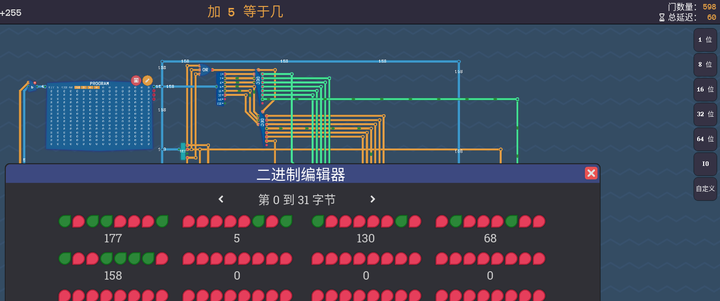
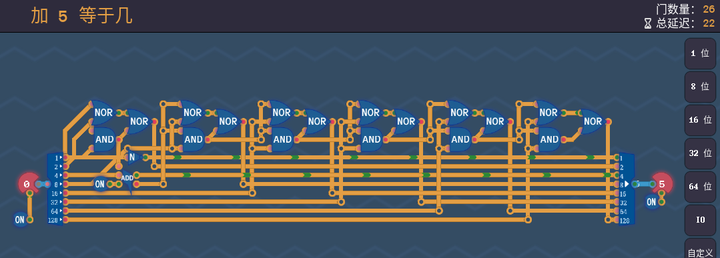
**编程**

**1、加五等于几**

开始熟悉机器码编程

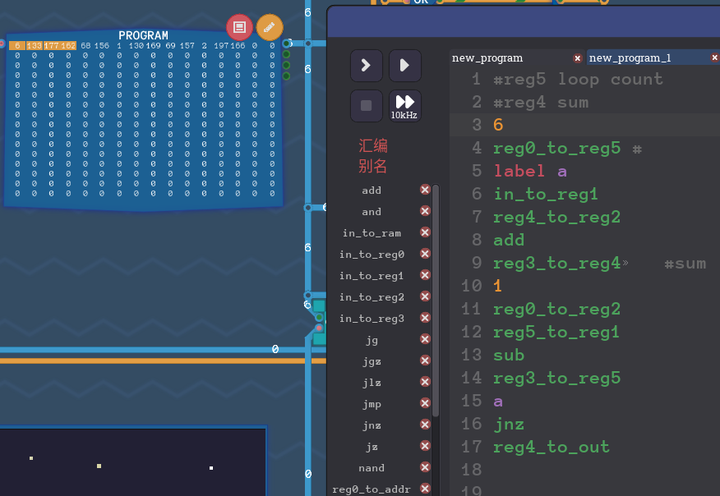


不过也可以继续挑战用硬件实现，多练习总没坏处。

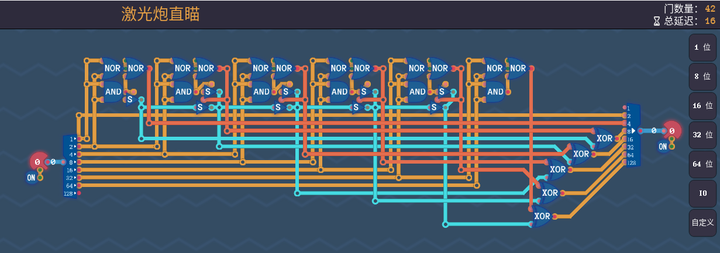


**2、激光炮直瞄**

输入乘6，正常编程将输入相加6次即可

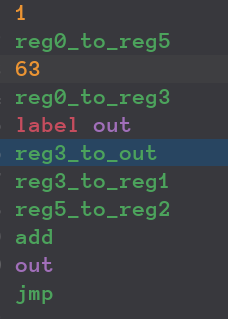


也可以用硬件实现，原理是把乘6看成2\*2+2，相当于把输入左移两位和左移一位再相加，就得到下面的电路，这是我合并优化后的结果，如果直接看不好理解，可以把输入先分开再合并。

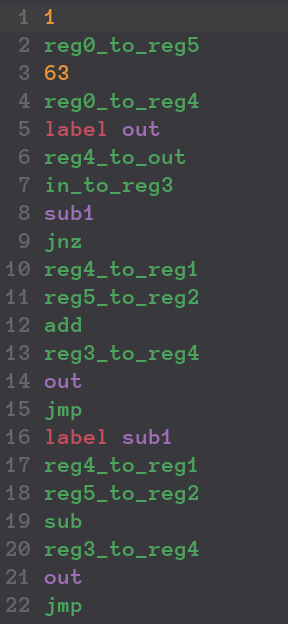


**3、密码锁**

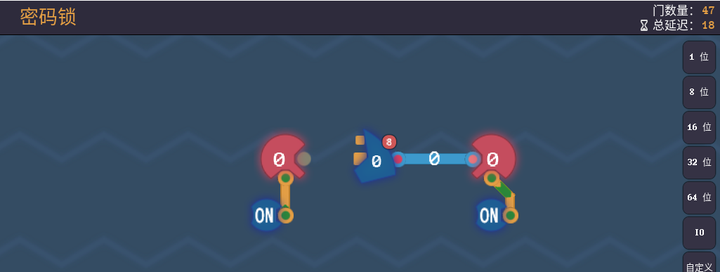
这个理论上用二分法能很快找到答案，不过目前的计算机实现二分不太容易，只好用最笨的办法一直加1，此计算机立即数最大是63，就从63开始吧。



