第3章 微处理器的指令系统

【学习目标】

8086/8088 CPU 的指令系统是 Intel 80x86 系列 CPU 共同的基础，其后续高型号微处理器的指令系统都是在此基础上新增了一些指令逐步扩充形成的。同时，它也是目前应用范围最广的一种指令系统。因此，本章重点讨论 8086/8088 CPU 的指令系统。

通过本章对 8086/8088 CPU 寻址方式和指令系统的学习，应该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 的设置和进位位的处理。
- 学习串操作类指令时，着重理解重复前缀的使用。
- 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。

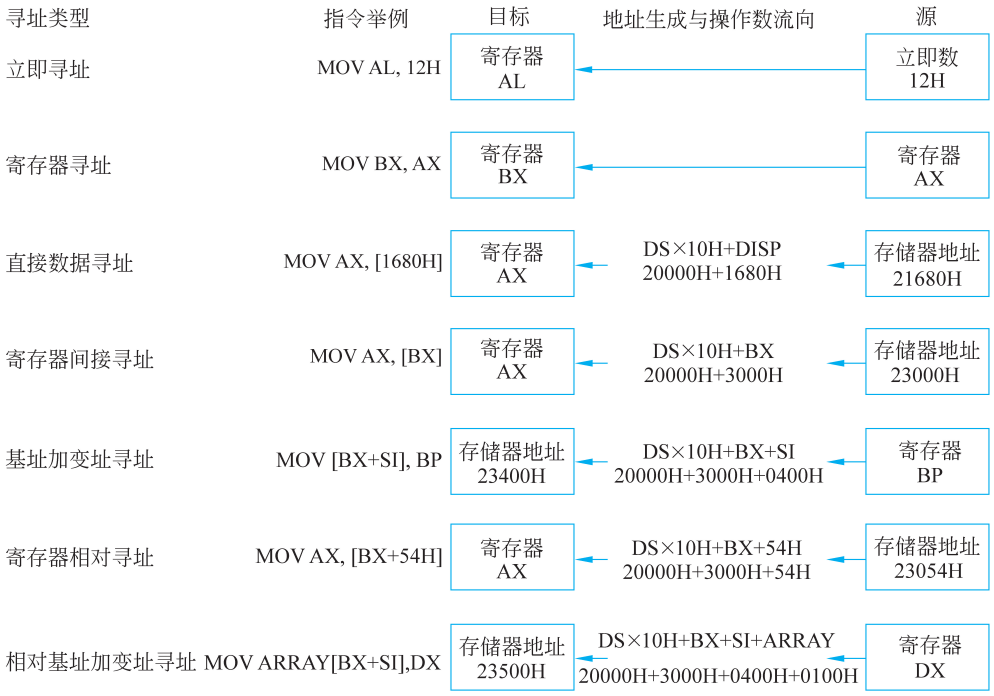
3.1 8086/8088 的寻址方式

指令格式包括操作码和操作数(或地址)两部分，根据操作码所指定的功能去寻找操作数所在地址的方式就是寻址方式。要熟悉指令的操作首先要了解寻址方式。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，图 3-1 给出了各种数据寻址方式的类型、指令举例以及存储器地址生成方法与数据流向，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）。下面将分别对各种寻址方式给予更详细的说明。



注：BX=3000H, SI=0400H, ARRAY=0100H, DS=2000H

图 3-1　8086/8088 数据寻址方式

1. 立即寻址

立即寻址是将立即数传送到目标寄存器或存储器中。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（经队列缓冲器）取得该立即数，而不必执行总线周期。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。表 3-1 列出了各种立即数寻址的 MOV 指令。

2. 寄存器寻址

寄存器寻址是最通用的数据寻址方式。其操作数就放在 CPU 的寄存器中，而寄存器名在指令中指出。对 16 位操作数来说，寄存器可以为 8 个 16 位通用寄存器，而对 8 位操作数来说，寄存器只能为 AH、AL、BH、BL、CH、CL、DH、DL。在一条指令中，源操作数或/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 内部进行，不需要使用总线周期，所以执行速度快。注意，使用时源与目标操作数应有相同的数据类型长度。

表 3-2 列出了各种寄存器寻址的 MOV 指令。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，这可能引起系统运行的紊乱。

表 3-1　使用立即寻址的 MOV 指令示例

汇 编 语 句	长度/位	操　　作
MOV AH,4CH	8	把 4CH 传送到 AH 中
MOV AX,1234H	16	把 1234H 传送到 AX 中
MOV DI,0	16	把 0000H 传送到 DI 中
MOV CL,100	8	把 100(64H)传送到 CL
MOV AI,'A'	8	把 ASCII 码 A(41H)传送到 AL 中
MOV AX,'AB'	16	把 ASCII 码 BA* (4241H)传送到 AX 中
MOV CL,10101101B	8	把二进制数 10101101 传送到 CL 中
MOV WORD PTR [SI], 6180H	16	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

