李忠源　郑甫京　沈金发

IBM-PC　长城

微机基本输入输出系统

BIOS

清华大学出版社

IBM-PC 长城
微机基本输入输出系统 BIOS

李忠源 郑甫京 沈金发

清华大学出版社

内 容 简 介

BIOS 是长城系列微机及 IBM-PC 系列微机 DOS 操作系统的核心部分，它固化在 ROM 中。本书在剖析 BIOS 的基础上，详细介绍微机外设各通讯接口的编程方法、各外设输入/输出的实现方式、BIOS 各模块的逻辑功能、汉字的输入、显示及打印的实现等内容，同时列出了 BIOS 程序、汉字输入程序 GWINT16.COM 和打印机汉字驱动程序，并详细加以注释。这是一本学习微机系统结构、汇编程序、操作系统的参考书，也是微机课程的实用教材。适合于广大计算机应用开发的科技人员以及计算机专业的师生使用。

(京)新登字 158 号

IBM-PC 长城微机基本输入输出系统 BIOS

李忠源　郑甫京　沈金发

☆

清华大学出版社出版

(北京　清华园)

国防科工委印刷厂印刷

新华书店总店科技发行所发行

☆

开本：787×1092 1/16　印张：37　字数：878 千字

1992 年 5 月第 1 版　1992 年 5 月第 1 次印刷

印数：0001—8000

ISBN 7-302-00981-3/TP・361

定价：15.60 元

前　言

我国微型计算机的应用发展很快，应用领域也不断扩大，例如，在系统工程、计算机辅助设计、计算机辅助制造、现代化管理、人工智能和专家系统以及机器人系统等领域的广泛应用。这就需要提高广大微型计算机用户的技术和使用水平，掌握微型计算机内部的系统结构和工作原理。本书正是应这种需求而编写的。

本书在全面剖析 BIOS 的基础上，详细介绍了微型计算机长城 0520CH 和 IBM-PC 外设各通信接口的编程方式、各外设输入/输出的实现方式、BIOS 各模块的逻辑功能、汉字的输入、显示以及打印的实现等内容，同时列出了 BIOS 程序，汉字输入程序 GWINT16.COM 和打印机汉字驱动程序，并详细加以注释。由于长城系列微型计算机使用相同的操作系统，因此本书也适用于长城系列（长城 0520A、长城 286、286B、386）以及与 IBM-PC 相兼容的微型计算机。

本书是一本学习微型计算机的系统结构、汇编程序、操作系统的参考书，也是微型计算机课程的实用教材，适合于广大计算机应用开发的科技人员以及计算机专业的师生使用。

书中若有不妥之处，敬请广大读者批评、指正。

作　者

目　录

第一章　Intel 8088 指令系统及汇编语言 …… 1
1.1　Intel 8088 的结构 …… 1
1.2　寻址方式 …… 6
1.3　指令系统 …… 8
1.4　中断方式及中断优先权 …… 19
1.5　汇编语句 …… 22
1.6　长城 0520CH 的基本配置 …… 27
第二章　磁盘操作系统 DOS …… 29
2.1　DOS 的功能及其组成部分 …… 29
2.2　DOS 的主要命令 …… 32
2.3　DOS 使用的按键 …… 41
2.4　DOS 的树型目录 …… 46
2.5　DOS 的数据区 …… 49
2.6　DOS 引导程序 …… 58
第三章　BIOS 的结构及功能 …… 64
3.1　BIOS 与 DOS 的关系 …… 64
3.2　BIOS 中断向量及调用命令 …… 64
3.3　BIOS 数据区 …… 98
3.4　GWBIOS 外部模块命令 …… 104
第四章　软盘输入输出处理程序 …… 112
4.1　软盘上的数据记录方式 …… 112
4.2　5.25 英寸软盘适配器和驱动器 …… 113
4.3　软盘控制器的寄存器 …… 117
4.4　FDC 的命令状态寄存器 …… 120
4.5　软盘控制器的命令 …… 124
4.6　软盘处理程序的调用 …… 138
4.7　软盘参数表 …… 140
4.8　软盘处理程序 …… 142
4.9　软盘引导程序 …… 161
第五章　硬盘输入输出处理程序 …… 167
5.1　温盘驱动器概述 …… 167
5.2　温盘控制器及接口寄存器 …… 172
5.3　控制器命令 …… 177

