**Report\_lab1 实现LC-3机器乘法**

PB20151793 宋玮

***part 1: L版本程序***

1.设计思路

**L版本程序要求尽量编写更少的代码行数。**

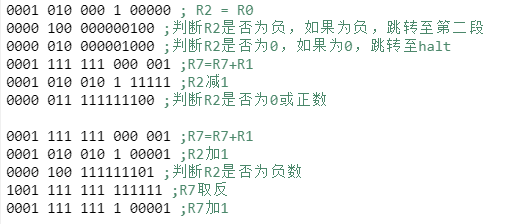
**最终程序行数：11行（如下图）**

***大致思路*：**利用其中一个乘数（R0）作为“计数器”（count），对另外一个乘数(R1)进行加操作。

首先判断R0正负：

如果是正数（或0），则每次对R7进行加R1操作后，R0减1，重复操作，直至R0减为0。由于该过程对R1加了R0次。因此R7中的结果为R0×R1。

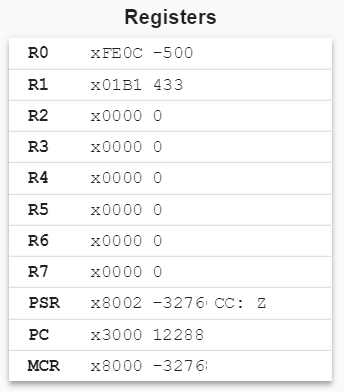
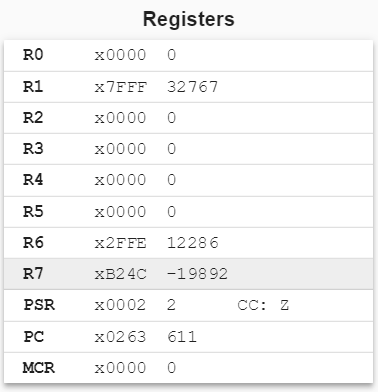
如果是负数，则每次对R7进行加R1操作后，R0加1，重复操作，直至R0变为0。最后对R7中的结果取反加一，可得到R0×R1。



***测试数据：***（此处仅列举两组数）

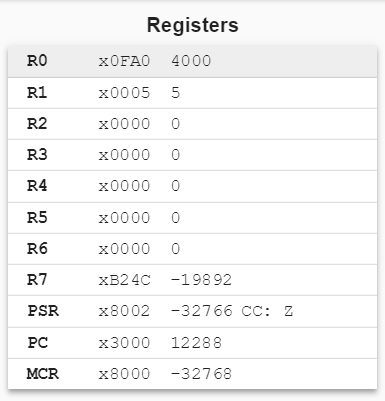
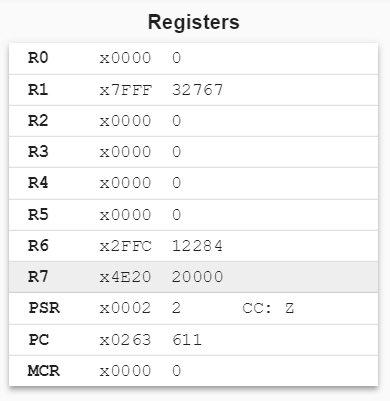
（1）-500 \* 433 （刻意溢出）