考虑到输入和通用性，可以把减一加入，不过针对此关，密码都在200多，减一根本用不上。



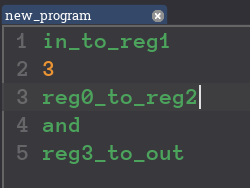
这个用硬件实现就更简单了，用计数器就OK



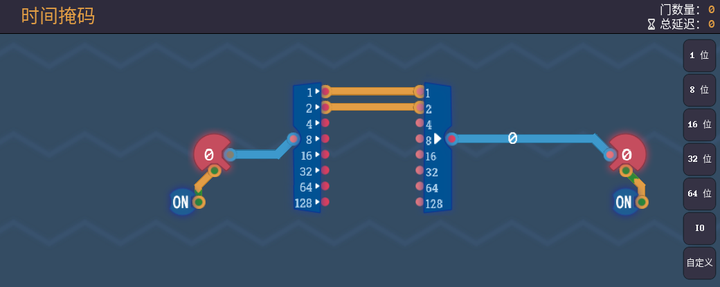
不过以上方法效率很低，应该有更好的办法。

**4、时间掩码**

这个比较简单，提示已经说的非常明白。



硬件就更简单了，有点作弊的感觉。



**5、太空入侵者**

这个方法很多，我采用以逸待劳的方式

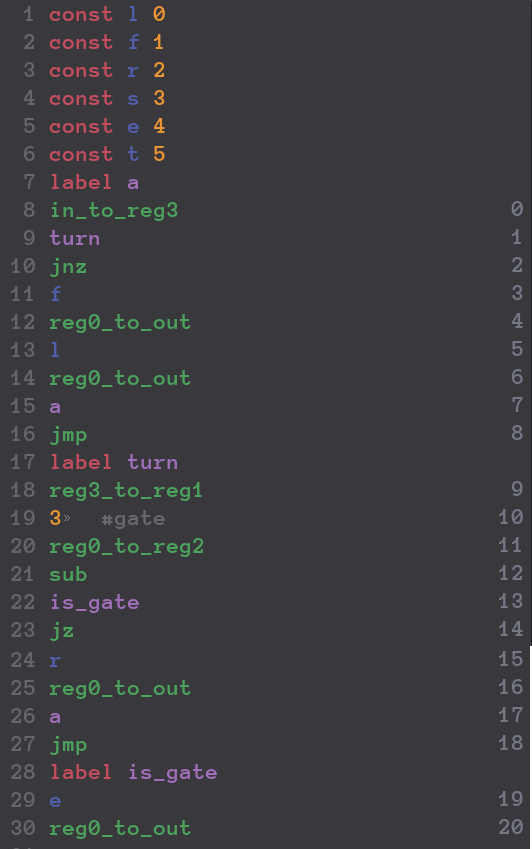


硬件实现，这里输入只有四种：空（0），箱子（40），太空老鼠（33），碎片（239），所以很容易通过位的组合判断，两个与门识别箱子和太空老鼠，可以直接发射激光，最高位为1是碎片，可以直接前进，如果不是这三种（空）就呆在原地等待。

