周明德　白晓笛　编著

高档微型计算机

下　册

清华大学出版社

高档微型计算机

（下）

周明德　白晓笛　编著

清华大学出版社

内 容 简 介

本书从8086入手，系统地分析了8086—80286—80386的结构，全面地介绍了它们的指令系统，以8086和80386为重点分别介绍了它们的汇编语言程序设计。本书以80386为重点，详细地介绍了它的几种工作方式、实地址方式，保护虚地址方式和虚拟的8086方式。介绍了它的保护机构，各种中断和异常处理。比较详尽地介绍了80386的各种时序。本书是学习和掌握以8086、80286、80386为CPU的各种微机的必读课本，是大学本科生，研究生学习高档微机的重要参考书。

高档微型计算机

（下）

周明德 白晓笛 编著

☆

清华大学出版社出版

北京 清华园

北京京辉印刷厂印刷

新华书店北京发行所发行

☆

开本：787×1092 1/16 印张：24 字数：614 千字

1989年5月第1版 1989年5月第1次印刷

印数：0001—8000

ISBN 7-302-00390-4/TP·129

定价：10.65元

前　言

美国Intel公司是世界主要的微处理器制造商。它的16位至32位微处理器芯片——8086—80186—80286—80386，由于被IBM公司采用作为其个人计算机系统(Personal System)的CPU芯片而得到更为广泛的重视。1987年春，IBM公司推出新的个人计算机系统——PS/2，震动了计算机界。此后，以80286为CPU的各种AT兼容机大量上市，使得在1987年AT及其兼容机的销售量第一次超过了PC/XT及其兼容机，成为当今世界个人计算机的主流机型。以80386为CPU的各种32位个人计算机、工作站和多用户系统也相继上市，这标志着微型计算机向更高档的方向发展。

80386（同样包括MC68020）具有极强大的功能，主振频率已达16—20MHz，执行指令的速度已达2—4 MIPS，直接寻址能力达到4 G字节，已经大大超过了80年代初超级小型机的能力，而价格却十分低廉。以80386为CPU的AT兼容型的个人计算机，其速度为PC/XT的10倍，内存2—4 M字节，外存为40—80M硬盘，价格仅为几千美元。以80386为CPU的工作站，已经给Apollo和Sun工作站造成极大的威胁。80386的强大功能，使其足以构成由几十个终端组成的多用户、多任务系统，足以向超级小型机甚至中大型机系统挑战。

8086—80286—80386是向上兼容的，分析80386必须从8086着手，学习80386也必须从8086开始。为了迎接今后二、三年内主流机型的到来，我们把本书奉献给广大读者。

周明德

一九八八年一月十五日

目　录

第一章　Intel 8086的结构

Intel 8086（简称 8086）是一种16位微处理器，是最早投入市场的16位微处理器产品之一。它是在Intel 8080与8085的基础上发展起来的一种16位微处理器。它的内部结构是 16位的，数据总线也是16位宽的。它能处理16位数据(具有16位运算指令包括乘法和除法指令)，也能处理 8 位数据。它能执行整套8080/8085的指令，所以它在汇编语言上与8080/8085是兼容的，并且还增加了许多16位操作指令。它有 20 条地址引线，所以直接寻址能力达到 1 M 字节。

第一节　8086 的编程结构

一、8086的寄存器结构

8086的寄存器结构如图 1-1 所示。它的内部寄存器都是16位的，所以能处理16位数。最上面的 4 个寄存器是 4 个16位的数据寄存器，用以暂存16位的操作数。其中 AX 是累加器，其它三个是通用的数据寄存器，用以暂存参与运算的操作数。但它们的通常用途可用表 1-1来说明。

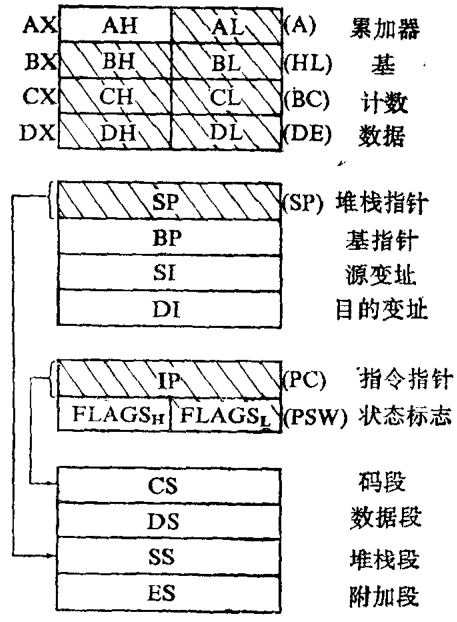

图 1-1　8086 的寄存器结构图

8086也能处理 8 位数，图 1-1 中的 4 个16位数据寄存器也可作为 8 个 8 位寄存器使用。图中打斜线的部分即相当于8080和8085中的通用寄存器。

