计算机软件开发系列丛书

汇编语言简易模块实例图解设计

蔡椿华 编著

学苑出版社

希望

汇编语言简易模块实例图解设计

蔡椿华　编著
王真华　改编
燕卫华　审校

学苑出版社
1994

(京)新登字 151 号

内 容 提 要

本书全面系统地介绍了汇编语言的强大功能，并配有大量的实例，力图使大部分甚至没有基础的人都能通过“实例”去了解并使用这种语言。另外，在每章后还配有练习题，以便读者在阅读的基础上加深理解，达到融会贯通的效果。本书内容详尽，结构严谨，可供大专院校计算机专业师生参考，也适合于广大计算机用户阅读。

欲购本书的用户，请直接与北京 8721 信箱联系，邮政编码 100080，电话 2562329。

版 权 声 明

计算机软件开发系列丛书

汇编语言简易模块实例图解设计

编　　著：蔡椿华
改　　编：王真华
审　　校：燕卫华
责任编辑：甄国宪
出版发行：学苑出版社　　邮政编码：100036
社　　址：北京市海淀区万寿路西街 11 号
印　　刷：双青印刷厂
开　　本：787×1092　1/16
印　　张：41.5　　字　　数：975 千字
印　　数：1～5000 册
版　　次：1994 年 3 月北京第 1 版第 1 次
ISBN7—5077—0779—2/TP・11
本册定价：59.00 元

学苑版图书印、装错误可随时退换

目　录

第一章　常驻程序结构的建立

1.0.1　有趣的常驻程序例子

在学习常驻程序设计原理及主题前，我们先来做一个例题，以增强对 TSR 的概念；谈到 TSR 必须谈到对内存的管理及时间分割的基本多路概念。

这些基本概念，作者在后面会慢慢地为您做分析。

例：设计一个程序，使程序常驻在系统，它可以检测到任何时候用户使用了软盘或硬盘，而且屏幕出现字符串标记(mark)并且喇叭发出声音。

提示

1. 软盘、硬盘使用必须用到 INT 13h BIOS 系统调用，也就是必须更改中断向量表及截取 INT 13h 的中断向量(详细的 int 13h，参阅附录 D)。
2. 改变中断向量后，再使用 DOS Ah＝31h，int 21h 功能，把程序常驻到内存中。
3. 详细原理，请在阅读完本章后，再探讨有疑问之处。
4. 程序请参考 TSRDISK2.ASM，如下：
5. 现在请拿起书上附的磁盘，执行＞tsrdisk2 ENTER，再执行＞chkdsk ENTER。

```
;FileName=tsrDISK2.asm.....Function=When Use Disk ..Mark & Sound Tell User
;  >masm tsrdisk2;>masm tsrsub5;>masm tsrsub6;>link tsrdisk2+tsrsub5+tsrsub6;
INCLUDE TSAY.MAC
.MODEL  SMALL
.STACK
.DATA
attribute       db      70h
show  db    ': Notice....Somebody Use Your Disk.....','$'
;-------------------------------------------------------------------------
.CODE
;.....................................................
EXTRN SHOWMARK:NEAR   ;Get From Tsrsub5.asm
EXTRN SOUND_ON:NEAR   ; Get From TsrSUB6.asm
PUBLIC  attribute     ;Pass To Tsrsub5.asm
;.....................................................
begin:  jmp     set_tsr
;.....................................
;INT 13  >AL=Sector Total,CH=Track No. CL=Sector No. DH=Side.
;INT 13  >DL=Driver(A:=00,B:=01,C:=80h,D:=81h )
;.....................................................
```

