

IBM-PC

指令系统与宏汇编

(计算所八室编)

中国科学院科技咨询开发服务部讲习班
北京信通电脑技术公司

前 言

本手册是为使用IBM个人计算机宏汇编语言的程序员而写的一本参考书。

本手册提供了如何使用MACRO汇编、交叉引用程序、伪操作等一些特殊的信息。

本手册的编排顺序如下：

第一章“引言”介绍MACRO汇编和它的性能。

第二章“MACRO汇编”对于在执行程序时所需要的命令给予了详细的介绍。

第三章“交叉引用程序(CREF)”介绍了一些功能，帮助你调试和列出程序。

第四章“汇编语言格式”解释了汇编语言的语句、数据、表达式和其它信息，使你能够有效地使用MACRO汇编。

第五章“伪操作”介绍了如何简化编码，减少产生错误的机会，和保证对标准程序的指令进行编码。

附录“信息”列出宏汇编过程中可能产生的一些信息。

第一部分 目 录

第一章 8086寄存器和标志	(1)
1.1 通用寄存器	(1)
1.2 指针寄存器	(2)
1.3 变址寄存器	(2)
1.4 段寄存器	(2)
1.5 标志寄存器	(2)
第二章 8086寻址方式	(4)
2.1 8086存贮器地址的计算	(4)
2.2 8086寻址方式	(4)
1)程序存贮器寻址方式	(5)
2)数据存贮器寻址方式	(5)
2.3 寻址方式字节	(9)
2.4 段替换	(10)
2.5 存贮器寻址方式表	(11)
第三章 指令系统助记符	(12)
第四章 8086汇编语言指令的详细介绍	(14)
第五章 8086指令组	(116)
5.1 数据传送指令	(116)
5.2 算术指令	(119)
5.3 逻辑指令	(123)
5.4 字符串指令	(128)
5.5 程序计数器控制指令	(128)
5.6 处理器控制指令	(132)

第二部分 目 录

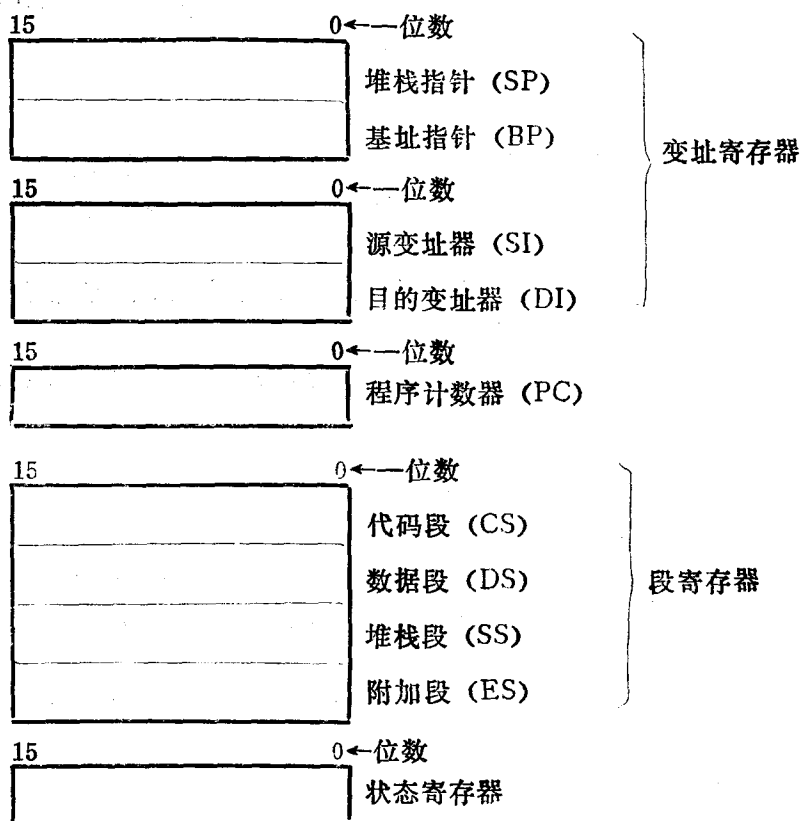
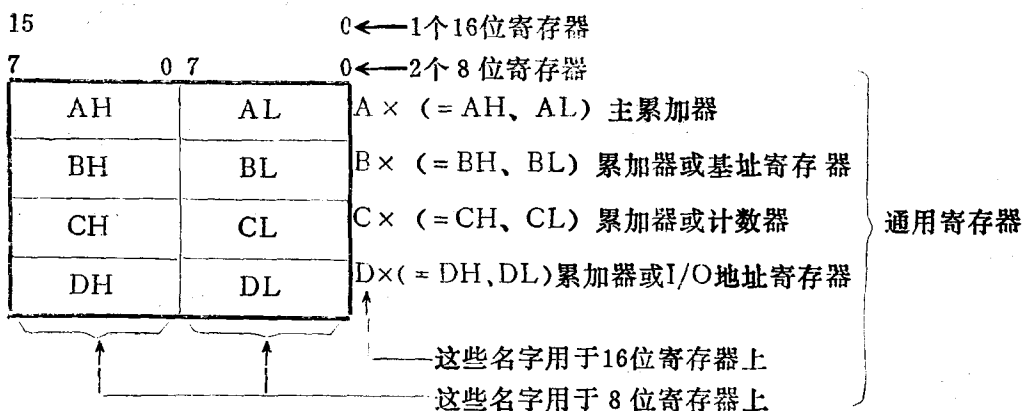
第一章 引言	(135)
第二章 宏汇编	(136)

