



**《计算机组成原理》**

**课程实验报告**

姓名：黄子安

学院：信息学院

系：软件工程

专业：软件工程

学号：22920212204396

目录

[一、实验环境 3](#_Toc135491188)

[二、实验目的 3](#_Toc135491189)

[三、实验内容 3](#_Toc135491190)

[2.1验证实验 3](#_Toc135491191)

[2.1.1单总线 MIPS 处理器（硬布线控制器）（5条指令） 3](#_Toc135491192)

[2.1.2单总线结构 MIPS 处理器（微程序控制器）（5条指令） 10](#_Toc135491193)

[2.1.3单周期 MIPS 处理器（硬布线控制器）（24条指令） 16](#_Toc135491194)

[2.1.4单周期 RISC-V 处理器（硬布线控制器）（9条指令） 20](#_Toc135491195)

[2.2设计实验 25](#_Toc135491196)

[2.2.1单总线结构 MIPS 处理器（微程序控制器）（增加1条add指令）（6条指令） 25](#_Toc135491197)

[2.2.2单总线结构 MIPS 处理器（微程序控制器）（再增加1条j指令）（7条指令） 29](#_Toc135491198)

[2.3挑战性设计实验 34](#_Toc135491199)

[2.3.1单总线结构 MIPS 处理器（硬布线控制器）（6条指令） 34](#_Toc135491200)

[2.3.2单总线结构 MIPS 处理器（硬布线控制器）（7条指令） 38](#_Toc135491201)

**第五次实验 中央处理器数据通路实验**

# 一、实验环境

* **MIPS汇编仿真器**
* **Windows系统或Mac os环境下运行Logisim软件**

# 二、实验目的

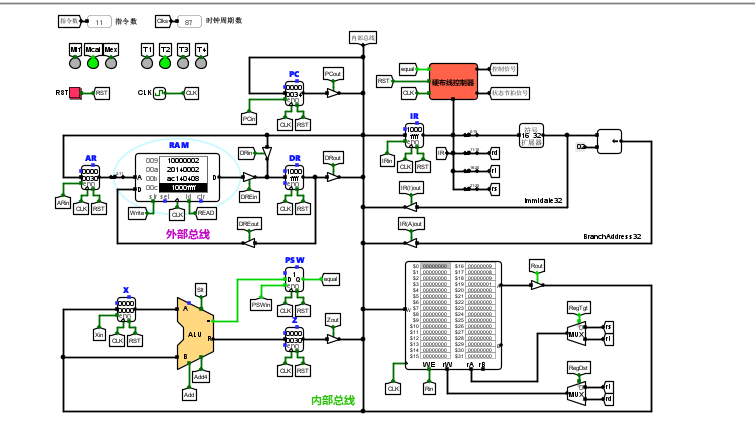
* **了解和掌握单周期处理器和单总线结构处理器的数据通路特点；**
* **了解和掌握微程序设计的设计原理；**
* **理解硬布线控制器的设计原理；**

# 三、实验内容

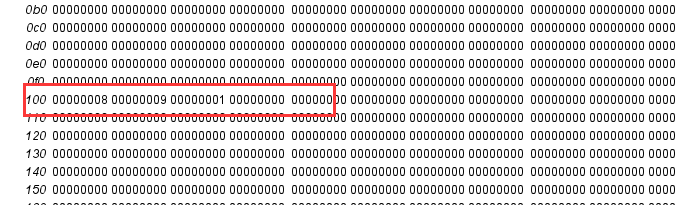
## 2.1验证实验

### 2.1.1单总线 MIPS 处理器（硬布线控制器）（5条指令）

单周期MIPS的数据通路

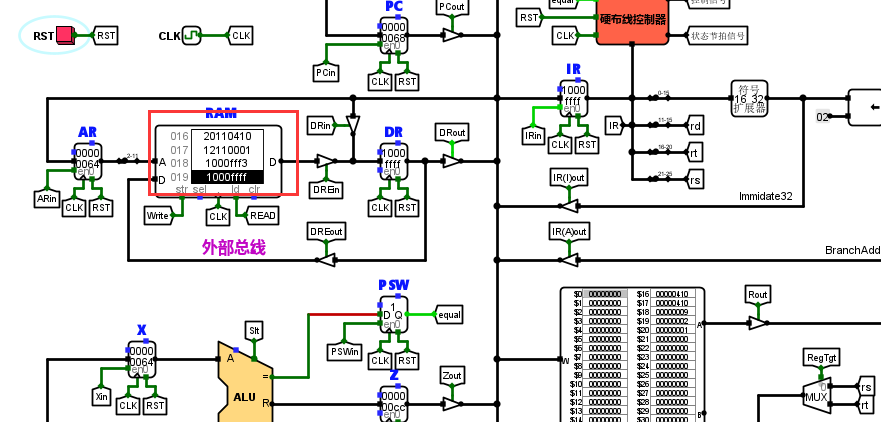


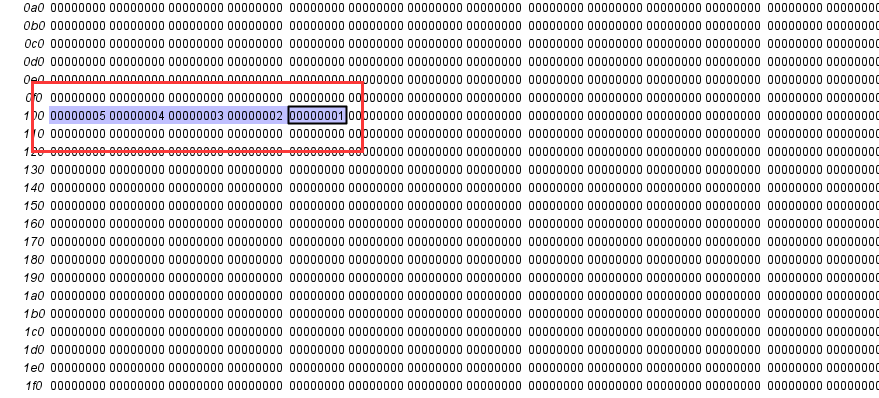
运行test1.hex文件，可以在存储器的100-102存储单元内看到预期的数字



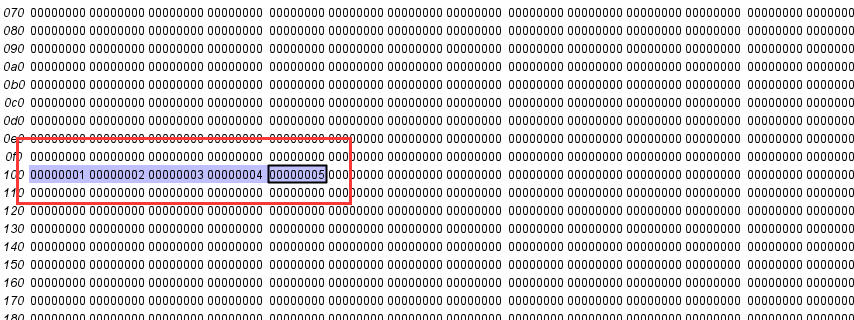
运行五个数的排序程序

在RAM中装入指令sort3\_mips\_bus.hex，之后运行程序，可以在存储器的100单元开始观察降序排序后的数组





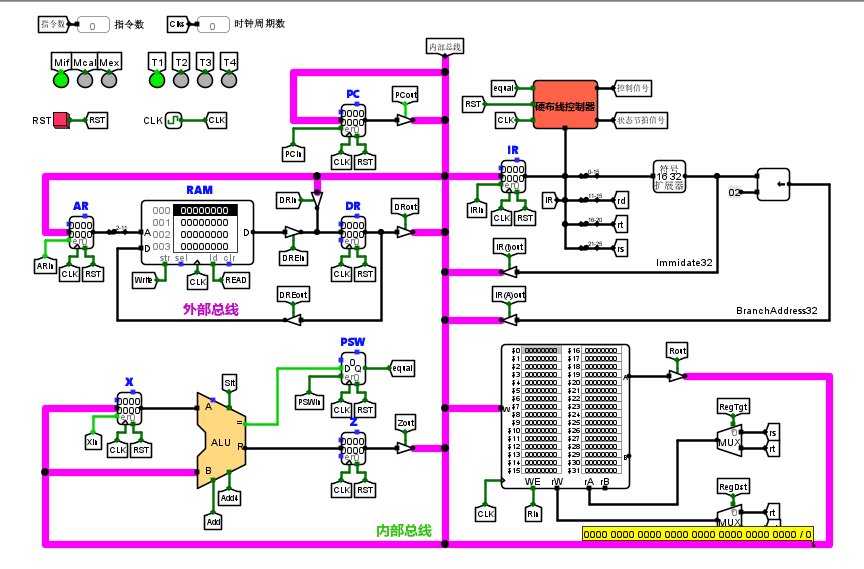
同理在RAM中装入sort4\_mips\_bus.hex，运行程序，可以在相同的位置找到升序排序后的数组



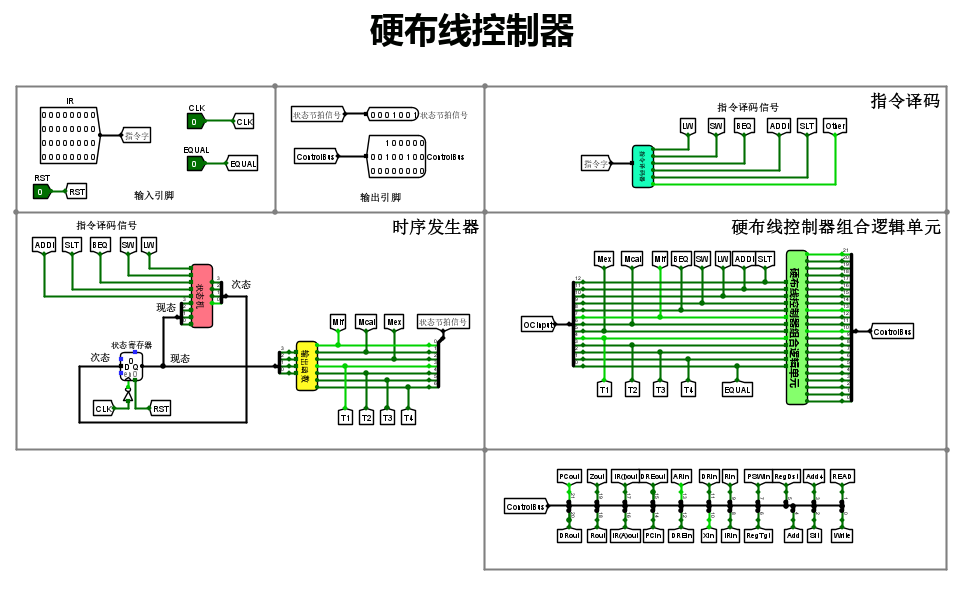
原理分析

数据通路为单总线结构，基本上所有数据（部分数据如标志信号除外）都要经过这一条内总线，为了防止数据冲突，每个对应部件的数据流向内部总线前都会被三态门和控制信号锁住，直到给出信号才会打通三态门放出数据，因此在该硬布线控制器中最重要的是协调22个控制信号给出。

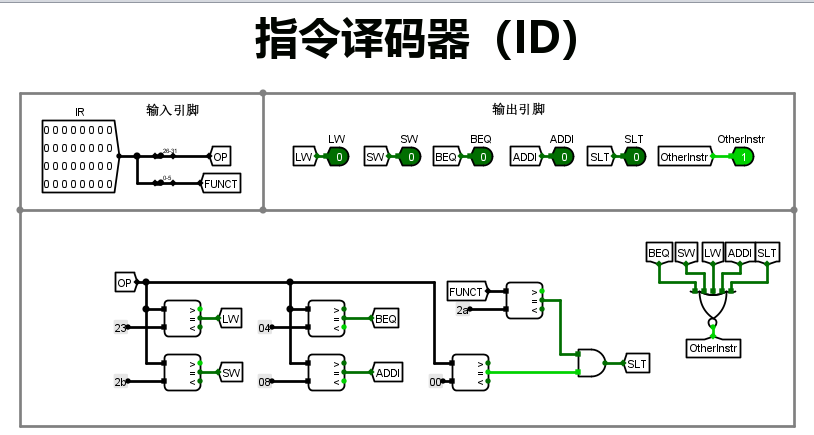
从宏观上出发，取指周期会先将PC的值从总线流向AR，同时流入X；再利用运算器进行PC+4；之后将PC+4存入PC，同时进行RAM寻址，获取指令内容存入DR；计算周期和执行周期也基本同理，因此关键在于根据指令和当前状态给出下一个时钟周期的状态的控制信号，从而实现信号的控制交替，而硬布线控制系统正是通过使用状态机从硬件上实现状态之间的传递。



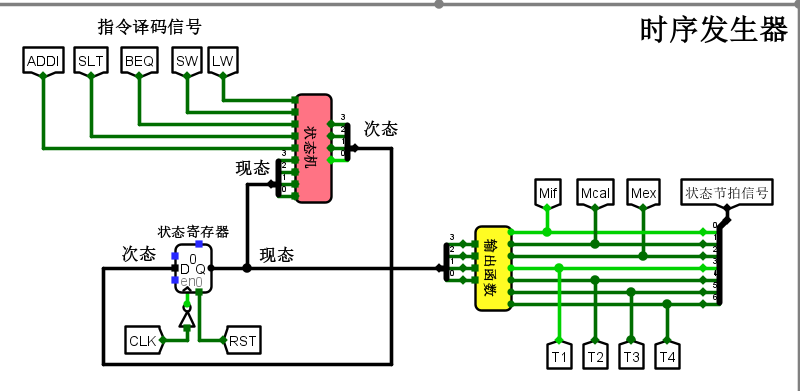
硬布线控制器分为以下部分，控制器会从数据通路中的IR中读取整条指令存入指令字，之后指令进入指令译码器获取该指令的类型



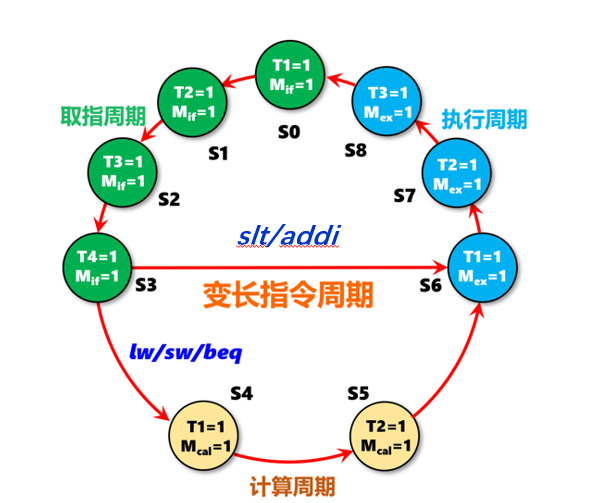
指令译码器如下图所示，从原理上非常直观，会先通过分线器取出指令的高6位，因为MIPS32是定长指令，指令格式规整，前6位就是该条指令的操作码，之后直接利用6个比较器并行比较确定指令的类型，其中slt比较特殊，因为其为R型指令，在MIPS32中R型指令的操作码都为0，是靠末尾的6位来区分，即图中的funct位，如果funct位是2a证明是slt指令，之后通过与门来确定是slt指令。除此之外会将所有指令进行或非处理，使得当所给指令不在支持的6条指令内时系统可以区分不会导致出错，因此最后指令译码器会给出唯一对应的指令信号。



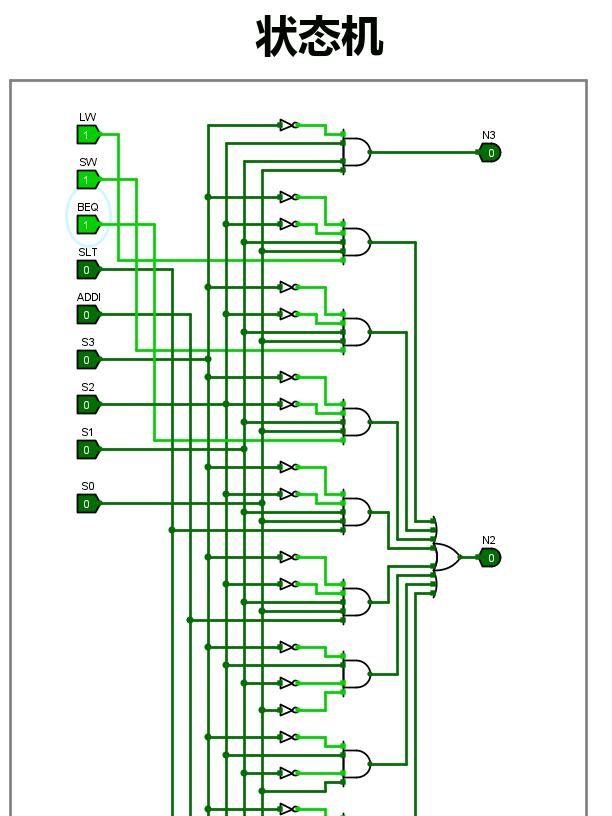
获取当前的指令类型之后，最重要的就是在每一个状态节拍生成22个对应控制信号以及下一个节拍的状态信号，接下来的时序发生器会将一个指令周期划分为多个机器周期，一个机器周期又包含多个节拍，之后由生成函数生成对应节拍下的的控制信号



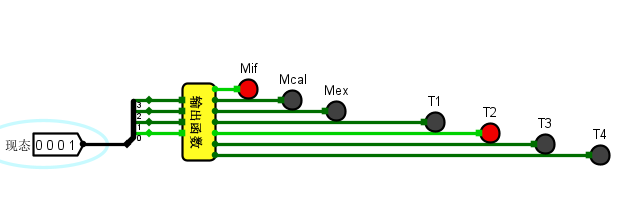
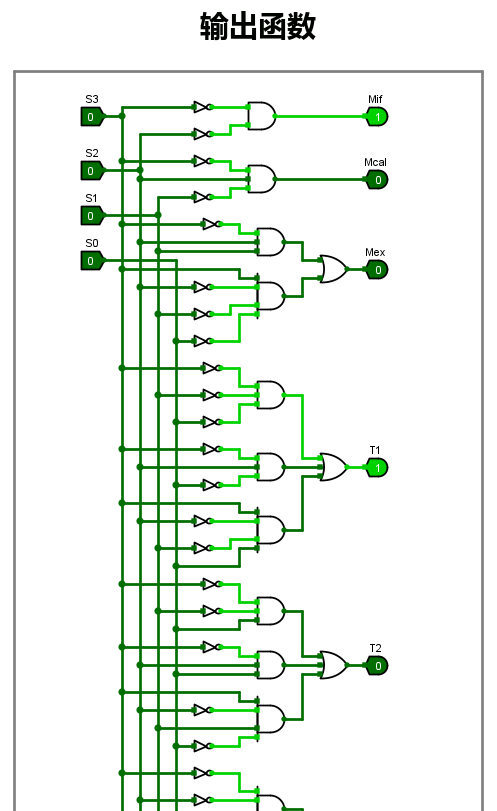
MIPS32的机器周期和节拍如下图所示，其给出了在不同指令下状态的跳转关系以及每一个机器周期和节拍内部的状态信号，lw、sw、beq会跑完这里的每一个小圈，而slt和addi会在中间发生跳跃，这些状态切换将由状态机完成



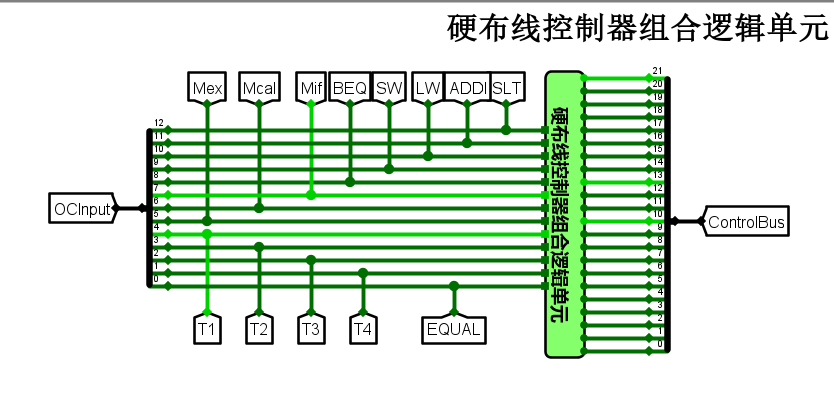
状态机内部如图所示，原理就是利用组合逻辑，根据当前状态和指令类型生成下一个状态的序号，具体实现属于逻辑演算的范畴

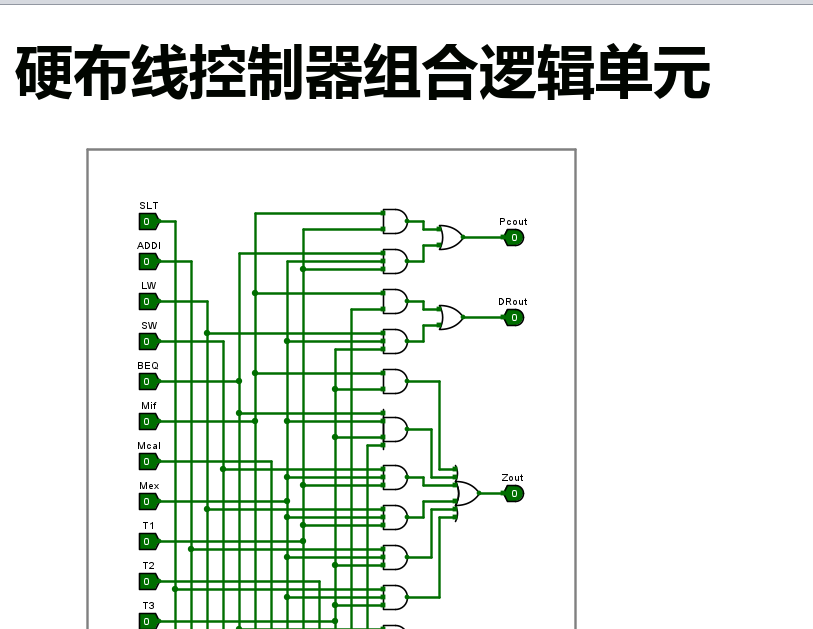


在能够实现节拍状态之间的切换之后，接下来需要将对应的状态用时序信号表示出来，在图上小圆圈可以看出是利用T1、T2、T3、T4，Mif、Mcal、Mex来在时序角度区分八个状态，输出函数内部将状态的0000、0001……1111和时序信号给出映射关系，这里也是属于组合逻辑演算的范畴，可以看到输出函数将输入的现态输出为了时序信号的状态周期电位和节拍电位，比如现态=0001=S1，则输出=Mif、T2两个信号

