课程设计2报告
设计思路
前置准备
实验流程
设计代码

课程设计2报告

设计思路

1. int 19 将 0 道 0 面 1 扇区（512） 的内容复制道 **0:7c00h** 处， CS:IP 改为 **0:7c00h**
    1. 所以 0 道 0 面 1 扇区功能为 复制 0 道 0 面 2 扇区 开始的3个扇区 中的内容(**Boot_end - Boot**)道 **0:7E00h** 处,CS:IP 改为 **0:7E00h**
2. 选功能的话即通过键盘中断来执行
    3. 一号功能 把 CS:IP 改为 **FFFF:0**
    4. 二号功能 把 c 盘 0 道 0 面 1 扇区（512） 的内容读到 **0:7c00h** 处
    5. 三号功能 从 CMOS 读出时间
    6. 四号功能 调用键盘中断读取键盘输入，并修改 CMOS 对应位置
        1. 新增退出功能
        2. 新增日期格式判断
            1. 没有对每个月是31,30还是28,29天判断，全部统一判断31天
            2. 年份没有判断
            3. 0<月<=12,0<小时<=24,0<=秒（分）<=60

前置准备

1. 安装virtual Box && win xp
2. 为 win xp 添加 软盘 && 共享空间

实验流程

1. 将编译的 asm 文件的可执行程序 放入 共享空间 并在 win xp 中的CMD终端中运行
2. 重启计算机，测试相应功能

设计代码

```
assume cs:code,ds:data,ss:stack

stack segment
 db 128 dup (0)
stack ends

data segment
```

```
; begin        db 512 dup (0) ;一个扇区
; begin_boot   db 512 dup (0)
;              db 512 dup (0)
;              db 512 dup (0)
data ends

code segment
start:
       mov ax,stack
       mov ss,ax
       mov sp,128

       call copy_introduce
       call copy_boot_disk


       mov ax,4c00h
       int 21h
;--------------------------------
introduce:                      ;引导程序，将程序复制到0:7c00处，
        mov bx,0
    mov ss,bx
    mov sp,7c00h

    call save_old_int9
    call copy_Boot_from_disk

    mov bx,0
    push bx
    mov bx,7e00h           ;设置cs：ip为0:7e00h执行Boot程序
    push bx
    retf
        ;-----------------------------
        copy_Boot_from_disk:
                mov bx,0
                mov es,bx
                mov bx,7e00h

                mov al,2
                mov ch,0
                mov cl,2
                mov dl,0
                mov dh,0
                mov ah,2
                int 13h

                ret
        ;---------------------------------
        save_old_int9:
                mov bx,0
                mov es,bx

                push es:[9*4]
                pop es:[200h]
                push es:[9*4+2]
                pop es:[202h]
                ret
        ;----------------------
```

```
                db 512 dup (0)
        introduce_end:nop
 ;================================
copy_introduce:
       mov bx,cs
       mov es,bx
       mov bx,offset introduce

       mov al,1
       mov ch,0
       mov cl,1
       mov dl,0
       mov dh,0
       mov ah,3
       int 13h
       ret
;----------------------
copy_boot_disk:
       mov bx,cs
       mov es,bx
       mov bx,offset Boot

       mov al,2
       mov ch,0
       mov cl,2
       mov dl,0
       mov dh,0
       mov ah,3
       int 13h
       ret
;-------------------------------
Boot:
        jmp Boot_start
        ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
        option1   db '(1) reset pc',0
        option2   db '(2) start system',0
        option3   db '(3) clock',0
        option4   db '(4) set clock',0

        address_option  dw offset option1 - offset Boot + 7e00h
                        dw offset option2 - offset Boot + 7e00h
                        dw offset option3 - offset Boot + 7e00h
                        dw offset option4 - offset Boot + 7e00h
        timestyle   db '00/00/00 00:00:00',0
        timeadress  db 9,8,7,4,2,0
        string_stack   db 12 dup ('0'),0
        error_string db 'time format error!!!!',0
        ;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

        Boot_start:
                call init_reg
                call clear_screen
                call show_option

                jmp short choose_option

                mov ax,4c00h
                int 21h
```

```
;----------------------
        choose_option:
                call clear_buff

                mov ah,0
                int 16h

                cmp al,'1'
                je choose1
                cmp al,'2'
                je choose2
                cmp al,'3'
                je choose3
                cmp al,'4'
                je choose4

                jmp choose_option

        choose1:mov di,160*3
                mov byte ptr es:[di],'1'
                mov bx,0ffffh
                push bx
                mov bx,0
                push bx
                retf
                jmp choose_option

        choose2:mov di,160*3
                mov byte ptr es:[di],'2'
                call start_old_system
                jmp choose_option

        choose3:mov di,160*3
                mov byte ptr es:[di],'3'
                call show_clock
                jmp Boot_start

        choose4:mov di,160*3
                mov byte ptr es:[di],'4'
                call set_clock
                jmp Boot_start
;----------------------
start_old_system:
        mov bx,0
        mov es,bx
        mov bx,7c00h

        mov al,1
        mov ch,0
        mov cl,1
        mov dl,80h      ;80h代表C盘
        mov dh,0
        mov ah,2
        int 13h

        mov bx,0
        push bx
        mov bx,7c00h
```

```
        push bx
        retf
;------------------------
set_clock:
        ; call clear_screen
        call clear_string_stack
        call show_string_stack
        call get_string

        cmp ah,01h
        je set_clock_ret

        call check_time_fromat
        call set_time

        set_clock_ret:
        ret
        ;--------------------
        check_time_fromat:
                mov bx,offset timeadress - offset Boot + 7e00h
                mov si,offset string_stack - offset Boot +7e00h
                mov cx,6
                ctf_lp1:
                        mov dx,ds:[si]
                        sub dh,30h
                        sub dl,30h
                        shl dl,1
                        shl dl,1
                        shl dl,1
                        shl dl,1
                        and dh,00001111b
                        or dl,dh

                        ;6Y 5M 4D 3H 2m 1S
                        cmp cx,6
                        je check_year
                        cmp cx,5
                        je check_month
                        cmp cx,4
                        je check_day
                        cmp cx,3
                        je check_hour
                        cmp cx,2
                        je check_min
                        cmp cx,1
                        je check_sec
                        jmp continue_check

