第九章　BIOS和DOS中断

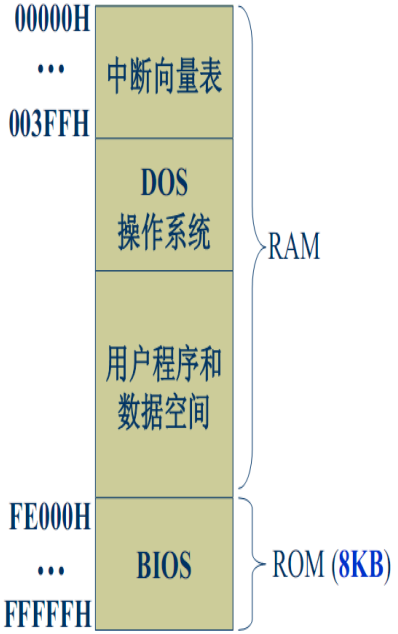
- DOS 和 BIOS 功能调用
- 键盘I/O
- 显示器I/O
- 打印机I/O

本章目标

- 理解DOS/BIOS功能调用概念
- 掌握DOS/BIOS功能调用程序设计方法

BIOS简介

- 固化在ROM中的基本输入输出系统BIOS (Basic Input / Output system)
  - 地址空间：FE000H-FFFFFH，8KB
  - 包含
    - 主要的I/O设备处理和接口控制程序
      - I/O设备硬件中断处理程序
      - I/O设备软件中断调用处理程序
    - 许多常用的系统例行程序
      - 系统加电自检、引导装入等功能模块
  - 一般以中断处理程序的形式存在、被调用
- 例如：软件中断调用
  - 显示输出：10H号中断处理程序
  - 打印输出：17H号中断处理程序
  - 键盘输入：16H号中断处理程序



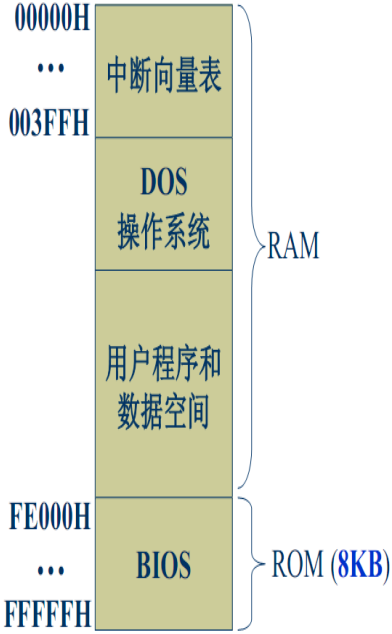
BIOS功能调用

附录5 P611：BIOS软件中断功能调用

- 显示器：INT 10H
- 磁　盘：INT 13H
- 串行口：INT 14H
- 键　盘：INT 16H
- 打印机：INT 17H
- 引导装入：INT 19H
- 鼠　标：INT 33H

DOS简介

- 磁盘操作系统DOS (Disk Operating System)
  建立在BIOS基础上的PC机操作系统
- 组成：
  - IBMBIO.COM：I/O设备处理程序，完成设备到内存或内存到外设数据传送
    - 例如：DOS调用BIOS显示输出程序完成显示输出，调用BIOS打印输出程序完成打印输出，调用BIOS键盘输入程序完成键盘输入等
  - IBMDOS.COM：作业管理与监控、文件管理程序、设备处理程序
    - 设备管理与监控：通过IBMBIO.COM 形成一个或多个BIOS调用



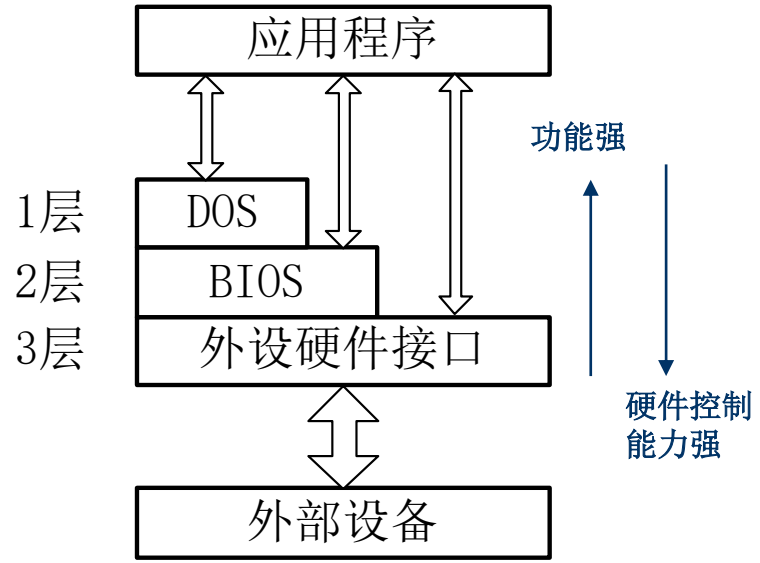
DOS系统功能调用

附录4 P595：DOS系统功能调用 INT 21H

- AH=1 键盘输入并回显
- AH=2 显示字符输出
- AH=9 显示字符串输出
- AH=0AH 键盘输入到缓冲区
- AH=4CH 带返回码终止

应用程序、DOS、BIOS和外设接口之间的关系

- DOS和BIOS都提供某些相同的功能，但它们之间的层次关系是不同的
  - BIOS直接建立在硬件的基础上
  - DOS建立在BIOS的基础上，通过BIOS控制硬件



应用程序、DOS、BIOS和硬件接口间的关系

应用程序、DOS、BIOS和外设接口之间的关系

- DOS、BIOS和硬件接口都为应用程序提供完成输入输出的功能，而且随着层次的加深访问外部设备的能力越强
- 从应用程序的编写角度出发，随着层次的加深，应用程序的编写难度和复杂度大大增加

应用程序、DOS、BIOS和外设接口之间的关系

- 编写应用程序推荐调用I/O的顺序如下：

  DOS--> BIOS--> 直接外设接口访问

  - 通常I/O操作应该首先选择调用DOS提供的系统功能，完成输入输出，这样实现容易，而且对硬件的依赖性最少，程序可移植性好
    - 通过BIOS,DOS屏蔽了底层硬件的差异
  - 如果DOS不提供某种服务或者不能使用DOS的场合可考虑BIOS调用
  - 应用程序也可以直接操纵外设接口来控制外设，从而获得最高效率，但编程复杂性最高

DOS和BIOS功能调用方法

- 采用软件中断的方式实现功能调用
- 应用程序在进行中断调用时应明确
  1. 中断类型号
  2. 功能号：功能号→AH；子功能号→AL
  3. 入口参数：通过寄存器提供专门的调用参数
- 应用程序调用时步骤如下：

  1、将调用所需的入口参数装入指定的寄存器

  2、如需功能号：AH←功能号

  3、如需子功能号：AL←子功能号

  4、按中断号调用DOS或BIOS中断：INT n

  5、检查返回参数是否正确等

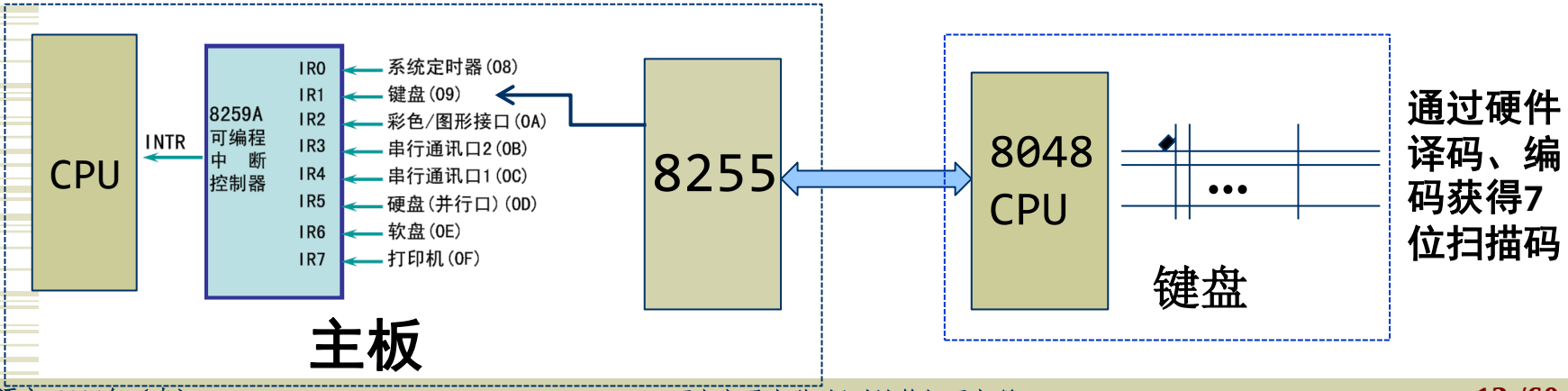
- 为什么DOS和BIOS功能调用使用中断方式实现调用？（站在系统设计员的角度）
  - 系统设计简单、高效： BIOS的功能调用也是调用BIOS中的基本硬件和异常等中断处理程序。中断调用方式与硬件中断处理程序转入方式统一，系统设计方便
    - 不必设置中断入口和子程序调用两种程序等
  - 用户编程方便、可移植性好： DOS的功能调用实质上是调用DOS中的扩展硬件等中断处理程序，中断调用方式与中断处理程序转入方式统一，使用透明，应用程序编写独立、方便
    - 应用程序不必关心低层功能处理程序入口，现场保存情况等
    - 程序编写独立，可移植性好

