Lab1

实验目的

1. 使用LC-3机器代码编写一个程序，能够计算在给定的数A的最低的B位中有多少1
2. 熟悉LC-3 Tool的使用方法
3. 熟悉LC-3汇编代码以及汇编代码翻译至机器代码的方法
4. 仅使用LC-3代码实现减法，模等操作

实验原理

- 如果我们想要得到一个二进制数字第n位是否是1，我们仅需将他n位前的所有位均置为0，再`mod 2^n`，即

$$10...11\underbrace{x00\cdots0}_{n} \equiv \underbrace{x00\cdots0}_{n}(mod\ 2^n) \tag{1}$$

- 即，我们想知道一个二进制数字的第n位是否是0，仅需要将第n位前的所有位置零，并对得到的结构取模，判断是否是0即可
- 所以我们的程序的思路是这样子的，我们从二进制数字的第一位开始判断，如果`mod 2^n`不等于零，那么我们就对他减去`2^n`，并将结果存到另一个寄存器当中，留待下一次使用，我们循环这一个过程，直到循环的次数等于B，写成C语言的形式：

```
int n = 1;
int tem = A; //初始化tem=A，在每次取模后若结果不为零，则减去2^n，这样来实现第n位前的所有位均为0
int m = 0;
int result = 0;
while(B){
    n = n + n; //考虑到汇编只能实现加法运算，这里的意思是在每次循环中都对n*2，以此来实现2^n操作
    m = tem % n; //对tem取模mod 2^n
    if(!m) {
        result++; //如果取模不为零，将result加一
      tem = tem - n; //将第n位清零
    }
    B--;
}
printf("%d",result);
```

- 需要注意的是LC-3没有直接的减法指令，我们使用下面的方式实现减法操作的：
  比如我们想计算`R2 = R1 - R0`，我们首先对`R0`取反，并将他存储在`R3`中，再计算`ADD R1, R3`

```
NOT R3, R0
ADD R1, R3
```

- 同样地，LC-3并没有直接的`mod`操作，我们可以这样实现`mod`操作，比如我们需要计算`R2 = R1 mod R0`

```
      NOT R3, R0
LOOP  ADD R1, R3 ;先执行减法操作
      BRzp LOOP; ;如果得到的数是大于等于零的，继续执行减法操作，直到得到小于等于零的结果
      ADD R2, R1, R0 ;得到最后的取模的值
```

- 考虑到我们将A限制在**0x0001** to **0x7FFF**，所以位数上限是15位，我们应在程序开始的时候加一个对于B的大小的判断，对于大于15的B我们只检测到15位
- 如果B的值小于等于零，那么我们直接将结果置为0
- 于是我们综合上述过程容易写出汇编代码

```
        .ORIG x3000 ; start the program at location x3000
        AND R0, R0, x0 ; clear R0, to be used for load A
        AND R1, R1, x0 ; clear R1, to be used for load B
        LD R0, A ; load A(x3100)
        LD R0, B ; load B(x3101)
        BRp #3 ; if B > 0, jump the next three instructions
        AND R3, R3, x0 ; clear R3, to be used to store the result
        ST R1, B ; store the result in the x3102
        halt ; if B <= 0, store 0 in x3102 and exit
        AND R7, R7, x0 ; clear R7
        ADD R7, R1, #-15 ; R7 = R1 - 15
        BRnz #2 ; ;if R1 <= 15, jump the next two instructions
        AND R7, R7, x0 ; clear R7
        AND R1, R7, #15 ;B = 15
        AND R2, R2, x0 ; clear R2, to be used for the mod
        ADD R2, R2, #1 ; initialize R2, store 1 in R2
        AND R3, R3, x0 ; clear R3, to be used to store the result
        AND R4, R4, x0 ; clear R4, to be used for the temp to calculate mod
        AND R5, R5, x0 ; clear R5, to be used for calculate subtraction
        AND R6, R6, x0 ; clear R6, to be used for the temp
        ADD R6, R0, x0 ; copy A to R6
LOOP    ADD R2, R2, R2 ; R2 = R2 + R2, the begining of the loop
        NOT R5, R2 ; R5 = ~R2. for the use of R4 - R2
        ADD R5, R5, 1 ; R5 = R5 + 1
        ADD R4, R6, x0 ; copy R6 to  R4
LOOP1   ADD R4, R4, R5 ; R4 = R4 - R2
        BRzp LOOP1 ; judge if R4 < 0
        ADD R4, R4, R2 ; R4 = R6 mod 2^n
        BRz #4 ; judge if the result equals 0
        ADD R3, R3, #1 ; Result++
        NOT R5, R4 ; R5 = ~R4. for the use of R6 - R4
        ADD R5, R5, 1 ; R5 = R5 + 1
```

```
        ADD R6, R6, R5 ; R6 = R6 - R4
        ADD R1, R1, xFFFF ; B = B - 1
        BRp LOOP ; if B > 0, jump to the begining of the loop
END     ST R1, B ; store the result in the x3102
        halt
```

