

## 教学提示

在正确理解每条指令的功能基础上，可以阅读和编写有实际意义的程序段

多多阅读程序段



- 位操作类指令以二进制位为基本单位进行数据的操作；这是一类常用的指令，都应该特别掌握
- 注意这些指令对标志位的影响

1、逻辑运算指令

**AND OR XOR NOT TEST**

2、移位指令

**SHL SHR SAR**

3、循环移位指令

**ROL ROR RCL RCR**

- 对两个操作数执行逻辑与运算，结果送到目的操作数

AND reg, imm/reg/mem ;  $\text{reg} \leftarrow \text{reg} \wedge \text{imm/reg/mem}$

AND mem, imm/reg ;  $\text{mem} \leftarrow \text{mem} \wedge \text{imm/reg}$

只有相“与”的两位都是1，结果才是1；否则，“与”的结果为0

- AND指令设置CF = OF = 0，根据结果设置SF、ZF和PF状态，而对AF未定义

- ▶ 对两个操作数执行逻辑或运算，结果送到目的操作数

OR reg, imm/reg/mem ;  $\text{reg} \leftarrow \text{reg} \vee \text{imm/reg/mem}$

OR mem, imm/reg ;  $\text{mem} \leftarrow \text{mem} \vee \text{imm/reg}$

只要相“或”的两位有一位是1，结果就是1；否则，结果为0

- ▶ OR指令设置CF = OF = 0，根据结果设置SF、ZF和PF状态，而对AF未定义

- 对两个操作数执行逻辑异或运算，结果送到目的操作数

`XOR reg, imm/reg/mem ; reg ← reg ⊕ imm/reg/mem`

`XOR mem, imm/reg ; mem ← mem ⊕ imm/reg`

只有相“异或”的两位不相同，结果才是1；否则，结果为0

- XOR指令设置CF = OF = 0，根据结果设置SF、ZF和PF状态，而对AF未定义

- 对一个操作数执行逻辑非运算

NOT reg/mem ; reg/mem  $\leftarrow$   $\sim$  reg/mem

按位取反，原来是“0”的位变为“1”；原来是“1”的位变为“0”

- NOT指令是一个单操作数指令
- NOT指令不影响标志位

### 例3.28：逻辑运算

`mov al, 45h` ; 逻辑与 `al=01h`  
`and al, 31h` ; `CF=0F=0, SF=0、ZF=0、PF=0`

`mov al, 45h` ; 逻辑或 `al=75h`  
`or al, 31h` ; `CF=0F=0, SF=0、ZF=0、PF=0`

`mov al, 45h` ; 逻辑异或 `al=74h`  
`xor al, 31h` ; `CF=0F=0, SF=0、ZF=0、PF=1`

`mov al, 45h` ; 逻辑非 `al=0bah`  
`not al` ; 标志不变





### 例3.29：逻辑指令应用

； AND指令可用于复位某些位（同0相与），不影响其他位：将BL中D3和D0位清0，其他位不变

**and** bl, 11110110B

； OR指令可用于置位某些位（同1相或），不影响其他位：将BL中D3和D0位置1，其他位不变

**or** bl, 00001001B

； XOR指令可用于求反某些位（同1相异或），不影响其他位：将BL中D3和D0位求反，其他不变

**xor** bl, 00001001B



- 对两个操作数执行逻辑与运算，结果不回送到目的操作数

TEST reg, imm/reg/mem ;  $\text{reg} \wedge \text{imm/reg/mem}$


TEST mem, imm/reg ;  $\text{mem} \wedge \text{imm/reg}$

只有相“与”的两位都是1，结果才是1；否则，“与”的结果为0

- AND指令设置CF = OF = 0，根据结果设置SF、ZF和PF状态，而对AF未定义

**例3.30：测试为0或1**

```
test al, 01h    ; 测试AL的最低位D0  
jnz there      ; 标志ZF=0, 即D0=1  
                ; 则程序转移到there  
...            ; 否则ZF=1, 即D0=0, 顺序执行  
there: ...
```

