微型计算机原理实验指导

江琪 编

人民邮电出版社

微型计算机原理
实验指导

江琪 编

人民邮电出版社

登记证号(京)143号

内 容 简 介

本书是《微型计算机原理》配套教材。以8086CPU为基础，介绍了汇编语言程序设计的上机过程，DEBUG调试程序的使用方法。书中安排了程序设计实验、接口芯片实验和综合练习。全书内容充实，由浅入深，可作为中专学校非计算机类专业的微型计算机原理课程的实验教材，有关大专院校、培训班的实验用书，亦可作为工程技术人员的参考用书。

微型计算机原理
实验指导
江琪 编
责任编辑 马月梅
*
人民邮电出版社出版发行
北京朝阳门内南竹杆胡同111号
北京朝阳展望印刷厂印刷
新华书店总店科技发行所经销
*
开本：787×1092 1/16 1995年2月 第一版
印张：4.75 页数：38 1995年5月 北京第2次印刷
字数：110千字 印数：8 001-16 000册
ISBN 7-115-05527-0/TP·153
定价：5.00元

前　言

随着邮电通信事业飞速发展，新技术、新理论、新装备日新月异。我司原组织编写的中专教材有些内容显得陈旧，难于适应新形势下教学的需要，为此我们对教学大纲进行了修订，并对原教材出版计划做了调整，重点突出了新技术方面的教材。今后将陆续出版。

教材是提高教学质量的关键。编写教材时力求以马列主义、毛泽东思想为指导，运用辩证唯物主义的观点阐明科学技术的规律，内容力求结合实际，提高学生的实践动手能力。

对于书中的缺点和错误之处，希望教师和同学们在使用过程中及时指出，以便修改提高。

邮电部教育司

1994年1月

编者的话

目前，国内大多数的工科院校都开设了微型计算机原理课程，并逐渐从8位微机向16位微机发展。开设16位微机原理课程的难点之一就是实验、实习的实现，为此，我们在经过两轮实际教学的基础上，根据邮电部教育司1994年3月颁发的中专“微型计算机原理教学大纲”和“微机原理课程设计大纲”编写了这本“实验指导”。它和邮电部教育司组织编写的《微型计算机原理》教材相配套。

本“实验指导”共有五章内容和五个附录。第一章和第二章共七个实验，加上汇编语言程序上机过程、DEBUG程序的使用，约需18学时。这部分内容对应于《微型计算机原理》教材第五章和第六章的内容。本“实验指导”第三章和第四章中的接口实验板工作原理及六个接口实验，约需14学时，它对应于《微型计算机原理》教材第七章和第八章的内容。第五章的四个题目难度较大，是为“微机原理”课程设计而安排的，共需72学时(两周)。

本书的第三章介绍了北京市电信学校研制的“十六位微机接口实验板”的工作原理，该接口实验板已由邮电部教育司向其所属各中专学校推荐使用。由于篇幅所限，本书没有对该产品进行更加详细的介绍。

本书的编写过程中得到了邮电部教育司杨荣老师和江苏省邮电学校高级讲师唐瑞庭老师的大力支持，我的同事赵莉军老师、刘刚老师和刘葳老师在文稿整理和实验题目的上机通过等方面作了很多工作，在此向他们表示感谢。

由于时间仓促，加之作者水平有限，书中可能会有错误和不妥之处，敬请广大师生批评指正。

编 者

1994.7

目　录

第一章　汇编语言程序的上机过程

汇编语言程序的上机过程与编译语言程序的上机过程相类似，必须经过以下几个步骤：

1. 用行编辑程序 EDLIN 或全屏幕编辑程序 WORDSTAR 等建立和修改以 .ASM 为扩展名的源程序文件。

2. 用宏汇编程序 MASM(或基本汇编程序 ASM.EXE)对源程序文件进行汇编，生成以.OBJ 为扩展名的目标码文件。

3. 用连接程序 LINK 对目标文件进行连接，形成一个可执行的浮动代码文件，文件的扩展名为 .EXE。

经过以上步骤，在盘上已经建立了可执行文件，则可在 DOS 的提示符下，直接键入文件名(可不要扩展名)，就可把执行文件从盘上装入内存，并立即执行此程序。然而，通常一个较复杂、较长的汇编语言源程序一次通过的可能性很小，这样，就需反复执行以上步骤，对汇编语言源程序进行修改，直至程序运行正确为止。汇编语言上机过程如图 1－1 所示：

