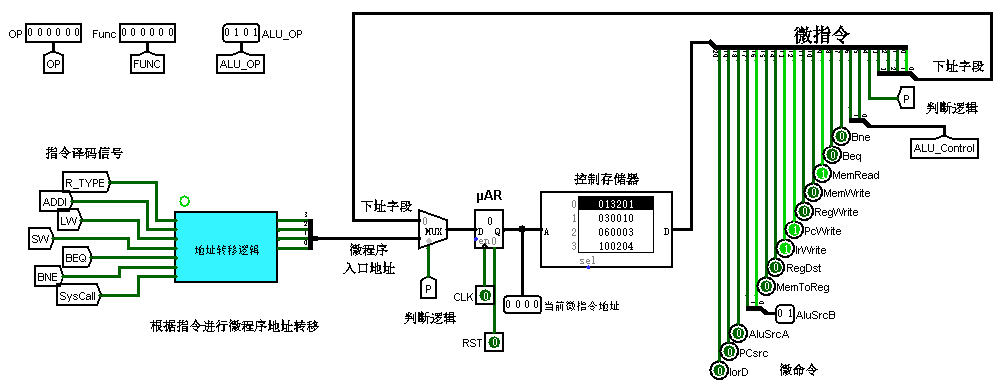


图1. 21 微程序控制器布线图

### 构建多周期CPU硬布线控制器

（1）构建指令译码信号、ALU控制信号

指令译码信号与ALU控制信号的构建与微程序控制器中的完全相同。

（2）构建状态机逻辑

状态机的输入为指令译码逻辑和现态，输出为次态。根据指令状态变换图和微指令控制信号表填写Excel表格硬布线控制器状态转换逻辑自动生成表。如表1. 18所示。

表1. 18 硬布线控制器状态转换逻辑自动生成表



表1. 19 硬布线控制器状态转换逻辑自动生成表达式



根据自动生成的表达式填入logisim中用自动生成电路功能生成状态机FSM电路。如图1. 22所示。

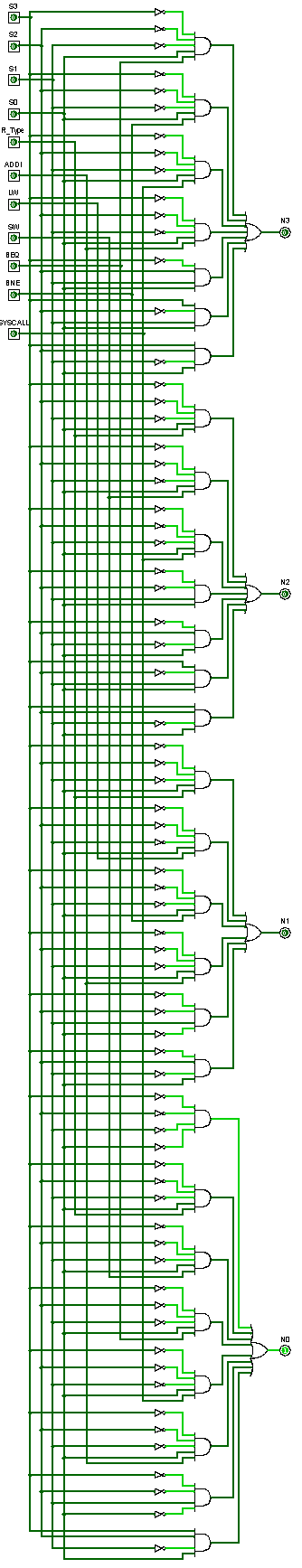


图1. 22 状态机FSM电路

（3）硬布线控制器组合逻辑生成

还须实现一个由现态到控制信号的组合逻辑，由于硬布线控制器的状态寄存器本质上等同于微程序控制器中的微地址寄存器，状态字等价于微地址，故在此复用微程序控制器中的控制存储器，将状态寄存器的输出作为控制存储器的输入，便可利用控制存储器生成各控制信号。