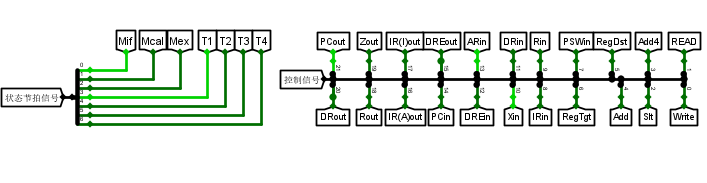


得到时序区分之后，要做得就是给出22个控制信号，这部分由硬布线控制器的组合逻辑单元完成，将综合使用指令类型和对应的时序信号，也是采用组合逻辑暴力穷举配合电路输出22个控制信号的状态，生成的22个信号被分线器合在一起输出到控制总线

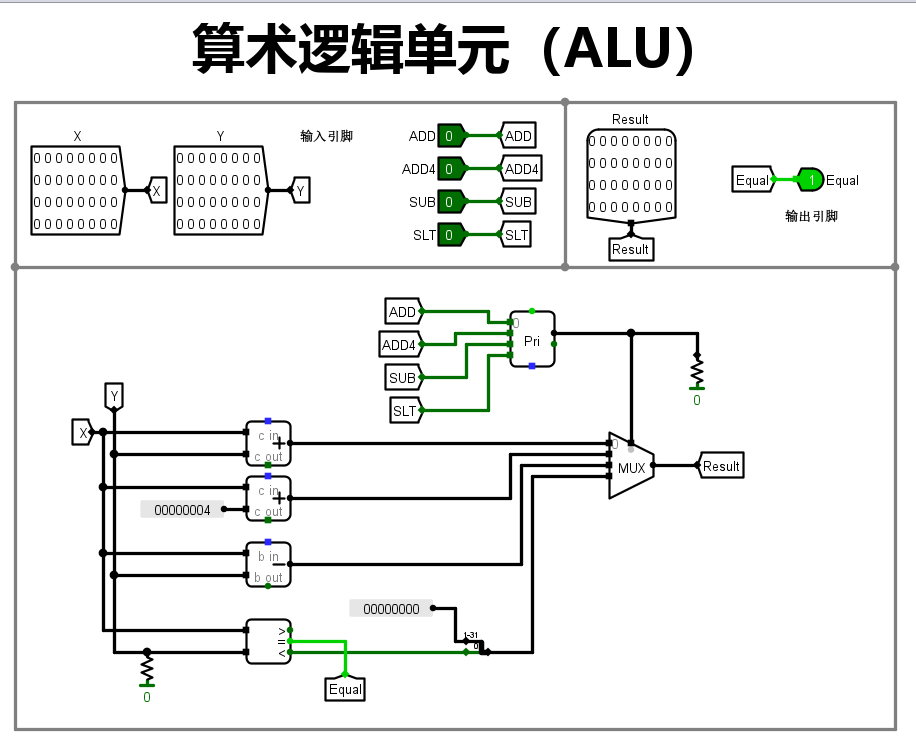


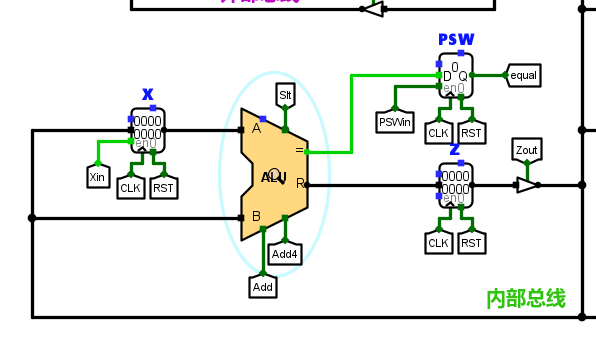


至此硬布线控制器完成了在不同的时钟周期给出对应的控制信号并传输到控制总线，于是便可以在数据通路利用这些信号通过22个三态门的连通来来控制单总线的使用，以此实现指令的有序执行



最后单总线MIPS处理器中还有一个ALU，其有四个对应的信号输入，但是SUB信号在实际数据通路中并没有被给出，因此一直是一个空信号，相等会用于状态寄存器的修改，因此被单独给出，其余的运算内容根据输入的控制信号和多路选择器即可完成运算。

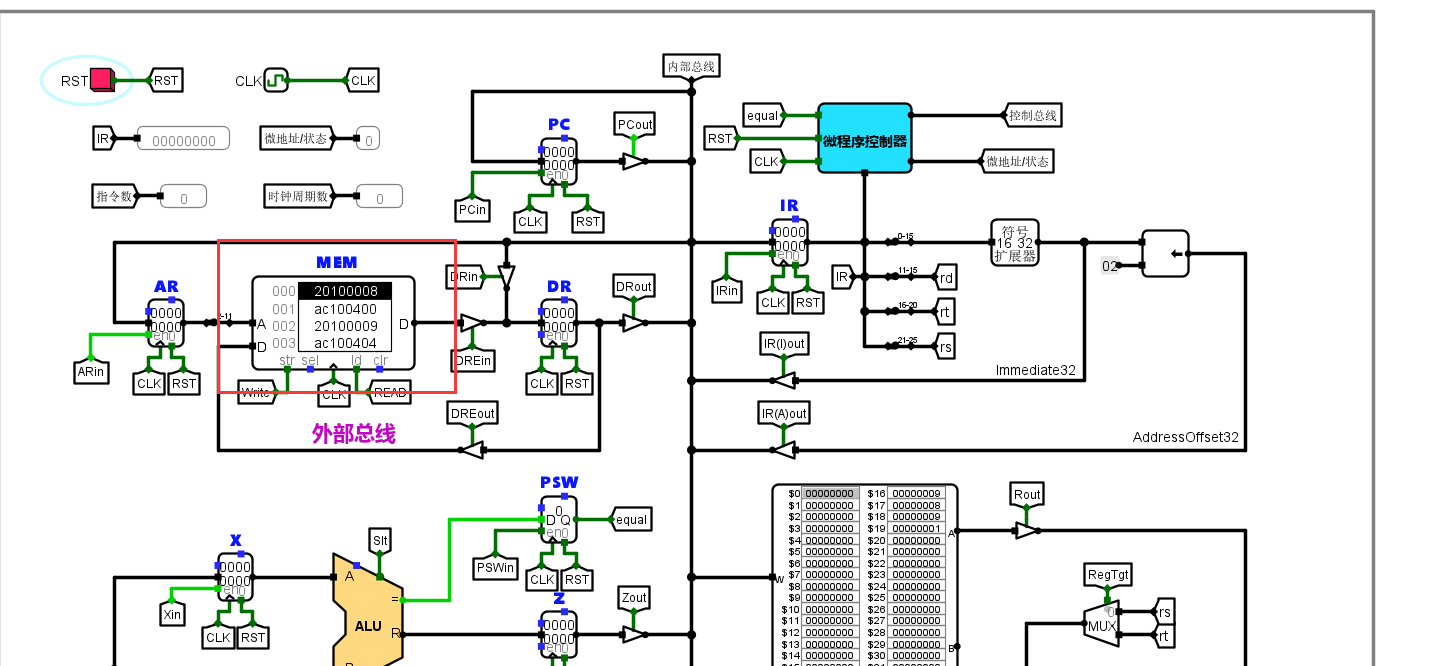


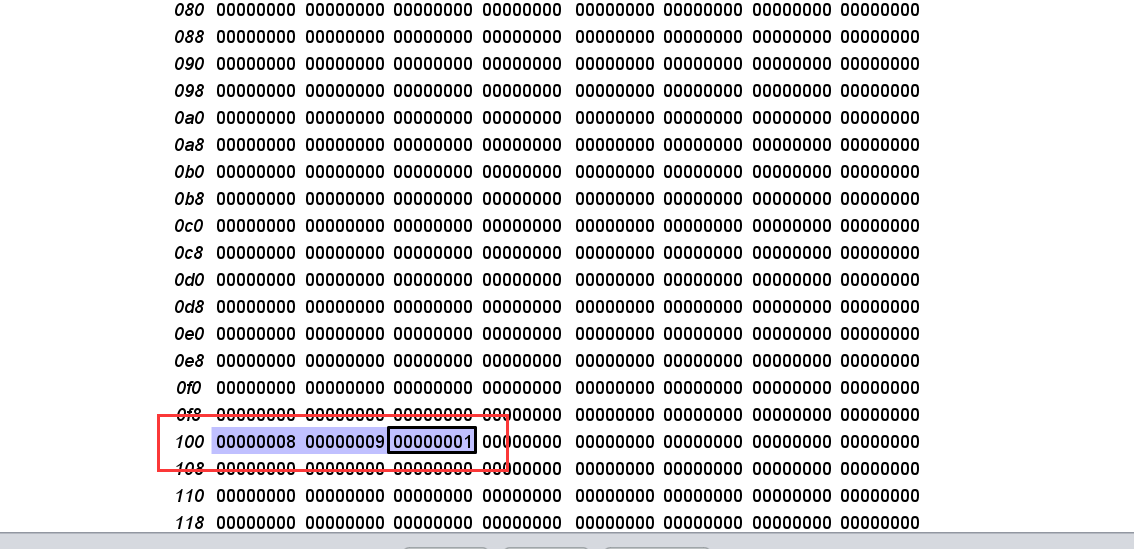


**在该单总线结构 MIPS 处理器（硬布线控制器）的数据通路上，不可以运行求累加和程序、计算费波那契数列程序，因为这两个程序需要用到ADD指令，在该处理器中没有实现，会被是为其他指令不会执行任何的对应操作**

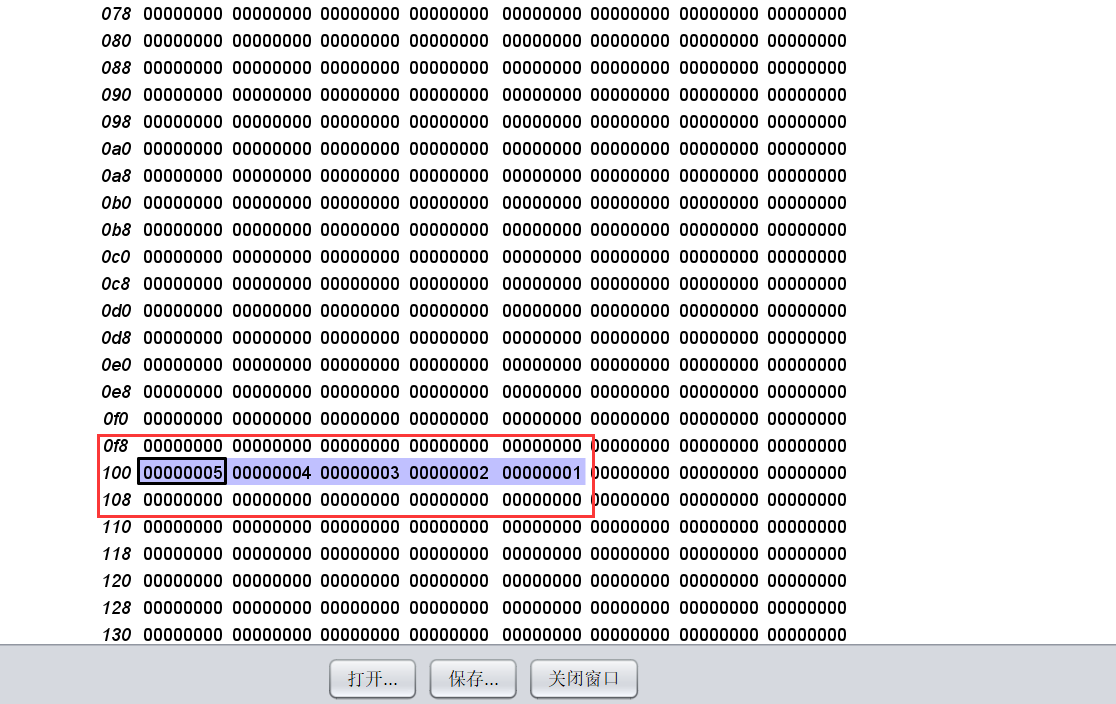
### 2.1.2单总线结构 MIPS 处理器（微程序控制器）（5条指令）

运行程序**test1.hex，**先装向RAM中装入指令，之后运行程序，在存储器的100号单元开始找到运行后的数据





运行程序**sort3\_mips\_bus.hex，**同理可以在存储器的100号单元开始可以看到降序排列的数组

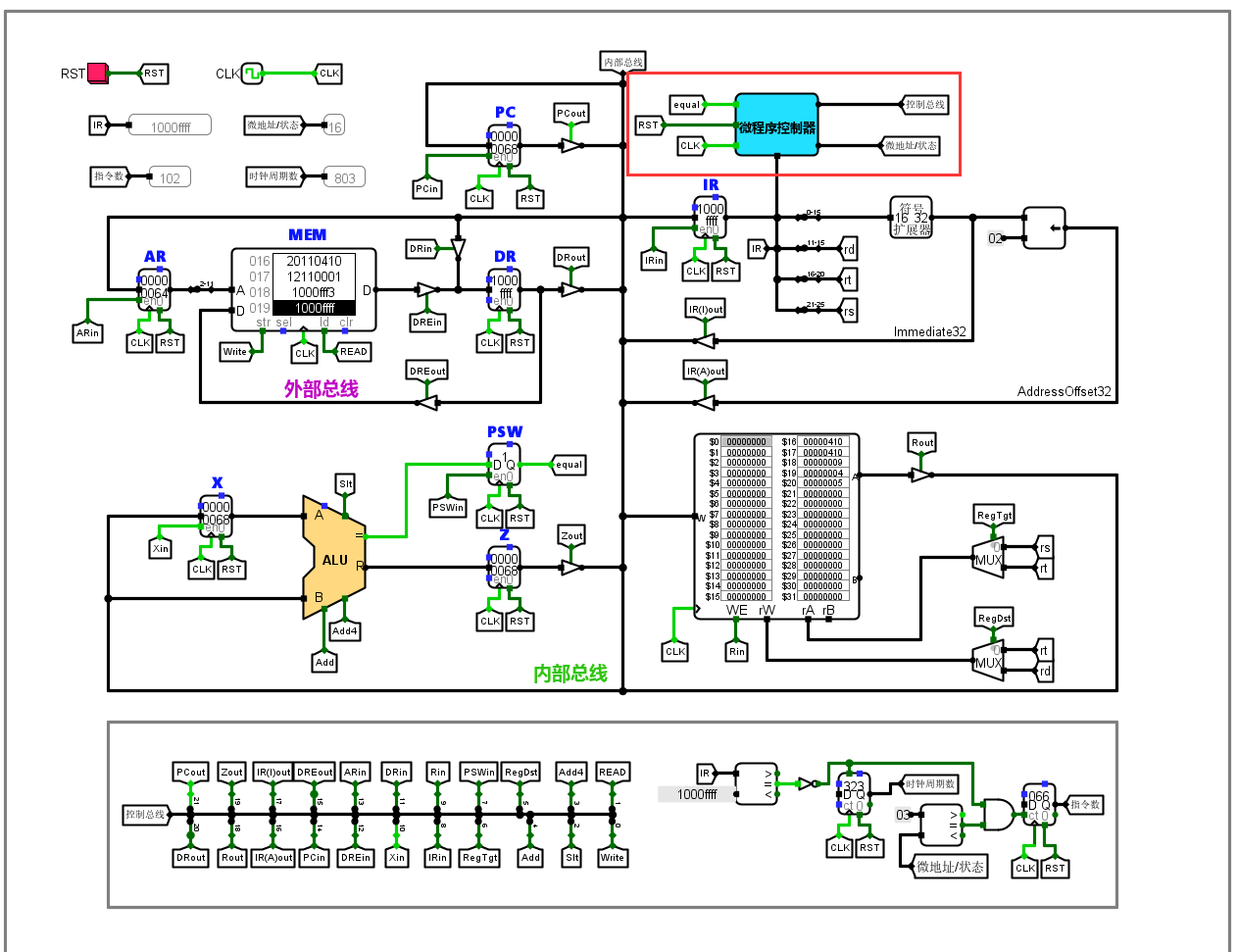


运行程序**sort4\_mips\_bus.hex，**同理可以在存储器的100号单元开始可以看到升序排列的数组



原理分析

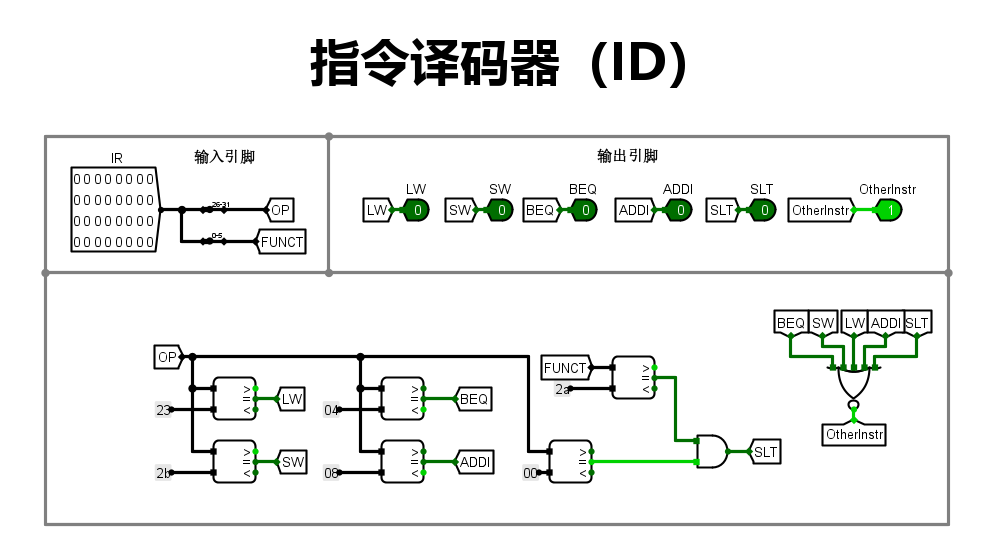
对于微程序控制的单总线处理器，在数据通路上和硬布线处理器完全相同，区别在于每一个时钟周期给出的控制信号将由微程序通过软件的形式给出



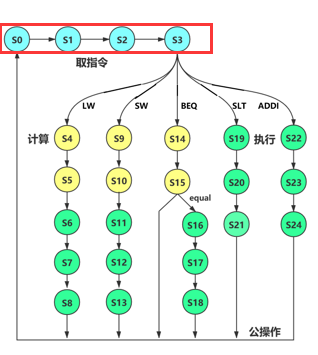
微程序控制器的第一步和硬布线控制器相同也是先进行对输入的指令取指令字并进行指令译码，生成对应指令的类型区分信号



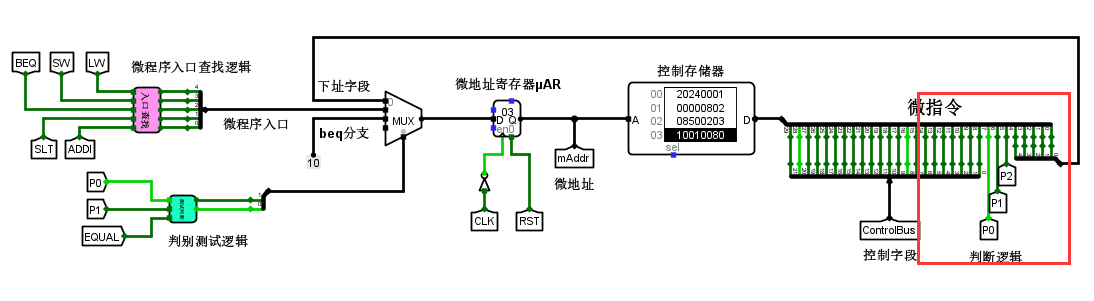
指令译码器和硬布线控制器的完全相同，因为这两个处理器都只支持相同的6条指令

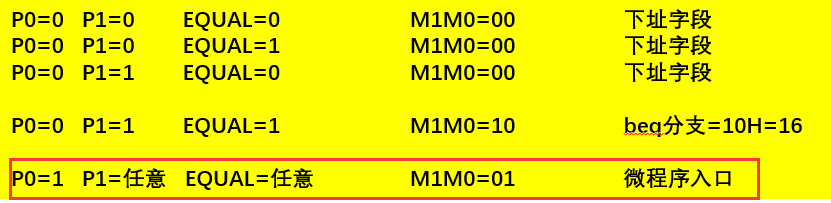
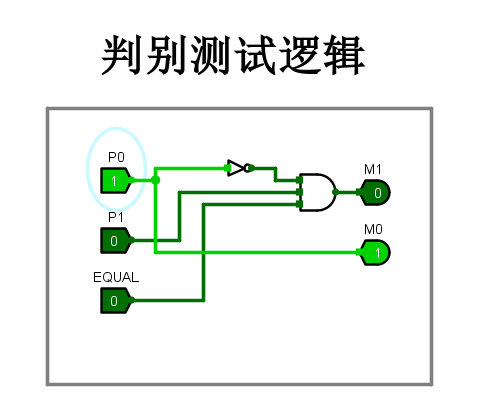


