

调试工具 DEBUG 的使用

1.1. 实验目的

- 1、学习如何启动在 **Windows** 的命令模式下启动 **DEBUG**;
- 2、掌握 **DEBUG** 的常用基本命令;
- 3、学习如何用 **DEBUG** 进行跟踪调试。

1.2. 预备知识

1、进制转换

需要同学们熟练掌握二进制、八进制、十进制和十六进制的互相转换算法。

2、寄存器

寄存器是 **CPU** 内部的数据存储资源，是汇编程序员能直接使用的硬件资源之一。寄存器的存取速度比 **Cache** 还要快。

在 **16 位 CPU** 中，总共有 **4 个 16 位数据寄存器 AX、BX、CX 和 DX**，每个 **16 位寄存器** 又可分为 **2 个 8 位寄存器**（例如 **AX** 的高八位称为 **AH**，低八位称为 **AL**）；**2 个变址寄存器 DI 和 SI**；**2 个指针寄存器 SP 和 BP**；**4 个段寄存器 ES、CS、SS 和 DS**；**1 个标志寄存器 FLAG**；**1 个指令指针寄存器 IP**。

3、标志位。

标志寄存器 **FLAG** 的每个位都可以作为标志位。**16 位 CPU** 使用其中八个位表示溢出、中断、进位等状态。每个标志位都有置位和复位两种状态，它们在 **DEBUG** 的表示方法见下表：

表 1.1 DEBUG 中标志位的符号表示

标志名称	溢出 OF	方向 DF	中断 IF	负号 SF	零 ZF	辅助进位 AF	奇偶 PF	进位 CF
置位状态	OV	DN	EI	NG	ZR	AC	PE	CY
复位状态	NV	UP	DI	PL	NZ	NA	PO	NC

1.3. DEBUG 的命令集

表 1.2 DEBUG 命令及其含义

命令格式	功能说明
A [地址]	输入汇编指令
C [范围] 起始地址	对由“范围”指定的区域与“起始地址”指定的同大小区域进行比较，显示不相同的单元
D [范围]	显示指定范围内的内存单元内容

E	地址 字节值表	用值表中的值替换从“地址”开始的内存单元内容
F	范围 字节值表	用指定的字节值表来填充内存区域
G	[=起始地址] [断点地址]	从起点(或当前地点)开始执行，到终点结束
H	数值 1 数值 2	显示二个十六进制数值之和、差
I	端口地址	从端口输入
L	[地址 [驱动器号 扇区 扇区数]]	从磁盘读
M	范围 地址	把“范围”内的字节值传送到从“地址”开始的单元
N	文件标识符 [文件标识符...]	指定文件名，为读/写文件做准备
O	端口地址 字节值	向端口输出
P	[=地址] [指令数]	按执行过程，但不进入子程序调用或软中断
Q		退出 DEBUG ，不保存正在调试的文件
R	[寄存器名]	显示和修改寄存器内容
S	范围 字节值表	在内存区域内搜索指定的字节值表。如果找到，显示起始地址，否则，什么也不显示
T	[=地址] [指令数]	跟踪执行，从起点(或当前地点)执行若干条指令
U	[范围]	反汇编，显示机器码所对应的汇编指令
W	[地址 [驱动器号 扇区 扇区数]]	向磁盘写内容，(BX 、 CX)为写入字节数

