**北京科技大学实验报告**

学院：计算机与通信工程学院 专业：计算机科学与技术 班级：计182

姓名：张兴泉 学号：41824071 实验日期：2020年12月7日

**实验名称：**创新设计——五子棋

**实验目的：**学习各接口芯片功能及应用特性，能够组合多个接口芯片，完成设计实验。

**实验要求：**根据已学习各接口芯片的应用特性，参考本讲义前述实验接线图及代码，自己设计完成综合设计实验。并通过现场验收后，完成本实验报告。

**坚决杜绝抄袭！**

**实验环境：**

操作系统：Windows 10

仿真软件：Proteus 8.6

硬件设备：CZ-CIUS型开放式微机接口实验系统

**实验内容：**

根据已学习各接口芯片的应用特性，参考本讲义前述实验接线图及代码，自己设计完成综合设计实验。要求所选多个芯片或模块的**加权值总和≥3**，权值列表见实验讲义的表3-1。并且，**所选芯片或模块中必须包含8259或8254芯片的其中一个作为功能模块部分**。

**特别要求：**

1、2020年创新实验不得设计**交通灯、电子琴/音乐盒、简单流水灯/跑马灯/流水式霓虹灯、抢答器、投票器**。

2、**至少要在实验箱上验证实现多模块创新设计**。

3、同时实现实验箱设计与Proteus仿真设计，获得相应分数，如果仅实现实验箱设计，则无法得到仿真设计部分的分数。

4、**不能是第二章验证实验中代码的简单叠加**，需要包含一定量自己编写的汇编代码，在验收时需说明自己所编写的代码量大概是多少。

5、代码逻辑具备一定的复杂度，可以折抵1分加权分，但在验收时，**需要提供程序流程图，以及说明自编代码行数。**

**实验结果与分析：**

1. **设计应用场景及实现功能**

- **游戏简介：**

五子棋是一种两人对弈的纯策略型[棋类](https://zh.wikipedia.org/wiki/%E6%A3%8B%E7%B1%BB" \o "棋类)游戏，现有一15×15的棋盘，棋手双方分别使用黑白两色的棋子，轮流下棋在棋盘直线与横线的交叉点上，先在横线、直线或斜对角线上形成5子连线者获胜，若超过指定时间仍未决出胜负，则平局。

**- 游戏规则：**

1.选择先手，通过键盘输入0代表白棋先下，输入1代表黑棋先下。

2.黑白棋手轮流下棋，棋手不能随意落子，落子位置受到以下条件约束：

* 落棋位置不能超出棋盘区域,即落子坐标x和y的取值范围为1-f。
* 若某处已有棋子，棋手不能将棋子下到该处。

如果棋手违背了以上两条落子规则，裁判会提示落子位置不合法“YOU CANNOT PUT HERE!”。

3.对局最长时间设置为999秒，使用电子时钟计时，如果超过999秒仍未有棋手胜出，裁判宣告平局“NO PLAYER HAS WIN!”。

4.如果在规定时间内，有棋手率先连成五子，则该棋手胜出，如果是执黑子的棋手胜出，裁判宣告“BLACK PLAYER HAS WIN!”，反之裁判宣告“WHITE PLAYER HAS WIN!”。

5.如果在规定时间内，若是执黑子的棋手放弃，则裁判宣告“BLACK PLAYER HAS WIN!”，反之裁判宣告“WHITE PLAYER HAS WIN!”

1. **设计思路**

**- 软件层面：**

数据结构：

定义CHESSBOARD变量作为棋盘缓冲区，存放棋盘中所有棋盘格的状态，假设棋盘格的坐标为（x,y）,则对应的棋盘格为CHESSBOARD[15\*x+y]。如果该棋盘格放的为黑子，则CHESSBOARD[15\*x+y]=1；如果是白子，则CHESSBOARD[15\*y+x]=2。

定义ERROR1、PUT、ORDER、WHITEWIN、BLACKWIN、PING、BLACKEXIT、WHITEEXIT变量，分别作为落子位置错误、请输入落子位置、请输入先手顺序、执白子的棋手赢、执黑子的棋手赢、平局、黑棋手放弃、白棋手放弃信息。

判断棋手获胜程序的实现思路：

1. 横向连成5个

CHESSBOARD[BX]、CHESSBOARD[BX+1]、CHESSBOARD[BX+2]、CHESSBOARD[BX+3]、CHESSBOARD[BX+4]同为1，黑棋手胜利，同为2，白棋手胜利。

1. 纵向连成5个

CHESSBOARD[BX]、CHESSBOARD[BX+15]、CHESSBOARD[BX+30]、CHESSBOARD[BX+45]、CHESSBOARD[BX+60]同为1，黑棋手胜利，同为2，白棋手胜利。

1. 斜上连成5个

CHESSBOARD[BX]、CHESSBOARD[BX-14]、CHESSBOARD[BX-28]、CHESSBOARD[BX-42]、CHESSBOARD[BX-56]同为1，黑棋手胜利，同为2，白棋手胜利。

1. 斜下连成5个

CHESSBOARD[BX]、CHESSBOARD[BX+16]、CHESSBOARD[BX+32]、CHESSBOARD[BX+48]、CHESSBOARD[BX+64]同为1，黑棋手胜利，同为2，白棋手胜利。

**- 硬件层面：**

8086芯片是程序驱动核心；

8254计时器实现对1MHz信号的分频，输出1Hz的时钟信号：8254的计数器1工作在方式3，初值为1000，输出方波；计数器2工作在方式3，初值为1000，输出方波；