获取指令类型区分信号之后，开始的四个时钟周期会先进行取指令，通过下址地址实现下一条微指令切换

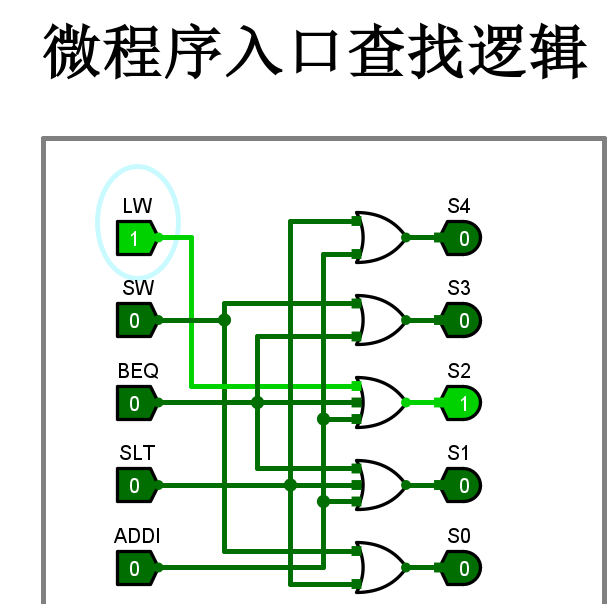


当进行到S3的时候，所给出的P0将被置为1，这时候判断测试逻辑将输出01，多路选择器将会将地址输入修改为微程序的入口地址而不再是下址地址





微程序入口的查找逻辑如图所示，会根据指令的类型寻找合适的入口地址，以此实现不同指令的区分



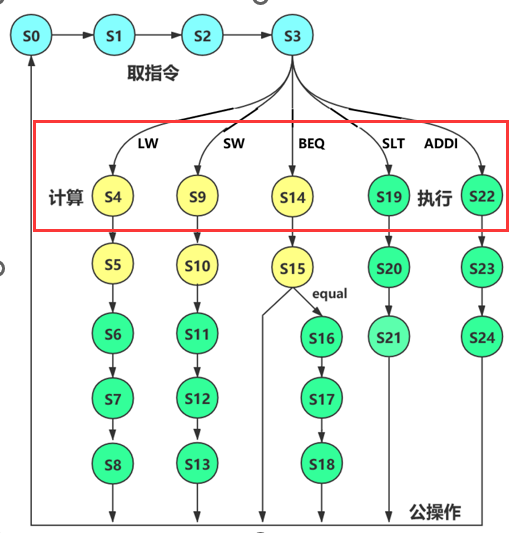
**LW=1 微程序入口地址=4**

**SW=1 微程序入口地址=9**

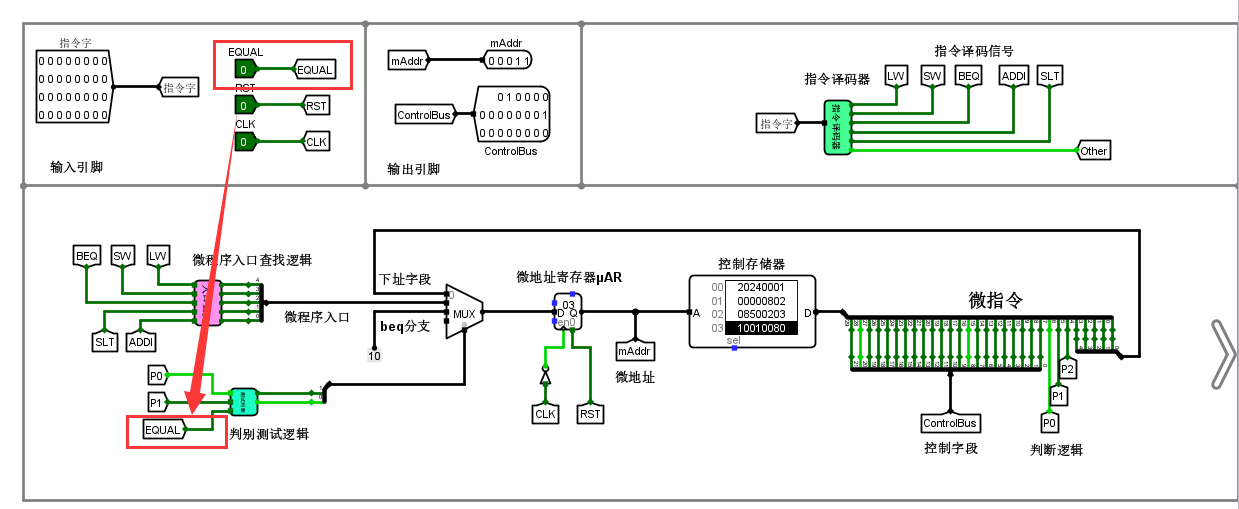
**BEQ=1 微程序入口地址=14**

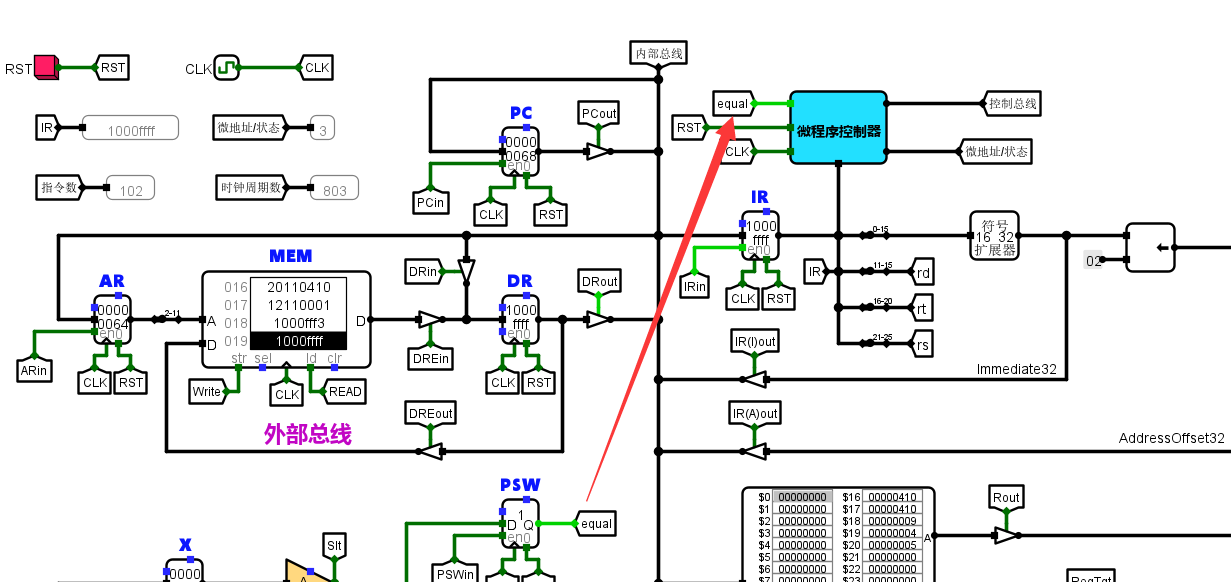
**SLT=1 微程序入口地址=19**

**ADDI=1 微程序入口地址=22**

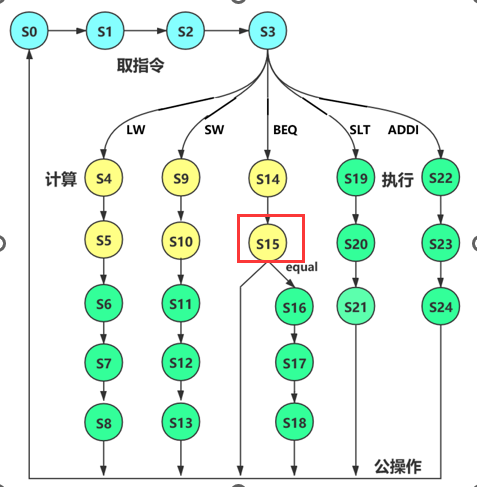


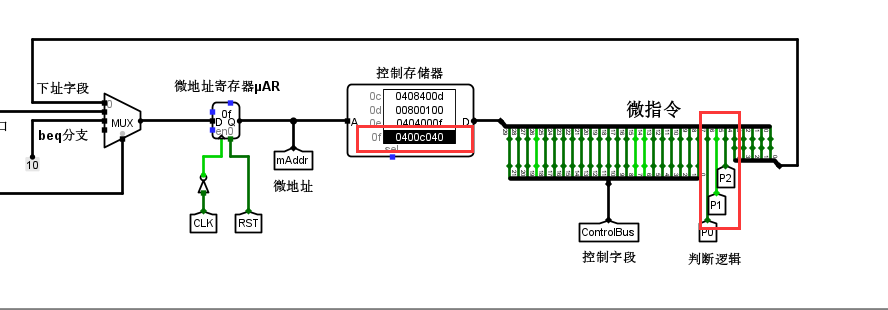
完成指令类型修改微程序入口之后便继续根据下址地址来实现状态的顺序切换，直到完成该条指令的执行，其中beq指令可能会发生跳转有所不同，在这个处理器中有EQUAL来实现控制，EQUAL由运算器生成并存储在状态寄存器PSW中，之后由总线运输到微程序控制器作为输入

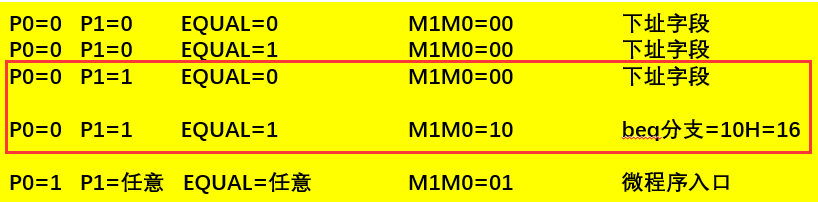


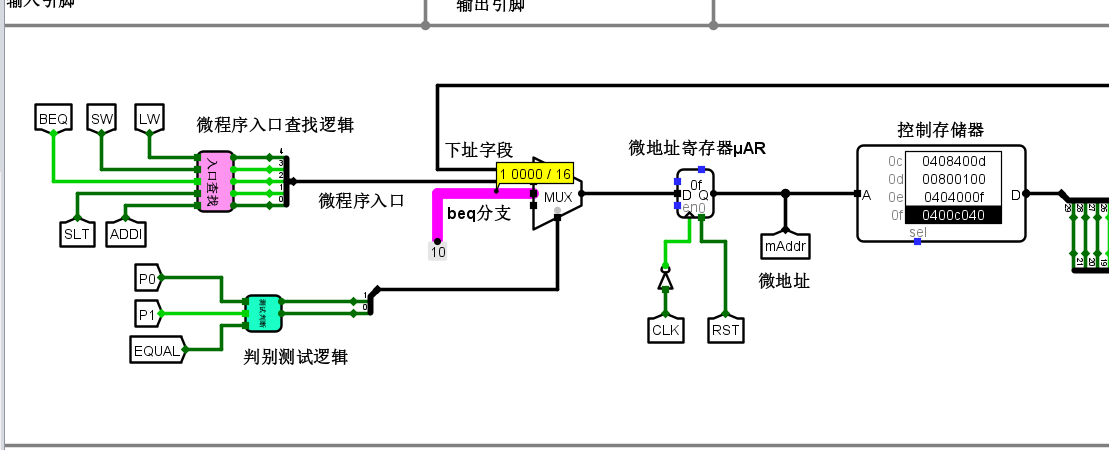
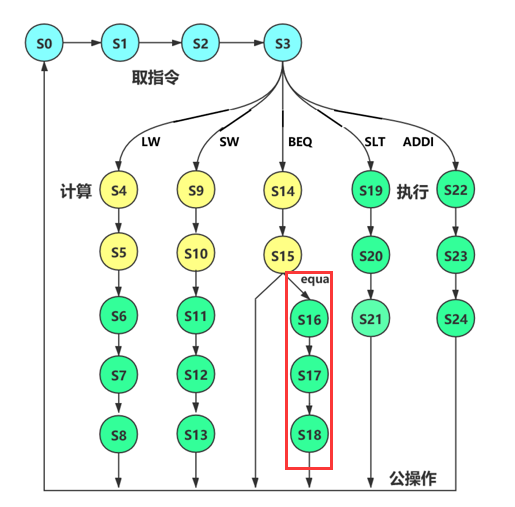


当微程序执行到S15的时候，将会把P1置为1





 这个时候将会根据equal的值和P1=1由判断测试逻辑决定是否发生跳转，如果需要跳转，这个时候多路选择器的输出会变为beq指令需要跳转情况下的状态入口S16，之后继续根据下址地址进行微指令的顺序执行

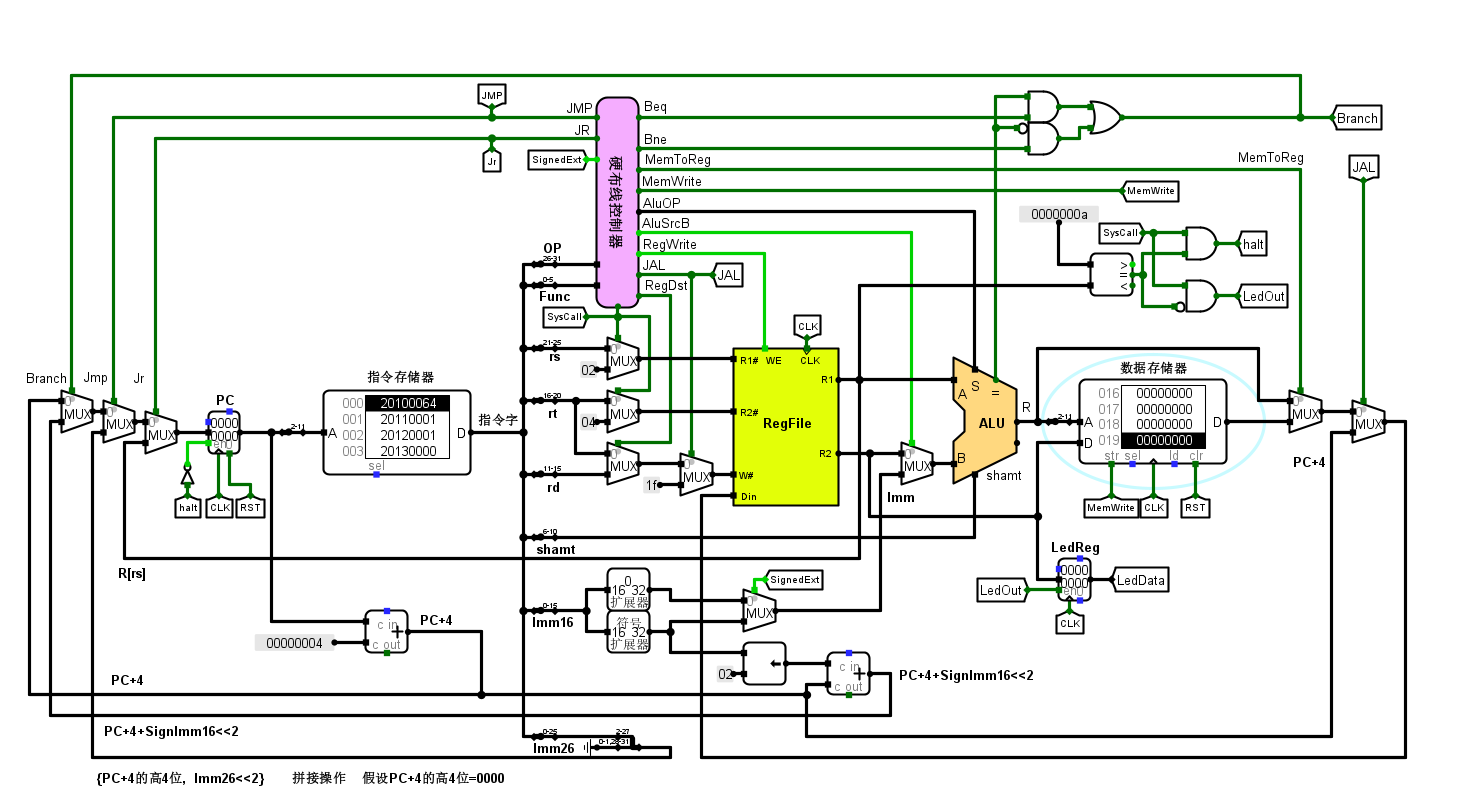
  


执行完成对应指令的所有状态之后将会根据下址地址跳转回S0，来实现下一条指令的执行，经过这些步骤将实现指令的有序执行

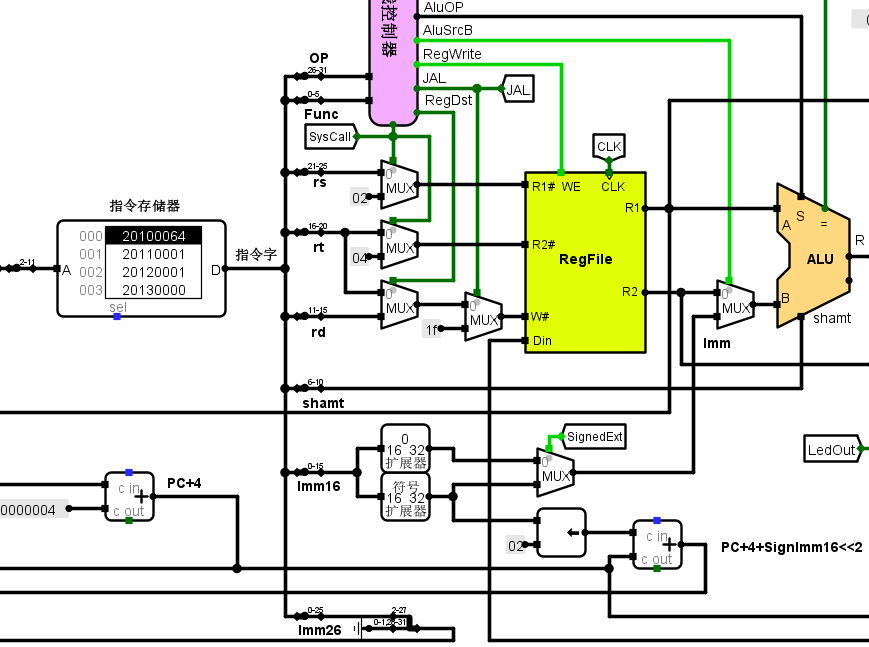
### 2.1.3单周期 MIPS 处理器（硬布线控制器）（24条指令）

原理分析

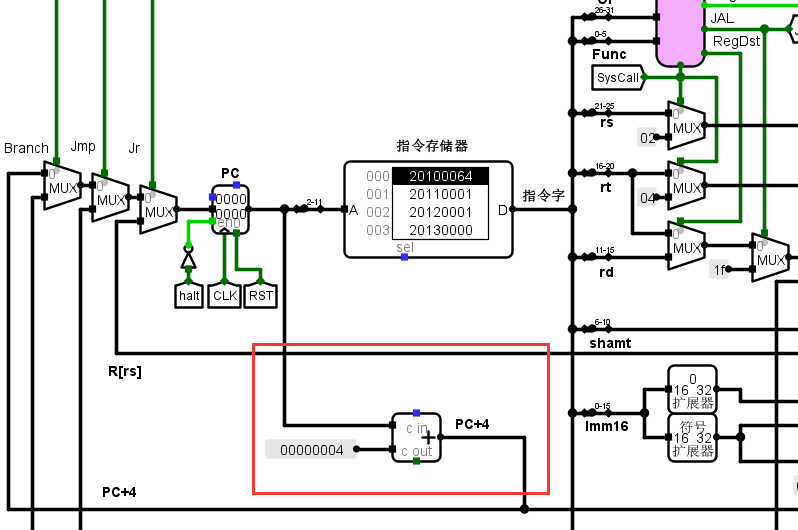
数据通路如图所示，这里的所有24个指令都会在一个周期之内执行完成，因此所有都需要走专线，会大量使用多路选择器



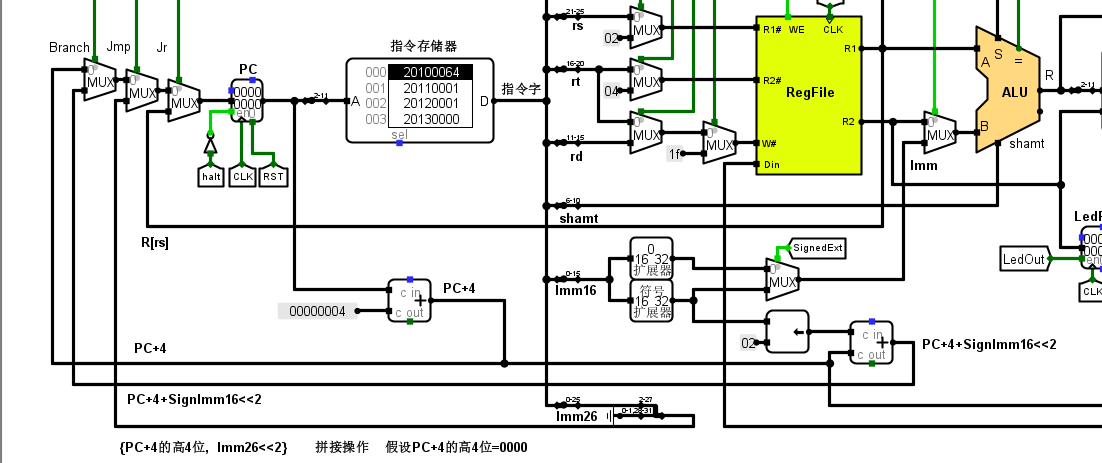
该部分进行指令的解析，将R、I、J三种类型指令在一起进行处理，按照三种指令的格式提取出rs、rt、rd、shamt、imm16和imm32，之后由控制器给出控制信号来确定应该使用哪部分数据用于寄存器的选取以及流入运算器