实验过程

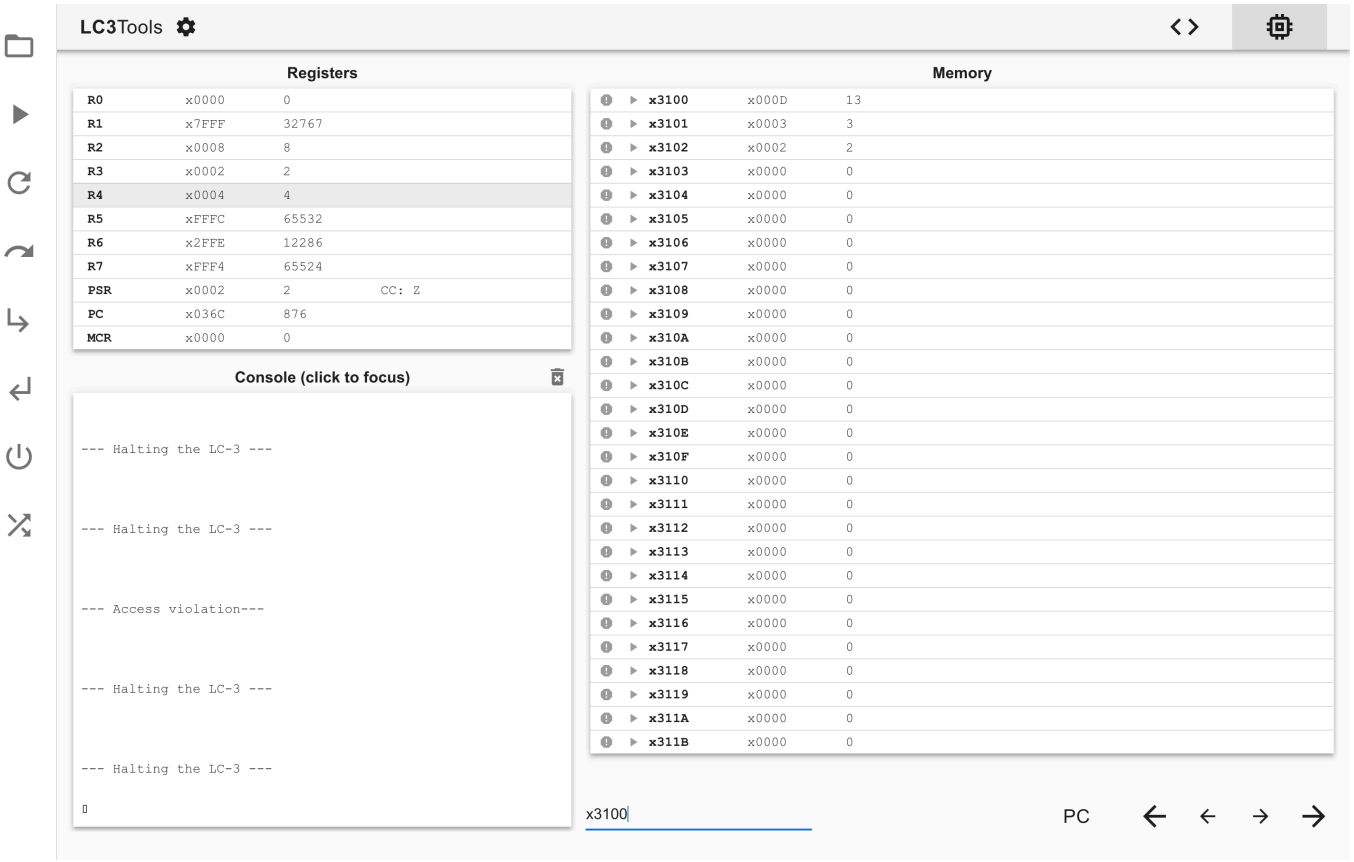
我们只需要将上述汇编代码翻译成机器代码即可

```
0011 0000 0000 0000 ; start the program at location x3000
0101 000 000 1 00000 ; clear R0, to be used for load A
0101 001 001 1 00000 ; clear R1, to be used for load B
0010 000 011111101 ; load A(x3100)
0010 001 011111101 ; load B(x3101)
0000 001 000000011 ; if B == 0, jump the next three instructions
0101 011 011 1 00000 ; clear R3, to be used to store the result
0011 011 011111011 ; store the result in the x3102
1111 0000 00100101 ; halt
0101 111 111 1 00000 ; clear R7, to be calculate if B > 15
0001 111 001 1 10001 ; R7 = R1 - 15
0000 110 000000010 ; if R7 <= 0, jump the next two instructions
0101 111 111 1 00000 ; clear R7
0001 001 111 1 01111 ; B = 15
0101 010 010 1 00000 ; clear R2, to be used for the mod
0001 010 010 1 00001 ; initialize R2, store 1 in R2
0101 011 011 1 00000 ; clear R3, to be used to store the result
0101 100 100 1 00000 ; clear R4, to be used for the temp
0101 101 101 1 00000 ; clear R5, to be used for calculate subtraction
0101 110 110 1 00000 ; clear R6, to be used for the temp3
0001 110 000 1 00000 ; copy A to R6
0001 010 010 0 00 010 ; R2 = R2 + R2, the begining of the loop
1001 101 010 111111 ; R5 = ~R2. for the use of R4 - R2
0001 101 101 1 00001 ; R5 = R5 + 1
0001 100 110 1 00000 ; copy R6 to  R4
0001 100 100 0 00 101 ; R4 = R4 - R2
0000 011 111111110 ; BRzp -2, judge if R4 < 0
0001 100 100 0 00 010 ; R4 = R6 mod 2^n
0000 010 000000100 ; judge if the result equals 0
0001 011 011 1 00001 ; R3 = R3 + 1
1001 101 100 111111 ; R5 = ~ R4
0001 101 101 1 00001 ; R5 = R5 + 1
0001 110 110 0 00 101 ; R6 = R6 - R4
0001 001 001 1 11111 ; B = B - 1
```

