21世纪 高等学校电子信息类专业规划教材

汇编语言程序设计实训教程

秦 莲 主 编
殷肖川 姬伟锋 孙 鹏 编 著

清华大学出版社
http://www.tup.tsinghua.edu.cn
北京交通大学出版社
http://press.bjtu.edu.cn

21世纪高等学校电子信息类专业规划教材

汇编语言程序设计实训教程

秦　莲　主编
殷肖川　姬伟锋　孙　鹏　编著

清华大学出版社
北京交通大学出版社
·北京·

内 容 简 介

汇编语言程序设计是高校计算机专业的经典课程之一。本书是与《汇编语言程序设计》教材配套的实训教程，编写的目的是使学生通过实验练习加深对理论课程的理解，全书选用多个具有代表性的实验，对汇编语言结构化和模块化程序进行了深入的解析，详细叙述了汇编语言程序的编程与调试过程，并给出正确结果。

全书正文共6章，主要内容包括：汇编语言程序的语句组成、汇编语言源程序的书写格式和汇编语言源程序上机调试运行方法；汇编语言程序设计基本结构实验，即顺序程序实验、分支程序实验、循环程序实验；汇编语言程序子程序调用实验和模块化程序设计实验；I/O 程序实验，即键盘扫描实验、显示控制实验和串口通信实验；WIN32 汇编程序实验，即显示程序实验、键盘消息处理实验和鼠标消息处理实验；汇编语言与 C/C++语言的混合程序设计方式，即 C/C++嵌入汇编程序实验和 C/C++调入汇编程序模块实验。

本书可作为高校计算机专业、自动化控制专业及相关专业本科生汇编语言程序设计实验课程的教科书，也可作为相关领域的工程技术人员的实验参考书。

图书在版编目(CIP)数据

汇编语言程序设计实训教程/秦莲主编. —北京：清华大学出版社；北京交通大学出版社，2005.5
（21 世纪高等学校电子信息类专业规划教材）
ISBN 7－81082－524－0

Ⅰ.汇… Ⅱ.秦… Ⅲ.汇编语言－程序设计－高等学校－教材 Ⅳ.TP313

中国版本图书馆 CIP 数据核字(2005)第 045601 号

责任编辑：周益丹
出 版 者：清华大学出版社 邮编：100084 电话：010－62776969
北京交通大学出版社 邮编：100044 电话：010－51686414
印 刷 者：北京鑫海金澳胶印有限公司
发 行 者：新华书店总店北京发行所
开 本：185×260 印张：8.5 字数：205 千字
版 次：2005 年 5 月第 1 版 2005 年 5 月第 1 次印刷
书 号：ISBN 7－81082－524－0/TP·194
印 数：1～4 000 册 定价：14.00 元

前　言

汇编语言是提供给用户直接访问计算机系统最快而又最为有效的一种编程语言，使用汇编语言编写程序能够充分发挥计算机硬件系统的功能，具有占用存储空间少、运行速度快以及代码质量高等优点，那些需要对计算机硬件进行控制或对运行时间和效率有要求的系统软件或应用软件，通常都是用汇编语言编写的，因此熟练掌握汇编语言源程序的设计方法是非常重要的。

本书是与《汇编语言程序设计》教材配套的实训教程，目的是使学生通过实验加深对理论课程的理解。本书在整个编写过程中，努力做到实验简单明了，具有一定的代表性，文字解释清晰，通俗易懂。全书正文共6章。

