汇编语言与接口技术

第3章 80X86指令系统和寻址方式

主编：王让定 朱莹

宁波大学信息学院

本章主要内容录

01

8086/8088指令系统概述

02

8086/8088的寻址方式和指令系统

03

80386的寻址方式和指令系统

04

80486/Pentium微处理器新增指令

8086/8088指令系统概述

PART 01

数据类型

- 无符号二进制数
- 带符号二进制数
- 浮点数
- BCD码
- 串数据
- ASCII码数据
- 指针类数据
- 位串：一串连续的二进制数
  - 字节串：一串连续的字节。
    - 字串：一串连续的字。
    - 双字串：一串连续的双字。

8086指令格式

指令的操作码

- 指令的操作码（简称OP）采用二进制代码表示本指令所执行的操作，在大多数指令的操作码中，常用某位指示某些具体操作信息。下图所示8086操作码，含有3个特征位，分别为W位、D位和S位。

操作码格式

单字节指令（隐含操作数）

操作码

单字节指令（寄存器模式）

操作码	REG

寄存器到寄存器

操作码	11	REG	R/M

不带位移量的寄存器和内存之间的传送

操作码	MOD	REG	R/M

带位移量的寄存器和内存之间的传送（设位移量为 16 位）

操作码	MOD	REG	R/M	位移量低位	位移量高位

立即数送寄存器（设立即数为 16 位）

操作码	11	操作码	R/M	数据低位	数据高位

立即数送内存（设带 16 位位移量）

操作码	MOD	操作码	R/M	位移量低位	位移量高位	数据低位	数据高位

REG——寄存器

MOD——模式

R/M——寄存器或内存

8086指令格式

几点说明：

① 一条指令可以包含一个操作数，也可以包含一个以上的操作数；一个操作数的指令称为单操作数指令，单操作数指令中的操作数可能由指令本身提供，也可能由指令隐含地指出。

② 若位移量或立即数为16位，那么在指令代码中，将低位字节放在前面，高位字节放在后面。

③ 8086指令系统中大多数指令的操作码只占用第一个字节，但有几条指令是特殊的，其指令中的第一个字节不但包含操作码成分，而且还隐含地指出了寄存器名，从而整个指令只占一个字节，成为单字节指令。这些指令字节数最少，执行速度最快，用得也最频繁。

指令的执行时间

- 指令执行时间取决于时钟周期长短和执行指令所需要的时钟周期数。
- 执行一条指令所需的总时间为基本执行时间、计算有效地址的时间和为了读取操作数和存放操作结果需访问内存的时间之和。

8086的寻址方式和指令系统

PART 02

寻址方式

- 数据寻址方式：是指获取指令所需的操作数或操作数地址的方式
- 程序寻址方式：是指程序中出现转移和调用时的程序定位方式

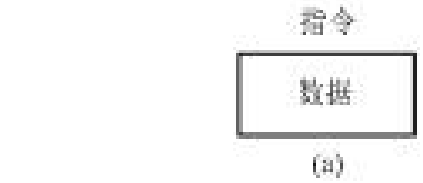
1．数据寻址方式

8086指令中所需的操作数来自以下几个方面。

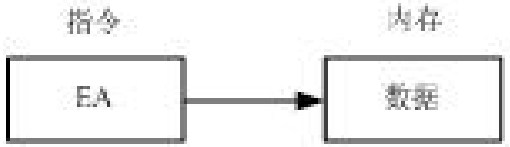
（1）操作数包含在指令中。在取指令的同时，操作数也随之得到，这种操作数被称为立即数。

（2）操作数包含在CPU的某个内部寄存器中，这种操作数被称为寄存器操作数。

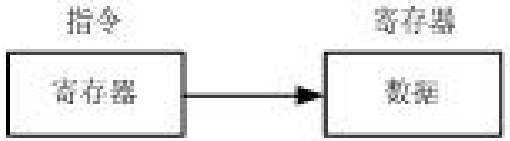
（3）操作数包含在存储器中，这种操作数被称为存储器操作数。



(a)

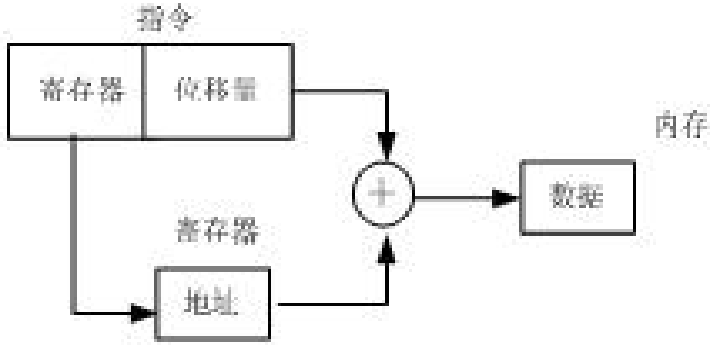


(b)

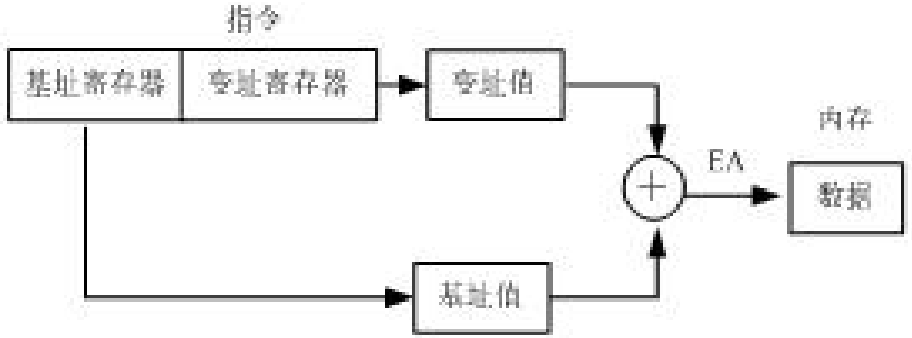


(c)

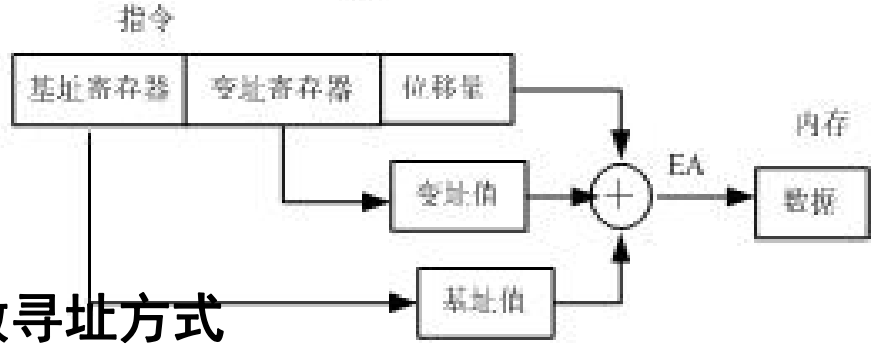
(d)



(e)



(f)



(g)

8086的操作数寻址方式

(a)立即数寻址　（b）直接寻址　（c）寄存器寻址　（d）寄存器间接寻址

（e）寄存器相对寻址　（f）基址加变址寻址　（g））相对的基址加变址寻址

1、立即寻址方式

操作数作为指令的一部分，在直接指令中给出。

例

```
MOV   AX, 0A7FH      ; AX←0A7FH，执行后，AH=0AH，AL=7FH
MOV   AL, 5H         ; AL←5H
```

立即寻址方式主要是用于给寄存器或存储单元赋初值

2、寄存器寻址

寄存器寻址是指操作数包含在CPU内部的寄存器中

例

```
MOV   AH, AL
```

3、存储器寻址

（1）直接寻址方式

有效地址只包含偏移量一种成分。

例 MOV　AX,[2000H]

直接寻址一般多用于存取某个存储器单元中的操作数，比如从一个存储单元取操作数，或者将一个操作数存入某个存储器单元。

（2）寄存器间接寻址方式

有效地址只包含在基址寄存器或变址寄存器中。这些寄存器可以是BX、BP、SI、DI之一

1）以BX、SI、DI进行寄存器间接寻址（BX、SI、DI作为地址指针）的方式，隐含的段寄存器为数据段寄存器DS

例

```
MOV　AX, [BX]          ; 物理地址=DS×16+BX
MOV　BX, [SI]          ; 物理地址=DS×16+SI
MOV　[DI], DX          ; 物理地址=DS×16+DI
```

2）以BP进行寄存器间接寻址（BP作为地址指针）的方式

隐含的段寄存器为堆栈段寄存器SS，操作数存放在堆栈段区域，将堆栈段寄存器SS的内容左移4位，再加上基址寄存器BP的内容，即为操作数的物理地址。

例

```
MOV   [BP], BX                    ; 物理地址=SS×16+BP
```

无论用BX、SI、DI或者BP作为间接寄存器，都允许段超越，即可以使用上面所提到的约定以外的其它段寄存器。

例

```
MOV   AX, ES:[BX]                 ; 物理地址=ES×16+BX
MOV   DS:[BP], DX                 ; 物理地址=DS×16+BP
```

（3）相对寄存器寻址

是在寄存器间接寻址基础上加上位移量，位移量可以是正的也可以是负的。

例 MOV　AX, 3003H[SI]

假设DS=3000H，SI=2000H，指令中的3003H即为位移量DISP。指令操作的存储器物理地址

=3000H×10H+EA=30000H+2000H+3003H=35003H.

（4）基址、变址寻址

基址变址寻址方式是用基址寄存器BX,BP和变址寄存器SI、DI变址寄存器进行的间接寻址。

例 MOV　AX, [SI+BX]

假设DS=3000H，SI=2000H，BX=1000H。指令操作的存储器物理地址

=3000H×10H+EA=30000H+1000H+3003H=34003H.