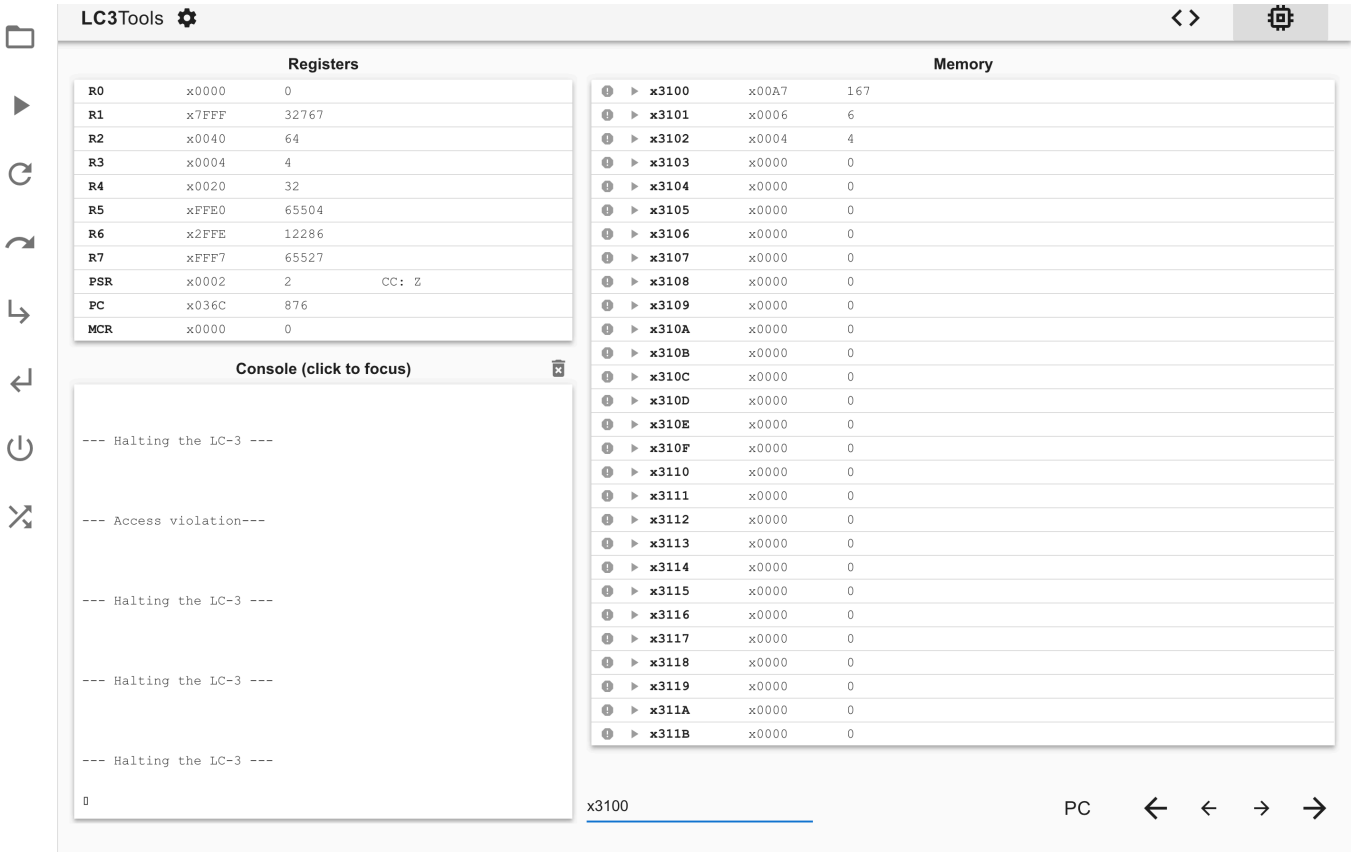
```
0000 001 111110010 ; if B > 0, jump to the begining of the loop
0011 011 011011111 ; store the result in the x3102
1111 0000 00100101 ; halt
```

