**第五次实验报告-221900180-田永铭**

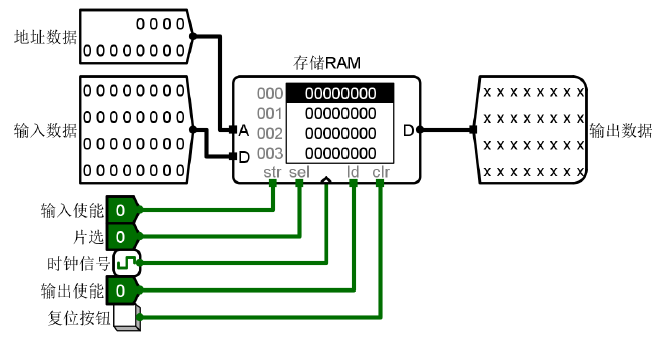
**实验目的**

1. 理解随机访问存储器RAM和只读存储器ROM的操作原理
2. 理解RISC-V指令类型和指令格式
3. 掌握使用Logisim软件实现取指、指令解析、立即数扩展、操作数存取的方法

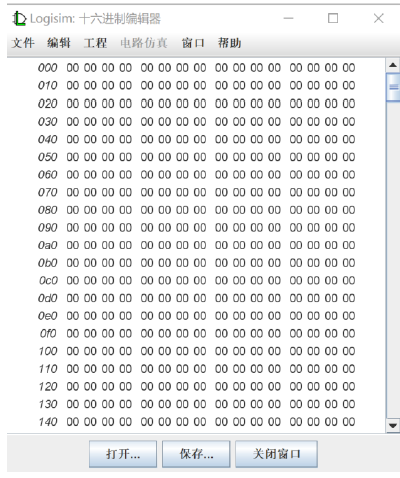
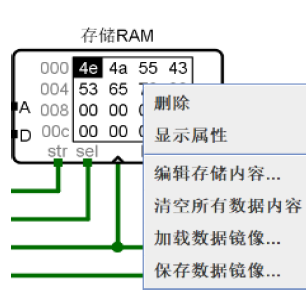
**实验1 存储器的写入与读取**

**（一）实验原理**

Logisim中RAM器件的地址位宽最多可设置为24位，数据位宽最多可设置为32位。在属性窗口的数据接口中有三种不同的工作模式。若设置为“分离的加载和存储引脚”模式， 则有两个数据端口分别连接输入数据和输出数据（如下所示）。



Logisim中RAM和ROM器件的数据输入可以采用Logisim十六进制编辑器和直接读取二进制编码文件的方法实现。把鼠标移到存储器组件上，点击鼠标右键，弹出菜单框，选中“编辑存储内容”，打开Logisim十六进制编辑器，可按照存储器设置的数据位宽，直接使用键盘输入数据。输入数据后，点击保存按钮，可把输入的数据保存到数据镜像文件中。当需要从数据镜像文件中加载存储时，在RAM器件上单击鼠标右键，菜单中选择“加载数据镜像文件”或在Logisim十六进制编辑器中打开数据镜像文件直接读入文件内容到存储器。