（5）相对的基址、变址寻址

将基址寻址方式和变址寻址方式联合起来的寻址方式称为基址、变址寻址方式。这种寻址方式，操作数在存储单元中，其物理地址由段寄存器内容左移4位加上一个基址寄存器和一个变址寄存器，再加上16位或8位位移量。

例

MOV　AX, MASK[BX][SI]

假设 MASK=64H，BX=A500H，SI=2200H，DS=6000H

物理地址=DS×10H+EA=6000H×10H+A500H+2200H+64H=6C764H

2. 程序寻址方式

（1）直接寻址方式

段内直接寻址方式也称为相对寻址方式，是指把指令本身提供的位移量加到指令指针寄存器中，形成有效目标地址的寻址方式。

例

```
JMP    1000H                ；转移地址在指令中给出
CALL   1000H                ；调用地址在指令中给出
```

（2）段内间接寻址方式

程序转移的地址存放在寄存器或存储单元中，这个寄存器或存储单元内容可用以上所述寄存器寻址或存储器寻址方式取得，所得到的转移有效地址用来更新IP的内容。由于此寻址方式仅修改IP的内容，所以这种寻址方式只能在段内进行程序转移。

例

```
JMP    BX                              ；转移地址由BX给出
CALL   AX                              ；调用地址由AX给出
JMP   WORD PTR [BP+TABLE]；转移地址由BP+TABLE所指的存储单元给出
```

（3）段间直接寻址方式

这种寻址方式是在指令中直接给出16位的段基值和16位的偏移地址，用来更新当前的CS和IP的内容。

例

```
JMP    2500H：3600H   ；转移的段地址和偏移地址在指令中给出
CALL   2600H：3800H ；调用程序段地址和偏移地址在指令中给出
```

（4）段间间接寻址方式

这种寻址方式是由指令中给出的存储器数据寻址方式，包括存放转移地址的偏移量和段地址。其低位字地址单元存放的是偏移地址，高位字地址单元中存放的是转移段基值。这样既更新了IP内容又更新了CS的内容，故称为段间间接寻址。

例

```
JMP   WORD PTR[BX]        ；转移到当前代码位置内
；有效地址存放在BX寻址的单元中
```

3. I/O地址空间

I/O端口寻址是对输入输出设备的端口地址寻址，可分为两种形式，即直接端口寻址和间接端口寻址。

（1）直接端口寻址

由指令直接给出端口地址，端口地址范围为0～255。

例

```
IN   AL，32H                    ；32H为8位端口地址
```

（2）间接端口寻址

由DX寄存器指出端口地址，这种方式给出的端口地址范围为0～65535。

例

```
IN   AL，DX                     ；DX寄存器的内容为端口寻址
```

4. 段寄存器的确定

访存类型	默认段寄存器	段超越前缀的可用性
代码	CS	不可用
PUSH、POP类代码	SS	不可用
串操作的目标地址	ES	不可用
以BP、SP间址的指令	SS	可用CS、DS，ES
其它	DS	可用CS、SS、ES

8086指令系统

- 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
- 串操作指令
- 程序控制指令
- 处理器控制指令。

1. 数据传送类指令

- 数据传送类指令用于实现CPU的内部寄存器之间
- CPU内部寄存器和存储器之间
- CPU累加器AX或AL和I/O端口之间的数据传送

此类指令除了SAHF和POPF指令外均不影响标志寄存器的内容。

在数据传送指令中，源操作数和目的操作数的数据长度必须一致。

（1）MOV指令

指令格式：MOV　dest，src　　；dest←src

指令功能：　MOV指令用于将一个操作数从存储器传送到寄存器，或从寄存器传送到存储器，或从寄存器传送到寄存器，也可以将一个立即数存入寄存器或存储单元，但不能用于存储器与存储器之间，以及段寄存器之间的数据传送，MOV指令传送关系如图3.4所示。

MOV指令有6种数据传输格式。

1）CPU的通用寄存器之间的数据传输

例 3.17

```
MOV   AL, BL              ; BL寄存器的8位数送到AL寄存器
MOV   SI, BX              ; BX寄存器的16位数送到SI寄存器
```

2）立即数（常数）到存储单元的数据传输

例 3.18

```
MOV   MEM_BYTE, 20H ; 将立即数20H送到MEM_BYTE存储单元
MOV   DS:[0005H], 4500H
                  ; 立即数4500H送到DS:0005H的存储单元中
```

3）立即数到通用寄存器的数据传输

例 3.19

```
MOV   AL, 20H                ；将立即数20H送到AL寄存器
MOV   SP, 2000H              ；将立即数2000H送入SP寄存器
```

4）通用寄存器和存储单元之间的数据传输

例 3.20

```
MOV   AL, DS:[1000H]
            ；将地址DS:1000H存储单元的内容送到AL
MOV   ES:[0002H], BX
            ；将BX中的16位数据传输到地址ES:0002H
            ；所指的两个相邻存储单元中
```

5)段寄存器和存储单元之间的数据传输

例 3.21

```
MOV   ES, [BX]                    ; BX寄存器所指内容
    传送到ES
MOV   [1000H], CS                 ; 将CS内容传送到地
    址为1000H的存储器单元
```

6)通用寄存器和段寄存器之间的数据传输。

例 3.22

```
MOV   AX, ES                      ; 将ES段地址传送到DS
    段
MOV   DS, AX
```

使用MOV指令时，必须注意以下几点：

- MOV指令可以传8位数据，也可以传16位数据，这决定于寄存器是8位还是16位，或立即数是8位还是16位。
- MOV指令中的目的操作数和源操作数不能都是存储器操作数，即不允许用MOV实现两个存储单元间的数据传输。
- 不能用CS和IP作为目的操作数，即这两个寄存器的内容不能随意改变。
- 不允许在段寄存器之间传输数据，例如MOV DS,ES 是错误的。
- 不允许用立即数作为目的操作数。
- 不能向段寄存器传送立即数。如果需要对段寄存器赋值，可以通过CPU的通用寄存器AX来完成。

（2）交换指令 XCHG

指令格式：XCHG　　dest, src　　；dest←→src

指令功能：XCHG指令用于交换两个操作数。这条指令实际上起到了三条MOV指令的作用，指令中的两个操作数可以是两个寄存器操作数或一个寄存器与一个存储器操作数。

交换指令可实现通用寄存器之间、通用寄存器与存储单元之间的数据(字节或字)交换。

例 3.24

```
XCHG  AL, BL ；（AL）与（BL）寄存器的字节型数据进行交换
XCHG  BX, CX ；（BX）与（CX）寄存器的字型数据进行交换
XCHG  DS: [2200H], DL；（DL）与地址DS：2200H的内容进行交换，（DH）与地址DS：2201H的内容进行交换
```

使用交换指令时应注意以下两点：

- 两个操作数不能同时为存储器操作数。
- 任一个操作数都不能使用段寄存器，也不能使用立即数。

（3）I/O指令

1）输入指令

指令格式：IN　累加器，端口

指令功能：把一个字节/字由输入端口传送到AL/AX中。

例 3.25

```
IN    AL, 21H                    ；将端口21H的8位数读到AL
MOV   DX, 201H
IN    AX, DX                     ；将端口201H和202H的16位数读到AX中
```

2）输出指令

指令格式：OUT　端口，累加器

指令功能：把AX中的16位数或AL中的8位数输出到指定端口。

例 3.26

```
OUT   22H, AL                    ；将AL中的数据传到端口22H
MOV   DX, 511H
OUT   DX, AX                     ；将AX的数据输到511H和512H端口
```

（4）取有效地址指令LEA

格式：LEA　r16，mem

要求：r16为一个16位通用寄存器，　mem为存储单元。

功能：LEA指令的作用是将有效地址（在这里指地址偏移值）

　　送通用寄存器，而不是将存储单元的内容送通用寄存器。

例 3.27

```
LEA   BX，DS:1000H ；将地址1000H送到BX，即（BX）=1000H
LEA   BX，BUFFER    ；将存储变量BUFFER的变量地址传送到BX
LEA   BX，[2728]                    ；将2728偏移地址送BX
LEA   SP，[BP][DI]    ；将[BP+DI]寻址方式的偏移地址送SP
```

在很多情况下，LEA指令可以用相应的MOV指令代替。

例 3.28

```
LEA   BX, VARWORD
MOV   BX, OFFSET VARWORD
```

这两条指令的执行结果是完全一样的。区别在于后者用伪指令OFFSET，由编译程序在编译时赋值；而前者在执行时赋值。

注意：

两个操作数不能同时为存储器操作数。

任一个操作数都不能使用段寄存器，也不能使用立即数。

（5）装入地址指令

指令格式：LDS dest，src ；将内存中连续4个字节内容送到DS和指定的通用寄存器

指令功能：LDS指令把src操作数所指的内存中连续4个字节单元内容的低16位数据存入dest指定的通用寄存器中，高16位存入DS中；LES指令把src操作数所指的内存中连续4个字节单元内容的低16位数据存入dest指定的通用寄存器中，高16位存入ES中。

