Lab 1 Report

Purpose

For this lab, we are asked to write a program in LC-3 machine language that counts how many 1s are in the lower B bits of a given number A, and stores the output in memory.

The program starts at memory location x3000. The value of A and B are set manually in x3100 and x3101 respectively. A is assumed as a positive number ranging from 0x0001 to 0x7FFF. The program stores the output in x3102.

Principles

Since our purpose is to count how many 1s there are in the lower B bits of the number A, we can natually think of performing a bit-wise comparison between A and 1, so as to figure out whether or not each bit is 1.

Therefore we can write the following pseudocode for our program:

```
initialize A, B;
weight = 0;
judge = 1;
for (i = 0 to B - 1)
        temp = A & judge;
        judge *= 2;
        if (temp != 0) ++weight;
return weight;
```

Procedure

turning the pseudocode into binary code

There are mainly two problems we need to focus on: how to write the loop, and how to renew the value of judge.

For the loop, we can use the BRz instruction. When judging whether or not a bit in A is equal to 1, with every judgement B minus 1, so long as B is not equal to 0, we jump back to the front to continue the cycle; when B is equal to 0, the loop ends and the result is stored.

For the renewal of judge, we would like to have an instruction that allows us to do bitwise movement so that we can move judge to the right. Unfortunately no such instruction exists in LC-3, so instead we use the ADD instruction, which allows us to double judge and rewrite it back into the program.

writing the code step by step

1. Use the LD instruction to extract the data from x3100 and x3101 and write them into R0, R1.
   `0010000011111111 ;LD R0, x00F`
   `0010001011111111 ;LD R1, x00F`
2. Use the AND instruction to clear R2, R3, R4 in order to prevent any effects the registers' current value may have on the result. Note that R2 is the judge, R3 is the judgment, and R4 is the weight.
   `0101010010100000 ;AND R2 , R2 , x00`
   `0101100100100000 ;AND R4 , R4 , x00`
   `0101011011100000 ;AND R3 , R3 , x00`
3. Use the BRp instruction to dtermine whether or not it is R2's first time entering the loop. If so, then there need be no bit movement.
   `0000001000000001 ;BRp x001`
4. Use ADD to shift the data in R2.
   `0001010010000010 ;ADD R2 , R2 , R2`
5. Use AND to determine if a bit of R0 is 1.
   `0101011000000010 ;AND R3 , R0 , R2`
6. Use BRz, if the result isn't 0, then increment R4 by 1; if it is, R4 does not increase.
   `0000010000000001 ;BRz x001`
   `0001100100100001 ;ADD R4 , R4 , x01`
7. Use ADD to do minus 1 to R1.
   `0001001001111111 ;ADD R1 , R1 , x1F`
8. Use BRp, if R1 isn't 0 then continue the cycle above until R1 is equal to 0.
   `0000001111111010 ;BRp x1FA`
9. Save the result to x3102.
   `0011100011110100 ;ST R4 , x0F4`

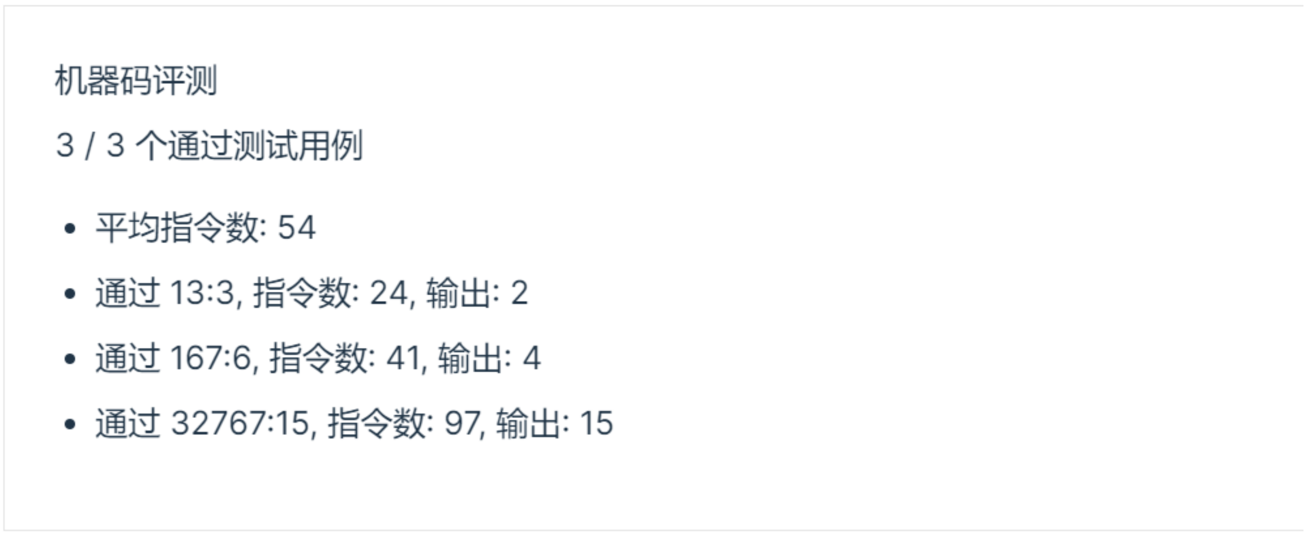
the code

```
0011000000000000
0010000011111111 ;LD R0, x00F
0010001011111111 ;LD R1, x00F
0101010010100000 ;AND R2, R2, x00
0101100100100000 ;AND R4, R4, x00
0101011011100000 ;AND R3, R3, x00
0001010010100001 ;ADD R2, R2, x01
```

```
0000001000000001 ;BRp x001
0001010010000010 ;ADD R2, R2, R2
0101011000000010 ;AND R3, R0, R2
0000010000000001 ;BRz x001
0001100100100001 ;ADD R4, R4, x01
0001001001111111 ;ADD R1, R1, x1F
0000001111111010 ;BRp x1FA
0011100011110100 ;ST R4, x0F4
1111000000100101
```

Result

Using the online judge a TA provided, here are the results for tests performed on my code:


机器码评测

3 / 3 个通过测试用例

- 平均指令数: 54
- 通过 13:3, 指令数: 24, 输出: 2
- 通过 167:6, 指令数: 41, 输出: 4
- 通过 32767:15, 指令数: 97, 输出: 15

To this point we can assume that the basic task asked of us has been completed correctly.