 **TEST**指令通常用于检测一些条件是否满足，但又不希望改变原操作数的情况

- 将操作数移动一位或多位，分成逻辑移位和算术移位，分别具有左移或右移操作

SHL reg/mem, 1/CL

演示

SHR reg/mem, 1/CL

SAL与SHL相同

SAL reg/mem, 1/CL

SAR reg/mem, 1/CL

图示

； 算术右移，最低位进入CF，最高位不变

- 移位指令的第一个操作数是指定的被移位的操作数，可以是寄存器或存储单元
- 后一个操作数表示移位位数，该操作数为1，表示移动一位；当移位位数大于1时，则用CL寄存器值表示，该操作数表达为CL

- 按照移入的位设置进位标志**CF**
- 根据移位后的结果影响**SF**、**ZF**、**PF**
- 对**AF**没有定义
- 如果进行一位移动，则按照操作数的最高符号位是否改变，相应设置溢出标志**OF**：如果移位前的操作数最高位与移位后操作数的最高位不同（有变化），则**OF = 1**；否则**OF = 0**。当移位次数大于1时，**OF**不确定

### 例3.31：移位指令

```
mov cl, 4
mov al, 0f0h      ; al=f0h
shl al, 1         ; al=e0h
; CF=1, SF=1、ZF=0、PF=0, OF=0
shr al, 1         ; al=70h
; CF=0, SF=0、ZF=0、PF=0、OF=1
sar al, 1         ; al=38h
; CF=0, SF=0、ZF=0、PF=0、OF=0
sar al, cl        ; al=03h
; CF=1, SF=0、ZF=0、PF=1
```



- 将操作数从一端移出的位返回到另一端形成循环，分成不带进位和带进位，分别具有左移或右移操作

图示

ROL reg/mem, 1/CL ; 不带进位循环左移

ROR reg/mem, 1/CL ; 不带进位循环右移

RCL reg/mem, 1/CL ; 带进位循环左移

RCR reg/mem, 1/CL ; 带进位循环右移

图示



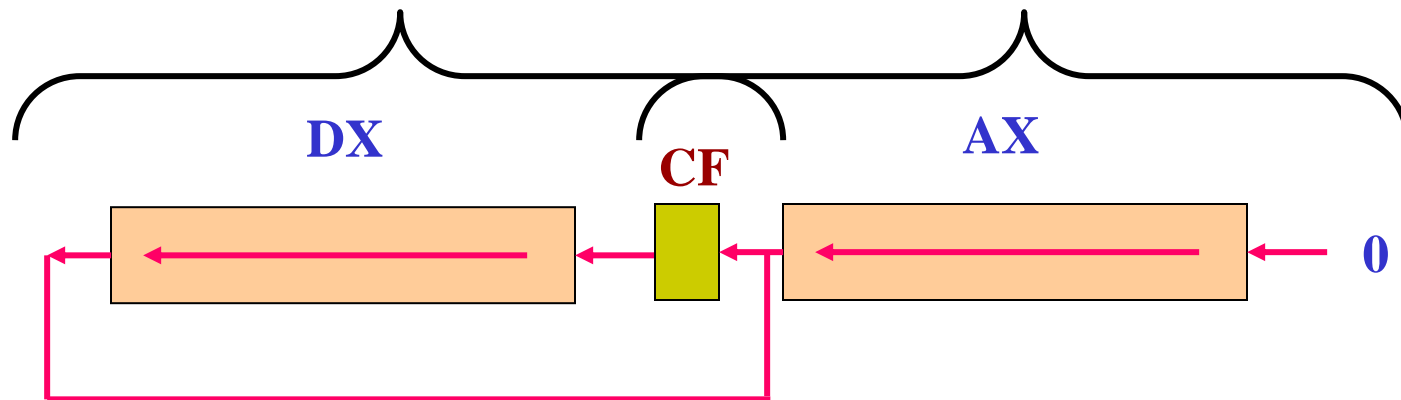
- 按照指令功能设置进位标志**CF**
- 不影响**SF**、**ZF**、**PF**、**AF**
- 如果进行一位移动，则按照操作数的最高符号位是否改变，相应设置溢出标志**OF**：如果移位前的操作数最高位与移位后操作数的最高位不同（有变化），则**OF = 1**；否则**OF = 0**。当移位次数大于1时，**OF**不确定

