16位MS-DOS/BIOS程序设计

郑贵滨

2021年6月14日

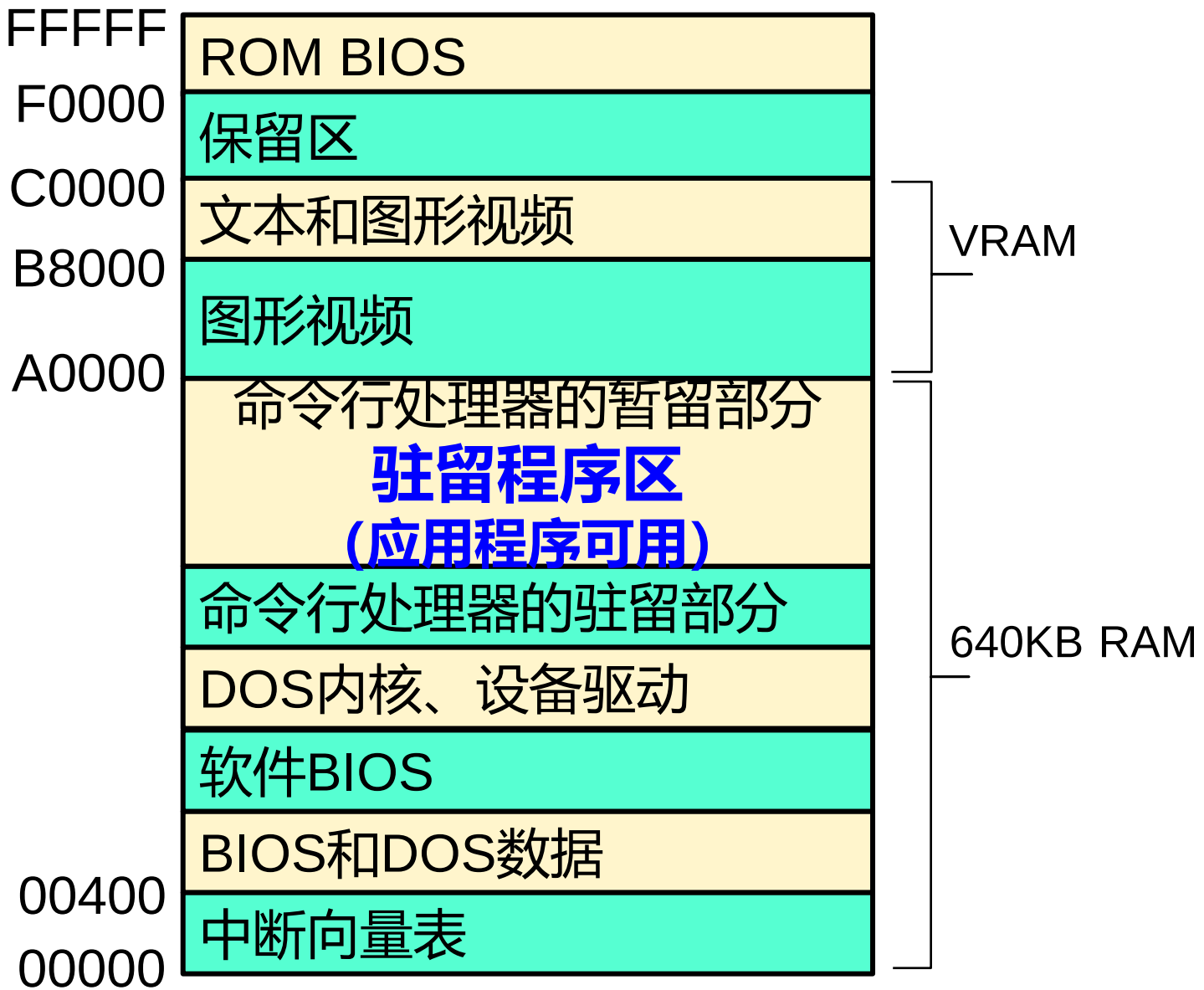
主要内容

- 1、MS-DOS16位程序设计
  - MS - DOS
  - 软件中断
  - 16位程序编写
  - MS - DOS功能调用
- 2、BIOS程序设计
  - INT16h BIOS键盘中断

