要将它作为数值来考虑。另外，还有一点非常重要，就是要对各种逻辑运算分别能实现什么有一个整体印象。形成这样的印象后，即使不看真值表也能判断出运算的结果。

图 2-12 表示的是对 NIKKEI 的头两个字母 NI 这一图形模式进行各种逻辑运算后的结果。假设白色部分表示 1，黑色部分表示 0。通过图 2-12，我们就会对逻辑运算有一个具体的把握，即“逻辑非是所有位的取反操作”“逻辑与是将一部分变为 0（复位到 0）的操作”“逻辑或是将一部分变为 1（复位到 1）的操作”“逻辑异或是将一部分进行取反（相同取 0，不同取 1）的操作”。

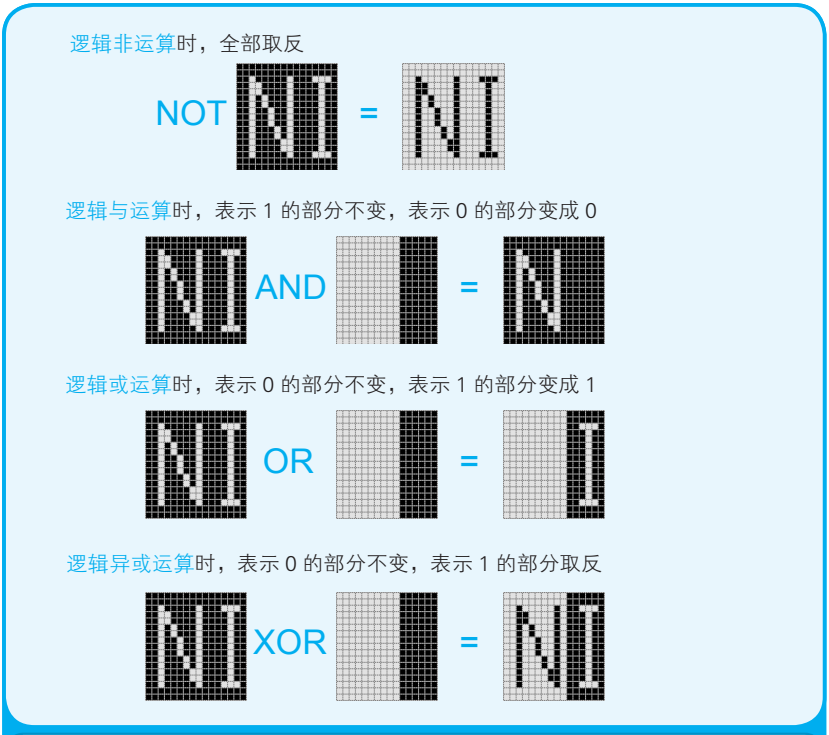
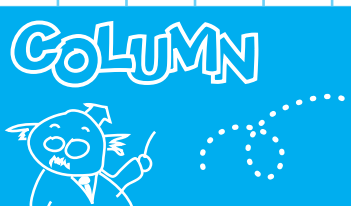


图 2-12　对图形模式进行 4 种逻辑运算的结果（这里白色部分表示 1，黑色部分表示 0）

学完本章后，大家应该对二进制数、移位运算、逻辑运算都十分了解了吧。不过，二进制数的小数 1011.0011 用十进制数来表示的话是多少呢？大家知道吗？想必大家也都很关心如何用二进制数来表示小数这一问题。下一章会有详细说明。



如果是你，你会怎样介绍？

向小学生讲解 CPU 和二进制

下面，我想邀请正在阅读本书的各位读者来进行一个挑战，那就是向完全不了解程序的人介绍程序的工作原理。如果理解了程序的本质，相信大家都可以用通俗易懂的语言进行讲解。当然了，介绍时不可以使用计算机专业术语。本书的专栏是笔者向一年级小学生及老奶奶介绍程序工作原理的一个尝试。亲爱的读者们，如果是你，你会怎样介绍呢？请在阅读以下内容的同时也思考一下吧。

