# Lab 5 report

The difficulty of this Lab is when we can store the data in stack, and when we should load it. Also, we need to think carefully which data to be stored in stack. We have many kinds of data, so we need to figure out their function and when to change. Because registers of the LC-3 are few, some registers may store different data and we need to make sure that the data in the registers are what we want when we calculate.

**Arithmetic**

Firstly, we need to solve the problem of determining map boundaries. So I named two memory as X, Y which will show if we have reached the boundary. The first of all is (1,1). Then I have a variable which show how much recursion we have done. In order to find the maximum distance, I store the maximum distance of each place from the initial position in a corresponding memory, and finally traverse these memory blocks to find the maximum.

As for the whole Lab, we need to initialize data we need firstly. Then we need a loop to give all the positions a chance to be the initial position. Before the recursion, we need to store the data in stack. And to save time, we can check whether the corresponding distance data is not 0, if the data is not 0, it means that it is impossible to reach the maximum distance with this initial address. We could go to next loop straightly after we get the data ready.

In recursion, we should store R5 and R7 firstly. Then calculate the distance and store it. After that, we could judge if we can keep going in a certain direction. And remember to store the data in stack before JSR. At the end of DFS, we need restore the R5 and R7, and also reduce skiing distance.

**Code**

The loop parts.

1. LOOP        LD      R6, US\_SP       ;重新压栈
2. ADD     R6, R6, #-4
3. STR     R1, R6, #3
4. STR     R3, R6, #2
5. LD      R2, X
6. STR     R2, R6, #1
7. LD      R2, Y
8. STR     R2, R6, #0      ;压入地图地址，距离地址，X，Y
9. LDR     R2, R3, #0      ;是否已有滑道长度
10. BRp     NEXTLOOP ;已存入长度非零，直接下一循环
11. AND     R2, R2, #0
12. ST      R2, LEN         ;当前滑道长为 0
13. JSR     DFS
14. NEXTLOOP    LD      R2, RE\_P ;已遍历数目加1与地图规格比较
15. ADD     R2, R2, #1 ;RE\_P为已遍历地图初始位置
16. ST      R2, RE\_P
17. LD      R4, SIZE
18. ADD     R4, R4, R2      ;结果为0，到达地图末尾
19. BRz     TERMINATE       ;进入结束准备阶段
20. LD      R2, X
21. ADD     R2, R2, #1
22. ST      R2, X
23. LD      R4, M ;X与M比较
24. ADD     R4, R4, R2          ;是否达到了行末
25. BRnz    C ;未到则跳过步骤
26. AND     R2, R2, #0 ;到达行末需重新赋值XY
27. ADD     R2, R2, #1
28. ST      R2, X               ;换行X重新赋值1
29. LD      R2, Y
30. ADD     R2, R2, #1
31. ST      R2, Y
32. C           LDR     R3, R5, #-2         ;取出地图地址，长度地址，+1
33. ADD     R3, R3, #1
34. LDR     R1, R5, #-1
35. ADD     R1, R1, #1
36. BR      LOOP

The recursion parts (only part of the direction judgment)

1. RIGHT       LDR     R2, R5, #2
2. LD      R4, M
3. ADD     R4, R4, R2      ;是否边界
4. BRz     DOWN
5. LDR     R1, R5, #4      ;载入地图地址
6. LDR     R0, R1, #0      ;载入当前地址高度
7. ADD     R1, R1, #1      ;下一处地址
8. LDR     R4, R1, #0      ;下一处地址高度
9. NOT     R4, R4
10. ADD     R4, R4, #1      ;取相反数
11. ADD     R4, R4, R0      ;比较高度
12. BRnz    DOWN
13. ADD     R6, R6, #-4     ;申请内存
14. LDR     R3, R5, #3      ;读取当前长度地址
15. STR     R1, R6, #3      ;存入下一级地址
16. ADD     R3, R3, #1      ;长度地址+1
17. STR     R3, R6, #2      ;存入长度地址
18. ADD     R2, R2, #1      ;X+1
19. STR     R2, R6, #1      ;存X
20. LDR     R2, R5, #1
21. STR     R2, R6, #0      ;存Y
22. JSR     DFS

**Q&A**

The function of R5 and if R5 can be replaced by R6?

R5 is control link and it can be used to find other values in the same set of data. We use R5 only because it is changeless in the same set of data and we don’t need to calculate a lot. But we could also use R6 to find the data if we arrange the order of the program because R6 doesn’t change much.