8086中的堆栈指针SP（Stack Pointer）类似于8080和8085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。但是在8086中，SP还必须与SS（Stack Segment 堆栈段寄存器）一起才能确定堆栈的实际位置（关于几个段寄存器的功能，我们在后面详细介绍）。

在8086中增加了三个16位寄存器即 BP（Base Pointer Register）、SI（Source Index Register）和DI（Destination Index Register），增加了几种寻址方式，从而能更灵活地寻找操作数。它们的具体功能在第二章寻址方式这一部分再详细介绍。

8086中的指令指针IP（Instruction Pointer），类似于8080和8085中的程序计数器 PC（Program Counter）。但是，它们略有区别。一方面8080和8085中的PC总是指向下一条即将执行的指令，在8086中的IP总是指向下一次要取出的指令，在8086中这两者是有区别的；另一方面，在8086中IP要与CS（Code Segment）寄存器相配合才能形成真正的物理地址。

表 1-1 寄存器的通常用途

寄存器	操作
AX	字乘法，字除法，字I/O
AL	字节乘法，字节除法，字节I/O，转移，十进制算术运算
AH	字节乘法，字节除法
BX	转移
CX	串操作，循环次数
CL	变量移位或循环
DX	字乘法，字除法，间接I/O

在8080和8085中，状态寄存器PSW（Processor Status Word）是一个字节，有5个标志位；在8086中保留了这5个标志位，又增加了4个标志位，所以要占两个字节。

此外，在8086中，还有四个16位的内存段寄存器：码段寄存器CS（Code Segment Register）、数据段寄存器DS（Data Segment Register）、堆栈段寄存器（Stack Segment Register）、附加段寄存器ES（Extra Segment Register），使8086能在1M字节的范围内对内存进行寻址。

