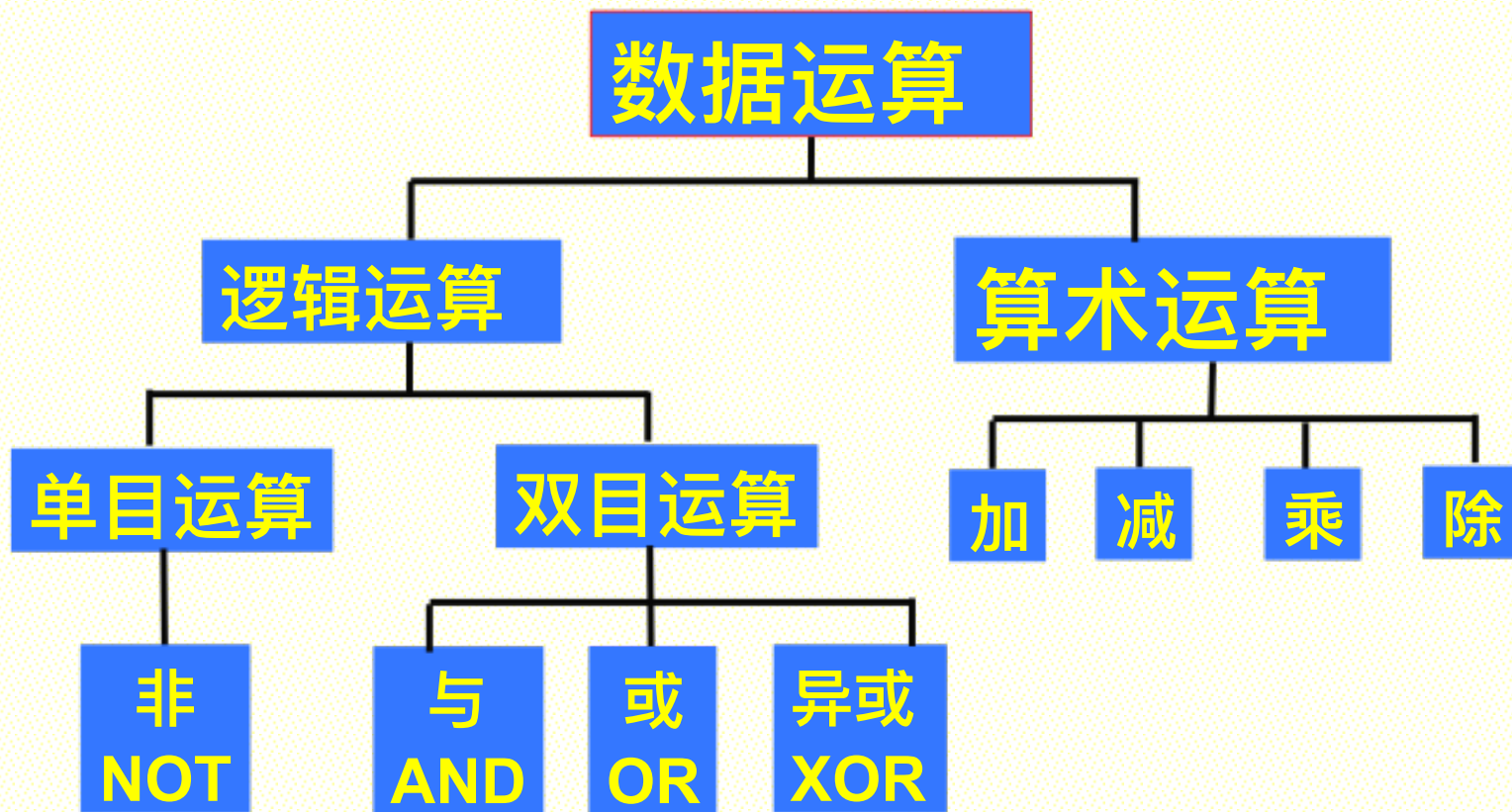


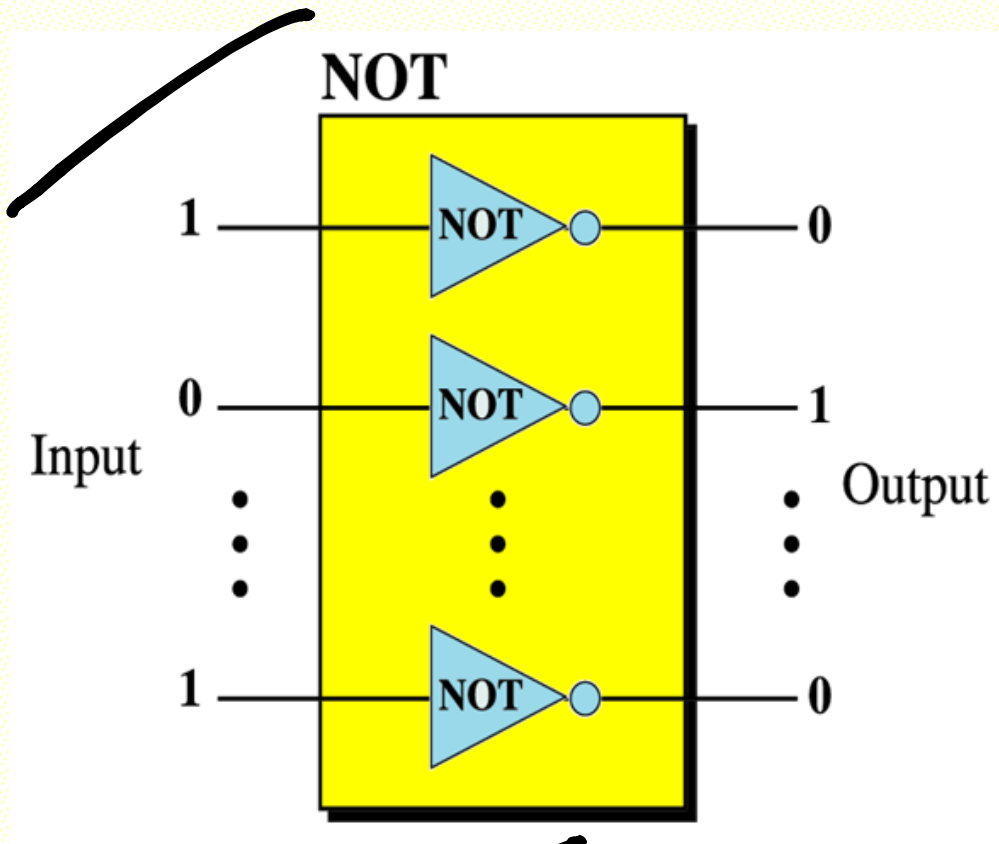
# 第四章 数据运算


- 逻辑运算：
  - 与、或、非、异或
  - 逻辑运算的用途
- 移位运算
- 补码的算术运算：加法和减法
- 浮点数的加法和减法运算

# 数据运算



# 非 (NOT) 操作



NOT 

x	NOTx
0	1
1	0

取反

# 非 (NOT) 操作实例

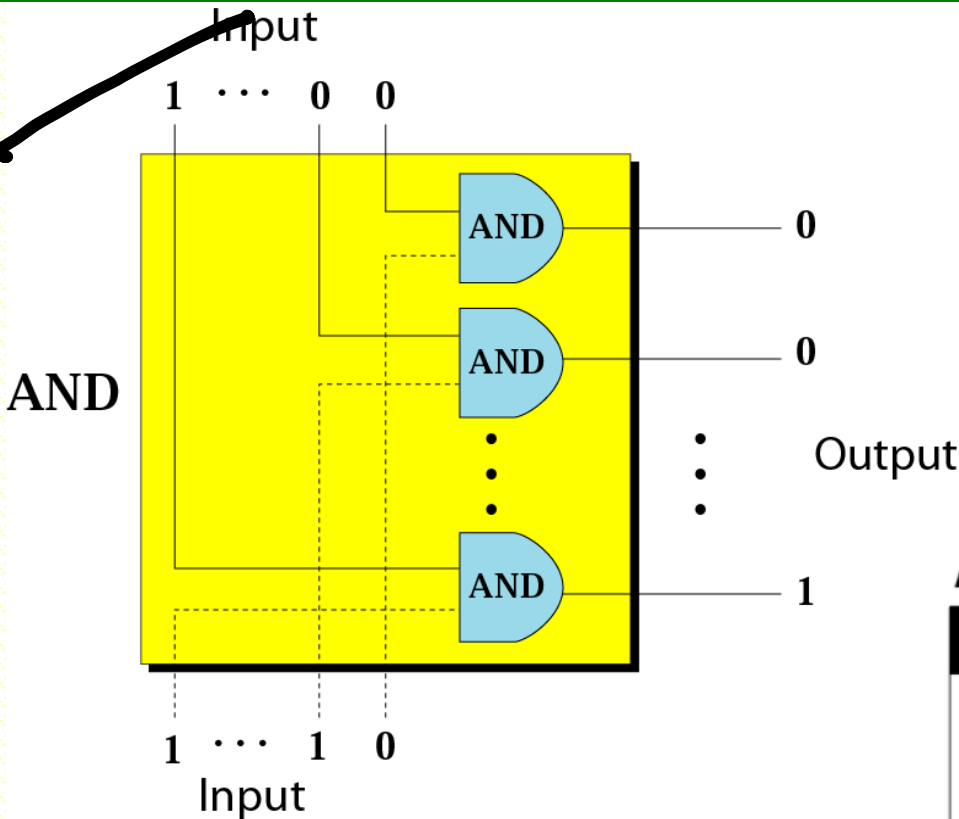
将二进制数 10011000 进行非运算

解：

原数：            1 0 0 1   1 0 0 0        *NOT*

结果：            0 1 1 0   0 1 1 1

