保护模式和实模式之间的切换方法

宋广平

摘　要　该文简要介绍了80286、80386芯片的工作方式和存储器管理方法，阐述了保护模式与实模式之间的切换方法并给出一个实例，说明如何进行这种切换。

关键词　实模式　保护模式　V86模式

目前，大多数286、386PC 上都安装了扩展内存(Extended Memory)，但想在用户程序中直接访问它，不是件容易的事。由于 DOS 是在实模式下运行的，不能寻址到扩展内存单元，对扩展内存的操作必须切换到保护模式进行，然后再返回到实模式，恢复 DOS 操作。另外，80286、80386PC 的许多优越性能只有在保护模式下才能使用。因而，实模式和保护模式之间的切换是很重要的。

一、80286、80386的工作方式

80286有两种工作方式：实模式和保护模式，80386有3种工作方式：实模式、保护模式和虚拟86模式。在实模式下，80x86(x＝2,3)几乎是一个快速的8086(当然增加了一些新的指令)，寻址1MB 的内存空间，这时80x86的先进性能远没有发挥出来。保护模式是80x86达到其本身设计功能(多任务、多用户环境)的一种方式。首先，存储空间将大大增加，对80286而言，内存可达16MB；对80386而言，分段内存可达4GB。其次，支持多任务和多用户，特别是80386灵活多样的存储管理方式和处理速度，在这方面就更胜一筹。虚拟86模式又称 V86模式，它的目的是使80386可以重复而且迅速地在 V86模式和保护模式之间转换。把 CPU 标志寄存器中的 VM 位置“1”，即可进入 V86模式，执行一个8086程序；而把 VM 位复位，即可退出 V86模式而进入保护模式。

二、存储器管理

80x86的能力在保护模式下才能充分发挥出来，保护模式下的存储器管理方法与实模式下完全不同。实模式下的指令仍然可用在保护模式下，但对直接处理指针的程序，则不能在保护模式下运行。

1. 存储器地址和段描述符

保护模式下存储器的地址同样以逻辑地址表示，由段选择符(selector)和偏移量(offset)组成，其中段选择符为16位，偏移量80286为16位，80386为32位。这里，偏移量的意义和在实模式下的一样，根据不同的存储器访问状态，由 IP、SP 和指令操作数来给定。而段选择符与基址并无直接的关系，只是一个从逻辑上识别段的索引而已，它指向段描述符表中的一个段描述符。段描述符(descriptor)才真正定义了逻辑段所对应的物理存储器的地址。80x86段选择符和段描述符的结构见图1。

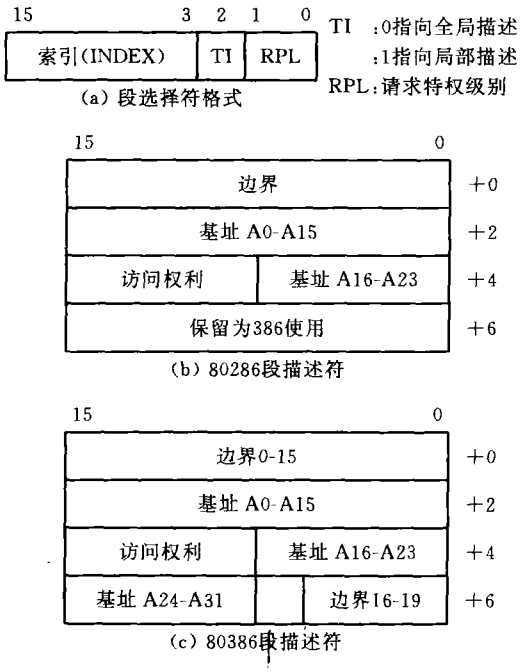

图1　段选择符和段描述符的数据结构

可见，80286段描述符定义的物理存储器的边界为16位，基址为24位；而80386分别为20位和32位。在实模

式下，边界固定为FFFFH，基址被设定为段选择符值左移4位(即16倍)；而在保护模式下，则可灵活地加以设置。另外，80286段描述符保留字段设置为0，则80286保护模式程序可在80386上运行。为了加快生成物理地址的速度，每一个段寄存器都有相应的段高速缓存器(对80286为48位，对80386为64位)，这对程序员来说是不可见的，只有CPU可以访问它。当一个选择符的值装入段寄存器时，选择符指向的段描述符自动被装入段高速缓存器。此后，访问这个段内的单元时，就使用高速缓存器中相应的字段值，执行从逻辑地址到物理地址的高速变换。通常，由于不需要对段寄存器进行频繁的修改，这种使用段高速缓存器的方法有着非常好的效果。

