一、 实验目的

1、 熟练使用无条件跳转指令和条件跳转指令。

2、 熟练掌握各个指令对 Flags 的影响。能够根据简单的判断设计分支程序。

3、 熟练指令寻址，以及变量的操作。

4、 能够通过 emu 查看各种所需要的信息，以便验证自己的结论。

二、 实验设备

硬件环境：个人通用 PC

软件环境：Emu8086

三、 实验内容和要求

1、已定义两个整数变量 A 和 B，试编写程序完成以下功能

（要求用两种实现方法）

（1）若两个数有一个是奇数，则将奇数存入 A 中，偶数存入 B 中

（2）若两个数均为奇数，则将两个数加 1 后存回原变量

（3）若两个数均为偶数，则两个变量均不变

2、测试在 STATUS 中的一个字节，如果第 1、3、5 位均为 1，则转移到 ROUTINE_1；

如果此三位中有两位为 1，则转移到 ROUTINE_2；如果此三位中只有一位为 1，则转移到 ROUTINE_3；如果此三位全为 0，则转移到 ROUTINE_4。

四、 实验步骤

1、实验 1（方法 1）

设计思路：第一种设计思路是先判断 A 是否是奇数，然后再判断 B 是否是奇数。如果我们一开始 AL:=A,BL:=B（:=为赋值运算）。则在(AL=奇，BL=奇)和(AL=偶，BL=奇)时才需要执行操作。其中判断奇数可以使用 test ?,1 + jz。在关键的分支点定义符号，可以实现跳转。

伪代码：

```
al := a
bl := a
if (al 是奇数):
    if (bl 是奇数):
        // 做+1 的操作
    else:
        do nothing
else: //entry<branch>
    if (bl 是奇数):
        // 做交换的操作
```

```
    else:
        do nothing
//entry<exit>
```

流程图：

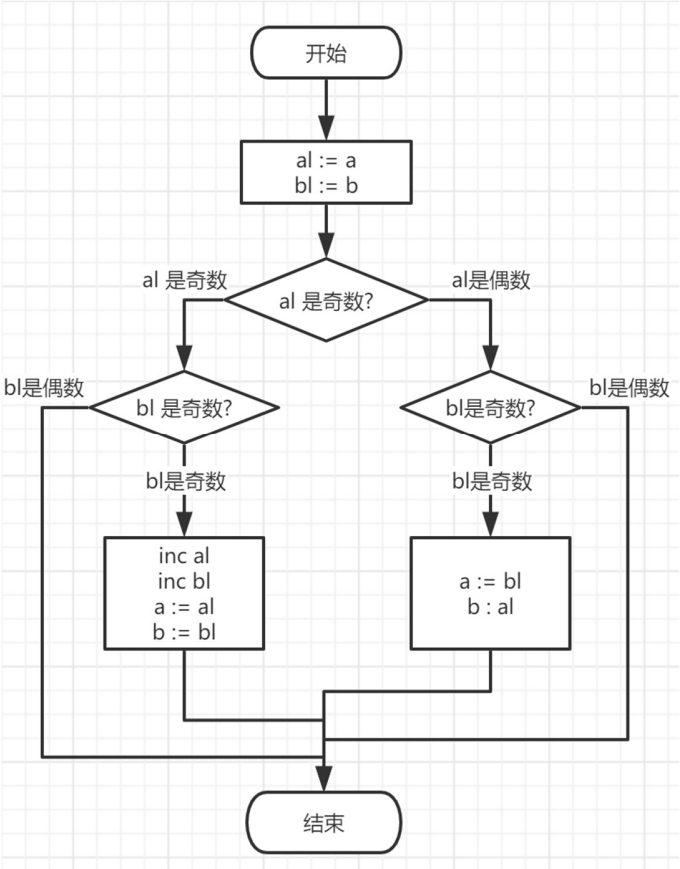

图 2-1 流程图

实验代码：

```
    mov al, a
    mov bl, b
    test al, 1 ; justify whether the al is odd
    jz branch ; jump to branch is al is even.
    test bl, 1 ; justify whether the bl is odd.
    jz exit ; jump to exit is al & bl is eve.
    ; the code will be excuted if al & bl is odd.
    inc al
    inc bl
    mov a, al
    mov b, bl
    jmp exit ; to the exit
```

```
branch:
    test bl, 1
    jz exit ; jump to exit is al & bl is even.
    mov a, bl
    mov b, al
exit:
```

