第 3 章　80x86 的寻址方式与指令系统

3.1　指令系统概述

计算机的指令系统是计算机指令的集合，是计算机硬件的语言系统，程序员通过计算机指令来使用计算机的硬件，按指令系统提供的指令有序地组成程序，完成特定的功能。在计算机中，指令用二进制编码表示，一条条指令对应各种基本操作，某计算机能执行什么操作、具备什么功能，都是由指令系统中的指令来决定的。

指令包括操作码和操作数两大部分，对于 80x86/88 系统，其操作数又包括目的操作数和源操作数。80x86/88 的指令格式如图 3-1 所示。

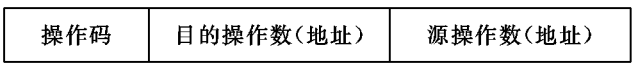

图 3-1　指令的格式

操作码用于指定指令的操作功能，如数据传送指令、程序控制指令等，在汇编语言中，用具有一定意义的助记符来代替二进制编码，方便用户编程使用。

操作数部分针对不同功能的指令，可以是两个、一个或没有。操作数可以是被操作数据本身，更多时候是被操作数据的地址，根据给出的地址遵循 80x86/88 的寻址方式，得到实际的数据参加运算。因此指令一般分为零地址指令、一地址指令、二地址指令等，80x86/88 的目的操作数和源操作数参加运算，其结果送目的操作数，目的操作数原来的数据被覆盖。操作数在计算机中也是用二进制表示的，为方便编程使用，操作数采用编码或编号形式，如：

```
MOV  AX,012FDH
```

其中，MOV 是操作码，该指令表示数据传送功能，AX 是寄存器编号，012FDH 是数据的十六进制形式编码。该指令的作用是把数据 012FDH 送到 AX 寄存器，AX 寄存器原来的数据被覆盖。

用符号化的汇编语言指令编写的程序输入计算机后，由“汇编程序”把它翻译成二进

制机器语言形式，才能在机器上执行。二进制机器语言指令长度是可变的，是由操作码、寻址方式和操作数长短等决定的，一般在1～6个字节间变化。最长、最完整的指令格式如图3-2所示。

图3-2　机器语言指令格式

其中：OP代表操作码，MOD表示寻址方式，DISP代表位移量，DATA代表数据。对于不同功能的指令，除OP字段是必需的，其他字段可多可少可有可无。

OP字段是指令的第一个字节，当8位不够时，还可占用第二个字节的3位，除指定操作功能外，OP字段还包含操作对象的特征信息，其具体的信息如图3-3所示。

图3-3中的d位用在双操作数指令中，因为80x86/88规定双操作数指令的两个操作数必须有一个操作数放在寄存器中，d位用于指定所用寄存器用于目的操作数(d=1)还是源操作数(d=0)；w位用于表示指令对字(w=1)还是字节(w=0)操作。当指令中有立即数，即立即寻址方式，s=1，表示把8位立即数扩展为16位(把低字节最高位的符号位扩展到高字节)。

MOD字段用于表示指令寻址方式，占1个字节，其格式如图3-4所示。

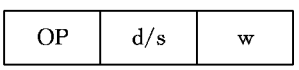

图3-3　OP字段的具体信息

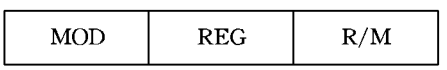

图3-4　MOD字段的具体信息

图3-4中的MOD占2位，用于区分寄存器寻址和存储器寻址，以及使用多少字节的偏移量。具体如下：00=存储器寻址方式，表示无偏移量；01=存储器寻址方式，表示使用1个字节偏移量(−128≤DISP≤127)；10=存储器寻址方式，表示使用2个字节偏移量(0≤DISP≤65535)；11=寄存器寻址方式，表示R/M表示寄存器，并与OP字段的w位一起决定寄存器是8位还是16位。

图3-4中的REG用于选择寄存器，并与OP字段的w位一起决定寄存器是8位还是16位，如表3-1所示。

表3-1　MOD中寄存器的选择

REG或MOD=11的R/M	w=0	w=1	REG或MOD=11的R/M	w=0	w=1
000	AL	AX	100	AH	SP
001	CL	CX	101	CH	BP
010	DL	DX	110	DH	SI
011	BL	BX	111	BH	DI