2 个外部子程序在 Tsrsub5.asm tsrsub6.asm 内

```
int13h_address   LABEL   dword
offset_int13h    dw      ?    ;Place For Save Origin INT 13h Offset Address
segment_int13h   dw      ?    ;Place For Save Origin INT 13h Segment Address
;.....................................
reentry_flag     db      1            ;Define reentry_flag=1..Can Do TSR
;=====================================================
isr_int13h:      串接BIOS系统调用 int 13h
        pushf  ;when call..push cs,push ip..when int-->pushf,push cs,push ip
        CALL   cs:int13h_address       ;call old int 13h isr,in order to change
                                  reentry flag=1,表示可以执行 ISR
;.....................................................
        PUSHREG                ;save all register to stack,include in tsrtsay.mac
        cmp    CS:reentry_flag,1  ;-->If reentry <> 1-->Do Nothing
        jne    do_nothing ;-->If reentry_flag=1-->Can Do ISR NOW.....
;.............................................
        mov    CS:reentry_flag,0  ;--->From Now..Can Do Not ISR Again
;.............................
        mov    ax,@data
        mov    ds,ax
        lea    si,show
        CALL   SHOWMARK   ;In Tsrsub5.asm,Pass In Si & attribute
        CALL   SOUND_ON    ;In Tsrsub6.asm,Pass Nothing
        mov    CS:reentry_flag,1  ;--->Isr Is Do Over..Set reentry=1..
;.............................................
do_nothing:
        POPREG         ;restore all register from stack...include in tsrtsay.mac
        IRET
tsr_final_address   SEGMENT
tsr_final_address   ENDS
;=====================================================
set_tsr:
        GET_VECT  13h            ;ah=35h,int 21h..TO [0:13h*4] get cs:ip to ES:BX
        mov    CS:offset_int13h,bx     ;save offset address of int13h(bx)
        mov    CS:segment_int13h,es    ;save segment address of int13h(ES) to memory
;.....................................
        push   cs
        pop    ds   ;--> .EXE Format,So..Set Ds=Cs
;.....................................          更改 int 13h 中断向量值
        cli                      ;set I=0,disable all IRQ when changing vect
        CHG_VECT  13h,isr_int13h ;ah=25,int 21h..mov address isr_int13h to 13h*4
;.............................
;       CHG_VECT  1ch,isr_int1ch ;ah=25,int 21h..mov address isr_int13h to 1ch*4
```

```
        sti                         ;set I=1,enable all IRQ
;.....................................
        mov     ah,62h
        int     21h             ;get PSP Segment address save to
;.............................
        mov     dx,SEG tsr_final_address
        sub     dx,bx       使程序常驻
;.....................................................
        KEEP_RES  0,dx              ;ah=31h,int 21h...keep code TSR,200h is size
;       lea     dx,set_tsr          ;you can use int 27 to keep TSR
;       int     27h                 ;but only can use below int 1ch
END     begin
```

程序分析

1. 在尚未执行 TSRDIRK2 前，先查看系统内存的分配占用情形。使用命令：
   ＞MEM/D ENTER (DOS 的命令)。

```
D:>BOOK>mem/d
Conventional Memory Detail:          没有看到 TSRDISK2 使用内存

  Segment        Total                Name          Type
  -------    -----------              --------      ------
   00000        1039     (1K)                      Interrupt Vector
   00040         271     (0K)                      ROM Communication Area
   00050         527     (1K)                      DOS Communication Area
   00070        2752     (3K)          IO          System Data
                                        CON         System Device Driver
                                        AUX         System Device Driver
                                        PRN         System Device Driver
                                        CLOCK$      System Device Driver
                                        A: - D:     System Device Driver
                                        COM1        System Device Driver
                                        LPT1        System Device Driver
                                        LPT2        System Device Driver
                                        LPT3        System Device Driver
                                        COM2        System Device Driver
                                        COM3        System Device Driver
                                        COM4        System Device Driver
   0011C        5456     (5K)          MSDOS       System Data
   00271       54865     (54K)         IO          System Data
               1152      (1K)            XMSXXX    Installed Device=HIMEM
               3104      (3K)            EMMXXX    Installed Device=EMM386
```

```
                44400   (43K)      DBLSB1      Installed Device=DBLSPA
                 1488   (1K)                   FILES=30
                  256   (0K)                   FCBS=4
COMMAND.COM常驻部分 12   (1K)                   BUFFERS=20
                  800   (1K)                   LASTDRIVE=I
                 3008   (3K)                   STACKS=9,256
    00FD6          80   (0K)       MSDOS       System Program
    00FDB        2640   (3K)       COMMAND     Program
    01080          80   (0K)       MSDOS       -- Free --
    01085        2064   (2K)       COMMAND     Environment
    01106         176   (0K)       MEM         Environment
    01111       88608   (87K)      MEM         Program
    026B3      496832   (485K)     MSDOS       -- Free --
```