实验结果：

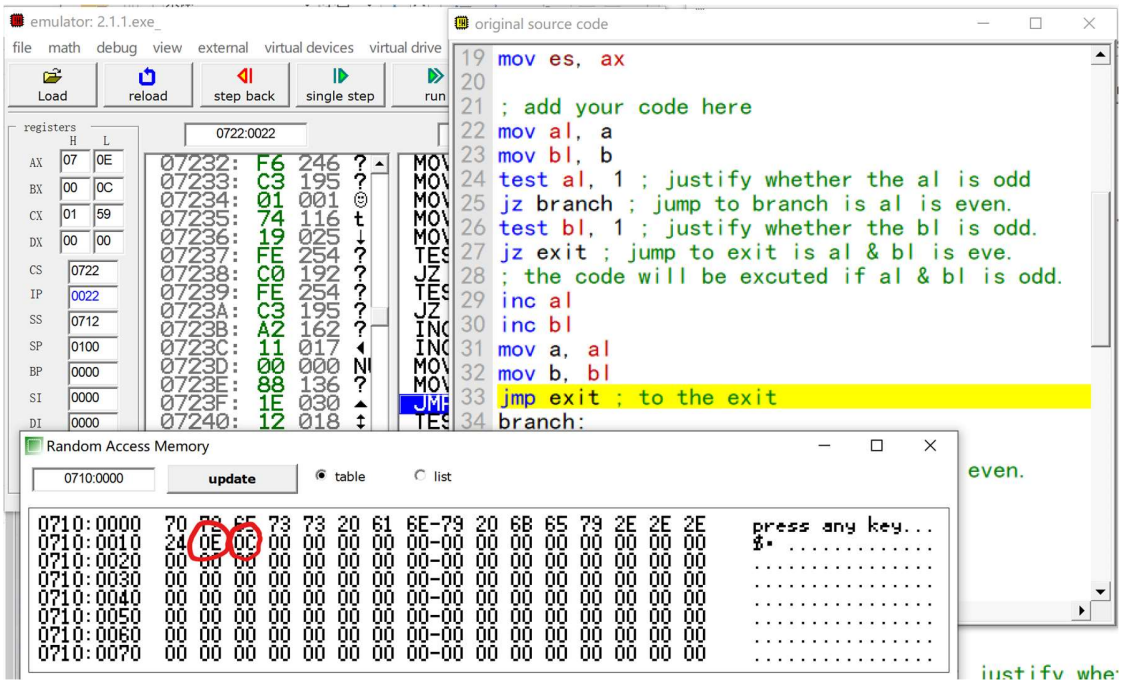

图 2-2 实验结果

（在 a,b 均是奇数的验证结果，执行前 a=13,b=11，执行后 a=14,b=12，符合预期）

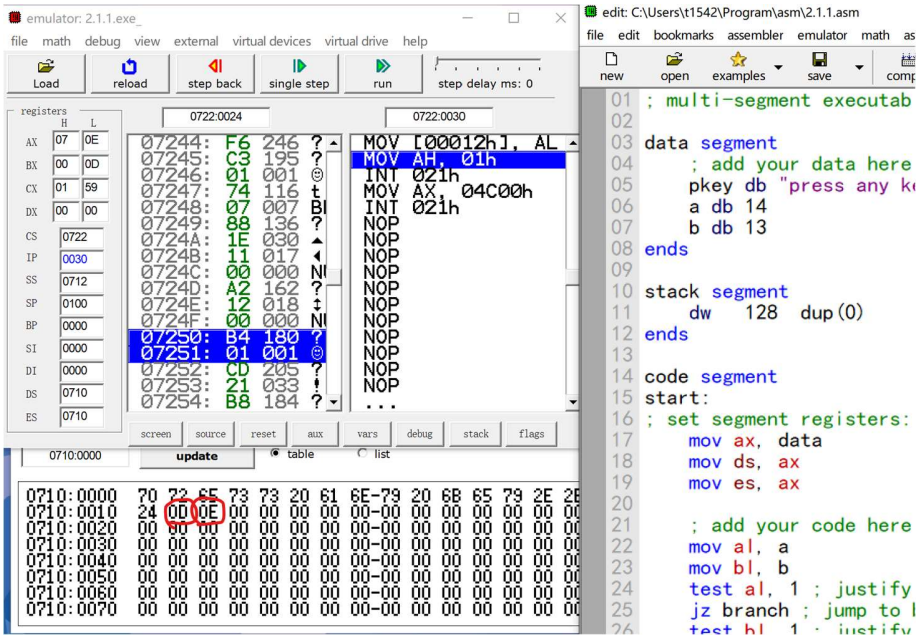

图 2-3 实验结果

（在 a 是偶数，b 是奇数的验证结果，执行前 a=14, b=13，执行后 a=13, b=14，符合预期）

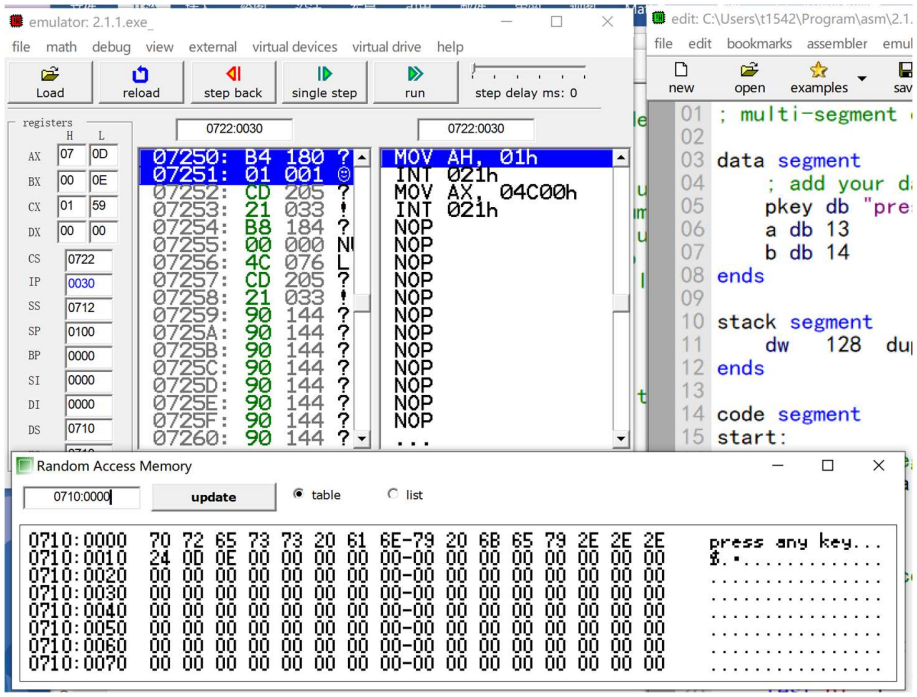

图 2-3 实验结果

（在 a 是偶数，b 是奇数的验证结果，执行前 a=13, b=14，执行后 a=13, b=14，符合预期）

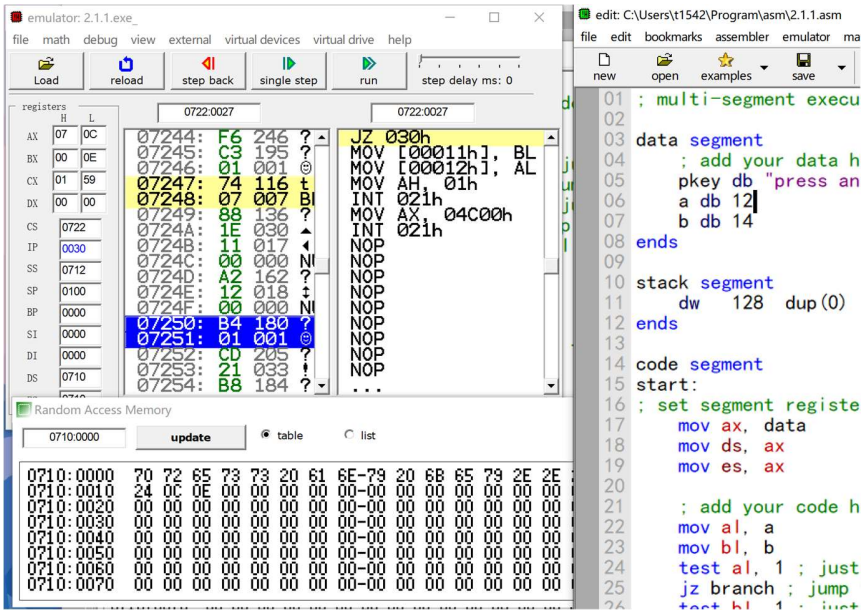

