# 第 3 章　微型计算机指令系统

## 本章学习目标

本章着重介绍 8086/8088 的操作数的寻址方式和指令系统，它是了解 Intel 系列微处理器进行数据处理的基础。通过本章的学习，读者应该了解和掌握以下内容:

- 掌握 8086/8088 处理器的寻址方式。
- 掌握 8086/8088 处理器的指令系统。
- 了解 32 位新增指令。

目前的微型计算机都是基于冯·诺依曼原理，即存储程序控制原理工作的。程序和数据事先放在计算机的存储器中，计算机的工作就是从存储器中取出数据，并按照程序的要求对数据进行操作。此过程涉及 3 个术语：指令、程序和指令系统。

- 指令就是指定电子计算机执行某种操作（控制或运算）的命令。
- 完成一个任务的一组完整的指令序列，就是程序。
- 计算机所能执行的各类指令的总和，称为指令系统。

指令通常由操作码和操作数两大部分组成。操作码规定操作性质，操作数则指定“操作数”或“操作数的地址”。从指令中找到操作数的方法，就是操作数的寻址方式。

讨论 CPU 的指令系统和寻址方式，必须了解汇编语言知识。下面介绍 8086/8088 的寻址方式和指令系统。

## 3.1　8086/8088 处理器的寻址方式

寻址方式指的是确定指令涉及的“操作数”的来源和操作结果送到哪里去存放。它不仅与计算机硬件结构紧密相关，而且与使用的软件密切相关。现就 8086/8088 CPU 采用汇编语言常见的指令系统和基本寻址方式分别加以介绍。

除了指令涉及的操作数的寻址方式外，指令系统中还会遇到程序执行中发生指令转移或指令调用的情形，涉及转移地址或者调用地址的提供方式。因此，寻址方式可分为两种：操作数的寻址方式和程序转移地址的寻址方式。

### 3.1.1　与数据有关的寻址方式

1. 立即寻址

立即寻址（Immediate Addressing）又称为立即数寻址。在这种寻址方式下，操作数直接包含在指令中，它可以是 8 位或 16 位的常数，也叫立即数。立即数紧跟在指令操作码之后并和操作码一起存放在代码段中。使用立即寻址的指令常用来给寄存器赋初值。

汇编格式：n（n 为立即操作数）

功能：紧挨指令下一单元的内容为操作数 n。

【例 3-1】

```
MOV   AX,1234H      ;将立即数 1234H 值赋给寄存器 AX
                    ;请注意：操作数 n 存放在紧挨指令操作码的下一单元
ADD   AX,5678H      ;将 AX 中的数据与立即数 5678H 进行相加，其结果又赋给寄存器 AX
MOV   AX,'CB'       ;将 C 字符的 ASCII 码值“43H”送入 AH 寄存器中，将 B 字符的 ASCII
                    ;码值“42H”送入 AL 寄存器中
```

如图 3-1 所示。

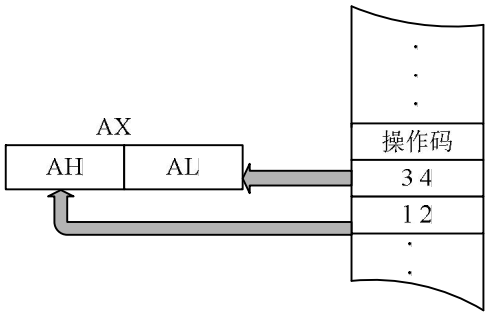


图 3-1　立即寻址方式示意图

2. 直接寻址

与立即寻址不同，为了获取指令所指定操作的操作数，直接寻址（Direct Addressing）必须访问操作数的物理地址[PA]，它是通过段首址左移 4 位加偏移地址 EA = n 得到的，n 是某个常数。

汇编格式：[n]。这里[n]表示操作数在某内存单元中，该内存单元默认在数据段中，其偏移地址 EA = n。

功能：指明操作数的偏移地址 n = EA。

操作数是指令的一部分，但操作数通常默认在数据段中，其偏移地址是 EA，如例 3-2 中 n =EA = 2000H。

【例 3-2】

```
MOV   AL,[2000H]        ;将逻辑地址为 DS:2000 单元内的字节送入 AL
```

若段基址(DS)= 4000H，则操作数的物理地址为段基址左移 4 位，即 40000H，再加上偏移地址[EA]。此指令的操作是：将数据段中物理地址为 42000H 单元的内容 56H 传送至 AL 寄存器。

直接寻址方式示意如图 3-2 所示。PA =(段首址)×10H+EA。

说明：

1）当用一个常量作为操作数的偏移地址时，为了防止与立即寻址相混淆，必须给常量加一对中括号。

2）直接寻址的汇编格式中，操作数默认存放在数据段中，也可存放在其他段中。若在其他段中，则应在[n]前面注明。如例 3-2 中的[2000H]改在扩展段 ES，则该指令应改写成

```
MOV   AL,ES:[2000H]
```

并告知(ES)的值。

3）如果已先行定义某变量存放在数据段中，该变量的偏移地址（称为符号地址）已知，

则可以直接将该变量的代号当操作数使用，如例 3-3 所示。同理，若在其他段中，则应注明，如 ES:BUF。

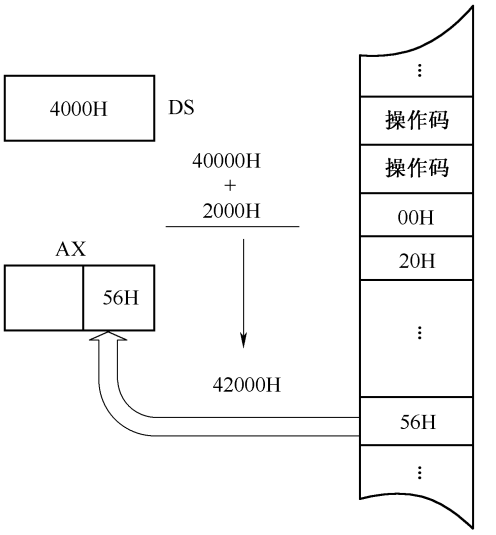


图 3-2　直接寻址方式示意图

【例 3-3】 设 BUF 为数据段定义的变量，其偏移地址为 3000H，(DS)= 4000H，(43000H)= 3469H，问执行指令 MOV AX,BUF 后的结果。

答：该指令中 BUF 提供的是参与指令操作的操作数的偏移地址（3000H），由于操作数的物理地址 = 段首址（4000H）×10H（左移 4 位）+偏移地址（3000H），其计算出的操作数的物理地址为 43000H，而该物理地址中的操作数为 3469H，也就是说这条指令的作用是：将物理地址为 43000H 中的数 3469H 赋给寄存器 AX。

3. 寄存器寻址

参与指令所指定操作的操作数就存放在指定的寄存器中，因此要进行寄存器寻址（Register Addressing）。

汇编格式：R（R 是寄存器名）

功能：寄存器 R 的内容就是操作数。

【例 3-4】

```
MOV     BX,0201H        ;将立即数 0201H 放进 BX 寄存器中
MOV     AX,BX           ;将寄存器 BX 的内容送入 AX 中
```

这两条指令运行的结果是：先将立即数 0201H 放进 BX 寄存器中，再将寄存器 BX 的内容 0201H 送入 AX 中。第一条指令的源操作数是立即寻址，第二条指令的源操作数才是寄存器寻址。

说明：

1）在寄存器寻址方式中，操作数存放在指令规定的寄存器中，不需访问内存，工作效率高。

2）对于 16 位操作数，寄存器可以是 AX、BX、CX、DX、SI、DI、SP 或 BP；而对 8 位操作数，寄存器可以是 AH、AL、BH、BL、CH、CL、DH 或 DL。

寄存器寻址方式示意如图 3-3 所示。

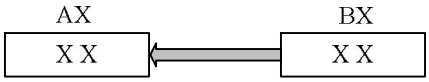


图 3-3　寄存器寻址方式示意图

4. 寄存器间接寻址

寄存器间接寻址（Register Indirect Addressing）与寄存器寻址的不同之处在于，指令指定的寄存器中的内容不是操作数，而是操作数的偏移地址，偏移地址加上左移 4 位之后的段首址得到操作数的物理地址，参与指令所指定的操作的操作数就在这个物理地址中。

汇编格式：[R]（R 是寄存器名）

功能：R 的内容为操作数所在内存的偏移地址 EA。

【例 3-5】 设(DS)= 2500H，(SS)= 3000H，(BX)= 1000H，(BP)= 2000H，(26000H)= 4321H，(32000H)= 8765H。问执行以下指令后的结果：

```
MOV AX,[BX]
MOV CX,[BP]
```

