

3

微型计算机指令系统

本章学习目标

本章着重介绍 8086/8088 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：

- 8086/8088 CPU 是如何访问数据的（即寻址方式）。
- 寄存器与存储器之间的关系。
- 微处理器进行数据处理常用的命令（指令）。

现在的微型计算机都是冯·诺依曼结构型的，程序和数据事先放在计算机的存储器中。计算机的工作就是执行指令序列。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。每条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“操作数在存储器中的地址”。从指令中找到操作数的方法，就是操作数的寻址方式问题。

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以及汇编语言程序设计的一些基础知识。

3.1 寻址方式

对于一条汇编语言指令来说，所要做的事情有两件：

第一，指出进行什么操作，这由指令操作码来表明。

第二，指出大多数指令涉及的操作数的来源和操作结果送到哪里去，也就是操作数的寻址方式问题。

除了操作数的寻址方式外，指令系统中还有两类指令，它们分别是“转移指令”和“调用指令”，用于指令转移，涉及到转移地址或者调用地址的提供方式，称为程序转移地址的寻址方式。

这样，8086/8088 的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。

由于数据存储的形式不同，其对应的寻址方式也有所不同，现就 8086/8088 CPU 常见的的基本寻址方式分别加以介绍。

3.1.1 与数据有关的寻址方式

讨论操作数的寻址方式，只讨论源操作数或目的操作数之一，因为另一操作数只能是寄存器寻址。

1. 立即寻址 (Immediate Addressing)

立即寻址又称为立即数寻址，其实并不需要进行寻址，因为已经在指令中指定了具体的操作数，称为立即数。

立即操作数可以是 8 位或 16 位，并且是指令的一部分。立即数总是紧跟在指令操作码之后并和操作码一起存放在内存的代码段中，因而立即数总是和操作码一起被放入 BIU 中的指令队列里，在指令执行时不需再由存储器取得数据。使用立即寻址的指令主要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

立即寻址方式示意如图 3-1 所示。

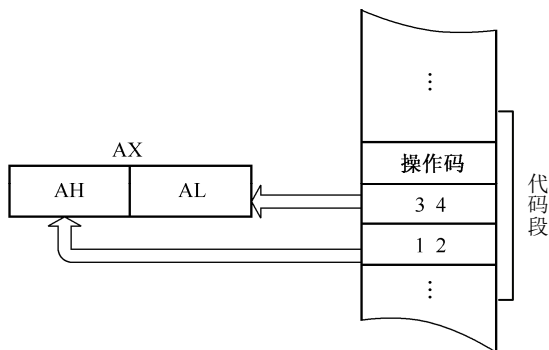


图 3-1 立即寻址方式示意图

2. 直接寻址 (Direct Addressing)

与立即寻址不同, 为了获取指令所指定操作的操作数, 直接寻址必须访问操作数的物理地址[PA], 它是通过段首址左移四位加偏移地址[EA]得到的。

汇编格式: 含有变量的地址表达式或段寄存器名: [EA]

功能: 指令下一字单元的内容是操作数的偏移地址 EA。

操作数的有效地址 EA, 如例 3-2 中 $EA = 2000H$, 是指令的一部分, 它与操作码一起存放在代码段中, 但操作数一般是在数据段中。

【例 3-2】`MOV AL, DS: [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 $56H$ 传至 AL 寄存器。

直接寻址方式的物理地址 $PA = (\text{段基址}) \times 10H + EA$

【例 3-3】^注 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 = 段基址 ($4000H$) $\times 10H$ (左移四位) + 偏移地址 ($3000H$), 由此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。如图 3-2 所示。

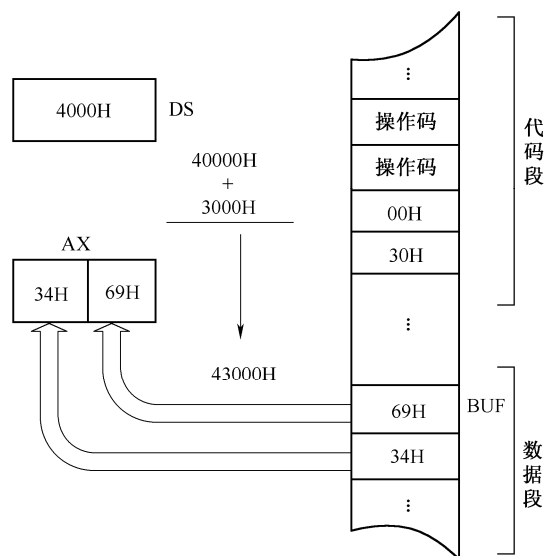


图 3-2 直接寻址方式示意图

注: 例 3-3 题目中有 “ $(43000H) = 3469H$ ” 字样, 严格说来, 应该写成 “ $(43000H) = 69H$, $(43001H) = 34H$ ”。本章中所有例题中的类似说法, 都应该如此理解。

