中南大学
计算机原理与汇编语言
课程设计报告

题　　目　堆叠砖块

学生姓名　

指导教师　

学　　院　计算机学院

专业班级　计科 1805

2020 年 4 月

目录/含有超链接

实验报告

一.题目描述

编制程序当输入字母(大写！) S 时，开始在屏幕上码砖块，砖块的大小事先确定，当码到屏幕顶部或者敲击任意键时停止，砖块的颜色有差别。

二.任务难点分析

1.需要确定怎样显示一个砖块
2.需要确定检查键盘缓冲区的方法
3.需要找到延时的方法
4.需要确定砖块什么时候向左右码，什么时候一行码满换行
5.解决彩色显示的问题

相关资料查找

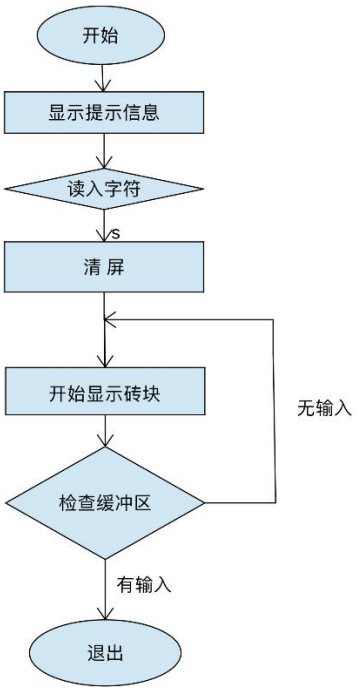
1. 实现图形变化功能，怎样画图
https://wenku.baidu.com/view/8cbfaa3764ce0508763231126edb6f1afe007117.html
2. 怎样再输入字符的时候停止
https://www.docin.com/p-766453897.html
3. 系统调用的参考
https://wenku.baidu.com/view/caa8c0edb8f67c1cfad6b84c.html
4. 扫描码表 https://blog.csdn.net/qq_37232329/article/details/79926440
5 彩色字符 https://blog.csdn.net/cgzldfend/article/details/80325125
https://blog.csdn.net/gooding300/article/details/90127513
6. 彩色显示字符串
https://blog.csdn.net/gooding300/article/details/90127513
7. 清屏 https://www.zhihu.com/question/284039672
https://www.bbsmax.com/A/8Bz8LELozx/
8. 关于如何写显存
https://www.cnblogs.com/xautxuqiang/p/5468115.html

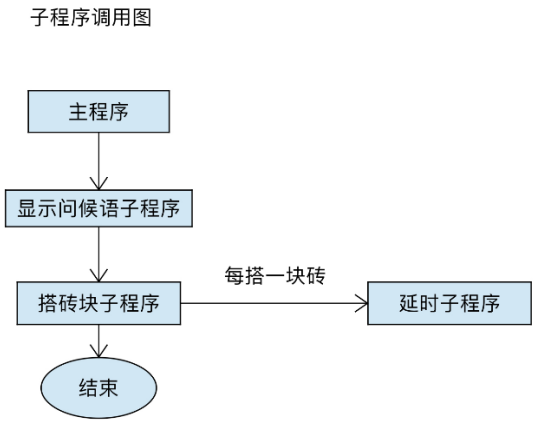
位	7	6	5	4	3	2	1	0
属性	闪烁	背景R	背景G	背景B	高亮	R	G	B

R	G	B	颜色	示例
0	0	0	黑色	示例
0	0	1	蓝色	示例
0	1	0	绿色	示例
0	1	1	青色	示例
1	0	0	红色	示例
1	0	1	粉色	示例
1	1	0	黄色	示例
1	1	1	白色	示例

