# 中国科学技术大学计算机学院
# 《计算机系统概论》实验报告


## LAB 04: The Game of Nim

姓名:王晨 学号:PE20060014

完成日期:2020 年 12 月 30 日

# 一、实验要求：

用LC-3汇编语言编写Nim游戏，具体规则见实验要求报告。

# 二、实验环境：

Mac OS

LC-3 Simulator Mac Version

# 三、算法思路：

由于本实验相对比较复杂，代码行数较多，这里将分模块阐述各个子程序，包括各个子程序所调用的寄存器和读取的内存说明。各个模块需要根据游戏流程中的各个步骤来进行编写，每个模块的子程序完成游戏中的某些功能或者步骤，具体游戏流程及步骤如下：

1. 初始化并打印棋盘
2. 检查棋盘中的Rock是否均被移出，是则游戏结束，打印胜利者信息，否则打印当前棋盘信息
3. 检查当前由哪个玩家进行操作
4. 要求当前玩家输入ROW和Rock Number
5. 检查ROW合法性，若不合法则跳转到步骤 3
6. 若Rock Number合法则更新棋盘，轮换玩家，跳转到步骤 2，否则跳转到步骤 3

步骤 2 由ENTRY和PROMPT的两个代码模块（具体代码见下一部分）实现，实现思想是在ENTRY中检查每一行中是否有o这个字符，也就是ROW A: 冒号后面的第一个字符是o还是结束符/0，如果任一行中有o，说明游戏未结束，通过PROMPT打印每一行的棋谱信息。否则游戏结束，打印胜利者的信息。

步骤 3，4分别由CHECKPLAYER和INPUT模块实现，R1用作标记当前玩家的标识寄存器，R1=0表示Player1，否则表示Player2。CHECKPLAYER检查R1寄存器是否为0，并在控制台输出相应的提示信息。INPUT通过调用TRAP x20 x21来获取键盘中的输入信息并回显到控制台，每次输入的单个字符都将保存在R0，并用ST指令保存到memory中的相应位置InputROW和InputROCKS。

步骤 5 由CHECKROW实现，方法是依次比较InputROW和ABC的ASCII码来确定是哪一行，然后跳转到INPUT_A, INPUT_B, INPUT_C更新棋谱，若InputROW不是ABC中的一个则用ERROR打印错误信息后跳转到步骤3。

步骤 6 由INPUT_A, INPUT_B, INPUT_C调用子程序UPDATE实现，首先在代码段后的数据域针对ROW A/B/C设置几个label，以A为例：

ROWA .STRINGZ "ROW A: ooo" 这是棋谱中A行的打印信息。

NEG_RocksA保存A行Rock数量的相反数，初始值为-3。

LAST_AddrA保存RowA字符串中结束符x0000的地址，初始值为[ROWA+10]。

用R4读取InputROCKS，这是要被移出的rock数量，检查其是否是小于当前行中rock数的正数，如果合法，以A为例，则更新NEG_RocksA和LAST_AddrA，并将LAST_AddrA位置的值由ASCII（o）写为x0000，这样在用PUTS打印时，就会打印到这个地方为止，完成了棋盘的更新。更新后将R1取反，表示轮换玩家，并跳转回步骤2。