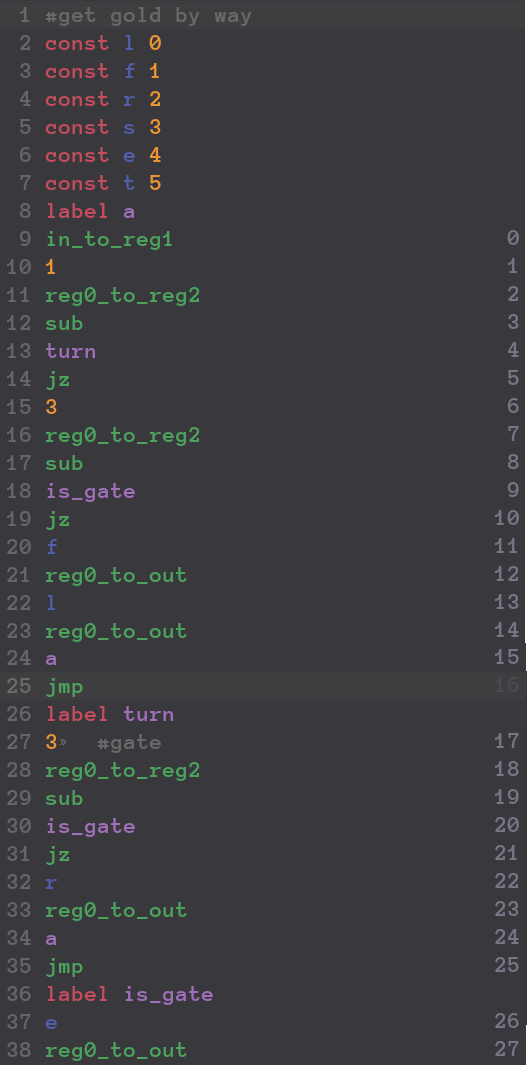


**6、迷宫**

提示给了解法，一直靠一边走就能出去，下图是靠左走的版本，把r换成l，l换成r就改成靠右版本，实测靠右快一些。



还有个吃路过金币的版本，吃金币可以累积，金币通过这关后的小游戏会用到，大家可以自行挑战吃全部金币。



硬件实现，思路跟软件一样，下图是靠右走的版本



迷宫过关后，手动走到迷宫的门口可以进入门内，会进入新的地图（外星人的飞船），在这里有4个任务，完成后可以达成“机器志愿者”的成就，还有两个隐藏成就也在这里完成。



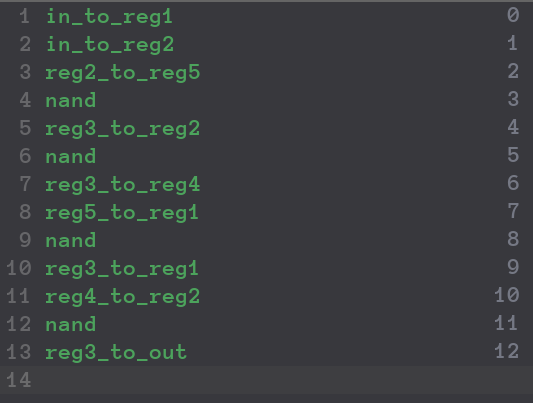
怎么达成玩家自己探索吧。实在不行看后边的链接，有所有成就的解法。

<https://steamcommunity.com/sharedfiles/filedetails/?id=2798423875>

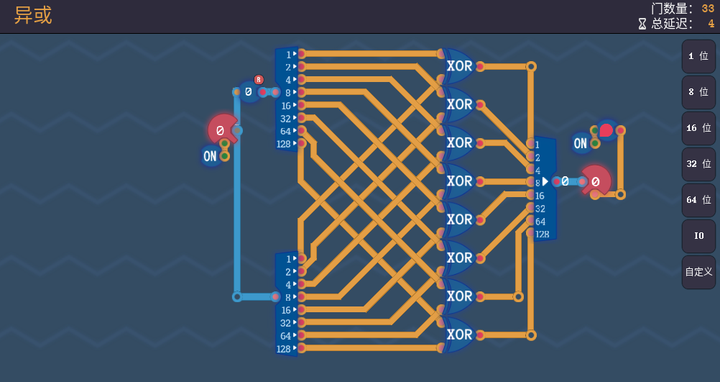
**处理器架构2**

**1、异或**

之前硬件实现过，换成代码即可



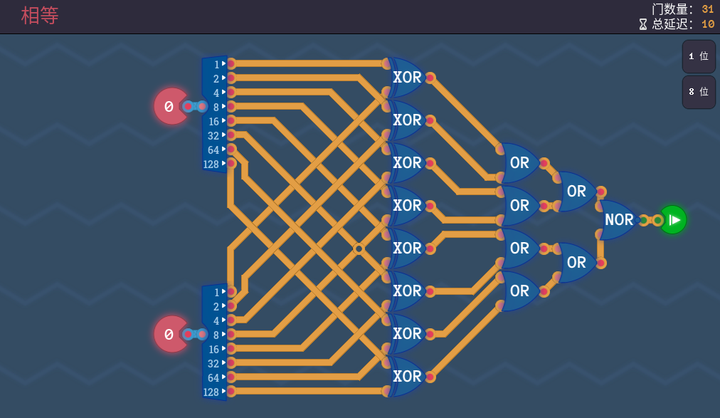
硬件挑战



后面8位异或、8位常数都比较容易，这里就不罗嗦了。

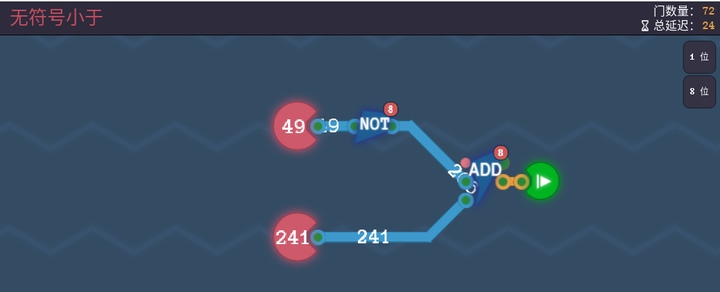
**2、相等**

两个数相等对应各个位异或后必为0，然后或非就OK了。

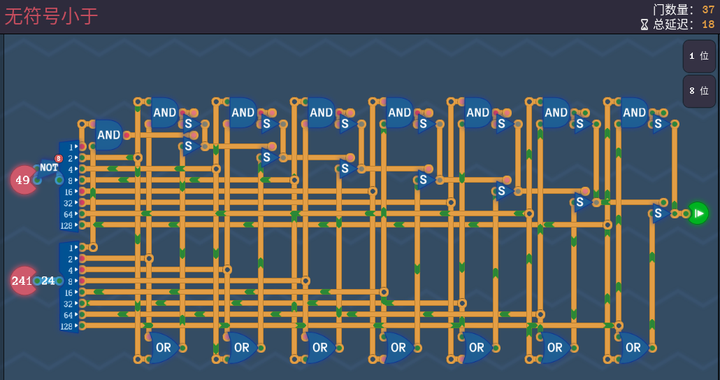


**3、无符号小于**

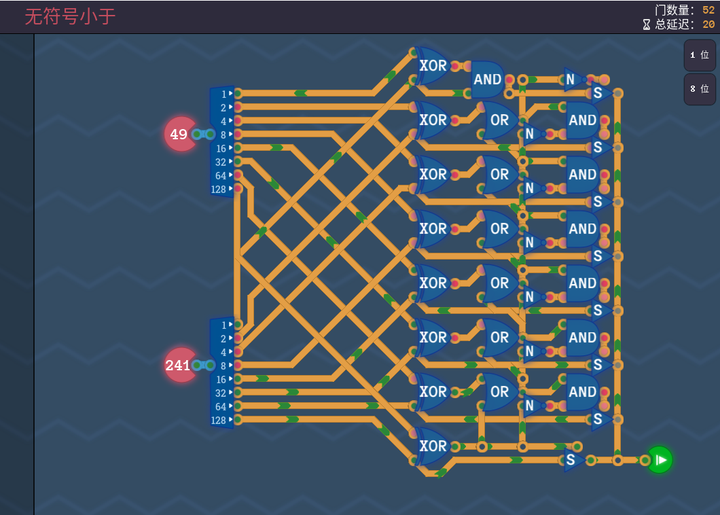
一个数A取反~A和自己相加必为255，A+(~A)=255,只有比A大的加~A会溢出，所以加法溢出第一个数必然小于第二个。