答：第一条指令中寄存器 BX 提供的是操作数的偏移地址（1000H），加上左移 4 位之后的数据段的首地址得到操作数的物理地址，即 PA=2500H*10H + 1000H = 26000H，然后从 26000H 物理地址中取出操作数 4321H 赋给寄存器 AX。

第二条指令中寄存器 BP 提供的也是操作数的偏移地址（2000H），加上左移 4 位之后的堆栈段的首地址得到操作数的物理地址，即 PA = 3000H*10H+2000H = 32000H，然后从 32000H 物理地址中取出操作数 8765H 赋给寄存器 CX。

说明：

1）操作数存放在存储器中。

2）指令指定的寄存器只能为基址寄存器 BX、BP 或变址寄存器 SI、DI。

3）如果指令指定的寄存器为 BX、SI 和 DI，则操作数在数据段（DS）中；如果指令指定的寄存器为 BP，则操作数在堆栈段（SS）中。

4）寄存器间接寻址和寄存器寻址在汇编格式上相比较多了一个中括号，它们的寻址方式截然不同，寄存器寻址不需访问内存，操作数就在指令指定的寄存器中，而寄存器间接寻址需要访问内存，操作数的偏移地址 EA 就是寄存器的内容。

请注意：3）、4）两点也适用于以后所讲的各种寻址方式中。

寄存器间接寻址示意如图 3-4 所示。

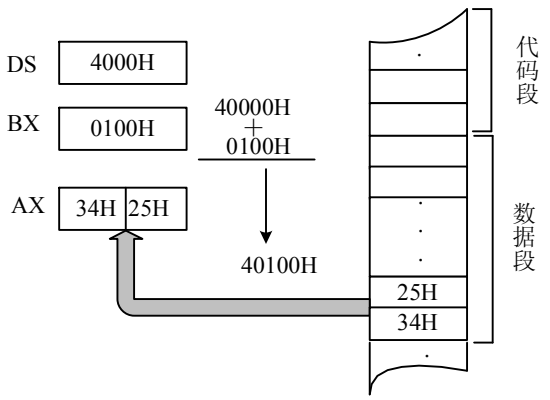


图 3-4　寄存器间接寻址示意图

5. 直接变址寻址

直接变址寻址（Indexed Addressing），或称寄存器相对寻址。与寄存器间接寻址的不同之处在于，指令指定的寄存器中的内容不是操作数，也不是偏移地址，操作数的偏移地址是由指令指定的寄存器的内容加上指令指定的位移量之和求得。其他与寄存器间接寻址方式相同。

汇编格式：X[R] 或 [R+X]（其中 X 为位移量，R 为寄存器名）

功能：寄存器 R 中的内容加位移量 X 作为操作数的偏移地址。

【例 3-6】设(SI)= 0100H，(BP)= 0200H，(DS)= 1000H，(SS)= 2000H，(10108)= 1234H，(20206)= 5432H。问执行以下指令后的结果：

```
MOV   AX,8[SI]
ADD   6[BP],AX
```

答：第一条指令中的 8[SI]也可以写成[SI+8]。第一条指令说明由变址寄存器 SI 提供的 0100H 值加上位移量 8 才是操作数的偏移地址 EA=0100H+8H = 0108H，加上左移 4 位之后的数据段首地址得到操作数的物理地址，其计算过程为 PA=1000H×10H+0108H = 10108H，然后在操作数的物理地址当中取出 1234H 操作数赋给寄存器 AX。

第二条指令中基址寄存器 BP 提供的 0200H 值加上位移量 6 才是操作数的偏移地址：EA = 0200H + 6H = 0206H，加上左移 4 位之后的堆栈段首地址得到操作数的物理地址，其计算过程 PA = 2000H ×10H + 0206H = 20206H，然后在操作数的物理地址中取出 5432H 与 AX 的内容 1234H 相加，结果再放到 20206H 内存单元中。

说明：

1）操作数在存储器中，位移量为 8 位或 16 位二进制补码表示的有符号数。

2）指令指定的寄存器只能为基址寄存器 BX、BP 或变址寄存器 SI、DI。

3）在变址寻址中，指令指定的寄存器一定要加[]括号。

直接变址寻址方式示意如图 3-5 所示。

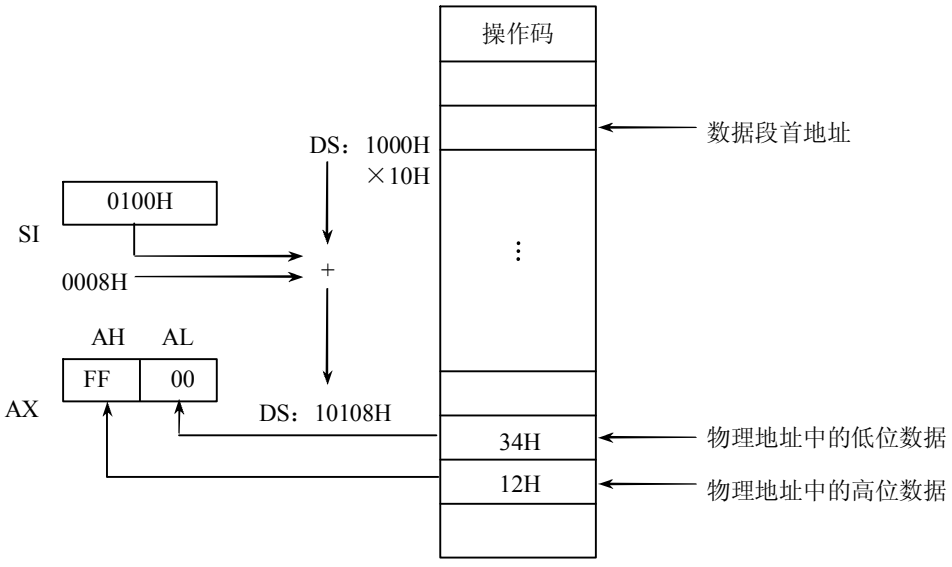


图 3-5　直接变址寻址方式示意图

6. 基址变址寻址

有效地址 EA 是基址寄存器 BX（或基址指示器 BP）的内容与变址寄存器（DI 或 SI）的

内容之和，这就需要用到基址变址导址（Based Indexed Addressing）。

汇编格式：[BX+DI]或[BX] [DI]（其中 BX 为基址寄存器，DI 为变址寄存器）

功能：寄存器 BX（或 BP）中的内容加寄存器 DI（或 SI）的内容作为操作数的偏移地址。

【例 3-7】 MOV AX,[BX+SI];BX 的内容与 SI 的内容之和作为操作数的有效地址。传送数据段中的一个字，如图 3-6 所示。

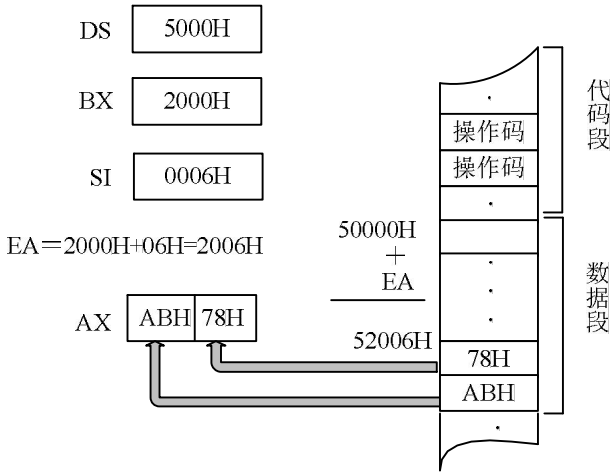


图 3-6　基址变址寻址方式示意图

7. 相对基址加变址寻址

相对基址加变址寻址（Relative Based Indexed Addressing）方式中，操作数的偏移地址 EA 是一个基址寄存器（BX 或 BP）和一个变址寄存器（SI 或 DI）的内容再加上指令中指定的 8 位或 16 位位移量之和。

汇编格式：X[BX+DI]或 X[BX] [DI] 或[BX+DI+X]（BX（或 BP）为基址寄存器，DI（或 SI）为变址寄存器，X 为位移量）

功能：寄存器 BX（或 BP）中的内容加寄存器 DI（或 SI）的内容再加位移量 X 作为操作数的偏移地址。

【例 3-8】 设{AX} = 0100H，(BX)= 0200H，(BP)= 0400H，(SI)= 0300H，(DS)= 2000H，(SS)= 3000H，(20506H)= 4567H，(30706H)= 1234H。问执行以下指令后的结果。

```
MOV AX,6[BX+SI]
ADD 6[BP+SI],AX
```

答：第一条指令中的基址寄存器 BX 提供的 0200H 值加上变址寄存器 SI 提供的 0300H 值再加上位移量 6 才是操作数的偏移地址 EA = 0200H + 0300H + 6H = 0506H，再将数据段 DS 首地址左移 4 位，之后加上 EA，得到操作数的物理地址 PA = 2000H×10H + 0506H = 20506H，然后在 20506 地址中读取相应操作数参与指令所指定的操作。

