微型计算机

IBM PC

的原理与应用

（续篇）

张福炎　周根林等　编著

南京大学出版社

封面设计　郝庆祥

ISBN 7-305-00178-3
TP·15 定价：7.30元

微型计算机IBM PC的原理与应用（续篇）

南京大学出版社

微型计算机
IBM PC 的原理与应用

（续 篇）

张福炎　周根林　李滨宇
滕小羽　蒋新儿　薛　行　编著

南京大学出版社
1988·南京

内 容 简 介

本书是《微型计算机IBM PC的原理与应用》的续篇，它进一步介绍了IBM PC机的各种常用软硬件技术的原理与应用。全书分六章叙述。内容包括IBM PC机的汉字信息处理，宏汇编语言程序设计，8087协处理器的原理与使用，C语言，dBASEⅢ数据库管理系统，以及OMNINET和ETHERNET局部地区网络。本书所取资料新颖、丰富，内容系统全面，构思严谨。书中列举的近百个程序实例，可便于读者学习和掌握有关内容。该书具有比较广泛的适用性，不仅对IBM PC机和国产长城0520系列机，而且对其他微型计算机的广大应用开发人员均有一定的参考价值，同时也可作为高等院校计算机及有关专业的教学用书或参考书。

微型计算机 IBM PC的原理与应用

（续　篇）

张福炎　周根林　李滨宇

滕小羽　蒋新儿　薛　行　　编著

责任编辑　丁　益

南京大学出版社出版

（南京大学校内）

江苏省阜宁印刷厂印刷

江苏省新华书店发行　各地新华书店经销

开本：787×1092　1/16　印张：31　字数：790（千字）

1985年9月第1版　　1988年2月第4次印刷

印数：67001—92000

ISBN 7—305—00178—3/TP·15　　定价：　7.30元

微型计算机
IBM PC 的原理与应用

（续 篇）

张福炎　周根林　李滨宇
滕小羽　蒋新儿　薛　行　编著

南京大学出版社
1988·南京

内 容 简 介

本书是《微型计算机IBM PC的原理与应用》的续篇，它进一步介绍了IBM PC机的各种常用软硬件技术的原理与应用。全书分六章叙述。内容包括IBM PC机的汉字信息处理，宏汇编语言程序设计，8087协处理器的原理与使用，C语言，dBASEⅢ数据库管理系统，以及OMNINET和ETHERNET局部地区网络。本书所取资料新颖、丰富，内容系统全面，构思严谨。书中列举的近百个程序实例，可便于读者学习和掌握有关内容。该书具有比较广泛的适用性，不仅对IBM PC机和国产长城0520系列机，而且对其他微型计算机的广大应用开发人员均有一定的参考价值，同时也可作为高等院校计算机及有关专业的教学用书或参考书。

微型计算机 IBM PC的原理与应用

（续　篇）

张福炎　周根林　李滨宇

滕小羽　蒋新儿　薛　行　　编著

责任编辑　丁　益

*

南京大学出版社出版

（南京大学校内）

江苏省阜宁印刷厂印刷

江苏省新华书店发行　各地新华书店经销

*

开本：787×1092　1/16　印张：31　字数：790（千字）

1985年9月第1版　　1988年2月第4次印刷

印数：67001—92000

ISBN 7—305—00178—3/TP·15　　定价：　7.30元

前　言

本书是《微型计算机IBM PC的原理与应用》（1984年11月版）的续篇，它仍以面向应用开发为宗旨，介绍有关IBM PC的各种常用软硬件的基本概念和技术。全书共分六章。第一章介绍IBM PC的汉字信息处理技术；第二章是8086/8088宏汇编，重点介绍在IBM PC机上使用宏汇编语言进行程序设计的一些常用方法；第三章介绍协处理器8087的硬件原理和在汇编语言及各种高级语言中的使用；第四章叙述C语言，它是一种效率较高、在应用开发中受到欢迎的高级语言；第五章是dBASE Ⅲ数据库管理系统，它的性能比dBASE Ⅱ已有明显提高；最后介绍以IBM PC组成的局部地区网络，详细地叙述了比较流行的OMNINET网和ETHERNET网（以太网）的硬件结构和软件原理。本书的内容具有比较广泛的适用性。它不仅对IBM PC机和国产长城0520系列机，而且对其他微型计算机的应用开发人员都有一定的参考价值。

虽然本书的许多内容与上册互有联系，但每一章的内容和结构实际上都具有独立性，适合于各类读者学习时的需要。

本书在编写过程中力求做到概念清楚、深入浅出。书中所列举的程序实例都已在机器上调试通过，可供读者在工作和学习时参考。

本书大部分章节的内容是经过集体讨论并且在机器上经过验证后编写的。其中，第二章的一至三节、第五章的一至四节以及第六章的三、四节由张福炎执笔；第一章的一至四节和第六章的一、二、五、六节由周根林执笔；第一章的第五节、第四章和附录Ⅰ、Ⅱ、Ⅲ由李滨宇执笔；第三章由滕小羽执笔；第五章的五、六节由蒋新儿执笔；第二章的四、五节由薛行执笔。全书由张福炎主持编写并最后修改定稿。

本书第一章中CCDOS部分系参考电子工业部六所的有关资料写成，并得到该所黄国健等同志的审阅，承蒙提出不少宝贵的意见。在编写过程中还得到了南京大学电子计算机厂、南京大学计算机科学系、江苏省建筑科学研究所电算室、江苏无线电厂、常州计算机厂、南通计算机厂、以及苏州计算机厂等许多单位和同志们的支持及帮助，编者在此谨向他们表示衷心的感谢。

由于时间仓促以及限于编者水平，书中错误和不妥之处在所难免，敬请读者不吝批评指正。

编　者

1985年7月

目　录

第一章　IBM PC汉字信息处理

我国是使用汉字的国家，电子计算机的推广应用对汉字信息处理的要求极为迫切，尤其是办公自动化、事务管理等数据处理领域，几乎完全离不开汉字信息的处理。

将IBM PC改造成为具有汉字处理功能的微型机系统，是推广应用IBM PC机所必须进行的工作之一。近年来我国已在这方面做出了不少成绩，如长城0520微型机系列以及汉字操作系统CCDOS就是一个很突出的例子。

本章主要介绍如何把IBM PC扩充成为汉字系统以及汉字操作系统的结构和使用，并通过实例说明在汉字操作系统支撑下，若干实用程序和高级语言程序中使用汉字的方法。

第一节　IBM PC汉字信息处理系统的构成

1．概述

微型计算机IBM PC是一个西文信息处理系统，它具备丰富的硬件资源和软件资源。为了充分有效地利用这些资源，在对IBM PC机进行汉字处理功能的扩充时，必须考虑以下几方面问题：

① 中西文兼容性　在扩充了汉字处理功能的同时，IBM PC机上运行的原西文处理软件应能继续使用，少修改或甚至不修改即可处理汉字信息。

② 友好的用户接口　具有便于用户掌握的多种汉字处理方法及交互式的操作方式。

③ 系统的可扩充性　当用户要求增加汉字输入方式或变更外围设备时，必须具备可扩充的能力。

④ 低成本　在改造系统时，用尽可能低的成本实现汉字处理功能。

扩充汉字处理功能是使IBM PC具有汉字输入、汉字显示、汉字打印以及汉字文件的存贮、传输等能力。为此，必须首先了解中国汉字与西文的特征差异所在。与西文相比，汉字字种多、字形复杂，形、音、义缺乏有机联系。所以，对汉字的处理远比西文处理困难。

汉字字种之多，在世界各国文字中占首位，属大字符集语种。根据对我国汉字使用频度的研究，可把汉字划分为高频字（约100个），常用字（约3000个），次常用字（约4000个），罕见字（约8000个）和死字（约45000个）。也就是说，正在使用的汉字字种达15000余个。根据我国1981年公布的《通讯用汉字字符集（基本集）及其交换码标准》GB2312-80方案，把高频字、常用字和次常用字归结为汉字基本字符集（共6763个字），再按出现的频度分为一级汉字3755个（按拼音排序）和二级汉字3008个（按部首排序），加上西文字母、数字、图形符号等700多个，如果再加上用户自行定义的专用汉字和符号等，那末一个适用的汉字系统应具有能处理多达8000余个汉字字符的能力。

图1-1是《通讯用汉字字符集（基本集）及其交换码标准》（GB2312-80）的示意

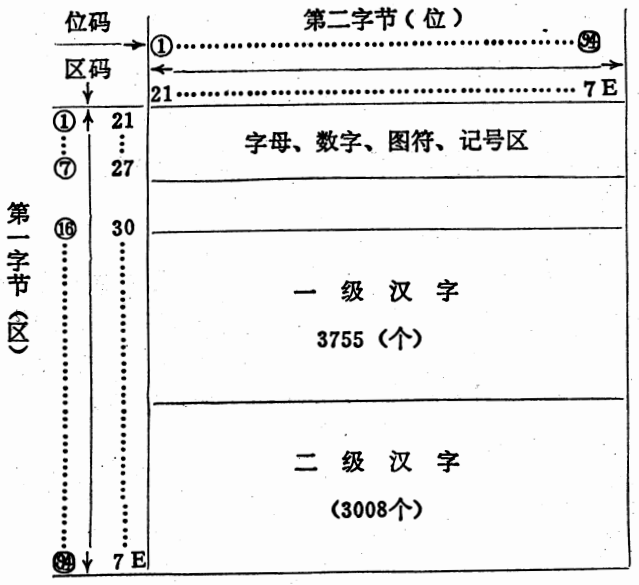


图1-1 国家标准（GB2312-80）汉字字符集

图。由图可见，代码表分成94个区，每区94位，区编号为第一字节，位编号为第二字节。因此，汉字必须由两个以上字节来表示，而西文字母只需一字节表示。为了保证中、西文的兼容性，一字节的西文代码应能与汉字字符相互有所区别。这就带来了汉字代码在机内表示的复杂性。

此外，汉字字形远比西文字母的字形复杂，它的笔画繁简不一，少至一笔一字，多至三十画一字，笔画方向及形状变化也多。因此在用计算机显示汉字时，通常是把单个汉字离散成网点，每点以一个二进位表示，这样就组成了该汉字的点阵式字模(见图1-2)。根据信息交换用汉字点阵字模的国家标准起草工作组的建议，三种规格的汉字点阵如表1-1所示。

	字节		字节	
......**.....	0	03H	1	00H
......**.....	2	03H	3	00H
......**.....	4	03H	5	00H
......**.....*..	6	03H	7	04H
**********.****.	8	FFH	9	FEH
......**.....	10	03H	11	00H
......**.....	12	03H	13	00H
......**.....	14	03H	15	00H
......**.....	16	03H	17	00H
......***.....	18	03H	19	80H
.....**..*.....	20	06H	21	40H
....**....*.....	22	0CH	23	20H
...**.....**....	24	18H	25	30H
...*.......**...	26	10H	27	18H
..*.........***.	28	20H	29	0EH
**...........*..	30	C0H	31	04H

图1-2 汉字点阵字模（16×16）

· 表1-1　汉字点阵字模分类

字　　型	点阵(列×行)	每个汉字占用字节	特　　征
简 易 型	16×16	32 (30)	字体骨架，一级字有笔锋
普 及 型	24×24	72	一、二级字有笔锋，仿宋体
提 高 型	32×32	128	仿宋体、黑体
精 密 型	48×48	288	同上，能表示更复杂字型

显然，为了实现近8000个汉字在IBM PC上的显示和打印，就必须为它配备一个庞大的汉字点阵字模库。

IBM PC汉字系统的构成通常如图1-3所示。图中汉字字模库、汉字打印机、汉字整字键盘等硬件，以及汉字处理软件（包括汉字输入、显示、打印模块）等均为处理汉字信息所必需的扩充部分。下面分别简要地介绍在IBM PC机上实现汉字输入、显示和打印的方法。

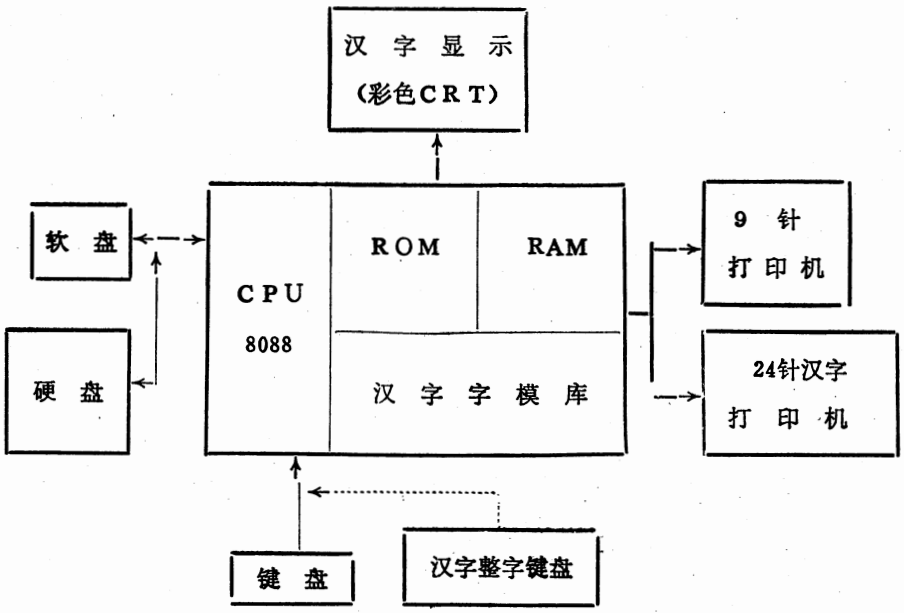


图1-3　IBM PC 汉字处理功能的扩充

2．汉字输入技术

在计算机系统上使用汉字，首先遇到的问题就是如何有效地把汉字输入机内。为了能直接使用西文键盘进行输入，就必须为汉字设计相应的编码，即用字母数字串来代替汉字。一种好的汉字输入编码应该具有下列特点：

* 易记忆，甚至不需记忆。
* 字母数字串尽可能地短，以加快输入速度。

* 编码与汉字的对应性好，尽量减少重码。

我国已研究了数百种编码方案，并从中优选出十余种，但大多数还不完全理想。当前，汉字输入的问题仍然是影响汉字系统普及应用的关键问题之一。

IBM PC机上已实现的汉字输入编码有多种，例如：

* 以GB2312-80为基准的国标码、国标区位码。
* 以发音为基础的拼音码。
* 以字形为参考的首尾码、拼形码。
* 以音、形结合为前提的声韵部形码。
* 在电讯业中已通用的电报码。
* 其它编码。

下面对上述编码的要点作些介绍。

（1）国标码和国标区位码。

国标码就是国家标准信息交换用汉字编码GB2312-80所规定的机器内部编码。每个汉字对应用4个十六进制数字来表示，在键盘上键入4次即可输入一个汉字，其优点是无重码，但难以记忆。

国标区位码是国标码的一种变形。它把国标汉字分为94区，其中1—15区是字母、数字、符号；16—87区为一、二级汉字，每区分94位。这样每个汉字就可用二—十进制区码和位码来表示，输入一个汉字仍需4键，这种做法虽然便于查找，但同样难以记忆。

图1-4是汉字“大”的国标码和国标区位码表示法。

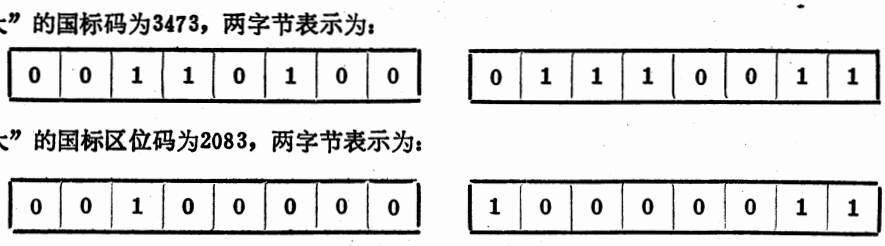


图1-4 国标码和国标区位码的表示

（2）拼音码

这是以文字改革委员会公布的汉语拼音方案为基础的输入编码，只要掌握汉语拼音便可以输入汉字，不需要记忆是其最重要的特点，因此人们乐于使用。但由于汉字同音字为数众多，拼音字母键入后还必须进行同音字选择（利用屏幕底部的提示行来进行），故输入速度稍低。

在汉字操作系统CCDOS下输入拼音码时，只要直接键入汉字的拼音码即可。为了减少按键次数，对某些经常联用的声母和韵母作了简化替代。例如，要键入汉字“中”，它的拼音为ZHONG，其中“ZH”简化为a，“ONG”简化为s。输入时，只要键入as即可。但用户必须记忆这些简化规则，所以在一定程度上也影响了拼音码的推广使用。

（3）首尾码

这是对汉字字形（部首）进行简化后规定的编码，编码的记忆量少，使用较方便。

将汉字的左上部笔画约定为字首码，右下部笔画约定为字尾码。分首尾的原则是先左右、后高低，不分笔画顺序；对于内外形汉字，取外形为字首，内形为字尾。例如：

琉：字首码为“王”，字尾码为“儿”

田：字首码为“囟”，字尾码为“十”

因此，只要按两次键，即可输入一个汉字。然而，用以上方法表示的汉字仍然有重码，还要通过选择的方法确定所需输入的是哪一个汉字。此外，也可在首尾码后再添加一个“首音”码，即该汉语拼音的第1个字母，如“琉”为L，“田”为T。这样就可使重码率大大减少，提高了输入的效率。

首尾码的笔形与输入键盘的对应关系亦比较复杂，不易记忆，一般应在字母数字键上刻上相应的笔形，组成便于首尾码输入的键盘。

（4）声韵部形码

这是根据汉语拼音和字形结构两个因素所规定的汉字编码。编码规则虽比较复杂，需要记忆，但是无重码。常用字只需1—2键即可完成输入，一般汉字也只要输入4键，总的输入效率较高，适合于专业操作人员使用。

图1-5是声韵部形码的编码方法示意图。每个汉字按声母、韵母、部首分类码、起笔码四个部分各一个字母组成，对应的码可由声韵码表、部首分类表以及起笔对照表来确定。

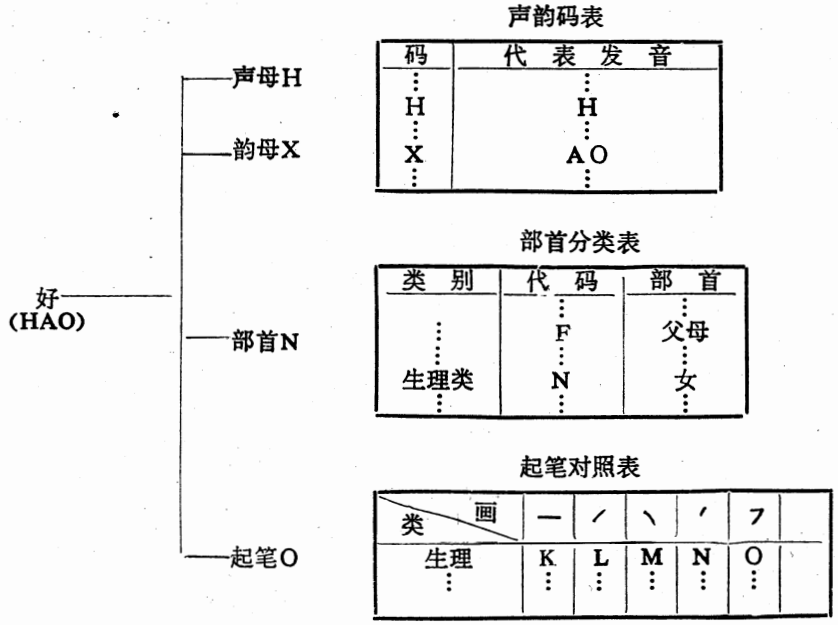


图1-5 声韵部形码编码原理

例如，汉字“好”的拼音为HAO，查声韵码表可知，声母为H，韵母“AO”对应字母键X，“好”的部首“女”属生理类，查部首分类表得N，“好”的部首起笔为“㇀”，查起笔对照表得代码为O，这样“好”的声韵部形码为HXNO。

为了提高输入效率，声韵部形码对常用字的编码作了简化。如“我”的编码为W，“的”的编码为D等等，因而敲一个键即可完成某些常用汉字的输入。

（5）电报码

这是把邮电系统已广为使用的电报明码直接作为汉字输入的方式，每个汉字用4位数字表示，这对于邮电部门专业人员是极为适宜的。

（6）汉字整字键盘输入

上述汉字编码输入方法是直接使用微型机的字母、数字键盘（也称“ASCII键盘”）进行输入操作的。但由于编码法大多需要记忆，难以为一般的用户所接受。为此，有时还加配一个专门用于汉字输入的整字键盘（俗称“汉字大键盘”），作为汉字的辅助输入设备。

汉字整字键盘是一种专用的外部设备。按盘面汉字字数可分为2K键盘（两千字左右）和4K键盘（四千字左右）两种。汉字排列的标准化工作目前正在进行。图1-6为汉字键盘的结构原理。它主要由盘面输入设备、控制器及代码变换等部分所构成。盘面设备用来将所选定的汉字在盘面上的位置信息送入控制器。键盘开关以矩阵形式组成，信息形成可以有多种方法，如压感式、电容感应式及电磁感应式等，其中以直接借助于按键时的压力，将键盘开关矩阵纵横线交叉点接通的方式最为简便。键盘控制器是一个单片微型计算机，通常为4位机，它对键盘开关矩阵不断地进行扫描，把检测得到的位置码送入存有汉字国标码的只读存贮器ROM，ROM输出两字节汉字代码，它在控制器的控制下以并行形式（一次一个字节，分两次）或以串行形式输入到主机中去。

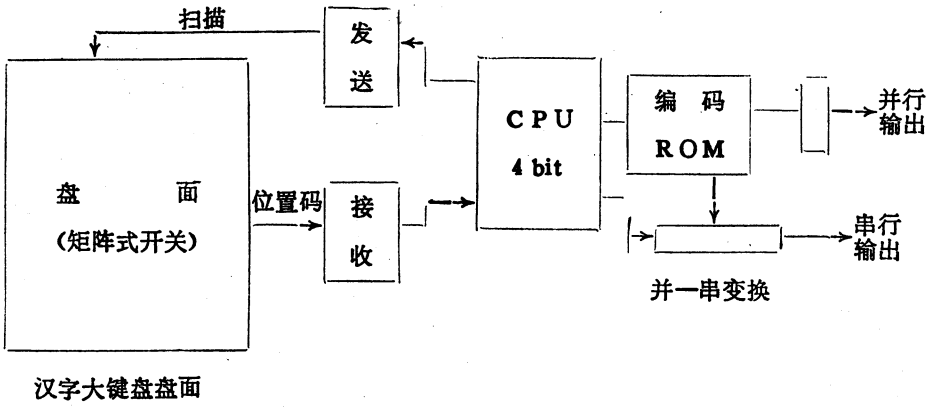


图1-6 汉字整字键盘原理

汉字整字键盘允许一键一个汉字，操作极为简便、直观，无需记忆汉字的编码，但由于盘面字数很多，非熟练人员查找比较困难。不过，平常在输入少量信息时，整字键盘还是相当方便的。

汉字整字键盘可与IBM PC的一个串行口相接，对PC机的原键盘输入程序作简单修改之后即可完成其连接。

（7）汉字识别

随着技术的发展，更为理想的汉字输入设备，是利用汉字识别技术能直接阅读汉字的汉字OCR（汉字光学字符阅读器）。目前，印刷体汉字的阅读设备已实用化，手写体汉字的阅

读装置正在开发。由于这些设备价格昂贵，一般不与 IBM PC之类的微型机连接使用。

3．汉字显示与汉字字模库

（1）IBM PC的汉字显示

汉字显示的实现是微机汉字系统中的重要环节，也是技术上较为复杂的一个部分。但使用 IBM PC的彩色图形显示控制卡和彩色显示器即可较为简单地实现汉字显示的功能。其工作原理如图1－7所示。

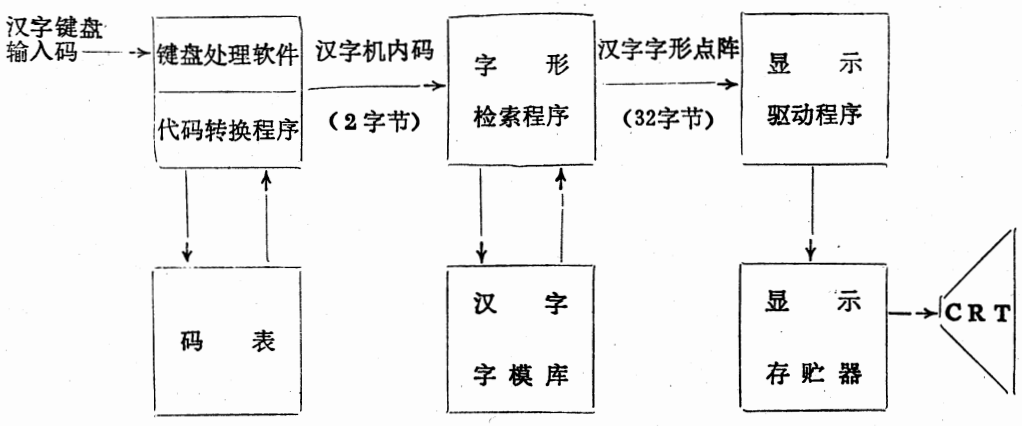


图1-7 IBM PC 汉字显示原理

从键盘进入主机的各种汉字编码，首先由键盘处理软件变换成机内统一的汉字内码，并保存在主存贮器的显示文本区内。然而，两字节的内码并不是该汉字本身的字形点阵。前面已经说明，一个汉字的字形通常要用32个字节的二进制数据来表示，所以还必须根据内码向汉字字模库中检索出该汉字的点阵数据。接着，在显示驱动程序的控制下，这些点阵数据被送到位于彩色图形显示控制卡上的显示存贮器中，它的每一位二进位数据与CRT显示器屏幕上的一个点相对应。这样，在CRTC的控制下，显示存贮器中的点阵数据顺次整屏读出，每秒重复60次，于是就可以在CRT显示器上稳定地看到该汉字了。

本书的上册已介绍了IBM PC彩色图形显示卡的工作原理，比较一下就可以发现，上述的汉字显示过程，实际上与光栅图形显示的过程是类似的，不同之处仅仅是显示存贮器中的数据不是图形而是汉字字模而已。所以这种汉字显示方式又称为图形型汉字显示方式。

（2）汉字字模库与汉卡

必须指出，IBM PC汉字显示的屏幕特性直接取决于原来机器的图形显示能力。由于屏幕上显示一行汉字需占用16行扫描线，加上行间距占2行，共占18行。而IBM PC在图形模式时最多只能允许显示200行，故一共只能显示11行汉字。同样，IBM PC图形显示的水平分辨率为640点，一个汉字横向为15点，加1点作字间距，故一行显示的汉字数为40个。这样，满屏汉字总数为40字/行×11行＝440字。

在320×200中分辨率的彩色图形模式下，满屏可显示的汉字数为：

20字／行×11行＝220字

当然，这种显示特性并不理想，一是因为满屏只有440个汉字，显示的信息量太少；二是行数只能是西文显示的一半，不少西文处理软件（如表格软件等）在改为用中文方式运行时将面目全非。

如果把IBM PC的彩色显示卡改为更高分辨率的显示卡，例如分辨率为640×400，就可以实现40字/行×20行（有行间距）或40字/行×25行（无行间距，但与西文行数相同）的显示，这就使汉字屏幕处理效果大大改善。这些高分辨率显示卡的主要做法是把显示存贮器由原来16KB扩充到64KB甚至128KB，再配用性能更好的监视器即可。这样做，IBM PC的汉字处理性能将得到显著增强，从而更加受到用户的欢迎。

与IBM PC兼容的国产长城0520C，对汉字显示功能作了较好的改进。它使用了分辨率为640×450、有八种颜色的彩色/图形显示卡，并配之以高分辨率彩色监视器，实现了每屏28行×40字的汉字显示能力。其中25行为文本行，其余3行为汉字提示行，因而使用性能大为改善。

图1-7中的字模库是驻留在主存贮器中的。由于MOS存贮器芯片所保存的信息在断电后即自行消失，因此在每次开机加电时，需要从软盘或硬盘上把字模库加载到主存贮器的字模库区域中去。这样的汉字字模库称为软字库。软字库的实现比较简单，不需要对原IBM PC机作任何硬件方面的改动，是一种颇受欢迎的字模库建立方式。

显然，由于汉字字模库容量很大（8000个16×16汉字的字模库约需占用256KB主存贮区），在进入汉字工作模式时，需化费一定的字库加载时间，用户可使用的存贮空间也大为减小。因此，用软字库工作的IBM PC机，要求主存贮器容量大于384KB，一般要求有512KB的主存，才能提供有效的汉字作业区域。字模库在主存贮器空间的位置如图1-8所示，其中VRAM为显示存贮器区域。

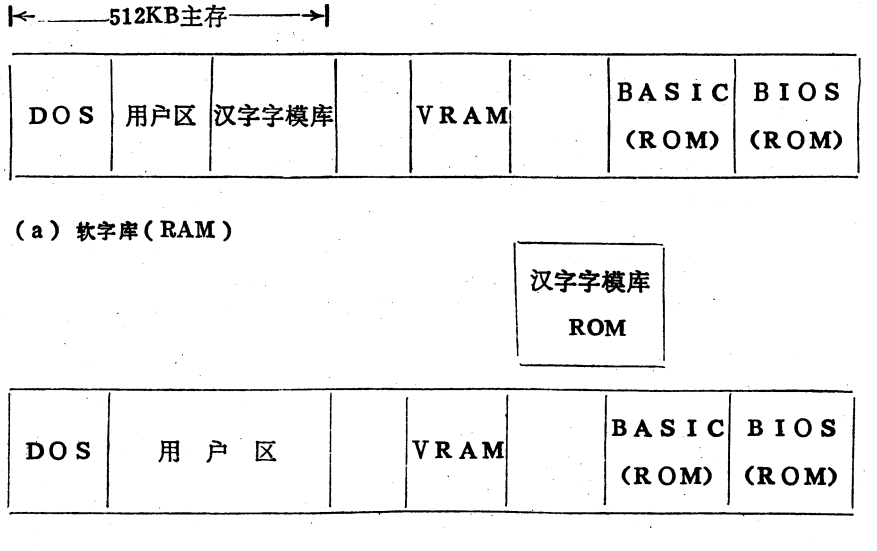


图1-8 字库所占存贮器空间的位置

使用软字库方案虽然比较灵活，但毕竟不太方便，而且字库在内存中还占据了随机存贮器空间，另外随机存贮器比只读存贮器也要贵一些，因此，实现汉字库的另一种做法是使用

EPROM或Mask-ROM芯片制成汉字字模库卡（俗称汉卡或中文卡），把它插入IBM PC的扩充槽中，即可代替软方案中RAM字库的工作，这就是硬字库方案。由于硬字库容量较大，或者为了不占用主存空间，因此通常把它安排在另一个存贮空间内（图5-8(b)）。

随着大规模集成电路技术的发展，ROM芯片的集成度大大提高，价格也相应降低。硬字库的方案显然比软字库更吸引人。

通常，硬字库采用EPROM电路写入汉字字模。ROM芯片的选择是制作汉卡的关键。当前可采用的EPROM芯片及用其组成汉卡（整字点阵字库方式）时所需的用量如表1-2所示。

表1-2　硬字库所需的EPROM芯片数量

电路型号	单片容量	一级汉字库	一级加二级汉字库
2764	8KB	16片	32片
27128	16KB	8片	16片
27256	32KB	4片	8片

在半导体电路制作过程中，直接将汉字点阵做在ROM芯片内的制品称为Mask-ROM，它比EPROM价格低，适合于大量应用的场合。不久，汉字Mask-ROM将会大量地为汉字系统所采用。

汉卡的电路结构比较简单，其功能仅是为IBM PC扩充一个ROM存贮区，汉卡的控制电路用来完成I/O扩充总线接口逻辑、控制寻址、读出和变换。汉卡上还有ROM字库存贮体、地址译码驱动和读出缓冲电路等，其典型结构如图1-9所示。

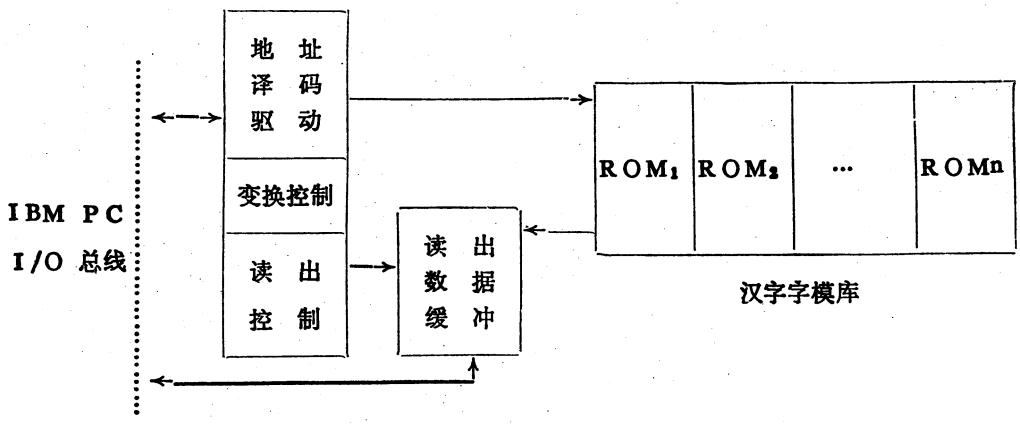


图1-9　汉卡的逻辑结构

实际上，在汉字显示过程中，还涉及到汉字的属性、颜色、光标显示等问题，这些将在后面再作介绍。

4. 汉字打印

汉字信息处理的另一个环节是在打印机上输出包括有汉字在内的处理结果。由于用户对

汉字输出文本质量的要求不同，再加上系统可配接的打印机种类繁多、性能不一，所以汉字打印输出的实现是比较复杂的。

在IBM PC机上实现汉字打印输出不需要增加或修改硬件，可以直接利用机器上原有的打印接口板配接合适的打印机。但是，所配接的打印机必须具备图形打印功能，最理想的是直接配接汉字打印机。

（1）汉字打印原理

汉字打印输出的工作原理可用图1-10来表示。需要打印的汉字文本预先送入主存的打印缓冲区。打印输出时，再从缓冲区逐个取出汉字内码（两字节），根据汉字内码，检索程序从汉字字模库中取出汉字点阵数据，放入主存的字模缓冲区内。IBM PC/XT供打印用的汉字字模库有两种：一种是与汉字显示公用的（16×16）点阵汉字字模库，另一种是放置在硬盘中的（24×24）点阵汉字字模库。后者能使正式文本打印时获得高质量的打印结果。

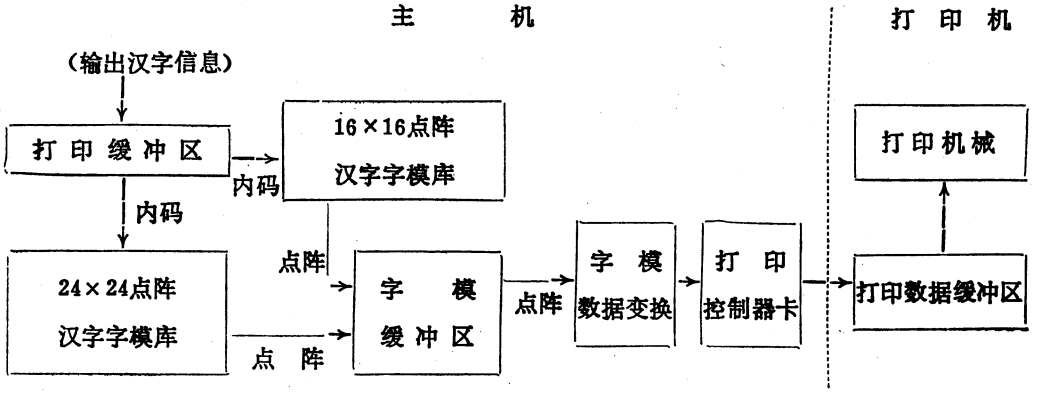
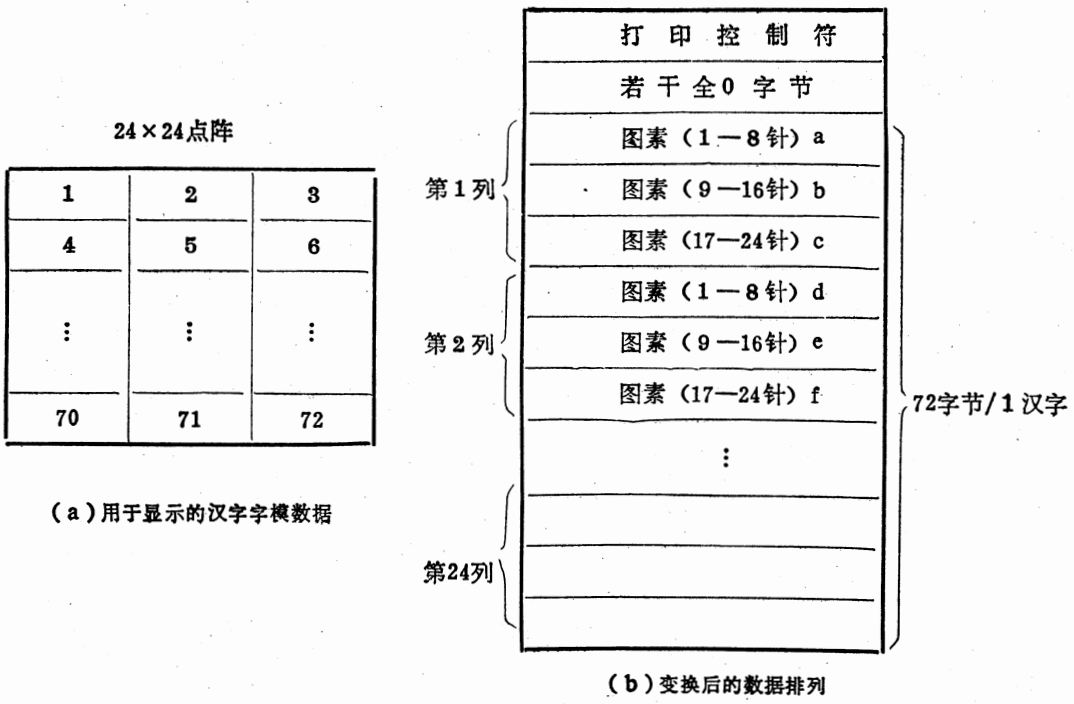


图1-10　汉字打印输出的工作原理

字模缓冲区中存放的字模数据并不能直接送出打印，这是因为汉字字模的点阵数据是横向排列的，而提供给打印机针头的信息却需要按纵向排列。所以必须再对字模数据进行变换，组织成打印机按图形方式工作时所需要的数据格式，这个任务是由字模数据变换程序来完成的。经过变换后的字模数据，再通过IBM PC机的打印卡送至打印机去。打印机内有一个打印数据缓冲器，汉字字模数据在这里依次存放，直到一行打印信息全部到齐，再启动打印机开始打印。此时，打印头横向移动，变换后的字模数据驱动打印针电磁铁，于是纵向排列的打印针头就击打色带，从而在纸上印出汉字。

（2）字模数据变换

图1-11是24×24汉字字模数据变换的示意图。若字模库中每个汉字的字模数据按字节排列为1，2，3，4，…，72（这是为适应汉字显示方式而安排的），进行字模数据变换时，把第1，4，7，…，22共八个字节的第1位组成字节a；把第25，28，31，…，46共八个字节的第1位组成字节b；把第49，52，55，…，70共八个字节的第1位组成字节c。这样得到的a、b、c为第1列的24点数据，d、e、f为第2列数据，把这72个字节重新在存贮器中存放好，再加上打印控制符以及必须的字间距信息（全“0”字节），就完成了字模数据的变换。



（a）用于显示的汉字字模数据

（b）变换后的数据排列

图1-11 字模数据的变换（24×24汉字）

按照变换后的格式，打印控制器就能控制打印机构打印出汉字。当然，不同的打印机所要求的格式是不同的，因此变换要求也就不同，这一点必须注意。

南京大学计算机科学系 （正常体）

南京大学计算机科学系 （扁体）

南京大学计算机科学系 （长体）

南京大学计算机科学系 （大号）

南京大学计算机科学系 （加浓正常体）

南京大学计算机科学系 （加浓扁体）

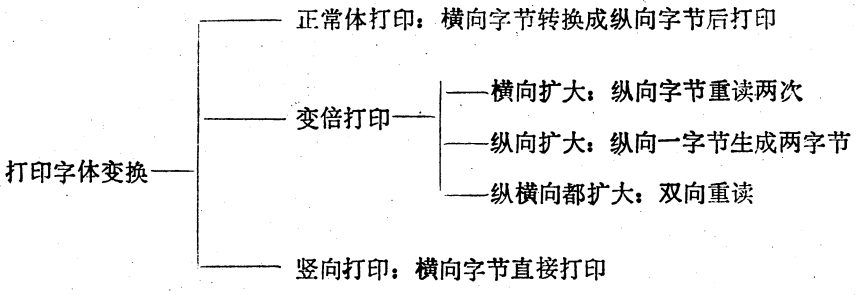
南京大学计算机科学系 （加浓长体）

南京大学计算机科学系 （加浓大号）

图1-12 打印字体的变化

在实际使用中，常常要求汉字打印输出具有各种大小字体以及横向和竖向打印等变化，这些变化都可以通过适当的字模数据转换来实现。例如，24×24点阵的汉字，经变换后可生成正常体（24×24）、扁体（24×48）、长体（48×24）和大号（48×48）以及加浓（竖线条加宽）和竖体(正常体转动90°后的字形)等各种方式（图1-12）。同样，对16×16点阵的汉字，也存在（16×16）、(32×16)、（16×32）、（32×32）等字体尺寸以及横打、竖打等变化。这样就为用户提供了较为丰富的打印能力，以便生成更加适用的文件和表格。

由上述可知，实现字体尺寸以及横打、竖打等变化的方法是进行字模数据变换，变换的方法都是相似的。现将它们归纳如下：



（3）汉字打印机

可以配接IBM PC进行汉字打印的打印机有多种，根据打印头上打印针的多少，可分为9针、16/18针、24针三种。

9针打印机如FX-100、CP-80、IBM图形打印机等。由于一次只能打9个点，对16×16汉字的打印必须分两次才能完成（第1次打一行汉字的上半部分8个点，第2次再打该行的下半部分8个点）。这种打印机打印速度慢，打出的汉字字形不太美观，但是它造价低，又是西文系统的标准设备，因此在要求不高的场合下还是可以使用的。

16针打印机如SM-16P，18针如FT-8000，是一种适合打印16×16点阵汉字的打印机。一行汉字一次打成，字形好，造价不高，对一般汉字处理用户来说，还是一种比较实用的设备。

24针打印机如NM-9400，TH3070，M-2024，M-1570等，是一次打印出24×24点阵的汉字打印机。IBM PC汉字系统最好能配接24针打印机，因为它可完成上面所说的各种字体的打印，适合多种汉字应用领域，价格适中，打印速度也较快。

表1-3列出了一些常用汉字打印机的性能。

必须着重指出的是，汉字打印机已向着自带汉字字模库的方向发展，上述各种打印机内部均可装入由ROM芯片构成的汉字字模库。由于这样，汉字打印的功能及速度将大大增加，打印过程也变得极其简单。对IBM PC来说，只需将要打印的汉字内码（两字节一个汉字）直接送往汉字打印机，一切字模点阵的变换和控制都可由打印控制器自行完成。主机只要把打印机规定的控制码（“ESC”控制序列）送给打印控制器，就可以完成各种变倍、横向或竖向打印，以及加浓（即重打）等操作要求，充分发挥了打印机本身的功能。由于主机与打印机之间只须要传送汉字内码，而不要传送字模信息，所以大大节省了打印数据的传送时间，提高了系统的工作效率。

表1-3　常用汉字打印机性能

性能 \ 型号	NEC PC-PR-201-X	东芝TH-3070	Brother M-2024	EPSON LQ-1500 UP-130K
打印速度：				
汉　字	50CPS	35CPS	40／60CPS	45CPS
汉字(压缩)	100CPS	70CPS	—	—.
西文（高密度）	73CPS	—	—	—
西文（高速）	146CPS	93CPS	120／164CPS	—
西文（超高速）	220CPS	—	—	200CPS
彩色打印	可	不可	不　可	不　可
行　宽	西文136字符/行 汉字96字符/行	同　左	同　左	同　左
拷贝能力	4　张	3　张	5　张	4　张
汉字字库	1、2级汉字	可选1级	可选1级	1　级

综上所述，由IBM PC扩充而成的汉字信息处理系统，其必不可少的硬件和软件成分有：

* 具有图形显示功能的显示控制卡和监视器。
* 具有图形打印能力的打印机或汉字打印机。
* 尽可能大的用来存放汉字字模库的内存贮器（RAM或ROM）。
* 能支持汉字输入、汉字显示、汉字打印及代码转换的有关软件。

第二节　汉字操作系统CCDOS的原理

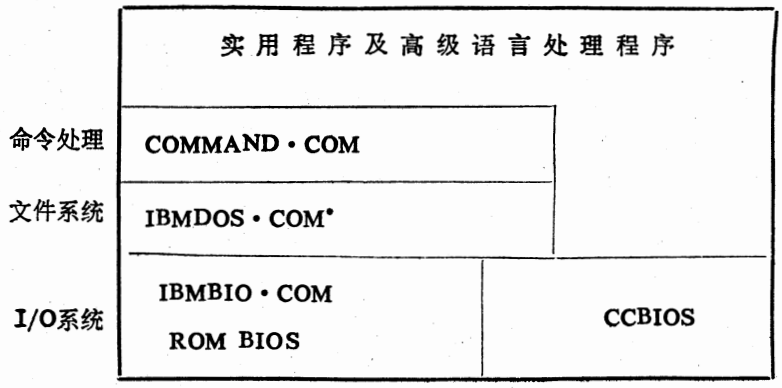
电子工业部第六研究所为IBM PC开发的汉字操作系统CCDOS是在MS-DOS的基础上，扩充了汉字处理功能而实现的。与MS-DOS的版本相对应，CCDOS也有1.1版、2.0/2.1版和3.0版之分。本节主要介绍CCDOS第2版的结构与原理。

1. CCDOS的系统结构

汉字操作系统的功能与原西文操作系统类似，主要是进行汉字文件管理和汉字设备管理，设备管理中还要解决汉字的输入、显示、打印、传输等功能。

（1）CCDOS结构

CCDOS是在MS-DOS的基础上，对其中文件管理系统（IBMDOS·COM）和基本输入输出系统（BIOS）扩充了汉字功能而成的。所以CCDOS亦是层次结构，如图1-13所示。



（ *装入内存后有修改和扩充 ）

图1-13 CCDOS的层次结构

本书上册第二章已经提及，ROM BIOS存放在主机板上的8 KB EPROM芯片中，它控制着系统所必需的主要外部设备的工作。用户软件在使用键盘输入、CRT显示及磁盘操作时，频繁地调用ROM BIOS中的各个设备驱动模块。因此，为了响应对汉字外部设备的调用要求，就必须在8 KB的ROM BIOS之外，再添加用以进行汉字键盘输入、汉字显示、汉字打印等操作的驱动模块。这部分软件在编写中尽量利用了原来的ROM BIOS程序，使之更为简练。扩充部分称为CCBIOS，是CCDOS最核心的部分。而对IBMDOS·COM部分的修改，只限于一些字符处理程序和引导程序的适配，但这也是汉字内码能在原西文操作系统上通行的关键之处。

如同第一节所述，为了适应汉字系统各组成部份对汉字信息处理的不同要求，操作系统必须对如下各种汉字代码作变换处理：

* 输入码　　这是用户在键盘上敲入的各种汉字编码，又称外码。
* 内码　　系统中处理、存贮汉字信息均使用统一的内码，它也包括西文ASCII码。
* 字模点阵码　对16×16点阵汉字为32字节码，24×24点阵汉字为72字节码，ASCII字符的点阵为（8×8），占八个字节。字模码用以显示和打印汉字。
* 打印字模码　这是针对打印机对打印图形汉字的要求，从字模码变换而来的。
* 传输码　　数据在多台微机之间传送时，使用的汉字代码。

CCDOS所提供的各种模块，可以用来完成上述代码的变换和处理。图1-14是这些功能模块的工作情况和相互关系。其中键盘管理模块、显示管理模块、字库管理模块和打印管理模块均以文件形式存放在磁盘上。在CCDOS运行时，它们与原BIOS一起，就组成了CCBIOS。

（2）内码的选择

内码是系统中用来表示西文或中文信息的代码。在IBM PC原西文系统中，每个西文字符的机内码是采用一个字节来表示的ASCII码，一般只使用前七位表示128种字符。汉字系统中的机内码的选取必须考虑到：

* 不能产生二义性，即ASCII码和汉字内码应严格区分。
* 汉字机内码的长度应顾及显示体制和打印体制，并应尽可能短。

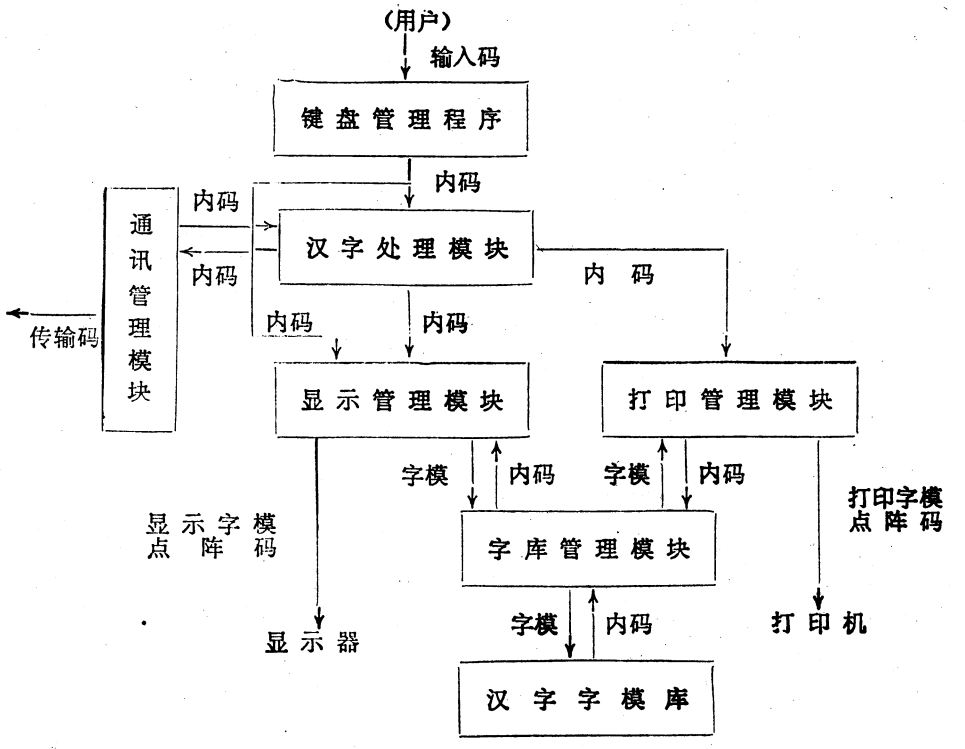


图1-14 CCDOS的功能模块与汉字代码转换

* 应与国标GB2312-80汉字字符集有尽量简单的对应关系，以便于对汉字库的处理、查找。

据此，直接使用GB2312-80规定的国标码是不行的，因为国标码规定，组成每个汉字的两个字节代码为：

0	×	×	×	×	×	×	×
7	6	5	4	3	2	1	0

0	×	×	×	×	×	×	×
7	6	5	4	3	2	1	0

它们每个字节的高位均为“0”。这样，当ASCII码和国标码同时存在时，国际码往往被误认为是两个独立的ASCII码，从而产生二义性。

CCDOS使用的汉字机内码是一种高位为“1”的两字节内码，此方案是将GB2312-80规定的国标码的每个字节中最高位置“1”，作为汉字机内码。以汉字“大”为例：

国标码 3473	0	0	1	1	0	1	0	0
	7	6	5	4	3	2	1	0
机内码 B4F3	1	0	1	1	0	1	0	0
	7	6	5	4	3	2	1	0

0	1	1	1	0	0	1	1
7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1
7	6	5	4	3	2	1	0

这是一种广泛使用的内码。由于IBM PC绝大部分软件都允许使用高位为“1”的字节，所以这些软件很易在CCDOS支持下获得汉字处理功能。这种机内码表示方法，只用两个字节代表一个汉字，与国标码有着极简单的对应关系。两字节代码与汉字显示特性和打印特性又是互相匹配的，因为显示与打印汉字时，一个汉字恰好占据两个西文字符的位置。这就使得显示和打印中西文文件的编辑工作比较容易进行。

但在某些场合下，例如不同计算机之间进行信息传送时，或者在运行某些特殊的软件时，往往一个字节的最高位用作校验位，或不允许该位为“1”，这时，高位为“1”的方案就不适用了。

不同的汉字操作系统可选用不同的机内码，它们各有特色和局限性。随着CCDOS的发展，在IBM PC机中可能会引入并且使用新的功能更强的内码。

（3）CCDOS的存贮器布局

CCDOS装入内存后，其信息布局必须考虑到原MS-DOS的内存分配。装入内存的CCDOS分为程序区与数据区两部分。程序区主要是CCBIOS，即对汉字设备的管理程序，它用来代替部分原来位于ROM中的设备驱动程序；数据区是汉字字模库和一些输入码转换表等。为了保证原MS-DOS的工作不受影响，只能将CCDOS置于原用户区的底层。CCDOS在存贮器中的信息布局如图1-15所示。

0000：0000	ROM BIOS 的中断向量表 （包括CCBIOS的各处理模块中断入口）
0040：0000	ROMBIOS 数据区
0050：0000	DOS 与 ROMBIOS 的通讯区
0060：0000	DOS BIOS 程序区
00BF：0000	DOS 程序区
	用 户 程 序 区
•4E69：0000	DOS命令解释程序覆盖区
•4E69：1848	CCDOS各个处理模块（程序区）
•4E69：2B98	CCDOS 数据区
•	汉字字模区
	未安装的 RAM
B000：0000	单色显示存贮区
B800：0000	彩色显示存贮区
C000：0000	ROM 区 （未装配）
F600：0000	ROM BASIC 区
FE00：0000	ROM BIOS 区

•地址供参考

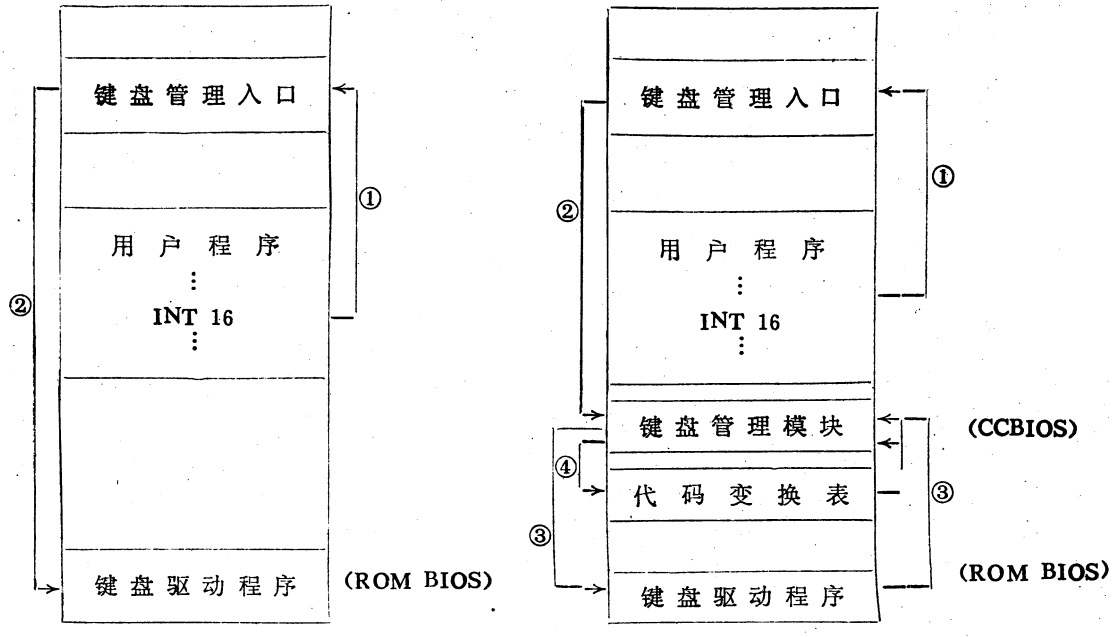
图1-15 CCDOS在存贮器中的信息布局

IBM PC机的系统软件或应用软件在进行输入输出操作时，最终总是通过软件中断调用ROM BIOS中的有关驱动程序来实现的。其中与汉字处理有关的软件中断主要有：

* INT 10H　显示器驱动程序
* INT 5H　屏幕打印程序
* INT 16H　键盘输入驱动程序
* INT 17H　打印机输出驱动程序

在CCDOS的控制之下，任何程序调用上列中断时，就不再象MS-DOS工作时那样，去访问ROM BIOS，而是改成访问驻在内存的CCBIOS中的有关模块。这种改变是通过CCDOS装入完毕后立即修改位于内存最上端的中断向量表来实现的。

图1-16是在MS-DOS下输入一个字符以及在CCDOS下输入一个汉字的调用过程示意图。虽然都是使用INT16，但在MS-DOS的情况下，中断向量表使访问直接指向ROM BIOS中的键盘驱动程序，完成一个字符的输入。而在CCDOS的情况下，修改了的中断向量表使中断访问先指向CCBIOS的键盘管理模块。这时，若输入为西文方式，则仍由ROM BIOS中的键盘驱动程序完成。若进行汉字输入，则需要从键盘上接收多个字符才能表示一个汉字。每接收一个字符，CCBIOS的键盘管理模块就要去调用一次ROM BIOS的键盘驱动程序，直至代表该汉字的字符全部接收完毕，然后再由键盘管理模块进行输入码到内码的变换，从而最终实现一个汉字的输入。可见汉字输入的过程要比西文复杂得多。



(a) MS-DOS下输入一个字符　　(b) CCDOS下输入一个汉字

图1-16　MS-DOS和CCDOS下键盘输入过程

2. 键盘管理模块

(1) 工作流程

键盘管理模块的功能主要是从键盘上输入汉字并把它转换成为其内码。CCDOS为用户提供了多种汉字输入编码方法，根据它们有无重码来分类，一般可分为重码类编码（拼音码，首尾码等）和无重码类编码（国标区位码，电报码，声韵部形码，见字识码等等），而按检索的方法不同又可分为计算检索编码（国标区位码等）和查表检索编码（电报码，拼音码，拼形码等）。

整个键盘管理模块由两部分组成。其一是设备管理，这部分扩充了ROM BIOS中键盘驱动程序的相应功能；其二是代码处理，包括输入模式的切换、人机对话管理和各种输入码转换模块等。

图1-17是在汉字输入模式下，键盘管理模块的大致工作过程。当从键盘上键入一个字符时，键盘管理模块首先判别它是否汉字代码。若是汉字码，立即转入代码处理程序，并将输入的字符送入汉字缓冲区，继续接收后面的代码，等该汉字的全部代码输入完毕后，再根据输入码的类型，按相应的方法查找出汉字内码，并送入主机进行处理，这时，若为无重码类编码，则将找到的汉字显示在屏幕的光标所在处；若为重码类编码，则在提示行处显示出一组重码汉字，用人机对话进行选择，然后送屏幕显示。

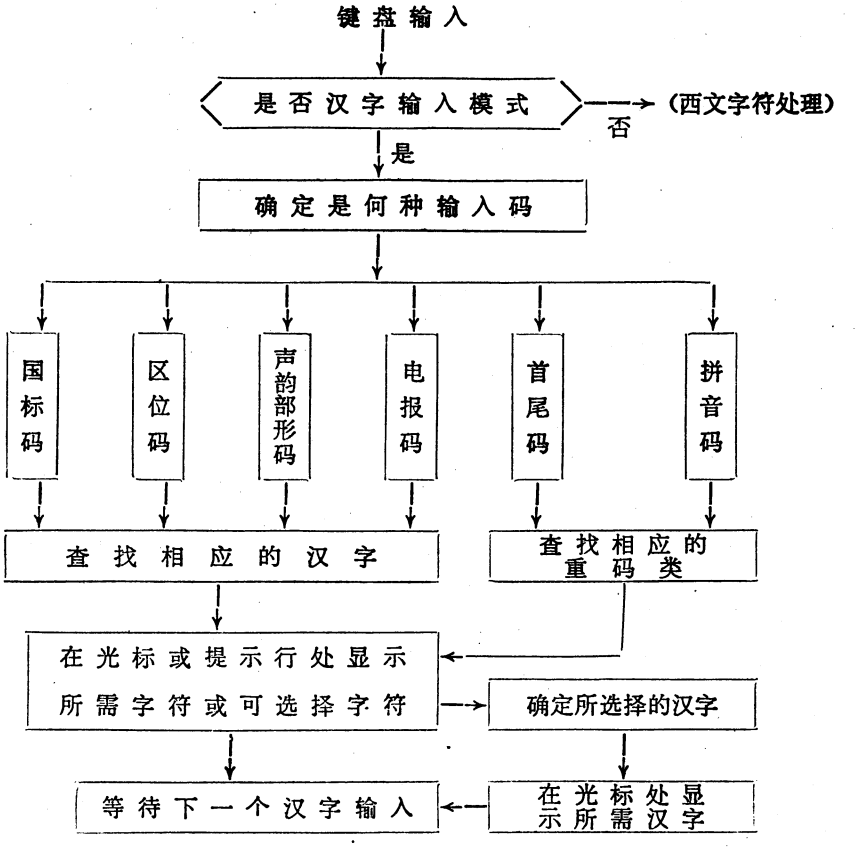


图1-17 键盘管理模块的工作流程

（2）代码识别

整个汉字输入过程可区别为代码识别和代码转换两部分。代码识别是一个关键，其操作

流程如图1-18所示。

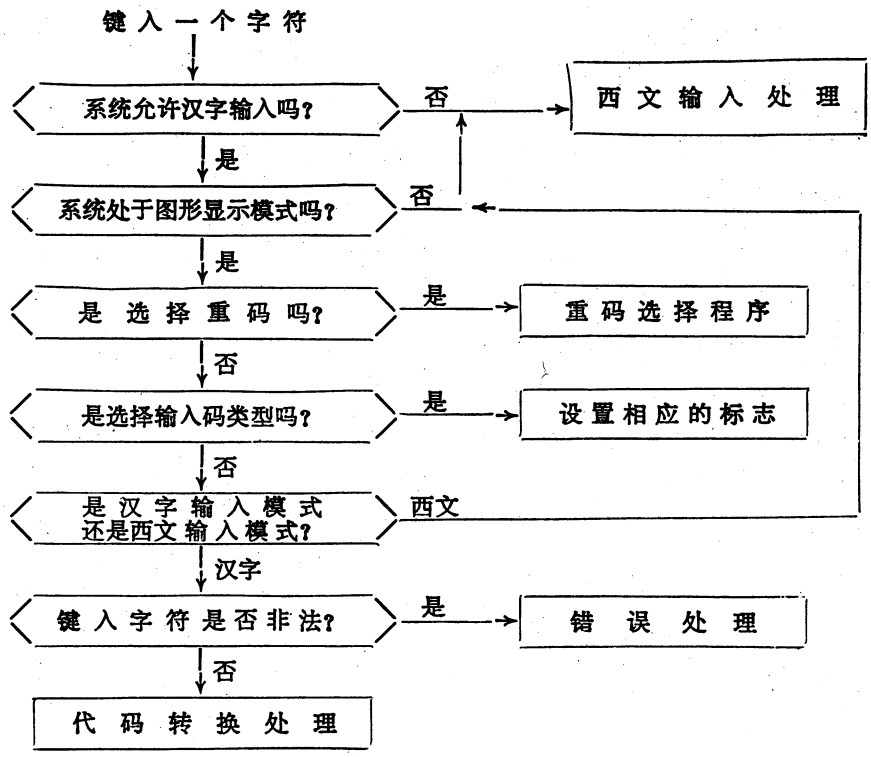


图1-18 代码识别程序的工作流程

每当在键盘上输入一个字符后，键盘管理程序将输入字符送入AX寄存器保存。然后，代码识别程序先判断系统是否允许使用汉字，再判断系统是否处于图形显示方式，这样才能确保汉字处理的正确进行。接着，代码识别程序判断这个字符是汉字的输入码，还是重码选择过程中的选择号。若是后者，则转向重码选择程序；若不是，则再判断是否为选择输入码类型，若是就转去置相应的标志，若不是才可确定当前键盘输入的字符确是汉字代码。至此，代码识别工作方告结束，接着进入代码转换程序。

（3）代码转换

对每一种汉字输入码方案，均对应有一个代码转换程序把它们转换成为统一的机内码。

① 国标码：将接收的两个字节均在最高位加“1”，即得到内码。

② 国标区位码：先将区位码转换成国标码，再按国标码处理。转换成国标码的方法是先将十进制的区码和位码转换为十六进制，然后两个字节分别加上80H。例如，“重”字区位码为1702，变换成十六进制表示为1102H。于是

国标码为：2020H＋1102H＝3122H

内码为：　8080H＋3122H＝B1A2H

③ 声韵部形码：这是通过查表法完成代码转换的。输入码按相应汉字的国标顺序排列成表，将键盘输入代码与表中数据作顺序比较，并统计比较的次数，相符时，把累计次数（即

输入码在表中的位置）作适当的计算即可得到内码。例如，“重”的声韵部形码为BXOC，查表得到它的位置是96，用96／94得商为1（每区94个字），余数为2；表示它是国标码第16＋1区的第2个字，即国标区位码为1702，再按第（2）小段中的方法求得其机内码为B1A2H。

④ 电报码：方法与声韵部形码相同。

⑤ 首尾码：这是有重码出现的一种输入码，其代码转换过程如图1-19所示。首尾码允许键入首码和尾码后进行选字，也允许在键入首音码后进行选字。

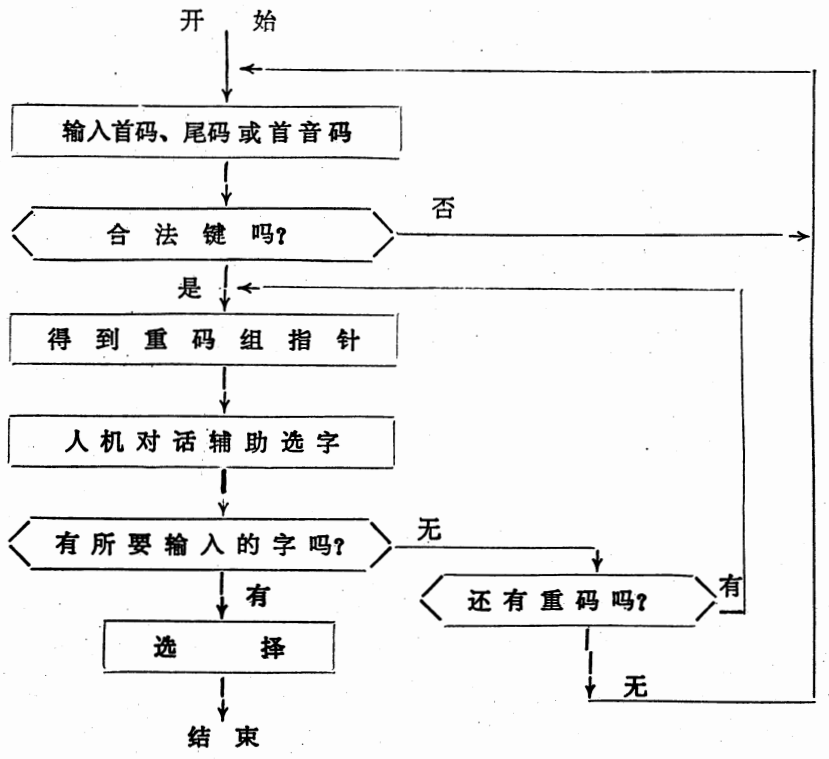


图1-19 首尾码的代码转换过程

首尾码的检索表是按三级检索的要求编排的。第1级为首尾检索表，共36个字节，分别代表首尾码所用的各键；第2级是首尾重码组指针检索表，表的每项为两个字节，其中存放各重码组的相对指针(即相对于三级表头的偏移量)；第3级是首音检索表，每项一个字节存放首音字符，每个可查到的汉字都有其对应项，还有一个与三级表对应的三级国标表，每项2个字节，存放相应的国标码，检索过程如图1-20所示。

⑥ 拼音码：这也是一种有重码出现的输入码，但检索表与首尾码不同，因为国标一级汉字是按拼音排列的，故拼音检索表亦按国标顺序排列。这对检索操作提供了一定的方便。拼音码允许输入1—4键的四种选择，提高了人机对话能力。

总之，重码类的汉字输入编码的选字工作要依靠提示行处理功能来完成，这里包括对输入码的行编辑以及辅助选字操作等，这部分的处理与显示管理模块直接有关。

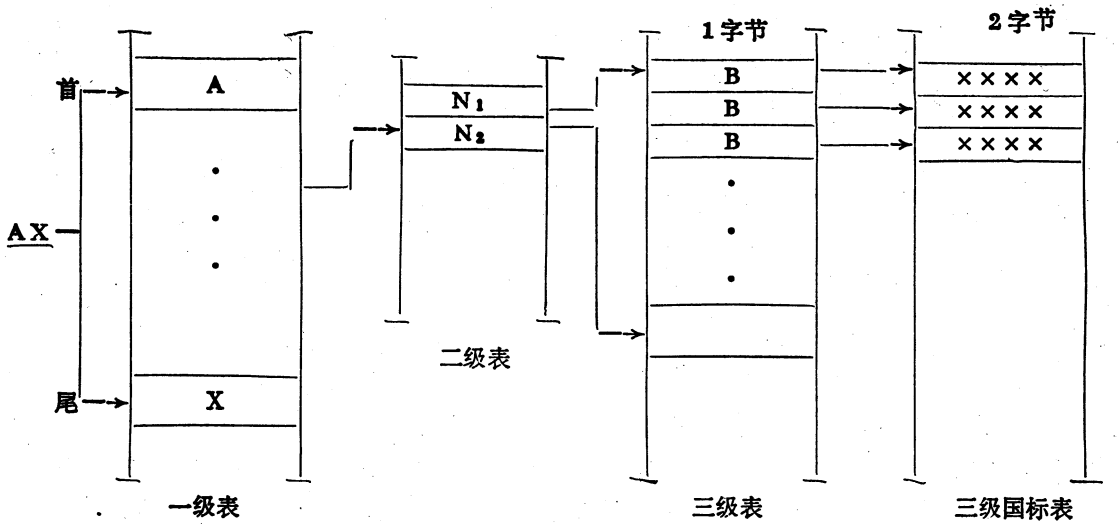


图1-20 首尾码的检索过程

3. 显示管理模块

显示管理模块完成屏幕光标定位、汉字显示、读入屏幕上的汉字、屏幕滚动以及提示行显示处理等功能。

（1）显示缓冲区的结构

需要显示输出的汉字文本必须预先送入内存的一个数据区中，这个数据区称为“显示缓冲区”，它按80×25字节为一块，共三块，分别用来存贮汉字代码、字符标志和字符属性（称作字符区、标志区和属性区）。对汉字而言，字符区存放汉字的二字节内码，标志区指出相应的汉字内码是第1字节还是第2字节；如果采用四字节内码则用来存放第3、4字节，属性区指出该汉字的彩色等属性，当然，相邻两个字节是相同的。前面已经说过，汉字显示是在PC机的图形显示模式下进行的，因此，显示缓冲区与彩色图形显示卡上的16KB的刷新存贮器VRAM相互对应。但由于VRAM所对应的显示屏幕分辨率的限制，一屏只能容纳40字×11行汉字点阵，而内存显示缓冲区可容纳40字×25行汉字内码，所以两者之间必须依靠指针来建立相互的对应关系。

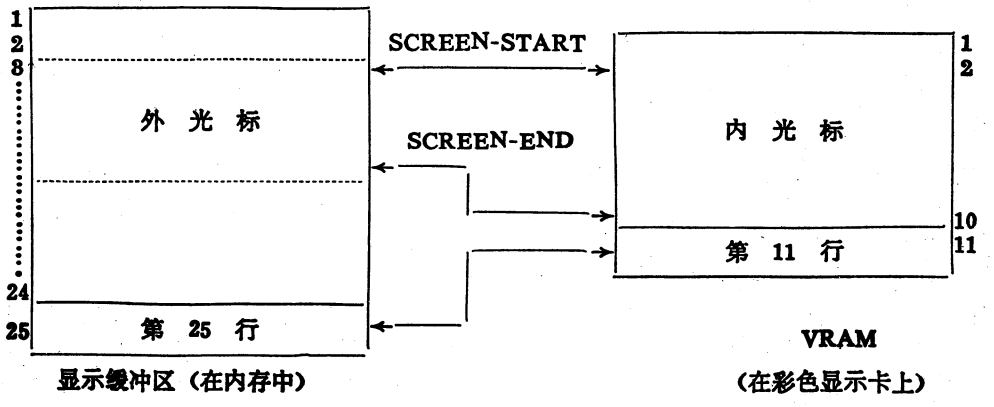


图1-21 显示缓冲区与VRAM的对应关系

图1-21说明了显示缓冲区与VRAM的对应关系，从图中可以看出，由称为屏幕头（SCREEN-START）的指针和屏幕尾（SCREEN-END）的指针，决定着当前屏幕上显示出来的是显示缓冲区中的哪一部分。头指针与尾指针的间距为10行，屏幕的第11行固定地与缓冲区的第25行相对应，用作提示行。当头、尾指针移动时，显示缓冲区中24行汉字文本都可以显示出来，VRAM的作用如同一个显示窗口。

（2）汉字的显示与读入

汉字的显示与读入都是根据光标位置来进行的，而光标位置由光标指针来确定。由于显示缓冲区与VRAM的容量不同，使光标定位变得十分复杂。为此，CCDOS建立了一个以内、外光标为基础的光标系统。外光标用以模拟和记录用户软件对25行一屏显示的操作（对大多数IBM PC软件而言，都是以25行作为一屏的）。而内光标用于产生和记录一屏10行的VRAM的实际操作。外光标到内光标的转换将根据内外光标的指针进行一定的换算来完成。

在屏幕上显示一个汉字的过程大体如下：首先将要显示的汉字内码送至内存显示缓冲区中外光标所指定的位置，然后修改外光标，并按汉字内码从汉字库中取得汉字字模，将其送入图形控制卡VRAM中内光标所指出的位置处，修改内光标，于是屏幕上原光标指定位置处就会显示出该汉字来，从而完成了显示汉字的全过程。

CCDOS的显示管理模块允许直接读取当前光标位置上的汉字字符。该功能是这样完成的：先根据当前光标指针和屏幕头指针（SCREEN-START），找到该汉字在显示缓冲区中的位置，然后从缓冲区的字符区和属性区中，就可以得到该汉字的代码和属性。

（3）滚行处理与提示行输出

屏幕滚行处理功能也是在显示过程中必须提供的。由于高层软件规定的屏幕滚动参数（滚动区界限、滚动行数）都是按每屏25行考虑的，在汉字显示的情况下要与之适应，必须先滚动显示缓冲器的内容，包括字符区、标志区、属性区要一起滚动，把滚动后空白部分的值填写入显示缓冲区，然后再根据光标指针和屏幕头指针，计算出当前窗口中实际上要滚动的行数，来滚动屏幕、建立光标。

显示管理模块中还提供了一个提示行处理子模块。许多IBM PC的软件使用屏幕第25行作为提示行，故在汉字处理场合提示行被规定为屏幕的底行（共11行）。这样，当高层软件使用第25行时，总是与屏幕底行相对应，而不必改变窗口指针。当键盘输入汉字时，屏幕底行就用作提示行。

提示行处理子模块提供了清除底行、根据底行光标指针写入字符、移动或建立底行光标等功能。

4．字模库管理模块与打印机管理模块

CCDOS使用两种字模，即16×16点阵和24×24点阵，因此字模库是以每个字模32字节或72字节为单位构成的。字模按国标顺序排列，故称为国标字模库，一般简称为字库。

CCDOS的汉字内码到字库的检索可以通过字库指针简单地完成。所谓字库指针是指某个汉字字模在字库中的起始字节位置，从该首字节地址开始的32（72）个字节，就是这个汉字的字模数据。只要把这32（72）个字节从库中读出就可用于显示或打印。显然字库的容量受到内存容量的限制。

在国标GB2312-80的规定中，图形符号区（1—7区）和汉字区（16区起）之间尚有一片空区还未定义。为节省字库空间，国标空区未保留，而是让汉字从第8区开始存放，所以字库中的第8区就对应于国标第16区。

CCDOS的16×16点阵字模库是在开机时一次装入内存的，字库的起始地址写在一个叫做C-CHAR-START的单元中，所以只要修改该单元的内容就可以得到不同的字模库起点。在使用汉卡上固化的字模库时，由于与内存地址无关，不需更改起始地址。

打印机管理模块完成汉字文本的打印输出和屏幕信息的硬拷贝等功能，它是以西文打印程序为基础、扩充了对汉字作处理的几个功能块所组成的。

打印汉字的工作原理在第一节中已作了详细介绍，这里仅以图1-22所示的流程图作一小结。需要补充说明的是字型处理、字型处理功能模块对每次送出来的一个字节，先判别是否为控制字符（“ESC”）和字母“I”。若不是则转汉字或ASCII代码处理程序，若是则转去设置有关的控制字，进而再根据随后送出的字母（A—H中的某一个）去置相应的字型位，确定下面要打印的汉字字型。

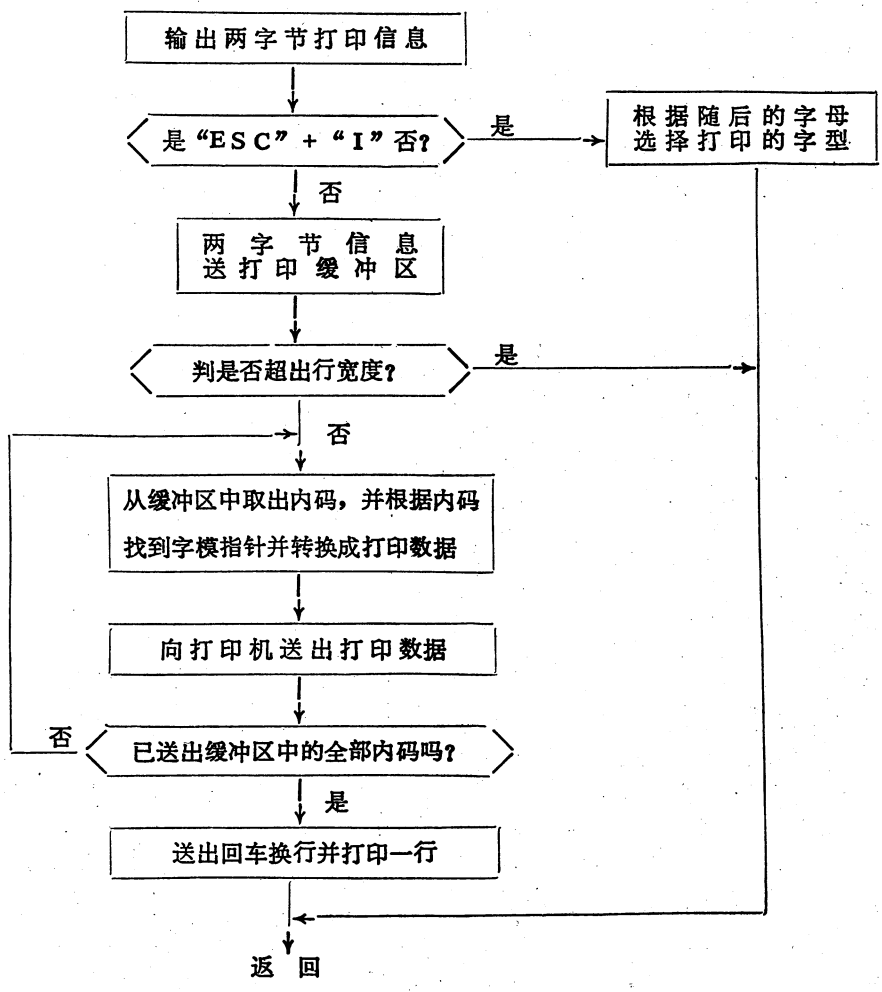


图1-22 汉字打印模块的流程

屏幕硬拷贝亦是经常使用的操作，它直接将CRT屏幕上显示的内容传送给打印机输出。由于屏幕上的内容是存贮在从B8000起的图形刷新存贮器VRAM中的，因此，程序只要依次取出其中的内容，转换成打印数据（转换方法与前述一样），再把打印机设置成图形方式，就可以直接打出屏幕上显示的汉字信息。

第三节　CCDOS的操作与使用

1．系统的启动

要在IBM PC上运行CCDOS，首先必须具备汉字系统所需要的基本硬件环境，这在第一节里已作了介绍。其次，要有必备的软件，这些软件以文件形式保存在软盘上，开机后它们先后装入内存才能正常工作。

CCDOS是在MS-DOS基础上扩充各种处理模块而成的。因此，CCDOS本身也包含了全部MS-DOS软件，其组成如下：

IBMDOS·COM
IBMBIO·COM　　} MS-DOS系统文件
COMMAND·COM

CCCC·EXE
FILE1·EXE　　} CCDOS主体
CCLIB

D320·EXE／D32024·EXE
ALL24P·EXE／2024P·EXE／ALL9P·EXE　　} 打印模块
KK33／KK44

AUTOEXEC·BAT

其中，CCCC.EXE是CCDOS的核心文件，它包含了前面介绍的CCBIOS中的各种基本处理模块；CCLIB是16×16汉字字模库，包括两级汉字共约240KB，它是一个数据文件；而FILE1·EXE是CCDOS中完成引导装入、为字库开辟内存区、初始处理及模式切换等功能的程序。以上文件与MS-DOS配合就可利用键盘、显示器进行基本的汉字处理作业。

打印处理模块因系统配置的打印机不同而有所区别。其中D320.EXE是打印24×24点阵汉字的程序，ALL24P.EXE是打印16×16点阵汉字的程序，包括中断INT5，INT17的修改扩充部分，这两个程序用以配合TH-3070汉字打印机工作；而与此对应的D32024.EXE和2024P.EXE用来与M-2024汉字打印机配接；当使用9针打印机（CP-80，MX-80，FX-100等）时，则要使用ALL9P.EXE。此外，KK33和KK44是一个打印实用程序，用以配合行编辑程序EDLIN使用汉字，并可指定打印参数，但KK33不能指定打印字型。

IBM PC在MS-DOS启动成功之后，将自动执行批文件AUTOEXEC.BAT。AUTOEXEC·BAT的内容是：

```
ECHO OFF          ; 关闭显示
CLS               ; 清屏
FILE 1            ; 模式切换
CCCC              ; CCDOS和字模库引入
ECHO ON           ; 打开显示
```

可见，执行AUTOEXEC·BAT将对MS-DOS进行模式切换并装入扩充的汉字处理模块和汉字字模库，此后IBM PC就在汉字操作系统CCDOS的控制下工作了。

对于PC基本型，必须把两张软盘（第1张盘包括CCCC和FILE 1，第2张包括CCLIB）分别插入A、B驱动器中，再启动机器工作。

对于PC／XT机，以上全部文件均事先由软盘装入硬盘。

系统启动成功后，将在屏幕上显示出如下初始画面：

```
CCBIOS 2.10
中国电子工业部第六研究所 1984年8月

C>
```

这就表示CCDOS操作系统已正常地启动成功了。

由于硬盘的存贮容量大，可以装入24×24点阵汉字字模库，即允许实现24×24点阵汉字的打印输出。（24×24）二级汉字字模库需要580KB的容量，它只能保存在硬盘里，供打印汉字时调用。为了将（24×24）汉字字模库装入硬盘，必须准备好如下文件：

LOAD24·BAT	加载字模库的批文件
CLIB24	(24×24) 汉字字模库
CLIB241	
CLIB242	

加载时，启动执行LOAD24.BAT批文件，即可依次将CLIB24、CLIB241、CLIB242（分装于三张软盘片上）装入硬盘。此后，PC／XT就能支持24针打印机打印输出汉字了。

2．汉字输入操作

系统在冷启动或热启动时，自行执行AUTOEXEC.BAT文件之后，即进入CCDOS操作系统的控制。此时系统仍然处于西文输入模式。当需要输入汉字时，进入汉字输入模式。目前，CCDOS 2.0版和2.1版可以支持四种汉字输入模式。系统规定：

ALT+F1	选择区位码汉字输入模式
ALT+F2	选择首尾码汉字输入模式
ALT+F3	选择拼音码汉字输入模式
ALT+F4	选择快速汉字输入模式

若需要把系统恢复为西文输入模式，则打入ALT+F6即可。

（1）区位码的输入操作

区位码可输入的汉字包括：

- 国标GB2312-80所规定的一、二级汉字共6763个。
- 各种符号202个（如间隔符、标点、运算符、制表符、单位符等）。
- 序号60个（10—20、（1）—（20）、①—⑩、（一）—（十）等）。
- 数字22个（0—9、I—XI）。
- 英文字母大小写共52个。
- 日文假名169个（平假名83个、片假名86个）。
- 希腊字母大小写共48个。
- 俄文字母大小写共66个。

国标区位码中区码在前，取01—94；位码在后，取01—94。

选择用区位码输入汉字时，同时按下ALT键和F1键，此时屏幕底部的提示行是：

区位：_

区位码回显区

它的行首显示“区位”两字，冒号之后将回显键入的国标区位码。当第4个数字键入后，在屏幕上光标位置处即出现该区位码所代表的汉字。

例如，当键入1605时，光标处显示出“哎”。

C>哎_

区位：1605_

（2）首尾码的输入操作

这是有重码的汉字输入方式，操作比区位码复杂。需要用首尾码输入汉字时，同时按下ALT和F2键，此时屏幕底部的提示行为：

首尾：_ 〔 〕

首尾码回显区　　重码汉字显示区　　剩余重码字数

提示行行首出现“首尾”两字作为输入编码的提示信息；接下来是用ASCII字符键入的首尾码回显区（4位）；然后是重码汉字显示区（单色图形方式时可显示10个重码汉字供选择，彩色图形方式时仅显示5个重码汉字）；行末在括号中显示的是在当前输入码的条件下重码汉字的剩余个数，供继续选择重码时参考。当重码数很大时，用“>”键（不必按SHIFT键，下同），可将接下去的10个重码汉字替换显示出来；用“<”键则可退回到前10个重码汉字的显示。当相同拼音需输入两个汉字时，用“ALT”+“—”键可恢复前页，

“ALT”＋“=”键可恢复后页，ALT+“0—9”为当前页重码字的选择。

例如，要输入汉字“光”，先键入首码“d”，屏幕提示行为：

首尾:d　　0:哀1:瓣2:半3:衰4:敝5:弊6:变7:卞8:辨9:辩〔282〕_

它表示以“d”为首码的汉字还有282个，“光”字在已显示的10个重码汉字中未找到，于是可以按下“>”键，显示接下去的10个重码汉字，此时括号中的重码数减为272，仿此不断按“>”键就可找出所需汉字。

首尾:d　　0:辨1:鳖2:憋3:斌4:冰5:部6:糙7:产8:颤9:尝〔272〕_

但是，如果在首码“d”输入后，再输入尾码“t”，则提示行呈现：

首尾:dt　　0:充1:瓷2:刀3:光4:毫5:竟6:竞7:就8:卷9:觉〔017〕_

可见，重码汉字数大大减少。此例中所要输入的汉字“光”已出现在位置“3”上，这时只要按下“3”键，即可在屏幕光标位置处显示出所选的汉字。

有时，首尾两个码输入后，重码汉字还很多，那末可以再输入一个首音码。例如，要输入“施”字，如果在首尾码“dt”输入后，再打入其首音码“u”，则提示行出现：

首尾:dtu　　0:施〔000〕_

它表示首尾码“dtu”所对应的汉字正好是“施”，无重码字。此时，不必再按任何键，此汉字已直接在屏幕上显示出来了。

（3）拼音码输入操作

拼音码也是重码类的输入码。所以，操作方法与首尾码情况完全相似。当选用拼音码输入汉字时，同时按下ALT键和F3键，此时提示行为：

拼音:_

此后的操作与首尾码一样，但是，拼音码的重码汉字数在只输入单个字母时显得太大，通常要按三次键以上才能作出有效的选择。为此，可以将有些联用的声母和韵母简化为一个字母，以减少击键次数。表1-4是常用声母、韵母组合及其对应键的一览表。

表1-4　声母韵母替换表

拼　音	键	拼　音	键
zh	a	ai	l
ch	i	en	f
sh	u	eng	g
an	j	ing	y
ang	h	ong	s
ao	k	Ü	v

键入汉字的拼音码时，上述简化字母也适用于对单韵母字（如“安”拼音为“an”，必须键入j)以及拼音不超过三个键的字（如“知”，拼音为“zhi”，必须键入ai），凡不足三个键的拼音，可用结束键“〔”来补充，这样做可以减少重码数。

以“春”字（汉语拼音为chun）的输入为例，其操作过程如下：

先键入该字的声母“ch”，即应键入i：

```
C>_
拼音:i          0:插1:叉2:茬3:茶4:查5:碴6:搽7:察8:岔9:差〔264〕_
```

再键入第1韵母“u”，得提示行信息为：

```
C>_
拼音:iu         0:初1:出2:橱3:厨4:躇5:锄6:雏7:滁8:除9:楚〔064〕_
```

再键入次韵母“n”：

```
C>_
拼音:iun        0:春1:椿2:醇3:唇4:淳5:纯6:蠢7:莼8:鹑9:蝽〔000〕_
```

此时，所欲输入之“春”字位于“0”选择号下，故可输入0键，于是在屏幕光标处得到了“春”字，而提示行信息被清除，等待着下一个汉字的输入。

在拼音码全部键入后，若重码汉字仍太多，可以再补充键入该字的首尾码，以便减少重码字。

（4）快速输入操作

这是对首尾码输入操作的补充。当操作人员能熟记汉字的首尾码时，可以不通过选择的环节，用一键加空格、两键加空格或者三键加空格直接输入汉字。此操作是通过同时按下ALT键和F4键来选择的。

如快速键入br和空格，在屏幕上就显示出汉字“宝”：

```
C>宝_
快速:br_
```

（5）词组输入操作

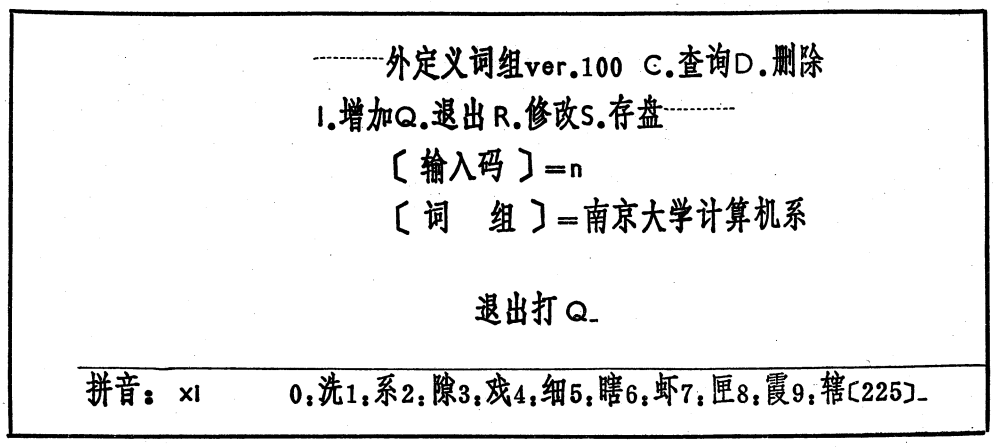
在进行“首尾”码、“拼音”码以及“快速”法输入操作时，可以随时使用“词组”输入功能，从而使输入效率大为提高。

要提供词组输入功能，必须事先建立词库。建立词库，首先要通过定义词组的程序CZ对欲使用的词组进行定义，然后利用装入词库命令CZLOAD，将词库连入系统，于是才可供使用。用户经常使用的词组可由用户自行建入词库。定义词组的操作如下：打入CZ LIU命令，调出词组定义程序CZ，其中LIU为用户名，此时屏幕上出现下列提示信息：

```
.....外定义词组ver.1.00  C.查询D.删除
I.增加Q.退出R.修改S.存盘.....
```

等待输入_

然后键入I（必须在ASCII输入方式下进行），表示要在词库中增加新的词组。于是屏幕上出现：



按提示信息要求，如果在“输入码”处要键入词组调用名“n”，“词组”处键入欲登录的词组“南京大学计算机系”，然后按下S键，把这个新定义的词组加入到词库之中，那末以后直接可以用字母“n”来输入“南京大学计算机系”这个词组了。

词组的输入，必须在“首尾”码、“拼音”码或“快速”输入模式下才能进行。以上面所定义的词组为例，在拼音码输入模式下，键入“n”时得：

拼音：n 0:拿1:哪2:呐3:钠4:那5:娜6:纳7:氖8:乃9:奶〔150〕_

然后，输入词组调用标志“；”，上面的提示行立即改变为词组提示行形式：

A>南京大学计算机系_
词组:n 〔000〕 0:南京大学计算机系_

由于在词组输入码“n”之下，只登录了一个词组“南京大学计算机系”，所以不需要进行选择，在光标处就可显示出该词组。但是当一个调用名下有多个词组被登录时，就要采用与“拼音”码一样的方法来选择所需的词组。

上述词组管理及其使用在实际工作中常感不便，例如使用编辑程序编辑一个汉字文件时，如果要调用的词组尚未建立在词库中，就必须退出正在编辑的汉字文件，转去调出词组程序，将该词组建入词库，然后再恢复文件编辑后才可使用该词组。

“动态词组调用”允许在文件编辑中随时登录和调用新词组，它提高了汉字输入的效率，该项功能是对CCDOS（2.1版）进行修改扩充后实现的①，其功能特点如下：

* 词组调用及定义可在输入文本过程中进行。可登录词组的数量不限，词组可以是

① 此项功能是为后面要介绍的文字处理软件CWP提供的。

纯中文、纯西文或中西文混合方式。

* 使用提示行方式工作，毋需记忆命令。
* 支持汉字整字键盘（汉字大键盘）的输入。

其操作键定义为：

* ALT F5　词典管理
* ALT F9　词组输入

建立词典库的操作是：系统启动时，先自动检查有无词典文件DICTION·DAT，若无此文件，需用词典管理功能去建立词典库。此时可按“ALT”＋“F5”，则屏幕提示行为：

```
A>_
词典管理:(盘A) INS:插入 DEL:删除 C:紧缩词典 D:设置词典盘号 ESC:退出_
```

然后用键入D来指定词典库所在盘号，用“INS”键进行词组建立：

```
A>_
新词组名: _        (数字字符, 空格, 回车)        ESC: 退出
```

此时用“ALT”＋“F6”进入ASCII码输入状态，以键入词组的定义码。例如，要定义“南京大学”为词组时，用户输入自选择的定义码为“njdx”（南京大学四字的拼音首字母），

```
A>_
ASCII_        新词组: njdx;                    回车结束
```

接着再用ALT F1—F4，进入汉字编码输入状态键入词组“南京大学”四个汉字（词组长度不能超过25个汉字或50个字符）。这样，该词组已自动建入词典库中。

要调用词组时，可在汉字输入方式下随时键入“ALT”＋“F9”，提示行出现下列信息：

```
A>_
词组:        (数字字符, *, 空格, 回车)        ESC: 退出
```

键入n、j、d、x后即出现词组“南京大学”：

```
A>_
词组:njdx_        南京大学〔NJDX〕                回车选择好
```

若在同一定义码下登录了几个不同的词组，则可以用光标选择其中的某个词组。

词典中的词组可用DEL键删除，删除后词典中出现空区，可以定期使用键“C”来紧缩词典，除去空区。

使用上述功能，用户可以在工作中不断为自己建立一个常用词库，从而大大提高了汉字输入的效率。

（6）键盘输入时的控制操作

① 建立／取消全中文方式

在汉字文件中，经常会出现一些西文字母和符号（例如产品型号、年代等）。为了文件

的美观，希望这些西文字符与汉字的大小相称，在屏幕上也占一个汉字的位置。更重要的是它们的内码也要使用两字节来表示，使它们象汉字信息一样进行处理。如果西文字母为大写，则可在汉字输入过程中直接输入，如果要输入小写字母，则可用CTRL＋F9键进入所谓“全中文方式”。这时，用户通过“ALT”和“F6”键选择西文输入模式后，键入的所有大、小写西文字符就是两字节代码，它们在屏幕上的显示形式如下：

c> Ａ Ｂ Ｃ Ｄ南大ａｂｃｄ_

但是，由于全中文模式下输入的西文字符，其内码不再是一字节的ASCII码，因此它不能作为西文软件的输入信息（命令、数据、字符串等）使用，否则西文软件就会出错。为此，可以再次使用“CTRL＋F９”键使系统取消全中文方式，这时通过“ALT＋F６”选择西文输入模式后，输入的所有字符就都是标准的ASCII代码了，它在屏幕上仅占半个汉字位置，其形式如下：

C>ABCD南大abcd

在“全中文方式”下，用“BACK SPACE”（退格）键，一次可删除两个字节。而在“非全中文方式”下，却只能删除一个字节（半个汉字或一个西文字符）。

② 纯西文方式／中文方式

在汉字操作模式时，屏幕为图形显示方式，一屏只有11行，这在不需要汉字的西文处理

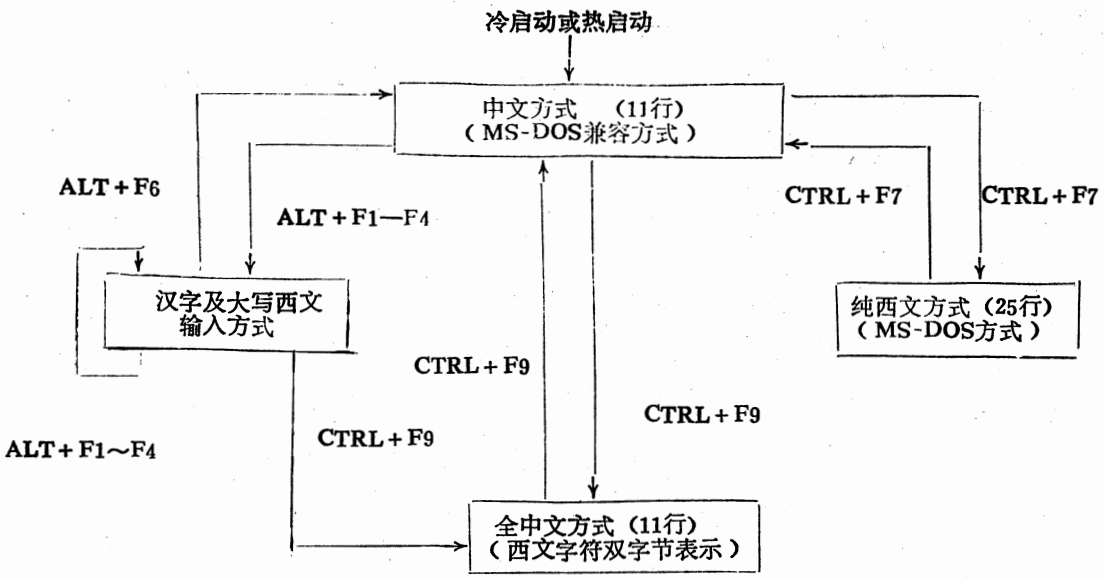


图1-23 CCDOS的工作方式

中很不方便，故可用“CTRL+F7”键将屏幕由11行切换回25行。当再次要求进入汉字模式时，使用同样的键可重新切换回来。图1-23是CCDOS的几种工作方式及其切换关系。其中，纯西文方式即MS-DOS方式，屏幕为25行，只允许输入西文字符；中文方式时屏幕上为11行，也只允输入西文字符，但每个西文字符的内码仅一字节，屏幕上每行可容纳80个字符，实际上这是一种CCDOS下的西文兼容模式；欲输入汉字时，必须通过ALT+F1（或F2、F3、F4）键进入汉字输入模式（可输入汉字及大写西文字母）；欲输入两字节西文时，则必须通过CTRL+F9进入全中文方式。

3. 汉字打印操作

把已保存在磁盘上的一个汉字文件在打印机上打印出来，是汉字信息处理最基本的操作之一。

CCDOS自举后，系统即进入汉字处理状态，但它还没有汉字打印能力，操作员必须根据系统所配置的打印机型号，装入与打印机型号相应的打印控制程序：

- ALL9P　CP-80、FX-100等9针打印机
- ALL24P(16×16字库)或D320（24×24字库）　TH3070、LQ-1500等24针打印机
- 2024P（16×16字库）或D32024（24×24字库）　M2024 24针打印机

（1）使用实用程序打印

在打印汉字文件之前，为了选定汉字的字形、尺寸及打印格式，需要先启动执行打印实用程序KK44。下面是打印汉字文件ABC.DAT的过程。

操作员打入“KK44”命令后，该程序就被调出执行。它通过人机交互方式询问如下各种打印参数：

```
页长（66行/页）49
输入文件名ABC.DAT
输入字型号码（A，B，C，D，…,H）A
行长度（字节/行40）50
纸宽（字节/行）80
输入文件名_
```

然后自动调出汉字打印程序，打印出该汉字文件。

在询问打印参数时，其中的输入字型号码共有八种：A表示16×16或24×24的方形汉字，B表示横向扩大一倍，C表示纵向扩大一倍，D表示横向、纵向均扩大一倍，E、F、G、H与A、B、C、D类似，但它们表示的字形要旋转90°，也就是以纵排方式打印。

（2）使用操作系统的有关命令进行打印

① 用控制键设置打印参数

在进行汉字处理作业时，可以直接使用控制键CTRL+F10来设置打印参数，当同时按下这两个键时，屏幕上的提示行出现如下信息：

```
打印字号（A—P）：_　　纸宽（80—134）：
```

这时根据打印文件的需要，选择所使用的汉字字型和字号，按下回车键后，光标移向纸宽选

择，再键入纸宽数据，按下回车键，打印参数就设置完毕。在字型、字号参数中，A—H的含义与前面所述相同，I—P分别与A—H相对应，但输出的不是标准字形，而是横向宽度比标准字形小一半的浓缩字形。

② 屏幕打印

CCDOS与MS-DOS一样，也可以提供屏幕硬拷贝功能。在汉字处理作业中，当需要把屏幕上显示的汉字文件，原样不动地从打印机上打印出来时，只要同时按下SHIFT和PrtSc两个键。这就是屏幕硬拷贝的功能。从打印机打出的图形，实际上是整屏图形，这时汉字是作为图形来处理的。

③ 用“CTRL-P”命令进行汉字打印

在MS-DOS中，常用“CTRL-P”键来达到边显示边打印输出的目的。而在CCDOS中，同样也可以利用这个键，把屏幕上出现的任何信息（包括汉字），同时从打印机上打印出来。这对于使用DIR命令列出带汉字名文件的目录，以及使用TYPE命令打印出汉字文件都是十分方便的。

④ 用“PRINT”命令输出汉字文件

“PRINT”命令是MS-DOS的假脱机输出打印命令，它可以使计算机在打印文件的同时又可进行其它的作业处理。在CCDOS下，PRINT命令的使用方法与MS-DOS相同。由于汉字打印速度较低，利用PRINT命令是提高系统运行效率的有效方法。

在使用“PRINT”命令打印文件之前，若键入D320或D32024（由汉字打印机的类型确定），则可以打印出（24×24）点阵的汉字文件。当然（24×24）点阵的汉字库必须预先装入硬盘之中。

（3）程序语言中汉字信息的打印

在MS-DOS下，用于西文处理的各种高级程序设计语言，以及在实用程序中进行打印输出的语句，在CCDOS下一般都可用来实现汉字的打印输出。例如：

BASIC：	LPRINT	语句
FORTRAN：	WRITE	语句
COBOL：	WRITE	PRINT-LINE FROM
PASCAL：	WRITELN	语句

程序在输出打印汉字之前，字形、字号的转换是通过输出转义符“ESC”所引导的控制码来完成的。如前所述，字形分为两类，A—H为一类，是标准字形；I—P为另一类，输出的是横向宽度比标准字形小一半的浓缩字形。每一行中，同类字形不同尺寸可以混合出现，这对于很多应用是必须的。但是不同类字形不能在同一行出现。“ESC”控制序列规定如下：

“ESC”＋“I”＋“A”　打印16×16标准点阵汉字。

“ESC”＋“I”＋“B”　打印16×32横向扩大一倍的汉字。

“ESC”＋“I”＋“C”　打印32×16纵向扩大一倍的汉字。

“ESC”＋“I”＋“D”　打印32×32点阵，横向、纵向均扩大一倍的汉字。

“ESC”＋“I”＋“E”（或“F”、“G”、“H”）汉字点阵如“A”（或“B”、“C”、“D”），但字形横向转体90°，得到纵排方式输出。

“ESC”＋“I”＋“I”—“P”　相应于“A—H”，为变体字形（24×16）。

关于“ESC”＋“I”打印控制序列的使用方法，可用下面的BASIC程序作为说明：

```
10      LPRINT  CHR$(27)+"I"+"A";"汉";
20      LPRINT  CHR$(27)+"I"+"B";"字";
30      LPRINT  CHR$(27)+"I"+"C";"处";
40      LPRINT  CHR$(27)+"I"+"D";"理"
50      END
RUN
汉字处理
```

这个程序执行后，能在同一行上打印出相同字形但不同大小的四个汉字。

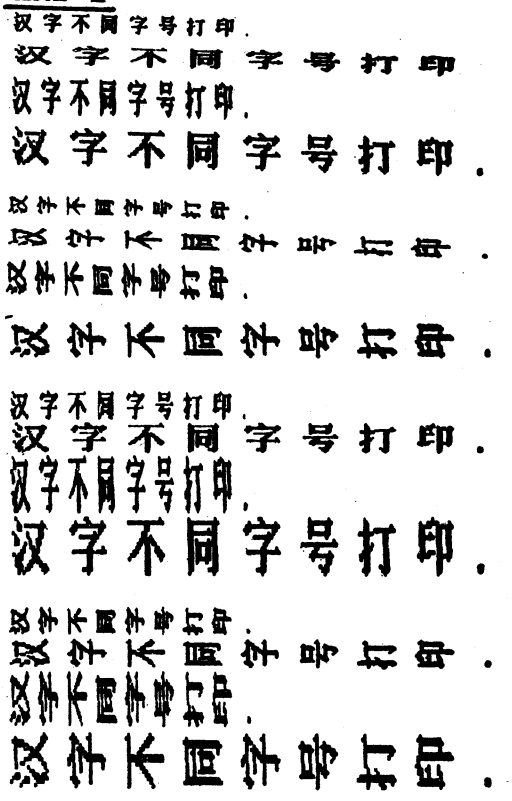
在汇编程序中，通过上述类似的处理，也可以实现字体变换。例1-1是变换字体的汇编程序及其运行结果。

例1-1 汉字字体变换

```
;EX1-1 汉字不同字号打印
STACK   SEGMENT PARA STACK 'STACK'
        DB      256 DUP (0)
STACK   ENDS
DATA    SEGMENT PARA PUBLIC 'DATA'
MSG     DB      '汉 字 不 同 字 号 打 印
        DB      0DH
        DB      0AH
        DB      '$'
CWN     DB      65
DATA    ENDS
CODE    SEGMENT PARA PUBLIC 'CODE'
START   PROC    FAR
        ASSUME CS:CODE
        PUSH    DS
        MOV     AX,0
        PUSH    AX
        MOV     AX,DATA
        MOV     DS,AX
        ASSUME  DS:DATA
        MOV     CX,16
NLOOP:  MOV     AH,5
        MOV     DL,27               ;CHR$(27)
        INT     21H
        MOV     DL,73               ;"I"
        INT     21H
        MOV     DL,CWN              ;字号的 ASCII 码
        INT     21H
        MOV     AH,09H
        MOV     DX,OFFSET MSG
        INT     21H
```

```
        MOV     AL,CWN
        INC     AX
        MOV     CWN,AL
        LOOP    NLOOP
        RET
START   ENDP
CODE    ENDS
        END     START
```

EX1-1



其它语言编的应用程序，原则上也可以按照这种方式进行处理。

必须指出，目前CCDOS的打印功能还在不断完善之中，尤其是用户对打印文本的质量要求越来越高，对24×24点阵汉字，甚至32×32和48×48点阵汉字的打印软件的需求也越来越迫切。图1-24是在PC/XT机上打印出来的48×48点阵的汉字字样，显然它非常美观，能适用于多种场合。

很久以来，这里民间就流传着“杜康造酒，酒醉刘伶三年方醒”的故事。三国时曹操留下了“慨当以慷，忧思难忘。何以解忧？唯有杜康”的诗句。

图1-24 48×48点阵汉字打印输出字样

进行汉字、西文、图符（如表格线）混合打印时，必须注意这些字形的宽度各不相同。在编写程序时需要仔细安排，才能获得所要求的汉字文件输出格式。

4. 实用程序与汉字信息处理

CCDOS是在MS-DOS基础上开发的。MS-DOS的实用程序一般都允许使用扩展的ASCII码(每个字节的高位为“1”)，因此，实用程序把CCDOS中的汉字内码作为两个扩展ASCII码来处理。这样，原来由MS-DOS支撑的大多数西文软件不作改动，就可以在CCDOS下获得汉字信息处理能力。

IBM PC上一般的西文软件在处理汉字时，可能会出现如下一些问题：

* 汉字与西文的显示屏幕长度不同，西文为25行，汉字显示只有11行。如果软件功能与屏幕的特性有关，则会出现屏幕字符行频频滚动的现象。

* 汉字与西文的显示模式不同，前者是图形模式，后者为字符模式。如果西文软件是以字符模式为基础设计的，则必须先作显示模式转换后，才能使用汉字。

* 有的西文软件使用滤符程序，不允许使用高位为“1”的扩展ASCII码。这时，必须修改相应软件中的滤符程序才能使用汉字。

（1）汉字键盘命令

在键盘命令中，命令名可以使用汉字，所有命令中的文件名参数也可以使用汉字。必须注意的是，由于一个汉字占用两个字节，原西文系统中，规定文件名为八个字节，扩展名为三个字节。在改用汉字作为文件名时，文件名最多只能用四个汉字，扩展名只能是一个汉字（一般不要更改文件的扩展名）。例如：

```
C>rename load24.bat 加载字库.bat
```

该操作将文件LOAD 24.BAT改名为“加载字库.BAT”。以后，当需要向硬盘加载字库时，只要打入命令“加载字库”即可。注意，虽然文件名使用汉字以后一目了然，但由于汉字输入效率低，不宜在经常使用的文件上所采用。

（2）EDLIN在编辑汉字文件时的应用

行编辑程序EDLIN，不加修改就可以在CCDOS下用来编辑和修改带有汉字信息的文本文件（如汇编语言及高级语言源程序，批处理文件，数据文件等）。所以，这是一个十分重要的实用程序。

EDLIN以行为单位进行编辑操作，在西文处理中，每行不得超过253个字符（包括〈CR〉）。因此，使用两字节汉字内码时，每行不得超过126个汉字。

EDLIN对汉字的处理与西文字符串相同。例如，要编辑一个如何使用EDLIN功能键的汉字说明书时，操作过程如下：

```
C>edlin 功能键.HLP
New file
*I

1:〈Del〉删除一个字符
```

2：〈Esc〉删除当前行
3：〈Ins〉插入若干字符
4：〈F1〉复制一个字符（半个汉字）
5：〈F2〉复制到指定的字符
6：〈F3〉复制到行末
7：〈F4〉删除到指定的字符为止
8：〈F5〉终止当前行并存入暂存器
9：〈F6〉∧Z

在操作中，可随时使用“ALT”＋“Fi”（i＝1—4）键来选择区位码、拼音码、首尾码等输入汉字。

在EDLIN中，查找和替换字符串操作也同样适用于汉字。例如上面的文件中，将“复制”两字改为“拷贝”，可使用替换命令R：

*R 复制∧Z 拷贝
4：〈F1〉拷贝一个字符(半个汉字)
5：〈F2〉拷贝到指定的字符
6：〈F3〉拷贝到行末

可见，说明书中的“复制”全部都被替换成为“拷贝”两字了。

必须指出，汉字内码的两字节没有分前、后字节，因此，如果要查找的汉字，与某个汉字的后字节加上相邻汉字的前字节所对应的汉字正好相同，那么会出现查找错误。但如果查找和替换的不是单个的汉字，而是两个字以上的词组，则发生这种差错的概率也就很小了。

（3）DEBUG与汉字信息

动态调试程序DEBUG是调试汇编语言程序的有力工具，在CCDOS下使用DEBUG程序与MS-DOS时没有什么不同。

常用的DEBUG操作命令有：

汇编命令A	反汇编命令U
显示／修改寄存器R	显示内存D
修改内存E	填充命令F
运行命令G	跟踪命令T
查找命令S	移动命令M
比较命令C	

其中命令E、F、S和A均可以使用汉字，但查找命令S在用来查找汉字时，同样会出现因代码交叉而造成的查找错误，这在使用中必须注意。

汇编命令A用于小段汇编程序的逐行汇编以及修改目标程序，以下是汇编一段带汉字的程序实例：

```
DEBUG
-A
50CE:0100 MOV AH,09
50CE:0102 MOV DX,0107
50CE:0105 JMP 110
50CE:0107 DB '汉字处理$'
50CE:0110 INT 21
50CE:0112 INT 20
50CE:0114 ^C
-G
汉字处理
Program terminated normally
```

（4）汉字WORD STAR

本书上册第二章介绍过WORD STAR是一个全屏幕通用文字编辑程序，它不仅可以用来编辑源程序文件，而且也可以用来编辑西文的书信和文章。但正由于它是一个全屏幕的编辑程序，所以程序中的许多处理都与屏幕参数（每屏25行，每行80个字符）有关，把它直接用来编辑中文文件就不太合适了。

汉字WORD STAR是在西文WORD STAR的基础上经过适当修改而成的。它主要用来编辑带有汉字的源程序文件以及各种中文书信、文章和报告等。汉字WORD STAR的使用与西文WORD STAR相同，读者可参阅本书上册第二章中有关介绍。

汉字WORD STAR是CCDOS操作系统支持下工作的。它主要由三个文件所组成：WS·COM，WSMSGS·OVR及WSOVLY1．OVR。其中文件WSMSGS·OVR所提供的提示信息都已从英文译成中文，用户使用时十分方便。

要启动汉字WORD STAR，只要打入命令

A>WS

于是在屏幕上出现由汉字给出的主命令菜单：

《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
M 合并打印	

由于汉字显示屏幕只有11行，菜单部分几乎占据2/3，剩下的屏幕空间用以编辑汉字文

本显得窄小。故通常须退出菜单后再进行编辑作业，这是汉字WORD STAR的不足之处。

按下“D”键后就进入中文书信和文件的编辑状态，此时屏幕上显示出使用说明：

使用本命令建立新文书文件或更改现存文件。文件名前是一个驱动器字母A-B及冒号，如省略则隐含当前驱动器。文件名取1－4个汉字，扩展名取1个汉字。文件名是：

文件名字?

键入文件名（最多四个汉字并可带一个汉字的扩展名）后，屏幕上的说明被消去，让出文件编辑空间，进行编辑作业。可以看到，由于屏幕显示只有11行，原来西文编辑作业中显示的编辑提示命令这里全部省略，仅保留下列提示信息，这在使用上有一定不便。

```
C: AAA.A  页号1行号1列01        INSERT ON
L ..... ! ..... ! ..... ! ..... ! ..... ! ..... R
```

菜单中的“D”命令主要用于中文文书编辑。对于建立或修改源程序文件，则应用“N”命令。执行程序时，使用“R”命令。用“X”命令可以退出汉字WORD STAR软件，回到CCDOS。这些功能的操作与使用都与西文WORD STAR相同，这里不再赘述。

（5）IBM PC中文文字处理软件CWP

汉字WORD STAR是西文WORD STAR软件汉字化而成的，由于WORD STAR是为西文文字处理开发的，除了设计上固有的一些不足之外，它对于中文文字处理的特殊性并无考虑。因此，尽管汉字WORD STAR已经具有了处理中文文书的基本能力，但还不是一个理想的中文文字处理软件。南京大学计算机科学系开发的中文文字处理软件CWP提供了功能比较完善，使用比较直观、方便的文字处理手段，下面对CWP作简要的介绍。

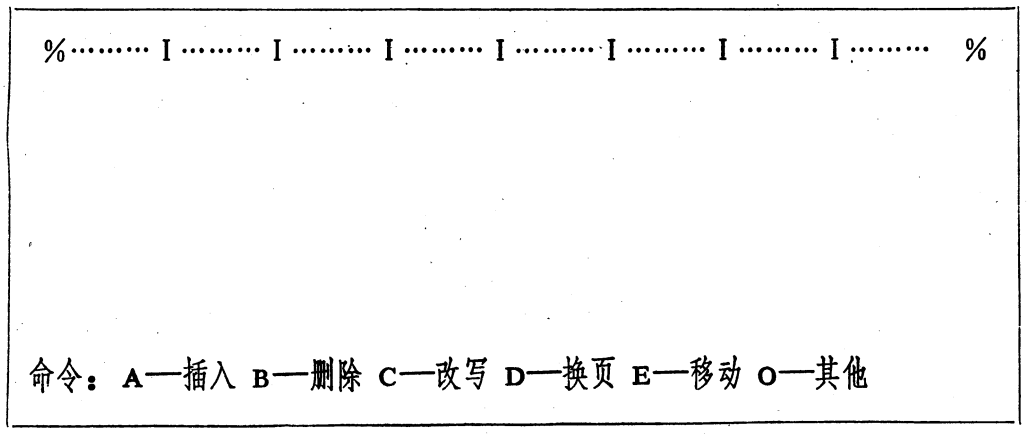
CWP对文件（中文文书）的处理，从输入、编辑、修改直到打印均可按屏幕上的菜单和提示信息进行操作，并可实现宽幅处理，表格线的自由制作，同一文件插入不同对象名的重复打印（插入打印功能），数据按小数点自动对准，编辑过程中可随时进行词组登录和调用等功能，以及一些常用的辅助功能（词典管理，造字等）。它由写入新文件、修改老文件、打印文件和辅助操作等四部分组成。

例如当需要修改老文件时，屏幕上将显示出盘上所有中文文书文件的全部目录：

文件清单				剩余项目：××项			
文件号	文件标题：	版号	页数	文件号	文件标题	版号	页数
1	×××××××××	×	×	2	×××××××××	×	×
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
21	×××××××××	×	×	22	×××××××××	×	×

键入欲修改的文件编号后，即显示出编辑画面，屏幕最上面一行为标尺行，用以定位汉

字，用户可在这里设置行端、行末、以及编辑各种宽度的文书，最底一行为操作提示行；第2—10行为正文显示区，正文的页号在最下一行显示。



利用指示行给出的操作命令，可以实现的主要编辑修改操作有：

插入A	删除B	行端控制K
改写C	换页D	表格绘制L
移动E	登录H	求助M
查找 I	替换 J	退出ESC
复写F	打印G	其它O

除此之外，CWP还提供有17种控制键操作，包括光标控制、上下滚动、对中、右靠和水平列表设置等。

汉字的输入方法与CCDOS相同，还增加了随时登录和调用词组的功能以及大键盘输入功能，使用时极为方便。

如果需要把中文文书按一定格式打印出来，则系统先显示盘上的文件清单，键入需要打印的文件编号之后再指定打印参数，如打印份数，横写还是竖写，何种字体以及插入打印功能等，然后将该文件按格式要求打印输出一份或多份。

CWP还提供了一些辅助功能，例如：打印文件清单、文件删除、盘间文件复写、词组管理、盘片初始化，定义／修改字形等等。

CWP是在修改后的CCDOS（2.1版）上利用C语言开发而成的，它在办公室自动化的应用中是一个有效的中文文字处理程序。

第四节　高层软件与汉字信息处理

IBM PC在汉字操作系统CCDOS的支持下，许多高层软件，特别是高级语言处理程序，往往不经修改就能处理汉字信息。这主要是因为在这些高层软件中，凡是出现西文字符串的地方，原则上也都可以使用中文字符串（也叫汉字串）。高级语言程序具备了汉字输入输出以及对汉字数据的处理能力之后，所开发的应用软件通常具有下列三方面的优点：

① 提高了人机会话的友好程度，使软件的使用和操作更加方便，结果更加直观有用。

② 改善了程序的可读性（由于使用汉字注解），使程序的维护、修改更加容易。

③ 允许直接处理汉字数据，例如汉字信息的输入、存贮、检索、处理、打印报表等，处理效率会明显提高。

但是，由于高级语言处理程序一般是面向西文字符串处理而设计的，因此许多字符串处理功能（包括内部定义的函数和过程）未必适合于处理汉字串，或者对汉字串没有什么意义。另外，CCDOS在汉字输入输出模式下屏幕上只有十行是有效显示区。因此，凡与屏幕特性有关的语言成分，一般不能继续保持其正确性，这在使用汉字进行程序设计时必须注意。

CCDOS中汉字内码的两个字节高位均为1，由于IBM PC能处理包括高位为1的扩展ASCII码，所以，许多在MS-DOS下处理西文的软件也能在CCDOS下处理汉字。例如BASIC、FORTRAN语言等。

但是有些高层软件，如COBOL语言的字符串显示以及dBASE Ⅱ的MODIFY COMMAND编辑操作中，都不允许使用高位为“1”的字符。要使COBOL、dBASE Ⅱ能很好地处理汉字，就必须对高层软件本身进行适当的改造。例如，C-COBOL和C-dBASE Ⅱ就是改造过的软件。用CCDOS运行其它西文软件时，也要注意这些区别。

本节以PC BASIC语言为主，介绍在高级语言程序设计中汉字信息处理功能的使用。其它高级语言中汉字的使用原则上都是一样的，所以仅给出一些例子对汉字FORTRAN程序、汉字COBOL程序以及dBASE Ⅱ程序作些简单的介绍。

1. BASIC语言的汉字信息处理

IBM PC的BASIC语言作为一种通用的交互式语言，简单而实用，尤其是在引入汉字信息处理功能之后，交互式功能更为直观明了，因而更加受用户欢迎。

（1）BASIC程序中汉字的使用场合

在CCDOS支撑下，凡在BASIC程序中出现的字符串，需要时原则上都可以使用汉字串。在下面一些场合，BASIC程序中可以使用汉字：

① 注释语句可使用汉字注释，使程序更易阅读和理解。

② PRINT／LPRINT语句中，可直接使用汉字串作为显示或打印输出的信息。如语句

```
PRINT “The first”；N；“prime numbers are：”；
```

可以改写成为：

```
PRINT “前”；N；“个质数是：”
```

从而使输出结果的含义更加清楚。

③ INPUT和LINE INPUT语句中要求用户输入数据时的提示信息可以使用汉字串。例如语句

```
INPUT “Enter the number of prime numbers：”；N
```

可以改写成：

```
INPUT “输入质数的个数：”；N
```

由于会话中的提示信息使用了汉字，使得程序执行中人机交互的友好程度大为改善。这是汉字信息处理引入BASIC程序中的主要目的之一。

④ 为字符串变量赋值所使用的赋值语句，如READ／DATA语句，INPUT语句，

LINE INPUT语句，INPUT$（n）函数等都可以把汉字串赋给字符串变量。下面是一个电话号码表的打印程序。

例1-2　电话号码表打印程序

```
10    'EX 1-2 电话号码打印
20    PRINT TAB(6); "电话号码一览表"
30    A$ = "单位": B$ = "地点": C$ = "分机号码"
40    PRINT: PRINT A$; TAB(12); B$; TAB(20); C$
50    PRINT: NUM = 0
60    READ N$
70    WHILE N$ <> "END"
75    NUM=NUM+1
80    READ R$, T
90    PRINT N$; TAB (12); R$; TAB (20); T
100   READ N$
110   WEND
120   DATA 计算机系, 北大楼, 2325
130   DATA 计算中心, 中心楼, 3092
140   DATA 计算机厂, 西园, 2862
150   DATA END
160   END
```

PC BASIC语言允许使用字符串数组，因而也就允许使用汉字串数组，这对于处理汉字表格特别有用。一般说来，一个两维数组可以表示出一张表格。例如，下面是用汉字串数组建立的一个通讯录。

例1-3　建立一个200个人的通讯录

程序：

```
10    'EX1-3 建立通讯录
20    DIM ADDRBK$(199, 2)
30    PRINT "==请输入通讯录数据=="
40    PRINT "姓名,    地址,                电话号码<cr>"
50    FOR I = 0 TO 199
60    INPUT"- - >",ADDRBK$(I, 0),ADDRBK$(I, 1),ADDRBK$(I,2)
```

```
70      NEXT
80      END
```

其中二维数组ADDRBK$（199，2）可以用来存放汉字串。下面是程序运行时屏幕上的提示信息以及用户输入的部分数据：

```
==请输入通讯录数据==
姓名，      地址，          电话号码 <cr>
->? 王新，南京市上海路29号，43295
->? 赵民，南京市人民路2号，42295
->? 李黎，南京市广州路135号，32774
->? 程成，南京市汉口路276号，32549
```

（2）BASIC语言对汉字串的处理

与西文字符串一样，BASIC语言对汉字串也可以进行比较、运算和转换操作，但需要注意的是汉字字符为两字节这一特点。

① 汉字串的比较

汉字有不同的排列顺序，最常用的有拼音序、笔画序和部首序。与西文字符集不同，汉字的机内码不能直接作为序值使用。汉字的序值是隐含的、多值的。目前尚无序值的国家标准。

PC BASIC语言对汉字串的比较是以其内码的大小为依据的，因此，只能用来作相等的比较，一般不能用作汉字串的大小比较。例如，要从例1-2的通讯录中查找王新的地址，其查找程序可按如下格式写出：

例1-4　在通讯录中查找某人的地址

程序：

```
110     'EX1-4 从通讯录中查找某人地址
120     INPUT "姓名"；ANAME$
130     FOR I= 0 TO 199
140     IF ANAME$ = ADDRBK$ (I, 0 ) THEN 180
150     NEXT I
160     PRINT "没有找到！"
170     GOTO 120
180     PRINT: PRINT "=====通讯录====="
190     PRINT ANAME$, ADDRBK$ (I, 1), ADDRBK$ (I, 2)
200     END
```

运行结果为：

姓名？王新

=====通讯录=====

王新　　南京市上海路29号　　43295

② 汉字串的运算

BASIC语言为西文字符串提供的唯一的一种运算——并置（＋）运算，也可以用来把两个或多个汉字串合并成一个新的汉字串。例如：

```
A$ = "推广"
B$ = "应用"
C$ = "微型计算机"
PRINT  A$ + B$ + C$
```

其输出结果为：

推广应用微型计算机

③ 字符串函数用于汉字串的处理

PC BASIC语言有许多内部函数是作用于西文字符串的，其中一部分也能作用于汉字串，并有着直接的意义。但汉字是两字节，因此函数中的有关参数的设置要注意这个特点。

假设X$和Y$均为汉字串，则下列函数的意义为：

* LEFT（X$，n）　取汉字串X$中最左边的n/2个汉字，n必须是偶数。
* RIGHT（X$，n）　取汉字串X$中最右边的n/2个汉字，n必须是偶数。
* MID（X$，n，m）　取汉字串X$中从第(n+1)/2个汉字开始的m/2个汉字，n必须是奇数，m必须为偶数。
* INSTR（n,X$，Y$）　从第(n+1)/2个汉字开始对汉字串进行检查，若汉字子串Y$存在,则返回其序号,否则返回0，n必须为奇数。

此外，其它西文字符串函数如STRING$（n，m），STRING$（n，X$），ASC（X$），CHR$（m），VAL（X$），STR$（V）等对于汉字串来说，意义和作用就不明显了。为了使用的需要，用户可以自行定义一些处理汉字串的函数，这会给程序设计带来许多方便。

（3）汉字报表的打印

在CCDOS支持下，PC BASIC语言可以使用表格符、汉字，以及控制汉字字型及大小的“ESC”＋“I”换码序列，即可打印出各种应用部门所需要的数据报表。

第三节中已经介绍了汉字输出打印时“ESC”＋“I”序列的功能。关于打印汉字报表，请参见本节的例1-10。

（4）汉字数据文件的处理

汉字数据文件，是按规定形式组织起来的包括大量汉字信息的数据集合体。在需要进行汉字信息处理时，程序中总离不开对汉字数据文件的处理。

与西文数据文件一样，汉字数据文件是由一组记录所组成的。每个记录又可分成若干字段。汉字数据文件与西文数据文件的不同之处在于：数据文件中的西文字符串允许以汉字串来取代。例如姓名、地址、职称、通讯处等。此外数据文件的文件名也可以使用汉字。

汉字数据文件也可以组织成随机文件和顺序文件两种，它们与西文数据文件基本相同。下面例1-5是把本书上册例 3 -17（教师档案管理程序）改写成为随机汉字数据文件的处理程序。由此可见，使用汉字进行人机会话和结果打印等更有利于开发应用程序。关于本例中数据文件的结构、程序模块的划分、工作原理等这里不再解释，请读者参阅上册第三章第三节的有关内容。

例1-5　教师档案管理程序

```
10 'EX1-5 教师档案管理程序
20 CLS:PRINT:PRINT "档案管理程序"
30 RECLEN%=64:PRINT"您想查找什么文件 ?"
40 INPUT"文件名";FILENAME$:FILENAME$=FILENAME$+".DAT"
50 OPEN FILENAME$ AS #1 LEN=RECLEN%
60 FIELD  #1, 2 AS M$, 8 AS N$, 2 AS S$, 2 AS A$,
          16 AS T$, 4 AS P$, 30 AS D$
70 FILELEN%=LOF(1)/RECLEN%
80 CLS : PRINT "工作菜单"
90 PRINT 1,"输入记录"
100 PRINT 2,"显示记录"
110 PRINT 3,"修改记录"
120 PRINT 4,"删除记录"
130 PRINT 5,"统计"
140 PRINT 6,"结束"
150 PRINT : INPUT "请选择";CHOICE%
160 IF CHOICE%<1 OR CHOICE%>6 THEN 70
170 ON CHOICE% GOSUB 200,400,600,800,900,190
180 CLOSE :GOTO 50
190 CLOSE : END
200 '输入记录模块
210 CLS : PRINT : PRINT
220 PRINT"请输入数据,结束用 END ,,,,,"
230 PRINT : PRINT
240 PRINT"姓名, 性别(男/女), 年龄(1-100), ";
245 PRINT"职称, 工资(***.**), 地址"
250 INPUT ":",NAME1$,SEX$,AGE%,TITLE$,SALARY!,ADDR$
260 IF NAME1$="end" OR NAME1$="END" THEN 350
270 LSET M$=CHR$(0) : LSET N$=NAME1$ : LSET S$=SEX$
280 LSET A$=MKI$(AGE%) : LSET T$=TITLE$
290 LSET P$=MKS$(SALARY!) : LSET D$=ADDR$
300 FILELEN%=FILELEN%+1
310 PUT #1,FILELEN%
320 GOTO 250
350 '完
390 RETURN
```

```
400 '显示记录模块
410 CLS : PRINT : PRINT"文件长度是 ";FILELEN%
420 INPUT "起始记录号，结束记录号";R1%,R2%: PRINT
430 IF R1%<1 OR R2%>FILELEN% THEN 410
440 PRINT"   姓名     性别  年龄  职称      工资      地址"
450 PRINT"   ----     ----  ----  ----      ----      ----"
460 IF R1%=R2% THEN I%=R1% : GOSUB 510 : GOTO 490
470 FOR I%=R1% TO R2%
480 GOSUB 510 : NEXT
490 IF INKEY$="" THEN 490
500 RETURN
510 GET #1,I%
520 IF M$=CHR$(255) THEN 590
530 PRINT USING "¥          ¥";N$;
540 PRINT S$; : PRINT USING "   ###  ";CVI(A$);
550 PRINT USING "¥        ¥";T$;
560 PRINT USING " ###.##  ";CVS(P$);
570 PRINT D$
590 RETURN
600 '修改记录模块
610 CLS : PRINT : PRINT
620 INPUT "请输入要修改的记录号",R1%
630 PRINT"1:姓名 2:性别 3:年龄 4:职称  5:工资 6:地址"
640 I%=R1% : GOSUB 510
660 PRINT : INPUT"请输入要修改的项目",N%
670 ON N% GOTO 680,690,700,710,720,730
680 INPUT "姓名=",NAME1$ : LSET N$=NAME1$ : GOTO 750
690 INPUT "性别=",SEX$ : LSET S$=SEX$ : GOTO 750
700 INPUT "年龄=",AGE% : LSET A$=MKI$(AGE%) : GOTO 750
710 INPUT "职称=",TITLE$ : LSET T$=TITLE$ : GOTO 750
720 INPUT "工资=",SALARY!:LSET P$=MKS$(SALARY!):GOTO 750
730 INPUT "地址=",ADDR$ : LSET D$=ADDR$ : GOTO 750
750 PUT #1,R1%
790 RETURN
800 '删除一个记录模块
810 CLS : PRINT
820 INPUT "请输入要删除的记录号",R1%
830 LSET M$=CHR$(255)
840 PUT #1,R1%
890 RETURN
900 '统计模块
910 AGE=0 : PAY=0 : COUNT%=0
920 FOR I%=1 TO FILELEN%:GET #1,I%
925 IF M$=CHR$(255) GOTO 940
930 COUNT%=COUNT%+1 : AGE=AGE+CVI(A$) : PAY=PAY+CVS(P$)
940 NEXT : CLS
950 PRINT : PRINT "记录总数是 ";COUNT%
960 PRINT : PRINT USING "平均年龄是 = ##.#";AGE/COUNT%
970 PRINT : PRINT USING "平均工资是= ##.##";PAY/COUNT%
980 IF INKEY$="" THEN 980
990 RETURN
```

2. 汉字FORTRAN程序举例

原则上，在CCDOS支持下，FORTRAN程序的各成分中，使用字符串的场合都可以使用汉字串。因此，一个FORTRAN程序的下列成分可以使用汉字：

* 注释行
* 输出语句
* 格式语句
* DATA语句

下面是使用汉字的FORTRAN程序的例子。

例1-6　学生成绩管理之一——数据录入

设有三名学生，每名学生有四门课的成绩需要录入机内，相应的FORTRAN程序如下：

```
C       EX1-6学生成绩录入程序
        CHARACTER*8 NAMES(3)
        DIMENSION   SCRBK(3,4)
        DO 15 I=1,3
        WRITE(*,8)
        READ(*,'(A)')NAMES(I)
        WRITE(*,10)
        WRITE(*,9)
        READ(*,*)(SCRBK(I,J),J=1,4)
15      CONTINUE
9       FORMAT('数学  语文  英语  政治')
10      FORMAT('成绩请输入整数')
8       FORMAT('姓名')
        END
```

本例中，作为变量类型说明，姓名NAMES为一维数组，数组元素是字符类型，长度为8个字节（相当于4个汉字）。成绩SCRBK为二维数组，整数类型。READ语句从键盘上按FORMAT语句规定的自由格式输入信息，该语句中含有隐循环成分（SCRBK(i, j), j=1, 4），它相当于输入信息的安排次序为：

NAMES(1), SCRBK(1, 1), SCRBK(1, 2), SCRBK(1, 3), SCRBK(1, 4)
NAMES(2), SCRBK(2, 1), SCRBK(2, 2), SCRBK(2, 3), SCRBK(2, 4)
NAMES(3), SCRBK(3, 1), SCRBK(3, 2), SCRBK(3, 3), SCRBK(3, 4)

请注意，汉字的输入要用格式说明符A，其中A_8表示输入8个字符或4个汉字。

例1-7　学生成绩管理之二——计算总分

假设学生成绩已经录入机内，则计算总分并打印出每个学生成绩单的程序如下：

```
C       EX1-7 总分计算及成绩单打印
        CHARACTER* 8 NAMES(3),SUM
        DIMENSION SCRBK(3, 4), TOTAL(3)
        DATA SUM/' 总分'/
        CALL SCRIN(NAMES, SCRBK)
        DO 10 I =1, 3
        TOTAL(I)= 0
        DO 10 J=1, 4
        TOTAL(I)=TOTAL(I)+SCRBK(I,  )
10      CONTINUE
        WRITE(*, 45)SUM
45      FORMAT(13X,' 姓名    数学    物理    英语',
     1, 4X' 政治', 4X, A8)
        WRITE(*, 50)
50      FORMAT(13X, '==================='
     1'========')
        WRITE(*, 52)
52      FORMAT(1X)
        DO 55 I=1, 3
        WRITE(*,60)NAMES(I),(SCRBK(I,J),J=1,4),TOTAL(I)
60      FORMAT(14X, A8, 2X, 5(F6.2, 2X))
55      CONTINUE
        END
```

执行程序后，可得到一张学生的成绩统计单：

姓名	数学	物理	英语	政治	总分
李星	100.00	98.00	87.00	90.00	375.00
王成	70.00	82.00	65.00	60.00	277.00
赵志	89.00	100.00	73.00	84.00	346.00

最后再举一个科学计算的例子，由于采用了汉字提示信息，使程序执行结果更为直观。

例1-8　设一物体以初速V_0从距地面S_0米处下落，求它在下落的最初5秒内，每秒末的高度。

```
C       EX1-8 落体速度计算
C       G=9.8M/S**2
        G=9.8
        T=1.0
C       S0=500, V0=2
        WRITE(*, 3)
        READ(*, *)S0, V0
100     S=V0*T+0.5*G*T**2
        HEIGHT=S0-S
        WRITE(*, 2)T, HEIGHT
        T=T+1
        IF(T-5)100, 100, 200
200     STOP
3       FORMAT('请输入：高度, 初速度')
2       FORMAT(/10X,4H当T=, F3.0, 4H秒时,5X,6H高度为,1F6.1, 3H米.)
        END
```

执行结果为：

```
当T=1.秒时        高度为493.1 米.
当T=2.秒时        高度为476.4 米.
当T=3.秒时        高度为449.9 米.
当T=4.秒时        高度为413.9 米.
当T=5.秒时        高度为367.5 米.
```

3. 汉字COBOL程序举例

CCDOS下运行的汉字COBOL编译程序（C-COBOL）是对IBM PC COBOL编译（1.00版）作部分修改后实现的，修改的主要部分是显示语句（DISPLAY），使之能适应汉字处理的要求。

原则上，汉字COBOL中可以对字符串进行操作的语句，都可以用来对汉字串进行处理。因此，一个COBOL程序中的下列一些成分可以使用汉字：

* 注释行（REM）。

* 对汉字数据项赋值（VALUE）。
* 显示汉字数据项（DISPLAY）。
* 传递含有汉字的字符串或数据项（MOVE）。
* 对汉字字符串作比较、排序等操作。
* 对含有汉字的文件进行读写操作。

下例是用汉字COBOL语言设计的一个工资程序（已简化）。

例1-9　汉字COBOL的应用

设工资文件(GZ.DAT)已建立在B盘上，文件的记录由职工号（BH）、姓名（XM）、基本工资（JBGZ）、房租水电费(FZXD)和实发工资（SFGZ）五个字段组成。COBOL程序的字型子句PICTURE是用X（n）描述字符串的，它同样也可用来描述汉字字符串，但"n"必须为偶数。例如，程序中工资记录的PIC描述，姓名（XM）为三个汉字的汉字串，PIC描述为X(6)。

程序运行时是以菜单方式工作的。选择项目1，输入职工的号码、姓名、基本工资及房租水电费；选择项目2，可以通过输入职工的号码查询该职工的工资。下面是查询操作时输出信息的显示画面及程序清单。

```
1.职工号      123        2.姓　名 王明明
3.基本工资    68.00      4.房租水电 3.45
5.实发工资    64.55
继续查询吗? (Y/N_)
```

```
IDENTIFICATION DIVISION.
PROGRAM-ID.   EXAM.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT GZ ASSIGN TO DISK ORGANIZATION IS INDEXED
    ACCESS MODE IS DYNAMIC FILE STATUS IS S
    RECORD KEY IS BH.
DATA DIVISION.
FILE SECTION.
FD GZ LABEL RECORD IS STANDARD VALUE OF FILE-ID IS "B:GZ.DAT".
01 GZ-REC.
    02 BH PIC 9(6).
    02 XM PIC X(6).
    02 JBGZ PIC 9(3)V99.
    02 FZXD PIC 99V99.
    02 SFGZ PIC 9(3)V99.
WORKING-STORAGE SECTION.
01 GZ-TERM.
    02 BHT PIC Z(6).
    02 XMT PIC X(6).
    02 JBGZT PIC Z(3).ZZ.
    02 FZXDT PIC ZZ.ZZ.
    02 SFGZT PIC Z(3).ZZ.
```

```
77 S PIC XX.
77 N PIC 9.
77 F PIC X.
SCREEN SECTION
01 SCR1.
   02 SC1.
    03 S1 LINE 3 COLUMN 3 VALUE "1.职工号".
    03 S2 LINE 3 COLUMN 30 VALUE "2.姓     名".
    03 S3 LINE 5 COLUMN 3 VALUE "3.基本工资".
    03 S4 LINE 5 COLUMN 30 VALUE "4.房租水电"
PROCEDURE DIVISION.
BEGIN. GO TO BEG.
CLEANS. DISPLAY(1, 1) ERASE.
    DISPLAY(4, 1) ERASE.
    DISPLAY(7, 1) ERASE.
BEG.    PERFORM CLEANS.
    DISPLAY(2, 31) "工 资 管 理".
    DISPLAY(4, 28) "1.插入个人记录"
    DISPLAY(5, 28) "2.显示个人记录"
    DISPLAY(6, 28) "3.退出系统".
    DISPLAY(8, 24) "请键入选择键号 1--3".
    ACCEPT N.
    IF N NOT > 0 OR > 3 GO TO BEG.
    GO TO INS DIS ED  DEPENDING ON N.
ED. PERFORM CLEANS.
    DISPLAY(8, 3) "退出系统".
    STOP RUN.
INS.    PERFORM CLEANS.
        OPEN I-O GZ IF S = "30" CLOSE GZ OPEN OUTPUT GZ.
    DISPLAY SC1.
INC.    ACCEPT(3, 14) BH.
    ACCEPT(3, 41) XM.
    ACCEPT(5, 14) JBGZ.
    ACCEPT(5, 41) FZXD.
    SUBTRACT FZXD FROM JBGZ GIVING SFGZ.
    WRITE GZ-REC INVALID KEY DISPLAY(8, 3) "记录错"
    DISPLAY(9, 3) "继续插入吗?  (Y/N)  " ACCEPT F.
      IF F = "Y" MOVE 0 TO BHT JBGZT FZXDT MOVE SPACES TO XMT
    PERFORM MMM GO TO INC ELSE CLOSE GZ GO TO BEG.
DIS.    PERFORM CLEANS.
    DISPLAY SC1 DISPLAY(7, 3) "5.实发工资".
    OPEN INPUT GZ.
ACCBH.  DISPLAY(8, 30) "请键入职工号 ".
    ACCEPT(8, 43) BH.
    READ GZ RECORD INVALID KEY DISPLAY(8, 1) ERASE
    DISPLAY(8, 3) "无此记录" ACCEPT F GO TO ACCBH.
        MOVE BH TO BHT.
    MOVE XM TO XMT.
    MOVE JBGZ TO JBGZT.
    MOVE FZXD TO FZXDT.
    MOVE SFGZ TO SFGZT.
MMM.  DISPLAY(3, 14) BHT.
    DISPLAY(3, 41) XMT.
    DISPLAY(5, 14) JBGZT.
    DISPLAY(5, 41) FZXDT.
DDD.    DISPLAY(7, 14) SFGZT.
       DISPLAY(8, 1) ERASE DISPLAY(9, 3) "继续查询吗?(Y/N)".
    ACCEPT F
    IF F = "Y" DISPLAY(8, 1) ERASE GO TO ACCBH ELSE
    CLOSE GZ GO TO BEG.
```

4. 汉字dBASE Ⅱ

汉字dBASE Ⅱ是在西文dBASE Ⅱ的基础上，对其中DBASE.COM模块中的常驻模块以及DBASEMSG.OVR覆盖模块作了能适应汉字处理的修改而实现的。它的部分提示信息已改为汉字，屏幕显示时每屏仅有10行。但汉字dBASE Ⅱ（C-dBASE Ⅱ）与西文dBASE Ⅱ功能上完成相同，且沿用了dBASE Ⅱ的全部西文命令。编写汉字dBASE Ⅱ应用程序时，下列情况可使用汉字：

① 凡使用文件名作参数的地方，均可使用汉字文件名，汉字文件名最多允许四个汉字。

② 字段名及存贮变量名可使用汉字，最多可允许五个汉字。

③ 字符串型存贮变量的赋值（STORE）和输入（INPUT，ACCEPT）命令可以使用汉字。如，

```
ACCEPT "产品名称＝" TO PZMC
INPUT  "数目＝" TO N
```

④ 数据库文件的数据输入和文件修改命令，可以在字符型字段中输入汉字串。

⑤ 格式输出命令（@…SAY）和格式输入命令（@…GET）可以使用汉字字符串作为提示符，并输出和输入汉字信息。例如：

```
@3,10  SAY "请将软盘A插入驱动器"
@4,12  SAY "今天的日期＝" GET DATA
```

⑥ 注释语句（NOTE，*）可用汉字注释，执行时不显示。但使用REMARK作注释时，屏幕上显示出汉字注释。

⑦ 宏替换函数可用来替换一个长的汉字字符串。例如：

```
STORE "SAVE TO 设备管理程序" TO 设
```

则&设相当于字符串 "SAVE TO 设备管理程序"

⑧ 打印输出时汉字字型的控制，可使用 "ESC" + "I" 序列进行。例如，欲用A型字体打印时，使用下述命令：

```
SET PRINT ON
? CHR(27) + "I" + "A"
```

汉字dBASE Ⅱ是数据处理应用中的有力工具，但它还存在若干不足之处。例如，字段名最多只能用五个汉字，这在实际应用中就显得很不便；显示和打印格式功能弱，不灵活，报表生成不方便。但有一个值得推荐的做法，在CCDOS支持下，利用能处理汉字的BASIC程序与西文或汉字dBASE Ⅱ联合使用，相互取长补短，这样，比单用汉字dBASE Ⅱ更能取得好的应用效果。

例1-10　汉字报表的生成

图1-25是南京市某百货商店1984年五月份各项经济指标完成情况表，它是在CCDOS支持下利用西文dBASE Ⅱ（2.4版）和BASIC程序共同实现的。设计步骤大体如下：

① 用dBASE Ⅱ建立一个数据库文件XB1·DBF，其结构和内容为：

```
·LIST  STRU
STRUCTURE  FOR  FILE,B:  XB1      .DBF
NUMBER  OF  RFCORDS:  00026
DATE  OF  LAST  UPDATE:  02/02/85
PRIMARY  USE  DATABASE
FLD      NAME      TYPE      WIDTE      DEC
001      F0        O         020
002      F1        N         008        002
003      F2        N         008        002
004      F3        N         008        002
005      F4        N         008        002
**TOTAL**                    00053
```

00001	商品销售额	1125.11	939.62	5477.48	4921.54
00002	商品销售毛利	153.37	126.82	737.77	632.36
00003	商品销售毛利率%	13.63	13.50	13.47	12.85
00004	商品流通费	26.68	24.47	129.05	113.57
00005	费用水平%	2.37	2.60	2.36	2.31
00006	商品经营利润	93.24	74.16	444.84	371.14
00007	商品经营利润率%	8.29	7.89	8.12	7.54
00008	利润总额	92.13	73.57	439.57	368.06
00009	利润率%	8.19	7.83	8.03	7.48
00010	全部流动资金平均占用	1341.46	1060.14	1226.83	997.59
00011	全部流动资金周转天数	36.00	34.00	34.00	30.00
00012	商品资金平均占用	1102.92	938.03	1042.09	882.20
00013	商品资金周转天数	29.00	30.00	29.00	27.00
00014	总人数	1583.00	1563.00	1583.00	1555.00
00015	其中：商业职工人数	1474.00	1452.00	1473.00	1444.00
00016	劳效（元）	7633.00	6471.00	37186.00	34083.00
00017	营业员人数	1009.00	1013.00	1008.00	1008.00
00018	劳效（元）	1151.00	9276.00	54340.00	48825.00
00019	大集体人数	364.00	361.00	364.00	358.00
00020	相加差错率（万分之几）	0.96	0.91	0.95	0.73
00021	经营品种	30715.00	27968.00	30715.00	27968.00
00022	实有品种	24492.00	21797.00	24492.00	21797.00
00023	进货总额	1050.91	82078.00	5040.92	4298.74
00024	其中：外地进货	645.24	383.07	3158.51	2054.33
00025	占进货总额比重%	61.40	46.67	62.66	47.79
00026	流动资金利润率%	6.87	7.04	35.83	36.89