通过对显存的了解，我们可以通过对 `B8000H - B8FFFH` 的内存进行修改来实现显示带有样式的字符。

三.设计原理以及流程图





开始清屏并显示问候语和提示信息
之后读入字符 S 以开始动画
清屏后调用显示砖块的子程序
调用中每显示一个砖块就调用检查是否有输入的中断以及延时子程序

四.关键功能的实现

由上文，难点在于
1.需要确定怎样显示一个砖块
2.需要确定检查键盘缓冲区的方法
3.需要找到延时的方法
4.需要确定砖块什么时候向左右码，什么时候一行码满换行
5.解决彩色显示的问题
下面是解决的办法
1.显示砖块既可以用画图的方式，也可以用字符串进行背景填充完成。本次使用后者，即字符串|_____|进行背景填充后为一个砖块
2.用-int 16h 中断
1 号调用 检查缓冲区是否有字符
3.通过-int 15h 中断（具体用法在下文）调用实现延迟
4.通过对循环的控制，cx 的压栈出栈来确定左右和上下
5.通过直接对显存的更改，实现任意位置的书写以及字符串颜色的变换

五.相关程序调用及格式

中断调用：
-int21 中断

1.字符输入存到 al 里面

```
mov ah,1

int 21h
```

-int10h 中断
3 号调用 （清屏）

```
mov ax,3h

int 10h
```

-int 16h 中断
1（检查缓冲区是否有字符）
相应的调整 zf

```
mov ah,01h

int 16h
```

-int 21h 中断
4c00（将控制权转回系统）

```
mov ax,4c00h

int 21h
```

-int 15h 中断
延时显示 用法：

```
        mov cx, 0007h

        mov dx, 0A120h      ;cx:dx 是延迟的微秒，现在的方案是 0.5 秒 16 进制字母开
头加 0

        mov ax, 0

        mov ah, 86h



        int 15h             ;中断调用
```

六.单元程序设计

1.数据段的编写

```
data segment
        db '|______|'                      ;一块砖
        ;db 00000010B
        db 00100100B
        db 01110001B
        db 00100100B
        db 11111001B                       ;4 种不同的配色方案
        db 'Welcome to Block Build!'
        db 'Created by QinYongsheng'
        db '---Press S to start----'
        db '--Press Anykey to stop-'       ;问候语
data ends
```

2.主函数的编写

```
start: mov ax,stack
        mov ss,ax
        mov sp,128

        mov ax,data                        ;数据段基地址放入 ds
        mov ds,ax
 ;-----------------------------------------清屏（清除 dos 界面的指令）
        mov ax,3h
        int 10h
 ;-----------------------------------------显示问候语
```

```
        call show_title
;-----------------------------------------输入 s 以开始
input:   mov ah,1
         int 21h
         cmp al,'S'
         jne input

;-----------------------------------------清屏（清除问候语）
        mov ax,3h
        int 10h
;-----------------------------------------开始画图
        call build

next:
        mov ax,4c00h
        int 21h
;=========================================主程序结束
```

3.问候语的子程序

```
show_title proc near            ;问候语的子程序

        mov bx,0b800h          ;显存的基地址存到 es
        mov es,bx
        mov si,12              ;问候词的起始位置

        mov bx,8               ;配色方案的起始位置
        mov di,12*160+56       ;开始写的显存位置
```

```
        mov cx,4                ;四行问候语

nextgreet:
        push cx
        push di
        mov cx,23               ;问候词字符串的长度是 23
putchar:
        mov dl,ds:[si]            ;ds:[si]里是待写的字符
        mov dh,ds:[bx]            ;ds:[bx]里是配色方案
        mov es:[di],dx            ; es:[di]  字符与颜色，写到显存
        inc si                    ;下一个字符
        add di,2                  ;下一个待写的显存位置
        loop putchar

        pop di
        pop cx
        inc bx
        add di,160                 ;换下一行写
        loop nextgreet
        ret
        show_title endp
;---------------------------------------
```