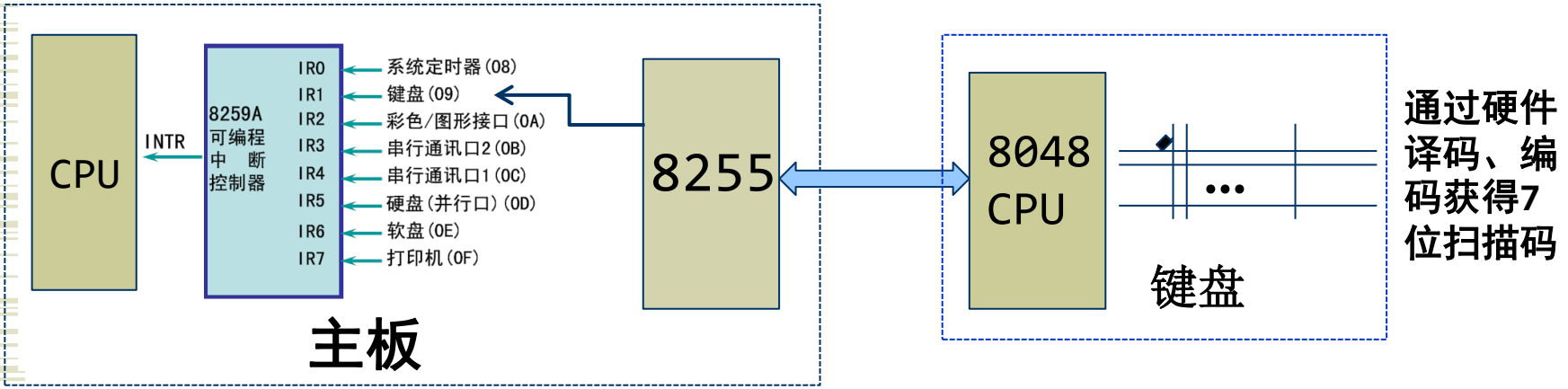
    BIOS：机器不同，硬件配置不同，BIOS具体程序实现等不同
    DOS：功能子程序每次装入位置不同；类型/版本不同实现不同
  - 最主要的原因是低层BIOS和DOS的修改（处理程序的入口地址改变）对上层应用透明，应用程序不变，可移植性好

9.1 键盘I/O

- **主板上：**
  - **键盘接口芯片：** 8255可编程序外围接口芯片
  - **键盘中断允许：** 8259中屏蔽寄存器（21H端口）第1位=0
- **键盘上：**
  - **按 键**16*8矩阵排列
  - **由**Intel 8048**单片机控制对键盘扫描**

7	6	5	4	3	2	1	0
打印机	软盘	硬盘	COM1	COM2	彩显	键盘	定时器





- 键盘**硬件中断(09H)**：如果有键按下，且中断也允许
  - 向CPU发中断，由BIOS中的键盘外设中断程序处理，转换成字符码，并将字符码和扫描码存储在**内存缓冲区中**
- 键盘**软件中断**：DOS(21H)和BIOS(16H)软件中断调用
  - INT 21H, INT 16H
  - 应用程序可以使用DOS和BIOS软件中断调用获得存储在**内存缓冲区中**的字符码和扫描码
- 键盘**软件中断**和键盘**硬件中断**在BIOS中由不同的中断处理程序完成，功能不同

9.1.1 字符码与扫描码

- 从键盘输入端口60H读取一个字节，其低7位是扫描码（01H-53H，见表9.3）
- 每个键的扫描码有两个：

  通码（键按下，0）、断码（键放开，1）
- BIOS键盘中断处理程序将扫描码转换为字符码
  - ASCII码，或0（非ASCII码键，如功能键）
- 转换成的字符码和扫描码存储在内存的键盘缓冲区KB_BUFFER中

```
0040:0001A   BUFF_HEAD DW ?
0040:0001C   BUFF_TAIL DW ?
0040:0001E   KEY_BUFFER DW 16 DUP(?)
0040:0003E   KEY_BUFFER_END LABEL WORD
```

缓冲区是先进先出的循环队列

9.1.2 BIOS键盘中断

表9.4 BIOS键盘中断（INT 16H）

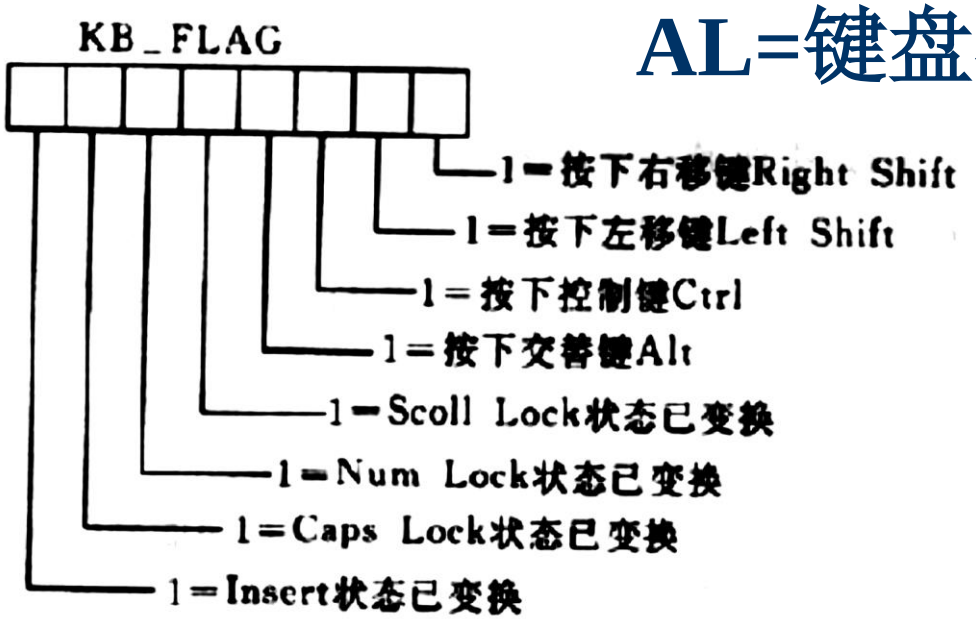
AH	功能	返回参数
0	从键盘读一字符	AL=字符码 AH=扫描码
1	检测键盘缓冲区	如ZF=0 AL=字符码 AH=扫描码 如ZF=1，缓冲区空
2	取键盘状态字	AL=键盘状态字

这里读键盘字符是从内存的键盘缓冲区中读取

如何判断不具有ASCII码的功能键动与否？

- 用INT 16H, AH=2 读键盘状态字

AL=键盘状态字



这里读键盘字符是从内存的键盘缓冲区中读取

9.1.3 DOS键盘功能调用

功能更强大
使用更方便

使用功能1时，如果按下Ctrl_C或Ctrl_Break，DOS在返回前直接结束程序

AH	功 能	调用参数	返回参数
1	从键盘输入一个字符并回显在屏幕上		AL=字符
6	读键盘字符	DL=0FFH	若有字符可取， AL=字符 ZF=0 若无字符可取， AL=0Z F=1
7	从键盘输入一个字符，不回显		AL=字
8	从键盘输入一个字符，不回显，检测Ctrl_Break		AL=字符
A	输入字符到缓冲区		
B	读键盘状态		
C	清除键盘缓冲区，并调用一种键盘功能	DS：DX=缓冲区首址 AL=键盘功能号（1，6，7，8或A）	AL=0FFH有键入 AL=00无键入

- **注意：** 调用DOS或BIOS从内存的键盘缓冲区中读取键盘按键字符后，相应的键盘字符就会从内存的键盘缓冲区中清除掉。

例1 从键盘读一个字符

```
; BIOS调用完成从键盘读一个字符
    MOV   AH, 0
    INT   16H          ; 字符码 →AL, 扫描码 →AH
```

表9.4 BIOS键盘中断（INT 16H）

AH	功能	返回参数
0	从键盘读一字符	AL=字符码 AH=扫描码

```
; DOS调用完成从键盘读一个字符
    MOV   AH, 7
    INT   21H          ; 字符码 →AL
```