初始状态 运行后结果

------------------🡪

（2）4000 \* 5

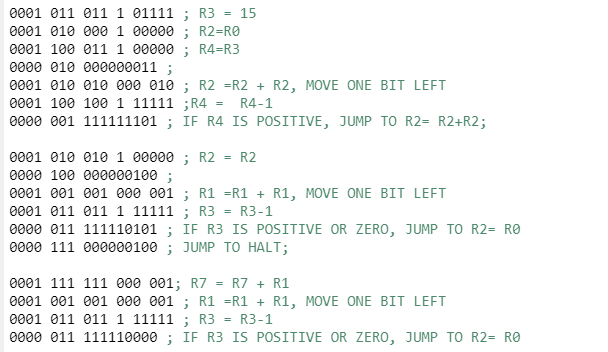
初始状态 运行后结果

------------------🡪

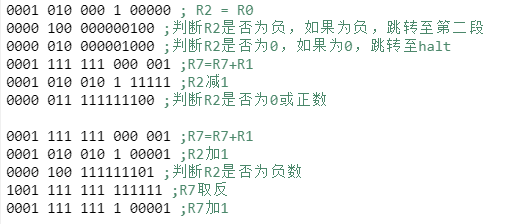
2.改良过程

起初我想采用移位方式进行乘法的计算。即对其中一个乘数进行左移操作（左移15位，14位，13位……）,另一个乘数每次左移一位，判断正负，以决定是否进行加操作，该过程即列竖式乘法计算，**详见p版本程序**。

经过一系列改动，该思路写出的程序最短17行（如下图）。不符合要求。



因此，我改变思路，即最终方案为上述**1.设计思路**中的方案，**程序最短为11行。**



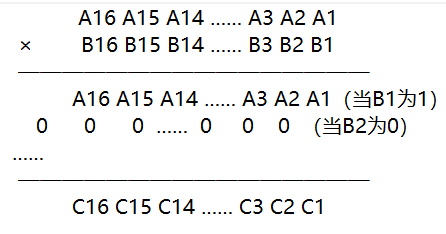
***part 2: P版本程序***

1.设计思路

**P版本尽量让程序执⾏更少的指令。**

**最终程序平均执行指令条数约为：132行**

***大致思路：***通过移位和加法模拟列竖式计算。如下图：



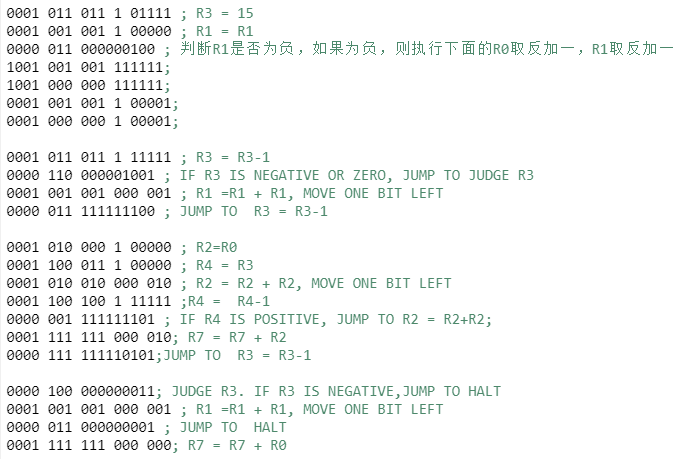
由于R0,R1均为01串。因此在进行列竖式乘法计算时，每一行结果要么是R0，要么是0，取决于R1对应位上是1还是0。再通过对R0的移位后相加，则可以得到结果。

目前，仅能实现的移位方式是左移，即通过R0=R0+R0，可以实现对R1的左移一位。因此，本算法也采取自加左移方式。

而判断R1对应位上是0还是1，也需要对R1进行左移，然后通过判断正负，判断该位上是0还是1。

注：当R1为负数时，采取R1和R0同时取反加一的操作，即同时取相反数，可以缩减指令执行条数。

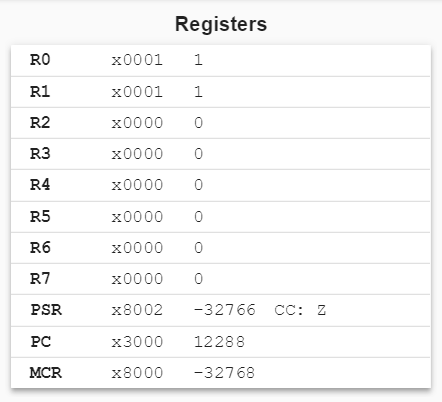
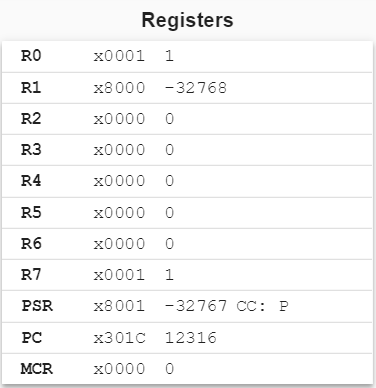
代码如下：



