第 3 章　80x86 指令系统和寻址方式

指令是计算机所能接受软件工作者命令的最小工作单元，它最终是由计算机内的电器元件状态来体现的，这一状态为译码器所识别，并经一定时间周期付诸实施，我们称这种指令为机器指令。一条机器指令应包含两部分内容：一是要给出此指令要完成何种操作，这部分称为指令操作码部分；另一部分是要给出参与操作的对象是什么，此部分称为操作数部分，在指令中可以直接给出操作数的值，或者操作数存放在何处，操作的结果应送往何处等信息。处理器可根据指令字中给出的地址信息求出存放操作数的地址——称为有效地址，然后对存放在有效地址中的操作数进行存取操作。指令中关于如何求出存放操作数有效地址的方法称为操作数的寻址方式。计算机按照指令给出的寻址方式求出操作数有效地址和存取操作数的过程，称为寻址操作。计算机的指令系统就是指该计算机能够执行的全部指令的集合。

本章首先介绍 16 位微处理器 8086/8088 的指令系统和寻址方式，在此基础上以 80386 为对象讲述 32 位微处理器的指令系统和寻址方式。

本章的内容有：

- 8086 指令系统概述。
- 8086 的寻址方式和指令系统。
- 80386 的寻址方式和指令系统。
- 80486/Pentium 微处理器新增指令。

3.1　8086 指令系统概述

1. 数据类型

计算机中的数是用二进制表示的，数的符号也是用二进制表示的。把一个数连同其符号在内在机器中的表示加以数值化称为机器数。计算机执行指令过程中需要处理各种类型的机器数，可处理的数据类型有七种。

1）无符号二进制数

指不带符号位的一个字节、字或双字的二进制数。

2）带符号二进制数

此类数有正、负之分，均以补码表示。正、负数补码采用符号＋绝对值表示，即数的最高有效位为 0 表示符号为正，否则为负，数的其余部分则表示数的绝对值。需注意的是，负数的补码可先写出该数的正数补码，然后将其按每位求反，最后在末位加 1。

3）BCD 码

有非压缩 BCD 码和压缩 BCD 码两种，前者用一个字节表示一个 0～9 的十进制数，这些数为无符号数并按字节存放，数的大小由低半字节确定；后者用一个字节来表示两个 0～9 十进制数。高半字节为高位，低半字节为低位。每半个字节中的取值只能是 0～9，

CPU 只支持 8 位非压缩 BCD 码和压缩 BCD 码。

4) 数的定点和浮点表示法

在计算机中,针对小数点的处理有两种方法:定点表示法和浮点表示法。

(1) 定点数表示法。

定点表示法就是小数点固定在某个位置上。在定点计算机中,一般将小数点定在最高位(即纯小数)或将小数点定在最低位(即纯整数)。

(2) 浮点数表示法。

浮点表示法就是小数点的位置不固定。浮点数在计算机中通常的表示形式为"浮点数=2 的正/负阶码次方 * 尾数",其中阶码是个正整数,尾数是个小数,我们规定尾数的区间为[0.5,1],如果尾数不在此区间,通过调节阶码来满足区间,此方法称为规格化。

5) 串数据

- 位串:一串连续的二进制数。
- 字节串:一串连续的字节。
- 字串:一串连续的字。
- 双字串:一串连续的双字。

6) ASCII 码数据

ASCII(American Standard Code for Information Interchange)美国信息交换标准码,包括 ASCII 码字符、ASCII 码数和 ASCII 控制符等。ASCII 码表详见附录 A。

7) 指针类数据

包括近程指针和远程指针。前者为 16 位的偏移量,用于段内的寻址;后者由一个 16 位段值和一个 16 位的偏移量组成,用于段间数据访问或段间转移。

另外,在 8086/8088 处理机中,为了达到数据结构的最大灵活性和最有效地使用内存,字不必定位于偶数地址,即允许不按 2 的整次边界对齐。但是,对于对齐的字可以一次传送,而未对齐的字则需要两次传送。为了获得最佳性能,可将字操作对齐于偶地址。

