lab1 实验报告

李远航
PB20000137

1. 实验内容

- 本次实验任务是实现乘法，写出对应程序机器码。两个运算数分别放置于 `R0` 和 `R1`，结果需要存储到 `R7`，其他寄存器状态不做限制（即不限结束状态）。初始状态：`R0`和`R1`存放待计算数，其余寄存器全部为0。
- 评估自己程序的代码行数、完成实验功能所需要执行的指令数。
- 提交两个版本的代码。L版本尽量编写更少的代码行数，P版本尽量让程序执行更少的指令。

2. L版本程序

- 设计思路
  乘法可以通过循环表示成加法的形式，如2×3可以表示成$2 + 2 + 2$的形式，具体思路如下所示（需要注意区分正负号）

```cpp
#include <iostream>
short times(short a, short b)
{
    short flag = -1;
    if (b < 0)
        flag = 1;
    short answer = 0;
    while (b != 0)
    {
        answer += a;
        b += flag;
    }
    return (-flag) * answer;
}
int main()
{
    short a, b;
    std::cin >> a >> b;
    std::cout << times(a, b) << std::endl;
    return 0;
}
```

- 将上述`c++`代码转化成机器码的形式，得到最初版L版本程序(与最终版相同)

```
0001 010 010 1 11111    ;R2=-1
0001 001 001 1 00000    ;计算R1
0000 010 000001001      ;R1==0直接结束
0000 001 000000001      ;如果R1>0跳转
0101 010 010 1 00001    ;R2=R2&1(R2=1)
0001 111 111 0 00 000   ;R7=R7+R0
0001 001 001 0 00 010   ;R1=R1+R2
0000 101 111111101      ;循环
0001 010 010 1 00000    ;为了判断对R2进行运算
0000 100 000000010      ;判断
1001 111 111 111111     ;R7=~R7
0001 111 111 1 00001    ;R7=R7+1
```

由于机器码程序运行的过程与`C++`代码运行过程一致，对溢出情况的处理也会相同

- 测试样例运行结果：
  - 1×1

R7	x0001	1
  - 5×4000

R7	x4E20	20000
  - 4000×5

R7	x4E20	20000
  - -500×433

R7	xB24C	-19892
  - -114×-233

R7	x67C2	26562
- L版本程序使用了12行代码，完成实验功能执行的代码条数取决于`R1`寄存器存储的数，在极限情况下，只需要执行3步，但是同时，如果`R1`寄存器的数过大， 执行的指令数会非常多

3. P版本程序

- 设计思路

  完全使用加法的程序显然会浪费过多的时间，可以使用位运算的方式来进行乘法，具体的实现思路如下`C++`代码所示：

```cpp
#include <iostream>
short times(short a, short b)
{
    if (b == 0)
        return 0;
    int flag = 0;
    if (b < 0)
    {
        flag = 1;
        b = -b;
    }
    short answer = 0;
    while (b)
    {
```

```
        if (b & 1)
            answer += a;
        a = a << 1;
        b = b >> 1;
    }
    if (flag == 1)
        answer = -answer;
    return answer;
}
int main()
{
    short a, b;
    std::cin >> a >> b;
    std::cout << times(a, b) << std::endl;
    return 0;
}
```

- 将上述`C++`代码转化成机器码的形式，得到最初版P版本程序(与最终版本相同)

  注意到`LC-3`中没有左移右移操作码，但是左移操作码相当于$\times 2$，所以可以用`ADD`指令替代，但是较难实现右移操作，但是可以通过对1进行左移再`&`的方法，来间接的实现右移操作

```
0001 010 010 1 00001    ;R2=1
0101 110 001 1 10000    ;计算R6
0000 011 000000010      ;如果R6>=0跳转
1001 001 001 111111     ;R1=~R1
0001 001 001 1 00001    ;R1=R1+1
0101 011 010 0 00 001   ;R3=R2&R1
0000 010 000000001      ;判断
0001 111 111 0 00 000   ;R7=R7+R0;
0001 000 000 0 00 000   ;R1<<1
0001 010 010 0 00 010   ;R2<<1
0000 001 111111010      ;R2>0?
0001 110 110 1 00000    ;运算
0000 011 000000010      ;判断
1001 111 111 111111     ;R7=~R7
0001 111 111 1 00001    ;R7=R7+1
```

  由于机器码程序运行的过程与`C++`代码运行过程基本一致，对溢出情况的处理也会相同

- 测试样例运行结果：
  - 1×1

R7	x0001	1

  - 5×4000

R7	x4E20	20000

  - 4000×5

R7	x4E20	20000

  - -500×433

R7	xB24C	-19892

  - -114×-233

```
R7              x67C2           26562
```

- 代码行数及判断程序运行的时间

  P版本程序使用了15行代码

  要观察程序运行时经过的指令数，可以用未使用的寄存器来储存执行的指令数，这里我使用`R5`来记录程序执行的步骤，修改后的机器码如下所示，注意修改跳转的步数

```
0001 101 101 1 00011    ;记录步数（计算 判断 R2=1）
0001 010 010 1 00001    ;R2=1
0101 110 001 1 10000    ;计算R6
0000 011 000000011      ;如果R6>=0跳转
1001 001 001 111111     ;R1=~R1
0001 001 001 1 00001    ;R1=R1+1
0001 101 101 1 00010    ;记录步数（R1=~R1 R1=R1+1）

0001 101 101 1 00010    ;记录步数(R3=R2&R1 判断)
0101 011 010 0 00 001   ;R3=R2&R1
0000 010 000000010      ;判断

0001 101 101 1 00001    ;记录步数(R7=R7+R0)
0001 111 111 0 00 000   ;R7=R7+R0

0001 101 101 1 00011    ;记录步数(R1<<1 R2<<1 R2>0?)
0001 000 000 0 00 000   ;R1<<1
0001 010 010 0 00 010   ;R2<<1
0000 001 111110111      ;R2>0?

0001 101 101 1 00010    ;记录步数（运算 判断）
0001 110 110 1 00000    ;运算
0000 011 000000011      ;判断

0001 101 101 1 00010    ;记录步数（R7=~R7 R7=R7+1）
1001 111 111 111111     ;R7=~R7
0001 111 111 1 00001    ;R7=R7+1
```

  测试样例运行步骤如下所示：

  - 1×1

    ```
    R5              x0051           81
    ```

  - 5×4000

    ```
    R5              x0056           86
    ```

  - 4000×5

    ```
    R5              x0052           82
    ```

  - -500×433

    ```
    R5              x0055           85
    ```

  - -114×-233

    ```
    R5              x0059           89
    ```

  在最极端的情况下，该种方式执行的指令也不会超过100条，符合题目要求

相较于L版本的程序，虽然在一些情况下会多运行很多冗余的行数，但是在更多情况下，移位的写法，运行的指令数要显然优于L版本

4. 实验总结

- 更加深入了解了LC-3的指令
- 学会了用机器码编写简单的程序
- 认识到实现同一个目的，有很多不同的方法，对不同目的可以有相应的解决方案