2.1 命令格式.....	(136)
2.2 不指明的文件扩展名.....	(136)
2.3 怎样开始使用汇编程序.....	(137)
2.3.1 选择方式 1.....	(137)
2.3.2 选择方式 2.....	(138)
2.3.3 选择方式 3.....	(139)
2.3.4 宏汇编程序结束.....	(140)
2.4 参数 (/params).....	(140)
2.5 设备标识符.....	(140)
第三章 交叉引用程序CREF.....	(140)
3.1 产生一个交叉引用文件.....	(141)
3.2 怎样开始CREF.....	(141)
3.2.1 CREF选择方式 1.....	(141)
3.2.2 CREF选择方式 2.....	(142)
3.2.3 CREF选择方式 3.....	(142)
3.2.4 CREF结束.....	(142)
3.2.5 CRF格式.....	(142)
第四章 汇编语言格式.....	(143)
4.1 符号/数据的表示.....	(144)
4.2 数据项.....	(144)
4.3 常数(立即数).....	(144)
4.4 第一遍扫描和第二遍扫描.....	(145)
4.5 标号.....	(146)
4.6 变量.....	(146)
4.7 操作数.....	(147)
4.8 属性操作.....	(148)
第五章 伪操作.....	(152)
5.1 数据伪操作.....	(153)
5.2 条件伪操作.....	(167)
5.3 宏伪操作.....	(168)
5.4 列表伪操作.....	(176)
5.5 假条件块.....	(178)
5.6 程序员控制.....	(178)
附录: 信息.....	(180)

第一章 8086寄存器和标志

8086有4个16位通用寄存器，2个16位指针寄存器，2个16位变址寄存器，1个16位程序计数器，4个16位段寄存器，1个16位标志寄存器。一共有14个16位寄存器。

图例说明如下：



1.1 通用寄存器

通用寄存器用来存放8位或16位算术/逻辑操作数。即高8位和低8位可以分开寻址。

Ax寄存器为主累加器。I/O操作要通过这个寄存器。另外,一些字符串操作和算术指令也需要使用这个寄存器。

Bx寄存器。在计算数据存储器地址时使用它作为基址寄存器,并使用DS寄存器作为未指明的段寄存器。

Cx寄存器称为计数寄存器。在字符串和循环操作时,用它来控制重复循环操作次数。它也用在多位移位和循环移位操作上。

Dx寄存器也称为数据寄存器。这个寄存器对某些I/O指令提供了I/O地址。

1.2 指针寄存器

指针寄存器用于访问堆栈段中的数据。在所有的16位算术、逻辑操作中,他们也可作为操作数使用。

SP寄存器称为堆栈指针。它指出当前堆栈栈顶的地址的偏移量,在存储器寻址中,凡涉及到SP寄存器则使用SS寄存器作为段寄存器。

BP寄存器称为基址指针。这个寄存器主要用在访问堆栈段中的数据。

1.3 变址寄存器

变址寄存器用于访问数据存储器中的数据。这些寄存器广泛地用在字符串操作上。他们也可以在所有的16位算术、逻辑操作中作为操作数使用。

1.4 段寄存器

段寄存器包括整个存储器寻址计算。每个段寄存器在8086存储器寻址空间中规定为64K的存储器块。该64K存储器块叫做段寄存器的当前段。例如,DS寄存器规定的64K段称作当前数据段。

CS寄存器称作代码段寄存器。在每条指令的取指期间,程序计数器的内容加上CS寄存器的内容作为被取的指令所在的存储器地址。

DS寄存器也称作数据段寄存器。对数据存储器的访问都是相对于数据段寄存器。有三个例外:

- 1) 利用堆栈指针计算堆栈地址。
- 2) 使用BP寄存器计算数据存储器地址,均是相对于堆栈段。
- 3) 字符串操作(在地址计算中使用DI寄存器)均是相对于附加段。

SS寄存器也叫做堆栈段寄存器。在地址计算中使用SP或BP访问所有的数据存储器均是相对于SS寄存器。因此,所有面向堆栈的指令(例如,PUSH, POP, CALL, RET, INT)均利用SS寄存器作为段寄存器。

