

```

mov ax,0000    (B8 00 00)
add ax,0123    (05 23 01)
mov bx,ax      (8B D8)
jmp bx         (FF E3)

```

这段长度为 10 字节的指令,存在从 123B0H~123BAH 的一组内存单元中,我们就可以认为,123B0H~123BAH 这段内存是用来存放代码的,是一个代码段,它的段地址为 123BH,长度为 10 字节。

如何使得代码段中的指令被执行呢?将一段内存当作代码段,仅仅是我们在编程时的一种安排,CPU 并不会由于这种安排,就自动地将我们定义的代码段中的指令当作指令来执行。CPU 只认被 CS:IP 指向的内存单元中的内容为指令。所以,要让 CPU 执行我们放在代码段中的指令,必须要将 CS:IP 指向所定义的代码段中的第一条指令的首地址。对于上面的例子,我们将一段代码存放在 123B0H~123BAH 内存单元中,将其定义为代码段,如果要想这段代码得到执行,可设 CS=123BH、IP=0000H。

2.9 节~2.12 节 小 结

1. 段地址在 8086CPU 的段寄存器中存放。当 8086CPU 要访问内存时,由段寄存器提供内存单元的段地址。8086CPU 有 4 个段寄存器,其中 CS 用来存放指令的段地址。
2. CS 存放指令的段地址,IP 存放指令的偏移地址。
8086 机中,任意时刻,CPU 将 CS:IP 指向的内容当作指令执行。
3. 8086CPU 的工作过程:
 - (1) 从 CS:IP 指向内存单元读取指令,读取的指令进入指令缓冲器;
 - (2) IP 指向下一条指令;
 - (3) 执行指令。(转到步骤(1),重复这个过程。)
4. 8086CPU 提供转移指令修改 CS、IP 的内容。

检测点 2.3

下面的 3 条指令执行后,CPU 几次修改 IP? 都是在什么时候? 最后 IP 中的值是多少?

```

mov ax,bx
sub ax,ax
jmp ax

```

实验 1 查看 CPU 和内存,用机器指令和汇编指令编程

1. 预备知识: Debug 的使用

在以后所有的实验中,都将用到 Debug 程序,首先学习一下它的主要用法。

(1) 什么是 Debug?

Debug 是 DOS、Windows 都提供的实模式(8086 方式)程序的调试工具。使用它, 可以查看 CPU 各种寄存器中的内容、内存的情况和在机器码级跟踪程序的运行。

(2) 我们用到的 Debug 功能

- 用 Debug 的 R 命令查看、改变 CPU 寄存器的内容;
- 用 Debug 的 D 命令查看内存中的内容;
- 用 Debug 的 E 命令改写内存中的内容;
- 用 Debug 的 U 命令将内存中的机器指令翻译成汇编指令;
- 用 Debug 的 T 命令执行一条机器指令;
- 用 Debug 的 A 命令以汇编指令的格式在内存中写入一条机器指令。

Debug 的命令比较多, 共有 20 多个, 但上述 6 个命令是和汇编语言学习密切相关的。在以后的实验中, 我们还会用到一个 P 命令。

(3) 进入 Debug

Debug 是在 DOS 方式下使用的程序。我们在进入 Debug 前, 应先进入到 DOS 方式。用以下方式可以进入 DOS:

- ① 重新启动计算机, 进入 DOS 方式, 此时进入的是实模式的 DOS。
- ② 在 Windows 中进入 DOS 方式, 此时进入的是虚拟 8086 模式的 DOS。

下面说明在 Windows 2000 中进入 Debug 的一种方法, 在 Windows 98 中进入的方法与此类似。

单击【开始】菜单中的【运行】命令, 如图 2.28 所示, 打开【运行】对话框, 如图 2.29 所示, 在文本框中输入“command”后, 单击【确定】按钮。