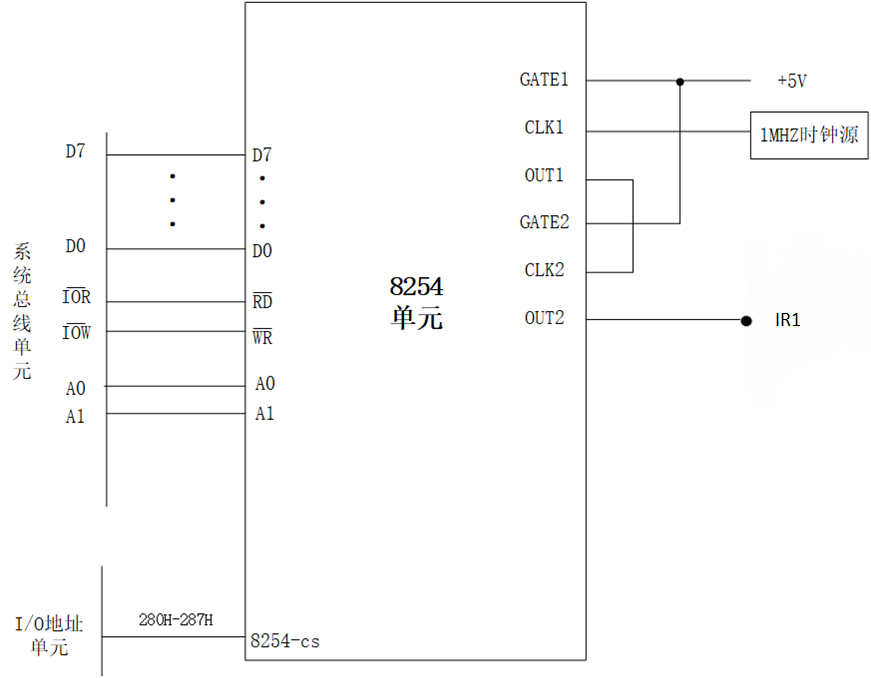
8255芯片用于输出控制数码管的信号，端口A和B均为输出，工作在方式0。

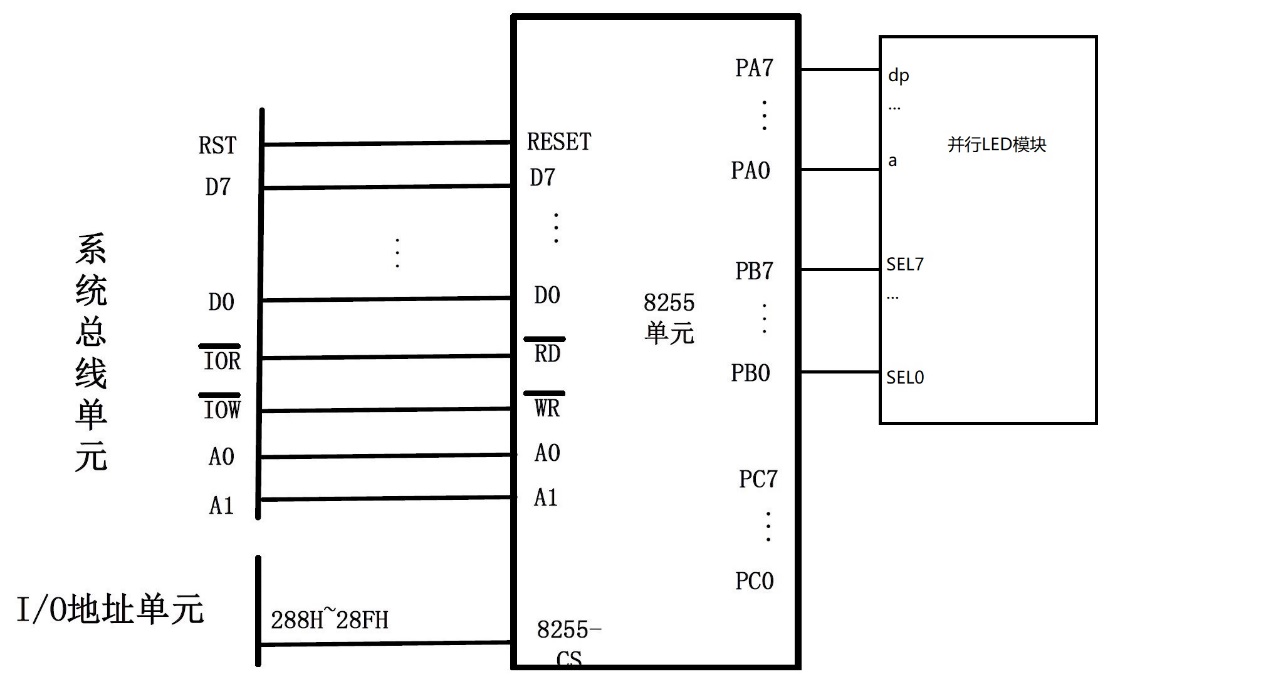
8259用于接收中断信号和8086相连实现查询中断，控制字的设置要求为：边沿触发、单片8259、需要ICW4、中断自动结束、系统为8086、开放中断请求IR0和IR1。

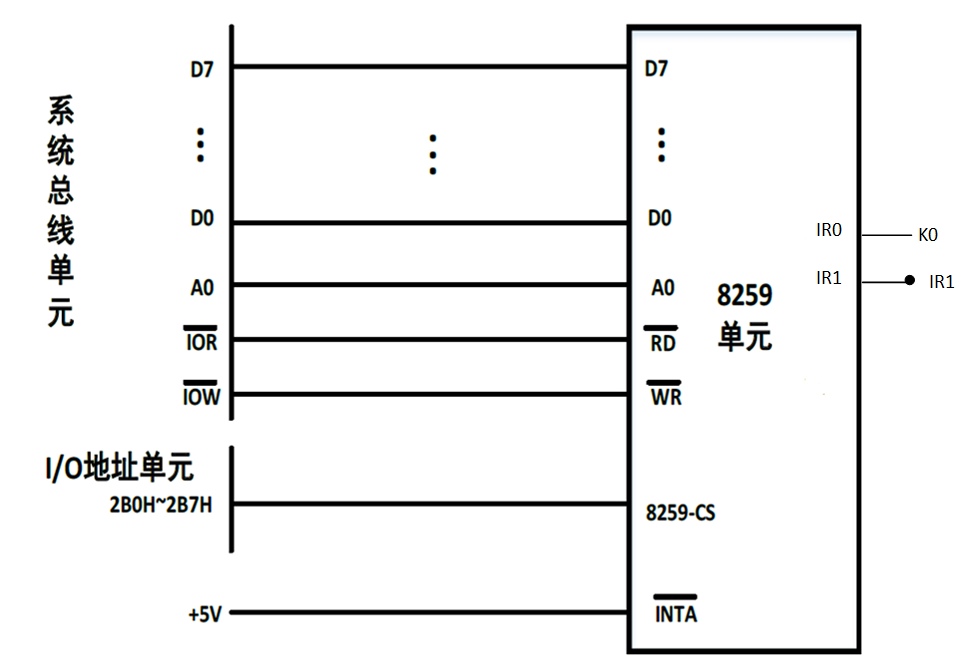
IR1与8254相连，1秒触发一次IR1中断，控制8255芯片输出控制数码管的信号，从而实现倒计时功能，IR0与开关K0相连，接收开关K0给出的中断信号，拨动开关K0触发IR0中断，时钟停止计时，转到中断服务程序，通过键盘在屏幕上输入棋手下棋位置。

