HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY PRES

微机应用系统设计

WEIJI YINGYONG XITONG SHEJI

朱定华 马爱梅 林卫 编著

华中理工大学出版社

TP36
Z81k
736

微机应用系统设计

朱定华　马爱梅　林　卫　　编著

华中理工大学出版社

前　言

本书是作者近10年来在华中理工大学的提高班、少年班、电子与信息工程系的本科和专科的“微机原理实验”及“微机应用系统设计”等课程的教学实践基础上，为加强计算机基础教学工作，促进计算机基础教学水平尽快跃上一个新台阶，根据原国家教委高教司“加强工科非计算机专业计算机基础教学工作的几点意见”和“工科非计算机专业计算机基础教学指南”的精神而编写的。

计算机的普及与应用使人们传统的工作、学习、生活乃至思维方式发生了巨大变化。不会使用计算机进行读写，不会利用计算机进行工作和学习，将成为下一世纪的“文盲”。21世纪是信息的时代，也是人才竞争的时代，为培养出面向21世纪的高质量的具有创造性的科技人才，特对微机原理及应用的实验教学独立设课，单独编写教材，并始终坚持改革，不断提高教学水平。在学校和电子信息工程系领导的全力支持下，我们完成了多项教学研究项目，曾多次获校级、省级优秀教学成果奖，逐步将微机实验室建设成为具有一定规模的80X86微机接口实验的硬件实验基地。实验室面向全校学生开放。实验的做法是，将有关的接口芯片、元件及实验安装板发给每个学生，由学生自行设计、安装、调试实验的电路和程序，培养学生设计微机应用系统的能力。经过11届本科学生和专科学生的实践证明，这种实验改革是成功的。“微机原理实验”课程于1997年被评为湖北省普通高等学校优质课程。

使用PC/Pentium微机做接口实验时，接口电路的故障或调试时的失误可能会损坏计算机。为了保护计算机，我们设计了一块保护接口卡，将该保护接口卡插入PC/Pentium微机的扩展槽中，实验所需要的计算机的总线信号均通过保护接口卡，由扁平电缆线引到机箱外面的实验板上。这就为学习、开发微机应用系统提供了安装调试的实践环境。

考虑到本书既能作为高校学生的实验教材，又能作为学习微机原理及微机应用系统设计的参考书，笔者打破了传统的实验指导书的格式，自成体系地以基础课的形式与要求组织编写。第一章为软件设计基础，介绍汇编语言程序设计的基础知识。第二章为软件设计技术，介绍汇编语言程序设计的基础方法与技巧，结果的显示方法及现代高级汇编程序设计的新的工程设计方法。第三章为硬件设计基础，介绍硬件实践的必要环境及常用检测技术。第四章为硬件设计技术，介绍常用的典型接口的设计技术

及应用方法。第五章为微机应用系统设计，介绍几个典型的微机应用系统的设计与分析方法。这几个设计实例主要选自第二届和第三届全国大学生电子设计竞赛的赛题。

本书主要以占有主导地位的PC/Pentium微机为样机。本书在内容的选取、概念的引入、文字的叙述、图形的描绘等方面，都力求遵循面向应用、重视实用、便于实践的原则。

本书由朱定华、马爱梅、林卫负责编写。瞿坦教授、庞丽萍教授、王福昌教授对本书的出版给予了极力的推荐，电子与信息工程系的领导、电路与系统教研室的老师们给予了大力的支持和帮助，在此一并表示衷心感谢。

由于作者水平有限，加之计算机的发展日新月异，书中不妥之处在所难免，恳请读者批评指正。

编著者

1998年9月

第一章　软件设计基础

本章介绍汇编语言源程序的建立、汇编、调试与运行。通过本章的学习，以达到熟练掌握计算机的指令，提高编程能力的目的，为学习微机原理与微机应用系统设计打下坚实的基础。

在 X86 系统微机上运行汇编语言源程序的过程如下：

①用多功能窗口软件 SK(SIDEKICK)或其他编辑软件建立或修改扩展名为. ASM 的源文件程序(注意：扩展名必须为. ASM)。

②用宏汇编程序 MASM 将. ASM 文件汇编，生成扩展名为. OBJ 的目标文件。由于汇编所得到的目的代码的存放地址并不是可执行的绝对地址，而是浮动的相对地址，因此，必须链接之后才能运行。