2. 8086 指令格式

1) 指令的操作码

指令的操作码(简称 OP)采用二进制代码表示本指令所执行的操作,在大多数指令的操作码中,常用某位指示某些具体操作信息。图 3.1 显示了 8086 操作码,含有 3 个特征位,分别为 W 位、D 位和 S 位。它们的含义如下:

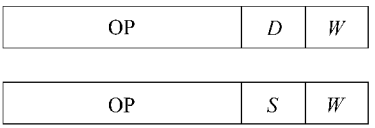

图 3.1 操作码格式

(1) W 位是字操作标志位。当 $W=1$ 时,表示当前指令进行字操作;当 $W=0$ 时,表示当前指令进行字节操作。

(2) D 位是对目的操作数进行寄存器寻址的标志。对于双操作数指令(立即数指令和串操作指令除外),其中一个操作数必定由寄存器指出。这时,寄存器名通过后面的 REG 域指出,此外,要用 D 位指出寄存器所寻址的是源操作数还是目的操作数,如 $D=1$,则寄存器所寻址的是目的操作数;如 $D=0$,则寄存器所寻址的是源操作数。

(3) S 位是符号扩展位。一个 8 位补码可以扩展为 16 位补码。扩展放大就是使所有高位等于低位字节的最高有效位,这种扩展叫符号扩展,在加法指令、减法指令和比较指令

中，S 位和 W 位结合起来指出各种情况。如 S=0、W=0 时，为 8 位操作数；S=0、W=1 时，为 16 位操作数。

2）指令格式

8086 的指令为 1～6B。一般用指令的第一个字节或者头两个字节表示指令的操作码和寻址方式，通常称为操作码域。操作码指出了执行这条指令时，CPU 要做什么操作，寻址方式则表示执行指令时所用的操作数的来源。图 3.2 显示了 8086 指令格式。

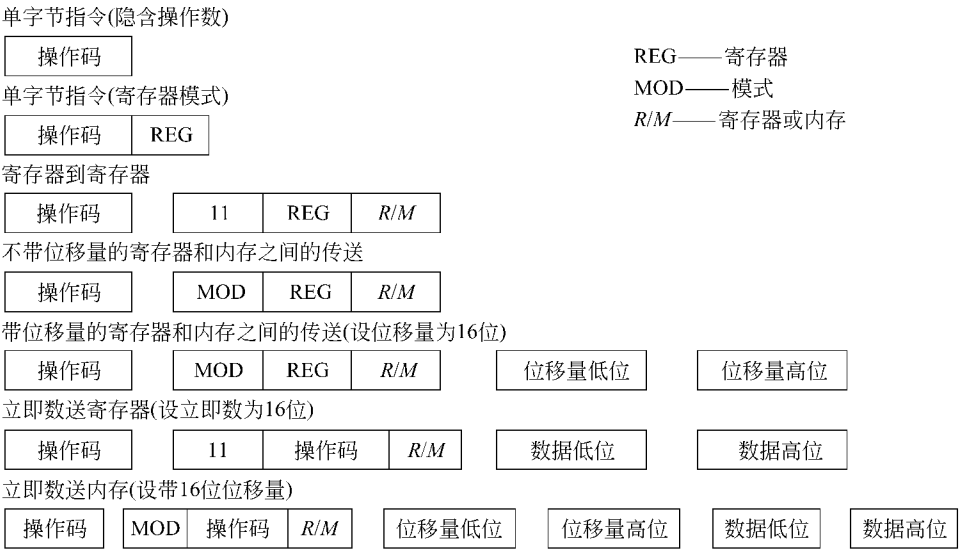

图 3.2　8086 指令格式

在操作码域后所跟的字节一般称操作数域。至于操作数域属于上述情况中的哪一种，要由指令前半部分的操作码和寻址方式来定。这里有几点需要指出：

（1）一条指令可以包含一个操作数，也可以包含一个以上的操作数；一个操作数的指令称为单操作数指令，单操作数指令中的操作数可能由指令本身提供，也可能由指令隐含地指出。

