

课 程 设 计 报 告

课程名称： 微机系统与接口技术

班 级：计算机科学与技术183

学 号： 201805120512

姓 名： 何润泽

**2021年1月**

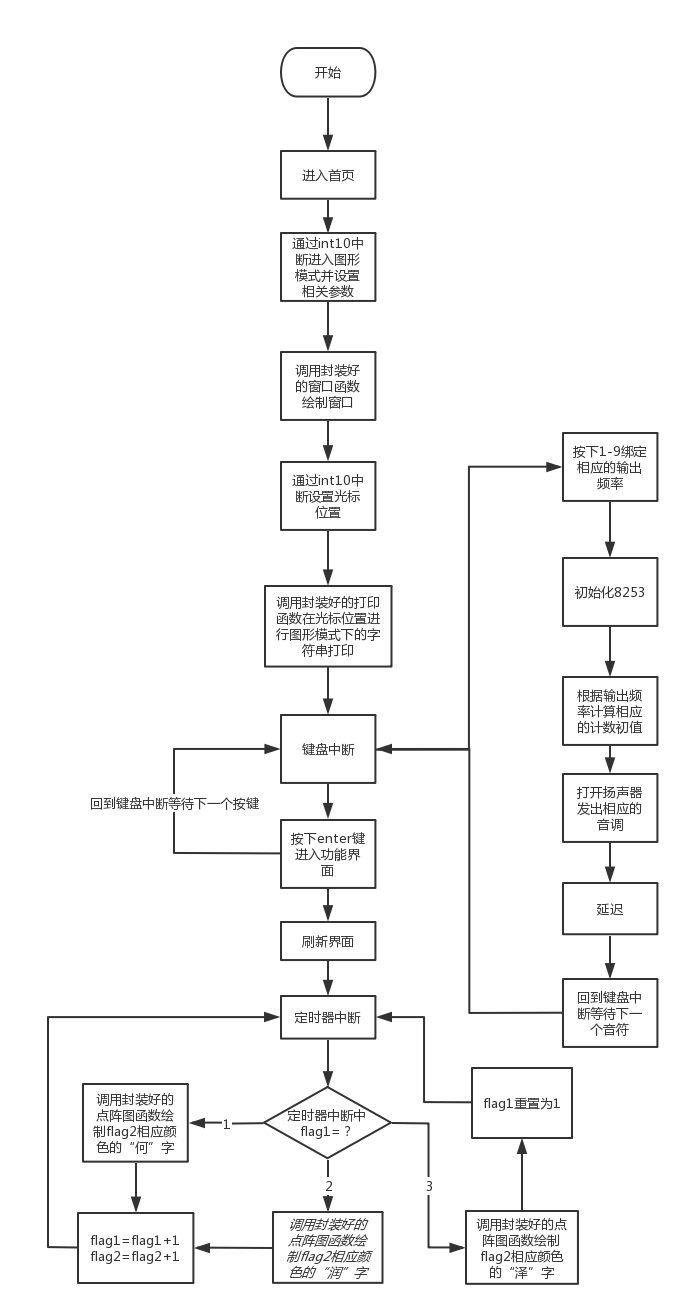
#### 功能介绍

本次课设实现的是一个在Boches环境下的多页面的多线程文字显示的音乐盒

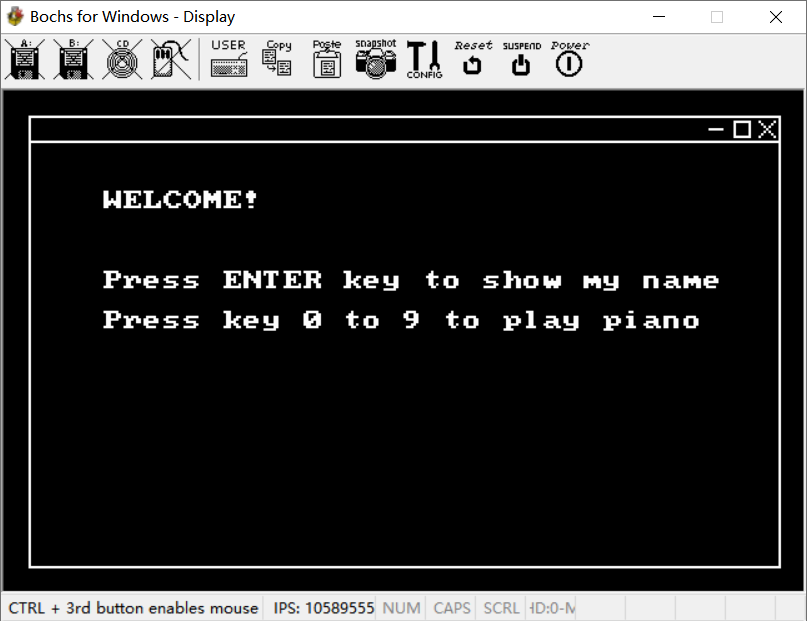
#### 所用技术

1. int 10h中断进入图形模式
2. 键盘中断
3. 定时器中断
4. 窗口绘制
5. 点阵图转文字
6. 使用8253芯片发声
7. 延迟技术
8. 光标重定位
9. 图形模式下打印字符串
10. 通过push和pop对同一个寄存器多次使用
11. 结合键盘中断和8253芯片实现电子琴
12. 结合定时器中断和点阵图转文字实现多线程动态文字

#### 3.整体实现



#### 4.具体实现



**进入程序后在图形模式下绘制一个的模拟窗口，由封装好的窗口函数实现**

;窗口绘制主函数

frame: mov ah,00h

mov al,13h

int 10h

mov ax,0x0a000

mov es,ax

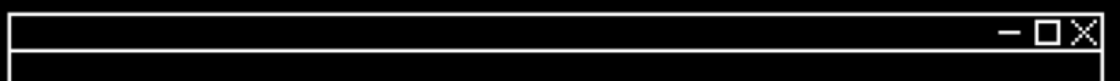
mov si,color

call dis ;窗口颜色设置

call box ;绘制方框

call guide ;绘制导航栏

ret



;导航栏函数

guide:

mov bx,10

mov ax,20

mov cx,300

call lineh

mov bx,282

;绘制最小化按钮

mov ax,15

mov cx,6

call lineh

;绘制最大化按钮

mov bx,292

mov ax,12

mov cx,6

call lineh

mov bx,292

mov ax,18

mov cx,6

call lineh

mov bx,292

mov ax,12

mov cx,6

call linev

mov bx,298