注：* 'AB'在内存中的数据结构为 ASCII 码 BA。

表 3-2　使用寄存器寻址的 MOV 指令示例

汇 编 语 句	长度/位	操　　作
MOV AL,BL	8	把 BL 复制到 AL 中
MOV BH,BL	8	把 BL 复制到 BH 中
MOV CX,AX	16	把 AX 复制到 CX 中
MOV SP,BP	16	把 BP 复制到 SP 中
MOV DI,SI	16	把 SI 复制到 DI 中
MOV AX,ES	16	把 ES 复制到 AX 中

下面将讨论属于存储器寻址的各种寻找方式。指令系统中采用的复杂的寻址方式主要是针对存储器操作数而言的。当 CPU 寻找存储器操作数时，必须先经总线接口单元(BIU)的总线控制逻辑电路进行存取。当执行单元(EU)需要读写位于存储器的操作数时，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)，并将它送给 BIU，同时请求 BIU 执行一个总线周期，BIU 将某个段寄存器的内容左移 4 位，加上由 EU 送来的偏移量形成一个 20 位的物理地址，然后执行总线周期，读写指令所需的操作数。8086/8088 CPU 所寻址的操作数地址的有效地址 EA，是一个无符号的 16 位地址码，表示操作数所在段的首地址与操作数地址之间的字节距离。所以，它实际上是一个相对地址。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

直接寻址简单、直观，其含义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。

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。

以上两种方式都是把位移量加到默认的数据段地址或其他段地址上形成的。表 3-3 列出了使用 AX、AL 直接寻址的指令示例；表 3-4 列出了使用位移量直接数据寻址的指令示例。

表 3-3　使用 AX、AL 直接寻址的指令示例

汇 编 语 句	长度/位	操　　作
MOV AX,[1680H]*	16	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	16	把数据段存储器地址 NUMBER 中的字内容复制到 AX 中
MOV TWO,AL	8	把 AL 的字节内容复制到数据段存储单元 TWO 中
MOV ES:[3000H],AX	16	把 AX 的字内容复制到附加数据段存储单元 3000H 中
MOV AX,DATA	16	把数据段存储单元 DATA 的字内容复制到 AX 中

注：* 汇编语言中很少采用绝对偏移地址，通常采用符号地址。

表 3-4　使用位移量直接数据寻址的指令示例

汇 编 语 句	长度/位	操　　作
MOV CL,COW	8	把数据段存储单元 COW 的内容(字节)复制到 CL 中
MOV ES,NUMBER	16	把数据段存储器地址 NUMBER 中的内容(字)复制到 ES 中
MOV CX,DATA2	16	把数据段存储单元 DATA2 中的内容(字)复制到 CX 中
MOV DATA3,BP	16	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	16	把数据段存储单元 SUM 的字内容复制到 DI 中
MOV NUMBER,SP	16	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 4B 长而不是 3B 长。

【例 3-1】　MOV CL,[2000H]指令与 MOV AL,[2000H]指令的操作相同，但 MOV CL,[2000H]指令为 4B 长，而 MOV AL,[2000H]指令为 3B 长。

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[]。

【例 3-2】　设 BX=3000H，DS=2000H，当执行 MOV AX,[BX]指令后，则数据段存储单元为 23000H 处的字内容将被复制 AX 中，即 23000H 的内容送到 AL，23001H 的内容送到 AH。指令中的方括号[]在汇编语言中表示间接寻址。表 3-5 给出了寄存器间接寻址的指令示例。

表 3-5　寄存器间接寻址的指令示例

汇 编 语 句	长度/位	操　　作
MOV AL,[BX]	8	把数据段中以 BX 作为有效地址的存储单元的内容(字节)复制到 AL 中
MOV [SI],BL	8	把寄存器 BL 的内容复制到数据段以 SI 作为有效地址的存储单元
MOV CX,[DX]	16	把数据段由 DX 寻址的存储单元的内容(字)复制到 CX 中
MOV [BP],CL*	8	把寄存器 CL 的内容复制到堆栈段以 BP 作为有效地址的存储单元中
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

注：* 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。

当使用 BX、DI 和 SI 寻址存储器时,寄存器间接寻址或任何其他寻址方式都默认使用数据段,而使用基址指针寄存器 BP 寻址存储器时,则默认使用堆栈段。

在使用寄存器间接寻址时,要注意在某些情况下,要求用指定的类型运算伪指令 BYTE PTR、WORD PTR 或 DWORD PTR 来规定传送数据的长度。

【例 3-3】 MOV AL,[SI]指令的书写格式是对的,因为汇编程序能够清楚地根据 AL 来判明[SI]是指定存储器数据为字节传送类型。

【例 3-4】 MOV [SI],6AH 指令的书写格式是模糊的,因为汇编程序不能根据立即数 6AH 确定[SI]存储单元的数据类型的长度。如果将此指令书写成 MOV BYTE PYR[SI],6AH,则汇编程序就能清楚地判明 SI 所寻址的存储单元为字节类型。

5. 基址加变址寻址

基址加变址寻址类似于间接寻址,它也是间接地寻址存储器数据。其操作数的有效地址 EA 是一个基址寄存器(BX 或 BP)的内容与一个变址寄存器(SI 或 DI)的内容之和。

