IBM-PC微機匯編語言及接口技術

劉鏡年 編著

武漢測繪科技大學出版社

IBM-PC 微机汇编语言及接口技术

刘镜年　编著

武汉测绘科技大学出版社

(鄂)新登字 14 号

IBM-PC 微机汇编语言及接口技术

编　　著　刘镜年
责任编辑　倚　农
装

武汉测绘科技大学出版社出版发行
武汉测绘科技大学出版社激光照排
武汉测绘科技大学印刷厂印刷
开本:787×1092 1/16　印张:14.75　字数:378 千字
1992 年 7 月第一版　1993 年 4 月第二次印刷
印数:2001—4000 册

ISBN 7-81030 170 5/T・32　定价:11.50 元

前　言

随着科学技术的进步，计算机技术已成为当代最重要的科学技术之一。当今掌握计算机的基本工作原理、操作和软件开发应用，不仅是计算机专业工作者的任务，而且已成为各专业技术人员的迫切要求。高校各非计算机专业都已陆续开设微机原理及应用课程，但大部分选用8位微机为背景的教材。当今微机发展很快，16位微机已相当普及、书刊资料也不少，但适用于非计算机专业学生的16位微机教材却很难找到。为此，编者在几年来为本校计算机专业讲授《汇编语言》和非计算机专业讲授《IBM-PC原理及应用》课程讲义的基础上，参阅国内外有关资料并根据授课经验，几经修改综合整理而成本教材。

本书以Intel 8088微机为背景，从硬件和软件结合上对IBM-PC微机的组成、汇编语言程序设计和接口技术进行了详细讨论。本书涉及面广，根据情况可以选讲其中一部分。例如，对于学习汇编语言可选择前四章和第八章。只要读者对8088微处理器和存贮器分段结构有一定的了解，就可以学习汇编语言程序设计，并且此微机的操作系统为用户提供了许多功能调用。这种调用比较简单，它同高级语言类似，可以不管硬件接口的结构，通过调用就可以访问许多外设或者进行文件管理，为用户编程带来方便。如作为微机原理教材，则可继续学习第五章至第七章，这三章主要叙述IBM-PC个人计算机系统结构、接口芯片及其编程，并将它们应用在该机常用的几种接口电路中，着重讨论各种接口原理框图，然后通过接口程序把中央处理器和外部设备连接起来。这样避免了烦杂的电子线路，既节省了篇幅，又使读者思路清晰。为适应非计算机专业人员的需要，本书采用深入浅出、循序渐进的方式进行编写，选材上除注意系统性、先进性和完整性外，还注意到软、硬件结合，理论与实践结合，加强编程与上机实习的原则，以培养学生编程和调试程序的能力。本书可作为高校非计算机专业微机原理教材或微机培养班教材，也适用于各专业工程技术人员自学。

本书由华中理工大学傅望月教授担任主审，对原稿提出了许多宝贵意见和建议；华中理工大学刘乐善副教授、王元参副教授、武汉测绘科技大学夏正钧副教授对全书编写给予了极大支持，并仔细审阅了原稿。在此谨向他们表示衷心感谢！本书出版得到了武汉测绘科技大学计算机科学与工程系主任李锦祥教授、副主任李鸣山副教授的鼓励与支持，还得到了武汉测绘科技大学出版社的大力支持和赞助，编者在此谨致诚挚的谢意！书中全部插图由胡启平同志用微机绘制，对他的辛勤工作表示感谢！

限于水平，加之时间仓促，书中疏漏和不妥之处，敬请读者批评指正。

由于篇幅限制，与教材配套的微型计算机基础部分及实验指导书将由另书完成。

编者

1992年2月

目　录

第一章　8086/8088 的体系结构

本章先介绍 8086/8088 微机处理器结构特点和工作原理、存贮器组织和标志寄存器等。本章并不准备详细描述它们的电子线路和有关的物理过程，而主要从软件和应用开发的角度出发，讨论 8086/8088 的可编程寄存器、存贮器的段结构以及它们呈现在程序设计方面的特性和逻辑功能。其次概括介绍 8086/8088 的指令集：包括指令类型、寻址方式和指令特点，作为学习汇编语言的基础知识。

1.1　引言

虽然高级语言易学易用，但也有不足之处，即直接用高级语言编写的程序，计算机是不能直接执行的，需要由编译或解释程序将它翻译成对应的机器语言程序，机器才能接受。通过编译或解释程序生成的机器语言程序往往比较冗长，例如，BABIC 语言的 PRINT 语句就是用 100 条 8088 机器指令完成的。执行 BASIC 程序时，轮到每一条语句必须被译码，然后，使用相应的机器指令序列实现该语句的功能。因此，占用存贮空间较大，执行速度较慢。高级语言程序员无法直接利用机器硬件系统的许多特点，例如寄存器、标志位以及一些特殊指令等，影响许多程序设计技巧的发挥。