二、8086的功能结构

8086 CPU从功能上来说分成两大部分：总线接口单元BIU（Bus Interface Unit）和执行单元EU（Execution Unit），如图1-2所示。

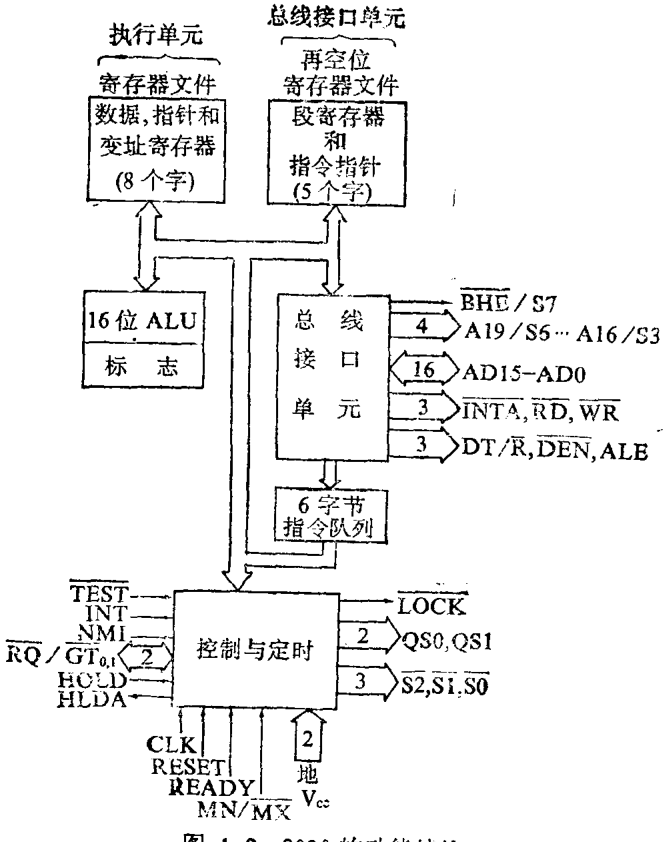

图 1-2 8086的功能结构

BIU负责与存储器接口，即8086 CPU与存储器之间的信息传送，都是由BIU进行的。具体地说，即BIU负责从内存的指定部分取出指令，送至指令流队列中排队；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部分负责指令的执行。这样，取指部分与执行指令部分是分开的，于是在一条指令的执行过程中，就可以取出下一条（或多条）指令，在指令流队列中排队。在一条指令执行完以后就可以立即执行下一条指令，减少了CPU为取指令所需要的等待时间，提高了CPU的利用率，提高了整个运行速度。

在8080和8085以及Z80等标准的8位微处理器中，程序的执行是由取指和执行指令的循环来完成的。即执行

的顺序为取第一条指令，执行第一条指令；取第二条指令，执行第二条指令；……直至取最后一条指令，执行最后一条指令。这样，在每一条指令执行完以后，CPU必须等待，直至下一条指令取出来以后才能执行。所以，它的工作顺序如图 1-3 所示。

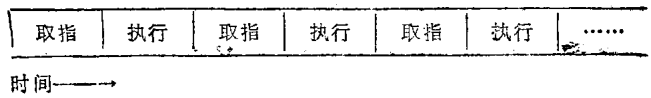

图 1-3　8 位微机的指令执行顺序

但在8086中，由于BIU和EU分开，所以，取指和执行可以重叠，它的执行顺序如图1-4所示。于是就大大减少了等待取指所需的时间，提高了CPU的利用率。一方面可以提高整个执行速度，另一方面又降低了对与之相配的存储器的存取速度的要求。这种重叠的操作技术称为指令的流水线结构，过去只在大型机中才使用。

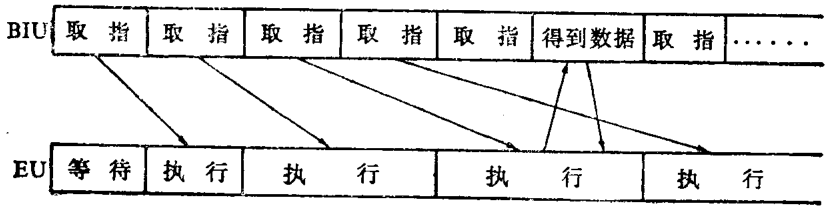

图 1-4　8086 指令的执行顺序

第二节　8086 中的存器储组织

8086有20条地址引线，它的直接寻址能力为 2^{20} = 1 M字节。所以，在一个8086组成的系统中，可以有多达 1 M字节的存储器。这 1 M字节逻辑上可以组织成一个线性矩阵。地址从00000H到FFFFFH。给定一个20位的地址，就可以从这 1 M字节中取出所需要的指令或操作数。但是，在8086内部，这20位地址是如何形成的呢？如前所述，8086 内部的 ALU 能进行16位运算，有关地址的寄存器如IP、SP，以及BP、SI、DI等也都是16位的，因而8086对地址的运算也只能是16位的。这就是说，对于8086来说，各种寻址方式，寻找操作数的范围最多只能是64K字节。所以，整个 1 M字节存储器以64K为最大范围分为若干段。在寻找一个具体物理单元时，必须要由一个基地址再加上由SP或IP或 BP 或 SI 或 DI等可由CPU处理的16位偏移量来形成实际 的 20 位物理地址。这个基地址就是由8086中的段寄存器即CS、SS、DS和ES中的一个来形成 的。在 形 成 20位物理地址时，段寄存器中的16位数会自动左移 4 位，然后与16位偏移量相加，以形成20位地址，如图 1-5 所示。

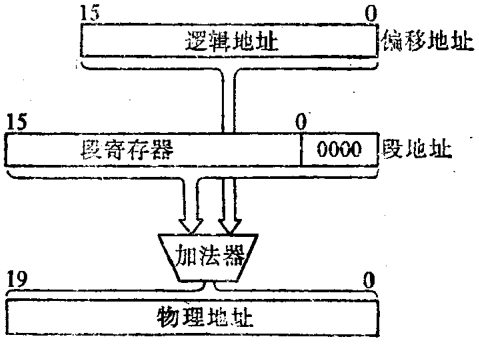

图 1-5　8086 中的物理地址的形成

每次在需要产生一个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位的地址偏移量相加，以产生所需要的20位物理地址。每当取指令的时候，则自