第1章介绍了汇编语言与机器语言的特点，对汇编语言程序的语句组成、汇编语言源程序的书写格式和汇编语言源程序上机调试运行方法进行了重点论述。

第2章介绍了汇编语言程序设计基本结构实验，即顺序程序实验、分支程序实验和循环程序实验。对每类实验都进行了详细的问题分析，实验验证。

第3章介绍了汇编语言程序子程序调用实验和模块化程序设计实验。在子程序调用实验中对主程序与子程序都进行了详细的论述。

第4章介绍了I/O程序实验，对键盘扫描实验、显示控制实验、串口通讯实验等典型实验进行了详细论述。

第5章介绍了WIN32汇编程序的框架，重点叙述了显示程序实验、键盘消息处理实验和鼠标消息处理实验。

第6章介绍了汇编语言与C/C++语言的混合程序设计方式，重点叙述了C/C++嵌入汇编程序实验和C/C++调入汇编程序模块实验。

本书由秦莲同志负责组织编写。具体编写分工是：第1,2,3章由秦莲编写；第4章由姬伟锋编写；第5章由孙鹏编写；第6章由殷肖川编写。

由于编者水平所限，书中难免存在错误和不妥之处，敬请广大读者批评指正。

作　者

2005年5月

目　录

第1章 概 述

1.1 汇编语言简介

程序设计语言的种类很多,总体可分为低级语言和高级语言两种。低级语言有机器语言和汇编语言,高级语言有 C/C ++ ,Pascal,BASIC 等。

1.1.1 汇编语言与机器语言

计算机能够直接识别的是二进制数 0 和 1 组成的代码。机器指令(Instruction)就是二进制编码指令,一条机器指令控制计算机完成一个操作。每种处理器都有各自的机器指令集,某种处理器支持的所有指令的集合就是该处理器的指令系统(Instruction Set)。指令集及使用它们编写程序的规则被称作机器语言(Maching Language)。用机器语言形成的程序是计算机惟一能够直接识别并执行的程序,而用其他语言程序编写的程序必须经过翻译转换成机器语言程序才能执行。因此,机器语言程序常被称为目标程序(或目的程序)。

机器指令一般由操作码(Opcode)和操作数(Operand)构成。操作码是表明操作性质的代码,操作数则表明参加操作的数或数所在的地址。一条机器指令是一组二进制代码,一个机器语言程序就是一段二进制代码序列。用机器语言编写程序的最大缺点是难以理解,因而极易出错,也难以发现错误。所以只是在计算机发展的早期或不得已的情况下,才用机器语言编写程序。现在除了有时在程序某处需要直接采用机器指令填充外,几乎没有人采用机器语言编写程序了。

为了克服机器语言的缺点,人们采用便于记忆、并能描述指令功能的符号来表达机器指令。表示指令操作码的符号称为指令助记符,简称助记符;助记符一般是表明指令功能的英语单词或其缩写。指令操作数同样也可以用易于记忆的符号表示。用助记符表示的指令就是汇编格式指令。汇编格式指令以及使用它们编写程序的规则就形成汇编语言(Assembly Language)。用汇编语言编写的程序就是汇编语言程序。汇编语言是一种符号语言,它用助记符表示操作码,比机器语言容易理解和掌握,也容易调试和维护。但是,汇编语言源程序要翻译成机器语言程序才可以由处理器执行。这个翻译的过程称为“汇编”,完成汇编工作的程序就是汇编程序(Assembler)。

用汇编语言编程时,程序员可以直接使用存储器、寄存器、I/O 端口进行处理,同时也能直接使用 CPU 指令系统及其提供的各种寻址方式,编出高质量的程序,所以用汇编语言编写的程序要比与它等效的高级语言程序生成的目的代码精简很多,因此占用内存少,执行速度快。但是汇编语言程序与所要解决问题的数学模型之间的关系不直观,使得编程难度增大,出错可能性也增大,因而进行程序设计和调试的时间比较长。另外,不同的 CPU 系列具有不同的汇编语言,因此汇编语言在不同机器间的可移植性较差。

1.1.2　汇编语言的语句组成

汇编语言源程序是由一条条语句组成的，语句则由名称、操作助记符、操作数、注释4部分组成。例如：

```
NEXT:MOV AL,80H;AL←80H
DATA  DB   02H,03H,04H,05H,06H,07H,08H
```

1. 名称

语句中的名称是一个标识符，可以由字母(a,b,c,d,e,…,z)，数字(0,1,2,…,9)及特殊符号(?,.,@,-,$)组成。名称必须由字母开头，若名称中有圆点符，则圆点符必须用作第一个字符。注意，数字不能用作第一个字符。构成名称的字符总数可多达31个，若超过31个字符，则只有前31个字符有效。

2. 操作助记符

操作助记符将指出该语句的基本操作功能，每条语句必须有操作助记符，它是汇编语言中规定了明确含义的一组专用符号，所以不能随意使用。例如，MOV是传送指令的助记符，ADD是加法指令的助记符，DB是定义字节变量的伪指令助记符。