# 与 (AND) 操作



全1时结果  
为1

其它结果为0

AND

x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1

## 与 (AND) 操作实例

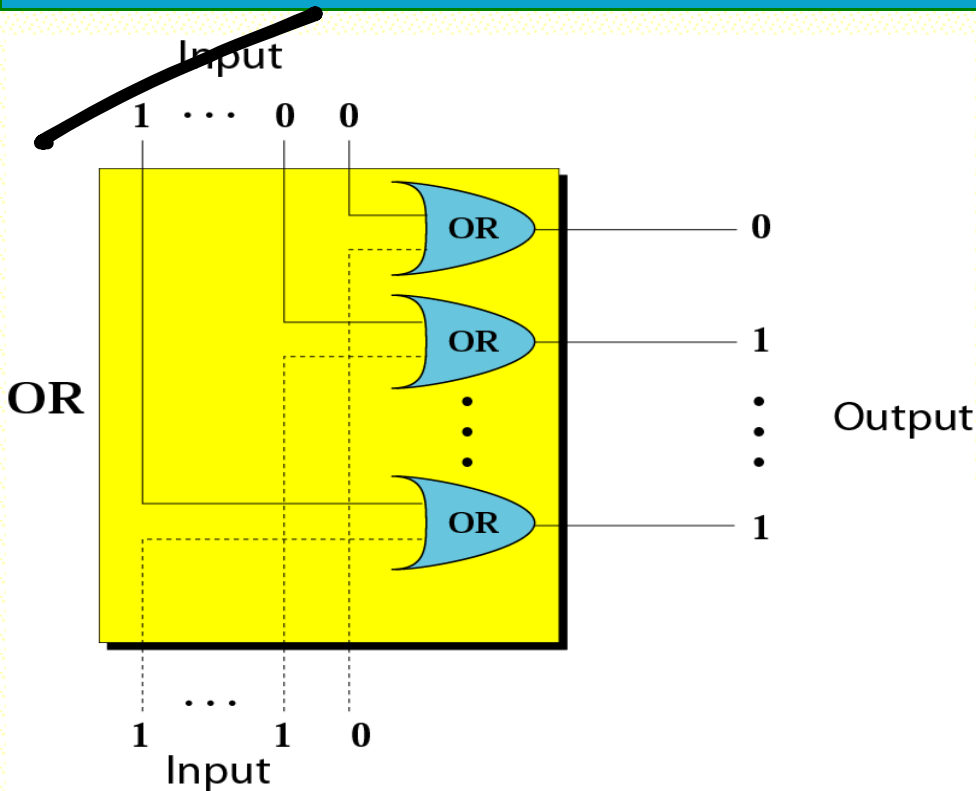
使用 AND 操作对以下两个二进制数运算

10011000 和 00110101.

**解:**

<b>原数:</b>	1	0	0	1	1	0	0	0	<i>AND</i>
	0	0	1	1	0	1	0	1	
	<hr/>								
<b>结果:</b>	0	0	0	1	0	0	0	0	

# 或 (OR) 操作



有1结果就  
为1

OR		V	
x	y	x	OR y