（2）若位移量或立即数为 16 位，那么在指令代码中，将低位字节放在前面，高位字节放在后面。

（3）8086 指令系统中大多数指令的操作码只占用第一个字节，但有几条指令是特殊的，其指令中的第一个字节不但包含操作码成分，而且还隐含地指出了寄存器名，从而整个指令只占一个字节，成为单字节指令。这些指令字节数最少，执行速度最快，用得也最频繁。

3. 指令的执行时间

指令执行时间取决于时钟周期长短和执行指令所需要的时钟周期数。如果涉及内存操作，那么执行一条指令的时间为基本执行时间加上计算有效地址所需要的时间。

8086 的总线部件和执行部件是并行工作的，因此在计算指令的基本执行时间时，都假定要执行的指令已经预先放在指令队列中，也就是说，没有计算取指时间。有些指令在执行过程中要多次访问内存，因此，访问内存的次数也是考虑指令执行总时间的重要因素。

可见执行一条指令所需的总时间为基本执行时间、计算有效地址的时间和为了读取操作数和存放操作结果需访问内存的时间之和。

3.2 8086的寻址方式和指令系统

3.2.1 8086的寻址方式

在计算机的存储器系统中,存储着两类信息,一是数据,二是程序。这里所说的程序就是指令。8086/8088的寻址方式包括程序寻址方式和数据寻址方式。数据寻址方式是指获取指令所需的操作数或操作数地址的方式。程序寻址方式是指在程序中出现转移和调用时的程序定位方式。

1. 数据寻址方式

8086指令中所需的操作数来自以下几个方面。

(1) 操作数包含在指令中。在取指令的同时,操作数也随之得到,这种操作数被称为立即数。

(2) 操作数包含在CPU的某个内部寄存器中。

(3) 操作数包含在存储器中。

在8086/8088微机系统中,任何内存单元的地址由段基址和偏移地址(又称偏移量)组成,段基址由段寄存器提供,而偏移地址由以下四个基本部分组成。

(1) 基址寄存器。

(2) 变址(间址)寄存器。

(3) 比例因子,采用2、4或8几种不同的比例因子。

(4) 位移量。

这四个部分被称为偏移地址的四元素。一般将这四个元素按某种计算方法组合形成的偏移地址称为有效地址(EA,Effective Address)。它们的组合和计算方法为

有效地址EA=基址+变址×比例因子+位移量

根据8086/8088系统的存储器组织方式,指令中不能使用实际地址,而只使用逻辑地址或称为操作数的线性地址,因此在求得有效地址EA后,可由有效地址和段地址形成逻辑地址。这四种元素可优化组合出9种存储器寻址方式。图3.3给出这四种元素的优化组合的寻址计算方法。

1) 立即寻址

8086/8088指令系统中,有一部分指令所用的8位或16位操作数由指令机器码提供,这种方式称为立即寻址方式,即操作数直接包含在指令机器码中,它紧跟在操作码的后面,与操作码一起放在代码区域中。