2. 描述符表

描述符表定义了80x86系统使用的所有段。在80x86系统中有3种类型的描述符表，它们分别是全局描述符表(GDT)、局部描述符表(LDT)和中断描述符表(IDT)。其中，GDT含有供系统所有任务使用的描述符；LDT含有与某一给定任务相关的描述符；IDT含有指向多达256个中断服务程序位置的描述符。每一个80x86系统都有一个GDT，含有操作系统使用的代码段和数据段，以及任务状态段和系统中各个LDT的描述符。每个描述符表最多可含8192(64K÷8)个8字节描述符，选择符的高13位作为描述符表的索引。本文为了简化问题，程序只设一个全局描述符表。而相应的描述符表的位置由全局描述符表寄存器(GDTR)、局部描述符表寄存器(LDTR)和中断描述符表寄存器(IDTR)决定。其中，80286的GDTR为40位寄存器，由16位边界字段和24位基址字段组成；80386的GDTR为48位寄存器，由32位基地址和16位边界组成。

3. 寻址模式

为了寻址一个存储单元，必须给出该单元所在段的段描述符和在段内的偏移量，然后，由段描述符中的基址字段的值加上偏移量，即得单元的物理地址(图2)。

段选择符中的TI位用于选择描述符表，当TI＝0时，访问GDT；当TI＝1时，访问该任务的LDT。

三、实模式到保护模式的切换

80x86CPU处于何种模式，由MSW(80286)或控制寄存器CR0(80386)的PE位决定，PE＝0，处于实模式下；PE＝1，处于保护模式下。对80386而言，在保护模式下，将标志寄存器(EFLAGS)的VM位置1，则进入虚拟86模式；VM位清零，则退出虚拟86模式而回到保护模式。80x86复位后，PE＝0，因此以实模式动作。要想从实模式转换到保护模式，就需将MSW或CR0的PE位置1。这可用LMSW指令将数据装入MSW或CR0。不过仅有这些还不能正确执行保护模式，必须进行如下的初始设定。

1. 建立全局描述符表

为使80x86在保护模式下工作，必须在存储器中定义全局描述符表，并把该GDT的基址和边界装入GDTR。在描述符表中建立保护模式下的代码段描述符及必要的堆栈段和数据段描述符。其中，代码段描述符的访问权字节可设置为9BH，数据段和堆栈段描述符的访问权字节可设置为93H，基址可根据实际的代码段、数据段和堆栈段的物理地址加以设置，为简单起见，边界可设置为FFFFH。这样，进入保护模式后，CPU能在定义的代码空间和数据空间继续执行指令，而不至于产生异常中断或死机。不过应注意，GDT中的第一个描述符在软件上不能使用，对应的选择符0称为空选择符(NULL SELECTOR)，当向段寄存器加载该选择符时，将中止指令的执行或产生异常保护中断(中断13)。

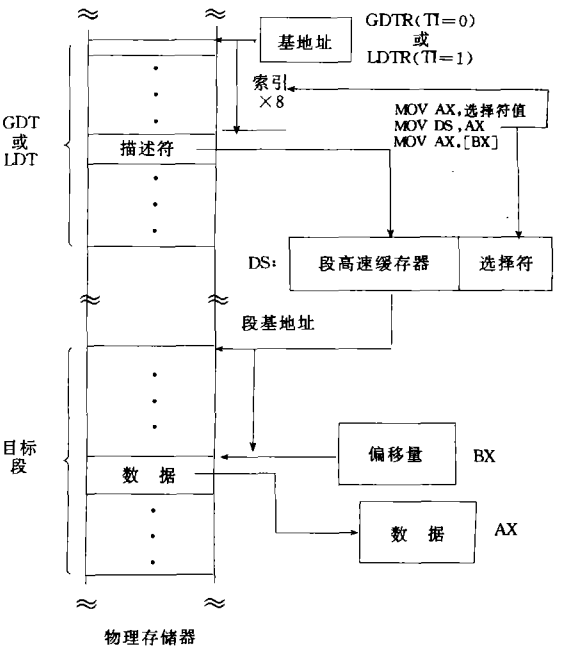

图2 保护模式下操作数的寻址过程

2. 释放地址线A20

为保证和8086/88软件的兼容性，目前一般的286和386PC上，利用外部附加逻辑对A20地址线可用程序进行控制，系统在初始化完成之后，将A20置成低电平。因而，在进入保护模式之前必须解除这种控制。事实上，A20是受键盘控制器电路的Intel 8042(8742)控制的，可向输入缓冲器(I/O口地址为64H)写入命令字D1H，随后向I/O口地址60H写入DFH释放A20或写入DDH禁止A20。