自此，便完成了硬布线控制器的构建（如图1. 23所示）。

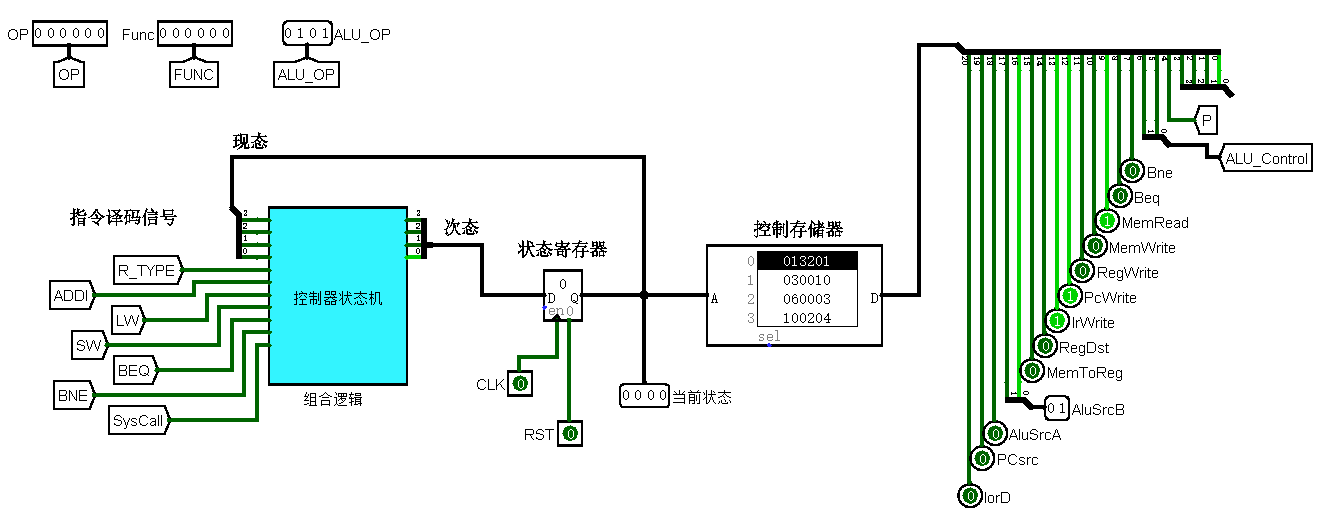


图1. 23 硬布线控制器布线图

## 故障与调试

### 故障1

**故障现象：**执行完最后一条指令后，电路一直震荡。

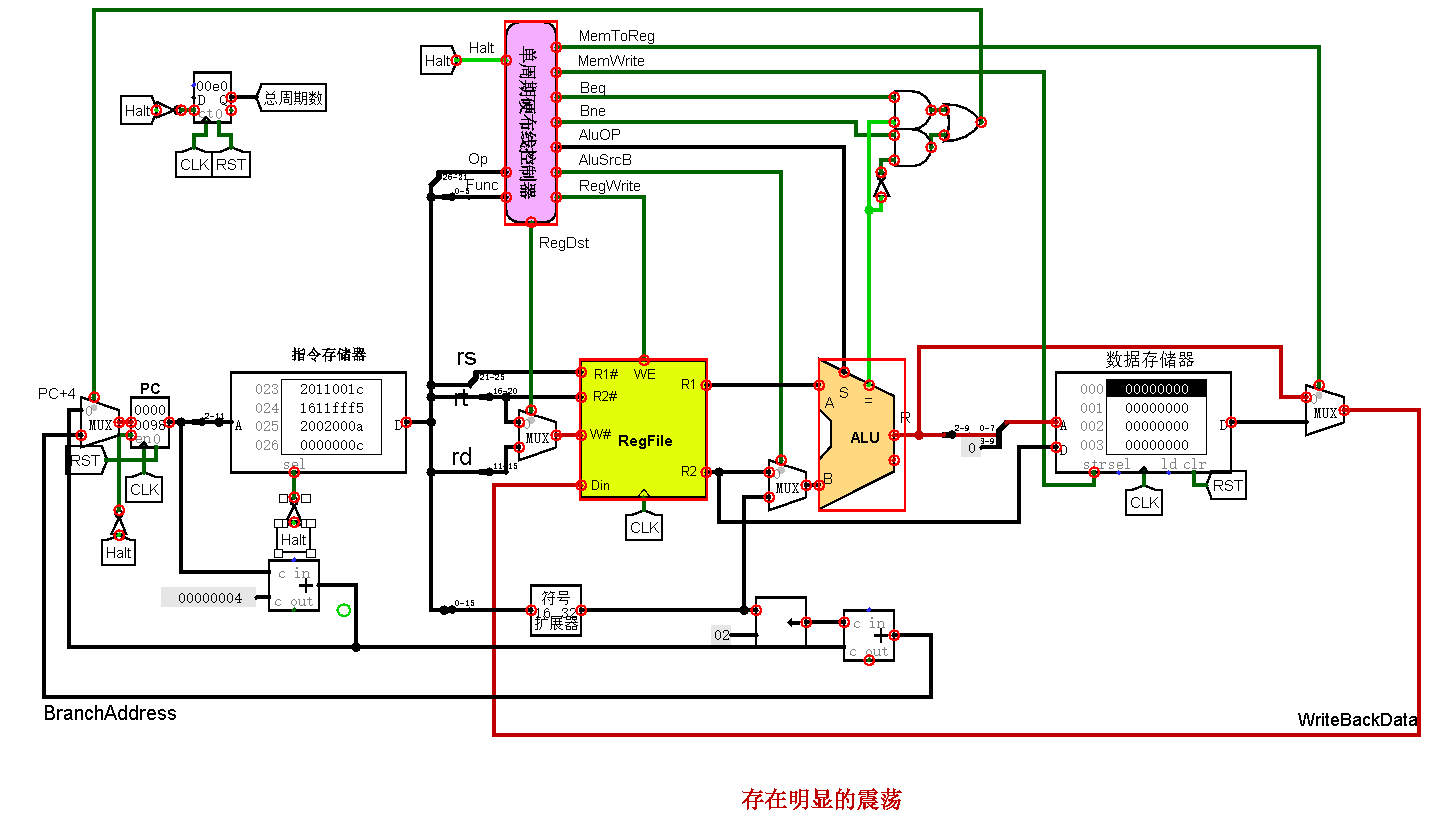


图1. 故障1图

**原因分析：**如图1. 24所示，运行完最后一条指令后电路开始震荡。分析是因为syscall指令的实现方式有问题。将halt停机信号接到了指令寄存器的使能端，使得停机时指令寄存器被禁用，导致指令寄存器输出不确定，造成电路震荡。

**解决方案：**去掉接到指令寄存器使能端的停机信号。

### 故障2

**故障现象：**进行sort.hex测试后结果异常。



图1. 故障2图

**原因分析：**如图1. 25所示，进行sort.hex测试后结果与预期不符，且为升序，但有数据写入。猜测是MemWrite控制信号出了问题。

**解决方案：**后来重新检查了下数据通路，发现MemWrite信号和MemToReg信号连岔了。将其调换即可。

## 测试与分析

### 单周期CPU执行sort.hex测试程序

1. 加载数据存储器内容，发现在80号单元处开始有6,5,4,3,2,1,ffff的有符号降序数据，如图1. 26所示，对比知结果正确。



图1. 单周期执行sort.hex数据存储器内容

1. 执行完毕后，系统停机，时钟周期数为224，如图1. 27所示，对比知结果正确。

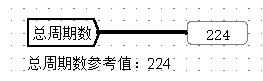


图1. 单周期执行sort.hex所需时钟周期数

