Lab 3 实验报告

- 姓名：谭旭
- 学号：PB20030775

实验内容

通过阅读另一位同学的 Lab2 代码，推测其学号，并优化其代码，使得优化后的代码运行时实际执行指令数尽可能少。

实验过程

Step 1

该同学程序中后四行填充的数字为

```
.FILL #930
.FILL #246
.FILL #386
.FILL #646
```

通过以下 c++ 代码，求得 $F(0)$ 到 $F(99)$，并与上述结果比对。

```cpp
#include <cstdio>

int f[100] = {1, 1, 2};
int ans[4] = {930, 246, 386, 646};

int main()
{
    for (int i = 3; i < 100; i++)
    {
        f[i] = (f[i - 1] + f[i - 3] * 2) % 1024;
    }
    for (auto x : ans)
    {
        for (int i = 0; i < 100; i++)
        {
            if (x == f[i])
            {
                printf("%d", i);
                break;
            }
        }
    }
    return 0;
}
```

得到该同学学号为

```
20111677
```

Step2

Ver 1

考虑到实验要求使用尽可能少的指令数完成对 $F(n)$ 的计算，

原先的程序为了追求行数尽可能短，在执行的指令数上并不优秀。

考虑到实验要求输出的结果对 1024 取模，很容易想到，该数列一定存在循环，即

$$\exists T, \forall n, F(n+T) = F(n)$$

使用一 `c++` 代码求出其循环节长度

```cpp
#include <cstdio>

int main()
{
    int a = 1;
    int b = 2;
    int c = 4;
    int d = 0;
    int cnt = 0;
    do
    {
        cnt++;
        d = c;
        c += a << 1;
        c &= 1023;
        a = b;
        b = d;
    }while (!(a == 1 && b == 2 && c == 4) && cnt != -1);
    printf("%u", cnt);
    return 0;
}
```

可惜的是，经过测试，循环节长度并不在 2^{16} 范围内，因此对于 16 位的 lc3 并无优化作用，该方案不可行。

Ver 2

考虑到测试点数据仅有 10 个，且其中的最大值为 `14000` ，

可以将所有不大于 `14000` 的计算结果数据按顺序保存在内存中，

对于输入的 R0 ，使用指令 `LDR` 直接读取对应位置的数据，得到答案。

而 `14000` 也就是 `x36B0` ，

lc3 的 UserSpace 为 `x3000` 到 `xFDFF` ，

将上述计算结果数据保存至内存，需要占用的空间不大于 UserSpace ，因此该方案可行。

以下是生成 lc3 代码的 `c++` 代码

```cpp
#include <cstdio>

int main()
{
    int n = 14000;
    int f[14001] = {1, 1, 2};
    freopen("code.txt", "w", stdout);
    printf("\t.ORIG x3000\n");
    printf("\tLD R1, NUM\n");
    printf("\tADD R0, R0, R1\n");
    printf("\tLDR R7, R0, #0\n");
    printf("\tTRAP x25\n");
    printf("NUM\t.FILL x3005\n");
    for (int i = 3; i <= n; i++)
    {
        f[i] = (f[i - 1] + 2 * f[i - 3]) % 1024;
    }
    for (int i = 0; i <= n; i++)
    {
        printf("\t.FILL #%d\n", f[i] % 1024);
    }
    printf("\t.END");
    return 0;
}
```

生成得到的部分 lc3 代码如下

```
    .ORIG x3000
    LD R1, NUM
    ADD R0, R0, R1
    LDR R7, R0, #0
    TRAP x25
NUM .FILL x3005
```

该代码首先将 $F(0)$ 所位于的地址读入 R1 ，并令 R0 与之相加，

最终读取 R0 所指向的内存，得到 $F(n)$ 。

该程序运行时实际执行的指令条数为 3 。