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

## 或 (OR) 操作实例

使用 OR 操作对以下两个二进制数运算

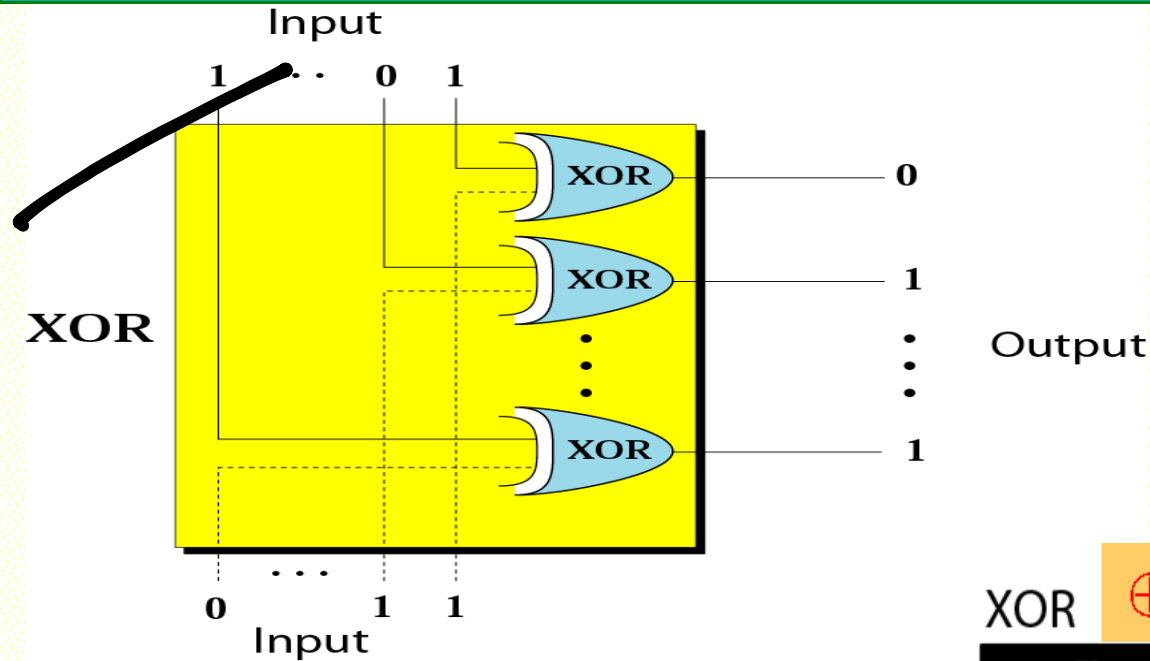
10011000 , 00110101

解:

原数	1	0	0	1	1	0	0	0	OR
				0	0	1	1	0	1
				-----					
结果	1	0	1	1	1	1	0	1	



# 异或 (XOR) 操作



**相同为0  
不同为1**

XOR



x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

# 异或 (XOR) 操作实例

使用 XOR 操作对以下两个二进制数运算

10011000 , 00110101.

