中国电子教育学会高教分会推荐

高等学校新工科人才培养“十三五”规划教材

微机原理与系统设计综合实践教程

侯 叶 林 华 张菊香 吴 涛 编著

西安电子科技大学出版社
http://www.xduph.com

中国电子教育学会高教分会推荐

高等学校新工科人才培养“十三五”规划教材

微机原理与系统设计综合实践教程

侯叶　林华　张菊香　吴涛　编著

西安电子科技大学出版社

内 容 简 介

本书共分 5 章，内容分别为汇编语言程序设计实验、微机实验系统介绍、微机接口硬件实验、微机扩展性与综合性实验开发、微机控制系统设计与开发。

本书内容紧密结合理论教学，涵盖了理论教学中主要的知识点，包括汇编语言程序设计、存储器、常用输入输出接口电路(中断控制器 8259、并行接口芯片 8255、定时/计数器 8254、串行接口芯片 8251、模/数转换器 ADC0809、数/模转换器 DAC0832 及 DMA 控制器 8237)等。

本书既包含知识点的基础实验，又涉及知识点的拓展实验，可培养学生的基本技能。本书基于实验箱的功能模块扩展了一些综合性与应用性实验，不仅可提高学生综合运用知识的能力，也便于学生做课程设计时参考。另外，针对一定的工程背景，本书基于实验箱资源进行了控制系统的开发，可提高学生的工程实践能力，激发学生从事科学研究与探索的兴趣，也可供学生课程设计与毕业设计参考。

本书可作为高等学校自动化、电子、电气、计算机等相关专业“微机原理与系统设计”课程的实验及课程设计教材或参考教材。

图书在版编目(CIP)数据

微机原理与系统设计综合实践教程 / 侯叶等编著. —西安：西安电子科技大学出版社，2019.9

ISBN 978-7-5606-5450-8

Ⅰ. ① 微… Ⅱ. ① 侯… Ⅲ. ① 微型计算机—理论—高等学校—教材 ② 微型计算机—接口技术—高等学校—教材 Ⅳ. ① TP36

中国版本图书馆 CIP 数据核字(2019)第 199607 号

策划编辑 刘小莉

责任编辑 王晓莉 阎 彬

出版发行 西安电子科技大学出版社(西安市太白南路 2 号)

电 话 (029)88242885 88201467 邮 编 710071

网 址 www.xduph.com 电子邮箱 xdupfxb001@163.com

经 销 新华书店

印刷单位 陕西天意印务有限责任公司

版 次 2019 年 9 月第 1 版 2019 年 9 月第 1 次印刷

开 本 787 毫米×1092 毫米 1/16 印 张 13.25

字 数 310 千字

印 数 1～3000 册

定 价 30.00 元

ISBN 978-7-5606-5450-8 / TP

XDUP 5752001-1

如有印装问题可调换

此为试读，需要完整PDF请访问：www.ertongbook.com

前　言

“微机原理与系统设计”是一门理论与实际紧密结合、实践性很强的课程。通过实验教学可加深学生对理论知识的理解，进一步掌握微型计算机的基础知识、汇编语言程序的编程、输入输出接口的设计与应用等，加强学生的计算机应用能力、实际动手能力与实验研究能力，以及应用微机技术对相关系统进行设计的能力等。

本书分为 5 章。

第一章为汇编语言程序设计实验，通过实验使学生掌握汇编语言程序的设计与编程。这部分内容可独立于实验系统，在普通 PC 机房完成，也可在实验室完成，使教师的教学与学生的学习不受环境的影响。

第二章为微机实验系统介绍，主要介绍实验系统构成、实验系统的硬件环境和软件环境。

第三章为微机接口硬件实验，是实验系统所提供的硬件实验，包含基础实验和综合实验，不同专业的教学可根据专业的特点选择相关的内容，以帮助学生对所学知识点有更深入的理解，掌握要求的基本技能。

第四章为微机扩展性与综合性实验开发，是基于实验箱资源开发的一些扩展性与综合性的实验。通过开发例程启发学生的思维，培养学生对接口芯片的实际运用和编程能力，使之具备独立进行接口电路设计的能力。也可为学生课程设计提供参考。