图 2.28 选择【运行】命令

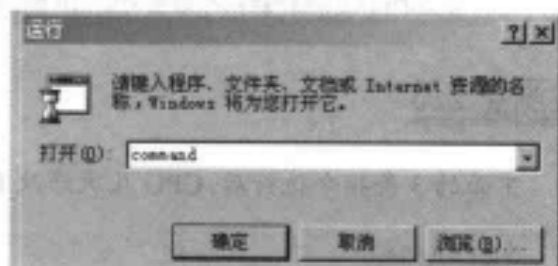


图 2.29 在文本框中输入“command”

进入 DOS 方式后, 如果显示为窗口方式, 可以按下 Alt+Enter 键将窗口变为全屏方式。然后运行 Debug 程序, 如图 2.30 所示, 这个程序在 Windows 98 中通常在 C:\windows98\command 下, 在 Windows 2000 中通常在 C:\winnt\system 下。由于系统指定了搜索路径, 所以在任何一个路径中都可以运行。

```
Microsoft(R) Windows DOS
(C)Copyright Microsoft Corp 1990-1999.

C:\>debug
```

图 2.30 运行 Debug 程序

(4) 用 R 命令查看、改变 CPU 寄存器的内容

我们已知道了 AX、BX、CX、DX、CS、IP 这 6 个寄存器，现在看一下它们之中的内容，如图 2.31 所示。其他寄存器 SP、BP、SI、DI、DS、ES、SS、标志寄存器等先不予理会。

```
Microsoft(R) Windows DOS
(C)Copyright Microsoft Corp 1990-1999.

C:\>debug
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0CA2 ES=0CA2 SS=0CA2 CS=0CA2 IP=0100  NO UP EI PL NZ NA PO NC
0CA2:0100 027548      ADD     DH,[DI+48]          DS:0040=00
```

图 2.31 使用 R 命令查看 CPU 中各个寄存器中的内容

注意 CS 和 IP 的值，CS=0CA2，IP=0100，也就是说，内存 0CA2:0100 处的指令为 CPU 当前要读取、执行的指令。在所有寄存器的下方，Debug 还列出了 CS:IP 所指向的内存单元处所存放的机器码，并将它翻译为汇编指令。可以看到，CS:IP 所指向的内存单元为 0CA2:0100，此处存放的机器码为 027548，对应的汇编指令为 ADD DH,[DI+48](这条指令的含义我们还不知道，先不必深究)。

Debug 输出的右下角还有一个信息：“DS:0040=0”，以后会进行说明，这里同样不必深究。

还可以用 R 命令来改变寄存器中的内容，如图 2.32 所示。

```
C:\>debug
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B39 ES=0B39 SS=0B39 CS=0B39 IP=0100  NO UP EI PL NZ NA PO NC
0B39:0100 40      INC     AX
-r ax
AX 0000
:1111
-r
AX=1111 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B39 ES=0B39 SS=0B39 CS=0B39 IP=0100  NO UP EI PL NZ NA PO NC
0B39:0100 40      INC     AX
```

图 2.32 用 R 命令修改寄存器 AX 中的内容

若要修改一个寄存器中的值，比如 AX 中的值，可用 R 命令后加寄存器名来进行，输入“rax”后按 Enter 键，将出现“:”作为输入提示，在后面输入要写入的数据后按 Enter 键，即完成了对 AX 中内容的修改。若想看一下修改的结果，可再用 R 命令查看，如图 2.33 所示。

```

C:\>debug
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B39 ES=0B39 SS=0B39 CS=0B39 IP=0100  NO UP EI PL NZ NA PO NC
0B39:0100 40      INC     AX
-r ip
IP 0100
:200
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B39 ES=0B39 SS=0B39 CS=0B39 IP=0200  NO UP EI PL NZ NA PO NC
0B39:0200 5B      POP     BX
-r cs
CS 0B39
:FF00
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B39 ES=0B39 SS=0B39 CS=FF00 IP=0200  NO UP EI PL NZ NA PO NC
FF00:0200 51      PUSH    CX