### 多周期CPU（微程序）执行sort.hex测试程序

1. 加载存储器内容，发现在80号单元处开始有6,5,4,3,2,1,ffff的有符号降序数据，如图1. 28所示，对比知结果正确。

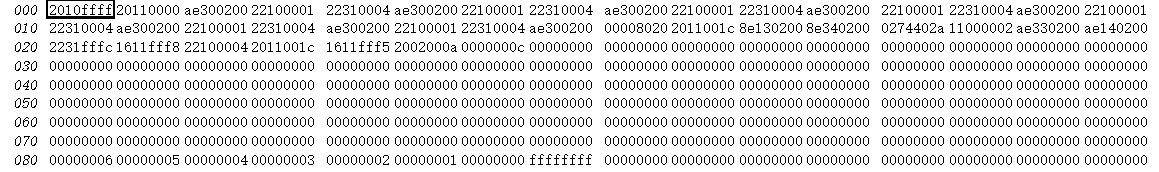


图1. 多周期CPU（微程序）执行sort.hex后存储器内容

1. 时钟周期，执行完毕后，系统停机，时钟周期数为891，如图1. 29所示，对比知结果正确。

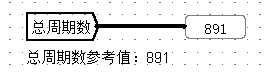


图1. 多周期CPU（微程序）执行sort.hex所需时钟周期数

### 多周期CPU（硬布线）执行sort.hex测试程序

1. 加载存储器内容，发现在80号单元处开始有6,5,4,3,2,1,ffff的有符号降序数据，如图1. 30所示，对比知结果正确。

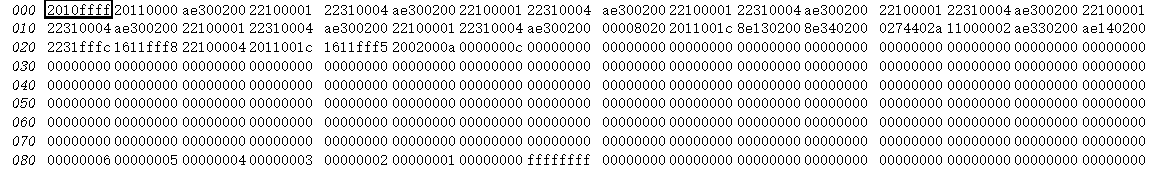


图1. 多周期CPU（硬布线）执行sort.hex后存储器内容

1. 时钟周期，执行完毕后，系统停机，时钟周期数为891，如图1. 31所示，对比知结果正确。

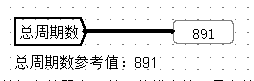


图1. 多周期CPU（硬布线）执行sort.hex所需时钟周期数

# 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. 设计、实现了单周期MIPS CPU数据通路；
2. 设计、实现了单周期MIPS CPU硬布线控制器指令译码逻辑；
3. 设计、实现了单周期MIPS CPU硬布线控制器ALU控制逻辑；
