第3章 微处理器的指令系统

【学习目标】

本章重点讨论 8086/8088 CPU 的指令系统；对扩展指令集的实用知识，应有所了解。

通过本章对 8086/8088 CPU 寻址方式和指令系统的学习，应能掌握汇编语言程序设计所需要的汇编语言和编写程序段的基础知识。

【学习要求】

- 在理解与掌握各种寻址方式的基础上，着重掌握存储器寻址的各种寻址方式。
- 应熟练掌握 4 类数据传送指令。难点是 XLAT、IN、OUT 指令。
- 学习算术运算类指令中的难点是带符号乘、除指令与十进制指令。
- 学习逻辑运算和移位循环类指令时，要着重理解 CL 的设置以及进位位的处理。
- 学习串操作类指令时，着重理解重复前缀(REP)的使用。
- 学习程序控制类指令时，着重理解条件转移的条件及测试条件。
- 理解指令集的发展趋势，了解几种扩展的指令集。

3.1 8086/8088 的寻址方式

要熟悉指令的操作先要了解指令的寻址方式。8086/8088 的寻址方式分为两种不同的类型：数据寻址方式和程序存储器寻址方式。前者是寻址操作数地址；后者是寻址程序地址(在代码段中)。

3.1.1 数据寻址方式

数据寻址方式有多种，都是寻找操作数。在指令格式中，所有操作数的流向都是由源到目标，即它们在指令汇编语言格式的操作数区域中都是规定由右到左。源和目标可以是寄存器或存储器，但不能同时为存储器(除个别串操作指令 MOVS 外)。

1. 立即寻址

立即寻址的操作数就在指令中，当执行指令时，CPU 直接从紧跟着指令代码的后续地址单元中取得该立即数。立即数可以是 8 位，也可以是 16 位；并规定只能是整数类型的源操作数。这种寻址主要用来给寄存器赋初值，指令执行速度快。

下面列出了各种立即数寻址的 MOV 指令示例。

```
MOV AH,4CH                      ;把 4CH 传送到 AH 中
MOV AX,1234H                    ;把 1234H 传送到 AX 中
MOV CL,100                      ;把 100(64H)传送到 CL
MOV AX,'AB'                     ;把 ASCII 码 BA*(4241H)传送到 AX 中
MOV CL,10101101B                ;把二进制数 10101101 传送到 CL 中
MOV WORD PTR[SI],6180H   ;把立即数 6180H 传送到数据段由 SI 和 SI+1 所指的两存储单元中
```

2. 寄存器寻址

寄存器寻址的操作数就放在寄存器中，而寄存器名在指令中指出。在一条指令中，源操作数和目的操作数都可以采用寄存器寻址方式。这种寻址的指令长度短，操作数就在 CPU 内部，执行速度快。注意，使用时源与目标操作数应有相同的数据类型长度。

下面列出了各种寄存器寻址的 MOV 指令示例。注意，代码段寄存器不能用 MOV 指令来改变，因为若只改变 CS 而 IP 为未知数，则下一条指令的地址将是不确定的。

```
MOV BH,BL                       ;把 BL 复制到 BH 中
MOV CX,AX                       ;把 AX 复制到 CX 中
MOV DI,SI                       ;把 SI 复制到 DI 中
MOV AX,ES                       ;把 ES 复制到 AX 中
```

下面将讨论属于存储器寻址的各种数据寻址方式。存储器寻址比较复杂，当 CPU 寻找存储器操作数时，应根据指令给出的寻址方式，由 EU 先计算出操作数地址的偏移量（即有效地址 EA）。EA 的值由汇编程序根据指令所采用的寻址方式自动计算得出。计算 EA 的通式为：

EA＝基址值(BX 或 BP)＋变址值(SI 或 DI)＋位移量 DISP

3. 直接数据寻址

直接数据寻址有两种基本形式：直接寻址和位移寻址。

(1) 直接寻址

直接寻址是指令中以位移量方式直接给出存储器操作数的偏移地址，即有效地址 EA＝DISP。这种寻址方式的指令执行速度快，用于存储单元与 AL、AX 之间的 MOV 指令。