4.搭建砖块的子程序

```
;---------------------------------------
```

```
build proc near                    ;搭建砖块的子程序

        mov bx,0b800h             ;显存的基地址存到 es
        mov es,bx

        mov bx,8                  ;配色方案的起始位置
        mov di,24*160             ;开始写的显存位置
        mov cx,23                 ;总共擂 23 行
 ch_row:push cx
        mov cx,5                  ;一行放 5 快砖

 re:    push cx
        mov cx,2                  ;两块砖的时候配色方案重新开始循环

writeblock:
        push cx
        ;push di

        mov si,0                  ; 砖块字符串的开始地址
        mov cx,8                  ;一块砖有 8 个字符


writeletter:
        mov dl,ds:[si]            ;ds:[si]里是待写的字符
        mov dh,ds:[bx]            ;ds:[bx]里是配色方案
        mov es:[di],dx            ; es:[di]  字符与颜色，写到显
存
```

```
        inc si                          ;下一个字符
        add di,2                       ;下一个待写的显存位置
        loop writeletter



;--------------------------------------延时和判断是否有字符输入了
        call delay
        mov ah,01h
        int 16h
        jnz next                              ;有字符输入则回到 next
;-------------------------------------


        ;pop di
        pop cx
        inc bx
        ;add di,16                            ;因为 push 了 di，下一个写
的显存位置/已废弃
        loop writeblock


        pop cx
        sub bx,2
        loop re


        pop cx
        sub di,2*160                         ;这一行铺满了砖，写上一行
        inc bx                               ;换配色方案，使得两行之间
的配色方案可以交错
```

```
        cmp bx,10                           ;换配色方案，使得两行之间
的配色方案可以交错
        jne continue                        ;换配色方案，使得两行之间
的配色方案可以交错
        sub bx,2                            ;换配色方案，使得两行之间
的配色方案可以交错

continue:
        loop ch_row
        jmp next

        ret
        build endp
```

5.延时子程序

```
;-------------------------------------延时子程序
delay proc near
        push cx
        push dx
        push ax

        mov cx, 0007h
        mov dx, 0A120h      ;cx:dx 是延迟的微秒，现在的方案是 0.5
秒 16 进制字母开头加 0
        mov ax, 0
        mov ah, 86h
```

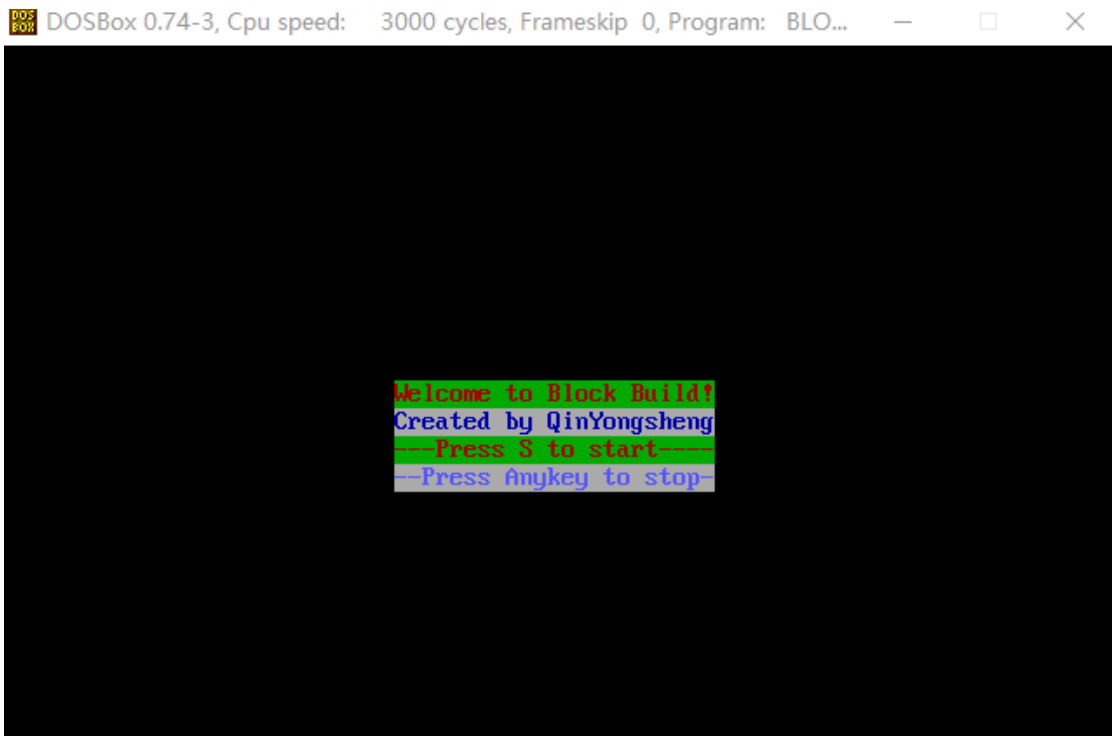
```
        int 15h             ;中断调用

        pop ax
        pop dx
        pop cx
ret
delay endp
```