2. 执行 TSRDISK2 后，再用 DOS 指令>MEM/D/P 查看内存分配情况(TSRDISK2.EXE 在磁盘内)。

```
Conventional Memory Datail:

  Segment        Total              Name          Type
  -------      ----------           --------      ------
    00000        1039    (1K)                    Interrupt Vector
    00040         271    (0K)                    ROM Communication Area
    00050         527    (1K)                    DOS Communication Area
    00070        2752    (3K)       IO           System Data
                                      CON         System Device Driver
                                      AUX         System Device Driver
                                      PRN         System Device Driver
                                      CLOCK$      System Device Driver
                                      A: - D:     System Device Driver
                                      COM1        System Device Driver
                                      LPT1        System Device Driver
                                      LPT2        System Device Driver
                                      LPT3        System Device Driver
                                      COM2        System Device Driver
                                      COM3        System Device Driver
                                      COM4        System Device Driver
    0011C        5456    (5K)       MSDOS        System Data
    00271       54865    (54K)      IO           System Data
                 1152    (1K)         XMSXXXX0   Installed Device=HIMEM
                 3104    (3K)         EMMXXXX0   Installed Device=EMM386
                44400    (43K)        DBLSBIN$   Installed Device=DBLSPACE
```

```
                1488    (1K)                    FILES=30
发              256     (0K)                    FCBS=4
现              512     (1K)                    BUFFERS=20
TSRDISK2        800     (1K)                    LASTDRIVE=I
常              3008    (3K)                    STACKS=9,256
00FD6 驻         80     (0K)    MSDOS           System Program
00FDB 了        2640    (3K)    COMMAND         Program
01080 ↓          80     (0K)    MSDOS           -- Free --
01085            2064    (2K)    COMMAND         Environment
01106            192     (0K)    TSRDISK2        Environment   ← MCB1
01112            3584    (4K)    TSRDISK2        Program       ← MCB2
011F2            176     (0K)    MEM             Environment
011FD            88608   (87K)   MEM             Program
0279F            493056  (485K)  MSDOS           -- Free --
```

在执行 TSRDISK2 后，发现 TSRDISK2 程序占用 2 块内存：

(1) 环境变量(在地址 01106h 处)(此地址会因系统不同而不同)

(2) 程序本身(在地址 01112h 处)

由于程序一直驻于内存，随时等候使用。所以我们必须选择触发来源，此处选择了 int 13h，int 13h BIOS 系统调用，参阅附录 D.

(3) 程序分析，请在阅读完本章后，再回头重新看一遍，自然而然便大致了解。此处只先做概略的分析。后面几个例子，可以更详细地看出 TSR 设计的基础。

程序流程 TSRDISK2.ASM(见下页)

程序说明

1. EXTRN SHOWMARK：NEAR

   EXTRN SOUND_ON：NEAR

   外部子程序，在 tsrsub5.asm 及 tsrsub6.asm 内，功能是协助程序 tsrdisk2.exe 发出声音及在屏幕出现反白的字符串标记(MARK)。