数码管用来显示倒计时数字，段码输入端与8255的PA输出端相连，位码输入端与8255的PB输出端相连。

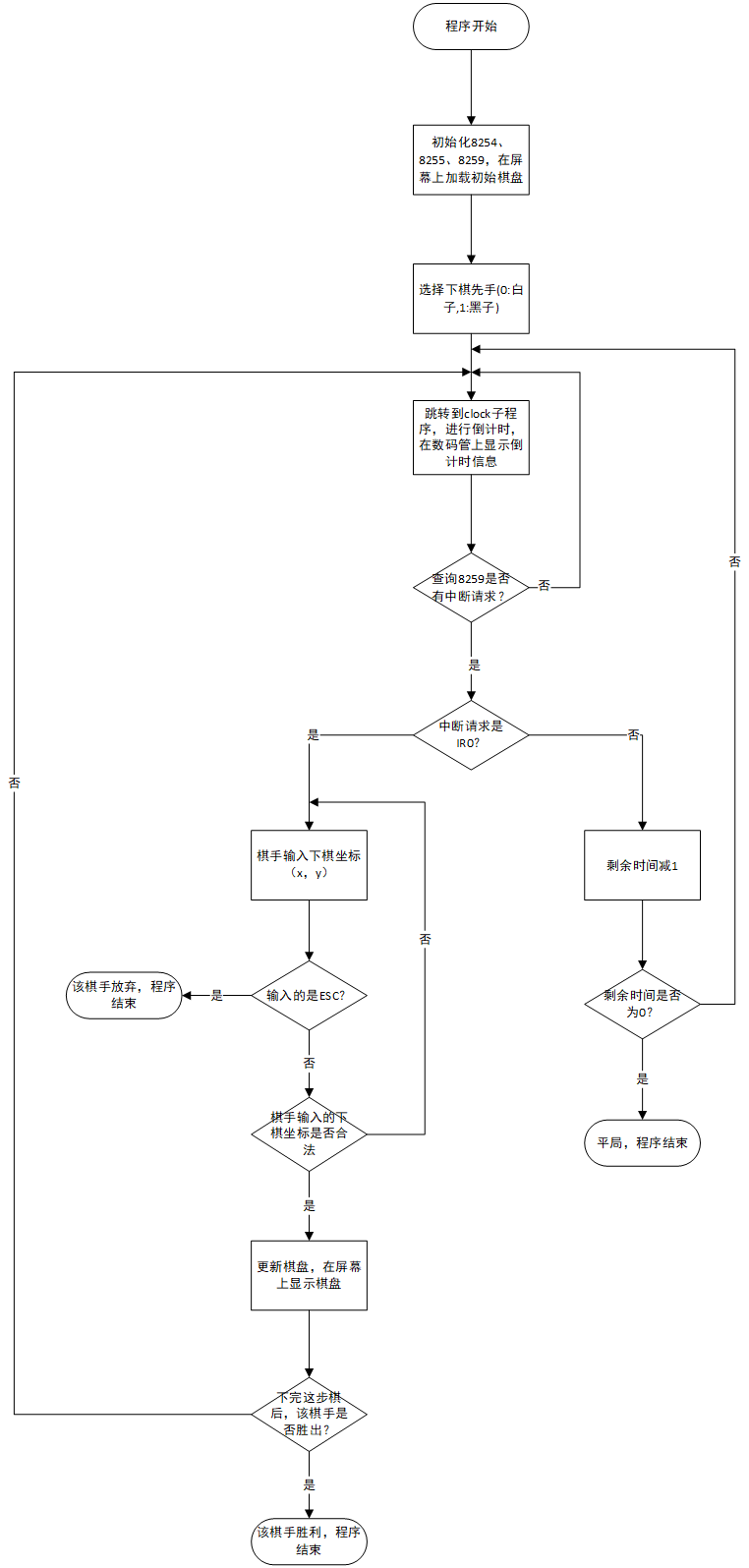
1. **设计接线图**

**



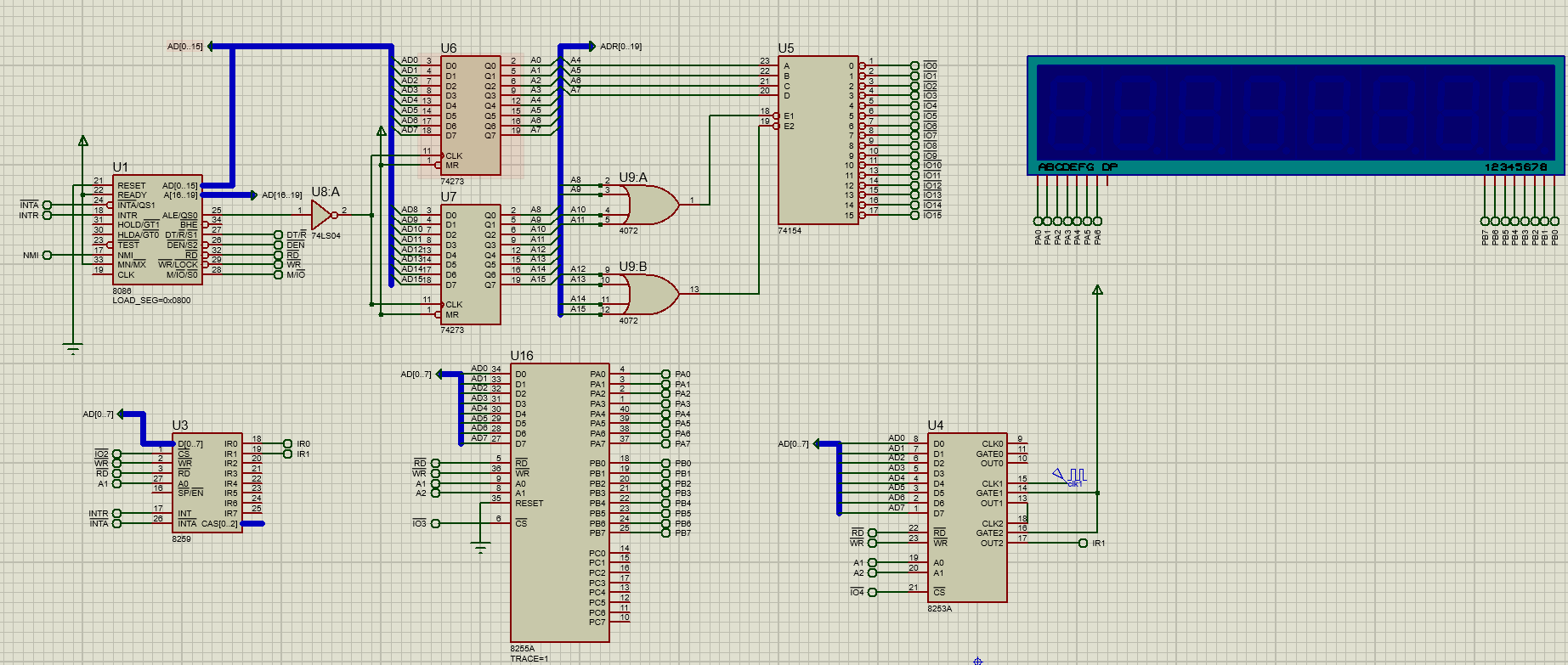
**

1. **程序流程图**

****

1. **仿真设计实现**

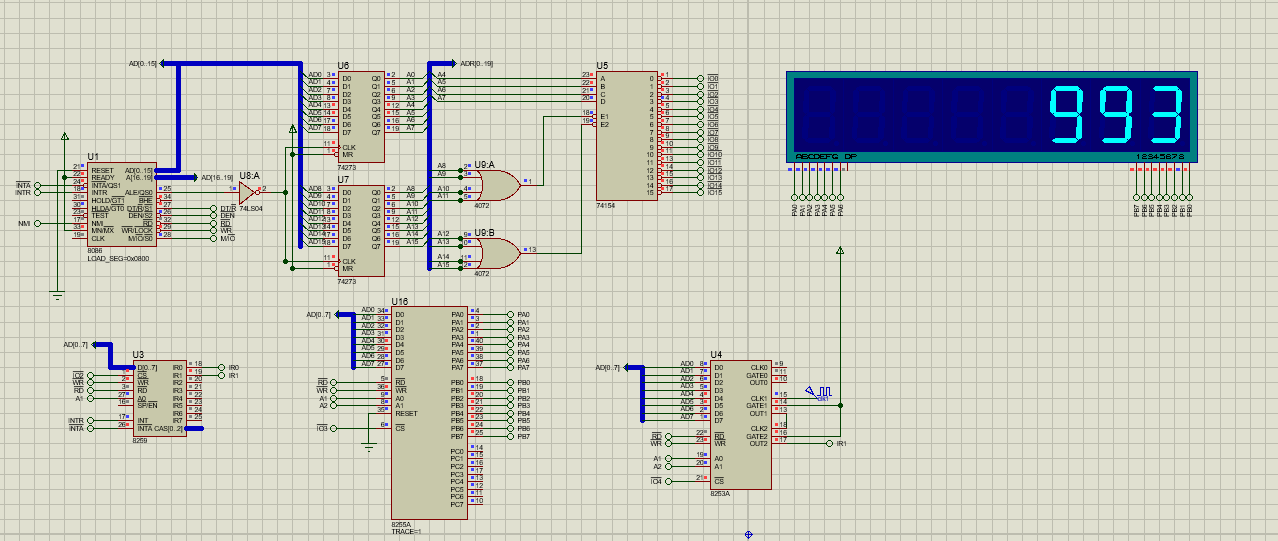
**设计图：**



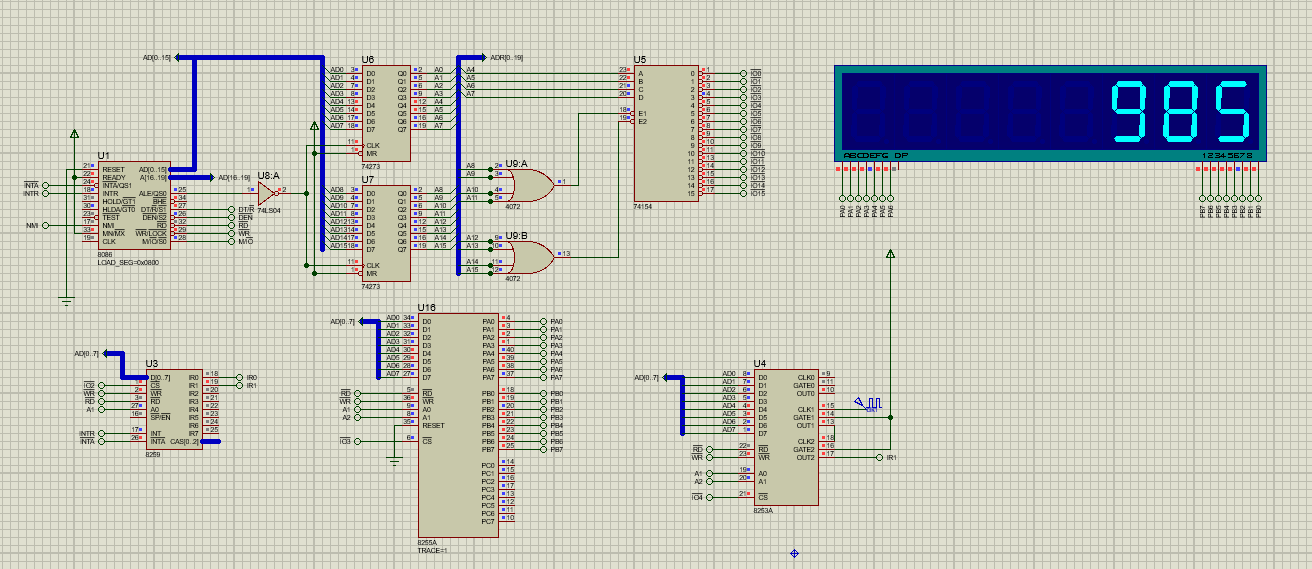
**源代码：**

|  |
| --- |
| ;中断向量设置，X为中断向量，Y为中断子程序名  SET\_INT MACRO X, Y  PUSH DS  MOV AX, 0  MOV DS, AX  MOV DI, X\*4  MOV AX, OFFSET Y  MOV [DI], AX  MOV AX, SEG Y  MOV [DI+2], AX  POP DS  ENDM  DATA SEGMENT  letter db 3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh  IO8255\_MODE equ 36H ;11mode  IO8255\_A equ 30H ;00A  IO8255\_B equ 32H ;01B  IO8255\_C equ 34H ;10C  bit dw 9,9,9  IO8254\_MODE EQU 46H ;8254控制寄存器端口地址  ;8254计数器0端口地址 0b000h  IO8254\_COUNT1 EQU 42H ;8254计数器1端口地址  IO8254\_COUNT2 EQU 44H ;8254计数器2端口地址  I8259\_1 EQU 20H ; 8259的ICW1端口地址  I8259\_2 EQU 22H ; 8259的ICW2端口地址  I8259\_3 EQU 22H ; 8259的ICW3端口地址  I8259\_4 EQU 22H ; 8259的ICW4端口地址  O8259\_1 EQU 22H ; 8259的OCW1端口地址  O8259\_2 EQU 20H ; 8259的OCW2端口地址  O8259\_3 EQU 20H ; 8259的OCW3端口地址  DATA ENDS  STACK SEGMENT  DW 1024H DUP(?)  STACK ENDS  CODE SEGMENT  ASSUME CS:CODE, DS:DATA  START:    MOV AX, DATA  MOV DS, AX  MOV ES, AX  mov si,0  mov cx,0  CLI  ; 8259A初始化段代码  SET\_INT 81H, INT\_1  MOV DX, I8259\_1 ;初始化8259的ICW1  MOV AL, 13H ;边沿触发、单片8259、需要ICW4  OUT DX, AL    MOV DX, I8259\_2 ;初始化8259的ICW2  MOV AL, 80H  OUT DX, AL  MOV AL, 03H ;初始化8259的ICW4, 主片，自动EOI，8086系统  OUT DX, AL    ;MOV DX,O8259\_1 ;初始化8259的中断屏蔽操作命令字ocw1  ;MOV AL,11111101b ;打开IR0、IR1屏蔽位  ; OUT DX,AL  MOV DX, IO8254\_MODE ;初始化8254工作方式  MOV AL, 00110110B ;计数器0，方式3  OUT DX, AL  ;计数器1的初值为1000(3E8H)  MOV DX, IO8254\_COUNT1 ;装入计数初值al  MOV AX,1000  OUT DX,AL  MOV AL, AH  OUT DX,AL    MOV DX, IO8254\_MODE ;初始化8254工作方式  MOV AL, 01110110B ;计数器1，方式3  OUT DX, AL  ;计数器1的初值为1000(3E8H)  MOV DX, IO8254\_COUNT1 ;装入计数初值al  MOV AL, 0E8H ;先读低八位  OUT DX,AL  ;装入计数初值ah  MOV AL, 03H ;后读高八位  OUT DX,AL  MOV DX, IO8254\_MODE ;初始化8254工作方式  MOV AL, 10110110B ;计数器2，方式3  OUT DX, AL    ;计数器2的初值为1000  MOV DX, IO8254\_COUNT2 ;装入计数初值al  MOV AL, 0E8H ; 先读低第八位  OUT DX,AL  ;装入计数初值ah  MOV AL, 03H ; 后读高八位  OUT DX,AL      MOV dx,IO8255\_MODE  MOV al,80H ; 端口A、B、C均为方式0，输出  OUT dx,al  ;初始化8259、8254  STI ;开中断  QUERY:  call CLOCK  ; call query\_8259  JMP QUERY    CLOCK PROC  ;计时  MOV dx,IO8255\_A  MOV al,0  OUT dx,al    MOV dx,IO8255\_B  MOV al,11111110b ;S0=0,最低位数码管亮，低电平选中  OUT dx,al    MOV dx,IO8255\_A  MOV si,bit[4] ;个位  MOV al,letter[si] ;读入一个字符段码  OUT dx,al ;A口输出    MOV dx,IO8255\_A  MOV al,0  OUT dx,al    MOV dx,IO8255\_B  MOV al,11111101b ;S1=0,次低位数码管亮，低电平选中  OUT dx,al    MOV dx,IO8255\_A  MOV si,bit[2] ;十位  MOV al,letter[si] ;读入一个字符位码  OUT dx,al ;A口输出    MOV dx,IO8255\_A  MOV al,0  OUT dx,al    MOV dx,IO8255\_B  MOV al,11111011b ;S2=0,次低位数码管亮  OUT dx,al    MOV dx,IO8255\_A  MOV si,bit[0] ;百位  MOV al,letter[si] ;读入一个字符段码  OUT dx,al ;A口输出  RET  CLOCK ENDP    INT\_1 PROC  ;MOV DX,O8259\_3 ;向8259发送查询命令  ;MOV AL,6CH ;设置为中断查询方式  ;OUT DX,AL  ;IN AL,DX ;读出查询字  ;TEST AL,80H ;判断中断是否已响应，d7是否为1  ;AND AL,07H ;保留低3位，d2、d1、d0  ;CMP AL,01H    ;JE IR1ISR ;若为IR1请求，跳到IR1处理程序  ;JMP NO\_SIGNAL  ;IR1ISR:  CLI  PUSH AX  PUSH BX  PUSH CX  PUSH DX    MOV ax,bit[4]  CMP ax,0  JZ shiwei ;个位为0，跳转到十位  DEC ax  MOV bit[4],ax ;个位不为0减1  JMP EOI ;结束中断  shiwei:  MOV ax,bit[2]  CMP ax,0  JZ baiwei ;十位为0，跳转到百位  DEC ax  MOV bit[2],ax ;十位不为0减1  MOV bit[4],9 ;个位变成9  JMP EOI ;结束中断  baiwei:  MOV ax,bit[0]  DEC ax  MOV bit[0],ax ;百位不为0减1  MOV bit[2],9 ;十位变成9  MOV bit[4],9 ;个位变成9  JMP EOI ;结束中断  EOI:  POP DX  POP CX  POP BX  POP AX  STI ;开中断  IRET ;中断返回  INT\_1 ENDP    CODE ENDS  END START |