汇编语言程序是直接利用机器提供的指令系统编写程序，它与机器语言程序一一对应，中央处理器可以直接执行，省去了译码辅助操作，因而速度快，占用内存空间少。

用汇编语言编程可以充分发挥计算机硬件的功能并提高编程质量。为此学习汇编语言程序设计首先应该熟悉机器的指令系统。而指令系统又是与具体机器的内部结构密切相关的，因此要熟悉机器的内部结构，特别是中央处理器(CPU)和存贮器的结构，还应熟悉机器中与编程有关的其它部分的结构，例如中断系统、视频显示、键盘接收、通迅、定时功能等等。另外我们还必须知道系统为我们提供的软件环境，如存放在 ROM 中的监控程序、存在磁盘上的操作系统、汇编程序、调试程序等。我们可以充分利用它们的支持，直接调用其中的子程序等等。

是否采用汇编语言编写程序，要看具体应用场合，在软件的开发时间及软件的质量方面进行权衡和抉择。一般来说某些对执行时间和存贮器容量要求较高的程序采用汇编语言编写，如实时控制系统、智能化仪器仪表及高性能软件等方面。

1.2　8086/8088 CPU 结构

本章我们将首先介绍 8086/8088 CPU 的内部结构和存贮结构，作为学习汇编语言程序设计的基础知识。

Intel 8086/8088 是两种第三代微处理器。它们有 20 条地址线，直接寻址能力达 1MB。8088 具有 8 位数据通道可以与存贮器或输入输出交换信息。而 8086 则为 16 位数据通道。其它方面，两个处理器都是相同的，为其中一个 CPU 写的软件可以不需修改地在另一个 CUP 上执行。IBM-PC 系列机及兼容机上广泛采用了 8088。

8086/8088 CPU 就功能而言分为两大部分：总线接口单元 BIU(Bus Interface Unit)和执行单元 EU(Execution Unit)。如图 1.2.1 所示。

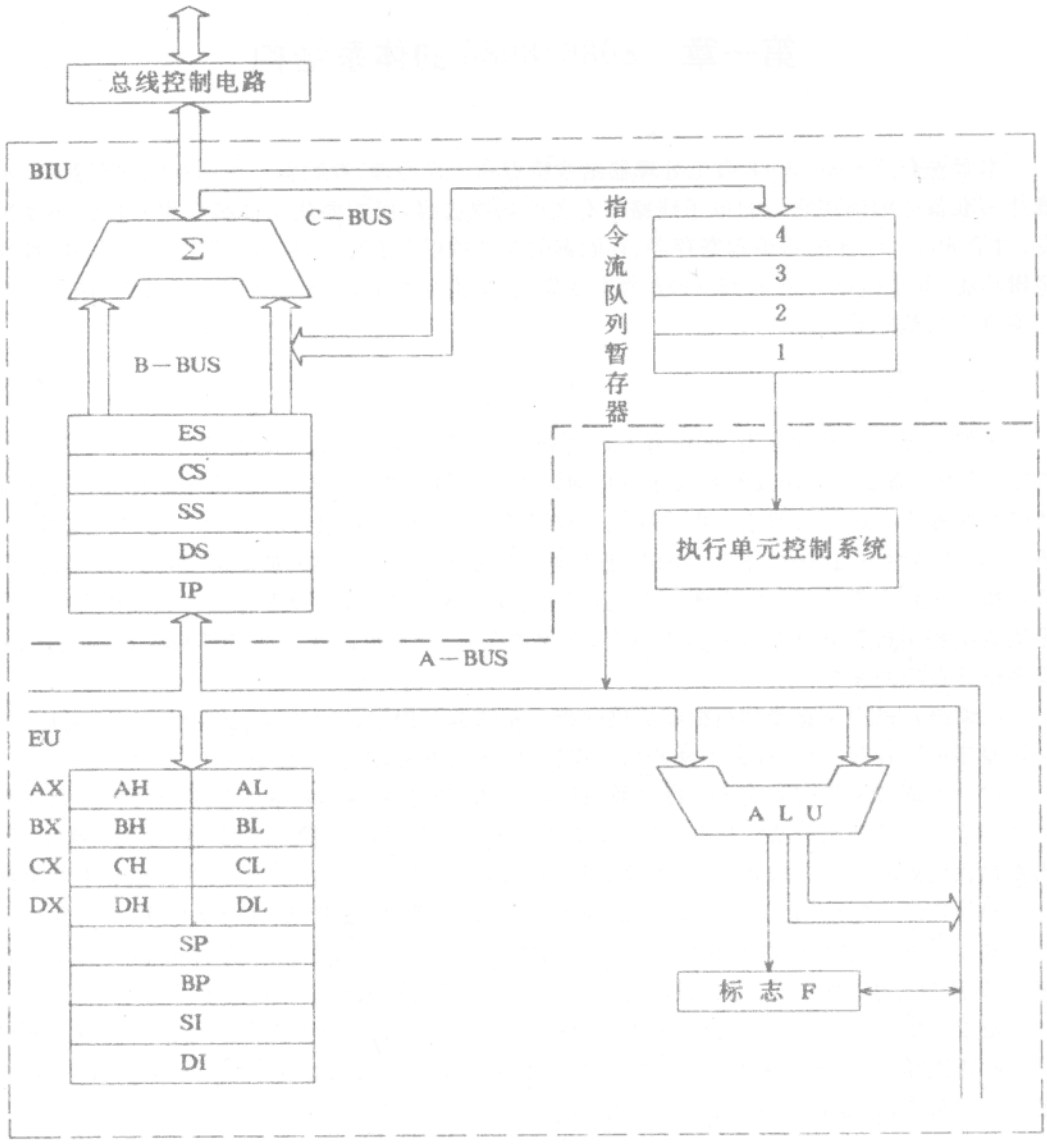

图 1.2.1 8088 微处理器功能结构

BIU 负责 8086/8088 CPU 与存贮器和外部设备之间的信息传送。具体地说，即 BIU 负责从内存指定部分取出指令送至指令流队列机构中排队。在执行指令时，所需的操作数，也由 BIU 从内存的指定区域取出，传送给 EU 部分去执行。