另外，DOS 还为我们提供了一个用于动态调试程序的 DEBUG 调试程序。用它可以单步执行程序、断点执行程序，可以显示修改 CPU 内部寄存器的内容、标志位的内容以及存储单元的内容等，这样就便于查找程序中的错误了。

可见，要想建立、汇编、连接和调试运行汇编语言源程序，应该建立一张系统盘，在这张盘上除了要具备启动 DOS(或 CCDOS)所必需的系统文件之外，至少还需有如下文件：

ws.exe 或 edlin.exe

masm.exe 或 asm.exe

link.exe

cref.exe

debug.com

以下，我们都默认把此系统盘插在 A 驱动器中用于启动机器，而在 B 驱动器中插入一张格式化好的数据盘。这样以后所有的操作都具有以下形式：

a>b:　　<CR>

b>a:masm　　<CR>

或 b>a:link　　<CR>

或 b>a:debug　　<CR>

下面将逐节介绍以上内容。

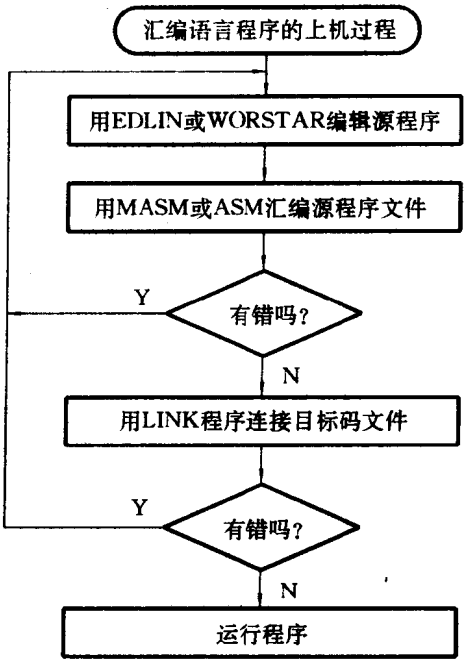

图 1－1　汇编语言上机过程示意图

§1.1 WORDSTAR 全屏幕编辑程序

调用 EDLIN 行编辑程序和 WORDSTAR 全屏幕编辑程序都可完成汇编语言源程序的建立和修改。但是，EDLIN 程序以行方式操作，比较麻烦，而且显示不直观。WORDSTAR 程序以全屏幕方式编辑源程序，显示直观，操作方便，故一般均采用 WORDSTAR 来编辑源程序。

WORDSTAR 是一个文字处理软件，它具有很丰富的功能，而在此我们只介绍与编辑程序有关的一些命令。

一、WORDSTAR 的启动

将系统盘(带有 WS.EXE 文件)插入到 A 驱动器，开机启动系统，然后键入：

A>B:

B>A:WS

一会，在屏幕上出现起始命令表，显示如下：

《起始命令》

D	进入编辑	E	更换文件名
P	打印文件/中断	O	拷贝文件
R	运行程序	Y	删除文件
N	编辑非文书文件	X	退出

这时，键入 N，完成选择编辑非文书文件功能。然后屏幕显示提示：文件名字？再输入要编辑的文件名即可，如 EXAM.ASM。如果 B 盘上已经存在了 EXAM.ASM 文件，则屏幕显示出该文件的内容。如果这是一个新文件，则屏幕显示"NEW FILE"，然后显示下列信息：