3. 操作数

语句的操作数部分，可以是数据本身，也可以是数据所在的地址。前者可以是一个常数，也可以是代表常数的一个标识符或表达式，后者通常是以某种寻址方式给出的存放操作数的地址。不是每条语句都必须含有操作数的，例如：HALT暂停指令就不含操作数。

4. 注释

注释仅用作对语句或程序段的说明，它不是程序的可执行部分，汇编时不形成目标代码，注释必须以分号“;”开头，它可以作为语句的一个部分，也可以作为一个单独的语句。

1.1.3　汇编语言中的常数与表达式

1. 常数

常数可以分为数值常数和字符串常数两类。数值常数按其基数的不同，可以有二进制数、八进制数、十进制数、十六进制数等几种不同的表示形式，汇编语言采用不同的后缀加以区分。例如：101010B=52O=42D=2AH

另外，应该注意，汇编语言中的数值常数的第一位必须是数字，否则汇编时将被看成是标识符。例如：AFH应写成0AFH

字符串常数是以单引号“'”括起来的一串字符，例如：'WELCOME TO YOU';'1234567'需要指出的是，此处单引号中的内容在计算机内部均是以ASCII码表示的。

2. 表达式

表达式由操作数和操作符组成，操作数可以是常数或标识符，也可以是子表达式。操作符在宏汇编语言中非常丰富，可分为算术操作符、逻辑操作符、关系操作符、属性操作符及其他操作符等。具体内容如下。

算术操作符有：+,-,*,/,MOD。

逻辑操作符有：AND(逻辑与)、OR(逻辑或)、NOT(逻辑非)、XOR(逻辑异或)。

关系操作符有:EQ(相等)、NE(不等)、LT(小于)、GT(大于)、LE(小于或等于)、GE(大于或等于)。

1.1.4 汇编语言中的标号及变量

1. 标号

标号是由标识符表示的指令名称,用以标示对应指令的位置(地址)。标号有三个属性:段地址、偏移地址和类型。标号的段地址和偏移地址属性是指该标号对应的指令所在段的段地址和段内的偏移地址。标号的类型属性有两种,即定义为 NEAR 类型表示标号在段内使用,定义为 FAR 类型表示标号在段间使用。

标号的基本定义方法是在指令的操作助记符前添加标识符和冒号,该标识符就是我们所要定义的标号。例如:

```
START:MOV AX,2000H
```

这里 START 代表了 MOV 指令的地址,因此标号可以作为程序转移指令的操作数(即要转向的地址)。

2. 变量

• 定义变量

汇编语言中变量是通过伪指令定义的,常用的定义变量的伪指令格式如下:

```
变量名  DB   表达式  ;定义字节变量
变量名  DW   表达式  ;定义字变量
变量名  DD   表达式  ;定义双字变量
变量名  DQ   表达式  ;定义长字变量
变量名  DT   表达式  ;定义一个十字节变量
```

• 变量属性

变量具有下列属性。

段地址(SEG):变量所在段的段地址。

偏移地址(OFFSET):变量所在段内的偏移地址。

类型(TYPE):变量的类型是所定义的每个变量所占据的字节数。

长度(LENGTH):变量定义时,一个变量名所定义的变量个数。

大小(SIZE):变量定义语句中分配给同一变量名的所有变量总的字节数,其值为该变量的类型与长度的乘积。

1.2 汇编语言源程序的典型结构

汇编语言程序设计与其他程序设计语言一样,需要多种系统软件的支持,要经过编辑、汇编、调试之后才能运行,汇编语言的翻译器(汇编程序)对汇编语言源程序的格式有严格的要求,汇编语言源程序的格式是汇编语言程序设计必须遵循的语法规则。用汇编语言编写的源程序,其结构上具有以下特点:首先,由若干逻辑段组成,由伪指令语句定义和说明。其次,整个源程序以 END 伪指令结束。最后,每个逻辑段由语句序列组成。各语句名可以