动选择码段寄存器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。每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。每当涉及到一个操作数，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。在这儿的16位偏移量，可以是包含在指令中的直接地址，也可以是某一个16位地址寄存器的内容，也可以是指令中的位移量加上16位地址寄存器中的内容等等。这取决于指令的寻址方式。

所以，在8086系统中，存储器的访问如图 1-6 所示。

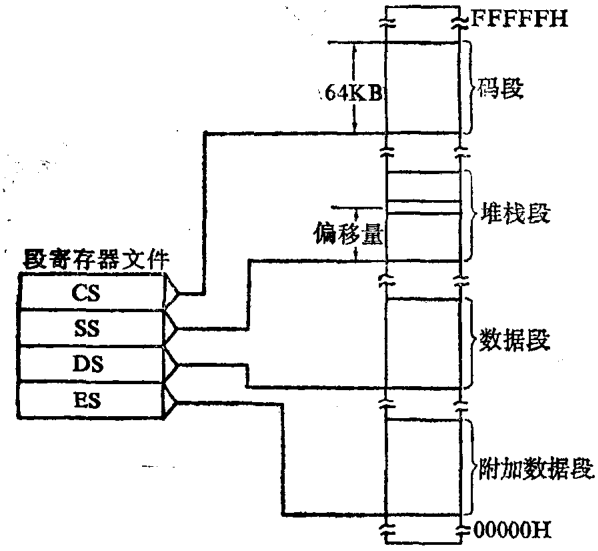

图 1-6　8086对存储器的访问

在不改变段寄存器值的情况下，寻址的最大范围是64KB。所以，若有一个任务，它的程序长度、堆栈长度以及数据区长度都不超过64KB，则可在程序开始时，分别给DS、SS和CS赋值，然后在程序中就可以不再考虑这些段寄存器，程序就可以在各自的区域中正常地进行工作。若某一个任务所需的总的存储器长度（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、DS相等，程序也能正常工作。

上述这种存储器分段的方法，对于要求在程序区、堆栈区和数据区之间互相隔离的一类任务来说是非常方便的。

这种存储器分段方法，对于在一个程序中要用的数据区超过64KB，或要求从两个（或多个）不同区域中去存取操作数的任务来说，也是十分方便的。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。

这种分段方法，也适用于程序的再定位要求。在不少情况下，要求同一个程序能在内存

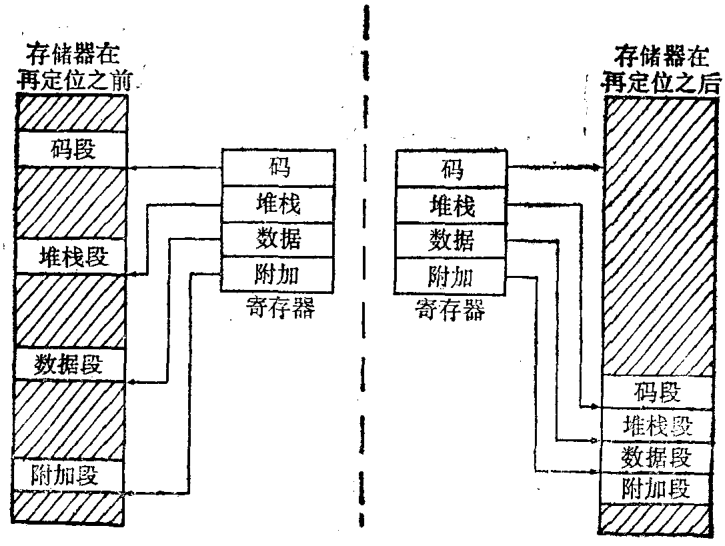

图 1-7　存储器的再定位

的不同区域中运行，而不改变程序本身。这在8086中是可行的。只要程序中的转移指令都使用相对转移指令，而在运行这个程序前，设法改变各个段寄存器的值就可以了，如图 1-7 所示。