同样的原理用分立元件实现，由于我们只关心进位而不关心结果，所以可以在加法器的基础上优化，如下



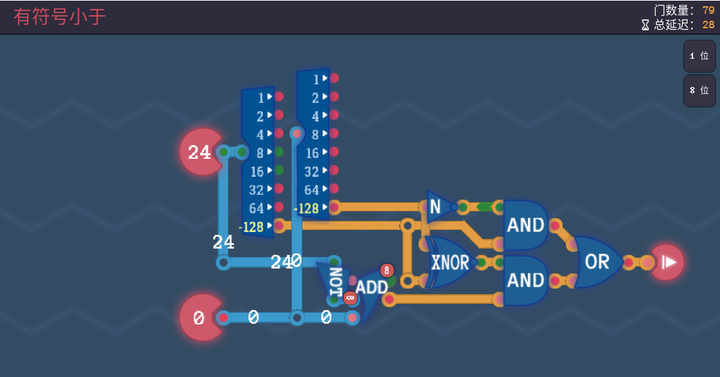
还可以通过从高到低逐位比较判断，



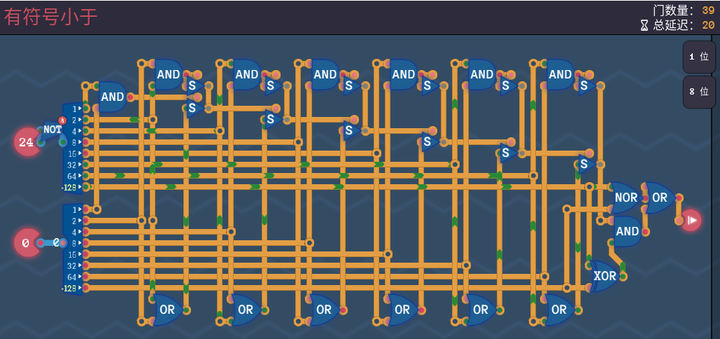
这个解法多样，大家可以继续挑战。

**4、有符号小于**

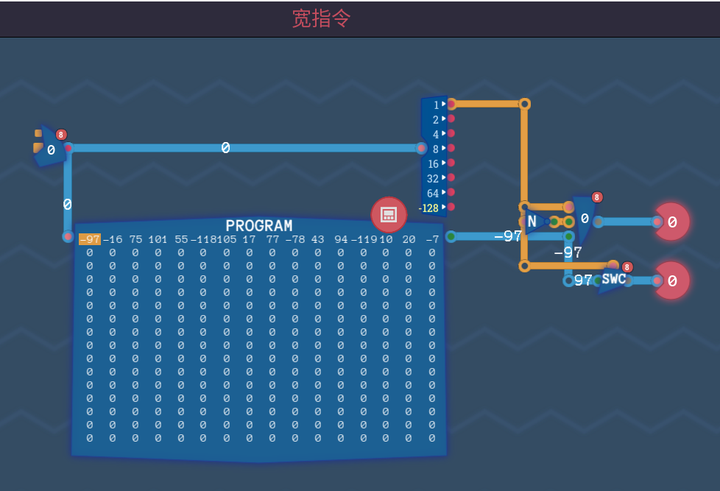
这个在无符号小于的基础上结合符号判断即可，先根据符号判断大小，如果同号再用无符号的办法判断。