# 四、程序代码及注释：

nim.asm

```
;PE20060014_LAB04 The Game of Nim
            .ORIG x3000
            AND R1,R1,#0          ; 0 Means P1, else Means P2, P1 always goes first
            LEA R0,ROWA
            ADD R0,R0,#10
            ST R0,LAST_AddrA      ;store the last memory address in LABEL ROWA into LAST_AddrA
            LEA R0,ROWB
            ADD R0,R0,#12
            ST R0,LAST_AddrB
            LEA R0,ROWC
            ADD R0,R0,#15
            ST R0,LAST_AddrC

ENTRY       LEA R0,ROWA
            LDR R0,R0,#7
            BRnp PROMPT
            LEA R0,ROWB
            LDR R0,R0,#7
            BRnp PROMPT
            LEA R0,ROWC
            LDR R0,R0,#7
            BRnp PROMPT
            BRz CHECKWINNER         ;if all the rocks in each ROW are removed,Go CHECKWINNER

PROMPT      LEA R0,ROWA
            PUTS
            JSR NEXTLINE
            LEA R0,ROWB
            PUTS
            JSR NEXTLINE
            LEA R0,ROWC
            PUTS
            JSR NEXTLINE

CHECKPLAYER ADD  R1,R1,#0           ;Check R1, if not 0, Go Player2
            BRnp PLAYER2

PLAYER1     LEA R0,P1
            PUTS
            BR  INPUT

PLAYER2     LEA R0,P2
            PUTS
            BR  INPUT

INPUT       GETC                    ;Read a single Char,and the ASCII Code is copied into R0
            OUT                     ;echo the Input, we can not use IN as we don't want a default prompt.
            ST R0,InputROW
            GETC
            OUT
            ST R0,InputROCKS
            JSR NEXTLINE

CHECKROW    LD R3,ASCII_A
            LD R4,InputROW
            ADD R3,R3,R4
            BRz INPUT_A
            LD R3,ASCII_B
            ADD R3,R3,R4
            BRz INPUT_B
            LD R3,ASCII_C
            ADD R3,R3,R4
            BRz INPUT_C
            BRnp ERROR              ;R4 failes to match ASCII A/B/C,then Go ERROR

```

```
66  INPUT_A     LD R2,NEG_RocksA
67              LD R3,LAST_AddrA
68              JSR UPDATE
69              ST R2,NEG_RocksA
70              ST R3,LAST_AddrA
71              NOT R1,R1               ;reverse R1 to change Player
72              JSR NEXTLINE
73              BR ENTRY
74
75  INPUT_B     LD R2,NEG_RocksB
76              LD R3,LAST_AddrB
77              JSR UPDATE
78              ST R2,NEG_RocksB
79              ST R3,LAST_AddrB
80              NOT R1,R1               ;reverse R1 to change Player
81              JSR NEXTLINE
82              BR ENTRY
83
84  INPUT_C     LD R2,NEG_RocksC
85              LD R3,LAST_AddrC
86              JSR UPDATE
87              ST R2,NEG_RocksC
88              ST R3,LAST_AddrC
89              NOT R1,R1               ;reverse R1 to change Player
90              JSR NEXTLINE
91              BR ENTRY
92
93  ERROR       LEA R0,INVALID
94              PUTS
95              JSR NEXTLINE
96              BR CHECKPLAYER
97
98  CHECKWINNER ADD R1,R1,#0
99              BRnp P2WINS             ;when all rocks are removed,the P1 wins if R1=0
100             LEA R0,P1WIN
101             PUTS
102             BR EXIT
103
104 P2WINS      LEA R0,P2WIN
105             PUTS
106 EXIT        HALT
107
108 UPDATE      LD R4,InputROCKS
109             ADD R4,R4,#-16          ;convert inputROCKS from ASCII into decimal number
110             ADD R4,R4,#-16
111             ADD R4,R4,#-16
112             BRnz ERROR              ;inputROCKS must be positive
113             ADD R2,R2,R4
114             BRp ERROR               ;inputROCKS must be less than that ROWA has
115             NOT R4,R4
116             ADD R4,R4,#1
117             ADD R3,R3,R4            ;update the last Rock location in the ROW
118             AND R5,R5,#0
119             STR R5,R3,#0            ;clear the word from 'o' to '\0' at the last Rock location
120             RET
121
122 NEXTLINE    ST R0,SAVER0
123             LD R0,NEWLINE
124             OUT
125             LD R0,SAVER0
126             RET
127 ;DATA FIELD
128 NEG_RocksA   .FILL xFFFD            ;Negative Rock Number in Row A, initialized as -3
129 NEG_RocksB   .FILL xFFFB
130 NEG_RocksC   .FILL xFFF8
131 LAST_AddrA   .BLKW #1
132 LAST_AddrB   .BLKW #1
133 LAST_AddrC   .BLKW #1
134 InputROW     .BLKW #1               ;stores the row the player chooses
135 InputROCKS   .BLKW #1               ;stores the rock number the player chooses
136 SAVER0       .BLKW #1
137 ASCII_A      .FILL xFFBF            ;Negative ASCII 'A' -65
138 ASCII_B      .FILL xFFBE            ;-66
139 ASCII_C      .FILL xFFBD            ;-67
140 NEWLINE      .FILL x000A            ;ASCII for '\n'
141 ROWA         .STRINGZ "ROW A: ooo"
142 ROWB         .STRINGZ "ROW B: ooooo"
143 ROWC         .STRINGZ "ROW C: oooooooo"
144 P1           .STRINGZ "Player 1, choose a row and number of rocks: "
145 P2           .STRINGZ "Player 2, choose a row and number of rocks: "
146 INVALID      .STRINGZ "Invalid move. Try again."
147 P1WIN        .STRINGZ "Player 1 Wins."
148 P2WIN        .STRINGZ "Player 2 Wins."
149             .END
```