表 9.5 DOS 键盘操作（INT 21H）

AH	功能	调用参数	返回参数
7	从键盘输入一个字符，不回显		AL=字符码

例2．先清除键盘缓冲区，然后再从键盘读一个字符

表 9.4　BIOS 键盘中断（INT 16H）

AH	功能	返回参数
0	从键盘读一字符	AL=字符码 AH=扫描码
1	读键盘缓冲区的字符	如ZF=0 AL=字符码 AH=扫描码 如 ZF=1，缓冲区空
2	取键盘状态字	AL=键盘状态字

```
；BIOS调用
        ┇
REPT：  MOV    AH，1
        INT    16H          ；判缓冲区空？
        JZ     SKIP         ；为空，转移
        MOV    AH，0
        INT    16H          ；从键盘缓冲区取一个字符码
        JMP    REPT         ；继续判断
SKIP：  MOV    AH，0        ；等待键盘输入新的字符码
        INT    16H
```

；DOS调用

```
        ┆
REPT:
        MOV     AH，0BH         ；参看P595
        INT     21H             ；判有无输入？
        CMP     AL，0
        JZ      SKIP            ；为空，转移
        MOV     AH，7
        INT     21H             ；从键盘缓冲区取走一个字符
        JMP     REPT            ；继续判断
SKIP:
        MOV     AH，7           ；等待键盘输入有效字符
        INT     21H
        ┆
```

或者如下调用：

```
        MOV     AL，7
        MOV     AH，0CH
        INT     21H
```

表 9.5　DOS 键盘操作（INT 21H）

AH	功能	调用参数	返回参数
1	从键盘输入一个字符，并回显		AL=字符码
6	读键盘字符	DL=0FFH	若有字符可取： AL=字符码 ZF=0 若有字符可取： AL=0 ZF=1
7	从键盘输入一个字符，不回显		AL=字符码
8	从键盘输入一个字符，不回显，检测 Ctrl_Break		AL=字符码
A	输入字符到缓冲区	DS:DX=缓冲区首址	参见图 9.3
B	读键盘状态		AL=0FFH 有键入 AL=00 无键入
C	清除键盘缓冲区，并调用一种键盘功能	AL=键盘功能号（1，6，7，8，A）	

例3. 显示按键，在检测到所按下的CTRL键后结束运行

- 调用16H号中断的2号功能
  - 取得键盘状态字节
  - 判断是否按下了CTRL键
    - 若按下CTRL键则结束运行；
    - 否则显示按键
- 汇编语言源程序如下：

```
sseg    segment stack           ; 建立堆栈段
        dw      200 dup(?)
tos     label   word
sseg    ends
ctrl=00000100b          ; 定义变换键ctrl判断字
cseg    segment                 ; 建立代码段
        assume cs:cseg, ss:sseg
begin   proc    far
        mov     sp, offset  tos  ; 初始化堆栈指针
        mov     ax, sseg
        mov     ss, ax
        push    ds   ; 压入返回DOS地址
        mov     ax, 0
        push    ax
start:  mov     ah, 2           ; 取 KB_FLAG
        int     16h
        test    al, ctrl        ; 判断是ctrl键吗?
        jnz     finish          ; 是ctrl键，结束运行
         mov    ah, 1
        int     16h
        jz      start           ; 缓冲区空，无键可读，转移到start
        mov     dl, al          ; 显示
        mov     ah, 2
        int     21h
       mov      ah,0
        int     16h
        jmp     start           ; 继续下一轮
finish: ret
begin   endp
cseg    ends
        end     begin
```

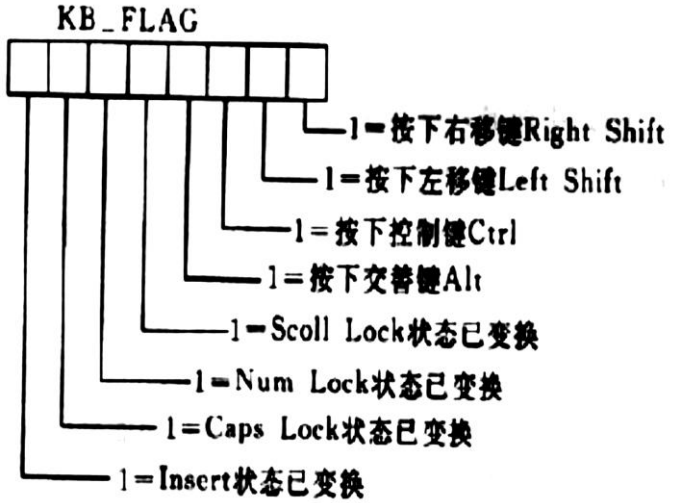


图 9.2 键盘状态字节

表 9.4 BIOS 键盘中断（INT 16H）

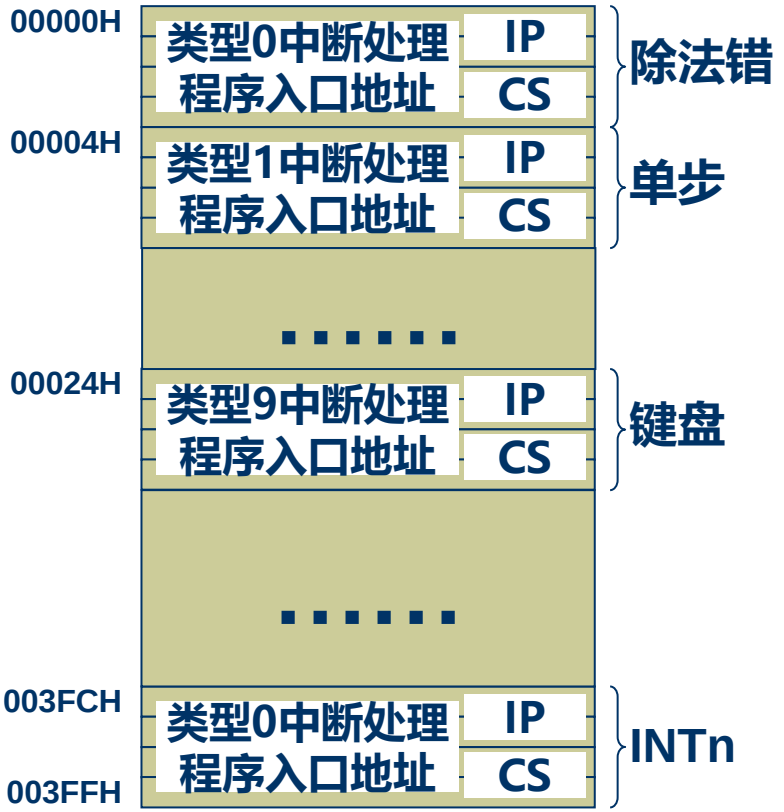
AH	功能	返回参数
0	从键盘读一字符	AL=字符码 AH=扫描码
1	读键盘缓冲区的字符	如ZF=0 AL=字符码 AH=扫描码 如 ZF=1，缓冲区空
2	取键盘状态字	AL=键盘状态字

BIOS Keyboard Support Functions

Function # (AH)	Input Parameters	Output Parameters	Description
0		al- ASCII character ah- scan code	Read character. Reads next available character from the system's type ahead buffer. Wait for a keystroke if the buffer is empty.
1		ZF- Set if no key. ZF- Clear if key available. al- ASCII code ah- scan code	Checks to see if a character is available in the type ahead buffer. Sets the zero flag if not key is available, clears the zero flag if a key is available. If there is an available key, this function returns the ASCII and scan code value in ax. The value in ax is undefined if no key is available.
2		al- shift flags	Returns the current status of the shift flags in al. The shift flags are defined as follows: bit 7: Insert toggle bit 6: Capslock toggle bit 5: Numlock toggle bit 4: Scroll lock toggle bit 3: Alt key is down bit 2: Ctrl key is down bit 1: Left shift key is down bit 0: Right shift key is down

如何从键盘直接读扫描码？

1. 查询方式：
   - 给8259的21H端口送控制命令，关掉键盘中断允许位
   - 通过键盘接口，编写查询程序
2. 中断方式：
   - 主程序中，修改向量表中键盘硬件中断向量指向自己的中断处理程序
   - 编写中断程序，读键盘扫描码，按自己需要进行处理后放到一个自定义内存缓冲区
   - 主程序中，从自定义内存缓冲区读键盘相关数据进行处理



注意关/开中断、中断向量保存/恢复等要适时、正确

想读读自己机器的BIOS键盘软中断程序吗？（BIOS键盘中断INT 16H）

1、计算BIOS键盘中断向量地址

16H*4=0058H

2、看中断向量表，获得中断程序入口

```
-d 0000:0058
0000:0050                                C4 09 10 02 8B 05 10 02
```