下面列出了使用 AX、AL 的直接寻址指令示例。

```
MOV AX,[1680H]①          ;把数据段存储器地址 1680H 和 1681H 两单元的字内容复制到 AX 中
MOV AX,NUMBER            ;把数据段存储器地址 NUMBER 中的字内容复制到 AX 中
```

① 注意：汇编语言中很少采用绝对偏移地址(如 1680H)，通常采用符号地址。

```
MOV TWO,AL                ;把 AL 的字节内容复制到数据段存储单元 TWO 中
MOV ES: [3000H],AX        ;把 AX 的字内容复制到附加数据段存储单元 3000H 中
MOV AX,DATA               ;把数据段存储单元 DATA 的字内容复制到 AX 中
```

(2) 位移寻址

位移寻址也以位移量方式直接给出存储器操作数的偏移地址,但适合于几乎所有将数据从存储单元传送到寄存器的指令。

下面列出了使用位移量的直接数据寻址的示例。

```
MOV CL,COW                ;把数据段存储单元 COW 的字节内容复制到 CL 中
MOV ES,NUMBER             ;把数据段存储器地址 NUMBER 中的字内容复制到 ES 中
MOV CX,DATA2              ;把数据段存储单元 DATA2 中的字内容复制到 CX 中
MOV DATA3,BP              ;把基址指针寄存器 BP 的内容复制到数据段存储单元 DATA3 中
MOV DI,SUM                ;把数据段存储单元 SUM 的字内容复制到 DI 中
MOV NUMBER,SP             ;把 SP 的内容复制到数据段存储单元 NUMBER 中
```

位移寻址与直接寻址的操作相同,只是它的指令为 4 字节而不是 3 字节。

4. 寄存器间接寻址

寄存器间接寻址的操作数一定是在存储器中,而存储单元的有效地址 EA 则由寄存器保存,这些寄存器是基址寄存器 BX、基址指针寄存器 BP、变址寄存器 SI 和 DI 之一或它们的某种组合。书写指令时,这些寄存器带有方括号[]。

下面给出了寄存器间接寻址的指令示例。

```
MOV AL,[BX]               ;把数据段中用 BX 寻址的存储单元的字节内容复制到 AL 中
MOV [SI],BL               ;把寄存器 BL 的内容复制到数据段由 SI 寻址的存储单元
MOV CX,[DX]               ;把数据段由 DX 寻址的存储单元的字内容复制到 CX 中
MOV [BP],CL①              ;把寄存器 CL 的内容复制到堆栈段由 BP 寻址的存储单元中
MOV [SI],[BX]             ;除数据串操作指令外,不允许由存储器到存储器的传送
```

5. 基址加变址寻址

基址加变址寻址类似于间接寻址,其操作数的有效地址 EA 由基址寄存器(BX 或 BP)的内容与变址寄存器(SI 或 DI)的内容之和来确定。

在使用基址加变址寻址时,通常,用基址寄存器保持存储器数组的起始地址,而变址寄存器保持数组元素的相对位置。如果是用 BP 寄存器寻址存储器数组,则由 BP 寄存器和变址寄存器两者生成有效地址。

下面给出了基址加变址寻址的指令示例。

```
MOV CL,[BX+SI]            ;把由 BX+SI 寻址的数据段存储单元的字节内容复制到 CL
MOV CX,[BP+DI]            ;把由 BP+DI 寻址的堆栈段存储单元内的数据字内容复制到 CX
MOV [BX+DI],SP            ;把 SP 的字内容存入由 BX+DI 寻址的数据段存储单元
MOV [BP+SI],CH            ;把寄存器 CH 的字节内容存入到由 BP+SI 寻址的堆栈段存储单元
```

① 注意:系统把由 BP 寻址的数据默认为在堆栈段中,其他间接寻址方式均默认为数据段。

6. 寄存器相对寻址

寄存器相对寻址是带有位移量 DISP 的基址或变址寄存器(BX、BP 或 DI、SI)寻址。下面给出了寄存器相对寻址的指令示例。

```
MOV CL,[SI+200H]          ;把由 SI+200H 寻址的数据段存储单元的字节内容装入 CL
MOV ARRAY[DI],BL          ;把 BL 中的字节内容存入由 ARRAY+DI 寻址的数据段存储单元
MOV LIST[DI+3],AX         ;把 AX 的字内容存入由 LIST+DI+3 之和寻址的数据段存储单元
MOV AX,ARRAY[BX]          ;把数据段中由 ARRAY+BXD 寻址的字内容装入 AX
MOV SI,[AL+12H]           ;把由 AL+12H 寻址的数据段存储单元的字内容装入 SI
```