**Solutio**

**n**

**原数**

1 0 0 1 1 0 0 0 **XOR**

0 0 1 1 0 1 0 1

-----

**结果**

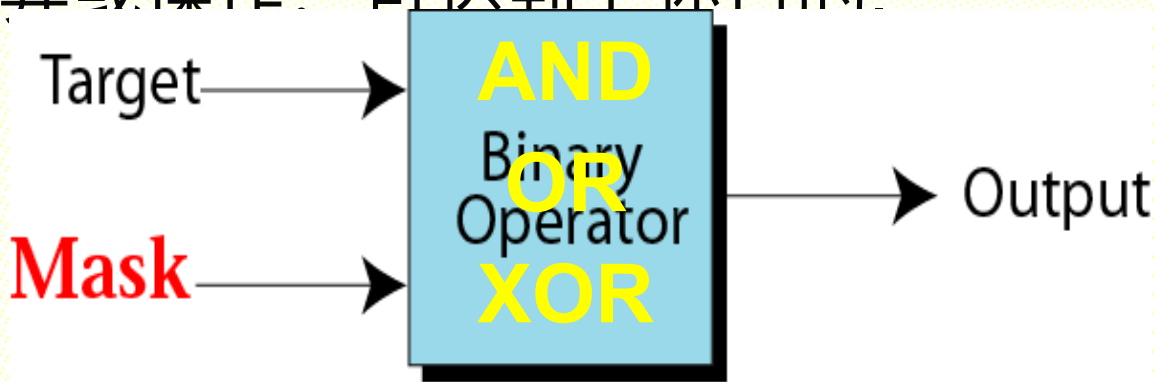
1 0 1 0 1 1 0 1

# 练习

1. 100 and 24  
202 and 255  
202 or 255
2. x99 and x99  
x99 or x99  
x99 xor x99
3. x55 and xAA  
x55 or xAA  
x55 xor xAA