③用链接程序 LINK 将. OBJ 文件链接，生成扩展名为. EXE 的可执行文件。

④用调试程序 DEBUG 调试. EXE 文件。

经过上述前三步生成的. EXE 文件，在 DOS 状态下，直接键入文件名，就可将. EXE 文件从磁盘调入内存并立即执行。然而，一个较复杂的汇编语言源程序一次通过的可能性是很小的，免不了会出现一些意想不到的错误。通过汇编与链接的汇编语言程序仅是一个无语法错误的程序，不一定是正确的程序。因此，必须利用调试程序对. EXE 文件进行调试，使其成为既无语法错误，又无逻辑错误的正确程序。一旦程序中存在错误，就必须从修改源程序. ASM 文件开始重复上述过程，直到程序完全正确为止。

第一节　汇编语言源程序. ASM 文件的建立

可以用多功能窗口软件 SK 或其他编辑软件建立汇编语言源程序。建议初学者使用多功能窗口软件 SK。虽然这些编辑软件的窗口都是全屏幕文本编辑窗口，但仅有 SK 与其编辑文本长驻内存，使得编辑与修改源程序十分方便。初学者开始出现的错误多是语法错误，往往要经过多次修改，才可能消除语法错误，通过汇编。使用 SK 仅需按一次〈Alt〉+〈Ctrl〉键就可以很方便地实现全屏幕文本编辑窗口和汇编间的切换。这样，可以大大地缩短排除语法错误的时间。初学者感觉 SK 不便之处主要是在于 SK 没有下拉菜单，录入新文件不太方便。用户可以事先建立一个 3 个段格式的源文件解决这一问题。该文件是所有汇编语言源程序都要有的公用部分(即本书程序中的小写字母部分)。在录入一个新文件时，都在 SK 中先建立该新文件，再用 SK 的块操作命令〈Ctrl〉+K+R 将 3 个段格式的源文件调入建立的新文件中，其他的录入操作就是在数据段和代码段中录入程序。

此为试读，需要完整PDF请访问：www.ertongbook.com

一、多功能窗口软件 SK 简介

SK 是一种比较新颖的多功能窗口软件，它提供了操作方便的全屏幕窗口，主要配置了编辑、计算器、万年日历、ASCII 表、电话自动拨号、文件设置等窗口软件。其编辑窗口是一个与 WORDSTAR/TURBO 相兼容的全屏幕文本编辑系统，用它可以方便地建立、编辑源文件。进入编辑窗口的步骤如下：

①在 DOS 状态下键入 SK 后按回车键，这便将 SK 程序装入内存，且长驻内存。此后便可随时按下面的步骤调用而不必再键入 SK。

②按下〈Alt〉+〈Ctrl〉键，进入主选择窗口。

③按下〈F2〉或〈Alt〉+N 键或将主选择窗口的指示线移至 F2 Notepad 下按下回车键，即可进入编辑窗口。

进入编辑窗口后，〈Alt〉+〈Ctrl〉就是互锁键，即可退出或直接进入编辑窗口，而不需要再经过主选择窗口。初次进入编辑窗口时，其上部会出现一个显示文件目录的小窗口，移动光标至欲编辑的文件名处，按下回车键，该文件便被调入。如不编辑当前文件，按下〈Esc〉键便可退出显示目录的小窗口，输入待建立的新文件名；或键入路径名 *.*，让小窗口显示该路径中的文件目录。若文件太多，查找不便，也可在退出小窗口后键入文件名，将该文件调入。

文件调入编辑窗口之后，便可利用编辑窗口的操作命令和功能键对源文件进行编辑。编辑窗口的操作命令和功能键较多，可随时按下〈F1〉键查询。为方便读者查阅，下面将编辑窗口的常用操作命令和功能键分 3 类进行介绍。

1. 基本命令和功能键

(1)功能键〈F1〉 求助

显示关于编辑窗口的详细信息。

(2)功能键〈F2〉 存储

将当前正在编辑的源文件(简称在编文件)按文件打开时的名称存盘，而将磁盘上原来的同名文件改为后备文件作备份(.BAK 类型)。对于需要存盘的在编文件，在编辑完毕准备退出或再编辑其他文件之前，一定要按〈F2〉键保存。

(3)功能键〈F3〉 建立或调入一个新文件

按〈F3〉键后，显示：New note file：

·若输入文件名(包括扩展名)后回车，则将该文件调入窗口，等待编辑。

·键入 *.* 或直接按下回车键，则显示当前目录中的全部文件，这时可通过移动光标来选择所需要的文件。一旦选中，按下回车键后，该文件就被调入窗口，等待修改。

注意：①若输入当前目录中不存在的文件名时，SK 将提示用户，应首先在当前目录中建立相应的文件。当需要建立一个新文件时，常用这一方法。

②当窗口上方显示文件目录时，只有小键盘上的键有效，所以只能靠小键盘上的光标控制键移动光标。若要在前页或下页目录区中寻找文件，必须首先用〈PgUp〉键或〈PgDn〉键将光标移到前页或下页，然后再用光标控制键→、←、↑、↓来移动光标，选择所需要的文件。

