

第3章 8086/8088指令系统4



表 4-3 逻辑运算类指令

类别	指令名称	指令格式
逻辑运算	非(字节/字)	NOT 目标
	与(字节/字)	AND 目标, 源
	或(字节/字)	OR 目标, 源
	异或(字节/字)	XOR 目标, 源
	测试(字节/字)	TEST 目标, 源

(1) 逻辑“与” AND

对两个操作数进行按位逻辑“与”操作。

格式: **AND dest, src**

用途: 保留操作数的某几位, 清零其他位。

例1: 保留AL中低4位, 高4位清0。

AND AL, 0FH

例2: AL中有字符'a'~'z', 将其转换成大写。

AND AL, 01011111B

例3: 测试AL的bit7,bit5,bit2是否都是1。

AND AL, 10100100B

CMP AL, 10100100B

JZ YES ; if match, go to

YES

... ; if not match

...

YES: ... ; goes here if all '1'

(2) 逻辑“或” OR

对两个操作数进行按位逻辑“或”操作。

格式: **OR dest, src**

用途: 对操作数的某几位置1;
对两操作数进行组合。

例1: 把AL中的非压缩BCD码变成相应十进制数的ASCII码。

OR AL, 30H

例2: 把AH和AL中的非压缩BCD码组合成压缩的BCD码, 放到AL中。

```
MOV CL, 4
```

```
SHL AH, CL
```

```
OR AL, AH
```

例3: 把AL的第5位置为1

```
OR AL, 00100000B
```

(3) 逻辑“非”(取反) NOT

对操作数进行按位逻辑“非”操作。

格式: **NOT mem/reg**

例: **NOT CX**

NOT BYTE PTR[DI]

(4) 逻辑“异或” XOR

对两个操作数按位进行”异或”操作。

格式: **XOR dest, src**

用途: 对reg清零(自身异或)

把reg/mem的某几位变反(与'1'异或)

例1: 把AX寄存器清零。 例2: 把DH的bit4, 3变反

① MOV AX, 0

XOR DH, 18H

② XOR AX, AX

③ AND AX, 0

④ SUB AX, AX

(5) 测试指令TEST

操作与AND指令类似,但不将”与”的结果送回,只影响标志位。

AND与TEST间的区别类似于SUB与CMP间的区别

作用: TEST指令常用于位测试,与条件转移指令一起用。

例: 测试AL的内容是否为负数。

```
TEST  AL, 80H    ; 检查AL中D7=1?
```

```
JNZ   MINUS     ; 是1(负数), 转MINUS
```

```
... ..        ; 否则(正数)不转移
```

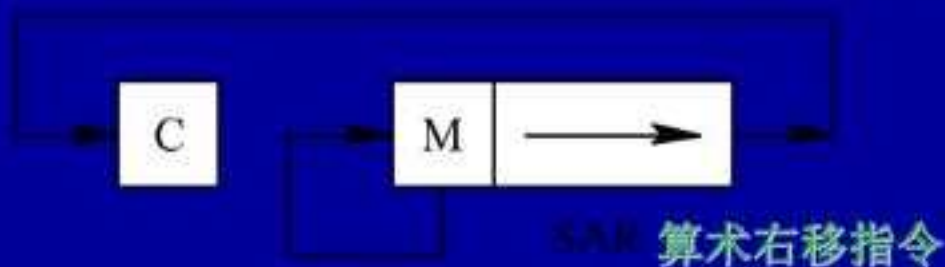
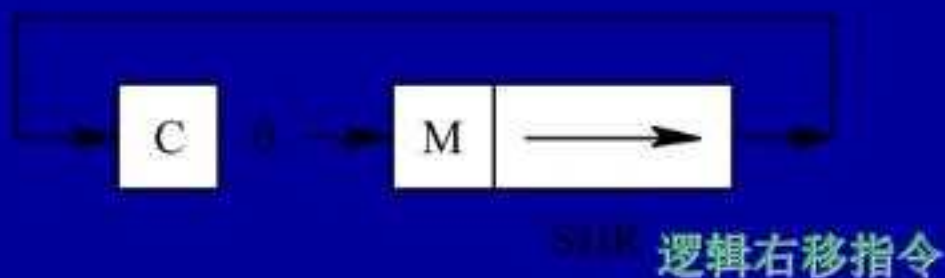
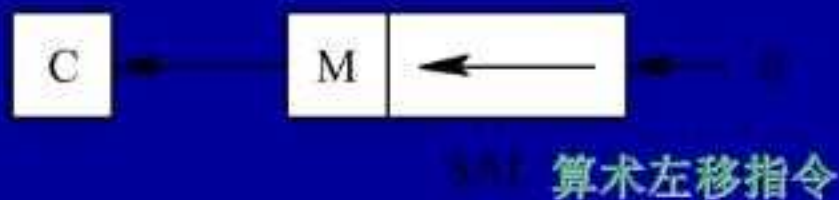
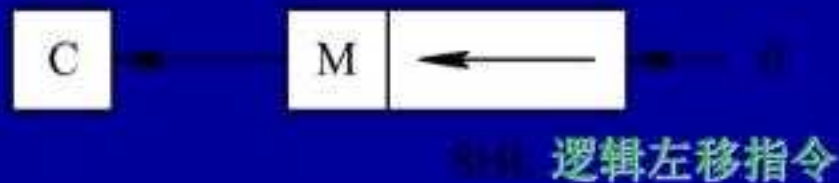
```
MINUS: ... ..
```

```
... ..
```