实验结果

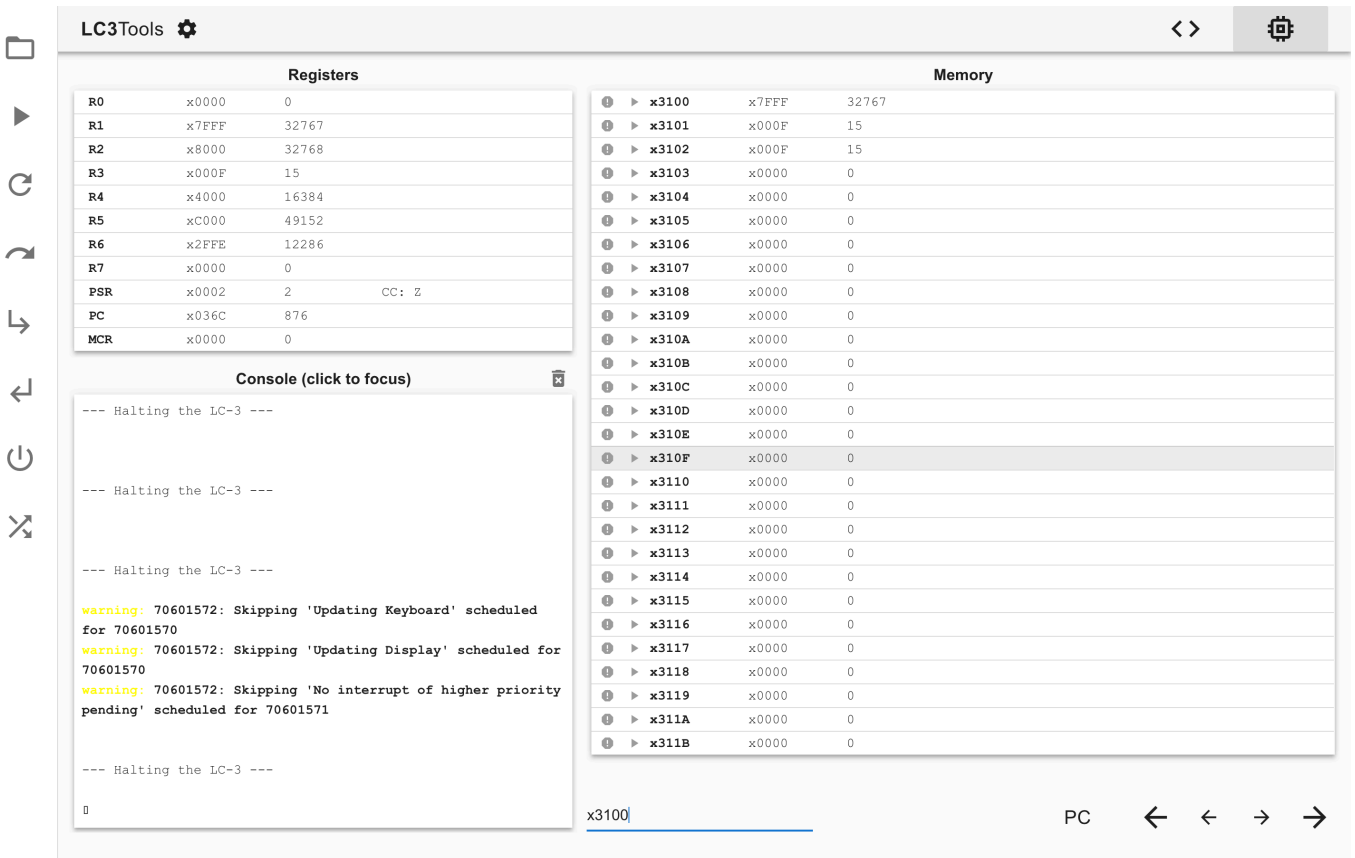
对于三组测试结果：

- 

-

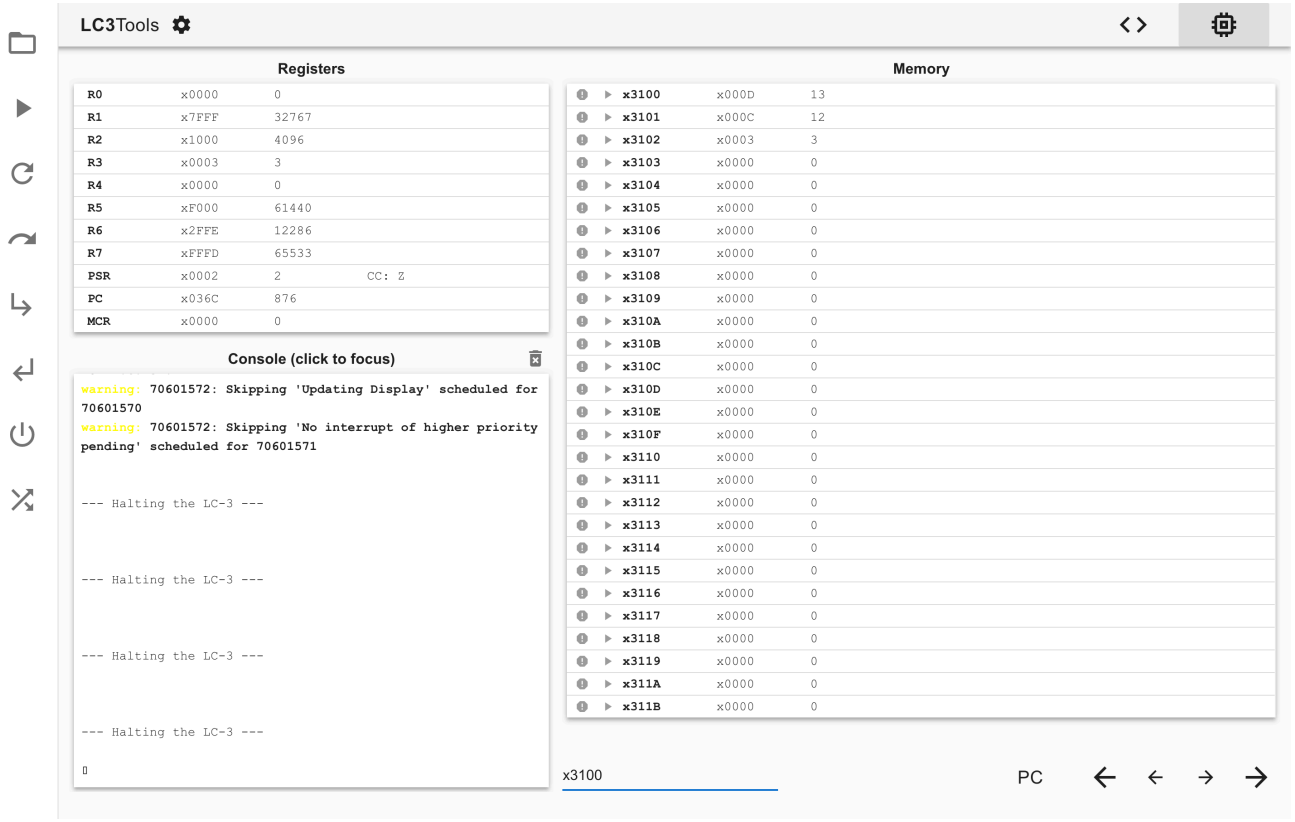