注意：dest必须是通用寄存器之一，src必须是内存操作数

例 3.29

LDS　DI，［2130］；将2130H和2131H单元的内容送DI

；将2132H和2133H单元的内容送DS

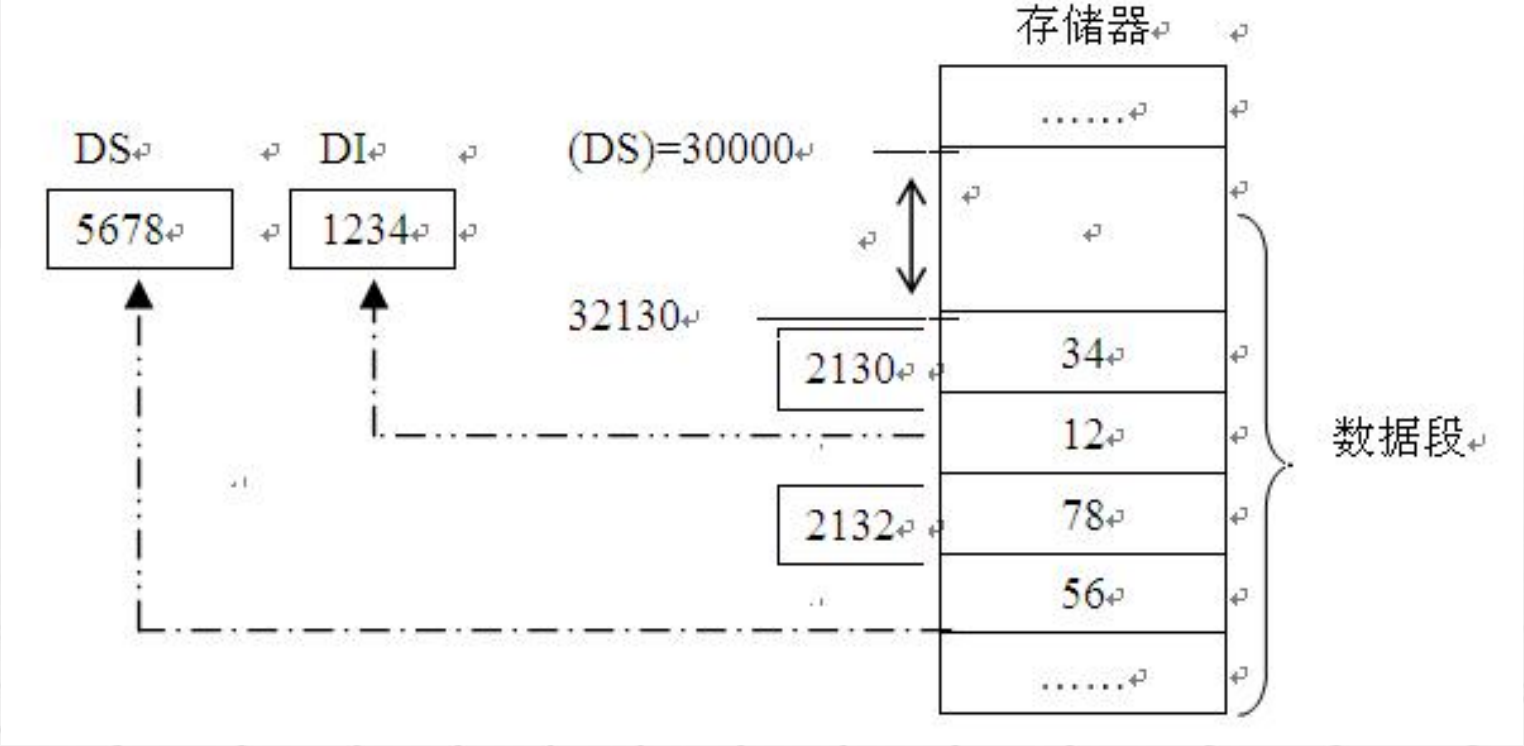


图3.5　LDS指令示意图

（6）标志传送指令

指令格式：LAHF

SAHF

指令功能：标志传送指令有取标志指令LAHF和存标志指令SAHF，LAHF指令用于将标志寄存器的低8位送入AH，SAHF指令用于将AH的内容送入标志寄存器的低8位。

（7）表转换指令XLAT

指令格式：XLAT label

XLAT ；AL← DS:（（BX）+（AL））

指令功能：XLAT指令以转换表中的一个字节来代换AL寄存器中的内容，可用于码的转换。其中DS:BX指向转换表的首址，转换前AL内容为序号，转换后AL内容为对应的码。

例如，将ASCII码转换为EBCDIC码，也可用于代码加密。

例 3.31

将AL中的一个数字字符的ASCII码，经变换加密后从42H端口输出。表3.2给出了密码转换表。

表 3.2 密码转换表

TAB_POINT O	'5'
1	'7'
2	'9'
3	'1'
4	'3'
5	'6'
6	'8'
7	'0'
8	'2'
9	'4'

有关程序段如下：

```
SUB    AL, ‘0’          ；将AL中ASCII字符转换成表
    XMIT_TABLE中的序号
LDS    BX, TAB_POINT     ；将表头地址指针送DS：BX
XLAT   XMIT_TABLE        ；查转换表，将对应的加密码放入AL
OUT    42H, AL           ；从42H端口输出加密后的ASCII码
TAB_POINT  DD XMIT_TABLE
XMIT_TABLE DB  ‘5791368024’
```

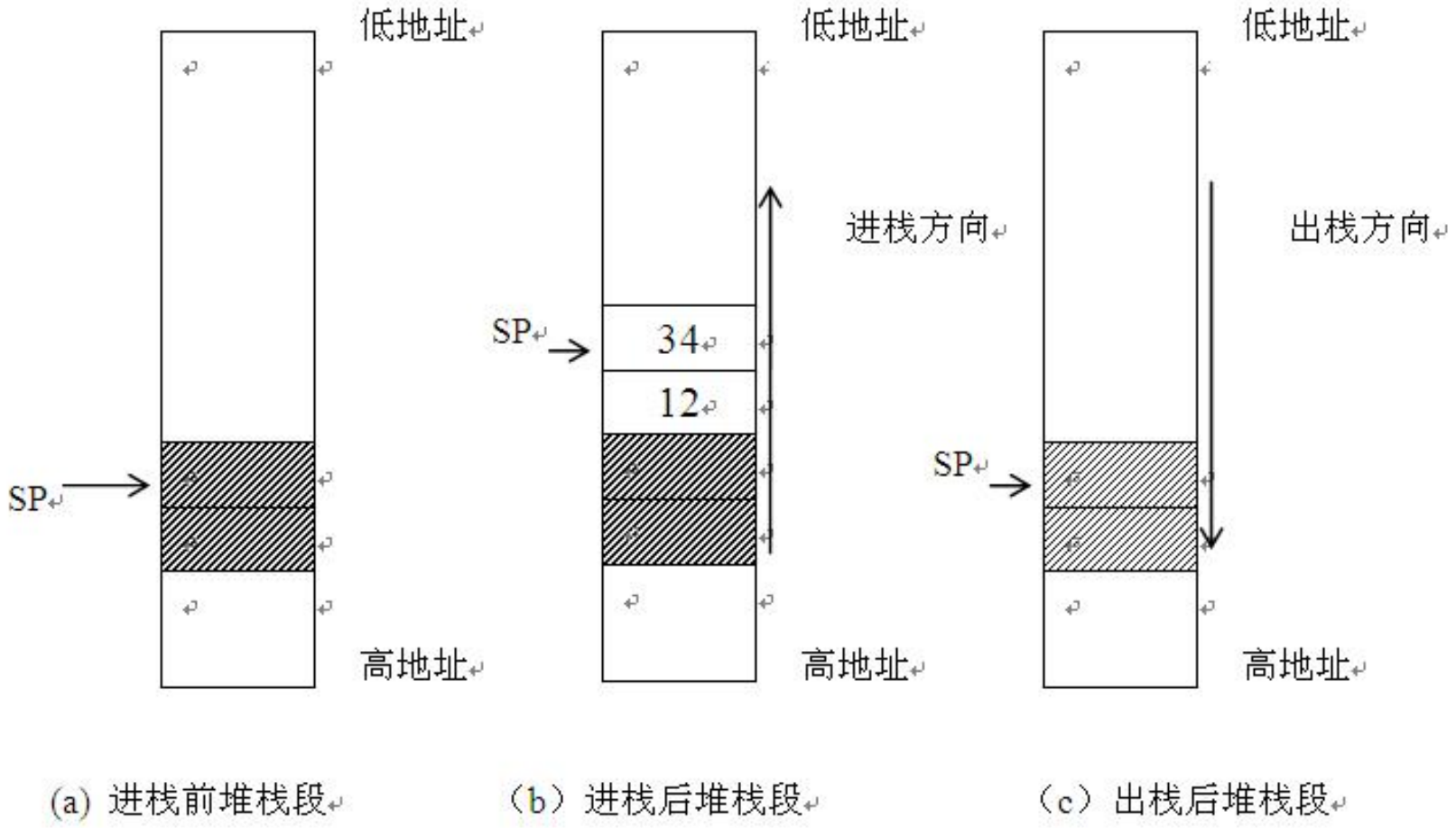
（8）堆栈操作指令

堆栈是向下生成的，并且栈顶是满的。也就是说，压栈操作是先将SP减2，再将数据压入SS：SP指向的单元；弹出操作则先将SS：SP指向的数据弹出，再将SP加2。

指令格式：　PUSH　src

POP　　dest

指令功能：PUSH指令用于压入存储器操作数，寄存器操作数或立即数。注意，src和dest为16位的寄存器或存储单元，图3.6 给出了进栈和出栈操作示意图。



(a) 进栈前堆栈段　　（b）进栈后堆栈段　　（c）出栈后堆栈段

图 3.6　进栈和出栈操作示意图

例 3.32

```
PUSH   AX                              ；将AX的内容压入堆栈
PUSH   [2000]   ；将DS段逻辑地址为2000H的内容压入堆栈
```

POP指令与PUSH指令相反，将栈顶的数据弹出到通用寄存器或存储器中。

例 3.33

```
POP   AX                    ；将栈顶的内容，弹出到AX寄存器
POP   M_ADD                 ；将栈顶的内容，弹出到M_ADD指
      定的存储器中
```

注意：

- PUSH和POP指令只允许按字访问堆栈，即两类指令的操作数必须是16位寄存器或存储单元的操作数。
- CS不能作为目的操作数

(9) 符号扩展指令

1）字节扩展指令

指令格式：CBW

指令功能：将AL中的单字节数的符号扩展到AH中。若AX<80H，则0→AH；若AL≥80H，则FFH→AH。

例 3.34

```
MOV   AL, 6FH                    ; AL=01101111B
CBW                               ; AH=00000000B, AL
    内容不变
MOV   AL, 0AFH                   ; AL=10101111B
CBW                               ; AH=11111111B, AL
    内容不变
```

2）字扩展双字指令

指令格式：CWD

指令功能：将AX中的单字节数的符号扩展到DX中。若

AX<8000H，则0→DX；若AL≥8000H，则FFFFH→DX。

例 3.35

```
MOV   AX, 4F0AH                    ; AX=4F0AH
CWD                                ; DX=0000H, AX内容不变
MOV   AX, 0EF0AH                   ; AX=0EF0AH;
CWD                                ; DX=0FFFFH, AX内容不变
```

