第三章 寻址方式与指令系统

本章主要内容：

- 8086/8088的各种寻址方式
- 8086/8088的传送类指令
- 8086/8088的基本算术类指令
- 8086/8088移位指令
- 8086/8088逻辑指令
- 8086/8088处理器控制类指令
- 8086/8088指令编码

3.1 寻址方式

一条指令通常由两大部分构成：

操作码	操作数

操作码：表示该指令应完成的具体操作，如加法、减法、乘法、移位等等。在汇编语言中使用一定的符号来表示，称为助记符。如ADD 、PUSH、POP、MOV等等。

操作数：表示该指令的操作对象。如移位操作的被移位数，加法操作的加数等等。它可以是一个操作数，也可以是多个操作数。这取决于操作码部分的具体需要。

寻址方式:寻找指令中所需操作数的各种方法，也就是提供指令中操作数的存放信息的方式。

Intel 8086/8088 各指令中提供操作数的方法有以下四种：

（1）立即数操作数——操作数在指令代码中提供

（2）寄存器操作数——操作数在CPU的通用寄存器或段寄存器中

（3）存储器操作数——操作数在内存的存储单元中

（4）I/O端口操作数——操作数在输入/输出接口的寄存器中

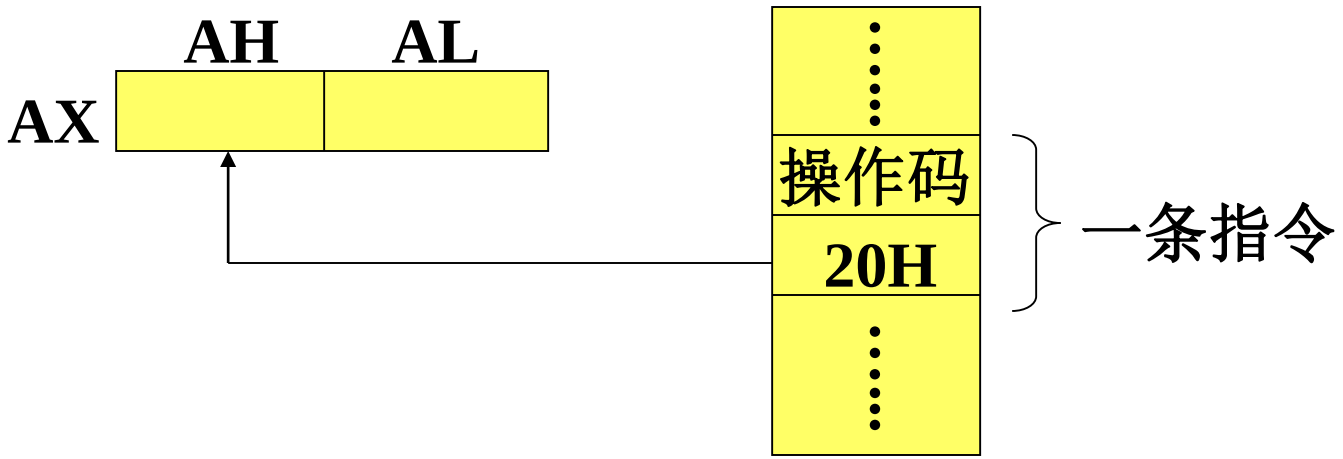
1.立即数寻址

立即数寻址方式的指令中，所需操作数直接包含在指令代码中，这种操作数称为立即数。

立即数可以是8位，也可以是16位。

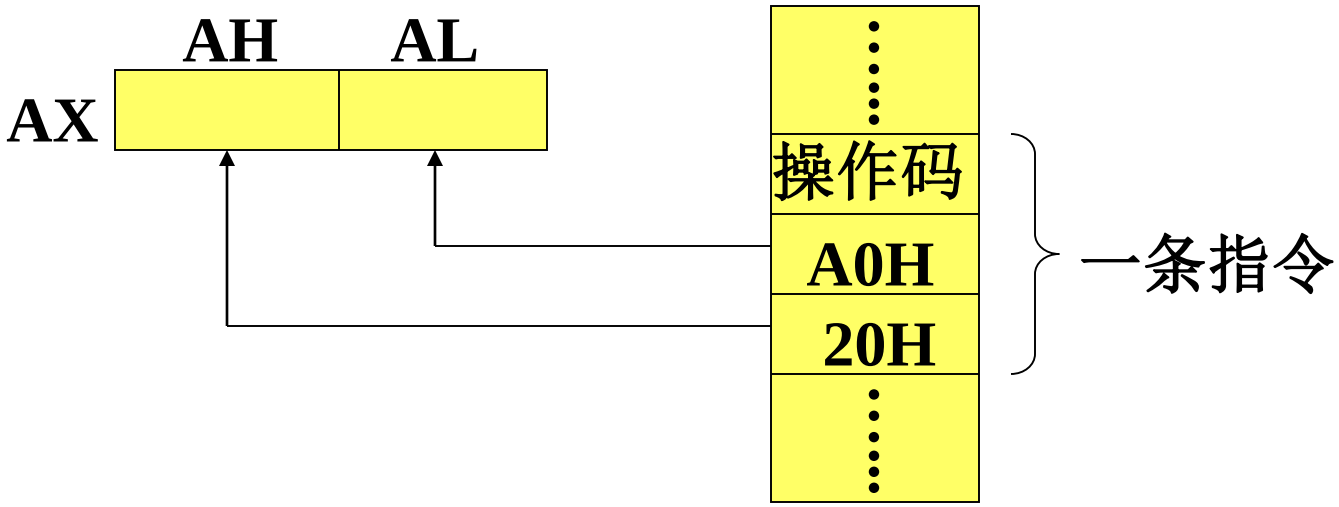
例：MOV AH，20H

它表示将8位立即数20H送入AH中



例：MOV AX，20A0H

它表示将16位立即数20A0H送入AX中



由于在指令执行过程中，立即数作为指令的一部分直接从BIU的指令队列中取出，它不需另外占用总线周期，因此这种寻址方式执行速度快。

注意：立即数只能作为源操作数，而不能作为目的操作数。

2.寄存器寻址

寄存器寻址方式是指指令中所需的操作数在CPU的某个寄存器中。寄存器可以是8位或16位通用寄存器，或者是段寄存器。如：AH、AL、AX、CX、DS、ES等。

例如：MOV AX，BX
MOV DS，AX

由于存取寄存器操作数完全在CPU内部进行，不需要总线周期，所以执行速度很快。

后面介绍的几种寻址方式其操作数都是在存储器中，它们的主要区别就是操作数在内存中存放地址的形成方法不同。

一个存储单元逻辑地址表示形式：段基值：偏移量

段基值由某个段寄存器提供.