关于参数的几点说明：

1. 进制：在 **DEBUG** 中输入或显示的数据都是十六进制形式
2. 分隔：命令和参数、参数和参数之间要用空格、逗号或制表符等分隔
3. 地址：用“段值：偏移量”的形式来表示地址，也可用段寄存器来代表“段值”
例如：1000:0，ds:10，es:200，cs:30 等
4. 范围：表示地址范围，它有二种表示方式：“地址 1 地址 2”和“地址 1 长度”。
其中：“地址 1”表示起始地址，要用“段值:偏移量”来表达；
“地址 2”表示终止地址，只用“偏移量”来表示；
“长度”用字母'L'开头的数值来表示。
例如：100:50 100——段值为 **100**，偏移量从 **50** 到 **100** 的内存区域；
100:50 L100——段值为 **100**，偏移量从 **50** 开始的 **100** 个字节区域。
5. 端口地址：二位十六进制数值
6. 字节值：二位十六进制数值
7. 字节值表：由若干个字节值组成，也可以是用引号括起来的字符串
8. 驱动器号：**0**—驱动器 **A**、**1**—驱动器 **B**、**2**—驱动器 **C**、**3**—驱动器 **D** 等

1.4. DEBUG 使用示例

1.4.1 启动 DEBUG

1. 打开 Windows 命令窗口

在 **Windows 95/98** 的环境中，打开命令窗口的步骤为：点击“开始”→“运行”，输入“**command**”命令；

在 **WindowsXP** 的环境中，打开命令窗口的步骤为：点击“开始”→“运行”，输入“**cmd**”命

令；

2. 启动 DEBUG

在命令窗口中启动 **DEBUG**，启动命令一般为：**DEBUG [文件名] [参数表]**。其中：文件名指定被调试的文件，其**包括名和后缀**，参数表是被调试文件运行时所需要的参数。被调试的文件可以是系统中的任何文件，但**通常它们的后缀为.EXE 或.COM**。

当 **DEBUG** 启动成功后，将显示连接符“-”，这时，可输入各种 **DEBUG** 命令。**DEBUG** 中标志位的符号表示如表 1.1 所示，其所有命令及其含义如表 1.2 所示。

关于使用命令的几点说明：

1. 在提示符“-”下才能输入命令，在按“回车”键后，该命令才开始执行
2. 命令是单个字母，命令和参数的大小写可混合输入
3. 可用 **F1**、**F2**、**F3**、**Ins**、**Del**、左移键、右移键等编辑键来编辑本行命令
4. 当命令出现语法错误时，将在出错位置显示“^ Error”
5. 可用 **Ctrl+C** 或 **Ctrl+Break** 来终止当前命令的执行，还可用 **Ctrl+S** 或 **Ctrl+Num Lock** 来暂停屏幕显示(当连续不断地显示信息时)

以下通过实现十九个示例来熟悉 **DEBUG** 的命令集和基本的汇编指令。

1.4.2. R 命令的使用

R 命令作用：**观看和修改寄存器的值**。

在提示符“-”下输入以下命令：**R**。**DEBUG** 将会显示出当前所有寄存器和标志位的状态。

```
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0AF5 ES=0AF5 SS=0AF5 CS=0AF5 IP=0100  NU UP EI PL NZ NA PO NC
0AF5:0100 B400          MOV     AH,00
-
```

接下来再输入命令 **RCX**。在提示符“:”后输入 **100**。该命令的作用是将寄存器 **CX** 的值设置为 **100**（注意：**DEBUG** 使用的是十六进制，这里的 **100** 相当于十进制的 **256**。）

最后再执行 **R** 命令，观看修改后的寄存器值。

```
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0AF5 ES=0AF5 SS=0AF5 CS=0AF5 IP=0100  NU UP EI PL NZ NA PO NC
0AF5:0100 B400          MOV     AH,00
-rcx
CX 0000
:100
-r
AX=0000 BX=0000 CX=0100 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0AF5 ES=0AF5 SS=0AF5 CS=0AF5 IP=0100  NU UP EI PL NZ NA PO NC
0AF5:0100 B400          MOV     AH,00
-
```