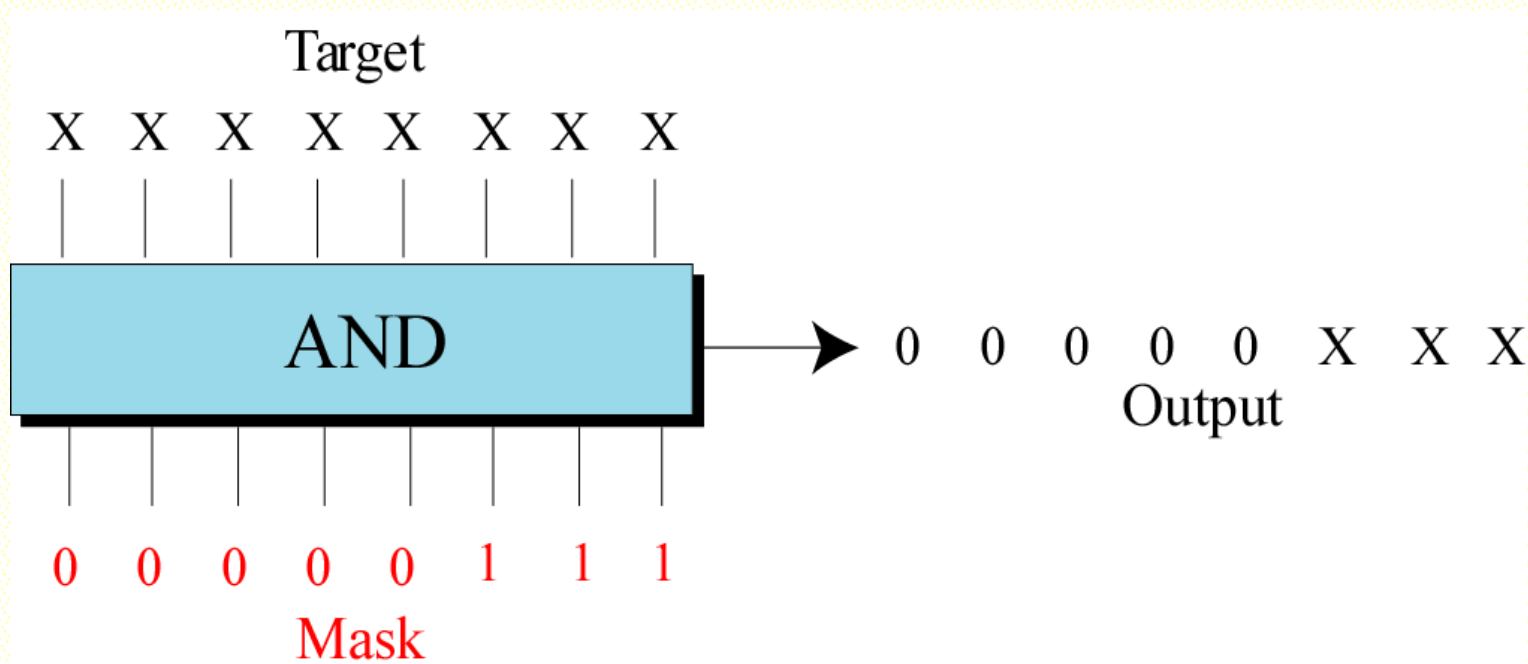
# 逻辑运算的应用

- 使用三种逻辑运算，可以改变位模式中的部分位，从而修改位模式。
- 可以使指定的位**清0**、**置1**、及**取反**。
- 使用掩码（**mask**）与原位模式进行与、或、异或操作，可达到上述目的。



# 清 0 操作

□ 把要清0的位，掩码中对应的位为0，其它位为1，进行逻辑与操作



# 清 0 操作实例1

使用掩码将位模式 10100110 的高5位清0.

解:

*The mask is 00000111.*

*Target*            1 0 1 0 0 1 1 0            *AND*

*Mask*                0 0 0 0 0 1 1 1

*Result*             0 0 0 0 0 1 1 0

## 清 0 操作实例2

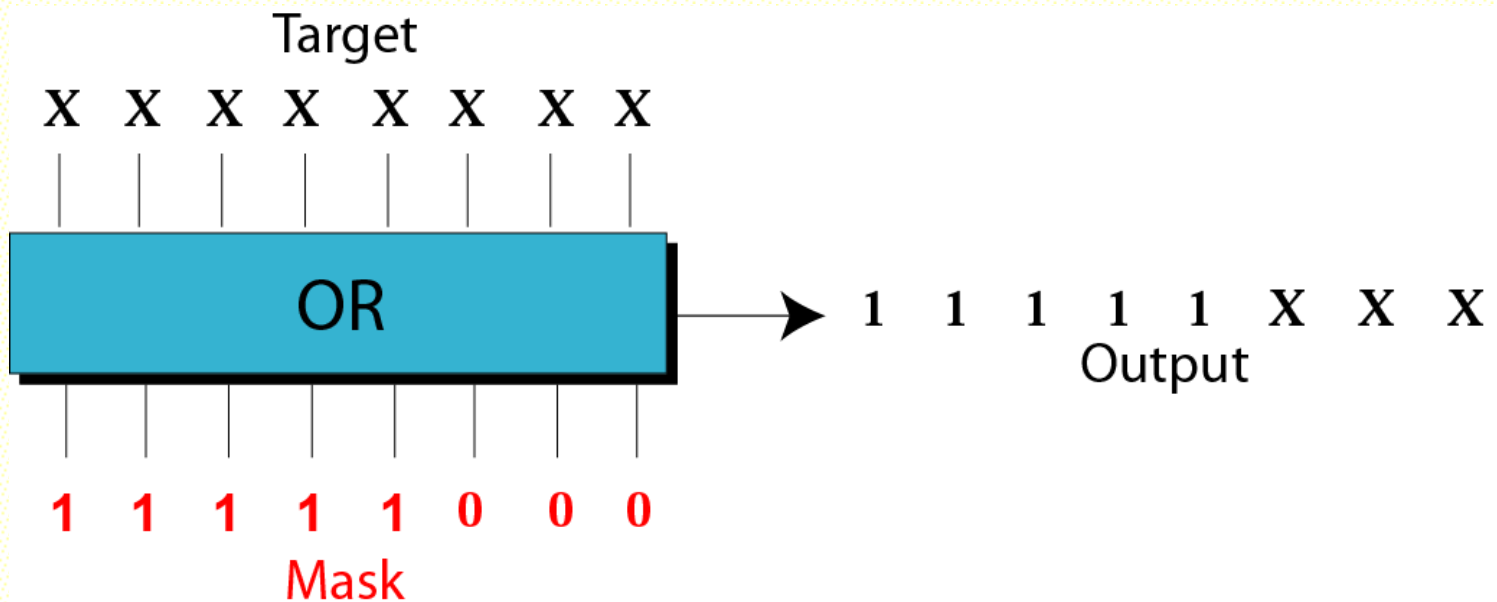
使用掩码将位模式 1 1 0 0 0 1 1 1 的第6位清0.

解:

<i>Target</i>	1 1 0 0 0 1 1 1	<i>AND</i>
<i>Mask</i>	1 0 1 1 1 1 1 1	
	-----	
<i>Result</i>	1 0 0 0 0 1 1 1	

# 置1 操作

❑ 把要置1的位，掩码中对应的位为1，其它位为0，进行逻辑**或**操作





# 置1 操作实例

使用掩码，将位模式 10100110 的高5位置1.

解：

*The mask is 11111000.*

*Target*            1 0 1 0 0 1 1 0            *OR*

*Mask*                1 1 1 1 1 0 0 0

*Result*             1 1 1 1 1 1 1 0

## 置1 操作实例2

使用掩码，将位模式 1000 00111 的第5位置1.

解：

Use the mask **00100000**.