第五章为微机控制系统设计与开发，是基于实验箱资源，进一步设计控制系统，开发新设备及新实验，使学生能针对一定的工程背景与应用，综合运用所学知识，设计与实现控制系统功能，从而提高学生的综合设计能力、创新能力及工程实践能力。每一个控制系统开发例程都可作为一个独立的实验开设，不仅为本科生实验课程、课程设计提供参考，也可为学生毕业设计提供借鉴。

本书的第二、三章的编写得到了清华大学科教仪器公司的支持，尤其是陈楠工程师给予了极大的帮助，在此表示由衷的感谢。在第四、五章的编写中，西安电子科技大学自动化专业的学生程坤、朱熙、石展倩、严子龙、陈亮等对实验进行了整理与调试，在此也一并表示感谢！

限于编者的水平，书中难免有疏漏与不妥之处，敬请读者批评指正。

编　者

2019 年 5 月

目　录

第一章　汇编语言程序设计实验

本章主要内容有 DEBUG 程序调试、汇编语言程序设计实验步骤、四种常用程序设计的方法(顺序、循环、分支、子程序调用)、DOS 系统功能调用、字符串指令编程及宏指令编程等。

通过汇编语言程序设计实验，可以进一步理解汇编语言指令系统、寻址方式以及程序的设计方法。实验中要求能够根据问题，制定解决问题的方案，并使用合适的程序设计方法设计程序，然后通过编程与调试对程序的性能进行测试，最后对数据进行有效的处理，对结果进行合理的分析。

1.1　汇编语言程序的编辑、汇编、连接及调试

1.1.1　基础知识

1. 程序设计步骤

汇编语言程序设计的主要步骤如图 1.1-1 所示。

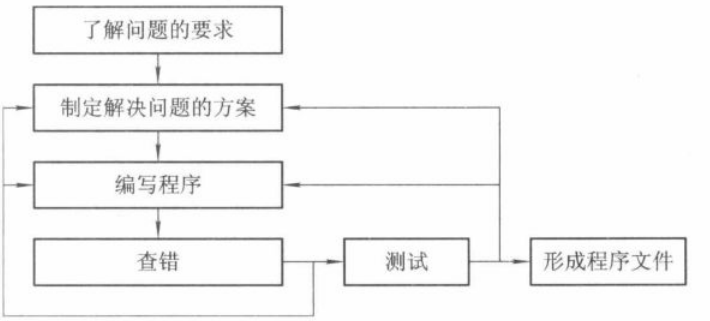

图 1.1-1　程序设计步骤

(1) 了解问题的要求。这是编程的第一步，尤其对于实际工程问题十分重要，它是后续工作的依据。

(2) 制定解决问题的方案。根据要求进行分析，制定解决问题的方案，如确定算法、选用合适的程序设计方法、拟定全过程的流程图等。

(3) 编写程序。把待解决问题及要求分解，用各种汇编语言指令编程实现其功能。编程时需注意以下几点：

- 详细了解所用 CPU 的编程模型、指令系统、寻址方式及有关伪指令；

- 对存储空间和工作单元进行合理分配；
- 合理使用子程序和宏指令；
- 绝对地址和常数尽量用标号或变量代替。

(4) 查错。编制程序后，首先进行书面审查、修正，然后进行编辑、汇编、连接。汇编、连接过程中，机器可能指出程序一些语法上的不正确之处，供进一步修改。此过程要反复多次，最后形成可执行文件。

(5) 测试。依据问题要求，加入各种数据，检验程序能否完全正确运行。若有错，则继续进行修正。

(6) 形成程序文件。程序调试无误，满足设计的要求后，可形成程序文件。程序文件一般包括程序研制报告、程序流程图、程序清单及参数定义说明、内存分配表、程序测试方案及结果说明、程序维护说明等。

2. 汇编语言源程序的编辑、汇编、连接与调试

汇编语言源程序不能直接被计算机识别和执行，必须经过编辑、汇编、连接、调试等一系列系统软件处理。处理关系如图 1.1-2 所示，其中编辑程序、汇编程序、连接程序及 DEBUG 调试程序均是系统程序。