注意：位移量可以是在方括号[]内加到寄存器上的一个带符号的数。例如，可以是指令 MOV AL,[SI+4]，也可以是指令 MOV AL,[SI−2]。位移量还可以是加在[]前面的符号偏移地址，例如 MOV AL,DATA[SI]。也可以同时出现两种形式的位移量，例如 MOV AL,DATA[SI+3]。在 8086/8088 中，位移量的取值范围是 −32 768(8000H)～+32 767(7FFFH)。

7. 相对基址加变址寻址

相对基址加变址寻址是用基址、变址与位移量 3 个分量之和形成有效地址的寻址方式。

下面给出了相对基址加变址寻址的指令示例。

```
MOV BL,[BX+SI+100H]       ;把由 BX+SI+100H 寻址的数据段存储单元的字节内容装入 BL
MOV AX,ARRAY[BX+DI]       ;把由 ARRAY+BX+DI 之和寻址的数据段存储单元的字内容装入 AX
MOV LIST[BP+DI],BX        ;把 BX 的字内容存入由 LIST+BP+DI 之和寻址的数据段存储单元
MOV FILE[BP+DI+2],DL      ;把 DL 存入由 BP+DI+2 之和寻址的堆栈段存储单元
```

相对基址加变址寻址方式一般很少使用，通常用来寻址存储器的二维数组数据。

3.1.2 程序存储器寻址方式

程序存储器寻址方式即转移类指令(转移指令 JMP 和调用指令 CALL)的寻址方式。这种寻址方式最终是要确定一条指令的地址。

在 8086/8088 系统中，由于存储器采用分段结构，所以转移类指令有段内转移和段间转移之分。所有的条件转移指令只允许实现段内转移，而且是段内短转移，即只允许转移的地址范围在 −128～+127 字节内，由指令中直接给出 8 位地址位移量。对于无条件转移和调用指令又可分为段内短转移、段内直接转移、段内间接转移、段间直接转移和段间间接转移等 5 种不同的寻址方式。

3.1.3 堆栈存储器寻址方式

堆栈是存储器中一个很重要的特定存储区，它用来暂时存放数据，并为程序保存返回地址。堆栈存储器是按“后进先出”(LIFO)原理操作的。用 PUSH 指令将数据压入堆

栈，用 POP 指令将其弹出堆栈。CALL 指令用堆栈保存程序返回地址，而 RET（返回）指令从堆栈取出返回地址。

堆栈存储器由堆栈段寄存器 SS 和堆栈指针 SP 寻址。SS 中记录的是其 16 位的段地址，它将确定堆栈段的段基址，而 SP 的 16 位偏移地址将指定当前堆栈的栈顶，即指出从堆栈段的段基址到栈顶的偏移量；栈顶是堆栈操作的唯一出口，它是堆栈地址较小的一端。

下面列出了可使用的一些 PUSH 和 POP 指令的示例。

```
PUSHF                ;把标志寄存器的内容复制到堆栈中
POPF                 ;把从堆栈弹出的一个字装入标志寄存器 FLAGS
PUSH DS              ;把 DS 的内容复制到堆栈中
POP CS               ;非法操作
PUSHA                ;把通用寄存器 AX、CX、DX、BX、SP、BP、DI、SI 的内容复制到堆栈中
POPA                 ;从堆栈中弹出数据并顺序装入 SI、DI、BP、SP、BX、DX、CX、AX 中
```

3.1.4 其他寻址方式

1. 串操作指令寻址方式

数据串（或称字符串）指令不能使用正常的存储器寻址方式来存取数据串指令中使用的操作数。执行数据串指令时，源串操作数第 1 个字节或字的有效地址应存放在源变址寄存器 SI 中（不允许修改），目标串操作数第 1 个字节或字的有效地址应存放在目标变址寄存器 DI 中（不允许修改）。在重复串操作时，8086/8088 能自动修改 SI 和 DI 的内容，以使它们能指向后面的字节或字。