```

图 2.33 用 R 命令修改 CS 和 IP 中的内容

在图 2.33 中，一进入 Debug，用 R 命令查看，CS:IP 指向 0B39:0100，此处存放的机器码为 40，对应的汇编指令是 INC AX；

接着，用 R 命令将 IP 修改为 200，则 CS:IP 指向 0B39:0200，此处存放的机器码为 5B，对应的汇编指令是 POP BX；

接着，用 R 命令将 CS 修改为 ff00，则 CS:IP 指向 ff00:0200，此处存放的机器码为 51，对应的汇编指令是 PUSH CX。

(5) 用 Debug 的 D 命令查看内存中的内容

用 Debug 的 D 命令，可以查看内存中的内容，D 命令的格式较多，我们这里只介绍在本次实验中用到的格式。

如果想知道内存 10000H 处的内容，可以用“d 段地址:偏移地址”的格式来查看，如图 2.34 所示。

```

C:\>debug
-d 1000:0
1000:0000 72 64 73 20 63 6F 6D 6D-65 6E 74 73 20 28 72 65      rds comments (re
1000:0010 6D 61 72 6B 73 29 20 69-6E 20 61 20 62 61 74 63      marks) in a batc
1000:0020 68 20 66 69 6C 65 20 6F-72 20 43 4F 4E 46 49 47      h file or CONFIG
1000:0030 2E 53 59 53 2E 00 0A 00-0A 52 45 4D 20 5B 63 6F      .SYS.....REM [co
1000:0040 6D 6D 65 6E 74 5D 00 0A-6B 53 75 73 7D 65 6E 64      mment]..kSuspend
1000:0050 73 20 7D 72 6F 63 65 73-73 69 6E 67 2D 6F 66 2D      s processing of
1000:0060 61 2D 62 61 74 63 68 2D-7D 72 6F 67 72 61 6D 2D      a batch program
1000:0070 61 6E 64 2D 64 69 73 7D-6C 61 79 73 2D 74 68 65      and displays the

```

图 2.34 用 D 命令查看内存 1000:0 处的内容

要查看内存 10000H 处的内容，首先将这个地址表示为段地址:偏移地址的格式，可以是 1000:0，然后用“d 1000:0”列出 1000:0 处的内容。

使用“d 段地址:偏移地址”的格式，Debug 将列出从指定内存单元开始的 128 个内存单元的内容。图 2.34 中，在使用 d 1000:0 后，Debug 列出了 1000:0~1000:7F 中的内容。

使用 D 命令, Debug 将输出 3 部分内容, 如图 2.34 所示。

中间是部分从指定地址开始的 128 个内存单元的内容, 用十六进制的格式输出, 每行的输出从 16 的整数倍的地址开始, 最多输出 16 个单元的内容。从图中, 我们可以知道, 内存 1000:0 单元中的内容是 72H, 内存 1000:1 单元中的内容是 64H; 内存 1000:0~1000:F 中的内容都在第一行; 内存 1000:10 中的内容是 6D, 内存 1000:1 处的内容是 61; 内存 1000:10~1000:1F 中的内容都在第二行。注意在每行的中间有一个“-”, 它将每行的输出分为两部分, 这样便于查看。比如, 要想从图中找出, 1000:6B 单元中内容, 可以从 1000:60 找到行, “-”前面是 1000:60~1000:67 的 8 个单元, 后面是 1000:68~1000:6F 的 8 个单元, 这样我们就可以从 1000:68 单元向后数 3 个单元, 找到 1000:6B 单元, 可以看到, 1000:6B 中的内容为 67H。

左边是每行的起始地址。

右边是每个内存单元中的数据对应的可显示的 ASCII 码字符。比如: 内存单元 1000:0、1000:1、1000:2 中存放的数据是 72H、64H、73H, 它对应的 ASCII 字符分别是“r”、“d”、“s”; 内存单元 1000:36 中的数据是 0AH, 它没有对应可显示的 ASCII 字符, Debug 就用“.”来代替。