【例 3-5】 MOV [BX+SI],CL 指令是将寄存器 CL 中的字节内容复制到数据段中由 BX 加 SI 寻址的存储单元中。

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 寄存器寻址堆栈段存储器数组,则由 BP 寄存器和变址寄存器两者生成有效地址。

【例 3-6】 当执行指令 MOV DX,[BP+SI]时,若 BP=2000H,SI=0300H,SS=1000H,则指令执行后,将把堆栈段中 12300H 单元的字数据传送到 DX 寄存器。表 3-6 给出了基址加变址寻址的指令示例。

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。

【例 3-7】 在 MOV AX,[SI+4000H]指令中,假设 SI=0500H,DS=2000H,则指令执行时,微处理器按段加偏移寻址机制得到的有效地址为 EA=SI+4000H=4500H,再加上 DS×10H=20000H,生成所寻址的存储器物理地址为 24500H,于是,指令执行后将把数据段存储单元 24500H 中的字内容送到 AX。表 3-7 给出了寄存器相对寻址的指令示例。

表 3-6　基址加变址寻址的指令示例

汇 编 语 句	长度/位	操　　作
MOV CL,[BX+SI]	8	把以 BX+SI 作为有效地址的数据段存储单元的内容(字节)复制到 CL
MOV CX,[BP+DI]	16	把以 BP+DI 作为有效地址的堆栈段存储单元内的内容(字)复制到 CX
MOV [BX+DI],SP	16	把 SP 的内容(字)存入以 BX+DI 作为有效地址的数据段存储单元
MOV [BP+SI],CH	8	把寄存器 CH 的内容(字节)存入以 BP+SI 作为有效地址的堆栈段存储单元
MOV [AX+BX],CX	16	把 CX 中的内容(字)存入以 AX+BX 作为有效地址的数据段存储单元

表 3-7　寄存器相对寻址的指令示例

汇 编 语 句	长度/位	操　　作
MOV CL,[SI+200H]	8	把以 SI+200H 作为有效地址的数据段存储单元的字节内容装入 CL
MOV ARRAY[DI],BL	8	把 BL 中的字节内容存入以 ARRAY+DI 作为有效地址的数据段存储单元
MOV LIST[DI+3],AX	16	把 AX 的字内容存入以 LIST+DI+3 之和作为有效地址的数据段存储单元
MOV AX,ARRAY[BX]	16	把数据段中以 ARRAY+BX 作为有效地址的字内容装入 AX
MOV SI,[AL+12H]	16	把以 AL+12H 作为有效地址的数据段存储单元的字内容装入 SI

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。

【例 3-8】　在 MOV AX,[BX+DI+200H]指令中,设 BX=0100H,DI=0300H,DS=4000H。当指令执行时,先计算出有效地址为 EA=BX+DI+200H=0600H,指令运行后,将把数据段存储单元 40600H 中的字内容装入 AX。表 3-8 给出了相对基址加变址寻址的指令示例。

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。

【例 3-9】　存储器中有一个文件 FILE 包含 A、B、C、D 共 4 个记录,每个记录又包含 10 个元素,如果要求将其中存储在单元 RECA 中的记录 A 的元素 0 复制到记录 D 的元素 4,这时,可以用位移量寻址文件,用基址寄存器 BX 寻址记录,而用变址寄存器 DI 寻址记录中的元素。程序段如下。

```
MOV BX,OFFSET RECA          ;寻址记录 A 的存储单元 RECA
MOV DI,0                    ;寻址单元 0
MOV AL,FILE[BX+DI]          ;取出记录 A 的元素 0
```

```
MOV BX,OFFSET RECD          ;寻址记录 D 的存储单元 RECD
MOV DI,4                    ;寻址单元 4
MOV FILE[BX+DI],AL          ;复制到记录 D 的元素 4 中
```

表 3-8　相对基址加变址寻址的指令示例

汇 编 语 句	长度/位	操　　作
MOV BL,[BX+SI+100H]	8	把以 BX+SI+100H 作为有效地址的数据段存储单元的字节内容装入 BL
MOV AX,ARRAY[BX+DI]	16	把以 ARRAY+BX+DI 之和作为有效地址的数据段存储单元的字内容装入 AX
MOV LIST[BP+DI],BX	16	把 BX 的字内容存入以 LIST+BP+DI 之和作为有效地址的堆栈段存储单元
MOV AL,LIST[BX+DI]	8	把以 LIST+BX+DI 之和作为有效地址的数据段存储单元的字节内容装入 AL
MOV FILE[BP+DI+2],DL	8	把 DL 存入以 BP+DI+2 之和作为有效地址的堆栈段存储单元