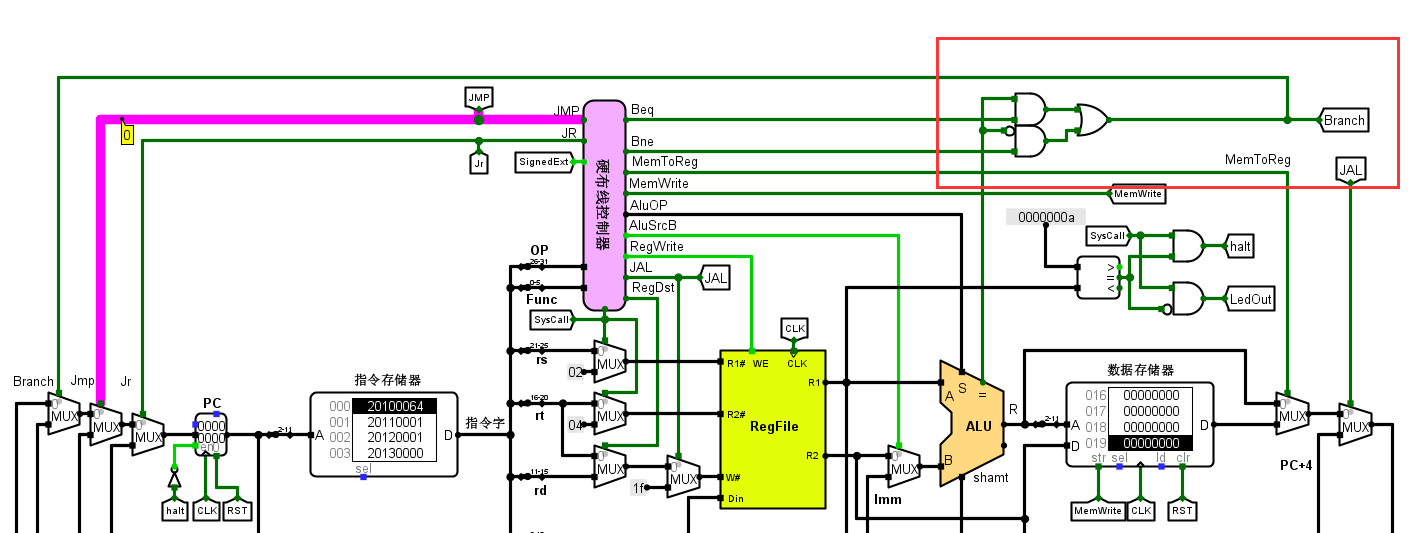


除此之外为了保证所有指令在一个周期内完成，要将PC+4这一个步骤进行单独处理，在处理器中单独设置了一个+4的加法器

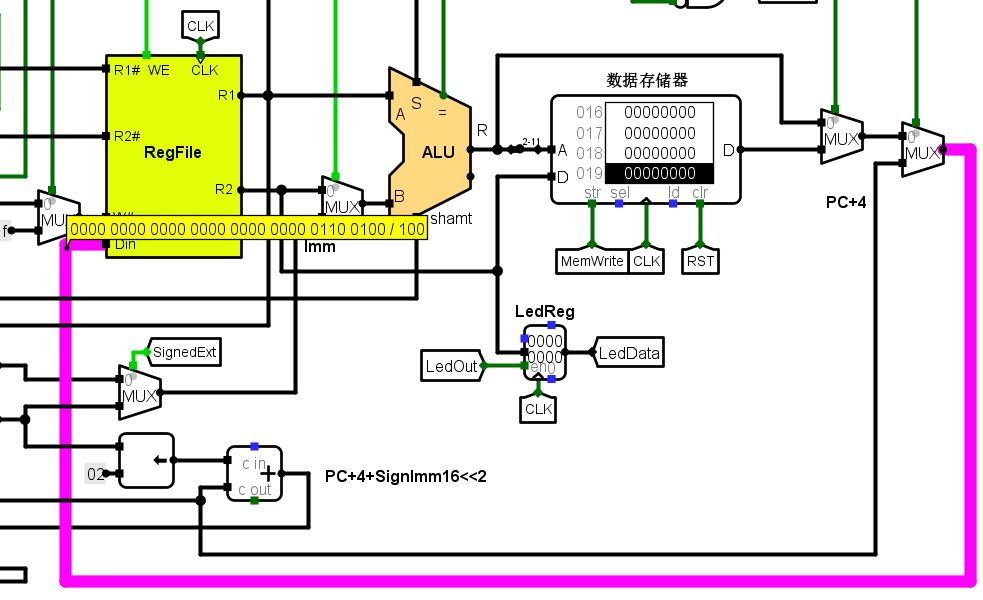


除了顺序执行之外，PC可能也会发生跳转，这里使用控制信号进行确定，所有PC的改变情况有三种，分别为PC+4，PC+4+signimm16<<2，imm26，这里由两个三个多路选择器来实现是使用哪一种改变方式

 其中无条件跳转的控制信号由控制器直接给出，而有条件跳转需要进行条件判断后给出，控制器给出是相等跳转还是不相等跳转的信号，之后由运算器提供两者是否相等的状态判断结果来实现判断是否跳转，如果需要跳转就会将Branch置为1，由多路选择器来确定PC的输入



除了分支跳转外还有子程序跳转也根据类似的方法实现，但是此时会将PC+4存入寄存器中，用于后续的返回



halt用于控制停机，当控制器给出syscall信号时，PC计数器使能端被置为0，从而终止程序

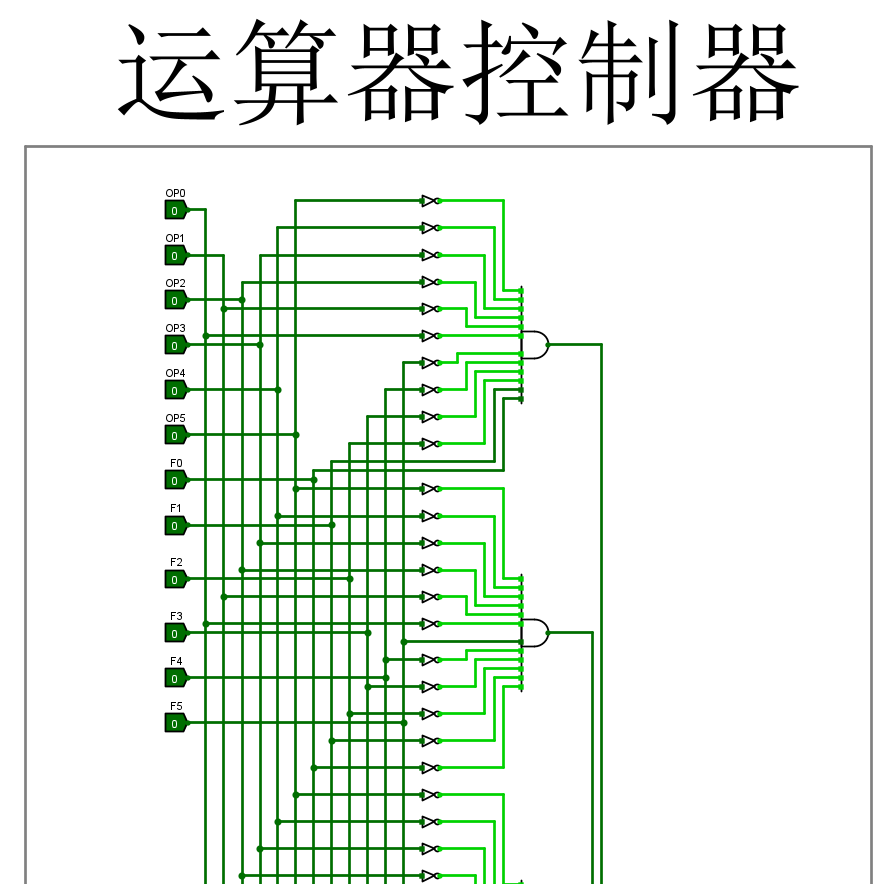


之后重点是由硬布线控制器根据指令生成对应的控制信号，因为该处理器是单周期，所以给出一条指令硬布线控制器需要直接给出所有对应的控制信号

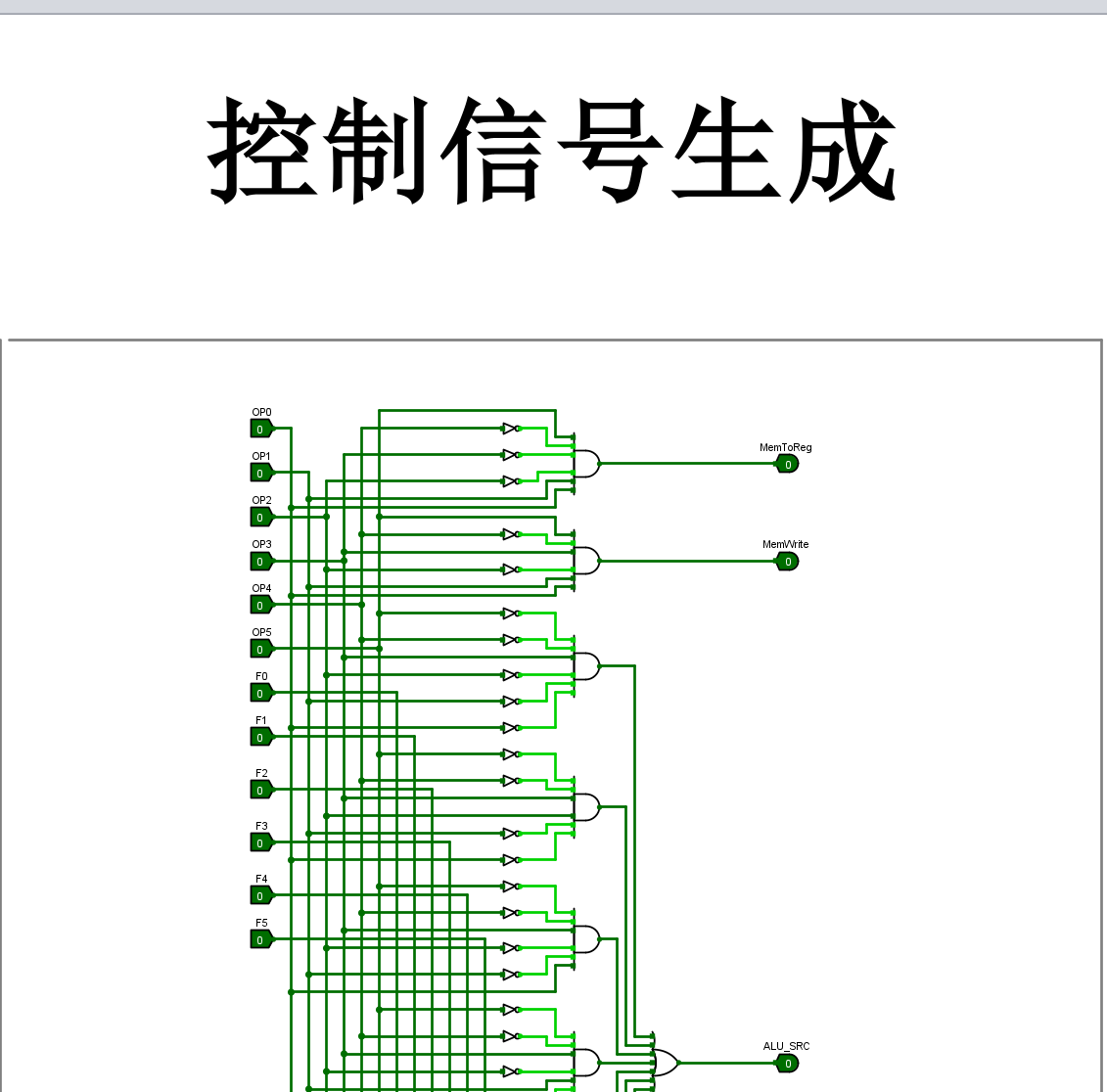


运算器控制器根据对应的操作码和指令的低6位给出对应的Aluop信号，其中使用组合逻辑进行硬布线，使用逻辑门电路生成对应的信号，当OP=0时，说明是R型指令，之后根据Fun的值确定需要的运算器控制信号；如果不是R型指令，则利用OP来决定给出运算器控制信号





控制信号的生成和运算器控制信号原理上基本相同，也是通过组合逻辑利用OP和FUNC决定需要给出对应的控制信号



由此通过这两个控制电路硬布线控制器可以在指令给出的同时直接给出对应的8个控制信号

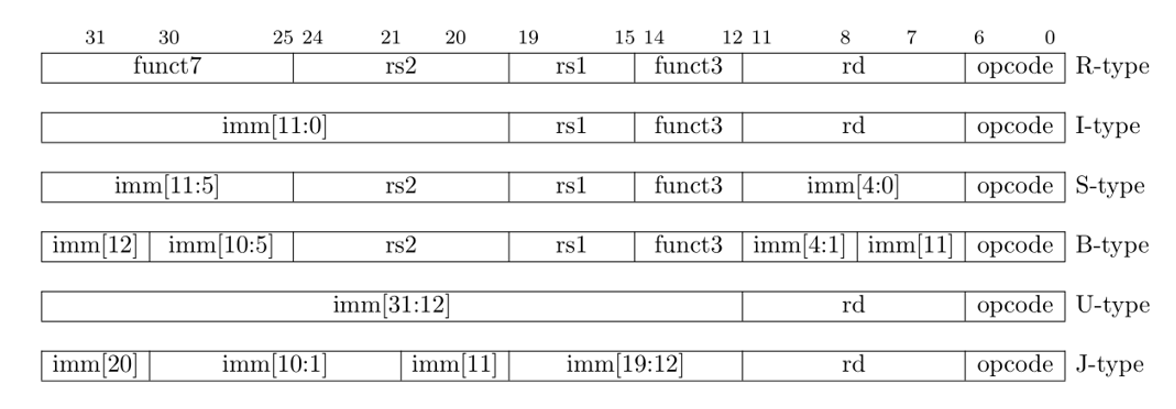
该单周期 MIPS 处理器的控制器理论上可以采用微程序控制器的方法设计，根据指令中的操作码进行寻址使用对应的微程序，但是实际上单周期处理器每一条指令的控制信号都是固定的，微程序使用微程序控制器会引入不必要的复杂性和时钟周期延迟，不符合单周期处理器的设计目标。

### 2.1.4单周期 RISC-V 处理器（硬布线控制器）（9条指令）

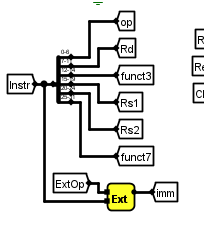
原理分析

RISC-V的程序计数器部分如图所示，当没有给出跳转信号时会执行PC+4操作完成顺序执行，如果给出跳转信号则使用扩展后的立即数进行跳转，之后会从指令存储器读取指令，其中Jump是无条件跳转的控制信号，Branch和Zero用于条件跳转

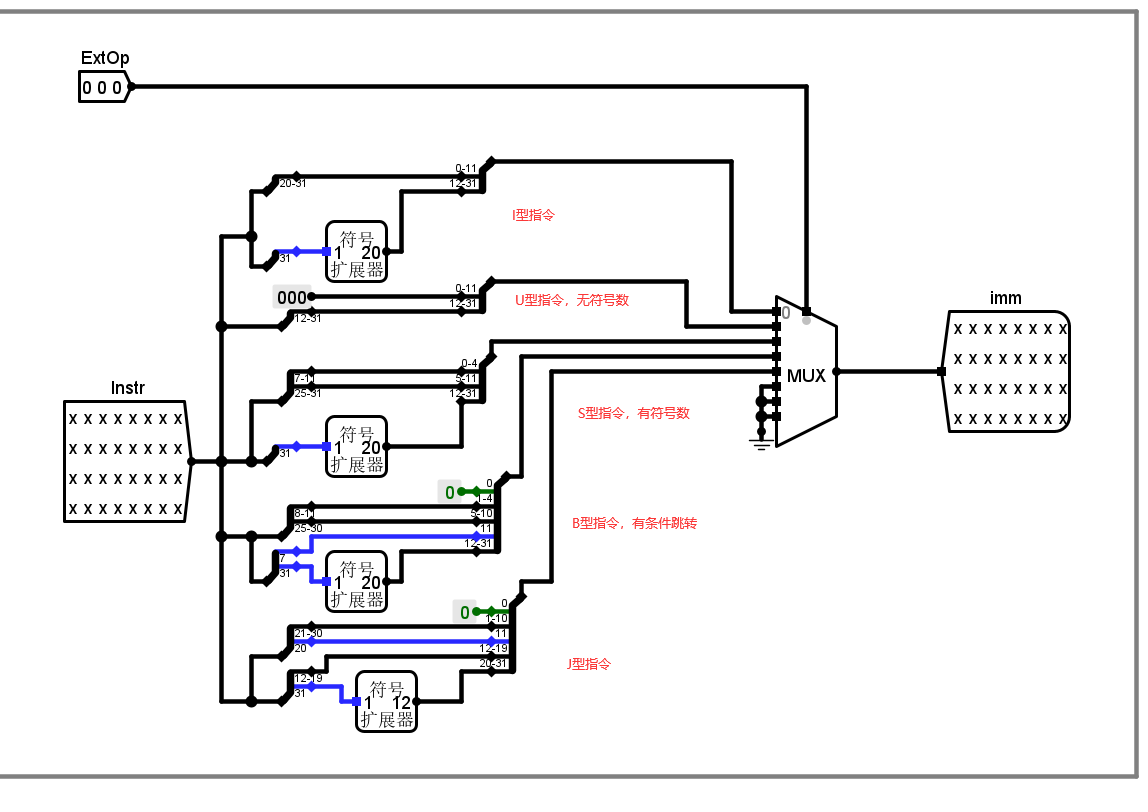




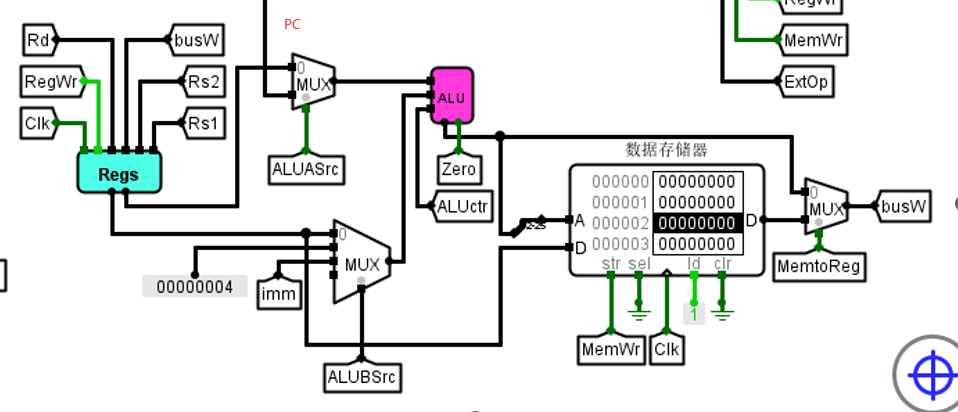
RISC-V的指令格式如上图所示，其指令格式非常整齐，可以直接利用分线器获取指令中的各部分数据信息，同时因为不同指令中的立即数存放方式不同，所以需要根据指令来给出具体的立即数扩展方式，立即数的不同存放方式总共有5中，因此ExtOp的位宽是3



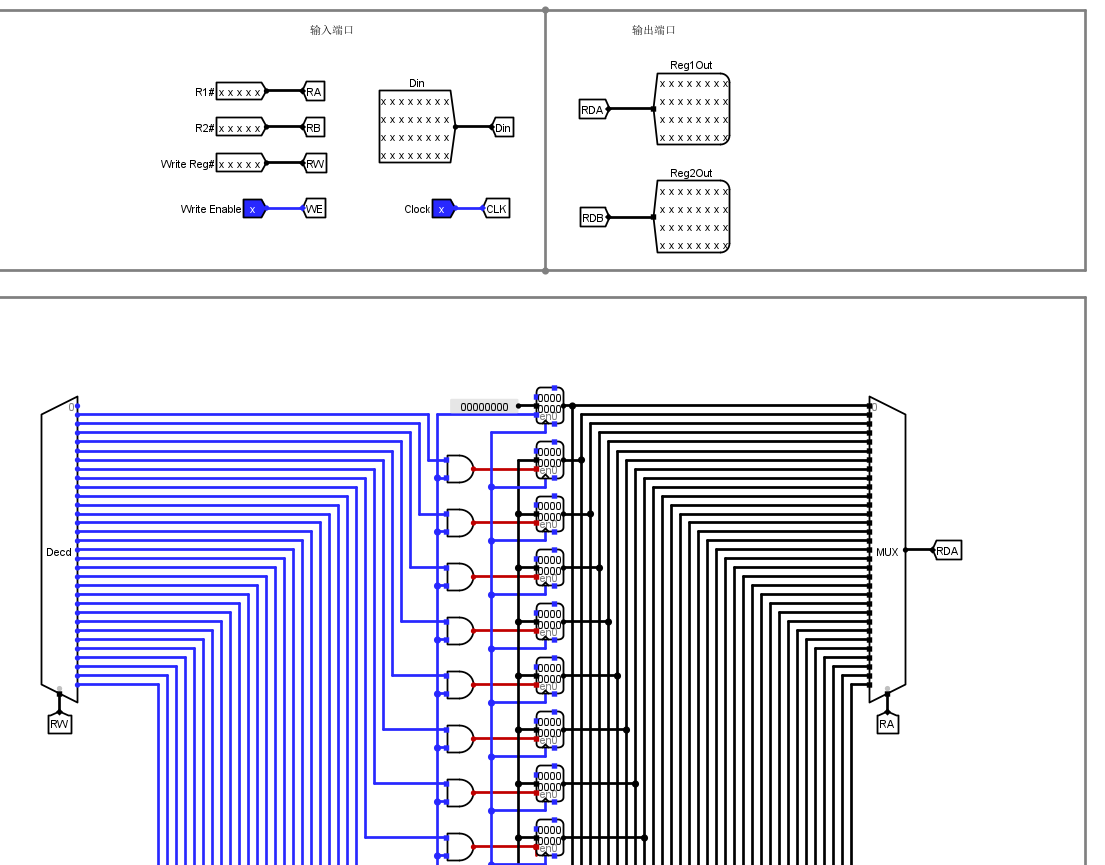
符号扩展器的内部如图所示，会根据ExtOp选择合适的扩展方式



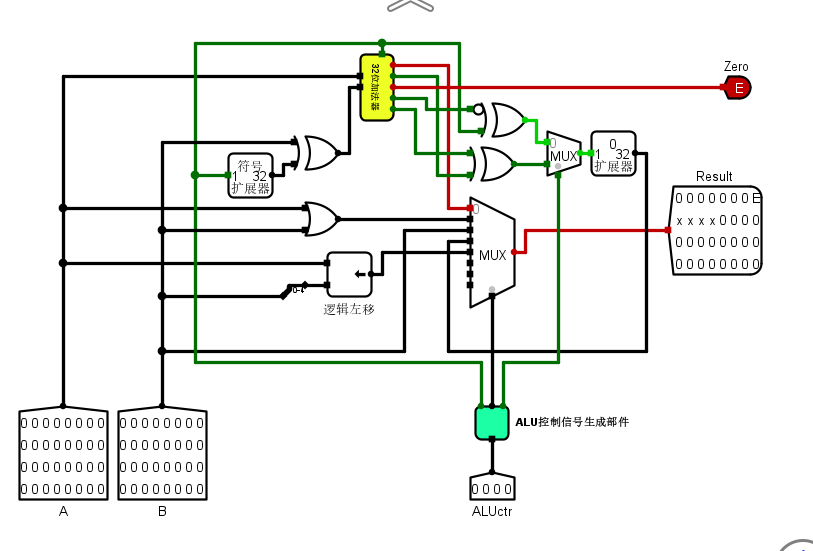
该部分是处理器中进行运算的部分，寄存器堆会根据刚才获取的指令中的寄存器信息完成对应寄存器的选择，同时结合控制信号实现寄存器操作，这里的busW是运算后的结果，用于数据的写回，这里有个特殊之处，ALUASrc可以选择输入的是PC，对于beq和jal指令已经有专门的加法器来实现PC的改变，这里将PC输入运算器是为了实现一些特殊的跳转指令，如可能需要比较两个值并根据比较结果来决定是否跳转，这时可能会将PC的值作为ALU的输入之一，与其他操作数进行比较运算