② 数据库文件XB1·DBF的内容用命令：

```
COPY    TO    XB1.TXT    DELIMITED
```

复制成为一个BASIC能处理的正文文件XB1.TXT文件。

③ 由于该报表的表头较为复杂，加上每一栏的数据表达形式不一致，“比同期”一栏对各行的计算方法也不完全相同，所以在dBASE Ⅱ中直接设计打印该报表的应用程序比较困难。但借助于下面的BASIC程序，可以达到要求。因为这个程序的逻辑结构并不太复杂，因此不多加解释了。

```
1 'EX1-10
5 WIDTH "lpt1:",132
10 LPRINT TAB(20);"南京市×××百货商店一九八四年二季度五月份各项经济指标完成情况"
20 LPRINT TAB(19);"=================================================================="
30 LPRINT TAB(98);"单位: 万元
40 FOR A=1 TO 110:LPRINT "-" ;:NEXT :LPRINT
50 LPRINT "|          \    时    间   |    本    月    份    实    绩   |   年   度   计   划   检   查   |"
60 LPRINT "|            \           |-------------|--------------|-------------|---------------|--------------|-------------|"
70 LPRINT "|  项    目    \     |  本   期   |  同   期  |比同期 + - % |1 至 5 月累计|  去 年 同 期 |比同期 + - % |"
75 OPEN "xb1.txt" FOR INPUT AS #1
76 IF EOF(1) THEN 200
77 INPUT #1,N$,F1,F2,F3,F4
80 GOSUB 1000
85 READ J
90 IF J=0 THEN 300
100 X=F1-F2:Y=F3-F4
110 GOTO 310
200 FOR A=1 TO 110:LPRINT "-";:NEXT
210 LPRINT
220 CLOSE:END
300 X=(F1-F2)/F2*100:Y=(F3-F4)/F4*100
310 L=LEN(N$)
340 L=L-2
400 A$=MID$(N$,2,L)
405 LPRINT "|";
410 LPRINT USING "\                                     \";A$;
415 LPRINT "|";
416 ON J+1 GOSUB 420,420,500,460
417 GOTO 76
420 LPRINT USING "#########.## |";F1;F2;X;F3;F4;Y
430 RETURN
460 LPRINT USING "############ |";F1;F2;X;F3;F4;Y
470 RETURN
500 LPRINT USING "############ |";F1;F2;
501 X$=STR$(X):L=LEN(X$)-1:X$=MID$(X$,2,L)
502 IF X=0 THEN X$="       平"
503 IF X<0 THEN X$="  快  "+X$+"  天  |"
504 IF X>0 THEN X$="  慢  "+X$+"  天  |"
510 LPRINT USING "\             \";X$;
520 LPRINT USING "############ |";F3;F4;
```

```
521 L=LEN(STR$(Y))-1:Y$=STR$(Y):Y$=MID$(Y$,2,L)
522 IF Y=0 THEN Y$="        平"
523 IF Y<0 THEN Y$="   快  "+Y$+"   天  |"
524 IF Y>0 THEN Y$="   慢  "+Y$+"   天  |"
530 LPRINT USING "\                    \";Y$
540 RETURN
1000 LPRINT "|-----------------------------|";
1010 FOR A=1 TO 6:LPRINT "--------------|";:NEXT
1020 LPRINT
1030 RETURN
1180 DATA 0,0,1,0,1,0,1,0,1,0,2,0,2,3,3,3,3,3,3,1,3,3,0
,0,1,1
```

南京市×××百货商店一九八四年二季度五月份各项经济指标完成情况

单位：万元

时间 / 项目	本月份实绩			年度计划检查		
	本期	同期	比同期+一%	1至5月累计	去年同期	比同期+一%
商品销售额	1125.11	939.62	19.74	5477.48	4921.54	11.30
商品销售毛利	153.37	126.82	20.94	737.77	632.36	16.67
商品销售毛利率%	13.63	13.50	0.13	13.47	12.85	0.62
商品流通费	26.68	24.47	9.03	129.05	113.57	13.63
费用水平%	2.37	2.60	-0.23	2.36	2.31	0.05
商品经营利润	93.24	74.16	25.73	444.84	371.14	19.86
商品经营利润率%	8.29	7.89	0.40	8.12	7.54	0.58
利润总额	92.13	73.57	25.23	439.57	368.06	19.43
利润率%	8.19	7.83	0.36	8.03	7.48	0.55
全部流动资金平均占用	1341.46	1060.14	26.54	1226.83	997.59	22.98
全部流动资金周转天数	36	34	慢 2 天	34	30	慢 4 天
商品资金平均占用	1102.92	938.03	17.58	1042.09	882.20	18.12
商品资金周转天数	29	30	快 1 天	29	27	慢 2 天
总人数	1583	1563	20	1583	1555	28
其中：商业职工人数	1474	1452	22	1473	1444	29
劳效（元）	7633	6471	1162	37186	34083	3103
营业员人数	1009	1013	-4	1008	1008	0
劳效（元）	11151	9276	1875	54340	48825	5515
大集体人数	364	361	3	364	358	6
相加差错率（万分之几）	0.96	0.91	0.05	0.95	0.73	0.22

（续表）

经营品种	30715	27968	2747	30715	27968	2747
实有品种	24492	21797	2695	24492	21797	2695
进货总额	1050.91	82078.00	-98.72	5040.92	4298.74	17.27
其中，外地进货	645.24	383.07	68.44	3158.51	2054.33	53.75
占进货总额比重%	61.40	46.67	14.73	62.66	47.79	14.87
流动资金利润率%	6.87	7.04	-0.17	35.83	36.89	-1.06

图1-25 汉字报表

第五节 MS-DOS 3.0和CCDOS 3.0

MS-DOS 3.0版是MICROSOFT公司1984年开发的MS-DOS操作系统的新版本，它可以在PC，PC/XT和PC/AT上运行。DOS 3.0不仅包含了DOS 2.1的全部功能，而且有了新的扩充。但它们的结构和操作方法都很相似。

DOS 3.0常驻内存部分占用36KB存贮空间（DOS 2.1占用24KB），与DOS 2.1相比，它主要增加了以下几个功能：

* 可以支持IBM PC/AT机所配置的1.2MB高密度软盘驱动器和20MB的大容量硬盘。
* 提供一个虚拟磁盘（Virtual disk），即利用内存贮器模拟的磁盘。
* 增加了多用户系统和网络用户所必须的文件共享功能。
* 增加了一些新的外部命令，并对DOS 2.0原有的部分命令在功能上作了扩充。
* 增加了系统功能调用，增强了文件管理能力。
* 支持BASIC 3.0版，允许用户自行安装基于BASIC的设备驱动程序和执行DOS命令。
* 扩充了系统重构能力，增加了四条系统重构命令。

继DOS 3.0版之后，更新的MS-DOS 3.1版亦已问世。与DOS 3.0比较，它还增加了对IBM PC网络硬件和软件的支撑。例如：

* 对网络转发程序的支持。
* 在IBM PC网络硬件下，增加了文件服务和打印服务程序软件包。
* 提供用户开启安装在IBM PC网络中其它计算机上的打印机和磁盘设备的手段。

DOS 3.1占用的存贮器空间亦是36KB，它可以完全代替DOS 3.0。如果不使用网络功能，那末两者几乎完全相同。以下简要介绍DOS 3.0对DOS 2.0的功能扩充以及CCDOS 3.0对CCDOS 2.0的若干改进。

1. 概述

（1）DOS 3.0的启动

DOS 3.0的启动过程与DOS 2.1类似，但当DOS 3.0运行在PC/AT上时，引导程序引

导DOS 3.0时的查找顺序为：先在A：盘上查找，再在B：盘上查找，最后才在C：盘上查找。下面是从A：盘启动MS-DOS 3.0成功后的画面：

```
VDISK Version 1.0 Virtual disk D:
    Buffer size adjusted
    Sector size adjusted
    Directory entries adjusted
    Buffer size:      64KB
    Sector size:      128
    Directory entries: 64
Current date is Tue 1-01-1980
Enter new date (mm-dd-yy): 5-8-85
Current time is 0:01:07.94
Enter new time: 14
The IBM Personal Computer DOS
Version 3.00(C)Copyright IBM Corp 1981, 1982, 1983, 1984
A>
```

其中，前七行输出的信息表示系统初启时已安装了一个盘符为D：的虚拟盘，该盘的容量为64KB，每个扇区的长度为128B，盘上的文件目录最多允许64个。

（2）磁盘的类型

DOS 3.0可以支持三种类型的磁盘，即硬盘、软盘和虚拟盘。其中软盘又可分为三种，如表1-5所示。PC/AT的高密度软盘驱动器可以读写PC/XT的单面或双面软盘，但使用PC/AT高密度软盘驱动器写入信息后的盘片，在PC/XT上读出时，也许会遇到一些困难。

表1-5　DOS3.0支持的三种软盘片

类　型	容　量	面　数	每面道数	每道扇区数
单　面	160KB／180KB	1	40	8／9
双　面	320KB／360KB	2	40	8／9
双面高性能	1.2MB	2	80	15

DOS 3.0支持的虚拟盘（简称虚盘）可用来加快程序的调试和运行过程。其使用方法是，启动系统后将工作所需的全部文件复制到虚盘，然后就以虚盘为主进行各种文件操作，因而大大减少了费时的磁盘输入输出操作，工作完成后再将虚盘上的内容复制到软盘或硬盘上保存。应当注意，虚盘是使用内存作为存贮介质的，所以断电后就不再保留盘上的内容。

下面是一个使用虚盘操作的例子：

```
C>COPY BASICA.COM D:
C>D:
```

```
D > BASICA
  ⋮
D > COPY  MYPROG.BAS  C:
```

2. DOS 3.0命令简介

DOS 3.0比DOS 2.1增加了四条外部命令，即ATTRIB，SHARE，LABEL 和SELECT。此外它还对有关的全盘操作命令扩充了处理高性能软盘的能力，并修改了BACKUP和RESTORE这两条外部命令的功能。

DOS 3.0对调用外部命令的范围作了扩充，它不仅可以象DOS 2.0那样调用当前目录中的外部命令，还可以调用其它目录中的外部命令。修改后的外部命令的格式如下：

〔＜盘符＞〕〔＜路径＞〕＜命令＞{/＜开关＞}{＜参数＞}＜CR＞

例如，要调用目录＼DOS3下的命令DISKCOPY，把A盘的内容复制到B盘上，可打入命令：

```
C > ＼DOS3＼DISKCOPY  A: B:
```

下面分类介绍MS-DOS3.0的部分命令。

（1）磁盘操作命令

① 磁盘初始化命令FORMAT

FORMAT 命令可用来初始化硬盘、高密度软盘、普通双面软盘和单面软盘，它的命令格式如下：

FORMAT〔＜盘符＞〕〔/S〕〔/V〕〔/B〕〔1/1〕〔/8〕〔/4〕

其中，开关/S表示要在被初始化的盘上写上DOS系统（即IBMBIO·COM，IBMDOS·COM和COMMAND·COM）；/V表示在盘上写上卷名；/B表示以每道8个扇区的形式初始化单面和双面盘，并留出系统区供以后使用SYS命令写入DOS 1.00，2.00或3.00的系统；/1表示初始化单面盘；/8表示初始化每道8个扇区形式的单面或双面盘；/4则表示使用高性能驱动器初始化360KB双面盘（40道软盘）。

例如，使用高密度的A盘分别执行命令：

```
A > C: ＼BIN＼FORMAT  A: /S/V
A > C: ＼BIN＼FORMAT  A: /4
```

可以初始化1.2MB盘和普通的360KB盘。

② 状态检查命令CHKDSK

CHKDSK命令可以用来检查磁盘或文件的状态，其格式如下：

CHKDSK 〔＜路径名＞〕〔/F〕〔/V〕

其中/F开关表示出错时要修复文件定位表；/V开关表示要显示指定盘上指定路径的所有文件的文件名及其路径。

（2）目录操作命令

① 设置文件属性命令ATTRIB

MS-DOS文件属性中的只读属性，在2.0版中无法使用命令直接进行设置或修改，在3.0版中增加了ATTRIB命令，它可以用来设置或显示文件的属性，其命令格式如下：

ATTRIB〔+R｜−R〕＜路径名＞

其中开关＋R表示把指定的文件设置成只读文件；－R则表示把指定文件设置成读写文件；当不选用开关时表示显示指定文件的属性。

具有只读属性的文件只允许读，不允许写和删除。如果企图删除一个只读文件，DOS会给出出错信息："Access denied"。

另外，使用COPY命令复制具有只读属性的文件时，复制的目标文件不具有只读属性。

② 支撑共享文件命令SHARE

SHARE命令是多用户系统和网络系统所必须使用的一条命令，它用于提供文件共享设施，其命令格式为：

SHARE〔/F：〈文件空间〉〕〔/L：〈锁〉〕

其中，/F：〈文件空间〉表示为了记录文件共享所必须的信息而分配的存贮区域的字节数目，缺省值为2048字节。打开使用的每一个文件需要文件名长度再加上11个字节。〔/I：〈锁〉〕用于为使用的锁分配存贮空间，缺省值是20把锁。

如果使用SHARE命令，则文件的读写请求都要进行文件共享码的检查，以便决定操作是否进行。

③ 修改卷名命令LABEL

为了标识磁盘片，可以使用该命令建立、修改和删除盘片的卷名，它的格式为：

LABEL〔〈盘符〉〕〔〈卷名〉〕

盘片的卷名也可以在盘片初始化时使用命令

FORMAT〔〈盘符〉〕/V

写入，必要时再用LABEL命令修改。

（3）文件操作命令

① BACKUP命令

BACKUP命令用来建立某盘上若干文件的备份，它的功能比DOS 2.0有了一点扩充，其命令格式如下：

BACKUP〈盘符〉〔〈路径名〉〕〈盘符〉〔/S〕〔/M〕〔/A〕〔/D：mm-dd-yy〕其中第1个参数表示源文件所在盘或源文件所在目录路径；第2个参数为备份盘的盘符；开关/S表示要复制包括子目录在内的所有文件；/M表示仅复制上次备份后修改过了的文件；/A表示在复制备份前不删除备份盘上已有的文件；/D：mm-dd-yy表示仅复制从指定日期以后修改过了的文件。

BACKUP可用来将硬盘的内容复制到软盘、软盘的内容复制到硬盘、软盘复制到软盘以及硬盘复制到硬盘。当使用软盘做备份盘时，会破坏软盘的原有内容，而使用硬盘做备份盘时，备份文件将建立在子目录/BACKUP下。

BACKUP命令与COPY命令不同，它不仅产生源文件的备份文件，还产生RESTORE命令所需要使用的控制数据（文件名为BACKUP·@@@），所以备份盘最好不要直接使用。

② RESTORE命令

RESTORE命令是BACKUP命令的逆命令，它用来把使用BACKUP命令复制的备份盘恢复到工作盘上，该命令的格式为：

RESTORE 〈盘符〉 〈路径名〉〔/S〕〔/P〕

中，盘符指出备份盘片所在的盘；路径名用来指出把文件恢复到目的盘的指定目录之下。

命令中的开关/S表示要复制备份盘中子目录下的所有文件；/P开关表示备份盘复制之后又有了修改的文件或只读文件时给出提示，由用户选择是否复制。

（4）其它命令

① KEYB××命令

KEYB××命令包括KEYBUK，KEYBGR，KEYBFR，KEYBIT和KEYBSP五条命令，分别用来将ROM BIOS中的键盘处理程序，以英、德、法、意大利和西班牙文的键盘处理程序作替换，便于处理不同国家的文字。该命令的格式为：

KEYB××

其中“××”可以是UK，GR，FR，IT或SP。

键盘的标准方式为美国格式（US）。在选择了某个国家的键盘处理程序之后，如果需要回到标准格式时，只要按下Ctrl+Alt+F1。而要从标准格式再次进入选定的格式，只要按下Ctrl+Alt+F2。

② SELECT命令

SELECT命令是一条MS-DOS 3.0新增加的命令，它可以根据不同的国家选择不同的键盘分布以及日期格式。该命令的格式为：

SELECT〈日期格式编码〉〈键盘编码〉

其参数可选择的值如表1-6所示。

表1-6 键盘编码

国 家	日期格式编码	键盘编码
美 国	001	US
法 国	033	FR
西班牙	034	SP
意大利	039	IT
联合王国	044	UK
德 国	049	GR

SELECT命令的执行过程为：首先使用DISKCOPY命令复制一张系统盘，再用DISKCOMP命令作比较，然后在新建立的系统盘上写一个包含“COUNTRY=〈日期格式编码〉”命令的系统重构文件CONFIG.SYS（关于系统重构命令参见本节第3段）并在该盘上建立一个包含KEYB××命令的自动启动批文件AUTOEXEC.BAT。

下面是执行SELECT命令的一个例子：

```
A>SELECT 049 GR
Insert SOURCE Diskette in Drive A:
Insert TARGET Diskette in Drive B:
Press any key when ready…
Copying 40 tracks
 9 Sectors/Track, 2 Side(s)
Copy another diskette (Y/N)? N
```

```
Insert FIRST diskette in drive A
Insert SECOND diskette in drive B
Press any key when ready…
Comparing 40 tracks
 9 sectors per track, 2 side(s)

Compare OK
Compare another diskette (Y/N) ? N
```

B盘上文件CONFIG·SYS和AUTOEXEC·BAT的内容如下：

```
B>TYPE CONFIG.SYS
COUNTRY=049
B>TYPE AUTOEXEC.BAT
KEYBGR
ECHO OFF
CLS
DATE
TIME
VER
```

③ MODE

MODE命令的功能是用来设置显示器、打印机以及异步通讯器的工作模式，它的功能与MS-DOS 2.0一样，其命令格式为：

MODE〔<设备名>〕<参数表>

参数表中参数的个数和含义，对于不同的设备是不相同的。在选择显示器模式时，一般不用设备名，其格式为：

MODE n 或 MODE〔n〕, m〔, T〕

其中，n的取值范围和含义如表1-7所示；m用来调整显示信息在屏幕上的位置，它为L时表示向左调整，为R时表示向右调整；“T”表示要显示一行测试信息供调整时参考。

表1-7 显示器模式

模 式 n	含 义
40	选用彩色图形显示器（每行40个字符）
80	选用彩色图形显示器（每行80个字符）
BW40	选用彩色图形显示器，黑白（每行40字符）
BW80	选用彩色图形显示器，黑白（每行80字符）
CO40	选用彩色图形显示器，彩色（每行40字符）
CO80	选用彩色图形显示器，彩色（每行80字符）
MONO	切换成单色字符显示器（每行40字符）

使用MODE命令设置打印模式时，有如下两种格式：

MODE LPTe：[n] 〔，〈m〉〕〔，P〕

MODE LPTe〔：〕=COMn

第1式用来设置打印机的工作模式，其中e为1到3，表示打印机号码；n为每行的字符数，可为80或132；m为每时打印的行数，可取值6或8；P表示当发现超时错误时，自动连续不断地重执行直至按下Ctrl-Break为止。第2式表示使用串行接口打印机，该式中的n可为1或2。

MODE命令可以用来设置异步通讯器工作模式，其格式如下：

MODE COMn：〈波特率〉〔，〈校验〉〔，〈数据位〉〔，〈停止位〉〔，P〕〕〕〕

其中，n为异步通讯器号码，可为1或2；波特率可以是110，150，300，600，1200，2400，4800和9600之一，设置波特率时可以仅输入前两个字符；校验方式有N（无），O（奇校验）和E（偶校验），其缺省值为E；数据位可为7或8，缺省值为7；停止位为1或2；“P”表示该串行口发生超时错误时，连续地进行重执直至输入Ctrl-Break为止。

3. 系统重构

MS-DOS提供了一组系统重构命令，用户可以根据机器的配置情况和使用要求对系统进行重构，在系统扩充外设和使用各种局部网时，这些命令尤为重要。

MS-DOS 3.0的系统重构命令可以完成以下功能：

* 设置磁盘缓冲区的个数。
* 设置使用FCB打开的文件的最大个数。
* 设置一次能打开的文件最大个数。
* 设置驱动器的个数。
* 安装设备驱动程序。
* 设置Ctrl-Break检查方式
* 设置顶层命令处理程序的文件名。
* 设置日期和时间的格式。

当用户需要对系统进行重构时，使用编辑程序建立或修改系统盘上的CONFIG.SYS文件。CONFIG.SYS由若干条系统重构命令组成，系统在每次启动后自动查找该文件并对其中的命令加以解释执行。

下面对主要的系统重构命令作一简略介绍。

① BUFFERS命令

BUFFERS命令用来设置系统中磁缓盘冲区的个数，它的格式如下：

BUFFERS=n

其中n为欲设置的缓冲区的个数，可取值1至99，其缺省值为2（PC/XT为3）。

磁盘缓冲区是一块内存区域，它用来暂存磁盘的读写数据。一个缓冲区一次可存放512B的一个扇区。当系统有多个磁盘缓冲区时，系统就轮流使用它们。例如，用户程序需要读入一个128B长的记录，系统则将包含该记录的整个磁盘扇区读入一个缓冲区，然后将指定的记录传送给用户程序。下次用户程序再要读入一个记录时，系统首先在几个缓冲区中进行查找，如果有就直接从缓冲区中将数据传送给用户程序，否则进行磁盘操作。

缓冲区的最佳个数与用户程序的性质以及系统内存的大小（每个缓冲区占528B）有关。当系统配有硬盘时（PC/XT或PC/AT），其数目最好不要小于3。根据经验，当用户程序要处理大量数据时，缓冲区的数目在10至20之间较为合适。

② FCBS

FCBS命令用来设置使用FCB（文件控制块）进行文件操作时，同时能打开的文件个数，其格式为：

FCBS = m, n

其中，m表示一次能同时打开的文件个数，其值为1至255缺省值为4；n表示当用户要打开多于m个文件时不允许关闭的文件数目，它可取值0至255，缺省值为0，n不能大于m。

例如，需要同时打开10个文件，而当打开第11个文件时不允许自动关闭10个文件中的任何一个文件，FCBS命令如下：

FCBS = 10, 10

如果n小于m且用SHARE命令装入了共享程序，那么当用户程序提出要打开第m + 1个文件时，系统会自动从已打开的文件（前n个文件不包括在内）中找出一个最近最少使用的文件，并将它关闭，再打开新文件。在这之后，如用户程序要访问这个已关闭的文件，系统会给出下列出错提示信息：

FCB not available

如系统未装入共享程序，则同时打开的文件个数不受限制。

③ FILES命令

FILES命令用来设置使用文件号进行文件操作时，允许同时打开的文件个数，其格式如下：

FILES = n

其中n的取值范围为8至255，缺省值为8。

FILES命令设置的可同时打开的文件个数是对整个系统而言的，而每个进程只允许最多同时打开20个文件，其中还包括3个标准输入输出设备文件。

④ LASTDRIVE命令

该命令用来设置允许访问的最后一个驱动器的盘符，它的格式如下：

LASTDRIVE = ×

其中×可以是字母A到Z，缺省值为E。应当注意，使用该命令设置的驱动器不允许小于系统中物理驱动器的个数。例如

LASTDRIVE = P

就是使用该命令的一个例子，它表示系统中允许使用16个驱动器。

⑤ DEVICE命令

DEVICE命令用来安装特定的设备驱动程序，其格式如下：

DEVICE = 〈文件路径名〉〔〈参数〉〕

其中，文件路径名表示欲安装的设备驱动程序的文件名，参数表示该驱动程序所需要的参数。

设备驱动程序可以是系统提供的，也可以是用户自行编制的。例如：

DEVICE = VDISK.SYS
DEVICE = CORDRV.BIN
DEVICE = MYPROG.COM

其中，第1行表示安装DOS盘提供的虚拟盘驱动程序；第2行表示安装OMNINET局部网的驱动程序；第3行表示安装自行编制的一个驱动程序。

安装虚拟驱动程序的命令格式如下：

DEVICE = 〔〈目录路径名〉〕VDISK.SYS〔〈盘容量〉〕〔〈扇区大小〉〕〔〈目录项个数〉〕〔/E〕

其中，盘容量的单位为KB，取值范围从1直到可用内存的最大值，缺省值为64。

当DOS 3.0运行在配有扩充内存板的PC/AT上时，可选用/E开关把虚拟盘驱动程序的缓冲区放在扩充内存板上，从1M地址处开始使用，这时可以方便地安装多个虚拟驱动器。

例如：

DEVICE = VDISK.SYS 160 512 64

表示安装一个容量为160K字节，每个扇区512字节，目录项为64个的虚拟盘。

⑥ BREAK命令

BREAK命令用来设置Ctrl-Break检查状态，它的格式为：

BREAK = 〔ON/OFF〕

其缺省值为OFF。当设置为OFF时，只有使用标准输入输出设备操作才检查Ctrl-Break。而设置为ON时，任何输入输出情况都检查Ctrl-Break，其主要目的是用来使Ctrl-Break能退出编译程序的执行。

⑦ SHELL命令

SHELL命令用来设定DOS初始化时所装入的命令处理程序所在的目录，其格式如下：

SHELL = 〈路径名〉

⑧ COUNTRY命令

COUNTRY命令用来设置不同国家所使用的日期和时间的不同格式，其命令格式如下：

COUNTRY = 〈日期格式编码〉

其中，日期格式的编码及其含义请参看本节第2段中的表1-6。

4. CCDOS 3.0

（1）概述

电子工业部六所已经在MS-DOS 3.0基础上，开发出扩充了汉字功能的CCDOS 3.0操作系统。与CCDOS 2.0相比，增加了DOS 3.0上很多有用的功能。同时，由于利用了MS-DOS 3.0版的灵活的系统重构能力，因而汉字输入输出的处理能力也大大增强。

除了DOS 3.0本身带来的功能扩充之外，CCDOS 3.0与CCDOS 2.0的差别还有：

* 可支持硬字库（使用0520汉卡）。
* 可支持包括彩色打印机在内的多种打印机。
* 输入编码方式作了调整，加入了查询方式及笔形码，对拼音码作了改进。
* 增加了字典查询功能等。

熟悉使用CCDOS 2.0的用户，可以毫无困难地使用CCDOS 3.0版操作系统。

（2）汉字输入

在键盘输入时，CCDOS 3.0对功能键的使用作了如下调整：

ALT+F1　区位码输入方式
ALT+F2　查询方式编码（或声韵部形码等）输入方式
ALT+F3　拼音输入方式
ALT+F4　拼形码（或笔形、声声码等）输入方式
ALT+F5　字典建立与取消
ALT+F6　ASCII码输入方式
Ctrl+F7　纯西文和纯中文方式转换
Ctrl+F8　自动光标的建立与取消
Ctrl+F9　全中文方式的建立与取消
Ctrl+F10　选择打印输出时的字型与纸宽

汉字输入操作与CCDOS 2.0相同，这里仅作一些简单的补充说明：

① CCDOS 3.0采用高位为1的两字节内码

② 在拼音码输入中增加了如下的功能

* 允许一字多音，例如“车”，其拼音为“che”和“Ju”，输入该字的拼音码时两者均可。
* 增加了拼音音调，以减少重码个数。系统规定：

阴平—“1”键　　阳平—“2”键
上声—“3”键　　去声—“4”键

例如，‘春’字可输入iun1。

③ 字典的建立与取消功能

用以查询某个汉字的笔形码、拼音码和笔画数目等信息。同时按ALT+F4，建立换字字典：

```
A>_
建立字典方式-
```

然后，转换到某种输入方式，输入所需查询的汉字。此时，屏幕底行即出现该字的各种编码信息。例如，在拼音下键入a，并选择重码号‘0’，得‘啊’字：

```
CCBIOS 3.00

中国电子工业部第六研究所1985.1


A>_
汉字:啊　笔形码:0521　　拼音码:a□□1　　笔划数:10_
```

（3）汉字打印输出

CCBIOS 3.0共配有六种打印机驱动程序，它们配用的打印机是，TH3070、TH1350、

TH1357、M2024（以上是24针打印机）；HP-1018（18针彩色打印机）及MX-100（9针打印机）。打印输出时需配用0520汉卡。

以3070打印机为例，需打印时首先装入驱动程序：

A > ALL24P

再同时按下CTRL+F10，在屏幕的指示行上显示如下操作信息：

打印字号(A-R)：_ 纸宽(80-134)：颜色(0-6)纯中文方式(Y/N)：

TH-3070、1380、1351及M-2024可选8种打印字号(A—H)，HP-1018为5种（A—E），而FX-100为18种打印字号（A—R）。

对纯中文方式的选择，若键入“Y”，那末打印时ASCII字符和汉字字符都将按图形方式打印。

该打印驱动程序没有屏蔽掉打印机本身所具有的控制功能，用户可在BASIC、DEBUG、DBASE等状态下，用相应的命令与语句直接对打印机施行控制，这对增加打印功能是有用处的。

汉字CCDOS 3.0的很多功能尚有待开发与完善，具有网络支持功能的CCDOS 3.1也有待开发，它们将会在IBM PC机的推广应用中发挥更大的作用。

第二章　8086/8088宏汇编语言及其程序设计

IBM PC配备有BASIC，C，PASCAL，COBOL和FORTRAN等多种高级语言，这些语言虽然比较容易学习和使用，但是用它们编写的程序执行速度较慢，控制也不灵活。而汇编语言却具有内存容量省、执行速度快、以及可以直接使用计算机的全部资源等优点。所以，汇编语言始终是程序设计的主要手段之一。

IBM PC配有多种汇编语言，各种汇编语言的功能和使用方法都大致相同。美国MICROSOFT公司开发的宏汇编语言MASM(1.25版）具有宏处理、条件汇编及其它多种功能，并可支持8087协处理器的操作，程序的开发以及调试手段也比较完善。与MASM相应的还有一个小汇编程序（ASM），它的功能比宏汇编稍差，但两者保持兼容。

本章侧重介绍MASM和ASM的基本语法和常用伪指令的功能，以及有关实用程序的使用，并通过一些实例介绍宏汇编语言程序设计的方法。文中把MASM和ASM统称为汇编语言或汇编程序，必要时将指出两者的差别。

第一节　宏汇编语言的基本语法

1．概述

大家知道，直接使用机器语言（代码）编制程序是一件非常困难和乏味的工作。汇编语言实质上是机器语言的一种符号表示。它的主要特点是可以使用符号名来表示机器指令的操作码、地址码、常量和变量；程序员不必为程序（代码及数据）在存贮器中的物理位置进行具体的安排。汇编程序（Assembler）可以把用汇编语言编写的程序（称汇编源程序）汇编成机器语言程序（称目标代码或目标程序）。由于汇编语言兼有机器语言和高级语言的一些主要优点，因此在软件开发中十分有用。

（1）汇编语言源程序

在详细介绍8086／8088汇编语言的语法之前，首先通过一个简单的汇编语言源程序的实例，使读者对汇编语言程序有一个大致了解。

例　打印输出“This is a sample”

```
; SAMPLE PROGRAM 1 DISPLAY MESSAGE
STACK    SEGMENT PARA STACK 'STACK'
         DB 256 DUP(0)              ;256 BYTES OF STACK SPACE
STACK    ENDS
DATA     SEGMENT PARA PUBLIC 'DATA'
MESSAGE  DB      This is a sample program.$'     ;MESSAGE TO DISPLAY
DATA     ENDS
CODE     SEGMENT PARA PUBLIC 'CODE'
START    PROC FAR
```

```
        ASSUME CS:CODE
        PUSH DS                     ;SAVE PSP SEG ADDR
        MOV AX,0
        PUSH AX                     ;SAVE RET ADDR OFFSET(PSP+0)
        MOV AX,DATA                 ;ESTABLISH DATA SEG ADDRESSABILITY
        MOV DS,AX
        ASSUME DS:DATA
        MOV AH,9                    ;SYSTEM CALL FUNCTION NUMBER
        MOV DX,OFFSET MESSAGE
        INT 21H
        RET                         ;RETURN TO DOS
START   ENDP
CODE    ENDS
        END START
```

汇编语言程序一般总是由几个段构成的，每个段都以SEGMENT语句开始，以ENDS 语句结束。本例中的程序一共有三个段。第1段称为STACK，它用来在存贮器的某个地方建立一个堆栈区；第2段称为DATA，它在存贮器中存放了供打印输出使用的数据；第3段的名字为CODE，其中包括了许多以符号表示的指令，用于实现数据的打印输出。至于这些段在内存中的具体位置，程序不必给出，它将由汇编程序和其它实用程序负责安排。

源程序是由一行行语句组成的，所有语句均应当包含在一定的段中，每个段必须有个名字（段名），从性质上可以分成代码段、数据段、堆栈段和附加段等四种。

各个段在源程序中的顺序可以任意，段的数目可根据需要确定，原则上不受限制。

程序中的所有段都必须用段结束语句（ENDS）结束，所有的过程（用PROC语句定义）必须用过程结束语句（ENDP）结束。同样，整个源程序也必须用结束语句（END）来结束，它通知汇编程序停止汇编其它语句。END后面的标号START表示该程序执行时的启动地址。

本例中的源程序是一个独立的汇编单位，称为一个模块。有些程序可以由几个模块组成，各个模块都具有不同的功能，把它们分别汇编后，再用连接程序把它们组织起来，形成一个可执行的程序。在这种情况下，只有主模块的END语句需要指出程序的启动地址，而其它模块不能给出启动地址。

（2）汇编处理过程

使用汇编语言开发软件的过程如图2-1所示，现对这些过程分别予以简要的说明。

① 用EDLIN或WORDSTAR编辑宏汇编语言源程序，它的文件扩展名为.ASM。

② 用宏汇编程序对上述源程序文件进行汇编，产生扩展名为.OBJ的可重定位目标代码文件。需要时宏汇编还能产生以下两种文件：

〈2c〉列表文件　该文件列出了汇编出的代码以及与其有关的地址、源语句和符号表，其扩展名为.LST。

〈2d〉交叉参考文件　这是一个夹有特定控制字符的特殊文件，扩展名为.CRF。在这个文件的基础上，实用程序CREF（2e）产生一个列出定义各个符号的所有源程序行号以及引用各个符号的所有源程序行号的清单，供程序调试时使用。

③ 使用连接程序LINK连接一个或多个.OBJ模块，生成一个扩展名为.EXE（3a）的可执行目标代码文件。在程序开发中，有时希望把各个单独的模块放入程序库中，以便在需要时，通过LINK把目标文件与库中的模块连接起来，成为一个可执行的目标代码文件。使

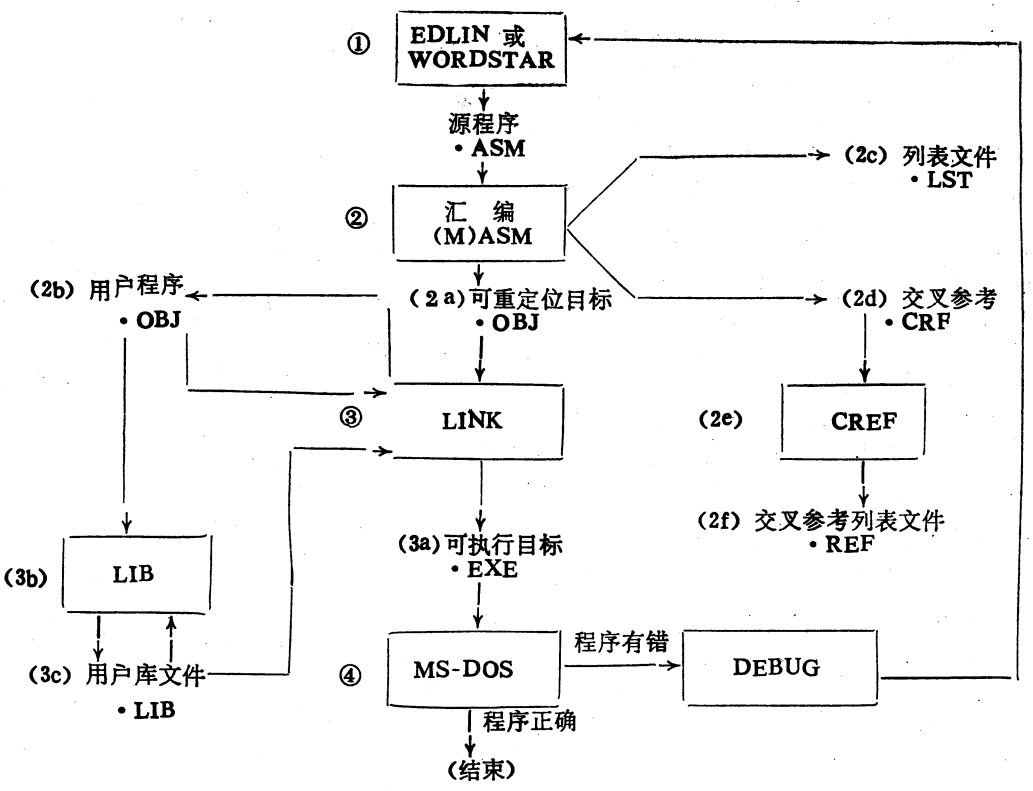


图2-1 使用汇编语言开发软件的过程

用库管理程序LIB（3b），可以利用现有的库文件（3c）或用户的目标代码文件.OBJ生成新的库文件。

④ 在MS-DOS下，运行汇编、连接好的可执行目标代码文件.EXE（3a），如出现错误，可用动态调试程序DEBUG进行调试。找出错误后，用EDLIN或WORDSTAR修改源程序，再重复过程 ②—④，直到程序能正确运行。

2. 语句

汇编语言中一共有两类语句：指令性语句和指示性语句（伪操作语句）。前者如例子中的PUSH，MOV，INT等语句，汇编程序都要为之产生机器指令代码；后者如SEGMENT，ENDS，DB，ASSUME，PROC等语句，它仅仅告诉汇编程序对后面的指令性语句应该如何产生代码，因此也叫做伪操作语句。一条语句在源程序中一般只占一行，超过一行时必须使用续行符号“&”指出。

（1）指令性语句

指令性语句绝大部分对应于8086/8088的机器指令。8086/8088的指令由一个操作码以及必要的确定操作数地址的一些字段所组成。因此，指令性语句中必须包含一个指令助记符和足够的寻址信息，才能使汇编程序产生出这条指令的目标代码。

指令性语句的格式为（方括号表示可以任选的成分）：

〔标号：〕〔前缀〕〔指令助记符〕〔操作数〕〔；注释〕

其中，指令助记符就是所有8086/8088指令的符号名，它们已在本书上册第一章中分类作过简单介绍。指令助记符也可以是程序员使用宏定义语句(MACRO)定义过的一个宏指令名，这将在第三节中再作介绍。

前缀指的是8086/8088中的一些特殊指令，它们是与其它指令配合使用的。例如，串操作指令中的五条重复指令（REP，REPE，REPNE，REPZ，REPNZ）和总线封锁指令（LOCK）。下面是使用前缀的指令性语句的例子：

EX：REP MOVS DEST，SOURCE

语句中的操作数可能是一个或者两个，也可能没有，这将由具体的指令所决定。有两个操作数时，目标操作数在先，源操作数在后，两者用逗号隔开。如果指令助记符为宏指令，则也许需要使用更多的操作数，这由宏指令的定义所决定。

标号是后面紧跟着“：”的一个名字，它代表该语句的存贮器地址，供转移指令（JMP），调用指令（CALL）或循环指令（LOOP）作为操作数使用。

注释是以“；”开始的一个字符串，简明扼要的注释可以提高程序的可读性。

（2）指示性语句

指示性语句（伪指令）的格式为：

〔名字〕〔伪操作命令〕〔操作数〕〔；注释〕

一般说来，其中名字后面不能有冒号“：”，这是它与指令性语句的一个突出的区别（注：MASM1.25版允许在数据定义伪操作前面冠以标号）。指示性语句中的名字可以是符号、常量名、变量名、过程名、段名、组名、……，这是根据不同的伪操作命令决定的。

伪操作命令大约有50—60种，按照它们的功能大致可以分成九类（表2-1）。其中，有“*”的伪操作命令是宏汇编MASM所独有的，这是它与小汇编ASM的主要差别所在。各种伪操作命令将在本章有关部分分别进行介绍。

表2-1　MASM伪操作命令

分　类	伪　操　作　命　令
符号定义伪操作	EQU，=，LABEL
数据定义伪操作	DB，DW，DD，DQ，DT，RECORD*，STRUC*
段定义伪操作	SEGMENT，ENDS，GROUP，ASSUME，ORG
模块定义与通讯伪操作	EXTRN，PUBLIC，NAME，END
过程定义伪操作	PROC，ENDP
宏处理伪操作	MACRO*，ENDM*，EXITM*，LOCAL*，REPT*，IRPC*，IRP*，PURGE*
条件伪操作	IF，ENDIF，IF1，IF2，IFB，IFNB，IFE，IFDIF，IFDEF，IFNDEF，IFIDN，ELSE
列表伪操作	PAGE，TITLE，SUBTTL，.LIST，.XLIST，%OUT
其　它	COMMENT，.RADIX，INCLUDE，EVEN

指示性语句是否需要操作数，需要几个以及需要何种操作数，随伪操作命令不同而不同，其作用也各不相同。它们的格式和含义，将在介绍每条伪操作命令时再给予专门的解释。

3．操作数的寻址方式与汇编表示

指令性语句中的两个主要成分是指令助记符（操作码的符号表示）和操作数。指令助记符比较简单，但操作数的汇编表示方法和规则却比较复杂。这是因为操作数的表示既要能充分体现出汇编语言中使用符号操作数和指令助记符的许多优越性，使程序员尽可能地减少在存贮器分配和地址计算等方面的负担，又要能被汇编程序有效地翻译成对应的特定处理器所具有的各种寻址方式。

（1）8086/8088寻址方式

本书上册第一章第三节中已经详细介绍过8088的指令格式和各种寻址方式，这里只作简要的复习，以便更好地理解机器指令在汇编语言中的表示方法。

8088的指令长度可以是1—6个字节，操作数的个数由操作码决定，操作数的寻址方式由操作码和寻址模式字节（MOD）共同决定，它们可以分成如下八种方式：

① 寄存器寻址　操作数在指定的寄存器中，如MOV AX，BX（AX是目标操作数，BX是源操作数）。

② 立即寻址　操作数就包含在指令中，如MOV AX,15(15是一个立即操作数)。

③ 直接寻址　操作数在内存中，但逻辑地址由指令直接给出。

④ 相对寻址　适用于转移指令和循环指令，转移地址在指令中给出，但必须与指令地址相加后才能得到逻辑地址。

⑤ 寄存器间接寻址　逻辑地址是SI、DI、BX或BP寄存器的内容。

⑥ 基址寻址　逻辑地址是BP、BX寄存器的内容与指令中位移量的和数。

⑦ 索引寻址(变址)　逻辑地址是SI、DI寄存器的内容与指令中位移量的和数。

⑧ 基址索引寻址　逻辑地址是BP，BX寄存器内容加上SI或DI内容再加上指令中位移量的和数。

这八种寻址方式中的操作数可以分成三类：寄存器操作数、立即型操作数和存贮器操作数，下面对它们的汇编表示方法分别进行叙述。

（2）寄存器操作数

8086/8088处理器中可以作为操作数参加运算的寄存器有：通用寄存器（四个16位的通用寄存器AX，BX，CX和DX可以兼作八个8位的寄存器使用）、索引寄存器、段寄存器、指针寄存器。标志寄存器不能作为独立的操作数使用，但大多数指令执行后会改变其中的某些标志位。指令中使用的寄存器操作数是用寄存器名字来表示的，表2-2是8086/8088寄存器在汇编语言中的符号名、大小、作用和作为操作数使用时所适用的主要指令。

（3）立即操作数

立即操作数又叫做数值操作数，它在指令中直接给出，既不需要使用寄存器，也不涉及存贮器访问操作。当然，立即操作数只能作为源操作数使用，不能用作目标操作数，它在指

表2-2　8086/8088寄存器

寄存器名	大　小	用　　途	作为操作数使用时所适用的主要指令
AH BH CH DH	1字节 1字节 1字节 1字节	累加器的高字节 基址寄存器的高字节 计数寄存器的高字节 数据寄存器的高字节	大多数算术和逻辑运算指令，传送指令MOV，移位指令，增量、减量指令等
AL BL CL DL	1字节 1字节 1字节 1字节	累加器的低字节 基址寄存器的低字节 计数寄存器的低字节 数据寄存器的低字节	同上，输入输出指令（AL）
AX BX CX DX	2字节 2字节 2字节 2字节	累加器（整字） 基址寄存器（整字） 计数寄存器（整字） 数据寄存器（整字）	算术指令和逻辑指令，传送指令，输入输出指令（AX，DX），堆栈指令，移位指令，增量、减量指令
BP SP	2字节 2字节	基址指针 栈指针	传送指令MOV，加、减、比较指令，逻辑指令，增量、减量指令，移位指令，堆栈指令
SI DI	2字节 2字节	源变址 目的变址	同上
CS SS DS ES	2字节 2字节 2字节 2字节	代码段寄存器 栈段寄存器 数据段寄存器 附加段寄存器	传送指令MOV，堆栈指令（CS除外）
AF CF DF IF OF PF SF TF ZF	1　位 1　位 1　位 1　位 1　位 1　位 1　位 1　位 1　位	辅助进位标志 进位标志 方向标志 中断允许标志 溢出标志 奇偶位 符号位 自陷位 零标志	不作为操作数使用

令执行过程中不会发生变化。

立即操作数是整数，它有两种形式，一字节立即数和两字节立即数。因此，它的数值在范围0—65535之间。

可以使用立即操作数的主要指令有：传送指令（MOV）、输入输出指令（IN，OUT）、算术指令（加、减、比较）、逻辑指令（与、或、异或）等。在输入输出指令中，立即操作数作为端口地址（0—255）使用。

汇编语言中立即操作数是用常量（字面常量或符号常量）来表示的，也可以用常量及有关运算符组成的数值表达式表示。当然，它们的数值应当符合各种不同指令的具体要求，否则在汇编时将发生错误。

（4）存贮器操作数

前面所说的八种寻址方式中的后六种（③—⑧），都以指定存贮器单元中的内容作为指令处理的对象，因此操作数实际上就是存贮单元的逻辑地址，它们或者可以由指令直接给出（直接寻址），或者可以由寄存器的内容给出（寄存器间接寻址），或者由寄存器内容及指令中的位移量相加后给出（基址寻址、索引寻址和基址索引寻址）。

直接寻址操作数在汇编语言中有两种表示方法：

① 直接给出存贮器的地址码。为了与直接操作数相区别，地址码以方括号括出。例如，

```
MOV 〔0000〕，0000H
```

表示把数值0000H存放在逻辑地址为0的存贮单元中（即由DS确定的数据段中位移量为0的单元）。

② 以变量名的形式出现。因为指令的处理对象存放在存贮器单元中，实质上它在程序中是一个变量，如果预先给它定义了一个名字，汇编语言就可以通过这个名字来访问存放该变量的存贮单元。例如，

```
FOO  DB   ?
     MOV  FOO, AL
```

表示把寄存器AL中的内容存放到名字为FOO的存贮单元中去。

寄存器间接寻址方式的操作数在汇编语言中的表示很简单，只要把寄存器名放在方括号中即可。例如，

```
MOV  AX, 〔BX〕
```

表示把逻辑地址为BX内容的一个操作数送到AX寄存器。必须注意，可以用作寄存器间接寻址的BX，BP，SI和DI四个寄存器，在形成物理地址时，它们使用的段寄存器并不完全一样，其规定如下：

寄存器	使用的段寄存器
〔BX〕	DS
〔BP〕	SS
〔SI〕	DS
〔DI〕	DS

值得注意的一点是，寄存器间接寻址和变址操作数的表示方法有时会引起指令含义的不确定，例如，

```
INC 〔BX〕
```

这条指令是把一字节操作数还是两字节操作数加1，汇编程序就无法确定，因而也就无法翻译出目标代码。这个问题的解决方法将在第6小节再作介绍。

使用基址寻址、索引寻址及基址索引寻址方式的操作数，它们的汇编表示方法分别如下：

```
DELTA 〔BP〕
DELTA 〔BX〕
DELTA 〔DI〕
DELTA 〔SI〕
```

DELTA[BP][DI]
DELTA[BP][SI]
DELTA[BX][DI]
DELTA[BX][SI]

其中，DELTA可以是变量名，它给出变量所在存贮区的首地址，也可以是常量，它表示写在指令中的位移量；方括号有相加运算的意思，因此，下面的几种书写方式都是等价的：

5[BX][SI]
[BX+5][SI]
[BX+SI+5]
[BX]5[SI]
[BX+SI]5

在形成物理地址时段寄存器的使用规则是：凡使用了BP寄存器寻址的，段寄存器为SS；其它情况都使用段寄存器DS。

总之，无论哪一种寻址方式，指令中的存贮器操作数总是一个地址，它指出该指令所要处理的存贮器单元的位置。汇编语言中存贮器操作数的表示方法比较复杂，它的一般形式是由基址寄存器、索引寄存器、常量、变量及有关运算符所组成的一个地址表达式。

4．常量与数值表达式

（1）常量及其表示

所谓常量，就是在汇编时已经确定的值。在汇编语言中，常量主要用作指令性语句中的直接操作数，也可作为存贮器操作数的组成部分（如位移量），或者在伪操作指令中用于给变量赋初值。

为了便于程序设计，常量有多种表示形式：二进制、十进制、十六进制、八进制、字符串以及十进制实数和十六进制实数形式(小汇编不允许使用实数)，它们的格式各不相同，并采用不同的基数标记加以区分。如果不带基数标记，则认为是缺省的基数——十进制。使用伪操作命令·RADIX可以改变缺省的基数。表2-3列出了各种形式常量的格式。

常量可以以数值形式直接写在汇编语言的语句中，也可以预先为它定义一个名字，然后在语句中用名字来表示该常量。前者称为字面常量，后者称为符号常量。一般情况下，它们可统称为常量。

使用符号常量的优点是可以改善程序的可读性，它的定义需要使用伪操作命令“EQU”或“=”。例如，

```
PORT    EQU  77
ALPHA   =35*21
        MOV  AX, ALPHA
        OUT  PORT, AX
```

其中，ALPHA表示立即操作数735，FORT表示端口地址77。有关伪操作命令EQU和=的功能，将在第三节详细介绍。

表2-3　各种形式常量的格式

数据形式	格　式	×取值范围	例　子	注　解
二进制	××××××××B	0--1	01110001B	
八进制	×××O ×××Q	0—7 0—7	735 O 412 Q	英文字母O
十进制	××××× ×××××D	0—9 0—9	65535 1000D	缺省形式 用.RADIX改为非十进制基数时
十六进制	××××H	0—F	0FFFFH	最前面一个字符必须是0—9
ASCII	‘××’ “××”	ASCII字符 ASCII字符	‘OM’ “OM”	只有在DB命令中指定两个字符以上时使用
十进制实数	××.××E±××	0—9	25.23E-7	浮点数形式
十六进制实数	×…×R	0—F	8F76DEA9R	最前面一个数字必须是0—9，如果第1个数字是0，那么这个数的总位数必须是9，17或21，否则总位数是8，16，20

（2）数值表达式

汇编语言允许对常量进行三种类型的运算，它们是：

① 算术运算（+，－，*，/，MOD，SHR，SHL）

加（+）、减（－）、乘（*）和除（/）不需解释，模除（MOD）表示两个整数相除后取余数。SHR和SHL分别是右移和左移运算符，例如，设NUMBER为01010101，则

NUMBER　SHR　2的结果为00010101

NUMBER　SHL　3的结果为10101000

② 逻辑运算（AND，OR，XOR，NOT）

对整数常量（包括字符常量）可以进行与（AND）、或（OR）、异或（XOR）和非（NOT）四种逻辑运算，它们都是按位进行的，结果仍为一个整数常量。要注意的是，AND、OR、XOR和NOT既是常量运算符，又是指令助记符。作为常量运算符使用时，它必定出现在语句的操作数部分，而且运算是在汇编期间完成的；而作为指令助记符使用时，它们出现在语句的操作码部分，其运算要在程序执行期间进行。例如，

AND　DX，PORT　AND　0FFH

③ 关系运算（EQ，NE，LT，GT，LE，GE）

两个常量可以进行六种关系运算，即相等（EQ）、不等（NE）、小于（LT）、大于（GT）、小于等于（LE）和大于等于（GE），它们的运算结果是两个特殊的常量。若关系不成立则为0，若关系成立就是0FFFFH（16位全1）。

常量和上述三类运算符可以组合成表达式，其结果是数值，因此称为数值表达式。在汇编语言中，凡可使用常量的场合，原则上都可使用数值表达式。数值表达式值的计算是在汇

编期间进行的，这给汇编语言程序设计提供了很大的方便。下面是使用数值表达式的一个例子：

```
MOV  BX, ((PORT LT 5 ) AND 30 ) OR((PORT GE 5 ) AND 20 )
```

当符号常量之值小于5时，汇编程序将把它汇编成：

```
MOV  BX, 30
```

否则汇编成

```
MOV  BX, 20
```

5．标号

（1）循环、转移和调用指令

循环、转移和调用指令的操作数用来指出应该转去执行的目标指令的地址。

各种循环指令和条件转移指令的操作数是在范围为－128—127之间的一个数值，用来表示本指令与目标指令的距离，它将被加到指令地址寄存器（IP）中，以便实现段内的向后或向前的短距离相对转移。

无条件转移指令（JMP）的情况比较复杂。它可以使用16位的数值（与目标指令的距离）实现段内直接长转移（范围－32768—32767），也可以使用8位数值实现段内直接短转移（范围－128—127），还可以实现段内间接转移（使用寄存器操作数或使用存贮器操作数）。上述段内转移并不修改CS寄存器的内容。当程序中需要从某一段跳转到另外一段时，就必须修改CS寄存器的内容，这称为段间转移，无条件转移指令可以实现段间转移，它提供了两种方式：段间直接转移——指令中直接给出转向地址的段地址和段内位移量；段间间接转移——指令中给出存贮器操作数，据此可以从内存中找到转向地址的段地址和段内位移量。无论是段内转移还是段间转移，间接转移方式都可用来动态地（程序执行期间）计算转移地址，但一般使用得较少。

子程序调用指令（CALL），除了需要在栈中保存返回地址（段内调用仅保存IP的值，段间调用还需保存CS的值），且不准使用段内直接短调用（范围在－128—127之间）之外，与JMP指令的处理完全一样。它们的转移方式及其汇编表示方法总结在表2-4中。

表2-4　JMP和CALL指令的转移方式及汇编表示

类别	转移方式	操作数类型	操作数使用方式	汇编表示法举例
段内转移	直接	2字节立即数	加入IP	JMP PROGIA
		1字节立即数	加入IP	JMP SHORT QUEST
	间接	寄存器操作数	送入IP	JMP BX
		存贮器操作数（2字节）	送入IP	JMP [BP+JTABLE]
段间转移	直接	4字节立即数	送入CS IP	JMP NEXTROUTINE
	间接	存贮器操作数（4字节）	送入CS IP	JMP INTERS[BX]

为了便于编写汇编语言程序，循环指令、条件转移指令、转移指令以及调用指令中的直接转移方式（段内或段间），如果预先给目标指令定义一个名字，则它们的转向地址（操作数）就可以使用符号名来表示，然后由汇编程序在汇编时自动计算出它们的数值，并填入翻译后生成的目标指令代码中。这种转向地址的符号表示就叫做标号。例如，表2-4中的PROGIA、QUEST和NEXTROUTINE都是标号。

（2）标号及其属性

标号是为一组机器指令所起的名字，它用作汇编语言程序中转移指令（JMP）、调用指令（CALL）和循环指令（LOOP）等的操作数（转向地址），因此一般只在代码段内进行定义和使用。

因为标号表示的是指令地址，所以它有三个属性，即段地址、段内位移量和类型。

段地址指的是该标号所在段的段地址（单位为16字节）。当程序中使用这个标号时，该地址必须在代码段寄存器（CS）之中。

位移量即该标号距离所在段首址的偏移量，其单位为字节，这是一个16位的无符号整数。

标号的类型有两种：NEAR类型和FAR类型。凡属NEAR类型的标号仅能在定义该标号的段内使用，而FAR类型标号则无此限制。宏汇编程序对类型不同的标号作不同的处理。当定义一个标号为NEAR类型时，宏汇编程序仅仅使用两个字节的指针给出其位移量属性；而当标号定义为FAR类型时，则要使用四个字节给出它的段地址及段内位移量，因而汇编程序对转向NEAR标号和FAR标号的转移指令所生成的目标代码当然也是不同的。

（3）标号的定义

定义标号的方法有下列两种：

① 在机器指令助记符之前，使用〈标号〉并紧跟着一个冒号“：”，它表示该名字被定义成一个类型为NEAR的标号。〈标号〉：也可以单独成为一行，但紧接着的一般应该是机器指令语句。例如，下面语句中的CLEAR和SUBROUTINE都是NEAR标号。

```
CLEAR: MOV  AL, 20H
SUBROUTINE:
       MOV  AL, 20H
```

② 使用过程定义伪操作（PROC）定义一个过程时，为该过程所起的名字也是一个标号，可以作为CALL指令的操作数使用。PROC语句的格式是：

〈过程名〉 PROC 〔NEAR〕

〈过程名〉 PROC FAR

前者定义的过程名是一个NEAR标号，后者定义的是一个FAR标号。有关PROC伪操作指令在第三节再作详细介绍。

（4）标号的使用

一般而言，标号只有在循环、转移和调用指令中才会作为操作数使用。循环指令和条件转移指令中使用的标号类型必须是NEAR，这时使用标号的语句和定义该标号的语句的距离必须在范围－128—127之内，否则汇编时将发生错误。无条件转移指令和调用指令所使用的标号，如果是在其它段内定义的，必须是FAR类型；如果是在同一段中进行定义的则应该是

NEAR类型。无条件转移指令中的NEAR标号，如果定义标号和引用的两语句的距离在-128—127之内，则可以在标号前加一运算符SHORT，表示汇编时只要生成一字节的偏移量，因而节省一个字节的目标代码。

如果程序中的标号是先引用后定义（称为“提前引用”），则引用该标号时必须明确指出该标号的类型，具体方法在后面的地址表达式部分再作介绍。

至于采用段内或段间的间接方式的转移指令以及调用指令，它们不再使用标号作为其操作数，而是与普通的存贮器操作数的表示方法相同，这将在下面再详细讨论。

6．变量与地址表达式

（1）变量及其属性

汇编语言中，变量实际上就是内存中的一个数据区的名字，它可以作为指令中的存贮器操作数来引用。因此，变量具有如下三个属性：

① 变量的段属性，即与该变量相对应的数据区所在段的段地址，也就是定义该变量时，所在的段址（单位为16）。当访问该变量时，它必须存放在某一段寄存器内。

② 变量的位移量属性，即该变量所在段的段起始地址到该变量的距离。

③ 变量类型，它指出数据区中数据项的存取单位是字节（BYTE）、字（WORD）、双字（DWORD），四字（QWORD）还是十个字节（TBYTE）等。

可见，变量与标号的差别在于变量指的是数据，而标号意味着是指令代码。变量的类型指的是数据项存取单位的大小，标号的类型则是指使用该标号的距离的远近（NEAR或FAR）

（2）变量的定义

变量可以使用伪操作命令DB、DW、DD、DQ和DT来进行定义，它们的格式为：

〔〈变量名〉〕〈数据区定义伪操作命令〉〈表达式〉

五种数据区定义伪操作命令所确定的变量类型及数据存取单位如表2-5所示。这些伪操作命令除了定义数据区中数据项的类型外，还可以通过语句中的表达式确定数据区的大小和它们的初值。表达式有下列几种：

表2-5　数据区定义伪操作所确定的变量类型

伪操作命令	数据项类型	数据存取单位
DB	BYTE	1 字节
DW	WORD	2 字节
DD	DWORD	4 字节
DQ	QWORD	8 字节
DT	TBYTE	10 字节

① 数值表达式

② ASCII字符串（只有DB命令允许字符串长度超过三个字符）

③ 地址表达式（只适用于DW和DD命令）

④ ？（表示所定义数据项无确定初值）

⑤ 〈n〉DUP（？）（n为正整数，它是重复因子，表示定义了n个数据项，它们都无确

定的初值）

⑥〈n〉DUP(〈表达式〉〔,〈表达式〉,…〕)（定义了n个数据项，其初值由表达式确定）

下面是使用数据区定义伪操作的一些例子：

```
MSG   DB   'MSGTEST', 13, 10          (MSG变量共九个字节，且有确定初值)
TABLE DB 100 DUP(5 DUP(4), 7)        (TABLE中有600个字节，初始内容为4、
                                      4、4、4、4、7…4、4、 4、 4、 4、7)
BSIZE  DW  4*128                     (BSIZE变量共一个字，初值为512)
DBPTR  DD  TABLE                     (DBPTR变量共一个双字，初值为 地址表
                                      达式TABLE的位移量和段地址)
LONGREAL  DQ  3.14159                (LONGREAL变量用于存放长实数，其初值
                                      为3.14159)
DECIMAL   DT  1234567890             (DECIMAL中存放紧凑十进制数，其初值
                                      为1234567890)
DBPTRI DW TABLE+10,TABLE+20          (DBPTRI中的两个字，分别存放两个地址
                                      表达式中的位移量)
```

（3）记录与结构

汇编语言中的变量类型除了用数据区定义伪操作定义的五种之外，也可以象高级语言那样由程序员自己定义更复杂的类型，这需要使用RECORD和STRUC两种伪操作命令。

① 记录

伪操作命令RECORD用来设计一个单字节或双字节的记录格式，RECORD命令为：

〈记录名〉RECORD　　字段1，字段2，…

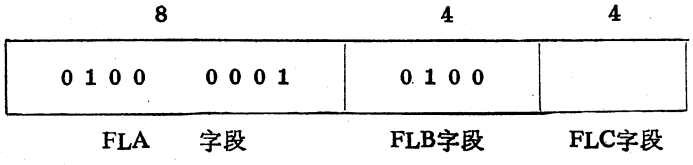
其中各个字段的格式为：

字段名：宽度〔=表达式〕

宽度的单位是位（Bit），表达式的值是该字段的初值。如果记录中各字段宽度之和不超过8位，则该记录为单字节记录，超过8位时（不得大于16位）则为双字节记录，下面是使用RECORD命令定义一个记录名为AREA的例子：

```
AREA  RECORD  FLA: 8='A', FLB: 4=4, FLC: 4
```

这个记录的格式是：



当需要定义一个类型为AREA的变量时，可以如下进行：

〈变量名〉〈记录名〉〈〔〈表达式〉〕〔,…〕〉

〈变量名〉〈记录名〉〈n〉DUP(〈〔〈表达式〉〕〔,…〕〉)

例如，

```
VAR1  AREA  <'B', , 6>
```

```
VAR2 AREA  20 DUP( <, 5, 10> )
```

其中，VAR1变量的类型是AREA，初值为“0100001001000110”；VAR2变量的类型也是AREA，但数据区中有20个这样的记录，每个记录的初值都是“0100000101011010”。

② 结构

用STRUC伪操作命令可以设计一个结构。结构和记录一样包含有许多字段，但结构可以允许多个字节，而记录只能是单字节或双字节。结构中的字段宽度以字节为单位，它们可以用数据区定义伪操作进行定义和赋初值，其变量名就是结构中的字段名。结构定义中只允许使用数据区定义伪操作以及注解，它的格式为：

```
<结构名>     STRUC
      ⋮
<字段名>     <数据区定义伪操作><表达式>
      ⋮
<结构名>     ENDS
```

例如，名为S的一个结构定义如下：

```
S       STRUC
FIELD1  DB   1, 2
FIELD2  DB   'DOBOSKY'
S       ENDS
```

当需要定义一个变量，其类型为结构S时，可用下列语句进行：

```
DBAREA   S <, 'ANY' >
```

这样，DBAREA就是一个格式为S的结构变量，它包含两个字段：FIELD1是两个字节，初值为1和2，FIELD 2是七个字节，初值为‘ANY’。

（4）变量的使用

如同上面所说，变量是存贮器数据区的符号表示，因此，指令中的存贮器操作数可以用变量的形式来给出。使用变量作为存贮器操作数时需要注意下列几个问题：

① 变量的类型必须与指令的要求相符合，否则汇编时将发生错误。例如，

```
WTABLE  DW 20 DUP( 0 )
        MOV AX, WTABLE
        MOV AL, WTABLE
```

其中，第1条MOV指令是正确的；第2条MOV指令不正确，因为WTABLE的类型是字，而指令要求进行的是字节传送操作。

② 变量仅仅对应于数据区中的第一个数据项，如果需要对数据区中其它数据项进行操作时，必须用地址表达式指出哪一个数据项是指令中的操作数。例如，

```
WTABLE DW 20 DUP( ? )
       MOV AX, WTABLE+38
       MOV BX, 38
K1:    SUB BX, 2
       ADD AX, WTABLE〔BX〕
       CMP BX, 0
```

```
        JNE K1
         ⋮
```

其中，WTABLE＋38指出数据区中的第20个数据项，WTABLE〔BX〕则用来逐次表示数据区中的第19，18，17，…，1个数据项，BX寄存器起着索引（下标）的作用，这就是索引寻址的典型汇编符号表示。对于更复杂的数据区域（如二维结构）中的数据项，汇编语言程序中可以使用下列形式的变址索引寻址方式来进行存取：

```
DATAITEM[BX][SI]
DATAITEM[BX][DI]
DATAITEM[BP][SI]
DATAITEM[BP][DI]
```

其中DATAITEM是为该数据区定义的变量名字，两个寄存器可分别用来指出数据项所在的行号和列号。

③ 被定义为结构类型的变量，其整体不能作为操作数使用，只有其中的某个字段才能作为指令的操作数，它的表示形式为：

〈结构变量名〉.〈字段名〉

这种操作数称为结构操作数。

④ 被定义为记录类型的变量，可以作为单字节或双字节存贮器操作数使用。借助于三种专用的运算，这种变量对于字节中指定的位（或位组）进行抽取或测试显得特别方便。三种专用运算为：

字段名　表示该字段最低位右移到记录最右端所需的移位次数

WIDTH　字段名　或　WIDTH　记录变量名　表示该字段或该记录的宽度（单位是位）

MASK　字段名　表示用于抽取该字段的一个屏蔽字节或屏蔽字，即指定字段的各位均为1、其它字段为0时该记录的值

下面是使用记录类型变量作为操作数的例子。先定义两个记录变量VAR1和VAR 2如下：

```
RECNAME RECORD FREE：1，EMPTY：1，INDEX：14
VAR1      RECNAME〈0，1，256〉
VAR2      RECNAME〈1，，1024〉
```

为了把VAR1中的EMPTY字段送到VAR 2 中的EMPTY字段中去，可以使用下列的指令序列：

```
MOV AX, VAR 1
AND AX, MASK EMPTY
MOV BX, VAR 2
AND BX, NOT MASK EMPTY
OR  AX, BX
MOV VAR 2, AX
```

⑤ 变量作为存贮器操作数使用时，它的段属性（段址）与所在指令使用的缺省段寄存器的内容必须相符，如果不符的话，则必须使用跨段前缀，否则指令无法从存贮器中取得正确的操作数进行操作。8086/8088中各种指令在形成操作数的物理地址时所使用的缺省的段寄存器规定如下：

* 使用到SP寄存器的所有指令（如PUSH，POP，CALL，RET，INT等），总是缺省地使用段寄存器SS。

* 字节串或字串处理指令中，若使用DI寄存器作变址用，则使用的段寄存器是ES。

* 循环、条件转移、无条件转移及调用指令中使用NEAR标号的，总使用CS寄存器作为段址，并不修改它的值，而JMP及CALL指令中使用FAR标号的，则向CS寄存器送入新的值。

* 所有其它情况下，指令中的存贮器操作数的物理地址的形成总是使用DS寄存器，只有在地址表达式中方括号内使用到BP寄存器时，缺省的段寄存器才是SS。

因此，在使用变量以前，必须把该变量的段地址预先送入相应的段寄存器中去，然后才能在指令中使用该变量。

（5）地址表达式

上面已经多次提到了地址表达式。汇编语言中表达式分成两类：数值表达式和地址表达式。数值表达式用来产生一个数值结果，它只有大小但没有属性，可以作为指令中的立即操作数和数据区中的初值使用。地址表达式的值是向量（存贮器地址），它由段属性、位移量属性及类型属性三个分量所组成，主要用来表示指令中的一个存贮器操作数或一个标号。

地址表达式是由变量、标号、常量、寄存器BP、BX、DI、SI的内容（用寄存器名以及方括号表示）以及一些运算符组成的。单个的变量、标号、寄存器内容是地址表达式的一个特例。

地址表达式中可以使用的运算符及使用规则如下。

① 加法和减法运算符（+，-）

变量或标号可以加上或减去某个结果为整数的数值表达式，其结果仍为变量或标号。它的类型不变，段址也不变，仅修改其位移量属性。由于数值表达式可以在地址表达式中出现，因此一切数值表达式的运算符都有可能在地址表达式中出现。

注意，同一段内的两个变量或两个标号可以相减，但结果不是地址而是一个数值，表示两者之间的距离。

② 方括号及寄存器BX、BP、SI和DI

BX、BP、SI和DI四个寄存器的内容既可作操作数使用，又可作为地址使用，区别这两种用法的标志是运算符〔〕。〔〕中可以出现上述四个寄存器中的一个或两个，但BX和BP或SI和DI不能同时出现。另外也可以加、减某个数值表达式，但不能有变量或标号出现。

单独使用的方括号〔〕，例如〔BX+7〕、〔SI-100H〕、〔DI+BP〕、〔BX+SI〕等，它的段分量是DS寄存器的内容或SS寄存器的内容（当〔〕中有BP出现时），它的位移量是〔〕中数值表达式的结果，而类型属性是未知的，应当由它所在的指令来确定。例如，

```
MOV AX, 〔BX〕      ；由于AX是一个字，所以〔BX〕的类型也是字（WORD）
MOV BH, 〔DI+4〕    ；由于BH是一个字节，所以〔DI+4〕的类型也是字节
                     （BYTE）
```

但是，在另外一些情况下，〔〕的类型又无法由所在指令确定，例如：

```
MOV 〔DI〕, 4       ；源操作数是字节4（8位）还是字4（16位）无法确定
JMP 〔BP〕          ；是段内间接转移还是段间间接转移无法确定
```

遇到这种情况，必须使用下面要介绍的PTR运算符来指出其类型，否则汇编时将会出错。

方括号前面伴有变量时，往往用来表示该变量对应的数据项的地址，〔〕起着下标或索引的作用。这种地址表达式的值，其段属性和类型属性与该变量相同，位移量则是该变量的位移量与〔〕中数值表达式的值相加的结果。注意，这种情况下，允许〔〕中只使用数值而不出现寄存器名。

③ 类型运算符PTR

PTR运算符用来明确指出某个变量、标号或地址表达式的类型属性，或者使它们临时兼有与原定义所不同的类型属性，但保持它们原来的段属性和位移量属性不变，它的格式是：

〈类型〉PTR〈地址表达式〉

其中，类型就是临时赋予地址表达式的新类型（如BYTE，WORD，DWORD，NEAR，FAR），它只在所在指令内有效。例如：

```
MOV BYTE PTR〔DI〕，4  ；字节4（8位）传送到内存
JMP WORD PTR〔BP〕     ；段内间接转移
JMP DWORD PTR〔BP〕    ；段间间接转移
```

又如：

```
VECTOR DW 1000DUP（0）
        ⋮
MOV AL，BYTE PTR VECTOR
```

表示将变量VECTOR（定义为字类型）临时按字节类型进行操作。当然，也可以使用伪操作命令LABEL为VECTOR再定义一个名字和一种新类型，但前者在偶尔使用时更为合适。

PTR运算符的另一个重要作用是：当汇编语言中出现先使用后定义的变量或标号时（称为“提前引用”），借助于PTR可以明确其类型属性，从而避免汇编时第1趟扫描与第二趟扫描不一致引起的所谓“相位出错”（Phase error）。

④ 跨段前缀等

运算符“：”用于临时给变量、标号或地址表达式指定一个段属性，它的格式为：

〈段寄存器〉：〈地址表达式〉
〈段名〉：〈地址表达式〉
〈组名〉：〈地址表达式〉

其中，段寄存器（CS、DS，SS，ES）、段名或组名就是地址表达式的临时段属性，它只在所在指令中有效，地址表达式的位移量和类型属性均不改变。组名是伪操作命令GROUP所定义的名字（见后）。

正如前述，存贮器操作数的物理地址，总是依据特定的指令码以及操作数，使用规定的段寄存器及操作数位移量来形成的。如果要访问一个在其它段内的操作数，并不用ASSUME伪操作命令来改变缺省段寄存器的内容时，可以使用段址运算符。例如，

```
MOV AX，ES：〔BX+SI〕
```

表示不再使用缺省的段寄存器DS，而是临时使用ES来形成物理地址，取ES段中的一个操作数。又如，

```
MOV CSEG：FARNAME〔BP〕，AX
```

表示目标操作数的物理地址是使用段名CSEG所在的段寄存器（由ASSUME命令预先确定），

而不是使用缺省的段寄存器SS形成的。

注意，汇编程序在遇到上述情况时将会自动生成一个“跨段前缀字节”（Segment Override Prefix），但当表达式中指出的段寄存器与缺省的段寄存器一致时，则前缀字节不予生成。另外，段寄存器CS、ES和涉及堆栈操作时使用的SS，均不能被跨越。例如，下面的写法是错误的：

```
MOVS   DS:BYTE  PTR [DI], ES:[SI]
```

跨段前缀符“：”在提前引用某个段中的数据时也很有用，这里不再细述。

下面几个运算符也是作用于地址表达式的，但结果是数值，它们在汇编语言程序设计中很有用处。

⑤运算符SEG，OFFSET，TYPE，SIZE，LENGTH，HIGH和LOW

它们的格式为：

〈运算符〉〈变量或标号〉

使用SEG运算符的结果是变量或标号的段址，OFFSET的结果是位移量，TYPE的结果是变量或标号的类型（变量的类型用字节个数表示，标号的类型用NEAR或FAR来表示），运算符SIZE和LENGTH只能作用于变量，其结果分别是数据区的字节总数和数据区中数据项的个数。运算符HIGH和LOW是字节分离运算符，它们取地址表达式或16位绝对值的高8位或低8位作为其结果。例如，

```
MOV AH, HIGH WORDVALUE
MOV AL, LOW  0FFFFH
```

下面是一个把向量中各元素进行累加的例子，它把向量BVEC的位移量送入基址寄存器BX中，然后通过索引寄存器SI来访问向量中的各个元素。

```
BVEC   DW 100 DUP(?)
       ASSUME DS:SEG BVEC
            ⋮
       MOV BX, OFFSET BVEC
       MOV SI, 0
MORE:
            ⋮
       ADD AX, [BX+SI]
            ⋮
       JMP MORE
```

最后，再介绍一个从数值表达式合成一个变量或标号的方法，它的格式为：

〈数值表达式1〉：〈类型〉 PTR〈数值表达式2〉

其中，数值表达式1用作合成的变量或标号的段属性，类型用来指出是何种变量或何种标号，数值表达式2则是合成变量或标号的位移量属性。

无论是地址表达式还是数值表达式，它们使用的运算符在汇编处理时都有一定的优先级。表2-6按从上到下的顺序列出了各种运算符的优先级。当然，与其它语言一样，圆括号（）可以改变表达式中运算的优先顺序。

表2-6　汇编语言中运算符的优先级

运　　算　　符
LENGTH，SIZE，WIDTH，MASK，以及（ ）、[ ]、< >中的项目，结构变量
段 取 代 运 算 符：
PTR，OFFSET，SEG，TYPE，THIS
HIGH，LOW
*，/，MOD，SHL，SHR
+，－（单目或双目运算符）
EQ，NE，LT，LE，GT，GE
逻辑NOT
逻辑AND
逻辑OR，XOR
SHORT

第二节　8086/8088指令系统及其汇编表示

8086/8088处理器大约可以执行100种不同的指令，这些指令可以分成六类，它们是：①数据传送指令，②算术运算指令，③逻辑运算指令，④串处理指令，⑤控制转移指令，⑥处理器控制指令。8086/8088所有指令的助记符及其分类参见表2－7。

表2-7　指令助记符及其分类

指令类别		助记符
数据传送	通用传送	MOV，POP，PUSH，XCHG，XLAT
	输入输出	IN，OUT
	目标地址传送	LEA，LDS，LES
	标志传送	LAHF，SAHF，PUSHF，POPF
算术运算	加　法	ADD，ADC，AAA，DAA，INC
	减　法	SUB，SBB，AAS，DAS，DEC，NEG，CMP
	乘　法	MUL，IMUL，AAM
	除　法	DIV，IDIV，AAD
	转　换	CBW，CWD

（续表2-7）

逻辑运算	逻　辑	NOT, AND, OR, XOR, TEST
	移　位	SHL, SAL, SHR, SAR, ROL, ROR, RCL, RCR
串处理	串操作	MOVS, CMPS, SCAS, LODS, STOS
	重复控制	REP, REPE/REPZ, REPNE/REPNZ
控制转移	无条件转移	JMP
	条件转移	JA/JNBE, JAE/JNB, JB/JNAE, JBE/JNA, JC, JCXZ, JE/JZ, JNS, JO, JS, JG/JNLE, JGE/JNL, JL/JNGE, JLE/JNG, JNC, JNE/JNZ, JNO, JNP/JPO, JP/JPE
	循　环	LOOP, LOOPE/LOOPZ, LOOPNE/LOOPNZ, JCXZ
	调用与中断	CALL, RET, INT, INTO, IRET
处理器控制		STI, CLI, HLT, WAIT, ESC, LOCK, NOP, STC, CLC, CMC, STD, CLD

下面以指令助记符的字母为序，分别介绍每条指令的功能，硬件执行过程，标志位变化情况，允许的操作数类型及其汇编表示，以及汇编后生成的目标代码字节数目，以便读者在编写汇编语言程序时参考。所使用的符号约定如下：

操作数的符号表示：

DEST	目标操作数
SRC	源操作数
TARGET	循环、转移和调用指令的操作数
register	寄存器操作数、字节或字
reg 8	寄存器操作数、字节
reg16	寄存器操作数、字
memory	存贮器操作数、字节或字
mem 8	类型为字节的存贮器操作数
mem16	类型为字的存贮器操作数
mem32	类型为双字的存贮器操作数
accumulator	累加器AL，AH或AX
seg-reg	段寄存器
immediate	立即操作数、字节或字
immed 8	8 位的立即操作数
short-label	短标号
near-label	NEAR 标号
far-label	FAR标号

标志位变化情况中使用的符号：

0	复位
1	置位
X	根据指令执行的结果而定
U	无确定值

R　恢复成以前被保护值

硬件执行过程中使用的符号：

（）　括号中的寄存器或存贮单元的内容

←　左端的值被右端所替换

&　逻辑乘

MOD　相除后取余数

∨　逻辑加

AAA　（十进制数加法的ASCII调整）

Flags O D I T S Z A P C U U U X U X	操作数类型	字节数	汇编举例
若((AL)&0FH)＞9或 (AF)＝1则 (AL)←(AL)+6; (AH)←(AH)+1; (AF)←1; (CF)←(AF); (AL)←(AL)&0FH	(无)	1	AAA

AAD　（十进制数除法的ASCII调整）

Flags O D I T S Z A P C U X X U X U	操作数类型	字节数	汇编举例
(AL)←(AH)*0AH+(AL) (AH)←0	(无)	2	AAD

AAM　（十进制数乘法的ASCII调整）

Flags O D I T S Z A P C U X X U X U	操作数类型	字节数	汇编举例
(AH)←(AL)/0AH (AL)←(AL) MOD 0AH	(无)	1	AAM

AAS　（十进制数减法的ASCII调整）

Flags O D I T S Z A P C U U U X U X	操作数类型	字节数	汇编举例
若((AL)&0FH)＞9 或(AF)＝1，则 (AL)←(AL)－6; (AH)←(AH)－1; (AF)←1; (CF)←(AF) (AL)←(AL)&0FH	(无)	1	AAS

ADC DEST, SRC （包括进位的加法）

Flags ODITSZAPC X XXXXX	操作数类型	字节数	汇编举例
若(CF)=1，则 (DEST)←(DEST)+ (SRC)+1 否则 (DEST)←(DEST) +(SRC)	register,register	2	ADC AX,SI
	register,memory	2-4	ADC DX,BETA[SI]
	memory,register	2-4	ADC ALPHA[BX][SI],DI
	register,immediate	3-4	ADC BX,256
	memory,immediate	3-6	ADC GAMMA,30H
	accumulator,immediate	2-3	ADC AL,5

ADD DEST, SRC （加法）

Flags ODITSZAPC X XXXXX	操作数类型	字节数	汇编举例
(DEST)←(DEST)+(SRC)	register,register	2	ADD CX,DX
	register,memory	2-4	ADD DI,[BX].ALPHA
	memory,register	2-4	ADD TEMP,CL
	register,immediate	3-4	ADD CL,2
	memory,immediate	3-6	ADD ALPHA,2
	accumulator,immediate	2-3	ADD AX,200

AND DEST, SRC （逻辑乘）

FLags ODITSZAPC 0 XXUX0	操作数类型	字节数	汇编举例
(DEST)←(DEST)&(SRC) (CF)←0; (OF)←0	register,register	2	AND AL,BL
	register,memory	2-4	AND CX,FLAG_WORD
	memory,register	2-4	AND ASCII[DI],AL
	register,immediate	3-4	AND CX,0F0H
	memory,immediate	3-6	AND BETA,01H
	accumulator,immediate	2-3	AND AX,01010000B

CALL TARGET （过程调用）

Flags ODIT ZAPC	操作数类型	字节数	汇编举例
段间调用执行①和②， 段内调用只执行② ①(sp)←(sp)-2 ((sp)+1,(sp))←(cs) (cs)←新段址 ②(sp)←(sp)-2 ((sp)+1,(sp))←(Ip) (Ip)←位移量	near-proc	3	CALL NEAR_PROC
	far-proc	5	CALL FAR_PROC
	memptr16	2-4	CALL PROC_TABLE[SI]
	regptr16	2	CALL AX
	memptr32	2-4	CALL [BX].TASK[SI]

CBW （把字节转换为字）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
若(AL)<80H,则(AH)←0 否则(AH)←0FFH	(无)	1	CBW

CLC （清除进位标志）

Flags ODITSZAPC 0	操作数类型	字节数	汇编举例
(CF)←0	(无)	1	CLC

CLD （清除方向标志）

Flags ODITSZAPC 0	操作数类型	字节数	汇编举例
(DF)←0	(无)	1	CLD

CLI （清除中断标志）

FLags ODITSZAPC 0	操作数类型	字节数	汇编举例
(IF)←0	(无)	1	CLI

CMC （进位标志取反）

FLags ODITSZAPC X	操作数类型	字节数	汇编举例
若(CF)=0，则(CF)←1, 否则(CF)←0	(无)	1	CMC

CMP DEST, SRC （比较）

Flags ODITSZAPC X XXXXX	操作数类型	字节数	汇编举例
(DEST)-(SRC) 只改变Flags,不改变操作数	register,register	2	CMP BX,CX
	register,memory	2-4	CMP DH,ALPHA
	memory,register	2-4	CMP [BP+2],SI
	register,immediate	3-4	CMP BL,02H
	memory,immeciate	3-6	CMP [BX].RADAR[DI],3420H
	accumulator,immediate	2-3	CMP AL,00010000B

CMPS　　DEST, SRC　　（字节串或字串比较指令）

Flags ODITSZAPC X　XXXXX	操作数类型	字节数	汇编举例
(DEST)-(SRC) 若(DF)=0则 (SI)←(SI)+1(或2) (DI)←(DI)+1(或2) 否则 (SI)←(SI)-1(或2) (DI)←(DI)-1(或2)	dest-string,source-string (repeat)dest-string,source-string	1 1	CMPS BUFF1,BUFF2 REPE CMPS ID,KEY

串比较指令在汇编语言中的典型用法为：

```
CLD
MOV SI, OFFSET STRING 1
MOV DI, OFFSET STRING 2
CMPS STRING 1 , STRLNG 2
```

其中，STRING 1 和STRING 2 是代表字节串或字串的两个变量名，它们作为CMPS指令操作数仅仅是为了让汇编程序了解其类型（是字节串比较还是字串比较），并检验其段属性的正确性，机器执行CMPS指令时并不需要使用这两个操作数，后面的MOVS，SCAS，LODS和STOS都属于这种情况。

CWD　　（把字转换成双字）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
若(AX)<8000H,则(DX)←0, 否则(DX)←0FFFFH	(无)	1	CWD

DAA　　（紧凑十进制数加法的调整）

Flags ODITSZAPC X　XXXXX	操作数类型	字节数	汇编举例
若((AL)&0FH)>9或(AF)=1, 则(AL)←(AL)+6; (AF)←1 若(AL)>9FH或(CF)=1,则 (AL)←(AL)+60H; (CF)←1	(无)	1	DAA

DAS （紧凑十进制数减法的调整）

Flags O D I T S Z A P C U X X X X X	操作数类型	字节数	汇编举例
若((AL)&0FH)>9或(AF)=1,则 (AL)←(AL)-6; (AF)←1 若(AL)>9FH或(CF)=1,则 (AL)←(AL)-60H; (CF)←1	(无)	1	DAS

DEC **DEST** （减量指令）

Flags O D I T S Z A P C X X X X X	操作数类型	字节数	汇编举例
(DEST)←(DEST)-1	reg16	1	DEC AX
	reg8	2	DEC AL
	memory	2-4	DEC ARRAY[SI]

DIV **SRC** （无符号整数除法）

Flags O D I T S Z A P C U U U U U	操作数类型	字节数	汇编举例
若为字节除法，则	reg8	2	DIV CL
(AL)←(AX)/(SRC)	reg16	2	DIV BX
(AH)←(AX)MOD(SRC)	mem8	2-4	DIV ALPHA
若为字除法，则	mem16	2-4	DIV TABLE[SI]
(AX)←(DX,AX)/(SRC)			
(DX)←(DX,AX)MOD(SRC)			

ESC 外操作码，SRC（向外处理器提供指令）

Flags O D I T S Z A P C	操作数类型	字节数	汇编举例
	immediate,memory	2-4	ESC 6,ARRAY[SI]
	immediate,register	2	ESC 20,AL

ESC指令用来向外部的其它处理器提供一条指令。立即数表示外操作码（0—63），它连同源操作数一起放在总线上，供外处理器取用。

HLT　　（停机，直到出现外部中断或系统复位为止）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
	(无)	1	HLT

IDIV　SRC　　（整数除法）

Flags ODITSZAPC U　　UUUUU	操作数类型	字节数	汇编举例
与DIV相同，但被除数和除数都是有符号位的整数，余数的符号与被除数相同	reg8	2	IDIV BL
	reg16	2	IDIV CX
	mem8	2-4	IDIV DIVISOR_BYTE[SI]
	mem16	2-4	IDIV [BX].DIVISOR_WORD

IMUL　SRC　　（整数乘法）

Flags ODITSZAPC X　　UUUUX	操作数类型	字节数	汇编举例
若为字节乘法，则 (AH,AL)←(AL) * (SRC) 若为字乘法，则 (DX,AX)←(AX) * (SRC) 两者均为有符号的整数	reg8	2	IMUL CL
	reg16	2	IMUL BX
	mem8	2-4	IMUL RATE_BYTE
	mem16	2-4	IMUL RATE_WORD[BP][DI]

IN　　累加器，端口号　　（输入）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
从指定端口取一个字节或字送入AL或AX	accumulator,immed8	2	IN AL,0EAH
	accmulator,DX	1	IN AX,DX

INC　　DEST　　（增量）

Flags ODITSZAPC X　　XXXX	操作数类型	字节数	汇编举例
(DEST)←(DEST)+1	reg16	1	INC CX
	reg8	2	INC BL
	memory	2-4	INC ALPHA[DI][BX]

INT　　类型　　（中断）

Flags ODITSZAPC 0 0	操作数类型	字节数	汇编举例
(SP)←(SP)－2 ((SP)＋1,(SP))←Flags (IF)←0,(TF)←0 (SP)←(SP)－2 ((SP)＋1,(SP))←(CS) (CS)←(类型＊4＋2) (SP)←(SP)－2 ((SP)＋1,(SP))←IP (IP)←(类型＊4)	immed8(type＝3) immed8(type≠3)	1 2	INT 3 INT 67

INTO　　（溢出中断）

Flags ODITSZAPC 0 0	操作数类型	字节数	汇编举例
若OF＝0,则不操作 若OF＝1,则按“INT 4”执行	(无)	1	INTO

IRET　　（中断返回）

Flags ODITSZAPC RRRRRRRRR	操作数类型	字节数	汇编举例
(IP)←((SP)＋1,(SP)) (SP)←(SP)＋2 (CS)←((SP)＋1,(SP)) (SP)←(SP)＋2 Flags←((SP)＋1,(SP)) (SP)←(SP)＋2	(无)	1	IRET

下面是19条条件转移指令，它们都使用短标号作为操作数，目标代码是两个字节，执行后Flags寄存器各位均保持不变，所不同的仅仅是控制转移与否的条件各不相同。当满足转移条件时，处理器将根据汇编程序在汇编时计算出来的本指令所在位置与短标号之间的距离进行相对转移，即：

（IP）←（IP）＋距离

这19条指令的汇编表示均为：

〈指令助记符〉〈短标号〉

其中有些指令有两个助记符。下面是它们的指令助记符、转移条件及指令含义一览表。

指令助记符	转移条件	指令含义
JA／JNBE	(CF) = 0且(ZF) = 0	不低于等于转移，高于转移
JAE／JNB	(CF) = 0	高于或等于转移，不低于转移
JB／JNAE	(CF) = 1	低于转移，不高于等于转移
JBE／JNA	(CF) = 1或(ZF) = 1	低于等于转移，不高于转移
JC	(CF) = 1	进位转移
JCXZ	(CX) = 0	CX为零转移
JE／JZ	(ZF) = 1	等于转移
JG／JNLE	(ZF) = 0且(SF) = (OF)	大于转移，不小于等于移转
JGE／JNL	(SF) = (OF)	大于等于转移，不小于转移
JL／JNGE	(SF) ≠ (OF)	小于转移，不大于等于转移
JLE／JNG	(ZF) = 1或(SF)≠(OF)	小于等于转移，不大于转移
JNC	(CF) = 0	无进位转移
JNE／JNZ	(ZF) = 0	不等于转移
JNO	(OF) = 0	不溢出转移
JNP／JPO	(PF) = 0	奇校验转移
JNS	(SF) = 0	符号位为零转移
JO	(OF) = 1	溢出转移
JP／JPE	(PF) = 1	偶校验转移
JS	(SF) = 1	符号位为1转移

条件转移中的前四条指令测试的是无符号整数运算的结果(标志CF和ZF)，由于无符号整数常常用于表示地址，因此这些指令的用途是根据两个地址的高低或相等与否进行转移。

JMP　TARGET　（无条件转移）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
若为段内直接转移，则 (IP) ← (IP) +与标号的距离；	short-label	2	JMP SHORT
	near-label	3	JMP WITHIN_SEGMENT
若为段内间接转移，则 (IP) ←操作数的位移量；	far-label	5	JMP FAR_LABEL
	memptr16	2-4	JMP [BX].TARGET
若为段间转移，则 (IP) ←操作数的位移量 (CS) ←操作数的段址	regptr16	2	JMP CX
	memptr32	2-4	JMP OTHER_SEG[SI]

LAHF　（把标志装入AH）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
(SF),(ZF),(AF),(PF)和(CF)装入AH寄存器的第7，6，4，2，0位	(无)	1	LAHF

LDS　寄存器，SRC　（装入数据段寄存器DS）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
DS，寄存器← (SRC)	reg16,mem32	2-4	LDS SI,DATA.SEG[DI]

LEA　寄存器，SRC　（装入有效地址）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
寄存器← (SRC)	reg16,mem16	2-4	LEA BX,[BP][DI]

LES　寄存器，SRC　（装入附加数据段寄存器ES）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
ES，寄存器← (SRC)	reg16,mem32	2-4	LES DI,[BX].TEXT_BUFF

LOCK　（封锁总线）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
发出封锁总线信号	(无)	1	LOCK XCHG FLAG,AL

本指令用作其它指令的前缀，使指令执行期间发出封锁总线的信号。

LODS　SRC　（装入字节串或字串）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
AL或AX← (SRC) 若(DF) = 0 则(SI)←(SI) + 1 (或2)； 若(DF) = 1 则(SI)←(SI) − 1 (或2)	source-string (repeat)source-string	1 1	LODS CUSTOMER_NAME REP LODS NAME

下面的三条循环指令，它们的操作数都是短标号，目标代码是两个字节，Flags寄存器不变。机器执行这些指令时，首先把CX内容减1，然后当满足循环条件时再执行：

（IP）←（IP）+（循环指令与标号间的距离+2）

它们的汇编表示、循环条件及指令含义如下：

汇 编 表 示	循 环 条 件	指 令 含 义
LOOP 短标号	(CX) ≠0	CX不等于0，则循环
LOOPE／LOOPZ 短标号	(CX)≠0 且(ZF) = 1	CX不等于0且ZF标志为1，则循环
LOOPNE／LOOPNZ 短标号	(CX) ≠0 且(ZF) = 0	CX不等于0且ZF标志为0，则循环

MOV DEST，SRC（字节或字的传送指令）

Flags ODITSZAPC	操作数类型	字节数	汇 编 举 例
(DEST) ← (SRC)	memory,accumulator	3	MOV ARRAY[SI],AL
	accumulator,memory	3	MOV AX,TEMP_RESULT
	register,register	2	MOV AX,CX,
	rgister,memory	2-4	MOV BP,STACK_TOP
	memory,register	2-4	MOV COUNT[DI],CX
	register,immediate	2-3	MOV CL,2
	memory,immediate	3-6	MOV MASK[BX][SI],2CH
	seg-reg,reg16	2	MOV ES,CX
	seg-reg,mem16	2-4	MOV DS,SEGMENT_BASE
	reg16,seg-reg	2	MOV BP,SS
	memory,seg-reg	2-4	MOV [BX].SEG_SAVE,CS

MOVS DEST，SRC（字节串或字串传送指令）

Flags ODITSZAPC	操作数类型	字节数	汇 编 举 例
(DEST) ← (SRC)	dest-string,source-string	1	MOVS LINE,EDIT__DAT
若(DF) = 0，则(SI)←(SI) + 1(或2)	(repeat) dest-string,	1	REP MOVS SCREEN,
(DI) ← (DI) + 1 (或2)	source-string		BUFFER
若(DF) = 1，则(SI)←(SI) − 1(或2)			
(DI) ← (DI) − 1 (或2)			

MOVS指令中的操作数若不写出来，那末为了让汇编程序了解是字节传送还是字传送，必须使用指令助记符MOVSB（字节传送）或MOVSW（字传送）。

MUL SRC（无符号整数乘法）

Flags ODITSZAPC X UUUUX	操作数类型	字节数	汇 编 举 例
字节乘法：	reg8	2	MUL BL
(AX) ← (AL) • (SRC)	reg16	2	MUL CX
字乘法：	mem8	2-4	MUL MONTH[SI]
(DX,AX) ← (AX) • (SRC)	mem16	2-4	MUL BAUD__RATE

NEG　DEST（取负）

Flags ODITSZAPC X XXXX1	操作数类型	字节数	汇编举例
(DEST) ← 0-(DEST)	register memory	2 2-4	NEG AL NEG MULTIPLIER

NOP　（空操作）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
	(无)	1	NOP

NOT　DEST（取反）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
字节：(DEST) ←0FFH-(DEST) 字：(DEST) ←0FFFFH-(DEST)	register memory	2 2-4	NOT AX NOT CHARACTER

OR　DEST, SRC（逻辑加）

Flags ODITSZAPC 0 XXUX0	操作数类型	字节数	汇编举例
(DEST) ← (DEST)∨(SRC)	register,register	2	OR AL,BL
	register,memory	2-4	OR DX,PORT_ID[DI]
	memory,register	2-4	OR FLAG_BYTE,CL
	accumulator,immediate	2-3	OR AL,01101100B
	register,immediate	3-4	OR CX,01H
	memory,immediate	3-6	OR [BX].CMD_WORD,0CFH

OUT　端口号，累加器（输出）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
把AL或AX中的字节或字向指定端口送出	immed8,accumulator DX,accumulator	2 1	OUT 44,AX OUT DX,AL

POP　　DEST（出栈）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
(DEST) ← ((SP) + 1,(SP)) (SP) ← (SP) + 2	register seg-reg(CS不可) memory	1 1 2-4	POP DX POP DS POP PARAMETER

POPF　　（出栈并送入标志寄存器）

Flags ODI TSZAPC RRRRRRRRR	操作数类型	字节数	汇编举例
(Flags) ← ((SP) + 1,(SP)) (SP) ← (SP) + 2	(无)	1	POPF

PUSH　　SRC（入栈）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
(SP) ← (SP) − 2 ((SP) + 1, (SP)(←)SRC)	register seg-reg(CS可) memory	1 1 2-4	PUSH SI PUSH ES PUSH RETURN_CODE[SI]

PUSHF　　（标志寄存器入栈）

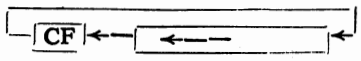
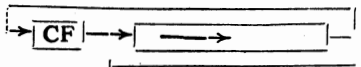
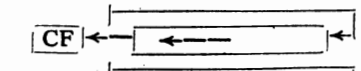
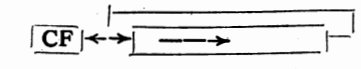
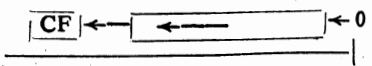
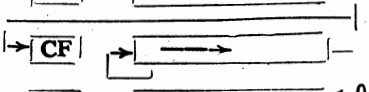
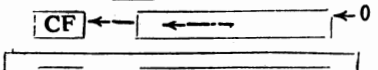
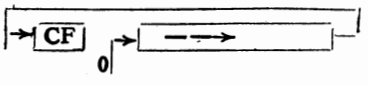
Flags ODITSZAPC	操作数类型	字节数	汇编举例
(SP) ← (SP) − 2 ((SP) + 1, (SP)) ←Flags		1	PUSHF

下面的八条移位指令都用来实现目标操作数（字节或字）的移位操作，移位次数或者是1位，或者由CL寄存器的内容指出。它们所允许的操作数类型以及汇编表示如下所示：

DEST, COUNT	目标字节数	汇编举例
register,1	2	RCL CX,1
register,CL	2	RCL AL,CL
memory,1	2-4	RCL ALPHA,1
memory,CL	2-4	RCL [BP].PARM,CL

当然，不同的移位指令其操作不同，功能也不一样。下表是八条移位指令的助记符、功能、

机内操作、标志寄存器状态变化的一览表。

助记符	指令功能	机内操作示意	Flags O D I T S Z A P C
RCL	通过进位的循环左移		X X
RCR	通过进位的循环右移		X X
ROL	循环左移		X X
ROR	循环右移		X X
SAL	算术左移		X X
SAR	算术右移		O X X U X X
SHL	逻辑左移		X X
SHR	逻辑右移		X X

其中，OF标志位按如下规则确定，当移位次数＝1时，若左移且（CF）≒(DEST)的最高位，或者右移且（DEST）的最高位与次高位不相等，则（OF）←1，否则（OF）←0。当移位次数≒1时，（OF）结果不确定。

下列三条重复指令没有操作数，而且在汇编语言中都作为串操作指令的前缀指令使用。它们的助记符、功能、适用的串操作指令、汇编举例等如下表所示：

助记符	指令功能	适用的串操作	字节数	汇编举例
REP	(CX) ≒0，则执行其后的串操作	MOVS	1	REP MOVS DEST,SRCE
REPE/REPZ	(CX)≒0或(ZF)＝1，则执行其后的串操作	CMPS, SCAS	1	REPE CMPS DATA,KEY
REPNE/REPNZ	(CX)≒0或(ZF)＝0，则执行其后的串操作	CMPS, SCAS	1	REPNE SCAS INPUTLINE

RET　出栈字节数（从过程返回）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
(IP)←((SP)＋1，(SP))	(intra-segment,no pop)	1	RET
(SP)←(SP)＋2	(intra-segment,pop)	3	RET 4
若为段间返回，则还应	(inter-segment, no pop)	1	RET
(CS)←((SP)＋1,(SP))	(inter-segment, pop)	3	RET 2
(SP)←(SP)＋2			
(SP)←(SP)＋出栈字节数			

SAHF　（把AH内容送入标志寄存器）

Flags ODI TSZA PC RRRRR	操作数类型	字节数	汇编举例
(AH)的第7，6，4，2，0位分别送入 (SF)，(ZP)，(AF)，(PF)，(CF)	(无)	1	SAHF

SBB　DEST，SRC（带借位的减法）

Flags ODITSZAPC X　XXXXX	操作数类型	字节数	汇编举例
若 (CF) = 1，则	register,register	2	SBB BX,CX
(DEST) ← (DEST) - (SRC) - 1	register,memory	2-4	SBB DI,[BX].PAYMENT
否则 (DEST) ← (DEST) - (SRC)	memory,register	2-4	SBB BALANCE,AX
	accumulator,immediate	2-3	SBB AX,2
	register,immediate	3-4	SBB CL,1
	memory,immediate	3-6	SBB COUNT[SI],10

SCAS　DEST（字节串或字串扫描指令）

Flags ODIT S Z APC X　XXXXX	操作数类型	字节数	汇编举例
(AL) 或 (AX) 与 (DEST) 比较	dest-string	1	SCAS INPUT_LINE
若 (DF) =0，则 (DI)←(DI)+1(或2)	(repeat)dest-string	1	REPNE SCAS BUFFER
否则 (DI) ← (DI) - 1 (或2)			

STC　（置进位标志）

Flags ODITSZAPC 1	操作数类型	字节数	汇编举例
(CF) ←1	(无)	1	STC

STD　（置方向标志）

Flags ODITSZAPC 1	操作数类型	字节数	汇编举例
(DF) ←1	(无)	1	STD

STI （置中断允许标志）

Flags ODITSZAPC 1	操作数类型	字节数	汇编举例
(IF) ← 1	(无)	1	STI

STOS **DEST**（保存字节串或字串）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
(DEST) ← (AL) 或 (AX)	dest-string	1	STOS PRINT_LINE
若 (DF) =0, 则(DI)←(DI)+1(或2) 否则 (DI) ← (DI) − 1 (或2)	(repeat)dest-string	1	REP STOS DISPLAY

SUB DEST, SRC（减法）

Flags ODIT SZAPC X XXXXX	操作数类型	字节数	汇编举例
(DEST) ← (DEST) − (SRC)	register,register	2	SUB CX,BX
	register,memory	2-4	SUB DX,MATH_TOTAL[SI]
	memory,register	2-4	SUB [BP+2],CL
	accumulator,immediate	2-3	SUB AL,10
	register,immediate	3-4	SUB SI,5280
	memory,immediate	3-6	SUB [BP],BALANCE,1000

TEST DEST, SRC（测试）

Flags ODIT SZA P C 0 XXUX 0	操作数类型	字节数	汇编举例
(DEST) & (SRC) 决定(SF), (ZF), (PF) 之值	register,register	2	TEST SI,DI
	register,memory	2-4	TEST SI,END COUNT
	accumulator,immediate	2-3	TEST AL,00100000B
	register,immediate	3-4	TEST BX,0CC4H
	memory,immediate	3-6	TEST RETURN_CODE,01H

WAIT （等待）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
若芯片上$\overline{\text{TEST}}$电平为高, 则CPU进入等待状态	(无)	1	WAIT

XCHG　DEST, SRC　（交换）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
(DEST) 与 (SRC) 交换	accumulator, reg16	1	XCHG AX,BX
	memory, register	2-4	XCHG SEMAPHORE,AX
	register, register	2	XCHG AL,BL

XLAT　SRC　（翻译）

Flags ODITSZAPC	操作数类型	字节数	汇编举例
(AL) ← ((BX) + (AL))	source-table	1	XLAT ASCII_TAB

翻译指令中的操作数（类型为字节）实际上已预先送入BX寄存器中，写在XLAT指令中是为了汇编程序检查类型正确性时使用的。

XOR　DEST, SRC　（异或）

Flags ODITSZAPC 0 XXUX0	操作数类型	字节数	汇编举例
	register, register	2	XOR CX,BX
	register, memory	2-4	XOR CL,MASK_BYTE
	memory, register	2-4	XOR ALPHA[SI],DX
	accumulator, immediate	2-3	XOR AL,01000010B
	register, immediate	3-4	XOR SI,00C2H
	memory, immediate	3-6	XOR RETURN_CODE,0D2H

第三节　伪操作命令

汇编语言中的指令性语句使用的操作符是8086/8088处理器的指令助记符。在汇编过程中，汇编程序把它们翻译成相应的指令代码，但并不执行。汇编语言中的另一类语句——指示性语句，所使用的是不与任何处理器指令对应的伪操作命令。这些伪操作命令由汇编程序在汇编时解释执行，它们主要用来“控制指挥”汇编程序如何把指令性语句翻译成目标代码。指示性语句本身除了可以申请分配一部分存贮空间用作数据区和堆栈区之外，不产生任何目标代码。

MICROSOFT公司的8086/8088宏汇编语言提供了多种伪操作命令，它们的主要功能如下：

* 变量定义及存贮器申请
* 过程定义
* 符号的定义，如符号常量、标号、变量名及其它各种符号名
* 程序模块的定义与通讯

* 程序分段及存贮器分配
* 宏定义及宏调用
* 条件汇编
* 格式控制、列表控制及其它功能

下面将分组介绍有关的伪操作命令。

1. 变量定义及存贮器申请

程序通常由两部分组成：指令及其处理对象——数据。用汇编语言编制程序时，既要为处理的数据设计其结构、赋予初值、安排内存以及定义符号名字等，又要为处理过程中必使用的数据工作区（如中间工作单元，栈区）申请存贮器空间。这些都是由一些伪操作指来完成的。关于这些伪操作指令的功能、格式都已在第1节作过介绍，这里就不再赘述。

（1）DB

用于申请一个数据项为字节的数据区，需要时可以用数值表达式赋予初值。如果该数据定义作为一个变量，则变量类型是BYTE。

（2）DW

同DB，但数据项为字，允许用地址表达式为数据项赋初值（取位移量属性）。变量类为WORD。

（3）DD

同DB，但数据项为双字，允许用地址表达式为数据项赋初值（取段属性及位移量属）。变量类型为DWORD。

（4）DQ

同DB，但数据项为四字。变量类型为DWORD。

（5）DT

同DB，但数据项为十个字节的紧凑十进制数。变量类型为TWORD。

（6）RECORD

用于设计一个单字节或双字节记录的格式，以便此后定义类型为该记录的变量（申请内并赋初值）。

（7）STRUC

用于设计一个包含有多个字段的结构的格式，以便此后在程序中定义类型为该结构的变（申请内存及赋初值）。

2. 过程定义伪操作

在程序中，把具有一定功能的程序段设计成为一个过程（子程序），是良好的程序设计法之一。使用过程不仅减少了目标代码的数量，便于实现模块化的程序结构，并且为建立使用程序库提供了方便。汇编语言中，必须使用过程定义伪操作命令（PROC和ENDP）来义一个过程，然后再通过机器指令CALL调用该过程。

过程定义伪操作PROC和ENDP的使用格式如下：

```
〈过程名〉    PROC    NEAR或FAR
               ⋮
             RET
               ⋮
〈过程名〉    ENDP
```

其中，〈过程名〉是为该过程起的名字。在调用这个过程的时候，它在CALL指令中起着标号的作用。除了段属性、位移量属性之外，它的类型属性分成NEAR和FAR两种，不明确指出时为NEAR类型，过程定义的最后一个语句必须是名字相同的ENDP语句。过程中原则上必须包含返回指令RET，但可以多于一条，也不一定要是最末一条机器指令。

指出一个过程是NEAR还是FAR，对汇编程序十分重要，这是因为：

① 当汇编程序汇编到调用这个过程的CALL语句时，汇编程序会清楚地知道，应该为CALL指令生成段内调用还是段间调用的目标代码（三字节还是五字节）。

② 当汇编程序汇编到该过程定义中的返回指令RET时，就知道应该为它生成段内返回还是段间返回的指令目标码（均为一字节，但操作码不同）。

过程允许嵌套进行定义，也可以嵌套地进行调用。例如：

```
FARNAME     PROC  FAR
              CALL NEARNAME
              RET
FARNAME     ENDP
              ⋮
NEARNAME    PROC
              ⋮
SUBNAME     PROC
              ⋮
SUBNAME     ENDP
              CALL SUBNAME
              ⋮
              RET
NEARNAME    ENDP
```

如果所定义的过程是由MS-DOS直接装入内存并启动执行的，则该过程必须定义为FAR过程，如同本章第一节例子中所示的那样。

3. 符号定义伪操作

汇编语言中的所有变量名、标号名、过程名、记录名、指令助记符、寄存器名等等统称为符号。这些符号可以通过伪操作重新命名，也可以通过伪操作为其定义其它名字及新的类型属性，因而给程序设计带来很大的灵活性。

（1）EQU

EQU伪操作命令用来为常量、表达式、其它符号等定义一个符号名，但它并不申请分配内存。通过EQU命令的使用可以使汇编语言程序简洁明了。例如：

① 为常量定义一个符号名，以便在程序中使用符号来表示常量。它的格式为：

〈符号常量名〉 EQU 〈数值表达式〉

例如，

```
THREE    EQU    3
COUNT    EQU    THREE * 2 - 1
```

② 给变量或标号定义新的类型属性并起一个新的名字。它的格式为：

〈变量名或标号〉EQU 〔〈类型〉PTR〕〈变量或标号〉

例如，

```
BYTES      DB       4 DUP(?)
FIRSTW     EQU      WORD PTR BYTES
FIRSTDW    EQU      DWORD PTR BYTES
                    ⋮
INCHES     MOV      BYTES, AL
                    ⋮
MILES:     EQU      FAR  PTR  INCHES
                    ⋮
           JMP      MILES
```

③ 可以给由地址表达式指出的任意存贮器单元定义一个名字。其格式为：

〈符号名〉 EQU 〈地址表达式〉

其中，符号名或者是变量或者是标号，取决于地址表达式的类型。例如：

```
XYZ      EQU      ALPHA 〔SI〕 + 3
A        EQU      ARRAY 〔BX〕 〔SI〕
P        EQU      ES: ALPHA
```

这里再介绍一个在地址表达式中使用的运算符“THIS”。THIS用于产生一个指定类型的存贮器操作数，它的段址和位移量是汇编时下一个可分配单元的段址及位移量，其类型在THIS中指出。使用THIS运算符产生的操作数，一般总是通过EQU伪操作命令为它定义一个名字。例如，

```
MYBYTE    EQU    THIS    BYTE
MYWORD    DW     ?
```

表示利用THIS运算符和EQU命令定义了一个类型为BYTE的变量MYBYTE，其段址和位移量与紧随其后的变量MYWORD相同。其实，使用PTR运算符也可达到同样目的：

```
MYBYTE    EQU    BYTE    PTR    MYWORD
```

利用THIS可以很方便地定义一个类型为FAR的标号，例如：

```
MILES     EQU    THIS  FAR
          MOV    BYTES, AL
                 ⋮
          JMP    MILES
```

不言而喻，用THIS运算符定义FAR标号比使用PTR运算符显得更加方便。

④ EQU伪操作还用来为汇编语言中的任何符号(如指令助记符、寄存器名、宏定义名、变量名、标号、段名、组名、记录名、结构名等）定义一个新的名字（替补名字），它的格式为：

〈替补符号名〉 EQU 〈符号名〉

例如，

```
COUNT    EQU    CX          (寄存器)
RADD     EQU    ADDR        (宏定义名)
```

在汇编语言中，一个符号允许有多个替补名，符号名和它的替补名作用完全相同。

使用EQU伪操作时必须注意，EQU左端的符号名不能是程序中已经定义过的符号名。另外，在定义一个结构时，STRUC和ENDS之间不准使用EQU命令。

（2）=

除了下列两点差别之外，=伪操作命令与EQU伪操作命令的功能完全相同。

① 使用=定义的符号名可以被重新定义，使其具有新的值。例如下面的语句都是正确的：

```
EMP  =  6
EMP  =  7
EMP  =  EMP+1
```

② 习惯上，=伪操作在程序中主要用来定义符号常量。

（3）LABEL

LABEL伪操作命令用于向汇编程序指出，为当前存贮单元定义一个指定类型的变量或标号。它的格式为：

〈变量名或标号〉 LABEL 〈类型〉

其中，类型可以是BYTE、WORD、DWORD、结构名、记录名、NEAR以及FAR。例如：

```
PUFF    LABEL    BYTE
        DB       21
```

它等价于

```
PUFF    DB       21
```

又如：

```
TRANS:  MOV      AX, CX
```

可以用下面两条语句来代替：

```
TRANS:  LABEL    NEAR
        MOV      AX, CX
```

可见，LABEL的功能与EQU THIS相似。

使用LABEL伪操作可以给数据区或程序段定义新的类型和新的名字，从而使它们适合于不同的应用。例如：

```
BARRAY   LABEL      BYTE
ARRAY    DW         100 DUP(0)
         ⋮
         ADD        AL, BARRAY[99]；取其中第100个字节作加法
         ⋮
         ADD        AX, ARRAY〔98〕；  取其中第50个字作加法
```

使用LABEL语句后同一数据区的内容，既可作为字节又可作为字参加运算。当然，若不使用LABEL语句的话，第1条加法指令改成

```
ADD    AL, BYTE PTR ARRAY〔99〕
```

也是一样的。

对于标号的处理也是如此。例如，

```
SUBRTE    LABEL    FAR
SUBRT:    MOV      AX, BX
            ⋮
```

表示以MOV指令开始的一段代码，既可以从同一段内的转移指令转来执行（使用NEAR标号SUBRT），又可以从其它段转来此处执行（使用FAR标号SUBRTF）。必须注意，用PROC和ENDP伪操作定义的一个过程，可以用LABEL命令定义多个不同的入口。但其类型必须始终保持一致，否则返回指令就无法正确返回。例如，

```
ENTRY 1    PROC    FAR
             ⋮
ENTRY 2    LA EL   FAR
             ⋮
ENTRY 3    LABEL   FAR
             ⋮
           RET
ENTRY 1    ENDP
```

利用LABEL命令还可以为数据区定义标号，或者为代码区定义变量名。这些做法只有在特定的场合下才使用，这里不再赘述。

4．程序模块的定义与通讯

为了有效地开发软件，汇编语言还提供了一些功能，可以把程序划分成许多模块，并对每个模块独立地进行汇编和调试。图2-2是一个汇编语言程序划分成两级子模块的示意图。从图中可以看出，程序模块的汇编及调试过程，一般是从低层到高层逐步进行的，所有模块汇编完毕后，可以通过连接程序（LINK）连接装配成为一个完整的可执行程序。

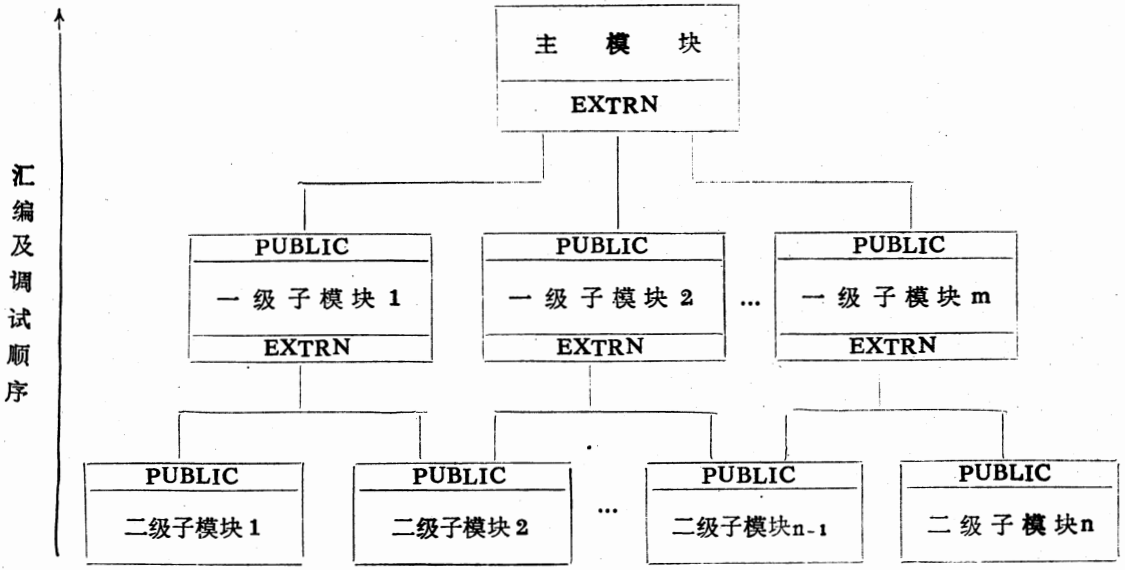


图2-2 汇编语言程序的模块结构

为了提供上述模块式程序设计的功能，汇编语言除了必须具备模块划分并命名的能力之外，更重要的是应当具有模块与模块之间共享数据和程序代码的能力，也就是模块之间通讯

的能力。所有这些，汇编语言都是通过下述四条伪操作命令来完成的。

（1）程序模块的定义（NAME，END）

定义一个模块需要使用NAME和END两条伪操作命令。它的格式是：

```
NAME    〈模块名〉
  ⋮
（模块中的所有语句）
  ⋮
END     〈标号〉
```

其中，模块命名语句（NAME）可以缺省。这时，该模块的命令规则为：如果模块中使用了TITLE语句（列表输出的页标题命令），则TITLE语句中的页标题的前六个字符就是模块名；如果模块中没有使用TITLE语句，则该模块的源程序文件名就是模块名。

汇编处理时，一个模块就是一个独立的汇编单位。汇编处理只进行到模块结束语句（END）为止，此后的所有语句将被忽略。如果是主模块的话，END语句可以指出一个标号，它表示该程序的启动地址。

（2）PUBLIC

程序模块中的PUBLIC语句，用来说明该模块中定义的哪些符号常量、变量和标号（包括过程名），可以被其它模块所引用。它的格式是：

```
PUBLIC〈符号表〉
```

符号表中所给出的符号常量、变量、标号及过程都必须是在本模块中定义的符号。注意，寄存器名和值为实数、或者是值超过两个字节的整数的符号常量都不得作为“公共常量”使用。下面就是把一个过程说明为“公共过程”，以便其它模块调用的例子。如：

```
        PUBLIC  INFO
INFO    PROC    FAR
          ⋮
INFO    ENDP
```

注意，几个模块公用的过程一般都是属于FAR类型。

（3）EXTRN

伪操作命令EXTRN，用来指出本模块中需要引用但却是在其它模块（可能分散在几个模块）中定义并说明为PUBLIC的那些符号，这些符号可以是符号常量、变量、标号（包括过程名），它的格式为：

```
EXTRN   〈符号：类型〉〔，…〕
```

其中，类型可以是BYTE、WORD、DWORD、NEAR、FAR和ABS。ABS表示该符号是符号常量而不是变量或标号。当然，这里所有符号的类型与它们在其它模块中定义时的类型必须保持一致。例如，某模块需要调用上一个例子中的INFO过程时，必须在模块中使用下述语句：

```
EXTRN   INFO：FAR
```

5．程序分段与存贮分配

上册第一章已经介绍过，8086/8088CPU具有1兆字节的寻址能力，它的物理地址共有20位。由于CPU中寄存器均为16位字长，因此物理地址总是由段寄存器内容乘2^4再加上16

位的逻辑地址（称为段内位移量）而形成的。段寄存器的值固定之后，从段起址开始的连续64KB存贮空间中存放的代码或数据，仅用位移量就可以存取其中的数据或跳转到某条指令了。因此生成的指令代码短，且执行速度也较快。从段起址开始的连续64KB存贮空间称为一个物理段（参见上册第一章图1-8）。显然，如果要访问段外的数据或跳转到段外的指令，则必须指出新的段地址和段内位移量才行。这样，指令就比较复杂，而执行速度也较慢。

8086/8088有四个段寄存器：CS，DS，ES和SS。使用这四个寄存器可以在存贮器中同时定义四个物理段。如果程序代码、数据及堆栈分别存放或者分配在这些段中，那么程序的效率就比较高。宏汇编语言的段定义伪操作命令SEGMENT及ASSUME命令就是为此目的服务的。

（1）SEGMENT和ENDS命令

① 段的定义和汇编时的处理

汇编语言程序中，利用SEGMENT和ENDS伪操作命令可以把程序模块中的语句分成若干段（称为逻辑段，简称段），分段的目的是：

* 把指令代码、数据及堆栈区定义成不同的段，以便分别装入由CS、DS和SS所指出的不同物理段中。如果指令、数据和堆栈区处在同一段中，则CS、DS和SS寄存器的值必须一致，即它们被集中在仅64KB的同一物理段中，这样，程序功能受到很大限制。

* 指令代码或数据的数量超过64KB时，如果划分成若干段，可以使不同的逻辑段分配在存贮器不同的物理段中，从而充分使用8086/8088 1MB的存贮空间。

在汇编语言程序中，使用SEGMENT和ENDS定义一个段的格式是：

```
〈段名〉  SEGMENT  〈定位方式〉〈联合方式〉〈类别名〉
                    ⋮
                  汇编语句
                    ⋮
〈段名〉  ENDS
```

其中，段名是为该段起的名字，用来指出汇编程序为该段分配的存贮器起始位置，它有段址和段内位移量两个属性。SEGMENT语句中其它三个参数将在下面再作介绍。一个程序模块可以定义为若干段，段名可以各不相同，也可以重复。汇编程序对段进行处理的过程如图2-3所示。设汇编模块中共定义了五个段，段名分别为A，B，C，A，B。当使用汇编程序对此模块进行汇编时，每遇到一个新名字的段，就在段表中填入其段名，同时为该段配备一个初值为0的位置计数器。然后，在对该段内的语句进行汇编时，凡申请分配存贮器的语句（如DB，DW等）及产生目标代码的语句，均按照其需要的字节数目在位置计数器中累计计数，并在对应段的目标代码区中生成代码或保留数据区。段内定义的所有标号及变量的位移量属性就是当时计数器中的值。如果汇编时扫描到一个已定义过的段（例如A），则它不再作为新的段处理，而是使用A计数器继续累计汇编生成的字节数，并把对段中语句汇编生成的代码或申请保留的字节添加到目标代码段A中。整个模块汇编完毕后，在段表内填入它们的长度。但它们的段起址要到连接（LINK）操作时才能确定。

② 联合方式选择

一个稍大些的汇编语言程序可能由若干个模块组成，每个模块内又划分成若干段。如果这些段都不太大（一般都是如此），而又分别定义了不同的段名，则当这些模块连接起来并

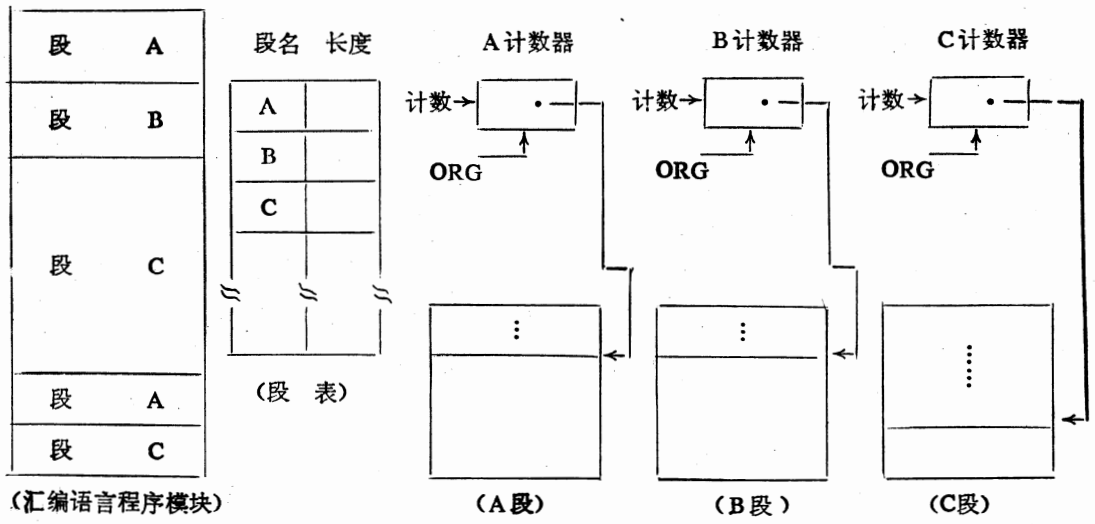


图2-3 汇编程序对段定义的处理

被装入机器运行时，无法都安排在四个不同的物理段内，因此程序运行时极为不便。如果不同模块中相同性质的段（如代码段或数据段）使用同样的段名，则当这些模块进行连接时就可以把同名的段按照指定的方式联合起来，从而达到更有效更方便地使用存贮器的目的。

联合方式共有六种选择：

* PUBLIC方式　表示该段与其它模块中说明为PUBLIC方式的同名段相互联合成一个逻辑段，公用一个段址，运行时装入同一个物理段中。

* STACK方式　与PUBLIC方式相同。连接时，将把所有STACK方式的同名段连接成一个段。在运行时就是SS寄存器所指出的物理段，且栈指针SP必须指向该段的起始地址。注意，每个程序中必须至少有一个STACK段。

* COMMON方式　表示该段与其它模块中所有也说明为COMMON的同名段共享相同的存贮区域，即这些段的起始地址都相同。共享的公共存贮区域的长度是各模块同名段中的最大长度。使用COMMON方式就可以使得不同模块中的不同变量或标号作用于相同的存贮器单元。

* MEMORY方式　表示该段在存贮器中应该定位在所有其它段的上面(即地址较大的区域)。如果若干模块中不止一个段选用MEMORY方式，则除了第1个遇到的MEMORY段之外，其它段均作为COMMON段处理。

* AT〈数值表达式〉　表示该段应按绝对地址定位，其段址即为数值表达式的值，位移量为0。

* 不指定方式　若定义一个段时不指定上述任何一种方式，则该段与其它模块中的段（同名段或不同名的段）没有任何关系，运行时它将是一个独立的段，使用分配给它的段址进行工作。

图2-4表示连接程序（LINK）是如何根据段定义语句中的联合方式把各个模块中的许多段联合在一起的。图中三个模块共有七个段，当它们被连接时，LINK程序检查各个段的联合方式（图中P，C，S分别代表PUBLIC，COMMON和STACK方式），根据要求把有

关的段联合在一起生成四个新的段。新段的长度是被联合在一起的各段长度之和（用PUBLIC或STACK方式时），或者是各段中最长一段的长度（COMMON方式）。用PUBLIC或STACK联合起来的段中的变量及标号的位移量，必须作适当的修改。修改的方法是把原来的位移量加上新段中在该段前面的所有段的长度。例如，原模块2中A段内的所有变量、标号的位移量都必须加上模块1中A段的长度。

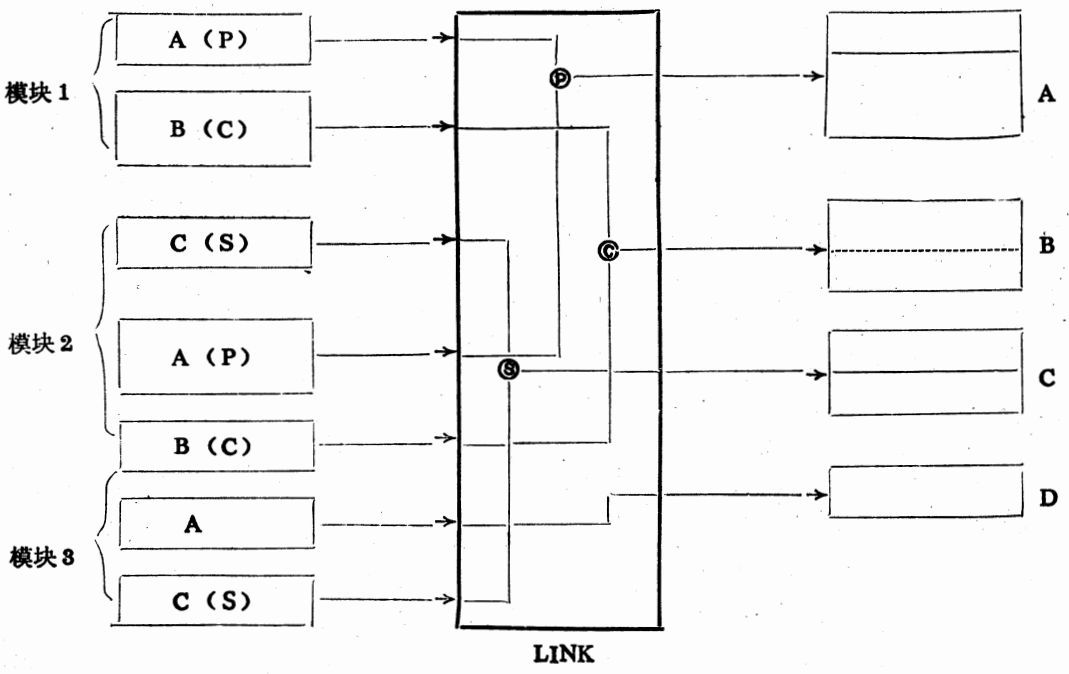


图2-4 连接程序对段进行联合处理的示意图

③ 定位方式选择

汇编后生成的目标代码模块(.OBJ)中段址和位移量都没有最后确定，LINK程序除了完成段与段的联合操作之外，另一个任务是把联合后得到的各个段相互衔接起来，为了有效地利用存贮器空间，LINK程序总是使得程序中的所有段在存贮器中相继存放。相邻两段之间应该如何衔接叫做定位方式。定位方式是由段定义语句指出的，它们共有四种方式：即PARA方式、BYTE方式、WORD方式和PAGE方式，缺省时为PARA方式。这四种定位方式的含义可以通过图2-5来说明。

设图2-4中经过联合后生成的A、B、C、D四个段的长度分别为1376H字节、A47H字节、1234字节和405H字节。并假设分配给A段的物理地址是00000（段址：0000，位移量：0000）。当选择不同的定位方式时，各段定位后的段地址如图2-5所示。从中可以看出，PARA定位方式(或缺省方式)在定位时每段的起始地址总是16的整倍数(即位移量为0)，这种定位方式虽然最简单，但段与段之间往往留有空隙，存贮器使用效率低。图中（b）表示选用BYTE定位方式。它使得段间不留间隙。因此每个段的起始地址中位移量不一定是0。（c）中PAGE方式要求段起始地址是256的整倍数，也就是说，段的边界必须是页的边界；WORD方式只要段起始地址是偶数地址，它特别适用于数据项的类型为字的数据段。

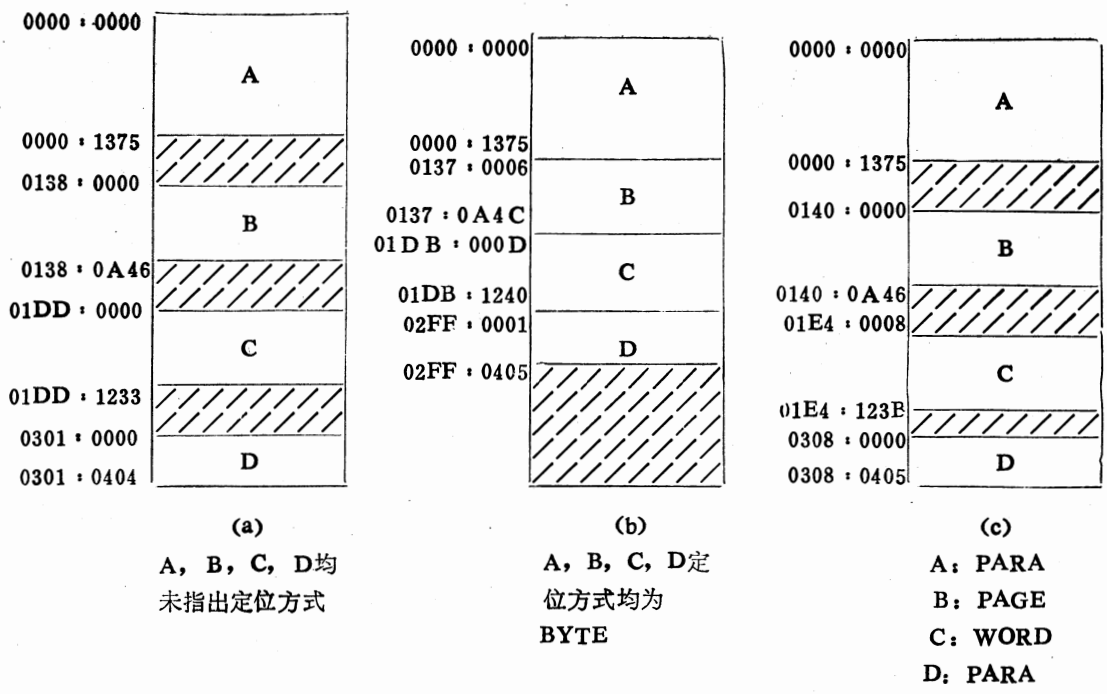


图2-5 定位方式的选择

在LINK程序根据定位方式进行各个段的定位之后，凡是起始地址中位移量不为0的段，都必须再次修改该段中所有变量和标号的位移量的值。修改的方法就是加上段起始地址中的位移量。当然，在LINK程序中，对标号以及变量的位移量可能发生的两次修改，实际上是合并在一起完成的。

在经过上述处理之后的目标代码，称为装入模块（.EXE文件），仍然保持其可重定位的性质，只是在装入内存运行的时候才最后确定它装入到内存中的物理地址。

④ 段的类别

在定义每一个段的时候，需要时还可以在SEGMENT语句中给出该段的类别名。类别名是一个用单引号括起来的字符串。在进行连接处理时。LINK程序把类别名相同的所有段（它们未必同名）放在连续的存贮区域内（但仍然是不同的段）；同类的各个段在连接时，先出现的在前，后出现的在后。下面是分属于三个类别的五个不同的段，经过LINK处理之后生成的装入模块中各段的相对位置：

```
A  SEGMENT   'FAT'      'FAT'
B  SEGMENT   'BAZ'          A
C  SEGMENT   'BAZ'          E
D  SEGMENT   'ZOU'       'BAZ'
E  SEGMENT   'FAT'          B
                            C
```

'ZOU'

D

（LINK前）　　（LINK后）

（2）GROUP

GROUP伪操作命令用来把模块中若干不同名的段集合成一个组，并赋予一个组名，使它们都装在同一个物理段中（64KB空间）。把若干段定义成一个组的优点是能够得到比较紧凑的代码，这样，组内各段之间的跳转都可以看作是段内跳转。GROUP命令的格式为：

〈组名〉　GROUP　〈段名〉〔，…〕

其中，段名也可以是表达式SEG〈变量〉或SEG〈标号〉。由于同一组内的各个段的联合方式可以任意，因此，这些段合并成一个组之后，其大小是否能在64KB的范围内，汇编程序处理时无法知道，只有在进行连接处理时才能确定。如果超过64KB，LINK程序将会给出出错信息。

组名和段名一样，它表示该组的段地址（位移量为0），因此程序中可以把它作为直接量或跨段前缀使用。例如，假设GROUP1是一个已定义的组名，FXP是变量，则下面的用法都是正确的：

```
MOV    AX, GROUP1
MOV    DS, AX
MOV    BX, OFFSET  GROUP1 : FXP
```

（3）ORG

汇编程序对源程序中的段进行汇编处理时，图2-3中的位置计数器的初值一般总是0，然后依次累计段内语句被汇编后生成的目标代码的字节数目，利用伪操作命令ORG可以改变位置计数器的内容，这在某些场合下是很有用处的。ORG命令的格式为：

ORG　〈数值表达式〉

它表示把位置计数器设置成数值表达式的值。这样，ORG后面的一条指令性语句或数据区定义命令就从指定位置处进行汇编。数值表达式的值应该是非负的整数，且总是取65536的模数，以保证指针定位在0到65535之间。程序中可包含的ORG语句的数目不受限制，也不要求它们一定以递增顺序来确定地址。当然，这种情况下必须十分小心，以免在相同的位置上重复地生成目标代码。下面是使用ORG语句的例子：

```
CSEG    SEGMENT
        ORG  2
         ⋮
        ORG  $+3
         ⋮
CSEG    ENDS
```

它表示该段的目标代码从位置2的地方开始产生，到中间的某处又跳过三个字节后再继续生成目标代码，其中$表示位置计数器的当前值，它可以在数值表达式中使用。

（4）ASSUME

ASSUME伪操作命令用来通知汇编程序，段寄存器CS、DS、ES和SS的内容将被设定

为那些段（组）的段址，这样在把指令性语句翻译成目标代码时汇编程序能够检查出语句中所引用的变量／标号是否可以通过某个段寄存器正确地进行访问。ASSUME命令的格式为：

ASSUME ＜段寄存器＞：＜段名＞〔，＜段寄存器＞：＜段名＞〕

其中，段名可以是程序中已定义过的任何段名或组名，也可以是表达 式SEG＜变量＞或SEG＜标号＞，或者是关键字NOTHING。例如，

ASSUME DS：SEGA，ES：SEGB，SS：NOTHING，CS：SEGC

其中，把段寄存器设置成NOTHING表示以前为该寄存器所做的假设已被 取 消，此后指令运行时不再用到该寄存器，除非再用ASSUME重新假设。

使用ASSUME命令，仅告诉汇编程序有关的段寄存器即将被设定成那个段 的 段 址，而段址的设定则必须通过给段寄存器赋予段起始地址的指令性语句来完成。它的正确使用方式如下：

```
SEGC    SEGMENT
        ASSUME  DS：SEGA，ES：SEGB，SS：NOTHING，CS：SEGC
        MOV     AX，SEGA
        MOV     DS，AX
        MOV     AX，SEGB
        MOV     ES，AX
        ⋮
SEGC    ENDS
```

注意，段寄存器CS的值是在装入模块（·EXE文件）被装入内存时 由MS-DOS设 定 的，程序中可以不必进行处理。但ASSUME语句中却一定要给出CS寄存器的正确 值（即ASSUME语句所在段的段名）。

汇编程序通过ASSUME语句了解到运行时各段寄存器的设定值之后，就可以对被汇编的指令语句中的变量和标号进行下列检查：

① 检查指令中所引用的 变 量和标号是否合理，即它们的段属性是否与某个段寄存器的假设值相符。

② 检查是否需要为所 引 用的变量和标号产生跨段前缀 字 节，即检查变量和标号的段属性是否与硬件为该指令所规定的段寄存器的假设值相符。

图2-6是汇编程序对指令中引用的变量/标号是否合理以及是否需要生成跨段前缀字节处理过程的示意图。

下面是使用ASSUME语句和跨段前缀运算符的几个例子。假设 变 量VA、VB和VC分 属SEGA，SEGB和SEGC等不同的段，则下列语句可以正 确地完成VC←VA＋VB的操作：

```
        ASSUME  DS：SEGA、ES：SEGB、SS：SEGC
        ⋮
        MOV     AX，VA
        ADD     AX，VB
        MOV     VC，AX
```

其中，“ADD AX，VB”和“MOV VC，AX”两条指令，经过汇编处理后都会自 动 加上正确的跨段前缀字节。但如果ASSUME语句被修改成：

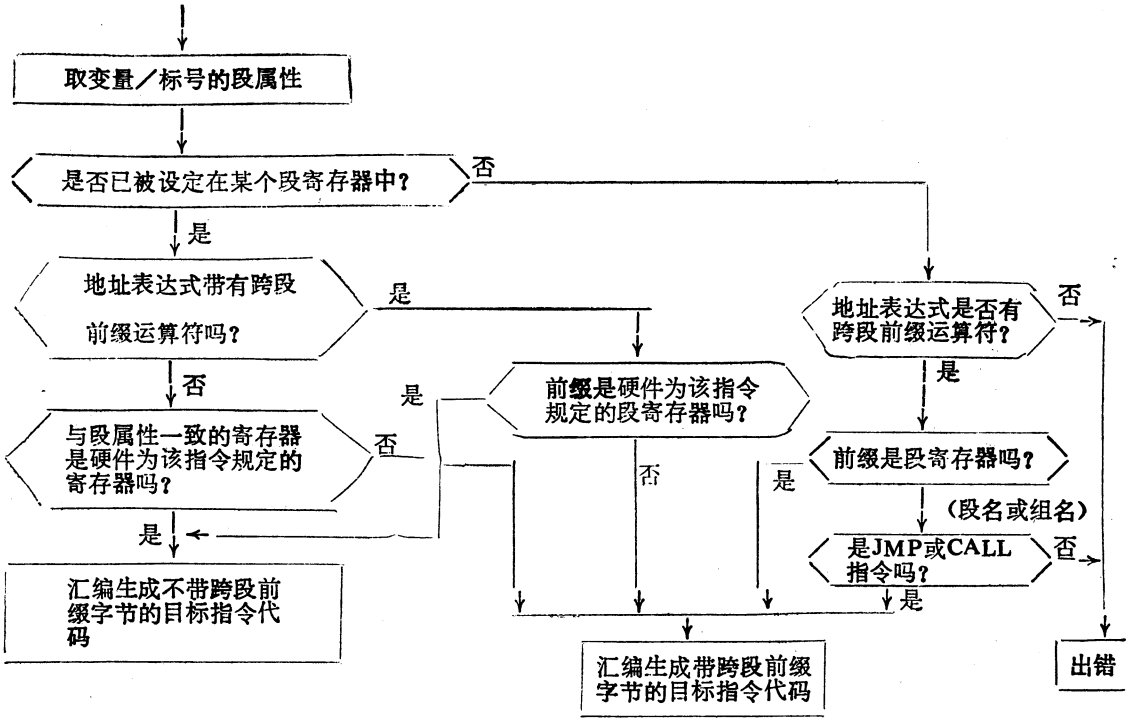


图2-6 汇编程序对变量／标号段属性的检查与处理

```
ASSUME    NOTHING
```

则指令中必须明确地为变量指出跨段前缀后，才能正确地完成所需要的操作：

```
MOV       AX, DS:VA
ADD       AX, ES:VB
MOV       SS:VC, AX
```

当然，寄存器DS、ES和SS预先必须置成SEGA，SEGB和SEGC的段属性值，否则程序运行结果就不正确。

从图2-6中可以看出，在编写汇编语言程序的时候，如果变量的段属性被ASSUME语句设定在与硬件规定不同的段寄存器中，该变量在指令中并不需要使用跨段前缀运算符，汇编程序能为它自动生成一个正确的跨段前缀字节。但是，为了改善程序的可读性，建议还是在变量前面加上跨段前缀为好。

6．条件伪操作

汇编语言提供了一组条件伪操作，它用来指示汇编程序应该测试的条件，使汇编程序能根据测试的结果有选择地对源程序中的语句进行汇编处理。所有条件伪操作都采用如下格式：

```
IF××××〔〈表达式〉〕
  ⋮
〔ELSE
  ⋮
ENDIF〕
```

其中，IF××××表示条件伪操作命令，它必须与ENDIF相配对。汇编程序根据条件伪操作命令的要求，对有关条件作检测。如果满足条件，则汇编整个条件块，若其中有ELSE命令的话，则只汇编从IF××××到ELSE之间的那一块；如果检测结果不满足条件，则忽略整个条件块或只汇编从ELSE到ENDIF的那一部分。

表2-8列出了各种条件伪操作命令、其表达式的形式及测试条件。

表2-8　条件伪操作命令

伪操作命令	表达式的形式	检测条件
IF	〈数值表达式〉	值是否不等于0
IFE	同上	值是否等于0
IF1		是否为第1趟扫描
IF2		是否为第2趟扫描
IFDEF*	〈符号〉	符号是否已定义或被说明为外部符号
IFNDEF*	同上	与IFDEF相反
IFB*	〈参数〉	是否没有参数或参数为空（仅尖括号，无内容）
IFNB*	同上	与IFB相反
IFIDN	〈参数1〉，〈参数2〉	是否两个字符串参数相同
IFDIF	同上	与IFIDN相反

（* 表示小汇编无此伪操作命令）

7. 宏处理伪操作

使用宏处理伪操作命令可以把重复出现的代码块定义成一条宏指令。此后在宏指令出现的地方，宏汇编程序总是自动地把它们替换成相应的代码块。因此，程序员不必在每次需要该代码块时都重写一遍。

在宏指令定义中，可以使用“形式参数”，它们在引用宏指令时被给出的一些名字或数值（称为实在参数）所取代。实际上，形式参数只为实在参数保留了一些空间，它们指出了在何处以及怎样使用实在参数。使用形式参数给宏指令带来了很大的灵活性。

小汇编（ASM）和宏汇编（MASM）的主要差别之一就是前者不能提供有宏处理的伪操作命令。下面就宏指令的定义和引用进行详细讨论。

（1）MACRO（宏定义）

宏指令是由MACRO伪操作命令定义的。其格式为：

```
〈宏指令名〉  MACRO  〔〈形式参数〉，…〕
                ⋮
             ENDM
```

其中，从MACRO语句行开始到ENDM语句之间的所有语句构成了宏指令的体，宏体中使用的形式参数必须在MACRO语句中列出，当有两个以上形式参数时必须用逗号隔开。

宏指令必须先定义后引用。如果以后想要改变的话，允许对它重新定义。宏指令可以嵌套定义，也就是说，在宏指令体内还可以再定义宏指令。汇编程序在遇到宏指令定义的时候并不对它进行汇编，只有在程序中引用它的时候汇编程序才把对应的宏指令体调出进行汇编处理，这个过程称为宏扩展。

引用宏指令的格式为：

〈宏指令名〉〔〈参数〉，…〕

其中参数项将一一对应地替换宏指令体中的形式参数。当有两个以上参数时，中间用逗号、空格或列表符隔开。如果用逗号隔开参数并且用尖括号把它们括起来的话，汇编程序将把尖括号内的所有项作为一个参数处理。例如，

```
GEN     1，2，3，4，5
```

传递五个参数，而以下这样的形式只作为一个参数传递：

```
GEN     〈1，2，3，4，5〉
```

引用宏指令时，实在参数的数目并不一定要和形式参数的数目一致。当实在参数多于形式参数时忽略多余的参数而当实在参数少于形式参数时，多余的形式参数假设为空白。宏指令的使用举例如下：

```
GEN     MACRO   XX，YY，ZZ
        MOV     AX，XX
        ADD     AX，YY
        MOV     ZZ，AX
        ENDM
```

若在汇编语言程序中出现引用上述宏指令的语句：

```
        GEN     DUCK，DON，FAT
```

则宏汇编程序生成以下语句：

```
        MOV     AX，DUCK
        ADD     AX，DON
        MOV     FAT，AX
```

（2）EXITM

在进行宏扩展时，若因为发现某种条件而想中途退出宏扩展的话，可以使用EXITM。通常EXITM和条件伪操作命令一起使用。如果含有EXITM的宏指令体嵌套在另一宏指令体当中时，仅退出内层的宏指令而继续扩展外层的宏指令。

（3）LOCAL

有时候需要在宏指令体内部定义和使用符号，这样如果多次扩展宏指令体的话，就会造成重复定义符号的错误。在这种场合下可以使用LOCAL伪操作命令，其格式为：

LOCAL　〈形式参数〉〔，〈形式参数〉，…〕

LOCAL伪操作仅能在宏指令定义块中使用，并且它必须是其中的第1个语句。在宏扩展时，汇编程序将给LOCAL中的形式参数指定特殊的符号，然后用这些符号替换宏指令体中

LOCAL指出的形式参数，从而避免了重复定义符号的错误。宏汇编生成的符号的范围是??0000开始到??FFFF。

例如：

```
FUN          SEGMENT
             ASSUME  CS: FUN, DS: FUN
SAM          MACRO   NUM, Y
             LOCAL   A, B
A:           MOV     AX, Y+1
B:           ADD     AX, NUM+1
             JMP     A
             ENDM
A:           MOV   AX, BX
             SAM   0C00H, 0BEH
+??0000:     MOV   AX, 0BEH+1
+??0001:     ADD   AX, 0C00H+1
+            JMP   ??0000
             JMP   A
             SAM   03C0H, 0FFH
+??0002:     MOV   AX, 0FFH+1
+??0003:     ADD   AX, 03C0H+1
+            JMP   ??0002
```

注意，前面加+号的表示是宏扩展出来的语句。

（4）PURGE（取消宏指令定义）

PURGE伪操作的格式为：

PURGE 〈宏指令名〉[，…]

PURGE将取消其后列举的所有宏指令定义。使用PURGE伪操作之后，不能再引用已被取消的宏指令，否则将会出错。

（5）REPT（重复）

利用重复伪操作命令可以按指定的次数重复宏指令体中的操作。重复伪操作REPT的格式为：

```
REPT  〈表达式〉
  ⋮
ENDM
```

它表示按表达式所指定的次数，重复REPT和ENDM之间的语句块，表达式的取值范围是0—65535。例如：

```
X       =       0
        REPT    10      ; generates  DB1—DB10
```

```
X       =       X+1
        DB      X
        ENDM
```

重复伪操作命令和宏指令处理的主要差别是：在程序中定义了宏指令之后，可以在程序中不同的地方多次引用，而重复伪操作命令只能在程序的某一个地方重复指定的语句块。

（6）IRP（不定次数重复）

IRP和REPT伪操作不同，它不是按照给定的重复次数，而是按照参数的个数进行重复。对应于给出的每个参数，分别执行指定的语句块一次，并用参数代替形式参数。IRP的格式是：

```
IRP〈形式参数〉，〈用尖括号括起来的参数〉
⋮
ENDM
```

例如：

```
IRP     X，〈1，2，3，4，5，6，7，8，9，10〉
DB      X
ENDM
```

本例和REPT中的例子效果相同。

（7）IRPC

IRPC可以按照其语句行中字符串的字符数目进行重复，并且每重复一次就依次用字符串中的字符代替形式参数。IRPC的格式为：

```
IRPC    〈形式参数〉，〈字符串〉
⋮
ENDM
```

例如：

```
IRPC    X，0123456789
DB      X+1
ENDM
```

本例和上述两个例子的效果相同。

（8）特殊宏处理操作符

除了上面介绍的宏处理伪操作命令之外，宏汇编程序还提供了一些特殊的宏处理操作符，它们增加了定义和引用宏指令的灵活性。其中较常用的是&记号，它可以在宏扩展时连接文本或符号。通常，宏扩展时并不识别符号和字符串当中的形式参数，但是在它前面加上一个&记号后，宏汇编程序就能够用实在参数代替这个形式参数了。例如：

```
ERRGEN          MACRO   X
ERROR&X:        PUSH    BX
                MOV     BX，'&X'
                JMP     ERROR
```

如果用“ERRGEN　　A”引用这个宏指令的话，将生成下列语句：

```
ERRORA:    PUSH    BX
           MOV     BX, 'A'
           JMP     ERROR
```

8. 列表伪操作及其它

（1）列表伪操作

宏汇编的列表伪操作有格式控制和列表控制两个功能。利用格式控制伪操作命令可以对源程序进行分页、加页标题等处理，列表控制伪操作命令可以控制输出汇编源程序的一部分或全部清单。

① 格式控制伪操作命令（PAGE，TITLE，SUBTTL）

PAGE命令可以指定开始输出新的一页或指定每页行数及每行字符数。如果只写PAGE或PAGE+表示换一页。如果写成PAGE〈页长度〉，〈行宽〉的话，则指定了一种新的输出方式。页长度可以是10—255，缺省值是50行/页。行宽的范围是60—135字符，缺省值是80字符/行。

TITLE伪操作给源程序指定一个标题，以后在每页的头上都将显示这个标题。例如，

```
TITLE    PROG1-1st program
```

另外，还可以用SUBTTL给每页程序加上子标题，子标题在标题的下一行显示，标题和子标题都不得超过60个字符。一旦用SUBTTL指定子标题后，汇编程序将给每一页都加上这个子标题直至指定新的子标题为止。如果某一部分不要子标题时，可只写SUBTTL，其后不加文本，这样就停止输出子标题了。

② 列表控制伪操作（.LIST，.XLIST，%OUT）

伪操作命令.LIST和.XLIST可以打开或关闭列清单输出。当用户在汇编程序提示要求输入列表文件名时（参见下一节），指定列表文件名后将生成一个包含所有语句的列表文件。但是，如果在源文件中有.XLIST时，将关闭其后的源语句和所产生目标代码的列表输出，直到遇到.LIST伪操作命令时才重新向列表文件输出。

%OUT命令可以在汇编时显示接在其后的文本，这对显示一个长源程序的汇编过程中进行的状况以及显示条件汇编开关的值十分有用。例如：

```
%OUT  *Assembly half done*
```

以及

```
IF1
%OUT   *Pass 1 started*
ENDIF
```

（2）其它伪操作

① COMMENT

可以在语句行中使用分号，表示其后的文本是注释。但是，当要在程序中插入很长一段注释时，就要在每行之前都加一个分号，这就显得比较麻烦。在这种场合可以用COMMENT伪操作命令方便地插入注释。COMMENT命令的格式为：

COMMENT〈分隔符〉〈文本〉〈分隔符〉

COMMENT后面遇到的第1个非空格字符被认为是分隔符。

② INCLUDE

利用INCLUDE伪操作命令，可以把另一个源文件插入当前的源文件一起汇编，从而可以避免重复输入几个源文件中相同的语句序列。其格式为：

INCLUDE　〈文件名〉

宏汇编程序汇编到INCLUDE伪操作语句之后立即打开INCLUDE文件，并汇编到当前的源文件中。遇到文件结束时，汇编程序接着汇编INCLUDE之后的语句。INCLLDE可以嵌套，也就是说，用INCLUDE语句插入的文件中还可以包含INCLUDE语句。

例如：

```
INCLUDE  ENTRY
INCLUDE  B:RECORD.TST
```

③ .RADIX

宏汇编程序假设所有常量的缺省基数都是十进制数。但是，用户可以通过.RADIX伪操作把缺省基数改为1—16范围内的任意基数。其格式是：

.RADIX　〈表达式〉

其中，表达式与当前缺省基数无关，一定是十进制数。例如：

```
MOV      BX, 0FFH
.RADIX   16
MOV      BX, 0FF
```

本例中两条MOV指令含义相同。

.RADIX伪操作不影响DD、DQ和DT伪操作命令。在这些伪操作中，输入的数值只要没有加上数值类型都认为是十进制数。

④ EVEN

使用EVEN伪操作，可以把段内位置计数器的值置为偶数地址边界。如果遇到EVEN时计数器的值不是偶数，则汇编程序将插入NOP指令，使其为偶数。EVEN的格式为：

```
EVEN
```

例如，在使用前计数器的值为0019H，使用“EVEN”后计数器的值为001AH，汇编程序在0019H处插入了一条NOP命令。

第四节　宏汇编及有关实用程序的操作与使用

在使用汇编语言开发软件的过程中，除了必须使用汇编程序（MASM或ASM）对源程序进行汇编之外，还需要使用其它有关的实用程序，如连接程序LINK、库管理程序LIB、交叉参考程序CREF和动态调试程序DEBUG。本节中将简要地介绍上述程序的操作使用方法。

1. 汇编程序（MASM或ASM）

汇编程序用于对汇编语言源程序进行汇编，产生可重定位的的目标代码文件(.OBJ)。

然后再用连接程序LINK把可重定位目标代码文件连接起来，这样就可生成可执行的目标代码文件（.EXE）。

前面已经介绍过，一个汇编语言程序可能是由若干模块组成的，每个模块可以单独进行汇编处理，然后再用LINK程序把它们连接起来，整个过程如图2-7所示。

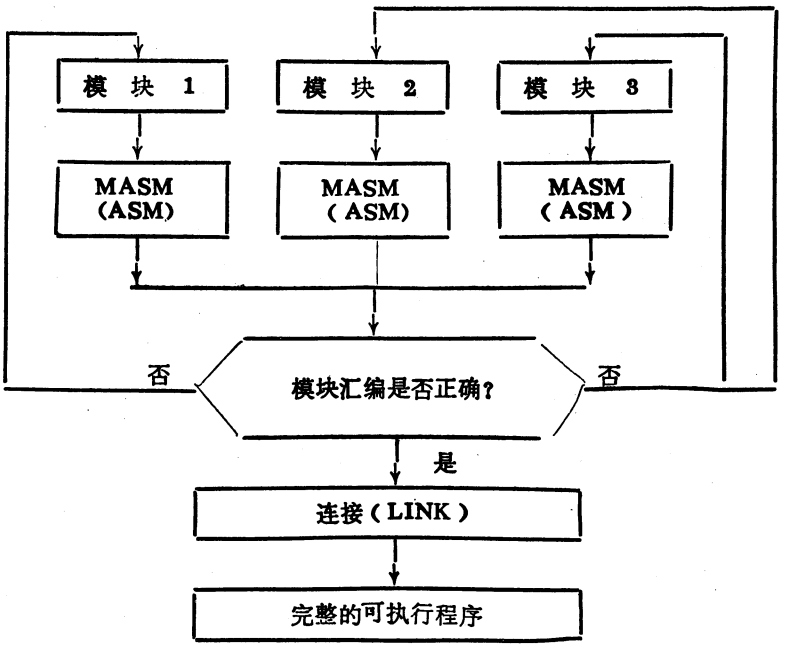


图2-7 各模块分别汇编后连接成一个程序

汇编程序有两种启动方式：

（1）提问方式

在MS-DOS下输入命令行：

MASM 或 ASM

这时MS-DOS将装入并启动执行汇编程序，然后汇编程序就逐次向用户提问，用户必须根据要求予以回答。在回答信息的最后，可输入一个或多个开关，汇编程序将按照回答信息及开关的定义作相应处理。

汇编程序的提示信息和所要求的回答如表2-9所示。允许使用的开关及其功能如表2-10所示。其中，开关/R和/E只允许在MASM1.20版以上使用。

表2-9　宏汇编的提示信息及回答

提　示　信　息	回　　答
Source filename[.ASM]：	欲汇编的.ASM源文件名
Object filename[Source.OBJ]：	可重定位目标代码文件名 (缺省：源文件名.OBJ)
Source listing[NUL.LST]：	列表文件名 (缺省：无列表文件)
Cross reference[NUL.CRF]：	交叉参考文件用的文件名（该文件将由交叉参考程序CREF处理) (缺省：无交叉参考文件)

表2-10　宏汇编的开关及其功能

开　关	功　　能
/D	两趟扫描中都给出列表文件
/O	在列表文件中用八进制表示生成的目标代码和位移量
/X	列表伪操作在条件为假时不作列表
/R	对源程序中的8087指令进行汇编，并产生8087目标代码
/E	对源程序中的8087指令进行汇编，产生8087仿真目标代码

在回答提示信息时，可以使用以下两个命令字符：

；　　用于省略对后续提示的回答，仅取缺省值。

CTRL-C　　当因为输入了错误的文件名或其它原因而想中途退出汇编程序时，按下CTRL-C键即可退出。

（2）命令方式

以命令方式启动汇编程序时，必须在MS-DOS下打入下列格式的命令：

MASM（或ASM）　〈源文件〉,〔〈目标文件〉〕,〔〈列表文件〉〕,〔〈交叉参考文件〉〕〔/开关〕

于是在装入汇编程序后将立即开始进行汇编。命令中MASM（或ASM）后面的项目，分别顺次对应于提问方式中对各提示信息的回答，各项目之间用逗号隔开。/开关可放在任一项目之后。如果对某一提示信息采用缺省值，则只需在对应的项目处连续输入两个逗号。例如：

MASM　FUN, , FUN/D/X, FUN

2. 连接程序LINK

连接程序LINK用来连接汇编程序或高级语言生成的可重定位目标代码模块（.OBJ）以及指定的库文件，产生一个可执行的装入模块（.EXE）。前一节中已对连接处理过程作了

大致的阐述，这里主要介绍LINK的使用方法。

连接程序有三种启动方式：

（1）提问方式

在MS-DOS下输入命令行：

LINK

这时MS-DOS将把连接程序装入内存，并向用户提问，用户根据要求输入相应的回答。LINK的提示信息和所要求的回答如表2-11所示。允许使用的开关及功能如表2-12所示。

表2-11　连接程序的提示信息及回答

提示信息	回答
Object Modules[.OBJ]:	目标代码模块表（各模块之间用+号隔开）
Run File [object.EXE]:	连接后生成的装入模块名
List File [NUL.MAP]:	列表文件名（缺省：无列表文件）
Libraries [.LIB]	库文件名表（各文件名之间用+号隔开）

表2-12　LINK的开关及其功能

开关	功能
/D	把数据装到数据区DS的上部
/H	把装入模块（.EXE）装到存贮器的高区
/L	在列表文件中加入行号和地址
/M	以字典顺序在列表文件中列出全部公共变量
/P	暂停连接（使用此开关可以在LINK输出.EXE文件前换磁盘）
/S：n	指定装入模块的栈的大小（n的值最大为65535）
/N	禁止连接缺省库

除了在汇编程序一段中叙述过的两个命令字符以外，连接程序的回答信息中还可使用命令字符+号，它用以分隔多个要连接的目标代码模块或库文件。当显示器上一行输入不下时还可使用+号另起一行。

（2）命令方式

以命令方式启动连接程序时，必须在MS-DOS下输入如下格式的命令：

LINK　〈目标代码模块表〉,[〈装入模块名〉],[〈列表文件名〉],
[〈库文件名表〉] [/开关]

命令中LINK后面的各项目顺次对应于提问方式中对各提示信息的回答，各项目之间用逗号隔开。如对某一项提示信息采用缺省值，则只需在对应的项目处连续输入两个逗号。例如：

LINK FUN+TEXT+TABLE+CARE/P/M, , FUNLIST, COBLIB.LIB

（3）文件方式

以文件方式启动LINK的命令格式为：

LINK @〈文件名〉

其中，文件名用来指出一个包含有对于LINK提示信息回答的正文文件，文件中每一行正文对应一个回答，因而用户可以不必一一回答LINK的提示，而由LINK程序直接从文件中取得回答。当然，在使用文件方式之前，用户必须建立一个这样的响应文件。

3. 库管理程序LIB

所谓“程序库”（扩展名为.LIB），是若干常用的可重定位目标代码模块的集合，它供连接程序LINK使用。实用程序LIB是一个程序库的管理程序，它可以建立和修改程序库。使用LIB可以建立一个适用于各种各样应用的通用程序库，也可以针对特定的应用建立一个专用的程序库。

库管理程序LIB的启动方式有以下三种：

（1）提问方式

在MS-DOS下输入命令行：

LIB

这时MS-DOS将把LIB装入存贮器并向用户提问，用户根据要求输入相应的回答。LIB的提示信息及其回答如表2-13所示，回答中可用的命令字符如表2-14所示。

表2-13 LIB的提示信息及其回答

提示信息	回答
Library name:	欲进行操作的库名（缺省文件扩展名为.LIB）
Operation:	命令字符及模块名或目标文件名
List file:	交叉参考列表文件名 （缺省：NUL，无交叉参考列表文件）

表2-14 库管理命令字符

命令字符	功能
+	把目标代码文件作为最后一个模块加入库中
-	从库中删除一个模块
*	从库中取出模块，写入目标文件中（库中仍保留）
;	剩下的提示取缺省值
&	提示信息在一行内回答不下时，键入此字符将另起一行
CTRL-C	终止库管理操作

（2）命令方式

命令的格式为：

LIB　〈库文件名〉〈操作〉,〔〈列表文件〉〕

接在LIB后面输入的项目是对提问方式下各提示信息的回答。其中库文件就是要修改或建立的库名，LIB程序假设其扩展名是.LIB，但也可以指定别的扩展名。如果指定的文件名不存在时，LIB将显示下列信息，并要求回答：

Library file does not exist. Create?