mov ax,12

mov cx,7

call linev

;绘制关闭按钮

mov bx,302

mov ax,12

mov cx,7

call slashl

mov bx,308

mov ax,12

mov cx,7

call slashr

ret

;窗口方框函数

box:

mov bx,10

mov ax,10

mov cx,300

call lineh

mov bx,10

mov ax,10

mov cx,180

call linev

mov bx,10

mov ax,190

mov cx,300

call lineh

mov bx,310

mov ax,10

mov cx,181

call linev

ret

;竖线绘制函数

linev:

mov dx,320d

mul dx

add bx,ax

x1: mov byte[es:bx],1

add bx,320d

dec cx

cmp cx,0

jne x1

ret

;横线绘制函数

lineh:

mov dx,320d

mul dx

add bx,ax

x2: mov byte[es:bx],1

inc bx

dec cx

cmp cx,0

jne x2

ret

;左上至右下斜线绘制函数

slashl:

mov dx,320d

mul dx

add bx,ax

x3: mov byte[es:bx],1

add bx,321

dec cx

cmp cx,0

jne x3

Ret

;右上至左下斜线绘制函数

slashr:

mov dx,320d

mul dx

add bx,ax

x4: mov byte[es:bx],1

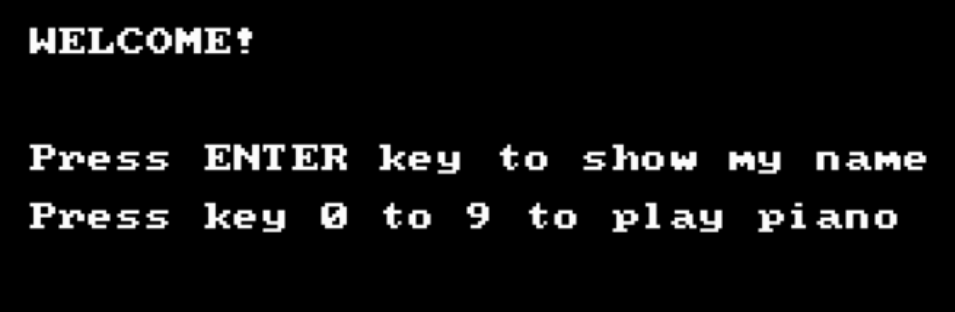
add bx,319

dec cx

cmp cx,0

jne x4

ret



**文字部分首先通过封装好的光标重定位函数设置打印的位置，然后在设置好的打印位置通过封装好的字符串打印函数实现图形模式下的字符串显示**

;打印字符串

print:

mov dh,5

mov dl,5

call setcursor

mov si,0

p1: cmp si,8

je s1

mov al,byte[ds:text1+si]

call write

inc si

jmp p1

s1: mov dh,9

mov dl,5

call setcursor

mov si,0

p2: cmp si,31

je s2

mov al,byte[ds:text2+si]

call write

inc si

jmp p2

s2: mov dh,11

mov dl,5

call setcursor

mov si,0

p3: cmp si,30

je exit1

mov al,byte[ds:text3+si]

call write

inc si

jmp p3

exit1: ret

;打印单个字符

write:

mov ah,0eh

mov bl,1

int 10h

ret

;设置光标位置

setcursor:

mov ah,02h

mov bh,0

int 10h

ret

**按下Enter键进入第二模块，此时也可以通过0-9按键进行电子琴的弹奏，这些通过键盘中断来实现**

**;键盘驱动**

int\_key:

mov dx,0x60

in al,dx

cmp al,0x9c

je enter

cmp al,0x82

je key1

cmp al,0x83

je key2

cmp al,0x84

je key3

cmp al,0x85

je key4

cmp al,0x86

je key5

cmp al,0x87

je key6

cmp al,0x88

je key7

cmp al,0x89

je key8

cmp al,0x8a

je key9

cmp al,0x8b

je key0

next: mov dx,0x20

mov al,0x61

out dx,al

call delay

in al,61h

and al,0fch

out 61h,al

iret

key1: mov bx,word[ds:frequency]

jmp play

key2: mov bx,word[ds:frequency+2]

jmp play

key3: mov bx,word[ds:frequency+4]

jmp play

key4: mov bx,word[ds:frequency+6]

jmp play

key5: mov bx,word[ds:frequency+8]

jmp play

key6: mov bx,word[ds:frequency+10]

jmp play

key7: mov bx,word[ds:frequency+12]

jmp play

key8: mov bx,word[ds:frequency+14]

jmp play

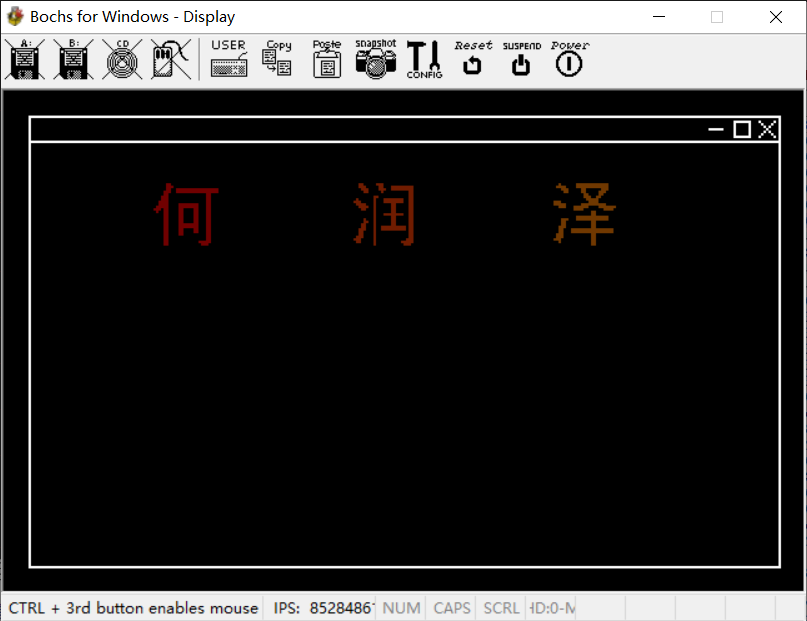
key9: mov bx,word[ds:frequency+16]

