ICS Lab Report - lab2

SCGY Cao Gaoxiang PB20000061

December 9, 2021

Lab Name

Lending Your Name

Lab Purpose

Store the n-th number of a sequence into register R7.

Lab Content

In mathematics, the Fibonacci numbers, commonly denoted F_n, form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is $F(0) = 0$, $F(1) = 1$, $F(n) = F(n-1) + F(n-2)$.
As the Fibonacci sequence grows quite rapidly, we made a little difference with the origin Fibonacci sequence:

$$\begin{aligned}
&F(0) = 1\\
&F(1) = 1\\
&F(2) = 2\\
&F(n) = (F(n-1) + 2 * F(n-3)) \mod 1024 \ (1 <= n <= 16384)
\end{aligned}$$

In the beginning, n is stored in R0. All other registers' initial values are 0.
Your tasks are:

- Store $F(n)$ in register R7.
- Divide your student number into four equal segments, labelling them with a, b, c and d. For example, TA's student ID is PB17000144, so $a = 17$, $b = 0$, $c = 14$, $d = 44$. Store value of $F(a)$, $F(b)$, $F(c)$ and $F(d)$ at the end of your code with ".FILL" pseudo command.

Lab Environment

Windows 11 Home Edition version 21H2, Visual Studio Code, LC3Tools v2.0.2.

Lab Procedure

This time, it is only acquired to provide l-version of the program. To help understand the content of this lab, also help to verify the correctness of the program, a C program is firstly written as below.

```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char const *argv[]) {
    int n = 0;
    if (argc == 1) scanf("%d", &n);
    else sscanf(argv[1], "%d", &n);
    int a0, a1 = 2, a2 = 1, a3 = 1;
    if (n >= 3) for (int i = 3; i <= n; ++i) {
        a0 = (a1 + 2 * a3) % 1024;
        a3 = a2; a2 = a1; a1 = a0;
    }
    else if (n == 0) a0 = 1;
    else if (n == 1) a0 = 1;
    else if (n == 2) a0 = 2;
    printf("%d\n", a0);
    return 0;
}
```

However, the C program is only easy to read but not easy to translate to LC-3 Assembly Language. So after variable renaming and rearrange of statements, we get the code below which is easy to translate.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char const *argv[]) {
    int16_t R0 = 0;
    if (argc == 1) scanf("%hd", &R0);
    else sscanf(argv[1], "%hd", &R0);
    int16_t R7, R2 = 2, R3 = 1, R4 = 1;
    if (R0 == 0) R7 = 1;
    else if (R0 == 1) R7 = 1;
    else if (R0 == 2) R7 = 2;
    else for (int16_t i = 3; i <= R0; ++i) {
        R7 = (R2 + 2 * R4) % 1024;
        R4 = R3; R3 = R2; R2 = R7;
    }
    printf("%hd\n", R7);
    return 0;
}
```

Standing here, it seems that all we need to do is translating. But notice that decimal number 1023 cannot be stored using a 5-bit 2's complement number, we need to store it at another position of memory.
Another thing is to "lend my name". My student number is "PB20000061", which will be divided into $a = 20$, $b = 0$, $c = 0$, $d = 61$. After computing we know that $F(a) = 930$, $F(b) = 1$, $F(c) = 930$, $F(d) = 930$.
Finally we get the answer code below, whose length is 28, satisfying the standard of full marks.