如果回答Y，则建立一个新的库，否则停止处理。

“操作”用来指定是删除模块、添加模块还是复制库中模块，这三种操作分别对应于+，－、＊三个命令字符。列表文件将指示LIB建立一个包含库中模块的PUBLIC符号的交叉参考列表文件。例如：

LIB　ASMLIB+SAMPLE-SAMPLE, ASMCROSS.PUB

（3）文件方式

以文件方式启动LIB的命令格式为：

LIB　@〈文件名〉

其中，文件名用来指出一个包含有对于LIB提示信息回答的正文文件，LIB将直接从其中取得回答。下面是一个使用文件fun启动LIB的示例：

```
A>copy  con: fun
asmlib
+sample-sample
asmcross.pub
^Z
        1 File(s)     copied
A>lib   @fun
      Microsoft   Library   Manager   V2.00
(C) Copyright   1983   by   Microsoft   Inc.
Library   name:   asmlib
Operations: +sample-sample
List  file: asmcross.pub
```

4. 交叉参考程序CREF

交叉参考程序CREF可以根据汇编程序生成的交叉参考文件（.CRF），按照其中所有符号的字典顺序建立一个交叉列表文件。用户按照这个交叉列表文件的行号，可以很快找到自己程序中所有符号的位置。这对于调试用汇编语言书写的程序十分有用。注意，在使用CREF生成交叉列表文件之前，必须使用汇编程序建立一个交叉参考文件.CRF。

启动CREF的方式也有提问方式和命令方式两种。使用提问方式，只要打入命令

CREF

然后再回答输入的交叉参考文件名和输出的交叉列表文件名即可。使用命令方式则应输入：

CREF 〈交叉参考文件名〉，〈交叉列表文件名〉

在交叉参考列表文件中，用户程序的所有符号都按字典顺序列出。在各符号右边的是该符号在源程序中出现的行号，其中加＃号的是定义该符号的行号。下面是第一节中所举例子Sample.asm的交叉参考列表文件的内容：

```
Symbol    Cross    Reference    ( #  is  definition )      Cref-1

CODE . . . . . . . . . . . . .  15#      15      17      29
DATA . . . . . . . . . . . . .   8#       8      14      21      23
MESSAGE. . . . . . . . . . . .   9#      25
STACK  . . . . . . . . . . . .   2#       2       7
START  . . . . . . . . . . . .  16#      28      30
```

第五节　宏汇编语言程序设计举例

前面几节已经介绍了汇编语言的基本语法、8086/8088机器指令的汇编表示和汇编语言的伪操作命令等编制汇编语言程序所必须掌握的知识。本节将通过一些具体的例子，介绍如何在IBM PC系统环境下编制和运行汇编语言程序，使读者对汇编语言程序的设计以及对IBM PC系统的软硬件接口有更深入的了解。在介绍具体例子之前，先简单讨论一下宏汇编语言程序的运行环境。通常，汇编语言程序的运行环境包括两个方面：

* 硬件环境　中央处理器和协处理器，存贮器，以及各种外部设备（如显示器、打印机、磁盘）。
* 软件环境　基本输入输出系统（BIOS），磁盘操作系统（DOS）等。

与高级语言程序相比，汇编语言程序与这些软、硬件环境联系更加紧密，它允许直接深入到系统的深处，有效地发挥系统的功能。因此，这就要求程序员对软、硬件环境有更多、更深入的了解。

上册书中第一章已对IBM PC的硬件结构以及各种外部设备的程序设计原理作了详细介绍，这里不再重复。下面将扼要介绍汇编语言程序与BIOS和MS-DOS的接口。

1．汇编语言程序与BIOS的接口

IBM PC写在只读存贮器中的程序，一部分是BASIC语言的解释程序，另一部分就是基本输入输出系统（BIOS）。BIOS的主要功能是驱动系统中所配置的常用外部设备，如显示器，键盘、打印机，磁盘驱动器以及异步通讯接口等，使程序员不必过多地关心这些设备具体的物理特性和逻辑结构细节（如端口地址，命令及状态格式等），从而能方便地控制各种输入输出操作。

汇编语言程序以软中断（10H—1AH）的方式调用BIOS中的各个子程序。如果子程序具有多种功能的话，则用AH寄存器来指出所需要的某种功能。表2-15到表2-20是常用BIOS子程序的功能以及入口参数和出口参数一览表，供汇编语言程序设计时参考。

2．汇编语言程序与MS-DOS的接口

汇编语言程序除了可以调用BIOS中的子程序之外，还可以通过软中断20H—27H调用

表2-15　显示器驱动程序(INT 10H)

功　能	入　口　参　数	出　口　参　数
(AH) = 0 置显示模式	(AL) = 0　40×25 黑白 (AL) = 1　40×25 彩色 (AL) = 2　80×25 黑白 (AL) = 3　80×25 彩色 图　形　模　式 (AL) = 4　320×200 彩色 (AL) = 5　320×200 黑白 (AL) = 6　640×200 黑白	无
(AH) = 1 设置光标类型	$(CH)_{4-0}$ = 光标起始线 $(CL)_{4-0}$ = 光标终止线	无
(AH) = 2 设置光标位置	(DH,DL) = 行,列 如（0，0）为左上角 (BH) = 页号（图形模式为0）	无
(AH) = 3 读光标位置	(BH) = 页号（图形模式为0）	(DH,DL) = 行,列 (CH,CL) = 当前光标模式
(AH) = 4 读光笔位置		(AH) = 0 未按光笔开关 (AH) = 1 寄存器中光笔值有效 (DH,DL) = 光笔所在行,列 (CH) = 扫描线（0—199） (BX) = 象素列号（0—319,639）
(AH) = 5 选择当前显示页 (字符方式有效)	(AL) = 新页号 { 0—7用于模式0或1；0—3用于模式2或3 }	无
(AH) = 6 当前页上滚	(AL) = 行数 (从窗口底部算起，空白的行数) (AL) = 0 为整个窗口空白 (CH,CL) = 滚动区域的左上角的行、列 (DH,DL) = 滚动区域右下角的行、列 (BH) = 空白行的属性	无

（续表2-15）

功　　能	入　口　参　数	出　口　参　数
(AH) = 7 当前页下滚	(AL) =行数（从窗口顶部算起空白的行数） (AL) = 0为整个窗口空白 (CH,CL) =滚动区域左上角的行、列 (DH,DL) =滚动区域右下角的行、列 (BH) =空白行的属性	无
(AH) = 8 读当前光标位置处的属性/字符	(BH) =显示页号(字符模式有效)	(AL) =读出字符 (AH) =读出字符属性（字符模式有效）
(AH) = 9 写属性/字符到当前光标位置处	(BH) =显示页号(字符模式有效) (CX) =字符计数 (AL) =欲写字符 (BL) =字符属性（字符模式）/字符颜色（图形模式）	无
(AH) =10 仅写字符到当前光标位置	(BH) =显示页号(字符模式有效) (CX) =字符计数 (AL) =字符	无
(AH) =11 置彩色调色板	(BH) =当前使用的调色板彩色号（0—127） (BL) =彩色值	无
(AH) =12 写点	(DX) =行号 (CX) 列号 (AL) =彩色值	无
(AH) =13 读点	(DX) =行号 (CX) =列号	(AL) =所读的点
(AH) =14 写字符到光标位置，光标进一	(AL) =欲写字符 (BL) =前台彩色(图形模式)	无
(AH) =15 读当前显示状态		(AL) =当前显示模式 (AH) =屏幕上字符列数 (BH) =当前显示页

表2-16　磁盘驱动程序(INT 13H)

功　能	入口参数	出口参数
(AH) = 0 磁盘复位		(AH) =磁盘状态
(AH) = 1 读磁盘状态		(AH) =磁盘状态
(AH) = 2 读指定扇区	(DL) =驱动器号 (0—3) (DH) =面号 (0—1) (CH) =道号 (0—39) (CL) =扇区号 (1—9) (AL) =扇区数 (1—8) (ES:BX) =欲读/写数据的地址	(AH) =磁盘状态 CF= 0成功 / 1出错 (AL) =读出的扇区数
(AH) = 3 写指定扇区	同上	(AH) =磁盘状态 CF= 0成功 / 1出错
(AH) = 4 检验指定扇区	同上	同(AH) = 2
(AH) = 5 对指定磁道格式化	同上	同(AH) = 3

表2-17　键盘驱动程序(INT 16H)

功　能	入口参数	出口参数
(AH) = 0 读键盘		(AH) =键入字符的扫描码 (AL) =键入字符的ASCII码
(AH) = 1 判有无键入		ZF= 0 键盘有输入 / 1 键盘无输入
(AH) = 2 读特殊键状态		(AL) =特殊键状态: $(AL)_7$ = insert键 $(AL)_6$ = Caps lock键 $(AL)_5$ = num lock 键 $(AL)_4$ = scroll lock键 $(AL)_3$ = alt 键 $(AL)_2$ = ctrl键 $(AL)_1$ =左 shift 键 $(AL)_0$ =右 shift 键

表2-18　打印机驱动程序(INT 17H)

功　能	入　口　参　数	出　口　参　数
(AH) = 0 打印字符	(AL) = 欲打印字符 (DX) = 欲使用打印机号(0—2)	(AH) = 1 越时
(AH) = 1 初始化打印机	(DX) = 欲使用打印机号(0—2)	(同下)
(AH) = 2 读打印机状态	同上	(AH) = 打印机状态： $(AH)_7$ = 空闲 $(AH)_6$ = 响应 $(AH)_5$ = 无纸 $(AH)_4$ = 已选中 $(AH)_3$ = 出错 $(AH)_0$ = 越时 $(AH)_{1-2}$未用

表2-19　通讯驱动程序(INT 14H)

功　能	入　口　参　数	出　口　参　数
(AH) = 0 初始化串行口	(AL) = 初始化参数* (DX) = 串行口号码(0－2)	(AX) = 串行口状态 (AH中为通讯线状态，AL中为MODEM状态，详见上册第一章)
(AH) = 1 发送数据字符	(AL) = 欲发送字符 (DX) = 串行口号码(0－2)	(AH) = 通讯线状态 $(AH)_7$ = 1 表示传送失败
(AH) = 2 接收字符	(DX) = 串行口号码(0－2)	(AH) = 通讯线状态 $(AH)_7$ = 1 表示传送失败 (AL) = 字符
(AH) = 3 读串行口状态	(DX) = 串行口号码(0－2)	(AX) = 串行口状态

*AL 寄存器作为初始化参数时各位定义为：

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

D_1 D_0 字符长：
1 0 = 7 位
1 1 = 8 位

波特率：
110 = 0 0 0
150 = 0 0 1
300 = 0 1 0
600 = 0 1 1
1200 = 1 0 0
2400 = 1 0 1
4800 = 1 1 0
9600 = 1 1 1

D_2 停止位数：0 = 1 位；1 = 1.5(如字符长为5位) = 2(如字符长为6,7或8位)

D_3 校验：0 = 不校验；1 = 校验

D_4 0 = 不校验

D_3 D_4 1 0 奇校验
D_3 D_4 1 1 偶校验

表2-20　BIOS中其它子程序

软中断号	功　能	入口参数	出口参数
1AH	(AH) = 0 读当前时钟		(CX) =计数的高位部分 (DX) =计数的低位部分 如果上次读时钟到现在未超过24小时 (AL) = 0，否则 (AL) ≠0
	(AH) = 1 置时钟	(CX) =计数的高位部分 (DX) =计数低位部分 (计数速率约为每秒18.2次)	
11H	检测系统配置情况		(AX) =系统配置情况： $(AX)_{15,14}$=连接的打印机数目 $(AX)_{12}$=连接的游戏I/O $(AX)_{11,10,9}$=连接的RS232的数目 $(AX)_{7,6}$=驱动器数目 $(AX)_{5,4}$=初始显示模式 00—不用 01—40×25 黑白（彩色板） 10—80×25 黑白（彩色板） 11—80×25 黑白（黑白板） $(AX)_{3,2}$=底板RAM容量(00=16K，01=32K，10=48K，11=64K) $(AX)_{0}$=系统有无软盘驱动器 $(AX)_{1,8,13}$不用
12H	检测存贮器容量		(AX) =全部存贮容量（以1K字节为单位）
19H	引导程序		
18H	启动ROM BASIC		

MS-DOS所提供的功能。尤其是使用21H软中断，可以直接调用MS-DOS所提供的有关输入输出操作、目录操作、文件操作等七十多种功能，编制程序极为方便有效。MS-DOS的软中断及系统功能调用请参考上册第二章有关内容。

宏汇编语言程序在汇编和连接之后生成的可执行的代码文件（称为装入模块、文件扩展名为.EXE）能被MS-DOS当作外部命令文件一样装入存贮器并启动执行。DOS在装入程序时，将生成一个称为“程序前缀区”(PSP)的控制块。控制块中包含了用户启动该程序时打入的键盘命令中的有关信息，如输入参数等。PSP的格式请参见上册第二章。紧接在这个控制块之后，DOS装入.EXE文件中的代码段内容（指令部分），然后是数据段内容，最后是堆

栈段内容。装入完毕后，存贮器中的信息布局如图2-8所示。

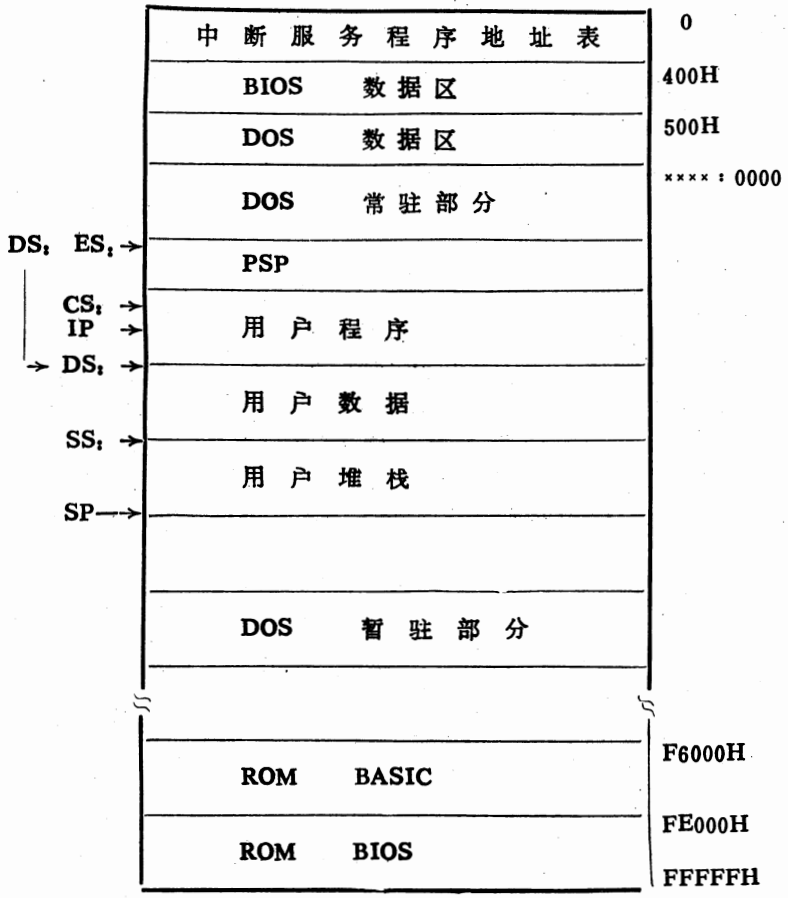


图2-8 用户程序装入后的存贮器布局

图2-8还给出了DOS在程序装入后对各段寄存器所置的初值。其中，DS和ES段寄存器指向程序前缀区的始地址；CS指向代码段内存的起始地址；SS指向堆栈段内存的始地址。堆栈指针SP由堆栈段的大小而定，即SP指向堆栈的底部，指令地址寄存器IP的内容被置为代码段中一个特定的位移量地址，以保证程序在指定起始点开始运行。

值得注意的是，.EXE程序被装入后，DS和ES并未指向用户自己所定义的数据段和附加数据段的起始地址。因此，汇编语言程序在使用数据之前，必须在程序中自行为DS或ES赋值，使它们指向所需要的段。

3. 汇编语言程序设计举例

例2-1 信息检索

功能：按照输入程序名后接着输入的参数值（0—9）检索并显示出对应的字符串信息。当参数值非法时显示出错信息。

程序：

```
00020 ; SAMPLE PROGRAM   - MESSAGE RETRIEVAL
00030 ;
00040 STACK   SEGMENT PARA STACK 'STACK'
00050         DB      256 DUP (0)      ;256 BYTES OF STACK SPACE
00060 STACK   ENDS
00070 ;
00080 DATA    SEGMENT PARA PUBLIC 'DATA'
00090 THIRTY  DB      30               ;VALUE FOR MUL INSTRUCTION
00100 PARM    DB      128 DUP (0)      ;MOVE PARAMETER TO HERE
00110 ; MESSAGE TABLE:
00120 MSG0    DB      'I LIKE MY IBM PC-------------'
00130 MSG1    DB      '8088 PROGRAMMING IS FUN-------'
00140 MSG2    DB      'TIME TO BUY MORE DISKETTES----'
00150 MSG3    DB      'THIS PROGRAM WORKS------------'
00160 MSG4    DB      'TURN OFF THAT PRINTER!--------'
00170 MSG5    DB      'I HAVE MORE MEMORY THAN YOU---'
00180 MSG6    DB      'THE PSP CAN BE USEFUL---------'
00190 MSG7    DB      'BASIC WAS EASIER THAN THIS----'
00200 MSG8    DB      'DOS IS INDISPENSABLE----------'
00210 MSG9    DB      'LAST MESSAGE OF THE DAY-------'
00220 ; ERROR MESSAGE:
00230 ERRMSG  DB      'ERROR !!! INVALID PARAMETER !!'
00240 DATA    ENDS
00250 ;
00260 CODE    SEGMENT PARA PUBLIC 'CODE'
00270 START   PROC    FAR
00280 ;
00290 ; STANDARD PROGRAM PROLOGUE EXCEPT RETAIN DS AS PTR TO PSP
00300 ;
00310         ASSUME  CS:CODE
00320         PUSH    DS       ;SAVE PSP SEG ADDR
00330         MOV     AX,0
00340         PUSH    AX       ;SAVE RET ADDR OFFSET (PSP+0)
00350         MOV     AX,DATA
00360         MOV     ES,AX    ;ESTABLISH EXTRA SEG ADDRESSABILITY
00370         ASSUME  ES:DATA
00380 ;
00390 ; MOVE PARAMETER AREA FROM PSP TO OUR DATA SEGMENT
00400 ;
00410         MOV     SI,80H   ;SOURCE STRING OFFSET (WITHIN PSP)
00420         MOV     DI,OFFSET PARM ;DEST. STRING OFFSET
00430         MOV     CX,128   ;STRING LENGTH TO MOVE
00435         CLD              ;SET 'FORWARD' STRING OPERATIONS
00440         REP     MOVSB    ;MOVE PARM INTO OUR DATA AREA
00450 ;
00460 ; ESTABLISH NORMAL DATA SEGMENT ADDRESSABILITY
00470 ;
00480         MOV     DS,AX
00490         ASSUME  DS:DATA
00500 ;
00510 ; CHECK FOR VALID PARAMETER AND CONVERT IT TO NUMERIC VALUE
00520 ;
00530         CMP     PARM,2   ;IS PARAMETER LENGTH = 2 ?
00540         JNZ     ERROR    ;BRANCH IF NOT
00550         MOV     AL,PARM+2 ;GET PARAMETER ITSELF
00560         SUB     AL,'0'   ;CONVERT FROM ASCII TO BINARY
```

```
00570           JC      ERROR   ;BRANCH IF NOT NUMERIC ( < '0' )
00580           CMP     AL,9    ;CHECK FOR VALID NUMERIC
00590           JA      ERROR   ;BRANCH IF NOT VALID ( > '9'
00600 ;
00610 ; SELECT APPROPRIATE MESSAGE FROM MESSAGE TABLE
00620 ;
00630           MOV     BX,OFFSET MSG0  ;POINT TO FIRST MESSAGE
00640           MUL     THIRTY  ; AX = AL * 30
00650           ADD     BX,AX   ;POINT TO DESIRED MESSAGE
00660           CALL    DISPLAY ;DISPLAY THE MESSAGE AT [BX]
00670           RET             ;RETURN TO DOS
00680 ;
00690 ; DISPLAY ERROR MESSAGE FOR INVALID PARAMETER:
00700 ;
00710 ERROR:    MOV     BX,OFFSET ERRMSG
00720           CALL    DISPLAY ;DISPLAY THE MESSAGE AT [BX]
00730           RET             ;RETURN TO DOS
00740 ;
00750 ; SUBROUTINE TO DISPLAY A MESSAGE ON THE SCREEN.
00760 ; ENTER WITH BX -> MESSAGE TO BE DISPLAYED.
00770 ; MESSAGE IS ASSUMED TO BE 30 CHARACTERS LONG.
00780 ;
00790 DISPLAY PROC      NEAR
00800           MOV     CX,30   ;NUMBER OF CHARACTERS TO DISPLAY
00810 DISP1:    MOV     AL,[BX] ;GET NEXT CHARACTER TO DISPLAY
00820           CALL    DISPCHAR ;DISPLAY IT
00830           INC     BX      ;POINT TO NEXT CHARACTER
00840           LOOP    DISP1   ;DO IT 30 TIMES
00850           MOV     AL,0DH  ;CARRIAGE RETURN
00860           CALL    DISPCHAR
00870           MOV     AL,0AH  ;LINE FEED
00880           CALL    DISPCHAR
00890           RET             ;RETURN TO CALLER OF 'DISPLAY'
00900 DISPLAY ENDP
00910 ;
00920 ; SUBROUTINE TO DISPLAY A CHARACTER ON THE SCREEN.
00930 ; ENTER WITH AL = CHARACTER TO BE DISPLAYED.
00940 ; USES VIDEO INTERFACE IN BIOS.
00950 ;
00960 DISPCHAR PROC     NEAR
00970           PUSH    BX      ;SAVE BX REGISTER
00980           MOV     BX,0    ;SELECT DISPLAY PAGE 0
00990           MOV     AH,14   ;FUNCTION CODE FOR 'WRITE'
01000           INT     10H     ;CALL VIDEO DRIVER IN BIOS
01010           POP     BX      ;RESTORE BX REGISTER
01020           RET             ;RETURN TO CALLER OF 'DISPCHAR
01030 DISPCHAR ENDP
01040 START     ENDP
01050 CODE      ENDS
01060           END     START
```

运行结果：

```
C>ex2-1    5
I HAVE MORE MEMORY THAN YOU---
C>ex2-1  10
ERROR !!! INVALID PARAMETER !!
```

说明：

① 程序在开始运行时,DS指向前缀区PSP，其中偏移量为80H的字节单元中是输入参数的长度，从81H开始是输入参数。程序中410—440是将参数长度及参数部分全部从PSP中搬入变量名为PARM的数据区中，然后再对它们进行处理。

② 程序在下列情况显示出错信息：参数长度不等于2，参数小于0或大于9。

③ 程序显示的信息长度都是30，在显示信息时利用软中断INT10H调用BIOS的字符显示子程序。

例2-2　台钟

功能：在命令行上输入“本程序名　hh：mm：ss”启动程序，然后按任何一键开始计时，这时IBM PC将成为一个相当准确的数字式自动计时台钟（这里hh代表小时，mm代表分，ss代表秒）。

程序：

```
00030 ;  DESK CLOCK  - ILLUSTRATES USE OF THE 8253 TIMER CHIP.
00040 STACK   SEGMENT PARA STACK 'STACK'
00050         DB      256 DUP (0)      ;256 BYTES OF STACK SPACE
00060 STACK   ENDS
00070 ;
00080 DATA    SEGMENT PARA PUBLIC 'DATA'
00090 COUNT100 DB     100      ;DIVIDE BY 100 SOFTWARE COUNTER
00100 TENHOUR DB      0        ;TENS DIGIT FOR HOUR
00110 HOUR    DB      0        ;ONES DIGIT FOR HOUR
00120         DB       :
00130 TENMIN  DB      0        ;TENS DIGIT FOR MINUTE
00140 MINUTE  DB      0        ;ONES DIGIT FOR MINUTE
00150         DB       :
00160 TENSEC  DB      0        ;TENS DIGIT FOR SECOND
00170 SECOND  DB      0        ;ONES DIGIT FOR SECOND
00180 DATA    ENDS
00190 ;
00200 CODE    SEGMENT PARA PUBLIC 'CODE'
00210 START   PROC    FAR
00220 ;
00230 ; STANDARD PROGRAM PROLOGUE EXCEPT RETAIN DS AS PTR TO PSP
00240 ;
00250         ASSUME  CS:CODE
00260         PUSH    DS       ;SAVE PSP SEG ADDR
00270         MOV     AX,0
00280         PUSH    AX       ;SAVE RET ADDR OFFSET (PSP+0)
00290         MOV     AX,DATA
00300         MOV     ES,AX    ;ESTABLISH EXTRA SEG ADDRESSABILITY
00310         ASSUME  ES:DATA
00320 ;
00330 ; MOVE 8 BYTE PARAMETER FROM PSP TO OUR DATA SEGMENT
00340
00350         MOV     SI,82H   ;SOURCE STRING OFFSET (WITHIN PSP)
00360         MOV     DI,OFFSET TENHOUR ;DEST. STRING OFFSET
00370         MOV     CX,8     ;STRING LENGTH TO MOVE
00375         CLD              ;SET 'FORWARD' STRING OPERATIONS
00380         REP     MOVSB    ;MOVE PARM INTO OUR DATA AREA
00390 ;
00400 ; ESTABLISH NORMAL DATA SEGMENT ADDRESSABILITY
00410 ;
00420         MOV     DS,AX
00430         ASSUME  DS:DATA
00440 ;
```

```
00450 ; WAIT FOR A KEY TO BE STRUCK TO  START' THE CLOCK:
00460 ;
00470         MOV     AH,0      ;SELECT BIOS FUNCTION = READ KEY
00480         INT     16H       ;CALL BIOS KEYBOARD ROUTINE
00490 ;
00500 ; SETUP OUR OWN TIMER INTERRUPT SERVICE ROUTINE:
00510 ;
00520         CLI               ;DISABLE ALL INTERRUPTS
00530         MOV     AX,0
00540         MOV     ES,AX     ;POINT EXTRA SEGMENT AT THE...
00550 ;               ...INTERRUPT SERVICE ROUTINE ADDRESS TABLE
00560         MOV     DI,20H    ;OFFSET OF ENTRY FOR TYPE CODE 08H
00570         MOV     AX,OFFSET TIMER ;OFFSET OF OUR SERVICE ROUTINE
00580         STOSW                   ;PLACE IT IN THE TABLE
00590         MOV     AX,CS           ;SEG OF OUR SERVICE ROUTINE
00600         STOSW                   ;PLACE IT IN THE TABLE
00610 ;
00620 ; PROGRAM CHANNEL 0 OF THE 8253 TIMER TO REQUEST AN
00630 ;  INTERRUPT 100 TIMES A SECOND:
00640 ;
00650         MOV     AL,36H    ;TIMER COMMAND: SELECT CHANNEL 0;
00660 ;                                         READ/WRITE LSB-MSB;
00670 ;                                         MODE 3: BINARY.
00680         OUT     43H,AL    ;SEND COMMAND TO 8253 COMMAND REG
00690         MOV     BX,11932 ;DESIRED COUNT VALUE FOR 100HZ RESULT
00700         MOV     AL,BL     ;SEND LSB OF VALUE FIRST
00710         OUT     40H,AL    ;SEND LSB TO LATCH REGISTER
00720         MOV     AL,BH     ;SEND MSB OF VALUE LAST
00730         OUT     40H,AL    ;SEND MSB TO LATCH REGISTER
00740 ;
00750 ; NOW PROGRAM 8259 INTERRUPT CONTROLLER TO ALLOW INTERRUPTS
00760 ;  FROM THE KEYBOARD AND THE TIMER:
00770 ;
00780         MOV     AL,0FCH ;ENABLE TIMER AND KYBD IRUPTS
00790         OUT     21H,AL  ;WRITE INTERRUPT MASK REGISTER
00800         STI             ;ENABLE INTERRUPTS TO THE 8088
00810 ;
00820 ; DISPLAY TIME ON THE SCREEN, WAIT FOR IT TO CHANGE AND THEN
00830 ;  DISPLAY IT AGAIN.  LOOP HERE FOREVER:
00840 ;
00850 FOREVER: MOV    BX,OFFSET TENHOUR       ;START OF STRING
00860         MOV     CX,8                    ;STRING LENGTH
00870 DISPCLK: MOV    AL,[BX]                 ;GET CHARACTER
00880         CALL    DISPCHAR                ;DISPLAY IT ON SCREEN
00890         INC     BX                      ;POINT TO NEXT CHARACTER
00900         LOOP    DISPCLK                 ;DO ALL 8 CHARACTERS
00910         MOV     AL,0DH                  ;ISSUE A CARRIAGE RETURN
00920         CALL    DISPCHAR
00930         MOV     AL,SECOND       ;GET DIGIT THAT CHANGES MOST OFTEN
00940 WAIT:   CMP     AL,SECOND       ;HAS IT CHANGED?
00950         JZ      WAIT            ;WAIT UNTIL IT DOES CHANGE
00960         JMP     FOREVER         ;AND THEN REFRESH THE DISPLAY
00970 ;
00980 ; TIMER IS OUR OWN TIMER INTERRUPT SERVICE ROUTINE:
00990 ;
01000 TIMER   PROC    FAR
01010         PUSH    AX        ;SAVE ALL ALTERED REGISTERS!!
01020 ;
01030 ; COUNT DOWN THE  COUNT100' COUNTER AND GENERATE THE
01040 ;  'SOFTWARE SIGNAL' WHEN IT REACHES ZERO:
01050 ;
```

```
01060           DEC     COUNT100            ;DECREMENT THE COUNTER
01070           JNZ     TIMERX
01080           MOV     COUNT100,100        ;RESET THE COUNTER FOR NEXT TIME
01090 ;
01100 ; WE FALL THROUGH TO THE ROUTINE BELOW WHEN THE COUNTER
01110 ;  REACHES ZERO, THIS IS ONCE A SECOND:
01120 ;
01130           INC     SECOND              ;TICK CLOCK BY ONE SECOND
01140           CMP     SECOND,'9'          ;DO WE HAVE TO ADJUST TENSEC?
01150           JLE     TIMERX              ;BRANCH IF NOT, WE'RE DONE!
01160           MOV     SECOND,'0'          ;SECOND OVERFLOWS, RESET IT
01170           INC     TENSEC              ;AND CARRY OVER TO THE NEXT DIGIT
01180           CMP     TENSEC,'6'          ;HAS A MINUTE GONE BY?
01190           JL      TIMERX              ;BRANCH OF NOT, WE'RE DONE!
01200           MOV     TENSEC,'0'          ;RESET TENSEC
01210           INC     MINUTE      ;AND CARRY OVER TO NEXT (TICK MINUTES)
01220           CMP     MINUTE,'9'  ;DO WE HAVE TO ADJUST TENMIN?
01230           JLE     TIMERX      ;BRANCH IF NOT, WE'RE DONE!
01240           MOV     MINUTE,'0'  ;MINUTE OVERFLOWS, RESET IT
01250           INC     TENMIN      ;AND CARRY OVER TO THE NEXT DIGIT
01260           CMP     TENMIN,'6'  ;HAS AN HOUR GONE BY?
01270           JL      TIMERX      ;BRANCH OF NOT, WE'RE DONE!
01280           MOV     TENMIN,'0'  ;RESET TENMIN
01290           INC     HOUR        ;AND CARRY OVER TO NEXT (TICK HOURS)
01300           CMP     HOUR,'9'    ;HAVE WE GONE PAST 9 O'CLOCK?
01310           JA      ADJHOUR     ;BRANCH IF YES, WE MUST ADJUST TENHOUR
01320           CMP     HOUR,'3' ;COULD WE HAVE JUST GONE PAST 12 O'CLOCK?
01330           JNZ     TIMERX              ;BRANCH IF NOT, WE'RE DONE!
01340           CMP     TENHOUR,'1'         ;DID WE JUST GO PAST 12 O'CLOCK?
01350           JNZ     TIMERX              ;BRANCH IF NOT, WE'RE DONE!
01360           MOV     HOUR,'1'            ;YES: SET 1 O'CLOCK
01370           MOV     TENHOUR,'0'         ;YES: SET 1 O'CLOCK
01380           JMP     SHORT TIMERX        ;WE'RE DONE!
01390 ADJHOUR:  INC     TENHOUR             ;CARRY OVER FROM 9 O'CLOCK...
01400           MOV     HOUR,'0'            ;... TO 10 O'CLOCK
01410 ;
01420 ; NOW INDICATE "END OF INTERRUPT" TO THE INTERRUPT CONTROLLER:
01430 ;
01440 TIMERX:   MOV     AL,20H              ;SEND "EOI" COMMAND...
01450           OUT     20H,AL              ;...TO 8259 COMMAND REGISTER
01460           POP     AX                  ;RESTORE ALL ALTERED REGISTERS
01470           IRET                        ;RETURN FROM INTERRUPT
01480 TIMER     ENDP
01490 ;
01500 ; SUBROUTINE TO DISPLAY A CHARACTER ON THE SCREEN.
01510 ; ENTER WITH AL = CHARACTER TO BE DISPLAYED.
01520 ; USES VIDEO INTERFACE IN BIOS.
01530 ;
01540 DISPCHAR PROC    NEAR
01550           PUSH    BX          ;SAVE BX REGISTER
01560           MOV     BX,0        ;SELECT DISPLAY PAGE 0
01570           MOV     AH,14       ;FUNCTION CODE FOR 'WRITE'
01580           INT     10H         ;CALL VIDEO DRIVER IN BIOS
01590           POP     BX          ;RESTORE BX REGISTER
01600           RET                 ;RETURN TO CALLER OF 'DISPCHAR'
01610 DISPCHAR ENDP
01620 START     ENDP
01630 CODE      ENDS
01640           END     START
```

说明：

① 程序采用与例2-1相似方式取得用户输入的现行时钟值，并送入数据区。

② 通过软中断INT16H调用BIOS的键盘输入子程序，等待用户打入任一键后开始计时。

③ 650—730是为时钟计时器置初值，43H端口是指8253三路定时器中的命令寄存器，所使用的命令字36H表示选用0号定时器，以3号模式（产生方波）计数等。向端口40H送出的计数器初值选用11932是要求8253每秒发出100次时钟中断（08H）。21H端口为中断屏蔽寄存器，用以开放外部中断。

④ 520—600是将系统中原来为08H中断设置的入口地址换成本程序中TIMER过程的始地址，这样就能保证在发生08H中断时，程序自动进入TIMER过程。

⑤ 程序的运行过程大致是：主程序不停地显示时钟值，当发生时钟中断08H时，转入TIMER过程。TIMER查看中断计数是否达到100次，如没有达到则返回主程序；如已达到，表明又过了1秒钟，则对时钟的值进行修改，然后再返回主程序。

例2-3　验证字符显示器的显示属性

功能：用八种不同的属性在屏幕上显示字母“A”。当从键盘上按下任一键后，清除屏幕并返回DOS。

程序：

```
00020 ; PROGRAM TO DEMONSTRATE DIFFERENT DISPLAY ATTRIBUTES
00030 ; ON THE MONOCHROME DISPLAY
00040 ;
00050 ;
00060 STACK    SEGMENT PARA STACK 'STACK
00070          DB      256 DUP(0)
00080 STACK    ENDS
00090 ;
00100 DATA     SEGMENT PARA PUBLIC 'DATA'
00110 ATTRIBUTES DB    07H      ;NORMAL
00120          DB      0FH      ;INTENSE
00130          DB      8FH      ;INTENSE/BLINKING
00140          DB      87H      ;BLINKING
00150          DB      01H      ;UNDERLINED
00160          DB      70H      ;REVERSE VIDEO
00170          DB      0F0H     ;REVERSE VIDEO/BLINKING
00180          DB      0F8H     ;REVERSE VIDEO/INTENSE/BLINKING
00190 DATA     ENDS
00200 ;
00210 CODE     SEGMENT PARA PUBLIC 'CODE'
00220 START    PROC    FAR
00230 ;
00240 ;STANDARD PROGRAM PROLOGUE
00250 ;
00260          ASSUME  CS:CODE
00270          PUSH    DS       ;SAVE PSP SEG ADDR
00280          MOV     AX,0
00290          PUSH    AX       ;SAVE RETURN ADDRESS OFFSET (PSP+0)
00300          MOV     AX,DATA
00310          MOV     DS,AX    ;ESTABLISH DATA SEGMENT ADDRESSABILITY
00320          ASSUME  DS:DATA
00330 ;
00340 ;PART 1: CLEAR THE DISPLAY
00350          MOV     AX,0B000H        ;SEGMENT ADDRESS OF MEMORY
00360 ;                                 ON THE MONOCHROME ADAPTER
```

```
00370           MOV     ES,AX     ;ES SEGMENT REGISTER WILL POINT
00380                             TO THE MEMORY ON THE ADAPTER CARD
00390           MOV     DI,0      ;INITIAL OFFSET ADDRESS INTO SEGMENT
00400           MOV     AL,       ;BLANK CHARACTER TO FILL DISPLAY
00410           MOV     AH,07H    ;ATTRIBUTE BYTE FOR NORMAL DISPLAY
00420           MOV     CX,2000   ;NUMBER OF WORDS OF MEMORY TO INITIALIZE
00430           CLD               ;SO STOSW GOES FORWARD IN MEMORY
00440           REP     STOSW     ;BLANK OUT THE DISPLAY
00450 ;
00460 ;PART 2: WRITE THE CHARACTER 'A' WITH 8 DIFFERENT ATTRIBUTES
00470 ;        DOWN THE CENTER OF THE SCREEN WITH A BLANK LINE
00480 ;        IN BETWEEN EACH CHARACTER
00490 ;
00500           MOV     CX,8      ;LOOP COUNTER FOR 8 ATTRIBUTE BYTES
00510           MOV     DI,240    ;INITIAL OFFSET INTO DISPLAY MEMORY
00520 ;                           FOR CENTER OF SECOND LINE
00530           MOV     BX,OFFSET ATTRIBUTES     ;POINTER TO THE
00540 ;                                          ATTRIBUTE BYTES
00550           MOV     AL,'A'    ;CHARACTER TO BE DISPLAYED
00560 ;LOOP TO DISPLAY 'A' WITH 8 ATTRIBUTES
00570 DISPLAY: MOV      AH,[BX]           ;GET NEW ATTRIBUTE BYTE
00580           MOV     ES:[DI],AX        ;MOVE CHARACTER CODE AND ATTRIBUTE
00590                                     BYTE TO ADAPTER MEMORY
00600           ADD     DI,320            ;POINT TO DISPLAY POSITION
00610                                     TWO LINES DOWN.
00620           IN      BX        ;POINT TO NEXT ATTRIBUTE BYTE
00630           LOOP    DISPLAY   ;DO IT FOR 8 ATTRIBUTE BYTES
00640 ;
00650 ;PART 3: AFTER DISPLAYING ALL THE ATTRIBUTES,
00660 ;        WAIT FOR A KEY TO BE STRUCK, THEN BLANK THE
00670 ;        DISPLAY AND RETURN TO DOS.
00680 ;
00690           MOV     AH,0      ;GET KEYBOARD INPUT CODE
00700           INT     16H       ;BIOS CALL
00710 ;WHEN CONTROL IS RETURNED WE MUST CLEAR THE SCREEN
00720           MOV     DI,0      ;INITIAL OFFSET ADDRESS
00730           MOV     AL,       ;BLANK CHARACTER
00740           MOV     AH,07H    ;ATTRIBUTE CODE FOR NORMAL DISPLAY
00750           MOV     CX,2000   ;NUMBER OF WORDS OF MEMORY TO INITIALIZE
00760           REP     STOSW     ;BLANK OUT THE SCREEN
00770           RET               ;RETURN TO DOS
00780 START     ENDP
00790 CODE      ENDS
00800           END     START
```

说明：

① 程序在数据段定义了八种不同属性的控制字节，定义方法见上册第一章第四节。

② 程序的第一部分通过往单色显示器字符/属性存贮器中送满空白字符以达到清屏幕的目的；第二部分是从屏幕上第2行的中间位置（显示字符/属性存贮器地址240）开始每隔一行（DI+320），顺序放入由数据区描述的八种不同属性的字符A。

③ 最后，程序等待用户任按一键，清屏后返回MS-DOS。

例2-4　文本文件列表

功能：在磁盘上打开一个文本文件，读出其中内容并显示在屏幕上。它的功能 比DOS的

内部命令TYPE要强些，例如它可以在显示满一屏后自行停止，待用户任打一键后继续显示以后的文件内容。

程序：

```
00030 ; PROGRAM TO LIST TEXT FILES ON THE DISPLAY.
00040 ; ILLUSTRATES USE OF SEQUENTIAL DISK I/O UNDER DOS.
00050 ;
00060 STACK    SEGMENT PARA STACK 'STACK'
00070          DB      256 DUP (0)      ;256 BYTES OF STACK SPACE
00080 STACK    ENDS
00090 ;
00100 DATA     SEGMENT PARA PUBLIC 'DATA'
00110 FCB      DB      36 DUP (0)       ;FILE CONTROL BLOCK
00120 LINECOUNT DB     0                ;COUNT OF LINES ON SCREEN
00130 CHARPOS DB       0                ;CHARACTER POSITION ON LINE
00140 DTA      DB      0                ;DISK TRANFER AREA
00150 ; ERROR MESSAGE:
00160 ERRMSG   DB      'FILE ACCESS ERROR !!!'
00170 DATA     ENDS
00180 ;
00190 CODE     SEGMENT PARA PUBLIC 'CODE'
00200 START    PROC    FAR
00210 ;
00220 ; STANDARD PROGRAM PROLOGUE EXCEPT RETAIN DS AS PTR TO PSP
00230 ;
00240          ASSUME  CS:CODE
00250          PUSH    DS       ;SAVE PSP SEG ADDR
00260          MOV     AX,0
00270          PUSH    AX       ;SAVE RET ADDR OFFSET (PSP+0)
00280          MOV     AX,DATA
00290          MOV     ES,AX    ;ESTABLISH EXTRA SEG ADDRESSABILITY
00300          ASSUME  ES:DATA
00310 ;
00320 ; MOVE FCB PARAMETER FROM PSP TO OUR DATA SEGMENT:
00330 ;
00340          MOV     SI,5CH   ;SOURCE STRING OFFSET (WITHIN PSP)
00350          MOV     DI,OFFSET FCB  ;DEST. STRING OFFSET
00360          MOV     CX,12    ;STRING LENGTH TO MOVE
00370          CLD              ;SET 'FORWARD' STRING OPERATIONS
00380          REP     MOVSB    ;MOVE PARM INTO OUR DATA AREA
00390 ;
00400 ; ESTABLISH NORMAL DATA SEGMENT ADDRESSABILITY
00410 ;
00420          MOV     DS,AX
00430          ASSUME  DS:DATA
00440 ;
00450 ; SET DTA AND OPEN FILE:
00460 ;
00470          MOV     DX,OFFSET DTA    ;ADDRESS OF DTA
00480          MOV     AH,1AH   ;DOS FUNCTION = 'SET DTA'
00490          INT     21H      ;INVOKE DOS FUNCTION
00500          MOV     DX,OFFSET FCB    ;ADDRESS OF FCB
00510          MOV     AH,0FH   ;DOS FUNCTION = 'OPEN FILE'
00520          INT     21H      ;INVOKE DOS FUNCTION
00530          CMP     AL,0     ;DID THE 'OPEN' WORK?
00540          JNZ     ERROR    ;BRANCH IF NOT
00550 ;
00560 ; INITIALIZE THE 'RECORD SIZE', 'CURRENT BLOCK , AND
00570 ;  'CURRENT RECORD' FIELDS WITHIN THE FCB:
00580 ;
00590          MOV     WORD PTR FCB+0CH,0       ;'CURRENT BLOCK' = 0
00600          MOV     WORD PTR FCB+0EH,1       ;'RECORD SIZE' = 1
```

```
00610           MOV     FCB+20H,0               ;'CURRENT RECORD' = 0
00620 ;
00630 ; READ A CHARACTER FROM THE FILE AND DISPLAY IT ON
00640 ;  THE SCREEN, CHECK FOR SPECIAL CONTROL CHARACTERS
00650 ;  'TAB' (09H), 'END OF FILE' (1AH), AND 'LINE FEED' (0AH)
00660 ;  AND HANDLE THEM ACCORDINGLY:
00670 ;
00680 AGAIN:    MOV     DX,OFFSET FCB   ;ADDRESS OF FCB
00690           MOV     AH,14H  ;DOS FUNCTION = 'SEQUENTIAL READ'
00700           INT     21H     ;INVOKE DOS FUNCTION
00710           CMP     AL,0    ;DID THE 'READ' WORK?
00720           JNZ     ERROR   ;BRANCH IF NOT
00730           MOV     AL,DTA  ;GET THE CHARACTER JUST READ
00740           CMP     AL,1AH  ;IS IT 'CONTROL Z'?
00750           JZ      EOF     ;BRANCH IF IT IS (INDICATES END OF FILE)
00760           CMP     AL,09H  ;IS IT 'TAB'?
00770           JZ      TAB     ;BRANCH IF IT IS (MUST EXPAND THE TAB
00780           CALL    DISPCHAR ;DISPLAY THE CHAR ON THE SCREEN
00790           INC     CHARPOS ;MAINTAIN CURRENT CHARACTER POSITION
00800           CMP     DTA,0AH ;WAS THAT THE END OF A LINE?
00810           JNZ     AGAIN   ;BRANCH IF NOT (GO DO MORE)
00820 ;
00830 ; A COMPLETE LINE HAS BEEN READ AND DISPLAYED, COUNT UP
00840 ;  THE NUMBER OF LINES ON THE SCREEN AND WAIT IF THE
00850 ;  SCREEN HAS BEEN FILLED:
00860 ;
00870           MOV     CHARPOS,0       ;RESET THE CHAR POSITION COUNTER
00880           INC     LINECOUNT       ;COUNT UP # OF LINES ON SCREEN
00890           CMP     LINECOUNT,24    ;IS THE SCREEN FULL?
00900           JNZ     AGAIN           ;BRANCH IF NOT (OK TO CONTINUE)
00910 ; THE SCREEN IS FULL, WAIT FOR THE USER TO HIT A KEY:
00920           MOV     AH,0            ;FUNCTION = 'READ KEYBOARD'
00930           INT     16H             ;CALL BIOS KEYBOARD ROUTINE
00940 ; THE USER HIT A KEY: ITS OK TO PROCEED:
00950           MOV     LINECOUNT,0     ;RESET COUNT OF LINES ON SCREEN
00960           JMP     AGAIN           ;RESUME MAIN LOOP
00970 ;
00980 ; A 'TAB' CHARACTER HAS BEEN ENCOUNTERED, DISPLAY BLANKS
00990 ;  UNTIL WE REACH THE NEXT TAB STOP:
01000 ;
01010 TAB:      MOV     AL,' '
01020           CALL    DISPCHAR        ;DISPLAY A BLANK
01030           INC     CHARPOS         ;MAINTAIN CHARACTER POSITION
01040           TEST    CHARPOS,7       ;ARE WE AT A TAB STOP?
01050           JZ      AGAIN           ;BRANCH IF WE ARE (TAB EXPANDED)
01060           JMP     TAB             ;CONTINUE UNTIL WE REACH TAB STOP
01070 ;
01080 ; END OF FILE HAS BEEN ENCOUNTERED, CLOSE THE FILE AND
01090 ;  RETURN TO DOS:
01100 ;
01110 EOF:      MOV     DX,OFFSET FCB   ;ADDRESS OF FCB
01120           MOV     AH,10H          ;DOS FUNCTION = 'CLOSE FILE'
01130           INT     21H             ;INVOKE DOS FUNCTION
01140           RET                     ;RETURN TO DOS
01150 ;
01160 ; A FILE ACCESS ERROR HAS OCCURRED, TELL USER AND QUIT:
01170 ;
01180 ERROR:    MOV     BX,OFFSET ERRMSG
01190           CALL    DISPLAY ;DISPLAY THE MESSAGE AT [BX]
01200           RET                     ;RETURN TO DOS
01210 ;
01220 ; SUBROUTINE TO DISPLAY A MESSAGE ON THE SCREEN.
```

```
01230 ; ENTER WITH BX -> MESSAGE TO BE DISPLAYED.
01240 ; MESSAGE IS ASSUMED TO BE 30 CHARACTERS LONG.
01250 ;
01260 DISPLAY PROC    NEAR
01270         MOV     CX,30     ;NUMBER OF CHARACTERS TO DISPLA
01280 DISP1:  MOV     AL,[BX]   ;GET NEXT CHARACTER TO DISPLAY
01290         CALL    DISPCHAR  ;DISPLAY IT
01300         INC     BX        ;POINT TO NEXT CHARACTER
01310         LOOP    DISP1     ;DO IT 30 TIMES
01320         MOV     AL,0DH    ;CARRIAGE RETURN
01330         CALL    DISPCHAR
01340         MOV     AL,0AH    ;LINE FEED
01350         CALL    DISPCHAR
01360         RET               ;RETURN TO CALLER OF ' DISPLAY'
01370 DISPLAY ENDP
01380 ;
01390 ; SUBROUTINE TO DISPLAY A CHARACTER ON THE SCREEN
01400 ; ENTER WITH AL = CHARACTER TO BE DISPLAYED
01410 ; USES VIDEO INTERFACE IN BIOS.
01420 ;
01430 DISPCHAR PROC   NEAR
01440         PUSH    BX        ;SAVE BX REGISTER
01450         MOV     BX,0      ;SELECT DISPLAY PAGE 0
01460         MOV     AH,14     ;FUNCTION CODE FOR 'WRITE'
01470         INT     10H       ;CALL VIDEO DRIVER IN BIOS
01480         POP     BX        ;RESTORE BX REGISTER
01490         RET               ;RETURN TO CALLER OF 'DISPCHAR
01500 DISPCHAR ENDP
01510 START   ENDP
01520 CODE    ENDS
01530         END     START
```

说明：

① 本程序通过系统功能调用21H来访问磁盘文件。它首先在数据段定义一个区域作为文件控制块（FCB），并定义一个DTA字节用以接收从磁盘文件中读出的信息。程序首先将程序前缀区（PSP）中的文件名和文件类型名送到FCB中相应位置，然后打开指定的文件（使用INT 21、AH=1AH），并将FCB中当前块号置为0，记录长度置为1，当前记录置为0。接着，采用功能调用（INT 21，AH=14H）顺序地读出文件中的记录。

② 由于记录长度为1，程序每次仅接收一个字符。字符0AH表示一行的结束，字符为CTRL-Z（1AH）时表示整个文件结束。

③ 程序每读一记录，系统会自动对FCB中当前块号、当前记录号等信息进行修改以达到顺序读出文件中各个记录的目的。字符读出后即显示在屏幕上，显示行计数达到24时（此时已满一屏），等待用户任打一键后再继续下去。当文件结束（碰到CTRL-Z）时关闭文件（INT 21，AH=10H）。

④ 程序在系统功能调用执行后将检查出口参数(AL)以确定是否有输入输出错，如有错则显示出错信息。

例2-5　列文件目录

功能：与DOS内部命令DIR相同，但本程序可在每行列出四个文件名，这样在屏幕上可同时看到92个文件。

程序：

```
00010 STACK   SEGMENT PARA STACK 'STACK'
00020         DB      256 DUP (0)      ;256 BYTES OF STACK SPACE
00030 STACK   ENDS
00040 DATA    SEGMENT PARA PUBLIC 'DATA'
00050 DIRECTORY       DB      3584 DUP (0)     ;DIRECTORY SECTORS
00060 NAMECOUNT       DB      0                ;COUNT OF NAMES
00070 ERRMSG  DB      'DISK ACCESS ERROR !!!
00080 DATA            ENDS
00090 ;
00100 CODE    SEGMENT PARA PUBLIC 'CODE'
00110 START   PROC    FAR
00120         ASSUME  CS:CODE
00130         PUSH    DS       ;SAVE PSP SEG ADDR
00140         MOV     AX,0
00150         PUSH    AX       ;SAVE RET ADDR OFFSET (PSP+0)
00160         MOV     AX,DATA
00170         MOV     DS,AX    ;ESTABLISH DATA SEG
00180         ASSUME  DS:DATA
00190         MOV     ES,AX    ;ESTABLISH EXTRA SEG
00200         ASSUME  ES:DATA
00210         MOV     CX,3     ;TRY TO DO THE 'READ' 3 TIMES
00220 RETRY:  PUSH    CX       ;SAVE RETRY COUNT
00230         MOV     BX,OFFSET DIRECTORY ;ADDRESS TO READ INTO
00240         MOV     AX,0     ;DRIVE A
00250         MOV     DX,5     ;LOGIC SECTOR 5
00260         MOV     CX,7     ;NUMBER OF SECTORS = 4
00270         INT     25H      ;CALL DOS DISK I/O ROUTINE
00280         pop     cx
00290         POP     CX       ;RESTORE RETRY COUNT
00300         JNC     READOK   ;BRANCH IF READ WAS OK
00310         MOV     AH,0     ;FUNCTION = 'READ'
00320         INT     13H
00330         LOOP    RETRY    ;TRY IT AGAIN
00340 ERROR:  MOV     BX,OFFSET ERRMSG
00350         CALL    DISPLAY  ;DISPLAY THE ERRMSG
00360         RET              ;RETURN TO DOS
00370 READOK: MOV     CX,112   ;NUMBER OF ENTRIES IN DIRECTORY
00380 NEXT:   CMP     BYTE PTR [BX],0E5H
00390         JZ      EMPTY
00400         CMP BYTE PTR [BX],0
00410         JZ      EMPTY    ;BRANCH IF THE ENTRY EMPTY
00420         PUSH    BX       ;SAVE START OF THIS ENTRY
00430         MOV     DL,8     ;LENGTH OF PRIMARY FILENAME
00440 PNAME:  MOV     AL,[BX] ;GET CHAR OF FILENAME
00450         CALL    DISPCHAR;DISPLAY IT
00460         INC     BX       ;GO ON TO NEXT
00470         DEC     DL       ;ANY MORE ?
00480         JNZ     PNAME    ;BRANCH IF THERE ARE
00490         MOV     AL,'.'   ;DELIMITER BETWEEN PRI AND EXT
00500         CALL    DISPCHAR
00510         MOV     DL,3     ;LENGTH OF EXT NAME
00520 ENAME:  MOV     AL,[BX] ;GET CHAR OF EXT NAME
00530         CALL    DISPCHAR;DISPLAY IT
00540         INC     BX       ;GO ON TO NEXT CHAR
00550         DEC     DL       ;ANY MORE ?
00560         JNZ     ENAME    ;BRANCH IF THERE ARE
00570         POP     BX       ;RESTORE START OF THIS ENTRY
00580         MOV     DL,8     ;NUMBER OF BLANKS
```

```
00590  BLANK:  MOV     AL,' '
00600          CALL    DISPCHAR
00610          DEC     DL
00620          JNZ     BLANK
00630          INC     NAMECOUNT ;# OF NAMES CURRENTLY ON SCREEN
00640          CMP     NAMECOUNT,92  ;IS SCREEN FULL ?
00650          JNZ     EMPTY    ;BRANCH IF NOT
00660          MOV     AH,0     ;FUNCTION = 'READ KEYBOARD'
00670          INT     16H      ;CALL BIOS KEYBOARD ROUTINE
00680          MOV     NAMECOUNT,0  ;RESET COUNT OF NAMES
00690  EMPTY:  ADD     BX,32    ;POINT TO NEXT DIRECTORY
00700          LOOP    NEXT
00710          RET
00720  ;
00730  DISPLAY PROC    NEAR
00740          MOV     CX,30    ;NUMBERS OF CHAR TO DISPLAY
00750  DISP1:  MOV     AL,[BX]  ;GET NEXT CHAR
00760          CALL    DISPCHAR ;DISPLAY IT
00770          INC     BX       ;POINT TO NEXT CHAR
00780          LOOP    DISP1    ;DO IT 30 TIMES
00790          MOV     AL,0DH   ;CARRIAGE RETURN
00800          CALL    DISPCHAR
00810          MOV     AL,0AH   ;LINE FEED
00820          CALL    DISPCHAR
00830          RET              ;RETURN TO CALLER
00840  DISPLAY ENDP
00850  ;
00860  DISPCHAR        PROC    NEAR
00870          PUSH    BX       ;SAVE BX REGISTER
00880          MOV     BX,0     ;SELECT DISPLAY PAGE 0
00890          MOV     AH,14    ;FUNCTION CODE FOR 'WRITE'
00900          INT     10H      ;CALL VIDEO DRIVER
00910          POP     BX       ;RESTORE BX REGISTER
00920          RET              ;RETURN TO CALLER
00930  DISPCHAR        ENDP
00940  START   ENDP
00950  CODE    ENDS
00960          END     START
```

说明：

① 程序在数据段开辟了一个目录区，用来存放A盘上的112个目录项，这是系统对5¼时双面软盘所允许的最大目录数。

② 240—270用软中断（INT 25H）将A驱动器中双面软盘上的全部目录项读入名为DIRECTORY的3584（32×112）个字节的数据区中，程序还利用中断的输出参数（标志位CF），在读出有错时连续重读三次。三次都错时显示出错信息。值得注意的是调用软中断时，由于盘上每道为9个扇区，所以目录区是从0道0面6区开始的，而程序中采用的是INT 25H，用的是逻辑扇区号，所以使用5作为逻辑扇区号。

③ 本程序对每个目录项只用到了文件名和类型名，每行显示四个文件名(带类型名)，两文件名之间空8个字符位置。当文件名显示计数达到92时，表明已满一屏，这时程序等待用户按任一键继续。由于程序中忽略了文件的属性，所以隐含文件名也能列出来。

例2-6　信息显示

功能：它包含两个程序模块，要求用户分块汇编，然后连接成一个完整的程序。第1个模块是主程序，它首先清屏幕，然后提示用户输入一个名字，接着调用另一模块将这个名字

显示若干遍，最后返回DOS。这个程序功能虽不复杂，但较多地使用了宏汇编语言的各种伪操作命令，对读者掌握和理解汇编语言有一定帮助。

模块1：

```
00010  PAGE 27,132
00020  TITLE CALLER  - SAMPLE ASSEMBLER ROUTINE
00030  COMMENT *
00040          PROLOG - MODULE DESCRIPTION
00050  This program will demonstrate how to write two programs
00060  that will have to be linked together. The first program
00070  ,called CALLER will do the following:
00080          CLEAR THE VIDEO MONITOR
00090          MOVE THE CURSOR TO TOP LEFT POSITION (0,0)
00100          PRINT A MESSAGE PROMPTING FOR A NAME
00110          WAIT FOR THE USER TO TYPE IN THE NAME
00120          CLEAR THE VIDEO MONITOR
00130          CALL THE SECOND ROUTINE TO PRINT THE NAME
00140          TERMINATE  *
00150          EXTRN RECEIVR : FAR     ;EXTERNAL SUBROUTINE
00160          SUBTTL DEFINITIONS OF MACROS
00170                  PAGE
00180  SCROLL  MACRO
00190          MOV     AX,600H          ;AH=6 Request scroll
00200          BIOSCALL         ;Invoke BIOS to scroll the window
00210          ENDM
00220  ;
00230  KEYBOARD MACRO
00240          MOV     AH,10    ;Function = read keyboard
00250          DOSCALL          ;Invoke DOS to read keyboard
00260          ENDM
00270  ;
00280  LOCATE  MACRO
00290          MOV     AH,2     ;Function = locate
00300          BIOSCALL
00310          ENDM
00320          PAGE
00330  DISPLAY MACRO   TEXT
00340          MOV     DX,OFFSET TEXT ;Point to the message
00350          MOV     AH,9     ;Function = console output
00360          DOSCALL          ;Invoke DOS to display
00370          ENDM
00380  ;
00390  DOSCALL MACRO
00400          INT     21H      ;Request DOS service, id in AH
00410          ENDM
00420  BIOSCALL MACRO
00430          INT     10H      ;Request BIOS service, id in AH
00440          ENDM
00450  SUBTTL DESCRIPTION OF THE DOS READ PARAMETERS
00460          PAGE
00470  ;       This STRUCTURE defines the parameter list
00480  MAXCHAR EQU     25               ;Limit of 25 char for input
00490  ;
00500  PARMLIST STRUC
00510  BUFSIZE DB      MAXCHAR ;Limit of number of char
00520  NAMESIZE DB     ?       ;Number of chars typed
00530  NAMETEXT DB     MAXCHAR DUP(' ');Text of user response
00540  TERMINATOR DB   '$'     ;Text delimiter
00550  PARMLIST ENDS
00560  SUBTTL DESCRIPTION OF THE STACK SEGMENT
00570          PAGE
00580  STACK   SEGMENT PARA STACK 'STACK'
```

```
00590           DB      64 DUP('STACK     ')
00600   STACK   ENDS
00610   SUBTTL DESCRIPTION OF THE DATA WORKAREA
00620           PAGE
00630   WORKAREA SEGMENT PARA PUBLIC 'DATA'
00640   PARMS   PARMLIST <>     ;Define area for parameters
00650   MESSG   DB      'What is your name?  ' ;Prompt msg
00660           DB      '$'     ;End of prompt msg
00670   WORKAREA ENDS
00680   SUBTTL DESCRIPTION OF DOS INTERFACES
00690           PAGE
00700   CSEG    SEGMENT PARA PUBLIC 'CODE'
00710   START   PROC    FAR
00720           ASSUME CS:CSEG,DS:WORKAREA,SS:STACK,ES:NOTHING
00730   SUBTTL ESTABLISH ENTRY LINKAGE FROM DOS
00740           PAGE
00750   ;Set up the stack to contain the proper values
00760   ;so this  program can return to DOS.
00770   ;
00780           PUSH    DS      ;Save psp seg addr
00790           SUB     AX,AX   ;Clear AX
00800           PUSH    AX      ;Save ret addr offset (psp+0)
00810           MOV     AX,WORKAREA     ;Get addr of workarea
00820           MOV     DS,AX   ;Establish data seg
00830   ;
00840   SUBTTL CLEAR THE SCREEN,PUT CURSOR AT (0,0)
00850           PAGE
00860           CALL    CLS     ;Clear screen
00870           MOV     DX,0    ;DH=0,DL=0 (DH=Row,DL=Column)
00880           MOV     BH,0    ;Screen # to use
00890   ;                        Call to BIOS: This is where
00900           LOCATE          ;Do the locate
00910   SUBTTL DISPLAY A PROMPT MESSAGE TO CONSOLE
00920           PAGE
00930           DISPLAY MESSG   ;Call DOS to print message
00940   SUBTTL READ FROM THE KEYBOARD
00950           PAGE
00960           MOV     DX,OFFSET PARMS ;Addr to put name
00970           KEYBOARD        ;Read from keyboard
00980           CALL    CLS     ;Call clear screen routine
00990   SUBTTL INVOKE AN EXTERNAL SUBROUTINE
01000           PAGE
01010           MOV     AH,0    ;Clear high byte of reg
01020           MOV     AL,PARMS.NAMESIZE ;Get count of chars
01030           MOV     SI,AX   ;Pass char count to subroutine
01040           MOV     BX,OFFSET PARMS.NAMETEXT;Pass name addr
01050   ;
01060           CALL    RECEIVR
01070   SUBTTL  RESTORE THE ENTRY CONDITIONS,RETURN TO DOS
01080           PAGE
01090           RET             ;Far return to DOS
01100   START   ENDP
01110   SUBTTL COMMON CODE LOCAL SUBROUTINE , CLEARS SCREEN
01120           PAGE
01130   CLS     PROC    NEAR
01140           MOV     CX,0    ;Upper left coord (0,0)
01150           MOV     DX,2479H ;Bottom right position
01160           MOV     BH,7    ;Normal attribute for CLS
01170           SCROLL          ;Clear entire screen
01180           RET             ;Return to caller
01190   CLS     ENDP
01200   CSEG    ENDS
01210           END     START
```

模块 2：

```
00010           PAGE 27,132
00020   TITLE RECEIVR - SUBROUTINE TO DISPLAY NAME
00030   ;       PROLOG - MODULE DESCRIPTION
00040   ;This is a subroutine invoked by the main routine
00050   ;It only requires two parameters:
00060   ; BX=offset to the name that was typed in
00070   ; SI=number of chars , including the carriage return
00080   ;It will do the following:
00090   ;       LOCATE THE CURSOR BACK TO POSITION 0,0
00100   ;       REPLACE THE ODH IN THE NAME TO A 20H
00110   ;       PRINT NAME MANY TIMES
00120   ;       RETURN TO CALLING PROGRAM (CALLER)
00130   SUBTTL DEFINITION OF MACROS
00140           PAGE
00150   LOCATE  MACRO
00160           MOV     AH,2    ;Function = locate
00170           INT     10H     ;BIOS video call
00180           ENDM
00190   ;
00200   DOSCALL MACRO
00210           INT     21H     ;Request DOS service,id in AH
00220           ENDM
00230   ;
00240   ;                       ;locate EQU
00250   SPACE   EQU             ;Character blank
00260   LOOPCTR EQU     300     ;Loop counter,number of times
00270   SUBTTL SUBROUTINE ENTRY POINT,POSITION CURSOR
00280           PAGE
00290   SUBSEG SEGMENT PARA PUBLIC 'CODE'
00300   RECEIVR PROC    FAR
00310           ASSUME  CS:SUBSEG
00320           PUBLIC  RECEIVR
00330   ;
00340   ;                       ENTRY POINT
00350           PUSH    BX      ;Save addr of name
00360   ;
00370   ;this first routine does the locate then replaces
00380   ;the ODH(carriage return) to a 20H (space)
00390   ;
00400           MOV     DX,0    ;Position 0,0
00410           MOV     BH,0    ;Screen # to use
00420           LOCATE          ;Move cursor
00430   ;
00440                           ;SI has length of name
00450           POP     BX      ;Restore addr of name char string
00460           MOV     BYTE PTR [BX][SI],SPACE ;Replace carriage return
00470   SUBTTL DISPLAY INPUT CHAR STRING MANY TIMES ON SCREEN
00480           PAGE
00490           MOV     CX,LOOPCTR      ;How many times to print
00500           MOV     AH,9            ;DOS print message routine
00510   ;
00520   PRINTMR: MOV    DX,BX           ;Point to name
00530           DOSCALL                 ;Print it
00540           LOOP    PRINTMR         ;Go back and again
00550   ;
00560           RET                     ;Return to the main program
00570   RECEIVR ENDP
00580   SUBSEG  ENDS
00590           END
```

说明：

① 程序首先通过过程调用清除屏幕，过程中嵌套地使用了宏指令，通过中断INT 10（AH＝6）使屏幕上滚达到清屏幕的目的。然后再通过宏指令用中断INT 10（AH＝2）把光标定在屏幕左上角。

② 程序模块1中的带形式参数宏指令DISPLAY用于显示提示信息："What is your name？"。宏指令KEYBOARD通过系统功能调用（INT 21，AH＝10）把用户从键盘上输入的名字送往一个类型为结构的变量PARMS中，然后通过寄存器BX和SI把名字的位置及字符数目传递给程序模块2。

③ 程序模块2主要用于将用户输入的名字显示300遍，这是通过系统功能调用(INT21，AH＝9）来实现的。

第三章　8087协处理器及其在IBM PC中的使用

目前，8位和16位的微处理器芯片，一般不具备通用计算机那样复杂的数值运算指令，因此不能够很好地承担高速数值运算的任务。为此，IBM PC使用INTEL8087（IBM PC/AT使用80287）作为其辅助处理器。8087是一种专门用于处理数值数据的处理器，它协同系统中的中央处理器（CPU）一起操作，有效地扩充了8088/8086CPU的寄存器和指令系统，增加了新的数据类型，大大地提高了系统的数值计算速度和能力。所以，8087也叫作CPU的协处理器。

本章首先介绍8087芯片的功能和结构，然后介绍8087的数据类型与格式、指令及其程序设计方法，最后再扼要地说明如何使用8087支持IBM PC的一些高级语言。

第一节　概　述

INTEL8087是INTEL公司在八十年代开发的一种单片式数值数据处理器硬件，它和美国电气与电子工程师协会（IEEE）在1979年为微机及小型机推荐的二进制浮点运算工业标准相一致，具有单精度和双精度处理的所有功能，并进行了许多新的扩充。

一个8088或8086和一个8087相结合之后，对于程序员来讲仍如同是一台处理器，因为8087只是给CPU增加了新的数据类型和寄存器，以及增加了一些新的、功能更强的指令而已。

8088/8086只能对8位或16位字长的二进制数和十进制数进行操作，而8087可以处理的数据类型却要复杂得多（见表3-1）。

表3-1　8087的数据类型

数据类型	位数（二进制）	有效位数（十进制）	数的大约范围（十进制）
单字整数	16	4	$-32768 \leqslant X \leqslant +32767$
短整数	32	9	$-2\times10^{9} \leqslant X \leqslant +2\times10^{9}$
长整数	64	18	$-9\times10^{18} \leqslant X \leqslant +9\times10^{18}$
装配十进制数	80	18	$-99\cdots99 \leqslant X \leqslant +99\cdots99$（18位数）
短实数	32	6—7	$8.43\times10^{-37} \leqslant \lvert X \rvert \leqslant 3.37\times10^{38}$
长实数	64	15—16	$4.19\times10^{-307} \leqslant \lvert X \rvert \leqslant 1.67\times10^{308}$
临时实数	80	19	$3.4\times10^{-4932} \leqslant \lvert X \rvert \leqslant 1.2\times10^{4932}$

其中，前六种是外部类型，最后一种临时实数为8087的内部类型。所有不同外部类型的整数或实数在8087内部都是用统一的临时实数类型来表示并进行运算的。8087硬件在取数和存数操作时，能自动进行外部类型与临时实数类型之间的格式转换，程序员对此毫无感觉。

这样芯片内部的结构有所简化，而又不会增加程序设计的负担。

在8087中，由八个寄存器组成了一个寄存器栈，每个寄存器长度分别为80位，正好能存放一个临时实数。8087的所有运算处理操作，都是对栈中寄存器的内容进行的。由于栈空间比较大，在计算中所用的常数以及中间结果也可以存放在栈中，从而减少了不必要的主存贮器访问次数。

8087对于上述各种数据类型，能够进行多种不同的运算和处理。表3-2给出了8087指令的类型及其主要指令。

表3-2　8087的主要指令

类　别	指　令
数据传送	取数（所有数据类型）、存数（所有数据类型）、交换
算术运算	加、减、乘、除、反减、反除、换算、求余数、取整、变符号、取绝对值、开平方、分解实数
比　较	比较、检验、测试
超越函数	正切、反正切、2^X-1、$Y\cdot\log_2(X+1)$、$Y\cdot\log_2(X)$
常　数	0、1、π、$\log_{10}2$、$\log_e2$、$\log_210$、$\log_2e$
处理器控制	取控制字、存控制字、存状态字、取环境、存环境、保护、恢复、允许中断、禁止中断、清除事故、初始化

使用了8087之后，科学和工程计算问题中的许多常用运算（实数的四则运算，常用函数等）都能由硬件直接完成，不必再使用8088/8086指令所组成的仿真程序来解释执行。因此，数值运算的速度大约能提高10—100倍。关于8087指令的执行时间与对应的8088/8086仿真程序的执行时间的对比可参见表3-3。

表3-3　8087指令与8088/8086仿真程序的速度对比

指　令	近似的执行时间（μS）（5MHz时钟）	
	8087	8088/8086仿真
乘（单精度）	19	1,600
乘（双精度）	27	2,100
加	17	1,600
除（单精度）	39	3,200
比较	9	1,300
取数（单精度）	9	1,700
存数（单精度）	18	1,200
平方根运算	36	19,600
正切运算	90	13,000
指数运算	100	17,100

8087芯片的安装和使用比较简单。用户只需要打开IBM PC机箱的盖板，找到与8088处

理器相邻的空插座，把8087芯片的缺口方向对准插座缺口方向插入即可。操作中务必注意防止静电损坏芯片。芯片插好后，把底板上一号组合开关中的第2个开关，从“ON”状态拨到“OFF”状态，表示系统中已经安装了8087协处理器。需要注意的是，美国INTEL公司早期生产的8088芯片不能与8087配用，否则8087无法正常工作。

系统中装上8087之后，必须使用可以支持8087操作的汇编程序和高级语言的解释或编译程序，才能发挥8087的高速数值计算的效用。但用高级语言书写的源程序不必修改，只需要重新编译一次即可，所以8087的使用十分方便。关于在高级语言编译系统中如何使用8087的问题，将在本章第五节再作专门介绍。

第二节　8087的逻辑结构

8087协处理器的逻辑结构如图3-1所示，它可以分成控制器（CU）和运算器（NEU）两个部分。CU负责取指令、读出或写入存贮器操作数，执行处理器控制类的指令。NEU负责进行涉及寄存器堆栈的一切指令，包括算术运算指令、比较指令、超越函数指令、常数指令以及数据传送指令。CU和NEU可相互独立地进行操作，而CU和CPU则保持

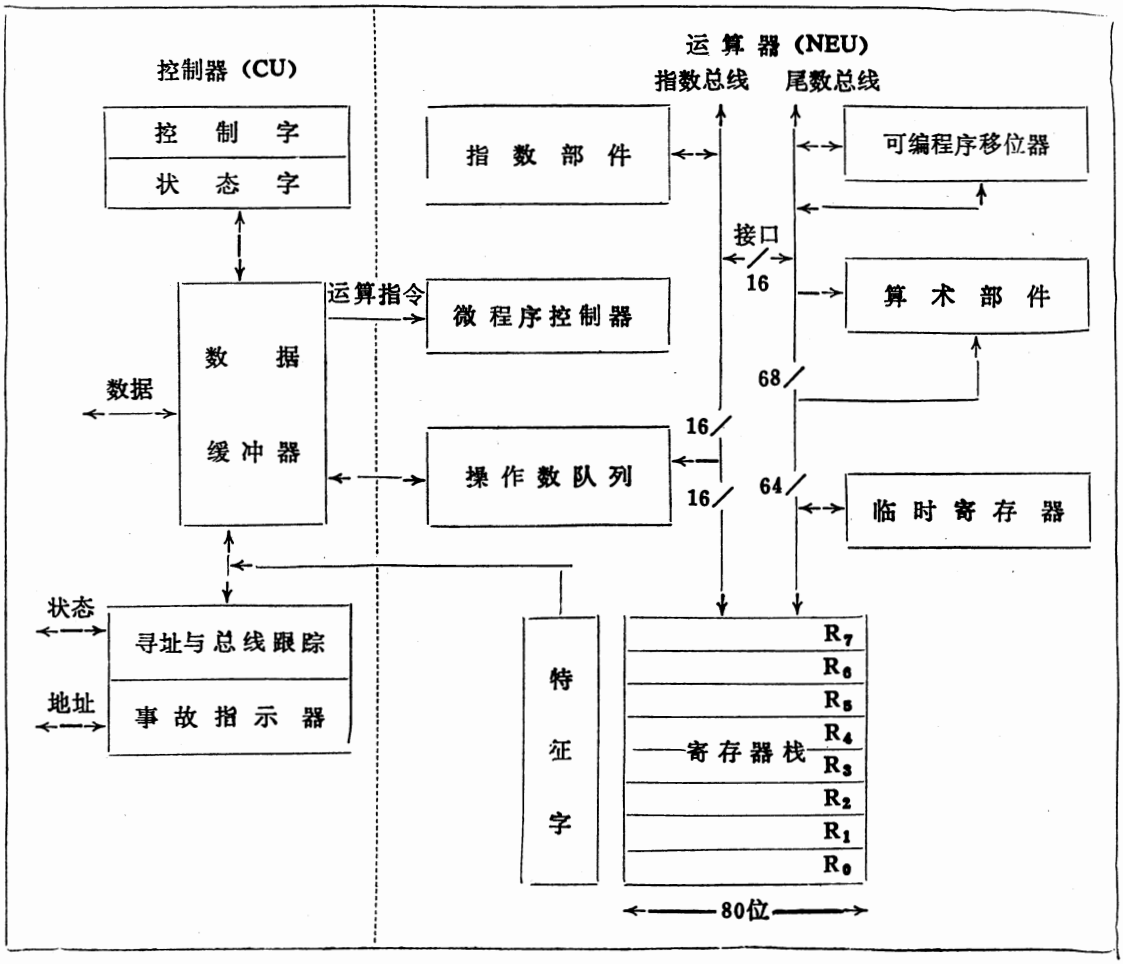


图3-1　8087协处理器结构框图

同步，使8087与8088能同步获取指令并各自对其译码。当它识别到一条8087指令时，即负责完成该指令所规定的全部操作。

1．寄存器栈与特征字

寄存器栈中有八个寄存器，每个80位。一个寄存器正好存放一个临时实数，它由符号位、阶码及尾数三部分组成，见图3－2。这八个寄存器通常是一个整体，头尾相接，组成一个能容纳八个临时实数的后进先出的堆栈。

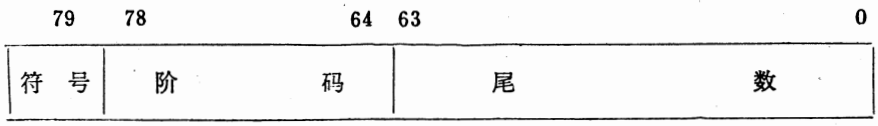


图3-2 栈寄存器结构

在状态字中的ST字段是一个指针，它总是指向当前栈顶寄存器。入栈操作把ST减1后再把数值装入新的栈顶寄存器，出栈操作是先取出当前栈顶寄存器中的值，然后将ST加1。图3－3是入栈操作和出栈操作的一个示意图。

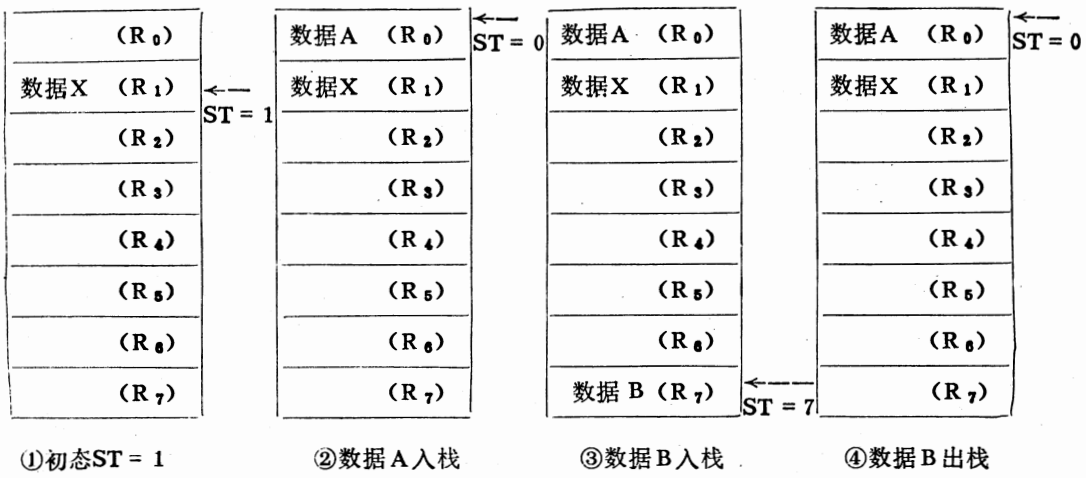
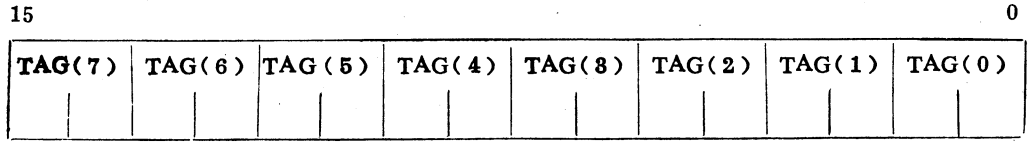


图3-3 入栈和出栈操作示意图

8087指令对寄存器堆栈中的数进行操作时，若不明确指出是哪一个寄存器，则总是指针ST所指向的那个寄存器（即栈顶）。如指令FSQRT表示将栈顶的数开平方。如果指令中需要明确指出对哪一个（或两个）寄存器中的数进行操作，则必须以栈顶指针ST为依据，用ST(i)表示参加运算的寄存器，ST(i)表示栈中从ST算起的第i个寄存器（0≤i≤7）。例如，若ST＝011B（R3寄存器是栈顶），那末指令FADD ST、ST(2)就表示R3寄存器和R5寄存器的内容相加。若执行该指令时ST＝6（R6为栈顶），则表示R6与R1的内容相加。8087的这种堆栈结构以及相对于栈顶进行寄存器寻址的方法，对简化子程序的编制极为有利。

为了能表征栈中每个寄存器R 0 —R 7 的状态，8087中使用了一个特征寄存器，其 中 每两位与一个寄存器对应，其特征值的编码和含义如图 3 - 4 所示。



特征值：“00”寄存器内容为有效值（规格化或非规格化）

“01”寄存器内容为零

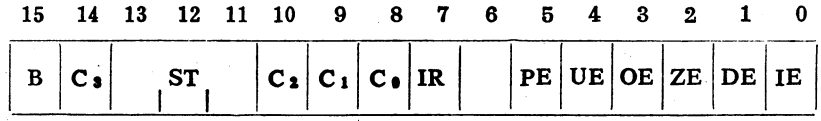
“10”寄存器内容为特殊值

“11”寄存器空

图3-4 特征字格式

2．状态字

状态字用来反映8087的操作状态，它可以被8087指令装入内存，再由CPU指 令 进 行 检测。状态字的格式如图 3 - 5 。



事故标志位：（为 1 时发生事故）

DE：不可规格化操作数

OE：上溢

PE：精度事故

IR：中断申请位

ST：栈顶指针

IE：无效操作

ZE：除数为 0

UE：下溢

第 6 位不用（备用位）

C_0—C_3：条件码

B：忙位

图3-5 状态字格式

第15位是“忙”位，B＝ 1 表示8087正在执行指令，B＝ 0 表示它空闲。

第 0 — 5 位表示执行指令时，8087检查出有异常事故发生，不同的事故类型使相应的标志位置为“ 1 ”。

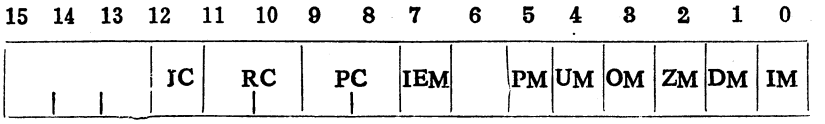
第 7 位是中断申请位IR。当某事故标志位为“ 1 ”而控制字中相应事故屏蔽位为“ 0 ”时，则中断申请位为“ 1 ”。

第11—13位是当前栈顶指针。如ST＝000，表示R 0 为栈顶寄存器；ST＝001，表示R 1 为栈顶寄存器等。注意，在ST＝000时，入栈操作将使ST＝111；而ST＝111时，出栈操作将使ST＝000。

第 8 —10和14位为条件码，它们的定义在第三节讲到有关比较、测试、检验等指 令 时再作说明。

3．控制字和事故指示器

为了满足不同的使用要求，8087提供了多种处理方式的选择功能。这些选择是通过把内存中的一个字装入控制字寄存器来进行的。图3-6是控制字中各字段的格式和编码。其中舍入控制RC占两位。当计算结果不能用目标数据类型精确地表示出来时，需要进行舍入。



事故处理屏蔽位：（为1时屏蔽）

DM：不可规格化操作数

OM：上溢

PM：精度事故

PC：精度控制

IC：无穷大控制

IM：无效操作

ZM：除数为零

UM：下溢

IEM：中断允许屏蔽

RC：舍入控制

第6，13—15位：保留位

图3-6　控制字格式

8087有四种舍入方式，见表3-4。舍入处理过程的步骤为：假设b是不能精确表示的一个数，8087就选用两个最接近b而且可以用目标数据类型表示的数a和c，使得a<b<c，然后处理器根据RC字段选择其中之一。8087初始化时　RC字段为00。

表3-4　RC中的舍入控制

RC字段	舍入方式	舍入结果
0 0	最近舍入	舍入到最接近的数，如果一样接近，选择偶数
0 1	向下舍入（趋向－∞）	a
1 0	向上舍入（趋向＋∞）	c
1 1	截尾（趋向0）	a和c中绝对值较小者

精度是根据控制字中的PC字段来选择的，可以选择64位、53位和24位精度进行实数运算。通常都采用64位精度，其它精度是考虑到IEEE标准而设计的。

IC字段为无穷大控制。当IC＝0时，8087没有＋∞和－∞之分；而当IC＝1时，＋∞和－∞有两种不同的编码表示。8087初始化时，IC＝0。

控制字中的第0—5位，是用来控制8087执行指令中发生异常事故时是否需要向CPU申请中断的，这六位分别与六类事故对应。“1”表示屏蔽，“0”表示需要申请中断处理。8087指令执行时常见的六类事故是：

① 无效操作　例如，堆栈已满尚需操作数入栈，堆栈已空尚需操作数出栈，执行一次

0/0或负数开平方之类的不确定操作，操作数不是数值数据等。

② 不可规格化的操作数　参加运算的操作数极小，规格化时将发生下溢等。

③ 被零除　　除法操作时发现除数为0。

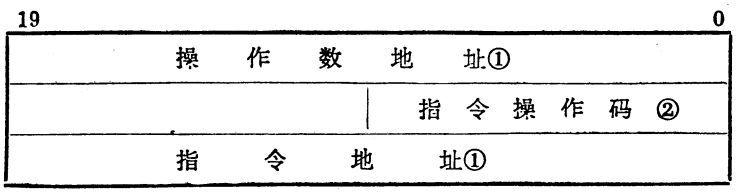
④ 上溢　　结果操作数指数太大无法表示。

⑤ 下溢　　结果操作数指数太小无法表示。

⑥ 精度事故　操作结果不能用目标操作数类型精确地表示，必须进行舍入处理。

通常，除了无效操作之外，对其它五类事故，8087都可以自行“合理”地进行处理，不一定需要向CPU申请中断。当发生某种事故而相应的屏蔽位为“0时，则引起状态字中中断申请位IR置“1”。如果这时控制字中的中断允许屏蔽位IEM又处于“0”时，则8087将向8088/8086 CPU发出中断申请信号，请求引出用户自行编写的事故处理程序来进行处置。

事故指示器（见图3-7）供用户编写事故处理程序时使用。它可以提供事故发生时的指令地址、操作码、操作数地址等信息。事故处理程序可以把这些信息取入内存进行事故分析。



①20位实际地址

②11位操作码，其中最高5位为11011

图3-7　事故指示器格式

8087初始化之后，所有事故屏蔽位及中断允许屏蔽位均为“1”。

4．8087和8088的互连与通讯

（1）8087和8088的互连

在IBM PC中协处理器8087与主处理器8088的连接如图3-8。由图可见，它们公用一个

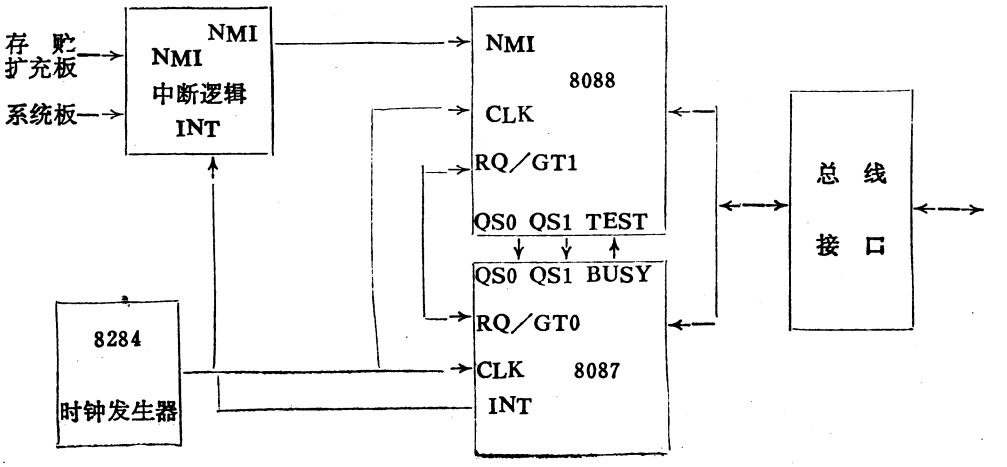


图3-8　PC机中8087与8088的互连

时钟发生器和系统总线接口器件。CPU的队列状态线（QS 0和QS 1）使8087能和CPU同步地获取指令并各自对其译码。8087的BUSY（忙）信号通知CPU它是否正在操作，CPU的WAIT指令可测试这个信号以确定8087是否可以执行下一条指令。

（2）存贮器访问

8087的所有机器指令高五位都是“11011”，8088则把它当作指令ESC，并按常规产生操作数地址后指令即告结束。因此，8088的任何一种内存寻址方式都能用于8087的存贮器操作数。当8087译出一条这样的指令后，如果指令要从内存中取操作数，那么它就从总线上捕获8088形成的操作数第1个字的地址，该地址是CPU启动“虚读周期”而放在总线上的。如果操作数长于一个字，8087通过请求/允许线(RQ/GT 0）请求使用局部总线，并不断递增它所保存的地址，连续执行总线周期直至获得全部数据为止。

如果进行的操作是把8087中的数存到存贮器，那么8087和CPU都不管由于CPU虚读而放在总线上的数据，而由8087申请局部总线，连续地向存贮器写入操作数。

（3）主处理器与协处理器的同步

在8087指令执行时，CPU完成ESC的处理远比8087来得快。例如，8087需要用180个时钟周期才能完成平方根运算指令，而CPU对此条指令的解释只要两个时钟周期，此时CPU将对下一条指令进行译码并执行，而8087在进行平方根运算的同时，它的控制器CU也在做同样的工作。如果接下去的指令都不是8087指令，那么此时8087和CPU都能正常地同步工作。一般说来，CPU和8087能否正常地同步工作受到以下两点限制：

* 如果8087的运算器正忙于执行前一条指令，那末它就一定不能再执行需要使用运算器的8087指令。

* 如果8087正在访问存贮器操作数，那末在此操作完成之前，CPU不该访问那个单元。

对前一种情况，8088可以采用WAIT指令等待。每当8087运行时，它使BUSY线有效，这个信号被连到CPU的$\overline{TEST}$端。WAIT指令的意义是：当$\overline{TEST}$有效就等待，CPU每隔5个时钟周期检查一次$\overline{TEST}$端。因此，可在程序中每条要使用运算器的8087指令前都放一条WAIT指令，以保证运算器空闲后再执行新的8087指令。因为除了控制类指令外，所有8087指令都涉及运算器。为简化设计，WAIT指令由语言翻译程序自动提供。对于后一种情况，可用8087的FWAIT指令来解决。当8087指令后面新的一条CPU指令试图对8087指令正在访问的同一存贮单元进行访问时，只要将FWAIT指令放在该8087指令后面，FWAIT指令会产生一条8088的WAIT指令。

8087将其请求/允许线($\overline{RQ}/\overline{GT}_0$）接在CPU的请求/允许线（$\overline{RQ}/\overline{GT}_1$）上，这一连线是双向的，以便与CPU公用局部总线传送数据。8087在完成数据传送之后，就将局部总线归还CPU。而当8087指令向存贮器写数据或从存贮器读多于一个字的数据时，8087就迫使CPU转让局部总线。

（4）8087中断

IBM PC机中，8087的INT是经过NM（不可屏蔽中断）中断逻辑向CPU申请中断的。下列三种情况下，意外事故不会引起CPU中断：

① 8087控制字中的事故屏蔽位为“1”或中断允许屏蔽位为“1”。

② 系统板的1号组合开关的第2个开关位于“on”。

③ NMI屏蔽寄存器为“0”。

在开电源时，NMI屏蔽寄存器被清0，禁止NMI。任何使用协处理器的软件在用到中断时都要防止这一情况，否则将会产生无穷等待的错误，即CPU等待8087非BUSY，而8087又在等待CPU响应中断。

由于IBM PC的存贮器奇偶错也会产生NMI中断，因此中断处理程序必须先读8255端口以确定有无奇偶错，然后通过执行FNSAVE或FNCLEX指令来查看8087的状态字，识别并处理8087所发生的事故。

第三节　数据格式和指令系统

1. 数据类型及其格式

第一节已经讲过，8087处理的数据类型共有七种，其数据格式如图3-9所示。

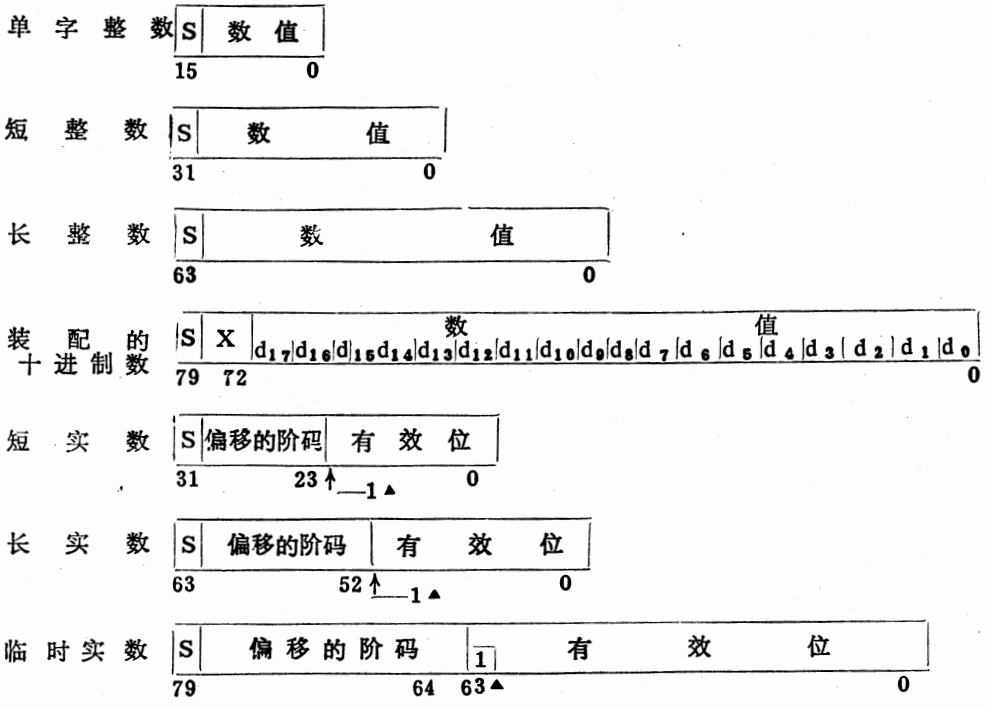


图3-9　数据格式

这七种数据类型可以归纳为三类：

* 二进制整数
* 装配十进制整数
* 二进制实数

下面将分别予以讨论。

（1）二进制整数

除了长度不同（因而可表示的数的范围也不同）之外，三种二进制整数的格式都一样。最左边是符号位S，“0”表示正数，“1”表示负数，负数采用标准的补码表示。单字整数格式与8088可以处理的16位带符号整数的格式完全相同。

（2）十进制整数

十进制整数以装配十进制方式表示，最左面的一个字节用来表示数的正负，其中S为符号位，X无意义，其余九个字节的每个字节均由两个十进制数字“装配”而成。负数以原码形式表示。

（3）实数

8087三种实数的格式均包含有三个字段。有效位（也叫做尾数）用于存放实数的有效数字，偏移阶码表示2的指数值，它决定了小数点在有效数字中的位置，符号位用来表示该数的正负。所以，实数往往也叫做浮点数。

8087的实数通常都以规格化的形式来表示，即非“0”实数的有效位的最高位总是1。也就是说，实数的有效位均呈如下形式：

1▲fff…ff

其中▲表示假想的二进制小数点，小数的位数依据实数的类型当定，短实数时为23位，长实数时为52位，临时实数为63位。注意，短实数和长实数中的整数“1”是隐含的，实际并不存在，该整数位“1”只是在临时实数格式才存在。要使实数保持规格化的表示形式，需要依靠调整阶码的值来保证做到这一点。这是8087在进行实数的运算处理时自动完成的。

实数的阶码采用所谓的“偏移指数”形式表示，即真实的指数（以“2”为底）都被加上一个常数（偏移量）来表示。例如，短实数的偏移量是127（十六进制的7FH），长实数为1023（3FFH），临时实数为16383（3FFFH）。

表3-5是实数178.125的几种不同的表示法，读者可以从中看出8087的实数格式是如何形成的。

表3-5　实数的表示法

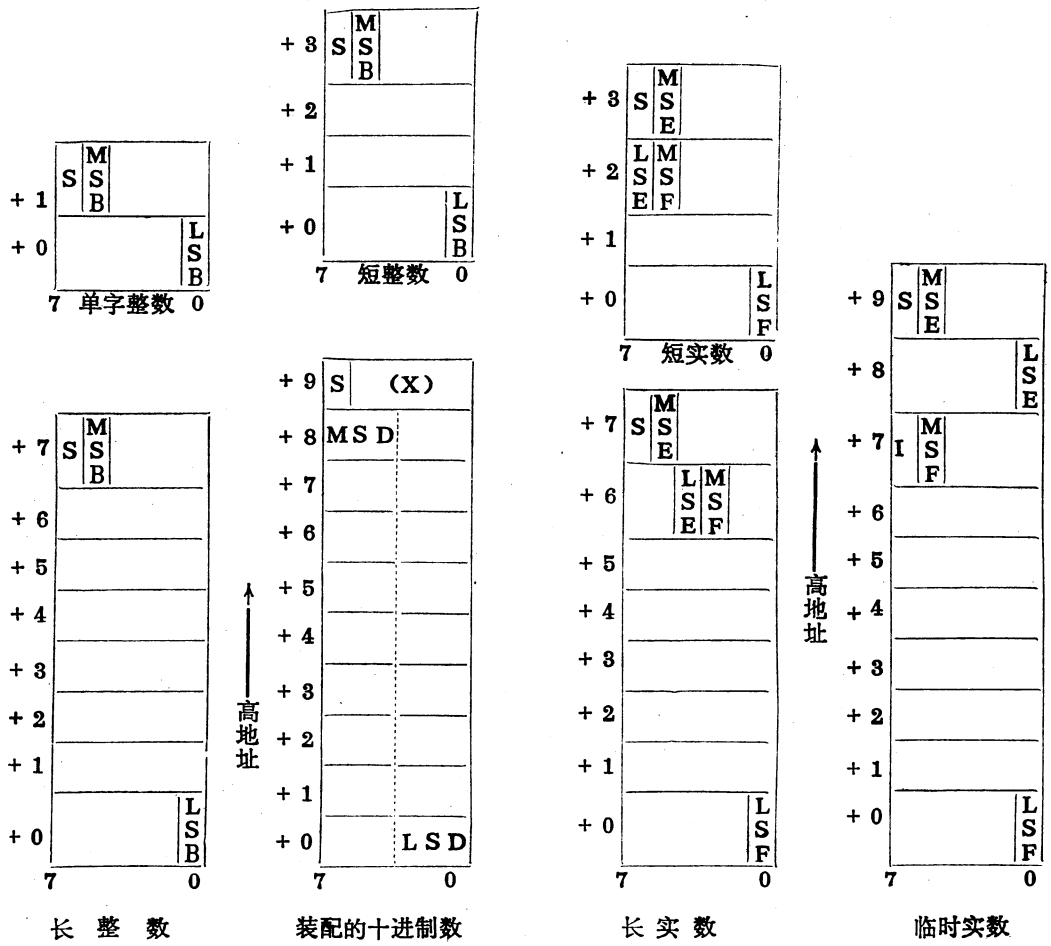
实数表示	数值		
原始十进制数	178.125		
科学计算表示法（十进制）	1▲78125E2		
科学计算表示法（二进制）	1▲0110010001E111		
科学计算表示法（二进制偏移阶码）	1▲0110010001E10000110		
8087短实数表示法（规格化形式）	符号	偏移阶码	有效位
	0	10000110	01100100010000000000000 ↑ 1▲（隐含的）

8087的长、短实数格式只存在于主存贮器中，如果它们被装入8087的寄存器，就会被自

动转换为临时实数格式。同样，8087寄存器中的临时实数也能在放回内存去的时候自动转换成长、短实数格式。需要注意的是，临时实数也可以不经转换直接存放到内存中去，或从内动中取入8087，这在某些场合下特别有用。例如，当计算过程中的中间结果太多，使8087的寄存器容纳不下时，就可以把它们暂时存放在主存贮器中。

图3-10说明了8087的七种不同格式的数据是如何存贮在主存贮器中的。从图中可以看出，无论是整数还是实数，有效数字的最低位总是存放在低地址字节单元中，而其高位数（或指数）以及符号位总是顺序地存放在高地址字节单元中。单字整数的存放与8088/8086的16位带符号整数的存放完全一样，因而需要时也可直接用CPU指令来处理。

8088/8086主CPU为8087产生的存贮器操作数的地址，总是该操作数在内存中的最低字节的地址，8087捕获并保存这个地址，然后不断递增所保存的地址，直到存取完毕操作数的所有字节为止。



S：符号位
MSB/LSB：最高/最低有效位
MSD/LSD：最高/最低有效十进制数
（X）：无效位

S：符号位
MSE/LSE：最高/最低有效阶码位
MSF/LSF：最高/最低有效尾数位
I：有效整数位

图3-10 8087数据的存贮形式

2. 数据传送指令

8087共有69条指令，大体分成下列六个功能组：

* 数据传送指令
* 算术运算指令
* 比较指令
* 超越函数指令
* 常数指令
* 处理器控制指令

典型的8087指令一般需要使用一个或两个操作数，进行了规定的运算处理之后产生一个结果。若操作数在指令执行后不被指令所改变，则该操作数称为源操作数（S）；若操作数在指令执行后被得到的结果所取代，则它被称为目标操作数（D）。

8087可以对一个或两个操作数进行运算处理，操作数既可以在指令中明确地指出，也可以在指令中只写出一个或者两个都不写出来。明确写出的操作数叫做“显式操作数”，不明确写出的则称为“隐式操作数”。若两个操作数均为隐式操作数，通常指的就是栈顶单元ST（作为源操作数S）和栈顶的下一单元ST（1）（作为目标操作数D）。若仅使用一个隐式操作数，则指的就是栈顶单元。

例如，加法指令的操作数有如下三种不同的形式：

FADD D, S　　表示D操作数与S操作数作实数相加后，结果取代D操作数。

FADD S　　表示栈顶单元内容与S操作数作实数相加后结果放入栈顶单元。

FADD　　表示栈顶的下一单元内容与栈顶单元内容作实数相加后，结果放入栈顶下一单元。

为了更简洁地进行表达，上述加法指令所允许的三种操作数形式可综合表达为：

FADD /S/D, S

其中，“/”表示“或者”，“//”表示操作数为隐式。

关于对操作数的限制，在讲到算术运算指令时再作说明。在下面的叙述中，凡存贮器操作数均使用标号（名字）给出，而寄存器操作数则使用ST（i）给出。ST（i）表示栈顶寄存器下面的第i个寄存器，当i=0时，ST（0）就是栈顶寄存器，也常用ST表示。

数据传送指令允许在栈寄存器之间或在栈顶和内存之间传送操作数。在8087内部，所有数据都为临时实数格式，而在存贮器中操作数却能以指定的七种数据类型之一存放。因此，数据传送指令会根据需要而自动进行数据格式的转换。下面是8088的九条数据传送指令：

① FLD S　　取实数

取实数指令将源操作数取入堆栈，即堆栈指针减1后把源操作数写入指针所指向的一个寄存器中。源操作数可以是存贮器中的一个实数，也可以是堆栈中的第i个寄存器ST（i）的内容。源操作数如果是长实数或短实数，则自动变为临时实数。

② FST D　　存实数

将栈顶数据传送到目标，目标可以是栈中其它寄存器ST（i），也可以是长实数或短实数类型的存贮区域，此指令不能把临时实数类型存入存贮器。执行此指令后不改变栈指针。

③ FSTP D　　存实数并出栈

将栈顶数据传送到目标，目标可以是栈中其它寄存器ST（i），也可以是长、短实数或者是临时实数格式的存贮区域。然后，栈指针加1，指向下一寄存器。

④ FXCH //D

寄存器交换指令。将栈顶寄存器内容与目标操作数交换，目标只能是寄存器ST（i）。如果目标未指明，则表示与ST（1）交换。

例如，下面的一段程序表示将栈顶下第3个寄存器的内容求平方根。

```
FXCH ST ( 3 )
FSQRT
FXCH ST ( 3 )
```

⑤ FILD S　　取整数

把源操作数入栈并转换成临时实数格式。

⑥ FIST D　　存整数

将栈顶内容舍入为整数，然后将结果存入目标。目标内容可以为单字整数，也可以为短整数格式，但不能为长整数。舍入不影响栈顶内容。

⑦ FISTP D　　存整数并出栈

同上条指令，但被保存的栈顶内容要出栈。目标操作数格式可为任意一种整数类型。

⑧ FBLD S　　取装配十进制数

此指令将源操作数的内容从装配十进制变为临时实数压入堆栈。

⑨ FBSTP D　　存装配十进制数并出栈

将栈顶内容变为装配十进制整数存入目标存贮器单元，然后退出堆栈。操作中要在栈顶非整数值上加0.5，然后舍掉小数点后的值。

3. 算术运算指令和比较指令

8087的算术运算指令在一般的加、减、乘、除之外，又增加了一些新的功能和变化。例如反减、反除、取绝对值、求平方根等。

算术指令的操作数可以在寄存器中或者在存贮器中。但对不同类型的指令有各自的限制。表3-6指出了这些指令各自的限制。

表3-6　　基本算术指令及其操作数

指令操作形式	指令记忆形式	操作数格式 D，S	汇编例子
一般堆栈操作	Fop	{ST (1)，ST}	FADD
寄存器操作	Fop	ST (i)，ST／ST，ST (i)	FSUB ST，ST(3)
寄存器操作并出栈	FopP	ST (i)，ST	FMULP ST(2)，ST
实型数存贮器操作	Fop	{ST，}短实数／{ST，}长实数	FDIV　AZIMUTH
整型数存贮器操作	FIop	{ST，}单字整数／{ST，}短整数	FIDIV　PULSES

在表中，Fop表示8087实数操作指令的一般形式，FopP指操作后出栈的指令，FIop指整

数操作指令。用{ }括住的内容，表示在写指令时不必明确写出，这里为了方便起见，是用来指明操作数D、S的允许范围。值得注意的一点是表中分寄存操作、堆栈操作和存贮器操作。堆栈操作无显式操作数，目标D总是ST(1)，源S总是ST(即ST(0))；寄存器操作的操作数均为显式表示，此时在非出栈操作时，指ST和另一寄存器之间的操作，而在出栈操作时，是指源S为ST、目标为任一寄存器ST(i)之间的操作；存贮器操作只有一个显式的存贮器操作数，作为源S，隐式操作数定义为ST并作为目标D。

① 加法指令

加法指令共有三条。它们的功能是把目标操作数(被加数)和源操作数(加数)相加，结果送入目标。这三条指令的汇编记忆符及其所允许的操作数表示形式如下：

```
* FADD/S/D, S          实数加法
* FADDP D, S           实数加法并出栈
* FIADD S              整数加法
```

各种形式的加法指令及其所完成的功能举例如下：

```
FADD                    ; ST(1)中的内容加上ST中的内容，结果送ST(1)。
FADDP  ST(1), ST        ; ST(1)中的内容加上ST中的内容，ST弹出，结果在新的
                          栈顶中。
FADD real_memory        ; ST中的内容加上内存单元 real_memory中的内容,结果送
                          ST。
FADD ST(i), ST          ; ST(i)中的内容加上ST中的内容，结果送ST(i)。
FADDP ST(i), ST         ; ST(i)中的内容加上ST中的内容，结果送ST(i)，弹
                          出ST，结果在ST(i-1)中。
FADD ST, ST(i)          ; ST中的内容加上ST (i)中的内容，结果送ST中。
FIADD pulses            ; 栈顶内容与存贮单元pulses中的整数相加后送入栈顶。
```

而以下指令的写法都是错误的(参见表3-6)：

```
FADD ST, VECTOR〔SI〕    ; ST应为隐式操作数，不应写出来。
FADD ST(2)              ; 不应省略显式操作数ST。
FADDP ST, ST(1)         ; 对有出栈操作的指令，ST不能为目标，而必须为源。
FIADD ST, ST(1)         ; 整数操作指令，应取存贮器操作数做源操作数，而目标ST
                          应为隐式表示。
```

有关加法指令的操作以及对操作数的限制，可推广到减法、乘法和除法等指令中去。

② 减法指令

三条减法指令的功能都是从目标操作数中减去源操作数把差送入目标，而三条反减指令则是从源操作数中减去目标操作数，把差数放入目标中。

```
* FSUB/S/D, S          实数减法
* FSUBP D, S           实数减法且减数出栈
* FISUB  S             整数减法
* FSUBR/S/D, S         实数反减
* FSUBRP D, S          实数反减且被减数出栈
* FISUBR  S            整数反减
```

③ 乘法

乘法指令把源操作数和目标操作数相乘，积送目标。

* FMUL /S/D, S　　实数乘法
* FMULP D, S　　实数乘法且乘数出栈
* FIMUL S　　整数乘法

例如，FMUL ST, ST(0)可求出栈顶值的平方。

④ 除法

除法指令把目标操作数除以源操作数（反除指令相反），然后把商存入目标。

* FDIV /S/D, S　　实数除法
* FDIVP D, S　　实数除法且除数出栈
* FIDIV S　　整数除法
* FDIVR /S/D, S　　实数反除
* FDIVRP D, S　　实数反除且被除数出栈
* FIDIVR S　　整数反除

⑤ 其它算术指令

* FSQRT　　求平方根

求栈顶平方根，结果取代栈顶的内容。

* FABS　　取绝对值

将栈顶内容的符号位置为“0”，表示正数。

* FCHS　　改变符号

栈顶内容的符号位取反。

* FSCALE　　换算

将ST(1)中的值求整，再将此值加到栈顶数值的指数部分上。即ST←ST·$2^{ST(1)}$。此条指令提供了对2的整数幂的快速乘法或除法。

* FPREM　　求部分余数

栈顶被其下一个栈单元作模除，ST(1)中是模数。余数符号与原被除数的符号相同，求出的余数在ST中。

FPREM的一个重要用途是减小周期性超越函数的自变量，使其能落在函数指令允许的范围内。例如，求正切函数值时要求幅角小于π/4，则使用π/4作为模数，FPREM指令就可以使其幅角落入正切函数指令所要求的范围之内。

* FRNDINT　　舍入成整数

将栈顶内的实数舍入成整数。

例：ST中是十进制的155.625，如果控制字中的RC字段置成向下舍入，那末指令将其变为155；如为向上舍入，指令将其变为156。

* FXTRACT　　分解指数和尾数

将栈顶中的实数分解为两个数，分别表示该数的指数和尾数的实际值。指数部分取代原操作数，尾数压入新的栈顶。它们都表示成临时实数。ST中的符号位同原栈顶实数，ST(1)的符号位取决于原栈顶实数的指数（是否大于或小于偏移值）。

例如，设ST中有一数，其指数部分为＋4（即指数字段的内容为4003H），尾数部分为

1 ▲11010…0B，符号位为正。经过FXTRACT指令处理后，新的栈顶中指数部分为3FFFH（真0），尾数部分为1▲11010…0B（尾数不变），符号为正（同原符号位）；ST（1）中的指数部分为4001H(+2)，尾数部分为1▲00…0B，符号为正（原指数为正）。

⑥ 比较指令

比较指令常用于比较栈顶单元中的数和另一操作数之间的大小关系，然后在状态字条件码中表示出比较的结果。为了在程序中使用比较的结果，可通过FSTSW指令把状态字送入存贮器以便后续的指令对它进行检查。

比较指令所置的条件码反映的结果如表3-7所示。

表3-7　比较指令所置的条件码

C_3	C_0	结　　果
0	0	ST>S
0	1	ST<S
1	0	ST=S
1	1	两数无法比较

* FCOM /S　　实数比较

把栈顶数与源操作数比较。

* FCOMP /S　　实数比较并出栈

把栈顶数与源操作数比较，栈顶数退出堆栈。

* FCOMPP　　实数比较并两次出栈

栈顶数与ST（1）比较，两者均退出堆栈。

* FICOM　S　　整数比较

把单字整数或短整数的源操作数变为临时实数，并把栈顶与其比较。

* FICOMP　　整数比较并出栈

同上，并作出栈处理。

* FTST　　测试

将栈顶中的数与0比较，相当于源操作数为0的比较指令。

* FXAM　　检验

检验栈顶中数据的状态，它置条件码的情况见表3-8。

4. 超越函数指令和常数指令

这类指令对三角函数、反三角函数、双曲函数、反双曲函数及对数和指数函数进行处理，常数指令则用来生成多种常用的常数。每种函数的操作数都有范围限制，并要求是规范化的，这是程序员必须注意的。此外，所有的函数指令都对栈顶的一个或两个单元进行操作，结果也在栈中。

① FPTAN　　部分正切

表3-3　FXAM执行后的条件码

条件码				说明
C_3	C_2	C_1	C_0	
0	0	0	0	非规格化正数
0	0	0	1	正的非数值数据
0	0	1	0	非规格化负数
0	0	1	1	负的非数值数据
0	1	0	0	正规格化数
0	1	0	1	正无穷大
0	1	1	0	负规格化数
0	1	1	1	负无穷大
1	0	0	0	正　零
1	0	0	1	空
1	0	1	0	负　零
1	0	1	1	空
1	1	0	0	不可规格化正数
1	1	0	1	空
1	1	1	0	不可规格化负数
1	1	1	1	空

用于计算函数Y/X=TAN（θ），θ从栈顶取出，范围是$0<\theta<\frac{\pi}{4}$。操作结果是一个比值，Y取代θ，X压入堆栈，成为新的栈顶。

② FPATAN　　部分反正切

用于计算 θ=ARCTAN（Y/X）。X从栈顶取得，Y从ST（1）中取得。Y和X必须符合：$0<Y<X<\infty$。该指令将X退出堆栈，把结果取代Y值。

其它三角函数如SIN、COS、ARCSIN、ARCCOS都可以从上述两个函数推导出来。

③ F2XM1　　2^x-1

用于计算函数$Y=2^x-1$。X取自栈顶，它必须在$0\leqslant X\leqslant 0.5$范围内。结果Y取代X。例如求$Y=2^x$，只要将计算结果加1。

④ FYL2X　　$Y\cdot\log_2 X$

计算$Z=Y\cdot\log_2 X$。X取自栈顶，Y取自ST(1)，操作数在$0<X<\infty$，$-\infty<Y<+\infty$范围内。

其它底的X次幂的计算公式如下：

$$10^x=2^{x\cdot\log_2 10}$$

$$e^x=2^{x\cdot\log_2 e}$$

$$Y^x=2^{x\cdot\log_2 y}$$

下面将要介绍的常数指令可以用来装入常数$\log_2 10$和$\log_2 e$，而指令FYL2X可用来计

算X·$\log_2$Y。可见8087指令可以很容易地计算出以任意数为底的X次幂。

⑤ FYL2XP1　　Y·$\log_2$(X+1)

计算Z=Y·$\log_2$(1+X)。X取自栈顶，并且必须满足：$0<|X|<(1-\sqrt{2}/2)$；Y取自ST(1)，必须满足$-\infty<Y<+\infty$。该指令使X退出堆栈，用结果Z取代Y。本指令可用来计算很接近于1的数的对数。

⑥ 常数指令

常数指令一共有七条，每条常数指令都把一个常数压入堆栈。常数值都是临时实数的格式，其精度可达19位十进制数。

* FLDZ　　装+0.0
* FLD1　　装+1.0
* FLDPI　　装π
* FLDL2T　　装$\log_2 10$
* FLDL2E　　装$\log_2 e$
* FLDLG2　　装$\log_{10} 2$
* FLDLN2　　装$\log_e 2$

5. 处理器控制指令

处理器控制指令用于控制和检验8087的内部状态，如：初始化，异常事故处理和任务转换等。有些处理器控制指令有两种助记形式，分别用“/”分隔。

① FINIT/FNINIT　　处理器初始化

本指令的操作和硬件RESET信号的功能等效。

② FDIS/IFNDISI　　禁止中断

将控制字中中断屏蔽位置“1”，以防止8087发出中断申请。

③ FENI/FNENI　　允许中断

清除控制字中断屏蔽位，允许8087形成一个中断申请。

④ FLDCW　S　　装入控制字

用源操作数定义的字取代协处理器现行的控制字。

⑤ FSTCW/FNSTCW　D　　保存控制字

把现行控制字写入目标定义的存贮器单元。

⑥ FSTSW/FNSTSW　D　　保存状态字

将8087状态字的值写入存贮器中的目标操作数。该指令可跟一条比较指令或FPREM指令作分支转移或查询8087是否忙；或者在不使用中断的环境中引出事故处理程序。

⑦ FCLEX/FNCLEX　　清除事故

清除状态字中所有事故标志、中断申请标志和“忙”标志，使8087的INT和BUSY线变为无效，它常常在返回被中断的计算程序之前使用，以免多次响应中断。

⑧ FSAVE/FNSAVE　D　　保护状态

本指令把全部8087的状态（环境再加上寄存器栈）都存入目标定义的操作数中。共需要94个字节的存贮区域，它保留了8087运行任一指令后的状态。

⑨ FRSTOR　S　　恢复状态

把源操作数所指出的94个字节存贮区内容装入8087。
图3-11是上面两条指令所定义的存贮区的分配。

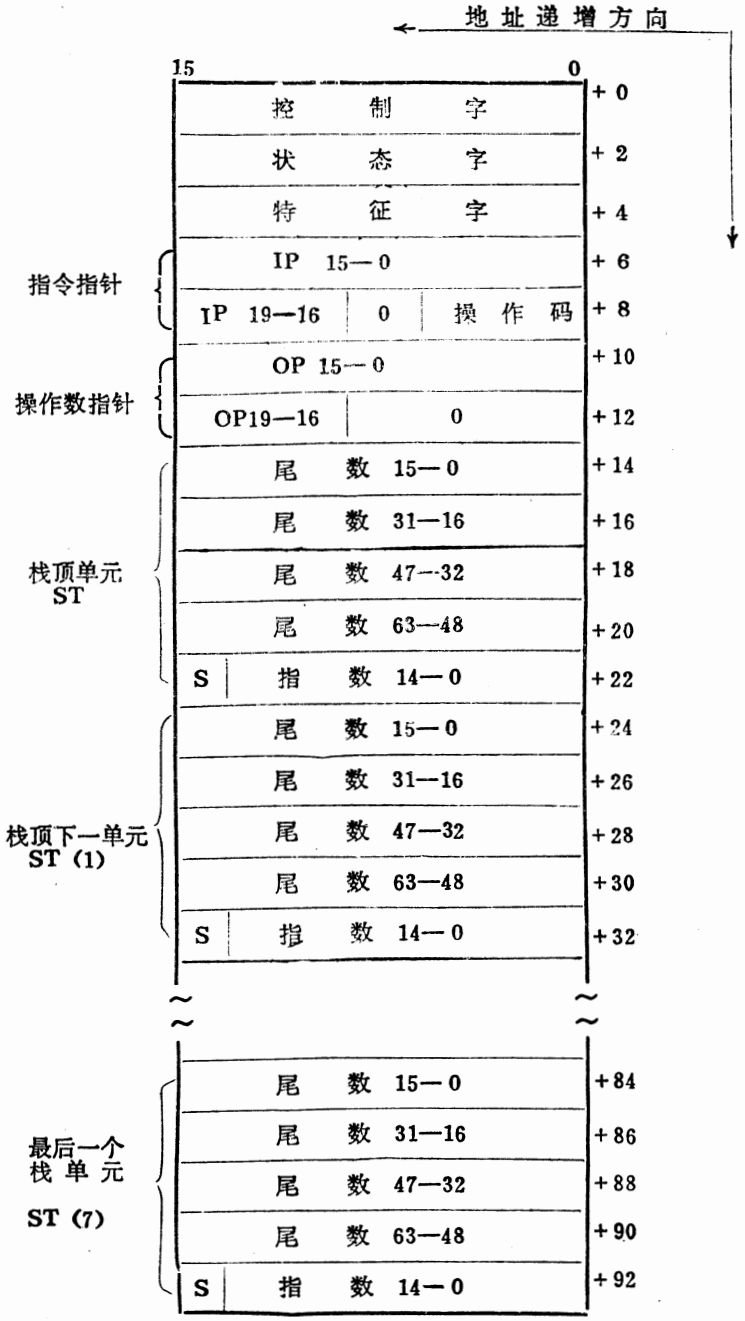


图3-11 FSAVE／FRSTOR的存贮器分配

⑩ FSTENV/FNSTENV D 保存环境

把8087的控制字、状态字、特征字以及异常事故指针写入目标操作数所指出的存贮器

单元中，一般放入CPU栈中。

⑪FLDENV　S　　装入环境

从源操作数所指示的存贮区中恢复8087环境现场。

上述两条指令的存贮分配可参见图3-12。

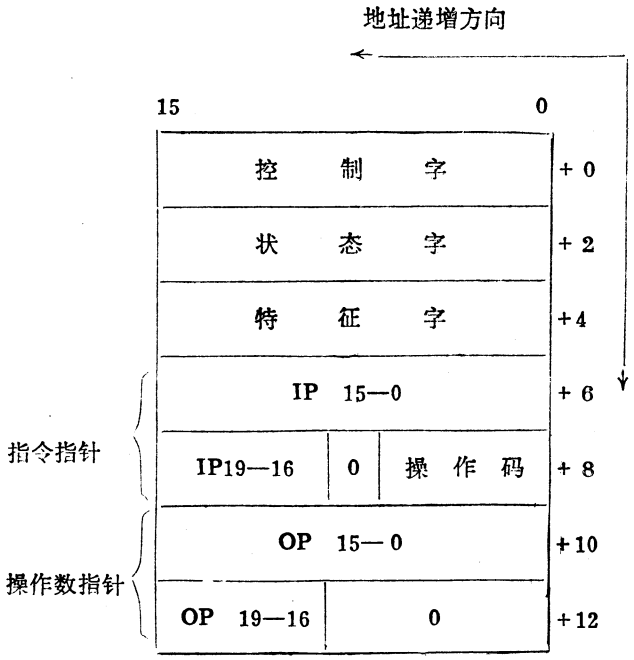


图3-12　FSTENV／FLDENV存贮器分配

⑫ FINCSTP　　堆栈指针递增

将状态字中栈顶指针（ST）值加1。不改变特征字或寄存器内容，不进行数据传送，若ST＝7，本指令执行后使ST＝0。

⑬ FDECSTP　　堆栈指针递减

状态字中栈顶指针（ST）值减1。若ST＝0，指令执行后ST＝7。

⑭ FFREE　D　　寄存器空闲

把目标寄存器的特征变为空，寄存器内容不受影响。

⑮ FNOP　　空操作

把栈顶存入栈顶（相当于FST ST，ST(0)），实际为空操作。

⑯ FWAIT　　CPU的WAIT

本指令不是8087指令，它只是CPU的WAIT指令的另一种助记符而已。

下面的一段程序说明怎样用FWAIT迫使CPU等待8087。

```
FNSTSW    STATUS          ；保存8087的状态字
FWAIT                     ；CPU等待8087把FNSTSW执行完
MOV    AX，STATUS         ；CPU取入8087的状态字
```

第四节　8087汇编语言程序设计举例

本节向读者介绍12个8087程序设计的实例，这些例子都是应用上一章介绍的宏汇编语言写成的，其中凡以“F”或“f”开头的指令助记符均为8087指令。当然，这些程序必须经过宏汇编程序处理后产生目标模块，再通过连接程序（LINK）生成可执行程序，然后才能装入机内运行。

Microsoft公司开发的宏汇编程序MASM（1.0版）不能处理8087指令，只有1.20版以后的宏汇编才能识别8087指令。如果使用Microware公司的宏汇编程序，则必须分两步进行处理。首先用87MACRO对程序进行预处理，把其中的8087指令转换成为ESC指令或目标代码，然后再用MASM87进行汇编处理，产生目标文件。本节所举的例子都是使用Microsoft公司的MASM（1.25版）在机器上调试而成的。

MASM（1.25版）为用户使用8087指令提供了两个程序开关：

/R　表示机器上已安装了8087芯片，程序中的8087指令助记符将直接翻译成8087指令码

/E　表示机器上未安装8087芯片，程序中的8087指令只能仿真执行，因此在连接时必须使用包含有8087仿真程序的程序库

使用汇编语言编写的8087程序，在形式上可以是能够独立运行的一个完整的计算程序，但在更多的情况下它常常以子程序的形式出现，供BASIC、FORTRAN、PASCAL等高级语言调用。下面一些例子，着重向读者介绍呈子程序形式的8087汇编语言程序。尽管使用不同高级语言调用汇编语言子程序时，参数的传递规程并不统一，但汇编子程序的编写原则大体上还是相同的。因此，本节主要以BASIC语言和FORTRAN语言作为调用子程序的宿主语言，至于其它高级语言调用汇编子程序的细则，可参阅有关手册。

例3-1　数据求和

程序功能：将存贮器中的十个数（1.0，2.0，3.0，…，10.0）相加，结果放入存贮器指定单元之中。

程序：

```
STAT_DATA SEGMENT PARA PUBLIC 'DATA'
ARRAY     DD      1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0
N         DW      10D
DSUM      DD      ?
STAT_DATA ENDS
STAT_STACK SEGMENT PARA STACK 'STACK'
           DW      200H    DUP(0)
STAT_STACK ENDS
STAT_CODE SEGMENT PARA PUBLIC 'CODE'
BEGIN     PROC    FAR
ASSUME CS:STAT_CODE,DS:STAT_DATA,SS:STAT_STACK,ES:NOTHING
          MOV     AX,STAT_DATA
          MOV     DS,AX
```

```
            MOV     AX,STAT_STACK
            MOV     SS,AX
            MOV     CX,N
            FINIT
            FLDZ
START:      MOV     SI,CX
            DEC     SI
            SHL     SI,1
            SHL     SI,1
            FADD    ARRAY[SI]
            LOOP    START
            FST     DSUM
            INT     3
BEGIN       ENDP
STAT_CODE   ENDS
            END BEGIN
```

说明：

（1）本例中十个原始数据放在数组ARRAY中，相加后的结果放入DSUM单元中，它们均为8087的短实数格式，每个数据需占用四个字节。经过汇编后，十个原始数据的内码（十六进制）为：

00 00 80 3F	00 00 00 40
00 00 40 40	00 00 80 40
00 00 A0 40	00 00 C0 40
00 00 E0 40	00 00 00 41
00 00 10 41	00 00 20 41

它们均为前面介绍的IEEE浮点数格式，读者不妨自行验证一下。

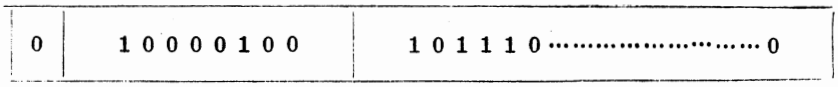
```
-u0,27
0966:0000 B82309          MOV     AX,0923
0966:0003 8ED8            MOV     DS,AX
0966:0005 B82609          MOV     AX,0926
0966:0008 8ED0            MOV     SS,AX
0966:000A 8B0E2300        MOV     CX,[0028]
0966:000E 9B              WAIT
0966:000F DBE3            FINIT
0966:0011 9B              WAIT
0966:0012 D9EE            FLDZ
0966:0014 8BF1            MOV     SI,CX
0966:0016 4E              DEC     SI
0966:0017 D1E6            SHL     SI,1
0966:0019 D1E6            SHL     SI,1
0966:001B 9B              WAIT
0966:001C D8840000        FADD    DWORD PTR [SI+0000]
0966:0020 E2F2            LOOP    0014
0966:0022 9B              WAIT
0966:0023 D9162A00        FST     DWORD PTR [002A]
0966:0027 CC              INT     3
```

（2）程序中使用了四条8087指令：FINIT，FLDZ，FADD和FST。经过汇编）使用/R开关）和连接之后，生成一个可运行文件(.EXE文件)。如果使用DEBUG（2.00版）程序将它装入内存，再把它反汇编出来，其结果如上页所示。可以看出，为了8088CPU和8087之间的同步，汇编程序在每条8087指令之前都自动插入了一条WAIT指令。

（3）程序中的FINIT指令对8087进行初始化，FLDZ把8087栈顶清0，FADD用来把存贮器操作数加到8087栈顶单元，常数N（10）送入寄存器CX，用来控制循环次数，相加完毕后结果由栈顶单元送入内存DSUM中保存。INT 3是为了用DEBUG调试这个程序而使用的。通过DEBUG可以看到这个程序的运行结果（即DSUM单元中的内码，按地址由大到小排列）为：

425C0000H

把它们表示为8087的短实数格式：



其数值为：

$2^5 \times (1.10111B) = 110111B = 55$

可见结果正确。

例3-2　求一组数的平均值和均方差

程序功能：

本程序用来计算n个数的平均值A和均方差D，即

$$A = (V_1 + V_2 + \cdots + V_n) / N$$

$$D = \sqrt{[(V_1 - A)^2 + (V_2 - A)^2 + \cdots + (V_n - A)^2] / N}$$

程序：

```
STAT_DATA SEGMENT PARA PUBLIC 'DATA'
AVERAGE           DD    ?
STANDARD_DEV      DD    ?
VECTOR_SIZE       DW    9
VECTOR            DD    1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0
BENCH_NUMBER      DW    1000D
STAT_DATA ENDS

STAT_STACK SEGMENT PARA STACK 'STACK'
                  DW       200H
STAT_STACK ENDS

STAT_CODE SEGMENT PARA PUBLIC 'CODE'
BEGIN    PROC     FAR
ASSUME CS:STAT_CODE,DS:STAT_DATA,SS:STAT_STACK,ES:NOTHING
         MOV      AX,STAT_DATA
         MOV      DS,AX
```

```
          MOV       AX,STAT_STACK
          MOV       SS,AX
          MOV       CX,BENCH_NUMBER
          FINIT                           ;initialize 8087
BENCH_LOOP:                               ;start of benchmark loop
          PUSH      CX                    ;save CX for use later
          FLDZ                            ;0.0-->ST0
          MOV       CX,VECTOR_SIZE        ;9-->CX
START:    MOV       SI,CX                 ;SI=SI-1,SI from 8 to 0
          DEC       SI                    ;while CX from 9 to 1
          SHL       SI,1
          SHL       SI,1                  ;SI=SI*4
          FLD       VECTOR[SI]            ;VEC[SI]->ST0,0->ST1
          FADDP     ST(1),ST              ;ST1=ST1+ST0, ST1->ST0
          LOOP      START                 ;decrement CX,go back
          FIDIV     VECTOR_SIZE           ;ST0/VEC_SIZ->ST0
          FST       AVERAGE               ;ST0-->AVE
          FLDZ                            ;acc=0-->ST0,AVE->ST1
          MOV       CX,VECTOR_SIZE        ;9-->CX
MAIN_LOOP:
          MOV       SI,CX
          DEC       SI                    ;SI from 8 to 0
          SHL       SI,1                  ;while CX from 9 to 1
          SHL       SI,1                  ;SI=SI*4
          FLD       VECTOR[SI]            ;ST0=VEC[SI],ST1=acc
                                          ;ST2=AVE
          FSUB      ST,ST(2)
          FLD       ST                    ;AVE-VEC[SI]->ST0,ST1
          FMULP     ST(1),ST              ;ST0=(VEC[SI]-AVE)**2
                                          ;ST1=acc,ST2=AVE
          FADDP     ST(1),ST              ;ST1=ST1+ST0,then pop
          LOOP      MAIN_LOOP             ;loop 9 times
          FIDIV     VECTOR_SIZE           ;ST0=ST0/VEC_SIZ
          FSQRT                           ;(ST0)**.5-->ST0
          FSTP      STANDARD_DEV          ;save STA_DEV
          FSTP      AVERAGE               ;save AVE
          POP       CX
          LOOP      BENCH_LOOP
          INT       3                     ;jump to DEBUG
BEGIN     ENDP
STAT_CODE ENDS
          END BEGIN
```