③当窗口上方显示文件目录时，如不需要编辑文件，必须按〈Esc〉键退出显示目录的窗口。

(4)功能键〈F9〉 扩展窗口

按一下此键，就可以利用小键盘上的方向键调节窗口，使之随自己的意愿扩大。再按此键，窗口尺寸即告固定，方向键恢复原来的功能。

(5)功能键〈F10〉 收缩窗口(与〈F9〉具有相反的效果)

(6)功能键〈Esc〉 退出现行窗口

注意：〈Esc〉键每按一次，只能退出现行窗口，此时，屏幕上出现的是该窗口出现之前的信息。如果屏幕上所示的新窗口信息仍不是所希望的，还可再次按〈Esc〉键，直至满意为止。

(7)光标控制键

小键盘上的4个键↑、↓、→、← 上下右左移动光标。

〈Home〉 光标移到行首。

〈End〉 光标移到行尾。

〈Pgup〉 光标移到前页。

〈Pgdn〉 光标移到下页。

〈Ctrl〉+〈Home〉 光标移到页首。

〈Ctrl〉+〈End〉 光标移到页尾。

〈Ctrl〉+〈Pgup〉 光标移到文件首。

〈Ctrl〉+〈Pgdn〉 光标移到文件尾。

(8)插入和删除命令

〈Ins〉(或〈Ctrl〉+V) 插入状态打开/关闭，若屏幕右上角出现"Insert"，则表示可以插入字符；若出现"Overwrite"，则表示不能插入，只能覆盖；若要插入，则再按一次〈Ins〉键，直到出现"Insert"为止，这时可以在光标处插入任意个字符。

〈Ctrl〉+N 行插入。

〈Ctrl〉+Y 行删除。

〈Ctrl〉+Q+Y 删除从光标处开始，至行尾处结束。

〈Ctrl〉+Q+I 自动缩排打开/关闭。当屏幕右上角出现"Indent"时，表示可以自动缩排，关闭时，按〈Ctrl〉+Q+I命令，则"Indent"消失。

〈Del〉(〈Ctrl〉+G) 删除光标所在字符。

另外，若在插入状态将光标移至该行的第一个字符上，再按空格键，则可达到将整行字符右移的目的。

2. Notepad 窗口的块操作命令

〈Ctrl〉+K+B(或〈F7〉) 设定文件块起始标志。

〈Ctrl〉+K+K(或〈F8〉) 设定文件块结束标志。

〈Ctrl〉+K+H 隐去文件块标志。

〈Ctrl〉+K+C 复制文件块。

〈Ctrl〉+K+V 移动文件块。

〈Ctrl〉+K+Y 删除文件块，且每次只能删除一个块。

〈Ctrl〉+K+R 将磁盘上的指定文件读入到窗口光标处。

〈Ctrl〉+K+W 写文件块到磁盘上的指定文件中。

〈Ctrl〉+K+S 文件块分类。

〈Ctrl〉+K+P 文件块打印。

对于由〈F7〉与〈F8〉定义的一个文件块，可以用〈Ctrl〉+K+V 命令轻易地将其移动到任意光标位置；可以用〈Ctrl〉+K+C 命令对文件块进行复制；可以用〈Ctrl〉+K+P 命令将文件块送往打印机输出。编辑窗口与文件之间的数据交换是随时都可以进行的。如果键入〈Ctrl〉+K+R 命令，那么，编辑窗口会询问要读入的文件名，并立即执行将指定的文件读入到编辑窗口光标处的操作。反之，〈Ctrl〉+K+W 命令会执行一个逆操作，即将文件块写到指定的文件中。在特殊情况下，可以利用〈Ctrl〉+K+W 命令来做文件的复制工作。若是编辑窗口文件中的某一部分不再需要，也可以用〈Ctrl〉+K+B 与〈Ctrl〉+K+K 命令将这一部分定义为一个文件块，然后用〈Ctrl〉+K+Y 命令在瞬间将之删除。显然，编辑窗口的文件块命令功能是很强的。