3. 其它方面

如果在保护模式下需处理中断，则必须建立IDT及相应的描述符。同时必须先保存中断控制器的状态，以便返回到实模式后进行恢复。由于BIOS中断向量不能在保护模式下工作，因而，进入保护模式的程序必须

建立它本身的IDT，以处理保护模式下发生的中断，而且不能覆盖实模式下的中断向量表。

以上几方面都准备好后，用LMSW指令将PE位置1，转入保护模式，紧接着用一条段间跳转指令清除指令队列，并转至相应的代码段，在保护模式下继续执行指令。注意，该跳转指令在保护模式下执行，但已在实模式下被预取入指令队列，因而不用担心由于存储器管理方法的改变而取不到该指令的情况。如果程序进入保护模式后，没有必要再回到实模式，那么可以利用BIOS INT 15H中断的AH＝89H功能，很容易地将CPU转换到保护模式。具体的调用格式如下：

输入：AH＝89H

BH＝前8个硬件中断向量在IDT中的起始位置

BL＝后8个硬件中断向量在IDT中的起始位置

ES:SI＝指向GDT的指针

输出：AH＝00　成功(CF＝0)地进入保护模式

AH＝FFH　失败

CF＝0　没错

CF＝1　有错

用户提供的GDT的格式见附表。

附　表

表中单元	离ES:SI的偏移量	大小(B)	意　义
0	00H	8	空描述符，初始化成0
1	08H	8	描述该GDT的描述符
2	10H	8	用户定义的IDT的描述符
3	18H	8	用户的数据段描述符
4	20H	8	用户的附加段描述符
5	28H	8	用户的堆栈段描述符
6	30H	8	该调用将返回到的代码段的描述符
7	38H	8	该调用使用的BIOS代码段描述符

该GDT中除第七单元外，其余均由用户填写。第七单元的描述符将由该功能设置，其中，访问权为9BH，基址为F0000H，边界为FFFFH，CS、DS、ES和SS分别被该功能设置成0030H、0018H、0020H和0028H，因而，CPU将在保护模式下，在用户定义的空间中继续运行。

四、保护模式到实模式的切换

由保护模式切换至实模式，80286与80386应区别对待。下面分别加以说明

80286　上电复位，回到实模式，将使内存中的数据和程序丢失(包括操作系统)，因而不能采用这个方法。分析系统板电路，可以发现，也可通过软件使80286的RESET引脚变成高电平。实际上，键盘控制器电路的Intel 8042的P20(P2口的bit0)可控制RESET信号，只要P20输出低电平，即可使RESET变为高电平，使80286进入复位状态。一旦复位，则将转到POST(Power On Self Tests)程序，POST将读取CMOS RAM的0FH处的停机码字节，以决定是否需要重新初始化系统。当停机码字节为05H时，POST把控制转移到40:67H双字处。通常，这里放置转向的入口处的远地址。

根据上述原理，我们可以先在40:67H处放置返回到实模式程序的起始地址，然后在保护模式下，用指令复位80286，这样，便回到了实模式，而且避免了重新初始化系统。接着禁止地址线A20，如修改了中断控制器状态，则须加以恢复。

80386　对于80386，除了可以利用80286的方法外，还可以用指令LMSW AX(AX＝0)复位CR0的PE位(或指令MOV CR0,EAX)，而回到实模式。紧接着用一条段间跳转指令清除指令队列，并转到相应的代码段，在实模式下继续执行指令，随后禁止地址线A20，如还修改了中断控制器的状态，则须加以恢复。

五、举　例

下面给出一个在实模式和保护模式之间切换的演示程序。所有操作都在0级特权上进行。当切换至保护模式时，屏幕上将缓慢地显示："Now Switched To Protected Mode From Real Mode! Waiting...."，接着又回到实模式，并显示："Now Switched To Real Mode From Protected Mode!"，随后返回到DOS。由于在保护模式下，不能利用DOS和BIOS的资源，故程序直接对显示缓冲区操作。用TASM example.asm汇编生成example.obj目标码，然后用TLINK/t example生成example.com可执行文件。程序中使用了条件汇编伪指令，以便生成不同机器和显示器的代码。例如：