IP　　CS

3、反汇编，读程序

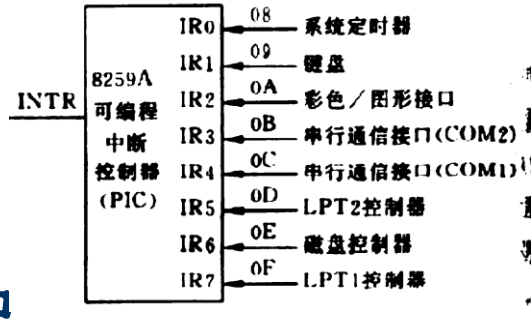
```
-u 0210:09c4
0210:09C4 1E              PUSH    DS
0210:09C5 53              PUSH    BX
0210:09C6 BB4000          MOV     BX,0040
0210:09C9 8EDB            MOV     DS,BX
0210:09CB 80FC10          CMP     AH,10
0210:09CE E8E8FD          CALL    07B9
0210:09D1 7203            JB      09D6
0210:09D3 E9E000          JMP     0AB6
0210:09D6 0AE4            OR      AH,AH
0210:09D8 743E            JZ      0A18
0210:09DA FECC            DEC     AH
0210:09DC 7474            JZ      0A52
0210:09DE FECC            DEC     AH
0210:09E0 7411            JZ      09F3
0210:09E2 FECC            DEC     AH
```

内存的键盘缓冲区KB_BUFFER：

```
0040:001A  BUFF_HEAD DW ?
0040:001C  BUFF_TAIL DW ?
0040:001E  KEY_BUFFER DW 16 DUP(?)
0040:003E  KEY_BUFFER_END LABEL WORD
```

如何读DOS或BIOS中的中断程序？
（进一步学习汇编和微机接口原理的方法）

找相关文档，或者按如下方式：



1. Debug中寻找中断程序入口地址
   - 如BIOS中的键盘硬件中断程序入口
     - Debug中显示中断向量（键盘中断向量地址09H*4=0024H）

```
-d 0000:0024
0000:0020                      0A 04 0E 02-3A 00 98 03 54 00 98 03     ....:...T...
0000:0030  6E 00 98 03 0A 00 98 03-A2 00 98 03 FF 03 0E 02   n...............
0000:0040  A9 00 0E 02 A4 07 0E 02-AA 09 0E 02 5D 04 0E 02   ............]...
0000:0050  B0 07 0E 02 0D 02 DD 02-C4 09 0E 02 0B 05 0E 02   ................
0000:0060  0E 0C 0E 02 14 0C 0E 02-1F 0C 0E 02 AD 06 0E 02   ................
0000:0070  AD 06 0E 02 A4 F0 00 F0-37 05 0E 02 F1 0F 00 C0   ........7.......
0000:0080  72 10 A7 00 7C 10 A7 00-4F 03 FF 0D 8A 03 FF 0D   r...|...O.......
0000:0090  17 03 FF 0D 86 10 A7 00-90 10 A7 00 9A 10 A7 00   ................
0000:00A0  B8 10 A7 00                                       ....
-
```

2. Debug中反汇编中断程序

```
-u 020e:040a
020E:040A 50              PUSH    AX
020E:040B 33C0            XOR     AX,AX
020E:040D C4C4            LES     AX,SP
020E:040F 0958E9          OR      [BX+SI-17],BX
020E:0412 3802            CMP     [BP+SI],AL
020E:0414 90              NOP
020E:0415 90              NOP
020E:0416 E460            IN      AL,60
020E:0418 90              NOP
020E:0419 90              NOP
020E:041A C4C4            LES     AX,SP
```

9.2 显示I/O

9.2.1 显示器简介

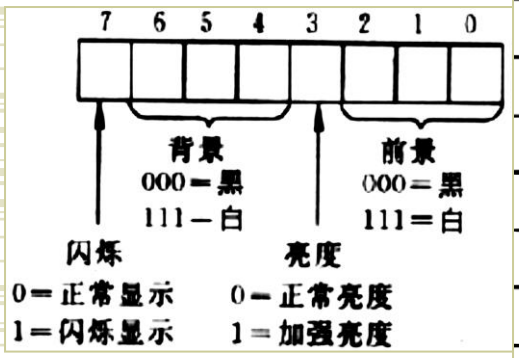
(1) 显示属性

- 屏幕上显示的字符取决于字符码及字符显示属性

(2) 显示缓冲区

- 显示适配卡带有显示存储器，用于存放屏幕上显示文本的代码及属性或图形信息
- 显示存储器作为系统存储器的一部分，可用访问普通内存的方法访问显示存储器
  - 显示屏幕是“存储器映像”
- 直接存取显示存储器内容进行显示的方法称为直接写屏

- 在单色显示时，字符显示属性定义了闪烁、反向和高亮度等显示特性



属性值（二进制）	属性值（十六进制）	显示效果
00000000	00	无显示
00000001	01	黑底白字，下划线
00000111	07	黑底白字，正常显示
00001111	0F	黑底白字，高亮度
01110000	70	白底黑字，反相显示
10000111	87	黑底白字，闪烁
11110000	F0	白底黑字，反相闪烁

- 在彩色显示时，字符显示属性定义了前景色和背景色

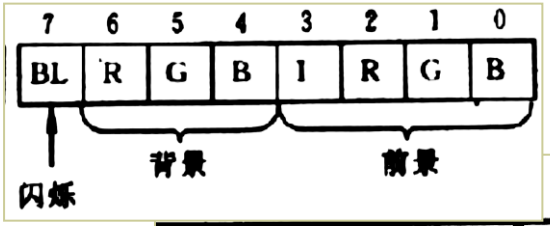


表 9.7 16种颜色的组合

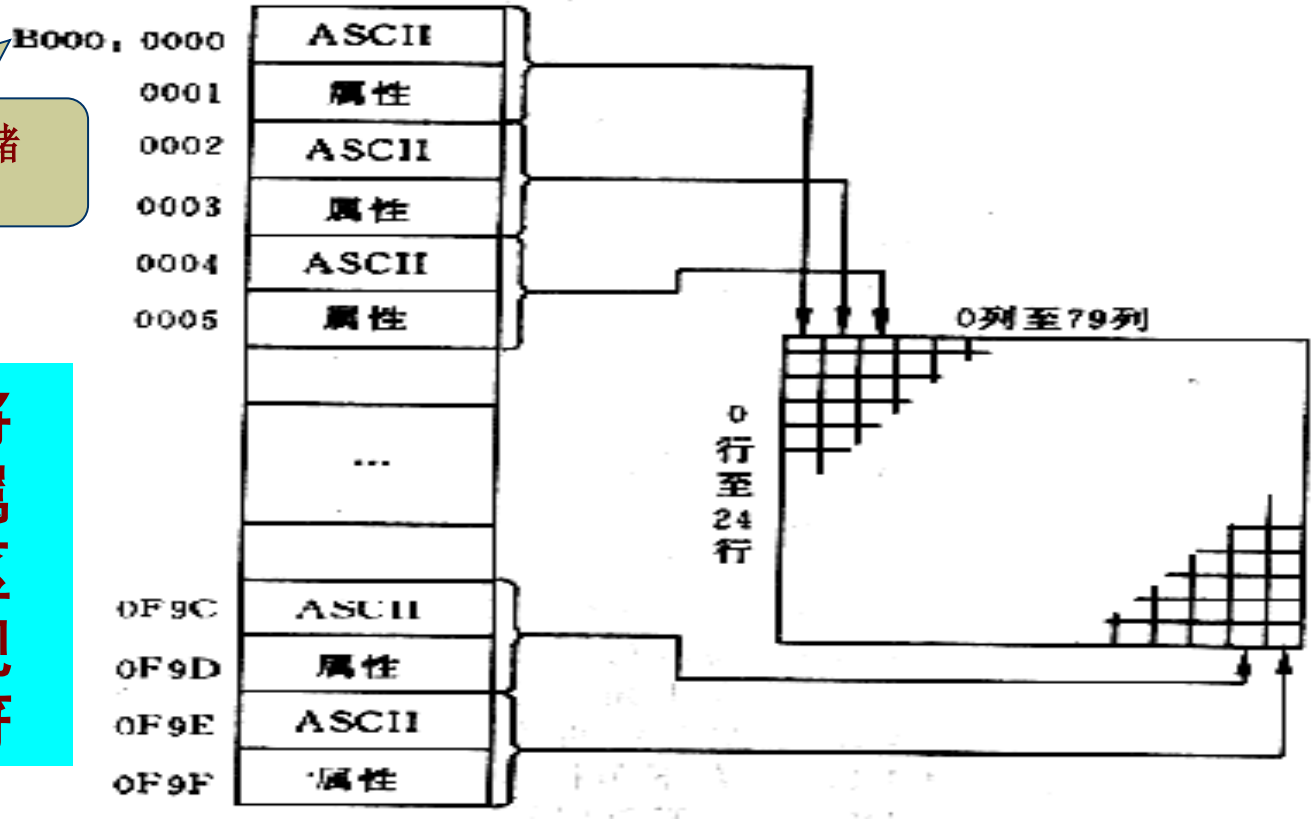
颜色	IRGB	颜色	IRGB	颜色	IRGB	颜色	IRGB
黑	0000	灰	1000	红	0100	浅红	1100
蓝	0001	浅蓝	1001	品红	0101	浅品红	1101
绿	0010	浅绿	1010	棕	0110	黄	1110
青	0011	浅青	1011	灰白	0111	白	1111

