# DOS DEBUG 3.31 详解

李启龙　编写

北京希望电脑公司
1992.8

# 目　录

# 第一章　DOS 基础

## 1.1　中断和标志寄存器

MS－DOS 是一个建立在 8086 系列 CPU 上的一个操作系统，它的硬件中断通过口地址 21h 的设置来进行屏蔽、许可，软件中断则通过执行 Int n 指令来完成。

中断屏蔽寄存器（口地址 21h）各位定义如下：

| 位： | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
|  | lpt1 | fdc | hdc | com2 | com1 | i/o | kbd | clock |

如果某一位为 0 则允许该设备中断，为 1 则禁止该设备中断，使之不能影响系统。

对程序执行过程产生影响的有关软件中断有以下几个：

|  |  |
|---|---|
| Int0 | 执行除数为零的除法运算时产生此中断 |
| Int1 | 单步中断，TF 等于 1 时每执行一条指令产生此中断 |
| Int2 | 不可屏蔽中断 |
| Int3 | 断点中断 |
| Int4 | 溢出中断，OF 为 1、执行 Int0 时产生 |
| Int20h | 结束程序 |
| Int22h | 程序结束地址 |
| Int23h | Ctrl－Break 退出地址 |
| Int24h | 致命错误处理程序 |

8086 系列 CPU 含有一个十六位的标志寄存器，其各位定义如下：

| 位： | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | — | — | — | — | OF | DF | IF | TF | SF | ZF | — | AF | — | PF | — | CF |

意义如下：

|  |  |
|---|---|
| CF： | 进位标志。运算指令执行之后，最高位产生进位、借位时，置 1。 |
| PF： | 奇偶校验标志。运算指令执行后，结果数值低 8 位中含 1 的位数为偶数时，置 1 |
| AF： | 辅助进位标志。运算指令执行后，低 4 位上产生借位或大于 10 产生进位时，置 1 |
| ZF： | 全零标志。运算指令执行后，结果为全零时，置 1 |
| SF： | 符号标志。运算指令执行后，最高位的值保持原值不变 |
| TF： | 陷阱标志。TF＝1 时，一执行指令就产生中断 |
| IF： | 中断允许标志。IF＝1 时可以接收中断，IF＝0 时中断被屏蔽 |
| DF： | 方向标志。用于指定字符串处理指令的方向，DF＝0 时由低位地址向高位地址处理，DF＝1 时相反。 |

OF：　溢出标志。运算指令执行后，结果数超过表示范围时，置 1

## 1.2　存储器组织和地址形成

8086 系列 CPU 系统具有不少于 1Mb 的存储器地址空间，但它的内部寄存器都只有 16 位、不能直接寻址 1Mb 的地址空间，因此引入了分段内存的思想。一个实际的物理地址由段和偏移二个 16 位寄存器一起来指定。设 Addr 为物理地址、Seg 为段地址、Ofs 为偏移地址，则由段和偏移映射的物理地址为：

Addr ＝　( Seg SHL 4 ) ＋ Ofs

若已知物理地址，则段和偏移可以为：

Seg ＝ Addr SHR 4

Ofs ＝ Addr AND 0000Fh

一组偏移和段地址映射出唯一的物理地址，而一个物理地址可以映射出不同的段和偏移的组合。比如，物理地址 410h 可以为：40h：10h、或 41h：0h、或 0：410h。

## 1.3　程序段前缀 PSP

打入一条外部命令、或通过 EXEC 功能调用执行程序，DOS 确定出最低的可用地址作为程序可用内存的开始地址。这一区域称之为程序段。

在程序段偏移量 0 处，DOS 建立一个 256 字节的程序段前缀 PSP 控制块。EXEC 在偏移量 100h 处装入程序并给与控制权。

对 COM 程序

- 四个段寄存器指向 PSP
- IP 设置为 100h
- SP 寄存器设置到程序段末尾
- 所有内存分配给程序，PSP 段偏移量六处包含有本段可用字节数的段落大小

PSP 格式如表 1－1 所示：

表 1－1　程序段前缀 PSP

| 开始字节 | 长度（字节数） | 内容 |
|---|---|---|
| 0 | 2 | Int 20h 指令 |
| 2 | 2 | DOS 内存顶部 |
| 4 | 1 | Reserve |
| 5 | 5 | DOS Far CALL |
| 0Ah | 4 | 程序结束地址 |
| 0Eh | 4 | Ctrl－Break (int23h) Vector |
| 12h | 4 | Int24h Vector |
| 16h | 2 | 父 PSP 段地址 |

| | | |
|---|---|---|
| 18h | 20 | FHT ( File Handler Table ) |
| 2Ch | 2 | 环境段地址 |
| 2Eh | 4 | DOS SS、SP 保留区 |
| 32h | 2 | FHT 长度 |
| 34h | 4 | FHT 地址 |
| 38h | 24 | Reserve |
| 50h | 3 | Int 21h 指令 |
| 53h | 2 | Reserve |
| 55h | 7 | FCB1Ext |
| 5Ch | 9 | FCB1 |
| 65h | 7 | FCB2Ext |
| 6Ch | 20 | FCB2 |
| 80h | 1 | 命令行长度 |
| 81h | 127 | 命令行 Buffer |
| 80h | 128 | 磁盘传送区 DTA |

对 EXE 程序

- DS、ES 指向 PSP
- CS、IP、SS 和 SP 为连接程序传送值

DOS 有一组操作 PSP 的中断调用，如下所述：

1. 建立新的程度段

   Entry： AH = 26h or 55h DX = Seg Address

   当前程度段位置 0 处的 256 个字节被 Copy 到新的程度段位置 0 处，并更新新的 PSP 中有关内容。

2. 取当前 PSP 地址

   Entry： AH = 51h or 62h

   Exit ： BX = Active PSP Address

3. Set PSP Address

   Entry： AH = 50h, BX = PSP Address

   Set Active PSP to [BX]

## 1.4 可执行程序及其结构

DOS 的可执行程序有 BAT、COM、和 EXE 三类。BAT 批处理程序是文本文件，COM 和 EXE 是二进制文件。

表 1－2 EXE 文件头

| 开始字节 | 长度（字节数） | 内容 |
|---|---|---|
| 0 | 2 | 标志字 5A4Dh |
| 2 | 2 | 模为 512 的映象长度 |
| 4 | 2 | 以 512 字节为增量的文件大小 |
| 6 | 2 | 重定位表项个数 |
| 8 | 2 | 以 16 字节为增量的文件头大小 |
| 0Ah | 2 | 在装入程序尾端上方需要的 16 字节段落的最少数量 |
| 0Ch | 2 | 在装入程序尾端上方需要的 16 字节段落的最大数量 |
| 0Eh | 2 | SS 位移值 |
| 10h | 2 | SP 值 |
| 12h | 2 | 字检验和 |
| 14h | 2 | IP 值 |
| 16h | 2 | CS 位移值 |
| 18h | 2 | 第一个重定位项在文件中的位移 |
| 1Ah | 2 | 覆盖号 |
| ··· | ··· | 可变保留区 |
| ··· | ··· | 重定位表（起点由 18h 指出） |
| ··· | ··· | 可变保留区 |

COM 程序是一种将代码段、数据段和堆栈段合一的紧凑格式的程序，所有信息都组合在一个段内、不得超过 64K 字节，而且起始地址必须是 100h。因它不需重定位，所以装入速度较快。

EXE 程序不同于 COM 程序，可以有独立的数据段和堆栈段，甚至可以有多个代码段，程序信息也可以超过 64K 字节、起始地址可以任意指定。这种结构的程序适合于较复杂的大型程序和多用户实时环境。

一个规则的 EXE 程序由二部分组成：文件头和装入模块，实际中、大一点的 EXE 程序还可能包含一个附加部分，此部分由开发者用连接程序以外的工具附加到程序末尾、不属于装入模块、也不直接装入内存，仅供程序本身使用。比如：AUTOCAD 的 ACAD. EXE，DOS 文件有 1. 77Mb、而装入模块只有 131Kb ，Turbo C＋＋ Ver 2. 00 的 BC. EXE，DOS 文件有 1Mb 多、而装入模块只有 155Kb 。

文件头包含有文件的控制信息和重定位信息，供 DOS 装入程序时重定位和转移控制用、不装入内存。其格式如表 1－2 所示：

DOS 加载 EXE 程序后，根据文件头的重定位表项按以下次序进行重定位：

1. 将每个重定位项中的段值加上开始段值

2. 由此段值和重定位项中的位段值指向内存被装模块中的一个字
3. 取出该字再加上开始段值
4. 将上述结果存入原位置中

然后初始化寄存器：

1. 取文件头位移 10h—11h 的值送 SP，位移 0Eh—0Fh 的值加开始段值送 SS
2. 将 PSP 的段值送 DS、ES
3. 取位移 16h—17h 的值加开始段值送 CS，取位移 14h—15h 的值送 IP，最后将控制转向 CS：IP，执行程序。

# 第二章 · DEBUG 调试程序

DEBUG 提供了一个可控制的检测环境、使你能监视程序的执行，直接对 COM 或 EXE 文件做些改变而不必先重新汇编源文件、然后立即执行它以决定该变化能否定位错误，DEGUG 允许你装入、修改或显示任何文件，和执行目标文件。

## 2.1 还原 DEBUG. COM

DOS 3.31 提供给用户的 DEBUG 是一个大小为 16000 个字节的 COM 程序，不过、经过分析就会发现，它的内核实际上只是一个大小为 15805 字节的 EXE 程序。

DEBUG. COM 的第一条指令是跳转指令、紧跟着是一个说明字符串及 512 字节的 EXE 文件头。第一条指令转去执行初始化程序，完成三个工作：

1. 根据文件头，对调入内存的映象文件重定位
2. 将 EXE 映象代码移位到 PSP 段的 100h 偏移处
3. 转去执行映象代码

所以我们可以用 DEBUG 将 DEBUG. COM 调入内存，将它还原成 EXE 文件，以后分析或还原成汇编语言程序就会方便一些。

还原 DEBUG. COM 的过程如下：

```
c:>   DEBUG debug.com
—     M CS:110 L 3DBD CS:100
—     R CX
:     3DBD
—     N x
—     W
—     Q
c:>   REN x debug.exe
```

DEBUG. COM 中　DEBUG. EXE 的文件头

```
0100   E9 3DD2     jmp    3ED5h ;DEBUG.COM 的第一条指令
                  ————————
                         说明字符串
                         ————————
                                 EXE 文件的文件头
                                 ————————
         ————————————————————————————————————————————————————
000000: E9 D2 3D 43 6F 6E 76 65  72 74 65 64 00 00 00 00 ..=Converted....
         ————————————————————————————————————————————————————>
000010: 4D 5A BD 01 1F 00 03 00  20 00 00 00 FF FF 66 03 MZ...........f.
000020: 6A 01 7D A4 00 01 28 00  1E 00 00 00 01 00 DE 03 j.}...(.........
000030: 28 00 E5 03 28 00 EE 03  28 00 00 00 00 00 00 00 (...(...(.......
000040: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000080: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000090: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0000A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0000B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0000C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0000D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0000E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0000F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000100: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000110: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000120: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000130: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000140: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000150: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000160: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000170: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000180: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000190: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0001A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0001B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0001C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0001D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0001E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
0001F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
000200: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................
```

DEBUG.COM 的初始化程序：

```
        3ED5  call  3ED8h            ; goto 3ED9h
        3ED8  pop   bx               ; bx <—— 3ED8h
        3ED9  push  ax               ; ax = 0
        3EDA  mov   ax,es            ; PSP
        3EDC  add   ax,10h           ; PSP:100h
        3EDF  mov   cx,[11Eh]        ; SS offset
        3EE3  add   cx,ax            ; SS
        3EE5  mov   [bx-5],cx        ; save SS
        3EE8  mov   cx,[126h]        ; CS offset
```

```
3EE9  add   cx,ax              ; CS
3EEE  mov   [bx-9],cx          ; save CS
3EF1  mov   cx,[120h]          ; SP
3EF5  mov   [bx-7],cx          ; save SP
3EF8  mov   cx,[124]           ; IP
3EFC  mov   [bx-0Bh],cx        ; save IP
3EFF  mov   di,[128h]          ; 1st reloc table' s offset
3F03  mov   dx,[118h]          ; head' s length
3F07  mov   cl,4
3F09  shl   dx,cl              ; convert pargraph to bytes
3F0B  mov   cx,[116]           ; reloc tables
3F0F  jcxz  3F2Bh
3F11  lds   si,es:[di + 110h]  ;重定位
3F16  add   di,4
3F19  mov   bp,ds
3F1B  add   bp,es:[118h]
3F20  add   bp,1
3F23  add   bp,ax
3F25  mov   ds,bp
3F27  add   [si],ax
3F29  loop  3F11h
3F2B  push  cs
3F2C  pop   ds
3F2D  mov   di,100h
3F30  mov   si,dx
3F32  add   si,110h
3F36  mov   cx,bx
3F38  sub   cx,si
3F3A  rep   movsb              ; move image codes
3F3C  pop   ax
3F3D  cli
3F3E  mov   ss,[bx-5]
3F41  mov   sp,[bx-7]
3F44  sti
3F45  jmp   dword ptr [bx-0Bh]; goto to image code
3F48  db 0Dh, 0Ah, 2Ah, 2Ah, 2Ah, 20h, 28h, 43h, 29h
3F51  ' Copyright Compaq Computer Corporation 1987 * * *'
```

## 2.2 进入 DEBUG

在 DOS 状态下，为启动 DEBUG、请按下列格式键入：

[ 驱动器 ][ 路径 ] DEBUG [ 驱动器 ][ 路径 ][ 文件名[ 扩展名 ]]
[ 参数 1][ 参数 2 ]

可以只键入 DEBUG 命令，或包含文件说明。参数 1 和参数 2 表示要调试的程序的命令行参数。

进入 DEBUG 后出现提示符"－"，"－"告诉用户 DEBUG 已经就绪，可以接受修改、显示或执行内存中程序内容等命令。如果你只键入 DEBUG 而无文件说明，可以对出现在内存的内容进行工作，或者用 N(命名)和 L(装入)命令把需要的文件装入内容。

进入 DEBUG 后，寄存器的值初始化如下：

- 段寄存器( CS、DS、ES，和 SS )设置到空闲内存的底部，即 DEBUG 程序后的第一段
- IP <—— 100h
- SP 设置到该段末尾或装入程序暂存部分的底部中位置较低的地方。PSP 段位移 6 处的段大小减少 100h 以供 Stack 用
- 其余寄存器( AX、BX、CX、DX、BP、SI，和 DI )全置成 0。如果进入 DEBUG 时带有文件说明、则 BX、CX 含有文件大小，其中 BX 为高位部分
- 标志寄存器设置为：
  NV UP EI PL NZ NA PO NC
- 缺省的 DTA 为 CS:80h

进入 DEBUG 后，所有可用内存均被分配、因此被装程度不能分配内存。

- 如装入 EXE 文件，则重定位并根据文件头参数设置有关寄存器：CS、IP、SS、SP。DS、ES 置成 PSP，BX、CX 为映象代码的大小
- 如果装入的是 HEX 文件，则假设为 INTEL 16 进制格式、并转换成二进制。

进入 DEBUG 提示状态后就可以输入命令行，命令行可以是单个字母、或后随一个或多个参数，命令行可以有小写或大写或大、小写混合的字符，命令和参数由界符分隔、不过只有连续的二个 16 进制数才需分隔。DEBUG 的数值均为十六进制数。

DEBUG 共有 19 个合法命令，如表 2－1 所示。有关参数的说明见表 2－2。

表 2－1 DEBUG 命令一览表

| 命令 | 格式说明 |
|---|---|
| A( 汇编 ) | A [address] |
| C( 比较 ) | C range address |
| D( 转储 ) | D [address]、or D [range] |
| E( 改写 ) | E address [list] |
| F( 填写 ) | F range list |

| | |
|---|---|
| G( 执行 ) | G [=address] [address [address. . . ]] |
| H( HEX 运算 ) | H value value |
| I( 端口输入 ) | I portaddress |
| L( 装入 ) | L [address [drive sector sector]] |
| M( 传送 ) | M range address |
| N( 命名 ) | N [d:] [path] filename[. ext] [param [param. . . ]] |
| O( 端口输出 ) | O portaddress byte |
| P( 步进 ) | P [=address] [value] |
| Q( 退出 ) | Q |
| R( 寄存器 ) | R [registername] |
| S( 检索 ) | S range list |
| T( 跟踪 ) | T [=address] [value] |
| U( 反汇编 ) | U [address] |
| W( 写 ) | W [address [drive sector sector]] |

表 2—2　DEBUG 命令参数说明

| 参数 | 说明 |
|---|---|
| address<br>地址 | 有三种格式:<br>1.　段寄存器:偏移值,如　CS:100<br>2.　段值:偏移值,如　3F6:100<br>3.　偏移值,如　100<br>值可以由 1—4 个 HEX 字符组成 |
| byte<br>字节 | 由 1—2 个 HEX 字符组成 |
| drive<br>驱动器 | 1 或 2 个数字,0 为驱动器 A、1 为驱动器 B |
| filespec<br>文件标识 | 由驱动器名、文件名和后缀组成 |
| list<br>列表 | 由 1 个或多个字节或/和字串组成 |
| portaddress<br>口地址 | 用 1 到 4 个 HEX 字符定义的 8 或 16 位端口地址 |
| range<br>范围 | 有二种格式:<br>1.　address address<br>第二个地址只需偏移值<br>2.　address L value |
| registername | 参见 R 命令 |

| | |
|---|---|
| 寄存器名 | |
| sector sector<br>扇区　扇区 | sector 由 1－3 个 HEX 字符组成，定义起始相对扇区号和扇区数 |
| string<br>字符串 | 由单引号或双引号括住的一串 ASCII 字符 |
| value<br>值 | 1－4 个 HEX 字符组成的数据 |

## 2.3 DEBUG 总控程序

DEBUG 总控程序如下：

```
start:
          Verify DOS version
          Save vector of INT01h & INT03h, for restore
          Set INT22h address
          Creat sub _ PSP
          Get screen parameters
          处理命令行参数
loop:
          Initialize registers & flags unit
          Printf("－")
          Get KBDline then UPPER it
          Dispatch to subroutine, if is Q command then exit DEBUG
          Goto loop
```

## 2.4 A（汇编）'命令

格式：A [ 地址 ]

A 命令将 IBM PC 宏汇编语句直接写入连续的内存地址中，而不需要汇编、连接。未指定地址时，用 CS:100h 或上一次 A 命令的后续地址作起始地址。

输入出错时，报告：

```
      ^ Error
```

并重新显示当前地址，等待输入。

DEBUG 支持标准的 8086/ 8088 汇编语言语法和 8087 指令系统。

例：

```
－A
178B:0100 NEG BYTE PTR DS:[200]
                             Error
178B:0100 DS:
```

```
178B:0101 NEG BYTE PTR [200]
178B:0105 INC WO [SI]
178B:0107 FWAIT FADD ST,ST(3)
178B:010A POP [BP+DI]
178B:010C RETF
178B:010D DB 1,AB,"HELLO"
178B:0114
```

## 2.5 C（比较）命令

格式:C 范围 地址

C 命令比较内存中两个数据块的内容，长度由输入的范围确定。缺省的段地址是 DS。如果找到了不匹配的单元，按下面格式显示：

地址 1　字节 1　　字节 2　地址 2

例：

```
C 100 L50 200
```

## 2.6 D（转储）命令

格式:D [ 地址 ]、或　D [ 范围 ]

D 命令用于显示内存中内容，缺省的段地址是 DS、缺省的偏移地址是 100h 或上一次 D 命令的后续地址，缺省的长度为 80 字节。

显示的格式为：

地址　十六进制码　ASCII 码

不可打印字符的 ASCII 码用'.' 显示，第一行自动调整边界。

例：

```
-d15 120
4436:0010                    3B 61 3A-01 01 01 00 02 FF FF FF       ;a:........
4436:0020  FF FF FF FF FF FF FF FF-FF FF FF FF 40 3B 25 08  ............@;%.
4436:0030  61 3A 14 00 18                                    a:...
-
```

## 2.7 E（改写）命令

格式:E 地址　[ 清单 ]

E 命令用清单中所包含的值替换从指定地址开始的 1 个字节或多个字节的内容，也可用于按顺序方式显示和修改字节。缺省的段地址是 DS。

用于顺序方式的修改时，输入空格将步进到下一地址、输入'－' 则退回到前一地址，输入回车结束。

例：

```
—E 200 AB 02 'Hello'
—E
  ^ Error
—E 200
3404:0200  AB.cd   02.04
```

## 2.8 F（填充）命令

格式:F 范围　清单

F 命令用清单中的值填写指定范围的内存单元,缺省的段地址是 DS。　如果清单字节数少于地址范围,则重复使用清单内容,直到填满。如果清单字节数大于地址范围,则忽略多余部分。

例:

```
—F 200 L40 'Hello'
—
```

## 2.9 G（执行）命令

格式:G [ = 起始地址 ] [ 断点地址 [ 断点地址 ... ] ]

G 命令执行程序,直到结束或遇到断点(遇到断点时还显示寄存器、标志位和下一条命令)。缺省的起始地址是 CS: IP。

DEBUG 用插入 INT03h 的办法设置断点,最多可以设置 10 个。

不能在 ROM 中设置断点。

例:

```
—G=0 1A

AX=0000  BX=179C  CX=4E60  DX=17BA  SP=0400  BP=A000  SI=0000  DI=0000
DS=17BA  ES=17A4  SS=17D2  CS=17B7  IP=001A   NV UP EI PL NZ NA PO NC
17B7:001A 891E0800      MOV      [0008],BX                   DS:0008=0000
```

## 2.10 H（运算）命令

格式:H 值 1　值 2

H 命令计算两个数的和和差,显示格式如下:

和　差

例:

```
—H 12 0A
001C  0008
—
```

## 2.11 I（输入）命令

格式:I 端口地址

I 命令用于从端口输入一个十六进制字节，显示如下：

一个字节

例：

```
    -I 21
    B8
```

## 2.12 L（装入）命令

格式:L [地址[驱动器　起始扇区　扇区数]]

L 命令用于装入文件或磁盘扇区。缺省的段地址是 CS。

DEBUG 装入的文件说明在 CS：80h 处，必须由 DEBUG 的命令行或用 N 命令指定。

EXE 文件和 COM 文件在 CS：100h 处装入。

一次读取的最大磁盘扇区数是 80h 。

例：

```
    -L
    -L DS:100 0 0 2
```

## 2.13 M（传动）命令

格式:M 范围　地址

M 命令将范围指定的内存单元内容传送到以地址开始的单元中。缺省的段地址是 DS。

例：

```
    -M CS:200,350 400
```

## 2.14 N（命名）命令

格式:N [ D：][ 路径 ]文件名[ 扩展名 ][ 参数 ]

N 命令将头两个文件的 FCB 格式化到 CS：5C 和 CS：6C 中，供 L 、W 命令使用。返回时 AX＝0FFh 则说明输入的文件中驱动器无效。

例：

```
    -N \JTLEE\EXE\CATALOG. EXE
```

## 2.15 O（输出）命令

格式:O 端口地址　字节

O 命令将一个字节输出到指定端口。

例：

```
—O 21 B8
```

## 2.16　P（步进）命令

格式：P [=起始地址][步数]

P 命令使执行子程序调用、循环指令、中断或重复字符串指令后停在下一指令处。使用插入 INT03h 的方法停止执行。

例：

```
—P

AX=0000  BX=179C  CX=4E60  DX=17BA  SP=0400  BP=A000  SI=0000  DI=0000
DS=17BA  ES=17A4  SS=17D2  CS=17B7  IP=001E   NV UP EI PL NZ NA PO NC
17B7:001E 892E2A00      MOV     [002A],BP                       DS:002A=0000
```

## 2.17　Q（退出）命令

格式：Q

用于退出 DEBUG 。

## 2.18　R（寄存器）命令

格式：R [寄存器名]

R 命令用于显示、修改单个寄存器内容，显示当前寄存器和下一条要执行的指令。有效的寄存器名是：

```
AX   SP   CS   IP
BX   BP   DS   F
CX   SI   ES
DX   DI   SS
```

其中 F 是标志寄存器，IP 是指令指针。

标志寄存器各位表示如下：

| 名称 | 置位(1) | 清位(0) |
|---|---|---|
| 溢出(是、否) | OV | NV |
| 方向(减、增) | DN | UP |
| 中断(允许、禁止) | EI | DI |
| 符号(负、正) | NG | PL |
| 零(是、否) | ZR | NZ |
| 辅助进位(有、无) | AC | NA |
| 奇偶(偶、奇) | PE | PO |

进位(有、无)　　　CY　　　NC

例:

```
-R
AX=0000  BX=179C  CX=4E60  DX=17BA  SP=0400  BP=A000  SI=0000  DI=0000
DS=17BA  ES=17A4  SS=17D2  CS=17B7  IP=001E   NV UP EI PL NZ NA PO NC
17B7:001E 892E2A00       MOV     [002A],BP                       DS:002A=0000
-R AX
AX 0000
:12
-R F
NV UP EI PL NZ NA PO NC  -OV
-
```

## 2.19 S(检索)命令

格式:S 范围　清单

S 命令用于检索范围地址中含有清单内容的地址,找到匹配单元则显示其地址、否则没有任何显示。缺省的段地址是 DS。

例:

```
-S CS:0 L 1000 CD 21
17B7:05B5
17B7:05B9
17B7:0BE1
-
```

## 2.20 T(跟踪)命令

格式:T [=起始地址][步数]

T 命令用于跟踪执行指令,通过 INT01h 实现。

例:

```
-T 2

AX=0000  BX=0000  CX=4E60  DX=17BA  SP=0400  BP=A000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17D2  CS=17B7  IP=0003   OV UP EI PL NZ NA PO NC
17B7:0003 2E             CS:
17B7:0004 89162B00       MOV     [002B],DX                       CS:002B=0000

AX=0000  BX=0000  CX=4E60  DX=17BA  SP=0400  BP=A000  SI=0000  DI=0000
DS=17A4  ES=17A4  SS=17D2  CS=17B7  IP=0008   OV UP EI PL NZ NA PO NC
17B7:0008 8B2E0200       MOV     BP,[0002]                       DS:0002=A000
-
```

## 2.21 U（反汇编）命令

格式:U [地址],或 U [范围]

U 命令用于反汇编内存中指令,缺省的段地址是 CS、偏移量是 100h 或上一次 U 命令的后续地址。

例:

```
—U25 2B
17B7:0025 50            PUSH     AX
17B7:0026 B80000        MOV      AX,0000
17B7:0029 50            PUSH     AX
17B7:002A CB            RETF
17B7:002B BA1700        MOV      DX,0017
—
```

## 2.22 W（写）命令

格式:L [地址[驱动器 起始扇区 扇区数]]

说明见 L 命令,区别在于 W 是写、L 是读。

不能写 HEX 、EXE 文件。

例:

```
—W
EXE and HEX files cannot be written
—W DS:100 0 0 2
—
```

## 2.23 DEBUG 的汉化

西文软件的汉化一是输入的汉化,一是输出的汉化。DEBUG 的输入通过调用 DOS 的中断 AH = 0Ah 实现,本身可以接收汉字、不需改动,所以汉化的问题主要是输出汉字。经分析发现有二个子程序与此有关:一个是 SUB _ 23 xitoa() 子程序中的语句( offset=406h)

```
        and al, 7Fh
```

将大于 7Fh 的字符屏蔽掉了,另一个是 SUB _ 31 转储命令显示的子程序中的一段指令:

```
        ( offset = 56Dh )
locloop _ 75
        lodsb           ; load [si] to al
        cmp  al,7Fh
        jae  loc _ 76   ; mask extended ascii chars
        cmp  al,20h     ;
        jae  loc _ 77
```

```
loc_76:
        mov al,2Eh      ; '.', if ! isprint( [al] ) then print '.'
loc_77:
        stosb           ; Store al to es:[di] for print
        loop locloop_75
```

将英文中不可打印字符用'.'显示。所以将 offset = 406h、570h 处的指令分别改成二个空语句后即可显示汉字。

## 2.24 增强 DEBUG 的跟踪能力

DEBUG 使用中断 INT01h、INT03h 来单步执行程序和设置断点，其它的调试程序也一样有些程序采取反跟踪措施，运行中修改中断 1、3 的入口地址，或将中断 1、3 的向量改成运行中需要的数据，从而给分析和跟踪带来困难。所以要是用其他的中断替代中断 1 和 3 就会方便一些。如果替代的中断向量能在 DEBUG 状态下动态指定则更好。

DEBUG 的 T 命令使用 INT01h，标志 TF=1 时每执行一条指令即产生中断 1、由硬件控制改起来比较麻烦，得到的效果最多也只能和 P 命令一样，所以就不要修改了。

DEBUG 的 P、G 命令使用 INT03h，通过插入一字节的 INT03h 指令设置断点、观察程序的运行，程序运行到断点处后执行 INT03h、INT03h 恢复断点处一字节指令和调试环境、将控制转移到下一语句，显示有关内容。G 命令最多可以设置十个断点，P 命令则执行一步设置一个断点。

INT03h 以外的中断指令有二个字节，所以使用替代中断需要修改四个地方：插入断点中断处、保存断点内容处、恢复断点指令内容处，和设置向量 INT01h、INT03h( P 、G 、T 命令共用部分 )。他们的地址分别在 1039h(P 命令 )、133Dh(G 命令 )、128Fh(INT03h )、1168h—117Bh( PGT ) 。修改如下：

| 命令 | 地址 | 原指令 | 新指令 |
|---|---|---|---|
| P 命令: | 1039 | mov al,es:[di] | mov ax,es:[di] |
| | | mov ds:data_243e,al | mov data_243e,ax |
| | | mov byte ptr es:[di],0CCh | mov word ptr [di],0CDxxh |
| G 命令 | 133D | movsb | movsw |
| | | mov byte ptr [si—1],0CCh | mov word ptr [si—2],0CDxxh |
| | 131C | add di,5 | add di,6 |
| | 1320 | cmp bx,0Bh | cmp bx,06h |
| INT03h: | 128F | movsb | movsw |
| P G T : | 1168 | mov wo ds:int03_ofs, offset Int0x03 | |
| | | | mov wo ds:[xx * 4], offset xx |
| | | mov ds:int03_seg,cs | mov ds:[xx * 4 + 2], cs |
| | 1172 | mov wo ds:int01_ofs, offset Int0x01 | |
| | | | jmp 117C |

```
1178  mov ds:int01 _ seg, cs
117C
```

其中 xx 为替代的空闲中断号、如 60h。

以上的修改可以保证其他地方不要改动，将 G 命令的最大断点数由 10 个减少到了 5 个，如果增加断点数、则需增加缓冲区的长度。

P 、G 、T 命令每次执行时都设置中断 1、3 的向量，使用替代中断后就不能这样作了，必须将 1168 处的 SetVector Int03h 改成 SetVector Intxxh、将 1172 处的 SetVector INT01h 改成空指令。

## 2.25 DEBUG 的命令扩展

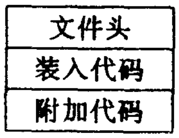
有了源程序，您就可以方便地增加自己需要的命令。当然、您也可以用扩展 EXE 文件的方法作同样的事情。比如，您可以增加这样一些命令：设置替代中断向量、设置 SP、创建文件、输出到文件、关闭文件、允许重定向，驻留内存等。

# 第三章　可执行文件的扩展和加密

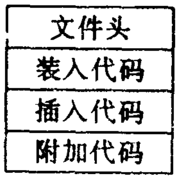
设计、编写软件很费时，也很费脑筋。如果自己辛辛苦苦写出的软件，别人未经许可就能轻易拿去使用，一定是件不痛快的事，相信没人愿意看到。尽管市面上有各种各样的加密软件出售，但用起来往往有不尽人意的地方，并且这些加密软件因为留通广、目标大，一出来往往就遇到克星，因而构造自己的加密程序将是一件非常有益的事情。下面提供一些方法，使您可以给您的软件产品加上 Password、磁盘特征、机器特征、使用限制，为了防止别人跟踪，还可以加上一些反跟踪措施，如修改 Int1 和 Int3 、将 SP 设到死区、运行中生成程序等。

## 3.1　EXE 文件的扩展

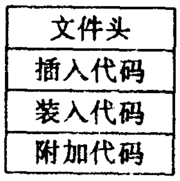
一个规则的 EXE 程序由二部分组成：文件头和装入模块，实际中、大一点的 EXE 程序还可能包含一个附加部分，此部分由开发者用连接程序以外的工具附加到程序末尾、不属于装入模块、也不直接装入内存，仅供程序本身使用。比如：AUTOCAD R11 的 ACAD. EXE，DOS 文件有 1. 77Mb、而装入模块只有 131Kb ，Turbo C＋＋ Ver 2. 00 的 BC. EXE，DOS 文件有 1Mb 多、而装入模块只有 155Kb 。



3－1　EXE 文件构成



3－2　EXE 后端插入扩展



3－3　EXE 前端插入扩展

EXE 文件中的位置信息均为相对位移，DOS 调入 EXE 文件时先用文件头的定位信息进行重定位，然后转移控制、执行。因此我们可以将一段代码插入到 EXE 文件中而照样能运行原文件。根据图 3－1 的情况，有二种插入方法，一是前端插入法、在文件头的后面插入代码，一是后端插入法、在装入代码后插入，分别如图 3－2、3－2 所示。前端插入法须修改文件头中的文件大小和定位信息字节，后端插入法则只需修改文件头中的文件大小二个字。有关步骤分别如下，

前端插入法：

```
1.  copy head code of src. exe to tag. exe
    head _ len = word( ofs 0x02 )
2.  append inserted code to tag. exe, you should make sure
    the inserted _ code' s length is times of 0x10
    suppose the propsed length is l
    l _ seg = l >> 4
3.  append image code of src. exe to tag. exe
4.  append appended code of src. exe to tag. exe
```

```
5.  modify tag.exe
    a. modify size _ double _ word at offset 2 to 5,
       suppose (file _ size / 512) = x..y
       word( ofs 4 ) <== (y ! =0 ) ? x+1 : x
       word( ofs 2 ) <== (y ! =0 ) ? (file _ size-x * 512) : 0
    b. modify ss
       word( ofs 0x0e ) += l _ seg
    c. modify cs
       word( ofs 0x16 ) += l _ seg
    d. modify relocate _ tabel _ items
       suppose
       the relocate tabel items tbl _ items = word( ofs 6 )
       the first _ relocate _ tabel _ items' address
       tbl _ addr = word( ofs 0x18 )
       then loop tbl _ items do
       word( ofs (tbl _ addr+4 * (z-1)+2) ) += l _ seg, z=1..tbl _ items
    e. modify relocate tabel
       loop tbl _ items do
       word( word(ofs(tbl _ addr+4 * (z-1)+2))<<4 +
       word(ofs tbl _ addr+4 * (z-1)) +
       head _ leng )
       += l _ seg, z=1..tbl _ items
```

后端插入法：

```
1.  copy head code of src.exe to tag.exe
    head _ len = word( ofs 0x02 )
2.  append image code of src.exe to tag.exe
3.  append inserted code to tag.exe, you should make sure
4.  append appended code of src.exe to tgt.exe
5.  modify tag.exe' s size _ double _ word at offset 2 to 5
```

如果我们要扩充可执行文件，比如、加口令，就可以此为基础。下面分别讲述 COM 、EXE 文件的扩展和扩充，假设要加入的程序为 RET. EXE 、被扩充的程序为 SRC. EXE 、或 SRC. COM ，最后生成的程序为 TGT. EXE 。

RET. EXE 中含有您的扩充程序，RET 必须完成对 SRC 的搬移、重定位、将控制转移到 SRC 等。

## 3.2 EXE 文件的扩充

DOS 根据文件头装入 EXE 影象代码后还要进行重定位，因此 RET 除了将控制转移到 SRC 外还须完成对 SRC 的重定位。

根据 EXE 文件的构造，可以有二种方式在 EXE 文件中插入 EXE 文件，如图 3－4、3－

5 所示。

| |
|---|
| 文件头 RET |
| 装入代码 RET |
| 文件头 SRC |
| 装入代码 SRC |
| 附加代码 SRC |

3—4 前端扩充

| |
|---|
| 文件头 RET |
| 装入代码 SRC |
| 文件头 SRC |
| 装入代码 RET |
| 附加代码 SRC |

3—5 后端扩充

扩充的方法如下，

前端扩充：

```
1.  copy head _ code of RET. EXE to TGT. EXE
2.  append image _ code of RET. EXD to TGT. EXE
3.  append head _ code of SRC. EXE to TGT. EXE
4.  append image _ code of SRC. EXE to TGT. EXE
5.  append appended _ codes of SRC. EXE to TGT. EXE if necessary
6.  modify size _ double _ word of TGT. EXE
7.  modify the part SRC. EXE in TGT. EXE
    a. SS
    b. CS
    c. reloc _ table _ items
    d. relocate tables
```

后端扩充：

```
1.  copy head code of src. exe to tag. exe
    head _ len = word( ofs 0x02 )
2.  append inserted code to tag. exe, you should make sure
    the inserted _ code' s length is times of 0x10
    suppose the propsed length is l
    l _ seg = l >> 4
3.  append image code of src. exe to tag. exe
4.  modify tag. exe
    a. modify size _ double _ word at offset 2 to 5,
       suppose (file _ size / 512) = x. . y
       word( ofs 4 ) <== (y ! =0 ) ? x+1 : x
       word( ofs 2 ) <== (y ! =0 ) ? (file _ size—x * 512) : 0
    b. modify ss
       word( ofs 0x0e ) += l _ seg
    c. modify cs
       word( ofs 0x16 ) += l _ seg
    d. modify relocate _ tabel _ items
       suppose
    the relocate tabel items tbl _ items = word( ofs 6 )
    the first _ relocate _ tabel _ items' address
    tbl _ addr = word( ofs 0x18 )
    then loop tbl _ items do
    word( ofs (tbl _ addr+4 * (z-1)+2) ) += l _ seg, z=1..tbl _ items
  e. modify relocate tabel
    loop tbl _ items do
    word( word(ofs(tbl _ addr+4 * (z-1)+2))<<4 +
    word(ofs tbl _ addr+4 * (z-1)) +
    head _ leng )
    += l _ seg, z=1..tbl _ items
```