七.程序编写与调试分析

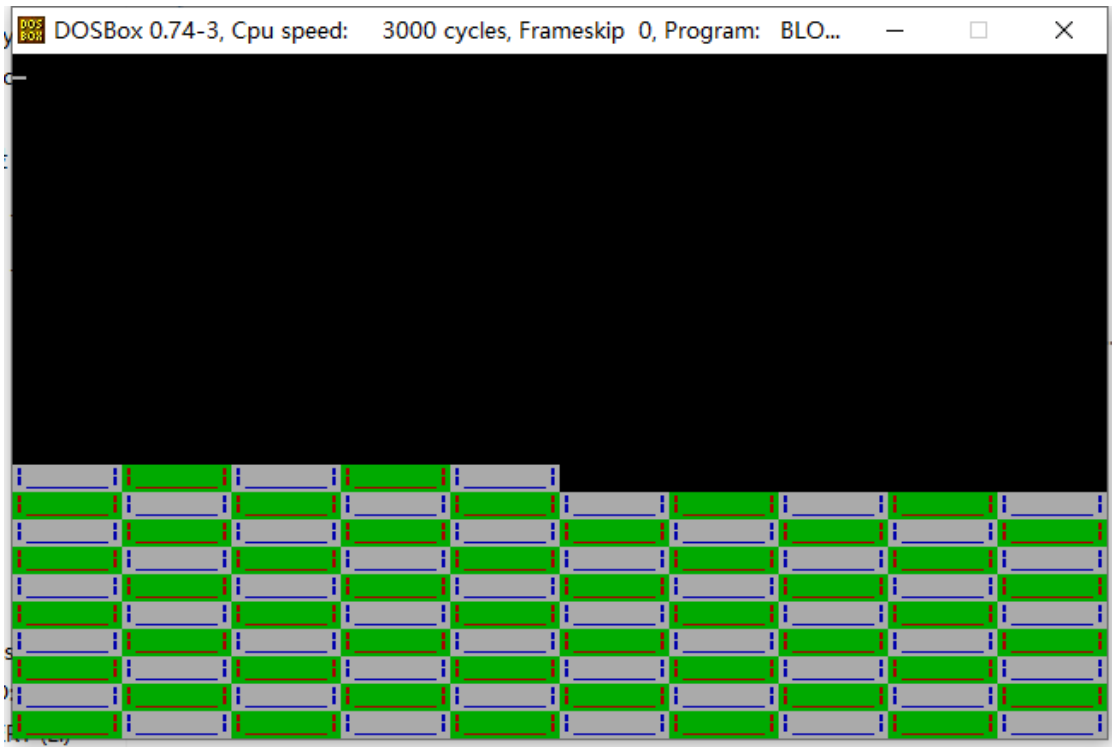
程序编写的使用使用了 emu8086 平台，对程序进行测试，汇总，汇编的时候用到了 masm 以及连接器，dosbox，平台为 win10*64

```
C:\>masm blocks.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [blocks.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51592 + 448568 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link blocks.obj

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [BLOCKS.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
```