3. 编辑窗口的其他命令

〈Ctrl〉+K+D(或〈F2〉)　文件存盘。

〈Ctrl〉+I(或〈Tab〉)　横表。

〈Ctrl〉+L　重复前次检索。

〈Ctrl〉+Q+F　检索。

〈Ctrl〉+Q+A　检索并替换。

〈Ctrl〉+Q+I　自动 Tab 开/关。

〈Ctrl〉+Q+L　恢复行。

灵活而又熟练地使用编辑窗口的检索、替换功能，能大大地加快修改源文件的速度。当在文件中欲查找某一特定标志时，〈Ctrl〉+Q+F 命令会自动按照用户键入的关键字串执行全程检索，直到第一个目标被发现。如需继续执行前次检索，只要键入〈Ctrl〉+L 命令即可。字符串的替换命令〈Ctrl〉+Q+A 和检索命令的不同之处在于：在继关键字串输入完毕后，编辑窗口还要求给出置换用的字符串。这样可以让 SK 用正确的字符串一次置换出源文件中所有相同的错误。使用〈Ctrl〉+Q+L 命令可以取消对当前行所做的任何修改，使该行恢复成原来的内容，而前提是光标没有离开该行，一旦离开，原内容就无法恢复了。需要说明的是：〈Ctrl〉+Q+L 命令只对〈Del〉命令所删除的行有效，而对〈Ctrl〉+Y 命令所删除的行是无效的。

命令〈Ctrl〉+Q+F 是和命令〈Ctrl〉+L 配合使用的，其中查找命令的选择项有：

B：向回找，即从当前光标位置开始向文首移动，寻找与关键字串相匹配的字符串。

n(n 为正整数)：从当前光标位置开始计数，寻找第 n 个与关键字串相匹配的字符串。

U：把大小写字母看成是等价的。

W：仅查找整个字，跳过那些嵌入在其他字中的匹配串。

BU：向回找，不区分大小写。

在〈Ctrl〉+Q+A 命令中的选择项有：

B：向回找，即从当前光标位置开始向文首移动，查找并替换与关键字串相匹配的字符串，可用〈Ctrl〉+L 命令继续执行下去。

G：全程查找并替换，即在整个文件中查找并替换字符串。

n(n 为正整数)：从光标位置开始计数，查找并替换第 n 个与关键字中相匹配的字符串。

N：不用询问，直接替换，并且以〈Ctrl〉+L 命令继续执行下去，从光标位置开始向文尾查找并替换字符串。

GW：在整个文件中查找并替换整个字，不区分大、小写。

最后需要说明的一点是：通常编辑窗口能编辑的源文件不超过 40K 个字节，对于一般用户来说，这个容量是足够的，若需要改变这个约定的容量，可通过文件设置窗口实现，对此不再介绍，读者需要时可查阅有关 SK 的使用说明。

二、3 个段基本格式源文件 ABC. ASM 的建立

在空目录 ASM 下，键入 SK 后按下回车键，SK 程序被调入内存。按下〈Alt〉+〈Ctrl〉键，进入主选择窗口，按下〈F2〉键，进入 SK 的编辑窗口。同时还出现一个小窗口，并提示“No file (s)”。按下〈Esc〉键，退出小窗口，并在显示屏的上部显示“New note file:”提示行，光标位于提示行的冒号后。此时键入 ABC. ASM 后按下回车键，提示行显示“G:\ASM\ABC. ASM”，光标位于编辑窗口的左上角。此时可以输入下面的源程序。

```
stack   segment stack 'stack'
        dw 32 dup(?)
stack   ends
data    segment

data    ends
code    segment
begin   proc far
        assume ss:stack, cs:code, ds:data
        push ds
        sub ax,ax
        push ax
        mov ax,data
        mov ds,ax

        ret
begin   endp
code    ends
        end begin
```

在最后一行 end begin 之后一定要按下回车键，否则汇编会出现一个错误(85:End of file, no END directive)。

该源程序输入结束后，按下〈F2〉键，将其存盘。至此，3 个段基本格式的源文件 ABC. ASM 的建立就告结束，可供后面所有的源程序读入。

三、汇编语言源程序的建立

下面以“镜子”程序 MIRROR. ASM 为例，说明汇编语言源程序的建立过程。在 SK 的编辑窗口，键入〈F3〉，再在“New note file:”提示行后，键入 MIRROR. ASM 后按回车键。待提示行显示“G:\ASM\MIRROR. ASM”，光标进入编辑窗口闪烁后，键入〈Ctrl〉+K+R 命令。等

到提示行提示“Read block from file:”后，键入 ABC. ASM 后按回车键，将 3 个段基本格式源文件调入编辑窗口。将编写的“镜子”程序的数据段和代码段分别插入 ABC. ASM 中，得到如下完整的镜子程序：

```
stack   segment stack 'stack'
        dw 32 dup (?)
stack   ends
data    segment
OBUF    DB '〉',0DH,0AH,'$'
IBUF    DB 0FFH,0,255 DUP(?)
data    ends
code    segment
begin   proc far
        assume ss:stack,cs:code,ds:data
        push ds
        sub ax,ax
        push ax
        mov ax, data
        mov ds, ax
        MOV DX,OFFSET OBUF          ;显示提示符“〉”并回车换行
        MOV AH,9
        INT 21H
        MOV DX,OFFSET IBUF          ;键入并显示字符串
        MOV AH,10
        INT 21H
        MOV BL,IBUF+1               ;将“$”送于键入字符串后
        MOV BH,0
        MOV IBUF[BX+2],'$'
        MOV DL,0AH                  ;换行
        MOV AH,2
        INT 21H
        MOV DX,OFFSET IBUF+2        ;再显示键入的字符串
        MOV AH,9
        INT 21H
        ret
begin   endp
code    ends
        end begin
```