2. 算术运算指令

（1）加法类指令

1）不带进位的加法指令

指令格式：ADD　dest，src　　；dest←（dest）+（src）

指令功能：ADD指令用来对源操作数和目的操作数字节或字相加，结果放在目的操作数中。

说明：

ADD指令不允许两个存储器单元内容相加，两个操作数不能同时为存储器操作数。

ADD指令也不允许在两个段寄存器之间相加。

对标志位有影响，主要是CF、ZF、OF、SF标志位，其他还有AF，PF位。

2）带进位标志的加法指令

指令格式：ADC dest，src ；dest←（dest）+（src）+（CF）

指令功能：ADC指令和ADD指令的功能类似，区别在于ADC在完成两个字或2个字节数相加的同时，还要考虑进位标志CF的值，若进位位CF为1，则将结果加1。

注意：src源操作数和 dest目的操作数不能同时为存储单元。段寄存器不能进行算术运算。

例 3.39

有两个4字节数分别放在FIRST和SECOND开始的存储区，低字节在低地址处，编写一程序将两数相加，并将结果存入FIRST开始的存储区。

```
MOV   AX, FIRST                ; 第一个数的低16位→AX
ADD   AX, SECOND               ; 两数的低16位相加→AX
MOV   FIRST, AX                ; 低16位相加结果存入FIRST及FIRST+1单元
MOV   AX, FIRST+2              ; 第一个数的高16位→AX
ADC   AX, SECOND+2             ; 两数的高16位连同低16位进位相加→AX
MOV   FIRST+2, AX              ; 高16位相加的结果存入FIRST+2及FIRST+3单元
```

3）加1指令

指令格式：INC　reg/mem　　；reg/mem←（reg）/（mem）+1

要求：reg为8位或16位通用寄存器，mem为8位或16位存储单元

指令功能：将源操作数加1，再送回该操作数。这条指令一般用于循环程序的指针修改，INC指令只有一个操作数。

说明：

操作数可以是寄存器或存储单元，但不能为立即数。

INC指令影响标志位AF、OF、PF、SF和ZF，但不影响CF位。

INC指令将操作数视为无符号数。

（2） 减法类指令

1）不带借位的减法指令

指令格式：SUB　dest，src　　；dest←（dest）-（src）

指令功能：将目的操作数减去源操作数，结果送到目的操作数，并根据结果设置标志。

说明：

SUB指令不允许两个存储器单元内容相减，两个操作数不能同时为存储器操作数。

SUB指令也不允许在两个段寄存器之间相减。

对标志位有影响，主要是CF、ZF、OF、SF，以及AF，PF。

2）带借位的减法指令

指令格式：SBB　dest，src

；dest←（dest）-（src）-（CF）

SBB指令与SUB指令的功能相似，区别是SBB在完成字节或字相减的同时，还要减去借位CF。

说明：

src源操作数和 dest目的操作数不能同时为存储单元。

段寄存器不能进行算术运算。

3）减1指令

指令格式：DEC　reg/mem　　；reg/mem←（reg）/（mem）-1

减1指令只有一个操作数。操作数可以为寄存器或者存储单元，不能为立即数。该指令实现将操作数中的内容减1，又叫减量指令。

例 3.43

```
DEC　CX                                    ；（CX）←(CX) - 1
DEC　DS:[100H+2] ；将数据段DS偏移地址100H+2所指单元内容
；减1，结果送回该单元
```

注意：DEC指令和INC指令一样，执行后对CF不产生影响。

- Overflow problem in signed number operations

Example 6-4. example of overflow:

```
DATA1    DB          +96
DATA2    DB          +70
         ...         ...
         MOV         AL, DATA1 ;AL=0110  0000 (AL=60H)
         MOV         BL, DATA2 ;BL=0100  0110 (BL=46H)
         ADD         AL, BL    ;AL=1010  0110 (AL=A6H=-90 INVALID!)
         +  96  (60H)  0110  0000
         +  70  (46H)  0100  0100
         +166  (A6H) 1010  0110   (OF=1, SF=1, CF=0)
         According to the CPU, this is 90, which is wrong.
```

In the example above, the addition result in AL is A6H, which is the 2's complement of -5AH(-90 in decimal).

- CPU sets up OF bit when:
  - Byte size operand (see examples 6-5 6-6):
    - There is a carry from D6 to D7 but no carry out of D7(CF=0)
    - There is a carry from D7 out (CF=1) but no carry from D6 to D7
  - Word size operand (see examples 6-8 6-9:
    - There is a carry from D14 to D15 but no carry out of D15 (CF=0)
    - There is a carry from D15 out (CF=1) but no carry from D14 to D15

Example 6-10. rewrite example 6-4 to provide for handling the overflow problem.

S	AH	AL	
0	000 0000	0110 0000	+96 after sign extension
0	000 0000	0100 0110	+70 after sign extension
0	000 0000	1010 0110	+166

```
DATA1     DB      +96
DATA2     DB      +70
RESULT    DW      ?
          …       …
          SUB     AH, AH      ;clear AH
          MOV     AL, DATA1   ;AL=0110  0000 (AL=60H)
          MOV     BL, DATA2   ;BL=0100  0110 (BL=46H)
          ADD     AL, BL      ;AL=1010  0110 (AL=A6H=-90 INVALID!)
          JNO     OVER        ;If OF=0 then goto over
          MOV     AL, DATA2   ;Otherwise get operand 2 to
          CBW                 ;Sign extend it
          MOV     BX, AX      ;Save back operand 1 to
          MOV     AL, DATA1   ;sign extend it
          CBW                 ;Sign extend it
          ADD     AX, BX      ;Add them and
OVER:     MOV     RESULT, AX;Save it
```

(3) 取补指令

指令格式：NEG　reg/mem ；reg/mem←0-（reg）/（mem）

指令功能：将操作数取补后送回源操作数，即将操作数连同符号逐位取反，然后在末位加1，适用于操作数在机器内用补码表示的场合。

NEG指令的操作数可以是8位或16位通用寄存器和存储器操作数，不能为立即数。

例 3.44

```
MOV  AL, 00000001B                    ;
   AL=00000001B
NEG  AL                               ; 将AL中的数取
   补, AL=11111111B
NEG  WORD PTR [SI+1]                  ; 将SI+1、SI+2
   单元中的内容取补
```

(4) 比较指令

指令格式：CMP　dest，src　；（dest）-（src）

指令功能：将目的操作数与源操作数相减，不送回结果，只根据结果置标志位。

例 3.45

```
CMP   AX,BX                    ；将AX-BX后，置标志位
CMP   AL,20H                   ；将AX-20H后，置标志位
```

说明：本指令通过比较（相减）结果置标志位，表示两个操作数的关系。比较有以下几种情况，以CMP A,B为例说明。

判断两个操作数是否相等：可根据ZF标志位判断，若ZF=1，说明A=B；若ZF=0，说明A≠B。

判断两个操作数的大小：

①判断两个无符号操作数的大小：可根据CF标志位来判断，若CF=1，说明A＜B；若CF=0，说明A≥B。

②判断两个带符号操作数的大小：可根据SF及OF标志来判断，若SF⊕OF=1，即SF与OF不同时，说明A<B；若SF⊕OF=0，即SF与OF相同时，则A≥B。

两个异符号数比较：

如果OF＝0，仍可用SF标志判断大小。

如果OF＝1，说明结果的符号位发生错误，

所以　SF＝0，则AX＜BX

SF＝1，　则AX≥BX

综上所述：　当OF=0，SF=0，　则AX≥BX

SF＝1，则AX＜BX

当OF＝1，SF＝0，则AX＜BX

SF＝1，　则AX≥BX

用逻辑表达式表示为：若OF⊕SF＝0，则AX≥BX

若OF⊕SF＝1，　则AX＜BX

(5) 乘法指令

1）无符号数乘法指令

指令格式：MUL　reg/mem　　；dest←隐含被乘数AL/AX

乘以乘数reg/mem

MUL　dest，src　　；dest←被乘数（dest）乘以乘数（src）

要求：dest 目的操作数可为8位、16位通用寄存器或存储单元，

src源操作数可为8位、16位通用寄存器、存储单元或立即数，但dest，src不能同为存储器。

指令功能：完成两个不带符号的8位或16位二进制数的乘法计算。乘积存放在AH，AL或DX，AX中。

例 3.48

```
MOV   AL, NUMBER1
MUL   NUMBER2                    ; 将AL的内容乘以NUMBER2,
      乘积存放在AX中
```

例 3.49

```
MOV   AX, NUMBER1
MUL   NUMBER2                    ; 将AX的内容乘以
      NUMBER2, 乘积的高16位
; 放在DX中, 低16位放在AX中
```

2）带符号乘法指令

指令格式：IMUL　reg/mem

IMUL　dest，src

要求：dest 目的操作数可为8位、16位通用寄存器或存储单元，src源操作数可为8位、16位通用寄存器、存储单元或立即数，但dest，src不能同为存储器。

指令功能：完成两个带符号的8位或16位二进制数乘法计算。乘积存放在AH，AL或DX，AX中。

- 源操作数为通用寄存器或存储器操作数(不能是立即数)，
- 目的操作数缺省存放在AL、AX或EAX中，
- 乘积存AX、DX:AX或EDX:EAX中（IMUL指令，积采用补码形式表示）。
- 字节乘：AL×SRC→AX
- 字乘：AX×SRC→DX:AX
- 双字乘：EAX×SRC→EDX:EAX
- MUL、IMUL指令执行后，
- 如果CF = OF = 0，表示乘积高位无有效数据，
- 如果CF = OF = 1，表示乘积高位含有效数据，
- 对其它标志位无定义。
- 在使用IMUL指令时，需在IMUL指令后加一条判断溢出的指令，溢出时转错误处理执行程序。

Multiplication	Operand1	Operand2	Result
Byte x Byte	AL	Reg/Mem	AX[1]
Word x Word	AX	Reg/Mem	DX AX[2]
Word x Byte	AL = byte CBW	Reg/Mem	DX AX[2]

1. CF=1 OF=1 if AH has part of the result; CF=0 OF=0 if AH holds the extended sign
2. CF=1 OF=1 if DX has part of the result; CF=0 OF=0 if DX holds the extended sign

（6）除法指令

1）无符号数除法指令

格式：DIV　src　　；隐含被除数AX/DX或AX除以除数（src）

　　DIV　dest，src　　；被除数（dest）除以除数（src）

要求：dest目的操作数可为8位、16位通用寄存器或存储单元，src源操作数可为8位、16位通用寄存器、存储单元或立即数，但dest，src不能同为存储器。

指令功能：当用DIV指令进行无符号数的字或字节相除时，所得的商和余数均为无符号数，分别放在AL和AH中，若进行无符号数的双字或字相除时，所得的商和余数也是无符号数，分别放在AX和DX中。