TASM/dMDA example.asm

产生单色显示器代码，其它选择如/d－386、/dINT15读者可自己考虑。

本文只是简单地介绍了80x86 CPU的工作模式和存储器管理方面的一些基本知识，事实上，80x86还有很多优越的性能，例如：存储保护，特权管理，任务管理，异常保护和80386的存储器分页功能等。有兴趣的读者可参考有关资料。

程序示例

```
.286p
IFNDEF MDA
  crt_buffer_base_lo_word EQU 8000h
  crt_buffer_base_hi_byte EQU   0bh ;彩色显示器缓冲区
                                    的物理地址
ELSE
  crt_buffer_base_lo_word EQU 0000h ;单色显示器缓冲区
                                    的物理地址
    crt_buffer_base_hi_byte EQU  0bh
ENDIF
```

```
data_seg_access EQU 10010011b  ;数据段访问权字节
code_seg_access EQU 10011011b  ;代码段的访问权字节

data_input_buf_port    EQU 60h
_8042_input_buf_port   EQU 64h
_8042_status_port      EQU 64h  ;Intel 8042的状态口
cmos_port              EQU 070h ;CMOS RAM 的口地址
descriptor STRUC               ;段描述符结构
  seg_limit     dw 0  ;边界
  base_lo_word  dw 0  ;地址低16位
  base_hi_byte  db 0  ;地址高8位
  access_right  db 0  ;访问权利
  reserved      dw 0  ;保留字节
descriptor  ENDS

  jumpfar  MACRO jumpfar1,jumpfar2 ;跳转指令的宏表示
           DB 0eah
           DW (OFFSET jumpfar1)
           DW jumpfar2
           ENDM

  lodcr0   MACRO                   ;MOV CR0,EAX 的
                                    宏表示
           DB 0fh
           DB 22h
           DB 0c0h
           ENDM

bios_data_seg SEGMENT AT 040h
              ORG 0067h
to_real_off   DW?
to_real_seg   DW?
bios_data_seg ENDS

cseg SEGMENT public para 'code'
     ASSUME cs:cseg,ds:cseg
     ORG 100h
start: jmp main
;* * * * * * * * * * * * * * * * * * * * * * * * * * *
;*   以下是全局描述符表－GDT,为了也能利用 INT 15h   *
;*   切换到保护模式,采用该功能所要求的 GDT 格式      *
;* * * * * * * * * * * * * * * * * * * * * * * * * * *
gdt LABEL qword
dummy_des  descriptor <>
gdt_des    descriptor <gdt_leng,,,data_seg_access,>
idt_des    descriptor <idt_leng,,,,data_seg_access,>
data_des   descriptor <0ffffh,,,data_seg_access,>
esdata_des descriptor <0ffffh,,,data_seg_access,>
ss_des     descriptor <0ffffh,,,data_seg_access,>
cs_des     descriptor <0ffffh,,,code_seg_access,>
bios_des   descriptor <>
gdt_leng EQU $ -gdt
;* * * * * * * * * * * * * * * * * * * * * * * * * * *
;*   以下是中断描述符表－IDT,由于该程序不           *
;*   处理中断,因而,该 IDT 为空                      *
;* * * * * * * * * * * * * * * * * * * * * * * * * * *
idt LABEL qword

idt_leng EQU $ -idt

IFNDEF_386
  int_imr1_port EQU 21h
  int_imr2_port EQU 0a1h
  imr1_byte db 0
  imr2_byte db 0
ENDIF

main PROC
IFNDEF_386
  ASSUME ds:bios_data_seg   ;ds 指向 bios_bata_seg
    mov ax,bios_data_seg
    mov ds,ax
    mov to_real_off,OFFSET user_code_r
                            ;在40:67h 处写入实模式的返回地址
    mov ax,cs
    mov to_real_seg,ax
    mov al,0fh              ;在 CMOS RAM 0f 处设置停机码05h
    out cmos_port,al
    mov al,05h
    out cmos_port+1,al
  ASSUME ds:cseg
    cli
    in al,int_imr1_port ;保存中断屏蔽状态
    mov imr1_byte,al
    in al,int_imr2_port
    mov imr2_byte,al
    mov dx,cs               ;ds 指向 cseg
    mov ds,dx
ELSE                        ;如用 TASM/d_386汇编该程序,
  ASSUME ds:cseg  ;生成的代码用软件复位 PE 位,
    mov ax,cs               ;只适用于386机。否则,硬件复位 PE 位
    mov ds,ax               ;ds 指向代码段
    mov real_seg,ax ;实模式返回的段址
ENDIF

;* * * * * * * * * *
;*   初始化 GDT     *
;* * * * * * * * * *
  mov dx,ds ;初始化 gdt_des 描述符
  mov cx,OFFSET gdt
  call addr_24bit_form ;形成24位物理地址
  mov gdt_des.base_lo_word,cx ;填入 gdt_描述符的相应
                              字段
  mov gdt_des.base_hi_byte,dl
  mov dx,ds             ;初始化中断描述符表的段描述符
  mov cx,OFFSET idt
  call addr_24bit_form
  mov idt_des.base_to_word,cx
  mov idt_des.base_hi_byte,dl
  mov data_des.base_lo_word,crt_buffer_base_lo_word