说明：

（1）数据段中的VECTOR是原始数据组成的数组，VECTOR_SIZE为数组元素的数目，AVERAGE用来存放平均值，STANDARD_DEV存放均方差，BENCH_NUMBER为程序的重复遍数。

（2）程序大体分三部分：第1部分（从BEGIN开始）用来完成一些准备工作，第2部

分（从START开始）计算出平均值，第3部分（从MAIN_LOOP开始）用来完成均方差的计算。

（3）本例主要用来说明8087寄存器堆栈的使用技巧。图3-13是第1趟进入MAIN LOOP循环后栈中各寄存器内容的变化，读者由此不难推测第2趟、第3趟以及此后各次循环中栈寄存器内容的变化情况。当循环（MAIN_LOOP）结束后，把栈顶内容除以数组元素个数，再开平方便可得到均方差。

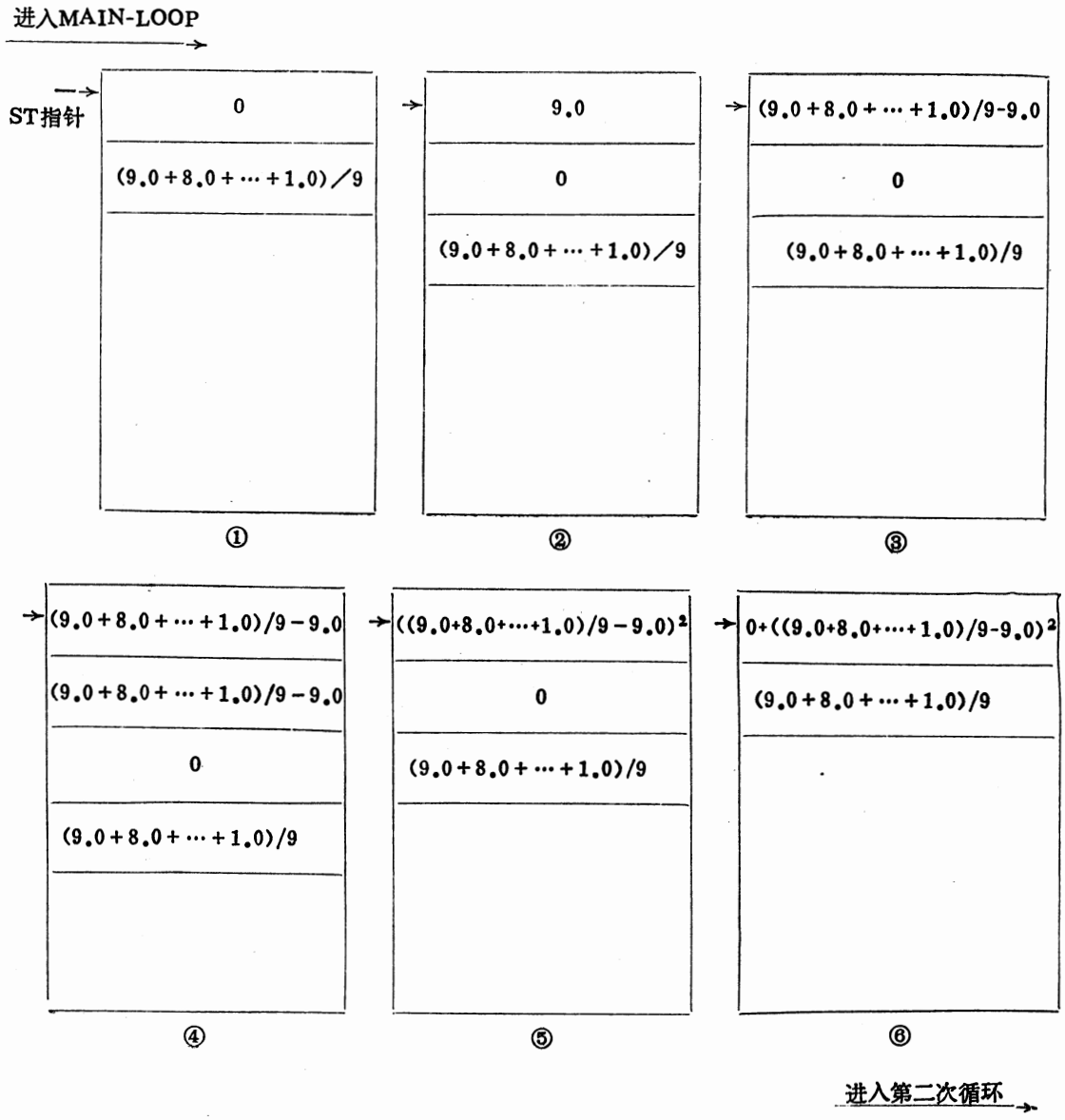


图3-13　例3-2求均方差时第一次循环中栈寄存器内数的变化

（4）程序执行完毕后，通过DEBUG可以从内存单元中查出所得到的计算结果为(十六

进制内码，IEEE短实数格式）：

A = 40A00000H

D = 40253F4EH

请读者自行验证其正确性。

例3-3　数据求和子程序

程序功能：与例3-1一样，用来计算一组数据之和。但本例是一个子程序，仅 供BASIC程序调用。

程序：

```
;SUBROUTINE SUM(ARRAY,N,DSUM)
;ARRAY IS A SINGLE PRECISION ARRAY OF LENGTH N
;N IS INTEGER
;DSUM IS DOUBLE PRECISION

        PUBLIC      SUM
CSEG    SEGMENT     'CODE'
        ASSUME      CS:CSEG
SUM     PROC    FAR
        PUSH    BP
        MOV     BP,SP
        MOV     BX,[BP]+10          ;BX=ADDR(ARRAY)
        MOV     SI,[BP]+8           ;SI=ADDR(N)
        MOV     CX,[SI]             ;CX=N
;now all set up to go
        FLDZ                        ;initialize st=0
        CMP     CX,0H
        JLE     DONE
;the next 3 instructions do all the hard work
ADD_LOOP:
        FADD    DWORD PTR[BX]
        ADD     BX,4
        LOOP    ADD_LOOP
DONE:
;now file answer back in dsum
        MOV     DI,[BP]+6
        FSTP    QWORD PTR[DI]       ;qword=>double precision
        POP     BP
        FWAIT                       ;be sure 8087 is done
        RET     6
SUM     ENDP
CSEG    ENDS
        END
```

说明：

（1）本例是一个可以由解释BASIC或编译BASIC程序调用的汇编语言子程序，其名为SUM，有三个参数：ARRAY表示数组，其中存放着欲进行累计求和的若干数据；N是数 组

中元素的个数；DSUM为和数，它是双精度实数。本书上册第三章已经介绍过，BASIC语言用CALL SUM（ARRAY(0)，N，DSUM）调用这个汇编子程序时，参数的传递是通过堆栈来进行的。图3-14给出了在执行CALL语句后，当控制转移到汇编子程序时，堆栈中内容的变化情况。

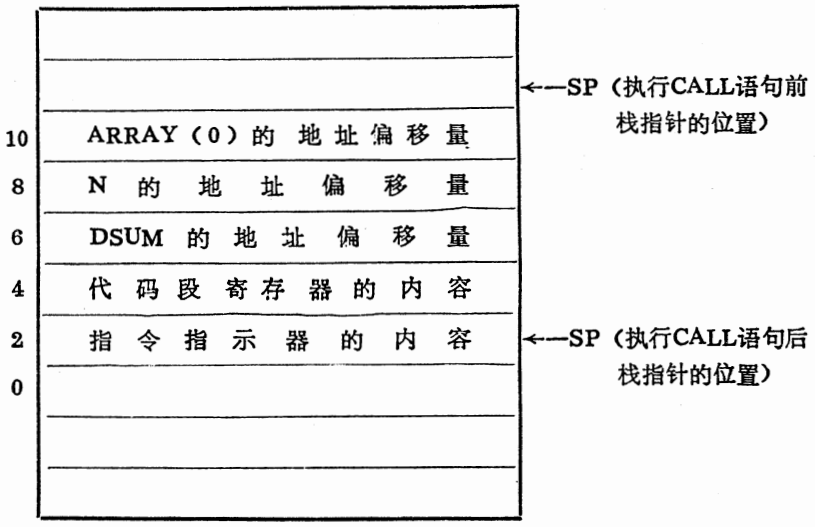


图3-14 BASIC语言程序调用汇编子程序时参数的传递

（2）进入汇编子程序后，首先需要取得各参数的地址。为此，先把BP寄存器保护入栈，再把栈指针SP的内容送入BP作为基地址。于是子程序有n个参数时，第i个参数（从左向右排序）的地址就在存贮单元

（BP）＋6＋2＊（n－i）

中，例如第1个参数的地址存贮在（BP）＋10单元中。

（3）汇编子程序的最后一条指令必须是

RET 2＊n

其中n为参数的个数。只有这样才能在返回BASIC程序后使堆栈恢复为调用前的状态。

（4）使用解释BASIC调用汇编子程序的方法和步骤已在上册中作了介绍，下面是调用SUM子程序的一个示范程序：

```
10      DIM ARRAY!(9)
20      DEF SEG=&H7F94
30      BLOAD "CONVREAL.SAV",0
40      SRMSIEEE=&H10:SRIEEEMS=0:LRIEEEMS=&H20
45      N%=10
50      FOR I=0 TO 9
60      ARRAY!(I)=I
70      CALL SRMSIEEE(ARRAY!(I))
80      NEXT I
90      DEF SEG=&H7FA0
95      SUM=0:DSUM#=0
```

```
100       BLOAD "SUM.SAV",0
110       CALL SUM(ARRAY!(0),N%,DSUM#)
120       DEF SEG=&H7F94
130       FOR I=0 TO 9
140       CALL SRIEEEMS(ARRAY!(I))
150       PRINT ARRAY!(I)
160       NEXT I
170       CALL LRIEEEMS(DSUM#)
180       PRINT DSUM#
190       END
```

其中，CONVREAL.SAV是一个格式转换程序。因为使用的解释BASIC语言与8087不兼容，它的实数表示法采用Microsoft格式，与8087的IEEE格式不一致，因此调用SUM前和返回BASIC后都必须进行实数格式的转换。调用程序用到了转换程序中的三个过程：

* SRMSIEEE　　短实数转换Microsoft格式→IEEE格式
* SRIEEEMS　　短实数转换IEEE格式→Microsoft格式
* LRIEEEMS　　长实数转换IEEE格式→Microsoft格式

程序SUM.SAV就是求和子程序，是本例的主体。CONVREAL.SAV和SUM.SAV都用不同的段定义，通过BLOAD语句装入。装入前它们必须转换成BASIC可装入的二进制内存映象文件，步骤如下：

① 用MASM（1.25版）进行汇编，得到.OBJ文件。

② 分别在高区连接得到.EXE文件，连接时需设立.MAP文件，以便得到各个过程的起始偏移量以及程序长度。上面BASIC调用程序40句中的对SRMSIEEE等变量的十六进制赋值，就是由.MAP文件中得到的。

③ 用DEBUG装入BASIC.COM文件。

④ 用“R”命令显示寄存器的值，记下CS、SS、IP和SP的内容。

⑤ 打入“N SUM.EXE”（“N CONVREAL.EXE”）和“L”命令，装入.EXE文件。

⑥ 用“R”命令显示各寄存器的值，记下CS和IP的新内容。

⑦ 用“R”命令把SS和SP寄存器恢复成第④步前的值。

⑧ 打入“G=CS：IP”命令，在DEBUG下起动BASIC运行。

⑨ 执行语句：

DEF SEG=第⑥步CS的值

再执行语句：

BSAVE “SUM.SAV”，第⑥步IP的值，.MAP文件中的长度

（BSAVE “CONVREAL.SAV”，第⑥步IP的值，.MAP文件中的长度）

⑩ 用SYSTEM和Q分别退出BASIC和DEBUG。

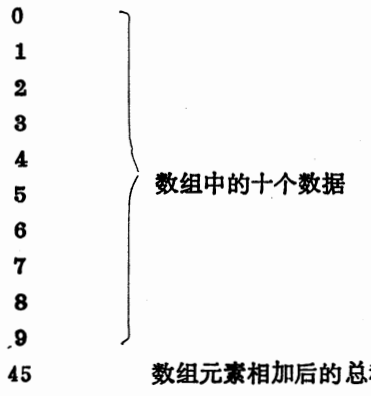
（5）使用编译BASIC调用汇编语言子程序的方法要比解释BASIC简单得多，就本例而言，大体步骤如下：

① 略去解释BASIC程序中的20，30，40，90，100，120句，共6条语句，然后把它编译成目标模块。

② 对SUM.ASM和CONVREAL.ASM进行汇编，生成目标模块。

③ 对上述三个目标模块进行连接，生成一个可执行文件。

（6）本例用解释BASIC或编译BASIC程序调用后运行结果为：



（7）关于实数格式转换程序，它纯系8088指令编制而成，限于篇幅，这里就不作介绍了。

例3-4　数组相加

程序功能：将一维数组A、B中的元素逐个对应相加，相加的结果送入数组C。

程序：

```
';SUBROUTINE VADD(A,B,C,N)
;         A,B,C ARE SINGLE PRECISION ARRAYS
;         N IS INTEGER
;
          PUBLIC  VADD
CSEG      SEGMENT 'CODE'
          ASSUME  CS:CSEG
VADD      PROC    FAR
          PUSH    BP
          MOV     BP SP
          MOV     SI,[BP]+6          ;SI=ADDR(N)
          MOV     CX,[SI]            ;CX=N
          MOV     BX,[BP]+12         ;BX=ADDR(A)
          MOV     SI,[BP]+10         ;SI=ADDR(B)
          MOV     DI,[BP]+8          ;DI=ADDR(C
;
;NOW ALL SET UP TO GO
          CMP     CX,0H              ;HOPE N>0
          JLE     DONE
ADD_LOOP:
          FLD     DWORD PTR[BX]      ;LOAD A(I)
          ADD     BX,4               ;READY FOR NEXT A
          FADD    DWORD PTR[SI]      ;ADD B(I)
          ADD     SI,4               ;READY FOR NEXT ELEMENT
```

```
            FSTP    DWORD PTR[DI]     ;C(I)=A(I)+B(I)
            ADD     DI,4              ;READY FOR NEXT C
            LOOP    ADD_LOOP
DONE:
            POP     BP
            FWAIT                     ;BE SURE 8087 IS DONE
            RET     8
VADD        ENDP
CSEG        ENDS
            END
```

说明：

（1）本例的调用程序传递给子程序四个参数。其中A、B、C都是具有相同元素个数的一维数组，N表示三个数组的元素个数。

（2）程序分两部分，第1部分用来取得四个参数的地址，并把它们分别送入有关寄存器，作为数组元素的指针，第2部分则用来完成A+B→C的功能。

（3）本例的主要目的在于说明如何建立多个地址指针。可以发现，在例3-3中，由于只有一个数组A，因而仅需使用一个寄存器BX作为变址寄存器指向A中的各个元素。而本例有三个数组A、B、C，所以需要使用BX、SI、DI分别作为各自的变址寄存器。

（4）如果本例中把指令FADD改为FSUB、FMUL、FDIV，就可以分别完成C=A-B，C=A*B及C=A/B的向量运算。

例3-5　标量与矩阵的加法

程序功能：有两个n×m矩阵A和B，将标量SCALAR分别加到A矩阵的各个元素上，结果送入矩阵B。

程序：

```
;SUBROUTINE SCALADD(A,SCALAR,B,N,M)
;          A,B ARE SINGLE PRECISION N BY M MATRICES
;          SCALAR IS SINGLE PRECISION
;          N,M ARE INTEGERS
           PUBLIC  SCALADD
CSEG       SEGMENT 'CODE'
           ASSUME  CS:CSEG
SCALADD    PROC    FAR
           PUSH    BP
           MOV     BP,SP
           MOV     SI,[BP]+12        ;SI=ADDR(SCALAR)
           FLD     DWORD PTR[SI]     ;PUSH SCALAR ONTO STACK
           MOV     BX,[BP]+6         ;BX=ADDR(M)
           MOV     DX,[BX]           ;DX=# OF COLUMNS
           MOV     SI,[BP]+14        ;SI=ADDR(A)
           MOV     DI,[BP]+10        ;DI=ADDR(B)
           FWAIT
COLUMN_LOOP:
           MOV     BX,[BP]+8         ;BX=ADDR(N)
```

```
        MOV     CX,[BX]             ;CX=COLUMN LENGTH
        MOV     BX,0
ADD_LOOP:
        FLD     DWORD PTR[BX][SI];LOAD A(I,J)
        FADD    ST(0),ST(1)         ;ADD SCALAR
        FSTP    DWORD PTR[BX][DI];B(I,J)=SCALAR+A(I,J)
        ADD     BX,4                ;READY FOR NEXT ELEMENT
        LOOP    ADD_LOOP
;NOW MOVE TO NEXT COLUMN BY ADDING 4*N TO SI AND DI
        MOV     BX,[BP]+8           ;BX=ADDR(N)
        MOV     AX,[BX]             ;AX=COLUMN LENGTH
        SHL     AX,1                ;MULTIPLY AX
        SHL     AX,1                ;BY 4
        ADD     SI,AX
        ADD     DI,AX
;ARE WE DONE YET ?
        DEC     DX
        CMP     DX,0
        JGE     COLUMN_LOOP
        FSTP    ST(0)               ;GET RID OF SCALAR
        POP     BP
        FWAIT
        RET     10
SCALADD ENDP
CSEG    ENDS
        END
```

说明：

（1）本例的主体部分是一个两重循环，它用来完成矩阵的标量加法运算。运算次序是按列进行的，即先对第0列元素进行运算，然后第1列，第2列，…，直到第m－1列为止。

（2）矩阵中的各元素在存贮器中按列排列，BASIC为二维矩阵分配存贮器时就是这样安排的。例如有一个2×3的矩阵A，其中各元素均为短实数格式，假设a(0，0)的起始地址为100，那末各元素的地址分别为：

a(0，0)——100　　a(0，1)——108　　a(0，2)——116

a(1，0)——104　　a(1，1)——112　　a(1，2)——120

例3-6　向量内积

程序功能：这是一个用来求向量内积的子程序。它适用于短实数、长实数两种数据类型，并能控制元素之间距离。因此可以用来实现各种矩阵乘法，如$A \cdot B$、$A \cdot B^T$、$A^T \cdot B$等。

程序：

```
;SUBROUTINE GINP(A,B,TYPEA,TYPEB,SKIPA,SKIPB,N)
;         A,B ARE ADDRESSES OF N-ARRAYS
;         TYPEA,TYPEB,SKIPA,SKIPB,N ARE INTEGERS
;        THERE MUST BE AT LEAST 2 FREE LOCATIONS ON
;        THE 8087 STACK AND AT LEAST 1 FREE BYTES ON
```

```
;       THE MEMORY STACK

        ASSUME  CS:CSEG
GINP    PROC    NEAR
        PUSH    BP
        MOV     BP,SP
;THIS IS A NEAR PROCEDURE. ARGUMENTS BEGIN AT [BP]+4
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        PUSH    DI
        FLDZ                            ;SET RUNNING SUM=0
        MOV     CX,[BP]+4               ;CX=N
        MOV     SI,[BP]+16              ;SI=ADDR(A)
        MOV     DI,[BP]+14              ;DI=ADDR(B)
        MOV     AX,[BP]+10              ;AX=TYPEB
        MUL     WORD PTR[BP]+6          ;AX=TYPEB*SKIPB
        MOV     BX,AX                   ;BX=B ELEMENT DISTANCE
        MOV     AX,[BP]+12              ;AX=TYPEA
        MUL     WORD PTR[BP]+8          ;AX=A ELEMENT DISTANCE
        JCXZ    DONE
GINP_LOOP:
        CMP     WORD PTR[BP]+12,4;IS A SINGLE ?
        JNE     A_DOUBLE
        FLD     DWORD PTR[SI]           ;LOAD SINGLE A(I)
        JMP     MULT_B
A_DOUBLE:
        FLD     QWORD PTR[SI]           ;LOAD DOUBLE A(I)
MULT_B: CMP     WORD PTR[BP]+10,4;IS B SINGLE ?
        JNE     B_DOUBLE
        FMUL    DWORD PTR[DI]           ;MULTIPLY SINGLE B(I)
        JMP     NEXT_ELEMENT
B_DOUBLE:
        FMUL    QWORD PTR[DI]           ;MULTIPLY DOUBLE B(I)
NEXT_ELEMENT:
        FADDP   ST(1),ST                ;SUM=SUM+A(I)*B(I)
        ADD     SI,AX
        ADD     DI,BX
        LOOP    GINP_LOOP
DONE:
        POP     DI
        POP     SI
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        POP     BP
        RET     14
GINP    ENDP
```

说明：

（1）本程序有七个参数，其中A、B是计算内积的两个一维数组，TYPEA、TYPEB分别表示A、B中元素的类型（其值为4表示短实数，为8表示长实数），SKIPA、SKIPB是两个数组中用作运算的元素之间的距离，N为参加运算的元素个数。

（2）注意，本程序是一个NEAR过程，即此程序只能为其它汇编子程序调用而不能为BASIC所直接调用。所以，其参数从〔BP〕＋4开始，而不是通常那样从〔BP〕＋6开始。

（3）由于是NEAR过程，所以进入子程序后首先必须将需要使用的一些寄存器保护入栈，待子程序结束时再恢复它们原先的内容。

（4）本例值得特别注意的是数据的类型可以选择，这种方法很有用处。

例3-7　矩阵乘法

功能：A、B两矩阵相乘，结果送入矩阵C。

程序：

```
ESEG      SEGMENT 'DATA'
          DW        50        DUP(?)
STACK_TOP           EQU THIS WORD
LOCAL_SPACE         DW        20 DUP(?)
ESEG      ENDS
;SUBROUTINE MATMULT(A,B,C,L,M,N)
;         A,B,C ARE SINGLE PRECISION MATRICES
;                 A IS L BY M
;                 B IS M BY N
;                 C IS L BY N
;                 L,M,N ARE INTEGERS
;
          PUBLIC MATMULT
CSEG      SEGMENT 'CODE'
          ASSUME  CS:CSEG,ES:ESEG
FIRST_INST EQU THIS WORD
MATMULT   PROC    FAR
          PUSH    BP
          MOV     BP,SP
;SET UP STACK AREA IN ESEG
          PUSH    ES
          CALL    NEXT
NEXT:     POP     AX
          SUB     AX,(OFFSET NEXT)-(OFFSET FIRST_INST)
          MOV     CL,4
          SHR     AX,CL
          MOV     BX,CS
          ADD     BX,ESEG
          SUB     BX,CSEG
          ADD     AX,BX
          MOV     ES,AX
          MOV     LOCAL_SPACE,SS
```

```
        MOV     LOCAL_SPACE+2, SP
        MOV     AX, ES
        MOV     SS, AX
        MOV     SP, OFFSET STACK_TOP
;
;TO CALL GINP WE MUST PUSH ONTO THE STACK:
;A(I, 0)
;B(0, J)
;4
;4
;L
;1
;M
;
;ON RETURN THE RESULT GOES IN C(I, J)
;
SOME_SPACE      EQU     LOCAL_SPACE+4
ADDRA_HOLD      EQU     SOME_SPACE
ADDRB_HOLD      EQU     ADDRA_HOLD+2
L_HOLD          EQU     ADDRB_HOLD+2
M4_HOLD         EQU     L_HOLD+2
M_HOLD          EQU     M4_HOLD+2
N_HOLD          EQU     M_HOLD+2
        MOV     BX, DS:[BP]+16      ;BX=ADDR(A(0, 0))
        MOV     ADDRA_HOLD, BX
        MOV     SI, DS:[BP]+14      ;SI=ADDR(B(0, 0))
        MOV     ADDRB_HOLD, SI
        MOV     DI, DS:[BP]+12      ;DI=ADDR(C(0, 0))
        MOV     BX, DS:[BP]+10      ;BX=ADDR(L)
        MOV     AX, [BX]
        MOV     L_HOLD, AX          ;L_HOLD HAS L
        MOV     BX, DS:[BP]+8       ;BX=ADDR(M)
        MOV     DX, [BX]            ;DX=M
        MOV     M_HOLD, DX
        MOV     M4_HOLD, DX
        SHL     M4_HOLD, 1          ;GET 4*M
        SHL     M4_HOLD, 1
        MOV     BX, DS:[BP]+6       ;BX=ADDR(N)
        MOV     CX, [BX]            ;CX=N
        MOV     N_HOLD, CX
        MOV     AX, 4               ;SAVE USEFUL 4
        MOV     BX, 1               ;SAVE USEFUL 1
;
COL_LOOP:
        CMP     N_HOLD, 0           ;COL DONE ?
        JE      DONE
        MOV     SI, ADDRA_HOLD
        MOV     CX, L_HOLD
ROW_LOOP:
        PUSH    SI                  ;A(I, 0)
```

```
        PUSH    ADDRB_HOLD          ;B(0,J)
        PUSH    AX                  ;4
        PUSH    AX                  ;4
        PUSH    L_HOLD              ;L
        PUSH    BX                  ;1
        PUSH    DX                  ;M
        CALL    GINP
        FSTP    DWORD PTR[DI]
        ADD     DI,4                ;NEXT C
        ADD     SI,4                ;NEXT A
        LOOP    ROW_LOOP            ;NEXT ROW
        MOV     SI,M4_HOLD          ;SKIP TO NEXT COLUMN
        ADD ADDRB_HOLD,SI           ;NEXT B
        DEC     N_HOLD
        JMP     COL_LOOP

DONE:
        MOV     SP,LOCAL_SPACE+2
        MOV     SS,LOCAL_SPACE
        POP     ES
        POP     BP
        FWAIT
        RET     12
MATMULT ENDP
;SUBROUTINE GINP(A,B,TYPEA,TYPEB,SKIPA,SKIPB,N)
;       A,B ARE ADDRESSES OF N-ARRAYS
;
GINP    PROC    NEAR
        PUSH    BP
        MOV     BP,SP
;THIS IS A NEAR PROCEDURE,ARGUMENTS BEGIN AT [BP]+4
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
        PUSH    SI
        PUSH    DI
        FLDZ                        ;SET RUNNING SUM=0
        MOV     CX,[BP]+4           ;CX=N
        MOV     SI,[BP]+16          ;SI=ADDR(A)
        MOV     DI,[BP]+14          ;DI=ADDR(B)
        MOV     AX,[BP]+10          ;AX=TYPEB
        MUL     WORD PTR[BP]+6      ;AX=TYPEB*SKIPB
        MOV     BX,AX               ;BX=B ELEMENT DISTANCE
        MOV     AX,[BP]+12          ;AX=TYPEA
        MUL     WORD PTR[BP]+8      ;AX=A ELEMENT DISTANCE
        JCXZ    DONE1
GINP_LOOP:
        CMP     WORD PTR[BP]+12,4;IS A SINGLE ?
        JNE     A_DOUBLE
```

```
            FLD     DWORD PTR[SI]    ;LOAD SINGLE A(I)
            JMP     MULT_B
A_DOUBLE:
            FLD     QWORD PTR[SI]    ;LOAD DOUBLE A(I)
MULT_B:     CMP     WORD PTR[BP]+10,4;IS B SINGLE ?
            JNE     B_DOUBLE
            FMUL    DWORD PTR[DI]     MULTIPLY SINGLE B(I)
            JMP     NEXT_ELEMENT
B_DOUBLE:
            FMUL    QWORD PTR[DI]    ;MULTIPLY DOUBLE B(I)
NEXT_ELEMENT:
            FADDP   ST(1),ST         ;SUM=SUM+A(I)*B(I)
            ADD     SI,AX
            ADD     DI,BX
            LOOP    GINP_LOOP
DONE1:
            POP     DI
            POP     SI
            POP     DX
            POP     CX
            POP     BX
            POP     AX
            POP     BP
            RET     14
GINP        ENDP
CSEG        ENDS
            END
```

说明：

（1）矩阵乘法子程序MATMULT的参数有六个，它们的含义是：A、B、C分别为L×M，M×N和L×N的矩阵。

（2）由于矩阵乘法需要调用例3-6介绍的求向量内积的子程序GINP，调用时要传递七个参数，而且后者还需要使用七个栈单元来保护寄存器的内容。为了运行可靠起见，MATMULT自行在附加数据段中开辟了一个50个字的局部栈区，以供调用GINP使用，而不再使用BASIC程序的堆栈。

（3）为了汇编后得到的目标模块可以任意地重定位，即在解释BASIC调用MATMULT时可以装入内存任意位置而都能正确运行，程序中必须根据CSEG、ESEG以及运行时CS寄存器的值来计算出局部栈区及栈顶指针的实际位置，程序的第1段就是用来完成这个任务的。计算栈区位置的公式是：

$$16\times ESEG-16\times CSEG+(16\times DEF\text{-}SEG+OFFSET)$$

其中DEF_SEG和OFFSET 是解释BASIC装入该子程序时所定义的段地址和段内偏移量。

（4）MATMULT从BASIC栈中取得A，B，C，L，M，N等参数的地址，按GINP的要求，送入局部堆栈传递给GINP。由于BASIC解释程序总是使得SS和DS保持相同，因此程序可以借助于DS和BP寄存器从BASIC栈中找到参数地址，再通过SS和SP放入局部堆栈中。

（5）不断地调用GINP子程序，在求出A×B的每一个元素之后（共循环L×N次），必须从内存工作单元LOCAL_SPACE+2及LOCAL_SPACE中恢复BASIC的栈段及栈指

针之值，再恢复ES和BP寄存器，于是过程全部结束，最后正确地返回调用的BASIC程序。

（6）本例是一个完整的子程序，并且已经包括了上例所介绍的GINP。这样，本例在汇编后可以直接被BASIC语言程序调用。

例3-8　自然对数和常用对数

程序功能：对栈顶内容求以e为底的对数和以10为底的对数。

程序：

```
;NATURAL LOG(ST)
;SUBROUTINE LN
LN      PROC    NEAR
        FLDLN2                  ;PUSH LOG BASE E OF 2
        FXCH                    ;SWAP ST ST(1)
        FYL2X                   ;POP AND REPLACE ST
                                ;WITH NATURAL LOG
        RET
LN      ENDP
;
;COMMON LOG(ST)
;SUBROUTINE LOG10
LOG10   PROC    NEAR
        FLDLG2                  ;PUSH LOG BASE 10 OF 2
        FXCH                    ;SWAP ST,ST(1)
        FYL2X                   ;POP AND REPLACE ST
                                ;WITH COMMON LOG
        RET
LOG10   ENDP
```

说明：

（1）程序的算法利用了对数换底公式$\log_n X = \log_n 2 \times \log_2 X$。

（2）本例是一个NEAR过程，不能用BASIC直接调用。需要由BASIC调用时，应改写成一个FAR过程，读者不妨自行试试。

（3）从本例可以看出，8087的常数指令和超越函数指令给程序编写带来了许多方便。

例3—9　10^x，e^x，Y^x

程序功能：分别求以10、e、y为底，X为指数的幂。

程序：

```
;2 TO THE Z (ST)
;SUBROUTINE TWO2THEZ
;
;THIS ROUTINE ASSUMES THAT THE FOLLOWING MEMORY LOCATIONS
;HAVE BEEN DEFINED
;STATUS_WORD 2 BYTES
;CONTROL_WORD 2 BYTES
```

```
;CONTROL_WORD_TEMP 2 BYTES
;HALF HAS 0.5 IN SHORT_REAL FORMAT
;
;Z IS ASSUMED TO BE FOUND IN ST
;THERE MUST BE AT LEAST 2 FREE STACK LOCATIONS
TWO2THEZ PROC   NEAR
         PUSH   AX                      ;SAVE AX
         FSTCW  CONTROL_WORD            ;SAVE CONTROL WORD
         FSTCW  CONTROL_WORD_TEMP       ;USE TEMP TO CHANGE
                                        ;ROUNDING CONTROL(RC)
         FWAIT
         AND    CONTROL_WORD_TEMP,0F3FFH  ;CLEAR RC BITS
         OR     CONTROL_WORD_TEMP,0400H   ;RC=ROUND_DOWN
         FLDCW  CONTROL_WORD_TEMP         ;ROUND DOWN
         FLD    ST(0)                   ;PUSH COPY OF Z ONTO ST
         FRNDINT                        ;OK, ST=Z1, ST(1)=Z
         FLDCW  CONTROL_WORD            ;RETURN THINGS TO NORMAL
         FSUB   ST(1),ST                ;ST(1)(Z2)=Z-Z1
         FXCH                           ;ST=Z2,ST(1)=Z1
         FLD    HALF                    ;LOAD 1/2 ONTO THE STACK
         FXCH                           ;ST=Z2,ST(1)=1/2
         FPREM                          ;ST HAS Z2 OR Z2=1/2
                                        ;C1=1 IN THE LATTER CASE
         FSTSW  STATUS_WORD
         FWAIT                          ;NOW WE'VE GO FLAGS SET
         FSTP   ST(1)                   ;GET RID OF THE 1/2
         F2XM1                          ;ST=(2 TO THE ST)-1
         FLD1
         FADDP  ST(1),ST
         TEST   BYTE PTR STATUS_WORD+1,00000010B
                                        ;ST HAS Z2 IF BIT 1 ON
         JZ     WAS_Z2                  ;OTHERWISE  Z2-1/2
         FLD1
         FADD   ST,ST(0)                ;MULTIPLY BY THE
         FSQRT                          ;SQUARE ROOT OF 2
         FMULP  ST(1),ST

WAS_Z2:                                 ;WE JUST NEED TO SCALE
         FSCALE
         FSTP   ST(1)
         POP    AX
         RET
TWO2THEZ ENDP
;EXP(X)  (ST)
EXP      PROC   NEAR
         FLDL2E                         ;PUSH LOG E BASE 2
         FMULP  ST(1),ST                ;ST=X*(LOG E BASE 2)
         CALL   TWO2THEZ                ;ST=EXP(X)
         RET
EXP      ENDP
```

```
;10 TO THE X (ST)
TEN2THEX PROC    NEAR
         FLDL2T                        ;PUSH LOG 10 BASE 2
         FMULP   ST(1),ST
         CALL    TWO2THEZ              ;ST=10 TO THE X
         RET
TEN2THEX ENDP
;Y TO THE X (ST(1),ST)
;ASSUMES Y IS POSITIVE IN ST
;ASSUMES X IN ST(1)
Y2THEX   PROC    NEAR
         FYL2X                         ;ST1=X*(LOG Y BASE 2)
         CALL    TWO2THEZ              ;ST=Y TO THE X
         RET
Y2THEX   ENDP
```

说明：

（1）本例共包含四个NEAR过程，分别用来计算2^x、e^x、10^x、Y^x。X、Y分别在ST和ST(1)中，算法的依据是公式$Y^x = 2^{x \cdot \log_2 y}$。

（2）程序的关键部分在于计算2^z，即NEAR过程TWO2THEZ。程序先求出小于Z的最大整数Z_1，再计算$Z_2 = Z - Z_1$。如$Z_2 > \frac{1}{2}$，则进行运算$2^{(Z_2 - \frac{1}{2})} \times 2^{\frac{1}{2}}$；如$Z_2 < \frac{1}{2}$，则直接计算$2^{Z_2}$。最后将结果乘上$2^{Z_1}$。

（3）此程序利用指令F2XM1对$0 \leqslant X \leqslant 0.5$的变量进行运算，而利用FSCALE对整数进行运算。

（4）程序在求Z_1时，使用了指令FSTCW和FLDCW，并将控制字的RC字段临时清0，取向下舍入，保证Z_1一定小于Z。

（5）调用本例的子程序之前，调用程序必须预先定义常数HALF和工作单元STATUS_WORD，CONTROL_WORD，CONTROL_WORD_TEMP等，且保证栈中至少尚有两个单元是空的。

例3-10　ARCTAN(Z)

程序功能：对栈顶单元求反正切函数之值。

程序：

```
;ARCTAN (ST)
;ST IS ASSUMED TO BE A NORMAL NUMBER
;THERE MUST BE AT LEAST 3 FREE STACK LOCATIONS
;THIS ROUTINE ASSUMES THAT THE FOLLOWING MEMORY
;LOCATIONS HAVE BEEN DEFINED:
;STATUS_WORD 2 BYTES
;SING_STORE 1 BYTE
ARCTAN   PROC    NEAR
```

```
        PUSH    AX
;THE FIRST PROBLEM IS TO CHECK FOR A ZERO OR
        NEGATIVE ARGUMENT
        MOV     SING_STORE,0        ;ASSUME NON_NEGATIVE
        FTST
        FSTSW   STATUS_WORD
        FWAIT
        MOV     AH,BYTE PTR STATUS_WORD+1
        SAHF                        ;AH-->FLAG
        JA      POSITIVE
        JZ      ZERO        ;ASSUME ITS ZERO
        JMP     NEGATIVE
ZERO:
;ARCTAN(0)=0
        FSTP    ST(0)
        FLDZ
        JMP     DONE
NEGATIVE:
;DEAL WITH A NEGATIVE ARGUMENT USING IDENTITY

;ARCTAN(-X)=-ARCTAN(X)
        FCHS
        MOV     SING_STORE,-1
POSITIVE:
        FLD1
        FCOM
        FSTSW   STATUS_WORD
        FWAIT
        MOV     AH,BYTE PTR STATUS_WORD+1
        SAHF
        JA      Z_LT_1
        JC      Z_GT_1
;EXACTLY 1 RETURN ARCTAN(1)=PI/4
        FCHS                ;ST NOW=-1
        FADD    ST(0),ST(0)     ;ST=-2
        FLDPI
        FSCALE              ;ST NOW PI/4
        FSTP    ST(1)
        JMP     RESTORE_SING
Z_GT_1:
;USE IDENTITY ATAN(X)=PI/2-ATAN(1/X)
        FXCH                ;ST=Z,ST(1)=1
        FPATAN
        FLD1                ;NOW ADJUST BY PI/2
        FCHS
        FLDPI
        FSCALE
        FSTP    ST(1)
        FSUBRP  ST(1),ST
        JMP     RESTORE_SING
Z_LT_1:
```

```
        FPATAN              ;ST=1,ST(1)=Z
RESTORE_SING:
        TEST    SING_STORE,0FFH
        JZ      DONE
        FCHS
DONE:
        POP     AX
        RET
ARCTAN  ENDP
```

说明：

（1）本例的核心指令是FPATAN。通常情况下，此指令要用到的两个操作数X、Y，分别在ST和ST(1)中，并且指令限定它们应满足 $0<Y<X<\infty$。为了用一个操作数Z进行运算，程序假设X=1，Y=Z，并注意到公式：

$$\text{Arctan}(Z) = -\text{Arctan}(-Z)$$

$$\text{Arctan}(Z) = \pi/2 - \text{Arctan}(1/Z)$$

以解决操作数Z小于0和小于1的情况。

（2）本例中值得注意的是程序曾两次将8087的状态字传送至内存，供主处理器8088作转移控制之用。

例3-11　8087基准测试程序

程序功能：本例是测试8087性能的基准程序。程序用BASIC语言写成，它调用两个8087汇编语言子程序，分别用来计算5000个数的平方根和进行两个50×50阶矩阵的乘法。

程序：

① 求5000个数的平方根的8087汇编语言子程序：

```
;SUBROUTINE SQ(A)
;BEN1 FOR 5000 SQUARE ROOTS
        PUBLIC SQ
CSEG    SEGMENT 'CODE'
        ASSUME  CS:CSEG
SQ      PROC    FAR
        PUSH    BP
        MOV     BP,SP
        MOV     SI,[BP]+6
        MOV     CX,5000
S_LOOP:
        FLD     DWORD PTR[SI]
        FSQRT
        FSTP    DWORD PTR[SI]
        ADD     SI,4
        LOOP    S_LOOP
        POP     BP
        FWAIT
        RET     2
SQ      ENDP
CSEG    ENDS
        END
```

② 两个矩阵乘法的8087汇编语言子程序（见例3-7矩阵乘法子程序）。

③ 基准测试程序：

```
10      DIM D!(5000),A!(49,49),B!(49,49),C!(49,49)
20      FOR I=1 TO 5000
30      D!(I)=I
40      CALL SRMSIEEE(D!(I))
50      NEXT I
60      PRINT "START TIME FOR 5000 ROOTS IS ";TIME$
65      FOR I=1 TO 10
70      CALL SQ(D!(1))
75      NEXT I
80      PRINT "END TIME FOR 5000 ROOTS IS ";TIME$
90      FOR I=0 TO 49
100     FOR J=0 TO 49
110     A!(I,J)=(I+J)*0.05
120     B!(I,J)=(J-I)*0.01
130     CALL SRMSIEEE(A!(I,J))
140     CALL SRMSIEEE(B!(I,J))
150     NEXT J
160     NEXT I
165     M%=50
170     PRINT "START TIME FOR MATRIX MULTIPLICATION IS "
175     PRINT TIME$
180     CALL MATMULT(A!(0,0),B!(0,0),C!(0,0),M%,M%,M%)
190     PRINT "END TIME FOR MATRIX MULTIPLICATION IS ";
195     PRINT TIME$
200     END
```

以上基准程序的测试结果为：

```
START TIME FOR 5000 ROOTS IS 00:07:21
END TIME FOR 5000 ROOTS IS 00:07:24
START TIME FOR MATRIX MULTIPLICATION IS
00:07:35
END TIME FOR MATRIX MULTIPLICATION IS
00:07:43
```

为了数据精确起见，对5000个数求平方根共做了10遍。实际上，求5000个平方根只用了大约0.3秒钟的时间，而对两个50×50阶矩阵的乘法花了约8秒钟时间。

例3-12　数据求和子程序

程序功能：与例3-1、例3-3一样，用来对一组数据求和。所不同的只是本例为一个供FORTRAN（2.00）程序调用的子程序。

程序：

```
        PAGE    ,132
FRAME   STRUC
SAVEDS  DW      ?       ;saved copy of DS
SAVEBP  DW      ?       ;saved copy of BP
RETADDR DD      ?       ;4 bytes return address
DSUMA   DD      ?       ;4 bytes address of DSUM
NADDR   DD      ?       ;4 bytes address of N
ARRAYA  DD      ?       ;4 bytes address of array
FRAME   ENDS
DATA    SEGMENT PUBLIC 'DATA'
DATA    ENDS
DGROUP  GROUP DATA
;SUBROUTINE SUM(ARRAY,N,DSUM)
;ARRAY IS A SINGLE PRECISION ARRAY OF LENGTH N
;N IS AN INTEGER
;DSUM IS DOUBLE PRECISION
;
        PUBLIC      SUM
CSEG    SEGMENT     'CODE'
        ASSUME      CS:CSEG,DS:DGROUP,SS:DGROUP
SUM     PROC    FAR
        PUSH    BP
        PUSH    DS
        MOV     BP,SP
        LES     BX,[BP].ARRAYA
        LES     SI,[BP].NADDR
        MOV     CX,ES:[SI]
;now all set up to go
        FLDZ                        ;initialize st=0
        CMP     CX,0H
        JLE     DONE
;the next 3 instructions do all the hard work
ADD_LOOP:
        FADD    DWORD PTR ES:[BX]
        ADD     BX,4
        LOOP    ADD_LOOP
DONE:
;now file answer back in dsum
        LES     DI,[BP].DSUMA   ;DI=ADDR(DSUM)
        FSTP    QWORD PTR ES:[DI]
        POP     DS
        POP     BP
        FWAIT                       ;be sure 8087 is done
        RET     12
SUM     ENDP
CSEG    ENDS
        END
```

说明：

（1）用FORTRAN（2.00）调用汇编语言子程序不仅与BASIC语言不同，而且与FORTRAN（1.00）也不同。尽管这里所用参数的个数和参数类型与例3-3是一样的，但由于参数传递规程不一致，所以子程序中的处理也有差别。读者可将本例与例3-3进行对比。图3-15给出了FORTRAN（2.00）在调用汇编语言前后堆栈中的内容分布：

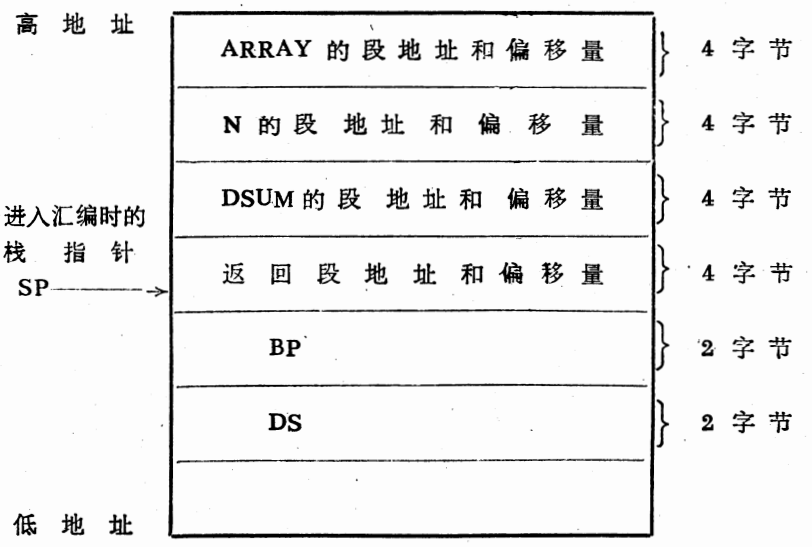


图3-15 FORTRAN（2·00）调用汇编子程序前后栈中内容

（2）由于CALL命令执行时压入堆栈的参数地址既包括段地址也包括偏移量，而且子程序中已用伪指令STRUC为这些参数分配了存贮空间，所以程序中采用了LES指令来传递段地址和偏移量。

（3）参数地址在栈中保存占4个字节，汇编语言子程序返回FORTRAN时指令应为：

```
RET   4 * n          （n为参数个数）
```

（4）调用汇编语子程序的FORTRAN（2.00）示范程序如下：

```
      DIMENSION ARRAY(10)
      INTEGER*2 N
      REAL*8 DSUM
      DO 10 I=1,10
      ARRAY(I)=FLOAT(I)
10    CONTINUE
      N=10
      CALL SUM(ARRAY(1),N,DSUM)
      WRITE(*,*)'DSUM=',DSUM
      END
```

只需对上述FORTRAN程序和汇编子程序分别编译和汇编，产生目标模块，然后再将两个模块连接成一个可运行文件即可以了。由于FORTRAN（2.00）本身采用了IEEE实数格式，因此不存在格式转换问题。

（5）程序的运行结果为：　　DSUM＝55.00000000000000

第五节　8087在高级语言中的使用

上一节介绍了高级语言调用8087汇编子程序的一些方法，这只是高级语言使用8087的一种间接方式。更为直接的使用方式是高级语言中所有算术运算、函数计算等都直接采用8087来完成。这种使用方式，其最大特点是简单方便。一般说来，用户不必对高级语言源程序进行修改，也不需要对8087的细节有任何了解，就可享用它的高速度、高精度等优越性能。由于MS-DOS下的高级语言编译系统的早期版本一般都不支持8087，所以可用8087的高级语言分成两种情况：一是对早期的版本进行扩充修改，如BASIC语言的编译87BASCOM等；二是使用新推出的版本，如FORTRAN（2.00）等。

1．在编译BASIC中的使用

如果用解释BASIC来运行8087，一般要更换ROM芯片或相应软件，但意义不大。因为8087的主要特点是速度快，而解释BASIC用在对语句解释上的时间相当长，8087又不具备完成这些功能的字符串指令。一般地说，8087只能用于改善算术运算的速度，不宜用于进行字符串处理。因此，即使有了支持8087的解释BASIC，效果也不会有明显改变。解释BASIC中使用8087的方法通常是：先用8087指令编写汇编语言子程序，然后再通过BASIC语句来调用这些子程序。

对于编译BASIC，情况就不同了。因为编译处理常常只是一次性的，而运行则是经常的，因此用编译BASIC来运行8087效果很明显。下面介绍的87BASCOM就是一种可用8087的编译BASIC，它是Microware公司对Microsoft编译BASIC(1.00)版进行扩充修改而成的。

（1）87BASCOM编译系统概况

87BASCOCM是Microware公司1983年的产品，它必须在机器上安装了8087芯片的条件下才能运行。

87BASCOM编译系统主要由以下四个文件组成：

* 87BASCOM·COM　　扩充了支持8087功能的BASIC编译程序
* 87BASCOM·LIB　　运行时不需要运行包的，可支持8087的库程序
* 87BASRUN·EXE　　可支持8087的运行包，它应改名为BASRUN·EXE
* CONVREAL·OBJ　　实数格式转换程序

这些文件再加上原来的MS编译BASIC的BASRUN·LIB文件（需要运行包的程序库），就可以进行正常的BASIC程序的编译和运行了。

（2）87BASCOM的操作

使用87BASCOM的编译过程与使用原编译BASIC一样，可选用下列两种程序库中的任何

一种：

* BASRUN.LIB程序库　这时必须使用原编译BASIC的BASRUN.LIB和改名为BASRUN.EXE的87BASRUN.EXE文件。

* 87BASCOM.LIB程序库　这时必须使用"O"开关，并在连接时对库名提示回答：87BASCOM或将87BASCOM.LIB改名为BASCOM.LIB。

下面用一个例子来说明编译过程。

例3-13　逼近法求π值

```
10 DEFDBL A-Z
20 THETA = .5 : DELTHETA =1
30 WHILE ABS(DELTHETA) >  0000000000000001#
40 RESULT = SIN(THETA)
50 DELTHETA = (.5#-RESULT)/RESULT
60 THETA = THETA +DELTHETA*THETA
65 LPRINT THETA, 6#*THETA
70 WEND
```

此例的算法是依据 $\sin(\frac{\pi}{6})=0.5$，用逐步逼近的方法计算π值，当误差小于10^{-15}时结束。

现将使用两种程序库分别编译上例的情况介绍如下。

使用BASRUN.LIB的操作过程

```
C>a:87bascom

IBM Personal Computer BASIC Compiler
(C)Copyright IBM Corp 1982 Version 1.00
(C)Copyright Microsoft, Inc. 1982

87BASIC(tm) Version 2.08
(C)Copyright MicroWare, Inc. 1983

Source filename [.BAS]: pi


Object filename [PI.OBJ]:
Source listing  [NUL.LST]:

22151 Bytes Available
21763 Bytes Free

     0 Warning Error(s)
     0 Severe  Error(s)

C>link
```

```
   Microsoft Object Linker V1.10
(C) Copyright 1981 by Microsoft Inc.

Object Modules [.OBJ]: pi
Run File [PI.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
```

使用87BASCOM.LIB的操作过程

```
C>a:87bascom

IBM Personal Computer BASIC Compiler
(C)Copyright IBM Corp 1982 Version 1.00
(C)Copyright Microsoft, Inc. 1982

87BASIC(tm) Version 2.08
(C)Copyright MicroWare, Inc. 1983

Source filename [.BAS]: pi/o
Object filename [PI.OBJ]:
Source listing  [NUL.LST]:

22151 Bytes Available
21763 Bytes Free

     0 Warning Error(s)
     0 Severe  Error(s)

C>link

   Microsoft Object Linker V1.10
(C) Copyright 1981 by Microsoft Inc.

Object Modules [.OBJ]: pi
Run File [PI.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]: a:87bascom
```

可见，两种操作方法只对目标程序的执行速度产生影响，而计算的结果则完全相同。例3-13的运算结果如下：

.5214574107333372	3.1287444644002 32
.5233998746607134	3.14039924796428
.5235802618353128	3.141481571011877
.5235970519955423	3.141582311973254
.523598615130633	3.141591690783798
.5235987606587108	3.141592563952265
.5235987742074185	3.141592645244511
.5235987754688075	3.141592652812845
.5235987755862431	3.141592653517459
.5235987755971765	3.141592653583059
.5235987755981943	3.141592653589166
.5235987755982892	3.141592653589735
.523598775598298	3.141592653589788
.5235987755982988	3.141592653589793
.5235987755982989	3.141592653589794

（3）数据格式的转换

如前所述，8087内部的数据格式是IEEE格式，这种格式在实数的表示形式上与Microsoft格式不相同。为了节省格式转换的时间，支持8087的高级语言往往直接采用IEEE格式，而87BASCOM就是这样。由于这一原因，如果用8087BASIC语言写的程序，要使用解释BASCI或原先的编译BASIC写的程序所产生的数据文件中的数据(这些数据一定是MS格式)，就必须经过格式转换。格式转换是通过子程序调用来实现的，这些子程序都在CONVREAL.OBJ中。实现格式转换的四个子程序是：

- SRIEEEMS　转换短实数：IEEE→Microsoft
- SRMSIEEE　转换短实数：Micrsoft→IEEE
- LRIEEEMS　转换长实数：IEEE→Microsoft
- LRMSIEEE　转换长实数：Microsoft→IEEE

如果程序中需要使用这些程序，则必须在连接操作时再加上目标模块CONVREAL。

（4）基准测试程序

用来对8087性能进行测试的一种BASIC基准程序如下：

```
10      DEFINT I-N:DEFDBL O-Z
20      DIM A(50,50),B(50,50),C(50,50)
30      PRINT"Start time for 5000 square roots is ";TIME$
35      FOR J=1 TO 10
40      FOR I=1 TO 5000
50      S=SQR(I)
60      NEXT I
65      NEXT J
70      PRINT"End time for 5000 square roots is ";TIME$
```

```
80      FOR I=1 TO 50
90      FOR J=1 TO 50
100     A(I,J)=(I+J)*0.05
110     B(I,J)=(J-I)*0.01
120     NEXT J
130     NEXT I
140     PRINT"Start time for matrix multiplication is ";
145     PRINT TIME$
150     FOR I=1 TO 50
160     FOR J=1 TO 50
170     C(I,J)=0.0
180     FOR K=1 TO 50
190     C(I,J)=C(I,J)+A(I,K)*B(K,J)
200     NEXT K
210     NEXT J
220     NEXT I
230     PRINT"End time for matrix multiplication is ";
235     PRINT TIME$
240     END
```

程序先求出5000个整数的平方根，然后对两个50阶的矩阵相乘。用BASIC编译程序1.0版和87BASCOM编译2.08版分别进行编译后，所得到的目标程序运行时间的测试数据见表3-9。为了使数据精确，5000个数开平方也做了10次，表中的数据是除以10后的结果。

表3-9　BASIC基准程序测试结果

编译程序名	使用的程序库	运行时间（秒）	
		5000个平方根	50×50阶矩阵乘法
BASIC编译1.0	BASRUN·LIB	7.5	121
BASIC编译1.0	BASCOM·LIB	6.2	98
87BASCOM2.08	BASRUN·LIB	2.8	71
87BASCOM2.08	87BASCOM·LIB	1.6	48

2. 在FORTRAN（1.00）编译系统中的使用

FORTRAN（1.00）原来是不能使用8087的。但是如果在连接（LINK）操作时，用修改后的可支持8087的程序库来代替以前的老程序库，8087也就可以使用了。Microware提供的8087软件包中就有这样一个库（文件名仍为FORTRAN.LIB）。整个编译过程不变，下面通过例3-14来说明FORTRAN（1.00）使用8087的情况。

例3-14　8087在FORTRAN（1.00）中的使用

此例假设在银行里有5美元存款，年息15.25%，每7天取1%，求多少天取完全部存款（小于1美分时）。FORTRAN源程序如下：

```
$NODEBUG
$STORAGE:2
      PROGRAM DEMO
C N WILL BE MY COUNTER FOR THE NUMBER OF DAYS
C MY ACCOUNT IS OPEN
      N = 1
C $5.00 IS MY STARTING BALANCE
      X = 5.00
10    X = (1.0 + (.1525/365.0))*X
      N = N+1
      IF (MOD(N,7) .EQ. 0) CALL DEDUCT(X)
      IF (X .LT. .01) THEN
         WRITE(*,'('' NUMBER OF DAYS '',I8)')N
         WRITE(*,'('' I NOW HAVE $'',F9.7)')X
         STOP
      ENDIF
      GOTO 10
      END
C THIS SUBROUTINE WILL JUST REDUCE THE AMOUNT IN
C MY ACCOUNT BY 1% WEEKLY
      SUBROUTINE DEDUCT(A
      A = .99*A
      RETURN
      END
```

运行结果如下：

不用8087时大约需要10秒钟时间。计算完毕输出结果为：

```
NUMBER OF DAYS 6111
I NOW HAVE $ .0099301
Stop-Program terminated.
```

使用了8087后，只要花约5秒钟时间就可得出如下结果：

```
NUMBER OF DAYS 6111
I NOW HAVE $ .0099305
Stop-Program terminated.
```

由本例可知，使用了8087后计算速度加快了1倍，而且精度也不一样。精度表现在余额的计算上。不使用8087时为0.99301美元，使用了8087时为0.99305美元。

3. 在FORTRAN（2.00）编译系统中的使用

（1）FORTRAN（2.00）编译系统的组成

FORTRAN(2.00)是1984年推出的新版本编译程序，它不仅修正了FORTRAN（1.00）中的若干缺陷和错误，而且对FORTRAN（1.00）的功能进行了较大的扩充，对使用8087也提供了许多方便。

FORTRAN（2.00）编译系统共有三张盘片，它主要由下列文件所组成：

- FOR1·EXE　　第1趟扫描程序
- FOR2·EXE　　第2趟扫描程序
- FOR3·EXE　　第3趟扫描程序
- LINK·EXE　　连接程序（2.20版）
- DOS20·LIB　　DOS（2.0）程序库
- SETUP·BAT　　建立工作盘的批处理程序
- MAKELIB·BAT　　建立FORTRAN程序库的批处理程序
- 8087.COM　　8087状态检查／设置程序
- IBMFOR.LIB　　系统日期、时钟检查／设置程序库
- REGMATH.LIB　　不使用8087硬件的程序库
- 8087ONLY.LIB　　使用8087硬件的程序库
- EMULATOR.LIB　　可用、可不用8087硬件的程序库
- LIB.EXE　　程序库管理程序

FORTRAN编译系统中的主要文件在编译过程中的作用可以用图3-16进行说明。

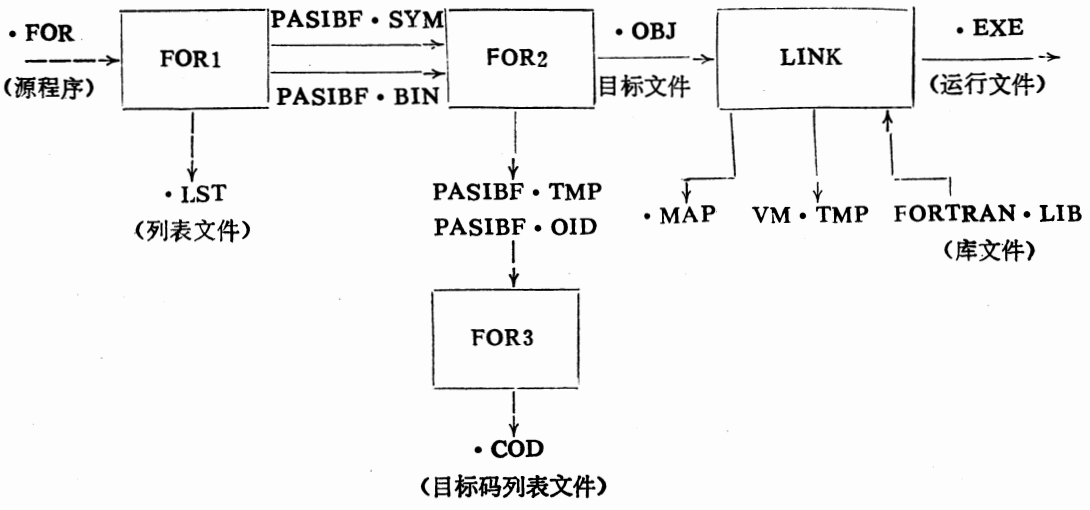


图3-16　FORTRAN(2.00)编译过程

其中，FOR1是编译的第1趟扫描，功能是进行语法检查。它输入带有.FOR类型名的FORTRAN源程序，经过处理后产生两个中间文件PASIBF.SYM和PASIBF.BIN作为FOR2的输入文件，前者是符号表文件，后者是中间二进制码文件。当需要时，FOR1还可以产生一个源程序的列表文件。如第1趟扫描未发现错误，便可进入第2趟扫描程序FOR2。

FOR2对程序作优化处理，它产生的PASIBF.TMP和PASIBF.OID文件供FOR3使用，同时产生以.OBJ为类型名的目标文件。

第3趟扫描FOR3可以运行也可以不运行，这取决于是否需要产生目标码列表文件。如不运行FOR3，需删除文件PASIBF.TMP和PASIBF.OID，以节省盘空间。

最后，用LINK连接目标模块文件和库文件，产生一个可以由MS-DOS装入执行的运行文件（.EXE文件）。

（2）FORTRAN.LIB的生成

从前面的叙述中可以知道，目标模块文件必须通过连接程序(LINK)与FORTRAN程序库（FORTRAN.LIB）进行连接操作后，才能得到可执行的运行文件。FORTRAN(2.00)可以使用三种不同的程序库：8087ONLY、REGMATH和EMULATOR。

* 8087ONLY

使用这个程序库时目标程序必须在装有8087的机器上才能运行，它产生的运行文件有最快的速度和最高的精度，且存贮空间比较节省。

* REGMATH

它在未装有8087的PC机上使用，精度较差，速度也低。

* EMULATOR

可在安装或未安装8087的两种条件下使用这个程序库。如未安装8087，运行文件能仿真8087指令的操作，其精度与8087ONLY相当，但速度大大降低，甚至低于REGMATH；如安装了8087，则运行速度和精度都能达到8087ONLY的同样水平，但EMULAOR生成的运行文件所化费的存贮空间比较大。

用户可根据自己的实际情况来选用最合适的程序库。表3-10是使用不同的程序库来解算例3-14时所得到的有关数据，供读者对比参考。

表3-10　三种库的运行情况

库　　名	运行时间（秒）	运行文件所占空间	结　　果	精度
EMULATOR	5.5（有芯片） 25.2（无芯片）	34826	·0099298	高
8087ONLY	4.9	27658	·0099298	高
REGMATH	11.3	29994	·0099310	低

（3）FORTRAN（2.00）编译系统的辅助程序

为了操作的方便，一般总是根据机器的具体情况和使用要求，从编译系统中挑选若干必须的文件组成一个工作盘。

批处理文件SETUP·BAT和MAKELIB·BAT是用来帮助用户建立工作盘的。下面先对SETUP进行介绍。

SETUP命令需要四个参数，无参数输入时则向用户给出使用说明。它的参数格式如下：

SETUP<盘符><程序库名><DOS版本号><磁盘类别>

其中，<盘符>指出要生成的工作盘片所在的驱动器号，<程序库名>表示选用三种库中的哪一个，<DOS版本号>表示所使用的是DOS1.1版还是DOS2.0版，<磁盘类别>则用来指出是单面软盘、双面软盘，还是硬盘（分别用参数1、2或不用参数加以区别）。

例如，在双面软盘上建立使用8087ONLY程序库的FORTRAN编译系统（需要两张格式

化好的空白盘），只需要打入下列命令：

```
A>SETUP B: 8087ONLY DOS20 2
```

这条命令打入后，批处理程序SETUP就开始建盘工作。经过上述一系列操作之后，可得到两张工作盘，目录如下：

第1张盘：

```
FOR1      EXE
FOR2      EXE
FOR3      EXE
```

第2张盘：

```
LINK      EXE
FORTRAN   LIB
```

对三种不同的程序库来说，第1张盘都相同，不同的是第2张盘中的FORTRAN·LIB文件。

为了使用不同的程序库文件组成FORTRAN·LIB，可以使用一个称为LIB·EXE的库管理程序，它的功能是对目标模块库进行构造和编辑。目标文件和别的库文件可通过它往一个库文件中加入，也可从一个库文件中移出或删除。

库管理程序的命令行格式为：

LIB 〔library-file 〔Pagesize〕 operations 〔, 〔list-file〕 〕 〔, 〔newlib〕 〔;〕 〕

其中：

* library-file　　老的库文件名
* 〔Pagesize〕　　页长，即一个目标模块的长度，其形式为“/Pagesize：N”或“/P：N”，N可以是16，32，64、128、256或512
* Operations　　所要完成的操作表，它可用操作符+、-、*、-+、-*对库文件进行添加、删除、移出、替换、移出删除等操作
* 〔, list-file〕　相互对照表的文件名
* 〔newlib〕　　生成的新的库文件名

在建立三种不同的FORTRAN程序库时，只要分别使用下面的键盘命令：

* 8087ONLY对应的FORTRAN·LIB

```
LIB PARTIAL·LIB/P:512+IEEEMATH·LIB+DOS20·LIB+8087 ONLY·LIB,
NUL, 〔B:〕 FORTRAN·LIB
```

* EMULATOR对应的FORTRAN·LIB

```
LIB PARTIAL·LIB/P:512+IEEEMATH·LIB+DOS20·LIB+EMULATOR·LIB,
NUL, 〔B:〕 FORTRAN·LIB
```

* REGMATH　　对应的FORTRAN·LIB

```
LIB PARTIAL·LIB/P:512+REGMATH·LIB+DOS20·LIB, NUL, 〔B:〕
FORTRAN·LIB
```

FORTRAN（2.00）编译系统中还有一个8087·COM文件，它通过如下命令实现对8087的开、关和测试：

* 8087　　测试8087的开关状态

* 8087 on 打开8087

* 8087 off 关闭8087

这个文件也应该拷贝到工作盘上，而且如果PC机已装了8087并想使用它，那么在装入FORTRAN（2.00）编译、连接后所得到的可运行目标程序并开始启动运行之前，必须键入命令：8087 on；不想运行8087，键入8087 off；要测试其状态，键入8087，机器会回答8087是在“on”还是在“off”状态。

和编译BASIC一样，FORTRAN（2.00）的编译程序也采用IEEE实型数格式，与以前版本的Microsoft格式不同。需要时两种格式可用以下两个库子程序进行转换：

* SUBROUTINE M2ISQQ（RIBM，RIEEE）将MS格式变为IEEE格式

* SUBROUTINE I2MSQQ（RIEEE，RIBM）将IEEE格式变为MS格式

其中，RIBM和RIEEE分别为MS格式的实型变量和IEEE格式的实型变量。

（4）基准测试程序

测试FORTAN（2.00）的一种基准程序清单如下：

```
      PROGRAM BENCH
      DIMENSION A(50,50) B(50 50  C(50 50
      LOGICAL*2 X,SETTIM
      INTEGER*2 HOUR,MIN,SEC,HSEC
      X=SETTIM(0,0,0,0)
      DO 10 I=1,5000
      S=SQRT(FLOAT(I)
10    CONTINUE
      CALL GETTIM(HOUR,MIN,SEC,HSEC)
      WRITE(*,4000)HOUR,MIN,SEC,HSEC
4000  FORMAT(' TIME FOR 5000 SQUARE ROOTS IS',4I4)
      DO 20 I=1,50
      DO 20 J=1,50
      A(I,J)=(I+J)*0 05
      B(I,J)=(J-I)*0 01
20    CONTINUE
      X=SETTIM(0,0,0,0)
      DO 30 I=1,50
      DO 30 J=1,50
      C(I,J)=0 0
      DO 30 K=1,50
      C(I,J)=C(I,J)+A(I,K)*B(K,J)
30    CONTINUE
      CALL GETTIM(HOUR,MIN,SEC,HSEC)
      WRITE(*,5000)HOUR,MIN,SEC,HSEC
5000  FORMAT(' TIME FOR MATRIX MULTIPLICATION IS  4I4)
      END
```

它也是首先求5000个数的平方根，然后进行两个50阶矩阵的乘法运算。为了测试时间，

程序利用函数SETTIM设定起始时间，用子程序GETTIM测试终止时间，它们都在库程序IBMFOR·LIB中。

编译和连接过程如下：

```
C>for1

IBM Personal Computer FORTRAN Compiler
Version 2.00
(C)Copyright IBM Corp 1982, 1984
(C)Copyright Microsoft Corp 1982, 1984

Source filename [.FOR]: bench
Object filename [BENCH.OBJ]:
Source listing  [NUL.LST]:
Object listing  [NUL.COD]:
 Pass One     No Errors Detected
              27 Source Lines

C>for2

 Code Area Size = #039B   (  923)
 Cons Area Size = #0038   (   56)
 Data Area Size = #759F   (30111)

 Pass Two     No Errors Detected

C>link

IBM Personal Computer Linker
Version 2.20 (C)Copyright IBM Corp 1981, 1982, 1983, 1984

Object Modules [.OBJ]: bench
Run File [BENCH.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]: fortran+ibmfor
```

当使用三种不同的程序库运行时，FORTRAN的基准测试程序的有关数据见表3-11。

表3-11　FORTRAN基准程序测试数据

库　　名	运行文件所占空间	运行时间（秒）	
		5000个平方根	50×50阶矩阵乘法
EMULATOR	63660	24.0（无芯片）	262.76（无芯片）
		1.86（有芯片）	50.2（有芯片）
8087ONLY	55148	1.75	41.96
REGMATH	59910	3.73	115.23

第四章　C语言及其程序设计

C语言是由美国贝尔实验室的D.M.Ritchie在1972年提出来的一种通用程序设计语言。它是一种较“低级”的语言，可以用来书写系统程序和应用程序。C语言的设计者在设计C语言时，较好地处理了简洁性与实用性、可移植性和高效率之间的矛盾。

从使用者的角度来看，C语言主要具备下列特点：

* C语言适当地考虑了背景机的特点，目标程序的质量较高，比较适合编制系统程序。例如，著名的UNIX操作系统主要是使用C语言书写的。

* C语言程序通常由若干个C函数组成，它具有良好的模块化结构，程序结构清晰且紧凑，同时也易于使用若干个较小的程序模块组成较大的程序。

* C语言具有丰富的运算符、实用的表达式和先进的数据结构与控制结构，表达能力强且灵活。

* C语言程序书写简炼，易学易写。

当然，C语言也存在一些缺点和不足。例如，它的某些运算符的优先顺序不符合日常习惯；复合语句多层嵌套时语句括号匹配情况不够醒目；C语言的弱类型转换具有潜在的不安全因素等等。尽管如此，C语言由于其较高的效率和较强的表达能力而受到广大程序设计者的欢迎。

C语言与UNIX操作系统有着紧密的联系，可以说，如果要使用UNIX，就不能不使用C语言。随着UNIX或类UNIX操作系统在微型计算机上的普遍使用以及微机操作系统的“UNIX化”，C语言也在微型机用户中流行起来，成为微型机上有影响的一种程序设计语言。

IBM PC的主操作系统MS-DOS已配有多种不同版本的C语言，其中美国Computer Innovations公司的优化CI-C86是一个功能齐全的C语言，它的主要特点如下：

* 提供丰富的系统库函数，其中有些函数是CI-C86扩充的。

* 在大模式下，允许用户程序使用大于64KB的动态数据区。

* 产生的目标程序采用Microsoft公司的.OBJ文件的兼容格式，可直接使用标准的Microsoft连接程序LINK进行连接。

* 支持Intel8087浮点运算协处理器。

* 可以生成汇编形式的目标文件，该文件可以使用Microsoft公司的宏汇编MASM进行汇编。

* 提供了一组基本绘图函数，可利用IBM PC的图形显示器显示图形。

本章将以优化CI-C86 2.00版（下文中简称为C86）为例介绍C语言。其它公司为IBM PC所开发的各种C语言，其功能和操作大体上都与C86类似，本书就不一一介绍了。

第一节　C语言基础

C语言书写风格独特，表达形式也比较精炼，语言的效率较高，但同时也给初学者带来

了一定的困难。为了使读者对C语言有个直观的了解，本节对C语言的主要成分作一入门性的介绍，并给出几个用C语言写成的例子，便于读者能够较快地熟悉和掌握C语言的表达方式。

1．C语言程序的结构

用C语言编写的程序通常由包括一个main（）函数在内的一组函数组成。其中main（）函数是一个特殊函数，称为主函数。一个程序总是从它的主函数开始执行，所以任何一个程序都不能没有main（）函数。

C语言中的函数是一个独立的实体，它可以单独编译。一个函数的定义由函数名及其参数表、参数类型说明和函数体等三部分组成。函数名前可以冠以类型说明符，用来说明函数的类型。函数体是用一个复合语句（或称为分程序）表示的函数的实现部分。而复合语句则由语句括号“{”和“}”（类似PASCAL语言中的“BEGIN”和“END”）括起来的一组说明和语句组成。

下例就是一个完整的C程序。

例4-1　求整数平方

（1）功能：打印2的平方，再打印该平方的平方，直至256。

（2）程序与运行结果：

```
/* EX4-1 Print the square of 2, the square of
** that square, and so forth up to 256 */
main() 主函数
{
    int i = 2;

    while(i < 256)
        Printf("%d\n", i = square(i));
}
square(x)     /* square the integer x */
int x;
{
    return(x * x);
}


A>EX4-1
4
16
256
```

（3）说明：

① 该程序由两个函数组成，其中main（）是主函数；square（）为一个自定义函数，用来计算一个整数平方（C语言不提供乘方运算符）。main（）函数没有参数，而square（）为有参函数，其参数x为欲求平方的整型量。

② 主函数中的语句

```
int i= 2;
```

为说明语句，它用来指明变量i的类型为整型，初值为2。

③ main函数主要由一个while重复语句（也称为循环语句）组成，它的重复条件为“i＜256”。其中语句部分是一个系统库中的格式输出函数printf（）的调用语句。printf（）的第1个参数为输出格式串，用来指出输出的格式。其中“%d”为转换说明，表示将欲输出的表达式的值转换成十进制整数形式输出；“\n”为新行符，表示输出数据后要输出回车换行符。第2个参数为输出表达式（详见第三节）。所以，这条printf（）调用语句的含义为：以十进制整数的形式输出赋值表达式i＝square（i）的值（即左部变量i的值），并输出一个新行符“\n”。

④ C语言中的“＝”称为赋值运算符，它的含义是将右部表达式的值赋给左部变量。含赋值运算符的表达式称为赋值表达式，它的值等于左部变量的值。

⑤ C语言的分号“；”是语句结束符，任何语句（复合语句除外）都必须以分号结尾。例如，主函数中while语句后的分号是不可缺省的。

⑥ square（）函数中的语句

```
return（x＊x）;
```

是函数返回语句，它的功能是将表达式“x＊x”的值回送给该函数的调用函数，并把控制返回到该调用函数。

⑦ C语言的注释是由“/＊”和“＊/”括起来的一串字符。注释可以在空格和新行符能出现的任何地方出现。

⑧ C语言的关键字都是由小写字母组成的，使用时应格外小心。根据经验，如无特殊需要，在使用C语言时最好全部使用小写字母。

2．常量与变量

C语言中使用的变量用标识符来表示，在使用前必须使用变量说明语句加以说明。变量说明既可以指定变量的类型，也可以指定变量的初值。

C语言的常量可分为直接常量和符号常量两类，符号常量在使用前必须使用编译命令“＃define”定义，而直接常量则可直接使用。例4－2是使用符号常量的一个例子。

例4-2　素数的测试

（1）功能：判别一个给定的整数是不是素数，若是，输出“prime”，否则输出“not prime”。

（2）程序与运行结果：

```
/* EX4-2 Test an integer to see if
** it's a prime number */
#define ZERO 0
#define ONE 1
#define NUMBER 1234
main()
{
```

```
    int i = ONE, n = NUMBER;

    while(++i < n)
        if(n % i == ZERO)
        {
            printf("not prime\n");
            break;
        }
    if(i == n)
        printf("prime\n");
```

```
A>EX4-2
not prime
```

（3）说明：

① 本例首先使用#define命令定义了三个符号常量，即ZERO、ONE和NUMBER。其中ZERO和ONE分别代表0和1，它们有助于改善程序的可读性。NUMBER是一个需要判别是否为素数的整数，这种用法有利于程序的修改，例如要判别“567”是不是素数，只要将程序的第3行改为

```
#define    NUMBER  567;
```

即可。

② 本例的while语句中使用了几个运算符，其中“++”是C语言的单目增量运算符。表达式“++i”的含义为“i=i+1”；“%”为取模运算符；“==”为关系运算符“等于”。

③ 程序中的break语句是C语言的一个中断语句，它用来退出while语句。

④ 标识符的大小写在C86中是有区别的，使用时应当注意。通常使用大写的标识符表示符号常量，而使用小写的标识符表示变量。

3. 运算符

C语言的运算符十分丰富，使用方法也比较独特，这是C语言的主要特点之一。C语言共有40余个运算符，关于这些运算符的具体含义以及使用方法将在第二节中介绍，下面仅给出一个例子。

例4-3　运算符的检验

（1）功能：检验C语言的几个运算符。

（2）程序与运行结果：

```
/* EX4-3 Description of some operators */
#define PRINT(I) printf("I = %d\n", I)
main()
{
    int x, y, z;
```

```
 x = y = z = 2;
 z = x == y; PRINT(z);
 PRINT(x++); PRINT(++y);
 z += x++; PRINT(z);
 PRINT(x < y ? y : x);
}
A>EX4-3
z = 1
x++ = 2
++y = 3
z = 4
x < y ? y : x = 4
```

（3）说明：

① 该程序中的语句

```
x=y=z= 2;
```

是多重赋值语句。因为赋值运算符的结合特性是自右至左（详见第二节中的表4-6），即赋值运算自右至左逐个进行，所以该语句等效于下列语句：

```
x=（y=（z= 2））;
```

该语句首先把常量2赋值给z，再把表达式“z= 2”的值（等于2）赋给y，最后将表达式“y=（z= 2）”的值（也等于2）赋给x。

② 因为运算符“==”的优先级高于运算符“=”，所以语句

```
z=x==y;
```

等效于

```
z=（x==y）;
```

这里的x和y的值又都等于2，故x==y为真。C语言中没有逻辑型数据类型。逻辑量通常使用整型量表示，用0表示“假”，用非0的值（多用1）表示“真”。所以该语句执行后，z的值为1。

③ C语言的增量“++”和减量“--”运算符可以前置，也可以后置，它们都把运算分量加1（或减1）。但前置时表达式的值为增量（或减量）后的运算分量的值，而后置时则为增量（或减量）前的运算分量的值。所以，本例在执行了下列两个语句

```
PRINT（x++）; PRINT(++y);
```

之后，输出的结果为：

```
x++= 2
++y= 3
```

但这时x和y的值都为3。

④ 语句

```
z+=x++;
```

中的运算符“+=”是C语言的自反运算符。该语句等效于：

```
z=z+（x++）;
```

因为在执行该语句时z和x的值分别为1和3，所以执行该语句后z的值为4。

⑤ 程序的最后一个语句

```
PRINT(x<y? y:x);
```

中使用了C语言的三目条件运算符“? :”。使用条件运算符的表达式“x<y? y:x”的含义为：当“x<y”为真时，表达式的值取y；当“x<y”为假时，表达式的值取x。即表达式“x<y? y:x”表示取x和y中较大的一个为值。

⑥ C语言允许使用编译命令#define定义一个宏替换，即用一个标识符（宏替换的名）表示一段程序（宏替换的体）。在使用时（调用宏替换时），用宏替换名即可代替那一段程序；而在编译的预处理时，再用宏替换的体替换它的名（称为宏扩展）。定义宏替换时还可以使用形式参数，这些形式参数在宏扩展时将使用实在参数进行替代。例如，本例中将“PRINT(I)”定义为下列库函数的调用：

```
printf( "I=%d\n", I)
```

表示以十进制整数形式输出一个数据。其中I为形式参数，在宏替换的调用时用实在参数代入。

4. 表达式与语句

C语言的表达式由运算符运算分量组成，由于C语言的运算符相当多，故其表达式也比其它语言丰富。下列是一些C语言的表达式：

```
i=square(i)
(a>b)? a:b
z+=-x++ + ++y
```

C语言的语句可分为四类。第1类是说明语句，它用来说明变量的类型和初值。第2类是表达式语句，它由表达式加分号组成。第3类是程序控制语句，用来描述语句的执行条件与执行顺序。最后一类是复合语句，它由用语句括号“{”和“}”括起来的一组语句组成。

下面是一个标准输入到标准输出的复制程序。

例4-4　输入到输出的复制

（1）功能：将标准输入设备（键盘）的输入逐行复制到标准输出设备（显示器）上，直至输入文件结束符ctrl-z为止。

（2）程序：

```
/* EX4-4 Copy input to output */
#include "stdio.h"
main()
{
    int c;
    c = getchar();
```

```
    while(c != EOF)
    {
        putchar(c);
        c = getchar();
    }
}
```

（3）说明：

① 程序中的编译命令

#include “stdio·h”

用来嵌入一个标准输入输出的说明文件，该文件定义了一些与输入输出有关的符号常量和宏替换。例如，EOF在stdio·h中被定义为－1。

② 程序中的语句

c＝getchar（）；

putchar（c）；

都是表达式语句。其中getchar（）和putchar（）可以看成两个系统库函数，分别用来从标准输入输出设备上输入和输出一个字符。while语句是程序控制语句，其循环体是一个复合语句。“int c；”是说明语句，它用来指出c是整型变量（C语言允许使用整型量表示字符型数据）。

③ 使用getchar（）输入数据时，当遇到文件结束字符ctrl-z时，该函数会返回－1。这时c就和EOF相等，程序就结束了。

④ 原则上说，使用C语言的标准输入输出函数就可以访问任何文件和设备，这是通过利用MS-DOS输入输出重定向功能来实现的。利用本例进行输入、输出和复制文件的操作命令如下：

```
A>EX4－4   <SFILE   >DFILE      /*复制文件*/
A>EX4－4   <SFILE   >PRN        /*打印文件*/
A>EX4－4   <CON     >DFILE      /*输入文件*/
```

5．函数

C语言的函数是一个能完成一定功能的封闭模块，使用起来十分方便。函数的类型由函数定义时函数名前的类型说明符确定，类型说明符缺省时为整型。

函数的参数在函数定义时应予以说明，如不说明则为整型。C语言传递参数的方式是按值传送，在函数体内修改参数的值对调用函数是无效的，故一般不在函数体内修改参数的值。返回参数（函数值）是使用语句return传递给调用函数的，其类型为函数的类型。

程序在调用一个函数之前，应对被调函数的类型加以说明（整型函数可缺省），否则会产生意想不到的结果。函数调用中实参的个数以及顺序应与函数定义中的形参相匹配。形参和实参的类型一般也必须一致，但对于整型形参可用字符型实参匹配，长实型形参可用实型实参匹配。

例4-5　求实数平方

（1）功能：求实数1.5的平方，求这个平方的平方，直至256.0。

（2）程序与运行结果：

```
/* EX4-5 Print the square of 1.5, the square of
** that square, and so forth up to 256.0 */
main()
{
    float i = 1.5;
    double xsquare();

    while(i < 256.0)
        printf("%f\n", i = xsquare(i));
}
double xsquare(x) /* square a float or a double */
double x;
{
    return(x * x);
}
```

```
A>EX4-5
2.250000
5.062500
25.628906
656.840836
```

（3）说明：

① 该程序定义了一个长实型函数xsquare（），它的参数也为长实型。该函数的功能是用来计算一个实型数的平方。

② 在main（）函数中使用说明语句

```
double  xsquare();
```

说明main（）函数中使用的函数xsquare（）是长实型函数。如不使用该说明语句，程序不会得出正确的结果。

③ 尽管xsquare（）函数的参数被说明为长实型，但也可用它求实型数的平方，不过不能用它直接求出整数的平方。例如，下列语句：

```
printf( "%d\n", xsquare( 4 ) );
```

输出的结果是错误的，应改为

```
printf( "%f\n", xsquare( 4.0 ) );
```

才正确。C语言对这类错误不作为语法错误给出出错信息，所以初学者应格外小心。

第二节　数据类型及其操作

C语言的数据类型可以分成基本类型（字符、整型和实型）和导出类型（指针、数组、结构和联合）两类，其中导出类型中的数组、结构和联合又称为复合类型。本节将分别介绍这些数据类型及其有关的操作。

1. 基本类型

（1）基本类型的表示

① 字符

字符类型（char）可用来表示一个ASCII字符，也可表示－128—＋127之间的整数。无符号字符（unsigned char）可以用来表示一个字节或表示0—255之间的整数。字符和无符号字符在内存中占用1个字节。

字符常量可用由单引号括起来的字符表示，例如：‘1’，‘A’，‘.’等都是字符常量。对于不可显示的控制字符，可以采用以反斜杠开头的专用转义字符或八进制的ASCII代码表示。例如，用‘\n’表示新行符，用‘\7’表示告警（响铃）符等。常用控制字符的转义序列如表4－1所示。

表4-1　控制字符的转义序列

转义序列	含义	转义序列	含义
\0	空字符	\r	回车
\b	退格符	\f	换页
\t	水平制表	\\	反斜杠
\n	新行符	\×××	ASCII码为八进制×××的字符

② 整型

C语言的整型可以用来表示整数、数组下标、逻辑值和二进制位串。它又可以分为整型（int）、无符号整型（unsigned）、短整型（short）、无符号短整型（unsigned short）、长整型（long）和无符号长整型（unsigned long）六类。C86中short与int、unsigned short与unsigned的表示相同，故实际上仅有四种不同的整数类型。

四种整型的表示范围以及在内存中占的字节数如下：

* 整型变量的取值范围为－32768—＋32767；无符号整型变量的取值范围为0-65535。它们在内存中均占2个字节。

* 长整型变量的取值范围为－2147483648—＋2147483647；无符号长整数的取值范围为0—4294967295。两者在内存中均占用4个字节。

整型常数有十进制、八进制和十六进制三种表示形式。其中，在八进制和十六进制常数之前分别必须冠以“0”和“0x”。例如，－21表示十进制数－21，07表示八进制数7，0xff表示十六进制数ff。长整型常数在数后要后置一个“l”，例如：271，0x307fl等都是长实型常数。

③ 实型

实型（或称为浮点类型）用于表示实数，它又分为实型（float）和长实型（double）两类。它们的表示范围及机内长度如下：

* 实型变量的取值范围约为－1.0e38—＋1.0e38，它在内存中占4个字节。

* 长实型变量的取值范围约为－1.0e308—＋1.0e308，它在内存中占8个字节。

实型常数在C语言中总为长实型量，它可以用普通小数或科学记数法来表示。例如：－123.0，.123，1.23e2等都是实型常数。它们的精度分别为6和15位。

（2）类型说明

C语言的变量在使用前对它的存贮类、类型和初值都应加以说明。C语言的变量说明语句的格式如下：

〔〈存贮类〉〕〈类型〉〈标识符〉〔＝〈初值〉〕{，〈标识符〉〔＝〈初值〉〕}；

其中，“存贮类”可以是auto或register，也可以是static或extern；在C86中“类型”主要有：char（字符），unsigned char（无符号字符），int（整型），unsigned(无符号整型)，long（长整型），unsigned long（无符号长整型），float（实型）和double（长实型）八种；“标识符”即为变量名；“初值”为常量表达式，其值将作为指定变量的初值。

下面解释一下存贮类的含义以及定义初值的方法。

① 存贮类

C语言存贮类用来说明变量的作用域、生存期和存贮方式。函数内的局部说明中允许出现这四种存贮类的任意一种，而在外部说明中则仅允许出现static和extern这两种存贮类。现分别介绍这四种存贮类的含义。

* auto存贮类

使用auto存贮类说明的变量称为自动变量，它的作用域局限于定义该变量的复合语句。当离开该复合语句后，这个变量就无意义了，它的值也不再保留。例如，程序

```
main ( )
 {
      addone ( ) ; addone ( ) ; addone ( ) ;
 }
addone ( )
 {
      auto int i= 0 ;
      printf ( "%d  ", ++i ) ;
 }
```

的运行结果为：

```
1   1   1
```

auto存贮类仅能用于局部说明，它也是局部说明的缺省存贮类。上例addone ( ) 函数中的说明语句可简写为：

```
int i= 0 ;
```

* register存贮类

使用register存贮类说明的变量为寄存器变量，它的作用域与自动变量相同，但编译程序尽量把它保存在寄存器中。C86目前没有register存贮类，它把寄存器变量作为自动变量处理。

* static存贮类

static存贮类用来说明静态变量（或称为固有变量）。静态变量的作用域为定义该变量

的复合语句（局部说明）或程序文件（外部说明）。离开它的定义域后，该变量就无意义了。但它的值仍然被保留，并作为下次进入该定义域时该变量的初值。例如，执行程序

```
main ( )
{
      inc ( ) ; inc ( ) ; inc ( ) ;
}
inc ( )
{
      static i= 0 ;
      printf ( "%d " , ++i) ;
}
```

得到的结果为：

```
1   2   3
```

这是因为离开函数inc（）后，变量i的值仍保留且作为下次进入inc（）时的初值的缘故。

* extern存贮类

extern存贮类说明的变量是在函数外或别的程序中定义的外部变量，它的作用域是整个程序。外部变量可以在程序的任何文件中定义，定义时必须使用不带存贮类的外部说明，也可以赋初值。外部变量在使用前必须用带extern存贮类的局部说明或外部说明加以说明。下面是使用外部变量的一个例子：

```
int x = 123;
main ( )
{
      extern int x, y;
      printf ( "%d %d \n" , x, y) ;
}
int y = 321;
```

其运行结果如下：

```
1 2 3   3 2 1
```

② 定义初值

C语言在变量说明中可以定义变量的初值，使用起来很方便。下面就是一些说明语句的例子：

```
char  sex='F', linefeed='\n', bell='\7';
int  studentnumber=187+40, graduatingclass=1985;
double  loge=2.718281828459;
float  x, y, z=1.23;
```

变量如果不定义初值，其初值就是不可预测的废码（如本例最后一行中的x和y）。

（3）类型转化

C语言在表达式计算时能自动地对变量进行类型转换。转换规则如图4-1所示。字符型

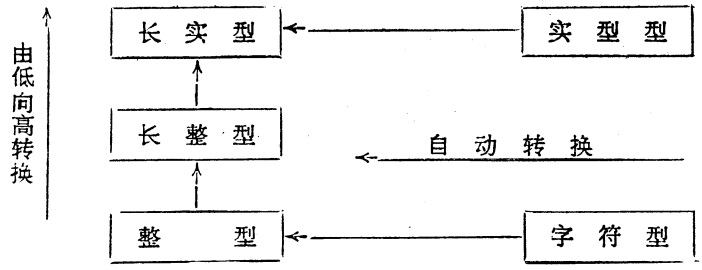


图4-1 类型转换的规则

的量在运算之前总是先转换成整型量，而实型量在运算之前总是转换成为长实型量。对含有两种类型操作数的表达式，操作数先转换成表达式中级别较高的类型，然后才参加运算。

赋值表达式的结果类型与赋值运算符左部变量的类型相同，当左部变量的类型级别低于右部表达式的类型级别时，要进行舍入。例如，当d为长实型量而 i 为整型量时，执行语句

```
i=d=100/3.0;
```

后，d和i的值分别为33.333333和33。

对于表达式，还可以使用强制类型转换运算符进行类型转换。强制类型转换运算符的使用格式如下：

（〈类型〉）〈表达式〉

它表示把给定的表达式的类型强制转换成前面圆括号中给定的类型。例如，要利用例4-5中的长实型函数xsquare（）来求整数i的平方，可用下列语句实现：

```
i = xsquare ( (double) i );
```

此外，C86还提供了一组用于类型转换的系统函数（如表4-2所示），其中许多函数使用了指针类型的参数（例如atoi（）函数中的形参string）。关于指针和字符串的有关概念请参见本节的第2和第3段。

表4-2 类型转换函数

函数及其类型	参数类型	功能及举例
long atoi(string)	char * string;	把字符串表示的数转换为长整数返回，例如字符串“32”的返回值为32
int itoa(n, buf)	int n; char * buf;	把整型量n转换为字符串表示的数放入buf，并返回其长度，例如可用该函数将-32转换为“-32”
int ltoa(n, buf)	long n; char * buf;	把长整型量n转换为字符串放入buf并返回其长度
int itoh(n, buf)	unsigned n; char * buf;	把无符号整数n转换为十六进制格式的字符串放入buf并返回它的长度

（续表4-2）

int ltoh(n, buf)	unsigned long n; char *buf;	把无符号长整数转换成十六进制的字符串放入buf并返回其长度
int utoa(n, buf)	unsigned n; char *buf;	把无符号整数n转换成十进制字符串放入buf并返回其长度
int ltos(n, buf, b)	long n; char *buf; int b;	把长整数n转换成b进制的字符串放入buf，返回它的长度
double atof(string)	char *string;	把字符串string转换为长实型数返回
int ftoa(n, buf, i, f)	double n; char *buf; unsigned i, f;	把长实数n转换成含i位整数和f位小数的字符串放入buf，返回它的长度
double ceil(n)	double n;	把长实数n的上整数返回，例如 ceil(45.67) =46.0, ceil(-2.89) =-2.0
double floor(n)	double n;	把长实数n的下整数返回，例如 floor (45.67) =45.0, floor(-2.89) =-3.0
double frexp(n, eptr)	double n; int *eptr;	把n分为阶码和尾数两部分，用eptr指向阶码并返回尾数，它们的关系如下： n==frexp(n, &i) *pow(2, i) 其中pow(2, i) 表示2的i次方
double ldexp(m, e)	double m; int e;	以阶码e和尾数m组成一个长实数返回，该函数是frexp ( ) 的逆函数
double modf(n, iptr)	double n; double *iptr;	把n的整数部分和小数分开，用iPtr指向整数部分并返回小数部分，例如 modf(34.56, &d) =.56, d=34

（4）基本类型的运算

由基本类型的隐式转换规则可以看出，基本类型可分为两大类。第1类为整型，包括字符、整型和长整型等类型；第2类为实型，包括实型和长实型。下面分别介绍定义在这些类型上的运算符和有关的系统库函数。

① 整型量的算术运算

C语言对整型量可进行如下算术运算：

* 算术运算：－（单目负），＋＋（增量），－－（减量），＋，－，*，／，％（取模）。
* 位串运算：～（求反），&（按位与），｜（按位或），∧（按位异或），《（左移），》（右移）。

* 逻辑运算：&&（与），||（或），!（非）。

* 关系运算：<，>，<=，>=，==（等于），!=（不等于）。

* 判断类型的大小（包括实型）：sizeof（表达式）或sizeof（（类型））。

算术运算中的增量和减量运算符是单目运算符，它们的运算分量可允许为变量，并且有前置和后置两种使用方法，其含义不完全相同。例如，当i为5时i++的值为5，而++i的值为6。

位串运算是一种按位进行的逻辑运算。例如，“&”运算符表示逐位求与，“0x33&0x0f”的值为“0x03”。逻辑运算是将变量的值作为逻辑值来进行运算的，例如“0x33&&0x0f”的值为1（真）。

位串运算中的左移“<<”和右移“>>”运算是一种算术移位。移位表达式

x<<n 和 x>>n

分别表示x左移n位低位补零和x右移n位高位补符号位。这里的n必须为正整数。例如：

4<<2 和 -8>>3

的值分别为16和-1。

单目运算符sizeof可用来求一个基本类型或基本变量在内存中占用的字节数。下面是一个使用运算符sizeof的例子。

例4-6　测试基本类型

功能：打印C语言的6个基本类型在内存中所占的字节数。

程序与运行结果：

```
/* EX4-6 Test types */
#define TESTTYPE(TY) printf("TY\t%d byte(s)\n", \
                     sizeof(TY))
main()
{
    TESTTYPE(char);
    TESTTYPE(int);
    TESTTYPE(short);
    TESTTYPE(long);
    TESTTYPE(float);
    TESTTYPE(double);
}
```

```
A>EX4-6
char     1 byte(s)
int      2 byte(s)
short    2 byte(s)
long     4 byte(s)
float    4 byte(s)
double   8 byte(s)
```

② 实型量的算术运算

对实型量可以进行下列运算和操作：

* 算术运算：－（单目负），＋，－，＊，／。
* 关系运算：＜，＞，＜＝，＞＝，＝＝，!＝。
* 系统库函数：如表4－3所示。

表4-3　算术运算函数

函数及其类型	参数类型	功能
double sin(x)	double x;	正弦函数
double cos(x)	double x;	余弦函数
double tan(x)	double x;	正切函数
double cotan(x)	double x;	余切函数
double asin(x)	double x;	反正弦函数
double acos(x)	double x;	反余弦函数
double atan(x)	double x;	反正切函数
double atan2(x, y)	double x, y;	x/y的反正切函数
double fabs(x)	double x;	绝对值函数
double sqrt(x)	double x;	平方根函数
double exp(x)	double x;	指数函数
double log(x)	double x;	自然对数函数
double log10 (x)	double x;	常用对数函数
double pow(x, y)	double x, y;	幂函数（x的y次方）

③ 赋值运算与自反运算

C语言中的“＝”是一个特殊的运算符，称为赋值运算符。由它组成的赋值表达式“x＝y”意味着计算出右部表达式y的值赋给左部变量x。该表达式的值为x的值。赋值运算符使用起来既方便又简洁。例如，下列两个语句

```
x=1985; printf ( "%d" , x );
```

可简写成一个语句：

```
printf ( "%d" , x=1985 );
```

此外，还可以实现多重赋值。比如，下面三个语句：

```
x=1985; y=1985; z=1985;
```

可简写成一个语句：x＝y＝z＝1985；

C语言还引入了一种自反运算，它的使用格式如下：

〈变量〉〈双目运算符〉＝〈表达式〉

其含义为：

〈变量〉＝〈变量〉〈双目运算符〉（〈表达式〉）

例如，“x＋＝y”的含义为“x＝（x＋y）”。自反运算符包括了＋＝，－＝，＊＝，／＝，％＝，》＝，《＝，＆＝，＾＝和｜＝。

C86中还提供了一个整数交换函数，它的格式如下：

```
iswap ( inta, intb )
```

其中，inta和intb为整型指针参数，该函数等效于：

tmp = * inta; * intb = * inta; * inta = tmp;

即交换inta和intb指向的两个整型量。

④ 其它运算

定义在基本类型上的运算符还有条件运算符、顺序运算符和强制类型转换运算符等。此外，C86还提供了一组字符操作函数。下面分别叙述。

* 条件运算符

条件运算符是一个三目运算符，由它可以组成条件表达式，其格式为：

〈逻辑表达式〉？〈表达式1〉：〈表达式2〉

其中逻辑表达式的类型为整型。该表达式的含义为：当逻辑表达式的值为真（非零）时，取“表达式1”为值，否则取“表达式2”为值。

* 顺序运算符

顺序运算符“，”用来把两个或多个表达式合并成一个表达式。顺序表达式：

〈表达式1〉，〈表达式2〉，…，〈表达式n〉

表示从左至右依次计算各表达式值。顺序表达式的值为“表达式n”的值。例如：

t= 3，t+ 2

的值为5。

* 强制类型转换运算符

强制类型转换运算符的使用格式如下：

（〈类型〉）〈表达式〉

它用来把表达式的类型强制转换为指定的类型。例如，表达式

（int）3.1415926和（int）3.789

的类型都为整型，它们的值也都为3。

* 有关字符操作的系统库函数

C86提供了一组字符类别判断函数和大小写转换函数，如表4-4所示。其中字符类别判断

表4-4 字符操作函数

函数及其类型	参数类型	功能
int isalnum(cc)	char cc;	cc是字母或数字吗？
int isalpha(cc)	char cc;	cc是字母吗？
int isdigiter(cc)	char cc;	cc是数字吗？
int isascii(cc)	char cc;	cc是基本ASCII字符（0x00—0x7f）吗？
int iscntrl(cc)	char cc;	cc是控制字符（0x00—0x1f，0x7f）吗？
int isprint(cc)	char cc;	cc是可打印字符（0x20—0x7e）吗？
int isspace(cc)	char cc;	cc是空格符(0x20，‘\t’，‘/n’)吗？
int ispunct(cc)	char cc;	cc是标点符号吗？
int islower(cc)	char cc;	cc是小写字母吗？
int isupper(cc)	char cc;	cc是大写字母吗？
int tolower(cc)	char cc;	把cc变换为小写字母返回
iut toupper(cc)	char cc;	把cc变换为大写字母返回

函数的整型值为1（真）时，表示给定的字符属于指定的字符集合。

2. 指针

（1）指针及其表示

指针是C语言提供的另一种数据类型，指针变量的值为另一个变量(对象变量)的地址。例如，当ptr为指向变量i的指针且i的地址为g、值为v时，它们之间的关系如图4-2所示。

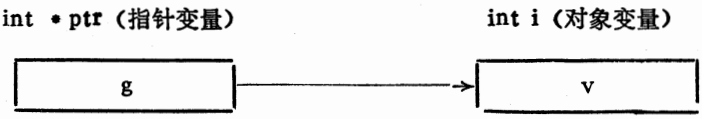


图4-2 指针变量与对象变量之间的关系

C语言中的指针允许指向任何类型的对象变量，包括指向其它指针变量。它通常用来间接地访问对象变量，也可以把某种类型数据的地址传递给某个函数。指针是C语言的重要特点之一，它可以提高程序设计的灵活性。

C86的指针数据在内存中占2个字节或4个字节，编译小模式下占用2个字节，编译大模式下占用4个字节（C86的编译模式请参见第四节）。小模式下的2个字节表示指定段内位移量，大模式下的4个字节包括了2个字节的段地址和2个字节的位移量。

（2）指针的说明

指针说明的格式为：

〔<存贮类>〕<类型> *<标识符>〔=<初值表达式>〕

其中，标识符是指针变量名，存贮类和类型均是说明该指针所指向的对象变量。指针在说明时还可以定义初值，其初值表达式的类型是整型或长整型。例如，下列都是指针说明语句：

```
int    *iptr;
double  *dptr = 0x1000;
auto   char *cptr;
```

（3）指针的运算

C语言可以对指针进行运算。与指针有关的运算有以下几类：

* 取地址运算

使用取地址运算符“&”的格式通常为：

<指针变量> = &<对象变量>

它的含义是把对象变量的地址赋给指针变量。当然，这两个变量的类型应该匹配。

* 取内容运算

取内容运算符“*”的常用格式如下：

<变量> = *<指针变量>

它的含义为把指针变量所指向的对象变量的值赋给左部变量。取内容运算是使用指针的基本手段，它与取地址运算互为逆运算。例如：

ptr = &var1; var2 = *ptr;

等效于

var2 = var1;

* 关系运算

如果两个指针变量指向同一数组的元素（数组请参见本节第3段），则这两个指针可以使用关系运算进行比较。例如，表达式

ptr1 < ptr 2

当ptr1指向的数组元素在ptr 2 指向的元素之前时为真。

此外，任何指针还可以同NULL（零）作相等或不相等比较。因为NULL常为指针的初值，所以这种比较可以判别指针的初值是否改变。

* 算术运算

指针变量可进行的算术运算有增量减量运算和减法。指针变量还可以与整型量进行加减以及加减自反运算，也可用整型量赋初值。下面就介绍这几种指针运算。

指针的增量运算“++”和减量运算“--”的含义与整型量的增量减量运算不完全相同。例如，ptr++表示增加ptr，使它指向原对象元素的下一个元素，即ptr的值增加n。这个n称为比例因子，它等于对象变量在内存中存放的长度。例如，整型量的比例因子为2，长实型量的比例因子为8等。

当两个指针指向同一个数组的元素时，这两个指针可进行减法运算，其结果为两者之间的元素个数，结果类型为整型。下面是指针减法运算的一个例子。

例4-7　字符串长度

功能：打印给定字符串的长度。

程序与运行结果：

```
/* EX4-7 String length */
main()
{
     int n;

     n = strlen1("This is a string!");
     printf(" n = %d\n", n);
}
strlen1(s)  /* return the length of string s */
char *s;
{
     char *p = s;

     while(*p != '\0')
          p++;
     return(p - s);
}

A>EX4-7
 n = 17
```

说明：

* 函数strlenl（）中定义为一个指针p，它的初值为s第1个元素的地址。这句说明语句等效于下列语句：

```
char * p = &s〔0〕
```

* 执行while语句后，指针p指向s的最后一个元素。这时p－s得到的值即为字符串s的长度。

指针变量与整型变量进行加减、加减自反以及赋值运算所得到的结果类型都是指针类型。其中，加减和加减自反运算中的整型运算分量在运算时自动乘上一个比例因子n。例如，ptr＋＝i表示增加ptr的值使其指向原对象变量后的第i个元素，ptr的值（地址）增加i＊n。但是，如果使用赋值运算将一个整型量（或长整型量）赋给一个指针变量，该整型量不乘以比例因子。例如，执行语句

```
ptr = 0x40;
```

后，ptr的值为0x40。

C86还提供了两个地址转换函数，它们用于8088的20位绝对地址与32位大模式指针之间的转换。这两个函数的格式及含义如下：

* unsigned char * abstoptr（addr）	把无符号长整数addr（20位绝对地址）转换成大模式指针返回。
* unsigned long ptrtoabs（ptr）	把大模式指针ptr的值转换成20位绝对地址返回，ptr的对象类型为无符号字符。

3．数组

（1）数组

数组类型是一种复合类型，它表示一组有序的同类数据。C语言允许数组的元素为任何类型的变量，当数组的元素为数组时称为多维数组。C86的维数不允许超过7。

数组在使用前必须说明，说明语句的格式如下：

〔＜存贮类＞〕＜类型＞＜标识符＞‘〔’〔＜常量表达式＞〕‘〕’〔＝＜初值表＞〕；

其中，标识符表示数组名；存贮类和类型用来说明数组元素的类型，存贮类可以是auto，也可以是static或者extern；常量表达式用来指明数组中元素的个数（C语言的数组为静态数组）；初值表用来给出数组的初值。数组说明语句

```
int  array[10];
```

说明一个名为array的数组，它由10个整型元素组成。数组元素使用下标变量访问，array的下标范围为0—9。例如，array〔2〕表示该数组的第3个元素。

C语句允许在数组说明时对静态数组和外部数组赋初值，初值的格式为：

＜常量表达式＞｜‘{’＜常量表达式＞{，＜常量表达式＞}‘}’

它可以对数组的部分元素或全体元素赋初值。下面是带初值说明的数组说明语句：

```
static  int array[10]= 4 ;
static  double  data[3]={10.5, 2.5, 1.0};
```

第1例仅定义array〔0〕的初值；第2例是对整个数组赋初值。

多维数组定义初值的格式有两种，例如下列两式的含义完全相同：

```
static int matrix[2][3]={{1, 0, 0}, {2, 3, 0}};
static int matrix[2][3]={1, 0, 0, 2, 3, 0};
```

它们都使二维数组matrix的初值为：

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \end{pmatrix}$$

此外，第1式还可以简写为：

```
static int matrix[2][3]={{1}, {2, 3}};
```

（2）数组的使用

C语言不允许对除字符数组以外的整个数组进行运算，只允许对数组中的单个元素（下标变量）进行处理。下标变量的格式为：

〈数组名〉'〔'〈下标表达式〉'〕'

例如，array[0]表示数组array中的第1个元素。

例4-8　求和

① 功能：求出数组n中每行元素的和并打印。

② 程序与运行结果：

```
/* EX4-8 Sum up each row of the array */
double n[3][6] = {
        {54.6, 5.0, 8.0, 2.0, 0.0, -3.2},
        {42.8, 5.0, 8.0, 1.0, -10.0, -4.0},
        {68.7, 5.0, 8.0, 0.0, -5.0, -9.0}};
main()
{
    int i = -1, j;
    double total;

    while(++i < 3)
    {
        j = 0; total = 0.0;
        while(j < 6) total += n[i][j++];
    }   printf("Total %d: %f\n", i, total);

}

A>EX4-8
Total 0: 66.400000
Total 1: 42.800000
Total 2: 67.700000
```

③ 说明：

* 数组n为定义在main（）函数外的外部数组，可以在说明语句中定义初值。

* while 重复语句的重复条件表达式中的增量运算符为前缀形式，所以i 的初值应定义为－1。

在C语言中，下标变量中的方括号也可以看作是一个运算符。例如，array〔i〕的含义为：

*（array+i）

其中“array”表示数组array的基地址，即数组第1个元素array[0]的地址。因此，除了可以使用下标变量访问数组元素之外，还可以使用指针访问数组元素。

（3）指针与数组

在实际使用中，指针所指向的对象变量，很少是基本类型，而大多数是导出类型。例如，可以使用指针来指向数组中的某元素。 因为数组在内存中是连续存放的，所以使用基地址加上一定的位移量可以求出数组中某个元素的地址，再使用取内容运算访问这个元素就十分方便。

除了可以利用指针访问数组元素之外，还可以利用指针变量表示数组。例如，使用说明语句

char *charptr; 和int *a[ ];

分别代替说明语句

char charptr[]; 和int a[ ][ ];

这两种形式在程序中可以任意使用，下面就是使用字符指针来访问字符数组的一个例子。

例4-9 字符串的反序输出

① 功能：把给定的字符串从后至前一个字符一个字符地输出。

② 程序与运行结果：

```
/* EX4-9 Print a string backward */
#include "stdio.h"
main()
{
    static char byebye[] = "That's all, folks!";
    char *charptr = &byebye[18];

    while(--charptr >= &byebye[0])
        putchar(*charptr);
    putchar('\n');
}
```

```
A>EX4-9
!sklof ,lla s'tahT
```

③ 说明：

* 程序中使用了一个字符数组（字符串）byebye〔 〕和一个指向它的字符指针charptr。其中byebye[ ]的初值为“That's all, folks! ”，charptr的初值为byebye[]中最末一个字符的地址，根据字符串的内部表示，该地址中存放字符串的尾标'/0'。

* 在while重复语句中，charptr首先减1再与byebye〔 〕的始址& byebye[0]相比较。若不小于始址就输出其所指向的字符，否则表示字符串已输出完毕，并结束循环。

* 一维字符数组（字符串）的初值是用双引号括起来的一串字符。例如，本例中就使用了一个包含初值定义的字符串说明语句：

```
static char byebye[ ]= “That's all, folks! ”;
```

（4）字符串

一维字符数组是一种使用较为频繁的数据类型，它是以空字符'/0'结尾的一串字符，所以也称为字符串。字符串常量是一串由双引号括起来的字符，例如：

“That is a string! ”

就是一个字符串常量。

C86提供了一组较为丰富的字符串处理函数，如表4-5所示。使用这些函数加工字符串十分方便。

表4-5　字符串处理函数

函数及其类型	参数类型	功能及举例
unsigned strlen(str)	char * str;	返回字符串str的长度
char * strcpy(s1, s2)	char * s1, * s2;	把s2的内容复制到s1中并返回指向s1的指针
char * strncpy(s1, s2, n)	char * s1, * s2; unsigned n;	把s2的前n个字符复制到s1并返回指向s1的指针
char * strcat(s1, s2)	char * s1, * s2;	把s1和s2并置起来放入s1并返回s1的指针，例如 strcat(“ab”, “cd”) = = “abcd”
char * strncat(s1, s2, n)	char * s1, * s2; unsigned n;	把s2的前n个字符添加到s1的末尾。
int strcmp(s1, s2)	char * s1, * s2;	比较s1与s2，如果s1小于s2返回－1；s1等于s2返回0；s1大于s2则返回1。例如strcmp (“ab”, “a”) =1
unsigned char * upper(s)	unsigned char * s;	把s中的小写字母全部转换为大写字母返回，例如upper (“abc”) = = “ABC”
unsigned char * lower(s)	unsigned char * s;	把s中的大写字母全部转换为小写字母返回
char * index(s, cc)	char * s, cc;	在s中查找字符cc，如找到返回指向该字符的指针，否则返回0
char * rindex(s, cc)	char * s, cc;	在s中从后向前查找，若找到cc则返回指向该字符的指针，否则返回0

下面是利用strcmp ( ) 函数写的一个排序函数。

例4-10　排序函数

① 功能：把字符串v[0]…v[n－1]按递增顺序进行排序。

② 程序（以函数形式给出）：

```
/* EX4-10 Sort string v[0] ... v[n-1] into
** increasing order */
sort(v, n)
char *v[];
int n;
{
    int gap, i, j;
    char *temp;

    for(gap = n/2; gap > 0; gap /= 2)
        for(i = gap; i < n; i++)
            for(j = i - gap; j >= 0; j -= gap)
            {
                if(strcmp(v[j], v[j + gap]) <=
                    break;
                temp = v[j];
                v[j] = v[j +gap];
                v[j + gap] = temp;
            }
}
```

③ 说明：

* 本例中sort（）函数的参数v是一个字符指针数组，它用来表示n个字符串v[0]…v[n－1]。

* for语句是C语言的一种重复语句。例如，第1个for语言含义为将变量gap赋初值n/2，如果满足条件gap＞0则重复执行其后的复合语句，然后再把gap除以2，如此重复直至不满足条件gap＞0为止。

* break为C语言的一个中断语句，其功能为退出当前for重复语句。

* temp为一个字符指针，用来指向一个存放中间结果的字符数组。

4．结构与联合

（1）结构及其说明

C语言中的结构（structure）是一种复合类型，它表示一组不同类型的数据。它常用来扩充C语言的数据类型并对详细的结构说明进行缩写。

结构说明的格式如下：

struct〔＜结构名＞〕'{'{＜成员类型＞＜成员名＞}'}'〔＜结构变量＞〕{,＜结构变量＞}〔=＜初值＞〕〕；

其中，结构名表示说明了的结构，以后可作为详细结构说明的缩写来使用；成员名用来标识结构的成员，供按名访问结构成员；结构变量则为被定义成该种结构类型的变量。

例如，一个表示日期的结构date可说明如下：

```
struct date
{
    char *monthname;
    int day;
    int year;
}
```

该结构可用来表示日期类型的数据。例如，表示出生日期的结构变量birthdate和当前日期结构变量currentdate，可以使用下列说明语句进行说明：

```
struct date birthdate, currentdate;
```

其中，date作为日期结构的缩写。此外，还可以直接使用详细结构说明语句来说明结构变量。例如，上列说明birthdate和currentdate的过程可以并成一步完成：

```
struct
{
    char *monthname;
    int day;
    int year;
} birthdate, currentdate;
```

静态结构和外部结构在说明时也可以定义初值，如

```
static struct date birthdate={"may", 10, 1965};
```

由于实现上的原因，C86目前还不能对结构定义初值，因此使用时应予以注意。

结构中的分量也可以是一个结构，例如下面的结构中就含有一个结构类型的分量：

```
struct student
{
    char *name;
    char sex;
    struct date birthdate;
};
```

它可以用来表示一张学生登记卡。

（2）结构的使用

C语言一般不允许对整个结构进行运算，使用结构时可以通过分量名访问结构中的某个分量。与使用结构有关的运算符有以下三个。

① 取地址运算

取结构首地址运算的使用格式如下：

```
&〈结构变量〉
```

使用该运算求出的结构首址可以赋给一个结构指针，以便用结构指针访问结构的成员。

② 取成员运算

使用取成员运算是访问结构成员的一种基本手段，它的使用格式如下：

〈结构变量〉.〈成员名〉

例如，要访问结构birthdate中的成员year可通过分量名birthdate.year来实现。

③ 指向成员运算

使用结构成员的另一种方法是通过使用结构指针以及指向成员运算来完成。这种使用方法的格式如下：

〈结构指针〉—〉〈成员名〉

这里结构指针指向要访问的结构。该式等效于：

(*〈结构指针〉).〈成员名〉

如果结构指针dateptr指向birthdate，那末可用

dateptr—〉year

访问birthdate.year。该式等效于：

(*dateptr)·year

下面是利用结构指针访问结构成员的一个程序片段：

```
{
    struct date *dateptr=&birthdate;
    birthdate.year=1966;
    printf("%d\n", dateptr->year);
}
```

(3)结构数组

实际使用中常常把若干个相同的结构组成一个数组，这就是所谓的结构数组。例如，某班的学生登记表可以使用一个结构数组(见例4-11)来表示：

```
struct student class[20];
```

例4-11　学生登记表的统计

① 功能：打印1966年出生的学生的人数，并统计全班共有男女生各多少人。

② 程序与运行结果：

```
/* EX4-11 Print name */
#include "stdio.h"
#define BSIZE 128

main()
{
    struct student
    {
        char sex;
        char *name;
        int birthyear;
    } class[5];
    char bf[BSIZE + 2], *fp[3];
    int i;
```

```
    int count66 = 0, countm = 0, countf = 0;

    printf("NAME,SEX(m, f),BIRTH YEAR\n");
    for(i = 0; i < 5; i++)
    {
        basicget(stdin, bf, BSIZE, fp, 3);
        class[i].name = fp[0];
        class[i].sex = *fp[1];
        class[i].birthyear = atoi(fp[2]);
        if(class[i].birthyear == 1966)
            ++count66;
        if(class[i].sex == 'm')
            ++countm;
        else ++countf;
    }
    printf("male : %d\n",countm);
    printf("female : %d\n", countf);
    printf("birthyear = 1966 : %d\n", count66);
}
```

```
A>EX4-11
NAME,SEX(m, f),BIRTH YEAR
wang,f,1965
li,m,1966
qi,f,1966
wu,m,1967
zhang,f,1966
male : 2
female : 3
birthyear = 1966 : 3
```

③ 说明：

* 程序中使用了一个结构数组class[ ]，它表示一张学生登记表。
* basicget( )是一个系统函数，它用来读入类似BASIC语言中记录格式的数据。本例中用来读入一行以逗号分隔的数据。
* atoi( )是一个系统库函数，它用来把字符串表示的数转换成整型数。

（4）联合

联合（union）是一种类似于结构的复合类型。与结构的主要区别在于，在任何给定的时刻，只有一个成员驻留在联合中，而结构是所有成员一直驻留在结构中，通常使用联合比使用结构更能节省一些存贮空间，但访问的速度比较慢。

联合的说明格式如下：

union[〈联合名〉] '{' {〈成员类型〉〈成员名〉} '}' [〈联合变量〉]{,〈联合变量〉}[=〈初值〉]；

例如：union{double d; float f[2]; }u;

说明u是一个联合变量，它的值在某一时刻或者一个长实型数，或者是两个实数组成的数组。

联合中成员的长度最好相同，以便成员之间的相互覆盖。当成员的长度不同时，采用最长的成员在内存中占的字节数存放联合。

5. 类型定义与运算符的优先级

（1）类型定义

C语言除了提供上述基本类型和导出类型外，还提供了一种扩充新的数据类型的设施——类型定义。类型定义可以使用C语言的标准类型和已定义了的类型定义新的数据类型。

类型定义的格式为：

typedef 〈老类型名〉〈新类型名〉{，〈新类型名〉}

其中老类型名可以是标准类型，也可以是已定义了的类型。

定义新类型的一些例子如下：

```
typedef char *STRING; /*字符串*/
typedef struct { float real, imag; } COMPLEX; /*复数*/
typedef unsigned long DWORD; /*双倍长机器字（32位）*/
typedef DWORD DWORDARRAY[]; /*双字数组*/
typedef DWORDARRAY *DWAPTR; /*双字指针数组*/
```

其中最后两个例子是使用已定义的类型名来定义新的类型。

（2）运算符的优先级与结合特性

上面已经介绍了C语言的数据类型及其运算，现在开始讨论C语言运算符的优先级和它们的结合特性。

在计算含有多个不同运算符的表达式时为了不发生二义性，给运算符规定了一个属性，这就是运算符的优先级。它可以决定运算符的运算顺序，优先级高的运算符先运算，低的后运算。例如，计算表达式

```
a+b*c
```

时，由于乘法的优先级高于加法，因此一定是先做乘法后做加法。

运算符的结合性是规定连续的几个相同优先级的运算符的计算顺序。结合性为从左至右时，计算从左至右逐个进行；结合性为从右至左时，计算从右至左逐个进行。例如，因为加法的结合性是从左至右，所以表达式：

```
a+b+c
```

的计算顺序为从左至右，即该表达式等价于：

```
(a+b)+c
```

但是，因为赋值运算符的结合特性为从右至左，所以表达式：

```
a=b=c
```

等效于：

```
a=(b=c)
```

C语言运算符的优先级与结合特性如表4-6所示。

表4-6　运算符的优先级与结合特性

优先级	类　型	运　算　符	名　　称	结合特性
15	初等运算	( ) [ ] ·，—›	函数参数表 数组下标 结构取成员	从左至右
14	单目运算	! ～ ++，-- - (〈类型〉) * & sizeof	逻辑非 位串补 增量与减量 算术负 强制类型转换 取内容运算 取地址运算 判断类型大小	从右至左
13	双　目	*，／，%	乘、除与取模	从左至右
12	算术运算	+，-	加和减	
11	移位运算	<<，>>	算术左移与右移	从右至左
10	关系运算	〈，〈=，〉，>=	小于，小于等于，大于，大于等于	从左至右
9		==，!=	等于与不等于	
8	位串运算	&	按位与	从左至右
7		∧	按位异或	
6		|	按位或	
5	逻辑运算	&&	逻辑与	从左至右
4		||	逻辑或	
3	三目运算	?：	条件运算	从右至左
2	赋值运算与自反运算	= +=，-=，*=，／=，%=，《=，》=，&=，∧=，|=等	赋值运算 自反运算	从右至左
1	顺序运算	,	顺序运算	从左至右

第三节　C语言的程序结构

C语言程序由一个或多个程序文件组成，这些文件中包含并且只包含一个main（）函数。每个程序文件由类型定义、数据说明和函数说明组成，它们的顺序可以任意。程序文件的格式如下：

{〈类型定义〉|〈外部数据说明〉|〈函数说明〉}

类型定义主要用于扩充C语言的标准数据类型。类型定义的格式为：

typedef 〈老类型名〉〈新类型名〉{，〈新类型名〉}

外部数据说明描述程序文件使用的外部数据以及局部于程序文件的静态变量。外部数据说明的格式为：

〔extern | static〕〈类型〉〈变量〉〔=〈初值〉〕{，〈变量〉〔=〈初值〉〕}

其中，存贮类extern表示说明的变量是其它程序文件中已定义的外部变量，这时不允许赋初值；static表示说明的变量是局部于本程序文件的静态变量；当外部数据说明前没有存贮类时，即为外部变量的定义，使用它定义的外部变量可以在其它程序文件中使用。

程序文件中的函数说明是C语言程序的主体部分，用来描述程序的算法以及函数内部所使用的数据。算法的描述是通过语句、系统库函数调用以及自定义函数调用等实现的，本节将对它们分别加以介绍。局部数据的定义是通过局部数据说明来完成的。下面是函数说明的格式：

〔extern | static〕〔〈类型〉〕〈函数名〉(〈参数表〉)〔〈参数说明〉〕
〈复合语句〉

其中函数中使用的局部变量在“复合语言”中加以说明。关于函数说明的详细使用方法将在本节的第2段中介绍。

1. 语句

C语言的语句可以分成说明语句、表达式语句、程序控制语句和复合语句四类。其中说明语句已作过介绍，现对其余三类语句作一说明。

（1）表达式语句

C语言的任意表达式加上分号后就构成一个表达式语句，其格式为：

〈表达式〉；

其中，由赋值表达式加分号构成的赋值语句以及由函数调用表达式加分号构成的函数调用语句是最常用的。

C语言的表达式语句表达能力很强，使用也十分方便。下面是几个表达式语句：

```
n=k++;
plot(x, y, z);
++k;
n=4, t=n+2;
a&&b==a? (b?1:0):0;
```

仅由一个分号组成的语句称为空语句，这是表达式语句的特例。

（2）复合语句

复合语句是由多个说明和语句组合而成的一种语句，也称为分程序。其格式为：

‘{’ {〈类型定义〉 | 〈局部数据说明〉}{〈语句〉} ‘}’

其中，局部数据说明描述本复合语句要使用的局部数据和外部数据，类型定义用来定义复合语句中要使用的新数据类型。例如：

```
{
```

```
    typedef unsigned  WORD;
    WORD  i=4;
    extern  j;
    printf("%d\n", i+j);
}
```

是一个复合语句，其中外部变量j是在该复合语句外定义的。

在C语言中，任何可以出现语句的地方，都可以使用复合语句。

（3）选择语句

C语言有两种选择语句，第1种是条件语句，第2种是情况语句，这两种语句都可以根据某表达式的值选择执行不同的程序段。

条件语句的格式为：

```
if(〈表达式〉)
    〈语句〉
〔else〈语句〉〕
```

其中，“表达式”为选择条件，当它不为0时执行随后的语句，否则执行else后的语句或空语句。当选择条件为“e！＝0”这种形式时，可简写为“e”。

条件语句关键字else后面的语句也允许是条件语句，这时的条件语句称为多重条件语句。在多重条件语句中，C语言规定else与它前面最接近的if配对。

C语言的情况语句是一种多路选择机构，其格式为：

```
switch(〈表达式〉)
    '{'{〈类型定义〉|〈局部数据说明〉}  {{case〈常量表达式〉:
  |default:}〈语句〉}'}'
```

但是实际使用switch语句的格式常为下列形式：

```
switch(〈表达式〉)
{
   case  〈常量表达式〉:{〈语句〉}
   case  〈常量表达式〉:{〈语句〉}
              ⋮
   default:{〈语句〉}
}
```

它的含义为当表达式的值等于某个表达式时，执行以下的所有语句。例如运行程序

```
main(  )
{
  int i=2;
  switch(i)
  {
    case 1: printf("case  1\n");
    case 2: printf("case  2\n");
    case 3;
```

```
            case 4: printf("case 3 and 4\n");
            default: printf("default\n");
            }
        }
```

后，得到的结果为：

```
case 2
case 3 and 4
default
```

如果希望在执行某个case语句后就退出switch语句，可以在每个case之后加上一个break语句。例如：

```
main( )
{
    int i=2;
    switch(i)
    {
        case 1: printf("case 1\n"); break;
        case 2: printf("case 2\n"); break;
        case 3:
        case 4: printf("case 3 and 4\n"); break;
        default: printf("default\n");
    }
}
```

的运行结果为：

```
case 2
```

（4）重复语句

C语言提供了三个重复语句，它们是while语句、do语句和for语句。这三个语句都可以根据某种条件重复执行一个语句或一个复合语句。

while语句的格式为：

while（〈表达式〉）〈语句〉

它表示当“表达式”为真时反复执行“语句”，直至“表达式”变假时为止。当“表达式”一开始就为假时，while语句等效于空语句。

do语句十分类似于while语句，只是先执行给定的语句，后检验重复条件。do语句的格式为：

do〈语句〉while（〈表达式〉）；

其中“语句”至少要执行一次。

for语句的格式为：

for（〔〈表达式1〉〕；〔〈表达式2〉〕；〔〈表达式3〉〕）〈语句〉

它的含义如下：

〔〈表达式1〉；〕
while（〔〈表达式2〉〕）
‘{’〈语句〉；〔〈表达式3〉〕；‘}’

其中，“表达式1”重复语句的初始化（如循环变量赋初值）；“表达式2”为重复条件；“表达式3”重复后的处理（如修改循环变量）。“表达式2”的缺省值为真，它表示不停地重复。

（5）中断语句

C语言的中断语句主要用来中断正常的程序执行，例如可以中断重复语句、情况语句以及函数。C语言的中断语句有以下四种：

```
break;
continue;
goto〈标号〉;
return（〈表达式〉）;
```

break语句用来跳出最内层的重复语句或退出switch语句，而continue语句则用来终止最内层的重复语句中的执行部分，并继续执行该重复语句，但不象break语句那样要跳出整个重复语句。例如，程序段

```
for（i=0；i<N；i++）
{
    if（a[i]<0）
        continue;
    printf（“%d”，a[i]）;
}
```

输出数组a中的正元素而跳过负元素。但若将其中的continue语句改为break语句：

```
for（i=0；i<N；i++）
{
    if（a[i]<0）
        break;
    printf（“%d”，a[i]）;
}
```

这个程序段的功能为输出正元素，遇到负元素时就跳出这个for语句。

goto语句用来转移到指定标号的语句处继续执行，常用来跳出多重循环。C语言的标号是一个带冒号的标识符，下面是一个使用goto语句的例子：

```
if（a[i]==v）goto found;
    ⋮
found: printf（“%d”，a[i]）;
```

使用goto语句有时会破坏程序的静态结构，应当尽可能避免使用。

return语句用来终止函数的执行并把控制返回到调用该函数的程序。如果return后带有表达式，则将表达式作为函数值返回；如果return后无表达式，那末函数的返回值无意义。

在每个函数的末尾隐含着一个不带表达式的return语句，所以在写函数时，如不需要返回参数，则可将return语句省去。

2. 函数

（1）函数的定义与调用

程序中的任何函数都有两个侧面，即函数的定义与函数的调用。除库函数外，任何函数在调用前都必须定义。函数定义是通过函数说明来进行的，函数说明的格式如下：

〔extern | static〕〔〈类型〉〕〈函数名〉（〔〈参数表〉〕）〔〈参数说明〉〕〈复合语句〉

其中存贮类和类型用来限定函数值的类型。C语言的函数类型只允许是基本类型或指针，它的缺省类型为整型。当函数说明前不给出存贮类时，定义的函数为外部函数。参数表中的形式参数的类型由参数说明描述，如果一个形式参数的参数说明缺省就表示该参数的类型为整型。

函数体是一个复合语句，其中不允许再定义函数。函数体内可以说明该函数使用的局部数据、外部数据以及使用的外部函数。下例是一个函数说明：

```
double sh(x)
double x;
{
    extern double exp( );
    return ((exp(x)-exp(-x))/2.0);
}
```

已定义的函数可以通过函数调用来使用。函数调用可以作为运算分量在表达式中出现，也可以加上一个分号组成调用语句。函数调用的格式为：

〈函数名〉（〈实参表〉）

其中，实参表应与函数说明中的形参表的类型和顺序保持一致（字符和整型可视为同一类型，实型和长实型可视为同一类型），否则会产生意想不到的结果。

在调用一个函数前，必须说明函数的类型。当函数的类型为整型时，说明可缺省。应当注意，即使使用系统库函数，只要其类型不为整型，也必须加以说明。例如，要调用系统库函数exp（），则必须先用下列语句进行说明：

```
extern double exp( );
```

（2）函数的参数

函数的参数为基本类型时，一般都使用赋值方式传递，而对复合类型的参数通常采用指针来进行传递。

返回参数（函数值）为基本类型直接返回，为复合类型时则返回指向它的指针。如果需要返回多个参数，则可利用外部变量传递返回参数，也可以利用指针参数传递返回参数。例如，在下面的程序中，函数addsub（）利用外部变量返回两个参数。

```
int c, d;
main( )
```

```
{
    int  a, b
    a= 4 ; b= 3 ;
    addsub ( a, b ) ;
    printf ( "%d  %d\n" , c, d ) ;
}
addsub ( x, y )
int  x, y;
{
    c=x+y;
    d=x-y;
}
```

如果要利用指针参数传送返回参数，则程序可改写为如下形式：

```
main ( )
{
    int  a, b, c, d;
    a= 4 ;   b= 3 ;
    addsub ( a, b, &c, &d ) ;
    printf ( "%d  %d\n", c, d ) ;
}
addsub ( x, y, cptr, dptr )
int  x, y, *cptr, *dptr;
{
    *cptr=x+y;
    *dptr=x-y;
}
```

C语言中的main ( ) 函数是一个具有特殊地位的函数，它的定义格式通常为：

```
main ( argc, argv )
int  argc; /*参数个数*/
char *argv [] ; /*存放参数的字符指针数组*/
〈复合语句〉
```

其中参数可省略；“复合语句”为用户编制的过程体。C程序在运行时从main ( ) 函数开始执行，它的参数由操作系统的命令行提供。例如，要执行一个名为MYPROG的C程序，如果在命令行中打入命令：

```
A > MYPROG  FILE1.DAT  FILE2.DAT
```

则“c”，“FILE1.DAT”和“FILE2.DAT”就作为参数被程序MYPROG中的main ( ) 函数接收，它们分别存放在argv[0]，argv[1]和argv[2]中，而argc的值则为 3，表示包括标志c在内共有三个参数。

下例中的main ( ) 函数使用了命令行参数。

例4-12　打印命令行参数

① 功能：打印命令行中的参数的个数以及所有参数。

② 程序与运行结果：

```
/* EX4-12 Demonstrate main() function */
#define PR(t, v) printf("%t\n", v)
main(argc, argv)
int argc;
char *argv[];
{
    int i;

    PR(d, argc);
    for(i = 0; i < argc; i++)
        PR(s, argv[i]);
}

A>EX4-12 FILE1.DAT FILE2.DAT
3
c
FILE1.DAT
FILE2.DAT
```

③ 说明：

*** 程序中定义的宏替换PR用来输出一个数据，它有两个形式参数，第1个表示输出的格式，第2个表示欲输出的变量。**

*** 程序中的for语句用来根据参数的个数argc输出所有参数。**

（3）C86的系统库函数

目前，C86提供了130多个系统库函数，它们大体上可分成下列几类：

- *** 类型转换函数**
- *** 算术运算函数**
- *** 字符操作函数**
- *** 字符串处理函数**
- *** 输入输出函数**
- *** 内存管理函数**
- *** 机器级函数**
- *** 程序退出函数**

前四类函数在第二节中已作了介绍，现介绍C语言的几个程序退出函数，其它函数将在本节后几段中陆续叙述。

C86的程序退出函数主要有三个：

- *** exit(v)　　终止程序执行并把v作为出错号返回操作系统。**
- *** _exit(v)　　同exit()函数，但退出前不关闭打开的文件。**
- *** abort(string)　　终止程序的执行，输出string中的信息，并把0x7fff作为出错号返回操作系统。**

例如，下面是使用了abort（）函数的一个例子：

```
main（argc，argv）
int  argc;
char *argv〔〕；
{
        ⋮
    if（argc==1）abort（"Not  found!\n"）；
        ⋮
}
```

3．标准输入输出函数

C语言没有提供输入输出语句，它的输入输出功能是由输入输出函数完成的。本段先介绍C86的标准输入输出函数，下一段再介绍C86的文件输入输出函数。

C语言中有三个标准输入输出设备：stdin（标准输入设备）、stdout（标准输出设备）和stderr（标准出错信息输出设备）。通常stdin定义为键盘，stdout和stderr均定义为显示器。但它们可以使用MS-DOS的I/O重定向功能指向任何设备或文件（参见第一节中的例4-4）。

此外，C语言还提供了一个标准输入输出定义文件stdio·h，它用来定义标准输入输出所要使用的符号常量和宏替换。用户在程序中需要使用标准输入输出函数时，应在程序文件的开头使用编译命令

```
#include  "stdio·h"
```

嵌入这个定义文件。

下面就分别介绍几组常用的标准输入输出函数。

（1）字符输入输出

用于输入输出字符的标准输入输出函数有两个：

```
getchar（）
putchar（c）
```

其中，getchar（）的类型为int；putchar（）的参数c的类型为char。

getchar（）用来从标准输入设备上读入一个字符返回，如果读入的字符为文件结束符（0x1a），则将返回EOF（－1）。函数getchar（）的值为整型量，即输入字符的ASCII码。

putchar（）用来把给定的字符c从标准输出设备上输出。

例如，下列是利用这两个函数写的一个程序：

```
/*echo  input  to  output*/
#include  "stdio·h"
main（）
{
    for（；；）
        putchar（getchar（））；
```

}

它把来自标准输入设备的字符逐个地从标准输出设备上输出。

（2）格式输入输出

函数scanf（）和printf（）可以按给定的格式输入和输出整型、实型、字符和字符串等类型的数据。scanf（）用来输入数据，printf（）用来输出数据，它们的格式为：

scanf（〈格式串〉{，〈指针变量〉}）

printf（〈格式串〉{，〈表达式〉}）

其中，“格式串”用来规定输入或输出的格式；格式串后面由“指针变量”或“表达式”给出的参数，其个数由格式串中转换说明的个数所决定，既可以没有也可以是多个。

格式串可含有多个由“%”引导的转换说明，也可以含有其它字符。对于非转换说明的字符，这两个函数要求按原样输入或输出。转换说明用来说明对应参数的类型及其输入输出格式，它的形式为：

%〔〈格式〉〕〈转换字符〉

其中，“格式”用来限定参数输入输出格式，它的形式和含义如下：

N	输入输出的最小域宽。当实际域宽小于N时以空格填充；当实际域宽大于N时按实际域宽输出。
〔N〕·M	N的含义同上。M表示实型数的小数位数，或字符串的最大输入输效出有字符的个数。
—	参数输出时在域内向左对齐。
*	跳过指定的输入参数。
l	以长整型或长实型形式输出参数。

转换字符用来限定参数的类型，当参数的类型与它不一致时，将参数转换成指定的类型。C86的转换字符及其含义在表4-7中列出，其中转换字符b是C86对标准C语言的扩充。

表4-7　C86的转换字符

转换字符	含义	转换字符	含义
d	十进制整型量	f	小数形式的实型量
o	八进制整型量	e	科学记数形式的实型量
x	十六进制整型量	g	f和e的较短形式的实型量
b	二进制整型量	c	字符
u	无符号整型量	s	以‘\0’结尾的字符串

下面是使用格式串的几个例子。

格式串	表达式	输出
“%10s”	“Hello␣”	␣␣␣␣␣Hello␣
“%-10s”	“Hello␣”	Hello␣␣␣␣␣␣
“%10.5s”	“Hello␣”	␣␣␣␣␣Hello
“%b”	10	1010

“%ld”	1234567891	123456789
“%f”	1982710.5376483	1982710.537648
“%e”	1982710.5376483	1.982711E+006
“%12.2f”	1982710.5376483	⊔⊔1982710.53

scanf( )的参数与printf( )的参数不同，它是指向输入变量的指针，在使用时应当注意。下面是使用scanf( )函数的一个例子：

```
main( )
{
    int  i;
    float  f;
    scanf( “%d  %*d  %f”, &i, &f);
    printf( “%d  %f  \n”, i, f);
}
```

它用来输入一个整数，再跳过一个整数输入一个实数。例如，执行这个程序时，若输入

```
123      456      78.901
```

输出结果为：

```
123      78.900993
```

因为scanf( )中的“%*d”跳过了输入行中的“456”。

现将两个与scanf( )和printf( )密切相关的函数介绍一下。

```
sscanf(〈字符串变量〉,〈格式串〉{,〈参数〉})
sprintf(〈字符串变量〉,〈格式串〉{,〈参数〉})
```

其中，sscanf( )与scanf( )类似，但输入不是来自stdin而是来自式中的“字符串变量”；同样，sprintf( )与printf( )类似，但它的输出不是送往stdout而是送往式中的“字符串变量”。

例如，执行了语句

```
sscanf( “123  456”, “%d  %d”, d1, d2);
```

之后，整型变量d1和d2的值分别为123和456。而执行了语句

```
sprintf(s, “%d %d”, 123, 456);
```

后，字符串变量s的值为“123 456”。

4. 文件输入输出函数

C语言除了可以通过标准输入输出函数对标准输入输出设备进行输入输出操作外，还可以对文件（数据文件和设备文件）进行输入输出操作，这也是通过提供一组系统库函数来完成的。

C语言的数据文件分成流式文件（stream file）和标准文件（regular file）两类。其中流式文件是C语言提供的，通常它由字符序列组成；标准文件是MS-DOS提供的标准文件，它的格式及使用方法由MS-DOS决定。

C语言把所有系统设备均作为设备文件来处理，它们既可以作为流式文件使用，也可以作为规则文件使用。除了标准输入输出设备文件以外，其它文件都需要先打开，然后才能使

用，用毕则必须关闭。

（1）流式文件

① 文件的打开与关闭

流式文件的类型在定义文件stdio.h中用类型定义

```
typedef    char    FILE;
```

定义为FILE。

流式文件在使用前必须使用函数fopen（）打开，fopen（）的使用格式如下：

〈文件变量〉=fopen（〈文件名〉，〈方式〉）；

其中，文件变量和函数fopen（）的类型为FILE指针；“文件名”是字符串变量，它用来表示欲打开文件(或设备)的文件名(或设备名)；“方式”表示打开的文件的访问方式。

fopen（）的功能为按给定方式打开指定的文件，并返回一个文件号（文件指针）以供使用。当文件号为0时，表示文件打开失败。

文件有以下几种打开方式：

“r”　以只读方式打开ASCII码文件。

“w”　以只写方式打开ASCII码文件，如该文件已存在，则将其清除。

“rw”　以读写方式打开一个已存在的ASCII码文件。

“wr”　以读写方式打开一个新的ASCII码文件，如该文件已存在，则将其清除。

“a”　以只写方式打开ASCII码文件，并将文件指针移至文件末尾以便添加。

“ar”　以读写方式打开ASCII码文件，并将文件指针移至文件末尾以便添加。

“br”　以只读方式打开二进制文件。

“bw”　以只写方式打开二进制文件。

“brw”　以读写方式打开一个已存在的二进制文件。

“bwr”　以读写方式打开一个新的二进制文件。

“ba”　以只写方式打开一个二进制文件，并将文件指针移至文件末尾以便添加。

“bar”　以读写方式打开一个二进制文件，并将文件指针移至文件末尾以便添加。

下面是使用函数fopen（）打开文件的一个例子。

```
FILE *fopen( ), *fp1, *fp2, *fp3;
      ⋮
fp1 = fopen( "mydata1.dat", "r" );
fp2 = fopen( "mydata2.dat", "ba" );
fp3 = fopen( "mydata3.dat", "w" );
```

文件使用完毕后，通常要使用fclose（）函数进行关闭处理，其调用格式如下：

〈整型变量〉=fclose（〈文件号变量〉）；

它表示把与指定文件号对应的那个文件关闭。其中“整型变量”为文件关闭的状态，当它的

值为－1时，表示关闭失败。

② 文件的定位

文件在读写之前有时要指定读写位置，与流式文件定位的有关函数有两个；即

ftell（〈文件号变量〉）

fseek（〈文件号变量〉，〈位移置〉，〈方式〉）

其中，ftell（ ）用来取得指定文件的当前读写位置（读写指针的位置），返回参数的类型为长整型，单位为字符。fseek（ ）用来设置读写指针，并返回其当前位置（负数表示出错）。fseek（ ）中的方式有以下几种：

0 读写指针定位在从文件起始位置开始算起指定位移量处，位移量必须为正数。

1 读写指针从当前读写位置移动给定的位移量。

2 读写指针定位在从文件末尾开始倒退指定位移量之处。

例如，当某个长400个字符的文件的读写指针位于100处时，下列三个语句都可以将读写指针移至200：

```
fseek（fp，200，0）；
fseek（fp，100，1）；
fseek（fp，201，2）；
```

表4-8 流式文件的读写函数

函数及其类型	参数类型	功能
int fgetc(stm)	FILE * stm;	从指定文件取得一个字符返回
int fputc(c, stm)	char c; FILE * stm;	把字符c输出到指定文件
int ungetc(c, stm)	char c; FILE * stm;	把c退回到指定的文件中去以供下次使用（仅允许退回一个字符）
char * fgets(s, n, stm)	char * s; unsigned n; FILE * stm;	把指定文件中读入的n个字符放入s中
int fputs(s, stm)	char * s; FILE * stm;	把字符串s输出到指定的文件
int getw(stm)	FILE * stm;	从指定文件中读入一个整数返回
int putw(i, stm)	int i; FILE * stm;	把整数i输出到指定的文件
int fscanf(stm, f, …)	FILE * stm; char * f;	以f为格式从指定文件中输入数据
int fprintf(stm, f, …)	FILE * stm; char * f;	以f为格式输出数据到指定的文件
int basicget(stm, buf, n, fldptr, fldcnt)	FILE * stm; unsigned char * buf; * fldptr [] ;	读一个BASIC语言格式的记录
int fflush(stm)	FILE * stm;	把所有缓冲区中数据写盘，以防止数据被破坏
int fread(buf, size, n, stm)	char * buf; unsigned size, n; FILE * stm	从指定文件中读入长度为size的n个项放入缓冲buf
int fwrite(buf, size, n, stm)	char * buf; unsigned size, n; FILE * stm;	把buf中的长为size的n个项写入指定的文件

③ 文件的读写

文件打开以后就能使用读写函数进行读写，读写可以从文件的当前读写位置开始，也可以从用fseek（）指定的任意位置开始。

C86的读写函数如表4-8所示，下面仅介绍较普遍使用的字符、字符串以及格式输入输出函数。

字符输入输出

首先讨论的两个文件输入输出函数是fgetc（）和fputc（），它们用来从文件中读入一个字符或把一个字符写入到一个文件中去。

fgetc（）的常用格式如下：

```
c=fgetc(fp);
```

从文件读入的字符放在c中，当读入的字符为文件结束符0 xla时，fgetc（）返回的值为EOF。

fputc（）的作用与fgetc（）相反，它用来把一个字符写入指定的文件，其常用格式为：

```
fputc(c, fp);
```

下面是使用fputc（）的一个实例。

例4-13　建立文件

功能：建立一个名为mydata的文件，并将键盘上输入的字符存入该文件，当键盘上输入ctrl-z时关闭该文件。

程序：

```
/* EX4-13 Echo input to a file */
#include "stdio.h"
main()
{
    extern FILE *fopen();
    extern int fputc(), fclose();
    FILE *fp;
    char c;

    fp = fopen("mydata", "w");
    do
    {
        c = getchar();
        fputc(c, fp);
    }
    while(c != EOF);
    fclose(fp);

}
```

说明：

程序中首先对要使用的三个文件输入输出函数加以说明，然后再说明一个文件指针变量fp和一个字符变量c。

程序的执行部分首先打开文件mydata.dat，然后执行一个do重复语句把标准输入设备上输入的字符送入fp指向的文件。当输入字符为ctrl-z时结束重复语句，并使用fclose（）函数关闭文件。

字符串输入输出

文件的字符串输入输出是通过fgets（）和fputs（）这两个函数来完成的。它们的使用格式如下：

fgets（〈字符串变量〉，〈字符串长度〉，〈文件号变量〉）

fputs（〈字符串变量〉，〈字符串长度〉，〈文件号变量〉）

其中，fgets（）从指定文件中读入一行以‘\n’或EOF结尾的字符串，当欲读入的字符串的长度大于设定的长度n时，仅读入前n－1个字符，读入的字符串以‘\0’结尾。fputs（）将给定的字符串输出到指定的文件，并输出一个新行符‘\n’。

例如，下面是使用这两个函数的一个程序：

```
#include "stdio.h"
main ( )
{
      extern char * fgets ( ) ;
      char buf[255]
      if ( fgets ( buf, 255, stdin ) = = 0 )
            printf ( "EOF or ERROR" ) ;
      else fputs ( buf, stdout ) ;
}
```

格式输入输出

文件格式输入输出函数的调用格式如下：

fscanf（〈文件号变量〉，〈格式串〉，{，〈参数〉}）

fprintf（〈文件号变量〉，〈格式串〉，{，〈参数〉}）

它们的使用方法与标准输入输出的格式输入输出函数相似，这里就不再赘叙了。

④ 文件的出错处理

在调用fopen（）和fclose（）等函数时，如果返回参数值等于或小于0，则表示文件操作有错，于是可以使用函数

ferror（〈文件号变量〉）

求出出错号（负数）。如返回值为0，则表示在进行文件操作时没有出现错误标志。

错误标志可以使用函数

clearerr（〈文件号变量〉）

进行清除。

（2）标准文件

标准文件是可以使用MS-DOS操作系统标准读写方式访问的数据文件，它的使用方法与流式文件的使用方法类似。限于篇幅这里不再详述。有关标准文件的输入输出函数如表

4-9所示。其中建立和打开文件的方式mode可以取值0，1，2，4，5和6，它们分别对应于流式文件的方式“r”，“w”，“rw”，“br”，“bw”和“brw”。

表4-9　标准文件的输入输出函数

函数及其类型	参数类型	功能
int creat(f, mode)	char * f; unsigned mode;	建立文件名为f的文件并返回分配的文件号（0—15）
int open(f, mode)	char * f; unsigned mode;	打开文件名为f的文件并返回文件号
int close(fd)	int fd;	关闭文件号为fd的文件
unsigned long ltell(fd)	unsigned fd;	返回当前文件读写指针的位置
long lseek(fd, offset, base)	int fd, base; long offset;	把读写指针移到指定的位置（类似fseek（ ）函数）
int read(fd, buf, cnt)	unsigned fd, cnt; char * buf;	从指定文件中读入cnt个字符放入buf，返回实际读入的个数
int write(fd, buf, cnt)	unsigned fd, cnt; char * buf;	把buf中的cnt个字符写入指定的文件

（3）设备输入输出

在C语言中输入输出设备都是作为文件来进行处理的，它们称为“设备文件”。常用的设备文件及其文件名如下：

CON：或KYBD：　　键盘
CON：或SCRN：　　显示器
PRN：或LPT1：　　打印机
AUX：或COM1：　　异步通讯器

以下是通过设备输入输出函数使用打印机的一个例子。

```
main ( )
{
    FILE * fopen ( ), * fp;
    fp = fopen ( "PRN:", "w" );
    fprintf ( fp, "Hello! \n" );
    fclose ( fp );
}
```