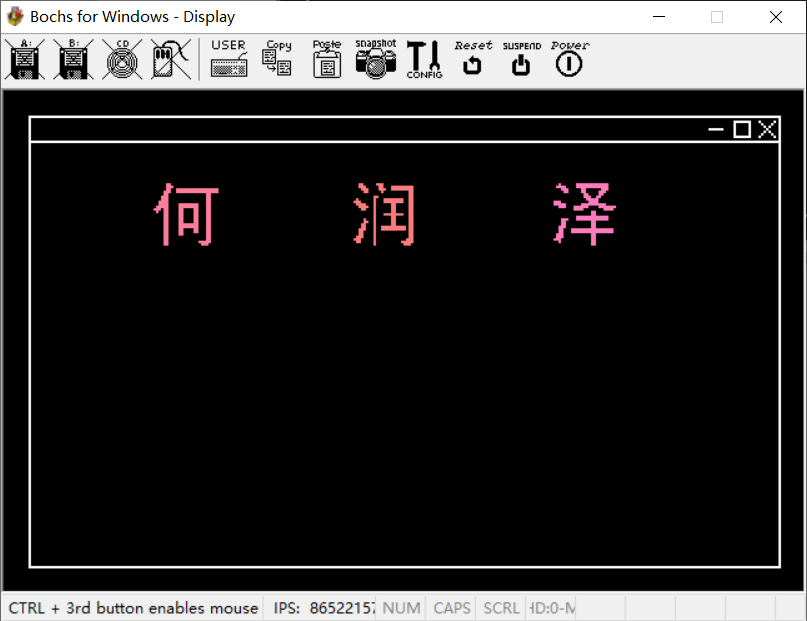
jmp play

key0: mov bx,word[ds:frequency+18]

jmp play

**接下来是多线程的动态名字显示，每个字颜色不同且动态变化**





**其中文字的显示通过封装好的点阵图函数来绘制，多线程的动态显示通过定时器中断实现**

;多线程姓名，通过flag1控制进入的线程，通过flag2控制颜色

int\_time0:

push si

push bx

push cx

cmp byte[ds:flag1],2 ;根据flag1判断此次中断需要打印的字

je i1

cmp byte[ds:flag1],3

je i2

mov bx,11900 ;文字打印位置

mov cx,0

mov si,0

mov bp,he

call zone

inc byte[ds:flag1]

jmp exit3

i1: mov bx,11980 ;文字打印位置

mov cx,0

mov si,0

mov bp,run

call zone

inc byte[ds:flag1]

jmp exit3

i2: mov bx,12060 ;文字打印位置

mov cx,0

mov si,0

mov bp,ze

call zone

mov byte[ds:flag1],1 ;打印完“泽”字后重置flag1

exit3:

inc byte[ds:flag2]

pop cx

pop bx

pop si

mov al,0x20

out 0x20,al

Iret

;点阵转文字处理

zone:

z0:

cmp byte[bp+si],0

je z1

mov al,byte[ds:flag2]

mov byte[es:bx],al

z1:

inc si

inc cx

cmp cx,25 ;换行判断

ja z2

inc bx

jmp z0

z2:

cmp si,625 ;25x25点阵遍历完成判断

ja exit

mov cx,0

add bx,295 ;换行

jmp z0

exit: ret

**最后是结合键盘中断和8253实现电子琴，附上测试曲目两只老虎（ 1231 1231 345 345 565431 565431 201 201）**

;电子琴启动

play:

call sound ;发声

call delay ;延迟

jmp next ;返回键盘中断

;发声函数

sound:

mov al,10110110b ;初始化8253

out 43h,al

mov dx,0012h

mov ax,348ch

mov di,bx

div di

out 42h,al

mov al,ah

out 42h,al

in al,61h

or al,03h

out 61h,al

ret

;延时函数

delay:

mov ax,0fffh

d1: mov cx,100

d2: loop d2

dec ax

jnz d1

ret

#### 心得体会

这次课设的实现，结合了这个学期老师所讲述的所有内容以及自己对微机接口的理解。因为这是第一次做相对来说比较大的一个微机项目，经验还是比较不足，一开始实现时还是按着之前做单个模块时的那种理解去做这个课设，就出现了不同的模块间不兼容的情况，此时每多加一个功能就要打上一大堆繁琐的代码，这样一来在调试时很难找出错误，二来程序逻辑不清结构比较紊乱，所以一开始写出来的代码整整有一千两百多行。于是我尝试着用做java课程设计的方法来操作，第一步先打出整个程序的框架，找出每个模块的功能，然后再找每个模块之间的联系，把有类似作用的模块进行泛化并整理，最后再封装。按照这样的思路我开始对我的代码进行大刀阔斧的修改，最后把一千行的代码精简到了五百行左右，而且此时的代码逻辑性强，一眼就能看明白我的设计思路，而不是图形画到哪儿就写到哪儿。在精简完成之后，一旦出现bug我能很快锁定bug出现的位置，这就是封装所带来的好处。总之，通过这次课程设计我学到了非常多，而且在答辩中老师提供的思路也让我对程序的设计思路有所扩展，我相信未来我可以做的更好。