是如下几种形式。

(1) 指令语句:对应于 CPU 指令系统中的一条指令,因此为可执行语句,汇编时译成目标码。

(2) 伪指令语句:CPU 不执行的语句,只是汇编时给汇编程序提供汇编信息,并不产生目标代码。

(3) 宏指令语句:实际上是一个指令序列,汇编时产生对应的目标代码序列。

(4) 注释语句:以分号";"开始的说明性语句,汇编程序不予处理,只起注释作用,使程序易于理解。

(5) 空行语句:为保持程序书写清晰,仅包含回车换行符的语句行。

汇编语言源程序的格式有完整格式和简化格式两种,下面分别给出这两种形式。

【例 1.1】已知自 first 开始的内存单元存有 10 字节的数,将它们累加,并将累加和存入 second 单元。

汇编语言源程序如下:

```
data      segment
first     db  00,01,02,03,04,05,06,07,08,09
second    db  ?
data      ends
code      segment
        assume  cs:code,ds:data
start:  mov  ax,data
        mov  ds,ax
        lea  si,first
        mov  cx,000ah
        mov  bl,00h
next:   mov  al,[si]
        add  bl,al
        inc  si
        loop next
        mov  second,bl
        hlt
code    ends
        end  start
```

【例 1.2】将 dat1 开始的内存单元的非压缩的十进制数转换为压缩的十进制数,结果仍存入 dat1 中。

汇编语言源程序如下:

```
          dosseg
          .model  small
          .stack  200h
          .data
  dat1    dw   0506h
```

```
        .code
start:  mov  ax,@ data
        mov  ds,ax
        mov  ax,dat1
        mov  cl,4
        sal  ah,cl
        rol  ax,cl
        rol  al,cl
        mov  byte ptr dat1,al
        mov  ah,4ch
        int  21h
        end  start
```

上述例1.1为完整格式,例1.2为简化格式。

1.2.1 上机调试过程

一个汇编语言源程序从编程到投入运行,一般要在多种软件系统的支持下,经过编辑、汇编、连接、装入、和调试等处理过程。

1. 编辑汇编源程序

上机调试的第一步是编辑源程序。编辑源程序可以通过任何一个文本编辑器实现。例如Windows系统中的记事本、写字版,DOS中的全屏幕文本编辑器EDIT等,也可以采用MASM程序员工作平台PWB中的编辑环境。

【例1.3】在Windows记事本中编写查询学生成绩程序。

汇编源程序如下:

```
.model  small
.stack
.data;      为初始变量和查询结果分配单元
   tab db   68H,78H,42H,84H,80H,85H,56H,77H,87H,56H
   no        db  6
   english   db  ?
.code
.startup
   lea  bx,tab          ; bx指向TAB表首地址
   mov  al,no           ;学号送al寄存器
   dec  al
   xlat tab             ;用换码指令查表
   mov  english,al      ;结果保存在english单元
.exit   0               ;结束退出
   end
```

程序中学生成绩按学号(从1开始)从小到大的顺序排列在tab表中,被查询的学生学号放在变量no中,查表结果放在变量english中。

假定将源程序存储到C:\目录下,命名为1-3.asm。

2. 用MASM汇编

汇编是将源程序翻译成由机器代码组成的目标模块文件的过程。

命令如下:

```
C:\masm 1-3.asm
```

如果源程序中没有语法错误,MASM将自动生成一个目标模块文件1-3.obj,否则MASM将给出相应的错误信息。这时应根据错误信息,重新编辑修改源程序后,再进行汇编。

3. 目标文件的连接

连接程序能把一个或多个目标文件和库文件合成一个可执行文件(.EXE,.COM文件)。

命令如下:

```
C:\link 1-3.obj
```

如果不带文件名,LINK连接程序将提示输入OBJ文件名,它还会提示生成的可执行文件名以及列表文件名,一般采用默认文件名就可以。如果没有严重错误,LINK将生成一个可执行文件1-3.exe,否则将提示相应的错误信息,这时需要根据错误信息重新修改源程序后再汇编、链接,直到生成可执行文件。

4. 程序调试

汇编语言源程序经过汇编和链接后就生成了.EXE文件,大部分程序必须经过调试阶段才能纠正程序执行中的错误,并得到正确的结果。DEBUG可以帮助我们实现以上功能。

下面我们以1-3.exe演示DEBUG的使用过程。

(1) 进入DEBUG并装入程序

C:\debug 1-3.exe ↙

-

DEBUG以短划线"-"作为提示符,出现短划线时可键入DEBUG命令。

(2) 使用反汇编命令

程序中学生成绩按学号(从1开始)从小到大的顺序排列在tab表中,要查询的学生学号放在变量no中,查表结果放在变量english中。要查询运行结果必须知道变量english对应的存储单元的地址。另外,查询运行结果时希望程序运行后停在返回DOS之前,因此要确定断点地址。