第二条指令中的基址寄存器 BP 提供的 0400H 值加上变址寄存器 SI 提供的 0300H 再加上位移量 6 才是操作数的偏移地址 EA = 0400H + 0300H + 6H = 0706H，偏移地址加上左移 4 位之后的堆栈段 SS 首地址得到操作数的物理地址 PA = 3000H * 10H + 0706H = 30706H，然后在 30706 地址中读取相应操作数参与指令所指定的操作。

说明：

1）操作数在存储器中，位移量为 8 位或 16 位二进制补码表示的有符号数。

2）该寻址方式的基址寄存器只能选择 BX 或 BP，变址寄存器只能用 SI 或 DI。

3）在该寻址方式中指令指定的基址寄存器和变址寄存器一定要用[]括号将两者包含在一起。

相对基址加变址寻址示意如图 3-7 所示。

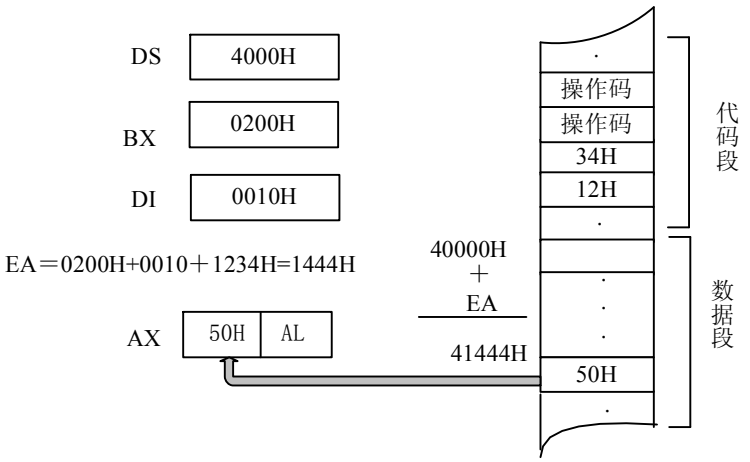


图 3-7　相对基址加变址寻址方式示意图

### 3.1.2　程序转移地址的寻址方式之一 ——JMP 无条件跳转指令

为了改变程序中指令执行的顺序，可以利用控制转移指令改变代码段段基址 CS 和指令指针 IP 的值来实现。换句话说，控制转移指令说明，程序执行顺序要从当前位置跳到新的位置。新位置的地址由（CS）和（IP）的值指明：如果只在本代码段内转移，只要知道（IP）的值即可；如果要跳到另一个代码段去，就要同时知道（CS）和（IP）两个值，（CS）指明跳到那个代码段的段基址，（IP）指明新位置在那个代码段内的偏移地址。（CS）与（IP）的值无需用户直接在汇编指令中给出，用户只需在汇编指令中给出某个符号地址即可，当然，该符号地址是事先定义过的。机器在执行指令时，指令转移的位移量由汇编或编译程序根据目标地址和转移指令的距离自动计算得出。

8086 CPU 提供了无条件转移指令和条件转移指令、过程调用指令、循环控制指令及中断等几类指令。这些指令就是程序转移地址的寻址方式。为了知识的系统性，只讨论无条件转移指令涉及的寻址方式，其他内容将在 3.2.6 小节讨论。

1. 段内直接寻址

段内直接寻址方式也称为相对寻址方式，相对当前 IP 值的转移距离不超过 8 位或 16 位的位移量，所以叫相对寻址，如图 3-8 所示。

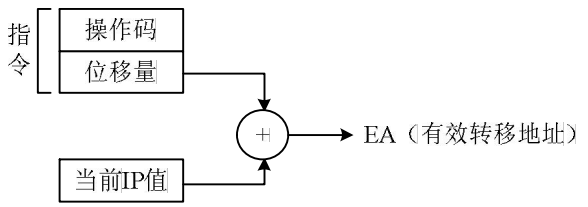


图 3-8　段内直接寻址方式示意图

（1）段内直接短转移。

格式：JMP　SHORT　OPR；OPR 表示新指令操作码的目标地址，用符号地址表示

功能：指令在同一代码段范围之内进行转移，（IP）= OPR。

执行的操作：(IP)←OPR

说明：SHORT 指明符号地址与当前 IP 的距离不大于 8 位位移量。

【例 3-9】

```
JMP   SHORT    QUEST    ;QUEST 为一转向符号地址，处于代码段内
```

（2）段内直接近转移。

格式：JMP　NEAR　PTR　OPR

功能：与前一种转移基本相同，只是位移量为 16 位。

说明：NEAR　PTR　指明符号地址与当前 IP 的距离大于 8 位不大于 16 位位移量。

【例 3-10】

```
JMP   NEAR    PTR    ADD1        ;ADD1 为一转向符号地址，处于代码段内
```

符号地址 ADD1 是事先定义过的，与当前 IP 的位移量为 16 位的二进制补码。

2. 段内间接寻址

格式：　JMP　R　　;R 表示寄存器

或　　JMP　WORD　PTR　OPR

功能：段内转移。

执行的操作：(IP)←(EA)

说明：转移目标的偏移地址 EA 采用寄存器寻址方式，或寄存器间接寻址方式，或寄存器相对寻址方式。即其转移的目标地址的偏移量，是寄存器或存储单元的内容，即以寄存器或存储器单元内容来更新 IP 的内容，所以是绝对偏移量。这种方式不能用于条件转移指令。如图 3-9 所示。这种方式也称段内间接转移。

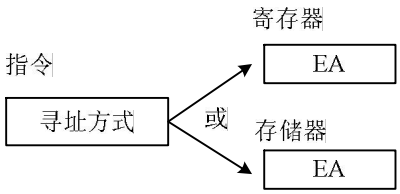


图 3-9　段内间接寻址方式示意图

【例 3-11】 设(DS)= 2000H，(BX)= 1256H，(SI)= 528FH，(BP) = 0100H，(SS)= 3000H，位移量= 20A1H，(232F7H)= 3280H，(264E5H)=2450H，(321A1) = 3487H。问执行以下指令后的结果：

```
JMP   BX
JMP   WORD    [BX]
JMP   WORD    PTR [BP+TABLE]
```

答：第一条指令是将寄存器 BX 的内容（默认在数据段内）作为新的 EA 值，送给 IP 寄存器，即

(IP) =　(BX) = 1256H

第二条指令是将存储单元的内容作为新的 IP 值，该存储器单元的地址由寄存器给出，即

(IP) = ((DS)×10H + (BP) +位移量) = (20000 + 1256 + 20A1) = (232F7) = 3280H

第三条指令也是将存储单元的内容作为新的 IP 值，该存储器单元的地址由寄存器 BP 的内容加位移量 TABLE 的值（默认在堆栈段内）给出，再将该存储单元的内容取出，作为新的 IP 值。即

(IP) = ((SS)×10H + (BP) +位移量) = (30000 + 0100 + 20A1) = (321A1) = 3487H

其中 WORD PTR 为操作符，用以指出寻址方式所取得的转向地址是一个字（16 位）范围内的偏移地址，也就是说它是一种段内转移。

3. 段间直接寻址

格式：JMP FAR PTR OPR

功能：段间直接转移。

执行的操作：(IP)←(OPR 的段内偏移地址)

(CS)←(OPR 所在段的段地址)

说明：这种方式用于段间直接转移，目标转向存储单元地址的段基值（CS）和偏移地址（IP），它们都是指令码的组成部分，用来更新当前 CS 和 IP 的内容，如图 3-10 所示。

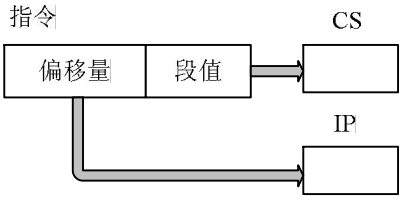


图 3-10 段间直接寻址方式示意图

【例 3-12】

```
JMP  FAR  PTR    NEXTADD      ;NEXTADD 是另一代码段的符号地址
```

4. 段间间接寻址

格式：JMP DWORD PTR OPR

功能：段间间接转移。

执行的操作：(IP)←(EA)

(CS)←(EA+2)

说明：这种方式用于段间间接转移，只不过当前 CS 和 IP 由目标存储单元中连续的两个字更新，低位地址的字更新 IP，高位地址的字更新 CS，存放新 IP 和 CS 的存储单元地址由前述存储器操作数的寻址方式决定，如图 3-11 所示。