每个显示字符占2个存储单元

- 25*80的单色显示文本方式下
  - 屏幕可有2000个字符位置，显存容量需要4000B≈4KB
  - 如果显存有16KB容量，可保存4屏幕的字符数据，即4页数据
- 可根据显示位置的行列值算出显示存储区的地址
- 屏幕上某一字符在显存中的偏移地址

Char_offset=page_offset+((row*width)+column)*byte

注意：各种适配器的显示存储器（显存）的启始地址不同



可以用指令直接将要显示的字符和属性写入显示存储区的相应单元，实现在屏幕上显示字符

9.2.2 BIOS显示中断

- 显示器BIOS中断调用(INT 10H)

  显示器BIOS程序提供功能如下：

  设置显示模式

  设置光标类型、设置光标位置、读光标位置

  读取光标位置处的字符和属性

  将字符和属性写到光标位置处

  选择当前显示页

  向上滚屏、向下滚屏

AH	功 能	调用参数	返回参数 / 注释
1	置光标类型	(CH)$_{5\sim3}$ = 光标开始行 (CL)$_{0\sim3}$ = 光标结束行	
2	置光标位置	BH = 页号 DH = 行 DL = 列	
3	读光标位置	BH = 页号	CH = 光标开始行 CL = 光标结束行 DH = 行 DL = 列
4	置显示页	AL = 显示页号	
5	选择活动显示页		
6	屏幕初始化或上卷	AL = 上卷行数 AL = 0 全屏幕为空白 BH = 卷入行属性 CH = 左上角行号 CL = 左上角列号 DH = 右下角行号 DL = 右下角列号	
7	屏幕初始化或下卷	AL = 下卷行数 AL = 0 全屏幕为空白 BH = 卷入行属性 CH = 左上角行号 CL = 左上角列号 DH = 右下角行号 DL = 右下角列号	
8	读光标位置的属性和字符	BH = 显示页	AH = 属性 AL = 字符
9	在光标位置显示字符及其属性	BH = 显示页 AL = 字符 BL = 属性 CX = 字符重复次数	
A	在光标位置只显示字符	BH = 显示页 AL = 字符 CX = 字符重复次数	
E	显示字符(光标前移)	AL = 字符 BL = 前景色	光标跟随字符移动
13	显示字符串	ES:BP = 串地址 CX = 串长度 DH, DL = 起始行列 BH = 页号 AL = 0, BL = 属性 串:Char,char,.....,char	光标返回起始位置
		AL = 1, BL = 属性 串:Char,char,.....,char	光标跟随移动
		AL = 2 串:Char,attr,.....,char,attr	光标返回起始位置
		AL = 3 串:Char,attr,.....,char,attr	光标跟随串移动

对某个窗口操作

INT 10H

实质是对显示存储区的操作

INT 10H (ah=01h)

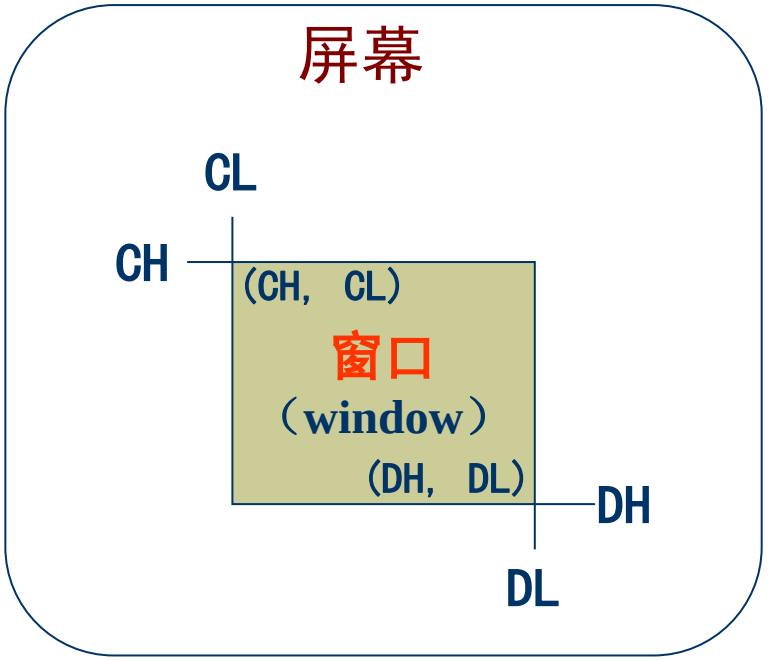
- AH = 01h CH = Scan Row Start, CL = Scan Row End
- Normally a character cell has 8 scan lines, 0-7. So, CX=0607h is a normal underline cursor, CX=0007h is a full-block cursor. If bit 5 of CH is set, that often means "Hide cursor". So CX=2607h is an invisible cursor.
- Some video cards have 16 scan lines, 00h-0Fh.
- Some video cards don't use bit 5 of CH. With these, make Start>End (e.g. CX=0706h)

例如： 窗口初始化或字符上卷

6	屏幕初始化或上卷	AL = 上卷行数 AL = 0 全屏幕为空白 BH = 卷入行属性 CH = 左上角行号 CL = 左上角列号 DH = 右下角行号 DL = 右下角列号
7	屏幕初始化或下卷	AL = 下卷行数 AL = 0 全屏幕为空白 BH = 卷入行属性 CH = 左上角行号 CL = 左上角列号 DH = 右下角行号 DL = 右下角列号

行向上移位

- AL>0时，要上卷的行数
  - 窗口上卷时，低端的行由空行代替，属性由BH决定
  - 上卷时移出窗口的行不能恢复
- AL=0时， 全窗口为空白
- BH=空白区域的视频属性
- CH, CL=窗口左上角的行列位置
- DH, DL=窗口右下角的行列位置



显示器BIOS中断应用举例

例1、 在当前光标位置处显示5个字符U (UUUUU)，但不移动光标

- 9号子功能调用

9	在光标位置显示 字符及其属性	BH = 显示页 AL = 字符 BL = 属性 CX = 字符重复次数

- 指令序列如下：

```
MOV     BH, 0          ; 显示页号，第0页
MOV     AL, 'U'        ; 显示字符的代码
MOV     BL, 4EH        ; 显示字符的属性，红底黄字
MOV     CX, 5          ; 字符重复次数
MOV     AH, 9          ; 显示I/O中断程序的功能号
INT     10H            ; 中断调用指令
```

例2、 清屏并把光标设置在左上角

- 在窗口滚屏时，如果滚屏行数为0，就表示清除整个窗口。设屏幕为25*80，先清除屏幕，然后把光标设定到左上角

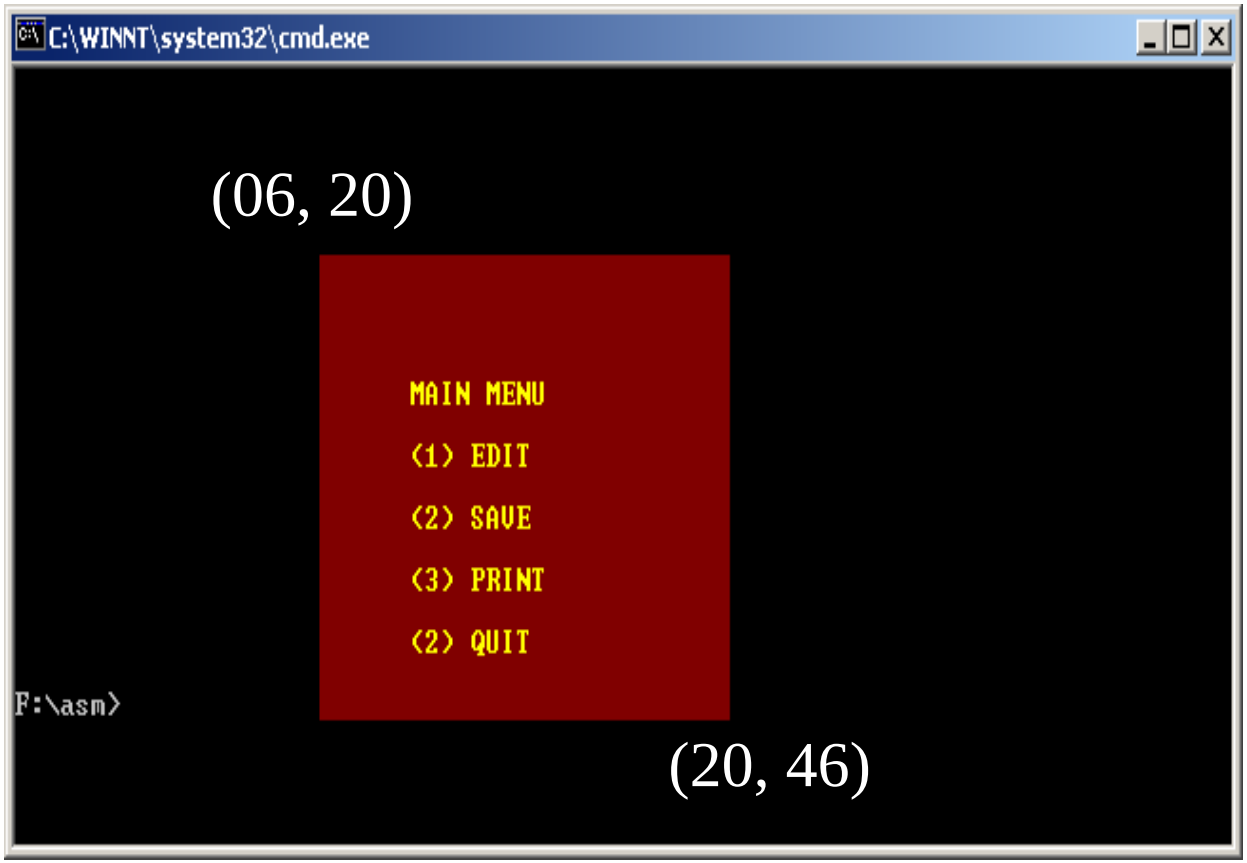
```
;清屏
        mov ah, 6
        mov al, 0
        mov bh, 7 ; 黑低白字
        mov cx, 0
        mov dh, 24
        mov dl, 79
        int 10h
;光标设置在左上角
        mov   dx,   0
        mov   ah,   2
        int 10h
```