图3-4中的R/M用于有效地址的形成，如表3-2所示。

表 3-2 R/M 对有效地址形成的规定

MOD / R/M	00	01	10	11		默认段寄存器
				w=0	w=1	
000	(BX)+(SI)	(BX)+(SI)+DISP8	(BX)+(SI)+DISP16	AL	AX	DS
001	(BX)+(DI)	(BX)+(DI)+DISP8	(BX)+(DI)+DISP16	CL	CX	DS
010	(BX)+(SI)	(BX)+(SI)+DISP8	(BX)+(SI)+DISP16	DL	DX	SS
011	(BX)+(DI)	(BX)+(DI)+DISP8	(BX)+(DI)+DISP16	BL	BX	SS
100	(SI)	(SI)+DISP8	(SI)+DISP16	AH	SP	DS
101	(DI)	(DI)+DISP8	(DI)+DISP16	CH	BP	DS
110	直接地址	(SP)+DISP8	(SP)+DISP16	DH	SI	SS
111	(BX)	(BX)+DISP8	(BX)+DISP16	BH	DI	DS

其中,默认段寄存器是没有段跨越前缀时隐含的段寄存器,但如果在指令前指定了段跨越前缀,则使用指定的段寄存器。段跨越前缀的格式如图 3-5 所示。

001	SEG	110

图 3-5 段跨越前缀的格式

其中,001 和 110 是标志位,SEG 的 2 位信息用于指定段寄存器,如表 3-3 所示。

表 3-3 SEG 的 2 位信息对段寄存器的指定

SEG	段寄存器	SEG	段寄存器
00	ES	10	SS
01	CS	11	DS

但段跨越前缀只能用于代码段的指令中,不用堆栈指令中,对串操作指令使用的目的寄存器 DI 只能使用 ES 而不能使用段跨越。

机器语言指令举例:

```
ADD  CL,BH 对应 0000001011001111=02CFH
ADD  AX,0123H 对应 10000001110000000010001100000001=81C0230H
ADD  DISP[BX][DI],DX 对应 00000001100100010100010100100011=0191423H
```

3.2 80x86 的寻址方式

由指令按功能需要有序地组织起来构成程序,执行程序中一条条指令,完成相应的功能。指令有序自动地执行靠的是指令的寻址。在 80x86/88 中,设置指令指针 IP 用于指示当前指令在内存中的位置,IP 起到程序计数器 PC 的作用。随着程序的执行,通过 IP 的变化,指向程序不同位置的指令执行。不管程序是按顺序执行,还是遇到转移的情况,

IP 始终指示正在执行的指令。

指令中操作数的寻址，即如何获得操作数地址是一个重要的问题。80x86/88CPU 的数据一般存放在指令操作数字段、寄存器或存储器中，它们支持多种操作数的寻址方式找到数据并参加运算。

3.2.1　立即寻址方式

立即寻址方式用于操作数本身就在指令中存放着的情况，如

```
MOV  AL,20H
```

数据 20H 可以立即送入 AL 寄存器。

3.2.2　直接寻址方式

直接寻址方式指的是指令直接给出了操作数的偏移地址，按地址从存储器把数据取出，如：

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

其中，方括号表示存储器地址内容，该指令用于访问存储器并取出数据，这里默认的段是数据段，即 DS * 10H＋2000H 得到数据的物理地址。

又如：

```
IN  AL,20H                          ;从地址端口 20H 读数据送 AL
```

3.2.3　寄存器寻址方式

CPU 的寄存器中存放着需要的数据，不需要访问存储器，具有执行速度快的特点。如：

```
MOV   AX,BX                         ;把 BX 寄存器内容送 AX 寄存器
PUSH  AX                            ;把寄存器 AX 内容压入堆栈
```

3.2.4　寄存器间接寻址方式

CPU 的寄存器中存放着需要数据的地址，而操作数本身位于存储器中，这里的寄存器称为间址寄存器，起到数据指针的作用，如：

```
MOV  AX,[SI]                        ;以 SI 的内容为地址，访问存储器取数据
```

如果(SI)＝1000H，数据段寄存器(DS)＝2000H，则操作数物理地址＝1000H×10H＋2000H＝12000H，即从存储器 12000H 单元开始取出数据字送 AX，并且高地址单元 12001H 的内容送 AH，低地址单元 12000H 的内容送 AL。

又如：

```
OUT  DX,AL                          ;把 AL 中的数据送 DX 指定的端口
```

3.2.5 寄存器相对寻址方式

操作数地址是寄存器的值与一个整数的和。操作数格式是变量名/符号名[基址寄存器/变址寄存器]或[变量名/符号名+基址寄存器/变址寄存器]。如：

```
MOV  AL,TABLE[SI]
```

