贪吃蛇

16 级通信工程 任思昊 10162100362 周二晚

实验目的：

实现用 F5 区键盘 控制 8x8 矩阵贪吃蛇吃食物。

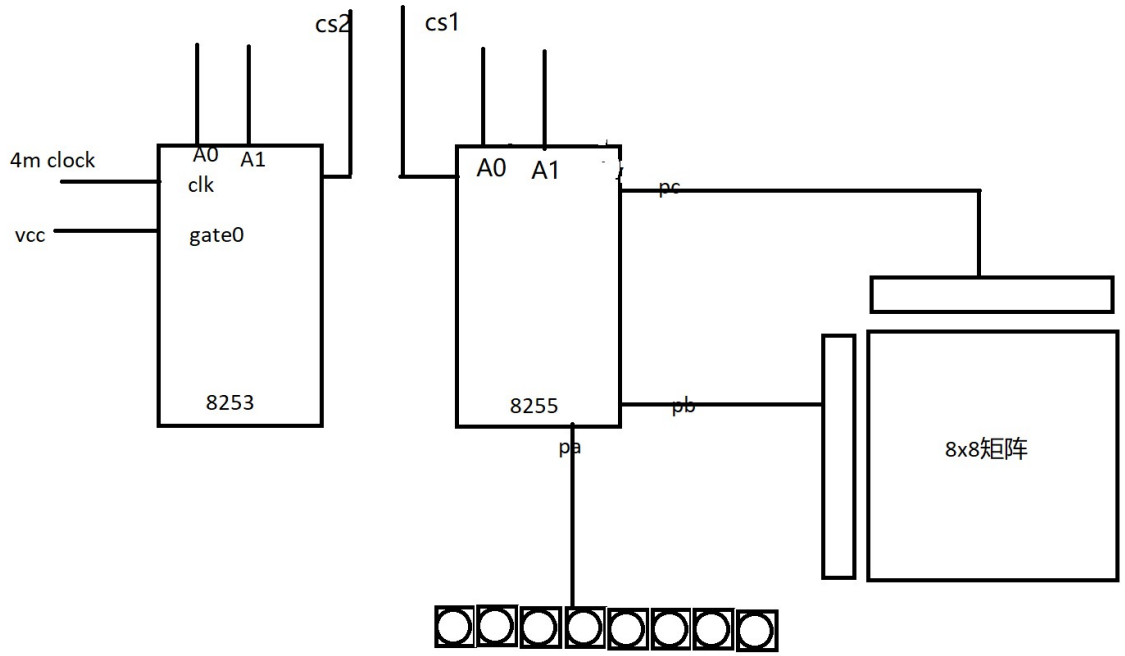
实验内容：

用 8253 来获取随机数用于生成食物，并检测食物是否生成在蛇身上。若是，则重新获取。然后用 8255 PA 口检测键盘输入，1234 控制蛇的移动，PB 和 PC 口分别控制 8x8 矩阵的行和列，每次点亮一个点，利用视觉暂留的原理来显示蛇和食物。蛇每吃一个食物变长一格，然后再通过 8253 随机生成下一个食物。当蛇撞到墙或自己时游戏结束，矩阵显示禁止通行的标志。