B:EXAM.ASM FC=1 FL=1 列 01 INSERT ON

<

<

<

显示的最上一行是状态行，它指出了：

(1) 当前正编辑着的文件名字为:EXAM.ASM。

(2) FC 显示了光标前有多少个字符。

(3) 当前光标所在的行号和列号，FL 为行号。

(4) 当前是否处于插入状态 (INSERT ON)。

状态行下面是程序编辑区，它相当于一张白纸，你可以在此区里一行一行地输入你的源程序。每输入一个字符，光标右移一个字符位置。按下回车键，则结束这一行，光标自动换到下一行，一屏写满后，所有行自动上滚，出现一行新的空白行。在输入过程中，你可以随意移动光标到任一页，任一行的任意一列，进行任意地删除、插入和修改。在程序编辑过程中，可随时存盘。

二、编辑命令

WORDSTAR 提供了丰富的编辑命令，在此我们仅介绍最主要的、常用的几种命令。

1. 帮助命令

在 WORDSTAR 操作过程中，如果某一时刻你忘记了某些命令，则可以通过按下 ^ J(表

示同时按下 CTRL 键和 J 键)得到系统提供的提示信息。提示信息格式如下：

《提示帮助》

F	文件操作	D	编辑
V	字符串和字块操作	B	排版
P	打印字体控制	I	命令索引

这样你就可以通过按下不同的字母键得到相应的操作提示信息。

2. *光标移动命令*

有多种光标移动命令，分别介绍如下：

↑：光标上移一行。

↓：光标下移一行。

←：光标左移一个 ASCII 码位置。

→：光标右移一个 ASCII 码位置。

PgUp：屏幕向前翻一页。

PgDn：屏幕向后翻一页。

Home：移动光标到文件头。

End：移动光标到文件尾。

^ W：光标不动而整个屏幕向下移一行。

^ Z：光标不动而整个屏幕向上移一行。

^ A：光标左移一句(或一个英文单词)。

^ F：光标右移一句(或一个英文单词)。

3. *删除与插入命令*

[Del]键：删除光标左边的字符。

^ G：删除光标处的字符。

^ T：删除本行中光标右边的所有字符。

^ Y：删除光标所在行的所有字符。

[Ins]键：变换插入/非插入状态。

^ N：在光标处加一行。

4. *字块操作命令*

(1) 本文件内部的字块操作

在对字块操作之前，一定要把光标分别移动到字块前、后，并加以标记，命令如下：

^ KB：在字块前加块首标记，成功后显示<B>。

^ KK：在字块后加块尾标记，成功后显示<K>。

^ KH：删除字块首尾标记。

在对字块加注了标记后，就可以对字块进行操作了。操作前，先将光标移到预想位置，命令如下：

^ KV：将加有首尾标记的字块移到当前光标位置。

^ KC：将加有首尾标记的字块拷贝到当前光标位置，原字块仍存在。

^ KY：删除加有首尾标记的字块(不用预先移动光标)。

(2) 文件间内容交换的命令

^ KW：把当前文件中加有首尾标记的字块写入指定文件。

^KR：将指定文件内容拷贝到当前文件中光标所在位置。

5. 退出编辑状态命令

退出编辑状态有以下几种命令形式：

[F1]键：将当前正编辑的文件存到磁盘上，然后退出编辑状态，返回到“起始命令表”。

[F2]键：放弃当前编辑的文件(即不存盘)，退出编辑状态，返回“起始命令表”。退出前，屏幕提问是否放弃当前编辑着的文本文件(Y/N)，以防按错键，造成编辑文件的丢失。