**实现结果：**



某一时刻时钟显示为993，表示剩余时间为993秒



8秒过后时钟显示为985，表示剩余时间为985秒

**仿真设计与实验箱设计的差异：**

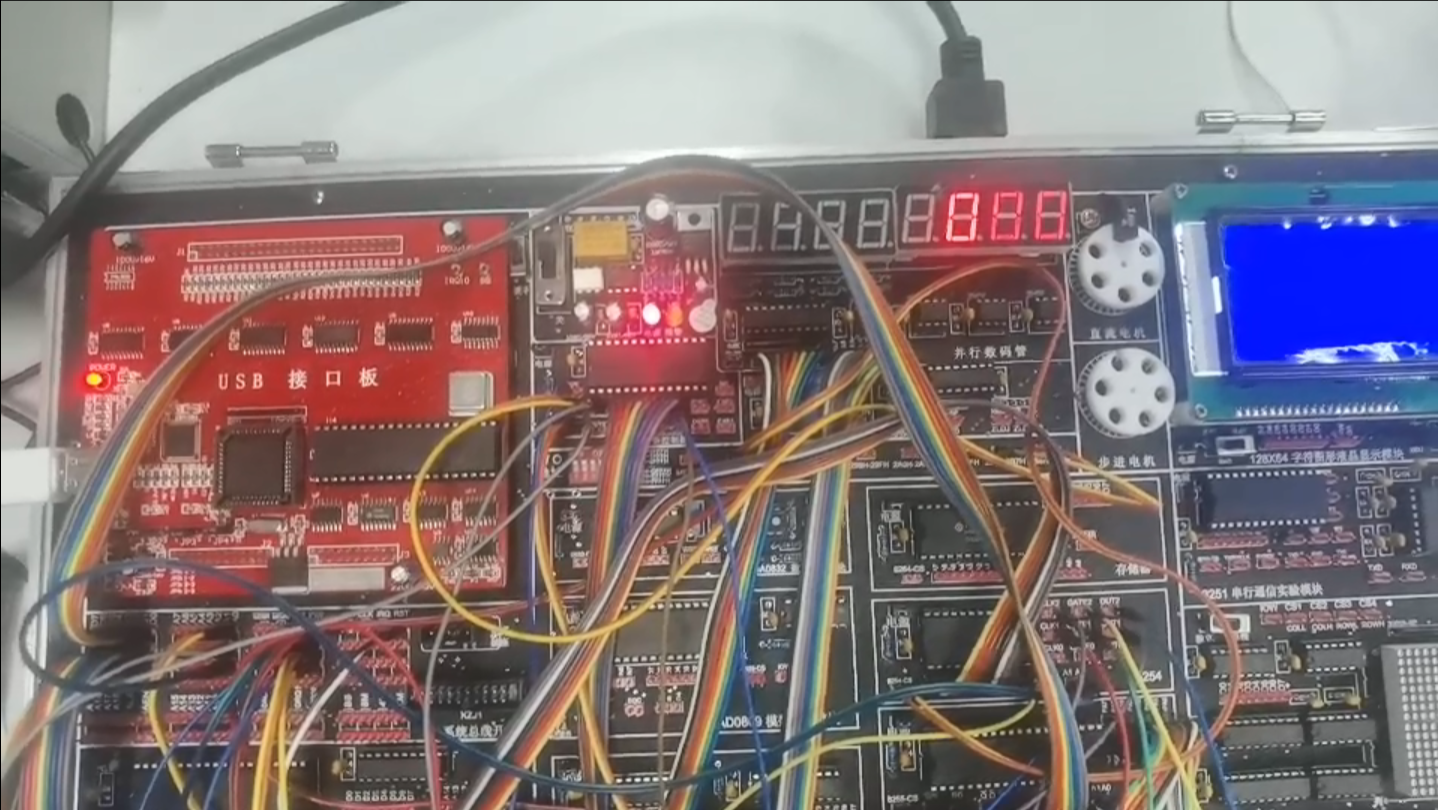
经探索与询问可知，对于中断的处理，Proteus仿真只支持中断向量，所以在实验箱上使用的查询中断不能用于Proteus仿真。并且Proteus仿真不支持DOS系统功能调用，故不能用Proteus仿真键盘输入和屏幕显示功能，所以我通过录屏对与屏幕和键盘相关的功能进行说明。