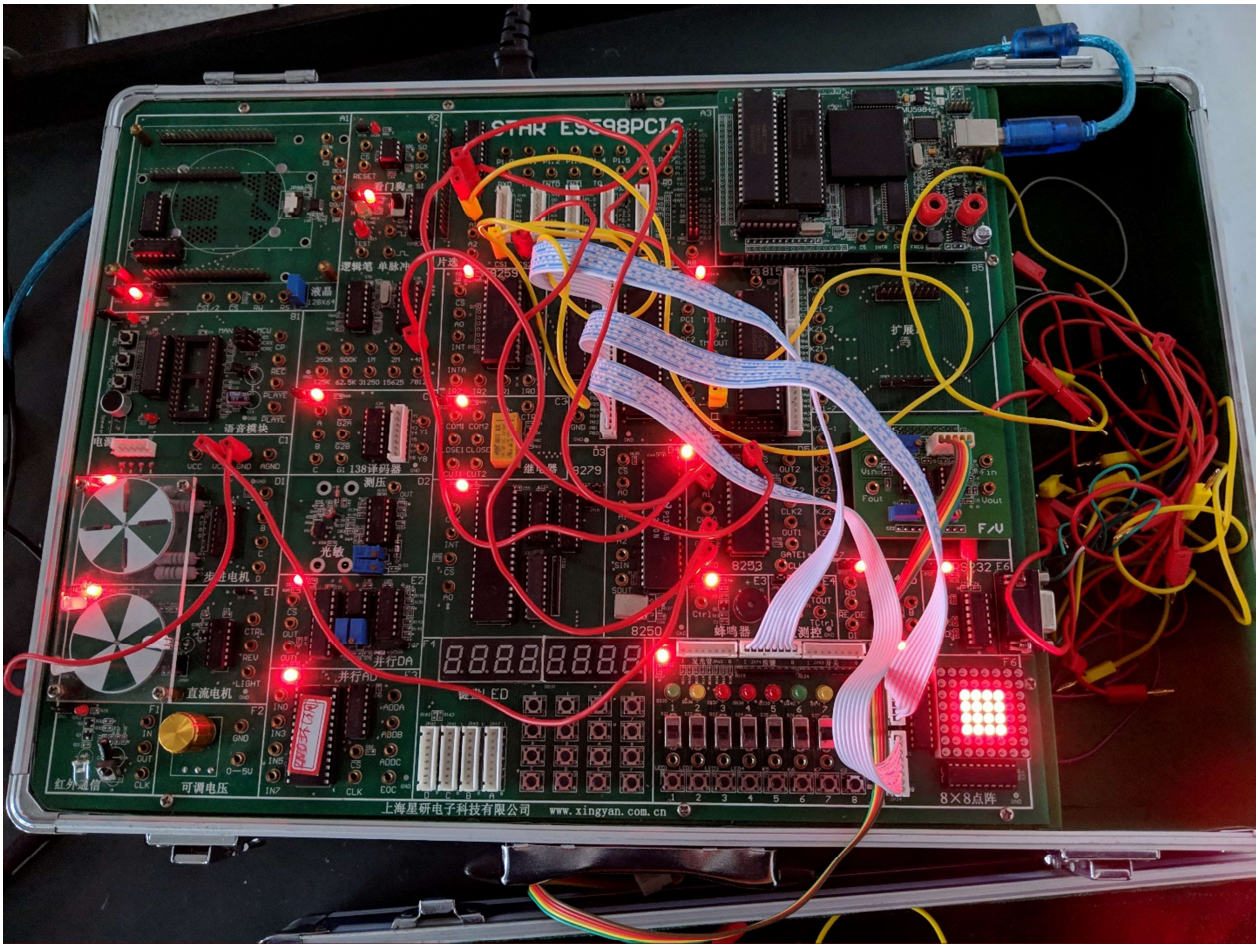
实验器材：

1. 星研实验箱
2. 星研集成开发环境

连接图：



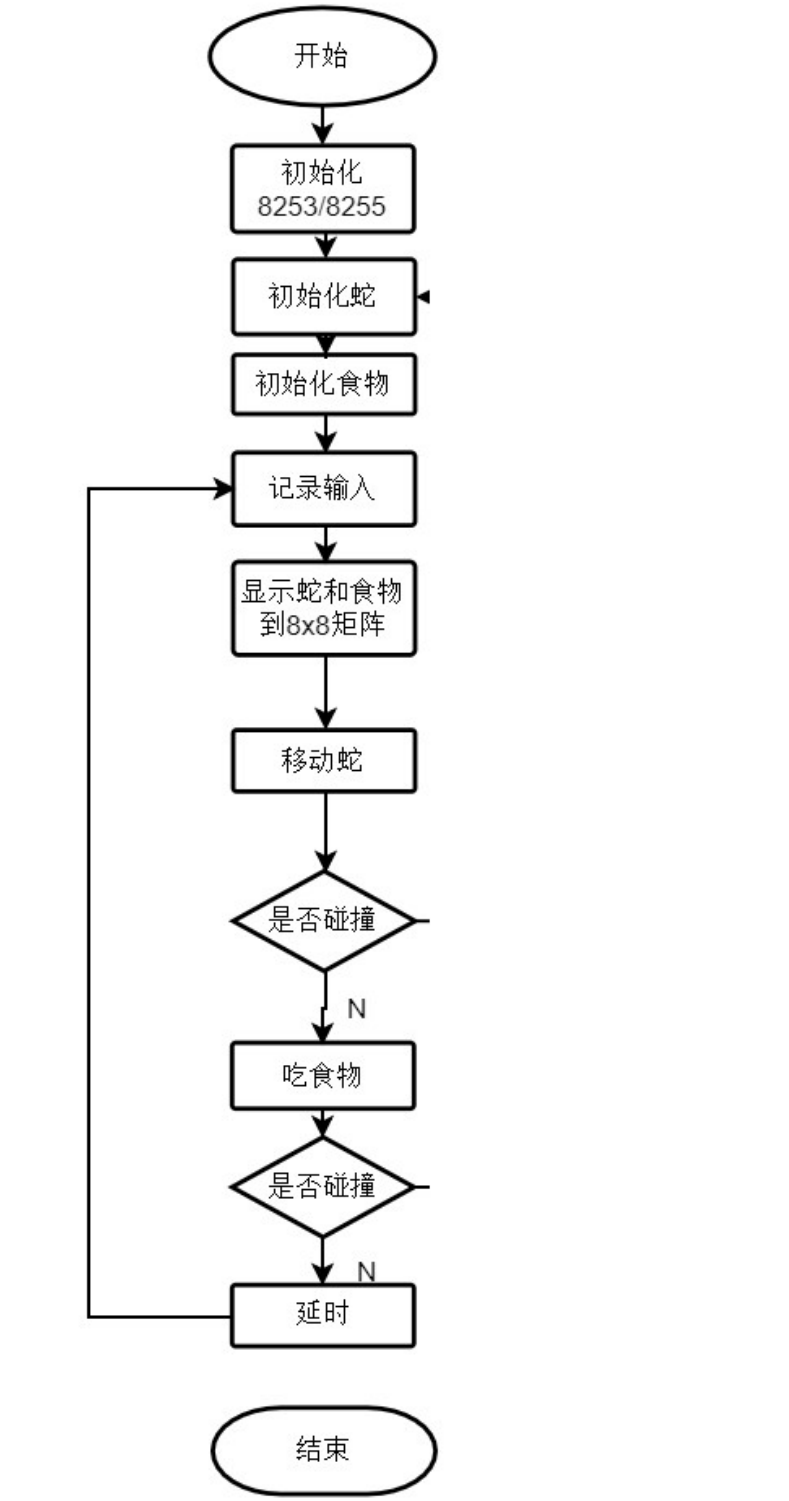
实物连接图：



连线方式：

8253		8255	
A0	片选 A0	A0	片选 A0
A1	片选 A1	A1	片选 A1
CLK0	4m clock	PB	JP24
Gate0	VCC	PC	JP23
CS	片选 CS2	CS	片选 CS1
		PA	JP74