注意: 我们看到的内存中的内容, 在不同的计算机中是不一样的, 也可能每次用 Debug 看到的内容都不相同, 因为我们用 Debug 看到的都是原来就在内存中的内容, 这些内容受随时都有可能变化的系统环境的影响。当然, 我们也可以改变内存、寄存器中的内容。

使用 d 1000:9 查看 1000:9 处的内容, Debug 将怎样输出呢? 如图 2.35 所示。

```
C:\>debug
~d1000:9
1000:0000 60 61 72 68 73 29 20 69-6E 20 61 20 62 61 74 63 nts (re
1000:0010 68 20 66 69 6C 65 20 6F-72 20 43 4F 4E 46 49 47 marks) in a batc
1000:0020 2E 53 59 53 2E 00 00 00-0A 52 45 40 20 58 63 6F h file or COMF16
1000:0030 60 60 65 6E 74 50 00 00-68 53 75 73 70 65 6E 64 .SVS.....REM [co
1000:0040 73 20 70 72 6F 63 65 73-73 69 6E 67 20 6F 66 20 mment]...kSuspend
1000:0050 61 20 62 61 74 63 68 20-70 72 6F 67 72 61 60 20 s processing of
1000:0060 61 6E 64 20 64 69 73 70-6C 61 79 73 20 74 68 65 a batch program
1000:0070 20 60 65 73 73 61 67 65-20 and displays the
1000:0080
```

图 2.35 查看 1000:9 处的内容

Debug 从 1000:9 开始显示, 一直到 1000:88, 一共是 128 个字节。第一行中的 1000:0~1000:8 单元中的内容不显示。

在一进入 Debug 后, 用 D 命令直接查看, 将列出 Debug 预设的地址处的内容, 如图 2.36 所示。

在使用“d 段地址:偏移地址”之后, 接着使用 D 命令, 可列出后续的内容, 如图 2.37 所示。

```

C:\>debug
-d
0CA2:0100 02 75 48 09 3E E6 99 FF 04 E6 99 C6 06 E8 99 FF .uH.>.....
0CA2:0110 C6 06 E9 99 00 E8 99 00 AC E8 29 E2 34 00 91 0C .....).4...
0CA2:0120 74 34 3A 04 B6 96 74 2E 3A C3 74 2A 3C 3A 74 03 t4:...t...t*:<t.
0CA2:0130 E9 5F FF 00 3E A4 90 02 75 05 E8 74 00 E8 09 46 ...>...u...t...F
0CA2:0140 E8 14 E9 40 FF 0A 09 8A E9 73 E5 0A 01 08 E9 80 ...M.....
0CA2:0150 E5 4E 5F 90 F9 C3 4E E8 51 80 CF 01 01 C0 00 80 .N...N.Q.....
0CA2:0160 E8 0A E1 46 E8 AC DF 74 00 E8 45 00 AC E8 41 00 ...F...t...E...A.
0CA2:0170 01 C0 00 40 E8 34 3C 00 75 09 80 00 AA 01 C0 00 ...E.4<u.....

```

图 2.36 列出 Debug 预设的地址处的内容

```

C:\>debug
-d1000:0
1000:0000 72 64 73 20 63 6F 60 60 65 6E 74 73 20 20 72 65 rds comments (re
1000:0010 60 61 72 68 73 29 20 69 6E 20 61 20 62 61 74 63 marks) in a batc
1000:0020 68 20 66 69 6C 65 20 6F 72 20 43 4F 4E 46 49 47 h file or CONFIG
1000:0030 2E 53 59 53 2E 00 0A 00 0A 52 45 40 20 58 63 6F .SYS.....REM [co
1000:0040 60 60 65 6E 74 50 00 0A 68 53 75 73 70 65 6E 64 mment]..kSuspend
1000:0050 73 20 70 72 6F 63 65 73 73 69 6E 67 20 6F 66 20 s processing of
1000:0060 61 20 62 61 74 63 68 20 70 72 6F 67 72 61 60 20 a batch program
1000:0070 61 6E 64 20 64 69 73 70 6C 61 79 73 20 74 68 65 and displays the
-d
1000:0080 20 60 65 73 73 61 67 65 20 22 50 72 65 73 73 20 message "Press
1000:0090 61 6E 79 00 0A 68 65 79 20 74 6F 20 63 6F 6E 74 any..key to cont
1000:00A0 69 6E 75 65 2E 2E 2E 2E 22 00 0A 00 0A 50 41 55 ine....."....PAU
1000:00B0 53 45 00 0A 40 44 69 73 70 6C 61 79 73 20 60 65 SE..Mdisplays me
1000:00C0 73 73 61 67 65 73 2C 20 6F 72 20 74 75 72 6E 73 ssages, or turns
1000:00D0 20 63 6F 60 60 61 6E 64 20 65 63 68 6F 69 6E 67 command-echoing
1000:00E0 20 6F 6E 20 6F 72 20 6F 66 66 2E 00 0A 00 0A 20 on or off.....
1000:00F0 20 45 43 48 4F 20 58 4F 4E 20 7C 20 4F 46 46 50 ECHO [ON | OFF]