5.4 硬盘处理程序的调用 …… 189
5.5 硬盘处理程序 …… 197
第六章 彩色显示器中断处理程序 …… 216
6.1 字母/数字显示方式 …… 216
6.2 彩色显示适配器 …… 220
6.3 6845 显示控制器各寄存器的功能 …… 227
6.4 图形显示的实现 …… 230
6.5 汉字显示的实现 …… 238
6.6 显示器内外 I/O 处理程序 …… 243
6.7 0520CH 内部显示器 I/O 处理模块 …… 253
6.8 0520CH 外部显示器 I/O 处理模块 …… 285
第七章 键盘输入处理程序 …… 373
7.1 键盘 …… 373
7.2 键盘中断程序 …… 375
7.3 键盘内外 I/O 处理程序 …… 388
7.4 内部键盘 I/O 处理模块 …… 390
7.5 汉字输入、输入码表及转换 …… 391
7.6 0520CH 键盘外部 I/O 处理模块 …… 396
第八章 打印机中断处理程序 …… 460
8.1 并行打印机适配器 …… 460
8.2 打印机控制代码 …… 463
8.3 打印机内外 I/O 处理程序 …… 465
8.4 内部打印机 I/O 处理模块 …… 467
8.5 汉字打印方式 …… 470
8.6 0520CH 打印机外部 I/O 处理模块 …… 473
第九章 开机测试程序 …… 514
9.1 主机板上的接口器件 …… 514
9.2 测试程序 …… 532
第十章 异步通信处理程序 …… 564
10.1 异步通信概述 …… 564
10.2 RS-232-C 通信适配器 …… 567
10.3 异步通信程序的调用 …… 577
10.4 通信处理程序 …… 579

第一章　Intel 8088 指令系统及汇编语言

1.1　Intel 8088 的结构

Intel 8088 微机处理器是 0520CH 微型计算机的核心，在这片大规模集成电路芯片中，包括了控制整个微型计算机工作的控制电路、寄存器组和算术逻辑部件，这就是我们通常所说的 CPU。

Intel 8088 是在 8080/8085 的基础上发展起来的准 16 位微处理器。它的内部组织结构是 16 位的，但对外数据总线是 8 位。它既可以处理 8 位(字节)数据，也可以处理 16 位(字)数据。有 20 条地址线，使直接寻址空间达 1MB(兆字节)。

1.1.1　Intel 8088 的功能结构

8088 内部由两个独立的工作单元组成：执行单元 EU(Execution Unit)和总线接口单元 BIU(Bus Interface Unit)。如图 1-1 所示。

执行单元 EU 负责执行指令，而总线接口单元 BIU 负责从存储器或外部设备中读取操作码、操作数，并将结果写入指定的地址中，这两个单元处于并行工作状态，可以同时进行存取操作和指令执行的操作。

(1) 执行单元 EU

执行单元包括一个 16 位的算术/逻辑运算器 ALU、一个 16 位的反应工作状态和控制标志的状态寄存器 FLAG 及一组通用寄存器、数据暂存寄存器和 EU 控制电路。执行单元 EU 完成指令所规定的操作过程是：从 BIU 中的指令队列寄存器中取得指令，由 EU 控制电路对其指令进行分析译码，确定本次操作的性质及操作对象。当要求从存储器或外部设备中读取操作数时，由 EU 中计算出操作数的地址送给 BIU，BIU 根据请求将读取操作数，经内部的 ALU 数据总线送 ALU 进行指定的运算。最后运算结果保存在通用寄存器或送 BIU 存储到存储器或外部设备中，将本次操作的状态标志存放在状态寄存器 FLAG 中。

(2) 总线接口单元 BIU

总线接口单元 BIU 根据 EU 的请求，完成 CPU 与存储器和外部设备之间的数据传送。

总线接口单元 BIU 中包括一组段寄存器(CS、DS、SS、ES)、一个指令指示器(IP)、4 个字节的指令队列、地址产生器和总线控制器。BIU 负责在执行指令期间提供系统总线控制信号，将段寄存器中的内容与段内偏移量送到地址加法器之中，形成 20 位存储器物理地址。当 EU 在执行指令期间要求和外界进行数据传送时，BIU 根据 EU 的要求访问存储器或外部设备。