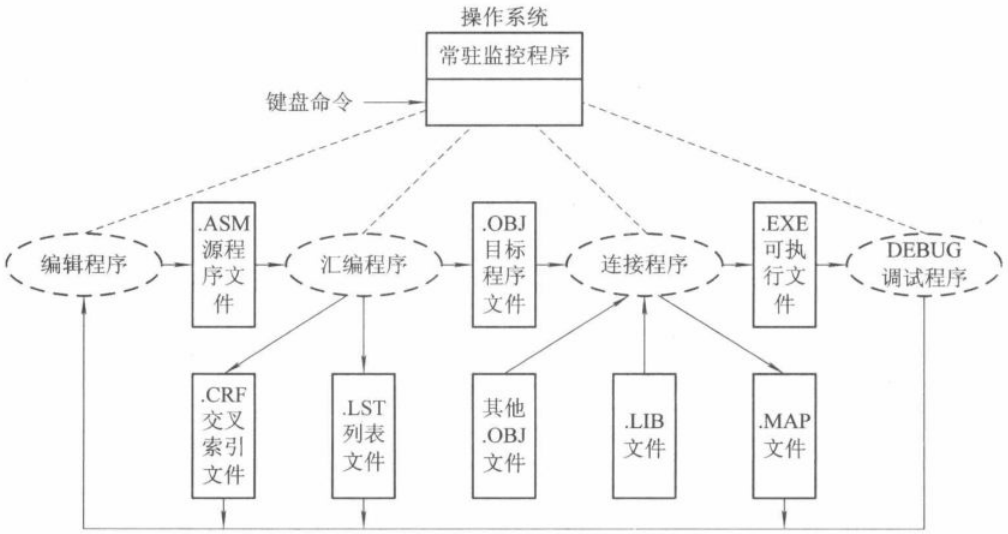

图 1.1-2　编辑、汇编、连接、调试的关系

1) 编辑

利用编辑程序，如 EDLIN、EDIT、记事本等编写源程序。汇编语言源程序名的扩展名为 ASM，例如 myfile.asm。

2) 汇编

由于汇编语言源程序是助记符语言编写的程序，为了使计算机识别，还需将源文件经汇编程序汇编。常用宏汇编程序为 MASM.EXE，其主要功能如下：

(1) 语法检查；

(2) 存储区域分配；

(3) 其他进制数与二进制数的转换、字符与 ASCII 码的转换；

(4) 计算常量表达式值；

(5) 将源程序翻译成目标程序。

源程序经汇编程序汇编后生成三个文件，分别为目标程序文件(.OBJ)、列表文件(.LST)、交叉索引文件(.CRF)。目标程序文件为源程序被汇编后的目标代码文件；列表文件中列出了程序代码、偏移地址以及出错信息，可以方便分页打印、装订；交叉索引文件列出了程序中所定义的所有标识符及其引用情况。

汇编程序 MASM 的使用格式为

```
MASM  Source, Object, List, Crossref
```

其中：

Source：源程序文件名(可不带扩展名)；

Object：目标文件名(可不带扩展名)；

List：列表文件名；

Crossref：交叉索引文件名。

汇编时常用的简略格式如下：

```
MASM  myfile;
MASM  myfile
```

其中，第一种格式表示将汇编源程序 myfile.asm 汇编，只生成 myfile.obj；第二种格式命令末没有分号，按屏幕提示进行操作，可自定义生成的目标代码文件名。

3) 连接

经过汇编程序处理产生的目标程序(.OBJ 文件)已经是二进制文件，但还须经过连接程序的连接才能运行。连接程序的主要工作是：

(1) 找到要连接的所有目标模块；

(2) 对所有要连接的目标模块中的所有段分配地址值，即确定所有段地址值；

(3) 确定所有汇编程序所不能确定的偏移地址值；

(4) 构成装入模块，并把它装入存储器。在模块化程序中，主程序以及多个子程序可以编制成不同的程序模块，各个模块在明确各自功能和相互之间的连接约定以后，就可以独立编写并调试，最后再把它们连接起来形成一个完整的程序。

连接程序要求输入两种文件，即目标程序文件和库程序文件；可以产生两个新文件，即扩展名为.MAP 的内存分配列表文件和扩展名为.EXE 的可执行文件。

连接程序 LINK 的使用格式为

```
LINK Object, Runfile, Mapfile, Liblist
```

其中：

Object：目标文件名；

Runfile：可执行文件名；

Mapfile：内存分配列表文件名；

Liblist：库文件名。

连接时常用的简略格式如下：

```
LINK  myfile;
LINK  myfile
```

此为试读，需要完整PDF请访问：www.ertongbook.com

其中，第一种格式将 myfile.obj 生成可执行文件 myfile.exe；第二种格式命令末没有分号，可按屏幕提示进行操作，可自定义生成的可执行文件名。

4) 调试

对源程序汇编、连接时可以检查出程序的语法及指令用法错误，但不能查出程序功能及逻辑上的错误，也不能发现程序不完善的地方。因此通常还需要通过调试程序来确定程序的正确性。汇编语言程序的调试可以借助于专门的调试工具 DEBUG 来实现。