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 5 种寻址方式。

3.1.3　堆栈存储器寻址方式

表 3-9 列出了可以使用的一些 PUSH 和 POP 指令的示例。

表 3-9　PUSH 和 POP 指令的示例

汇 编 语 句	操　　作
PUSHF	把标志寄存器 FLAGS 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
PUSH 12ABH	把 12ABH 压入堆栈
POP CS	非法操作
PUSH WORD PTR[BX]	把数据段中由 BX 寻址的存储单元内的字复制到堆栈中
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。因为指令中不必给出 SI 或 DI 的编码,所以串操作指令采用的是隐含寻址方式。

2. I/O 端口寻址方式

在 8086/8088 指令系统中,输入输出指令对 I/O 端口的寻址可采用直接或间接两种方式。

1) 直接端口寻址

这种寻址方式端口地址以 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,它所寻址的端口号范围只能是 0～255 内。

2) 间接端口寻址

这种寻址方式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。

下面将详细讨论 8086/8088 的指令系统。8086/8088 的指令按功能可分为 6 类:数据传送、算术运算、逻辑运算、串操作、程序控制和 CPU 控制。

3.2 数据传送类指令

数据传送类指令可完成寄存器与寄存器之间、寄存器与存储器之间以及寄存器与 I/O 端口之间的字节或字传送,除了 SAHF 和 POPF 指令对标志位有影响外,这类指令所具有的共同特点是不影响标志寄存器的内容。

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

```
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。

前面已介绍过 MOV 指令的使用例子。

注意： MOV 指令的源操作数可以是 8/16 位寄存器、存储器中的某个字节/字或者是 8/16 位立即数；目标操作数不允许为立即数，其他同源操作数。而且两者不能同时为存储器操作数。

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。这种双操作数指令中，必须有一个操作数是寄存器。例如，表 3-2～表 3-8 中所列的各种指令示例。

(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。例如，表 3-1 所列的各种指令示例。

(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 或 ES)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。

【例 3-10】 MOV DS,AX 指令是对的；而 MOV CS,AX 指令是错的。

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来完成这种传送。

【例 3-11】 MOV [SI],[BX]指令是错的；而用以下两条指令是对的。

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

【例 3-12】 要将数据段存储单元 ARRAY1 中的 8 位数据传送到存储单元 ARRAY2 中，用 MOV ARRAY2,ARRAY1 指令是错的；而用以下两条指令则可以完成该操作。

```
MOV AL,ARRAY1
MOV ARRAY2,AL
```

2. 堆栈操作指令 PUSH 和 POP

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。

PUSH 和 POP 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。

【例 3-13】 设当前 CS＝1000H，IP＝0030H，SS＝2000H，SP＝0040H，BX＝2340H，则 PUSH BX 指令的操作过程如图 3-2 所示。

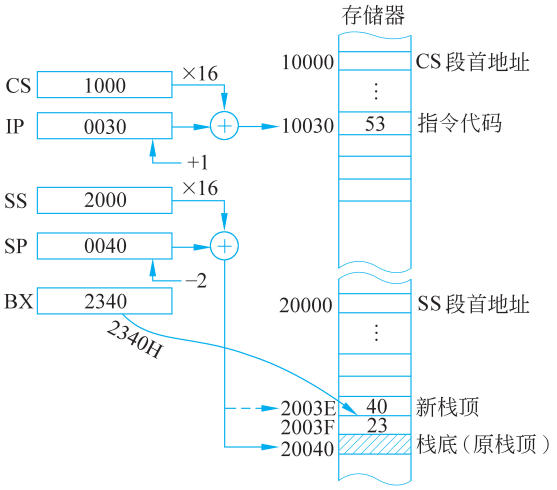


图 3-2 PUSH BX 指令的操作过程

该进栈指令执行时，堆栈指针被修改为 SP－2→SP，使之指向新栈顶 2003EH，同时将 BX 中的数据字 2340H 压入栈内 2003FH 与 2003EH 两单元中。

【例 3-14】 设当前 CS＝1000H，IP＝0020H，SS＝1600H，SP＝004CH，则 POP CX 指令执行时，将当前栈顶两相邻单元 1604CH 与 1604DH 中的数据字弹出并传送到 CX 中，同时修改堆栈指针，SP＋2→SP，使之指向新栈顶 1604EH。

PUSH 和 POP 两条指令可用来保存并恢复现场数据。由于堆栈中的内容是按 LIFO(后进先出)的次序进行传送的，因此，保存内容和恢复内容时，需按照对称的次序执行一系列压入指令和弹出指令。

【例 3-15】 若在一段子程序开头需要这样保存寄存器的内容：

```
PUSH AX
PUSH BX
PUSH DI
PUSH SI
```

则由子程序返回前，应该如下一一对应地恢复寄存器的内容：

```
POP SI
```

```
POP DI
POP BX
POP AX
```

使用堆栈操作指令时应该注意：

(1) 堆栈操作是按字(即两字节)进行的，没有单字节的操作指令。

(2) 每执行一条压入堆栈的指令，堆栈地址指针 SP 减 2，推入堆栈的数据放在栈顶，高位字节先入栈放在较高地址单元，低位字节后入栈放在较低地址单元(真正的栈顶地址单元)；而执行弹出指令时，正好相反，每弹出一个字，栈顶指针的值加 2。

(3) CS 段寄存器的值可以压入堆栈，但却不能从堆栈中弹出一个字到 CS 段寄存器。

3. 数据交换指令 XCHG