偏移量表示了该存储单元与段起始地址之间的距离，也叫做有效地址EA。

有效地址EA是以下三个地址分量的几种组合，由CPU的执行单元EU计算出来的。

(1)位移量：位移量是指令中直接给出的一个8位或16位数。一般源程序中以操作数名字(变量名或标号)的形式出现。

(2)基址：由基址寄存器BX或基址指针BP提供的内容。

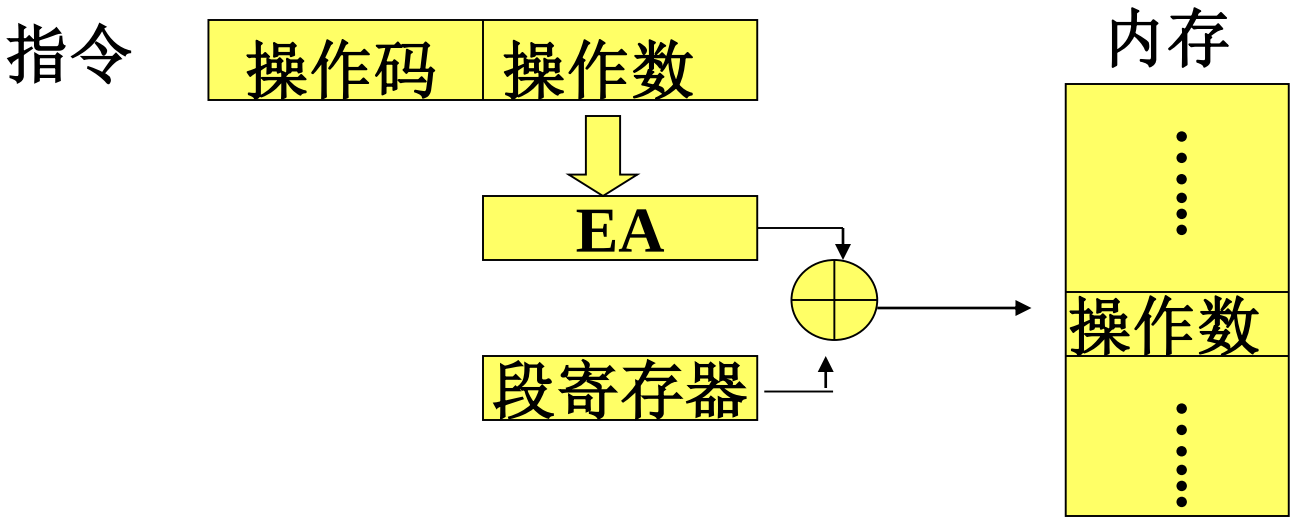
(3)变址：由源变址寄存器SI或目的变址寄存器DI提供的内容

位移量、基址和变址三个地址分量组合时，若有两个或两个以上分量时，将进行以2^{16}为模的十六位加法运算。

下面是由这三个地址分量的不同组合所形成的四种寻址方式。

3.直接寻址

在直接寻址方式的指令中，操作数的有效地址EA只有位移量地址分量。



在汇编语言源程序中，直接寻址方式用符号或常数来表示。

（1）用符号表示

例：MOV BX，VAR =>MOV BX，DS：VAR

它表示将数据段中，偏移了VAR个字节距离的字单元内容送到寄存器BX中。

MOV AL，DATA+2 => MOV AL, DS: DATA+2

它表示将数据段偏移了DATA+2的字节单元内容送入AL中。

（2）用常数表示

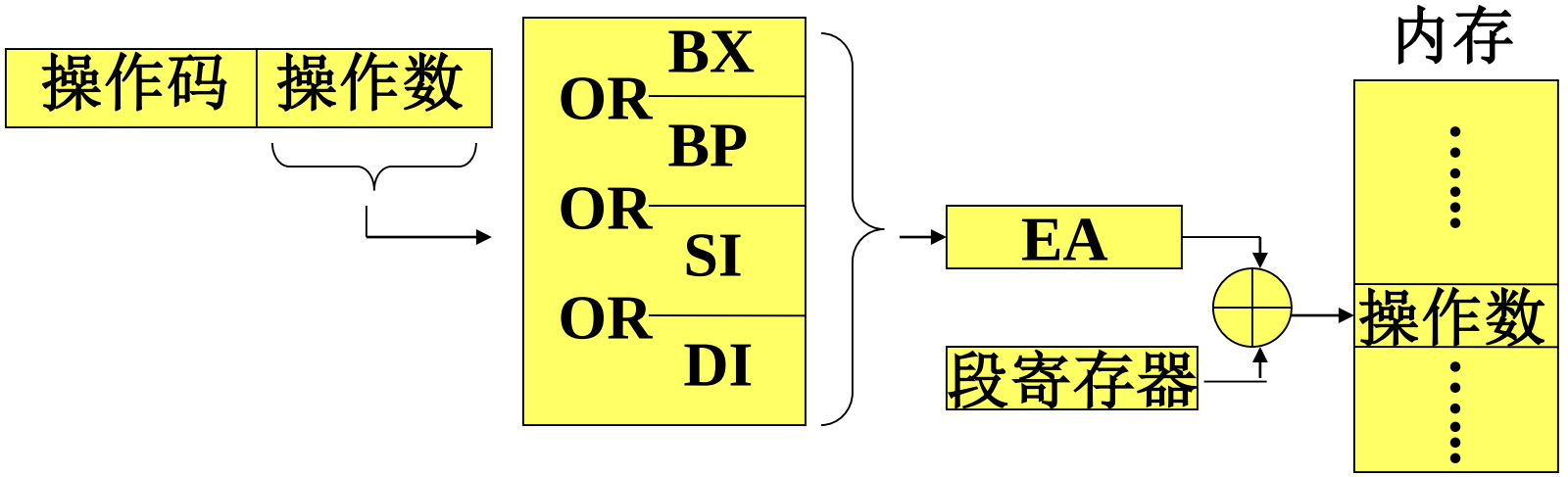
例：MOV AX，DS：[64H]

它表示从当前数据段开始，偏移100个字节的字单元内容送到AX中。不能写为：MOV AX，64H

注意：用常数表示时，必须用方括号括起来。段寄存器不能省略。

4.寄存器间接寻址

操作数有效地址EA直接从基址寄存器（BX或BP）或变址寄存器（SI或DI）中获得。



寄存器间接寻址就是事先将偏移量存放在某个寄存器(BX、BP、SI或DI)中，这些寄存器就如同一个地址指针。

在程序运行期间，只要对寄存器内容进行修改，就可以实现用同一条指令实现对不同存储单元进行操作。

指示存储器所在段的段寄存器可以省略，当指令中使用的是**BP寄存器**，则隐含表示使用**SS段寄存器**，其余情况则隐含使用**DS段寄存器**。

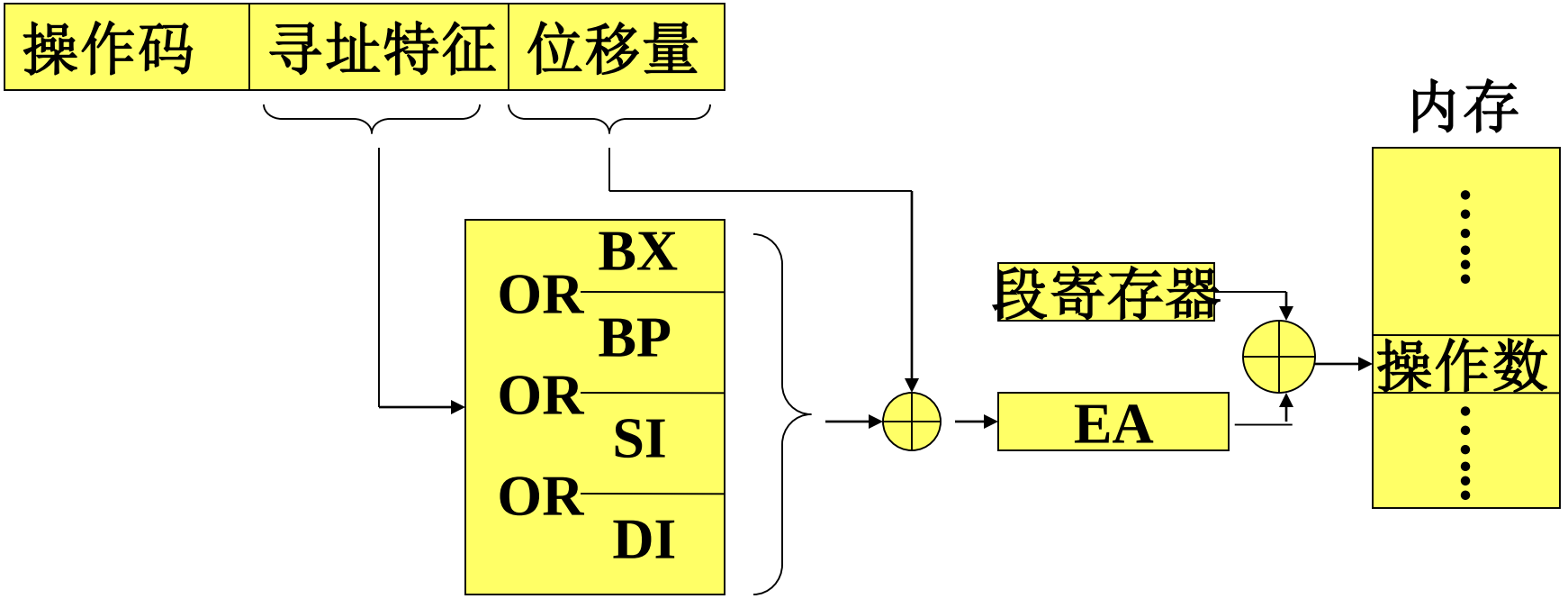
例如：MOV AX，[BX] => MOV AX，DS：[BX]

MOV BH，[BP] => MOV BH，SS：[BP]

MOV [DI]，BX => MOV DS：[DI]，BX

5.基址寻址/变址寻址

操作数的有效地址EA等于基址分量或变址分量加上指令中给出的位移量。



指令中使用BX或BP时为基址寻址。指令中使用SI或DI时为变址寻址。

段寄存器的隐含使用规则与寄存器间接寻址方式相同

例：MOV AX，10H [SI] => MOV AX，DS：10H [SI]

MOV TABLE [DI]，AL =>MOV DS：TABLE [DI]，AL

注意：当位移量为常数时，不能加方括号。

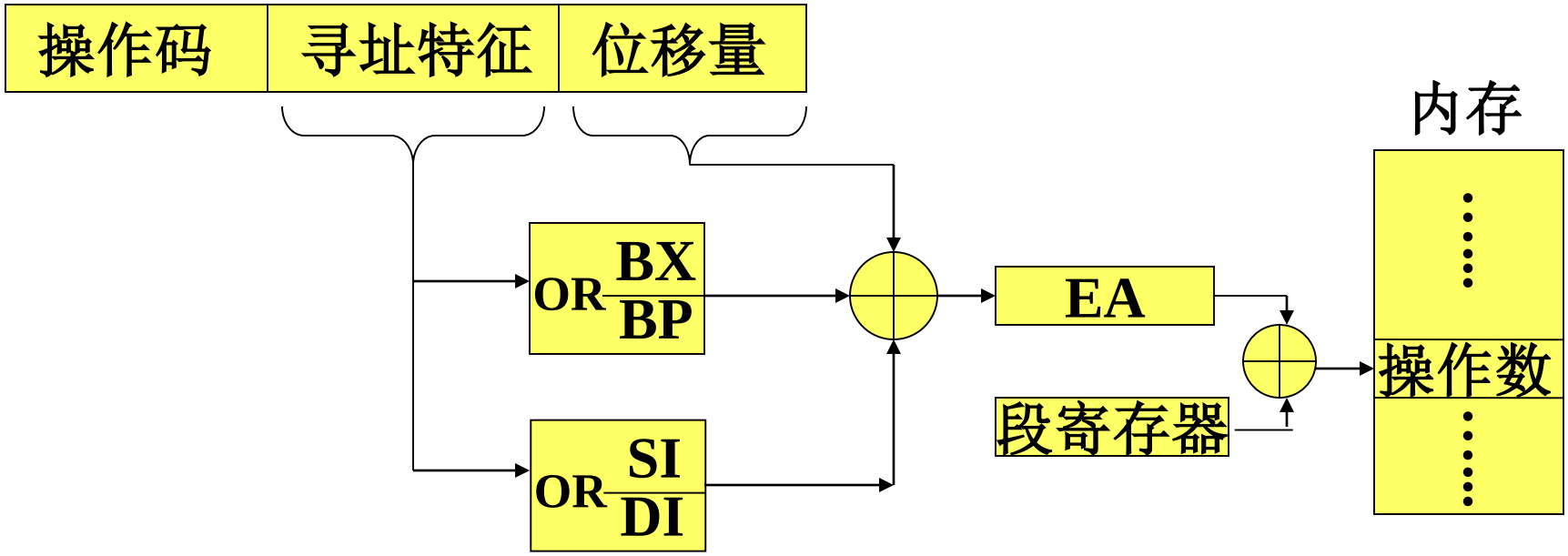
这两种寻址方式只需通过改变寄存器的内容就可用一条指令访问不同的存储单元，并且由于增加了一个位移量分量，因此它们能够很方便地访问数组和表格数据。

由于这两种寻址方式中寄存器中的内容是相对于由位移量指定的初始单元。因此也叫寄存器相对寻址。

6.基址变址寻址

操作数的有效地址是三个地址分量之和，即：

EA=基址+变址+位移量



当基址选用BX时隐含使用段寄存器DS，而选用BP时则隐含使用段寄存器SS。

例如：MOV CX，100H[BX][DI]