```

图 2.37 列出后续的内容

也可以用指定 D 命令的查看范围, 此时采用“d 段地址:起始偏移地址 结尾偏移地址”的格式。比如要看 1000:0~1000:9 中的内容, 可以用“d 1000:0 9”实现。如图 2.38 所示。

```

C:\>debug
-d1000:0 9
1000:0000 72 64 73 20 63 6F 60 60 65 6E rds commen

```

图 2.38 查看 1000:0~1000:9 单元中的内容

如果我们就想查看内存单元 10000H 中的内容, 可以用图 2.39 中的任何一种方法看到, 因为图中的所有“段地址:偏移地址”都表示 10000H 这一物理地址。

```

C:\>debug
-d 1000:0 0
1000:0000 72
-d 0fff:10 10
0fff:0010 72
-d 0100:f000 f000
0100:f000 72

```

图 2.39 用三种不同的段地址和偏移地址查看同一个物理地址中的内容

(6) 用 Debug 的 E 命令改写内存中的内容

可以使用 E 命令来改写内存中的内容, 比如, 要将内存 1000:0~1000:9 单元中的内容

分别写为 0、1、2、3、4、5、6、7、8、9，可以用“e 起始地址 数据 数据 数据……”的格式来进行，如图 2.40 所示。

```
C:\>debug
-d 1000:0 f
1000:0000 72 64 73 20 63 6F 6D 6D-65 6E 74 73 20 28 72 65  rds comments (re
-e 1000:0 0 1 2 3 4 5 6 7 8 9
-d 1000:0 f
1000:0000 00 01 02 03 04 05 06 07-08 09 74 73 20 28 72 65  .....ts (re
```

图 2.40 用 E 命令修改从 1000:0 开始的 10 个单元的内容

图 2.40 中，先用 D 命令查看 1000:0~1000:f 单元的内容，再用 E 命令修改从 1000:0 开始的 10 个单元的内容，最后用 D 命令查看 1000:0~1000:f 中内容的变化。

也可以采用提问的方式来一个一个地改写内存中的内容，如图 2.41 所示。

```
-d 1000:10 19
1000:0010 6D 61 72 6B 73 29 20 69-6E 20 marks> in
-e 1000:10
1000:0010 6D.0 61.1 72.2 6B.1c
```

图 2.41 用 E 命令修改从 1000:10 开始的 4 个单元的内容

如图 2.41 中，可以用 E 命令以提问的方式来逐个地修改从某一地址开始的内存单元中的内容，以从 1000:10 单元开始为例，步骤如下：

- ① 输入 e 1000:10，按 Enter 键。
- ② Debug 显示起始地址 1000:0010，和第一单元(即 1000:0010 单元)的原始内容：6D，然后光标停在“.”的后面提示输入想要写入数据，此时可以有两个选择：其一为输入数据(我们输入的是 0)，然后按空格键，即用输入的数据改写当前的内存单元；其二为不输入数据，直接按空格键，则不对当前内存单元进行改写。
- ③ 当前单元处理完成后(不论是改写或没有改写，只要按了空格键，就表示处理完成)，Debug 将接着显示下一个内存单元的原始内容，并提示读者进行修改，读者可以用同样的方法处理。
- ④ 所有希望改写的内存单元改写完毕后，按 Enter 键，E 命令操作结束。