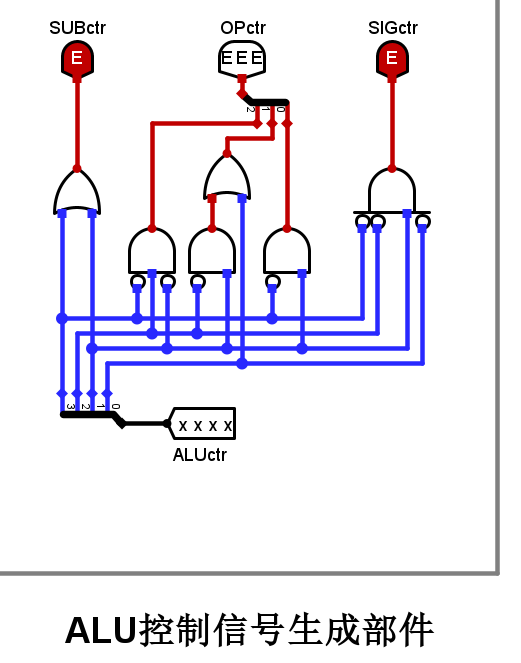
寄存器堆里电路如图所示，第一个通用寄存器的值恒为0，控制信号给出了读取哪一个寄存器以及写入到哪一个寄存器，以及写使能控制器来决定是读取还是写入数据，之后如果是读取会在两个端口输出对应的读取数据，具体的实现方式为5:32译码器来实现32个寄存器的选择



ALU结构如图所示



首先关注ALU控制信号生成部件，RISC-V支持减法和符号数操作，会先利用对应的ALUctr控制信号生成内部的控制信号

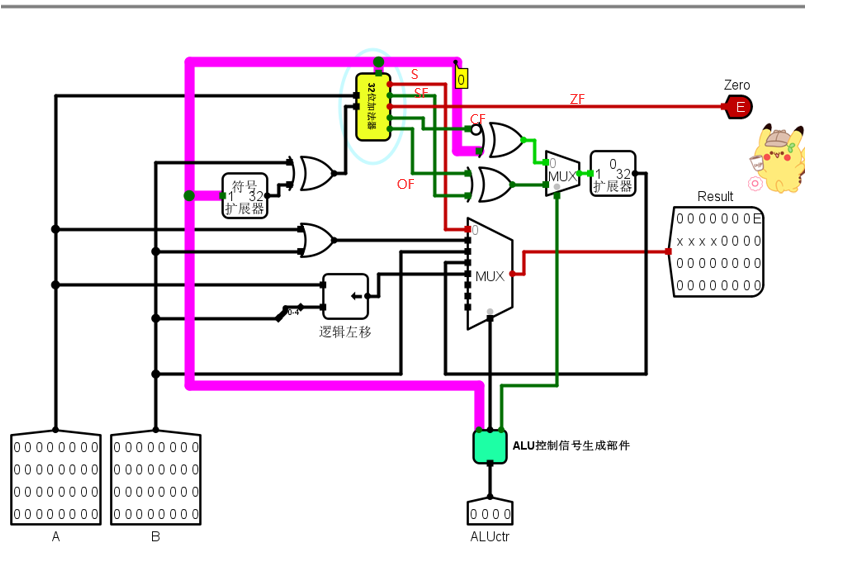


ALU可以实现加法、与、位移以及有符号数、无符号数的大小比较，其中红线传输的是减法控制信号，来控制加法器进行减法来实现大小比较，之后结合数据是有符号数还是无符号数利用异或门生成大小判断的结果

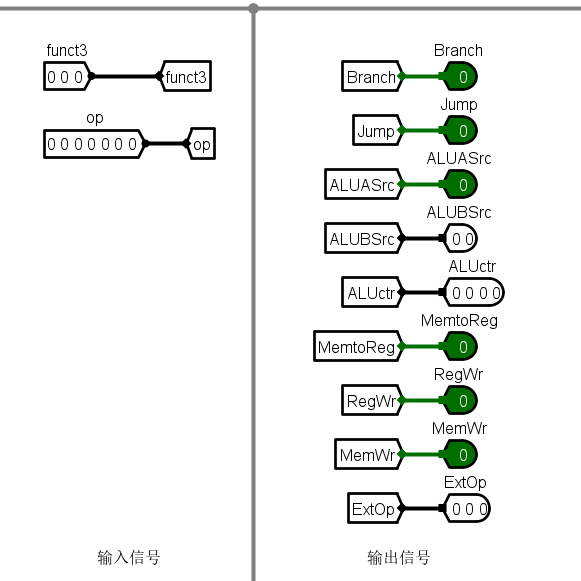
有符号数比较：没溢出的情况下SF=0说明前者大，SF=1说明后者大，溢出时情况相反，因此将OF和SF进行异或

无符号数比较：减法控制信号一直是1，CF=1说明发生借位，前者小，~CF=0，异或门结果是1，此时写回的结果是1；CF=0则写回0，此时前者大

最后比较的结果会写入rd对应的寄存器完成slt和sltu指令



控制器的信号如图所示，总共有9个控制信号，所有信号都已经在前面出现实现对应的指令功能



控制器的实现采用硬布线模式，利用组合逻辑根据输入的值直接获取需要输出的控制信号，输入的信号为OP和Funct3，之后输出9个对应的控制信号完成控制信号的释放



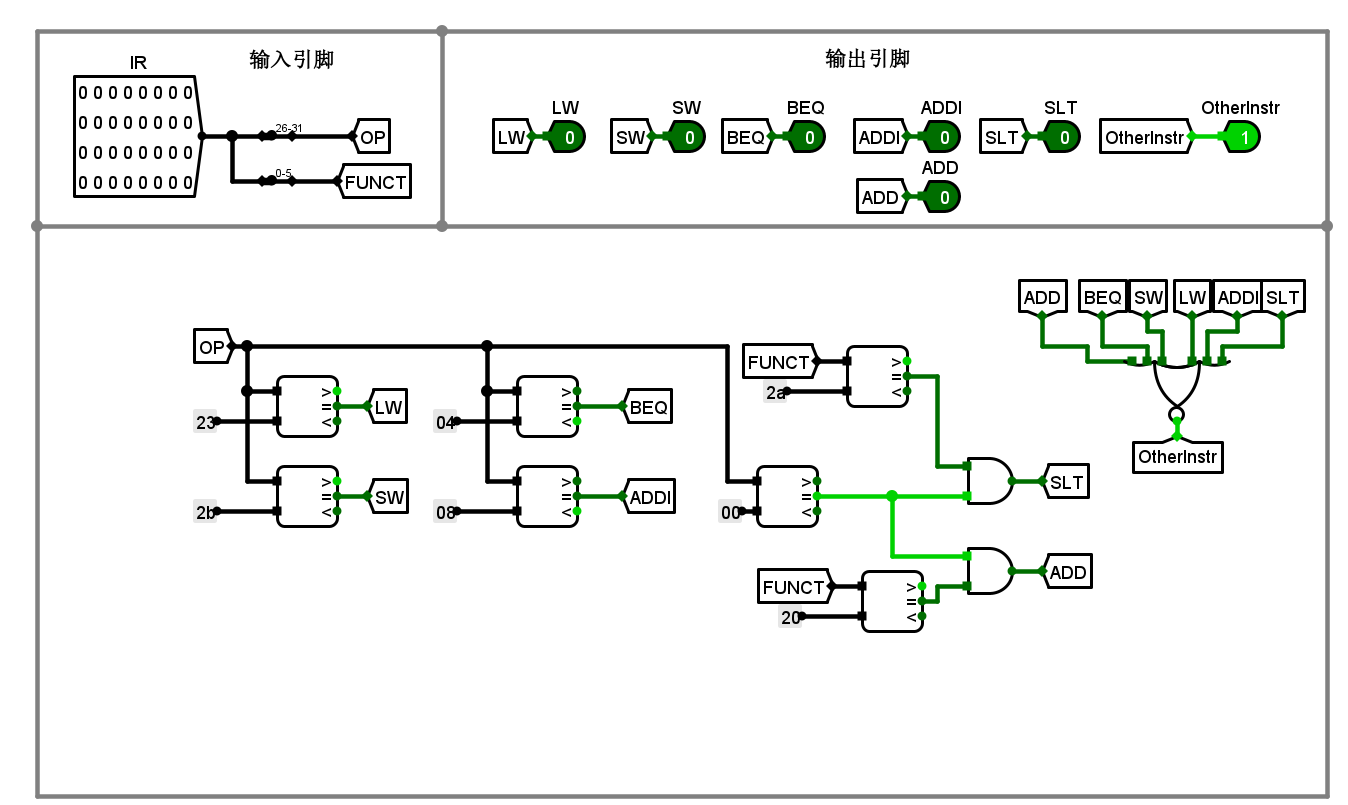
同理单周期MIPS电路，该单周期 RISC-V 处理器的控制器一般不会采用微程序控制器的方法设计

## 2.2设计实验

### 2.2.1单总线结构 MIPS 处理器（微程序控制器）（增加1条add指令）（6条指令）

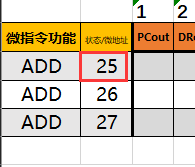
#### 1.修改指令译码器

先对指令译码器进行修改，增加一条add指令，查阅资料可知ADD指令的OP=0，FUNC=32（十六进制20）



#### 2.修改微程序入口查找逻辑

由所提供的表格“微程序自动生成（add指令）”可知ADD微程序入口是25



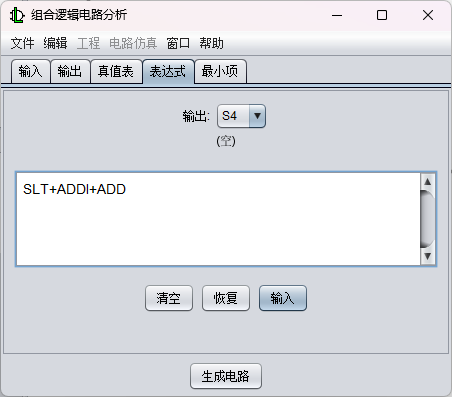
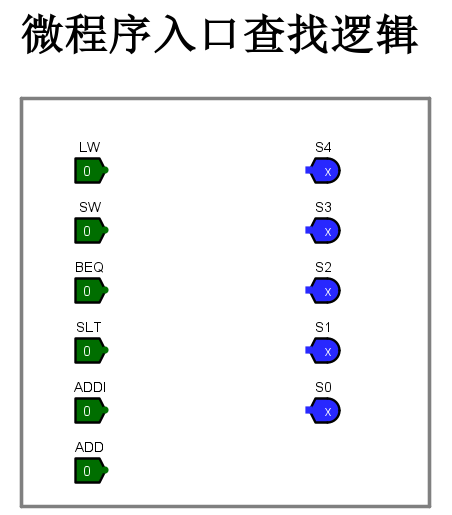
填写入表格“微程序入口查找逻辑自动生成（6条指令）”，之后将自动生成微程序入口查找逻辑公式。

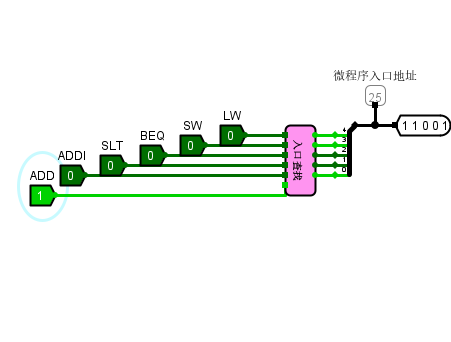
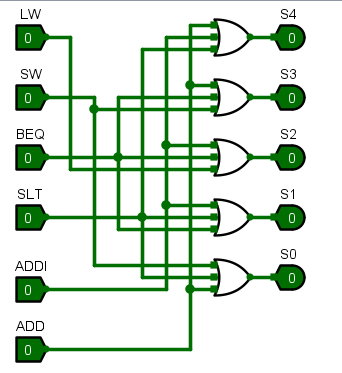




#### 3.在Logisim中生成逻辑电路

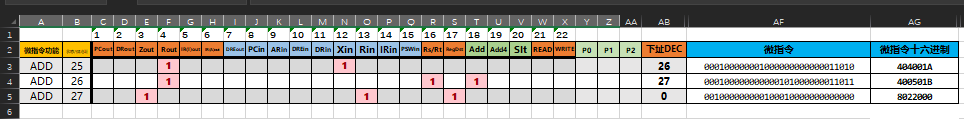
通过Excel就算出对应的逻辑表达式，之后利用Logisim的逻辑电路生成功能生成对应的组合逻辑电路，进行测试可以看到当指令为ADD时输出微程序入口地址为25

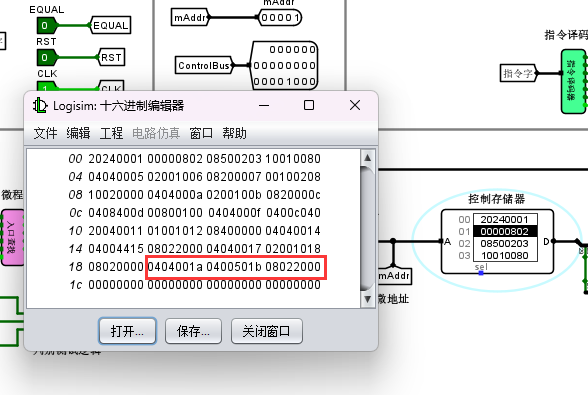
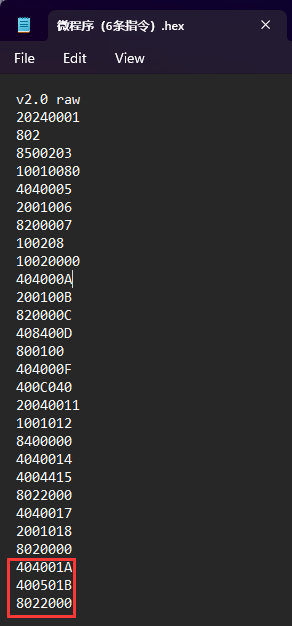




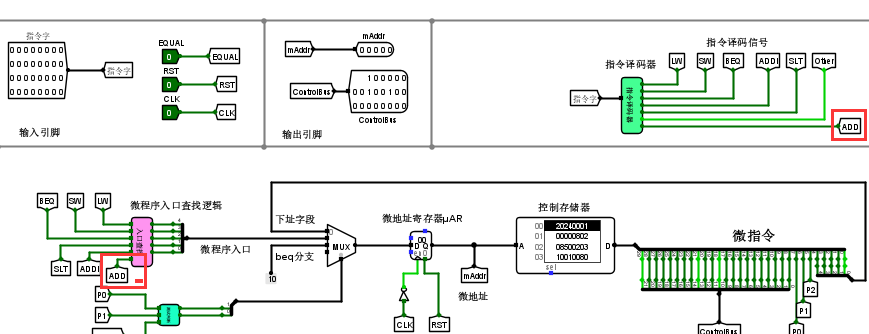
#### 4.修改控制存储器中的微程序

在“微程序自动生成（add指令）”中填写add指令执行周期里三个时钟周期的控制信号，之后自动生成微指令，之后将其写入控制存储器当中





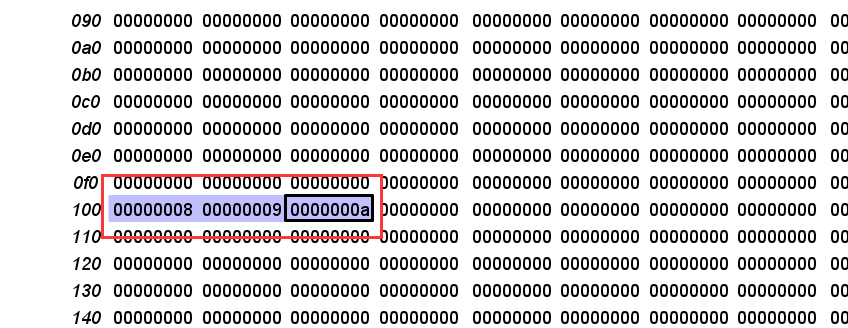
修改微程序控制器电路，增加ADD指令



程序测试

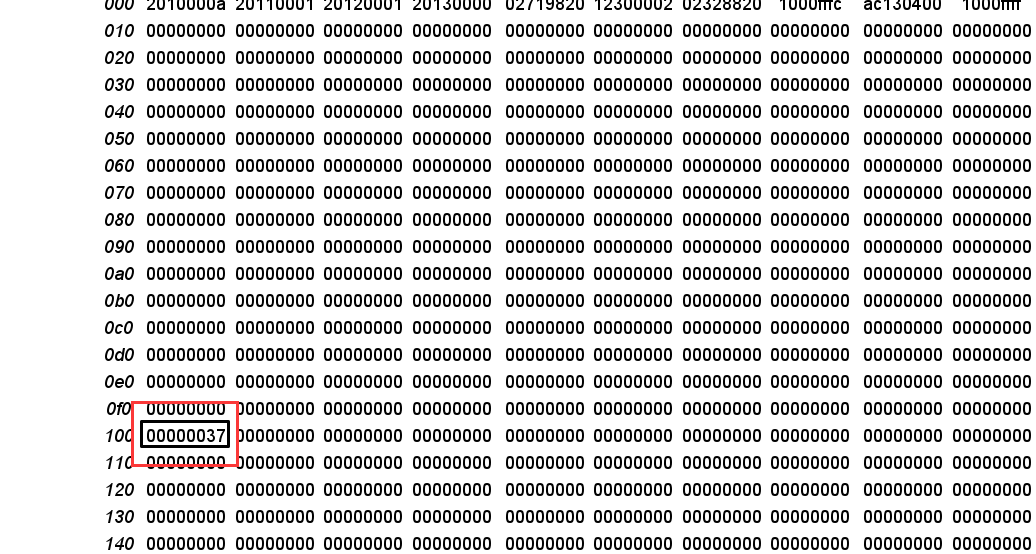
1.test2.hex

运行该测试文件，在100号开始的三个存储单元内可以看到出现了预期的数字，证明成功实现了add指令



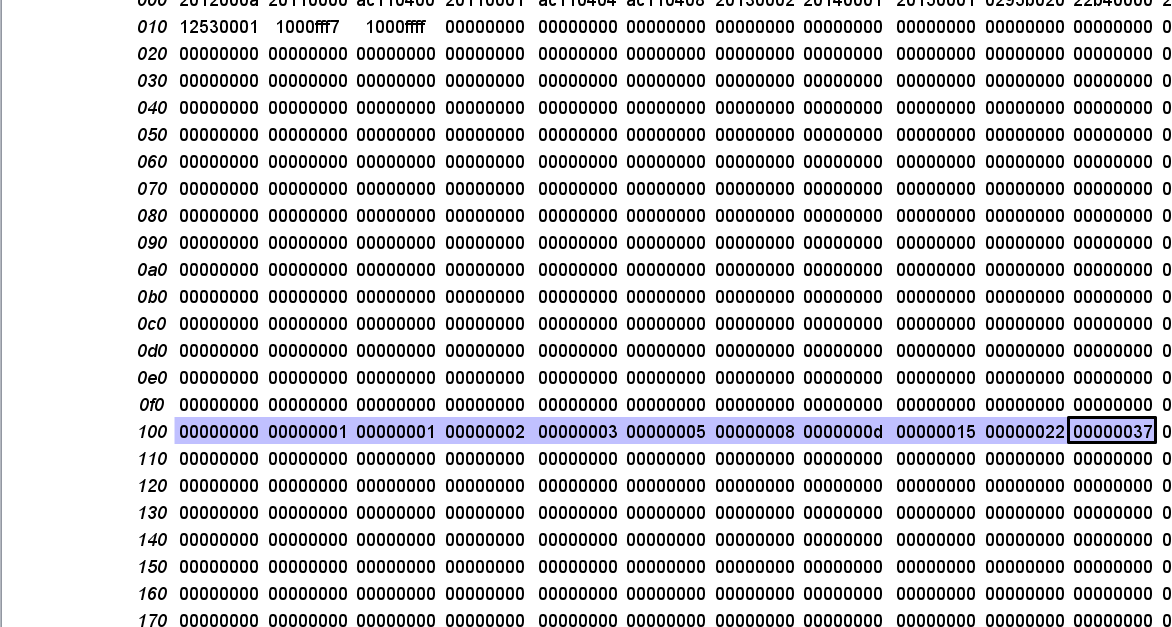
2.求累加和程序

累加和程序计算了从1到10的求和，求和结果是55，对应十六进制为37H，说明程序正确



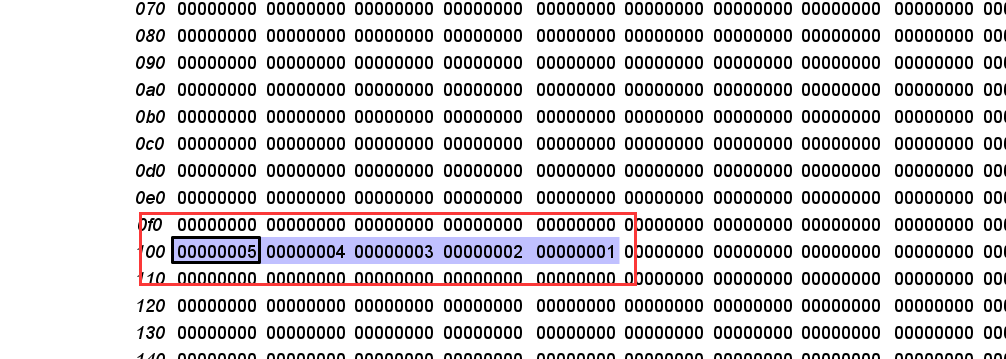
3.计算斐波那契数列

对应的斐波那契数列数列前11项成功被存储在了存储单元当中

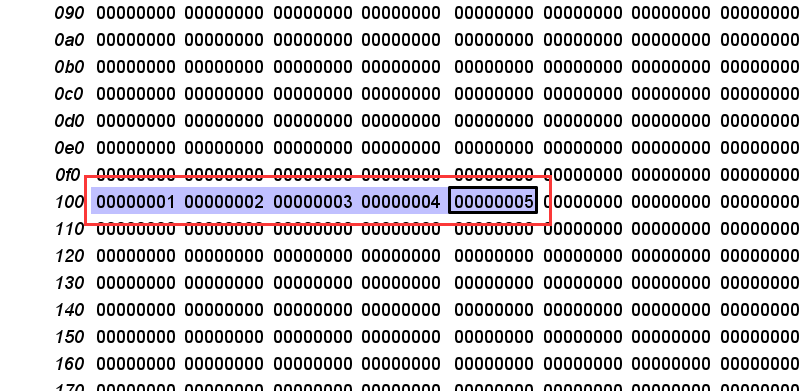


4.排序程序

降序排序



升序排序

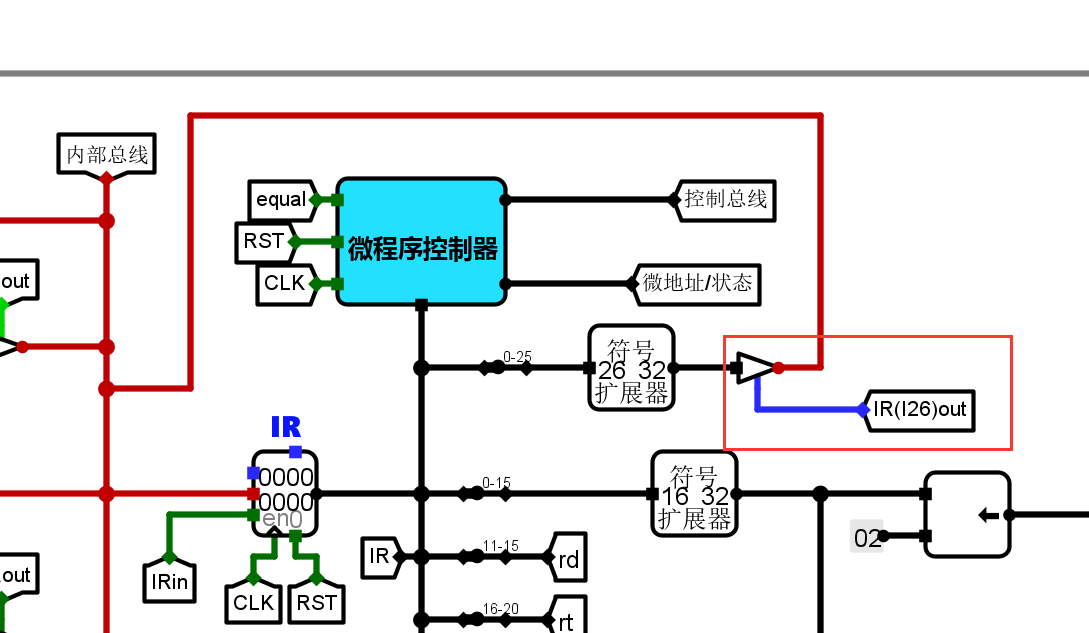


### 2.2.2单总线结构 MIPS 处理器（微程序控制器）（再增加1条j指令）（7条指令）

#### 1.修改数据通路

J型指令格式如图所示，原来的数据通路不支持J型指令，因此要先修改数据通路使得可以支持获取其中的26位立即数，在IR寄存器中获取低26位的立即数，之后为了能在总线上运输将位宽扩展到26位并新加一个三态门