***测试数据：***

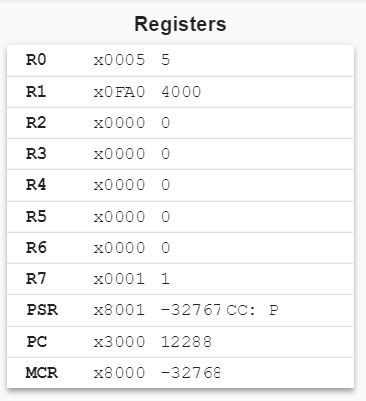
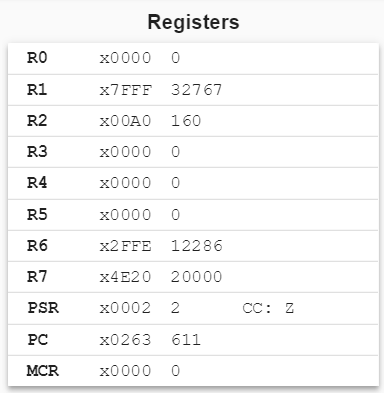
**指令条数是通过模拟器的step in至结束的执行次数得出的。**

（1）1\*1

------------------🡪 

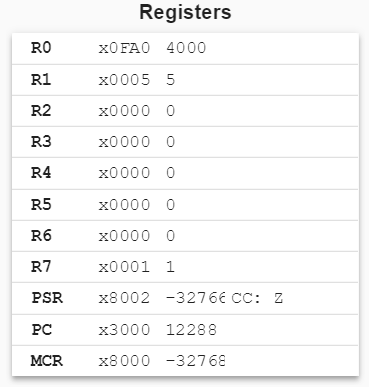
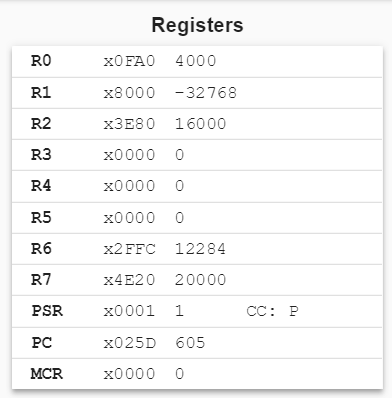
执行指令条数约为 66。

（2）5 \* 4000

------------------🡪

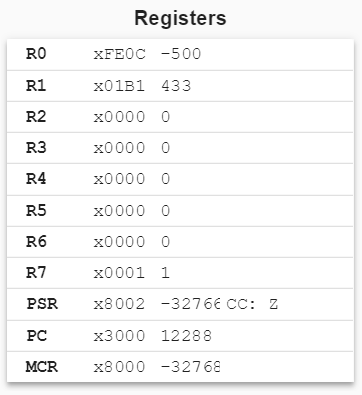
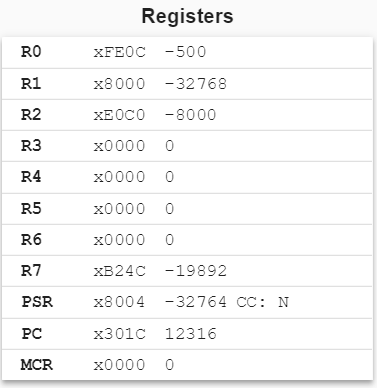
执行指令条数约为237。

