# 实验4实验报告

## 实验流程图

见diagram.pdf

## 程序代码

### Maze.asm

.ORIG x3000

LD R6,SPP

LD R4,SPR

AND R5,R5,#0

LD R1,STDATA

LDR R2,R1,#0

ADD R6,R6,#-1

STR R2,R6,#0

ADD R4,R4,#-1

STR R2,R4,#0

ADD R5,R5,R2

ADD R5,R5,R5

ADD R5,R5,R5;R5<-address

ADD R1,R1,#1

LDR R2,R1,#0

ADD R6,R6,#-1

STR R2,R6,#0

ADD R4,R4,#-1

STR R2,R4,#0

ADD R5,R5,R2

ADD R1,R1,R5

ADD R1,R1,#1

LDR R2,R1,#0

JSR FUNOR;mark the starting location

JSR MAZE

;OUTPUT

FAIL LEA R0,NO

PUTS

HALT

SUCCESS NOT R5,R4

ADD R5,R5,#1

LD R4,SPR

ADD R5,R4,R5;R5<-x7000-R4

ADD R4,R4,#-1

LD R1,ASCII

LOOP LDR R0,R4,#0

ADD R0,R0,R1

OUT

ADD R5,R5,#-1

ADD R4,R4,#-1

LDR R0,R4,#0

ADD R0,R1,R0

OUT

ADD R4,R4,#-1

ADD R5,R5,#-1

BRz STOP

LD R0,SPACE

OUT

BR LOOP

STOP HALT

MAZE

LDR R1,R6,#1

ADD R1,R1,R1

ADD R1,R1,R1

ADD R1,R1,R1;row\*8

LDR R2,R6,#0

ADD R1,R1,R2

LD R3,ADDNUM

ADD R1,R1,R3;R1<-the address of the current location

ADD R6,R6,#-1;push return address

STR R7,R6,#0;store return address

ADD R6,R6,#-1

STR R1,R6,#0;store the address of the current location

LDR R2,R1,#0;R2<-information about the current location

EXIT LD R0,NUM1

AND R0,R2,R0

BRp SUCCESS

LD R5,SaveR5

ADD R5,R5,#1

LD R3,COUNT

ADD R3,R5,R3

ST R5,SaveR5

BRzp FAIL

NORTH LD R0,NUM2

LDR R2,R1,#0

AND R0,R2,R0

BRz EAST

ADD R5,R1,#-8

LDR R2,R5,#0

BRn EAST

;store the new location to the stack

;R3<-row or column

LDR R3,R6,#3

ADD R3,R3,#-1;NORTH

ADD R6,R6,#-1

STR R3,R6,#0

ADD R4,R4,#-1

STR R3,R4,#0

LDR R3,R6,#3

ADD R6,R6,#-1

STR R3,R6,#0

ADD R4,R4,#-1

STR R3,R4,#0

JSR FUNOR

JSR MAZE

JSR BACK

;ADD R6,R6,#2

;ADD R4,R4,#2

EAST LD R0,NUM3

LDR R2,R1,#0

AND R0,R2,R0

BRz SOUTH

ADD R5,R1,#1

LDR R2,R5,#0

BRn SOUTH

;store the new location to the stack

LDR R3,R6,#3

ADD R6,R6,#-1

STR R3,R6,#0

ADD R4,R4,#-1

STR R3,R4,#0

LDR R3,R6,#3

ADD R3,R3,#1;EAST

ADD R6,R6,#-1

STR R3,R6,#0

ADD R4,R4,#-1

STR R3,R4,#0

JSR FUNOR