(1) DEBUG 调试程序的调用。DEBUG 调用的格式：

DEBUG [d:][path][文件名.扩展名]

其中："d:" 为磁盘符号；"path" 表示路径名；调试的文件名是包含扩展名的完整形式。

例如在 myfile.exe 文件所在的路径下，对 myfile.exe 进行调试，可以输入：

DEBUG myfile.exe↙　　;("↙" 表示回车)

此命令执行后进入 DEBUG 状态，并装入要调试的 myfile.exe 程序。

如果在缺省文件名时直接输入 DEBUG 命令，如图 1.1-13 所示，则也可进入 DEBUG 状态，出现提示符 "-"，等待 DEBUG 调试命令。但这个格式只把 DEBUG 程序本身装入内存，并没有把要调试的程序装入内存。

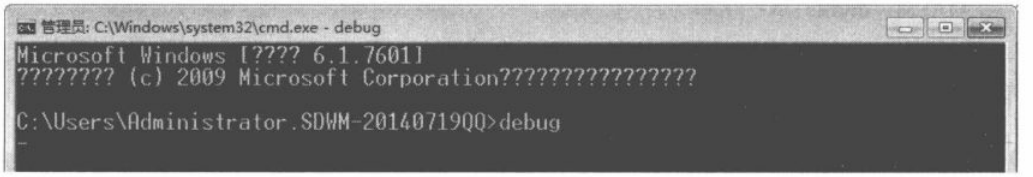

图 1.1-3　运行 DEBUG 的界面

(2) DEBUG 调试命令。DEBUG 提供了强大的调试功能，主要调试命令及功能如表 1.1-1 所示。

表 1.1-1　DEBUG 主要调试命令及功能

命令	功　能	命令	功　能
R	显示、修改寄存器的内容	G	设置断点并分段执行程序
D	显示内存单元的内容	T	单步执行汇编语言指令
E	显示并修改内存单元内容	P	单步执行汇编语言语句
F	填充内存单元内容	A	直接输入汇编语言指令
M	传送内存单元内容	Q	退出DEBUG软件
U	反汇编指令代码		

使用 DEBUG 命令时需注意的主要问题有以下几个：

- 字母不分大小写；
- 只使用十六进制数，但不带后缀字母 "H"；
- 命令如果不符合 DEBUG 的规则，则以 "error" 提示错误信息，并以 "^" 指示程序出错位置；
- 每个命令只有按下回车键后才有效；
- 命令字母和参数中，相邻两个十六进制数之间必须用逗号或空格分开，其他各部分

之间有无空格或逗号都可以。

DEBUG 主要调试命令的功能与操作介绍如下：

① R 命令：用于显示、修改各个寄存器的内容。

格式：

R [寄存器名]↙

在 DEBUG 状态下，在提示符“-”后输入命令：

-R↙

如图 1.1-4 所示，屏幕显示寄存器 AX、BX、CX、DX、SP、BP、SI、DI、DS、ES、SS、CS、IP 以及标志寄存器的值，所有值均以十六进制数表示。图 1.1-4 中第三行的右端部分给出了标志寄存器中的 8 个标志位，标志位的状态是用字母表示的，其含义如表 1.1-2 所示。图 1.1-4 中最后一行的内容表示所加载的一条程序即将执行的指令。

```
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0B85  ES=0B85  SS=0B85  CS=0B85  IP=0100   NV UP EI PL NZ NA PO NC
0B85:0100 BE9002        MOV     SI,0290
```

图 1.1-4　输入 R 命令后显示的内容

表 1.1-2　标志寄存器标志位的含义

标志位名称	表示 “1” 的符号	表示“0”的符号
溢出标志(OF)	OV	NV
方向标志(DF)	DN	UP
中断允许标志(IF)	EI	DI
符号标志(SF)	NG	PL
零标志(ZF)	ZR	NZ
半进位标志(AF)	AC	NA
奇偶标志(PF)	PE	PO
进位标志(CF)	CY	NC