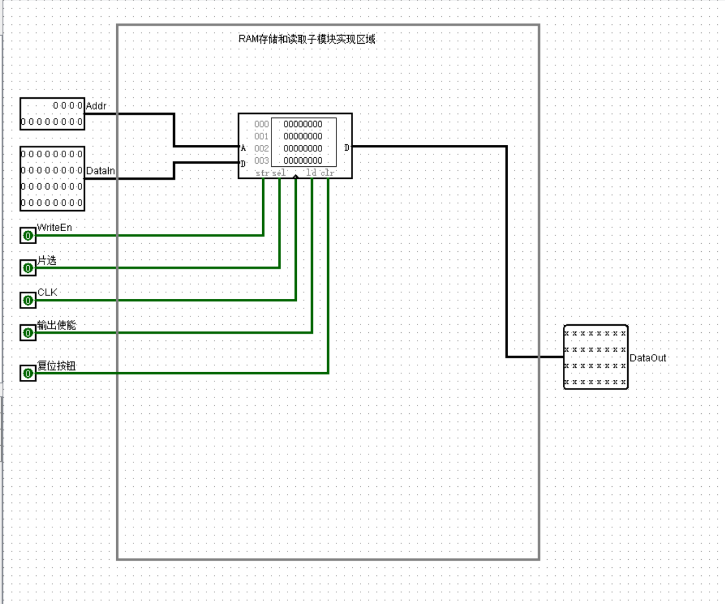
注意，当设置数据位宽为32位时，RAM采用按字编址方式（32位），而不是采用按字节编址方式。此次实验任务要求在RAM存储子电路中放置一个RAM组件，并设置地址位宽为12位，数据接口模式为“分离的加载和存储引脚”模式，顶层的测试部分会自动向RAM中写入数据，前16个时钟周期为写入测试，后16个时钟周期读取测试。通过观察本地验证区域的RAM测试结果验证自己的RAM存储实现正确。

**（二）实验步骤**

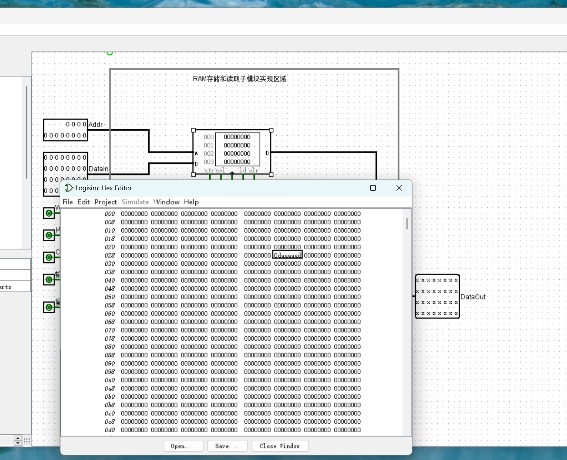
1.在电路中添加一个RAM，设置address bit width为12位，设置data bit width为32为，设置data interface为separate load and store ports。

2.添加WriteEn、片选、CLK、输出使能、复位按钮共5个输入引脚，并且给这些引脚加上标注。

3.按照下图方式摆放和连接。

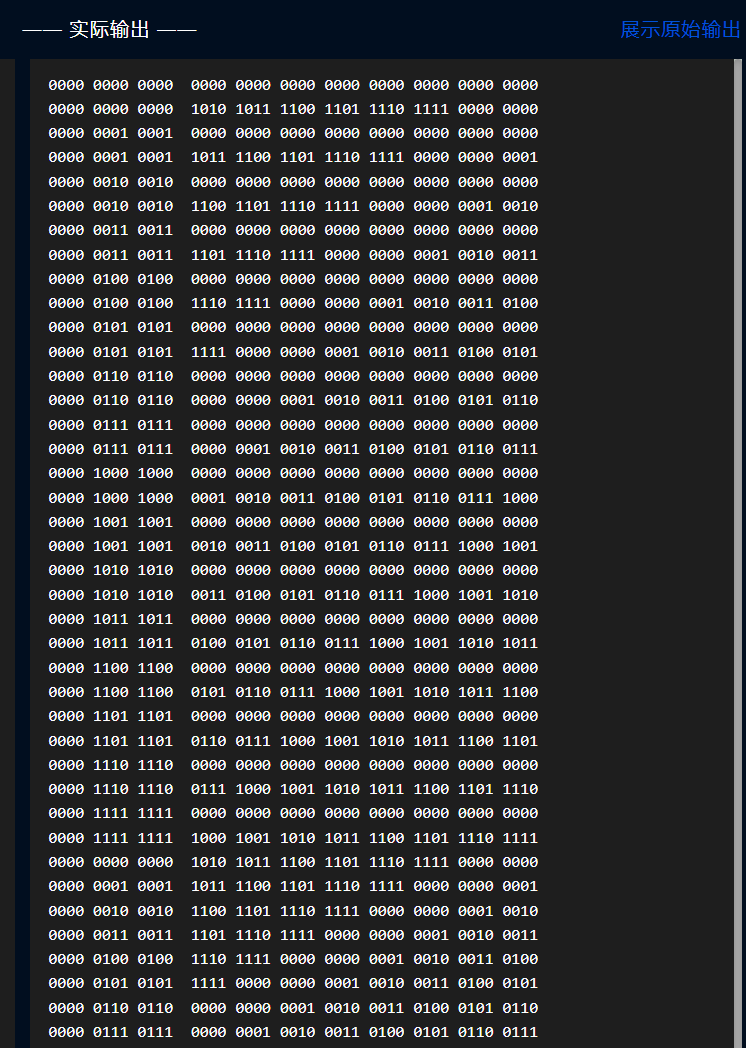
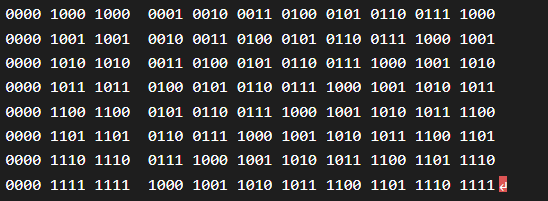