在 EU 进行内部操作期间，BIU 从存储器中取出指令，送入 4 字节的指令队列，供

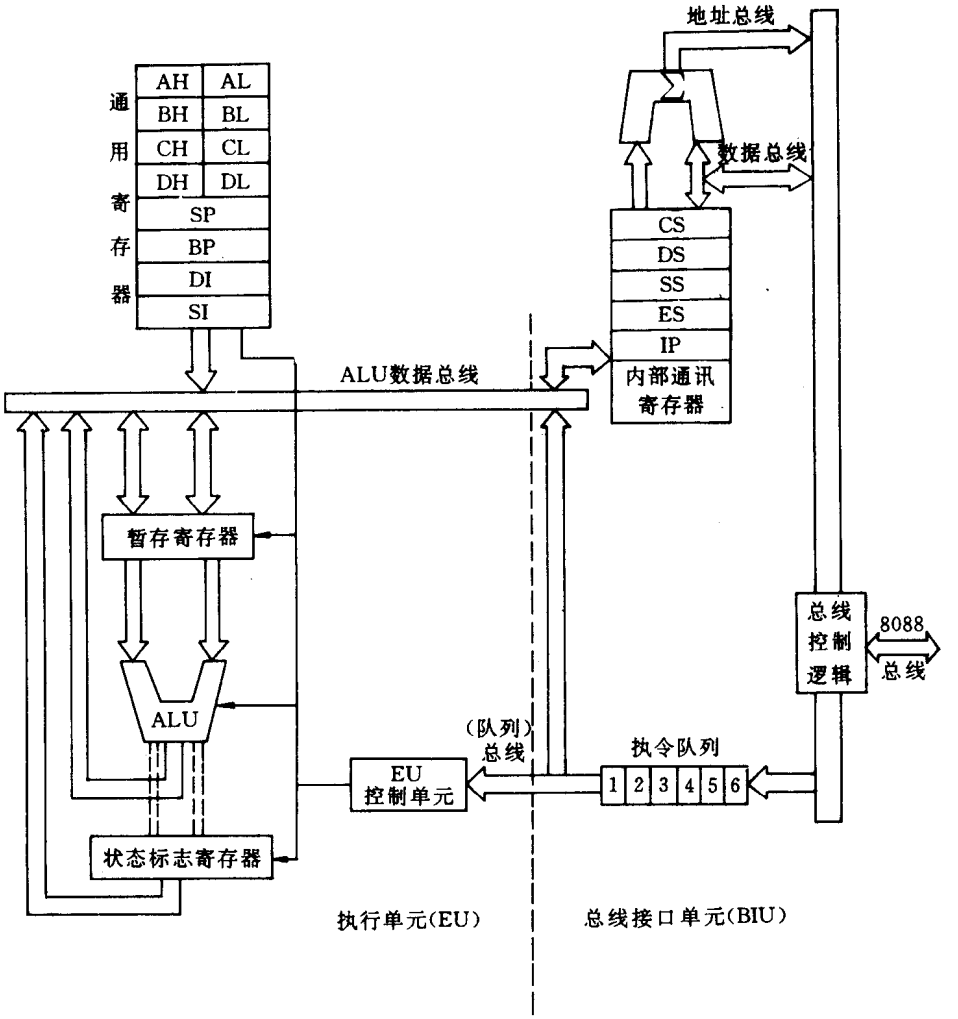

图 1-1　8088 结构框图

EU 执行。EU 从指令队列按先进先出的顺序取指令。在 EU 开始执行指令后，只要指令队列有空闲字节，BIU 就自动地从存储器取入指令操作码填满指令队列，以保证 EU 能够连续地执行指令。由此可见，这种指令预取技术把取指令和执行指令的操作重叠进行，取消了等待取指令的时间，可加快计算机的运行速度。只有在执行转移指令时，指令队列中的原有内容才会作废，EU 向 BIU 发出控制信号，要求 BIU 从新的地址中重新开始取指令。BIU 新取得的第一个指令字节将直接送到 EU 中去执行。然后，BIU 将随后取得的指令字节重新填入指令队列，冲掉原指令队列中所存放的指令字节。

在 EU 执行指令过程中，若需要读写存储器或外部设备数据，则向 BIU 发出信号，BIU 根据 EU 的请求，形成 20 位的访问地址和控制信号，执行存储器或外部设备的读写操作，按 EU 要求传递数据。

1.1.2 程序可访问的寄存器

Intel 8088 内部有 14 个可供程序直接访问的寄存器，它们都是 16 位的寄存器，包括 4 个通用寄存器、2 个指示器、2 个变址寄存器、4 个段寄存器、一个指令指示器和一个状态寄存器。如图 1-2 所示。