说明:

- 1) 操作数的偏移地址与操作符一起存放在存储器代码段中。
- 2) 操作数可存放在数据段中, 也可存放在其他段中。
- 3) 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, 并在常量前指明相应的段寄存器名, 如例 3-2 中的“DS:[2000H]”。

3. 寄存器寻址 (Register Addressing)

参与指令所指定操作的操作数就存放在指定的寄存器中。

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 送入寄存器 AX 中。第一条指令是立即寻址, 第二条指令才是寄存器寻址。

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; 而对于 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。

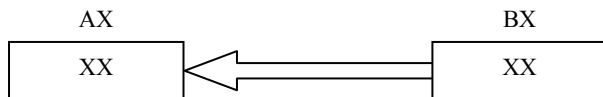


图 3-3 寄存器寻址方式示意图

4. 寄存器间接寻址 (Register Indirect Addressing)

寄存器间接寻址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 位之后的段基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 的内容为操作数的偏移地址 EA。

【例 3-5】设 (DS) = 4000H, (SS) = 3000H, (BX) = 0100H, (BP) = 2000H, (40100H) = 3425H, (32000H) = 8765H。问执行指令

MOV AX, [BX]

MOV CX, [BP]

后的结果?

答: 第一条指令中寄存器 BX 提供的是操作数的偏移地址 (0100H), 加上左移 4 位之后

的数据段的基址得到操作数的物理地址, 即 $PA=4000H \times 10H + 0100H = 40100H$, 然后从 40100H 物理地址中取出操作数 3425H 赋给寄存器 AX。寄存器间接寻址方式示意如图 3-4 所示。

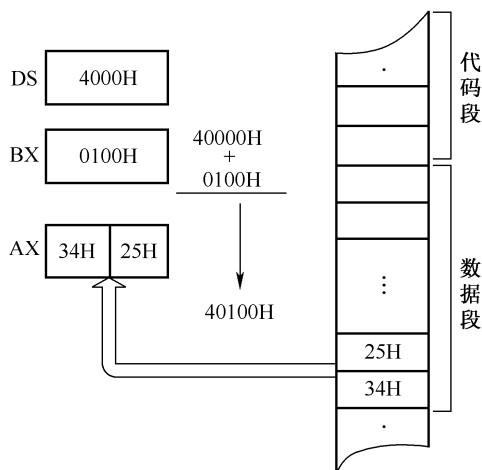


图 3-4 寄存器间接寻址方式示意图

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 位之后的堆栈段的基址得到操作数的物理地址, 即 $PA = 3000H \times 10H + 2000H = 32000H$, 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通用寄存器 BX、基址指针寄存器 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, 则操作数默认在数据段 (DS) 中; 如果指令指定的寄存器为 BP, 则操作数默认在堆栈段 (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: 2)、3)、4) 也适用于以后所讲的各种寻址方式之中。

5. 直接变址寻址 (Indexed Addressing)

直接变址寻址, 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 表示位移量, 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, (20206H) = 5432H。问执行指令

MOV AX, 8[SI]

ADD 6[BP], AX

后的结果?

答：第一条指令中的 8[SI]也可以写成[SI+8]。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 位之后的数据段基址得到操作数的物理地址，其计算过程为 $PA=1000H \times 10H + 0108H = 10108H$ ，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。直接变址寻址方式示意如图 3-5 所示。

