深圳大学实验报告

课程名称: 计算机系统(1)

实验名称: <u>简单游戏设计</u>

学院:数学与统计学院

指导老师: 血航

报告人: 王曦 组号: 五

学号: 2021192010 实验地点: LC-3模拟器

实验时间: 2022年05月26日

提交时间: 2022年05月26日

1. 实验目的

四子棋是一款普遍流行的简易型桌面游戏,据说,虎克船长曾因专注于此游戏而长期隐身在住所,当船员们发现船长的这一专长之后,他们称这个游戏为"船长的情妇"。

四子棋是个双人游戏,两人轮流下棋,棋盘由行和列组成的网格,每个选手每次下一个子直到两人中有一人的棋子连成一条水平线、垂直线或者是对角线。

本实验需要在LC-3中实现简易版四子棋的游戏,两位选手通过键盘和输出窗口轮流交互操作,棋盘由6 X 6的网格组成。

游戏规则如下:

两位选手依次轮流落子;

选手不能悔棋;

有子的地方不能继续落子;

直到有一方的四个棋子能够连成一条水平线、垂直线或者是对角线;

如果棋盘已满,无人获胜,则平局。

游戏最初时应该打印空的棋盘,可以用ASCII码"-"(即ASCII 码 x002D)来表示该处为空,"O"(ASCII 码 x004F)表示第一位选手的棋子,"X" (ASCII 码 x0058)来表示第二位选手的棋子,为了让棋盘更易于观察,在各列间加一个空格,第6列之后不要添加,初始棋盘应该如下:

.

- - - - -

选手一始终先下第一步棋,然后两者轮流落子,在每次落子之后,应该打印该选手的信息,提示他落子,以选手一为例,应该打印信息如下:

Player 1, choose a column:

为了明确选手的落子的位置,该选手应该输入数字1-6,然后回车,数字1-6指示在落子所在的列,从左到右,无需输入行号,程序应默认从行号6到行号1递减的顺序填入该棋子,若前后输入的列号相同,则行号减一。例如,如果选手第一次在左起第二列落子,应该输入2,然后回车,则该棋子落在行6列2处,当后面输入的列号再次为2时,则将棋子落子行5列2处,以此类推,详情见后续示例输出。程序应该确保选手输入的数字对应正确的列的范围,如果输入不合理,应该输出一条错误信息,提示该选手继续输入,例如,如果对于选手一:

Player 1, choose a column: D

Invalid move. Try again.

Player 1, choose a column: 7

Invalid move. Try again.

Player 1, choose a column:

程序应该一直提示该选手,知道输入正确的数字,当用户输入完成,程序应通过显示回馈给选手,然后通过换行符 (ASCII 码 x000A)换行。

当选手输入成功后,程序应打印更新后的棋盘,并检查是否有人获胜,如果没人获胜,则轮到下一位输入。

当其中一位获胜或平局时,游戏结束,程序显示最后的棋盘情况并终止(Halt)。例如,如果选手二有四子相连,应该输出:

Player 2 Wins.

如果平局,程序应该输出:

Tie Game.

2. 实验内容

2.1 整体思路

开一个36个字的连续空间表示棋盘,用0表示无棋子,1表示选手一的棋子,-1表示选手二的棋子,这样可直接通过正负来判断棋子的类型.

开一个计数器TIMES,初始为1,用于判断轮到哪位选手下棋以及是否平局.偶数时轮到玩家一下棋,奇数时轮到玩家二下棋.TIMES>36时平局.

子程序DISPLAY用于打印棋盘.

子程序INPUT从键盘输入一个字符并判断其是否介于 $1\sim 6$ 间,若是则 R_0 返回对应位置,否则 R_0 返回0并输出提示信息. 子程序PLAY判断 R_0 对应的列是否还有位置,若有位置则根据棋子是哪一方的更新键盘.

子程序JUDGE_W检查当前是否有玩家获胜,若有则 R_0 返回1,否则 R_0 返回0.

子程序ANSWER输出获胜信息,根据TIMES的奇偶判断哪一方获胜.

