ICS Lab Report - lab4

SCGY Cao Gaoxiang PB20000061

December 24, 2021

Lab Name

Reveal Yourself

Lab Purpose

Complete the two tasks. Details are listed below.

Lab Content

Find out the missing bit in a program.

1. In the first part, 4 bits in the code are missing. You need to find out the missing bits. Before the program starts, values in the registers(except PC) are 0. At the end of the program, the register status is as follow.

$$R0 = 5, R1 = 0, R2 = 300f, R3 = 0$$
$$R4 = 0, R5 = 0, R6 = 0, R7 = 3003$$

   The program is stored in task1.txt. Store the fixed program in 'rec.txt'.

2. The second part of the program is used to calculate the remainder. 15 bits int the code are missing. For 2, 4, 6, 8 the remainder is easy to calculate, but for 7 the remainder needs some skills. Here is a quicker way to do the remainder for 7.

   Hint: The program uses the term "divide by 8".

   The program is stored in task2.txt. Store the fixed program in 'mod.txt'.

Lab Environment

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

Lab Procedure

1. For line 3, if x is 0, the program will come to line 4, which will halt the program. So it can only be 1;

   For line 14, if x is 0, the program will jump too many lines, which is obviously wrong.

   So there are only two lines uncertained, only four cases left, which is possible to try one by one. After enumeration, only when both of them are 0, the answer is the same as the task required.

   So we get the answer as "rec.txt" listed.

2. First, after translation and some simple ratiocination, we get the assembly code as follows.

```
1  LD R1 #21
2  JSR DIVIDE
3  AND R2 R1 #7
4  ADD R1 R2 R4
5  ADD R0 0xx #-7
6  BRp 1xxx11011
7  ADD R0 0xx #-7
8  BRn #1
9  ADD R1 R1 #-7
10 HALT
11 DIVIDE AND R2 R2 #0
12     AND R3 R3 #0
13     AND R4 R4 #0
14     ADD R2 R2 #1
15     ADD R3 R3 #8
16     AND R5 R3 R1
17     BRz #1
```

```
18      ADD R4 R2 R4
19      ADD R2 R2 R2
20      ADD xxx R3 R3
21      BR xxx #-6
22  RET
23  0000 0001 0010 0000
```

We begin analyzing the code from the function. It can be predicted that the author of the code uses R4 to store the result, and the dividend is stored in R1. The author uses R3 as a probe, moving it bit by bit. When all the bits of R1 are accessed, the loop ends, when R3 becomes 0.

As for the other part of the program, it can be predicted that R1 stores the number which we need to mod it by 7. Supposing that $R1 = 8x + m$, where x and m are integers, and we need to calculate m. We have the congruent equation as follows:

$$8x + m \equiv x + m \pmod 7$$

When the program come to line 4, $R1 = 8x + m$, $R4 = R1/8$, $R2 = R1 \bmod 8$. After being updated in line 4, $R1 = x + m$, which has the same remainder as before. We keep doing these until R1 is not more than 7. Then we need to check whether R1 equals to 7. The remainder will be stored in R1.

Finally, we get the assembly code below. After translation, we get the answer as "mod.txt" listed.

```
1   LD R1 #21
2   JSR DIVIDE
3   AND R2 R1 #7
4   ADD R1 R2 R4
5   ADD R0 R1 #-7
6   BRp #-5
7   ADD R0 R1 #-7
8   BRn #1
9   ADD R1 R1 #-7
10  HALT
11  DIVIDE AND R2 R2 #0
12      AND R3 R3 #0
13      AND R4 R4 #0
14      ADD R2 R2 #1
15      ADD R3 R3 #8
16      AND R5 R3 R1
17      BRz #1
18      ADD R4 R2 R4
19      ADD R2 R2 R2
20      ADD R3 R3 R3
21      BRnp #-6
22  RET
23  0000 0001 0010 0000
```

Correctness Verification

Both code were simulated in LC3Tools for correctness verification.