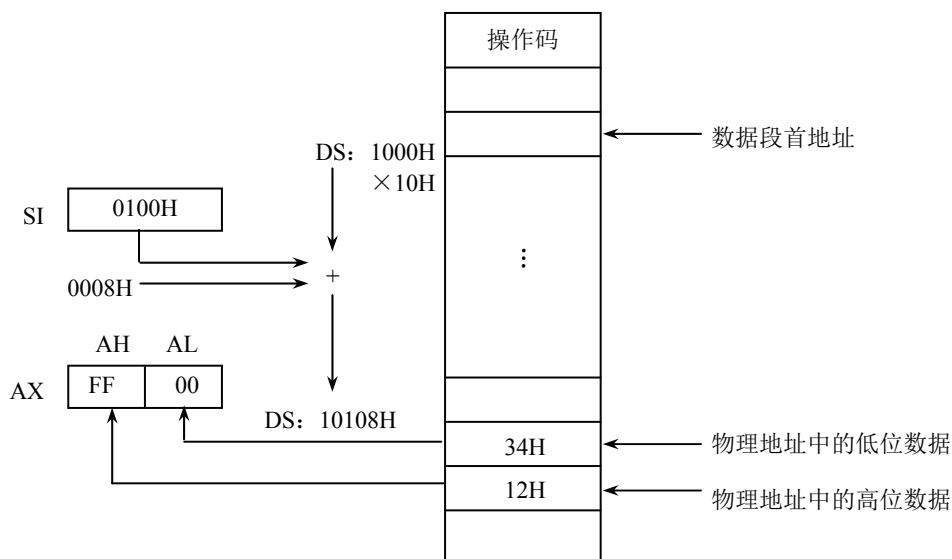


图 3-5 直接变址寻址方式示意图

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 位之后的堆栈段基址得到操作数的物理地址，其计算过程 $PA = 2000H \times 10H + 0206H = 20206H$ ，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) 在变址寻址中，指令指定的寄存器一定要加[]括号。

6. 基址变址寻址 (Based Indexed Addressing)

偏移地址 EA 为基址寄存器 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。

汇编格式: [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 2000H 与 SI 的内容 0006H 之和作为操作数的有效地址。加上左移 4 位之后的数据段基址 5000H 得到操作数的物理地址, 其计算过程为 $PA=5000H \times 10H + 2006H = 52006H$, 然后在操作数的物理地址当中取出操作数 AB78H 赋给寄存器 AX。传送数据段中的一个字。

基址变址寻址方式示意如图 3-6 所示。

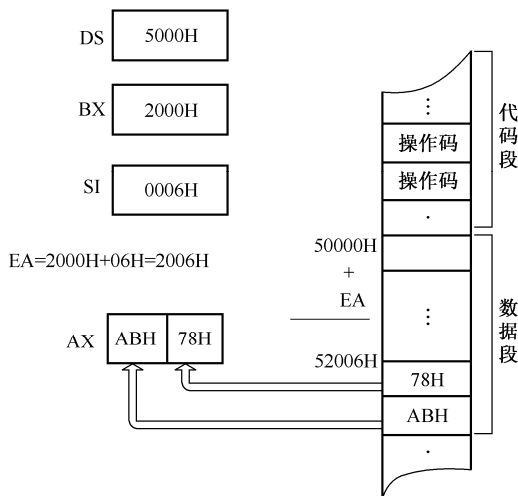


图 3-6 基址变址寻址方式示意图

7. 相对基址加变址寻址 (Relative Based Indexed Addressing)

相对基址变址寻址是在变址寻址方式中多了一项, 即操作数的偏移地址 EA 是由指令中的基址寄存器内容、变址寄存器内容、位移量三项相加组成的。

汇编格式: X[BX+DI] 或写成 [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 (BX) = 0200H, (DI) = 0010H, (DS) = 4000H, (SS) = 3000H, (41444H) = AB50H, (30706H) = 1234H。 (BP) = 0400H, (SI) = 0300H, 问执行指令

MOV AX, 1234H[BX+DI]

ADD 6[BP+SI], AX

后的结果?

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 DI 提供的 0010H 值再加上位移量 1234H 才是操作数的偏移地址 $EA = 0200H + 0010H + 1234H = 1444H$, 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 $PA = 4000H \times 10H + 1444H = 41444H$ ，然后在 41444H 地址中读取相应操作数 AB50H 送到 AX 寄存器。相对基址加变址寻址方式示意如图 3-7 所示。

