华南理工大学

South China University of Technology

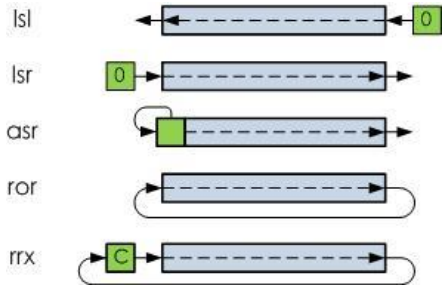
计算机与软件工程概论

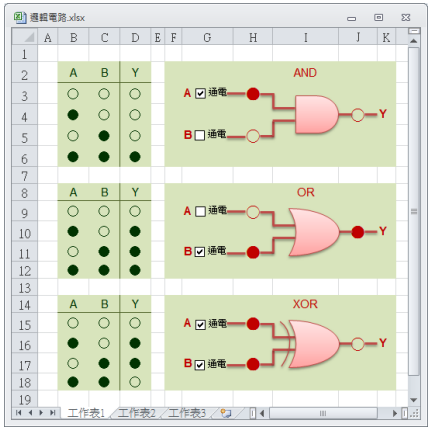
数据的运算

Dr 林育蓓
yupilin@scut.edu.cn
软件学院
2019

数据运算

- 计算机中的常见运算包括**算术运算**、**逻辑运算**和**移位运算**，这些运算是计算机进行数学计算与信息处理的基础。





所有图片均来自网络

算术运算

- 二进制数据的算术运算适用于整数和浮点数，包括加、减、乘、除等。
- 二进制补码的算术运算
  - 加法
  - 减法→加法

二进制补码中的加法运算

- 二进制补码中两个整数的相加法则： 两个位相加，将进位加到前一列。如果最左边的列相加后还有进位，则舍弃它（？？）。

表 2-4　两个二进制数按位相加的结果及进位

按位相加的二进制数	结果	进位
0+0	0	无
1+0 或 0+1	1	无
1+1	0	1

补码加法运算例子

- **例2-31** 用二进制补码方法计算(+17) + (+22)。

解：

$(+17)_{10}=(00010001)_2$ $(+22)_{10}=(00010110)_2$

进位　　1

0 0 0 1 0 0 0 1 +

0 0 0 1 0 1 1 0

结果 0 0 1 0 0 1 1 1

则 (+17) + (+22) = (+39)

补码加法运算例子

- **例2-32** 用二进制补码方法计算(+24) + (-17)。

解：

$(+24)_{10}=(00011000)_2$ $(-17)_{10}=(11101111)_2$

进位　1 1 1 1 1

　　0 0 0 1 1 0 0 0 +

　　1 1 1 0 1 1 1 1

结果　0 0 0 0 0 1 1 1

则 (+24) + (-17) = (+7)

补码加法运算例子

- **例2-34** 用二进制补码方法计算(+127) + (+3)。

解：

$(+127)_{10}=(11011101)_2$ $(+3)_{10}=(00010100)_2$

进位 1 1 1 1 1 1 1

0 1 1 1 1 1 1 1 +

0 0 0 0 0 0 1 1

结果 1 0 0 0 0 0 1 0

则 (+127) + (+3) = (-126)

在本例中，运算结果发生溢出错误。

二进制补码中的减法运算

- 二进制补码表示的一个优点是减法计算可以通过加法计算来实现。在进行减法运算时，首先把减数的符号取反，再与被减数相加。

补码减法运算例子

- **例2-35** 用二进制补码方法计算(+101) - (+62)。

解：

(+101)-(+62) = (+101) + (-62)

$(+101)_{10}=(01100101)_2$ $(+62)_{10}=(00111110)_2$ $(-62)_{10}=(11000010)_2$

进位　1 1

0 1 1 0 0 1 0 1 +

1 1 0 0 0 0 1 0

结果　0 0 1 0 0 1 1 1

则 (+101) - (+62) = (39)