```
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。但是，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 3-16】 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 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

```
XLAT
```

字节翻译指令也称为代码转换或查表指令，它特别适合于不规则代码的转换。

该指令通过查表方式完成代码转换功能，执行操作是：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。代码转换的操作步骤如下。

① 建立代码转换表(其最大容量为 256B)，将该表定位到内存中某个逻辑段的一片连续地址中，并将表的首地址的偏移地址置入 BX。

② 将待转换的一个十进制数在表中的序号(又称索引值)送入 AL 寄存器中。该值实际上就是表中某一项与表格首地址之间的位移量。

③ 执行 XLAT 指令。

【例 3-17】 已知 7 段显示码的编码规则为：0——01000000；1——01111001；2——00100100；3——00110000；4——00011001；5——00010010；6——00000010；7——01111000；8——00000000；9——00010000。设有一个十进制数 0～9 的 7 段显示码表被定位在当前数据段中，其起始地址的偏移地址值为 0030H。假定当前 CS=2000H，IP=007AH，DS=4000H。若欲将 AL 中待转换的十进制数 5 转换成对应的 7 段码 12H，试分析执行 XLAT 指令的操作过程。

首先，将数据段中该转换表的首地址的偏移地址 0030H 置入 BX；再将待转换的十进制

数在表中的序号 05H 送入 AL；然后，执行 XLAT 指令。代码转换指令的功能与操作过程如图 3-3 所示。

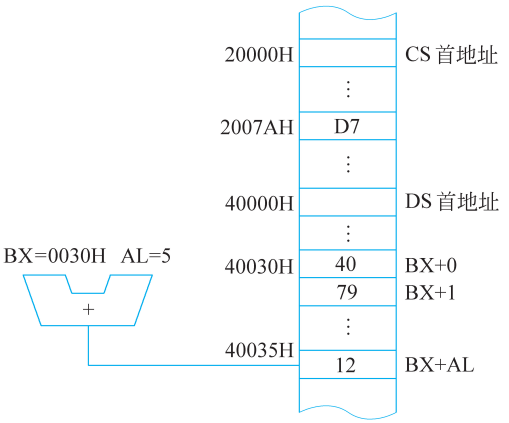


图 3-3 代码转换指令的功能

假设 0～9 的 7 段显示码表存放在偏移地址为 0030H 开始的内存中，则取出 5 所对应的 7 段码(12H)可以用如下 3 条指令的程序段完成。

```
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，共包含 3 条指令：LEA、LDS 和 LES。

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。这条指令常用来建立串操作指令所需要的寄存器指针。

【例 3-18】 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，则该指令的操作过程如图 3-4 所示。

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。

请注意比较 LEA 指令和 MOV 指令的不同功能。

【例 3-19】 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。

通常，LEA 指令用来使某个通用寄存器作为地址指针。

【例 3-20】 LEA BX,[BP+DI]指令是将内存单元的偏移量(BP+DI)送 BX。

LEA SP,[3768H]指令是使堆栈指针 SP 为 3768H。

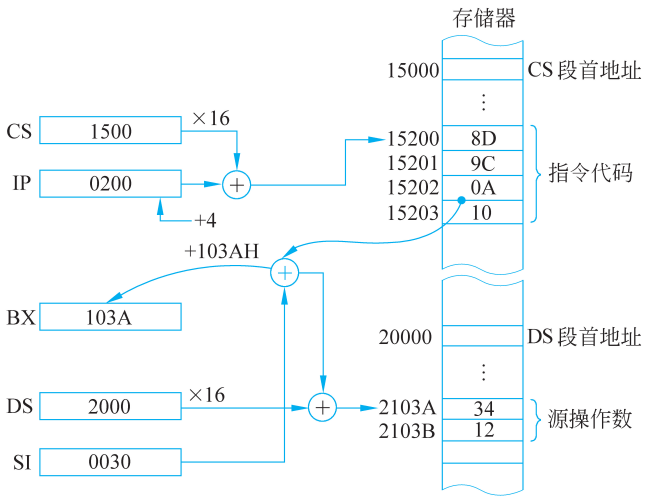


图 3-4　LEA BX,[SI+100AH]指令的操作过程

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 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,后两个字节(即变量的段地址)传送到 DS 段寄存器中。

【例 3-21】　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,则该指令的操作过程如图 3-5 所示。

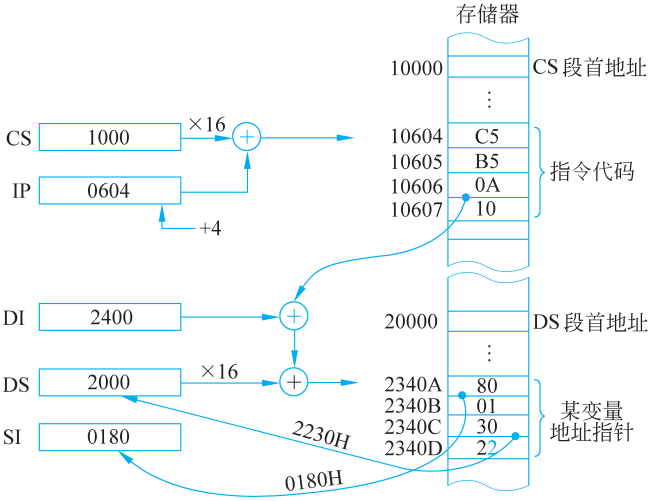


图 3-5　LDS SI,[DI+100AH]指令的操作过程

该指令执行后,将物理地址 2340AH 单元开始的 4 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操作数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。

上述 3 条指令都是装入地址,但使用时要准确理解它们的不同含义。LEA 指令是将 16 位有效地址装入任何一个 16 位通用寄存器;而 LDS 和 LES 是将 32 位地址指针装入任何一个 16 位通用寄存器及 DS 或 ES 段寄存器。

3.2.3 标志位传送指令

这类指令用于传送标志位,共有 4 条标志位传送指令: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。

LAHF 指令执行后,AH 的 D_7、D_6、D_4、D_2、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)、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 FLAGS 的 SF、ZF、AF、PF 与 CF 标志位分别置 1 或复 0。

上述两条指令只涉及对标志寄存器 FLAGS 的低 8 位进行操作,这是为了保持 8086 指令系统对 8088/8085 指令系统的兼容性。

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。

以上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 指令