2. pushf 把 F 标志寄存器的值，存到 STACK 去。

   CALL cs：int13h_address

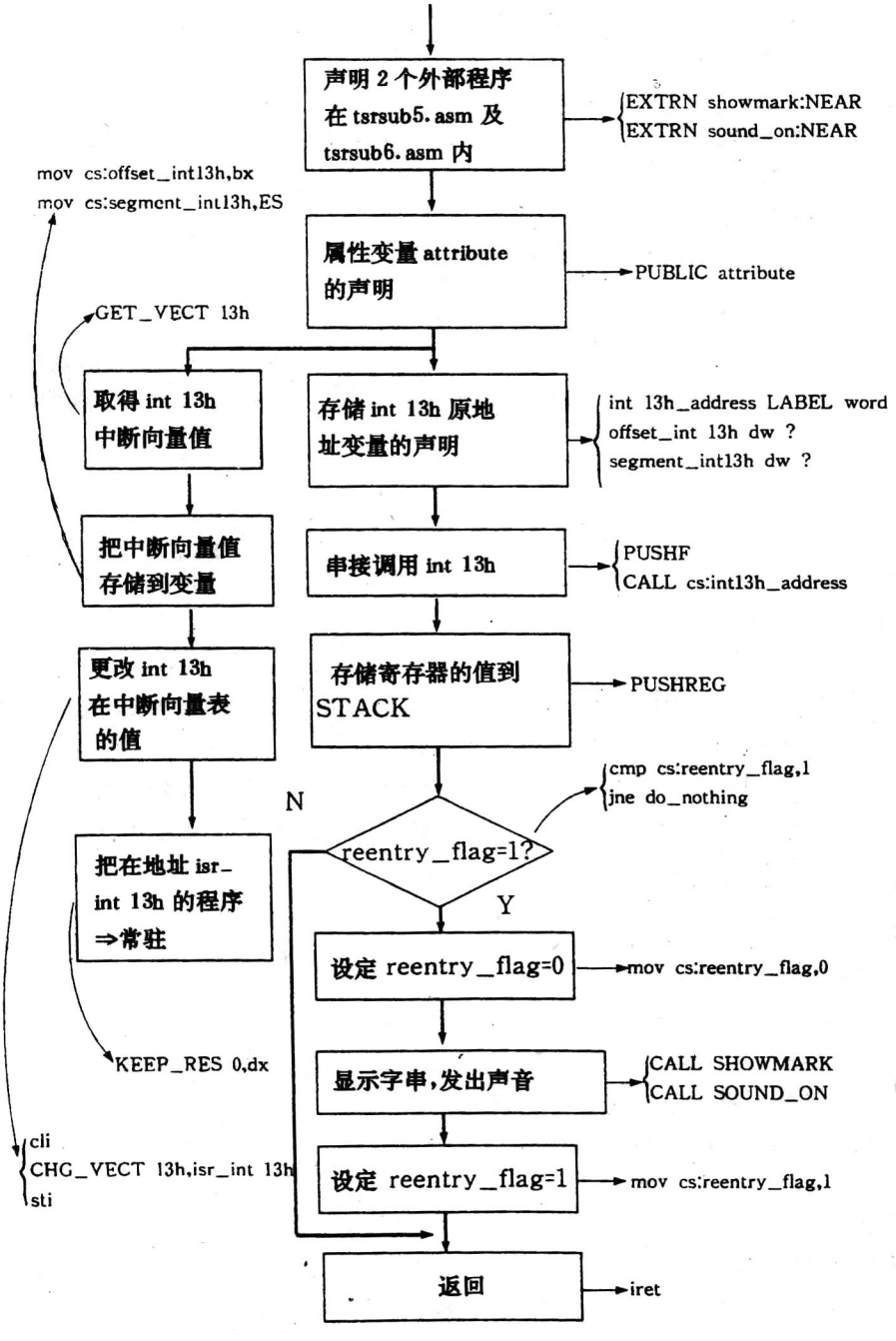
   调用原来的 int 13h 程序。CALL 对应 RET，但 int 必须对应 iret。而 iret 应先 popf，故使用 CALL，也需 pushf。

3. GET_VECT 13h...→取得 int 13h 中断向量值到 ES：BX，这是 DOS Ah＝35h 功能调用，放在 TSAY.H 内。

4. GHG_VECT 13h，isr_int13h...把中断向量表的值改变成 isr_int13h 所在地的地址，此宏定义在 TSAY.H 内，Ah＝25h...int 21h。

5. KEEP_RES 0，dx...，ah＝31h，int 21h，使程序常驻在内存。至于常驻原理就要谈到 M.C.B 的结构。请继续往下看。

   此处使用的宏定义指令，都定义在 TSAY.H 内，请详细查看附录 A。



1.1 什么是常驻程序

从 CP/M 到 DOS，一直都是属单用户操作系统，即无法在同一时间内，去处理两个以上的程序。

虽然 OS/2 解决了单用户的限制，成为多用户操作系统，但由于其后续软件的支持不足，未能流行，目前 Window 3.1 正在风行。

常驻程序这个概念的引入，给了 DOS 一个“多路”应用的可能性。常驻程序执行后，程序就隐藏在内存内，等待用户去触发或系统去触发，随时可以取用。就像 DOS 命令解释器（COMMAND.COM）常驻部分一样。常驻程序等待用户的触发的来源，如键盘，即俗称的“热键”，而由系统去触发的来源，如计时计数器（8253，8254 Counter #2），每 55ms 中断一次，引发 int 08h，再触发 int 1ch 的功能。如下图说明：