1.4.3. H 命令的使用

H 命令作用：**计算两个十六进制数的和与差**。

在提示符“-”下输入以下命令：**H 10 1**。观看命令执行结果。

```
-h 10 1
0011 000F
```

运行结果的前一个数是计算出来的和，后一个数是计算出来的差。计算结果均用十六进制形式表示。

1.4.4. D 命令的使用

D 命令作用：显示内存区域的内容。

在提示符“-”下连续执行命令 R、D、D。观看命令执行结果。

```
C:\>debug
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0AF5 ES=0AF5 SS=0AF5 CS=0AF5 IP=0100  NV UP EI PL NZ NA PO NC
0AF5:0100 B400          MOV     AH,00
-d
0AF5:0100  B4 00 8A 07 E8 DC E2 74-02 FE C4 AC 3C 3F 75 27  .....t....<?u'
0AF5:0110  80 FC 00 74 20 80 FC 01-75 22 3A CE 34 00 E4 0A  ....t...u"...4...
0AF5:0120  20 74 0A 80 3C 3F 75 14-83 F9 01 76 0F 8A 07 AA  t...<?u...v....
0AF5:0130  43 46 49 FE C4 8A 07 3C-20 74 01 AA 43 E2 BC C3  CFI....< t..C...
0AF5:0140  F6 46 04 02 75 43 8B 05-83 C2 05 57 B8 00 6C BB  .F..uC....W..l..
0AF5:0150  40 00 33 C9 8B F2 BA 01-01 CD 21 5F 73 15 E8 8C  @.3.....!s....
0AF5:0160  DB 3D 02 00 74 23 3D 03-00 74 1E 3D 05 00 74 19  .=..t#=.t.=.t..
0AF5:0170  E9 AB D8 8B D8 B8 00 44-C0 21 B4 3E CD 21 F6 C2  .....D.!>.!...
-d
0AF5:0180  80 75 53 F6 46 04 04 74-40 8B 56 05 80 FA 00 74  .u$.F...tM.V....t
0AF5:0190  05 80 FE 3A 74 02 B2 40-80 CA 20 80 EA 60 E8 3C  ....:t...@...`.<
0AF5:01A0  E4 73 06 E8 47 DB E9 75-D8 8B D5 83 C2 05 8A 7E  .s..G...u.....~
0AF5:01B0  04 80 E7 06 80 FF 06 75-18 8B 76 02 B3 3A 38 5C  .....u...v...:8\
0AF5:01C0  FE 75 06 C6 46 00 02 EB-05 C6 46 00 01 4E E9 83  .u..F....F..N...
0AF5:01D0  00 80 FF 02 75 05 C6 46-00 00 C3 E8 8C EB B4 3B  ....u..F.....;
0AF5:01E0  CD 21 72 39 8B FA 33 C0-8B C8 49 26 8A 05 47 0A  .!r9..3...I&..G.
0AF5:01F0  C0 74 DC 32 E4 E8 EB E1-74 F1 47 FE C4 EB EC 4F  .t.2....t.G....0
```

前面已经介绍过了，命令 R 的作用是显示当前寄存器的值。而命令 D 的作用是显示内存区域的内容，最左边是内存的起始地址，中间以十六进制的形式显示内存值，最右边是以 ASCII 码的形式显示内存值。每行最多显示 16 个字节的内容。

命令 D 可以带参数也可省略参数。设 DEBUG 启动时 DS 的值为 X，当省略参数时，命令 D 显示内容以 X:100 为起始，每次显示 128 个字节的内容。以后再执行不带参数的命令 D 时，DEBUG 将按上次的位置接着显示下去。

带参数时 DEBUG 能够显示指定地址范围的内容。带参数的方式有三种：

方式一：d [起始位置]。DEBUG 从起始位置开始显示 128 个字节的内容。在提示符“-”下执行命令 D 1AF5:100。观看命令执行结果。

```
-d 1af5:100
1AF5:0100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
1AF5:0110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
1AF5:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
1AF5:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
1AF5:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
1AF5:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
1AF5:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
1AF5:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
```