笔者：大家见过电脑吗？

小学生：当然了！

笔者：在哪里见过呢？

小学生：学校里就有。

笔者：大家通常用电脑做什么呢？

小学生：画图或者上网。

笔者：不错！看来大家经常用电脑呀。那么，大家知道电脑内部是怎么构成的吗？

小学生：不知道……

笔者：那就让叔叔来告诉你们吧。来，大家看这里！

小学生：这是什么呀？

笔者：这个叫作 CPU，是电脑的零部件。正因为有了它，大家才能在电脑上画图和上网。算术计算的时候也会用到哦。电脑中有很多部件，最重要的就是这个 CPU。

小学生：咦，上面有好多昆虫一样的小脚（引脚）呢。

笔者：不错，挺善于观察的嘛！这个引脚会有电流通过。

小学生：通电后会怎么样啊？会发光吗？

笔者：CPU 不会发光。但是，通过电流信号，我们就可以给 CPU 发送指令或者传递数字信息等。比

如说，让电脑计算 1+2 的时候，就要把进行加法计算的命令和 1 和 2 这两个数字传递给 CPU。

小学生：电流是怎么把指令和数字告诉 CPU 的呢？

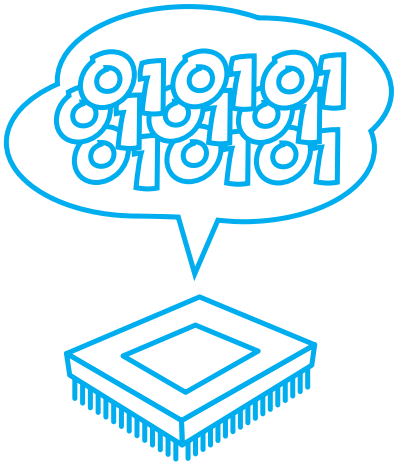
笔者：不错不错，又注意到一个有意思的地方。CPU 的引脚有电流通过时，数值为 1，没有电流通过的时候数值为 0，这是 CPU 里的规定。咱们平时使用的是 0 ～ 9 这 10 个数字，而电脑只用 0 和 1 这两个数字符号。怎么样，是不是很有意思呀？

小学生：就用 0 和 1，不会不够用吗？

笔者：不会啊！咱们来数数看。0、1、10、11、100、 …、1010。 你看，还是够用的。

小学生：1 的下一个是 10（一零），这好奇怪呀！

笔者：（呵呵呵，马上就要讲到重点了）不奇怪啊！这就是二进制数的计数方式。咱们用 0、1、2、3、…、9、10 这样的顺序来计数，数到 9 以后下一个就是 10，这就是十进制数的计数方式。电脑使用的是二进制数，用 0 和 1 来计



数，所以 0 和 1 的下一个数就是 10 了。

小学生：啊，不太明白呀……

笔者：（啊啊，不妙啊……）咱们换一种方式来考虑。咱们还是用 0～9 的数字来计数。但在遥远的宇宙边缘，生活着只用数字 0 和 1 的外星人。电脑就跟这个外星人差不多。这样讲大家明白了吧？

小学生：？？？

笔者：明白了吗？

小学生：嗯……

笔者：回答得这么不干脆啊？

小学生：差不多……明白了吧。

第3章 计算机进行小数运算时出错的原因

热身问答

阅读正文前，让我们先回答下面的问题来热热身吧。

1. 二进制数 0.1，用十进制数表示的话是多少？
2. 用小数点后有 3 位的二进制数，能表示十进制数 0.625 吗？
3. 将小数分为符号、尾数、基数、指数 4 部分进行表现的形式称为什么？
4. 二进制数的基数是多少？
5. 通过把 0 作为数值范围的中间值，从而在不使用符号位的情况下来表示负数的表示方法称为什么？
6. 10101100.01010011 这个二进制数，用十六进制数表示的话是多少？

怎么样？是不是发现有一些问题无法简单地解释清楚呢？下面是笔者的答案和解析，供大家参考。

答案

1. 0.5
2. 能表示
3. 浮点数（浮点数形式）
4. 2
5. EXCESS 系统表现
6. AC.53

解析

1. 二进制数的小数点后第一位的位权是 2^{-1}= 0.5。也就是说，二进制数 0.1 → 1 × 0.5 →十进制数 0.5。
2. 十进制数 0.625 转换成二进制数是 0.101。
3. 浮点数是指把小数用“符号 尾数 × 基数的指数次幂”这种形式来表示。
4. 二进制数的基数是 2，十进制数的基数是 10。以此类推，× × 进制数的基数就是 × ×。
5. EXCESS 是“剩余的”的意思。例如，把 01111111 看作是 0 的话，比这个数小 1 的 01111110 就是 −1。
6. 整数部分和小数部分一样，二进制数的 4 位，就相当于十六进制数的 1 位。

本章重点

大家可能会认为“万能的计算机是不会出现计算错误的”。但实际上，依然存在程序运行后无法得到正确数值的情况。其中，小数运算就是一个典型的例子。本章将会说明计算机进行小数处理的机制。这也是所有程序员都需要掌握的基础知识之一。掌握了这个知识，也就了解了计算机在运算时为什么会出错，以及应该如何避免出错。这个问题可能会有些难懂，因此本章进行了非常详细的说明，也请大家仔细阅读。