MOV TABLE[BX][DI]，AX

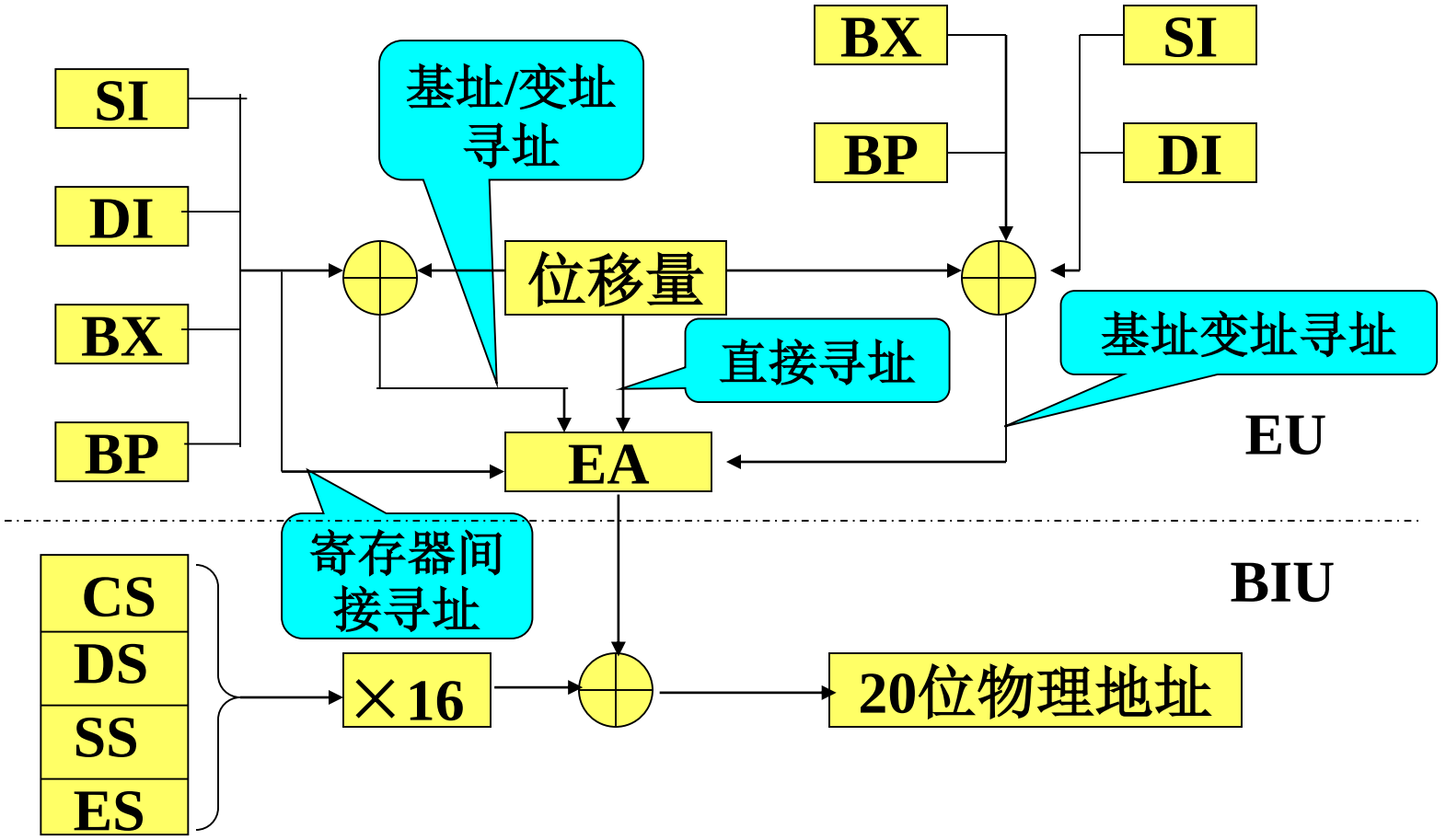
下面的用法是错误的。

MOV AX，ARRAY[BX][BP]

MOV AX，TABLE[SI][DI]

在基址变址寻址方式中，程序运行期间有两个地址分量可以修改。因此它是最灵活的一种寻址方式，可以方便地对二维数组进行访问。

存储器操作数寻址方式中地址形成小结



7.串操作寻址方式

8086/8088设置有专门用于串操作的指令，这些指令的操作数虽然也在存储器中，但它们不使用前面介绍的各种寻址方式，而隐含地使用变址寄存器**SI**和**DI**专门指示。

- 在寻找源操作数时，隐含使用**SI**作为地址指针。
- 在寻找目的串时，隐含使用**DI**作为地址指针。
- 在串操作完成之后，自动对**SI**和**DI**进行修改，使它们指向下一个操作数。

8. I/O端口寻址

在计算机系统，对I/O端口的寻址方式有以下两种方法。

➢ 存储器编址方法

将I/O端口视为存储器的一个单元，对端口的访问就如同访问存储单元一样。访问存储器的指令和各种寻址方式同样适用对I/O端口的访问。

特点: 程序设计灵活，但需要占用存储地址空间。

➢ I/O端口编址方法

I/O端口的地址与存储器地址分开，并使用专门的输入指令和输出指令。

8086/8088系统中就是采用的这种方式。可以最多访问64K个字节端口或32K个字端口，用专门的IN指令和OUT指令访问。寻址方式有如下两种。

（1） 直接端口寻址

在指令中直接给出端口地址，端口地址一般采用2位十六进制数，也可以用符号表示。

直接端口寻址可访问的端口数为0～255个。

例如：IN AL，25H

（2）寄存器间接端口寻址

寄存器间接端口寻址：把I/O端口的地址先送到DX中，用DX作间接寻址寄存器。

例如：MOV DX，378H
OUT DX，AL

如果访问的端口地址值大于255，则必须用I/O端口的间接寻址方式。

3.2 指令系统

一种计算机所能执行的各种类型的指令的集合称为该计算机的指令系统。

Intel8086/8088CPU指令系统的指令可以分为六大类：

1.传送类指令　　2.算术运算类指令　　3.位操作类指令

4.串操作类指令　　5.程序转移类指令　　6.处理器控制类指令

从指令的格式划分，一般可以分为三种：

1.双操作数指令：OPR DEST SRC
2.单操作数指令：OPR DEST
3.无操作数指令：OPR

对于无操作数指令，包含两种情况：

（1）指令不需要操作数，如暂停指令HLT。

（2）在指令格式中，没有显式地指明操作数，但是它隐含指明了操作数的存放地方，如指令PUSHF。

一、传送类指令

传送类指令的作用是将数据信息或地址信息传送到一个寄存器或存储单元中，可以分为以下四种情况。

1.通用数据传送指令

指令格式：MOV DEST，SRC

作用：将源操作数指定的内容传送到目的操作数，即DEST<=(SRC)。

当指令执行完后，目的操作数原有的内容被源操作数内容覆盖，即目的操作数和源操作数具有相同内容。

MOV指令对标志寄存器的各位无影响

MOV指令可以是*字节*数据传送也可以是*字*数据传送，但是源操作数和目的操作数的长度必须一致。

MOV指令可以分为以下几种情况：

（1）立即数传送到通用寄存器或存储单元

例：MOV AH，10H
MOV AX，2345H
MOV M-BYTE, 64H
MOV M-WORD，2364H

注意：立即数只能作为源操作数，立即数不能传送给段寄存器。

（2）寄存器之间的传送

例：MOV AH，CH
MOV DS, AX
MOV ES, BX
MOV AX，CS
MOV CS, AX;错误

注意：段寄存器CS只能作源操作数，不能作目的操作数。

（3）寄存器与存储单元之间传送

例：MOV AL，[SI]
MOV [DI]，AH
MOV AX, 10[BX]
MOV TABLE[BP], BX
MOV DS, [SI][BX]

MOV [BX], [BP][SI];错误

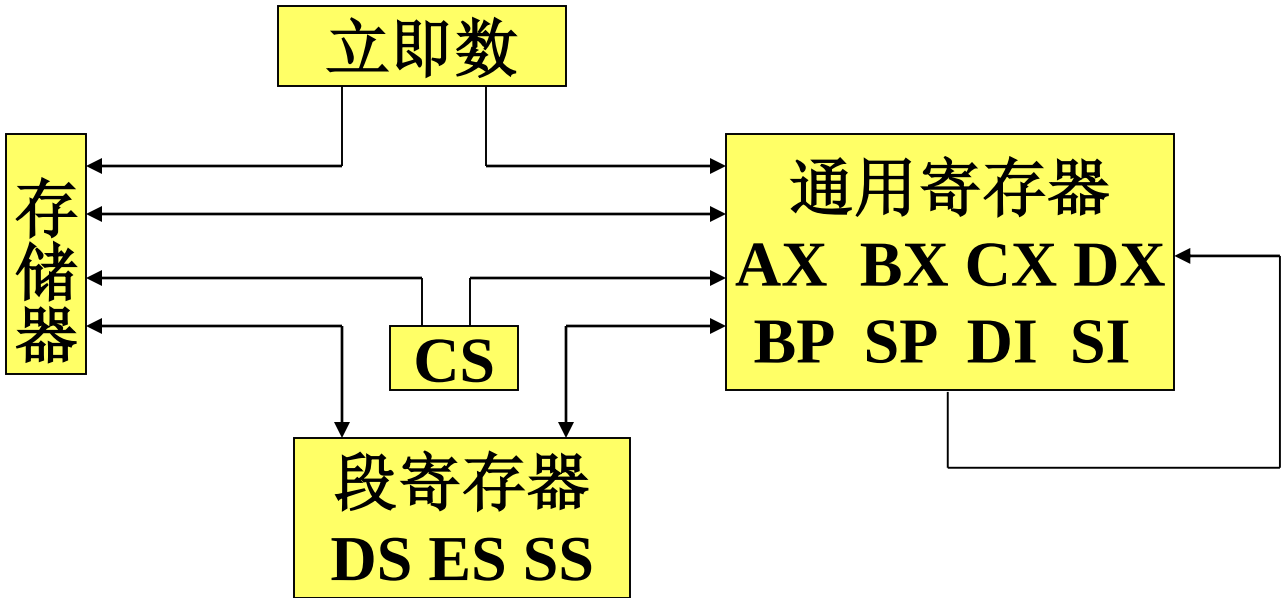
综合起来，MOV指令在使用时需注意以下几个问题：

（1）立即数只能作源操作数，且它不能传送给段寄存器。

（2）段寄存器CS只能作源操作数，段寄存器之间不能直接传送。

（3）存储单元之间不能直接传送数据

（4）MOV指令不影响标志位



2.交换指令

指令格式：XCHG DEST，SRC

作用：源操作数和目的操作数两者内容相互交换，即：（DEST）<=>(SRC)。

指令对标志寄存器各位无影响

数据交换可以在寄存器之间或寄存器与存储器单元之间进行。但是不能在存储单元之间直接进行数据交换。寄存器只能使用通用寄存器。

例 XCHG AX，BX
XCHG AH，CH

为了完成两个存储单元(DA_BYTE1和DA_BYTE2)之间的数据交换可以使用以下三条指令来实现。

```
MOV  AL，DA-BYTE1；     AL<=(DA_BYTE1)
XCHG  AL，DA-BYTE2 ；  (AL)<=>(DA-BYTE2
XCHG  AL，DA-BYTE1 ；  (AL)<=>(DA-BYTE1)
  或MOV DA-BYTE1，AL;(DA_BYTE1)<=(AL)
```