输入正确无误后，按下〈F2〉键，将其存盘。此时可按下〈Alt〉+〈Ctrl〉键，直接从 SK 的编辑窗口退出 SK，进入 DOS 状态，用 DIR 命令可查看到 MIRROR. ASM 文件，然后可对 MIRROR. ASM 进行汇编。

第二节　汇编语言程序的汇编和链接

一、汇编语言源程序的汇编

汇编语言源程序必须经过汇编，将源程序中的符号、符号指令和伪指令变为机器指令。该汇编工作可以用宏汇编程序 MASM 来完成。

汇编语言源程序经 MASM 汇编后，在磁盘上可以建立 3 个文件：第一个是.OBJ 目标程序文件，在此文件中，符号指令变为机器码，但地址操作数只是可浮动的相对地址，而不是内存的绝对地址；第二个是.LST 列表文件，它把源程序和目标程序都制成表，可打印出来供检查和调试用；第三个是.SBR 文件。

汇编 MIRROR.ASM 源程序，在 DOS 状态下调用 MASM 的汇编操作命令：

```
G:\ASM>MASM MIRROR,MIRROR,MIRROR,MIRROR↙
Microsoft (R) MASM Compatibility Driver
Copyright (C) Microsoft Corp 1991. All rights reserved.

Invoking: ML.EXE/I. /Zm/c/Fomirror.obj/Flmirror.lst/FR/Ta mirror.asm

Microsoft (R) Macro Assembler Version 6.00
Copyrght (C) Microsoft Corp 1981—1991. All rights reserved.

Assimbling: mirror.asm
```

若只生成.OBJ 文件，则可键入简化命令

```
G:\ASM>MASM MIRROR;↙
```

在汇编过程中若发现源程序有语法错误，则计算机可自动列出错误所在的行号和错误性质的编号，并指出错误的类型，最后列出错误的总数。此时应分析错误，修改源程序后再汇编，直至汇编后无错误。

汇编后建立的.LST 文件可用于打印，亦可以被调入 SK 的编辑窗口，以了解汇编后的情况。MIRROR.LST 文件如下：

```
0000                        stack segment stack 'stack'
0000 0020[                  dw 32 dup (?)
      0000
    ]
0040                        stack ends
0000                        data segment
0000 3E 0D 0A 24            OBUF DB '>',0DH,0AH,'$'
0004 FF 00                  IBUF DB 0FFH,0,255 DUP(?)
```


此为试读，需要完整PDF请访问：www.ertongbook.com

```
      00FF[
    ]
0105                        data ends
0000                        code segment
0000                        begin proc far
                            assume ss:stack,cs:code,ds:data
0000 1E                     push ds
0001 2B C0                  sub ax,ax
0003 50                     push ax
0004 B8- - - -R             mov ax,data
0007 8E D8                  mov ds,ax
0009 BA 0000 R              MOV DX,OFFSET OBUF
000C B4 09                  MOV AH,9
000E CD 21                  INT 21H
0010 BA 0004 R              MOV DX,OFFSET IBUF
0013 B4 0A                  MOV AH,10
0015 CD 21                  INT 21H
0017 8A 1E 0005 R           MOV BL,IBUF+1
001B B7 00                  MOV BH,0
001D C6 87 0006 R 24        MOV IBUF[BX+2],'$'
0022 B2 0A                  MOV DL,0AH
0024 B4 02                  MOV AH,2
0026 CD 21                  INT 21H
0028 BA 0002 R              MOV DX,OFFSET OBUF+2
002B B4 09                  MOV AH,9
002D CD 21                  INT 21H
002F CB                     ret
0030                        begin endp
0030                        code ends
                            end begin
```

二、目标文件的链接

由汇编程序生成的目标文件必须经过链接程序(LINK)链接以后才能成为可执行文件。链接程序并不是专为汇编语言设计的,利用它可把若干个模块链接在一起,这些模块可以是汇编程序产生的目标文件,也可以是其他高级语言编译程序产生的目标文件。在链接过程中,将产生几个新文件。链接过程如下:

```
G:\ASM>LINK↙
Microsoft (R) Segmented-Executable Linker Version 5.13
Copyright (C) Microsoft Corp 1984—1991. All rights reserved.
```