方式二：d [起始位置] [结束位置]。DEBUG 从起始位置开始一直显示到结束位置。在提示符“-”下执行命令 D DS:100 1FF。观看命令执行结果。

```
-d ds:100 1ff
0AF5:0100  B4 00 8A 07 E8 DC E2 74-02 FE C4 AC 3C 3F 75 27  .....t....<?u'
0AF5:0110  80 FC 00 74 20 80 FC 01-75 22 3A CE 34 00 E4 0A  ....t...u"...4...
```

方式三：d [起始位置] [L 长度]，长度以 L 参数为标识。DEBUG 从起始位置开始显示指定长度

的内容。在提示符“-”下执行命令 **D DS:100 L10**。观看命令执行结果。

```
-d ds:100 L10
0AF5:0100  B4 00 8A 07 E8 DC E2 74-02 FE C4 AC 3C 3F 75 27  .....t....<?u'
```

1.4.5. E 命令的使用

E 命令作用：**改变内存单位的内容。**

E 命令的使用方式为：**E [起始位置]**。

在提示符“-”下输入以下命令：**E 1AF5:100**。

```
-e 1af5:0100
1AF5:0100  00.12  00.34  00.  00.56
-d 1af5:100
1AF5:0100  12 34 00 56 00 00 00 00-00 00 00 00 00 00 00 00  .4.V.....
1AF5:0110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
```

DEBUG 首先显示[1AF5:0000]的内容 **00.**，这时可以修改该字节的值。如果还要修改后续的内容，可以按空格键继续。当要跳过某个字节时，可以按连续的两个空格跳到后一个字节去。

1.4.6. F 命令的使用

F 命令作用：**使用指定的值填充指定内存区域中的地址。**

F 命令的使用方式为：**F [范围] [填充列表]**。

在提示符“-”下输入以下命令：**F 1AF5:100 L20 1 2 3 4 5**。执行命令 **D 1AF5:100** 观看命令执行结果。

```
-F 1AF5:100 L20 1 2 3 4 5
-d 1AF5:100
1AF5:0100  01 02 03 04 05 01 02 03-04 05 01 02 03 04 05 01  .....
1AF5:0110  02 03 04 05 01 02 03 04-05 01 02 03 04 05 01 02  .....
1AF5:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
```

说明：该命令是用字节序列 **01、02、03、04、05** 轮流填充从 **1AF5:100** 开始长度为 **20H** 的内存区域。

在提示符“-”下输入以下命令：**F 1AF5:100 13F 41 42 43 44**。

```
-f 1af5:100 13f 41 42 43 44
-d 1af5:100
1AF5:0100  41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44  ABCDABCDABCDABCD
1AF5:0110  41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44  ABCDABCDABCDABCD
1AF5:0120  41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44  ABCDABCDABCDABCD
1AF5:0130  41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44  ABCDABCDABCDABCD
1AF5:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
1AF5:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
1AF5:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
1AF5:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  .....
```

说明：该命令是用字节序列 **41、42、43、44** 轮流填充从 **1AF5:100** 开始一直到 **1AF5:13F** 的内存区域。

1.4.7. M 命令的使用

M 命令作用：**将指定内存区域的数据复制到指定的地址去。**

M 命令的使用方式为：**M [范围] [指定地址]**。

在提示符“-”下输入以下命令：**M 1AF5:100 13F 1AF5:140**。执行命令 **D 1AF5:100** 观看命令执行结果。

```

-m 1af5:100 13f 1af5:140
-d 1af5:100
1AF5:0100 41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44 ABCDABCDABCDABCD
1AF5:0110 41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44 ABCDABCDABCDABCD
1AF5:0120 41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44 ABCDABCDABCDABCD
1AF5:0130 41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44 ABCDABCDABCDABCD
1AF5:0140 41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44 ABCDABCDABCDABCD
1AF5:0150 41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44 ABCDABCDABCDABCD
1AF5:0160 41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44 ABCDABCDABCDABCD
1AF5:0170 41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44 ABCDABCDABCDABCD