对程序用 dosbox 以及 masm，link 进行汇编连接

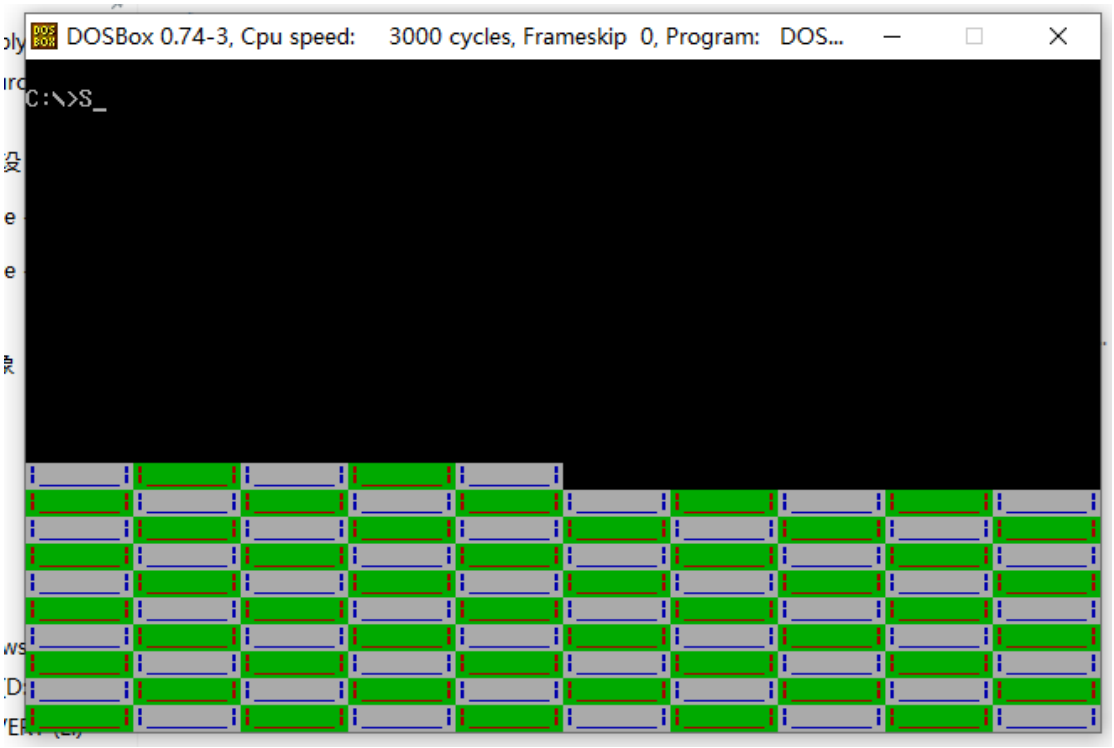


欢迎界面，显示提示语，输入 S 以开始。最后一行字有闪烁的效果



开始叠加砖块时，从左向右，从下到上，每次暂停 0.5s

输入任意字符将停止运行



八.小结

由于这个程序超出了课程教授的范围，去查了许多资料，由于没有现成的一样的程序可以直接参考，于是搜索大量的相关资料将功能逐一进行实现，还是比较有意思和成就感的。
实验过程中，都是每次需要一个功能，就去搜索相关的资料，对资料及示例代码进行研读以后做一个单独的子程序模块加到程序里。由于每一个功能相对都比较简单，而且主程序的逻辑也是比较清晰明了，所以调试基本都是一遍通过。
使用了 emu8086，相较于上学期直接 dosbox，link 来说效率与编程体验提高的不是一点半点。
下面是遇到的一些坑以及体会
1.有一些功能可以通过多种操作完成，比如清屏的功能，就有至少三种系统调用可以完成，还有一种比较麻烦，通过对显存中的每一块进行清除进行清屏。再比如，显示砖块既可以通过字符串进行拼凑，也可以直接画图。
2.十六进制字符要以字母开头的话要加 0（上学期就被这个点坑了）
3.尽量实现模块化编程（即子程序），会比较有条理，便于阅读
4.多重 loop 要对 cx 进栈出栈的顺序了如指掌
5.如果以写显存的方法实现，窗口的大小一定是固定的，不然会出现错位
6.基本功能完成后加了一些符加的功能，比如
1）每两行之间的颜色是交错显示的
2）开始时会分行显示问候语
7.汇编由于代码可读性差，所以程序流程图以及注释相当重要（所以注释我也好好写了）尤其是每一块之间用----------这种线分行非常重要