2. I/O 端口寻址方式

（1）直接端口寻址

端口地址以 8 位立即数方式在指令中直接给出。例如，IN AL，n 指令是将端口号为 8 位立即数 n 的端口地址中的字节操作数输入到 AL，它所寻址的端口号只能在 0～255 范围内。

（2）间接端口寻址

这类似于寄存器间接寻址，16 位的 I/O 端口地址在 DX 寄存器中，即通过 DX 间接寻址，故可寻址的端口号为 0～65 535。例如，OUT DX，AL 指令是将 AL 的字节内容输出到由 DX 指出的端口中去。

3.2 数据传送类指令

数据传送类指令包括 4 类：通用数据传送指令、目标地址传送指令、标志位传送指令、I/O 数据传送指令。

3.2.1 通用数据传送指令

通用数据传送指令包括基本的传送指令 MOV,堆栈操作指令 PUSH 和 POP,数据交换指令 XCHG 与字节翻译指令 XLAT。

1. 基本的传送指令

```
MOV d,s;d←s
```

指令功能：将由源 s 指定的源操作数送到目标 d。

MOV 指令可实现的数据传送类型可归纳为以下 7 种：

(1) MOV mem/reg1,mem/reg2

由 mem/reg2 所指定的存储单元或寄存器中的 8 位数据或 16 位数据传送到由 mem/reg1 所指定的存储单元或寄存器中,但不允许从存储器传送到存储器。

(2) MOV mem/reg,data

将 8 位或 16 位立即数 data 传送到由 mem/reg 所指定的存储单元或寄存器中。

(3) MOV reg,data

将 8 位或 16 位立即数 data 传送到由 reg 所指定的寄存器中。

(4) MOV ac,mem

将存储单元中的 8 位或 16 位数据传送到累加器 ac 中。

(5) MOV mem,ac

将累加器 AL(8 位)或 AX(16 位)中的数据传送到由 mem 所指定的存储单元中。

(6) MOV mem/reg,segreg

将由 segreg 所指定的段寄存器(CS、DS、SS、ES 之一)的内容传送到由 mem/reg 所指定的存储单元或寄存器中。

(7) MOV segreg,mem/reg

允许将由 mem/reg 指定的存储单元或寄存器中的 16 位数据传送到由 segreg 所指定的段寄存器(但代码段寄存器 CS 除外)中。

注意：MOV 指令不能直接实现从存储器到存储器之间的数据传送,但可以通过寄存器作为中转站完成这种传送。

2. 堆栈操作指令

(1) PUSH s

字压入堆栈指令,允许将源操作数 s(16 位)压入堆栈。

(2) POP d

字弹出堆栈指令,允许将堆栈中当前栈顶两相邻单元的数据字弹出到 d。

这是两条成对使用的进栈与出栈指令。其中,s 和 d 可以是 16 位寄存器或存储器两相邻单元,以保证堆栈按字操作。

PUSH 和 POP 是两条很重要的指令，用来保存并恢复来自堆栈存储器的数据。例如，在子程序调用或中断处理过程时，分别要保存返回地址或断点地址，在进入子程序或中断处理后，还需要保留通用寄存器的值；而在由子程序返回或由中断处理返回时，则要恢复通用寄存器的值，并分别将返回地址或中断地址恢复到指令指针寄存器中。

3. 数据交换指令

```
XCHG d,s
```

本指令的功能是将源操作数与目标操作数（字节或字）相互对应交换位置。交换可以在通用寄存器与累加器之间、通用寄存器之间、通用寄存器与存储器之间进行。但不能在两个存储单元之间交换，段寄存器与 IP 也不能作为源或目标操作数。例如：

```
XCHG AX,[SI+0400H]
```

设当前 CS＝1000H，IP＝0064H，DS＝2000H，SI＝3000H，AX＝1234H，则该指令执行后，将把 AX 寄存器中的 1234H 与物理地址 23400H（＝DS×16＋SI＋0400H＝20000H＋3000H＋0400H）单元开始的数据字（设为 ABCDH）相互交换位置，即 AX＝ABCDH；(23400H)＝34H，(23401H)＝12H。

4. 字节翻译指令

```
XLAT
```

字节翻译指令又称为代码转换指令，用于对不规则代码的转换。其具体功能是：可以将 AL 寄存器中设定的一个字节数值，变换为内存一段连续表格中的另一个相应的代码，以实现编码制的转换。