4. 把鼠标移到存储器组件上，点击鼠标右键，弹出菜单框，选中“编辑存储内容”，打开Logisim十六进制编辑器，可按照存储器设置的数据位宽，直接使用键盘输入数据。输入数据后，点击保存按钮，可把输入的数据保存到数据镜像文件中。



5.仿真验证电路，进入仿真状态，改变输入引脚赋值，记录输出引脚的数值，填写输入输出数据表，验证电路功能。

**（三）测试结果**

**（四）总结与思考**

通过实验，我掌握了使用Logisim构建存储器的方法，成功构建了一个存储器。相比于前几次实验。这次实验更加综合与复杂，牵涉到指令相关内容，并且可以通过键盘输入指令来验证电路功能。这样的实验，使我理解了随机访问存储器RAM的原理，对指令系统的理解也更深一步。

**实验2 指令读取和控制信号生成**

**（一）实验原理**

根据RISC-V的指令格式和取指令部件原理图设计RISC-V单周期处理器的取指令部件。其中，指令存储器使用Logisim内置库的ROM器件实现，要求指令长度位32位，指令存储器容量为1KB（即在按字编址的情况下，数据位宽为32位，地址位宽为10位，假设Logisim中的指令存储器表示为A[9:0]，当Logisim中的ROM设置数据位宽为32位时，每个地址中包含32位信息，相当于按字节编址的RISC-V存储器中的4个单元）。在Logisim中读取指令存储器时，原RISC-V设计原理图中的32位指令地址PC[31:0]，对应本次实验PC[11:2]=A[9:0]，即PC[31:0]其余的位（高20位和最低2位）无关。

**（二）实验步骤**

1.本次实验的任务需要首先在指令存储器子电路中放置ROM并在0号地址开始写入以下条指令:

（汇编指令描述方式对应教材表8.1，“x”代表寄存器编号，“0x”代表16进制数）

0x00a004b3 （汇编指令 add, x10, x0, x9）

0x0020af33 （汇编指令 slt, x30, x1, x2）

0x0020bf33 （汇编指令 sltu, x30, x1, x2）

0x0f00ef13 （汇编指令 ori, x30, x1, 0x0f0）

0x00412483 （汇编指令 lw, x9, x2, 0x004）

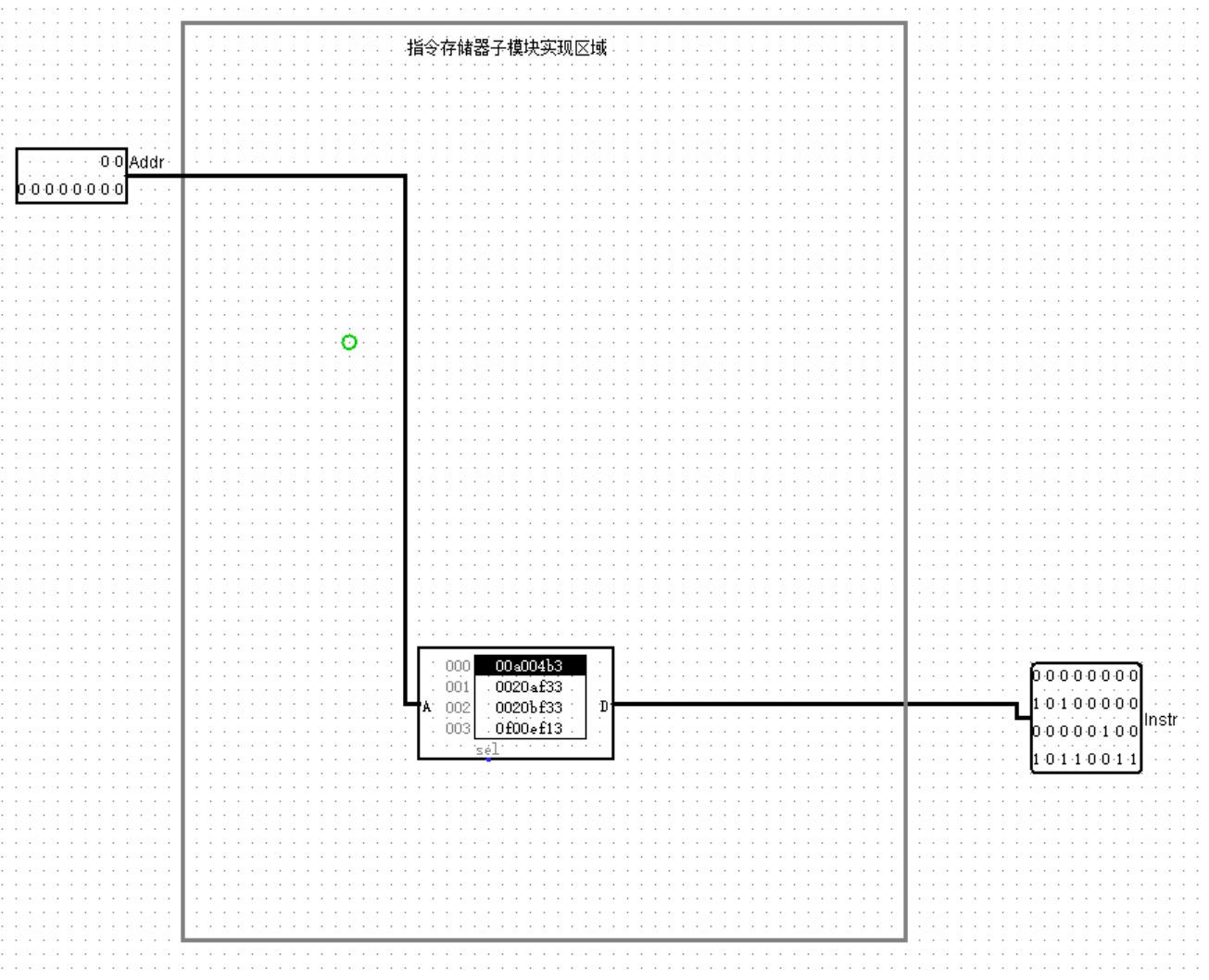
0x00008137 （汇编指令 lui, x2, 0x08000）