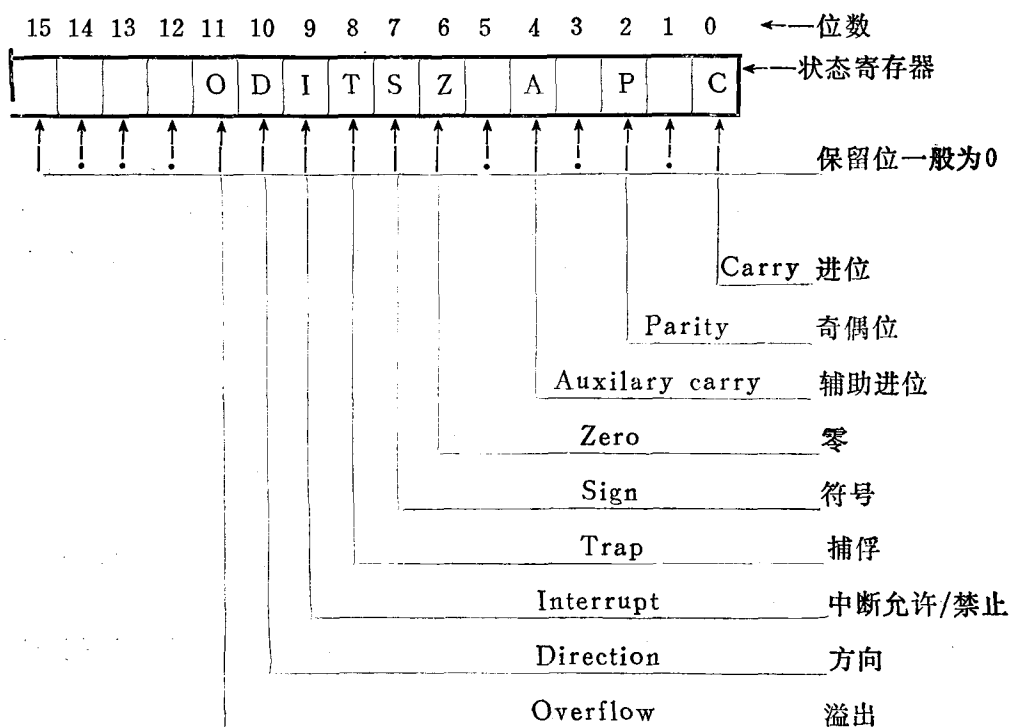
ES寄存器也叫做附加段寄存器。在字符串操作时,利用DI寄存器计算存储器地址,均是相对于ES寄存器。

段寄存器的使用是由指令支配的。

1.5 标志寄存器

8086有一个16位的标志寄存器,也叫做状态寄存器或者程序状态字(PSW)。

例如下：



Carry状态反映了算术操作中高位进位情况。Carry也可由移位和循环移位指令修改。为了从被减数中减去减数，减法指令使用了2的补码运算。但是进位状态是相反的。那就是说，在减法操作中，如果高位无进位输出，Carry状态置“1”。反之，若高位有进位输出，Carry状态置“0”。因此，这里Carry状态表示的是借位。

Overflow状态是在算术操作之后，高位的输出和进位的异或结果。表示带符号的二进制运算在数量上的溢出。

Sign状态就是算术操作之后的最高位。假定执行的是带符号的二进制运算。Sign状态为0，表示结果为正。Sign状态为1，表示结果为负。

Auxiliary Carry状态表示在低8位数据单元中位3的进位。

当任一数据操作结果的低8位中有偶数个1位，则Parity状态置“1”。若有奇数个1位，则Parity状态置“0”。

当数据的操作结果是零，则Zero状态位置“1”。否则置“0”。

Direction状态决定在字符串操作时，变址寄存器中的内容是自动地增加，还是自动地减少。如果Direction状态是1，SI和DI寄存器的内容就要减少。就是说字符串将从存储器最高地址开始向存储器最低地址进行存取。如果Direction状态是0，SI和DI寄存器的内容就要增加。就是说，字符串将从存储器最低地址开始存取。

Interrupt状态表示允许总的中断还是禁止总的中断。若要允许中断，则此位必须是1。若要禁止中断此位必须是0。

Trap状态是特殊的调试手段。它使8086进入到“单步”方式。单步方式在8086中断逻辑中进行详细介绍。

第二章 8086寻址方式

2.1 8086存储器地址的计算

20位地址的形成

8086有20条地址总线，它是采用字节或字访问存储器和I/O设备。因此可以直接寻址 2^{20} 个字节（即1兆或1048576字节），每个地址内容为1个字节。为了能够利用16位地址寻址到1兆的空间，因此采用了用段寄存器寻址和偏移量寻址相结合的办法，来构成实际的地址。

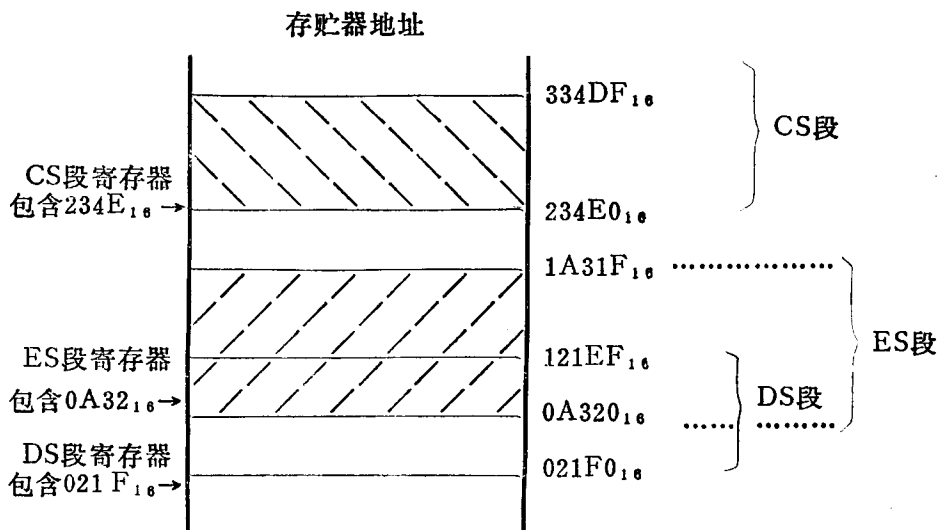
其计算方法是

段寄存器内容	x x x x x x x x x x x x x x x x 0 0 0 0
有效存储器地址	+ 0 0 0 0 y y y y y y y y y y y y y y y y
实际地址输出	x x x x z z z z z z z z z z z z z z y y y y