2. 移位指令

表4-4 移位操作类指令

类别	指令名称	指令格式
移位	逻辑左移(字节/字)	SHL 目标, 计数值
	算术左移(字节/字)	SAL 目标, 计数值
	逻辑右移(字节/字)	SHR 目标, 计数值
	算术右移(字节/字)	SAR 目标, 计数值
循环移位	循环左移(字节/字)	ROL 目标, 计数值
	循环右移(字节/字)	ROR 目标, 计数值
	带进位循环左移(字节/字)	RCL 目标, 计数值
	带进位循环右移(字节/字)	RCR 目标, 计数值



$$CF = (CF \oplus (M \oplus (M \gg 1)))$$

移位指令功能示意

2. 移位指令

(1) 非循环移位指令

算术左移指令 SAL (Shift Arithmetic Left)

算术右移指令 SAR (Shift Arithmetic Right)

逻辑左移指令 SHL (Shift Left)

逻辑右移指令 SHR (Shift Right)

这4条指令的格式相同, 以SAL为例:

SAL	mem/reg,	CL	;移位位数>1时
		1	;移位位数=1时

移位指令执行的操作如下图所示：



(a)算术/逻辑左移 SAL/SHL



(b)算术右移 SAR

(c)逻辑右移 SHR

非循环移位指令功能示意图

- ▶ 算术移位——把操作数看做有符号数；
- ▶ 逻辑移位——把操作数看做无符号数。
- ▶ 移位位数：一般放在CL寄存器中，但如果只移1位，也可以直接写在指令中。

例如：

```
MOV CL, 4
```

```
SHR AL, CL ; AL中的内容右移4位
```

- ▶ 对Flags影响：影响CF, PF, SF, ZF, OF标志。
- ▶ 移位效果：结果未溢出时：

左移1位 \equiv 操作数 $\times 2$

右移1位 \equiv 操作数 $\div 2$

例：把AL中的数 $x \times 10$

因为 $10=8+2=2^3+2^1$ ，所以可用移位实现乘10操作。程序如下：

```
SAL  AL,1    ; 2x
```

```
MOV  AH,AL
```

```
SAL  AL,1    ; 4x
```

```
SAL  AL,1    ; 8x
```

```
ADD  AL,AH   ; 8x+2x = 10x
```

(2) 循环移位指令

不含CF的(小)循环左移指令 ROL

不含CF的(小)循环右移指令 ROR

含CF的(大)循环左移指令 RCL

含CF的(大)循环右移指令 RCR

➤格式同非循环移位指令。

➤移位位数一般放在CL寄存器中

但如果只移1位,也可直接写在指令中。

➤对Flags的影响:只影响标志位CF和OF。

这4条指令的功能如下图示:



循环移位指令功能示意图

- 用移位操作代替乘法可提高运算速度

例：前例中计算 $x \times 10$ 。

(1) 采用乘法指令：

```
MOV BL, 10
```

```
MUL BL
```

共需70~77个T周期。

(2) 采用移位和加法指令：

```
SAL AL, 1 ; 2T
```

```
MOV AH, AL ; 2T
```

```
SAL AL, 1 ; 2T
```

```
SAL AL, 1 ; 2T
```

```
ADD AL, AH ; 3T
```

只需11个T周期, 仅相当于乘法的1/7。

循环移位举例：

例1：将AL的高4位与低4位互换。

```
MOV CL,4
```

```
ROL AL,CL
```

例2：将1A00H内存单元中的双字(32位)
循环左移1位。

```
CMP [1A00H],8000H
```

```
CMC
```

```
RCL WORD PTR[1A02H],1
```

```
RCL WORD PTR[1A00H],1
```

例：设在1000H开始存有四个**压缩的BCD码**12、34、56、78。要求把它们转换为**ASCII码**存放在3000H开始的单元中。

假定DS、ES都已设置为数据段的段基址。

程序见下页。

SI= 1000H

12
34
56
78

BX=4

DI= 3000H

32H
31H
34H
33H
36H
35H
38H
37H

```

MOV SI, 1000H ; SI ← BCD首址
MOV DI, 3000H ; DI ← ASCII首址
MOV BX, 4 ; 置计数器初值
BBB: MOV AL, [SI] ; AL ← BCD码, 第一次取12H
AND AL, 0FH ; 屏蔽高4位 → 02H
OR AL, 30H ; 转换为ASCII码 → 32H
STOSB ; 保存结果, 等价MOV[DI], AL INC DI
LODSB ; MOV AL, [SI] INC SI
MOV CL, 4
SHR AL, CL ; 逻辑右移4位
OR AL, 30H ; 得到高4位ASCII码
STOSB ; 保存结果, 等价MOV[DI], AL INC DI
DEC BX ; (BX) ← (BX) - 1
JNZ BBB ; (BX) ≠ 0, 则继续循环

```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/498115027073006047>