图 2-4 实验结果

（在 a 是偶数，b 是奇数的验证结果，执行前 a=12,b=14，执行后 a=12,b=14，符合预期）

2、实验 1（方法 2）

设计思路：将四种情况分成两类情况（a,b 奇偶性相同，记为情况 M，a,b 奇偶性相异，记为情况 N）。情况 M 下，两个变量可以通过+(1/0)将两种子情况合并为 1 种。情况 N 下，为是否变量交换的操作。

流程图：

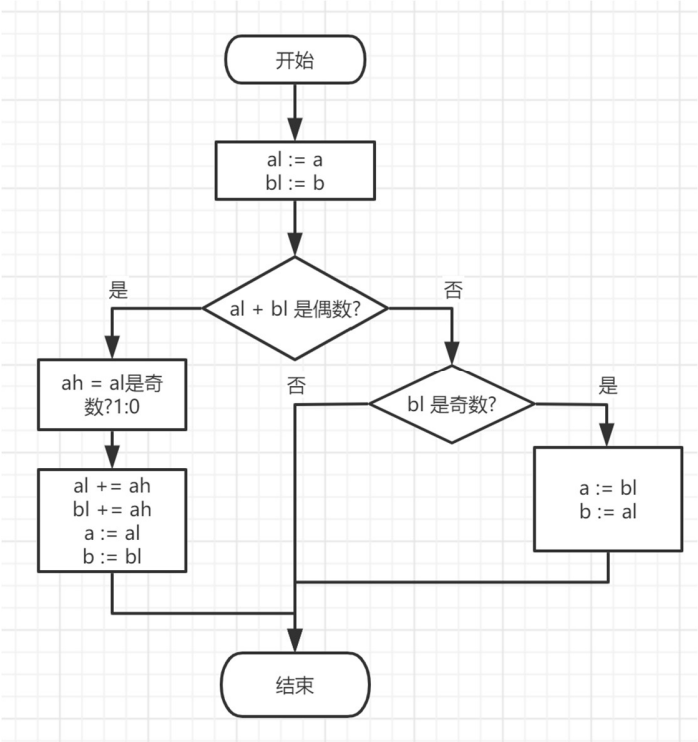

图 2-5 路程图

实验代码：

```
    mov al, a
    mov bl, b
    mov ah, al
    add ah, bl
    and ah, 1 ; justify if the a+b is odd.
    jnz branch
    ; a+b is odd
    mov ah, al
    and ah, 1
    add al, ah
    add bl, ah
    mov a, al
    mov b, bl
    jmp exit
branch:
    test bl, 1 ; justify if the b is odd
    jz exit
    mov a, bl
    mov b, al
```

实验结果：

实验结果与方法 1 相同，因此不再截图。

3、实验 2

设计思路：此实验主要在于统计 1 出现的次数，用一个寄存器(ah)，使用 and 后，ah 将保存与运算后的结果，其 1,3,5 为与原来数字相同，其他位均为 0，统计次数（使用 loop+shl+adc 来实现统计 1 的个数）后，使用 cmp + jz 来实现跳转。

