ICS experiment report-lab4 LC-3

算法(C)

给出用高级语言编写的算法

```
#include<stdio.h>
void print(int a,int b,int c){
    printf("ROW A:");
    for(int i=1;i<=a;i++){
        printf("o");
    }
    printf("\n");
    for(int i=1;i<=b;i++){
        printf("o");
    }
    printf("\n");
    for(int i=1;i<=c;i++){
        printf("o");
    }
    printf("\n");
}
int main(){
    int A=2,B=5,C=8;
    int turn=1;
    print(A,B,C);
    int choose;
    char c;
    while(A+B+C!=1){
        printf("The player %d should input:",turn);
        scanf("%c %d",&c,&choose);
        if(c=='A'){
            if(choose<=A){
                A-=choose;
                turn=(turn+3)%3+1;
            }
            else{
                printf("Error\n");
            }
        }
        else if(c=='B'){
            if(choose<=B){
                B-=choose;
                turn=(turn+3)%3+1;
            }
            else{
                printf("Error\n");
            }
        }
        else if(c=='C'){
            if(choose<=C){
                C-=choose;
                turn=(turn+3)%3+1;
            }
```

```
            else{
                printf("Error\n");
            }
        }
        else{
            printf("Error\n");
        }

    }
    printf("The player %d win\n",turn);
}
```

- `print`函数的三个参数分别是三排o的现在的个数
- 需要进行两个检查
  - 排号是合法的，必须是A,B,C之一
  - 选中的o数目是合法的，不能超过选中的那行o现存的数目
- 循环终止的条件是：只要最后三排的o数目之和为0则说明有人胜出

汇编语言程序设计

完整的源代码

```
.ORIG x3000
CHECK:LD R1,ROWA
LD R2,ROWB
LD R3,ROWC
ADD R4,R2,R1
ADD R4,R4,R3
NOT R4,R4
ADD R4,R4,#1
BRZ GAMESTOP
JSR PRINTROWA
JSR PRINTROWB
JSR PRINTROWC
LEA R0,PLAYER
PUTS
LD R0,CURRENTPALYER
OUT
LEA R0,CHOOSE
PUTS
GETC
OUT
ADD R1,R0,#0
GETC
OUT
ADD R2,R0,#0
LD R0,HUOCHE
OUT
OUT
LEA R0,A
LDR R0,R0,#0
NOT R0,R0
LD R3,ROWA
AND R0,R1,R0
BRZ JUDGEA
LEA R0,B
```

```
LDR R0,R0,#0
NOT R0,R0
LD R3,ROWB
AND R0,R1,R0
BRZ JUDGEB
LEA R0,C
LDR R0,R0,#0
NOT R0,R0
LD R3,ROWC
AND R0,R1,R0
BRZ JUDGEC
LEA R0,ERROR
PUTS
BRNZP CHECK

JUDGEA:LD R0,ZERO
ADD R3,R3,R0
ADD R4,R2,#0;R3,STORE HAVE O AND R2 STORE REQUIRE O,R1 STORE REQUEST
NOT R4,R4
ADD R4,R3,R4
ADD R4,R4,#1
BRZP RIGHT
LEA R0,ERROR
PUTS
BRNZP CHECK
RIGHT: ST R4,ROWA
LD R0,CURRENTPALYER
ADD R0,R0,#1
ST R0,CURRENTPALYER
BRNZP CHECK

JUDGEB:LD R0,ZERO
ADD R3,R3,R0
ADD R4,R2,#0;R3,STORE HAVE O AND R2 STORE REQUIRE O,R1 STORE REQUEST
NOT R4,R4
ADD R4,R3,R4
ADD R4,R4,#1
BRZP RIGHTB
LEA R0,ERROR
PUTS
BRNZP CHECK
RIGHTB: ST R4,ROWB
LD R0,CURRENTPALYER
ADD R0,R0,#1
ST R0,CURRENTPALYER
BRNZP CHECK


JUDGEC:LD R0,ZERO
ADD R3,R3,R0
ADD R4,R2,#0;R3,STORE HAVE O AND R2 STORE REQUIRE O,R1 STORE REQUEST
NOT R4,R4
ADD R4,R3,R4
ADD R4,R4,#1
BRZP RIGHTC
LEA R0,ERROR
PUTS
BRNZP CHECK
```

```
RIGHTC: ST R4,ROWC
LD R0,CURRENTPALYER
ADD R0,R0,#1
ADD R0,R0,#-3
ST R0,CURRENTPALYER
BRNZP CHECK


GAMESTOP:LD R0,CURRENTPALYER
OUT
LD R0 WIN
PUTS
HALT

PRINTROWA: ST R7,RETURNADDR1
LEA R0,hello
PUTS
LEA R0,A
PUTS
LD R2,ROWA
AND R3,R3,#0
CONA: NOT R4,R3
ADD R5,R2,R4
ADD R5,R5,#1
BRP PRINTOA
LD R0,HUOCHE
OUT
LD R7,RETURNADDR1
RET
PRINTOA: LD R0,O
OUT
ADD R3,R3,#1
BRNZP CONA

PRINTROWB:ST R7,RETURNADDR1
LEA R0,hello
PUTS
LEA R0,B
PUTS
LD R2,ROWB
AND R3,R3,#0
CONB: NOT R4,R3
ADD R5,R2,R4
ADD R5,R5,#1
BRP PRINTOB
LD R0,HUOCHE
OUT
LD R7,RETURNADDR1
RET
PRINTOB: LD R0,O
OUT
ADD R3,R3,#1
BRNZP CONB

PRINTROWC:ST R7,RETURNADDR1
LEA R0,hello
PUTS
LEA R0,C
```

```
PUTS
LD R2,ROWC
AND R3,R3,#0
CONC: NOT R4,R3
ADD R5,R2,R4
ADD R5,R5,#1
BRP PRINTOC
LD R0,HUOCHE
OUT
LD R7,RETURNADDR1
RET
PRINTOC: LD R0,O
OUT
ADD R3,R3,#1
BRNZP CONC

RETURNADDR1: .FILL #10
RETURNADDR2: .FILL #10
RETURNADDR3: .FILL #10
ROWA: .FILL #2
ROWB: .FILL #5
ROWC: .FILL #8
CURRENTPALYER: .STRINGZ "1\n"
O: .STRINGZ "o"
num1: .STRINGZ "1"
num2: .STRINGZ "2"
num3: .STRINGZ "3"
A: .STRINGZ "A:"
B: .STRINGZ "B:"
C: .STRINGZ "C:"
ZERO: .STRINGZ "0"
hello: .STRINGZ "ROW "
HUOCHE: .STRINGZ "\n"
PLAYER: .STRINGZ "player"
WIN: .STRINGZ "WIN\n"
CHOOSE: .STRINGZ " choose a row and number of rocks: "
ERROR: .STRINGZ "Invalid move. Try again.\n"
.END
```