```
.ORIG x3000
ADD R0, R0, #-1
BRz EXIT1
ADD R0, R0, #-1
BRz EXIT2
ADD R2, R2, #2
ADD R3, R3, #1
ADD R4, R4, #1
LD R5, MOD
LOOP ADD R0, R0, #-1
BRn OUTLOOP
ADD R6, R4, R4
ADD R7, R2, R6
AND R7, R7, R5
ADD R4, R3, #0
ADD R3, R2, #0
ADD R2, R7, #0
BRnzp LOOP
OUTLOOP BRnzp EXIT
EXIT1 ADD R7, R7, #1
BRnzp EXIT
EXIT2 ADD R7, R7, #2
BRnzp EXIT
MOD .FILL #1023
Fa .FILL #930
Fb .FILL #1
Fc .FILL #1
Fd .FILL #262
EXIT
.END
```

Initially, it was mistakenly considered that command HALT would change the value stored in R7. But that is only correct in the second edition of LC-3. Also, through rearranging the BR command to judge specificly, we get a shorter version of the answer whose length 26.

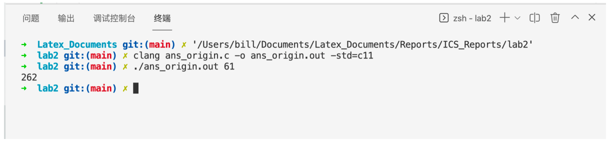
```
.ORIG x3000
ADD R0, R0, #-1
BRz EXIT2
ADD R0, R0, #-1
BRz EXIT1
ADD R2, R2, #2
ADD R3, R3, #1
ADD R4, R4, #1
LD R5, MOD
LOOP ADD R0, R0, #-1
BRn OUTLOOP
ADD R6, R4, R4
ADD R7, R2, R6
AND R7, R7, R5
ADD R4, R3, #0
ADD R3, R2, #0
ADD R2, R7, #0
BRnzp LOOP
OUTLOOP HALT
EXIT1 ADD R7, R7, #1
EXIT2 ADD R7, R7, #1
HALT
MOD .FILL #1023
Fa .FILL #930
Fb .FILL #1
Fc .FILL #1
Fd .FILL #262
.END
```

Correctness Verification

We use program written in LC-3 Assembly to calculate the answer of $F(n)$ and than compare the result with the one get from the C program above.

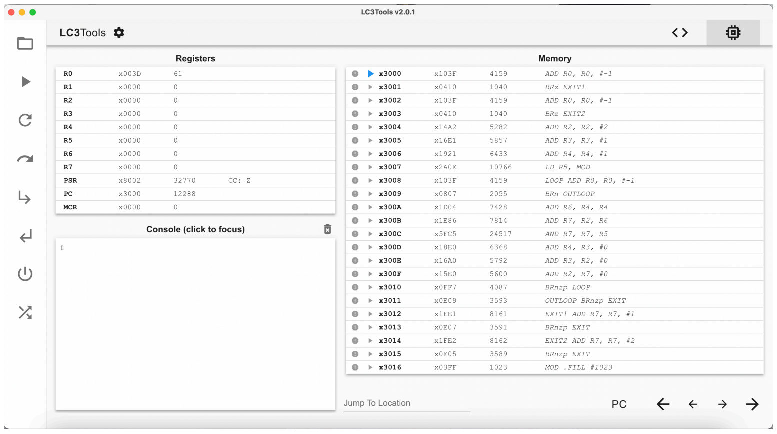
We have tested the integer n more than the following: 1, 2, 3, 20, 61, 16384. One of the cases in listed below as Figure 1.

Note: Due to some reasons, tests were run on macOS Monterey with LC3Tools v2.0.1.



```
➜  Latex_Documents git:(main) ✗ '/Users/bill/Documents/Latex_Documents/Reports/ICS_Reports/lab2'
➜  lab2 git:(main) ✗ clang ans_origin.c -o ans_origin.out -std=c11
➜  lab2 git:(main) ✗ ./ans_origin.out 61
262
➜  lab2 git:(main) ✗ █
```

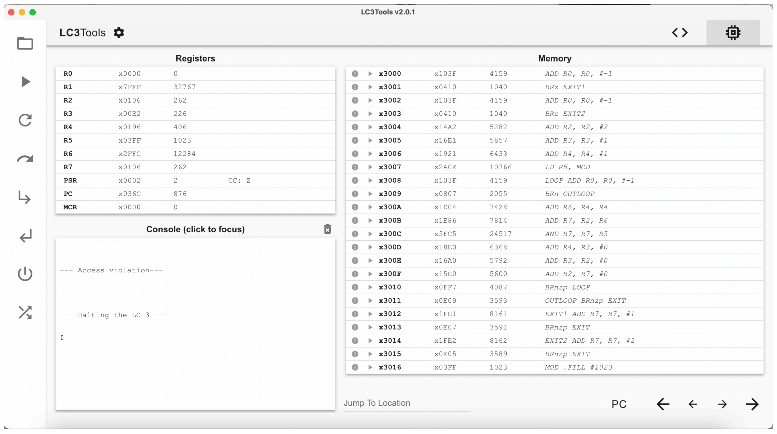
(a) The C Program



