21世纪军队院校计算机系列教材

计算机硬件技术基础实验教程

主编 吕红 王凤芹

21世纪军队院校计算机系列教材

计算机硬件技术基础实验教程

主　编　吕　红　王凤芹
副主编　王国卫
主　审　郭天杰

兵器工业出版社

内 容 简 介

本书是与《计算机硬件技术基础》配套使用的实验教材。全书共分四章，第一章介绍汇编语言程序的建立和执行及 DEBUG 的使用；第二章为汇编语言程序设计实验，共安排了 15 个实验，每个实验都给出了实验目的、实验内容和参考流程图；第三章介绍了 TPC－2003 通用 32 位微机接口实验系统的组成结构和安装步骤；第四章为接口部分实验内容，共安排了 11 个实验，每个实验都给出了实验目的、实验内容、连线方法、流程图及参考程序。上机实验共 30～40 学时，教师可根据教学要求对本书实验内容进行取舍。

本书可作为高等院校学生学习微机原理及接口的实验教学用书，也可作为自学者的参考书。

图书在版编目（CIP）数据

计算机硬件技术基础实验教程/吕红，王凤芹主编．
北京：兵器工业出版社，2007.10
（21 世纪军队院校计算机系列教材）
ISBN 978－7－80172－959－0

Ⅰ．计… Ⅱ．①吕…②王… Ⅲ．硬件－军事院校－教材
Ⅳ．TP303

中国版本图书馆 CIP 数据核字（2007）第 152589 号

出版发行：兵器工业出版社
发行电话：010-68962596，68962591
邮　　编：100089
社　　址：北京市海淀区车道沟 10 号
经　　销：各地新华书店
印　　刷：北京市登峰印刷厂
版　　次：2007 年 10 月第 1 版第 1 次印刷
印　　数：1—3750

责任编辑：王　强
封面设计：肖　璐
责任校对：郭　芳
责任印制：赵春云
开　　本：787×1092　1/16
印　　张：6.25
字　　数：150 千字
定　　价：13.50 元

《21 世纪军队院校计算机系列教材》

编 审 委 员 会

序

今天的人类已经进入21世纪，以计算机技术为核心的信息技术取得了日新月异的发展，标志着信息时代已经来临，并不断地改变着人类社会的工作方式、生活方式、学习方式和休闲方式。信息社会的发展使得人类离不开计算机，它已经成为人们工作、生活、学习和休闲的主要工具，而它本身也在不断地发展之中。根据解放军三总部对计算机基础教育的要求，满足新世纪军队高等院校教学改革和人材培养的需求，贯彻中央军委的强军策略，我们组织编写了《21世纪军队院校计算机系列教材》。参加编写的单位有海军工程大学、海军航空工程学院、大连舰艇学院、空军雷达学院、空军后勤学院和空军工程大学导弹学院等，参加编写的人员由长期战斗在教学科研第一线的、具有丰富教学实践经验的部分优秀教师组成。

本系列教材主要包括计算机文化基础、计算机软件技术基础、计算机硬件技术基础和计算机网络应用基础，主要依据解放军三总部下发的计算机基础教育的课程体系和教学大纲的要求，参考国家教委非计算机专业的计算机教育和计算机等级考试的相关要求，进行规划和组织编写的，主要面向军队高等院校本、专科教育教学使用。

为了适应新世纪军队高等院校教育发展的要求，达到培养掌握信息技术的军事人材的目标，本系列教材以培养学生具有较扎实的计算机基础理论知识、较强的计算机实际操作能力、较好的创新思维和较高的综合素质为目的，注重知识的更新和合理的知识结构，注意借鉴和汲取国内外优秀教材的精化，尽力反映最新的教学科研成果和作者的教学实践经验。本系列教材配有相当数量的习题和丰富的实验指南，

我们相信，通过作者们的共同努力，定将使本系列教材成为具有时代特色的、适合军队院校使用的、高质量的系列教材，为军队高等教育事业的发展和高素质军事专业人材的培养做出应有的贡献。

