Lab2 Report

PB21111726 杨晓晨

实验目的 利用汇编码编程，计算以下类斐波拉契数列:

$$F[0] = F[1] = 1$$

$$F[N] = F[N-2]\%p + F[N-1]\%q$$

$$p = 2^k (2 \leqslant k \geqslant 10), 10 \leqslant q \geqslant 1024$$

并将给定 N 对应的 $F[N]$ 值存入地址 $x3103$, 其中 p 存在 $x3100$,q 存在 $x3101$,N 存在 $x3102$

实验原理 我们知道 $p = 2^k$, 写成二进制形式即 $\underbrace{0...0}_{(16-k-1)\ 0}\ 1\underbrace{0...0}_{k\ 0}$, 所以

$$F[N-2]\%p = F[N-2] \quad AND \quad (p-1)$$

从而简化了运算

对于 $10 \leqslant q \geqslant 1024$, 我们可以用简单的累减法循环运算，当得到负数时停止循环并加上 q 得到余数 $F[N-1]\%q$, 随后便可得

$$F[N] = F[N-2]\%p + F[n-1]\%q (N >= 2)$$

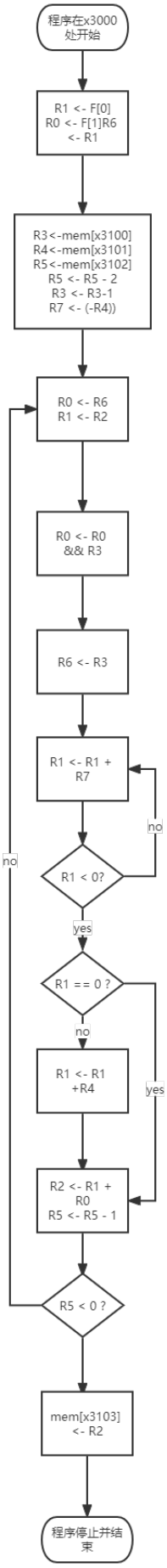
进行 (N-2) 次循环便可得到特定 N 对应的 $F[N]$

实验步骤 利用寄存器 $R0$ 存储 $F[N-2]$, 并将 $F[N-2]\%p$ 的结果同样存储在 $R0$ 中，利用寄存器 $R1$ 存储 $F[N-1]$, 并将 $F[N-1]\%q$ 的结果存储在 $R1$ 中。

为了让 $R0$ 得到 $F[N-2]$ 的值，我们要引入寄存器 $R6$ 存储 $F[N-1]$, 并在下一循环中将 $R6$ 赋值给 $R0$

对于 p, q, N, 引入寄存器 $R3, R4, R5$ 分别存储, 后续过程中我们要用到 $p-1, -q, N-2$, 因而用 $R3$ 存储 $p-1$, 用 $R7$ 存储 $-q$, 用 $R5$ 继续存储 $N-2$

该实验的步骤流程图如下:


程序在x3000
处开始
R1 <- F[0]
R0 <- F[1]R6
<- R1
R3<-mem[x3100]
R4<-mem[x3101]
R5<-mem[x3102]
R5 <- R5 - 2
R3 <- R3-1
R7 <- (-R4))
R0 <- R6
R1 <- R2
R0 <- R0
&& R3
R6 <- R3
R1 <- R1 +
R7
no
R1 < 0?
yes
R1 == 0 ?
no
R1 <- R1
+R4
yes
R2 <- R1 +
R0
R5 <- R5 - 1
R5 < 0 ?
no
mem[x3103]
<- R2
程序停止并结
束

