12.3.2 算术运算指令

- 加法运算指令
- 减法运算指令
- 乘法指令
- 除法指令

1 加法指令

- 普通加法指令ADD
- 带进位位的加法指令ADC
- 加1指令INC

加法指令的源操作数可以是通用寄存器、存储单元或立即数，但目的操作数只能是通用寄存器或存储单元，不能是立即数，并且源操作数和目的操作数不能同时为存储器操作数。

1）ADD指令

- 格式：

  ADD OPRD1，OPRD2

- 操作：

  OPRD1+OPRD2→OPRD1

ADD指令的执行对全部6个状态标志位都产生影响

例，

ADD CL，20H

ADD DX，[BX+SI]

合法指令

ADD [BX]，[SI]

ADD DS，AX

非法指令

例， MOV AL，78H

ADD AL，99H

分析指令执行后6个状态标志位的状态。

$$\begin{array}{rr} & 01111000 \\ + & 10011001 \\ \hline \boxed{1} & 00010001 \end{array}$$

标志位状态： CF= 1　　SF= 0

AF= 1　　ZF= 0

PF= 1　　OF= 0

2）ADC指令

- 格式：

ADC OPRD1，OPRD2

- 操作：

OPRD1+OPRD2+CF→OPRD1

ADC指令主要用于多字节加法运算，高16位（或高8位）相加时，必须要考虑低16位（低8位）产生的进位。

ADC指令的执行对全部6个状态标志位都产生影响

例，现有两个32位无符号数12345678H，8765ABCDH相加，其和仍然为一个32位无符号数，要求和的高16位送入DX中，和的低16位送入AX中。

```
MOV  AX, 5678H
ADD  AX, 0ABCDH
MOV  DX, 1234H
ADC  DX, 8765H
```

3）INC指令

格式：INC OPRD

操作：OPRD+ 1→OPRD

单操作数指令，其目的操作数可以是通用寄存器，也可以是存储单元，但是不能是立即数。

根据运算结果设置标志寄存器中的PF、AF、ZF、SF和OF位，但不影响CF位。

2 减法指令

- 普通减法指令SUB
- 带借位的减法指令SBB
- 减1指令DEC
- 求补指令NEG
- 比较指令CMP

减法指令对操作数的要求与对应的加法指令相同

1）SUB指令

- 格式：

  SUB OPRD1，OPRD2

- 操作：

  OPRD1−OPRD2→OPRD1

SUB指令的执行对全部6个状态标志位都产生影响

2）SBB指令

- 格式：

SBB OPRD1，OPRD2

- 操作：

OPRD1−OPRD2−CF→OPRD1

SBB指令主要用于多字节减法运算，高16位（或高8位）相减时，必须要考虑低16位（低8位）产生的借位。

SBB指令的执行对全部6个状态标志位都产生影响

3）DEC指令

格式：DEC OPRD

操作：OPRD−1→OPRD

该指令对操作数的要求及对标志位的影响与INC指令相同。

4）NEG求补指令

- 格式：

  NEG OPRD

- 操作：

  0−OPRD→OPRD

NEG指令的执行对全部6个状态标志位都产生影响

- 对进位标志CF的影响
  - 只有当操作数为零时，进位标志CF被置零；其它情况都被置1，即均有借位。
- 对溢出标志OF的影响
  - 当字节操作数为-128（80H），或字操作数为-32768（8000H）时，结果将无变化，但溢出标志OF被置1。

5）CMP比较指令

- 格式：

  CMP OPRD1，OPRD2

- 操作：

  OPRD1−OPRD2

CMP指令的执行对全部6个状态标志位都产生影响

用于比较两个数的大小，可作为条件转移指令的转移条件。

简单条件转移指令

标志位	指令	转移条件	含义
CF	JC	CF=1	有进位/借位转移
	JNC	CF=0	无进位/借位转移
ZF	JE/JZ	ZF=1	相等/等于0转移
	JNE/JNZ	ZF=0	不相等/不等于0转移
SF	JS	SF=1	负数转移
	JNS	SF=0	正数转移
OF	JO	OF=1	有溢出转移
	JNO	OF=0	无溢出转移
PF	JP/JPE	PF=1	偶数个1转移
	JNP/JPO	PF=0	奇数个1转移