6	屏幕初始化或上卷	AL = 上卷行数 AL = 0 全屏幕为空白 BH = 卷入行属性 CH = 左上角行号 CL = 左上角列号 DH = 右下角行号 DL = 右下角列号

2	置光标位置	BH = 页号 DH = 行 DL = 列

例3、 编制一程序。要求：在干净的屏幕上开一窗口（红底黄字），左上角坐标（06H，14H），右下角坐标（14H，2EH）；在窗口内显示：

MAIN MENU
Edit
Save
Print
Quit



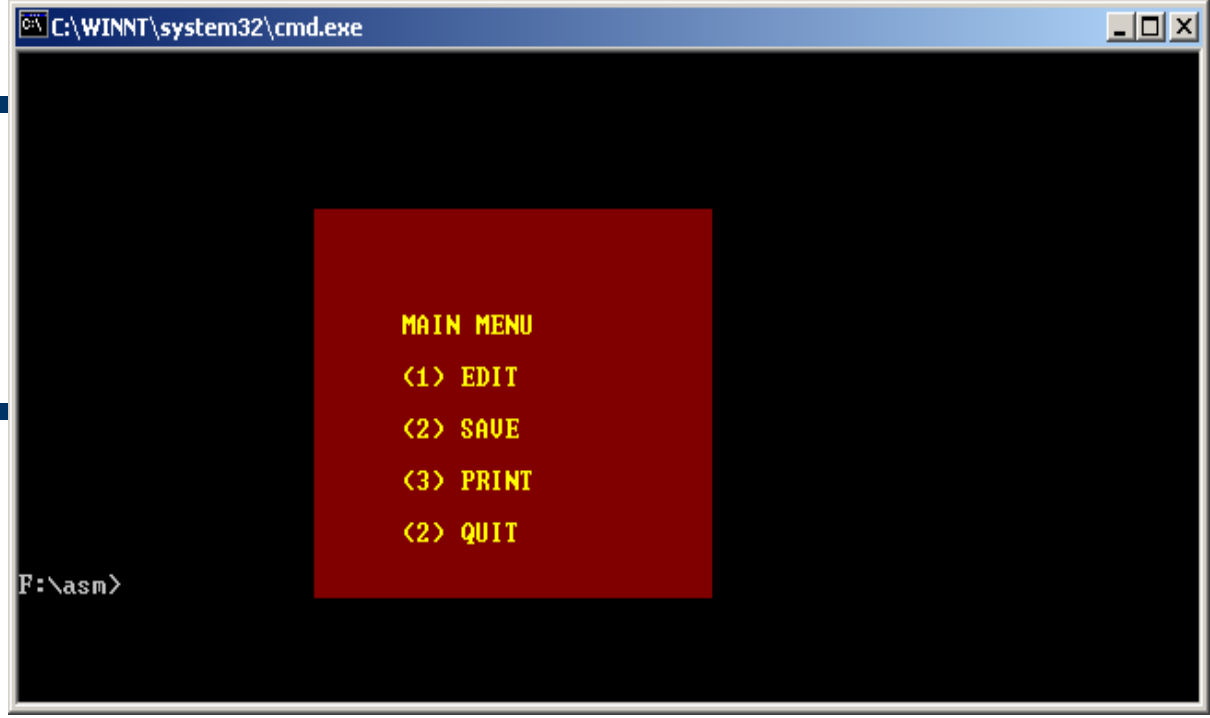
程序清单

```
; 开设空白窗口的宏定义
clrscr  macro  lu, rd, fb    ; lu=左上角坐标，rd=右下角坐标
                             ; fb=前景色和背景色
        mov    ax, 0600h
        mov    cx, lu
        mov    dx, rd
        mov    bh, fb
        int    10h
        endm

; 设置光标的宏定义
cursor  macro  row, col    ; row=行号，col=列号
        mov    bh, 0
        mov    dh, row
        mov    dl, col
        mov    ah, 2
        int    10h
        endm
```

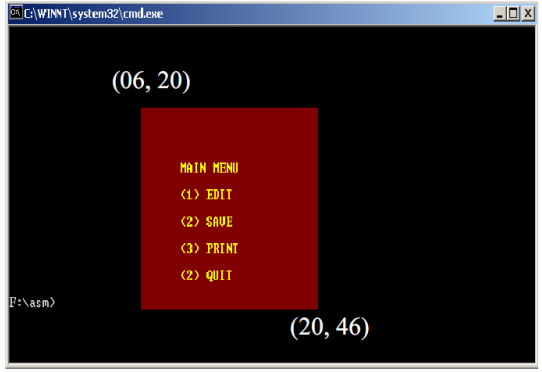
6	屏幕初始化或上卷	AL = 上卷行数 AL = 0 全屏幕为空白 BH = 卷入行属性 CH = 左上角行号 CL = 左上角列号 DH = 右下角行号 DL = 右下角列号

2	置光标位置	BH = 页号 DH = 行 DL = 列




```
; 定义数据段
dseg    segment
; 每个字符串以’$’结束利于判断
d1      db      'MAIN MENU', 0dh, 0ah, '$'
        db      '(1) Edit',0dh,0ah,'$'
        db      '(2) Save',0dh,0ah,'$'
        db      '(3) Print',0dh,0ah,'$'
        db      '(4) Quit',0dh,0ah,'$'
dseg    ends
```

```
;定义代码段
cseg    segment
        assume  cs:cseg,ds:dseg
start   proc    far
        mov     ax, dseg
        mov     ds, ax
        clrscr  0, 184fh, 07h          ; 清屏，黑底白字
        clrscr  0614h, 142eh, 4Eh      ; 开窗口，红底黄字
        cursor  10,26                  ; 菜单首行光标
        mov     si, offset d1
        call    dischs                 ; 显示菜单首行
        cursor  12, 26                 ; 菜单第二行光标
        call    dischs                 ; 显示菜单第二行
        cursor  14, 26
        call    dischs
        cursor  16, 26
        call    dischs
        cursor  18, 26
        call    dischs
        mov     ah, 4ch
        int     21h
start   endp
; 以‘$’结束的字符串显示的子程序
```




```
d1      db      'MAIN MENU', 0dh, 0ah, '$'
        db      '(1) Edit',0dh,0ah,'$'
        db      '(2) Save',0dh,0ah,'$'
        db      '(3) Print',0dh,0ah,'$'
        db      '(4) Quit',0dh,0ah,'$'
dseg    ends
```

```
dischs  proc    near
disch:  mov     al, [si]
        inc     si
        cmp     al, '$'
        je      exit
        mov     bx, 13h
        mov     ah, 0eh
        int     10h
        jmp     disch
exit:   ret
dischs  endp

cseg    ends
        end     start
```

E	显示字符 (光标前移)	AL = 字符 BL = 前景色	光标跟随字符移动

9.2.3 DOS显示功能调用

- DOS显示功能调用中断为 INT21H
- INT 21H显示操作

AH	功 能	调 用 参 数
2	显示一个字符（检查Ctrl-Break）	DL=字符， 光标跟随字符移动
6	显示一个字符（不检查Ctrl-Break）	DL=字符， 光标跟随字符移动
9	显示字符串	DS:DX=串地址， 串必须以 $ 结束，光标跟随串移动

- 不能有控制码的ASCII码
- 检查Ctrl-Break:
- 显示字符串中：串必须以$结束；如果希望光标能自动换行，在字符串结束符$前 加回车和换行的ASCII码

AH	功 能	调 用 参 数
9	显示字符串	DS:DX=串地址 串必须以$结束，光标跟随串移动

例1、显示一串字符

```
; 字符串的数据定义
    CR        EQU  0DH
    LF        EQU  0AH
    TAB       EQU  09H
    MESSAGE   DB   TAB, 'The sort operation is finished.'
              DB   CR, LF, '$'

; 显示字符串的指令
    MOV AH, 09H
    MOV DX, SEG MESSAGE
    MOV DS, DX
    MOV DX, OFFSET MESSAGE
    INT 21H
```