编审委员会

2005年8月

前　言

近十几年以来，微处理器技术和微型计算机技术得到突飞猛进的发展，其应用已经渗透到各行各业的各个领域，深入到科学计算、信息处理、过程控制、仪器仪表、事务管理、计算机辅助设计、制造、家用电器、网络通信服务等方面，极大地改变着人们的工作和生活方式，已经成为社会前进的巨大推动力。因此，学习微型计算机和接口技术已经成为现代科技人员和高等院校各专业学生不可缺少的课程。

本书配合《计算机硬件技术基础》课程，安排了汇编语言和接口程序设计两部分实验，力图很好地配合课堂教学，使学生在实践中通过实际操作、编程和接口实验而掌握计算机硬件知识和技能，有利于学生理论联系实际，有利于培养学生的创新思想和创新能力。

本书由吕红、王凤芹主编，王国卫副主编。其中第一、第二、第三章及附录由海军航空工程学院吕红编写，第四章由海军航空工程学院王凤芹、王国卫编写，海军航空工程学院郭天杰副教授审阅了全书并提出了许多宝贵的意见。

由于编者水平有限，加之时间仓促，尽管我们尽了最大的努力，书中错误和不妥之处仍在所难免，敬请读者批评指正。

编　者

2007 年 9 月

目　　录

第一章　汇编语言程序的建立和执行

第一节　汇编语言运行环境

要运行汇编语言程序，要求建立汇编语言的工作环境，应在磁盘上配置以下软件或文件：

(1) DOS 操作系统

(2) 编辑程序(EDIT. EXE、EDLIN. COM 或其他编辑程序)

(3) 宏汇编程序 MASM. EXE(或 ASM. EXE)

(4) 连接程序 LINK. EXE

(5) 调试程序 DEBUG. COM

其中 ASM. EXE 是普通汇编程序，它不支持宏汇编，如果要用宏汇编，则必须用 MASM. EXE。

第二节　汇编语言程序从建立到执行的过程

一、汇编语言程序从建立到执行的流程

汇编语言程序从建立到执行的流程如图 1.1 所示。

二、汇编语言程序的建立过程

下面以建立和执行用户程序 ABC. EXE 为例来说明上机过程。

(1) 用 EDIT 命令建立汇编语言源程序 ABC. ASM (后缀为 . ASM 的文件)

源程序是用汇编语言的语句编写的程序，它不能被机器识别。源程序必须以 ASM 为扩展文件名。

键入命令：

E:\>EDIT　ABC. ASM ↵(每个命令后面应键入回车键，以下均如此)

进入 EDIT 的编辑环境。输入汇编语言源程序。

(2) 用 MASM 命令产生目标文件[OBJ 文件]

建立源程序以后，接下来要用汇编程序 MASM. EXE 进行汇编。所谓汇编，就是把以 . ASM 为扩展名的源文件转换成用二进制代码表示的目标文件。目标文件以 . OBJ 为扩展名。进行汇编的过程中，汇编程序对源文件进行二次扫描，如果源程序中有语法错误，则汇编不成功，不能生成目标代码文件。汇编程序 MASM 在汇编过程结束后会指出源程序中的错误，这时，用户应该再次运行编辑程序，调源程序进入编辑状态，修改源程序中的错误。反复这个过程，直到最后得到没有语法错误的 OBJ 文件为止。