## 3.3 COM 文件的扩充

COM 文件不需重定位,DOS 在 PSP 后装入。装入时,段寄存器指向 PSP 段、SP 设置到程序段末尾、从 CS：100h 处开始执行。因此,最简单的方法是用 3.1 中的前端插入法,将 COM 文件插入到 RET. EXE 的文件头之后,方法如下：

```
1.  copy head _ code of RET. EXE to TGT. EXE
2.  append SRC. COM to TGT. EXE
3.  append Fille _ Bytes to TGT. EXE if necessary
4.  append image _ code of RET. EXE to TGT. EXE
5.  modify TGT. EXE
    a.  size _ double _ word
    b.  SS
    c.  CS
    d.  reloc _ table _ items
    e.  relocate table
```

RET 完成控制转移的代码如下：

```
pop  ax
pop  ax
mov  ah, 62h
int  21h  /* get psp to bx */
cli
mov  ds, bx
mov  es, bx
mov  ss, bx
mov  sp, 0FFFEh
sti
push bx
mov  ax, 100h
```

```
push ax
mov  ax, 0
mov  bx, 0
retf
```

## 3.4 构造自己的加密程序

掌握了上述方法，就可以动手构造自己的加密程序了。可以给您的软件产品加上 Password、不知道口令的人就用不了，可以加上磁盘特征和机器特征、没有一样的硬件环境就不能运行，可以加上使用次数的限制和使用时间的限制，也还可以加上一些反跟踪措施，如修改 Int1 和 Int3 、将 SP 设到死区、运行中生成程序等，也可以组合使用多种方法。您越异想天开、走旁门左道，取得的效果就可能越好。

为了便于理解，附录中给出的例子将扩展程序和种子程序分开给出，实际应用中您可以将它们的目标代码有机结合成一个可执行文件。本书给出的程序均用 Turbo C 2.0 在大模式下调试通过，对于种子文件而言，Turbo C 的初始目标文件 C0L.OBJ 有一些负作用，因而附录中给出了修改的初始目标文件。

# 附录 A.　DOS Ver 3.31 DEBUG 程序清单

```
;uses MASM Ver 5.10 to make exe file
.286c

seg_a   segment para public
assume cs:seg_a, ds:seg_a, ss:stack_seg_d

;function    : Creat PSP
;entry       : [BX] = PSP segment
sub_a_1 proc far
          mov ah,26h
          int 21h
          retf
sub_a_1 endp
db 11 dup (0)
seg_a   ends

seg_b   segment para public
assume cs:seg_b, ds:seg_b, ss:stack_seg_d
data_82   dw 1       ;stdout
data_83   db 0
data_84   db 0
data_85   db 0
data_86   db 0
data_87   db 0
data_88   dw 0
data_89   dw 0
data_90   db 20h     ;space
db ' 0123456789ABCDEFabcdef'
data_91   dw 0
data_92   dw 0
db 21 dup (0)
db 15h, 0

;function: fprintf(stdout, ...)
sub_1   proc far
          push bp
```

25-70

```
        push dx
        push cx
        push bx
        push ax
        push di
        push si
        push es
        push ds
        mov bp,sp
        push cs
        pop es
        mov di,26h
        mov bp,word ptr [bp+16h]
        mov si,ds: [bp]
        xor bx,bx       ; Zero register
        call sub _ 8    ; (0253),初始化数据
loc _ 1:
        lodsb           ; String [si] to al
        cmp al,25h      ; ' %'
        je loc _ 4
        or al,al                          ; Zero ?
        jz loc _ 2                        ; Jump if zero, ' \0'
        call sub _ 5                      ; (0222),fill buffer,if full then flush
        jmp short loc _ 1                 ; (0059)
loc _ 2:
        call sub _ 7                      ; (0243),flush buffer
        pop ds
        pop es
        pop si
        pop di
        pop ax
        pop bx
        pop cx
        pop dx
        pop bp
        pop word ptr cs:data _ 91         ; (66E9:0022=0)
        pop cs:data _ 92                  ; (66E9:0024=0)
        pop ax
        push cs:data _ 92                 ; (66E9:0024=0)
        push word ptr cs:data _ 91        ; (66E9:0022=0)
        retf
loc _ 3:
        call sub _ 5                    ; (0222)
        jmp short loc _ 1               ; (0059)
oc _ 4:
        lodsb                           ; String [si] to al
        cmp al,25h                      ; '%'
        je loc _ 3                      ; Jump if equal
        cmp al,2Dh                      ; '—'
        je loc _ 6                      ; Jump if equal
        cmp al,2Bh                      ; '+'
        je loc _ 8                      : Jump if equal
        cmp al,4Ch                      ; 'L'
        je loc _ 7                      ; Jump if equal
        cmp al,6Ch                      ; 'l'
        je loc _ 7                      ; Jump if equal
        cmp al,30h                      ; '0'
        jb loc _ 9                      ; Jump if below
        cmp al,39h                      ; '9'
        ja loc _ 9                      ; Jump if above
        cmp al,30h                      ; '0'
        jne loc _ 5                     ; Jump if not equal
        cmp cs:data _ 88,0              ; (66E9:0007=0)
        jne loc _ 5                     ; Jump if not equal
        mov cs:data _ 90,30h            ; (66E9:000B=20h) '0'
 _ 5:
        push ax
        mov ax,0Ah
        mul cs:data _ 88                ; (66E9:0007=0) ax = data * ax
        mov cs:data _ 88,ax             ; (66E9:0007=0)
        pop ax
        xor ah,ah                       ; Zero register
        sub al,30h                      ; '0'
        add cs:data _ 88,ax             ; (66E9:0007=0)
        jmp short loc _ 8               ; (00E2)
 6:
        inc byte ptr cs:data _ 83       ; (66E9:0002=0)
        jmp short loc _ 8               ; (00E2)
 7:
        inc cs:data _ 84                ; (66E9:0003=0)
```

```
loc_8:
        jmp short loc_4                 ; (008E)
loc_9:
        cmp al,58h                      ; 'X'
        je loc_12                       ; Jump if equal
        cmp al,61h                      ; 'a'
        jb loc_10                       ; Jump if below
        cmp al,7Ah                      ; 'z'
        jg loc_10                       ; Jump if >
        and al,0DFh
loc_10:
        cmp al,58h                      ; 'X'
        je loc_11                       ; Jump if equal
        cmp al,44h                      ; 'D'
        je loc_13                       ; Jump if equal
        cmp al,43h                      ; 'C'
        je loc_15                       ; Jump if equal
        cmp al,53h                      ; 'S'
        je loc_14                       ; Jump if equal
        call sub_8                      ; (0253)
        jmp loc_1                       ; (0059)
loc_11:
        mov cs:data_86,6                ; (66E9:0005=0)
loc_12:
        mov cs:data_89,10h              ; (66E9:0009=0)
        jmp short loc_24                ; (0192)
        nop
loc_13:
        mov cs:data_89,0Ah              ; (66E9:0009=0)
        jmp short loc_24                ; (0192)
        nop
loc_14:
        inc cs:data_87                  ; (66E9:0006=0)
loc_15:
        push si
        mov si,bx
        add bx,2
        mov si,ds:[bp+si+2]
        cmp cs:data_87,0                ; (66E9:0006=0)
        jne loc_16
```

```
            lodsb
            cmp al,0
            je loc _ 20
            call sub _ 5                  ; (0222)
            jmp short loc _ 20            ; (0179)
loc _ 16:
            mov cx,cs:data _ 88           ; (66E9:0007=0)
            or cx,cx
            jz loc _ 17
            cmp byte ptr cs:data _ 83,0   ; (66E9:0002=0)
            jne loc _ 17
            push si
            call sub _ 2                  ; (0180)
            pop si
loc _ 17:
            push si
loc _ 18:
            lodsb
            cmp al,0
            je loc _ 19
            call sub _ 5                  ; (0222)
            jmp short loc _ 18            ; (015A)
loc _ 19:
            pop si
            cmp byte ptr cs:data _ 83,0   ; (66E9:0002=0)
            je loc _ 20                   ; Jump if equal
            mov cx,cs:data _ 88           ; (66E9:0007=0)
            or cx,cx                      ; Zero ?
            jz loc _ 20                   ; Jump if zero
            call sub _ 2                  ; (0180)
loc _ 20:
            call sub _ 8                  ; (0253)
            pop si
            jmp loc _ 1                   ; (0059)

sub _ 2:                                  ;         Called from:  66E9:0155, 0176
            xor dx,dx                     ; Zero register
loc _ 21:
            lodsb                         ; String [si] to al
            or al,al                      ; Zero ?
```

```
        jz loc _ 22                     ; Jump if zero
        inc dx
        jmp short loc _ 21              ; (0182)
loc _ 22:
        sub cx,dx
        jbe loc _ ret _ 23              ; Jump if below or =
        call sub _ 4                    ; (0212)
loc _ ret _ 23:
        retn
loc _ 24:
        push si
        mov si,bx
        add bx,2
        mov ax,ds: [bp+si+2]
        cmp cs:data _ 84,0              ; (66E9:0003=0)
        je loc _ 25                     ; Jump if equal
        mov si,bx
        add bx,2
        mov dx,ds: [bp+si+2]
        jmp short loc _ 26              ; (01B1)
loc _ 25:
        xor dx,dx                       ; Zero register
loc _ 26:
        push bx
        mov si,cs:data _ 89             ; (66E9:0009=0)
        mov cx,cs:data _ 88             ; (66E9:0007=0)
        call sub _ 3                    ; (01CA)
        call sub _ 4                    ; (0212)
        call sub _ 8                    ; (0253)
        pop bx
        pop si
        jmp loc _ 1                     ; (0059)
        sub _ 1   endp

        sub _ 3   proc near             ;  Called from:  66E9:01BC, 01E0
        dec cx
        push ax
        mov ax,dx
        xor dx,dx                       ; Zero register
        div si                          ; ax,dx rem=dx:ax/reg
```

```
                mov bx,ax
                pop ax
                div si                          ; ax,dx rem=dx:ax/reg
                xchg bx,dx
                push ax
                or ax,dx
                pop ax
                jz loc _ 27                     ; Jump if zero
                push bx
                call sub _ 3                    ; (01CA)
                pop bx
                jmp short loc _ 28              ; (01F1)
loc _ 27:
                cmp byte ptr cs:data _ 83,0     ; (66E9:0002=0)
                jne loc _ 28                    ; Jump if not equal
                call sub _ 4                    ; (0212)
loc _ 28:
                mov ax,bx
                cmp al,0Ah
                jb loc _ 29                     ; Jump if below
                cmp cs:data _ 85,0              ; (66E9:0004=0)
                jne loc _ 29                    ; Jump if not equal
                add al,cs:data _ 86             ; (66E9:0005=0)
loc _ 29:
                mov bx,0Ch
                push ds
                push cs
                pop ds
                xlat                            ; al= [al+ [bx] ] table
                pop ds
                push cx
                call sub _ 5                    ; (0222)
                pop cx
                retn
                sub _ 3   endp

                sub _ 4   proc near
                or cx,cx                        ; Zero ?
                jle loc _ ret _ 31              ; Jump if < or =
                mov al,cs:data _ 90             ; (66E9:000B=20h)
```

```
locloop _ 30:
          push cx
          call sub _ 5                    ; (0222)
          pop cx
          loop locloop _ 30               ; Loop if cx > 0
loc _ ret _ 31:
          retn
          sub _ 4   endp

          sub _ 5   proc near
          stosb                           ; Store al to es: [di]
          cmp di,word ptr 003Ah
          je loc _ 33                     ; Jump if equal
loc _ ret _ 32:
          retn
loc _ 33:
          mov cx,14h
          sub _ 5   endp

          sub _ 6   proc near             ;          Called from:   66E9:024F
          push bx
          mov bx,cs:data _ 82             ; (66E9:0000=1)
          push ds
          push cs
          pop ds
          mov dx,26h
          mov ah,40h                      ; '@'
          int 21h   ; DOS Services  ah=function 40h, write file cx=bytes, to ds:dx
          pop ds
          pop bx
          mov di,26h
          retn
          sub _ 6   endp
          sub _ 7   proc near             ;          Called from:   66E9:0067
          cmp di,word ptr 0026h
          je loc _ ret _ 32               ; Jump if equal
          sub di,word ptr 26h
          mov cx,di
          call sub _ 6                    ; (022D)
          retn
```

```
        sub _ 7   endp

        sub _ 8   proc near
        xor ax,ax                       ; Zero register
        mov byte ptr cs:data _ 83,al    ; (66E9:0002=0)
        mov cs:data _ 84,al             ; (66E9:0003=0)
        mov cs:data _ 86,al             ; (66E9:0005=0)
        mov cs:data _ 88,ax             ; (66E9:0007=0)
        mov cs:data _ 90,20h            ; (66E9:000B=20h)
        mov cs:data _ 87,al             ; (66E9:0006=0)
        retn
        sub _ 8   endp
seg _ b   ends

seg _ c   segment para public
assume cs:seg _ c, ds:seg _ c, ss:stack _ seg _ d
dw 0
data _ 94   dw 0
db 252 dup (0)

                                        ;   Program Entry Point
db24   proc far
start:
        jmp short loc _ 35              ; (010B)
db ' Vers 2. 40'
loc _ 35:
        mov ah,30h                      ; ' 0'
        int 21h      ; DOS Services   ah=function 30h, get DOS version number ax
        cmp ax,1F03h
        je loc _ 36                     ; Jump if ver is 3. 31
        mov dx,offset ver _ err _ str   ;321Dh
        push cs
        pop ds
        mov ah,9
        int 21h                         ; display ' Incorrect DOS version'
        push es
        xor ax,ax
        push ax
        retf                            ; Return to Caller
loc _ 36:
```

```
mov ax,3503h
int 21h                        ; save vector of int03h
mov cs:data _ 207e,bx          ; (6710:3559=0)
mov cs:data _ 208e,es          ; (6710:355B=0)
mov ax,3501h
int 21h                        ;save vector of int01h
mov cs:data _ 205e,bx          ; (6710:3555=0)
mov cs:data _ 206e,es          ; (6710:3557=0)
mov cs:data _ 226e,ds          ; (6710:359C=0)
push cs
pop es
xor si,si                      ; Zero register
xor di,di                      ; Zero register
mov cx,100h
rep movsb                      ; copy DEBUG _ PSP to cs:0
push cs
pop ds
call sub _ 135                 ; (23BF), return only
mov ah,51h                     ; 'Q'
int 21h      ; DOS Services  ah=function 51h, get active PSP segment in bx
mov data _ 186,bx              ; (6710:31F5=0)
mov ds:data _ 215e,al          ; (6710:357F=0)
mov ax,cs
mov ds,ax
mov es,ax
call sub _ 9                   ; (0294), set program end address
push es
mov      ax,3524h
int      21h
mov      ds:[354Bh],bx       ;save vector of int24h
mov      ds:[354Dh],es
pop      es
mov      ax,2524h
mov      dx,offset int24h    ;                               02BCh

int      21h                 ;set vector int24h
mov      al,23h
mov      dx,032Fh
int      21h                 ;set vector int23h
mov      dx,cs
mov     ax,37B1h
mov     cl,04h
shr     ax,cl
add     dx,ax
mov     ax,cs
sub     ax,ds:[359Ch]
add     dx,ax
call    dword ptr  ds:[359Ah]     ;call sub_a_1, creat sub_PSP after DEBUG
mov     ax,dx
mov     di,3188h
cld
stosw                          ;save sub_PSP
stosw
stosw
stosw
mov     ds:[3586h],ax          ;set default values
mov     ds:[3582h],ax
mov     ds:[3590h],ax
mov     ax,0100h
mov     ds:[3584h],ax
mov     ds:[3580h],ax
mov     ds:[358Eh],ax
mov     ds,dx
mov     es,dx
mov     dx,0080h
mov     ah,1Ah
int     21h                    ;set disk buffer to ds:dx
mov     ax,ds:[0006]
mov     bx,ax
add     ax,0100h
push    cs
pop     ds
push    bx
dec     ax
dec     ax
mov     bx,ax
mov     word ptr es:[bx],0000
pop     bx
mov     ds:[3180h],ax
dec     ah
```

```
          mov     es: [0006] ,ax
          sub     bx,ax
          mov     cl,04
          shr     bx,cl
          add     es: [0008] ,bx
          mov     ah,0fh
          int     10h                    ;get CRT enviroment params
          cmp     ah,28h
          jnz     loc _ 37               ;020ah
          mov     byte ptr ds: [3198h] ,07
          mov     byte ptr ds: [3199h] ,04
          mov     word ptr ds: [319Ah] ,0040h
loc _ 37:
          mov di,5Ch
          mov si,81h
          mov ax,2901h
          int 21h      ; DOS Services   ah=function 29h, parse filenam @ds:si FCBes:di
          call sub _ 81                  ; find delimeter
          call sub _ 56                  ; copy command _ line, analysis filename
          push cs
          pop es
          mov di,80h
          cmp byte ptr es: [di] ,0
          je loc _ 39                    ; Jump if no command _ line _ params
loc _ 38:
          inc di
          cmp byte ptr es: [di] ,0Dh
          je loc _ 39                    ; Jump if equal
          cmp byte ptr es: [di] ,20h     ; ' '
          je loc _ 38                    ; Jump if equal
          cmp byte ptr es: [di] ,9
          je loc _ 38                    ; Jump if equal
          or data _ 177,1                ; (6710:3195=0), set params _ flag
          call sub _ 69                  ; (0BFC), load file
          push cs
          pop ds
          mov ax,cs _ save               ; (6710:318E=0), set default seg, ofs
          mov ds:data _ 219e,ax          ; (6710:3586=0)
          mov ds:data _ 217e,ax          ; (6710:3582=0)
          mov ax,ip _ save               ; (6710:3190=100h)
          mov ds:data _ 218e,ax          ; (6710:3584=0)
          mov ds:data _ 216e,ax          ; (6710:3580=0)

loc _ 39:                                ;command _ accept
          cld                            ; Clear direction
          mov ax,cs
          mov ds,ax
          mov es,ax
          cli                            ; Disable interrupts
          mov ss,ax                      ;set SS, SP
          mov sp,3178h
          sti                            ; Enable interrupts
          cmp byte ptr ds:data _ 209e,0
          je loc _ 40
          mov byte ptr ds:data _ 209e,0
loc _ 40:
          mov dx,33BCh
          call sub _ 20                  ; (03DA), fprintf(stdout, "—")
          call sub _ 11                  ; (0340), get KBDline, Upper to cs: [35B8h]
          call sub _ 14                  ; (0388), skip pre _ space char
          jz loc _ 39                    ; Jump if no input _ data
          lodsb                          ; String [si] to al
          sub al,41h                     ; 'A'
          jc loc _ 41                    ; error if less then 'A'
          cmp al,19h
          ja loc _ 41                    ; error if great then 'Z'
          shl al,1                       ; Shift w/zeros fill
          cbw                            ; Convrt byte to word
          xchg ax,bx
          call cs:Cmd _ TBL [bx]         ; 26 entries, dispatch
          jmp short loc _ 39             ; (0255), goto command _ accept procedure
loc _ 41:
          jmp loc _ 97                   ; (06B5), illegal command
db24   endp

;set vector of int22h
sub _ 9   proc near
          push ds
          push cs
          pop ds
          mov ax,2522h
          mov dx,2F4h
          int 21h    ; DOS Services   ah=function 25h, set intrpt vector al to ds:dx
          pop ds
          retn
sub_9   endp

;restore vectors of int01h, int03h
sub_10   proc near
          push ds
          push dx
          push ax
          lds dx,dword ptr cs:data_207e ; (6710:3559=0) Load 32 bit ptr
          mov ax,2503h
          int 21h    ; DOS Services   ah=function 25h, set intrpt vector al to ds:dx
          lds dx,dword ptr cs:data_205e ; (6710:3555=0) Load 32 bit ptr
          mov ax,2501h
          int 21h    ; DOS Services   ah=function 25h, set intrpt vector al to ds:dx
          pop ax
          pop dx
          pop ds
          retn
          sub_10   endp

int24h:
          test byte ptr cs:data_201e,0FFh ; (6710:354F=0)
          jz loc_42                    ; Jump if zero
          mov al,0
          iret                         ; Interrupt return
loc_42:
          pushf                        ; Push flags
          call dword ptr cs:data_200e  ; (6710:354B=0), call old int24h
          cmp al,2
          jne loc_ret_43               ; Jump if not equal
          push ax
          push bx
          mov ah,51h                   ; 'Q'
          int 21h    ; DOS Services  ah=function 51h,get active PSP segment in bx
          cmp bx,cs:data_226e          ; (6710:359C=0)
          pop bx
          pop ax
          jz loc _ 44                  ; Jump if zero
loc _ ret _ 43:
          iret                         ; Interrupt return
loc _ 44:
          mov byte ptr cs:data _ 201e,0FFh    ; (6710:354F=0)
          mov ah,0Dh
          int 21h    ; DOS Services   ah=function 0Dh, flush disk buffers to disk
          mov byte ptr cs:data _ 201e,0  ; (6710:354F=0)
          jmp loc _ 39                 ; (0255), goto command _ accept status

int _ 22h _ entry proc far
          cmp cs:data _ 183,0          ; (6710:31F1=0)
          jne loc _ 46                 ; end
          mov ax,cs:data _ 226e        ; (6710:359C=0)
          mov cs:data _ 186,ax         ; (6710:31F5=0)
          cmp cs:data _ 184,0          ; (6710:31F2=0)
          je loc _ 45                  ; terminated normally
          mov ax,cs
          mov ds,ax
          cli                          ; Disable interrupts
          mov ss,ax
          mov sp,3178h
          sti                          ; Enable interrupts
          mov ax,data _ 189            ; (6710:31FB=0)
          jmp loc _ 191                ; (0DE0)
loc _ 45:
          push cs
          pop ds
          mov dx,3251h     ;Program terminated normally, goto command _ accept status
          jmp short loc _ 47           ; (032C)
loc _ 46:
          call sub _ 10                ; (02A1), restore int01h, int03h
          mov ax,4C00h
          int 21h    ; DOS Services   ah=function 4Ch, terminate with al=return code
loc _ 47:
          call sub _ 20                ; (03DA)

Int0x23:
          mov ax,cs
          mov ds,ax
          cli                        ; Disable interrupts
          mov ss,ax
          mov sp,3178h
          sti                        ; Enable interrupts
          call sub_22                ; (03E7)
          jmp loc_39                 ; (0255), goto command_accept
int_22h_entry endp

;get converted KBDline to cs: [35B8h]
sub_11    proc near                  ; Called from:  6710:0275, 0979, 09A4, 13FD
          call sub_25                ; (040A)
          mov si,319Eh
          mov di,35B8h
loc_48:
          lodsb                      ; String [si] to al
          cmp al,61h                 ; 'a'
          jb loc_49                  ; Jump if below
          cmp al,7Ah                 ; 'z'
          ja loc_49                  ; Jump if above
          add al,0E0h                ;Upper
loc_49:
          stosb                      ; Store al to es: [di]
          cmp al,0Dh
          je loc_52
          cmp al,22h                 ; '"'
          je loc_50
          cmp al,27h                 ; '''
          jne loc_48
loc_50:
          mov ah,al
loc_51:                              ;引号中的内容不需转换
          lodsb                      ; String [si] to al
          stosb                      ; Store al to es: [di]
          cmp al,0Dh
          je loc_52
          cmp al,ah
          jne loc_51
          jmp short loc_48           ; (0349)
loc_52:
```

```
                mov si,35B8h                    ;转换后的始址
                call sub_22                     ; (03E7), fprintf(stdout, "\n")
                retn
                sub_11   endp


                                                ;fprintf(stdout, "\0x20\0x08")
                sub_12   proc near
                push dx
                mov dx,321Bh
                call sub_20                     ; (03DA)
                pop dx
                retn
                sub_12   endp

                ;去掉前导空白和',' ,ZF==1 if 除回车外无输入
                sub_13   proc near
                call sub_14                     ; (0388)
                cmp byte ptr [si] ,2Ch          ; ','
                jne loc_55                      ; Jump if not equal
                inc si

                ;skip space & TAB, ZF==1 if no other chars except RET
                sub_14:
loc_53:
                push ax
loc_54:
                lodsb                           ; String [si] to al
                cmp al,20h                      ; ' '
                je loc_54                       ; Jump if equal
                cmp al,9
                je loc_54                       ; Jump if equal
                dec si
                pop ax
loc_55:
                cmp byte ptr [si] ,0Dh
                retn
                sub_13   endp

;command H
;function : hexadd
```

35 8-10

```
;format : h <v1> <v2>
;output : add _ v sub _ v
        sub _ 15   proc near
        mov cx,4
        call sub _ 36              ; (0643), xatoi(), len=4, get v1
        mov di,dx
        mov cx,4
        call sub _ 36              ; (0643), get v2
        call sub _ 42              ; (06AF), 处理非法输入
        push dx
        add dx,di                  ;和
        mov ds:data _ 265e,dx      ; (6710:3884=0)
        pop dx
        sub di,dx                  ;差
        mov ds:data _ 266e,di      ; (6710:3886=0)
        mov dx,3882h
        call sub _ 21     ; (03E1), fprintf(stdout, "%04X  %04X", 和, 差)
        retn
        sub _ 15   endp

;save ds, si for print
        sub _ 16   proc near
        mov cs:data _ 259e,ds      ; (6710:3863=0)
        mov cs:data _ 260e,si      ; (6710:3865=0)
        retn
        sub _ 16   endp

;save es, di for print
        sub _ 17   proc near
        mov ds:data _ 75e,es       ; (66E8:3863=0)
        mov ds:data _ 76e,di       ; (66E8:3865=0)
        retn
        sub _ 17   endp

;fprintf(stdout, "%04X:%04X", ...)
        sub _ 18   proc near
        mov byte ptr ds:data _ 74e,0   ; (66E8:37A2=0)

;fprintf(stdout, "%04X:%04X %s", ...)
        sub _ 19:
```

3AR-70

```
            mov dx,3861h

    ;fprintf(stdout, ... )
            sub _ 20:
            push dx
            call far ptr sub _ 1            ; (66E9:003D)
            retn
            sub _ 18    endp

    ;fprintf(stdout, ... , "\n")
            sub _ 21    proc near
    loc _ 56:
            push dx
            call far ptr sub _ 1            ; (66E9:003D)

    ;fprintf(stdout, "\n")
            sub _ 22:
    loc _ 57:
            mov dx,3216h
            push dx
            call far ptr sub _ 1            ; (66E9:003D)
            retn
            sub _ 21    endp

    ;es: [di] <== XITOA( [al] )
            sub _ 23    proc near
            mov ah,al
            push cx
            mov cl,4
            shr al,cl                       ; Shift w/zeros fill
            pop cx
            call sub _ 24                    (03FE)
            mov al,ah
    sub _ 24:
            and al,0Fh
            add al,90h
            daa                             ; Decimal adjust
            adc al,40h
            daa                             ; Decimal adjust
            and al,7Fh
```

```
        stosb                           ; Store al to es: [di]
        retn
        sub _ 23   endp

;get KBDline
        sub _ 25   proc near
        push ax
        push dx
        mov ah,0Ah
        mov dx,319Ch
        int 21h                         ; DOS Services   ah=function 0Ah
             ;  get keybd line, put at ds:dx
        pop dx
        pop ax
        retn
        sub _ 25   endp

;fill es: [di] with space
        sub _ 26   proc near
        mov al,20h                      ;' '
        stosb                           ; Store al to es: [di]
        retn
        sub _ 26   endp

;fill space from es: [di] to es: [di+cx] , di <== di+cx
        sub _ 27   proc near
locloop _ 58:
        jcxz loc _ ret _ 59             ; Jump if cx=0
        call sub _ 26                   ; (0416)
        loop locloop _ 58               ; Loop if cx > 0
loc _ ret _ 59:
        retn
        sub _ 27   endp

Cmd _ TBL   dw offset sub _ cmd _ A     ;13CDh ;422h,' A'
        dw offset sub _ 152             ;' B'
        dw offset sub _ cmd _ C         ;' C'
        dw offset sub _ cmd _ D         ;' D'
        dw offset sub _ 46              ;' E'
        dw offset sub _ cmd _ F         ;' F'
```

```
          dw offset sub _ 80              ;' G'
          dw offset sub _ 15              ;' H'
          dw offset sub _ 77              ;' I'
          dw offset sub _ 152             ;' J'
          dw offset sub _ 152             ;' K'
          dw offset sub _ cmd _ L         ;' L'
          dw offset sub _ 32              ;' M'
          dw offset sub _ 66              ;' N'
          dw offset sub _ 78              ;' O'
          dw offset sub _ 75              ;' P'
          dw offset sub _ cmd _ Q         ;' Q'
          dw offset sub _ 54              ;' R'
          dw offset sub _ 33              ;' S'
          dw offset sub _ 76              ;' T'
          dw offset sub _ cmd _ U         ;' U'
          dw offset sub _ 152             ;' V'
          dw offset sub _ cmd _ W         ;' W'
          dw offset sub _ 152             ;' X'
          dw offset sub _ 152             ;' Y'
          dw offset sub _ 152             ;' Z'
sub _ cmd _ Q:
          inc byte ptr ds:data _ 183      ; [31F1h]
          mov bx,ds:data _ 186            ; [31F5h]
loc _ 61:
          mov ah,50h                      ; ' P'
          int 21h    ; DOS Services   ah=function 50h, set active PSP segmnt from bx
          call sub _ 136                  ; (23C0), return only
          call sub _ 10                   ; (02A1), restore vector int01, int03
          mov ax,4C00h
          int 21h    ; DOS Services   ah=function 4Ch,terminate with al=return code


;get range
;exit : ax=seg, dx=ofs, cx=len
          sub _ 28    proc near
          mov bp,ds _ save                ; (6710:3188=0)
          mov word ptr ds:data _ 225e,80h (6710:3592=0)


          sub _ 29:
          call sub _ 44                   ; (06CE), get seg, ofs
          push ax
          push dx
          call sub _ 13                 ; (037F), skip space & TAB & ' ,'
          mov al, [si]
          cmp al,4Ch                    ; 'L' , length token char
          je loc _ 65                   ; 直接取 len
          mov dx,ds:data _ 225e         ; (6710:3592=0)
          call sub _ 38                 ; (0655)
          jc loc _ 64                   ; Jump if carry Set, 遇非十六进制字符
          mov cx,4
          call sub _ 36                 ; (0643),xatoi(),
          mov cx,dx
          pop dx
          sub cx,dx
          jnc loc _ 63                  ; Jump if carry=0
loc _ 62:
          jmp loc _ 98       ; (06B6), illegal if target _ addr less then source _ addr
loc _ 63:
          inc cx
          pop ax
          retn
loc _ 64:
          pop cx
          push cx
          neg cx
          jz loc _ 66                   ; Jump if zero
          cmp cx,dx
          jae loc _ 66                  ; Jump if above or =
          jmp short loc _ 67            ; (04B7)
          nop
loc _ 65:
          inc si
          mov cx,4
          call sub _ 36                 ; (0643), get len
loc _ 66:
          mov cx,dx
loc _ 67:
          pop dx
          mov ax,cx
          add ax,dx
          jnc loc _ 68                  ; Jump if carry=0
```

```
        cmp ax,1
        jae loc _ 62                ; illegal if length over current segment
loc _ 68:
        pop ax
        retn

;get range
;exit : ax=seg, dx=ofs, cx=len
        sub _ 30:
        call sub _ 13               ; (037F)
        jz loc _ 69                 ; no input then get default values
        mov ds:data _ 225e,cx       ; (6710:3592=0)
        call sub _ 29               ; (0477)
        jmp loc _ 96                ; (06AF), 处理非法输入
loc _ 69:
        mov si,di
        lodsw                       ; String [si] to ax
        mov dx,ax
        lodsw                       ; String [si] to ax
        retn

;command D
;function : dump memory
;format : d [<range>]
;exit    : seg:ofs [...] byte _ v ... [- byte _ v ...]    chars...
sub _ cmd _ D:
        mov bp,ds _ save            ; (6710:3188=0)
        mov cx,disp _ chars         ; (6710:319A=80h)
        mov di,358Eh
        call sub _ 30               ; (04C5), get range
        mov ds,ax
        mov si,dx
        push si
        mov al,ss:[3198h]           ;dispc _ per _ line , (6710:3198=0Fh)
        xor ah,ah                   ; Zero register
        xor ax,0FFFFh
        and si,ax
        mov di,37A2h
        call sub _ 16               ; (03BE), save ds, si for print
        pop si
        mov ax,si
        mov ah,3
        and al,ss:[3198h]              ;dispc _ per _ line    ; (6710:3198=0Fh)
        mul ah                         ; ax = reg * al, 调整边界
        or al,al                       ; Zero ?
        jz loc _ 70                    ; Jump if zero
        push cx
        mov cx,ax
        call sub _ 27                  ; (041A),设置边界空白
        pop cx
loc _ 70:
        push si
loc _ 71:
        call sub _ 26                  ; (0416),es:[di] <== itoa( [al] )
loc _ 72:
        lodsb                          ; String [si] to al
        call sub _ 23                  ; (03F1)
        pop dx
        dec cx
        jz loc _ 74                    ; 显示其余部分
        mov ax,si
        test al,ss:[3198h]             ;dispc _ per _ line    ; (6710:3198=0Fh)
        jz loc _ 73                    ; Jump if 转换了一行
        push dx
        test al,7
        jnz loc _ 71                   ; Jump if not zero
        mov al,2Dh                     ; '—',已排七个字节则插入'—'
        stosb                          ; Store al to es:[di]
        jmp short loc _ 72             ; (051A)
loc _ 73:
        call sub _ 31                  ; (0540), 显示一行
        mov di,37A2h
        call sub _ 16                  ; (03BE)
        jmp short loc _ 70             ; (0516)

;display ds:si hex _ v... char _ v...
        sub _ 31:
loc _ 74:
        push cx
        mov ax,si
```

```
            dec al
            and al,ss:[3198h]           ;dispc_per_line   ;(6710:3198=0Fh)
            inc al
            sub al,10h
            dec al
            neg al
            cbw                         ; Convrt byte to word
            mov cx,ax
            shl ax,1                    ; Shift w/zeros fill
            add cx,ax
            mov ax,dx
            and al,ss:[3198h]           ;dispc_per_line   ;(6710:3198=0Fh)
            xor ah,ah                   ; Zero register
            add cx,ax
            call sub_27                 ; (041A),调整边界
            mov cx,si
            mov si,dx
            sub cx,si
locloop_75:
            lodsb                       ; String [si] to al
            cmp al,7Fh
            jae loc_76                  ; mask extended ascii chars
            cmp al,20h                  ; ' '
            jae loc_77                  ; Jump if above or =
loc_76:
            mov al,2EH                  ; '.', if ! isprint( [al] ) then print '.'
loc_77:
            stosb                       ; Store al to es:[di]
            loop locloop_75             ; Loop if cx > 0
            mov al,0
            stosb                       ; Store al to es:[di]
            push ds
            push cs
            pop ds
            call sub_19     ; (03D7). fprintf(stdout, "%04X:%04X%s", ...)
            call sub_22                 ; (03E7), fprintf(stdout, "\n")
            pop ds
            pop cx
            mov ss:data_223e,si         ; (6710:358E=0). set next address
            mov ss:data_224e,ds         ; (6710:3590=0)
```

```
        retn

;command M
;function  : move memory
;format  : M <range> <address>
        sub _ 32:
        call sub _ 28                   ; (046D), get range
        push cx
        push ax
        push dx
        call sub _ 44                   ; (06CE), get address
        call sub _ 42                   ; (06AF), 处理非法输入
        pop si
        mov di,dx
        pop bx
        mov ds,bx
        mov es,ax
        pop cx
        cmp di,si
        sbb ax,bx
        jc loc _ 78                     ; Jump if carry Set
        dec cx
        add si,cx
        add di,cx
        std      ; Set direction flag,如果目标地址大于源地址则置反向传送标志
        inc cx
loc _ 78:
        movsb                           ; Mov [si] to es: [di]
        dec cx
        rep movsb                       ; Rep while cx>0 Mov [si] to es: [di]
loc _ ret _ 79:
        retn

;command F
;function  : fill memory
;format  : F <range> <list>
sub _ cmd _ F:
        call sub _ 28                   ; (046D), get range
        push cx
        push ax
```

```
		push dx
		call sub _ 41		; (06A1), get list
		pop di
		pop es
		pop cx
		cmp bx,cx
		mov si,35B8h
		jcxz loc _ 80		; Jump if cx=0
		jnc loc _ 78		; Jump if len _ list > len _ range
loc _ 80:
		sub cx,bx		; cx <== len -n
		xchg cx,bx		; bx <== len-n, cx <== n
		push di
		rep movsb	; Rep while cx>0 Mov [si] to es: [di] , fill the 1st block
		pop si
		mov cx,bx
		push es
		pop ds
		jmp short loc _ 78		; (05B6), fill the others

;command S
;function : search
;format : S <range> <list>
;exit : display matched address
sub _ 33:
		call sub _ 28		; (046D), get range
		push cx
		push ax
		push dx
		call sub _ 41		; (06A1), get list
		dec bx
		pop di
		pop es
		pop cx ,
		sub cx,bx
loc _ 81:
		mov si,35B8h
		lodsb			; String [si] to al
locloop _ 82:
		scasb			; Scan es: [di] for al, search
```

```
        loopnz locloop _ 82             ; Loop if zf=0, cx>0
        jnz loc _ ret _ 79              ; Jump if not zero, 无匹配,返回
        push bx
        xchg bx,cx
        push di
        repe cmpsb
              ; Rept zf=1+cx>0 Cmp [si] to es: [di] , compare the others bytes
        mov cx,bx
        pop di
        pop bx
        jnz loc _ 83                    ; Jump if not zero, 无匹配
        dec di                          ; 匹配地址
        call sub _ 17                   ; (03C9)
        inc di
        call sub _ 18                   ; (03D2), fprintf(stdout, "%04X:%04X  ", ...)
        call sub _ 22                   ; (03E7), fprintf(stdout, "\n")
loc _ 83:
        jcxz loc _ ret _ 79             ; Jump if cx=0
        jmp short loc _ 81              ; (05ED), search again

;dx <== xatoi( [si] .. [si+cx] )
        sub _ 34:
        mov word ptr ds:data _ 261e,0 ; (6710:3869=0)
        call sub _ 13                   ; (037F)

;dx <== xatoi( [si] .. [si+cx] )
        sub _ 35:
        xor dx,dx                       ; Zero register
        call sub _ 38                   ; (0655)
        jc loc _ ret _ 88               ; Jump if carry Set
        mov dl,al
loc _ 84:
        inc si
        dec cx
        call sub _ 38                   ; (0655)
        jc loc _ 87                     ; Jump if carry Set
        stc                             ; Set carry flag
        jcxz loc _ ret _ 88             ; Jump if cx=0
        push cx
        mov cx,4
```

```
locloop_85:
          shl word ptr ds:data_261e,1      ; (6710:3869=0) Shift w/zeros fill
          shl dx,1                         ; Shift w/zeros fill
          adc word ptr ds:data_261e,0      ; (6710:3869=0)
          loop locloop_85                  ; Loop if cx > 0
          pop cx
          or dl,al
          jmp short loc_84                 ; (0623)

;dx <== xatoi( [si] .. [si+cx] )
          sub_36:
          mov word ptr ds:data_261e,0      ; (6710:3869=0)
          call sub_34                      ; (0611)
          jmp short loc_86                 ; (0651)

;dx <== xatoi( [si] .. [si+cx] )
          sub_37:
          call sub_35                      ; (061A)
loc_86:
          jc loc_98                        ; Jump if carry Set
loc_87:
          clc                              ; Clear carry flag
loc_ret_88:
          retn

;cf <== 0 , [al] <== xatoi( [si] ) if isxdigit( [si] )
          sub_38:
          mov al, [si]

;cf <== 0 , [al] <== xatoi( [al] ) if isxdigit( [al] )
          sub_39:
          sub al,30h                       ; '0'
          jc loc_ret_89                    ; Jump if carry Set
          cmp al,0Ah
          cmc                              ; Complement carry
          jnc loc_ret_89                   ; Jump if carry=0
          and al,5Fh                       ; '_'
          sub al,7
          cmp al,0Ah
          jb loc_ret_89                    ; Jump if below
          cmp al,10h
          cmc                             ; Complement carry
loc_ret_89:
          retn

;xatoi() one byte
          sub_40:
          call sub_13                     ; (037F)
          call sub_38                     ; (0655)
          jc loc_91                       ; Jump if carry Set
          mov cx,2
          call sub_36                     ; (0643), xatoi(), len=2
          mov [bx],dl
          inc bx
loc_90:
          clc                             ; Clear carry flag
          retn
loc_91:                                   ;  xref 6710:0672, 引号内的值不需转换
          mov al, [si]
          cmp al,27h                      ; '''
          je loc_92                       ; Jump if equal
          cmp al,22h                      ; '"'
          je loc_92                       ; Jump if equal
          stc                             ; Set carry flag,非法输入
          retn
loc_92:
          mov ah,al
          inc si
loc_93:                                   ;  xref 6710:069F,处理引号
          lodsb                           ; String [si] to al
          cmp al,0Dh
          je loc_97                       ; Jump if equal
          cmp al,ah
          jne loc_94                      ; Jump if not equal
          cmp ah, [si]
          jne loc_90                      ; Jump if not equal
          inc si
loc_94:
          mov [bx],al
          inc bx
          jmp short loc _ 93                ; (068E)

;get list from KBDline, bx <== converted bytes
          sub _ 41:
          mov bx,35B8h
loc _ 95:
          call sub _ 40                     ; (066C), convert one byte
          jnc loc _ 95                      ; Jump if carry=0
          sub bx,35B8h
          jz loc _ 98                       ; Jump if zero, error

;如有 space, TAB, RET 外的字符则输入非法
          sub _ 42:
loc _ 96:
          call sub _ 14                     ; (0388)
          jnz loc _ 98                      ; Jump if not zero
          retn

               ;Illegal command, —B,—J,—K,—V,—X,—Y,—Z
               ;          Called from:  6710:028B, 0450, 0452, 0454
          sub _ 152:
loc _ 97:
          dec si
loc _ 98:
          sub si,35B7h
          mov cx,si                         ;确定出错字符位置
          mov di,37A2h
          call sub _ 27                     ; (041A), fill space
          mov byte ptr [di] ,0
          mov dx,32FAh                      ; fprintf(stdout, "%s^  Error", ...)
loc _ 99:
          call sub _ 21                     ; (03E1)
          jmp loc _ 39                      ; (0255), goto command _ accept
          sub _ 28    endp

;get address, ax=seg, dx=ofs
          sub _ 44    proc near
          call sub _ 45                     ; (06D5)
          jc loc _ 98                       ; Jump if 输入非法
loc _ 100:
        stc                         ; Set carry flag
        retn
        sub _ 44   endp

;get address from KBDline
        sub _ 45   proc near
        call sub _ 13               ; (037F), skip space, TAB, ' ,'
        mov al, [si+1]
        cmp al,53h                  ; ' S'
        je loc _ 103                ; Jump if 输入含有段寄存器
        mov cx,4
        call sub _ 34               ; (0611), xatoi(), len=4
        jc loc _ 100                ; Jump if carry Set
        mov ax,bp
        cmp byte ptr [si] ,3Ah      ; ' :'
        jne loc _ 102               ; Jump if only offset
        push dx
loc _ 101:
        inc si
        mov cx,4
        call sub _ 34               ; (0611), get offset
        pop ax
        jc loc _ 100                ; Jump if carry Set
loc _ 102:
        clc                         ; Clear carry flag
        retn
loc _ 103:
        mov al, [si]
        mov di,718h                 ;' CSED' ,段寄存器第一字符
        mov cx,4
        repne scasb                 ; Rept zf=0+cx>0 Scan es: [di] for al
        jnz loc _ 100               ; Jump if 输入非法段寄存器
        inc si
        inc si
        shl cx,1                    ; Shift w/zeros fill
        mov bx,cx
        cmp byte ptr [si] ,3Ah      ; ' :'
        jne loc _ 100               ; 段寄存器后无 offset 则非法
        push ds _ save [bx]         ; (6710:3188=0)
        jmp short loc _ 101         ; (06EF), get offset
```

```
          sub _ 45   endp
seg _ name db ' CSED'
loc _ 104:
          call sub _ 41                    ; (06A1), get list
          pop di
          pop es
          mov si,35B8h
          mov cx,bx
          rep movsb       ; Rep while cx>0 Mov [si] to es: [di] , copy converted value
          retn

;command E
;function  : enter value
;format  : E <address> [<list>]
          sub _ 46   proc near
          mov bp,ds _ save                 ; (6710:3188=0)
          call sub _ 44                    ; (06CE), get address
          push ax
          push dx
          call sub _ 14                    ; (0388), skip space & TAB
          jnz loc _ 104                    ; Jump if not zero
          pop di
          pop es
loc _ 105:
          call sub _ 17                    ; (03C9), save es, di for print
          push di
          push es
          push ds
          pop es
          mov di,37A2h
          call sub _ 26                    ; (0416), fill es: [di] with space
          xor al,al                        ; Zero register
          stosb                            ; Store al to es: [di] , set string _ end
          call sub _ 19     ; (03D7), fprintf(stdout, "%04X:%04X   %s", ...)
          pop es
          pop di
loc _ 106:
          mov al,es: [di]                  ; get orignal value
          push di
          push es
```