9.3 打印机

- 打印机I/O中断
  - DOS中断调用　INT 21H
  - BIOS中断调用　INT 17H

表 9.11　打印机 I/O 中断

INT	AH	功　能	调用参数	返回参数
21H	5	打印一个字符	DL = 字符	
17H	0	打印一个字符	AL = 字符	AH = 状态字节
		并回送状态字节	DX = 打印机号	
17H	1	初始化打印机	DX = 打印机号	AH = 状态字节
		回送状态字节		
17H	2	回送状态字节	DX = 打印机号	AH = 状态字节

- **主机输出给打印机的信息包括两类：**
  - **字符码（ASCII）**
  - **完成一定特定动作的控制码或功能码**

表9.12 打印机常用的标准控制字符

功能码	十进制	十六进制（ASCII 码）	功能含义
SP	08	08	空格
HT	09	09	水平制表（Tab）
LF	10	0A	换行
VT	11	0B	垂直制表（Tab）
FF	12	0C	换页
CR	13	0D	回车

- **水平制表：仅当打印机有此功能，并被置成打印机Tab状态时才能实现**
  否则，不执行此命令，或打印多个空格代替Tab
  Tab相当于8个字符宽度

许多打印机不认识TAB字符(09H)，这时程序就要检查TAB字符，若输出的字符是TAB，就要插入空格，把当前光标位置移到8，16，24，...字符位置上

INT	AH	功　能	调用参数	返回参数
21H	5	打印一个字符	DL = 字符	

9.3.1 dos打印功能

- 打印一个字符：INT 21H，AH=5，DL=字符
- 如果需要回车、换行等打印机控制功能，必须由汇编程序送出回车、换行等控制码给DOS

```
TEXT    DB      0CH, 'Hello, everybody!', 0DH, 0AH, 0AH
COUNT   EQU     $-TEXT
......
        MOV     CX,   COUNT
        MOV     BX,   0
NEXT:   MOV     AH,   5
        MOV     DL,   TEXT[BX]
        INT     21H
        INC     BX
        LOOP    NEXT
```

0CH：换页
0DH：回车
0AH：换行

DOS打印机操作自动测试打印机状态

9.3.2 BIOS打印功能

- BIOS提供的打印I/O程序中断号为17H
- 系统可连接多台打印机，用打印机号选择打印机，打印机号为0，1，2

INT	AH	功 能	调用参数	返回参数
17H	0	打印一个字符	AL = 字符	AH = 状态字节
		并回送状态字节	DX = 打印机号	
17H	1	初始化打印机	DX = 打印机号	AH = 状态字节
		回送状态字节		
17H	2	回送状态字节	DX = 打印机号	AH = 状态字节

INT	AH	功　能	调用参数	返回参数
17H	0	打印一个字符 并回送状态字节	AL = 字符 DX = 打印机号	AH = 状态字节

例1、将AL中的字符输出到打印机

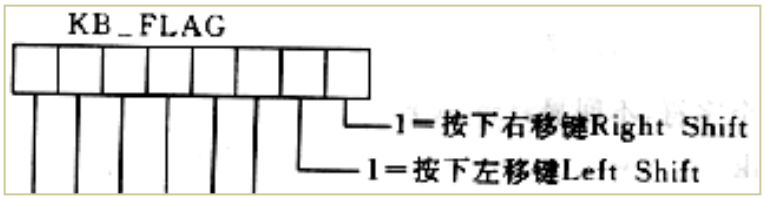
```
MOV    AH，0
MOV    AL，CHAR
MOV    DX，0
INT    17H
```

例2、将键盘接收到的字符显示在显示器上，并由打印机输出，当键盘上按下shift键即退出程序。

```
; 定义堆栈段
sseg        segment stack
            dw  100 dup(?)
tos         label   word
sseg        ends

; 定义代码段
cseg        segment
            assume cs:cseg, ss:sseg
key_prt     proc        far
            mov         ax, sseg
            mov         ss, ax
            mov         sp, offset  tos   ; 设置堆栈指针
            call        cls               ; 调用清屏子程序
            mov         ah, 2
            mov         dl, 0ah
            int         21h               ; 光标指向下一行
key_ch:
            mov         ah,2
            int         16h               ; 取变换键状态
            test        al, 03h           ; 有shift按下吗？字节的低两位为 l_shift + r_shift的状态
            jnz         endprog           ; 有shift键按下则转移到程序结束
```

AH	功 能	调 用 参 数
2	显示一个字符（检查 Ctrl-Break）	DL=字符， 光标跟随字符移动



2	取键盘状态字节	AL＝键盘状态字节

```
        mov     ah,1
        int     16h          ; 判断键盘有键可读吗?
        jz      key_ch       ; 无键可读，转去读键盘状态
        mov     ah,0
        int     16h          ; 读取键盘字符
        push    ax
        mov     dl, al
        mov     ah, 2        ; 显示所读的字符
        int     21h
        pop     ax
        push    ax
        cmp     al, 0dh      ; 是回车，加上换行符
        jne     nnn
        mov     dl, 0ah      ; 换行符
        mov     ah, 2
        int     21h
nnn:                         ; 打印键盘键入的字符
        mov     ah, 0
        mov     dx, 0
        int     17h          ; 打印al中的字符
        pop     ax
        cmp     al, 0dh      ; 是回车，加上换行符
        jne     key_ch
        mov     al, 0ah
        mov     ah,0
        mov     dx,0
        int     17h
        jmp     key_ch
endprog:                     ; 返回dos
        mov     ah,4ch
        int     21h
key_prt endp
```

AH	功 能	返回参数
0	从键盘读一字符	AL＝字符码 AH＝扫描码
1	读键盘缓冲区的字符	如 ZF＝0 AL＝字符码 AH＝扫描码 如 ZF＝1，缓冲区空

AH	功 能	调 用 参 数
2	显示一个字符（检查 Ctrl-Break）	DL=字符， 光标跟随字符移动

17H	0	打印一个字符 并回送状态字节	AL = 字符 DX = 打印机号	AH = 状态字节

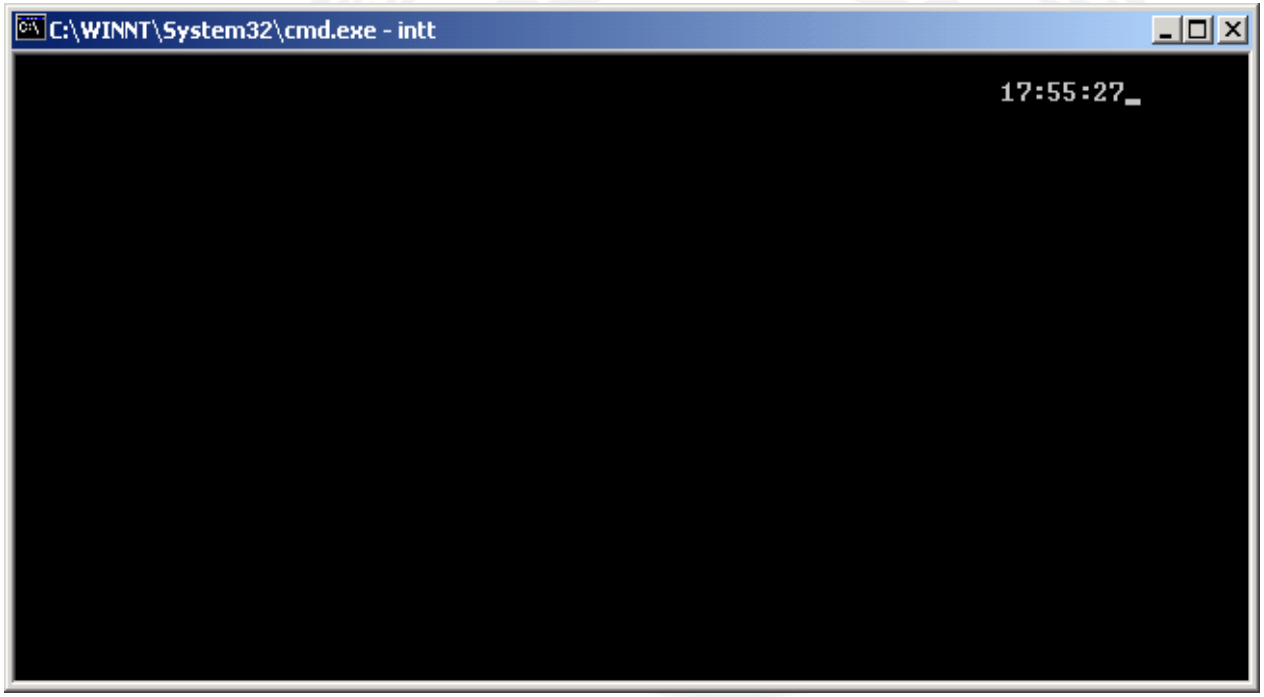
```
;清屏子程序
cls     proc          near
            mov     ax, 0600h
            mov     cx, 0
            mov     dx, 184fh
            mov     bh, 36h
            int     10h
            ret
cls         endp

cseg        ends
            end     key_prt
```