流程图：


开始
初始化
8253/8255
初始化蛇
初始化食物
记录输入
显示蛇和食物
到8x8矩阵
移动蛇
是否碰撞
N
吃食物
是否碰撞
N
延时
结束

代码：

```
.model  tiny
        ;8255 control
        com55_add   equ     0f003h
        pa55_add    equ     0f000h
        pb55_add    equ     0f001h
        pc55_add    equ     0f002h
        ;8253 control
        com53_add   equ     0e003h
        out53_add   equ     0e000h

.stack 100
.data
        alterd      db      0
        ranfood     dw      ?
        inkey       db      ?
        snake       db  33  dup(0)
        dir         db  18  dup(0)
        snakelen    db      2
        up          db      1
        left        db      2
        down        db      3
        right       db      4
        ;end         db      8
        die         db      0
        rowmat      db      0, 01111111b, 10111111b,
11011111b, 11101111b, 11110111b, 11111011b, 11111101b,
11111110b
        colmat      db      0, 10000000b, 01000000b,
00100000b, 00010000b, 00001000b, 00000100b, 00000010b,
00000001b
        headcount   db       0
        seed            db  10
.code
start:  mov     ax, @data
        mov     ds, ax
        nop
        ;resurrection:
```

```
        ; initialize 8255
        ; use 8255 to get direction and control 8x8 led
matrix
        ; cs - connect to cs1
        ; use pa as input  - connect to sec f5 keyboard
        ; use pb as output - connect to jp24 (0 to select
row -)
        ; use pc as output - connect to jp23 (1 to select
col |)
        mov     al, 90h
        mov     dx, com55_add
        out     dx, al
        ; initialize 8253
        ; use 8253 to get the seed of the random number
        ; use Middle Square Weyl Sequence RNG method to
        ; get the random number from 1 to 8

        ; cs - connect to cs2
        ; gate0 - high (vcc)
        ; clk0 - clock ()
        mov     dx, com53_add
        mov     al, 14h
        out     dx, al
        mov     dx, out53_add
        mov     al, 8
        out     dx, al
        call    initSnake ; done
       call    getfood ;done
             mov       ax, 0407h
        mov     ranfood, ax
        resnake: ;unclear

        call    input
        call    display1
        call    snakeMove
        call    changdir

        call    hitsome
        call    snakeeater
```

```
        cmp     die,1
        jne     gonodie
        call    endgame
        gonodie: call    delay2
        jmp     resnake

        quit:   mov     ax, 4c00h
        int     21h
changdir     proc
        xor     ah, ah
        mov     al, snakelen
        mov     cx, ax
        mov     al, 2
        mov     si, ax
       cha: mov     bh, dir[si]
        dec     si
        mov     dir[si], bh
        inc     si
        inc     si
        loop    cha

        ret
changdir     endp
display1     proc
        mov     bx, ranfood
        call dispone
        xor ax,ax
        mov     al, snakelen
        mov     cx, ax
        add     al, al
        mov     si, ax
  redisp:      mov     bh, snake[si]
        dec     si
        mov     bl, snake[si]
        dec     si
        call dispone
        loop    redisp
        ; ====================
        ; NOT SURE
```

```
        mov     dx,pb55_add
        mov     al, 0ffh
        out     dx, al
        mov     dx, pc55_add
        mov     al, 0
        out     dx, al
        ; ===================
        ret
display1     endp
dispone     proc
        push bx
        push dx
        push cx
        push ax
        push si
        ;the point is at bh-high,bl-low

        xor     ax, ax
        mov     al, bh
        mov     si, ax
        mov     al, rowmat[si]
        mov     dx, pb55_add
        out     dx, al

        xor     ax, ax
        mov     al, bl
        mov     si, ax
        mov     al, colmat[si]
        mov     dx, pc55_add
        out     dx, al
        ;
        mov     al, 11111111b
        mov     dx, pb55_add
        out     dx, al
        mov     al, 0b
        mov     dx, pc55_add
        out     dx, al
        pop     si
        pop     ax
```

```
        pop     cx
        pop     dx
        pop     bx
        ret
dispone     endp
endgame     proc
        mov     al, 11000011b
        mov     dx, pb55_add
        out     dx, al
        mov     al, 00111100b
        mov     dx, pc55_add
        out     dx, al
        still:

        jne     still
        ;jmp     resurrection

        ret
endgame     endp
snakeeater  proc
        xor     ax,ax
        mov     al, snakelen
        add     al, al
        mov     si, ax
        mov     bh, snake[si]
        dec     si
        mov     bl, snake[si]   ; snake head at bx
        mov     ax, ranfood
        cmp     ax,bx
        jne     snout1
        mov     al, snakelen
                inc     al
                mov     snakelen, al
                xor     ah,ah
                mov     si, ax

                mov     ah, dir[si]
                cmp     ah, up
                je      upSnake1
```

```
                cmp     ah, down
                je      doSnake1
                cmp     ah, left
                je      leSnake1
                cmp     ah,right
                je      riSnake1
        upSnake1:mov    ax, ranfood
         xor     bx,bx
                mov     bl, snakelen