（4）目录与文件操作

C86提供了一组MS-DOS命令级的目录和文件操作函数，如表4-10所示。它们的功能与对应的MS-DOS命令相似。例如，程序

```
main ( )
{
    mkdir ( "\bin" );
    cddir ( "\bin" );
```

}

用来建立一个子目录\bin，并将当前目录改为\bin。

表4-10　目录与文件操作函数

函数及其类型	参数类型	功能
int mkdir(Path)	char *path;	建立一个子目录
int rmdir(path)	char *path;	删除一个子目录
int cddir(path)	char *path;	改变当前目录
int rename(f1, f2)	char *f1, *f2;	把文件f1改名为f2
int unlink(filename)	char *filename;	删除文件
unsigned char *makefnam (s1, s2, s3)	unsigned char *s1, *s2, *s3;	利用s1和s2构成一个文件名放入s3

5．存贮管理与机器级函数

（1）存贮管理

C语言允许用户程序在运行时自行管理部分内存区域，这块内存区域是存放动态数组的堆（heap）（见第四节的图4-5），特别是在C86的大编译模式下，用户可以管理大于64KB的内存区域。

C86提供了一组内存管理函数，它们可以用来申请和归还一块内存空间。这些函数如表4-11所示。

表4-11　内存管理函数

函数及其类型	参数类型	功能
char *alloc(size)	unsigned size;	在堆中分配一块长size字节的存贮区域并初始化，返回存贮区域的指针，大模式下size不能大于65516B
char *malloc(size)	unsigned size;	功能同alloc（）函数，但不初始化存贮区
char *calloc(n, size)	unsigned n, size;	申请n项长size字节的内存并返回存贮区的指针，总长度不允许超过65512B
char *realloc(oldp, size)	char *oldp; unsigned size;	改变存贮区的大小，oldp指向旧存贮区，size表示新存贮区的大小，返回新存贮区指针
char *sbrk(size)	unsigned size;	申请一块内存，必要时调整整个内存分配以增加堆的大小
int free(ptr)	char *ptr;	归还动态申请的存贮区
unsigned coreleft( )		返回栈区内自由区的大小

其中较常用的alloc（）函数、calloc（）函数和free（）函数分别叙述如下。

* alloc（size）

该函数用来分配一块大小为size字节的内存区域，把它初始化为0，并返回指向该存贮区的指针。在大编译模式下，一次分配到的存贮区的大小可达65516字节并能多次申请。因此在C86的大模式下，所使用的数据区只受存贮容量的限制，并可以大于64KB。但该函数在没有足够的存贮空间分配时，则返回0。

* calloc（num，size）

calloc（）函数用来分配一块num个字段，每个字段长size字节的一块存贮区域，将它初始化为0，并返回指向它的指针。在大模式下，一次申请的内存量不能大于65516字节，但总申请量仅受存贮容量的限制。如果申请得不到满足则返回0。

* free（ptr）

归还一块动态申请的存贮区。

例4-14　动态数组

功能：申请一个长255个字节的动态数组，从标准输入设备上读入一个字符串放入这个动态数组，并将它从标准输出设备上输出。

程序与运行结果：

```
/* EX4-14 Allocate a storage region
** on the heap */
#include "stdio.h"
main()
{
     extern char *alloc(), *fgets();
     extern int fputs(), free();
     char *buffer;

     buffer = alloc(255);
     fputs("Enter a line of data > ", stdout);
     fgets(buffer, 255, stdin);
     fputs(buffer, stdout);
     free(buffer);
}
```

```
A>EX4-14
Enter a line of data > Hello!
Hello!
```

说明：语句

buffer = alloc（255）；

表示动态申请一块255字节长的存贮区，将用buffer指向该存贮区。而语句

free（buffer）；

用来释放这块存贮区。

（2）机器级函数

所谓机器级函数是指函数的作用对象由内存地址直接给出，因而与机器指令比较类似。虽然它们的级别很低，但很有用处。比如在编制较低级程序时，使用了这组函数，可以不必或尽可能少地调用汇编子程序（汇编函数），从而能提高编程序的效率。机器级函数的功能一般来说都类似汇编指令或汇编子程序，因此不作详细介绍，这里仅对这些函数的功能简要地予以阐述。

① 与存贮器有关的函数

这组函数用来读写存贮器单元、移动内存块、填充内存块以及在内存区域中进行查找。下面一一进行介绍。

* peek（offset, seg）

读入段地址为seg，位移量为offset（下文中表示为seg：offset）的内存单元的内容。其中参数offset和seg的类型均为无符号整型，返回参数为一个16位的机器字。

* pokeb（offset, seg, byte）

把给定的字节byte写到内存单元seg：offset中去。

* pokew（offset, seg, word）

把给定的机器字word写到内存单元seg：offset中去。

* setmem（address, count, value）

用字节value填充指针address指向的内存区，共填入count个字节。

* movblock（soffset, sseg, doffset, dseg, count）

把从sseg：soffset开始长为count的内存块移至dseg：doffset开始的内存区中。

* movmem（sptr, dptr, count）

把sptr指向的内存区中的count个字符移到指针dptr指向的区域。在大编译模式下，其功能与movblock相类似。

* qsort（ptr, num, width, cmpf）

利用比较函数cmpf（）对ptr指向的存贮区中的元素进行排序，这个存贮区由num个长度为width的元素组成。下面是使用qsort（）函数进行排序的一个例子。

例4-15　使用qsort（）函数进行排序

功能：从标准输入设备上读入几个字符串，按字典顺序排序后输出。

程序与运行结果：

```
/* EX4-15 Sort lines from stdin into
** ascending sequence */
#include "stdio.h"
#define MAXLINES 1000

unsigned char *line[MAXLINES];
comp(a, b)
unsigned char **a, **b;
{
```

```
    return strcmp(*a, *b);
}
main()
{
    extern char *alloc(), *fgets();
    int j, k;
    unsigned char buffer[132];

    for(j = 0; j < MAXLINES; ++j)
    {
        if(!fgets(buffer, 130, stdin)) break;
        line[j] = alloc(strlen(buffer) + 1);
        strcpy(line[j], buffer);
    }
    qsort(line, j, 2, comp);
    for(k = 0; k < j; ++k) printf("%s", line[k]);
}
```

```
A>EX4-15
sort
an
array
of
records
in
memory
^Z
an
array
in
memory
of
records
sort
```

说明：函数comp ( ) 在排序时使用，它作为qsort ( ) 函数的函数指针类型的参数被传递给qsort ( ) 函数。在qsort ( ) 的函数体中，comp ( ) 函数用作排序比较函数。

* wqsort (num, cmpf, xchgf, buf)

把buf中num个字段利用cmpf ( ) 函数进行排序，在排序过程中使用xchgf ( ) 函数交换两个字段的位置。

② 端口输入输出函数

* inportb (portno)

从指定的端口portno中读入一个字节，portno的类型为unsigned。

* inportw (portno)

从指定的端口portno中读入一个字 (16位) 。

* outportb (portno, byte)

从端口portno上输出指定的字节byte。

* outputw (portno, word)

从端口portno上输出指定的字word。

③ 其它机器级函数

* farcall (offset, seg, srv, rrv)

调用始址为seg：offset的一段子程序，srv指向入口寄存器参数，rrv指向出口寄存器参数。srv和rrv均为指向结构的指针，它们的定义为：

```
struct{unsigned  ax, bx, cx, dx, si, di, ds, es; }
```

* srv, * rrv;

其中，ax, bx, …, es表示8088CPU中对应的寄存器。

* segread (rv)

读入四个8088段寄存器的当前值，其中，rv指向存放段寄存器的结构，它定义为：

```
struct {int scs, sss, sds, ses } rv;
```

其中，scs,sss,sds和ses分别表示段寄存器cs，ss,ds和es。下面是打印段寄存器内容的例子。

例4-16　打印段寄存器

功能：打印当前寄存器的内容。

程序与运行结果：

```
/* EX4-16 Read the current segment registers */
main()
{
     extern int segread();

     struct {int scs, sss, sds, ses;} rv;
     unsigned int code_segment, data_segment,
                  stack_segment, extra_segment;

     segread(&rv);
     code_segment = rv.scs;
     stack_segment = rv.sss;
     data_segment = rv.sds;
     extra_segment = rv.ses;
     printf("8088 segment registers:\n");
     printf("\nCS: %4x\nDS: %4x\nSS: %4x\nES: %4x\n",
          code_segment, data_segment,
          stack_segment, extra_segment);
}
A>EX4-16
8088 segment registers:

CS: 4D25
DS: 4FCE
SS: 4FCE
ES: 4FCE
```

* intrinit (func, stack, vecno)

安装中断处理程序。其中func ( ) 是欲安装的中断处理程序；vecno是中断号；stack是func ( ) 要使用的栈的大小，它由函数的大小加上128字节得到。

(3) 与MS-DOS的接口

C86允许用户程序调用MS-DOS的所有软中断和系统功能调用。C86与MS-DOS的接口是通过一组系统库函数来完成的，现把这几个函数介绍如下。

* sysint (vecno, srv, rrv)

该函数的功能是调用MS-DOS的vecno号软中断，入口参数为srv，出口参数为 rrv，函数的返回值为标志寄存器的内容。关于MS-DOS的软件中断请参阅第二章。下面的一个例子使用了sysint ( ) 函数。

例4-17　执行INT指令

功能：调用软中断INT10将显示器改为中分辨率图形模式。

程序：

```
/* EX4-17 Execute an INT instruction */
main()
{
    struct { int ax, bx, cx, dx, si, di, ds, es;}
    rv;

    rv.ax = 0x0004;
    sysint(0x10, &rv, &rv);

}
```

说明：INT10用来调用BIOS中的显示器驱动程序。当AH = 00，AL = 04时，将显示器置为中分辨率图形模式 (参见第二章) 。

* sysint21 (srv, rrv)

该函数的功能是调用MS-DOS的系统功能调用，调用号放在srv.ax的高字节 (AH) 中，函数的返回值为标志寄存器。函数中的参数srv和rrv的类型与sysint ( ) 函数的参数srv和rrv相同。关于MS-DOS的系统功能调用请参见本书上册第二章第五节。

例4-18　系统功能调用

功能：调用MS-DOS的 2 号系统功能调用输出一个字符 'A' 。

程序：

```
/* EX4-18 Call a MS-DOS system call */
main()
{
    struct {int ax, bx, cx, dx, si  di, ds, es;}
    rv;
```

```
rv.ax = 0x0200;
rv.dx = 0x0041;
sysint21(&rv, &rv);
}
```

第四节 C86编译系统的操作与使用

C86是Computer Innovations公司开发的一个功能齐全的C语言，它的编译程序由四趟扫描程序组成。C86的目标文件（.OBJ文件）可以使用Microsoft公司的标准连接程序LINK进行连接。此外，C86的编译系统还提供一个目标库管理程序和一个源程序管理程序，可供用户扩充系统库和建立专用库（例如绘图函数库）使用。本节将分别介绍C86的编译程序以及库管理程序的使用方法。

1. 概述

（1）C86编译系统的组成

C86的编译系统由编译程序、库管理程序和系统函数库等部分组成。它主要包含以下几个文件：

CC1.EXE	编译程序的第1趟扫描（预处理）
CC2.EXE	编译程序的第2趟扫描（语法分析）
CC3.EXE	编译程序的第3趟扫描（代码生成）
CC4.EXE	编译程序的第4趟扫描（代码优化）
MARION.EXE	目标库管理程序
ARCH.EXE	源程序库管理程序
CSLIB.LIB	小模式系统函数库
CBLIB.LIB	大模式系统函数库
8087S.LIB	小模式8087库
8087B.LIB	大模式8087库
ALIB.ARC	汇编源程序库
CLIB.ARC	C语言源程序库
STDIO.H	标准输入输出定义文件

根据编译的小模式和大模式，其中的每种目标库都分别提供了两个，用户可根据需要自行选用。

标准输入输出定义文件stdio.h已在前面几节中多次使用过，它的具体内容如下：

```
/*standard i/o header file for c86
*/

#define NULL 0
#define EOF (-1)        /* standard end of file */
#define EOS '\0'        /* standard end of string */
```

```
#ifdef _C86_BIG
#define stdin 0x8000L
#define stdout 0x8001L
#define stderr 0x8002L
#else
#define stdin 0x8000     /* standard input */
#define stdout 0x8001    /* standard output */
#define stderr 0x8002    /* standard error */
#endif

#define AREAD 0          /* ascii read */
#define AWRITE 1          /* ascii write */
#define AUPDATE 2         /* ascii update */
#define BREAD 4           /* binary update */
#define BWRITE 5          /* binary write */
#define BUPDATE 6         /* binary update */
typedef char FILE;
#define getchar() fgetc(stdin)
#define getc(x) fgetc(x)
#define putchar(x) fputc(x,stdout)
#define putc(x,y) fputc(x,y)
#define ungetch(c) ungetc(c,stdin)

/* definition for setjmp and longjmp
*/

#ifdef _C86_BIG
typedef int jmp_buf[4];
#else
typedef int jmp_buf[3];
#endif

/* end of standard header file
*/
```

（2）C程序运行时的内存分配

C86的目标程序装入机内运行时的内存分配如图4-3所示，整个C程序在内存中分为两个区域——代码区和数据区。对于C86小模式下的程序，代码区和数据区分别都不超过64KB。而在大模式下，代码区和数据区都可以大于64KB。但数据区中的静态数据区和栈区仍然限制在64KB以内。

（3）C语言程序的开发过程

和其它编译语言一样，C86的开发过程也分为源程序的编辑、编译以及目标模块的连接和执行等几个阶段（见图4-4）。

首先用编辑程序EDLIN(或WORDSTAR)建立源程序，接着根据源程序是汇编程序还是C程序分别进行汇编或编译，然后把需要使用的几个目标程序模块以及目标库连接生成.EXE

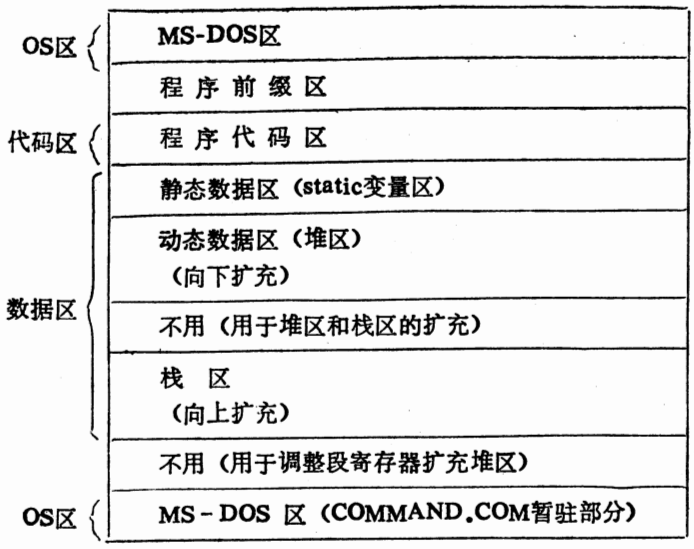


图4-3 C86程序的内存分配

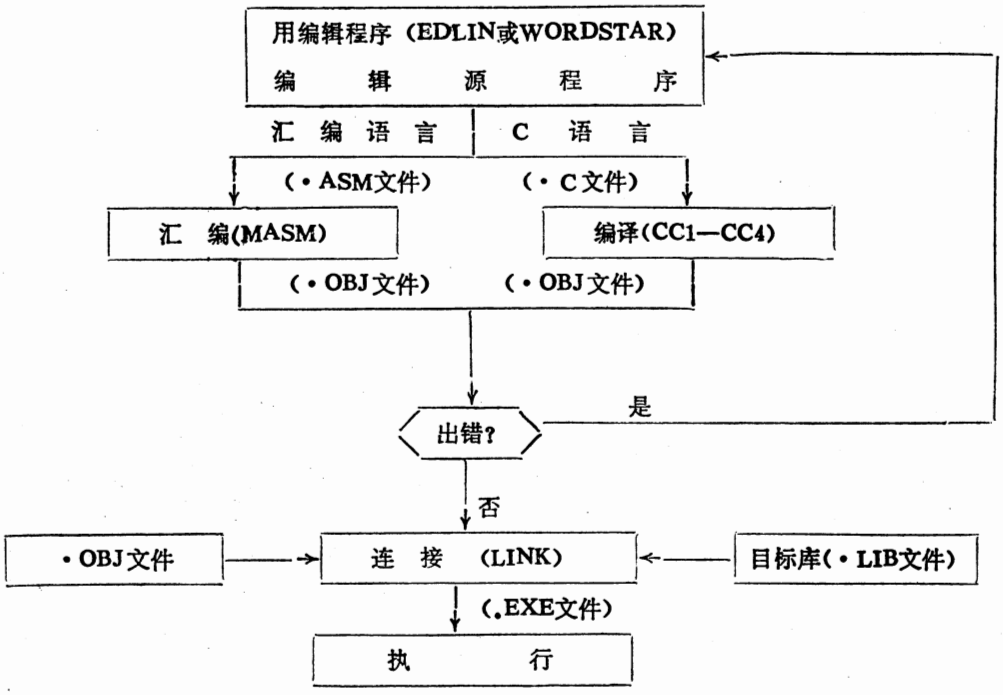


图4-4 C程序的开发过程

文件，最后执行.EXE文件。例如，要编制一个名为“SAMPLE.C”的C语言程序，其操作步骤如下：

C>EDLIN SAMPLE.C

```
C>CC1  SAMPLE
C>CC2  SAMPLE
C>CC3  SAMPLE
C>CC4  SAMPLE
C>LINK  SAMPLE, , , CSLIB
C>SAMPLE
```

应当注意，C86在编译时仅能处理7位的字符，而对最高位需要进行屏蔽，所以在CCDOS下不能正确地输出汉字。如果希望输出汉字，在编译以前必须使用ALLBITS程序进行处理。该程序的清单如下：

```
#include "stdio.h"
main()
{
     int cc;

     while((cc=getchar())!=EOF){
     if(cc&0x80)printf("\\%3o",cc);
     else printf("%c",cc);
   }
}
```

程序的功能为逐个读入输入文件中的字符，如果该字符大于0x80，则将它换成八进制换码序列。使用方法如下：

```
ALLBITS '<' <源文件> '>' <结果文件>
```

例如，对SAMPLE.C文件

```
main ( )
{
     printf ( "计算机科学" ) ;
}
```

使用命令

```
A>ALLBITS  <SAMPLE.C  >TMP.C
```

处理后，得到的文件TMP.C的内容为：

```
main ( )
{
     printf ( "\274\306\313\343\273\372\277\306\321\247" ) ;
}
```

以下是一个C86的编译操作的批文件C86.BAT：

```
echo off
edlin %1.c
allbits < %1.c > tmp.c
cc1 tmp
if ERRORLEVEL 1 goto abc
```

```
cc2 tmp
if ERRORLEVEL 1 goto abc
cc3 tmp
if ERRORLEVEL 1 goto abc
cc4 tmp
link tmp,,,cslib
copy tmp.exe %1.exe
%1
:abc
```

2．C86的编译命令

C86程序中可以使用以“#”开头的编译命令嵌入一个源文件（或定义文件）、定义一些符号常量，定义宏替换及实现条件编译。编译命令的格式如下：

#〈编译命令〉{〈参数〉}

它通常位于程序文件的首部，也可以单独组成一个定义文件，以供使用时嵌入。下面就介绍C86的编译命令。

（1）嵌入文件命令#include

用于嵌入文件的编译命令有以下两种形式：

#include“文件名”

#include〈文件名〉

这两种形式都会将指定的文件嵌入到该编译命令所在的位置处。前者查找文件的顺序为程序文件所在的盘、缺省盘和A盘，后者只在缺省盘和A盘查找。

编译命令#include能用于多种场合，如可以嵌入标准输入输出定义文件stdio.h，也可以嵌入几个程序文件共用的外部变量说明文件，还可以嵌入已存在的程序文件。

（2）宏定义命令#define

宏定义命令可以用来定义符号常量和“宏”，它有以下两种形式：

#define〈宏替换名〉〈宏替换体〉

#define〈宏替换名〉(〈形参〉)〈宏替换体〉

其中，宏替换体为一串字符，形参为出现在宏替换体内的形式参数。

#define命令用来定义符号常量时，宏替换名表示该常量，而宏替换的体为该常量的值，它为常量表达式。下面是一些定义常量的例子：

```
#define  PI  3.1415926536
#define  PISQUARE  PI * PI
#define  TRUE  1
#define  FALSE  0
#define  TAB  '\t
#define  EOF  -1
```

#define命令也可以将多个语句或表达式定义成一个宏替换，使用时可用宏名代替那些语句（称为宏调用）。如

```
#define PRINTX printf("%d\n", x)    /* PRINTX用来表示一个
printf()函数，它用来输出一个整型量x */
#define PRA printf("%d", a); putchar('\n')/*PRA表示两个函数，
它输出一个整型量a和新行符'\n' */
```

#define命令还能用来改变C语言的语法记号，以下列程序为例：

```
#define PROGRAM()  main()
#define BEGIN      {
#define END        }
 PROGRAM()
  BEGIN
  printf("Hello\n");
  END
```

#define命令在定义宏时可以带参数，下面是一些带参数的宏定义：

```
#define area(x)     (PI*x*x)     /*将area(x)定义为PI*x² */
#define max(a, b)   (a>b? b:a)     /*将max(a, b)表示为取a
和b较大的一个为值*/
#define PR(x)  printf("x=%d\t", (int)(x))
#define PRT(x)  PR(x); printf("\n")
#define PRT2(x)  PR(x); PRT(y)
#define TAB(c, i, oi, t)   if(c=='\t')\
        for(t=8-(i-oi-1)%8, oi=i; t; t--)\
        putchar('')
```

其中，最后一例中的“\”为续行符，表示宏替换体超过一行。

编译程序中的第1趟扫描CC1在进行预处理时，对程序中出现宏调用处逐一用宏定义中相应的宏体进行替换，宏体中的形式参数用宏调用中的实在参数来代替。在调用带有参数的宏替换时应格外小心，防止参数错误。例如，下列的程序是不正确的：

```
#define square(n)  n*n
main()
{
      printf("%f\n", 27.0/square(3.0));
}
```

这是因为printf()中宏调用square(3.0)被替换后其形式为：

```
printf("%f\n", 27.0/3.0*3.0);
```

显然，这不符合程序的要求。改正的办法是将宏定义修改为：

```
#define square(n)  ((n)*(n))
```

这样才能得到正确的结果。

对已定义的宏替换可以使用命令

```
#undef <宏替换名>
```

删除，删除后的宏替换名可重新定义。

（3）条件编译命令

条件编译命令用来控制某段程序是否要进行编译，因为这些命令不常使用，故不作细述，仅列出这些命令及其含义：

#ifdef〈宏名〉	如果指定的宏替换已定义，则编译随后的程序段(到#endif为止)。
#ifndef〈宏名〉	如果指定的宏替换未定义，则编译随后的程序段。
#if〈表达式〉	如果表达式为真，则编译随后的程序段。
#else	不满足#if的条件时，编译随后的程序段。
#endif	条件编译结束。

3. C86的编译开关

C86的编译程序分四趟扫描，它们各自完成一定的编译功能。每趟扫描程序都有若干个编译开关，这些编译开关可用来设置目标程序的存贮模式，列出源程序清单以及列出目标程序的汇编清单。下面逐一介绍这四趟扫描程序的有关开关。

（1）预处理程序CC1

预处理程序CC1用来处理C程序中以#开头的编译命令，其格式为：

CC1 〔-bcdip〕〈文件名〉

其中，“文件名”为源程序文件的文件名，缺省的类型名为“.C”。关于命令行中编译开关的含义如下：

-b 把程序的存贮方式置为大模式

-c 允许注释嵌套

-d 宏定义（类似#define编译命令）

-i 把标识符的有效长度从8改为31个字符

-p 从标准输出设备上列出源程序清单（包括嵌入文件）

（2）语法分析程序CC2

语法分析程序CC2的使用格式为：

CC2 〔-su〕〈文件名〉

其中编译开关的含义为：

-u 把所有char型数据作为unsigned char来处理

-s 把所有字符数组作为无符号字符数组处理

（3）代码生成程序CC3

CC3用来生成目标代码，格式为：

CC3 〔-nt〕〈文件名〉

其中编译开关的含义如下：

-n 使用8087处理浮点运算（机内要有8087芯片）

-t 产生可以动态调试的目标

-t开关实际上在每个函数的有效代码前产生了一个跟踪函数$entry( )的调用语句。$entry( )函数可由用户任意修改，以达到动态调试程序的目的。$entry( )的初始定义为检查栈是否溢出，若溢出则报告“NOCORE”（内存容量不够）。

（4）代码优化程序CC4

这个程序用来对CC3生产的目标进行优化，并输出目标代码文件·OBJ或汇编文件·ASM，格式如下：

CC4 〔-a〕〈文件名〉

其中，“-a”开关表示要生成一个汇编目标文件，但不生成代码目标文件。这个汇编目标文件可以用宏汇编MASM进行汇编。

4. 库管理程序

C86的编译系统提供了两个库管理程序分别用来管理目标程序库和源程序库，这些库程序既可以是系统库，应用软件库，也可以是用户自行编制的用户函数库。

C语言的库管理程序为使用者扩充系统库和建立用户库提供了极为方便的手段，同时也为开发C语言支撑的应用软件包提供了良好的途径。

（1）目标库管理程序MARION

marion用来管理Microsoft格式的目标模块库，它的命令格式如下：

marion 〔-b d e l m u x〕〈库名〉{〈目标模块名〉}

其中，“库名”的缺省类型名为“.LIB”；“目标模块”的缺省的类型名为“.OBJ”，marion的开关含义如下：

-u 用给定的目标模块建立一个库（当指出库不存在时）或增加（库中无此模块时）、替换（库中有此模块时）库中的一个模块

-d 从库中删除一个模块

-x 把库中指定的模块取出以.OBJ文件格式记盘

-l 列出库中模块名、模块大小以及定义的外部变量

-m 按自然顺序列出库中的模块名

-b 建立后备库

-e 抑制校验和出错信息输出

例如，要用第五节例4-25中的几个绘图子程序建立一个绘图函数库，可用下列命令来实现：

marion -u calcomp plots plot

这表示用plots和plot两个模块建立了一个绘图函数库calcomp。若需要在calcomp.lib中再加入facfor和newpen两个新模块，则可用下列命令实现：

marion -u calcomp factor newpen

如果模块plot经过了修改，则可用下列命令更新：

marion -u calcomp plot

它等效于：

```
marion -d plot
marion -u plot
```

（2）源程序库管理程序ARCH

arch可用来建立源程序库以及对已存在的源程序库增加、删除和替换一个源程序文件，命令格式如下：

```
arch [-d t u x] <库名>{<源程序文件名>}
```

其中，“库名”的缺省类型名为“.arc”，源程序文件的缺省类型为“.c”。

arch的开关一次仅能使用一个，它们的含义如下：

-u 建立源程序库或者在源程序库中加入或更新一个源程序模块

-d 删除源程序模块

-t 显示源程序库的内容

-x 把库中指定的模块取出以源程序的形式记盘

源程序库为用户库存、查找、阅读和修改库函数提供了方便。它依次由各个源程序文件组成，每个文件开头有一个头标，其形式为：

```
-ARCHIVE-<文件名><文件的字节数>
```

源程序库按其是汇编程序还是C语言程序分两类，例如alib.arc和8087.arc为汇编源程序，而clib.arc是C语言源程序库。

第五节 C语言程序设计举例

本节准备通过几个C语言程序的实例，向读者提供写C语言程序的示范，同时对C语言的基本内容作些补充。所有的例子都是按CI-C86 2.00版的要求写成的，其中有些与标准C语言不完全兼容，一般情况下，也不能直接在其它版本的C语言下运行。但是这些程序的设计方法还是可以借鉴的。

在这些例子中，一般都涉及到C语言的某个成分或C86的一些系统库函数，必要时将在有关例子之前作些简单的说明。

1. 递归函数

例4-19 阶乘

（1）功能：输出1至6的阶乘。

（2）程序与运行结果：

```
/* EX4-19 Fact */
main()
{
    int i = 0;

    while (++i < 6)
      printf("%d! = %d\n", i, factorial(i));
}
```

```
factorial(n)     /* Return n! */
int n;
{
     if (n == 1)
       return(1);
     else
       return(n * factorial(n - 1));
}
```

```
A>EX4-19
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
```

（3）说明：

本例使用一个递归定义的函数factorial（）计算阶乘，其递归出口条件为n==1。factorial（）函数直接由阶乘的递归定义

$$\begin{cases} f(1)=1 \\ f(n)=f(n-1)\times n \end{cases}$$

得到。

2．字符串处理

例4-20　检验是否回文

（1）功能：检验两个给定的字符串，看它们是不是回文（即正向和反向阅读完全相同的一个句子）。

（2）程序与运行结果：

```
/* EX4-20 Submit two strings
** to the palindrome test */
#define TRUE 1
#define FALSE 0
char string1[] = "able was i ere i saw elba";
char string2[] = "fourscore and seven years ago";
main()
{
     if(isitpal(string1) == TRUE)
         printf("string1 is a palindrome\n");
     if(isitpal(string2) == TRUE)
         printf("string2 is a palindrome\n");
}
isitpal(strptr) /* test a string for palindromia */
char *strptr;
{
```

```
    char *strptr1, *strptr2;

    for (strptr1 = strptr, strptr2 = strptr+
      strlen(strptr) - 1;
        strptr1 < strptr2;
        strptr1++, strptr2--)
        if(*strptr1 != *strptr2)
            return(FALSE);
    return(TRUE)
}
```

```
A>EX4-20
string1 is a palindrome
```

（3）说明：

① 在程序给定的两个供检验的字符串中，string 1 是回文，string 2 不是回文。

② isitpal()函数中for语句的初值表达式（表达式1）和结束处理表达式（表达式3）是使用顺序运算符的一个例子，它充分显示了C语言表达的简洁性。

③ 本例是用指针来访问数组元素的，其中 * strptr1和 * strptr 2 分别表示指针strptr 1和strptr 2 所指向单元的内容。

④ isitpal（）函数的功能是检验strptr指向的字符串是否为回文，若是则返回“真”，否则返回“假”。

3．数组处理

例 4-21　查表

（1）功能：在数组table〔 〕中查找x，将其在table中的位置放入index，若找不到则index为－1。

（2）程序与运行结果：

```
/* EX4-21 Lookup */
int table[] = {3456, 6750, 1245, 8905, 1254,
               1357, 4098, 1347};
int x = 1254, index;
main()
{
    lookup(&table[0], x, &index, 8);
    printf("%d\n", index);
}
lookup(t, val, i, n)
int *t, *i, val, n;
{
    int k;
    for(k = 0; k < n; ++k)
        if(*(t +k) == val)
        {
            *i = k;
```

```
            return;

        *i = -1;
}

A>EX4-21
4
```

（3）说明

① main（）函数中lookup（）的实参&table〔0〕和&index分别表示数组table〔〕的始址和index的地址。这是C语言常用的参数传递方式。

② lookup（）函数中的if语句中的表达式“*（t+k）”表示将指针t加上k个单位（这里2个字节为一个单位），然后取它的内容。

4. 重复语句

例4-22　显示一个字符串中所有字符的ASCII码

（1）功能：输入一个字符串，将它的所有字符的ASCII码显示出来，并要求输入下一个字符串，直至输入ctr1 -z为止。

（2）程序与运行结果：

```
/* EX4-22 Display the ASCII values of a string */
#include "stdio.h"
main()
{
    extern char *fgets();
    char string[80], *strptr;

    printf("When prompted, type a line.\n");
    do
    {
        putchar(':');
        strptr = fgets(string, 80, stdin);
        do
        {
            printf("%c = %2x\n", *strptr, *strptr);
        } while (*strptr++ != '\0');
    } while (*string != '\0');
}
A>EX4-22
When prompted, type a line.
:hello.
h = 68
e = 65
l = 6C
```

```
l = 6C
o = 6F

 =  A
  =  0
:a b c
a = 61
  = 20
b = 62
  = 20
c = 63

 =  A
  =  0
:^z
  =  0
```

（8）说明：程序中使用了两个do重复语句，do语句先执行循环体，后判断出口条件。

5. 流式文件1

例4-23　复制文件

（1）功能：复制文件。　　（2）程序：

```
/* EX4-23 Copy files */
#include "stdio.h"
main(argc, argv)
char *argv[];
int argc;
{
    FILE *fp1, *fp2, *fopen();

    if(argc != 3)
        abort("Bad command!");
    if((fp1 = fopen(*++argv, "r")) == NULL)
        {
          fprintf(stderr, "Can't open %s\n , *argv)
          exit(1);
        }
    if((fp2 = fopen(*++argv, "w")) == NULL)
        {
          fprintf(stderr, "Can't open %s\n", *argv);
          exit(1);
        }
    filecopy(fp1, fp2);
    fclose(fp1); fclose(fp2);
    exit(0);
}
filecopy(fp1, fp2) /* copy file fp1 to fp2 */
FILE *fp1, *fp2;
```

```
{
    int c;

    while((c = getc(fp1)) != EOF)
        putc(c, fp2);
}
```

（3）使用的例子：

A>EX4-23 SFILE DFILE

表示把SFILE复制到DFILE，它的功能类似MS-DOS的复制命令。

（4）说明

① 函数filecopy（）用来复制文件，在调用该函数前，使用的两个文件必须打开。

② main（）函数中首先判别命令行中的参数是否为三个，如不对则给出出错提示信息并退出。main（）取到的三个参数中，第1个为“c”（标志）；第2个和第3个分别为源文件名和目的文件名。

③ 在打开源文件和目的文件时，如打开失败，则给出出错信息：

“can't open……”

6. 流式文件2

例4-24 分页打印

（1）功能：分页打印ASCII码文件并加上行号。

（2）程序：

```
/* EX4-24 Type a file */
#include "stdio.h"
#define TLINE 60
#define LINEP 2
main(argc, argv)
int argc;
char *argv[];
{
    int flag, page, line, i;
    char buf[100];
    FILE *fp;
    if(argc < 2) abort("\nNo file name\n");
    fp = fopen(argv[1], "r");
    if(fp == 0) abort("\nCannot open file\n");
    flag = page = line = 1;
    while(flag) {
        for(i = 0; i < LINEP; i++) printf("\n");
        printf("%s\n", argv[1]);
        for(i = 0; i < TLINE; i++) {
            setmem(buf, 100, 0);
            if((fgets(buf, 100, fp)) == 0) flag = 0;
            if(flag) {
                printf("%6d  %s", line++, buf);
```

```
                if(buf[98] != 0) printf("\n");
            }
            else
                for(i += 2; i < TLINE; i++)
                    printf("\n");
        }
        printf("\n%72s%d.\n", ".", page++);
        for(i = 0; i < LINEP; i++) printf("\n");
    }
    fclose(fp);
}
```

（3）运行结果

使用该程序输出文件的命令和清单如下：

A>EX4-24　　EX4-24.C

EX4-24.C

```
 1  /* EX4-24 Type a file */
 2  #include "stdio.h"
 3  #define TLINE 60
 4  #define LINEP 2
 5  main(argc, argv)
 6  int argc;
 7  char *argv[];
 8  {
 9      int flag, page, line, i;
10      char buf[100];
11      FILE *fp;
12      if(argc < 2) abort("\nNo file name\n");
13      fp = fopen(argv[1], "r");
14      if(fp == 0) abort("\nCannot open file\n")
15      flag = page = line = 1;
16      while(flag) {
17          for(i = 0; i < LINEP; i++) printf("
18          printf("%s\n", argv[1]);
19          for(i = 0; i < TLINE; i++) {
20              setmem(buf, 100, 0);
21              if((fgets(buf, 100, fp)) ==
22              if(flag) {
23                  printf("%6d  %s", line++,
24                  if(buf[98] != 0) printf("
25              }
26              else
27                  for(i += 2; i < TLINE; i++
28                      printf("\n");
29          }
30          printf("\n%72s%d.\n", ".", page++);
31          for(i = 0; i < LINEP; i++) printf("
32      }
33      fclose(fp);
34  }
```

（4）说明：

① 该程序首先检查命令行中有无参数，若没有就结束程序的执行，并显示下列信息：

ABORT: -

NO file name

若有便打开这个文件，如这个文件不存在时，终止程序并显示：

ABORT: -

Cannot open file

② while语句用来输出文件的内容并加行号和页号。

③ setmem ( ) 函数用来初始化缓冲区。语句

if (buf [98] ! =0) printf ("\n") ;

用来判别是否超过99个字符的行，若是，则在输出该行的前99个字符后插入一个新行符。

7. 绘图函数

C86提供了几个基本绘图函数，使用它们可以在IBM PC的图形显示器上绘图，这些函数及其功能如下：

* Crt-mode (mode) 设置显示器模式，当mode为3时表示80×25字符模式，当mode为4时表示彩色中分辨率模式（320×200），当mode为6时表示高分辨模式（640×200）。

* Crt-line (x1, y1, x2, y2, color) 以颜色color在（x1, y1）和（x2, y2）之间画一条直线。其中color在中分辨模式下一般定义如下：

color= 0	黑色
color= 1	蓝色
color= 2	红色
color= 3	白色

* crt-rdot (row, column) 读指定点的颜色。
* crt-wdot (row, column,color) 在指定点以颜色color画点。
* crt-srcp (row, column, page) 把光标设置在page页的row行column列。

例4-25是利用这些绘图函数写的一个程序，该程序由EX4-25.C, plots.c, plot.c, factor.c, newpen.c和calcomp.h等六个文件组成。

例4-25 绘制正方形

（1）功能：

① 主程序EX4-25.C

以蓝红白三种颜色交替显示12个大小不同的正方形。

② 初始化模块plots.c

初始化各函数使用的公共数据并把显示器设置成中分辨率模式。该模块中还提供了一个坐标转换函数xfer ( )，供计算屏幕坐标使用。

③ 画直线模块plot.c

该模块由一个函数plot ( ) 组成，它用来画直线以及进行绘图结束处理。该函数的调用格式为：

plot (< x坐标 >, < y坐标 >, < 方式 >)

当方式为2时，表示在当前点和（x, y）之间画一条直线；当方式为3时，将当前点移至（x, y）；而方式为999时，进行绘图结束处理。

④ 缩放变换模块factor.c

该模块中的函数factor ( ) 可以改变随后画出图的比例尺寸，其参数fctr为放大倍数。

当其值大于0时表示放大，而小于0时表示缩小。

⑤ 改换颜色模块newpen.c

该模块中的函数newpen（）用来更换绘图的颜色。

⑥ 公共外部变量说明文件calcomp.h

该文件中包含几个模块中要使用的外部变量的说明。

（2）程序：

① EX4-25.c

```
/* EX4-25 Box */
double d[2][4] =
    {{3.0, 3.0, 0.0, 0.0}, {0.0, 3.0, 3.0, 0.0}};
main()
{
    int i  j;

    plots(0.0, 0.0, 0);
    for(i = 0; i < 12; i++)
    {
        factor((double)(1 + i * 0.5));
        newpen(i % 3 + 1);
        for(j = 0; j < 5; j++)
            plot(d[0][j], d[1][j], 2);
    }
    plot(10.0, 0.0, 3);
    plot(0.0, 0.0, 999);
}
```

② plots.c

```
double o_x, o_y, start_x, start_y, fctr0;
unsigned color, ip[10];
plots(dumy1, dumy2, ldev)
double dumy1, dumy2;
unsigned ldev;
{
    o_x=start_x=dumy1;
    o_y=start_y=dumy2;
    color = 1; fctr0 = 1.0;
    crt_mode(4);
}
xfer(x, y)
double x, y;
{
    ip[0] = (start_x + o_x) * 10 * fctr0;
    ip[1] = (start_y + o_y) * 10 * fctr0;
    ip[2] = (x + o_x) * 10 * fctr0;
    ip[3] = (y + o_y) * 10 * fctr0;
}
```

③plot.c

```
#include "calcomp.h"
plot(x, y, ipen)
double x, y;
int ipen;
{
    if(ipen==999)
    {
        putchar('\7');
        getchar();
        crt_mode(3);
    }
    xfer(x,y);
    if(ipen == 2)
        crt_line(ip[0],ip[1],ip[2],ip[3],color);
    start_x = x; start_y = y;
}
```

④ factor.c

```
#include "calcomp.h"
factor(fctr)
double fctr;
{
fctr0 = fctr;
}
```

⑤ newpen.c

```
#include "calcomp.h"
newpen(inp)
int inp;
{
    color = inp;
}
```

⑥ calcomp.h

```
#include "stdio.h"
extern double o_x, o_y;
extern double start_x, start_y, fctr0;
unsigned color, ip[10];
```

（3）运行结果：

该程序的运行结果如图4-5所示。

（4）说明：

① EX4-25中定义了一个二维外部数组d,它用来存放最小正方形的四个端点坐标。在该

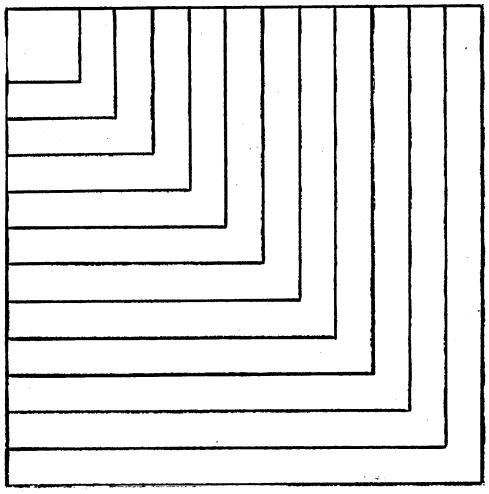


图4-5　12个正方形

文件中，首先调用plots（）进行绘图初始化，再依次画出12个正方形。每次放大一些并改变颜色。最后使用“plot（0.0，0.0，999）；”语句进行绘图结束处理。

② 程序文件plots.c中定义了一些外部变量，它们用于在函数之间传递参数。其中的函数plots（）对这些变量进行初始化。

③ 该程序的编译连接方法有两种，首先将EX4-25，plots，plot，newpen和factor分别编译生成.OBJ文件，然后打入命令：

A＞LINK EX4-25+PLOTS+PLOT+FACTOR+NEWPEN，，，
CSLIB〈CR〉

或者打入下列命令：

A＞MARION -U CALCOMP PLOTS PLOT FACTOR
NEWPEN 〈CR〉
A＞LINK EX4-25，，，CALCOMP+CSLIB〈CR〉

其中库名CALCOMP和CSLIB的顺序不能颠倒。

8．调用MS-DOS软中断和系统功能调用

例4-26　英制到公制的转换

（1）功能：由菜单选择A…D分别把英吋化为厘米，英呎化为米，英哩化为公里或英磅化为公斤。选择Z表示退出。

（2）程序：

```
/* EX4-26 English standard measures
** to metric system */
#include "stdio.h"
main()
{
```

```
    char choice;
    menu();
    while(1) {
        choice = inkey();
        choice = tolower(choice);
        switch(choice) {
            case 'a': out("inches ", 2.54, "cm");
                      break;
            case 'b': out("feet   ", 0.3048, " m");
                      break;
            case 'c': out("miles  ", 1.609, "km");
                      break;
            case 'd': out("pounds ", 0.45, "kg");
                      break;
            case 'z': crt_mode(2); exit();
            dafault : break;
        }
        crt_srcp(15, 22, 0); printf(" \b");
    }
}
out(info, data, unit)
char *info, *unit;
double data;
{
    double d;
    printf("\n\nHow many %s      \b\b\b\b", info);
    scanf("%lf", &d);
    printf("%g %s = %g %s   \n", d, info,
           d * data, unit);
}
menu()
{


   crt_mode(1);
   cls();
   crt_srcp( 6,2,0);
   printf("Select conversion from this list:");
   crt_srcp( 8,8,0);
   printf("A  Inches to Centimeters");
   crt_srcp( 9,8,0);
   printf("B  Feet to Meters");
   crt_srcp(10,8,0);
   printf("C  Miles to Kilometers");
   crt_srcp(11,8,0);
   printf("D  Pounds to Kilograms");
   crt_srcp(12,8,0); printf("Z  Exit");
   crt_srcp(15,8,0); printf("Your choice > ");

}
cls()
{
    struct {int ax, bx, cx, dx, si, di, ds, es;}
    regs;
    regs.ax = 0x0600;
    regs.bx = 0x1700;
```

```
        regs.dx = 0x184f;
        regs.cx = 0x0000;
        sysint(0x10, &regs, &regs);
}
inkey()
{
        struct {int ax, bx, cx, dx, si, di, ds, es;}
        regs;
        regs.ax = 0x0100;
        sysint21(&regs, &regs);
        return(regs.ax & 0x00ff);
}
```

（3）运行结果：

执行该程序的方法为：首先打入

A>EX4-26

屏幕上显示出下列画面：

```
Select conversion from this list:

     A  Inches to Centimeters
     B  Feet to Meters

     C  Miles to Kilometers
     D  Pounds to Kilograms
     Z  Exit


     Your choice >
```

接着可以选择某项功能执行，例如要知道10哩等于多少公里，可以打入：

C10<CR>

这时屏幕上的画面为：

```
Select conversion from this list.

     A  Inches to Centimeters
     B  Feet to Meters
     C  Miles to Kilometers
     D  Pounds to Kilograms
     Z  Exit


     Your choice >

How many miles    10
10 miles     = 16.09 km
```

由此可知，10哩等于16.09公里。

（4）说明：

① main（）函数首先显示出一个功能选择清单，然后执行一个重复条件恒为真的while语句，这是避免使用goto语句的一个例子。该程序的出口为switch语句中“z”情况后的exit（）函数。

② main()函数的while语句中，首先利用自定义函数inkey（）输入一个字符，如为大写则将其转化为小写。再根据不同的输入执行不同的功能。

③ 程序中的out（）函数分别进行各类英公制转换并输出结果。

④ menu（）函数用来显示功能选择清单。它首先将显示器置成40×25的彩色字符模式，再调用cls（）函数清除屏幕，并选用蓝色底色，最后显示功能选择清单，每行输出使用crt_srcp（）函数定位。

⑤ cls（）函数中调用了一条10H号软中断，它的入口参数的含义请参见第二章。

⑥ inkey（）函数调用了MS-DOS的01H号系统功能调用，它用来从键盘上读入一个字符。

9. 调用汇编子程序

C86调用汇编子程序十分方便，调用方法与调用外部函数的方法完全一样。但对汇编子程序的书写有一定的要求，在大模式下它的格式通常为：

```
            include   model.h              ; 定义编译模式符号常量@bigmodel
            include   prologue.h           ; 定义有关参数传递的符号常量
            public〈子程序名〉
〈子程序名〉proc  far
            push  bp
            mov   bp, sp                   ; 取参数区基地址
                  ⋮
            ; 从bp+6, …, bp+6+n中取出参数（大模式）
                  ⋮
            ; 子程序体
                  ⋮
            ; 结果（返回参数）放在AX, BX, CX和DX中
            pop   bp
〈子程序名〉endp
            include   epilogue.h           ; 子程序结束
            end
```

当汇编子程序有多个参数时，在调用时从右至左逐个压入堆栈，最后压入返回地址。所以汇编子程序取它们的地址分别为bp+6，bp+6+sizeof（类型），……

返回参数根据其类型分别放入AX（整型和字符），AX+BX（长整型）或AX+BX+CX+DX（长实型）中。

下面列出有关的三个定义文件model.h，prologue.h和epilogue.h。

* model.h

```
;define big or small model
;for library assembly code

FALSE   equ     0       ;for small model
TRUE    equ     1       ;for big model

@bigmodel equ TRUE ;select this for model desired
```

* prologue.h

```
;prologue.h     11/5/83

;standard prologue for c86 assembly code

;DEFINE ARGUMENT BASE RELATIVE FROM BP

IF      @BIGMODEL
@AB     EQU     6
ELSE
@AB     EQU     4
ENDIF
@CODE   SEGMENT BYTE PUBLIC 'CODE'
@CODE   ENDS
@DATAB  SEGMENT PARA PUBLIC 'DATAB'
@DATAB  ENDS
@DATAC  SEGMENT BYTE PUBLIC 'DATAC'
@sb     label   byte
@sw     label   word
@DATAC  ENDS
@DATAI  SEGMENT BYTE PUBLIC 'DATAI'
@ib     label   byte
@iw     label   word
@DATAI  ENDS
@DATAT  SEGMENT BYTE PUBLIC 'DATAT'
@DATAT  ENDS
@DATAU  SEGMENT BYTE PUBLIC 'DATAU'
@ub     label   byte
@uw     label   word
@DATAU  ENDS
@DATAV  SEGMENT BYTE PUBLIC 'DATAV'
@DATAV  ENDS
DGROUP  GROUP @DATAB,@DATAC,@DATAI,@DATAT,@DATAU,@DATAV
@CODE   SEGMENT BYTE PUBLIC 'CODE'
        ASSUME  CS:@CODE,DS:DGROUP

;END OF PROLOGUE h
```

```
    * lpilogue.h

;end of any assembly code

@CODE   ENDS
```

以下为调用汇编子程序的一个例子。

例4-27　字符串长度

（1）功能：调用汇编子程序strlen2来求字符串的长度。

（2）程序：

① EX4-27.C

```
/* EX4-27 String lenth */
main()
{
     int n;

     n = strlen2("This is a string!");
     printf(" n = %d\n", n);
}
```

② STRLEN2.ASM

```
;return the length of a string

        include model.h
        include prologue.h

;enter 1, the address of the string

;exit the length of the string in reg ax

        public  strlen2

if      @bigmodel
strlen2 proc    far
else
strlen2 proc    near
endif
        cld     ;set in the direction of up
        push    bp
        mov     bp,sp
if      @bigmodel
        les     di,dword ptr @ab[bp] ;get pointer
```

```
else
        mov     di,ds
        mov     es,di
        mov     di,@ab[bp]
;get the address of the string
endif
        xor     ax,ax
        xor     cx,cx       ;set the counter
        dec     cx          ;to 64k
        repnz   scasb       ;till zero
        sub     ax,2
        sub     ax,cx
        pop     bp
        ret
strlen2 endp

        include  epilogue.h

        end
```

（3）运行结果：

A>EX4-27

n=17

（4）说明：

① 该例只有一个入口参数，它是字符串的首址。在汇编程序中，使用指令

lcs di, dword ptr @ab〔bp〕

取出，其中@ab在prologue·h文件已定义为6（大模式下）。其返回参数为一个整型量，用AX返回。

② 该程序的编译连接过程如下：

A>CC1 -B EX4-27

A>CC2 EX4-27

A>CC3 EX4-27

A>CC4 EX4-27

A>MASM STRLEN2;

A>LINK EX4-27+STRLEN2, , , CBLIB

10. 存贮管理

例4-28 求指数

（1）功能：输入8189个实数，求出以e为底的指数输出。

（2）程序：

```
/* EX4-28 Exponentiel function */
#define NUMBER 8189
#define SIZE 8

main()
{
    extern char *calloc();
    extern double exp();
    double *d1, *d2;
    int i = 0;

    d1 = (double *)calloc(NUMBER, SIZE);
    d2 = (double *)calloc(NUMBER, SIZE);
    if(!d1 || !d2) abort("Ug, too big\n");
    printf("Please input data:\n");
    for(i = 0; i < NUMBER; i++)
    {
        scanf("%lf", &d1[i]);
        d2[i] = exp(d1[i]);
    }
    printf("\n      N\t\t      EXP(N)\n");
    for(i = 0; i < NUMBER; i++)
        printf("%f\t%f\n", d1[i], d2[i]);
}
```

（8）运行结果（局部）：

```
A>EX4-28
Please input data:
10.5
-2.2
3.08
4.98
-5.2
10.9
2.07
-4.1
-1.9
5.98
 .
 .
 .
```

```
          N                    EXP(N)
10.500000              36315.502674
-2.2000000             0.110803
3.0800000              21.758402
4.9800000              145.474382
-5.2000000             0.005517
10.900000              54176.363797
2.0700000              7.924823
-4.1000000             0.016573
-1.9000000             0.149569
5.9800000              395.440368
  •                      •
  •                      •
  •                      •
```

（4）说明：

① 本例使用了两个65512（8189×8）个字节长的动态数组，总长度大于64k。

② 语句

```
d1 = (double *) calloc (NUMBER, SIZE);
```

中的“(double *)”用来把calloc( )函数返回的字符指针转换为长实型指针。

③ 程序中使用的exp( )函数是一个系统库函数，它用来求实型量以e为底的指数。

第五章　数据库管理系统dBASEⅢ及其应用

美国Ashton-Tate公司开发的数据库管理系统dBASEⅡ，最初主要用于8位微型计算机。由于它具有功能较全、简单实用的优点，因此在国内外颇为流行。随着16位微型计算机的大量推出，dBASEⅡ很快就被移植到包括IBM PC在内的许多新机种上。与8位微型计算机相比，16位微型机的内存容量一般都在256KB以上，并可配接容量在10MB字节以上的硬盘作外存贮器。为了更有效地发挥硬件的功能，克服dBASEⅡ在实际应用中发现的一些不足和缺点，Ashton-Tate公司于1984年6月推出了一个新的数据库管理系统dBASEⅢ。

dBASEⅢ的主要部分都是用C语言编写的，在MS-DOS 2.0版或更高的版本支持下，它能在IBM PC上运行。无论在功能还是性能指标方面，它比dBASEⅡ都有明显的改善和提高。但是，就基本概念和主要功能而言，它与dBASEⅡ大体上保持一致。因此，本章对dBASEⅢ中与dBASEⅡ相同的那些基本概念只作简要阐述，重点将介绍dBASEⅢ所扩充的新功能以及dBASEⅢ和dBASEⅡ之间的差异。对dBASEⅡ还不熟悉的读者，建议在学习本章的同时，连同本书上册第六章一起进行学习。

第一节　概　述

1. dBASEⅢ的功能与特点

（1）数据库文件

dBASEⅢ是一个小型关系式数据库管理系统。一个关系式数据库可以看作为一张或多张由若干行、若干列所组成的表格。每一张这样的表格称为一个“关系”，它们在数据库中用文件的形式存放，叫做“数据库文件”。表格中的行称为记录，列称为字段。每个记录都有一个序号，叫做记录号，相当于该记录所对应的“行”在表格中的行号。每个字段都有一个名称，叫做字段名，相当于表格中的栏目。

关系式数据库管理系统具有提供关系与关系之间（即表格与表格之间或dBASEⅢ中的数据库文件与文件之间）相互对照、相互引用的能力。例如，图5-1中的两张表格分别是学生登记表和任课教师分配表，通过相互比较（查看“年级”这个栏目）就能发现，学生B的任课老师是丙。

dBASEⅢ的数据库文件均放在磁盘上，当需要使用时才把它从磁盘上读入内存进行处理，为此必须预先打开这个文件，为它安排一定的工作区。dBASEⅢ最多可以有10个工作区，因此允许同时打开10个数据库文件供用户使用。数据库文件使用之后，必须把它关闭，否则会引起操作出错。当一个文件打开之后，在没有关闭以前不能再次被打开使用。

数据库管理系统的许多操作都是以文件中的记录为单位进行处理的，因此，每个工作区中均有一个指针，它用来指向某个记录，该记录就称为当前记录，其记录序号就是指针的值。打开一个文件后，指针一般总是指向文件中的第1个记录。

序号	姓 名	年 级	性 别	住 址
1	A	3	男	8-301
2	B	2	女	4-502
3	C	1	女	4-518
4	D	4	男	7-304
5	E	3		

序号	姓 名	年 级	职 称	任课名称
1	甲	3	讲 师	数 学
2	乙	1	助 教	外 语
3	丙	2	讲 师	物 理
4	丁	4	副教授	计 算 机

图5-1 表格之间的相互对照

一个数据库文件形式上相当于一张表格，它的每一个记录就是表中的一行，记录中的每一个字段就是表中的一格。但是，为了操作处理的需要，除了在文件中给出记录序号、字段名和字段宽度之外，还必须指出每个字段的类型。dBASE Ⅲ规定，字段的类型有下列五种：

C ——字符串型
N ——数值型
L ——逻辑型
D ——日期型
M——备忘型

其中，日期型字段用来存放日期信息，如出生日期、偿还日期，休假日期等；备忘型字段用来存放一段长度可变的文字信息，一般作为备注或说明之用。

（2）数据库管理系统的功能

作为一个在16位微型计算机上开发的小型关系式数据库管理系统，dBASE Ⅲ是微型机在事务管理应用方面的一个通用工具，它能有效地进行数据的存贮、修改、分类、检索、统计、生成报表等各类管理业务。从用户角度来看，它具有下列功能：

① 数据库文件的定义

在向数据库输入数据以前，dBASE Ⅲ允许用户按照实际使用的要求预先定义数据库文件的结构，也允许对已有的数据库文件的结构进行修改。

② 数据的输入和更新

用户可以以记录或字段为单位向数据库文件输入数据。dBASE Ⅲ允许用户向数据库文件

添加记录、插入记录、删除记录、替换记录中的某些字段内容等。为了便于数据库文件的更新，系统提供了一组灵活的记录定位、编辑以及文字处理等功能的操作命令。

③ 数据的操作

dBASE Ⅲ允许用户对数据库文件中的数据进行各种常用的事务管理方面的操作。例如，检索、排序、统计和求和等等。

④ 报表的生成

数据处理操作的结果，可以在屏幕上显示出来，也可以作为新的文件存贮在数据库中，更为常用的是以报表形式打印出来。报表的格式可以由用户进行设计。

⑤ 应用程序的开发

dBASE Ⅲ的所有命令可以交互式（对话式）地使用，也可以通过一组控制命令（语句）组织起来自动地或成批地执行。因此用户可以方便地开发dBASE Ⅲ的应用程序。

⑥ 辅助功能

如文件管理、计算器功能、联机求助功能、文件转换功能等。

尽管dBASE Ⅲ基本功能与dBASE Ⅱ大体相同，但由于dBASE Ⅲ在实现方面比dBASE Ⅱ有了许多改进，例如检索算法和缓冲区管理算法的修改，缓冲区数目的增加，内存容量的扩大和使用效率的提高等，因此dBASE Ⅲ的性能比dBASE Ⅱ要好得多，在实际应用中更加受到用户的欢迎。表 5 - 1 是dBASE Ⅲ与dBASE Ⅱ主要性能的对比。

表5-1　dBASE Ⅲ与dBASE Ⅱ主要性能对比

性　　能	dBASE Ⅲ	dBASE Ⅱ
数据库文件中的最大记录总数	1,000,000,000	65,535
每个记录中最大字符个数	4,000	1,000
每个记录中最大字段个数	128	32
可同时使用的数据库文件个数	10	2
存贮器变量的个数	256	64
存贮器变量的最大字节数	6,000	1,536
数值运算的精度	16位（IEEE浮点格式）	10位
分类命令使用的关键字段个数	允许多个	允许一个

（3）数据库管理系统的操作

数据库中存放了一个或多个数据库文件，数据库管理系统的操作对象主要就是这些数据库文件。dBASE Ⅲ向用户提供了一组内容丰富的命令（在应用程序中称语句），它的全部功能几乎都体现在这些命令上。这些命令都是解释执行的，在使用方式上它们与BASIC语言比较类似，即每条命令既可以会话式地由dBASE Ⅲ立即执行，也可以预先组织成为一个程序（叫做“应用程序”），然后由DO命令启动或调用执行。

下面是dBASE Ⅲ所提供的按功能分组的命令一览表。

表5-2　dBASE Ⅲ命令分类表

类　别	命　令　名　称
建立文件类	COPY, COPY FILE①, CREATE, INDEX, JOIN, MODIFY COMMAND②, MODIFY LABEL①, MODIFY REPORT②, MODIFY STRUCTURE②, SAVE, SORT, TOTAL
数据添加与数据编辑类	APPEND②, BROWSE②, CHANGE②, DELETE, EDIT, INSERT, PACK, READ, RECALL, REPLACE, UPDATE, ZAP①
数据显示类	@…SAY②, ?, AVERAGE①, BROWSE②, COUNT, DISPLAY, LIST, REPORT②, SUM, TEXT①
记录定位类	CONTINUE, FIND, GOTO②, LOCATE, SEEK①, SKIP
数据库文件操作类	APPEND FROM, CLOSE①, COPY, DIR②, ERASE②, MODIFY STRUCTURE, REINDEX, RENAME, SELECT②, SORT②,USE②
其它类型文件操作类	MODIFY COMMAND②, CREATE/MODIFY LABEL①, CREATE/MODIFY REPORT①
存贮变量使用类	@…GET②, ACCEPT, AVERAGE①, CLEAR ALL②, CLEAR MEMORY, COUNT, DISPLAY MEMORY②, INPUT, READ, RELEASE, RESTORE, SAVE, STORE, SUM, WAIT②
程序设计类	ACCEPT, CANCEL, DO②, DO WHILE, ENDDO,DO CASE, CASE, OTHERWISE, ENDCASE, EXIT①,IF,ELSE, ENDIF, INPUT, LOOP,MODIFY COMMAND②,PARAMETERS①, PRIVATE①, PROCEDURE①, PUBLIC①, QUIT, RETURN②, WAIT②
参数设定和外围控制类	CLEAR②, EJECT, SET①, SET〈控制参数〉TO〈参数〉②, SET〈控制参数〉ON/OFF②,
辅助用户类	ASSIST①, DIR②, DISPLAY MEMORY②, DISPLAY STATUS, DISPLAY STRUCTURE, HELP②

注：①表示新增加的命令.　②表示功能或名称有修改的命令.

表中所列的各种命令将在下面各节中分别叙述，并在第六节中按字母顺序给出dBASE Ⅲ全部命令的语法结构及其简要解释，供读者使用时查阅。

2. dBASE Ⅲ的文件类型及文件管理操作

dBASE Ⅲ一共有九类专门格式的磁盘文件用来存贮信息，它们在dBASE Ⅲ进行数据管理操作时各有其特定的作用，现简单介绍如下（括号中为文件类型名）：

① 数据库文件（.DBF）

这是数据库中最基本的文件，用户的数据均以记录和字段(行和列)的形式存放在其中。数据库主要就是由若干这样的数据库文件所组成。dBASE Ⅲ允许用户建立和修改数据库文件并能方便地对文件中的数据进行检索、排序、统计计算等各种操作处理。

② 数据库备忘文件（.DBT）

数据库备忘文件是数据库文件(.DBF)的辅助文件，它用来存放.DBF文件中的备忘字段（MEMO）的可变长内容。同一个数据库文件中的所有MEMO字段的所有内容都存放在同一个备忘文件中。

数据库文件中每个记录最多可以有128个备忘字段，它们的内容虽然也是字符信息，但每个备忘字段可以包含多达4000个字符的信息。由于在创建具有MEMO字段的.DBF文件时，dBASE Ⅲ自动地创建了一个同名的.DBT文件，它把.DBF中所有备忘字段内容都存放在对应的.DBT文件中。所以.DBF文件中每个备忘字段只需占用10个字节的空间。

③ 索引文件（.NDX）

这是用于进行快速数据操作的一个辅助文件，它由索引命令（INDEX）所生成。文件中只包含排了序的字段名以及对应的指针（记录序号）。由于数据库文件是以物理顺序（数据输入的先后顺序即记录序号）排列的，有了索引文件，就可以按逻辑顺序（字段内容的顺序）来使用数据库文件中的数据，而不必对数据库文件在物理上进行重排。

④ 命令文件或应用程序文件（.PRG）

该文件由一连串的dBASE Ⅲ命令所组成，用来完成特定的应用功能，所以叫做dBASE Ⅲ的应用程序文件。用户可以通过DO命令启动执行应用程序。应用程序文件是一个ASCII文件，它可以用dBASE Ⅲ中的MODIFY COMMAND命令或MS-DOS中的EDLIN或WORD STAR来编辑生成。

⑤ 屏幕格式文件（.FMT）

屏幕格式文件主要是@…SAY、@…GET和NOTE三种命令所组成的一种特殊的程序，它允许用户在数据输入和数据输出时使用由用户自行安排的屏幕格式。屏幕格式文件可以由dBASE Ⅲ的一个实用程序dFORMAT来生成，也可以使用编辑程序自行编制。

⑥ 标签格式文件（.LBL）

标签格式文件中包含了LABEL命令显示或打印标签时所需要的格式描述信息。它由MODIFY LABEL命令编辑设计而成。

⑦ 存贮变量文件（.MEM）

dBASE Ⅲ在进行数据处理的时候，常常需要使用一些存贮变量来存放常数、中间结果、最终结果等等，一次最多可以使用256个变量。这些变量的内容可以通过SAVE命令保存到磁盘文件中去供以后使用，该文件就叫做存贮变量文件。用RESTORE命令可以把存贮变量文件中的内容恢复到内存中。

⑧ 报表格式文件（.FRM）

报表格式文件中包含了用来生成报表的REPORT语句所需要的报表格式信息，它可以使用MODIFY REPORT命令建立或修改。

⑨ 正文文件（.TXT）

正文文件主要用来作为dBASE Ⅲ与其它软件的接口，它们都是ASCII文件，可以用MS-DOS的TYPE命令打印和显示出来。数据库文件的正文文件可以由dBASE Ⅲ的COPY命令建立，也可以通过APPEND FROM命令把正文文件内容添加到数据库文件中去。

图5-2是dBASE Ⅲ数据库管理系统的示意图，它表示了上述各类文件在系统中的作用和相互关系。

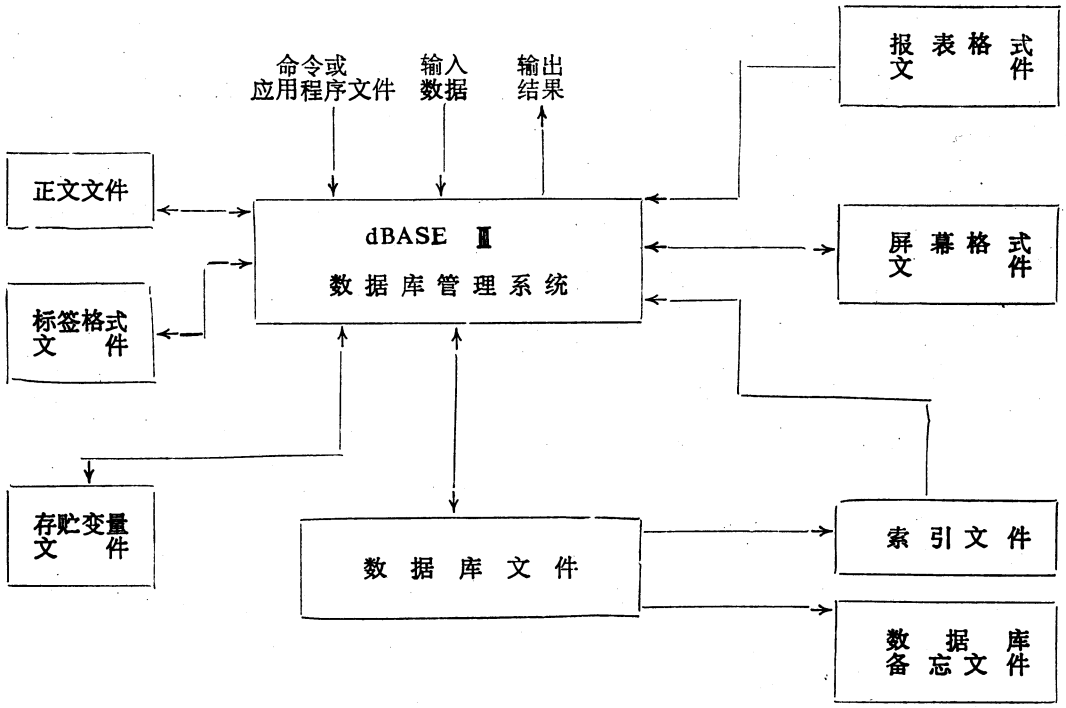


图5-2 dBASEⅢ数据库管理系统示意图

与MS-DOS的文件管理类似，dBASE Ⅲ对于上述各种文件也提供了一些通用的文件管理操作，如列出文件目录，文件重命名，删除文件，复制文件和编辑文件等。下面是这些命令的简单介绍。

* 列出文件目录(DIR)

命令格式：

DIR〔<驱动器号：>〕〔<路径>〕〔<文件名>〕

本命令把指定驱动器和指定路径下的一个或一组文件（<文件名>可以出现*、？等字符）的目录显示出来，如果不指出任何参数，仅打入“DIR”，则把缺省驱动器上的所有数据库文件的目录显示出来。

* 文件重命名（RENAME）

命令格式：

RENAME 〈老文件名〉 TO 〈新文件名〉

- 删除文件（ERASE）

命令格式：

ERASE 〈文件名〉

- 复制文件（COPY FILE）

命令格式：

COPY FILE 〈源文件名〉 TO 〈目标文件名〉

- 正文编辑（MODIFY COMMAND）

命令格式：

MODIFY COMMAND 〈文件名〉

这是一条调用dBASE Ⅱ的正文编辑程序的命令，它可以用来编辑修改任何ASCII文件。如果指出的文件已存在，则读入内存进行编辑修改，原来的文件改作后备文件（类型名为·BAK）。如果指出的文件不存在，则用来建立一个新的正文文件，若不指出类型名，类型名即为.PRG。编辑操作中所使用的光标控制键和编辑键如表5-3所示。

表5-3 MODIFY COMMAND命令使用的控制键

光标控制键	功能	编辑命令键	功能
←	光标左移	退格键	删去光标左面的一个字符
→	光标右移	Del	删去光标所在处的一个字符
↑	光标上移	CTRL-Y	删去光标所在处的一行字符
↓	光标下移	CTRL-T	删除从光标起直到下一字段开始处的所有字符
PgDn	屏幕下滚18行	CTRL-V (Ins)	在插入方式与迭写方式之间切换
PgUp	屏幕上滚18行	CTRL-N	在光标处插入空行
Home	光标移至现行字或前一字之首字符处	CTRL-KR	把一个文件读入被编辑文件的光标处
End	光标移至下一字之首字符处	CTRL-KW	把整个文件写到另一个文件
↵	光标移至下一行之首	CTRL-Q (ESC)	不保存文件而退出编辑程序
		CTRL-W	保存文件并退出编辑程序

- 打印文件（TYPE）

命令格式：

TYPE 〈文件名〉〔TO PRINT〕

本命令可以显示或打印输出一个ASCII文件。

3. dBASE Ⅲ的启动

（1）系统组成

dBASE Ⅲ1.00版由两张软盘片组成，一片是系统盘，另一片是实用程序盘。系统盘上的文件主要有：

DBASE.EXE	dBASE Ⅲ的总控程序及常驻内存模块
DBASE.OVL	dBASE Ⅲ的可覆盖模块
ASSIST.HLP	ASSIST命令使用的文件
HELP.DBS	HELP命令使用的文件
CONFIG.SYS	MS-DOS的系统配置文件，其中规定系统一次最多可以打开20个文件(FILES＝20)，缓冲区个数为24(BUFFERS＝24)
READ.ME	dBASE Ⅲ的使用说明

实用程序盘上的文件较多，主要的有：

DFORMAT.EXE	格式文件生成程序
DFM.MSG	DFORMAT的联机说明
DCONVERT.EXE	dBASE Ⅱ和dBASE Ⅲ文件转换程序

dBASE Ⅲ系统比较庞大，仅常驻内存部分（DBASE.EXE)大约就有113KB，再加上可覆盖模块及MS-DOS本身，所以IBM PC机至少需要有256KB内存容量才能运行dBASE Ⅲ。

dBASE Ⅲ的原版软盘片有一定的保护设施，用DISKCOPY命令复制系统盘会引起系统盘的破坏。为了得到一张系统盘的备份，可以使用“COPY A：*.* B：”命令进行。但备份盘无法直接使用，仅供系统盘上某些文件不小心受到破坏以后重新恢复时使用。在IBM PC/XT机器上如果要使用硬盘运行dBASE Ⅲ，则应该先用命令“COPY A：*.* C：”把dBASE Ⅲ系统盘上的文件复制到硬盘上，然后保留系统盘在A驱动器中，就可以运行硬盘上的dBASE Ⅲ了。

（2）系统启动

为了启动dBASE Ⅲ运行，在MS-DOS提示符之下，打入命令：

DBASE〈CR〉

于是屏幕上将会出现下列dBASE Ⅲ的签到信息：

```
dBASE III version 1.00 14 June 1984 IBM／MSDOS * * *
COPYRIGHT (c) ASHTON-TATE 1984
AS AN UNPUBLISHED LICENSED PROPRIETARY WORK.
ALL RIGHTS RESERVED.
                    ⋮
Press the F1 key for help
Type a command (or ASSIST) and press the return key (↵)
```

屏幕上最末行的“.”是dBASE Ⅲ的提示符，它表示此时dBASE Ⅲ正处于对话方式，等待着接受用户打入的命令。

（3）系统控制参数

dBASE Ⅲ的控制结构设计得非常灵活，它允许用户按照自己的要求去选择合适的工作方式，这是通过设定和修改dBASE Ⅲ的系统控制参数来实现的。

dBASE Ⅲ的系统控制参数（状态)可以用DISPLAY STATUS命令在屏幕上显示出来。下面是系统初态时所有控制参数的状态，这些参数可以分成三组，它们的含义在下面分别作简单解释（读者可以暂时跳过这一段，待学完本章后再阅读）。

```
display status

File search path:
Default disk drive: B:
ALTERNATE  - OFF   DEBUG      - OFF   ESCAPE     - ON    MENU       - OFF
BELL       - ON    DELETED    - OFF   EXACT      - OFF   PRINT      - ON
CARRY      - OFF   DELIMITERS - OFF   HEADING    - ON    SAFETY     - ON
CONFIRM    - OFF   DEVICE     - SCRN  HELP       - ON    STEP       - OFF
CONSOLE    - ON    ECHO       - OFF   INTENSITY  - ON    TALK       - ON
UNIQUE     - OFF

Margin =     0

Function key  F1  - help;
Function key  F2  - assist;
Function key  F3  - list;
Function key  F4  - dir;
Function key  F5  - display structure;
Function key  F6  - display status;
Function key  F7  - display memory;
Function key  F8  - display;
Function key  F9  - append;
Function key  F10 - edit;
```

① 工作方式开关

* **ALTERNATE** 把所有键盘输入信息及屏幕输出信息记录到一个正文文件之中，以备查用。
* **BELL** 在输入无效数据时，发出报警声。
* **CARRY** 添加或插入一个记录时，前面一个记录中的数据自动复制在新记录中。
* **CONFIRM** 数据输入时，只有按下〈CR〉后光标才移到下一个字段；OFF状态时，字段填满后光标会自动移到下一字段。
* **CONSOLE** OFF状态时，显示器无输出，但键盘输入和打印输出不受影响。出错信息仍然显示。
* **ECHO** 应用程序执行时使每一条语句在屏幕上输出。
* **DEBUG** 使ECHO处于ON状态时的输出信息送到打印机上打印出来而不在屏幕上输出。
* **STEP** 应用程序执行时，每执行一条语句就暂停一次。
* **TALK** ON状态时使语句执行后dBASE Ⅲ的反应在屏幕上显示出来，即系统工作为交互方式，OFF状态时不显示反应信息。

（上述ECHO、DEBUG、STEP和TALK四个开关均为调试工具）

* DELETED　不处理有删除标记的记录（INDEX和REINDEX除外）。

* DELIMITER　ON状态时用“：”或自定义的分隔符来指出一个字段的宽度，OFF状态时用反视频显示来指出字段的宽度。

* ESCAPE　使“ESC”键起作用，用来终止一条命令或程序的执行，回到dBASE Ⅱ提示符状态。

* EXACT　ON状态时，两个字符串完全相同才认为相等；OFF状态时只要第2字符串与第1字符串的开始部分相同就可以认为相等。例如：

'ABC' = 'ABCDEF' 的值为假
'ABCDEF' = 'ABC' 的值为真　（OFF状态时）
'ABCDEF' = 'ABC' 的值为假　（ON状态时）

* FIXED　ON状态时表示所有数值计算结果（包括整数结果）均以固定位数小数输出，OFF状态时则由实际计算的结果和参数DECIMALS而定（不影响整数结果）。缺省状态为OFF。

* HEADING　使DISPLAY，LIST，SUM和AVERAGE等命令执行后，输出信息中包含有一行栏标题，OFF状态时不再显示栏标题。

* HELP　用户打入有错误的命令后，使能输出“Do you want some help？（Y/N）”，需要时并给出说明信息。

* INTENSITY　在EDIT和APPEND命令时，字段以反视频方式显示。

* MENU　在全屏幕操作方式下，屏幕上部显示光标控制键菜单。ASSIST命令执行后，自动设置为ON。

* PRINT　连接打印机，相当于MS-DOS中的〈Ctrl-P〉键，使信息既在屏幕上也在打印机上输出。

* SAFETY　在文件操作中，当可能破坏盘上已有的同名文件时提供警告信息。

* UNIQUE　用INDEX命令建立索引文件时，使索引中无重复关键字。

② 工作参数

* ALTERNATE　ALTERNATE开关处于ON状态时，用于定义记录信息的一个正文文件名。

* DECIMALS　当FIXED处于ON状态时，用来指定输出结果的小数位数；当FIXED处于OFF，它只对除法、开平方、对数、指数起作用。缺省值为两位。

* DELIMITER　当DELIMITER开关处于ON状态时，用于标出字段宽度的自定义字符，缺省时为“：”。

* COLOR　使用彩色显示器时用于定义所选择的底色、显示色和边

界色。

* DEFAULT　存取文件时缺省的盘驱动器号。
* PATH　存取文件时，若现行目录中找不到指定文件，用于指定进一步查找的路径。
* DEVICE　@命令的输出设备，可以是屏幕显示器或打印机。
* FORMAT　使APPEND、CHANGE、EDIT和INSERT命令的输出按指出的屏幕格式文件所规定的格式进行显示；若未指出格式文件，则按系统的标准格式显示。
* MARGIN　打印输出时左边界的位置，缺省值为0。
* FILTER　选择条件，使dBASE Ⅲ所有操作都只处理数据库文件中满足指定条件的那些记录。
* INDEX　已打开的索引文件一览表，表中第一个文件为主控文件。
* PROCEDURE　已打开的过程文件，其中所包含的过程（最多32个）可以任意调用。该文件用CLOSE PROCEDURE关闭。
* RELATION　在两个数据库文件之间建立的关系，这两个文件必须都按相同关键字建立索引。建立关系后辅文件的记录指针随主文件记录指针的改变而改变，反之不成立。

③ 功能键指派

IBM PC有十个功能键，在dBASE Ⅲ初态时，它们的功能分配前面已经给出，每个键相当于执行一个或多个dBASE Ⅲ命令，其中每个命令的字符串结尾处的“；”表示回车符〈CR〉。

（4）系统控制参数的修改

上述dBASE Ⅲ的各种控制参数，为用户使用dBASE Ⅲ提供了极大的灵活性，这些参数的初始状态（缺省状态）是大多数用户正常情况下的工作方式。如果用户需要按照自己的特殊要求使用dBASE Ⅲ，只要修改上述有关参数或开关状态即可。修改控制参数的方法有三种，下面分别叙述。

① 全屏幕成批修改

在dBASE Ⅲ控制下，任何时候只要输入一条SET(参数设定)命令，就可以以全屏幕方式成批地修改系统的控制参数。修改操作是会话式进行的，系统首先显示出当前的参数状态，然后用户用光标选择有关修改项目并打入新的参数，即可达到修改目的。

② 单个参数修改

如果只要修改个别参数，则使用“SET…ON/OFF”或“SET…TO…”命令即可。例如，若需要定义一个ALTERNATE文件并打开ALTERNATE开关，只要打入下面两条命令：

```
SET ALTERNATE  TO〈文件名〉
SET ALTERNATE  ON
```

若需要指定缺省的盘驱动器号为C：（硬盘），则应使用命令：

```
SET DEFAULT TO  C：
```

若需要把功能键F10的功能修改为SET命令（全屏幕修改控制参数），只要打入命令：

SET FUNCTION 10 TO 'SET;'

③ 系统启动时修改控制参数

如果需要在dBASE Ⅲ启动后立即设定有关的控制参数（即改变某些约定的缺省状态或参数），那么需要预先在系统盘上建立一个名字为"CONFIG·DB"的文件，该文件是一个正文文件，它可以用MS-DOS的EDLIN或WORDSTAR 或"COPY CON：CONFIG.DB"命令建成。文件中每一行均呈下述形式：

〈控制参数名〉=〈参数值〉

例如，前面所列举的一些参数修改在CONFIG.DB文件中可以表达成为：

ALTERNATE=〈文件名〉

ALTERNATE=ON

DEFAULT=C：

F10='SET;'

当dBASE Ⅲ启动成功之后，它将自动地在系统盘上查找CONFIG·DB文件。若找到，则装入内存并按照要求改变系统中有关的控制参数。

如果CONFIG·DB文件中除了参数设定的命令之外，还有命令

COMMAND=DO 〈应用程序名〉

时，那么系统就自动装入该应用程序并执行这个程序，这好象MS-DOS中的自动启动批处理文件一样。

最后需要提请读者注意的是，当dBASE Ⅲ系统盘上的剩余空间已经不多时，为了系统盘上文件的安全起见，建议所有数据库文件和工作文件不要再放在系统盘上（即A盘上），而应存放在B盘或C盘上。为了今后操作的方便，dBASE Ⅲ启动成功后，可首先打入一条参数设定命令：

SET DEFAULT TO B：（或C：）

使B盘或C盘成为缺省的文件存取磁盘，或者用CONFIG.DB文件自动设置。

4. dBASE Ⅲ的求助设施

dBASE Ⅲ是一个人机接口设计得相当友好的软件系统。在使用过程中它可以向用户提供许多帮助，这主要体现在HELP和ASSIST两条命令上。

（1）求助命令（HELP）

HELP命令的功能是以联机方式向用户提供有关dBASE Ⅲ的命令及其使用的说明书，它与dBASE Ⅱ的HELP命令功能相同，但说明文字组织得更有条理，层次分明。HELP命令的执行有下列三种方式：

① 自动执行方式

当用户向dBASE Ⅲ打入一条有语法错误的命令时，系统除了指出语法错误不能执行这条命令以外，还询问用户：Do you want some help? （Y/N）如果用户回答需要帮助（Y），dBASE Ⅲ就自动执行HELP命令，对刚刚打入的有错误的命令中的谓语动词（命令名）提供说明书，供用户查对输入命令中的错误。下面是一个具体例子。

例5-1　HELP命令的自动执行

设用户打入了一条语法上有错误的命令，在系统询问要不要帮助时回答“要”（Y），于是dBASE Ⅲ自动执行HELP命令，针对用户打入的命令给出如下说明：

```
  RENAME OLDFILE.DBE NEWFILE.DBF
Unrecognized phrase/keyword in command
                ?
RENAME OLDFILE.DBE NEWFILE.DBF
Do you want some help? (Y/N) Yes
```

RENAME

```
                        RENAME
                        ======

Syntax          RENAME <current file.extension> TO <new file.extension>

Description :   Changes the name of a file.  An open file cannot be
                RENAMEd.  The default file extension is .dbf.


PgUp=previous screen, Esc=exit HELP, ^Home=previous menu, or enter command:
                ENTER >
```

其中，标题即为用户打入的命令名，然后是它的语法公式和简要说明及进一步参考的命令，最后一行是提示信息，它向用户指出接下去的操作可以有四种选择：翻到联机说明书的前一页；退出HELP状态返回dBASE Ⅲ提示符状态；退到HELP的上一级菜单；直接打入所要进一步阅读的命令名、函数名或其它求助关键字。

② 按需要求助方式

用户在操作过程中，如果忘记了某个命令或某个函数的语法和功能，则可以随时在dBASE Ⅲ提示符之下打入HELP命令请求帮助，命令格式如下：

HELP　〈命令名/函数名/其它〉

例如，需要了解dBASE Ⅲ新增加的函数ROUND的含义时，打入HELP ROUND命令，即可得到下列信息，其中右上角是求助关键字，一般是命令名、函数名和菜单名。

ROUND()

```
                      Round Function
                      ==============

Syntax        :  ROUND(<expN>,<decimal>)

Description      Rounds the value of a numeric expression to the
                 specified decimal place.
```

```
Example      :  ROUND(14.746321,2) returns the number 14.750000

See also     :  INT()

 PgUp=previous screen, Esc=exit HELP, ^Home=previous menu, or enter command

                    ENTER >
```

③ 会话式求助

用户如果需要系统地了解dBASE Ⅲ的一些基本概念、操作方法、命令和函数的介绍等，则可以通过会话方式取得系统的帮助，系统将会把存放在机器里的一本手册（即系统盘上的HELP.DBS文件）提供给用户联机使用。

进入会话式求助只要打入命令HELP或功能键F1即可，此时屏幕上显示出一张菜单（说明书目录）：

```
Help                                                          MAIN MENU

maximum help -

                        dBASE III Main Menu
                        ===================


                        1 - Getting Started
                        2 - What Is a ...
                        3 - How Do I ...
                        4 - Creating a Database
                        5 - Using an Existing Database
                        6 - Commands and Functions




 ↑↓=menu selection, PgUp=previous screen, Esc=exit HELP, or enter command.

                    ENTER >
```

它表示整个说明书共分成六个部分，其内容大致如下：

第1部分　介绍HELP命令本身的功能及其使用。

第2部分　介绍dBASE Ⅲ中的一些基本概念，如命令、表达式、字段、文件、关键字段、内存变量和运算符等。

第3部分　介绍dBASE Ⅲ的操作方法：怎样退出dBASE Ⅲ，怎样联接打印机，怎样保存文件，怎样显示文件目录，怎样对文件重新命名、删除和复制，怎样打开数据库文件等。

第4部分　介绍如何创建一个数据库文件，其中包括文件的命名、字段的命名、字段类型及宽度的选择等。

第5部分　介绍如何使用现有的数据库文件，如怎样对数据库文件添加、删除、复制、编辑、查找记录，怎样修改数据库文件的结构，怎样检索数据，怎样做索引等。

第6部分　dBASE Ⅲ的命令、函数一览表。

用户可以根据自己的需要选择其中的有关部分获得有用的信息。

（2）辅导命令（ASSIST）

对于刚开始学习使用dBASE Ⅲ的用户来说，要正确地记住各种命令的格式和功能并熟练地使用它们是有一定困难的。虽然有HELP命令的帮助，但毕竟不够方便。为此，dBASE Ⅲ比dBASE Ⅱ增加了一条ASSIST命令，这条命令的功能是：使用户以菜单方式来使用dBASE Ⅲ的大部分命令，减轻用户记忆各种命令和不断查阅手册的负担。

为了进入菜单方式（即ASSIST方式）对数据库文件进行各种操作，首先打入功能键F2或命令：

ASSIST

于是屏幕上出现一些有关光标控制键的使用说明，它告诉用户在ASSIST方式下选择菜单项目的几个专用键是：

→	选择光标右移一个菜单项目	Home	回到第1个菜单项目
←	选择光标左移一个菜单项目	↑	退回到上一个菜单
End	回到最右边的菜单项目	↓	进入到下一个子菜单

然后按下↓键，屏幕上就显示出ASSIST方式下的主菜单：

dBASE III Assistant　　READY

Set Up　Modify　Position　Retrieve　Organize　Utilities

它表示dBASE Ⅲ的大部分命令已被归纳为六大组，每一组与一个菜单项目相对应，这六个菜单项目的功能及有关命令大体如下：

① Set Up　包括打开一个已有的数据库文件，建立一个新的数据库文件，建立报表格式文件，建立标签文件等。

② Modify　包括对数据库文件进行编辑修改的一些命令，如记录的添加、插入、删除、编辑、显示等。

③ Position　包括用来查找所需记录的各种记录定位命令。

④ Retrieve　包括用于检索数据和统计计算的各种命令，如显示、求和、求均值、标签和报表生成等命令。

⑤ Organize　包括用来对数据库文件进行重新组织的一些命令，如索引、排序、复制、整理（PACK）命令等。

⑥ Utilities　这类命令用于常规的文件维护工作，如列文件目录、重命名、删除、复制等。

用户可以根据需要选择上述有关项目，然后系统又会给出新的说明信息和下一层子菜单，供操作时选用。如果在操作过程中还有什么困难，则可以使用功能键F1，请求HELP命令给出更详细的联机求助信息。

总之，利用ASSIST命令可以使dBASE Ⅲ成为一个按菜单方式操作的系统，这样，用户使用起来就十分方便。

第二节　dBASE Ⅲ的语法基础

1. 语句（命令）和表达式

（1）语句（命令）

dBASE Ⅲ的命令是指通过键盘向计算机发出的指令，在程序中则叫做语句。每条命令都有一定的结构格式，叫做该命令的语法。

dBASE Ⅲ的大部分命令都以一个谓语动词（叫做“命令名”）开始，宾语就是它的操作对象，有时还加上一些短语作为该命令的修饰或补充。命令长度超过一行时，可以以“;”作为结尾，表示下一行是续行。不同命令有不同的语法。例如，用来对数据库文件进行操作的命令具有下列格式：

〈命令名〉〔〈范围〉〕〔〈表达式表〉〕〔FOR/WHILE〈条件〉〕

其中，〈范围〉用来指出该命令对数据库文件中哪些记录起作用。如果缺省的话，那末或者是指文件中的现行记录，或者是指文件中的全部记录，由各命令自行规定。〈范围〉的取值可以有下面三种：

RECORD n	表示只对文件中第n个记录进行处理。
NEXT n	表示对文件中从当前记录开始的n个记录都进行处理。
ALL	表示对文件中所有的记录都进行规定的处理。

〈表达式表〉是一组用逗号隔开的表达式。当表达式为字段名时，表示该命令只对规定范围内的记录中的指定字段进行操作处理；当〈表达式表〉缺省时，则应对记录中全部字段进行处理。

FOR〈条件〉是一个条件短语，其中〈条件〉表示条件表达式。当命令中包含有FOR这个短语时，表示应对规定范围内所有满足规定条件（使表达式取值为真）的记录进行该语句所指定的操作。

WHILE〈条件〉也是一个条件短语，它表示从当前记录开始，在指定的范围内，只要条件成立，该命令就应继续执行命令所规定的操作。如果条件不成立，则命令执行完毕，指针定位在第一个条件为假的记录上。

dBASE Ⅲ各条命令的语法及其功能简介将在第六节中给出，以便读者查阅。

（2）表达式

与通常的程序设计语言一样，表达式也是dBASE Ⅲ语句（命令）中的一个重要组成部分。dBASE Ⅲ的表达式由下列五个成分组成：

* 运算符
* 常数
* 存贮变量
* 字段变量（即字段名）
* 函数

根据表达式的值的类型，它可以分成数值表达式（算术表达式）、逻辑表达式（条件表达式）、字符串表达式和日期表达式。

表达式中的运算共有四类，按优先级为序在表5-4中分类列出。

表5-4　dBASE Ⅲ的运算符

算术运算符	关系运算符	逻辑运算符	字符串运算符
单目+，－	＜	·NOT·	$，+，－
**一或^	＞	·AND·	
*，/	=	·OR·	
+，－	＜＞或#		
	＜=		
	＞=		

其中，六个关系运算符的优先级是相同的，字符串运算符$和+/－优先级也相同。“$”表示子字符串比较。例如，A$B的值仅当A字符串为B字符串的子串或两者相等时为真；并置运算符“+”表示把两个字符串合并成一个串；“－”的功能与“+”相同，但它把第1个字符串尾部的空格移至结果字符串的尾部。

同一个表达式中出现不同类型的运算符时，其优先次序为：

① 算术运算符和字符串运算符。

② 关系运算符。

③ 逻辑运算符。

优先级相同时按从左向右的次序进行运算，括号（）可以提高其中运算符的优先级。

2．常量、字段变量和存贮变量

（1）常量和变量

dBASE Ⅲ中的常量和变量有四种类型：数值型、逻辑型、字符串型和日期型。

数值型常量可以是整数或实数。例如5，3.14159，－0.00001等。数值型数据的位数可达15位有效数字（小数点不算在内）。

逻辑型常量只有两个：“真”用·T·或·t·表示，“假”用·F·或·f·表示。赋值时可以用·Y·或·y·代替真，用·N·或·n·代替假。

字符串常量是使用单引号或双引号或方括号括起来的一串字符，如‘ABCD’，“Example of a string”等。

日期型常量用来表示一个日期，其形式为mm/dd/yy（月/日/年）。例如11/05/85表示1985年11月5日，5/11/84表示1984年5月11日。

变量有字段变量和存贮变量两种。字段变量用字段名给出，它的类型就是该字段的类型（备忘类型除外，因为此类字段的名一般不会在表达式中出现）。表达式中的字段变量可以是单个变量，也可以是一组变量，这取决于包含该表达式的命令的作用范围是单个记录还是一组记录。字段变量的值就是被处理的记录中对应字段的内容。

存贮变量是独立于数据库文件存放的变量，它可以用来存放常数、中间结果和最终结果，也可以用来控制应用程序的运行。dBASE Ⅲ允许使用的存贮变量已由dBASE Ⅱ的64个增加到256个，且最多可以占用6000字节内存容量。

（2）存贮变量的赋值命令

存贮变量的赋值可以由STORE，ACCEPT，INPUT和WAIT等命令来完成，这些命令简述如下：

① 存贮变量赋值命令（STORE）

命令格式：

STORE〈表达式〉TO〈存贮变量表〉

或　　〈存贮变量〉＝〈表达式〉

这条命令用来给一个或几个存贮变量赋值。例如：

```
STORE .F. TO Truth
STORE '04/27/84' TO Cdl
STORE 203 TO A, B, C
Tax = Cost * 0.06
```

② 存贮变量接受命令（ACCEPT）

命令格式：

ACCEPT〔〈提示信息〉〕TO〈存贮变量〉

这条命令在屏幕上显示出〈提示信息〉之后，就等待着用户在键盘上输入一串字符，它将赋值给指出的〈存贮变量〉。例如：

```
ACCEPT "Enter your name: " TO Name
```

③ 存贮变量输入命令（INPUT）

命令格式：

INPUT〔〈提示信息〉〕TO〈存贮变量〉

这条命令与ACCEPT类似，但它既可输入字符串，也可输入数值、逻辑值或日期，既可输入常量，也可输入表达式。当输入表达式时，系统先计算出表达式的值，然后赋值给存贮变量。

④ 等待命令（WAIT）

命令格式：

WAIT〔〈提示信息〉〕〔TO〈存贮变量〉〕

执行本命令时，系统给出〈提示信息〉之后就等待用户输入。如果命令中不出现〔TO〈存贮变量〉〕，则用户任敲一键就可以结束等待状态，否则，用户敲入的一个字符就作为〈存贮变量〉的值。

⑤ 编辑修改命令（READ）

命令格式：

@〈行号，列号〉GET〈变量〉[PICTURE〈子句〉]〔RANGE〈表达式，表达式〉〕

READ

格式输入命令与READ命令需配合使用，READ命令用于将@…GET命令中的存贮变量（它已预先赋值）或字段变量的内容进行编辑修改，RANGE用来指出变量取值的限制范围，PICTURE子句用来规定数据的格式。

（3）存贮变量的输出命令

存贮变量内容的输出可以用专门的命令进行，也可以与表达式统一起来进行处理。有关的命令介绍如下：

① 显示存贮变量命令（DISPLAY MEMORY或LIST MEMORY）

命令格式：

DISPLAY MEMORY 〔TO PRINT〕

或 LIST MEMORY 〔TO PRINT〕

这两条命令的功能都一样。它们在屏幕上或打印机上输出当前所有存贮变量的有关信息，如存贮变量名、性质、类型、值，并给出已定义的变量数目和占用的内存字节数，以及还可使用的存贮变量数及内存字节数等统计数字。显示过程中满一屏幕时DISPLAY命令会暂停输出，而LIST命令则否。

② 输出命令（？或??）

命令格式：

？〈表达式表〉

或 ??〈表达式表〉

这两个命令都用来计算并输出表达式的值。？命令先执行回车换行后再输出，??命令则从屏幕光标位置处进行输出。

③ 格式输出命令（@…SAY）

命令格式：

@〈行号，列号〉〔SAY〈表达式〉〔PICTURE〈子句〉〕〕〔CLEAR〕

在屏幕上指定的行、列位置处，按照PICTURE〈子句〉的格式规定，输出表达式的值，PICTURE〈子句〉的作用是指出数据的输出格式，PICTURE字符请参考第六节。〔CLEAR〕表示把光标开始处到屏幕结束处均擦除干净。格式输出命令和格式输入命令常常组合成一个命令使用，此时，SAY子句输出的多数是提示信息。

（4）存贮变量的保存、恢复和删除

有些应用程序中存贮变量的个数太多，或者由于某种原因需要把它们全部或其中的一部分作为一个磁盘文件（存贮变量文件）保存起来，则可以使用SAVE命令，必要时再把它们从盘上读入内存（RESTORE）。这两条命令的格式分别是：

SAVE TO〈文件名〉〔ALL LIKE／EXCEPT〈成组存贮变量名〉〕

RESTORE FROM〈文件名〉〔ADDITIVE〕

其中〈成组存贮变量名〉是指含有百搭字符“？”（任意一个字符）和“*”（任意一组字符）在内的存贮变量名，它往往代表一组存贮变量。下面是SAVE命令和RESTORE命令的若干例子：

SAVE TO B：Myfile 把全部内存变量保存在B盘Myfile文件中。

SAVE ALL LIKE d* TO Myfile 把全部以d开头的存贮变量保存在缺省盘中的Myfile文件中。

SAVE ALL EXCEPT ？A* TO Myfile 把除了名字中第2个字母为A的存贮变量以外的所有存贮变量保存在Myfile文件中。

RESTORE FROM Myfile	从存贮变量文件Myfile中读出全部存贮变量装入内存，内存中原有变量删去。
RESTROE FROM Myfile ADDITIVE	从Myfile中读出全部存贮变量添加到内存中，内存中原有变量保持不变，但同名变量的值被存贮变量文件中的值所替换。

存贮变量的一部分或全部均可删除，以便释放它们所占的内存空间供作它用，这是用RELEASE，CLEAR MEMORY和CLEAR ALL命令来完成的，它们的格式分别为：

RELEASE〔〈存贮变量表〉〕〔ALL〔LIKE／EXCEPT〈成组存贮变量名〉〕〕

CLEAR MEMORY

CLEAR ALL

其中CLEAR MEMORY和RELEASE ALL作用完全一样，用来删去全部存贮变量；CLEAR ALL则除了删除全部存贮变量之外，还要关闭所有已打开的数据库文件及有关联的索引文件、屏幕格式文件、备忘文件等，通常用来使dBASE Ⅲ恢复为初态。

（5）全局存贮变量和局部存贮变量

为了在应用程序中更方便、更有效地使用存贮变量，dBASE Ⅲ把存贮变量区分为全局变量和局部变量两种。所谓全局变量，是指在任何命令和任何一级应用程序中均可使用的存贮变量，它可以在dBASE Ⅲ提示符之下用LIST（或DISPLAY）MEMORY命令显示出来，且只能用RELEASE(或CLEAR MEMORY)命令才能删除。把存贮变量说明为全局变量的方法有两种，一是在应用程序之外，用对话方式定义并赋值的存贮变量均为全局变量，二是在任何一级应用程序中用PUBLIC命令说明的存贮变量也都是全局变量。PUBLIC命令的格式是：

PUBLIC〈存贮变量表〉

在应用程序中未说明为全局变量的其它存贮变量都是局部变量。局部变量只在使用它的应用程序及更低层的应用程序中有效，程序执行结束时自动被删除。

如果局部变量的名与高层应用程序中的局部变量或全局变量同名，为了避免同名变量在使用上的混淆，必须用PRIVATE命令把这些同名局部变量宣布为隐蔽局部变量。PRIVATE命令的格式如下：

PRIVATE〔ALL〔LIKE／EXCEPT〈成组存贮变量名〉〕〕〔〈存贮变量表〉〕

进入某级应用程序时，在高层应用程序中或全局变量中，凡与被PRIVATE命令宣布为隐蔽局部变量同名的那些变量均被藏起来，只有当该应用程序执行结束才被恢复。

下面是说明全局变量和局部变量概念的一个例子。

例5-2 全局变量与局部变量

假设有一应用程序A，它又调用另一个应用程序A_1，它们的程序清单如下：

A.PRG

```
X = 1
Y = 2
Z = 3
? "PROGRAM  A: "
```

A_1.PRG

```
? "PROGRAM  A1: "
PRIVATE  X
PUBLIC  AY
X = 10
```

```
LIST  MEMORY                    Y=20
DO  A1                          AX=11
? "PROGRAM  A:"                 AY=22
LIST  MEMORY                    LIST  MEMORY
RETURN                          RETURN
```

然后，在dBASE Ⅲ提示符状态下，输入下列命令：

```
·SET  DECIMAL  TO  5
·SET  FIXED  ON
·RELEASE  ALL
·P=SQRT(3)
   1.73205
·DO  A
```

于是，系统将执行应用程序A，并输出如下信息：

```
1.00000
2.00000
3.00000
PROGRAM A:
P           pub   N          1.73205  (         1.73205081)
X           priv  N          1.00000  (         1.00000000)      B:A.prg
Y           priv  N          2.00000  (         2.00000000)      B:A.prg
Z           priv  N          3.00000  (         3.00000000)      B:A.prg
    4 variables defined,      36 bytes used
  252 variables available.   5964 bytes available
PROGRAM A1:
10.00000
20.00000
11.00000
22.00000
P           pub   N          1.73205  (         1.73205081)
X           priv  (hidden)  N          1.00000  (         1.00000000)B:A.p
                                                                      rg
Y           priv  N         20.00000  (        20.00000000)      B:A.prg
Z           priv  N          3.00000  (         3.00000000)      B:A.prg
AY          pub   N         22.00000  (        22.00000000)
X           priv  N         10.00000  (        10.00000000)      B:A1.prg
AX          priv  N         11.00000  (        11.00000000)      B:A1.prg
    7 variables defined,      63 bytes used
  249 variables available.   5937 bytes available
PROGRAM A:
P           pub   N          1.73205  (         1.73205081)
X           priv  N          1.00000  (         1.00000000)      B:A.prg
Y           priv  N         20.00000  (        20.00000000)      B:A.prg
Z           priv  N          3.00000  (         3.00000000)      B:A.prg
AY          pub   N         22.00000  (        22.00000000)
    5 variables defined,      45 bytes used
  251 variables available.   5955 bytes available
```

从显示结果中可以看出：

① 全局变量一共有两个，一个是在提示符状态下定义的存贮变量P，另一个是在程序A1中用PUBLIC命令说明的AY。

② X,Y，Z均为局部于程序A的局部变量，而A1中的局部变量是X和AX。A1可以使用高层的应用程序A中的局部变量，反之则不行。

③ 由于A1和A中有同名变量X,所以A1中用PRIVATE命令把X宣布为隐蔽局部变量，这样A1中对X的操作处理并不影响A中X变量的值。

④ 因为工作方式开关FIXED被设置为ON，参数DECIMALS被置为5，所以程序中所有数值结果均以五位小数的形式输出，整数也不例外。而括号中的数据，是对应存贮变量的值的内部表示，它们均取九位小数，这是不随参数DECIMALS而变化的。

3. dBASEⅢ的函数

表5-5　dBASEⅢ函数分类表

类别	函数名	自变量类型	函数值类型	功能	等价的dBASEⅢ函数名
数学函数	EXP	N	N	指数函数	
	INT	N	N	取整	INT
	LOG	N	N	自然对数	
	ROUND	N	N	舍入到指定的小数位数。例如，ROUND(14.746,2)的值为14.750	
	SQRT	N	N	平方根	
字符串操作函数	&	C	C	宏替换函数	&
	AT	C	C	查找子字符串	@
	LOWER	C	C	大写字母改小写	
	SPACE	N	C	产生空格。例如，SPACE (10)是一个包含10个空格的字符串	
	SUBSTR	C	C	选择子字符串	$
	TRIM	C	C	取消尾部空格字符	TRIM
	UPPER	C	C	小写字母改大写	!
类型转换函数	ASC	C	N	字符转换为ASCII码	RANK
	CHR	整数	C	ASCII码转换为字符	CHR
	STR	N	C	数值类型转换为字符串类型	STR
	VAL	C	N	字符串类型转换为数值类型	VAL
测试函数	BOF	•	L	测试记录反向定位时是否跑过数据库文件中的第1个记录	
	COL	•	N	取得光标在屏幕上的列号	
	DELETED	•	L	测试当前记录是否有删除标记	•
	EOF	•	L	测试记录正向定位时是否跑过数据库文件中的最末记录	EOF
	FILE	C	L	测试文件存在否	FILE
	LEN	C	N	取得字符串的长度	LEN
	PCOL	•	N	取得打印机当前的列号	
	PROW	•	N	取得打印机当前的行号	
	RECNO	•	N	取得指针指向的数据库文件中的记录号	•
	ROW	•	N	取得光标在屏幕上的行号	
	TYPE	C	C	测试表达式是否有效以及它的类型	

日期函数	CDOW	D	C	从日期计算出星期几(用英文表示)。例如，设系统日期为05/15/84，则CDOW(DATE())的值是Tuesday	DATE
	CMONTH	D	C	从日期计算出月份(用英文表示)。例如,CMONTH(DATE())的值为May	
	CTOD	C	D	字符串类型转换成日期类型	
	DATE	•	D	给出系统所设定的日期。例如，DATE( )之值为05/15/84	
	DAY	D	N	给出日期中的日，例如，DAY(DATE()+10)之值为25	
	DOW	D	N	从日期计算出星期几(用数值表示，1表示星期天，2表示星期一等等)。例如，DOW(DATE())之值为3	
	DTOC	D	C	日期类型转换为字符串类型	
	MONTH	D	N	从日期计算出月份(用数值表示)。例如，MONTH(DATE())之值为5	
	TIME	•	C	以hh：mm：ss给出系统设定的时间	
	YEAR	D	N	从日期计算出年份。例如，YEAR(DATE()+365)之值为1985	

• 表示该函数无自变量

函数是表达式的重要组成部分，它大大地加强了dBASE Ⅲ命令的功能。dBASE Ⅲ的函数无论在功能上还是数量上都比dBASE Ⅱ有了较大的变更，表5-5是按功能分类的所有函数的一览表。

从表中可以看出，dBASE Ⅲ有10个函数的名称和功能与dBASE Ⅱ完全一样；有6个函数的功能与dBASE Ⅱ一样，但函数名有了修改，修改的目的是使得应用程序更容易阅读和理解；还有一个TYPE函数的功能有了加强。与dBASE Ⅱ相比，dBASE Ⅲ增加了20个新函数，因而使使用更加方便。

所有与dBASE Ⅱ功能相同的函数，其解释及举例请参考本书上册第六章第一节。新增加的函数大部分很易理解和使用，稍复杂一些的也已在表中给出例子作为示范。需要注意的是日期函数中自变量类型为日期时，它表示自变量允许是日期类型的表达式，其中整数（表示天数）可以参加加、减运算。

TYPE函数的功能是判断表达式是否正确，被测试的表达式必须用单引号、双引号或方括号给出。若不正确，其值为U(表示表达式无定义)；若正确，则给出表达式取值的类型。数值型、逻辑型、字符串型、日期型和备忘型分别取值为N、L、C、D和M。下面是TYPE函数的几个例子：

```
. ? TYPE( 'TEST' )          (值为U，因为该存贮变量还不存在)
. TEST='testing'
. ? TYPE( 'TEST' )          (值为C)
. ? TYPE( 'TEST/100' )      (值为U，因为字符串不能被数除，表达式不正确)
```

```
. ? TYPE( 'DATE( ) -100' )      (值为D，因为表达式指出现行日期之前100天
                                 的日期，是个正确的日期表达式)
. ? TYPE( ".Y." )               (值为L)
```

最后再扼要介绍一下宏替换函数&。&的功能是把字符串变量的内容直接替换在宏替换函数所出现的位置上，它的格式是：

&〈字符串型变量〉

宏替换函数在程序中运用恰当，常常能提高程序的灵活性。例如，先执行命令：

```
ACCEPT "which command do you need to execute? " TO C
```

用户如果打入某条命令（如DIR），则再执行命令

```
&C
```

时，实际上执行的就是用户所打入的DIR命令。

同样，若执行下述两条命令：

```
ACCEPT "Which condition do you select? " TO D
LIST FOR &D
```

则屏幕上将显示出数据库文件中满足某条件（这个条件是在执行命令ACCEPT时由用户临时选择的）的所有记录。

4. 应用程序

dBASE Ⅲ的命令可以以交互方式执行，即每次从键盘上输入一条命令后，dBASE Ⅲ立即执行这条命令并给出其结果。这种交互方式虽然很方便灵活，但毕竟效率不高，特别是在要反复地执行某些相同的命令时非常困难。dBASE Ⅲ执行命令的另一种方式是批量方式，即预先把命令组织成为一个程序——叫做dBASE Ⅲ的应用程序，然后再启动这个程序运行，于是dBASE Ⅲ就会自动地连续执行程序中的有关命令。

编制应用程序可以用MODIFY COMMAND命令来进行，它是一个屏幕正文编辑程序。也可以使用MS-DOS下的任何一个正文编辑程序来生成应用程序，只要文件的扩展名为·PRG即可。

应用程序源文件建立之后，可以使用DO命令启动它执行，打入的命令格式为：

```
DO〈应用程序名〉〔WITH〈参数表〉〕
```

也可以在MS-DOS下打入命令：

```
DBASE  〈应用程序名〉
```

使得dBASE Ⅲ启动成功之后立即执行指定的应用程序，如果在CONFIG.DB文件中有如下命令：

```
COMMAND = DO〈应用程序名〉〔WITH〈参数表〉〕
```

则dBASE Ⅲ启动成功之后也会自动地执行这个应用程序。

从编制程序的角度看来，dBASE Ⅲ就好象是一个普通的高级语言（解释执行方式），虽然它的主要处理对象是数据库文件，因而提供了许多文件处理方面的语句，但用于构造程序的控制语句却与普通高级语言基本相同。图5-3是从编程角度对dBASE Ⅲ语句的分类。

由于在编程语句方面dBASE Ⅲ比dBASE Ⅱ有了不少改进，例如可以区分全局变量与局部变量，子程序调用允许带有参数等，因此编写应用程序时更为方便有效。

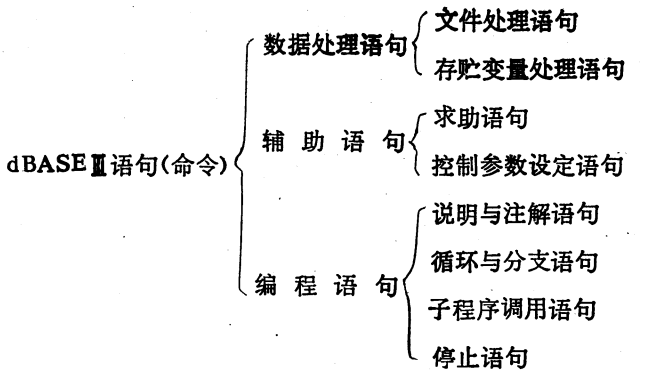


图5-3 dBASEⅢ语句分类

下面简单地介绍dBASEⅢ的编程语句。

（1）说明与注解语句

为了改善程序的可读性，程序中可以使用注解。注解命令为：

NOTE/*〈注解正文〉

如果注解语句以“；”结尾，则表示下一行仍为注解。

用来说明某些存贮变量为全局变量的PUBLLC命令和说明某些变量为隐蔽局部变量的PRIVATE命令，已在前面介绍过，这里不再重复。

作为子程序使用的应用程序可以带有形式参数，说明形式参数的命令格式如下：

PARAMETERS 〈形式参数表〉

其中形式参数表均为局部存贮变量。该命令必须在子程序中作为第1条有效命令出现。

dBASEⅡ中一个子程序，就是一个独立的存放在磁盘上的应用程序文件。调用子程序时，必须从磁盘上读入内存，作为一个已打开的文件。这种做法一方面效率低，另一方面由于可以同时打开的文件数量有限，因而又限制了多个子程序的嵌套调用。为了克服这个缺点，dBASEⅢ允许把多达32个子程序组合成为一个“过程文件”（文件扩展名也是.PRG），过程文件中每个子程序均使用PROCEDURE命令进行说明，其格式为：

PROCEDURE 〈子程序名〉

例如，下面就是一个由三个子程序组成的过程文件：

```
** This is an example of procedure file
PROCEDURE Message1
     @ 15,0 CLEAR
     @18,10 SAY "This is message1"
RETURN
PROCEDURE Message2
     @ 10,0 CLEAR
     @ 15,10 SAY "Please enter a command"
RETURN
PROCEDURE Message3
     ACCEPT "What is the condition?" TO D
     LIST FOR &D
RETURN
```

过程文件与应用程序一样，可以用MODIFY COMMAND命令生成，并存放在磁盘上。

（2）循环与分支语句

这一组语句与dBASE Ⅱ完全一样，现简单介绍如下：

① 条件语句

```
IF〈条件〉
    〈命令组〉
〔ELSE〕
    〈命令组〉
ENDIF
```

② 情况语句

```
DO CASE
    CASE〈条件〉
        〈命令组〉
    〔CASE〈条件〉〕
        〈命令组〉
    〔OTHERWISE〕
        〈命令组〉
ENDCASE
```

③ 循环语句

```
DO WHILE〈条件〉
    〈命令组1〉
〔LOOP〕
    〈命令组2〉
ENDDO
```

其中LOOP命令用来直接跳向循环开始处而不再执行〈命令组2〉，它经常与IF命令或CASE命令配合使用。下面是使用LOOP命令的一个例子：

```
SET TALK OFF
STORE 0 TO x, s
DO WHILE x<=5
    IF s<10
        s=s+x
        x=x+1
        ? x, s
        LOOP
    ENDIF
    x=x+1
ENDDO
SET TALK ON
```

RETURN

（3）子程序调用语句

子程序（即过程）是一个以RETURN命令结尾的应用程序。调用子程序的语句格式为：

DO〈应用程序名〉〔WITH〈参数表〉〕

其中，〈参数表〉用来给出有关参数，它必须与子程序中用PARAMETERS定义的参数相匹配。调用子程序的实例如下，它的计算结果为24。

（主程序）	（名为Areacalc的子程序）
Area = 0	PARAMETERS Length, Width, A
Do Areacalc WITH 4 , 6 , Area	A = Length * Width
? Area	RETURN

DO命令所调用的子程序也可以是过程文件中的子程序，在这种情况下过程文件必须预先用SET PROCEDURE命令打开。SET PROCEDURE命令的格式是：

SET PROCEDURE TO 〈过程文件名〉

dBASEⅢ规定一段时间只能打开一个过程文件，过程文件用后须用下面的命令来关闭：

CLOSE PROCEDURE

下面是使用DO命令调用过程文件中子程序的一个例子。

例5-3　调用过程文件

设应用程序为Setproc.prg，过程文件名为Proctest.prg，它包含有三个子程序，它们的程序清单如下：

```
C > TYPE SETPROC.PRG
* * Setproc.prg - demonstrates SET PROCEDURE TO
SET PROCEDURE TO Proctest
* Opens the procedure file
DO Proc 1
* Executes procedure one
DO Proc 2
* Executes procedure two
CLOSE PROCEDURE
* Close the procedure file
* EOF: Setproc.prg
C > TYPE PROCTEST.PRG
* Proctest.prg - procedure
PROCEDURE Proc 1
    ? "This is a message from Proc1."
    RETURN
PROCEDURE Proc2
    ? "Greetings from Proc2."
```

```
    DO Proc3
    RETURN
PROCEDURE Proc3
    ? "Setproc never calls me! "
    RETURN
* EOF : Proctest.PRG
```

则执行应用程序Setproc后得到的结果为：

```
. DO SETPROC
This is a message from Proc1.
Greetings from Proc2.
Setproc never calls me!
```

（4）停止语句

dBASE Ⅲ应用程序中可以使用的停止语句有下列四条：

① QUIT语句　　关闭所有文件，结束dBASE Ⅲ的运行，返回MS-DOS操作系统。

② CANCEL语句　关闭所有文件，结束应用程序运行，返回dBASE Ⅲ提示符状态。

③ RETURN语句　停止应用程序的执行并返回到调用程序，如无调用程序则返回到dBASE Ⅲ提示符状态。

RETURN TO MASTER　　停止应用程序的执行并返回到最高层的调用程序。

④ EXIT语句　　在DO WHILE…ENDDO循环的执行中，用来跳出循环，然后立即执行ENDDO后面的命令。

```
  TYPE EXITDEMO.PRG
** Exitdemo.prg - demostrates EXIT and RETURN
SET TALK OFF
A=1
DO WHILE .T.
    IF A>=250
        EXIT
    ENDIF
    A=A+1
ENDDO
SET TALK ON
? "A is now"+STR(A,4)+" so I have exited"
RETURN

  DO EXITDEMO
A is now 250 so I have exited
```

图5-4　EXIT的使用

例5-4　EXIT命令和RETURN命令的使用

应用程序的清单如图5-4所示，当变量A的值达到250时，循环即告结束。

第三节　数据库文件的建立和修改

1．文件结构的定义和修改

数据库文件必须首先建立，然后才能使用。建立一个新的数据库文件，包括定义该文件的结构和向文件中输入数据两方面的含义。下面先介绍数据库文件结构的定义和修改。

（1）文件结构的定义

定义一个数据库文件的结构，需要确定该文件的名字，以及它所包含的字段个数，每个字段的名称、类型和宽度等，这是使用创建命令（CREATE）来进行的。CREATE命令的格式为：

CREATE〈文件名〉

其中，〈文件名〉即为要建立的一个新数据库文件的名字。系统执行这条命令之后，就要求用户输入文件中每个字段的名称、类型、宽度和小数位数（如果是数值类型的话）。类型只需要输入一个英文字母（C、N、L、D或M）即可确定，也可以用空格键显示用户所需要的类型，然后输入〈CR〉键选定。对于日期类和备忘类两种字段，它们的宽度分别是8和10，无需用户输入。如果文件结构中有备忘字段的话，dBASEⅢ将自动地建立一个同名的备忘文件（.DBT），用来存放备忘字段的内容。注意，若字段内容少于255个字符时，不宜把该字段定义为备忘字段，而应定义为字符型字段，这样在使用时比较方便。有些字段的内容看起来都是数字，但实际上并不参加计算（如电话号码、邮政信箱等），这就应该定义成字符类型而不是数值类型。

CREATE命令是一条全屏幕工作方式的命令。在定义字段时，允许光标后退对已定义过的字段修改（↑键），也允许插入一个新的字段（CTRL-N键）或删去一个已有字段（CTRL－U）。操作过程中，屏幕顶行显示出数据库文件名、已定义的字段个数和尚余字节数（已定义字段的宽度之和与最大记录长度的差额）。当所有字段都定义、修改完毕后，使用CTRL-END键或CTRL-W就可结束CREATE命令的工作。

这里顺便介绍一下全屏幕工作方式，因为dBASEⅢ的许多命令都以这种方式进行工作。所谓全屏幕方式，是指用户输入数据是以填充方式或修改方式进行的。屏幕上的提示信息以正视频方式显示，数据区均以反视频方式显示，用户可以用光标控制键在数据区内任意移动光标，选择需要填入或修改的数据项目，因而操作极为方便。表5-6是全屏幕方式下公共控制键一览表，特殊键的使用将在各有关命令中介绍。

表5-6　全屏幕方式的公共控制键

键	功　　能	键	功　　能
↑	光标上移一行或一个字段	退　格	删去光标左边一个字符
↓	光标下移一行或一个字段	Del	删去光标所在处的一个字符
←	光标左移一位或一个菜单项目	CTRL-N	插入一行或一个字段定义
→	光标右移一位或一个菜单项目	Ins	在字符插入方式与选字方式之间
PgUp	屏幕上滚一页或一个记录		切换
PgDn	屏幕下滚一页或一个记录	Esc	不保留任何所作的修改而退出
Home	光标左移一字		
End	光标右移一字		

尽管dBASE Ⅲ允许每个数据库文件最多可以有128个字段，但实际经验表明，只有使字段数目保持在20个以下，才能比较有效地使用数据库。在一般情况下，最好不要设计字段数目太多的数据库文件。

图5-5是一个建立学生数据库文件的屏幕画面。该文件的名为STUDENTS.DBF，共有五个字段组成：姓名（NAME，C型，宽度为10），年级（GRADE，C型，宽度为2）入学成绩（POINTS，N型，宽度为6，小数1位），出生日期（BIRTHDAY，D型，宽度为8），数学是否超过100分（MATH100，L型，宽度为1）。从画面右上角可知，该文件的结构中每个记录的长度为28（4000—3973＋1）个字节，一共有五个字段。

```
C:STUDENTS dbf        Bytes remaining:  3973
                      Fields defined:      5

     field name    type       width    dec
    ============================================
1    NAME          Char/text    10
2    GRADE         Char/text    2
3    POINTS        Numeric      6       1
4    BIRTHDAY      Date         8
5    MATH100       Logical      1
6                  Char/text
```

图5-5　用CREATE命令建立STUDENTS文件

（2）文件结构的显示

对于数据库中已经存在的数据库文件，用户可以使用DISPLAY STRUCTURE或LLST STRUCTURE命令来了解它们的文件结构。但是在此之前，必须先把数据库文件打开（打开的数据库文件称为工作文件），为此必须使用下面的USE命令：

USE〈数据库文件名〉

如果数据库文件中有备忘类型字段，则对应的.DBT文件也将自动打开。

显示文件结构的两条命令格式如下：

DISPLAY STRUCTURE 〔TO PRINT〕

LIST STRUCTURE 〔TO PRINT〕

它们的功能完全一样，而前者在输出一屏信息后会自动暂停，后者则否。命令中若含有TO PRINT，则输出信息会在打印机上打印出来。

图5-6是使用DISPLAY STRUCTURE命令显示STUDENTS.DBF文件得到的结果。

```
USE STUDENTS

DISPLAY STRUCTURE

Structure for database : B:STUDENTS.dbf
Number of data records :        0
Date of last update    : 06/10/85
Field  Field name  Type        Width     Dec
    1  NAME        Character      10
    2  GRADE       Character       2
    3  POINTS      Numeric         6       1
    4  BIRTHDAY    Date            8
    5  MATH100     Logical         1
** Total **                      28
```

图5-6 文件结构的显示画面

从图中可以看出，除了文件中各个字段的名称、类型、宽度等信息外，它还给出了文件中已经包含的数据记录的总数及该文件最近一次修改的日期。

（3）文件结构的修改

对于一个已经建立的数据库文件，如果要修改它的文件结构，可以使用命令：

MODIFY STRUCTURE〈文件名〉

该命令与dBASE Ⅰ相比已有了很大的改进，它把原来的.DBF文件自动修改成同名的后备文件（.BAK），原来文件中的数据自动复制到已修改了结构的新文件中（dBASE Ⅰ无此功能）。需要注意，当某字段的字段名和宽度同时有变化时，该字段的内容不被复制到新文件中。为此可以两次使用MODIFY STRUCTURE命令，第1次先修改字段名，第2次再改变它的宽度。另外，该命令不能用来删除字段，需要删除某个字段时必须使用COPY命令。

MODIFY STRUCTURE也是一条全屏幕工作方式命令，下面是利用它在STUDENTS文件中增加一个说明字段（NOTE，M型）的显示画面：

MODI STRU

B: STUDENTS.dbf　　Bytes remaining: 3963

Fields defined: 6

	field name	type	width	dec
1	NAME	Char/text	10	
2	GRADE	Char/text	2	
3	POINTS	Numeric	6	1
4	BIRTHDAY	Date	8	
5	MATH100	Logical	1	
6	NOTES	Memo	10	
7		Char/text		

图5-7　MODIFY STRUCTURE修改文件结构的画面

（4）结构描述文件的建立

通过前面的介绍可以发现，数据库文件的结构描述本身也是一张表格。不同的文件结构，对应的表格格式完全相同，仅仅是表的行数和表中内容有所不同而已。由于一张表对应着一个数据库文件，因此任何一个数据库文件的结构描述也可以转换成一个数据库文件，故可把它称为“结构描述文件”。为某个数据库文件生成它的结构描述文件需要先使用命令USE打开文件，再使用语句：

COPY TO〈文件名〉STRUCTURE EXTENDED

下面是为STUDENTS文件生成结构描述文件的一个例子。

例5-5　结构描述文件的生成

使用COPY STRUCTURE EXTENDED命令生成STUDENTS文件的结构描述文件STRU1的过程如下，STRU 1的结构和内容也一起用LIST命令列出。

```
. USE STUDENTS
. COPY TO STRU1 STRUCTURE EXTENDED
. USE STRU1
. LIST STRU
Structure for database : B:STRU1.dbf
```

```
Number of data records :        6
Date of last update    : 06/10/85
Field  Field name  Type        Width     Dec
    1  FIELD_NAME  Character      10
    2  FIELD_TYPE  Character       1
    3  FIELD_LEN   Numeric         3
    4  FIELD_DEC   Numeric         3
** Total **                       18

. LIST

Record#  FIELD_NAME  FIELD_TYPE  FIELD_LEN  FIELD_DEC
      1  NAME        C                  10          0
      2  GRADE       C                   2          0
      3  POINTS      N                   6          1
      4  BIRTHDAY    D                   8          0
      5  MATH100     L                   1          0
      6  NOTES       M                  10          0
```

2. 数据库文件的数据输入

利用CREATE命令定义了一个新的数据库文件的结构之后，可以紧接着就向该数据库文件中输入数据，也可以在以后需要时再向文件中添加数据。

（1）从键盘（向数据库文件）添加数据

从键盘向数据库文件中添加新的记录时，必须使用命令：

```
APPEND
```

或

```
APPEND  BLANK
```

后者是向文件中添加一个空白的记录，即只有记录序号而无内容的记录。

APPEND命令以全屏幕方式让用户从键盘上输入数据，一屏一个记录，它总是添加在原来文件的尾部。当输完一个记录的全部数据后，屏幕上又提供下一个空白记录供用户输入数据，此时用〈CR〉键或CTRL-END键就可结束APPEND命令的操作。下面是向STUDENTS文件中添加第1个记录时的屏幕画面：

```
   APPEND
Record  No          1
NAME
GRADE
POINTS
BIRTHDAY        /  /
MATH100         ?
NOTES           memo
```

可以看出，数值型字段的小数点位置，日期型字段，逻辑型字段及备忘型字段均已预先在画面上标出，从而使数据输入时减少错误。

为了向备忘型字段中输入数据，必须把光标移到字段开始处，然后敲入CTRL-PgUp键（或CTRL-Home），此时屏幕上出现空白画面，用户就可以输入备忘信息（操作方法与MODIFY COMMAND命令一样）。输入完毕，可以用CTRL-PgUp，CTRL-W或CTRL-End键（保存输入的备忘信息），或者ESC键或CTRL-Q键（不保存）返回到原来的记录添加画面。

在APPEND命令下，也可以修改文件中已有的数据。此时只要打入CTRL-C或CTRL-R键，便可以向前或向后移动记录画面，进行编辑修改。

（2）从其它数据库文件向工作文件添加数据

除了从键盘上向数据库文件输入数据之外，也可以从其它数据库文件中把有关数据输入到工作数据库文件中，使用的命令格式为：

APPEND FROM〈源文件名〉〔FOR／WHILE〈条件〉〕

其中，条件可用来指出源文件中某些满足条件的记录，如果不指出，则为全部记录。数据添加操作进行时只有字段名相同、类型也相同的那些字段内容，才能作为新的记录添加到工作文件尾部去。若源文件中字段长于工作文件中同名字段，则字符型数据将截去尾部，数值型数据用＊填入。图5-8是APPEND FROM命令的示意图。

文件ADDRESS.DBF

NAME	GRADE	ADDRESS
Qian	1	
Sun	3	
Li	1	
Zhou	4	

(APPEND FROM ADDRESS)

工作文件STUDENTS.DBF（执行APPEND命令前）

NAME	GRADE	POINTS	BIRTHDAY	MATH100	NOTES
Zhao	2	581	05/16/65	F	

工作文件STUDENTS.DBF（执行APPEND命令后）

NAME	GRADE	POINTS	BIRTHDAY	MATH100	NOTES
Zhao	2	581	05/16/65	F	
Qian	1				
Sun	3				
Li	1				
Zhou	4				

图5-8 APPEND FROM命令示意图

不难理解，使用APPEND FROM命令可以大大地减少数据录入的工作量。

（3）从正文文件向数据库文件输入数据

向数据库文件输入数据的另一种方法是从正文文件输入数据。这种方法常用于dBASE Ⅱ与其它高级语言程序交换数据，即dBASE Ⅱ可以从高级语言程序建立的正文文件中取得数据。

用来向数据库文件添加数据的正文文件必须符合一定的格式。一种格式叫SDF格式，其中的每一个数据项目必须严格地按照字段的宽度对齐，每个记录以回车换行符结束。另一种

格式叫DELIMITED格式，其中的每一项数据均用逗号“，”作为分隔符隔开，字符串则用单引号或双引号括出，每个记录也以回车换行符结束。图5-9是与数据库文件STUDENTS的结构相对应的两种正文文件的内容，它们各含六个记录。

```
.TYPE SDF.TXT
Zhao      2   581.519650516T
Qian      1   560.019650724F
Sun       3   495.019641201F
Li        1   525.019660324F
Zhou      4   620.519650831T
Wu        2   591.019660101T
.TYPE DELIF1.TXT
“Zhao”，“2”，581.5，19650516，T
“Qian”，“1”，560.0，19650724，F
“Sun”，“3”，495.0，19641201，F
“Li”，“1”，525.0，19660324，F
“Zhou”，“4”，620.5，19650831，T
“Wu”，“2”，591.0，19660101，T
```

图5-9 用于向数据库文件添加数据的正文文件

从正文文件向数据库文件添加数据需要使用下述命令：

APPEND FROM〈正文文件名〉 SDF/DELIMITED

当使用SDF格式的正文文件时，数据添加是按记录进行的；当使用DELIMITED格式的正文文件时，数据添加是以字段为单位进行的。

（4）数据库文件的复制

把数据库文件复制成为结构和内容均相同的另一个数据库文件，可以使用第一节中介绍过的COPY FILE命令进行，或者用MS-DOS下的COPY命令进行。但如果要有选择地把数据库文件中的一部分内容（一部分字段、一部分记录）复制成另一个数据库文件，或者是把数据库文件转换成为正文文件，则必须使用COPY命令。COPY命令的格式是：

COPY TO〈文件名〉〔〈范围〉〕〔FIELDS〈字段名表〉〕
〔FOR/WHILE〈条件〉〕〔SDF/DELIMITED〔WITH〈分隔符〉〕〕

该命令用来把指定范围内或满足条件的所有记录均复制到新的数据库文件中去，若不指出范围或条件，则为文件中的全部记录。复制时，仅复制那些由字段名表所指出的字段，若不指出，则为全部字段。如果命令中出现SDF或DELIMITED，则表示数据库文件被转换成正文文件。若为DELIMITED格式的正文文件，还可以用指定的分隔符。缺省时，分隔符为双引号。需要注意，备忘字段中的内容不被转换进正文文件。

下面是使用COPY命令复制数据库文件和正文文件的一个例子。

例5-6 数据库文件的复制

该数据库文件STUDENTS中有六个记录，它的内容及使用COPY命令后所得到的新文件

的内容如图5-10所示。其中命令COPY TO…STRUCTURE表示只复制文件的结构，不复制其中的内容。

```
COPY TO NEWFILE1 FIELDS NAME,GRADE,POINTS
COPY TO NEWFILE2 RECODE 3
COPY TO NEWFILE3 FIELDS NAME,GRADE,POINTS;
FOR POINTS>550
COPY TO NEWFILE4 WHILE YEAR(DATE())-YEAR(;
BIRTHDAY)=20
COPY TO STRU2 FIELDS NAME,GRADE,BIRTHDAY STRUCTURE
COPY TO TXTFILE1 FIELDS NAME,POINTS,BIRTHDAY,MATH100 SDF
COPY TO TXTFILE2 DELIMITED WITH /
```

```
. LIST

Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      1  Zhao           2  581.5 05/16/65 .T.     Memo
      2  Qian           1  560.0 07/24/65 .F.     Memo
      3  Sun            3  495.0 12/01/64 .F.     Memo
      4  Li             1  525.0 03/24/66 .F.     Memo
      5  Zhou           4  620.5 08/31/65 .T.     Memo
      6  Wu             2  591.0 01/01/66 .T.     Memo

. USE NEWFILE1

. LIST

Record#  NAME       GRADE POINTS
      1  Zhao           2  581.5
      2  Qian           1  560.0
      3  Sun            3  495.0
      4  Li             1  525.0
      5  Zhou           4  620.5
      6  Wu             2  591.0
  USE NEWFILE2

  LIST

Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      1  Sun            3  495.0 12/01/64 .F.     Memo
  USE NEWFILE3

. LIST

Record#  NAME       GRADE POINTS
      1  Zhao           2  581.5
      2  Qian           1  560.0
      3  Zhou           4  620.5
      4  Wu             2  591.0
```

```
. USE NEWFILE4

. LIST

Record#  NAME        GRADE  POINTS  BIRTHDAY  MATH100  NOTES
      1  Zhao        2       581.5  05/16/65  .T.      Memo
      2  Qian        1       560.0  07/24/65  .F.      Memo
. USE STRU2

. LIST STRU

Structure for database : B:STRU2.dbf
Number of data records :        0
Date of last update    : 06/01/85
Field  Field name  Type        Width     Dec
    1  NAME        Character      10
    2  GRADE       Character       2
    3  BIRTHDAY    Date            8
** Total **                       21

. USE STRU2

. LIST

. TYPE TXTFILE1.TXT

Zhao        581.519650516T
Qian        560.019650724F
Sun         495.019641201F
LI          525.019660324F
Zhou        620.519650831T
Wu          591.019660101T

. TYPE  TXTFILE2.TXT

/Zhao/,/2/,581.5,19650516,T
/Qian/,/1/,560.0,19650724,F
/Sun/,/3/,495.0,19641201,F
/LI/,/1/,525.0,19660324,F
/Zhou/,/4/,620.5,19650831,T
/Wu/,/2/,591.0,19660101,T
```

图5-10 数据库文件的复制及结果

其中语句

COPY TO STRU2 FIELDS NAME, GRADE, BIRTHDAY STRUCTURE

的作用是，根据数据库文件STUDENTS的结构创建一个新的数据库文件STRU2，它相当于一条CREATE STRU2语句，但却节省了根据提示输入STRU2文件的各个字段描述的操作过程，方便了用户。新建立的STRU2是一个空文件。

3. 数据库文件的修改

在实际应用中，数据库文件的内容经常需要进行修改。例如，插入一些新记录，替换一部分记录或删除一部分记录等。

（1）记录的定位与插入

要在数据库文件中间插入一个新的记录，首先必须进行定位操作。当使用USE语句打开一个数据库文件后，dBASE Ⅲ内部有一个指针就指着文件的某个记录，该记录称为当前记录。文件刚开始打开时，当前记录为1号记录。

定位操作的目的就是移动指针到所需要的某个记录位置处，它可以用三种命令来完成：

① GOTO命令　它有如下几种形式：

GOTO（或GO）〈数值表达式〉	定位到表达式所指出的某个记录
GOTO（或GO） TOP	定位到第1个记录
GOTO（或GO） BOTTOM	定位到最末一个记录

② SKIP命令　它有两种形式：

SKIP	指针向后（增大方向）移一个记录
SKIP ±〈数值表达式〉	指针向后（+号）或向前（−号）移动表达式所指出的记录个数

需要指出的是，当指针向后移动而超过文件中最后一个记录时，函数RECNO（）的值为最末记录序号加1，EOF（）的值为真；当指针向前移动而超过文件的第1个记录时，RECNO（）之值为1，BOF（）之值为真。

③ LOCATE命令

LOCATE命令用来把指针定位到满足某种条件的记录所在处，它的格式为：

LOCATE〔〈范围〉〕FOR〈条件〉

当不指定范围时，LOCATE命令将从文件的第1个记录开始顺序查起，把指针定位在首先遇到的满足规定条件的某个记录上，若所有记录都不符合条件，则指针定位在最末一个记录的后面。如果指定范围为NEXT n，则查找从当前记录开始，且限定只查n个记录，查不到符合条件的记录，则定位在该范围的最末一个记录之后。

图5-11是打开STUDENTS文件（文件内容见例5-6）后用LOCATE命令进行定位操作的显示画面。其中第1次定位操作是成功的，然后用CONTINUE命令按同样条件在同样范围内再查找，结果为记录序号5、6；当第3次执行CONTINUE命令时，由于查不到满足规定条件（入学分数＞580）的记录，这时EOF（）为真。

把指针定位到所需要的位置处以后，就可以用插入命令INSERT插入新的记录。INSERT命令有下列四种形式：

INSERT BLANK	在当前记录的后面插入一个空白记录
INSERT BEFORE BLANK	在当前记录的前面插入一个空白记录
INSERT	在当前记录后面插入一个新记录
INSERT BEFORE	在当前记录前面插入一个新记录

如果设当前记录序号为5，则在它前面插入的新记录序号就为5，而原来的5号记录变为6号，并依此类推。若当前记录序号为5，而在它后面插入新记录，则新记录序号为6，

原6号记录改成7号，并依此类推。

需要插入新记录时，用户将以全屏幕方式工作。数据输入的操作与APPEND命令完全相同，这里不再赘述。

```
. USE STUDENTS

. LOCATE FOR POINTS>580

Record =        1


. CONT

Record =        5
. CON-T

Record =        6
. CONT

End of locate scope
. ? RECNO()

           7
. ? EOF()

.T.
```

图5-11　用LOCATE命令进行定位操作

（2）记录的删除与恢复

删除数据文件中的某些记录可以用DELETE命令来完成，它的格式是：

DELETE〔〈范围〉〕〔FOR/WHILE〈条件〉〕

这表示将指定范围中满足规定条件的所有记录都打上删除标记（*）。当然这些记录在这时并未真正被删去，而还保存在数据库文件之中，只是此后它们不再参加包括数据计算在内的一些操作。

如果需要把那些有删除标记的记录恢复成有效记录，则需要使用RECALL命令，它的格式为：

RECALL〔〈范围〉〕〔FOR/WHILE〈条件〉〕

如果不指出范围或条件，则它仅仅恢复当前记录（如果是有删除标记的记录）。

如果确实被删除的记录以后再也不需要恢复使用，则可以使用PACK（整理）命令把它们从文件中永远删去，以节省存贮空间。

如果想把整个数据库文件的全部记录都永久性地删去，则可使用ZAP命令。

dBASE Ⅲ的控制参数中有一个工作方式开关DELETED，它的缺省状态为OFF，表示凡

有删除标记的那些记录在执行其它命令时，应被考虑在内。而当开关状态为ON时，则不再受到处理。

（3）记录的替换

数据库文件中的记录，也可以用替换命令（REPLACE）来更换其中一部分或全部字段的内容。REPLACE命令的格式如下：

REPLACE〔〈范围〉〕〈字段〉WITH〈表达式〉〔，〈字段〉WITH〈表达式〉…〕〔FOR／WHILE〈条件〉〕

该命令把指定范围中满足规定条件的那些记录中的有关字段的内容更换成为表达式之值，若不指出范围和条件，则只对当前记录进行替换操作。

例5-7　记录的置换操作

设数据库文件STUDENTS的内容与例5-6相同，在执行了几次REPLACE命令后的结果如图5-12所示。

```
. USE STUDENTS

. APPEND BLANK

. REPLACE NAME WITH "Chen",GRADE WITH "2",POINTS WITH
647,BIRTHDAY WITH CTOD("10/01/64"),MATH100 WITH .F.

. REPLACE ALL BIRTHDAY WITH BIRTHDAY-30

     7 records replaced
. REPLACE RECORD 3 POINTS WITH 500

     1  record replaced
. REPLACE NEXT 3 GRADE WITH "3" FOR POINTS<600

     2 records replaced
. LIST

Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      1  Zhao       2       581.5 04/16/65 .T.     Memo
      2  Qian       1       560.0 06/24/65 .F.     Memo
      3  Sun        3       500.0 11/01/64 .F.     Memo
      4  LI         3       525.0 02/22/66 .F.     Memo
      5  Zhou       4       620.5 08/01/65 .T.     Memo
      6  Wu         2       591.0 12/02/65 .T.     Memo
      7  Chen       2       647.0 09/01/64 .F.     Memo
```

图5-12　记录置换操作

4．数据库文件的编辑

数据库文件的内容，除了可以用上面介绍的插入、删除和替换等命令进行自动修改外，也可以使用下面的EDIT、BROWSE和CHANGE命令手动地在键盘上进行编辑修改。

（1）EDIT命令

EDIT命令的格式为：

EDIT〔〔RECORD〕〈数值表达式〉〕

它用来对表达式所指定的一个记录进行编辑修改，如果命令中不指定记录，则它对当前记录进行编辑修改。

编辑修改是以全屏幕方式进行的，每屏一个记录。使用PgUp（CTRL-R）键和PgDn（CTRL-C）键，可分别进入上一个记录或下一个记录的编辑修改。如果要对记录中备忘类型字段的内容作修改时，把光标移到该字段开始处，打入CTRL-PgUp（CTRL-Home）键即可，修改完毕再用CTRL-PgUp（保留修改）ESC键（取消修改）退回到EDIT模式。一个或一组记录的内容编辑修改完毕后，使用CTRL-End（或CTRL-W）键（修改有效）或ESC键（取消当前记录的修改）结束EDIT命令。

（2）BROWSE命令

BROWSE命令的格式为：

BROWSE 〔FIELDS〈字段名表〉〕

它也是一条全屏幕方式的编辑命令。与EDIT命令不同之处在于：EDIT命令一次只修改一个记录，而BROWSE命令却可以对整个文件进行修改，它一次在屏幕上显示多达17个记录供用户任意挑选修改（当前记录用反视频显示），而且利用FIELDS列出的字段名表，可以只对指出的那些字段内容作修改，字段名表中字段的次序，就是它们在屏幕上出现的次序，因而十分方便。

当字段个数较多，屏幕上容纳不下所有字段的信息时，使用CTRL-←键和CTRL-→键可以使显示数据在屏幕窗口中左移或右移一个字段。PgUp和PgDn键则利用来显示文件中接下去的17个记录或退回到前面的17个记录。

使用CTRL-Y可以删去一个字段的内容，而使用CTRL-U键则把当前记录标以删除标记，再次打入CTRL-U键时，则去掉删除标记。

用↓键把当前记录定位到文件最末记录的后面时，BROWSE命令可以用来向数据库文件添加新记录，但记录中的备忘型字段内容无法用BROWSE命令修改或添加。

结束BROWSE命令的工作需要使用CTRL-End键（修改有效）或Esc键（修改无效）。

BROWSE命令不但是一个全屏幕工作方式的命令，而且它还可以用菜单选择方式来进行操作。图5-13是以STUDENTS作为工作文件打入命令

BROWSE FIELDS NAME, POINTS, BIRTHDAY, GRADE

后屏幕上的显示画面，当打入CTRL-Home键后即可进入菜单选择操作。被选中的菜单项目以反视频显示，其中各菜单项目的含义如下：

Bottom	表示把当前记录定位到文件的最末记录
Top	表示把当前记录定位到文件的第1个记录
LocK	定义固定在屏幕上的显示字段，它们不随〈CTRL－→〉〈CTRL－←〉左右移动
Record #	把当前记录定位到指定的记录

Freeze　　只对一个字段进行编辑

```
Bottom      Top      Lock      Record #      Freeze
NAME------ POINTS BIRTHDAY GRADE
Zhao        581.5 05/16/65 2
Qian        560.0 07/24/65 1

 Sun        495.0 12/01/64 3
 Li         525.0 03/24/66 1
 Zhou       620.5 08/31/65 4
 Wu         591.0 01/01/66 2
```

图 5-13　BROWSE命令的操作画面

（3）CHANGE命令

CHANGE命令的格式如下：

CHANGE〔〈范围〉〕〔FIELDS〈字段名表〉〕〔FOR/WHILE〈条件〉〕

它同时具有EDIT命令和BROWSE命令的一些优点，即可以对指定范围内满足规定条件的一组记录中的有关字段（用字段名表列出）的内容进行编辑修改，因而使用十分方便。它也是一条全屏幕工作方式的命令，一屏只显示一个记录。编辑修改的操作方法与EDIT命令相同，这里不再重复，通过这条命令还可以对备忘字段进行编辑修改。

第四节　数据库文件的使用

数据库管理系统最重要的功能主要反映在数据库文件的使用方面，即数据的排序、检索、统计及报表生成等各种操作。在这方面，dBASE Ⅲ比dBASE Ⅱ功能更好，操作速度也明显提高。

为了叙述的方便，本节假设已经建立了三个数据库文件，即学生文件(STUDENTS.DBF)，成绩文件（SCORE.DBF）和任课教师文件（TEACHERS.DBF）。它们的文件结构以及文件内容如图5-14所示。本节将使用这些文件进行举例，以便更好地阐明有关内容。

```
. USE STUDENTS
. LIST STRU
Structure for database : B:STUDENTS.dbf
Number of data records :        6
Date of last update    : 06/01/85
Field  Field name  Type        Width     Dec
    1  NAME        Character      10
    2  GRADE       Character       2
    3  POINTS      Numeric         6       1
    4  BIRTHDAY    Date            8
    5  MATH100     Logical         1
    6  NOTES       Memo           10
```

```
** Total **                        38

  LIST
Record#  NAME        GRADE POINTS BIRTHDAY MATH100 NOTES
      1  Zhao        2       581.5 05/16/65 .T.     Memo
      2  Qian        1       560.0 07/24/65 .F.     Memo
      3  Sun         3       495.0 12/01/64 .F.     Memo
      4  Li          1       525.0 03/24/66 .F.     Memo
      5  Zhou        4       620.5 08/31/65 .T.     Memo
      6  Wu          2       591.0 01/01/66 .T.     Memo
  LIST NOTES
Record#  NOTES
      1  Zhao was a typist before, just started on computer
         courses. If graduate with good result, he will be
         promoted to a senior position with a pay raise.

      2
      3
      4  Li worked mainly with the manager and worked with
         the confidential file that was kept in the
         manager's office.

      5
      6

. USE SCORE
. LIST STRU
Structure for database : B:SCORE.dbf
Number of data records :        6
Date of last update    : 06/01/85
Field  Field name  Type       Width    Dec
    1  NAME        Character     10
    2  CHEMISTRY   Numeric        3
    3  PHYSICS     Numeric        3
    4  ENGLISH     Numeric        3
** Total **                      20

    LIST
Record#  NAME        CHEMISTRY PHYSICS ENGLISH
      1  Wu                 95      77      84
      2  Li                 84      90      88
      3  Sun                92      80      92
      4  Zhao               85      70      95
      5  Zhou               76      94      99
      6  Qian               70      85      78
. USE TEACHERS
. LIST STRU
Structure for database : B:TEACHERS.dbf
Number of data records :        6
```

```
Date of last update     : 06/01/85
Field  Field name  Type        Width     Dec
    1  TNAME       Character      10
    2  AGE         Numeric         2
    3  GRADE       Character       2
    4  COURSE      Character      10
** Total **                      25

  LIST
Record#  TNAME       AGE GRADE COURSE
      1  Chen         45 2     Math
      2  Wang         31 1     Physics
      3  He           26 4     Chemistry
      4  Zhang        58 4     English
      5  Song         38 3     Computer
      6  Zheng        60 3     Physics
```

图5-14 三个数据库文件的结构及内容

1. 数据排序和索引文件

（1）排序命令（SORT）

排序操作就是把数据库文件中的记录按照某个字段值的大小进行重新排列。作为排序依据的这个字段被称为“关键字段”。排序可以按关键字段的值由小到大（即增序）进行，也可以从大到小地（即减序）进行。作为关键字段的字段类型必须是数值型、字符串型和日期型。逻辑型和备忘型字段不能作为排序的依据。排序操作的结果将得到一个新的数据库文件。