在8086的存储器中，仍以一个字节作为一个基本存储单元，而16位微处理器的基本操作长度是一个字（16位），这两者之间是如何协调的呢？实际上 1 M字节存储器可以分成两 个体（bank），即从地址来说，凡是偶数地址单元形成了一个体，它们的数据线连到 数 据 总线的低 8 位（D 7 —D 0）；而奇数地址单元形成另一个体，它们的数据线连到数据总 线 的高 8 位（D15—D8）。CPU是用地址线A19—A1来区分512K不同的地址，而由 A0 来区分两个体。8086提供了两条信号线$\overline{\text{BHE}}$和A0，以决定是访问高序字节还是低序字节还是整个字，如表 1-2 所示。

根据指令，BIU会自动完成所需访问存储器的次数。若从偶数地址开始访问一个字，则只需一个存储器访问周期；若从奇数地址开始访问一个字，则需两个存储器访问周期。

表 1-2 对存储器高低字节的访问

$\overline{\text{BHE}}$	A0	性　　能
0	0	访问整个字
0	1	高序字节
1	0	低序字节
1	1	无

第二章　8086的寻址方式和指令系统

第一节　8086的寻址方式

8086包含有8080和8085的全部寻址方式。此外，由于在8086 CPU的内部增加了好几个有关地址的寄存器如BP、SI和DI等，因而使8086的寻址方式有了很大的发展。

一、立即寻址（*Immediate Addressing*）

这种寻址方式所提供的操作数直接放在指令中，紧跟在操作码的后面与操作码一起放在码段区域中，如图2-1所示。

立即数im可以是8位的，也可以是16位的。若是16位的，则imH放在高地址字节（与在8080和8085中的相同）。若是字操作数，但它是低位字节符号扩展的，则在指令中的立即数只具有低位字节。

立即寻址主要是用来给寄存器或存储单元赋初值。

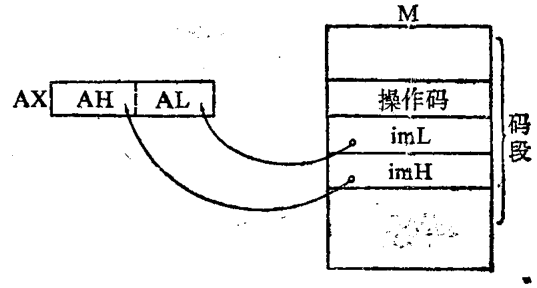

图 2-1　立即寻址方式

二、直接寻址（*Direct Addressing*）

操作数的地址的16位偏移量，直接包含在指令中，它与操作码一起在码段区域中。但操作数一般在数据段区域中，它的地址为数据段寄存器DS的内容左移4位，再与指令中的这个16位地址偏移量相加，如图2-2所示。

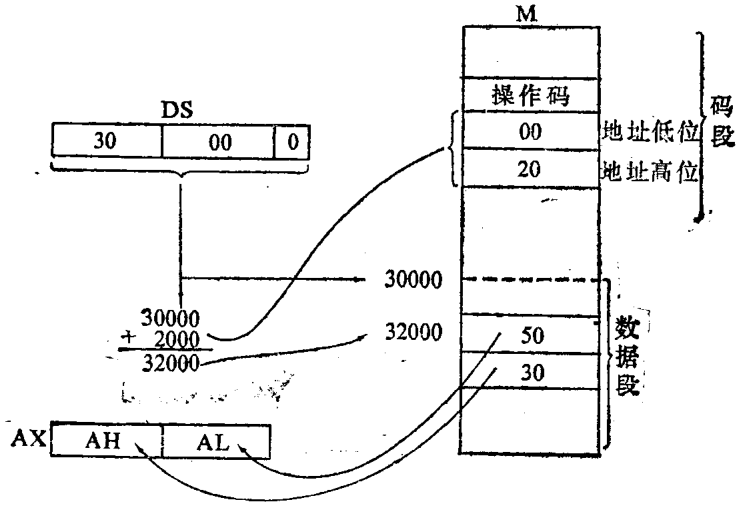

图 2-2　直接寻址方式

指令中的16位地址偏移量是低位字节在前，高位字节在后。

这种寻址方式，是以数据段寄存器的地址为起点，可在多达64KB的范围内寻找操作数。而且，在8086中允许段超越，即对于寻找操作数来说，还允许操作数在以码段寄存器或堆栈段寄存器或附加段寄存器为基准的区域中，即只要在指令中指明是段超越的，则16位地址偏移量可以与CS或SS或ES相加，作为操作数的地址。