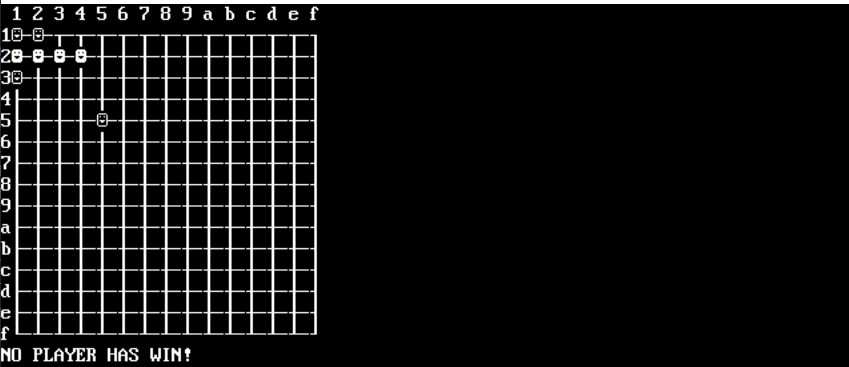
1. **实验源代码**

|  |
| --- |
| DATA SEGMENT  letter db 3fh,06h,5bh,4fh,66h,6dh,7dh,07h,7fh,6fh  IO8255\_MODE equ 28BH  IO8255\_A equ 288H  IO8255\_B equ 289H  IO8255\_C equ 28AH  bit dw 9,9,9  CHESSBOARD DB 218,13 DUP(194),191,13 DUP(195,13 DUP(197),180),192,13 DUP(193),217 ;设置棋盘的缓冲区，1是黑子，2是白子  X DB 0 ;落子坐标 x  Y DB 0 ;落子坐标 y  FLAG DB 0 ;判断是否可以落子的标记，1为可以，0为不可以  OVER DB 0 ;判断是否比赛结束，CALL ISWIN 0为没有结束，1为结束。结束时，最后落子方获胜  TEMP DB 0 ;判断该下黑子还是白子，TEMP=0白子，TEMP=1黑子  TI DB ' 1 2 3 4 5 6 7 8 9 a b c d e f',0AH,0DH,'$' ;棋盘的x,y坐标  ERROR1 DB 'YOU CANNOT PUT HERE!',0AH,0DH,'$' ;报错,"你不能放在这里"  CLEAN DB 72 DUP(32),0AH,0DH,72 DUP(32),0AH,0DH,72 DUP(32),0AH,0DH,72 DUP(32),0AH,0DH,72 DUP(32),0AH,0DH,72 DUP(32),0AH,0DH,72 DUP(32),0AH,0DH,72 DUP(32),0AH,0DH,'$';更新棋盘  PUT DB 'PLEASE INPUT THE POSITION(X Y): ',0AH,0DH,'$' ;请输入棋子的位置(x,y)  ORDER DB 'PLEASE INPUT THE FIRST PLAYER.(0:white,1:black)',0AH,0DH,'$' ;下棋先手  WHITEWIN DB 'WHITE PLAYER HAS WIN!',0AH,0DH,'$' ;白子赢  BLACKWIN DB 'BLACK PLAYER HAS WIN!',0AH,0DH,'$' ;黑子赢  PING DB 'NO PLAYER HAS WIN!',0AH,0DH,'$'  BLACKEXIT DB 'BLACK PLAYER HAS QUIT!',0AH,0DH,'$'  WHITEEXIT DB 'WHITE PLAYER HAS QUIT!',0AH,0DH,'$'  ENTER DB 0AH,0Dh,'$' ;回车换行  DATA ENDS  IO8254\_MODE EQU 283H ;8254控制寄存器端口地址  IO8254\_COUNT1 EQU 281H ;8254计数器1端口地址  IO8254\_COUNT2 EQU 282H ;8254计数器2端口地址  I8259\_1 EQU 2B0H ; 8259的ICW1端口地址  I8259\_2 EQU 2B1H ; 8259的ICW2端口地址  I8259\_3 EQU 2B1H ; 8259的ICW3端口地址  I8259\_4 EQU 2B1H ; 8259的ICW4端口地址  O8259\_1 EQU 2B1H ; 8259的OCW1端口地址  O8259\_2 EQU 2B0H ; 8259的OCW2端口地址  O8259\_3 EQU 2B0H ; 8259的OCW3端口地址  CODE SEGMENT  ASSUME CS:CODE, DS:DATA  START:  MOV AX, DATA  MOV DS, AX  MOV ES, AX  mov si,0  mov cx,0  CALL INITIAL ;初始化8259、8254  MOV AL,11H ;在屏幕上显示  MOV AH,00H  INT 10H ;设置显示器模式640×480 2 色,清空屏幕  CALL PRINT ;打印棋盘  MOV DX,OFFSET ORDER ;选择先手，0：while，1：black  MOV AH,09H  INT 21H    mov AL,0  MOV AH,1  INT 21H  sub AL,30h  MOV TEMP,AL  MOV DX,OFFSET ENTER ;回车换行  MOV AH,09H  INT 21H  QUERY:  JMP CLOCK  CHAXUN:  MOV DX,O8259\_3 ;向8259发送查询命令  MOV AL,00001100b ;设置为中断查询方式  OUT DX,AL  IN AL,DX ;读出查询字  TEST AL,80H ;判断中断是否已响应，d7是否为1  JZ QUERY ;没有响应则继续查询  AND AL,07H ;保留低3位，d2、d1、d0  CMP AL,00H  JE IR0ISR ;若为IR0请求，跳到IR0处理程序  CMP AL,01H  JE IR1ISR ;若为IR1请求，跳到IR1处理程序  IR0ISR: ;GAME  MOV DX,OFFSET PUT ;放置棋子  MOV AH,09H ;在屏幕上显示输入的内容  INT 21H  MOV AH,1 ;若输入的是ESC则退出  INT 21H  CMP AL,27 ;若输入的是ESC  JE QUIT ;退出游戏  JMP RXY1 ;否则输入坐标X Y  QUIT: ;退出游戏的信息  JMP QUIT\_1  RXY1: ;记录坐标X Y(ASCII码)  MOV X,AL ;记录x的坐标    INT 21H ;读入间隔符  CMP AL,27 ;若是ESC则退出  JE QUIT ;退出    INT 21H ;读入y  CMP AL,27 ;若是ESC则退出  JE QUIT ;退出  MOV Y,AL ;记录y的坐标    N1: MOV AH,07 ;无回显输入  INT 21H  CMP AL,27 ;若是ESC则退出  JE QUIT ;退出游戏  CMP AL,13 ;若是回车则继续，否则等待回车  JNE N1 ;继续执行N1程序    MOV AH,2  MOV DL,0AH  INT 21H ;输出回车换行  MOV DL,0DH  INT 21H    MOV FLAG,1 ;flag的值为1  CALL CHECK ;检查可否落子，将X，Y改变为真实的数值    CMP FLAG,1 ;可以落子  JE THERE1 ;可以落子则判断落子  JMP EOI ;如果不可以落子则重新输入    THERE1:  CALL PUTDOWN1 ;落子v  CALL ISWIN ;判断输赢，有结果则OVER=1  CALL PRINT ;打印棋盘  CMP OVER,1 ;游戏结束  JNZ EOI  END1:  CMP TEMP,1  JNZ BLACK  MOV DX,OFFSET WHITEWIN ;白子赢  JMP END\_DISPLAY  BLACK:  MOV DX,OFFSET BLACKWIN ;黑子赢  JMP END\_DISPLAY  END2:  MOV DX,OFFSET PING ;游戏结束的信息提示  JMP END\_DISPLAY  IR1ISR:  PUSH ax  MOV ax,bit[4]  CMP ax,0  JZ shiwei ;个位为0，跳转到十位  DEC ax  MOV bit[4],ax ;个位不为0减1  JMP EOI ;结束中断  shiwei:  MOV ax,bit[2]  CMP ax,0  JZ baiwei ;十位为0，跳转到百位  DEC ax  MOV bit[2],ax ;十位不为0减1  MOV bit[4],9 ;个位变成9  JMP EOI ;结束中断  baiwei:  MOV ax,bit[0]  CMP ax,0  JZ END2 ;百位为0，平局  DEC ax  MOV bit[0],ax ;百位不为0减1  MOV bit[2],9 ;十位变成9  MOV bit[4],9 ;个位变成9  JMP EOI ;结束中断  EOI:  MOV DX,O8259\_2 ;向8259发送中断结束命令  MOV AL,00100000b ;一般的中断结束命令  OUT DX,AL  JMP QUERY  CLOCK:  ;计时  MOV dx,IO8255\_A  MOV al,0  OUT dx,al    MOV dx,IO8255\_B  MOV al,00000001b ;S0=1,最低位数码管亮  OUT dx,al    MOV dx,IO8255\_A  MOV si,bit[4] ;个位  MOV al,letter[si] ;读入一个字符段码  OUT dx,al ;A口输出    MOV dx,IO8255\_A  MOV al,0  OUT dx,al    MOV dx,IO8255\_B  MOV al,00000010b ;S1=1,次低位数码管亮  OUT dx,al    MOV dx,IO8255\_A  MOV si,bit[2] ;十位  MOV al,letter[si] ;读入一个字符段码  OUT dx,al ;A口输出    MOV dx,IO8255\_A  MOV al,0  OUT dx,al    MOV dx,IO8255\_B  MOV al,00000100b ;S2=1,次低位数码管亮  OUT dx,al    MOV dx,IO8255\_A  MOV si,bit[0] ;百位  MOV al,letter[si] ;读入一个字符段码  OUT dx,al ;A口输出  JMP CHAXUN    QUIT\_1 PROC ;退出游戏的信息  CMP TEMP,0  JNZ BLACK\_EXIT  MOV DX,OFFSET WHITEEXIT ;白子退出  JMP END\_DISPLAY  BLACK\_EXIT:  MOV DX,OFFSET BLACKEXIT ;黑子退出  JMP END\_DISPLAY  QUIT\_1 ENDP  ;检验落子位置是否合法  CHECK PROC NEAR ;落子位置是否合法的检查信息  PUSH AX ;保存CPU现场  PUSH BX  PUSH CX  PUSH DX  CMP X,'a' ;输入小于a，合法  JL CMPDX ;则进行数字判断  CMP X,'f' ;若输出大于f，不合法  JG ERR ;报错信息  SUB X,39  JMP CMPDY  CMPDX: ;X的数字判断  CMP X,'1' ;输入小于1，不合法  JL ERR ;报错信息  CMP X,'9' ;输入小于9，不合法  JG ERR ;报错信息  CMPDY: ;输入X合法，比较Y  CMP Y,'a' ;输入小于A，合法  JL CMPDY1 ;则进行数字判断  CMP Y,'f' ;输入大于F，不合法  JG ERR ;报错信息  SUB Y,39  JMP SUBXY  CMPDY1: ;Y的数字判断  CMP Y,'1' ;输入小于1，不合法  JL ERR ;不合法  CMP Y,'9' ;输入小于9，不合法  JG ERR ;不合法  SUBXY:  SUB X,'1' ;将X改变为真实的值  SUB Y,'1' ;将Y改变为真实的值  