§1.2 MASM.EXE 宏汇编程序

经过 WS 程序编辑存盘的汇编语言源程序(扩展名为.ASM)，要在机器上运行必须先由宏汇编 MASM 或基本汇编程序 ASM 汇编成机器码的目标程序。

ASM 是 MASM 的一个功能子集，它不支持有关结构、记录、宏、条件汇编等伪指令和伪操作符，故在此我们仅介绍 MASM 的用法。在 DOS 状态下，键入 MASM 以后，屏幕显示及操作如下：

```
B>A:MASM <CR>
Microsoft (R)Macro Assembler version 4.00
Copyright (C) Microsoft Corp 1981,1983,1984,1985. All rights reserved.
Source filename [.ASM]: EXAM
object filename [EXAM.OBJ]:
source listing [NUL.LST]:
cross-reference [NUL.CRF]:

  50942 Bytes symbol space free

      0 Warning Errors
      0 Severe Errors
```

可见，在键入 MASM <CR>以后，屏幕先显示版本号，然后出现第一个提示，询问要汇编的文件名；用户输入文件名(扩展名.ASM，但输入时可省略)后，又出现第二个提示，询问目标文件名，若同意括号中的缺省文件名，则按回车键即可，否则重新输入目标文件名，再回车；接着出现第三个提示，询问是否要建立列表文件，括号中的内容表示是否需要，若需要，则输入列表文件名(缺省扩展名为.LST)再回车，否则直接回车即可。最后询问是否要建立交叉索引文件，回答方法类似于第三个询问。四个问题一旦回答完，MASM 立即开始对源程序进行汇编。MASM 对一个给定的源程序进行汇编是一个两遍扫描过程，每遍扫描过程都以遇到 END 指令而结束。第一遍扫描的目的是检查名字以产生一个符号表，第二遍扫描的目的是产生机器码。

正确汇编以后，输出的文件有三个，第一个是目标文件，它以 OBJ 为扩展名，产生 OBJ 文件是我们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。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清单和机器语言程序清单，从而，可以使调试变得方便。列表文件是 ASCII 文本文件，可用 DOS 的 Type 命令把它显示出来。列表文件是可有可无的。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指令所在行号和引用的指令行号，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。交叉索引文件按字母顺序列出程序中使用的符号、标号和变量名，名字后面给出定义它和引用它的行号。为把交叉索引文件打印出来，还要先用 CREF 程序把它变成可打印文件。例如：

```
B>A:cref
cref filename [.CRF]:EXAM
list filename [EXAM.REF]
B>type exam.ref
```

汇编完成以后，若在汇编过程中发现源程序中有语法错误，则指出错误的语句和错误的性质。语法错误分为警告错误(Warning Errors)和严重错误(Severe Errors)两种。Warning Errors 指出汇编程序所认为的一般性错误，Severe Errors 则指出汇编程序认为已使汇编程序无法进行正确汇编的错误。

如果你的源程序有错，则应重新调用 WS 编辑程序修改错误，并重新汇编，直到正确为止。当然，汇编程序只能指出程序中的语法错误，至于程序的算法或编制程序中的其它错误，则应在程序调试时去解决。

§1.3 LINK.EXE 连接程序

汇编后产生的二进制的目标文件(.OBJ 文件)是浮动地址，它不能直接上机运行，必须经过 LINK.EXE 连接程序连接之后才能成为可执行文件。另外，LINK 程序除了生成一个以 .EXE为扩展名的可执行的浮动代码文件以外，根据选择还可生成一个 .MAP 内存分配图文件，这个文件提供了内存地址分配的有关信息。操作过程如下：

```
B>A:LINK
Microsoft (R)Macro Assembler version 4.00
Copyright (C) Microsoft Corp 1981, 1983, 1984, 1985. All rights reserved.
Object Modules [.OBJ]: EXAM
Run File [EXAM.EXE]:
List File [NUL.MAP]:
Libraries [.Lib]:
```