48-70

```
        push ds
        pop es
        mov di,37A2h
        call sub_23              ; (03F1), es: [di] <== itoa( [AL] )
        mov al,2Eh               ; '.'
        stosb                    ; Store al to es: [di]
        xor al,al                ; Zero register
        stosb                    ; Store al to es: [di]
        mov dx,37F2h
        call sub_20              ; (03DA), fprintf(stdout, "%2X.", ...)
        pop es
        pop di
loc_107:
        mov cx,2
        mov dx,0
locloop_108:
        call sub_48              ; (0808), get char to al from KBDline
        mov ah,al
        call sub_39              ; (0657), isxdigit( [AL] ) ?
        xchg ah,al
        je loc_110               ; Jump if no
        mov dh,dl                ; xatoi()
        mov dl,ah
        loop locloop_108         ; Loop if cx > 0
loc_109:
        call sub_48              ; (0808)
loc_110:
        cmp al,8
        je loc_112               ; Jump if equal
        cmp al,7Fh
        je loc_111               ; Jump if equal
        cmp al,2Dh               ; '—'
        je loc_117               ; Jump if equal
        cmp al,0Dh
        je loc_118               ; Jump if equal
        cmp al,20h               ; ' '
        je loc_115               ; Jump if equal
        mov al,8
        call sub_49              ; (080D), fprintf(stdout, "%c", [al] )
        call sub_12              ; (0376), fprintf(stdout, "\0x20\0x08")
        jcxz loc _ 109                ; Jump if cx=0
        jmp short locloop _ 108       ; (076F)
loc _ 111:
        mov al,8
        call sub _ 49                 ; (080D), fprintf(stdout, "%c", [AL] )
loc _ 112:
        cmp cl,2
        je loc _ 113                  ; Jump if ! isxdigit( 1st char )
        inc cl                        ;第二个字符非 HEX 时重输第二个字符
        mov dl,dh
        mov dh,ch
        call sub _ 12                 ; (0376),fprintf(stdout, "\0x20\0x08")
        jmp short locloop _ 108       ; (076F)
loc _ 113:
        mov al,2Eh                    ; '.'
        call sub _ 49                 ; (080D), fprintf(stdout, ".")
        jmp short loc _ 107           ; (0769)

;           Called from:  6710:07D2, 07FB, 0802
;如果正确输入了二个 HEX 字符则替换
  sub _ 47:
        cmp cl,2
        je loc _ 114                  ; Jump if equal
        push cx
        mov cl,4
        shl dh,cl                     ; Shift w/zeros fill
        pop cx
        or dl,dh
        mov es: [di] ,dl
loc _ 114:
        inc di
        retn
loc _ 115:
        call sub _ 47                 ; (07C0), if input correct then replace
        inc cx
        inc cx
        push di
        mov di,37A2h
        push es
        push ds
        pop es
        call sub _ 27                   ; (041A)
        xor al,al                       ; Zero register
        stosb                           ; Store al to es: [di]
        mov dx,37F2h
        call sub _ 20                   ; (03DA), print space
        pop es
        pop di
        mov ax,di
        and al,7
        jz loc _ 116                    ; Jump if zero
        jmp loc _ 106                   ; (074E), 输入下一单元
loc _ 116:
        call sub _ 22                   ; (03E7), printf("\n")
        jmp loc _ 105                   ; (0739)
loc _ 117:
        call sub _ 47                   ; (07C0), if correct then replace
        dec di
        dec di
        jmp short loc _ 116             ; (07F5)
loc _ 118:
        call sub _ 47                   ; (07C0)
        jmp loc _ 57                    ; (03E7), print("\n") then return
        sub _ 46   endp

;get KBD char to al with echo
        sub _ 48   proc near
        mov ah,1
        int 21h   ; DOS Services   ah=function 01h, get keybd char al, with echo
        retn
        sub _ 48   endp

;printf("%c", [al] )
        sub _ 49   proc near
        push di
        push dx
        push es
        push ds
        pop es
        mov di,37F4h
        stosb                       ; Store al to es: [di]
        mov al,0
        stosb                       ; Store al to es: [di]
        mov dx,37F6h
        call sub _ 20               ; (03DA)
        pop es
        pop dx
        pop di
        retn
        sub _ 49   endp

;作 [CX] 次 printf(" %s= %04X", ...)
        sub _ 50   proc near
locloop _ 119:
        mov ds:data _ 13e,si        ; (0000:38A5=0)
        add si,3
        mov ax, [bx]
        add bx,2
        mov ds:data _ 14e,ax        ; (0000:38A7=0)
        mov dx,38A3h
        call sub _ 20               ; (03DA)
        loop locloop _ 119          ; Loop if cx > 0
        retn
        sub _ 50   endp

;fill flags string
        sub _ 51   proc near
        mov di,37A2h
        mov al,20h                  ; ' '
        stosb                       ; Store al to es: [di]

;fill flags string
        sub _ 52:
        mov si,3038h
        mov cx,10h                  ; 标志寄存器 16 位
        mov dx,flags _ save         ; (6710:3192=0F202h)
locloop _ 120:
        lods word ptr cs: [si]      ; String [si] to ax
        shl dx,1                    ; Shift w/zeros fill
        jc loc _ 121                ; Jump if carry Set
        mov ax,cs: [si+1Eh]
  _121:
        or ax,ax                    ; Zero ?
        jz loc _ 122                ; Jump if zero, 无效位不用显示
        stosw                       ; Store ax to es: [di]
        mov al,20h                  ; ' '
        stosb                       ; Store al to es: [di]
loc _ 122:
        loop locloop _ 120          ; Loop if cx > 0
        xor al,al                   ; Zero register
        stosb                       ; Store al to es: [di] ,置串结束标志
        retn
        sub _ 51    endp
            ;           Called from:  6710:11AE, 129F
;显示寄存器内容和 cs:ip 处指令列表
        sub _ 53    proc near
loc _ 123:
        mov si,300Eh                ;start address of registers string
        mov di,37A2h
        mov bx,3178h                ;start address of registers value
        mov byte ptr ds:reg _ num,0Dh   ; (0000:3197=0Eh)
        mov ch,0
        mov cl,ds:reg _ per _ line  ; (0000:3199=0)
loc _ 124:
        sub ds:reg _ num,cl         ; (0000:3197=0Eh)
        call sub _ 50               ; (0823), printf("%s=%04X", ...)
        call sub _ 22               ; (03E7), printf("\n")
        xor ch,ch                   ; Zero register
        mov cl,ds:reg _ per _ line  ; (0000:3199=0)
        cmp cl,ds:reg _ num         ; (0000:3197=0Eh)
        jb loc _ 124                ; Jump if below
        mov cl,ds:reg _ num         ; (0000:3197=0Eh)
        call sub _ 50               ; (0823), printf("%s=%04X", ...)
        call sub _ 51               ; (083B), fill flags string
        mov dx,37F2h
        call sub _ 21               ; (03E1), print flags string
        mov ax,ip _ save            ; (6710:3190=100h)
        mov ds:data _ 218e,ax       ; (6710:3584=0)
        push ax
        mov ax,cs _ save            ; (6710:318E=0)
        mov ds:data _ 219e,ax           ; (6710:3586=0)
        push ax
        mov word ptr ds:data _ 234e,0FFFFh     ; (6710:35AB=0)
        call sub _ 102                  ; (1E7E)
        pop word ptr ds:data _ 219e     ; (66E8:3586=4520h), restore cs, ip
        pop word ptr ds:data _ 218e     ; (66E8:3584=6E69h)
        mov ax,ds:data _ 234e           ; (66E8:35AB=6163h)
        cmp al,0FFh
        jne loc _ 125                   ; Jump if not equal
        jmp loc _ 57                    ; (03E7), printf("\n") then return
loc _ 125:
        cmp ah,0FFh
        je loc _ 126                    ; Jump if equal
        xchg al,ah
loc _ 126:
        cbw                             ; Convrt byte to word
        mov bx,ax
        shl bx,1                        ; Shift w/zeros fill
        mov ax,ds:data _ 40e [bx]       ; (66E8:2413=0B80Dh), "ESCSSSDS"
        mov di,37A2h
        stosb                           ; Store al to es: [di]
        xchg al,ah
        stosb                           ; Store al to es: [di]
        xor al,al                       ; Zero register
        stosb                           ; Store al to es: [di]
        mov dx,ds:data _ 222e           ; (66E8:358C=2072h)
        mov ds:data _ 79d,dx            ; (66E8:38DD=0)
        mov dx,38D9h
        call sub _ 20                   ; (03DA), printf("%s:%04X=", ...)
        mov bx,ds:sreg _ save _ addr [bx]     ; (66E8:3006=8FFFh)
        push ds
        mov ds, [bx]
        mov bx,cs:data _ 222e           ; (6710:358C=0)
        mov bx, [bx]
        pop ds
        mov ds:data _ 81e,bx            ; (66E8:38EF=0)
        mov dx,38EDh                    ;printf("%04X\n", [AL] )
        test byte ptr ds:data _ 230e.0FFh     ; (66E8:35A7 =6Ch)
        jnz loc _ 127                   ; Jump if no error
        xor bh,bh
        mov ds:data _ 5[illegible],bx          ; (66E8:38E6=0)
        mov dx,38E4h                  ; printf(″%02X\n″, ...)
loc _ 127:
        call sub _ 21                 ; (03E1)
loc _ ret _ 128:
        retn
        sub _ 53    endp

loc _ 129:
        jmp loc _ 123                 ; (0863)

;command R
;function : register
;format : R [<reg>]
        sub _ 54    proc near
        call sub _ 13                 ; (037F)
        jz loc _ 129                  ; Jump if no params
        mov dl, [si]
        inc si
        mov dh, [si]
        cmp dh,0Dh
        je loc _ 134                  ; Jump if only one char
        inc si
        call sub _ 42                 ; (06AF), 处理非法输入
        cmp dh,20h                    ; ' '
        je loc _ 134                  ; Jump if equal
        mov di,300Eh                  ;寄存器名串首址
        xchg ax,dx
        push cs
        pop es
        xor cx,cx                     ; Zero register
loc _ 130:
        cmp ax, [di]
        je loc _ 131                  ; Jump if equal
        add di,3
        inc cx
        cmp di,3038h                  ;寄存器名串末址
        jb loc _ 130                  ; Jump if below
        jmp short loc _ 133           ; (098D),非法寄存器
        nop
```

```
loc_131:
        cmp di,3038h
        jne loc_132                    ; Jump if not equal
        dec di
        dec di
        dec di
        mov ax,cs:[di-2]
loc_132:
        push di
        mov di,37A2h
        stosb                          ; Store al to es:[di]
        xchg al,ah
        stosb                          ; Store al to es:[di]
        xor al,al                      ; Zero register
        stosb                          ; Store al to es:[di]
        pop di
        push ds
        pop es
        lea bx,[16Ah + di]             ; (6710:016A=0B8h) Load effective addr
        sub bx,cx
        mov dx,[bx]
        mov ds:data_267e,dx            ; (6710:3897=0)
        mov dx,3893h
        call sub_20                    ; (03DA), printf("%s %04X\n", ...)
        call sub_11                    ; (0340), get KBD line
        call sub_14                    ; (0388), skip space & TAB
        jz loc_ret_128                 ; Jump if zero
        mov cx,4
        call sub_37                    ; (064E), xatoi(), len=4
        call sub_42                    ; (06AF), check illegal_input
        mov [bx],dx
        retn
loc_133:
        mov dx,33CDh                   ;printf("br Error"), then goto command_accept
        jmp short loc_139              ; (09EA)
        nop
loc_134:                               ;  xref 6710:0928, 0931, change flages
        cmp dl,46h                     ; 'F'
        jne loc_133                    ; Jump if not equal, error
        mov di,37A2h
```

```
                call sub_52                 ; (0841), fill flags string
                mov dx,33C3h                ;printf("%s —", ...)
                call sub_20                 ; (03DA)
                call sub_11                 ; (0340), get KBDline
                call sub_14                 ; (0388)
                xor bx,bx                   ; Zero register
                mov dx,flags_save           ; (6710:3192=0F202h)
loc_135:
                lodsw                       ; String [si] to ax
                cmp al,0Dh
                je loc_140                  ; no more input then return
                cmp ah,0Dh
                je loc_141                  ; Jump if "bf Error"
                mov di,3038h
                mov cx,20h
                push cs
                pop es
                repne scasw                 ; Rept zf=0+cx>0 Scan es:[di] for ax
                jnz loc_141                 ; Jump if no matched flags
                mov ch,cl
                and cl,0Fh
                mov ax,1
                rol ax,cl                   ; Rotate
                test ax,bx
                jnz loc_137                 ; Jump if not zero
                or bx,ax
                or dx,ax
                test ch,10h
                jnz loc_136                 ; Jump if not zero
                xor dx,ax
loc_136:
                call sub_13                 ; (037F)
                jmp short loc_135           ; (09B0), get next
loc_137:
                mov dx,33C7h
loc_138:
                call sub_55                 ; (09F4)
loc_139:
                mov ds:data_268e,dx         ; (6710:38B4=0)
                mov dx,38B2h
```

```
          jmp loc_99       ; (06C8), printf("%s Error", ...) then goto command_accept

             ;          Called from:   6710:09E7
          sub_55:
loc_140:
          mov flags_save,dx          ; (6710:3192=0F202h)
          retn
loc_141:
          mov dx,33CAh               ;"bf Error"
          jmp short loc_138          ; (09E7)
          sub_54   endp

;          Called from:   6710:0218, 0B14
;传送命令行及分析文件名
          sub_56   proc near
          mov es,ds_save             ; (6710:3188=0)
          push si
          mov di,81h
loc_142:
          lodsb                      ; String [si] to al, transfer command line
          stosb                      ; Store al to es: [di]
          cmp al,0Dh
          jne loc_142                ; Jump if not equal
          sub di,82h
          xchg ax,di
          mov es:Param_Len,al        ; (0000:0080=0F5h), len of command line
          pop si
          mov di,5Ch
          mov ax,2901h
          int 21h                    ; DOS Services   ah=function 29h
                                     ;  parse filenam @ds:si FCBes:di
          mov byte ptr ax_h_save,al  ; (6710:3178=0)
          call sub_81                ; (135E)
          mov di,6Ch
          mov ax,2901h
          int 21h                    ; DOS Services   ah=function 29h
                                     ;  parse filenam @ds:si FCBes:di
          mov ax_l_save,al           ; (6710:3179=0)
loc_ret_143:
          retn
```

```
        sub _ 56   endp

;unlink file
        sub _ 57   proc near
        mov byte ptr ds:data _ 53e,41h ; (66E8:3575=72h) 'A'
        jmp short loc _ 144            ; (0A6C)

;find the position of "." in filename
        sub _ 58:
        mov byte ptr ds:data _ 53e,0   ; (6710:3575=0)
        jmp short loc _ 144            ; (0A6C)

;load file
        sub _ 59:
        mov byte ptr cs:data _ 53e,4Bh   ; (6710:3575=0) 'K'
        mov byte ptr cs:data _ 210e,1  ; (6710:3574=0)
        jmp short loc _ 144            ; (0A6C)

;open file for read or read _ write
        sub _ 60:
        mov byte ptr ds:data _ 53e,3Dh   ; (6710:3575=0) '='
        mov byte ptr ds:data _ 210e,2  ; (6710:3574=0)
        call sub _ 62                  ; (0A6C)
        jnc loc _ ret _ 143            ; Jump if carry=0
        mov byte ptr ds:data _ 53e,3Dh; (6710:3575=0) '='
        mov byte ptr ds:data _ 210e,0  ; (6710:3574=0)
        jmp short loc _ 144            ; (0A6C)

;creat file
        sub _ 61:
        mov byte ptr ds:data _ 53e,3Ch ; (66E8:3575=72h) '<'

;file operation
        sub _ 62:
loc _ 144:
        push ds
        push es
        push ax
        push bx
        push cx
```

```
          push dx
          push si
          xor ax,ax                   ; Zero register
          mov cs:data _ 213e,ax       ; (6710:3577=0)
          mov ah,37h                  ; '7'
          int 21h                     ; DOS Services  ah=function 37h, get switch chars
          mov cs:data _ 212e,dl       ; (6710:3576=0)
          mov si,81h
loc _ 145:
          call sub _ 63               ; (0ADF), upper( [si++] )
          call sub _ 65               ; (0B05)
          jz loc _ 148                ; Jump if [AL] == 0x0d or switch char
          call sub _ 64               ; (0AF2)
          jz loc _ 145                ; Jump if [AL] is delimeter
          mov dx,si
          dec dx
loc _ 146:
          cmp al,2Eh                  ; '.'
          jne loc _ 147               ; Jump if not equal
          mov cs:data _ 213e,si       ; (6710:3577=0)
loc _ 147:
          call sub _ 63               ; (0ADF), upper( [si++] )
          call sub _ 64               ; (0AF2)
          jz loc _ 148                ; Jump if [AL] is delimeter
          call sub _ 65               ; (0B05), zf=1 if [AL] == 0x0d or switch char
          jnz loc _ 146               ; Jump if not zero
loc _ 148:
          dec si
          push word ptr [si]
          mov byte ptr [si] ,0
          mov al,cs:data _ 210e       ; (6710:3574=0)
          mov ah,cs:data _ 53e        ; (6710:3575=0)
          or ah,ah                    ; Zero ?
          jz loc _ 149                ; Jump if zero
          mov cs:data _ 204e,dx       ; (6710:3553=0)
          mov cs:data _ 203e,si       ; (6710:3551=0)
          push cs
          pop es
          mov bx,31FDh
          xor cx,cx                   ; Zero register
```

```
        int 21h
        mov cs:data _ 214e,ax          ; (6710:3579=0)
loc _ 149:
        pop word ptr [si]
        pop si
        pop dx
        pop cx
        pop bx
        pop ax
        pop es
        pop ds
        retn
        sub _ 57   endp

;upper( [si++] )
        sub _ 63   proc near
        lodsb                          ; String [si] to al
        cmp al,61h                     ; 'a'
        jb loc _ ret _ 150             ; Jump if below
        cmp al,7Ah                     ; 'z'
        ja loc _ ret _ 150             ; Jump if above
        sub al,20h                     ; ' '
        mov [si-1] ,al
loc _ ret _ 150:
        retn
        sub _ 63   endp

        cmp al,5Bh                     ;' ['
        je loc _ ret _ 150             ; Jump if equal

;zf=1 if [AL] in {space, TAB, '    ;', '=', ',')
        sub _ 64   proc near
        cmp al,20h                     ; ' '
        je loc _ ret _ 150             ; Jump if equal
        cmp al,3Bh                     ;''
        je loc _ ret _ 150             ; Jump if equal
        cmp al,3Dh                     ; '='
        je loc _ ret _ 150             ; Jump if equal
        cmp al,9
        je loc _ ret _ 150             ; Jump if equal
```

```
          cmp al,2Ch
loc_ret_151:
          retn
          sub_64    endp

;zf=1 if [AL] == 0x0d or switch char
          sub_65    proc near
          cmp al,cs:data_212e            ; (6710:3576=0)
          je loc_ret_151                 ; Jump if equal
          cmp al,0Dh
          retn
          sub_65    endp

;command N
;function : name
;format : N name
;action : 传送命令行和 FCB
          sub_66    proc near
          or data_177,1                  ; (6710:3195=0)
          call sub_56                    ; (09FE), transfer command line & analysis filename
          mov al,byte ptr ax_h_save      ; (6710:3178=0)
          mov ds:data_215e,al            ; (6710:357F=0)
          push es
          pop ds
          push cs
          pop es
          mov si,5Ch
          mov di,si
          mov cx,52h
          rep movsw                      ; Rep while cx>0 Mov [si] to es: [di]
loc_ret_152:
          retn
          sub_66    endp
loc_153:
          mov dx,326Fh                   ;"file not found"
          jmp loc_47                     ; (032C)

;zf=1 if is HEX file
          sub_67    proc near
          cmp byte ptr ds:data_215e,0FFh       ; (6710:357F=0)
          je loc _ 153                    ; Jump if equal
          call sub _ 58                   ; (0A37)
          mov bx,ds:data _ 213e           ; (6710:3577=0)
          cmp word ptr [bx] ,4548h
          jne loc _ ret _ 152             ; Jump if not equal
          cmp byte ptr [bx+2] ,58h        ; 'X'
          retn
          sub _ 67   endp

;zf=1 if is EXE file
          sub _ 68   proc near
          push bx
          mov bx,ds:data _ 213e           ; (6710:3577=0)
          cmp word ptr [bx] ,5845h
          jne loc _ 154                   ; Jump if not equal
          cmp byte ptr [bx+2] ,45h        ; 'E'
loc _ 154:
          pop bx
          retn
          sub _ 68   endp

;command L
;function : load file or disk sectors
;format : L [<address> [<drive> <rec> <recs>] ]
sub _ cmd _ L:
          mov byte ptr ds:RWfunc _ no,3Fh   ;DOS Read Function
          jmp short loc _ 155             ; (0B68)

;command W
;function : write file or disk sectors
;format : W [<address> [<drive> <rec> <recs>] ]
sub _ cmd _ W:
          mov byte ptr ds:RWfunc _ no,40h    ; DOS Write Function
loc _ 155:
          mov bp,ds:cs _ save             ; default segment
          call sub _ 14                   ; (0388)
          jnz loc _ 156                   ; Jump if not zero
          jmp loc _ 162                   ; (0BFC), L/W only, is L/W file
loc _ 156:
          call sub _ 44                   ; (06CE), get address
```

```
         call sub _ 14                    ; (0388), more parameters ?
         jnz loc _ 157                    ; Jump if yes
         jmp loc _ 164                    ; (0C0C), address only, is L/W file
loc _ 157:
         push ax
         mov bx,dx
         mov cx,2
         call sub _ 36                    ; (0643), dx=drive# <== xatoi()
         push dx
         mov cx,8
         call sub _ 36                    ; (0643), dx=rec <== xatoi()
         mov cx,cs:data _ 261e            ; (6710:3869=0)
         mov cs:data _ 262e,cx            ; (6710:386B=0)
         push dx
         mov cx,3
         call sub _ 36                    ; (0643), dx=recs <== xatoi()
         call sub _ 42                    ; (06AF), 处理非法输入
         mov cx,dx
         pop dx
         pop ax
         cbw                              ; Convrt byte to word
         mov ds:drive _ char,al           ; (66E8:32C8=4Eh)
         push ax
         push bx
         push dx
         mov dl,al
         mov ah,0Dh
         int 21h                          ; DOS Services   ah=function 0Dh
             ;  flush disk buffers to disk
         inc dl
         mov ah,32h                       ; '2'
         int 21h                          ; DOS Services   ah=function 32h
             ;  get ds:bx ptr to disk block
         pop dx
         pop bx
         or al,al                         ; Zero ?
         pop ax
         pop ds
         jnz loc _ 160                    ; Jump if R/W error
         push di
```

```
        mov cs:data_263e,bx          ; (6710:3873=0), ofs
        mov cs:data_264e,ds          ; (6710:3875=0), seg
        push cs
        pop ds
        mov bx,386Dh                 ;R/W buffer
        mov [bx] ,dx
        mov dx,cs:data_262e          ; (6710:386B=0)
        mov [bx+2] ,dx
        mov [bx+4] ,cx
        mov cx,0FFFFh
        cmp cs:RWfunc_no,40h         ; is write ?
        je loc_158                   ; Jump if yes
        int 25h                      ; Absolute disk read, drive al
        jmp short loc_159            ; (0BF0)
loc_158.
        int 26h                      ; Absolute disk write, drive al
loc_159:
        jnc loc_161                  ; Jump if R/W success
loc_160:
        jmp loc_463                  ; (23C1), R/W error
loc_161:
        popf                         ; Pop flags
        pop di
        mov ah,0Dh
        int 21h                      ; DOS Services   ah=function 0Dh
            ;   flush disk buffers to disk
        retn

; Called from:   6710:023E
; L/W only
        sub_69   proc near
loc_162:
        mov ax,cs_save               ; default segment
        mov dx,100h                  ; prepared for PSP
        call sub_67                  ; (0B32), is HEX file ?
        jnz loc_165                  ; Jump if no
        xor dx,dx                    ; HEX 文件不需 PSP
loc_163:
        jmp loc_194                  ; (0EC2)
```