CMP A, B

无符号数条件转移指令

指令	转移条件	含义
JA/JNBE	CF=0且ZF=0	A>B
JAE/JNB	CF=0	A≥B
JB/JNAE	CF=1	A<B
JBE/JNA	CF=1或ZF=1	A≤B

有符号数条件转移指令

指令	转移条件	含义
JG/JNLE	SF=OF且ZF=0	A>B
JGE/JNL	SF=OF或ZF=1	A≥B
JL/JNGE	SF≠OF且ZF=0	A<B
JLE/JNG	SF≠OF或ZF=1	A≤B

两个无符号数的大小关系　CMP A，B

- 当A>B时，此时A-B不产生借位，并且结果不为0，即CF=0并且ZF=0
- 当A=B时，则ZF=1
- 当A<B时，此时A-B产生借位，并且结果不为0，即CF=1并且ZF=0

两个有符号数的大小关系 CMP A，B

A>B

（1） A和B都为负数

此时，若要A>B，则A-B的结果一定是正数（SF=0),并且不发生溢出（OF=0），并且结果不为零（ZF=0）。

（2）A和B都为正数

此时，若要A>B，则A-B的结果一定是正数（SF=0），并且不发生溢出（OF=0），并且结果不为零（ZF=0）。

（3）A为正数，B为负数

不发生溢出（OF=0），这时结果为正数（SF=0），并且结果不为零（ZF=0）。

发生溢出（OF=1），这时结果变为负数（SF=1），并且结果不为零（ZF=0）。

因此SF⊕OF=0，且ZF=0时，则A>B;

A<B

（1） A和B都为负数

此时，若要A<B，则A-B的结果一定是负数（SF=1),并且不发生溢出（OF=0），并且结果不为零（ZF=0）。

（2）A和B都为正数

此时，若要A<B，则A-B的结果一定是负数（SF=1），并且不发生溢出（OF=0），并且结果不为零（ZF=0）。

（3）A为负数，B为正数

不发生溢出（OF=0），这时结果为负数（SF=1），并且结果不为零（ZF=0）。
发生溢出（OF=1），这时结果变为正数（SF=0），并且结果不为零（ZF=0）。

因此，当SF ⊕ OF=1，且ZF=0时，则A<B。

- 当A=B时，则ZF=1

例，在内存数据段从DATA1开始的存储单元存放了两个8位的无符号数，试比较它们的大写，并将大的数送MAX存储单元。

```
        LEA     BX,     DATA1
        MOV     AL,     [BX]
        INC     BX
        CMP     AL,     [BX]
        JNC     DONE
        MOV     AL,     [BX]
DONE:   MOV     MAX,    AL
        HLT
```

```
        MOV   AL,   DATA1
        CMP   AL,   DATA1+1
        JAE   DONE
        MOV   AL,   DATA1+1
DONE:   MOV   MAX,  AL
        HLT
```

3 乘法指令

- 无符号的乘法指令MUL
- 带符号的乘法指令IMUL

➢乘法指令采用隐含寻址，隐含的是存放被乘数的累加器AL或AX及存放结果的AX，DX；

➢若运算结果的高半部分是无效数值，则OF=CF=0，否则OF=CF=1。

1）MUL指令

- 格式：

  MUL OPRD

- 操作：

  - 字节乘法：OPRD×AL→AX
  - 字乘法：OPRD×AX→DX:AX

OPRD为通用寄存器或存储器操作数

- **MUL只对CF和OF标志产生有效影响,其他标志位的值不确定。**
- **若结果的AH（字节运算）或DX（字运算）为全0，则CF=OF=0，否则 CF=OF=1。**