排序操作也叫分类操作，这是数据处理应用中经常使用的一种操作。例如，把学生按入学成绩排序，按班级排序或按照年龄大小排序等等。

dBASE Ⅲ的排序命令SORT，不仅在功能上比dBASE Ⅱ有了扩充，而且由于改进了算法，执行的速度也有明显提高。SORT命令的格式为：

SORT TO〈新文件名〉ON〈关键字段名〉〔/A〕〔/D〕
〔，〈关键字段名〉〔/A〕〔/D〕…〕
〔〈范围〉〕〔FOR〈条件〉〕

它表示把工作文件中指定范围内满足规定条件的那些记录，按照给出的一个或多个关键字段进行排序（/A表示增序，/D表示减序）并产生一个新的数据库文件。排序命令中不指出增序或减序时即为增序。有多个关键字段时，先出现的字段为优先排序的关键字段。

例5-8　数据库文件的排序

设有数据库文件STUDENTS，现要求产生：

① 按入学分数（从高到低）排序的新文件STUDENT 1；

② 先按年级（由低到高）再按生日（由小到大）排序的新文件STUDENT 2；

③ 入学分数在550分以上且年龄不大于20岁的学生按姓名排序的文件STUDENT 3。

图5-15是产生这些文件所进行的排序操作及新文件的数据内容，请读者自行校验其正确性。

```
. USE STUDENTS
. SORT TO STUDENT1 ON POINTS/D
  100% Sorted              6 Records sorted .. Copying text file.
. SORT TO STUDENT2 ON GRADE/A,BIRTHDAY/A
  100% Sorted              6 Records sorted .. Copying text file.
. SORT TO STUDENT3 ON NAME FOR (POINTS>550);
.AND.((YEAR(DATE())-YEAR(BIRTHDAY))<=20)
  100% Sorted              4 Records sorted .. Copying text file.
. USE STUDENT1
. LIST
Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      1  Zhou           4  620.5 08/31/65 .T.     Memo
      2  Wu             2  591.0 01/01/66 .T.     Memo
      3  Zhao           2  581.5 05/16/65 .T.     Memo
      4  Qian           1  560.0 07/24/65 .F.     Memo
      5  Li             1  525.0 03/24/66 .F.     Memo
      6  Sun            3  495.0 12/01/64 .F.     Memo

USE STUDENT2
  LIST
Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      1  Qian           1  560.0 07/24/65 .F.     Memo
      2  Li             1  525.0 03/24/66 .F.     Memo
      3  Zhao           2  581.5 05/16/65 .T.     Memo
      4  Wu             2  591.0 01/01/66 .T.     Memo
      5  Sun            3  495.0 12/01/64 .F.     Memo
      6  Zhou           4  620.5 08/31/65 .T.     Memo
  USE STUDENT3
  LIST
Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      1  Qian           1  560.0 07/24/65 .F.     Memo
      2  Wu             2  591.0 01/01/66 .T.     Memo
      3  Zhao           2  581.5 05/16/65 .T.     Memo
      4  Zhou           4  620.5 08/31/65 .T.     Memo
```

图5-15　数据库文件的排序

（2）索引命令（INDEX）

如同例5-8所示，同一个数据库文件，由于不同用途而需要进行不同的排序。使用SORT命令虽然很方便，但由于建立了许多内容相同而仅仅排列次序不同的文件，从而造成了大量数据的冗余。在更新原始的数据库文件时，这些新生成的排了序的文件不得不重新生成，实际使用中既不方便，也很容易造成数据的不一致。

解决这个问题的一个办法是建立索引文件（.NDX文件）。所谓索引文件是指这样一种文件，它只包含排了序的关键字段的内容以及它们在原始文件中的记录序号。例如，如果关键字段为POINTS，则STUDENTS文件的索引文件内容可以想象为：

关键字段内容（已排序）	记录序号
495.0	3
525.0	4
560.0	2
581.0	1
591.0	6
620.0	5

可见，索引文件的数据量比原始文件少得多，且索引文件并不改变原来文件中的记录的物理位置。此后，原始数据文件中的记录序号称为“物理顺序”，而对应索引文件中的排序称为“逻辑顺序”。需要注意的是，索引文件不是ASCII文件，它也无法在屏幕上显示出来。

建立索引文件需要使用索引命令（INDEX），它的格式如下：

INDEX ON〈关键字段表达式〉 TO 〈索引文件名〉

需要排序的关键字段多于一个时，必须用字符串运算符“+”（并置）把它们组合成一个表达式，其中的数值型字段和日期型字段需分别用函数STR（）和DTOC（）转换成字符串。索引文件中均按关键字段表达式的值增序排列。

例5-9 索引文件的生成

设数据库文件STUDENTS需生成如下几个索引文件：

① 分别以入学分数、姓名、年级、年龄作关键字段的索引文件POINTS.NDX，NAME.NDX，GRADE.NDX和BIRTHDAY.NDX；

② 以年级和年龄作关键字段的索引文件GRADEAGE.NDX；

③ 以姓名中第2个字母作关键字段的索引文件NAME2.NDX；

④ 以入学分数（由高到低）作关键字段的索引文件POINTS1.NDX。

那末使用如下的索引命令即可：

```
. USE STUDENTS
. INDEX ON POINTS TO POINTS
      6 records indexed
. INDEX ON NAME TO NAME
      6 records indexed
. INDEX ON GRADE TO GRADE
      6 records indexed
. INDEX ON BIRTHDAY TO BIRTHDAY
      6 records indexed
. INDEX ON GRADE+DTOC(BIRTHDAY) TO GRADEAGE
      6 records indexed
. INDEX ON SUBSTR(NAME,2,1) TO NAME2
      6 records indexed
. INDEX ON STR((1000-POINTS),6,1) TO POINTS1
      6 records indexed
```

（3）索引文件的使用

有了索引文件以后，用户就可以借助索引文件而对原始数据库文件中的内容进行排序。使用索引文件的方法有两种，一种是在使用USE命令打开原始数据库文件的同时打开需要的索引文件，另一种办法是需要时用SET INDEX命令临时打开。

使用USE命令打开数据库文件的同时，也可以同时打开多达7个有关的索引文件，它的命令格式是：

USE〈数据库文件名〉〔INDEX〈索引文件表〉〕

其中，列在索引文件表中的第1个索引文件即为主索引文件。USE命令在打开数据库文件的

时候，如果不使用索引文件，则当前记录就定位在第1个记录上，即按照物理顺序定位。如果主索引文件为POINTS.NDX时，指针将指向STUDENTS文件中的第3号记录，该记录即为文件打开后的当前记录。此后的所有操作即按索引文件所给出的逻辑顺序进行。

例5-10　使用索引文件进行排序

设已经为STUDENTS文件建立了如例5-9中给出的那些索引文件，如果要按照例5-8中的要求对STUDENTS文件的内容进行排序后输出，则可按照图5-16所示进行操作，但要注意使用索引文件GRADEAGE和GRADE，BIRTHDAY的差别。

```
. USE STUDENTS INDEX POINTS
. DISPLAY ALL
Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      3  Sun            3  495.0 12/01/64 .F.     Memo
      4  Li             1  525.0 03/24/66 .F.     Memo
      2  Qian           1  560.0 07/24/65 .F.     Memo
      1  Zhao           2  581.5 05/16/65 .T.     Memo
      6  Wu             2  591.0 01/01/66 .T.     Memo
      5  Zhou           4  620.5 08/31/65 .T.     Memo
. USE STUDENTS INDEX GRADEAGE
. DISPLAY ALL
Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      4  Li             1  525.0 03/24/66 .F.     Memo
      2  Qian           1  560.0 07/24/65 .F.     Memo
      6  Wu             2  591.0 01/01/66 .T.     Memo
      1  Zhao           2  581.5 05/16/65 .T.     Memo
      3  Sun            3  495.0 12/01/64 .F.     Memo
      5  Zhou           4  620.5 08/31/65 .T      Memo

  USE STUDENTS INDEX GRADE,BIRTHDAY
  DISPLAY ALL
Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      2  Qian           1  560.0 07/24/65 .F.     Memo
      4  Li             1  525.0 03/24/66 .F.     Memo
      1  Zhao           2  581.5 05/16/65 .T.     Memo
      6  Wu             2  591.0 01/01/66 .T.     Memo
      3  Sun            3  495.0 12/01/64 .F.     Memo
      5  Zhou           4  620.5 08/31/65 .T      Memo


. USE STUDENTS INDEX NAME
. DISPLAY FOR (POINTS>550).AND.((YEAR(DATE())-YEAR(;
BIRTHDAY))<=20)
Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      2  Qian           1  560.0 07/24/65 .F.     Memo
      6  Wu             2  591.0 01/01/66 .T.     Memo
      1  Zhao           2  581.5 05/16/65 .T.     Memo
      5  Zhou           4  620.5 08/31/65 .T.     Memo
```

图5-16　使用索引文件进行排序

从例5-10中可以看出，同一数据库文件的不同排序要求，需使用不同的索引文件。但每次使用USE语句时，数据库文件都要被反复地打开并从磁盘上读出。实际上没有这个必要，相反导致工作效率的降低。为此，可以使用只打开索引文件而不再打开数据库文件的命令。它的格式是：

SET INDEX TO〔<索引文件表>〕

其中，索引文件表中的第1个索引文件为主索引文件，若使用这条命令前已经有打开了的索引文件，则执行这条命令后将自动关闭。如果命令中不指出任何索引文件，则该命令的功能是关闭所有已打开的索引文件。

图5-17是例5-10改用SET INDEX命令操作的显示画面。

```
   USE STUDENTS INDEX POINTS1
   DISPLAY ALL
 Recard#   NAME         GRADE POINTS BIRTHDAY MATH100 NOTES
       3   Sun          3       495.0 12/01/64 .F.     Memo
       4   Li           1       525.0 03/24/66 .F.     Memo
       2   Qian         1       560.0 07/24/65 .F.     Memo
       1   Zhao         2       581.5 05/16/65 .T.     Memo
       6   Wu           2       591.0 01/01/66 .T.     Memo
       5   Zhou         4       620.5 08/31/65 .T.     Memo
   SET INDEX TO GRADE,BIRTHDAY
   DISPLAY ALL
 Record#   NAME         GRADE POINTS BIRTHDAY MATH100 NOTES
       2   Qian         1       560.0 07/24/65 .F.     Memo
       4   Li           1       525.0 03/24/66 .F.     Memo
       1   Zhao         2       581.5 05/16/65 .T.     Memo
       6   Wu           2       591.0 01/01/66 .T.     Memo
       3   Sun          3       495.0 12/01/64 .F.     Memo
       5   Zhou         4       620.5 08/31/65  T.     Memo

   SET INDEX TO NAME
   DISPLAY FOR (POINTS>550).AND.((YEAR(DATE())-YEAR(;
 BIRTHDAY))<=20)
 Record#   NAME         GRADE POINTS BIRTHDAY MATH100 NOTES
       2   Qian         1       560.0 07/24/65  F.     Memo
       6   Wu           2       591.0 01/01/66 .T.     Memo
       1   Zhao         2       581.5 05/16/65 .T.     Memo
       5   Zhou         4       620.5 08/31/65 .T.     Memo
```

图5-17 用SET INDEX命令打开索引文件

（4）索引文件的修改

在原始数据库文件的内容有了变化之后，对应的索引文件必须作修改才能保持其有效性。否则，就无法利用索引文件得到正确的结果。

在打开数据库文件的时候，如果也同时打开索引文件，则使用GOTO命令和SKIP命令进行记录定位时，均按逻辑顺序进行。当使用APPEND命令、INSERT命令和REPLACE命令添加、插入和替换记录时，被打开的索引文件也将自动得到修改。但REPLACE命令一次只

能替换记录中的一个字段，否则会引起出错。同样，使用EDIT命令、BROWSE命令或CHANGE命令对数据库内容进行修改之后，所有打开的索引文件也会自动地得到修改。而使用DELETE命令和RECALL命令删除或恢复某些记录时，索引文件不作更改，只在使用PACK命令和ZAP命令的时候，才会使索引文件得到修改。

下面是索引文件自动修改的一个例子。

例5-11　索引文件的自动修改

假设在打开STUDENTS文件的同时，也打开了POINTS和GRADE两个索引文件，然后使用定位命令定位到第3个记录，在它后面插入一个新记录，再检查排序输出的内容，可以发现新插入的记录序号为7。由于分数仅490，年级为1，因而排序在第1位。可见，POINTS和GRADE文件均已被正确地修改过了。操作过程及显示内容见图5-18。从图中还可以看出，接着进行的替换操作，由于仅仅打开数据库文件而未打开任何索引文件，虽然把12号记录的总分增加了150分，但由于索引文件没有修改，因此它仍错误地排在第1号位置。此外由于索引文件NAME没有打开，因此它未得到修改，按NAME排序输出时，新增加的7号记录根本不在其中。当接着使用REINDEX命令把索引文件NAME重新生成之后，再把数据库文件排序输出就完全正确了。

```
. USE STUDENTS INDEX POINTS,GRADE
. GOTO 3
. INSERT
. LIST
Record#  NAME       GRADE POINTS  BIRTHDAY MATH100 NOTES
      7  Wang           1    490.0 07/20/64 .F.     Memo
      3  Sun            3    495.0 12/01/64 .F.     Memo
      4  Li             1    525.0 03/24/66 .F.     Memo
      2  Qian           1    560.0 07/24/65 .F.     Memo
      1  Zhao           2    581.5 05/16/65 .T.     Memo
      6  Wu             2    591.0 01/01/66 .T.     Memo
      5  Zhou           4    620.5 08/31/65 .T.     Memo

. USE STUDENTS
. REPLACE POINTS WITH POINTS+150 FOR NAME="Wang"
      1  record replaced
. SET INDEX POINTS,GRADE
. LIST
Record#  NAME       GRADE POINTS  BIRTHDAY MATH100 NOTES
      7  Wang           1    640.0 07/20/64 .F.     Memo
      3  Sun            3    495.0 12/01/64 .F.     Memo
      4  Li             1    525.0 03/24/66 .F.     Memo
      2  Qian           1    560.0 07/24/65 .F.     Memo
      1  Zhao           2    581.5 05/16/65 .T.     Memo
      6  Wu             2    591.0 01/01/66 .T.     Memo
      5  Zhou           4    620.5 08/31/65 .T.     Memo
. SET INDEX TO NAME
. LIST
```

```
Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      4  Li             1  525.0 03/24/66 .F.     Memo
      2  Qian           1  560.0 07/24/65 .F.     Memo
      3  Sun            3  495.0 12/01/64 .F.     Memo
      6  Wu             2  591.0 01/01/66 .T.     Memo
      1  Zhao           2  581.5 05/16/65 .T.     Memo
      5  Zhou           4  620.5 08/31/65 .T.     Memo
. REINDEX
Rebuilding index - B:NAME.ndx
      7 records indexed
. LIST
Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      4  Li             1  525.0 03/24/66 .F.     Memo
      2  Qian           1  560.0 07/24/65 .F.     Memo
      3  Sun            3  495.0 12/01/64 .F.     Memo
      7  Wang           1  640.0 07/20/64 .F.     Memo
      6  Wu             2  591.0 01/01/66 .T.     Memo
      1  Zhao           2  581.5 05/16/65 .T.     Memo
      5  Zhou           4  620.5 08/31/65 .T.     Memo
```

图 5-18　索引文件的修改

从这个例子中可以看出，修改任何数据库文件时，只有打开的索引文件才会得到自动修改，未打开的索引文件必须用重做索引命令（REINDEX）重新生成一次。注意，REINDEX命令也只对已打开的索引文件才起作用。

此外需要说明的是，使用索引文件打开数据库文件后，INSERT命令与APPEND命令的作用一样，它所插入的记录实际上总添加在文件的尾部。

2. 数据检索

数据库文件的另一种使用方式是数据检索，即从数据库文件中有选择地取出符合用户需要的数据。检索的结果可以显示输出，也可以在打印机上打印出来，或者形成一个新的数据库文件。

dBASE Ⅲ的任何检索操作，几乎都是由三种基本检索运算组成的。这三种基本的检索运算是：选择（SELECT）、投影（PROJECT）和联结（JOIN）。

选择运算就是从数据库文件的全部或部分记录中，把满足规定条件的所有记录选出来，这是通过命令中的〈范围〉短语和FOR/WHILE〈条件〉短语来实现的。

投影运算就是从数据文件的所有字段中，挑选出规定字段的内容，或按要求组合生成新的字段，这是通过命令中给出的一组表达式而指出的。

联结运算需要使用两个不同的数据库文件，它把两个数据库文件中的数据按照规定的条件合并成一个新的数据库文件，这需要使用专门的JOIN命令来完成。

（1）选择和投影

用来实现选择和投影运算的命令有两条，它们的格式是：

DISPLAY 〔OFF〕 〔〈范围〉〕 〔〈表达式表〉〕 〔FOR/WHILE 〈条件〉〕 〔TO PRINT〕

LIST 〔OFF〕 〔〈范围〉〕 〔〈表达式表〉〕 〔FOR/WHILE 〈条件〉〕 〔TO PRINT〕

这两条命令的功能完全一样，差别仅仅在于：DISPLAY在输出显示一屏信息后会暂停一下，而LIST命令则否；另外若不指出范围和条件，DISPLAY命令只显示当前记录，而LIST命令则显示出全部记录。命令中的选项〔OFF〕表示不必给出记录序号，〔TO PRINT〕表示检索出来的数据也在打印机上输出。〔〈表达式表〉〕可以是字段名、变量名或者表达式，它们决定着检索数据时投影运算应如何进行，同时还作为输出信息的栏标题。〔〈表达式表〉〕缺省时表示需要输出除备忘类型以外的全部字段，需要输出备忘字段的内容时，必须在表达式表中明确指出。

另外，第三节中介绍过的COPY命令也具有与DISPLAY命令相似的选择和投影运算的功能，不过检索出来的数据不是显示输出而是形成一个新的数据库文件保存在盘上。下面是使用上述命令进行各种选择和投影运算及组合运算的例子。

例5-12　利用选择和投影运算进行数据检索

设有数据库文件STUDENTS，请按如下要求从文件中检索出有关数据来：

① 入学成绩在550分以上且数学成绩超过100分的所有学生的记录。

② 所有学生的说明信息。

③ 满足条件①的所有学生的姓名、入学成绩及年龄。

图5-19是使用DISPLAY命令完成上述数据检索要求的操作过程及结果。

```
  USE STUDENTS
  DISPLAY FOR (POINTS>550).AND.MATH100
Record#  NAME        GRADE POINTS BIRTHDAY MATH100 NOTES
      1  Zhao        2      581.5 05/16/65 .T.     Memo
      5  Zhou        4      620.5 08/31/65 .T.     Memo
      6  Wu          2      591.0 01/01/66 .T.     Memo
. DISPLAY ALL NOTES
Record#  NOTES
      1  Zhao was a typist before, just started on computer
         courses. If graduate with good result, he will be
         promoted to a senior position with a pay raise.

      2
      3
      4  Li worked mainly with the manager and worked with
         the confidential file that was kept in the
         manager's office
```

```
      5
      6
      7
DISPLAY NAME,POINTS,YEAR(DATE())-YEAR(BIRTHDAY);
FOR (POINTS>550).AND.MATH100
Record#  NAME        POINTS  YEAR(DATE())-YEAR(BIRTHDAY)
      1  Zhao         581.5                          20
      5  Zhou         620.5                          20
      6  Wu           591.0                          19
```

图5-19 利用选择和投影运算进行数据检索

（2）利用索引文件进行快速选择

选择操作除了用DISPLAY和LIST命令进行之外，也可以用第三节中介绍过的LOCATE命令来进行，但它们的速度都比较慢。利用索引文件和另外两条新命令——SEEK和FIND，则可实现快速的选择操作。

SEEK命令的格式为：

SEEK〈表达式〉

它表示按照表达式的值在索引文件中寻找第1个与之相符的记录。若找到，则该记录作为当前记录，否则给出出错信息："No find"，且指针定位在文件尾部，函数EOF()为"真"。

FIND命令的格式为：

FIND〈字符串〉

它表示在索引文件中寻找关键字段内容中头几个字符与规定字符串相同的第1个记录，其余处理均与SEEK命令相同。如果〈字符串〉由存贮变量的内容来决定，则应使用宏替换函数&。

例5-13 利用索引文件进行快速选择

本例要求使用索引文件及SEEK或FIND命令实现下列选择操作：

① 从STUDENTS文件中选出所有二年级同学的记录。

② 从STUDENTS文件中选出所有1965年出生的一年级同学的记录。

图5-20是为实现这些操作所编写的dBASEⅢ应用程序及其运行结果。

```
** EX5-13
SET SAFETY OFF
USE STUDENTS
INDEX ON GRADE TO GRADE
FIND "2"
IF (.NOT.EOF())
    DISPLAY NEXT 8 FOR GRADE="2"
ENDIF
```

```
INDEX ON YEAR(BIRTHDAY) TO BYEAR
SEEK 1965
IF (.NOT.EOF())
    DISPLAY NEXT 8 FOR GRADE="1".AND.YEAR(BIRTHDAY)=1965
ENDIF
SET SAFETY ON
RETURN

  DO EX5-13
      6 records indexed
Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      1  Zhao       2       581.5 05/16/65 .T.     Memo
      6  Wu         2       591.0 01/01/66 .T.     Memo
      6 records indexed
Record#  NAME       GRADE POINTS BIRTHDAY MATH100 NOTES
      2  Qian       1       560.0 07/24/65 .F.     Memo
```

图5-20 利用FIND和SEEK命令实现快速选择操作

其中，语句

SET SAFETY OFF

是为了在操作过程中，当生成一个与盘上已有文件同名的文件时不发出警告信息；使用IF(.NOT.EOF())语句是为了用来判断FIND语句和SEEK语句是否找到了满足条件的记录。只有在文件中确实找到满足条件的记录时，才使用DISPLAY语句进行检索操作，又由于此时指针已指在满足条件的第一个记录上，因此检索速度就加快了。

（3）联结

联结运算可以把两个或多个文件中的数据按照某种条件组合成一个新的数据库文件。例如，从学生文件和教师文件可以产生一个学生—教师对照表文件。

联结运算要求同时打开多个数据库文件，每个数据库文件分别存放在各自的内存工作区中，各个工作区各有一个指针，用来指出哪一个记录是当前记录。dBASE Ⅲ一共提供了10个工作区，因此可以同时打开多达10个数据库文件。

10个工作区中有一个而且只有一个是活动工作区。用来修改文件内容或结构的dBASE Ⅲ命令只对活动工作区中的数据库有效，对其它工作区中的文件无效。而有关数据检索等命令中，用户既可以使用活动工作区中的数据，也可以使用其它工作区中现行记录的数据。使用其它工作区中的数据时，必须指出该工作区的别名及有关的字段名，形式为：〈别名〉->〈字段名〉。

dBASE Ⅲ初态时，1号工作区是活动工作区。但可以使用选择命令（SELECT）来选择其它工作区作为活动工作区，它的格式为：

SELECT〈工作区号/别名〉

工作区号允许1—10，它们的缺省别名分别为A—J。如果用变量名来表示工作区号或别名，

则必须使用宏替换函数表示。

在选择了活动工作区之后，用户可以用USE命令打开一个数据库文件。USE命令打开文件时，还可以为该工作区定义一个别名，从而使编写应用程序时更为方便。为工作区定义别名的命令格式为：

USE〔〈数据库文件名〉〕〔INDEX〈索引文件表〉〕〔ALIAS〈别名〉〕

若不定义别名，则数据库文件名就是它所在工作区的别名。显然同一个数据库文件不能同时在不同的工作区中打开使用。

下面是使用SELECT命令和别名的例子。

例5-14　工作区的选择及别名的使用

```
. SELECT 1
. USE STUDENTS INDEX NAME,GRADE
  SELECT 3
  USE TEACHERS ALIAS T
. SELECT STUDENTS
. LIST NAME,GRADE,POINTS,T->TNAME FOR GRADE=T->GRADE
Record#  NAME        GRADE POINTS T->TNAME
      6  Wu              2      591.0 Chen
      1  Zhao            2      581.5 Chen
```

图 5-21　工作区的选择及别名的使用

其中，因为3区中的文件被打开后，指针始终定位在第1个记录上，因此满足检索条件的仅有两个记录。

dBASE Ⅲ的联结运算是通过JOIN命令实现的。它一次只能对两个文件进行联结，三个以上文件的联结需要通过多次使用JOIN命令才能完成。JOIN命令的格式是：

JOIN　WITH〈别名〉TO〈文件名〉FOR〈条件〉〔FIELDS〈字段表〉〕

它表示把活动工作区中的文件里的记录（从第1个开始），与别名所指出的工作区中的文件中的每一个记录作比较。如果满足规定的条件，则从两个记录中选择所需要的字段（若不指出需要的字段，则表示两个文件的全部字段），组成一个新记录放入新的数据库文件中，直到两个文件中所有记录都相互比较过为止。

显然，JOIN命令执行的时间比较长，而且当两个文件都相当大、符合条件的情况又很多时，可能会产生一个极其庞大的新数据库文件，这是使用时必须注意的。

例5-15　三个文件的联结

已知STUDENTS、TEACHERS和SCORE三个数据库文件，现希望产生一个满足如下要求的新的数据库文件：

① 入学成绩中数学在100分以上且英语满90分的学生情况才包含在文件中。

② 文件中包括学生名、年级、班主任姓名、英语成绩和物理成绩等字段。

操作过程及其结果如图5-22所示。

```
. SELECT 3
. USE SCORE
. SELECT 2
. USE TEACHERS
. SELECT 1
. USE STUDENTS
. JOIN WITH TEACHERS TO STUDENT1 FOR (GRADE=TEACHERS
->GRADE).AND.MATH100 FIELDS NAME,GRADE,TNAME
      4 records joined
. SELECT 2
. USE STUDENT1
. JOIN WITH SCORE TO STUDENT2 FOR NAME=SCORE->NAME;
.AND.SCORE->ENGLISH>=90 FIELDS NAME,GRADE,TNAME,;
ENGLISH,PHYSICS
      3 records joined
. SELECT 4
. USE STUDENT2
. LIST
Record#  NAME        GRADE TNAME       ENGLISH PHYSICS
      1  Zhao        2     Chen            95      70
      2  Zhou        4     He              99      94
      3  Zhou        4     Zhang           99      94
```

图5-22 三个文件的联结

（4）多文件检索操作

dBASE Ⅲ可以同时打开多达10个数据库文件的性能，为实际使用提供了很大的方便。下面以两个具体例子说明多个文件的同时使用。

例5-16 多个文件的数据检索（之一）

设有STUDENTS、SCORE和TEACHERS三个数据库文件，现要求从中找出入学分数最高的学生的姓名、年级、化学成绩以及化学老师的姓名。图5-23就是解决这个问题的一个应用程序及其运行结果。

```
** EX5-16
SET TALK OFF
SELE 1
USE STUDENTS
WMAX=POINTS
WTNAME=NAME
```

```
WGRADE=GRADE
SKIP
DO WHILE .NOT. EOF()
     IF POINTS>WMAX
     WMAX=POINTS
     WNAME=NAME
     WGRADE=GRADE
     ENDIF
     SKIP
ENDDO
SELECT 2
USE SCORE
LOCATE FOR NAME=WNAME
SELECT 3
USE TEACHERS
LOCATE FOR GRADE=WGRADE.AND.COURSE="Chemistry"
SELE 1
DISPLAY NAME,GRADE,SCORE->CHEMISTRY,TEACHERS->TNAME ;
FOR NAME=WNAME
SET TALK ON
RETURN
```

```
  DO C:EX5-16
Record#  NAME      GRADE SCORE->CHEMISTRY TEACHERS->TNAME
      5  Zhou      4                  76 He
```

图 5-23　使用多个文件检索数据（之一）

例5-17　多个文件的数据检索（之二）

使用上述三个文件，找出所有入学成绩在550分以上的学生，并列出每个学生的姓名及物理、化学、英语三门课的平均成绩和任课老师。图5-24是解决这个问题的应用程序及其运行结果。

```
** EX5-17
SET TALK OFF
SELE 1
USE STUDENTS
SELECT 2
USE SCORE
SELECT 3
USE TEACHERS
SELE 1
COPY TO STMP FIELDS NAME,GRADE,POINTS FOR POINTS>550
USE STMP
JOIN WITH TEACHERS TO STMP1 FOR GRADE=TEACHERS->;
GRADE FIELDS NAME,GRADE,POINTS,TNAME
USE STMP1
JOIN WITH SCORE TO STMP2 FOR NAME=SCORE->NAME FIELDS
NAME,GRADE,POINTS,TNAME,CHEMISTRY,PHYSICS,ENGLISH
USE STMP2
```

```
DO WHILE .T.
WNAME=NAME
WTNAME=TRIM(TNAME)
SKIP
     IF EOF()
          EXIT
     ELSE
     IF NAME=WNAME
          WTNAME=WTNAME+","+TRIM(TNAME)
          DELETE
          SKIP -1
          REPLACE TNAME WITH WTNAME
          PACK
          LOOP
     ENDIF
     ENDIF
ENDDO
LIST NAME,TNAME,(CHEMISTRY+PHYSICS+ENGLISH)/3 OFF

SET TALK ON
RETURN
```

```
  DO C:EX5-17
NAME        TNAME          (CHEMISTRY+PHYSICS+ENGLISH)/3
Zhao        Chen                                  83.33
Qian        Wang                                  77.67
Zhou        He,Zhang                              89.67
Wu          Chen                                  85.33
```

图 5-24　使用多个文件检索数据（之二）

从上面例子可以看出，同时打开的若干文件，在活动区文件记录指针移动时，并不引起其它工作区中指针的修改，这在某些应用中很不方便，为此dBASE Ⅲ增加了一条SET RELATION命令，它可以协助解决这个问题。这条命令的格式是：

SET　RELATION〔TO〈关键字段表达式〉/〈数值表达式〉〕
INTO〈别名〉

该命令的功能是把活动工作区中的文件与〈别名〉所指出的工作区中的文件，通过它们的公共关键字段而链接起来（称为建立了关系）。其中，〈别名〉所指的工作区中的文件，必须预先建立并打开按照该关键字段所生成的一个索引文件。这样，活动区中数据库文件的任何引起记录指针变化的操作，均会使〈别名〉区中的记录指针按照索引文件中给出的逻辑顺序而移动，把指针定位在关键字段值相等的第1个记录上。

如果SET　RELATION命令中使用的是数值表达式，则活动工作区中指针变化时，〈别名〉区中将以表达式计算的结果作为记录序号，把指针定位在对应记录上。在这种情况下，〈别名〉区中的数据库文件不再使用索引文件。

无论哪种情况，如果〈别名〉区中无法找到匹配的记录，指针将定位在文件最末记录的

后面。

一个活动工作区只能与一个〈别名〉区建立关系。dBASEⅢ允许最多同时建立10个关系。下面是使用SET RELATION命令的一个例子。

例5-18 关系的使用

设SCORE文件中学生姓名均修改成原来名字的第1个字母，STUDENTS文件已按照姓名NAME建立了一个索引文件（NAME·NDX），现要求列出物理、化学成绩之和超过160分的所有学生的姓名、班级以及物理和化学的成绩。本例中SCORE文件作为主文件使用，它与STUDENTS文件通过NAME字段建立了关系。图5-25是有关的应用程序及运行结果。

```
** EX5-18
SET TALK OFF
SELE 1
USE STUDENTS INDEX NAME
SELE 2
USE SCORE
SET RELATION TO NAME INTO STUDENTS
DISPLAY STUDENTS->NAME,STUDENTS->GRADE,PHYSICS,
CHEMISTRY FOR (PHYSICS+CHEMISTRY)>160
SET TALK ON
RETURN

  DO C:EX5-18
Record#  STUDENTS->NAME  STUDENTS->GRADE  PHYSICS  CHEMISTRY
      1  Wu                           2       77         95
      2  Li                           1       90         84
      3  Sun                          3       80         92
      5  Zhou                         4       94         76
```

图5-25 SET RELATION命令的使用

从本例可以看出，使用了SET RELATION命令后，原来需要用JOIN命令生成一个中间文件才能解决的问题，现在可以直接解决。这样做不但提高了操作速度，而且避免了许多数据不一致性的问题。此外，本例中的STUDENS文件还起着一个对照表的作用，它存放着学生姓名的全名，而SCORE文件中只存放学生姓名的缩写(或编码)，只是在使用数据时才通过对照表把全名显示输出。这种做法，可以减少数据录入的工作量。

SET RELATION命令不带参数时表示取消活动工作区与其它区建立的关系。

3. 数据计算

dBASEⅢ的计算功能比较简单，它直接提供了一些常用的统计操作，如汇总、求平均值、统计计数等，也可以借助于存贮变量和函数实现一些复杂的运算。

（1）计数（COUNT）

计数命令的格式为：

COUNT〔TO〈存贮变量〉〕〔〈范围〉〕〔FOR/WHILE〈条件〉〕

它的功能是把数据库文件中指定范围内满足规定条件的记录的数目存放在存贮变量中。若不给出〈范围〉和〈条件〉，则为文件中所有记录的总数；若不指出〈存贮变量名〉，则计数的结果仅显示在屏幕上，不保存在内存中。

例5-19　计数命令的使用

图5-26　是使用COUNT命令进行统计计算的例子。

```
USE STUDENTS
COUNT
     6 records
COUNT FOR POINTS>550
     4 records
COUNT TO MATH FOR MATH100
     3 records
? MATH
        3
```

图5-26　COUNT命令的使用

（2）求平均值（AVERAGE）

求平均值命令的格式是：

AVERAGE〔TO〈存贮变量表〉〕〔〈表达式表〉〕〔〈范围〉〕
〔FOR/WHILE〈条件〉〕

它把指定范围内满足规定条件的所有记录中的数值型字段（由〈表达式表〉选定）的内容求平均值，并分别存贮在一组存贮变量中。若不给出〈表达式表〉，则将所有数值型字段求出平均值。

（3）求和（SUM）

求和命令的格式是：

SUM〔TO〈存贮变量表〉〕〔〈表达式表〉〕〔〈范围〉〕
〔FOR/WHILE〈条件〉〕

它的功能与AVERAGE命令类似，但计算结果是和数而不是平均值。

例5-20　AVERAGE命令和SUM命令的使用

图5-27是使用求平均值命令和求和命令对SCORE文件进行统计计算的几个例子。

```
. SELECT 1
. USE SCORE
. AVERAGE
      6 records averaged
CHEMISTRY PHYSICS ENGLISH
       84       83      89

. COUNT TO Z
      6 records
. SUM PHYSICS,CHEMISTRY,ENGLISH TO PHY,CHEM,ENG
      6 records summed
   PHYSICS     CHEMISTRY      ENGLISH
       496           502          536
  ? PHY/Z,CHEM/Z,ENG/Z
         82.67          83.67          89.33
. SELECT 2
. USE STUDENTS INDEX NAME
. SELECT SCORE
. SET RELATION TO NAME INTO STUDENTS
. AVERAGE ENGLISH FOR STUDENTS->GRADE="2"
      2 records averaged
 ENGLISH
      90
```

图 5-27 AVERAGE和SUM命令的使用

（4）汇总（TOTAL）

汇总命令是对已经排序的或已经做了索引的数据库文件进行的。它把与关键字段具有相同值的所有记录中的数值型字段(或明确指出的那些数值型字段)的内容进行求和，作为一个记录存放在新的数据库文件中，其中非数值型字段的内容，就是原来文件中与关键字段值相同的那一组记录中的第1个记录的对应字段内容。汇总命令的格式是：

TOTAL ON〈关键字段〉TO〈文件名〉〔〈范围〉〕〔FIELDS〈字段表〉〕
〔FOR/WHILE〈条件〉〕

下面是使用汇总命令的一个例子。

例5-21 TOTAL命令的使用

本例用来生成一个新的数据库文件（GRADETOT·DBF），它共有两个字段：年级(GRADE)及同年级学生的入学成绩总和。图5-28是使用TOTAL命令产生这个文件的过程。

```
** EX5-21
USE STUDENTS
INDEX ON GRADE TO GRADE
TOTAL ON GRADE TO GRADET1 FIELDS POINTS
CLEAR
USE GRADET1
LIST
WAIT
COPY TO GRADETOT FIELDS GRADE,POINTS
USE GRADETOT
LIST
WAIT

RETURN

  DO C:EX5-21
       6 records indexed
       6 Record(s) totalled
       4 Records generated
Record#  NAME        GRADE POINTS BIRTHDAY MATH100
       1  Qian        1     1085.0 07/24/65 .F.
       2  Zhao        2     1172.5 05/16/65 .T.
       3  Sun         3      495.0 12/01/64 .F.
       4  Zhou        4      620.5 08/31/65 .T.
Press any key to continue...
       4 records copied
Record#  GRADE POINTS
       1  1     1085.0
       2  2     1172.5
       3  3      495.0
       4  4      620.5
Press any key to continue...
```

图5-28 TOTAL命令的使用

4. 报表生成

在大量实际问题中，数据处理的最终结果要求能以报表的形式从计算机中打印输出，因而数据库管理系统的功能之一是自动生成报表。报表生成的过程分为两步：设计报表格式和填表并输出。

（1）报表格式的设计

报表格式的设计需要使用CREATE REPORT命令，它的格式是：

CREATE REPORT〈报表格式文件名〉

这是一条全屏幕菜单选择式的命令，它用来建立一个报表格式描述文件（.FRM），供报表

命令使用。

在建立报表格式文件的过程中，系统以会话方式要求用户提供下列信息：

* 报表表头。
* 报表的左边界位置，右边界位置，页宽，每页行数。
* 是否需要每一类一页纸。
* 是否需要把数据库文件中的记录分类列表，并作分部求和。需要的话，给出关键字段名和分类表的表头。
* 是否需要在分类表中再分类列表，若需要则给出关键字段名和子分类表的表头。
* 在报表中为各个字段安排它们的位置（次序及宽度），并给予栏标题。
* 在需要分类并分部求和的情况下，指出哪些字段需要分部求和。

报表格式文件可以用MODIFY REPORT命令进行修改，修改也是以全屏幕菜单选择方式进行的，它的命令格式为：

MODIFY REPORT〈报表格式文件名〉

（2）报表输出

设计了报表格式文件之后，就可以把数据库文件的内容以报表形式输出。报表输出命令的格式为：

REPORT FORM〈报表格式文件〉〔〈范围〉〕〔FOR〈条件〉〕〔PLAIN〕
〔HEADING〈字符串〉〕〔NOEJECT〕〔TO PRINT〕
〔TO FILE〈文件名〉〕

它把指定范围中满足规定条件的所有记录以报表格式文件描述的形式生成一张报表。如果不指定范围和条件，即为全部记录；如果报表格式中要求进行分类列表求和，则数据库文件必须预先经过分类处理或产生并打开相应的索引文件。命令中的〔HEADING〕表示打印在每一页首行上的页标题；〔PLAIN〕表示除了首页之外，其它页均不打印页标题、页序号及日期等信息；〔TO PRINT〕表示报表在打印机上输出；〔NOEJECT〕表示打印之前不走页；〔TO FILE〈文件名〉〕表示产生的报表还应该以正文文件的形式记录在磁盘上。

例5-22 报表生成

本例用来生成一张学生入学成绩报表，其中包括姓名、年级、总分、物理、化学、英语，并要求按年级给出分类列表但不进行分部求和。图5-29是有关的应用程序及打印出来的报表，其中报表格式文件STUFORMAT是预先设计好的，这里不再给出设计过程，读者可自行练习。

```
. SELECT 1
. USE STUDENTS ALIAS S
. SELECT 2
. JOIN WITH S TO STUSCORE FOR NAME=S->NAME ;
FIELDS NAME, GRADE, POINTS, PHYSICS, CHEMISTRY, ENGLISH
      6 records joined
```

```
. SELECT 3
. USE STUSCORE
. INDEX ON GRADE TO GRADE
      6 records indexed
. REPORT FORM STUFORM
Page No.      1
06/01/85
                          Students Report
                          ===============

 Name        Grade   Points   Physics   Chemistry   English
 ------      ------  ------   -------   ---------   -------

** Grade: 1
 Li            1     525.0      90         84         88
 Qian          1     560.0      85         70         78

** Grade: 2
 Wu            2     591.0      77         95         84
 Zhao          2     581.5      70         85         95

** Grade: 3
 Sun           3     495.0      80         92         92

** Grade: 4
 Zhou          4     620.5      94         76         99
```

图5-29 报表生成

（3）标签的生成

除了生成报表外，dBASE Ⅱ还可以用来打印标签。所谓标签，是指从数据库文件中取出需要的字符串类型的数据，按照预先指定的格式，在标签纸上面打印出来。

标签打印的格式由标签格式文件（.LBL）给出，该文件使用下面的两条命令来建立或修改：

CREATE LABEL〈标签格式文件名〉

MODIFY LABEL〈标签格式文件名〉

建立标签格式文件的时候，用户可以指出下列参数：

- 标签的宽度和高度
- 标签左边界位置
- 标签与标签之间的间隔及每排打印的标签数目
- 标签的内容与位置安排
- 说明信息

标签输出需要使用LABEL FORM命令，它的格式为：

LABEL FORM〈标签格式文件名〉〔〈范围〉〕〔SAMPLE〕〔TO PRINT〕〔FOR/WHILE〈条件〉〕〔TO FILE〈文件名〉〕

它为指定范围中满足规定条件的每一个记录按照标签格式文件的要求，分别生成一个标签。

如果命令中有TO PRINT，则打印输出；如果有TO FILE〈文件名〉，则以ASCII文件保存在磁盘上。命令中的SAMPLE用来打印测试标签，以便调节标签纸在打印机上的位置。

例5-23 标签生成

图5-30是一组标签格式描述参数以及使用该组参数为STUDENTS文件所生成的两个标签。

```
Width of label:              50
Height of label:              7
Left margin:                 10
Lines between labels:         2

Spaces between labels:        0
Number of labels across:      1
```

```
                                Label contents:
1 "+-------------------------------------------+"
2 "| Computer Science Department               |"
3 "| Nanjing University                        |"
4 "| Nanjing,China                             |"
5 "|                                           |"
6 "|                        Name: "+NAME+"     |"
7 "+-------------------------------------------+"
```

```
LABEL FORM STULBL
        +-------------------------------------------+
        | Computer Science Department               |
        | Nanjing University                        |
        | Nanjing,China                             |
        |                                           |
        |                        Name: Li           |
        +-------------------------------------------+


        +-------------------------------------------+
        | Computer Science Department               |
        | Nanjing University                        |
        | Nanjing,China                             |
        |                                           |
        |                        Name: Qian         |
        +-------------------------------------------+
```

图5-30 标签格式的定义及标签生成

第五节　应用程序举例

本节通过一个微机用户管理系统UMS（USER MANAGEMENT SYSTEM）的设计，介绍dBASEⅢ应用程序的开发过程，供读者参考。

1. UMS的功能

该应用程序的目的是对用户使用机器情况（上机时间、使用机号、运行情况等）进行登录管理，供随时检索。

（1）上机登记

输入上机证号码后，自动检索登录文件，判明是否有上机资格，如果没有上机资格，提请办理申请手续，然后上机。如果有上机资格，则查询是否有空闲机器，有空闲则分配上机，否则请等待。

（2）下机登记

登记下机时间，撤消所占用的机器，需要时并登录运行情况。

（3）检索

按各种不同的要求检索用户上机情况，检索的关键字有：

① 证号　② 姓名　③ 日期　④ 任意条件

（4）办理申请手续

如用户的证号、姓名，用机期限等。

2. 数据库文件设计

（1）用机登记文件UR.DBF：

```
. LIST STRU
Structure for database : B:UR.dbf
Number of data records :        8
Date of last update    : 06/01/85
Field  Field name  Type        Width     Dec
    1  CODE        Numeric         6
    2  UDATE       Date            8
    3  TIME1       Character      10
    4  TIME2       Character      10
    5  MNO         Numeric         2
    6  UMEMO       Memo           10
** Total **                       47
```

其中CODE是上机证号码，UDATE为上机日期，TIME1是上机时间，TIME2是下机时间，MNO为所用的机号，UMEMO用于记录在上机过程中发生的特殊情况，一般不填写。

（2）申请文件RDB.DBF

```
. LIST STRU
Structure for database : B:RDB.dbf
Number of data records :        7
Date of last update    : 06/01/85
Field  Field name  Type        Width     Dec
    1  CODE        Numeric         6
    2  NAME        Character      10
    3  DATE        Date            8
    4  DAYS        Numeric         3
    5  EDATE       Date            8
    6  RMEMO       Memo           10
** Total **                       46
```

申请文件用于对上机申请进行登录，CODE为上机证号，NAME为上机者姓名，DATE为申请日期，DAYS为用机天数，EDATE为用机截止期（这一字段是多余的，可以用公式DATE+DAYS导出），RMEMO用于记录上机者的情况，如所属单位，介绍信编号等。

（3）机器情况文件MDB.DBF

```
. USE MDB
. LIST STRU
Structure for database : B:MDB.dbf
Number of data records :        3
Date of last update    : 06/01/85
Field  Field name  Type        Width     Dec
    1  MNO         Numeric         2
    2  MTYPE       Character      20
    3  MTIME       Character      10
    4  FLAG        Character       1
** Total **                       34
```

数据库文件MDB.DBF用于记录机器的使用情况。MNO为机号，MTYPE为机型，MTIME为上机者开始上机的时间，FLAG为机器状态，“R”表示等待使用，“*”表示正在使用中。

3．应用程序的设计

整个系统由两部分组成：主程序和过程文件。

（1）主程序

主程序通过主菜单让操作人员（用户）选择要执行的项目，主菜单是通过实用程序dFORMAT设计的。设计的主菜单画面如图5-31所示。

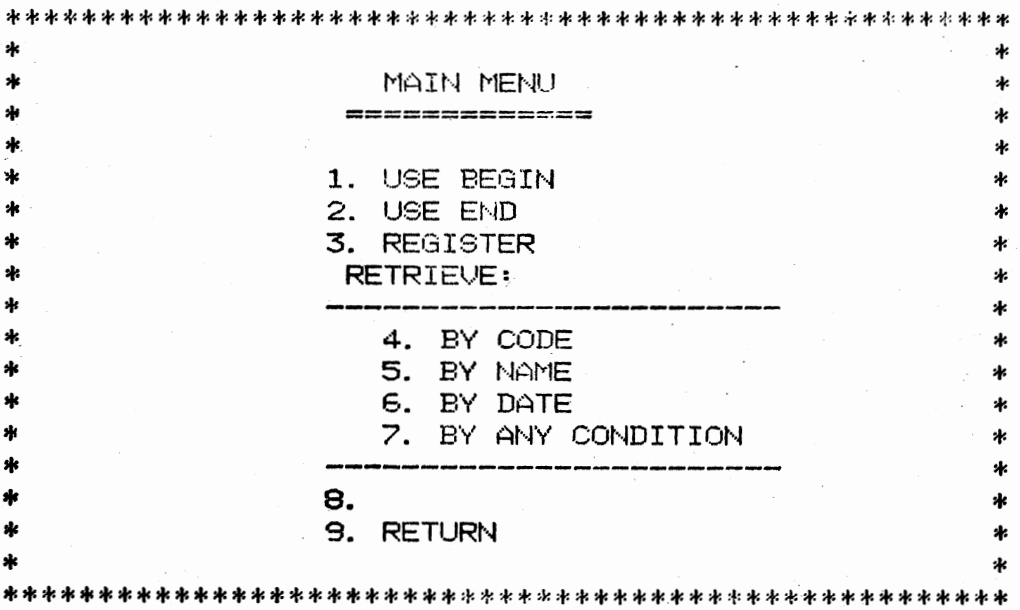


图5-31 UMS的主菜单

相应的格式文件叫做UMSMENU.FMT，该文件由dFORMAT实用程序自动生成，其内容如图5-32所示。

dFORMAT·EXE和dCONVERT·EXE是dBASE Ⅲ的两个实用程序，它们都独立存放在dBASE Ⅲ的实用程序盘片上。使用dFORMAT程序生成格式文件（·FMT）的大致步骤如下：

①从实用程序盘上装入dFORMAT并启动执行。

②从菜单上选择设计画面的操作，然后直接在屏幕上设计所需要的画面。

③选择修改操作可以对所设计的画面格式进行反复修改，直到满意为止。

④通过自动生成功能，把所设计的画面自动转换为格式文件，供应用程序使用。

```
  TYPE UMSMENU.FMT
* umsmenu.FMT
@ 3,12 SAY ."**********************************************"
@ 3,55 SAY "**********"
@ 4,12 SAY "*"
@ 4,64 SAY "*"
@ 5,12 SAY "*                          MAIN MENU"
@ 5,64 SAY "*"
@ 6,12 SAY "*                          ============"
@ 6,64 SAY "*"
@ 7,12 SAY "*"
@ 7,64 SAY "*"
@ 8,12 SAY "*                        1. USE BEGIN"
@ 8,64 SAY "*"
@ 9,12 SAY "*                        2. USE END"
@ 9,64 SAY "*"
@ 10,12 SAY "*                         3. REGISTER"
@ 10,64 SAY "*"
@ 11,12 SAY "*                          RETRIEVE:"
@ 11,64 SAY "*"
@ 12,12 SAY "*                         ------------------------"
@ 12,64 SAY "*"
@ 13,12 SAY "*                            4. BY CODE"
@ 13,64 SAY "*"
@ 14,12 SAY "*                            5. BY NAME"
@ 14,64 SAY "*"
@ 15,12 SAY "*                            6. BY DATE"
@ 15,64 SAY "*"
@ 16,12 SAY "*                            7. BY ANY CONDITION"
@ 16,64 SAY "*"
@ 17,12 SAY "*                         ------------------------"
@ 17,64 SAY "*"
@ 18,12 SAY "*                         8."
@ 18,64 SAY "*"
@ 19,12 SAY "*                         9. RETURN"
@ 19,64 SAY "*"
@ 20,12 SAY "*"
@ 20,64 SAY "*"
@ 21,12 SAY "**********************************************"
@ 21,55 SAY "**********"
@ 23,21 SAY "PLEASE SELECTION:"
@ 23,40 GET ACTION PICTURE "99"
READ
```

图5-32 UMSMENU·FMT清单

UMS的主程序UMSP.PRG列表如图5-33所示，它很简单，这里不再解释。

```
. TYPE UMSP.PRG
*** USER MANAGEMET SYSTEM -- UMS
SET TALK OFF
SELE 1
USE UR
SELE 2
USE RDB
SET FILTER TO EDATE>=DATE()
SELE 3
USE MDB
SELE 1
SET PROCEDURE TO UMSPRO
DO WHILE .T.
ACTION=0
CLEAR
DO UMSMENU.FMT
DO CASE
   CASE ACTION=1
   DO PRO1
   CASE ACTION=2
   DO PRO2
   CASE ACTION=3
   DO PRO3
   CASE ACTION=4
   DO PRO4
   CASE ACTION=5
   DO PRO5
   CASE ACTION=6
   DO PRO6
   CASE ACTION=7
   DO PRO7
   CASE ACTION=8
   DO PRO8
   CASE ACTION=9
   EXIT
   OTHERWISE
   LOOP
ENDCASE
ENDDO
SET TALK ON
CLOSE PROCEDURE
RETURN
```

图 5-33 UMS的主程序

（2）过程文件UMSPRO.PRG

过程文件中包含八个应用程序，它们分别用来实现有关的菜单项目。

PRO 1 用于上机登记，要求输入的信息是上机证号码（CODE），然后根据情况进行如下操作：

① 没有申请或用机期限已过，进行申请或重新申请。

② 如果符合上机条件但没有空闲机器，则请等待。

③ 能上机则请按指定的机号上机，并自动修改机器情况文件。

PRO 2 用于下机登记，要求输入的信息是上机证号码（CODE）和使用的机号。如果需要输入备忘则可通过〈CTRL-HOME〉进入字处理程序进行输入。PRO 2 也要根据输入的信息修改机器情况文件。

PRO 3 用于处理上机申请，要求输入的项目有证号（CODE）、姓名、用机天数。如果有备忘说明也可一起输入。

PRO 4 —PRO 7 是根据不同的要求进行用机情况检索。检索结果既可以在打印机上以报表形式输出，也可以在屏幕上显示出来。使用PRO 7 则可进行任意条件的检索操作。

PRO 8 通过宏替换函数允许操作人员任意地执行dBASE Ⅲ的命令，直到RETURN返回UMS的主菜单。这一过程的提示符为“：”。

过程文件UMSPRO的清单列表如图 5 -34所示。

```
C>TYPE UMSPRO.PRG
*** PROCEDURE OF USER MANAGMENT SYSTEM
PROCEDURE PRO1
CLEAR
@ 5,20 SAY '*****  USE BEGIN  ***** '
@ 8,18 SAY 'Today'
@ 8,38 SAY 'Time'
@ 10,18 SAY DATE()
@ 10,38 SAY TIME()
?
?
?
INPUT "            Please input your CODE: " TO WCODE
SELE 2
LOCATE FOR CODE=WCODE
IF EOF()
         ?
         ?
         ? "           Please register !!"
         ?
         WAIT
         RETURN
ENDIF
SELE 3
LOCATE FOR FLAG='R'
IF EOF()
         ?
         ?
```

```
        ? "          Please Waiting.  ,All\PC is busy "
        ?
        WAIT
ELSE
        REPLACE FLAG WITH "*",MTIME WITH TIME()
        CLEAR
        @ 10,20 SAY 'Please, Machine Number is
        SET COLOR TO 4/0
        @ 10,47 GET MNO
        ?
        SET COLOR TO 7/0
        WAIT
        SELE 1
        APPEND BLANK
        REPLACE CODE WITH WCODE,UDATE WITH DATE() TIME1 ;
WITH TIME(),MNO WITH C->MNC
ENDIF
RETURN
  **
  PROCEDURE PRO2
  CLEAR
  @ 5,20 SAY '*****  USE END  *****
  @ 8,18 SAY 'Today'
  @ 8,38 SAY 'Time'
  @ 10,18 SAY DATE()
  @ 10,38 SAY TIME()
  ?
  ?
  INPUT "          Please input your CODE: " TO WCODE
  SELE 2
  LOCATE FOR CODE=WCODE
  IF EOF()
          ?
          ?
          ? "                Bad code!!"
          ?
          WAIT
          RETURN
  ENDIF
  ?
  ?
  INPUT "           Please your Machine Number: " TO WMNC
  SELE 3
  LOCATE FOR MNO=WMNO
  REPLACE FLAG WITH "*"
  SELE 1
  LOCATE FOR MNO=WMNO.AND.CODE=WCODE.AND.TIME2="
  REPLACE TIME2 WITH TIME()
  ?
  ?
```

```
ACCEPT "          Write Memorandum ?(Y/N)" TO C
IF C="Y".OR.C="y"
?
?" <Ctrl-Home> Into Wordprocessor,<Ctrl-w><CR> Return"
WAIT
CHANGE NEXT 1 FIELDS UMEMO
ENDIF
RETURN
**
PROCEDURE PRO3
SELE 2
APPEND BLANK
CLEAR
@ 10,10 SAY "Your CODE are"
@ 10,25 GET CODE
@ 12,10 SAY "Your Name are  "
@ 12,25 GET NAME
@ 14,10 SAY "Allotted Time  "
@ 14,25 GET DAYS
READ
REPLACE DATE WITH DATE()
REPLACE EDATE WITH DATE+DAYS
?
?
ACCEPT "            Write Memorandum ?(Y/N)" TO C
IF C="Y".OR.C="y"
?
? "<Ctrl-Home> Into Wordprocessor,<Ctrl-w><CR> Ret
WAIT
CHANGE NEXT 1 FIELDS RMEMO
ENDIF
RETURN
**
PROCEDURE PRO4
CLEAR
?
?
?
?
INPUT "            Please input your CODE:  " TO WCODE
SELE 4
USE
SELE 1
COPY TO RTDB FOR CODE=WCODE
?
?
ACCEPT "                  Print(P) or Display(D)" TO C
SELE 4
USE RTDB
IF C="P".OR.C="p"
```

```
            REPORT FORM PPRO TO PRINT
    ENDIF
    LIST
    WAIT
    RETURN

**
PROCEDURE PRO5
CLEAR
?
?
?
?
SELE 4
USE
ACCEPT "            Please input your NAME:   " TO WNAME
SELE 2
LOCATE FOR NAME=WNAME
SELE 1
COPY TO RTDB FOR CODE=B->CODE
?
?
ACCEPT "                    Print(P) OR Display(D)" TO C
SELE 4
USE RTDB
IF C="P".OR.C="p"
        REPORT FORM PPRO TO PRINT
ENDIF
LIST
WAIT
RETURN
**
PROCEDURE PRO7
CLEAR
?
?
?
?
SELE 4
USE
ACCEPT "                    Please input CONDITION:   " TO C
SELE 1
COPY TO RTDB FOR &C
?
?
ACCEPT "                    Print(P) or Display(D)" TO C
SELE 4
USE RTDB
IF C="P".OR.C="p"
        REPORT FORM TO PRINT
ENDIF
```

```
LIST
WAIT
RETURN
**
PROCEDURE PRO8
CLEAR
? "                          Please input any command
DO WHILE .T.
        ACCEPT ":" TO C
        &C
ENDDO
**
PROCEDURE PRO6
CLEAR
?
?
?
?
SELE 4
USE
INPUT "                          Please input DATE : " TO WDATE
SELE 1
COPY TO RTDB FOR UDATE=WDATE
?
?
ACCEPT "                         Print(P) or Display(d)"TO C
SELE 4
USE RTDB
IF C="P".OR.C="p"
        REPORT FORM PPRO TO PRINT
ENDIF
LIST
WAIT
RETURN
```

图 5-34 UMS的过程文件

4. 操作

假设A盘是dBASE Ⅲ系统盘，B盘上记录着UMS的数据库文件和应用程序。打入下列命令：

A > DBASE

• SET DEFAULT TO B:

• DO UMSP

于是屏幕上显示出主菜单。首先选择 8，在机器情况文件中填入机房里可供使用的所有计算机的编号、型号和计算机可使用标记。该文件仅在首次使用UMS时需进行数据输入，在以后添加了新机器或在某些机器发生故障时才需要进行修改，此后的操作即可按菜单进行，这里

不再赘述。

第六节　dBASE Ⅱ与dBASE Ⅲ的比较

1. dBASE Ⅲ命令一览表

为便于查阅，现把dBASE Ⅲ全部命令的格式及简单介绍按字母顺序整理如下：

? 〈表达式表〉

显示表达式表的值。

?? 〈表达式表〉

同?，但结果显示之前不回车换行。

@〈行号，列号〉〔CLEAR〕

无CLEAR时，清除屏幕上指定行、列处右边的内容，有CLEAR时表示清除从指定行、列处开始直到屏幕结束的全部内容。

@〈行号、列号〉〔SAY〈表达式〉〔PICTURE〈格式描述〉〕〕
〔〔GET〈变量〉〔PICTURE〈格式描述〉〕〕
〔RANGE〈下界〉，〈上界〉〕

SAY表示把表达式的值按指定的格式显示或打印在由行号、列号指出的位置上；GET按指定的格式在指定的位置上显示变量的值，以供用户进行编辑修改；RANGE用于指明 数值型或日期型变量的上下界。格式描述中使用的功能符号主要有：

@B　数值数据左对齐
@Z　数值0显示为空字符串
@D　月/日/年　格式〈美国格式〉
@E　日/月/年　格式〈欧洲格式〉
@A　全部为字母
@!　全部为大写字母

格式描述中使用的格式字符有：

9　仅允许数字或＋、－号
#　仅允许数字、空白或＋、－号
A　仅允许字母
L　仅允许逻辑值
X　允许任意字符
N　允许字母和数字
!　小写字母转换为大写字母，对其它字符无影响
$　以$替换数值中高位的0
*　以*替换高位的0
.　指明小数点位置
,　用于整数部分的单位分隔

ACCEPT〔〈提示字符串〉〕TO〈存贮变量〉

从键盘输入字符串。

APPEND〔BLANK〕

为数据库文件添加空白记录。

APPEND FROM〈文件名〉〔FOR/WHILE〈条件〉〕〔SDF/DELIMITED〕

从其它数据库文件或文本文件中取数据添加到当前使用的数据库文件中。

ASSIST

使dBASE Ⅲ进入带提示的、菜单选择式的工作环境。

AVERAGE〈表达式表〉〔〈范围〉〕〔FOR/WHILE〈条件〉〕
TO〔〈存贮变量表〉〕

对满足条件的所有记录求各表达式的平均值。

BROWSE〔FIELDS〈字段表〉〕

对数据库文件作全屏幕全文件编辑修改。

CANCEL

终止程序运行，返回dBASE Ⅲ提示符状 态。

CHANGE〔〈范围〉〕〔FIELDS〈字段表〉〕〔FOR/WHILE〈条件〉〕

对数据库文件中指定记录的指定字段进行修改。

CLEAR

清屏幕，光标回左上角。

CLEAR ALL

关闭所有打开的数据库文件，释放全部存贮变量，选择工作区1。

CLEAR GETS

使此后的一条READ命令忽略在此之前的任何@…GET命令。

CLEAR MEMORY

释放所有存贮变量。

CLOSE〔ALTERNATE/DATABASE/FORMAT/INDEX/PROCEDURE〕

关闭指定类型的文件，但不影响存贮变量。

CONTINUE

与LOCATE命令配合，把指针定位到下一个满足条件的记录。

COPY FILE〈源文件名〉TO〈目标文件名〉

复制任何类型的文件，相当于DOS中的COPY命令。

COPY TO〈文件名〉〔〈范围〉〕〔FIELDS〈字段表〉〕
〔FOR/WHILE〈条件〉〕〔SDF/DELIMITED〔WITH〈分隔符〉〕〕

复制数据库文件中的部分或全部数据为另一个数据库文件或文本文件。

COPY STRUCTURE TO〈文件名〉〔FIELDS〈字段表〉〕〔EXTENDED〕

复制数据库文件的结构，创建结构描述文件。

COUNT〔〈范围〉〕〔FOR/WHILE〈条件〉〕〔TO〈存贮变量〉〕

统计指定范围内满足条件的记录数目。

CREATE〈文件名〉〔FROM〈结构描述文件〉〕

建立数据库文件的结构描述。

CREATE LABEL〈文件名〉

建立标签格式描述文件。

CREATE REPORT〈文件名〉

建立报表格式描述文件。

DELETE 〔〈范围〉〕〔FOR/WHILE〈条件〉〕

删除指定记录。

DIR〔〈驱动器号〉〕〔〈路径〉〕〔〈成组文件名〉〕

显示文件目录，不指出文件名时，则表示仅列出数据库文件目录。

DISPLAY〔〈范围〉〕〔FIELD〈字段表〉〕〔FOR/WHILE〈条件〉〕
〔OFF〕〔TO PRINT〕

显示数据库文件中指定范围内满足条件的数据。

DISPLAY MEMORY〔TO PRINT〕

显示存贮变量的名字、类型和值。

DISPLAY STATUS〔TO PRINT〕

显示系统当前工作状态及参数，以及已打开的数据库文件名，工作区号码，别名；打开的索引文件名及索引文件的关键字段名；已建立的关系；文件查找路径；缺省的盘驱动器号；工作方式开关的状态；左边界位置；功能键的定义等。

DISPLAY STRUCTURE

显示数据库文件的结构描述。

DO〈程序名〉〔WITH〈参数表〉〕

启动或调用应用程序，并传递参数。

DO CASE…ENDCASE

情况语句，用于实现多路分支。

DO WHILE〈条件〉…ENDDO

循环控制语句。

EDIT〔〔RECORD〕〈数值表达式〉〕

编辑指定的某个记录。

EJECT

向打印机发送换页命令，使从新的一页开始打印。

ERASE〈文件名〉

从盘上删除文件。

EXIT

在循环语句中，用来跳出循环转向ENDDO后面的语句。

FIND〈字符串〉/〈数值〉

对带索引的文件按关键字进行快速查找操作。

GO/GOTO〔BOTTOM/TOP/〈数值表达式〉〕

把指针定位到指定的记录。

HELP〔〈关键字〉〕

提供有关命令或函数的使用说明。

IF〈条件〉…ENDIF

条件语句。

INDEX ON〈关键字表达式〉TO〈文件名〉

按指定的关键字生成数据库文件的索引文件。

INPUT〔〈提示字符串〉〕TO〈存贮变量名〉

从键盘输入表达式并将它的值送到指定的存贮变量。

INSERT〔BLANK〕〔BEFORE〕

插入一个记录。

JOIN WITH〈别名〉TO〈文件名〉FOR〈条件〉〔FIELDS〈字段名表〉〕

联结两个数据库文件生成一个新文件。

LABEL FORM〈标签格式文件名〉〔〈范围〉〕〔SAMPLE〕
〔TO PRINT〕〔FOR/WHILE〈条件〉〕〔TO FILE〈文件名〉〕

按指定的标签格式文件为数据库文件输出标签。

LIST〔〈范围〉〕〔FOR/WHILE〈条件〉〕〔FIELDS〈字段表〉〕
〔OFF〕〔TO PRINT〕

输出从数据库文件中检索出来的数据。

LIST MEMORY〔TO PRINT〕

显示或打印出存贮变量的名字、类型和值。

LIST STATUS〔TO PRINT〕

同DISPLAY STATUS。

LIST STRUCTURE〔TO PRINT〕

同DISPLAY STRUCTURE。

LOCATE〔〈范围〉〕FOR〈条件〉

把指针定位到满足条件的第1个记录。

LOOP

跳过LOOP和ENDDO之间的所有语句，回到循环开始处继续循环。

MODIFY COMMAND〈文件名〉

调用正文编辑程序对正文文件如.PRG文件、.FMT文件等进行编辑或修改。

MODIFY LABEL〈文件名〉

修改标签格式描述文件。

MODIFY REPORT〈文件名〉

修改报表格式描述文件。

MODIFY STRUCTURE

修改数据库文件的结构，且保留原数据库文件的内容。

NOTE/*〈字符串〉

字符串不必用引号括住，表示程序文件中的注解。

PACK

清除数据库文件中有删除标志的记录。

PARAMETERS〈参数表〉

说明程序中的参数，它必须是程序的第1个语句。

PRIVATE〔ALL〔LIKE/EXCEPT〈成组变量名〉〕〕〔〈存贮变量表〉〕

当内层程序使用的局部存贮变量和外层程序同名时，PRIVATE把外层程序的同名存贮变量的值暂时隐藏起来，退出程序时再恢复其值。

PROCEDURE〈过程名〉

用于在过程文件中标识一个过程的开始。

PUBLIC〈存贮变量表〉

定义全局存贮变量。

QUIT

关闭所有文件，退出dBASEⅢ返回操作系统。

READ

使@…GET语句中的字段或存贮变量成为输入或编辑状态，以便从键盘为它们赋值。

RECALL〔〈范围〉〕〔FOR/WHILE〈条件〉〕

取消指定记录的删除标记。

REINDEX

重新建立所有打开的索引文件。

RELEASE〔〈存贮变量表〉〕〔ALL〔LIKE/EXCEPT〈成组变量名〉〕〕

删除指定的存贮变量，释放它们占用的内存空间。

RENAME〈老文件名〉TO〈新文件名〉

重命名文件。

REPLACE〔〈范围〉〕〈字段〉WITH〈表达式〉

〔，〈字段〉WITH〈表达式〉…〕〔FOR/WHILE〈条件〉〕

替换当前文件中有关记录的某些字段值。

REPORT FORM〈报表格式文件〉〔〈范围〉〕〔FOR〈条件〉〕〔PLAIN〕〔HEADING〈字符串〉〕〔NOEJECT〕〔TO PRINT〕〔TO FILE〈文件名〉〕

按指定格式把数据库文件中的有关数据以报表形式输出。

RESTORE FROM〈文件名〉〔ADDITIVE〕

恢复保存的存贮变量，若不使用ADDITIVE，恢复前要清除存贮变量区。

RETURN〔TO MASTER〕

返回调用程序，使用TO MASTER则可返回最外层的应用程序。

RUN〈命令〉/！〈命令〉

在dBASEⅢ状态下执行DOS的命令或程序（.EXE或.COM文件）。

SAVE TO〈文件名〉〔ALL LIKE/EXCEPT〈成组变量名〉〕

把全部或部分存贮变量保存到磁盘上。

SEEK〈表达式〉

对已建立索引的数据库文件，把记录指针定位到索引关键字等于表达式值的第1个记录。

SELECT〈工作区号/别名〉

选择文件的工作区，工作区号1—10或A—J，别名在USE命令中定义，缺省值即为数据库文件名。

下列SET命令无说明的请参看第一节第3段。

SET ALTERNATE TO〈文件名〉

SET ALTERNATE ON/OFF

SET BELL ON/OFF

SET CARRY ON/OFF

SET COLOR TO〈正常显示〉〔,〈反视频显示〉〕〔,〈边界色〉〕

为彩色显示器设置显示色、底色和边界色。颜色的定义如下表所示：

颜色	字母表示	数字表示	颜色	字母表示	数字表示
黑	SPACE	0	红	R	4
蓝	B	1	洋红	BR	5
绿	G	2	黄	GR	6
青	BG	3	白	W	7

SET CONFIRM ON/OFF

SET CONSOLE ON/OFF

SET DEBUG ON/OFF

SET DECIMALS TO〈数值表达式〉

SET DEFAULT TO〈盘驱动器号〉

SET DELETED ON/OFF

SET DELIMITER ON/OFF

SET DELIMITER TO〔〈字符串〉〕〔DEFAULT〕

SET DEVICE TO〈PRINT/SCREEN〉

SET ECHO ON/OFF

SET ESCAPE ON/OFF

SET EXACT ON/OFF

SET FILTER TO〈条件〉

SET FIXED ON/OFF

SET FUNCTION〈数值表达式〉TO〈字符串型表达式〉

SET FORMAT TO〈屏幕格式文件名〉

SET HEADING ON/OFF

SET HELP ON/OFF

SET INDEX TO〈索引文件表〉

SET INTENSITY ON/OFF

SET MARGIN TO〈数值表达式〉

SET MENUS ON/OFF

SET PATH TO〔〈路径表〉〕

SET PRINT ON/OFF

SET PROCEDURE TO〈过程文件名〉

SET RELATION 〔TO〈关键字表达式〉/〈数值表达式〉〕INTO〈别名〉

SET STEP ON/OFF

SET TALK ON/OFF

缺省值为ON，控制程序中语句的执行结果要否显示于屏幕或在打印机上打印出来。

SET UNIQUE ON/OFF

SKIP 〔±〈数值表达式〉〕

相对于当前记录定位记录指针。

SORT TO〈新文件名〉ON〈字段$_1$〉〔/A〕〔/D〕

〔,〈字段$_2$〉〔/A〕〔/D〕…〕〔〈范围〉〕〔FOR〈条件〉〕

以指定的字段为关键字，按增序或减序排序，排序结果在新文件中。

STORE〈表达式〉TO〈存贮变量表〉

把一个表达式的值赋给一个或多个存贮变量。赋给一个存贮变量的情况可以使用下面的赋值语句代替：

〈存贮变量〉＝〈表达式〉

SUM〔〈范围〉〕〔〈表达式〉〕〔TO〈存贮变量表〉〕〔FOR/WHILE〈条件〉〕

在指定范围内，对满足条件的记录中的有关字段求和。

TEXT〈文字〉ENDTEXT

在程序中用来输出一段文字。

TOTAL ON〈关键字段〉TO〈文件名〉〔〈范围〉〕〔FIELDS〈字段表〉〕

〔FOR/WHILE〈条件〉〕

对预先排序好或做了索引的文件，生成一个按关键字段分别累计的数据库文件。

TYPE〈文件名〉〔TO PRINT〕

显示或打印磁盘上正文文件的内容。

UPDATE ON〈关键字段〉FROM〈别名〉REPLACE〈字段$_1$〉WITH〈表达式〉

〔〈字段$_2$〉WITH〈表达式〉…〕〔RANDOM〕

根据关键字段，从另一个工作区的文件中取数据对当前文件内容进行更新，被更新和参加更新的数据库文件必须按关键字段排序或做索引。如果命令中使用RANDOM，被更新的数据库文件必须排序或做索引，参加更新的数据库文件则可以不必排序或做索引。

USE〔〈文件名〉〕〔INDEX〈索引文件表〉〕〔ALIAS〈别名〉〕

打开数据库文件和索引文件，索引文件最多可以同时打开7个，相应的.DBT文件被自动打开。

WAIT〔〈提示〉〕〔TO〈存贮变量〉〕

程序暂停，直到键盘输入后才继续下去。

ZAP

清除数据库文件的全部内容。

2. dBASE Ⅲ与dBASE Ⅱ的差别

下面简要地列出dBASE Ⅲ与dBASE Ⅱ之间的主要差别，供熟悉dBASE Ⅱ的读者参考。

（1）数据库文件

① dBASE Ⅲ的数据库文件，无论是记录的总数，记录长度，记录中字段字个数，以及字段的类型，都比dBASE Ⅱ有了很大扩充。

② dBASE Ⅲ的字段名不允许用“：”，而改用“-”代替。

③ 两者的数据库文件的内部结构不同，不经过转换无法直接使用。

④ 使用SKIP命令往回跳过文件头部的记录时，dBASE Ⅱ使记录序号为0，而dBASE Ⅲ保持记录序号为1。dBASE Ⅲ中的指针移动超出范围的处理原则是向上指向第1个记录，且BOF（）函数为真，向下指向最后一个记录序号加1，且EOF（）函数为真。

（2）分类与检索

① dBASE Ⅲ允许一次对多个字段作分类，而dBASE Ⅱ只能对一个字段作分类，dBASE Ⅲ的分类速度有了一定的提高。

② dBASE Ⅲ中的FIND命令找不到满足条件的记录时，使函数EOF（）之值为真，指针定位在文件尾部，而dBASE Ⅱ却把指针定位在记录序号为0的地方。

③ dBASE Ⅲ增加了SEEK命令，它使FIND命令的使用得到了简化。

（3）多文件处理

① dBASE Ⅲ允许同时打开十个数据库文件供使用，dBASE Ⅱ只允许两个。

② 使用当前工作区以外的数据库文件中的数据时，dBASE Ⅲ用别名和字段名来指出，而不再象dBASE Ⅱ那样使用P、S作为字段的前缀。

③ dBASE Ⅲ允许在两个文件之间建立关系，从而简化了联结操作，而dBASE Ⅱ无此功能。

（4）存贮变量

① dBASE Ⅲ允许使用的存贮变量个数已增加到256个，并对全局变量和局部变量进行区分，变量类型也增加了日期类型。变量名中不使用“：”而改用“-”。

② 变量的赋值可以用“=”进行。

③ 数值变量的精度已提高，且允许控制输出数据的小数点位数。

（5）函数

① dBASE Ⅱ的若干函数在dBASE Ⅲ中更改了名字，但功能仍保持不变。

② dBASE Ⅲ增加了一部分数值计算函数、转换函数和日期函数。

（6）程序控制

① dBASE Ⅲ允许使用带参数的程序调用，也允许使用过程文件。

② dBASE Ⅲ增加了EXIT，RETURN TO MASTER等控制语句。

（7）系统控制参数

dBASE Ⅲ可控制的系统参数比dBASE Ⅱ更多，因而使用时更为方便。参见本章第一节

的介绍。

（8）dBASE Ⅲ提供了与MS-DOS的连接手段

在dBASE Ⅲ状态下可以直接使用某些MS-DOS的内部命令（如DIR，TYPE，TIME等），也可以通过RUN命令或！命令调用DOS的外部命令，但目前dBASE Ⅲ1.00版在这方面的功能还不十分完善。

3. dBASE Ⅱ文件与dBASE Ⅲ文件的转换

许多dBASE Ⅱ的用户在改用dBASE Ⅲ时，常常需要把他们原来在dBASE Ⅱ下使用的各种文件（数据库文件、索引文件、存贮变量文件、报表格式文件、屏幕格式文件和应用程序文件）转换成dBASE Ⅲ可用的文件，以免重新输入数据和编制程序。完成这种转换需要使用实用程序DCONVERT.EXE，启动该程序的命令格式为：

DCONVERT〔S:〔D:〕〕

其中，S：表示dBASE Ⅱ文件所在的盘号，D：表示转换后生成的dBASE Ⅲ文件的所在盘号。程序启动后，屏幕上出现如图5-35所示的菜单。菜单项目1—6分别对应着需要转换的六种文件类型。

```
dBASE CONVERT - dBASE III File Conversion Aid  v1.0 6/14/84
      (c) 1984 By Ashton-Tate   All Rights Reserved

                dBASE II --> dBASE III

      1 - Database File          <.DBF>
      2 - Memory Variable File   <.MEM>
      3 - Report Format File     <.FRM>
      4 - Command File           <.PRG>
      5 - Screen Format File     <.FMT>
      6 - Index File Help        <.NDX>
      7 - Un-dCONVERT III->II    <.DBF>

      9 -          Instructions
      0 -               EXIT
```

图5-35 DCONVERT的菜单

在转换数据库文件的时候，字段名中的冒号（：）将被转变为下横线（-）。需要时，选择菜单项目7可以实现逆转换，即把dBASE Ⅲ的数据库文件转换为dBASE Ⅱ数据库文件。当然，被转换的dBASE Ⅲ的数据库文件中，不得包含日期类型和备忘类型的字段，字段的数目不能超过32，记录长度应小于1000字节，文件中的记录数目不得超过65535，否则无法实现逆转换。

转换索引文件时，它的转换结果不能与原数据库文件的转换结果相匹配。所以索引文件

必须在dBASE Ⅲ状态下根据转换后的数据库文件重新建立。

报表格式文件及屏幕格式文件的转换很简单，但转换后的文件不再是ASCII文件。

存贮变量文件转换后得到的文件将比原文件长度增加大约25%，其原因之一是因为数值的精度已从10位提高到16位。变量名中的冒号也被改成下横线。

由于dBASE Ⅲ与dBASE Ⅱ相比，语法方面有了许多变化，因此实用程序不能完全自动转换，许多情况下需要手工辅助进行转换。图5-36是使用DCONVERT程序转换一个dBASE Ⅱ应用程序的结果。其中大写字母印出的命令表示转换程序添加的新命令，当可以使用dBASE Ⅲ的新命令替换dBASE Ⅱ中某条命令时，如例中的ZAP命令，则转换程序会给出注解进行说明。转换后，被转换文件扩展名的第三个字母自动改为B。

```
A>TYPE B:OA25.PRB
use &name
copy to wfs stru extended
use wfs
copy to wf.txt sdf
delete all
pack
append blank
repl field:len with 20,field:name with "t",field:type with "c"
create wfield from wfs extended
use wfield
append from wf.txt sdf
return

A>DCONVERT B: C:

A>TYPE C:OA25.PRG
*!!*        dBASE CONVERT - dBASE III File Conversion Aid  v1.0 6/14/84
*
SET HEADING OFF
SET SAFETY OFF
use &name
copy to wfs stru extended
use wfs
copy to wf.txt sdf
*!! The new 'ZAP' command is much faster than 'DELETE ALL/PACK'.
delete all
pack
append blank
repl field_len with 20,field_name with "t",field_type with "c"
create wfield from wfs extended
use wfield
append from wf.txt sdf
return
```

图5-36 dBASEⅢ应用程序的转换

第六章　IBM PC局部地区网络

近几年来，微型计算机局部地区网络(Local Area Network或Local Network，以下简称局部网）取得了迅速的发展。局部网络把几台、几十台、甚至更多的微型机和外围设备相互连接起来，连接距离一般可达数百公尺至数公里，从而达到微型机之间相互通信，共享软、硬件资源的目的。局部网络结构简单灵活，成本低，可靠性好，已经成为办公自动化等应用领域中的重要环节。

本章首先介绍局部地区网络的一些基本概念和技术，然后具体介绍几种IBM PC的局部网的结构、原理、软件及其操作使用方法。

第一节　计算机局部网的基本原理

1．概述

局部网是指在有限的地理区域内构作的计算机网络，例如把分散在一座楼、一个大院内的许多计算机连接在一起，相互通信、共享资源，组成一个功能更强的计算机网络。根据IEEE的描述，局部网络技术乃是“把分散在一个建筑物或相邻几个建筑物中的计算机、终端、带大容量存贮器的外围设备、控制器、显示器，以及为连接其它网络而使用的网络连接器等相互连接起来，以很高的速度(1—20兆位/秒）进行通讯的手段”。图6-1就是一个典

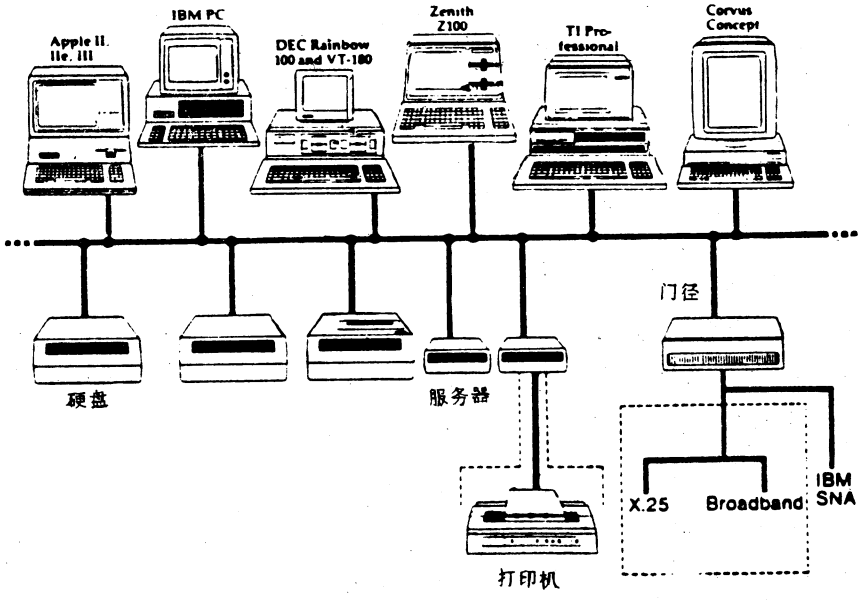


图6-1　微型计算机局部网的组成

型的微机局部网，其中包括微型计算机、磁盘存贮器及打印机等外围设备，通信线路及网络连接器（Gateway，又称“门径”）等部件。一般说来，局部网由下列四个部分所组成：

* 通信线路，即用于相互传输信息的介质及有关设备。
* 控制信息传输的接口机。
* 计算机和智能型外围设备。
* 网络管理软件及有关的应用软件。

局部网络与远程通信网络不同，前者的通信距离仅限于数十米至数公里的范围，后者则可跨越不同地区和不同国家。由于局部网络的地区范围小，因此，易于实现高速数据通信，信息传送速度可以高达数兆至数十兆位/秒。但是远程计算机网一般只使用数千位/秒的速度进行通信。从应用领域来看，目前局部网主要应用于办公自动化，故控制软件相对简单一些，且通信规程与远程网相比也比较单纯。局部网结构简单，成本低，一般不需占用邮电通信线路，也不必使用调制解调器，具有很好的保密性能。因此，局部网可以广泛地使用在机关、工厂、学校、部队等部门，是实现办公自动化的重要环节。

从使用的角度来看，微型机局部网具有如下功能：

* 设备共享　在局部网上所连接的大容量磁盘存贮器、高速打印机、磁带机等设备均可被网上的计算机所共享，提高了整个系统的性能价格比。

* 信息共享　在局部网上的计算机不仅可以使用本机的程序和数据、而且可以使用它机所保存的有关信息，因而增强了网络上计算机的处理能力。

* 相互通信　由于局部网上各计算机全部互连在一起，能进行高速数据通信，因此各台计算机之间可以方便地进行信息交换，如发送电子邮件等。

* 分布式处理　一项复杂的任务可以划分为许多部分，由网络内各计算机分别完成有关部分，使整个系统的效能大为加强。由于局部网络中一般不设置中央计算机，网上各计算机的地位是平等的，从而使网络工作不会因个别计算机的故障而失效，大大加强了网络的坚定性。

* 提高兼容性　微机局部网一般备有对各种类型微机及不同厂家设备的网络接口，从而使网络可以适应技术的发展，通过加入新机种，不断地扩展系统性能和提高处理能力。

* 多种形式信息的通信　除了能进行数据通信外，有些局部网还能传送声音、图象等多种形式的信息，这对于办公自动化系统极为有用。

局部网络的研究工作始于七十年代初，十多年来国外开发了许多成功的局部网络系统。特别是近几年来，微型机局部网的研究、开发十分迅速，局部网产品日益增加，品种繁多，仅连接IBM PC的局部网产品就有几十种。下面对局部网络的基本工作原理如传输介质、网络结构、访问控制方式、通信协议等进行简要介绍。

2. 通信系统的构成

局部网中通信系统的功能就是要可靠、快速地传输信息，因此必须对传输介质、网络结构和信息传输方式等进行考虑和选择。

（1）信息传输介质

局部网络可选择多种信息传输介质，如双绞线、同轴电缆、光缆等，也有使用微波及红外通信技术的。在选用时，主要考虑其性能、成本及使用环境。图 6 - 2 是同轴电缆和光缆的结构示意图。

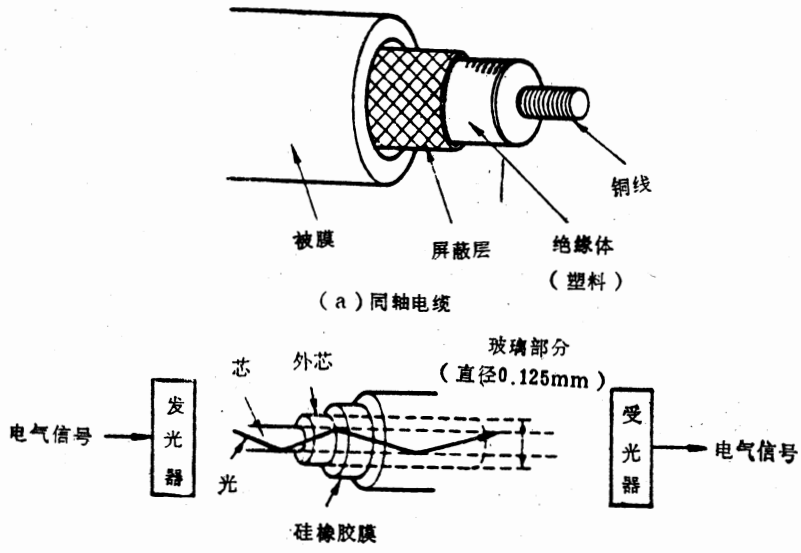


图6-2　同轴电缆和光缆

双绞线是一种价格低廉且易于连接的传输介质。由于导线为双绞形式，因此减少了外界电磁场的影响，信息传递速度一般低于 1 兆位/秒。性能较低的局部网 如OMNINET等 就 使用外包绝缘层的双绞线，信息传输距离通常为数百米。

同轴电缆由于导线外面有屏蔽层，抗干扰性能较强，连接也不太复杂，虽然价格高于双绞线，但传送速度可达数兆位/秒到数百兆位/秒，所以被中、高档局部网所广泛采用。

根据电缆上不同的信号传递方式，同轴电缆又可分为基带同轴电缆和宽带同 轴 电 缆 两类。采用基带方式时数字信号直接加到电缆上，连接简单，距离可达数公里，传输速度低于数十兆位/秒。由于实现简单，基带同轴电缆为较多的局部网所采用，如“以 太”网 等。而采用宽带方式时，信号要调制到规定的高频载波上。例如，利用公用电视系统所用的CATV电缆，传送速度可达数百兆位/秒，还可以进行视频信号的传送。在需要传输数 字、声 音、图象等多种信息的局部网络中，往往采用宽带同轴电缆，如 王 安 公 司 的 局 部 网（Wang-NET）及IBM公司的PC-NET网等。

近年来，以光导纤维管为线芯的光缆，发展迅速。这是因为光缆不受外 界 电、磁 场 干扰，几乎无限制的带宽可实现高达数千兆/秒的传输速率，传输线路与发送器之间在 电 气上完全绝缘，尺寸小，重量轻，传送距离可以达到数百公里，因而是一种引人注目的理想通信介质。目前它的价格较高，光纤的连接又需要专用连接器，因此影响了它的广泛使用，不过其发展前景是十分广阔的。

在电缆敷设不便或多个局部网之间作远距离的连接通信时，可采用无线 通 信 方 式。例

如，用微波通信方式作跨建筑物群间的局部网连接，用红外发射、接收器以及利用天花板上的红外转发器来实现局部网上众多计算机的互连通信等。这些都是有特色的通信介质。此外，利用电话线也可完成低速局部网的通信连接。

表6-1给出了上述通信介质的主要性能和特点，供读者参考。

表6-1　信息传输介质的性能与特点

传输介质	双绞线	基带同轴电缆	宽带同轴电缆	光缆	微波(UHF)
传输速度	＜1兆／秒	＜数十兆／秒	＜数百兆／秒	数千兆／秒	数万兆／秒
成本	很低	中	中	高	高
连接复杂性	简单	不太复杂	不太复杂	复杂	无
抗干扰性	中等	好	好	极好	差
传输距离	数百米	数公里	数十公里	数百公里	极远

（2）网络的拓扑结构——连接形态

构成局部网的微型计算机，大容量磁盘，高速打印机等部件均可看作网络上的一个节点，又称为一个工作站。所谓局部网的拓扑结构，就是网络节点的位置和互连的几何布局。局部网的拓扑结构一般采用总线结构和环形结构，根据应用场合的需要，也有采用星形结构和树形结构的，图6-3是各种局部网络的拓扑结构示意图，图中双空心圆为有控制功能的主站，实心圆为节点或工作站。

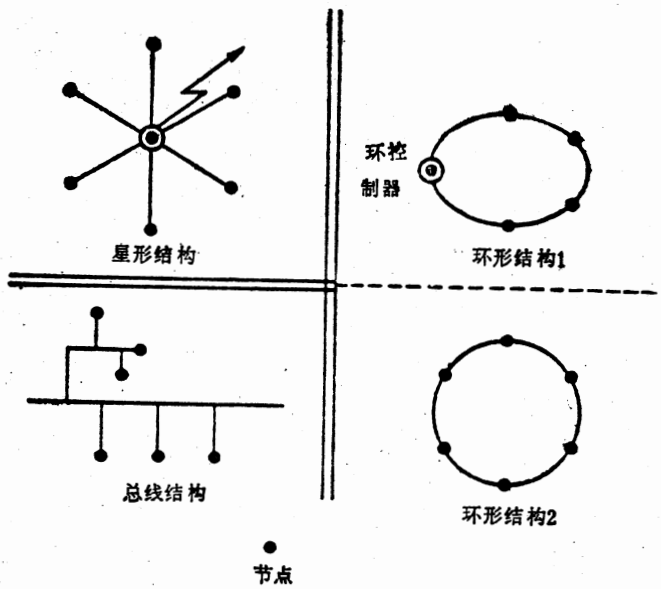


图6-3　局部网的拓扑结构

总线结构为线状连接，即用一条开环、无源的双绞线或同轴电缆通过抽头或收发器把工作站连接到电缆上，形成一条公共的多路访问总线。总线结构连接简单，在总线上添加工作站相当方便。当总线上某节点发生故障时，可简单地把它从总线上断开，并不会干扰或停止

网络的现行工作，因此网络的坚定性很好，这也是大多数局部网采用总线结构的理由。但总线上一般没有控制网络的设备，因而需要对总线上信号传输的冲突作出对策。

IBM PC微机局部网中常见的‘以太’网、PC-NET网、OMNINET网等都是总线结构，采用宽带同轴电缆的Wang-NET也是总线结构。

环形结构是一种闭合的总线结构。每个工作站通过重发器连接到公共的同轴电缆总线上形成一个封闭的环，节点与节点间通信通过重发器进行，信息在环上沿一个方向传送，由被寻址的节点获取信息。由于重发器是有源器件，易于实现高速传送和长距离传送，也易于控制。环形网的缺点是当节点发生故障时，会影响到整个网络不能工作。

值得指出的是光缆很适合于环形结构网，从而可实现传送速率极高、频带宽度非常宽和安全性相当好的局部网络。IBM公司为IBM PC开发的局部网中，也有使用光缆环形结构的产品。

采用环形结构的局部网还有IBM SERIES/1网络，POLYNET网，CLUSTER ONE网等。日本的H-8644 LOOP NET网（日立）、TOTAL-LAN-RING（东芝），均是采用光缆为传输介质的环形网。

树形结构的网络适合于军事单位、政府机构等上、下级界限相当严格的部门使用。处于不同级别的节点分担不同的职能，网络中任一通路出现故障时，只影响网络局部的运行，它的扩充性能也很好。由于这种结构与具体应用系统配置有关，通用产品还不多见。

星形结构的中央节点是充当整个网络控制的主控计算机，它与呈星形配置的其它所有节点相连接。各工作站间相互通信时必须通过中央节点。所以当中央控制装置发生故障时，整个网络便不能工作。此外，当众多节点同时工作时，中央节点将因负担过重而成为溢口。所以星形结构较适合于以电话交换线路进行通信的低速系统。现把各种局部网络拓扑结构的性能列在表6-2，以资比较。

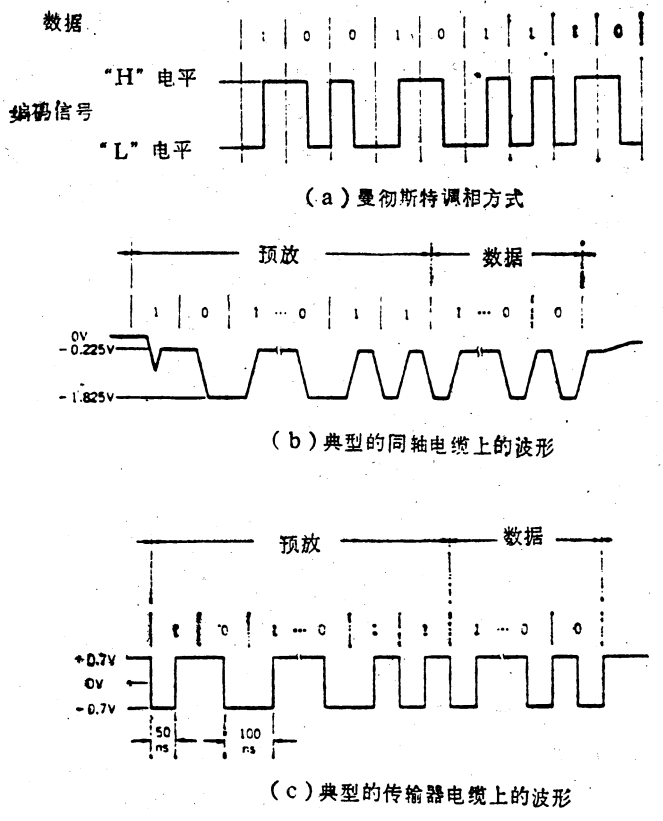
表6-2　局部网络拓扑结构比较

连接方式	连接费用	控制的复杂性	可扩展性	灵活性	适用领域
星形	中	较复杂	较差	较差	小型，低价格
环形	低	中等	中等	中等	分布式控制，实时应用
总线	低	不复杂	好	好	低价格，OA
树形	中	不复杂	好	很好	专用系统

（3）信息传输方式

信息传输方式指的是使信息能正确、可靠地在介质上进行传输的各种考虑。

① 信号变换方式　这是指如何把逻辑信号‘1’和‘0’变换成适合于在线路上传输的物理形式。基带传输局部网常常采用不同的电压或电流值直接与逻辑信号相对应的方式。图6-4是在以太网上采用的曼彻斯特调相信号原理图。这是一种双脉冲调制方式，1信号用从负电平跳变到正电平的两个脉冲表示，0信号用从正电平跳变到负电平的两个脉冲来表示。图中还画出了同轴电缆总线上信号的波形和经过传送接收器整形后的波形。由此可以看出，为使传输的数据信号在传送中达到稳定的电平状态，必须在发送数据信号前加上一段预



（a）曼彻斯特调相方式

（b）典型的同轴电缆上的波形

（c）典型的传输器电缆上的波形

图6-4 以太网采用的曼切斯特调相方式

放信号。以太网使用64位的预放信号“101010……”，传送完成后还需有电平恢复期，以确保信息传输的可靠性。这种调相方式易于实现高速信息传输，同步和控制也比较容易，故应用较多。在采用宽带同轴电缆作为传输介质时也常常使用CATV（电缆电视）的技术，即把逻辑信号调制成UHF（甚高频）信号后进行传输。使用光缆时则把信号调制成光强或波长的形式然后在光缆中进行传输。

② 差错控制方式　传输线路由于种种原因会引起信息传输的差错，这些差错常常带有突发性、成群性。因此，必须在信息传输过程中具有检错和纠错措施。最常用的方法是使用多项式码（也叫循环码或CRC码）。8位字符常常采用下列两种多项式：

$$CRC-16=X^{16}+X^{15}+X^{2}+1$$

$$CRC-CCITT=X^{16}+X^{12}+X^{5}+1$$

它们能查出所有的单位错和双位错，以及所有具有奇数位的差错和所有长度≤16位的突发性错误，还能查出99.997%的17位突发性错误和99%以上的18位或更长的差错。

③ 同步方式　局部网通常使用同步通信方式，以确保信息的高速传递。采用的方式有每位取同步时钟的位同步，以字符为单位取同步的字符同步，以及识别每个传送组的起始和终止的帧同步等多种形式。在微机通信上常用的异步通信方式，由于速度太低，不适宜在局部网上使用。

（4）安装设计与高可靠性对策

网络工作站与传输线路之间的接口部件一般可分成三部分（图6-5），其中信号变换部分是根据传输介质和信号变换方式及连接方法的不同进行设计的，这一部分包括驱动电缆和接收信号的发送器以及接收器，有调制解调器及时钟分离电路等。对双绞线常常使用差动驱动电路，而用同轴电缆时则要使用低阻抗驱动电路，对于光缆则要以发光二极管或半导体激光管送信，用特种光电二极管等进行收信。

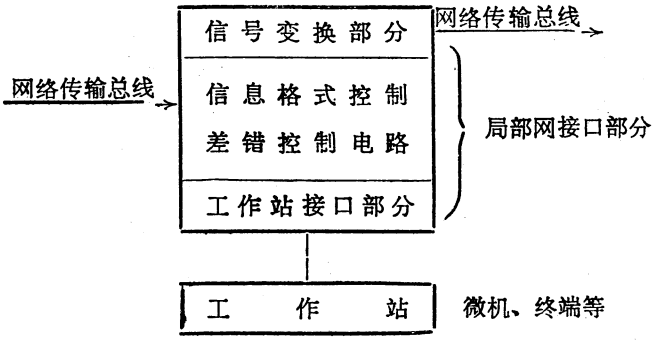


图6-5　局部网与工作站的接口部件

环状局部网中，由于节点相互串联，为了提高网络的可靠性，对故障发生的处理应有多种对策。图6-6是环状网络常用的提高可靠性的一些做法。

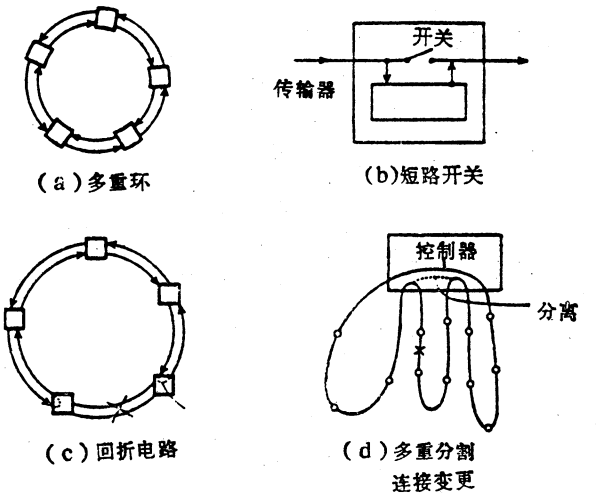


图6-6　环形网络提高可靠性的对策

此外，信号变换部分对传送线路的阻抗匹配和接地电平的分离也是很重要的，稍不注意就会产生不可靠的信息传输。图6-7是同轴电缆与传输器之间分别接地的示意图。

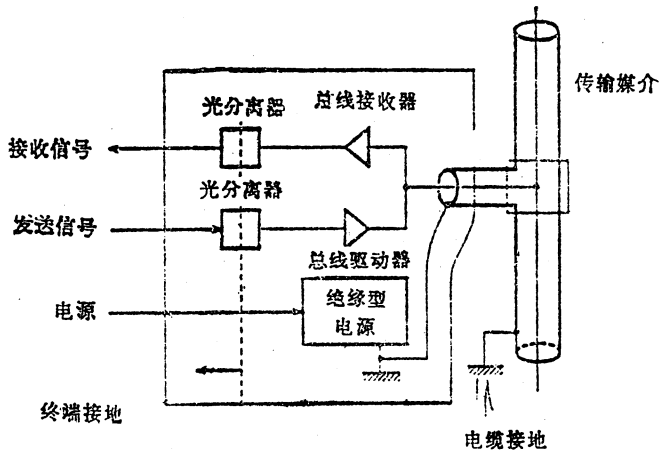


图6-7 同轴电缆连接时的接地方式

3. 访问控制方式

局部网为了进行高速信息传输，一般均采用分组交换方式而不采用线路交换方式，信息在传送时直接送到网络的传输总线上。根据网络拓扑结构可知，由于网络中一般不设主控计算机，因此访问操作（向传输总线送取信息的操作）由各节点处理机自行控制。这样，连接在网络上的某些节点若在同一时间都企图访问网络总线时，就必然发生冲突，从而导致信息传输的错误。所以必须采用能合理解决访问冲突的控制方法。

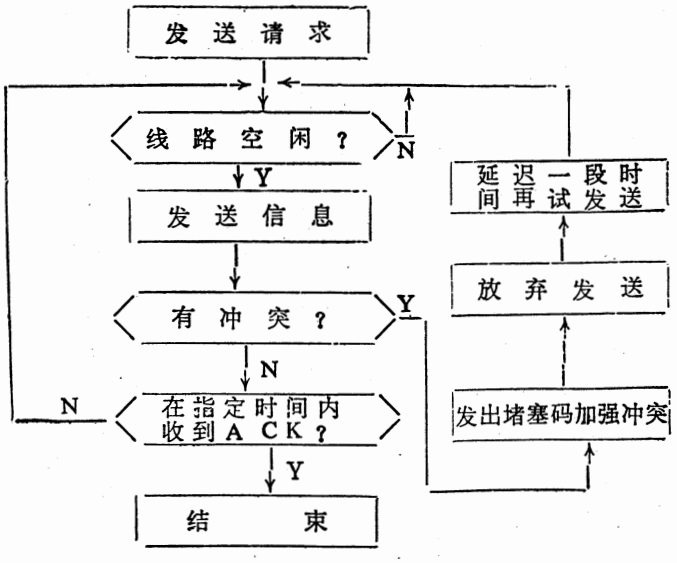
目前，大多数局部网采用的访问控制方式有两种：CSMA/CD方式和Token Passing方式。

（1）CSMA/CD（Carrier Sense Multiple Access/Collision Detect）

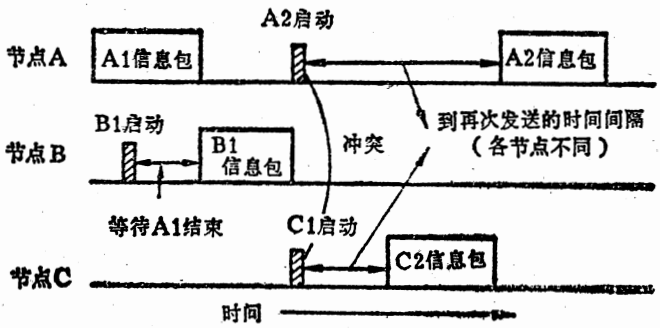
CSMA/CD叫做“线路监听多次存取/碰撞检测法”，多数总线结构的局部网都采用这种方式。当发送点把信息包发送到总线上时，网上其余节点几乎可以同时收到信息，这时，这些节点首先分析收到的信息包中的目标节点地址。若与某节点本身的地址一致，该节点就把跟在后面的数据读入节点机，如果未发现信息传输中有差错发生，便向送信节点发去确认信息包（称“ACK”包）；若接收时发现信息传输有错，则舍去接收到的数据。送信节点在送出信息包后便计算时间。如果一定时间内收不到接收节点发回的ACK信息包，就要重新发送信息，若始终接收不到ACK包，便作为传输失败处理。

由于总线上有许多节点，每个节点都可以独立地向总线发送信息包，因此往往可能发生冲突（碰撞）。如果没有有效措施来避免冲突的话，网络就无法正常工作。

采取CSMA/CD访问控制方式时，节点送信前首先必须检测总线是否空闲，若空闲就开始送信，否则就要等待总线为空闲时才发送信息，或者按一定算法等待一段随机时间后再行发送，这种方式称为竞争发送。其流程图如图6-8(a)所示。图6-8（b）为三个节点发生访问冲突时采用CSMA/CD的处理过程。从图中可以看出，节点A正在发送信息包A1的时候，节点B提出要求发送信息包B1，但由于节点B检测到网络不空，所以等待信息包A1传送结束后再发送B1，而在B1发送结束后，节点A和节点C又同时发出传送信息包A2、C1的要求，



（a）以太网竞争发送流程图



（b）冲突——后退处理

图6-8 CSMA/CD的原理图

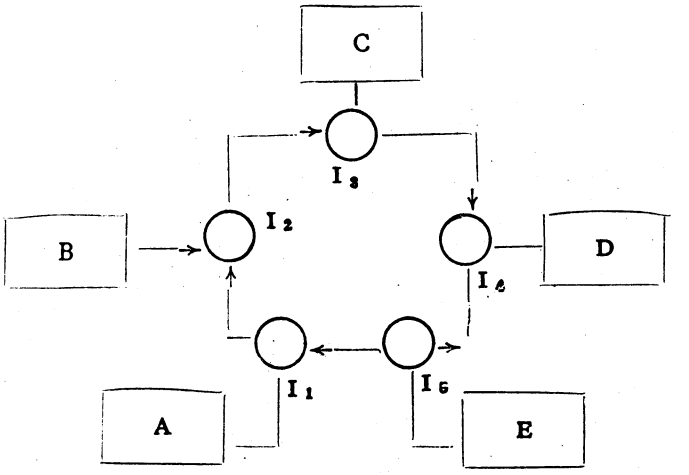
于是节点A和C都各自按照一定策略等待一段时间，然后再开始发送。由于各节点等待时间不同，使信息包C1和A2都得到了发送机会而不再发生冲突，因此提高了总线的利用效率。

CSMA/CD的主要优点是简单、可靠、传输延迟小且成本低，但它不能适应实时控制的需要，传输效率不高，只能在负载不太重的局部网中使用。

（2）令牌传递方式（Token Passing）

令牌传递法是环形结构局部网络经常采用的一种访问控制方式。由于在环形结构总线上，某一瞬间可以允许发送信息的节点只能有一个，因此有一个称为“Token”（令牌，也叫传递标志）的送信权数据在网络环路上不断传递，只有拥有此令牌的节点，才有权向环路上发送信息，而其它节点仅允许接收信息。节点在送信完毕之后，便把令牌交给网络上的下一个节点，如果此节点没有信息需要发送，便将此令牌再顺次交给下一节点。因此，表示送信权的令牌在环形总线上不断循环，环上每个节点都可获得送信权，而任何时刻只会有一个节点利用环路传送信息，因而在总线环路上保证不会发生送取信息包的访问冲突。

图6-9是令牌传递方式的工作原理。图中Ii为通信处理机，它的一个入口和一个出口分别与环形通路相接，接口机中有缓冲器，用来存贮转发信息。图6-9（b）是在环形网上传输的信息包的大致格式，它由开始标志指示信息包的开始，目标地址是本信息包的接收站地址，源地址是发送本信息的站地址，正文即为信息包中的数据，校验字节用以检查信息传输中的差错状况，状态位则用来指示此信件发出后是否为目标节点所接收，结束标志用来指出此信息包结束。



（a）通信处理机与工作站的连接

开始标志	目标地址	源地址	正　文	检验字节	状　态	结束标志

（b）信息包格式

图6-9　令牌传递方式的工作原理

若A站要发送数据给C站，则A站把目标地址和要发送的数据交给本站接口机I_1，由I_1组织成规定格式的信息包。一旦I_1从环上得到令牌（一个专用的信息包），它就从出口发送出该信息包。而I_2从其入口收到此信息后，查看目标地址与本站地址不符，便将原信件依次转送给I_3，I_3在查看目标地址时，即得知此信件是给本站的，便根据校验字节进行查错，如传输的信息没有差错，便将此信息包收下，并修改状态位，表示此信件已被正确接收，此时I_3再把原信件沿I_4、I_5送回I_1，I_1从返回信件的状态位得知信息传送已获成功，便从环上取消此信件，再把令牌放出转交给I_2。这样就完成了一次站间通信过程。

采用令牌传递方式的局部网，网上每一个节点都知道信息的来去动向，保证了较高的信息传输的确定性。由于能算出信息传输延迟时间，因此比较适合于实时系统中使用，而CSMA方式的信息传输时间是波动的。令牌传递方式中信息包长度不定，但对不同长度的信息包都有较高的传输效率，即使在负载增大的条件下也能可靠地工作，而不需冲突检测机构。这种方式的主要缺点是由于网络要求严格定时(同步)，因而增加了设计的复杂性，节点

加入及撤出都比较复杂。

表6-3是CSMA/CD和Token Passing两种访问控制方式的性能比较。

表6-3　网络访问控制方式的性能比较

性　　能	Token　Passing	CSMA／CD
传输延迟	与站的个数N成正比，N受限制	与N无关，N受限制小
介　质	双扭线，同轴电缆，光缆	双扭线，同轴电缆，用光缆有困难
可靠性	比较低	较　高
通路长度	比较长	有限制
驱动器负载	单个负载	多负载
硬件实现	比较容易	比较难
优先权	可以相同，也可加优先权	加优先权比较难
网的负载	对站的个数不灵敏	站太多会饱和

4．局部网络的通信协议

局部网工作时必然要进行各工作站之间的相互通信和对话。为此，对话双方预先需要做出某些约定，这种用于保证通信能正确进行的约定就称为通信规程或通信协议。不同局部网的通信协议各不相同。近年来，随着局部网络的迅速推广，网络通信协议的标准化是迫切需要解决的一个问题。

（1）ISO的七层通信协议模型

国际标准化组织（ISO）推荐的用于计算机网络的开放系统互连（OSI）的参考协议，规定把整个通信协议分成七层，各层之间，既相互独立实现自身的功能，又彼此联系，组成低层和高层的关系。图6-10是七层协议模型的示意图。

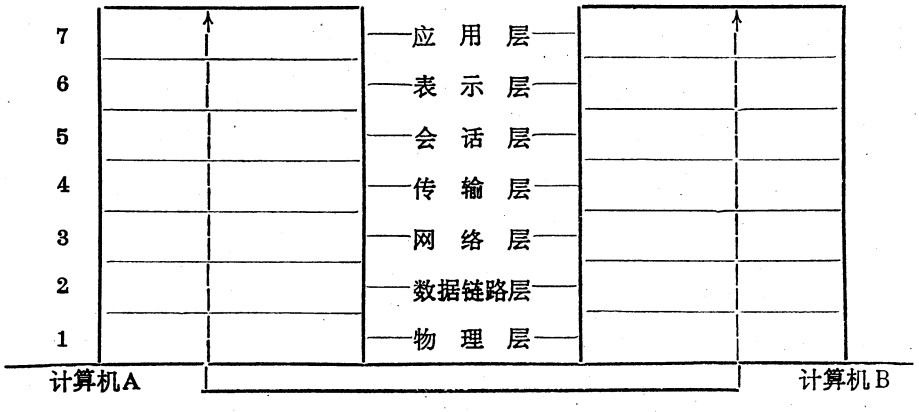


图6-10　ISO　OSI通信协议的七层模型

第1层是物理层。这一层在通信站之间提供交换“1”与“0”的能力，它主要对通信的

物理参数作出规定，如通信介质、调制技术、传送速率、接插头等有关局部网的电气和机械特性都在该层进行说明。

第2层称为数据链路层。它提供了信息如何在通信线路中可靠地传输所需要的功能，例如信息的分帧（数据加上报文头）、寻址、差错校验以及传输线的访问控制方式等。

第3层是网络层。它处理信息包从发送节点经由中间一些节点到达接收节点的路径选择。在局部网络中往往只有一条通路，因此不存在路径的选择问题。但当涉及几个局部网互连时就需要选择路径。本层控制站间信息的传送，并为第4层的数据传送建立连接。

第4层为传输层。这一层的主要任务提供可靠的主机到主机的通信，为第5层服务，并对第5层屏蔽通信网的具体硬件实现的细节。这一层往往由输入输出驱动程序来完成。

第5层是会话层。任务是建立、管理和拆除进程到进程之间的连接，处理同步和恢复问题，负责把面向网络的会话地址变换为相应工作站的物理地址等，此层常置于操作系统中。

第6层是表示层。它负责把数据从一种格式转换成另一种格式，进行不同文件格式的转换，甚至是不同类型的计算机、终端设备和数据库之间的数据格式转换等，这些功能常常由一种可以由用户调用的库程序来提供。

第7层是应用层。它处理网络应用方面的实用程序，诸如用户录入、电子邮件协议，分布式数据库的存取等都由该层处理。这一层是面向用户的。对不同的应用，有不同的要求，所有其它较低的层次都支持应用层。

上述七层模型是对一般计算机网提出的，对于局部网尤其是微型机网不尽适合。已有的局部网通信协议均是以此七层协议为基础，再考虑到局部网的特点进行必要的简化与修改而制订的。

（2）IEEE 802局部网络协议

办公自动化和企业管理是局部网最广泛的应用领域。在这一应用领域中，美国电气与电子工程师协会（IEEE）起着重要的作用。1980年该协会成立了IEEE局部网标准委员会（称IEEE 802委员会），几年来陆续提出了几种局部网标准。1983年3月经过IEEE的计算机通信委员会（TCCC）通过的C草案是目前局部网标准的依据，简称为IEEE 802局部网参考模式。

IEEE 802要求局部网具有如下功能：

* 能支持各种数据通信功能，可以连接各种类型的设备（包括计算机、终端、大容量存贮器、打印机、绘图仪、复印机、传真机、电话、电视摄像机等），可以连接其它局部网或远程网。
* 传输距离大于2公里。
* 传输速率为1—2 Mbps。
* 可支持的设备至少为200个，并能方便地增减设备。
* 访问控制方法为CSMA/CD，Token Bus（总线网令牌传递方式）和Token Ring（环状网令牌传递方式）。

IEEE 802局部网参考模式如图6-11所示，它与ISO OSI的第1、2层相对应。IEEE802的物理层分为四部分：物理信号层（PS）、接口电缆、物理介质附属设备（如收发器等）和物理介质。介质访问控制（MAC）体现了传输介质分享算法，即访问控制方式。逻辑链路

控制（LLC）提供寻址 差错控制以及信息流控制等功能。表6-4是IEEE 802局部网标准（草案）的各种选择方案。

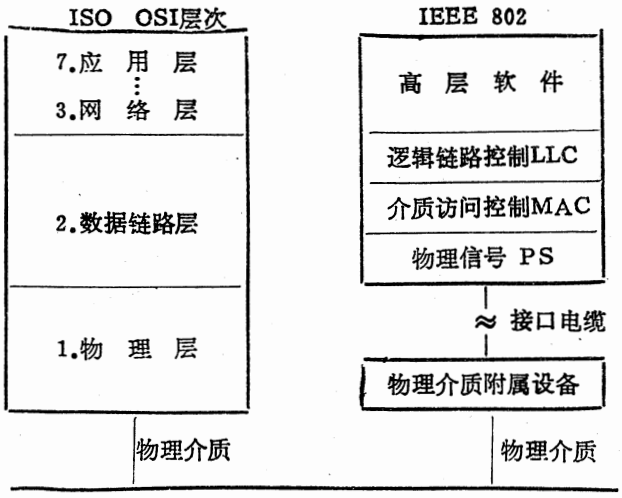


图6-11 OSI七层模型与IEEE802参考模型的对应关系

表6-4 IEEE802局部网标准的各种选择方案

IEEE 802		选择方案							
逻辑链路控制(LLC)	站点类别	1.数据报工作 2.数据报工作和虚电路工作 3.数据报和确认数据报							
	服务访问点	一个（全零地址） 多个							
介质访问控制(MAC)	地址格式	16位 48位——按本地管理分配地址或按通用乘积码分配地址							
	存取控制	CSMA/CD			Token Bus			Token Ring	
介质访问单元(MAU)	速率								40M
							20M		20M
		10M	10M	10M		10M	10M		
			5 M			5 M	5 M	4 M	4 M
			1 M		1 M		1 M	1 M	
	信号	基带	残边带		连续相位	相位相关	相移键控	基带	基带
			宽带				宽带		
	编码	曼彻斯特	密勒码		微分曼彻斯特			微分曼彻斯特	
	拓扑	树	有根树		总线	总线	有根树	环	环
物理介质		同轴电缆		光缆	同轴电缆			屏蔽线对	同轴电缆

5．IBM PC局部网简介

以IBM PC机为节点的局部网络有数十种，这里仅对几种有代表性的网络产品的特性和构成作概括介绍。

（1）Ethernet（以太网）

Ethernet网是一种较早出现的局部网络产品，IBM PC为节点机的以太网主要参数如下：

网络结构：总线式
访问控制方式：CSMA/CD
调制方式：基带
传输介质：同轴电缆
数据传输速率：10Mbps
最长距离：51.5KM
可连接节点数：1024

其主要特征为：

* 使用总线方式，一根同轴电缆就可以连接使用，敷设简单，投资少。
* 通信协议的低两层由硬件支持，各类设备容易连接入网。
* 网络为被动型，网络控制分散于各节点，故可靠性较高。
* 设备的增加、变更很容易，扩充性好，通过使用插头和重发器可简单地实现同轴电缆的延伸。
* 各种厂家的设备均已可连接入网，如DEC、DG、HP等公司的计算机产品均已实现连网接口，通过“门径”（Gateway）部件还可实现与大型远程计算机网相连接。

以太网的缺点是：当负载增大，即需要传输大量信息时，响应时间较长，传送过短的信息时效率不高。

（2）Omninet局部网

这是面向微型计算机的普及型局部网，多种8位以及16位微机均可联网，由美国Corvus系统公司开发。网络的主要参数如下：

网络结构：总线式
访问控制方式：CSMA/CA
调制方式：基带
传输介质：双绞线电缆
数据传输速率：1 Mbps
最长距离：1.2KM
可连接节点数：64

Omninet网的主要特征如下：

* 用低价的双绞线作为信息传输介质，敷设、连接简单，成本低。
* 传输硬件可以支持七层模式的低四层通信协议，从而易于实现异种微机在同一网

络上的连接。例如，IBM PC和Apple Ⅱ等微机均可连接在一个网络上相互通信，共享资源。

* 访问控制方式为CSMA/CA，用软硬件结合的方法来解决总线上的访问冲突，接口部件造价低。

* 提供多种低价的磁盘服务器和打印服务器，使用录像带作为硬盘后备存贮器，具有与电话线路连接的Modem设备和与其它局部网或远程网相连接的“门径”（Gateway）设备，使网络能适应众多的应用领域。

但是Omninet网的传输速度低，负载能力差，这是它的不足之处。

（3）PC-NET

这是美国AST公司专为IBM PC及PC/XT开发的局部网络。照顾到该机的特点，网络直接支持MS-DOS操作系统，使用简便，网络结构简单，建网也比较容易，它的主要参数如下：

网络结构：总线方式
访问控制方式：CSMA/CD
调制方式：基带
传输介质：标准75Ω CATV同轴电缆
信息传输速率：1 Mbps
最长距离：2100米/900米
可连接节点数：256站以上

PC-NET 实现的主要功能有：

* 磁盘共享　PC-NET的节点有两类，一类是共享PC（SPC），一类是用户PC（UPC）。网络上的UPC可以访问安装在SPC上的硬盘存贮器，它们可以把SPC所安装的硬盘作为共享的资源。

* 文件加锁（Locking）　当多个用户访问一个共享磁盘时，为了避免冲突，PC-NET提供了两级文件加锁措施。DOS命令级加锁让用户在使用中给文件建立标志，应用软件级加锁则允许对指定的记录进行存取控制。

* 远程命令执行　网络允许UPC的用户发送DOS命令给SPC执行，就好象用户在直接使用SPC键盘操作一样。对远程命令执行还提供了优先权管理。

* 打印机共享　允许网络上的PC机共享连接在其它PC机上的打印机资源，而不必考虑节点的名。

* 异步通信共享　允许网络上的某台PC机调制解调器（MODEM）被网络上其它PC机所共享，实现异步通信。

* RAM盘　使用Super Drive程序，可使PC机内的随机存贮器（RAM）的一个部分模拟一台磁盘驱动器，提供快速访问能力。

* 脱机打印　使用Superspool程序实现打印作业的后台处理，提高了系统利用效率。

（4）PLAN Series局部网

这是美国Nestar公司将已广泛应用于办公自动化中的著名的ARC-NET局部网络经过简化，使之适用于IBM PC的微机局部网。系列产品目前有三档：PLAN2000，PLAN3000

和PLAN4000。它们均采用分布式星形结构（簇形），网络结构简单、灵活，成本低。

网络功能中提供了网上各微机间的直接通信；允许网络中所有系统共享打印机和硬盘驱动器（可以是XT的硬盘）；并提供了电子邮政程序Eagle Mail，其特点是报文可分布于多个站点中。

由于在访问控制方式中采用了令牌传递方式，既避免了访问冲突，又有较好的确定性。PLAN4000局部网的主要参数如下：

网络结构：簇形（总线式，可任意分枝，但不带闭合回路）

通信访问方式：Token Passing

调制方式：基带

传输介质：同轴电缆

数据传输速率：2.5Mbps

最长距离：600米

可连接节点数：255个

PLAN系列局部网的组成部件有插在IBM PC或PC/XT机内的网络接口板、磁盘服务器、同轴电缆，以及用于连接、转换同轴电缆的线绝缘设备LID等。LID分有源和无源两类。无源LID有4个口，连接长度为10米；有源LID有6个口，连接长度达600米。网络的拓扑结构为可以任意分枝的总线式，这对用户十分方便。图6-12是PLAN4000的结构示意图。

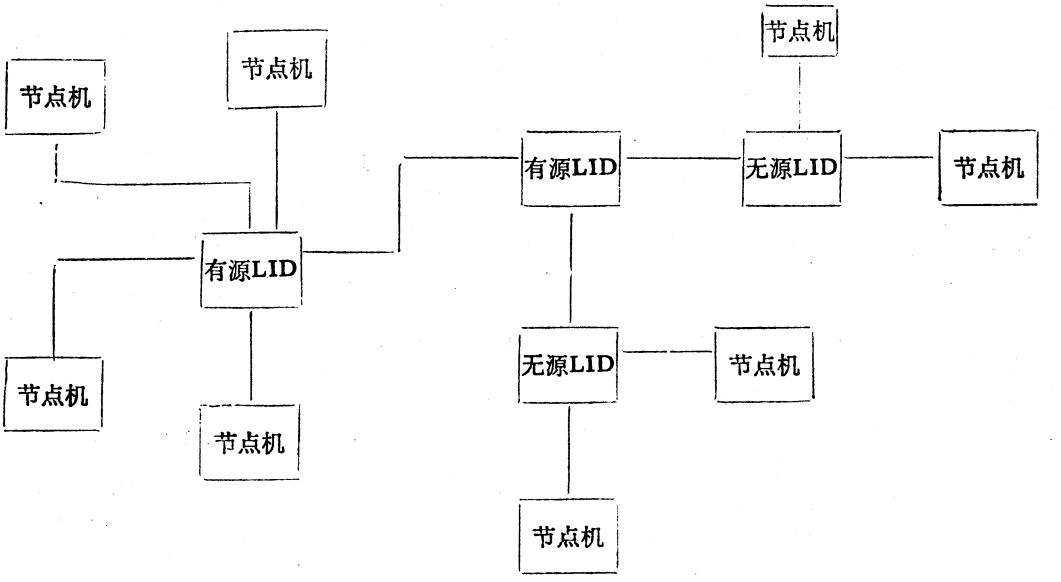


图6-12 PLAN4000局部网的构成

（5）ETA-NET局部网

这是我国自行开发的以IBM PC和PC/XT为节点机的局部网络，它吸收了“以太”网

和“OMNINET”网的长处，具有结构简单，使用方便和成本低廉的优点。

ETA-NET的网络操作系统PC-RDMN是在MS-DOS的基础上开发而成的，它扩充了一系列网络操作功能，又保持了对MS-DOS的兼容性。所提供的主要功能有：

* 实时文件传送。
* 实时的站间对话。
* 查看网络资源。
* 查看指定站点的文件目录。
* 会议功能。
* 汉字处理功能。

网络的主要参数如下：

网络结构：总线方式
访问控制方式：CSMA/CD
调制方式：基带
传输介质：双绞线
数据传输速率：1 Mbps
最长站间距离：1 KM
可连接节点数：128

ETA-NET网络传输器由通信处理机、接口及双绞线所组成。其中，通信处理机的主控部分由6809单片机来担任，它实现用户收发数据的打包、拆包、地址识别、状态控制以及与高层软件的接口等功能；节点机与传输线路的接口部分用来实现串/并转换、CRC代码的生成与校验、线路监听、冲突检测及信息同步、信号变换等功能，实现了网络通信协议中低层部分的有关规定。

ETA-NET网价格低于“以太”网和Omninet网，又是我国自行设计开发的产品，随着配套软件的不断丰富，该网络具有很好的发展前景。

除了上述各种局部网之外，IBM公司自行开发的局部网产品正在不断推出，例如IBM PC网等。有关介绍请参见本章第六节。

第二节 IBM PC/OMNINET局部网的结构原理

OMNINET局部网是美国CORVUS公司开发的微型机通用局部网。它的技术比较成熟，具有一定的先进性，同时价格低廉，通用性强，软、硬件设备又比较紧凑灵活，因此具有一定的特色，目前已广泛地应用于教学和小型事务处理等领域中。

OMNINET局部网已为多种微型机所采用，如APPLE Ⅱ，TRS-80，PC 8001，CROMEMCO，DEC LSI-11，IBM PC等。下面对以IBM PC为节点机的OMNINET网络的硬件结构原理进行简要介绍。

1. PC/OMNINET局部网的基本配置

以IBM PC机为节点的OMNINET局部网络的基本配置如图6-13所示，其中包括下列主要组成部分：

① 网络传输器（TRANSPORTER） 每一台连接在网络上的IBM PC机，必须配备一个传输器，插入机箱中扩充槽内。用于IBM PC机的传输器型号为OMNI-IB。

② 硬盘服务器（OMNIDRIVE） 这是连接在网络上为所有用户共享的硬盘服务器及硬盘子系统，产品有5.5，11.1，16.6，45.1MB等多种。

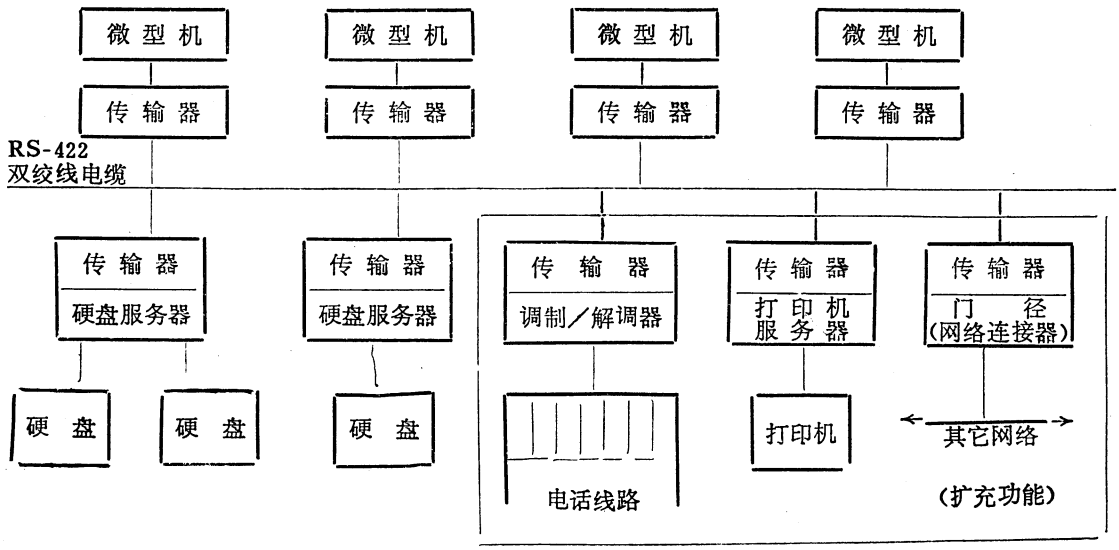


图6-13 OMNINET网的配置

③ 连接电缆（OMNINET CABLE） 用作网络干线，系带塑料套管的双绞线。

④ 电缆连接匣（TAP BOX PACK） 用以把各种网络节点（包括IBM PC机）连接到网络总线电缆上，节点间最大距离为300米，从设备到干线最大距离为15米。

⑤ 有源转接匣（ACTIVE JUNCTION BOX） 用来延长节点间传输距离，可使节点间距扩大到1.2公里。

⑥ 打印服务器（PRINTER SERVER） 用作网络共享打印机的控制器。

⑦ 终端器（TAP TERMINATOR） 装在网络干线两端，用于传输信息的匹配。

⑧ 网络管理软件（CONSTELLATION Ⅱ） 用以进行网络资源共享、数据通信的管理软件。为IBM PC配置的CONSTELLATION Ⅱ的版本分为C2M-IB-D_1（DOS1.1版）和C2M-IB-D2（DOS 2.0版）两种。

除上述基本组成部分之外，用以构成OMNINET网络扩充系统的硬件还有：

* OMNINET BANK系统 这是一种盒式磁带机，用作硬盘信息的后备装置。每盘带的存贮容量为100MB或200MB。

* NETWORK MIRROR系统 使用录像带作为存贮介质的磁带机，也用作磁盘

信息后备装置，每盘容量约20—60MB，是一种低价的后备存贮器。

* 门径服务器（CORVUS GATEWAY） 用以使OMNINET网络连接到其它局部网络或远程网络。

当然，上述扩充部件还必须配用相应的软件，才能在网络上正常工作。

2. OMNINET传输器的结构

OMNINET局部网的硬件核心部件是传输器，它完成OSI七层网络通信协议模型中的低四层功能，即从物理层到传输层的功能，其作用相当于网络中的一台前端通信处理机，因而大大减轻了工作站主机的负担。

传输器本身也是一个微型机，其逻辑框图如图6-14所示。各组成部分的功能简述如下：

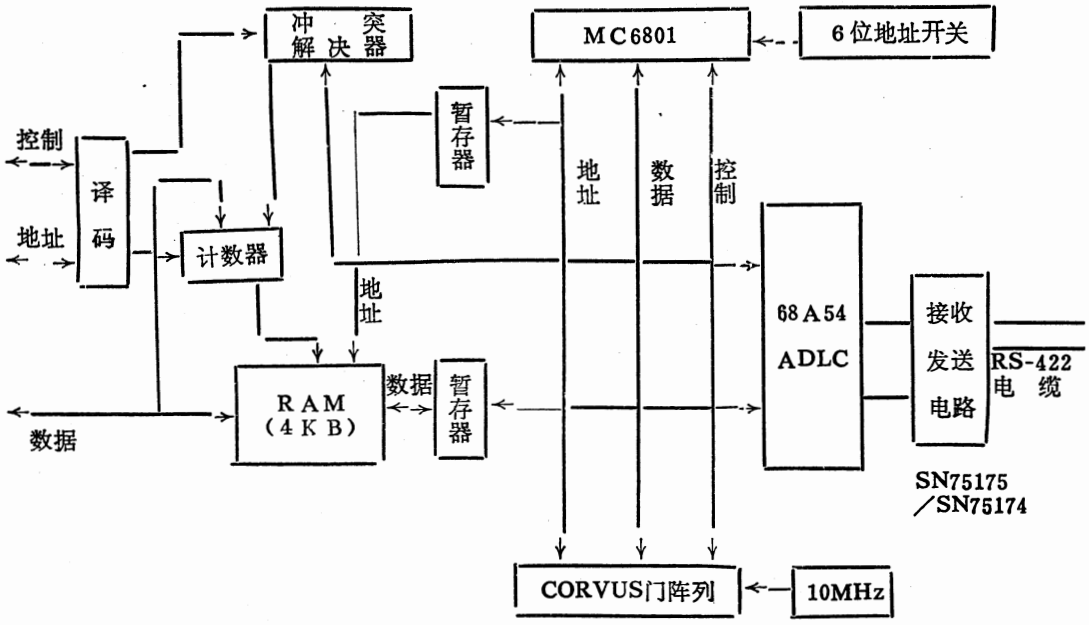


图6-14 传输器的逻辑结构

MC6801是一个8位的单片机，它的指令系统与MC6800兼容，并增加了17条指令。芯片本身带有2KB的ROM和128B的RAM，4个可编程的输入输出端口，1个定时器和1个计数器，以及32个内部寄存器，许多控制程序就固化在ROM中。MC6801的任务是负责控制网络节点机与传输器，以及传输器与网络传输介质之间的数据和控制信息的传送。

不同型号的节点主机需要使用不同的传输器。为了使传输器能适用于不同的节点机，除了与主机接口的线路有所不同之外，可以把MC6801改成MC68701，这样芯片内部的2KB ROM就改为2KB EPROM，其中的软件可以重新写入。网络通信部件由MC68A54 ADLC芯

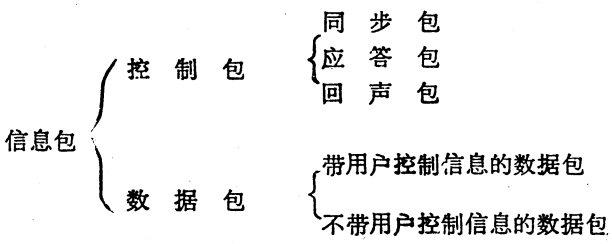
片来担任。MC68A54是一种高级数据链路传送控制器，它能完成网络数据链路层的许多功能。例如，信息帧格式的装配和拆卸，循环冗余码CRC的生成及校验，数据的并串变换，数据缓冲（三字节先进先出方式），信号的NRZI（反相不归零制）调制等。在发送和接收信息时，为了达到每秒1兆位的传输速度，传输器使用了专门的门阵列芯片作为快速DMA和各种信息交换的控制电路。在MC6801的管理下，门阵列芯片产生MC6801，4 KB RAM及MC68A54与网络收发器等部件之间信息传输的各种控制信息，完成通信命令执行过程中的各种信息交换的控制。

4 KB 的RAM是一个缓冲存贮器，在传输器中起着十分重要的作用。一方面，节点主机送出一个地址指针之后，可以通过I/O指令读写其中的内容；另一方面，在传输器进行通信操作的过程中，MC6801和ADLC也可以读写其中的信息。后者在门阵列控制下以高速的DMA方式进行读写，以确保1 Mbps的通信传输速率。

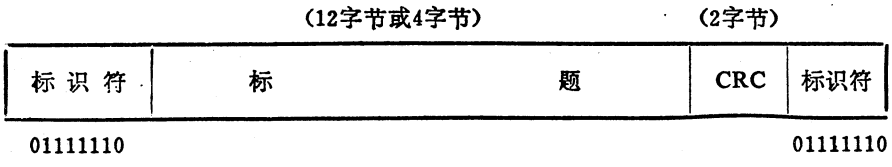
驱动/接收电路使用了差分驱动器（SN75174）和差分接收器（SN75175），用以驱动按RS-422通信规程动作的双绞线电缆。这些电路通过滞后作用具有很强的噪音抑制能力，由于平衡差分信号在其共模电压中高达24V的电压差，故不需要隔离变压器或光电隔离器，只需采用普通的100Ω阻抗的双绞线即可。收发器的开启与关闭，传输信号的同步及时钟采样等也都是门阵列直接控制的。

3. 信息包格式

OMNINET网络中传输的信息包共有如下五种：



控制包用于网络上的控制及查询，它们中不包含用户的数据信息。其中同步包是某个节点执行系统复位操作或初始化时向其它节点发出的广播信息包，它不要求任何回答信息，仅用于网络系统同步的目的。应答包是目标节点正确地收到一个信息包之后，向源节点发回的一个用作确认的信息包。回声包是一个节点为了查询另一节点是否连网而发出的一个查询包，若取得对方的应答包，则可以肯定被查询节点已经连接在网络上。所有这三种控制包都比较简短，它们的格式为：

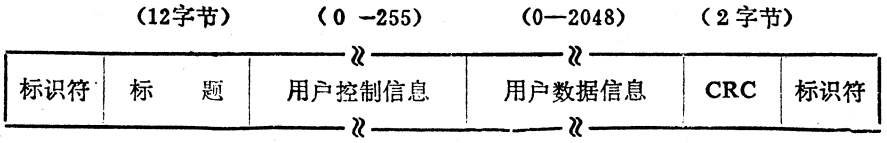


其中标题信息用于区分不同的包，它们的内容见表6-5所示。

表6-5　OMNINET信息包的标题字段的格式

字节序号	信息包类别 / 字节意义	不带控制信息的数据包	带控制信息的数据包	同步包	应答包	回声包
1	目标地址	目标地址	目标地址	60	50	目标地址
2	源地址	源地址	源地址	源地址	返回码	源地址
3	有效码	A5	A5	A5	A5	A5
4	套接字	B0 或 90 A0 或 80	A0或B0	00	24	00
5	重发次数	重发次数	重发次数	00		重发次数
6	奇偶校验位	80 或 81 80 或 01	90 或 81 80 或 01	01		00或01
7	数据长度（高8位）	F0	F0	FF		FF
8	数据长度（低8位）	00	00	随机数		随机数
9	控制信息长度	00	00	00		00
10	数据长度（高8位）	00	F0	FF		FF
11	数据长度（低8位）	0X	XX	随机数		随机数
12	控制信息长度	00	00	00		00

数据包除了标题之外，还包括被传输的数据和控制信息。控制信息不能超过256字节，数据长度最大可达2048字节。把控制信息与数据信息分开的目的是为了更便于高层软件的编制，它们传输到目标节点之后被放入存贮器不同的区域之中，数据包的格式如下：



其中，控制信息的头尾各有7个和6个冗余字节，数据信息开始处有10个冗余字节，它们用来缓和对接收方传输器中MC6801的速度要求。

数据包的标题信息中除包含源节点和目标节点的地址之外，它还给出了控制信息长度、数据长度、出错重发次数及套接字（Socket）等信息。每个传输器有四个套接字，编号分别为80H，90H，A0H，B0H，它们各自与节点机的四个内存区域相对应。有控制信息的数据包只能使用A0H和B0H两个套接字。各种信息包的标题内容如表6-5所示。

4．CSMA/CA访问控制方式的实现

OMNINET网采用的访问控制方式为CSMA/CA方式，它利用软件和门阵列硬件来避免访问的冲突，其基本方法是采用下述的两次检测技术。

第一次为准备性检测。要发送信息包的传输器，当ADLC检测到传输总线已处于空闲状态时(连续8个以上的“1”信号)，即通过中断告诉MC6801把网络状态空闲位置“1”。信息包发送前若检测到空闲位为“1”则执行“准备发送”程序，但从启动MC6801执行该

程序到向ADLC发出命令，真正将数据送到传输总线上还需化费大约40微秒时间。这段时间内，由门阵列高速硬件对传送总线的状态作第二次检测，如果总线上出现300ns以上的“0”信号，则监听触发器清“0”（表示总线已占用），若总线仍保持空闲，则打开发送器电路进行发送。如果监听触发器为“0”，则表示总线不空，此时，就由门阵列封锁发送器，停止数据发送，并按照延迟算法等待一段随机时间后，再重新发送，这样即可解决冲突。当然，若在临近发送前的300毫微秒之内总线被其它节点占用，则仍会发生访问冲突，但这种情况发生的概率极小。图6-15是OMNINET网CSMA/CA访问控制方式的工作流程图。

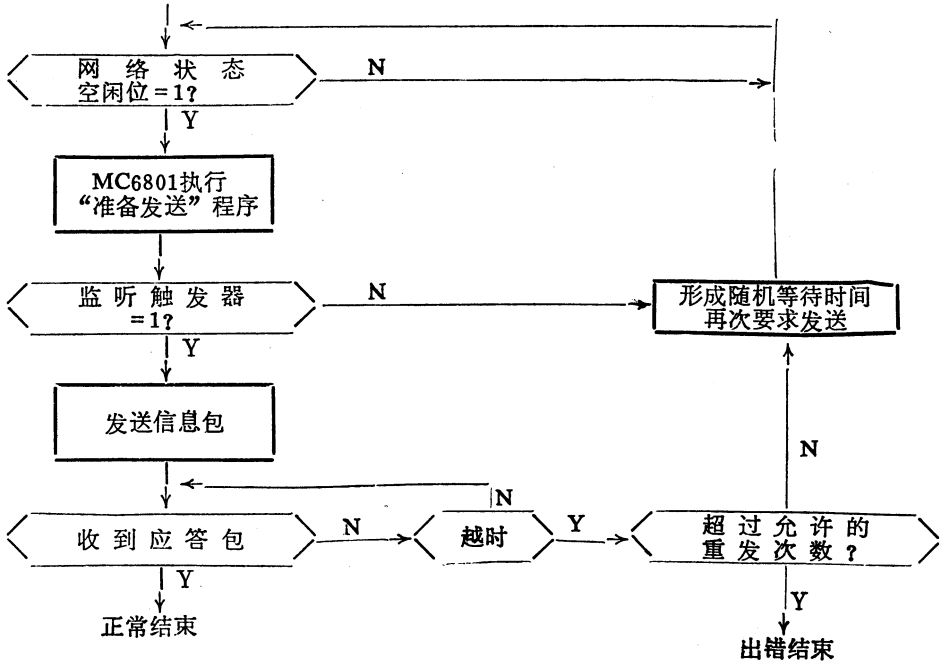


图6-15 CSMA/CA工作流程图

信息包被发送到传输总线上之后，其它节点对此信息包报头中目标地址进行检验。若与本节点传输器地址一致，则将后续的数据（从0—2K字节可变长）取入本节点，并对所接收的数据进行CRC校验，确认无误后，便立即向发信节点发出确认信息包（ACK包），通知送信节点数据已被正确接收。如果由于访问冲突及线路干扰使信息包内容受到破坏，发送节点接收不到确认信息包，则经过一定时间，发送节点会再次发送信息包，直至收到ACK包后通信过程即告结束。

5．通信命令及传输器程序设计

从程序设计的角度来看，传输器是节点机的外部设备之一。节点机通过输入输出指令可

以启动传输器完成各种网络控制和信息传输操作。其过程大体如下：

① 节点机首先在4KB缓存中构成一个命令控制块（CCB），然后节点机把CCB的首地址（8个字节）用输出命令发送到传输器的相应端口。

② 传输器收到CCB首地址后，对缓存中CCB内容进行分析并执行。

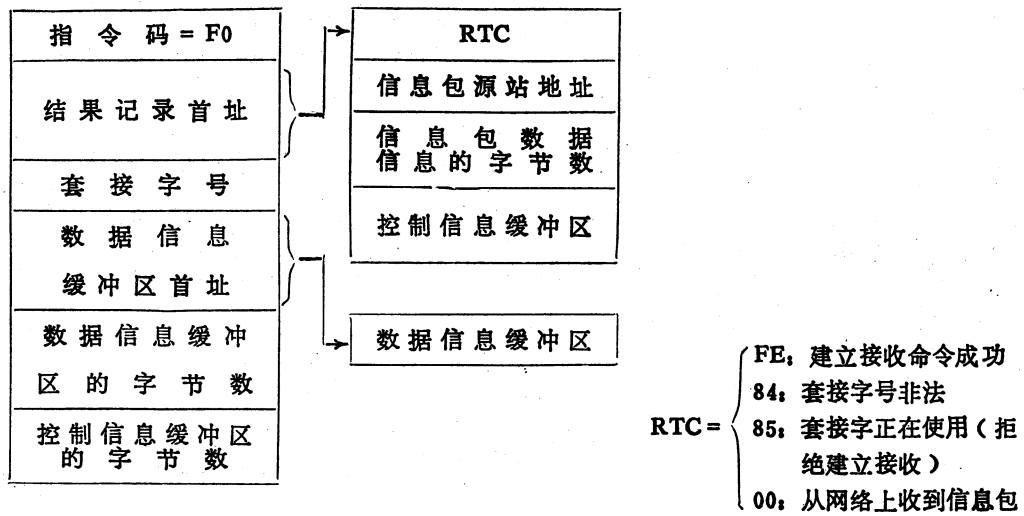
③ 命令执行后，传输器将返回码（RTC）送回到CCB所指定的结果记录缓冲区。

④ 节点机对返回信息进行检查处理。

OMNINET网传输器可以接收来自主机的通信命令共有七种，下面简单介绍它们的含义及格式。

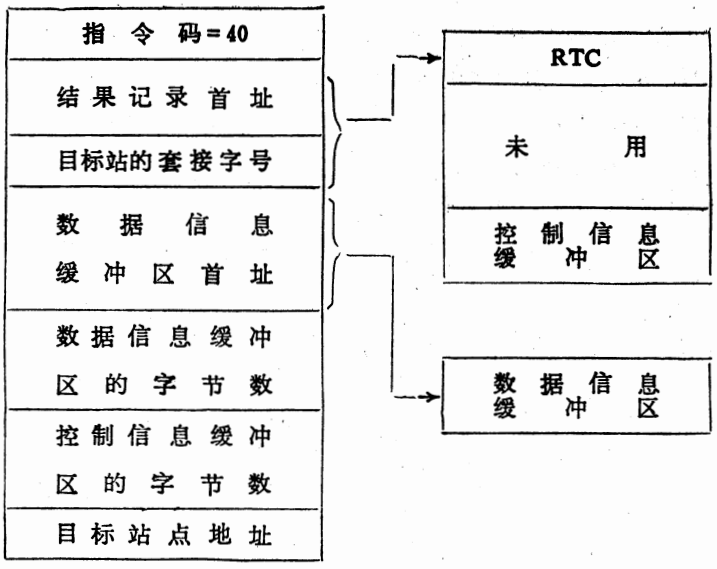
（1）建立接收命令（SET UP RECEIVE）

将传输器设置成接收信息状态，传输器应按指定的套接字准备接收信息包。若套接字为80H或90H时，只接收一个数据包（数据）；若为A0H或B0H时，除了数据外还有相应长度的控制信息需要接收。传输器执行这条命令产生两次回答：第一次指出该命令已被执行，套接字已准备好（或有错）；第二次指出已从网络上收到信息包（RTC=00）。



（2）发送命令（SEND DATA）

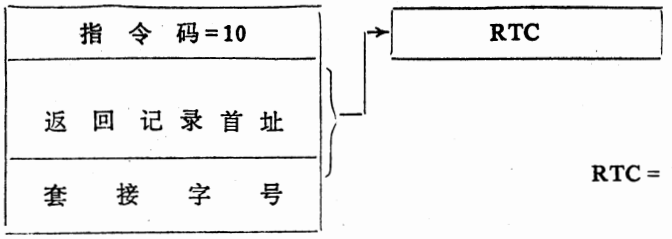
命令传输器发送一个信息包到目标节点机的相应套接字所指出的内存缓冲区中，若为广播式发送，则目标站地址为FF。命令要求在CCB中给出要发送的数据缓冲区的起始地址及长度。如果有控制信息，还需要指出控制信息长度。控制信息放在返回记录缓冲区的第5个字节开始的区域。



RTC =
- 00：一次发送成功
- nn：发送nn次后成功（nn<10）
- 80：超过最大重发次数
- 81：接收方的数据信息缓冲区溢出
- 82：接收方的套接字未建立接收
- 83：收发双方的控制信息缓冲区容量不符
- 84：套接字号非法
- 86：目标站地址非法