5A 8-70

```
; L/W address
loc_164:
        call sub_67                 ; (0B32), is HEX file ?
        jz loc_163                  ; Jump if yes
loc_165:
        call sub_68                 ; (0B4B), is EXE file ?
        jnz loc_166                 ; Jump if no
        cmp RWfunc_no,3Fh           ; is READ ?
        je loc_167                  ; Jump if yes
        mov dx,333Dh                ; error, "EXE files cannot be written"
        jmp loc_47                  ; (032C)
loc_166:
        cmp RWfunc_no,40h           ; is WRITE ?
        je loc_175                  ; Jump if yes
        cmp word ptr [bx],4F43h     ;'CO', is COM file ?
        jne loc_175                 ; Jump if no
        cmp byte ptr [bx+2],4Dh     ; 'M'
        jne loc_175                 ; Jump if no
loc_167:                            ;   xref 6710:0C1B, Read EXE / COM files
        dec si
        cmp dx,100h
        jne loc_168                 ; 应予留 256 字节给 PSP
        cmp ax,cs_save              ; (6710:318E=0)
        je loc_169
loc_168:
        jmp loc_98                  ; (06B6), goto illegal command
loc_169:
        call sub_60                 ; (0A4C), open file for read
        jnc loc_170                 ; Jump if carry=0
        mov ax,2
        jmp loc_192                 ; (0E97), error, "file not found"
loc_170:
        xor dx,dx
        xor cx,cx
        mov bx,ds:data_214e         ; file handle
        mov al,2
        mov ah,42h
        int 21h                     ; DOS Services   ah=function 42h
            ;  move file ptr to file end
        call sub_68                 ; (0B4B), is EXE file ?
```

```
            jnz loc _ 171                   ; Jump if no
            sub ax,200h                     ; EXE head is 200h bytes
loc _ 171:
            mov bx _ save,dx                ; (6710:317A=0), save file size low word
            mov cx _ save,ax                ; (6710:317C=0), save file size high word
            mov ah,3Eh
            int 21h                         ; DOS Services   ah=function 3Eh
                                            ;   close file, bx=file handle
            jmp loc _ 189                   ; (0DC4)

loc _ 172:
            mov dx,32EEh                    ; "Insufficient memory\n"
            call sub _ 21                   ; (03E1)
            jmp loc _ 39                    ; (0255)

loc _ 173:
            jmp loc _ 185                   ; (0D56), write

loc _ 174:
            jmp loc _ 182                   ; (0D4A), open file error

loc _ 175:
            push ax                         ;segment
            push dx                         ;offset
            cmp RWfunc _ no,40h             ; WRITE ?
            je loc _ 173                    ; Jump if yes
            call sub _ 60                   ; (0A4C), open file for read
            jc loc _ 174                    ; Jump if error
            mov bx,ds:data _ 214e           ; (6710:3579=0)
            mov ax,4202h
            xor dx,dx                       ; Zero register
            mov cx,dx
            int 21h                         ; DOS Services   ah=function 42h
                                            ;   move file ptr to file end
            mov si,dx                       ;file size
            mov di,ax
            mov ax,4200h
            xor dx,dx
            mov cx,dx
            int 21h                         ; DOS Services   ah=function 42h
```

```
                                    ;  move file ptr to file head
        pop ax                      ;ofs
        pop bx                      ;seg
        push bx
        push ax
        add ax,0Fh
        rcr ax,1                    ; Rotate thru carry
        mov cl,3
        mov cl,4
        shr ax,cl                   ; Shift w/zeros fill
        add bx,ax
        mov dx,si                   ;file size
        mov ax,di
        cmp dx,10h
        jae loc _ 172               ; "Insufficient memory", if >= 1 MB
        mov cx,10h
        div cx                      ; ax,dx rem=dx:ax/reg, convert to paragraph
        or dx,dx
        jz loc _ 176
        inc ax
loc _ 176:
        add ax,bx
        jc loc _ 172                ; "Insufficient memory"
        cmp ax,cs:data _ 94         ; (6710:0002=0), memory top
        ja loc _ 172                ; "Insufficient memory"
        mov cx _ save,di            ; (6710:317C=0)
        mov bx _ save,si            ; (6710:317A=0)
        pop dx
        pop ax

; block R/W, close file
        sub _ 70:
loc _ 177:
        mov bx,dx
        and dx,word ptr 000Fh
        mov cl,4
        shr bx,cl                   ; Shift w/zeros fill
        add ax,bx
        push ax
        push dx
        mov ds:data_55e,dx          ; (66E8:357B=0A232h)
        mov ds:data_56e,ax          ; (66E8:357D=4537h)
        mov cx,0FFF0h
        or si,si                    ; Zero ?
        jnz loc_178                 ; Jump if not zero
        mov cx,di
loc_178:
        push ds
        push bx
        mov bx,ds:data_214e          (66E8:3579=0F000h)
        mov ah,ds:RWfunc_no         ; (66E8:3196=3)
        lds dx,dword ptr ds:data_55e ; (66E8:357B=0A232h) Load 32 bit ptr
        int 21h                     ; DOS Services
        pop bx
        pop ds
        jc loc_179
        cmp byte ptr ds:RWfunc_no,40h       ; WRITE ?
        jne loc_180
        cmp cx,ax
        je loc_180                  ; R/W 字符数确认
loc_179:
        mov cx,ax                   ; 实际 读/写 字符数
        stc
        pop dx
        pop ax
        retn
loc_180:
        mov cx,ax
        sub di,cx
        sbb si,0
        or cx,cx                    ; Zero ?
        pop dx
        pop ax
        jz loc_181                  ; Jump if zero
        add dx,cx
        mov bx,si
        or bx,di
        jnz loc_177                 ; R/W again
loc_181:
        push ax
```

```
                push bx
                mov bx,ds:data _ 214e          ; (66E8:3579=0F000h)
                mov ah,3Eh                     ;
                int 21h                        ; DOS Services   ah=function 3Eh
                                               ;   close file, bx=file handle
                pop bx
                pop ax
                retn
loc _ 182:
                mov dx,3280h                   ; "Found not found"
                jmp loc _ 47                   ; (032C)
loc _ 183:
                mov dx,3374h                   ; "(w)rite error, no destination defined"
loc _ 184:
                jmp loc _ 47                   ; (032C)
loc _ 185:                                     ;   xref 6710:0C7E, write
                cmp byte ptr ds:data _ 177,0   ; (66E8:3195=79h)
                je loc _ 183
                call sub _ 61                  ; (0A67), creat file
                jc loc _ 187
                mov si,ds:bx _ save            ; (:317A=37Ch)
                cmp si,0Fh
                jle loc _ 186
                xor si,si
loc _ 186:
                mov ds:data _ 78e,si           ; (66E8:38CE=0)
                mov di,ds:cx _ save            ; (66E8:317C=7C16h)
                mov ds:data _ 77e,di           ; (66E8:38CC=0)
                mov dx,38CAh                   ; printf("writing %04LX bytes",...)
                call sub _ 21                  ; (03E1)
                pop dx
                pop ax
                call sub _ 70                  ; (0CE5), block write
                jnc loc _ 188                  ; Jump if carry=0
                call sub _ 71                  ; (0DBB), close file
                call sub _ 57                  ; (0A30), unlink file
                mov dx,32B3h                   ; "Insufficient space on disk"
                jmp short loc _ 184            ; (0D53)
                call sub _ 71                  ; (0DBB), close file
                jmp loc _ 39                   ; (0255)
```

```
loc _ 187:
        mov dx,ds:data _ 204e           ; (6710:3553=0)
        mov si,ds:data _ 203e           ; (6710:3551=0)
        push word ptr [si]
        mov byte ptr [si] ,0
        mov ax,4300h
        int 21h                         ; DOS Services   ah=function 43h
                                        ;   get/set file attrb, nam@ds:dx
        pop word ptr [si]
        mov dx,3296h                    ; "File creation error"
        jc loc _ 184
        test cx,7
        jz loc _ 184
        mov dx,3384h                    ; "Access denied"
        jmp short loc _ 184             ; (0D53)

;close file
        sub _ 71:
loc _ 188:
        mov ah,3Eh
        mov bx,ds:data _ 214e           ; (66E8:3579=0F000h)
        int 21h                         ; DOS Services   ah=function 3Eh
                                        ;   close file, bx=file handle
        retn

loc _ 189:
        pop word ptr ds:data _ 185
        inc byte ptr ds:data _ 184
        mov bx,ds:data _ 186            ;active PSP
        mov ax,ds:data _ 226e           ;debug' s PSP
        mov ds,ax
        cmp ax,bx
        je loc _ 190
        jmp loc _ 61                    ; (045E), terminate
loc _ 190:
        mov ax,cs:ds _ save             ; (6710:3188=0), subPSP
loc _ 191:
        mov byte ptr cs:data _ 184,0    ; (6710:31F2=0)
        mov cs:data _ 189,ax            ; (6710:31FB=0)
        push cs:data _ 185              ; (6710:31F3=0)
push ax
mov bx,cs
sub ax,bx
push es
mov es,cs:data_226e          ; (6710:359C=0)
mov bx,ax
add bx,10h                   ;PSP 占 10h 个 paragraph
mov ax,cs
sub ax,cs:data_226e          ; (6710:359C=0)
add bx,ax
mov ah,4Ah
int 21h                      ; DOS Services   ah=function 4Ah
      ;   change mem allocation, bx=siz
pop es
pop ax
mov cs:data_190,ax           ; (6710:3201=0)
mov cs:data_191,ax           ; (6710:3205=0)
mov cs:data_193,ax           ; (6710:3209=0)
push ds
push cs
pop ds
call sub_59                  ; (0A3E), EXEC load file
pop ds
mov ax,cs:data_214e          ; (6710:3579=0), error code
jc loc_192
call sub_9                   ;set vector int22h
mov ah,51h
int 21h                      ; DOS Services   ah=function 51h
      ;   get active PSP segment in bx
mov cs:data_186,bx           ; (6710:31F5=0)
mov cs:ds_save,bx            ; (6710:3188=0)
mov cs:es_save,bx            ; (6710:318A=0)
mov es,bx
mov word ptr es:int22h_ofs, offset int_22h_entry    ;2F4h
mov es:int22h_seg,cs         ; (0010:000C=4C6Eh)
les di,cs:data_195           ; (6710:320F=0) Load 32 bit ptr
mov cs:cs_save,es            ; (6710:318E=0)
mov cs:ip_save,di            ; (6710:3190=100h)
mov cs:data_219e,es          ; (6710:3586=0)
mov cs:data_218e,di          ; (6710:3584=0)
```

```
        mov cs:data _ 217e,es           ; (6710:3582=0)
        mov cs:data _ 216e,di           ; (6710:3580=0)
        mov cs:data _ 224e,es           ; (6710:3590=0)
        mov cs:data _ 223e,di           ; (6710:358E=0)
        mov bx,ds
        mov ah,50h
        int 21h                         ; DOS Services   ah=function 50h
              ;  set active PSP segmnt from bx
        les di,cs:data _ 194            ; (6710:320B=0) Load 32 bit ptr
        mov ax,es: [di]
        inc di
        inc di
        mov word ptr cs:ax _ h _ save,ax   ; (6710:3178=0)
        mov cs:ss _ save,es             ; (6710:318C=0)
        mov cs:sp _ save,di             ; (6710:3180=5Ah)
        retn
loc _ 192:                              ; error return
        push cs
        pop ds
        mov dx,3280h                    ; "File not found"
        cmp ax,2
        je loc _ 193
        mov dx,3384h                    ; "Access denied"
        cmp ax,5
        je loc _ 193
        mov dx,32EEh                    ; "Insufficient memory"
        cmp ax,8
        je loc _ 193
        mov dx,3317h                    ; "Error in EXE or HEX file"
        cmp ax,0Bh
        je loc _ 193
        mov dx,334Ch                    ; "EXEC failure"
loc _ 193:
        call sub _ 21
        jmp loc _ 39                    ;goto command _ accept
        sub _ 69   endp

loc _ 194:                              ;read HEX file
        mov ss:data _ 222e,dx           ; (6A4E:358C=0)
        mov dx,333Dh                    ; HEX files cannot be written
        cmp byte ptr ss:[3196h],40h       ; RWfunc_no, WRITE ?
        jne loc_195
        jmp loc_203                       ; (0F5A)
loc_195:                                  ; read
        mov es,ax
        call sub_60                       ; open file
        mov dx,3280h
        jnc loc_196
        jmp loc_47                        ; "File not found"
loc_196:                                  ; load file
        xor bp,bp                         ; set low significant size to zero
        mov si,37A2h                      ;set pointer to buffer tail
loc_197:
        call sub_72                       ; block read
        cmp al,3Ah                        ;':'
        jne loc_197
        call sub_74                       ;get 1 byte
        mov cl,al
        mov ch,0
        jcxz loc_201                      ; Jump if cx=0
        call sub_74                       ; get 1 byte
        mov bh,al
        call sub_74                       ; get 1 byte
        mov bl,al
        add bx,ss:data_222e               ; (6A4E:358C=0)
        mov di,bx
        call sub_74                       ; get 1 byte
locloop_198:
        call sub_74                       ; get 1 byte
        stosb
        cmp di,bp
        jbe loc_199
        mov bp,di
loc_199:
        loop locloop_198
        jmp short loc_197

;block read
        sub_72  proc near
        cmp si,37A2h                      ; buffer tail
```

```
          jne loc _ 200                   ; Jump if buffer not empty
          mov dx,35A2h                    ; buffer head
          mov si,dx
          mov ah,3Fh                      ; read file
          push bx
          push cx
          mov cx,200h                     ;block size = 512 bytes
          mov bx,ss:data _ 214e
          int 21h                         ; DOS Services  ah=function 3Fh
                                          ;  read file, cx=bytes, to ds:dx
          pop cx
          pop bx
          or ax,ax
          jz loc _ 201                    ; Jump if EOF
loc _ 200:
          lodsb
          cmp al,1Ah                      ;EOF
          je loc _ 201
          or al,al
          jz loc _ 201
          retn
loc _ 201:
          mov ss:[317Ch],bp
          mov word ptr ss:[317Ah],0
loc _ ret _ 202:
          retn
          sub _ 72  endp

;read one char from buffer or disk file for HEX file then convert to hex value
          sub _ 73  proc near
          call sub _ 72                   ; (0F19), read
          call sub _ 39                   ; (0657), xatoi() ?
          jnc loc _ ret _ 202
          mov dx,3317h                    ;"Error in EXE or HEX file\n"
loc _ 203:
          jmp loc _ 47                    ;goto command _ accept
          sub _ 73  endp

;get one byte
          sub _ 74  proc near
```

```
        call sub _ 73                 ; (0F4F)
        mov bl,al
        call sub _ 73                 ; (0F4F)
        shl bl,1                      ; Shift w/zeros fill
        shl bl,1
        shl bl,1
        shl bl,1
        or al,bl
        retn
        sub _ 74   endp

;command P
;function : 执行语句
;format : P [ =<address> ] [<value>]
        sub _ 75   proc near
        mov byte ptr ds:data _ 52e,0FFh    ; (6710:3550=0)
        call sub _ 79                 ; get address
        call sub _ 13                 ; skip space, TAB, ',' 
        call sub _ 38                 ; (0655), isxdigit( [si] ) ?
        mov dx,1
        jc loc _ 204                  ; Jump if no
        mov cx,4
        call sub _ 36                 ; get step value
        call sub _ 83                 ; if step value is 0 then goto illegal command
loc _ 204:
        mov ds:data _ 237e,dx         ; (6710:35AF=0)
        call sub _ 42                 ; 处理非法输入
        mov dx,work _ seg
        mov cs _ save,dx
        mov dx,work _ ofs
        mov ip _ save,dx
loc _ 205:
        mov es,cs _ save
        mov di,ip _ save
        xor dx,dx
loc _ 206:
        mov al,es: [di]               ; get one byte from code area
        cmp al,0F0h
        je loc _ 207                  ; Jump if is LOCK
        cmp al,26h
        je loc _ 207                   ; Jump if is ES;
        cmp al,2Eh
        je loc _ 207                   ; Jump if is CS;
        cmp al,36h
        je loc _ 207                   ; Jump if is SS;
        cmp al,3Eh
        jne loc _ 208                  ; Jump if is DS;
loc _ 207:
        inc di                         ;skip
        jmp short loc _ 206            ; get next byte
loc _ 208:
        cmp al,0E8h
        je loc _ 213                   ; Jump if is CALL NEAR
        cmp al,9Ah
        je loc _ 211                   ; Jump if is CALL FAR
        cmp al,0FFh
        je loc _ 210                   ; Jump if is INC, DEC, CALL, JMP, PUSH for EA
        cmp al,0CCh
        je loc _ 215                   ; Jump if is INT03h
        cmp al,0CDh
        je loc _ 214                   ; Jump if is INT
        cmp al,0E2h
        je loc _ 214                   ; Jump if is LOOP DISP8
        cmp al,0E1h
        je loc _ 214                   ; Jump if is LOOPZ DISP8
        cmp al,0E0h
        je loc _ 214                   ; Jump if is LOOPNZ DISP8
        and al,0FEh
        cmp al,0F2h
        je loc _ 209                   ; Jump if is REPN, REPNZ
        jmp loc _ 217                  ; trace
loc _ 209:                             ;REP
        mov al,es:[di+1]
        and al,0FEh
        cmp al,0A4h
        je loc _ 214                   ; Jump if is MOVS
        cmp al,0A6h
        je loc _ 214                   ; Jump if is CMPS
        cmp al,0AEh
        je loc _ 214                   ; Jump if is SCANS
        cmp al,0ACh
        je loc_214                  ; Jump if is LODS
        cmp al,0AAh
        je loc_214                  ; Jump if is STOS
        jmp short loc_217           ;trace
        nop
loc_210:                            ;  EA
        mov al,es:[di+1]
        and al,0F8h
        cmp al,50h
        je loc_213
        cmp al,58h
        je loc_213
        cmp al,90h
        je loc_212
        cmp al,98h
        je loc_212
        cmp al,0D0h
        je loc_214
        jmp short loc_217           ; trace
        nop
loc_211:                            ; dx=5
        inc dx
loc_212:                            ; dx=4
        inc dx
loc_213:                            ; dx=3
        inc dx
loc_214:                            ; dx=2
        inc dx
loc_215:                            ; dx=1
        inc dx
        add di,dx                   ; dx = bytes of statement
        mov ds:data_241e,di         ; save address
        mov ds:data_242e,es
        mov al,es:[di]              ; save one byte statement
        mov ds:data_243e,al
        mov byte ptr es:[di],0CCh   ; insert INT03h
        mov word ptr ds:data_236e,1
        jmp loc_226                 ; (1153)
```

```
;command T
;function : 执行指令
;format : T [=address] [value]
        sub_76:
        mov byte ptr ds:data_52e,0
        call sub_79                  ; get address
        call sub_13                  ; skip space, TAB, ','
        call sub_38                  ; isxdigit( [si] ) ?
        mov dx,1                     ;default value is 1
        jc loc_216
        mov cx,4
        call sub_36                  ; get value
        call sub_83                  ; if value==0 then goto illegal command
loc_216:
        mov ds:data_237e,dx          ; save count
        call sub_42                  ; 处理非法输入
        mov dx,work_seg
        mov cs_save,dx
        mov dx,work_ofs
        mov ip_save,dx
loc_217:
        mov word ptr ds:data_236e,0  ; (6710:35AD=0)
        mov es,cs_save               ; (6710:318E=0)
        mov di,ip_save               ; (6710:3190=100h)
        mov al,es:[di]
        cmp al,0E4h                  ; is IN AL, Port ?
        jne loc_218                  ; Jump if no
        cmp byte ptr es:[di+1],21h   ; Port == 21h ?
        jne loc_220                  ; Jump if no
        add ip_save,2                ; goto next instruction
        jmp short loc_219            ; goto INT1
        nop
loc_218:
        cmp al,0ECh                  ; is IN AL, DX ?
        jne loc_220                  ;jump if no
        cmp dx_save,21h              ; DX == 21h ?
        jne loc_220
        add ip_save,1                ;goto next instruction
loc_219:                             ;  goto INT1
        mov ax,word ptr ax_h_save
```

```
          in al,21h                         ; port 21h, 8259—1 int IMR
          mov word ptr ax _ h _ save,ax
          jmp loc _ 231                     ; INT1
loc _ 220:
          cmp al,0CDh
          je loc _ 222                      ; Jump if is INT
          cmp al,0CEh
          jne loc _ 221                     ; Jump if is INTO
          test flags _ save,800h
          jz loc _ 224                      ; Jump if OV not set
          mov bx,4
          dec ip _ save
          jmp short loc _ 223
loc _ 221:
          cmp al,0CCh
          jne loc _ 224                     ; Jump if not INT3
          mov bx,3
          dec ip _ save
          jmp short loc _ 223
loc _ 222:
          mov bl,es:[di+1]
          xor bh,bh
loc _ 223:                                  ;INT instruction
          shl bx,1
          shl bx,1
          xor di,di
          mov es,di
          mov ax,es:[bx]                    ;get vector
          mov bx,es:[bx+2]
          xchg ax,ip _ save                 ;置 INT 入口为将要执行的指令
          xchg bx,cs _ save
          mov es,ss _ save
          mov di,sp _ save
          mov cx,flags _ save
          sub di,2
          mov es:[di] ,cx                   ;pushf
          sub di,2
          mov es:[di] ,bx                   ;push seg
          sub di,2
          add ax,2
```

```
        mov es:[di],ax              ;push next ofs
        mov sp_save,di
        and cx,0FDFFh               ;mask IF
        and cx,0FEFFh               ;mask TF
        mov flags_save,cx
        mov bx,data_186
        mov ah,50h
        int 21h                     ; DOS Services  ah=function 50h
                                    ;   set active PSP segmnt from bx
        jmp loc_231                 ;INT1
loc_224:
        mov data_176,al             ; save one byte instruction
        or flags_save,100h          ; set TF
        cli                         ; Disable interrupts
        in al,21h                   ; port 21h, 8259-1 int IMR
        jmp short loc_225
loc_225:
        mov ds:data_199e,al
        mov al,0FFh
        out 21h,al                  ; 屏蔽外部中断
        sti

loc_226:                            ;P, T, G 公用
        mov bx,data_186
        mov ah,50h
        int 21h                     ; DOS Services  ah=function 50h
                                    ;   set active PSP segmnt from bx
        mov ax,5D0Ah
        mov dx,355Eh
        int 21h
        push ds
        xor ax,ax
        mov ds,ax
        mov word ptr ds:int03_ofs, offset Int0x03
        mov ds:int03_seg,cs
        mov word ptr ds:int01_ofs, offset Int0x01
        mov ds:int01_seg,cs
        cli
        mov word ptr ds:int23h_ofs, offset Int0x23x
        mov ds:int23h_seg,cs
```

```
          pop ds
          mov sp,3178h
          pop ax
          pop bx
          pop cx
          pop dx
          pop bp
          pop bp
          pop si
          pop di
          pop es
          pop es
          pop ss
          mov sp,sp_save
          push flags_save
          push cs_save
          push ip_save
          mov ds,ds_save
          iret                              ;执行指令

loc_227:
          call sub_22                       ; printf("\n")
          call sub_53                       ; 显示寄存器内容和指令列表
          test byte ptr ds:data_52e,0FFh
          jnz loc_228
          jmp loc_217                       ; T
loc_228:
          jmp loc_205                       ; P

Int0x23x:
          add sp,6
          jmp short loc_230

Int0x03:
          push bp
          mov bp,sp
db 0FFh,8Eh,02h,00h                           dec word ptr [bp+0002]
          pop bp
          jmp short loc_230
          nop
```

```
Int0x01:
        push bp
        mov bp,sp
        push ax
        mov al,cs:data_199e
        out 21h,al                          ;恢复外部中断许可
        mov al,cs:data_176
        cmp al,9Ch
        jne loc_229
        and word ptr [bp+8],0FEFFh      ;mask TF
loc_229:
        pop ax
        pop bp
loc_230:
        mov cs:sp_save,sp
        mov cs:ss_save,ss
        mov cs:flags_save,cs
        mov ss,cs:flags_save
        mov sp,318Ch
        push es
        push ds
        push di
        push si
        push bp
        dec sp
        dec sp
        push dx
        push cx
        push bx
        push ax
        push ss
        pop ds
        mov ss,ds:ss_save
        mov sp,ds:sp_save
        pop word ptr ds:ip_save
        pop word ptr ds:cs_save
        pop ax
        and ax,0FEFFh    ;mask TF
        mov ds:flags_save,ax
        mov ds:sp_save,sp
loc_231:
        push ds
        pop es
        push ds
        pop ss
        mov sp,3178h
        push ds
        xor ax,ax
        mov ds,ax
        mov word ptr ds:int23h_ofs, offset Int0x23    ;restore INT23h
        mov ds:int23h_seg,cs
        pop ds
        sti                             ; Enable interrupts
        cld                             ; Clear direction
        mov ah,59h
        int 21h                         ; DOS Services   ah=function 59h
                                        ;  get extended error info in ax
        mov ss:data_291e,ax
        mov ss:data_292e,bx
        mov ss:data_293e,cx
        mov ss:data_294e,dx
        mov ss:data_295e,si
        mov ss:data_296e,di
        mov ss:data_297e,ds
        mov ss:data_298e,es
        mov ax,cs
        mov ds,ax
        mov es,ax
        mov ah,51h
        int 21h                         ; DOS Services   ah=function 51h
            ;   get active PSP segment in bx
        mov data_186,bx                 ; (6710:31F5=0)
        mov bx,ds:data_226e             ; (6710:359C=0)
        mov ah,50h
        int 21h                         ; DOS Services   ah=function 50h
                                        ;   set active PSP segmnt from bx
        mov si,3608h                    ;start address of break points
        mov cx,ds:data_236e             ; number of break points
        jcxz loc_233
        push es
locloop _ 232:                               ; 恢复所有断点内容
        les di,dword ptr [si]                ; Load 32 bit ptr
        add si,4
        movsb                                ; Mov [si] to es: [di]
        loop locloop _ 232                   ; Loop if cx > 0
        pop es
loc _ 233:
        dec word ptr ds:data _ 237e          ; (6710:35AF=0)
        jz loc _ 234                         ; Jump if zero
        jmp loc _ 227                        ; 处理下一步
loc _ 234:
        call sub _ 22                        ; printf("\n")
        call sub _ 53                        ; 显示寄存器内容和指令列表
        jmp loc _ 39                         ; goto command _ accept
        sub _ 75   endp

;command I
;function  : inport
;format  : I <Port>
        sub _ 77   proc near
        mov cx,4
        call sub _ 36                        ; xatoi()
        call sub _ 42
        in al,dx
        push cs
        pop es
        mov di,37A2h
        call sub _ 23                        ; es: [di] <== xitoa()
        xor al,al
        stosb
        mov dx,37F2h
        jmp loc _ 56                         ; printf(" %s\n", ...)
        sub _ 77   endp

;command O
;function  : outport
;format  : O <port> <value>
        sub _ 78   proc near
        mov cx,4
```

```
          call sub _ 36                     ; get port
          push dx
          mov cx,2
          call sub _ 36                     ; get value
          call sub _ 42                     ; (06AF)
          xchg ax,dx
          pop dx
          out dx,al
loc _ ret _ 235:
          retn
          sub _ 78    endp

;get start address : seg, ofs for P, G, T
          sub _ 79    proc near
          mov dx,cs _ save                  ; set default value
          mov work _ seg,dx
          mov dx,ip _ save
          mov work _ ofs,dx
          mov bp,cs _ save
          call sub _ 13
          cmp byte ptr [si] ,3Dh            ; '='
          jne loc _ ret _ 235
          inc si
          call sub _ 44                     ; get address
          mov work _ seg,ax                 ; (6710:31F7=0)
          mov work _ ofs,dx                 ; (6710:31F9=0)
          retn
          sub _ 79    endp

;command G
;function ; execute program
;format ; G [=address] [address ... ]
          sub _ 80    proc near
          mov byte ptr data _ 176,0         ;initial
          call sub _ 79                     ; get start address
          or bx,bx
          mov di,3608h
loc _ 236:
          call sub _ 13
          jz loc _ 237
          mov bp,cs _ save
          push di
          call sub _ 44                 ; get the address of break points
          pop di
          mov [di] ,dx                  ;save break points
          mov [di+2] ,ax
          add di,5
          inc bx
          cmp bx,0Bh
          jne loc _ 236                 ; max break points is 10
          mov dx,33D0h
          jmp loc _ 139                 ; (09EA)
loc _ 237:
          mov ds:data _ 236e,bx
          mov cx,bx
          jcxz loc _ 239                ; Jump if no break points
          mov di,3608h
          push ds
locloop _ 238:
          lds si,dword ptr es: [di]
          add di,4
          movsb                         ; Mov [si] to es: [di]
          mov byte ptr [si-1] ,0CCh     ;Insert Int0x03
          loop locloop _ 238            ; Loop if cx > 0
          pop ds
loc _ 239:
          mov dx,work _ seg
          mov cs _ save,dx
          mov dx,work _ ofs
          mov ip _ save,dx
          mov word ptr ds:data _ 237e,1 ; (6710:35AF=0)
          jmp loc _ 226                 ; (1153)
          sub _ 80   endp

;scan char [si] until [si++] is delimiter
          sub _ 81   proc near
          mov ah,37h
          int 21h                       ; DOS Services   ah=function 37h, get switch char
          mov cs:data _ 212e,dl         ; (6710:3576=0)
loc _ 240:
        lodsb
        call sub _ 64               ; (0AF2)
        jz loc _ 241
        call sub _ 65               ; (0B05)
        jnz loc _ 240
loc _ 241:
        dec si
loc _ ret _ 242:
        retn
        sub _ 81    endp

;command C
;function : compare
;format : C <range> <address>
sub _ cmd _ C:
        call sub _ 28               ; (046D),   get range
        push cx
        push ax
        push dx
        call sub _ 44               ; (06CE), get address
        call sub _ 42               ; (06AF)
        pop si
        mov di,dx
        mov es,ax
        pop ds
        pop cx
        dec cx
        call sub _ 82               ; (138C)
        inc cx

;compare ds: [si] , es: [di] , len=cx and print nomatched bytes
        sub _ 82    proc near
loc _ 243:
        repe cmpsb                  ; Rept zf=1+cx>0 Cmp [si] to es: [di]
        jz loc _ ret _ 242          ; Jump if Ok
        dec si
        mov cs:data _ 269e,ds       ; (6710:3914=0), source seg
        mov cs:data _ 270e,si       ; (6710:3916=0), source ofs
        xor ah,ah
        lodsb                       ; String [si] to al
        mov cs:data_271e,ax          ; (6710:3918=0), source byte
        dec di
        mov al,es:[di]
        mov cs:data_272e,ax          ; (6710:391A=0), target byte
        mov cs:data_273e,es          ; (6710:391C=0), target seg
        mov cs:data_274e,di          ; (6710:391E=0), target ofs
        inc di
        push ds
        push cs
        pop ds
        mov dx,3912h    ;printf("%04X:%04X  %2X %2X  %04X:%04X", ...)
        call sub_21                  ; (03E1), 打印不匹配单元
        pop ds
        xor al,al                    ; Zero register
        jmp short loc_243            ; (138C)
        sub_82  endp

;if dx == 0 then goto illegal command
        sub_83  proc near
        or dx,dx                     ; Zero ?
        jnz loc_ret_242              ; Jump if not zero
        mov dx,32FAh
        jmp loc_97                   ; (06B5)
        sub_83  endp

;command A
;function : assumble
;format : A [address]
sub_cmd_A:
        mov bp,cs_save
        mov di,3580h
        call sub_30                  ;get address
        mov ds:data_216e,dx
        mov ds:data_217e,ax
        mov ds:data_221e,sp
loc_244:
        mov sp,ds:data_221e
        les di,dword ptr ds:data_216e
        call sub_17                  ; (03C9)
        push cs
        pop es
        push di
        mov di,37A2h
        xor al,al
        stosb
        mov dx,3861h                ;print("%04X:%04X  %s",...), 打印地址
        call sub _ 20               ; (03DA)
        pop di
        call sub _ 11               ; (0340), get KBDline
        call sub _ 14               ; (0388)
        jnz loc _ 245
        retn
loc _ 245:
        xor cx,cx
        mov di,2829h                ;指令串表首址
loc _ 246:
        xor bx,bx
loc _ 247:
        mov al, [bx+di]
        cmp al, [bx+si]
        je loc _ 249                ; Jump if equal
        inc cx
        cmp cx,0C1h                 ;一共 193 条指令
        jb loc _ 248
        jmp loc _ 312               ; Error, 重新输入
loc _ 248:
        inc di
        cmp byte ptr [di-1] ,0      ;本条指令结束 ?
        jne loc _ 248               ; Jump if no
        jmp short loc _ 246
loc _ 249:
        inc bx
        cmp byte ptr [bx+di] ,0     ;指令串结束 ?
        jne loc _ 247               ; Jump if no,继续比较
        xchg bx,cx
        mov ax,bx                   ;bx=第 n 条指令编号,cx=指令串长度
        shl ax,1                    ; Shift w/zeros fill
        add ax,bx
        add ax,2D83h                ;指令子程序表首址
        mov bx,ax
        xor ax,ax
        mov ds:data_230e,al
        mov ds:data_252e,ax
        mov ds:data_254e,al
        mov ah,0Ah
        mov al,[bx]
        mov ds:data_239e,ax
        mov byte ptr ds:data_238e,1
        add si,cx                       ;[si]<== 指令串下一字符位置
        jmp word ptr [bx+1]             ;转相应指令处理程序
        mov ah,0DEh
        jmp short loc_250               ;(145F)
        mov ah,0DBh
        jmp short loc_250               ;(145F)
        mov ah,0D9h
loc_250:
        xchg al,ah
        mov ds:data_239e,ax
        inc byte ptr ds:data_238e
        call sub_90                     ;(1CC0)
        call sub_13                     ;skip SPACE, TAB, ','
        push cs
        pop es
        jnz loc_245
        jmp loc_244                     ;(13E2)
        mov ah,0FFh
        jmp short loc_251               ;(147B)
        mov ah,8Fh
loc_251:
        mov ds:data_239e,ah
        mov ds:data_232e,al
        inc byte ptr ds:data_252e
        mov byte ptr ds:data_230e,2
        call sub_87
        call sub_84
        mov al,[di+2]
        cmp al,0C0h
        jb loc_254
        mov byte ptr [di],1
        cmp byte ptr ds:data_252e,2 ;(6710:3642=0)
        jne loc_252
        and al,18h
        or al,6
        cmp byte ptr ds:data_232e,0   ; (6710:35A9=0)
        jne loc_253
        or al,1
        jmp short loc_253             ; (14BE)
loc_252:
        and al,7
        or al,50h
        cmp byte ptr ds:data_232e,0   ; (6710:35A9=0)
        jne loc_253
        or al,58h
loc_253:
        mov [di+1] ,al
        jmp loc_311
        call sub_14                   ;skip SPACE, TAB
        mov cx,4
        call sub_34                   ; [DX] <== xatoi( [si] )
        jc loc_254
        dec byte ptr ds:data_239e
        add byte ptr ds:data_238e,2
        mov ds:data_240e,dx
loc_254:
        jmp loc_311
        call sub_14                   ;skip space, TAB
        mov cx,2
        call sub_34                   ; xatoi()
        jc loc_255
        mov al,dl
        cmp al,3
        je loc_254
        inc byte ptr ds:data_239e
        jmp loc_266
        call sub_14                   ; skip space, tab
        lodsw                         ; String [si] to ax
        cmp ax,4C41h                  ;' AL'
        je loc_257
        cmp ax,5841h                  ;' AX'
        je loc_256
```

```
loc_255:
        jmp loc_312                 ;Error，重新输入
loc_256:
        inc byte ptr ds:data_239e  ; (6710:35B2=0)
loc_257:
        call sub_13                 ;skip space, tab, ' ,'
        cmp word ptr [si] ,5844h    ;' DX'
        je loc_254
        mov cx,2
        call sub_34                 ;xatoi()
        jc loc_255
        and byte ptr ds:data_239e,0F7h
        mov al,dl
        jmp loc_266                 ; (1618)
        call sub_14                 ;skip space, tab
        cmp word ptr [si] ,5844h    ;' DX'
        jne loc_258
        inc si
        inc si
        jmp short loc_259           ; (1549)
loc_258:
        and byte ptr ds:data_239e,0F7h    ; (6710:35B2=0)
        mov cx,2
        call sub_34                 ;xatoi()
        jc loc_255
        inc byte ptr ds:data_238e   ; (6710:35B1=0)
        mov ds:data_240e,dl         ; (6710:35B3=0)
loc_259:
        call sub_13                 ; skip space, tab, ' ,'
        lodsw                       ; String [si] to ax
        cmp ax,4C41h                ;' AL'
        je loc_254
        cmp ax,5841h                ;' AX'
        jne loc_255
        inc byte ptr ds:data_239e
        jmp loc_254
        inc byte ptr ds:data_253e
        mov byte ptr ds:data_239e,0FFh
        mov ds:data_232e,al
        call sub_87                 ; (1A44)
```

```
          call sub_86                    ; (1A03)
          cmp byte ptr ds:data_250e,0    ; (6710:3640=0)
          jne loc_260
          cmp byte ptr ds:data_231e,0FFh     ; (6710:35A8=0)
          je loc_262
loc_260:
          cmp byte ptr ds:data_230e,1    ; (6710:35A7=0)
loc_261:
          je loc_255
          cmp byte ptr ds:data_230e,4    ; (6710:35A7=0)
          jne loc_265
          or byte ptr [di+2] ,8
          jmp short loc_265              ; (1600)
loc_262:
          mov ax,ds:data_255e            ; (6710:3645=0)
          mov dx,ds:data_256e            ; (6710:3647=0)
          mov bl,ds:data_230e            ; (6710:35A7=0)
          cmp byte ptr ds:data_249e,0    ; (6710:363F=0)
          je loc_261
          mov byte ptr [di] ,5
          mov [di+2] ,ax
          mov [di+4] ,dx
          mov al,9Ah
          cmp byte ptr ds:data_253e,0    ; (6710:3643=0)
          je loc_263
          mov al,0EAh
loc_263:
          mov [di+1] ,al
          cmp bl,4
          je loc_265
          or bl,bl                       ; Zero ?
          jnz loc_264
          cmp dx,ds:data_217e            ; (6710:3582=0)
          jne loc_265
loc_264:
          mov byte ptr [di] ,3
          mov al,0E8h
          or al,ds:data_253e             ; (6710:3643=0)
          mov [di+1] ,al
          mov ax,ds:data_255e            ; (6710:3645=0)
```

```
        sub ax,ds:data _ 216e           ; (6710:3580=0)
        sub ax,3
        mov [di+2] ,ax
        cmp byte ptr ds:data _ 253e,0   ; (6710:3643=0)
        je loc _ 265
        cmp bl,2
        je loc _ 265
        inc ax
        mov cx,ax
        cbw                             ; Convrt byte to word
        cmp ax,cx
        jne loc _ 267
        mov byte ptr [di+1] ,0EBh
        mov [di+2] ,ax
        dec byte ptr [di]
loc _ 265:
        jmp loc _ 311                   ; (19C9)
        mov bp,ds:data _ 217e           ; (6710:3582=0)
        call sub _ 45                   ; get address
        sub dx,ds:data _ 216e           ; (6710:3580=0)
        dec dx
        dec dx
        call sub _ 93                   ; (1CF5)
        cmp cl,1
        jne loc _ 269                   ; Error ! 重输
loc _ 266:
        inc byte ptr ds:data _ 238e     ; (6710:35B1=0)
        mov ds:data _ 240e,al           ; (6710:35B3=0)
loc _ 267:
        jmp loc _ 311
        call sub _ 14                   ; skip space, tab
        lodsw                           ; String [si] to ax
        mov cx,8
        mov di,offset reg _ x _ str
        call sub _ 89                   ; (1CB6)
        jz loc _ 269
        shl al,1
        shl al,1
        shl al,1
        mov ds:data _ 232e,al           ; (6710:35A9=0)
        call sub_13
        call sub_87
        cmp byte ptr ds:data_230e,0   ; (6710:35A7=0)
        jne loc_269                   ; Error ! 重输
        call sub_85
        jmp short loc_268             ; (1670)
        mov byte ptr ds:data_239e,0FEh   ; (6710:35B2=0)
        mov ds:data_232e,al           ; (6710:35A9=0)
        call sub_87
        call sub_84
        test byte ptr [di+1] ,1
        jz loc_268
        mov al, [di+2]
        cmp al,0C0h
        jb loc_268
        and al,0Fh
        or al,40h
        mov [di+1] ,al
        dec byte ptr [di]
loc_268:
        jmp loc_311
loc_269:
        jmp loc_312                   ; Error ! 重输
        inc byte ptr ds:data_230e   ; (6710:35A7=0)
        call sub_14
        mov cx,2
        call sub_34                   ; xatoi()
        cmp dx,40h                    ; dx >= 64 ?
        jae loc_269                   ; yes
        call sub_13
        mov ax,dx
        mov cl,3
        shr dx,cl
        or ds:data_239e,dl   ; (6710:35B2=0)
        and al,7
        shl al,cl
        jmp loc_276
        call sub_96
        call sub_88
        call sub_86
```

15.8-70

```
        cmp byte ptr ds:data_233e,0C0h     ; (6710:35AA=0)
        jne loc_271
        mov al,ds:data_258e            ; (6710:364A=0)
        or al,al
        jz loc_272
        or [di+1] ,al
        xor byte ptr [di+2] ,8
        jmp short loc_272
        call sub_96
        mov byte ptr ds:data_258e,0    ; (6710:364A=0)
        jmp short loc_270
        call sub_96
loc_270:
        call sub_88
        call sub_86
        cmp byte ptr ds:data_233e,0C0h     ; (6710:35AA=0)
        jne loc_271
        mov al,ds:data_258e            ; (6710:364A=0)
        or [di+1] ,al
        jmp short loc_272              ; (16E1)
loc_271:
        call sub_97
loc_272:
        jmp loc_311                    ; (19C9)
        mov ah,5
        jmp short loc_273              ; (16EE)
        mov ah,2
        jmp short loc_273              ; (16EE)
        mov ah,0FFh
loc_273:
        mov ds:data_230e,ah            ; (6710:35A7=0)
        call sub_96
        call sub_87
        cmp byte ptr ds:data_233e,0C0h     ; (6710:35AA=0)
        je loc_275                     ; Error ! 重输
loc_274:
        call sub_86
        jmp short loc_272              ; (16E1)
        mov byte ptr ds:data_230e,0FFh     ; (6710:35A7=0)
        call sub_96
```

