DEBUG 命令详解

DEBUG 是 DOS 中的一个外部命令，从 DOS 1.0 起就带有此命令，因此可见此命令的重要性了。虽然此命令的功能非常强大，可以解决许多问题，可是对许多人来说，尤其是初学者来说，却非常不易掌握。因此，现将 DEBUG 的命令详细介绍一番，以让大家知道它的使用。
Debug:A（汇编）

直接将 8086/8087/8088 记忆码合并到内存。

该命令从汇编语言语句创建可执行的机器码。所有数值都是十六进制格式，必须按一到四个字符输入这些数值。在引用的操作代码（操作码）前指定前缀记忆码。

a [address]

参数

address

指定键入汇编语言指令的位置。对 address 使用十六进制值，并键入不以“h”字符结尾的每个值。如果不指定地址，a 将在它上次停止处开始汇编。

有关将数据输入到指定字节中的信息，请单击“相关主题”列表中的 Debug E（键入）。

有关反汇编字节的信息，请单击“相关主题”列表中的 Debug U（反汇编）。

说明

使用记忆码

段的替代记忆码为 cs:、ds:、es: 和 ss:。远程返回的记忆码是 retf。字符串处理的记忆码必须明确声明字符串大小。例如，使用 movsw 可以移动 16 位的字串，使用 movsb 可以移动 8 位字节串。

汇编跳转和调用

汇编程序根据字节替换自动将短、近和远的跳转及调用汇编到目标地址。通过使用 near 或 far 前缀可以替代这样的跳转或调用，如下例所示：

-a0100:0500

```
0100:0500 jmp 502 ; a 2-byte short jump
0100:0502 jmp near 505 ; a 3-byte near jump
0100:0505 jmp far 50a ; a 5-byte far jump
```

可以将 near 前缀缩写为 ne。

区分字和字节内存位置

当某个操作数可以引用某个字内存位置或者字节内存位置时，必须用前缀 word ptr 或者前缀 byte ptr 指定数据类型。可接受的缩写分别是 wo 和 by。以下范例显示两种格式：

```
dec wo [si]
neg byte ptr [128]
```

指定操作数

Debug 使用包括在中括号 ([ ]) 的操作数引用内存地址的习惯用法。这是因为另一方面 Debug 不能区分立即操作数和内存地址的操作数。以下范例显示两种格式：

```
mov ax,21 ; load AX with 21h
mov ax,[21] ; load AX with the
; contents of
; memory location 21h
```

使用伪指令

使用 a 命令提供两个常用的伪指令：db 操作码，将字节值直接汇编到内存，dw 操作码，将字值直接汇编到内存。以下是两个伪指令的范例：

```
db 1,2,3,4,"THIS IS AN EXAMPLE"
db 'THIS IS A QUOTATION MARK:"'
db "THIS IS A QUOTATION MARK:'"
dw 1000,2000,3000,"BACH"
```

范例

a 命令支持所有形式的间接注册命令，如下例所示：

```
add bx,34[bp+2].[si-1]
pop [bp+di]
push [si] )
```

还支持所有操作码同义词，如下例所示：

```
loopz 100
loope 100
ja 200
jnbe 200
```

对于 8087 操作码，必须指定 wait 或 fwait 前缀，如下例所示：

```
fwait fadd st,st(3) ; this line assembles
; an fwait prefix
```

Debug:C（比较）

比较内存的两个部分。

c range address

参数

range

指定要比较的内存第一个区域的起始和结束地址，或起始地址和长度。有关有效的 range 值的信息，请单击“相关主题”列表中的“Debug 说明”。

address

指定要比较的第二个内存区域的起始地址。有关有效 address 值的信息，请单击“相关主题”列表中的“Debug 说明”。

说明

如果 range 和 address 内存区域相同，Debug 将不显示任何内容而直接返回到 Debug 提示符。如果有差异，Debug 将按如下格式显示：
address1 byte1 byte2 addess2