可见，LINK 程序在显示了版本号以后，首先询问要连接的目标文件名，此时应输入文件名(扩展名应为.OBJ，可省略)作为回答。第二个提示询问要产生的可执行文件的文件名，一般直接回车表示采用括号内规定的隐含文件名。第三个提示询问是否建立内存分配图文件，输入文件名(扩展名一般缺省，隐含为.MAP)再回车表示要建立，否则直接回车表示不要建立。最后提示询问是否用到库文件，此时直接按回车即可。四个问题一旦回答完毕，LINK 立即开始工作。若连接过程有错则显示错误信息，需重新调用 WS 文件编辑程序，修改源程序，然后重新汇编、连接，直至无错。连接以后产生的 . MAP 文件提供了内存地址分配的有关信息。

§1.4 程序的执行

连接以后建立的可执行文件(扩展名是.EXE)，可以在 DOS 下直接输入文件名(不必输入

扩展名)而得以执行,操作如下:

B>EXAM <CR>

当输入 EXAM 文件名执行程序时,EXAM.EXE 文件被读入内存并立即执行。实际上,每次执行程序时,随着执行文件装入内存的还有一个程序段前缀 PSP,在内存中它被安装在执行文件的前面。PSP 主要用于向执行文件传递一些参数并提供程序结束正常或异常返回 DOS 的途径。

在 PSP 的最开始处放着 INT 20H 指令的机器码(CDH,20H)。INT 20H 是程序非驻留时常用的结束指令,但发生 INT 20H 软中断指令时,一定要保证 CS 中存放的是 PSP 的段地址。解决的办法是先把 PSP 段地址压入堆栈保存,程序结束时再从堆栈中取回置入 CS。而当 DOS 刚把控制权交给用户程序时,DS、ES 段寄存器中的内容正好是 PSP 的段地址。所以 IBM 建议使用以下的程序结构:

```
code segment
    assume cs:code
begin proc far
    push ds
    sub ax,ax
    push ax
    .......
    ret
begin endp
code ends
    end begin
```

上面程序的一开始把一个 PSP 段地址和一个全 0 字压入堆栈,而过程又定义为远过程,因此 RET 是一个远程(段间)返回指令。于是程序最后执行 RET 指令时,使 CS=PSP 段地址,IP=0000H。这样就自动转到 PSP 最开始处,而正确地执行"INT 20H"指令,完成程序的结束退出。

§1.5 DEBUG.EXE 调试程序

大部分的程序必须经过调试才能纠正程序设计中的错误,从而得到正确的结果。DEBUG 是专为汇编语言设计的一种调试工具,是汇编语言程序员必须掌握的调试手段。启动 DEBUG 程序的过程如下:

B>A:DEBUG <CR>

DEBUG 程序调入后出现提示符"-",以后便可以使用 DEBUG 的所有命令了。

1. 显示内存单元内容命令 D

格式为:-D 地址 或

-D 范围

例如:按指定范围显示存储单元内容的方法为:

```
-D 100 126
18E4:0100  E9 84 3D 43 6F 6E 76 65-72 74 65 64 00  00 00 00  ..=Converted....
```

```
18E4:0110  4D 5A 6F 01 1F 00 03 00-20 00 00 00 FF FF 62 03  MZo..... .....b.
18E4:0120  6A 01 81 5E 00 01 28                             j..^..(
```

其中0100H至0126H是DEBUG显示的单元内容。左边用十六进制表示每个字节，第8、9字节间有一连接字符"—"；右边用ASCII码字符表示每个字节，"."表示不可显示字符。这里没有指定段地址，D命令自动显示DS段的内容。如果D命令后面只给出一个地址，则显示从此地址开始的80个字节的内容。如果没有指定地址，则接着上一个D命令显示以后的80个字节的内容。

2. 修改存储单元内容的命令E

它有两种基本格式：

(1) 用命令给定的内容表去代替指定范围的内存单元的内容，命令格式为：

—E 地址 内容表

例如：

—E DS:120 00 11 'ABC' 22

内存单元DS:120到DS:125这6个单元的内容由表中给定的6个字节内容(其中3个字节用十六进制数表示，即00,11,22；另三个用字符表示，则把它们的ASCII码值代入)所代替。