键入反汇编命令U后,显示信息如下:

```
-U
   0B84:0000 BA860B          MOV     DX,0B86
   0B84:0003 8EDA            MOV     DS,DX
   0B84:0005 8CD3            MOV     BX,SS
   0B84:0007 2BDA            SUB     BX,DX
   0B84:0009 D1E3            SHL     BX,1
   0B84:000B D1E3            SHL     BX,1
```

```
0B84:000D D1E3          SHL     BX,1
0B84:000F D1E3          SHL     BX,1
0B84:0011 FA            CLI
0B84:0012 8ED2          MOV     SS,DX
0B84:0014 03E3          ADD     SP,BX
0B84:0016 FB            STI
0B84:0017 8D1E0A00      LEA     BX,[000A]
0B84:001B A01200        MOV     AL,[0012]
0B84:001E FEC8          DEC     AL
0B84:0020 D7            XLAT
0B84:0021 A21300        MOV     [0013],AL
0B84:0024 B8004C        MOV     AX,4C00
0B84:0027 CD21          INT     21
```

由反汇编结果可知 tab 对应的内存单元的起始地址为“000A”,no 对应的内存单元的起始地址为“0012”,english 对应的内存单元的起始地址为“0013”,程序执行到“0024”时调用 INT 21 返回 DOS,因此执行断点设为“0024”可保证程序完整执行。

注意:由于程序中使用. startup 伪指令作为程序的默认入口地址,程序汇编链接后会产生一个程序前缀,因此程序代码段的第一条指令的偏移地址非 0。例如,例 1.3 中的 lea bx, tab 指令的偏移地址为“0017”,但是利用 G 命令执行程序时起始地址仍为“0000”。如果使用标号作为入口地址,程序代码段第一条指令的偏移地址一般为 0。

(3) 断点执行

得到程序运行结果的存储地址后,就可以断点执行该程序。

键入 G 0024,信息如下:

```
-g 0024
AX=0077  BX=000A  CX=0034  DX=0B86  SP=0420  BP=0000  SI=0000  DI=0000
DS=0B86  ES=0B74  SS=0B86  CS=0B84  IP=0024  NV UP EI PL NZ NA PE NC
0B84:0024 B8004C        MOV     AX,4C00
```

此时程序停在断点处,并显示出所有寄存器以及各标志位的当前值,最后一行给出了将要执行指令的地址、机器语言和汇编语言。根据这些信息可以判断程序执行是否正确,例如,由 AX=0077 可知 AL=77,正是我们要查找的学生的成绩。

(4) 察看运行结果

根据已知的 tab、no、english 的地址,利用显示内存单元内容命令 D 观察运行结果。

键入 D 000A,信息如下:

```
-D 000A
0B86:0000                                78 84 80 85 56 77                x...Vw
0B86:0010  87 56 06 77 26 FF 36 84-13 26 FF 36 82 13 26 8E    .V.w&.6..&.6..&.
0B86:0020  34 7E 13 55 8B EC F7 46-06 00 02 5D 74 01 FB CF    4~.U...F...]t...
0B86:0030  68 01 80 60 1E 06 8B EC-E9 E0 00 68 02 80 60 1E    h..`.......h..`.
0B86:0040  06 8B EC E9 D5 00 68 03-80 60 1E 06 8B EC E9 CA    ......h..`......
```

其中左边给出数据的起始地址(格式为数据段地址:偏移地址),然后顺序给出每个字节单元的内容,中间用十六进制数表示内容,右边用字符显示内容。由以上内容可以看出从000AH开始的8个连续存储单元存储了成绩表,0012H单元存储学生学号6,0013H单元存储学号为6的学生成绩77,执行结果正确。

(5) 退出DEBUG

程序调试结束,利用Q命令退出DEBUG。

键入Q,信息如下:

```
-q
C:\
```

上面介绍了DEBUG调试执行程序的一般过程,大家可以结合程序的具体情况利用其他DEBUG命令灵活运用。DEBUG是一个功能强大调试工具,在程序调试过程中会为我们提供很大的帮助。

1.2.2 常用DEBUG命令

DEBUG程序是专门为分析、研制和开发汇编语言程序而设计的一种调试工具,具有跟踪程序执行、观察中间运行结果、显示和修改寄存器或存储单元内容等多种功能。它能使程序设计人员或用户"触及"机器内部,是我们学习汇编语言必须掌握的调试工具。