可以用 E 命令向内存中写入字符，比如：用 E 命令从内存 1000:0 开始写入：数值 1、字符“a”，数值 2，字符“b”，数值 3，字符“c”，可采用图 2.42 中所示的方法进行：

从图 2.42 中可以看出，Debug 对 E 命令的执行结果是，向 1000:0、1000:2、1000:4 单元中写入数值 1、2、3，向 1000:1、1000:3、1000:5 单元中写入字符“a”、“b”和“c”的 ASCII 码值：61H、62H、63H。

也可以用 e 命令向内存中写入字符串，比如：用 E 命令从内存 1000:0 开始写入：数值 1、字符串“a+b”、数值 2、字符串“c++”、字符 3、字符串“IBM”。如图 2.43 所示。

```
C:\>debug
-e 1000:0 1 'a' 2 'b' 3 'c'
-
-d 1000:0 f
1000:0000 01 61 02 62 03 63 00 00 00 00 00 00 00 00 00 00  .a.b.c.....
-
```

图 2.42 用 E 命令向内存中写入字符

```
C:\>debug
-e 1000:0 1 "a+b" 2 "c++" 3 "IBM"
-
-d 1000:0 f
1000:0000 01 61 2B 62 02 63 2B 2B 03 49 42 40 00 00 00 00  .a+b.c++.IBM....
-
```

图 2.43 用 E 命令向内存中写入字符串

(7) 用 E 命令向内存中写入机器码，用 U 命令查看内存中机器码的含义，用 T 命令执行内存中的机器码。

如何向内存中写入机器码呢？我们知道，机器码也是数据，当然可以用 E 命令将机器码写入内存。比如要从内存 1000:0 单元开始写入这样一段机器码：

机器码	对应的汇编指令
b80100	mov ax,0001
b90200	mov cx,0002
01c8	add ax,cx

可用如图 2.44 中所示的方法进行。

```
C:\>debug
-e 1000:0 b8 01 00 b9 02 00 01 c8
-
```

图 2.44 用 E 命令将机器码写入内存

如何查看我们写入的或内存中原有的机器码所对应的汇编指令呢？可以用 U 命令进行。比如可以用 U 命令将从 1000:0 开始的内存单元中的内容翻译为汇编指令，并显示出来。如图 2.45 所示。

图 2.45 中，首先用 E 命令向从 1000:0 开始的内存单元中写入了 8 个字节的机器码；然后用 D 命令查看内存 1000:0~1000:1f 中的数据(从数据的角度看一下我们写入的内容)；最后用 U 命令查看从 1000:0 开始的内存单元中的机器指令和它们所对应的汇编指令。

U 命令的显示输出分为 3 部分：每一条机器指令的地址、机器指令、机器指令所对应的汇编指令。我们可以看到：

1000:0 处存放的是我们写入的机器码 b8 01 00 所组成的机器指令，对应的汇编指令是 mov ax,1;

1000:3 处存放的是我们写入的机器码 b8 02 00 所组成的机器指令；对应的汇编指令是 mov cx,2;

1000:6 处存放的是我们写入的机器码 01 c8 所组成的机器指令;对应的汇编指令是 add ax,cx;