源代码较长，现在分段介绍功能

主控程序

```
CHECK:LD R1,ROWA
LD R2,ROWB
LD R3,ROWC
ADD R4,R2,R1
ADD R4,R4,R3
NOT R4,R4
ADD R4,R4,#1
BRZ GAMESTOP ;IF ROWA+ROWB+ROWC==0,THEN LOOP END
JSR PRINTROWA
JSR PRINTROWB
JSR PRINTROWC
LEA R0,PLAYER
PUTS
LD R0,CURRENTPALYER
```

```
OUT
LEA R0,CHOOSE
PUTS
GETC
OUT
ADD R1,R0,#0
GETC
OUT
ADD R2,R0,#0
LD R0,HUOCHE
OUT
OUT
LEA R0,A
LDR R0,R0,#0
NOT R0,R0
LD R3,ROWA
AND R0,R1,R0
BRZ JUDGEA
LEA R0,B
LDR R0,R0,#0
NOT R0,R0
LD R3,ROWB
AND R0,R1,R0
BRZ JUDGEB
LEA R0,C
LDR R0,R0,#0
NOT R0,R0
LD R3,ROWC
AND R0,R1,R0
BRZ JUDGEC
LEA R0,ERROR
PUTS
BRNZP CHECK
```

实现的功能

- 初始化，这个初始的o数目存在内存中，先要加载到寄存器中
- 判断循环结束，按照三个row长度和是否为0判断，循环结束进入输出程序
- 打印图案，跳转到PRINTROWA,PRINTROWB,PRINTROWC三个子程序中
- 读入和判断，使用系统调用和JUDGEA,JUDGEB,JUDGEC三个子程序判断
- default情况，即输入不合法的处理，这个输出一个字符串ERROR

打印图案PRINTROWA

```
PRINTROWC:ST R7,RETURNADDR1
LEA R0,hello
PUTS
LEA R0,C
PUTS
LD R2,ROWC
AND R3,R3,#0
CONC: NOT R4,R3
ADD R5,R2,R4
ADD R5,R5,#1
BRP PRINTOC
LD R0,HUOCHE
OUT
```

```
LD R7,RETURNADDR1
RET
PRINTOC: LD R0,O
OUT
ADD R3,R3,#1
BRNZP CONC
```

按照输出的格式，输出ROW A/B/C：o(数目与当前数目相同)

conc是一个循环，输出o

注意这里采用了跳转-连接的保存现场的方式，由于系统调用也会存在这样的过程，因此需要保存R7的内容到内存中，这就是保存现场

判断输入是否合法(输入的0数目是否满足要求)

```
JUDGEC:LD R0,ZERO
ADD R3,R3,R0
ADD R4,R2,#0;R3,STORE HAVE O AND R2 STORE REQUIRE O,R1 STORE REQUEST
NOT R4,R4
ADD R4,R3,R4
ADD R4,R4,#1
BRZP RIGHTC
LEA R0,ERROR
PUTS
BRNZP CHECK
RIGHTC: ST R4,ROWC
LD R0,CURRENTPALYER
ADD R0,R0,#1
ADD R0,R0,#-3
ST R0,CURRENTPALYER
BRNZP CHECK
```

这里需要注意的是，从键盘读到的数据是字符形式，需要转换为数字形式进行比较

结束，输出赢家

```
GAMESTOP:LD R0,CURRENTPALYER
OUT
LEA R0 WIN
PUTS
HALT
```

输出字符串即可

结果测试

```
ROW A:oo
ROW B:ooooo
ROW C:oooooooo
player1 choose a row and number of rocks: D1

Invalid move. Try again.
ROW A:oo
```

```
ROW A:oo
ROW B:ooooo
ROW C:oooooooo
player1 choose a row and number of rocks: A2

ROW A:
ROW B:ooooo
ROW C:oooooooo
```

```
ROW A:
ROW B:ooooo
ROW C:oooooooo
player2 choose a row and number of rocks: B1

ROW A:
ROW B:oooo
ROW C:oooooooo
```

```
ROW B:o
ROW C:oooooooo
player3 choose a row and number of rocks: C8

ROW A:
ROW B:o
ROW C:
player1 choose a row and number of rocks: B1

2WIN
```