7A8-7U

```
        call sub _ 88
        cmp byte ptr ds:data _ 258e,0  ; (6710:364A=0)
        jne loc _ 274
loc _ 275:
        jmp loc _ 312                  ; (19CF), Error ! 重输
        call sub _ 96                  ; (1D1D)
        mov byte ptr ds:data _ 258e,0  ; (6710:364A=0)
        call sub _ 87                  ; (1A44)
        cmp byte ptr ds:data _ 233e,0C0h   ; (6710:35AA=0)
        je loc _ 275                   ; Error ! 重输
        call sub _ 86                  ; (1A03)
        call sub _ 97                  ; (1D46)
        jmp short loc _ 272            ; (16E1)
        mov byte ptr ds:data _ 239e,0F6h   ; (6710:35B2=0)
loc _ 276:
        mov ds:data _ 232e,al          ; (6710:35A9=0)
        call sub _ 87                  ; (1A44)
        call sub _ 84                  ; (19E7)
        jmp short loc _ 272            ; (16E1)
        mov byte ptr ds:data _ 239e,0D0h   ; (6710:35B2=0)
        mov ds:data _ 232e,al          ; (6710:35A9=0)
        call sub _ 87                  ; (1A44)
        call sub _ 84                  ; (19E7)
        call sub _ 13                  ; (037F)
        cmp byte ptr [si] ,31h         ; ' 1'
        je loc _ 278
        cmp word ptr [si] ,4C43h       ; ' CL'
        je loc _ 277
        jmp loc _ 312
loc _ 277:
        or byte ptr ds:data _ 239e,2   ; (6710:35B2=0)
loc _ 278:
        jmp loc _ 311
        inc byte ptr ds:data _ 253e    ; (6710:3643=0)
        inc byte ptr ds:data _ 253e    ; (6710:3643=0)
        jmp short loc _ 279            ; (1778)
        inc byte ptr ds:data _ 252e    ; (6710:3642=0)
loc _ 279:
        xor ax,ax
        jmp short loc _ 280            ; (1781)
        mov byte ptr ds:data _ 239e,80h    ; (6710:35B2=0)
loc _ 280:
        mov ds:data _ 232e,al            ; (6710:35A9=0)
        push ax
        call sub _ 87
        call sub _ 85
        call sub _ 13
        mov al,ds:data _ 238e            ; (6710:35B1=0)
        push ax
        call sub _ 87
        pop ax
        mov [di] ,al
        pop ax
        mov bl,ds:data _ 230e            ; (6710:35A7=0)
        or bl,bl
        jz loc _ 281                     ;Error ! 重输
        dec bl
        and bl,1
        or [di+1] ,bl
        cmp byte ptr ds:data _ 250e,0   ; (6710:3640=0)
        jne loc _ 282
        cmp byte ptr ds:data _ 249e,0   ; (6710:363F=0)
        je loc _ 282
        cmp byte ptr ds:data _ 254e,0   ; (6710:3644=0)
        jne loc _ 281                    ;Error ! 重输
        cmp byte ptr ds:data _ 253e,2   ; (6710:3643=0)
        jne loc _ 283
loc _ 281:
        jmp loc _ 312                    ;Error ! 重输
loc _ 282:
        jmp loc _ 292
loc _ 283:
        mov al, [di+2]
        cmp byte ptr ds:data _ 252e,0   ; (6710:3642=0)
        je loc _ 285
        and al,0C0h
        cmp al,0C0h
        jne loc _ 289
        mov al, [di+1]
        and al,1
```

```
            pushf
            shl al,1
            shl al,1
            shl al,1
            or al, [di+2]
            and al,0Fh
            or al,0B0h
            mov [di+1] ,al
            mov ax,ds:data_255e          ; (6710:3645=0)
            mov [di+2] ,ax
            popf
            jz loc_284
            inc byte ptr [di]
loc_284:
            jmp loc_310                  ; (199F)
loc_285:
            and al,0C7h
            cmp al,0C0h
            je loc_286
            cmp byte ptr ds:data_253e,0  ; (6710:3643=0)
            jne loc_289
            cmp byte ptr ds:data_232e,8  ; (6710:35A9=0)
            je loc_289
            cmp byte ptr ds:data_232e,20h    ; (6710:35A9=0) ' '
            je loc_289
            cmp byte ptr ds:data_232e,30h  ; (6710:35A9=0) '0'
            je loc_289
            test byte ptr [di+1] ,1
            jz loc_289
            mov ax,ds:data_255e          ; (6710:3645=0)
            mov bx,ax
            cbw
            cmp ax,bx
            jne loc_289
            mov bl, [di]
            dec byte ptr [di]
            or byte ptr [di+1] ,2
            jmp short loc_290            ; (1858)
loc_286:
            mov al, [di+1]
```

```
          and al,1
          cmp byte ptr ds:data _ 253e,0   ; (6710:3643=0)
          je loc _ 287
          or al,0A8h
          jmp short loc _ 288             ; (1851)
loc _ 287:
          or al,ds:data _ 232e            ; (6710:35A9=0)
          or al,4
loc _ 288:
          mov [di+1] ,al
          dec byte ptr [di]
loc _ 289:
          mov bl, [di]
loc _ 290:
          xor bh,bh
          add bx,di
          inc bx
          mov ax,ds:data _ 255e           ; (6710:3645=0)
          mov [bx] ,ax
          inc byte ptr [di]
          test byte ptr [di+1] ,1
          jz loc _ 291
          inc byte ptr [di]
loc _ 291:
          jmp loc _ 310                   ; (199F)
loc _ 292:
          cmp byte ptr ds:data _ 254e,0   ; (6710:3644=0)
          je loc _ 295
          mov al,ds:data _ 231e           ; (6710:35A8=0)
          test al,10h
          jz loc _ 294
loc _ 293:
          jmp loc _ 312                   ; (19CF), Error ! 重输
loc _ 294:
          and al,7
          or [di+2] ,al
          and byte ptr [di+1] ,0FEh
          cmp byte ptr ds:data _ 250e,0   ; (6710:3640=0)
          jne loc _ 299
          jmp loc _ 310
```

```
loc_295:
        and byte ptr [di+2],0C7h
        mov al,[di+1]
        and al,1
        cmp byte ptr ds:data_252e,0   ; (6710:3642=0)
        je loc_296                    ; Jump if equal
        or al,88h
        jmp short loc_298             ; (18BD)
loc_296:
        cmp byte ptr ds:data_253e,0   ; (6710:3643=0)
        je loc_297                    ; Jump if equal
        or al,84h
        cmp byte ptr ds:data_253e,2   ; (6710:3643=0)
        jne loc_297                   ; Jump if not equal
        or al,2
loc_297:
        or al,ds:data_232e            ; (6710:35A9=0)
loc_298:
        mov [di+1],al
        cmp byte ptr ds:data_250e,0   ; (6710:3640=0)
loc_299:
        je loc_300
        jmp loc_304
loc_300:
        mov al,ds:data_231e           ; (6710:35A8=0)
        test al,10h
        jz loc_301
        cmp byte ptr ds:data_252e,0   ; (6710:3642=0)
        je loc_293                    ; Error ! 重输
        mov byte ptr [di+1],8Ch
loc_301:
        and al,7
        shl al,1                      ; Shift w/zeros fill
        shl al,1
        shl al,1
        or [di+2],al
        cmp byte ptr ds:data_253e,0   ; (6710:3643=0)
        je loc_302
        mov ah,[di+2]
        and ah,0C0h
```

```
          cmp ah,0C0h
          jne loc_302
          mov ah,[di+2]
          and ah,7
          shl ah,1                  ; Shift w/zeros fill
          shl ah,1
          shl ah,1
          mov al,[di+2]
          and al,38h                ; '8'
          shr al,1
          shr al,1
          shr al,1                  ; Shift w/zeros fill
          or al,ah
          and byte ptr [di+2],0C0h
          or [di+2],al
loc_302:
          cmp byte ptr ds:data_253e,2 ; (6710:3643=0)
          jne loc_310
          test byte ptr [di+1],1
          jz loc_310
          push ax
          mov al,[di+2]
          and al,0C0h
          cmp al,0C0h
          pop ax
          jc loc_310
          or al,al
          jz loc_303
          mov al,[di+2]
          and al,7
          jnz loc_310
          mov cl,3
          shr byte ptr [di+2],cl
loc_303:
          mov al,[di+2]
          and al,7
          or al,90h
          mov [di+1],al
          dec byte ptr [di]
          jmp short loc_310         ; (199F)
```

```
loc_304:
        cmp byte ptr ds:data_253e,0     ; (6710:3643=0)
        jne loc_305
        or byte ptr [di+1],2
loc_305:
        mov al,[di+2]
        cmp al,0C0h
        jb loc_312                      ; Error ! 重输
        cmp byte ptr ds:data_254e,0     ; (6710:3644=0)
        je loc_306
        and al,18h
        jmp short loc_307               ; (1974)
loc_306:
        and al,7
        shl al,1                        ; Shift w/zeros fill
        shl al,1
        shl al,1
loc_307:
        or al,ds:data_233e              ; (6710:35AA=0)
        or al,ds:data_231e              ; (6710:35A8=0)
        mov [di+2],al
        mov ax,ds:data_255e             ; (6710:3645=0)
        mov [di+3],ax
        mov byte ptr [di],2
        mov al,[di+2]
        and al,0C7h
        cmp al,6
        je loc_308
        and al,0C0h
        cmp al,40h
        je loc_309
        cmp al,80h
        jne loc_310
loc_308:
        inc byte ptr [di]
loc_309:
        inc byte ptr [di]
loc_310:
        cmp byte ptr ds:data_252e,0     ; (6710:3642=0)
        je loc_311
        mov al, [di+1]
        and al,0FCh
        cmp al,88h
        jne loc_311
        cmp byte ptr [di+2] ,6
        jne loc_311
        mov al, [di+1]
        and al,3
        xor al,2
        or al,0A0h
        mov [di+1] ,al
        dec byte ptr [di]
        mov ax, [di+3]
        mov [di+2] ,ax
loc_311:
        call sub_90                     ; (1CC0)
        jmp loc_244                     ; (13E2)
loc_312:                                ; Error ! 重输
        sub si,35AEh                    ; 确定错误位置
        mov cx,si
        mov di,37A2h
        call sub_27                     ; (041A)
        mov byte ptr [di] ,0
        mov dx,32FAh                    ; printf("%s ^ Error\n", ...
        call sub_21                     ; (03E1)
        jmp loc_244                     ; (13E2)

        sub_84   proc near
        mov al,ds:data_230e             ; (6710:35A7=0)
        or al,al
        jnz loc_314
loc_313:
        jmp short loc_312               ; (19CF),Error ! 重输
loc_314:
        dec al
        or [di+1] ,al

        sub_85:
        cmp byte ptr ds:data_249e,0   ; (6710:363F=0)
        je loc_315
        cmp byte ptr ds:data _ 250e,0  ; (6710:3640=0)
        je loc _ 313                   ; Error ! 重输

        sub _ 86:
loc _ 315:
        mov al,ds:data _ 231e          ; (6710:35A8=0)
        cmp al,0FFh
        je loc _ 315
        test al,10h
        jz loc _ 316
        cmp byte ptr ds:data _ 252e,0  ; (6710:3642=0)
        je loc _ 313                   ; Error ! 重输
        mov word ptr [di+1] ,8Eh
        inc byte ptr ds:data _ 252e    ; (6710:3642=0)
        inc byte ptr ds:data _ 254e    ; (6710:3644=0)
        and al,3
        shl al,1                       ; Shift w/zeros fill
        shl al,1
        shl al,1
        or al,0C0h
        mov [di+2] ,al
        retn
loc _ 316:
        and al,7
        or al,ds:data _ 233e           ; (6710:35AA=0)
        or al,ds:data _ 232e           ; (6710:35A9=0)
        mov [di+2] ,al
        mov ax,ds:data _ 255e          ; (6710:3645=0)
        mov [di+3] ,ax
        retn
        sub _ 84   endp

        sub _ 87   proc near
        mov byte ptr ds:data _ 257e,0  ; (6710:3649=0)

        sub _ 88:
        call sub _ 13                  ; (037F)
        xor ax,ax
        mov ds:data _ 255e,ax          ; (6710:3645=0)
        mov ds:data _ 244e,ax          ; (6710:363A=0)
        mov ds:data _ 246e,ax           ; (6710:363C=0)
        mov ds:data _ 248e,ax           ; (6710:363E=0)
        mov ds:data _ 250e,ax           ; (6710:3640=0)
        dec al
        cmp byte ptr ds:data _ 257e,0   ; (6710:3649=0)
        je loc _ 317
        mov al,1
loc _ 317:
        mov ds:data _ 231e,al           ; (6710:35A8=0)
loc _ 318:
        mov byte ptr ds:data _ 248e,0   ; (6710:363E=0)
loc _ 319:
        mov ax, [si]
        cmp al,2Ch                      ; ','
        je loc _ 322
        cmp al,0Dh                      ; RET
        je loc _ 322
        cmp al,3Bh                      ;''
        je loc _ 322
        cmp al,9
        je loc _ 320
        cmp al,20h                      ; ' '
        jne loc _ 321
loc _ 320:
        inc si
        jmp short loc _ 319             ; (1A70),Error ! 重输
loc _ 321:
        jmp loc _ 330                   ; (1B4D)
loc _ 322:
        mov di,35B1h
        mov byte ptr ds:data _ 233e,0C0h    ; (6710:35AA=0)
        mov byte ptr ds:data _ 238e,2   ; (6710:35B1=0)
        cmp byte ptr ds:data _ 250e,0   ; (6710:3640=0)
        jne loc _ 325                   ; Jump if not equal
        mov al,ds:data _ 249e           ; (6710:363F=0)
        or al,ds:data _ 251e            ; (6710:3641=0)
        jnz loc _ ret _ 323             ; Jump if not zero
        or al,ds:data _ 257e            ; (6710:3649=0)
        jz loc _ 324                    ; Jump if zero, Error ! 重输
        mov al, [di+1]
```

```
          or al,ds:data_258e            ; (6710:364A=0)
          cmp al,0DCh
          jne loc_ret_323               ; Jump if not equal
          mov byte ptr [di+1],0DEh
loc_ret_323:
          retn
loc_324:
          jmp loc_312                   ; (19CF), Error ! 重输
loc_325:
          mov byte ptr ds:data_233e,0    ; (6710:35AA=0)
          cmp byte ptr ds:data_249e,0   ; (6710:363F=0)
          je loc_327                    ; Jump if equal
          mov byte ptr [di],4
          mov ax,ds:data_244e           ; (6710:363A=0)
          or ax,ds:data_246e            ; (6710:363C=0)
          jnz loc_326                   ; Jump if not zero
          mov byte ptr ds:data_231e,6   ; (6710:35A8=0)
          retn
loc_326:
          mov byte ptr ds:data_233e,80h    ; (6710:35AA=0)
          call sub_94                   ; (1CFB)
          cmp cl,2
          je loc_327                    ; Jump if equal
          dec byte ptr [di]
          mov byte ptr ds:data_233e,40h    ; (6710:35AA=0) '@'
loc_327:
          mov bx,ds:data_246e           ; (6710:363C=0)
          mov cx,ds:data_244e           ; (6710:363A=0)
          xor dx,dx                     ; Zero register, dl=0
          mov al,bl
          add al,ch
          cmp al,2
          je loc_329
          inc dl                        ;dl=1
          mov al,bl
          add al,cl
          cmp al,2
          je loc_329
          inc dl                        ;dl=2
          mov al,bh
```

```
        add al,ch
        cmp al,2
        je loc _ 329
        inc dl                          ;dl=3
        mov al,bh
        add al,cl
        cmp al,2
        je loc _ 329
        inc dl                          ;dl=4
        or ch,ch
        jnz loc _ 329
        inc dl                          ;dl=5
        or cl,cl
        jnz loc _ 329
        inc dl                          ;dl=6
        or bh,bh
        jz loc _ 328
        cmp byte ptr ds:data _ 233e,0   ; (6710:35AA=0)
        jne loc _ 329
        mov byte ptr ds:data _ 233e,40h    ; (6710:35AA=0) '@'
        inc byte ptr [di]
        dec dl
loc _ 328:
        inc dl
loc _ 329:
        mov ds:data _ 231e,dl           ; (6710:35A8=0)
        retn

loc _ 330:
        cmp ax,454Eh                    ;'NE'
        jne loc _ 333
        mov dl,2
loc _ 331:
        call sub _ 92                   ; (1CDF)
loc _ 332:
        call sub _ 95                   ; (1D07)
        mov ax, [si]
        cmp ax,5450h                    ; 'PT'
        je loc _ 332
        jmp loc _ 318                   ; (1A6B)
```

```
loc _ 333:
        mov cx,5
        mov di,241Dh
        call sub _ 89                ; (1CB6)
        jz loc _ 334
        inc ai
        mov dl,al
        jmp short loc _ 331          ; (1B54)
loc _ 334:
        mov ax, [si]
        cmp byte ptr ds:data _ 257e,0  ; (6710:3649=0)
        je loc _ 335
        cmp ax,5453h                 ; 'ST'
        jne loc _ 335
        cmp byte ptr [si+2] ,2Ch     ; ','
        jne loc _ 335
        mov byte ptr ds:data _ 258e,0  ; (6710:364A=0)
        add si,3
        jmp loc _ 318                ; (1A6B)
loc _ 335:
        cmp ax,4853h                 ; 'SM'
        je loc _ 332
        cmp ax,4146h                 ; 'FA'
        jne loc _ 336
        cmp byte ptr [si+2] ,52h     ; 'R'
        jne loc _ 336
        add si,3
        mov dl,4
        jmp short loc _ 331          ; (1B54)
loc _ 336:
        cmp al,5Bh                   ; '['
        jne loc _ 339
loc _ 337:
        inc byte ptr ds:data _ 250e  ; (6710:3640=0)
loc _ 338:
        inc si
        jmp loc _ 318                ; (1A6B)
loc _ 339:
        cmp al,5Dh                   ; ']'
        je loc _ 337
        cmp al,2Eh                  ; '.'
        je loc _ 337
        cmp al,2Bh                  ; '+'
        je loc _ 338
        cmp al,2Dh                  ; '—'
        jne loc _ 340
        inc byte ptr ds:data _ 248e ; (6710:363E=0)
        inc si
        jmp loc _ 319               ; (1A70)
loc _ 340:
        cmp byte ptr ds:data _ 257e,0 ; (6710:3649=0)
        je loc _ 342
        cmp ax,5453h                ; 'ST'
        jne loc _ 342
        cmp byte ptr [si+2] ,28h    ; '('
        jne loc _ 342
        cmp byte ptr [si+4] ,29h    ; ')'
        jne loc _ 345               ; Error ! 重输
        mov al, [si+3]
        sub al,30h                  ; '0'
        jc loc _ 345                ; '0'<, Error ! 重输
        cmp al,7
        ja loc _ 345                ; >'7' , Error ! 重输
        mov ds:data _ 231e,al       ; (6710:35A8=0)
        inc byte ptr ds:data _ 251e ; (6710:3641=0)
        add si,5
        cmp word ptr [si] ,532Ch
        jne loc _ 341
        cmp byte ptr [si+2] ,54h    ; 'T'
        jne loc _ 341
        add si,3
loc _ 341:
        jmp loc _ 318               ; (1A6B)
loc _ 342:
        mov cx,14h
        mov di,23F3h
        call sub _ 89               ; (1CB6)
        jz loc _ 350
        mov ds:data _ 231e,al       ; (6710:35A8=0)
        inc byte ptr ds:data _ 251e ; (6710:3641=0)
        cmp byte ptr ds:data _ 250e,0  ; (6710:3640=0)
        jne loc _ 344
        call sub _ 91                  ; (1CD7)
loc _ 343:
        add si,2
        jmp loc _ 318                  ; (1A6B)
loc _ 344:
        cmp al,0Bh
        jne loc _ 347
        cmp word ptr ds:data _ 246e,0  ; (6710:363C=0)
        je loc _ 346
loc _ 345:
        jmp loc _ 312                  ; (19CF), Error ! 重输
loc _ 346:
        inc byte ptr ds:data _ 246e    ; (6710:363C=0)
        jmp short loc _ 343            ; (1C2A)
loc _ 347:
        cmp al,0Dh
        jne loc _ 348
        cmp word ptr ds:data _ 246e,0  ; (6710:363C=0)
        jne loc _ 345                  ; Error ! 重输
        inc byte ptr ds:data _ 247e    ; (6710:363D=0)
        jmp short loc _ 343            ; (1C2A)
loc _ 348:
        cmp al,0Eh
        jne loc _ 349
        cmp word ptr ds:data _ 244e,0  ; (6710:363A=0)
        jne loc _ 345                  ;-Error ! 重输
        inc byte ptr ds:data _ 245e    ; (6710:363B=0)
        jmp short loc _ 343            ; (1C2A)
loc _ 349:
        cmp al,0Fh
        jne loc _ 345                  ; Error ! 重输
        cmp word ptr ds:data _ 244e,0  ; (6710:363A=0)
        jne loc _ 345
        inc byte ptr ds:data _ 244e    ; (6710:363A=0)
        jmp short loc _ 343            ; (1C2A)
loc _ 350:
        mov bp,ds:data _ 217e          ; (6710:3582=0)
        cmp byte ptr ds:data _ 250e,0  ; (6710:3640=0)
```