其中x, y, z表示任一二进制数字。

由计算方法上可以看出将段寄存器的内容也称作段地址，左移4位与有效的存储器地址也称作偏移地址相加即得到实际地址输出。

段寄存器内容和偏移地址组成的实际地址可以指向存储器的任何一个单元。如下图所示：



因为段寄存器为16位字长，所以段地址最大容量为65,536个字节。每个段必须指出段的初值地址。

2.2 8086寻址方式

8086寻址方式可分为两种类型：

- (1) 程序存储器寻址方式
- (2) 数据存储器寻址方式

数据存储器分为相对于DS段寄存器的数据存储器，和相对于SS段寄存器的数据存储器两种类型

下面分别介绍这两种类型的寻址方式。

(1) 程序存储器寻址方式

每当取指令时，要取的指令地址总是由程序计数器（PC）给出的偏移量和CS寄存器中给出的段值相加之和决定。通常，在执行指令时，PC寄存器的内容是增加的。但是，Jump和Call指令可以修改PC寄存器的内容。修改方法有以下三种：

a、程序相对寻址。将指令中以立即数的形式提供8位或16位的偏移量，作为一个带符号的二进制数与PC寄存器的内容相加，产生一个新的PC，但这不会改变CS寄存器的内容。因此这叫段内操作。

b、直接寻址。将指令中以立即数形式出现的新的16位地址，装入到程序计数器PC和CS寄存器中，这叫做段间操作。

c、间接寻址。可使用任一种数据存储器寻址方式来读数据存储器中的数据。但是，输入的数据在Jump或Call指令中作为存储器地址。有两种间接寻址方式，一种可读单个的16位数据字，这个数据字被装入到程序计数器中。Jump和Call指令访问当前的CS段内某个存储器单元。另一种可读两个16位数据字，第一个字装入到程序计数器中，第二个字装入到CS段寄存器中。因此，可使用间接寻址方式Jump或Call任一可寻址的存储器单元。

(2) 数据存储器寻址方式

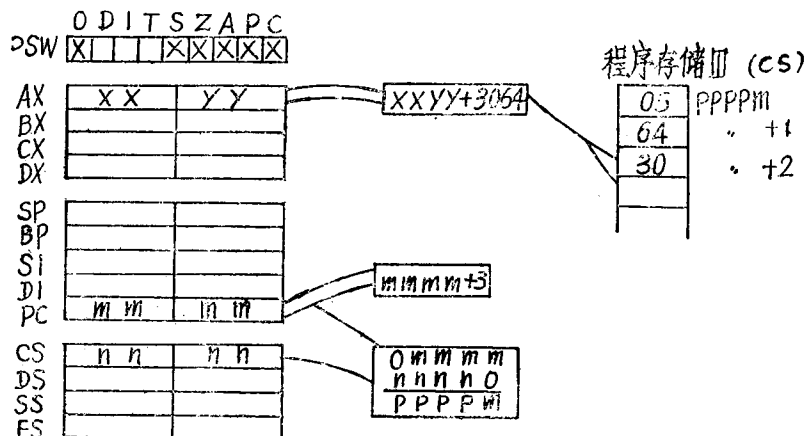
有六种基本寻址方式：立即寻址方式，直接存储器寻址方式，直接，变址存储器寻址方式，隐含存储器寻址方式，基本相对寻址方式，堆栈存储器寻址方式。

a、立即寻址方式 (Immediate)

在这种寻址方式中，立即数跟在寻址字节后面。

例如：ADD AX, 3064H

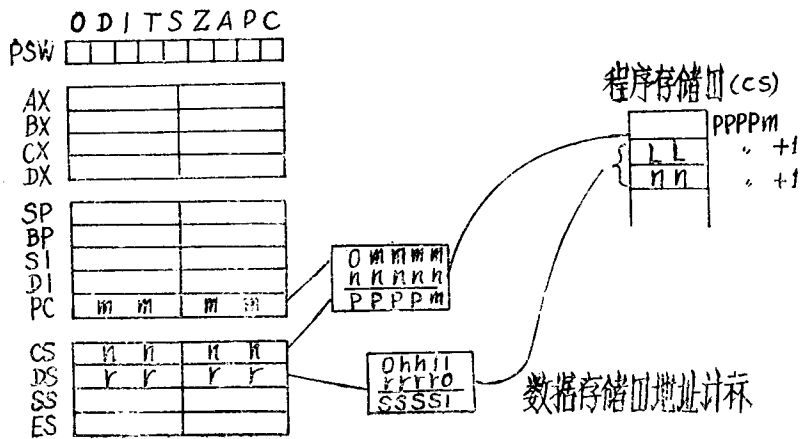
由汇编程序产生ADD指令，并将3064H加到AX寄存器中，可用下面的图例说明：



注意：在程序存储器中的16位立即操作数的低位字节在高位字节的前面。同样，16位操作数存入到数据存储器也是按这个顺序。

b、直接存储器寻址方式 (Direct)