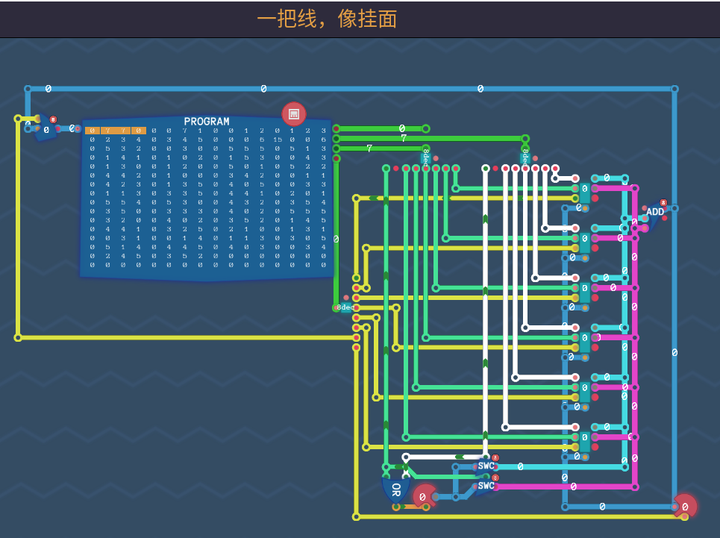
用分立元件实现如下，原理同无符号小于。



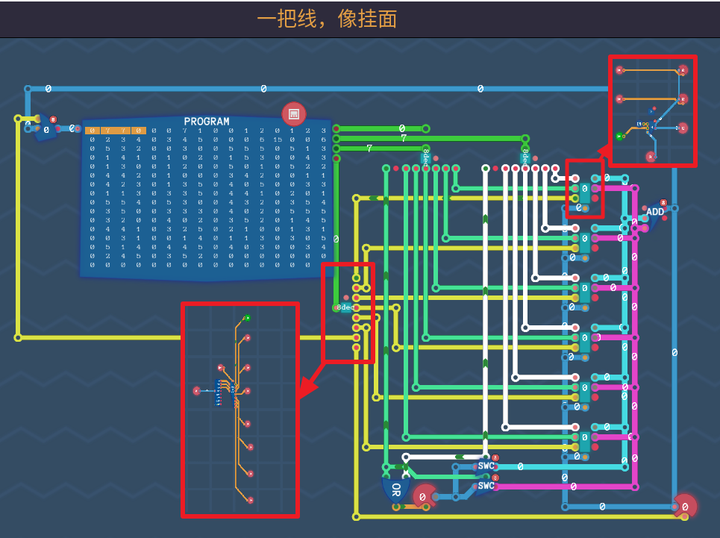
**5、宽指令**



**6、一把线，像挂面**

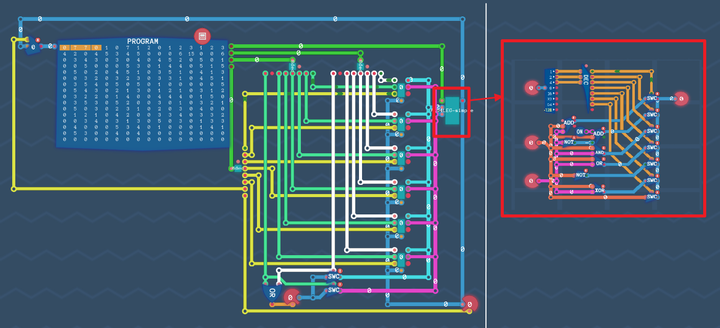


这里为了看着简洁，把分线器和译码器通过元件工坊集成了，把寄存器改成了双读取的，这样参数2和参数3可以分别控制一个读取引脚



**7、操作码**

扩展算数运算操作，把上一关中的ADD换成新的计算单元。



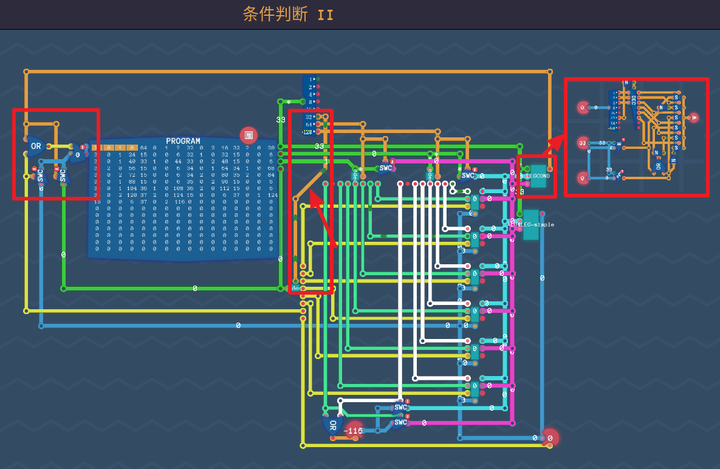
**8、立即数**

将操作码的高两位引出，分别禁用参数1和参数2对应的解码器，然后直接将两条线通过开关接到对应的总线上。



**9、条件判断II**

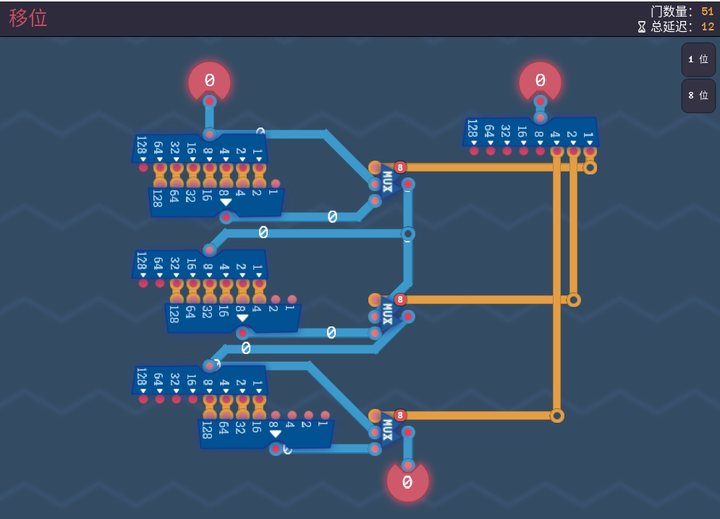
根据给定的操作码实现对应的条件判断，条件满足就启用计数器的擦写模式，此时参数4是跳转地址，也可以看作是立即数，所以要禁用参数4的解码器，并将参数4通过开关连接到计数器



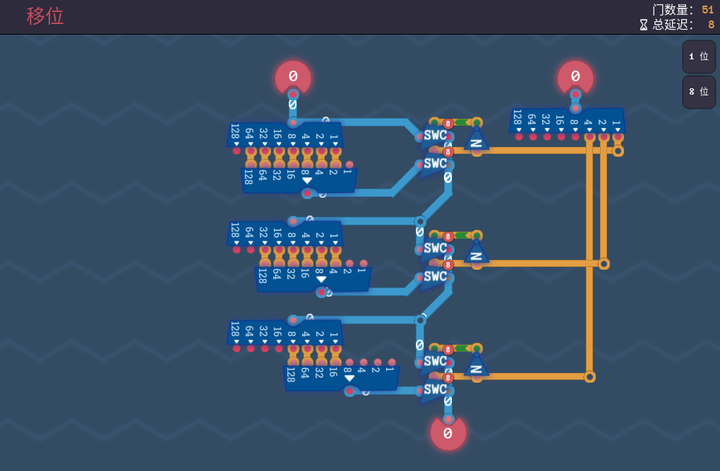
**函数**

**1、移位**

移位是第一个参数乘 2n ，n就是第二个参数，假设它的二进制表示为 b2b1b0 ,b代表一个二进制位，2n可以表示为  ,如下的电路就是按照这个搭出来的。这个容易跟乘法混淆，乘法是第一个参数乘n，注意区分。



使用分立元件会快一点

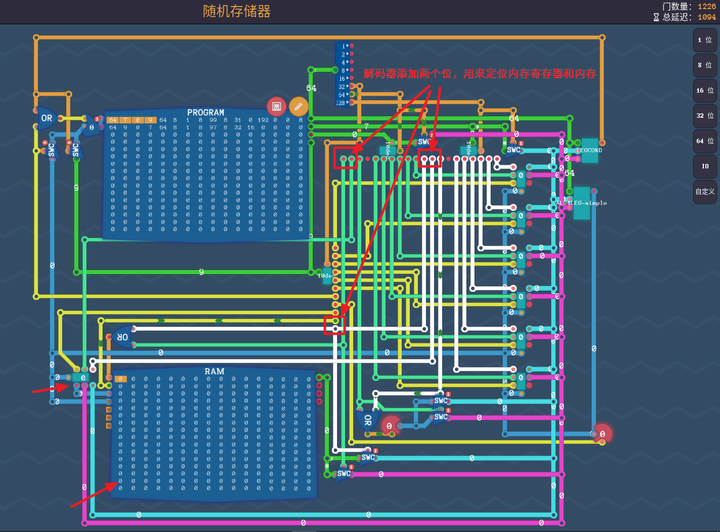


**2、随机存储器（RAM、内存）**

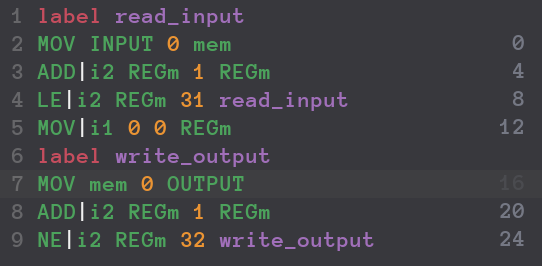
这关的实现方式灵活多样，可以把它跟寄存器统一编址，按照操作寄存器的方式操作内存，也可以把内存看作类似加法器的独立单元，设计新的指令码操作，下面分别介绍。

