指令中操作数寻址方式的基本约定：

双操作数指令：源操作数和目的操作数必须有一个为通用寄存器（除非源操作数是立即数）。
单操作数指令：操作数不允许为立即数。

一般的，源操作数：　通用寄存器、内存操作数或立即数
目的操作数：通用寄存器、内存操作数，

4.3.2　算术运算指令

1．加法指令

```
ADD   OPRD1，OPRD2   ;(add)
```
功能：　（OPRD1）＋（OPRD2）→ (OPRD1)

```
ADC   OPRD1，OPRD2   ;(add with carry)
```
功能：　（OPRD1）＋（OPRD2）＋CF → (OPRD1)

```
INC   OPRD           ;(increment)
```
功能：　（OPRD）＋1 → (OPRD)

- 影响标志位 AF、 CF、 OF、 PF、 SF、ZF（INC 不影响 CF）。

例：

```
ADD   AL，30
ADD   BX，3000H
ADD   AX，SI
ADD   DX，DATA[BX＋SI]
ADD   [SI]，100
ADD   [BX]，AX
```

例：ADC 指令用于多字节运算

```
      （DX、AX）= 0002，F365H
    +（BX、CX）= 0005，E024H
    ------------------------------
```
双精度加法

```
ADD   AX，CX
ADC   DX，BX
```

注：

```
第一条指令，      F365
                + E024
                ---------
              ↙1，D389
        CF=1          结果  （AX）= D389H，CF=1，SF=1，ZF=0，  OF=0

第二条指令，      0002
                  0005
              +      1（CF）
                ---------
                  0008
        CF=0          结果  （DX）= 0008H，CF=0，SF=0，ZF=0，  OF=0
```

例：INC 常用于在循环程序中修改地址指针和循环次数等。

```
INC  AL
INC  [SI]
```

例：1000H：0100H 为首地址的 1000 字节，传送到 2000H：0000H 开始的内存区。

```
        MOV  AX，1000H
        MOV  DS，AX
        MOV  SI，0100H      ; DS:SI = 1000H:0100H
        MOV  AX，2000H
        MOV  ES，AX
        MOV  DI，0000H      ; ES:DI = 2000H:0000H
        MOV  CX，1000
NEXT:   MOV  AL，[SI]       ; (DS:SI)→AL
        MOV  ES:[DI]，AL    ; AL→(ES:DI)
        INC  SI
        INC  DI
        LOOP NEXT
        HLT
```

例：以 1000H：0100H 为首地址的 10 个无符号字节数求和。

```
        MOV  AX，1000H
        MOV  DS，AX
        MOV  SI，0100H
        MOV  DX，0
        MOV  CX，10
NEXT:   MOV  AL，[SI]
        ADD  DL，AL
        ADC  DH，0
        INC  SI
        LOOP NEXT
        HLT
```

2．减法指令

```
SUB  OPRD1，OPRD2   ;(subtract)
     功能：  （OPRD1）-（OPRD2）→（OPRD1）

SBB  OPRD1，OPRD2   ;(subtract with borrow)
     功能：  （OPRD1）-（OPRD2）- CF →(OPRD1)

DEC  OPRD           ;(decrement)
     功能：  （OPRD）-1 →（OPRD)

NEG OPRD            ;(negate) 取(求)补指令
     功能：  FFFFH -（OPRD）+1 →（OPRD)

CMP OPRD1，OPRD2    ;(compare) 比较指令
     功能：  （OPRD1）-（OPRD2）
```

- 影响标志位 AF、 CF、 OF、 PF、 SF、ZF（DEC 不影响 CF）。

例：比较指令的使用

- 两个无符号数的比较
- 两个带符号数的比较

例：比较 DL、AL 中数据的大小

```
        CMP   DL，AL
        JNC   NEXT     ；若 CF=0（即 DL≥AL），跳转到 NEXT
        MOV   DL，AL
NEXT：  ……
```

3．乘法指令

(1) 无符号数乘法指令 MUL

```
MUL   OPRD   ;(unsigned multiple)
功能：  若 OPRD 为字节，（AL）*（OPRD）→  （AX）
        若 OPRD 为字，   （AX）*（OPRD）→  （DX、AX）
```

(2) 带符号数乘法指令 IMUL

```
IMUL   OPRD  ;(signed multiple)
功能：  若 OPRD 为字节，（AL）*（OPRD）→  （AX）
        若 OPRD 为字，   （AX）*（OPRD）→  （DX、AX）
```