8086通过直接由两个目标码字节组成的16位偏移量加到数据段寄存器中，以实现直接存储器寻址方式。其和就是实际的存储器地址。可用下面的图例说明：

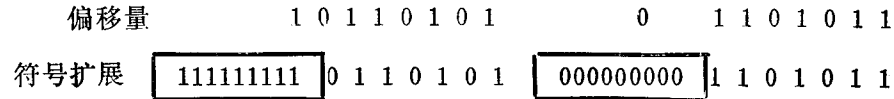


注：在数据存储器进行直接寻址时，DS寄存器必须提供段基址。

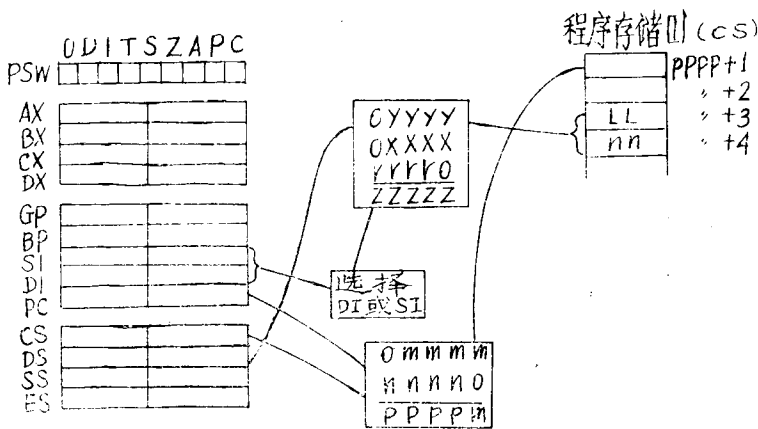
c、直接，变址存储器寻址 (Direct, Indexed)

在直接，变址寻址方式中，SI和DI寄存器作为变址寄存器。可选择8位或16位的偏移量加到指定的变址寄存器中，以产生有效的地址。

用两个目标码字节存储16位的偏移量，低位字节在前，高位字节在后。如图所示。如果指定的是8位偏移量，那么低位字节中的高位扩展到高位字节中去，以产生16位偏移量。举例如下：



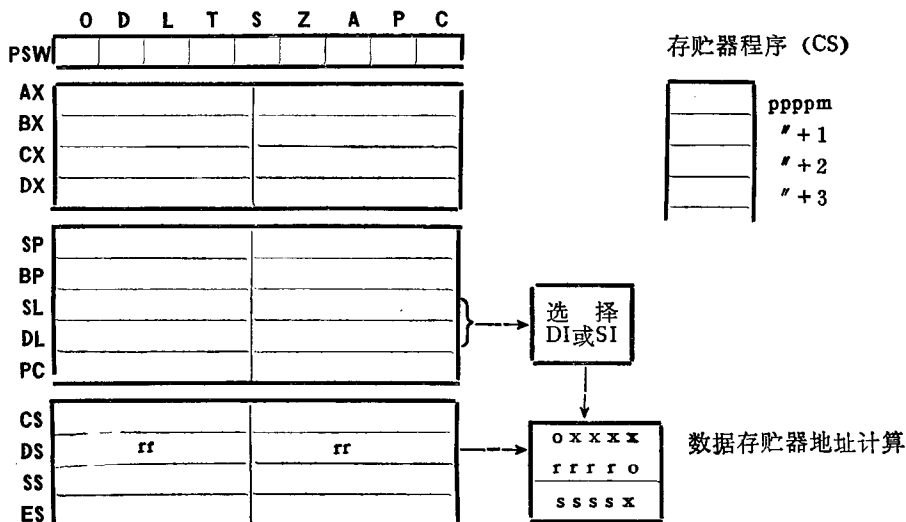
直接，变址寻址方式图例如下：



注：YYYY是从程序存储器中取来的16位或8位偏移量。XXXX是变址寄存器SI或DI中的内容。

d、隐含寻址 (Implied)

8086中隐含寻址是直接，变址存储器寻址的简并类型。当使用直接，变址存储器寻址方式时，如果不指明偏移量，那么实际上使用的就是通过DI或SI寄存器的隐含寻址方式。用图例表示：

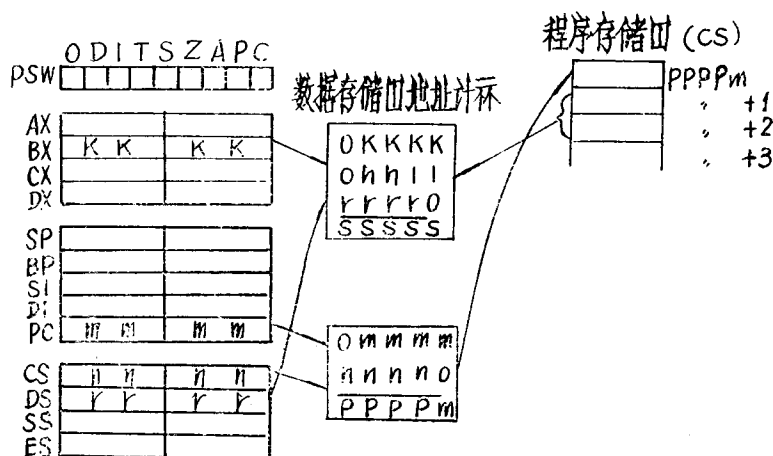


e、基本相对寻址 (Base Relative)