EU 负责指令的执行，并进行算术逻辑运算等。由于取指令部分和执行部分是分开的，所以在一条指令执行过程中，就可以取出一条或多条指令，放在指令流队列中排队。当一条指令执行完以后就可立即执行下一条指令，减少了 CPU 为取指令而等待的时间，提高了 CPU 的利用率，加快了系统运行速度。另一方面又降低了与之配合的存贮器的存取速度的要求。

1.3 8086/8088 CUP 的寄存器组

8086/8088 的寄存器组如图 1.3.1 所示。它能处理 16 位数。AX、BX、CX、DX 四个数据寄存器用以存放操作数。

8086/8088 也能处理 8 位数，图 1.3.1 中的 4 个 16 位数据寄存器也可以作为 8 个 8 位寄存器使用，图中打斜线部分即相当于 8 位微机中的通用寄存器。

8086/8088 中的堆栈指针 SP（Stack Pointer）类似于 8 位微机中的堆栈指针，用于确定在堆栈操作时，堆栈在内存中的位置。但 8086/8088 中 SP 还必须与 SS（Stack Segment Register）寄存器一起才能确定堆栈的实际位置。

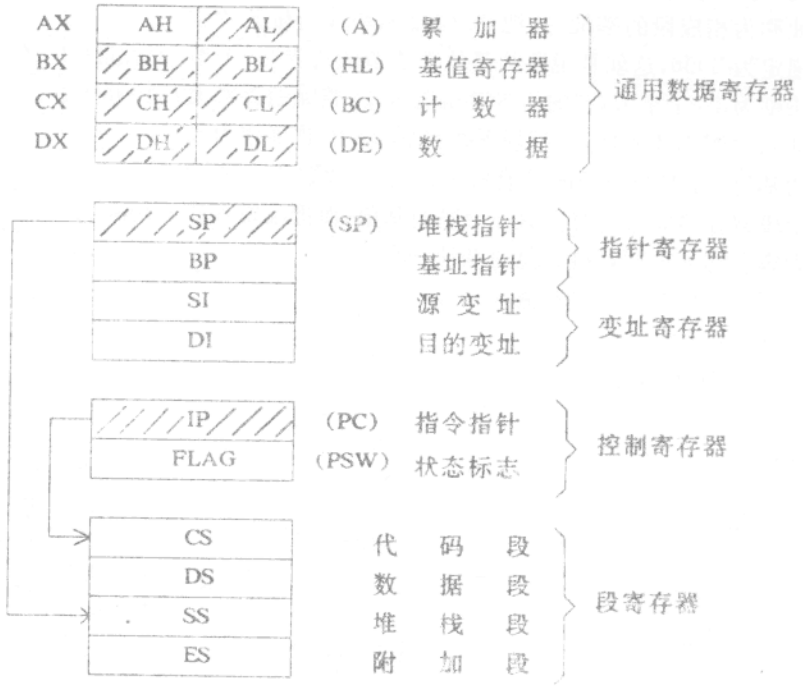

图 1.3.1 80866/8088 寄存器结构

在 8086/8088 中增加了三个 16 位寄存器，即基数指针寄存器 BP(Base Pointer Register)、源变址寄存器 SI(Source Index Register)和目的变址寄存器 DI(Destination Index Register)，还增加了几种寻址方式，从而能更灵活地寻找操作数。

8086/8088 中的指令指针 IP(Instruction Pointer)类似于 8 位微机中的程序计数器 PC。但它们也略有不同，后者的 PC 指向下一条即将要执行的指令，而在 8086/8088 中 IP 则指向下一次要取出的指令。另一方面，IP 要与 CS 寄存器(Code Segment Register)相配合才能形成真正的物理地址。

8086/8088 中的标志寄存器占用两个字节，共有九个标志位，此外 8086/8088 中还有 4 个 16 位的段寄存 CS、DS、SS、ES 使 8086/8088 能在 1MB 的范围内对内存进行寻址。

有关本节所涉及到寄存器及其使用，分别在以后有关章节中说明。

1.4 存贮器结构

8086/8088 有 20 条地址线，它的直接寻址能力为 1MB(即 $2^{20}\approx 1M$)。所以在一个系统中，可以有多达 1MB 的存贮器地址(从 00000H≈FFFFFH)。任意给定一个 20 位地址，就可以从这 1MB 中取出所需要的指令的操作数。8086/8088 CUP 内部的 ALU 只能进行 16 位运算。与地址有关的寄存器 SP、IP、以及 BP、SI、DI 等也都是 16 位的。因而对地址的运算也只能是 16 位。这就是说，对于 8086/8088 来说，各种寻找方式，寻找操作数的范围最多可能是 64KB。