-

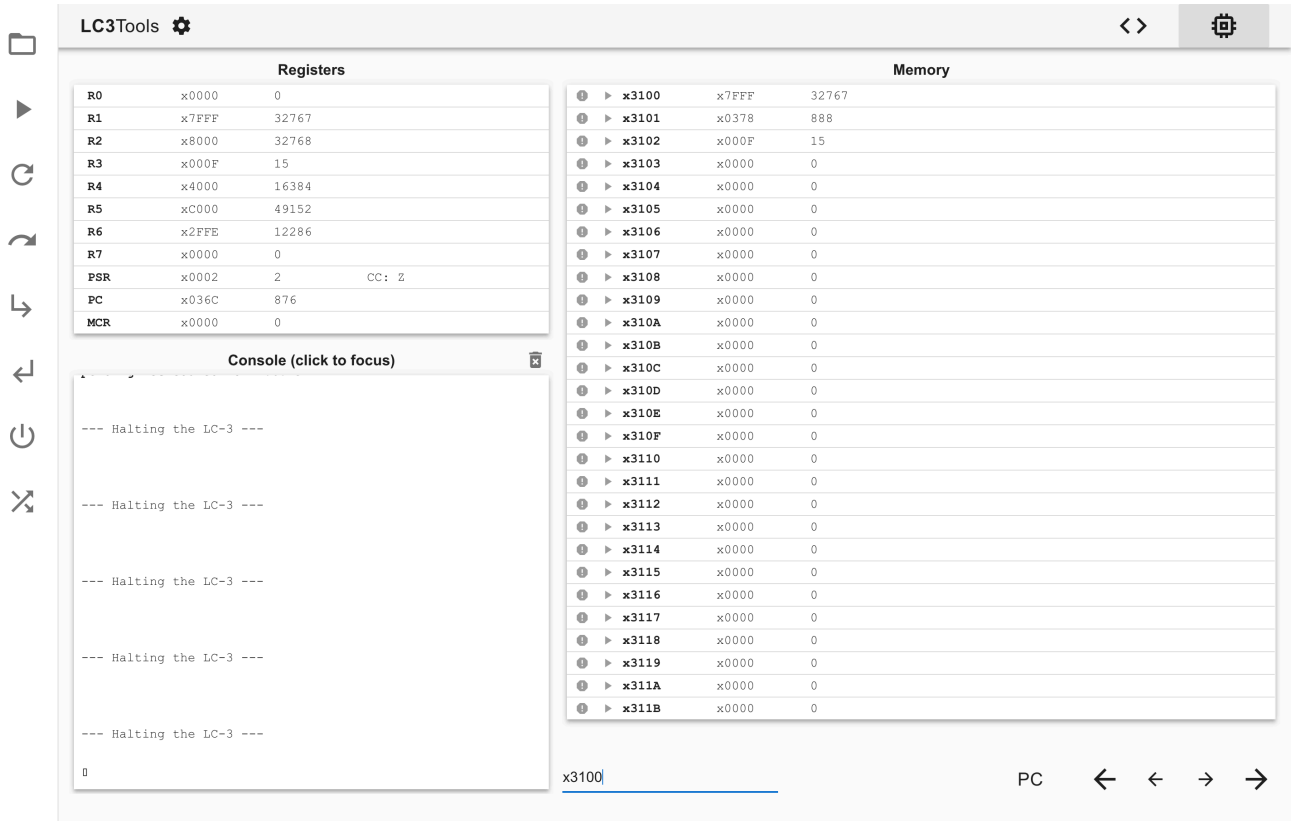


- 我们可以看到程序的鲁棒性非常棒，只要A在题目中规定的范围内，我们针对任何大小的B都有相应的处理

- 对于B的位数大于A的实际位数，只要将A拓展依然能得到正确结果



- 对于B的位数超过15，依然能正确处理



- 对于B=0或B<0的情况，我们将0存入x3102中，并退出程序