例3.1

```
MOV  AX,0A7FH              ;AX←0A7FH,执行后,AH=0AH,AL=7FH
MOV  AL,5H                 ;AL←5H
```

立即寻址方式主要是用于给寄存器或存储单元赋初值。因为操作数可以从指令中直接取得,不需要运行总线周期,所以立即数寻址方式的显著特点是执行速度快。

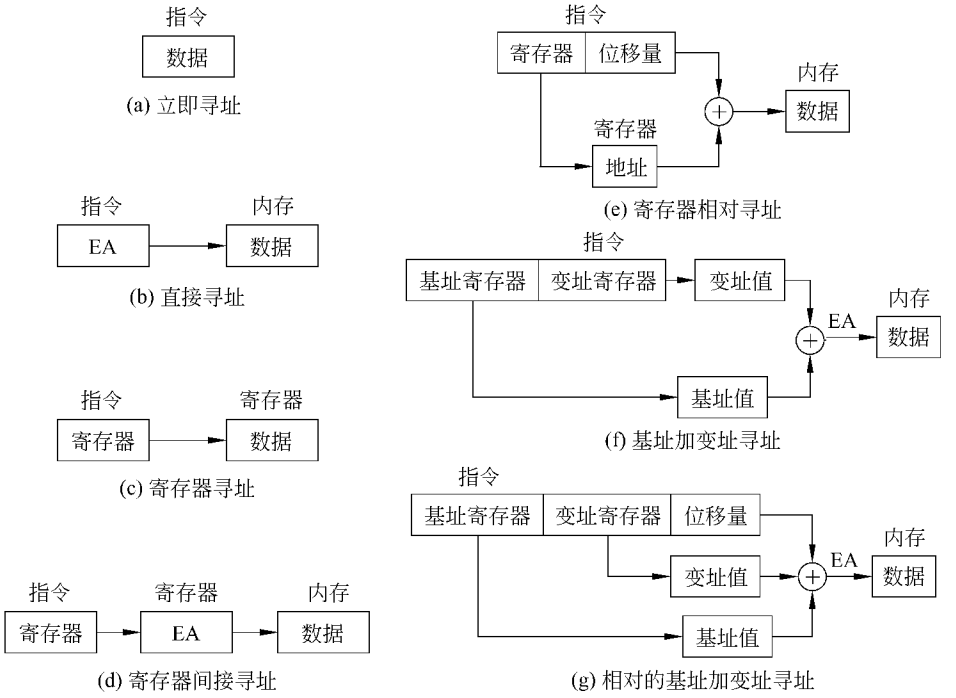

图 3.3　8086 的操作数寻址方式

2）直接寻址

直接寻址方式是在指令的操作码后面直接给出操作数的 16 位偏移地址，这个偏移地址也称有效地址，它与指令的操作码一起，存放在内存的代码段，也是低 8 位在前，高 8 位在后。操作数隐含数据段操作，存放在内存的数据段 DS 区域中，在给定的 16 位有效地址的地方。

例 3.2

```
MOV  AX,[2000H]
```

直接寻址指令中，对于表示有效地址的 16 位数，必须加上方括号。指令的功能不是将立即数 2000H 传送到累加器 AX 中，而是将一个内存单元的内容传送到 AX 中，该内存单元地址是 2000H。

假设该数据段寄存器 DS＝3000H，则内存单元的物理地址是 DS 的内容左移 4 位后再加上指令中的 16 位有效地址，形成有效访问内存的物理地址，即

```
3000H×10H+2000H=32000H
```

直接寻址一般多用于存取某个存储器单元中的操作数，比如从一个存储单元取操作数，或者将一个操作数存入某个存储器单元。

3）寄存器寻址

寄存器寻址是指操作数包含在 CPU 内部的寄存器中。对于 16 位操作数，寄存器可以是 AX、BX、CX、DX、SI、DI、SP、BP，而对于 8 位操作数来说，寄存器可以为 AH、AL、BH、BL、CH、CL、DH、DL。寄存器寻址的指令本身存放在存储器的代码段，而操作数则在 CPU

寄存器中。由于指令执行过程中不用访问存储器。因此执行速度很快。

寄存器寻址可以进行16位操作数操作，也可进行8位操作数操作。

例 3.3

```
MOV  AH,AL
```

4）寄存器间接寻址

寄存器间接寻址方式是指令中的操作数存放在存储器中，存储单元的有效地址由寄存器指出，这些寄存器可以是BX、BP、SI、DI之一，即有效地址等于其中某一寄存器的值，对寄存器指向的存储单元进行数据操作，这些用来存放存储器操作数偏移地址的寄存器称为地址指针。