a. 用户触发常驻程序－如键盘

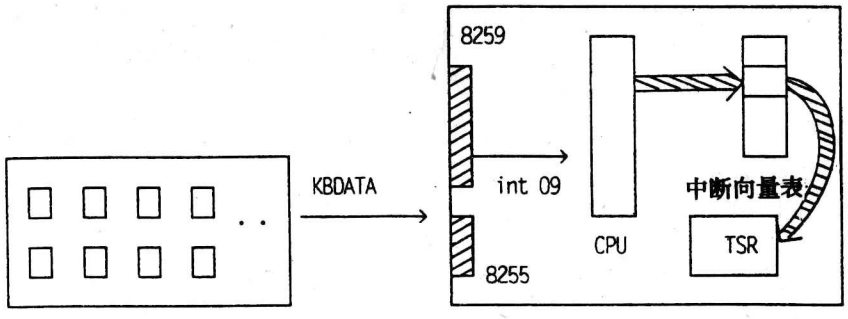


图 1.1

当用户触动键盘，若正好是预设的“热键”，即执行预先放在内存某处的“程序”，而此程序一直留在内存里，所以就叫此程序为常驻程序。

例如，DOS 也预设了“HARD－COPY”的热键，ptr＋SCR 键。

b. 由系统去触发－如 int 08h→int 1ch：由硬件定时（55ms）产生中断。

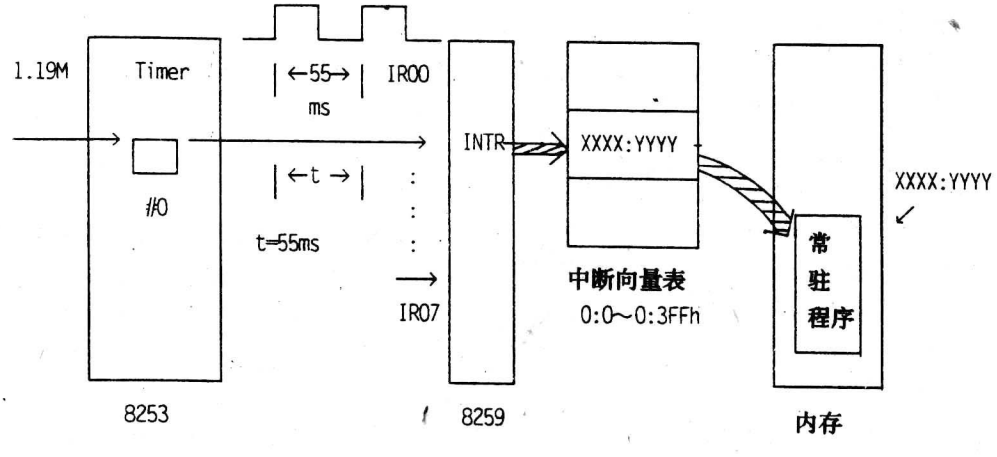


图 1.2

由系统计时计数到 0，每 55ms 产生中断要求信号，要求 CPU 执行程序。此时所占用 8259 的中断号码是 IRQ0。而 IRQ0，刚好对应中断服务程序 INT 08h。而 INT 08h 执行的功

能是：

1. 系统计时
2. 再执行 int 1ch

而在 int 1ch 内，只有一行指令“iret”—目的是留给程序员扩充其他常驻程序。

1.2 时间切割

在 DOS 下把常驻程序放到原来的 int 1ch 程序处，可以 55ms 去执行程序一次，如此窃取系统时间，来达到“多路”的目的。

事实上这种情况也不能叫“多路”，因为 CPU 在任何一个时间单位内只能去处理一个程序，只不过是把时间单位分得更短，用户感觉不出来而已（几个 ms），而这种时间的“切割(time－slice)”，正是多路操作系统的原理。