三、寄存器寻址（*Register Addressing*）

操作数包含在CPU内部的任一个通用寄存器中，即可在AX、BX、CX、DX、BP、SP、SI、DI的任一个中，如图2-3所示。

虽然操作数可在CPU的任一个通用寄存器中，且它们都能参与算术和逻辑运算，都能作为目的寄存器存放运算的结果，但是，AX是累加器，若结果存放在AX中的话，则通常指令要更短些、更紧凑一些。

图 2-3 寄存器寻址

四、寄存器间接寻址（*Register Indirect Addressing*）

在这种寻址方式中，操作数是在存储器中。但是，操作数地址的16位偏移量包含在以下四个寄存器即BX、BP、SI和DI之一中。这又可以分成两种情况：

1．若以BX或SI或DI间接寻址，则类似于8080中的HL间接寻址，操作数通常在现行的数据段区域中。也即数据段寄存器DS加上SI或DI或BX中的16位偏移量，为操作数的地址，如图2-4所示。

若在指令中指定是段超越的，则SI、DI、BX也可以与其它的段寄存器相加，形成操作数的地址。

2．若是以寄存器BP间接寻址，则操作数在堆栈段区域中，即以堆栈段寄存器SS与BP相加，作为操作数的地址，如图2-5所示。

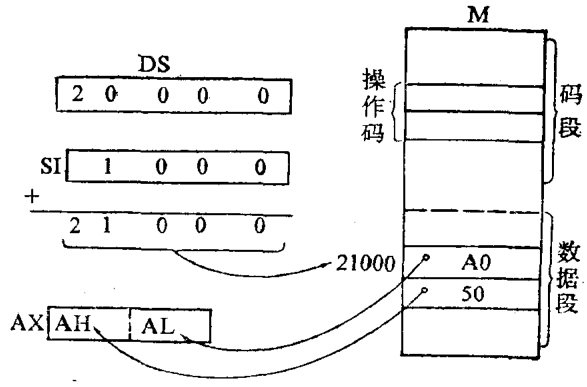

图 2-4 数据段的间接寻址

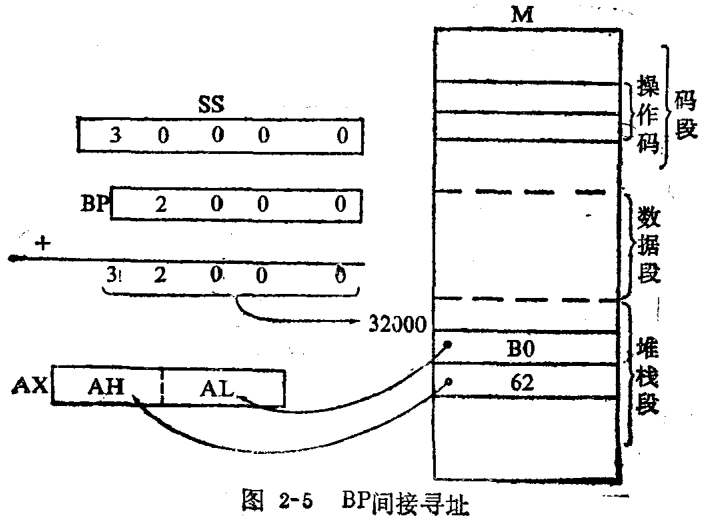

图 2-5 BP间接寻址

若在指令中规定是段超越的，则BP也可以与其他的段寄存器相加，形成操作数的地址。

五、变址寻址（*Index Addressing*）

上述可以作为寄存器间接寻址的四个寄存器SI、DI、BX、BP（BX、BP也可称为基址寻址），也可用作变址寻址。所谓变址寻址就是以指定的寄存器内容，加上指令中给定的8位或16位位移量，作为操作数地址的段内偏移量，如图2-6所示。