对 ABC. ASM 的汇编过程如下：

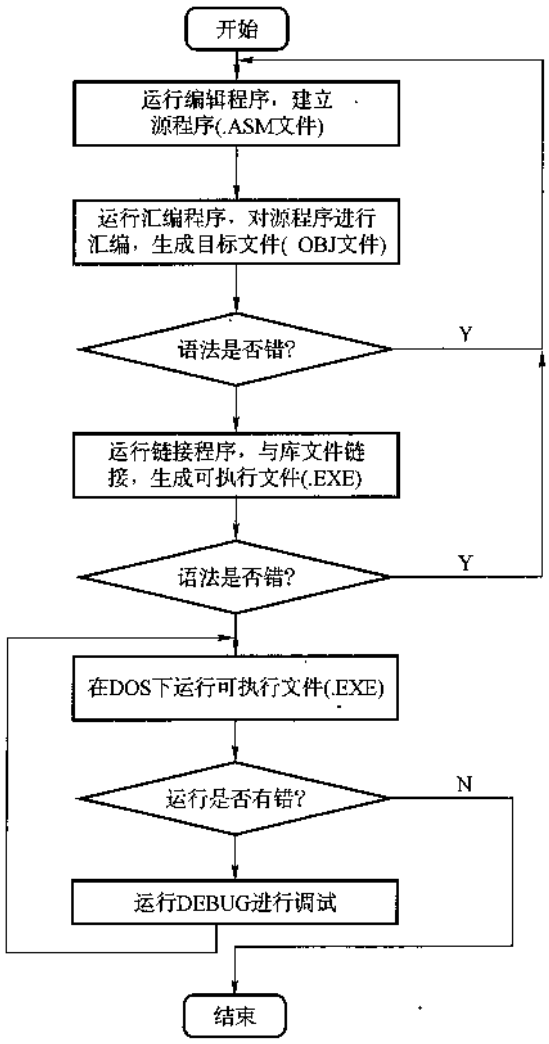

图 1.1　汇编语言程序处理流程

键入命令：

E:\>MASM ABC. ASM ↵

此时，汇编程序给出如下回答：

Microsoft (R) Macro Assembler Version 5. 00

Copyright (C)Microsoft Corp 1989

Object filename [ABC. OBJ]: ↵

Sou rce listing [NUL. LST]: ABC ↵

Crossre －ference [NUL. CRF]: ABC ↵

如果被汇编的程序没有语法错误，则屏幕上还给出如下信息：

Warning　Servers

Errers　Errors

0　0

从上面的操作过程中可以看出，汇编过程有3个步骤，每个步骤有1个输出文件。

(1) 第一个是目标文件。它以OBJ为扩展名，产生OBJ文件是进行汇编操作的主要目的，所以这个文件是一定要产生，也一定会产生的。一般可令目标文件的名字与源程序名字相同，在出现Object filename [ABC.OBJ]:时，方括号中为与源程序同名的默认文件名，这时只要回车就行了。如果要另外命名目标文件名，则输入目标文件名之后再按回车键。

(2) 第二个是列表文件。它以LST为扩展名。列表文件同时给出源程序和机器语言程序，从而，可以使调试变得方便。列表文件是可有可无的，如果不需要，则在屏幕上出现提示信息[NUL.LST]:后回车即可。如果需要，则输入列表文件名然后再按回车键。

(3) 第三个是交叉符号表。此表给出了用户定义的所有符号，对每个符号都列出了将其定义的所在行号和引用的行号，并在定义行号上加上"#"号。同列表文件一样，交叉符号表也是为了便于调试而设置的，对于一些规模较大的程序，交叉符号表为调试工作带来很大方便。当然，交叉符号表也是可有可无的，如果不需要，那么在屏幕上出现提示信息[NUL.CRF]:时，按回车键即可。

汇编过程结束时，屏幕上会给出源程序中的警告性错误[Warning Errors]和严重错误[Servers Errors]的信息，前者指一般性错误，后者指语法性错误。当存在这两类错误时，屏幕上除指出错误个数外，还给出错误信息代号，程序员可以通过查找手册弄清错误的性质。

如果汇编过程中发现有错误，则程序员应该重新用编辑程序修改源程序的语法错误，重新再次进行汇编，最终直到汇编正确通过为止。特别要指出的是，汇编过程只能指出源程序中的语法错误，并不能指出算法错误、功能错误和其他错误。

三、用LINK命令产生执行文件(EXE文件)

在汇编过程中根据源程序产生出二进制目标文件(OBJ文件)。但OBJ文件用的是浮动地址，它不能直接上机执行，所以还必须使用链接程序(LINK)将OBJ文件转换成可执行的EXE文件。LINK命令还可以将某一个目标文件和其他多个模块(这些模块可以是用户编写的，也可以是某个程序库中存在的)链接起来。

具体操作如下(以对ABC.OBJ进行链接为例):

```
E:\>LINK  ABC↵
```

此时，在屏幕上可见到类似如下的信息:

```
IBM 5552 Multistation Linker 2.00(C)Copyright IBM Corp. 1985
Run File [ABC.EXE]:        ↵
List File [NUL.MAP]:       ↵
Libraries [.LIB]           ↵
Warning:NOSTACK Segment
```