从 80286 到 80386 到 486 中的保护模式，也是采取这种“时间切割”的模式，来达到“时间共享”(time－sharing)的目的。

如下图 1.3 286 或 386 模式，有 4 个特权层次：

IOPL 4 个特权层次图(I/O Previlige Lever)

DPL－描述子特权等级的栏内，记载着描述子的特权等级，共 0～34 级。等级 0 最高，等级 3 最低。

80386 载入数据段到 DS，ES，FS，GS，SS 时检查三个特权：

1. CPL－现在特权等级。
2. RPL－指定目标选择子。
3. DPL－目标区描述子。

- 第三层特权等级最低，保留给用户使用。
- 操作系统在核心层执行
- 任务 A，任务 B，任务 C 分时执行，不互相交叉。

① 阶层 0＝核心(kernel)＝最高层(OS)

② 阶层 1＝系统服务

③ 阶层 2＝特定附加功能

④ 阶层 3＝应用系统＝最低层(用户应用程序)

在 4 个保护层中，最外层(阶层 3)要进入内一层，必须有特定的“调用门”也就是阶层 0，受到层层的保护。如此严密的防护，再使用时间分割一分配任务 A，任务 B，任务 C ……才能确保多路系统的安全，不被瓦解。这样的系统在 CPU 286 以上都有。但在“Real－mode”下，也就是 DOS 操作系统下，只能采取“分段”式的寻址，无法使用这种保护模式。