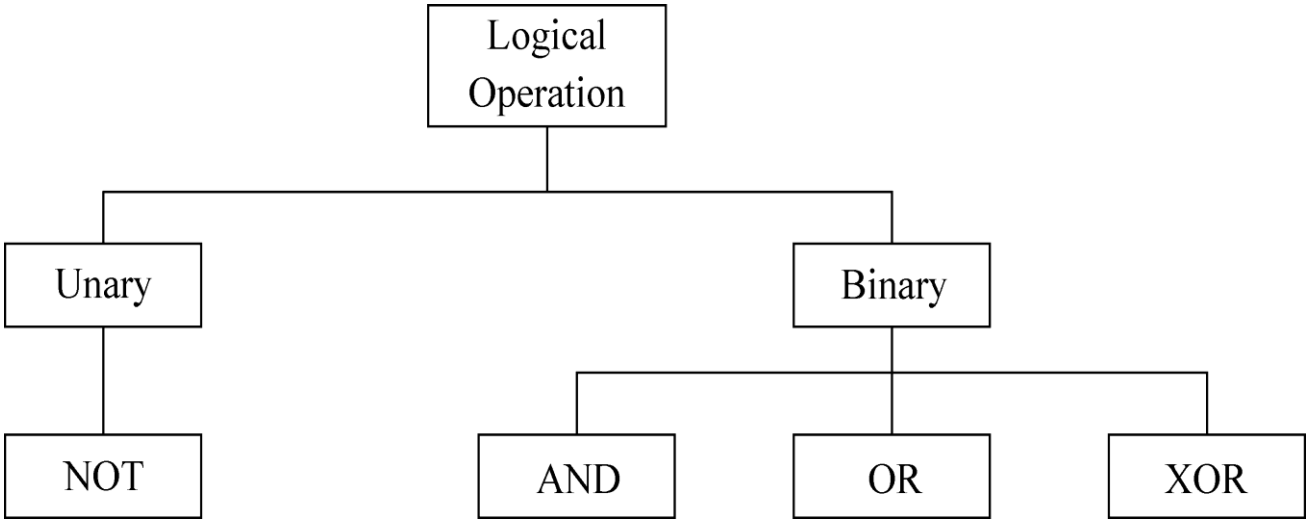
注意：在计算机上进行算术运算时，确保参与运算的数以及运算结果在指定位分配可表示的区间之内，避免发生溢出现象。

逻辑运算

- 逻辑运算中的变量有两个值： FALSE(假)和TRUE(真)，刚好对应二进制符号0和1。
- 一个位可能是0或1，可以假设**0代表逻辑“假”，1代表逻辑“真”**。
- 这样，存储在计算机存储器中的位就能代表逻辑“真”或逻辑“假”。

逻辑运算

- 分两类
  - 一元运算：逻辑运算作用在一个输入位上，即逻辑非运算
  - 二元运算：逻辑运算作用在两个输入位上，包括逻辑与（AND）、逻辑或（OR）、异或运算（XOR）



逻辑运算

- 逻辑规则如表2-5的逻辑真值表所示

表 2-5　逻辑真值表

输　入		输　出				
x	y	NOT(x)	NOT(y)	AND(x,y)	OR(x,y)	XOR(x,y)
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	1	0

逻辑非运算

- 非(NOT)运算是一元运算符，仅有一个输入，输出一个操作数。
- **运算过程**： 对输入的位模式逐位取反，即将0变为1，将1变为0。
- **例2-36** 对位模式10111010进行非运算。

模式1	1	0	1	1	1	0	1	0
NOT								
结果	0	1	0	0	0	1	0	1

逻辑与运算

- 与(AND)运算是二元运算符，有两个输入操作数，输出一个操作数。
- **运算过程**： 两个操作数对应位作为逻辑值，借助真值表进行运算。只有当参与运算的两个位都是1时，结果为1；否则结果为0。
- **例2-37** 对位模式10011000和00110101进行与运算。

模式1	1	0	0	1	1	0	0	0
模式2	0	0	1	1	0	1	0	1
AND								
结果	0	0	0	1	0	0	0	0

与运算过程中，如果参与运算的某位是0，则无论另一个输入对应的位是0还是1，运算结果均为0。

逻辑或运算

- **或(OR)**运算是二元运算符，有两个输入操作数，输出一个操作数。
- **运算过程**：对两个输入数逐位进行或运算，只有当两个位同为0时，结果为0；否则结果为1。
- **例2-38** 对位模式10011000和00110101进行或运算。

模式1	1	0	0	1	1	0	0	0
模式2	0	0	1	1	0	1	0	1
OR								
结果	1	0	1	1	1	1	0	1

或运算过程中，如果参与运算的某位是1，则无论另一个输入对应的位是0还是1，运算结果均为1。

逻辑异或运算

- **异或(XOR)**运算是二元运算符，有两个输入操作数，输出一个操作数。
- **运算过程**：对两个输入数逐位进行异或运算，只有当两位相同时，结果为0；否则结果为1。
- **例2-39** 对位模式10011000和00110101进行异或运算。

模式1	1	0	0	1	1	0	0	0
模式2	0	0	1	1	0	1	0	1
XOR								
结果	1	0	1	0	1	1	0	1

异或运算过程中，如果参与运算的某个位是1，则输出结果就是另一个输入对应位取反。

逻辑运算的作用

- 利用三种逻辑运算(与、或、异或)，可以修改位模式，如将指定位进行**复位(设置为0)**、**置位(设置为1)**或**反转**
- 掩码
  - 实现复位等功能的一种特殊构造的位模式