3.1 将 0.1 累加 100 次也得不到 10

首先，我们来看一个计算机运算错误（无法得到正确结果）的例子。代码清单 3-1 是将 0.1 累加 100 次，然后将结果输出到显示器上的 C 语言程序。

代码清单 3-1 将 0.1 累加 100 次的 C 语言程序

```
#include <studio.h>

void main(){
    float sum;
    int i;

    // 将保存总和的变量清 0
    sum = 0;

    //0.1 相加 100 次
    for (i = 1;  i <= 100; i++) {
        sum += 0.1;
    }

    // 显示结果
    printf("%f\n", sum);
}
```

首先把 0 赋值给变量 *sum*，然后在此基础上累加 100 次 0.1。*sum* += 0.1; 表示为现在的 *sum* 值加 0.1。for(*i* = 1; *i* <= 100; *i*++){...} 表示将 {} 内包含的处理重复 100 次。最后，使用 printf("%f\n", *sum*);，将累加 100 次 0.1 后的变量 *sum* 的值输出到显示器上。

大家心算一下就能知道，0.1 累加 100 次后的结果是 10。但是，代码清单 3-1 的程序运行后，显示器上显示的结果并不是 10（图 3-1）。

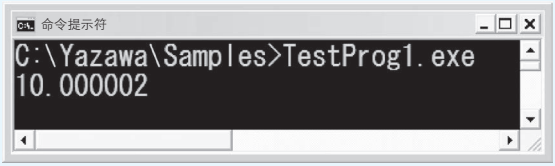


图 3-1　代码清单 3-1 的运行结果不是 10

程序没错，计算机也没有发生故障，当然，C 语言也没有什么问题。可为什么会出现这样的结果呢？这时，如果考虑一下计算机处理小数的机制，就讲得通了。那么，计算机内部是如何处理小数的呢？

3.2　用二进制数表示小数

在第 2 章中，我们对整数的二进制数表现方法做了说明。由于计算机内部所有的信息都是以二进制数的形式来处理的，因此在这一点上，整数和小数并无差别。不过，使用二进制数来表示整数和小数的方法却有很大的不同。

在说明计算机如何用二进制数表示小数的具体方法前，我们先做个热身，把 1011.0011 这个有小数点的二进制数转换成十进制数。小数

点前面部分的转换方法在第 2 章中已经介绍过了。只需将各数位数值和位权[①]相乘，然后再将相乘的结果相加即可实现。那么，小数点后面的部分要如何进行转换呢？其实，它的处理和整数是一样的，将各数位的数值和位权相乘的结果相加即可（图 3-2）。

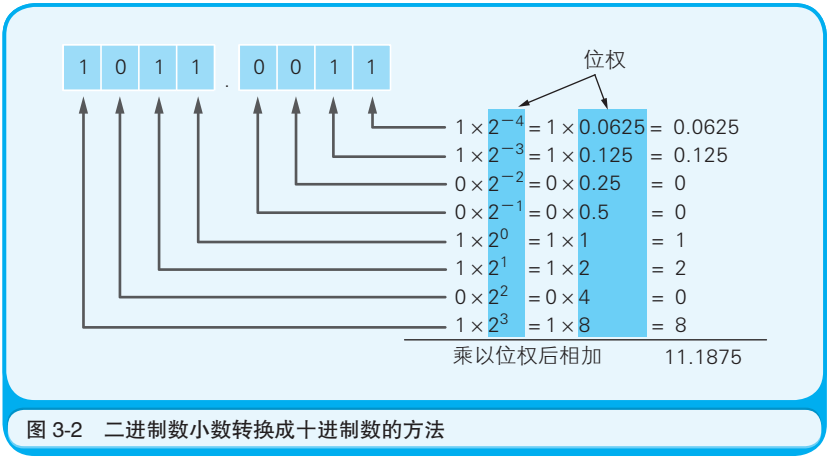


图 3-2 二进制数小数转换成十进制数的方法

二进制数小数点前面部分的位权，第 1 位是 2 的 0 次幂、第 2 位是 2 的 1 次幂……以此类推。小数点后面部分的位权，第 1 位是 2 的 −1 次幂、第 2 位是 2 的 −2 次幂，以此类推。0 次幂前面的位的位权按照 1 次幂、2 次幂……的方式递增，0 次幂以后的位的位权按照 −1 次幂、−2 次幂……的方式递减。这一规律并不仅限于二进制数，在十进制数和十六进制数中也同样适用。既然二进制数的小数点后第 3 位是 2 的 −3 次幂（0.125），第 4 位是 2 的 −4 次幂（0.0625），那么小数点以后的 .0011 转换成十进制数就应该是 0.125 + 0.0625 = 0.1875。此外，由于整数部分的 1011 转换成十进制数是 11。因此，二进制数 1011.0011 转换成十进制数就是 11 + 0.1875 = 11.1875。

① 位权是用来与各数字位的数字相乘的数值，具体请参照第 2 章。