R 命令可修改某个寄存器的内容。如要显示并修改 AX 寄存器的内容，则可以采用如下 R 命令：

-R AX↙

这时屏幕就会显示出 AX 寄存器的当前值，并提示用户输入要更改的值。如图 1.1-5 所示，屏幕显示 AX 的当前值“0000”和“：”。如在“：”后输入“1234”，然后按回车键，即可完成修改。若不需要修改，则可直接按回车键。最后用 R 命令可以查看修改结果。

```
-R AX
AX 0000
:1234
-R
AX=1234  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0B85  ES=0B85  SS=0B85  CS=0B85  IP=0100   NV UP EI PL NZ NA PO NC
0B85:0100 BE9002        MOV     SI,0290
```

图 1.1-5　R 命令修改 AX 寄存器的值

利用 R 命令还可以修改寄存器标志位，例如输入：

-R F↙

屏幕显示当前的标志位状态为“NV UP EI PL NZ NA PO NC -”，并等待用户输入更改值。输入“OV”进行修改，重新用 R 命令查看结果如图 1.1-6 所示。也可以输入多个标志位同时进行修改。

```
-R F
NV UP EI PL NZ NA PO NC  -OV
-R
AX=1234  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0B85  ES=0B85  SS=0B85  CS=0B85  IP=0100   OV UP EI PL NZ NA PO NC
0B85:0100 BE9002        MOV     SI,0290
-
```

图 1.1-6　R 命令修改标志寄存器的内容

② U 命令：用于反汇编目标代码。可以利用 U 命令将机器代码反汇编为汇编语言的指令符号。U 命令的三种常用格式如下：

U：从当前CS: IP地址开始，每次对32个字节的代码进行反汇编。

U addr：从指定地址(addr)开始进行反汇编。

U addr1，addr2：从地址1(addr1)反汇编到地址2(addr2)。

若只给出偏移地址，则使用 CS 当前值作为段地址。如图 1.1-7 所示是 U 命令的第二种格式，执行 U 命令后，屏幕左边显示的是内存单元的逻辑地址，中间是十六进制形式表示的机器码代码，右边是对应的汇编语言指令。

```
-U 1000
0B85:1000 44            INC     SP
0B85:1001 07            POP     ES
0B85:1002 207506        AND     [DI+06],DH
0B85:1005 F6440710      TEST    BYTE PTR [SI+07],10
0B85:1009 740B          JZ      1016
0B85:100B 8A4404        MOV     AL,[SI+04]
0B85:100E E84E00        CALL    105F
0B85:1011 FF36E591      PUSH    [91E5]
0B85:1015 41            INC     CX
0B85:1016 8A4403        MOV     AL,[SI+03]
0B85:1019 E84300        CALL    105F
0B85:101C FF36E591      PUSH    [91E5]
```

图 1.1-7　U 命令的操作

③ G 命令：用于设置断点并分段执行程序。可以利用 G 命令实现程序的分段执行。G 命令常用的四种格式如下：

G：从当前地址 CS:IP 开始执行程序，直到程序结束。

G = addr：从指定地址(addr)开始执行程序，直到程序结束。

G = addr1，addr2：从地址 1(addr1)执行到地址 2(addr2)。这种格式实际上是在所指定的地址 2 处设置了一个断点，当指令执行到断点时，就停止执行程序并在屏幕上显示当前所有寄存器和标志位的值，以及下一条将要执行的指令。

G addr：从当前地址 CS:IP 执行到指定的地址(addr)，即在 addr 处设置了断点。

若只给出偏移地址，则使用 CS 当前值作为段地址。如图 1.1-8 所示，默认的 CS 段地址为 140CH，命令中只给出偏移地址，即从地址 140CH:0100H 执行到地址 140CH:0112H，在 140CH:0112H 处设置了断点。

```
-G=0100,0112

AX=1000  BX=2000  CX=3000  DX=4000  SP=FFEE  BP=0000  SI=5000  DI=6000
DS=140C  ES=140C  SS=140C  CS=140C  IP=0112   NV UP EI NG NZ NA PO NC
140C:0112 BC0070        MOV     SP,7000
```