现在的问题是如何在16位的ALU，64kB的指令寻址空间来解决8086/8088的1MB的寻址范围呢？解决的方法是，把不超过1MB的存贮空间划分为若干逻辑段，每段为连续的64KB地址组成。把各个段的起始地址称为相应段的基址，并把20位段的基址的最低4位规定为0000，这就是说段的基址都是节的边界（一个节长度为16个字节，节的边界就是能被16整除的存贮地址），如图1.4.1所示。把段基址的前16位数字称为段的基值。于是每个段的基值只有16位有效数字，这样4个段寄存器即可同时存放4个段的基值，而把段寄存器内容左移4位即可得到段的基址；即

段基址＝段基值×16

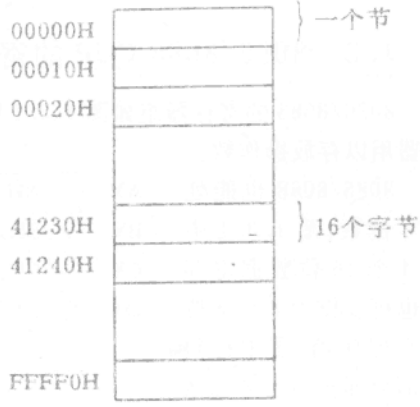

图1.4.1 可能的段地址

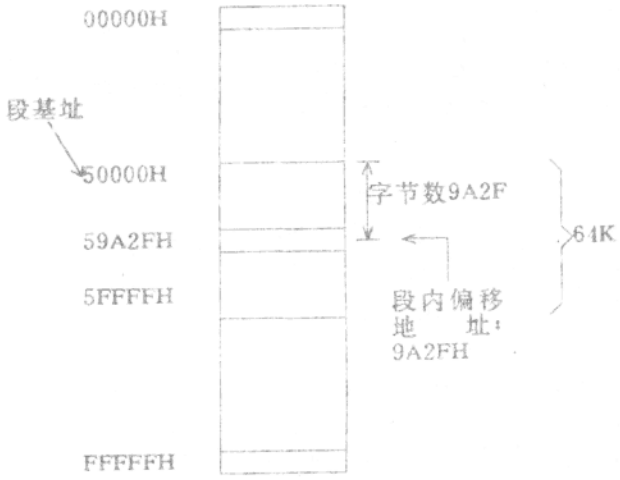

图1.4.2 64KB的逻辑段

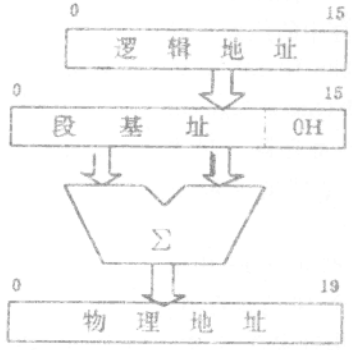

图1.4.3 8086/8088中物理地址的形成

某一存贮单元地址与它所在段的段基址之间的差值，称为该存贮单元的段内偏移量（或称段内偏移地址），如图1.4.2所示。

通过BIU的加法器$\sum$的段基址（即段地址）和偏移量相加，就可以得到20位的绝对地址（或称物理地址），它表示此存贮单元距离整个存贮器起始地址的字节数。

物理地址＝段基址＋偏移量＝段基值×16＋偏移量

如图1.4.3所示。

每次需要产生一个20位地址的时候，一个段寄存器会自动被选择，且能自动左移4位再与一个16位地址偏移量相加，产生所需要的20位物理地址。当取指令的时候，则自动选择代码段寄存器CS，再加上由IP所决定的16位偏移量，计算得到要取的指令的物理地址。当涉及到一个堆栈操作时，则自动选择堆栈段寄存器SS，再加上由SP所决定的16位偏移量，计算得到堆栈操作所需要的20位物理地址。当涉及到一个操作数时，则自动选择数据段寄存器DS或附加段寄存器ES，再加上16位偏移量，计算得到操作数的20位物理地址。16位的偏移量由寻址方式决定，可以是包含在指令中的直接地址，也可以是某一个16位寄存器中的值，也可以

是指令中的偏移量加上 16 位某个寄存器中的值等等。

当把存贮器分段以后，在程序执行时，微处理器会根据不同情况选择段寄存器。程序员设计的程序可以访问多至 4 个不同的地址空间，每个地址空间多至 64KB 的代码或数据。图 1.4.4 给出如何配置段寄存器的例子。在图 1.4.4 中，把段寄存器置为能访问最大的存贮量。按照这个配置，我们有 64KB 的指令，64KB 的堆栈空间，以及两个 64KB 的数据空间。图 1.4.5 给出一个更实际的配置。这里，我们在代码段中有一个 8KB 的程序，这个程序涉及到数据段里 2KB 的数据，并用 256 个字节的堆栈。由于数据存贮的需要是很小的，所以附加段就没有必要独立取一段了，因此 ES 寄存器置成与数据段重迭一块即可。若某一个任务所需的总存贮器长度（包括程序长度、堆栈长度和数据长度等）不超过 64KB，则可在程序开始时使 CS、SS、ES 和 DS 相等，程序也能正常工作。

这种存贮器分段方法，对于一个程序中要用的数据区超过 64KB 或要求从两个（或多个）不同的区域中存取操作数时，只要在取操作数以前，用指令给数据段寄存器重新赋值就可以了。