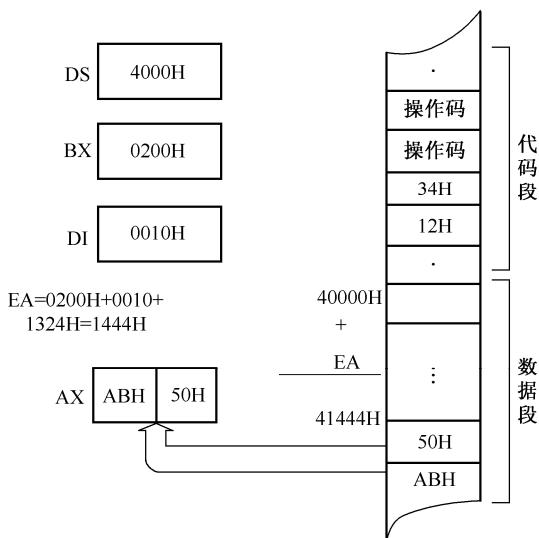


图 3-7 相对基址加变址寻址方式示意图

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 $PA = 3000H \times 10H + 0706H = 30706H$ ，然后在 30706 地址中读取相应操作数 1234H 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。

3.1.2 转移地址的寻址方式

通常指令是顺序执行的，但有时需要跳转到另一地方执行。指令跳转分为本代码段内跳转或跳到其他代码段（段间跳转）。本节讨论指令转移的转移地址寻址方式。

这种寻址方式用来确定转移指令（详见 3.2.6 节）及 CALL（详见 3.2.8 节）指令的转向地址。

1. 段内直接寻址

段内直接寻址方式也称为相对寻址方式，转移的目标地址是：相对于当前指令指针 IP 的一个 8 位或 16 位的位移量，再加上 IP 值之和，所以叫相对寻址。这种寻址方式适用于条件转

移或无条件转移。但条件转移指令只能有 8 位的位移量。无条件转移的位移量可以是 8 位, 也可以是 16 位。无条件转移的位移量是 8 位时, 称为短转移。机器指令中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计算得出, 汇编指令只需给出符号地址即可。如图 3-8 所示。

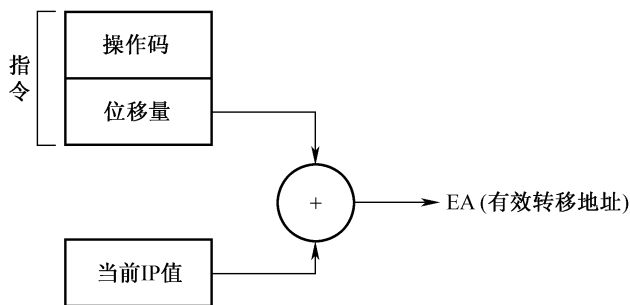


图 3-8 段内直接寻址方式示意图

【例 3-9】 `JMP NEAR PTR ADD1` ; ADD1 为一转向符号地址, 处于代码段内。
说明: 这条指令中, ADD1 指明位移量为 16 位的二进制补码, 前面必须加“NEAR PTR”。

$$(IP) = (IP) + (ADD1) \quad (CS \text{ 不变})$$

【例 3-10】 `JMP SHORT QUEST` ; QUEST 为一转向符号地址, 处于代码段内。
说明: 这条指令中, QUEST 指明位移量为 8 位二进制补码, 前面必须加“SHORT”。

$$(IP) = (IP) + (QUEST) \quad (CS \text{ 不变})$$

2. 段内间接寻址

这种方式也是在段内, 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, 注意和段内直接方式的相对偏移量的区别。若目标地址的偏移量为“存储单元”内容, 则该存储单元本身可由上述与存储器操作数有关的任何寻址方式寻址, 只是它里面的内容为新的 IP 值。

段内间接寻址方式示意如图 3-9 所示。这种方式不能用于条件转移指令。

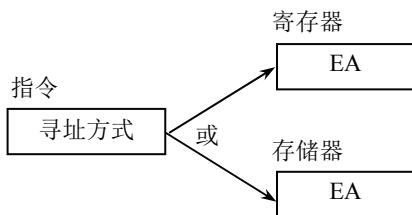


图 3-9 段内间接寻址方式示意图

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

`JMP BX`

```

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

```

后的结果？

答：第一条指令是将 BX 的内容作为新的 IP 值，送给 IP 寄存器。即

$(IP) = (BX)$ (CS 不变)

第二条指令是将 BX 的内容加位移量 TABLE 的值作为新的 EA 值，送给 IP 寄存器。即

$(IP) = ((DS) \times 10H + (BX) + \text{位移量}) = (20000 + 1256 + 20A1) = (232F7) = 3280H$
(CS 不变)