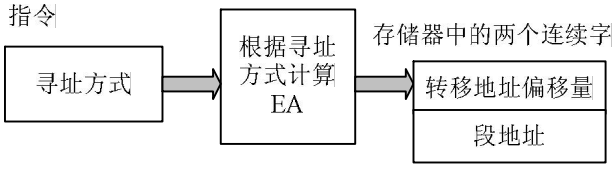


图 3-11 段间间接寻址方式示意图

【例 3-13】

```
JMP  DWORD PTR  [INTER+BX]  ;取 DS 段中偏移为存储器[INTER+BX]处的双字作为
                            ;新的 CS 和 IP
```

# 3.2　8086/8088 处理器的指令系统

8086/8088 指令按功能不同分类，可以分为数据传送指令、算术运算指令、逻辑运算和移位指令、控制转移指令及处理器控制指令，下面就对各类常用的指令做详细介绍。

## 3.2.1　数据传送指令

作用：它们在存储器和寄存器、寄存器和输入/输出端口之间传送数据。

分类：数据传送意为对数据传送的操作，又可以分为传送指令、交换指令、地址传送指令、堆栈操作指令、标志传送指令和输入/输出指令等。

1．传送指令

格式：MOV DST,SRC（DST 为目的操作数，SRC 为源操作数）

功能：该指令把一个字节或一个字从 SRC 送到 DST。

【例 3-14】

```
MOV   AH,AL          ;寄存器之间传送
MOV   AL,3           ;立即数与寄存器之间传送
MOV   AX,[DI]        ;寄存器与存储器之间传送（寄存器间接寻址）
```

说明：

1）源操作数可以是累加器、寄存器、存储单元或者是立即数。

2）目的操作数可以是累加器、寄存器和存储单元。

3）MOV 指令不改变 SRC。

2．交换指令

格式：XCHG OPRD1,OPRD2（OPRD 为操作数）

功能：该指令把 OPRD1 的内容与 OPRD2 的内容交换。

【例 3-15】

```
CHG   [SI+3],AL      ;存储器与寄存器之间交换数据
XCHG   DI,BX         ;寄存器之间交换数据
```

说明：OPRD1 和 OPRD2 可以是通用寄存器和存储单元，但不包括段寄存器，也不能同时为存储单元，不能包含立即数。

3．地址传送指令

地址传送指令又有 3 条指令。

（1）LEA 装入有效地址指令。

格式：LEA　REG,OPRD（REG 为寄存器，OPRD 为操作数）

功能：该指令把操作数 OPRD 的有效地址传送到 REG 寄存器中。

【例 3-16】

```
LEA   AX,[BX+3]      ;将操作数的有效地址送入寄存器 AX
LEA   DX,BUFFER      ;BUFFER 为变量名
```

说明：

1）OPRD 必须是一个存储器操作数。

2）REG 必须是一个 16 位通用寄存器。

（2）LDS 传送目标指针，把指针内容装入 DS 指令。

格式：LDS REG,OPRD（REG 为寄存器，OPRD 为操作数）

功能：该指令把操作数 OPRD 中包含的 32 位地址指针段值部分送到数据段寄存器 DS，把偏移部分送到通用寄存器 REG。

【例 3-17】

```
LDS DI,[BX]
LDS SI,FARPOINTER        ;FARPOINTER 是一个双字变量
```

说明：

1）REG 表示除段寄存器之外的 16 位操作数。

2）OPRD 表示双字的各种寻址方式的存储器操作数的首地址。

（3）LES 传送目标指针，把指针内容装入 ES 指令。

格式：LES REG,OPRD

功能：该指令把操作数 OPRD 中包含的 32 位地址指针的段值部分送到附加段寄存器 ES，把偏移部分送到通用寄存器 REG。

说明：

1）REG 表示除段寄存器之外的 16 位操作数。

2）OPRD 表示双字的各种寻址方式的存储器操作数的首地址。

4. 堆栈操作指令

在第 1 章中曾经简单介绍过堆栈。堆栈是指存储器中开辟的一个特殊存储区，称为堆栈段。其主要用来临时存放数据备用；也常用在执行中断指令或调用子程序 CALL，用堆栈来保存返回地址。所以堆栈在程序运行时是必不可少的。堆栈有进栈（存入数据）和出栈（取出数据）两种操作，编程时常成对使用。向堆栈存取数据必须以字为单位进行。堆栈就像一只带有标志（用堆栈指针寄存器 SP）的桶，标志总在货物的上方。最先向堆栈存入数据时是从栈底开始的，此时堆栈指针寄存器 SP 指向该存储区最高地址单元，即栈底的地址。以后，每存入一个字（2B）的数据，堆栈指针寄存器 SP 的内容自动减二；每取出一个字的数据，堆栈指针寄存器 SP 的内容自动加二；总之，数据进栈时 SP 指示地址递减，数据出栈时 SP 指示地址递增。每进行一次堆栈操作都会及时修改堆栈指针 SP 寄存器的内容，使得堆栈指针寄存器 SP 在任何时候都指向当前的栈顶（开始时指向栈底，即栈底单元的地址加 2）。

因为堆栈只有一个出入口，所以堆栈是以“先进后出”的方式进行数据操作的，即后进栈的数据先出栈，先进栈的数据后出栈。堆栈数据操作指令中只出现一个操作数，另一个操作数隐含在堆栈段中。如果是入栈操作，那么指令中出现的仅是源操作数，目的操作数隐含在堆栈中；如果是出栈操作，那么指令中出现的仅是目的操作数，源操作数则隐含在堆栈中。操作数可以是寄存器（16 位）、段寄存器或存储器操作数。

换成专业语言表述，堆栈是只允许在一端进行数据插入和数据删除操作的线性表，它是一段 RAM，其中地址最大的为栈底，地址最小的为栈顶（进行数据插入和删除操作的一端）。堆栈的段基址存放在段寄存器 SS 中，指针 SP 始终指向栈顶。

栈的操作遵循先进后出的原则。

（1）PUSH 把字压入堆栈指令。

格式：PUSH SRC（SRC 为源操作数）

功能：该指令把源操作数 SRC 压入堆栈，SP 随着压栈而减小。

【例 3-18】

```
PUSH SI
PUSH DS
PUSH [SI]
```

说明：数据进入堆栈时遵守“高高低低”的原则，即高位数据放在高字节中，低位数据放在低字节中。

（2）POP 把字弹出堆栈指令。

格式：POP DST（DST 为目的操作数）

功能：该指令从堆栈弹出一个字数据到目的操作数 DST，SP 随着出栈而增大。

【例 3-19】

```
POP [SI]
POP ES
POP SI
```

说明：DST 可以是通用寄存器及段寄存器（除 CS），也可以是字存储单元。

5. 标志传送指令

8086/8088 CPU 中有专门用于标志寄存器的指令。

（1）LAHF 标志位送 AH 指令。

格式：LAHF

功能：该指令把标志寄存器低 8 位（SF、ZF、AF、PF、CF）传送到寄存器 AH 的指定位（即 7、6、4、2、0）。

（2）SAHF 将 AH 送入标志寄存器指令。

格式：SAHF

功能：该指令把寄存器 AH 的指定位传送到标志寄存器的低 8 位（即该指令为 LAHF 逆操作）。

（3）PUSHF 标志寄存器进栈指令。

格式：PUSHF

功能：该指令把标志寄存器的内容压入堆栈。该指令不影响标志。

（4）POPF 标志寄存器出栈指令。

格式：POPF

功能：该指令把当前堆栈的一个字传送给标志寄存器，同时 SP 加 2，该指令影响对应的标志位。

6. 输入/输出端口传送指令

（1）IN I/O 端口输入指令。

格式 1：IN　AL,端口地址　或　　IN　AX,端口地址

功能：将 8 位端口的内容读入一个字节到 AL 寄存器中，或将 8 位端口的内容读入一个字到 AX 寄存器中。此指令只限于端口地址处于 0～255 之间时，是直接寻址方式。执行指令 IN AX,端口地址的结果一定是：AL←端口地址的内容；AH←端口地址+1 的内容。

格式 2：

```
MOV   DX,端口地址    ;端口地址先送入 DX 中
IN    AL,DX          ;将端口地址的内容送 AL 寄存器中
```

或

```
MOV   DX,端口地址    ;端口地址先送入 DX 中
```

```
IN      AX,DX              ;将端口地址的内容送 AX 寄存器中
```

功能：当从 16 位端口地址（其地址大于 255）传送数据时，应采用间接寻址方式。即先将端口地址送 DX，再将 DX 所指端口内容送 AX 寄存器中。如

```
MOV   DX, 0908H
IN      AX, DX                ;先将端口地址送 DX，再将 DX 所指端口内容送 AX 寄存器中
```

【例 3-20】 IN 指令中也可使用符号来表示地址。例如，下面指令从一个模/数（A/D）转换器读入一个字节的数字量到 AL 中：

```
ATOD   EQU   54H          ;A/D 转换器端口地址为 54H
IN       AL,ATOD          ;将 54H 端口的内容读入 AL 中
```

