用汇编编写DOS下的内存驻留程序(3) PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/259/2021_2022__E7_94_A8_E6_B1_87_E7_BC_96_E7_c98_259406.htm 三 中断矢量 3.1 IBM PC提供的中断 IBM PC有两种基本形态的中断.如果是由外围设备所产生的中断就叫做硬件中断(Hardware interrupt),譬如:键盘,磁盘机和时钟等外围设备都可以产生硬件中断.外围设备所产生的中断信号都连接到中断控制器,中断控制器可以根据它们之间的重要性来安排优先顺序,以便使CPU有效地处理这些硬件信号.另一种中断是软件中断(Software interrupt),软件中断也叫做陷井(Trap),它是由执行中的软件所产生.虽然软件包中断的处理方式和硬件中断完全相同,但是通常软件中断是希望执行操作系统所提供的服务. 表3.1是IBM PC所提供的中断,这些中断是根据中断号码和中断矢量(Interrupt vector)排列.IBM PC的用户或是编写应用程序的程序人员很少会直接接触到硬件中断,除非是使用某些特殊的硬件,或是需要较严格的要求时,最常被修改的硬件中断是敲键盘所产生的中断(9H),尤其是文本编辑的程序.大体而言,只有硬件设计者基是系统程序人员才会注意到所有在硬件中断.编写内存驻留程序的设计人员则只使用到部分硬件中断而已,尤其是:键盘中断和计时器(Timer)的中断. 反之,软件中断对于任何编写汇编程序的人,甚至对编写高级语言程序的人都相当的重要.软件中断是应用程序进入到IBM PC操作系统的接口,经由这些接口应用程序才可以执行所要求的系统服务. 其中软件中断中最重要,同时也是最常被汇编语言程序设计师所用到是DOS INT 21H.这个中断是执行DOS系统调用的软件中断,它可以让应用程序执行任

何DOS的操作. 接下来最有用的软件中断是ROM-BIOS(基本输入输出系统)所提供的中断.这些软件中断是IBM PC所提供的的低层次服务,譬如:键盘输入,显示器输出和磁盘机的输入与输出等.3.2 键盘输入的方法 以下就以IBM PC从键盘读取字符为例子,来说明中断的工作方式.IBM PC从键盘读取字符时,使用了两种不同形式中断,亦即:硬件中断和软件中断.当使用者从键盘敲下一个键时,键盘的线路就会送出一个信号.这个信号会造成硬件中断发生,从而触发低层次的键盘中断处理程序开始执行.这个中断处理程序马上从键盘的硬件读取使用者所敲入的字符,然后把它放到一个队列中,如果这个队列填满时,键盘中断处理程序会使IBM PC发出一声响.键盘中断处理程序做完这些事情之后,它就把控制权交还给原先被中断的程序.如果有一个程序希望从键盘读取一个字符时,它就发出适当的软件中断信号,这时候就由相对应的中断处理程序去检查键盘队列,并且传回队列中的第一个字符. 上面所介绍的键盘输入工作方式,在中断驱动系统中很普遍地采用.这和做法可以把实际上需要输入的应用程序和实际上执行输入的处理部分分开来.这种做法也可以用在其它不同形式的输入和输出外围设备. 3.3 改变输入矢量 中断矢量储存在IBM PC最前面的400H个字节中.每一个矢量的长度是四个字节组成,这四个字节内所存放的是中断处理程序执行的地址值.其中前两个字节包含地址值的位移(Offset)部分,后面的两个字节则包含了段(Segment)部分. 中断矢量有两种修改方法.可以直接地设置中断矢量的地址值,或是使用DOS所提供的系统调用设置中断矢量的地址值. 3.3.1 直接设置中断矢量 因为中断矢量只是存放地址值的存储位置,因此我们可以直接地把地址存放到存储位置中.以下是一个小例

子: mov ax,0 mov es,ax mov word ptr es:24,offset Keyboard mov word ptr es:26,seg Keyboard 在许多情况下,上面的程序都可以正确地执行.但是如果上面的程序正在执行时突然敲下一个键的话,就可能会问题.而最糟的情 况是发生:第三个MOV已经执行完毕,而第四个MOV尚未执行时.如果在此时敲下任何键的话,键盘中断矢量都没有任何意义,而造成整个系 统死机.因此我们可以在设置中断矢量时,让中断无效,譬如: mov ax,0 mov es,ax cli mov word ptr es:24,offset Keyboard mov word ptr es:26,seg Keyboard 上面的做法在大部分的情况下都可以正确地执行.但是CLI这个指令无法停止NMI中断(不可屏蔽中断),因此如果发生NMI中断时就 没用办法. 下面的这一种做法虽然比较复杂,但是对于所有的中断都有效,这包括了NMI中断在内: mov word ptr kbd-ptr[0],offset Keyboard mov word ptr kbd-ptr[2],seg Keyboard mov di,0 .Use Di to Set ES to zero mov es,di .Set ES to destination segment mov di,24 .Set DI to destination offset mov si,offset kbdptr .set SI to source offset mov cx,2 .Set word count to 2 cld .Set direction to forward cli .Disable interrupts rep movsw .Copy the new vector sti .Enable interrupts kbdptr dd ? 上面的程序中,kbdptr是两个字节(WORD)的指针(Pointer),其中包含了键盘中断处理程序的起始志趣值.REP这个指令将根据寄存 器CX所设置的次数来重复执行MOVSW,而整个指令就如同单一的指令一样.NMI中断不能够发生在一个完整的指令中.因为地址值搬移的操 作都能包含在一个单一指令中,因此可以免除任何中断的干扰. 3.3.2 使用DOS来设置中断矢量 因为要想安全地设置中断矢量需要一些技巧,因此DOS提供了一项特殊的服务,以帮助程序人员安全地设置中断矢量,如果只使用 DOS所提供的

这项服务来设定中断矢量的话,那么就不必担心会发生前面所叙述的差错.DOS同时也提供了:读取中断矢量的服务.因为读取中断矢量的内容不会修改系统的状态.因此若直接写程序读取,也很安全.但是如果你要自己直接读取中断矢量的内容时,就必须计算 出中断矢量的位置.而DOS已经提供了这项服务.使用DOS所提供的系统调用,来读取中断矢量的内容时,必须利用INT 21H中的函数35H(读取中断矢量),这个函数热气矢量号码来 计算中断矢量的地址,然后返回其中的内容.以下就是一个例子: Old_Keyboard_IO dd ? mov al,16h mov ah,35h int 21h mov word ptr Old_Keyboard_IO,bx .Offset of interrupt handler mov word ptr Old_Keyboard_IO,es .Segment of interrupt handler 用DOS来设置中断矢量例子: New_Keyboard_IO dd ? mov word ptr New_Keyboard_IO,bx .Offset of interrupt handler mov word ptr New_Keyboard_IO,es .Segment of interrupt handler mov al,16h mov ah,25h int 21h 