MOV CX,0 ;传送指令  MOV CL,Y  MOV BX,0 ;清空寄存器  MULX1:  ADD BL,15 ;棋子右移15单位  LOOP MULX1 ;循环MULX1  ADD BL,X ;棋子右移输入Y的值  CMP CHESSBOARD[BX],1 ;若此处已有棋子，输入不合法  JE ERR  CMP CHESSBOARD[BX],2 ;若此处没有棋子，输入合法  JNE RETURNC  ERR:  MOV FLAG,0 ;对于不合法的输入，显示错误信息  MOV DX,OFFSET ERROR1  MOV AH,09H ;在屏幕上显示输入错误的信息  INT 21H  RETURNC:  POP DX ;恢复CPU现场  POP CX  POP BX  POP AX  RET ;子程序结束返回  CHECK ENDP  END\_DISPLAY PROC NEAR  MOV AH,09H  INT 21H  MOV AH,4CH ;退出游戏  INT 21H  ret  END\_DISPLAY ENDP  ;落子子程序  PUTDOWN1 PROC NEAR ;单机落子的信息提示  PUSH AX ;保存CPU现场  PUSH BX  PUSH CX  PUSH DX  MOV CX,0 ;字符指针初始化  MOV CL,Y  MOV BX,0 ;清空寄存器  MULX2:  ADD BL,15 ;字符指针右移15个字节，行  LOOP MULX2 ;循环MULX2  ADD BL,X ;字符指针右移Y个字节，列  CMP TEMP,1 ;根据TEMP值，轮流放置黑子和白子  JE MM1  MOV CHESSBOARD[BX],2 ;放白子  MOV TEMP,1 ;根据TEMP值，轮流放置黑子和白子  JMP YY1  MM1: ;TEMP=1  MOV CHESSBOARD[BX],1 ;放黑子  MOV TEMP,0  YY1:  POP DX ;恢复CPU现场  POP CX  POP BX  POP AX  RET ;子程序结束返回  PUTDOWN1 ENDP    ;判断是否获胜  ISWIN PROC NEAR ;获胜的信息提示  MOV X,0 ;初始化X和Y  MOV Y,0  LOOPY:  MOV CX,0 ;字符指针初始化  MOV CL,Y  MOV BX,0 ;清空寄存器  MULX3:  ADD BL,15 ;字符指针右移15个字节  LOOP MULX3 ;循环MULX3  ADD BL,X ;BX=15\*X+Y  MOV DL,CHESSBOARD[BX]  CMP DL,1  JE PANDUAN  CMP DL,2  JE PANDUAN ;判断是否可以连成5个  JMP NEXT ;进入下一轮判断  PANDUAN: ;游戏胜利的判断  CALL TEST1 ;横着  CMP OVER,1 ;横着连成5个游戏结束  JE RETURNISWIN ;返回胜利的判断  CALL TEST2 ;竖着  CMP OVER,1 ;竖着连成5个游戏结束  JE RETURNISWIN ;返回胜利的判断  CALL TEST3 ;斜上  CMP OVER,1 ;斜上连成5个游戏结束  JE RETURNISWIN ;返回胜利的判断  CALL TEST4 ;斜下  CMP OVER,1 ;斜下连成5个游戏结束  JE RETURNISWIN ;返回胜利的判断  NEXT:  INC Y ;Y的字符指针右移  CMP Y,15 ;比较Y的值  JNE LOOPY  MOV Y,0 ;初始化Y的值  INC X ;X的字符指针右移  CMP X,15 ;比较X的值  JNE LOOPY  RETURNISWIN:  RET  ISWIN ENDP  ;判断横向是否连成5个  TEST1 PROC NEAR  PUSH BX  CMP Y,10 ;Y>10横向不能连成5个  JG RETURN1  CMP DL,CHESSBOARD[BX+1] ;判断棋盘横向是否有2个棋子连在一起  JNE RETURN1  CMP DL,CHESSBOARD[BX+2] ;判断棋盘横向是否有3个棋子连在一起  JNE RETURN1  CMP DL,CHESSBOARD[BX+3] ;判断棋盘横向是否有4个棋子连在一起  JNE RETURN1  CMP DL,CHESSBOARD[BX+4] ;判断棋盘横向是否有5个棋子连在一起  JNE RETURN1  MOV OVER,1 ;游戏结束  RETURN1:  POP BX  RET  TEST1 ENDP  ;判断纵向是否连成5个  TEST2 PROC NEAR  PUSH BX  CMP X,10 ;X>10纵向不能连成5个  JG RETURN2  CMP DL,CHESSBOARD[BX+15] ;判断棋盘纵向是否有2个棋子连在一起  JNE RETURN2  CMP DL,CHESSBOARD[BX+30] ;判断棋盘纵向是否有3个棋子连在一起  JNE RETURN2  CMP DL,CHESSBOARD[BX+45] ;判断棋盘纵向是否有4个棋子连在一起  JNE RETURN2  CMP DL,CHESSBOARD[BX+60] ;判断棋盘纵向是否有5个棋子连在一起  JNE RETURN2  MOV OVER,1 ;游戏结束  RETURN2:  POP BX  RET  TEST2 ENDP  ;判断斜上是否连成5个  TEST3 PROC NEAR  PUSH BX  CMP X,4 ;X<4斜上不能连成5个  JL RETURN3  CMP Y,4 ;Y<4纵向不能连成5个  JL RETURN3  CMP DL,CHESSBOARD[BX-14] ;判断棋盘斜上是否有2个棋子连在一起  JNE RETURN3  CMP DL,CHESSBOARD[BX-28] ;判断棋盘斜上是否有3个棋子连在一起  JNE RETURN3  CMP DL,CHESSBOARD[BX-42] ;判断棋盘斜上是否有4个棋子连在一起  JNE RETURN3  CMP DL,CHESSBOARD[BX-56] ;判断棋盘斜上是否有5个棋子连在一起  JNE RETURN3  MOV OVER,1 ;游戏结束  RETURN3:  POP BX  RET  TEST3 ENDP  ;判断斜下是否连成5个  TEST4 PROC NEAR  PUSH BX  CMP X,10 ;X>10斜下不能连成5个  JG RETURN4  CMP Y,10 ;Y>10斜下不能连成5个  JG RETURN4  CMP DL,CHESSBOARD[BX+16] ;判断棋盘斜下是否有2个棋子连在一起  JNE RETURN4  CMP DL,CHESSBOARD[BX+32] ;判断棋盘斜下是否有3个棋子连在一起  JNE RETURN4  CMP DL,CHESSBOARD[BX+48] ;判断棋盘斜下是否有4个棋子连在一起  JNE RETURN4  CMP DL,CHESSBOARD[BX+64] ;判断棋盘斜下是否有5个棋子连在一起  JNE RETURN4  MOV OVER,1 ;游戏结束  RETURN4:  POP BX  RET  TEST4 ENDP  ;打印棋盘  PRINT PROC NEAR ;打印棋盘  PUSH SI  PUSH AX  PUSH DX  MOV AH,02H ;使用10H中断的设置光标位置功能  MOV DL,00H ;光标从0,0开始  MOV DH,00H ;光标的列坐标  INT 10H  MOV DX,OFFSET TI ;指定字符串  MOV AH,09H ;屏幕显示字符串  INT 21H  MOV X,0 ;初始化X Y SI  MOV Y,0  MOV SI,0  LOOP2:  CMP Y,0 ;判断Y是否为0  JNE NOTHEAD  MOV DL,X  ADD DL,31H ;X的字符指针右移  CMP DL,'9' ;判断X是否大于等于9  JLE PP  ADD DL,39 ;X的字符指针右移39个字节  PP:  MOV AH,02H  INT 21H ;使用21H中断的输出字符功能  NOTHEAD:  MOV DL,CHESSBOARD[SI]  MOV AH,02H  INT 21H  INC SI ;SI、Y指针同时右移1个字节，指向下一个字符  INC Y ;SI、Y指针同时右移1个字节，指向下一个字符  CMP Y,15 ;判断Y的大小  JE NEXTLINE  MOV DL,'-' ;输出一个'-'  MOV AH,02H ;使用21H中断的输出字符功能  INT 21H  JMP LOOP2 ;回到循环2  NEXTLINE:  MOV DL,32  MOV AH,02H  INT 21H  MOV DL,0AH ;输出一个回车符（0AH）  MOV AH,02H ;使用21H中断的输出字符功能  INT 21H  MOV DL,0DH ;输出一个换行符（0AD）  MOV AH,02H ;使用21H中断的输出字符功能  INT 21H  INC X ;X的字符指针右移1个字节  MOV Y,0 ;初始化Y  CMP X,15  JNE LOOP2  MOV DX,OFFSET CLEAN ;更新屏幕的信息提示  MOV AH,09H ;使用21H中断的显示字符串功能  INT 21H  MOV AH,02H ;使用10H中断的设置光标位置功能  MOV DL,00H ;光标从0,17开始  MOV DH,10H ;设置光标的列坐标  INT 10H  POP DX  POP AX  POP SI  RET  PRINT ENDP  INITIAL PROC NEAR  MOV DX, I8259\_1 ;初始化8259的ICW1  MOV AL, 00010011b ;边沿触发、单片8259、需要ICW4  OUT DX,AL  MOV DX,I8259\_2 ;初始化8259的ICW2  MOV AL,0B0H ;10110000b  OUT DX,AL    MOV AL,03H ;初始化ICW4  OUT DX,AL    MOV DX,O8259\_1 ;初始化8259的中断屏蔽操作命令字ocw1  MOV AL,11111100b ;打开IR0、IR1  OUT DX,AL  MOV DX, IO8254\_MODE ;初始化8254工作方式  MOV AL, 00110110B ;计数器0，方式3  OUT DX, AL  ;计数器1的初值为1000(3E8H)  MOV DX, IO8254\_COUNT1 ;装入计数初值al  MOV AL, 0E8H ;先读低八位  OUT DX,AL  ;装入计数初值ah  MOV AL, 03H ;后读高八位  OUT DX,AL  MOV DX, IO8254\_MODE ;初始化8254工作方式  MOV AL, 01110110B ;计数器1，方式3  OUT DX, AL    ;计数器2的初值为1000  MOV DX, IO8254\_COUNT2 ;装入计数初值al  MOV AL, 0E8H ; 先读低第八位  OUT DX,AL  ;装入计数初值ah  MOV AL, 03H ; 后读高八位  OUT DX,AL    MOV dx,IO8255\_MODE  MOV al,80H ; 端口A、B、C均为方式0，输出  OUT dx,al  RET  CODE ENDS  END START |