```

1.4.8. C 命令的使用

C 命令作用：将两块内存的内容进行比较。

C 命令的使用方式为：**C [范围] [指定地址]**，意思就是将指定范围的内存区域与从指定地址开始的相同长度的内存区域逐个字节进行比较，列出不同的内容。

在提示符“-”下输入以下命令：**C 1AF5:100 13F 1AF5:140**。由于两块内容完全相同，所以命令执行后没有任何显示。

在提示符“-”下输入以下命令：**C 1AF5:100 107 1AF5:180**，比较的区域长度为 8 个字节。命令执行后列出比较结果不同的各个字节。

```

-C 1AF5:100 13F 1AF5:140
-C 1AF5:100 107 1AF5:180
1AF5:0100 41 00 1AF5:0180
1AF5:0101 42 00 1AF5:0181
1AF5:0102 43 00 1AF5:0182
1AF5:0103 44 00 1AF5:0183
1AF5:0104 41 00 1AF5:0184
1AF5:0105 42 00 1AF5:0185
1AF5:0106 43 00 1AF5:0186
1AF5:0107 44 00 1AF5:0187

```

1.4.9. S 命令的使用

S 命令作用：在指定的内存区域中搜索指定的串。

S 命令的使用方式为：**S [范围] [指定串]**。

在提示符“-”下输入以下命令：**D 1AF5:100 11F**。显示该区域的内存值。

在提示符“-”下输入以下命令：**S 1AF5:100 11F 41 42 43 44**。搜索该区域是否存在字节串 **41 42 43 44**，并将搜索结果一一列出。

```

-d 1af5:100 11f
1AF5:0100 41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44 ABCDABCDABCDABCD
1AF5:0110 41 42 43 44 41 42 43 44-41 42 43 44 41 42 43 44 ABCDABCDABCDABCD
-s 1af5:100 11f 41 42 43 44
1AF5:0100
1AF5:0104
1AF5:0108
1AF5:010C
1AF5:0110
1AF5:0114
1AF5:0118
1AF5:011C

```

从执行结果可以看出，总共搜索到八处。

1.4.10. A 命令的使用

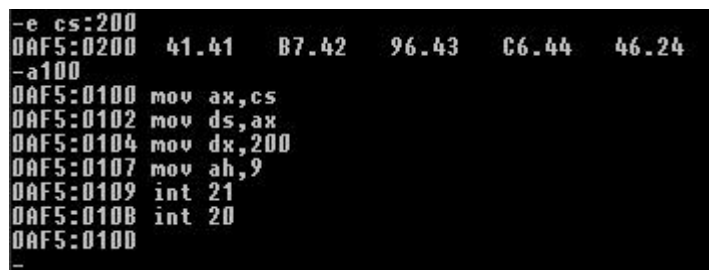
A 命令作用：输入汇编指令。

以下的程序要在屏幕上显示“ABCD”四个字符。

首先用 **E** 命令将“ABCD”四个字符预先放在内存 **CS:200** 处，然后执行 **A100** 命令输入汇编程序代码：

```
MOV AX,CS
MOV DS,AX
MOV DX,200
MOV AH,9
INT 21
INT 20
```

（说明：前两行汇编指令用于将段寄存器 **CS** 的值赋给段寄存器 **DS**。第三到第五行汇编代码的作用是显示以“\$”为结尾的字符串。最后一行用于结束程序。



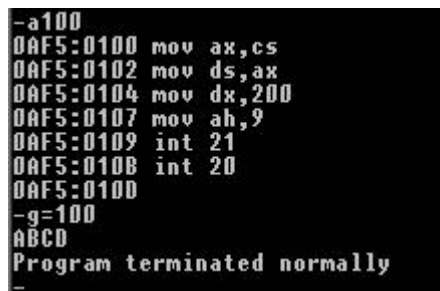
```
-e cs:200
0AF5:0200  41.41  B7.42  96.43  C6.44  46.24
-a100
0AF5:0100  mov ax,cs
0AF5:0102  mov ds,ax
0AF5:0104  mov dx,200
0AF5:0107  mov ah,9
0AF5:0109  int 21
0AF5:010B  int 20
0AF5:010D
-
```