- 采用 MUL 或 IMUL 由运算数的格式决定。

例：

```
MOV BL,10
MUL BL     ; AL * BL → AX
```

例：

```
MOV BX,10
MUL BX     ; AX * BX → DX,AX
```

4．除法指令

(1)无符号数除法指令 DIV

```
DIV   OPRD   ;(unsigned divide)
功能：  若 OPRD 为字节，     （AX）÷（OPRD）商   →  （AL）
                                  余数 →  （AH）
        若 OPRD 为字，   （DX，AX）÷（OPRD）商   →  （AX）
                                  余数 →  （DX）
```

(2) 带符号除法指令 ID1V

```
IDIV   OPRD   ;(signed divide)
功能：  若 OPRD 为字节，     （AX）÷（OPRD）商   →  （AL）
                                  余数 →  （AH）
        若 OPRD 为字，   （DX，AX）÷（OPRD）商   →  （AX）
                                  余数 →  （DX）
```

- 采用 DIV 或 IDIV 由运算数的格式决定。

4.3.3 逻辑运算和移位指令

1. 逻辑运算

NOT OPRD

*功能：NOT（OPRD）→（OPRD）；**按位**取反*

AND OPRD1，OPRD2

*功能：（OPRD1）AND（OPRD2）→（OPRD1）；**按位**与*

OR OPRD1，OPRD2

*功能：（OPRD1）OR（OPRD2）→（OPRD1）；**按位**或*

XOR OPRD1，OPRD2

*功能：（OPRD1）XOR（OPRD2）→（OPRD1）；**按位**异或*

TEST OPRD1，OPRD2

*功能：（OPRD1）AND（OPRD2）；按位与，**结果不保存***

例：逻辑指令的使用

AND *指令使操作数的某些位不变，其它位为 0。*

OR *指令使操作数的某些位不变，其它位为 1。*

XOR *指令使操作数的某些位不变，其它位取反。*

TEST *指令在不操作数的前提下，用来检测某几位的状态。*

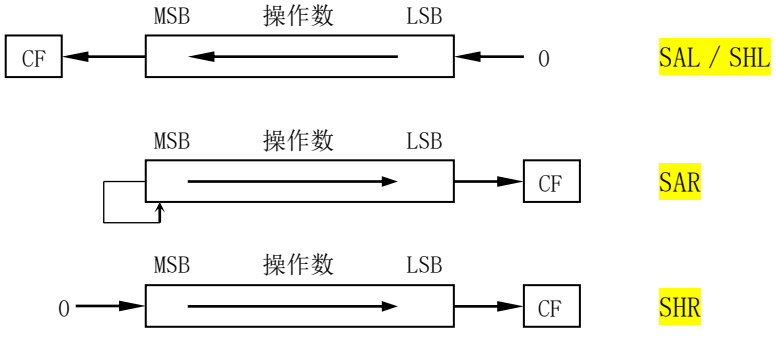
2. 移位指令

算术左移和逻辑左移指令 SAL / SHL OPRD，m ;(shift arithmetic/logical left)

算术右移指令 SAR OPRD，m ;(shift arithmetic right)

逻辑右移指令 SHR OPRD，m ;(shift logical right)

- OPRD 为通用寄存器 或 内存操作数。
- **m 为立即数 1 或 CL，**决定移位次数。



例：

SHR AL,2 ；错，改为 MOV CL,2

SHR AL,CL

*SHR AL,1 ；若之前 AL= 10110111,则之后 AL= **0**10110111 ~~1~~,CF=1*

*SAR AL,1 ；若之前 AL= 10110111,则之后 AL= **11**0110111 ~~1~~(补码右移),CF=1*

3. 循环移位指令

左循环移位指令　　ROL　OPRD，m　;(rotate left)

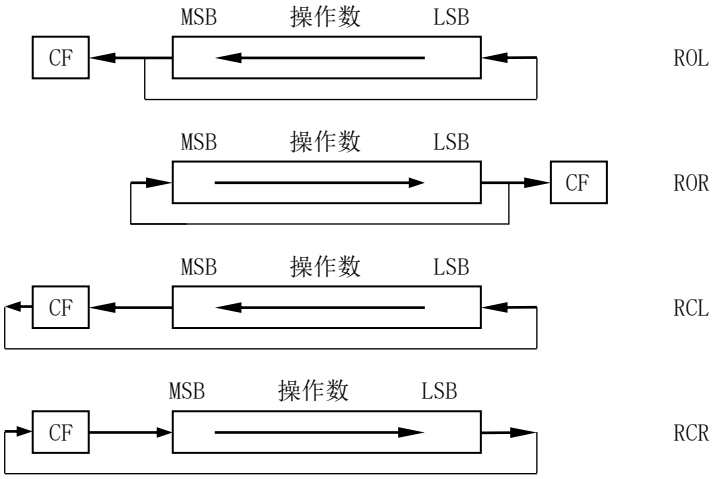
右循环移位指令　　ROR　OPRD，m　;(rotate right)