可以分为三个部分: 数据的初始化，斐波拉契循环，存储结果
数据初始化部分的代码如下:

$$
\begin{array}{lll}
ADDALL & ADD & R6, \quad R6, \quad 1; F(0), store \quad F[N-2] \\
 & ADD & R1, \quad R1, \quad 1 \quad ; F(0), store \quad F[N-1]\%q \\
 & ADD & R2, \quad R2, \quad 1 \quad ; F(1), store \quad F[N] \\
LOAD & LDI & R3, \quad P; initially \quad store \quad value \quad p, \quad then \quad store \quad value \quad p-1 \\
 & LDI & R4, \quad Q; store \quad value \quad q \\
 & LDI & R5, \quad N; store \quad value \quad N \\
 & ADD & R5, \quad R5, -2 \\
 & ADD & R3, \quad R3, -1; \quad R3 < -R3 - 1 \\
 & NOT & R7, \quad R4; \quad R7 < -not \quad R4 \\
 & ADD & R7, \quad R7, \quad 1; R7 < -not \quad R4 + 1//store \quad value \quad -q
\end{array}
$$

斐波拉契循环部分的代码如下:

$$
\begin{array}{lll}
 & loop & \\
 & ADD & R0, \quad R6, \quad 0; \quad R0 < -F(N-2) \\
 & ADD & R1, \quad R2, \quad 0; \quad R1 < -F(N-1) \\
 & & \\
 & modp \quad AND & R0, \quad R3, \quad R0 \\
 & & \\
 & ADD & R6, \quad R1, \quad 0 \\
 & modq \quad ADD & R1, \quad R1, \quad R7; R1 < -R1 + (-R4) \\
 & BRP & modq; \quad R1 \quad > 0, continue \quad the \quad loop \\
 & BRZ & add1; \quad R1 \quad = 0 \\
 & ADD & R1, \quad R1, \quad R4; \quad R1 < -R1 + (R4) \\
 & & \\
 & add1 \quad ADD & R2, \quad R1, \quad R0; \quad R2 < -R1 + R0 \\
 & ADD & R5, \quad R5, \quad -1 \\
 & BRZP & loop
\end{array}
$$

存储结果部分的代码如下:

$$STI \quad R2, \quad RESULT$$

汇编评测

8 / 8 个通过测试用例

- 平均指令数: 2807.625
- 通过 256:123:100, 指令数: 1282, 输出: 146
- 通过 512:456:200, 指令数: 2396, 输出: 818
- 通过 1024:789:300, 指令数: 3668, 输出: 1219
- 通过 512:301:230, 指令数: 2898, 输出: 438
- 通过 256:100:256, 指令数: 3476, 输出: 221
- 通过 64:50:70, 指令数: 845, 输出: 53
- 通过 128:56:332, 指令数: 4349, 输出: 114
- 通过 256:35:199, 指令数: 3547, 输出: 230

实验结果

8 个结果均正确，平均指令数为 2807.625, 可部分验证程序的正确性

对于极端数据 1024:10:1024, 评测结果如图

- 通过 1024:10:1024, 指令数: 106479, 输出: 351

问题与思考 1. 如何提高循环效率减少指令数:

一开始我将 $R3$ 中的值一直不变，导致寄存器不足, 要在循环中增加指令对 p, q 分别处理并用一个寄存器存下来，这就增加了大量指令处理，后来我发现 $R3$ 中的 p 并不是之后的循环中需要的值，从而预先求出 $p-1, -q$ 两个需要的量存入 $R3, R7$ 中，避免循环过程中的重复求值。

另外我一开始没有发现求 p 的余数的规律，对其和 q 做了同样的处理方式，从而导致循环中指令量大大增加，而发现了该规律之后就可以大大简化计算过程减少执行指令数

Something Interesting

Here are some questions worth thinking:

> **You don't have to answer them in your report!**
>
> **You can answer in your report, but that will bring you no more extra points.**
>
> **Don't worry!**

You may find that this Fibonacci sequence has periodicity sometimes. For example, assume (p,q) are (32, 16), (64, 32), or (32, 64) respectively:

1. Can you make a conclusion of the least positive period of these sequences?
2. Can your conclusion apply to all integer p that $4 \leq p \leq 1024$?
   If yes, prove that. If no, give a counterexample.

数列周期为 $p+q$