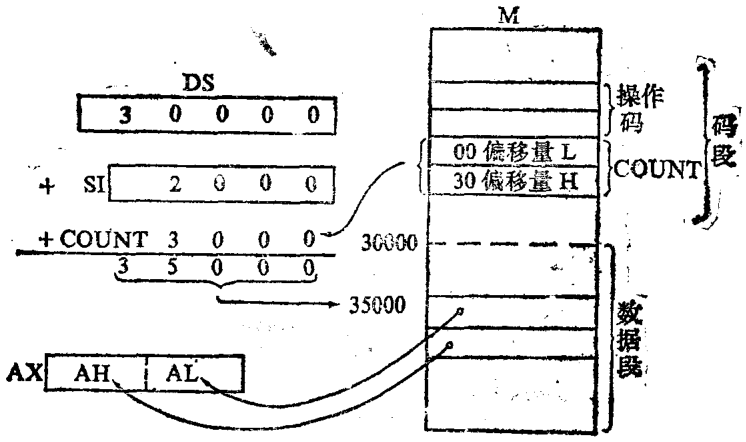

图 2-6 变址寻址

在正常情况下，若用SI或DI或BX作为变址寄存器，则操作数在数据段，以DS的内容作为段的起始地址；若用BP作为变址寄存器，则操作数在堆栈段，以SS的内容作为段的起始地址。

六、基址加变址的寻址方式

在8086中，通常把BX和BP看作基址寄存器，把SI、DI看作变址寄存器，可以把这两种寻址方式结合起来形成一种新的寻址方式。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，再加上指令中给定的8位或16位位移量作为操作数的地址的段内偏移量，与一个段寄存器相结合形成操作数的地址，如图2-7所示。

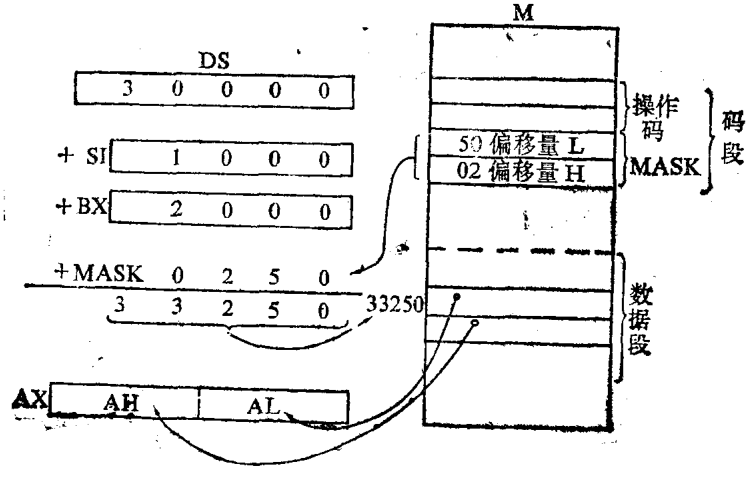

图 2-7 基址加变址寻址

在正常情况下，由基址寄存器决定操作数在哪一段。即若用BX作为基地址，则以DS作为段地址指针；若用BP作为基地址，则以SS作为段地址指针。

此外，8086中的条件转移指令都采用相对寻址方式，在满足条件时转移的目标地址为当前IP值加上指令中给定的以二进制补码表示的偏移量。

如上所述，8086中的存储器是分段的。我们寻找一个内存操作数，只能在某个段的64K字节范围内寻找。以什么寄存器间址、变址与基址加变址，则操作数在什么段区域中，在8086中有一个基本的约定。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，这就是所谓的默认（Default）状态。在8086中的这些基本约定和允许超越的情况，如表2-1所示。

表 2-1 8086中存储器基准的约定

存储器基准的类型	约定的段基准	可修改的段基准	逻辑地址
取指令	CS	无	IP
堆栈操作	SS	无	SP
源串	DS	CS、ES、SS	SI
目标串	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

表 2-2 寻址方式与数据结构

数据结构	数据存储器		堆栈
	不具有基址	有基址	
简单变量	直接	BX+OFFSET(偏移量)	BP+OFFSET
矩阵	SI DI	BX+SI BX+DI	BP+SI BP+DI
记录的矩阵	SI+OFFSET DI+OFFSET	BX+SI+OFFSET BX+DI+OFFSET	BP+SI+OFFSET BP+DI+OFFSET

表 2-3 有效地址与时钟周期

有效地址		时钟数*
只有偏移量		6
只有基址或变址寄存器 (BX、BP、SI、DI)		5
偏移量 + 基址或变址 (BX、BP、SI、DI)		9
基址 + 变址	(BP+DI、BX+SI) (BP+SI、BX+DI)	7 8
偏移量 + 基址 + 变址	BP+DI+DISP BX+SI+DISP BP+SI+DISP BX+DI+DISP	11 12

* 段超越再加两个时钟周期