图 1.1-8　G 命令的操作

④ D 命令：用于显示内存单元的内容。D 命令常用的三种格式如下：

D [Daddr:]Offset：从指定地址开始显示 128 个字节单元的内容。其中 Daddr 是指定的段地址，缺省时使用 DS 当前值作为段地址，它可以直接指定段地址值，也可使用 DS、ES、CS 和 SS 等段寄存器指定当前段地址；Offset 用于指定段内偏移地址。

D：可继续从上一次显示的内存位置结束的下一字节开始，显示 128 个字节单元的内容。

D [Daddr:]Offset1 Offset2：显示从指定段的偏移地址 1(Offset1)到偏移地址 2(Offset2)内存单元的内容。

如图 1.1-9 所示是第一种和第三种命令格式。若只给出偏移地址，则缺省使用 DS 当前值作为段地址。执行 D 命令后屏幕左边显示的是内存单元逻辑地址，中间是十六进制形式所表示的存储数据，右边是这些数据对应的 ASCII 码字符。

```
-D 0B8F:0100
0B8F:0100  BE 90 02 74 0A F7 D1 F7-D2 83 C2 01 83 D1 00 26   ...t...........&
0B8F:0110  8B 77 06 26 8A 04 3C 00-75 07 B0 01 34 00 7E 0B   .w.&..<.u...4.~.
0B8F:0120  90 46 26 8A 04 3C 00 74-60 46 2E F6 06 BE 90 80   .F&..<.t`F......
0B8F:0130  75 1E 26 3B 4C 03 72 36-77 06 26 3B 54 01 72 2E   u.&;L.r6w.&;T.r.
0B8F:0140  26 3B 4C 07 77 28 72 3A-26 3B 54 05 77 20 EB 32   &;L.w(r:&;T.w .2
0B8F:0150  26 3B 4C 03 7C 18 7F 06-26 3B 54 01 7C 10 26 3B   &;L.|...&;T.|.&;
0B8F:0160  4C 07 7F 0A 7C 1C 26 3B-54 05 7F 02 EB 14 83 C6   L...|.&;T.......
0B8F:0170  09 FE C8 75 B5 2E C7 06-B2 90 06 00 B0 01 B4 FF   ...u............
-D 0100 0150
0B8F:0100  BE 90 02 74 0A F7 D1 F7-D2 83 C2 01 83 D1 00 26   ...t...........&
0B8F:0110  8B 77 06 26 8A 04 3C 00-75 07 B0 01 34 00 7E 0B   .w.&..<.u...4.~.
0B8F:0120  90 46 26 8A 04 3C 00 74-60 46 2E F6 06 BE 90 80   .F&..<.t`F......
0B8F:0130  75 1E 26 3B 4C 03 72 36-77 06 26 3B 54 01 72 2E   u.&;L.r6w.&;T.r.
0B8F:0140  26 3B 4C 07 77 28 72 3A-26 3B 54 05 77 20 EB 32   &;L.w(r:&;T.w .2
0B8F:0150  26                                                &
```

图 1.1-9　D 命令操作

⑤ E 命令：用于显示并修改内存单元的内容。E 命令常用的两种格式如下：

E [Daddr:]Offset：从指定地址开始显示一个字节单元的内容，用户可以通过输入新值进行修改，按空格键表示确认修改，这时会自动显示下一个单元的内容；如果不修改该单元的内容，则可以直接按空格键。按回车键表示 E 命令结束。如果不指明段地址，则默认段地址为 DS 当前值。

E [Daddr:]Offset Expression：Expression 为多个字节内容构成的表达式，字节之间用空格间隔。例如命令 E100 10 20 30 40 50 表示将 DS:100H 开始的 5 个字节单元的内容改成“10H 20H 30H 40H 50H”。

如图 1.1-10 所示，首先用 D 命令查看地址 1000H:0000H 到 1000H:0005H 存储的数据，均为 0；然后使用 E 命令第二种格式，从地址 1000H:0000H 开始，输入表达式“1 2 3 ‘A’ ‘a’ 4”进行修改；最后用 D 命令查看修改结果。

```
-D1000:0 5
1000:0000  00 00 00 00 00 00                                     ......
-E1000:0 1 2 3 'A' 'a' 4
-D1000:0 5
1000:0000  01 02 03 41 61 04                                     ...Aa.
```