带进位左循环移位指令　RCL　OPRD，m　;(rotate left　through carry)

带进位右循环移位指令　RCR　OPRD，m　;(rotate right through carry)

- OPRD 为通用寄存器 或 内存操作数。

  m 为立即数 1 或 CL，决定循环移位次数。



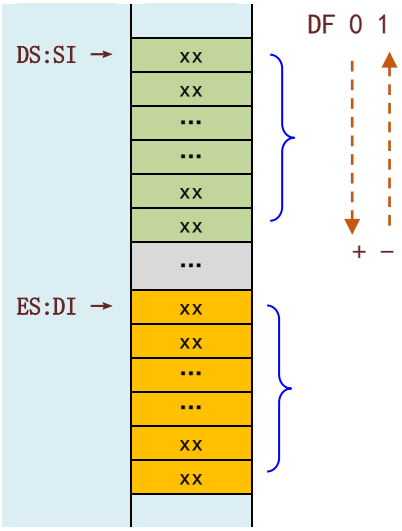
4.3.4 串操作指令

串操作：某个连续的内存区中存放的一串字或字节，可以是 BIN 数，也可以是 BCD 码或 ASCII 码。若对其中的每个字或字节均作同样的操作，就称为串操作。

串操作指令：完成串操作功能的指令称为字符串操作指令，简称为串操作指令。

串操作指令中的默认寻址：

DS:SI 源串
ES:DI 目的串
CX 计数器
AL/AX 累加器
DF 方向标志，DF=0，+1/+2；DF=1，-1/-2



1. MOVS、STOS、LODS 与 REP 配合

- MOVS 串传送指令

```
MOVS    DST，SRC  ；SRC 源串，  DST 目的串
MOVSB             ；字节传送
MOVSW             ；字传送
```

功能：（DS：SI）→ （ES：DI）
（SI）± 1/2 → （SI）
（DI）± 1/2 → （DI）

- REP 串指令前缀

```
REP  MOVS
REP  STOS
REP  LODS
```

功能：重复串指令执行，重复次数为（CX）。过程如下：
1、若（CX）=0，则退出，否则执行 2；
2、（CX）-1 → （CX）；
3、执行串指令；
4、重复 1 - 3。

例：1000H：0100H 为首地址，长度为 1000 字节的串，传送到 2000H：0000H 开始的串。

```
MOV  AX，1000H
MOV  DS，AX
MOV  SI，0100H      ; DS:SI = 1000H:0100H
MOV  AX，2000H
MOV  ES，AX
MOV  DI，0000H      ; ES:DI = 2000H:0000H
MOV  CX，1000
CLD                 ; DF=0
REP MOVSB
```

- STOS 串存贮指令（store to string）

  STOS DST ；DST 目的串
  STOSB ；字节操作
  STOSW ；字操作
  功能： （AL）/(AX）→ （ES：DI）
  （DI）± 1/2 → （DI）

- LODS 串装入指令 （load from string）

  LODS SRC ； SRC 源串
  LODSB ；字节操作
  LODSW ；字操作
  功能： （DS:SI）→ （AL）/(AX)
  （SI）± 1/2 → （DI）

2. CMPS、SCAS 与 REPE、REPNE 配合

- CMPS 串比较指令

  CMPS SRC，DST ；SRC 源串， DST 目的串
  CMPSB ；字节比较
  CMPSW ；字比较
  功能： （DS：SI）- （ES：DI） ；设定标志位
  （SI）± 1/2 → （SI）
  （DI）± 1/2 → （DI）

- SCAS 串扫描指令 (scan string)

  SCAS DST ；DST 目的串
  SCASB ；字节操作
  SCASW ；字操作
  功能： （AL）/（AX）-（ES：DI） ；设定标志位
  （DI）± 1/2 → （DI）