**统一编址**

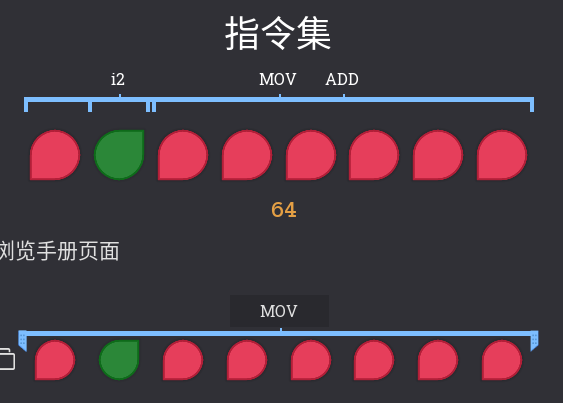
按照关卡的要求，需要添加一个内存和寄存器，先给寄存器写内存地址，然后再操作内存，读写内存分别需要两步操作。



代码如下：



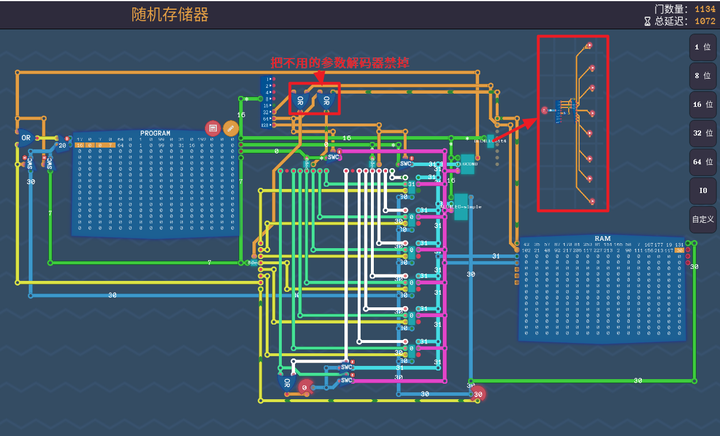
其中，MOV指令就是ADD指令让第二个参数始终为0即可，REGm表示内存地址寄存器，LE表示小于等于，NE表示不等于，i1、i2分别表示第一个参数和第二个参数为立即数。当然也可以添加各个指令的立即数操作指令，这样就能直接用而不用或了。



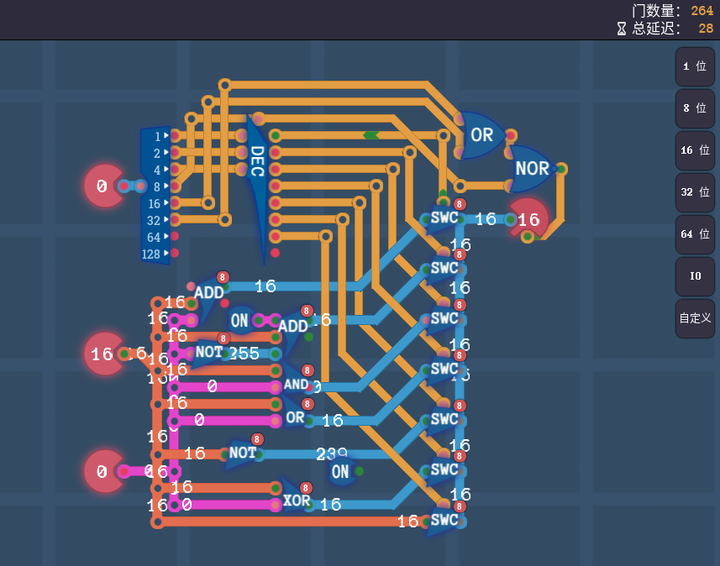
**专门的内存操作指令**

把内存当成独立的单元，通过操作码判断是否读写内存，这种方式不需要额外添加寄存器。

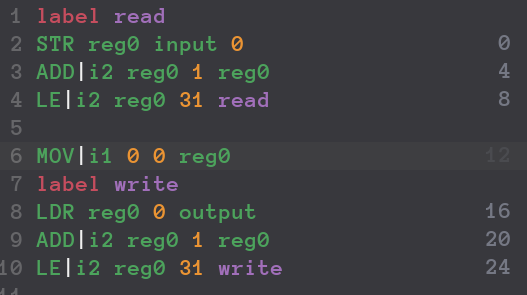
优势是读写和地址指定只需一条指令，可以使用立即数指定地址。



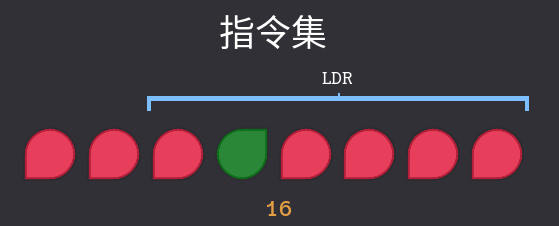
注意，这里需要对加法器进行一点修改，之前的加法器输出是普通输出，会跟RAM的输出冲突发生短路，所以需要将加法器的输出改为8位三态输出，并且只在操作码的3、4、5位为0的情况下输出，避免与其他指令冲突。

