

**微机课程设计实验报告**

班级：计算机科学与技术183班

姓名： 杜博奇

学号： 201805020121

教师： 舒红波

目录

[1. 程序总体介绍 3](#_Toc62244284)

[2. 大体功能模块 3](#_Toc62244285)

[3. 运用到的所学知识 3](#_Toc62244286)

[4. 操作流程 3](#_Toc62244287)

[5. 心得与体会 7](#_Toc62244288)

[6. 附件（源代码） 8](#_Toc62244289)

## 程序总体介绍

这个程序主题是一个方框和文字，文字可按键改变颜色，并可在方框中移动。当文字的边界与边框发生碰撞时，文字消失，出现警告的感叹号。按下反向键，回到原位，文字显示，感叹号消失

## 大体功能模块

写字：利用（0,1）点阵，0的颜色为背景色（全黑），1的颜色可更改。利用bx，

Si两个便宜量来确定画点的位置，用bp，si来确定点阵中的元素。

变色：传入不同颜色，在原位置重绘。

移动：修改位置，将原位置的图像用全‘0’点阵覆盖，在新的位置再绘。

碰撞警告：设置变量flag，当位置达到临界值时，全‘0’点阵覆盖原文字，并在指定位置画感叹号，flag变成1，当反向移动时，全‘0’点阵覆盖感叹号并绘画文字。Flag用作这些绘画的判定。

## 运用到的所学知识

键盘的中断：用点阵写字，设置颜色，画直线。

## 操作流程

文字在框中显示，可以按键（a,b）改变颜色，可以按（jlik）键改变方向，在文字外边框触碰到绿色框框的时候，文字会消失，旁边会出现感叹号警告。反向按键后，警告消失文字出现。

line: ;画线

mov bx,[di]

mov ax,[di+2]

mov cx,[di+4]

mov dx,320

mul dx

add bx,ax

add cx,bx

x1ine: mov byte [es:bx],2

inc bx

cmp bx,cx

jb x1ine

ret

writeword0: ;写字

push ax

push si

push bx

push bp

push cx

mov bp,0

mov cx,0

mov ax,[Y]

mov bx,320

mul bx

add ax,[X]

mov bx,ax

x1:

add bx,cx

mov si,0

x0: cmp si,15

ja xend0

mov al,[wd0+bp+si]

mov [es:bx+si],al

inc si

jmp x0

xend0: add bp,16

mov cx,320

cmp bp,255

ja xend

jmp x1

xend:

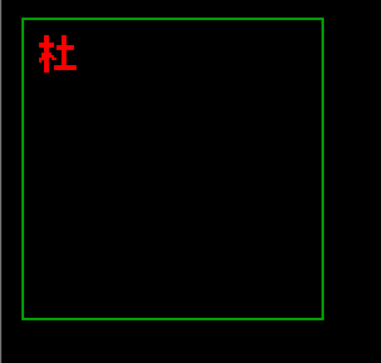
pop cx

pop bp

pop bx

pop si

pop ax

 ret

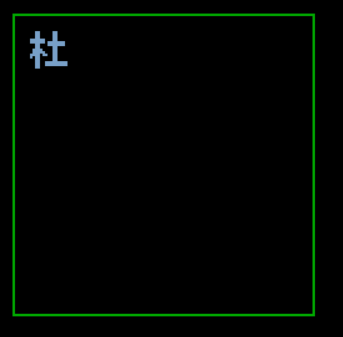
xa: cmp al,0x1e ;变色

jne xb

mov si,color1

call makecolor

call writeword0

 jmp k1

xup: cmp al,0x17 ;文字移动与碰撞警告。

jne xdown

call moveword

sub word[Y],1

cmp word[Y],9

jne up0

call moveword

call warning

mov word[flag],1

jmp k1

up0: call writeword0

cmp word[flag],0

je upd

call warning1

mov word[flag],0

upd: jmp k1

xdown: cmp al,0x25

jne xleft

call moveword

add word[Y],1

cmp word[Y],111

jne down0

call moveword

call warning

mov word[flag],1 ;flag为1表示碰撞发生

jmp k1

down0: call writeword0

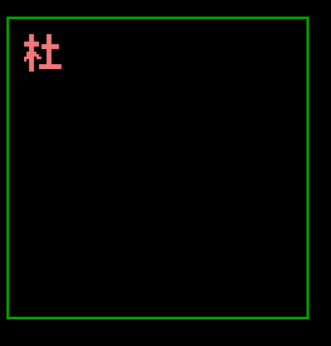
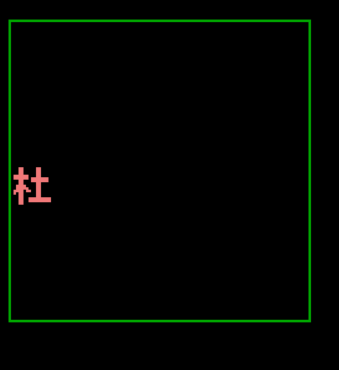
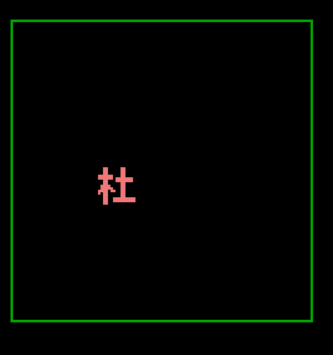
cmp word[flag],0 ;flag为0表示碰撞解除（或者未发生）

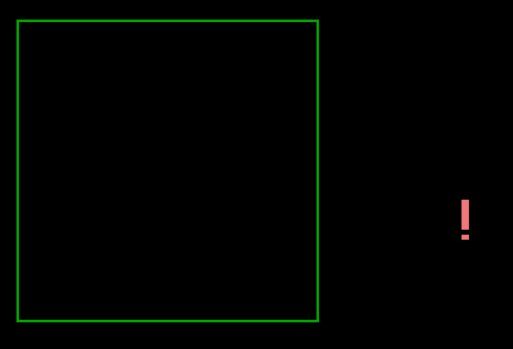
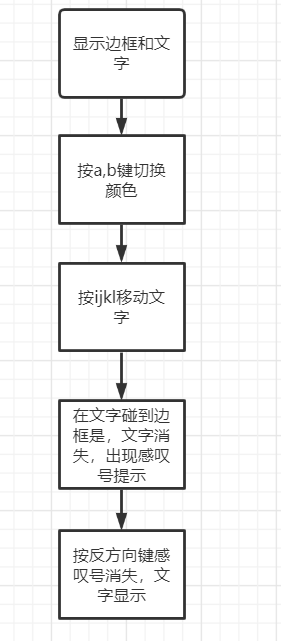
je downd

call warning1

mov word[flag],0

downd: jmp k1





## 心得与体会

本次课设的制作是对平时学习的知识的一种大汇总，用到了键盘中断，写汉字，画线等技术，但仅仅是吧平时的东西简单糅合是不足以完成我的所有功能的，比如变色和碰撞警告，需要更多的逻辑考虑，且汇编的逻辑和高级语言有些不同，处理起来要复杂一些。总的来说，这次课设让我对nasm的编程更进一步了。

## 附件（源代码）

org 0x8400

jmp start

X dw 0

Y dw 0

x dw 180

y dw 80

t db 8

h1 dw 8,8,120

h2 dw 8,128,121

s1 dw 8,8,120

s2 dw 128,8,120

flag db 0

bgd db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

db '0000000000000000'

warn db '0000001110000000'

db '0000001110000000'

db '0000001110000000'

db '0000001110000000'

db '0000001110000000'

db '0000001110000000'

db '0000001110000000'

db '0000001110000000'

db '0000001110000000'

db '0000001110000000'

db '0000001110000000'

db '0000001110000000'

db '0000000000000000'

db '0000000000000000'

db '0000001110000000'

db '0000001110000000'

wd0 db '0011000001100000'

db '0011000001100000'

db '0011000001100000'

db '1111110001100000'

db '1111110111111100'

db '0011000111111100'

db '0011000001100000'

db '0111100001100000'

db '0111110001100000'

db '1111011001100000'

db '1011000001100000'

db '0011000001100000'

db '0011001111111110'

db '0011001111111110'

db '0011000000000000'

db '0000000000000000'

color0 db 0,0,0

color db 255,0,0

color1 db 30,40,50

color2 db 60,30,30

start:

mov AL,0x13

mov AH,0x00

int 0x10

mov ax,0x0a000;往显存(0xa0000开始)物理地址

mov es,ax

mov cx,0

mov ds,cx

mov si,color

call makecolor

mov word[ds:X],15 ;汉字左上角起点x

mov word[ds:Y],15 ;汉字左上角起点y

call writeword0

mov di,h1

call line

mov di,h2

call line

mov di,s1

call line\_v

mov di,s2

call line\_v

cli

mov word [ds:0x24],int\_key ;写入键盘中断9号服务程序地址

mov word [ds:0x26],0

sti

jmp $

makecolor:

mov al,48

mov dx,0x3c8

out dx,al

mov al,0

mov dx,0x3c9

out dx,al

mov al,0

mov dx,0x3c9

out dx,al

mov al,0

mov dx,0x3c9

out dx,al

mov al,49

mov dx,0x3c8

out dx,al

mov al,[si]