1.1、DOS

- DOS
  - 首个实地址模式操作系统（16位）
  - 单用户单任务操作系统
  - 设计运行于**8086和8088**处理器
  - 也可运行于IA-32处理器的实地址工作方式
- 实地址模式通常称为16位模式
- 实地址模式程序的特点
  - 只能寻址1MB内存
  - 一次任务中只能运行一个程序（单任务）
  - 内存没有边界保护机制
  - 程序可以访问任意资源
  - 偏移量是16位的

1.1.1 DOS的内存组织



1.1.2 实地址存储模型

- 主存空间1MB(= 2^{20} B)

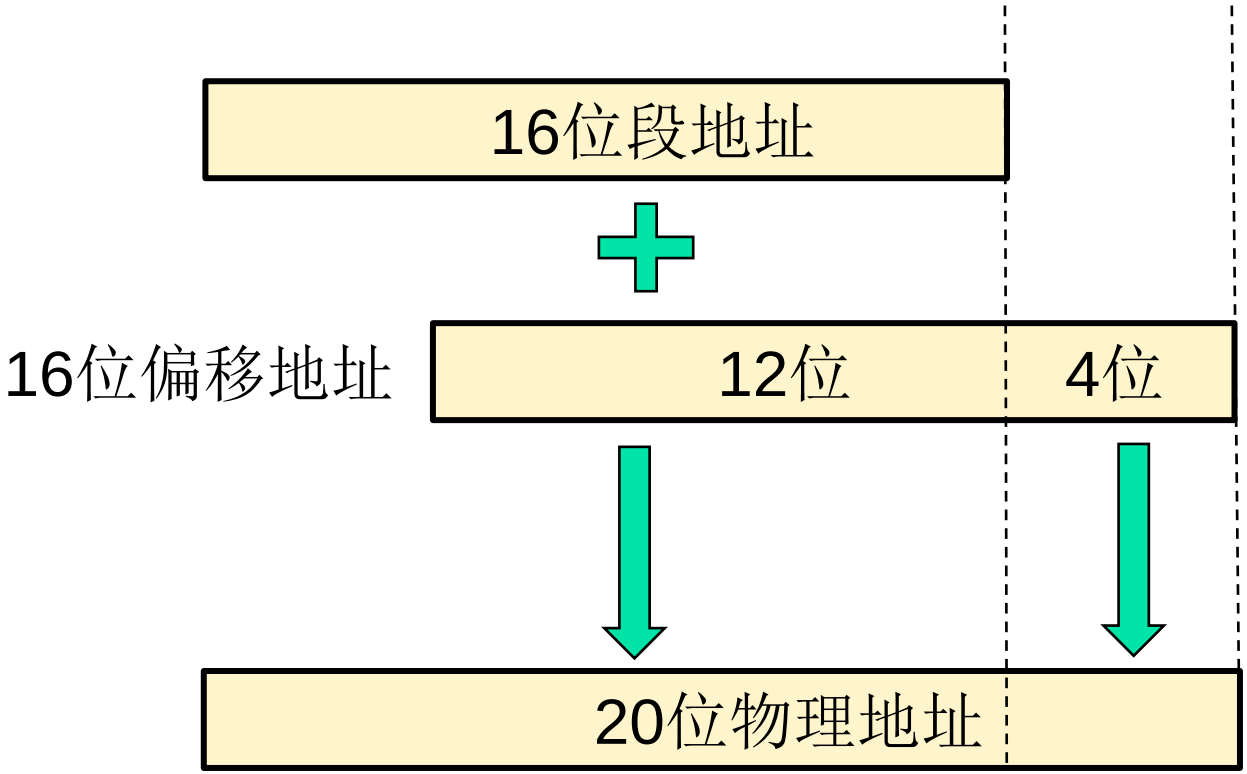
  00000H~FFFFFH

- 程序设计时分段管理，但有两个限制：
  - 每个段最大为64KB
  - 段只能开始于低4位地址全为0的物理地址处
- 逻辑地址 = 段地址:偏移地址
  - 16位段寄存器保存20位段起始地址的高16位
  - 偏移地址也用16位数据表示
- 物理地址 = 段地址×16 + 偏移地址