Object Modules [.OBJ]:MIRROR↙

Run File [MIRROR.EXE]↙

List File [NUL.MAP]:MIRROR↙

Libraries [.LIB]:↙

Definitions File [NUL.DEF]:↙

程序首先询问用户要链接的目标文件名，此时应键入文件名作为回答。如果有多个要链接的目标文件，应一次键入，各目标文件名之间用"+"号隔开，第一个文件名是默认的可执行文件名。第二个提示询问用户要产生的可执行文件的文件名，一般直接按回车键表示采用括号内规定的默认文件名。第三个提示询问用户要否建立地址分配文件。键入文件名再按回车键表示要建立，直接按回车键表示不要建立。最后询问是否用到.LIB 文件和.DEF 文件，如果没有用到，直接按回车键即可(即允许用户自己建立库文件.LIB 和模块定义文件.DEF)。

回答完以上问题后，链接程序开始进行链接，若链接过程有错，则显示错误信息，且要重新调用编辑程序修改源文件，然后重新汇编、链接，直至无错。

链接以后产生的.MAP 文件提供了地址分配的一些信息，可用 TYPE 命令将它显示出来。MIRROR.MAP 文件如下：

Start	Stop	Length	Name	CLASS
00000H	0003FH	0040H	STACK	STACK
00040H	00144H	0105H	DATA	
00150H	0017FH	0030H	CODE	

Origin Group

Program entry point at 0015:0000

从地址分配文件可以看出，DATA 段的起始地址是 00040H，结束地址是 00144H，字节数为 105H。其他段分配情况依此类推(对于小型、简单的程序，也可不建立该文件)。

同 MASM 一样，链接程序可以使用命令行形式。若汇编程序没有用到.LIB 和.DEF 文件，也不建立.MAP 文件，仅生成.EXE 文件，则只需键入命令：

G:\ASM>LINK MIRROR;↙

以上讨论的是单个模块的链接命令。如果一个程序由 3 个模块组成，各模块经分别汇编后，所得到的目标模块分别为 SUM.OBJ、DUMP.OBJ、MAX.OBJ，则需将它们链接在一起生成可执行文件 SUM.EXE。如果没有用到.LIB 和.DEF 文件，也不需要生成.MAP 文件，可采用以下形式的命令实现链接：

G:\ASM>LINK SUM+DUMP+MAX;↙

链接后生成 SUM.EXE 文件。

第三节　汇编语言程序的调试

建立的汇编语言源程序经过汇编和链接后，可执行文件已存放在磁盘上，此时，在 DOS 状态下，直接键入文件名(可不要扩展名.EXE)，就可把执行文件从磁盘上装入内存，并立即执行。

然而，设计一个较复杂、较长的汇编语言源程序，免不了会出现这样或那样的错误，很难一次通过，这就需要调用DOS支持下的DEBUG程序(调试程序)，调试所生成的可执行文件。在调试程序DEBUG的管理下，可以单步执行程序，也可以设置断点，还可以显示和修改CPU内部寄存器和标志位的内容等，为寻找程序中的错误提供了方便。

一、调试程序DEBUG简介

DEBUG是专为汇编语言程序设计的一种调试工具，它提供一个调试的环境，以便监视和控制被调试程序的执行；可以直接确定程序中出现的问题，并作相应修改，然后立即执行程序，以便判定这些问题是否已经解决，因而不必重新去汇编一个程序就能发现你的修改是否有效；可以装入、修改或显示某个文件。

1. DEBUG的启动

在DOS状态下键入DEBUG[文件引用名]后按回车键。方括号表示其里面的文件引用名是可缺省的，而不是要将文件引用名置于方括号内。若不缺省文件引用名，则文件引用名的扩展名只能是.EXE或.COM，其他扩展名的文件不能调试。若在DEBUG启动时规定了文件引用名，则在DOS下将DEBUG调入内存后，计算机还可自动将被调试的程序装入内存，用户可根据需要对装入的程序进行显示、修改或执行。若未规定文件引用名，则可用命名命令N和装入命令L将需要调试的文件装入内存，然后再对内存中的程序进行修改、显示或执行。对于简短的程序，还可以用汇编命令A直接编辑和汇编，免除前述的汇编、链接等步骤。

程序装入后，出现提示符"-"，这时系统已在DEBUG程序的管理之下，所有的DEBUG命令只有在出现此提示符后才有效。启动DEBUG时，不带文件引用名，各寄存器和标志位置成以下状态：