4. 设计、实现了单周期MIPS CPU硬布线控制器控制信号逻辑；
5. 完成了单周期MIPS CPU数据通路综合，最终的CPU能完整支持MIPS核心指令集。
6. 设计、实现了多周期MIPS CPU数据通路；
7. 设计了多周期MIPS CPU指令执行状态变换图；
8. 设计了多周期MIPS CPU指令控制信号；
9. 设计、实现了多周期MIPS CPU微程序地址转移逻辑；
10. 设计、实现了多周期MIPS CPU微程序控制器指令译码逻辑；
11. 设计、实现了多周期MIPS CPU微程序控制器ALU控制逻辑；
12. 设计、实现了多周期MIPS CPU微程序控制器控制信号逻辑；
13. 设计、实现了多周期MIPS CPU硬布线控制器状态机；
14. 完成了多周期MIPS CPU数据通路综合，最终的CPU能完整支持MIPS核心指令集。

## 实验心得

1. 更加了解了对logisim的使用，学会了logisim的一些高级用法，如自动生成电路。学会了使用logisim的一些快捷键，大大提高了实验效率。
2. 更加熟悉了MIPS核心指令的功能，初步掌握对不同指令的一些实现方式。
3. 加深了对课程所学内容的理解，对CPU设计的整个流程更加熟悉，对设计的一些细节也理解得更加透彻。
4. CPU设计实验感觉老师为我们做得略多了一点，将任务分解成一个个小任务给我们去完成的确减轻了我们的很多负担，但是感觉分解得过细，导致对CPU设计流程的整体认知不够，不利于形成整体上的设计思想。建议老师设计实验时不用分解得过于细致，身边很多同学都觉得CPU实验的难度甚至低于cache实验。

# 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:** |