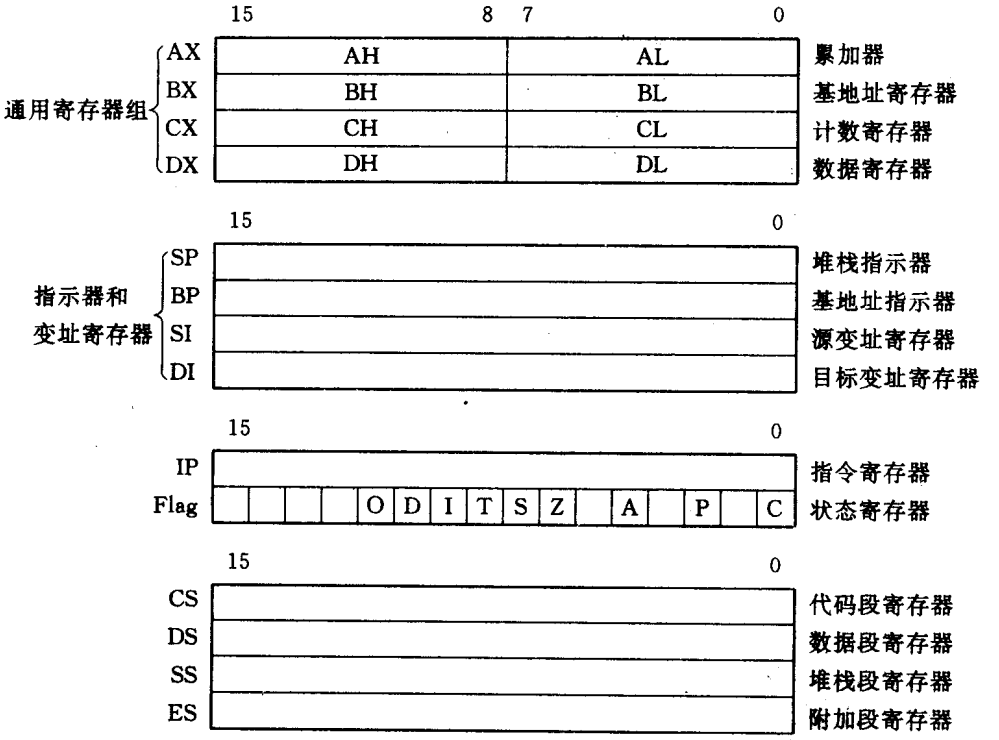

图 1-2 Intel 8088 的内部寄存器

(1) 数据寄存器

数据寄存器一般用来存放数据，它包括四个 16 位的寄存器：AX、BX、CX、和 DX。这些寄存器在进行字节操作时，又可以分别对它们的高 8 位和低 8 位进行寻址，成为 8 个 8 位的寄存器，它们的低 8 位称为 AL、BL、CL 和 DL，高 8 位为 AH、BH、CH 和 DH。

数据寄存器主要用来存放操作数和中间结果，以减少访问存储器的次数。在实际使用时，每个寄存器又各自有某种专门的用途。所以，对它们分别又有不同的称呼。AX 寄存器又被称为累加器、BX 称为基址寄存器、CX 称为计数寄存器、而 DX 则被称为数据寄存器。

(2) 指示器和变址寄存器

8088 的 EU 单元中还有四个 16 位的寄存器：指示器 SP 和 BP、变址寄存器 SI 和 DI，根据使用的不同，分别称为堆栈指示器 SP、基址指示器 BP、源变址寄存器 SI 和目标变址寄存器 DI。这些寄存器再加上 BX，用来形成操作数地址，也可以作为数据寄存器使用。

堆栈指示器 SP 用来存取当前堆栈段栈顶的数据，基址指示器 BP 用作间接寻址。SI 和 DI 寄存器在数据串操作指令中，用于指示源地址和目标地址。

(3) 段寄存器和物理地址

8088 能够寻址 1M 字节的地址空间，而指令中给出的地址仅有 16 位，这样达不到直接寻址 1M 字节的目的。为此，8088 提供了 4 个 16 位的段寄存器，用段寄存器将 1M 字节的地址空间分成若干个逻辑段，每个逻辑段的长度为 64KB(千字节)。

段寄存器是用来存放段的起始地址的，8088 中的 4 个段寄存器是代码段 CS、数据段 DS、堆栈段 SS 和附加段 ES。代码段寄存器 CS 用来给出当前的代码段，8088 执行的指令是从代码段 CS 中取得的。堆栈段寄存器 SS 用来给出程序当前所使用的堆栈段，堆栈操作的数据存取地址在这个段中。数据段寄存器 DS 用来给出程序当前使用的数据段。一般情况下，程序存取的数据存放在这个段中。附加段寄存器给出了程序当前使用的附加段，附加段一般用来存放处理以后得到的数据。