0x00912223 （汇编指令 sw, x2, x9, 0x004）

0x7ea500e3 （汇编指令 beq x10, x10, 0x7f0）

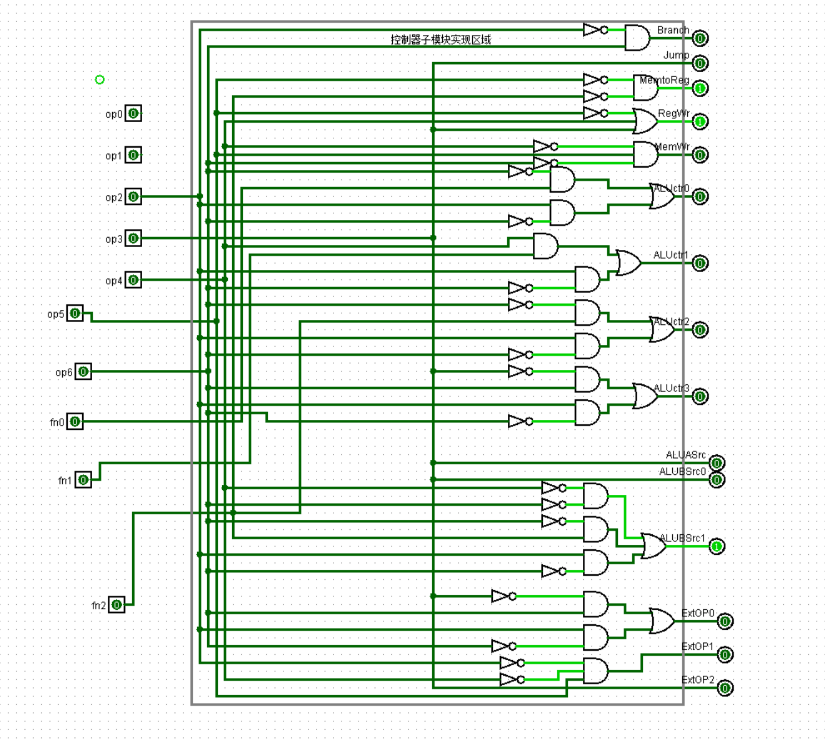
在指令存储器Rom的1023号地址写入下面这条指令：

0x818ff56f （汇编指令 jal, x10, 0xff80a）



2. 然后在指令解析测试子电路中利用Logisim内置库中的加法器实现指令的下地址逻辑，使得该子电路能够依次读入9条指令，并根据RISC-V指令格式将读出的指令解析为opcode、rd、funct3、rs1、rs2、funct7六个字段。