8/3-8-10

```
        je loc _ 353
loc _ 351:
        mov cx,4
loc _ 352:
        call sub _ 34                     ; (06F1)
        jmp short loc _ 354               ; (1C9D)
loc _ 353:
        mov cx,2
        cmp byte ptr ds:data _ 230e,1     ; (6710:35A7=0)
        je loc _ 352
        cmp ds:data _ 230e,cl             ; (6710:35A7=0)
        je loc _ 351
        call sub _ 45                     ; (06D5)
loc _ 354:                                ;  Error ! 重输
        jc loc _ 345
        mov ds:data _ 256e,ax             ; (6710:3647=0)
        cmp byte ptr ds:data _ 248e,0     ; (6710:363E=0)
        je loc _ 355
        neg dx
loc _ 355:
        add ds:data _ 255e,dx             ; (6710:3645=0)
        inc byte ptr ds:data _ 249e       ; (6710:363F=0)
        jmp loc _ 318                     ; (1A6B)
        sub _ 87   endp

;strnchr()
        sub _ 89   proc near
        push cx
        inc cx
        repne scasw                       ; Rept zf=0+cx>0 Scan es:[di] for ax
        pop ax
        sub ax,cx
        or cx,cx                          ; Zero ?
        retn
        sub _ 89   endp

;strncpy()
        sub _ 90   proc near
        push si
        les di,dword ptr ds:data _ 216e   ; (66E8:3580=6F72h) Load 32 bit ptr
```

```
          mov si,35B1h
          xor ax,ax
          lodsb
          mov cx,ax
          jcxz loc _ 356            ; Jump if cx=0
          rep movsb                 ; Rep while cx>0 Mov [si] to es: [di]
          mov ds:data _ 216e,di     ; (66E8:3580=6F72h)
loc _ 356:
          pop si
          retn
          sub _ 90   endp

          sub _ 91   proc near
          mov dl,1
          test al,18h
          jz loc _ 357              ; Jump if zero
          inc dl

          sub _ 92:
loc _ 357:
          cmp byte ptr ds:data _ 230e,0  ; (6710:35A7=0)
          je loc _ 358
          cmp ds:data _ 230e,dl     ; (6710:35A7=0)
          je loc _ 358
          pop dx
          jmp loc _ 345             ; (1C3B), Error ! 重输
loc _ 358:
          mov ds:data _ 230e,dl     ; (6710:35A7=0)
          retn
          sub _ 91   endp

          sub _ 93   proc near
          mov cl,4
          cmp ax,bp
          jne loc _ ret _ 359       ; Jump if not equal

          sub _ 94:
          mov cl,2
          mov ax,dx
          cbw
```

```
        cmp ax,dx
        jne loc_ret_359
        dec cl
loc_ret_359:
        retn
        sub_93  endp

        sub_95  proc near
loc_360:
        cmp byte ptr [si] ,0Dh         ;RET
        je loc_ret_359
        cmp byte ptr [si] ,5Bh         ; ' ['
        je loc_ret_359
        lodsb
        cmp al,20h                     ; ' '
        je loc_361
        cmp al,9
        jne loc_360
loc_361:
        jmp loc_53                     ; (0388)
        sub_95  endp

        sub_96  proc near
        mov byte ptr ds:data_239e,0D8h    ; (6710:35B2=0)
        mov ah,al
        and al,7
        shl al,1                       ; Shift w/zeros fill
        shl al,1
        shl al,1
        mov ds:data_232e,al            ; (6710:35A9=0)
        mov al,ah
        shr al,1
        shr al,1
        shr al,1
        or ds:data_239e,al             ; (6710:35B2=0)
        mov byte ptr ds:data_257e,1    ; (6710:3649=0)
        mov byte ptr ds:data_258e,4    ; (6710:364A=0)
        retn
        sub_96  endp
```

```
          sub_97   proc near
          mov al,ds:data_230e          ; (6710:35A7=0)
          test byte ptr [di+1] ,2
          jnz loc_362
          and byte ptr [di+1] ,0F9h
          cmp al,3
          je loc_ret_365
          cmp al,4
          je loc_364
          test byte ptr [di+1] ,1
          jz loc_366                   ; Jump if zero, Error ! 重输
          cmp al,5
          je loc_363
          jmp short loc_366            ; (1D8C), Error ! 重输
loc_362:
          cmp al,3
          je loc_ret_365
          cmp al,2
          je loc_364
          test byte ptr [di+1] ,1
          jz loc_366                   ; Jump if zero, Error ! 重输
          cmp al,4
          jne loc_366                  ; Jump if not equal, Error ! 重输
          or byte ptr [di+1] ,7
loc_363:
          or byte ptr [di+1] ,3
          or byte ptr [di+2] ,28h      ; '('
          jmp short loc_ret_365        ; (1D8B)
loc_364:
          or byte ptr [di+1] ,4
loc_ret_365:
          retn
loc_366:                               ;  Error ! 重输
          jmp loc_345                  ; (1C3B)
          sub_97   endp
          mov bp,1
          jmp short loc_367            ; (1D96)
          xor bp,bp
loc_367:
          mov di,35B1h
```

```
        dec byte ptr [di]
        inc di
loc_368:
        xor bl,bl
        call sub_13                     ; (037F)
        jnz loc_370
loc_369:
        jmp loc_311                     ; (19C9)
loc_370:
        or bl,bl
        jnz loc_372
        mov bh, [si]
        cmp bh,27h                      ; '''
        je loc_371
        cmp bh,22h                      ; '"'
        jne loc_373
loc_371:
        inc si
        inc bl
loc_372:
        lodsb
        cmp al,0Dh
        je loc_369
        cmp al,bh
        je loc_368
        stosb
        inc byte ptr ds:data_238e       ; (6710:35B1=0)
        jmp short loc_372               ; (1DB9)
loc_373:
        mov cx,2
        cmp bp,0
        je loc_374
        mov cl,4
loc_374:
        push bx
        call sub_34                     ; (0611)
        pop bx
        jnc loc_375
        jmp loc_312                     ; (19CF),Error ! 重输
loc_375:
```

```
          mov ax,dx
          cmp bp,0
          je loc _ 376
          stosw
          inc byte ptr ds:data _ 238e      ; (6710:35B1=0)
          jmp short loc _ 377              ; (1DEC)
loc _ 376:
          stosb
loc _ 377:
          inc byte ptr ds:data _ 238e      ; (6710:35B1=0)
          jmp short loc _ 368              ; (1D9C)
          mov bp,ds:data _ 217e            ; (6710:3582=0)
          call sub _ 45                    ; (06D5)
          mov ds:data _ 216e,dx            ; (6710:3580=0)
          mov ds:data _ 217e,ax            ; (6710:3582=0)
          jmp loc _ 244                    ; (13E2)

;command U
;function  : unassumble
;format    : U [range]
sub _ cmd _ U:
          mov bp,cs _ save
          mov di,3584h
          mov cx,disp _ chars
          shr cx,1
          shr cx,1                         ;set default Len
          call sub _ 30                    ; get range
          mov ds:data _ 218e,dx            ; ofs
          mov ds:data _ 219e.ax            ; seg
          mov ds:data _ 220e.cx            ; len
loc _ 378:
          call sub _ 102                   ; (1E7E)
          call sub _ 22                    ; (03E7), printf("\n")
          test word ptr ds:data _ 220e.0FFFFh   ; (66E8:3588=4558h)
          jnz loc _ 378
          retn
                                   Called from:  6710:214A, 21A1, 21B3

          sub _ 98  proc near
          push ds
          push si
```

```
        lds si,dword ptr ds:data _ 218e  ; (66E8:3584=6E69h) Load 32 bit ptr
        mov al, [si-1]
        pop si
        pop ds
        retn
        sub _ 98   endp

;xitoa()
        sub _ 99   proc near
        push ds
        lds si,dword ptr ds:data _ 218e  ; (6710:3584=0) Load 32 bit ptr
        lodsb                            ; String [si] to al
        pop ds
        mov ds:data _ 218e,si            ; (6710:3584=0)
        push ax
        push di
        mov di,ds:data _ 228e            ; (6710:35A3=0)
        call sub _ 23                    ; (03F1)
        mov ds:data _ 228e,di            ; (6710:35A3=0)
        pop di
        mov si,ds:data _ 220e            ; (6710:3588=0)
        or si,si                         ; Zero ?
        jz loc _ 379                     ; Jump if zero
        dec si
        mov ds:data _ 220e,si            ; (6710:3588=0)
loc _ 379:
        inc byte ptr ds:data _ 227e      ; (6710:35A2=0)
        pop ax
        retn
        sub _ 99   endp

sub _ 99 _ b:
        inc byte ptr ds:data _ 235e      ; (66E8:35AC=61h)

sub _ 99 _ x:
        inc byte ptr ds:data _ 235e      ; (66E8:35AC=61h)

        sub _ 100   proc near
        inc byte ptr ds:data _ 235e      ; (6710:35AC=0)
        sub _ 101:
        inc byte ptr ds:data _ 235e         ; (6710:35AC=0)

sub _ 101 _ x:
        pop bx
        call sub _ 103                      ; (1ED4)
        call sub _ 22                       ; (03E7)

        sub _ 102:
        push ds
        lds si,dword ptr ds:data _ 218e     ; (6710:3584=0) Load 32 oit ptr
        call sub _ 16                       ; (03BE), save ds,si for print
        pop ds
        call sub _ 18                       ; (03D2), printf("%04X:%04X", ...)
        mov byte ptr ds:data _ 227e,0       ; (6710:35A2=0)
        mov di,37F8h
        mov cx,32h
        mov al,0
        rep stosb                           ; Rep while cx>0 Store al to es: [di]
        mov di,37F8h
        mov cx,23h
        mov al,20h                          ; ' '
        rep stosb                           ; Rep while cx>0 Store al to es: [di]
        mov di,37A2h
        mov ds:data _ 228e,di               ; (6710:35A3=0)
        call sub _ 99                       ; (1E3B), xitoa()
        mov di,ds:data _ 228e               ; (6710:35A3=0)
        mov ah,0
        mov bx,ax                           ;取指令字节
        and al,1
        mov ds:data _ 230e,al               ; (6710:35A7=0)
        mov al,bl
        shl bx,1
        shl bx,1                            ;bx=0..0xff, 指令编码
        add bx,2429h                        ;offset Instr _ Table
        mov dx, [bx]
        mov ds:data _ 229e,dx               ; 指令串地址
        mov ds:data _ 228e,di               ; 指令列表串地址
        mov di,37F8h
        call word ptr [bx+2]                ; 反汇编,取指令串
        sub _ 103:
        mov ah,ds:data _ 227e          ; (6710:35A2=0)
        add ah,ah
        mov al,0Eh
        sub al,ah
        cbw                            ; Convrt byte to word
        xchg ax,cx
        mov di,ds:data _ 228e          ; (6710:35A3=0)
        call sub _ 27                  ; (041A)
        mov si,ds:data _ 229e          ; (6710:35A5=0)
        or si,si                       ; Zero ?
        jz loc _ 381                   ; Jump if zero
loc _ 380:
        lodsb                          ; String [si] to al
        or al,al                       ; Zero ?

sub _ 103 _ x:
        jz loc _ 381                   ; Jump if zero
        stosb                          ; Store al to es: [di]
        jmp short loc _ 380            ; (1EEF)
loc _ 381:
        mov al,9
        stosb                          ; Store al to es: [di]
        mov byte ptr [di] ,0
        mov dx,384Eh                   ; printf("%s %s",...),打印列表和指令串
        call sub _ 20                  ; (03DA)
        retn
        sub _ 100   endp

        sub _ 104   proc near
        call sub _ 99                  ; (1E3B)
        mov ah,al
        and al,7
        mov ds:data _ 231e,al          ; (66E8:35A8=65h)
        shr ah,1
        shr ah,1
        shr ah,1
        mov al,ah
        and al,7
```

```
          mov ds:data _ 232e,al              ; (66E8:35A9=73h)
          shr ah,1
          shr ah,1
          shr ah,1
          mov ds:data _ 233e,ah              ; (66E8:35AA=20h)
          retn
          sub _ 104   endp

sub _ 104 _ x:
          mov bx,2FD6h
          call sub _ 134                     ; (2381)
loc _ 382:
          call sub _ 133                     ; (2350)
          jmp short loc _ 384                ; (1F3B)

sub _ 104 _ y:
          call sub _ 104                     ; (1F04)
          jmp short loc _ 382                ; (1F2C)

          sub _ 105   proc near
          xor al,al
loc _ 383:
          call sub _ 113                     ; (1FBF)
loc _ 384:
          mov al,2Ch                         ; ','
          stosb
          test byte ptr ds:data _ 230e,0FFh    ; (6710:35A7=0)
          jnz loc _ 387

          sub _ 106:
loc _ 385:
          call sub _ 99                      ; (1E3B)
          jmp short loc _ 388                ; (1F72)

sub _ 106 _ x:
          push di
          mov di,359Eh
          call sub _ 107                     ; (1F63)
          pop di
          call sub _ 107                     ; (1F63)
```

```
        mov al,3Ah                    ; ' :'
        stosb                         ; Store al to es: [di]
        mov si,359Eh
        mov cx,4
locloop _ 386:
        lodsb
        stosb
        loop locloop _ 386            ; Loop if cx > 0
        retn

        sub _ 107:
loc _ 387:
        call sub _ 99                 ; (1E3B)
        mov dl,al
        call sub _ 99                 ; (1E3B)
        mov dh,al
        call sub _ 108                ; (1F72)
        mov al,dl
        sub _ 105    endp

        sub _ 108    proc near
loc _ 388:
        mov ah,al
        shr al,1
        shr al,1
        shr al,1
        shr al,1
        call sub _ 109                ; (1F81)
        mov al,ah
            ;           Called from:  6710:1F7C
        sub _ 109:
        and al,0Fh
        add al,90h
        daa
        adc al,40h                    ; '@'
        daa                           ; Decimal adjust
        stosb
        retn
        sub _ 108    endp
```

```
sub_108_b:
        call sub_99                 ; (1E3B)
        cmp al,0Ah
        jne loc_388                 ; Jump if not equal
        retn
sub_108_x:
        mov bx,2FD6h
        call sub_134                ; (2381)
        call sub_133                ; (2350)
        mov al,2Ch                  ; ','
        stosb                       ; Store al to es:[di]
            ;           Called from: 6710:2078
        sub_110  proc near
        call sub_99                 ; (1E3B)
        cbw                         ; Convrt byte to word
        mov dx,ax
        mov ah,al
        mov al,2Bh                  ; '+'
        or ah,ah                    ; Zero ?
        jns loc_389                 ; Jump if not sign
        mov al,2Dh                  ; '—'
        neg ah
loc_389:
        stosb                       ; Store al to es:[di]
        mov al,ah
        jmp short loc_388           ; (1F72)
        sub_110  endp
          ;     Called from:  6710:1ED1, 24AB, 24AF, 24CB, 24CF, 24EB, 24EF
        sub_111  proc near
        call sub_116                ; (2001)
        mov al,2Ch                  ; ','
        stosb                       ; Store al to es:[di]
            ;           Called from: 6710:200E
        sub_112:
        mov al,ds:data_232e         ; (6710:35A9=0)
            ;           Called from: 6710:1F38, 20D1
        sub_113:
loc_390:
        mov si,23F3h
        cmp byte ptr ds:data_230e,1  ; (6710:35A7=0)
        jne loc _ 392                   ; Jump if not equal
             ;           Called from:   6710:20C4
        sub _ 114:
loc _ 391:
        mov si,2403h
loc _ 392:
        cbw                             ; Convrt byte to word
        add si,ax
        add si,ax
        movsw                           ; Mov [si] to es: [di]
        retn
        sub _ 111   endp
sub _ 114 _ x:
        shr al,1
        shr al,1
        shr al,1
             ;           Called from:   6710:1FF7
        sub _ 115   proc near
loc _ 393:
        and al,3
        mov si,2413h
        jmp short loc _ 392             ; (1FCC)
        sub _ 115   endp
sub _ 115 _ x:
        and al,7
        jmp short loc _ 391             ; (1FC9)
sub _ 115 _ x1:
        mov byte ptr ds:data _ 230e,1   ; (66E8:35A7=6Ch)
        call sub _ 116                  ; (2001)
        mov al,2Ch                      ; ','
        stosb                           ; Store al to es: [di]
        mov al,ds:data _ 232e           ; (66E8:35A9=73h)
        jmp short loc _ 393             ; (1FD9)
sub _ 115 _ y:
        call sub _ 104                  ; (1F04)
        call sub _ 115                  ; (1FD9)
        mov byte ptr ds:data _ 230e,1   ; (6710:35A7=0)
        jmp short loc _ 394             ; (2011)
             ;           Called from:   6710:1FB6, 1FE9
        sub _ 116   proc near
```

```
                call sub _ 104                  ; (1F04)
                jmp short loc _ 395             ; (2014)
sub _ 116 _ b:
                mov byte ptr ds:data _ 230e,1   ; (6710:35A7=0)
                  ;     Called from:  6710:1ED1, 24B3, 24B7, 24D3, 24D7, 24F3, 24F7
                sub _ 117:
                call sub _ 104                  ; (1F04)
                call sub _ 112                  ; (1FBC)
loc _ 394:
                mov al,2Ch                      ; ','
                stosb                           ; Store al to es: [di]
loc _ 395:
                cmp byte ptr ds:data _ 233e,3   ; (6710:35AA=0)
                mov al,ds:data _ 231e           ; (6710:35A8=0)
                jz loc _ 390                    ; Jump if zero
                xor bx,bx                       ; Zero register
                mov byte ptr ds:data _ 234e,3   ; (6710:35AB=0)
                mov byte ptr [di] ,5Bh          ; ' ['
                inc di
                cmp al,6
                jne loc _ 396                   ; Jump if not equal
                cmp byte ptr ds:data _ 233e,0   ; (6710:35AA=0)
                je loc _ 406                    ; Jump if equal
loc _ 396:
                mov dl,al
                cmp al,1
                jbe loc _ 407                   ; Jump if below or =
                cmp al,7
                je loc _ 407                    ; Jump if equal
                cmp al,3
                jbe loc _ 397                   ; Jump if below or =
                cmp al,6
                jne loc _ 399                   ; Jump if not equal
loc _ 397:
                mov bx,word ptr bp _ save       ; (6710:3182=0)
                mov byte ptr ds:data _ 234e,2   ; (6710:35AB=0)
                mov ax,5042h
loc _ 398:
                stosw                           ; Store ax to es: [di]
loc _ 399:
        cmp dl,4
        jae loc_400                     ; Jump if above or =
        mov al,2Bh                      ; '+'
        stosb                           ; Store al to es: [di]
loc_400:
        cmp dl,6
        jae loc_402                     ; Jump if above or =
        and dl,1
        jz loc_408                      ; Jump if zero
        add bx,di_save                  ; (6710:3186=0)
        mov ax,4944h
loc_401:
        stosw                           ; Store ax to es: [di]
loc_402:
        mov al,ds:data_233e             ; (6710:35AA=0)
        or al,al                        ; Zero ?
        jz loc_404                      ; Jump if zero
        cmp al,2
        je loc_405                      ; Jump if equal
        call sub_110                    ; (1F9F)
loc_403:
        add bx,dx
loc_404:
        mov al,5Dh                      ; ']'
        stosb                           ; Store al to es: [di]
        mov ds:data_222e,bx             ; (6710:358C=0)
            ;             Called from:  6710:1ED1, 24C7, 24E7
        sub_118:
        retn
loc_405:
   mov al,2Bh                           ; '+'
        stosb                           ; Store al to es: [di]
loc_406:
        call sub_107                    ; (1F63)
        jmp short loc_403               ; (207B)
loc_407:
        mov bx,bx_save                  ; (6710:317A=0)
        mov ax,5842h
        jmp short loc_398               ; (2052)
loc_408:
```

```
                add bx,si _ save                ; (6710:3184=0)
                mov ax,4953h
                jmp short loc _ 401             ; (206C)
                sub _ 116   endp
        sub _ 116 _ x:
                call sub _ 99                   ; (1E3B)
                cbw                             ; Convrt byte to word
                add ax,ds:data _ 218e           ; (6710:3584=0)
                xchg ax,dx
        loc _ 409:
                mov al,dh
                call sub _ 108                  ; (1F72)
                mov al,dl
                jmp loc _ 388                   ; (1F72)
        sub _ 116 _ x2:
                call sub _ 99                   ; (1E3B)
                mov dl,al
                call sub _ 99                   ; (1E3B)
                mov dh,al
                add dx,ds:data _ 218e           ; (6710:3584=0)
                jmp short loc _ 409             ; (20A8)
        sub _ 116 _ y:
                and al,7
                call sub _ 114                  ; (1FC9)
                mov al,2Ch                      ; ','
                stosb                           ; Store al to es:[di]
                xor al,al                       ; Zero register
                jmp loc _ 391                   ; (1FC9)
        sub _ 118 _ x:
                xor al,al                       ; Zero register
                call sub _ 113                  ; (1FBF)
                mov al,2Ch                      ; ','
                stosb                           ; Store al to es:[di]
                                                ;          Called from:   6710:20E3
                sub _ 119   proc near
                mov al,5Bh                      ; '['
                stosb                           ; Store al to es:[di]
                xor bx,bx                       ; Zero register
                mov byte ptr ds:data _ 234e,3   ; (66E8:35AB=63h)
                jmp short loc _ 406             ; (2088)
        sub _ 119   endp
sub _ 119 _ x:
        call sub _ 119                  ; (20D7)
        mov al,2Ch                      ; ','
        stosb                           ; Store al to es:[di]
        xor al,al                       ; Zero register
        jmp loc _ 390                   ; (1FBF)
sub _ 119 _ y:
        mov byte ptr ds:data _ 230e,0   ; (66E8:35A7=6Ch)
        jmp short loc _ 410             ; (20FA)
sub _ 119 _ z:
        mov byte ptr ds:data _ 230e,1   ; (66E8:35A7=6Ch)
loc _ 410:
        and al,7
        jmp loc _ 383                   ; (1F38)
sub _ 119 _ x2:
        mov byte ptr [di] ,33h          ; '3'
        inc di
        retn
sub _ 119 _ x3:
        call sub _ 127                  ; (22B0)
        jz loc _ 411                    ; Jump if zero
        mov si,2D62h
        jmp short loc _ 413             ; (2138)
        nop
sub _ 119 _ x4:
        call sub _ 127                  ; (22B0)
        jz loc _ 411                    ; Jump if zero
        mov si,2D34h
        jmp short loc _ 417             ; (2161)
        nop
loc _ 411:
        mov al,dl
        test al,4
        jz loc _ 412                    ; Jump if zero
        jmp loc _ 448                   ; (22D1)
loc _ 412:
        and al,3
        mov si,2D52h
        mov cl,al
```

```
          call sub _ 125               ; (2284)
          jmp short loc _ 423          ; (219B)
          nop
sub _ 119 _ x5:
          call sub _ 127               ; (22B0)
          jz loc _ 414                 ; Jump if zero
          mov si,2D07h
loc _ 413:
          call sub _ 126               ; (22A3)
          call sub _ 124               ; (223A)
          jmp loc _ 395                ; (2014)
loc _ 414:
          mov al,dl
          test al,4
          jnz loc _ 416                ; Jump if not zero
loc _ 415:
          jmp loc _ 448                ; (22D1)
loc _ 416:
          call sub _ 98                ; (1E2F)
          and al,1Fh
          cmp al,4
          jae loc _ 415                ; Jump if above or =
          mov si,2D21h
          jmp short loc _ 427          ; (21B9)
          nop
sub _ 119 _ x6:
          call sub _ 127               ; (22B0)
          jz loc _ 419                 ; Jump if zero
          mov si,2C56h
loc _ 417:
          call sub _ 126               ; (22A3)
          and al,7
          cmp al,3
          ja loc _ 418                 ; Jump if above
          mov al,dl
          call sub _ 124               ; (223A)
loc _ 418:
          jmp loc _ 395                ; (2014)
loc _ 419:
          mov al,dl
```

```
                test al,4
                jnz loc_426                    ; Jump if not zero
                and al,7
                or al,al                       ; Zero ?
                jnz loc_420                    ; Jump if not zero
                mov ax,444Ch
                stosw                          ; Store ax to es: [di]
                jmp short loc_423              ; (219B)
loc_420:
                cmp al,1
                jne loc_421                    ; Jump if not equal
                mov ax,4358h
                stosw                          ; Store ax to es: [di]
                mov al,48h                     ; 'H'
                jmp short loc_422              ; (219A)
loc_421:
                cmp al,3
                jne loc_424                    ; Jump if not equal
                mov ax,5453h
                stosw                          ; Store ax to es: [di]
                mov al,50h                     ; 'P'
loc_422:
                stosb                          ; Store al to es: [di]
loc_423:
                mov al,9
                stosb                          ; Store al to es: [di]
                jmp short loc_432              ; (221B)
                nop
loc_424:
                call sub_98                    ; (1E2F)
                cmp al,0D0h
                je loc_425                     ; Jump if equal
                jmp loc_448                    ; (22D1)
loc_425:
                mov ax,4F4Eh
                stosw                          ; Store ax to es: [di]
                mov al,50h                     ; 'P'
                stosb                          ; Store al to es: [di]
                retn
loc_126:
```

```
            call sub_98                     ; (1E2F)
            mov si,2C78h
loc_427:
            and al,1Fh
            mov cl,al
            jmp loc_441                     ; (2284)
sub_119_y2:
            call sub_128                    ; (22BE)
            call sub_123                    ; (2235)
            mov al,dl
            cmp byte ptr ds:data_233e,3     ; (66E8:35AA=20h)
            je loc_428                      ; Jump if equal
            call sub_122                    ; (222B)
            mov al,9
            stosb                           ; Store al to es:[di]
            mov al,dl
            call sub_124                    ; (223A)
            jmp loc_395                     ; (2014)
loc_428:
            test al,20h                     ; ' '
            jz loc_429                      ; Jump if zero
            test al,4
            jz loc_429                      ; Jump if zero
            xor al,1
            mov dl,al
loc_429:
            call sub_122                    ; (222B)
            mov al,dl
            test al,10h
            jz loc_430                      ; Jump if zero
            mov al,50h                      ; 'P'
            stosb                           ; Store al to es:[di]
loc_430:
            mov al,9
            stosb                           ; Store al to es:[di]
            mov al,dl
            and al,7
            cmp al,2
            je loc_432                      ; Jump if equal
            cmp al,3
```

```
        je loc _ 432                ; Jump if equal
        mov al,dl
        test al,20h                 ; ' '
        jz loc _ 431                ; Jump if zero
        call sub _ 121              ; (221B)
        mov al,2Ch                  ; ' ,'
        stosb                       ; Store al to es: [di]
          ;          Called from:   6710:2215, 221B
        sub _ 120   proc near
        mov ax,5453h
        stosw                       ; Store ax to es: [di]
        retn
        sub _ 120   endp
loc _ 431:
        call sub _ 120              ; (2210)
        mov al,2Ch                  ; ' ,'
        stosb                       ; Store al to es: [di]
          ;          Called from:   6710:220A
        sub _ 121   proc near
loc _ 432:
        call sub _ 120              ; (2210)
        mov al,28h                  ; ' ('
        stosb                       ; Store al to es: [di]
        mov al,ds:data _ 231e       ; (66E8:35A8=65h)
        add al,30h                  ; ' 0'
        stosb                       ; Store al to es: [di]
        mov al,29h                  ; ' )'
        stosb                       ; Store al to es: [di]
        retn
        sub _ 121   endp
          ;          Called from:   6710:21CF, 21E9
        sub _ 122   proc near
        mov si,2BE9h
        mov cl,al
        and cl,7
        jmp short loc _ 441         ; (2284)

                                    ;          Called from:   6710:21C3
        sub _ 123:
        mov si,2C0Ch
        jmp short loc _ 433         ; (223D)
```

```
                                        Called from:  6710:213B, 216C, 21D7
          sub_124:
          mov si,2C16h
loc_433:
          cmp byte ptr ds:data_233e,3   ; (66E8:35AA=20h)
          jne loc_436                   ; Jump if not equal
          and al,38h                    ; '8'
          cmp al,10h
          je loc_434                    ; Jump if equal
          mov al,dl
          cmp al,33h                    ; '3'
          jne loc_435                   ; Jump if not equal
          cmp byte ptr ds:data_231e,1   ; (66E8:35A8=65h)
          je loc_435                    ; Jump if equal
loc_434:
          jmp short loc_447             ; (22D0)
          nop
loc_435:
          xor cl,cl                     ; Zero register
          jmp short loc_441             ; (2284)
loc_436:
          cmp al,3Dh                    ; '='
          je loc_437                    ; Jump if equal
          cmp al,3Fh                    ; '?'
          jne loc_438                   ; Jump if not equal
loc_437:
          mov cl,2
          jmp short loc_441             ; (2284)
loc_438:
          cmp al,1Dh
          je loc_439                    ; Jump if equal
          cmp al,3Ch                    ; '<'
          je loc_439                    ; Jump if equal
          cmp al,3Eh                    ; '>'
          je loc_439                    ; Jump if equal
          cmp al,1Fh
          jne loc_440                   ; Jump if not equal
loc_439:
          mov cl,5
          jmp short loc_441             ; (2284)
```

```
loc _ 440:
        mov cl,4
        shr al,cl                     ; Shift w/zeros fill
        mov cl,al
                                      ;          Called from:  6710:212A, 22A7, 2365
        sub _ 125:
loc _ 441:
        push ax
        inc cl
loc _ 442:
        dec cl
        jz loc _ 444
loc _ 443:
        lodsb
        cmp al,24h                    ; '$'
        je loc _ 442
        jmp short loc _ 443           ; (228B)
loc _ 444:
        lodsb
        cmp al,24h                    ; '$'
        je loc _ 446
        cmp al,40h                    ; '@'
        jne loc _ 445
        pop ax
        jmp short loc _ 447           ; (22D0)
loc _ 445:
        stosb
        jmp short loc _ 444           ; (2292)
loc _ 446:
        pop ax
        retn

        sub _ 126:
        and al,7
        mov cl,al
        call sub _ 125                ; (2284)
        mov al,9
        stosb
        mov al,dl
        retn
        sub _ 127:
        call sub _ 128                ; (22BE)
        mov al,46h                    ; 'F'
        stosb
        cmp byte ptr ds:data _ 233e,3 ; (66E8:35AA=20h)
        mov al,dl
        retn

        sub _ 128:
        and al,7
        mov dl,al
        call sub _ 104                ; (1F04)
        shl dl,1                      ; Shift w/zeros fill
        shl dl,1
        shl dl,1
        or al,dl
        mov dl,al
        retn
loc _ 447:
        pop di
loc _ 448:
        mov word ptr ds:data _ 229e,289Dh    ; (66E8:35A5=6966h)
        mov al,dl
        mov di,37F8h
        jmp short loc _ 449           ; (22E1)
        call sub _ 128                ; (22BE)
loc _ 449:
        call sub _ 108                ; (1F72)
        cmp byte ptr ds:data _ 233e,3 ; (66E8:35AA=20h)
        je loc _ 450
        mov byte ptr ds:data _ 230e,1 ; (66E8:35A7=6Ch)
        jmp loc _ 394                 ; (2011)
loc _ 450:
        mov al,2Ch                    ; ','
        stosb
        mov al,ds:data _ 231e         ; (66E8:35A8=65h)
        and al,7
        jmp loc _ 390                 ; (1FBF)
        sub _ 122   endp
```

```
sub_128_x0:
        call sub_130                    ; (2340)
        jmp short loc_451               ; (2306)
sub_128_x:
        call sub_129                    ; (233B)
loc_451:
        mov al,2Ch                      ; ','
        stosb
        jmp short loc_456               ; (2345)
        nop

sub_128_y0:
        call sub_130                    ; (2340)
        jmp short loc_452               ; (2314)

sub_128_y:
        call sub_129                    ; (233B)
loc_452:
        mov al,2Ch                      ; ','
        stosb
        jmp loc_385                     ; (1F45)
        stosw
        retn

sub_128_y1:
        mov bx,4C41h                    ; 'AL'
        jmp short loc_453               ; (2324)
sub_128_y2:
        mov bx,5841h                    ; 'AX'
loc_453:
        call sub_131                    ; (2345)
loc_454:
        mov al,2Ch                      ; ','
        stosb
        mov ax,bx
        stosw
        retn

sub_128_y2x:
        mov bx,4C41h                ; 'AL'
        jmp short loc_455           ; (2336)
sub_128_y3:
        mov bx,5841h                ; 'AX'
loc_455:
        call sub_106                ; (1F45)
        jmp short loc_454           ; (2327)

        sub_129   proc neai
        mov ax,4C41h                ; 'AL'
        jmp short loc_457           ; (2348)

        sub_130:
        mov ax,5841h                ; 'AX'
        jmp short loc_457           ; (2348)

        sub_131:
loc_456:
        mov ax,5844h                ; 'DX'
loc_457:
        stosw
        retn
        sub_129   endp

        sub_132   proc near
        mov bx,2FC6h
        call sub_134                ; (2381)

        sub_133:
loc_458:
        cmp byte ptr ds:data_233e,3 ; (66E8:35AA=20h)
        je loc_460
        mov si,2C16h
        mov cl,3
        test byte ptr ds:data_230e,0FFh    ; (66E8:35A7=6Ch)
        jnz loc_459
        inc cl
loc_459:
        call sub_125                ; (2284)
loc_460:
        jmp loc _ 395                   ; (2014)
        sub _ 132   endp

sub _ 132 _ b:
        call sub _ 132                  ; (234A)
        mov al,2Ch                      ; ' ,'
        stosb
        mov word ptr [di] ,4C43h        ; 'CL'
        add di,2
        retn

sub _ 132 _ x:
        call sub _ 132                  ; (234A)
        mov ax,312Ch
        stosw
        retn

        sub _ 134   proc near
        call sub _ 104                  ; (1F04)
        mov dl,al
        cbw
        shl ax,1                        ; Shift w/zeros fill
        add bx,ax
        mov ax, [bx]
        mov ds:data _ 229e,ax           ; (66E8:35A5=6966h)
        mov al,dl
        retn
        sub _ 134   endp

sub _ 134 _ x:
        mov bx,2FE6h
        call sub _ 134                  ; (2381)
        or al,al
        jz loc _ 461
        jmp short loc _ 458             ; (2350)
loc _ 461:
        jmp loc _ 382                   ; (1F2C)

sub _ 134 _ x2:
        mov bx,2FF6h
```

```
          call sub_134               ; (2381)
          cmp al,2
          jb loc_458
          cmp al,6
          jae loc_462
          test al,1
          jz loc_462
          mov ax,4146h               ; 'AF'
          stosw
          mov ax,2052h
          stosw
loc_462:
          jmp loc_395                ; (2014)

          sub_135   proc near
          retn
          sub_135   endp

          sub_136   proc near
          retn
          sub_136   endp

loc_463:
          mov dx,32B5h               ; 'Disk'
          or al,al
          jnz loc_464
          mov dx,32BAh               ; 'Write Protect'
loc_464:
          push cs
          pop ds
          push cs
          pop es
          mov ds:data_275e,dx        ; (6710:3937=0)
          add drive_char,41h         ; (6710:32C8=41h) 'A'
          mov si,32CAh               ;'reading'
          cmp RWfunc_no,40h          ; (6710:3196=3Fh) '@'
          jne loc_465
          mov si,32D2h               ;'writing'
loc_465:
          mov ds:data_276e,si        ; (6710:3939=0)
          mov dx,3935h
          mov ah,0Dh
          int 21h                        ; DOS Services  ah=function 0Dh
                                         ;   flush disk buffers to disk
          jmp loc_47                     ; (032C)
reg_h_str         db 'ALCLDLBLAHCHDHBH'
reg_x_str         db 'AXCXDXBX'
reg_p_str         db 'SPBPSIDI'
reg_s_str         db 'ESCSSSDS',0,0

db 'BYWODWQWTB',0,0

dw offset add_i, offset sub_111          ;0
dw offset add_i, offset sub_111
dw offset add_i, offset sub_117
dw offset add_i, offset sub_117
dw offset add_i, offset sub_105
dw offset add_i, offset sub_105

dw offset push_i, offset sub_114_x
dw offset pop_i, offset sub_114_x

dw offset or_i, offset sub_111
dw offset or_i, offset sub_111
dw offset or_i, offset sub_117           ;10
dw offset or_i, offset sub_117
dw offset or_i, offset sub_105
dw offset or_i, offset sub_105

dw offset push_i, offset sub_114_x
dw offset pop_i, offset sub_114_X

dw offset adc_i, offset sub_111
dw offset adc_i, offset sub_111
dw offset adc_i, offset sub_117
dw offset adc_i, offset sub_117
dw offset adc_i, offset sub_105          ;20
dw offset adc_i, offset sub_105

dw offset push_i, offset sub_114_x
dw offset pop_i, offset sub_114_x

dw offset sbb_i, offset sub_111
dw offset sbb_i, offset sub_111
dw offset sbb_i, offset sub_117
dw offset sbb_i, offset sub_117
dw offset sbb_i, offset sub_105
dw offset sbb_i, offset sub_105

dw 2B54h, offset sub_114_x          ;30
dw 2B4Ah, offset sub_114_x

dw offset and_i, offset sub_111
dw offset and_i, offset sub_111
dw offset and_i, offset sub_117
dw offset and_i, offset sub_117
dw offset and_i, offset sub_105
dw offset and_i, offset sub_105

dw 2BD5h, offset sub_101
dw 288Dh, offset sub_118

dw offset sub_i, offset sub_111     ;40
dw offset sub_i, offset sub_111
dw offset sub_i, offset sub_117
dw offset sub_i, offset sub_117
dw offset sub_i, offset sub_105
dw offset sub_i, offset sub_105

dw 2BD9h, offset sub_100
dw 2891h, offset sub_118

dw offset xor_i, offset sub_111
dw offset xor_i, offset sub_111
dw offset xor_i, offset sub_117     ;50
dw offset xor_i, offset sub_117
dw offset xor_i, offset sub_105
dw offset xor_i, offset sub_105

dw 2BDDh, offset sub_99_x
dw 2850h, offset sub_118

dw offset cmp_i, offset sub_111
dw offset cmp_i, offset sub_111
dw offset cmp_i, offset sub_117
dw offset cmp_i, offset sub_117
dw offset cmp_i, offset sub_105        ;60
dw offset cmp_i, offset sub_105

dw 2BE1h, offset sub_99_b
dw 285Ch, offset sub_118

dw offset inc_i, offset sub_115_x
dw offset inc_i, offset sub_115_x
dw offset inc_i, offset sub_115_x
dw offset inc_i, offset sub_115_x
dw offset inc_i, offset sub_115_x
dw offset inc_i, offset sub_115_x
dw offset inc_i, offset sub_115_x        ;70
dw offset inc_i, offset sub_115_x

dw offset dec_i, offset sub_115_x
dw offset dec_i, offset sub_115_x
dw offset dec_i, offset sub_115_x
dw offset dec_i, offset sub_115_x
dw offset dec_i, offset sub_115_x
dw offset dec_i, offset sub_115_x
dw offset dec_i, offset sub_115_x
dw offset dec_i, offset sub_115_x

dw offset push_i, offset sub_115_x        ;80
dw offset push_i, offset sub_115_x
dw offset push_i, offset sub_115_x
dw offset push_i, offset sub_115_x
dw offset push_i, offset sub_115_x
dw offset push_i, offset sub_115_x
dw offset push_i, offset sub_115_x
dw offset push_i, offset sub_115_x

dw offset pop_i, offset sub_115_x
dw offset pop_i, offset sub_115_x
dw offset pop_i, offset sub_115_x          90
dw offset pop_i, offset sub_115_x
dw offset pop_i, offset sub_115_x
dw offset pop_i, offset sub_115_x
dw offset pop_i, offset sub_115_x
dw offset pop_i, offset sub_115_x

dw offset db_i, offset sub_108
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108   ;100
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108   ;110
dw offset db_i, offset sub_108

dw 2AD7h, offset sub_116_x
dw 2AD3h, offset sub_116_x
dw 2A7Eh, offset sub_116_x
dw 2A76h, offset sub_116_x
dw 2A91h, offset sub_116_x
dw 2ABBh, offset sub_116_x
dw 2A7Ah, offset sub_116_x
dw 2A6Eh, offset sub_116_x
dw 2ADAh, offset sub_116_x        ;120
dw 2ACFh, offset sub_116_x
dw 2AC3h, offset sub_116_x
dw 2AC7h, offset sub_116_x
dw 2AABh, offset sub_116_x
dw 2A97h, offset sub_116_x
dw 2AA7h, offset sub_116_x
dw 2A9Bh, offset sub_116_x

dw 0000h, offset sub_104_x
dw 0000h, offset sub_104_x
dw 0000h, offset sub_104_x          ;130

dw 0000h, offset sub_108_x

dw 2BC1h, offset sub_117
dw 2BC1h, offset sub_117
dw 2BCBh, offset sub_117
dw 2BCBh, offset sub_117

dw offset mov_i, offset sub_111
dw offset mov_i, offset sub_111
dw offset mov_i, offset sub_117
dw offset mov_i, offset sub_117
dw offset mov_i, offset sub_115_x1          ;140

dw 2AE9h, offset sub_116_b
dw offset mov_i, offset sub_115_y
dw offset pop_i, offset sub_116
dw 2B39h, offset sub_118

dw offset xchg_i, offset sub_116_y
dw offset xchg_i, offset sub_116_y
dw offset xchg_i, offset sub_116_y
dw offset xchg_i, offset sub_116_y
dw offset xchg_i, offset sub_116_y
dw offset xchg_i, offset sub_116_y          ;150
dw offset xchg_i, offset sub_116_y

dw 2865h, offset sub_118
dw 2889h, offset sub_118

dw 2860h, offset sub_106_x

dw 2BC6h, offset sub_118
dw 2B4Eh, offset sub_118
dw 2B45h, offset sub_118
dw 2B8Ch, offset sub _ 118
dw 2AE0h, offset sub _ 118

dw offset mov _ i, offset sub _ 118 _ x          ;160
dw offset mov _ i, offset sub _ 118 _ x
dw offset mov _ i, offset sub _ 119 _ x
dw offset mov _ i, offset sub _ 119 _ x

dw 2B21h, offset sub _ 118
dw 2B27h, offset sub _ 118
dw 2879h, offset sub _ 118
dw 287Fh, offset sub _ 118

dw 2BC1h, offset sub _ 105
dw 2BC1h, offset sub _ 105

dw 2BB5h, offset sub _ 118          ;170
dw 2BBBh, offset sub _ 118
dw 2AF6h, offset sub _ 118
dw 2AFCh, offset sub _ 118
dw 2B95h, offset sub _ 118
dw 2B9Bh, offset sub _ 118

dw offset mov _ i, offset sub _ 119 _ y
dw offset mov _ i, offset sub _ 119 _ y
dw offset mov _ i, offset sub _ 119 _ y
dw offset mov _ i, offset sub _ 119 _ y
dw offset mov _ i, offset sub _ 119 _ y          ;180
dw offset mov _ i, offset sub _ 119 _ y
dw offset mov _ i, offset sub _ 119 _ y
dw offset mov _ i, offset sub _ 119 _ y
dw offset mov _ i, offset sub _ 119 _ z
dw offset mov _ i, offset sub _ 119 _ z
dw offset mov _ i, offset sub _ 119 _ z
dw offset mov _ i, offset sub _ 119 _ z
dw offset mov _ i, offset sub _ 119 _ z
dw offset mov _ i, offset sub _ 119 _ z
dw offset mov _ i, offset sub _ 119 _ z          ;190
dw offset mov _ i, offset sub _ 119 _ z
dw offset db_i, offset sub_108
dw offset db_i, offset sub_108

dw 2B80h, offset sub_107
dw 2B80h, offset sub_118
dw 2AEDh, offset sub_116_b
dw 2AE5h, offset sub_116_b
dw offset mov_i, offset sub_104_y
dw offset mov_i, offset sub_104_y
dw offset db_i, offset sub_108          ;200
dw offset db_i, offset sub_108
dw 2B7Bh, offset sub_107
dw 2B7Bh, offset sub_118
dw 2A59h, offset sub_119_x2
dw 2A59h, offset sub_106
dw 2A54h, offset sub_118
dw 2A60h, offset sub_118
dw 0000h, offset sub_132_x
dw 0000h, offset sub_132_x
dw 0000h, offset sub_132_b          ;210
dw 0000h, offset sub_132_b
dw 2858h, offset sub_108_b
dw 2854h, offset sub_108_b
dw offset db_i, offset sub_108
dw 2BD0h, offset sub_118
dw 0000h, offset sub_119_y2
dw 0000h, offset sub_119_x6
dw 0000h, offset sub_119_y2
dw 0000h, offset sub_119_x5
dw 0000h, offset sub_119_y2         ;220
dw 0000h, offset sub_119_x4
dw 0000h, offset sub_119_y2
dw 0000h, offset sub_119_x3
dw 2B02h, offset sub_116_x
dw 2B09h, offset sub_116_x
dw 2B1Ch, offset sub_116_x
dw 2A71h, offset sub_116_x
dw 2A5Dh, offset sub_128_y
dw 2A5Dh, offset sub_128_y0
dw 2B41h, offset sub_128_y2x        ;230
dw 2B41h, offset sub_128_y3
dw 2860h, offset sub_116_x2
dw 2AB7h, offset sub_116_x2
dw 2AB7h, offset sub_106_x
dw 2AB7h, offset sub_116_x
dw 2A5Dh, offset sub_128_x
dw 2A5Dh, offset sub_128_x0
dw 2B41h, offset sub_128_y1
dw 2B41h, offset sub_128_y2
dw 2AF1h, offset sub_101_x          ;240
dw offset db_i, offset sub_108
dw 2B66h, offset sub_101_x
dw 2B61h, offset sub_101_x
dw 2A42h, offset sub_118
dw 2875h, offset sub_118
dw 0000h, offset sub_134_x
dw 0000h, offset sub_134_x
dw 2869h, offset sub_118
dw 2BA9h, offset sub_118
dw 2871h, offset sub_118            ;250
dw 2BB1h, offset sub_118
dw 286Dh, offset sub_118
dw 2BADh, offset sub_118
dw 0000h, offset sub_134_x2
dw 0000h, offset sub_134_x2
db_i            db 'DB',0
dw_i            db 'DW',0
note_i          db ';',0
org_i           db 'ORG',0
add_i           db 'ADD',0
adc_i           db 'ADC',0
sub_i           db 'SUB',0
sbb_i           db 'SBB',0
xor_i           db 'XOR',0
or_i            db 'OR',0
and_i           db 'AND',0
aaa_i           db 'AAA',0
aad_i           db 'AAD',0
aam_i           db 'AAM',0
aas_i           db 'AAS',0
```

```
call_i          db 'CALL' ,0
cbw_i           db 'CBW' ,0
clc_i           db 'CLC' ,0
cld_i           db 'CLD' ,0
cli_i           db 'CLI' ,0
cmc_i           db 'CMC' ,0
cmpsb_i         db 'CMPSB' ,0
cmpsw_i         db 'CMPSW' ,0
cmp_i           db 'CMP' ,0
cwd_i           db 'CWD' ,0
daa_i           db 'DAA' ,0
das_i           db 'DAS' ,0
dec_i           db 'DEC' ,0
div_i           db 'DIV' ,0
esc_i           db 'ESC' ,0
fxch_i          db 'FXCH' ,0
ffree_i         db 'FFREE' ,0
fcompp_i        db 'FCOMPP' ,0
fcomp_i         db 'FCOMP' ,0
fcom_i          db 'FCOM' ,0
ficomp_i        db 'FICOMP' ,0
ficom_i         db 'FICOM' ,0
fnop_i          db 'FNOP' ,0
fchs_i          db 'FCHS' ,0
fabs_i          db 'FABS' ,0
ftst_i          db 'FTST' ,0
fxam_i          db 'FXAM' ,0
fldl2t_i        db 'FLDL2T' ,0
fldl2e_i        db 'FLDL2E' ,0
fldlg2_i        db 'FLDLG2' ,0
fldln2_i        db 'FLDLN2' ,0
fldpi_i         db 'FLDPI' ,0
fld1_i          db 'FLD1' ,0
fldz_i          db 'FLDZ' ,0
f2xm1_i         db 'F2XM1' ,0
fyl2xp1_i       db 'FYL2XP1' ,0
fyl2x_i         db 'FYL2X' ,0
fptan_i         db 'FPTAN' ,0
fpatan_i        db 'FPATAN' ,0
fxtract_i       db 'FXTRACT' ,0
```

```
fdecstp_i        db 'FDECSTP' ,0
fincstp_i        db 'FINCSTP' ,0
fprem_i          db 'FPREM' ,0
fsqrt_i          db 'FSQRT' ,0
frndint_i        db 'FRNDINT' ,0
fscale_i         db 'FSCALE' ,0
finit_i          db 'FINIT' ,0
fdisi_i          db 'FDISI' ,0
feni_i           db 'FENI' ,0
fclex_i          db 'FCLEX' ,0
fbld_i           db 'FBLD' ,0
fbstp_i          db 'FBSTP' ,0
fldcw_i          db 'FLDCW' ,0
fstcw_i          db 'FSTCW' ,0
fstsw_i          db 'FSTSW' ,0
fstenv_i         db 'FSTENV' ,0
fldenv_i         db 'FLDENV' ,0
fsave_i          db 'FSAVE' ,0
frstor_i         db 'FRSTOR' ,0
faddp_i          db 'FADDP' ,0
fadd_i           db 'FADD' ,0
fiadd_i          db 'FIADD' ,0
fsubrp_i         db 'FSUBRP' ,0
fsubr_i          db 'FSUBR' ,0
fsubp_i          db 'FSUBP' ,0
fsub_i           db 'FSUB' ,0
fisubr_i         db 'FISUBR' ,0
fisub_i          db 'FISUB' ,0
fmulp_i          db 'FMULP' ,0
fmul_i           db 'FMUL' ,0
fimul_i          db 'FIMUL' ,0
fdivrp_i         db 'FDIVRP' ,0
fdivr_i          db 'FDIVR' ,0
fdivp_i          db 'FDIVP' ,0
fdiv_i           db 'FDIV' ,0
fidivr_i         db 'FIDIVR' ,0
fidiv_i          db 'FIDIV' ,0
fwait_i          db 'FWAIT' ,0
fild_i           db 'FILD' ,0
fld_i            db 'FLD' ,0
```

```
fstp_i          db 'FSTP' ,0
fst_i           db 'FST' ,0
fistp_i         db 'FISTP' ,0
fist_i          db 'FIST' ,0
hlt_i           db 'HLT' ,0
idiv_i          db 'IDIV' ,0
imul_i          db 'IMUL' ,0
inc_i           db 'INC' ,0
into_i          db 'INTO' ,0
int_i           db 'INT' ,0
in_i            db 'IN' ,0
iret_i          db 'IRET' ,0
jnbe_i          db 'JNBE' ,0
jae_i           db 'JAE' ,0
ja_i            db 'JA' ,0
jcxz_I          db 'JCXZ' ,0
jnb_i           db 'JNB' ,0
jbe_i           db 'JBE' ,0
jb_i            db 'JB' ,0
jnc_i           db 'JNC' ,0
jc_i            db 'JC' ,0
jnae_i          db 'JNAE' ,0
jna_i           db 'JNA' ,0
jz_i            db 'JZ' ,0
je_i            db 'JE' ,0
jge_i           db 'JGE' ,0
jg_i            db 'JG' ,0
jnle_i          db 'JNLE' ,0
jnl_i           db 'JNL' ,0
jle_i           db 'JLE' ,0
jl_i            db 'JL' ,0
jnge_i          db 'JNGE' ,0
jng_i           db 'JNG' ,0
jmp_i           db 'JMP' ,0
jnz_i           db 'JNZ' ,0
jne_i           db 'JNE' ,0
jpe_i           db 'JPE' ,0
jpo_i           db 'JPO' ,0
jnp_i           db 'JNP' ,0
jns_i           db 'JNS' ,0
```

```
jno_i           db 'JNO',0
jo_i            db 'JO',0
js_i            db 'JS',0
jp_i            db 'JP',0
lahf_i          db 'LAHF',0
lds_i           db 'LDS',0
lea_i           db 'LEA',0
les_i           db 'LES',0
lock_i          db 'LOCK',0
lodsb_i         db 'LODSB',0
lodsw_i         db 'LODSW',0
loopnz_i        db 'LOOPNZ',0
loopz_i         db 'LOOPZ',0
loopne_i        db 'LOOPNE',0
loope_i         db 'LOOPE',0
loop_i          db 'LOOP',0
movsb_i         db 'MOVSB',0
movsw_i         db 'MOVSW',0
mov_i           db 'MOV',0
mul_i           db 'MUL',0
neg_i           db 'NEG',0
nop_i           db 'NOP',0
not_i           db 'NOT',0
out_i           db 'OUT',0
popf_i          db 'POPF',0
pop_i           db 'POP',0
pushf_i         db 'PUSHF',0
push_i          db 'PUSH',0
rcl_i           db 'RCL',0
rcr_i           db 'RCR',0
repz_i          db 'REPZ',0
repnz_i         db 'REPNZ',0
repe_i          db 'REPE',0
repne_i         db 'REPNE',0
rep_i           db 'REP',0
retf_i          db 'RETF',0
ret_i           db 'RET',0
rol_i           db 'ROL',0
ror_i           db 'ROR',0
sahf_i          db 'SAHF',0
```

```
sar_i           db 'SAR',0
scasb_i         db 'SCASB',0
scasw_i         db 'SCASW',0
shl_i           db 'SHL',0
shr_i           db 'SHR',0
stc_i           db 'STC',0
std_i           db 'STD',0
sti_i           db 'STI',0
stosb_i         db 'STOSB',0
stosw_i         db 'STOSW',0
test_i          db 'TEST',0
wait_i          db 'WAIT',0
xchg_i          db 'XCHG',0
xlat_i          db 'XLAT',0
es_i            db 'ES:',0
cs_i            db 'CS:',0
ss_i            db 'SS:',0
ds_i            db 'DS:',0
unknow_i        db '???',0
db 'ADD$MUL$COM$COMP$SUB$SUBR$DIV$DIVR$F$FI$F$FI$DWORD PTR
    $DWORD PTR $'
db 'QWORD PTR $WORD PTR $BYTE PTR $TBYTE PTR $LD$@$ST$STP$LDENV
    $LDCW$'
db 'STENV$STCW$CHS$ABS$@$@$TST$XAM$@$@$LD1 $LDL2T $LDL2E
    $LDPI$LDLG2$LDLN2'
db ' $LDZ $@$2XM1 $YL2X $PTAN $PATAN $XTRACT $@$DECSTP $INCSTP
    $PREM$YL2XP1$'
db 'SQRT $@$RNDINT $SCALE $@$@$ILD $@$IST $ISTP $@$LD $@$STP
    $ENI$DISI$CLEX$'
db 'INIT $LD $@$ST $STP $RSTOR $@$SAVE $STSW $FREE $XCH $ST $STP
    $ILD$@ $IST$'
db 'ISTP$BLD$ILD$BSTP$ISTP$'

;指令子程序表
db 0FFh,94h,1Dh         ;1
db 0FFh,8Fh,1Dh
db 0FFh,0E2h,13h
db 0FFh,0F2h,1Dh
db 0,7Ch,17h
db 10h,7Ch,17h
db 28h, 7Ch, 17h
[illegible] 18h, 7Ch, 17h
 30h, 7Ch, 17h
db 8, 7Ch, 17h            ;10
db 20h, 7Ch, 17h
db 37h, 68h, 14h
db 0D5h, 64h, 14h
db 0D4h, 64h, 14h
db 3Fh, 68h, 14h
db 10h, 62h, 15h
db 98h, 68h, 14h
db 0F8h, 68h, 14h
db 0FCh, 68h, 14h
db 0FAh, 68h, 14h         ;20
db 0F5h, 68h, 14h
db 0A6h, 68h, 14h
db 0A7h, 68h, 14h
db 38h, 7Ch, 17h
db 99h, 68h, 14h
db 27h, 68h, 14h
db 2Fh, 68h, 14h
db 8, 4Ch, 16h
db 30h, 33h, 17h
db 0D8h, 76h, 16h         ;30
db 9, 4, 17h
db 28h, 4, 17h
db 0D9h, 55h, 14h
db 3, 0BCh, 16h
db 2, 0BCh, 16h
db 13h, 19h, 17h
db 12h, 19h, 17h
db 0D0h, 5Dh, 14h
db 0E0h, 5Dh, 14h
db 0E1h, 5Dh, 14h         ;40
db 0E4h, 5Dh, 14h
db 0E5h, 5Dh, 14h
db 0E9h, 5Dh, 14h
db 0EAh, 5Dh, 14h
db 0ECh, 5Dh, 14h
db 0EDh, 5Dh, 14h
db 0EBh, 5Dh, 14h
db 0E8h, 5Dh, 14h
db 0EEh, 5Dh, 14h
db 0F0h, 5Dh, 14h          ;50
db 0F9h, 5Dh, 14h
db 0F1h, 5Dh, 14h
db 0F2h, 5Dh, 14h
db 0F3h, 5Dh, 14h
db 0F4h, 5Dh, 14h
db 0F6h, 5Dh, 14h
db 0F7h, 5Dh, 14h
db 0F8h, 5Dh, 14h
db 0FAh, 5Dh, 14h
db 0FCh, 5Dh, 14h          ;60
db 0FDh, 5Dh, 14h
db 0E3h, 59h, 14h
db 0E1h, 59h, 14h
db 0E0h, 59h, 14h
db 0E2h, 59h, 14h
db 3Ch, 0E4h, 16h
db 3Eh, 0E4h, 16h
db 0Dh, 0E8h, 16h
db 0Fh, 0E8h, 16h
db 2Fh, 0E8h, 16h          ;70
db 0Eh,0ECh, 16h
db 0Ch,0ECh, 16h
db 2Eh,0ECh, 16h
db 2Ch,0ECh, 16h
db 30h,04h, 17h
db 00h,0C6h, 16h
db 10h,19h,  17h
db 34h,04h, 17h
db 05h,9Ch, 16h
db 35h,04h, 17h            ;80
db 04h,9Ch, 16h
db 15h,19h, 17h
db 14h,19h, 17h
db 31h,04h, 17h
db 01h,0C6h,16h
db 11h,19h, 17h
db 36h,04h, 17h
db 07h,9Ch, 16h
db 37h,04h, 17h
db 06h,9Ch, 16h              ;90
db 17h,19h, 17h
db 16h,19h, 17h
db 9Bh,68h, 14h
db 18h,19h, 17h
db 08h,0BCh, 16h
db 0Bh,0BCh, 16h
db 2Ah,0BCh, 16h
db 1Bh,19h, 17h
db 1Ah,19h, 17h
db 0F4h,68h, 14h             ;100
db 38h,33h, 17h
db 28h,33h, 17h
db 00h,4Ch, 16h
db 0CEh,68h, 14h
db 0CCh,0DFh, 14h
db 0ECh,0F7h, 14h
db 0CFh, 68h,  14h
db 77h, 03h, 16h
db 73h, 03h,  16h
db 77h, 03h,  16h            ;110
db 0E3h, 03h, 16h
db 73h, 03h,  16h
db 76h, 03h,  16h
db 72h, 03h, 16h
db 73h, 03h,  16h
db 72h, 03h,  16h
db 72h, 03h, 16h
db 76h, 03h,  16h
db 74h, 03h,  16h
db 74h, 03h, 16h             ;120
db 7Dh, 03h,  16h
db 7Fh, 03h,  16h
db 7Fh, 03h, 16h
db 7Dh, 03h,  16h
db 7Eh, 03h,  16h
db 7Ch, 03h, 16h
db 7Ch, 03h,  16h
db 7Eh, 03h,  16h
db 20h, 5Eh, 15h
db 75h, 03h,  16h        ;130
db 75h, 03h,  16h
db 7Ah, 03h, 16h
db 7Bh, 03h,  16h
db 7Bh, 03h,  16h
db 79h, 03h, 16h
db 71h, 03h,  16h
db 70h, 03h,  16h
db 78h, 03h, 16h
db 7Ah, 03h,  16h
db 9Fh, 68h,  14h        ;140
db 0C5h, 22h, 16h
db 8Dh, 22h,  16h
db 0C4h, 22h, 16h
db 0F0h, 68h, 14h
db 0ACh, 68h,  14h
db 0ADh, 68h,  14h
db 0E0h, 03h, 16h
db 0E1h, 03h,  16h
db 0E0h, 03h,  16h
db 0E1h, 03h, 16h        ;150
db 0E2h, 03h,  16h
db 0A4h, 68h,  14h
db 0A5h, 68h, 14h
db 0C6h, 74h,  17h
db 20h, 33h,  17h
db 18h, 33h, 17h
db 90h, 68h,  14h
db 10h, 33h,  17h
db 0EEh, 27h,  15h
db 9Dh, 68h, 14h         ;160
db 00h, 79h,  14h
db 9Ch, 68h,  14h
db 30h, 75h,  14h
db 10h, 43h, 17h
db 18h, 43h, 17h
db 0F3h, 68h, 14h
db 0F2h, 68h, 14h
db 0F3h, 68h, 14h
db 0F2h, 68h, 14h
db 0F3h, 68h, 14h          ;170
db 0CBh, 0C4h, 14h
db 0C3h, 0C4h, 14h
db 0, 43h, 17h
db 8, 43h, 17h
db 9Eh, 68h, 14h
db 38h, 43h, 17h
db 0AEh, 68h, 14h
db 0AFh, 68h, 14h
db 20h, 43h, 17h
db 28h, 43h, 17h        ;180
db 0F9h, 68h, 14h
db 0FDh, 68h, 14h
db 0FBh, 68h, 14h
db 0AAh, 68h, 14h
db 0ABh, 68h, 14h
db 0F6h, 6Eh, 17h
db 9Bh, 68h, 14h
db 86h, 6Ah, 17h
db 0D7h, 68h, 14h
db 26h, 68h, 14h       ;190
db 2Eh, 68h, 14h
db 36h, 68h, 14h
db 3Eh, 68h, 14h

db 84h, 2Bh, 88h
db 2Bh, 59h, 2Bh
db 5Dh, 2Bh, 0A1h
db 2Bh, 0A5h, 2Bh
db 0E5h, 2Bh, 91h
db '+5(I(9(A(L(=(E('
db 85h, 28h, 0C1h, 2Bh, 0E5h
db '+=+5+I+K*'
db 99h, 28h, 46h, 2Ah, 50h, 2Ah
db 95h, 28h, 60h, 28h, 60h, 28h
db 0B7h, 2Ah, 0B7h, 2Ah, 54h, 2Bh
db 0E5h
db 2Bh
sreg_save_addr  dw offset es_save
                dw offset cs_save
                dw offset ss_save
                dw offset ds_save
reg_name        db 'AX',0, 'BX',0, 'CX',0, 'DX',0
                db 'SP',0, 'BP',0, 'SI',0, 'DI',0
                db 'DS',0, 'ES',0, 'SS',0, 'CS',0
                db 'IP',0, 'PC',0
flag_1_str      db 8 dup(0),'OVDNEI',0,0,'NGZR',0,0,'AC',0,0,'PE',0,0,'CY'
flag_0_str      db 8 dup(0),'NVUPDI',0,0,'PLNZ',0,0,'NA',0,0,'PO',0,0,'NC'
                db 256 dup (0)
ax_h_save       db 0
ax_l_save       db 0
bx_save         dw 0
cx_save         dw 0
dx_save         dw 0
sp_save         dw 5Ah
bp_save         dw 0
si_save         dw 0
di_save         dw 0
ds_save         dw 0
es_save         dw 0
ss_save         dw 0
cs_save         dw 0
; 1073, 1085, 10FD, 119E, 12D4, 12E4, 130E, 1349, 13CD, 1E03
ip_save         dw 100h
  ; 109B,10AE, 10D1, 10DE, 10F9, 11A2, 12DC, 1351
flags_save     dw 0F202h
       ;           1109, 112E, 1140, 119A, 11F2, 11F7
data_176        db 0
data_177        db 0
RWfunc_no       db 3Fh
reg_num         db 0Dh
dispc_per_line  db 0Fh
reg_per_line    db 8
disp_chars      dw 80h
key_buffer      db 50h, 0, 0Dh, 82 dup (0)
data_183        db 0
data_184        db 0
```

```
data_185          dw 0
data_186          dw 0
work_seg          dw 0
work_ofs          dw 0
data_189          dw 0
                  dw 0
                  db 80h, 0
data_190          dw 0
                  db 5Ch, 0
data_191          dw 0
                  db 6Ch
                  db 0
data_193          dw 0
data_194          dd 00000h
data_195          dd 00000h
ret_str           db 0Dh, 0Ah, 0
                  dw offset ret_str
tab_str           db 20h, 8, 0
                  dw offset tab_str
ver_err_str       db ' Incorrect DOS version $ '
end_str           db 0Dh, 0Ah, ' Program terminated normally' ,0
                  dw offset end_str
drv_no_str        db ' Invalid drive specification' ,0
                  dw offset drv_no_str
file_no_str       db ' File not found' ,0
                  dw offset file_no_str
file_err_str      db ' File creation error' ,0
                  dw offset file_err_str
disk_err_str      db ' Insufficient spa~~ee on disk~~' ,0
                  dw offset disk_err_str
disk_str          db ' Disk' ,0
protect_str       db ' Write protect' ,0
drive_char        db ' A' ,0
read_str          db ' reading' ,0
write_str         db ' writing' ,0
mem_err_str       db ' Insufficient memory' ,0
                  dw offset mem_err_str
str_str           db ' %s'
                  dw 205Eh
error_str         db ' Error' ,0
```

```
                     dw offset str _ str, 37A2h
exe _ err _ str      db ' Error in EXE or HEX file' ,0
                     dw offset exe _ err _ str
exe _ wrt _ err _ str db ' EXE and HEX files cannot be written' ,0
                     dw offset exe _ wrt _ err _ str
exec _ err _ str     db ' EXEC failure' ,0
                     dw offset exec _ err _ str
dest _ no _ err _ str db ' (W)rite error, no destination defined' ,0
                     dw offset dest _ no _ err _ str
access _ err _ str   db ' Access denied' ,0
                     dw offset access _ err _ str
parity _ err _ str  db ' Parity error or nonexistant memory error detected' ,0
                     dw offset parity _ err _ str
token _ str          db ' —' , 0
                     dw offset token _ str
str _ token _ str    db ' %s —' ,0
                     dw 33BEh, 37A2h
df _ str             db ' df' ,0
bf _ str             db ' bf' ,0
br _ str             db ' br' ,0
bp _ str             db ' bp'
seg _ c ends

stack _ seg _ d segment para stack
db 362 dup (0)
stack _ seg _ d ends

seg _ f segment para public
db 74 dup(0)
dw 100h
db 598 dup(0)
dw 37A2h, 0000h, 37F4h
db 81 dup(0)
db ' %S%S' ,00h
dw 3849h, 37A2h, 37F8h
db ' %04X:%04X %s' ,0
dw 3854h, 0000h, 0000h, 37A2h, 7 dup(0)
db ' %04X   %04X' ,00h
dw 3877h, 0000h, 0000h
db ' %s %04X' ,0Dh,0Ah,' :' , 00h
```