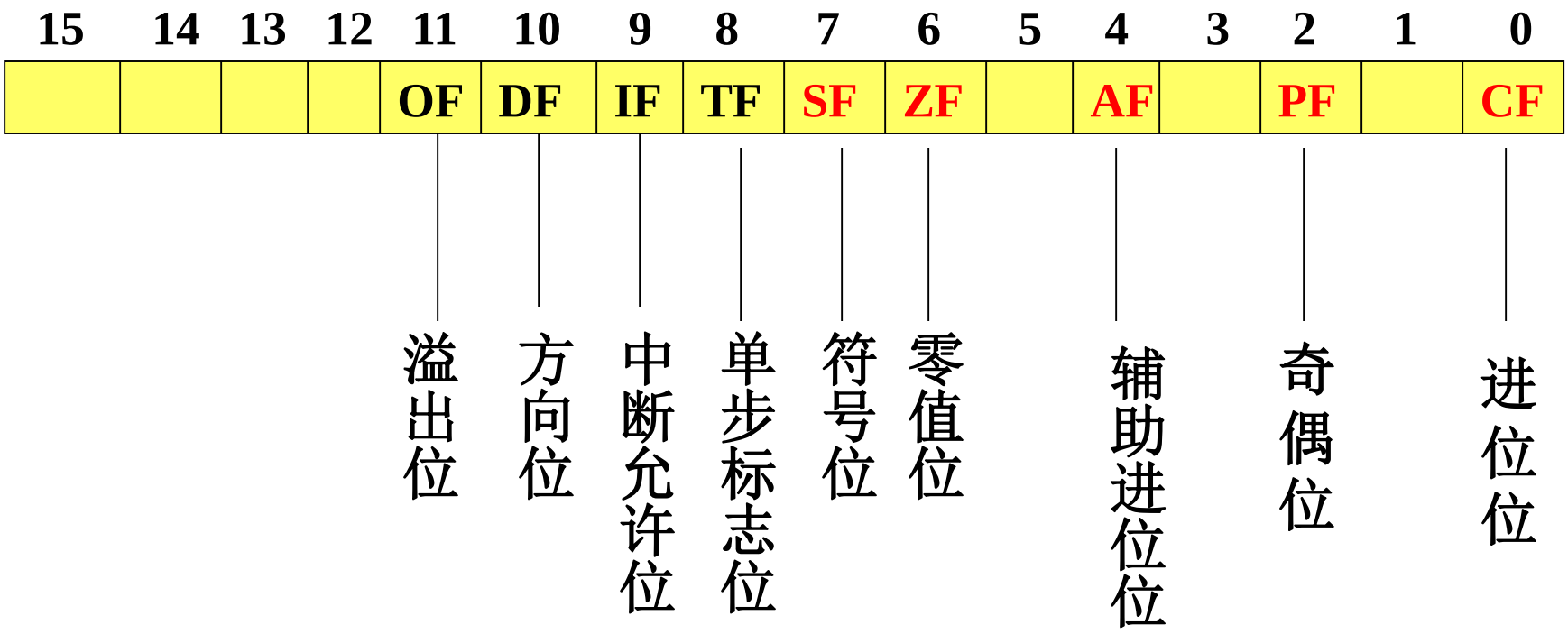
3.标志传送指令

对标志寄存器进行存取的指令有4条，它们都是无操作数指令，即指令隐含指定标志寄存器、AH寄存器或堆栈为操作数。

（1）取标志寄存器指令

指令格式：LAHF

作用：将标志寄存器的低8位送入AH寄存器，即将标志SF、ZF、AF、PF和CF分别送入AH的第7、6、4、2、0位，而AH的第5、3、1位不确定。



指令执行对标志寄存器各位无影响，即标志寄存器各位不变.

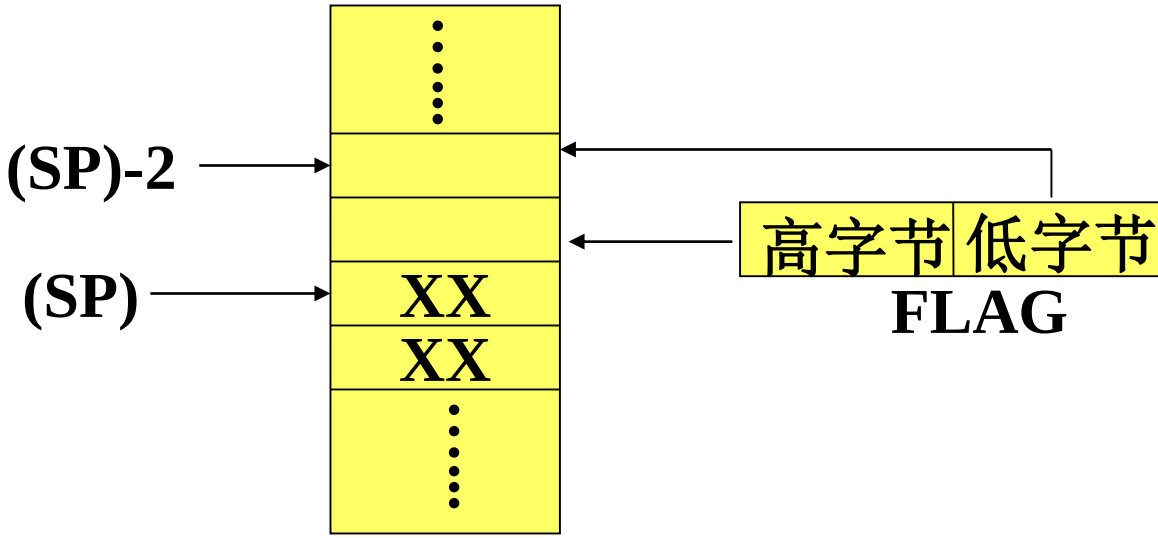
（2）存储标志寄存器指令

指令格式：SAHF

作用：将寄存器AH中的第7、6、4、2、0位分别送入标志寄存器的SF、ZF、AF、PF和CF各标志位。而标志寄存器高8位中的各标志位不受影响。

（3）标志进栈指令

指令格式：PUSHF



作用：先将堆栈指针SP减2，使其指向堆栈顶部的空字单元，然后将16位标志寄存器的内容送SP指向的字单元。

（4）标志出栈指令

指令格式：POPF

作用：将由SP指向的堆栈顶部的一个字单元的内容送入标志寄存器，然后SP的内容加2.

4.地址传送指令

这类指令有3条，它们的作用是将存储单元的地址送寄存器。

（1）装入有效地址

格式：LEA DEST，SRC

其中：源操作数SRC必须是一个字节或字存储器操作数（地址），DEST必须是一个16位通用寄存器。

作用：将SRC存储单元地址中的偏移量，即有效地址EA传送到一个16位通用寄存器中。

指令执行对标志寄存器各位无影响。

例1：LEA AX，[BX] [SI]

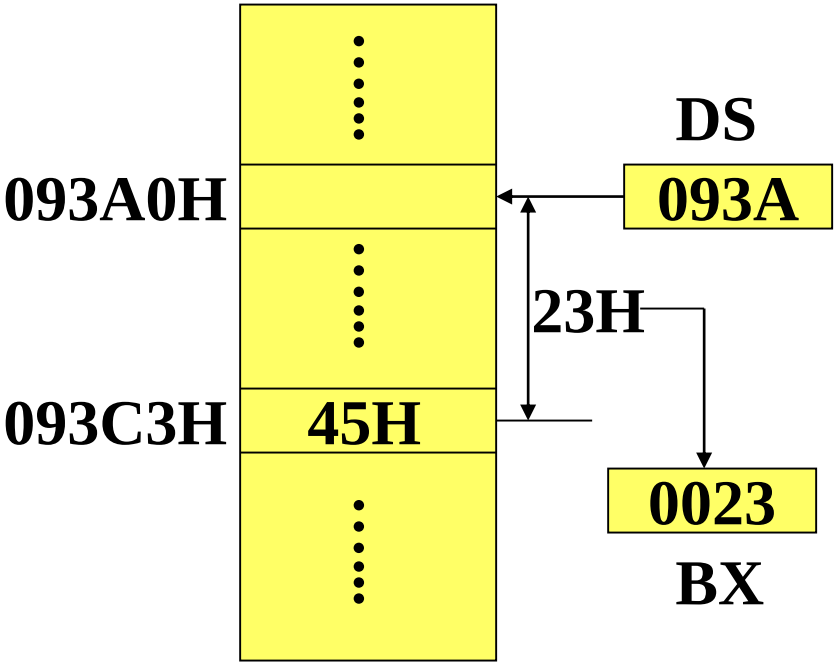
源操作数使用的是基址变址寻址方式,它所形成的有效地址就是BX的内容加上SI的内容。即

AX<=(BX)+(SI)

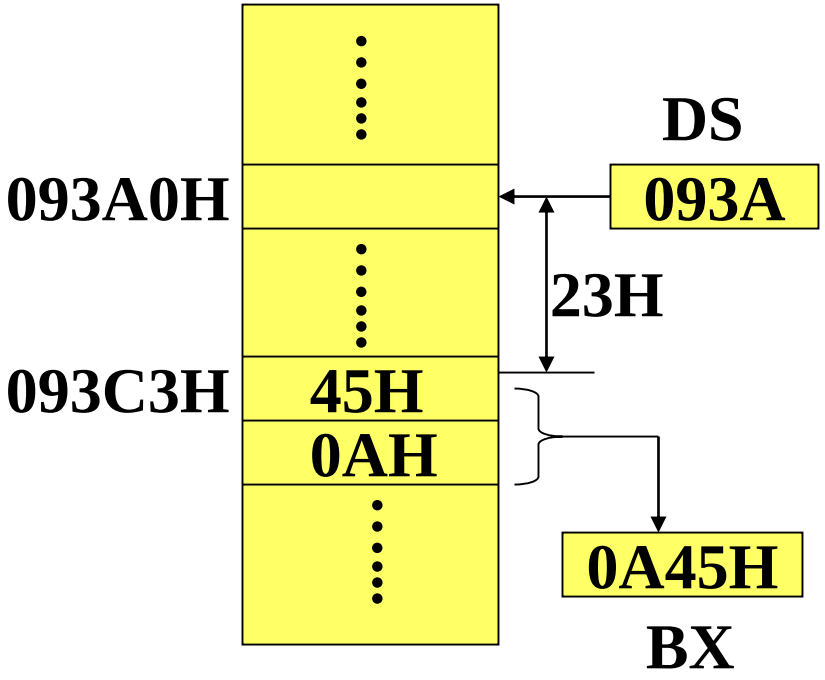
注意：它不是将BX和SI所寻址的存储单元的内容送入AX。

例2 比较指令 LEA BX，DS:[23H]与 MOV BX，DS:[23H]的功能

LEA BX，DS:[23H]