第三条指令是将寄存器 BP 内容加位移量 TABLE 的值作为堆栈段（请读者想一想，为什么是堆栈段？）中偏移量，该存储器字的内容作为新的 IP 值。即

$(IP) = ((SS) \times 10H + (BP) + \text{位移量}) = (30000 + 0100 + 20A1) = (321A1) = 3487H$
(CS 不变)

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的偏移地址，也就是说它是一种段内转移。

3. 段间直接寻址

这种方式用于段间转移，目标转向地址的段基值（CS）和偏移地址（IP）都是指令码的组成部分，用来更新当前 CS 和 IP 的内容。

段间直接寻址方式示意如图 3-10 所示。

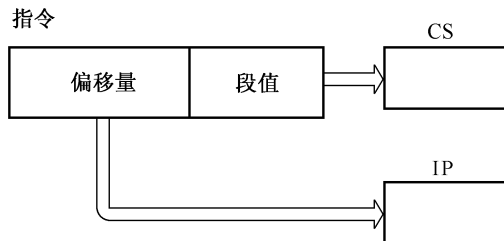


图 3-10 段间直接寻址方式示意图

【例 3-12】JMP FAR PTR NEXTADD ; NEXTADD 是另一段的符号地址。
(CS) 和 (IP) 全都更新。

4. 段间间接寻址

这种方式同样用于段间转移，只不过当前 CS 和 IP 由存储器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。

段间间接寻址方式示意如图 3-11 所示。

【例 3-13】JMP DWORD PTR [INTER+BX] ; 取 DS 段中偏移为[INTER+BX]处的双字作为新的 CS 和 IP

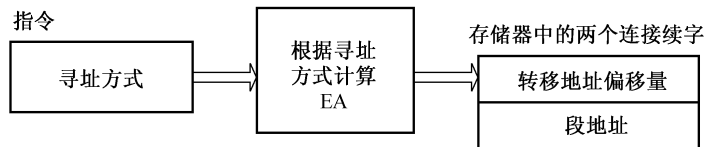


图 3-11 段间间接寻址方式示意图

3.2 8086/8088 处理器的指令系统

8086/8088 指令按功能分类可以分为：数据传送指令、算术运算指令、逻辑运算和移位指令、控制转移指令以及处理器控制指令，下面就对各类常用的指令作详细介绍。

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：数据传送指令又可以分为：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、查表指令、输入输出指令。

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 ; 通用寄存器之间传送
 MOV AL, 3 ; 立即数与通用寄存器之间传送
 MOV AX, [DI] ; 专用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 (除 CS 外) 和存储单元。
- 3) 禁止：立即数→段寄存器；存储器→存储器；段寄存器→段寄存器。
- 4) MOV 指令不改变 SRC；不影响标志位。

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】
 XCHG [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

地址传送指令又有三条指令。

(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 位地址指针段基址送到数据段寄存器 DS, 把偏移地址送到通用寄存器 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 位通用寄存器。

2) OPRD 表示双字的各种寻址方式的存储器操作数的段基址。

(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 位地址指针的段基址送到附加段寄存器 ES, 把偏移地址送到通用寄存器 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段基址。

4. 堆栈操作指令

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, 其中地址较大的为栈底, 地址较小的为栈顶 (进行数据插入和删除操作的一端)。堆栈的段值存放在段寄存器 SS 中, 堆栈指针 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 (PUSH 只对字操作)。

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, 堆栈指针 SP 随着压栈而减小。

【例 3-18】 PUSH SI ; 把寄存器 SI 的内容压入堆栈

PUSH DS ; 把寄存器 DS 的内容压入堆栈

PUSH [SI] ; 把由寄存器 SI 的内容指出地址的内存单元的内容压入堆栈

说明: 数据进入堆栈的时候遵守“高高低低”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 (POP 只对字操作)。

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 可以是通用寄存器以及段寄存器 (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: IN AL, 端口地址或 IN AX, 端口地址

功能: 从 8 位端口读入一个字节到 AL 寄存器中, 或从 16 位端口读入一个字到 AX 寄存器中。

(2) OUT I/O 端口输出指令。

格式: 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, 或将 AX 中的一个字写到一个 16 位端口。