DEBUG是DOS的一个外部命令,其命令格式为:

[path]DEBUG [filename] [parm1] [parm2]

其中[path]是DEBUG命令在磁盘上的路径;filename是要用DEBUG来处理的文件的名字,它包括文件的盘符、路径、主文件名和扩展名;参数parm1和parm2是文件filename运行时使用的参数。

当启动DEBUG时,将对CPU的各寄存器进行初始化,初始化操作将按以下几种方式进行。

(1) 如果启动时指定的filename是.EXE文件,则DEBUG启动后将自动把指定的文件装入内存,并置:

① CS为程序代码段段地址

② IP为第一条要执行指令的偏移地址

③ SS为堆栈段段地址

④ SP为堆栈底部+1单元的偏移地址

⑤ DS和ES是装入文件前第一个可用内存段的段地址(即DEBUG程序后的第一个段地址)

⑥ 标志寄存器的所有标志位为0

⑦ BX(0)和CX是装入的文件长度

⑧ 其余寄存器为0。

(2) 如果启动DEBUG时指定的文件filename不是.EXE文件,则DEBUG将把文件装入内存,并置:

① 4个段寄存器为DEBUG程序后面的第一个段地址

② IP指向100H

③ SP 指向这个段的段尾

④ 标志寄存器的所有标志位为 0

⑤ BX 和 CX 是装入的文件长度

⑥ 其余寄存器为 0。

(3) 如果启动 DEBUG 时不指定 filename,则只是把 CPU 的各寄存器进行初始化,初始化结果与上述的第(2)点相同。这时要想显示、修改文件,可以用 DEBUG 的子命令装入文件。

DEBUG 的命令都用单个字母表示,其后可跟一个或多个参数,参数之间用空格或逗号分隔。DEBUG 的命令参数大多数是地址或地址范围,其地址书写格式为:

[段地址:]偏移地址

其中的段地址可以用段寄存器名表示,也可以用一个十六进制数表示。例如:

```
ES:100 43A5:200
```

地址范围的书写格式有以下两种。

① [段地址:]起始偏移地址终止偏移地址

② [段地址:]起始领衔地址 L 长度

例如,CS:100 10F 和 CS:100 L10 所指的地址范围是一致的。

当输入的命令不正确时, DEBUG 将在该行底下指出错误所在。

注意:在 DEBUG 下,输入的数据和显示的数据都是十六进制数,不用在数据后加"H"。

下面介绍 DEBUG 常用命令,其他命令请参阅本书附录 C。

1. 汇编与反汇编命令

- 汇编命令 A

格式:A [地址]

功能:从键盘输入汇编程序,并逐条地把汇编指令翻译成机器代码指令存入对应内存单元。

说明:如果不指定汇编地址,则以 CS:IP 为地址

- 反汇编命令 U

格式:U [地址]/[地址范围]

功能:将指定地址范围内的机器代码翻译成汇编源程序指令显示出来,并同时显示地址及代码。

注意:反汇编时一定确认指令的起始地址,否则得不到正确的结果。

2. 显示与修改内存单元内容的命令

- 显示内存单元内容命令 D

格式 1:D [地址]

格式 2:D 地址范围

说明:D 命令在屏幕上显示的内容分为 3 部分,左边是每一行存储单元的起始地址,中间是各字节单元的内容,右边是各单元内容对应的 ASCII 码字符(不可显示的字符用"."代替)。

例如：

```
-D 200↙
0B2E:0200  E8 DA E1 46 E8 AC DF 74-0D E8 45 00 AC E8 41 00   ...F...t..E...A.
0B2E:0210  81 CD 00 40 EB 34 3C 0D-75 09 B0 00 AA 81 CD 00   ...@ .4<.u.......
0B2E:0220  40 EB CE E8 2B 00 E8 B4-DF 06 57 51 0E 07 BF A3   @ ...+.....WQ....
0B2E:0230  8F B9 06 00 81 CD 00 40-F2 AE 75 0B 81 E5 FF BF   .......@ ..u.....
0B2E:0240  B8 01 00 D3 E0 0B E8 59-5F 07 B0 00 AA 5F 9D F8   .......Y_....._..
0B2E:0250  C3 AA 41 FE 06 E8 99 C3-2E C7 06 55 91 00 00 2E   ..A........U....
0B2E:0260  89 0E DF 91 2E 89 26 E1-91 2E 89 36 E3 91 FC 2E   ......&....6....
0B2E:0270  89 0E 48 91 2E C7 06 4A-91 00 00 2E C7 06 5D 91   ..H....J......].
```