存储器分段时，对于段起始地址的要求只有一个，即必须能被 16 整除。各段之间可以完全重叠、部分重叠、互相邻接或互相分离。如图 1-3 所示。

8088 与存储器之间的数据交换使用物理地址，物理地址为 20 位。段地址和指令中给出的段内偏移量都是 16 位的。8088 的总线接口单元 BIU 用下述方法把这两个 16 位的数形成 20 位的物理地址：首先将段寄存器内容左移 4 位，右边补上 4 个“0”，形成 20 位的段起始地址，再加上 16 位的段内偏移量，从而形成 20 位的存储器物理地址。如图 1-4 所示。

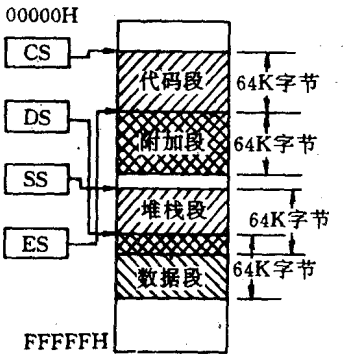

图 1-3 存储器的分段举例

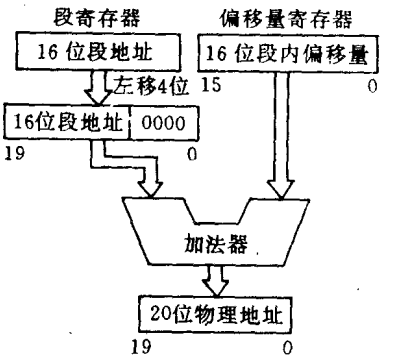

图 1-4 物理地址的形成

例：段地址是 114A，段内偏移量是 2523，则对应的物理地址是 114A0 + 2523 = 139C3。

(4) 指令寄存器 IP

指令寄存器 IP 是 16 位的，它指向代码段内当前要取出指令操作码的存储单元。IP 中的内容是段内偏移量，它和代码段寄存器 CS 形成下一条将要执行的指令字节的物理地址，每取一个指令操作码字节后，IP 自动加 1。程序不能对指令寄存器 IP 进行直接存取。

(5) 状态寄存器

8088 的 EU 单元中有一个 16 位的状态寄存器，设置 9 位标志，其中 6 个是状态标志，3 个是控制标志。状态标志位反映了 8088 执行算术或逻辑运算后的结果，有些指令的执行要受到这些标志的影响。控制状态位用来控制 8088 的工作条件。

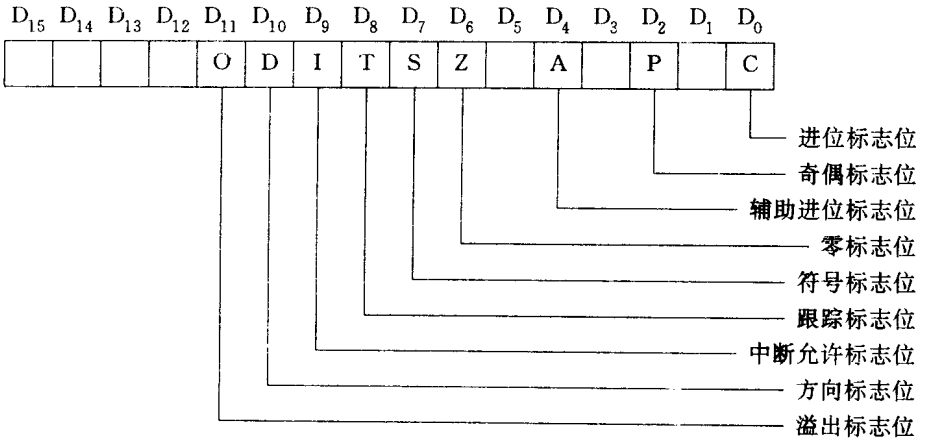

图1-5 状态寄存器

6 个状态标志位分别是：

进位标志位 C：当 C=1 时，表示指令执行后的结果在最高位上产生一个进位或借位；C=0 则表示无进位或借位产生。

辅助进位标志位 A：当 A=1 时，表示低 4 位向高 4 位上产生一个进位或借位。

奇偶标志位 P：当 P=1 时，表示指令操作的结果中 1 的个数为偶数；当 P=0 时表示为奇数个 1。

零标志位 Z：当 Z=1 时，表示指令操作的结果为零；当 Z=0 则表示指令操作的结果不为零。

符号标志位 S：当 S=1 时，表示运算结果为负数；当 S=0 则表示运算结果为正数。由于 8088 中带符号数是用补码表示的，即最高位为 0，表示正数；最高位为 1，表示负数。因此符号标志位 S 的值与运算结果的最高位相同。