- REPE/REPNE 条件重复前缀

  REPE/REPNE CMPS
  REPE/REPNE SCAS
  功能：
  REPE 相等重复指令，当 ZF＝1 且 CX 未减到 0，串指令重复执行。
  REPNE 不相等重复指令，当 ZF＝0 且 CX 未减到 0，串指令重复执行。

例：1000H：0100H 为首地址，长度为 1000 字节的串，传送到 2000H：0000H 开始的串。

```
MOV  AX，1000H
MOV  DS，AX
MOV  SI，0100H
MOV  AX，2000H
MOV  ES，AX
MOV  DI，0000H
MOV  CX，1000
CLD          ；DF=0
REP MOVSB
```

例：1000H：0100H 为首地址的源串，2000H：0000H 为首地址的目的串
比较源串和目的串的 500 个字节，找出第一个相同的字节，如果找到，则将这个数送 AL 中。

```
        MOV    AX，1000H
        MOV    DS，AX
        MOV    SI，0100H
        MOV    AX，2000H
        MOV    DS，AX
        MOV    DI，0000H
        MOV    CX，500
        CLD
        REPNE CMPB
        JNZ    NEXT
MATCH：DEC    SI
        MOV    AL，[SI]
 NEXT：  ┆
```

4.3.5 程序控制指令

- 程序控制指令主要是指**程序转移指令**。
- 转移类指令通过改变 CS 与 IP 的值或仅改变 IP 的值，以改变指令执行的顺序。

1. 无条件转移指令 JMP

该指令分为 直接转移 和 间接转移 两种。

其中，直接转移又可分 短程(SHORT)、近程(NEAR) 和 远程(FAR) 三种形式。

- 短程转移 JMP SHORT PTR OPRD ；*IP = IP + 8 位位移量，段内转移*
- 近程转移 JMP NEAR PTR OPRD ；*IP = IP + 16 位位移量，段内转移*
- 远程转移 JMP FAR PTR OPRD ；*CS、IP 均改变，段间转移*

2. 调用和返回指令

CALL 指令用来调用一个过程，在过程执行完后可使用返回指令 **RET**，使程序返回调用程序继续执行。

```
CALL  NEAR PTR  OPRD ；段内（即近程 NEAR）调用
CALL  FAR  PTR  OPRD ；段间（即远程 FAR）调用
```

其中 OPRD 为被调用的过程名（首地址）。

在段内调用时，CALL 指令相当于 *PUSH IP*
JMP SHORT（NEAR）PTR OPRD

RET 指令相当于 *POP IP*。

在段间调用时，CALL 指令相当于 *PUSH CS*
PUSH IP
JMP FAR PTR OPRD

RET 指令相当于 *POP IP*
POP CS

3. 条件转移指令

条件转移指令将上一条指令所设置的某些标志位的状态作为测试条件，条件满足则转向指令中所指示的目的地址。**条件转移指令只能为 短程(SHORT) 一种形式。**

(1) 以单个标志位为条件

```
JO          OPRD          ; OF=1，溢出转移
JNO         OPRD          ; OF=0，不溢出转移
JS          OPRD          ; SF=1，结果为负转移
JNS         OPRD          ; SF=0，结果为正转移
JC          OPRD          ; CF=1，进位转移
JNC         OPRD          ; CF=0，无进位则转移
JE / JZ     OPRD          ; ZF=1，等于或为零转移
JNE / JNZ   OPRD          ; ZF=0，不等于或不为零转移
```

```
例：        CMP   DL，AL ; DL-AL
            JNC   NEXT   ; 若 CF=0，即 DL≥AL，跳转到 NEXT
            MOV DL，AL   ; 若 CF=1，即 DL<AL，  DL←AL
    NEXT:   ·····        ; DL 中存放 max{DL,AL}
```

(2) 无符号数比较

```
JA / JNBE   OPRD          ; 高于或不低于等于转移
JAE / JNB   OPRD          ; 高于等于或不低于转移
JB / JNAE   OPRD          ; 低于或不高于等于转移
JBE / JNA   OPRD          ; 低于等于或不高于转移
```

其中，A above，B below，E equal，N not

```
例：        CMP   DL，AL ; DL-AL
            JAE   NEXT   ; 若 DL≥AL（无符号数），跳转到 NEXT
                         ; 等价于 JNC
            MOV DL，AL   ; 若 DL<AL，  DL←AL
    NEXT:   ·····        ; DL 中存放 max{DL,AL}无符号数
```