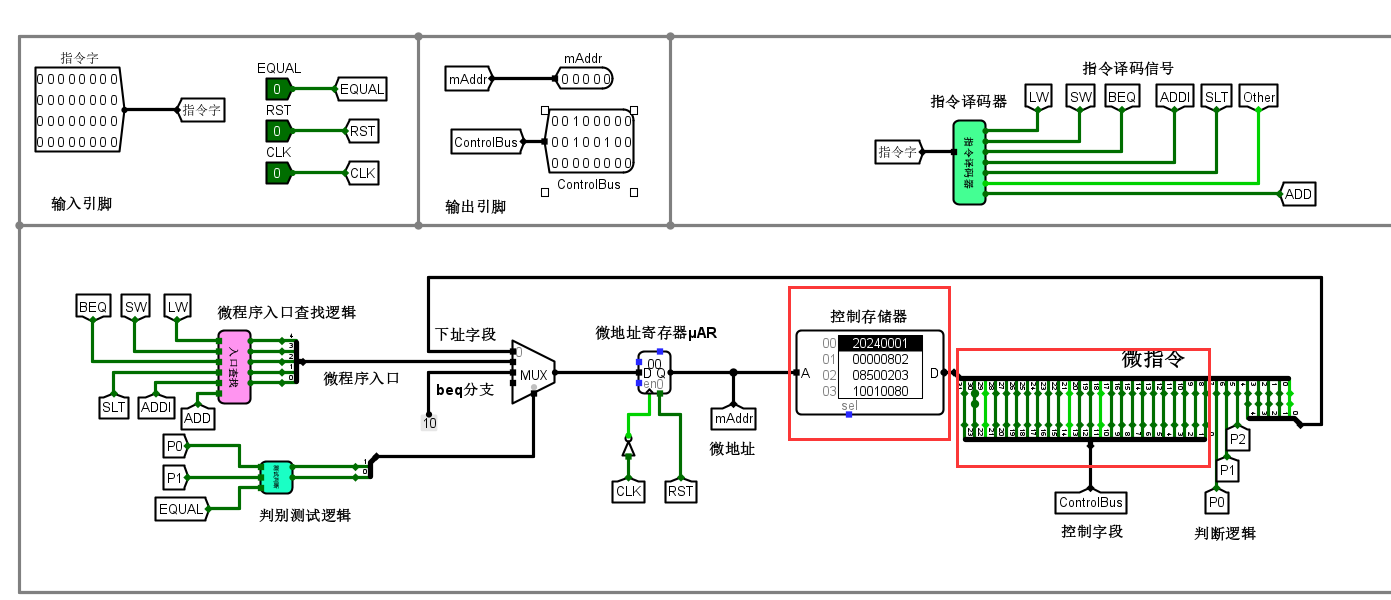
#### 2.修改ALU

ALU在7条指令的实际使用过程中没有使用减法，所以直接将减法SUB修改为无条件跳转J，之后实现(PC+4)31:28, imm26 <<2

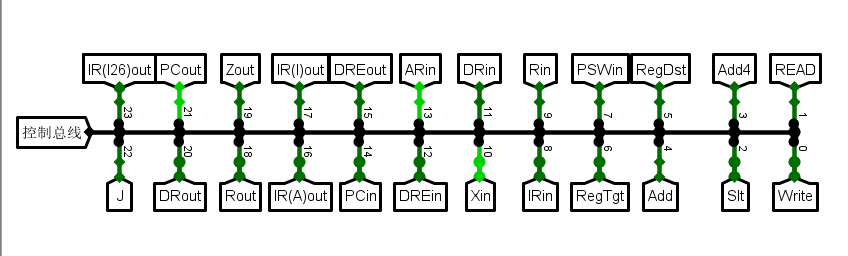


#### 3.修改控制总线

控制总线原来的位宽是22位，现在新增加了两个控制信号，所需要扩展为24位，先修改微程序控制器中的数据位宽，使其扩展为24位，同时也要修改控制存储器的数据位宽

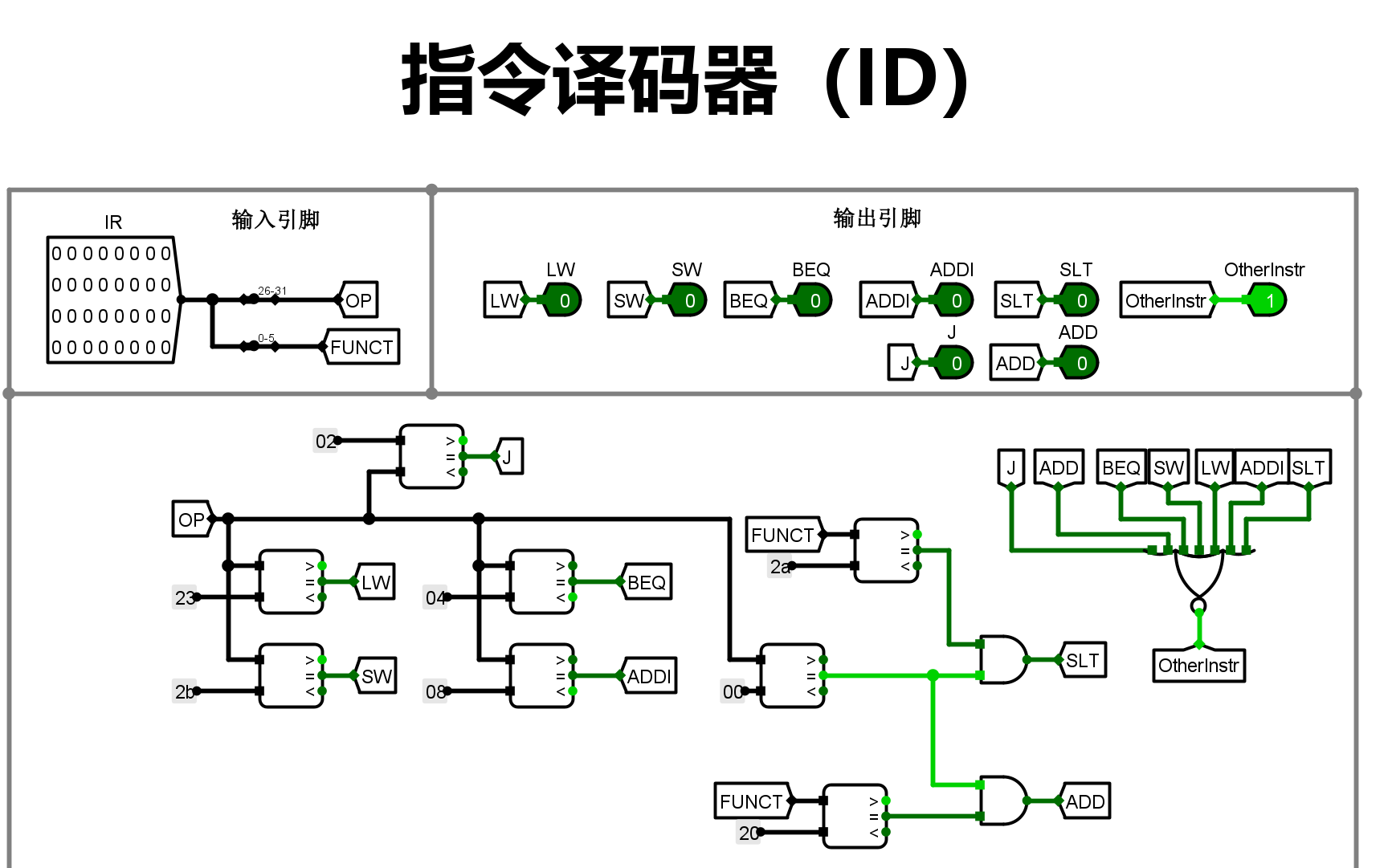


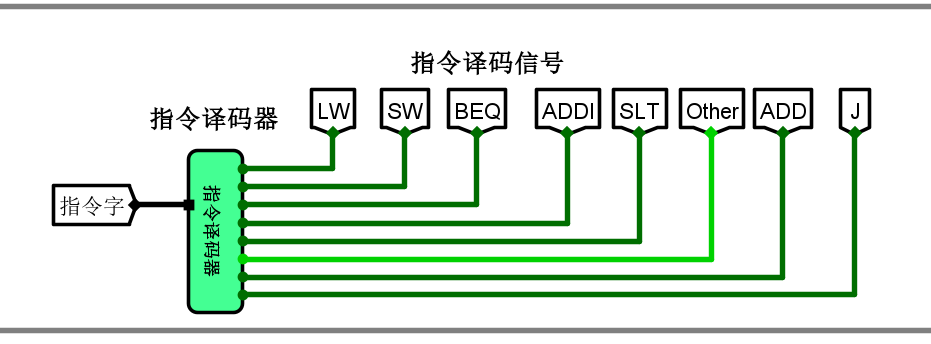
之后修改数据通路中的控制总线，扩展位宽为24位，将两个新的控制信号加入



#### 3.修改“指令译码器”电路

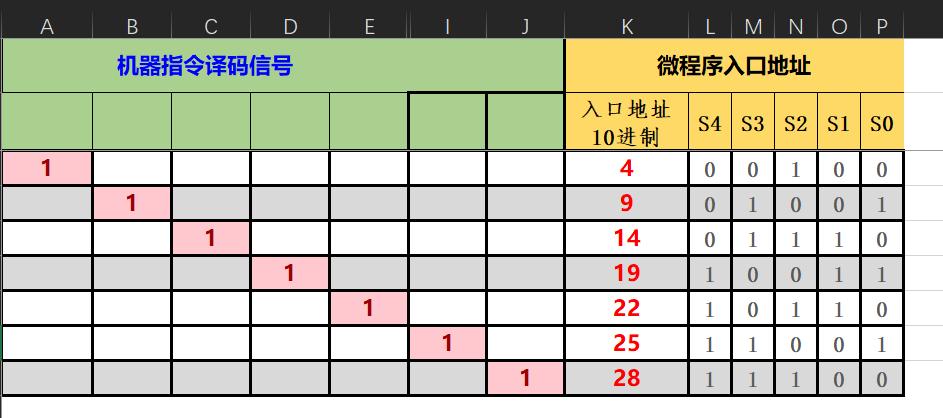
J指令的操作码是2，因此在指令译码器中新增一个J指令的译码

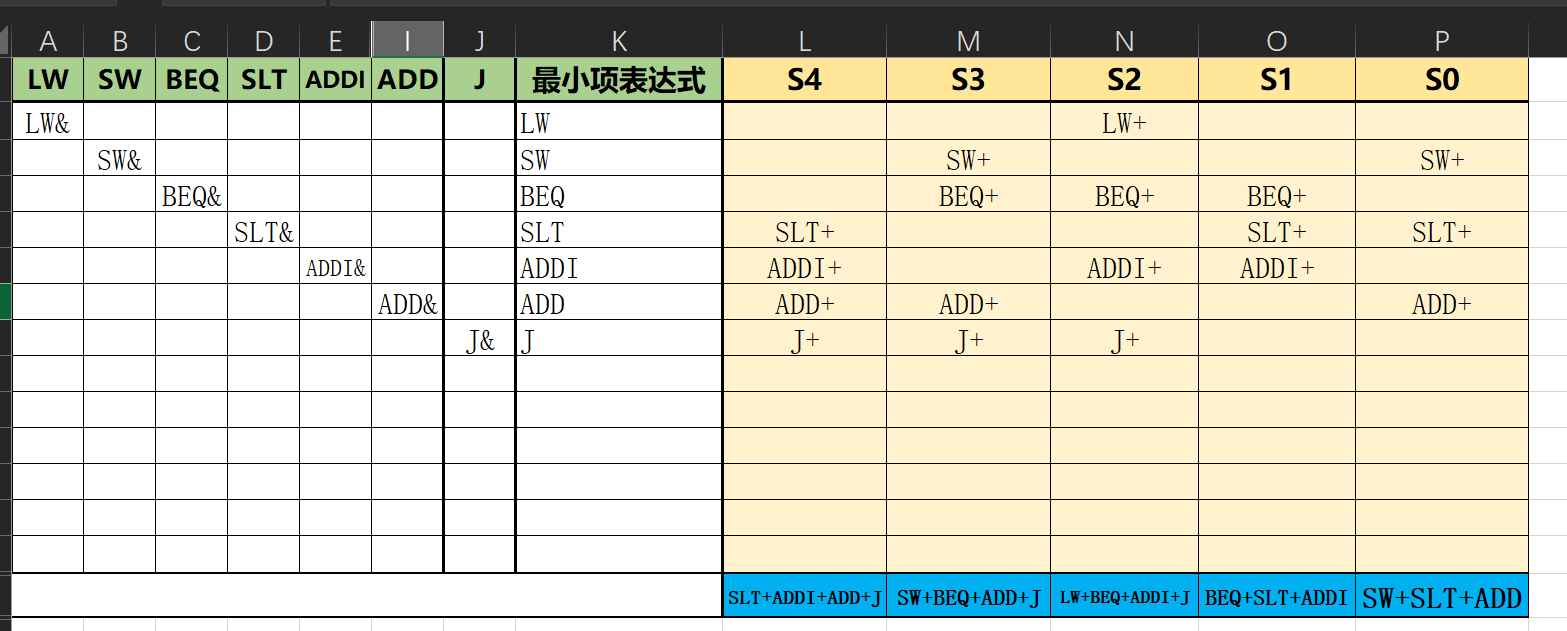




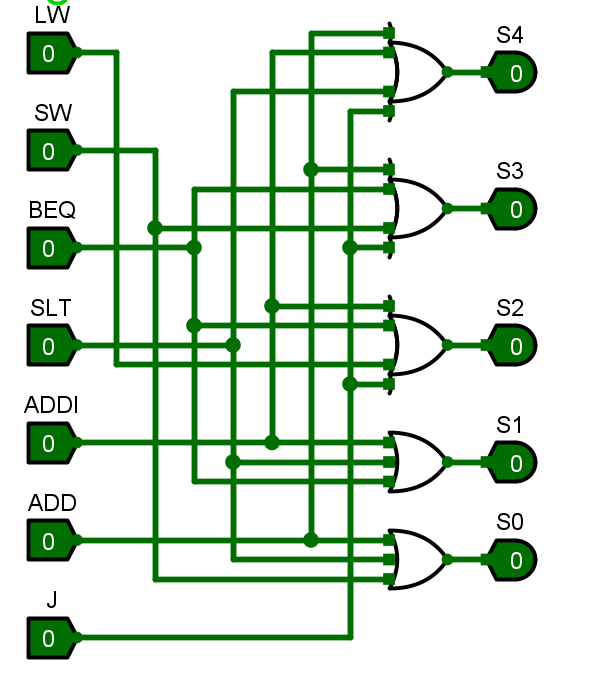
#### 4.修改“微程序入口查找逻辑”电路

J型指令的微程序入口是28，利用表格自动生成对应的微程序入口地址的逻辑表达式





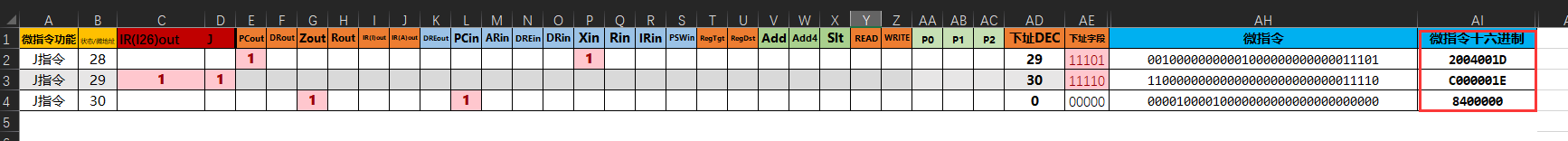
之后在Logisim中生成对应的组合逻辑电路



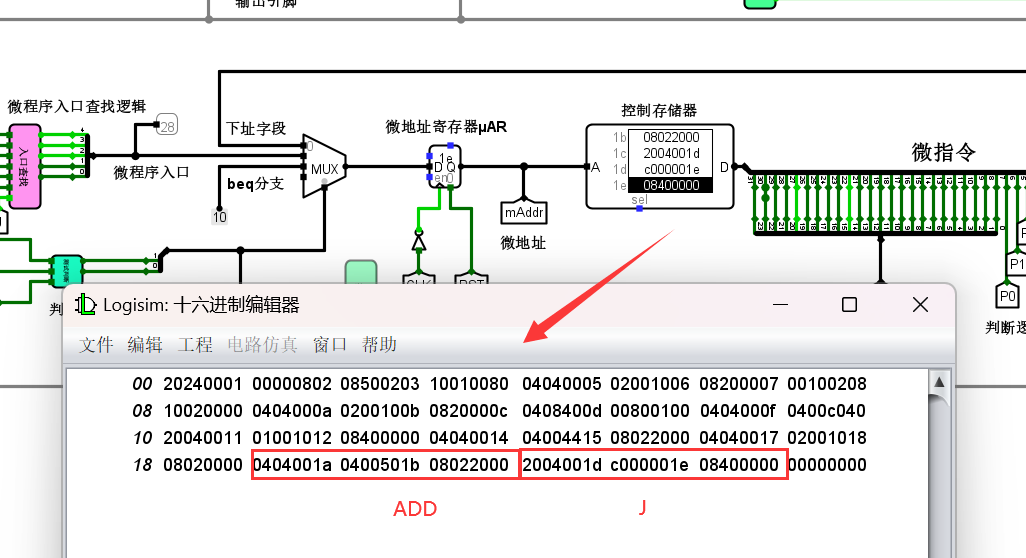
#### 5.修改“增加j指令的微程序”电路

微指令由30位扩展到32位，但是之前的6条指令的IR（I26）out和J都是0，并且Logisim的存储器输入数据时会按照.hex文件按行读取，如果读取到的宽度小于存储器的数据位宽，所以会自动高位补0，因此之前指令的控制存储器不需要修改，只需要新增J指令的微程序即可

J指令由三个状态构成，第一个状态将PC经过总线运输到存储器的X寄存器锁存，第二个阶段将立即数经过总线输入运算器的Y端口，同时给出J指令让运算器实现运算，第三阶段将运算结果打回PC实现指令的跳转