也就是说这样安全的防护，正是“DOS”所欠缺的。所以在 DOS 下设计常驻程序时，很容易就使系统的核心受到破坏而瓦解－死机。

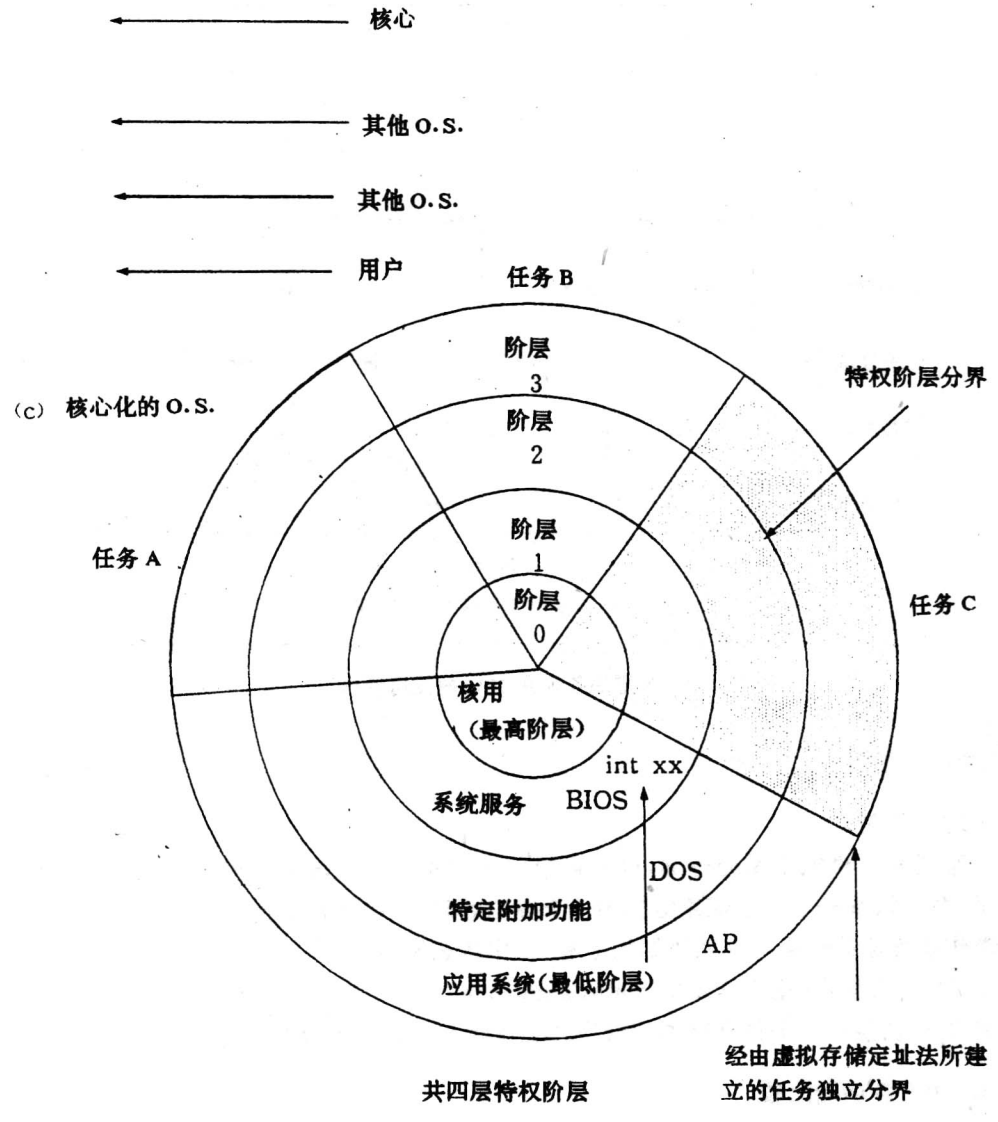


图 1.3　保护模式下的作业中有 4 个特权层次存在

1.3　多重进程改进 CPU 使用情形——分时实例

如下图 1.4,横轴代表“时间”,纵轴代表“任务”。如在 1 秒以内,键盘处理受到 5 次的激发,而命令解释器受到 3 次激发…这么短的时间内,CPU 必须执行 6 件工作。系统非得受到“严密保护”不可。

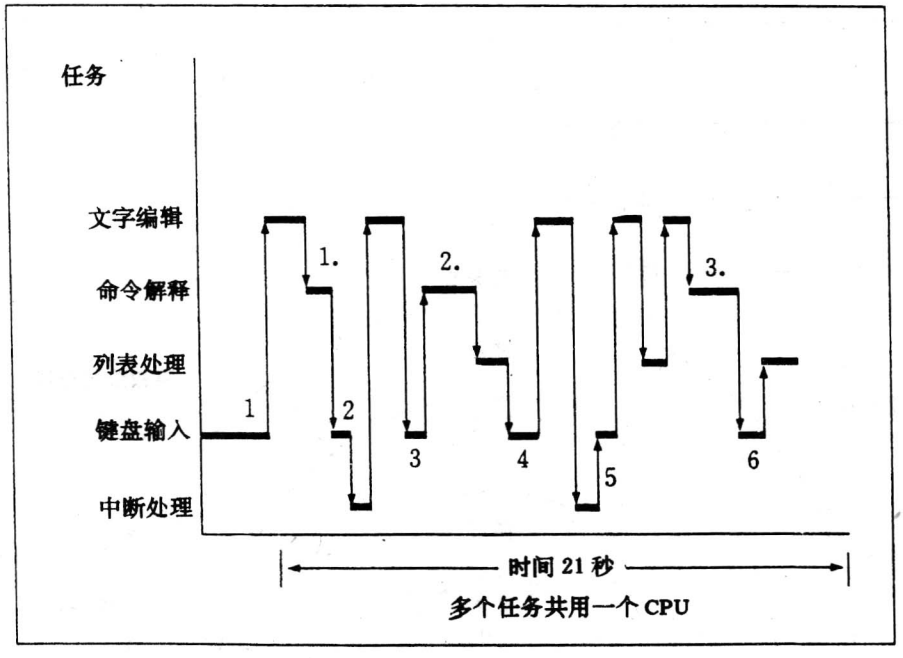


图 1.4 多重任务改进 CPU 的使用情形

80386 分时执行多路

1. 采时间分割。
2. 多路任务－模拟多重处理器，以提供给每个工作者一个虚拟处理器，在任何一时间里，系统指派一个实处理器给其中一个虚拟处理器，以执行工作。
3. 操作系统连续切换——系统结构上有“工作状态段”：
   ① TSS(工作状态段)(TASK status segment)
   ② 工作切换——工作寄存器记录现在工作状态(TR)(TASK Register)