1000:8 处存放的是内存中的机器码 03 49 42 所组成的机器指令;对应的汇编指令是 add cx,[bx+di+42]。

```
C:\>debug
-e 1000:0 b8 01 00 b9 02 00 01 c8
-d 1000:0 1f
1000:0000  B8 01 00 B9 02 00 01 C8-03 49 42 4D 00 00 00 00 .....IBM.
1000:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-u 1000:0
1000:0000  B80100      MOV     AX,0001
1000:0003  B90200      MOV     CX,0002
1000:0006  01C8        ADD     AX,CX
1000:0008  034942      ADD     CX,[BX+DI+42]
1000:000B  4D          DEC     BP
1000:000C  0000        ADD     [BX+SI],AL
1000:000E  0000        ADD     [BX+SI],AL
1000:0010  0000        ADD     [BX+SI],AL
1000:0012  0000        ADD     [BX+SI],AL
1000:0014  0000        ADD     [BX+SI],AL
1000:0016  0000        ADD     [BX+SI],AL
1000:0018  0000        ADD     [BX+SI],AL
1000:001A  0000        ADD     [BX+SI],AL
1000:001C  0000        ADD     [BX+SI],AL
1000:001E  0000        ADD     [BX+SI],AL
```

图 2.45 用 U 命令将内存单元中的内容翻译为汇编指令显示

由此,我们可以再一次看到内存中的数据 and 代码没有任何区别,关键在于如何解释。

如何执行我们写入的机器指令呢?使用 Debug 的 T 命令可以执行一条或多条指令,简单地使用 T 命令,可以执行 CS:IP 指向的指令,如图 2.46 所示。

```
-e 1000:0 b8 01 00 b9 02 00 01 c8
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
ES=0B39 FS=0B39 SS=0B39 CS=0B39 IP=0100  NO UP EI PL NZ NA PO NC
0B39:0100 40      INC     AX

-r cs
CS=0B39
1000

-r ip
IP=0100
c0

-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
ES=0B39 FS=0B39 SS=0B39 CS=1000 IP=0000  NO UP EI PL NZ NA PO NC
1000:0000 B80100      MOV     AX,0001

-r
AX=0001 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
ES=0B39 FS=0B39 SS=0B39 CS=1000 IP=0003  NO UP EI PL NZ NA PO NC
1000:0003 B90200      MOV     CX,0002
```

图 2.46 使用 T 命令执行 CS:IP 指向的指令

图 2.46 中,首先用 E 命令向从 1000:0 开始的内存单元中写入了 8 个字节的机器码;然后用 R 命令查看 CPU 中寄存器的状态,可以看到,CS=0b39H、IP=0100H,指向内存 0b39:0100;若要用 T 命令控制 CPU 执行我们写到 1000:0 的指令,必须先让 CS:IP 指向 1000:0;接着用 R 命令修改 CS、IP 中的内容,使 CS:IP 指向 1000:0。

完成上面的步骤后, 就可以使用 T 命令来执行我们写入的指令了(此时, CS:IP 指向我们的指令所在的内存单元)。执行 T 命令后, CPU 执行 CS:IP 指向的指令, 则 1000:0 处的指令 b8 01 00(mov ax,0001)得到执行, 指令执行后, Debug 显示输出 CPU 中寄存器的状态。

注意, 指令执行后, AX 中的内容被改写为 1, IP 改变为 IP+3(因为 mov ax,0001 的指令长度为 3 个字节), CS:IP 指向下一条指令。

接着图 2.46, 我们可以继续使用 T 命令执行下面的指令。如图 2.47 所示。

```

AX=0001 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B39 ES=0B39 SS=0B39 CS=1000 IP=0003  NO UP EI PL NZ NA PO NC
1000:0003 B8 02 00      MOV     CX,0002
-t
AX=0001 BX=0000 CX=0002 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B39 ES=0B39 SS=0B39 CS=1000 IP=0006  NO UP EI PL NZ NA PO NC
1000:0006 01 C8      ADD     AX,CX
-t
AX=0003 BX=0000 CX=0002 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B39 ES=0B39 SS=0B39 CS=1000 IP=0008  NO UP EI PL NZ NA PO NC
1000:0008 40      INC     AX

```

图 2.47 用 T 命令继续执行

在图 2.47 中, 用 T 命令继续执行后面的指令, 注意每条指令执行后, CPU 相关寄存器内容的变化。

(8) 用 Debug 的 A 命令以汇编指令的形式在内存中写入机器指令。

前面我们使用 E 命令写入机器指令, 这样做很不方便, 最好能直接以汇编指令的形式写入指令。为此, Debug 提供了 A 命令。A 命令的使用方法如图 2.48 所示。

```

C:\>debug
-a 1000:0
1000:0000 mov ax,1
1000:0003 mov bx,2
1000:0006 mov cx,3
1000:0009 add ax,bx
1000:000B add ax,cx
1000:000D add ax,ax
1000:000F
-d 1000:0 f
1000:0000 B8 01 00 BB 02 00 B9 03 00 01 D8 01 C8 01 C8 00 .....