该指令是通过查表方式完成代码转换功能的，执行的操作是 AL←[BX＋AL]。执行结果是将待转换的序号转换成对应的代码，并送回 AL 寄存器中，即执行 AL←[BX＋AL]的操作。

例如，假设 0～9 的 7 段显示码表存放在偏移地址为 0030H 开始的内存区，如果要取出 5 所对应的 7 段码(12H)，则可以用如下 3 条指令完成：

```
MOV BX,0030H
MOV AL,5
XLAT
```

3.2.2 目标地址传送指令

这是一类专用于 8086/8088 中传送地址码的指令，可传送存储器的逻辑地址（即存储器操作数的段地址或偏移地址）至指定寄存器中。

1. LEA d,s

这是取有效地址指令，其功能是把用于指定源操作数（它必须是存储器操作数）的

16 位偏移地址(即有效地址),传送到一个指定的 16 位通用寄存器中。例如:

```
LEA BX,[SI+100AH]
```

设当前 CS=1500H,IP=0200H,DS=2000H,SI=0030H,源操作数 1234H 存放在[SI+100AH]开始的存储器内存单元中,则该指令执行的结果,是将源操作数 1234H 的有效地址 103AH 传送到 BX 寄存器中。

注意:LEA 指令和 MOV 指令的功能不同,比如 LEA BX,[SI]指令是将 SI 指示的偏移地址(SI 的内容)装入 BX;而 MOV BX,[SI]指令则是将由 SI 寻址的存储单元中的数据装入 BX。

2. LDS d,s

这是取某变量的 32 位地址指针的指令,其功能是从由指令的源 s 所指定的存储单元开始,由 4 个连续存储单元中取出某变量的地址指针(共 4 个字节),将其前两个字节(即变量的偏移地址)传送到由指令的目标 d 所指定的某 16 位通用寄存器,后两字节(即变量的段地址)传送到 DS 段寄存器中。例如:

```
LDS SI,[DI+100AH]
```

设当前 CS=1000H,IP=0604H,DS=2000H,DI=2400H,待传送的某变量的地址指针其偏移地址为 0180H,段地址为 2230H,则该指令执行后,将物理地址 2340AH 单元开始的 4 个字节中前两个字节(偏移地址值)0180H 传送到 SI 寄存器中,后两个字节(段地址)2230H 传送到 DS 段寄存器中,并取代它的原值 2000H。

3. LES d,s

这条指令与 LDS d,s 指令的操作基本相同,其区别仅在于将把由源所指定的某变量的地址指针中后两个字节(段地址)传送到 ES 段寄存器,而不是 DS 段寄存器。例如:

```
LES DI,[BX]
```

设当前 DS=B000H,BX=080AH,B080AH 单元指定的存储字为 05A2H,B080CH 单元指定的存储字为 4000H,该指令执行后,则将某变量地址指针的前两个字节(即偏移地址)05A2H 装入 DI,而将地址指针的后两个字节(即段地址)4000H 装入 ES,于是,DI=05A2H,ES=4000H。

3.2.3 标志位传送指令

标志位传送指令共有 4 条:LAHF、SAHF、PUSHF 和 POPF。

1. LAHF

指令功能:将标志寄存器 FLAGS 的低字节(共包含 5 个状态标志位)传送到 AH 寄存器中。

LSHF 指令执行后，AH 的 D_7、D_6、D_4、D_2 与 D_0 这 5 位将分别被设置成 SF(符号标志)、ZF(零标志)、AF(辅助进位标志)、PF(奇偶标志)与 CF(进位标志)5 位，而 AH 的 D_5、D_3、D_1 这 3 位没有意义。