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, SRC 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 3-20】执行如下指令:

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、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 DST 中。

【例 3-21】设 CF=0, 则执行如下指令:

MOV AX, 4653H ; 类似于 (AX) = 4653H

ADD AX, 0F0F0H ; 类似于 (AX) = (AX) + 0F0F0H (产生进位 CF = 1)

MOV DX, 0234H ; 类似于 (DX) = 0234H

ADC DX, 0F01H ; 类似于 (DX) = (DX) + 0F01 + (CF) (求产生进位 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 的内容自加 1 之后, 结果存入原地址。

【例 3-22】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 3-23】设 (DS) = 3000H, (SI) = 0050H, (30064) = 4336H, 则执行指令:

MOV AX, 0136H ; 类似于 (AX) = 0136H

SUB [SI+14H], AX ; 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 $PA = 3000H \times 10H + 0064H = 30064H$, 然后将 30064H 地址中的 4336H 操作数减 AX 寄存器中的操作数 0136H, 即 $4336H - 0136H = 4200H$, 最后将结果再存入 30064 物理地址当中。

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、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 3-24】设 CF=0, DSUB 为定义的双字变量, 初值为 0, 则执行指令:

MOV AX, 2F65H ; 类似于 (AX) = 2F65H

SUB AX, 2F65H ; 类似于 (AX) = 2F65H - 3412H (需借位, 产生 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, 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 3-25】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 3-26】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、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, 比较大小, 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、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, 高位字送 DX 寄存器, 低位字送 AX 寄存器。

【例 3-27】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、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 3-28】设 (AL) = 0B4H, (BL) = 11H, 则执行指令:

IMUL BL ; 类似于 $(AX) = (0B4H) \times (11H) = FAF4H$

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、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 3-29】设 $(AX) = 0400H$, $(BX) = 0C8H$, 则执行指令:

DIV BX ; 类似于 $(AL) = 05H$, $(AH) = 18H$

分析: AX 寄存器中的内容 0400H 是无符号十进制数 1024D, BX 寄存器中的 0C8H 是无符号十进制数 200D, 将 AX 内容与 BX 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、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 3-30】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, 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 3-31】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: $1 \text{ AND } 1 = 1$, $1 \text{ AND } 0 = 0$, $0 \text{ AND } 0 = 0$, $0 \text{ AND } 1 = 0$, 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: $1 \text{ OR } 1 = 1$, $1 \text{ OR } 0 = 1$, $0 \text{ OR } 0 = 0$, $0 \text{ OR } 1 = 1$, 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: $1 \text{ XOR } 1 = 0$, $1 \text{ XOR } 0 = 1$, $0 \text{ XOR } 0 = 0$, $0 \text{ XOR } 1 = 1$, 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 的内容逐位取反之后将结果送 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, 最低位补入相应个数的 0, CF 的内容为最后移入位的值。

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。



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 (以保证符号不变), CF 的内容为最后移入位的值。

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。

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。

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。

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。



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。

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:

MOVS DST, SRC

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

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 再自加/减 1; 如果是字操作则将 AX 的内容存入[DI], 然后 DI 再自加/减 2。

(2) 与 MOVS 指令第 2) 点说明相同。

3. LODS 取串指令

格式:

LODS SRC

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 再自加/减 1; 如果是字操作则将[SI]送入 AX, 然后 SI 再自加/减 2。

(2) 与 MOVS 指令第二点说明相同。

4. CMPS 串比较指令

格式:

CMPS SRC, DST

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:

SCAS DST

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 MOVSB, LODSB 或 STOSB)

功能：用 CX 计重复次数，每重复一次减 1。当 CX = 0 时不执行 *strpri* 给定的指令，否则继续执行。

7. REPE/REPZ 重复前缀指令

格式：REPE（或 REPZ） *strpri* (*strpri* 可为 CMPSB 或 SCASB)

功能：用 CX 计重复次数，每重复一次减 1。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 CMPSB 或 SCASB)