其中，TABLE 是定义的数据表，表示数据表的首地址，SI 是表中数据的指针，二者相加作为数据的偏移地址，再加上(DS) * 10H，可以得到数据在存储器中的地址。

又如：

```
MOV  AL,ES:TABLE[SI]                ;数据地址是(ES) * 10H+TABLE 首地址+(SI)
```

3.2.6 基址加变址寻址方式

操作数的地址是由基址寄存器和变址寄存器提供的，二者的和形成操作数的偏移地址。如：

```
MOV  AX,[BX][SI]                    ;以(BX)+(SI)的值为数据偏移地址，访问存储器
```

3.2.7 相对基址加变址寻址方式

操作数偏移地址是指定寄存器内容与相对偏移量的和。如：

```
MOV  AX,TABLE[BX][SI]               ;数据偏移地址是 TABLE 首地址+(BX)+(SI)
```

3.3 80×86 的指令系统

8086/88 的指令系统包括数据传送、算术运算、逻辑运算、程序控制、处理器控制、串操作六种类型的指令，共 115 个助记符，91 种操作，其功能能够满足基本编程的需要。

指令格式：

```
操作符 目的操作数，源操作数
```

其中，操作符是指令完成的功能，目的操作数和源操作数可以是寄存器的名称、操作数地址或具体数据，而且根据具体指令和寻址方法，目的操作数和源操作数不一定完整，有的操作数隐藏了。

3.3.1 数据传送指令

1. 基本传送指令 MOV

基本传送指令用于完成源操作数到目的操作数的传送，如：

```
MOV  CX,3000H                       ;立即数 3000H 送 CX 寄存器
MOV  [2000H],AL                     ;AL 寄存器内容送地址为[2000H]的存储单元
```

```
MOV  DI,CX                          ;CX 内容送 DI
⋮
```

但立即数不能直接送 DS、ES、SS，中间须通过 AX；存储器地址间不能传送；需要源操作数和目的操作数的数据类型一致。

2. 堆栈指令 PUSH 和 POP

堆栈指令用于完成堆栈操作，微型计算机使用软堆栈，即在存储器空间开辟堆栈，形成堆栈段，段起始地址为 SS，堆栈顶指针 SP，SP 对栈顶操作，使用 PUSH 时，入栈，SP 减小，用 POP 指令，出栈，SP 增加。如：

```
PUSH  AL                            ;把 AL 压入堆栈
POP   BL                            ;把栈顶数据弹出，送 BL
```

3. 查表指令 XLAT

查表指令用于完成对数据段定义的数据表的查找，数据表首地址在 BX 中，AL 用于存放表中数据的顺序号，指令把对应的数据送 AL。如：

```
MOV  BX,TABLE                       ;数据表 TABLE 的首地址送 BX
MOV  AL,6                           ;查找第 6 个数据
XLAT                                ;把查到的数据送 AL
```

4. 交换指令 XCHG

交换指令用于在存储单元和寄存器或寄存器之间进行数据交换，如：

```
XCHG  AX,BX                         ;把 AX 和 BX 的数据对换
```

5. I/O 端口指令 IN 和 OUT

I/O 端口指令用于从端口输入/输出数据，端口地址在 00～FFH，可采用直接寻址，在 0100～FFFFH 必须使用间接寻址。如：

```
IN   AL,60H                         ;从端口 60H 读入数据到 AL
OUT  DX,AL                          ;从端口 DX 输出数据
```

6. 地址传送指令 LEA、LDS、LES

LEA、LDS、LES 用于完成偏移地址、数据段地址、附加段地址的传送，如：

```
LEA  BX,TABLE                       ;把数据段定义的 TABLE 的偏移地址(TABLE 首地址)送 BX
LDS  AX, TABLE                      ;把数据段定义的 TABLE 的段地址送 AX
```

3.3.2　算术运算指令

算术运算指令包括二进制和十进制的四则运算指令，用于完成无符号数和有符号数(补码)运算，影响标志寄存器。

1. 加法指令

加法运算指令有 ADD、ADC、INC 以及 DAA、AAA，如：

```
MOV  AL,37H
MOV  BL,02H
ADD  AL,03H                    ;(AL)+03H→AL,结果 AL=3AH
ADD  AL,BL                     ;(AL)+(BL)→AL,结果 AL=3AH+02H=3CH
```