流程图：

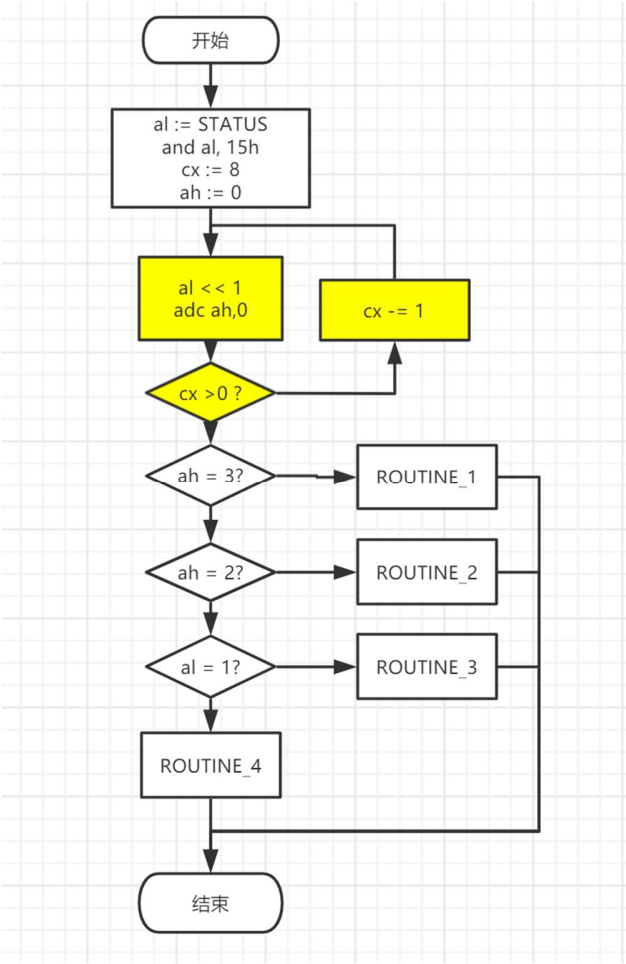

图 2-6 流程图

实验代码：

```
mov al, STATUS
    and al, 15h
    mov ah, 0
    mov cx, 8
; calculate the count of 1 in al
```

```
; and store the result to ah
calculate:
    shl al, 1 ; left move
    adc ah, 0 ; add with carry
    loop calculate
    cmp ah, 3h
    jz ROUTINE_1
    cmp ah, 2h
    jz ROUTINE_2
    cmp ah, 1h
    jz ROUTINE_3
    jmp ROUTINE_4
    ; jmp exit
ROUTINE_1:
    mov bl, 3
    jmp exit
ROUTINE_2:
    mov bl, 2
    jmp exit
ROUTINE_3:
    mov bl, 1
    jmp exit
ROUTINE_4:
    mov bl, 0
    jmp exit
```