2.2 Labels

```
1 ;Labels
   TIE .STRINGZ "Tie Game."
3 INVALID .STRINGZ "Invalid move. Try again. "
4 PROMPT1 .STRINGZ "Player 1, choose a column: "
5
   PROMPT2 .STRINGZ "Player 2, choose a column: "
  WIN1 .STRINGZ "Player 1 Wins."
6
   WIN2 .STRINGZ "Player 2 Wins."
8
9
   CHESS_O .FILL X004F
10 CHESS_X .FILL X0058
11 CHESS_N .FILL X002D
12
13 TIMES .FILL #1
   BOARD .BLKW 36 #0
14
15
16 SAVEO .FILL 0
17
   SAVE1 .FILL
18 SAVE2 .FILL 0
19 SAVE3 .FILL 0
20 SAVE4 .FILL 0
21 SAVE5 .FILL 0
22 SAVE6 .FILL
                 0
23 SAVE7 .FILL
```

```
24 SAVE77 .FILL 0
25
26 NUM_ROW .FILL #6
27 NUM_END .FILL #-36
28 NUM_CH .FILL #-48
29
30 SPACE .FILL X0020
31 ENDL .FILL X000A
```

2.2 子程序DISPLAY

先备份需要用到的寄存器原来的值(下略),再将棋盘首地址BORAD load到R6中.以 R_1 、 R_2 分别为行、列指针, R_3 用于接收地址 R_6 中存放的值,遍历棋盘,打印空位或对应的棋子,每打印一个棋子后 R_6 + +,当 R_6 非正时停止输出空格并输出换行,否则输出空格.棋盘打印结束后,将寄存器备份的值放回原寄存器(下略).

```
1 | ;print the board
2 DISPLAY ST R1, SAVE1 ; backup
    ST R2, SAVE2
3
4
    ST R3, SAVE3
5
    ST R6, SAVE6
6
7
    LEA R6, BOARD ; calculate the offset
8
     LD R2, NUM_ROW
9 COL_L LD R1, NUM_ROW
10 ROW_L LDR R3, R6, #0 ;traverse the board
   BRnp TYPE_L1 ;get the type of the chess
11
12
    LD RO, CHESS_N ; no chess
13
    BRnzp TYPR_LD
14
15 TYPE_L1 BRp TYPE_L2
16 LD RO, CHESS_X ;X chess
17
    BRnzp TYPR_LD
18 TYPE_L2 LD RO, CHESS_O ;O chess
19 TYPR_LD ST R7, SAVE7 ;backup
    TRAP X21 ;print the chess
20
21
    LD R7, SAVE7
22
    ADD R6, R6, #1 ;board address++
23
    ADD R1, R1, #-1
24
    BRNZ SPACE_L
    LD RO, SPACE
25
26
    ST R7, SAVE7
     TRAP X21 ;print the space
27
28
    LD R7, SAVE7
29
    ADD R1, R1, #0
30 SPACE_L BRp ROW_L
31
    LD RO, ENDL
    ST R7, SAVE7
32
33
    TRAP X21 ;print endl
     LD R7, SAVE7
34
35
    ADD R2, R2, #-1
36
     BRp COL_L
     LD R1, SAVE1
37
38
     LD R2, SAVE2
     LD R3, SAVE3
39
```

```
40 LD R6, SAVE6
41 RET
```

2.3 子程序PROMPT

用于输出等待用户输入时的、输入非法时的提示信息.根据计数器TIMES奇偶判断当前轮到哪一方输入.

```
1 ;print prompt info
 2
    PROMPT ST R1, SAVE1 ; backup
3
    ST R2, SAVE2
4
     ST RO, SAVEO
 5
 6
    ADD RO, RO, #0
 7
     BRp PRO_L3
8
    LEA RO INVALID ;invalid input
9
     ST R7 SAVE7
10
     TRAP X22
    LD RO ENDL
11
12
     TRAP X21
13
    LD R7 SAVE7
14
15 PRO_L3 ST RO, SAVEO
16
    LD RO, TIMES
17
    ST R7, SAVE7
18
    JSR JUDGE_N
    ADD R0, R0, #0
19
20
    BRZ PRO_L1
21
    LEA RO, PROMPT1
22
     BRnzp PRO_L2
23 PRO_L1 LEA RO, PROMPT2
24 PRO_L2 LD R7, SAVE7
    ST R7, SAVE7
25
26
    TRAP X22
    LD R7, SAVE7
27
28
    LD R1, SAVE1
     LD R2, SAVE2
29
30
     RET
```

其中判断奇偶的子程序JUDGE_N用数与1相与得到的结果来判断正负:

```
1  ;judge the number is whether odd or even
2  JUDGE_N ST R1, SAVE1
3  LD R1, TIMES
4  AND R0, R0, #0
5  ADD R0, R0, #1
6  AND R0, R0, R1
7  LD R1, SAVE1
8  RET
```