溢出标志位 O：当 O=1 时，表示带符号的数做算术运算时结果超出了所能表达的范围，即产生溢出。当 O=1 时，则表示没产生溢出。

3 个控制状态位分别是：

中断允许标志位 I：这一位标志控制 8088 是否处理外部的可屏蔽中断请求。当 I=1 时，8088允许接受外部的可屏蔽中断请求，在当前指令执行结束后，便进入中断响应。当 I=0 时，8088 不允许接受外部的可屏蔽中断请求。

方向标志位 D：这一位用于在字符串操作中规定数据处理的方向。当 D=1 时，字符串操作指令每处理一个字节后地址自动减 1，即字符串处理从高地址到低地址的方向进行。当 D=0 时，字符串操作指令每处理一个字节后地址自动加 1，即字符串处理从低地址到高地址方向进行。

跟踪标志位 T：又称单步操作标志位，它控制 8088 的工作状态。当 T=1 时，为单步操作，即 8088 每执行完一条指令后就自动地进入内部中断，以便检查指令的执行情况。当 I=0 时，则 8088 处于正常的连续执行指令状态。

1.2 寻址方式

8088 访问操作数可采用多种灵活的寻址方式，从而使指令可方便地在整个 1 兆字节的存储器空间寻址。

1.2.1 隐含寻址

这种寻址方式在指令中不指明操作数，操作数隐含规定在某个寄存器中。

如指令： DAA

其作用是对寄存器 AL 进行十进制调整，结果仍存放在 AL 中。

1.2.2 立即数寻址

这种寻址方式所提供的操作数直接存放在指令中，紧跟在操作码后面，与操作码一起存放在代码段中。

如指令： MOV AX，100

其作用是将立即数 100 送到寄存器 AX 中。立即数是一个常数，它可为 8 位长，也可为 16 位长。

1.2.3 寄存器直接寻址

该寻址方式所提供的操作数存放在 8088 的内部寄存器中。

如指令： MOV DS，AX

其作用是将寄存器 AX 中的内容存放在段寄存器 DS 中。

1.2.4 存储器直接寻址

这是最简单的一种存储器寻址方式。它在指令中给出的常数并非操作数，而是操作数的有效地址。

如指令： MOV AX，[2000]

其作用是将当前数据段偏移量为 2000 的地址中的内容存放在寄存器 AX 中。实际上指令中一般不出现直接地址，而以简单变量表示，因此上述指令可改写为：

```
MOV AX，VALUE
```

这里 VALUE 可以用汇编指令定义。

1.2.5 寄存器间接寻址

在这种寻址方式中，操作数存放在存储器中，而操作数的有效地址包含在寄存器 SI、DI、BP、BX 中的一个当中。

如指令： ADD AX，[BX]

其作用是将由 BX 的内容作为有效地址取出的操作数与寄存器 AX 相加，结果存放在 AX 寄存器中。

使用这种寻址方式的好处是，通过适当地改变寄存器的内容，用同一条指令可以在很多不同的存储器地址中存取数据。

1.2.6 变址寻址

可以作为寄存器间接寻址的变址寄存器 SI 和 DI，也可以作变址寻址。变址寻址是指以变址寄存器 SI 或 DI 的内容，加上指令中给出的偏移量得到操作数的有效地址。

如指令： MOV AX ARRAY [SI]

假设 ARRAY 是一组数据，由它给出这组数据的偏移量，用变址寄存器 SI 中的值求这组数据中的任意数据。显然 SI 为 0 表示的是这组数据中的第一个数据。这样，上述指令的作用是根据 SI 给定的值，将数据组 ARRAY 中相应位置的数据存放到 AX 寄存器中，该数据的有效地址为 ARRAY+SI。

1.2.7 基址寻址

这种寻址方式与变址寻址方式十分相似，所不同的是用基址寄存器 BX、BP 来代替变址寄存器 SI、DI。也就是说，操作数的有效地址为指令中指出的基址寄存器加上偏移量。

如指令： MOV AX，[BX+6]

其有效地址为基址寄存器加上偏移量 6。需要指出的是：若用 BX 作为基址寻址时，规定被寻址的操作数位于当前数据段，而用 BP 作为基址寻址时，规定被寻址的操作数位于当前堆栈段。

1.2.8 基址变址寻址

这种寻址方式是上述二种寻址方式的组合。操作数的有效地址是由基址寄存器、变址寄存器和偏移量三者相加产生的。

如指令： MOV AX，6[BX] [SI]