其中 0B2E 是段基址。

• 修改内存单元内容命令 E

格式 1：E 地址内容表

说明：内容表可以是以逗号或空格分隔的两位 16 进制数，也可以是用单引号' '或双引号""括起来的字符串，还可以是二者的组合。

格式 2：E 地址

说明：在修改数据时可用以下键进行不同操作。

◇ 键入空格键。修改后一个字节单元的内容。

◇ 输入减号“-”。另起一行，修改前面一个字节单元的内容。

◇ 输入回车键，结束内存单元的修改。

例如：

```
-E 100 'ABC' 12 34↙      ;修改 DS:0100 为起始单元的连续五个字节单元的内容
-D 100↙                  ;显示修改后的结果
0B2E:0100  41 42 43 12 34 61 2E 61-73 6D 41 96 2B 00 00 05   ABC.4a.asmA.+...
0B2E:0110  53 54 41 43 4B 05 5F 44-41 54 41 06 34 00 1D 0B   STACK._DATA.4...
```

• 填充内存命令 F

格式：F 地址范围内容表

功能：将 <内容表> 的值逐个填入指定地址范围，内容表中的内容用完后再重复使用。

3. 显示与修改寄存器内容命令

格式 1：R

功能：显示当前所有寄存器内容，状态标志及将要执行的下一条指令的地址、代码和汇编指令形式。

格式 2：R 寄存器名

功能：显示并修改指定寄存器的内容

例如：

```
-R↙
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B2E ES=0B2E SS=0B2E CS=0B2E IP=0100 NV UP EI PL NZ NA PO
NC B2E:0100 99            CWD
```

```
-R AX↙          ;输入命令
AX 0000         ;显示 AX 的内容
:               ;供修改,不修改按回车。
```

4. 运行和跟踪命令

• 运行程序命令 G

格式:G [=起始地址] [断点地址]

功能:从起始地址开始执行程序,直到程序结束或遇到断点地址为止。

说明:如果不指定起始地址,则从 CS:IP 处开始执行。如果程序执行到结束,则显示"Program terminated normally"(程序正常结束)。如果遇到断点,则程序停止执行,并显示当时各寄存器的内容和下一条要执行的指令。

• 跟踪运行命令 T

格式:T [=起始地址] [指令条数]

功能:逐条跟踪程序的运行,同时显示出各寄存器的内容、状态标志和下一条要执行的指令,当执行指令达到指定的指令数后就暂停程序的运行。

说明:如果不指定起始地址,则从 CS:IP 处开始执行。不指定指令条数时,认为只执行一条指令。

• 继续命令 P

格式:P [=起始地址] [指令条数]

功能:与 T 命令一样完成跟踪程序的运行,但遇到子程序、中断程序、循环时并不跟踪下去,而是把它们当作一条指令来执行。

5. 磁盘读写命令

• 文件命名命令 N

格式:N 文件名

功能:指定要装入内存或写到磁盘的文件的名字(包括盘符和路径)。

• 装入命令 L

格式:L [地址] [驱动器号 扇区号 扇区数]

功能:把指定文件或磁盘扇区的内容装入到内存指定地址

说明:地址的默认值为 CS:100。驱动器号用 0 表示 A 盘,1 表示 B 盘,2 表示 C 盘。

• 写磁盘命令 W

格式:W [地址] [驱动器号 扇区号 扇区数]

功能:将指定内存地址的一片单元内容写到磁盘中。

说明:地址的默认值为 CS:100。要将内存内容写入文件时,必须先用 N 命令命名一个文件,并置 BX 和 CX 为文件长度。

注意:W 命令不能写入以. EXE 和. HEX 为扩展名的文件。

6. DEBUG 的其他命令

• 移动内存命令

格式:M 源地址范围 目标起始地址

功能:把 < 源地址范围 > 中的内容顺序移到 < 目标起始地址 > 起的一片连续内存单元。

注意:源区域的数据不因移动而消失,其内容仍保持不变。源、目标中的地址只要不指