范例

以下命令具有相同效果：

```
c100,10f 300
```

c100110 300

每个命令都对 100h 到 10Fh 的内存数据块与 300h 到 30Fh 的内存数据块进行比较。

Debug 响应前面的命令并显示如下信息（假定 DS = 197F）：

```
197F:0100 4D E4 197F:0300
197F:0101 67 99 197F:0301
197F:0102 A3 27 197F:0302
197F:0103 35 F3 197F:0303
197F:0104 97 BD 197F:0304
197F:0105 04 35 197F:0305
197F:0107 76 71 197F:0307
197F:0108 E6 11 197F:0308
197F:0109 19 2C 197F:0309
197F:010A 80 0A 197F:030A
197F:010B 36 7F 197F:030B
197F:010C BE 22 197F:030C
197F:010D 83 93 197F:030D
197F:010E 49 77 197F:030E
197F:010F 4F 8A 197F:030F
```

注意列表中缺少地址 197F:0106 和 197F:0306。这表明那些地址中的值是相同的。

Debug:D（转储）

显示一定范围内存地址的内容。

d [range]

参数

range

指定要显示其内容的内存区域的起始和结束地址，或起始地址和长度。有关有效的 range 值的信息，请单击“相关主题”列表中的“Debug 说明”。如果不指定 range，Debug 程序将从以前 d 命令中所指定的地址范围的末尾开始显示 128 个字节的内容。

有关显示寄存器内容的信息，请单击“相关主题”列表中的 Debug R（寄存器）。

说明

当使用 d 命令时，Debug 以两个部分显示内存内容：十六进制部分（每个字节的值都用十六进制格式表示）和 ASCII 码部分（每个字节的值都用 ASCII 码字符表示）。每个非打印字符在显示的 ASCII 部分由句号 (.) 表示。每个显示行显示 16 字节的内容，第 8 字节和第 9 字节之间有一个连字符。每个显示行从 16 字节的边界上开始。

范例
假定键入以下命令：
dcs:100 10f
Debug 按以下格式显示范围中的内容：
04BA:0100 54 4F 4D 00 53 41 57 59-45 52 00 00 00 00 00 00 TOM.SAWYER......
如果在没有参数的情况下键入 d 命令，Debug 按以前范例中所描述的内容来编排显示格式。显示的每行以比前一行的地址大 16 个字节（如果是显示 40 列的屏幕，则为 8 个字节）的地址开头。
对于后面键入的每个不带参数的 d 命令，Debug 将紧接在最后显示的命令后立即显示字节内容。
如果键入以下命令，Debug 将从 CS:100 开始显示 20h 个字节的内容：
dcs:100 l 20
如果键入以下命令，Debug 将显示范围从 CS 段的 100h 到 115h 中所有字节的内容：
dcs:100 115

Debug:E（键入）

将数据输入到内存中指定的地址。
可以按十六进制或 ASCII 格式键入数据。以前存储在指定位置的任何数据全部丢失。

e address [list]

参数
address
指定输入数据的第一个内存位置。
list
指定要输入到内存的连续字节中的数据。
有关集成记忆码的信息，请单击“相关主题”列表中的 Debug A（汇编）。
有关显示内存部分内容的信息，请单击“相关主题”列表中的 Debug D （转储）。

说明

使用 address 参数

如果在没有指定可选的 list 参数的值情况下指定 address 的值，Debug 将显示地址和内容，在下一行重复地址，并等待您的输入。此时，您可以执行下列操作之一：

替换字节值。为此，请在当前值后键入新值。如果您键入的值不是有效的十六进制值，或该值包含两个以上的数字，则 Debug 不会回显无效或额外的字符。
进入下一个字节。为此，请按 SPACEBAR（空格键）。要更改该字节中的值，请在当前值后键入新值。如果按 SPACEBAR（空格键）时，移动超过了 8 位界限，Debug 程序将显示新的一

行并在行首显示新地址。
返回到前一个字节。为此，请按 HYPHEN 键 (-)。可以反复按 HYPHEN 键 (-) 向后移动超过多个字节。在按 HYPHEN 时，Debug 开始新行并显示当前地址和字节值。
停止执行 e 命令。为此，请按 ENTER 键。在任何字节位置都可以按 ENTER。
使用 list 参数

如果指定 list 参数的值，随后的 e 命令将使用列表中的值替换现有的字节值。如果发生错误，将不更改任何字节值。

List 值可以是十六进制字节或字符串。使用空格、逗号或制表符来分隔值。必须将字符串包括在单或双引号中。

范例

假定键入以下命令：

ecs:100

Debug 按下面的格式显示第一个字节的内容：

04BA:0100 EB.