（2）OUT I/O 端口输出指令。

上面对于输入指令 IN 的讨论，也适用于 OUT I/O 端口输出指令。

格式 1：OUT　端口地址,AL　或　OUT　端口地址,AX

功能：将 AL 中的一个字节写到一个 8 位端口，或把 AX 中的一个字写到一个 16 位端口。

格式 2：OUT　DX,AL　　;DX=端口地址　或　OUT　DX,AX

指令功能：将 AL 中的一个字节写到端口，或把 AX 中的一个字写到端口。同样，对 16 位端口进行输出操作时，也是对两个连续的 8 位端口进行输出操作。

【例 3-21】下面是几个用 OUT 指令对输出端口进行操作的例子：

```
OUT   85H,AL              ;8 位 85H 端口←AL 内容

MOV   DX,0FF4H            ;16 位端口地址送 DX
OUT   DX,AL               ;FF4H 端口←AL 内容

MOV   DX,300H             ;DX 指向 300H
OUT   DX,AX               ;300H 端口←AL 内容
                          ;30lH 端口←AH 内容
```

### 3.2.2　算术运算指令

算术运算指令完成对数值的加、减、乘、除等运算。

1. 加法指令

（1）ADD 加法指令。

格式：ADD DST,SRC（DST 为目的操作数，SCR 为源操作数）

功能：将 DST 内容与 SRC 内容相加，结果存入 DST 中，SRC 内容不变。

【例 3-22】执行以下指令：

```
MOV AX,1234H              ;类似于(AX)= 1234H
MOV BX,2211H              ;类似于(BX)= 2211H
ADD AX,BX                 ;类似于(AX)=(AX)+(BX)
```

说明：

1）当 SRC 是立即数或寄存器操作数时，DST 可以是寄存器或存储器操作数。

2）当 SRC 是存储器操作数时，DST 只能是寄存器操作数。

3）段寄存器操作数不能为 SRC 和 DST。

4）该指令会影响 AF、OF、PF、SF 和 ZF 标志位。

（2）ADC 带进位加法指令。

格式：ADC　DST,SRC（DST 为目的操作数，SRC 为源操作数）

功能：将 DST 内容加上 SRC 内容再加上 CF 进位标志，并将结果送入 DST 中。

【例 3-23】设 CF=0，则执行以下指令：

```
MOV   AX,4653H          ;类似于(AX)= 4653H
ADD   AX,0F0F0H         ;类似于(AX)=(AX)+ 0F0F0H（CF = 1）
MOV   DX,0234H          ;类似于(DX)= 0234H
ADC   DX,0F01H          ;类似于(DX)=(DX)+0F01 +(CF)（CF = 0）
```

说明：

1）该指令主要用于多字节（或多字）加法运算中。

2）当 CF = 0 时可以用 ADD，当 CF = 1 时必须用 ADC。

3）该指令会影响 AF、OF、PF、SF、ZF 标志位。

（3）INC 加 1 指令。

格式：INC OPR

功能：将 OPR 的内容自动加 1 之后，结果存入原地址。

【例 3-24】设(AX)= 011FFH，则执行指令：

```
INC   AX        ;类似于(AX)=(AX)+1
```

说明：

1）INC 指令是一个单操作数指令，即操作对象只有一个。

2）该指令中的操作数只能是寄存器或存储器操作数。

3）INC 指令不会影响 CF 标志位。

4）该指令常用做计数器和对地址指针进行调整。

2. 减法指令

（1）SUB 不带借位的减法指令。

格式：SUB DST,SRC

功能：将 DST 的内容减 SRC 的内容，结果存于 DST 中。

【例 3-25】设(DS)= 3000H，(SI)= 0050H，(30064)= 4336H，则执行指令：

```
MOV   AX,0136H（1）          ;类似于(AX)= 0136H
SUB   [SI+14H],AX（2）       ;类似于 [SI+14] = [SI+14]-AX
```

分析：第一条指令的作用是将立即数 0136H 赋给 AX 寄存器。

第二条指令的作用是先求出操作数的偏移地址 EA = 0050H +14H = 0064H，再求出操作数的物理地址 PA = 3000H × 10H + 0064H = 30064H，然后将 30064H 地址中的 4336H 操作数减 AX 寄存器中的 0136H 操作数，即 4336H−0136H = 4200H，最后将结果再存入 30064 物理地址中。

说明：

1）在完成数据减法操作时，如果没有借位可以使用 SUB 指令。

2）DST 可以是寄存器或存储器，而 SRC 还可以是立即数。

3）该指令影响 AF、OF、PF、SF、ZF 和 CF 标志位。

（2）SBB 带借位减法指令。

格式：SBB　DST,SRC

功能：将 DST 的内容减 SRC 的内容再减 CF，结果存于 DST 中。

【例 3-26】设 CF=0，DSUB 为定义的双字变量，初值为 0，则执行指令：

```
MOV     AX,3412H          ;类似于(AX)= 3412H
SUB     AX,2F65H          ;类似于(AX)= 3412H－2F65H（CF = 1）
MOV     DSUB,AX           ;将 AX 内容送到 DSUB 低位字中
MOV     BX,4275H          ;类似于(BX)= 4275H
SBB     BX,12A5H          ;类似于(BX)= 4275－12A5H－CF
MOV     DSUB+2,BX         ;将 BX 内容送到 DSUB 高位字中
```

说明：

1）SBB 指令主要用于双字减法操作。

2）DST 可以是寄存器或存储器，而 SRC 还可以是立即数。

3）该指令影响 AF、OF、PF、SF、ZF 和 CF 标志位。

（3）DEC 减 1 指令。

格式：DEC　OPR

功能：将 OPR 内容减 1，结果存入 OPR 中。

【例 3-27】设(CX)= 0A404H，则执行指令：

```
DEC   CX        ;类似于(CX)= 0A404H−1
```

说明：

1）OPR 可以是寄存器或存储器操作数。

2）该指令与前两个减法指令的不同在于它的操作不影响 CF 标志位。

3）该指令常用于对计数器和地址指针进行调整。

（4）NEG 求补码指令。

格式：NEG　OPR

功能：将 OPR 的内容每一位求反加 1，结果送 OPR 中。

【例 3-28】设(AL)= 0FFH，则执行指令：

```
NEG   AL        ;类似于(AL)= 01H
```

说明：该指令影响 AF、OF、PF、SF、ZF 和 CF 标志位。

（5）CMP 比较指令。

格式：CMP　OPR1,OPR2

功能：将 OPR1 内容减 OPR2 内容，结果不保存。

说明：

1）该指令执行之后，OPR1 与 OPR2 的内容均不改变。

2）CMP 指令用于比较两个操作数的大小，根据比较的结果标志位判断两个操作数的大小关系。

3）CMP 指令后面常跟条件转移指令，根据比较结果的不同产生不同的分支，具体实例见条件转移指令。

4）该指令影响 AF、OF、PF、SF、ZF 和 CF 标志位。

3. 乘法指令

（1）MUL 无符号数乘法指令。

格式：MUL　SRC

功能：如果是字节数据相乘，则将 SRC 的内容乘以 AL 寄存器的内容，得到字数据结果送 AX 寄存器；如果是字数据相乘，则将 SRC 的内容乘以 AX 寄存器的内容，得到双字数据

结果，高字送 DX 寄存器，低字送 AX 寄存器。

【例 3-29】设(AL)= 0A2H，(BL)= 11H，则执行指令：

```
MUL   BL        ;类似于(AX)= 0A2H * 11H = 0AC2H
```

说明：

1）SRC 可以是寄存器操作数或存储器操作数，而不能是立即数和段寄存器。

2）该指令只对 CF、OF 标志位有影响，而对 AF、SF、ZF 和 PF 未定义。

（2）IMUL 有符号数乘法指令。

格式：IMUL　SRC

功能：与 MUL 指令运算过程相同，只是操作对象是带符号的二进制数。

【例 3-30】设(AL)= 0B4H，(BL)= 11H，则执行指令：

```
IMUL   BL              ;类似于(AX)=(0B4H)*(11H)= FAF4H
```

分析：该指令是有符号指令，0B4H 用带符号十进制表示为−76D，11H 则为 17D，两数相乘之后，结果为−1292D，转换为十六进制为 FAF4H。

说明：

1）对于有符号数来讲，都是以补码形式存储数据的。

2）SRC 可以是寄存器操作数或存储器操作数，而不能是立即数和段寄存器。

3）该指令只对 CF、OF 标志位有影响，而对 AF、SF、ZF 和 PF 未定义。

4. 除法指令

（1）DIV 无符号数除法指令。

格式：DIV　SRC

功能：如果是字节除法就将 AX 寄存器的内容除以 SRC 的内容，商值送 AL，余数送 AH；如果是字除法就将 DX，AX 的内容除以 SRC 的内容，商值送 AX，余数送 DX。