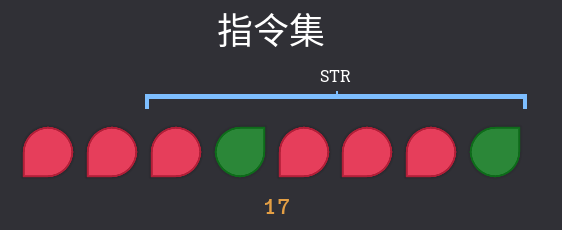


代码如下

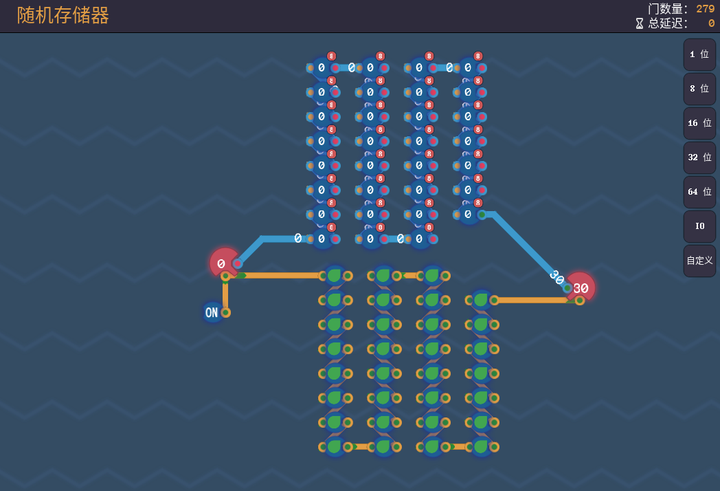


其中，LDR和STR如下，其他跟之前一样。



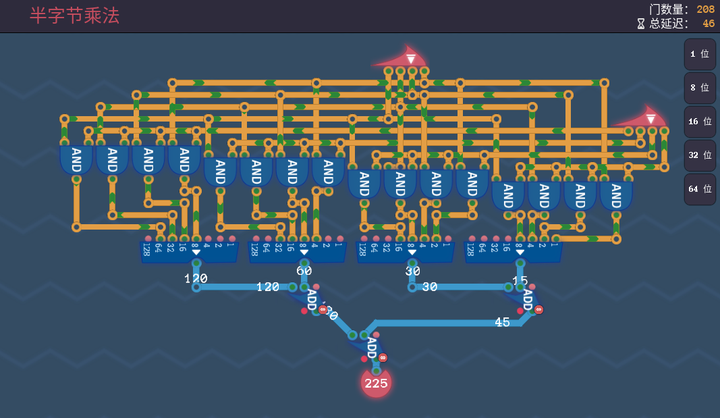


最后尝试硬件解法，如下投机取巧的解法，目的达到了，但对于实现计算机没啥意义。

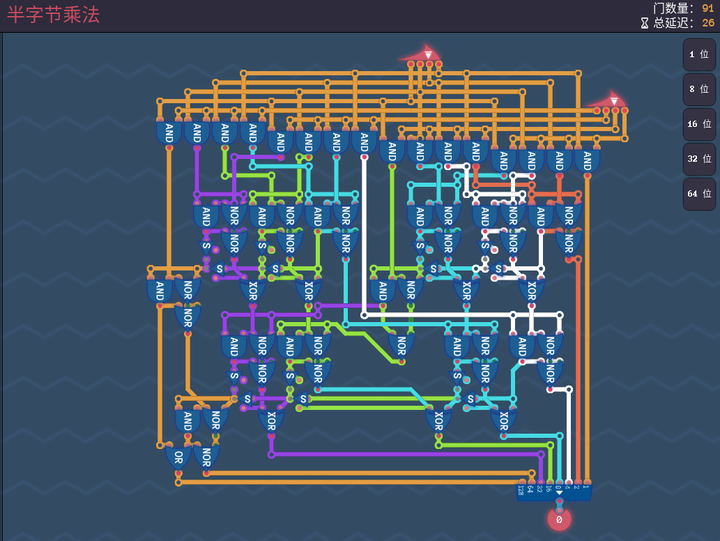


**3、半字节乘法**

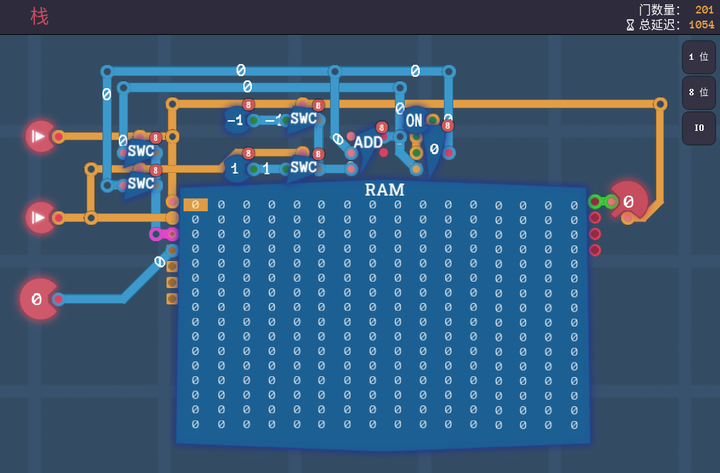
乘法的竖式计算方法



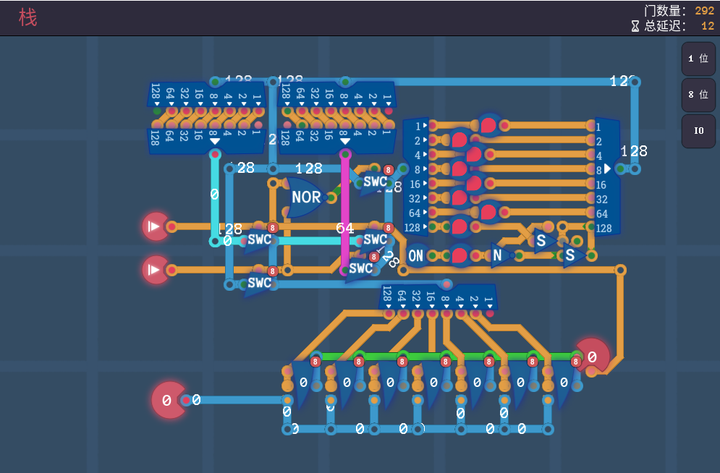
使用分立元件实现，可以省掉不少门和延迟，这个还能优化，我头疼就不研究了，大家挑战吧。



**4、栈**



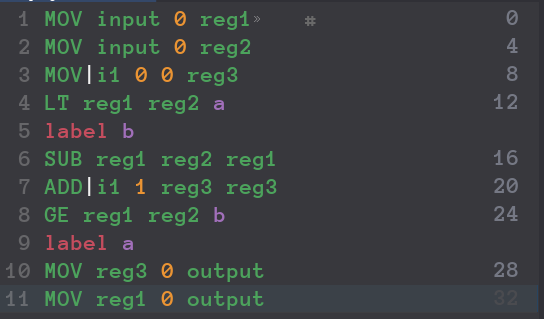
这个内存的延迟很高，可以采用投机取巧的办法（最多入栈了7个数）降低延迟，门数量增加了，不过总分是下降了不少的（这个游戏分越低越好）。