（3）计算 4000 \* 5

-----------------🡪

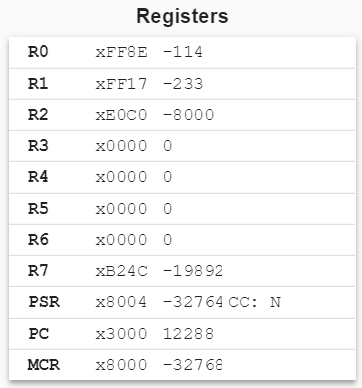
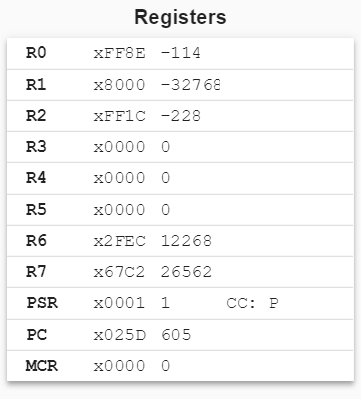
执行指令条数约为74。

（4）-500 \* 433 （刻意溢出）

-----------------🡪 

执行指令条数约为152。

（5）-114 \* -233

-----------------🡪

执行指令条数约为144。

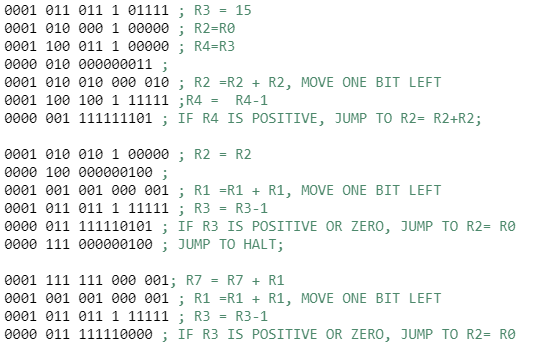
除去赋初值和trap指令，平均执行指令约为：134.6-3≈132

2.改良过程

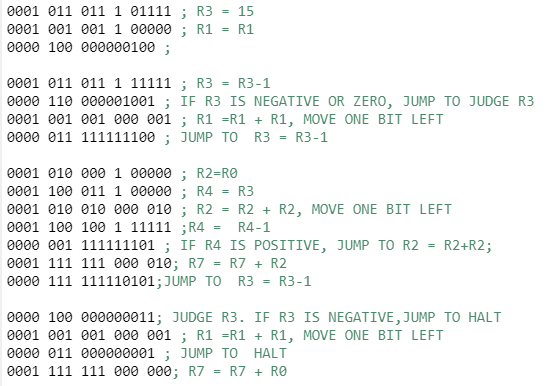
起初，我也是采用列竖式乘法计算，但是由于第一次尝试是R0一直在移动，因此每次执行条数都在**500**次左右。代码如图（1）。

然后，我将方案改成对R1第一位进行判断，只有为1时才对R0移位。平均执行条数在**187**条左右。代码如图二。

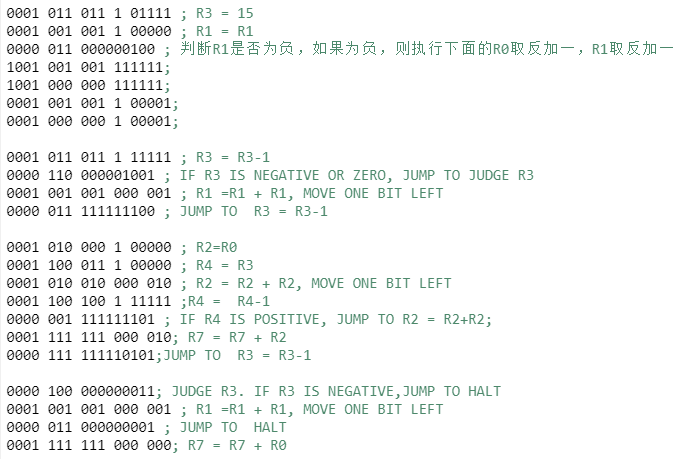
由于方案二执行条数仍然过大，并且我发现在计算-114\*-233时尤其大，达到了404次。于是我在方案二的基础上继续改进，改成先判断R1是否为负数，如果R1为负数，则对R1和R0都取相反数，即R1将变成正数，这样将减少R1中1的个数，大大减少执行条数，于是有了**1.设计思路**中所述的最终方案，去头去尾，平均执行指令条数大概是**132**条。



图（1）



图（2）



图（3）最终方案