2）IMUL指令

- 格式：

  IMUL OPRD

- 操作：
  - 字节乘法 ： OPRD×AL→AX
  - 字乘法 ： OPRD×AX→DX:AX

对标志位的影响：若乘积的高半部（AH或DX）是低半部的符号扩展，则CF=OF=0 ， 否则CF=OF=1 。

4 除法指令

- 无符号的除法指令DIV
- 带符号的除法指令IDIV

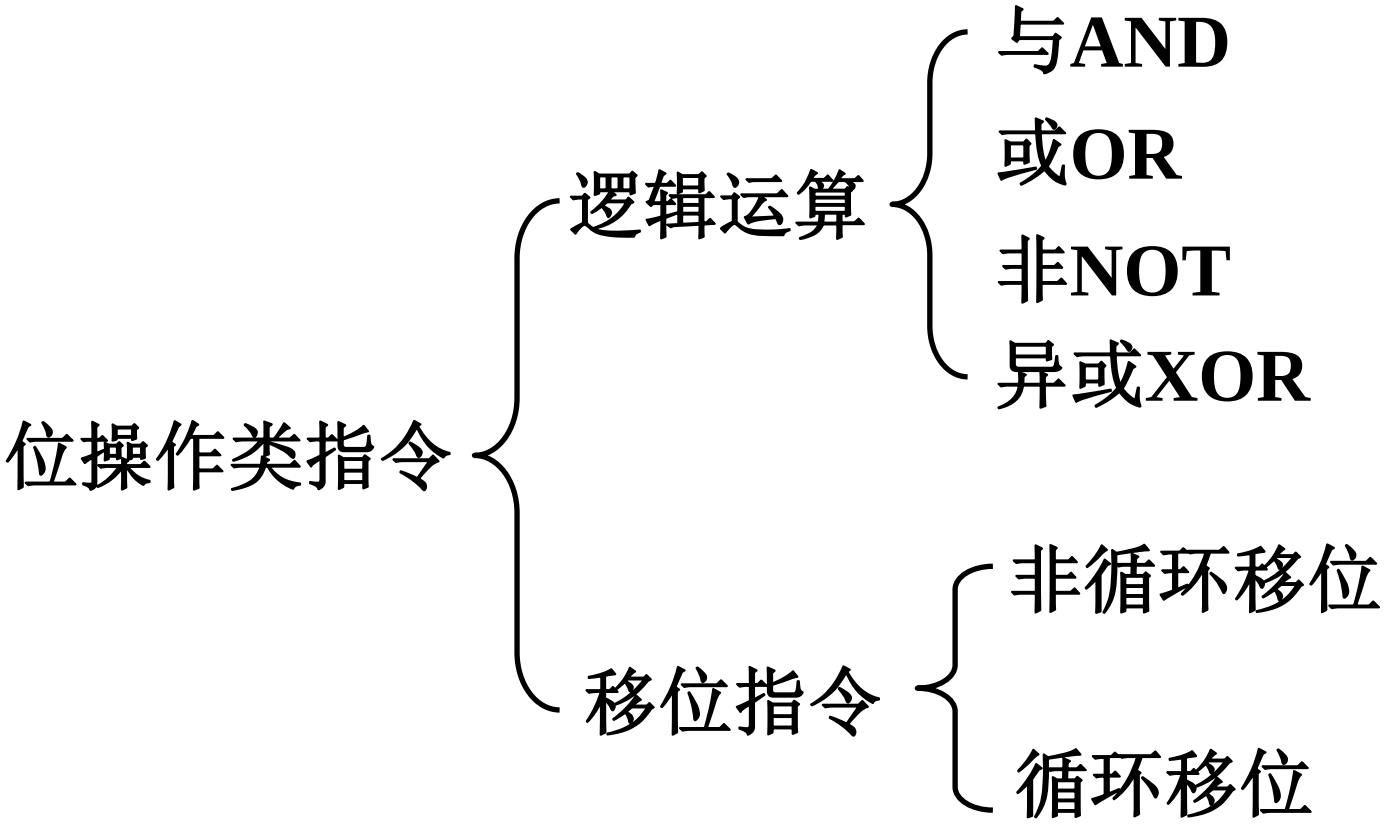
➢除法指令要求被除数的字长必须是除数的两倍；

➢除法指令采用隐含寻址，若除数为8位，则被除数为AX；若除数为16位，则被除数高位在DX中，低位在AX中。

- 格式：
  - DIV OPRD
  - IDIV OPRD
- 操作：
  - 字节除法：AX÷OPRD→AL(商)AH(余数)
  - 字乘法：DX:AX ÷ OPRD→AX(商)DX(余数)