【例 3-31】设(AX)= 0400H，(BX)= 0C8H，则执行指令：

```
DIV   BL        ;类似于(AL)= 05H，(AH)= 18H
```

分析：AX 寄存器中的内容 0400H 是无符号十进制的 1024D，BL 寄存器中的 0C8H 是无符号十进制的 200D，将 AX 内容与 BL 内容相除之后，商值 5D 送 AL，余数 24D 送 AH。

说明：

1）该指令可以进行字节、字操作，还可以进行双字操作。

2）对于 AF、CF、OF、PF、SF 和 ZF 标志位均未定义。

（2）IDIV 有符号数除法指令。

格式：IDIV　SRC

功能：与 DIV 指令相同，只不过各种数据都是带符号的，特别是余数与被除数的符号应该相同。

说明：使用本指令时应记住所用的操作数一定是有符号的，其数据都是以补码的形式存储的。

（3）CBW 字节转换为字指令。

格式：CBW

功能：将 AL 中的符号位数据扩展至 AH，若 AL 中的符号位是 0，则(AH)= 00H；若是 1，则(AH)= FFH。

【例 3-32】设(AL)= A5H，则执行指令：

```
CBW
```

分析：由于 A5H 数据存储时，第八位为 1，所以扩展成字数据之后，(AX)= FFA5H。

（4）CWD 字转换为双字指令。

格式：CWD

功能：将 AX 中的符号位数据扩展至 DX，若 AX 中的符号位是 0，则(DX)= 0000H；若是 1，则(DX)= FFFFH。其用法与 CBW 相同。

### 3.2.3 逻辑运算指令

逻辑运算指令完成对逻辑数据的运算。

1. AND 逻辑位与指令

格式：AND DST,SRC

功能：将 DST 的内容与 SRC 的内容进行按位与运算，结果送 DST。

【例 3-33】设(AL)= 11111111B，要将 AL 中的第 2 位和第 6 位清零，则可执行指令：

```
AND  AL,10111011B
```

说明：

1）该指令可以对数据的某些位进行清零。

2）逻辑与的运算规则为：1 AND 1 = 1，1 AND 0 = 0，0 AND 0 = 0，0 AND 1 = 0，即运算的两边只要一边为 0，则结果为 0。

2. OR 逻辑位或指令

格式：OR DST,SRC

功能：将 DST 的内容与 SRC 的内容进行按位或运算，结果送 DST。

说明：

1）该指令可以对数据进行置 1 操作。

2）逻辑或的运算规则为：1 OR 1 = 1，1 OR 0 = 1，0 OR 0 = 0，0 OR 1 = 1，即运算的两边只要一边为 1，结果即为 1。

3. XOR 逻辑位异或指令

格式：XOR DST,SRC

功能：将 DST 的内容与 SRC 的内容进行按位异或运算，结果送 DST。

说明：按位异或运算的规则为：1 XOR 1 = 0，1 XOR 0 = 1，0 XOR 0 = 0，0 XOR 1 = 1，即运算的两边只要相同就为 0，不同才为 1。

4. NOT 逻辑位非指令

格式：NOT DST

功能：将 DST 的内容逐位取反，结果送 DST。

说明：位非运算的规则为：非 1 即 0，非 0 即 1。

5. TEST 测试指令

格式：TEST DST,SRC

功能：将 DST 的内容与 SRC 的内容进行逐位与运算，结果不保存，只根据结果设置状态标志。

说明：

1）该指令的作用可以检测数据某个位置上的状态。

2）该指令后面一般接跳转指令。

### 3.2.4　移位指令

移位指令完成对数据的移位操作。

1．SHL 逻辑左移指令

格式：SHL　OPR,CNT

功能：将 OPR 的内容左移 CNT 指定的次数，低位补入相应个数的 0，CF 的内容为最后移入位的值。每移一位，若新 OPR 的最高位与 CF 的内容不同，表示 OPR 的符号位改变，以 OF=1 作记号；否则，记 OF= 0。移位后的 PF、SF、ZF 的变化表示移位后的结果。这里 AF 的变化无意义，不予定义。

2．SAL 算术左移指令

格式：SAL　OPR,CNT

功能：与 SHL 相同。

SAL 算术左移指令示意如图 3-12 所示。其效果完全同 SHL。


图 3-12　SAL 算术左移指令示意图

3．SAR 算术右移指令

格式：SAR　OPR,CNT

功能：将 OPR 的内容右移 CNT 指定的次数，左边空出的位上补最高位内容（符号不变），CF 的内容为最后移入位的值。移位后会影响 CF、OF、PF、SF 和 ZF 的值。这里 AF 的变化无意义，不予定义。

SAR 算术右移指令如图 3-13 所示。


图 3-13　SAR 算术右移指令示意图

4．SHR 逻辑右移指令

格式：SHR　OPR,CNT

功能：与 SAR 基本相同，只是左边空出的位上补入相应个数的 0，最低位送 CF，CF 的内容为最后移入位的值。每移一位，若新 OPR 的最高位与次高位的内容不同，表示 OPR 的符号位改变，以 OF=1 作记号；否则，记 OF= 0。移位后的 PF、SF、ZF 的变化表示移位后的结果。这里 AF 的变化无意义，不予定义。

5．ROL 循环左移指令

格式：ROL　OPR,CNT

功能：将 OPR 内容的最高位与最低位连成一个环，移位时就在这个环中进行，左移次数由 CNT 决定，CF 的内容为最后移入位的值。每移一位，若新 OPR 的最高位与 CF 的内容不

同，表示 OPR 的符号位改变，以 OF=1 作记号；否则，记 OF= 0。移位后的 PF、SF、ZF 的变化表示移位后的结果。这里 AF 的变化无意义，不予定义。

ROL 循环左移指令如图 3-14 所示。


图 3-14　ROL 循环左移指令示意图

6. ROR 循环右移指令

格式：ROR　OPR,CNT

功能：与 ROL 基本相同，只是移位方向是向右。每移一位，若新 OPR 的最高位与次高位的内容不同，表示 OPR 的符号位改变，以 OF=1 作记号；否则，记 OF= 0。移位后的 PF、SF、ZF 的变化表示移位后的结果。这里 AF 的变化无意义，不予定义。

7. RCL 带进位循环左移指令

格式：RCL　OPR,CNT

功能：将 OPR 的内容连同 CF 标志内容一起向左循环移位 CNT 次。

RCL 带进位循环左移指令如图 3-15 所示。每移一位，若新 OPR 的最高位与 CF 的内容不同，表示 OPR 的符号位改变，以 OF=1 作记号；否则，记 OF= 0。移位后的 PF、SF、ZF 的变化表示移位后的结果。这里 AF 的变化无意义，不予定义。


图 3-15　RCL 带进位循环左移指令示意图

8. RCR 带进位循环右移指令

格式：RCR　OPR,CNT

功能：与 RCL 基本相同，只是移位方向是向右。每移一位，若新 OPR 的最高位与次高位的内容不同，表示 OPR 的符号位改变，以 OF=1 作记号；否则，记 OF= 0。移位后的 PF、SF、ZF 的变化表示移位后的结果。这里 AF 的变化无意义，不予定义。

### 3.2.5　串操作指令

串操作指令完成对字符串的各种操作，其寻址方式只用隐含寻址，源串固定使用 SI，目的串固定使用 DI。

1. MOVS 串传送指令

格式：可有 3 种，即

```
MOVS   DST,SRC        ;DST 和 SRC 分别是先已定义了的两个符号地址
                      ;DST 在附加段，SRC 在数据段
MOVSB（字节）
MOVSW（字）
```

功能：该指令可以把由 SI 指向的数据段中的一个字（或字节）送到由 DI 指向的附加段中的一个字（或字节）中去，同时根据方向标志及数据格式（字或字节）对 SI 和 DI 进行修改。

说明：

1）如果是字节操作，则 SI 与 DI 变化时是±1；如果是字操作，则 SI 与 DI 变化时是±2。

2）当方向标志 DF=1 时用−，当 DF= 0 时用+。

2．STOS 存入串指令

格式：

```
STOS   DST      ;DST 是先已定义了的符号地址，在附加段
STOSB（字节）
STOSW（字）
```

功能：该指令把 AL 或 AX 的内容存入由 DI 指定的附加段的某单元中，并根据 DF 的值及数据类型修改 DI 的内容。

说明：

1）如果是字节操作则先将 AL 的内容存入 DI 指定的附加段的某单元中，然后 DI 再自动加/减 1；如果是字操作则将 AX 的内容存入[DI]，然后 DI 再自动加/减 2。