# 五、调试和测试：

本次实验的测试严格按照Example.txt中的测试用例输入，在控制台看到的结果如下，与Example.txt完全一致：

**Console (click to focus)**

```
ROW A: ooo
ROW B: ooooo
ROW C: oooooooo
Player 1, choose a row and number of rocks: B2

ROW A: ooo
ROW B: ooo
ROW C: oooooooo
Player 2, choose a row and number of rocks: A1

ROW A: oo
ROW B: ooo
ROW C: oooooooo
Player 1, choose a row and number of rocks: C6

ROW A: oo
ROW B: ooo
ROW C: oo
```

```
Player 2, choose a row and number of rocks: G1
Invalid move. Try again.
Player 2, choose a row and number of rocks: B3

ROW A: oo
ROW B:
ROW C: oo
Player 1, choose a row and number of rocks: A3
Invalid move. Try again.
Player 1, choose a row and number of rocks: C2

ROW A: oo
ROW B:
ROW C:
Player 2, choose a row and number of rocks: A1

ROW A: o
ROW B:
```

```
ROW B:
ROW C:
Player 2, choose a row and number of rocks: A1

ROW A: o
ROW B:
ROW C:
Player 1, choose a row and number of rocks: A*
Invalid move. Try again.
Player 1, choose a row and number of rocks: &4
Invalid move. Try again.
Player 1, choose a row and number of rocks: A1

Player 2 Wins.

--- Halting the LC-3 ---

▮
```

# 七、实验心得：

本实验相对前几次实验的代码数和复杂度都大了一些，对于编写这类的简单小游戏，应该从游戏流程入手，分模块编写各个功能，这样可以简化程序编写的思路。在本实验的程序中，总共使用了2个Subroutine，分别是Update和Newline，Update用于计算更新后的棋盘（也就是每一行的字符串），Newline用于打印换行，这些都是需要被主程序段多次调用到的子程序，因此编写为Subroutine使得程序更加简洁。

注意到本程序对于全局寄存器的使用并不多，中间数值，例如输入的行号、Rock数量、当前某一行还存留的Rock数量，都保存在Memory中通过LD，ST进行存取了，这样的好处是每次的存取，修改都比较安全，不容易发生错误，例如被其他子程序覆盖掉这些中间数值。因此程序在最后只用了一个SaveR0保存现场，其他寄存器都不用保存，因此调用子程序时的开销很小。但是这样安全的代价是主程序运行时的速度相对较慢，有利有弊吧，为了减少Debug的工作量，这样的方法也是可取的。

在判断合法性时，可以看到对于输入的行号，会将其同ABC的ASCII码依次比较，如果都不相等，那就是非法输入。之后在比较输入行号后面的数字，将其减去x30后得到十进制数，再依次比较NEG_RocksA，NEG_RocksB，NEG_RocksC就可以得知输入的数字是否合法。

而移出某一行的Rock，是通过修改某一行ROW的字符串实现的，LAST_AddrA，LAST_AddrB，LAST_AddrC保存了每一行.STRINGZ的结束符的位置，由于用PUTS指令打印时会在x0000处结束，那么只需要在用Update更新棋盘时，更新结束符的位置即可，例如ROW C: ooooooooo移出3个o，只需要将LAST_AddrC-3的位置由’o’置为’\0’,变为ROW C: ooooo’\0’oo,就完成了棋盘的更新显示。