要将该值更改为 41，请在插入点键入 41，如下所示：

04BA:0100 EB.41_

可以用一个 e 命令键入连续的字节值。在键入新值后按 SPACEBAR(空格键),而不是按 ENTER 键。Debug 显示下一个值。在此范例中，如果按三次 SPACEBAR（空格键），Debug 将显示下面的值：

04BA:0100 EB.41 10. 00. BC._

要将十六进制值 BC 更改为 42，请在插入点键入 42，如下所示：

04BA:0100 EB.41 10. 00. BC.42_

假定决定值 10 应该是 6F。要纠正该值，请按 HYPHEN 键两次以返回到地址 0101（值 10）。Debug 显示以下内容：

04BA:0100 EB.41 10. 00. BC.42-
04BA:0102 00.-
04BA:0101 10._

在插入点键入 6f 更改值，如下所示：

04BA:0101 10.6f_

按 ENTER 停止 e 命令并返回到 Debug 提示符下。

以下是字符串项的范例：

eds:100 "This is the text example"

该字符串将从 DS:100 开始填充 24 个字节

Debug:F（填充）

使用指定的值填充指定内存区域中的地址。

可以指定十六进制或 ASCII 格式表示的数据。任何以前存储在指定位置的数据将会丢失。

f range list

参数

range

指定要填充内存区域的起始和结束地址，或起始地址和长度。关于有效的 range 值的信息，请单击“相关主题”列表中的“Debug 说明”。

list

指定要输入的数据。List 可以由十六进制数或引号包括起来的字符串组成。

说明

使用 range 参数

如果 range 包含的字节数比 list 中的数值大，Debug 将在 list 中反复指派值，直到 range 中的所有字节全部填充。

如果在 range 中的任何内存损坏或不存在，Debug 将显示错误消息并停止 f 命令。

使用 list 参数

如果 list 包含的数值多于 range 中的字节数，Debug 将忽略 list 中额外的值。

范例

假定键入以下命令：

f04ba:1001100 42 45 52 54 41

作为响应，Debug 使用指定的值填充从 04BA:100 到 04BA:1FF 的内存位置。Debug 重复这五个值直到 100h 个字节全部填满为止。

Debug:G（转向）

运行当前在内存中的程序。

g [=address] [breakpoints]

参数

=address

指定当前在内存中要开始执行的程序地址。如果不指定 address，Windows 2000 将从 CS:IP 寄存器中的当前地址开始执行程序。

breakpoints

指定可以设置为 g 命令的部分的 1 到 10 个临时断点。

有关执行循环、重复的字符串指令、软件中断或子程序的信息，请单击“相关主题”列表中的 Debug P（执行）。

有关执行指令的信息，请单击“相关主题”列表中的 Debug T（跟踪）。

Debug:H（十六进制）

对指定的两个参数执行十六进制运算。

h value1 value2

参数

value1

代表从 0 到 FFFFh 范围内的任何十六进制数字。

value2

代表从 0 到 FFFFh 范围内第二个十六进制数字。

说明

Debug 首先将指定的两个参数相加，然后从第一个参数中减去第二个参数。这些计算的结果显示在一行中：先计算和，然后计算差。

范例

假定键入以下命令：

h19f 10a

Debug 执行运算并显示以下结果。
02A9 0095

Debug:I（输入）

从指定的端口读取并显示一个字节值。

i port

参数

port

按地址指定输入端口。地址可以是 16 位的值。

有关将字节值发送到输出端口的信息，请单击“相关主题”列表中的 Debug O（输出）。

范例

假定键入以下命令：

i2f8

同时假定端口的字节值是 42h。Debug 读取该字节，并将其值显示如下：

Debug:L（加载）

将某个文件或特定磁盘扇区的内容加载到内存。

要从磁盘文件加载 BX:CX 寄存器中指定的字节数内容，请使用以下语法：

l [address]