2.4 子程序INPUT

从键盘输入一个字符存放到 R_0 ,若 $R_0 \leq 0$ 或 $R_0 > 6$ 则为非法输入,返回 $R_0 = 0$,否则返回 R_0 .

```
INPUT
          ST R1 SAVE1 ;backup
2
     ST R7 SAVE7
3
     TRAP X23
4
     LD R7 SAVE7
5
     LD R1 NUM_CH ; R1=-48
     ADD RO, RO, R1 ;char->int
6
7
     BRNZ INP_L1 ; check whether R0 > 0
     AND R1, R1, #0
8
9
    ADD R1, R1, #-6
10
    ADD R1, R0, R1
11
     BRp INP_L1 ; check whether RO < 7
12
     BRnzp INP_L2
13 INP_L1 AND R0, R0, #0
14 INP_L2 LD R1 SAVE1
15
    RET
```

2.5 子程序PLAY

检查棋盘中用于输入的列有无空位,若有则 R_0 返回1,根据TIMES的奇偶判断是哪一方的棋子并更新棋盘;否则 R_0 返回0.

```
1; start the game
   PLAY ST R1, SAVE1 ; backup
3
    ST R2, SAVE2
4
    ST R3, SAVE3
5
    AND R3, R3, #0
    ADD R3, R3, #6 ;R3=6
6
7
     LEA R1, BOARD ; calculate the offset
8
     ADD R1, R1, #+15 ;R1 points at the begin of 6th line
9
     ADD R1, R1, #+14
    ADD R1, R1, R0
10
11
    LD RO TIMES
12 PLA_L4 LDR R2, R1, #0
13
    BRZ PLA_L3
    ADD R3, R3, #-1
14
    BRNZ PLA_LD
15
16
    ADD R1, R1, #-6
17
     BRnzp PLA_L4
18 PLA_L3 ST R7 SAVE7
19
    ST RO SAVEO
20
    JSR JUDGE_N
21
    LD R7, SAVE7
22
     ADD R0, R0, #0
23
     BRZ PLA_L2
24
     AND R2, R2, #0
    ADD R2, R2, #1
25
    STR R2, R1, #0
26
27
     BRnzp PLA_L1
28 PLA_L2 AND R2, R2, #0
    ADD R2, R2, #-1
29
30
    STR R2, R1, #0
31 PLA_L1 LD R1, TIMES
32
     ADD R1, R1, #1
```

```
33 LEA R2, TIMES
34
     STR R1, R2, #0
35
    LD RO SAVEO
36
    BRnzp PLA_L5
37 PLA_LD LEA RO INVALID
38
    AND RO, RO, #0
39
  PLA_L5 LD R1, SAVE1
40
    LD R2, SAVE2
41
    LD R3, SAVE3
42
     RET
```

2.6 子程序SUM_N

根据地址参数 R_6 和方向参数 R_0 求连成线的4个值之和,若和为 ± 4 ,则有玩家胜利,返回 $R_0=1$,否则返回 $R_0=0$.

用 R_3 记录连成线的四个值之和,初始为0.读取地址到 R_6 中, R_1 接收地址 R_6 的值, R_1 加到 R_3 上后, R_6 跳到下一位置,即 R_6 加上方向参数 R_0 .

方向参数 R_0 取值:① $R_0=1$ 时,求横向的和;② $R_0=5$,求斜向左下的和;③ $R_0=6$,求纵向的和;④ $R_0=7$,求斜向右下的和.

```
1 ;get sum
   SUM_N ST R1 SAVE1 ;backup
3
    ST R2 SAVE2
4
    ST R3 SAVE3
5
    ST R6 SAVE6
    AND R3, R3, #0 ;sum
6
7
    AND R2, R2, #0
8
    ADD R2, R2, #4
9
  SUM_L1 LDR R1, R6, #0 ;R1=M[R6]
10
    ADD R3, R1, R3
11
    ADD R6, R6, R0 ; next row(R0=6) or next column(R0=1)
12
     ADD R2, R2, #-1
13
    BRp SUM_L1
     AND R1, R1, #0
14
15
     ADD R1, R1, #4
     AND RO, RO, #0
16
17
     ADD R1, R1, R3
18
    BRnp SUM_L2
    ADD R0, R0, #1
19
20
     BRnzp SUM_LD
21 SUM_L2 AND R1, R1, #0
    ADD R1, R1, #-4
22
23
    ADD R1, R1, R3
24
    BRnp SUM_LD
25
     ADD RO, RO, #1
    BRnzp SUM_LD ;if the total is 4 or -4 , return positive number(RO)
26
   SUM_LD LD R1 SAVE1 ;else return zero(R0)
27
28
    LD R2 SAVE2
29
     LD R3 SAVE3
30
     LD R6 SAVE6
     RET
31
```