图 1.1-10　E 命令操作

⑥ F 命令：用于给一块内存区域置入同一个值。F 命令常用的两种格式如下：

F [Daddr:]Offset1 Offset2 Expression：缺省段地址为 DS 当前值。表示从偏移地址 1(Offset1)到偏移地址 2(Offset2)的所有单元用表达式(Expression)的值依次写入。例如命令 F100 200 55 AA 表示将 DS:100H 到 DS:200H 的所有单元依次写入 55H 和 AAH。

F [Daddr:]Offset L length Expression：表示从地址(Offset)开始、长度为“length”值的所有单元用表达式(Expression)的值依次写入。例如命令 F100L100 55 AA 表示从 DS:100H 开始，长度为 100H 的所有单元依次写入 55H 和 AAH。

如图 1.1-11 所示，首先用 D 命令查看 2000H:0000H 到 2000H:0008H 单元的数据；然后用 F 命令的第一种格式将数据“AAH”写入地址 2000H:0000H 到 2000H:0008H 单元；写入后用 D 命令进行查看。

```
-D 2000:0000 0008
2000:0000  00 00 00 00 00 00 00 00-00
-F 2000:0000 0008 AA
-D 2000:0000 0008
2000:0000  AA AA AA AA AA AA AA AA-AA
```

图 1.1-11　F 命令操作

⑦ M 命令：用于将一块区域的内容传送到另一位置。

M 命令常用的两种格式如下：

M [Daddr:]Offset1 Offset2 Offset3：默认段地址为 DS 当前值。表示将从偏移地址 1(Offset1)到偏移地址 2(Offset2)的所有单元的内容传送到偏移地址 Offset3 开始的单元中。例如命令 M100 200 300 表示将 DS:100H 到 DS:200H 的所有单元内容传送到 DS:300H 开始的单元中。

M [Daddr:]Offset1 L length Offset2：将从偏移地址 1(Offset1)开始、长度为“length”值的所有单元的内容传送到偏移地址 Offset2 开始的单元中。

如图1.1-12所示，首先使用D命令查看了地址1000H:0000H到1000H:0008H和2000H:0000H到2000H:0008H单元存储的数据；然后使用M命令的第一种格式，将2000H:0000H到2000H:0008H单元存储的数据传送到1000H:0000H开始的单元；最后用D命令查看1000H:0000H到1000H:0008H单元存储的数据。

```
-D1000:0000 0008
1000:0000  41 42 61 62 62 63 00 00-00
-D2000:0000 0008
2000:0000  AA AA AA AA AA AA AA AA-AA
-M2000:0000 0008 1000:0000
-D1000:0000 0008
1000:0000  AA AA AA AA AA AA AA AA-AA
```

图 1.1-12　M 命令操作

⑧ A 命令：用于输入汇编语言指令。在 DEBUG 调试环境下，可以利用 A 命令直接输入汇编语言的指令。在使用 A 命令时，若只给出偏移地址，则使用 CS 当前值作为段地址。A 命令常用的两种格式如下：

A [Daddr:]Offset：从指定地址的偏移地址 Offset 开始输入汇编语言指令，每输入一条指令，DEBUG 软件会自动编译该指令，并生成相应的机器代码，同时计算出下一条指令

的存放地址，用户可以继续输入汇编语言指令。按回车键则结束 A 命令。

A：可从上一次 A 命令结束的地址输入汇编语言指令。如果是第一次使用，则默认从当前 CS:IP 地址开始输入汇编语言指令。

如图 1.1-13 所示为采用第一种命令格式，即从当前 CS 段地址下，偏移地址 100H 开始，输入了几条指令。

```
-A 100
0B8F:0100 MOV AX,1234
0B8F:0103 MOV BX,1111
0B8F:0106 ADD AX,BX
0B8F:0108
```

