DOS DEBUG3.31详解

李启龙　编写

北京希望电脑公司

DOS DEBUG 3.31 详解

李启龙　编写

北京希望电脑公司
1992.8

内 容 简 介

本书讲述了DOS基础，介绍了DEBUG程序的命令行、所有命令的用法，讨论了可执行文件的扩展和加密，并提供具体的程序。在最后给出了DEBUG程序被反汇编后的汇编语言程序清单。

如果要对某一个可执行程序进行解密和加密，则不仅要了解.COM和.EXE文件的结构，而且要了解DEBUG的工作原理。本书全面剖析了DOS的DEBUG程序，是加密和解密所必需了解的知识。

有购本书者可与北京8721信箱联系，电话2562329。

DOS DEBUG 3.31 详解

李启龙 编写

北京希望电脑公司出版发行

1992年9月第一版 1992年9月第一次发行

准印证号：3565－91565

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

寄存器名	
sector sector 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 字符串	由单引号或双引号括住的一串 ASCII 字符
value 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：