Lab02

梁峻滔 PB19051175

1. 算法思想

给定两个正整数，常用的求最大公约数的方法有辗转相除法和更相减损术，但是由于除法在 LC-3 中相对不容易实现，而减法则可以直接使用 NOT 和 ADD 指令来实现，所以采用更相减损术。算法如下：

任意给定两个正整数，以较大的数减较小的数，接着把所得的差与较小的数比较，并以两者中的大数减小数。继续这个操作，直到所得的减数和差相等为止。此时的减数或者差就是所求得的最大公约数。

例. 用更相减损术求 98 与 63 的最大公约数。

解：把 98 和 63 以大数减小数，并辗转相减：

98-63=35

63-35=28

35-28=7

28-7=21

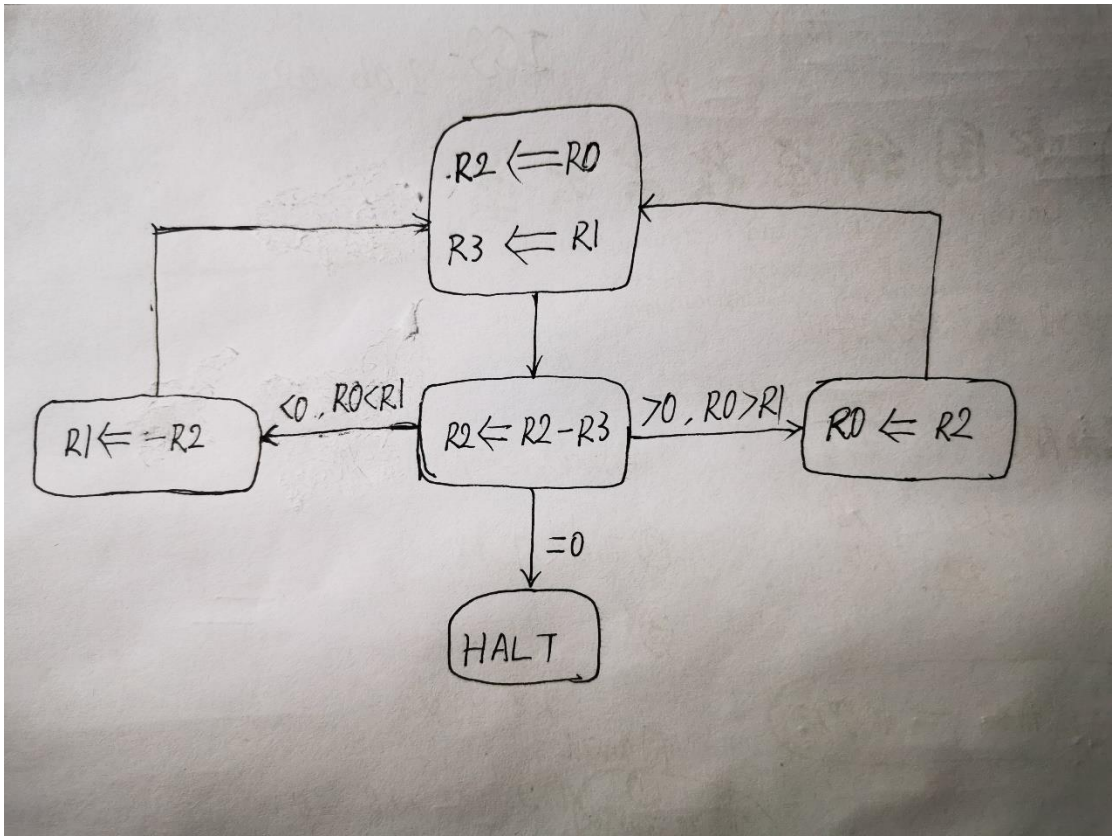
21-7=14

14-7=7

所以，98 和 63 的最大公约数等于 7。

在 LC-3 中，可以使用另外两个寄存器 R2 和 R3 来进行相减和比较，并根据相减的结果(可以同时得到差和两个操作数的相

对大小)来更新 R0 和 R1，按算法重复操作，直到相减结果为 0，此时 R0 和 R1 中都将存放着最大公约数。流程图如下：

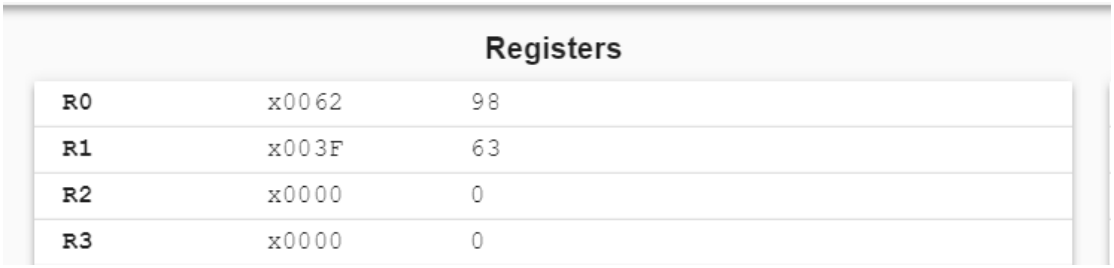


2. 代码实现

```
;使用更相减损术求给定两个正整数的最大公约数
;给定的两个正整数储存在R0和R1
;最后求出的最大公约数存放在R0
;手动设置R0和R1的初始值
;
        .ORIG   x3000
UPDATE  ADD     R2,R0,#0      ;将R0、R1放在R2、R3中作相减和比较
        ADD     R3,R1,#0
;
;相减
;
        NOT     R3,R3
        ADD     R3,R3,#1
        ADD     R2,R2,R3      ;R2中存放相减的结果