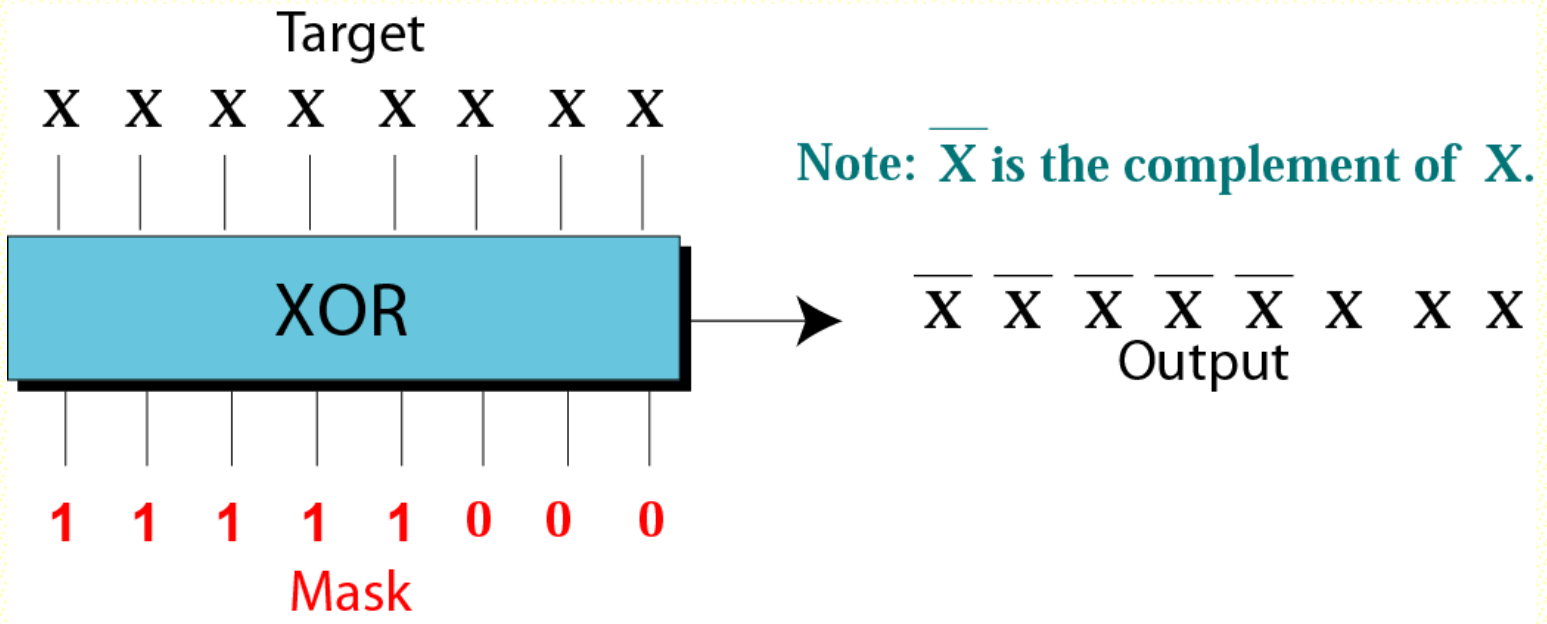
**Target**            1 0 0 0 0 1 1 1            **OR**

**Mask**                0 0 1 0 0 0 0 0

-----  
**Result**              1 0 1 0 0 1 1 1

# 取反 操作

❑ 把要取反的位，掩码中对应的位为1，其它位为0 进行逻辑**异或**操作



# 取反 操作实例1

使用掩码，将位模式 1010 0110 的高5位置取反。

解：

<b>Target</b>	1 0 1 0 0 1 1 0	<b>XOR</b>
<b>Mask</b>	1 1 1 1 1 0 0 0	
	-----	
<b>Result</b>	0 1 0 1 1 1 1 0	

## 取反 操作实例2

使用掩码，将位模式 1010 0110的所有位取反

解：

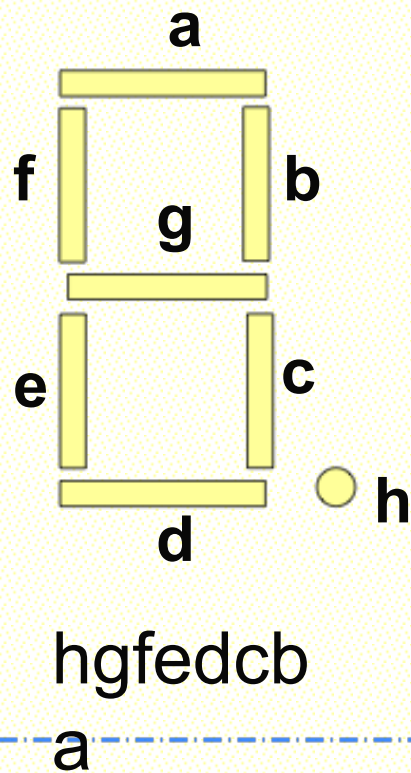
<i>Target</i>	1 0 1 0 0 1 1 0	<i>XOR</i>
<i>Mask</i>	1 1 1 1 1 1 1 1	
	-----	
<i>Result</i>	0 1 0 1 1 0 0 1	