实验结果：

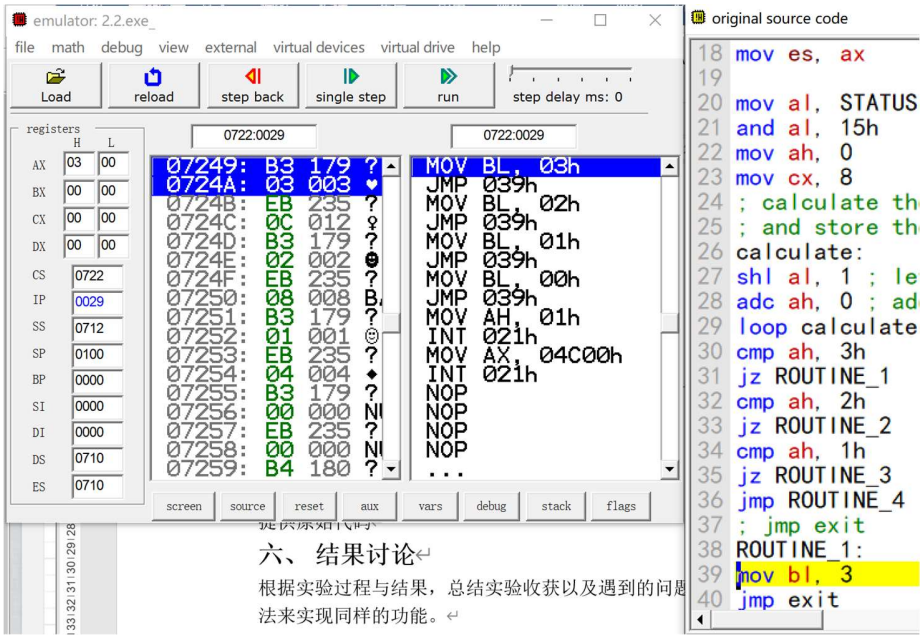

图 2-7 实验结果

（STATUS=00010101B 的实验结果，符合预期）

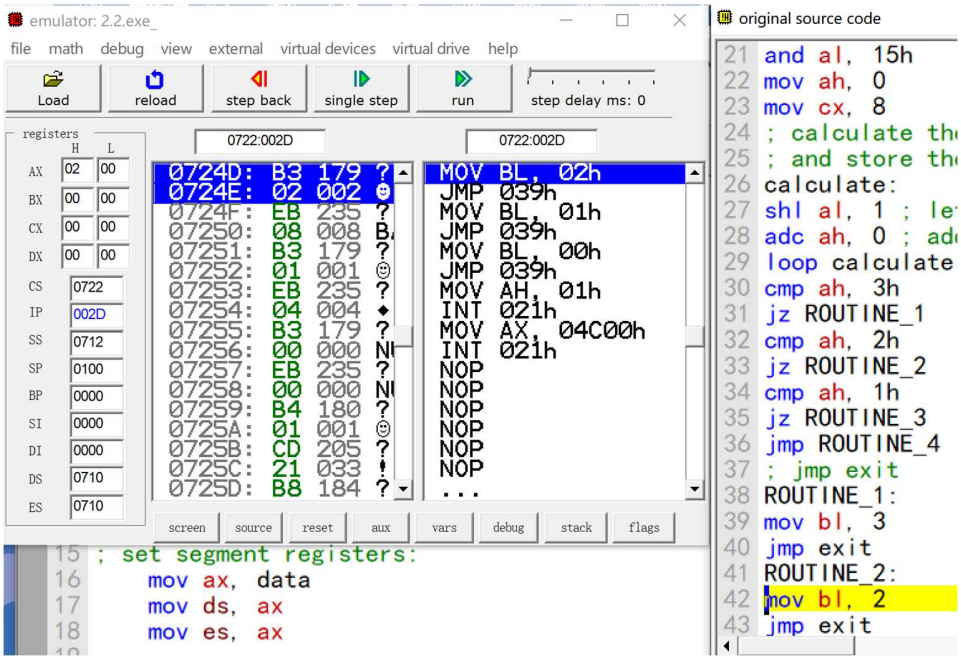

图 2-8 实验结果

（STATUS=00010001B 的实验结果，符合预期）

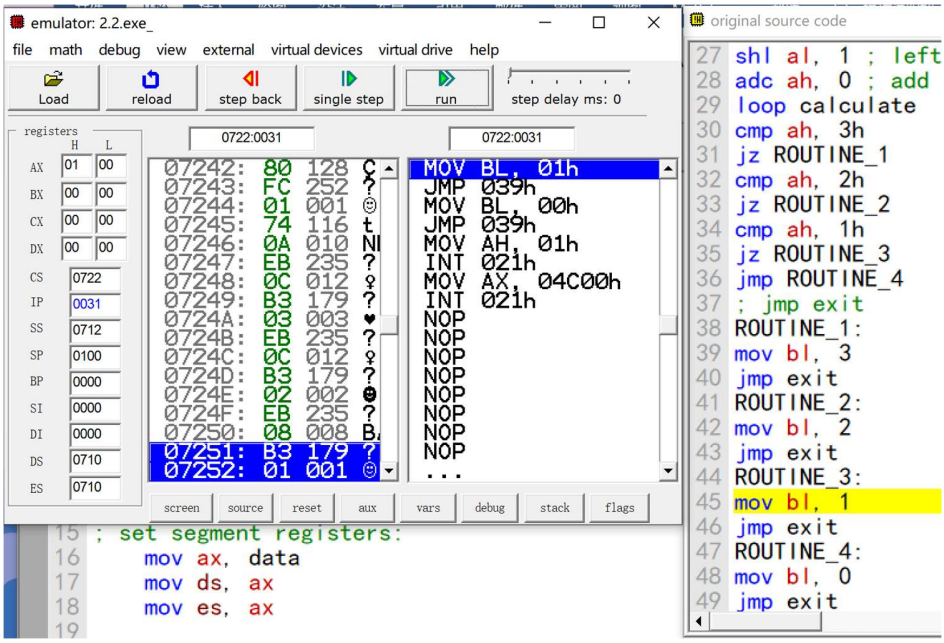

图 2-9 实验结果

（STATUS=00010001B 的实验结果，符合预期）

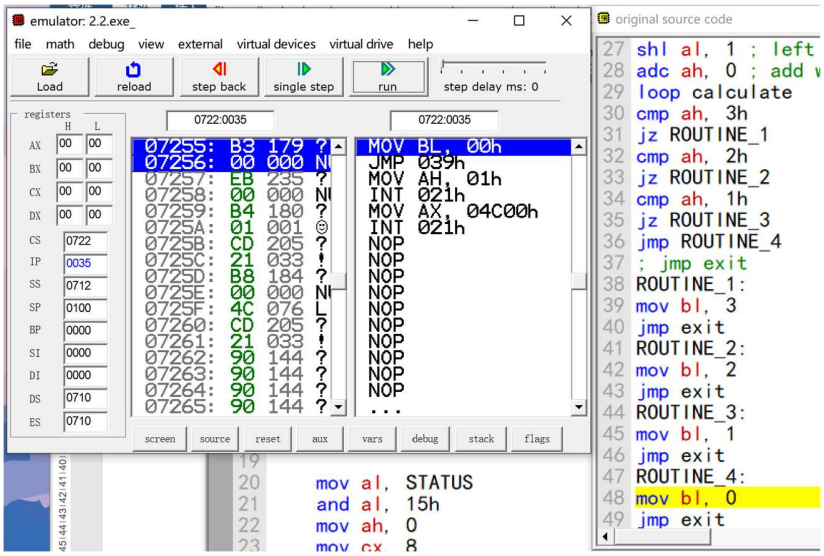

图 2-10 实验结果

（STATUS=00001010B 的实验结果，符合预期）

五、 实验结果分析

1、实验 1 结果分析

实验 1 的结果符合预期，在四种情况下代码的运行步骤和理论是一致的。在方法 1 中，使用了传统的分支判断方法，跳转的位置是正确的，并没有奇怪的地方。方法 2 中，需要两次使用到奇偶判断的结果（一次是 a+b，另外一次是 a），且都需要使用到判断结果（如果只是跳转只需要使用 test），因