MOV BX, DS:[23H]



（2）装入地址指针指令

格式：LDS DEST，SRC
LES DEST，SRC

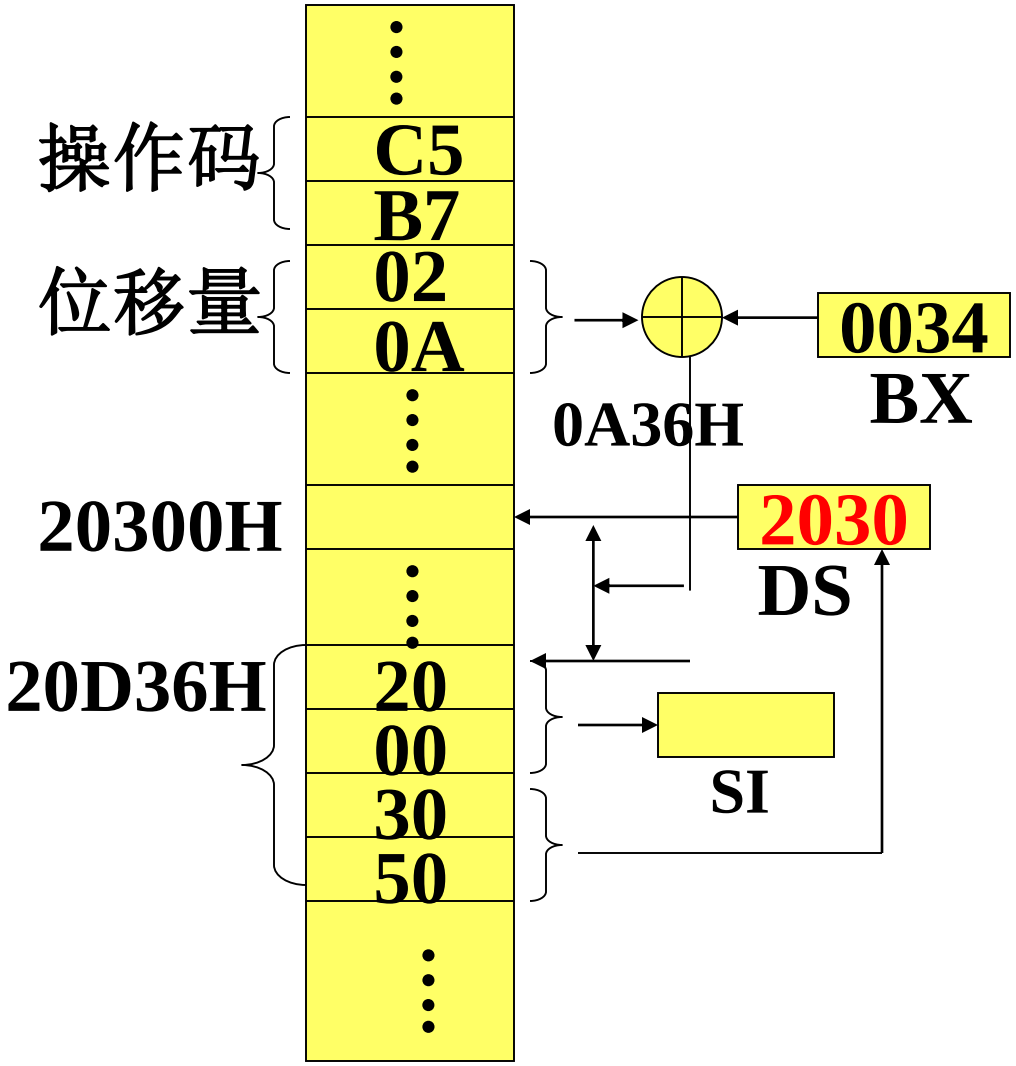
其中:DEST是任意一个16位通用寄存器。SRC必须是一个存储器操作数。

作用：把SRC存储单元开始的4个字节单元的内容(32位地址指针)送入DEST通用寄存器和段寄存器DS（LDS指令）或ES（LES指令），其中低字单元内容为偏移量送通用寄存器，高字单元内容为段基值送DS或ES。

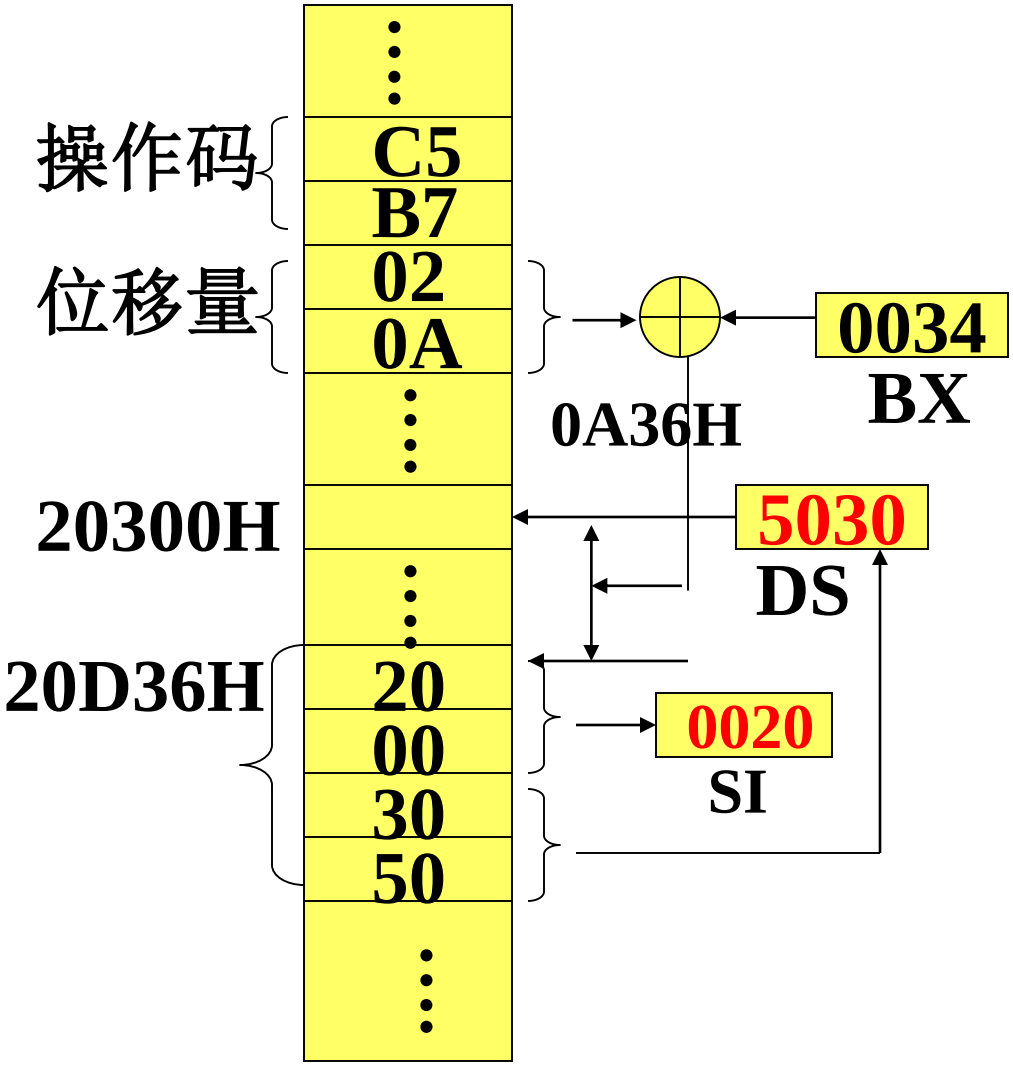
例：LDS SI，TABLE[BX]

设 TABLE的值为0A02H, (BX)=34H,(DS)=2030H

指令执行前



指令执行后


操作码
C5
B7
位移量
02
0A
0034
BX
0A36H
20300H
5030
DS
20D36H
20
00
30
50
0020
SI

二、算术运算类指令

8086/8088指令系统中有加、减、乘、除指令，这些指令可以对字节数据或字数据进行运算。

参加运算的数可以是无符号数，也可以是带符号数。带符号数用补码表示。

参加运算的数可以是二进制数，也可以是十进制数（以**BCD**码表示）。

在本节中只讨论基本的加法和减法指令，其它指令在以后的章节中介绍.

1.加法指令

指令格式：ADD DEST，SRC

功能：目的操作数和源操作数相加，其和存放到目的操作数中，而源操作数内容保持不变，即

DEST<=(DEST)+(SRC)。

根据相加的结果将影响到标志寄存器的CF、PF、AF、ZF、SF和OF。

DEST只能是通用寄存器或存储器操作数。不能是立即数。

SRC可以是通用寄存器、存储器或立即数操作数

DEST和SRC不能都为存储器操作数。

ADD指令可以是字节操作数相加，也可以是字操作数相加。

例 分析下列各指令功能

(1) ADD AX,CX

功能：将寄存器AX的内容与CX的内容相加，结果传送到AX中

(2) ADD AH,DATA_BYTE

功能：将由直接寻址方式所指示的存储单元的内容与AH内容相加，结果送回AH中。

(3) ADD CX,10H

功能：将常数10H加入到CX中。为字操作数指令.