九.完整代码

最后，完整代码如下（程序以及代码也附在了压缩包中）

```
;-------------------------------------------------
;--------------砖块堆叠动画----------------------
;请将 dos 窗口设置为 80*25 的显示大小以获得正确的体验
;-------------------------------------------------


    assume cs:code,ss:stack,ds:data
;-----------------------------------------数据段
data segment
        db '|______|'                      ;一块砖
        db 00100100B
        db 01110001B
        db 00100100B
        db 11111001B                        ;4 种不同的配色方案
        db 'Welcome to Block Build!'
        db 'Created by QinYongsheng'
        db '---Press S to start----'
        db '--Press Anykey to stop-'        ;问候语
data ends
;-----------------------------------------堆栈段
stack segment
        db 128 dup (0)
stack ends
;-----------------------------------------
```

```
code segment
;=========================================主程序
 start: mov ax,stack
        mov ss,ax
        mov sp,128

        mov ax,data                          ;数据段基地址放入 ds
        mov ds,ax
 ;----------------------------------------清屏（清除 dos 界面的指令）
        mov ax,3h
        int 10h
 ;----------------------------------------显示问候语
        call show_title
;----------------------------------------输入 s 以开始
input:   mov ah,1
         int 21h
         cmp al,'S'
         jne input

;----------------------------------------清屏（清除问候语）
        mov ax,3h
        int 10h
;----------------------------------------开始画图
        call build

next:
        mov ax,4c00h
```

```
        int 21h
;=========================================主程序结束
show_title proc near            ;问候语的子程序

        mov bx,0b800h           ;显存的基地址存到 es
        mov es,bx
        mov si,12               ;问候词的起始位置

        mov bx,8                ;配色方案的起始位置
        mov di,12*160+56        ;开始写的显存位置

        mov cx,4                ;四行问候语

nextgreet:
        push cx
        push di


        mov cx,23               ;问候词字符串的长度是 23


putchar:
        mov dl,ds:[si]            ;ds:[si]里是待写的字符
        mov dh,ds:[bx]            ;ds:[bx]里是配色方案
        mov es:[di],dx            ; es:[di]  字符与颜色，写到显存
        inc si                    ;下一个字符
        add di,2                  ;下一个待写的显存位置
```

```
        loop putchar


        pop di
        pop cx
        inc bx
        add di,160                  ;换下一行写
        loop nextgreet




        ret
        show_title endp
;-------------------------------------
build proc near                     ;搭建砖块的子程序

        mov bx,0b800h              ;显存的基地址存到 es
        mov es,bx

        mov bx,8                  ;配色方案的起始位置
        mov di,24*160            ;开始写的显存位置
        mov cx,23                  ;总共擂 23 行
 ch_row:push cx
        mov cx,5                     ;一行放 5 快砖

 re:    push cx
        mov cx,2                    ;两块砖的时候配色方案重新开始循环
```

```
writeblock:
        push cx
        ;push di

        mov si,0                ; 砖块字符串的开始地址
        mov cx,8                ;一块砖有 8 个字符


writeletter:
        mov dl,ds:[si]            ;ds:[si]里是待写的字符
        mov dh,ds:[bx]             ;ds:[bx]里是配色方案
        mov es:[di],dx              ; es:[di]  字符与颜色，写到显存
        inc si                      ;下一个字符
        add di,2                   ;下一个待写的显存位置
        loop writeletter



;-------------------------------------延时和判断是否有字符输入了
        call delay
        mov ah,01h
        int 16h
        jnz next                        ;有字符输入则回到 next
;-------------------------------------

        ;pop di
        pop cx
        inc bx
```

```
        ;add di,16                          ;因为push了di，下一个写的显存位
置/已废弃
        loop writeblock


        pop cx
        sub bx,2
        loop re


        pop cx
        sub di,2*160                        ;这一行铺满了砖，写上一行
        inc bx                              ;换配色方案，使得两行之间的配色方案
可以交错
        cmp bx,10                           ;换配色方案，使得两行之间的配色方案
可以交错
        jne continue                        ;换配色方案，使得两行之间的配色方案
可以交错
        sub bx,2                            ;换配色方案，使得两行之间的配色方案
可以交错


continue:
        loop ch_row
        jmp next


        ret
        build endp
;-------------------------------------延时子程序
delay proc near
```

```
        push cx
        push dx
        push ax

        mov cx, 0007h
        mov dx, 0A120h      ;cx:dx 是延迟的微秒，现在的方案是 0.5 秒 16 进制字母开头加 0
        mov ax, 0
        mov ah, 86h

        int 15h             ;中断调用

        pop ax
        pop dx
        pop cx
ret
delay endp

code ends

end start
```