Lab 2 Report

In this lab, we need to input a string, and then found people whose name contain the string. We need to know how to use trap service routines. But the most important thing is figuring out the connection between address and data. That is, thinking out the operant you get is an address or an ascii value.

Arithmetic

The solution to the lab is easy. We just need to get one character in one time. Store them in continuous memory. Then we should traverse the address book which contain one contiguous block of memory for storing addresses and the other for storing the string (name and room number). To judge whether two characters are equal, we can subtract the value of two characters. If the result is 0, we can say the two characters are same. When the address book ends, we need to check whether we find a match to determine what to output which needs a block of memory to be used as counter.

Code

```
1. INPUT   GETC
2.         OUT
3.         STR         R0, R1, #0
4.         ADD         R1, R1, #1
5.         ADD         R0, R0, #-10    ;回车标志输入结束
6.         BRnp        INPUT
7.         ADD         R1, R1, #-1
```

```
8.          STR         R0, R1, #0
```

These are the codes for inputting. Press “enter” to finish typing. Remember to assign 0 to the end of the string.

```
1. ;check
2.          LD          R1, BOOK
3.          LDR         R1, R1, #0      ;载入 0 节点地址
4.          BRz         ENDCH           ;空 Book，结束
5. LOOP     ST          R1, NODE_R1     ;NODE_R1 暂时存储当前节点首地址
6. ------------------------------------------------------------
7. NEXT     LD          R1, NODE_R1
8.          LDR         R1, R1, #0
9.          BRz         ENDCH           ;下一节点为 0,链表结束
10.         BRnp        LOOP            ;下一节点循环
```

Because R1 also need to store the string address of the node under examination, we should set aside one location to store the node address. Since the address of next node is stored in the location of this node, we just need to load the data of R1 into R1 itself. If the data stored in the R1 equals to x0000, the address book ends and we need to jump to the end of the program.

```
1.          LDR         R1, R1, #2;
2.          LEA         R3, DATA
3. FCHAR    LDR         R4, R3, #0
4.          LDR         R2, R1, #0
5.          BRnp        CMP1            ;非零，进入比较
6.          ADD         R4, R4, #0
7.          BRnp        CMP1            ;非零，比较
8.          JSR         YES             ;两字符串为零，相同，输出
9.          BRnzp       NEXT            ;下一节点
10.
11.CMP1     NOT         R2, R2
12.         ADD         R2, R2, #1
13.         ADD         R2, R4, R2
14.         BRnp        LCHAR           ;非零则跳出到姓
```

```
15.         ADD         R1, R1, #1
16.         ADD         R3, R3, #1
17.         BRnzp       FCHAR           ;零再比较
```

Comparison for first name is similar to comparison for last name, so we only specify first name. If the value of the character is not 0, we need to compare the two characters. If the values are all 0, the two strings are same and we could come to next. If the result of subtraction is 0, we need to compare next characters in the two strings. If not, come to “last name” or next node.

```
1. ENDCH    LD          R0, NUM
2.          BRnp        ENDP
3.          LEA         R0, NOCH
4.          PUTS
5. ENDP     HALT
```

If the number of the match is 0, we need to print “`Not found`”. If the number is not 0, we could just halt the LC-3 .

The question of TA

How to replace LDI with LDR and LEA?

We can use one LEA and two LDR to replace one LDI. For example:

```
1.   LEA    R1, A ; R1<- MEM[MEM[A]]
```

This code equals to

```
1.      LEA       R1,A
2.      LDR      R1,R1,#0
3.      LDR      R1,R1,#0
```