ADC 用于完成带进位的加,即加上 CF,用于 2 个字节数据的运算。

INC 实现加 1 功能。

DAA、AAA 用于实现十进制加法的压缩 BCD 码和非压缩 BCD 码调整,如:

```
MOV  AL,35H                    ;压缩 BCD 码
ADD  AL,37H                    ;结果 AL=6CH
DAA                            ;加 6 调整后,AL=72H
MOV  AL,05H                    ;非压缩 BCD 码
ADD  AL,07H                    ;结果 AL=0CH
AAA                            ;加 6 调整后,AL=02H,CF=1,结果为 12H
```

2. 减法指令

减法运算指令有 SUB、SBB、DEC、NEG、CMP 以及 DAS、AAS,如:

```
SUB  AX,CX                     ;AX-CX→AX
```

SBB 实现带借位的减法,即需要减去 CF。

DEC 实现减 1 操作。

NEG 是求补指令,实现 0-操作数。

CMP 是比较指令,实现目的操作数—源操作数,但不存放结果,只改变 CF,用于比较判断目的操作数与源操作数的大小,结果可用于不同分支。

DAS、AAS 用于实现十进制减法的压缩 BCD 码和非压缩 BCD 码调整。

3. 乘法指令

乘法运算指令有 MUL、IMUL,用于完成无符号乘法和有符号乘法操作,以及十进制乘法的调整指令 AAM,如:

```
MUL  CL                        ;(AL)*(CL)→AX
MUL  BX                        ;(AX)*(BX)→DX:AX
```

指令执行影响标志位 CF 和 OF。

4. 除法指令

除法运算指令有 DIV、IDIV,用于完成无符号除法和有符号除法操作,以及十进制除法的调整指令 AAD,如:

```
MOV  AX,1000
MOV  BL,100
DIV  BL                        ;(AX)÷(BL)→AL(商)…AH(余数)
```

5. 符号扩展指令

乘除法运算时,需要按要求把被乘数和被除数扩展成相应的位数,以获得所需操作数

长度，对于无符号数，进行0扩展即可，对于有符号数，必须使用CBW、CWD扩展转换对符号位进行扩展，如：

```
MOV  AL,NUMB1          ;被除数NUMB1,8位
CBW                    ;扩展被除数NUMB1为16位
DIV  NUMB2             ;除以除数NUMB2
MOV  ANSQ,AL           ;商AL存入ANSQ
MOV  ANSR,AH           ;余数AH存入ANSR
```

3.3.3　逻辑运算和移位指令

逻辑运算指令包括NOT、AND、OR和XOR指令，用于完成求反、与运算、或运算、异或运算，逻辑运算按位操作；移位指令包括左移(L)和右移(R)、算术移位(SA)和逻辑移位(SH)、循环移位(RO)和带进位的循环移位(RC)等。

1. 逻辑运算指令

逻辑运算指令用于实现按位操作，执行影响标志位SF、ZF、PF，而CF、OF清0，如：

```
NOT  AL                ;AL各个位取反
AND  AL,0FH            ;AL的高4位清0,低4位不变
XOR  CX,CX             ;CX=0
```

2. 移位指令

移位指令用于根据左移和右移、算术和逻辑移位、循环移位带进位与否情况组合成SHR、SHL、SAR、SAL、ROR、ROL、RCR、RCL，如：

```
SHL  AX,1              ;把AX逻辑左移1位,相当于*2
MOV  BX,AX             ;存入BX
SHL  AX,1              ;把AX再次逻辑左移1位,相当于再*2
SHL  AX,1              ;再次左移1位,至此AX相当于*8
ADD  AX,BX             ;与*2后的AX相加,完成AX*10的操作
```

3.3.4　控制转移指令

程序全局是顺序执行的，但在实际应用时，需要根据指令执行情况进行不同的处理，需要转移到不同功能的程序段执行，程序控制指令可以实现程序的分支和循环。这类指令包括无条件转移指令、条件转移指令、循环指令、调用和返回指令以及中断指令等。

1. 无条件转移指令JMP

JMP用于实现无条件的转移到指定地址并从该地址开始执行程序，如：

```
      JMP  NEXT
      MOV  AL,30H
      …
NEXT: MOV  AL,20H
```

2. 条件转移指令

与 JMP 不同，条件转移指令是根据状态标志、比较结果、CX 计数情况等条件发生转移的。常用指令有 JZ/JE、JNZ/JNE、JB/JNB、JA/JNA、JGE/JNG 等，如：

```
    ⋮
    CMP  AX,BX                    ;比较 AX 和 BX,相等则转 SS1,否则顺序执行
    JZ   SS1
SS0: …
SS1: …
```