除法指令对标志位均无影响

12.3.3 逻辑运算和移位指令



1 逻辑运算指令

1）逻辑与指令

- 格式：

AND　OPRD1，OPRD2

- 操作：

OPRD1∧OPRD2→OPRD1

实现两个操作数的按位相与。

指令对操作数的要求与对应的加法指令相同

指令执行后会影响6个状态标志位，并CF=OF=0

主要应用：

①AND指令主要用于使目的操作数某些位保持不变而另一些位清0。要执行这样的操作就是将要保持不变的位与“1”相与，将要清0的位与“0”相与。

②使操作数不变，但影响6个状态标志位，并使CF=OF=0。例如：

```
AND  AX，AX
```

后续指令会根据需要对状态标志进行判断处理。

2）逻辑或指令

- 格式：

OR　OPRD1，OPRD2

- 操作：

OPRD1∨OPRD2→OPRD1

①实现两个操作数的按位相或。

②OR指令主要用于使目的操作数某些位保持不变而另一些位置1。要执行这样的操作就是将要保持不变的位与“0”相或，将要置1的位与“1”相或。

③使操作数不变，但影响6个状态标志位，并使CF=OF=0。例如：

```
OR  AX,  AX
```

例，若AL中低7位为有效数据，最高位留作校验位，初始值为0，将AL中的数据生成偶校验码。

```
        OR  AL，AL
        JPE  GOON
        OR  AL，80H
GOON：….
```