2. SAHF

指令功能：将 AH 寄存器内容传送到标志寄存器 FLAGS 的低字节。

SAHF 与 LAHF 的功能相反，它常用来通过 AH 对标志寄存器的 SF、ZF、AF、PF 与 CF 标志位分别置 1 或复 0。

3. PUSHF

指令功能：将 16 位标志寄存器 FLAGS 内容入栈保护。其操作过程与前述的 PUSH 指令类似。

4. POPF

指令功能：将当前栈顶和次栈顶中的数据字弹出送回到标志寄存器 FLAGS 中。

PUSHF 与 POPF 这两条指令常成对出现，一般用在子程序和中断处理程序的首尾，用来保护和恢复主程序涉及的标志寄存器内容；必要时可用来修改标志寄存器的内容。

3.2.4 I/O 数据传送指令

1. IN 累加器，端口号

端口号可以用 8 位立即数直接给出；也可以将端口号事先安排在 DX 寄存器中，间接寻址 16 位长端口号(可寻址的端口号为 0～65 535)。IN 指令是将指定端口中的内容输入到累加器 AL/AX 中。其指令如下：

```
IN AL,PORT      ;AL←(端口 PORT),即把端口 PORT 中的字节内容读入 AL
IN AX,PORT      ;AX←(端口 PORT),即把由 PORT 两相邻端口中的字内容读入 AX
IN AL,DX        ;AL←(端口 (DX)),即从 DX 所指的端口中读取一个字节内容送 AL
IN AX,DX        ;AX←(端口 (DX)),即从 DX 和 DX+1 所指的两个端口中读取一个字内容送 AX
```

2. OUT 端口号，累加器

与 IN 指令相同，端口号可以由 8 位立即数给出，也可由 DX 寄存器间接给出。OUT 指令是把累加器 AL/AX 中的内容输出到指定的端口。其指令如下：

```
OUT PORT,AL     ;端口 PORT←AL,即把 AL 中的字节内容输出到由 PORT 直接指定的端口
OUT PORT,AX     ;端口 PORT←AX,即把 AX 中的字内容输出到由 PORT 直接指定的端口
OUT DX,AL       ;端口 (DX)←AL,即把 AL 中的字节内容输出到由 DX 所指定的端口
OUT DX,AX       ;端口 (DX)←AX,即把 AX 中的字内容输出到由 DX 所指定的端口
```

注意：I/O 指令只能用累加器作为执行 I/O 数据传送的机构，而不能用其他寄存器代替。另外，当用直接寻址的 I/O 指令时，寻址范围仅为 0～255，这适用于较小规模的微机系统；当需要寻址大于 255 的端口地址时，则必须用间接寻址的 I/O 指令。

3.3 算术运算类指令

算术运算类指令包括加、减、乘、除与十进制调整5类指令。

3.3.1 加法指令

1. ADD d,s;d←d+s

指令功能：将源操作数与目标操作数相加，结果保留在目标中。并根据结果置标志位。

源操作数可以是8/16位通用寄存器、存储器操作数或立即数；目标操作数不允许是立即数，其他同源操作数。而且不允许两者同时为存储器操作数。例如：

```
ADD WORD PTR[BX+106BH],1234H
```

设当前CS=1000H，IP=0300H，DS=2000H，BX=1200H，则该指令执行后，将立即数1234H与物理地址为2226BH和2226CH中的存储器字3344H相加，结果4578H保留在目标地址2226BH和2226CH单元中。

2. ADC d,s;d←d+s+CF

带进位加法(ADC)指令的操作过程与ADD指令基本相同，唯一的不同是进位标志位CF的原状态也将一起参与加法运算，待运算结束，CF将重新根据结果置成新的状态。

ADC指令一般用于16位以上的多字节数字相加的软件中。

3. INC d;d←d+1

指令功能：将目标操作数当作无符号数，完成加1操作后，结果仍保留在目标中。

目标操作数可以是8/16位通用寄存器或存储器操作数，但不允许是立即数。

注意：对于间接寻址的存储单元加1指令，数据的长度必须用TYPE PTR、WORD PTR或DWORD PTR类型伪指令加以说明，否则，汇编程序不能确定是对字节、字还是双字加1。

另外，INC指令只影响OF、SF、ZF、AF、PF 5个标志，而不影响进位标志CF，故不能利用INC指令来设置进位位，否则程序会出错。

3.3.2 减法指令

1. SUB d,s;d←d−s

指令功能：将目标操作数减去源操作数，其结果送回目标，并根据运算结果置标志位。源操作数可以是8/16位通用寄存器、存储器操作数或立即数；目标操作数只允许是通用寄存器或存储器操作数。并且，不允许两个操作数同时为存储器操作数，也不允许做段寄存器的减法。例如：