整数除法运算

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

整数的除运算

- 对于带符号整数来说，n位整数除以n位整数，除$-2^{n-1}/-1=2^{n-1}$会发生溢出外，其余情况都不会发生溢出。Why?

  因为商的绝对值不可能比被除数的绝对值更大，因而不会发生溢出，也就不会像整数乘法运算那样发生整数溢出漏洞。

- 因为整数除法，其商也是整数，所以，在不能整除时需要进行舍入，通常按照朝0方向舍入，即正数商取比自身小的最接近整数（Floor，地板），负数商取比自身大的最接近整数（Ceiling，天板）。

  例如，7/2=?，-7/2=？

  7/2=3，-7/2=-3

- 整除0的结果可以用什么机器数表示？

  整除0的结果无法用一个机器数表示！

- 整数除法时，除数不能为0，否则会发生“异常”，此时，需要调出操作系统中的异常处理程序来处理。

整数的除运算

代码段一：

```
int a = 0x80000000;
int b = a / -1;
printf("%d\n", b);
```

运行结果为-2147483648

用objdump看代码段一的反汇编代码，得知除以 -1 被优化成取负指令neg，故未发生除法溢出

代码段二：

```
int a = 0x80000000;
int b = -1;
int c = a / b;
printf("%d\n", c);
```

为什么显示是“浮点异常”呢？

学完第7章应该能回答这个问题！

运行结果为“Floating point exception”，显然CPU检测到了异常

为什么两者结果不同！

变量与常数之间的除运算

- 对于整数除法运算，由于计算机中除法运算比较复杂，而且不能用流水线方式实现，所以一次除法运算大致需要30个或更多个时钟周期，比乘法指令的时间还要长！
- 为了缩短除法运算的时间，编译器在处理一个变量与一个2的幂次形式的整数相除时，常采用右移运算来实现。
  - 无符号：逻辑右移
  - 带符号：算术右移
- 结果一定取整数
  - 能整除时，直接右移得到结果，移出的为全0

    例如，12/4=3：0000 1100>>2=0000 0011

    -12/4=-3：1111 0100 >>2=1111 1101
  - 不能整除时，右移移出的位中有非0，需要进行相应处理

变量与常数之间的除运算

- 不能整除时，采用朝零舍入，即截断方式

  - 无符号数、带符号正整数（地板）：移出的低位直接丢弃
  - 带符号负整数（天板）：加偏移量(2^k-1)，然后再右移k位 ，低位截断（这里K是右移位数）

  举例：

  无符号数　14/4=3：0000 1110>>2=0000 0011

  带符号负整数　-14/4=-3

  若直接截断，则 1111 0010 >>2=1111 1100=-4≠-3

  应先纠偏，再右移: k=2, 故$(-14+2^2-1)/4=-3$

  即： 1111 0010+0000 0011=1111 0101

  1111 0101>>2=1111 1101=-3

变量与常数之间的除运算—举例

- 假设*x*为一个int型变量，请给出一个用来计算*x*/32的值的函数div32。要求不能使用除法、乘法、模运算、比较运算、循环语句和条件语句，可以使用右移、加法以及任何按位运算。

解：若x为正数，则将x右移k位得到商；若x为负数，则x需要加一个偏移量(2^k-1)后再右移k位得到商。因为$32=2^5$，所以 k=5。

即结果为: (x>=0 ? x : (x+31))>>5

但题目要求不能用比较和条件语句，因此要找一个计算偏移量b的方式

这里，x为正时b=0，x为负时b=31. 因此，可以从x的符号得到b

x>>31 得到的是32位符号，取出最低5位，就是偏移量b。

如果是正数，那么取出来的就是五个0,
如果是负数，那么取出来的就是五个1

```
int div32(int x)
{  /* 根据x的符号得到偏移量b */
    int b=(x>>31) & 0x1F;
    return (x+b)>>5;
}
```