3）逻辑非指令

- 格式：

  NOT　OPRD

- 操作：

  $\overline{\text{OPRD}} \rightarrow \text{OPRD}$

①实现操作数的按位取反。

②对所有标志位无影响。

4）逻辑异或指令

- 格式：

XOR　OPRD1，OPRD2

- 操作：

OPRD1⊕OPRD2→OPRD1

①XOR指令主要用于使目的操作数某些位保持不变,而另一些位变反。要执行这样的操作就是将要保持不变的位与“0”相异或，将要变反的位与“1”相异或。

②若两个操作数相同，则结果为0。

③影响6个状态标志位，且CF=OF=0。

5）测试指令

- 格式：

TEST　OPRD1，OPRD2

- 操作：

OPRD1∧OPRD2

两操作数按位相与但结果不送回目的操作数；

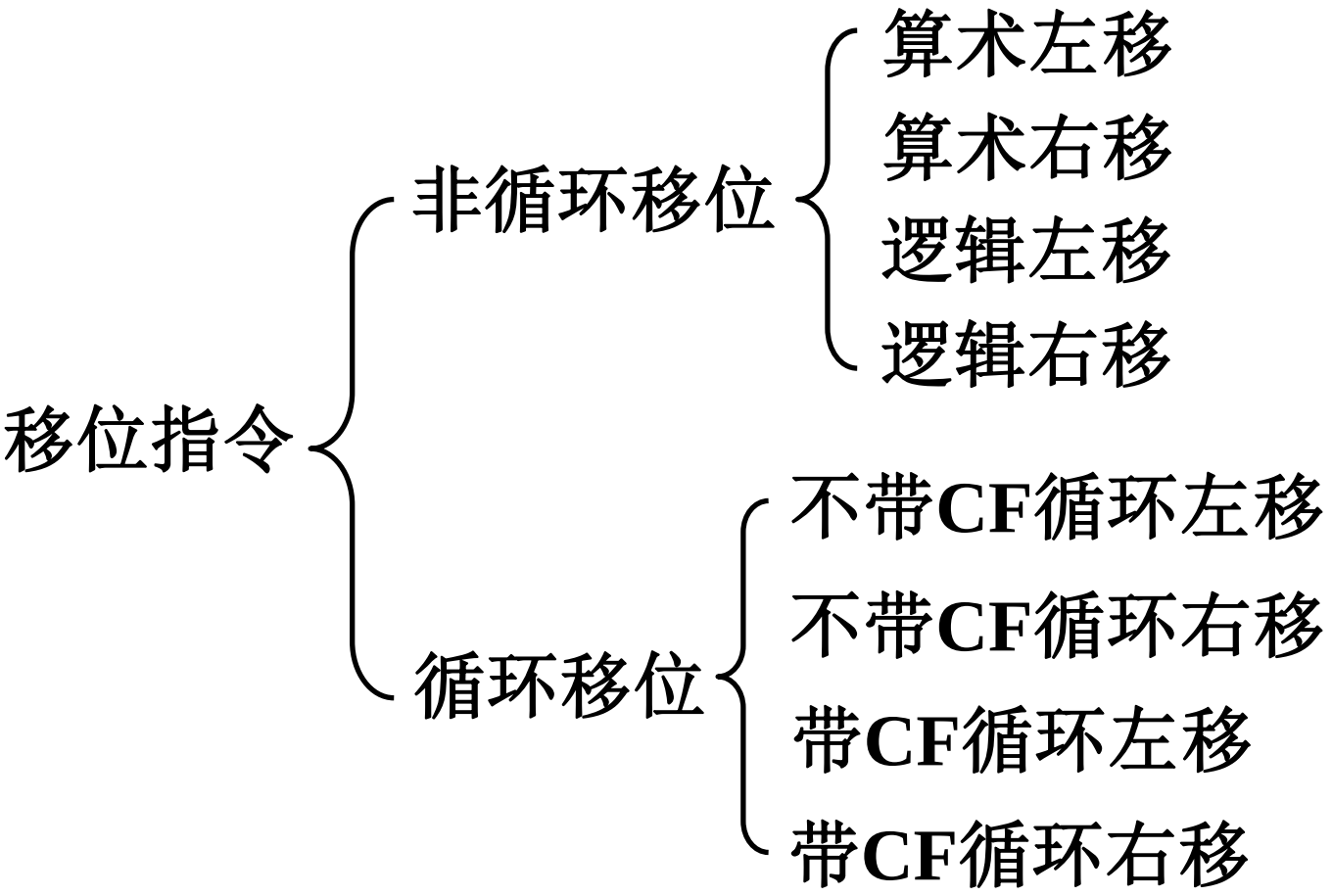
执行完测试指令后操作数均不会发生改变；

指令执行会影响6个状态标志位，并CF=OF=0

例，数据段中从4000H开始的单元中有32个8位的有符号数，要求统计出其中负数的个数并将统计结果保存到DX寄存器中。

```
        XOR     DX,   DX
        MOV     SI,   4000H
        MOV     CX,   32
AGAIN:  MOV     AL,   [SI]
        INC     SI
        TEST    AL,   80H
        JZ      NEXT
        INC     DX
NEXT:   DEC     CX
        JNZ     AGAIN
        HLT
```

2 移位指令



源操作数为移位次数（1或CL）；

目的操作数为通用寄存器或存储单元。

1）非循环移位指令

①算术左移SAL和逻辑左移SHL

- 算术左移指令格式：

  SAL OPRD，1

  SAL OPRD，CL

- 逻辑左移指令格式：

  SHL OPRD，1

  SHL OPRD，CL

- 算术左移指令SAL操作:

- 逻辑左移指令SHL操作:

若1次移位之后操作数的最高位与CF不相同，则OF=1，否则OF=0；OF=1对于SHL不表示溢出，对于SAL则表示溢出。

②算术右移SAR逻辑右移SHR

- 算术右移指令格式：

  SAR OPRD，1

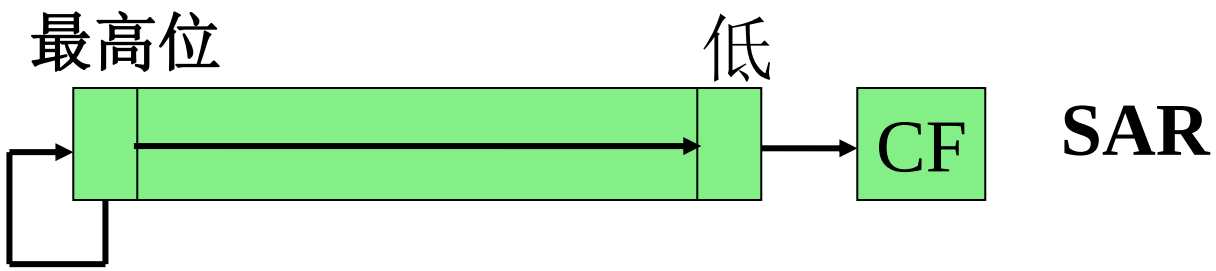
  SAR OPRD，CL

- 逻辑右移指令格式：

  SHR OPRD，1

  SHR OPRD，CL

- **算术右移指令SAR操作:**



- **逻辑右移指令SHR操作:**

指令SAL和SAR当移位次为n时，在结果不产生溢出的条件下，其作用分别相当于乘以2^n和除以2^n 。

例， 设AX中存放一个带符号数，若要实现(AX) ×5÷2，可由以下几条指令完成，不考虑溢出时。

```
MOV     DX，AX
SAL     AX，1
SAL     AX，1
ADD     AX，DX
SAR     AX，1
```