(1) 以BX、SI、DI进行寄存器间接寻址(BX、SI、DI作为地址指针)的方式。

隐含的段寄存器为数据段寄存器DS，操作数存放在现行数据段中，将数据段寄存器DS的内容左移4位，再加上BX、SI或DI寄存器的内容便可得到操作数的物理地址。

例 3.4

```
MOV  AX,[BX]                    ;物理地址=DS×16+BX
MOV  BX,[SI]                    ;物理地址=DS×16+SI
MOV  [DI],DX                    ;物理地址=DS×16+DI
```

(2) 以BP进行寄存器间接寻址(BP作为地址指针)的方式。

隐含的段寄存器为堆栈段寄存器SS，操作数存放在堆栈段区域，将堆栈段寄存器SS的内容左移4位，再加上基址寄存器BP的内容，即为操作数的物理地址。

例 3.5

```
MOV  [BP],BX                    ;物理地址=SS×16+BP
```

无论用BX、SI、DI或者BP作为间接寄存器，都允许段超越，即可以使用上面所提到的约定以外的其他段寄存器。

例 3.6

```
MOV  AX,ES:[BX]                 ;物理地址=ES×16+BX
MOV  DS:[BP],DX                 ;物理地址=DS×16+BP
```

5）变址寻址

变址寻址方式是用SI、DI变址寄存器进行的间接寻址。操作数存放在存储单元中，操作数的物理地址是段寄存器的内容左移4位加上变址寄存器SI或DI的内容，再加上由指令中所指出的16位或8位位移量DISP。

例 3.7

```
MOV  AX,3003H[SI]
```

假设DS=3000H，SI=2000H，指令中的3003H即为位移量DISP。指令操作的存储器物理地址=3000H×10H+EA=30000H+2000H+3003H=35003H。

6）基址寻址

基址寻址与变址寻址类似，不同之处在于基址指令中使用基址寄存器BX或基址指针

寄存器 BP，而不用变址寄存器 SI 和 DI。

需要指出，当使用 BX 寄存器实现基址寻址时，隐含的段地址在 DS 寄存器中；当使用 BP 时，隐含的段地址在 SS 寄存器中。同样允许段超越。

基址寻址方式，操作数在存储单元中，操作数的物理地址是段寄存器的内容左移 4 位加上 BX 或 BP 的内容，再加上指令中指出的 16 位或 8 位位移量。

例 3.8

```
MOV  SI,08H[BX]                    ;物理地址= DS×16+ BX+ 08H
MOV  AX,[BX+100H]                  ;物理地址= DS×16+ BX+ 100H
MOV  AL,[BP+08H]                   ;物理地址= SS×16+ BP+ 08H
MOV  0200H[BP],AX                  ;物理地址= SS×16+ BP+ 0200H
```

7）基址、变址寻址

将基址寻址方式和变址寻址方式联合起来的寻址方式称为基址、变址寻址方式。这种寻址方式，操作数在存储单元中，其物理地址由段寄存器内容左移 4 位加上一个基址寄存器和一个变址寄存器，再加上 16 位或 8 位位移量。

例 3.9

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

假设 MASK＝64H，BX＝A500H，SI＝2200H，DS＝6000H

物理地址＝DS×10H＋EA＝6000H×10H＋A500H＋2200H＋64H＝6C764H

2. 程序寻址方式

程序在存储器中的寻址方式分为段内转移和段间转移。段内转移指转移地址与转移指令地址在同一段中，段间转移指目标地址与转移指令地址不在同一个段中。

1）直接寻址方式

段内直接寻址方式也称为相对寻址方式，是指把指令本身提供的位移量加到指令指针寄存器中，形成有效目标地址的寻址方式。

例 3.10

```
JMP   1000H                        ;转移地址在指令中给出
CALL  1000H                        ;调用地址在指令中给出
```