对于以上信息，如果不改变文件名，则直接按回车键，如果用户程序要用到库函数，则LINK命令需要一个输入文件，即OBJ文件，此时，对于提示信息Libraries[.LIB]，要输入库名。

LINK过程产生两个输出文件，一个是扩展名为EXE的执行文件，产生可执行文件是LINK过程的主要目的，另一个是扩展名为MAP的列表分配文件，也称它为映像文件，它给出每个段在内存中的分配情况。比如某一个列表分配文件为如下内容:

```
Warning: NO      Stack     Segment
Start     Stop    Length    NameClass
000H      0015H   0016H     CODE
0020H     0045H   0026H     DATA
0050H     0061H   0012H     EXTRA
origin    Group
Program entry Point at 0000: 0000
```

MAP 文件也不是必须有的。

从 LINK 过程的提示信息中，可看到最后给出了一个“NO Stack Segment——无堆栈段”的警告性错误信息，其实，无堆栈段并不一定影响程序的执行，但是如果源程序中设置了堆栈段，则不会出现这样的提示信息。

四、可执行程序的运行

生成可执行的 .EXE 文件后，就可以执行程序了。只要在 DOS 提示符输入文件名并按下回车键即可。例如要执行 ABC 程序，则：

```
E:\ ABC ↵
```

五、程序的调试

一般来说，大部分程序必须经过调试阶段才能纠正程序设计中的错误，从而得到正确的结果。所谓调试阶段，就是用调试程序(DEBUG 程序)发现错误，再经过编辑、汇编、链接，纠正错误。