(4) ADD AX, [BX][SI]

功能：将由基址变址寻址方式所指示的存储单元的内容加入到AX中。

2.带进位加法指令

指令格式：ADC DEST，SRC

该指令的功能与ADD基本相同，所不同的是其结果还要加上进位标志CF的值，即：

DEST<=(DEST)+(SRC)+CF

根据相加的结果设置标志寄存器中的CF、PF、AF、ZF、SF和OF

注意：参加运算的进位CF是本条指令执行之前的值。

用ADC指令可实现数据长度大于16位的两数相加

例：计算12349678H＋377425H

```
MOV  AX, 1234H
MOV  BX，9678H
ADD  BX，7425H
ADC  AX，37H
```

指令执行后，结果的高16位在AX，低16位在BX中。

3.加1指令

指令格式：INC DEST

该指令为单操作数指令，其功能是将目的操作数加1，并送回到目的操作数，即：

DEST<=(DEST)+1

目的操作数可以是任意的8位、16位通用寄存器或存储器操作数。目的操作数被视为带符号二进制数

根据指令执行结果设置PF、AF、ZF、SF和OF标志，但不影响CF。INC指令主要用于某些计数器的计数和修改地址指针。

例：INC CL
INC SI
INC COUNT

4.减法指令

指令格式：SUB DEST，SRC

功能:目的操作数的内容减去源操作数的内容，结果送入目的操作数，源操作数中内容保持不变。

即：DEST<=（DEST）－（SRC）

操作结果将影响标志位CF、PF、AF、ZF、SF和OF。

目的操作数DEST和源操作数SRC可以是8位或16位的通用寄存器、存储器操作数，但两者不能同时为存储器操作数。立即数只能作源操作数。

例：**SUB AX，BX**

SUB AH，10H

SUB DX,DA-WORD

SUB DA-BYTE,BL

注意：减法指令对借位标志的影响，若采用变减为加的运算方法，则产生的进位与CF标志结果相反。

5.带借位减法

指令格式：SBB DEST，SRC

该指令的功能与SUB指令基本相同，不同的是在两个操作数相减后再减去进位标志CF的值。

即：DEST<=(DEST)－(SRC)－CF。

注意：该CF的值是本条指令执行前的结果。

SBB指令在使用上与ADC类似，主要用于长度大于16位的数相减，即将低16位相减的结果引入高位部分的减法中。

根据指令执行结果设置PF、AF、ZF、SF 、OF和CF。

6.减1指令

指令格式：DEC DEST

该指令为单操作数指令，将目的操作数的内容减1后，送回到目的操作数。即：DEST<=(DEST)－1

DEST可以是8位或16位的通用寄存器存储器操作数，该指令将DEST看作是带符号二进制数。

根据指令执行结果设置PF、AF、ZF、SF和OF，但不影响CF。

DEC指令的使用类似INC指令。主要用于计数和修改地址指针，计数方向与INC指令相反。

```
例          MOV  AL，10H
    LOP：  DEC   AL
           JNC    LOP
```

上述程序段中，是一个错误应用DEC指令的例子。

7.求负数指令

指令格式：NEG DEST

功能：用零减去目的操作数的内容，并送回目的操作数，即：DEST<=0-(DEST)

DEST可以是任意一个8位或16位的通用寄存器或存储器操作数，被视为带符号的操作数。

由于机器中带符号数用补码表示，求操作数的负数就是求补操作。因此，NEG指令也叫取补指令。

NEG指令将影响标志PF、AF、ZF、SF、CF和OF。

对进位标志CF的影响：

只有当操作数为零时，进位标志CF被置零，其它情况都被置1.

对溢出标志OF的影响：

当字节操作数为-128，或字操作数为-32768时，执行NEG指令的结果操作数将无变化，但溢出标志OF被置1.

例1　设AL中存放一个正数(AL)=25H，BL中存放一个负数：(BL)=－58H，求它们的相反数，即负数。

```
NEG   AL
NEG   BL
```

指令执行后，(AL)=-25H=11011011B
(BL)= 58H=01011000B

例2 一个32位带符号数存放在DAW开始的四个字节存储单元中，DAW字节单元存放最低字节。求该数的负数，并存入原存储单元中。

```
NEG   WORD PTR DAW
MOV  AX，0
SBB   AX，DAW+2
MOV  DAW+2，AX
```

结果的低16位由指令NEG直接得到，而高16位还要考虑低16位产生的借位，因此使用了带借位的指令SBB。

三、位操作类指令

1.逻辑运算指令

逻辑运算指令共有4条，它们的指令格式分别是：

逻辑“与”指令	AND DEST，SRC
逻辑“或”指令	OR DEST，SRC
逻辑“异或”指令	XOR DEST，SRC
逻辑“非”指令	NOT DEST

这4条指令都是执行按位逻辑运算，如下表所示：

DEST	SRC	AND	OR	XOR	NOT
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

DEST和SRC可以是8位或16位的通用寄存器或存储器操作数，但两者不能同时为存储器操作数，SRC可以为立即数。

逻辑指令对标志位的影响：

NOT指令对标志无影响。而其余三条指令将根据结果影响SF、ZF和PF，而CF和OF总 是置0，AF为不确定。

逻辑运算指令除用来实现各种逻辑运算之外，还常用于对字节或字数据的某些位的组合、分离或位设置。

例1：AND AH，0F0H；　分离出AH中的高4位.

AND AH，0FH；　分离出AH中的低4位

OR　AH，01H；　将AH中最低位置1

AND AL，7FH；　将AL的最高位置0

XOR AX，0FFH；　将AX的低字节变反

XOR　BX，8000H；将BX的符号位变反

例2：下面的程序段将中断标志位IF清0，其它标志位保持不变。

```
PUSHF                ;将标志寄存器压栈
POP  AX              ;将栈中的标志字送AX
AND  AX，0FDFFH      ;将AX的第9位清0
PUSH  AX             ;将第9位清0后的AX内容压栈
POPF                 ;将堆栈中的值返回到标志寄存器
```

2.测试指令

指令格式：TEST DEST，SRC

该指令的功能与AND指令相似，实现源操作数与目的操作数进行按位“逻辑与”运算，对标志位的影响与AND指令相同，但运算的结果不送入目的操作数，即目的操作数内容也将保持不变。

TEST指令主要用于测试某一操作数的一位或几位的状态。

例1 TEST AL，01
JZ ZERO
…….
ZERO: ……

该程序段检查AL寄存器的最低位是否为0，如果为0，则程序转移到ZERO处执行。

例2 LAHF
TEST AH，04H
JZ TARGET

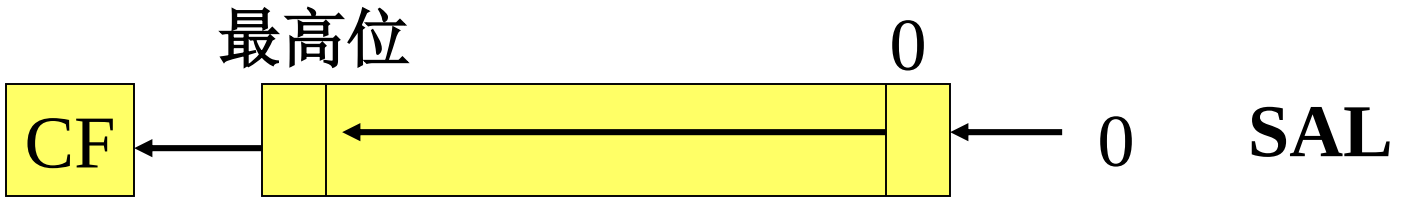
该程序段检查标志寄存器的PF位（第2位）是否为0，如果为0，则执行后标志ZF为1。因此通过测试ZF标志即可。