2）段内间接寻址方式

程序转移的地址存放在寄存器或存储单元中，这个寄存器或存储单元内容可用以上所述寄存器寻址或存储器寻址方式取得，所得到的转移有效地址用来更新 IP 的内容。由于此寻址方式仅修改 IP 的内容，所以这种寻址方式只能在段内进行程序转移。

例 3.11

```
JMP   BX                           ;转移地址由 BX 给出
CALL  AX                           ;调用地址由 AX 给出
JMP   WORD PTR [BP+TABLE]          ;转移地址由 BP+TABLE 所指的存储单元给出
```

3）段间直接寻址方式

这种寻址方式是在指令中直接给出 16 位的段基值和 16 位的偏移地址，用来更新当前

的 CS 和 IP 的内容。

例 3.12

```
JMP   2500H:3600H                ;转移的段地址和偏移地址在指令中给出
CALL  2600H:3800H                ;调用程序的段地址和偏移地址在指令中给出
```

4）段间间接寻址方式

这种寻址方式是由指令中给出的存储器数据寻址方式，包括存放转移地址的偏移量和段地址。其低位字地址单元存放的是偏移地址，高位字地址单元中存放的是转移段基值。这样既更新了 IP 内容又更新了 CS 的内容，故称为段间间接寻址。

例 3.13

```
JMP  WORD PTR[BX]                ;转移到当前代码位置内
                                 ;有效地址存放在 BX 寻址的单元中
```

3. I/O 地址空间

I/O 端口寻址是对输入输出设备的端口地址寻址，可分为两种形式，即直接端口寻址和间接端口寻址。

1）直接端口寻址

由指令直接给出端口地址，端口地址范围为 0～255。

例 3.14

```
IN  AL,32H                       ;32H 为 8 位端口地址
```

2）间接端口寻址

由 DX 寄存器指出端口地址，这种方式给出的端口地址范围为 0～65535。

例 3.15

```
IN  AL,DX                        ;DX 寄存器的内容为端口寻址
```

4. 段寄存器的确定

为了简化指令系统，8086/8088 的指令在形式上只给出了地址偏移值(有效地址)，而隐含着对某段寄存器的操作，表 3.1 给出的默认规定指出了所选用的隐含段寄存器。

表 3.1　段寄存器选择规定

访 存 类 型	默认段寄存器	段超越前缀的可用性
代码	CS	不可用
PUSH、POP 类代码	SS	不可用
串操作的目标地址	ES	不可用
以 BP、SP 间址的指令	SS	可用 CS、DS、ES
其他	DS	可用 CS、SS、ES

这种隐含选择规定可以被段超越前缀改变。当在一条指令前面加上段超越前缀时，则由段超越前缀所指定的段寄存器取代默认的段寄存器。

例 3.16

```
MOV  BX,ES:[DI]                  ;源操作数在 ES 指定的段中
```

3.2.2 8086 的指令系统

8086 的指令分为数据传送指令、算术运算指令、逻辑运算和移位指令、控制转移指令、串操作指令、程序控制指令和处理器控制指令等。

1. 数据传送指令

数据传送指令用于实现 CPU 的内部寄存器之间、CPU 内部寄存器和存储器之间、CPU 累加器 AX 或 AL 和 I/O 端口之间的数据传送，此类指令除了 SAHF 和 POPF 指令外均不影响标志寄存器的内容。需要注意的是，在数据传送指令中，源操作数和目的操作数的数据长度必须一致。

1) MOV 指令

指令格式：

```
MOV  dest,src                    ;dest←src
```

这里 dest 为目的操作数，src 为源操作数，以下类同。

指令功能：MOV 指令用于将一个操作数从存储器传送到寄存器，或从寄存器传送到存储器，或从寄存器传送到寄存器，也可以将一个立即数存入寄存器或存储单元，但不能用于存储器与存储器之间，以及段寄存器之间的数据传送，MOV 指令传送关系如图 3.4 所示。