### 例3.33：32位数移位

； 将DX. AX中32位数值左移一位

shl ax, 1

rcl dx, 1



### 例3.34：位传送

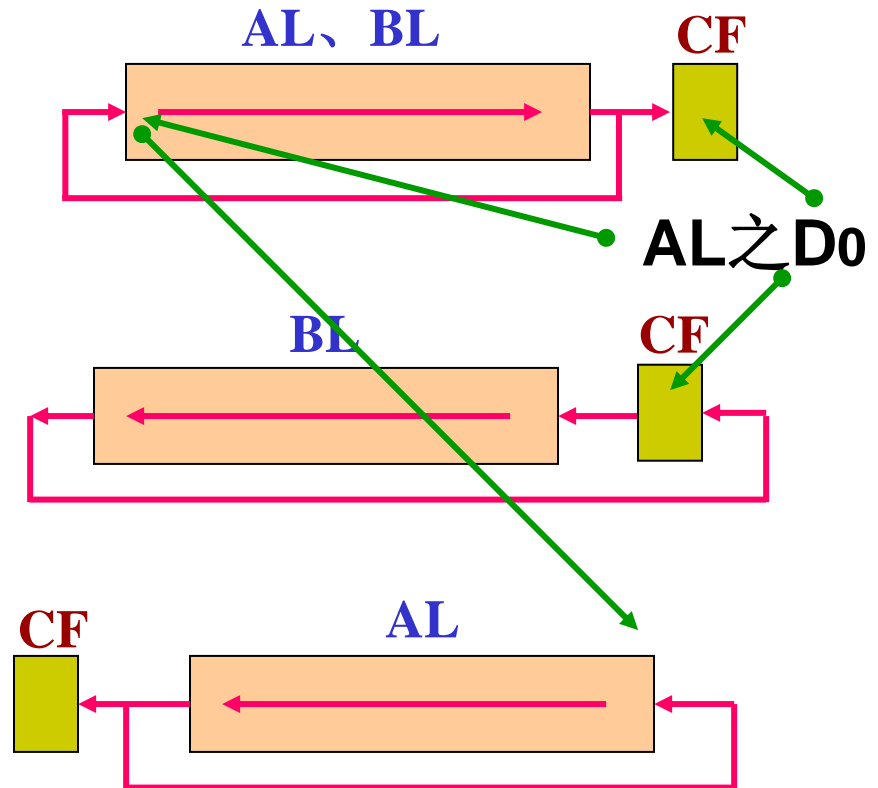
；把AL最低位送BL最低位，保持AL不变

`ror bl, 1`

`ror al, 1`

`rcl bl, 1`

`rol al, 1`



### 例3.35: BCD码合并

； AH. AL分别存放着非压缩BCD码的两位

； 将它们合并成为一个压缩BCD码存AL

`and ax, 0f0fh` ； 保证高4位为0

`mov cl, 4`

`rol ah, cl` ； 也可以用`shl ah, cl`

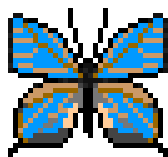
`add al, ah` ； 也可以用`or al, ah`

➤ 串操作指令是8086指令系统中比较独特的一类指令，采用比较特殊的数据串寻址方式，在操作主存连续区域的数据时，特别好用、因而常用

重点掌握: **MOVS STOS LODS**

**CMPS SCAS REP**

一般了解: **REPZ/REPE REPNZ/REPNE**



- 串操作指令的操作数是主存中连续存放的数据串（**String**）——即在连续的主存区域中，字节或字的序列
- 串操作指令的操作对象是以字（**W**）为单位的字串，或是以字节（**B**）为单位的字节串