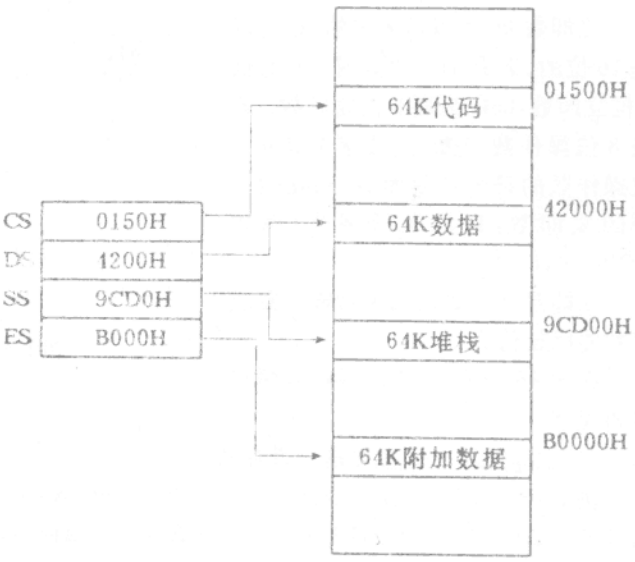

图 1.4.4 最大的段配置

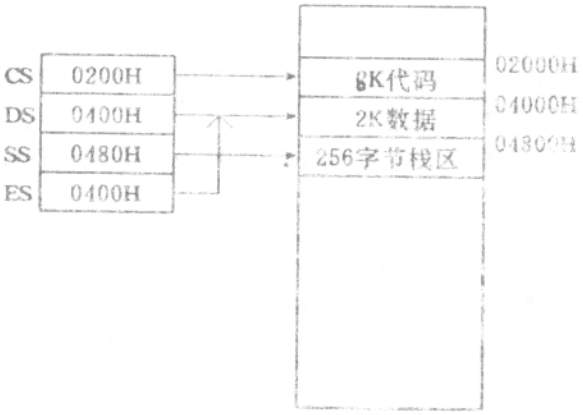

图 1.4.5 较小的段配置

1.5 8086/8088 的指令系统

本节介绍 8086/8088 的部分指令、寻址方式和标志寄存器的标志位的作用，以利于进一步学习程序设计方法。

1.5.1 寻址方式

8086/8088 包含 8080/8085 八位微机的全部寻址方式，由于 8086/8088 CPU 内部增加了几个有关地址的寄存器（如 BP、SI 和 DI 等），因此，PC 机的寻址能力有很大发展。取得操作数地址的方法，称为寻址方式。基本的寻址方式有六种。

1. 立即寻址

这种寻址方式提供的操作数直接包含在指令中。操作数紧跟在操作码的后面，与操作码一起放在代码段区域中。如图 1.5.1 所示

例:MOV AX,im

立即数 im 可以是 8 位的,也可以是 16 位的。若是 16 位的,则 imL 为低 8 位立即数,imH 为高 8 位立即数。若是 8 位操作数,它的高位字节是由 8 位操作数的符号扩展而成,则在指令中的立即数,只具有低字节(例如 MOV AX,5)。

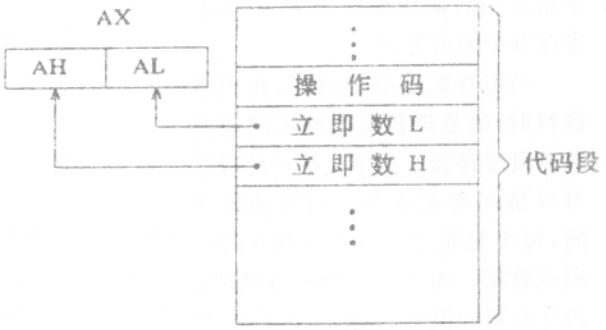

图 1.5.1 立即寻址方式示意图

立即寻址主要是用来给寄存器或存贮器赋初值。

在 8086/8088 中立即寻址有两点需要说明:

(1)立即数不允许直接送段寄存器

例如:若程序已用伪指令定义了一个名为 SEG-A 的段为数据段。现在需要为访问这个数据段而给 DS 寄存器设置段地址(以节为单位),应编程为:

MOV AX,SEG-A

MOV DS,AX

不应写成

MOV DS,SEG-A

段名 SEG-A 是个符号化的立即数(以节为单位的段地址值),立即数是不能以 MOV 指令直接送入段寄存器的。

(2)给存贮单元(或存贮区)赋初值时,变量名必须先定义后运用。

例如有下面两条指令

MOV VALUE1, 1234H

MOV VALUE2, 12H

此两条指令表面上看都是立即数送存贮器,但前者一般认为 VALUE1 是定义为字;而后者 VALUE2 一般定义为字节;但也可以认为定义为字,这时看作把 0012H 送存贮器,所以变量名应用前必须先定义,才不至含糊不清。关于变量名如何定义在第二章中我们还会详细讲到。

2. 直接寻址

操作数地址的 16 位偏移量直接包含在指令中,它与操作码一起存放在代码段区域。操作数一般在数据段,它的地址为 DS 的内容左移 4 位加上这 16 位地址偏移量,如图 1.5.2 所示。

