用汇编编写DOS下的内存驻留程序(1) 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_259410.htm 绪言 0.1 内存驻留与中断 内存驻留程序英文叫Terminate and Stay Resident Program,缩写为TSR.这些程序加载进内存,执行完后,就驻留在内存里,当满足条件时,调到前台来执行。 内存驻留程序的常用形式有: >诸如Borland 的SideKick弹出式实用程序 >日历系统 >网络服务器 >通讯程序 >本地的DOS扩展(如CCDOS,UCDOS等中文系统都属于这个范畴) >一些可恶的人利用TSR技术制作很多可恶的病毒程序,几乎所有的病毒程序都是TSR程序. 就象多任务系统调度一个进程有一个调度程序一样,在PC中从前台程序进入到一个TSR,也要有一个调度者,只是PC操作系统的调度不称为调度程序,而只称为触发机制.触发机制调度TSR执行在PC机上党称为激活一个TSR.触发机制主要有以下几种: >硬件中断:党用的是键盘中断INT 9H,时钟中断INT 8H,通讯中断INT 14H,磁盘中断INT 13H等等. >软件中断:党用的是键盘中断INT 16H,时钟中断INT 1CH,DOS中断INT 21H,等等. >以上各种的结合. 从以上的触发机制可以看出,TSR和PC机的中断系统有着密切的关系.每种激活方式实际上都是与中断有关的.常用特殊的击键序列的识别码是通过截获INT 9H和INT 16H来实现.实际上不管TSR程序的哪一个环节,都与中断有着密切的关系.因此在具体进行TSR和程序设计之前,先介绍PC中断系统.在此只作简单说明. 在PC机内存的最低端(0000H开始)的1K字节中,存放着256个指针即常说的中为向量或中断矢量(Interrupt vertor)，每个中断向量都指向一个子程序，该程

序称为中断处理程序（Interrup handler). 一个中断向量由四个字节组成，有一个字是中断处理程序的偏移量值，后一个字是中断处理程序的段值. 256中断向量一起称为中断向量表. 手式计算中断向量的首址,可通过以下的公式来求得: X号中断向量的首址=0000H:X*4 当产生一个中断时,处理器都按顺序执行以下步骤: >在堆栈上压入处理器的标志(相当于指令PUSHF). >在堆栈上压入当前CS和IP值(相当于指令PUSH CS和PUSH IP). >关闭中断(CLI) >从中断向量加载的CS和IP,执行中断处理程序. 当执行完中断处理程序后,一般用IRET返回,它的作用是: >从堆栈上取出保存的IP和CS(相当于指令POP CS和PUSH CS). >同时恢复中断前的处理器标志(相当于指令POPF). 中断有多种分类,由触发的原因和实现的性质来分,可分为硬件中断和软件中断,从操作系统分层实现来说,可以分成BIOS中断,BOS中断和用户中断. 一方面,BIOS和DOS通过中断系统向用户提供一个操作系统功能界面.也就是说用户(一般来说是前台程序)的功能主要是通过调用DOS和BIOS的中断服务来实现的,具体来说就是通过INT指令来实现的.另一方面,BIOS和DOS由中断系统所构成,BIOS对硬件成为高层的功能,并通过中断的形式向用户提供. 如果在当前程序执行的同时,能将一块代码放在内存,把中断向量指向代码中的子程序,那么在当前程序执行中产生中断时,就有可能执行不属于当前程序和操作系统的代码,产生的中断可能是当前程序产生的软件中断,也可能是由硬件产生的硬件中断.这就是单任务的PC操作系统可能执行多于一个进程的简单说明. 在PC中断系统中有几个中断具有周期性,即INT 8H,INT 1CH和INT 28H.它们或者周期性被执行用于时间计时,或者周期性产生用于等待.它

们是在实现TSR时进行轮询触发的基础.键盘中断(INT 9H和INT 16H)当用户击键时发生,利用它们是进行热键处理的基础.串行口通讯也是触发的一个重要机制.此外众多的软件中断也是触发的媒介. 0.2 DOS的可重入性分析 一个多任务操作系统之所以能使多个进行并存,是因为操作系统的大部分代码是可以了重的,对于临界资源有相应的PV操作,使得当调度一个新的进程时,能完整地保存前一个里程的现场,当再一次调度被挂起的进程时能象没有被中断一样继续执行. 对于PC机来说,代码的重入性比较弱,对临界资源没有PC操作.当我们用中断程序启动用户的TSR时,如果只保存标志和寄存器,以及当前进程一些信息,那么只保存了当前程序的一部分现场,DOS的临界资源不会自动保存.在进行TSR设计时,一定要了解PC操作系统的重入性和临界资源. 重入性总是体现在代码上,所谓可重入代码的指这样的代码,即该代码被执行时还没有从中退出,由于某种原因又一次或者多次进入相同的代码,该代码每次的执行结果都是正确的,就说该代码是可重入的.相反,如果结果不正确,那么就就该代码是不可重入的.下面是一个可重入的子程序的例子: Add proc near cmp DS:word ptr [si],0 je DonotAddTheValue add ax,DS:word ptr [si] DonotAddTheValue: ret Add endp 上面的例子不管在其中任何一处再一次执行该子程序,执行结果不变.为了说明,只举多种可能性中的一种. mov ds,0100h .ds=0100h mov si,0010h .si=0010h mov ax,0001h .ax,=0001h call Add cmp 0100h:word ptr [0010h],0 .Call Add subroutine push ds .Interrupted push si push ax mov ds,0200h .ds=0200h mov si,0200h .si=0020h mov ax,0003h .ax=0003h call Add cmp 0200h:word ptr [0020h],0 .0200:0020h=0004h jne add

ax,0200h:word ptr [0020h] .ax=0007h ret .Return pop ax .ax=0001h pop si .si=0010h pop ds .ds=0100h iret .Return to Add subroutine jne add ax,0100h:word ptr [0100h] .ax= 0001h .0100h:0010h= 0002h .---------- .ax = 0003h ret mov bx,ax