(3) 带符号数比较

```
JG / JNLE   OPRD          ; 大于或不小于等于转移
JGE / JNL   OPRD          ; 大于等于或不小于转移
JL / JNGE   OPRD          ; 小于或不大于等于转移
JLE / JNG   OPRD          ; 小于等于或不大于转移
```

其中，G greater，L less，E equal，N not

```
例：        CMP   DL，AL ; DL-AL
            JGE   NEXT   ; 若 DL≥AL（有符号数），跳转到 NEXT
            MOV DL，AL   ; 若 DL<AL，  DL←AL
    NEXT:   ·····        ; DL 中存放 max{DL,AL}有符号数
```

4. 循环控制指令 LOOP

```
LOOP    OPRD        ; CX≠0 循环，相当于  DEC  CX
                                          JNZ  OPRD
```

例：1000H：0100H 为首地址的 1000 字节，传送到 2000H：0000H 开始的内存区

```
        MOV  AX，1000H
        MOV  DS，AX
        MOV  SI，0100H  ; DS:SI
        MOV  AX，2000H
        MOV  ES，AX
        MOV  DI，0000H  ; ES:DI
        MOV  CX，1000
NEXT:   MOV  AL，[SI]   ; (DS:DI)-- AL
        MOV  ES：[DI]，AL  ；段超越，指定 ES 为段寄存器 ES:DI, AL---(ES:DI)
        INC  SI
        INC  DI
        LOOP NEXT   ;  DEC  CX
                    ;  JNZ  NEXT
        HLT
```

例：以 1000H：0100H 为首地址的 10 个无符号字节数，求最大数。

```
        MOV  DX，1000H
        MOV  DS，DX
        MOV  SI，0100H   ; DS:SI = 1000H:0100H，指向第 1 个数
        MOV  DL，[SI]    ; 取第 1 个数到 DL
        INC  SI          ; SI = 0101H，指向第 2 个数
        MOV  CX，9
AGAIN:  MOV  AL，[SI]
        CMP  DL，AL      ; DL-AL
        JAE  NEXT        ; DL≥AL
        MOV  DL，AL      ; DL 中存放 max{DL,AL}
NEXT:   INC  SI          ; SI 加 1，指向下一个数
        LOOP AGAIN
        HLT
```

例：以 1000H：0100H 为首地址的 10 个有符号字节数，求最大数

```
        MOV  DX，1000H
        MOV  DS，DX
        MOV  SI，0100H
        MOV  DL，[SI]
        INC  SI
        MOV  CX，9
AGAIN:  CMP  DL，[SI]
        JGE  NEXT       ;大于等于
        MOV  DL，[SI]
NEXT:   INC  SI
        LOOP AGAIN
        HLT
```

例：以 1000H：0100H 为首地址的 10 个无符号字节数求和

```
        MOV  AX, 1000H
        MOV  DS, AX
        MOV  SI, 0100H
        MOV  DX, 0
        MOV  CX, 10
NEXT:   MOV  AL, [SI]
        MOV  AH, 0
        ADD  DX, AX
        INC  SI
        LOOP NEXT     ;  DEC  CX
                      ;  JNZ  NEXT
        HLT
```

4.3.6 处理器控制指令

1. 标志位操作指令

```
STC 、CLC、CMC    ; CF （ Set, CLear, CoMplement）
STD 、CLD         ; DF
STI 、CLI         ; IF
```

2. 外部同步指令

```
HLT    ; 暂停指令
NOP    ; 空操作指令
WAIT   ; 等待指令
```

4.3.7 输入输出指令

输入指令的一般格式为：

```
IN   ACC,  PORT     ; 从接口到 CPU 的输入
```

输出指令的一般格式为：

```
OUT  PORT, ACC      ; 从 CPU 到接口的输出
```

其中，PORT 为接口地址

1. 直接寻址：接口地址用一个字节表示，故可寻址的接口地址只有 256 个（00H 到 FFH）。

例如：

```
IN   AL,  35H
OUT  44H, AX
```

2. 寄存器间接寻址：接口地址为（DX），寻址的接口地址为 64 K 个（0000H 到 FFFFH）。

例如：

```
MOV   DX，03F8H
IN    AL，DX
OUT   DX，AL
```

- *应注意的是：1）I/O 寻址：［ 35H ］，［ DX ］，不同于内存寻址。*
  2）采用直接寻址仅限于接口地址的前 256 个（0～FFH），
  当接口地址≥256 时，必须采用 DX 间接寻址。