1. **实验现象**

时钟计数到0，仍为决出胜者，屏幕上输出平局信息。

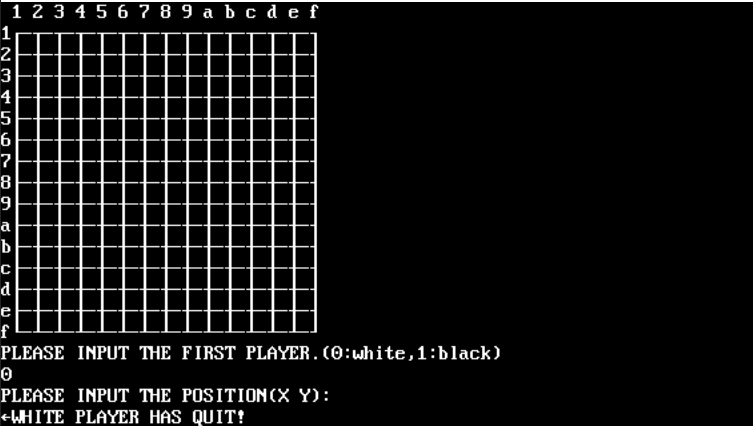




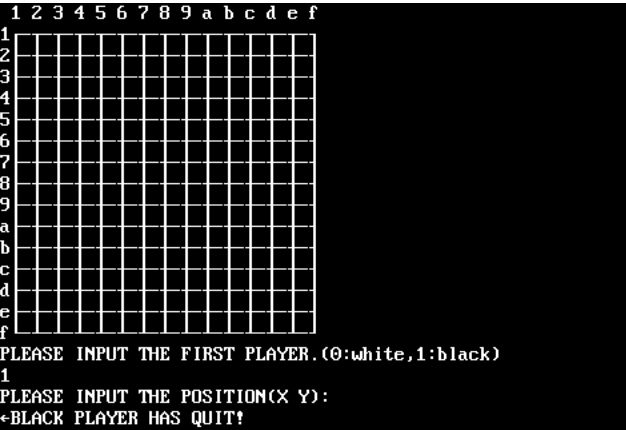
下棋位置不合法，屏幕输出结果如下图：



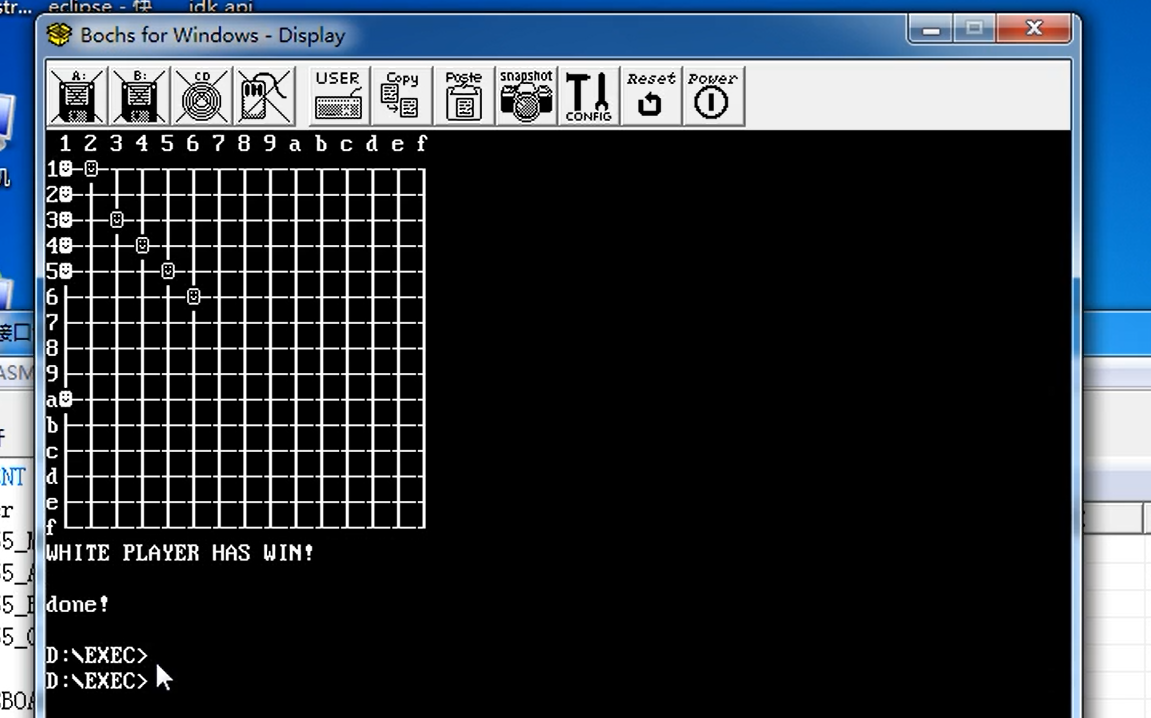
白棋手放弃，屏幕输出结果如下图：



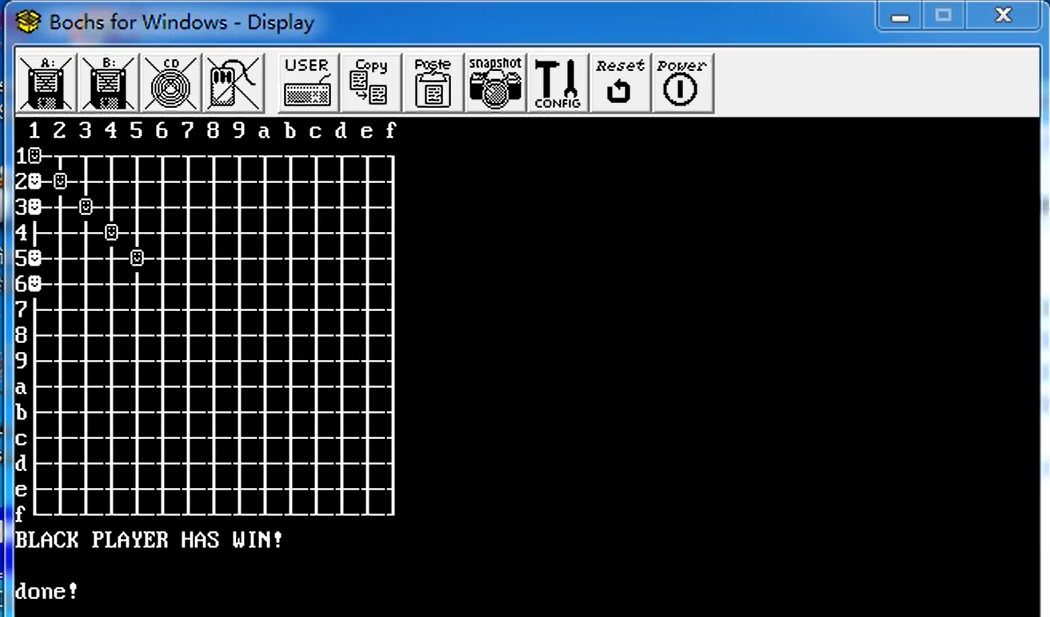
黑棋手放弃，屏幕输出结果如下图：



白棋手胜出，屏幕输出结果如下图：



黑棋手胜出，屏幕输出结果如下图：



详细运行结果展示可以查看演示视频。

1. **是否在实现功能的基础上进行了代码的优化或改进**

在代码中，我将许多重复的工作封装为子程序，从而减少代码量。如打印棋盘的PRINT，判断棋手是否胜利的ISWIN，检验落子位置是否合法CHECK、下棋PUSHDOWN1等。

我实现的五子棋的第一个版本，默认是白棋手先下棋，并不能由用户修改先手次序，而且只要有棋手胜出，就会输出胜利信息，结束程序，并不能判断是哪一个棋手赢得了棋局，当有棋手放弃时，也不能判断是哪一个棋手放弃了。老师在验收时，指出了这些功能设计上的缺陷，我在之后对程序进行了修改，最终实现了由用户决定先手和判断哪位棋手胜出或放弃的功能。具体实现方法如下：

程序中有一TEMP变量，TEMP=0时落子程序要下白子，TEMP=1时落子程序要下黑子。在初始化阶段，将TEMP变量赋值为0则先下白子，赋值为1则先下黑子。在落子之前，落子程序要判断棋手输入的是否是ESC，即判断是否有棋手退出， 如果检测到ESC的输入并且TEMP=0，则是白棋手放弃，如果TEMP=1,则是黑棋手放弃。白棋手落子之后，TEMP变为1，黑棋手落子之后，TEMP变为0，所以在判断输赢时，有棋手胜出且TEMP=1表示白棋手胜出，反之黑棋手胜出。

1. **实现效果的局限性分析**

针对实验最初设定的应用场景，最后的运行结果与预期相符。但是由于我不是专业的棋手，对五子棋的规则理解得还不够准确，只是实现了休闲时所下的五子棋的功能，可能还不适用于专业的比赛，后续可以认真学习五子棋的相关规则，进一步完善该程序的逻辑功能。

**实验结论（讨论）：**

本次接口实验，我通过查阅各类参考书和参考代码，在老师的悉心指导下，在同学的交流中，凭借自己的探索，终于掌握了x86汇编语言的基本用法，使用龙芯实验箱学习了通过汇编语言直接控制硬件的过程，进一步理解了8086 CPU的工作过程。我还学习了8254定时器/计数器芯片、8255并口通信芯片、8259中断控制芯片的用法以及数码管的使用，最终制作出可以通过键盘输入，在屏幕上显示的五子棋小游戏，收获了很大的成就感。总的来说，接口实验课程令我受益良多。