要略过 Windows 2000 文件系统并直接加载特定的扇区，请使用以下语法：

l address drive start number

参数

address

指定要在其中加载文件或扇区内容的内存位置。如果不指定 address，Debug 将使用 CS 寄存器中的当前地址。

drive

指定包含读取指定扇区的磁盘的驱动器。该值是数值型：0 = A, 1 = B, 2 = C 等。

start

指定要加载其内容的第一个扇区的十六进制数。

number

指定要加载其内容的连续扇区的十六进制数。只有要加载特定扇区的内容而不是加载 debug 命令行或最近的 Debug n（名称）命令中指定的文件时，才能使用 drive、start 和 number 参数。

有关指定用于 l 命令的文件的信息，请单击“相关主题”列表中的 Debug n（名称）。

有关写入调试到磁盘的文件的信息，请单击“相关主题”列表中的 Debug w（写入）。

注意

使用不带参数的 l 命令

当使用不带参数的 l 命令时，在 debug 命令行上指定的文件将加载到内存中，从地址 CS:100 开始。Debug 同时将 BX 和 CX 寄存器设置为加载的字节数。如果不在 debug 命令行指定文件，所装入的文件将是最近使用 n 命令经常指定的文件。

使用具有 address 参数的 l 命令

如果使用带 address 参数的 l 命令，Debug 将从内存位置 address 开始加载文件或指定扇区的内容。

使用带全部参数的 l 命令

如果使用带所有参数的 l 命令，Debug 将加载指定磁盘扇区的内容而不是加载文件。

加载特定扇区的内容

指定范围内的每个扇区均从 drive 读取。Debug 从 start 开始加载，直到在 number 中指定的扇区数中的内容全部被加载。

加载 .exe 文件

Debug 忽略 .exe 文件的地址 address 参数。如果指定 .exe 文件，Debug 将文件重新定位到 .exe 文件的标题中指定的加载地址。在 .exe 文件被加载到内存前，标题自身从 .exe 文件脱离，因此磁盘上的 .exe 文件大小与内存中的不同。如果要检查整个 .exe 文件，请使用不同的扩展名重命名文件。

打开十六进制文件

Debug 将具有 .hex 扩展名的文件认为十六进制格式文件。键入不带参数的 l 命令，可以加载从十六进制文件中指定的地址处开始的十六进制文件。如果键入的 l 命令包含 address 参数，Debug 将把指定的地址加到在十六进制文件中找到的地址上，以确定起始地址。

范例

假定启动 Debug 并键入以下命令：

nfile.com

现在可以键入 l 命令以加载 File.com。Debug 将加载文件并显示 Debug 提示符。

假定需要从驱动器 C 将起始逻辑扇区为 15 (0Fh) 的 109 (6Dh) 个扇区的内容加载到起始

地址为 04BA:0100 的内存中。为此，请键入以下命令：
l04ba:100 2 0f 6d

Debug:M（移动）

将一个内存块中的内容复制到另一个内存块中。

m range address

参数

range

指定要复制内容的内存区域的起始和结束地址，或起始地址和长度。

address

指定要将 range 内容复制到该位置的起始地址。

说明

复制操作对现有数据的影响

如果新数据没有写入正在被复制的数据块中的地址，则源数据将保持不变。但是，如果目标块已经包含数据(就象它在覆盖副本操作中一样)，则将改写该数据。(覆盖复制操作是指那些目标数据块部分内容覆盖原数据块部分内容的操作。)

执行覆盖复制操作

m 命令执行目标地址的覆盖复制操作，而不丢失数据。将改写的地址内容首先复制。因此，如果将较高位地址的数据复制到较低位地址，则复制操作从原块的最低位地址开始并向最高位地址进行。反之，如果要将数据从低地址复制到高地址，复制操作从原块的最高地址开始，向最低地址进行。

范例

假定键入以下命令：

mcs:100 110 cs:500
Debug 首先将 CS:110 地址中的内容复制到地址 CS:510 中，然后将 CS:10F 地址中的内容复制到 CS:50F 中，如此操作直至将 CS:100 地址中的内容复制到地址 CS:500 中。要查看