6	屏幕初始化或上卷	AL = 上卷行数 AL = 0 全屏幕为空白 BH = 卷入行属性 CH = 左上角行号 CL = 左上角列号 DH = 右下角行号 DL = 右下角列号

9.4 BIOS中断举例

例1、 显示实时钟，遇到按键时退出



- 系统加电期间，系统定时器初始化为每隔55毫秒发一次中断请求，每秒要调用约18.2次系统时钟中断处理程序
- 参看“表9.1BIOS中断类型---8259中断类型”
  - CPU在响应定时中断请求后转入08H号中断处理程序
  - BIOS提供的08H号中断处理程序中，有一条软中断指令“INT 1CH”，而BIOS的1CH号中断处理程序处只有一条中断返回指令（IRET），实际上并没有作任何工作
  - 这样安排的目的是为应用程序留下一个软接口，应用程序只要修改中断向量表中1CH号中断向量，转向新的中断处理程序，就能实现某些周期性的工作

- 新的1CH号中断处理程序功能如下：
  - 清屏，利用1AH号中断处理程序的2号功能获取当前时间；
  - 在屏幕的右上角显示当前时间；
  - 记录调用该中断处理程序的次数，当计数满18次后(1秒钟到)，更新显示时间
- 主程序的功能：
  - 保存原1CH号的中断向量；
  - 设置新的1CH号的中断向量；
  - 在主程序完成其他工作后，恢复原1CH号的中断向量

```
; 定义代码段
cseg    segment
        assume cs:cseg, ds:cseg
start   proc    far
        push    cs
        pop     ds
        mov     ax, 351ch
        int     21h                          ; 获取原1CH号的中断向量→ES:BX
        mov     ds:word ptr old1c,  bx
        mov     ds:word ptr old1c+2, es      ; 保存原1CH号中断向量
        mov     dx, offset int1c             ; 设置新1CH号中断向量
        mov     ax, 251ch
        int     21h
; 此后，每55毫秒就进入一次新的1CH号的中断处理程序
waitn:  mov     ah, 1
        int     16h                          ; 查有无键按下
        jz      waitn                        ; 转等待键按下
        mov     ah, 0
        int     16h                          ; 读键盘
        lds     dx, ds:old1c
        mov     ax, 251ch
        int     21h                          ; 恢复原1ch中断向量
        mov     ah, 4ch
        int     21h                          ; 返回dos
start   endp
```

```
;定义数据空间
old1c   dd      ?               ;  保存原中断向量
count   dw      0               ;  调用1ch中断程序的次数
hhh     db      ?,?,':'         ;  保存:  “时”
mmm     db      ?,?,':'         ;  保存:  “分”
sss     db      ?,?,‘$’         ;  保存:  “秒”
```

数据是代码段的一部分，不提倡这种编程方法！！！

```
;定义新的1CH号的中断处理程序
int1c   proc   far
        cmp     count, 0
; 调用次数为“0”时（1秒钟到），显示系统时间
        jz      next
        dec     count   ; 显示次数递减（第一次18-1）
        iret
next:
        mov     count, 18   ; 置计数次数初值
        sti
        push    ds                  ; 保护现场
        push    es
        push    ax
        push    bx
        push    cx
        push    dx
        push    si
        push    di
        mov     ah, 2
        int     1ah                 ; 读实时时钟
        mov     al, ch              ; 时送al
        call    ttasc               ; 转换成ascii码
        mov     word    ptr hhh, ax     ; 保存时
        mov     al, cl              ; 分转换
        call    ttasc
        mov     word    ptr mmm, ax
        mov     al, dh              ; 秒转换
        call    ttasc
        mov     word    ptr sss, ax
        call    cls         ; 清屏
        mov     bh, 0
        mov     dx, 0140h
        mov     ah, 2
        int     10h     ; 设置光标（2，65）
        push    cs
        pop     ds
        mov     dx, offset hhh
        mov     ah, 9
        int     21h         ; 显示实时时钟
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        pop     es
        pop     ds
        iret                ; 中断返回
int1c   endp
```

```
;将AL中的BCD数据转换成ASCII码存入AX中
ttasc   proc
        push    cx
        mov     ah, al
        and     al, 0fh
        mov     cl, 4
        shr     ah, cl
        add     ax, 3030h
        xchg    ah, al
        pop     cx
        ret
ttasc   endp

;清屏子程序
cls     proc
        mov     ax, 0600h
        mov     cx, 0
        mov     dx, 184fh
        mov     bh, 7
        int     10h
        ret
cls     endp
cseg    ends
        end     start
```

MOV AH, 02
INT 1A

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Pr...

```
-q

C:\CH9>debug TEST2.EXE
-t

AX=0776  BX=0000  CX=0067  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0766  ES=0766  SS=0775  CS=0777  IP=0003   NV UP EI PL NZ NA PO NC
0777:0003 8ED8          MOV     DS,AX
-t

AX=0776  BX=0000  CX=0067  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0776  ES=0766  SS=0775  CS=0777  IP=0005   NV UP EI PL NZ NA PO NC
0777:0005 B402          MOV     AH,02
-t

AX=0276  BX=0000  CX=0067  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=0776  ES=0766  SS=0775  CS=0777  IP=0007   NV UP EI PL NZ NA PO NC
0777:0007 CD1A          INT     1A
-
-p

AX=0276  BX=0000  CX=2219  DX=4500  SP=0000  BP=0000  SI=0000  DI=0000
DS=0776  ES=0766  SS=0775  CS=0777  IP=0009   NV UP EI PL NZ NA PO NC
0777:0009 8AC5          MOV     AL,CH
```

变量定义在程序代码段中

```
; 定义代码段
cseg   segment
       assume cs:cseg
start  proc   far
       push   cs
       pop    ds
       mov    ax, 351ch
       int    21h                    ; 获取原1CH号的中断向量
       mov    cs:word ptr old1c, bx
       mov    cs:word ptr old1c+2, es        ; 保存原1CH号中断向量
       mov    dx, offset int1c       ; 设置新1CH号中断向量
       mov    ax, 251ch
       int    21h
; 此后，每55毫秒就进入一次新的1CH号的中断处理程序
waitn: mov    ah, 1
       int    16h                    ; 查有无键按下
       jz     waitn                  ; 转等待键按下
       mov    ah, 0
       int    16h                    ; 读键盘
       lds    dx, cs:old1c
       mov    ax, 251ch
       int    21h                    ; 恢复原1ch中断向量
       mov    ah, 4ch
       int    21h                    ; 返回dos
start  endp

;定义数据空间
old1c  dd     ?            ; 保存原中断向量
count  dw     0            ; 调用1ch中断程序的次数
hhh    db     ?,?,':'      ; 保存：“时”
mmm    db     ?,?,':'      ; 保存：“分”
sss    db     ?,?,'$'      ; 保存：“秒”
```

```
;定义新的1CH号的中断处理程序
int1c  proc   far
       cmp    count, 0
; 调用次数为“0”时（1秒钟到），显示系统时间
       jz     next
       dec    count   ; 显示次数递减（第一次18-1）
       iret
next:
       mov    count, 18   ; 置显示次数初值
       sti
       push   ds                 ; 保护现场
       push   es
       push   ax
       push   bx
       push   cx
       push   dx
       push   si
       push   di
       mov    ah, 2
       int    1ah                ; 读实时时钟
       mov    al, ch             ; 时送al
       call   ttasc              ; 转换成ascii码
       mov    word   ptr hhh, ax   ; 保存时
       mov    al, cl             ; 分转换
       call   ttasc
       mov    word   ptr mmm, ax
       mov    al, dh             ; 秒转换
       call   ttasc

       mov    word   ptr sss, ax
       call   cls       ; 清屏
       mov    bh, 0
       mov    dx, 0140h
       mov    ah, 2
       int    10h    ; 设置光标（2，65）
       push   cs
       pop    ds
       mov    dx, offset hhh
       mov    ah, 9
       int    21h      ; 显示实时时钟
       pop    di
       pop    si
       pop    dx
       pop    cx
       pop    bx
       pop    ax
       pop    es
       pop    ds
       iret            ; 中断返回
int1c  endp
```

```
;将AL中的BCD数据转换成ASCII码存入AX中
ttasc  proc
       push   cx
       mov    ah, al
       and    al, 0fh
       mov    cl, 4
       shr    ah, cl
       add    ax, 3030h
       xchg   ah, al
       pop    cx
       ret
ttasc  endp

;清屏子程序
cls    proc
       mov    ax, 0600h
       mov    cx, 0
       mov    dx, 184fh
       mov    bh, 7
       int    10h
       ret
cls    endp
cseg   ends
       end    start
```

谢谢！