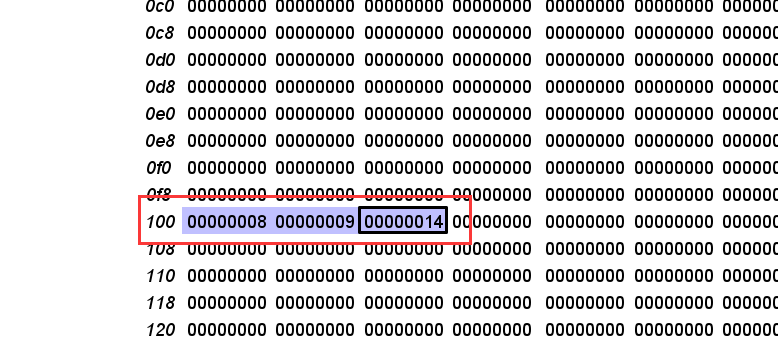
最后将微程序保存到控制存储器中，即可实现新增加一条J指令



程序测试

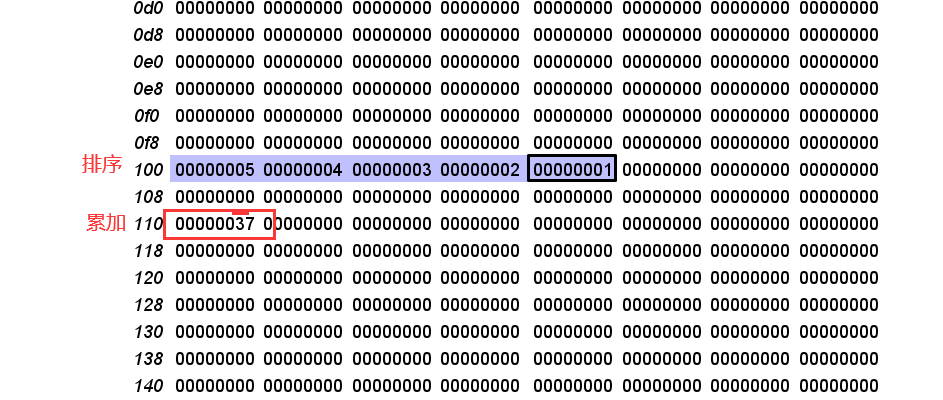
1.test3.hex

在102号存储单元看到14H，说明J指令被正确执行



2. sort\_sum\_mips\_bus.hex

在100-105号存储单元可以找到降序排序后的数组，在110号存储单元可以看到累加的结果，说明程序正确

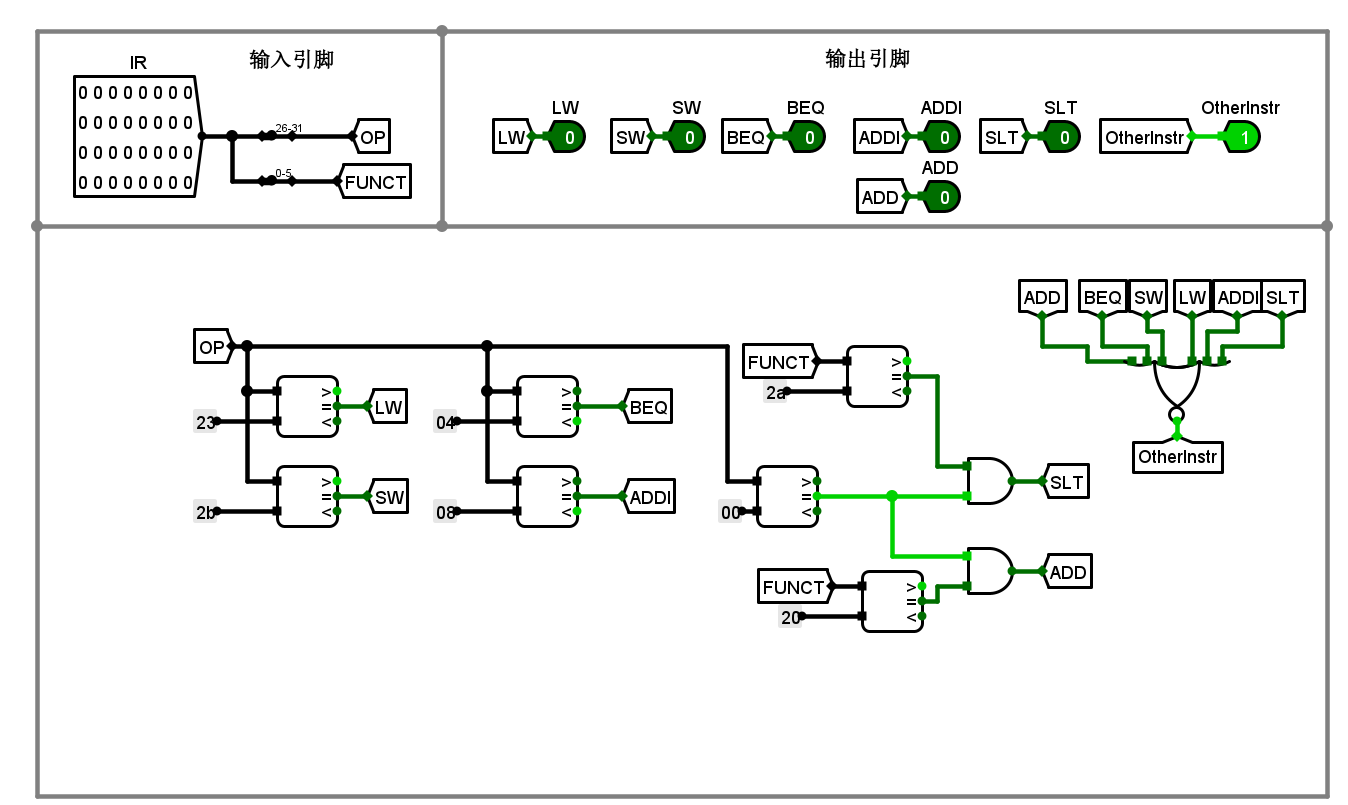


## 2.3挑战性设计实验

### 2.3.1单总线结构 MIPS 处理器（硬布线控制器）（6条指令）

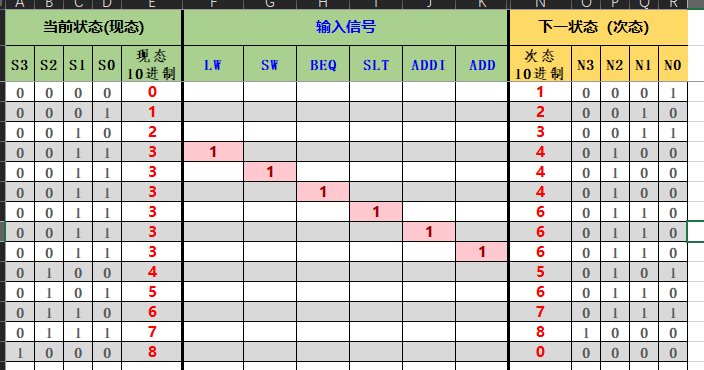
#### 1.修改指令译码器

先对指令译码器进行修改，具体步骤完全同理于微程序控制器的指令译码器，增加一条add指令，查阅资料可知ADD指令的OP=0，FUNC=32（十六进制20）

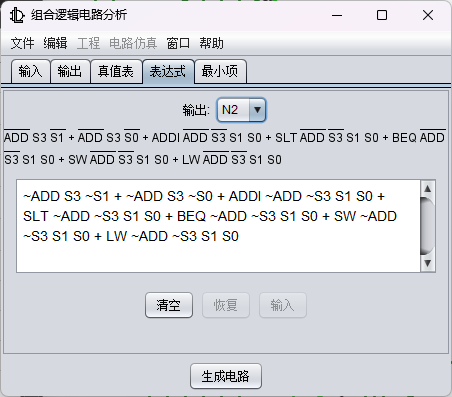
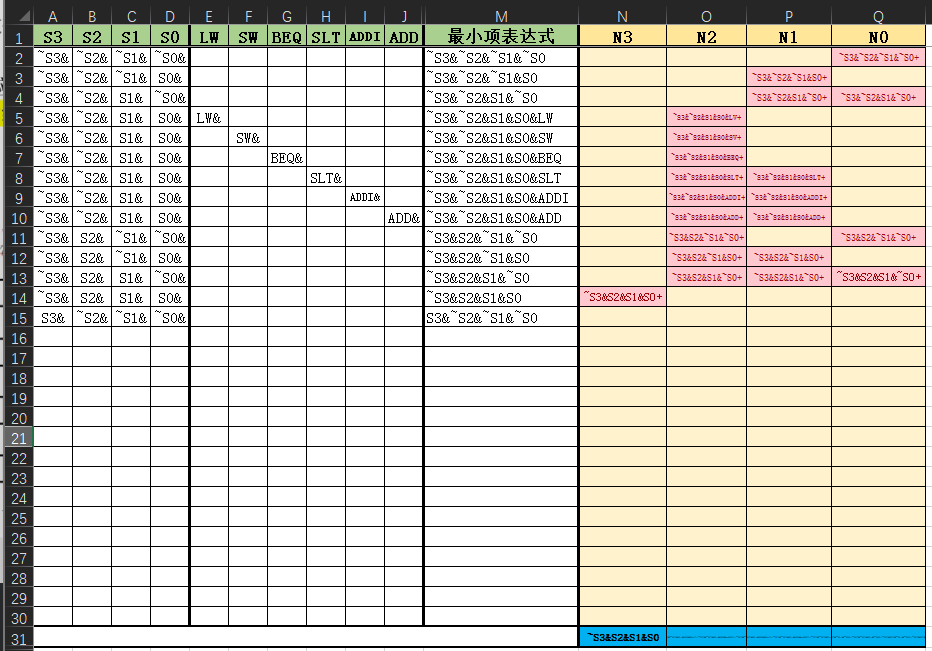


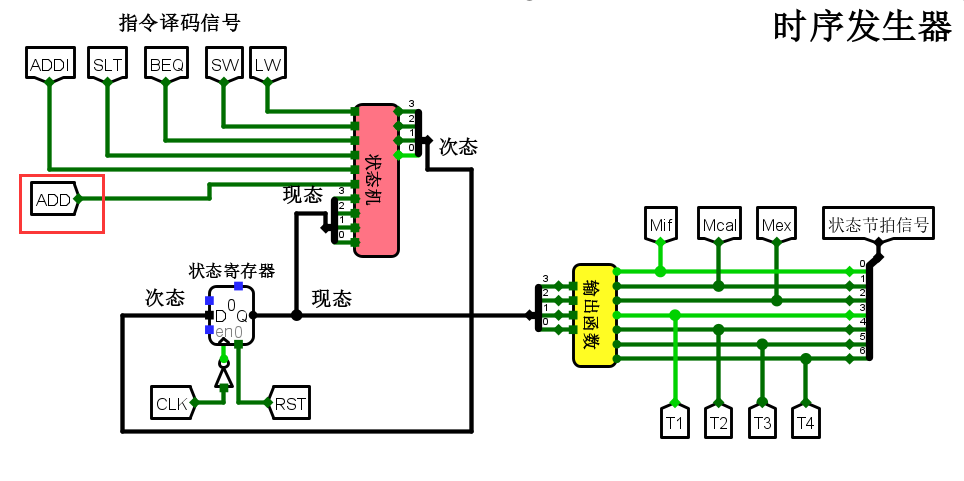
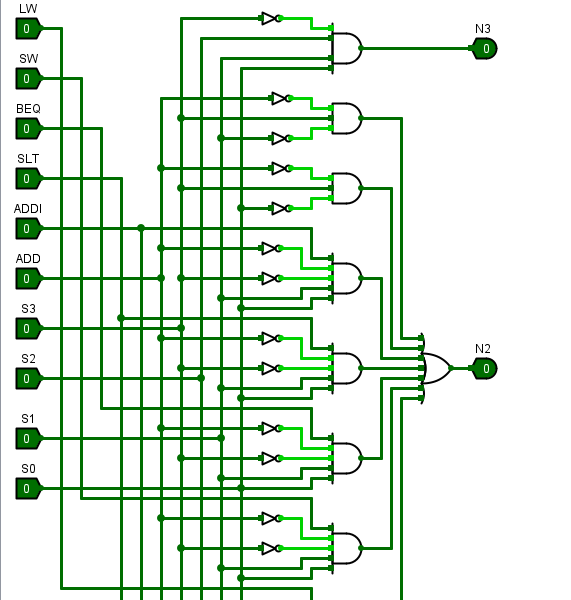
#### 2.修改状态机

ADD的状态跳转和ADDI相同，都会在S3直接跳转到S6阶段，所以在当前状态为S3、输入信号为ADD时的下一状态应该是S6



由Excel生成逻辑表达式，之后复制到Logisim生成组合逻辑电路

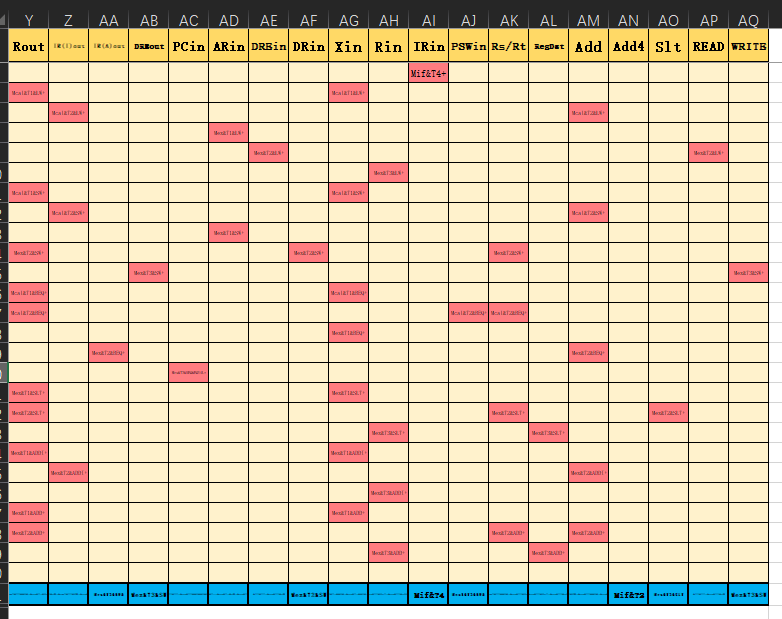




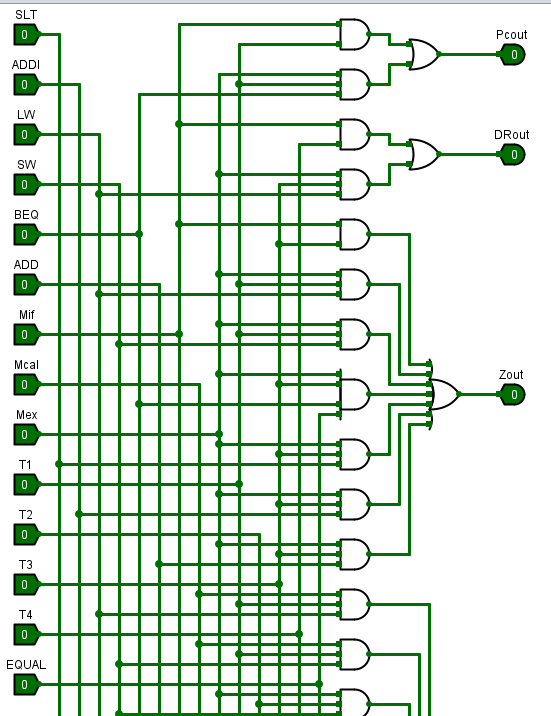
#### 3.修改硬布线控制器组合逻辑单元

利用Excel生成逻辑表达式，ADD指令执行周期三个状态如图，第一个状态将rs寄存器中的数据流入ALU的X寄存器所存，第二个状态将rt寄存器的内容送入ALU进行加法运算，第三个状态将运算结果写入目的寄存器rt

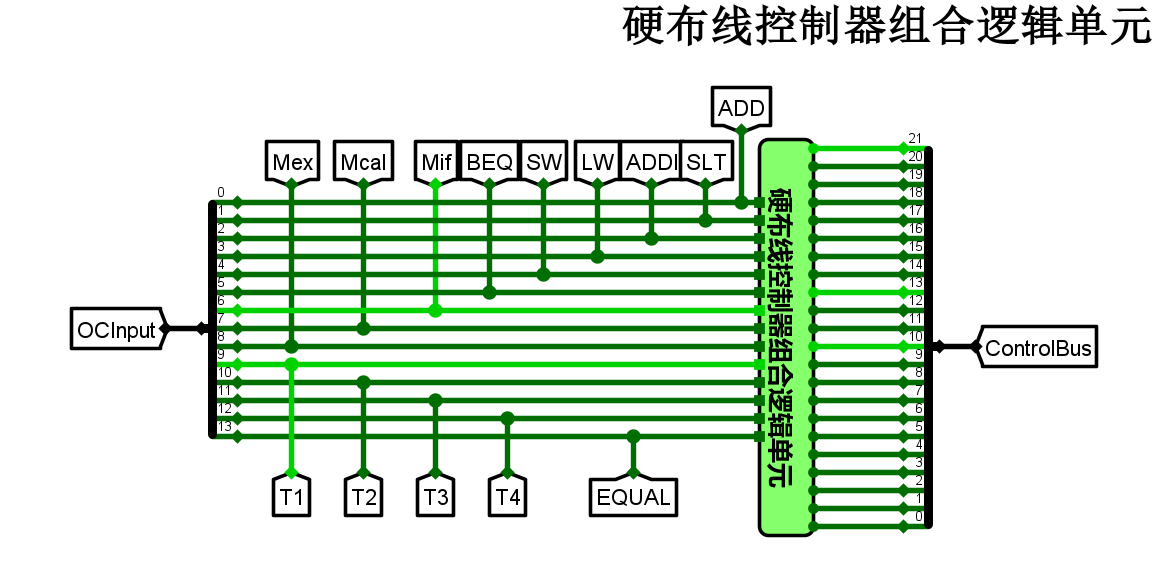




之后生成新的组合逻辑单元电路



修改电路封装和外部的连接，至此完成ADD指令的添加

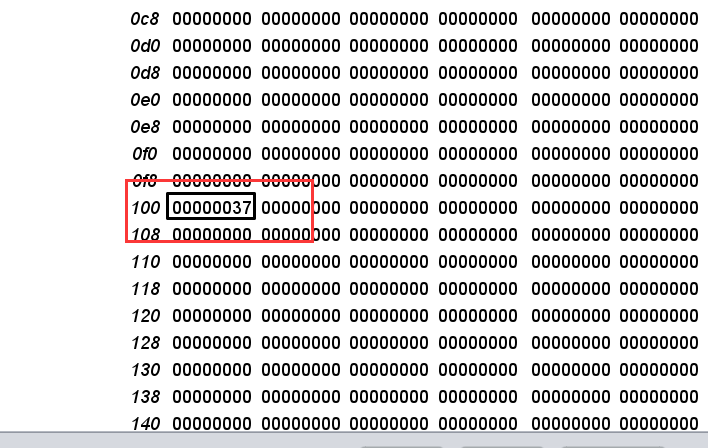


程序测试

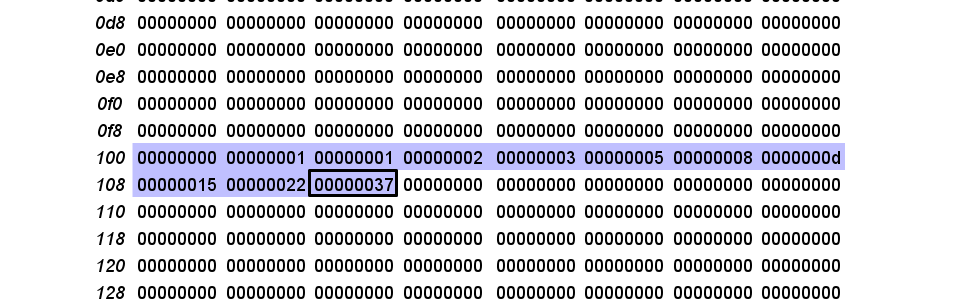
1.test1.hex



2.求累加和程序

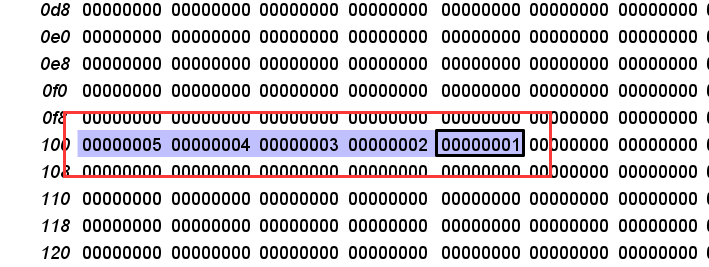


3.计算费波那契数列程序

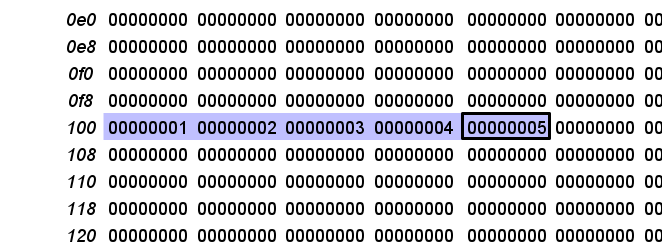


4.排序程序（降序排序、升序排序）

降序排序



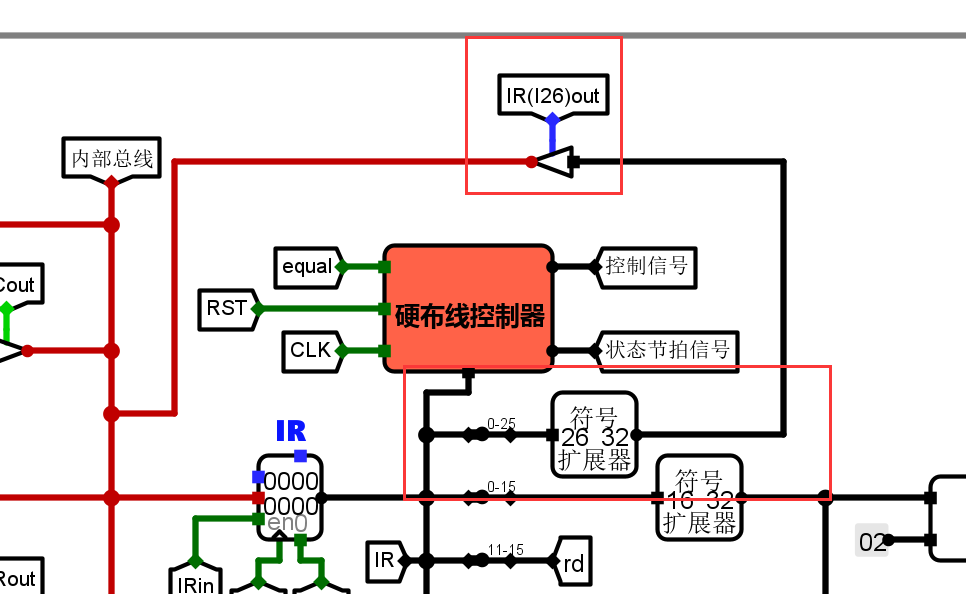
升序排序



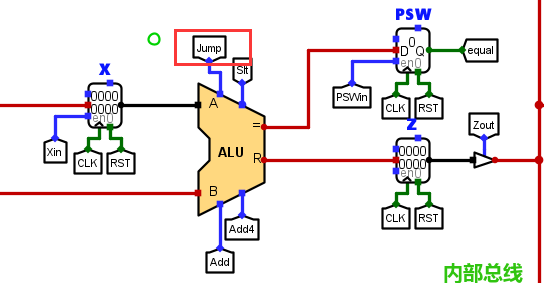
### 2.3.2单总线结构 MIPS 处理器（硬布线控制器）（7条指令）

#### 1.修改数据通路

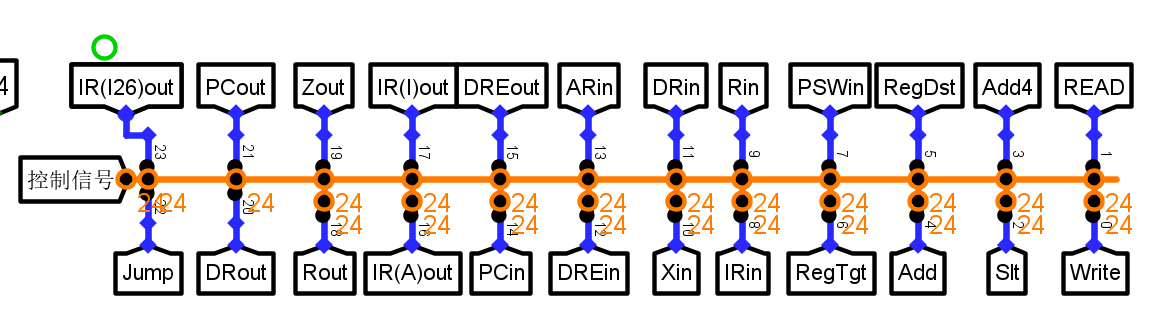
修改后的数据通路和7条指令的微程序控制器数据通路基本相同，首先增加一个提取J指令的低26位立即数，并增加一个三态门用于状态控制