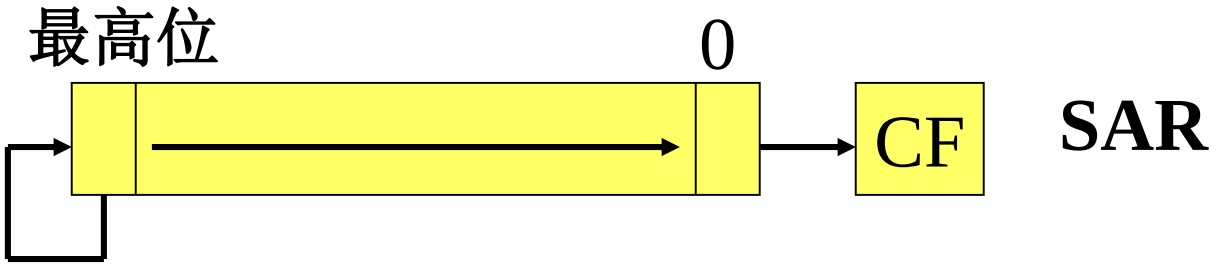
3.移位/循环移位指令

这一类指令共有**8条**,分为**3类**。

（1）算术移位

算术左移 **SAL DEST，COUNT**
算术右移 **SAR DEST，COUNT**

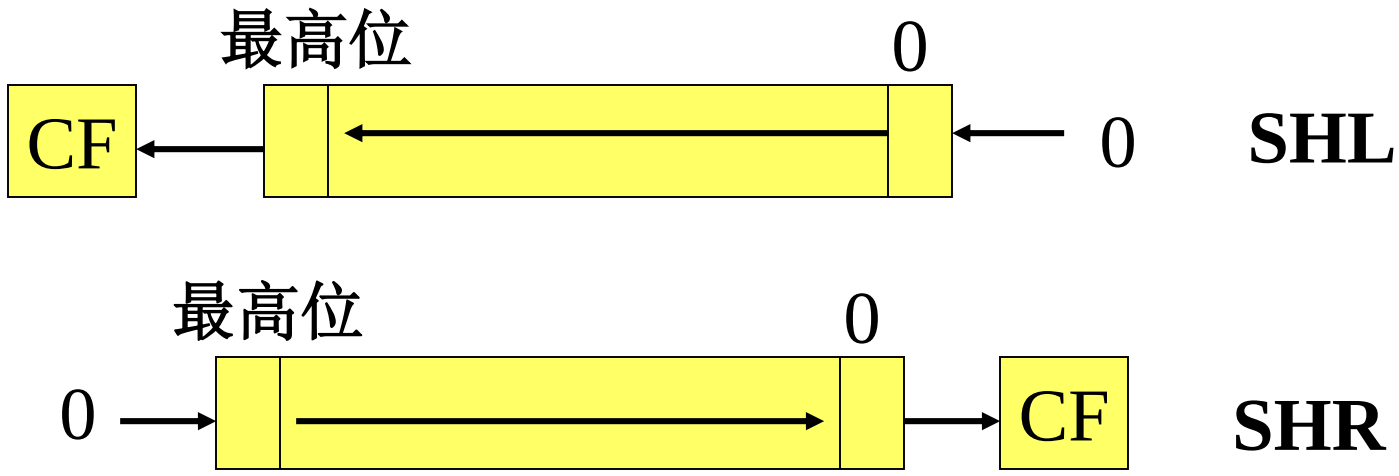




（2）逻辑移位

逻辑左移 SHL DEST，COUNT
逻辑右移 SHR DEST，COUNT

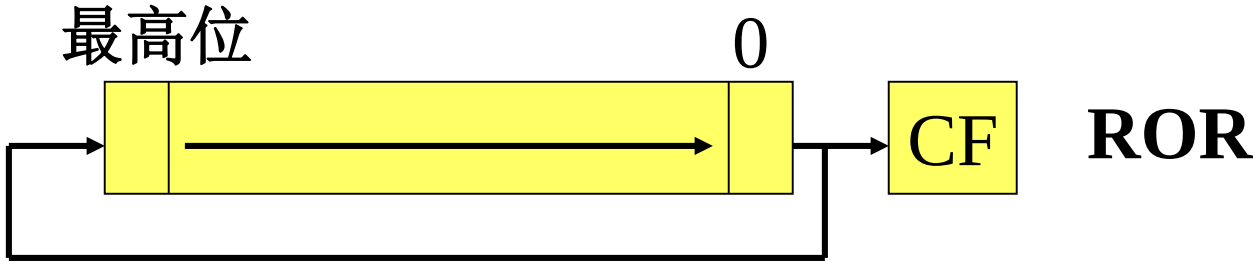
逻辑左移SHL与算术左移SAL功能相同。



（3）循环移位

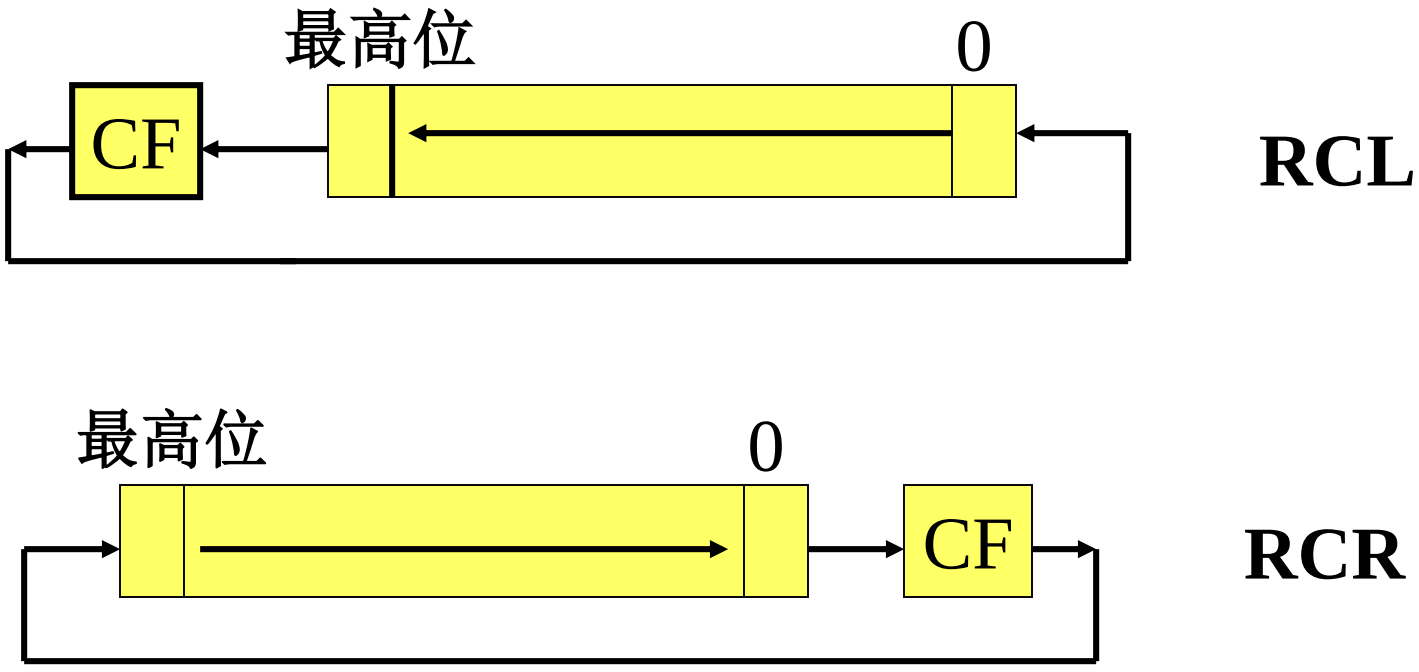
小循环：

循环左移 **ROL DEST，COUNT**
循环右移 **ROR DEST，COUNT**



大循环：

带进位循环左移 **RCL DEST，COUNT**
带进位循环右移 **RCR DEST，COUNT**



这8条指令具有以下几个共同点：

（1）DEST为操作对象，它可以是字节或字操作数，可以是通用寄存器或存储器操作数。

（2）COUNT用来决定移位/循环的位数，即确定移位的次数。

当移位次数为1时，使用常数1或寄存器CL。

当移位次数大于1时，必须使用寄存器CL。

例1：SAL AX，1；将AX的内容左移1位，其中最高位移入CF中，而低位补0.

例2：MOV CL，2
SAR AX，CL；将AX的内容算术右移2位。

（3）在执行移位时，根据指令不同，每移位一次,最高位（左移）或最低位（右移）都要送到进位标志CF。

例3：MOV AL，10010011B
SHL AL，1 ；执行后CF标志为1
SAR AL，1 ；执行后CF标志为0

（4）前4条移位指令根据移位结束后修改标志位CF、PF、ZF、SF和OF，而AF不确定。而后4条循环移位指令根据移位结束后的结果仅修改CF和OF

对溢出标志位OF的影响：

移位次数为1时，移位前后操作数的符号位发生变化，则OF被置1，否则置0。移位次数大于1时，OF不确定。

```
例4：MOV  AL，11000000B；(AL)=-64
     MOV   BL，01111111B；(BL)=127
     SAL   AL，1 ；(AL)=10000000B=-128,OF=0
     SAL   BL，1 ；(BL)=11111110B=-2, OF=1
```

指令SAL和SAR当移位次为n时，其作用相当于乘以2^n或除以2^n，因此被叫做算术移位指令。

为了保持其算术运算结果的正确性，移位后的结果不能发生溢出。

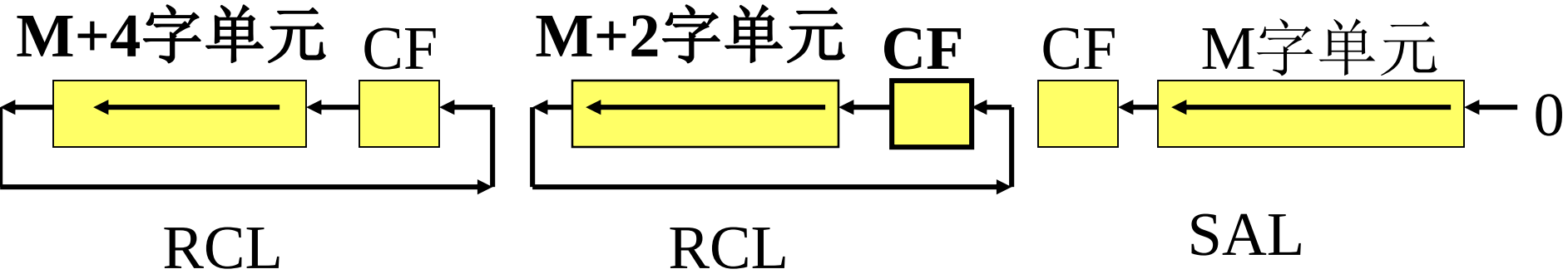
例5 设AX中存放一个带符号数，若要实现(AX) ×5÷2，可由以下几条指令完成。

```
MOV    DX，AX
SAL    AX，1
SAL    AX，1
ADD    AX，DX
SAR    AX，1
```

对于多字节或多字数据的移位，需要使用带进位循环移位指令。

例6　下面程序段对从存储单元M开始的三字数据执行左移一位。

```
SAL　M，1
RCL　M+2，1
RCL　M+4，1
```



下面的程序段实现将上述三字数据右移一位。

```
SAR   M+4，1
RCR   M+2，1
RCR   M，1
```

四、处理器控制类指令

处理器控制类指令包括以下三种情况。

1.标志位操作指令

它们都是无操作数指令，操作数隐含为标志寄存器的某个标志位。能直接操作的标志位有CF、IF和DF。

（1）清除进位标志　　CLC ；置CF为0

（2）置1进位标志　　STC ；置CF为1

（3）进位标志取反　　CMC ；CF的值取反

（4）清除方向标志	CLD；置DF为0
（5）置1方向标志	STD；置DF为1
（6）清除中断标志	CLI；置IF为0
（7）置1中断标志	STI；置IF为1

2、与外部事件同步的指令

HLT ；暂停指令

WAIT ；等待指令

ESC ；外部协处理器指令前缀

LOCK ；总线锁定指令

3、空操作指令 NOP

执行一次NOP占用CPU三个时钟周期，它不改变任何寄存器或存储单元内容，主要用于延时。

3.3 指令编码

汇编:将汇编语言程序转换为机器语言程序的过程

汇编程序:在计算机中实现汇编过程的系统程序

Intel8086/8088汇编指令的编码格式有四种基本格式。

1.双操作数指令编码格式

2.单操作数指令编码格式

3.与AX或AL有关的指令编码格式

4.其它指令编码格式

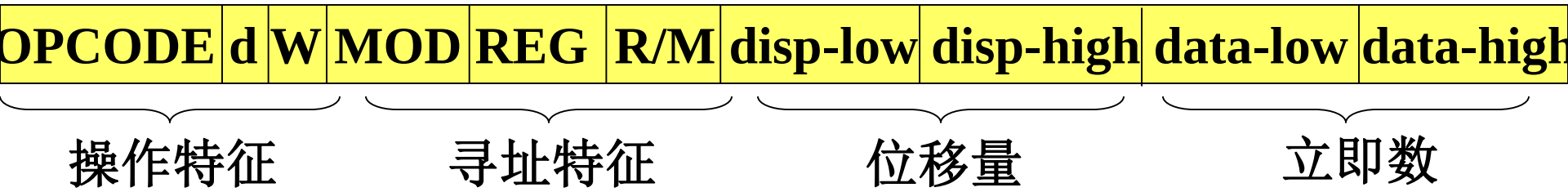
一、双操作数指令编码格式

对于象MOV、ADD、AND等双操作数指令，操作数可以是以下两种情形：

- 一个操作数在寄存器中，另一操作数在寄存器或存储器中。

- 目的操作数在寄存器或存储器中，源操作数是立即数。

这类指令的机器目标代码长度为2～6个字节



整个指令编码可以包含4个部分，但其中某些部分在一些指令的编码中可以没有。

1.操作特征部分

这部分为指令编码的首字节，它又分为以下三个段。

（1）OPCODE:操作码字段

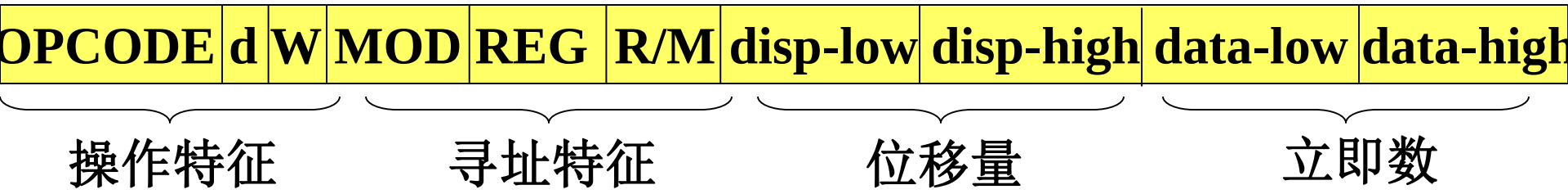
该字段长度为6bit。它表示了该指令所执行的功能和两个操作数的来源。

例如：

操作码	指令	目的操作数	源操作数
100010	MOV	REG	R/M
1100011	MOV	R/M	Imm
000000	ADD	REG	R/M
001000	AND	R/M	REG
1000000	AND	R/M	Imm
1000000	OR	R/M	Imm