结果，请使用 Debug d（转储）命令，并使用 m 命令指定目标地址

Debug:N（名称）

指定 Debug l（加载）或 w（写入）命令的可执行文件的名称，或者指定正在调试的可执行文件的参数。

n [drive:][path] filename

要指定测试的可执行文件的参数，请使用以下语法：

n file-parameters

参数

如果在没有参数的情况下使用，则 n 命令清除当前规范。
[drive:][path] filename

指定要测试的可执行文件的位置和名称。

file-parameters

为正在测试的可执行文件指定参数和开关。

有关将文件或指定磁盘扇区的内容加载到内存中的信息，请单击“相关主题”列表中的 Debug L（加载）。

有关写入调试到磁盘的文件的信息，请单击“相关主题”列表中的 Debug W（写入）。

说明

n 命令的两个用途

可以按两种方式使用 n 命令。首先，您可以使用它以指定后面的 l（加载）或 w（写入）命令所使用的文件。如果在没有命名所调试文件的情况下启动 Debug，必须在使用 l 命令加载文件之前使用命令 nfilename。在 CS:5C 为文件控制块 (FCB) 正确编排文件名的格式。其次，可以使用 n 命令指定被调试文件的命令行参数和开关。

内存区域

以下四个内存区域都会受到 n 命令的影响：

内存位置
内容

CS:5C
文件 1 的文件控制数据块 (FCB)

CS:6C
文件 2 的文件控制数据块 (FCB)

CS:80
n 命令行的长度（以字符表示）

CS:81
n 命令行字符的开头

为 n 命令指定的第一个文件名被放在 CS:5C 的 FCB 中。如果指定第二个文件名，此名称将放置到 CS:6C 的 FCB 中。n 命令行上键入的字符数（除第一个字符之外，n）存储在位置 CS:80。n 命令行上的实际字符（再次，除了字母 n 之外）存储在以 CS:81 开头的位置。注意这些字符可以是在 Windows 2000 命令提示符下键入的命令中有效的任何开关和分隔符。

范例

假定已经启动 Debug，并加载了正在调试的程序 Prog.com。接着您决定为 Prog.com 指定两个参数并运行此程序。以下是此范例的命令序列：

```
debug prog.com
nparam1 param2
g
```

在这种情况下，Debug g（转向）命令会运行该程序，就好像您已在 Windows 2000 命令提示符后键入了如下命令：

```
prog param1 param2
```

所以，测试和调试反映 Prog.com 通常的运行时间环境。

在下面的命令序列中，第一个 n 命令将 File1.exe 指定为后接的 l（加载）命令的文件，该命令将 File1.exe 加载到内存。第二个 n 命令指定 File1.exe 将使用的参数。最后，g 命令将运行 File1.exe 文件，就好像您在 Windows 2000 命令行中键入了 File1 File2.dat File2.dat 一样。

```
nfile1.exe
```

l
nfile2.dat file3.dat
g

注意
不要在 n 命令的第二种形式后使用 l 命令。还要注意,如果现在使用 w(写入)命令,Windows 2000 将使用名称 File2.dat 保存正在调试的文件 File1.exe。为避免出现此结果，应该总是在 l 或 w 命令之前立即使用 n 命令的第一种形式。

Debug:O（输出）

将字节值发送到输出端口。

o port byte-value

参数

port

通过地址指定输出端口。端口地址可以是 16 位值。

byte-value

指定要指向 port 的字节值。

有关从输入端口读取字节值的信息，请单击“相关主题”列表中的 Debug I（输入)。

范例

要将字节值 4Fh 发送到地址为 2F8h 的输出端口，请键入以下命令：
o2f8 4f

Debug:P（执行）

执行循环、重复的字符串指令、软件中断或子例程；或通过任何其他指令跟踪。

p [= address] [number]

参数

=address

指定第一个要执行指令的位置。如果不指定地址，则默认地址是在 CS:IP 寄存器中指定的当前地址。

number

指定在将控制返回给 Debug 之前要执行的指令数。默认值为 1。

有关运行当前在内存中程序的信息，请单击“相关主题”列表中的 Debug G（转向）。

有关执行指令的信息，请单击“相关主题”列表中的 Debug T（跟踪）。

说明
控制传送到要测试的程序

当 p 命令将控制从 Debug 传送到要测试的程序时，该程序不间断运行，直到循环、重复字符串指令、软件中断或者完成了指定地址的子例程为止，或者直到执行了指定数量的机器指令为止。控制返回到 Debug。

地址参数的限制

如果 address 参数没有指定段,Debug 将使用被测试程序的 CS 寄存器。如果省略 address，程序将从 CS:IP 寄存器所指定的地址开始执行。必须在 address 参数之前使用等号 (=) 以便将它与 number 参数区分。如果在指定地址处的指令不是循环、重复的字符串指令、软件中断或子例程，则 p 命令与 Debug t（跟踪）命令的作用相同。