在运算器的A输入处增加一个Jump控制信号

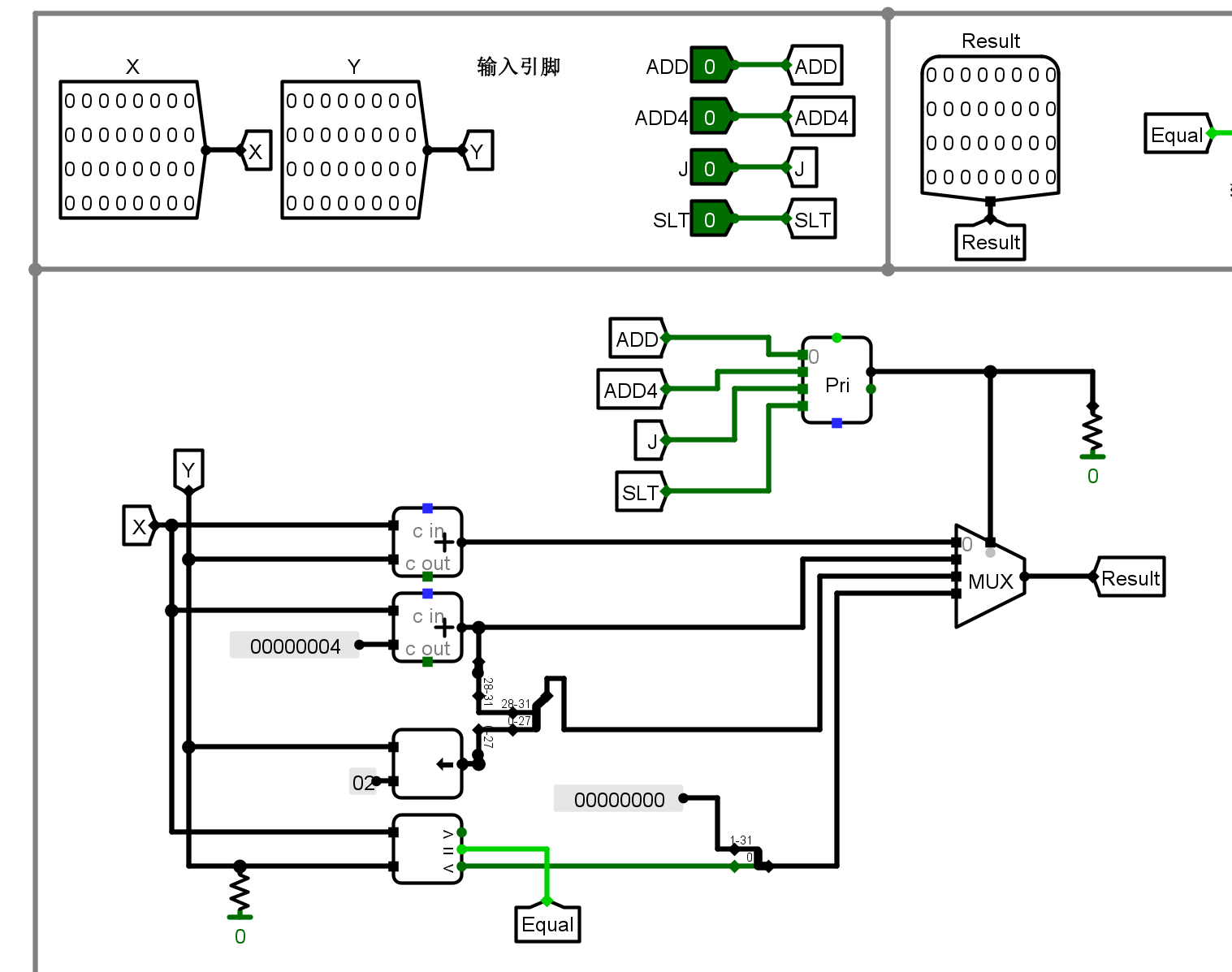


在控制总线中加入两个新的控制信号，实现控制总线的扩展



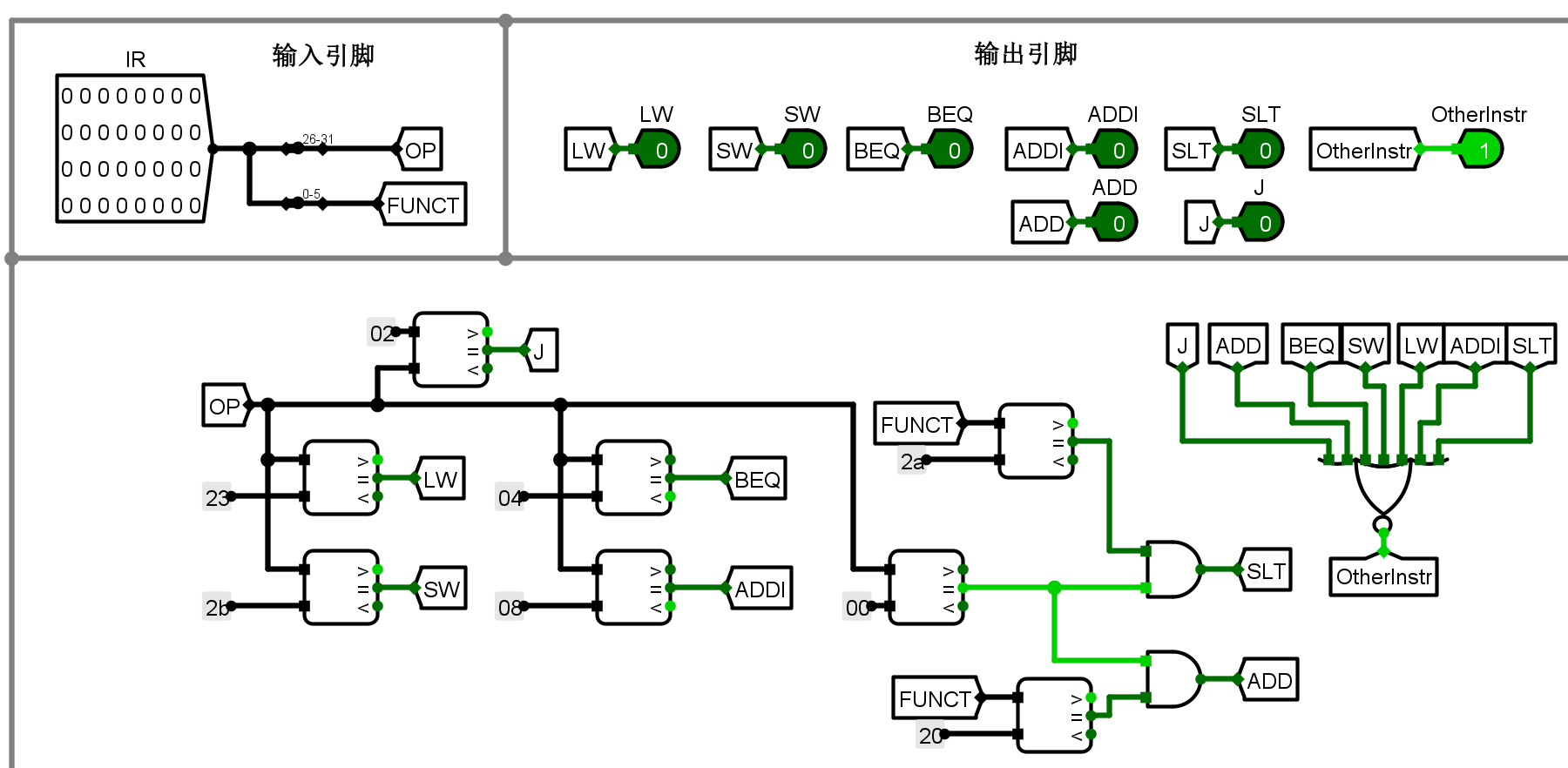
#### 2.修改ALU

ALU修改和对应的微程序控制电路完全相同



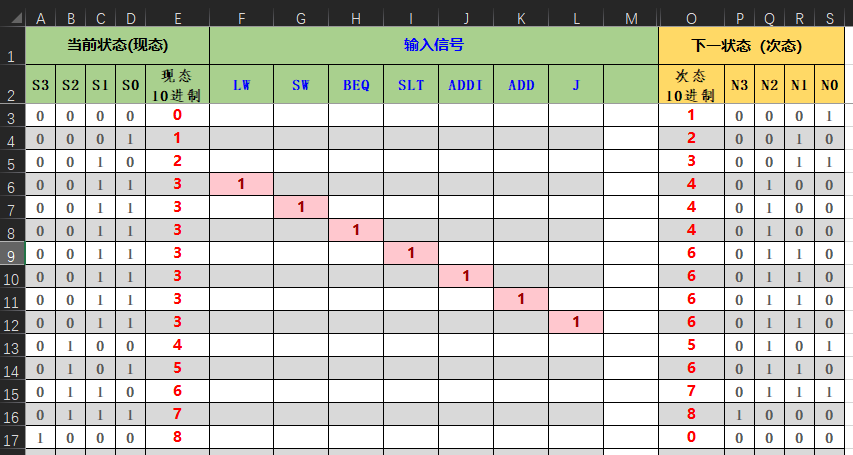
#### 3.修改指令译码器

同理修改后的指令译码器和对应微指令控制的电路也完全相同

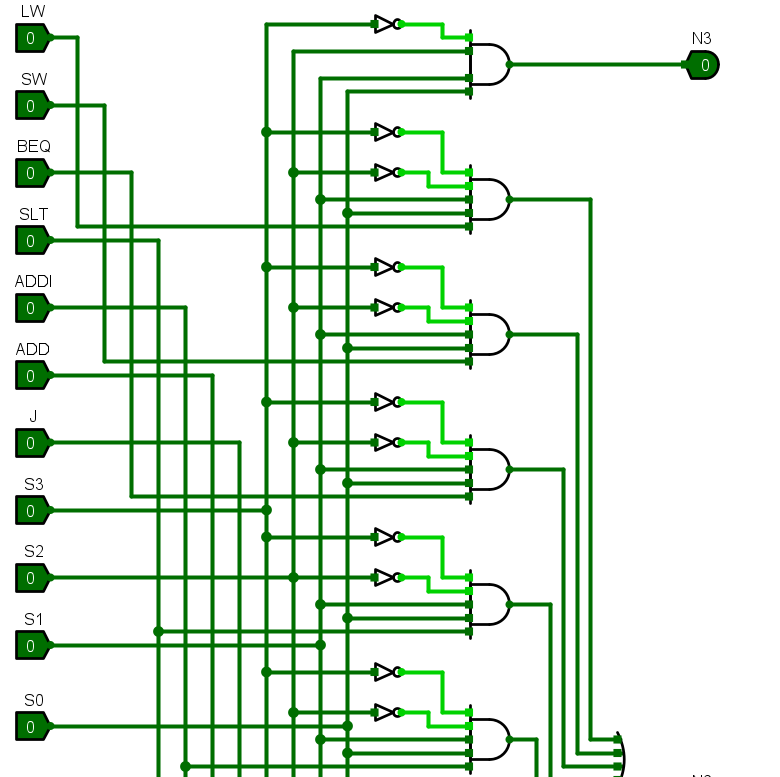


#### 4.修改状态机

J型指令无需计算周期，因此直接从S3跳转到S6

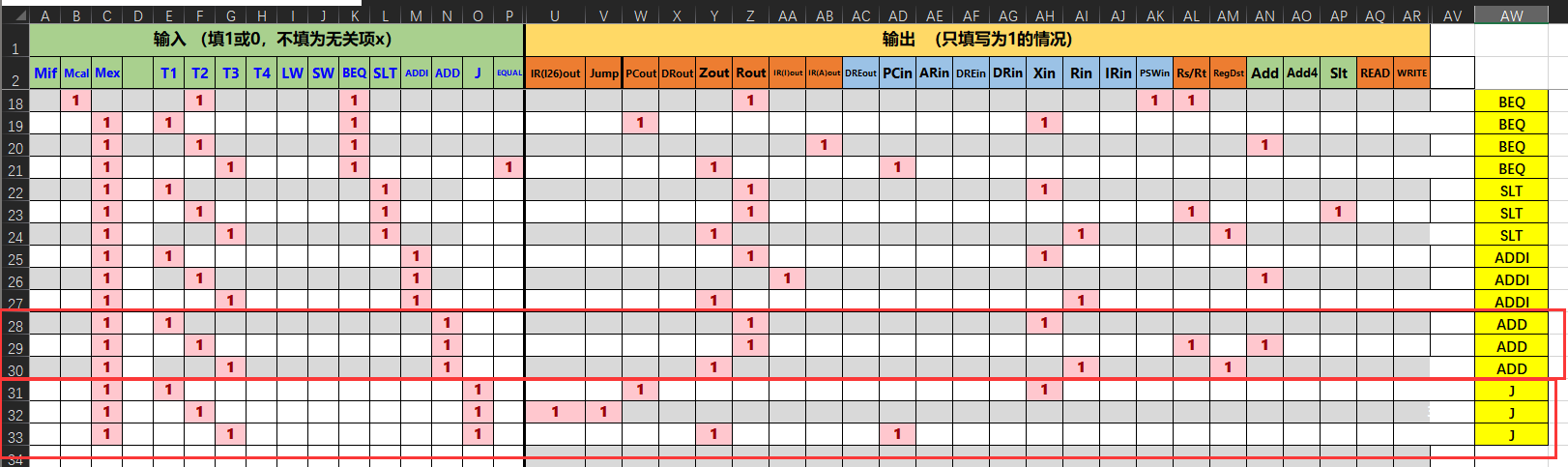


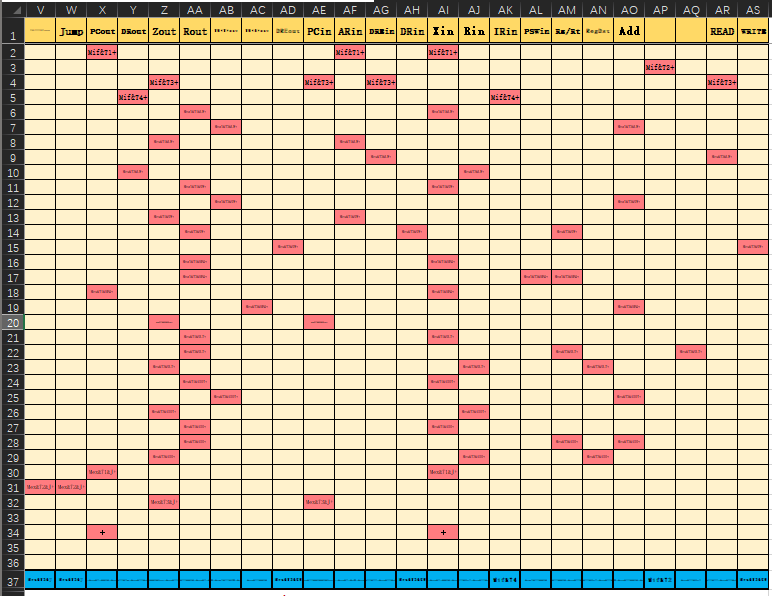
之后生成对应的Logisim电路



#### 5. 修改硬布线控制器组合逻辑

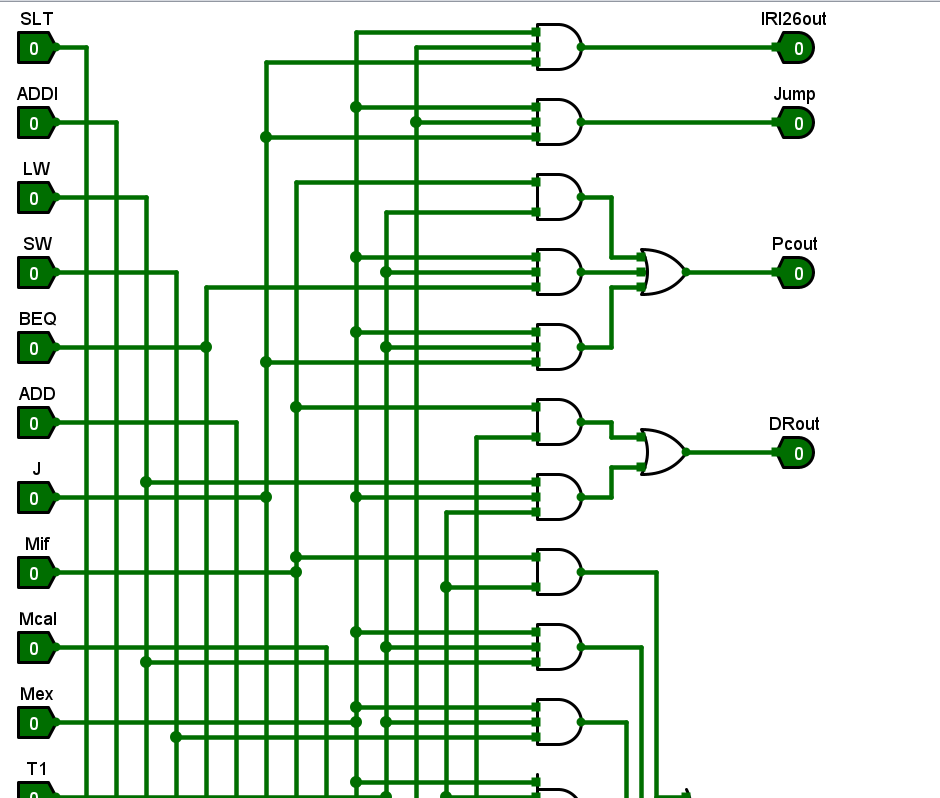
J指令在硬布线的三级时序中执行周期内的三个时钟周期分别依次完成以下操作：第一个时钟周期将PC里的内容流入运算器X端口的寄存器锁存；第二个时钟周期将低26位的立即数流向运算器Y端口并给出J控制信号使运算器完成PC值的计算并存储到Z寄存器进行锁存；第三个时钟周期将Z里的运算结果流入PC，因此对应的控制信号在下图中给出



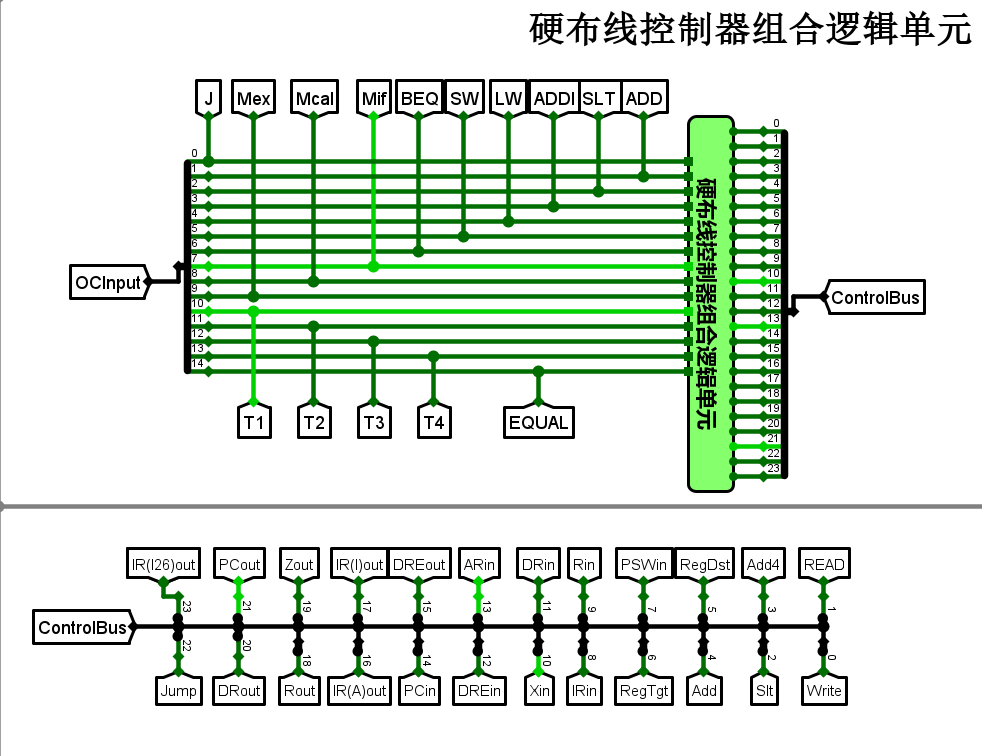
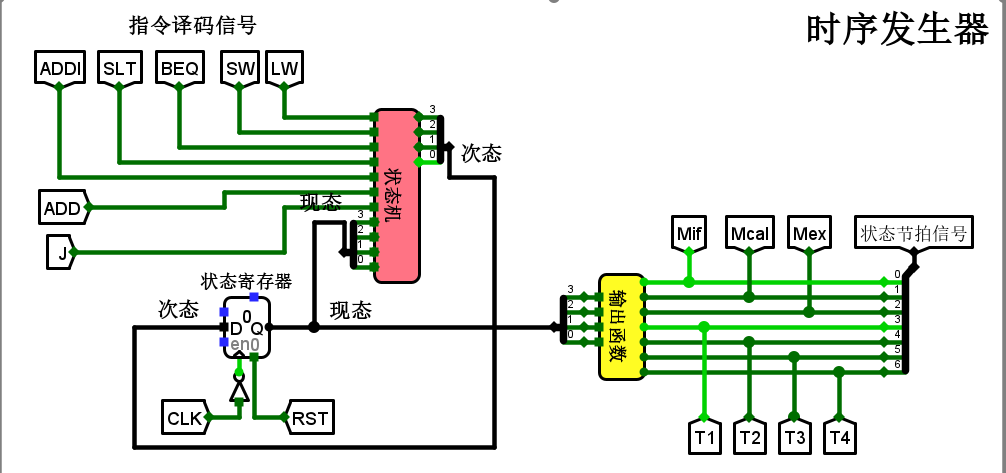


（这里的Excel好像有bug（？），会自动生成两个加号，要撤销保护模式去掉两个加号，或者在Logisim中去掉多余的加号）

生成的组合逻辑电路



之后适当修改细节后完成电路的改装

程序测试

1.test3.hex



2. sort\_sum\_mips\_bus.hex

找到排序和累加的结果，证明程序正确