六、汇编程序上机过程示例

1. 用 EDIT 全屏幕编辑程序进行输入和编辑，产生源程序 AA1. ASM。

(1) 在 E 盘的根目录下进入 EDIT 的操作界面如图 1.2 所示：

```
E:\>edit AA1.ASM
```

图 1.2 进入 EDIT 操作界面

(2) 进入 EDIT 后的操作界面如图 1.3 所示。

(3) 输入程序内容如图 1.4 所示。

(4) 保存之后退出，如图 1.5 所示。

2. 用 MASM 汇编程序进行汇编，产生目标程序 AA1. OBJ。

```
E:\>masm ↵
```

3. 用 LINK 链接程序进行链接，产生 AA1 . EXE 文件。键入链接命令

```
E:\>LINK ↵
```

4. 在 DOS 下执行程序。

```
E:\>AA1 ↵
```

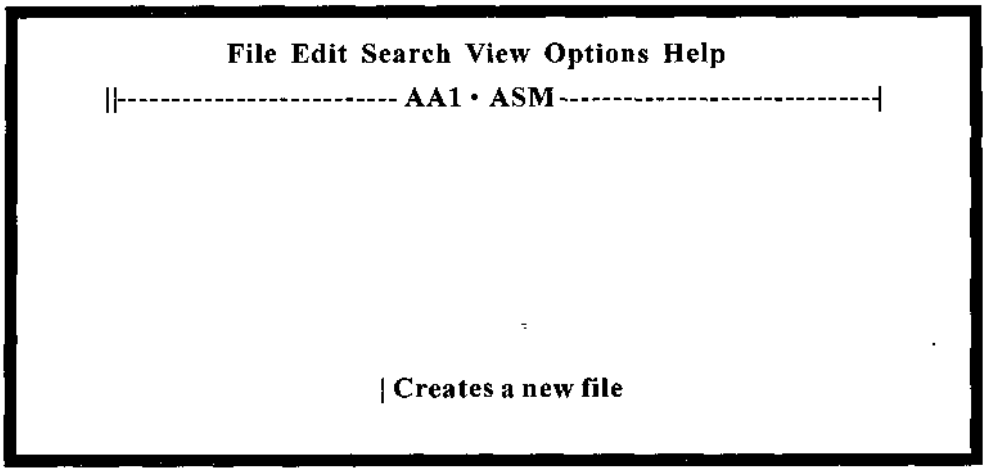

图 1.3　EDIT 操作界面

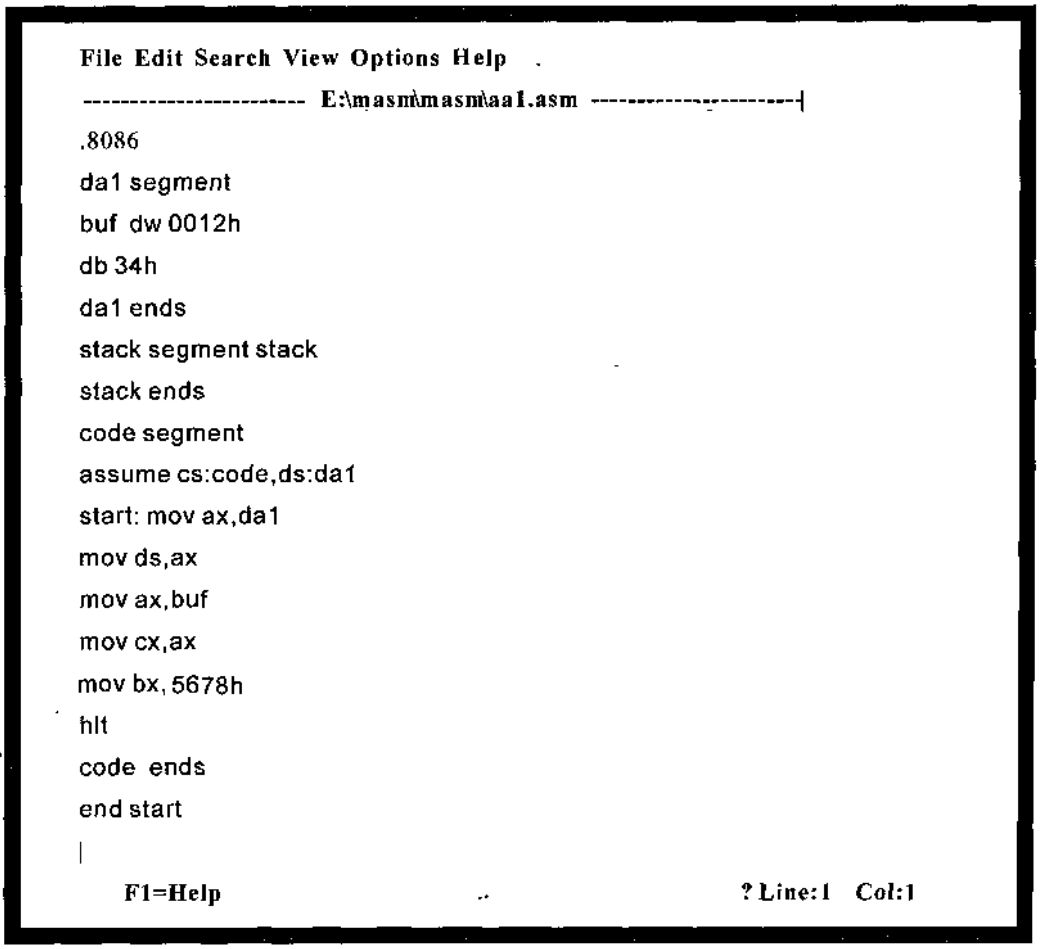

图 1.4　EDIT 编辑界面

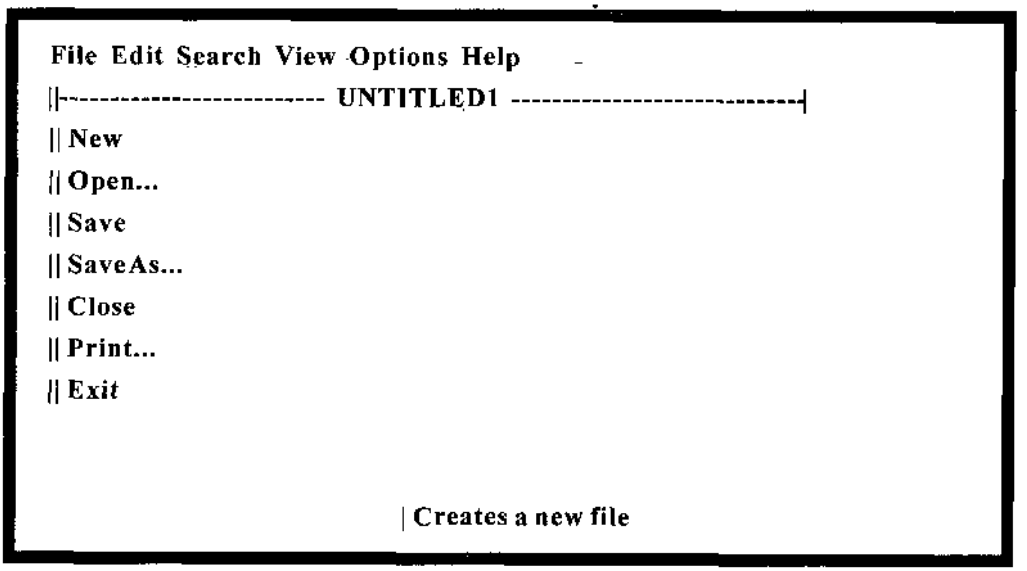

图 1.5 EDIT 保存退出界面

第三节 动态调试程序 DEBUG

对汇编语言程序产生的目标代码进行调试比较困难。动态调试程序 DEBUG 是调试汇编语言程序的一个有力工具，使用 DEBUG 可以方便地调试汇编语言程序。关于 DEBUG 程序中的各种命令，可参阅 DOS 手册。在此给出常用的几个命令和操作方法。

一、DEBUG 的启动

DEBUG 的启动格式是：

DEBUG [路径名,(〈参数〉)]

其中路径名指的是被调试的程序，参数就是该程序所涉及的参数。例如，在 A 盘键入下面的命令：

A>DEBUG MYPROG. EXE

此时，屏幕上出现一个短画线"_"，这是 DEBUG 的提示符，在此提示符下，可以输入 DEBUG 的命令。在这种方式下，系统先将 DEBUG 装入内存，然后找到被调试的程序并从盘上将它装入内存(对于 EXE 文件，DEBUG 将它装入到最低可用的区段中，并从 100H 开始装入)，然后显示出提示符"_"。

如果在键入 DEBUG 时，不键入程序名，例如在 E 盘键入

E> DEBUG

在这种方式下，系统则直接显示提示符"_"，等待用户进一步输入命令。

二、DEBUG 的地址格式

在 DEBUG 命令中使用的地址其格式约定如下：

[<段地址>：]<位移量>