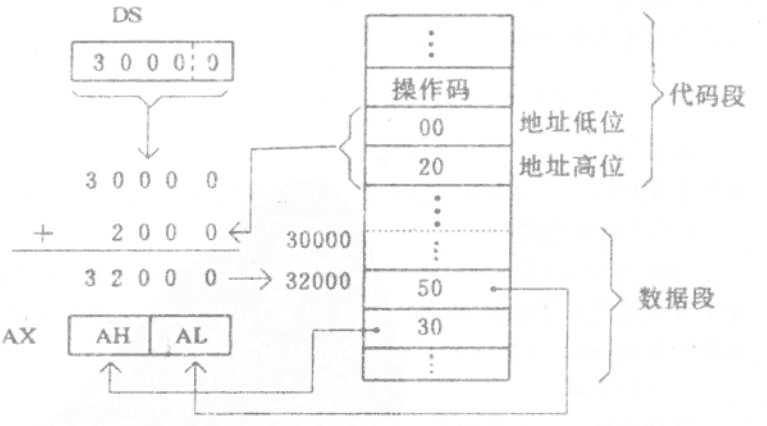

图 1.5.2 直接寻址示意图

例如:MOV AX,DS:[2000]

指令中的16位地址偏移量是低位字节在前，高位字节在后，这种寻址方法是以数据段的地址为基础，故可在多达64kB的范围寻找操作数。

直接寻址更普遍的情况可写成：

```
MOV   AX,VALUE1
```

VALUE1是事先在数据段定义的存贮单元的变量名，当汇编后此变量名也是一个16位的地址偏移量。此指令的意思是把VALUE1单元的内容送AL，VALUE1+1单元的内容送AH。

3. 寄存器寻址

操作数包含在CPU的内部寄存器中，例如AX、BX、CX、DX等。

```
例如：MOX   AX,BX          ;AX←BX
      MOV   DS,AX          ;DS←AX
```

虽然操作可在CPU内部通用寄存器的任意一个中，且它们都能参与算术或逻辑运算、存放运算结果等，但是AX是累加器，若结果存放在AX中，则通常指令执行时间要短些。

4. 寄存器间接寻址

在这种寻址方式中，操作数是在存贮器中。但是，操作数地址的偏移量包含在以下四个寄存器SI、DI、BP、BX之一中。这又可以分成两种情况：

(1)若以SI、DI、BX间接寻址，通常操作数在现行数据段中。也即DS的内容左移4位后加上SI、DI、BX中的16位偏移量作为操作数的地址。例如

```
MOV   AX,DS:[SI]
MPV   DS:[BX],AL
```

由于SI、BX和DI规定在数据段中寻找操作数(即用DS作段寄存器)，所以在指令中可以省缺DS，所谓默认(Default)状态。即上两条指令写为：

```
MOV   AX,[SI]
MOV   [BX],AL
```

重要的是SI、BX寄存器要用传送指令先给它们赋值，然后才能运用。赋值指令可以有两种形式，例如给SI赋值：

①MOV SI,OFFSET BUFFER

②LEA SI,BUFFER

这两条指令的意思都是把16位的地址偏移量BUFFER送SI寄存器。BUFFER必须在数据段先定义，经汇编程序汇编后BUFFER就是一个16位地址编移量。关于这方面内容下一章有详细叙述

(2)若是以寄存器BP间接寻址，则操作数在堆栈段中，即以SS寄存器内容左移4位与BP相加作为操作数地址。例如

```
MOV AX,[BP]
```

BP寄存器同样要用传送指令或LEA指令先赋值，然后才能应用。

5. 变址寻址

变址寻址就是以指令寄存器(SI、DI、BX、BP之一)的内容，加上指令中给定的8位或16位偏移量之和作为该指令操作数的总偏移量。例如：MOV AX,COUNT[SI]可用图1.5.3表示寻址示意图。

在正常情况下，若用SI、DI和BX的内容作为变址值，则与数据段寄存器相加，形成操作数地址；若用BP的内容作为变址值，则与堆栈段寄存器相加，形成操作数地址。

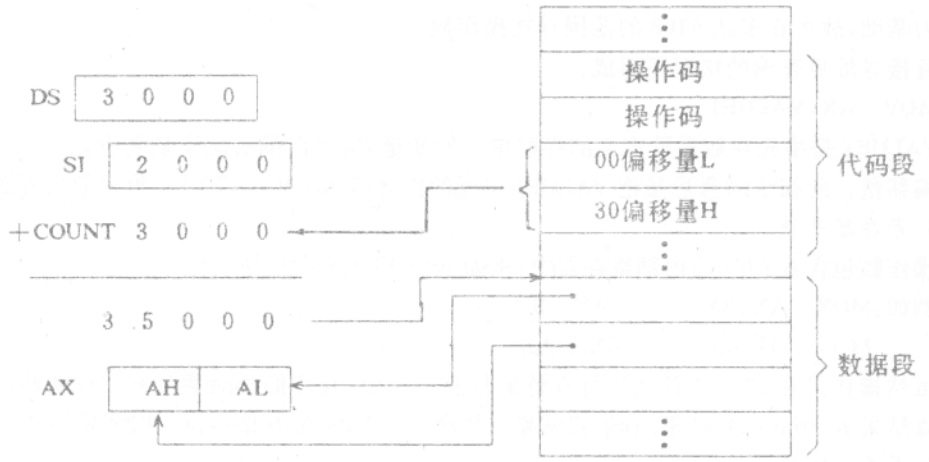

图 1.5.3 变址寻址示意图

6. 基址加变址寻址

在 8086/8088 中，通常把 BX 和 BP 看作是基址寄存器，把 SI、DI 看作变址寄存器，这种寻址方式是把一个基址寄存器(BX 或 BP)的内容，加上一个变址寄存器(SI 或 DI)的内容再加上指令中指定的 8 位或 16 位偏移量的三项之和作为操作数的总偏移量地址如图 1.5.4 所示。

