

lab5 实验报告

李远航

PB20000137

1. 实验内容

- task

```
1  int judge(int r0) {
2      int i = 2;
3      r1 = 1;
4      while (i * i <= r0) {
5          if (r0 % i == 0) {
6              r1 = 0;
7              break;
8          }
9          i++;
10     }
11     return r1;
12 }
```

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：

```
1  .ORIG x3000
2  ADD R1, R1, #1
3  ADD R2, R2, #2
4  NOT R3, R0
5  ADD R3, R3, #1
6  ADD R4, R2, R3
7  BRzp #3
8  JSR JUDGE
9  ADD R4, R3, R4
10 BRnz JUDGE
11 HALT
12 JUDGE ADD R4, R0, #0
13 NOT R5, R2
14 ADD R5, R5, #1
15 ADD R4, R4, R5
16 BRzp #-2
17 ADD R4, R4, R2
18 BRnp #2
19 AND R1, R1, #0
20 HALT
21 ADD R2, R2, #1
22 AND R4, R4, #0
23 ADD R5, R2, #0
```

```

24 ADD R4, R4, R2
25 ADD R5, R5, #-1
26 BRp #-3
27 RET
28 .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

```

1  .ORIG x3000
2  ADD R1, R1, #1
3  ADD R3, R3, #-3
4  ADD R3, R3, R0
5  BRnz #10
6  AND R1, R1, #0
7  AND R3, R0, #1
8  BRz #7
9  ADD R1, R1, #1
10 ADD R2, R2, #3
11 NOT R3, R0
12 ADD R3, R3, #1
13 JSR JUDGE
14 ADD R4, R3, R4
15 BRnz JUDGE
16 HALT
17 JUDGE ADD R4, R0, #0
18 NOT R5, R2
19 ADD R5, R5, #1
20 ADD R4, R4, R5
21 BRzp #-2
22 ADD R4, R4, R2
23 BRnp #2
24 AND R1, R1, #0
25 HALT
26 ADD R2, R2, #2
27 AND R4, R4, #0
28 ADD R5, R2, #0
29 ADD R4, R4, R2
30 ADD R5, R5, #-1
31 BRp #-3
32 RET
33 .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

```
1  .ORIG x3000
2  LD R2,PTR
3  ADD R2, R0, R2
4  LDR R1, R2, #0
5  HALT
6  PTR .FILL #12293
7  .FILL #1
8  .FILL #1
9  .....
10 .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