基址变址寻址是最灵活的一种寻址方式。采用这种寻址方式计算有效地址，除了偏移量不能改变之外，基址和变址寄存器中的内容都是可以改变的。

1.2.9 数据串寻址

这种寻址方式只能用于数据串指令。在数据串指令中由 SI 寄存器给出源操作数的有效地址，由 DI 寄存器给出目标操作数的有效地址。

如指令： MOVSW

其作用是将当前数据段 DS 中由 SI 寄存器指出的操作数传送到当前附加段 ES 中由 DI 寄存器指出的地址，并且自动修改 SI 和 DI 寄存器以便指向下一个操作数地址。

1.2.10 I/O 端口寻址

对 I/O 端口的寻址方式取决于 I/O 端口是同存储器统一编址还是独立编址。若与存储器统一编址，则上述任何一种存储器的寻址方式都能用于寻址。若 I/O 端口独立编址，则有两种寻址方式，直接端口寻址和间接端口寻址。

在直接端口寻址方式中，端口号(即端口地址)是由指令直接提供的一个 8 位立即数，这种寻址方式只能用来寻址端口号为 0～255 的 I/O 端口。

如指令： IN AL，43

间接端口寻址方式类似于寄存器间接寻址方式，被寻址的端口地址由 DX 寄存器给出。

如指令： IN AL，DX

1.3 指令系统

8088 的指令系统按功能可以划分成六种指令类型，即数据传送、算术运算、逻辑运算、数据串操作、程序转移和处理机控制。本节将从这六个方面讨论 8088 的指令系统。

1.3.1 数据传送指令

14 种数据传送指令又可以分为四类：通用数据传送指令、累加器专用传送指令、目标地址传送指令和标志传送指令。

(一) 通用数据传送指令

这类指令有四条：MOV、PHSH、POP 和 XCHG。只有这类指令(XCHG 除外)才唯一允许段寄存器作为指令中的操作数使用。

(1) MOV 指令

格式： MOV 目的，源

它是把一个字节或字的源操作数传送到目的。源操作数可以是累加器、寄存器、存储器以及立即数，而目的操作数可以是累加器、寄存器和存储器。具体地说，一条指令可实现：

① 8088 内部寄存器之间数据的传送(除代码段寄存器 CS 和指令指示器 IP 以外)。

② 立即数传送到 8088 内部的通用寄存器(即字节时为 AL、BL、CL、DL、AH、BH、CH、DH，字时为 AX、BX、CX、DX、BP、SP、SI、DI)。

③ 8088 内部寄存器(除 CS 和 IP 之外)与存储器(所有寻址方式)之间的数据传送。

④ 立即数传送到存储器单元。

但是，它不能实现存储器单元之间的传送。如果需要实现这类传送，可以借助于寄存器作为过渡，或者直接使用数据串指令。

(2) XCHG 指令

格式： XCHG 目的，源

这是一条交换指令，用于将一个字节或一个字的源操作数与目的操作数相互交换。交

换能在通用寄存器与累加器之间、通用寄存器之间、通用寄存器与存储器之间进行。但段寄存器不能作为操作数。

(3) PUSH 和 POP 指令

PUSH 指令格式： PUSH 源

POP 指令格式： POP 目的

PUSH 指令将源操作数压入由 SP 所指向的栈顶。POP 指令将当前栈顶内容弹出到目的操作数中。

PUSH 和 POP 指令使用的操作数必须是字。此外，不允许代码段寄存器 CS 作为指令的目的操作数。

(二) 累加器专用传送指令

共有三条累加器专用传送指令，它们是 IN、OUT、XLAT。

(1) IN 和 OUT 指令

IN 指令格式： IN 累加器，端口地址

IN 累加器，DX

OUT 指令格式： OUT 端口地址，累加器

OUT DX，累加器

IN 指令将一个字节或一个字由指定的输入端口，传送到 AL(一个字节)或 AX(一个字节)中。OUT 指令将 AL 中的一个字节或 AX 中的一个字，传送到指定的输出端口。

(2) XLAT 指令

格式： XLAT 转换表名

这条指令实现表的查字节转换。BX 指向一张 256 个字节的表的起点，AL 中是表的索引值。执行这条指令后，AL 中即为查表所得到的内容。XLAT 经常用于把一种代码转换成另一种代码。