又如：

```
     MOV  AL,NUB                  ;NUB 存储了的一个 8 位符号数送 AL
     CMP  AL,0                    ;与 0 比较
     JGE  BIG                     ;大于等于 0 转 BIG,否则顺序执行
     MOV  AL,0FFH                 ;小于 0,则赋值-1
     JMP  EXIT                    ;跳转到 ZERO
BIG: JE   ZERO                    ;=0 转 ZERO
     MOV  AL,01H                  ;大于 0,则赋值 1
     JMP  EXIT
ZERO:MOV  AL,0                    ;等于 0,则赋值 0
EXIT:MOV  RES,AL                  ;把结果存入 RES
```

3. 循环指令 LOOP

LOOP 指令借助默认的 CX 循环计数器，可以在－128～127 的范围内实现程序有规律的循环，类似的指令还有 LOOPZ/LOOPE、LOOPNZ/LOOPNE，如：

```
    MOV   CX,100                  ;循环次数
    MOV   AX,0                    ;初始值
L1: INC   BX                      ;每次循环+1
    ADD   AX,BX                   ;实现每次增 1 的累加
    LOOP  L1                      ;CX 没有减到 0 则循环
```

4. 子程序调用指令 CALL 和返回指令 RET

特定功能的程序段组织成一段程序，形成子程序，子程序可被多次调用，子程序的调用与返回使用的是指令 CALL 和 RET。子程序与 JMP 不同，它不是把程序转移到别处，而是在现行执行程序中插入子程序的运行，即在调用子程序时，保存程序断点，把 CS:IP 入栈保存，保存子程序的段地址与偏移地址赋值 CS:IP，当子程序执行完毕时，RET 返回指令负责原 CS:IP 的恢复，按原 CS:IP 执行原来的程序。

5. 中断指令 INT 和中断返回指令 IRET

系统的中断调用类型号对应特定的功能程序，这种程序称为中断服务程序，用户使用 INT n 可以利用系统提供的资源，用户可以自行编写 INT n 对应的程序，只需把自行编写的程序的入口地址(即 CS:IP)按中断类型号加载到对应的中断向量表。自行编写的中断服务程序的末尾使用 IRET 来完成中断断点和标志寄存器 FLAGS 的恢复。

3.3.5　处理器控制指令

这类指令可以对处理器本身进行控制和设定，包括标志设置指令和以处理器为操作对象的控制指令。

1. 标志设置指令

这类指令具有对标志寄存器的 CF、DF、IF 标志置位/复位的功能，以及间接控制处理器执行的功能。如：

```
CLC    ;0→CF
STC    ;1→CF
CMC    ;CF→CF
CLD    ;0→DF
STD    ;1→DF
CLI    ;0→IF
STI    ;1→IF
```

2. 其他处理器控制指令

这类指令用于对处理器本身进行控制，包括 NOP、HLT、WAIT、LOCK、ESC，如：

```
NOP          ;空操作，在需要时间延时和调试程序时使用
HLT          ;停机，一旦有中断发生、复位操作或 DMA 操作时处理器脱离停机状态
WAIT         ;处理器处于等待状态，并不断测试TEST引脚，TEST=0 则脱离等待状态
```

LOCK　XXX 指令名称；总线封锁指令，可放在任何一条指令前，作用是使处理器放弃总线控制权，直到 LOCK 后面的指令执行为止

ESC 存储器寻址方式；为协处理器提供操作码，数据总线把存储单元内容送出并开始协处理器指令的执行；当遇到协处理器助记指令码，汇编程序把它转换为 ESC 指令的机器码，表示此处为协处理器的操作码。

3.3.6　字符串操作指令

字符串操作指令可以完成对若干数据的操作，其功能相当于一段程序，这类指令包括 MOVS、CMPS、SCAS、LODS、STOS，它们可对字节和字进行操作，其功能分别是字符串传送、比较、扫描、取、存。默认地址为 DS:SI→ES/DS:DI，地址变化方向由 DF 决定；并且结合重复前缀 REP/REPE/REPNE 来实现重复传送，重复次数必须由 CX 指定。如：

```
   LEA  SI,SRC          ;字符串源的首地址→SI
   LEA  DI,DST          ;字符串目的的首地址→DI
   MOV  BX,100          ;传送次数
   CLD                  ;方向 DF=0,地址增量
L: MOVSB                ;传送一个字节
   DEC  BX              ;传送 1 次,次数减 1
   JNZ  L               ;没完成,循环传送
```