2）带符号除法指令

格式：IDIV　src

IDIV　dest，src

要求：dest 目的操作数可为8位、16位通用寄存器或存储单元，src源操作数可为8位、16位通用寄存器、存储单元或立即数，但dest，src不能同为存储器。

指令功能：IDIV指令用于两个带符号数相除，其功能和对操作数长度的要求与DIV指令类似，本指令执行时，将被除数、除数都作为带符号数，其相除操作和DIV相同。

字节除法：AX/SRC　　商→AL，余数→AH

字除法：　DX:AX/SRC　　商→AX，余数→DX

双字除法：EDX:EAX/SRC　商→EAX，余数→EDX

Division	Numerator	Denominator	Quotient	Rem.
Byte / Byte	AL=byte CBW	Reg/Mem	AL	AH
Word / Word	AX	Reg/Mem	AX	DX
Word / Byte	AX=word	Reg/Mem	AL[1]	AH
Doubleworld/word	DXAX=doubleword	Rem/Mem	AX[2]	DX

1. Divide error interrupt if -127 > AL > +127ign
2. Divide error interrupt if -32768> AL > +32768

例 3.52

```
MOV   DX, NUMBER_MSB
MOV   AX, NUMBER_LSB
IDIV  DIVSR ；被除数高16位在DX中，低16位在AX中，除数
；为16位，其结果商在AX中，余数在DX中
```

（7）BCD调整指令

BCD调整指令分为两类，即组合的和未组合的BCD调整指令。

1）组合的BCD调整指令

指令格式：　DAA

　　DAS

第一条指令对在AL中的和（由两个组合的BCD码相加后的结果）进行调整，产生一个组合的BCD码；

第二条指令对在AL中的差（由两个组合的BCD码相减后的结果）进行调整，产生一个组合的BCD码。

① DAA指令的调整方法：

如AL中的低四位在A – F之间，或AF为1，则AL ← (AL)+6，且AF位置1。

如AL中的高四位在A – F之间，或CF为1，则AL ← (AL)+60H，且CF位置1。

② DAS指令的调整方法：

如AL中的低四位在A – F之间，或AF为1，则AL ← (AL)-6，且AF位置1。

如AL中的高四位在A – F之间，或CF为1，则AL ← (AL)-60H，且CF位置1。

DAA 和DAS指令影响标志AF，CF，PF，SF和ZF，但不影响标志OF。

例 3.53

```
MOV   AL, 34H
ADD   AL, 47H          ; AL=7BH,   AF=0,   CF=0
DAA                    ; AL=81H,   AF=1,   CF=0
ADC   AL, 87H          ; AL=08H,   AF=0,   CF=1
DAA                    ; AL=68H,   AF=0,   CF=1
ADC   AL, 79H          ; AL=E2H,   AF=1,   CF=0
DAA                    ; AL=48H,   AF=1,   CF=0
```

2）未组合的BCD调整指令

指令格式：AAA

AAS、AAM、AAD

指令功能：AAA 指令用于对ADD指令运算后AL中的内容进行调整。AAS 指令用于对SUB指令运算后AL中的内容进行调整。AAM 指令用于对MUL指令运算后AX中的乘积进行BCD调整。除法的情况却不同，AAD指令用于调整DIV运算前AH、AL中除数，以便除法所得的商是有效的未组合的BCD数。AAA和AAS两条指令会影响AF与CF，而对OF、PF、SF和ZF没有意义。

例 3.54

AAA这条指令对在AL中的和（由两个未组合的BCD码相加后的结果）进行调整，产生一个未组合的BCD码。调整方法如下：

① 如AL中的低4位在0-9之间，且AF为0，则转③。

② 如AL中的低4位在A- F之间，或AF为1，则AL ← (AL)+6，AH ← (AH)+1，AF位置1。

③ 清除AL中的高4位。

④ AF位的值送CF位。

下面是该指令的一个程序片段：

```
  MOV   AX, 7
  ADD   AL, 6      ; AL=0DH,    AH=00H,    AF=0, CF=0
  AAA               ; AL=03H,    AH=01H,    AF=1, CF=1
  ADC   AL, 6      ; AL=09H,    AH=01H,    AF=0, CF=0
  AAA               ; AL=09H,    AH=01H,   AF=0, CF=0
ADD   AL, 39H       ; AL=42H,    AH=01H,   AF=1, CF=0
AAA                 ; AL=08H,    AH=02H,   AF=1, CF=1
```

3. 逻辑指令

逻辑运算指令包括AND、OR、XOR、NOT和TEST指令。

指令格式：

```
AND  dest, sr; dest← (dest) ∧ (src)
OR   dest, src          ; dest← (dest) ∨ (src)
XOR  dest, src          ; dest← (dest) ⊕ (src)
NOT  reg/mem   ; reg/mem←OFFFFH- (reg) / (mem)
TEST  dest, src       ;  (dest) ∧ (src)
```

- 指令要求：dest为8位或16位通用寄存器或存储单元，src为8位或16位通用寄存器、存储单元或立即数。reg为8位或16位通用寄存器，mem为8位或16位存储单元。
- 指令功能：AND、OR和XOR指令执行按位逻辑“与”、“或”和“异或”。这些指令使用操作数的下列组合：两个寄存器操作数、一个通用寄存器操作数和一个存储器操作数、一个立即数与一个通用寄存器操作数或一个存储器操作数。TEST指令将两个操作数按位“与”，只置标志不存结果。NOT指令是一元运算指令，用于将指定操作数的各位变反，其操作数可为寄存器或存储器。
- 注意：src源操作数和dest目的操作数不能同时为存储单元，段寄存器不能进行逻辑运算。

例 3.55

```
NOT   BX
AND   AX, 0F0F0H
OR    AX, CX
```

例 3.56

```
TEST   AL, 00010000B
JZ     ABC        ; AL中第4位为0则转ABC，否则顺序执行
 ┆
ABC:
 ┆
```

4. 移位与循环移位指令

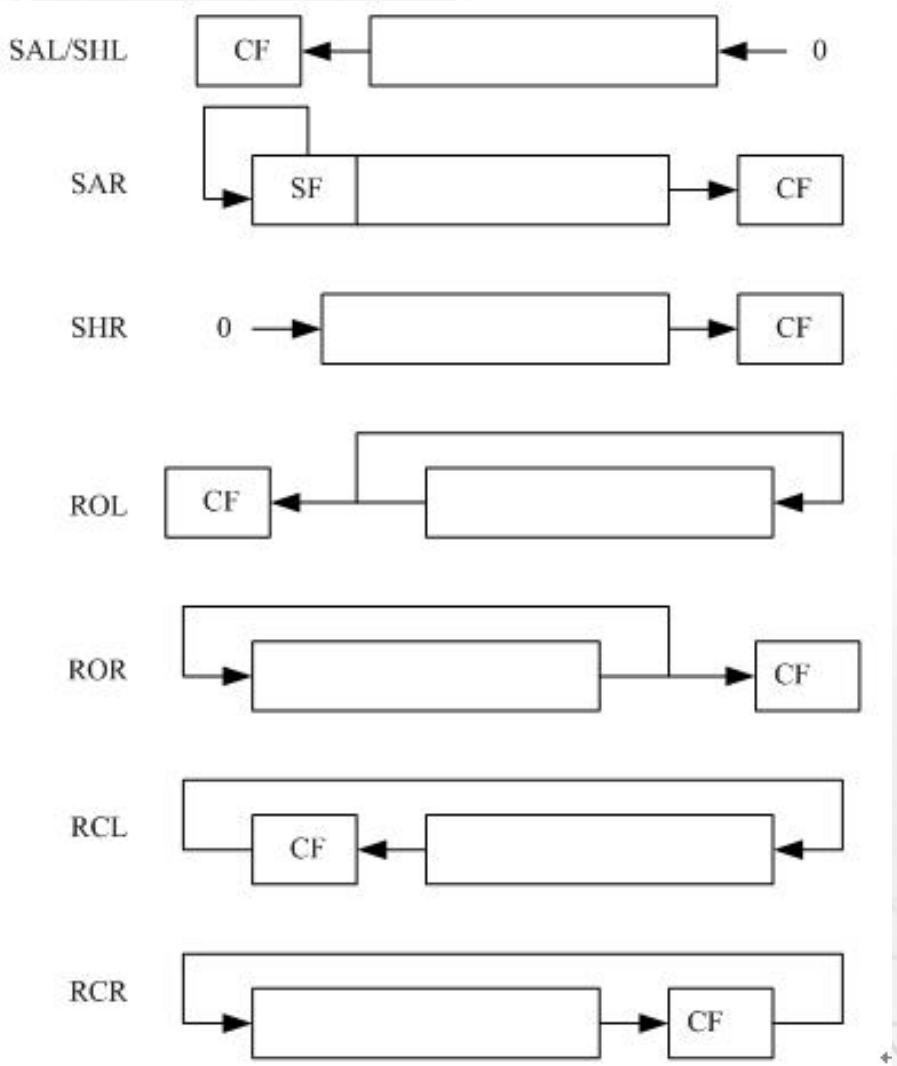


图 3.7 移位指令示意图

（1）算术移位指令

指令格式： SAL reg/mem，1/CL

SAR reg/mem，1/CL

要求： reg为8位或16位通用寄存器，mem为8位或16位存储单元。

指令功能：SAL为算术左移指令，算术左移1次或CL指定的次数。SAR为算术右移指令，算术右移1次或CL指定的次数。

例 3.57

```
MOV  AL, 8CH                          ; AL=8CH
SHL  AL, 1 ; AL=18H, CF=1, PF=1, ZF=0, SF=0, OF=1
MOV  CL, 6                            ; CL=6
SHL  AL, CL  ; AL=0, CF=0, PF=1, ZF=1, SF=0, OF=0
```

（2）逻辑移位指令

指令格式： SHL reg/mem，1/CL

SHR reg/mem，1/CL

要求：reg为8位或16位通用寄存器，mem为8位或16位存储单元。

指令功能：SHL为逻辑左移指令，逻辑左移1次或CL指定的次数，与SAL相同。SHR为逻辑右移指令，逻辑右移1次或CL指定的次数。

例 3.58

下面的程序片段实现把寄存器AL中的内容（设为无符号数）乘10，结果存放在AX中。

```
XOR   AH, AH                    ;  (AH) =0
SHL   AX, 1                     ; 2×AX
MOV   BX, AX                    ; 暂存
SHL   AX, 1                     ; 4×AX
SHL   AX, 1                     ; 8×AX
ADD   AX, BX                    ; 8×AX+2×AX
```