2）与 MOVS 指令第 2）点说明相同。

3．LODS 取串指令

格式：可有 3 种，即

```
LODS   SRC      ;SRC 是先已定义了的符号地址，在数据段
LODSB
LODSW
```

功能：该指令把由 SI 指定的数据段中某单元的内容送到 AL 或 AX 中，并根据方向标志及数据类型修改 SI 的内容。

说明：

1）如果是字节操作则先将由 SI 指定的单元内容送入 AL 中，然后 SI 再自动加/减 1；如果是字操作则将[SI]送入 AX，然后 SI 再自动加/减 2。

2）与 MOVS 指令第 2 点说明相同。

4．CMPS 串比较指令

格式：可有 3 种，即

```
CMP   SRC,DST
   ;DST 和 SRC 分别是已定义了的两个符号地址，DST 在附加段，SRC 在数据段
CMPSB
CMPSW
```

功能：指令把由 SI 指向的数据段中的一个字（或字节）与由 DI 指向的附加段中的一个字（或字节）相减，但不保存结果，只根据结果置条件码。

5．SCAS 串搜索指令

格式：可有 3 种，即

```
SCAS   DST      ;DST 是先已定义了的符号地址，在附加段
SCASB
SCASW
```

功能：该指令把 AL（或 AX）的内容与由 DI 指定的在附加段中的一个字节（或字）进行比较，但不保存结果，只根据结果置条件码。

6. REP 重复前缀指令

格式：REP　strpri（strpri 可为 MOVS、LODS 或 STOS）

功能：当 CX = 0 时，不执行 strpri 给定的指令；否则继续执行。

【例 3-34】程序段

```
MOV   CX,6          ;将字符串长度 6 送入 CX
CLD                 ;清方向标志，字符串地址增量
REP   MOVSB         ;重复送串中的各个字节，直到 CX= 0
```

7. REPE/REPZ 重复前缀指令

格式：REPE（或 REPZ）　strpri（strpri 可为 CMPS 或 SCAS）

功能：当 CX = 0 或 ZF = 0（即某次比较的结果两个操作数不等）时不执行 strpri 给定的指令；否则继续执行。

8. REPNE/REPNZ 重复前缀指令

格式：REPNE（或 REPNZ）　strpri（strpri 可为 CMPS 或 SCAS）

功能：与 REPE/REPZ 相同，只是退出重复执行的条件为 CX = 0 或 ZF = 1。

### 3.2.6 控制转移指令

控制转移指令的作用为直接或根据条件是否满足，改变任务的执行顺序，跳转到指定的地址执行指令。

1. JMP 无条件跳转指令

已在 3.1.2 小节中讨论过了，此处不再赘述。

2. 条件转移指令

条件转移只能有 8 位的位移量。条件转移是程序转移与否，要看条件满足不满足。其指令代码的特点是在字母 J 后面跟一个条件字母。例如，JZ 表示运算结果为零，就转移；JNZ 表示运算结果不为零，则转移。请注意测试条件。详述如下：

（1）根据单个条件标志的设置情况转移。

1）JZ（或 JE）结果为零（或相等）则转移。

格式：JE（或 JZ）　标号

测试条件：ZF = 1。

2）JNZ（或 JNE）结果不为零（或不相等）则转移。

格式：JNZ（或 JNE）　标号

测试条件：ZF = 0。

3）JS 结果为负则转移。

格式：JS　标号

测试条件：SF = 1。

4）JNS 结果为正则转移。

格式：JNS　标号

测试条件：SF = 0。

5）JO 溢出则转移。
格式：JO　标号。
测试条件：OF = 1。
6）JNO 不溢出则转移。
格式：JNO　标号
测试条件：OF = 0。
7）JP（或 JPE）奇偶位为 1 则转移。
格式：JP　　标号
测试条件：PF = 1。
8）JNP（或 JPO）奇偶位为 0 则转移。
格式：JNP（或 JPO）　标号
测试条件：PF = 0。
9）JB（或 JNAE，JC）低于或不高于或等于或进位位为 1 则转移。
格式：JB（或 JNAE，JC）　标号
测试条件：CF = 1。
10）JNB（或 JAE，JNC）不低于或高于或等于或进位位为 0 则转移。
格式：JNB（或 JAE，JNC）　标号
测试条件：CF = 0。
（2）比较两个无符号数，并根据比较的结果转移。
1）JB（或 JNAE，JC）。已讨论过。
2）JNB（或 JAE，JNC）。已讨论过。
3）JBE（或 JNA）低于或等于或不高于则转移。
格式：JBE（或 JNA）　标号
测试条件：CF 或 ZF = 1。
4）JNBE（或 JA）不低于或等于或高于则转移。
格式：JNBE（或 JA）　标号
测试条件：CF 或 ZF = 0。
（3）比较两个带符号数，并根据比较的结果转移。
1）JL（或 LNGE）小于或不大于或等于则转移。
格式：JL（或 JNGE）　标号
测试条件：SF 异或 OF = 1。
2）JNL（或 JGE）不小于或大于或等于则转移。
格式：JNL（或 JGE）　标号
测试条件：SF 异或 OF = 0。
3）JLE（或 JNG）小于或等于或不大于则转移。
格式：JLE（或 JNG）　标号
测试条件：（SF 异或 OF）或 ZF = 1。
4）JNLE（或 JG）不小于或等于或大于则转移。
格式：JNLE（或 JG）　标号
测试条件：（SF 异或 OF）或 ZF = 0。

（4）测试 CX 的值为 0 则转移指令。

JCXZ CX 寄存器的内容为 0 则转移。

格式：JCXZ　标号

测试条件：(CX)= 0。

### 3.2.7　循环指令

循环指令的作用为：根据条件是否满足完成一串重复的操作。

1．LOOP 循环指令

格式：LOOP　标号

测试条件：(CX)不等于 0。

2．LOOPZ/LOOPE）循环指令

格式：LOOPZ（或 LOOPE）　标号

测试条件：(CX)不等于 0 且 ZF = 1。

3．LOOPNZ/LOOPNE 循环指令

格式：LOOPNZ（或 LOOPNE）　标号

测试条件：(CX)不等于 0 且 ZF = 0。

这 3 条指令执行的步骤是：

1）先将 CX 的内容减 1，即(CX)←(CX)–1。

2）再检查是否满足测试条件，如果满足则(IP)←(IP)+D8 的符号扩充。

### 3.2.8　过程调用和返回指令

在第 2 章中已简单讲述了过程调用的概念。现在做深入讨论。

令 PROC 和 ENDP 来把它定义为一个子程序。

在汇编语言中，要使某一程序段成为一个子程序，必须首先通过定义：

```
<过程名>    PROC    [NEAR]/FAR  ;NEAR 表示过程在本代码段内，FAR 表示
                                ;过程在其他代码段内
            …                   ;
            RET                 ;返回指令
<过程名>    ENDP
```

子程序以 PROC 开头，用 ENDP 结束。

子程序可以采用 CALL 指令来调用。调用一个过程的语句格式有以下 4 种：

（1）段内直接调用：CALL　过程名。

（2）段内间接调用：CALL　寄存器名或内存地址。子程序地址用数据的寻址方式。

（3）段间直接调用：CALL　FAR　PTR　过程名。子程序地址用 JMP 的寻址方式。

（4）段间间接调用：CALL　DWORD　PTR 内存地址。子程序地址同数据的寻址方式，此语句可以插入到上述定义的中间。

子程序也可以不写成“过程”形式，一般的子程序可以认为从子程序的入口地址开始到 RET 指令结束的一段程序。调用时 CALL 指令中的“过程名”用“子程序中第一条可执行语句的名字”来代替。

### 3.2.9　中断指令

在第 2 章中已简单讲述了中断的概念。因为第 7 章还要深入讨论，现在只简单介绍中断指令。

中断指令的作用为：调用中断程序及中断程序执行完之后返回。

1．INT 中断调用指令

格式：INT　TYPE

或　　INT

功能：执行断点保护操作。

2．INTO 结果溢出中断指令

执行的操作：若 OF = 1，则：

(SP)←(SP)–2

((SP)+1,(SP))←(PSW)

(SP)←(SP)–2

((SP)+1,(SP))←(CS)

(SP)←(SP)–2

((SP)+1,(SP))←(IP)

(IP)←(10H)

(CS)←(12H)