使用 p 命令显示的邮件

当 p 执行完一段说明后，Debug 显示出程序的寄存器内容、标志的状态以及下一段将要被执行的指令的解码形式。

警告

不能使用 p 命令跟踪只读内存 (ROM)。

范例

假定正在测试的程序在地址 CS:143F 处包含一个 call 指令。要运行 call 目标位置的子程序然后将控制返回到 Debug，请键入以下命令：

p=143f

Debug 按以下格式显示结果：

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=2246 ES=2246 SS=2246 CS=2246 IP=1443 NV UP EI PL NZ AC PO NC
2246:1442 7505 JNZ 144A

Debug:Q（退出）

停止 Debug 会话，不保存当前测试的文件。

当您键入 q 以后，控制返回到 Windows 2000 的命令提示符。

q

参数

该命令不带参数。
有关保存文件的信息，请单击“相关主题”列表中的 Debug W（写入）。

Debug:R（寄存器）

显示或改变一个或多个 CPU 寄存器的内容。

r [register-name]

参数

无

如果在没有参数的情况下使用，则 r 命令显示所有寄存器的内容以及寄存器存储区域中的标志。

register-name

指定要显示其内容的寄存器名。

有关显示内存部分内容的信息，请单击“相关主题”列表中的 Debug D（转储）。

有关反汇编字节的信息，请单击“相关主题”列表中的 Debug U（反汇编）。

说明

使用 r 命令

如果指定了寄存器名称，Windows 2000 将显示以十六进制标记表示的寄存器的 16 位值，并将冒号显示为提示符。如果要更改包含在寄存器中的值，除非键入新值并按 ENTER 键；否则，请按 ENTER 键返回 Debug 提示符。

有效寄存器名

以下是 register-name 的有效值：ax、bx、cx、dx、sp、bp、si、di、ds、es、ss、cs、ip、pc 及 f。ip 和 pc 都引用指令指针。

如果指定寄存器名称，而不是从前面的列表中指定，Windows 2000 将显示以下消息：

br error

使用 f 字符而不是寄存器名

如果键入 f 字符代替寄存器名，Debug 将每个标记的当前设置显示为两字母代码，然后显示 Debug 提示符。要更改标志的设置，请从下表中键入适当的两字母代码：

标志名
设置
清除

溢出
ov
nv

方向
dn（减）
up（增）

中断
ei（启用）
di（禁用）

正负
ng（负）
pl（正）

零
zr
nz

辅助进位

ac
na

奇偶校验
pe（偶校验）
po（奇校验）

进位
cy
nc

可以按任何顺序键入新的标志值。不需要在这些值之间留出空格。要停止 r 命令，请按 ENTER 键。任何没有指定新值的标志保持不变。

用 r 命令显示的邮件

如果为标记指定了多个值，Debug 将显示以下消息：

df error

如果指定没有在前面的表中列出的标志代码，Debug 将显示以下消息：

bf error

在这两种情况下，Debug 将忽略所有在无效项目之后指定的设置。

Debug 的默认设置

在启动 Debug 时，会将段寄存器设置到空闲内存的低端，指令指针设置为 0100h，清除所有标志，并且将其余寄存器设置为零，除了被设置为 FFEEh 的 sp 之外。

Debug:R

范例

要查看所有寄存器的内容、所有标记的状态和当前位置的指令解码表，请键入以下命令：

r

如果当前位置是 CS:11A，显示外观将类似于以下内容：

```
AX=0E00 BX=00FF CX=0007 DX=01FF SP=039D BP=0000 SI=005C DI=0000
DS=04BA ES=04BA SS=04BA CS=04BA IP=011A NV UP DI NG NZ AC PE NC
04BA:011A CD21 INT 21
```