1.4.11. G 命令的使用

G 命令作用：执行汇编指令。

G 命令的使用方法是：**G** [=起始地址] [断点地址]，意思是从起始地址开始执行到断点地址。如果不设置断点，则程序一直运行到中止指令才停止。

在设置完示例九的内存数据并且输入完示例九的程序后运行这些汇编代码。在 **DEBUG** 中执行命令 **G=100**，观看运行结果。



```
-a100
0AF5:0100  mov ax,cs
0AF5:0102  mov ds,ax
0AF5:0104  mov dx,200
0AF5:0107  mov ah,9
0AF5:0109  int 21
0AF5:010B  int 20
0AF5:010D
-g=100
ABCD
Program terminated normally
-
```

汇编程序运行后在屏幕上显示出“ABCD”四个字符。

接下来在 **DEBUG** 中执行 **G=100 10B**，意思是从地址 **CS: 100** 开始，一直运行到 **CS: 10B** 停止。观看运行结果。

命令执行后，不但显示出字符串“ABCD”，而且列出当前寄存器和标志位的值。

```

-a100
0AF5:0100 mov ax,cs
0AF5:0102 mov ds,ax
0AF5:0104 mov ah,9
0AF5:0106 mov dx,200
0AF5:0109 int 21
0AF5:010B int 20
0AF5:010D
-g=100
ABCD
Program terminated normally
-g=100 10b
ABCD
AX=0924 BX=0000 CX=0000 DX=0200 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0AF5 ES=0AF5 SS=0AF5 CS=0AF5 IP=010B  NV UP EI PL NZ NA PO NC
0AF5:010B CD20          INT     20

```

1.4.12. U 命令的使用

U 命令作用：对机器代码反汇编显示。

U 命令的使用方法是：U [范围]。如果范围参数只输入了起始地址，则只对 20H 个字节的机器代码反汇编。执行命令 U100，观看反汇编结果。

```

-g=100
ABCD
Program terminated normally
-g=100 10b
ABCD
AX=0924 BX=0000 CX=0000 DX=0200 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0AF5 ES=0AF5 SS=0AF5 CS=0AF5 IP=010B  NV UP EI PL NZ NA PO NC
0AF5:010B CD20          INT     20
-u100
0AF5:0100 8CC8          MOV     AX,CS
0AF5:0102 8ED8          MOV     DS,AX
0AF5:0104 B409          MOV     AH,09
0AF5:0106 BA0002    MOV     DX,0200
0AF5:0109 CD21          INT     21
0AF5:010B CD20          INT     20
0AF5:010D 3F            AAS
0AF5:010E 7527          JNZ     0137
0AF5:0110 80FC00        CMP     AH,00
0AF5:0113 7420          JZ      0135
0AF5:0115 80FC01        CMP     AH,01
0AF5:0118 7522          JNZ     013C
0AF5:011A 3ACE          CMP     CL,DH
0AF5:011C 3400          XOR     AL,00
0AF5:011E E40A          IN      AL,0A

```

执行命令 U100 10B，观看反汇编结果。该命令的作用是对从 100 到 10B 的机器代码进行反汇编。

```

-u100 10c
0AF5:0100 8CC8          MOV     AX,CS
0AF5:0102 8ED8          MOV     DS,AX
0AF5:0104 B409          MOV     AH,09
0AF5:0106 BA0002    MOV     DX,0200
0AF5:0109 CD21          INT     21
0AF5:010B CD20          INT     20

