

ICS-lab2 实验报告

孔浩宇 PB20000113

2022 年 12 月 11 日

目录

1	实验目的	2
2	实验原理	2
2.1	数列递推	2
2.2	二进制数取余	2
2.2.1	$R_j \% q$	2
2.2.2	$R_i \% p$	2
2.3	循环终止条件	2
3	实验步骤	3
3.1	初始化	3
3.2	循环	3
3.3	储存结果	3
3.4	结束	4
3.5	代码	4
4	实验结果	5
5	实验改进	5

1 实验目的

使用 LC-3 汇编语言求类斐波那契数列第 N 项 $F(N)$. 其中

$$F(0) = F(1) = 1$$

$$F(N) = F(N-2)\%p + F(N-1)\%q \quad (2 \leq N \leq 1024)$$

$$p = 2^k \quad (2 \leq k \leq 10), \quad 10 \leq q \leq 1024$$

2 实验原理

2.1 数列递推

在计算 $F(N)$ 时, 不妨用 R_i 保存 $F(N-2)$, R_j 保存 $F(N-1)$, 令

$$R_m = F(N-2)\%p, \quad R_n = F(N-1)\%q$$

则

$$F(N) = R_m + R_n.$$

此时令

$$R_i = R_j = F(N-1), \quad R_j = F(N).$$

2.2 二进制数取余

2.2.1 $R_j\%q$

利用以下算法

(1) 令 $R_j = R_j - q$, 若 $R_j < 0$, 转 (2), 否则转 (1).

(2) 令 $R_j = R_j + q$, 此时 R_j 即为所求, 算法结束.

设 $R_j = J$ 不难看出在求 $R_n\%q$ 的过程中, 时间花费为 $1 + J/q$.

2.2.2 $R_i\%p$

设 $R_i = I[15:0]$, 由 $p = 2^k$ 可得

$$R_i\%p = I[k-2:0], \quad \text{即 } R_i\%p = I\&(p-1)$$

2.3 循环终止条件

利用寄存器 R_k 来判断当前 R_j 存储的数列下标. 初始状态 $R_k = N$, $R_j = F(0) = 1$, $R_i = 0$.

(0) $R_k = R_k - 1$, 若为负, 则转 (5)

(1) 进行一次递推算法

(2) 转 (0)

(3) 循环终止, 输出 R_j 即为所求 $F(N)$.

3 实验步骤

3.1 初始化

(0) 标号

```
P      .FILL x3100      ;p
Q      .FILL x3101      ;q
N      .FILL x3102      ;N
S      .FILL x3103      ;F(N)
```

(1) 读入 p, q, N

```
.ORIG    x3000
        LDI R0, P        ; R0=P
        LDI R1, Q        ; R1=Q
        LDI R2, N        ; R2=N
```

(2) 初始化其他变量

```
ADD R0, R0, #-1        ; R AND R0 = R % P
ADD R3, R3, #1         ; R3 <= F(N)
AND R4, R4, #0         ; R4 <= F(N-1)
NOT R7, R1
ADD R7, R7, #1         ; R + R7 = R - Q
```

3.2 循环

```
AGAIN  ADD R2, R2, #-1    ;
        BRn  END          ; now the program end
        ADD R5, R4, #0    ; R5 = R4 - F(N-2)
        ADD R6, R3, #0    ; R4 = R3 - F(N-1)
        AND R5, R5, R0    ; R5 = F(N-2) % P
BE      ADD R6, R6, R7    ; R6 = R6 - Q
        BRzp BE          ; Re
        ADD R6, R6, R1    ; R6 = F(N-1) % Q
        ADD R4, R3, #0    ; R4 = F(N-1)
        ADD R3, R5, R6    ; R3 = R4 + R5 - F(N)
        BRnzp AGAIN      ; jump to AGAIN
```

3.3 储存结果

```
END      STI R3, S        ; store result
```

3.4 结束

HALT.

3.5 代码

```
.ORIG    x3000
    LDI R0, P           ; R0=P
    LDI R1, Q           ; R1=Q
    LDI R2, N           ; R2=N
    ADD R0, R0, #-1     ; now R & R0 = R mod P
    ADD R3, R3, #1      ; R3 is the result F(N)
    AND R4, R4, #0      ; R4 is F(N-1)
    NOT R7, R1
    ADD R7, R7, #1      ; now R + R7 = R - Q
    ; Initial

AGAIN  ADD R2, R2, #-1   ;
    BRn END             ; now the program end
    ADD R5, R4, #0      ; R5 = R4 F(N-2)
    ADD R6, R3, #0      ; R6 = R3 F(N-1)
    AND R5, R5, R0      ; R5 = F(N-2) % P
BE     ADD R6, R6, R7    ; R6 = R6 - Q
    BRzp BE             ; Re
    ADD R6, R6, R1      ; R6 = F(N-1) % Q
    ADD R4, R3, #0      ; R4 = F(N-1)
    ADD R3, R5, R6      ; R3 = R4 + R5 F(N)
    BRnzp AGAIN        ; jump to AGAIN
END    STI R3, S         ; store result
    HALT

P      .FILL x3100
Q      .FILL x3101
N      .FILL x3102
S      .FILL x3103
.END
```

4 实验结果

汇编评测

3 / 3 个通过测试用例

- 平均指令数: 2459.6666666666665
- 通过 256:123:100, 指令数: 1293, 输出: 146
- 通过 512:456:200, 指令数: 2407, 输出: 818
- 通过 1024:789:300, 指令数: 3679, 输出: 1219

5 实验改进

将 $F(N-2)$ 与 $F(N-2)\%p$ 的存储寄存器改为一个, 即 $R_4 \leftarrow R_4\%q = F(N-2)\%q$, 先利用 R_5 存储 $F(N-1)$, 即 $R_5 \leftarrow R_3$, 再将 $F(N-1)$ 与 $F(N-1)\%p$ 的存储寄存器改为一个, 即 $R_3 \leftarrow R_3\%p = F(N-1)\%q$, 之后令 $R_3 \leftarrow R_3 + R_4 = F(N)$, 再令 $R_4 \leftarrow R_5 = F(N-1)$, 一样可以完成循环, 且节省一个寄存器.

利用节省下的寄存器改进 $R_j\%q$ 的循环过程, 令 $R_6 = -2q$, 改进算法为

(1) 令 $R_j = R_j - 2q$, 若 $R_j < 0$, 转 (2), 否则转 (1).

(2) 令 $R_j = R_j + q$, 若 $R_j \geq 0$, 转 (3), 否则转 (2).

(3) 此时 R_j 即为所求, 算法结束.

设 $R_j = J$, 在改进后的求 $R_n\%q$ 的过程中, 时间花费至多为 $2 + J/2q$, 当 J 较大时可大幅减少花费指令数.