左移二进制**4**位（十六进制**1**位）

1.1.2 实地址存储模型

- 8086 物理地址的计算



1.2 16位程序的注意事项

- 16位DOS环境默认采用16位操作数和偏移量
- 堆栈以16位为单位压入PUSH和弹出POP数据
- IA-32处理器的实地址工作方式
  - 允许使用：32位寄存器、操作数和寻址方式
  - 大多数新增的32位通用指令
  - 运行于DOS、Win95/98/Me下的实地址程序可以访问硬件端口、中断向量和系统内存，在windows NT/2000/XP下不可以;
- **特别注意：偏移地址只有16位**

1.2 16位程序的注意事项

- 16位程序的编写

.MODEL small
.STACK 200H
.386 ;使用32位寄存器
如用变量（有数据段），需初始化**DS**
用伪指令**.startup**
或者：**mov ax, @data**
mov ds, ax

- 程序结束返回操作系统:
用伪指令**.exit**
或dos功能调用:
mov ah, 4ch
int 21h

1.2 16位程序的注意事项

- 16位存储器寻址方式

16位有效地址 = 基址寄存器 + 变址寄存器 + 位移量

  - 基址寄存器：BX、BP
  - 变址寄存器：SI、DI
  - 位移量：8或16位有符号值

- 多种主存寻址方式
  - 直接寻址　　MOV AX,WVAR
  - 寄存器间接寻址　　MOV AX,[BX]
  - 寄存器相对寻址　　MOV AX,[BP+4]
  - 基址变址寻址　　MOV AX,[BX+SI]
  - 相对基址变址寻址　　MOV AX,[BX+DI-2]

1.2 DOS应用程序框架

```
;example.asm in DOS
    .model small
    .stack  100h;设置堆栈大小
    .data          ;定义数据段
    ……             ;数据定义（数据待填）
    .code          ;定义代码段
start:             ;程序执行起始位置,或main proc
    mov ax,@data
    mov ds,ax
    ……             ;主程序（指令待填）
    .exit 0        ;程序正常执行结束
                   ;                    或 main endp
    ……             ;子程序（指令待填）
    end start      ;汇编结束          或 end main
```