2）循环移位指令

①不带CF的循环移位指令

- 循环左移指令格式：

  ROL OPRD，1

  ROL OPRD，CL

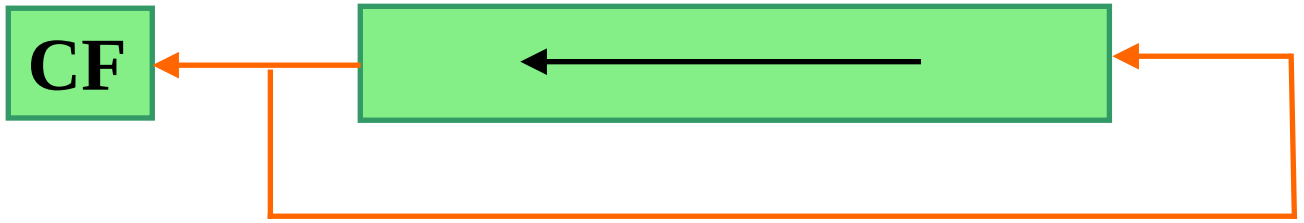
- 循环右移指令格式：

  ROR OPRD，1

  ROR OPRD，CL

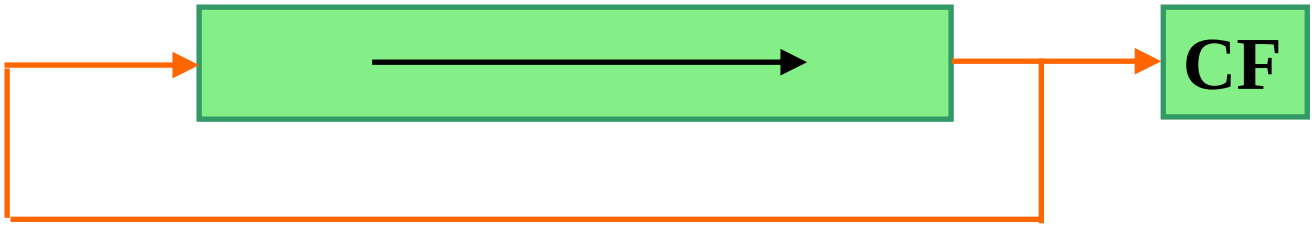
- 循环左移指令ROL操作：

ROL：



- 循环右移指令ROR操作：

ROR：



②带CF的循环移位指令

- 循环左移指令格式：

  RCL OPRD，1

  RCL OPRD，CL

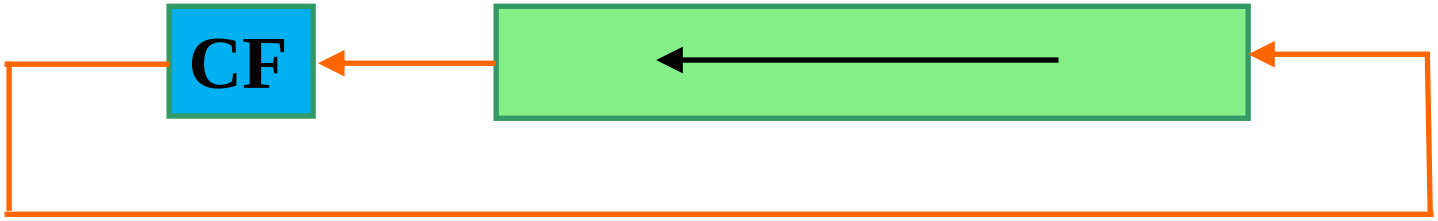
- 循环右移指令格式：

  RCR OPRD，1

  RCR OPRD，CL

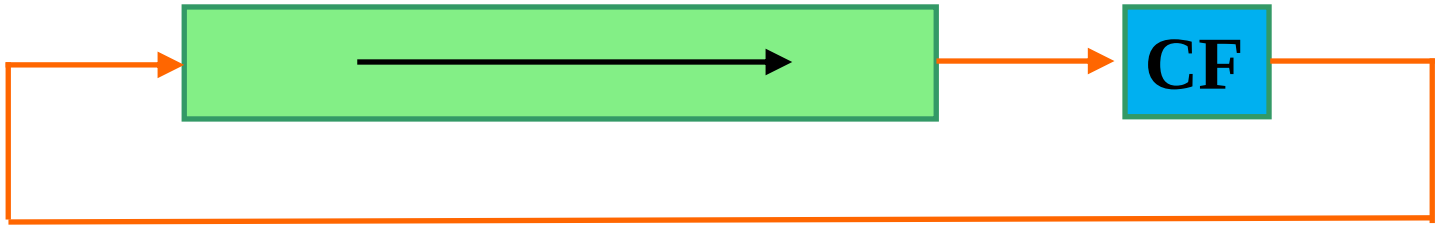
- 循环左移指令RCL操作:

RCL:



- 循环右移指令RCR操作:

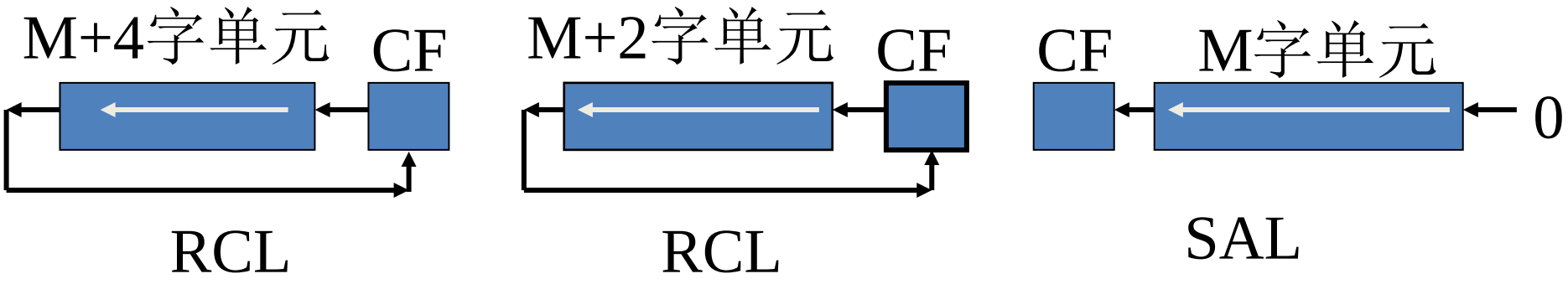
RCR:



例，下面程序段对从数据段存储单元M开始的三字数据执行左移一位。

```
SAL   M，1
RCL   M+2，1
RCL   M+4，1
```

移位指令共同特点：移出位都送给CF状态位。



例，将DX和AX两个寄存器组成的32位有符号数，其中AX为低位部分。先进行1位右移操作，然后再进行1位左移操作。

```
SAR    DX,    1
RCR    AX,    1