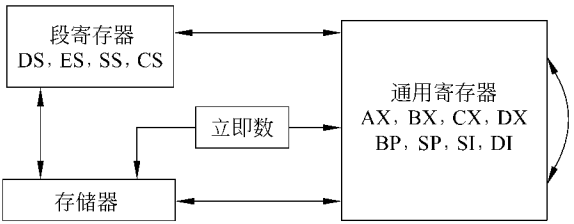

图 3.4　MOV 指令传送关系示意图

MOV 指令有 6 种数据传输格式。

(1) CPU 的通用寄存器之间的数据传输。

例 3.17

```
MOV  AL,BL                       ;BL 寄存器的 8 位数送到 AL 寄存器
MOV  SI,BX                       ;BX 寄存器的 16 位数送到 SI 寄存器
```

(2) 立即数(常数)到存储单元的数据传输。

例 3.18

```
MOV  MEM_BYTE,20H                ;将立即数 20H 送到 MEM_BYTE 存储单元
MOV  DS:[0005H],4500H            ;将立即数 4500H 送到 DS:0005H 存储单元
```

(3) 立即数到通用寄存器的数据传输。

例 3.19

```
MOV  AL,20H                      ;将立即数 20H 送到 AL 寄存器
MOV  SP,2000H                    ;将立即数 2000H 送入 SP 寄存器
```

(4) 通用寄存器和存储单元之间的数据传输。

例 3.20

```
MOV  AL,DS:[1000H]                ;将地址 DS:1000H 存储单元的内容送到 AL
MOV  ES:[0002H],BX                ;将 BX 中的 16 位数据传输到地址 ES:0002H
                                  ;所指的两个相邻存储单元中
```

(5) 段寄存器和存储单元之间的数据传输。

例 3.21

```
MOV  ES,[BX]                      ;BX 寄存器所指内容传送到 ES
MOV  [1000H],CS                   ;将 CS 内容传送到地址为 1000H 的存储器单元
```

(6) 通用寄存器和段寄存器之间的数据传输。

例 3.22

```
MOV  AX,ES                        ;将 ES 段地址传送到 DS 段
MOV  DS,AX
```

使用 MOV 指令时，必须注意以下几点：

- MOV 指令可以传 8 位数据，也可以传 16 位数据，这决定于寄存器是 8 位还是 16 位，或立即数是 8 位还是 16 位。
- MOV 指令中的目的操作数和源操作数不能都是存储器操作数，即不允许用 MOV 实现两个存储单元间的数据传输。
- 不能用 CS 和 IP 作为目的操作数，即这两个寄存器的内容不能随意改变。
- 不允许在段寄存器之间传输数据，例如 MOV DS,ES 是错误的。
- 不允许用立即数作为目的操作数。
- 不能向段寄存器传送立即数。如果需要对段寄存器赋值，可以通过 CPU 的通用寄存器 AX 来完成。

例 3.23

```
MOV  AX,1000H                     ;将数据段首地址 1000H 通过 AX 传入 DS 中
MOV  DS,AX
```

2) 交换指令 XCHG

指令格式：

```
XCHG  dest,src                    ;dest←→src
```

指令功能：XCHG 指令用于交换两个操作数。这条指令实际上起到了三条 MOV 指令的作用，指令中的两个操作数可以是两个寄存器操作数或一个寄存器与一个存储器操作数。

交换指令可实现通用寄存器之间、通用寄存器与存储单元之间的数据(字节或字)交换。

例 3.24

```
XCHG  AL,BL                       ;(AL)与(BL)寄存器的字节型数据进行交换
XCHG  BX,CX                       ;(BX)与(CX)寄存器的字型数据进行交换
XCHG  DS: [2200H],DX              ;(DL)与地址 DS:2200H 的内容进行交换
                                  ;(DH)与地址 DS:2201H 的内容进行交换
```