复位掩码

- 对指定的位进行**复位**操作
  - **与运算**可以将目标位模式中的指定位进行复位，复位掩码满足以下条件： 对于目标位模式中需要置0的位，掩码对应的位设置为0； 对于目标位模式中需要保持不变的位，掩码的相应位设置为1。

复位						
是/否	N	Y	N	N	Y	Y
模式	0	0	1	0	1	0
掩码	1	**0**	1	1	**0**	**0**
AND						
结果	0	**0**	1	0	**0**	**0**

复位掩码例子

- **例2-40** 构造一个掩码复位(清除)一个位模式的最左边5位，使用10100110进行测试。

复位								
是/否	Y	Y	Y	Y	Y	N	N	N
模式	1	0	1	0	0	1	1	0
掩码	**0**	**0**	**0**	**0**	**0**	1	1	1
AND								
结果	**0**	**0**	**0**	**0**	**0**	1	1	0

置位掩码

- 对指定的位进行置位操作
  - **或运算**可以将目标位模式中的指定位进行置位，置位掩码满足以下条件： 对于目标位模式中需要置1的位，掩码对应的位设置为1； 对于目标位模式中需要保持不变的位，掩码的相应位设置为0。

置位						
是/否	N	Y	N	N	Y	Y
模式	0	0	1	0	1	0
掩码	0	**1**	0	0	**1**	**1**
OR						
结果	0	**1**	1	0	**1**	**1**

置位掩码例子

- **例2-41** 构造一个掩码置位(置1)一个位模式的最左边5位，使用 10100110进行测试。

置位								
是/否	Y	Y	Y	Y	Y	N	N	N
模式	1	0	1	0	0	1	1	0
掩码	**1**	**1**	**1**	**1**	**1**	0	0	0
OR								
结果	**1**	**1**	**1**	**1**	**1**	1	1	0

反转掩码

- 对指定的位进行**反转**操作
  - **异或运算**可以将目标位模式中的指定位进行反转，也就是把指定位的值从0变为1、从1变为0。异或掩码满足以下条件： 对于目标位模式中需要反转的位，掩码对应的位设置为1； 对于目标位模式中需要保持不变的位，掩码的相应位设置为0。

反转						
是/否	N	Y	N	N	Y	Y
模式	0	0	1	0	1	0
掩码	0	**1**	0	0	**1**	**1**
XOR						
结果	0	**1**	1	0	**0**	**1**

反转掩码例子

- **例2-42** 构造一个掩码反转一个位模式的最左边5位，使用10100110进行测试。

反转								
是/否	Y	Y	Y	Y	Y	N	N	N
模式	1	0	1	0	0	1	1	0
掩码	**1**	**1**	**1**	**1**	**1**	0	0	0
XOR								
结果	**0**	**1**	**0**	**1**	**1**	1	1	0

移位运算

- 移位运算的功能是将一个数据所有的位左移或者右移指定的位数。
- 如果某个数据是用二进制补码格式表示的整数，对这个数据进行右移一位相当于进行一次除2运算； 对这个数据进行左移一位相当于进行一次乘2运算。在这个意义上的移位运算通常也称为算术移位运算。
  - 算术右移运算
  - 算术左移运算

算术右移运算

- 运算规则：
  - (1) 数据中所有位均右移一位；
  - (2) 最右端的位被移除；
  - (3) 最左端的位由原数据符号位复制。
- **例2-43** 对二进制补码表示格式的数据$(00010100)_2$进行1位的算术右移运算。

算术右移									
模式	0	0	0	1	0	1	0	0	$(20)_{10}$
右1	0	0	0	0	1	0	1	0	0
结果	0	0	0	0	1	0	1	0	$(10)_{10}$

算术左移运算

- 运算规则：
  - (1) 数据中所有位均左移一位；
  - (2) 最左端的位被移除；
  - (3) 最右端的位由0补充。
- **例2-45** 对二进制补码表示格式的数据$(00010100)_2$进行1位的算术左移运算

算术左移									
模式	$(20)_{10}$	**0**	0	0	1	0	1	0	0
左1	**0**	0	0	1	0	1	0	0	0
结果	$(40)_{10}$	0	0	1	0	1	0	0	0

移位运算-溢出

- 因为用二进制补码格式表示的数是一个**有符号的整数**，所以完成算术移位运算后，如果新的符号位与原先的相同，表示运算成功；如果新的符号位与原先的不同，表示发生了溢出，运算结果非法。

移位运算-溢出

- **例2-47** 对二进制补码表示格式的数据$(01010000)_2$进行1位的算术左移运算。

算术左移									
模式	$(80)_{10}$	**0**	1	0	1	0	0	0	0
左1	**0**	1	0	1	0	0	0	0	0
结果	$(-96)_{10}$	1	0	1	0	0	0	0	0