# 逻辑操作综合应用

- 假设教室有16组灯，每组有1个开关控制，“1”表示灯亮，“0”表示灯灭，且所有灯的初始状态为关，按位模式的高位从前到后控制。
  - (1) 将低10组的灯打开
  - (2) 将间隔一组的灯打开
  - (3) 将高6组的灯关闭
  - (4) 将已打开的灯关闭，已关闭的灯打开

# 8段发光二极管

当某段中流过电流  
则该段发光



数字	发光的相应段	编 码	H
0	a、 b、 c、 d、 e、 f	00111111	3FH
1	b、 c	00000110	06H
2	a、 b、 d、 e、 g	01011011	5BH
3	a、 b、 c、 d、 g	01001111	4FH
4	b、 c、 f、 g	01100110	66H
5	a、 c、 d、 f、 g	01101101	6DH
6	a、 c、 d、 e、 f、 g		
7	a、 b、 c		
8	a、 b、 c、 d、 e、 f、 g		
9	a、 b、 c、 d、 f、 g		
A	a、 b、 c、 e、 f、 g	01110111	77H
b	c、 d、 e、 f、 g	01111100	7CH
C	a、 d、 e、 f		
d			
E			
F	a、 e、 f、 g	01110001	71H

## 4.2 移位运算

- 逻辑移位运算：

- 应用于不带符号位的数的模式

- 左移、右移

- 循环左移、循环右移

- 算术移位运算：

- 针对于补码模式

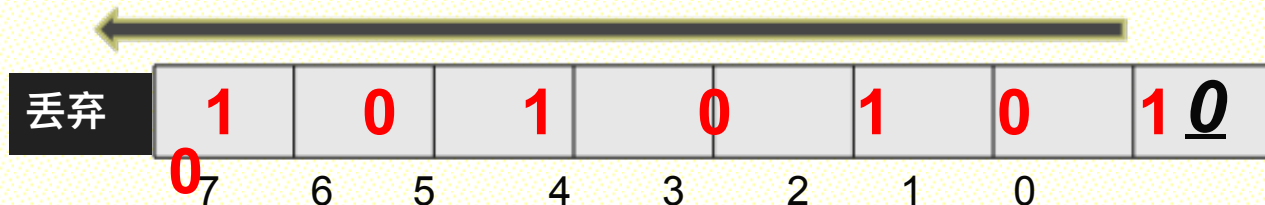
- 算术移位不能改变符号，否则，发生溢出

- 左移相当于乘2，右移相当于移除2

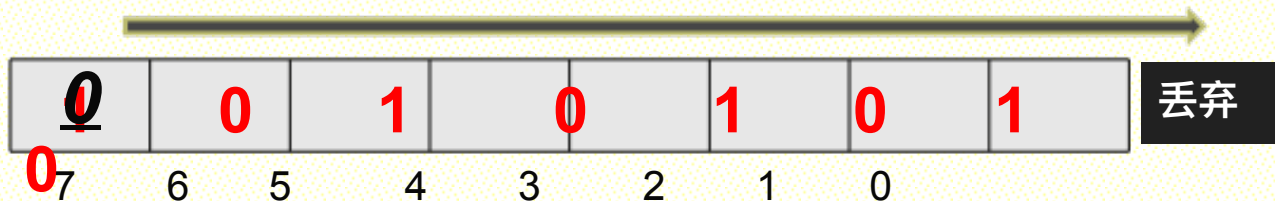


# 逻辑移位运算

- 逻辑左移：依次向左移一位，右边补0

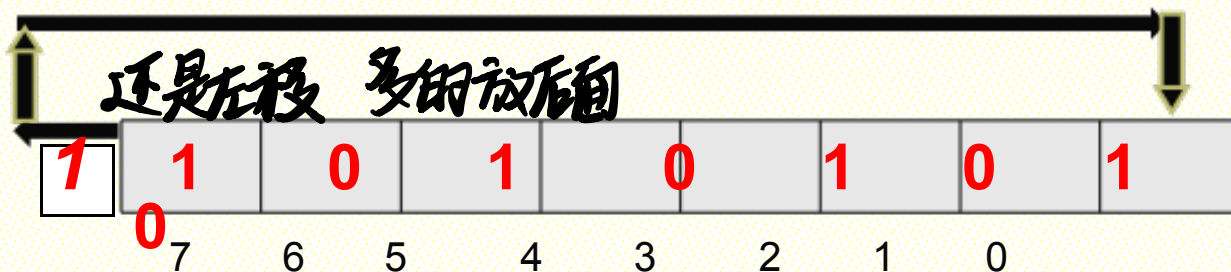


- 逻辑右移：依次向右移一位，左边补0



# 循环移位运算

循环左移：首尾相连，逆时针移动



• 循环右移：首尾相连，顺时针移动



# 逻辑移位运算实例

位模式: 10 10 10 01

逻辑右移:

10 10 10 01  
01 01 01 00

逻辑左移:

10 10 10 01  
01 01 00 10

循环右移:

10 10 10 01  
11 01 01 00

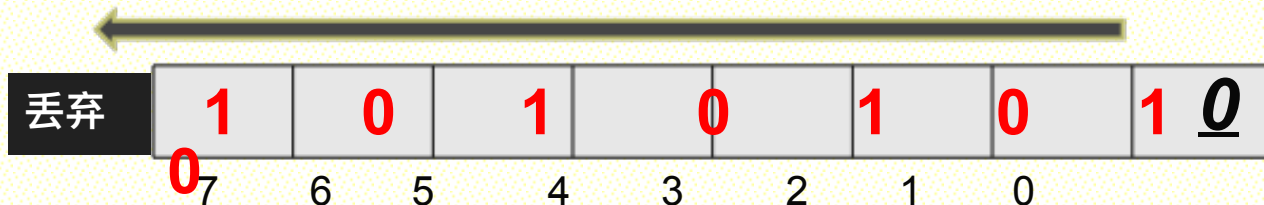
循环左移:

10 10 10 01  
01 01 00 11

数全是补码

# 算术移位运算

- 算术左移：依次向左移一位，右边补0



- 算术左移后，符号位与原数不同时，则溢出

算术右移：依次向右移一位，  
算术右移不会发生溢出



# 算术移位运算实例

将位模式：10 01 10 01 算术右移1位

10 01 10 01      -127  
11 00 11 00      -52

将位模式：11 01 10 01 算术左移1位

11 01 10 01      -39  
10 11 00 10      -78

将位模式：01 11 11 11 算术左移1位

01 11 11 11      +127  
11 11 11 10      -2

发生溢出

## 4.3 算术运算：整数加法

规则：

第1个数	第2个数	结果	进位
0	0	0	
0	1	1	
1	0	1	
1	1	0	1

超出高位的，舍弃

# 整数加法

利用补码计算:  $17 + 22$

解:

$(+17)$  补  $\square$  0001 0001

$(+22)$  补  $\square$  0001 0110

	0	0	0	1	0	0	0	1	
+	0	0	0	1	0	1	1	0	
Carry				1					
结果:	0	0	1	0	0	1	1	1	$\square$ 39

# 整数加法

利用补码计算:  $24 + (-17)$

解:

$(-17)$  补  $\square$  1110 1111

$(+24)$  补  $\square$  0001 1000

	0	0	0	1	1	0	0	0	
+	1	1	1	0	1	1	1	1	
Carry		1	1	1	1				
	1								
Result		0	0	0	0	0	1	1	1
+7									



# 整数加法

利用补码计算:  $(-35) + (+20)$

**解:**

$(-35)$ 补  $\square$  1101 1101

$(+20)$ 补  $\square$  0001 0100

	1	1	0	1		1	1	0	1	
+	0	0	0	1		0	1	0	0	
				1		1		1		
Carry										
	-----									
Result		1	1	1	1		0	0	0	1
-15										$\square$

# 整数减法

利用补码计算:  $101 - 62 = (+101) + (-62)$

**解:**

$(101)_{\text{补}} \square 0110 \ 0101$

$(-62)_{\text{补}} \square 1100 \ 0010$

	0	1	1	0	0	1	0	1	
	+	1	1	0	0	0	0	1	0
Carry		1	1						
	-----								
Result	0	0	1	0	0	1	1	1	<input type="checkbox"/>
39									

**进位舍弃**

# 整数加法

利用补码计算:  $127 + 5$

**解:**

127补  $\square$  0111 1111

5补  $\square$  0000 0101

	0	1	1	1	1	1	1	1	
+	0	0	0	0	0	1	0	1	
Carry		1	1	1	1	1	1	1	
-----									
Result	1	0	0	0	0	1	0	0	$\square$

**-124**

两正数相加, 结果为负

两负数相加, 结果为正

**发生溢出**

**超出表示范围**

## 4.4 算术运算：浮点数加法

先将IEEE表示的两个浮点数转化为真值表示：

1. S.E.M

**(s) 2<sup>n</sup> x 1.M**

2. 对阶：指数相同，小阶向大阶对齐

3. 将两数相加

4. 检查相加后的浮点数是否是规范化的数

5. 将规范化后的结果，转化为浮点数表示

将结果转化为16进制表示

## 4.4 算术运算：浮点数加法

将下面的两个IEEE表示的浮点数相加：

0 10000100 1011000000000000000000000000

0 10000010 0110000000000000000000000000

**解：**

1. 求出两数的真值：  $+25 \times 1.1011$  和  $+23 \times 1.011$

2. 对阶：  $23 \times 1.011 = 25 \times 0.01011$

3. 求和：  $(+25 \times 1.1011) + (+25 \times 0.01011)$   
 $= +25 \times 10.00001$

4. 规范化：  $+26 \times 1.000001$

5. 结果： 0 10000101 0000010000000000000000000000

# 实 练

1. 将浮点数-100.5625 用单精度的 IEEE表示，并将结果转化为16进制
2. 两个单精度浮点数分别为44D80000H, 4630000H  
求这两个浮点数的和，并将结果表示为16进制。

# 作业题

## P62 复习题

2

5

16

17

18

(以上不用交)

## P62 练习题

4

5

6

7

8

12

18: a