                                ;-------------------------------
                                check_year:
                                        jmp continue_check
                                check_month:
                                        cmp dl,12h
                                        ja print_error
                                        cmp dl,0h
                                        je print_error
                                        jmp continue_check
                                check_day:
```

```
                                        cmp dl,31h
                                        ja print_error
                                        cmp dl,0h
                                        je print_error
                                        jmp continue_check
                                check_hour:
                                        cmp dl,24h
                                        ja print_error
                                        jmp continue_check
                                check_min:
                                        cmp dl,60h
                                        ja print_error
                                        jmp continue_check
                                check_sec:
                                        cmp dl,60h
                                        ja print_error
                                        jmp continue_check

                        ;------------------------------
                        continue_check:
                                add si,2
                                inc bx
                        loop ctf_lp1
                        jmp check_time_fromat_ret

                                ;------------------------------
                                print_error:
                                        push si
                                        push di
                                        mov si,offset error_string -
offset Boot + 7e00h
                                        mov di,160*20
                                        call showstr
                                        pop si
                                        pop di
                                        call delay

                                        mov cx,7

                                        jmp check_time_fromat_ret
            check_time_fromat_ret:
                                ret
                        ;------------------------------
                        delay:
                                push ax
                                push dx

                                mov dx,10000h
                                mov ax,0

                                s1: sub ax,1
                                        sbb dx,0
                                        cmp ax,0
                                        jne s1
                                        cmp dx,0
                                        jne s1

                                        pop dx
```

```
                        pop ax

                        ret

;--------------------
set_time:

        cmp cx,7
        je set_time_ret

        mov bx,offset timeadress - offset Boot + 7e00h
        mov si,offset string_stack - offset Boot +7e00h
        mov cx,6
        settime:

                mov dx,ds:[si]
                sub dh,30h
                sub dl,30h
                shl dl,1
                shl dl,1
                shl dl,1
                shl dl,1
                and dh,00001111b
                or dl,dh
                mov al,ds:[bx]
                out 70h,al
                mov al,dl
                out 71h,al

                add si,2
                inc bx


                loop settime
        set_time_ret:
                ret
;-------------------------
get_string:
        mov si,offset string_stack - offset Boot + 7e00h
        mov bx,0
        getstring:
                call clear_buff
                mov ah,0
                int 16h
                cmp al,'0'
                jb notnumber
                cmp al,'9'
                ja notnumber
                call char_push
                call show_string_stack

                jmp getstring
        getstringret:
                ret
        notnumber:
                cmp ah,0eh      ;backspace
                je isbackspace
                cmp ah,01h      ;ese
```

```
                        je getstringret
                        cmp ah,1ch
                        je getstringret ;enter
                        jmp getstring
                isbackspace:
                        call char_pop
                        call show_string_stack
                        jmp getstring
                ;-------------------------
                char_pop:
                        cmp bx,0
                        je charpopret
                        dec bx
                        mov byte ptr ds:[si+bx],'0'
                        charpopret:
                                ret
                ;-------------------------
                char_push:
                        cmp bx,11
                        ja charpushret
                        mov ds:[si+bx],al
                        inc bx
                        charpushret:
                                ret
                ;-------------------------

        ;-------------------
        show_string_stack:
                push si
                push di
                mov si,offset string_stack - offset Boot + 7e00h
                mov di,160*4
                call showstr
                pop di
                pop si
                ret
        ;--------------------
        clear_string_stack:
                push bx
                push cx
                push es
                push si
                push di

                mov si,offset string_stack - offset Boot + 7e00h
                mov dx,3030h

                mov cx,6
                clearstringstack:
                        mov ds:[si],dx
                        add si,2
                        loop clearstringstack

                        pop di
                        pop si
                        pop es
                        pop cx
                        pop bx
```

```
                        ret