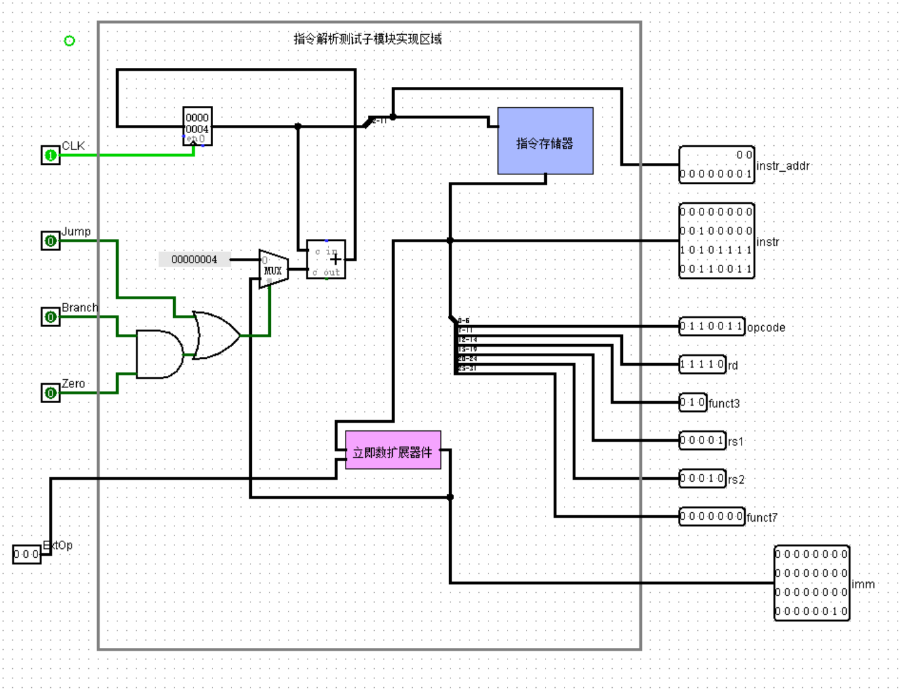
在这一步中，我选择了让系统自动生成，再手动合并。

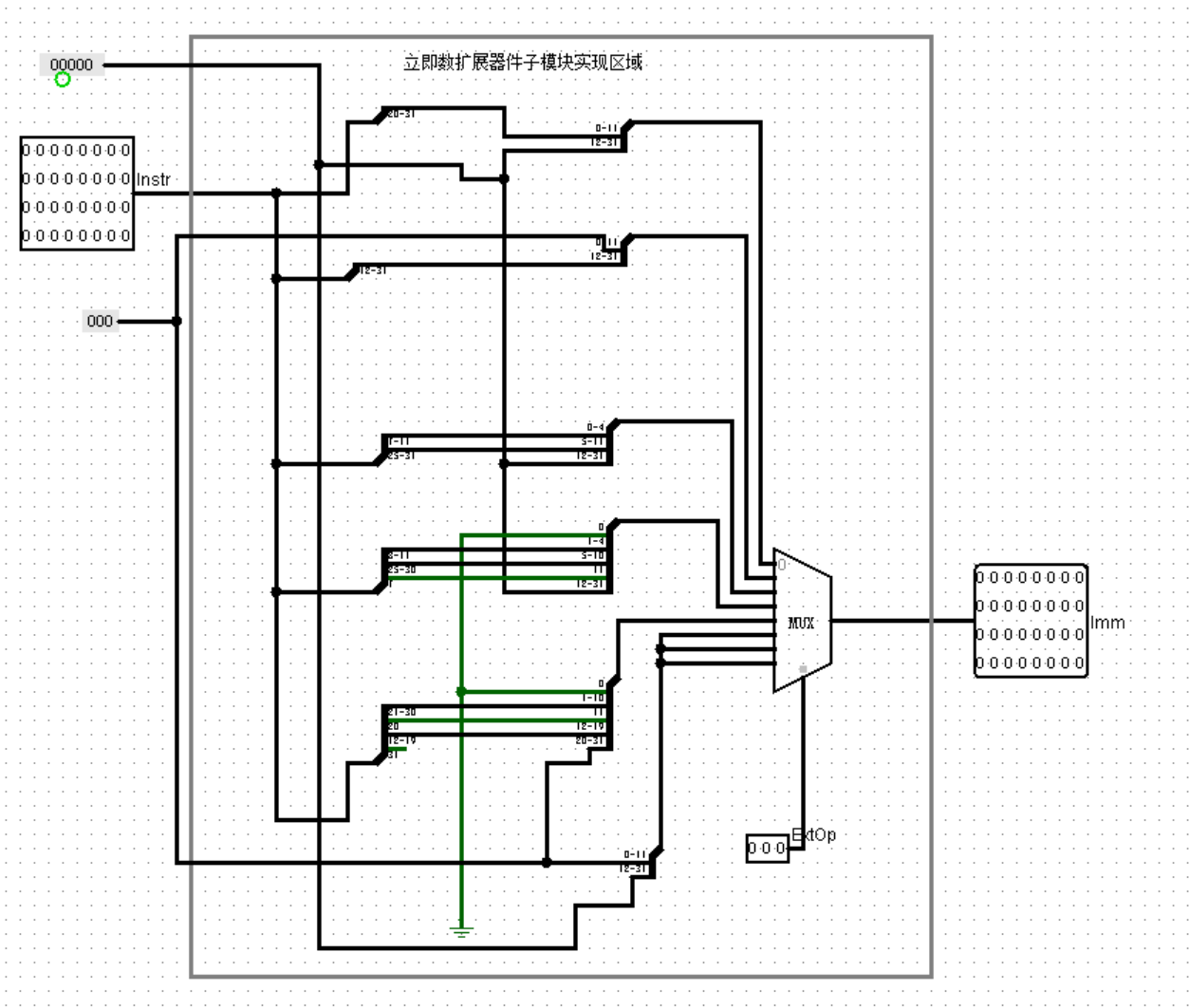


3. 接下来，根据下面所示的控制信号功能表在控制信号子电路模块实现指令对应的控制信号生成电路，然后根据立即数扩展部件原理图，在立即数扩展器件子电路中对指令中的立即数按照需要扩展为32位立即数。注：在作答区文件中，我们给出了一个控制器的参考示例，但是请同学们注意该示例中的输入输出信号的高位均在左侧。

4. 最后，在指令解析测试子电路中，将指令的下地址逻辑、指令存储器、立即数扩展器件连接起来，使其能够产生正确的控制信号和指令跳转。

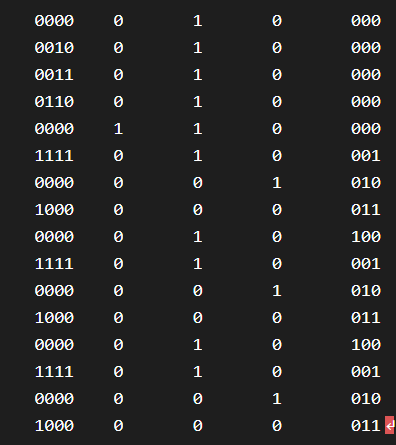