3．IRET 中断返回指令

格式：IRET

功能：执行断点返回操作。

注意：过程调用和中断两者都需要进行保护断点，即下一条指令地址→跳至子程序或中断服务程序→保护现场→子程序或中断处理→恢复现场→恢复断点（即返回主程序）等操作。

### 3.2.10　处理机控制指令

1．标志处理指令

（1）CLC 进位位置 0 指令，即 CF←0。

（2）CMC 进位位求反指令，即 CF←（CF 取反）。

（3）STC 进位位置 1 指令，即 CF←1。

（4）CLD 方向标志置 0 指令，即 DF←0。

（5）STD 方向标志置 1 指令，即 DF←1。

（6）CLI 中断标志置 0 指令，即 IF←0。

（7）STI 中断标志置 1 指令，即 IF←1。

2．其他处理机控制指令

这些指令可以控制处理机状态，它们都不影响条件码。

（1）NOP 空操作或无操作指令。

功能：该指令不执行任何操作，其机器码占用一个字节，在调试程序时往往用这条指令占有一定的存储单元，以便在正式运行时用其他指令取代。

（2）HLT 停机指令。

功能：该指令可使机器暂停工作，使处理机处于停机状态，以便等待一次外部中断到来，中断结束后可继续执行下面的程序。

（3）WAIT 等待指令

功能：该指令使处理机处于空转状态，它也可以用来等待外部中断的发生，但中断结束后仍返回 WAIT 指令继续运行。

（4）ESC 换码指令。

格式：ESC　mem　（mem 指出一个存储单元）

功能：ESC 指令把该存储单元的内容送到数据总线上，当然，ESC 指令不允许使用立即数和寄存器寻址方式。这条指令在使用协处理机（Coprocessor）执行某些操作时，可从存储器取得指令或操作数。协处理机（如 8087）则是为了提高速度而可以选配的硬件。

（5）LOCK 封锁指令。

功能：该指令是一种前缀，它可与其他指令联合，用来维持总线的锁存信号，直到与其联合的指令执行完毕为止。当 CPU 与其他处理机协同工作时，该指令可避免破坏有用信息。

## 3.3　32 位新增指令简介

1985 年 Intel 公司正式公布了 32 位微处理器 80386，其内、外部数据线都是 32 位（80386 SX CPU 外部数据线为 16 位），地址线 32 根，CPU 内部寄存器扩展到 16 个，指令系统得到扩展，随后，Intel 80386、Intel 80486 及 Pentium 系列微处理器都继承了 80386 的 32 位指令系统，并在此基础上又新增了若干专用指令，有效地增强了 32 位微处理器的功能。

32 位微处理器中的 8 个 32 位通用寄存器分别是 EAX、EBX、ECX、EDX、ESI、EDI、ESP 和 EBP，是在 16 位的基础上扩展而成的。段寄存器在原有的 4 个基础上增加了两个附加数据段寄存器，即 FS 和 GS，其长度还是 16 位。指令指针 IP 扩展为 32 位 EIP。标志寄存器也扩展为 32 位，增加的标志主要用于 CPU 的控制，很少在应用程序中使用。

32 位新增指令有以下几种：

- 双精度左移指令 SHLD。
- 双精度右移指令 SHRD。
- 前向扫描 16/32 位操作指令 BSF。
- 后向扫描 16 位操作指令 BSR。
- 位操作指令 BT、BTC、BTR、BTS。
- 条件设置指令 SETX（X 为条件）。
- 字节交换指令 BSWAP。
- 交换加指令 XADD。
- 比较交换指令 CMPXCHG。
- 高速缓存无效指令 INVD。
- 回写及高速缓存无效指令 WBINVD。
- TLB 无效指令 INVLPG。
- 8 字节交换指令 CMPXCHG8B。
- 处理器特征识别指令 CPUID。

- 读时间标记计数器指令 RDTSC。
- 读模型专用寄存器指令 RDMSR。
- 写模型专用寄存器指令 WRMSR。
- 系统管理方式返回指令 RSM。

## 本章小结

微型计算机指令系统就是指微型计算机中所有的机器指令的集合。指令系统是表征一台计算机性能的重要因素，它的格式与功能不仅直接影响到机器的硬件结构，而且也影响到系统软件。

计算机中的一条完整指令包括两个部分：①操作码字段部分，它指出了计算机所要执行的操作；②操作数字段部分，它指出了执行操作指令过程中所需要的操作数，而根据操作码字段的个数不同，可以分为一地址、二地址或三地址指令。指令字的长度一般有字节、字、双字 3 种形式，但在高档微型计算机中多采用 32 位长度的单字长形式。

寻址方式有指令寻址方式和数据寻址方式两种，立即寻址、直接寻址、寄存器寻址、寄存器间接寻址、变址寻址、基址加变址寻址是常用的数据寻址方式，而指令寻址方式有顺序寻址和跳跃寻址两种，由程序计数器来跟踪。

不同的计算机具有不同的指令系统。一般情况下都包含数据传送类指令、算术运算类指令、逻辑运算类指令、程序控制类指令、输入/输出（I/O）类指令、字符串类指令和系统控制类指令。

## 习题三

### 一、填空题

1．直接变址寻址操作数的偏移地址是由_____和_____之和求得。
2．逻辑运算指令包括_____、_____、_____和_____等操作。
3．段内转移指令将改变_____的值。
4．段间转移指令将改变_____及_____的值。
5．指令 CLC 的作用是_____。
6．设(AH)=13H，执行指令 SHL AH,1 后，(AH)=_____。
7．指令 MOV　AX,[Bx+SI+6]，其源操作数的寻址方式为_____。
8．若(AX)=2000H，则执行指令 CMP AX,2000H 后，(AX)=_____，ZF=_____。

### 二、选择题

1．对某个寄存器中操作数的寻址方式称为（　　）寻址方式。
A．直接　　B．间接
C．寄存器　　D．寄存器间接
2．设(AX)=1234H，(BX)=5678H，执行下列指令后，AL 的值应是（　　）。
PUSH　AX

```
PUSH  BX
POP   AX
POP   BX
```

A．12H　　B．34H

C．56H　　D．78H

3．已知 SP=2001H，[2001H]=34H，[2002H]=12H，经操作 POP BX 后，将 2002H、2001H 单元的内容弹到 BX，(SP)=（　　）。

A．2001H　　B．2002H

C．2003H　　D．2004H

4．下列指令错误的是（　　）。

A．RCR　DX,CL　　B．RET

C．IN　AX,0268H　　D．OUT　80H,AL

5．若将 AL 中的值高 4 位取反，低 4 位保持不变，使用下列指令（　　）。

A．AND　AL,F0H　　B．OR　AL,F0H

C．NOT　AL,F0H　　D．XOR　AL,F0H

6．BL 寄存器高 4 位保持不变，低 4 位置“1”的操作正确的是（　　）。

A．AND　BL,0FH　　B．OR　BL,0FH

C．NOT　BL,0FH　　D．XOR　BL,0FH

7．指令 ADD AX，[SI]中源操作数的寻址方式是（　　）。

A．基址寻址　　B．基址和变址寻址

C．寄存器间接寻址　　D．寄存器间接寻址

8．逻辑移位指令 SHL 用于（　　）。

A．带符号数乘 2　　B．带符号数除 2

C．无符号数乘 2　　D．无符号数除 2

## 三、判断改错题

（　　）1．在寄存器寻址方式中，指定的寄存器中存放的是操作数地址。

改错：

（　　）2．用某个寄存器中操作数的寻址方式称为寄存器间接寻址。

改错：

（　　）3．转移类指令能改变指令执行顺序，因此，执行这类指令时，IP 和 SP 的值都将发生变化。

改错：

（　　）4．指令的寻址方式有顺序和跳跃两种方式，采用跳跃寻址方式，可以实现堆栈寻址。

改错：

## 四、名词解释

指令、程序、指令系统、寻址方式。

## 五、综合题

1．段寄存器 CS=1200H，指令指针寄存器 IP=FF00H，此时指令的物理地址为多少？指向这一物理地址的 CS 值和 IP 值是唯一的吗？

2．指出下列各条指令中源操作数的寻址方式，并指出下列各条指令执行之后 AX 寄存器的内容。设有关寄存器和存储单元的内容为：(DS)=2000H，(BX)=0100H，(SI)=0002H，(20100H)=12H，(20101H)=34H，(20102H)=56H，(20103H)=78H，(21200H)=2AH，(21201H)=4CH，(21202H)=0B7H，(21203H)=65H：

（1）MOV　AX,1200H

（2）MOV　AX,BX

（3）MOV　AX,[1200H]

（4）MOV　AX,[BX]

（5）MOV　AX,1100H[BX]

（6）MOV　AX,[BX],[SI]

（7）MOV　AX,1100H[BX],[SI]

3．指出下列指令的错误：

（1）MOV　DS,0200H

（2）MOV　AH,BX

（3）MOV　BP,AL

（4）MOV　AX,[SI][DI]

（5）OUT　310H,AL

（6）MOV　[BX],[SI]

（7）MOV　CS,AX

（8）PUSH　CL

4．试用以下 3 种方式写出交换寄存器 SI 和 DI 的内容：

（1）用数据交换指令实现。

（2）不用数据交换指令，仅使用数据传送指令实现。

（3）用栈操作指令实现。

5．试分析下面程序段完成什么功能。

```
MOV　CL,4
SHR　AX,CL
MOV　BL,DL
SHR　DX,CL
SHL　BL,CL
OR　 AH,BL
```