（3）循环移位指令

指令格式： ROL reg/mem, 1/CL

ROR reg/mem, 1/CL

RCL reg/mem, 1/CL

RCR reg/mem, 1/CL

要求：reg为8位或16位通用寄存器，mem为8位或16位存储单元。

指令功能：ROL为循环左移指令，循环左移1次或CL指定的次数。

ROR为循环右移指令，循环右移1次或CL指定的次数。RCL为带进位循环左移指令，带进位循环左移1次或CL指定的次数。

RCR为带进位循环右移指令，带进位循环右移1次或CL指定的

5．处理器指令

（1）标志位处理指令 ■

在前面介绍的有关指令中，已经涉及到了对标志位的影响问题，由此可以利用标志位的状态对所执行的程序进行控制。通过标志位处理指令，可直接对指定的标志位进行设置或消除操作。（1）清进位位指令CLC ，即CF←0

(2) 进位标志位置“1”指令STC, 即CF←1。

(3) 进位位求反指令CMC， 即CF←CF的反。

(4) 方向标志位清“0”指令CLD，即DF←0。

(5) 方向标志位置“1” 指令STD，即DF←1。 ■

(6) 中断标志位清“0”指令CLI， 即IF←0。

(7) 中断标志位置“1”指令STI，即IF←1。

（2）同步与控制指令

1）停机指令HLT。

HLT指令使CPU进入暂停状态。暂停状态期间， CPU可响应外部中断。如RESET复位信号，INTR、NMI的中断请求。中断返回后，CPU将退出暂停状态。通常在程序中使用HLT指令等待中断请求。

2）等待指令WAIT。

WAIT指令可以使CPU进入等待状态，等待状态期间（TEST变为无效）CPU可被中断。当TEST为有效或允许外部中断请求时， CPU会停止执行WAIT。

3）换码或协处理器指令ESC。 ■

ESC指令可向协处理器提供一条可执行的指令及相应的操作数。 当执行ESC指令时，协处理器监视系统总线， 并可取得操作码。由于80287协处理器无寻址能力，当取得操作需要存储器访问时，80286将指定存储单元的内容送到数据总线上，否则不需要80286做任何事情。

4）封锁指令LOCK。

LOCK是一个可用在有关指令前面的前缀，使用了这个指令前缀的指令，可在指令执行期间封锁局部总线，以保证在多处理器及多任务下的数据安全。

（3）空操作指令NOP

NOP指令不执行任何操作，它作为单字节指令占用一个字节的单元。一般用它来占用字节单元，便于插入指令进行程序调试，还可利用它对定时程序中的时间进行调整，占用3个时钟周期。

80386的寻址方式和指令系统

PART 03

80386的寻找方式

- 80386支持先前微处理器所支持的各种存储器寻址方式，并且还支持32位偏移和存储器寻址方式。80386允许内存地址的偏移可以由三部分内容相加构成，一是一个32位基址寄存器，二是一个可乘上比例因子1、2、4或8的32位变址寄存器，三是一个8位到32的常数偏移量。

存储器寻址方式中的有关名词概念定义如下：

- 基址：任何通用寄存器都可作为基址寄存器，其内容为基址
- 位移量：在指令操作码后面的32位、16位或8位的数。
- 变址：除了ESP寄存器外，任何通用寄存器都可以作为变址寄存器，其内容即为变址值。

- 比例因子：变址寄存器的值可以乘以一个比例因子，根据操作数的长度可为1字节、2字节、4字节或8字节，比例因子相应地可为1、2、4或8。

由上面4个分量计算有效地址的方法为

- EA=基址+变址×比例因子+位移量

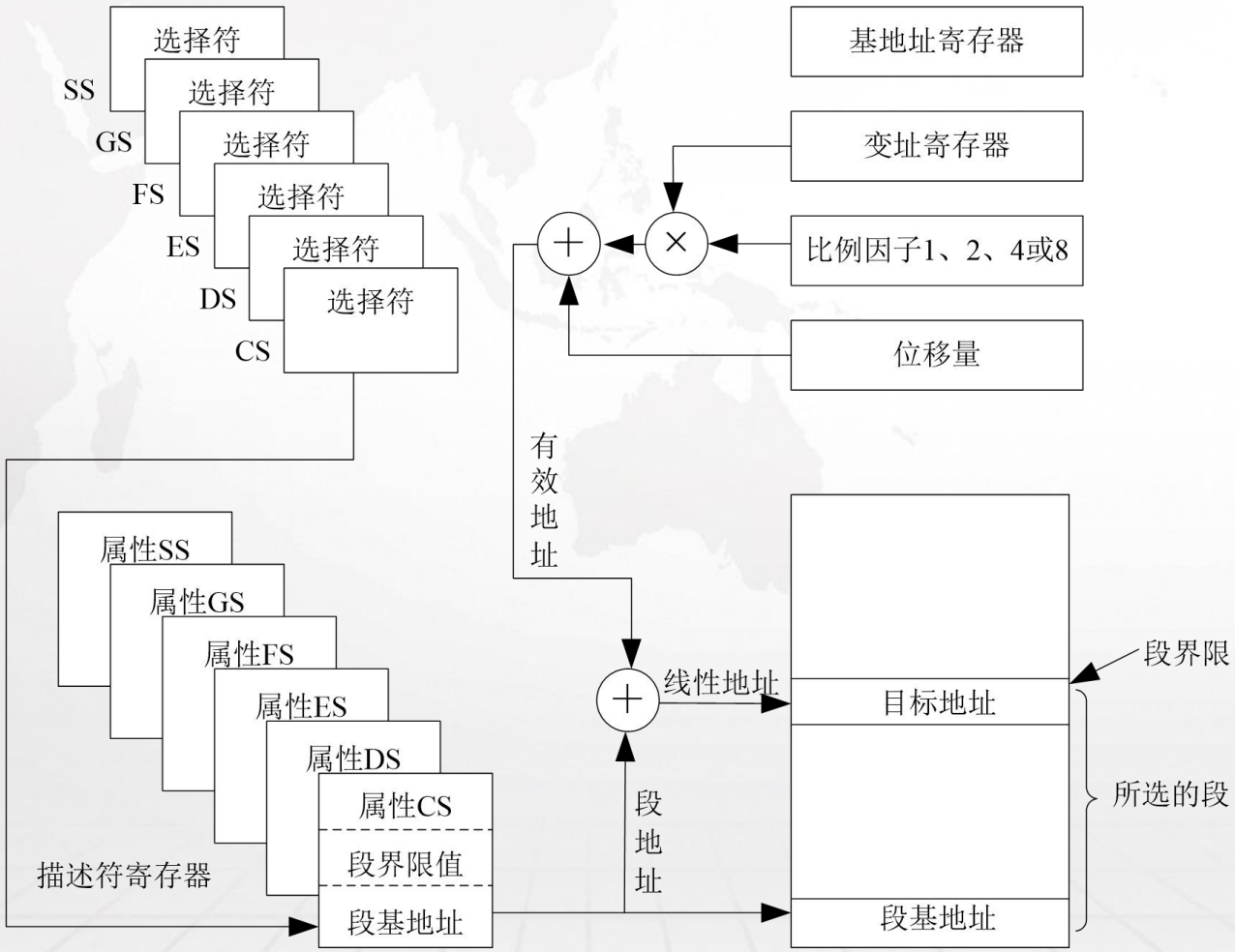

选择符
SS
选择符
GS
选择符
FS
选择符
ES
选择符
DS
选择符
CS
基地址寄存器
变址寄存器
比例因子1、2、4或8
位移量
有效地址
属性SS
属性GS
属性FS
属性ES
属性DS
属性CS
段界限值
段基地址
描述符寄存器
线性地址
段地址
段界限
目标地址
所选的段
段基地址

图3.9 寻址计算图解

80386指令系统

1．传送类指令

有5组传送指令，即通用传送指令、累加器传送指令、标志传送指令、地址传送指令和数据类型转换指令。

- 在8086指令系统中，通用传送指令的操作数之一为寄存器或存储器，另一个为寄存器或立即数。用操作符MOV时，两个操作数的位数必须相同。
- 在80386指令系统中，可以实现双字数据传送，另外新增了符号扩展传送指令和零扩展传送指令MOVSX和MOVZX
- 格式：MOVSX dest，src
- MOVZX dest，src
- 要求：dest为16位或32位通用寄存器，src为8位或16位通用寄存器或存储单元。
- 功能：MOVSX指令将8位符号数带符号扩展成16位或32位、或者将16位符号数带符号扩展成32位数再传送。MOVZX指令将8位数通过在高位加0扩展成16位或32位、或者将16位数通过在高位加0扩展成32位数再传送。

例

```
MOV     BL, 92H
MOVSX   AX, BL              ; AX←FF92H
MOVSX   ESI, BL
 ; ESI←FFFFFF92H, 将BL内容带符号扩展为32 位送入ESI
MOVZX   EDI, AX
; EDI←0000FF92H, 将AX内容在高位加0扩展成32位送EDI
```

在80386指令系统中，XCHG指令除了可进行字节交换、字交换外，还可以实现双字交换。

例

```
XCHG  EAX, EDI          ；寄存器和寄存器进行双字交换
XCHG  ESI, MEM_DWORD    ；寄存器和存储器进行双字交换
```

在80386指令系统中，PUSH指令的操作数除了可以是寄存器或是存储器外，还可以是立即数（POP指令无此功能）。

例

```
PUSH  0807H                ；将立即数0807H压入堆栈
```

- 另外，80386中，用PUSHA一条指令就可以将全部的16位寄存器压入堆栈。而用PUSHAD一条指令则可将全部的32位寄存器压入堆栈。POPA、POPAD则进行相反的弹出操作。
- PUSHAD压入堆栈的次序为：EAX、ECX、EDX、EBX、ESP、EBP、ESI、EDI。其中，进栈的ESP内容是在EAX压入堆栈前的值。

（2）累加器传送指令

累加器传送指令包括IN、OUT以及XLAT、XLATB指令。这里，IN和OUT指令的使用方法和8086一样，端口地址可直接在指令中给出，也可由DX寄存器间接给出。

XLAT和XLATB指令功能相同，都称为换码指令，前者是8086中延续下来的指令，以BX为基址，而后者以80386寄存器32位的EBX为基址。

（3）标志传送指令

标志传送指令除了8086中已有的LAHF、SAHF、PUSHF、POPF外，80386还增加了两条指令，即

```
PUSHFD
；将标志寄存器的内容作为一个双字压入堆栈
POPFD
 ；从堆栈顶弹出双字到标志寄存器
```