图 1.1-13　A 命令操作

⑨ T 及 P 命令：用于程序的单步执行。在 DEBUG 调试环境下，可以利用 T 或 P 命令单步执行程序。T 命令常用的两种格式如下：

T[=addr]：从指定地址起执行一条指令后停止，并显示当前所有寄存器和标志位的值，以及下一条将要执行的指令。如不指定地址则从当前 CS:IP 地址开始运行。

T[=addr][value]：该格式可以跟踪执行多条指令，从指定地址起执行 n 条指令后停下来，n 值由 value 指定。

如图 1.1-14 所示，利用 T 命令跟踪执行图 1.1-13 中的几条指令。首先修改 IP 寄存器值为 100H，然后从 CS:100H 处开始单步执行，从 AX、BX 寄存器的值可查看指令执行的结果。

```
-RIP
IP 0105
:100
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0290  DI=0000
DS=0B8F  ES=0B8F  SS=0B8F  CS=0B8F  IP=0100   NV UP EI PL NZ NA PO NC
0B8F:0100 B83412        MOV     AX,1234
-T

AX=1234  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0290  DI=0000
DS=0B8F  ES=0B8F  SS=0B8F  CS=0B8F  IP=0103   NV UP EI PL NZ NA PO NC
0B8F:0103 BB1111        MOV     BX,1111
-T

AX=1234  BX=1111  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0290  DI=0000
DS=0B8F  ES=0B8F  SS=0B8F  CS=0B8F  IP=0106   NV UP EI PL NZ NA PO NC
0B8F:0106 01D8          ADD     AX,BX
-T

AX=2345  BX=1111  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0290  DI=0000
DS=0B8F  ES=0B8F  SS=0B8F  CS=0B8F  IP=0108   NV UP EI PL NZ NA PO NC
```

图 1.1-14　T 命令操作

P 命令的格式如下：

P[=addr][value]

P 命令与 T 命令的区别为：T 命令每次只执行汇编语言的一条指令，而 P 命令每次执行汇编语言的一条语句；对于像 CALL sub、INT n 这样的语句，执行 T 指令进入子程序或中断服务子程序，而执行 P 命令时，则执行完整个子程序或中断服务子程序。

⑩ Q 命令：用于退出 DEBUG 调试环境。在 DEBUG 状态下，输入 Q 命令可以退出 DEBUG 调试环境。

3. 汇编源程序在微机上运行的环境要求

为了能在微机上调试和运行汇编语言程序，须准备以下软件：

(1) DOS 操作系统或者 Windows 操作系统下的 DOS 运行环境；

(2) 编辑软件(常用记事本)、宏汇编程序 MASM.EXE、连接程序 LINK.EXE。

使用 Windows 操作系统下的 DOS 运行环境启动 DEBUG 时，需要注意微机系统是 32 位还是 64 位的，在非集成开发环境下，使用 64 位机时需要安装 DosBox 软件。由于 64 位系统没有且不支持 DEBUG 调试程序，所以还需要安装 DEBUG.EXE 软件。

本章的汇编语言程序设计实验可在满足以上软件环境的 PC 上完成，也可在后面 2.3 节介绍的 TPC-ZK-II 集成开发环境下完成。

1.1.2 DEBUG 程序调试

一、实验目的

(1) 学习使用 DEBUG 常用调试命令；

(2) 学习用 DEBUG 命令调试简单程序；

(3) 通过调试程序，熟悉汇编语言指令、寄存器、标志位、堆栈等知识点。

二、实验内容

用 DEBUG 的命令对简单程序段进行调试练习。

三、调试步骤及调试练习

1. 进入 DEBUG 调试环境

启动 DEBUG 程序。当机器控制权由 DOS 成功地转移给调试程序 DEBUG 后，将显示符号“-”，它是 DEBUG 的状态提示符，表示可以接受调试命令了。

• 32 位机下启动 DEBUG 程序

在微机桌面单击“开始”菜单中的“运行”命令，在弹出对话框中输入“cmd”，进入 DOS 命令环境，然后在命令行中输入“DEBUG”指令，进入 DEBUG 调试环境。

• 64 位机下启动 DEBUG 程序

例如在“D”盘建立文件夹“zhangsan”。在“zhangsan”文件夹下拷入“debug.exe”系统程序。安装 DosBox 软件，在桌面打开 DosBox 软件。如图 1.1-15 所示，在“Z:\>”提示符下输入：

```
mount D D:\zhangsan ↙        ; 将用 D 代替 D:\zhangsan
```

在“Z:\>”提示符下输入：

```
D:↙
```

在“D:\>”提示符下输入：

```
DEBUG↙
```

即可显示“-”，表示已启动了 DEBUG 调试程序，进入了调试环境。

此为试读，需要完整PDF请访问：www.ertongbook.com