要只查看标志的状态，请键入以下命令：

rf

Debug 按以下格式显示信息：

NV UP DI NG NZ AC PE NC - _

现在，您可以按任意顺序键入一个或多个有效的标志值，其中可以有或没有空格，如下所示：

nv up di ng nz ac pe nc - pleicy

Debug 结束 r 命令并显示 Debug 提示符。要查看更改，请键入 r 或 rf 命令。Debug 将显示以下内容：

NV UP EI PL NZ AC PE CY - _
按 ENTER 返回到 Debug 提示符。

Debug:S（搜索）

在某个地址范围搜索一个或多个字节值的模式。

s range list

参数

range

指定要搜索范围的开始和结束地址。有关 range 参数有效值的信息，请单击“相关主题”列表中的 Debug。

list

指定一个或多个字节值的模式，或要搜索的字符串。用空格或逗号分隔每个字节值和下一个字节值。将字符串值包括在引号中。

说明

如果 list 参数包含多个字节值，Debug 将只显示出现字节值的第一个地址。如果 list 只包含一个字节值，Debug 将显示指定范围内出现该值的所有地址。

范例

假定需要查找包含值 41 并且范围从 CS:100 到 CS:110 的所有地址。为此，请键入以下命令：

scs:100 110 41

Debug 按以下格式显示结果：

04BA:0104
04BA:010D
-

以下命令在 CS:100 到 CS:1A0 的范围内搜索字符串“Ph”。
scs:100 1a0 "Ph"

Debug:U（反汇编）

反汇编字节并显示相应的原语句，其中包括地址和字节值。反汇编代码看起来象已汇编文件的列表。

u [range]

参数

无

如果在没有参数的情况下使用，则 u 命令分解 20h 字节（默认值），从前面 u 命令所显示地址后的第一个地址开始。

range

指定要反汇编代码的起始地址和结束地址，或起始地址和长度。有关 range 参数有效值的信息，请单击“相关主题”列表中的 Debug。

有关集成记忆码的信息，请单击“相关主题”列表中的 Debug A（汇编）。

有关显示内存部分内容的信息，请单击“相关主题”列表中的 Debug D（转储）。

范例

要反汇编 16 (10h) 字节，从地址 04BA:0100 开始，请键入以下命令：

u04ba:100l10

Debug 按以下格式显示结果：

```
04BA:0100 206472 AND [SI+72],AH
04BA:0103 69 DB 69
04BA:0104 7665 JBE 016B
04BA:0106 207370 AND [BP+DI+70],DH
04BA:0109 65 DB 65
04BA:010A 63 DB 63
04BA:010B 69 DB 69
04BA:010C 66 DB 66
04BA:010D 69 DB 69
04BA:010E 63 DB 63
04BA:010F 61 DB 61
```

如果只显示从 04BA:0100 到 04BA:0108 特定地址的信息，请键入以下命令：

u04ba:0100 0108

Debug 显示以下内容：

```
04BA:0100 206472 AND [SI+72],AH
04BA:0103 69 DB 69
04BA:0104 7665 JBE 016B
04BA:0106 207370 AND [BP+DI+70],DH
```

Debug:W（写入）

将文件或特定分区写入磁盘。

要将在 BX:CX 寄存器中指定字节数的内容写入磁盘文件，请使用以下语法：

w [address]

要略过 Windows 2000 文件系统并直接写入特定的扇区，请使用以下语法：

w address drive start number

参数

address

指定要写到磁盘文件的文件或部分文件的起始内存地址。如果不指定 address，Debug 程序将从 CS:100 开始。关于 address 参数有效值的信息，请在“相关主题”列表中单击 Debug。