#### 6.完整实验代码

org 0x8400

jmp go

go:

sti

mov ax,0xb800

mov es,ax

mov ax,0

mov ds,ax

call frame ;绘制窗口

call print ;在图形模式下打印字符串

cli

mov word[ds:0x24],int\_key ;键盘中断

mov word[ds:0x26],0

sti

jmp $

;键盘驱动

int\_key:

mov dx,0x60

in al,dx

cmp al,0x9c

je enter

cmp al,0x82

je key1

cmp al,0x83

je key2

cmp al,0x84

je key3

cmp al,0x85

je key4

cmp al,0x86

je key5

cmp al,0x87

je key6

cmp al,0x88

je key7

cmp al,0x89

je key8

cmp al,0x8a

je key9

cmp al,0x8b

je key0

next: mov dx,0x20

mov al,0x61

out dx,al

call delay

in al,61h

and al,0fch

out 61h,al

iret

key1: mov bx,word[ds:frequency]

jmp play

key2: mov bx,word[ds:frequency+2]

jmp play

key3: mov bx,word[ds:frequency+4]

jmp play

key4: mov bx,word[ds:frequency+6]

jmp play

key5: mov bx,word[ds:frequency+8]

jmp play

key6: mov bx,word[ds:frequency+10]

jmp play

key7: mov bx,word[ds:frequency+12]

jmp play

key8: mov bx,word[ds:frequency+14]

jmp play

key9: mov bx,word[ds:frequency+16]

jmp play

key0: mov bx,word[ds:frequency+18]

jmp play

;电子琴

play:

call sound

call delay

jmp next

;第二模块

enter:

mov word[ds:0x20],int\_time0

mov word[ds:0x22],0

mov ah,00h

mov al,13h

int 10h

mov ax,0xb800

mov es,ax

mov ax,0

mov ds,ax

call frame

;多线程姓名，通过flag1控制进入的线程，通过flag2控制颜色

int\_time0:

push si

push bx

push cx

cmp byte[ds:flag1],2

je i1

cmp byte[ds:flag1],3

je i2

mov bx,11900

mov cx,0

mov si,0

mov bp,he

call zone

inc byte[ds:flag1]

jmp exit3

i1: mov bx,11980

mov cx,0

mov si,0

mov bp,run

call zone

inc byte[ds:flag1]

jmp exit3

i2: mov bx,12060

mov cx,0

mov si,0

mov bp,ze

call zone

mov byte[ds:flag1],1

exit3:

inc byte[ds:flag2]

pop cx

pop bx

pop si

mov al,0x20

out 0x20,al

iret

;发声

sound:

mov al,10110110b

out 43h,al

mov dx,0012h

mov ax,348ch

mov di,bx

div di

out 42h,al

mov al,ah

out 42h,al

in al,61h

or al,03h

out 61h,al

ret

;延时

delay:

mov ax,0fffh

d1: mov cx,100

d2: loop d2

dec ax

jnz d1

ret

;名字处理

zone:

z0:

cmp byte[bp+si],0

je z1

mov al,byte[ds:flag2]

mov byte[es:bx],al

z1:

inc si

inc cx

cmp cx,25

ja z2

inc bx

jmp z0

z2:

cmp si,625

ja exit

mov cx,0

add bx,295

jmp z0

exit: ret

;打印字符串

print:

mov dh,5

mov dl,5

call setcursor

mov si,0

p1: cmp si,8

je s1

mov al,byte[ds:text1+si]

call write

inc si

jmp p1

s1: mov dh,9

mov dl,5

call setcursor

mov si,0

p2: cmp si,31

je s2

mov al,byte[ds:text2+si]

call write

inc si

jmp p2

s2: mov dh,11

mov dl,5

call setcursor

mov si,0

p3: cmp si,30

je exit1

mov al,byte[ds:text3+si]

call write

inc si

jmp p3

exit1: ret

;设置光标位置

setcursor:

mov ah,02h

mov bh,0

int 10h

ret

;打印单个字符

write:

mov ah,0eh

mov bl,1

int 10h

ret

;画出窗口

frame: mov ah,00h

mov al,13h

int 10h

mov ax,0x0a000

mov es,ax

mov si,color

call dis

call box

call guide

ret

;导航栏

guide:

mov bx,10

mov ax,20

mov cx,300

call lineh

mov bx,282

mov ax,15

mov cx,6

call lineh

mov bx,292

mov ax,12

mov cx,6

call lineh

mov bx,292

mov ax,18

mov cx,6

call lineh

mov bx,292

mov ax,12

mov cx,6

call linev

mov bx,298

mov ax,12

mov cx,7

call linev

mov bx,302

mov ax,12

mov cx,7

call slashl

mov bx,308

mov ax,12

mov cx,7

call slashr

ret

;窗口框

box:

mov bx,10

mov ax,10

mov cx,300

call lineh

mov bx,10

mov ax,10

mov cx,180

call linev

mov bx,10

mov ax,190

mov cx,300

call lineh

mov bx,310

mov ax,10

mov cx,181

call linev

ret

;竖线

linev:

mov dx,320d

mul dx

add bx,ax

x1: mov byte[es:bx],1

add bx,320d

dec cx

cmp cx,0

jne x1

ret

;横线

lineh:

mov dx,320d

mul dx

add bx,ax

x2: mov byte[es:bx],1

inc bx

dec cx

cmp cx,0

jne x2

ret

;左上至右下斜线

slashl:

mov dx,320d

mul dx

add bx,ax

x3: mov byte[es:bx],1

add bx,321

dec cx

cmp cx,0

jne x3

ret

;右上至左下斜线

slashr:

mov dx,320d

mul dx

add bx,ax

x4: mov byte[es:bx],1

add bx,319

dec cx

cmp cx,0

jne x4

ret

;设置颜色

dis: mov dx,0x3c8

mov al,[si]

out dx,al

mov dx,0x3c9

mov al,[si+1]

out dx,al

mov al,[si+2]

out dx,al

mov al,[si+3]

out dx,al

add si,4

ret

flag1 db 1

flag2 db 1

color db 1,255,255,255 ;窗口颜色

text1 db "WELCOME!"

text2 db "Press ENTER key to show my name"

text3 db "Press key 0 to 9 to play piano"

frequency dw 261,293,329,349,392,440,493,542,585,200

he db 0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

db 0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

db 0,0,0,0,0,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

db 0,0,0,0,1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1

db 0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0

db 0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0

db 0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0

db 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0

db 0,0,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,0

db 0,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,0

db 1,1,1,0,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,1,0,0,0

db 0,1,0,0,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0

db 0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0

run db 0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0

db 0,0,1,1,1,0,0,0,0,0,1,1,0,0,0,1,1,1,1,1,1,1,1,1,0,0

db 0,0,0,1,1,1,0,0,0,0,1,1,1,0,0,1,1,1,1,1,1,1,1,1,0,0

db 0,0,0,0,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0

db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0

db 0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0

db 0,1,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,1,1,1,0,0,1,1,0,0

db 1,1,1,0,0,0,0,0,1,0,0,0,1,1,1,1,1,1,1,1,0,0,1,1,0,0

db 0,1,1,1,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0

db 0,0,1,1,1,1,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0

db 0,0,0,1,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0

db 0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0

db 0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1,1,1,1,1,0,0,1,1,0,0

db 0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0

db 0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0

db 0,0,0,0,1,1,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0

db 0,0,0,1,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,1,1,0,0

db 0,0,0,1,1,0,0,0,1,0,0,1,1,1,1,1,1,1,1,1,1,0,1,1,0,0

db 0,0,1,1,1,0,0,0,1,0,0,1,1,1,1,1,1,1,1,1,1,0,1,1,0,0

db 0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0

db 0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0

db 0,1,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0

db 1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0

db 0,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0

db 0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0

ze db 0,0,1,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0

db 0,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0

db 0,0,0,1,1,1,0,0,0,0,0,1,1,0,0,0,0,0,0,1,1,1,0,0,0,0

db 0,0,0,0,1,1,0,0,0,0,0,1,1,0,0,0,0,0,1,1,1,0,0,0,0,0

db 0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,0,0

db 0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0

db 0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0

db 1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0

db 1,1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0

db 0,0,1,1,1,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,0

db 0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,0,0

db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0

db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0

db 0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0

db 0,0,0,1,1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0

db 0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0

db 0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0

db 0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0

db 0,0,1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0

db 0,0,1,1,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0

db 0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0

db 0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0

db 1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0

db 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0

db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0