（3）结束接收命令（END RECEIVE）

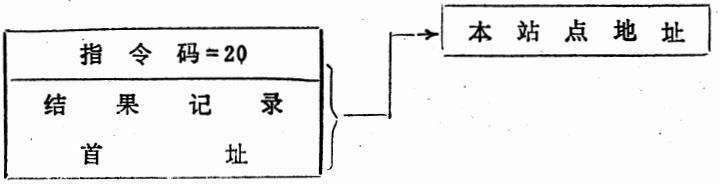
解除传输器的接收状态，将指定的套接字从建立接收状态释放成自由状态。



RTC =
- 00：结束接收成功
- 84：套接字号非法

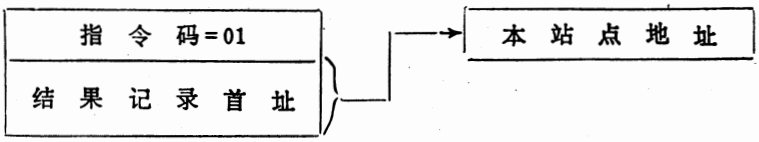
（4）传输器初始化命令（INITIALIZE TRANSPORTER）

使传输器复位为初始状态。



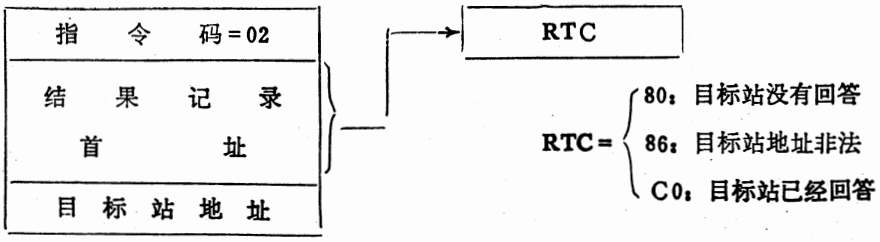
（5）查询站号命令（WHO AM I）

读出节点机中传输器的地址，即传输器板上地址开关的设定值。



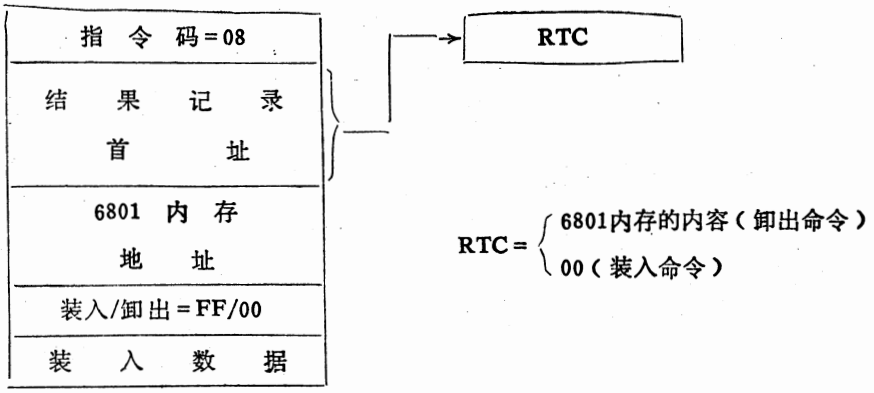
（6）回声命令（ECHO）

命令传输器向指定的目标节点发送一个回声包，查询对方是否连接在网络上。



（7）装入/卸出命令（PEEK POKE）

用以变更MC6801内部控制程序的参数（位于MC6801的128B的随机存贮器内）。例如，00E1H是信息包的最大重发次数，00E2H是发送信息包后的空隙长度，00E3H是重发信息包的随机等待常数的初值等。



传输器的程序设计是所在节点机的输入输出指令通过四个双向的输入输出端口进行的，这些端口的名称及其功能如表6-6所示，其中输入端口仅使用三个，有一个未用。节点机启动传输器执行一条通信命令的过程大体如图6-16所示。

表6-6　传输器与节点机硬件端口的接口

地　址	端　口　名	功　　能
24B	RD-INC	读传输器4KB缓存中的数据，缓存地址自动增1
24A	RD-DATA	读传输器4KB缓存中的数据
248	RD-STAT	读传输器状态（Bit7＝1表示传输器就绪）
24B	WR-INC	向传输器缓存写数据，缓存地址自动增1
24A	WR-LO	向传输器输出缓存地址码的低字节
248	WR-HI	向传输器输出缓存地址码的高字节（Bit_4＝1表示允许传输器发出中断请求）
249	WR-STR	向传输器输出命令块首址的一个字节

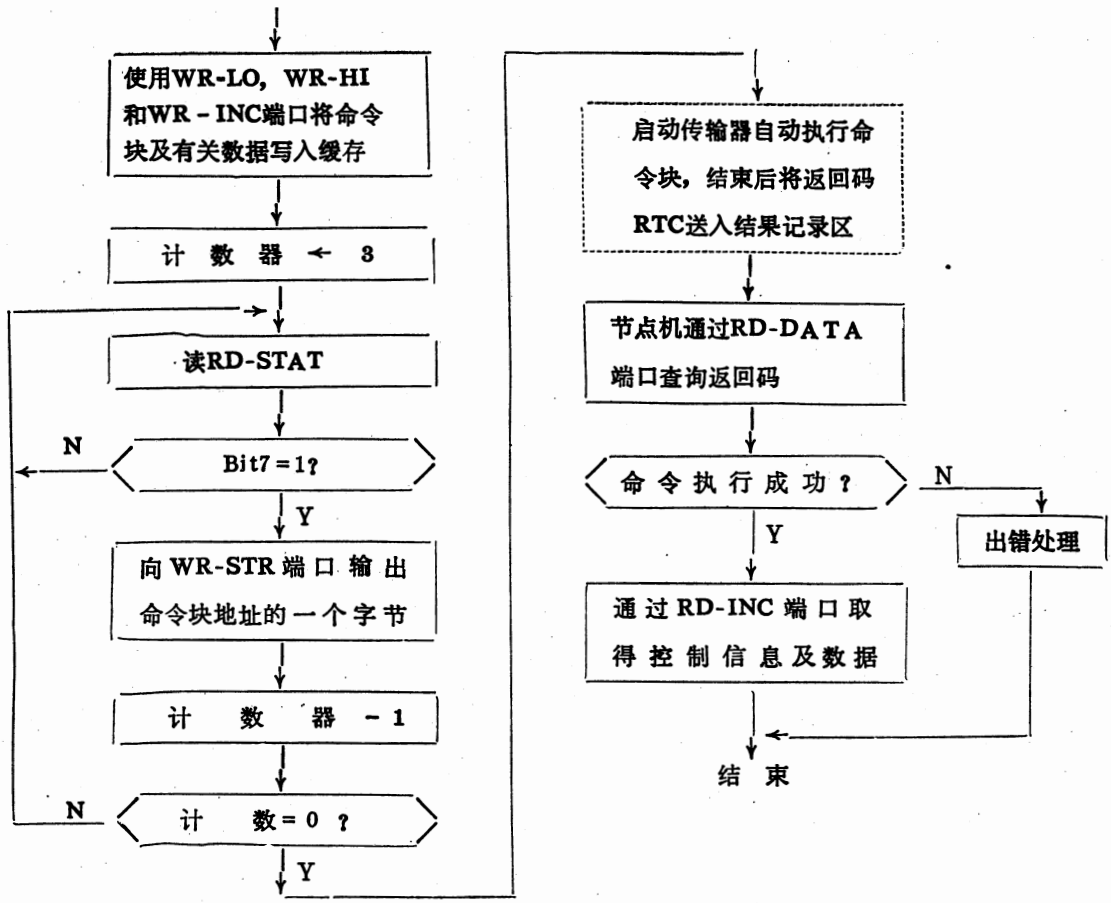


图6-16　传输器通信命令的执行过程

OMNINET网络上的各个节点可以相互进行通信，通信是通过收发双方使用相同的套接字进行的。每个节点的传输器均可使用四个套接字，它们分别是MC6801单片机中128字节的RAM（地址为0080—00FF）中的16个字节。例如，0080—008F是套接字80，0090—009F是套接字90等等。它们用来存放接收信息时的有关参数。图6-17是A、B、C三个节点交互通信（A收到B、C送来的信件后，分别转发给C和B）时的原理示意。

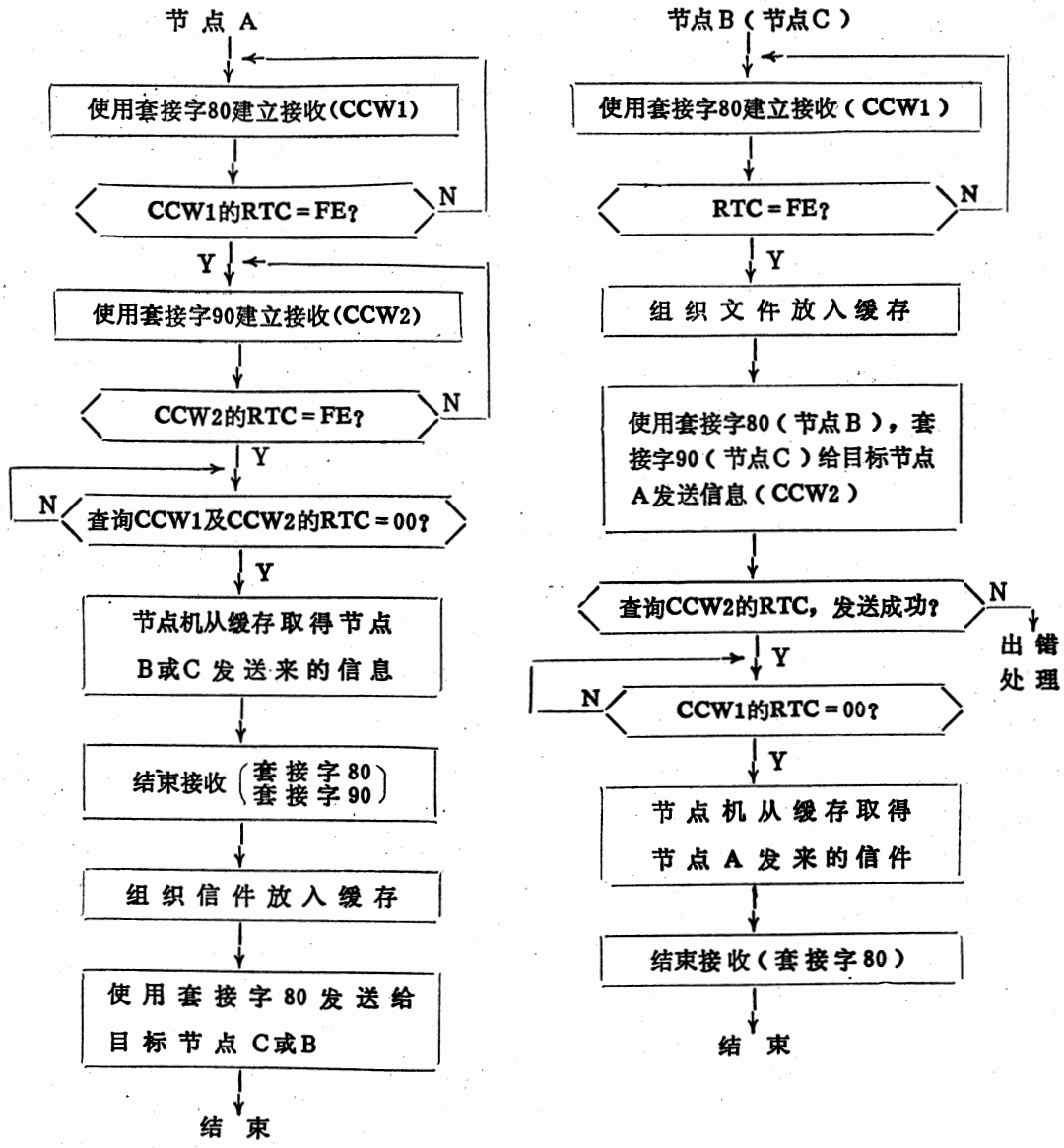


图6-17 节点A、B、C相互通信的过程

6. 硬盘服务器

硬盘服务器是连接在网络上供所有节点计算机共享的磁盘存贮器及其有关控制器，它由两部分所组成：

（1）具有智能的硬盘子系统

硬盘子系统中硬盘的容量最大可达45MB或80MB。它的控制器是由Z80微处理器组成的微机系统，加电启动后具有自举和诊断的能力，能够从硬盘上读出磁盘固件程序。除了具有通常的扇区读/写、格式化等功能之外，它还具有一些特殊的网络操作功能，如管道操作，信号量操作等等。

（2）硬盘服务器

硬盘服务器又有两个组成部分：用于连接网络的传输器以及用于和硬盘子系统连接的接口部分。后者包括常驻的管理程序，用来实现各节点工作站访问硬盘子系统的管理。硬盘服务器根据网络上节点机发来的命令，转达给硬盘子系统，并把回答信息发送到正确的地点。它能够接收节点机发来的各种服务请求，并进行登记，按一定的优先次序处理这些请求。

硬盘服务器和网络上节点机的通信过程大体如下：节点机通过CCW启动本站的传输器发送包含有磁盘操作命令在内的信息包给服务器，其套接字为B0。然后服务器向节点机发一个回答信息，节点机的套接字也是B0。由于磁盘服务器缓冲容量很小，如果磁盘操作命令超过四个字节，则节点机需分成两次发送。第一次发送给服务器的套接字B0，等待服务器回答信息“GO”之后，再把磁盘操作命令的第二部分由传输器发送给服务器的A0套接字。这种情况称为长磁盘命令。磁盘服务器的套接字80用来接收广播信息。网络上某节点以广播地址（FFH）发送信息时，可以通过服务器的回答信息来了解磁盘服务器中传输器的物理地址。

图6-18是节点机向磁盘服务器发出长磁盘命令时的四个步骤以及每次所使用的CCW块的格式，供读者参考。如果节点机向服务器发出的磁盘操作命令是短命令（不超过4个字节），则只要两步即可。第1步是建立接收（格式如图6-18第3步），第2步是发送磁盘操作命令（格式如图6-18第2步，但数据长度应填入磁盘操作命令实际的字节数）。关于磁盘操作命令的种类、格式及含义请参见有关技术资料，这里不再细述。

7. 传输器驱动程序

前面已经讲过，每一个节点工作站中的传输器，从程序设计的角度看来，它也是节点机的一个外围设备。IBM PC机的常用外围设备的驱动程序，一般均固化在PC机底板上的ROM存贮器中，称为BIOS。传输器的驱动程序则固化在传输器板上的EPROM中，共2KB，它们在PC机存贮空间中占用DF00：0000—DF00：07FF。

传输器驱动程序共分成四个模块，每个模块可以由驱动程序首部的转移表进入。转移表中的四条转移指令分别用来转向不同的模块，四条转移指令的位置是：DF00：0000，DF00：0003，DF00：0006和DF00：0009。

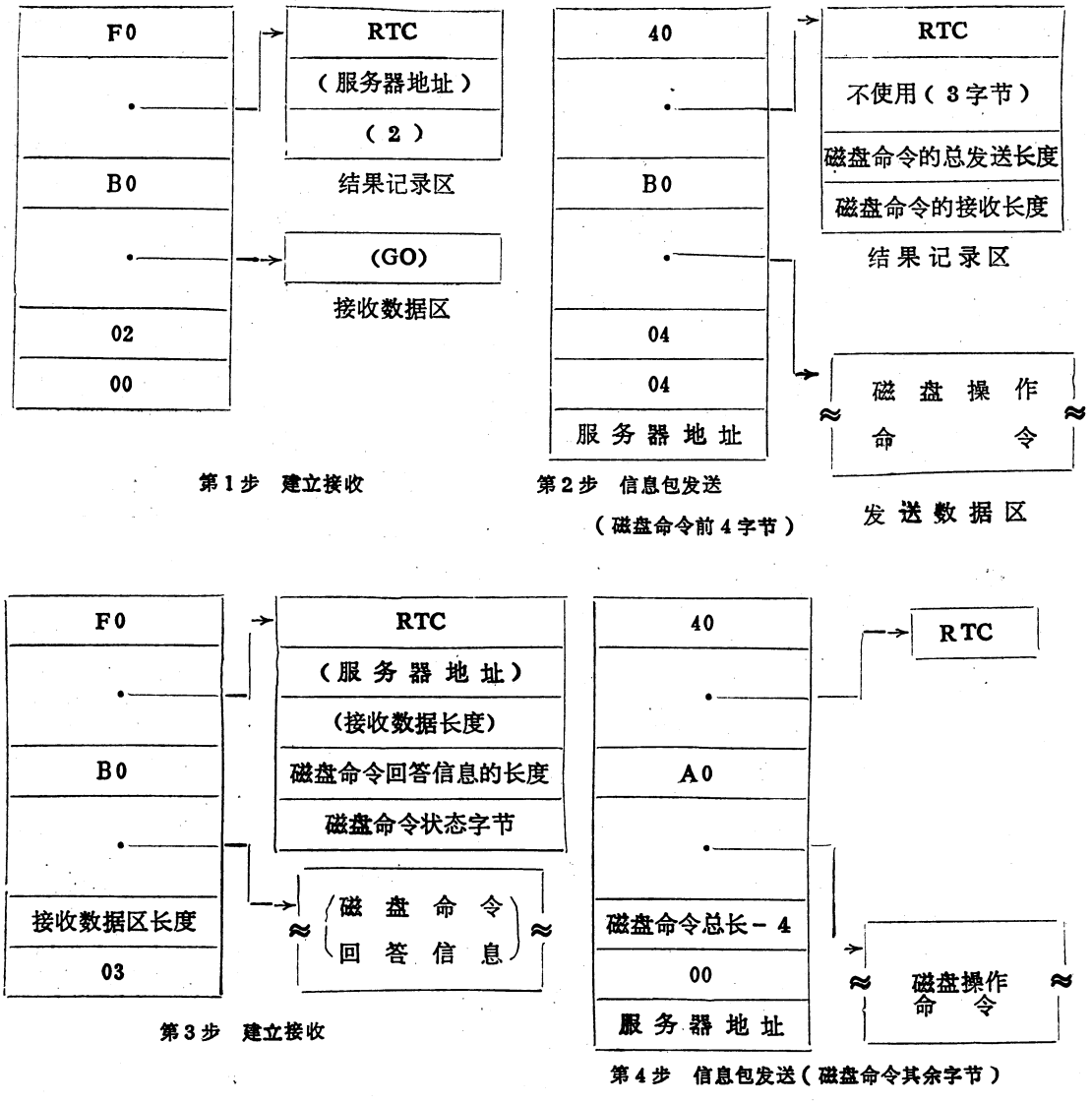


图6-18 节点机向磁盘服务器发送长磁盘命令的过程

驱动程序中第1个模块的功能是：确定本传输器地址在网上是否唯一；测定本节点主机的内存容量；取得网络上硬盘服务器的地址，然后向服务器发出一条短磁盘命令（引导命令），从硬盘中读出5.5KB的引导程序装入所在节点机的0000：7C00区域中并转向执行。

第2个模块的功能是在传输器的4 KB缓存中形成12个CCW块。由于在网络操作的过程中，各节点之间的通信都需要使用特定格式的CCW，并启动MC6801解释执行，如果每次都由节点主机进行CCW的生成，则势必增大程序空间而又很不方便。因此，2 KB驱动程序中已固化有12个CCW命令块(共95个字节)。本模块是将它们一次传送到传输器的4 KB缓存中的最前面95个字节单元之中，4 KB缓存从0095H字节开始的区域用作数据传送区，其中可以存放长磁盘/短磁盘命令或其它命令，也可以存放读写硬盘的数据。

第3个模块的启动地址是DF00：0006，它是驱动程序中最主要的组成部分。这个模块由六个功能子程序组成，分别用来完成不同的通信命令。不同的功能子程序由AH寄存器内容加以区别，高层软件调用时统一使用指令：

CALL DF00：0006

表6-7列出了六个功能子程序各自的功能以及它们的入口参数和出口参数，供读者编

表6-7　传输器驱动程序中的功能子程序

编　号	功　　能	入　口　参　数	出　口　参　数
0*	确定本传输器地址在网中是否唯一	（AH）=00	（AH）=本传输器地址 （AL）=00 地址唯一 FF 地址不唯一
1*	发送短磁盘命令	（AH）=01 （DS：SI）=磁盘命令缓冲区首址 （ES：DI）=长磁盘命令的第5字节首址，也是命令执行后返回数据的缓冲区首址 （CX）=磁盘命令长度（长磁盘命令的前半部分长度） （DX）=长磁盘命令后半部分长度，短磁盘命令读操作时，为要读取的字节数 （AL）=磁盘服务器地址 （BL）=时间常数 （BH）=重发次数	(AL)=｛00 成功 FF 不成功 （CX）=接收到的数据长度
2*	套接字80发送信息包	（AH）=02 （DS：SI）=发送内存缓冲区首址 （CX）=发送长度 （AL）=目标节点的地址 （BH）=重发次数 （BL）=时间常数	(AL)=｛00 成功 FF 不成功
3*	用套接字80接收并发送信息包	（AH）=03 （ES：DI）=接收信息缓冲区首址 （DS：SI）=发送信息缓冲区首址 （CX）=发送信息长度 （DX）=要接收的信息长度 （BH）=重发次数 （BL）=时间常数	(AL)=｛00 成功 FF 不成功 （CX）=实际收到信息的长度
4*	广播发送分析包查服务器地址	（AH）=04	
5*	发送长磁盘命令	（AH）=05 其它参数均同1*子程序	(AL)=｛00 成功 FF 不成功

制高层网络软件时参考。这六个子程序执行时，首先根据要求设置或修改4 KB缓存中CCB的有关字节，然后将对应CCB的首地址分三次输出给MC6801，MC6801收到CCB地址后，就能从缓存中取出CCB自动解释执行而完成规定的网络通信操作。

最后一个模块只有一条中断返回指令，在网络操作系统中，作为处理软中断1BH和1CH之用。

第三节　IBM PC/OMNINET的网络管理软件CONSTELLATION Ⅱ

1. 概述

Constellation Ⅱ是美国CORVUS公司为OMNINET局部网开发的以共享资源为目标的一个网络管理软件。它允许网络各节点共享网络资源（硬盘、打印机等）并进行相互通信，为用户在网络上开发各种应用程序提供了简便的手段。

Constellation Ⅱ可以支持多种操作系统和多种微型机的工作，它允许不同操作系统之间进行文件传递。Constellation Ⅱ以软件包的形式提供，它与节点机操作系统相结合就构成了网络操作系统，为IBM PC节点机的MS-DOS操作系统扩充了网络通信接口，增加了网络管理程序和网络实用程序。

Constellation Ⅱ与MS-DOS之间的层次关系如图6-19所示。

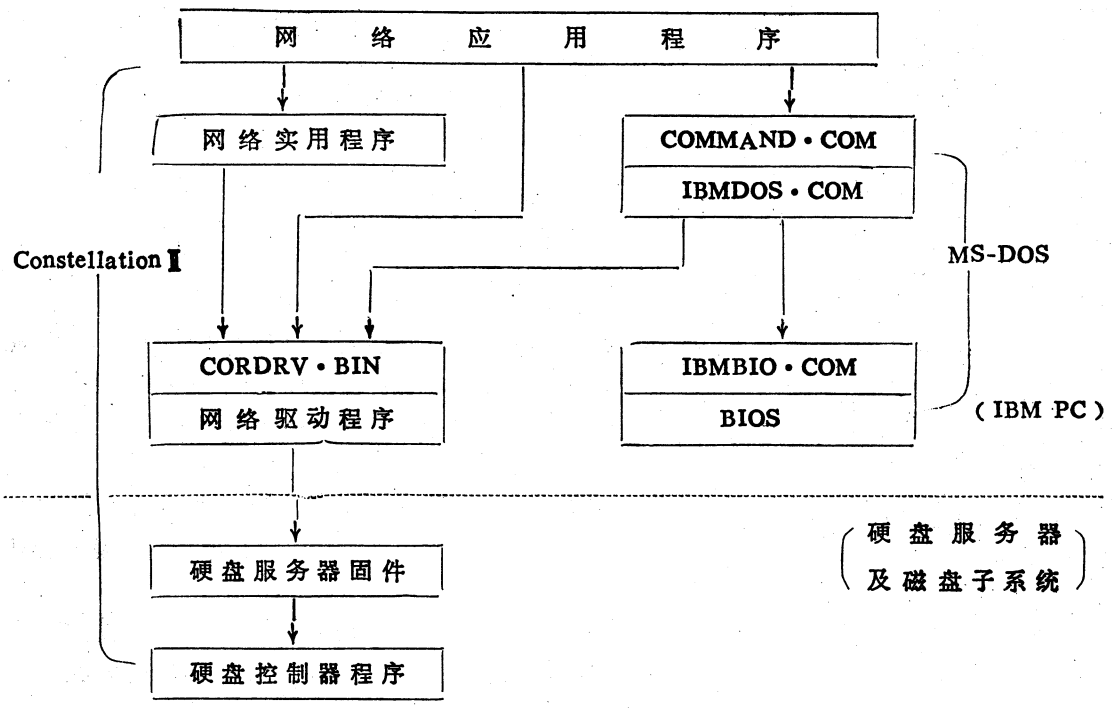


图6-19　Constellation Ⅱ与MS-DOS的关系

其中，网络驱动程序分布在各个节点机传输器插件板上，它是IBM PC与网络的接口软件。对操作系统MS-DOS而言，相当于在BIOS这一层上扩充了控制网络硬件的功能模块。CORDRV.BIN是传输器管理程序，它是一个接口模块，为高层软件调用网络驱动程序提供了方便。

图6-20是网络工作过程的示意图。当用户的应用软件中出现对网络上共享硬盘的访问要求时，MS-DOS进行解释后调用网络驱动程序中有关的功能子程序，通过传输器发出内容为磁盘操作命令的信息包给磁盘服务器，从而实现对共享硬盘的访问。同样，应用程序中要求进行工作站之间的通信时，也是由网络驱动程序通过收/发信息包而实现的。

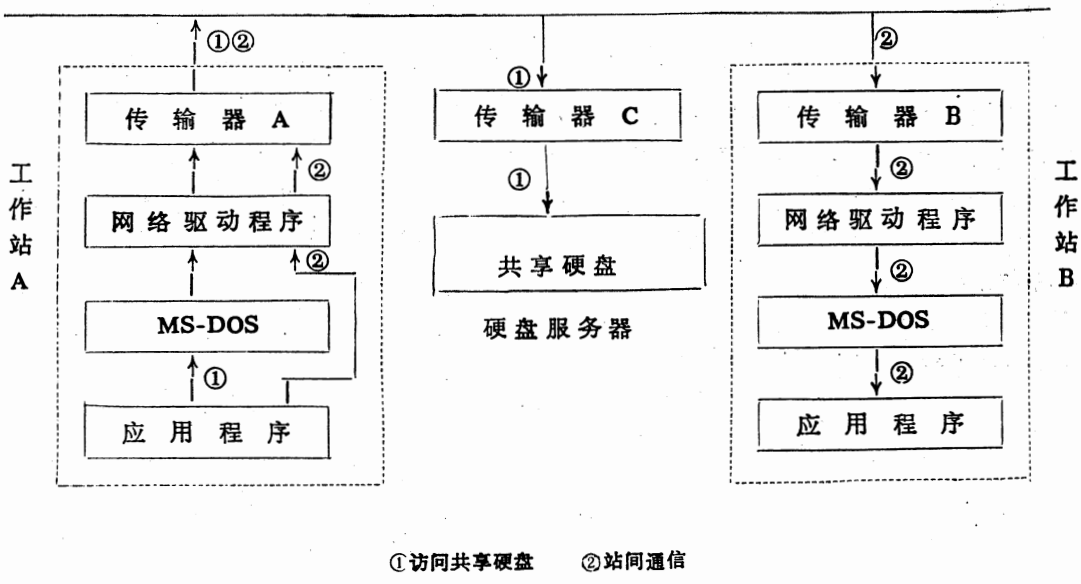


图6-20 网络驱动程序的工作示意图

在Constellation Ⅱ的管理下，网络上的每个节点机从功能上来讲都是一样的。网络资源的共享和节点间的通信都通过“帐户”（User account）来管理。每个帐户由用户名和口令两部分组成。只有经过系统管理员“开户”后的帐户才能使用网络资源和利用网络进行通信，否则不能使用网络资源。

Constellation Ⅱ把网络的共享硬盘划分为许多盘卷（Volume）。一个盘卷可以看作是一个虚拟软盘，盘卷的大小允许互不相同。对于网络上的用户来说，可以象使用软盘一样任意访问共享硬盘中的有关盘卷，每个用户最多可以同时使用10个盘卷。

Constellation Ⅱ的主要功能包括系统管理、系统生成、系统维护和一般用户功能等（图6-21）。从结构上来说，其中的绝大多数程序模块均存放在共享硬盘中，它们只是在执行相应操作时才从共享硬盘中读出装入到工作站节点中运行。

从ISO计算机网络的七层参考模型来看，OMNINET的传输器实现了网络模型中的低四层，而Constellation Ⅱ网络管理软件、网络驱动程序等则与模型中的高三层相对应。图

6-22是按照ISO七层模型对OMNINET局部网的功能所作的粗略划分。

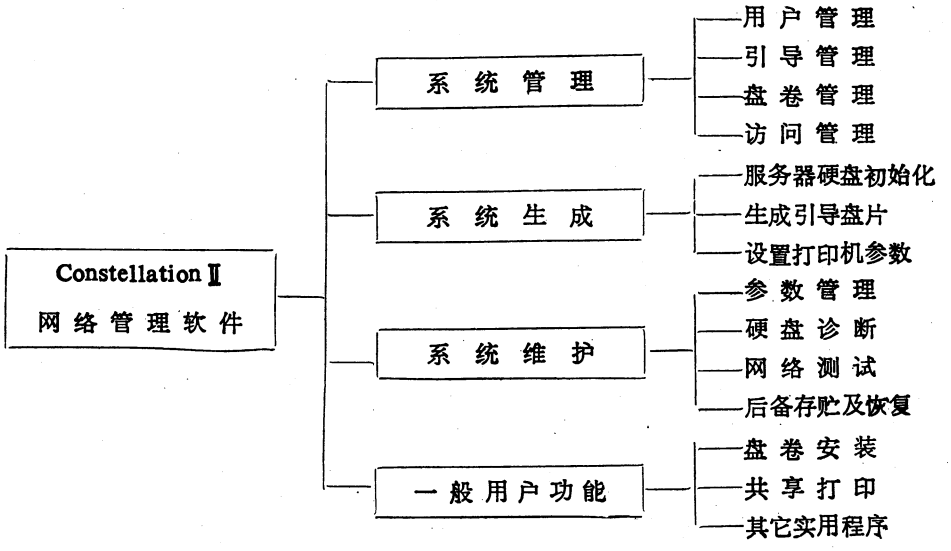


图6-21 Constellation Ⅱ的功能和结构

应用层：	Constellation Ⅱ：一般用户功能，系统维护，电子邮件等
表示层：	Constellation Ⅱ
会话层：	Constellation Ⅱ（网络驱动程序）
传输层：	传输器
网络层：	传输器
链路层：	传输器
物理层：	RS-422 双绞线

图6-22 OMNINET局部网的ISO七层模型

2．盘卷和用户

（1）共享硬盘的存贮分配

在Constellation Ⅱ管理下的网络共享硬盘的存贮空间分配情况如图6-23所示。硬盘不管总容量多少，均以块（512字节）为单位进行分配。0—8块为硬盘的控制程序。9—1032块是为了与Constellation Ⅰ兼容而保留的区域，Constellation Ⅰ的引导区、盘卷目录区等都在此处，如果网络上不运行Constellation Ⅰ，则保留区也可分配给用户使用。CORVUS盘卷用来存放系统信息、引导程序和操作系统，该区的大小与用户数目、盘卷数目、网络上的服务器数目等有关。因此，CORVUS卷和用户卷的物理位置在硬盘中是浮动的，它们的大小和位置在系统生成时由盘卷的设置情况而定。

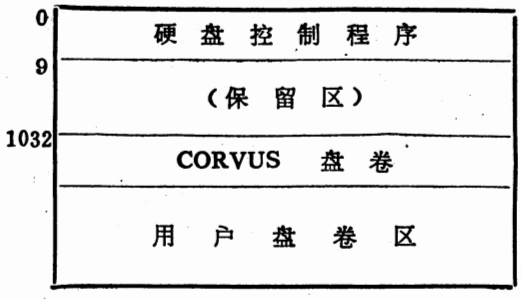


图6-23 共享硬盘存贮区域的划分

（2）盘卷及其属性

前面已经介绍过，共享硬盘的存贮区域被划分成许多盘卷，每一个盘卷是一个虚拟的软盘，但它们的大小可按照用户的使用要求而决定。在正常运行情况下，一个共享硬盘中可以包含有上百个盘卷。

每个盘卷都有一个名字，称为“盘卷名”或简称“卷名”。卷名用不多于10个字符的字符串来表示。卷的大小以块为单位按需要指定，例如可以是512块、1024块等，最大可以为32767块（约16MB）。盘卷在硬盘上的位置用它的起始块号来指出。

盘卷的类型是指该盘卷与哪一个操作系统兼容。由于不同操作系统需要不同格式的软盘片，所以要求有不同的盘卷类型。OMNINET网络可以支持的盘卷类型有多种，如MSDOS，UCSD，CP/M，CCDOS等类型。

盘卷的存取属性指的是用户使用该盘卷时的权限。存取属性有两种：只读（RO）属性和读写（RW）属性。具有只读属性的盘卷只允许读出其中的信息，不允许写入信息，就好象软盘片上贴有写保护标签一样。

共享硬盘中的CORVUS卷是一个特殊的盘卷，它是在系统生成时产生的，由网络管理程序使用。CORVUS卷中存有一张“盘卷表”，其中包含有硬盘上所有盘卷的一览表，内容包括盘卷名、盘卷类型、盘卷位置、长度、存取属性、所在硬盘驱动器号码、口令、使用状态标志（公用、专用或自由状态）以及加锁标志（信号量）等。

（3）用户及其对盘卷的访问

网络用户（简称用户）指的是允许使用网络资源的操作人员，他必须预先在网络上“开户”后才能取得网络用户的资格。

每个开户后的用户都有一个用户名和一个口令。口令是为了不让其他无关人员使用该用户在网络中的资源而设置的。此外，用户还必须有一组可以使用的盘卷，否则无法使用共享硬盘中的程序或数据文件。

用户可以访问的共享硬盘中的盘卷，必须预先得到系统管理员的授权。授权时必须指出对该盘卷的使用方式，例如只读方式或者读写方式。若使用要求高于该盘卷的存取属性（对存取属性为只读的卷要求使用读写方式），则只允许按只读方式使用。

一个盘卷可以授权给多个用户访问，也可以只授权给某一个用户访问，前者称为公用盘卷，后者称为专用盘卷。未授权给任何用户使用的称为自由盘卷。

公用盘卷可以被许多用户使用，但最好只允许有一个用户使用读写方式，而其他用户全部使用只读方式进行访问。否则，如果两个以上用户同时对盘卷中的一个文件进行修改时，将会导致信息破坏或丢失。在分布式处理中，如果必须允许多个用户都能以读写方式使用公共盘卷中的数据文件，又要保证不会发生信息丢失，则必须通过信号量（Semaphore）对文件进行加锁和解锁操作。被某用户加锁后的文件，其他用户必须等到解锁之后才能使用。Constellation Ⅱ中的许多管理程序都使用信号量进行加锁和解锁处理，以确保网络上只有一个管理员可以运行那些极其重要的管理程序。如果运行这些程序时系统掉电或者操作员未按正常操作步骤退出程序，结果该程序仍被信号量锁住，即使系统再次启动后该程序还是无法运行，那末此时必须用开锁命令进行释放，或者用信号量初始化命令把它们清除才行。

用户被授权可以访问的那些盘卷，必须被“激活”后才能正式使用。所谓激活一个盘卷，是指把它安装在某一个逻辑驱动器（除了A盘、B盘等物理驱动器之外的磁盘号）上，就好象把软盘片插入驱动器中一样。如果盘卷不再使用，则可把它“拆下”，以便空出逻辑驱动器来安装其它要用的盘卷。Constellation Ⅱ允许每个MS-DOS用户最多同时安装10个盘卷。

共享磁盘中的CORVUS卷中保存了一张用户表，其中包含着有权使用该硬盘的用户名及其口令等信息。另外在CORVUS卷中还保存着一张每个用户的所有可访问的盘卷一览表（称为“用户/盘卷关联表”），表中包含着该用户可以访问的所有盘卷的盘卷名、安装状态、使用方式、盘卷类型等信息，供系统管理程序中的“访问管理模块”使用。

3. 信息渠道及其操作命令

信息渠道（PIPES）是OMNINET网络上共享硬盘内的一个专用盘卷，它的作用相当于整个网络上的一个缓冲存贮区。利用它可以实现用户间相互通信和共享打印机。PIPES盘卷中的渠道（PIPE）可以有很多个，一个渠道相当于一个文件，它的大小并不固定，随发给它的数据的多少而增减。

用户之间进行通信时，发送节点先把全部数据发送到信息渠道，接收节点再从信息渠道取走发送给它的信息。一个渠道只能被一个节点机所访问。如果有多个节点试图向同一个渠道中放入数据，则建立名字相同的多个渠道，以免多个文件互相重叠在一起。一个渠道中的数据被取走之后，该渠道就随之消失。

PIPES盘卷必须建立在网络上的0号服务器的第1个硬盘内，盘卷名为PIPES，其大小可以是512块或1024块，视网络的规模而定。PIPES盘卷的类型必须是UCSD。

PIPES盘卷中的任何一个渠道的正常生命期为：

发送站建立一个渠道⟶向渠道中发送数据⟶关闭渠道⟶接收站打开渠道⟶从渠道中读出数据⟶取消渠道

Constellation Ⅱ向用户提供了使用渠道进行用户间通信和使用共享打印机进行打印的功能。但为了方便用户使用信息渠道自行编制网络程序，Constellation Ⅱ也向用户提供了七条有关信息渠道操作的命令，供编制应用程序时使用。下面是七条信息渠道命令的简单介绍。

① 打开（写）信息渠道命令（OPENPIPE）

这是10个字节构成的命令，由命令码和信息渠道名组成。命令执行后，送回4个字节的回答信息。本命令以给定的名字建立一个信息渠道，供写入信息之用。它的格式如下：

命 令：

1BH
80H
信息渠道名 （8个字节）

回答信息：

磁盘状态码
PIPE 出错码
PIPE 编号
PIPE 状态码

如果发送站送出的信息是给打印机打印用的，则使用专门的信息渠道名“PRINTER”。

② 打开（读）信息渠道命令（OPENREAD）

本命令打开一个信息渠道，以便读出其中的信息，命令格式同①，但操作码为1BH，C0H。

③ 写信息渠道命令（PIPEWRITE）

其功能是把数据写入信息渠道中。命令由操作码、信息渠道号及数据长度和紧跟着的要写入的数据块所组成，一次最多可以写入512个字节。执行命令后服务器送回4个状态字节作为回答。

命 令：

1AH
21H
信息渠道号
数据长度
数据 ≈ （<512B） ≈

回答信息：

磁盘状态码
信息渠道出错码
实际数据长度

④ 读信息渠道命令（PIPEREAD）

接收站使用本命令读出指定信息渠道中的信息。命令执行后，返回4个字节状态信息以及从信息渠道中读出的实际数据，一次命令最多可以读出512个字节。

命 令：

1AH
20H
信息渠道号
数据长度

回答信息：

磁盘状态码
信息渠道出错码
实际数据长度
≈ 数据 ≈

⑤ 关闭读、写信息渠道命令（CLOSEREAD，CLOSEWRITE）

这两条命令用来关闭一个已经写好的信息渠道，或关闭一个已读过的信息渠道。命令码和格式都一样，区别仅在于信息渠道状态字节。当状态字节为FEH时是关闭写，为FDH时是关闭读。而当状态字节为0时，则执行信息渠道清除命令，表示删去一个不需要的信息渠道。

命令：

1AH
40H
信息渠道号
信息渠道状态
空字节（1B）

回答信息：

磁盘状态码
信息渠道出错码

⑥ 信息渠道初始化命令

初始化命令用来对信息渠道盘卷进行初始化，命令中必须指出PIPE区的位置和大小，命令执行后服务器返回两个状态字节。

命令：

1BH
A0H
起始块号
区域大小
≈ 空字节（4B） ≈

回答信息：

磁盘状态码
信息渠道出错码

⑦ 信息渠道状态命令

它用来检测信息渠道的状态。执行命令后，服务器送回一个磁盘状态码和两张表，一张是实际信息渠道的名称表（共512个字节），另一张是指针表，其中包括每个信息渠道两端的指针及状态信息。

命令：

1AH
41H
≈ 空字节（8B） ≈

回答信息：

磁盘状态码
≈ 名称表（512B） ≈
≈ 指针表（512B） ≈

程序中使用了上述各种命令，就可以利用信息渠道进行用户之间或用户与打印机之间的通信。图6-24和图6-25是写/读一个信息渠道的过程。

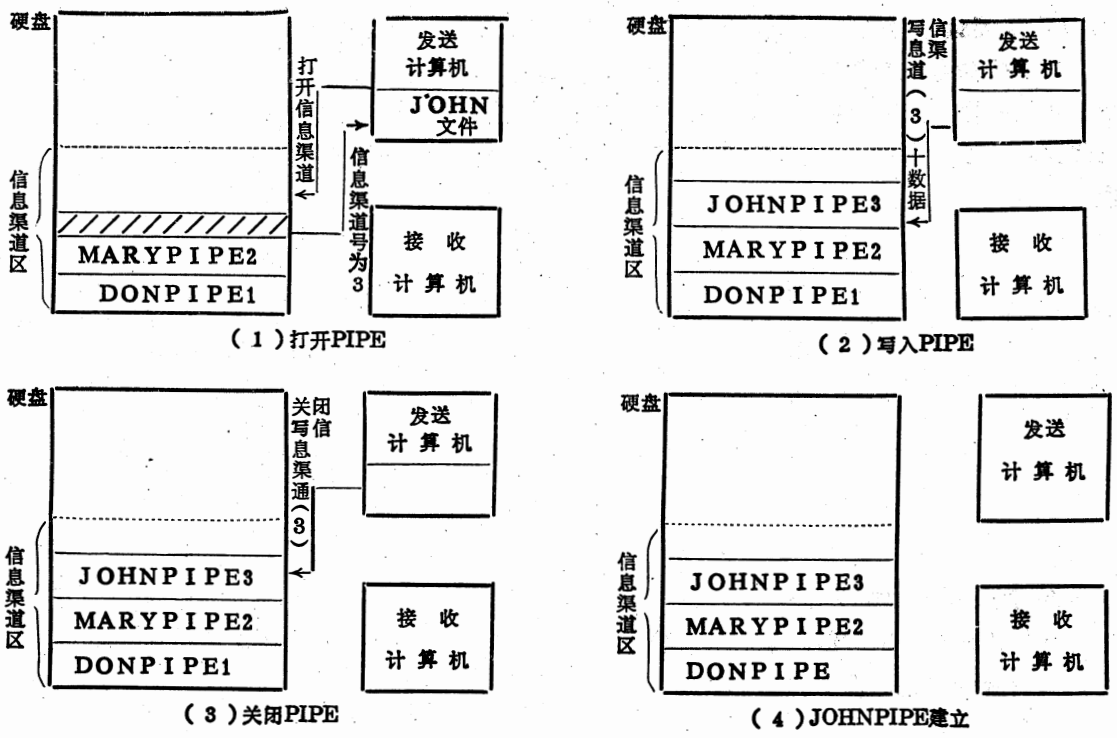


图6-24 写信息渠道

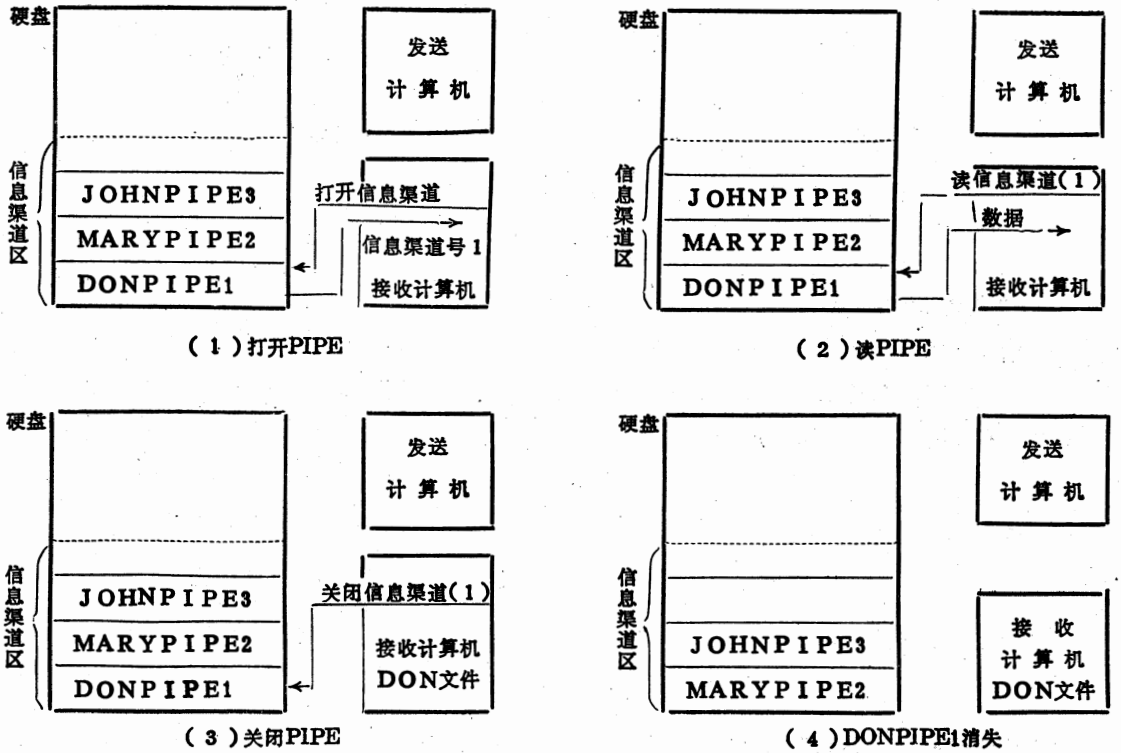


图6-25 读信息渠道

4．网络操作系统的生成

在OMNINET网络硬件的安装全部完成之后，必须在网络上建立网络操作系统，然后网络用户才能正常地工作。

建立网络操作系统主要是针对网络的共享硬盘服务器进行的，其目的是：

* 将服务器和硬盘初始化。
* 在共享硬盘上安装网络管理程序Constellation Ⅱ。
* 生成用户引导盘，供各网络用户启动节点机工作之用。
* 在共享硬盘中复制常用的网络程序，供网络用户共享。

建立网络操作系统时，网络上至少应当有一台磁盘服务器，其传输器地址必须为“0”，并且还需要使用网络上的一台PC机。除MS-DOS盘片之外，还应具备以下CORVUS IBM UTILITIES盘片：

① CORMS21（系统生成程序，硬盘初始化程序，引导程序，修改程序等）

② CORMS22和CORMS23（网络管理程序Constellation Ⅱ）

③ CORMS24（网络实用程序）

④ CORMS25（系统重构文件，传输器驱动程序的接口模块等）

下面简要地介绍网络操作系统的生成过程。

（1）服务器及硬盘的初始化

① 将CORVUS硬盘服务器加电，在网上任意一台PC机上将CORMS21盘片插入A驱动器，启动PC机工作后，就自动从软盘中读出并执行系统生成程序SYSGEN。

② SYSGEN程序提问“系统生成口令”，此时必须回答HAI，于是屏幕上就出现下列菜单：

I－Initialize a New Drive
M－Modify an Existing Drive
D－Display Drives On－Line
H－Help
E－Exit

③ 打入“I”，表示选择新磁盘的初始化操作，于是系统生成程序提问“服务器号码”。

④ 输入服务器号码“0”后，生成程序又提问“磁盘驱动器号码”。

⑤ 输入驱动器号码“1”后，生成程序又依次提问：

服务器名（例如回答SERVER 0）
服务器口令（例如回答SERVER 0）
驱动器名（例如回答DRIVER 1）
驱动器口令（例如回答DRIVER 1）

⑥ 系统生成程序提问是否按标准配置进行网络操作系统的生成。所谓标准配置，它允许系统内最多有512个用户，512个盘卷，三种不同的计算机类型，四个服务器，3072个用户/盘卷访问关系。这种配置对于大部分使用情况都已足够。但如果不使用标准配置的话，那末

可以根据表6-8中的数据来选择共享硬盘中CORVUS卷的大小（标准配置时CORVUS卷共200块）。

表6-8　CORVUS卷大小的选择

CORVUS卷的块数	最大允许值			
	用户数	盘卷数	引导块	用户／盘卷访问关系块
100—199	128	128	38—83	34—83
200—600	512	512	48—100	48—100

⑦ 于是系统生成程序进行下列操作：

* 在共享硬盘上从第9块起建立CORVUS盘卷并进行初始化。
* 打开BOOT·IBMPC文件（引导程序），写入CORVUS卷中（共11块）。
* 建立并初始化IBMSYS、IBMBOOT、IBMBACK和IBMMS四个盘卷，连同CORVUS盘卷在内，它们称为Constellation Ⅱ的系统盘卷。标准配置下，它们一共占用硬盘上连续的2538块。

至此，硬盘服务器的初始化操作已经完成。

（2）安装MS-DOS和Constellation Ⅱ

① 把MS-DOS（2.0版单面）盘片插入A驱动器，只要按一下空格键，MS-DOS操作系统就能被安装在硬盘中。

② 仍把CORMS21盘片插入A驱动器，按一下空格键，于是盘片上的修改程序(UPDATE)就被装入内存并执行修改操作。

③ 插入CORMS22盘到A驱动器中，按空格键，于是其上的内容就被复制到硬盘中。

④ 同上，但使用CORMS23盘片。操作完成后，被修改过的MS-DOS和网络管理程序Constellation Ⅱ就都在硬盘中了。

在硬盘服务器初始化并安装了网络操作系统之后，系统中已经为四个特殊的用户开设了帐户，这四个用户是：

IBMGR
IBMUSER
IBMBACKUP
TEMP

（3）生成引导盘并向IBMMS卷中复制程序

为了能从网络上的任何一个PC工作站启动并引导网络操作系统进行工作，必须生成一张引导盘。引导盘是在原有MS-DOS 2.0的基础上，从CORMS25盘片中复制三个文件而成的。这三个文件是CORDRV.BIN（传输器驱动程序的接口模块）、SPLDRV.BIN（假脱机驱动程序）和CONFIG.SYS（系统配置文件）。

有了引导盘片之后，就可以把CORMS24盘片中的常用网络实用程序复制到IBMMS盘卷

中去。操作过程如下：

① 把引导盘放入A驱动器。

② 启动机器工作，PC机读入引导盘0道1扇区的点火程序并执行。它首先判断PC机上是否插有传输器板，然后显示以下提示信息：

```
*
        *     CORVUS  SYSTEMS
                 CONSTELLATION II
              *
                    (LOGON V3.2B) (00)
*             *
Please enter your name:
```

③ 输入用户名IBMUSER，点火程序接收了用户名之后，调用网络驱动程序中的0号模块（引导模块）执行。引导模块确定了本工作站的站号是网络上唯一站号并查询了网上服务器的地址之后，就发送一条“引导命令”（短磁盘命令）给服务器，服务器执行这条引导命令，把CORVUS卷中的BOOT.IBMPC文件（共11块，5.5KB）读出，送回给工作站并装在内存的0000：7C00H开始的区域中，接着便转入执行该引导程序。引导程序根据用户登记表和其它表格，核对用户所输入的名字及口令（如果有的话），核对无误后，就从磁盘中取出经过修改以后的MS-DOS操作系统装入到内存0060：0000H区域，再将控制转向0060：0000H执行。由于IBMUSER这个用户名已在系统生成时自动开户，并且不使用口令，因此上述过程应能顺利地进行。

④ 与单机工作时一样，向系统输入了日期和时间以后，屏幕上出现提示符A>，接着就可以开始复制网络实用程序。

⑤ 在A驱动器中插入CORMS24盘片，打入下列命令：

```
COPY  A:*.*  D:
```

于是，CORMS24中的三个网络实用程序（MNTMGR2.EXE，SPOOL.EXE和DESPOOL.EXE)就被复制到D驱动器上。由于系统引导时已经把IBMMS盘卷自动安装在D驱动器上，因此这三个实用程序就存入该盘卷，成为所有网络用户可以共享的程序。

（4）以IBM PC/XT作为网络服务器的系统生成

把IBM PC/XT作为网络服务器使用，是一种经济、有效的构造PC/OMNINET网络的方法（图6-26）。这种做法要求使用OMNISHARE磁盘服务器仿真程序，作为网络服务器使用的PC/XT的工作站地址必须设置成0，内存容量应大于256KB，硬盘驱动器号为C：。OMNISHARE共三张盘片（OMNISHR01，OMNISHR02，OMNISHR03）。系统生成的过程简述如下：

① 用MS-DOS 2.0启动PC/XT工作。

② 在A驱动器中插入OMNISHR01盘片，打入命令SETUP。

③ SETUP执行后，再打入INSTALL命令，于是系统提问是否使用标准配置。

④ 回答“Y”之后，便在硬盘上复制PCS·BIN，CORDRV.BIN和CONFIG.SYS三个文件，并建立目录PCSHARE，以便在其中存放运行仿真程序所必须的文件。约两分钟之后，

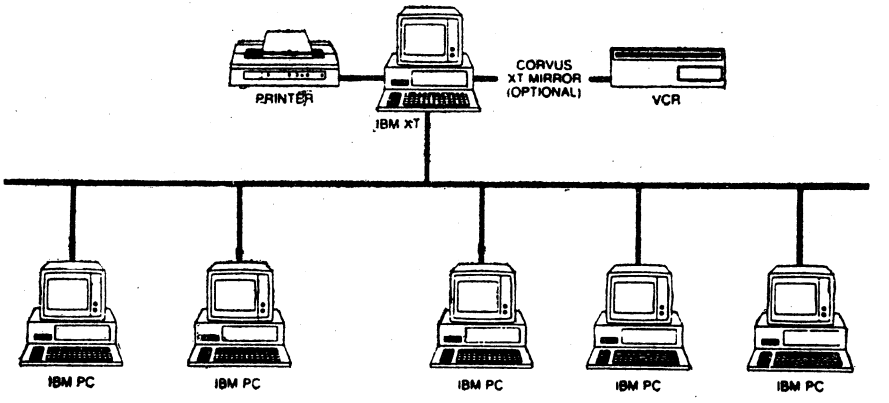


图6-26 以PC/XT为服务器的OMNINET网

文件PCSHARE.SY1，PCSHARE.SY2和PCSHARE.USR便复制在该目录中。接着又把Constellation Ⅱ软件复制到硬盘上。

⑤ 把OMNISHR02盘片插入A驱动器，按任意一键，复制Constellation Ⅱ的操作继续执行，约需4分钟。

⑥ 再把OMNISHR03盘片插入A驱动器，按任意一键，把Constellation Ⅱ的其余部分继续复制到硬盘中。复制完成后，PC/XT就仿真了一台网络磁盘服务器。

重新启动PC/XT，屏幕上将显示出仿真器初始化的一些信息，然后出现Constellation Ⅱ的登录消息。于是，网络操作就可以开始。

为了替网络上的用户生成引导盘，操作可按如下进行：

① 用IBMUSER名登录入网。

② 输入日期、时间后，把现行驱动器切换成D盘（IBMMS卷已安装在D盘）。

③ 打入命令：

D>MAKEBOOT

当屏幕上出现提示信息后，在A驱动器插入一张空盘，任敲一键后，便开始进行引导盘的生成，当屏幕上再次出现提示符D>时，引导盘便生成完毕。此后，在其他工作站上的用户就可以使用该引导盘，启动网络操作系统并登录入网。

作为服务器使用的PC/XT称为网络上的主站。网络中的主站应该最先加电，最后断电。服务器中的Constellation Ⅱ在硬盘中建立了五个专用的盘卷（称为VOL1，VOL2，VOL3，VOL4和VOL5），一个公共的盘卷（IBMMS）以及信息渠道盘卷（PIPES）。系统生成时还自动建立了五个用户的帐户，它们是USER1，USER2，USER3，USER4和USER5。每个用户均可以分别使用一个专用盘卷（可读可写）和IBMMS公共盘卷（只读）。使用上述用户名在工作站登录入网时，专用盘卷和IBMMS盘卷将被自动安装在该工作站的空闲逻辑驱动器上（如C：和D：或D：和E：）。

公共盘卷IBMMS只能由一个特殊的用户——IBMUSER——进行读写，系统生成后其中已包含有常用的网络实用程序如SPOOL.EXE，DESPOOL.EXE，MNTMGR2.EXE，MAKEBOOT.BAT，CORDRV.BIN和SPLDRV.BIN，如果需要还可以把常用的文字处

理、表格处理、数据库管理系统等软件放在其中，供网络上所有用户共享，以便节省硬盘空间。

使用OMNISHARE仿真程序使PC/XT作为网络服务器的缺点是它只能支持网络上的五个用户工作站。如果网络上有更多工作站时，可以在网中再添加CORVUS的磁盘服务器。无论采取何种服务器，PC/OMNINET网络的管理员操作和用户操作都完全相同，它们是互相兼容的。

5．网络的管理

（1）引言

作为一个局部网，OMNINET网络上的资源、用户及运行情况必须进行统一的管理，使得网络上的资源既不浪费又不滥用，重要的信息能够得到后备，网络能定期地进行维护，所有这些都是网络管理员的责任。

网络管理员是一个名字为“IBMGR”、口令为“HAI”的特殊用户，只有该用户才有权进行各种管理操作。为了保密起见，建议把他们的口令进行更换，使得整个网络只有负责人才能进行管理员操作。

管理员操作可以在网络中的任何一个工作站上进行。首先使用引导盘启动该工作站运行，然后输入用户名（IBMGR）和口令（HAI）。登录入网之后，服务器共享硬盘中的Constellation Ⅱ网络管理程序就被读出装入到该工作站，屏幕上出现下列形式的主菜单：

```
CORVUS MANAGEMENT UTILITY DS
Version [2.5]                Drive
(c) Copyright 1982, 1983 Corvus Systems, Inc.
```

```
D - Drive Management
B - Backup Utilities
M - Maintenance Utilities
C - Configure System
U - Utility Server Manager
T - Transfer Manager
I - Initialize Drive
L - List Drives
H - Help
```

```
Please select an option:
```

它表示网络的全部管理操作均是在Constellation Ⅱ的控制下进行的。Constellation Ⅱ是CORVUS公司使用UCSD P系统开发的一组用于网络管理的实用程序，它以菜单方式进行操作，使用十分方便。但必须注意，此时工作站上的IBM PC机运行的是UCSD P操作系统而不是MS-DOS操作系统。关于UCSD P系统的功能及操作请参阅本书上册第四章。

组成网络管理程序Constellation Ⅱ的主要程序模块及其与菜单的对应关系如图6-27所示。

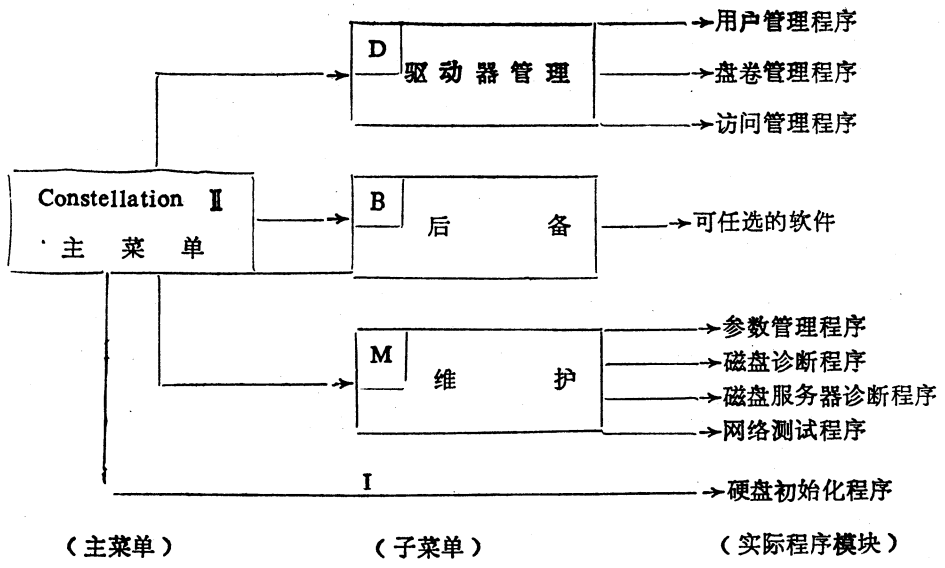


图6-27 Constellation Ⅰ的主要程序模块

网络管理员经常进行的一项工作是为新的用户开户。它的操作过程大致如下：

① 选择菜单项目D（驱动器管理），屏幕上又给如下子菜单：

U－User/Device Manager
V－Volume Manager
A－Access Manager
B－Boot Manager
S－Select Drive
L－List Drives
E－Exit

② 选择V（盘卷管理），这时Constellation Ⅰ要求管理员输入有关的服务器名字、口令及与该服务器连接的某台共享硬盘的名字和口令，全部正确后，屏幕上出现下列盘卷管理程序的菜单：

Volume Manager [2.3]　　DS SERVER 0
Main Menu　　Drive DRIVE1
(c) Copyright 1982, 1983 Corvus Systems, Inc.

A－Add a volume
R－Remove a volume
C－Change volume attributes
L－List volumes
X－Extended list
F－Free space list

```
H - Help
E - Exit
-------------------------------------------
Enter VOLMGR function:
```

③ 选择A（添加一个新盘卷），然后以会话方式输入新盘卷的名字、大小、起始位置和类型，并对它进行初始化。需要添加多个新盘卷时，本操作可反复进行多次。最后退回到驱动器管理菜单。

④ 选择U（用户/设备管理），屏幕上显示出下列用户/设备管理程序菜单：

```
User Manager [2.6]      DS SERVER 0
Main menu               Drive DRIVE1
(c) Copyright 1982, 1983 Corvus Systems, Inc.
-------------------------------------------
A - Add a user/device
R - Remove a user/device
C - Change user attributes
L - List users/devices
M - Merge user tables
H - Help
E - Exit
-------------------------------------------
Enter USERMGR function:
```

⑤ 选择A（添加用户/设备），再选择U（添加用 户），然后以会话方式输入新用户的名字、口令、主服务器及其引导操作系统的类型。最后再退回驱动器管理程序菜单。

⑥ 从驱动器管理菜单中再选择A(访问管理)并输入了用户名之后，屏幕上显示下列信息：

```
Access Manager [2.2d]      User ZHOU
Main menu                  DS SERVER 0
                           Drive DRIVE1
(c) Copyright 1982, 1983 Corvus Systems, Inc.
-------------------------------------------
G - Grant volume access
R - Remove volume access
C - Change volume access
L - List volumes accessible
N - Next User
H - Help
E - Exit
-------------------------------------------
Please select an option:
```

⑦ 选择项目G（授权盘卷访问），接着再输入该用户允许访问的盘卷名、访问方式(读写方式RW或只读方式RO）和被安装的逻辑驱动器号码。一个用户所允许访问的盘卷往往有很多个，所以本操作过程需要反复进行多次。

最后，连续输入三次“E”命令便可退回到Constellation Ⅱ的主菜单。至此，为一个新用户开户的所有工作全部完成。以后，除非该用户被网络管理员除名，否则他就可以在网络

上任何一个工作站登录入网，使用各种为用户所提供的网络操作功能，共享网络资源并和其他用户进行相互通信。

为了防止由于网络中两个以上工作站同时进行管理员操作而破坏系统所使用的有关数据，驱动器管理程序被调用时使用了信号量加锁的方法进行保护，只有操作正常结束才能使它解锁。因此，管理员操作每次都必须退回到Constellation Ⅱ主菜单。如果由于操作不慎或掉电等原因中途停止了驱动管理程序的运行，由于锁未解除，下次将无法再次启动它运行（出现107号错误）。解决的办法是：

① 从Constellation Ⅱ主菜单选择M（维护操作）。

② 显示维护操作菜单后，选择P（参数修改）。

③ 选择S（信号量设置与显示），可以看到所有未解锁的信号量。

④ 选择U（解锁），然后打入信号量名字CRVSEMA 4，于是驱动管理程序的信号量被解除。

（2）用户管理

驱动器管理程序中的用户管理模块的主要功能如图6-28所示，其中垂直方向表示子模块关系。所谓添加用户表示为一个新用户开户，取消用户表示把一个老用户从网络中除名，在添加一个新用户时，管理员除了输入该用户的名字及口令之外，还必须输入他的主磁盘服务器和所用的操作系统类型。所谓主服务器，是指包含有IBMMS卷的硬盘所连接的那个服务器。在用户启动工作站开始工作的时候，主服务器所连接的硬盘中，所有该用户可以访问的盘卷将会自动地安装在指定的逻辑驱动器上；而不在主服务器磁盘中的那些盘卷，则用户必须自行安装。

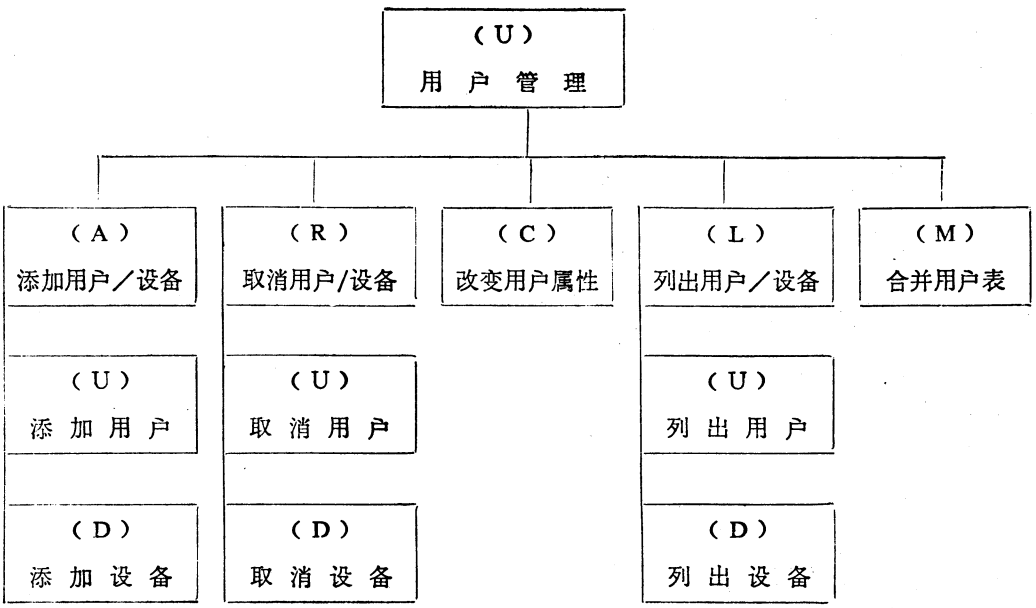


图6-28 用户管理的主要功能

改变用户属性主要用来修改用户的口令，用户名无法通过改变属性命令进行修改。如要修改用户名，必须把原用户删去后再建立新用户名。

列出用户命令可以给出网络中所有用户的名字、口令、主磁盘服务器名及引导操作系统类型的一览表，显示形式如下所示：

```
User Manager [2.6]      DS SERVER 0
List users/devices  Drive DRIVE 1
```

	User name	Password	Home DS	Boot type
1.	IBMBACKUP		SERVER 0	UCSDIV. 0
2.	IBMGR	HAI	SERVRE 0	UCSDIV. 0
3.	IBMUSER		SERVER 0	MSDOS
4.	TEMP		SERVER 0	MSDOS
5.	ZHOU	AAA	SERVER 0	MSDOS
5	users listed.			

设备的添加操作指的是在系统中添加某项任务专用的一些设备，例如打印服务器之类的设备，此处不再细述。

（3）盘卷管理

盘卷管理程序用来在共享硬盘中添加盘卷、删去盘卷、修改盘卷属性及列出盘卷目录等，其模块结构如图6-29所示。

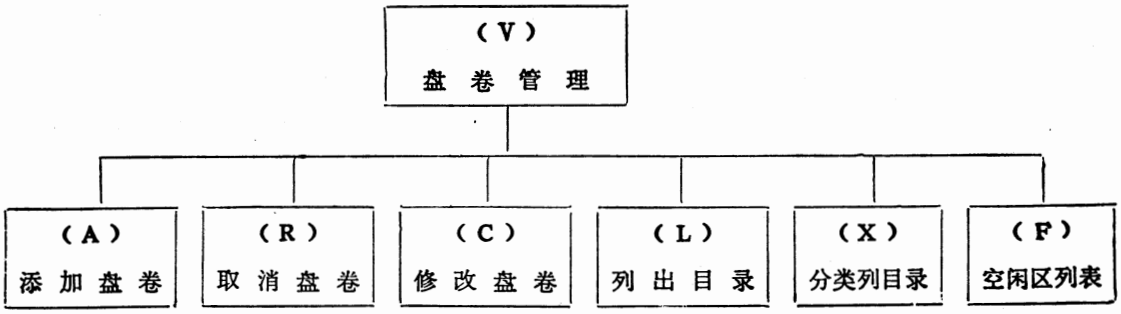


图6-29 盘卷管理的主要功能

在共享硬盘中添加新盘卷时，必须输入盘卷名、大小、位置及盘卷类型。在输入盘卷位置时，盘卷管理程序会指出硬盘中最大空闲区域的位置。如果其大小合适的话，输入回车表示默认，否则，可以通过F命令列出盘卷中所有空闲区域的大小及位置，挑选合适的空闲区域使用。建立新盘卷的最后一步是进行初始化操作，即在该盘卷中形成一个与所用操作系统类型相符的目录区。对于IBMMS盘卷来说．必须给出簇的大小（文件的分配单位，可以是1、2、4、8、16…个扇区）、目录项的数目和盘卷中保留扇区的数目。簇的大小与盘卷中存放的文件平均大小有关。若文件大，簇可以放大一些；若都是小文件，则使用小一些的簇。保留扇区用于存放引导信息，有些应用软件也会使用它，因此建议每个盘卷有一个保留扇

区。表6-9是不同大小的盘卷所使用的不同初始化参数的建议值，用户可以按具体情况灵活处理。

表6-9　不同大小盘卷的初始化参数

盘卷大小	簇的大小	目录个数	FAT 扇区数目	保留 扇区数目	容量
1024	4	256	1	1	512KB
4000	8	256	2	1	2000KB
8000	8	256	6	1	4000KB
16000	8	256	64	1	8000KB

列出盘卷目录命令（L）可以在屏幕上显示出共享硬盘中已有盘卷的名字、起始地址、长度、存取属性和盘卷类型。下面是盘卷目录的一个显示画面，其中RW栏中标以“×”者表示存取属性为可读可写，否则表示存取属性为只读类型，屏幕底部显示了硬盘中空闲区域的个数、总的大小、最大空闲区的大小、已有盘卷总数、硬盘已分配块数及最大盘卷的大小等等信息。如果按下“F”键，上述盘卷目录可以作为一个正文文件被保存到硬盘或软盘上，供打印输出之用。

```
Volume Manager [2.3]        DS SERVER 0
List Volumes                Drive DRIVE1

    Volume      Address   Length   RW   Type
 1. CORVUS          9      300     ×    UCSD
 2. IBMSYS        309     1024     ×    UCSD
 3. IBMBOOT      1333       90     ×    UCSD
 4. IBMBACK      1423      200     ×    UCSD
 5. IBMMS        1623     1024     ×    MSDOS
 6. VOL1         2647     1024     ×    MSDOS
   <unused>      3671    84529

      Total free blocks on drive:    84529
       Total free areas on drive:        1
  Largest free space (blocks):       84529
        Total volumes on drive:         6
Total blocks allocated on drive:     3662
 Largest volume size (blocks):       1024

Press <space> to continue, or
press F to list to a file.
```

需要修改盘卷名或存取属性时，可以使用命令“C”进行。如果需要修改盘卷的大小、位置或类型，则必须先把它的内容复制到软盘中，删去该盘卷然后重新建立一个新盘卷，再把原来内容恢复到新盘卷中即可。

（4）访问管理

访问管理程序的主要任务是授与或取消用户对盘卷的访问权，修改对盘卷的访问权，或者列出可访问的盘卷一览表（图6-30）。

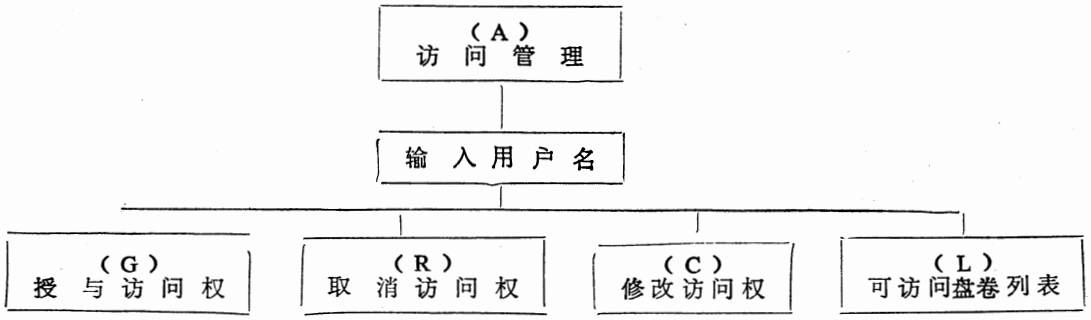


图6-30 访问管理的主要功能

访问管理程序给用户授与盘卷访问权时，管理员应该输入盘卷名，对该盘卷的访问要求（RO代表只读，RW代表可读可写）、以及该盘卷是否需要自动安装等。应当注意，为许多用户同时共享的公共盘卷，只能给一个用户授与可读可写的访问权，其他用户只能享有只读方式的访问权。另外，存取属性为“只读”的盘卷不允许授与用户可读可写的访问权。

用户可访问的盘卷必须安装后才能使用。安装有两种方法：系统自举时自动安装或用户自行安装。能自动安装的盘卷必须在该用户的主服务器所连接的磁盘上，且在授权过程中应该指出它的安装单元号码（1—10），供以后自动安装的时候与该工作站空闲的驱动器号相匹配。例如，假设该工作站已有两个软盘（A：和B：）和一个硬盘（C：），则系统自举时安装单元为1的盘卷被自动安装在驱动器D：上，安装单元为2的盘卷被装在驱动器E：上，其它以此类推。所有IBM PC用户共享的盘卷IBMMS,在授权时应该被设置为自动安装状态为宜。

为了检查用户所有可访问的盘卷，使用命令“L”可以在屏幕上列出如下的一览表，其中UNIT栏表示该盘卷是否自动安装，若是，则给出安装单元号码。

```
Access Manager [2.2d]          User   ZHOU
List Volumes                   DS     SERVER0
                               Drive  DRIVE1
```

	Volume	Unit	RW	Type
1.	IBMBOOT	-*		UCSD
2.	IBMMS	2		MSDOS
3.	VOL1	3	×	MSDOS
4.	VOL2	-	×	MSDOS

5. VOL3 - × MSDOS

Number of volumes accessible: 5.

Press < space > to continue, or

Press F to list to a file.

使用“C”命令允许修改用户对可访问盘卷的访问方式、安装状态及安装单元的号码。用户不再需要访问的盘卷，可以用“R”命令删去。

（5）信息渠道管理

如前所述，信息渠道是一个特殊的盘卷，它的名字为PIPES，类型为UCSD。信息渠道相当于是一个电子信息中心，它在网络上起着信息交换枢纽的作用。由于信息渠道也是一个盘卷，所以有关信息渠道的管理操作如建立、初始化、维护等都是由盘卷管理程序完成的，下面予以简要介绍。

① 建立名字为PIPES的盘卷

信息渠道盘卷必须建立在网络中的0号服务器的1号硬盘上，大体步骤如下：

* 从驱动器管理程序中，进入盘卷管理程序。
* 选择A（添加盘卷），然后输入盘卷名PIPES。盘卷大小可以是512或1024块，视网络上工作站多少而定，盘卷位置必须在硬盘的前32000块之内，盘卷类型为UCSD。
* 对PIPES盘卷作初始化操作。由于其类型为UCSD，因此不完全象MS-DOS那样操作。

这样就建立了PIPES盘卷。

② 为了把PIPES盘卷作为信息渠道使用，还必须进行下列处理：

* 从Constellation Ⅱ主菜单上选择M（维护程序），于是屏幕上显示如下信息：

CORVUS UTILITY [2.5] DS SERVER 0

Maintenance Utilities Drive DRIVE1

P—Parameter Manager

D—Disk Diagnostic

O—OmniDrive Diagnostic

B—Bank Diagnostic

T—Omninet Test

U—Update Utilities

S—Select Drive

L—List Drives

E—Exit

Please select an option:

* 选择P（参数管理），屏幕上显示：

M—Master Multiplexer

P—Pipes

S—Set/Display semaphore parameters

* 再次选择P（信息渠道维护），屏幕上给出如下信息：

```
PARMGR [2.2b] : Corvus Parameter Manager Program
(c.) Copyright 1982, 1983 Corvus Systems, Inc.

    L—List active pipes
    R—Reinitialize (delete all pipes)
    I—Initialize pipes area
    C—Close a pipe
    P—Purge a pipe
    E—Exit

    Select Pipes option:
```

* 选择I（初始化），操作完成后返回Constellation Ⅱ主菜单即可。

③ 信息渠道区的维护

由于使用中的差错（例如发送工作站和接收站使用的信息渠道名字不一致等），经常会导致PIPES卷中残留一些无用的信息渠道。如不定期对PIPES卷进行维护清理的话，则不断积累会使PIPES中的有效区域大为缩小。信息渠道区的维护过程如下：

* 进入维护程序后，选择P(参数管理),然后再选择P，系统进入信息渠道维护操作。
* 按下“L”键，屏幕上将列出PIPES卷中所有信息渠道的名字、状态（读或写）和大小，例如：

```
Active Pipes are:
1.PIPI      Closed ··· Contains data      2 blocks
2.BAS       Open       Read Empty         0 blocks
3.PRINTER   Closed ··· Con'ains data      6 blocks
4.MAIL      Open       Read Empty         0 blocks
Press 〈space〉 to Continue
```

其中，处于打开状态的渠道可能有某个工作站正在向里面写入数据，或者正在被某个工作站读出其中的数据。处于关闭状态的渠道表示发送站已写完全部数据，但接收站尚未开始读出其中的数据。名字为PRINTER的渠道，接收站一定是打印机。如果一个信息渠道处于打开状态而它的大小始终不变，则该渠道就有问题。解决的办法是或者把它关闭，或者把它取消。若一个关闭着的渠道始终不被某工作站取走，则也应该把它取消。

* 选择P（取消），然后输入要取消的信息渠道的号码。或者选择C（关闭），输入要关闭的信息渠道的号码。经过清理后，再退回Constellation Ⅱ主菜单即可。

第四节　IBM PC/OMNINET网络的使用

1. 用户工作站和网络MS-DOS操作系统

（1）网络MS-DOS的启动

OMNINET网络上的每台节点计算机，除了用于仿真磁盘服务器的PC/XT机之外，均可以作为用户工作站使用。只有经过网络管理员开户之后的用户，才能使用工作站进行有关网络资源共享和相互通信的操作。

用户可以使用网络中的任何一台PC节点机进行工作，但在工作之前首先必须登录入网，其步骤为：

* 使用引导盘启动PC机工作。
* 输入用户名和口令。

此后，该工作站就从磁盘中读出已被修改了的MS-DOS操作系统装入内存中运行，该用户可访问的、且处于自动安装状态的盘卷（如IBMMS）被自动安装在节点机的空闲逻辑驱动器上，登录操作便到此完成，屏幕上出现MS-DOS的提示符“A>”。

在工作站上运行的被修改过的网络MS-DOS操作系统，与原来的单机MS-DOS相比，主要增加了两个功能模块：

* 网络硬盘访问管理程序。
* 传输器驱动程序。

后者常驻在传输器板上2KB的ROM存贮器中，其功能已在第二节作过介绍。

网络硬盘访问管理程序添加在原MS-DOS的IBMBIO中，它使得IBMBIO增加了工作站磁盘/共享硬盘的驱动器号码的判断、共享硬盘读写操作的管理等功能。当用户进行键盘命令操作时或运行用户程序时，只要出现了磁盘操作的要求，首先通过查表操作（表的内容在用户登录时由Constellation Ⅱ填入）判断涉及的是工作站磁盘还是网络上的共享硬盘，若为后者，则通过共享硬盘读写操作的管理程序进行。此时，必须由传输器向服务器发送包含有长/短磁盘操作命令的信息包，服务器接收信息包后，把磁盘操作命令交给共享硬盘控制器去执行。

MS-DOS的IBMDOS·COM和COMMAND·COM两个模块，在网络MS-DOS中基本上未作改动。

可见，从操作系统的角度来说，网络MS-DOS与原单机MS-DOS完全兼容，仅仅扩充了访问网络中共享硬盘盘卷的功能而已。

（2）盘卷的安装与拆卸

网络管理员授权给用户的所有可访问的盘卷，在使用前必须进行安装。用户在工作站上启动网络MS-DOS运行时，在该用户可访问的盘卷中，凡处于自动安装状态的盘卷都已进行了安装，它们根据安装单元号码，依次安装在该工作站的空闲的逻辑驱动器上，由于网络MS-DOS最多只能安装A—J共10个盘卷（工作站所连接的软、硬盘均包括在内），而可访问的盘卷总数一般远远大于10，因此在需要时，用户可以使用盘卷安装管理程序来自行安装盘卷或替换已安装的盘卷。

盘卷安装管理程序（MNTMGR2）是一个网络实用程序，在系统生成时它已存放在IBMMS盘卷中（下文假设IBMMS已安装在D驱动器上）。为了启动该程序运行，只要输入下列命令即可：

```
D：MNTMGR2
```

于是屏幕上出现下列信息：

```
CORVUS MOUNT MANAGER UTILITY      〔2.1D〕
Number of disk servers:  1
Building access table for: SERVER0
```

```
MOUNT MANAGER - [2.1D]
(c)1983, 1984 by Corvus: Inc.
User: ZHOU
Server: SERVER0
Drive: *
```

```
C—Change Mount Status
L—List Units and Volumes
S—Select Drive
H—Help
E—Exit
```

```
Enter MOUNT MANAGER option:
```

为了安装或拆卸一个盘卷，需要进行如下操作：

① 通过S命令（选择共享硬盘），输入被安装的卷所在的服务器名和驱动器名。

② 通过L命令（列表）列出当前已安装的盘卷的一览表，其形式如下：

```
Mount Unit Assignments -
```

unit	volume	write acc	length	clus	res sects	dir ents	FAT sects	server	drive
D	IBMMS	no	1024	4	0	256	1	SERVER0	DRIVE1
E	VOL1	yes	1024	4	1	256	1	SERVER0	DRIVE1
+F			0	0	0	0	0		
+G			0	0	0	0	0		

```
+indicates available unit
Press <SPACE> to continue
```

其中，驱动器D和E已分别安装了两个不同的盘卷，驱动器F和G尚未安装任何盘卷。当按下空格键时，屏幕上将显示出该用户在指定的共享硬盘上所有可访问的盘卷的一览表，形式如下：

```
Volumes Accessible -
```

unit	Volume	write acc	length	clus	res sects	dir ents	FAT sects	server	drive
D	IBMMS	no	1024	4	0	256	1	SERVER0	DRIVE1
E	VOL1	yes	1024	4	1	256	1	SERVER0	DRIVE1
-	VOL2	yes	1024	4	1	256	1	SERVER0	DRIVE1
-	VOL3	yes	1024	4	1	256	1	SERVER0	DRIVE1

```
-indicates unmounted volume
press <SPACE> to continue
```

由此可见，其中VOL2和VOL3两个盘卷尚未被安装，记下它们的名字以及FAT所需要扇区的数目，供安装时使用。

③ 回到主菜单后，选择C（修改盘卷的安装状态）以便安装或拆卸一个盘卷，再输入盘卷名（如果该盘卷不在已选择的服务器和驱动器上，则在盘卷名之前加上服务器名和驱

动器名，或者加一个“*”号），于是屏幕上显示出：

```
Location of volume -
                    Server: SERVER 0
                    Drive: DRIVE1
Current Status of Volume -
                Mount Status: Unmounted
Enter New Status of Volume -
                Mount Status (M/U): M
```

于是，通过会话方式输入安装还是拆卸（M或U）以及安装的逻辑驱动器号码以及盘卷的访问方式（只读或可写），该盘卷就安装（或拆卸）完毕。

需要注意的是，在已安装了盘卷的逻辑驱动器上再次安装一个盘卷时，原盘卷将自动拆卸，但新安装盘卷的FAT不能超过原来盘卷的FAT的大小。在空闲的逻辑驱动器上安装一个盘卷时，该盘卷的FAT不能超过任何一个已安装盘卷的FAT的大小。

为了保护某个盘卷的内容不受破坏，可以把对该盘的访问方式改为只读方式，或者把该盘卷修改成未安装状态。

已安装的盘卷，用户可以在操作命令或应用程序中通过其逻辑驱动器号码直接使用，它们与工作站本身所连接的软盘或硬盘在功能上没有任何差别。

（3）盘卷的后备与恢复

用户专用盘卷中的重要信息，应当定期地从共享硬盘中复制到软盘上保存起来，防止因硬盘故障或网络失效而丢失信息。

在进行备份操作之前，应根据欲保存的信息量，准备好足够的软盘片，并预先在MS-DOS下进行格式化。下面是进行备份操作的几个主要步骤：

① 用引导盘启动工作站运行，输入一个特殊的用户名IBMBACKUP，把备份操作程序从共享硬盘的IBMBACK卷中读出，装入工作站中执行。

② 然后再输入需要进行备份操作的用户名及其口令，于是屏幕上显示出备份操作菜单：

```
CORVUS BACKUP-TO-FLOPPY UTILITY [1.3]
Copyright 1983 by  Corvus Systems, Inc.
-------------------------------------------
  B—Backup a Volume
  R—Restore a Volume
  I —Identify a Diskette
  L—List Volumes
  S—Set Options
  E—Exit
-------------------------------------------
  Please select option:
```

③ 选择S（设置附加功能），备份操作程序将询问是否需要校验和口令保护。若要求在制作备份的过程中同时进行校验工作，则在提示符Verification后回答“Y”，若要求防止他人从备份软盘上读取信息，则在提示符Password Protection（口令保护）后回答“Y”，这样，备份软盘必须查对用户名和口令后才能读取。

④ 返回主菜单后，用L命令列出该用户的所有盘卷的名字和长度，根据盘卷的大小再按照表6-10就可以推算出所需要的软盘片的数量。

表6-10　盘卷大小与软盘数量的对应关系

卷的大小	512块	1024块	2048块（1MB）	4096块
软盘片数	1	2	3	6

⑤ 从主菜单打入B命令进入备份操作，屏幕上显示：

```
CORVUS BACKUP-TO-FLOPPY UTILITY 〔1.3〕
          Backup a Volume
-------------------------------------------
Please enter:
  Volume name:
        Date:
```

⑥ 输入欲复制的盘卷名字及日期后，复制操作便立即开始，直到盘卷中所有信息均复制完毕为止。

从备份软盘上恢复原有盘卷内容的操作与上述过程相似，其大体步骤如下：

① 从备份操作实用程序的菜单中执行命令S（设置附加功能），选择是否需要校验以及是否有口令保护。

② 执行命令I（软盘识别），把备份软盘插入驱动器A中，按下空格键后，屏幕上就显示出该备份软盘的一些标识信息，其形式如下：

```
CORVUS BACKUP_TO_FLOPPY UTILITY 〔1.3〕
      Identify a Diskette
-------------------------------------------
Volume Backed Up          : WORK
Date of Backup            : 5-03-85
     User Name            : ZHOU
Size of Volume (blocks)   : 1024
Total number of Diskettes : 5
Number of this Diskettes  : 1
```

用户必须核对一下备份软盘是否正确无误。

③ 通过命令L（列盘卷目录），列出该用户所有可访问的盘卷的名字、大小及访问方式，从中决定哪一个盘卷供恢复信息使用。

④ 执行命令R（恢复盘卷），然后输入共享硬盘上被恢复内容的盘卷名和备份软盘上的盘卷名（两者允许不同名字，但大小和类型应该相同）。如果备份软盘有保护的话，还要输入用户名和口令。此后，复制操作便自动进行，直到备份软盘中的信息全部被恢复到盘卷中为止。

2. 工作站通信及打印机共享

OMNINET网络的用户利用信息渠道（PIPES盘卷）可以实现相互之间的通信，并在网

络中的共享打印机上打印输出。这些操作是分成两个阶段来进行的。首先，发送信息的工作站必须指定一个信息渠道的名字并向它送出一个文件；其次，接收信息的工作站（包括打印工作站）在PIPES盘卷中查找有无应该接收的文件。若有则接收指定渠道中的一个文件，并把它保存在接收站用户的盘卷中，或者显示在屏幕上，也可以从打印机上输出。

为了方便起见，发送站发送信息时所使用的信息渠道名，通常就是接收站的用户名。如果信息是发送到网络上供共享打印机打印之用，则信息渠道名为PRINTER。

假如网络上连接有CORVUS公司提供的打印服务器的话，打印服务器将不断地自动查找PIPES盘卷中名为PRINTER的信息渠道，并把其中的文件内容在打印机上打印出来。如果网络上未连接打印服务器，那末可以把某个连接有打印机的PC工作站兼作打印服务器使用，只要该服务站有空闲，就应启动执行接收程序（DESPOOL），把名为PRINTER的信息渠道中的内容打印出来。

下面介绍向信息渠道发送文件和从信息渠道接收文件的过程。

（1）利用SPOOL实用程序发送文件

SPOOL程序是一个网络实用程序，发送工作站利用这个程序可以把本用户的某个文件以指定的渠道名送入PIPES盘卷。它的主要功能模块有三个，即发送一个文件（S）、修改发送参数（C）以及显示信息渠道的状态（D）。利用SPOOL程序向PIPES发送一个文件的过程（假设SPOOL程序存放在IBMMS盘卷中，IBMMS已安装在驱动器D上）介绍如下：

① 用户登录入网后，输入“D：SPOOL”命令，该程序就从共享硬盘中读出，装入到工作站主机并启动运行。它的主菜单画面为：

```
S—Spool A File
C—Change Spool Parameters
D—Display Pipes
```

② 为了指定信息渠道的名字，选择命令C（改变发送参数），屏幕上显示出下列信息：

```
Corvus Spool Utility [3.0]
Change Spool Parameters
```

```
P—Pipe name              : PRINTER
L—Lines per page         : 55
C—Chaining symbol        : (*I
N—New page symbol        : (*P
T—Tab length             : 8
S—Strip high bit         : FALSE
F—File type              : TEXT
E—Exit to Main Menu
```

```
Please select an option:
```

其中，画面右部为缺省的发送参数，它表示信息渠道名为PRINTER（打印机），每页55行，文件类型为正文文件，不删除字符中的高位“1”，列表符长度为8个字符，换页控制

符为（＊P等等。如果有关参数不符合要求，可以选择画面左部的相应命令进行修改。其中链接符号表示当正文文件中出现该符号时，SPOOL程序会自动在该处插入一个被链接的文件发送到PIPES中去。

例如，发送站需要把文件发送给另一个工作站（而不是打印输出）时，首先选择P命令输入信息渠道的名字（接收站用户名），然后再选择F命令，把文件类型改成适当的类型（如DATA类型）。如果发送站把文件送到名字不是PRINTER的共享打印机去打印，则使用P命令修改成正确的信息渠道名即可。

③ 回到SPOOL主菜单后，执行S命令(假脱机输出），然后输入要发送的文件名。如果该文件是送到网络打印机去打印的话，还可以输入一行附加信息（例如，打印文件的用户名和日期），这行信息将作为打印输出时的标题使用，以便区分不同用户的打印输出。于是SPOOL程序便把指定文件以规定的信息渠道名送入PIPES盘卷之中。如果需要发送多个文件的话，可以重复进行上述操作。

④ 为了检查发送操作是否正确完成，可使用D命令(显示信息渠道），并接着输入信息渠道的名字，这样SPOOL程序就会把PIPES盘卷中具有该名字的所有渠道显示出来。例如：

2.PAUL Closed…Contains data 84 blocks

它表示PIPES卷中有一个名为PAUL的信息渠道，编号为2，处于已关闭状态，渠道中包含有84块二进制数据。如果正确无误，发送文件的过程便到此结束。

（2）利用SPOOL DRIVER发送文件

上面所介绍的向网络的其它节点发送文件的方法，是在发送工作站上由用户手动进行的。如果需要在应用程序中向网络其它节点发送文件，则只要把SPOOL DRIVER（假脱机驱动程序）安装在网络MS-DOS中，就可以利用程序中普通的输出语句（如PRINT语句，WRITE语句等）向网络发送信息。

假脱机驱动程序驻留在网络MS-DOS的引导盘中，其文件名为SPLDRV·BIN。由于引导盘中的系统配制文件CONFIG.SYS包含有下列信息：

DEVICE=SPLDRV.BIN

因此，系统启动成功后，SPLDRV.BIN将代替原MS-DOS中标准的打印机驱动程序PRN和LPT1。这样，该工作站所连接好的打印机（称为本地打印机）将不起作用，而所有操作命令或程序中的打印输出均被假脱机驱动程序SPLDRV以PRINTER作为信息渠道名发送到PIPES盘卷中去。例如，下列两条命令的执行结果均能把文件REPORT·TXT送入PIPES中：

```
PRINT REPOPT·TXT
COPY REPORT.TXT PRN
```

为了灵活地使用SPLDRV程序，SPLDRV的许多参数都可以由一个实用程序CSD·COM进行修改。CSD·COM是修改假脱机驱动程序的实用程序。假设它在A盘上，只要打入“A：CSD.COM”命令就可调出执行。该程序运行时，屏幕上显示出：

```
CHANGE SPOOL DRIVER - 〔1.03〕
Copyright (c) 1984, CORVUS SYSTEMS, INC.
```

```
CURRENT PARAMETERS
Pipe Name      : PRINTER          File Name       : OUTPUT.TEXT
Server         : SERVER 0         Print Device    : LOCAL
Time Out       : 15 sec.          Control String  : ^Z^Z
-------------------------------------------------------------

MENU
C—Change Control String           S—Change Spool Server
D—Change Print Device              T—Change Timeout
E—Close Pipe
F—Change File Name                 H—HELP
P—Change Pipe Name                 X—EXIT
-------------------------------------------------------------

Please select an option:
```

其中，当前正在使用的假脱机输出参数为：信息渠道名是PRINTER，服务器为SERVER 0，打印纸上输出的文件名为OUTPUT·TXT，打印输出设备N表示非本地打印机（即网络中的共享打印机），LOCAL表示本地打印机，控制字符串^Z^Z即文件的结束字符，当SPLDRV遇到该字符时就表示文件输出结束，于是关闭信息渠道。Time out 15表示超过15秒不向信息渠道该写入信息，则该信息渠道自动关闭。

上述参数均可使用菜单中的有关命令进行修改。例如，D命令可以把共享打印机改成本地打印机（LOCAL），使工作站自带的打印机又可联机工作；P命令可以修改信息渠道名，使文件不是传送给共享打印机而是留在PIPES卷中等待接收工作站取用。

用CSD程序修改SPLDRV的参数仅仅是临时性的，当系统重新启动时，这些参数又恢复成初始状态。如果要永久性地修改SPLDRV中的某些参数，甚至SPLDRV不必安装到网络MS-DOS中去，则必须修改系统配置文件CONFIG.SYS。有关做法请参见第一章，这里不再细述。

（3）用DESPOOL程序接收文件

发送站送出的文件均保留在PIPES盘卷中等待接收站取走。除了网络上的打印机服务器能够自动取走并打印出名字为PRINTER的信息渠道中的文件之外，其它信息渠道中的文件必须由接收工作站运行DESPOOL程序才能被取走。

DESPOOL是一个网络实用程序，通常它保存在IBMMS盘卷中，用户工作站需要时可以把它调出执行。执行该程序时屏幕上显示出下列主菜单：

```
Corvus Despool Utility [3.7]
Main Menu
-------------------------------------------

    D—Despool a file
    C—Change despool parameters
    H—Help
    E—Exit
```

为了从PIPES中接收一个文件，操作过程大体如下：

① 打入C，修改接收参数，这时屏幕上显示：

```
Corvus Despool Utility [3.7]
Change Parameters
-------------------------------------------------------
P—Pipe name                              : PRINTER
O—Output device                          : PRINTER
L—Do you want <lf> after <cr>            : YES
S—Single page printing                   : NO
E—Exit to main menu
-------------------------------------------------------
Please select an option:
```

于是，使用命令P可以改变信息渠道名，命令O可以选择输出设备（打印机、文件或控制台显示器），命令L用来决定回车符后面是否增加一个换行符，命令S可以控制单页还是多页打印。

② 修改好接收参数之后，再回到DESPOOL程序的主菜单，执行D命令，屏幕上显示出下列信息：

```
Currently despooling pipe named    : ZHOU
Pipe number                        : 10
Spooled filename                   : A : CONFIG.SYS
Filename to despool to             : C : CONFIG.BAK
```

其中，第1行和第2行为已选定的信息渠道名和在PIPES卷中的编号，第3行为保存在该渠道中的文件名，第4行为建议接收站用户使用的文件名，它与原文件名一致。如果用户需要自行给出接收文件名的话，直接输入即可。接着便进行文件接收操作，结束后屏幕上给出被接收文件的块数，然后再在PIPES卷中寻找下一个同名的信息渠道。找到后，如果用户需要接收，就可重复进行上述操作。

（4）点对点直接通信

从上面的叙述中可以看出，OMNINET网络的用户通信都是间接地经过共享硬盘来完成的，接收方一般不知道发送方何时要求与其通信，因此在某些使用场合中极为不便。同时，网络硬盘系统一旦工作不正常，就会造成整个网络瘫痪。但是，如果网络上各工作站之间能直接通信（称为点对点通信），那末文件就可以直接传递到指定接收站的软盘或硬盘中去，也可以利用广播方式把文件和信息直接传递到网络上的所有工作站。这样，不但提高了现有OMNINET网的功能，也大大提高了网络上文件传递的速度，并使可靠性、保密性有所改善。

实现点对点通信需要修改OMNINET网络中的部分硬件与软件。修改硬件的目的，是使原来的传输器在每执行完毕一条通信命令后便能向CPU发出一次中断请求。方法是引出MC6801芯片的18脚，经过适当的控制电路处理后，作为IBM PC输入输出总线上的中断信号INT 3使用(图6-31)。软件修改的目的，则是使得原来的传输器驱动程序能够在不访问硬盘

时允许产生点对点通信中断，而不让访问共享硬盘的操作产生任何不必要的中断，并增加处

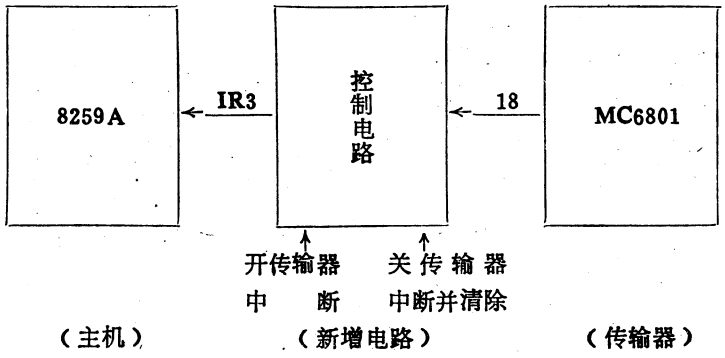
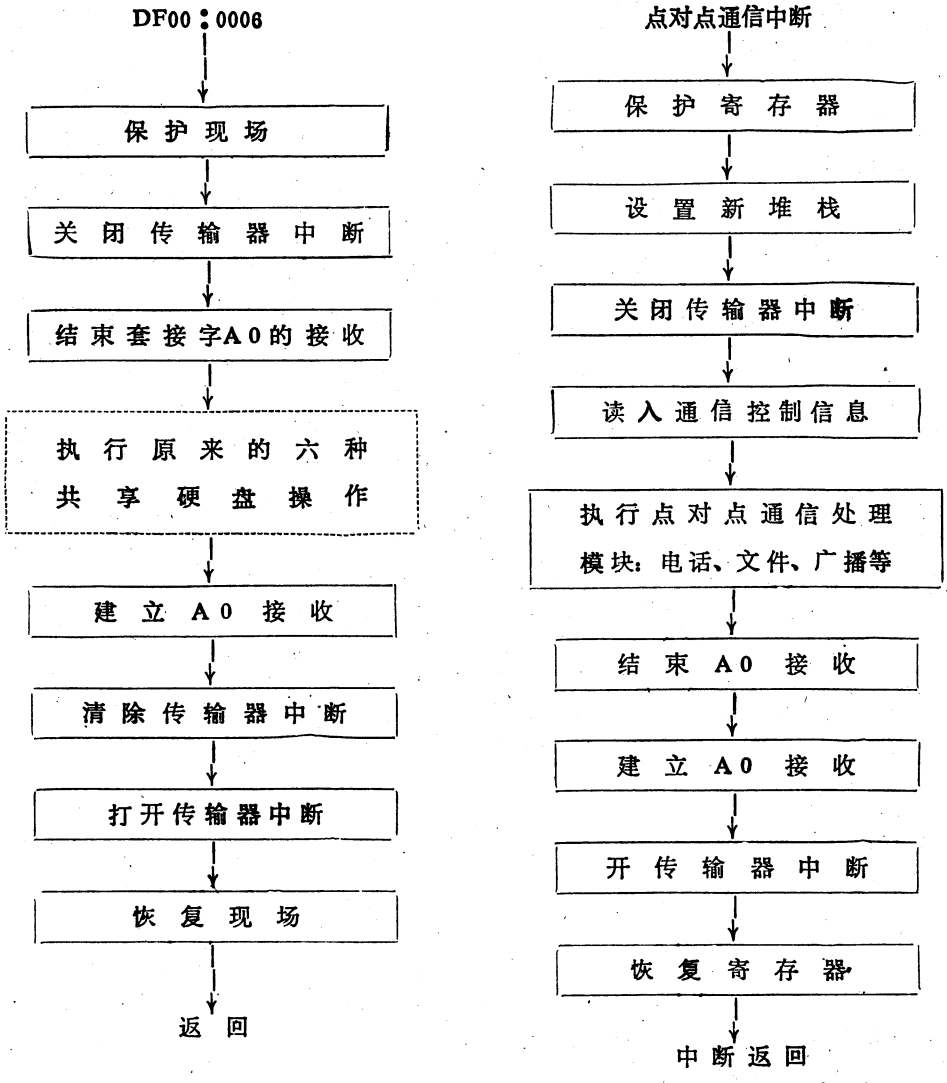


图6-31 传输器中断及其控制



（a）修改后的传输器驱动程序　　（b）新增的通信中断服务程序

图6-32 实现点对点通信的软件修改

理点对点通信中断的中断服务程序。图6-32是传输器驱动程序中第3模块（访问共享硬盘）经过修改后的流程图和新增加的点对点通信中断服务程序的流程图。

3. 电子邮政软件

（1）概述

电子邮政软件是一个通信工具，它允许局部网络或局部网与远程网上的用户相互邮寄信件和文件（以下统称邮件），这在办公自动化中特别有用。美国Software Connections公司为PC/OMNINET局部网开发的LAN：MAIL MONITOR就是一种网络应用软件。运行这个软件，就相当于在安装了局部网的单位内部自行建立了一套邮政服务措施：一个邮局和许多分布在各用户工作站的邮箱，以及邮件准备、寄出、分发、接收、保存等一整套自动化管理手段，使用十分方便、可靠。

MAIL MONITOR的主要特色有：

* 网络工作站用户可以相互发送信件，也可以通过调制解调器和电话线与其他网络用户进行通信。远程信件可以有两种发送方式：立即发出或者等候到预定时间再发出。
* 使用口令确保邮件的安全，非收件人无法取得和阅读邮件。
* 采用许多辅助手段用来提高邮件准备阶段的工作效率，如文字处理、自定义格式、邮件分发表等等。此外，除了发送信件外，还可发送文件，例如表格、程序、数据等。
* 允许根据发信人姓名、发送日期、信件主题等有选择地接收邮件，邮件收到后可以进行打印、存档、转发等各种处理。
* 邮件投递速度快、效率高。
* 允许远程非网络用户与网络用户进行通信。

MAIL MONITOR电子邮政软件的主要组成部分及其功能如图6-33所示。其中，MMGR（MAIL MANAGER）是邮政管理员程序，它用来定义用户及其邮件分发表、定义网络类

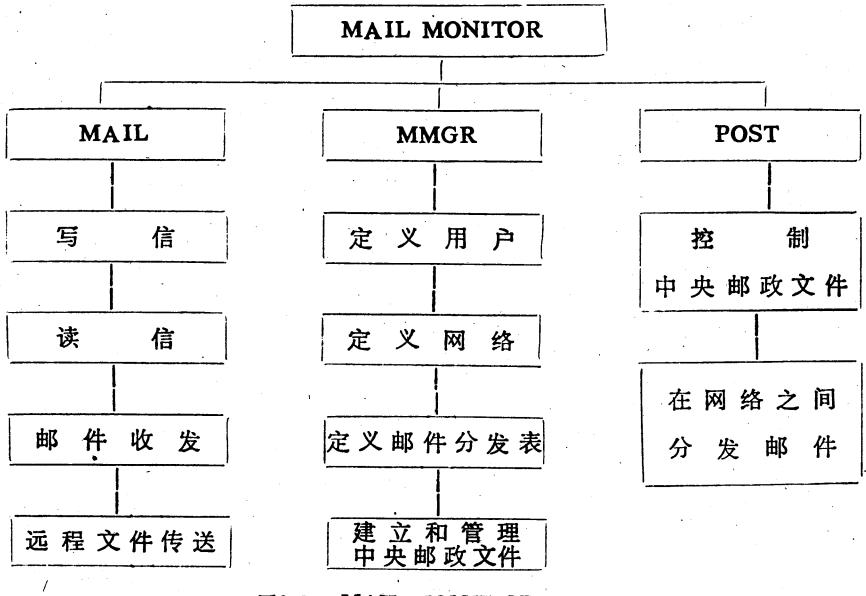


图6-33 MAIL MONITOR的结构与功能

型，建立并维护一个中央邮政文件（POSTOFF.MM）。POSTOFF相当于一个电子邮局，它包含有整个邮政系统中必须的一些信息，如邮箱名、口令、网络名、电话号码、文件大小等等。

MAIL是各用户工作站所使用的软件，它用于邮件的准备、收发及管理。POST程序的作用是与远程网络交换邮件，系统中无此要求时可以不必使用。

图6-34是系统运行过程中邮件的流通示意图。当用户寄出一个邮件（MAIL.PO）后，它被送到邮局（POSTOFF.MM）并保存在那里，等待收件的用户把它取走，或者等待由POSTMASTER程序发送给远程用户或其它网络。邮政管理员定期地对邮局进行管理，例如删除一些无法投递的“死信”等等。

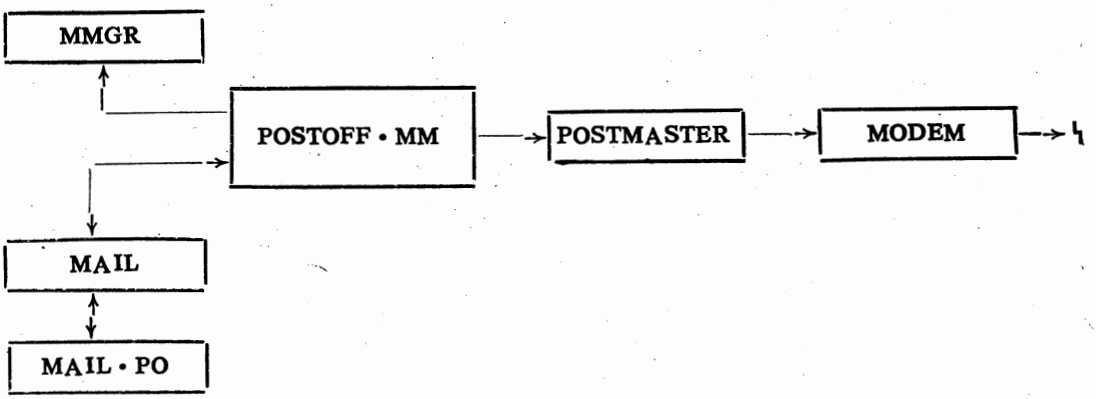


图6-34　电子邮件流通图

（2）发信

经过邮政管理员批准的用户，就可以在网络工作站使用所有电子邮政的各种功能。

为了向其它用户发信，一般需要经过信件准备、发出和检查三个步骤。下面作简略介绍。

用户在工作站上启动MAIL程序运行后，输入发信人的名字（即邮件名，下文假设为Jeff Green）和口令，然后执行“写信”命令，屏幕上将会显示如下“信头”信息：

```
From: Jeff  Green
Subject: _____________________________
Date: 07/20/1985
To: _____________________  _____________________
    _____________________  _____________________
    _____________________  _____________________
    _____________________  _____________________
Encl: ___________________  _____________________
```

其中发信人及日期已由程序自动填好，用户只要补充输入Subject（事由）和收信人名字。

收信人可以不止1个（最多20个）。如果使用预先定义的分发表（所有收信人一览表），则输入分发表名字即可。如果需要随信寄出有关的文件，则应在Encl（附件）栏中填入文件名，可随信寄出的文件不能超过10个。

信头准备完毕后，再使用文字处理软件起草信件正文。信件正文最多60行，允许从已有正文文件中进行复制。如果文件太长，则作为附件寄出。

整个信件准备完毕后，可以在打印机（本地打印机或共享打印机）上打印输出，也可作为文件存档。

接着便可使用发信命令把信发出。如果收信人未在邮政系统中预先定义，则邮局拒收信件，并通知用户收信人有错。

信寄出后，过一段时间用户可以检查该信件是否已被收信人取走，或者还保存在邮局中，这只要输入一个“未投递邮件”命令，MAIL程序就会列出该用户已发出但尚未被收信人取走的所有信件的一览表（包括收信人名、事由、发信日期等），供发信人查询。

（3）收信

电子邮政系统中的用户，只要一启动MAIL程序工作，立即会得知邮局中有几封信等待他取走。执行读信命令（Read）后，屏幕上显示出所有来信的一览表（包括发信人姓名、事由、发信日期等信息），用户可以根据事由的轻重缓急，选择其中的一些信件首先处理。

再次执行读信命令，被选中信件的附件一览表首先在屏幕上显示出来，用户可以视需要把它们保存在自己的盘卷中，然后屏幕上显示信件的内容。读完以后，用户可以把该文件打印出来，也可以转发给另外一个用户。

如果需要转发信件，则执行转发命令。假设原信的发信人为Jeff Green，转信人为Liz Ward，屏幕上显示出：

From: Liz Ward　　Author: Jeff Green
Subject: Computer Training Meeting
Date: 07/21/1985
To:
*** Original letter follows ***
（原始信件内容）

用户只要填入收信人姓名，再执行发信命令，此信即可转发给另一个用户。必要时，转发人可以在转发信件中增加附件或信件内容，使用十分方便。

为供以后查阅之便，每一封信件阅读完毕后都可以存档。在来信很多的情况下，用户还可以有选择地阅读处理信件。选择的条件可以是发信人名字、发信日期和事由。这些条件可以结合若干运算符使用，相当方便有效。

存档信件的处理功能有：列出存档信件目录，按条件检索出需要的信件供查阅，删除一些不必要的信件，打印存档信件，转发存档信件等等。

（4）远程邮件

为了实现远程邮件的收发处理，必须使用MAIL MONITOR的远程网络版本。同时，在OMNINET网络中选择一台装上调制解调器（1200波特或300波特）的PC机，让该机运行

POSTMASTER程序（参看图6-34），使它成为一个专用工作站，负责远程邮件的自动收发操作。当然，远程通信另一端的网络也应具有同样的软、硬件结构。

在进行远程通信之前，首先应由邮政管理员为远程网络和远程网中的用户“开户”。每一个远程网络应在POSTOFF.MM文件中登记有它的名字、网络电话号码以及信件送往该网络的时间（有信即发或在指定时间发信，以便节省电话线路的费用）。每一个远程用户的名字都由用户名以及所在的网络名组成。

用户在工作站上从远程收信以及向远程发信的操作与本地通信的过程完全一样。但是，由于远程邮件是由专用站上运行的POSTMASTER程序接收并通过电话线路转发出去的，因此很易发生错误。为此，POSTMASTER采取了一系列办法：如发信时如果线路忙或者对方无回答，则15分钟后自动重发，并可反复进行五次；如果POSTMASTER无法向远程网发出信件的话，则向所在网络的邮政管理员发出一封报告出错原因的信件，供系统排除故障使用。又如，POSTMASTER通过检查对方来信而确定远程用户是否已经连网，如果未找到用户的话，则把信件发给对方网络的邮政管理员，而决不丢弃任何信件。

4．网络数据库

连接在OMNINET网络上的IBM PC机用户，如果使用原来的dBASE Ⅱ或dBASE Ⅲ数据库管理系统进行工作就不太合适。因为dBASE Ⅱ（或dBASE Ⅲ）只是为单用户环境设计的数据库管理系统，它缺乏多用户同时使用数据库文件时的各种保护措施。为了有效地提高局部网络这样一种多用户工作环境的工作效率，必须为它配置具有数据文件共享和并发存取能力的网络数据库管理系统。

网络数据库系统分成集中式和分布式两大类。前者对系统中的所有数据库文件进行集中存放和集中管理。后者把数据库文件分散在网络各节点机上存放和管理，并允许交叉进行存取，数据的位置和存取控制对用户完全透明。分布式数据库结构灵活、坚固，功效较好，但是它的实现比较复杂。

美国Software Connections公司为PC/OMNINET网开发的LAN：DATASTORE，是一个在局部网络上运行的集中式多用户关系数据库系统。与单用户数据库系统相比，它的主要特点有：

* 允许网络上多个用户同时存取数据库文件。即使有两个以上用户同时都要修改某个数据文件，由于采取了保护措施，因此不会引起数据丢失，保证了数据的准确性。

* 数据库管理员可以根据用户情况控制每个用户对数据库文件的存取权限，因而系统的保密性较好。

* 采取了口令和保存修改记录的措施，提高了系统的安全性。

除了单用户数据库管理系统所具有的各种数据库管理功能之外，DATASTORE为多用户环境提供了如下措施：

① 只有经过数据库管理员授权的用户才能使用数据库。

② 用户使用口令进入系统。

③ 每个用户对数据库文件的存取权限有四级，数据库管理员视具体情况向各个用户分别授权：

可读——只允许查看数据库文件中的记录

可写——允许向数据库文件中添加记录

可修改——允许修改数据库文件的记录

可删除——允许删除数据库文件中的记录

④ 可以限制用户只能对数据库文件中指定字段的内容或满足某种条件的那些记录的内容进行操作。例如，可以规定某部门的用户只能查看属于本部门的那些记录。也可规定，记录中的“工资数额”不得由一般用户查看等等。

⑤ 数据库管理员通过一张“修改记录表”，可以了解到各用户对数据库文件所执行过的各种修改操作。修改记录表中包含数据记录最后一次被修改的日期、时间和修改者的名字。

图6-35是DATASTORE主要功能的示意图，供读者参考。

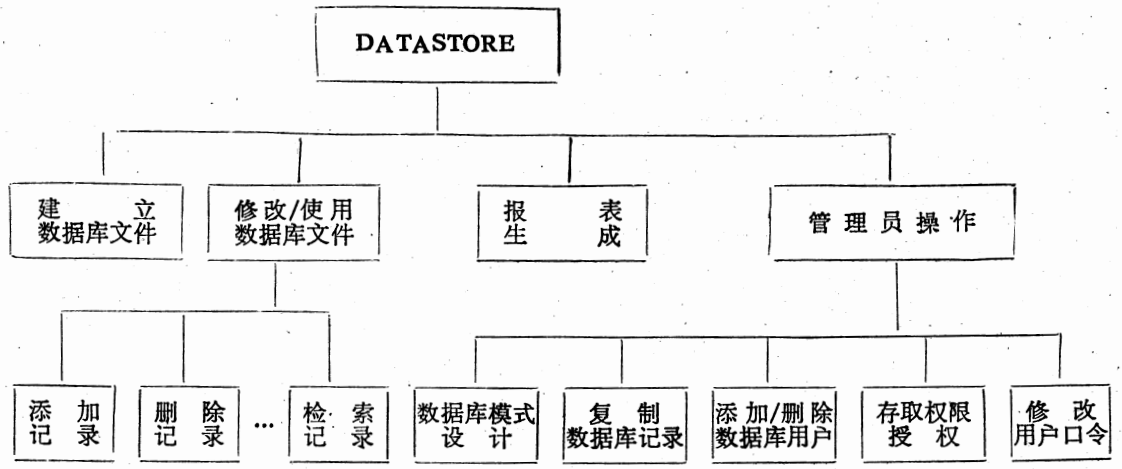


图6-35 网络数据库DATASTORE的功能结构

此外，Software Connections公司在DATASTORE的基础上，还提供了一个多用户关系式数据库的应用开发工具DATACORE。它用PASCAL语言编程，使用起来相当方便，可以大大减少网络数据库应用程序的开发时间。

第五节 IBM PC/ETHERNET局部网

Ethernet局部网又叫以太网，它是美国XEROX公司在1975年为小型机研制成功的第一个总线竞用式局部地区网络。1980年9月，XEROX公司与DEC和INTEL公司共同提出了以太网规范，成为局部网络产品的第一个规范。1982年，美国3 COM公司为IBM PC推出了组成以太网的软、硬件产品——Ether Series。近年来，不少工厂生产以太网的产品，但大都不能保证互相兼容。

本节以3 COM的Ether Series为背景，介绍以IBM PC为节点机的以太网的网络硬件、网络管理软件和网络应用软件的原理与使用。

1. PC/Ethernet局部网的结构与原理

（1）Ethernet网的基本配置

Ethernet网的标准配置如图6-36所示，它包括下面一些主要硬件成分。

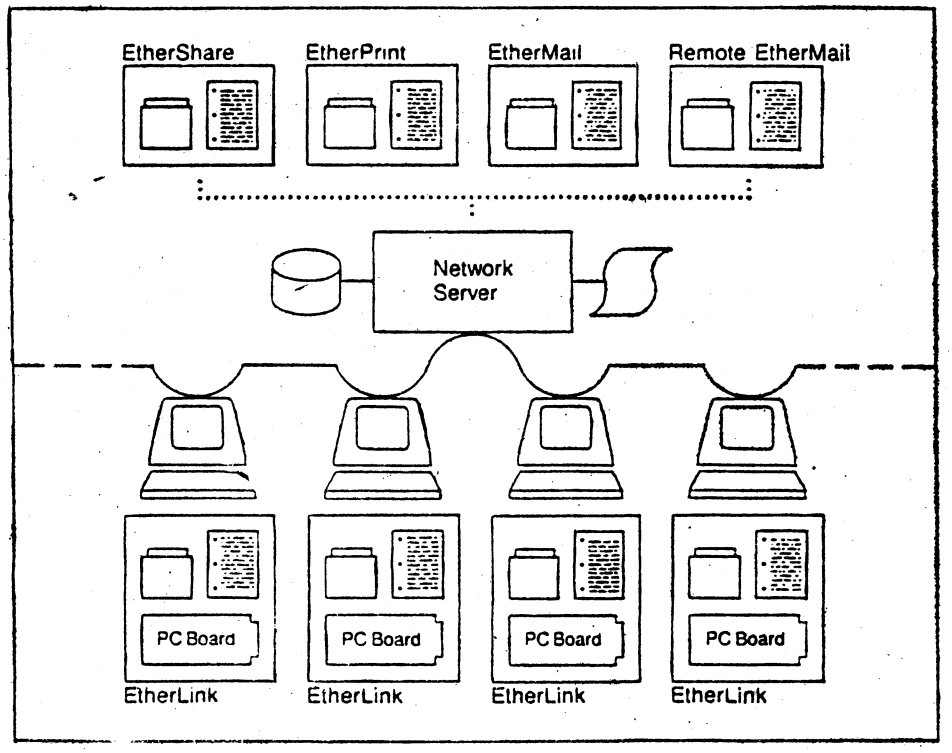


图6-36 以太网的结构

① 以太连接卡（Ether Link）。它的作用相当于OMNINET网中的传输器，每台节点机均需配置一块。

② AP网络服务器（Network Server）。它带有两个30MB的硬盘，还可以带有后备匣式磁带机。网络服务器的资源可由网络各节点共享。一个AP服务器大约可以支持40个用户。

③ 3 Server网络服务器。它带有1—6个36MB的硬盘存贮器和60MB的磁带后备存贮器，可以同时支持50个以上工作站。

④ IBM PC/XT也可用作网络服务器，但其硬盘容量小，处理能力较低，只能支持2—8个网络用户。

一个Ethernet网可以安装多个服务器，甚至可以采用VAX小型机作为服务器，使网络性能大大提高。

⑤ 细型以太网电缆。这是阻抗为50欧姆的标准同轴电缆，它作为以太网的网络干线使用。通过BNC接头转接，网的最大范围只有304米，设备到网络干线的最大距离为1米，比较轻巧、便宜且易于安装。

⑥ 粗型（标准型）同轴电缆。阻抗为500欧姆，也作为以太网的网络干线使用，但需采用DIX（Digital-Intel-Xerox）系列接头作连接，并需要有一个外部收发器和一条收发器电缆。粗线网允许的最大电缆长度可达1000米，节点机与干线最大距离为50米。粗、细两种电缆可以混合使用，但需要通过专用的转接插头座来连接。

Ethernet网络需要配置的软件主要有：

① Ether Share　这是以太网上的资源共享管理软件，它允许多个用户共享硬盘和其中的数据、文本及程序，并由用户软件和服务器软件两部分组成。根据网络上使用的服务器的类型，服务器软件分成Ether Share/PC服务器软件和Ether Share/AP服务器软件。安装不同的服务器就应使用不同的Ether Share服务器软件。

② Ether Print　这是共享打印机的管理软件，它允许网络上的用户可以同时使用服务器上的打印机。其服务器程序可分成Ether Print/PC和Ether Print/AP等多种，分别在不同的服务器中使用，它们均能支持服务器上的两台打印机作为网络上的共享打印机使用。

③ Ether Mail　这是电子邮政管理软件，它允许网络上的用户方便地进行数据通信。其服务器软件部分的作用相当于“邮局”，而用户软件部分使得每个工作站的PC相当于是一个“邮箱”，可以建立、分发、检索和显示各种邮件。根据服务器类型的不同，服务器软件也有不同的版本。

④ 远程Ether Mail　这个软件允许PC机的用户在距离网络远处的场地上，通过调制

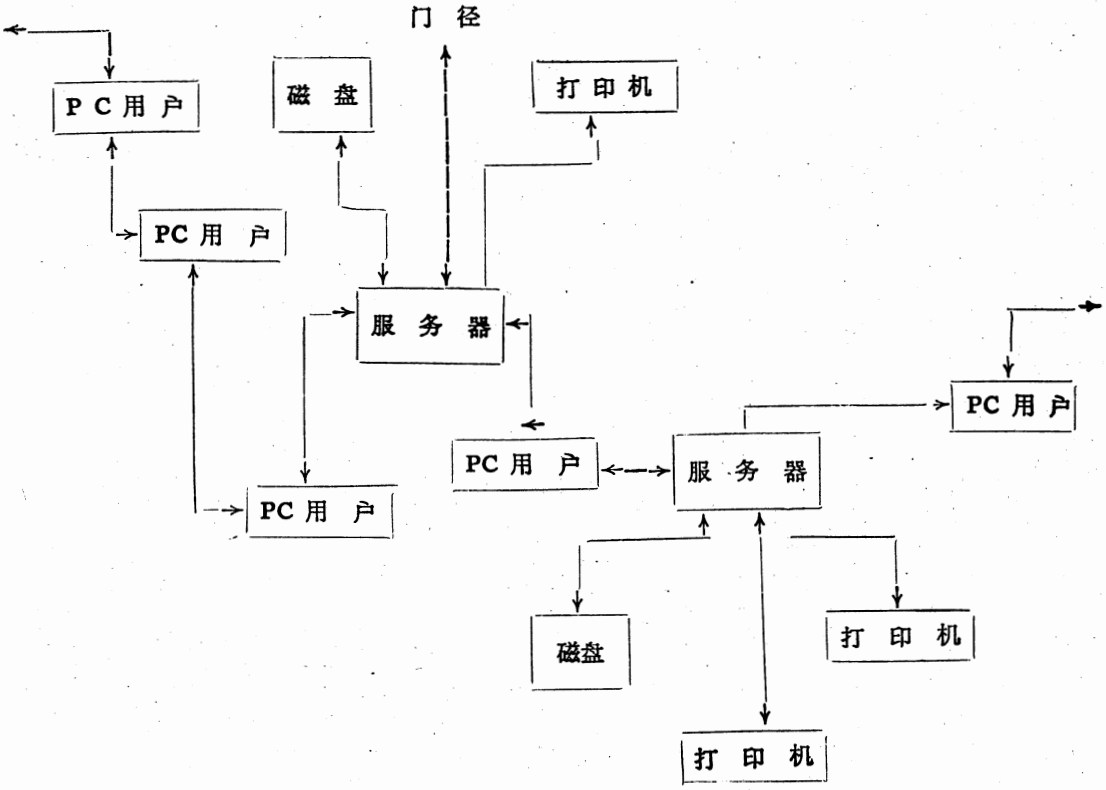


图6-37　中等规模以太网结构示意图

解调器和电话线路，向网上的其他用户发送和接收邮件。它由用户软件和AP服务器软件两部分组成。

⑤ Ether Start　这是插在以太连接卡（Ether Link）上面的一个固件，它允许工作站中的PC机可以不要使用软盘驱动器，因而既降低了成本，又提高了网络中数据的安全性。

⑥ Etherterm　用以将网络上工作站改变为智能终端，以便与大型机联网使用。

⑦ 网络应用软件　如表格处理软件Visicalc Ⅳ、文件处理软件Visiword等，供网络中各用户使用。

与OMNINET一样，网络上还可以安装各种后备存贮器，也可以安装与异种网络互连的门径服务器，以便与其它网络互连。规范较大一些的以太网可以划分成若干段，每段长1000米，可支持大约100个节点，段与段之间用中继装置（REPEATER）沟通。一个网最多可以有10个段，网络总长度可达10公里。图6-37是一个中等规模以太网的结构示意图。

（2）以太连接卡

3COM公司的以太连接卡（EtherLink）是以太网的通信处理机，它插在PC机的扩充槽内，把节点计算机与网络互连起来。图6-38是以太连接卡的原理示意图。从图中可以看出，它主要由三个部分组成。

① 以Z80CPU为核心的主控制器，它用来实现收发数据的打包、拆包、网络访问控制、地址识别、接口内部数据传输控制以及与高层软件接口等各种控制功能。

② 通道逻辑，用来提供串/并和并/串转换、信息的调制及解调、CRC码的生成及校验、冲突检测等功能。这一部分实际上是由超大规模集成电路（以太网数据链路控制器）所实现的。

③ 发送/接收器（Transceiver），用来完成电缆驱动及信息接收，是网络的电气连接部分，包括T型电缆接头等。

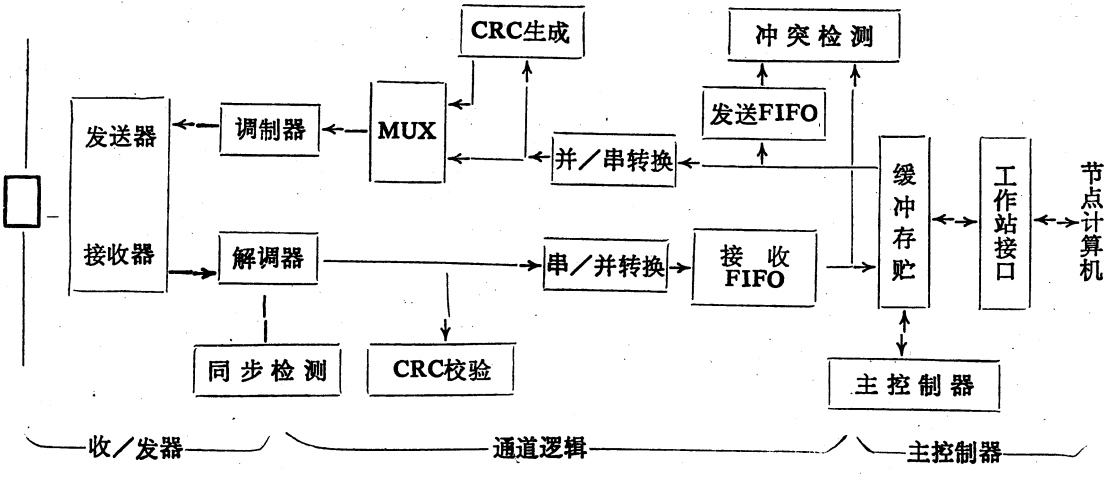


图6-33　以太连接卡的原理图

连接卡在插入IBM PC/XT机内时，必须对粗细电缆选择插头（BNC/DIX）、数据传送的DMA通道号、中断号进行设置。通常DMA为1#，中断信号为INT3，输入输出端口地址为300－30F。但用户也可以根据自己的情况，通过跨接插头加以改变。

连接卡在发送数据时，首先将数据从缓冲存贮器读出，经过并/串转换，再经过CRC码生成器，然后加入同步代码成为报头，形成完整的一帧信息，最后通过接收/发送器发送到电缆上。在发送信息的同时，接收/发送器也从电缆上接收信息，并立即对发送信息与接收信息进行比较。若相等，则表明无访问冲突；若不相等，则说明总线上有访问冲突，必须停止发送。

接收数据时，先由同步检测电路对同步信号序列进行检测，然后经CRC校验确认正确无误后，再由串/并转换电路将一帧信息分解，并把其中有效数据送到缓冲存贮器，于是就完成了接收过程。每次正确接收一帧信息后，便由连接卡发送出一个ACK（认可）信息包。若接收有错，则发送一个NAK信息作为对发送站的回答。

（3）Ethernet的网络协议

Ethernet局部网在网络协议上，只规定了OSI七层模型中的最低两层，即物理层和链路层。而Ethernet的高五层均称为委托层（Client layer），由制造厂自行规定。所以，不同的Ethernet网络产品，相互不能在高层协议上兼容。

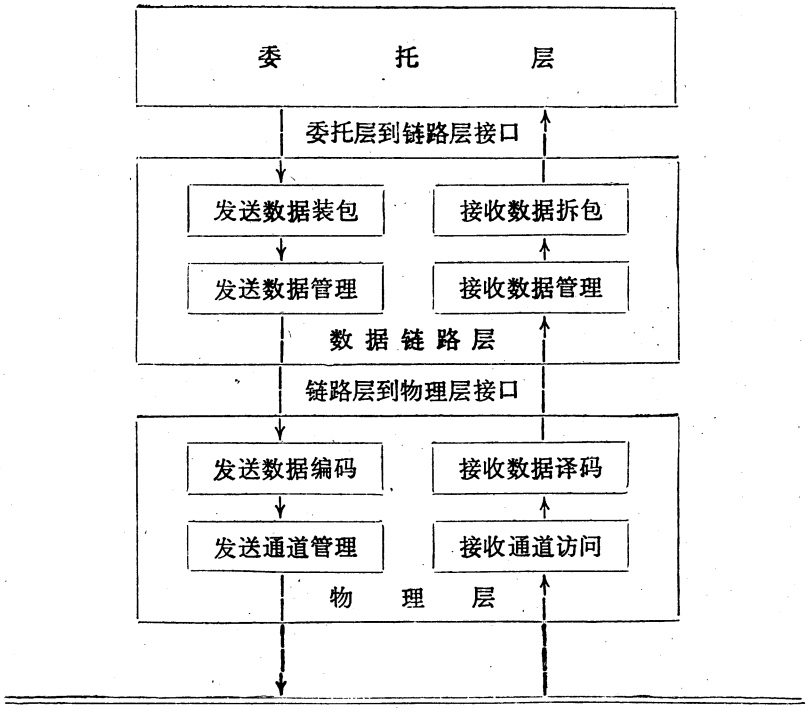


图6-39　Ethernet结构分层示意图

Ethernet网络的结构分层如图6-39所示，其中物理层和链路层的功能如下。

①物理层。它又分成两个子层：

* 通道访问子层。其功能为码位的接收和发送；载波检测，查明传输线上有无信号传输；冲突检测，检查传输线上是否发生访问冲突。

* 数据编码子层。其功能为添加和删除用于同步的报头信息，以及对信息位进行调制与解调。

② 链路层。它也分成两个子层：

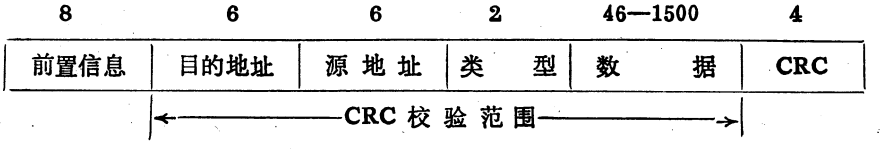
* 链路管理子层。其功能为处理访问冲突和分配传输通道，例如线路侦听、帧间间隔、冲突检测和强化、冲突后退和重发等等。

* 数据封装子层。把信息划分成帧，对源地址和目的地址进行处理等。

Ethernet网络采用CSMA/CD访问控制方式，其原理已在第一节中作了简要说明，这里不再重复。

（4）信息包格式

在以太网上传输的信息包的格式如下：



其中，八个字节的前置信息由62位相间的“1”、“0”信息加上两个连续的“1”共64位组成，用以指示正文信息的到达。目的是为了保证载波检测能稳定可靠地进行。

目的地址采用了48位的长地址，这是一种所谓“唯一地址”的编址策略，即任何一个Ethernet网络中的任一工作站，它们的物理地址都互不相同，因为48位地址码共有100万亿以上不同的地址可供选用。这样，当工作站从一个网络移到另一个网络时，工作站地址不需要改变。另外，“唯一地址”对跨网寻址也是很有利的。

当地址码中第1位为“0”时表示物理地址，即某个特定工作站的站地址；当第1位为“1”时，表示多向地址，表示信息应发送给网络中一组工作站。当48位地址码全为“1”时，表示广播地址，表示网络中所有工作站均要接收该信息包。

源地址是发送本信息包的工作站地址，供连接卡执行确认应答等操作时使用。信息包中的“类型”共两个字节，其内容不作规定，由委托层进行设置和解释。数据段的长度最少为46个字节，最多为1500个字节，内容不限，由高层软件处理。

四字节的CRC字段用来对信息包进行校验，它的生成多项式为：

$$G(X)=X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$$

信息包的发送过程大体如下：在发送信息之前，每一个工作站先进行载波线路的检测，因为只有在线路状态为空闲时才允许发送。由于信息发送需要一段准备时间，如果这时有另外一个工作站，因侦听到线路空闲也开始发送信息的话，就会出现“碰撞”。若一旦发生冲突，发送站并不立即停止发送，而是先送出至少32位（但不超过48位）的任意序列码来加强冲突，使所有发送站均能检测出线路上的访问冲突。前一次发送因冲突而终止后，不允许马

上再发送，而要延迟一段时间后再作试探。延迟间隔由所谓的“二进制指数补偿算法”来决定，其公式为：

$$t_W = 2a2^{i-1}$$

其中i表示第几次重发。可见失败次数越多，表示线路越忙，因而等待时间也就逐渐增加。这种发送方法称为Ethernet网络的原始工作方式。除此之外，还可以使用回答式或预约式方法来发送信息，其性能比原始方式有所改进。

每一个信息包在接收过程中必须进行信息包判定（判定包的边界和长度是否正确）、地址识别、CRC校验等处理，然后“拆出”包中的数据送交委托层继续处理。

在接收过程中还包括接收链路管理，主要目的是为了滤去从网络上收到的各信息包因冲突而产生的“碎片”。Ethernet网规定最小有效信息包长度为64字节，凡是小于64字节的信息均认为是“碎片”而被滤去。

2. 以太网管理程序——EtherShare/PC服务器程序

以太网上的硬盘及其服务器，可以使用专用的AP服务器，也可以使用PC/XT机来承担其任务。使用服务器之后，可以使得网络上的节点机共享服务器磁盘以及磁盘中的程序和数据，而且在专门的软件（EtherPrint和EtherMail）配合下，还可以共享打印机并允许用户之间相互交换信息。网络上的服务器可以安装多台，但每台服务器都必须在共享资源的管理程序（EtherShare服务器程序）作用下才能发挥服务器的作用。由于EtherShare/AP和EtherShare/PC的功能、原理和操作几乎完全相同，因此下面就以EtherShare/PC服务器程序为例进行介绍。

（1）EtherShare/PC服务器程序的基本操作模式

用作以太网服务器的IBM PC/XT机除带有硬盘外，至少应该有256KB内存。如果该机还要运行应用软件，则内存应大于384KB。

PC/XT作为服务器使用时，必须运行共享资源的管理程序EtherShare/PC。这时，服务器可以有两种不同的工作模式：标准模式和专用模式。两种模式可以任意选择，需要时还可以相互切换。注意在切换模式之前，所有用户都必须从网络上撤出。

① 标准方式（Standard mode）

用作网络服务器的IBM PC/XT机处于标准模式工作时，它既可以作为网络用户共享的硬盘使用，也可以作为一个普通节点，象网络上的其他用户一样，运行自己的应用程序。当然，由于该机担任了网络服务器的工作，因此不能任意关机，以免影响其他用户的正常操作。

在标准模式下，PC/XT服务器仅能支持一台共享的硬盘，且不支持EtherMail电子邮政软件的运行，服务器的许多管理功能也无法实现。

② 专用模式（Dedicated mode）

处于专用模式的PC/XT网络服务器，它完全致力于网络上的各种共享资源服务，不能再同时作为一台PC节点机来使用。

专用模式下的服务器可以支持15个共享硬盘，可以运行包括电子邮政软件在内的全部应用程序。

由于两种模式共享硬盘的能力不同，在进行模式切换时，必须考虑到有关文件的位置所在，以免出错。

与OMNINET网不同，一个Ethernet网络可以有多个PC/XT网络服务器，且可以放置在网络的任何地方。不同的服务器允许各自在不同的模式下运行。除了在标准模式工作状态下的服务器，作为节点机使用时不能访问其它服务器之外，网络用户在联网后可以访问网络上任何服务器中的程序和数据。这样就为用户提供了有效的分布式资源环境。

（2）EtherShare/PC服务器程序的安装

共享网络资源的管理程序EtherShare/PC由服务器软件和用户软件组成。其中，服务器软件在网络服务器上运行并对共享资源进行管理，允许系统管理员对网络状态进行监督和检测。而用户软件则在工作站PC机上运行，它使用户可以共享网络中软、硬件资源。

EtherShare/PC服务器软件包括两张软盘片，它们只与一个以太连接卡相对应，一旦安装好之后，就不能再与别的连接卡配合使用。

服务器软件具有两方面的功能：一方面是管理网络上各PC机对访问服务器硬盘或打印机的请求；另一方面是用于执行网络的管理职能，例如服务器的配置与维护等等。

把IBM PC/XT安装成为网络服务器的主要过程如下：

① 将DOS2.0或2.1版软盘片插入驱动器A并打入命令：

```
A>FORMAT C:/S/V
A>COPY *.* C:
```

这样，MS-DOS就在PC/XT的硬盘中了。

② 把EtherShare/PC用户软件的盘片插入A驱动器，打入命令：

```
A>COPY ES.COM C:
```

把文件ES.COM复制到C盘。

③ 从硬盘上删除$$AUTO.BAT、$$CONFIG.SYS、$$CONFIG.DED、$$CONFIG.STD等四个文件（如果它们存在的话）。

④ 将EtherShare/PC服务器软件的第1张盘片（去掉写保护）插入A驱动器，打入命令：

```
A>INSTALL
```

当出现指示信息时，将EtherShare/PC服务器软件的第2张软盘插入A驱动器，再按〈SPACE〉键，若安装成功，则屏幕上将显示安装完毕的有关信息。

为了使该服务器投入使用，还必须继续进行下列操作。

⑤ 重新启动系统（Ctrl-Alt-Del），在输入日期和时间后，屏幕上显示服务器程序的主菜单如下：

```
1—START THE DEDICATED MODE SERVER
2—SWITCH TO STANDARD MODE
3—START THE ADMINISTRATIVE FUNCTION
4—EXIT TO DOS WITHOUT STARTING SERVER
```

⑥ 选择“3”（管理）并按下“ESC”键，于是出现管理操作的菜单如下：

```
1—INSTALLATION
2—MAINTENANCE UTILITIES
```

3—DONE

选择“1”（安装），屏幕上显示出安装操作的菜单如下：

1—REGISTER ETHERSHARE
2—DE-REGISTER ETHERSHARE
3—INSTALL NETWORK APPLICATION
4—INSTALL EXPANSION PERIPHERAL
5—DISPLAY ETHERSHARE NAMES
6—DONE

为了把新的服务器在网络中注册，选择“1”（注册），于是屏幕上会显示出网络上所有已注册的服务器名。此时，打入要注册的新服务器名即可。

⑦ 注册完毕，选择“6”，返回管理操作菜单，再按〈ESC〉键退出。

⑧ 系统再次重新启动，屏幕上又显示出网络服务器程序的主菜单。

这次选择“1”（启动专用模式），于是屏幕上显示出服务器程序在专用模式下运行时的初始画面，屏幕上方出现的是该服务器注册的名字（NANJING）。

NANJING
is
NOW IN OPERATION
as a
3Com EtherSeries Network Server

⑨ 建立系统盘卷。它是一个公共盘卷，其中存放着所有网络用户共享的软件。建立系统盘卷的步骤是：

* 在一台已经连网的用户PC机上，使用Etherseries/DOS盘片启动PC机工作（Etherseries/DOS盘的生成见下一小节）。

* 用服务器名（NANJING）作为用户名，打入下列命令进行用户登录：

A>ES LOGIN NANJING

* 打入盘卷建立命令，在服务器上建立一个名字为SYS2、容量为500KB的盘卷：

A>ES CREATE SYS2/500KB

* 把驱动器D与盘卷SYS2连接起来：

A>ES LINK D:SYS2

* 把MS-DOS的文件从A盘复制到系统盘卷SYS2中：

A>COPY *.* D:

* 使用同样的操作，再把Ethershare/PC、EtherPrint/PC和EtherMail的用户软件盘片中的所有文件复制入SYS2中。

* 把盘卷从驱动器D上卸下，打入命令：

A>ES UNLINK D:

* 通过修改命令把SYS2盘卷转换成公共盘卷，打入命令：

A>ES MODIFY SYS2/PUB

至此，系统盘已建立完毕。

⑩ 最后，可以用下列命令为每个用户建立一个用户名：

A>ES UCREATE USER1

当新用户登录后，若要使用共享盘卷，只要使用“ES LINK”命令把SYS.SYS2盘卷连接到某个驱动器即可。这样，网络用户就可以使用共享的系统盘卷了。

（3）服务器的网络管理功能

服务器程序安装完毕并在服务器PC/XT机上启动运行之后，以太网上的其它节点机就可以各自运行并共享网络资源了。如前所述，服务器在标准模式下工作时，除了负责处理网络上各种资源共享的管理之外，还可以使服务器兼作一台节点机使用。标准模式下服务器的操作及功能如图6-40所示。

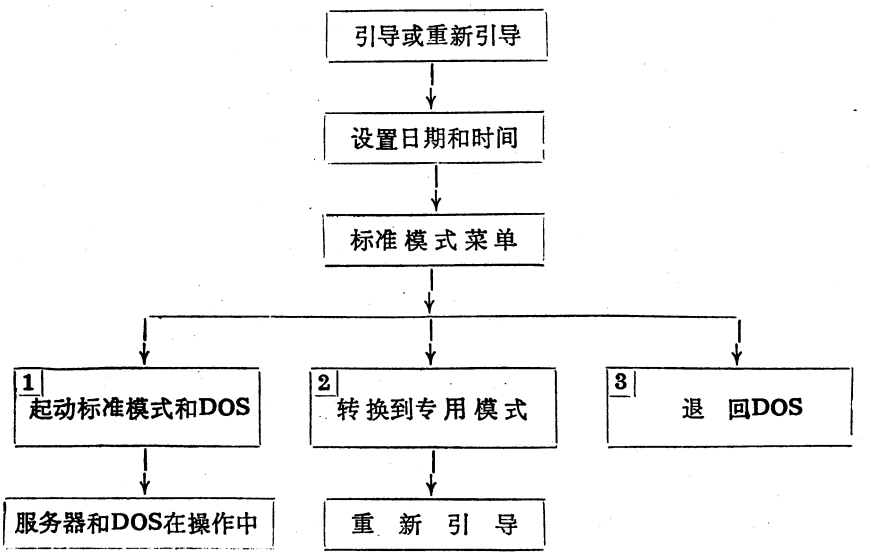


图6-40 标准模式下服务器的功能与操作

如果把服务器切换到专用模式，服务器虽然不再能兼作节点机使用，但它除了负责处理网络上各种共享资源的请求之外，还能进行整个网络的控制、状态显示、软/硬件安装和维护等各种管理操作，其功能和操作的过程大体如图6-41所示。有关功能将在下面进行解释。

3．以太网的用户操作

（1）基本概念

连接在以太网中的每台PC机，只有通过运行EtherShare/PC用户软件，才能访问网络中连接在服务器上的共享硬盘。

服务器上的共享硬盘被分成许多盘卷，盘卷的容量为64KB—32MB，它可以被看作PC机上新增加的软盘。

通过使用EtherShare/PC的用户软件，用户可以在共享硬盘上建立和使用盘卷，并控制盘卷的访问权。生成的新盘卷要赋予盘卷名和口令。

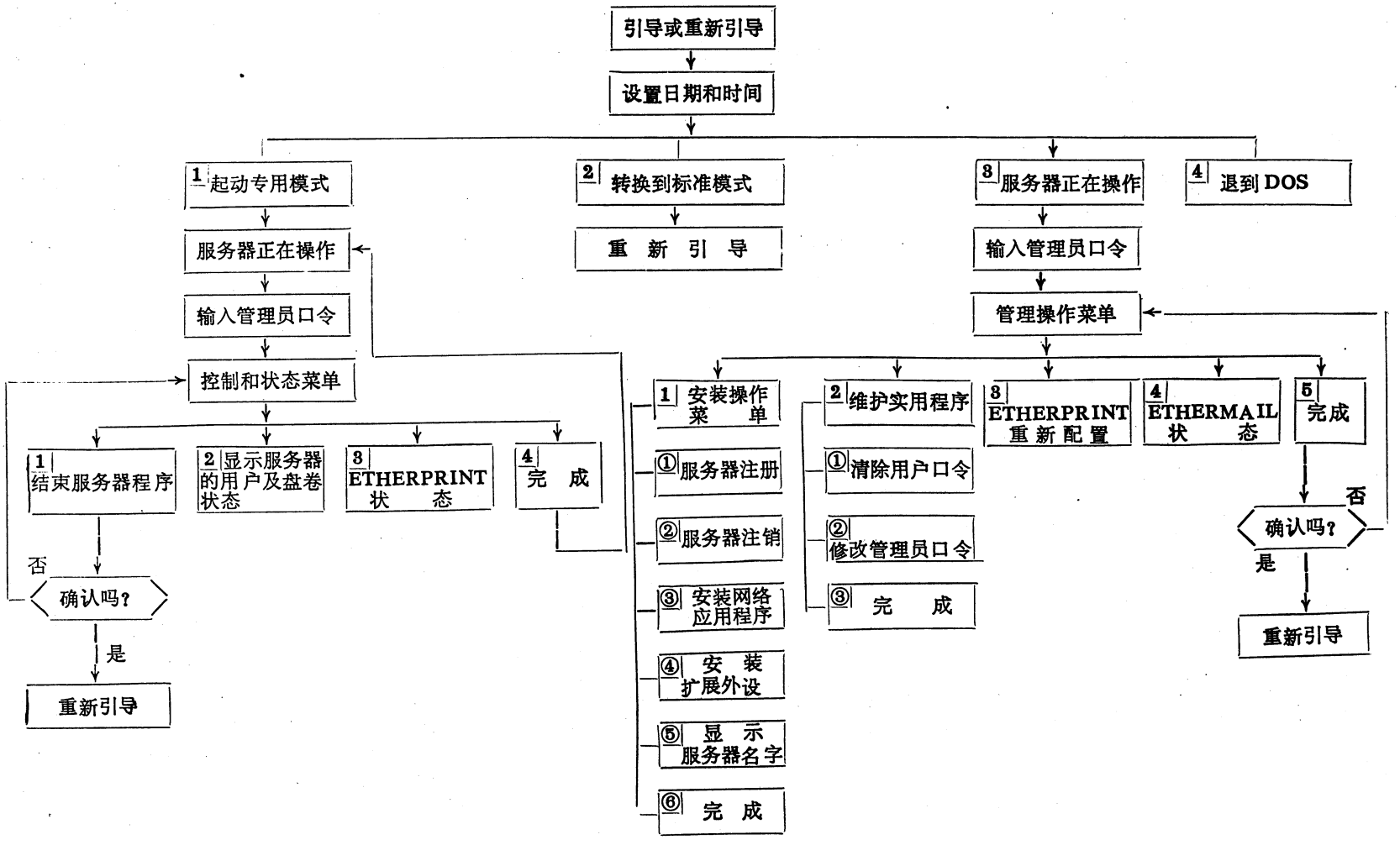


图6-41 专用模式下服务器的功能与操作

以太网的盘卷分成私有、公共和共享三种类型，它们的特性如表6-11所示。

表6-11　以太网的盘卷类型及特性

卷类型	特　　性	设置口令	不设置口令
私有型（PRIV）	• 可读／写 • 一个时间内仅一个用户使用	全部用户（建卷用户除外）必须提供口令	仅建卷用户可访问该卷
公共型（PUB）	• 只读 • 多用户可同时访问	同　上	任何用户均可访问该盘卷
共享型（SHAR）	• 可读／写 • 多用户同时访问	同　上	仅建卷用户可访问该卷