8086有两种方法执行基本相对寻址：

- (1) 数据存储器基本相对寻址，是在DS段内（数据存储器）寻址。
- (2) 堆栈基本相对寻址，是在SS段内（堆栈存储器）寻址。

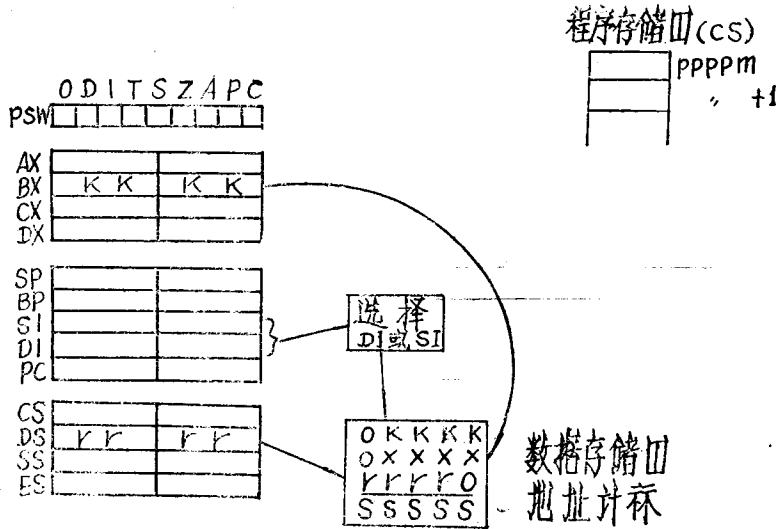
数据存储器基本相对寻址方式使用BX寄存器内容作为有效地址的基址。实际上，数据存储器基本相对寻址，就是将BX寄存器的内容加上有效的存储器地址。举例说明：



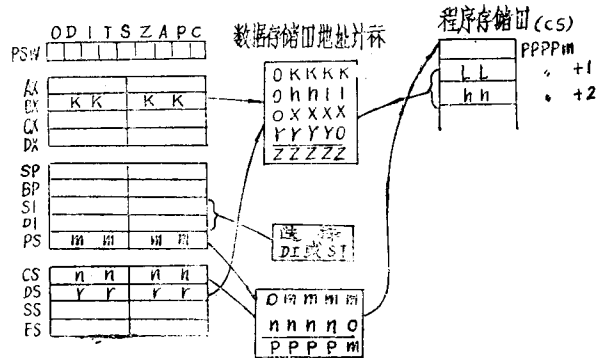
前面已介绍了简单的直接寻址总是要产生一个16位偏移量。基本相对，直接寻址允许有偏移量。如上图中的hhll，是16位偏移量。8位偏移量可以扩展成16位的。或者无偏移量。

基本相对隐含寻址就是BX寄存器的内容加上选择的变址寄存器的内容，其和作为有

效的存储器地址。图示如下：

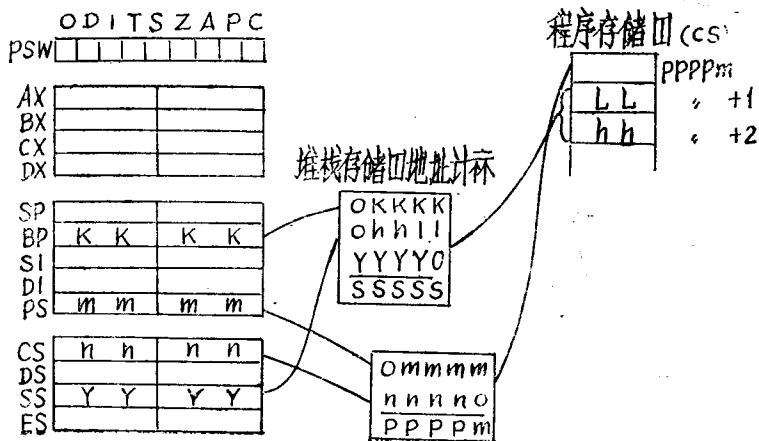


基本相对，直接，变址数据存储器寻址看来变得复杂了，实际上并不是那样。只要将 BX 寄存器的内容加上有效的存储器地址，而有效的存储器地址按一般的直接，变址寻址方式那样计算地址。这样基本相对，直接，变址数据存储器寻址可用下面图示表示：



f、堆栈存储器寻址 (Stack)

8086也具有堆栈存储器的各种各样的基本相对寻址方式。在这里BP寄存器作为基址寄存器。如下图：

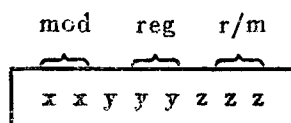


在上面的图中，偏移量HLLL或是16位偏移量，或是带有符号扩展的8位偏移量。基本相对堆栈存储器寻址需要指定的偏移量，即或是0

2.3 寻址方式字节

8086提供了广泛的寻址方式，可是在目标码中是怎样实现这些寻址方式？在8086指令的目标码中，用一个字节规定了大多数存储器的寻址方式。这个字节叫做寻址方式字节。与寻址方式字节一起可以有一个或二个附加的偏移量字节。通常寻址方式字节是指令目标码中的第2个字节。