JSR MAZE

JSR BACK

; ADD R6,R6,#2

; ADD R4,R4,#2

SOUTH LD R0,NUM4

LDR R2,R1,#0

AND R0,R2,R0

BRz WEST

ADD R5,R1,#8

LDR R2,R5,#0

BRn WEST

;store the new location to the stack

LDR R3,R6,#3

ADD R3,R3,#1;SOUTH

ADD R6,R6,#-1

STR R3,R6,#0

ADD R4,R4,#-1

STR R3,R4,#0

LDR R3,R6,#3

ADD R6,R6,#-1

STR R3,R6,#0

ADD R4,R4,#-1

STR R3,R4,#0

JSR FUNOR

JSR MAZE

JSR BACK

; ADD R6,R6,#2

; ADD R4,R4,#2

WEST LD R0,NUM5

LDR R2,R1,#0

AND R0,R2,R0

BRz MAZEEND

ADD R5,R1,#-1

LDR R2,R5,#0

BRn MAZEEND

;store the new location to the stack

LDR R3,R6,#3

ADD R6,R6,#-1

STR R3,R6,#0

ADD R4,R4,#-1

STR R3,R4,#0

LDR R3,R6,#3

ADD R3,R3,#-1;WEST

ADD R6,R6,#-1

STR R3,R6,#0

ADD R4,R4,#-1

STR R3,R4,#0

JSR FUNOR

JSR MAZE

JSR BACK

; ADD R6,R6,#2

; ADD R4,R4,#2

MAZEEND ADD R6,R6,#1;pop local variable

LDR R7,R6,#0

ADD R6,R6,#1

RET

FUNOR LD R0,ORNUM

NOT R0,R0

LDR R2,R1,#0

NOT R2,R2

AND R2,R0,R2

NOT R2,R2

STR R2,R1,#0

RET

BACK ;pop two parameterss

ADD R6,R6,#2

ADD R4,R4,#2

;reload the information of the current location

LDR R1,R6,#0

LDR R2,R1,#0

LD R0,ANDNUM

AND R2,R0,R2

STR R2,R1,#0

RET

STDATA .FILL x4000;

SPP .FILL x6000;stack pointer of the procedure

SPR .FILL x7000;stack pointer of the result

NUM1 .FILL x10

NUM2 .FILL x8

NUM3 .FILL x4

NUM4 .FILL x2

NUM5 .FILL x1

ADDNUM .FILL x4002

ORNUM .FILL x8000

ANDNUM .FILL x7FFF

ASCII .FILL x30

SPACE .FILL x20

COUNT .FILL x800F

SaveR5 .BLKW 1

NO .STRINGZ "NO"

.END

## 程序运行结果

以下结果分别对应

01no exist

02no exist-copy

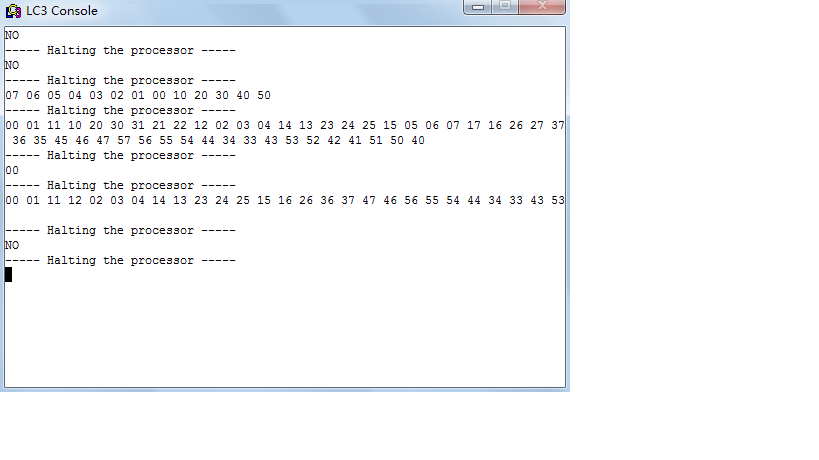
04loop

05no branch

07start

06branch

08no exist+loop



## 算法分析

本程序的基本思想即深搜回溯，对于当前位置，分别判断四个位置是否可走并且所到达的位置是否未到过，如果同时满足上述两个条件，则选择向这个方向扩展。当搜索表明向这个方向扩展是错误的时候，进行回溯操作，并考虑向别的方向扩展。当到达终点位置时，输出路径，并终止程序。

考虑到程序的输入数据规模较小，所以对于无路径的判断由以下想法实现：用一个计数器判断函数递归的次数，当次数达到一个比较大的值时，足以说明本图无解。此时输出无解，中断程序。

关于边界和终点的问题，我开始的理解和助教希望我理解的有出入。我虽然坚持认为按照本题的题意理解，边界判断是多余的，但是进行边界判断可以使程序应对更一般的情况。