  mov esdata_des.base_lo_word,crt_buffer_base_lo_word
  mov esdata_des.base_hi_byte,crt_buffer_base_hi_byte
  mov dx,ss             ;初始化堆栈段描述符
  xor cx,cx
  call addr_24bit_form
  mov ss_des.base_lo_word,cx
  mov ss_des.base_hi_byte,dl
  mov dx,cs             ;初始化代码段描述符
  xor cx,cx
```

```
    call addr_24bit_form
    mov cs_des.base_lo_word,cx
    mov cx_des.base_hi_byte,dl
    mov dx,cs          ;初始化 IDT 的段描述符
    mov cx,OFFSET idt
    call addr_24bit_form
    mov idt_des.base_lo_word,cx
    mov idt_des.base_hi_byte,dl

;* * * * * * * * * * * * * * * * * * * * * * * * * * *
;*   利用 INT 15h,ah=89h 切换至保护模式              *
;* * * * * * * * * * * * * * * * * * * * * * * * * * *
   push cs
   pop es
   mov si,OFFSET gdt   ;ES:SI 指向 GDT
IFDEF INT15   ;如用 TASM/dINT15 example 汇编此程序
   mov bh,08h   ;则生成的代码使用 INT 15H 的89H 功能
   mov bl,70h   ;切换至保护模
                ;式
   mov ah,89h   ;否则,由 to_protected_mode 切换至保护模
                式
   int 15h
   jmp user_code_p
ELSE
   call to_protected_mode
ENDIF
   main ENDP

;* * * * * * * * * * * * * * * * * * * * * * * * * * *
;*   该过程将 CPU 由实模式切换到保护模式             *
;* * * * * * * * * * * * * * * * * * * * * * * * * * *
addr20_enable_data EQU 0dfh   ;释放地址线 A20字节0DFH
to_protected_mode PROC
            mov al,addr20_enable_data ;释放地址线 A20
            call addr20_dis_or_enable
            or al,al
            jz addr20_succ
            mov ah,0ffh
            stc
   exit_fail:  mov dx,OFFSET failure     ;失败
            mov ah,9
            int 21h
            int 20h
failure db "Freeing Address Line A20 Fails,Can't Switch CPU
To Protected Mode! $"
addr20_succ: lidt qword ptr es:[si+10h] ;由于不处理中断,
                                        可省略该指令
            lgdt qword ptr es:[si+08h]
            cli
            smsw ax
            or ax,0001h
            lmsw ax ;置位 MSW 的 PE 位,将 CPU 切换至
                    保护模式
            jumpfar user_code_p,030h ;清除指令队列,并
                                     转至用户代码
            ret
to_protected_mode ENDP

;* * * * * * * * * * * * * * * * * * * * * * * * * * *
;*  一退出 INT 15H 后,在保护模式下,将进入下面用    *
;*  户定义的过程并在屏幕上显示:"Now Switched To     *
;*  Protected Mode From Real Mode! Waiting..."       *
;* * * * * * * * * * * * * * * * * * * * * * * * * * *
user_code_p PROC          ;保护模式下的用户过程
            cld
            mov ax,18h
            mov ds,ax     ;ds 指向 data_des
            mov ax,20h
            mov es,ax     ;es 指向 esdata_des
            xor si,si
            xor di,di
            mov cx,80 * 25
            mov ax,00h    ;清屏
            rep stosw
            mov cx,mesg-leng
            xor di,di
            add di,80 * 4
            mov bx,OFFSET protected_mesg
            mov ah,07h
   aga:     mov al,cs:[bx]
            stosw
            in_bx
            call delay    ;延迟
            loop aga
            xor si,si
            xor di,di
            mov cx,80 * 25
   agal:    Lodsw         ;正常显示
            mov ah,07h
            stosw
            loop agal
            call delay
            call to_real_mode   ;切换至实模式
            ret
protected_mesg DB "Now Switched To Protected Mode From
Real Mode! Waiting..."
mesg_leng EQU $ -protected_mesg
user_code_p ENDP

;* * * * * * * * * * * * * * * * * * * * * * * * * * *
;*   该过程将 CPU 切换至实模式                       *
;* * * * * * * * * * * * * * * * * * * * * * * * * * *
to_real_mode PROC
IFNDEF_386
            mov al,0feh   ;控制 P20输出宽约6μs 的低电平
                          脉冲
            out_8042_input_buf_port,al ;复位 CPU
            hlt           ;停机
            ELSE
