微机系统与接口实验-第三次实验报告

学号: 21210710

姓名: 宋斌

测试题目:

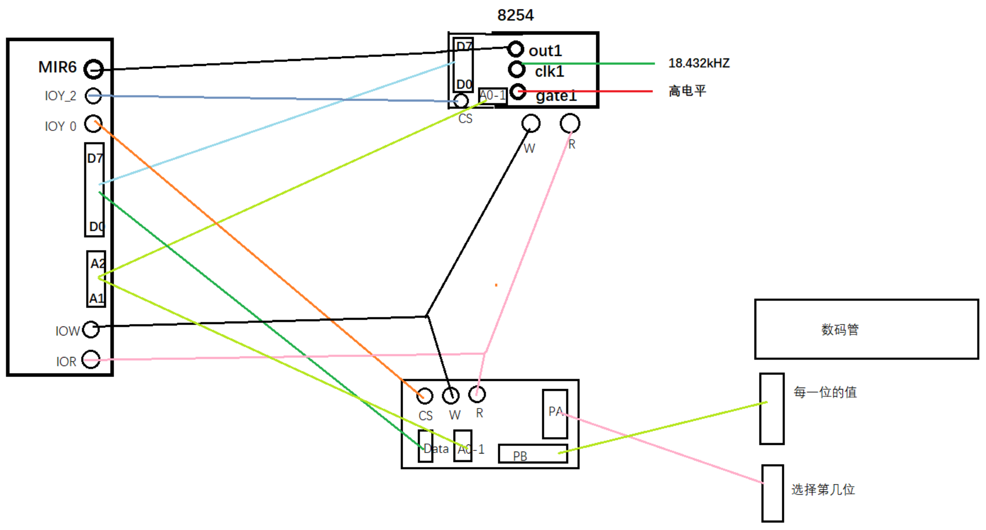
使用定时器控制数码管显示：初始时，在数码管的最右一位上显示“0”;每隔1秒钟（8254定时），数码管上显示的数字加1;加到“9”之后，下一秒从“0”重新开始，重复上述过程。

实验原理:

1. 8254
2. 8259
3. 8255
4. 数码管

连线:

(大致是这样的)