此使用了 and（and 会改变寄存器的值，所以 and 之前又要 mov），导致寄存器操作次数过多，效率降低。

2、实验 2 结果分析

使用了 LOOP 来进行循环，使用经典的逻辑左移+ADC 来统计 1 的个数。因为之前使用了 and al, 15h，所以能够截取 1，3，5 三位的数字，因此能够正确的统计个数。

六、 结果讨论

1、实验 1 中方法 1 使用了简单的判断方法，而方法 2 更加复杂些。从效率看，方法 1 是更优的。不过，在方法 1 中，如果 b 是偶数，则一定不需要进行操作，因此可以通过先判断 b 来减少判断的次数，从而提高程序的效率。

2、实验 2 如果使用 1 中传统的判断，则判断次数会过多，导致效率降低，因此使用了统计个数的方法。统计个数使用了循环左移的方式，虽然代码简单，但仍然存在不必要的步骤（在此例中 2,4,6 位是明显不需要的），可以通过添加代码跳过这几位，从而提高效率，但同时，代码会复杂些。

3、可以发现，分支程序在高等程序和汇编中虽然可以通过跳转指令实现等价的，但是汇编的跳转指令容易破坏代码的上下文结构，从而让代码阅读起来困难。一种常用的思路是先编写分支语句，在转化为等价的条件跳转指令。