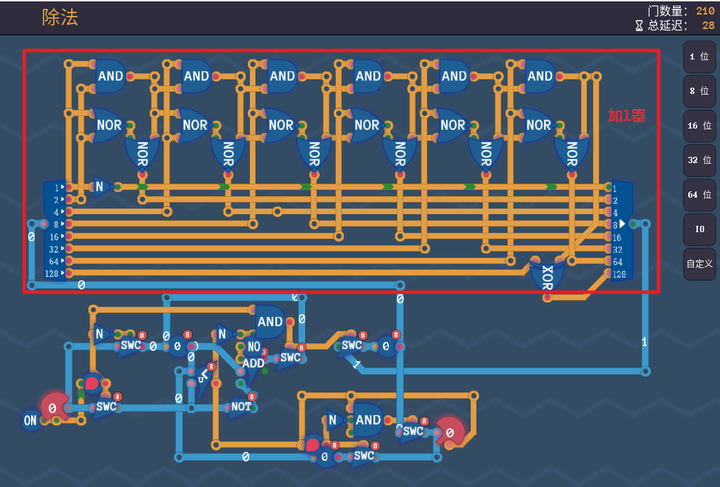
排行榜还有更优的办法，我实在想不出了，有做到的小伙伴欢迎交流分享。

**5、除法**

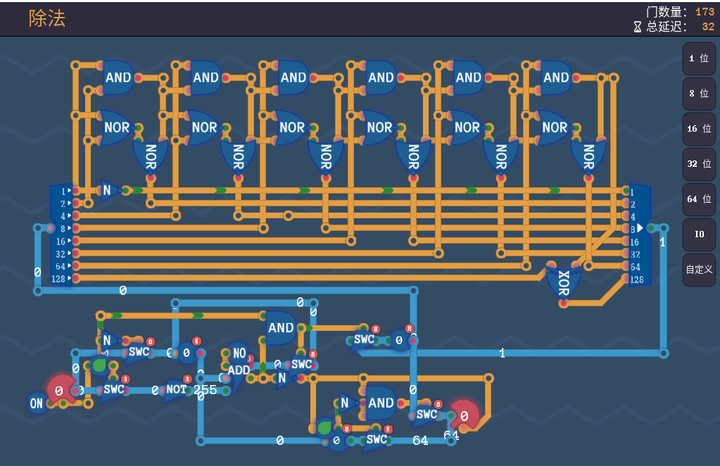
常归做法就是比较除数和被除数的大小，如果被除数大就减去除数，计数，直到小于为止，最后计数就是商，减的剩下的就是余数。



硬件实现，原理同上，

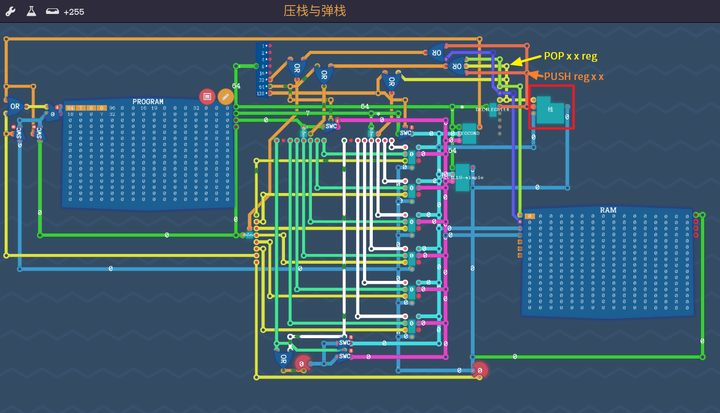


如果被除数大于除数，那么加法器会有进位，否则没有进位，通过进位也可以判断是否小于，如下，这种方式延时多了不过省掉了小于比较器。

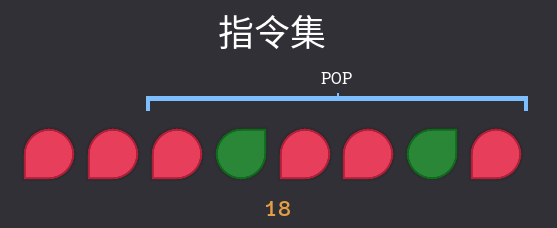


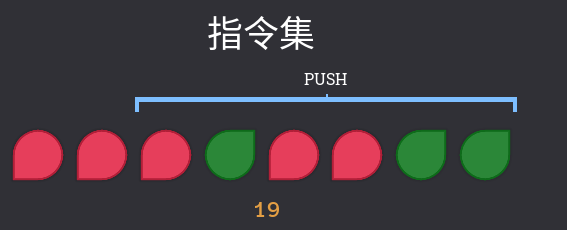
**6、压栈与弹栈**

添加新指令POP和PUSH，把之前的栈添加到电路中，根据参数使用情况禁用不需要的参数解码器。



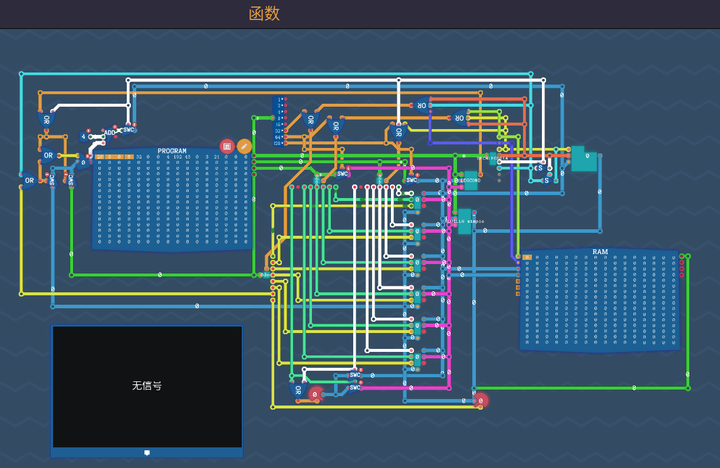
POP和PUSH指令如下：





**7、函数**

这关没啥新东西，不过细节比较多，CALL的时候需要开启入栈，RET的时候需要开启出栈，需要改写计数器，需要禁用所有的参数解码器，如果不仔细很容易出错。



到此，我们的计算机功能已经比较完善了，可以实现比较复杂的逻辑和运算了。