①4个段寄存器的值相同，置于自由存储空间的底部，也就是DEBUG程序后的第一个段。

②指令指示器IP置为0100H。

③堆栈指示器SP置到段的结尾处，或者是装入程序的临时底部，哪个更低就指向哪个。除SP外其余7个通用寄存器均置为0。

④标志位均清除。

但是，若启动DEBUG时规定了文件引用名，则BX和CX置为以字节表示的文件长度，且当被装入的被调试程序为.EXE文件时，DEBUG还必须进行再分配，将段寄存器、IP和SP置为程序所规定的值。

2. 有关DEBUG命令的一些共同信息

DEBUG命令都是一个字母，在命令字母后，可以跟一个或多个参数。命令和参数之间用定界符逗号或空格分隔。两个相邻的十六进制数之间必须用定界符分隔。

命令中的参数以及程序中的数据都规定为十六进制数，由1～4个十六进制数的字符组成，因此，其后不能跟十六进制标志"H"。

命令中所使用的地址范围及其格式约定如下：

[段地址:]始偏移地址[末偏移地址]

或[段地址:]始偏移地址[L 长度]

其中,段地址可以是段寄存器名,也可以是一个十六进制的数,默认当前段地址时也可以缺省。末偏移地址和 L 长度也可以缺省。若缺省,命令采用规定的地址范围。

每一个命令,只有在按下回车键后才执行。

可以按下〈Ctrl〉+〈Break〉键来终止命令的执行。

若某个命令产生多屏输出,为了能在屏幕上看清楚上卷的行,可以按〈Ctrl〉+〈Numlock〉键,暂停屏幕上卷,按任一键后继续上卷而显示新的内容。

3. 常用的 DEBUG 命令

(1)显示内存单元内容的命令(Dump Command)

为了了解程序执行的结果,检查内存单元的内容是十分重要的。此命令能检查指定范围的存储单元的内容。

命令格式:D[地址范围]

其中,D 是命令字母,[地址范围]是指定要显示的存储单元的范围,可以缺省。

存储单元的内容用两种方式显示:一种是每一个存储单元的内容(每一字节)用两位十六进制数据显示;另一种是用相应的 ASCII 字符显示,句号(.)表示不可显示的字符。每一行显示 16 个字节。第 8、9 字节间有一连字符(-)。

若命令中没有指定地址范围,则从上一个 D 命令所显示的最后一个单元的下一个单元开始显示 8 行,即 80H 个字节。若命令中仅指定了起始地址,则从指定的地址开始显示。若在命令的起始地址中,仅给出了偏移地址,则命令认为段地址包含在 DS 中。

(2)修改存储单元内容的命令(Enter Command)

此命令用于修改存储单元的内容,即逐个单元修改或用命令中给定的字节串去代替指定的单元的内容。

命令格式:E 始地址[字节串]

例如:

-E DS:100 F3 "XYZ"8D↙

DS:100H~DS:104H 这 5 个单元的内容由其后字节串给定的 5 个字节的内容(其中 2 个字节用十六进制数表示,即 F3、8D;另外 3 个用字符表示,就是 XYZ 的 ASCII 码)所代替,该命令执行后,DS:100H~DS:104H 这 5 个单元的内容依次为:0F3H、58H、59H、5AH、8DH。

若命令中没有给出字节串,则屏幕上显示指定单元的地址和原有的内容之后,等待键入:若需要修改原单元的内容,则输入一个字节的十六进制数,以代替原单元中的内容。若不需要修改,则可以按回车键结束此命令,或者按空格键或连接号(-),则该单元的修改完成,且显示下一个(空格键)或者前一个(连接号键)单元的地址和原有的内容。若要修改的话,则输入两位十六进制数,再按空格键或"-"键,……,这样,就可以连续地进行修改。若某一单元的内容不需要修改,而操作要进行下去,则可直接按空格键或"-"键,最终以回车键结束 E 命令。

(3)检查和修改寄存器内容的命令(Register Command)

为了了解程序运行是否正确,检查寄存器内容的操作是十分重要的。

命令格式:R[寄存器名]

若命令未给出寄存器名,则显示 CPU 内部所有寄存器的内容和 8 个标志位的状态。如:

-R↙

AX=0000 BX=1000 CX=014A DX=0000 SP=0064 BP=0000 SI=0000 DI=0000

DS=07B5 ES=0765 SS=07C6 CS=07C6 IP=0000 NV UP DI PL NZ NA PO NC

07C6:0000 B8C507　　MOV　AX,07C5

前两行显示了所有CPU内部寄存器的内容和8个标志位的状态(其含意在下面介绍),第三行显示了现行CS:IP所指的指令的机器码、汇编符号指令以及指令的逻辑地址,这就是下条即将要执行的指令。

若命令中给出了一个寄存器,则该命令的功能是显示和修改该寄存器的内容。例如,为了检查和修改寄存器BX的内容,可键入以下命令:

-R BX↙

则系统显示如下信息:

BX * * * *

其中,* * * * 表示BX的内容。

若不需要改变其内容,则直接按回车键。若需要改变其内容,可输入1～4个十六进制数,再按回车键,以实现修改。例如:

-R BX↙

BX A869

:058F↙

则BX中的内容由A869H改变为058FH。

在8086/8088中,共有9个标志位,其中跟踪标志T不能直接用指令改变,其他8个标志位可以显示和修改。在显示时,8个标志的显示次序和符号如表1-1所示。

表1-1　8个标志位的置位/复位符号

标　志　名	置　位	复　位
溢出 Overflow(是/否)	OV	NV
方向 Direction(减量/增量)	DN	UP
中断 Interrupt(允许/屏蔽)	EI	DI
符号 Sign　(负/正)	NG	PL
零 Zero　(是/否)	ZR	NZ
辅助进位 Auxiliary Carry(是/否)	AC	NA
奇偶 Parity (偶/奇)	PE	PO
进位 Carry　(是/否)	CY	NC

键入命令

-RF↙

系统可能显示如下信息:

OV DN EI NG ZR AC PE CY-

若不需要修改任一个已设置的标志状态,可直接按回车键。若有一个或多个标志需要修改,则可以输入与此标志相反的值,然后按回车键,以实现修改。输入时与标志的次序无关,输入的各个标志之间,可以没有空格。例如:

OV DN EI NG ZR AC PE CY-PONZDINV↙

此命令执行后,各标志的状态如下:

NV DN DI NG NZ AC PO CY

即 OF=0,DF=1,IF=0,SF=1,ZF=0,AF=1,PF=0,CF=1。

(4)运行命令(Go Command)

为了检查程序运行是否正确,希望在运行中能设置断点,以便一段一段地对程序进行调试。该命令还可以连续执行程序。

命令格式:G[=address][address[address…]]

其中,第一个参数=address,规定了执行的起始地址,即以 CS 的内容作为段地址,以等号后面的地址作为偏移地址。在输入时,等号是不可缺少的(以便与后面输入的断点地址相区分)。若不输入起始地址,则以 CS:IP 作为起始地址。后面的地址参数是断点地址。若在命令行中,除了起始地址以外,再没有地址参数,则程序执行时没有断点。

在开始调试程序时,往往要设置断点。DEBUG 程序中允许最多可设置 10 个断点,这些断点地址的次序是任意的。设置多个断点的好处是:若程序有多个通路,不管是哪一个通路执行,都会在相应的断点处停下来,以便于调试和修改。

DEBUG 程序用一个中断类型为 3 的指令(操作码为 CCH)来代替被调试的程序在断点地址处的指令操作码。当程序执行到一个断点地址时就停了下来,显示 CPU 内部所有寄存器的内容以及 8 个标志位的状态(相当于一条 R 命令);被调试程序的所有断点的指令被恢复(程序执行未遇到断点,则不恢复);全部断点被取消;返回 DEBUG。此时,可以利用 DEBUG 的各种命令来检查程序运行的结果和进行必要的修改。

注意:命令中的地址参数所指的地址单元,必须是有效指令的第一个字节,否则会出现不可预料的结果。对于 G 命令,堆栈必须至少包含有 6 个可用的字节,否则会出现不可预料的结果。

(5)跟踪命令(Trace Command)

命令格式:T[=address][n]

T 命令用于从"="号后的地址开始,执行 n 条指令后返回 DEBUG,显示所有寄存器和 8 个标志位。若缺省"=address",则从 CS:IP 的现行值开始执行;若缺省 n,则只执行一条指令。

若调试的程序中有过程调用(包括软中断调用 INT n 指令),则使用 P 命令(Proceed)调试较使用 T 命令更为适宜。P 命令将过程执行完;T 命令则要进入过程,跟踪过程中的指令。P 命令的用法与 T 命令相同。

(6)汇编命令(Assemble Command)

若在调试中发现程序中的某一部分要改写,或要增补一段,则可以直接在 DEBUG 中输入、汇编、运行和调试这一段程序。这比在 DOS 状态下每一次修改都要经过编辑、汇编、链接的过程简便多了。

命令格式:A[地址范围]

该命令将输入的汇编符号指令汇编成机器码,并从指定的地址单元处开始连续存放。

若在命令中没有指定地址范围,但前面用过汇编命令,则接着上一个汇编命令存放的最后一个单元开始存放。

若输入的指令中有错,DEBUG 就显示 Error,然后重新显示现行的汇编地址,等待新的输入。

退出 A 命令可按〈Ctrl〉+C 键或在空行上按回车键。

下面用成块传送程序段来说明 A 命令的使用。

此为试读,需要完整PDF请访问: www.ertongbook.com