IIA 8-70

```
dw 3888h, 37A2h, 0000h
db ' %s=%04X  ',00h
dw 3899h, 0000h, 0000h
db ' %s Error',00h
dw 38A9h, 0000h
db 'Writing %04LX bytes',00h
dw 38B6h, 0000h, 0000h
db ' %s:%04X=',00h
dw 38D0h, 37A2h, 0000h
db ' %02X',00h
dw 38DFh, 0000h
db ' %04X',00h
dw 38E8h, 0000h
db ' %04X:%04X  %02X  %02X  %04X:%04X',00h
dw 38F1h, 6 dup(0)
db ' %s error %s drive %c',00h
dw 3920h, 2 dup(0), 32C8h
seg_f  ends

int01_ofs  equ 4
int01_seg  equ 6
int03_ofs  equ 0Ch
int03_seg  equ 0Eh
Param_Len  equ 80h
int23h_ofs  equ 8Ch
int23h_seg  equ 8Eh
data_9e  equ 6B5h
data_10e equ 0F05h
data_13e equ 38A5h
data_14e equ 38A7h
data_15e equ 6E20h
data_16e equ 7620h
data_17e equ 7A20h
data_18e equ 7E20h
data_19e equ 9120h
data_20e equ 9720h
data_21e equ 9B20h
data_22e equ 0A503h
data_23e equ 0A720h
data_24e equ 0AB20h
```

```
data_25e equ 0AD20h
data_26e equ 0B506h
data_27e equ 0BB20h
data_28e equ 0C320h
data_29e equ 0C720h
data_30e equ 0CB20h
data_31e equ 0CF20h
data_32e equ 0D320h
data_33e equ 0DA20h
int22h_ofs equ 0Ah
int22h_seg equ 0Ch
data_37e equ 373h
data_38e equ 3E1h
data_39e equ 228Dh
data_40e equ 2413h          ;offset reg_s_str
data_41e equ 2D87h
data_52e equ 3550h
data_53e equ 3575h
data_55e equ 357Bh
data_56e equ 357Dh
data_74e equ 37A2h
data_75e equ 3863h
data_76e equ 3865h
data_77e equ 38CCh
data_78e equ 38CEh
data_79e equ 38DDh
data_80e equ 38E6h
data_81e equ 38EFh
data_199e equ 354Ah
data_200e equ 354Bh
data_201e equ 354Fh
data_203e equ 3551h
data_204e equ 3553h
data_205e equ 3555h
data_206e equ 3557h
data_207e equ 3559h
data_208e equ 355Bh
data_209e equ 355Dh
data_210e equ 3574h
data_212e equ 3576h
```

```
data _ 213e equ 3577h
data _ 214e equ 3579h
data _ 215e equ 357Fh
data _ 216e equ 3580h
data _ 217e equ 3582h
data _ 218e equ 3584h
data _ 219e equ 3586h
data _ 220e equ 3588h
data _ 221e equ 358Ah
data _ 222e equ 358Ch
data _ 223e equ 358Eh
data _ 224e equ 3590h
data _ 225e equ 3592h
data _ 226e equ 359Ch
data _ 227e equ 35A2h
data _ 228e equ 35A3h
data _ 229e equ 35A5h
data _ 230e equ 35A7h
data _ 231e equ 35A8h
data _ 232e equ 35A9h
data _ 233e equ 35AAh
data _ 234e equ 35ABh
data _ 235e equ 35ACh
data _ 236e equ 35ADh
data _ 237e equ 35AFh
data _ 238e equ 35B1h
data _ 239e equ 35B2h
data _ 240e equ 35B3h
data _ 241e equ 3608h
data _ 242e equ 360Ah
data _ 243e equ 360Ch
data _ 244e equ 363Ah
data _ 245e equ 363Bh
data _ 246e equ 363Ch
data _ 247e equ 363Dh
data _ 248e equ 363Eh
data _ 249e equ 363Fh
data _ 250e equ 3640h
data _ 251e equ 3641h
data _ 252e equ 3642h
data _ 253e equ 3643h
data _ 254e equ 3644h
data _ 255e equ 3645h
data _ 256e equ 3647h
data _ 257e equ 3649h
data _ 258e equ 364Ah
data _ 259e equ 3863h
data _ 260e equ 3865h
data _ 261e equ 3869h
data _ 262e equ 386Bh
data _ 263e equ 3873h
data _ 264e equ 3875h
data _ 265e equ 3884h
data _ 266e equ 3886h
data _ 267e equ 3897h
data _ 268e equ 38B4h
data _ 269e equ 3914h
data _ 270e equ 3916h
data _ 271e equ 3918h
data _ 272e equ 391Ah
data _ 273e equ 391Ch
data _ 274e equ 391Eh
data _ 275e equ 3937h
data _ 276e equ 3939h
data _ 277e equ 4423h
data _ 278e equ 0C11Fh
data _ 285e equ 6437h
data _ 291e equ 355Eh
data _ 292e equ 3560h
data _ 293e equ 3562h
data _ 294e equ 3564h
data _ 295e equ 3566h
data _ 296e equ 3568h
data _ 297e equ 356Ah
data _ 298e equ 356Ch
end      start
Segments and Groups:
                 N a m e                   Length   Align  Combine Class
SEG _ A . . . . . . . . . . . . .          0010     PARA   PUBLIC
SEG _ B . . . . . . . . . . . . .          0270     PARA   PUBLIC
SEG _ C . . . . . . . . . . . . .            33D2    PARA  PUBLIC
SEG _ F . . . . . . . . . . . . .            03ED    PARA  PUBLIC
STACK _ SEG _ D . . . . . . . . . .          016A    PARA  STACK
```

# 附录B.　扩充用初始模块

```
        INCLUDE RULES. ASI
DGROUP GROUP _STACK, _DATA, _BSS, _BSSEND
        ASSUME      CS:_TEXT, DS:DGROUP
        SUBTTL      Start Up Data Area
        PAGE
        External References

ExtProc@            main,     __CDECL__
PSPHigh             equ       00002h
PSPEnv              equ       0002ch
PSPCmd              equ       00080h
;         At the start, DS and ES both point to the segment prefix.
;         SS points to the stack segment except in TINY model where
;         SS is equal to CS
;
_TEXT   SEGMENT
        ORG         0
IFDEF   __TINY__
        ORG         100h
ENDIF
STARTX              PROC      NEAR
IFDEF   __TINY__
                    mov       dx, cs             ;DX = GROUP Segment address
ELSE
                    mov       dx, DGROUP         ;DX = GROUP Segment address
ENDIF
                    mov       cs:DGROUP@@, dx
                    mov       bp, ds:[PSPHigh]  ;BP = Highest Memory Segment Addr
                    mov       bx, ds:[PSPEnv ]  ;BX = Environment Segment address
                    mov       ds, dx
                    mov       _Init_CS@, cs
                    mov       _psp@, es         ; Keep Program Segment Prefix address
                    mov    ax, seg main@
                    push ax
                    mov    ax, offset main@
                    push ax
```

```
                        retf
STARTX ENDP
PubSym@ DGROUP@, <dw      ? >, __PASCAL__
_TEXT       ENDS
_DATA       SEGMENT word public 'DATA'
;           The CopyRight string must NOT be moved or changed without
;           changing the null pointer check logic
CopyRight               db              4 dup(0)
lgth_CopyRight          equ             $ - CopyRight
PubSym@                 _psp,           <dw     0>,        __CDECL__
PubSym@                 _Init_CS,       <dw     0>,        __CDECL__
IF      LDATA EQ false
        IFNDEF  __NOFLOAT__
;                                       Emulator variables
                INCLUDE emuvars.asi
         ENDIF
ENDIF
_DATA           ENDS
_BSS            SEGMENT   word public 'BSS'
bdata@          label     byte
_BSS            ENDS
_STACK          SEGMENT   STACK 'STACK'
                dw        512 dup (0)
_STACK          ENDS
_BSSEND         SEGMENT
edata@ label byte
_BSSEND         ENDS
                END       STARTX
```

# 附录C.　扩展 COM 文件源程序

```
/*************************************************/
/* ENCRYCOM.C                                    */
/* 扩展 COM 文件的主程序                          */
/* 种子文件为 SEEDCOM.C,初始模块为 CS.OBJ          */
/*************************************************/
char
        cc, *fn1="seedcom.exe", fn2[80], fn3[80];
unsigned int
        head_len, ins_seg, reloc_tbl_items;
unsigned long
        fsize, ins_len, com_len, fill_len, loop;
FILE
        *fp1, *fp2, *fp3;
const
        Fill_Byte = 'FF',
        size_ofs = 2,
        head_ofs = 0x08,
        reloc_tbl_item_ofs = 6,
        ss_ofs = 0x0e,
        cs_ofs = 0x16,
        reloc_tbl_ofs = 0x18;
main()

        printf("\n\nEncrypt COM Files");
        printf("Source COM File: ");
        scanf(" %s", fn3);
        printf("Target EXE File: ");
        scanf(" %s", fn2);
        if( ((fp1=fopen(fn1,"rb"))   == NULL) ||
           ((fp3=fopen(fn3,"rb"))   == NULL) ||
           ((fp2=fopen(fn2,"wb"))   == NULL) ) {
                printf("\n\nopen file failure !");
                exit(1);
                }
        fseek(fp3, 0, SEEK_END);
            com_len = ftell( fp3 );
```

```
rewind( fp3 );
if( ( (com _ len / 16l) * 16l ) ! = com _ len
    ins _ len = (com _ len/16l + 1l) * 16l;
else ins _ len = com _ len;
ins _ seg = ins _ len >> 4;
1l _ len = ins _ len - com _ len;
eek(fp1, head _ ofs, SEEK _ SET);
ad _ len = getw( fp1 ) << 4;
wind( fp1 );
eek(fp1, 0, SEEK _ END);
ize = ftell( fp1 ) - head _ len;
wind( fp1 );
intf("\ncopying head ( %u ) ...", head _ len);
r(loop=0; loop<head _ len; loop++)   putc(getc(fp1), fp2);
intf("\nappending com _ codes ( %lu ) ...", com _ len);
r(loop=0; loop<com _ len; loop++) putc(getc(fp3), fp2);
intf("\nappending fill _ codes ( %lu ) ...", fill _ len);
r(loop=0; loop<fill _ len; loop++) putc(Fill _ Byte, fp2);
printf("\nappending image _ codes ( %lu ) ...", fsize);
r(loop=0; loop<fsize; loop++) putc(getc(fp1), fp2);
fclose(fp1);
fclose(fp3);

{ unsigned int hi ,lo, d;
          printf("\nmodifying size _ double _ word ...");
           fseek(fp2, 0, SEEK _ END);
           fsize = ftell( fp2 );
           d = fsize / 512;
           hi = ((d * 512) ~~== fsize~~) ? d : d+1;
           lo = ((d * 512) == fsize) ? 0 ; fsize - d * 512;
           fseek(fp2, size _ ofs, SEEK _ SET);
           putw(lo, fp2);
           putw(hi, fp2);
}

{ unsigned int ss;
           printf("\nmodifing ss ...");
           fseek(fp2, ss _ ofs, SEEK _ SET);
           ss = getw( fp2 ) + ins _ seg;
           fseek(fp2, ss _ ofs, SEEK _ SET);
          putw(ss, fp2);
    }

    { unsigned int cs;
          printf("\nmodifing cs ...");
          fseek(fp2, cs_ofs, SEEK_SET);
          cs = getw( fp2 ) + ins_seg;
          fseek(fp2, cs_ofs, SEEK_SET);
          putw(cs, fp2);
    }

    { unsigned int seg_value, ofs, reloc_tbls, reloc_tbl_addr, loop;
          fseek(fp2, reloc_tbl_item_ofs, SEEK_SET);
          reloc_tbls = getw( fp2 );
          printf("\nmodifing relocate_table_items ( %u ) ...", reloc_tbls);
          fseek(fp2, reloc_tbl_ofs, SEEK_SET);
          reloc_tbl_addr = getw( fp2 );
          for(loop=0; loop<reloc_tbls; loop++) {
                ofs = reloc_tbl_addr + loop * 4 + 2;
                fseek(fp2, ofs, SEEK_SET);
                seg_value = getw( fp2 ) + ins_seg;
                fseek(fp2, ofs, SEEK_SET);
                putw(seg_value, fp2);
          }
    }

    { unsigned int hi, lo, value, reloc_tbls, reloc_tbl_addr, head_size, loop;
          unsigned long ofs;
          fseek(fp2, head_ofs, SEEK_SET);
          head_size = getw( fp2 ) << 4;
          fseek(fp2, reloc_tbl_item_ofs, SEEK_SET);
          reloc_tbls = getw( fp2 );
          printf("\nmodifing relocate tables ( %u ) ....", reloc_tbls);
          fseek(fp2, reloc_tbl_ofs, SEEK_SET);
          reloc_tbl_addr = getw( fp2 );
          for(loop=0; loop<reloc_tbls; loop++) {
                fseek(fp2, reloc_tbl_addr+loop * 4, SEEK_SET);
                lo = getw( fp2 );
                hi = getw( fp2 );
                ofs = hi << 4 + lo + head_size;
```

```
                        fseek(fp2, ofs, SEEK _ SET);
                        value = getw( fp2 ) + ins _ seg;
                        fseek(fp2, ofs, SEEK _ SET);
                        putw(value, fp2);
                }
        }
        fclose(fp2);
}


/*********************************************/
/* SEEDCOM.C                                   */
/* 扩展 COM 文件的种子程序                      */
/* 扩展程序为 ENCRYCOM.C,初始模块为 CS.OBJ      */
/*********************************************/

far main()
{
        asm pop   ax
        asm pop   ax
        asm mov   ah, 62h
        asm int   21h /* get psp to bx */
        asm mov   ds, bx
        asm mov   es, bx
        asm mov   ss, bx
        asm mov   sp, 0FFFEh
        asm push  bx
        asm mov   ax, 100h
        asm push  ax
        asm mov   ax, 0
        asm mov   bx, 0
        asm retf
}
```

# 附录 D.　扩展 EXE 文件的源程序

```
/ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * /
/ *  SEEDL. C                                                             * /
/ *  扩展 EXE 文件的主程序 1,种子程序将附加到源程序的后面                  * /
/ *  种子程序为 SEEDEXE1. C,初始模块为 C0S. OBJ                           * /
/ * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * /

/ *  encrypt EXE file
     seeded file is ret—exe. EXE
     source EXE file is SRC. EXE
     target encrypted file is TGT. EXE
     algorithm:
     1. copy head _ code of RET—EXE. EXE to TGT. EXE
     2. append image _ code of SRC. EXD to TGT. EXE
     3. append Fille _ Bytes to TGT. EXE if necessary
     4. append head _ code of SRC. EXE to TGT. EXE
        FORMAT :
        reloc _ tbls ofs   :word, seg:word ,...
        ss                 : word
        sp                 : word
        cs                 : word
        ip                 : word
        image _ len        : double word
        reloc _ items      : word

        the length is      : 5 * 2 + reloc _ items * 4
                           = 14 + reloc _ items * 4

     5. append image _ code of RET—EXE. EXE to TGT. EXE
     6. append appended _ codes of SRC. EXE to TGT. EXE if necessary
     7. modify TGT. EXE (ref. the following), head _ length includes the head _ code of SRC.
        EXE, but not includes the appended _ code _ length
     8. RET—EXE: relocate image _ code of SRC. EXE with head _ code of SRC. EXE then move
        the image _ code of SRC. EXE to started _ segment
* /

/ *  insert code to exe file start from head  * /
/ * algorithm:
        1. copy head code of src. exe to tag. exe
           head _ len = word( ofs 0x02 )
        2. append inserted code to tag. exe, you should make sure the inserted _ code' s length is times
           of 0x10 suppose the propsed length is l
                            l _ seg = l >> 4
        3. append image code of src. exe to tag. exe
        4. modify tag. exe
           a. modify size _ double _ word at offset 2 to 5,
              suppose (file _ size / 512) = x. . y
                        word( ofs 4 ) <== (y ! =0 ) ? x+1 : x
                        word( ofs 2 ) <== (y ! =0 ) ? (file _ size-x * 512) : 0
           b. modify ss
              word( ofs 0x0e ) += l _ seg
           c. modify cs
              word( ofs 0x16 ) += l _ seg
           d. modify relocate _ tabel _ items
              suppose
                        the relocate tabel items tbl _ items = word( ofs 6 )
                        the first _ relocate _ tabel _ items' address
                          tbl _ addr = word( ofs 0x18 )
              then loop tbl _ items do
                        word( ofs (tbl _ addr+4 * (z-1)+2) ) += l _ seg, z=1. . tbl _ items
           e. modify relocate tabel
              loop tbl _ items do
                        word( word(ofs(tbl _ addr+4 * (z-1)+2))<<4 +word(ofs tbl _ addr
                              +4 * (z-1)) +head _ leng )+= l _ seg, z=1. . tbl _ items
* /

#include <stdio. h>
#include <alloc. h>

/ * common varibles * /
char
          cc;
unsigned int
          ins _ seg, buf _ size;
unsigned long
          ins _ len, fill _ len, loop;
```

```
/* seeded file : RET.EXE */
char
        fn1[80], *buf1;
unsigned
        head_len1, reloc_tbl_items1;
unsigned long
        fsize1, image_len1;
FILE
         *fp1;

/* source exe file : SCR.EXE */
char
        fn3[80], *buf3;
unsigned
        head_len3, reloc_tbl_items3, reloc_tbl_addr3, ss3, sp3, cs3, ip3;
unsigned long
        head_fsize3, dos_fsize3, image_len3;
FILE
         *fp3;

/* target exe file : TGT.EXE */
char
        fn2[80], *buf2;
FILE
         *fp2;

const
        Fill_Byte = 'FF',
        size_ofs = 2,
        head_ofs = 0x08,
        reloc_tbl_item_ofs = 6,
        ss_ofs = 0x0e,
        sp_ofs = 0x10,
        cs_ofs = 0x16,
        ip_ofs = 0x14,
        append_head = 14,
        reloc_tbl_ofs = 0x18;

main()
{
```

```
printf("\n\nEncrypt EXE Files which has no any appended _ codes");
printf("\n\ninserts codes to exe file starting from image _ code");
printf("\n\nSeeded EXE File(SEED1. EXE): ");
scanf(" %s", fn1);
printf("Source EXE File: ");
scanf(" %s", fn3);
printf("Target EXE File: ");
scanf(" %s", fn2);

if( ((fp1=fopen(fn1,"rb")) == NULL) ||
    ((fp3=fopen(fn3,"rb")) == NULL) ||
    ((fp2=fopen(fn2,"wb")) == NULL) ) {
         printf("\n\nopen file failure !");
         exit(1);
}
buf _ size = (coreleft() - 512) / 2;
buf2 = malloc( buf _ size );
buf3 = malloc( buf _.size );
setvbuf(fp3, buf3, _ IOFBF, buf _ size);
setvbuf(fp2, buf2, _ IOFBF, buf _ size);

{ unsigned int hi, lo;


          fseek(fp3, 0, SEEK _ END)
          dos _ fsize3 = ftell( fp3 );
          rewind(fp3);

          fseek(fp3, size _ ofs, SEEK _ SET);
          lc = getw(fp3);
          hi = getw(fp3);
          if(lo == 0)
                    head _ fsize3 = (unsigned long) hi * 512;
          else
                    head _ fsize3 = (unsigned long) (hi-1) * 512 + lo;
          if(head _ fsize3 ! = dos _ fsize3) {
                    printf("\n\nsource EXE file has appended codes %lu bytes", \
                              dos _ fsize3-head _ fsize3);
          }

          fseek(fp3, ss _ ofs, SEEK _ SET);
          ss3 = getw(fp3);

          fseek(fp3, sp_ofs, SEEK_SET);
          sp3 = getw(fp3);

          fseek(fp3, cs_ofs, SEEK_SET);
          cs3 = getw(fp3);

          fseek(fp3, ip_ofs, SEEK_SET);
          ip3 = getw(fp3);

          fseek(fp3, reloc_tbl_item_ofs, SEEK_SET);
          reloc_tbl_items3 = getw(fp3);

          fseek(fp3, reloc_tbl_ofs, SEEK_SET);
          reloc_tbl_addr3 = getw(fp3);

          fseek(fp3, head_ofs, SEEK_SET);
          head_len3 = getw(fp3) << 4;
          image_len3 = head_fsize3 - head_len3;
      }

      { unsigned long x;

          x = append_head + reloc_tbl_items3  * 4 + image_len3;
          if( ( (x / 16l) * 16l ) ! = x )
                ins_len = (x/16l + 1l) * 16l,
          else ins_len = x;
          ins_seg = ins_len >> 4;
          fill_len = ins_len - x;
      }

      fseek(fp1, head_ofs, SEEK_SET);
      head_len1 = getw( fp1 ) << 4;
      rewind( fp1 );
      fseek(fp1, 0, SEEK_END);
      fsize1 = ftell( fp1 );
      image_len1 = fsize1 - head_len1;
      rewind( fp1 );
```

12A 8-70

```
printf("\ncopying seeded_head ( %u ) ...", head_len1);
rewind(fp1);
rewind(fp2);
for(loop=0; loop<head_len1; loop++)  putc(getc(fp1), fp2);

printf("\nappending src_image_codes ( %lu ) ...", image_len3);
fseek(fp3, head_len3, SEEK_SET);
for(loop=0; loop<image_len3; loop++) putc(getc(fp3), fp2);

printf("\nappending fill_codes ( %lu ) ...", fill_len);
for(loop=0; loop<fill_len; loop++) putc(Fill_Byte, fp2);

printf("\nappending src_head ( %u ) ...", 0x0e+reloc_tbl_items3 * 4);
fseek(fp3, reloc_tbl_addr3, SEEK_SET);
for(loop=0; loop<reloc_tbl_items3; loop++)  {
        putw(getw(fp3), fp2);
        putw(getw(fp3), fp2);
}
putw(ss3, fp2);
putw(sp3, fp2);
putw(cs3, fp2);
putw(ip3, fp2);
putw((unsigned int)image_len3, fp2);
putw((unsigned int) (image_len3 >> 16), fp2);
putw(reloc_tbl_items3, fp2);

printf("\nappending seeded_image_codes ( %lu ) ...", image_len1);
fseek(fp1, head_len1, SEEK_SET);
for(loop=0; loop<image_len1; loop++) putc(getc(fp1), fp2);

{ unsigned long x;
        x = dos_fsize3 - head_fsize3;

        printf("\nappending src_appended_codes ( %lu ) ...", x);
        fseek(fp3, head_fsize3, SEEK_SET);
        for(loop=0; loop<x; loop++) putc(getc(fp3), fp2);
}

fclose(fp1);
fclose(fp2);
fclose(fp3);
free(buf3);
free(buf2);

buf_size += buf_size;
buf2 = malloc(buf_size);
fp2 = fopen(fn2, "rb+");
setvbuf(fp2, buf2, _IOFBF, buf_size);

/* a. modify size_double_word */
{ unsigned int hi ,lo;
        unsigned long d, size;
        printf("\nmodifying size_double_word ...");
        size = head_len1 + ins_len + image_len1;
        d = size / 512l;
        hi = ((d * 512) == size) ? d : d+1;
        lo = ((d * 512) == size) ? 0 : size - d * 512;
        fseek(fp2, size_ofs, SEEK_SET);
        putw(lo, fp2);
        putw(hi, fp2);
}

/* b. modify ss */
{ unsigned int ss;
        printf("\nmodifing ss ...");
        fseek(fp2, ss_ofs, SEEK_SET);
        ss = getw( fp2 );
        ss += ins_seg;
        fseek(fp2, ss_ofs, SEEK_SET);
        putw(ss, fp2);
}

/* c. modify cs */
{ unsigned int cs;
        printf("\nmodifing cs ...");
        fseek(fp2, cs_ofs, SEEK_SET);
        cs = getw( fp2 );
        cs += ins_seg;
        fseek(fp2, cs_ofs, SEEK_SET);
        putw(cs, fp2);
```

```
}

/* d. modify relocate_tabel_items */
{ unsigned int seg_value, ofs, reloc_tbls, reloc_tbl_addr, loop;
        fseek(fp2, reloc_tbl_item_ofs, SEEK_SET);
        reloc_tbls = getw( fp2 );
        printf("\nmodifing relocate_table_items ( %u ) ...", reloc_tbls);
        fseek(fp2, reloc_tbl_ofs, SEEK_SET);
        reloc_tbl_addr = getw( fp2 );
        for(loop=0; loop<reloc_tbls; loop++) {
                ofs = reloc_tbl_addr + loop * 4 + 2;
                fseek(fp2, ofs, SEEK_SET);
                seg_value = getw( fp2 );
                seg_value += ins_seg;
                fseek(fp2, ofs, SEEK_SET);
                putw(seg_value, fp2);
        }
}

/* e. modify relocate tabel */
{ unsigned int hi, lo, value, reloc_tbls, reloc_tbl_addr, head_size, loop;
        unsigned long ofs;
        fseek(fp2, head_ofs, SEEK_SET);
        head_size = getw( fp2 ) << 4;
        fseek(fp2, reloc_tbl_item_ofs, SEEK_SET);
        reloc_tbls = getw( fp2 );
        printf("\nmodifing relocate tables ( %u ) ...", reloc_tbls);
        fseek(fp2, reloc_tbl_ofs, SEEK_SET);
        reloc_tbl_addr = getw( fp2 );
        for(loop=0; loop<reloc_tbls; loop++) {
                fseek(fp2, reloc_tbl_addr+loop * 4, SEEK_SET);
                lo = getw( fp2 );
                hi = getw( fp2 );
                ofs = hi;
                ofs = ofs << 4;
                ofs += lo;
                ofs += head_size;
                fseek(fp2, ofs, SEEK_SET);
                value = getw( fp2 );
                value += ins_seg;
```

```
                    fseek(fp2, ofs, SEEK_SET);
                    putw(value, fp2);
                }
        }

        printf("%s", "\nOk !");
        free( buf2 );
        fclose( fp2 );


/************************************/
/* SEED1.C                                             */
/* 扩展 EXE 文件的种子程序 1,SEEDEXE1 将附加到源程序的后面     */
/* 扩展程序为 ENCRY1.C,初始模块为 C0—SEED.OBJ             */
/************************************/

extern _Init_CS;
unsigned
        ss_h, sp_h, cs_h, ip_h, reloc_items_h, seg, ofs, loop;
unsigned long
        addr, start_CS, image_len_h;
const
        ss_h_ofs = 14,
        sp_h_ofs = 12,
        cs_h_ofs = 10,
        ip_h_ofs = 8,
        image_len_h_ofs = 6,
        reloc_items_h_ofs = 2,
        append_head = 14,
        fpsp_ofs = 0x16;
main()
{
  /* get parameters */
        start_CS = ((unsigned long)_Init_CS) << 4;
        ss_h = *((int far *)(start_CS - ss_h_ofs));
        sp_h = *((int far *)(start_CS - sp_h_ofs));
        cs_h = *((int far *)(start_CS - cs_h_ofs));
        ip_h = *((int far *)(start_CS - ip_h_ofs));
        reloc_items_h = *((int far *)(start_CS - reloc_items_h_ofs));
        addr = start_CS - image_len_h_ofs;
        image_len_h = *((int far *)(start_CS - image_len_h_ofs + 2));
        image_len_h = image_len_h << 16;
        image_len_h += *((int far *)(start_CS - image_len_h_ofs));
/* Relocate image */
        start_CS = (unsigned long)_Init_CS << 4;
        addr = start_CS - append_head - (reloc_items_h << 2);
        for(loop=0; loop<reloc_items_h; loop++) {
                seg = peek(addr >> 4, addr & 0x0f +2) + psp + 0x10;
                ofs = peek(addr >> 4, addr & 0x0f);
                poke(seg, ofs, peek(seg, ofs) + psp + 0x10);
                addr += 4;
        }
/* goto EXE */
        asm mov  ax, ss_h
        asm add  ax, _psp
        asm add  ax, 10h
        asm mov  ss, ax
        asm mov  sp, sp_h
        asm mov  ax, cs_h
        asm add  ax, _psp
        asm add  ax, 10h
        asm push ax
        asm mov  ax, ip_h
        asm push ax
        asm mov  ax, _psp
        asm mov  ds, ax
        asm mov  es, ax
        asm mov  ax, 0
        asm retf
```