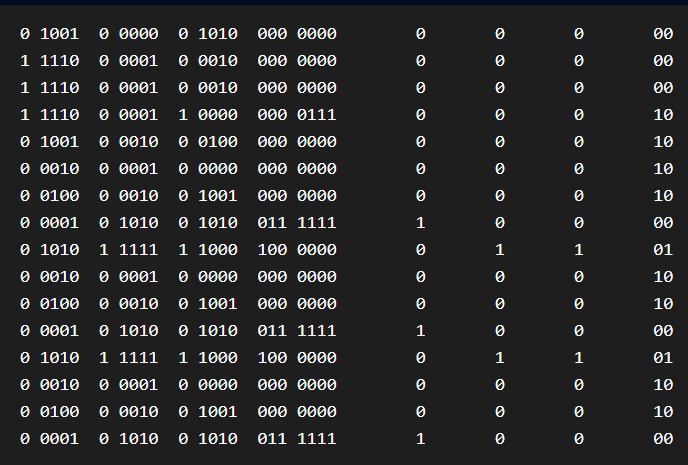
（3.4两步图片见下）





**（三）测试结果**

**输出分别是这么多指令的数字，经检验，结果符合。**



**（四）总结与思考**

这个实验，我一共做了三十几天，累计花费在期中的时间超过10个小时。

这是一道非常复杂的实验题，在做题的过程中我犯了这几个错误：

1.程序计数器用的是counter而不是争取的register；

2.利用Logisim自动生成电路的时候，忘了检验电路的合理性以及连线是否真正连接上去。

3.有一些位数不对应，忘了改。

错误不止这些。

不过，好在最终我完成了。我认为，该实验不仅增强了我对指令的理解，还锻炼了我的细心能力，我觉得这题出的太好了。

数字系统实验到此就做完了，我希望自己能更进一步！