用户要使用网络中的共享资源，首先必须使用ES LOGIN命令和已注册的用户名登录入网，有时还需回答一个口令。当以太网存在着多个服务器时，每个服务器都有自身的一张联机用户表。用户可以从网上任何一台PC机上通过任何一台服务器的管理联机入网，而不必局限于某一台特定的PC机。并且，用户还可以访问任何一台服务器上的盘卷。若盘卷设置有口令时则必须提供此口令。

每台服务器在注册入网时已建立了一个服务器名，用户可以使用“ES SDIR”命令显示出网络中全部服务器的清单。

服务器在注册后，系统自动产生一个和该服务器同名的用户，它便作为在新服务器上建立新用户时登录入网的用户名。它也可以用SYS这个用户名来替代。每个服务器都包含一个名叫SYS 2的公共盘卷，它归用户SYS使用，也可以被该服务器上其它用户作为SYS.SYS 2来引用，这就是“系统盘卷”。系统盘卷包含有DOS 2.0和网中各用户所需要的其它网络软件（如EtherMail等）。

盘卷必须通过连接命令（LINK）安装到节点机上的某个驱动器中才能使用。安装操作允许把任何一个服务器中的盘卷安装使用，这种功能叫做“服务器的交叉连接”，它对于使用网络中的公共文件特别有用。例如，EtherMail的邮件分配表可保留在某个服务器上名为FINANCE.DIST的盘卷中，网上全部用户均可使用。

以太网的用户进行各种资源共享和数据通信操作，都必须在网络操作系统EtherSeries/DOS下进行。EtherSeries/DOS是在原MS-DOS的基础上扩充若干网络实用程序而成的，具体的生成过程如下：

将MS-DOS盘插入驱动器A中，将以太网的用户软件盘片插入驱动器B中，执行命令

A＞B：SETUP

再使用“DEL SETUP.BAT”命令把A盘中的SETUP.BAT文件删去，于是驱动器A中的盘片就是网络操作系统EtherSeries/DOS了。

如果要把EtherSeries/DOS建立在硬盘上，则必须在驱动器A中插入以太网用户软件盘片，打入下列命令即可：

C＞A：FDSETUP

网络操作系统EtherSeries/DOS中所包含的文件有两部分，一部分是网络实用程序，一

部分是原MS-DOS文件，它们的具体内容如下：

EtherLink Files:	DOS Files:	
ES.COM	ANSI.SYS	FIND.EXE
EP.COM	ASSIGN.COM	FORMAT.COM
CONFIG.SYS	BASIC.COM	GRAPHICS.COM
ENET.SYS	CHKDSK.COM	MODE.COM
LOGIN.BAT	COMMAND.COM	MORE.COM
PRINT.BAS	COMP.COM	PRINT.COM
	DISKCOMP.COM	SORT.EXE
	DISKCOPY.COM	SYS.COM
	EDLIN.COM	TREE.COM

（2）以太网的用户操作

在EtherSeries/DOS管理下，以太网上的每一个节点计算机，除了提供原有的所有单机操作功能之外，还可以进行下列网络操作。

① 登录入网和出网

要进行各种网络操作，首先必须登录入网，操作完成后则必须从网络中撤出，这是使用下述两条命令来完成的。

* 登录（ES LOGIN）

用户登录入网使用“ES LOGIN”命令及自己的用户名字和口令，但用户必须预先已在网络上“开户”，否则无法登录入网。如前所述，网络上每一个新的服务器注册入网时，都自动产生了一个与服务器同名的用户名。因此使用该用户名登录之后，就可以为其它新用户开户了。

用户登录入网成功后，屏幕上会给出该用户使用的服务器名，以及所在节点机可以安装盘卷的所有驱动器号码。

* 撤出（ES LOGOUT）

此命令把所有已安装的盘卷从用户所在节点机上拆下，然后从网络中撤出，这样，该节点机就只可使用单机操作命令，而不能再使用网络操作命令了。

② 用户管理

用户管理主要是为新用户开户，老用户除名，修改用户口令，显示用户目录等等。

* 为新用户开户（ES UCREATE）

任何一个老用户均可以在网络的同一个服务器中建立一个新用户，但用户名不能与其它用户重复。

* 删除老用户（ES UERASE）

不再需要使用网络任何资源的用户可以从网络中除名，但在此之前必须预先删去属于该用户的所有盘卷。

* 显示用户目录（ES UDIR）

使用“ES UDIR”命令可以列出网络中所有已开户的用户名字，也可以看到当时已登录入网（正在网上运行）的那些用户名字。下面是显示出来的用户目录的一个例子：

```
A > ES UDIR
Server MARKET:
    BILLS
    BOBP
    FREDH
    MARYS       logged in
    MIKEP       logged in
Server SALES:
    CAROLR      logged in
    JOHNH
    NANCYV
    STEVEP
    WHNDYK      logged in
```

* 修改用户口令（ES UMODIFY）

此命令允许用户自己设置或修改一个口令。

③ 盘卷管理

使用下述命令可以生成、删除和修改盘卷：

* 生成新盘卷（ES CREATE）

生成新盘卷的过程如下：

```
A > ES CREATE?
  Name? VOLUME1      回答新建盘卷名
  Password? ( )      新盘卷的口令
  Size? /2           设定盘卷大小
                         /1为单面（160KB）盘卷
                         /2为双面（320KB）盘卷
                         /numKB为numKB盘卷（范围
                         为64—32000）
```

参数指定后，就在服务器上生成一个新盘卷，新盘卷一定是私有型盘卷。

* 修改盘卷（ES MODIFY）

盘卷修改命令允许把已有盘卷的名字、口令和类型进行修改，操作过程如下：

```
A>ES MODIFY?
    Volume? VOLUME1        打入想要修改的盘卷名
    New name? VOLUME2      新盘卷名，只按回车时盘卷名不变
    New Password? ( )      新口令，只按回车时保持原口令
    New access? /PUB       新的盘卷类型
    VOLUME1 modified
```

注意，盘卷的大小是不能修改的。另外，当一个盘卷已安装在某驱动器上时，也不能进行修改。

* 删除盘卷（ES ERASE）

不再使用的盘卷可以通过删除命令从服务器硬盘上删去，以便释放磁盘空间另作他用。

④ 盘卷的安装和拆下

盘卷相当于一张软盘片，它必须在使用前装上某个驱动器，用完后再从驱动器上拆下，这是使用LINK和UNLINK命令来完成的。

* 盘卷安装（ES LINK）

ES LINK命令用来把一个盘卷“装上”节点机的某驱动器，操作如下：

```
A > ES LINK
    Drive id? E:              被安装的逻辑驱动器号
    Volume? VOLUME 1          盘卷名
```

如果盘卷设置有口令时，安装过程中必须回答正确的口令，否则安装不会成功。

* 盘卷拆下（ES UNLINK）

盘卷不需使用时，可从驱动器上拆下，以便该驱动器再安装其它盘卷。如果用户从网络上撤下（LOGOUT），则所有盘卷均自动从该节点机的驱动器上拆下。

⑤ 批处理文件LOGIN

网络操作系统EtherSeries/DOS中包括了一个批处理文件LOGIN，它可以协助用户自动地完成常用的登录入网操作，其功能为：

* 按用户名登录入网。
* 把系统盘卷SYS.SYS 2安装在驱动器D：上，并把缺省驱动器选择为D：。
* 与服务器的第1台打印机建立连接关系。

该批处理文件的内容如下，其中有些命令或参数在下文中还会介绍。

```
ECHO  OFF
ES LOGIN %1; LINK D:SYS.SYS 2/NP
IF ERRORLEVEL 1 GOTO FAIL
D:
EP LINK/NP
:FAIL
```

（3）网络操作命令

以太网供用户使用的命令共14种，它们的主要功能前面已作了简要叙述。这些命令的操作使用方式有菜单式和命令式两种。如果用户仅打入“ES”命令，则屏幕上显示出全部14种网络命令供用户选择使用。命令方式时，允许用户在命令后面紧跟以必要的参数，也可以等待机器给出提示信息后再回答有关参数。表6-12是所有14种网络操作命令的一览表。

（4）信号量与文件的加锁/解锁操作

如上所述，以太网可以允许多个用户共享服务器的同一文件。多个用户同时读一个文件是完全允许的，并且也不会引起什么问题。但是如果多个用户同时修改一个文件则十分危险。为了确保任何时刻网络上都只有一个用户修改指定的某个文件，就必须采取一定的措施。

为了防止多个用户同时修改共享卷中的某个文件或文件中的某些记录，用户修改文件时必须对文件或记录加锁，只有在修改操作完成之后才能解锁。加锁和解锁操作都是用户程序

表6-12　以太网用户操作命令

命　令　名	功　　能	参　　数	参　数　含　义
ES CREATE	建立一个新的EtherShare盘卷，并为之指定格式	盘 卷 名 〔(口令)〕 〔盘卷大小〕	所建盘卷的名字。名字最长为8个字符，同一用户的盘卷名不得重复，不同用户可以有相同的盘卷名 口令是用来控制盘卷使用的。除所有者外，必须使用口令才能访问盘卷。在指定口令时，还必须指出是私有盘卷(/PRIV)，还是共享盘卷(/SHAR) 盘卷大小可以是64KB到32MB
ES DIR	列出 EtherShare盘卷	〔用户名.〕盘卷名 /L /P /W	盘卷的名字。如果自己不拥有该盘卷，则必须在用户名处给出所有者名。若不给出盘卷名时将列出用户所有的盘卷 显示连接到本节点机的盘卷和它们所连接的驱动器名 当显示满一屏时暂停 给出本开关时每行列出多个盘卷名，否则每行只列一个
ES ERASE 或ES DEL	删除一个EtherShare盘卷	盘 卷 名	欲删除的EtherShare盘卷
ES HELP	提供有关EtherShare命令的信息	〔命令名〕	欲获得有关信息的EtherShare命令，如果不指出命令名时，将显示所有EtherShare命令的清单和它们的功能简介
ES LINK	建立IBM节点机驱动器名与EtherShare盘卷的连接	驱 动 器 〔用户名.〕盘卷名 〔(口令)〕	指定欲连接的驱动器，根据节点机的构成，可以是C：，D：，E：，F： 欲与之连接的EtherShare盘卷名。用户名是所有者的名字，如果该盘卷不属于本用户，则必须指定所有者名 给EtherShare盘卷定的口令
ES LOGIN	用户登录入网	用 户 名 〔(口令)〕	用来标识EtherShare用户的名字，该名字由ES UCREATE设置 由ES MODIFY命令给用户名加的口令，口令在使用时必须用括号括起来
ES LOGOUT	用户从网上撤出	无	

（续表6-12）

ES MODIFY 或ES RENAME	改变EtherShare盘卷的名字、口令和/或访问权限	盘 卷 名 〔新名〕 〔（新口令）〕 〔新访问权限〕	欲修改的EtherShare盘卷名 给盘卷定的新名字 给该盘卷定的新口令，口令必须在括号中。空括号（　　）表示除去现有口令 可输入以下参数来改变当前访问权限 ／PUB表示公用访问权限 ／PRIV表示私有访问权限 ／SHAR表示共享访问权限
ES SDIR	列出网络上的所有服务器名	〔服务器名〕 ／W ／P	安装服务器时给它定的名字，可以使用DOS的百搭字符＊和？来列出特定的服务器，如果不给出特定的服务器名，将列出所有服务器名 给出本开关时，每行列出多个服务器名，否则每行只列出一个 当显示满一屏时暂停
ES UCREATE	在EtherShare中加入新用户	用 户 名	用以标识EtherShare用户的名字
ES UDIR	列出网络上所有EtherShare用户的名字	〔〔服务器·〕用户名〕 ／P ／W	欲获得有关信息的用户名，使用服务器名参数可列出特定服务器或一组服务器上的用户。如果不指定服务器名将列出网络上的所有用户 当显示满一屏时暂停 给出本开关时每行列出多个服务器名，否则每行只列出一个
ES UERASE 或ES UDEL	从EtherShare中删除一个用户名	用户名	被删除的用户名字
ES UMODIFY	给本用户加上或改变口令	（新口令）	改变给本用户名定的口令，空括号（　）表示除去口令
ES UNLINK	解除驱动器名与EtherShare盘卷之间的连接	驱动器名 〔用户名·〕盘卷名	与EtherShare盘卷连接的驱动器名 当前连接的EtherShare盘卷名

通过60H软中断自行安排的。其他用户要使用被锁住的文件时必须等到该文件的锁被解除之后才行。使用共享文件卷的所有用户程序都必须按照如下步骤进行文件操作：

——使用一个信号量给文件加锁

——读出文件/记录

——写入新的或修改过的数据

——清除文件区

——为该文件解锁

给文件加锁或解锁时都要使用一个信号量，它是锁的标志。信号量是一个可以长达31个字节的ASCII字符串，它可以使用百搭字符“*”，但*只能作为最末一个字符出现（且字符串长度应大于9）。

表6-13　文件加锁/解锁操作(INT60H)

操作名称	功能	入口参数	出口参数
LOCK/WAIT	使用信号量给文件加锁，成功则立即返回，不成功则等待一段时间，直到成功或越时为止	AH=11H AL=驱动器名（不用时为0） DS：BX=锁名（信号量）指针 ES：SI=指向以太网地址的指针（不用时SI=0） 如果AL和SI都为0，则使用登录时所用服务器的地址 DX=等待锁定的时间（以秒为单位）	在AL中返回的状态码： 0=操作成功 1=越时 2=服务器不响应 3=无效信号量名 4=信号量表已满 5=无效驱动器名 6=无效以太网地址 7=未登录 8=向网络写失败 9=文件已由该PC加锁
LOCK/RETURN	使用信号量给文件加锁，不管加锁是否成功，立即返回一个状态码	AH=12H AL=驱动器名（不用时为0） DS：BX=锁名（信号量）指针 ES：SI=指向以太网地址的指针（不用时SI=0） 如果AL和SI都为0，则使用登录时所用服务器的地址	在AL中返回的状态码： 0=操作成功 1=文件当前已加锁 2—9与LOCK/WAIT操作相同
UNLOCK	给一个当前加锁的文件解锁	AH=13H AL=驱动器名（不用时为0） DS：BX=指向锁名（信号量）的指针 ES：SI=指向以太网地址的指针（不用时SI=0） 如果AL和SI都为0时，使用登录时所用服务器的地址	在AL中返回的状态码： 0=操作成功 1=文件未加锁 2—8与LOCK／WAIT操作相同

文件加锁/解锁操作都是在相应服务器的管理程序中实现的。指定服务器的方法可以有三种：或者用已安装了共享文件卷的驱动器名来指出，或者用服务器的以太网地址（连接卡的48位地址）来指出，两者均不指明时则为用户登录入网时的那台服务器。

表6-13是以太网用于文件共享的三种加锁/解锁操作功能、入口参数及出口参数表。

4．共享打印机的管理与使用

（1）概述

使用EtherPrint/PC软件可以让网络上的所有用户共享连接在PC/XT网络服务器上的打印机。它提供以下功能：

* 将用户与服务器上的打印机建立联系，一台服务器可连接两台打印机供选用。

* 用户在自己的PC机上使用DOS命令、语言程序或屏幕拷贝（PrtSc键）等手段打印输出时，如同所在节点机本身带有打印机一样方便。

* 网上多个用户可以同时送出需要打印的信息，被打印的文件存贮在服务器内，以先到先服务的次序打印输出。

* 打印输出速度仅与数据在网络上的传输速度有关，而与打印机的速度无直接关系。

综上所述，EtherPrint软件使许多用户可以共享一台高性能的打印机，从而大大提高了整个网络系统的性能/价格比。

在使用EtherPrint/PC软件之前，必须先将它安装在PC网络服务器上。被安装的PC/XT服务器要求系统至少配置有256KB存贮器，并带有DOS 2.×及已安装好的EtherShare/PC服务器软件。安装的步骤简述如下：

① 启动PC/XT服务器，把它切换成专用模式。

② 选择专用模式下的菜单项目3，起动执行管理功能，在提供管理员口令后，进入管理菜单。

③ 再选择菜单项目1，进入安装操作菜单。

④ 选择安装菜单中的项目3，服务器进入安装网络应用程序菜单。

⑤ 把标有“EtherPrint/PC Server Software”的软盘片插入驱动盘A，输入回车键，菜单上显示出被安装的应用程序名之后，安装过程完成。

⑥ 由于打印机是多个用户共享的，每个用户使用打印机的方式可能互不相同。如有的用窄体形式打印，有的以宽体形式打印等，故每次使用后都应使打印机复位。由于不同型号的打印机使用不同的复位序列码，所以必须告诉“EtherPrint/PC服务器程序”所使用的打印机型号。这是通过“重新配置EtherPrint”功能来实现的。PC/XT服务器重新启动后，在管理操作菜单上选择项目3，即可进入“重新配置EtherPrint”操作，然后输入共享打印机的名字和类型即可。

到此为止，共享打印机的安装便全部完成，网络上的所有用户可以利用它来打印输出各种信息。在网络工作过程中，通过服务器程序可以了解到该打印机的使用情况，并进行适当的控制和干预。具体操作过程如下：

① 把服务器切换成专用模式，启动后选择菜单项目1，输入服务器口令之后，屏幕上显示出如下的“控制和状态菜单”：

ETHERSHARE CONTROL AND STATUS n.n

Registered EtherShare Name：EULER

1—SHUT DOWN ETHERSHARE

2—ETHERSHARE STATUS

3—ETHERPRINT STATUS

4—DONE

② 选择项目3，显示EtherPrint状态，此时屏幕出现：

ETHERPRINT STATUS n.n (Page 1 of 1)

PRINTER * 1：PROWRITER		PRINTER * 2：EPSON	
Owner	Length	Owner	Length
GREGS	5622		
STEVEP	1765	(NO FILES IN THIS PRINT QUEUE)	
MARIANC	12560		

这里列出了每一台打印机当前正在排队等待打印的用户和文件长度，排在第1个的是正在打印的文件主人的名字。

正在排队等待输出的文件统称为“假脱机”文件，需要的话可以把它们从队列中删去。此时只要按一下“D”键，然后使用光标控制键↑和↓把光标移到欲删除的文件处，再按一下SPACE键即可删去该文件。如果被删除的是第1个文件，则打印输出立即停止。

（2）共享打印机的使用

共享打印机安装完毕之后，网络用户就可以在各自的节点机上使用共享打印机。使用共享打印机是通过网络操作系统中的实用程序EP.COM来进行的。

① 建立打印机连接

将EtherSeries/DOS盘插在节点机的A驱动器上，启动计算机，然后使用“EP LINK”命令按照下述步骤建立打印机与PC机的连接：

A>EP LINK ？　　输入连接命令

Your printer id？　　输入欲连接的共享打印机名，如PRN：(或LPT1：)、LPT2：或LPT3：，若不给名，则默认为PRN：

To whom？SHARE1　　输入与该打印机连接的服务器名

Your name？NDUSER　　输入用户名，该用户名将印在每一页打印纸上，用以与其他用户的打印输出相区别

SHARE1 linked to PRN：　　表明打印机连接已完成

如果用户在登录入网后再使用打印机连接命令，则上述操作将有所简化，

② 打印文件

用户登录入网并建立了与打印机的连接之后，便可以使用共享打印机输出屏幕、文本或报告等各种信息了。打印输出的方法与单机操作时相同。若干打印输出的命令举例如下：

A>COPY REPORT PRN：　　将文件REPORT送到服务器打印

A>COPY E：MEMO PRN：　　将安装在驱动器E：上的盘卷中的文件MEMO送至服务器打印

“SHIFT”＋“PrtSc”（同时按下）　　在服务器上打印出当前屏幕内容（一屏）

③ 选择打印机

由于网络上有多台服务器，每台服务器可以连接两台打印机，因此用户在建立了打印机连接之后，需要时还可以更换为另一台打印机。为此，先要用EP DIR命令了解一下打印机的编号和参数：

A>EP DIR SHARE1　　列出连在服务器SHARE1上的打印机的编号和参数

Printer（s）Supported by the server：

1—DRAFT QUALITY
2—LETTER QUALITY
} 连在服务器上的两台打印机

然后再用EP LINK命令选择所需要的打印机，如：

A>EP LINK/2　　选择2号标志打印机

A>EP LINK SHARE2　　选择连接在服务器SHARE2上的默认打印机PRN：，即1号打印机

注意，第2条命令中的服务器可以是网中任一服务器，不一定是用户登录入网的那个服务器。

④ 拆除打印机连接

如果共享打印机不再使用，则可以打入下列命令拆除与该打印机的连接：

A>EP UNLINK

（3）共享打印机的操作命令

表6-14是以太网共享打印机的用户操作命令一览表，供读者参考。

5．电子邮件的管理和使用

（1）电子邮政程序的安装和管理

以太网电子邮政系统（EtherMail/PC）是以太网的一个应用程序。使用EtherMail/PC电子邮政系统就可以使网络上的用户相互交换电子信息。

EtherMail/PC分为两部分。一是EtherMail/PC用户软件，它在用户工作站上运行，用于写作、发送、接收和阅读电子邮件。另一部分是EtherMail/PC服务器程序，它在PC/XT网络服务器上运行，用以进行电子邮件的管理和分发，起着邮局的作用（图6-42）。

电子邮政的服务器程序的功能是：

* 接收用户发送来的邮件，并对送信用户给出反应。
* 把接收到的邮件分拣好，保留在邮局，直到用户请求时才转送到用户邮箱内。

在使用电子邮政系统之前，首先必须把EtherMail/PC服务器程序安装到网络服务器

表6-14　共享打印机操作命令

命令名	功能	参数	参数含义
EP DIR	列出连接在指定服务器上的打印机	〔服务器名〕	欲列出连接在其上打印机的服务器名（默认值为登录入网时的服务器名）
EP HELP	提供有关EtherPrint的命令：EP LINK，EP UNLINK，EP DIR和EP HELP的帮助信息	〔命令名〕	欲得到其有关信息的命令名
EP LINK	建立节点机和连接在网络服务器上的打印机的连接	〔打印机名〕	用来连接服务器打印机的打印机设备名。它们可以是PRN：（也称为LPT1：），LPT2：和LPT3：（默认值为PRN：）
		〔服务器名〕	连接打印机的服务器名，若已登录到EtherShare，则登录入网时的服务器可用作服务器名的默认值
		/1 /2	标识欲使用的打印机，它们可以是连接在指定服务器上两个打印机中任一个（默认值为1）
		/HOLD	延迟打印，直到使用EP UNLINK结束EtherPrint link或为这个打印机给出新的EP LINK命令为止
		/PLOT	除在打印请求之间不打印标志之外与/HOLD相同
EP UNLINK	解除节点机和服务器打印机的连接	〔打印机名〕	用EP LINK命令与之连接的打印机设备名，可以是PRN：，LPT1：，LPT2：，或LPT3：（默认值为PRN：）

上，安装过程与EtherPrint/PC服务器程序的过程类似，这里不再赘述。然后，再把EtherMail/PC用户软件复制到服务器中的系统盘卷SYS 2中，以便网络上的所有用户都能进行电子邮件操作。

把EtherMail/PC用户软件复制到系统盘卷SYS 2中的操作过程如下：

① 在网络的任何一个工作站上，用ES LOGIN命令以服务器名字作为用户名进行登录入网：

A>ES LOGIN〈服务器名〉

② 用ES LINK命令把系统盘卷安装在驱动器D：上：

A>ES LINK D: SYS2

③ 将标有“EtherMail/PC user Software for IBM PC Version 2.×”的软盘片插入驱动器A：，将其中所有文件拷贝到系统盘卷上：

A>COPY *.* D：

④ 重新把EtherSeries/DOS软盘片插入驱动器A：后，打入下列命令从网中退出：

A>ES LOGOUT

至此，电子邮政用户程序已经存贮在系统盘卷SYS2中了。此后，网上所有用户均可进行电子邮政的各种操作。

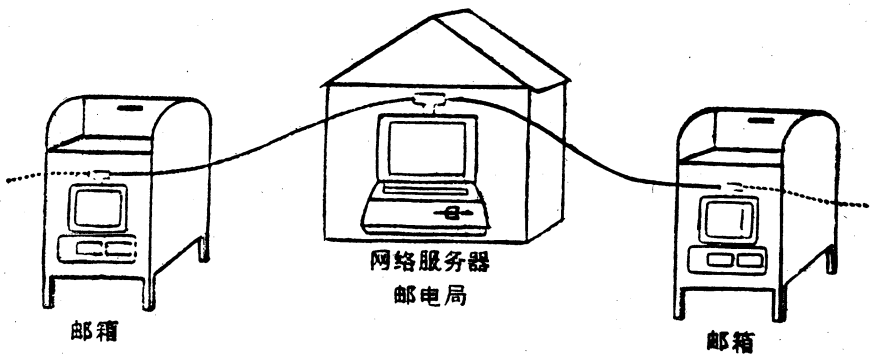


图6-42 以太网电子邮件的概念

电子邮件的管理，主要是显示服务器中未发出的邮件，并从中删去不需要的或过期的邮件。为此，首先要在PC/XT服务器上给出电子邮件的状态报告。这只要在服务器的专用模式下，进入管理操作菜单，然后选择项目4（显示电子邮件状态），显示屏上就会给出类似于如下形式的电子邮件状态报告：

ETHERMAIL STATUS n.n (page 1 of 1)

Total message: 16 Total bytes in use: 19546

Recipient	# Messages	Bytes
MARIANC	3	2093
TEST9	0	0
HOWARDC	10	13343
JEFFM	3	4110

其中，Recipient一栏中给出了所有待发邮件的用户名，#Messages为待发邮件的数量，Bytes是待发邮件的总长度。

如果需要删去一些不再需要传递的邮件时，可以按下“R”键，启动删除邮件操作。在输入管理员口令之后，就可以使用↑和↓键把光标定位到某个用户名，按一下〈SPACE〉键，就把等待发送给该用户的所有邮件全部删去。

电子邮政程序为网络服务器上的所有用户自动地建立“邮箱”，如果使用ES UDEL命令将用户从服务器中删除的话，则该用户的邮箱将不再被使用，于是就要用上述操作把不再

有任何用户接收的邮件进行删除。

（2）电子邮政程序的启动

电子邮政用户程序允许网络工作站上的用户交互式地进行下列操作：

- 接收新邮件
- 阅读报文
- 报文写作
- 发送报文
- 回复收到的报文
- 转发收到的报文
- 报文存档
- 报文打印
- 发报过程中存贮报文
- 删除报文

在具体介绍电子邮政程序的使用之前，先引入下列几个概念：

- 邮夹（Mail Folder） 用户每次启动电子邮政程序后，就会被告知邮局中是否有新的邮件等待他取走。如果有的话，则把邮件取来，并把其中的报文存放到一个邮夹中。邮夹是专门用来存放邮件的一个盘卷，也可以是一张软盘片。报文以DOS文件的形式保存在邮夹中，需要时可以从邮夹中取出报文进行阅读。

- 分发表(Distribution Lists） 邮件分发表是一张经常需要向他们发送报文的那些用户的一览表。为了方便起见，分发表可以当作一个收信人使用。分发表是一种DOS格式的文件，它可以用报文编辑程序建立和修改。

- 报文编辑程序 报文编辑程序用来输入、编辑报文的正文，在报文写作、回复报文或转发报文时就需要使用报文编辑程序。

- 报文附件（Attachments） 报文附件指的是附加在报文中与报文一起发出的任何DOS文件（正文文件、程序文件或数据文件）。一个报文最多可以有26个附件。

电子邮政用户程序是由三个相互链接在一起的文件组成的。这三个文件是MAIL.COM，ZZMAIL.COM和MED.COM，它们已复制在系统盘卷之中。为了启动电子邮政程序运行，必须打入下列命令：

MAIL〔邮夹驱动器〕〔邮夹路径名〕〔分发表驱动器〕〔分发表路径名〕

为了操作方便，系统中还提供了四个批命令供用户使用：

① EXECMAIL〔用户名〕

它把用户登录入网，把系统盘卷SYS 2安装在驱动器D：上，把盘卷INBOX（假定已存在）安装在驱动器E：上，与共享打印机建立连接，然后启动电子邮政程序并指定邮夹在驱动器E：上（即INBOX盘卷）。

② NEWINBOX

为已登录的用户建立一个新盘卷INBOX。

③ RENUMBER

把盘卷INBOX（邮夹）中的报文重新编号。

④ RUNMAIL

当用户登录后，使用这个批处理命令将把盘卷INBOX作为邮夹并启动电子邮政程序运行。这个批文件的内容为：

```
ECHO OFF
ES LINK E:INBOX/NP
IF ERRORLEVEL1 GOTO FAIL
MAIL E:
IF ERRORLEVEL1 GOTO FAIL
ES UNLINK E: /NP
:FAIL
```

使用上述几条批处理命令，就可以方便地启动电子邮政程序工作。

（3）电子邮件的接收、阅读．打印和存贮

① 电子邮件的主画面

网络上的用户接连使用三条批命令（LOGIN、NEWINBOX、RUNMAIL）后即可启动电子邮政程序运行，电子邮政程序启动后将显示如下主画面：

```
EtherMail n.n  (c) Copyright  3Com Corp 1982, 1983
1      12-8-82      marys    Market Survey
2     *12-15-82     steve    RE: Staff Meeting
8       1-20-83     joep     FYI: Febrary Budget
8 A    Attachment             BUDGET
9      In progress  fredj     Monthly Report
```

主画面中列出了用户邮夹中的全部邮件及有关信息，如果在一屏内容纳不下，还可以用PgUp和PgDn键来继续显示。主画面中各栏目的含义是：

* 邮件编号。这是由EtherMail设置并用来标识邮件的。

* 邮件发送日期，前面加有*号的表示该邮件尚未被读过。

* 发送邮件的用户名。

* 邮件科目。例如，通知、货单等。在前面冠以RE：的，表示该邮件是一个答复邮件；前面加有FYI：的，表示这是一个转发邮件。

* 待处理（In Progress）。表示这是一个未发出的邮件，暂存在邮夹中。

* 报文附件（Attachment)。表示这是一个邮件上的附件，编号8 A是编号为8的邮件的附件。任何DOS文件均可附加在邮件上，与邮件一起发送。

* 状态行。在主画面的下面有一个状态行，它将显示一些提示信息和出错信息，便于用户使用EtherMail。

② 接收新邮件

当主画面的状态行上指出网络服务器上有新邮件时，必须先把它转移到自己的邮夹中然后才能阅读。

按一下“F8”键，EtherMail将把新邮件转移到邮夹。任何时候按F8键，EtherMail都将检查是否有新邮件。如有的话则把它转移到邮夹中，没有的话，在状态行上给出信息。

③ 选择和阅读报文

通过在主画面上用↑和↓键把光标移到要处理的邮件编号下面，就可以选择某一个邮件进行阅读、打印、转发或其它处理。选好之后，按一下“F2”键（阅读），EtherMail就把该邮件的内容显示在屏幕上。在屏幕的上部还显示出该邮件的发送者（From）、接收者（To）、转发去向（cc）和科目（Subj）。

报文阅读完毕之后，可以打印、删除、转发等，也可以再按一下“F2”键继续阅读下一个报文。

④ 打印和存贮报文

如果需要打印报文，可以在主画面上选择要打印的邮件，或直接把屏幕上正在显示的邮件通过共享打印机打印出来，也可以用用户节点机上的打印机PRN：打印出来。无论是哪一种情况，只要按一下“F4”键，即可开始打印。

用户邮夹中的任何报文或附件均可作为DOS文件存贮起来。选择报文后，按一下“F5”键，这时状态行上将显示：

File name?

回答一个文件名之后，所选择的报文就作为DOS文件存贮在盘上，以后就可以用DOS命令或应用程序来访问这个文件。

（4）邮件的发送、回复和转发

用户写作一个新的报文之后，即可进行发送。此外，在对收到的邮件作回复以及转发一个邮件时也要执行发送操作，它们的发送过程是相同的，仅邮件标头有所不同。

① 写作新报文

在主画面上按功能键“F9”，即可开始写作新的报文。在写作新报文时，首先必须填写报头，然后才能用编辑程序编写正文。

报头包含了该邮件的投递信息，它分为若干字段，需逐段填写。不一定要填写的可选字段，用回车键可以跳过。填写内容超过一行时，须在行末打入逗号或分号。需要填写的字段有：

- * From：　发信用户名。由EtherMail自动填上。
- * To：　接收用户名。当接收者为多人时，用逗号或分号隔开。
- * cc：　转发去向用户名。这是可选字段，其填写方法与To相同。
- * Subj：　邮件科目。这也是可选字段。
- * Attach：　邮件的附件。这也是可选字段，填入作为附件的DOS文件名。最多可附带26个文件。

报头填写完毕之后，即进入编辑画面，这时就可以写作正文了。写作是使用编辑程序来完成的，EtherMail的编辑程序提供了插入、删除、移动、拷贝等一系列功能，使得正文编辑较为方便。

② 发送编辑好的邮件

正文编辑完毕之后，按“F10”键即可退出编辑状态。这时按功能键“F6”就可以把该邮件发送出去。当报头各字段填写无误时，邮件将送到服务器中，然后屏幕返回主画面，准备执行下一步操作。

如果报头填写有误时，状态行上将显示出错信息，这时必须按“F7”键，返回编辑程序纠正错误。当遇到无效的接收用户名或服务器不响应时，系统将询问是否继续发送。若回答Y，则将把邮件送往那些有效的接收者而忽略无效的收信者。若回答N，则可以按“F7”返回编辑程序进行修改，或按“F10”把该邮件暂存起来，以后主画面上该邮件处将标有“In Progress”。按“F10”后也可放弃该邮件。

③ 回复

当需要回复一份邮件时，可以在主画面上选择该邮件后或在阅读该邮件时，按一下“F6”（Repl），即进入回复操作。

回复报文的报头部分中，字段From、To、cc及Subj均由程序自动填写好。在Subj字段上插入了“RE：”，表示是回复报文，所以回复邮件的报头只需填写可选字段Attach。填写好后，将进入编辑画面。回复报文的编辑与写作新报文相同，编辑完成后，按“F6”即可将回复报文发给原送信者。

④ 转发

若要将已收到的邮件转发给有关的用户，可以使用转发功能。选择需转发的邮件后，按功能键“F7”（Forw），即可开始转发操作。

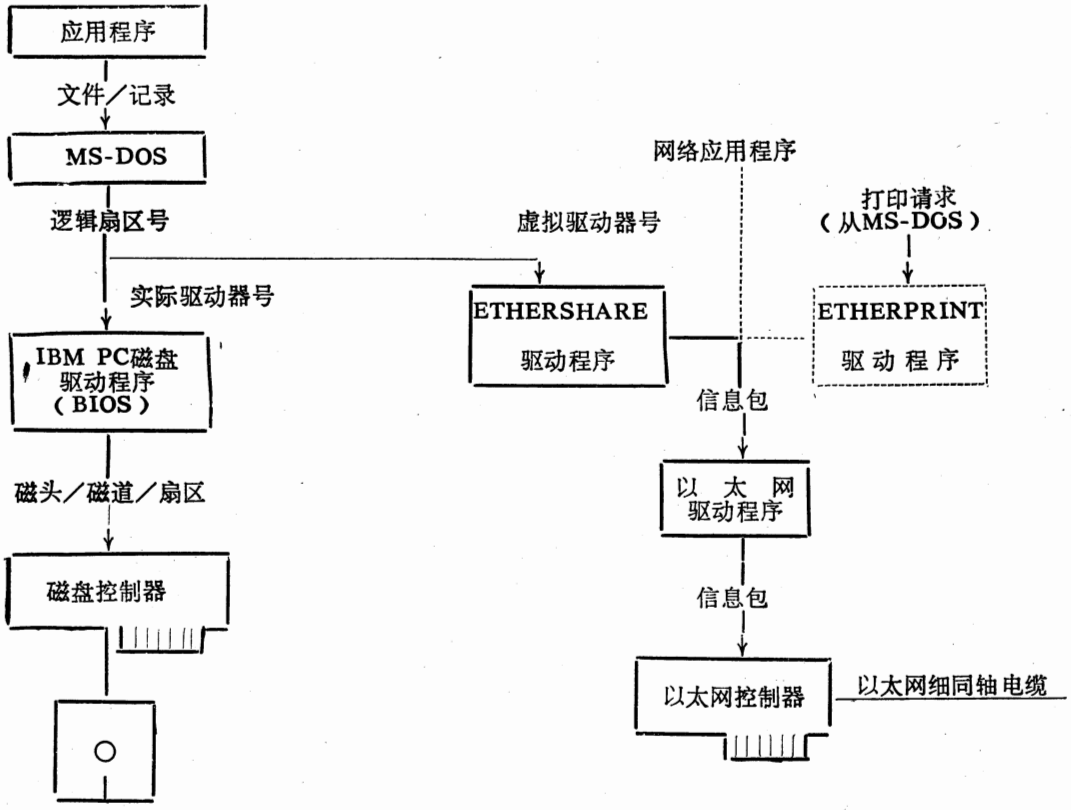


图6-43 以太网磁盘操作原理示意图

转发邮件的报头部分中From及Subj字段已由程序自动填好，在Subj的科目之前，添加了“FYI：”标记，表示这是转发邮件。字段To：填写转发去向的接收者用户名。字段cc：填写再转发的用户名（即传阅方式）。这里也可以填上自己的用户名，在转发的同时，自己也收到了该邮件。字段Attach填完之后，将进入编辑画面。原来的报文全部包含在转发报文的正文之中。另外，在头上还留有一些空行可以插入批注。编辑完成后，按“F10”退出编辑，然后再按“F6”即可进行转发操作。

从本节的介绍可以看出，以太网的主要特点是使用虚拟磁盘（盘卷）的概念来达到网络用户共享资源和相互通信的目的。全部网络软件虽然长达85000行语句，但它们与MS-DOS的界面非常清楚、简洁。这样不仅使现有的MS-DOS应用软件全部能在网上运行，而且也不影响MS-DOS进一步升级后在网上的正确运行。图6-43是以太网磁盘操作处理过程的剖示图，读者从中可以领会到以太网软、硬件的一些特点。

第六节　IBM公司的微型计算机网络

早在七十年代，IBM公司就提出了网络体系结构（SNA）的设想，并逐步在该公司的各类计算机产品上得到实现。这给计算机网络技术带来了较为深远的影响。IBM公司的各类大、中、小型计算机相互之间的通信以及联网均以SNA为标准，随着IBM PC微型计算机的推广使用以及微机局部地区网络技术的发展，IBM公司也为IBM PC各档机种开发了多种微机局部网络产品。由于技术先进，又能充分使用PC机上原有软件资源，并且考虑了与SNA的连接等因素，因此IBM公司的PC局部网有较高的性能/价格比和广泛的适用范围，它对微机局部网技术的发展也将产生重大的影响。

1. 概述

在发表IBM PC/AT的同时，IBM公司也正式发表了PC机的网络产品，根据功能、规模及应用领域，可以把它们分为四类。

① IBM PC群集系统　这是面向小型用户的IBM PC低档局部网络，它适合于在一个小型企业单位甚至一个教室的范围内使用。

群集系统采用总线式多站结构，使用IBM PC/XT作服务器，IBM PC作工作站，最大节点数为64个，但不能支持IBM PC/AT。网络的数据传输速率为375Kbit/秒，是一种低速的局部网。操作系统使用MS-DOS 2.1版。

② IBM PC网络（IBM PC-NET）　这是IBM局部网的主要产品，面向中、小规模用户。

网络为总线式结构，IBM PC、PC/XT、PC/AT等微机都可连接入网，AT、XT可作服务器使用，最大节点数为72个，网络数据传输速率为2Mb/秒，操作系统使用MS-DOS 3.1版。

③ Token 环敷线系统（Cabling system）　它是连接IBM公司各种计算机和设备的网络，用来实现相互间信息交换、系统资源及大规模数据库的共享，是一种大型的网络系统。其特点是使用平衡式双绞线，而不是使用同轴电缆。并且，它象敷设电话线一样，在整个建筑物的每个房间内都敷设好线路并引出插座。

④ 过程控制局部网　这是为了满足工厂生产要求而开发的局部网，用以连接工厂中的数据采集设备、机器人、机床、数控装置及各种工业用计算机的网络。

上述四类网络产品最终可能相互连接，以便组成适用于各种应用领域的网络应用系统。

表6-15是这四种局部网络的产品性能。

表6-15　IBM的PC网络产品性能表

项　目	群集系统	IBM PC局部网	敷线系统	过程控制局部网
适用机种	IBM PC、PC/XT	IBM PC、PC/XT、PC/AT	S370，S/38，S/36，其它中型3270控制器和3270PC	序列/1，CAD/CAM工厂数据采集设备，机器人，数传设备
网络连接卡	群集系统适配卡	网络适配卡		
访问控制	CSMA/CA	CSMA/CD	Token环	Token总线
网络拓扑	总线式	总线式（可分枝）	环	总线式
传输介质	基带同轴电缆	宽带同轴电缆	双绞线，光导纤维电缆	宽带同轴电缆
传输速率	375Kb/S	2Mb/S	1或4Mb/S（双绞线）4，16或100Mb/S（光导纤维电缆）	1，5或10Mb/S
应用面向	小型用户、企业管理、教学系统	中型用户	大型用户	过程控制

2. IBM PC 群集系统

（1）硬件配置

IBM PC群集系统的硬件由IBM PC、PC/XT机、群集系统适配卡及连接电缆所组成。

每台IBM PC机必须配置一块群集系统适配卡，它提供了将IBM PC与网络传输线相互连接的能力，主要特性为：

* 总线式网络结构，传输介质采用75欧姆同轴电缆。
* 基带传输方式，数据传输速率为375K位/秒。
* 访问控制方式采用CSMA/CA。
* 每个群集可有64个工作站。
* 一个群集允许有5个工作站同时工作。
* 主干线最大长度为1000米。
* 工作站与总线的连接电缆最长为5米。

采用IBM公司提供的群集器电缆套件，可以方便地构成群集系统。套件包括10米总线电缆，3米分路电缆，T型连接器（与总线电缆连接用）、BNC连接器（与PC机上适配卡连接用）以及终端插头。群集系统的网络结构示意如图6-44所示。

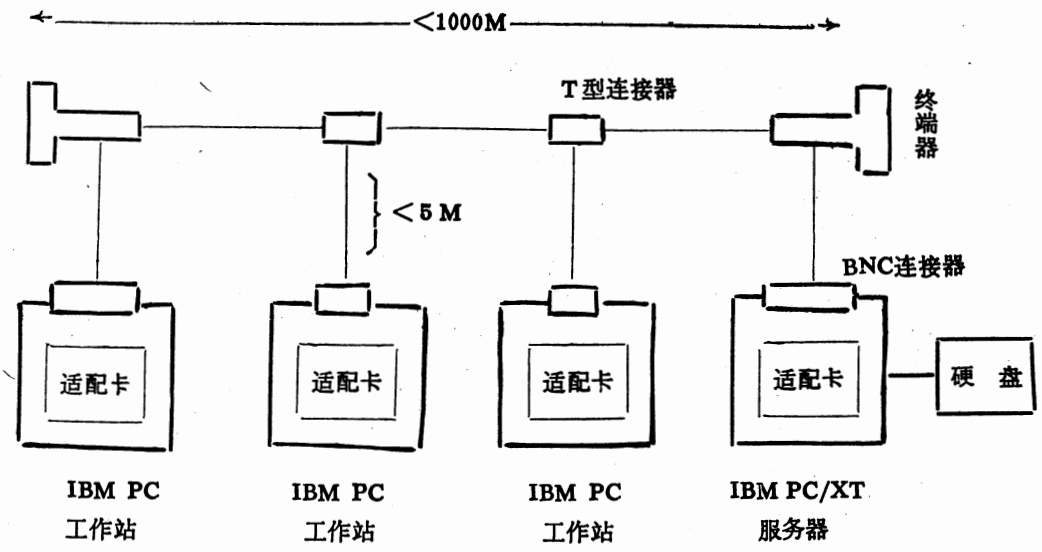


图6-44 IBM PC群集系统结构示意图

（2）群集程序

群集系统使用IBM群集程序作为网络管理软件，从而能在一个群集内实现资源共享和数据通信。

群集程序指定群集中某一台IBM PC/XT（或带有硬盘的IBM PC机）作为硬盘服务器，网络上其它工作站可以共享服务器上的硬盘资源。服务器上的硬盘被划分为若干盘卷，它们可以是供所有工作站共享的公用盘卷，供用户系统在初始程序加载时(IPL)使用的系统盘卷，以及供群集内各工作站自己使用的专用盘卷，剩余的磁盘空间供服务器（PC/XT）自身使用。各类盘卷可赋以“不访问”、“只读”、“只写”或“读写”等不同的访问权限。对工作站来说，服务器硬盘上的公用盘卷和各自的专用盘卷如同工作站自己的附加软盘一样，可以方便地使用。群集程序支持MS-DOS 2.1版及各种应用软件，为IBM PC开发的各种软件包均可在网上运行。

群集程序还能支持一个群集系统中的任意两个计算机之间的信息通信（点对点通信），操作采用菜单方式。

由于群集系统易于构作，价格低，因此这是一种把IBM PC机扩充成为网络系统的简便途径。

3. IBM PC NETWORK（IBM PC-NET）

（1）IBM PC-NET硬件配置

IBM PC-NET局部网由PC、XT或AT等节点机及网络适配卡、全网络传输单元和各种网络连接装置组成。IBM PC-NET网络的主要技术数据如下：

* 可分支的总线式结构，传输线采用闭路电视使用的75欧姆标准同轴电缆。

* 宽带传输方式，发送频率为50.75MHz，正向接收频率为219.00MHz，通频带带宽6MHz。数据传输速率为2Mbps（位/秒）。

* 采用CSMA/CD访问控制方式。

* 使用标准电缆套件时可支持72台PC，采用双向闭路电视系统时，可以支持大约5公里范围内的1000个工作站。

* 使用48位的工作站地址。

为了确保在较大范围内实现宽带传输，IBM PC-NET使用了各种网络传输部件。例如：

① 网络传送器。这是无源部件，适用于单通道网络传输，允许连接距离在60米之内的8个工作站，可以与网络联结扩展器连接。

② 网络联结扩展器。它可以连接8个短距、中距或远距离配套器，是一种有源部件。

③ 短距离配套器，与扩展器直接连接，并可带有8个工作站，距离半径为60米。

④ 中距离配套器，与扩展器的连接距离为120米，本身可连接距离在60米之内的8个工作站。

⑤ 远距离配套器，与扩展器连接距离为240米，本身也可以连接距离在60米之内的8个工作站。

IBM PC-NET的网络连接示意图见图6-45。

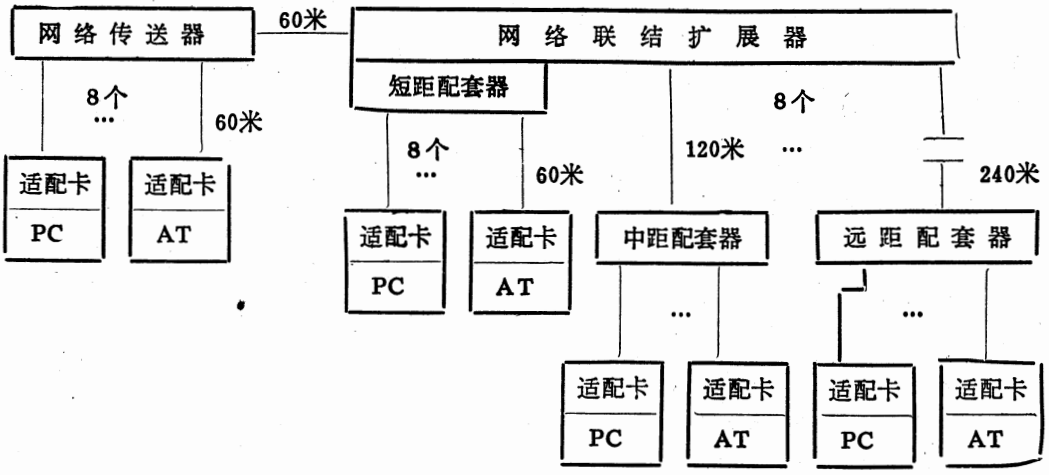


图6-45 IBM PC-NET结构示意图

（2）网络适配卡

IBM PC-NET所使用的网络适配卡是用来把IBM PC、PC/XT、PC/AT连网的专用卡，它插在各节点机的扩充槽内。

IBM PC-NET的通信协议与IEEE802.3的国际标准基本相符。网络适配卡提供了通信协议的硬件实现手段，适配卡上的网络处理机Intel 80188和网络BIOS固件(NET BIOS)实现了ISO OSI七层模型中的低五层功能和一部分最高层（应用层）的功能，这是一种功能相当强的网络连接卡。

图6-46是IBM PC-NET网络适配卡的结构示意。适配卡的各组成部分的功能已在图中标出，这里不再重复。

网络适配卡使用PC机I/O通道的3＃DMA信号和2＃（或3＃）INT信号与主机接口，I/O端口地址为360H-367H或368H-36FH，由开关进行选择。

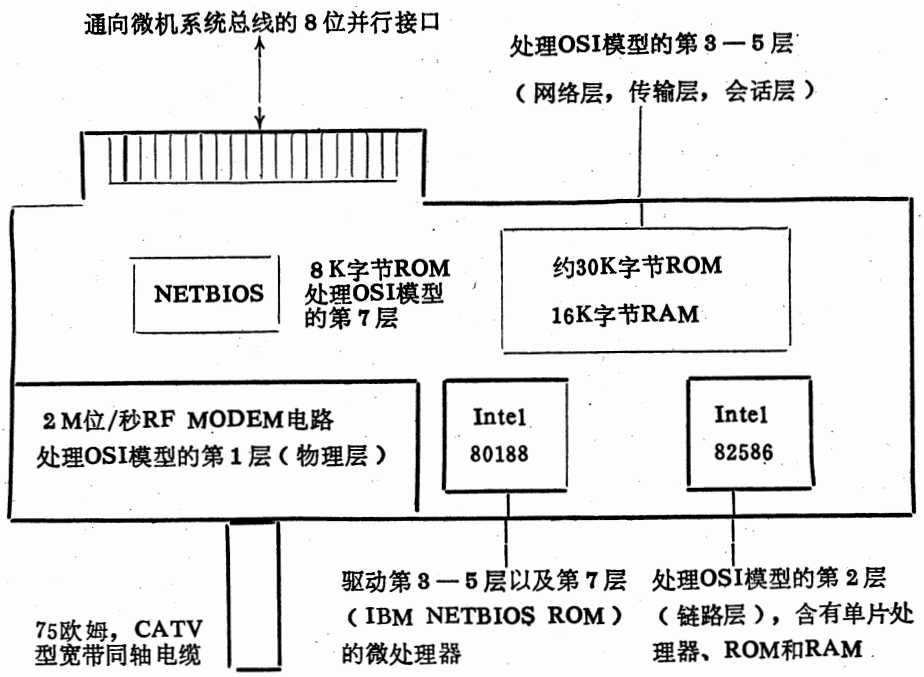


图6-46 IBM PC-NET网络适配卡的结构图

（3） IBM PC-NET网络软件

PC-NET网络软件由适配卡上的NETBIOS固件和PC-NET网络程序两部分组成。

① NETBIOS（网络BIOS）

NETBIOS是安装在网络适配卡中ROM芯片里的固件，它为所有的网络程序提供下列基础功能：

* 用户级点对点通信的能力。
* 数据块的发送、接收和广播功能。
* 定义一个站点中各个用户的用户名。
* 查询网络适配卡的状态并对适配卡的操作进行控制。

NETBIOS分担了主处理机的许多功能，使网络应用程序大为简化。NETBIOS允许多个用户名与适配卡相对应，从而使用户可以在操作和使用状态下动态地改组网络的结构。

② IBM PC-NET的网络程序

网络程序为网络上各节点提供了资源共享和用户间相互通信等功能。网络程序规定了网

络上的PC机有四种工作模式，任何节点机都可选择其中的一种。所有的模式都允许存取和使用网络上的共享硬盘、公共数据文件和网络打印机。节点机的四种工作模式按功能递增的顺序是：

输入输出转发器模式（REDIRECTOR） 它允许许多现存程序不加修改地可以存取另一个节点的数据，还可以共享网络上的打印机。这种模式需要有128K内存。

接收器模式（RECEIVER） 除上述功能外，还能接收和记录从其它节点送来的信息，需要192KB内存。

发送器模式(MESSENGER) 除上述两项功能外，还能执行"转向"等更为复杂的信息传输功能，并能与同一节点机中的其它应用程序异步地使用全屏幕接口设备，需要256KB内存。

服务器模式（SERVER） 除以上全部功能外，还能使接点机响应网络中由REDIRECTOR发送来的各种服务请求，使节点机兼有服务器的功能，这种模式需要320KB内存。

IBM PC-NET网络程序的功能大体上分为两个方面。一方面它以命令和菜单形式向用户提供下列操作功能：

- 为节点机配置网络程序。
- 在网络中建立或删除PC机名。
- 建立文件共享环境。
- 处理打印顺序。
- 与其它节点机通信。
- 完成数据文件的传送操作。

另一方面它又通过文件服务、打印服务和输入输出转发等功能，使用户不必改动现有的应用程序，就可以得到使用远距离的网络文件操作和打印操作来代替本地操作的种种方便。

IBM PC-NET的网络程序发展并充实了其它各种局部网的功能，在DOS3.1的支持下使IBM PC局部网技术得到了新的提高。

4. IBM PC网的发展

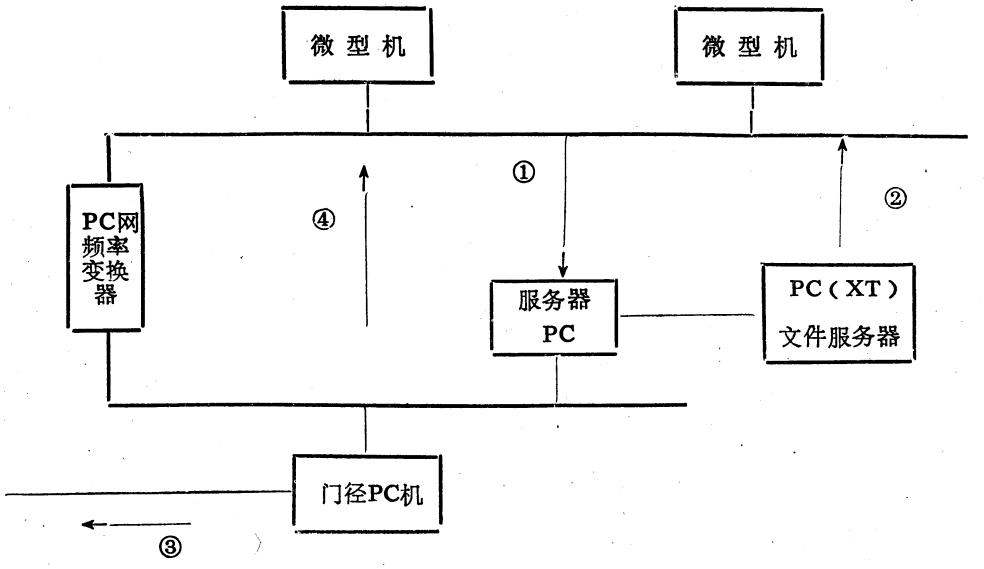


图6-47 IBM PC-NET上的3270仿真技术

为了使IBM PC-NET能与IBM公司的大、中型计算机相连接，PC-NET网络上可以设置一个“门径”计算机（Gateway-PC），网络上的其它节点就可以通过3270仿真器来实现对远程（或近程）的大、中型机的访问。其结构示意如图6-47所示，大致工作过程如下：

① IBM PC-NET局部网上的某节点需要与大型机通信时，先调出3270仿真程序。

② 连接在网上的文件服务器把3270仿真程序加载到该节点，从而使该节点看起来是一个3270工作站。

③ 门径计算机仿真3274控制器，实现与大型机的通信。

④ 门径计算机对“3270”终端机进行会话处理。

有了上述功能，局部网中的每一个节点都可以使用大、中型计算机上的大量软、硬件资源，从而大大增强了IBM PC个人计算机的处理能力。

此外，前面所提及的IBM公司的三种局部网也可以实现相互连接，其可能的结构形状如图6-48所示。可以相信，这种多功能的计算机网络系统必将得到人们极大的关注。

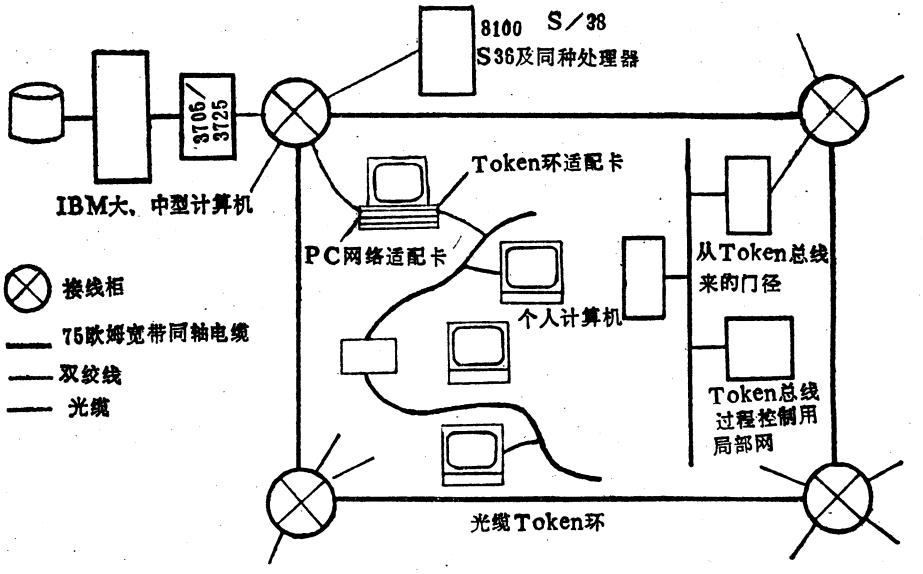


图6-48 IBM公司不同局部网的互连结构

附 录

附录Ⅰ　IBM PC/AT简介

1. 概述

IBM PC/AT是IBM公司采用先进技术设计成的一种高性能个人计算机，它具有较快的处理速度和较大的存贮容量。PC/AT既可以作为单用户系统使用，也可以组成一个多用户系统。目前，PC/AT上可以运行的操作系统主要有MS-DOS 3.0和XENIX两种。MS-DOS 3.0是在MS-DOS 2.0的基础上开发的一个单用户单任务操作系统(详见第一章)。XENIX则是Microsoft公司根据UNIX Ⅲ而开发的一个多用户多任务操作系统，它一般可以支持三个用户。

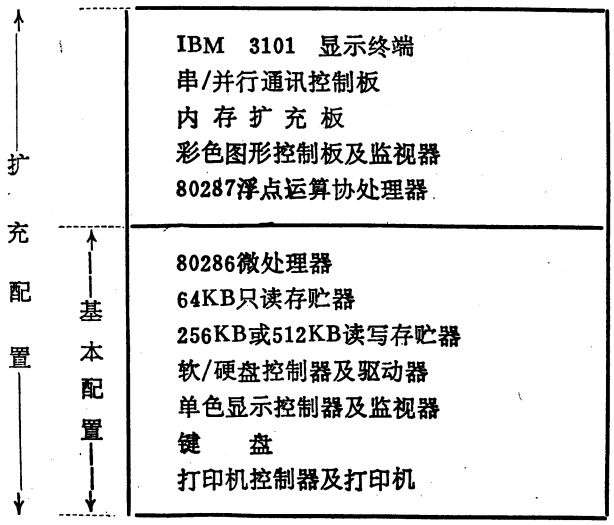
当运行MS-DOS 3.0时，PC/AT对PC或对PC/XT都具有较好的向上兼容性。许多PC软件可以不加修改地直接在PC/AT上正确地运行，从而有利于PC用户的升档。

此外，与PC或PC/XT相比，PC/AT还具有以下一些特点：

* 采用先进的Intel80286微处理器为中心组成中央处理器，并可选用Intel80287协处理器进行浮点运算，处理速度通常是PC/XT的2—3倍。
* 内存在运行MS-DOS时可达640KB，而在运行XENIX时可达3MB。
* 采用1.2MB的高密度软盘驱动器和20MB的硬盘驱动器作为系统外存，提高了外存的容量（最大可达41.2MB）。
* 提供八个与PC兼容的扩充槽，可直接插入多种PC选件板，其中六个槽还加有PC/AT专用扩充槽，可以插入PC/AT的16位选件板。
* 提供一个电池供电的实时钟，可以不间断地计时。

2. 系统配置

PC/AT在运行MS-DOS 3.0时，其硬件配置由主机箱、键盘、显示器和打印机等四个部分组成。其中主机箱中安装有系统板（包括中央处理器、内存和I/O通道）、软/硬盘控制板、显示/打印控制板和软/硬盘驱动器等部件。当运行XENIX时，还可配接两台IBM 3101显示终端以及五块512KB内存扩充板（PC/AT的八个扩充槽的另外三个，已插入必不可少的三块扩充板）。PC/AT的基本配置和扩充配置如下图所示。



3. 中央处理器

PC/AT的中央处理器以主频为6 MHz的80286微处理器为核心组成，必要时还可加接80287浮点运算协处理器。中央处理器的其它组成部分主要有时钟控制器、中断控制器和DMA控制器等。

（1）Intel80286微处理器

Intel80286是Intel公司在Intel8086/88基础上开发的一种16位微处理器芯片。它与8086/88保持向上兼容并提供较强的存贮管理和存贮保护功能。它的主要特点如下：

* 具有较高的时钟频率和较强的指令系统，处理速度与8086/88相比有较大的提高。

* 提供实地址和虚地址两种操作模式，其中实模式与8086/88完全兼容，而虚模式下可充分利用80286的存贮管理和存贮保护功能。

* 有24根地址线，寻址能力较强。实模式下可直接访问1 MB内存；虚模式下可访问的实存空间为16MB，虚存空间达1000MB/任务。

* 有16根数据线，可直接处理16位数据。

* 与8086/88的软件具有向上兼容性。实模式下具有目标代码级的兼容性，即8086/88的代码程序可直接在80286上运行；虚模式下具有源代码级的兼容性，即8086/88的软件经过重汇编或重编译后，就可以在80286上运行。

* 虚模式下80286具有四层（系统核、系统服务、应用服务和应用程序）存贮管理和存贮保护功能，便于多任务操作系统的实现。

* 配套电路较为齐全。除了可以使用8位外围电路外，还可以使用16位外围电路，例如82284时钟发生器、82288总线控制器和82289总线判优器等。

* 可以配接多种协处理器，例如80287浮点运算协处理器、802730字符/图形显示协处理器和802586以太网协处理器等。

* 和8086/88一样，80286对内存进行分段管理，每段的长度不大于64KB。

* 80286封装在68条引线的正方形管壳中，采用四面引线的方式，在不增加管壳面积的条件下增加了引线数目。

（2）中断控制器

系统板上有两片Intel8259A中断排优控制器，用来对16路中断信号进行排序。其中与80286直接连接的仅有一片8259A，另一片8259A通过第1片8259A的IRQ 2接入802 86。

PC/AT已使用的中断信号主要有系统时钟（IRQ 0）、键盘中断（IRQ 1）、软盘中断（IRQ 6）、硬盘中断（IRQ14）、串行口中断（IRQ 3和IRQ 4）、并行口中断（IRQ 7和IRQ 5）和协处理器中断（IRQ13）等。

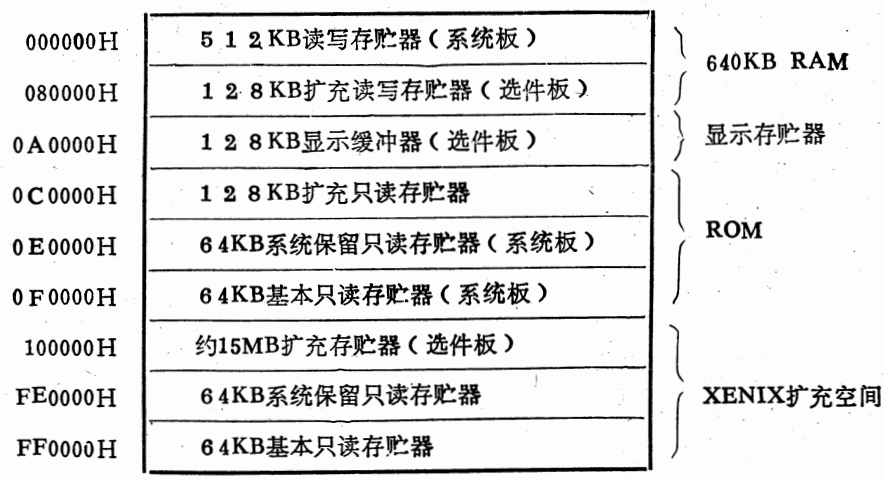
（3）DMA控制器

PC/AT的七路DMA控制器由两片Intel8257A-5 DMA控制器组成。第1片8257A提供的DMA通道0—3可以用于在8位I/O设备与内存之间高速交换数据，该片8257A通过第2片8257A的通道4与80286连接。第2片8257A直接与80286连接，它提供的DMA通道5—

7用来使16位I/O设备与内存高速交换数据。

4. 内存空间布局

PC/AT的内存空间的总容量为16MB。实模式（MS-DOS）下仅能访问前1MB空间，虚模式（XENIX）下能访问全部16MB空间。但目前限于扩充槽的个数以及内存扩充板的容量，XENIX最多能访问3MB读写存贮器。PC/AT的内存空间布局如下图所示：



运行MS-DOS时，可访问的读写存贮器最大为640KB，由系统板上的512KB RAM和128KB扩充板组成。运行XENIX时，可访问3MB读写存贮器，由系统板上的512KB RAM和五块512KB扩充板组成。

系统保留的64KB ROM和基本ROM分别对应0E0000H—0FFFFFH和FE0000H—FFFFFFH两块地址空间，即分别位于实模式和虚模式存贮空间的尾部。

PC/AT的64KB基本ROM中包含磁带BASIC解释程序、加电自检与引导程序和BIOS等。其中BIOS与PC或PC/XT的BIOS相比，功能有以下几项扩充：

* 支持多任务操作系统。在调用BIOS子程序时，BIOS会发出等待信号，子程序执行完毕后会给出执行结束信号。
* 为了实现向上和向下的兼容性，提供操纵杆和微秒级间隔时钟的驱动程序。
* 提供标准内存和扩充内存之间的数据块传送功能，以便利用512KB扩充板作为虚拟盘的介质。

应当注意，BIOS仅能在实模式下工作，支持工作在实模式下的操作系统，例如，MS-DOS和Digital Research公司的并发DOS（与MS-DOS兼容）。工作在虚模式下的操作系统，例如XENIX和IBM公司的Topview则必须自带BIOS。

5. 输入输出通道

PC/AT提供八个I/O通道（扩充槽），它们都可以插入PC选件板。其中六个长槽还可插入PC/AT的专用选件板。长槽由两部分组成，第1部分为与PC相同的62线插座，第2部

分为扩充的36线插座。长槽可以向选件板提供下列信号：

* 24根地址线；8或16根数据线。
* 100H—3FFH之间的I/O端口地址。
* 中断信号；DMA信号。
* I/O等待状态信号。
* 通道CPU抢权信号（MASTER）。

其中MASTER信号允许选件板上的CPU暂时取得系统的控制权，以访问系统的内存和外存。所以，PC/AT的扩充槽允许插入智能外设控制板和协处理器板。

6．键盘

PC/AT的键盘是在PC键盘的基础上设计而成的，它与PC键盘不兼容，不能互相换用。与PC键盘相比，PC/AT的键盘有以下一些改进：

* 增加了〈Shift〉，〈Ctrl〉，〈CR〉，〈Tab〉和〈BS〉等键的面积，调整了〈Esc〉，〈PrtSc〉和反斜杠等键的位置，使这些常用键的使用更加方便。
* 增加了三个按键状态指示灯，它们分别用来指示〈CapsLock〉，〈ScrollLock〉和〈NumLock〉这三个键的按下状态。
* 增加了一个新键〈SysReq〉，但目前MS-DOS不使用该键。

7．日时钟

PC/AT使用一个真正的日时钟，它为系统提供日期和时间数据。在机器断电期间，该时钟使用电池供电继续计时。

实时钟提供的日期和时间数据保存在一块64B的CMOS存贮器中，该CMOS在断电时也使用电池供电。该CMOS存贮器除了记录日期和时间外，还记录着机器的配置数据，例如存贮器容量、软/硬盘类型和是否安装协处理器等。这些数据可以使用系统设置程序进行修改。

8．磁盘存贮器

PC/AT使用高密度的5¼时软盘（容量为1.2MB）和20MB的硬盘作为系统外存，外存容量比PC/XT增加了许多。此外，PC/AT的软盘和硬盘合用一块选件板，结构较为紧凑。

（1）软盘驱动器

PC/AT既可以配接1.2MB的高密度软盘驱动器，也可以配接普通的360KB软盘驱动器。普通360KB软盘驱动器用来与PC和PC/XT交换文件。

高密度软盘驱动器需要使用特制的高密度软盘片，该盘片为双面倍轨四密度的5¼时软盘，每面有80道（轨），每道有15个扇区，每个扇区可以记录512个字节。该盘的总容量达1.2MB。

高密度软盘驱动器也可以用来读写普通的360KB软盘。但由于该驱动器写的信息道较窄，故在PC或PC/XT上有可能读不出。根据经验，在大多数情况下是可以读出的，所以不

一定非要安装360KB软盘驱动器。

（2）硬盘驱动器

PC/AT可以配置容量为20MB的温彻斯特硬盘，该盘的转速为3573转/分，数据转输速率为5兆位/秒。

（3）软/硬盘控制器

PC/AT的软/硬盘控制器可以用来控制两个软盘驱动器和一个硬盘驱动器，或者控制一个软盘驱动器和两个硬盘驱动器。软盘驱动器可以是高密度软盘驱动器，也可以是普通的360KB软盘驱动器。

9. PC/AT的兼容性

IBM公司在设计PC/AT时，充分考虑了它与PC和PC/XT的向上兼容性。PC/AT允许使用许多PC选件板，MS-DOS2.0支持的大多数软件也都可以在执行MS-DOS3.0的PC/AT上运行。

（1）硬件兼容性

PC/AT可以使用许多PC选件板，但由于PC/AT具有较高的主频，因而使一些PC选件不能在PC/AT上工作。对于这些电路板，IBM公司提供了相应的PC/AT选件板。常用的PC选件板的兼容性如下：

兼容的PC选件板	不兼容的PC选件板
IBM单色显示/打印控制板 IBM彩色/图形控制板 IBM SDLC通讯接口板 IBM BSC通讯接口板 PC网络接口板	存贮器扩充板 多功能接口板 异步通讯接口板 并行打印机接口板

（2）软件兼容性

虽然大多数PC软件都能在PC/AT上运行，但由于PC/AT与PC之间存在着一些差异，使得绕过DOS直接访问机器硬件的一些软件不能在PC/AT上正确运行。造成软件不兼容的原因主要有：

* PC/AT的时钟速率较快，一些利用软件计时的教学软件和游戏软件不能正确运行。

* PC/AT的键盘与PC不兼容，直接访问键盘的游戏软件不能运行。

* PC/AT使用的协处理器80287与8087不兼容，使用8087的软件必须重新汇编或编译后才能运行。

* 一些具有防止复制措施的PC软件不能在PC/AT上启动。

常用PC软件的兼容性如下表所示：

兼容的PC软件	不兼容的PC软件
Word Star	EasyWriter
dBASE Ⅱ和dBASE Ⅲ	BASIC 2.0
SuPerCalc2	VisiCalc 1.0
MultiPlan和Multimate	CP/M 86
Lotus 1-2-3	UCSD P-系统1.0
CI-C86	一些游戏软件
Lattice C	

附录Ⅱ IBM PC/5550简介

1. 概述

IBM PC/5550是一种中文个人计算机，它具有较强的中文处理能力和较好的图形显示功能。PC/5550主要具备以下特点：

* 能方便地处理中文数据，既能显示又能打印。
* 图形显示器的分辨率较高，图形模式下的分辨率可达1024×768。
* 机器结构较为紧凑，充分利用人机工程学中的先进技术，人机接口较好。
* 具有较丰富的中文处理软件，配有中文DOS、中文Multiplan、中文dBASE Ⅱ和中文BASIC等多种能处理中文数据的软件。

PC/5550分成中文5550和日文5550两类，每类又分成16×16字模系统和24×24字模系统两种型号。这里介绍的是24×24字模的中文5550。

2. 系统配置

PC/5550由主机箱、监视器、键盘和打印机等四部分组成。其中主机箱中安装了一块系统板、一只8.1MB的硬盘驱动器和一只720KB软盘驱动器。5550的典型配置如下：

15吋单色图形显示器 中文键盘 24针中文打印机 RS232C异步通讯器
Intel8086CPU（主频8MHz） 16KB只读存贮器 256—640KB读写存贮器 I/O控制器和I/O通道 720KB的5吋软盘驱动器 8.1MB的硬盘驱动器

3. 系统板

5550的系统板分成中央处理器、只读存贮器、读写存贮器、输入输出控制器和输入输出通道（扩充槽）等五部分。其中中央处理器的核心是一片Intel8086微处理器芯片，它的工作频率为 8 MHz。8086是16位微处理器，它有16根数据线和20根地址线，寻址能力为 1 MB。8086的指令系统与8088完全兼容。

系统板上的存贮器有16KB只读存贮器（ROM BIOS）和256KB读写存贮器。使用三块128KB内存扩充板可将系统的读写存贮器扩充至640KB。

系统板上的输入输出控制电路包括键盘控制电路、显示控制器、软/硬盘控制器和打印控制电路等。此外，系统板上还有五个I/O扩充槽，它们可以插入选件板以扩充5550的功能。可选用的选件板有128KB内存扩充板和RS232C异步通讯板等。

4. 中文键盘

5550采用分离式键盘，键盘体内包含一个可调节音量的扬声器。键盘上的键分成数据键（白色）和功能键(灰色)两类。数据键用来输入汉字或字符，其中字母键上标有汉字部首。功能键用来完成特定的功能，其中许多键换用了中文名称，如下表所示：

5550	PC	5550	PC
〈PF1〉—〈PF10〉	〈F1〉—〈F10〉	〈换行〉	〈CR〉
〈选择键面〉	〈Alt〉	〈倒退〉	〈BS〉
〈插入〉	〈Ins〉	〈删除〉	〈Del〉
〈印出屏幕〉	〈PrtSc〉	Ctrl-〈印出屏幕〉	Ctrl-PrtSc
〈Pause〉	Ctrl-NumLock	〈选择键面〉-Pause	Ctrl-Break
Ctrl-〈选择键面〉-〈删除〉	Ctrl-Alt-Del	〈选择键面〉-〈大写〉	〈CapsLock〉
¥	\		

此外，5550还增加了一些功能键，其含义如下：

功能键	含义
〈汉字输入〉	进入汉字输入模式
〈英数〉	进入英文数字输入模式
〈半型〉	半个汉字位置字符模式
〈选择键面〉-〈全型〉	全汉字位置字符模式
〈空格1〉—〈空格6〉	选择输入汉字〈编号〉
〈学习〉	进入学习汉字编码状态
〈执行〉	内码输入结束符
〈前进〉	显示下六个汉字
〈内码输入〉	进入内码输入模式

5．显示器与打印机

（1）显示器

5550配置的是15吋单色图形显示器（555-BPI），每屏可以显示1025个（41字×25行）字模为24×24的汉字(占26×29点阵)，或2050(82符×25行)个字模为12×24的西文字符。汉字字体为仿宋体，字型较为美观。当显示器处于图形模式时，它的分辨率为1024×768。

555-BPI显示器的屏幕为15吋的黄绿色屏幕，并采用长余辉和表面不反光技术，操作时不易疲倦。

（2）打印机

5550可以配置24针的中文打印机5553-BPI，它可以打印24×24点阵的汉字，该打印机的打印速度为40字/秒或60符/秒。

6．磁盘存贮器

5550的外存贮器包括一个5¼吋软盘驱动器和一个硬盘驱动器。软盘的容量为720KB，是普通PC软盘的两倍。硬盘的容量为8.1MB。使用5550的720KB的软盘驱动器可以用来读写普通的360KB的PC软盘。5550的双面双密倍轨软盘共两面，每面有80道，每道有9个扇区，每个扇区有512个字节。

7．中文DOS 2.2

5550上运行的中文DOS 2.2是在MS-DOS 2.1的基础上增加汉字处理功能而构成的。它具有中文信息输入输出、存贮和复制等功能。

中文DOS的汉字库中有国标一、二级库中的6768个汉字字模。该字模为24×24的码点，字形为仿宋体。此外，用户还可以使用造字程序自己定义500个汉字，这些自定义汉字的字模可以为26×29。

汉字的内码使用高位加1的国标区位码表示，这与PC上CCDOS使用的两字节内码有所不同，但可以使用一个变换程序将它们相互转换。中文DOS提供部首输入法和内码输入法等两种汉字输入法。

中文DOS 2.2与MS-DOS 2.1相比增加了一条SWITCH命令，并修改了几条与设备有关的命令。

中文DOS 2.2支持的中文软件主要有中文BASIC、宏汇编、FORTRAN、PASCAL、中文Multiplan和dBASE Ⅱ等。此外，中文DOS下还可以运行的西文软件有编译BASIC、COBOL WordStar、Multitool file和Multitool chart等。

许多PC用户程序，例如BASIC程序、FORTRAN程序和PASCAL程序等，都可以在5550上运行。

附录Ⅲ　IBM XT/370和IBM PC/3270简介

1. IBM XT/370

（1）概述

IBM XT/370是IBM公司在PC/XT基础上设计的一种个人计算机。它可以作为标准的PC/XT使用，也可以作为终端与IBM公司的中、大型机连接，还可以模拟运行IBM中型机系统/370的部分软件。

XT/370有PC/XT模式（PC模式）、IBM 3277终端模式（3277模式）和IBM 370工作站模式（370模式）等三种工作模式。它们的功能如下：

* PC模式　与标准PC/XT相同，可运行MS-DOS 2.0以及它所支持的各种软件。

* 3277模式　仿真IBM 3277终端，作为单色显示终端直接与IBM 370、43××或30××系列主机连接。它通过主机上的VM/SP分时操作系统，使用主机的软、硬件资源。

* 370模式　仿真IBM 370。具有480KB实存和4096KB虚存。它可以单机运行，利用VM/PC操作系统对IBM 370进行仿真；也可以通过一个Coax/3274连接器与IBM 370连接或连入IBM SNA网中。

（2）XT/370的组成

XT/370是在标准PC/XT(256KB内存)中插入三块专用扩充板后构成的。这三块扩充板是370处理器板（PC/370-P）、512KB存贮器板（PC/370-M）和3277显示器仿真板（PC/370EM），它们分别安装在4号、3号和2号扩充槽中。

① 处理器板PC/370-P

PC/370-P板包含三个微处理器芯片和一张页表。三个微处理器为一个标准的Motorola68000、一个专用的Motorola68000和一个专用的Intel8087，它们用来仿真执行IBM 370的机器指令。页表中的每项用来表示370 VM/CMS虚存的1页（4KB）。因为页表的长度为1024项，故XT/370虚存可达4MB。

XT/370将IBM 370的指令系统分成三组，分别用三个微处理器仿真执行。由IBM公司专门定做的68000，其微程序按IBM公司的要求作了修改，用来仿真大部分IBM 370定点指令，并完成取指令、译码等功能。该68000的寄存器用来表示IBM 370的通用寄存器和程序状态字PSW。

IBM 370的浮点指令使用一个微程序经过修改的协处理器8087来执行。该8087的浮点格式为IBM格式，而不是IEEE格式。8087的寄存器作为IBM 370的浮点寄存器使用。

PC/370-P板上的68000是标准的Motorola产品，它用来管理页表、处理意外事件以及执行部分IBM 370定点指令。在执行370指令前，必须通过RAM中的数据进行换码。

② 存贮器板PC/370-M

PC/370-M板包含512KB RAM，它既可以被系统板上的8088 CPU访问，也可以被370处理器板访问。8088的访问优先级高于370处理器板。

当使用8088访问PC/370-M板（运行MS-DOS）时，该板表示4000H—9FFFH之间的384KB存贮空间，加上系统板上的256KB RAM，共计640KB。当使用370处理器板访问

PC/370-M板（运行VM/PC）时，仅能访问前480KB存贮器（从0开始编址），余下的32KB存贮器用来存放标准68000所需的微码。

③ 3277仿真板PC/370EM

PC/370EM板用来仿真3277显示控制器，并用来与主机通讯，通过一个Coax/3274连接器可以与IBM的中大型机连接。

（3）VM/PC操作系统

XT/370可以运行标准的MS-DOS 2.0，也可以运行VM/PC。VM/PC是IBM公司分时操作系统VM/CMS的微机版本。当XT/370工作在PC模式时，只需要运行MS-DOS，而处在其它两种模式下则要同时运行VM/PC和MS-DOS。

MS-DOS在XT/370上完成类似IBM 370上VM监控程序的功能，进行文件处理、输入输出管理和通讯管理。370程序通过DIAGNOS指令调用MS-DOS的I/O功能（370上的该指令用来调用VM的功能）。

VM/PC除具有VM/CMS的功能外，还完成3277仿真、XT/370和IBM 370之间传送CMS文件、以及CMS文件与MS-DOS文件变换等功能。

当XT/370与IBM 370联机使用时，通过VM/PC，用户可以使用370主机上的软、硬件资源。此外，VM/PC还支持IBM公司的编辑程序XEDIT、命令处理程序EXEC2、动态调试程序DEBUG以及多种高级语言的编译程序。实际上，对实存和虚存的要求不大于416KB和4MB的370程序，原则上都可以在XT/370上运行。

2. IBM PC/3270

（1）概述

IBM PC/3270是另一种能与IBM大、中型机连接的个人计算机，它除了具有与IBM大、中型机通讯的能力外，还具有多窗口的彩色显示能力。PC/3270既适合需要使用微机的IBM大、中型机用户使用，也适合需要使用大、中型机的微机用户使用。

PC/3270有PC和3278工作站两种工作模式。在PC模式下，PC/3270类似一台标准的PC，可运行MS-DOS及其支持的各种软件。在3278模式下，它作为3278显示工作站直接连入IBM 43××或308×系列机的终端接口，这时可以在PC/3270的显示器上定义多达七个窗口，其中包括四个主机窗口、一个MS-DOS窗口和两个电子便笺窗口。

（2）PC/3270的组成

PC/3270是由插入了三块专用扩充板的主机，配上特制的键盘和监视器组成的。插入主机的三块扩充板是3270显示控制板、通讯控制板和键盘/时钟控制板。

型　号	屏幕大小	分辨率	颜　色	控制器型号
5272	14″	720×350	8	5151／5272控制板（带APA板）
5279	14″	720×512	8	5278显示控制板
彩色5378	19″	960×1000	16	彩色5378显示控制板
单色5378	19″	960×1000	4种灰度	单色5378显示控制板

① 显示控制器与监视器

PC/3270根据需要可以配置多种规格的显示控制板和监视器，它们的型号与参数如上页表格所示。

② 通讯控制板

通讯控制板用来控制PC/3270与IBM大、中型机的通讯接口，它通过四路电缆与一台或多台主机相连。在某一时刻它选择一路进行通讯。

③ 键盘/时钟控制板和键盘

键盘/时钟控制板的功能是提供I/O通道、显示控制板和通讯控制板所使用的时钟信息，并控制PC/3270的特制键盘。

PC/3270的键盘是结合IBM 3278终端的键盘和IBM PC的键盘的特点设计而成的，它可以作为PC键盘使用，也可以作为3278键盘使用。PC/3270的键盘上有122个键，其中有些键（如光标控制键）设置了两组，一组用于PC模式，另一组用于3278模式。

（3）PC/3270的控制程序与窗口操作

① PC/3270的控制程序

PC/3270的控制程序集成在一片门阵列芯片中，它用于管理窗口。PC/3270窗口软件通过调用该控制程序完成其功能。窗口软件的功能为设置窗口、复制窗口内容、保存和恢复屏幕映象文件（Profile）以及与主机进行通讯。

窗口软件在MS-DOS启动后自动引入，它和MS-DOS同时驻留内存工作，共同管理文件和窗口。

② 窗口的种类

一个PC/3270的窗口是在显示屏上的一个可视区域。通过窗口观察画面，就象通过窗子看世界一样，一次通常只能看到部分景物。一个窗口最大就是整个屏幕，最小可以仅占一个字符的位置。PC/3270最多允许在屏幕上开七个窗口。

PC/3270的窗口分成MS-DOS窗口、便笺窗口和主机窗口三类。MS-DOS窗口仅允许有一个（A），其上可运行普通的MS-DOS。便笺窗口可设置两个（B和C），它类似日常生活中使用的便笺。它可以用来打草稿、写备忘录和记录中间数据。

主机窗口最多允许设置四个（D—H），允许作为四台3278终端与一台或多台主机连接。在这些窗口中，可以分别调用四个主机程序运行，并可同时显示出它们的运行结果。

③ 窗口操作

除了设定窗口外，PC/3270还允许进行选择窗口、复制窗口和利用窗口复制文件等项操作。当窗口不能显示出整个屏幕的内容时，可以使用光标控制键移动窗口的位置。

窗口复制操作可以把源窗口中的一块信息复制到目的窗口中，它可以在三类窗口中任意执行但不允许目的窗口为MS-DOS窗口。

文件复制操作使用SEND和RECEIVE来实现，该操作是通过对应的窗口来实现的。

附录Ⅳ　通讯用汉字字符集（基本集）及其交换码国家标准（GB2312-80）

1．汉字的分级

本标准收汉字6763个，分成两级。第一级汉字3755个，置于16区至55区；第二级汉字3008个，置于56区至87区。

2．汉字的排列

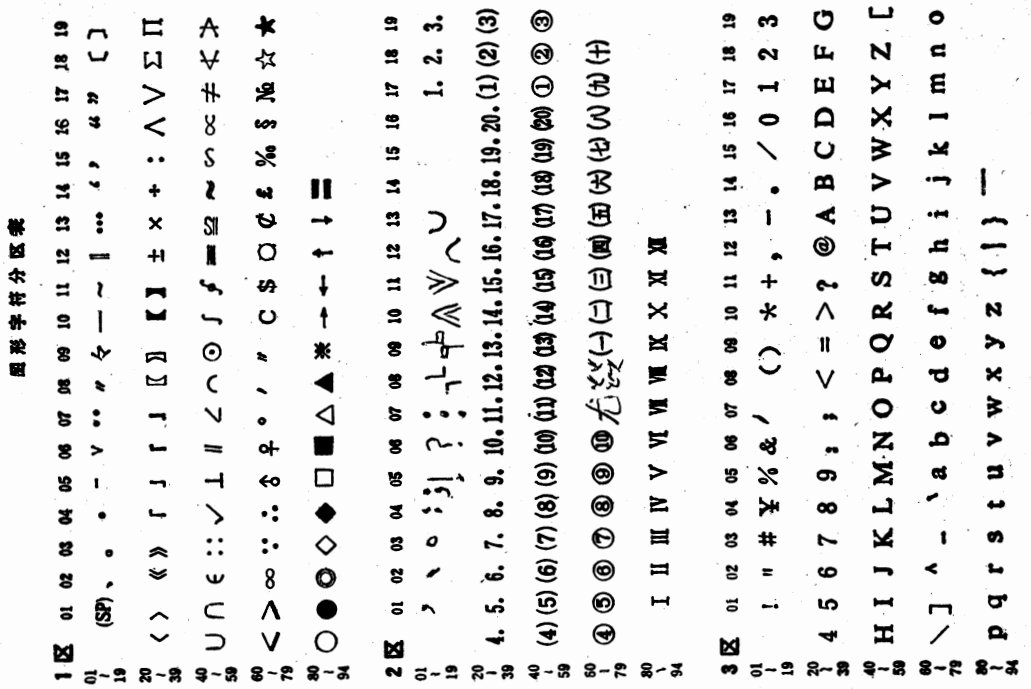
第一级汉字按汉语拼音字母顺序排列，同音字以笔形顺序横（一）、直（丨）、撇（丿）、点（丶）、折（乙，包括一、亅、乚等笔形）为序。起笔相同按第二笔，依此类推。

第二级汉字按部首排列，本标准采用的部首与一般字典用的部首基本相同，略有改并。部首次序及同部首字按画数排列，同画数的字以笔形顺序横（一）、直（丨）、撇（丿）、点（丶）、折（乙，包括一、亅、乚等笔形）为序。起笔相同按第二笔，依此类推。

上述排列方法及部首只适用于本标准。

汉字的字音以普通话审音委员会发表的《普通话异读词三次审音总表初稿》（1963年出版）为准，其中没有的字参考《汉字正字小字汇》编辑小组编的《汉字正字小字汇(初稿)》（文字改革出版社，1966年出版）、《新华字典》（1979年修订重排本）、《现代汉语词典》（中国社会科学院语言研究所词典编辑室编，1979年出版）及《辞海》（上海辞书出版社，1979年出版）。

汉字的字形以中华人民共和国文化部、中国文字改革委员会公布的《印刷通用汉字字形表》（1964年）为准，其中没有的字参考《新华字典》、《现代汉语词典》及《辞海》。



4区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01/19	ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く	ぐ	け	げ	こ
20/39	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た	だ	ち	ぢ	っ	つ	づ	て	で
40/59	と	ど	な	に	ぬ	ね	の	は	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ
60/79	ぼ	ぽ	ま	み	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
80/94	ゐ	ゑ	を	ん																

5区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01/19	ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク	グ	ケ	ゲ	コ
20/39	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ
40/59	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ
60/79	ボ	ポ	マ	ミ	ム	メ	モ	ヤ	ヤ	ユ	ユ	ヨ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
80/94	ヰ	ヱ	ヲ	ン	ヴ	カ	ケ													

6区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01/19		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ
20/39	Υ	Φ	Χ	Ψ	Ω									α	β	γ	δ	ε	ζ	η
40/59	θ	ι	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ	φ	χ	ψ	ω			
60/79																				
80/94																				

7区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01/19		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С
20/39	Т	У	Х	Ф	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я						
40/59									а	б	в	г	д	е	ё	ж	з	и	й	
60/79	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
80/94	ю	я																		

8区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01/19		ā	á	ǎ	à	ē	é	ě	è	ī	í	ǐ	ì	ō	ó	ǒ	ò	ū	ú	ǔ
20/39	ù	ǖ	ǘ	ǚ	ǜ	ü	e											ㄅ	ㄆ	ㄇ
40/59	ㄈ	ㄉ	ㄊ	ㄋ	ㄌ	ㄍ	ㄎ	ㄏ	ㄐ	ㄑ	ㄒ	ㄓ	ㄔ	ㄕ	ㄖ	ㄗ	ㄘ	ㄙ	ㄚ	ㄛ
60/79	ㄜ	ㄝ	ㄞ	ㄟ	ㄠ	ㄡ	ㄢ	ㄣ	ㄤ	ㄥ	ㄦ	ㄧ	ㄨ	ㄩ						
80/94																				

9区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01/19				—	—	|	|	···	···	⋮	⋮	·····	·····	⋮	⋮	⌐	⌐	⌐	⌐
20/39	¬	¬	¬	¬	└	└	└	└	┘	┘	┘	┘	├	├	├	├	├	├	├	├
40/59	┤	┤	┤	┤	┤	┤	┤	┤	┬	┬	┬	┬	┬	┬	┬	┬	┴	┴	┴	┴
60/79	┴	┴	┴	┴	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼
80/94																				

16区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		a 啊	阿	ai 埃	挨	哎	唉	哀	皑	癌	蔼	矮	艾	碍	爱	隘	an 鞍	氨	安	俺	A
20~39	按	暗	岸	胺	案	ang 肮	昂	盎	ao 凹	敖	熬	翱	袄	傲	奥	懊	澳	ba 芭	捌	扒	B
40~59	叭	吧	笆	八	疤	巴	拔	跋	靶	把	耙	坝	霸	罢	爸	bai 白	柏	百	摆	佰	
60~79	败	拜	稗	ban 斑	班	搬	扳	般	颁	板	版	扮	拌	伴	瓣	半	办	绊	bang 邦	帮	
80~94	梆	榜	膀	绑	棒	磅	蚌	镑	傍	谤	bao 苞	胞	包	褒	剥						

17区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		薄	雹	保	堡	饱	宝	抱	报	暴	豹	鲍	爆	bei 杯	碑	悲	卑	北	辈	背	
20~39	贝	钡	倍	狈	备	惫	焙	被	ben 奔	苯	本	笨	beng 崩	绷	甭	泵	蹦	迸	bi 逼	鼻	
40~59	比	鄙	笔	彼	碧	蓖	蔽	毕	毙	毖	币	庇	痹	闭	敝	i 弊	必	辟	壁	臂	
60~79	避	陛	bian 鞭	边	编	贬	扁	便	变	卞	辨	辩	辫	遍	biao 标	彪	膘	表	bie 鳖	憋	
80~94	别	瘪	bin 彬	斌	濒	滨	宾	摈	bing 兵	冰	柄	丙	秉	饼	炳						

18区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		病	并	bo 玻	菠	播	拨	钵	波	博	勃	搏	铂	箔	伯	帛	舶	脖	膊	渤	
20~39	泊	驳	bu 捕	卜	哺	补	埠	不	布	步	簿	部	怖	ca 擦	cai 猜	裁	材	才	财	睬	C
40~59	踩	采	彩	菜	蔡	can 餐	参	蚕	残	惭	惨	灿	cang 苍	舱	仓	沧	藏	cao 操	糙	槽	
60~79	曹	草	ce 厕	策	侧	册	测	ceng 层	蹭	cha 插	叉	茬	茶	查	碴	搽	察	岔	差	诧	
80~94	chai 拆	柴	豺	chan 搀	掺	蝉	馋	谗	缠	铲	产	阐	颤	chang 昌	猖						

19区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		场	尝	常	长	偿	肠	厂	敞	畅	唱	倡	chao 超	抄	钞	朝	嘲	潮	巢	吵	
20~39	炒	che 车	扯	撤	掣	彻	澈	chen 郴	臣	辰	尘	晨	忱	沉	陈	趁	衬	cheng 撑	称	城	
40~59	橙	成	呈	乘	程	惩	澄	诚	承	逞	骋	秤	chi 吃	痴	持	匙	池	迟	弛	驰	
60~79	耻	齿	侈	尺	赤	翅	斥	炽	chong 充	冲	虫	崇	宠	chou 抽	酬	畴	踌	稠	愁	筹	
80~94	仇	绸	瞅	丑	臭	chu 初	出	橱	厨	躇	锄	雏	滁	除	楚						

20区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		础	储	矗	搐	触	处	chua 揣	chuan 川	穿	椽	传	船	喘	串	chuang 疮	窗	幢	床	闯	
20~39	创	chui 吹	炊	捶	锤	垂	chun 春	椿	醇	唇	淳	纯	蠢	chuo 戳	绰	ci 疵	茨	磁	雌	辞	
40~59	慈	瓷	词	此	刺	赐	次	cong 聪	葱	囱	匆	从	丛	cou 凑	cu 粗	醋	簇	促	cuan 蹿	篡	
60~79	窜	cui 摧	崔	催	脆	瘁	粹	淬	翠	cun 村	存	寸	cuo 磋	撮	搓	措	挫	错	da 搭	达	D
80~94	答	瘩	打	大	dai 呆	歹	傣	戴	带	殆	代	贷	袋	待	逮						

21区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		怠	dan 耽	担	丹	单	郸	掸	胆	旦	氮	但	惮	淡	诞	弹	蛋	dang 当	挡	党	
20~39	荡	档	dao 刀	捣	蹈	倒	岛	祷	导	到	稻	悼	道	盗	de 德	得	的	deng 蹬	灯	登	
40~59	等	瞪	凳	邓	di 堤	低	滴	迪	敌	笛	狄	涤	翟	嫡	抵	底	地	蒂	第	帝	
60~79	弟	递	缔	dian 颠	掂	滇	碘	点	典	靛	垫	电	佃	甸	店	惦	奠	淀	殿	diao 碉	
80~94	叼	雕	凋	刁	掉	吊	钓	调	die 跌	爹	碟	蝶	迭	谍	叠						

22区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		丁(ding)	盯	叮	钉	顶	鼎	锭	定	订	丢(diu)	东(dong)	冬	董	懂	动	栋	侗	恫	冻
20~39	洞	兜(dou)	抖	斗	陡	豆	逗	痘	都(du)	督	毒	犊	独	读	堵	睹	赌	杜	镀	肚
40~59	度	渡	妒	端(duan)	短	锻	段	断	缎	堆(dui)	兑	队	对	墩(dun)	吨	蹲	敦	顿	囤	钝
60~79	盾	遁	掇(duo)	哆	多	夺	垛	躲	朵	跺	舵	剁	惰	堕	蛾(e)	峨	鹅	俄	额	讹 E
80~94	娥	恶	厄	扼	遏	鄂	饿	恩(en)	而(er)	儿	耳	尔	饵	洱	二					

23区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		贰	发(fa)	罚	筏	伐	乏	阀	法	珐	藩(fan)	帆	番	翻	樊	矾	钒	繁	凡	烦 F
20~39	反	返	范	贩	犯	饭	泛	坊(fang)	芳	方	肪	房	防	妨	仿	访	纺	放	菲(fei)	非
40~59	啡	飞	肥	匪	诽	吠	肺	废	沸	费	芬(fen)	酚	吩	氛	分	纷	坟	焚	汾	粉
60~79	奋	份	忿	愤	粪	丰(feng)	封	枫	蜂	峰	锋	风	疯	烽	逢	冯	缝	讽	奉	凤
80~94	佛(fo)	否(fou)	夫(fu)	敷	肤	孵	扶	拂	辐	幅	氟	符	伏	俘	服					

24区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		浮	涪	福	袱	弗	甫	抚	辅	俯	釜	斧	脯	腑	府	腐	赴	副	覆	赋
20~39	复	傅	付	阜	父	腹	负	富	讣	附	妇	缚	咐(ga)	噶	嘎	该(gai)	改	概	钙	盖 G
40~59	溉	干(gan)	甘	杆	柑	竿	肝	赶	感	秆	敢	赣	冈(gang)	刚	钢	缸	肛	纲	岗	港
60~79	杠	篙(gao)	皋	高	膏	羔	糕	搞	镐	稿	告	哥(ge)	歌	搁	戈	鸽	胳	疙	割	革
80~94	葛	格	蛤	阁	隔	铬	个	各	给(gei)	根(gen)	跟	耕(geng)	更	庚	羹					

25区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		埂	耿	梗	工(gong)	攻	功	恭	龚	供	躬	公	宫	弓	巩	汞	拱	贡	共	钩(gou)
20~39	勾	沟	苟	狗	垢	构	购	够	辜(gu)	菇	咕	箍	估	沽	孤	姑	鼓	古	蛊	骨
40~59	谷	股	故	顾	固	雇	刮(gua)	瓜	剐	寡	挂	褂	乖(guai)	拐	怪	棺(guan)	关	官	冠	观
60~79	管	馆	罐	惯	灌	贯	光(guang)	广	逛	瑰(gui)	规	圭	硅	归	龟	闺	轨	鬼	诡	癸
80~94	桂	柜	跪	贵	刽	辊(gun)	滚	棍	锅(guo)	郭	国	果	裹	过	哈(ha)					H

26区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		骸	孩(hai)	海	氦	亥	害	骇	酣(han)	憨	邯	韩	含	涵	寒	函	喊	罕	翰	撼
20~39	捍	旱	憾	悍	焊	汗	汉	夯(hang)	杭	航	壕(hao)	嚎	豪	毫	郝	好	耗	号	浩	呵(he)
40~59	喝	荷	菏	核	禾	和	何	合	盒	貉	阂	河	涸	赫	褐	鹤	贺	嘿(hei)	黑	痕(hen)
60~79	很	狠	恨	哼(heng)	亨	横	衡	恒	轰(hong)	哄	烘	虹	鸿	洪	宏	弘	红	喉(hou)	侯	猴
80~94	吼	厚	候	后	呼(hu)	乎	忽	瑚	壶	葫	胡	蝴	狐	糊	湖					

27区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01~19		弧	虎	唬	护	互	沪	户	花(hua)	哗	华	猾	滑	画	划	化	话	槐(huai)	徊	怀
20~39	淮	坏	欢(huan)	环	桓	还	缓	换	患	唤	痪	豢	焕	涣	宦	幻	荒(huang)	慌	黄	磺
40~59	蝗	簧	皇	凰	惶	煌	晃	幌	恍	谎	灰(hui)	挥	辉	徽	恢	蛔	回	毁	悔	慧
60~79	卉	惠	晦	贿	秽	会	烩	汇	讳	诲	绘	荤(hun)	昏	婚	魂	浑	混	豁(huo)	活	伙
80~94	火	获	或	惑	霍	货	祸	击(ji)	圾	基	机	畸	稽	积	箕					J

28区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		肌	饥	迹	激	讥	鸡	姬	绩	缉	吉	极	棘	辑	籍	集	及	急	疾	汲
20~39	即	嫉	级	挤	几	脊	己	蓟	技	冀	季	伎	祭	剂	悸	济	寄	寂	计	记
40~59	既	忌	际	妓	继	纪	jia 嘉	枷	夹	佳	家	加	荚	颊	贾	甲	钾	假	稼	价
60~79	架	驾	嫁	jian 歼	监	坚	尖	笺	间	煎	兼	肩	艰	奸	缄	茧	检	柬	碱	硷
80~94	拣	捡	简	俭	剪	减	荐	槛	鉴	践	贱	见	键	箭	件					

29区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		健	舰	剑	饯	渐	溅	涧	建	jiang 僵	姜	将	浆	江	疆	蒋	桨	奖	讲	匠
20~39	酱	降	jiao 蕉	椒	礁	焦	胶	交	郊	浇	骄	娇	嚼	搅	铰	矫	侥	脚	狡	角
40~59	饺	缴	绞	剿	教	酵	轿	较	叫	窖	jie 揭	接	皆	秸	街	阶	截	劫	节	桔
60~79	杰	捷	睫	竭	洁	结	解	姐	戒	藉	芥	界	借	介	疥	诫	届	jin 巾	筋	斤
80~94	金	今	津	襟	紧	锦	仅	谨	进	靳	晋	禁	近	烬	浸					

30区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		尽	劲	jing 荆	兢	茎	睛	晶	鲸	京	惊	精	粳	经	井	警	景	颈	静	境
20~39	敬	镜	径	痉	靖	竟	竞	净	jiong 炯	窘	jiu 揪	究	纠	玖	韭	久	灸	九	酒	厩
40~59	救	旧	臼	舅	咎	就	疚	jiu 鞠	拘	狙	疽	居	驹	菊	局	咀	矩	举	沮	聚
60~79	拒	据	巨	具	距	踞	锯	俱	句	惧	炬	剧	juan 捐	鹃	娟	倦	眷	卷	绢	jue 撅
80~94	攫	抉	掘	倔	爵	觉	决	诀	绝	jun 均	菌	钧	军	君	峻					

31区

	01	02	03	04	05	ka 06	07	08	09	kai 10	11	12	13	14	kan 15	16	17	18	19
01~19		俊	竣	浚	郡	骏	喀	咖	卡	咯	开	揩	楷	凯	慨	刊	堪	勘	坎	砍	K
20~39	看	kang 康	慷	糠	扛	抗	亢	炕	kao 考	拷	烤	靠	ke 坷	苛	柯	棵	磕	颗	科	壳
40~59	咳	可	渴	克	刻	客	课	ken 肯	啃	垦	恳	keng 坑	吭	kong 空	恐	孔	控	kou 抠	口	扣
60~79	寇	ku 枯	哭	窟	苦	酷	库	裤	kua 夸	垮	挎	跨	胯	kuai 块	筷	侩	快	kuan 宽	款	kuang 匡
80~94	筐	狂	框	矿	眶	旷	况	kui 亏	盔	岿	窥	葵	奎	魁	傀					

32区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		馈	愧	溃	kun 坤	昆	捆	困	kuo 括	扩	廓	阔	la 垃	拉	喇	蜡	腊	辣	啦	lai 莱	L
20~39	来	赖	lan 蓝	婪	栏	拦	篮	阑	兰	澜	谰	揽	览	懒	缆	烂	滥	long 琅	榔	狼
40~59	廊	郎	朗	浪	lao 捞	劳	牢	老	佬	姥	酪	烙	涝	le 勒	乐	lei 雷	镭	蕾	磊	累
60~79	儡	垒	擂	肋	类	泪	leng 棱	楞	冷	li 厘	梨	犁	黎	篱	狸	离	漓	理	李	里
80~94	鲤	礼	莉	荔	吏	栗	丽	厉	励	砾	历	利	傈	例	俐					

33区

	01	02	03	04	05	06	07	08	lia 09	lian 10	11	12	13	14	15	16	17	18	19
01~19		痢	立	粒	沥	隶	力	璃	哩	俩	联	莲	连	镰	廉	怜	涟	帘	敛	脸
20~39	链	恋	炼	练	liang 粮	凉	梁	粱	良	两	辆	量	晾	亮	谅	liao 撩	聊	僚	疗	燎
40~59	寥	辽	潦	了	撂	镣	廖	料	lie 列	裂	烈	劣	猎	lin 琳	林	磷	霖	临	邻	鳞
60~79	淋	凛	赁	吝	ling 拎	玲	菱	零	龄	铃	伶	羚	凌	灵	陵	岭	领	另	令	liu 溜
80~94	琉	榴	硫	馏	留	刘	瘤	流	柳	六	long 龙	聋	咙	笼	窿					

34区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		隆	垄	拢	陇	楼	娄	搂	篓	漏	陋	芦	卢	颅	庐	炉	掳	卤	虏	鲁
20~39	麓	碌	露	路	赂	鹿	潞	禄	录	陆	戮	驴	吕	铝	侣	旅	履	屡	缕	虑
40~59	氯	律	率	滤	绿	峦	挛	孪	滦	卵	乱	掠	略	抡	轮	伦	仑	沦	纶	论
60~79	萝	螺	罗	逻	锣	箩	骡	裸	落	洛	骆	络	妈	麻	玛	码	蚂	马	骂	嘛	M
80~94	吗	埋	买	麦	卖	迈	脉	瞒	馒	蛮	满	蔓	曼	慢	漫					

35区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		谩	芒	茫	盲	氓	忙	莽	猫	茅	锚	毛	矛	铆	卯	茂	冒	帽	貌	贸
20~39	么	玫	枚	梅	酶	霉	煤	没	眉	媒	镁	每	美	昧	寐	妹	媚	门	闷	们
40~59	萌	蒙	檬	盟	锰	猛	梦	孟	眯	醚	靡	糜	迷	谜	弥	米	秘	觅	泌	蜜
60~79	密	幂	棉	眠	绵	冕	免	勉	娩	缅	面	苗	描	瞄	藐	秒	渺	庙	妙	蔑
80~94	灭	民	抿	皿	敏	悯	闽	明	螟	鸣	铭	名	命	谬	摸					

36区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		摹	蘑	模	膜	磨	摩	魔	抹	末	莫	墨	默	沫	漠	寞	陌	谋	牟	某
20~39	拇	牡	亩	姆	母	墓	暮	幕	募	慕	木	目	睦	牧	穆	拿	哪	呐	钠	那	N
40~59	娜	纳	氖	乃	奶	耐	奈	南	男	难	囊	挠	脑	恼	闹	淖	呢	馁	内	嫩
60~79	能	妮	霓	倪	泥	尼	拟	你	匿	腻	逆	溺	蔫	拈	年	碾	撵	捻	念	娘
80~94	酿	鸟	尿	捏	聂	孽	啮	镊	镍	涅	您	柠	狞	凝	宁					

37区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		拧	泞	牛	扭	钮	纽	脓	浓	农	弄	奴	努	怒	女	暖	虐	疟	挪	懦
20~39	糯	诺	哦	欧	鸥	殴	藕	呕	偶	沤	啪	趴	爬	帕	怕	琶	拍	排	牌	徘	OP
40~59	湃	派	攀	潘	盘	磐	盼	畔	判	叛	乓	庞	旁	耪	胖	抛	咆	刨	炮	袍
60~79	跑	泡	呸	胚	培	裴	赔	陪	配	佩	沛	喷	盆	砰	抨	烹	澎	彭	蓬	棚
80~94	硼	篷	膨	朋	鹏	捧	碰	坯	砒	霹	批	披	劈	琵	毗					

38区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		啤	脾	疲	皮	匹	痞	僻	屁	譬	篇	偏	片	骗	飘	漂	瓢	票	撇	瞥
20~39	拼	频	贫	品	聘	乒	坪	苹	萍	平	凭	瓶	评	屏	坡	泼	颇	婆	破	魄
40~59	迫	粕	剖	扑	铺	仆	莆	葡	菩	蒲	埔	朴	圃	普	浦	谱	曝	瀑	期	欺	Q
60~79	栖	戚	妻	七	凄	漆	柒	沏	其	棋	奇	歧	畦	崎	脐	齐	旗	祈	祁	骑
80~94	起	岂	乞	企	启	契	砌	器	气	迄	弃	汽	泣	讫	掐					

39区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		恰	洽	牵	扦	钎	铅	千	迁	签	仟	谦	乾	黔	钱	钳	前	潜	遣	浅
20~39	谴	堑	嵌	欠	歉	枪	呛	腔	羌	墙	蔷	强	抢	橇	锹	敲	悄	桥	瞧	乔
40~59	侨	巧	鞘	撬	翘	峭	俏	窍	切	茄	且	怯	窃	钦	侵	亲	秦	琴	勤	芹
60~79	擒	禽	寝	沁	青	轻	氢	倾	卿	清	擎	晴	氰	情	顷	请	庆	琼	穷	秋
80~94	丘	邱	球	求	囚	酋	泅	趋	区	蛆	曲	躯	屈	驱	渠					

40区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		取	娶	龋	趣	去	quan 圈	颧	权	醛	泉	全	痊	拳	犬	券	劝	que 缺	炔	瘸
20~39	却	鹊	榷	确	雀	qun 裙	群	ran 然	燃	冉	染	rang 瓤	壤	攘	嚷	让	rao 饶	扰	绕	re 惹 R
40~59	热	ren 壬	仁	人	忍	韧	任	认	刃	妊	纫	reng 扔	仍	日	rong 戎	茸	蓉	荣	融	熔
60~79	溶	容	绒	冗	rou 揉	柔	肉	ru 茹	蠕	儒	孺	如	辱	乳	汝	入	褥	ruan 软	阮	rui 蕊
80~94	瑞	锐	run 闰	润	ruo 若	弱	sa 撒	洒	萨	sai 腮	鳃	塞	赛	san 三	叁					

41区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		伞	散	sang 桑	嗓	丧	sao 搔	骚	扫	嫂	se 瑟	色	涩	sen 森	seng 僧	sha 莎	砂	杀	刹	沙
20~39	纱	傻	啥	煞	shai 筛	晒	shan 珊	苫	杉	山	删	煽	衫	闪	陕	擅	赡	膳	善	汕
40~59	扇	缮	shang 墒	伤	商	赏	晌	上	尚	裳	shao 梢	捎	稍	烧	芍	勺	韶	少	哨	邵
60~79	绍	she 奢	赊	蛇	舌	舍	赦	摄	射	慑	涉	社	设	shen 砷	申	呻	伸	身	深	娠
80~94	绅	神	沈	审	婶	甚	肾	慎	渗	sheng 声	生	甥	牲	升	绳					

42区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		省	盛	剩	胜	圣	shi 师	失	狮	施	湿	诗	尸	虱	十	石	拾	时	什	食
20~39	蚀	实	识	史	矢	使	屎	驶	始	式	示	士	世	柿	事	拭	誓	逝	势	是
40~59	嗜	噬	适	仕	侍	释	饰	氏	市	恃	室	视	试	shou 收	手	首	守	寿	授	售
60~79	受	瘦	兽	shu 蔬	枢	梳	殊	抒	输	叔	舒	淑	疏	书	赎	孰	熟	薯	暑	曙
80~94	署	蜀	黍	鼠	属	术	述	树	束	戍	竖	墅	庶	数	漱					

43区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		恕	shua 刷	耍	摔	shuai 衰	甩	帅	shuan 栓	拴	shuang 霜	双	爽	shui 谁	水	睡	税	shun 吮	瞬	顺
20~39	舜	shuo 说	硕	朔	烁	si 斯	撕	嘶	思	私	司	丝	死	肆	寺	嗣	四	伺	似	饲
40~59	巳	song 松	耸	怂	颂	送	宋	讼	诵	sou 搜	艘	擞	嗽	su 苏	酥	俗	素	速	粟	僳
60~79	塑	溯	宿	诉	肃	suan 酸	蒜	算	sui 虽	隋	随	绥	髓	碎	岁	穗	遂	隧	祟	sun 孙
80~94	损	笋	suo 蓑	梭	唆	缩	琐	索	锁	所	ta 塌	他	它	她	塔					T

44区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		獭	挞	蹋	踏	tai 胎	苔	抬	台	泰	酞	太	态	汰	tan 坍	摊	贪	瘫	滩	坛
20~39	檀	痰	潭	谭	谈	坦	毯	袒	碳	探	叹	炭	tang 汤	塘	搪	堂	棠	膛	唐	糖
40~59	倘	躺	淌	趟	烫	tao 掏	涛	滔	绦	萄	桃	逃	淘	陶	讨	套	te 特	teng 藤	腾	疼
60~79	誊	ti 梯	剔	踢	锑	提	题	蹄	啼	体	替	嚏	惕	涕	剃	屉	tian 天	添	填	田
80~94	甜	恬	舔	腆	tiao 挑	条	迢	眺	跳	tie 贴	铁	帖	ting 厅	听	烃					

45区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		汀	廷	停	亭	庭	挺	艇	tong 通	桐	酮	瞳	同	铜	彤	童	桶	捅	筒	统
20~39	痛	tou 偷	投	头	透	tu 凸	秃	突	图	徒	途	涂	屠	土	吐	兔	ttuan 湍	团	tui 推	颓
40~59	腿	蜕	褪	退	tun 吞	屯	臀	tuo 拖	托	脱	鸵	陀	驮	驼	椭	妥	拓	唾	wa 挖	哇 W
60~79	蛙	洼	娃	瓦	袜	wai 歪	外	wan 豌	弯	湾	玩	顽	丸	烷	完	碗	挽	晚	皖	惋
80~94	宛	婉	万	腕	wang 汪	王	亡	枉	网	往	旺	望	忘	妄	wei 威					

46区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		巍	微	危	韦	违	桅	围	唯	惟	为	潍	维	苇	萎	委	伟	伪	尾	纬
20~39	未	蔚	味	畏	胃	喂	魏	位	渭	谓	尉	慰	卫	瘟(wen)	温	蚊	文	闻	纹	吻
40~59	稳	紊	问	嗡(weng)	翁	瓮	挝(wo)	蜗	涡	窝	我	斡	卧	握	沃	巫(wu)	呜	钨	乌	污
60~79	诬	屋	无	芜	梧	吾	吴	毋	武	五	捂	午	舞	伍	侮	坞	戊	雾	晤	物
80~94	勿	务	悟	误	昔(xi)	熙	析	西	硒	矽	晰	嘻	吸	锡	牺					X

47区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		稀	息	希	悉	膝	夕	惜	熄	烯	溪	汐	犀	檄	袭	席	习	媳	喜	铣
20~39	洗	系	隙	戏	细	瞎(xia)	虾	匣	霞	辖	暇	峡	侠	狭	下	厦	夏	吓	掀(xian)	锨
40~59	先	仙	鲜	纤	咸	贤	衔	舷	闲	涎	弦	嫌	显	险	现	献	县	腺	馅	羡
60~79	宪	陷	限	线	相(xiang)	厢	镶	香	箱	襄	湘	乡	翔	祥	详	想	响	享	项	巷
80~94	橡	像	向	象	萧(xiao)	硝	霄	削	哮	嚣	销	消	宵	淆	晓					

48区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		小	孝	校	肖	啸	笑	效	楔(xie)	些	歇	蝎	鞋	协	挟	携	邪	斜	胁	谐
20~39	写	械	卸	蟹	懈	泄	泻	谢	屑	薪(xin)	芯	锌	欣	辛	新	忻	心	信	衅	星(xing)
40~59	腥	猩	惺	兴	刑	型	形	邢	行	醒	幸	杏	性	姓	兄(xiong)	凶	胸	匈	汹	雄
60~79	熊	休(xiu)	修	羞	朽	嗅	锈	秀	袖	绣	墟(xu)	戌	需	虚	嘘	须	徐	许	蓄	酗
80~94	叙	旭	序	畜	恤	絮	婿	绪	续	轩(xuan)	喧	宣	悬	旋	玄					

49区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		选	癣	眩	绚	靴(xue)	薛	学	穴	雪	血	勋(xun)	熏	循	旬	询	寻	驯	巡	殉
20~39	汛	训	讯	逊	迅	压(ya)	押	鸦	鸭	呀	丫	芽	牙	蚜	崖	衙	涯	雅	哑	亚
40~59	讶	焉(yan)	咽	阉	烟	淹	盐	严	研	蜒	岩	延	言	颜	阎	炎	沿	奄	掩	眼
60~79	衍	演	艳	堰	燕	厌	砚	雁	唁	彦	焰	宴	谚	验	殃(yang)	央	鸯	秧	杨	扬
80~94	佯	疡	羊	洋	阳	氧	仰	痒	养	样	漾	邀(yao)	腰	妖	瑶					

50区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		摇	尧	遥	窑	谣	姚	咬	舀	药	要	耀	椰(ye)	噎	耶	爷	野	冶	也	页
20~39	掖	业	叶	曳	腋	夜	液	一(yi)	壹	医	揖	铱	依	伊	衣	颐	夷	遗	移	仪
40~59	胰	疑	沂	宜	姨	彝	椅	蚁	倚	已	乙	矣	以	艺	抑	易	邑	屹	亿	役
60~79	臆	逸	肄	疫	亦	裔	意	毅	忆	义	益	溢	诣	议	谊	译	异	翼	翌	绎
80~94	茵	荫(yin)	因	殷	音	阴	姻	吟	银	淫	寅	饮	尹	引	隐					

51区	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		印	英(ying)	樱	婴	鹰	应	缨	莹	萤	营	荧	蝇	迎	赢	盈	影	颖	硬	映
20~39	哟(yo)	拥(yong)	佣	臃	痈	庸	雍	踊	蛹	咏	泳	涌	永	恿	勇	用	幽(you)	优	悠	忧
40~59	尤	由	邮	铀	犹	油	游	酉	有	友	右	佑	釉	诱	又	幼	迂(yu)	淤	于	盂
60~79	榆	虞	愚	舆	余	俞	逾	鱼	愉	渝	渔	隅	予	娱	雨	与	屿	禹	宇	语
80~94	羽	玉	域	芋	郁	吁	遇	喻	峪	御	愈	欲	狱	育	誉					

52区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01-19		浴	寓	裕	预	豫	驭	鸳	渊	冤	元	垣	袁	原	援	辕	园	员	圆	猿
20-39	源	缘	远	苑	愿	怨	院	曰	约	越	跃	钥	岳	粤	月	悦	阅	耘	云	郧
40-59	匀	陨	允	运	蕴	酝	晕	韵	孕	匝	砸	杂	栽	哉	灾	宰	载	再	在	咱
60-79	攒	暂	赞	赃	脏	葬	遭	糟	凿	藻	枣	早	澡	蚤	躁	噪	造	皂	灶	燥
80-94	责	择	则	泽	贼	怎	增	憎	曾	赠	扎	喳	渣	札	轧					

53区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01-19		铡	闸	眨	栅	榨	咋	乍	炸	诈	摘	斋	宅	窄	债	寨	瞻	毡	詹	粘
20-39	沾	盏	斩	辗	崭	展	蘸	栈	占	战	站	湛	绽	樟	章	彰	漳	张	掌	涨
40-59	杖	丈	帐	账	仗	胀	瘴	障	招	昭	找	沼	赵	照	罩	兆	肇	召	遮	折
60-79	哲	蛰	辙	者	锗	蔗	这	浙	珍	斟	真	甄	砧	臻	贞	针	侦	枕	疹	诊
80-94	震	振	镇	阵	蒸	挣	睁	征	狰	争	怔	整	拯	正	政					

54区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01-19		帧	症	郑	证	芝	枝	支	吱	蜘	知	肢	脂	汁	之	织	职	直	植	殖
20-39	执	值	侄	址	指	止	趾	只	旨	纸	志	挚	掷	至	致	置	帜	峙	制	智
40-59	秩	稚	质	炙	痔	滞	治	窒	中	盅	忠	钟	衷	终	种	肿	重	仲	众	舟
60-79	周	州	洲	诌	粥	轴	肘	帚	咒	皱	宙	昼	骤	珠	株	蛛	朱	猪	诸	诛
80-94	逐	竹	烛	煮	拄	瞩	嘱	主	著	柱	助	蛀	贮	铸	筑					

55区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01-19		住	注	祝	驻	抓	爪	拽	专	砖	转	撰	赚	篆	桩	庄	装	妆	撞	壮
20-39	状	椎	锥	追	赘	坠	缀	谆	准	捉	拙	卓	桌	琢	茁	酌	啄	着	灼	浊
40-59	兹	咨	资	姿	滋	淄	孜	紫	仔	籽	滓	子	自	渍	字	鬃	棕	踪	宗	综
60-79	总	纵	邹	走	奏	揍	租	足	卒	族	祖	诅	阻	组	钻	纂	嘴	醉	最	罪
80-94	尊	遵	昨	左	佐	柞	做	作	坐	座										

56区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01-19		亍	丌	兀	丐	廿	卅	丕	亘	丞	鬲	孬	噩	丨	禹	丿	匕	毛	夭	爻	一丿
20-39	卮	氐	囟	胤	馗	毓	睾	鼗	丶	亟	鼐	乜	乩	亓	芈	孛	啬	嘏	仄	厍	丶乙(乛) 了(亅)二 十厂
40-59	厝	厣	厥	厮	厍	赝	匚	匝	匼	匮	匾	赜	卦	卣	刂	刈	刎	刭	刳	刿	匕卜(卜) 讠
60-79	剀	剌	剞	剡	剜	剔	剟	剕	剿	剽	剐	冂	冏	亻	仃	仉	仂	仨	仡	仫	冂亻
80-94	仞	伛	仳	伢	佤	仵	伥	伧	伉	伫	佞	佧	攸	佚	佝						

57区

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
01-19		佟	佗	佾	伽	佶	佴	侑	侉	侃	侏	佾	佻	侪	佼	侬	侔	俦	俨	俪	
20-39	俅	俚	俣	俜	俑	俟	俸	倩	偌	俳	倬	倏	倮	俸	倭	俾	倜	倌	倥	倨	傧
40-59	偃	偕	偈	偎	偬	偻	傥	傧	傩	儆	傺	僳	僦	僮	僭	僬	仝	余	人(入)		
60-79	余	金	俎	龠	汆	籴	兮	巽	赏	鼢	镢	夔	勹	匍	匐	匏	凫	夙	兕	亠	八(一)勹 儿(冂)儿 亠
80-94	兖	亳	衮	袤	衮	裔	袁	裹	褒	襄	瀛	瀛	冫	冱	冽	冼					

58区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01~19 凇冖冢冥讠讦讧讪讴讵讷诂诃诋诏诎诒诓诔

20~39 诖诘诙诜诟诠诤诨诩诮诰诳诶诹诼诿谀谂谄谇

40~59 谌谏谑谒谔谕谖谙谛谘谝谟谠谡谥谧谪谫谮谯

60~79 谲谳谵谶卩卺阝阢阡阱阪阽阼陂陉陔陟陧陬陲

80~94 陴隈隍隗隰邗邛邝邙邬邡邴邳邶邺

59区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01~19 邸邰郏郅邾郐郄郇郓郦郢郜郗郛郫郯郾鄄鄢

20~39 鄞鄣鄱鄯鄹酃酆刍奂劢劬劭劾哿勐勖勰叟燮矍

40~59 廴凵凼鬯厶弁畚巯坌垩垡塾墼壅壑圩圬圪圳圹

60~79 圮圯坜圻坂坩垅坫垆坼坻坨坭坶坳垭垤垌垲埏

80~94 垧垴垓垠埕埘埚埙埒垸埴埯埸埤埝

60区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01~19 堋堍埽埭堀堞堙塄堠塥塬墁墉墚墀馨鼙懿艹

20~39 艽艿芏芊芨芄芎芑芗芙芫芸芾芰苈苊苣芘芷芮

40~59 苋苌苁芩芴芡芪芟苄苎芤苡茉苷苤茏茇苜苴苒

60~79 苘茌苻苓茑茚茆茔茕苠苕茜荑荛荜茈莒茼茴茱

80~94 莛荞茯荏荇荃荟荀茗荠茭茺茳荦荥

61区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01~19 荨茛荩荬荪荭荮莰荸莳莴莠莪莓莜莅荼莶莩

20~39 荽莸荻莘莞莨莺莼菁萁菥菘堇萘萋菝菽菖萜萸

40~59 萑萆菔菟萏萃菸菹菪菅菀萦菰菡葜葑葚葙葳蒇

60~79 蒈葺蒉葸萼葆葩葶蒌蒎萱葭蓁蓍蓐蓦蒽蓓蓊蒿

80~94 蒺蓠蒡蒹蒴蒗蓥蓣蔌甍蔸蓰蔹蔟蔺

62区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01~19 蕖蔻蓿蓼蕙蕈蕨蕤蕞蕺瞢蕃蕲蕻薤薨薇薏蕹

20~39 薮薜薅薹薷薰藓藁藜藿蘧蘅蘩蘖蘼廾弈夼奁耷

40~59 奕奚奘匏尢尥尬尴扌扪抟抻拊拚拗拮挢拶挹捋

60~79 捃掭揶捱捺掎掴捭掬掊捩掮掼揲揸揠揿揄揞揎

80~94 摒揆掾摅摁搋搛搠搌搦搡摞撄摭撖

63区 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19

01~19 摺撷撸撙撺擀擐擗擤擢攉攥攮弋忒甙弑卟叱

20~39 叽叩叨叻吒吖吆呋呒呓呔呖呃吡呗呙吣吲咂咔

40~59 呷呱呤咚咛咄呶呦咝哐咭哂咴哒咧咦哓哔呲咣

60~79 哕咻咿哌哙哚哜咩咪咤哝哏哞唛哧唠哽唔哳唢

80~94 唣唏唑唧唪啧喏喵啉啭啁啕唿啐唼

64区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		啧	啖	啵	啶	啷	唳	唰	啜	喋	嗒	喃	喱	喹	喈	喁	喟	啾	嗖	喑
20~39	啻	嗟	喽	喾	喔	喙	嗪	嗷	嗉	嘟	嗑	嗫	嗬	嗔	嗦	嗝	嗄	嗯	嗥	嗲
40~59	嗳	嗌	嗍	嗨	嗵	嗤	辔	嘞	嘈	嘌	嘁	嘤	嘣	嗾	嘀	嘧	嘭	噘	嘹	噗
60~79	嘬	噍	噢	噙	噜	噌	噔	嚆	噤	噱	噫	噻	噼	嚅	嚓	嚯	囔	囗	囝	囡
80~94	囵	囫	囹	囿	圄	圊	圉	圜	帏	帙	帔	帑	帱	帻	帼					

65区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		帷	幄	幔	幛	幞	幡	岌	屺	岍	岐	岖	岈	岘	岙	岑	岚	岜	岵	岢
20~39	岽	岬	岫	岱	岣	峁	岷	峄	峒	峤	峋	峥	崂	崃	崧	崦	崮	崤	崞	崆
40~59	崛	嵘	崾	崴	崽	嵬	嵛	嵯	嵝	嵫	嵋	嵊	嵩	嵴	嶂	嶙	嶝	豳	嶷	巅
60~79	彳	彷	徂	徇	徉	後	徕	徙	徜	徨	徭	徵	徼	衢	彡	犭	犰	犴	犷	犸
80~94	狃	狁	狎	狍	狒	狨	狯	狩	狲	狴	狷	猁	狳	猃	狺					

66区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		狻	猗	猓	猡	猊	猞	猝	猕	猢	猹	猥	猬	猸	猱	獐	獍	獗	獠	獬
20~39	獯	獾	舛	夥	飧	夤	夂	饣	饧	饨	饩	饪	饫	饬	饴	饷	饽	馀	馄	馇
40~59	馊	馍	馐	馑	馓	馔	馕	庀	庑	庋	庖	庥	庠	庹	庵	庾	庳	赓	廒	廑
60~79	廛	廨	廪	膺	忄	忉	忖	忏	怃	忮	怄	忡	忤	忾	怅	怆	忪	忭	忸	怙
80~94	怵	怦	怛	怏	怍	怩	怫	怊	怿	怡	恸	恹	恻	恺	恂					

67区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		恪	恽	悖	悚	悭	悝	悃	悒	悌	悛	惬	悻	悱	惝	惘	惆	惚	悴	愠
20~39	愦	愕	愣	惴	愀	愎	愫	慊	慵	憬	憔	憧	憷	懔	懵	忝	隳	闩	闫	闱
40~59	闳	闵	闶	闼	闾	阃	阄	阆	阈	阊	阋	阌	阍	阏	阒	阕	阖	阗	阙	阚
60~79	丬	爿	戕	氵	汔	汜	汊	沣	沅	沐	沔	沌	汨	汩	汴	汶	沆	沩	泐	泔
80~94	沭	泷	泸	泱	泗	沲	泠	泖	泺	泫	泮	沱	泓	泯	泾					

68区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		洹	洧	洌	浃	浈	洇	洄	洙	洎	洫	浍	洮	洵	洚	浏	浒	浔	洳	涑
20~39	浯	涞	涠	浞	涓	涔	浜	浠	浼	浣	渚	淇	淅	淞	渎	涿	淠	渑	淦	淝
40~59	淙	渖	涫	渌	涮	渫	湮	湎	湫	溲	湟	溆	湓	湔	渲	渥	湄	滟	溱	溘
60~79	滠	漭	滢	溥	溧	溽	溻	溷	滗	溴	滏	溏	滂	溟	潢	潆	潇	漤	漕	滹
80~94	漯	漶	潋	潴	漪	漉	漩	澉	澍	澌	潸	潲	潼	潺	濑					

69区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		濉	澧	澹	澶	濂	濡	濮	濞	濠	濯	瀚	瀣	瀛	瀹	瀵	灏	灞	宀	宄
20~39	宕	宓	宥	宸	甯	骞	搴	寤	寮	褰	寰	蹇	謇	辶	迓	迕	迥	迮	迤	迩
40~59	迦	迳	迨	逅	逄	逋	逦	逑	逍	逖	逡	逵	逶	逭	逯	遄	遑	遒	遐	遨
60~79	遘	遢	遛	暹	遴	遽	邂	邈	邃	邋	彐	彗	彖	彘	尻	咫	屐	屙	孱	屣
80~94	屦	羼	弪	弩	弭	艴	弼	鬻	屮	妁	妃	妍	妩	妪	妣					

70区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		妗	姊	妫	妞	妤	姒	妲	妯	姗	妾	娅	娆	姝	娈	姣	姘	姹	娌	娉
20~39	娲	娴	娑	娣	娓	婀	婧	婊	婕	娼	婢	婵	胬	媪	媛	婷	婺	媾	嫫	媲
40~59	嫒	嫔	媸	嫠	嫣	嫱	嫖	嫦	嫘	嫜	嬉	嬗	嬖	嬲	嬷	孀	尕	尜	孚	孥
60~79	孳	孑	孓	孢	驵	驷	驸	驺	驿	驽	骀	骁	骅	骈	骊	骐	骒	骓	骖	骘
80~94	骛	骜	骝	骟	骠	骢	骣	骥	骧	纟	纡	纣	纥	纨	纩					

71区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		纭	纰	纾	绀	绁	绂	绉	绋	绌	绐	绔	绗	绛	绠	绡	绨	绫	绮	绯
20~39	绱	绲	绳	绶	绺	绻	绾	缁	缂	缃	缇	缈	缋	缌	缏	缑	缒	缗	缙	缜
40~59	缛	缟	缡	缢	缣	缤	缥	缦	缧	缪	缫	缬	缭	缯	缰	缱	缲	缳	缵	幺
60~79	畿	巛	甾	邕	玎	玑	玮	玢	玟	珏	珂	珑	玷	玳	珀	珉	珈	珥	珙	顼
80~94	琊	珩	珧	珞	玺	珲	琏	琪	瑛	琦	琥	琨	琰	琮	琬					

72区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		琛	琚	瑁	瑜	瑗	瑕	瑙	瑷	瑭	瑾	璜	璎	璀	璁	璇	璋	璞	璨	璩
20~39	璐	璧	瓒	璺	韪	韫	韬	杌	杓	杞	杈	杩	枥	枇	杪	杳	枘	枧	杵	枨
40~59	枞	枭	枋	杷	杼	柰	栉	柘	栊	柩	枰	栌	柙	枵	柚	枳	柝	栀	柃	枸
60~79	柢	栎	柁	柽	栲	栳	桠	桡	桎	桢	桄	桤	梃	栝	桕	桦	桁	桧	桀	栾
80~94	桊	桉	栩	梵	梏	桴	桷	梓	桫	棂	楮	棼	椟	椠	棹					

73区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		椤	棰	椋	椁	楗	棣	椐	楱	椹	楠	楂	楝	榄	楫	榀	榘	楸	椴	槌
20~39	榇	榈	槎	榉	楦	楣	楹	榛	榧	榻	榫	榭	槔	榱	槁	槊	槟	榕	槠	榍
40~59	槿	樯	槭	樗	樘	橥	槲	橄	樾	檠	橐	橛	樵	檎	橹	樽	樨	橘	橼	檑
60~79	檐	檩	檗	檫	猷	獒	殁	殂	殇	殄	殒	殓	殍	殚	殛	殡	殪	轫	轭	轱
80~94	轲	轳	轵	轶	轸	轷	轹	轺	轼	轾	辁	辂	辄	辇	辋					

74区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		辍	辎	辏	辘	辚	軎	戋	戗	戛	戟	戢	戡	戥	戤	戬	臧	瓯	瓴	瓿
20~39	甏	甑	甓	攴	旮	旯	旰	昊	昙	杲	昃	昕	昀	炅	曷	昝	昴	昱	昶	昵
40~59	耆	晟	晔	晁	晏	晖	晡	晗	晷	暄	暌	暧	暝	暾	曛	曜	曦	曩	贲	贳
60~79	贶	贻	贽	赀	赅	赆	赈	赉	赇	赍	赕	赙	觇	觊	觋	觌	觎	觏	觐	觑
80~94	牮	犟	牝	牦	牯	牾	牿	犄	犋	犍	犏	犒	挈	挲	掰					

75区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		搿	擘	耄	毪	毳	毽	毵	毹	氅	氇	氆	氍	氕	氘	氙	氚	氡	氩	氤
20~39	氪	氲	攵	敕	敫	牍	牒	牖	爰	虢	刖	肟	肜	肓	肼	朊	肽	肱	肫	肭
40~59	肴	肷	胧	胨	胩	胪	胛	胂	胄	胙	胍	胗	朐	胝	胫	胱	胴	胭	脍	脎
60~79	胲	胼	朕	脒	豚	脶	脞	脬	脘	脲	腈	腌	腓	腴	腙	腚	腱	腠	腩	腼
80~94	腽	腭	腧	塍	媵	膈	膂	膑	滕	膣	膪	臌	朦	臊	膻					

76区

01~19 臁膦欤欷欹歃歆歙飑飒飓飕飙飚殳彀毂觳斐 欠风殳文

20~39 齑斓於旆旄旃旌旎旒旖炀炜炖炝炻烀炷炫炱烨 方火

40~59 烊焐焓焖焯焱煳煜煨煅煲煊煸煺熘熳熵熨熠燠

60~79 燔燧燹爝爨灬焘煦熹戾戽扃扈扉礻祀祆祉祛祜 灬户礻

80~94 祓祚祢祗祠祯祧祺禅禊禚禧禳忑忐

77区

01~19 怼恝恚恧恁恙恣悫愆愍慝憩憝懋懑戆肀聿沓 聿(肀)水

20~39 泶淼矶矸砀砉砗砘砑斫砭砜砝砹砺砻砟砼砥砬 石

40~59 砣砩硎硭硖硗砦硐硇硌硪碛碓碚碇碜碡碣碲碹

60~79 碥磔磙磉磬磲礅磴礓礤礞礴龛黹黻黼盱眄眍盹 龙黹目

80~94 眇眈眚眢眙眭眦眵眸睐睑睇睃睚睨

78区

01~19 睢睥睿瞍睽瞀瞌瞑瞟瞠瞰瞵瞽町畀畎畋畈畛 田

20~39 畲畹疃罘罡罟詈罨罴罱罹羁罾盍盥蠲钅钆钇钋 皿钅(金)

40~59 钊钌钍钏钐钔钗钕钚钛钜钣钤钫钪钭钬钯钰钲

60~79 钴钶钷钸钹钺钼钽钿铄铈铉铊铋铌铍铎铐铑铒

80~94 铕铖铗铙铘铛铞铟铠铢铤铥铧铨铪

79区

01~19 铩铫铮铯铳铴铵铷铹铼铽铿锃锂锆锇锉锊锍

20~39 锎锏锒锓锔锕锖锘锛锝锞锟锢锪锫锩锬锱锲锴

40~59 锶锷锸锼锾锿镂锵镄镅镆镉镌镎镏镒镓镔镖镗

60~79 镘镙镛镞镟镝镡镢镤镥镦镧镨镩镪镫镬镯镱镲

80~94 镳锺矧矬雉秕秭秣秫稆嵇稃稂稞稔 矢禾

80区

01~19 稹稷穑黏馥穰皈皎皓皙皤瓞瓠甬鸠鸢鸨鸩鸪 白瓜用鸟

20~39 鸫鸬鸲鸱鸶鸸鸷鸹鸺鸾鹁鹂鹄鹆鹇鹈鹉鹋鹌鹎

40~59 鹑鹕鹗鹚鹛鹜鹞鹣鹦鹧鹨鹩鹪鹫鹬鹱鹭鹳疒疔 疒

60~79 疖疠疝疬疣疳疴疸痄疱疰痃痂痖痍痣痨痦痤痫

80~94 痧瘃痱痼痿瘐瘀瘅瘌瘗瘊瘥瘘瘕瘙

81区

01~19 瘛瘼瘢瘠癀瘭瘰瘿瘵癃瘾瘳癍癞癔癜癖癫癯

20~39 翊竦穸穹窀窆窈窕窦窠窬窨窭窳衤衩衲衽衿袂 立穴衤

40~59 袢裆袷袼裉裢裎裣裥裱褚裼裨裾裰褡褙褓褛褊

60~79 褴褫褶襁襦襻疋胥皲皴矜耒耔耖耜耠耢耥耦耧 疋皮矛耒

80~94 耩耨耱耋耵聃聆聍聒聩聱覃顸颀颃 老耳西页

82区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		颉	颌	颍	颏	颔	颚	颛	颞	颟	颡	颢	颥	颦	虍	虔	虬	虮	虿	虺
20~39	虼	虻	蚨	蚍	蚋	蚬	蚝	蚧	蚣	蚪	蚓	蚩	蚶	蛄	蚵	蛎	蚰	蚺	蚱	蚯
40~59	蛉	蛏	蚴	蛩	蛱	蛲	蛭	蛳	蛐	蜓	蛞	蛴	蛟	蛘	蛑	蜃	蜇	蛸	蜈	蜊
60~79	蜍	蜉	蜣	蜻	蜞	蜥	蜮	蜚	蜾	蝈	蜴	蜱	蜩	蜷	蜿	螂	蜢	蝽	蝾	蝻
80~94	蝠	蝰	蝌	蝮	螋	蝓	蝣	蝼	蝤	蝙	蝥	螓	螯	螨	蟒					

83区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		蟆	螈	螅	螭	螗	螃	螫	蟥	螬	螵	螳	蟋	蟓	螽	蟑	蟀	蟊	蟛	蟪
20~39	蟠	蟮	蠖	蠓	蟾	蠊	蠛	蠡	蠹	蠼	缶	罂	罄	罅	舐	竺	竽	笈	笃	笄
40~59	笕	笊	笫	笏	筇	笸	笪	笙	笮	笱	笠	笥	笤	笳	笾	笞	筘	筚	筅	筵
60~79	筌	筝	筠	筮	筻	筢	筲	筱	箐	箦	箧	箸	箬	箝	箨	箅	箪	箜	箢	箫
80~94	箴	篑	篁	篌	篝	篚	篥	篦	篪	簌	篾	篼	簏	簖	簋					

84区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		簟	簪	簦	簸	籁	籀	臾	舁	舂	舄	臬	衄	舡	舢	舣	舭	舯	舨	舫
20~39	舸	舻	舳	舴	舾	艄	艉	艋	艏	艚	艟	艨	衾	袅	袈	裘	裟	襞	羝	羟
40~59	羧	羯	羰	羲	籼	敉	粑	粝	粜	粞	粢	粲	粼	粽	糁	糇	糌	糍	糈	糅
60~79	糗	糨	艮	暨	羿	翎	翕	翥	翡	翦	翩	翮	翳	糸	絷	綦	綮	繇	纛	麸
80~94	麴	赳	趄	趔	趑	趱	赧	赭	豇	豉	酊	酐	酎	酏	酤					

85区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		酢	酡	酰	酩	酯	酽	酾	酲	酴	酹	醌	醅	醐	醍	醑	醢	醣	醪	醭
20~39	醮	醯	醵	醴	醺	豕	鹾	趸	跫	踅	蹙	蹩	趵	趿	趼	趺	跄	跖	跗	跚
40~59	跞	跎	跏	跛	跆	跬	跷	跸	跣	跹	跻	跤	踉	跽	踔	踝	踟	踬	踮	踣
60~79	踯	踺	蹀	踹	踵	踽	踱	蹉	蹁	蹂	蹑	蹒	蹊	蹰	蹶	蹼	蹯	蹴	躅	躏
80~94	躔	躐	躜	躞	豸	貂	貊	貅	貘	貔	斛	觖	觞	觚	觜					

86区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		觥	觫	觯	訾	謦	靓	雩	雳	雯	霆	霁	霈	霏	霎	霪	霭	霰	霾	龀
20~39	龃	龅	龆	龇	龈	龉	龊	龌	黾	鼋	鼍	隹	隼	隽	雎	雒	瞿	雠	銎	銮
40~59	鋈	錾	鍪	鏊	鎏	鐾	鑫	鱿	鲂	鲅	鲆	鲇	鲈	稣	鲋	鲎	鲐	鲑	鲒	鲔
60~79	鲕	鲚	鲛	鲞	鲟	鲠	鲡	鲢	鲣	鲥	鲦	鲧	鲨	鲩	鲫	鲭	鲮	鲰	鲱	鲲
80~94	鲳	鲴	鲵	鲶	鲷	鲺	鲻	鲼	鲽	鳄	鳅	鳆	鳇	鳊	鳋					

87区		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
01~19		鳌	鳍	鳎	鳏	鳐	鳓	鳔	鳕	鳗	鳘	鳙	鳜	鳝	鳟	鳢	靼	鞅	鞑	鞒
20~39	鞔	鞯	鞫	鞣	鞲	鞴	骱	骰	骷	鹘	骶	骺	骼	髁	髀	髅	髂	髋	髌	髑
40~59	魅	魃	魇	魉	魈	魍	魑	飨	餍	餮	饕	饔	髟	髡	髦	髯	髫	髻	髭	髹
60~79	鬈	鬏	鬓	鬟	鬣	麽	麾	縻	麂	麇	麈	麋	麒	鏖	麝	麟	黛	黜	黝	黠
80~94	黟	黢	黩	黧	黥	黪	黯	鼢	鼬	鼯	鼹	鼷	鼽	鼾	齄					

参 考 资 料

〔1〕The IBM Personal Computer AT Technical Reference Manual, IBM Corp., 1984.

〔2〕Disk Operating System Version 3.00, IBM Corp., 1984.

〔3〕NEC PC-9800シリーズ MS-DOS 2.0 マクロアセンブラマニュアル，日本電気株式会社，1983.

〔4〕Macro Assembler (Version 1.00), IBM Corp., 1981.

〔5〕杜毅仁等编，十六位微型计算机（上册），上海交通大学出版社，1984.

〔6〕D.C.Willen, J.I.Krantz, 8088 Assembler Language Programming: the IBM PC, Indianapolis Inc., 1983.

〔7〕iAPX 86/88 User's Manual, Intel Corp., 1981.

〔8〕R. Startz, 8087 Applications and Programming for the IBM PC and Other PCs, R.J. Brady Corp., 1983.

〔9〕FORTRAN Compiler (Version 2.00), IBM Corp., 1984.

〔10〕Technic Manual for IBM PC/XT, IBM Corp., 1983.

〔11〕B.W.Kernighan, D.M.Ritchie, The C Programming Language, Bell Laboratories, 1978.

〔12〕Optimizing=C86 User's Manual (Version 2.00), Computer Innovations Inc., 1983.

〔13〕L.H.Cock, M.Krieger, The C Primer, McGraw-Hill Book Company, 1982.

〔14〕Feuer, R.Alan, The C Puzzle Book, Prentice-Hall, 1982.

〔15〕R.Byers etc., dBASE Ⅲ User Manual, Ashton-Tate, 1984.

〔16〕CORVUS Systems: MS-DOS 2.0 System Manager Guide, Corvus Systems Inc., 1984.

〔17〕CORVUS Systems: MS-DOS 2.0 Network Station User Guide, Corvus Systems Inc., 1984.

〔18〕CORVUS Systems: System Generation Guide (IBM PC MS-DOS 2.0), Corvus Systems Inc., 1983.

〔19〕CORVUS Systems: OmniShare Installation Guide (IBM Fixed Disk), Corvus Systems Inc., 1984.

〔20〕张公忠等，OMNINET 小特辑，小型微型计算机系统，No.2，1984.

〔21〕何绍德等，如何在OMNINET网上实现工作站点间通讯，小型微型计算机系统，No.6，1985.

〔22〕LAN: Mail Monitor, Electronic Mail for Local Area Networks, Software Connections Inc., 1984.

〔23〕LAN: Datastore, Multi-User Database Management System, Software Connections Inc., 1983.

〔24〕Ethernet Administrator's Guide Book, 3Com Corp., 1984.

〔25〕Ethernet User's Guide Book, 3Com Corp., 1984.

〔26〕A.Finger, IBM PC/AT, BYTE, 10, 5, 1985.

〔27〕B.Staff, The IBM PC/AT, BYTE, 9, 10, 1984.

〔28〕P.Wells, The 80286 Microprocessor, BYTE, 9, 11, 1984.

〔29〕E.Sabine, The IBM XT/370 Personal Computer, Fall 1984 BYTE Guide to the IBM PC

〔30〕L.Augustin, The Mainframe Connection: IBM's 3270 PC, Fall 1984 BYTE Guide to the IBM PC.