在80386的CPU中，PUSHF和POPF指令是将标志寄存器的低16位压入堆栈或弹出堆栈。

（4）地址传送指令

80386的地址传送指令实现6字节地址指针的传送。地址指针来自存储单元，目的地址为两个寄存器，其中一个是段寄存器，一个为双字通用寄存器。LEA指令允许32位操作数；LDS指令允许从数据段中取出32位的偏移量送给目标寄存器，取16位的段基址送DS；LES指令从当前数据段中取48位的地址指针送ES和目标寄存器。80386还新增了3条地址传送指令，即

```
LFS   r16/r32，mem   ；从当前数据段中取48位的地址指针送FS和r16/r32寄存器
LGS   r16/r32，mem  ；从当前数据段中取48位的地址指针送GS和r16/r32寄存器
LSS   r16/r32，mem   ；从当前数据段中取48位的地址指针送SS和r16/r32寄存器
```

例

```
TABLE   DD   TABLE1 ；定义变量TABLE1的类型为双字（4字节）
DATA    DF   DATA1 ；定义变量DATA的类型为长字（6字节）
LFS     SI, TABLE  ；2字节偏移地址送SI，2字节送段地址FS
LGS     ESI, DATA  ；4字节偏移地址送ESI，2字节送段地址GS
```

上例中的DF是数据定义的伪操作命令，它的作用是将变量的类型定义为6个字节（48位），DF伪操作是80386所特有的。

（5）数据类型转换指令

在80386指令系统中，数据类型转换指令除了和8086具有完全一样的CBW、CWD指令外，还增加了2条指令。即

CWDE　；将AX中的字进行高位扩展，成为EAX中的双字

CDQ　；将EAX中的双字进行高位扩展，得到EDX和EAX中的4字

CWDE和CDQ指令与CBW和CWD一样，对标志位没有影响。

2. 算术运算指令

80386指令系统的算术运算指令与8086并没有很大的区别，最大的改变只是对32位数据的支持。

① 除法运算指令DIV和IDIV用AX、DX+AX或者EDX+EAX存放16位、32位或者64位被除数，除数的长度为被除数的一半，可存放在寄存器或者存储器中。指令执行后，商放在源存放被除数的寄存器的低半部分，余数放在高半部分。

②乘法运算指令MUL、IMUL的操作数可为2个8位数、2个16位数或2个32位数。寄存器AL、AX或者EAX存放其中一个操作数并保存乘积的低半部分，另一个操作数为寄存器或存储器，也可为立即数。乘积的高半部分在8位×8位时存放在AH，在16位×16位时存放在EAX的高16位，且同时放在DX中，在32位×32位时放在EDX中。

【例】

```
IMUL   DX, BX, 500H
 ；将BX中内容乘以500H，结果送DX
IMUL   ECX, EDX, 1000H
 ；将EDX内容乘以1000H，结果送ECX
IMUL   EDX, MEM_DWORD, 30H
 ；将存储器中双字乘以30H，结果送EDX
```

3．逻辑指令

逻辑指令包括NOT、AND、OR、XOR、TEST、SHL、SHR、SAL、SAR、ROL、ROR、RCL、RCR、SHRD、SHLD。前面的13条指令是8086中就有的，但它们增加了对32位寄存器的支持，后面2条指令是80386特有的。

在80386中，增加了两条专用的双精度移位指令，即双精度左移位指令SHLD和双精度右移位指令SHRD，它们可以对64位的4字进行移位。

SHRD/SHLD r/m,, r, i8

SHRD/SHLD r/m,, r, CL

功能：将指令中的两个操作数连起来进行移位。其中，

第一操作数来自寄存器或存储器，第二操作数来自寄存器，在移位操作中，将第二操作数内容移入第一操作数，而第二操作数本身不变。进位位CF中的值为第一操作数移出的最后一位。

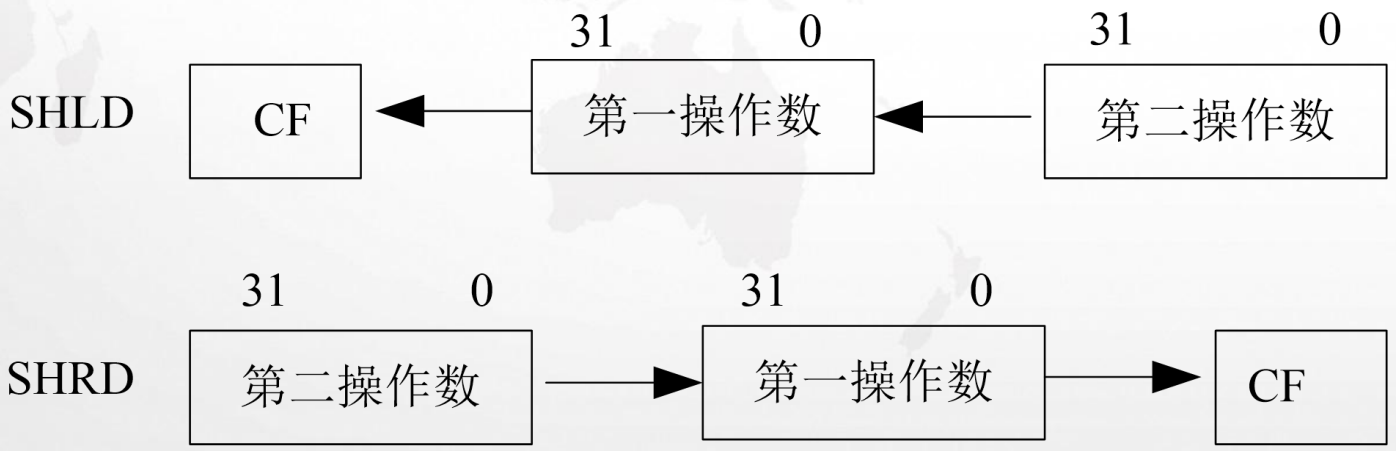


图　双精度移位示意图

例

```
SHLD   TABLE[EBX], EAX, 5
```

该指令表示将TABLE[EBX]指向的双字单元的内容左移5位、移空位由EAX的内容左移5位填入，但EAX的内容不改变。

4. 位串操作指令

- 串操作指令包括 MOVS、CMPS、SCAS、LODS、STOS、INS、OUTS。
- 前5个指令是8086中原有的，在80386指令系统增加了对32位寄存器的支持，如MOVS可以写成MOVSB、MOVSW、MOVSD。它们的源变址寄存器、目的变址寄存器及计数器是用ESI、EDI、ECX还是用SI、DI、CX由它所在的段决定。若是32位段，则使用前者，否则使用后者。

5. 转移、循环和调用指令

- 控制转移指令包括如JMP、JZ、JB、LOOP、CALL、RET、INT、INTO、IRET等，它们与在8086中的意义基本相同。
- 在80386中，条件转移指令的相对转移地址不受范围限制
- 80386指令系统增加了一条JECXZ指令。它的跳转条件是以ECX是否为0，而不是以CX是否为0作为标准。这个指令的跳转范围也仍然是－128 ~+127。

- 循环控制指令LOOP、LOOPZ/LOOPE、LOOPNZ/LOOPNE也归为条件转移指令一类，这一组指令的含义和用法与8086完全相同，转移范围也仅限于-128 ~ +127。
- 调用指令CALL和返回指令RET在用法和含义上类同于8086，只是在80386中，EIP寄存器为4个字节，所以，堆栈操作时，对应EIP的操作为4个字节。80386的返回指令和8086一样，RET后面也可带一个偶数，比如，RET 8，以便返回时丢弃栈顶下面一些用过的参数。

6. 条件设置指令

80386指令系统新增了条件设置指令，用于支持编译程序和高级语言生成的代码。这些指令根据当前的EFLAG中的标志，来设置对应的寄存器或存储器。

指令格式；SETcc　r/m

其中cc泛指所有的设置条件。若条件满足，则设置寄存器r或存储器单元m为指定的值。

例

```
MOV    AL, 0
CMP    EAX, EBX
SETZ   AL
```

表 80386指令系统新增条件设置指令的功能

指　　令	功　　能
SETZ或SETE	ZF=1则置r或m为1
SETS	SF=1则置r或m为1
SETO	OF=1则置r或m为1
SETP或SETPE	PF=1则置r或m为1
SETB或SETNAE或SETC	CF=1则置r或m为1
SETBE或SETNA	CF=1或ZF=1则置r或m为1
SETL或SETNGE	SF和OF不相等则置r或m为1
SETLE或SETNGE	SF和OF不相等或ZF=1则置r或m为1
SETNZ或SETNE	ZF=0则置r或m为1
SETNS	SF=0则置r或m为1
SETNO	OF=1则置r或m为1
SETNP或SETPO	PF=0则置r或m为1
SETNB或SETAE或SETNC	CF=0则置r或m为1
SETNBE或SETA	CF=0且ZF=0则置r或m为1
SETNL或SETGE	SF和OF相等则置r或m为1
SETNLE或SETG	SF和OF相等且ZF=0则置r或m为1

7. 中断指令

和中断有关的指令INT n、INTO及IRET，这些指令的含义和8086的一样。此外，80386还增加了IRETD指令，这条指令功能上和IRET类似，但执行时，从堆栈中先弹出4个字节装入EIP，再弹出2个字节装入CS。

8. 标志位指令

80386的标志指令和8086的完全一样，包含清除进位标志指令CLC、设置进位标志指令STC、进位标志求反指令CMC、清除方向标志指令CLD、设置方向标志指令STD以及中断允许标志清除指令CLI和中断允许标志设置指令STI。

9．位操作指令

80386设置了一系列进行位处理的指令，表3.5给出了80386指令系统新增位处理指令的功能。

指　令	功　能
BTS	将测试位的值送CF，并把指定的测试位置1
BTR	将测试位的值送CF，并把指定的测试位置0
BTC	将测试位的值送CF，并把指定的测试位取反
BT	测试指定的位，并将测试位的值送CF
BSF	从低到高扫描目标，如果全为0则ZF置1，否则ZF置0，并把为1位的序号放入目的寄存器
BSR	从高到低扫描目标，如果全为0则ZF置1，否则ZF置0，并把为1位的序号放入目的寄存器