            smsw ax
            and ax,0fffeh
            lodcr0        ;MOV CR0,EAX 置 CR0的 PE
                          位为0,切换至实模式
            db 0eah       ;JMP 指令的宏表示
            dw OFFSET user_code_r
   real_seg dw?
            ret           ;清指令队列,并转至用户代码
ENDIF
```

```
to_real_mode    ENDP

;* * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*   退出保护模式后,进入实模式,将在屏幕上显示:          *
;*   "Now Switched To Real Mode From                    *
;*    Protected Mode!"                                  *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * *
addr20_disable_data EQU 0DDH
user_code_r  PROC        ;实模式下的用户过程
             mov ax,cs
             mov ss,ax
             mov ds,ax
             mov al,addr20_disable_data
             call addr20_dis_or_enable
             or al,al
             jz addr20_dis_succ
             mov dx,OFFSET addr20_fail_mesg
             mov ah,09
             int 21h
             int 20h         ;返回到 DOS
addr20_dis_succ:
IFNDEF_386
             mov al,imr1_byte   ;恢复中断屏蔽状态
             out int_imr1_port,al
             mov al,imr2_byte
             out int_imr2_port,al
ENDIF
             mov ax,cs
             mov ds,ax
             sti
             mov dx,OFFSET to_real_success
             mov ah,9
             int 21h
             int 20h
             ret
addr20_fail_mesg DB "Inhibiting Address Line A20 Failed!
$"
to_real_success  DB "Now Switched To Real Mode From
Protected Mode! $"
user_code_r  ENDP

;* * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*   测试8042输入缓冲器是否空,如不空,则等待           *
;*   输出:AL=0,空                                      *
;*        :AL≠0,不空                                   *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * *
_8042_input_buf_empty PROC
             xor cx,cx
 again0:     in al,_8042_status_port
             and al,02h
             jz empty
             loop again0
             or al,al
             ret
 empty:      xor al,al
             ret
_8042_input_buf_empty ENDP

;* * * * * * * * * * * * * * * * * * * * * * * * * * * *
;* 释放或禁止地址线 A20                                 *
;* 输出:AL=0DFH,则释放 A20;AL=0DDH,则禁止 A20          *
;* 输出:AL=0,成功                                      *
;*      :AL≠0,失败                                     *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * *
addr_20_dis_or_enable PROC
        cli
        push cx
        mov ah,al
        call _8042_input_buf_empty
        or al,al
        jnz fail0
        mov al,input_cmd
        out _8042_input_buf_port,al
        call _8042_input_buf_empty
        jnz fail0
        mov al,ah
        out data_input_buf_port,al
        call_8042_input_buf_empty
 fail0: pop cx
        ret
addr20_dis_or_enable ENDP
;* * * * * * * * * * * * * * * * * * * * * * * * * * * *
;*   将段址:偏移量转变为24位地址                       *
;*   输入:DX=段址                                      *
;*        :CX=偏移量                                   *
;*   输出:DL=地址高8位                                 *
;*        :CX=地址低16位                               *
;* * * * * * * * * * * * * * * * * * * * * * * * * * * *
addr_24bit_form PROC
       push ax
       mov ax,16
       mul dx
       add cx,ax
       adc dx,0
       pop ax
       ret
addr_24bit_form ENDP
;* * * * * * *
;* 延迟过程  *
;* * * * * * *
delay PROC
      push ax
      push bx
      mov ax,0ffh
lp2:  mov bx,0ffffh
lp1:  nop
      nop
      dec bx
      jnz lp1
      nop
      nop
      dec ax
      jnz lp2
      pop bx
      pop ax
      ret
delay ENDP
cseg  ENDS
      END start
```