1.3 软件中断

- 软件中断——运行中断指令产生中断
  - 调用操作系统(DOS)功能
  - 调用BIOS的功能
- 中断指令

INT i8;中断调用指令：调用i8号中断服务程序

IRET;中断返回指令：从服务程序返回主程序

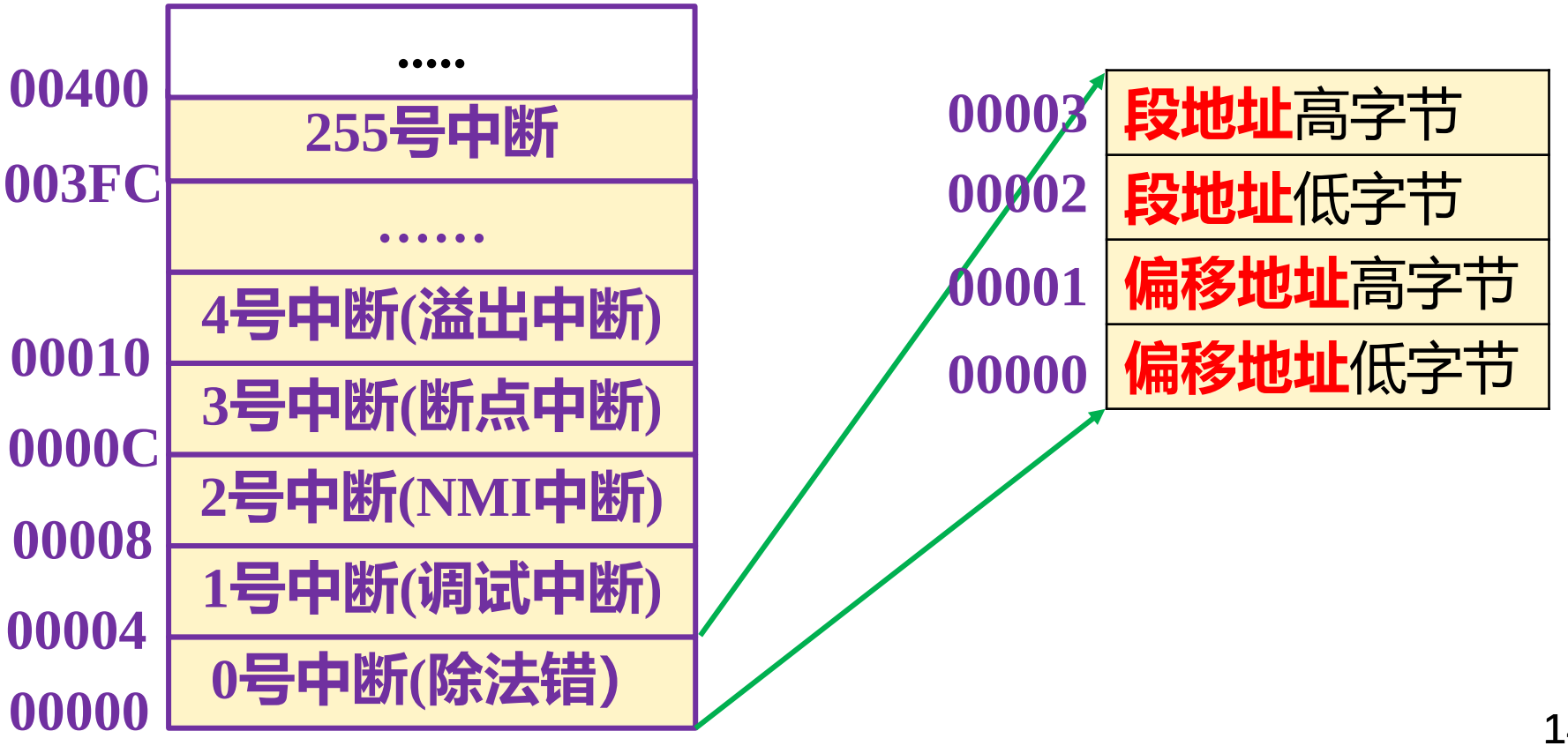
INTO;溢出中断指令：OF = 1，产生4号中断

STI;开中断指令：允许响应可屏蔽中断请求

CLI;关中断指令：禁止响应可屏蔽中断请求

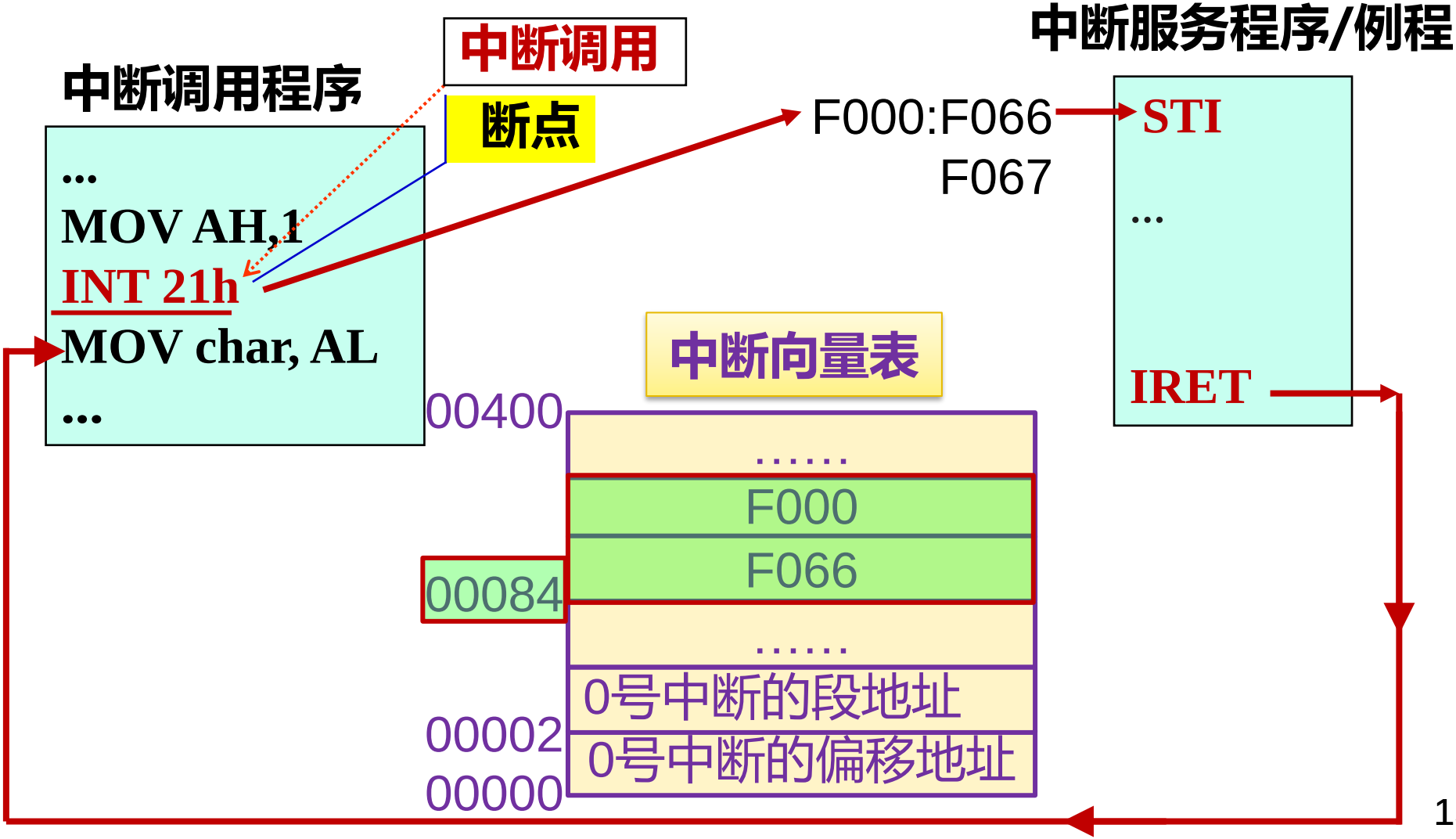
中断向量表

- 中断服务程序可存于主存任何位置
- 中断向量表：中断处理程序的入口地址表
- IA-32CPU能处理256个中断，中断号0~255



中断处理过程

- 中断处理过程



中断处理过程

- **INT n**执行的操作
  n为i8形式的参数，用伪指令描述如下：
  - **pushf**
  - **push cs**
  - ***push ip***
  - **cli**
  - **0:[n*4] → IP, 0:[n*4+2] → CS**

  ; **n** *4定位中断向量表项，设置**cs:ip指向中断服务程序**
- 中断服务程序
  开中断(**sti**)→保存现场→ …中断处理
  …. →恢复现场→ 中断返回(**I**RET)
- **IRET**执行的操作，伪指令描述如下：
  - ***pop ip***
  - **pop cs**
  - **popf**

1.4 DOS功能调用（INT 21H）

- 基本输入输出系统ROM-BIOS、操作系统DOS和Linux都采用中断调用方式提供系统功能
- DOS系统功能调用
  - 采用软件中断（指令**INT 21H**）进行功能调用
  - 使用寄存器传递参数
- DOS系统功能调用的4个步骤