        ;---------------------
        show_clock:
                call show_style
                call set_new_int9

                mov bx,offset timeadress - offset Boot + 7e00h
                showtime:
                        mov si,bx
                        mov di,160*20
                        mov cx,6
                        showdate:
                                mov al,ds:[si]
                                out 70h,al
                                in al,71h

                                mov ah,al
                                shr ah,1
                                shr ah,1
                                shr ah,1
                                shr ah,1
                                and al,00001111b
                                add ah,30h
                                add al,30h
                                mov es:[di],ah
                                mov es:[di+2],al
                                add di,6
                                inc si
                                loop showdate

                                jmp showtime
                        show_clockret:
                                call set_old_int9
                                ret
                ;----------------------
                show_style:
                        mov si,offset timestyle - offset Boot + 7e00h
                        ;mov si,offset error_string - offset Boot + 7e00h
                        mov di,160*20
                        call showstr
                        ret
                ;--------------------
                set_old_int9:
                                push bx
                                push es

                                mov bx,0
                                mov es,bx
                                cli
                                push es:[200h]
                                pop es:[9*4]
                                push es:[202h]
                                pop es:[9*4+2]
                                sti

                                pop es
                                pop bx
                                ret
```

```
                        ;---------------------
                        set_new_int9:
                                push bx
                                push es

                                mov bx,0
                                mov es,bx

                                cli
                                mov word ptr es:[9*4],offset newint9 - offset
Boot + 7e00h
                                mov word ptr es:[9*4+2],0
                                sti

                                pop es
                                pop bx
                                ret
                        ;------------------------
                        newint9:
                                push ax
                                call clear_buff

                                in al,60h
                                pushf
                                call dword ptr cs:[200h]

                                cmp al,01h
                                je inesc
                                cmp al,3bh
                                jne int9ret
                                call change_time_color

                                int9ret:
                                pop ax
                                iret
                                inesc:
                                pop ax
                                add sp,4
                                popf
                                jmp show_clockret
                        ;----------------------
                        change_time_color:
                                push bx
                                push cx
                                push es

                                mov bx,0b800h
                                mov es,bx
                                mov cx,17
                                mov bx,160*20+1
                                change_time_colors:
                                inc byte ptr es:[bx]
                                add bx,2
                                loop change_time_colors

                                pop es
                                pop cx
                                pop bx
```

```
                ;-------------------------
                clear_buff:
                        mov ah,1
                        int 16h
                        jz clearbuffret
                        mov ah,0
                        int 16h
                        jmp clear_buff
                        clearbuffret:
                                ret
        ;---------------------
        show_option:
                mov bx,offset address_option - offset Boot + 7e00h
                mov cx,4
                mov di,160*10 + 30*2
                show_options:
                        mov si,ds:[bx]
                        call showstr
                        add di,160
                        add bx,2
                        loop show_options
                        ret
        ;-----------------------------
        showstr:
                push cx
                push di
                showstrs:
                        mov cl,ds:[si]
                        cmp cl,0
                        je showstrret
                        mov es:[di],cl
                        add di,2
                        inc si
                        jmp short showstrs
                showstrret:
                        pop di
                        pop cx
                        ret

        ;----------------
        init_reg:
                mov bx,0b800h
                mov es,bx

                mov bx,0
                mov ds,bx
                ret
        ;------------------
        clear_screen:
                mov bx,0
                mov dx,0700h   ;清屏中对字符属性设置应该为07h，而不是0
                mov cx,2000
                clearscreen:
                        mov es:[bx],dx
                        add bx,2
                        loop clearscreen
                        ret
                ;----------------------
```

```
                        db 512 dup (0)
                Boot_end:
                nop

code ends
end start
```