如果指令的源操作数是立即数，则需要使用指令编码的第2字节中**REG**字段作辅助操作码。

前面例子中的最后两条指令，虽然其**OPCODE**字段相同，但它们的辅助操作码字段不同。



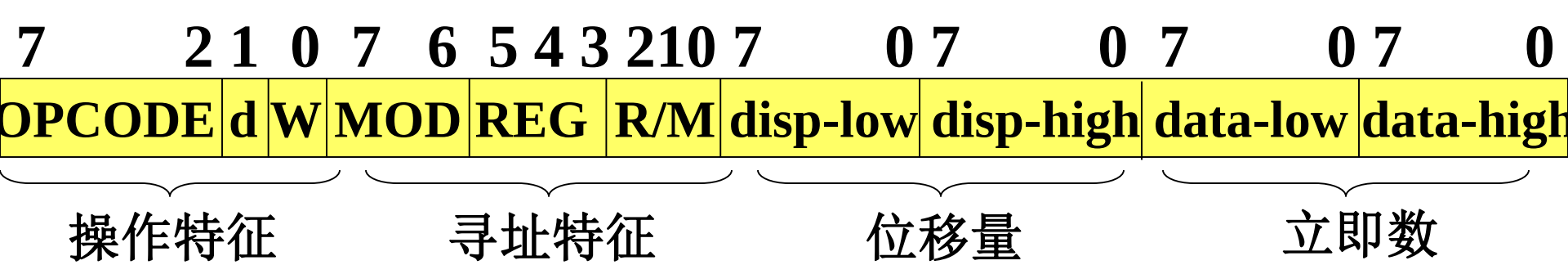
（2）方向字段d

该字段与第2部分寻址特征一起来决定源操作数和目的操作数的来源。

注意：当源操作数为立即数Imm时，d字段无效，它被并入 操作码字段。

（3）字/字节字段W

当W=1时，表示两操作数长度为字；当W=0时，表示两操作数长度为字节。



2.寻址特征部分

它与操作特征部分的方向字段d结合，指定两个操作数分别使用什么寻址方式，及使用哪个寄存器。

它包括MOD、REG和R/M三个字段，REG字段确定一个操作数,而MOD和R/M字段确定另一个操作数。

当d=1时，则目的操作数由REG字段确定，而源操作数由MOD和R/M字段确定。

当d=0时，则目的操作数由MOD和R/M 字段确定，而源操作数由REG字段确定。

（1）REG字段

由REG字段确定的一个操作数是某一通用寄存器的内容，即使用的是寄存器寻址方式。

第一部分中的W字段决定操作数是字或是字节。

它们配合使用可以有16种组合，也即可以分别指定16个寄存器之一。如下表所示：

REG	000	001	010	011	100	101	110	111
W=0	AL	CL	DL	BL	AH	CH	DH	BH
W=1	AX	CX	DX	BX	SP	BP	SI	DI

如果REG字段被用于指定段寄存器（用于MOV指令），则它的编码与指定的段寄存器如下。

REG	000	001	010	011
段寄存器	ES	CS	SS	DS

（2）寻址方式字段MOD和寄存器/存储器字段R/M

这两个字段共同确定一个操作数。该操作数可以在寄存器中，也可以在存储器中

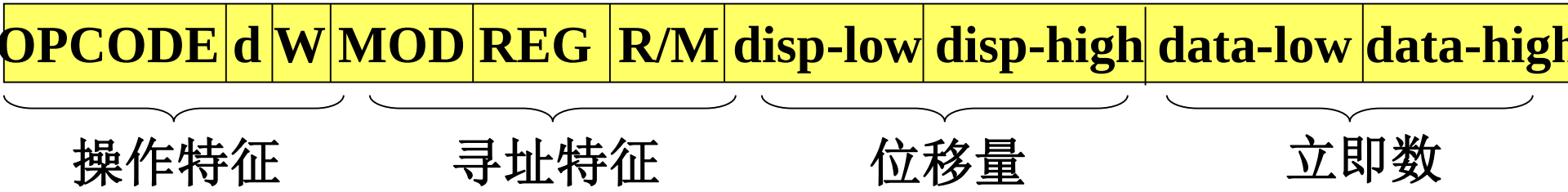
MOD、R/M和W字段共同确定操作数的寻址方式和所使用的寄存器，如下表所示。

R/M \ MOD	存储器有效地址计算方法			寄存器方式(11)	
	00	01	10	w=0	w=1
000	(BX)+(SI)	(BX)+(SI)+disp8	(BX)+(SI)+disp16	AL	AX
001	(BX)+(DI)	(BX)+(DI)+disp8	(BX)+(DI)+disp16	CL	CX
010	(BP)+(SI)	(BP)+(DI)+disp8	(BP)+(DI)+disp16	DL	DX
011	(BP)+(DI)	(BP)+(SI)+disp8	(BP)+(SI)+disp16	BL	BX
100	(SI)	(SI)+disp8	(SI)+disp16	AH	SP
101	(DI)	(DI)+disp8	(DI)+disp16	CH	BP
110	disp16	(BP)+disp8	(BP)+disp16	DH	SI
111	(BX)	(BX)+disp8	(BX)+disp16	BH	DI

注意：在表中没有使用BP作寄存器间接寻址方式，如果在指令中使用了[BP]，则将其汇编为[BP+0]，即基址寻址。

当MOD=11时，操作数为16个寄存器之一的内容。

当MOD=00,01,11时,操作数为存储器单元,可有24种有效地址EA计算方法。disp8和disp16分别为8位和16位位移量



3.位移量部分

根据寻址特征中MOD和R/M字段确定的有效地址计算方法，位移量可以是以下三种情况之一：

没有位移量　　1字节位移量disp8　　2字节位移量disp16

4.立即数部分

如果指令的源操作数为立即数，则指令编码中包含有该部分。它总是位于指令编码的最后1~2字节。

例1.MOV M-WORD，OAABBH

该指令功能为将16位立即数送存储单元，目的操作数为直接寻址方式。

查附录B可知：指令操作码为1100011

字操作，W=1

源操作数为立即数，REG字段为辅助操作码000

设M-WORD存储单元的偏移量为0010H

由于目的操作数为直接寻址，根据前面的MOD和R/M字段编码表可知MOD=00 R/M=110

则整个指令的编码为：

OPCODE	W	MOD	REG	R/M	disp-low	disp-high	data-low	data-high
1100011	1	00	000	110	10	00	BB	AA

用16进制数表示为：C7 06 10 00 BB AA共6个字节。

例2. MOV DS，AX

该指令将通用寄存器AX的内容送入段寄存器DS，因此REG字段必须用于指定DS，即为011。

MOD和R/M用于指定AX，即MOD=11 R/M=000

d＝1，w被作为OPCODE 查表为100011d0

整个指令编码为：

OPCODE	MOD	REG	R/M
10001110	11	011	000

16进制目标代码为：8ED8

例3 MOV AX，ES：[BX]

该指令为寄存器间址的存储单元内容送通用寄存器AX

指令中使用了段前缀ES，即由ES代替数据段DS。指令编码的第一个字节就为段前缀标记代码。

段前缀	代码
ES	00100110 (26)
CS	00101110 (2E)
SS	00110110 (36)
DS	00111110 (3E)

段前缀标记字节的前3位和后3位被固定为001和110，中间两位被用来指定不同的段寄存器。

该指令编码为：

段前缀码	OPCODE	d	w	MOD	REG	R/M
00100110	100010	1	1	00	000	111

16进制目标代码为：26 8B 07

二、单操作数指令编码格式

这种编码格式适用于只有一个操作数的指令，如INC、DEC、移位/循环等指令。指令编码为2~3字节。

操作特征部分：

包括OPCODE 、V和W三个字段，其中V字段只有移位/循环指令中才有该字段。其它指令中没有该字段。

V=0时，指令中使用常数1作为移位或循环次数。
V=1时，指令中使用寄存器CL作移位次数。

由于单操作数指令中只有一个操作数，因此寻址特征部分就不需要REG字段，而该字段被用作辅助操作码。

例1　INC　AL

该指令为将寄存器AL内容加1，查表可知其操作码和辅助操作码分别为1111111和000 。

该指令编码为

OPCODE	W	MOD	OPCODE	R/M
1111111	0	11	000	000

16进制目标代码为：FE　C0

例2　SHR AL，CL

该指令为对寄存器AL内容执行逻辑右移，移位次数由CL给出,即V字段为1。由MOD和R/M确定AL，即MOD=11 R/M=000

查附表可知：操作码和辅助操作码分别为110100和101

指令编码为：

OPCODE	V	W	MOD	OPCODE	R/M
110100	1	0	11	101	000

16进制目标代码为：D2 E8

三、与AX或AL有关的指令编码格式

这种编码格式用于隐含指定AX/AL作为一个操作数的双操作数指令，其编码格式为：

OPCODE	W	disp-low	disp-high	data-low	data-high
7 1	0	7 0	7 0	7 0	7 0

采用这种编码格式的指令，除一个操作数隐含指定为AX/AL外，另一个操作数可以是立即数或存储单元。

立即数：则编码中应有1~2字节的立即数

存储单元：只能使用直接寻址方式，位移量由disp字段给出。

例1 AND AL，0FH

该指令功能是将寄存器AL的内容与立即数0FH进行逻辑“与”。因此指令编码中包含了立即数（8位）部分，而不包含位移量。

查附录二可知其操作码部分为0010010，指令编码如下：

OPCODE	W	data
0010010	0	00001111

16进制目标代码为：24 0F

四、其它指令编码格式

除上述三种编码格式外，还有一些指令的编码格式更简单。如标志位操作指令、堆栈操作指令等。这些指令的编码格式一般只有一个字节。

例如 CLC清进位标志，该指令的编码只有一个字节的操作码。即：

7							0
OPCODE							
1	1	1	1	1	0	0	0

在有些单字节指令的编码中，将该字节划出部分位作为REG字段。例如PUSH指令，若压入堆栈的是通用寄存器，则编码格式为：

7 ~ 3	2 1 0
OPCODE	REG
01010	

若压栈的是段寄存器则编码格式为：

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