# 附录 E.　给 EXE 文件加口令的源程序

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*  PASSEXEb. C                                                  */
/*  给 EXE 文件加口令的主程序                                    */
/*  扩充 EXE 文件的主程序 2,种子程序将插入源程序前面             */
/*  种子程序为 EXEPASSb. c,初始模块是 C0S. OBJ                   */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

/*  encrypt EXE file
    seeded file is RET-EXEr. EXE
    source EXE file is SRC. EXE
    target  encrypted file is TGT. EXE
    algorithm:
    1. copy head _ code of RET-EXE. EXE to TGT. EXE
    2. append head _ code of SRC. EXE to TGT. EXE
       FORMAT :
       reloc _ tbls ofs:word, seg:word ,. . .
                                  , cs:ip[-(26+reloc _ items * 4+fill _ bytes _ len)]
       fill _ bytes : . . .
       XOR _ src _ start: double word, cs:ip[-26]
       XOR _ src _ len: double word , cs:ip[-22]
      ss            :word , cs:ip[-18] , 08 + reloc _ items * 4+fill _ len1
      sp            :word , cs:ip[-16] , 10 + reloc _ items * 4+fill _ len1
      cs            :word , cs:ip[-14] , 12 + reloc _ items * 4+fill _ len1
      ip            :word , cs:ip[-12] , 14 + reloc _ items * 4+fill _ len1
      reloc _ items :word , cs:ip[-10] , 16 + reloc _ items * 4+fill _ len1
      seed _ len    :double word , cs:ip[-8]
      source _ len  :double word , cs:ip[-4]

      cs:ip is the address of the first instruction
      Dx = 26+reloc _ items * 4
      fill _ bytes _ len = ( (x >> 4) << 4 ) == x ? \
                                0 : (x >> 4 +1 ) << 4 -- x
      source _ image _ start = 26 + reloc _ items * 4 + fill _ bytes _ len + \
                                  seed _ len + fill _ len2
    3. append Fill _ Bytes if necessary
    4. append image _ code of RET-EXE. EXE to TGT. EXE
```

```
        5. append Fill _ Bytes to TGT. EXE if necessary
        6. append image _ code of SRC. EXD to TGT. EXE
        7. append appended _ codes of SRC. EXE to TGT. EXE if necessary
        8. modify the part RET—EXE. EXE of TGT. EXE (ref. the following),
           head _ length includes the head _ code of SRC. EXE,
           but not includes the appended _ code _ length
        9. modify the part SRC. EXE of TGT. EXE (head, reloc _ tbl _ items)
           a. RET—EXE: relocate image _ code of SRC. EXE with head _ code of SRC. EXE then
              move the image _ code of SRC. EXE to started _ segment
  */

 /* insert code to exe file start from head */
 /* algorithm:
        1. copy head code of src. exe to tag. exe
           head _ len = word( ofs 0x02 )
        2. append inserted code to tag. exe, you should make sure the inserted _ code' s length is
           times of 0x10 suppose the propsed length is l _ seg = l >> 4
        3. append image code of src. exe to tag. exe
        4. modify tag. exe
           a. modify size _ double _ word at offset 2 to 5, suppose (file _ size / 512) = x.. y
              word( ofs 4 ) <== (y ! =0 ) ? x+1 : x word( ofs 2 ) <== (y ! =0 ) ?
              (file _ size—x * 512) : 0
           b. modify ss
              word( ofs 0x0e ) += l _ seg
           c. modify cs
              word( ofs 0x16 ) += l _ seg
           d. modify relocate _ tabel _ items
              suppose
                   the relocate tabel items tbl _ items = word( ofs 6 )
                   the first _ relocate _ tabel _ items' address
                        tbl _ addr = word( ofs 0x18 )
              then loop tbl _ items do
                   word( ofs (tbl _ addr + 4 * (z — 1) + 2) ) += l _ seg, z= 1.. tbl _ items
           e. modify relocate tabel
              loop tbl _ items do
                   word( word(ofs(tbl _ addr+4 * (z—1)+2))<<4 +word(ofs tbl _ addr + 4
                              * (z—1)) +head _ leng )+ = l _ seg, z=1.. tbl _ items
 */
 #include <dir. h>
 #include <io. h>
```

```
#include <dos.h>
#include <string.h>
#include <stdio.h>
#include <alloc.h>
#ifndef DEBUG
#include "password.c"
#endif

/* common varibles */
char
        Old_Password[40], XOR_PASS = 0xaa,
        cc;
unsigned int
        ins_seg, buf_size;
unsigned long
        ins_len1, ins_len, fill_len1, fill_len2, loop;

/* seeded file : RET.EXE */
char
        fn1[80], *buf1;
unsigned
        head_len1, reloc_tbl_items1;
unsigned long
        fsize1, image_len1;
FILE
        *fp1;

/* source exe file : SCR.EXE */$char
        fn3[80], *buf3, *tmpname = "x$x$x$x$.$x$"; $unsigned
        head_len3, reloc_tbl_items3, reloc_tbl_addr3, ss3, sp3, cs3, ip3;
unsigned long
        head_fsize3, dos_fsize3, image len3;
FILE
        *fp3;

/* target exe file : TGT.EXE */
char
        fn2[80], *buf2;
FILE
        *fp2;
```

```
/* XOR */
unsigned
        XOR_v = 0x55aa;
unsigned long
        XOR_src_start=0, XOR_src_len=0x2000;
char
        path[200], drive[3], dir[80], name[9], ext[4];

const
    Fill_Byte = 'FF',
    size_ofs = 2,
    head_ofs = 0x08,
    reloc_tbl_item_ofs = 6,
    ss_ofs = 0x0e,
    sp_ofs = 0x10,
    cs_ofs = 0x16,
    ip_ofs = 0x14,
    reloc_tbl_ofs = 0x18,

    ss_1_ofs = 8,
    sp_1_ofs = 10,
    cs_1_ofs = 12,
    ip_1_ofs = 14,
    reloc_items_1_ofs = 16,
    reloc_tbl_1_ofs = 0,
    append_head_len = 26;

main(int argc, char **argv)
{
    fnsplit(argv[0], drive,dir,name,ext);
    printf("\n\nAdd password for EXE file ");
    printf("\n\nSeeded EXE File(EXEPASSb.EXE): ");
    #ifdef DEBUG
        scanf("%s", fn1);
    #else
        printf("\n");
        fnmerge(fn1, drive, dir, "ExePassB", ".exe");
    #endif
```

```
printf("Source EXE File: ");
scanf("%s", fn3);

#ifdef DEBUG
    printf("Target EXE File: ");
    scanf("%s", fn2);
#else
    printf("\n");
    strcpy(fn2, tmpname);
#endif
if( ((fp1=fopen(fn1,"rb")) == NULL) ||
    ((fp3=fopen(fn3,"rb")) == NULL) ||
    ((fp2=fopen(fn2,"wb")) == NULL) ) {
        printf("\n\nopen file failure !");
        exit(1);
    }

#ifdef DEBUG
    strcpy(Old_Password, getpass("PASSWORD: "));
    { char loop, x;
#else
    { char pass[40], loop, x;
        while(1) {
            do {
                GetPassword( Old_Password );
            } while( Old_Password[0] == '\0' );
            do {
                GetPassword( pass );
            }while( Old_Password[0] == '\0' );
            if( strcmp(Old_Password, pass) ) {
                printf("\nIncorrect Password !");
                continue;
            }
            break;
        }
#endif
for(x=0, loop=0; Old_Password[loop] ! = '\0' ; loop++)
    x += Old_Password[loop];
    XOR_v = x;
}
```

```
buf_size = (coreleft() - 512) / 2;
buf2 = malloc( buf_size );
buf3 = malloc( buf_size );
setvbuf(fp3, buf3, _IOFBF, buf_size);
setvbuf(fp2, buf2, _IOFBF, buf_size);

{ unsigned int hi, lo;

    fseek(fp3, 0, SEEK_END);
    dos_fsize3 = ftell( fp3 );
    rewind(fp3);

    fseek(fp3, size_ofs, SEEK_SET);
    lo = getw(fp3);
    hi = getw(fp3);
    if(lo == 0)
        head_fsize3 = (unsigned long) hi * 512;
    else
        head_fsize3 = (unsigned long) (hi-1) * 512 + lo;
    if(head_fsize3 != dos_fsize3) {
        printf("\n\nsource EXE file has appended codes %lu bytes", \
               dos_fsize3-head_fsize3);
    }

    fseek(fp3, ss_ofs, SEEK_SET);
    ss3 = getw(fp3);

    fseek(fp3, sp_ofs, SEEK_SET);
    sp3 = getw(fp3);

    fseek(fp3, cs_ofs, SEEK_SET);
    cs3 = getw(fp3);

    fseek(fp3, ip_ofs, SEEK_SET);
    ip3 = getw(fp3);

    fseek(fp3, reloc_tbl_item_ofs, SEEK_SET);
    reloc_tbl_items3 = getw(fp3);
```

```
        fseek(fp3, reloc_tbl_ofs, SEEK_SET);
        reloc_tbl_addr3 = getw(fp3);

        fseek(fp3, head_ofs, SEEK_SET);
        head_len3 = getw(fp3) << 4;
        image_len3 = head_fsize3 - head_len3;
    }

    fseek(fp1, head_ofs, SEEK_SET);
    head_len1 = getw( fp1 ) << 4;
    rewind( fp1 );
    fseek(fp1, 0, SEEK_END);
    fsize1 = ftell( fp1 );
    image_len1 = fsize1 - head_len1;
    rewind( fp1 );

    { unsigned long x;

        x = append_head_len + reloc_tbl_items3 * 4;
        if( ( (x / 16l) * 16l ) ! = x )
            ins_len1 = (x/16l + 1l) * 16l;
        else ins_len1 = x;
        fill_len1 = ins_len1 - x;

        x = image_len1;
        if( ( (x / 16l) * 16l ) ! = x )
            ins_len = (x/16l + 1l) * 16l;
        else ins_len = x;
        fill_len2 = ins_len - x;

        ins_len += ins_len1;
        ins_seg = ins_len >> 4;
    }

    printf("\ncopying seeded_head ( %u ) ...", head_len1);
    rewind(fp1);
    rewind(fp2);
    for(loop=0; loop<head_len1; loop++)  putc(getc(fp1), fp2);

    { unsigned long size;
```

```
        printf("\nappending src_head ( %u ) ...", \
                append_head_len + reloc_tbl_items3 * 4);

        fseek(fp3, reloc_tbl_addr3, SEEK_SET);
        for(loop=0; loop<reloc_tbl_items3; loop++)  {
            putw(getw(fp3), fp2);
            putw(getw(fp3), fp2);
        }

        printf("\nappending head_fill_codes ( %u ) ...", fill_len1);
        for(loop=0; loop<fill_len1; loop++) putc(Fill_Byte, fp2);

        if(XOR_src_start > image_len3) XOR_src_start = image_len3;
        if((XOR_src_start + XOR_src_len * 2) > image_len3)
            XOR_src_len = (image_len3 - XOR_src_start) / 2;
        putw((unsigned)XOR_src_start, fp2);
        putw((unsigned) (XOR_src_start >> 16), fp2);
        putw((unsigned)XOR_src_len, fp2);
        putw((unsigned) (XOR_src_len >> 16), fp2);

        putw(ss3, fp2);
        putw(sp3, fp2);
        putw(cs3, fp2);
        putw(ip3, fp2);
        putw(reloc_tbl_items3, fp2);

        putw((unsigned)image_len1, fp2);
        putw((unsigned) (image_len1 >> 16), fp2);
        putw((unsigned)image_len3, fp2);
        putw((unsigned) (image_len3 >> 16), fp2);
    }

    printf("\nappending seeded_image_codes ( %lu ) ...", image_len1);
    fseek(fp1, head_len1, SEEK_SET);
    for(loop=0; loop<image_len1; loop++) putc(getc(fp1), fp2);

    printf("\nappending image_fill_codes_2 ( %lu ) ...", fill_len2);
    for(loop=0; loop<fill_len2; loop++) putc(Fill_Byte, fp2);

    {
        printf("\nappending src_image_codes ( %lu ) ...", image_len3);
        fseek(fp3, head_len3, SEEK_SET);
        for(loop=0; loop<image_len3; loop++) putc(getc(fp3), fp2);

        /*
        for(loop=0; loop<XOR_src_start; loop++) putc(getc(fp3), fp2);

        for(loop=XOR_src_start; loop<XOR_src_start+XOR_src_len; loop++) {
          v = getc(fp3);
            v = v ^ XOR_v;
            putc(v, fp2);
        }

        for(loop=XOR_src_start+XOR_src_len; loop<image_len3; loop++)
            putc(getc(fp3), fp2);
            */
    }

    { unsigned long x;
        x = dos_fsize3 - head_fsize3;

        printf("\nappending src_appended_codes ( %lu ) ...", x);
        fseek(fp3, head_fsize3, SEEK_SET);
        for(loop=0; loop<x; loop++) putc(getc(fp3), fp2);
    }

    fclose(fp1);
    fclose(fp2);
    fclose(fp3);
    free(buf3);
    free(buf2);

    buf_size += buf_size;
    buf2 = malloc(buf_size);
    fp2 = fopen(fn2, "rb+");
    setvbuf(fp2, buf2, _IOFBF, buf_size);

    /* modify the part RET-EXEr.EXE */
    ins_seg = ins_len1 >> 4;
```

```
/* a. modify size_double_word */
{ unsigned int hi ,lo;
    unsigned long d, size;
    printf("\nmodifying size_double_word ...");

    size = head_len1 + ins_len + image_len3;
    d = size / 512l;
    hi = ((d * 512) == size) ? d : d+1;
    lo = ((d * 512) == size) ? 0 : size - d * 512;
    fseek(fp2, size_ofs, SEEK_SET);
    putw(lo, fp2);
    putw(hi, fp2);
}

/* b. modify ss */
{ unsigned int ss;
    printf("\nmodifing ss ...");
    fseek(fp2, ss_ofs, SEEK_SET);
    ss = getw( fp2 );
    ss += ins_seg;
    fseek(fp2, -2, SEEK_CUR);
    putw(ss, fp2);
}

/* c. modify cs */
{ unsigned int cs;
    printf("\nmodifing cs ...");
    fseek(fp2, cs_ofs, SEEK_SET);
    cs = getw( fp2 );
    cs += ins_seg;
    fseek(fp2, -2, SEEK_CUR);
    putw(cs, fp2);
}

/* d. modify relocate_tabel_items */
{ unsigned int seg_value, ofs, reloc_tbls, reloc_tbl_addr, loop;
        fseek(fp2, reloc_tbl_item_ofs, SEEK_SET);
        reloc_tbls = getw( fp2 );
        printf("\nmodifing relocate_table_items ( %u ) ...", reloc_tbls);
        fseek(fp2, reloc_tbl_ofs, SEEK_SET);
        reloc_tbl_addr = getw( fp2 );
        for(loop=0; loop<reloc_tbls; loop++ ) {
            ofs = reloc_tbl_addr + loop * 4 + 2;
            fseek(fp2, ofs, SEEK_SET);
            seg_value = getw( fp2 );
            seg_value += ins_seg;
            fseek(fp2, -2, SEEK_CUR);
            putw(seg_value, fp2);
        }
}

/* e. modify relocate tabel */
{ unsigned int hi, lo, value, reloc_tbls, reloc_tbl_addr, head_size, loop;
    unsigned long ofs;
    fseek(fp2, head_ofs, SEEK_SET);
    head_size = getw( fp2 ) << 4;
    fseek(fp2, reloc_tbl_item_ofs, SEEK_SET);
    reloc_tbls = getw( fp2 );
    printf("\nmodifing relocate tables ( %u.) ...", reloc_tbls);
    fseek(fp2, reloc_tbl_ofs, SEEK_SET);
    reloc_tbl_addr = getw( fp2 );
    for(loop=0; loop<reloc_tbls; loop++) {
            fseek(fp2, reloc_tbl_addr+loop * 4, SEEK_SET);
            lo = getw( fp2 );
            hi = getw( fp2 );
            ofs = hi;
            ofs = ofs << 4;
            ofs += lo;
            ofs += head_size;
            fseek(fp2, ofs, SEEK_SET);
            value = getw( fp2 );
            value += ins_seg;
            fseek(fp2, -2, SEEK_CUR);
            putw(value, fp2);
    }
}

/* modify the part SRC.EXE */
ins_seg = ins_len >> 4;
```

13A8-70

```
/* g. modify ss */
{ unsigned int ss;
     unsigned long ofs;
     printf("\nmodifing sub_ss ...");
     ofs = head_len1 + ss_1_ofs + fill_len1 + reloc_tbl_items3 * 4;
     fseek(fp2, ofs, SEEK_SET);
     ss = getw( fp2 );
     ss += ins_seg;
     fseek(fp2, -2, SEEK_CUR);
     putw(ss, fp2);
}

/* h. modify cs */
{ unsigned int cs;
     unsigned long ofs;
     printf("\nmodifing sub_cs ...");
     ofs = head_len1 + cs_1_ofs + fill_len1 + reloc_tbl_items3 * 4;
     fseek(fp2, ofs, SEEK_SET);
     cs = getw( fp2 );
     cs += ins_seg;
     fseek(fp2, -2, SEEK_CUR);
     putw(cs, fp2);
}

/* i. modify relocate_tabel_items */
{ unsigned int seg_value, ofs, reloc_tbls, reloc_tbl_addr, loop;
     int row, col;

     ofs = head_len1 + reloc_items_1_ofs + fill_len1 + reloc_tbl_items3 * 4;
     fseek(fp2, ofs, SEEK_SET);
     reloc_tbls = getw( fp2 );
     printf("\nmodifing sub_relocate_table_items ( %u ) ...", reloc_tbls);
     row = wherex(); col = wherey();
     reloc_tbl_addr = head_len1;

     for(loop=0; loop<reloc_tbls; loop++) {
             gotoxy(row, col);
             printf(" [ %05u ]", loop+1);
             ofs = reloc_tbl_addr + loop * 4 + 2;
             fseek(fp2, ofs, SEEK_SET);
```

```
                seg _ value = getw( fp2 );
                seg _ value += ins _ seg;
                fseek(fp2, ofs, SEEK _ SET);
                putw(seg _ value, fp2);
        }
    }

    /* j. modify relocate tabel */
    { unsigned int hi, lo, value, reloc _ tbls, reloc _ tbl _ addr, head _ size, loop;
        unsigned long ofs;
        int row, col;

        fseek(fp2, head _ ofs, SEEK _ SET);
        head _ size = getw( fp2 ) << 4;
        ofs = head _ len1 + reloc _ items _ 1 _ ofs + fill _ len1 + reloc _ tbl _ items3 * 4;
        fseek(fp2, ofs, SEEK _ SET);
        reloc _ tbls = getw( fp2 );
        printf("\nmodifing sub _ relocate _ tables ( %u ) ...", reloc _ tbls);
        row = wherex(); col = wherey();
        reloc _ tbl _ addr = head _ len1;

        for(loop=0; loop<reloc _ tbls; loop++) {
                gotoxy(row, col);
                printf(" [ %05u ]", loop+1);
                fseek(fp2, reloc _ tbl _ addr+loop * 4, SEEK _ SET);
                lo = getw( fp2 );
                hi = getw( fp2 );
                ofs = hi;
                ofs = ofs << 4;
                ofs += lo;
                ofs += head _ size;
                fseek(fp2, ofs, SEEK _ SET);
                value = getw( fp2 );
                value += ins _ seg;
                fseek(fp2, ofs, SEEK _ SET);
                putw(value, fp2);
        }
    }

    fclose(fp2);
```

```
fp2 = fopen(fn2, "rb+");
setvbuf(fp2, buf2, _IOFBF, buf_size);

/* XOR source image code */
{ unsigned v;
    unsigned long start, loop;

    printf("\nXOR source image_codes %lu words", XOR_src_len);
    start = head_len1 + ins_len;
    fseek(fp2, start, SEEK_SET);
    for(loop=0; loop<XOR_src_len; loop++) {
            fseek(fp2, 0, SEEK_CUR);
            v = getw(fp2);
            v = v ^ XOR_v;
            fseek(fp2, -2, SEEK_CUR);
            putw(v, fp2);
    }
}

{ char loop, v;
    unsigned long ofs;
    ofs = head_len1 + ins_len1 + 0x142;
      /* 0x142 is the offset of Old_Password in COMPASS.EXE */
    printf("\nWrite Password ... ");
    fseek(fp2, ofs, SEEK_SET);
    for(loop=0; Old_Password[loop] ! = '\0' ; loop++) {
            fseek(fp2, 0, SEEK_CUR);
            v = getc( fp2 );
            fseek(fp2, -1, SEEK_CUR);
            v = XOR_PASS ^ Old_Password[loop];
            putc(v , fp2);
    }
}
putc('\0' , fp2);
printf("%s", "\nOk !");
free( buf2 );
fclose( fp2 );

{ int attrib;
    printf("\nErase Source File %s", fn3);
```

```
        attrib = _chmod(fn3, 0);
        _chmod(fn3, 1, FA_ARCH);
        unlink( fn3 );
        rename(tmpname, fn3);
        _chmod(fn3, 1, attrib);
    }
}

/*************************************/
/* EXEPASSb.C                        */
/* 给 EXE 文件加口令的种子程序            */
/* 扩充 EXE 文件的种子程序 2,种子程序将插入到源程序的前面 */
/* 主程序为 PASSEXEb.C,初始模块是 C0S-SEED.OBJ */
/*************************************/

#include <dos.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <ctype.h>

extern int _Init_CS;

unsigned char
        Old_Password[40] = "JTLEE5678901234567890",XOR_PASS = 0xaa;
static unsigned
        ss_h, sp_h, cs_h, ip_h, reloc_items_h,
        tbl_seg_h, tbl_ofs_h,
        fill_seed_len_h, fill_src_len_h,
        XOR_seg_h, XOR_ofs_h,
        psp, inc_seg,
        XOR_v=0, fpsp;
static unsigned long
        seed_len_h, source_len_h;

static unsigned long
        image_le

static unsigned
        XOR_src_v = 0x55aa;
```

```
static unsigned long
        start_CS;
static unsigned long
        XOR_src_start_h, XOR_src_len_h;

const
    size_ofs = 2,
    head_ofs = 0x08,
    XOR_src_start_ofs = 26,
    XOR_src_len_ofs = 22,
    ss_src_ofs = 18,
    sp_src_ofs = 16,
    cs_src_ofs = 14,
    ip_src_ofs = 12,
    reloc_items_src_ofs = 10,
    seed_len_src_ofs = 8,
    source_len_src_ofs = 4,
    fpsp_ofs = 0x16,
    append_head = 26;

far ReProduceInst();
far GetParam();
far XorSourceImage();
far RelocImage();

main()
{
        #ifndef TEST
        static unsigneu seg, ofs;
        #endif

        _DX = (unsigned) "\r\nHello ! \r\n
        ";
        _AH = 0x09;
        __int__(0x21);

        _AH = 0x62;
        __int__(0x21);
        psp = _BX;
        fpsp = peek(psp, fpsp_ofs);
```

```
XOR_v += (peek(psp, fpsp_ofs) - peek(fpsp, fpsp_ofs));

/* call ReProduceInst() */
{
    asm push cs
    asm push cs:XX00
    seg = FP_SEG( ReProduceInst );
    ofs = FP_OFF( ReProduceInst );
    seg += (ofs >> 4);
    ofs = ofs - ( (ofs >> 4) << 4 );
    asm push seg
    asm push ofs
    asm retf
    asm XX00 dw
  +2
}

/* call GetParam() */
{
    asm push cs
    asm push cs:XX01
    seg = FP_SEG( GetParam );
    ofs = FP_OFF( GetParam );
    seg += (ofs >> 4);
    ofs = ofs - ( (ofs >> 4) << 4 );
    asm push seg
    asm push ofs
    asm retf
    asm XX01 dw
  +2
}


{ static char pass[40], x, loop;
        do {
                GetPassword( pass );
                if( pass[0] =='\0' ) PRINT("\r\npassword can not be empty ! \
                                r\n");
                }while( pass[0] == '\0' );
        for(x = 0, loop=0; pass[loop] != '\0' ; loop++)
```

```
                x += pass[loop];

            XOR_src_v = x;

            _AH = 0x62;
            __int__(0x21);
            psp = _BX;
            fpsp = peek(psp, fpsp_ofs);
            XOR_src_v += (peek(psp, fpsp_ofs) - peek(fpsp, fpsp_ofs));

            CompXorPassword(Old_Password, pass, XOR_PASS);
    }

    /* call XorSourceImage() */
    {
        asm push cs
        asm push cs:XORimage_label
        seg = FP_SEG( XorSourceImage );
        ofs = FP_OFF( XorSourceImage);
        seg += (ofs >> 4);
        ofs = ofs - ( (ofs >> 4) << 4 );
        asm push seg
        asm push ofs
        asm retf
        asm XORimage_label dw
        +2
    }

    /* call RelocImage() */
    {
        asm push cs
        asm push cs:XX02
        seg = FP_SEG( RelocImage );
        ofs = FP_OFF( RelocImage );
        seg += (ofs >> 4);
        ofs = ofs - ( (ofs >> 4) << 4 );
        asm push seg
        asm push ofs
        asm retf
        asm XX02 dw
        +2
    }

    /* goto EXE */
    {
        asm mov  ax, ss_h
        asm add  ax, inc_seg
        asm mov  ss, ax
        asm mov  sp, sp_h

        asm mov  ax, cs_h
        asm add  ax, inc_seg
        asm push ax
        asm mov  ax, ip_h
        asm push ax

        asm mov  ax, psp
        asm mov  ds, ax
        asm mov  es, ax

        asm mov  ax, 0
        asm retf
    }

}

/*************************************/
Start_Func()
{
}
/*************************************/
#include "password.c"

SetXorV()
{
        poke(0, 0x03*4 + 2, (XOR_src_v << 1));
        poke(0, 0x03*4 + 0, XOR_src_v);
}

GetXorV()
```

```
{
        XOR_src_v = peek(0, 0x03 * 4 + 0);
}

far GetParam()
{ unsigned ofs, seg, x,
    unsigned long addr;

    start_CS = ((unsigned long)_Init_CS) << 4;
    ss_h = *((int far *)(start_CS - ss_h_ofs));
    sp_h = *((int far *)(start_CS - sp_h_ofs));
    cs_h = *((int far *)(start_CS - cs_h_ofs));
    ip_h = *((int far *)(start_CS - ip_h_ofs));
    reloc_items_h = *((int far *)(start_CS - reloc_items_h_ofs));

    addr = start_CS - XOR_src_start_ofs;
    seg = (addr >> 4);
    ofs =(unsigned) addr & 0x0f;
    XOR_src_start_h = peek(seg, ofs + 2);
    XOR_src_start_h = (unsigned long) (XOR_src_start_h << 16);
    XOR_src_start_h += (unsigned long) peek(seg, ofs);

    addr = start_CS - XOR_src_len_ofs;
    seg = (addr >> 4);
    ofs =(unsigned) addr & 0x0f;
    XOR_src_len_h = peek(seg, ofs + 2);
    XOR_src_len_h = (unsigned long) (XOR_src_len_h << 16);
    XOR_src_len_h += (unsigned long) peek(seg, ofs);


    addr = start_CS - seed_len_src_ofs;
    seg = (addr >> 4);
    ofs =(unsigned) addr & 0x0f;
    seed_len_h = peek(seg, ofs + 2);
    seed_len_h = (unsigned long) (seed_len_h << 16);
    seed_len_h += (unsigned long) peek(seg, ofs);

    addr = start_CS - source_len_src_ofs;
    seg = (addr >> 4);
    ofs =(unsigned) addr & 0x0f;
    source_len_h = peek(seg, ofs + 2);
    source_len_h = (unsigned long) (source_len_h << 16);
    source_len_h += (unsigned long) peek(seg, ofs);

    x = append_head + (reloc_items_h  << 2);
    fill_seed_len_h = ( (x >> 4) << 4 ) == x ? \
                                  0 : ( ((x >> 4)+1 ) << 4 )- x;

    addr = seed_len_h;
    fill_src_len_h = ( (addr >> 4) << 4 ) == addr ? \
                                  0 : ( ((addr >> 4) + 1) << 4 ) - addr;

    seg = append_head + fill_seed_len_h + (reloc_items_h << 2);
    inc_seg = _Init_CS - (seg >> 4);

    addr = append_head + fill_seed_len_h + (reloc_items_h << 2);
    addr =  start_CS - addr;
    tbl_seg_h = (unsigned) (addr >> 4);
    tbl_ofs_h = (unsigned) addr & 0x0f;

    addr = start_CS;
    addr += (fill_src_len_h + seed_len_h);
    addr += XOR_src_start_h;
    XOR_seg_h = (unsigned) (addr >> 4);
    XOR_ofs_h = (unsigned) addr & 0x0f;
}

/* XOR source image_codes */
far XorSourceImage()
{ unsigned seg, ofs, v;
    unsigned long addr, loop;

    addr = (unsigned long)(XOR_seg_h);
    addr = addr << 4;
    addr += XOR_ofs_h;

    _AH = 0x62;
    __int__(0x21);
    psp = _BX;
    fpsp = peek(psp, fpsp_ofs);
```

```
        XOR_src_v += (peek(psp, fpsp_ofs) - peek(fpsp, fpsp_ofs));

        for(loop=0; loop<XOR_src_len_h; loop++) {
            seg = (unsigned)(addr >> 4);
            ofs = (unsigned) addr & 0x0f;
            v = peek(seg, ofs);
            v = v ^ XOR_src_v;
            poke(seg, ofs, v);
            addr += 21;
          }
}

/* relocates the orignal image_code */
far RelocImage()
{ unsigned int seg, ofs, v, loop;
      for(loop=0; loop<reloc_items_h; loop++) {

                ofs = peek(tbl_seg_h, tbl_ofs_h + (loop << 2));
                seg = peek(tbl_seg_h, tbl_ofs_h + (loop << 2)+ 2);
                seg += inc_seg;

                v = peek(seg, ofs);
                v += inc_seg;
                poke(seg, ofs ,v);
      }
}

/*************************************/
End_Func()
{
}
/*******************************   ******/

/* reproduce instructions */
far ReProduceInst()
{ unsigned seg_start, ofs_start, seg_end, ofs_end, seg, ofs,
                loop, value, x, len;

      seg_start = FP_SEG( Start_Func );
      ofs_start = FP_OFF( Start_Func );
```

```
    seg_end  = FP_SEG( End_Func );
    ofs_end  = FP_OFF( End_Func );
    len = ( (seg_end << 4) + ofs_end ) -
          ( (seg_start << 4) + ofs_start);

    seg_start += (ofs_start >> 4);
    ofs_start -= ( (ofs_start >> 4) << 4 );
    for(loop=0; loop<len; loop++){
            x = loop >> 4;
            seg = seg_start + x;
            ofs = ofs_start + loop - (x << 4);
            value = peekb(seg, ofs);
            value = value ^ XOR_v;
            pokeb(seg, ofs, value);
    }
}

/* PASSWORD.C */

#ifdef DEBUG
    #include <dos.h>
    #include <stdio.h>
    #include <ctype.h>
#endif

#ifdef DEBUG
char
        * Password = "ABCD", XOR_Pass = 0;
unsigned  /* 16 bits , 4 bytes */
        Limit = 5;
main()
{
    char pass[100];
    GetPassword( pass );
    if( CompXorPassword(Password, pass, XOR_Pass) ) PRINT("\r\nOk ! \r\n
 ");
    else PRINT("\r\nIncorrect Password ! \r\n
 ");
}
```

```
#endif

PRINT(char *s)
{
      _DX = (unsigned) s;
      _AH = 0x09;
      __int__(0x21);
}

GETCH()
{
      _AH = 0x07;
      __int__(0x21);
      return _AL;
}

PUTCH(char cc)
{
      _DL = cc;
      _AH = 2;
      __int__(0x21);
}

GetPassword(char *s)
{ int loop;
      char value=0;

      #ifndef DEBUG
            { int psp, fpsp, fpsp_ofs = 0x16;
                  _AH = 0x62;
                  __int__(0x21);
                  psp = _BX;
                  fpsp = peek(psp, fpsp_ofs);
                  value += (peek(psp, fpsp_ofs) - peek(fpsp, fpsp_ofs));
            }
      #endif
            PRINT("\r\nPASSWORD:
      ");
            for(loop=0; ; loop++) {
                  s[loop] = '\0';
            while( kbhit() ) s[loop] = GETCH();
            s[loop] = GETCH();
            s[loop] = s[loop] ^ value;
            if(s[loop] == 0x0d) { /* ENTER */
                s[loop] = '\0';
                return;
            }
            if(s[loop] == 0x1b) { /* ESC */
                s[0] = '\0';
                return;
            }
        /*      if( ! isprint(s[loop]) ) {
                --loop;
                continue;
            }
        */
        #ifdef DEBUG
            PUTCH('*');
        #endif
            }
    }

    XorPassword(char *s, char XOR_v)
    { char value = XOR_v;
        int loop;
    #ifndef DEBUG
        { int psp, fpsp, fpsp_ofs = 0x16;
            _AH = 0x62;
            __int__(0x21);
            psp = _BX;
            fpsp = peek(psp, fpsp_ofs);
            value += (peek(psp, fpsp_ofs) - peek(fpsp, fpsp_ofs));
        }
    #endif
        for(loop=0; s[loop] != '\0'; loop++)
            s[loop] = s[loop] ^ value;
    }

    /* compare Old_Pass with (New_Pass xor XOR_v), return
        1, if equal
        0, if not equal
        Old _ Pass, New _ Pass not changed * /
    CompXorPassword(char * Old _ Pass, char * New _ Pass, char XOR _ v)
    { char value = XOR _ v;
        int loop;
    #ifndef DEBUG
        { int psp, fpsp, fpsp _ ofs = 0x16;
            _AH = 0x62;
            __int__(0x21);
            psp = _BX;
            fpsp = peek(psp, fpsp _ ofs);
            value += (peek(psp, fpsp _ ofs) - peek(fpsp, fpsp _ ofs));
        }
#endif
    for(loop=0; Old _ Pass[loop] ! = '\0' ; loop++)
        if( (New _ Pass[loop] ^ value) ! = Old _ Pass[loop] )
#ifdef DEBUG
    return 0;
#else
    { loop = 0;
        continue;
    }
#endif

    return 1;
}
```

[General Information]
书名=DOS DEBUG3.31 详解
作者=
页数=1000
SS号=
DX号=
出版日期=
出版社=

书名
目录
目录

## 附录E.　　给EXE文件加口令的源程序