- 源操作数用寄存器**SI**寻址，默认在数据段**DS**中，但允许段超越：**DS:[SI]**
- 目的操作数用寄存器**DI**寻址，默认在附加段**ES**中，不允许段超越：**ES:[DI]**
- 每执行一次串操作指令，**SI**和**DI**将自动修改：
  - $\pm 1$ （对于字节串）或 $\pm 2$ （对于字串）
  - 执行指令**CLD**指令后，**DF = 0**，地址指针增1或2
  - 执行指令**STD**指令后，**DF = 1**，地址指针减1或2



- 把字节或字操作数从主存的源地址传送至目的地址

MOVSB

; 字节串传送:  $ES:[DI] \leftarrow DS:[SI]$   
;  $SI \leftarrow SI \pm 1, DI \leftarrow DI \pm 1$

演示

MOVSW

; 字串传送:  $ES:[DI] \leftarrow DS:[SI]$   
;  $SI \leftarrow SI \pm 2, DI \leftarrow DI \pm 2$

演示

offset是汇编操作符，  
求出变量的偏移地址

### 例3.36：字节串传送

```
mov si, offset source
mov di, offset destination
mov cx, 100      ; cx ← 传送次数
cld              ; 置DF=0，地址增加
again: movsb     ; 传送一个字节
dec cx          ; 传送次数减1
jnz again
; 判断传送次数cx是否为0
; 不为0，则到again位置执行指令
; 否则，结束
```

演示



### 例3.36：字串传送

```
mov si, offset source
mov di, offset destination
mov cx, 50      ; cx ← 传送次数
cld             ; 置DF=0, 地址增加
again: movsw    ; 传送一个字
dec cx         ; 传送次数减1
jnz again      ; 判断传送次数cx是否为0
               ; 不为0, 则到again位置执行指令
               ; 否则, 结束
```



➤ 把AL或AX数据传送至目的地址

STOSB

; 字节串存储:  $ES:[DI] \leftarrow AL$   
;  $DI \leftarrow DI \pm 1$

STOSW

; 字串存储:  $ES:[DI] \leftarrow AX$   
;  $DI \leftarrow DI \pm 2$

## 例3.37：串存储

```
mov ax, 0  
mov di, 0  
mov cx, 8000h
```

```
; cx ← 传送次数 (32 × 1024)
```

```
cld ; DF=0, 地址增加
```

```
again: stosw ; 传送一个字
```

```
dec cx ; 传送次数减1
```

```
jnz again ; 传送次数cx是否为0
```

*DI为偶数即可*

● 可将CLD改为STD吗？如何改用STOSB？

● 可不用给DI赋值吗？



- 把指定主存单元的数据传送给AL或AX

LODSB

; 字节串读取:  $AL \leftarrow DS:[SI]$   
;  $SI \leftarrow SI \pm 1$

LODSW

; 字串读取:  $AX \leftarrow DS:[SI]$   
;  $SI \leftarrow SI \pm 2$

### 例3.38：串读取—1

```
mov si, offset block  
mov di, offset dplus  
mov bx, offset dminus  
mov ax, ds  
mov es, ax
```

； 数据都在一个段中， 所以设置es=ds

```
mov cx, count      ； cx←字节数  
cld
```





### 例3.38：串读取—2

```
go_on: lodsb          ; 从block取出一个数据
      test al, 80h
      ; 检测符号位，判断是正是负
      jnz minus
      ; 符号位为1，是负数，转向minus
      stosb
      ; 符号位为0，是正数，存入dplus
      jmp again
      ; 程序转移到again处继续执行
```

### 例3.38：串读取—3

```
minus: xchg bx, di ; bx存放dminus的起  
          始地址  
      stosb        ; 把负数存入dminus  
      xchg bx, di  
again: dec cx      ; 字节数减1  
      jnz go_on    ; 完成正负数据分离
```