2.7 子程序JUDGE_W

判断是否有玩家获胜.根据方向参数,依次判断横向、纵向、斜向左下、斜向右下有无四个棋子连成线.

```
1 | ; judge win
2
    JUDGE_W ST R1 SAVE1 ;backup
3
      ST R2 SAVE2
4
      ST R3 SAVE3
5
     ST R4 SAVE4
 6
      ST R5 SAVE5
7
      ST R6 SAVE6
8
      ;judge line
9
10
      LD R1 NUM_ROW ; first layer loop
11
      AND R2, R2, #0
12
      ADD R2, R2, #3 ;second layer loop
      LEA R6 BOARD ; board address
13
14
    WIN_L1 AND R0, R0, \#0
15
      ADD R0, R0, #1
16
      ST R7 SAVE7
17
      JSR SUM_N
18
      LD R7 SAVE7
19
      ADD R0, R0, #0
20
      BRp WIN_LW
21
      ADD R6, R6, #1
22
      ADD R2, R2, #-1
23
      BRp WIN_L1
24
      ADD R6, R6, #-3
25
      ADD R6, R6, #6
26
      ADD R2, R2, #3
27
      ADD R1, R1, #-1
28
      BRp WIN_L1
29
30
      ;judge column
31
      LD R1 NUM_ROW ; first layer loop
32
      AND R2, R2, #0
      ADD R2, R2, #3 ;second layer loop
33
34
      LEA R6 BOARD ; board address
   WIN_L2 AND R0, R0, #0
35
36
      ADD RO, RO, #6
37
      ST R7 SAVE7
38
      JSR SUM_N
39
      LD R7 SAVE7
      ADD R0, R0, #0
40
41
      BRp WIN_LW
42
      ADD R6, R6, #6
43
      ADD R2, R2, #-1
44
      BRp WIN_L2
      ADD R6, R6, #-9
45
46
      ADD R6, R6, #-9
47
      ADD R6, R6, #1
48
      ADD R2, R2, #3
49
      ADD R1, R1, #-1
50
      BRp WIN_L2
51
      ;judge down
52
53
      AND R1, R1, #0 ; first layer loop
54
      ADD R1, R1, #3
55
      LEA R6 BOARD ; board address
```

```
56 WIN_L4 AND R2, R2, #0
 57
       ADD R2, R2, R1 ;second layer loop
 58
     WIN_L3 AND R0, R0, #0
 59
       ADD R0, R0, #7
       ST R7 SAVE7
 60
 61
       JSR SUM_N
 62
       LD R7 SAVE7
 63
       ADD R0, R0, #0
 64
       BRp WIN_LW
 65
       ADD R6, R6, #1
 66
       ADD R2, R2, \#-1
 67
       BRp WIN_L3
 68
       AND R3, R3, #0
 69
       ADD R3, R3, R1
 70
       NOT R3, R3
 71
       ADD R3, R3, #1
 72
       ADD R6, R6, R3
 73
       ADD R6, R6, #6
 74
       ADD R1, R1, #-1
 75
       BRp WIN_L4
 76
 77
       ;judge up
 78
 79
       AND R1, R1, #0 ; first layer loop
       ADD R1, R1, #3
 80
 81
       LEA R6 BOARD
 82
       ADD R6, R6, #5 ;board address
 83
     WIN_L6 AND R2, R2, #0
 84
       ADD R2, R2, R1 ;second layer loop
 85
     WIN_L5 AND R0, R0, #0
       ADD RO, RO, #5
 86
 87
       ST R7 SAVE7
 88
       JSR SUM_N
 89
       LD R7 SAVE7
       ADD R0, R0, #0
 90
 91
       BRp WIN_LW
 92
       ADD R6, R6, #-1
 93
       ADD R2, R2, \#-1
 94
       BRp WIN_L5
       ADD R6, R6, R1
 95
 96
       ADD R6, R6, #6
 97
       ADD R1, R1, #-1
 98
       BRp WIN_L6
99
       AND R0, R0, #0
100
     WIN_LW LD R1, TIMES
101
       LD R2, NUM_END ; R2=-36
       ADD R2, R1, R2
102
103
       BRNZ WIN_LD
104
       ADD R0, R0, #1
105
     WIN_LD LD R1 SAVE1
106
       LD R2 SAVE2
107
       LD R3 SAVE3
108
       LD R4 SAVE4
109
       LD R5 SAVE5
110
       LD R6 SAVE6
111
       RET
```