```
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
```

【例 3-22】 IN AL,40H

设当前 CS=1000H,IP=0050H;8 位端口 40H 中的内容为 55H,则该指令的操作过程如图 3-6 所示。

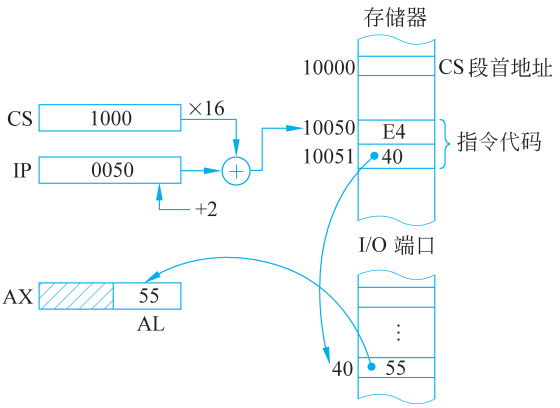


图 3-6 IN AL,40H 指令的操作过程

该指令执行后,把 40H 端口中输入的数据字节 55H 传送到累加器 AL 中。

2. OUT 指令

```
OUT 端口号,累加器
```

与 IN 指令相同,OUT 指令的端口号可以由 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
```

【例 3-23】 OUT DX,AL

设当前 CS=4000H,IP=0020H,DX=6A10H,AL=66H。则该指令的操作过程如图 3-7 所示。

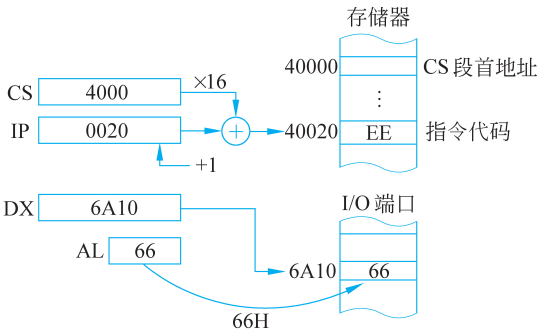


图 3-7 OUT DX,AL 指令的操作过程

该指令执行后,将累加器 AL 中的数据字节 66H 输出到 DX 指定的端口 6A10H 中。

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。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。在 IBM PC/XT 微机系统中,既用了范围为 0～255 的端口地址,也用了范围为 255～65 535 的端口地址。

从以上的讨论中可知,在 IN 和 OUT 指令中,I/O 设备(即 PORT,端口)的地址以两种形式存在:固定的端口和可变的端口。固定端口寻址允许 CPU 在 AL、AX 与使用 8 位 I/O 端口地址的设备之间传送数据。由于端口号在指令中是跟在指令操作码后面,所以称为固定端口寻址。如果固定端口地址存储在 RAM 中,它有可能被修改。

3.3 算术运算类指令

算术运算类指令有加、减、乘、除及十进制调整 5 种指令,它们能对无符号或有符号的 8/16 位二进制数以及无符号的压缩型/非压缩型(又称装配型/拆开型或者组合型/未组合型)十进制数进行运算。

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 3-24】 ADD WORD PTR[BX+106BH],1234H