其中<段地址>可以是段寄存器名，也可以是十六进制的值，也可以缺省。例如：

CS:100

4BA:100

而地址范围的格式为：

<段地址:>:<始位移量><末位移量>

或<段地址>:<始位移量> L<长度>

例如：

CS:100 110

4BA:100_10

三、DEBUG 的主要命令

1. 汇编命令 A

汇编命令的格式是

_A [<地址>]

A 命令用 Ctrl+C(^C) 退出。

A 是一条逐行汇编命令，主要用于小段程序的汇编以及修改目标程序。使用 A 命令来汇编小段程序往往比使用汇编、链接程序方便。使用逐行汇编命令汇编程序时，一般不允许使用标号和伪指令。但在 MS—DEBUG 中允许使用 DB 和 DW 这两条伪指令。

汇编中发现错误时，显示出一个"?"并要求重新输入。汇编好的程序可用写盘命令保存到盘上去。

例如：

```
A>DEBUG
_A
08F1:0100   MOV   AL,40
08F1:0102   MOV   CH,24
08F1:0104   JMP   110
08F1:0106   DB'DATA'00
08F1:010C   DW   1234   5678
08F1:0110   MOV   AH,01
08F1:0112   INT   20
08F1:0114   ^C
```

注意，A 命令后面不指出地址时，表示程序的起始地址就是 DEBUG 指针所指向的当前地址。

2. 反汇编命令 U

U 命令可以对二进制代码程序作反汇编，即将机器代码反汇编成源程序，常用于分析和调试目标程序。它的格式如下：

U [<地址范围>]

例如，上例中的程序可用 U 命令反汇编出来。

```
_U 100 104              反汇编出 100 至 104  共 6 个单元的程序代码。
08F1:0100  B040     MOV   AL,40
08F1:0102  B524     MOV   CH,24
08F1:0104  EB0A     JMP   0110
_U 110 112              反汇编出 110 至 112  共 4 个单元的程序代码。
08F1:0110   B401     MOV   AH,01
08F1:0112   CD20     INT   20
_U                      从当前地址开始反汇编
_U   200                从 CS 为 200 处开始反汇编
```