SAL    AX,    1
RCL    DX,    1
```

12.3.4 串操作指令

1 定义

存储器中地址连续的若干单元的字符或数据称为字符串或数据串。

串操作指令就是用来对串中每个字符或数据做同样操作的指令。

每执行一次操作后能够自动修改指针，再执行下一次操作。

2 共同特点

①源串默认为数据段，允许段重设，偏移地址用SI寄存器指定，即源串指针为DS:SI。

②目的串默认为附加段，不允许段重设，偏移地址用DI寄存器指定，即目的串指针为ES:DI。

③串长度值放在CX寄存器中。

④在对每个字节（或字）操作后，SI和DI寄存器的内容自动修改。修改方向与标志位DF有关，若DF=0，SI和DI按地址增量方向修改（对字节操作加1，对字操作加2），否则按地址减量方向修改。

⑤可以在串操作指令前使用重复前缀，在每次串操作后CX的内容自动减1，直至CX=0或不满足指定的条件为止。

3 重复前缀

①REP：无条件重复前缀，重复执行规定的操作，直至CX=0。

②REPE/REPZ：相等/结果为0时重复，即ZF=1,且CX≠0。

③REPNE/REPNZ：不相等/结果不为0时重复，即ZF=0,且CX≠0。

④重复前缀操作不影响标志位。

⑤先执行串操作指令，串操作指令可能会影响标志位。然后CX-1，并判断条件。

4 串操作指令

串操作指令是8086指令系统中唯一能直接处理源和目标操作数都在存储单元的指令。

1）串传送指令 MOVS OPRD1，OPRD2

MOVSB：一次完成一个字节的传送

MOVSW：一次完成一个字的传送

实现的操作：DS:[SI]→ES:[DI]