设当前 CS=1000H,IP=0300H,DS=2000H,BX=1200H,则该指令的操作过程如图 3-8 所示。

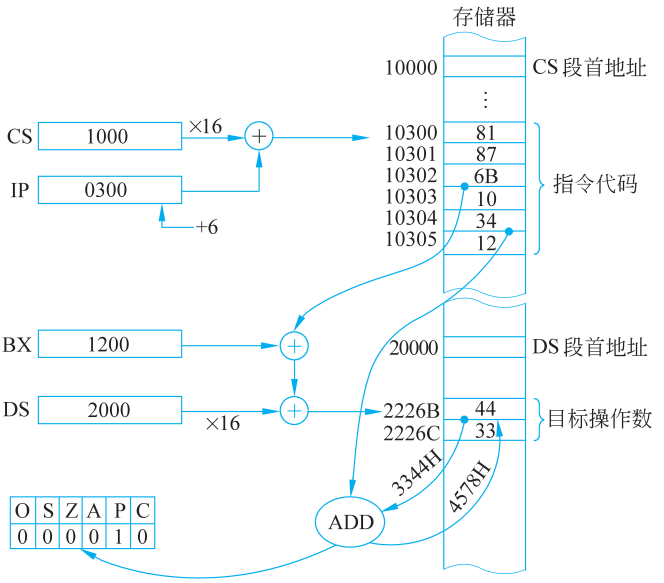


图 3-8　ADD WORD PTR[BX+106BH],1234H 指令的操作过程

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。根据运算结果所置的标志位也示于图左下方。

【例 3-25】　寄存器加法。若将 AX、BX、CX 和 DX 的内容累加，再将所得的 16 位的和数存入 AX，则加法程序段如下。

```
ADD AX,BX      ;AX←AX+BX
ADD AX,CX      ;AX←AX+BX+CX
ADD AX,DX      ;AX←AX+BX+CX+DX
```

【例 3-26】　立即数加法。当常数或已知数相加时总是用立即数加法。若将立即数 12H 取入 DL，然后用立即数加法指令再将 34H 加到 DL 中的 12H 上，所得的结果（即和数 46H）放在 DL 中，则程序段如下。

```
MOV DL,12H
ADD DL,34H
```

程序执行后，标志位的改变为：OF＝0（没有溢出），SF＝0（结果为正），ZF＝0（结果不是 0），AF＝0（没有半进位），PF＝0（奇偶性为奇），CF＝0（没有进位）。

【例 3-27】　存储器与寄存器的加法。假定要求将存储在数据段中其偏移地址为 NUMB 和 NUMB+1 连续单元的字节数据累加到 AL，则加法程序段如下。

```
MOV DI,OFFSET NUMB      ;偏移地址 NUMB 装入 DI
MOV AL,0                ;AL 清 0
ADD AL,[DI]             ;将 NUMB 单元的字节内容加 AL，和数存 AL
ADD AL,[DI+1]           ;累加 NUMB+1 单元中的字节内容，累加和存 AL
```

【例 3-28】　数组加法。存储器数组是一个按顺序排列的数据表。假定数据数组

(ARRAY)包括从元素0～9共10字节数。现要求累加元素3、元素5和元素7,则加法程序段如下。

```
MOV AL,0                ;存放和数的AL清0
MOV SI,3                ;将SI指向元素3
ADD AL,ARRAY[SI]        ;加元素3
ADD AL,ARRAY[SI+2]      ;加元素5
ADD AL,ARRAY[SI+4]      ;加元素7
```

本程序段中首先将AL清0,为求累加和做好准备。然后,把3装入源变址寄存器SI,初始化为寻址数组元素3。ADD AL,ARRAY[SI]指令是将数组元素3加到AL中。接着的两条加法指令是将元素5和7累加到AL中,指令用SI中原有的3加位移量2来寻址元素5,再用加4寻址元素7。

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。例如:

```
ADC AX,BX          ;AX=AX+BX+C(进位位)
ADC BX,[BP+2]      ;由BX+2寻址的堆栈段存储单元的字内容加BX和进位位,结果存入BX
```

ADC指令一般用于16位以上的多字节数字相加的软件中。

【例3-29】 假定要实现BX和AX中的4字节数字与DX和CX中的4字节数字相加,其结果存入BX和AX中,则多字节加法的程序段如下。

```
ADD AX,CX
ADC BX,DX
```

上述多字节相加的程序段中用了ADD与ADC两条不同的加法指令,由于AX和CX的内容相加形成和的低16位时,可能产生也可能不产生进位,而事先又不可能断定有无进位,因此,在高16位相加时,就必须要采用带进位位的加法指令ADC。这样,ADC指令在执行加法时,就会把在低16位相加后产生的进位标志1或0,自动加到高16位的和数中去。最后,程序把BX、AX的4字节内容加到DX、CX两个寄存器,而和数则存入BX、AX两个寄存器中。

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。例如:

```
INC SP                          ;SP=SP+1
INC BYTE PTR[BX+1000H]          ;把数据段中由BX+1000H寻址的存储单元的字节内容加1
INC WORD PTR[SI]                ;把数据段中由SI寻址的存储单元的字内容加1
INC DATA1                       ;把数据段中DATA1存储单元的内容加1
```

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。而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 3-30】 SUB AX,[BX]

