lab5 实验报告

李远航
PB20000137

1. 实验内容

- task

```
int judge(int r0) {
    int i = 2;
    r1 = 1;
    while (i * i <= r0) {
        if (r0 % i == 0) {
            r1 = 0;
            break;
        }
        i++;
    }
    return r1;
}
```

将上述内容转化为LC-3汇编码

2. 设计思路及代码

- 题目中给出的函数是一个循环，所以可以用跳转指令来实现乘法部分可以使用直接加法循环的方式，求余数的过程是一个不断减的过程，为了方便判断余数，可以使判断余数的循环到减成负数截止，再重新加上一次这个数字，便可以获得余数，一旦有一个余数判断为0，则直接跳出循环，该思路下的代码如下所示:

```
.ORIG x3000
ADD R1, R1, #1
ADD R2, R2, #2
NOT R3, R0
ADD R3, R3, #1
ADD R4, R2, R3
BRzp #3
JSR JUDGE
ADD R4, R3, R4
BRnz JUDGE
HALT
JUDGE ADD R4, R0, #0
NOT R5, R2
ADD R5, R5, #1
ADD R4, R4, R5
BRzp #-2
ADD R4, R4 ,R2
BRnp #2
AND R1, R1, #0
HALT
ADD R2, R2, #1
AND R4, R4, #0
ADD R5, R2, #0
```

```
ADD R4, R4, R2
ADD R5, R5, #-1
BRp #-3
RET
.END
```

- 如果不完全按照`cpp`代码来，每次递增时显然可以只判断是否是素数的倍数(但是奇数更好实现，部分数字可以特判），同时偶数显然不是素数，也可以特判，这样程序运行所需要执行的指令数会大幅缩小，依次可以获得以下的代码

```
.ORIG x3000
ADD R1, R1, #1
ADD R3, R3, #-3
ADD R3, R3, R0
BRnz #10
AND R1, R1, #0
AND R3, R0, #1
BRz #7
ADD R1, R1, #1
ADD R2, R2, #3
NOT R3, R0
ADD R3, R3, #1
JSR JUDGE
ADD R4, R3, R4
BRnz JUDGE
HALT
JUDGE ADD R4, R0, #0
NOT R5, R2
ADD R5, R5, #1
ADD R4, R4, R5
BRzp #-2
ADD R4, R4 ,R2
BRnp #2
AND R1, R1, #0
HALT
ADD R2, R2, #2
AND R4, R4, #0
ADD R5, R2, #0
ADD R4, R4, R2
ADD R5, R5, #-1
BRp #-3
RET
.END
```

- 再进一步的，由于题目只需要判断一定范围内的素数，可以将区间内的每个数字全部判断出来`.FILL`在程序内，就可以让指令数下降到3

```
.ORIG x3000
LD R2,PTR
ADD R2, R0, R2
LDR R1, R2, #0
HALT
PTR .FILL #12293
.Fill #1
.Fill #1
……
.END
```

3. 实验收获

- 熟悉了LC-3函数的书写方式
- 更加深刻认识到程序优化的重要性