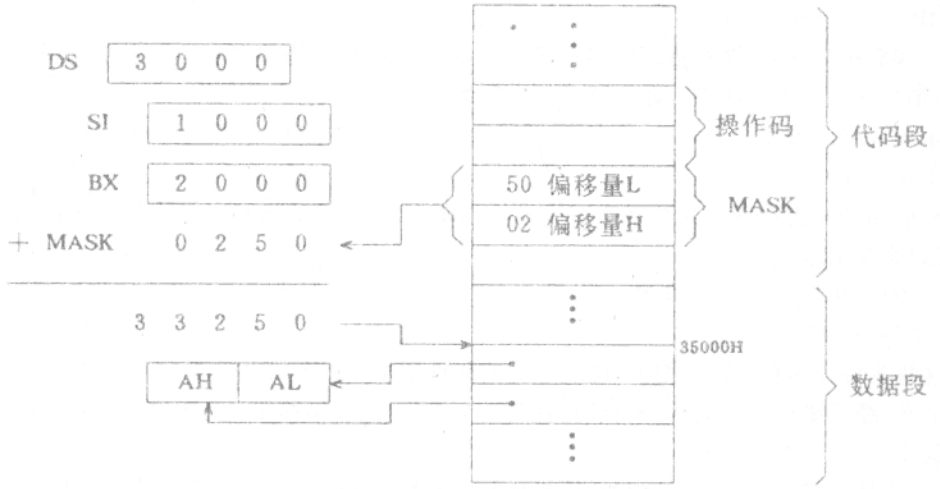

图 1.5.4 基址加变址寻址示意图

例如：MOV AX，MASK[BX][SI]

在正常情况下，由基地址决定哪一个段寄存器作为地址指针；即若用 BX 作为基地址，则操作数在数据段区域中；若用 BP 作为基地址，则操作数在堆栈段区域中。

如上所述，8086/8088 中的存贮器是分段的，寻找一个内存操作数，只能在某一个段的 64kB 范围内寻找，以什么寄存器间址、变址与基址加变址，则操作数应在什么段区域中，在 8086/8088 中有一个基本约定，只要在指令中不特别说明要超越这个约定，则正常情况就按这个基本约定来寻找操作数，这就是所谓的 Default(默认)状态。这些基本约定和允许超越的情况如表 1.1 所示。

表 1.1

存贮器存取方式	约定段基值	可修改的段基值	逻辑地址
取指令操作	CS	无	IP
堆栈操作	SS	无	SP
源串	DS	CS、ES、SS	SI
目的串	ES	无	DI
用 BP 作为基寄存器	SS	CS、DS、ES	有效地址
通用数据读写	DS	CS、ES、SS	有效地址

1.5.2 标志寄存器

标志寄存器反映运算结果的状态。条件转移指令用于测试这些状态以决定改变程序的流向。8086/8088 中有一个标志寄存器，占用 2 个字节，共有 9 个标志位，如图 1.5.5 所示。

各标志位功能分述如下

1. 辅助进位标志 AF

在字节操作时若由低半字节(一个字节的低 4 位)向高半字节有进位或借位；在字操作时，低位字节向高位字节有进位或借位，则 AF=1，否则为 0。这个标志用于十进制算术运算指令中。

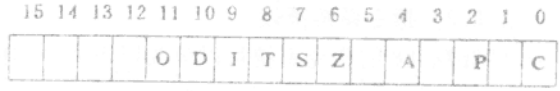

图 1.5.5 8086/8088 中的标志位

2. 进位标志 CF

当结果的最高位(字节操作时 D_7 或字操作时的 D_{15})产生一个进位或借位，则 CF=1，否则为 0。这个标志主要用于多字节的加、减法运算。移位和循环移位指令也能够把存贮器或寄存器中的最高位(左移时)或最低位(右移时)放入标志 CF 中。

3. 溢出标志 OF

在算术运算中，带符号数的运算结果超出 8 位或 16 位带符号数能表达的范围，即字节运算时＞＋127 或＜－128，在字运算时＞32767 或＜－32768 时，此标志置“1”。在计算机中是如何判断有溢出呢？如果算术运算结果，次高位产生一个进位到最高位，而最高位却没有进位，或反之亦然，则 OF=1，否则 OF=0。

进位标志和溢出标志指出：算术运算结果要存到目的操作数中的数是太大了。之所以需要这样两种标志是由于这样的事实：操作数可以是无符号数(即正数)或有符号数(补码表示)。如果无符号数操作结果在目的操作数中表示不了时，则 CF=1；对字节操作，这意味着结果超出 255；对于字操作，意味着超过 65535。如果有符号的数运算结果在目的操作数里表示不了时，即字节操作，意味着结果大于 127 或小于－128；字操作，意味着结果大于 32767 或小于－32768，则 OF=1。