寻址方式字节图例如下

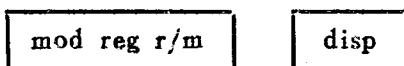


mod: 这是两位，它构成mod字段。mod字段用来区分是存储器寻址还是寄存器寻址。在存储器寻址情况下，又规定在寻址方式字节的后面有多少偏移量字节。

mod =

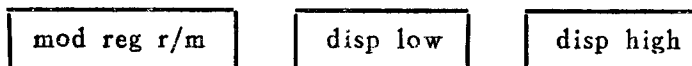
00: 寄存器寻址方式。由r/m指定确切的寻址方式。无偏移量字节。

01: 存储器寻址方式。由r/m指定确切的寻址方式。有一个偏移量字节。这个偏移量字节是一个带符号的数，它的范围是+127到-128。当这个数用在计算存储器地址时，它就由符号位扩展成16位。在这种情况下寻址方式字节可由下面的图示说明：



此处mod = 01, disp (偏移量) 为1个字节

10: 存储器寻址方式。由r/m指定确切的寻址方式。有二个偏移量字节。第一个偏移量字节是偏移量的低8位，第二个偏移量字节是偏移量的高8位。当这个数用来计算存储器地址时，就处理成无符号的16位数。在这种情况下，寻址方式字节可由下面的图示说明：



此处mod = 10, disp low是偏移量的低8位，disp high是偏移量的高8位。

11: 寄存器寻址方式。r/m表示寄存器。与w连在一起使用来确定选择8位还是16位寄存器。

reg: reg与w位一起使用来选择在本操作中使用哪个寄存器。w位是指令操作码的一部分，它决定执行8位操作还是执行16位操作。

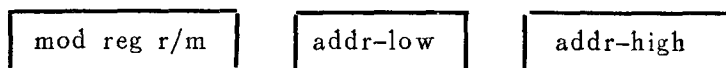
reg	w = 0	w = 1
0 0 0	AL	AX
0 0 1	CL	CX
0 1 0	DL	DX
0 1 1	BL	BX
1 0 0	AH	SP
1 0 1	CH	BP
1 1 0	DH	SI
1 1 1	BH	DI

某些指令——这些指令只需要单个的寄存器或存储器操作数（NOT，NEG等等），还有一个隐含的操作数（MOV立即数，DIV，MUL等等），或者利用寻址方式来计算目标地址（JMP，CALL等等）——利用reg字段扩展操作码字节以便指定所要求的指令。见ADC指令的介绍。

r/m：这是三位，它构成了r/m字段。它与mod字段联合使用来确定寻址方式。如下表所述

r/m	mod-00	mod-01	mod-10	mod-11	
				W = 0	W = 1
0 0 0	BX + SI	BX + SI + DISP	BX + SI + DISP	AL	AX
0 0 1	BX + DI	BX + DI + DISP	BX + DI + DISP	CL	CX
0 1 0	BP + SI	BP + SI + DISP	BP + SI + DISP	DL	DX
0 1 1	BP + DI	BP + DI + DISP	BP + DI + DISP	BL	BX
1 0 0	SI	SI + DISP	SI + DISP	AH	SP
1 0 1	DI	DI + DISP	DI + DISP	CH	BP
1 1 0	直接寻址	BP + DISP	BP + DISP	DH	SI
1 1 1	BX	BX + DISP	BX + DISP	BH	DI

这个表除了直接寻址之外，是解释性的表。当mod是00，r/m是110时，偏移地址就是寻址方式字节后面的两个字节，用图例说明：



在这里addr-low就是偏移地址的低8位，addr-high就是偏移地址的高8位。

2.4 段替换

有关段替换的内容请参考“SEG”指令介绍的未指明的段寄存器就由前缀字节中所指定的段寄存器来替换。前缀字节如下：

0	0	1	r	r	1	1	0
---	---	---	---	---	---	---	---

rr: 这是二位，它选择下面的指令要用到的段寄存器。

00为ES寄存器

01为CS寄存器

10为SS寄存器

11为DS寄存器

有三种情况不能使用段替换，它们是：

1. 堆栈访问指令（如PUSH和CALL）它用堆栈指针（SP寄存器）去计算偏移量，总是使用SS寄存器作为段寄存器。

2. 字符串指令使用DI寄存器总是使用ES寄存器作为段寄存器。在使用SI和DI寄存器的字符串操作中（例MOVS或CMPS），如果段替换前缀出现，则只替换SI偏移的段寄存器。

3. 段替换前缀不能和程序存储器寻址一起使用。所有的指令取出都是相对CS段寄存器进行的。

2.5 存储器寻址方式表

存储器寻址方式和存储器寻址字节组合在一起列在下面表中

rm/	mod = 00	mod = 01	mod = 10
000	基本相对变址 BX + SI	基本相对直接变址 BX + SI + DISP	基本相对直接变址 BX + SI + DISP
001	基本相对变址 BX + DI	基本相对直接变址 BX + DI + DISP	基本相对直接变址 BX + DI + DISP
010	基本相对变址堆栈 BP + SI	基本相对直接变址堆栈 BP + SI + DISP	基本相对直接变址堆栈 BP + SI + DISP
011	基本相对变址堆栈 BP + DI	基本相对直接变址堆栈 BP + DI + DISP	基本相对直接变址堆栈 BP + DI + DISP
100	隐含寻址 SI	直接，变址 SI + DISP	直接，变址 SI + DISP
101	隐含寻址 DI	直接，变址 DI + DISP	直接，变址 DI + DISP
110	直接地址	基本相对直接堆栈 BP + DISP	基本相对直接堆栈 BP + DISP
111	基本相对 BX	基本相对直接 BX + DISP	基本相对直接 BX + DISP