2.8 总代码

```
.ORIG x3000
2
    LOOP1 LD RO TIMES ; RO=1
3
     JSR DISPLAY ;print the board
4
   REINPUT JSR PROMPT ;output prompt info
5
     JSR INPUT
6
     ADD R0, R0, #0
7
     BRZ REINPUT ; invalid input
8
      JSR PLAY ; begin the game
9
      ADD R0, R0, #0
10
     BRZ REINPUT
11
      JSR JUDGE_W ; check whether someone wins
12
     ADD R0, R0, #0
      BRp OVER ; game over
13
14
      BRnzp LOOP1 ; game continues
15
    OVER JSR DISPLAY ; print the board
16
      JSR ANSWER ;show the result
17
      HALT ; OHHHHHHH
18
19
   ;print the board
20 DISPLAY ST R1, SAVE1 ; backup
21
     ST R2, SAVE2
22
     ST R3, SAVE3
      ST R6, SAVE6
23
24
25
     LEA R6, BOARD ; calculate the offset
26
      LD R2, NUM_ROW
27 COL_L LD R1, NUM_ROW
28
    ROW_L LDR R3, R6, #0 ;traverse the board
29
     BRnp TYPE_L1 ; get the type of the chess
30
     LD RO, CHESS_N ; no chess
31
     BRnzp TYPR_LD
32
33
   TYPE_L1 BRp TYPE_L2
34
     LD RO, CHESS_X ;X chess
35
      BRnzp TYPR_LD
36
   TYPE_L2 LD RO, CHESS_O ;O chess
37
   TYPR_LD ST R7, SAVE7 ;backup
38
      TRAP X21 ;print the chess
39
      LD R7, SAVE7
40
      ADD R6, R6, #1 ;board address++
     ADD R1, R1, #-1
41
42
     BRNZ SPACE_L
43
      LD RO, SPACE
44
      ST R7, SAVE7
45
      TRAP X21 ; print the space
      LD R7, SAVE7
46
47
      ADD R1, R1, #0
48
   SPACE_L BRp ROW_L
49
      LD RO, ENDL
      ST R7, SAVE7
50
51
     TRAP X21 ;print endl
52
     LD R7, SAVE7
53
      ADD R2, R2, #-1
54
     BRp COL_L
55
      LD R1, SAVE1
```

```
56
     LD R2, SAVE2
 57
       LD R3, SAVE3
 58
       LD R6, SAVE6
 59
       RET
 60
 61
     ;print answer
 62
     ANSWER ST R1 SAVE1 ; backup
       ST R2 SAVE2
 63
 64
       LD R1, TIMES
 65
       LD R2, NUM_END ; R2=-36
 66
       ADD R2, R1, R2
       BRNZ ANS_L1
 67
 68
       LEA RO TIE ; tie game
 69
       BRnzp ANS_LD
 70
    ANS_L1 LD RO, TIMES ;backup
 71
       ST R7 SAVE77
 72
       JSR JUDGE_N
 73
       ADD R0, R0, #0
 74
       BRp ANS_L2
 75
       LEA RO, WIN1
 76
       BRnzp ANS_LD
 77
     ANS_L2 LEA RO, WIN2
     ANS_LD LD R7, SAVE77 ;backup
 78
 79
       ST R7, SAVE7
       TRAP X22
 80
 81
       LD R7, SAVE7
 82
       LD R1 SAVE1
 83
       LD R2 SAVE2
 84
       RET
 85
     ;print prompt info
 86
 87
     PROMPT ST R1, SAVE1 ; backup
       ST R2, SAVE2
 88
 89
       ST RO, SAVEO
 90
 91
       ADD R0, R0, #0
 92
       BRp PRO_L3
 93
       LEA RO INVALID ; invalid input
 94
       ST R7 SAVE7
 95
       TRAP X22
 96
       LD RO ENDL
 97
       TRAP X21
       LD R7 SAVE7
98
99
     PRO_L3 ST RO, SAVEO
100
101
       LD RO, TIMES
102
       ST R7, SAVE7
103
       JSR JUDGE_N
104
       ADD R0, R0, #0
105
       BRZ PRO_L1
106
       LEA RO, PROMPT1
107
       BRnzp PRO_L2
108
     PRO_L1 LEA RO, PROMPT2
     PRO_L2 LD R7, SAVE7
109
       ST R7, SAVE7
110
111
       TRAP X22
```

```
112
     LD R7, SAVE7