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

（1）段内直接短转移。

格式：JMP SHORT OPR

功能：指令在同一代码段的范围之内进行转移，只需改变 IP 寄存器的内容，即用新的转移目标地址代替原有的 IP 的值就可达到转移的目的。

执行的操作： $(IP) \leftarrow (IP) + 8 \text{ 位移量}$ 。

（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。

（3）段内间接转移。

格式：JMP WORD PTR OPR

功能：跳转到由 OPR 的寻址方式确定的有效地址 EA 中，执行该地址中的相关指令。

执行的操作： $(IP) \leftarrow (EA)$ 。

（4）段间直接转移。

格式：JMP FAR PTR OPR

执行的操作： $(IP) \leftarrow (OPR \text{ 的代码段内偏移地址})$

(CS) ← (OPR 所在代码段的段基址)

(5) 段间间接转移。

格式: JMP DWORD PTR OPR

执行的操作: (IP) ← (EA)

(CS) ← (EA+2)

2. 条件转移指令

(1) 根据单个条件标志的设置情况进行指令转移。

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, 或者不高于或等于, 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, 或者高于或者等于, 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)

在条件转移指令 (1) 9) 中已述。

2) JNB (或 JAE, JNC)

在条件转移指令 (1) 10) 中已述。

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, 或者高于则转移。

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 JNGE) 小于, 或者不大于或者等于则转移。

格式: JL (或 JNGE) 标号

测试条件: SF 异或 $OF = 1$

2) JNL (或 JGE) 不小于, 或者大于或者等于则转移。

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, 或者不大于则转移。

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, 或者大于则转移。

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

这三条指令的步骤是:

- (1) 让 CX 的内容减 1, 即 $(CX) \leftarrow (CX) - 1$;
- (2) 再检查是否满足测试条件, 如满足则 $(IP) \leftarrow (IP) + D8$ 的符号扩充。

3.2.8 过程调用和返回指令

过程调用和返回指令的作用为: 调用子程序并将子程序加工之后的结果返回。

1. CALL 过程调用指令

格式: CALL 过程名

2. RET 结束返回指令

格式: RET

3.2.9 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

执行的操作:

- | | |
|-------------------------------------|-------------------|
| $(SP) \leftarrow (SP) - 2$ | |
| $((SP) + 1, (SP)) \leftarrow (PSW)$ | ; 相应状态标志位入栈 |
| $(SP) \leftarrow (SP) - 2$ | |
| $((SP) + 1, (SP)) \leftarrow (CS)$ | ; 代码段地址入栈 |
| $(SP) \leftarrow (SP) - 2$ | |
| $((SP) + 1, (SP)) \leftarrow (IP)$ | ; IP 中断地址入栈 |
| $(IP) \leftarrow (TYPE * 4)$ | ; 每个中断向量占 4 个字节单元 |
| $(CS) \leftarrow (TYPE * 4 + 2)$ | |

2. INTO 结果溢出中断指令

执行的操作: 若 OF = 1, 则:

- $(SP) \leftarrow (SP) - 2$

$((SP) + 1, (SP)) \leftarrow (PSW)$

$(SP) \leftarrow (SP) - 2$

$((SP) + 1, (SP)) \leftarrow (CS)$

$(SP) \leftarrow (SP) - 2$

$((SP) + 1, (SP)) \leftarrow (IP)$

3. IRET 中断返回指令

格式: IRET

执行的操作:

$(IP) \leftarrow ((SP) + 1, (SP))$

$(SP) \leftarrow (SP) + 2$

$(CS) \leftarrow ((SP) + 1, (SP))$

$(SP) \leftarrow (SP) + 2$

$(PSW) \leftarrow ((SP) + 1, (SP))$

$(SP) \leftarrow (SP) + 2$

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 $CF \leftarrow 0$ 。

(2) CMC 进位位求反指令, 即 $CF \leftarrow (CF \text{ 取反})$ 。

(3) STC 进位位置 1 指令, 即 $CF \leftarrow 1$ 。

(4) CLD 方向标志置 0 指令, 即 $DF \leftarrow 0$ 。

(5) STD 方向标志置 1 指令, 即 $DF \leftarrow 1$ 。

(6) CLI 中断标志置 0 指令, 即 $IF \leftarrow 0$ 。

(7) STI 中断标志置 1 指令, 即 $IF \leftarrow 1$ 。

2. 其他处理机控制指令

作用: 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, 其机器码占有一个字节, 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 指令继续执行。

(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 以及 Pentium 系列都继承了 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 2 个附加数据段寄存器: FS 和 GS,其长度还是 16 位。扩展之后的 32 位标志寄存器,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:

- 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

所谓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，指令计数器在 8086 系统中，由指令指针 IP 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、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