;
;比较和选择跳转
;
        BRp     POS
        BRz     EQUAL
        NOT     R1,R2         ;R2-R3结果为负，则R1比R0大，将差值取为正后存进R1
        ADD     R1,R1,#1
        BRnzp   UPDATE
POS     ADD     R0,R2,#0      ;R2-R3结果为正，则R0比R1大，将差值存进R0
        BRnzp   UPDATE
EQUAL   HALT                  ;相等时已经求得最大公约数，结束
        .END
```

3. 测试

首先设置 R0、R1 的初始值：



Registers

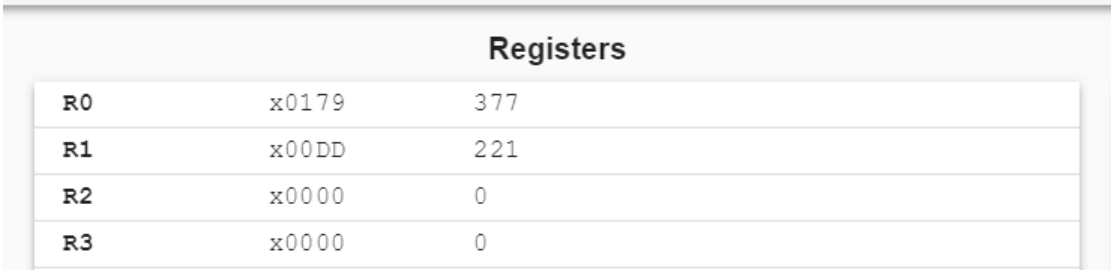
R0	x0062	98
R1	x003F	63
R2	x0000	0
R3	x0000	0

运行后得到了同所举例子相同的结果：

Registers

R0	x0007	7
R1	x0007	7
R2	x0000	0
R3	xFFF9	-7

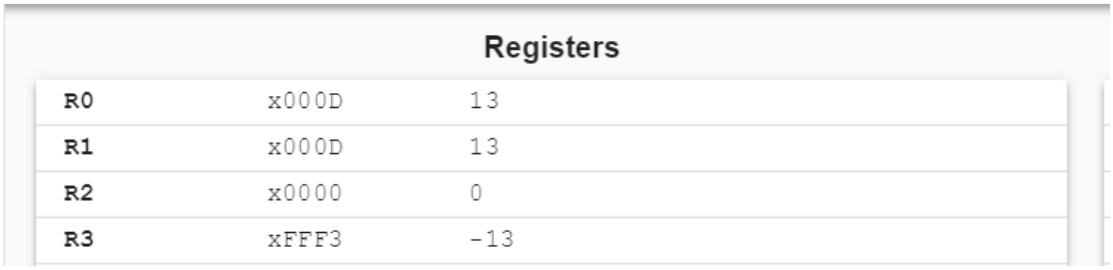
设置$\mathrm{R0} = 377 = 13 \times 29$，$\mathrm{R1} = 221 = 13 \times 17$，则预期运行后的结果为$\mathrm{R0} = \mathrm{R1} = 13$。



Registers

R0	x0179	377
R1	x00DD	221
R2	x0000	0
R3	x0000	0

运行后



Registers

R0	x000D	13
R1	x000D	13
R2	x0000	0
R3	xFFF3	-13

得到了预期的结果，程序正确地完成了任务。