(三) 目标地址传送指令

有三条目标地址传送指令：LEA、LDS 和 LES。

(1) LEA 指令

格式： LEA 目的 ，源

把源操作数的偏移量传送到目的操作数。源操作数必须是一个存储器操作数，目的操作数必须是一个 16 位的通用寄存器、基址或变址寄存器。

(2) LDS 指令

格式： LDS 目的 ，源

此指令完成一个地址指针的传送。地址指针包括段地址和偏移量。将段地址传送到数据段寄存器 DS，而偏移量传送到指定的 16 位通用寄存器、基址或变址寄存器。

(3) LES 指令

格式： LES 目的 ，源

此指令除了将段地址传送到附加段寄存器 ES 之外，其余与 LDS 指令完全相同。

(四) 标志传送指令

有四条标志传送指令：LAHF、SAHF、PUSHF 和 POPF。

(1) LAHF 指令

格式： LAHF

把状态寄存器中的符号标志S、零标志Z、辅助进位标志A、奇偶标志P和进位标志C传送到AH的指定位，即相应地传送7,6,4,2和0位，而5,3,1位的内容没有定义。如图1-6所示。

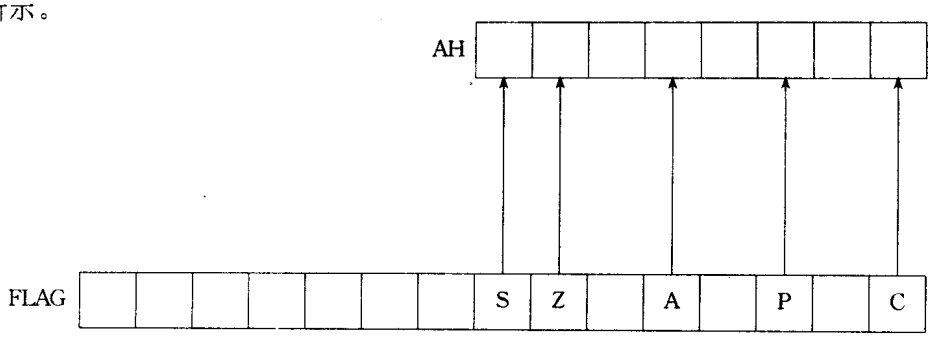

图1-6 LAHF指令的示意图

(2) SAHF 指令

格式： SAHF

这条指令与上一条的操作刚好相反，它是把寄存器AH中的0,2,4,6,7位传送到状态寄存位低8位的相应标志位上。

(3) PUSHF 指令

格式： PUSHF

把整个状态寄存器压入栈顶。

(4) POPF 指令

格式： POPF

这条指令把当前栈顶内容弹出到状态寄存器中。

1.3.2 算术运算指令

8088提供了加、减、乘、除四种基本操作。这些操作都可用于字节和字的运算，也可用于带符号数与无符号数的运算。若是符号数，则用补码表示。

(一) 加法

有五条加法操作指令：ADD、ADC、INC、AAA和DAA。

(1) ADD 指令

格式： ADD 目的，源

这个指令完成两个操作数相加，结果送到目的操作数。目的操作数可以是累加器、通用寄存器以及存储器(所有寻址方式)。

(2) ADC 指令

格式： ADC 目的，源

这条指令与上一条类似，只是在两个操作数相加时，把进位标志C的现行值加上去，结果送到目的操作数。

(3) INC 指令

格式： INC 目的

这条指令对指定的操作数加 1，然后结果再存入此操作数中。

(4) DAA 指令

格式： DAA

对 AL 中二个组合的十进制操作数相加结果进行调整，输出结果为一个组合的十进制和数。

(5) AAA 指令

格式： AAA

对 AL 中的非组合十进制数加法结果进行十进制数的调整。

(二) 减法指令

有七条减法指令：SUB、SBB、DEC、NEG、CMP、AAS 和 DAS。

(1) SUB 指令

格式： SUB 目的，源

这个指令完成两个操作数相减，结果送到目的操作数。目的操作数可以是累加器、通用寄存器以及存储器(所有寻址方式)。

(2) SBB 指令

格式： SBB 目标，源

这条指令与上一条类似，只是在两个操作数相减时，还要减去借位标志 C 的现行值，结果送到目的操作数。

(3) DEC 指令

格式： DEC 目的

这条指令对指定的操作数减 1，然后结果再存入此操作数中。

(4) NEG 指令

格式： NEG 目的

这条指令是对操作数取补，即用零值减去操作数，结果存回目的操作数。

(5) CMP 指令

格式： CMP 目的，源

将目的操作数减去源操作数，其结果只影响相应的状态标志位，但并不返回。

(6) DAS 指令

格式： DAS

此指令与 DAA 指令类似，对 AL 中二个组合的十进制操作数相减结果进行调整，输出结果为一个组合的十进制差数。

(7) AAS 指令

格式： AAS

此指令与 AAA 指令类似，对 AL 中的非组合十进制数减法结果进行十进制数的调整。

(三) 乘法指令