例

```
MOV  CX, 4
BT   [BX], CX
; 检查由BX指向的存储单元中的数的第4位，且位
; 4放入进位标志中
BITSA DW 1234H, 5678H
BITSB DD 12345678H
---
BT   BITSA, 4                    ; （BITSA）=1234H,
   CF←1
MOV  CX, 22
BTC  BITSA, CX                   ; （BITSA+2）=5638H,
   CF←1
MOVZX  EAX, CX
BTS  BITSB, EAX                  ; （BITSB）
   =12745678H, CF←0
```

10. LOCK前缀指令

表 可以使用LOCK前缀的指令

ADD mem, reg	BT mem, reg	OR mem, imm	BTC mem, imm
ADC mem, reg	BTR mem, reg	SBB mem, imm	DEC mem
AND mem, reg	BTS mem, reg	SUB mem, imm	INC mem
OR mem, reg	BTC mem, reg	XOR mem, imm	NEG mem
SBB mem, reg	ADD mem, imm	BT mem, imm	NOT mem
SUB mem, reg	ADC mem, imm	BTR mem, imm	XCHG reg,mem
XOR mem, reg	AND mem, imm	BTS mem, imm	XCHG mem,reg

11．系统寄存器的装入与存储指令

系统寄存器包括控制寄存器CR、调试寄存器DR和测试寄存器TR。通常它们只由系统程序员使用，也就是说，只有具有最高特权级（0级）的程序才能使用这些寄存器存取指令。

(1)与控制、调试和测试寄存器有关的传输指令

系统控制、调试和测试寄存器的装入与存储可通过在通用寄存器与系统寄存器之间的传送指令进行。

例

```
MOV   TR6, EAX                    ; 装入测试寄存器TR6
MOV   EBX, CR3                    ; 保存控制器CR3的内容
```

(2)装入机器状态字指令

格式：LMSW　r16/m16

功能：LMSW指令将操作数装入CR0的0～15位中。

(3)保存机器状态字指令

格式：SMSW　r16/m16

功能：SMSW指令将CR0的低16位（MSW）存入操作数所指定的2字节通用寄存器或存储单元中。

(4)装入中断描述符表寄存器指令

格式：LIDT　m16&32

功能：LIDT指令将操作数所指向的16位段界限和32位段基地址6个字节单元的48位数装入到中断描述符表寄存器IDTR 中，以确定中断向量表的位置与大小。

(5)保存中断描述符表寄存器指令

格式：SIDT　m16&32

功能：SIDT指令将中断描述符表寄存器IDTR的48位数写入操作数指向的6个字节单元中。

(6)装入全局描述符表寄存器指令

格式：LGDT　m16&32

功能：LGDT指令将操作数所指向的16位段界限和32位段基地址的6个字节单元内容装入全局描述符表寄存器GDTR，以确定内存中全局描述符表的位置与大小。

(7)保存全局描述符表寄存器指令

格式：SGDT　m16&32

功能：SGDT指令将全局描述符表寄存器GDTR的48位数写入操作数指向的6个字节单元中。

(8)装入局部描述符表寄存器指令

格式：LLDT　r16/m16

功能：LLDT指令将操作数所表示的16位选择符装入局部描述符表寄存器LDTR中。此时局部描述符表的描述符由系统自动装入到CPU内部相应的位描述符寄存器中，以确定该任务的局部描述符表的位置及大小。

(9)保存局部描述符表寄存器指令

格式：SLDT　r16/m16

功能：SLDT指令将局部描述符表寄存器LDTR的16位选择符写入操作数所表示的地址单元中。

(10)装入任务状态段寄存器指令

格式：LTR　r16/m16

功能：LTR指令将操作数所表示的任务状态段选择符装入到任务寄存器TR中，并自动将任务状态段描述符装入到内部相应的64位的描述符寄存器中。每个任务都有自己的任务状态段。

(11)保存任务状态段寄存器指令

格式：STR　r16/m16

功能：STR指令将任务状态段寄存器TR的16位选择符存放到由操作数所表示的双字节寄存器或内存单元中。

在上述指令中，LLDT、LTR、SLDT和STR只能在保护模式下使用。

12. 保护属性检查指令

(1)装入访问权限指令

格式：LAR　r16/r32，r16/m16

功能：LAR为装入访问权限指令。对32位属性，该指令将第二操作数所表示的选择符对应的描述符的第二个双字与00F0FF00H相“与”，取出相应访问权限部分送第一操作数所指定的32位通用寄存器，并将ZF位置1；对16位属性，则与0000FF00H相“与”后，其低字送16位寄存器。如果描述符的类型为有效的，并且访问是合法的（即CPL和RPL均小于等于DPL），上述操作才是有效的，否则只将ZF位清0。

(2)装入段限制指令

格式：LSL　r16/r32，r16/m16

功能：LSL为装入段限制指令。如果该指令的第二操作数所表示的选择符对应的描述符是有效的，并且对它的访问是合法的，就将描述符的段限制送入指令中指定的寄存器中，并将ZF位置1；否则只将ZF位清0，不改变寄存器内容。对于LSL指令的32位形式，如果限制单位是页，还要将限制换算成字节（左移12位再与00000FFFH相“与”）装入32位寄存器。

(3)验证段的可读性指令

格式：VERR　r16/m16

功能：VERR指令验证段的可读性。若指令中指出的选择符所对应的段是可读的，并且访问是合法的，则将ZF位置1；否则将ZF位清0。

(4)验证段的可写性指令

格式：VERW　r16/m16

功能：VERW指令验证段的可写性。若指令中指出的选择符所对应的段是可写的，并且访问是合法的，则将ZF位置1；否则将ZF位清0。

(5)调整选择符的特权级指令

格式：ARPL　r16/m16，r16

功能：ARPL指令用于调整选择符的特权级。ARPL指令带有两个操作数：第一操作数是一个16位选择符，存放在存储器中；第二操作数也是选择符，存放在寄存器中，该指令将两个选择符的特权级进行比较，若第一选择符的特权级高于第二选择符的特权级，则将第一选择符的特权级降低到与第二选择符的特权级相等。并将ZF位置1；否则只将ZF位清0。使用该指令能有效地防止用户破坏操作系统的数据。

13. 高级语言指令

80386指令系统新增了3个高级语言支持指令：BOUND、ENTER和LEAVE。

80486/ Pentium微处理器新增指令

PART 04

80486引入的有关的指令

（1）BSWAP双字交换指令

指令格式：BSWAP　r32

指令将指定的32位寄存器中双字的第31~24位与第7~0位交换，第23~16位与第15~8位交换，以此改变数据的存放方式。

（2）CMPXCHG比较交换指令

指令格式：CMPXCHG　r/m，r

指令将目的寄存器或存储器中的数和累加器中的数比较，如相等，则ZF为1，并将源操作数送目的操作数；否则ZF为0，并将目的操作数送累加器。

注意：

CMPXCHG 指令前可以加 LOCK 前缀。

操作数为8位、16位、32位通用寄存器或存储单元。

(3)XDDD　r/m, r　　字交换加法指令

指令将源操作数和目的操作数相加，其中，源操作数必须为寄存器，目的操作数可为寄存器或存储器数，结果送入目的操作数处，而目的操作数送源操作数处。

(5)WBINVD Cache清除和回写指令

指令将片内Cache中的内容清除，并启动一个回写总线周期，使外部电路将外部Cache中的数据回写到主存，再清除Cache中的内容

(6)INVLPG m　TLB项清除指令

指令使转换检测缓冲器TLB的32个表项中用m指出的当前项清除。分页机构为实现从线性地址到物理地址的转换，使用到转换检测缓冲器TLB，在转换检测缓冲器TLB中存放了32个最新的页表项，指令INVLPG m能修改转换检测缓冲器TLB。

Pentium 引入的有关指令

Pentium在80486基础上增加了3条处理器专用指令和5条系统控制指令。

（1）CMPXCHG8B　m　　8字节即64位比较指令

与80486的CMPXCHG　r/m, r指令类似，将EDX、EAX中的8个字节与m所指的存储器中的8个字节比较，如相等，则ZF为1，并将ECX、EBX中8个字节数据送到目的存储单元；否则ZF为0，将目的寄存器中8个字节数据送到EDX、EAX中。该指令可加LOCK前缀。

（2）RDTSC 读时钟周期指令

读取CPU中用于记录时钟周期数的64位计数器的值，并将读取的值送EDX、EAX，供有些应用软件通过前后两次执行RDTSC指令来确定执行某段程序需要多少时钟周期。

（3）CPUID 读取CPU的标识等有关信息

指令用来获得Pentium处理器的类型等有关信息。在执行此指令前，EAX中如为0，则指令执行后，EAX、EBX、ECX、EDX中内容合起来为Intel产品的标识字符串，如此前EAX中为1，则指令执行后，在EAX、EBX、ECX、EDX中得到CPU的级别（如PIV、PIII）、工作模式、可设置的断点数等。

（4）RDMSR 读取模式专用寄存器的指令

指令读取Pentium模式专用寄存器中的值。执行指令前，在ECX中设置寄存器号，可为0~14H，指令执行后，读取的内容在EDX、EAX中。

（5）WRMSR 写入模式专用寄存器的指令

指令将EDX、EAX中64位数写入模式专用寄存器，此前，ECX中先设置模式专用寄存器号，可为0~14H。

（6）RSM　复位到系统管理模式

（7）写控制寄存器CR_4 内容

例

MOV　CR_4，r32

；将32位寄存器中的内容送控制寄存器CR_4

（8）读控制寄存器CR_4 内容

例

MOV　r32，CR_4

；将CR_4中的内容送32位寄存器R32

以上指令，前3条为处理器专用指令，后5条为系统控制指令。

小 结

本章首先介绍了8086指令系统中的数据类型、指令格式，在此基础上，较详细地介绍了8086微处理器的13种寻址方式，包括9种操作数寻址方式和4种程序转移地址的寻址方式。然后，详细地讲解了8086的指令系统及它们的一些使用实例。8086的指令系统包括基本的数据传送类指令、算术运算指令、逻辑运算指令、移位和循环指令、位操作指令等。同时本章对80386微处理器的寻址方式和指令系统进行了介绍，重点讲述了80386与8086指令的区别以及80386新增指令的功能。另外，对80486和Pentium微处理器引入的有关指令做了简单介绍。