```

图 2.48 用 A 命令向从 1000:0 开始的内存单元中写入指令

图 2.48 中, 首先用 A 命令, 以汇编语言向从 1000:0 开始的内存单元中写入了几条指令, 然后用 D 命令查看 A 命令的执行结果。可以看到, 在使用 A 命令写入指令时, 我们输入的是汇编指令, Debug 将这些汇编指令翻译为对应的机器指令, 将它们的机器码写入内存。

使用 A 命令写入汇编指令时, 在给出的起始地址后直接按 Enter 键表示操作结束。

如图 2.49 中, 简单地用 A 命令, 从一个预设的地址开始输入指令。

```

C:\>debug
-a
0B39:0100 mov ax,1
0B39:0103 mov bx,2
0B39:0106 mov cx,3
0B39:0109 add ax,bx
0B39:010B add ax,cx
0B39:010D add ax,ax
0B39:010F

```

图 2.49 从一个预设的地址开始输入指令

本次实验中需要用到的命令

查看、修改 CPU 中寄存器的内容：R 命令

查看内存中的内容：D 命令

修改内存中的内容：E 命令(可以写入数据、指令，在内存中，它们实际上没有区别)

将内存中的内容解释为机器指令和对应的汇编指令：U 命令

执行 CS:IP 指向的内存单元处的指令：T 命令

以汇编指令的形式向内存中写入指令：A 命令

在预备知识中，详细讲解了 Debug 的基本功能和用法。在汇编语言的学习中，Debug 是一个经常用到的工具，在学习预备知识中，应该一边看书一边在机器上操作。

前面提到，我们的原则是：以后用到的，以后再说。所以在这里只讲了一些在本次实验中需要用到的命令的相关的使用方法。以后根据需要，我们会讲解其他的用法。

2. 实验任务

(1) 使用 Debug，将下面的程序段写入内存，逐条执行，观察每条指令执行后，CPU 中相关寄存器中内容的变化。

机器码	汇编指令
b8 20 4e	mov ax,4E20H
05 16 14	add ax,1416H
bb 00 20	mov bx,2000H
01 d8	add ax,bx
89 c3	mov bx,ax
01 d8	add ax,bx
b8 1a 00	mov ax,001AH
bb 26 00	mov bx,0026H
00 d8	add al,bl
00 dc	add ah,bl
00 c7	add bh,al
b4 00	mov ah,0
00 d8	add al,bl
04 9c	add al,9CH

提示： 可用 E 命令和 A 命令以两种方式将指令写入内存。注意用 T 命令执行时，CS:IP 的指向。

(2) 将下面 3 条指令写入从 2000:0 开始的内存单元中，利用这 3 条指令计算 2 的 8

次方。

```
mov ax,1  
add ax,ax  
jmp 2000:0003
```

(3) 查看内存中的内容

PC 机主板上的 ROM 中写有一个生产日期,在内存 FFF00H~FFFFFH 的某几个单元中,请找到这个生产日期并试图改变它。

提示: 如果读者对实验的结果不理解,请仔细阅读第 1 章中的 1.15 节。

(4) 向内存从 B8100H 开始的单元中填写数据,如:

```
-e B810: 0000 01 01 02 02 03 03 04 04
```

请读者:先填写不同的数据,观察产生的现象;再改变填写的地址,观察产生的现象。

提示: 如果读者对实验的结果不理解,请仔细阅读第 1 章中的 1.15 节。