（1）在**AH**寄存器中设置**系统功能调用号**
（2）在指定寄存器中设置入口参数
（3）用中断调用指令**INT 21H**执行功能调用
（4）根据出口参数分析功能调用执行情况

DOS基本功能调用（INT 21H）

功能号	功能	参数
AH = 01H	输入一个字符	AL = 输入字符的ASCII码
AH = 02H	输出一个字符	DL = 字符的ASCII码
AH = 09H	输出以$结尾的字符串	DX = 字符串地址
AH = 4CH	结束进程	AL = 返回代码

DOS功能调用程序

```
        .model small
        .686
        .stack
        .data                                   ;数据段
msg     byte 'Hello, Assembly!',13,10,'$'   ;显示的字符串
        .code                                   ;代码段
start:  mov ax,@data                        ; .startup
        mov ds,ax
        mov ah,9
        mov dx,offset msg                   ;指定字符串的偏移地址
        int 21h                             ;DOS功能调用显示字符串
        mov ax,4c00h                        ;结束程序运行
        int 21h
        end start                           ;汇编结束
```

例: 字符串输入

```
readstr   proc
          push ebx
          push ecx
          mov ebx,eax
          mov ecx,eax
rdm1:     mov ah,1
          int 21h
          cmp al,0dh
          jz rdm2
          mov [ebx],al
          inc ebx
          jmp rdm1
rdm2:     mov byte ptr [ebx],0
          cmp ebx,ecx
          jz rdm1
          sub ebx,ecx
          mov eax,ebx
          pop ecx
          pop ebx
          ret
readstr   endp
```

读入一个字符到AL中(ASCII码)

例：字符串显示

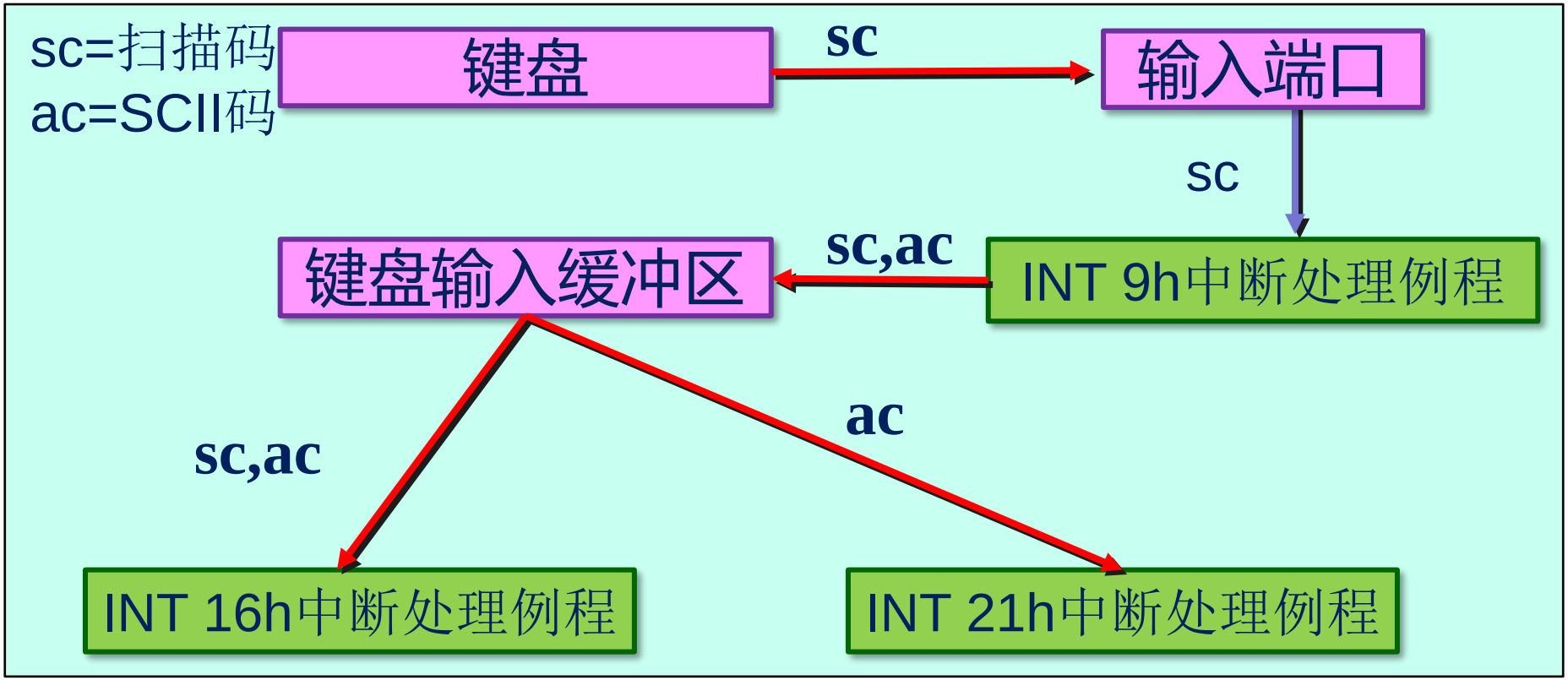
```
dispstr     proc
            push eax
            push ebx
            push edx
            mov ebx,eax
dispm1:     mov al,[ebx]
            test al,al
            jz dispm2
            mov ah,2
            mov dl,al
            int 21h
            inc ebx
            jmp dispm1
dispm2:     pop edx
            pop ebx
            pop eax
            ret
dispstr     endp
```