113
      LD R1, SAVE1
114
      LD R2, SAVE2
115
      RET
116
    ;Labels
117
118 | TIE .STRINGZ "Tie Game."
     INVALID .STRINGZ "Invalid move. Try again. "
119
120
     PROMPT1 .STRINGZ "Player 1, choose a column: "
     PROMPT2 .STRINGZ "Player 2, choose a column: "
121
122
     WIN1 .STRINGZ "Player 1 Wins."
123
     WIN2 .STRINGZ "Player 2 Wins."
124
125
     CHESS_O .FILL X004F
126 CHESS_X .FILL X0058
127
    CHESS_N .FILL X002D
128
129
     TIMES .FILL #1
130
     BOARD .BLKW 36 #0
131
132
    SAVEO .FILL
133 SAVE1 .FILL
134
     SAVE2 .FILL
135 SAVE3 .FILL
136 SAVE4 .FILL
137
     SAVE5 .FILL
138 SAVE6 .FILL
139
     SAVE7 .FILL
140 SAVE77 .FILL 0
141
142 NUM_ROW .FILL #6
143 NUM_END .FILL #-36
     NUM_CH .FILL #-48
144
145
    SPACE .FILL X0020
146
147
     ENDL .FILL X000A
148
149 INPUT ST R1 SAVE1 ; backup
150
     ST R7 SAVE7
      TRAP X23
151
152
      LD R7 SAVE7
     LD R1 NUM_CH ; R1=-48
153
154
      ADD RO, RO, R1 ;char->int
155
      BRNZ INP_L1 ; check whether R0 > 0
      AND R1, R1, #0
156
157
      ADD R1, R1, #-6
      ADD R1, R0, R1
158
159
      BRp INP_L1 ; check whether R0 < 7
160
       BRnzp INP_L2
     INP_L1 AND R0, R0, #0
161
     INP_L2 LD R1 SAVE1
162
      RET
163
164
165 ;start the game
166
    PLAY ST R1, SAVE1 ;backup
      ST R2, SAVE2
167
      ST R3, SAVE3
168
169
      AND R3, R3, #0
```

```
170
       ADD R3, R3, #6 ;R3=6
171
       LEA R1, BOARD ; calculate the offset
       ADD R1, R1, #+15 ;R1 points at the begin of 6th line
172
173
       ADD R1, R1, \#+14
       ADD R1, R1, R0
174
175
       LD RO TIMES
176
     PLA_L4 LDR R2, R1, #0
177
       BRZ PLA_L3
178
       ADD R3, R3, #-1
179
       BRNZ PLA_LD
180
       ADD R1, R1, #-6
181
       BRnzp PLA_L4
182
     PLA_L3 ST R7 SAVE7
183
       ST R0 SAVE0
184
       JSR JUDGE_N
185
       LD R7, SAVE7
       ADD R0, R0, #0
186
187
       BRZ PLA_L2
188
       AND R2, R2, #0
189
       ADD R2, R2, #1
190
       STR R2, R1, #0
191
       BRnzp PLA_L1
192
     PLA_L2 AND R2, R2, #0
193
       ADD R2, R2, #-1
194
       STR R2, R1, #0
     PLA_L1 LD R1, TIMES
195
196
       ADD R1, R1, #1
197
       LEA R2, TIMES
198
       STR R1, R2, #0
199
       LD RO SAVEO
200
       BRnzp PLA_L5
     PLA_LD LEA RO INVALID
201
       AND R0, R0, #0
202
203
     PLA_L5 LD R1, SAVE1
       LD R2, SAVE2
204
205
       LD R3, SAVE3
206
       RET
207
208
     ; judge the number is whether odd or even
209
     JUDGE_N ST R1, SAVE1
       LD R1, TIMES
210
       AND RO, RO, #0
211
212
       ADD R0, R0, #1
213
       AND R0, R0, R1
214
       LD R1, SAVE1
215
       RET
216
     ;judge win
217
218
     JUDGE_W ST R1 SAVE1 ;backup
219
       ST R2 SAVE2
       ST R3 SAVE3
220
221
       ST R4 SAVE4
222
       ST R5 SAVE5
223
       ST R6 SAVE6
224
225
       ;judge line
       LD R1 NUM_ROW ; first layer loop
226
227
       AND R2, R2, #0
```