drive

指定包含目标盘的驱动器。该值是数值型：0 = A, 1 = B, 2 = C,等等。

start

指定要写入第一个扇区的十六进制数。

number

指定要写入的扇区数。

有关指定用于 w 命令的文件的信息，请单击“相关主题”列表中的 Debug N（名称）。

有关将文件或文件扇区内容加载到内存中的信息，请单击“相关主题”列表中的 Debug L（加载）。

说明

必须在启动 Debug 时或者在最近的 Debug n（名称）命令中指定磁盘文件的名字。这两种方法都可以将地址 CS:5C 处文件控制块的文件名正确地编排格式。

在使用不带参数的 w 命令之前重新设置 BX:CX

如果使用了 Debug g（转向）、t（跟踪）、p（执行）或 r（寄存器）命令，必须在使用无参数的 w 命令之前，将 BX:CX 寄存器复位。

将修改后的文件写入磁盘

如果修改文件但不更改文件名、长度或起始地址，Debug 仍然可以正确地将文件写入源磁盘位置。

w 命令的限制

不能用该命令写入 .exe 或 .hex 文件。

警告

因为略过 Windows 2000 文件句柄，所以写入特定的分区非常危险。如果键入错误的值，则

磁盘文件结构很容易被损坏。

范例

假定要将起始地址为 CS:100 的内存内容写入到驱动器 B 的磁盘中。需要将数据从磁盘的逻辑扇区号 37h 开始并持续 2Bh 个扇区。为此，键入以下命令：

wcs:100 1 37 2b

当写操作完成时，Debug 再次显示 Debug 提示符。

Debug:XA（分配扩展内存）

分配扩展内存的指定页面数。

要使用扩展内存，必须安装符合 4.0 版的 Lotus/Intel/Microsoft 扩展内存规范 (LIM EMS) 的扩展内存设备驱动程序。

xa [count]

参数

count

指定要分配的扩展内存的 16KB 页数。

有关使用扩展内存的其他 Debug 命令的信息，请单击“相关主题”列表中的 XD（释放扩展内存）、XM（映射扩展内存页）或 XS（显示扩展内存状态）。

说明

如果指定的页面数可用，则 Debug 将显示消息，此消息表明所创建的句柄的十六进制数；否则，Debug 将显示错误消息。

Debug:XA

范例

要分配扩展内存的 8 个页面，请键入以下命令：

xa8

如果命令成功，Debug 将显示类似的以下消息：

Handle created=0003

Debug:XD（释放扩展内存）

释放指向扩展内存的句柄。

要使用扩展内存，必须安装符合 4.0 版的 Lotus/Intel/Microsoft 扩展内存规范 (LIM EMS) 的扩展内存设备驱动程序。

xd [handle]

参数

handle

指定要释放的句柄。

有关使用扩展内存的其他 Debug 命令的信息，请单击“相关主题”列表中 XA（分配扩展内存）、XM（映射扩展内存页） 或 XS（显示扩展内存状态）。

范例

要释放句柄 0003，请键入以下命令：

xd 0003

如果命令成功，Debug 将显示下列消息：
Hdle 0003 deallocated

Debug:XM（映射扩展内存页）

将属于指定句柄的扩展内存逻辑页映射到扩展内存的物理页。

要使用扩展内存，必须安装符合 4.0 版的 Lotus/Intel/Microsoft 扩展内存规范 (LIM EMS) 的扩展内存设备驱动程序。

xm [lpage] [ppage] [handle]

参数

lpage

指定要映射到物理页 ppage 的扩展内存的逻辑页面号。

ppage

指定将 lpage 映射到的物理页面号。

handle

指定句柄。

有关使用扩展内存的其他 Debug 命令的信息，请单击“相关主题”列表中的 XA（分配扩展内存）、XD（释放扩展内存）或 XS（显示扩展内存）。

范例

要将句柄 0003 的逻辑页 5 映射到物理页 2，请键入以下命令：

xm 5 2 0003

如果命令成功，Debug 将显示下列消息：

Logical page 05 mapped to physical page 02

Debug:XS（显示扩展内存状态）

显示有关扩展内存状态的信息。

要使用扩展内存，必须安装符合 4.0 版的 Lotus/Intel/Microsoft 扩展内存规范 (LIM EMS) 的扩展内存设备驱动程序。

xs

参数

该命令不带参数。

有关使用扩展内存的其他 Debug 命令的信息，请单击“相关主题”列表中的 XA（分配扩展内存）、XD（释放扩展内存）或 XM（映射扩展内存页）。

说明

Debug 显示的信息有如下格式：

```
Handle xx has xx pages allocated
Physical page xx = Frame segment xx
xx of a total xx EMS pages have been allocated
xx of a total xx EMS handles have been allocated
```

范例

要显示扩展内存信息，请键入以下命令：

```
xs
```

Debug 显示与以下类似的信息：

```
Handle 0000 has 0000 pages allocated
Handle 0001 has 0002 pages allocated
Physical page 00 = Frame segment C000
Physical page 01 = Frame segment C400
Physical page 02 = Frame segment C800
Physical page 03 = Frame segment CC00
2 of a total 80 EMS pages have been allocated
2 of a total FF EMS handles have been allocated
```