二进制正负数

Dezeming Family

2021 年 8 月 15 日

DezemingFamily 系列书和小册子因为是电子书，所以可以很方便地进行修改和重新发布。如果您获得了 DezemingFamily 的系列书，可以从我们的网站 [https://dezeming.top/] 找到最新版。对书的内容建议和出现的错误欢迎在网站留言。

20210815：完成第一版。

目录

一　负数的补码表示

暑假放了两周，因为感冒休息了一周，感冒好了之后开始变得懒惰，于是好好地玩了几天，这几天最大的成就就是和妈妈去了动物园，并且自己亲手喂了喂长颈鹿吧。不得不说，长颈鹿实在是又高又大，挺让人震撼的。休息的一周里也没有写什么文章，临近开工，准备写点简单的东西预热一下。

在计算机表示正负数时，需要注意的第一点是：正负数不能相互冲突。也就是说，一个二进制序列只能表示为一个数。如果只有这样的要求通常很简单，我们让最高位为 0 时表示为正数，为 1 时表示为负数即可，以下面的 8bit 二进制数为例：

```
1    0 0 0 0 1 1 0 1 // 13
2    1 0 0 0 1 1 0 1 // -13
```

但是现在有一个新的问题，-13 加上 1 以后，应该是-12，但是按照二进制表示：

```
1     1 0 0 0 1 1 0 1
2 +   0 0 0 0 0 0 0 1
3 =   1 0 0 0 1 1 1 0 // -14
```

得到的结果却是-14，这样可不是我们希望的。我们需要寻求一种方式，使得负数的表示满足正确的加减法。

我们首先思考一件事，现在时钟的时针指向 3，我们希望它指向 10，那么我们应该怎么拨动它？答案是顺时针转 7 个刻度，或者逆时针转 5 个刻度。而 5+7=12，这里的 12 表示时针总共有 12 个刻度，我们定义 12 为“模”。

因此可知，在时针系统里，我们设顺时针转等于“加”，逆时针转等于“减”，所以 3+7=3-5=10。我们这里称 5 和 7 互为补数（相加为模长）。

现在，假如我们的计算是在 8bit 的有符号数上进行的，我们要计算 23-13，我们思考一下怎么表示“减号”，“减”其实就是“加”个负数。对于 8bit 的数而言，模为 $2^8=256$，23-13 等于 10，我们回想钟表，怎么由 23 得到 10 呢，答案是要么减去 13，要么加上（256-13），即：

```
1 23 - 13 = 23 + (256 - 13) =
2 (0001 0111) + ((1 0000 0000) - (0000 1011))
```

也就是说，-13 表示为：

```
-13
= ((1 0000 0000) - (0000 1011))
= (1 + (1111 1111) - (0000 1011))
```

其实，(11111111-00001011) 就是将 00001011 进行位反转，因为对于被减数而言，是 1 的位相减后位 0，是 0 的位相减后位 1。

所以，负数的表示就是正数的按位取反再加 1，因为用到了补数，所以称为补码。

现在看补码是否满足正确的加减法。假设有一个补码表示的数：-13，8bit 的补码为：11110011，我们计算其加 2 的值，分解来看：

```
-13 + 2
= (1 + (1111 1111) - (0000 1011)) + 010
= (1 + (1111 1111) - (0000 1011 - 010))
= -11的补码
```

可以看到是正确的。补码取反加 1 的意义是得到模值与被减数相减的结果。

我们举另外一个例子：计算 5-(-12)。其实相当于 5+(-12 的补码表示)。而一个负数的补码正好是它的绝对值的原码表示形式：

$$12: \ 00001100 \tag{一.1}$$

$$-12: \ 11110100 \tag{一.2}$$

$$Negate(11110100) + 1 = 00001100 \tag{一.3}$$

所以补码是一种非常好的负数表示方法。

二　打印 Bit 的小程序

我们将整个程序列出：

```
#include <iostream>
using std::cout;
using std::endl;
template <class T>
void printBit(const T& ob)
{
    char *p_base = (char *)&ob;
    char *p = p_base + sizeof(T) - 1;
    for (; p >= p_base; p--)
        for (int i = 7; i >= 0; i--)
            cout << (((*p)&(1 << i)) ? 1 : 0);
}
int main(void) {
    char a = 13;
    printBit<char>(a);
    cout << endl;
    printBit<char>(-a);
    cout << endl;
    system("pause");
    return 0;
```

```
21 }
```

我们可以用该程序来验证负数补码。

参考文献

[1] 暂无参考文献。