                add     bl,bl
                mov     si, bx
                dec     ah
                cmp     ah, 0
                jnz     upk1
                mov     die, 1
                upk1: mov     snake[si], ah
                dec si
                    mov     snake[si],al
                    call    getfood
                jmp     snout

        doSnake1:mov    ax, ranfood
         xor     bx,bx
                mov     bl, snakelen

                add     bl,bl

                mov     si, bx
                inc     ah
                cmp     ah, 9
                jnz     dok
                mov     die, 1
                dok: mov     snake[si], ah
                    dec si
                    mov     snake[si],al
call    getfood
              snout1:  jmp     snout
```

```
leSnake1:mov    ax, ranfood
   xor     bx,bx
        mov     bl, snakelen

        add     bl,bl

        mov     si, bx
        dec     al
        cmp     al, 0
        jnz     lek
        mov     die, 1
        lek: mov     snake[si], ah
        dec si
            mov     snake[si],al
            call    getfood
        jmp     snout

riSnake1:mov    ax, ranfood
   xor     bx,bx
        mov     bl, snakelen

        add     bl,bl

        mov     si, bx
        inc     al
        cmp     al, 9
        jnz     rik
        mov     die, 1
        rik:mov     snake[si], ah
                dec si
            mov     snake[si],al
            call    getfood
        jmp     snout
snout:      xor ax, ax
            mov         al, snakelen
            inc         al
            mov si, ax
            mov         al, inkey
            mov         dir[si], al
```

```
                ret
snakeeater  endp

hitsome     proc
        cmp     die, 1
        je      aftermath
        xor     ax,ax
        mov     al, snakelen
        add     al, al
        mov     si, ax
        mov     bh, snake[si]
        dec     si
        mov     bl, snake[si]   ; snake head at bx
        xor     ax,ax
        mov     al, snakelen
        dec     al
        mov     cx, ax
        ;inc     al
        add     al,al
        mov     si, ax
        hitre:  mov     dh, snake[si]
                dec     si
                mov     dl, snake[si]
                cmp     dx, bx
                je      aftermath
                dec     si
                loop    hitre

        ret
        aftermath:
        call    endgame
hitsome     endp

snakeMove   proc
        mov     al, snakelen
        xor     ah,ah
        mov     si, ax
        mov     ah, dir[si]
        mov     al, inkey
```

```
        cmp     al, 0
        je      same
        cmp     al, ah
        je      same
        sub     ah,al
        cmp     ah, 0feh
        je      same
        cmp     ah, 02h
        je      same
        mov     dir[si], al
        ; 若蛇头与原方向相同，则不变，若蛇头与输入键位差为 2，则不变；其他情况，用 inkey 来替代蛇头的方向
        same:   mov     al, dir[si]
        inc     si
        mov     dir[si], al
        call    change

        ret
snakeMove   endp

change      proc
        mov     al, snakelen
        xor     ah, ah
        mov     si, ax
        mov     cx, ax
        neS:    mov     si, cx
                mov     ah, dir[si]
                cmp     ah, up
                je      upSnake
                cmp     ah, down
                je      doSnake
                cmp     ah, left
                je      leSnake
                cmp     ah,right
                je      riSnake
        upSnake:mov     al, cl
                add     al, al
                xor     ah, ah ; si is changed
```

```
                mov     si, ax
                mov     ah, snake[si]
                dec     ah
                cmp     ah, 0
                jnz     upok
                mov     die, 1
                upok: mov     snake[si], ah
                jmp     nextStage

        doSnake:mov     al, cl
                add     al, al
                xor     ah, ah ; si is changed
                mov     si, ax
                mov     ah, snake[si]
                inc     ah
                cmp     ah, 9
                jnz     dook
                mov     die, 1
                dook: mov     snake[si], ah
                jmp     nextStage
        neS1: jmp neS
        leSnake:mov     al, cl
                add     al, al
                xor     ah, ah ; si is changed
                mov     si, ax
                dec     si
                mov     ah, snake[si]
                dec     ah
                cmp     ah, 0
                jnz     leok
                mov     die, 1
                leok: mov     snake[si], ah
                jmp     nextStage

        riSnake:mov     al, cl
                add     al, al
                xor     ah, ah ; si is changed
                mov     si, ax
                dec     si
```

```
            mov     ah, snake[si]
            inc     ah
            cmp     ah, 9
            jnz     riok
            mov     die, 1
            riok:mov     snake[si], ah
            jmp     nextStage
        nextStage:

        ;         mov     al, headcount
        ;         cmp     al, 1
        ;         je      chakt
        ;         mov     al, 1
        ;         mov     headcount, al