设当前 CS＝1000H，IP＝60C0H，DS＝2000H，BX＝970EH，则该指令的操作过程如图 3-9 所示。

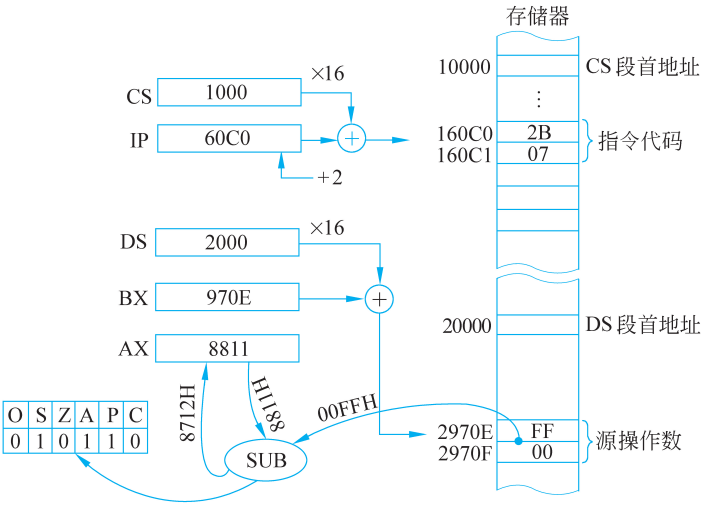


图 3-9 SUB AX,[BX]指令的操作过程

该指令执行后，将 AX 寄存器中的目标操作数 8811H 减去物理地址 2970EH 和 2970FH 单元中的源操作数 00FFH，并把结果 8712H 送回 AX 中。各标志位的改变为：O＝0(没有溢出)，S＝1(结果为负)，Z＝0(结果不为 0)，A＝1(有半进位)，P＝1(奇偶性为偶)，C＝0(没有借位)。

SUB 指令的寻址方式和汇编语句形式也很多，例如：

```
SUB CL,BL                ;CL=CL-BL
SUB AX,SP                ;AX=AX-SP
SUB BH,6AH               ;BH=BH-6AH
SUB AX,0AAAAH            ;AX=AX-0AAAAH
SUB DI,TEMP[SI]          ;从 DI 中减去由 TEMP+SI 寻址的数据段存储单元的字内容
```

2. SBB d,s　　;d←d－s－CF

本指令与 SUB 指令的功能、执行过程基本相同，唯一不同的是完成减法运算时还要再减去进位标志 CF 的原状态。运算结束时，CF 将被置成新状态。这条指令通常用于比 16 位数宽的多字节减法，在多字节减法中，如同多字节加法操作时传递进位一样，它需要传递借位。

SBB 指令的汇编语句形式很多，例如：

```
SBB AX,BX                  ;AX=AX-BX-CF
SBB WOLD PTR[DI],50A0H     ;从由 DI 寻址的数据段字存储单元的内容减去 50A0H 及 CF 的值
SBB DI,[BP+2]              ;从 DI 中减去由 BP+2 寻址的堆栈段字存储单元的内容及借位
```

【例 3-31】　假定从存于 BX 和 AX 中的 4 字节数减去存于 SI 和 DI 中的 4 字节数，则程序段为：

```
SUB AX,DI
SBB BX,SI
```

3. DEC d　　;d←d－1

减 1 指令功能：将目标操作数的内容减 1 后送回目标。

目标操作数可以是 8/16 位通用寄存器和存储器操作数，但不允许是立即数。例如：

```
DEC BL                     ;BL=BL-1
DEC CX                     ;CX=CX-1
DEC BYTE PTR[DI]           ;由 DI 寻址的数据段字节存储单元的内容减 1
DEC WORD PTR[BP]           ;由 BP 寻址的堆栈段字存储单元的内容减 1
```

从以上指令汇编语句的形式可以看出，对于间接寻址存储器数据减 1 指令，要求用 TYPE PTR 类型伪指令来标识数据长度。

4. NEG d　　;d←$\bar{d}$＋1

NEG 是一条求补码的指令，简称求补指令。

指令功能：将目标操作数取负后送回目标。

目标操作数可以是 8/16 位通用寄存器或存储器操作数。

NEG 指令是把目标操作数当成一个带符号数，如果原操作数是正数，则 NEG 指令执行后将其变成绝对值相等的负数（用补码表示）；如果原操作数是负数（用补码表示），则 NEG 指令执行后将其变成绝对值相等的正数。

若 AL＝00000100＝＋4，执行 NEG AL 指令后将各位变反，末位加 1 得 11111100＝$[-4]_{补}$；若 AL＝11101110＝$[-18]_{补}$，执行 NEG AL 指令后将变成 00010010＝＋18。

【例 3-32】　NEG BYTE PTR[BX]

设当前 CS＝1000H，IP＝200AH，DS＝2000H，BX＝3000H，且由目标[BX]所指向的存储单元（＝DS×16＋BX＝23000H）已定义为字节变量（假定为 FDH），则该指令执行后，将物理地址 23000H 中的目标操作数 FDH＝$[-3]_{补}$，变成＋3 送回物理地址 23000H