mov dx,0x3c9

out dx,al

mov al,[si+1]

mov dx,0x3c9

out dx,al

mov al,[si+2]

mov dx,0x3c9

out dx,al

ret

line:

mov bx,[di]

mov ax,[di+2]

mov cx,[di+4]

mov dx,320

mul dx

add bx,ax

add cx,bx

x1ine: mov byte [es:bx],2

inc bx

cmp bx,cx

jb x1ine

ret

line\_v:

mov bx,[di]

mov ax,[di+2]

mov cx,[di+4]

mov dx,320

mul dx

add bx,ax

mov ax,0

y1:

mov byte [es:bx],2

add bx,320

inc ax

cmp ax,cx

jb y1

ret

moveword:

push ax

push si

push bx

push bp

push cx

mov bp,0

mov cx,0

mov ax,[Y]

mov bx,320

mul bx

add ax,[X]

mov bx,ax

xx1:

add bx,cx

mov si,0

xx0: cmp si,15

ja xxend0

mov al,[bgd+bp+si]

mov [es:bx+si],al

inc si

jmp xx0

xxend0: add bp,16

mov cx,320

cmp bp,255

ja xxend

jmp xx1

xxend:

pop cx

pop bp

pop bx

pop si

pop ax

ret

warning1:

push ax

push si

push bx

push bp

push cx

mov bp,0

mov cx,0

mov ax,[y]

mov bx,320

mul bx

add ax,[x]

mov bx,ax

wx11:

add bx,cx

mov si,0

wx01: cmp si,15

ja wxend01

mov al,[bgd+bp+si]

mov [es:bx+si],al

inc si

jmp wx01

wxend01: add bp,16

mov cx,320

cmp bp,255

ja wxend1

jmp wx11

wxend1:

pop cx

pop bp

pop bx

pop si

pop ax

ret

warning:

push ax

push si

push bx

push bp

push cx

mov bp,0

mov cx,0

mov ax,[y]

mov bx,320

mul bx

add ax,[x]

mov bx,ax

wx1:

add bx,cx

mov si,0

wx0: cmp si,15

ja wxend0

mov al,[warn+bp+si]

mov [es:bx+si],al

inc si

jmp wx0

wxend0: add bp,16

mov cx,320

cmp bp,255

ja wxend

jmp wx1

wxend:

pop cx

pop bp

pop bx

pop si

pop ax

ret

writeword0:

push ax

push si

push bx

push bp

push cx

mov bp,0

mov cx,0

mov ax,[Y]

mov bx,320

mul bx

add ax,[X]

mov bx,ax

x1:

add bx,cx

mov si,0

x0: cmp si,15

ja xend0

mov al,[wd0+bp+si]

mov [es:bx+si],al

inc si

jmp x0

xend0: add bp,16

mov cx,320

cmp bp,255

ja xend

jmp x1

xend:

pop cx

pop bp

pop bx

pop si

pop ax

ret

int\_key:

mov dx,0x60 ;读取键盘缓冲区数据

in al,dx

xa: cmp al,0x1e

jne xb

mov si,color1

call makecolor

call writeword0

jmp k1

xb: cmp al,0x30

jne xup

mov si,color2

call makecolor

call writeword0

jmp k1

xup: cmp al,0x17

jne xdown

call moveword

sub word[Y],1

cmp word[Y],9

jne up0

call moveword

call warning

mov word[flag],1

jmp k1

up0: call writeword0

cmp word[flag],0

je upd

call warning1

mov word[flag],0

upd: jmp k1

xdown: cmp al,0x25

jne xleft

call moveword

add word[Y],1

cmp word[Y],111

jne down0

call moveword

call warning

mov word[flag],1

jmp k1

down0: call writeword0

cmp word[flag],0

je downd

call warning1

mov word[flag],0

downd: jmp k1

xleft: cmp al,0x24

jne xright

call moveword

sub word[X],1

cmp word[X],9

jne left0

call moveword

call warning

mov word[flag],1

jmp k1

left0: call writeword0

cmp word[flag],0

je leftd

call warning1

mov word[flag],0

leftd: jmp k1

xright: cmp al,0x26

jne end

call moveword

add word[X],1

cmp word[X],111

jne right0

call moveword

call warning

mov word[flag],1

jmp k1

right0: call writeword0

cmp word[flag],0

je rightd

call warning1

mov word[flag],0

rightd: jmp end

end:

k1: mov dx,0x20 ;键盘中断结束

mov al,0x61

out dx,al ;EOI=1,中断结束

iret