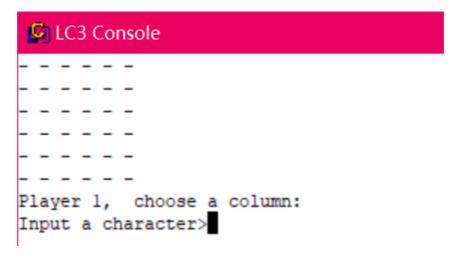
```
ADD R2, R2, #3 ;second layer loop
228
229
       LEA R6 BOARD ; board address
     WIN_L1 AND R0, R0, #0
230
231
       ADD R0, R0, #1
232
       ST R7 SAVE7
233
       JSR SUM_N
234
       LD R7 SAVE7
235
       ADD R0, R0, #0
236
       BRp WIN_LW
237
       ADD R6, R6, #1
238
       ADD R2, R2, \#-1
239
       BRp WIN_L1
240
       ADD R6, R6, #-3
       ADD R6, R6, #6
241
242
       ADD R2, R2, #3
243
       ADD R1, R1, #-1
244
       BRp WIN_L1
245
246
       ;judge column
247
       LD R1 NUM_ROW ; first layer loop
248
       AND R2, R2, #0
249
       ADD R2, R2, #3 ;second layer loop
250
       LEA R6 BOARD ; board address
251
     WIN_L2 AND R0, R0, #0
252
       ADD R0, R0, #6
       ST R7 SAVE7
253
254
       JSR SUM_N
255
       LD R7 SAVE7
256
       ADD R0, R0, #0
257
       BRp WIN_LW
       ADD R6, R6, #6
258
259
       ADD R2, R2, #-1
       BRp WIN_L2
260
261
       ADD R6, R6, #-9
       ADD R6, R6, #-9
262
263
       ADD R6, R6, #1
       ADD R2, R2, #3
264
265
       ADD R1, R1, #-1
266
       BRp WIN_L2
267
268
       ;judge down
       AND R1, R1, #0 ; first layer loop
269
270
       ADD R1, R1, #3
271
       LEA R6 BOARD ; board address
     WIN_L4 AND R2, R2, #0
272
       ADD R2, R2, R1 ;second layer loop
273
     WIN_L3 AND R0, R0, #0
274
275
       ADD R0, R0, #7
276
       ST R7 SAVE7
277
       JSR SUM_N
       LD R7 SAVE7
278
279
       ADD R0, R0, #0
280
       BRp WIN_LW
281
       ADD R6, R6, #1
282
       ADD R2, R2, #-1
283
       BRp WIN_L3
284
       AND R3, R3, #0
285
       ADD R3, R3, R1
```

```
286
       NOT R3, R3
287
       ADD R3, R3, #1
       ADD R6, R6, R3
288
289
       ADD R6, R6, #6
290
       ADD R1, R1, #-1
291
       BRp WIN_L4
292
293
294
       ;judge up
295
       AND R1, R1, #0 ; first layer loop
296
       ADD R1, R1, #3
297
       LEA R6 BOARD
298
       ADD R6, R6, #5 ;board address
299
     WIN_L6 AND R2, R2, #0
       ADD R2, R2, R1 ;second layer loop
300
301
     WIN_L5 AND R0, R0, #0
302
       ADD RO, RO, #5
303
       ST R7 SAVE7
304
       JSR SUM_N
305
       LD R7 SAVE7
306
       ADD R0, R0, #0
307
       BRp WIN_LW
308
       ADD R6, R6, #-1
309
       ADD R2, R2, #-1
310
       BRp WIN_L5
       ADD R6, R6, R1
311
312
       ADD R6, R6, #6
313
       ADD R1, R1, #-1
314
       BRp WIN_L6
315
       AND R0, R0, #0
    WIN_LW LD R1, TIMES
316
317
       LD R2, NUM_END ; R2=-36
       ADD R2, R1, R2
318
319
       BRNZ WIN_LD
       ADD R0, R0, #1
320
321 WIN_LD LD R1 SAVE1
322
       LD R2 SAVE2
323
       LD R3 SAVE3
324
       LD R4 SAVE4
325
       LD R5 SAVE5
326
       LD R6 SAVE6
327
       RET
328
329
     ;get sum
330
     SUM_N ST R1 SAVE1 ; backup
331
       ST R2 SAVE2
332
       ST R3 SAVE3
333
       ST R6 SAVE6
334
       AND R3, R3, #0
                       ;sum
335
       AND R2, R2, #0
336
       ADD R2, R2, #4
     SUM_L1 LDR R1, R6, \#0 ;R1=M[R6]
337
338
       ADD R3, R1, R3
339
       ADD R6, R6, R0
                      ;next row(R0=6) or next column(R0=1)
340
       ADD R2, R2, #-1
       BRp SUM_L1
341
342
       AND R1, R1, #0
343
       ADD R1, R1, #4
```

```
344
     AND RO, RO, #0
345
       ADD R1, R1, R3
346
       BRnp SUM_L2
347
     ADD R0, R0, #1
348
       BRnzp SUM_LD
349 SUM_L2 AND R1, R1, #0
350
       ADD R1, R1, #-4
       ADD R1, R1, R3
351
352
      BRnp SUM_LD
353
      ADD R0, R0, #1
354
       BRnzp SUM_LD ; if the total is 4 or -4 , return positive number(R0)
355 SUM_LD LD R1 SAVE1 ;else return zero(R0)
356
       LD R2 SAVE2
357
      LD R3 SAVE3
358
      LD R6 SAVE6
359
       RET
360
361
       . END
```