```

1.4.13. N 命令的使用

N 命令作用：设置文件名，为将刚才编写的汇编程序存盘做准备。

以下的 DEBUG 命令序列作用将刚才的汇编程序存为磁盘的 COM 可执行程序。

D200 20F


```

U100 10C
N E:\FIRST.COM
RCX
:110
W

```

第一和第二条命令的作用是检查一下刚才编写的汇编指令。第三条命令的作用是设置存盘文件名为 **E:\FIRST.COM**，第四条命令的作用是设置存盘文件大小为 **110H** 个字节。最后一条命令是将文件存盘。

```

-j200 20f
0AF5:0200 41 42 43 44 24 00 02 0A-E4 75 05 3A 45 FF 74 05 ABCD$....u.:E.t.
-u100 10c
0AF5:0100 8CC8      MOV     AX,CS
0AF5:0102 8ED8      MOV     DS,AX
0AF5:0104 B409      MOV     AH,09
0AF5:0106 BA0002    MOV     DX,0200
0AF5:0109 CD21      INT     21
0AF5:010B CD20      INT     20
-n e:\first.com
-rcx
CX 0000
:110
-w
Writing 00110 bytes
-

```

文件存盘后执行 **E:\FIRST.COM**，观看存盘的可执行文件的运行效果。

```

E:\>first
ABCD
E:\>

```

1.4.14. W 命令的使用

W 命令作用：将文件或者特定扇区写入磁盘。

在示例“N 命令的使用”中已经实验了如何使用 **W** 命令将文件存盘。

在没有很好地掌握汇编语言和磁盘文件系统前，暂时不要使用 **W** 命令写磁盘扇区，否则很容易损坏磁盘文件，甚至破坏整个磁盘的文件系统。

1.4.15. L 命令的使用

L 命令作用：从磁盘中将文件或扇区内容读入内存。

将文件调入内存必须先用 **DEBUG** 的 **N** 命令设定文件名。以下例子是将 **E:\FIRST.COM** 读入内容。

```

N FIRST.COM
L

```

观看调入程序的汇编代码可以使用 **DEBUG** 的 **U** 命令，用 **U100** 观看调入的 **COM** 文件。

```

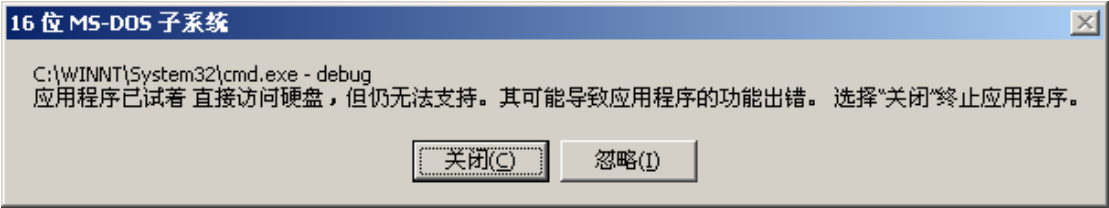
-n e:\first.com
-l
-u100
0B47:0100 8CC8      MOV     AX,CS
0B47:0102 8ED8      MOV     DS,AX
0B47:0104 B409      MOV     AH,09
0B47:0106 BA0002    MOV     DX,0200
0B47:0109 CD21      INT     21
0B47:010B CD20      INT     20

```

读取磁盘扇区的方式是：L [内存地址] [磁盘驱动器号] [起始扇区] [扇区数]。“内存地址”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“内存地址”的话，**DEBUG** 将使用 **CS** 寄存器中的当前地址。“磁盘驱动器号”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。“起始扇区”指定要加载其内容的第一个扇区的十六进制数。“扇区数”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文件时，才能使用[磁盘驱动器号] [起始扇区] [扇区数]参数。

例如：要将 C 盘第一扇区读取到内存 **DS:300** 的位置，相应的 **DEBUG** 命令为 **L DS:300 2 1 1**。但是由于 **Windows** 操作系统对文件系统的保护，这条命令可能会被操作系统禁止运行。