PC/XT、AT、386 各功能比较

CPU 8088/8086 的结构不适于“多路作业”，但仍然可以执行常驻程序，因为计时计数器，在 XT、AT、80386、80486 内都设计定位在同样的 int 08h 中，键盘处理也设计在 int 09h 中，各 CPU 的功能特色比较，如下图 1.5。

PC/XT 与 PC/AT 各模式功能比较

(一) 80386 系统功能

① 可执行所有 8086,8088,80286,80386 的机器码

② 对不同操作系统具有兼容性——如 UNIX,ZENIX,DOS

(二)80386 芯片硬件特色

① 多路任务(Mltitasking)＝＞把各项工作指定给虚拟处理器，每次切换动作只需 70～

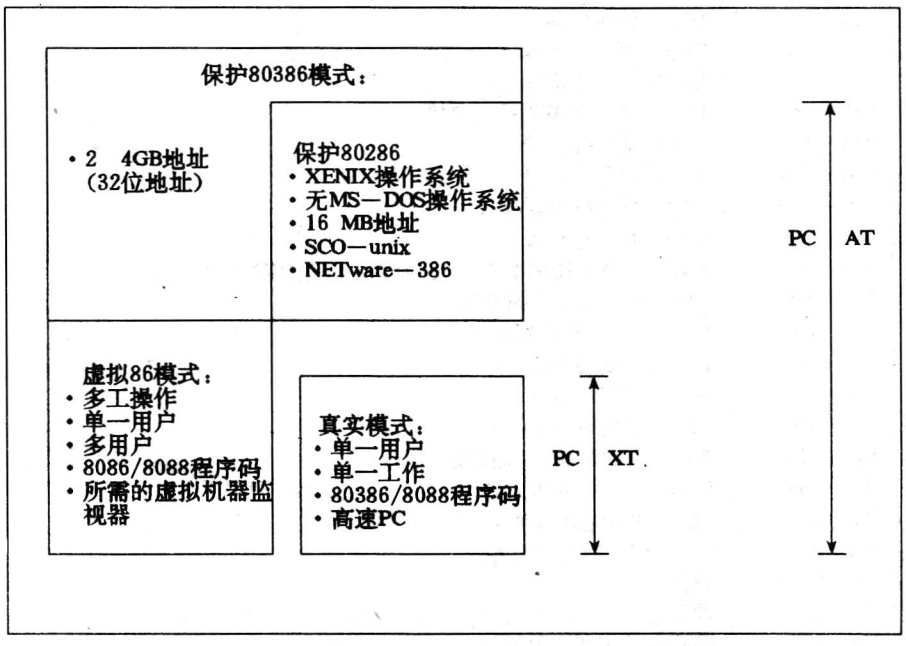


图 1.5　80386 系统实模式和虚拟模式的工作范围

80 个 32 BIT 的 WORD，时间需要 17—20 u。

② 高度的软件兼容性

(a) real mode

(b) Virtual mode

③ 高速的运算能力——采用 1.5 um CH MOS III 技术。16 MHZ 下，3～4 MIPS(million instruction per s)。在 20—25M 下，4～5 MIPS 每秒执行 4～5 百万个 risc 指令。

④ 高效能的操作结构—6 个单元，以上是对于 386 概略的探讨，若要深入研究，请参考附录的参考书目。

而 PC 系统的软、硬件中断 int0～int ffh 功能分配如下表 1.1：

表 1.1　系统中断分配表

编号	向量 (10 进制)	用途
00	000—003	CPU：表示遇到除以零的合理运算
01	004—007	CPU：单步执行
02	008—00B	CPU：NMI(不可屏蔽式的中断)
03	00C—00F	CPU：设定中断点(Break point)
04	010—013	CPU：数学运算的溢出错误(Overflow)
05	014—017	Hard copy(屏幕内容的复制，又称硬拷贝)
06	018—01B	保留(80286 使用) IRO 硬件中断对应的软件中断
07	01C—01C	保留