(2) 逐个单元相继修改，命令格式如下：

—E 地址

例如：

—E 100

在输入了上述命令后，屏幕上显示100H单元的地址和原有的内容：

18E4:0100 12._

如果需要把该单元的内容修改为78，则可以直接输入78，然后再按空格键，则完成修改。并显示下一个单元的地址和原有的内容。显示如下：

18E4:0100 12.78 1B._

这样，可以不断修改相继单元的内容直到用回车键结束该命令为止。

3. 检查和修改寄存器的命令R

它有三种命令格式：

(1) 显示CPU内所有寄存器内容和标志位状态，其格式为：

—R

显示格式如下：

```
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=3F65  ES=3F65  SS=3F65  CS=3F65  IP=0100     NV UP EI PL NZ NA PO NC
3F65:0100 3E              DS:
3F65:0101 B702            MOV       BH,02
```

(2) 显示和修改某个寄存器内容，其格式为：

—R 寄存器名

例如，输入

—R AX

系统将显示如下：

AX 0000

：

即 AX 寄存器的当前内容为 0000H，如不修改其内容则按回车键，否则，可输入欲修改的内容。

(3) 显示和修改标志位状态 R

其格式为：

－RF

系统将显示如下：

NV UP EI PL NZ NA PO NC-

此时，如不修改其内容可按回车键，否则，可键入欲修改的一个或多个标志位内容。例如：

NV UP EI PL NZ NA PO NC－CY DI NG ZR　<CR>

可见，键入的顺序可以是任意的。

8 个标志位的操作如表 1－1 所示。

表 1－1　　标志位符号表

标志位名		置位	复位
溢出	Overflow	OV	NV
方向	Direction	DN	UP
中断	Interrupt	EI	DI
符号	Sign	NG	PL
零	Zero	ZR	NZ
辅助进位	Auxiliary Carry	AC	NA
奇偶	Parity	PE	PO
进位	Carry	CY	NC

4. 运行命令 G

为了检查程序运行是否正确，希望在运行中能设置断点以便逐段调试程序。

G 命令的格式为：

G [＝地址 1]〔地址 2 [地址 3...]〕

其中地址 1 指定运行的起始地址，如省略，则从当前的 CS:IP 处开始运行。地址 2、地址 3...均为断点地址，DEBUG 程序最多允许设置 10 个断点。它用中断类型 3 指令（操作码 CCH）来代替被调试程序在断点地址处的指令操作码。当程序执行到断点时，就停止执行并显示当前所有寄存器及标志位的内容和下一条将要执行的指令，且全部断点被取消，断点处的指令码被恢复（若程序未遇到断点，则不恢复）。

注意，在命令格式中，各地址之间用空格隔开。

5. 跟踪命令 T

跟踪命令有两种命令格式：

(1) 逐条指令跟踪，其格式如下：

－T [＝地址]

程序执行一条指定地址处的指令后停下来，显示 CPU 当前所有寄存器及标志位的值以及下一条指令的地址及内容。若在指令中没有指定地址，则从 CS:IP 的现行地址处单步执行

指令。显然反复执行 T 命令便可起到单步执行指令的目的。

(2) 多条指令跟踪命令,其命令格式如下:

—T [=地址] [值]

此命令从指定的地址开始,执行由命令中的值所决定的若干条指令,然后停下来显示 CPU 当前所有寄存器的内容及全部标志位的值。

6. 汇编命令 A

若在调试程序时发现程序中的某一部分要改写,或要增补部分指令时,则可用 A 命令及其它一些命令来输入、汇编、运行、调试这一段程序。另外,调试简单的程序甚至可以直接利用 A 命令来完成。A 命令的格式为:

—A [地址]

它是从指定的地址(缺省时从 CS:IP 处)开始输入汇编语言的语句。A 命令把它们汇编成机器代码相继地存放到从指定地址开始的存储区中。若输入的语句有语法错误,则 DEBUG 显示:

^ ERROR

且重新显示现行的汇编地址等待新的输入。程序输入完毕后,最后一行不输入内容,直接按回车即可。DEBUG 支持 8086 汇编语言语法和 MASM 宏汇编语言。但是程序所支持的语法稍有不同。简述如下:

① 所有输入的数字值全为十六进制数,每个数字后面不能使用 H。

② 前缀助记符必须在相关的指令之前输入。

③ 支持两个最常用的 DB、DW 伪指令,用来把字和字节的值直接送入相应的存储单元。

7. 反汇编命令 U

若内存某一区域中,已经有了某一个程序的目标码,为了能清楚了解此程序的内容,希望能把目标程序反汇编为源程序,这就需要用到 U 命令。命令格式如下:

—U [范围]

在这里范围是可选项,若没有此项,则以上一条 U 命令的最后一条地址的下一单元作为起始地址开始反汇编 32 个字节。若没有使用过 U 命令,则从当前段的 100H 地址单元开始反汇编 32 个字节。若范围可选项中仅有一个地址,则从这个地址开始反汇编 32 个字节。若范围可选项中既有起始地址又有结束地址则反汇编此范围中的目标码。

8. 命名命令 N

不管是执行读盘命令 R 和写盘命令 W,一般都先要给文件命名,命令格式如下:

—N 文件名

这里,文件名可以有驱动器名和扩展名。例如:

—N A:EXAM.COM

9. 读盘命令 L

L 命令完成把指定文件读入到内存中的操作。若命令中规定了地址,则读入到指定区域中。否则,读入到 CS:0100 开始的内存区域中。但是若具有扩展名.COM 的文件,则始终是读入到 CS:0100 开始的区域中,而把具有.EXE 扩展名的文件读入到 CS:0000 开始的区域中。命令格式如下:

—L [地址]

在 BX 和 CX 中包含着读入文件的字节数,其中 BX 中放的是高位数,CX 中放的是低位

数。

10. 写盘命令 W

W 命令完成将内存区域中的程序和数据存盘的操作。命令格式如下：

－W[地址]

在 W 命令执行之前，必须由 N 命令指定文件名，由 BX 和 CX 指定要写入文件的字节数，其中 BX 中放的是高位数，CX 中放的是低位数。命令中若没有指定地址，则内存从 CS:100H 开始，否则从指定地址开始。

11. 退出命令 Q

当把程序调试完毕后，使用 Q 命令退出 DEBUG，而返回到 DOS。命令格式如下：

－Q

但是，Q 命令并不执行存盘操作。故退出前，如需存盘的话，必须先使用 W 命令存盘后再退出。

第二章　汇编语言程序设计实验

在这一章里，我们将要求学生掌握汇编语言程序设计的基本方法。掌握 DEBUG 调试程序的使用和汇编语言程序设计的上机过程；掌握分支程序结构、循环程序结构和子程序结构的编程方法。

实验一　DEBUG 调试程序的使用

实验目的

(1) 掌握 DEBUG 调试程序的使用方法。
(2) 掌握数据传送指令。

实验内容

利用 DEBUG 程序调试下面的程序段。

```
MOV AX,2034H
MOV BH,AH
MOV CL,AL
MOV SI,AX
MOV AL,11H
MOV [SI],AL
MOV BL,00H
MOV AL,CL
XLAT
XCHG AL,CL
MOV SI,0034H
MOV AL,[BX+SI]
```

实验准备

(1) 仔细阅读本书第一章，熟记 DEBUG 程序各种命令的使用方法。
(2) 阅读教材，熟练掌握数据传送指令。
(3) 预先阅读给出的程序段，预先回答实验步骤(3)、(5)中的问题。

实验步骤

(1) 利用 DEBUG 调试程序的 A 命令将上面的程序段输入到偏移地址为 0100H 处(段地址由 DOS 指定)。