```
R0     x003D   61
R1     x0000   0
R2     x0000   0
R3     x0000   0
R4     x0000   0
R5     x0000   0
R6     x0000   0
R7     x0000   0
PSR    x8002   32770     CC: Z
PC     x3000   12288
MCR    x0000   0
```

```
x3000   x103F   4159    ADD R0, R0, #-1
x3001   x0410   1040    BRz EXIT1
x3002   x103F   4159    ADD R0, R0, #-1
x3003   x0410   1040    BRz EXIT2
x3004   x14A2   5282    ADD R2, R2, #2
x3005   x16E1   5857    ADD R3, R3, #1
x3006   x1921   6433    ADD R4, R4, #1
x3007   x2A0E   10766   LD R5, MOD
x3008   x103F   4159    LOOP ADD R0, R0, #-1
x3009   x0807   2055    BRn OUTLOOP
x300A   x1D04   7428    ADD R6, R4, R4
x300B   x1E86   7814    ADD R7, R2, R6
x300C   x5FC5   24517   AND R7, R7, R5
x300D   x18E0   6368    ADD R4, R3, #0
x300E   x16A0   5792    ADD R3, R2, #0
x300F   x15E0   5600    ADD R2, R7, #0
x3010   x0FF7   4087    BRnzp LOOP
x3011   x0E09   3593    OUTLOOP BRnzp EXIT
x3012   x1FE1   8161    EXIT1 ADD R7, R7, #1
x3013   x0E07   3591    BRnzp EXIT
x3014   x1FE2   8162    EXIT2 ADD R7, R7, #2
x3015   x0E05   3589    BRnzp EXIT
x3016   x03FF   1023    MOD .FILL #1023
```

(b) Before Running the LC3 Program



```
R0     x0000   0
R1     x7FFF   32767
R2     x0106   262
R3     x00E2   226
R4     x0196   406
R5     x03FF   1023
R6     x2FFC   12284
R7     x0106   262
PSR    x0002   2         CC: Z
PC     x036C   876
MCR    x0000   0
```

```
--- Access violation---


--- Halting the LC-3 ---
```

```
x3000   x103F   4159    ADD R0, R0, #-1
x3001   x0410   1040    BRz EXIT1
x3002   x103F   4159    ADD R0, R0, #-1
x3003   x0410   1040    BRz EXIT2
x3004   x14A2   5282    ADD R2, R2, #2
x3005   x16E1   5857    ADD R3, R3, #1
x3006   x1921   6433    ADD R4, R4, #1
x3007   x2A0E   10766   LD R5, MOD
x3008   x103F   4159    LOOP ADD R0, R0, #-1
x3009   x0807   2055    BRn OUTLOOP
x300A   x1D04   7428    ADD R6, R4, R4
x300B   x1E86   7814    ADD R7, R2, R6
x300C   x5FC5   24517   AND R7, R7, R5
x300D   x18E0   6368    ADD R4, R3, #0
x300E   x16A0   5792    ADD R3, R2, #0
x300F   x15E0   5600    ADD R2, R7, #0
x3010   x0FF7   4087    BRnzp LOOP
x3011   x0E09   3593    OUTLOOP BRnzp EXIT
x3012   x1FE1   8161    EXIT1 ADD R7, R7, #1
x3013   x0E07   3591    BRnzp EXIT
x3014   x1FE2   8162    EXIT2 ADD R7, R7, #2
x3015   x0E05   3589    BRnzp EXIT
x3016   x03FF   1023    MOD .FILL #1023
```

(c) After Running the LC3 Program

Figure 1: One Example of Correctness Verification