1.4.16. T 命令的使用

T 命令作用：执行汇编程序，**单步跟踪**。

T 命令的使用方式是 **T [=地址] [指令数]**。如果忽略“地址”的话，**T** 命令从 **CS:IP** 处开始运行。“指令数”是要单步执行的指令的数量。

以下示例对 **E:\FIRST.COM** 进行单步跟踪。

N E:\FIRST.COM

L

U100 10B

R

T=100

T

```
-n e:\first.com
-l
-u 100 10b
0B47:0100 8CC8      MOV     AX,CS
0B47:0102 8ED8      MOV     DS,AX
0B47:0104 B409      MOV     AH,09
0B47:0106 BA0002    MOV     DX,0200
0B47:0109 CD21      INT     21
0B47:010B CD20      INT     20
-r
AX=0000 BX=0000 CX=0110 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0B47 ES=0B47 SS=0B47 CS=0B47 IP=0100  NU UP EI PL NZ NA PO NC
0B47:0100 8CC8      MOV     AX,CS
-t=100
AX=0B47 BX=0000 CX=0110 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0B47 ES=0B47 SS=0B47 CS=0B47 IP=0102  NU UP EI PL NZ NA PO NC
0B47:0102 8ED8      MOV     DS,AX
-t
AX=0B47 BX=0000 CX=0110 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0B47 ES=0B47 SS=0B47 CS=0B47 IP=0104  NU UP EI PL NZ NA PO NC
0B47:0104 B409      MOV     AH,09
```

第一、二条命令是装入文件，第三条命令是列出程序反汇编代码，第四条命令是显示当前寄存器值，第五条命令是从 **CS:100** 处开始单步跟踪，第六条命令是继续跟踪后续的指令。

1.4.17. P 命令的使用

P 命令作用：执行汇编程序，单步跟踪。与 **T** 命令不同的是：**P** 命令不会跟踪进入子程序或软中断。

P 命令的使用方式与 **T** 命令的使用方式完全相同。

```
AX=0B47 BX=0000 CX=0110 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0B47 ES=0B47 SS=0B47 CS=0B47 IP=0102  NU UP EI PL NZ NA PO NC
0B47:0102 8ED8          MOV     DS,AX
-t
AX=0B47 BX=0000 CX=0110 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0B47 ES=0B47 SS=0B47 CS=0B47 IP=0104  NU UP EI PL NZ NA PO NC
0B47:0104 B409          MOV     AH,09
-p
AX=0947 BX=0000 CX=0110 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0B47 ES=0B47 SS=0B47 CS=0B47 IP=0106  NU UP EI PL NZ NA PO NC
0B47:0106 BA0002       MOV     DX,0200
-p
AX=0947 BX=0000 CX=0110 DX=0200 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0B47 ES=0B47 SS=0B47 CS=0B47 IP=0109  NU UP EI PL NZ NA PO NC
0B47:0109 CD21          INT     21
-p
ABCD
AX=0924 BX=0000 CX=0110 DX=0200 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0B47 ES=0B47 SS=0B47 CS=0B47 IP=010B  NU UP EI PL NZ NA PO NC
0B47:010B CD20          INT     20
-
```

1.4.18. I 命令的使用

I 命令作用：从计算机输入端口读取数据并显示。

I 命令的用法是 **I** [端口地址]。例如从 **3F8** 号端口读取数据并显示的命令为：**I 3F8**。这里不对该命令做解释。

1.4.19. O 命令的使用

O 命令作用：向计算机输出端口送出数据。

O 命令的用法是 **O** [端口地址] [字节值]。例如向 **278** 号端口发出数据 **20H** 的命令为：**O 278 20**。这里不对该命令做解释。

1.4.20. Q 命令的使用

Q 命令的作用是退出 **DEBUG**，回到 **DOS** 状态。