3. 运行命令 G

G 命令用来运行一个程序或一段程序，它的格式如下：

G [=<始址>] [<断点>]

其中<始址>是开始执行程序的地址，<断点>是程序执行的中断处 。最多允许设置 10 个断点。如果 G 命令中不带参数，则从头运行装入的程序，运行后仍返回 DEBUG。如 G 命令带有断点参数，则程序执行到断点地址时暂停并显示出各寄存器状态。

例如，要从地址 100 处开始运行到 110 处暂时中断，然后显示中断处的寄存器的内容，就键入如下命令：

G==100 110

屏幕上将会出现如下内容，包括各寄存器的内容和标志寄存器中主要标志的状态，在最后一行还给出下一条将要执行的指令的地址、机器语言和汇编语言，程序员可以从显示出来的寄存器内容了解程序运行是否正确：

AX=1240 BX=0000 CX=2400 DX==0000 SP=-FFEE BP=0000
SI=0000 DI=0000

DS=08F1 ES=08F1 SS=08F1 CS=08F1 IP=0102 NV UP DI PL
NZ NA PO NC

08F1:0110 B401   MOV AH,01

例如，想把断点设置在 0120H 处，则可键入命令：

_G 120

此时，程序在 0120H 处停下，并显示出该断点处所有寄存器以及各标志位的当前值和断点处对应的汇编程序语句。

4. 跟踪命令 T

要确定错误到底由哪条指令的执行所引起，就要用到跟踪命令。T 命令用来逐条跟踪程序，跟踪命令也叫单步执行命令，此命令使程序每执行一条指令便给出所有寄存器的内容。

它的格式是：

T [=<地址>] [<跟踪条数>]

例如，要从 100 处开始执行两条指令，键入如下命令：

_T==100 2

则屏幕出现下面的内容：

AX=0140 BX=0000 CX=2400 DX=0000 SP=FFEE BP=0000 SI=0000

```
DI=0000
  DS=08F1  ES=08F1  SS=08F1  CS=08F1  IP=0102  NV  UP  DI  PL  NZ
NA  PO  NC
  08F1:0102  B524      MOV  CH,24
  AX=0140  BX=0000  CX=2400  DX=0000  SP=FFEE  BP=0000  SI=0000
DI=0000
  DS=08F1  ES=08F1  SS=08F1  CS=08F1  IP=0104  NV  UP  DI  PL  NZ
NA  PO  NC
  08F1:0104  EB0A      JMP'     0110
```

要从地址 200 处开始执行，共执行 4 条指令，则键入命令：

```
_T= 200  4
```

如键入：

```
_T 3
```

则从当前地址往下跟踪执行 3 条指令后停止。

5. 显示内存命令 D

对于某些程序段，仅仅在断点处观察寄存器的内容还不能了解到程序运行的结果，而需要了解数据段的内容，即内存单元的内容，此时可用 D 命令。D 命令按照十六进制的形式以及对应的 ASCII 字符形式显示内存内容。命令的格式是：

```
_D[ 地址 ]      或    _D[ 寄存器 ]
```

例如：

```
_DCS:106 10F
```

则显示如下内容：

```
08F1:0106  44 41_54 41 20 00 34 12 78 56      DATA  4  xV
```

又如：

```
_D 8F1:106 10F
```

则显示如下内容：

```
08F1:0106  44 41_54 41 20 00 34 12 78 56      DATA  4  xV
```

6. 修改内存命令 E

E 命令用来修改内存，它的格式为：

```
E<地址 [<字节串>]
```

其中，字节串为一系列用空格隔开的十六进制字节，或者是用单引号括起来的字符串。

例如：

```
_E DS:100  F3'XYZ' 8D
```

其中，F3、X、Y、Z、8D 各占一个字节，E 命令用这 5 个字节代替存储单元 DS:0100 到 0104 共 5 个存储单元的内容。

又如：先用 E 命令修改内存，再用 D 命令检查：

```
_E 112  00 'END OF PROGRAM'
_D 112  120
08F1:0112 00 45 4E 44  20 4F_46 20 50 52 4F 47 52 41   .END OF PROGRAM
08F1:0120 4D
```