SI+n→SI, DI+n→DI

常与无条件重复前缀REP联合使用，不影响标志位

例，将数据段2000H:1200H地址开始的100个字节传送到6000H:0000H开始的内存单元。

```
MOV     AX,     2000H
MOV     DS,     AX
MOV     AX,     6000H
MOV     ES,     AX
MOV     SI,     1200H
MOV     DI,     0
MOV     CX,     100
CLD
REP   MOVSB
HLT
```

2）串比较指令 CMPS OPRD1，OPRD2

CMPSB：按字节进行比较

CMPSW：按字进行比较

操作：DS:[SI]- ES:[DI]，不改变操作数

SI+n→SI, DI+n→DI

常与重复前缀REPE/REPZ或REPNE/REPNZ联合使用，用来检查两个字符串是否相等或不相等。

串比较指令影响标志位，CX是否为0不影响标志位

REPE/REPZ，ZF=1，且CX≠0重复比较

REPNE/REPNZ，ZF=0，且CX≠0重复比较

例，比较两个字符串是否相同，并找出其中第一个不相同字符的地址，将该地址送BX，不相同的源字符送AL，字符串长度均为200字节，M1为源串起始偏移地址，M2为目的串起始偏移地址。

```
        MOV   SI,    M1
        MOV   DI,    M2
        MOV   CX,    200
        CLD
        REPZ  CMPSB
        JZ    STOP
        DEC   SI
        MOV   BX,    SI
        MOV   AL,    [SI]
STOP:   HLT
```

3）串扫描指令 SCAS OPRD

SCASB：按AL内容对目的串进行扫描

SCASW：按AX内容对目的串进行扫描

操作：AL/AX- ES:[DI]

DI+n→DI

累加器AL或AX作为源操作数，ES:[DI]作为目的串操作数

不改变操作数及SI寄存器，自动改变DI寄存器的内容，同时会影响标志位。

例，在ES段中从2000H单元存放了10个字符，编程计数这串字符中有多少个“A”，并将计数值存入寄存器BX中。

```
        MOV  DI,    2000H
        MOV  BX,    0
        MOV  CX,    10
        MOV  AL,    'A'
        CLD

COUNT:  REPNZ  SCASB      ;  ZF=0, CX≠0
        JNZ    STOP       ;  ZF=0, CX=0
        INC    BX         ;  ZF=1,CX?
        CMP    CX,    0
        JNZ    COUNT      ;  ZF=1, CX≠0
                          ;  ZF=1, CX=0
STOP:   HLT
```

4）串装入指令　LODS　OPRD

LODSB：将源串按字节装入AL

LODSW：将源串按字装入AX

操作：DS:[SI]→AL/AX

SI+n→SI

DS:[SI]作为源串操作数,累加器AL或AX作为目的操作数

自动改变SI寄存器的内容，不影响标志位，同时一般不带重复前缀指令。

LODSB等效于：

```
MOV   AL,   [SI]
INC   SI
```

LODSW等效于：

```
MOV   AX,   [SI]
INC   SI
INC   SI
```

5）串存储指令　STOS OPRD

STOSB：将AL内容存入目的串

STOSW：将AX内容存入目的串

操作：AL/AX →ES:[DI]

DI+n→DI

累加器AL或AX作为源操作数，ES:[DI]作为目的串操作数

自动改变DI寄存器的内容，不影响标志位，利用重复前缀REP可对连续存储单元存入相同的值。

STOSB等效于：

```
MOV    ES:[DI],      AL
INC    DI
```

STOSW等效于：

```
MOV    ES:[DI],      AX
INC    DI
INC    DI
```

例，用串存储指令实现对6000H:1200H开始的100个字单元内容清零。

```
MOV     AX,  6000H
MOV     ES,  AX
MOV     DI,  1200H
MOV     CX,  100
CLD
MOV     AX,  0
REP     STOSW
HLT
```