4. 符号标志 SF

运算结果为负时 SF=1，结果为正 SF=0。

由于在 8086/8088 中有符号数是用补码表示的，所以 SF 表示了结果的符号，SF=0 为正，SF=1 为负。

5. 奇偶标志 PF

若操作结果 1 的个数为偶数，则 PF=1，否则 PF=0。这个标志可用于检查在数据传送过程中是否发生错误。

6. 零标志 ZF

若运算结果为 0，则 ZF=1，否则 ZF=0。

7. 方向标志 DF

该标志在串操作指令时有效。若 DF=1，则引起串操作指令为自动减量指令，也就是从高地址到低地址处理字符串，若 DF=0，则串操作指令就为自动增量指令，即以低地址向高地址处理字符串。

8. 中断允许标志 IF

若用指令 STI 置 IF=1，则允许 CPU 接收外部的可屏蔽中断请求；若用指令 CLI 置 IF=0，则屏蔽上述中断请求；但对内(部产生的)中断不起作用。

9. 追踪标志 TF

置 TF 标志，使处理进入单步方式，以便调试。在这个方式中，CPU 在每条指令执行以后，产生一个内部中断，允许程序在每条指令执行以后进行检查。正常情况，TF 处于零状态。

1.5.3 指令系统概述

8086/8088 的指令系统可以分为以下六个功能组(图 1.5.6 所示)。

1. 数据传送指令

MOV	传送	XCHG	交换
PUSH	压入堆栈	IN,OUT	I/O 口
POP	弹出堆栈	XLATB	转换
LEA	偏移量传送	LDS	地址传送

2. 算术运算指令

ADD	加	NEG	取补
SUB	减	CMP	比较
INC	递增	MUL	乘
DEC	递减	DIV	除

3. 逻辑运算指令

AND	逻辑乘	TEST	位测试
OR	逻辑或	SHL/SHR	左/右移
XOR	异或	ROL/ROR	左/右环移
NOT	取反		

4. 串操作指令

MOVS	串传送	STOS	串存贮
LODS	串装入	SCAS	串扫描
CMPS	串比较		

5. 程序控制指令

CALL	子程序调用	JMP	无条件转移
RET	子程序返回	JZ,JNZ	条件转移
INT	中断	LOOP	循环控制
IRET	中断返回	LOOPNE	条件循环

6. 处理器控制指令

CLC	清进位	CLI	清中断标志	CLD	清方向标志
STC	置进位	STI	置中断标志	STD	置方向标志
HLT	暂停				

图 1.5.6 8086/8088 主要指令集

1.5.4 关于MOV指令、算术指令和逻辑指令说明

1. 关于MOV、算术、逻辑指令共性说明

这六类指令中，传送指令、算术指令和逻辑指令占了指令集中的大部，它们在指令的形式上各不相同，但从寻址方式上考察几乎都是一样的，所以对这些指令宜抓住它们的共同特点加以叙述，可收到举一反三的作用。

首先以MOV指令为例，复习一下寻址方式

```
(1)MOV AX,5                    立即寻址
(2)MOV AX,BX                   寄存器寻址
(3)MOV AX,VALUE1               直接寻址
(4)MOV AX,[BX]                 间接寻址
(5)MOV AX,[SI+5]               变址寻址
(6)MOV AX,[BX+SI+5]            基址加变址寻址
```

上面6条指令中都是选一个字的数据并把它传送给AX寄存器。第一个例子给出立即寻址方式，在指令本身中指定常数5送给AX寄存器。第二个是简单的寄存器与寄存器的传送，即把BX寄存器的内容送到AX寄存器。第三个例子是一个直接寻址，以符号"VALUE1"指定的内存地址的内容送到AX寄存器。第四个例子是间接寻址，把BX寄存器的内容作为存贮单元地址，并从该地址中取出内容，送到AX寄存器。在第五个例子中，SI寄存器保持的也是存贮单元地址，但在这里还要加上位移量5才能确定要找的存贮单元地址。最后一个例子较复杂：BX的内容、SI的内容和位移量5都加到一起才能确定所要数据的那个存贮器单元地址。

除立即方式外，在上面的例子中，源寄存器(或源地址)可以和目的寄存器对调，上述指令仍然是合理可行的。从例2到例6可以改写为：

```
MOV     BX,AX
MOV     VALUE1,AX
MOV     [BX],AX
MOV     [SI+5],AX
MOV     [BX+SI+5],AX
```

上面讨论的MOV指令的寻址方式中，都是以字操作为例的，实际上对于字节操作也是适用的.

```
MOV     AL,5
MOV     AL,BL
MOV     AL,VALUE2
MOV     AL,[SI]
MOV     AL,[BX+5]
MOV     AL,[SI+BX+5]
```

上面讨论的寻址方式也适用于算术和逻辑运算类指令。例如：

```
ADD     AL,30
ADD     AX,3000H
ADD     AX,SI
ADD     AL,[BX+5]
 ⋮
SUB     AL,30
SUB     AX,3000H
SUB     AX,SI
SUB     AL,[BX+5]
```