将DL中的字符(ASCII码)输出显示

2、BIOS程序设计

- 编程的层次：
  - 在BIOS（基本输入输出系统）层
  - 在操作系统（DOS、Windows）层之下
- 使用16位实地址模式
- 编程方法：
  - 使用BIOS功能调用
    - **INT 16H 键盘**
    - INT 10H 视频显示
    - INT 33H 鼠标
  - 内存中的BIOS数据区

2.1 INT 16h 键盘BIOS中断



2.1 INT 16h 键盘BIOS中断

- INT 16H 的10H号功能

描　　述：等待键盘按键

接收参数：AH=10H

返 回 值：AH=扫描码、AL=ASCII码

注-----意：若键盘缓冲区无按键，则等待按键。

2.1 INT 16h BIOS键盘中断

- INT 16H 的11H号功能

描　　述：检查键盘缓冲区

接收参数：AH=11H

返 回 值：

- **如果有按键在等待（按下中），则：**

  ZF=0、AH=扫描码、AL=ASCII码

- **否则：ZF=1**

注------意：不从键盘缓冲区中删除字符。

```
TITLE Testing ClearKeyboard      (ClearKbd.asm)
INCLUDE Irvine16.inc
ClearKeyboard PROTO, scanCode:BYTE
ESC_key = 1                      ; scan code
.code
main PROC
L1:     ; Display a dot, to show program's progress
        mov     ah,2
        mov     dl,'.'
        int     21h
        mov     eax,300                 ; delay for 300 ms
        call    Delay
        INVOKE ClearKeyboard,ESC_key ; check for Esc key
        jnz     L1      ; continue loop if ZF=0
quit:   call Clrscr
        exit
main ENDP
```

```
ClearKeyboard PROC, scanCode:BYTE
        push    ax
L1:
        mov     ah,11h ; check keyboard buffer
        int     16h    ; any key pressed?
        jz      noKey  ; no: exit now (ZF=0)
        mov     ah,10h ; yes: read and remove from buffer
        int     16h
        cmp     ah,scanCode  ; was it the exit key?
        je      quit         ; yes: exit now (ZF=1)
        jmp     L1           ; no: check buffer again
noKey:                               ; no key pressed
        or      al,1         ; clear zero flag
quit:
        pop     ax
        ret
ClearKeyboard ENDP

END main
```