- 将主存中的源操作数减去至目的操作数，以便设置标志，进而比较两操作数之间的关系

**CMPSB**

； 字节串比较：DS:[SI]—ES:[DI]  
； SI←SI±1， DI←DI±1

**CMPSW**

； 字串比较：DS:[SI]—ES:[DI]  
； SI←SI±2， DI←DI±2

### 例3.39a: 比较字符串

```
mov si, offset string1
mov di, offset string2
mov cx, count
cld
again: cmpsb          ; 比较两个字符
      jnz unmat       ; 有不同字符, 转移
      dec cx
      jnz again       ; 进行下一个字符比较
      mov al, 0        ; 字符串相等, 设置00h
      jmp output      ; 转向output
unmat: mov al, 0ffh    ; 设置ffh
output: mov result, al ; 输出结果标记
```

- 将 **AL/AX** 减去至目的操作数，以便设置标志，进而比较 **AL/AX** 与操作数之间的关系

SCASB

; 字节串扫描:  $AL - ES:[DI]$   
;  $DI \leftarrow DI \pm 1$

SCASW

; 字串扫描:  $AX - ES:[DI]$   
;  $DI \leftarrow DI \pm 2$

### 例3.40a: 查找字符串

```
mov di, offset string
mov al, 20h
mov cx, count
cld
again: scasb      ; 搜索
      jz found   ; 为0 (ZF=1), 发现空格
      dec cx     ; 不是空格
      jnz again  ; 搜索下一个字符
      ...       ; 不含空格, 则继续执行
found: ...
```

- 串操作指令执行一次，仅对数据串中的一个字节或字量进行操作。但是串操作指令前，都可以加一个重复前缀，实现串操作的重复执行。重复次数隐含在**CX**寄存器中
- 重复前缀分**2**类，**3**条指令：
  - 配合不影响标志的**MOVS**、**STOS**（和**LODS**）指令的**REP**前缀
  - 配合影响标志的**CMPS**和**SCAS**指令的**REPZ**和**REPNZ**前缀



**REP** ; 每执行一次串指令, CX减1  
; 直到CX=0, 重复执行结束

- **REP**前缀可以理解为: 当数据串没有结束 (**CX≠0**), 则继续传送
- 例3.36和例3.37中, 程序段的最后3条指令, 可以分别替换为:

**REP MOVSB** 和 **REP STOSW**

**REPZ** ; 每执行一次串指令, CX减1  
; 并判断ZF是否为0,  
; 只要CX=0或ZF=0, 重复执行结束

- **REPZ/REPE**前缀可以理解为: 当数据串没有结束 (**CX≠0**), 并且串相等 (**ZF=1**), 则继续比较

**REP NZ** ; 每执行一次串指令, CX减1  
; 并判断ZF是否为1,  
; 只要CX=0或ZF=1, 重复执行结束

➤ **REP NZ/REP NE** 前缀可以理解为:  
当数据串没有结束 (**CX≠0**), 并且串不相等 (**ZF=0**), 则继续比较

### 例3.39b：比较字符串

```
mov si, offset string1
mov di, offset string2
mov cx, count
cld
repz cmpsb      ; 重复比较两个字符
jnz unmat       ; 字符串不等，转移
mov al, 0        ; 字符串相等，设置00h
jmp output      ; 转向output
unmat: mov al, 0ffh ; 设置ffh
output: mov result, al ; 输出结果标记
```

解释

### 例3.40b: 查找字符串

```
mov di, offset string
```

```
mov al, 20h
```

```
mov cx, count
```

```
cld
```

```
repnz scasb ; 搜索
```

```
jz found ; 为0 (ZF=1) , 发现空格
```

```
... ; 不含空格, 则继续执行
```

```
found: ...
```

## 第3章 教学要求 (4)

1. 熟悉串操作寻址特点
2. 掌握基本指令：**AND / OR / XOR / NOT / TEST、SHL / SHR / SAR、ROL / ROR / RCL / RCR、MOVS / LODS / STOS、REP**
3. 熟悉特色指令：**CMPS / SCAS**
4. 了解不常使用的指令：**REPZ / REPNZ**
5. 习题3 (p72)      **2.23    2.25    2.26**





课间休息