注：两个操作数的指令中，当一个操作数在CPU寄存器中时，指令将很有顺序地取出存储器中的另一个操作数。两个操作数也可以顺序地从CPU寄存器中取出。除若干个特殊的数据串处理指令外，8086不允许从存储器中取出两个操作数。

可以做如下选择

源操作数	目标操作数	结果
CPU寄存器	CPU寄存器	CPU寄存器
存储器单元	CPU寄存器	CPU寄存器
CPU寄存器	存储器单元	存储器单元

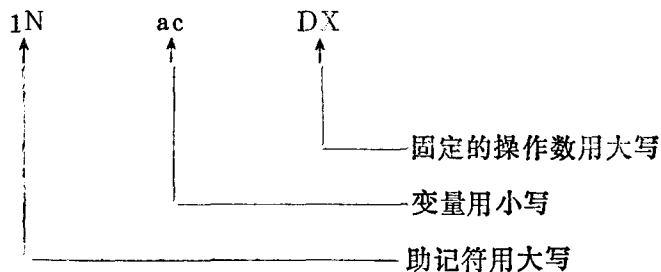
第三章 指令系统助记符

在讨论8086每条汇编语言指令之前，先将指令的构成介绍一下。

每条指令由6个部分构成：

1、指令的助记符和各种各样的操作数在一起。可变的操作数是用小写字母表示。助记符本身和任一固定的操作数是用大写字母表示。

例如：



2、指令的操作说明

3、指令的机器语言编码

4、指令操作例子。对于简单的指令就不举例说明了。

5、指令执行图，此图说明执行本指令时，对标志位、寄存器和存储器的影响。

6、注释部分中包括如何使用指令的一些短小例子，或在实际情况中的有关注意事项。

缩写：

下面是与助记符一起介绍操作数时使用的缩写字：

ac 如果指的是8位操作则就是AL寄存器。如果指的是16位操作则指的是AX寄存器。在汇编语言指令中用AL或AX寄存器表示。

addr 8086的地址是由两个16位地址组成的。即16位偏移地址和16位段地址。在汇编语言指令中用标号表示。

count 它是1或CL寄存器中的内容。在汇编语言指令中用1或CL表示。

data 8位或16位立即数。在汇编语言语句中它代表数据或表达式。

disp 8位带符号的偏移量用在Jump指令中。

disp16 16位二进制偏移量，用在Call, Jump和Return指令中。当用在Jump和Call指令时，用标号表示。在返回指令中，使用数字表达式表示16位偏移量。使用方法将在Return指令中说明。

mem 存储器操作数。用寻址方式来选择由寻址方式字节指定的操作数。典型是用标号表示，在这种情况下，汇编将选择合适的寻址方式字节，或按符号顺序允许选择特殊的寻址方式字节。在指令中由mod r/m表示。

mem/reg 存储器或寄存器操作数。查阅mem和reg的说明。

port I/O端口。它将由数据或表达式表示。端口数必须在 0_{16} 和 FF_{16} 之间。

reg 如果指定为8位操作,就是AH, AL, BH, BL, CH, CL, DH, DL寄存器。
如果指定为16位操作,就是AX, BX, CX, DX, SP, BP, SI, DI寄存器。

segreg段寄存器CS, DS, ES或SS。

在指令代码的介绍中使用了下面的缩写字母:

c 它是1位。用在移位和循环移位指令中。在执行移位和循环移位指令时,选择1或者CL寄存器中的内容作为移位或循环移位的次数。

C = 0 移位/循环 移位1次

C = 1 移位/循环移位CL寄存器指定的次数。

d 它是1位。用来表示方向,按这个方向执行操作。

disp 这是8位,在Jump指令中作为带符号的二进制偏移量。

jj 两位十六进制数字。用于表示立即数或16位偏移量中的一部分。

kk 两位十六进制数字。用于表示立即数或16位偏移量中的一部分。

mod reg r/m 8位寻址方式字节。

rrr 3位。选择8086通用寄存器之中的一个。

8位操作时	16位操作时
rrr = 000 AL	rrr = 000 AX
001 CL	001 CX
010 DL	010 DX
011 BL	011 BX
100 AH	100 SP
101 CH	101 BP
110 DH	110 SI
111 BH	111 DI

s 1位。符号扩展值。表示是否执行符号扩展的立即数。如果用立即数指定16位操作数,立即数由程序存储器中的一个字节表示。

S解释如下:

S = 0出现两个字节的立即数。

s = 1出现一个字节的立即数,而立即数的高8位是由低8位的符号位扩展形成。

ss 两位。选择8086段寄存器。

ss = 00 是ES

01 是CS

10 是SS

11 是DS

v 1位。表示软件中断时要进入的单元。如果 $v = 0$,那么中断服务子程序安放在 $0000C_{16}$ 单元指定的地址上。 $v = 1$ 中断服务子程序起始地址由下一个字节决定。

w 1位。指出执行的是8位还是16位操作。