        ;         mov     al, snakelen
        ;         add     al, al
        ;         inc     al
        ;         xor     ah, ah
        ;         mov     si, ax
        ;         mov     ah, snake[si]
        ;         cmp     ah, 0
        ;         je      nohead
                ; mov     al,alterd
                ; cmp     al, 1
                ; je      already
                ; mov     al, cl
                ; xor     ah,ah
                ; mov     si, ax
                ; inc     si
                ; mov     ah, dir[si]
                ; dec     si
                ; mov     al, dir[si]
                ; cmp     ah, al
                ; je      already
                ; mov     dir[si], ah
                ; dec     al
                ; xor     ah, ah
                ; mov     si, ax
```

```
                ; already:
                loop    neS1
                ret
change      endp

initSnake   proc
        mov     al, snakelen
        add     al, al
        xor     ah, ah
        mov     si, ax

        mov     snake[si], 4
        dec     si
        mov     snake[si], 4
        dec     si
        mov     snake[si], 4
        dec     si
        mov     snake[si], 3

        mov     al, snakelen
        xor     ah,ah
        mov     si,ax
        inc     si
        mov     ah, right
        mov     dir[si], ah
        dec     si
        mov     dir[si], ah
        dec     si
        mov     dir[si], ah
        ret
initSnake   endp ;check

input       proc
                ; use f5 sec to input direction
                ; 1 for up   2 for down
                ; 3 for left 4 for right
                ; 0 for no input
                ; result is in the inkey variable
                ;
```

```
                push    ax
                push    cx
                push    dx

                mov     dx, pa55_add
                in      al, dx
                cmp     al, 0ffh
                je      noput           ;no input jump out
          call    delay1
                in      al, dx
                cmp     al, 0ffh
                je      noput
                mov     cx, 8
                xor     ah, ah
                mov     ah, al
                xor     al, al
        in1:    rol     ah, 1
                inc     al
                jnc     in2
                jmp     in1
        in2:    mov     inkey, al
                jmp     in3
        noput:  mov     inkey, 0
        in3:    pop     ax
                pop     cx
                pop     dx
                ret
input       endp;check

getfood     proc
                ; generate food for the snake to eat
                ; use random number to generate
                ; check if the food is interference with the
snake
        rerand: call    rcrand
                mov     ax, ranfood
                call    collBody
                cmp ax, 0ffh
                je      rerand
```

```
                ret

getfood     endp

collBody    proc
                ; use ax to prealloc point
                ; use the point to pair with snake(without
head)
                ; if collide, set ax = 0ffh
                mov     bl, snakelen
                dec     bl
                mov     cl, bl
                xor     ch,ch
                add     bl, bl
                xor     bh,bh
                mov     si, bx
        collre: mov     bh, snake[si]
                dec     si
                mov     bl, snake[si]
                dec     si
                cmp     ax,bx
                je      coll
                loop    collre
                jmp     nocoll
        coll:   mov     ax, 0ffh
        nocoll:         ret
collBody    endp

random      proc
        ; generate random number from 1-8

        mov     al, 00010100b
        mov     dx, com53_add
        out     dx, al
        mov     dx, out53_add

        mov     ax, 08h
        out     dx, al
```

```
        mov     al, ah
        out     dx, al
        mov     al, 00h
        mov     dx, com53_add
        out     dx, al

        mov     dx, out53_add
        in      al, dx
        mov     ah, 0
        mov     bl, 8
        div     bl
        cmp     ah, 0
        jne     noadd

        noadd:
        mov     al, ah
             ret
random      endp

rcrand      proc
        push    ax
        push    bx
        call    random
        xor     ah, ah
        mov     ranfood, ax
        call    random
        mov     bx, ranfood
        mov     ah, bl
        mov     ranfood, ax
        ; check if the generated point is interference with
the snake
        pop     bx
        pop     ax
        ret
rcrand      endp

delay1  proc
        push    cx
        mov     cx, 1
```

```
lo1:    push    cx
        mov     cx, 10
lo2:    loop    lo2
        pop     cx
        loop    lo1
        pop     cx
        ret
delay1  endp

delay2  proc

        push    cx
        mov     cx, 1000
lo3:    push    cx
         call    display1
        mov     cx, 10000
lo4:   loop    lo4
        pop     cx
        loop    lo3
        pop     cx

        ret

       ; mov     cx, 2
       ; del: call    display1
       ; call    delay1
       ; loop    del
       ; ret
delay2  endp
        end     start
```