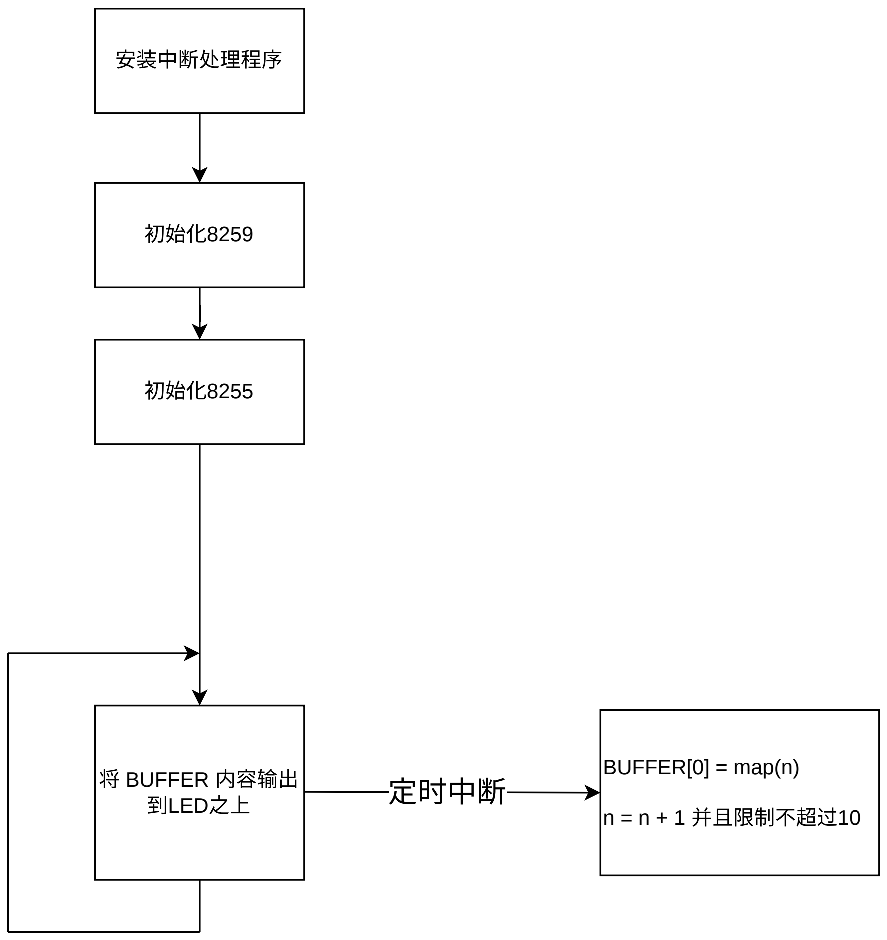


基本原理:

在中断处理程序里面给n 加1,然后转为 数码管字符放到 缓冲区里面即可。

主函数内一直输出 缓冲区的内容。

程序框图:



程序代码:

```
DATA    segment
        NumArray db 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH,77H,7CH,39H,5Eh,79H,71H

        last_pressed_key dw 00h

        ;save read numbers from keyboard.
        ;0xff means to display none.
        numbers db 00H,0FFh,0FFh,0FFh,0FFh,0FFh

        ;display buffer
        LED_BUFFER db 00h,00h,00h,00h,00h,00h
DATA    ends

CODE    SEGMENT
      ASSUME CS:CODE,DS:DATA



delay:
    push cx
```

```
  count:
    test di,di
    jz delay_end

        mov cx,01FFh

    delay_loop:
      test cx,cx
          jz delay_loop_end
          dec cx
          jmp delay_loop

    delay_loop_end:

    dec di
    jmp count

  delay_end:
    pop cx
    ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; interruption handler ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
irq6_handler:
    push bp
    mov bp,sp

    push ax


    ;convert numbers to led string.
    lea ax,[LED_BUFFER]
        push ax
        mov ax,ds
        push ax

        mov ax,6
        push ax

        lea ax,ds:[numbers]
        push ax

        mov ax,ds
        push ax
        call numstr2ledstr
        add sp,0Ah
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    mov al,[numbers]
    inc al
    cmp al,0Ah
    jb _save_numbers
    xor ax,ax   ;al >= 10.

  _save_numbers:
    mov [numbers],al

    ;
    pop ax
    mov sp,bp
    pop bp

    iret


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; intialization ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
install_interruption_handlers:
    push es

    xor ax,ax
    mov es,ax

      mov ax, OFFSET irq6_handler
    mov si, 0038H
    mov es:[si], ax
    mov ax, cs
    mov si, 003AH
    mov es:[si], ax


    pop es
    ret


init_8259:
        ;init 8259A
    cli

    mov al, 11H
    out 20H, al       ;ICW1

    mov al, 08H
    out 21H, al       ;ICW2

    mov al, 04H
    out 21H, al       ;ICW3

    mov al, 03H
    out 21H, al       ;ICW4


    mov al, 2FH       ;OCW1
    out 21H, al

    sti
    ret


init_8255:
        ;init 8255
    mov dx,606h
        mov al,89h
        out dx,al
        ret


init_timer:
    mov dx,0686h      ;counter 1
```

```
        mov al,076h
        out dx,al

        ;
    mov dx,0682h
        mov al,00h
        out dx,al

        mov al,48h
        out dx,al

    ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;display functions ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
display_ledchar:            ;display_char(led_idx,led_char)
        push bp
    mov bp,sp

    push cx
    push dx

    xor ax,ax
    mov ax,20h

    ;
    mov cx,[bp + 04h]   ;select ...
    shr ax,cl
    not al

    mov dx,600h
    out dx,al

    ;out number
    mov cx,[bp + 06h]
    mov al,cl

    mov dx,602h
    out dx,al

    pop dx
    pop cx
    pop bp
    ret

num2ledidx:     ; num2ledidx
        push bp
        mov bp,sp

        push si
        push bx

        xor ax,ax
        mov bx,[bp + 4]
        cmp bx,0FFh
        jnz __get_led_value_by_idx
        xor ax,ax
        jmp __num2ledidx_ret
```

```
 __get_led_value_by_idx:
        lea si,[NumArray]
        mov al,[si + bx]
    jmp __num2ledidx_ret

__num2ledidx_ret:
        pop bx
        pop si

        mov sp,bp
        pop bp

        ret


display_ledstr:            ;display_str(uint8_t * led_value,uint16_t len)
        push bp
        mov bp,sp

        sub sp,20h
        push bx
        push cx
        push si

        ;i = 0
        xor ax,ax
        mov [bp - 02h],ax


        mov es,[bp + 04h]   ;es = seg
        mov si,[bp + 06h]   ;si = offset.
        mov cx,[bp + 08h]   ;cx = len


     _display_str_loop_body:
        mov bx,[bp - 02h]
        cmp bx,cx
        jz  _display_str_loop_end
        ;

        ;display_char(i,led_value[i])
        mov bx,es:[si + bx]
        push bx         ;led_value

        mov bx,[bp - 02h]
        push ax         ;led_idx

        call display_ledchar
        add sp,04h

        ;i ++
        mov ax,[bp - 02h]
        inc ax
        mov [bp - 02h],ax


        ;delay(01h)
        mov di,01h
    call delay
```

```
        jmp _display_str_loop_body

     _display_str_loop_end:


        pop si
        pop cx
        pop bx

        mov sp,bp
        pop bp
        ret


numstr2ledstr:      ; numstr2ledstr(uint8_t * seg:in, word  len , uint8_t * seg:out)
        push bp
        mov bp,sp
        ;;
        sub sp,20h
        push bx
        push si
        push di
        push cx
        ;;
        mov es,[bp + 04h]
        mov si,[bp + 06h]

        mov cx,[bp + 08h]

        mov es,[bp + 0Ah]
        mov di,[bp + 0Ch]

        ;i = 0
        xor ax,ax
        mov word ptr [bp - 02h],ax

        ;while(i < cx)

   numstr2ledstr_loop_body:
        mov bx,[bp - 02h]
        cmp bx,cx
        jz numstr2ledstr_loop_end

        ;
        xor ax,ax
        mov es,[bp + 04h]
        mov al,byte ptr es:[si + bx]
        push ax
        call num2ledidx
        add sp,02h
        ;;
        ;out[idx] = ax
        mov bx,[bp - 02h]
        mov es,[bp + 0Ah]
        mov byte ptr es:[di + bx],al


        ;i ++
        mov ax,[bp - 02h]
        inc ax
        mov [bp - 02h],ax

        jmp numstr2ledstr_loop_body
   numstr2ledstr_loop_end:



        ;;;
        pop cx
        pop di
        pop si
        pop bx
        ;;
        mov sp,bp
        pop bp
        ret

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; display function ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
display_buffer:
  push bp
  mov bp,sp

  push si
  lea si,[LED_BUFFER]

  ;display numbers
    mov ax,6
    push ax

    push si

    mov ax,ds
    push ax

    call display_ledstr
    add sp,06h
  ;;

  pop si

  mov sp,bp
  pop bp
  ret



main:
    push bp
    mov bp,sp
    sub sp,20h  ;local vars..

    call install_interruption_handlers
    call init_8259
    call init_8255
    call init_timer

  loop_:
    call display_buffer
    mov di,01h
    call delay
    jmp loop_

  loop_end:
    mov sp,bp
    pop bp
    ret


start:
        ;save registers
        push ax
    push cx
    push dx
    push bx
    push si
    push di
    push ds
    push es
        pushf

        ;set new ds.
        mov ax,DATA
        mov ds,ax

        ;call main function.
    call main

    popf
    pop es
    pop ds
    pop di
    pop si
    pop bx
    pop dx
    pop cx
    pop ax

    mov ax,4c00h
    int 21h

code  ends
    end  start
```