C语言中的各类运算

南京大学
计算机科学与技术系
袁春风
email: cfyuan@nju.edu.cn
2015.6

开场白

上一讲谈到，在高级语言程序的表达式中的各类运算，会被编译器转换为相应的运算指令，程序运行时，**CPU**执行这些指令，控制操作数在运算电路中被处理。

本讲主要介绍**C**语言程序中涉及的运算，包括算术运算、按位运算、逻辑运算、移位运算等。

C语言程序中涉及的运算

- 算术运算（最基本的运算）
  - 无符号数、带符号整数、浮点数的+、-、*、/、%运算等
- 按位运算
  - 用途
    - 对位串实现“掩码”（mask）操作或相应的其他处理

    （主要用于对多媒体数据或状态/控制信息进行处理）
  - 操作
    - 按位或：“|”
    - 按位与：“&”
    - 按位取反：“~”
    - 按位异或：“^”

    如何从数据y中提取低位字节，并使高字节为0？

    可用“&”实现“掩码”操作：y & 0x00FF

    例如，当y=0x0B2C时，得到结果为：0x002C

C语言程序中涉及的运算

- 移位运算
  - 用途
    - 提取部分信息
    - 扩大或缩小2、4、8...倍
  - 操作
    - 左移:：x<<k;　右移：　x>>k
    - 从运算符无法区分逻辑移位还是算术移位，由x的类型确定
    - 若x为无符号数：逻辑左（右）移

      高（低）位移出，低（高）位补0，可能溢出！

      问题：何时可能发生溢出？如何判断溢出？

      若高位移出的是1，则左移时发生溢出
    - 若x为带符号整数：算术左移、算术右移

      左移：高位移出，低位补0。可能溢出！

      溢出判断：若移出的位不等于新的符号位，则溢出。

      右移：低位移出，高位补符，可能发生有效数据丢失。

例：某字长为8的机器中，x、y和z都是8位带符号整数，已知x=-81，则y=x/2=？z=2*x=？

-81=-1010001B，故x的机器数为10101111

y=x/2 → x>>1：11010111

z=2*x → x<<1：01011110

~~y=-41~~？有效数据丢失　~~z=94~~ 溢出

C语言程序中涉及的运算举例

移位运算和按位运算举例

对于一个n（$n \geq 8$）位的变量x，请根据C语言中按位运算的定义，写出满足下列要求的C语言表达式。

（1）x的最高有效字节不变，其余各位全变为0。

（2）x的最低有效字节不变，其余各位全变为0。

（3）x的最低有效字节全变为0，其余各位取反。

（4）x的最低有效字节全变1，其余各位不变。

参考答案：

（1）(x>>(n-8))<<(n-8)

（2）x & 0xFF

（3）((x^ ~0xFF) >>8)<< 8

（3）x | 0xFF

C语言程序中涉及的运算

- 逻辑运算

  - 用途
    - 用于关系表达式的运算

    例如，if（x>y and i<100）then ……中的“and”运算
  - 操作
    - “||”表示“OR”运算
    - “&&”表示“AND”运算

      例如， if ((x>y) && (i<100)) then ……
    - “!”表示“NOT”运算
  - 与按位运算的差别
    - 符号表示不同：& ~ && ；| ~ || ； ……
    - 运算过程不同：按位 ~ 整体
    - 结果类型不同：位串 ~ 逻辑值

C语言程序中涉及的运算

- 位扩展和位截断运算
  - –用途
    - 类型转换时可能需要数据扩展或截断
  - –操作
    - 没有专门操作运算符，根据类型转换前、后数据长短确定是扩展还是截断
    - 扩展：短转长

      无符号数：0扩展（前面补0）

      带符号整数：符号扩展（前面补符） 正数补0，负数补1
    - 截断：长转短

      强行将高位丢弃，故可能发生“溢出”

C语言程序中涉及的运算

例1（扩展操作）：

在大端机上输出si, usi, i, ui的十进制和十六进制值是什么？

```
short  si = -32768;
unsigned short  usi = si;
int  i = si;
unsingned  ui = usi ;
```

```
si = -32768     80 00
usi = 32768    80 00
i = -32768    FF FF 80 00
ui = 32768    00 00 80 00
```

带符号整数：符号扩展

无符号整数：0扩展

C语言程序中涉及的运算

例2（截断操作）：i 和 j 是否相等？

```
int i = 32768;
short si = (short) i;
int j = si;
```

不相等！

```
i = 32768   00 00 80 00
si = -32768   80 00
j = -32768     FF FF 80 00
```

原因：对i截断时发生了“溢出”，即：32768截断为16位数时，因其超出16位能表示的最大值，故无法截断为正确的16位数！