3. 实验结果

3.1 等待用户输入



3.2 非法输入

LC3 Console ------------Player 1, choose a column: Input a character>0 Invalid move. Try again. Player 1, choose a column: Input a character>

LC3 Console

LC3 Console

```
X - - - - -
0 - - - - -
0 X - - - -
Player 2, choose a column:
Input a character>1
- - - - -
X - - - - -
0 - - - - -
X - - - -
0 X - - - -
Player 1, choose a column:
Input a character>1
0 - - - - -
X - - - - -
Y - - - - -
Y - - - - -
Input a character>1
```

3.3 合法输入

LC3 Console

```
Player 1, choose a column:
Input a character>0
Invalid move. Try again.
Player 1, choose a column:
Input a character>7
Invalid move. Try again.
Player 1, choose a column:
Input a character>1
-----
-----
Player 2, choose a column:
Input a character>2
-----
O X ----
Player 1, choose a column:
Input a character>2
-----
Input a character>7
Input a character
```

3.4 平局

```
LC3 Console
хохохо
0 X 0 X 0 X
0 X 0 X 0 X
охохох
Player 1, choose a column:
Input a character>6
X O X O - O
хохохо
x \circ x \circ x \circ
0 X 0 X 0 X
0 X 0 X 0 X
охохох
Player 2, choose a column:
Input a character>5
x \circ x \circ x \circ
x \circ x \circ x \circ
хохохо
охохох
0 X 0 X 0 X
охохох
Tie Game.
A trap was executed with an illegal vector number.
```

---- Halting the processor -----

3.5 横向连线

```
LC3 Console
000---
Player 2, choose a column:
Input a character>3
X X X - - -
000---
Player 1, choose a column:
Input a character>4
X X X - - -
0000--
Player 1 Wins.
---- Halting the processor -----
```

3.6 纵向连线

```
🛂 LC3 Console
Player 2, choose a column:
Input a character>2
Player 1, choose a column:
Input a character>1
Player 1 Wins.
---- Halting the processor -----
```

3.7 斜向左下连线

```
LC3 Console
- - X - -
- - X X O -
0 X 0 X 0 0
Player 1, choose a column:
Input a character>5
- - - X O -
- - X X O -
Player 2, choose a column:
Input a character>4
- - - X - -
- - - X O -
- - X X O -
0 X 0 X 0 0
Player 2 Wins.
---- Halting the processor -----
```

3.8 斜向右下连线

```
LC3 Console
- X - - - -
- 0 - - - -
- O - X - -
- 0 X 0 X -
0 X 0 X 0 X
Player 1, choose a column:
Input a character>3
- X - - - -
- 0 - - - -
- 0 0 X - -
- 0 X 0 X -
0 X 0 X 0 X
Player 2, choose a column:
Input a character>3
- X - - - -
- 0 X - - -
- 0 0 X - -
- 0 X 0 X -
0 X 0 X 0 X
Player 2 Wins.
---- Halting the processor -----
```

4. 实验结论

- 1. 用子程序可以让代码思路更加清晰.
- 2. 进入子程序和从子程序返回主程序时要注意寄存器的备份.
- 3. 可通过一个数的二进制表示的末位来判断其奇偶性.