**2-3-2-2 2進表記**

在计算机内部中，表示数值的形式主要使用二进制。根据整数部分与小数部分的位数是否预先设定，分为：

* 固定小数点数：小数点位置固定，适合处理结构明确的数值；
* 浮动小数点数：小数点位置可随数值大小自动变化。

① **固定小数点数**

固定小数点数是一种将小数点**固定在某个位置**来表示数值的方法。

一般来说，小数点通常被固定在**最低位的右侧**，因此主要用于处理整数。

在这种表示方式中，先将数值转换成二进制，然后**根据小数点位置使用二进制表示**。

在固定小数点数中，**表示负数的方式**有以下两种。

a：**符号付き絶対値表現**

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。这种方式使用**最高位作为符号位**：正数时为 0，负数时为 1。其余的位则原样存放二进制的绝对值。

b：**補数表現**

在表示负数时，使用**正数的补码**作为负数的编码方式。

补码是以某个数为“基准值”，表示某个值**距离基准值还差多少**的方式。

在制n 进数中，存在两种补码概念：

* n−1 的补码：基准值是该位数中所有位的最大值（例如：999）
* n 的补码：基准值是高一位的最小值（例如：1000）

计算补码时，是将“要取补码的数”从“基准值”中减去来得到的。

文本

AI 生成的内容可能不正确。文本

AI 生成的内容可能不正确。**補数表現例:**

一般来说，在**固定小数点数**中，常常使用 **2 の補数**表示，原因如下：

**1．可以将减法转化为加法来处理**

如果使用a：**符号付き絶対値表現**，就需要先判断操作数的最高位（符号位）是正是负，从而决定是用加法还是减法。

文本

AI 生成的内容可能不正确。但如果使用b：**補数表現**，则**无需判断符号**，可以直接统一用加法进行计算, 从而简化运算逻辑。

采用这种思路，计算机只需具备**加法电路**和**补码求取电路**即可，无需单独的减法电路。此外，乘法可以用加法反复实现，除法可以通过减法反复实现，因此四则运算（**加、减、乘、除**）理论上只需“加法”一种运算就能全部完成。

2．**可表示数值的范围更广**

使用a：**符号付き絶対値表現**的表示法时，会出现“+0” 和 “−0” 这两种“0”的二进制形式。

而使用b：**補数表現**表示时，“0” 只有一种形式。

表格

AI 生成的内容可能不正确。因为无论哪种表示，位数信息量相同，因此补码表示法**能表示的数值范围更广**

图片包含 图表

AI 生成的内容可能不正确。**n 位补码能表示的数值范围为：**

例如：8位时，就是 -128 ~ +127

② **浮動小数点数**

当数值表示中可用的**位数（bit 数）受到限制**时，固定小数点方式所能表示的数值范围也会被限制。

因此，为了处理**非常大的数值**时，就必须增加可用位数。  
然而，实际上不可能无限制地增加用于记录数据的比特数。

于是，为了表示非常大的值或小数点以下非常小的值，便使用了**浮動小数点数**这种表示法。

在说明计算机内部的表示方式前，先用十进制来解释浮点小数的机制。  
假设有如下数值：

＋456,000,000,000

为了便于说明，假设记录符号或每一位数字需要一个单位的话，  
要完整记录这个数值需要 **13个单位**。

＋　4　5　6　0　0　0　0　0　0　0　0　0

此时，如果将数值转换为如下形式，仅记录必要的信息，就可以只用 **6个单位** 来表示：

+456,000,000,000

= + 0.456 × 10 ¹² ※（－1）⁰ = ＋1， （－1）¹ = －1

= （－1）⁰ × 0.456 × 10 ¹²

**这里分成的三部分可以表示为**

**符号**: 表示数值是正还是负: 0

**仮数**: 小数点以后的数值: 456

**指数**: 仮数要乘以10的几次方: 12

**这是浮動小数点数的思维方式。  
通过这种方式，可以用同样的位数表示从极大到极小的数值。**

此外，用来表示幂的“10”称为**基数**。在本例说明中使用的是十进制，因此为了便于小数点的移动而采用了“10”，但在计算机内部使用的是二进制，因此会使用“2”或“16”。。

**浮動小数点数的基本型:**

计算机内部实际使用的**浮動小数点数**表示形式会因设备而异。  
在这里介绍被标准化为 IEEE 754 格式的两种表示方式：

* 文本

  AI 生成的内容可能不正确。**単精度浮動小数点数（32位）**
* **倍精度浮動小数点数（64位）**

此外，**倍精度浮動小数点数**具有更大的数值范围与更高的精度。

**浮動小数点数各部的代表性表示方法**

**a：符号部（S）**

用于表示数值的正负。正数为 0，负数为 1

**b：指数部（E）**

指数部分用于表示对基数（如 2 或 10）的幂乘。主要有两种常见的表示方式：

1. **２の補数方式**
   * 直接使用补码表示正负指数。
2. **エクセス方式（Excess; 偏移表示法）**
   * 给指数加上一个固定值（称为**バイアス值 /偏移值 bias**）之后再存储。
   * 该**バイアス值**由指数部分的位数决定。

**単精度浮動小数点数表格

AI 生成的内容可能不正确。常见的エクセス方式**

**c：仮数部（M）**

用于表示小数点以下的数值。**通常会事先进行正规化**。  
此时，可能使整数部分为0，也可能设为1：

* 如果整数部分为 0：  
  例）0.101101 → 仮数部 = 101101
* 如果整数部分为 1（常见 IEEE 754 正规化方式）：  
  例）1.011010 → 仮数部 = 011010 （省略掉开头的 1）

**「正規化」**

正规化是为了增加**仮数位**中可用数字数量，从而保持数值精度而进行的操作。

通常通过减少小数点以下多余的 0 来实现。

箱线图

AI 生成的内容可能不正确。例如，若用7位的仮数位表示十进制数（0.000123456789）₁₀，  
将其原样记录到仮数部中时，结果如下：

图片包含 图示

AI 生成的内容可能不正确。而若在记录前先进行正规化，则仮数部内容如下：

换句话说，正规化后能够表示更多数字（**有效数字位数增加**），因此数值的精度更高。

**例题**

**例：将十进制数（1234.625）₁₀ 用下述 IEEE 754 格式的单精度浮点数（32位）表示。**

**符号位（1位）：正为0，负为1。  
指数位（8位）：采用基数为2的 Excess 127 表示法。  
仮数位（23位）：采用整数部分为1的正规化表示法。**

**步骤:**

1. **将十进制数（1234.625）₁₀ 转换为二进制**。

（1234.625）₁₀ ＝（10011010010.101）₂

1. **将上面求得的二进制进行整数部分为1正规化。**

正规化表示：＋（1.00110100101）₂ × 2 ¹⁰

1. **将指数用 Excess 127 的8位二进制表示**。

10次方 → 10 + 127 = 137 → （10001001）₂

1. 按照表示格式组合写出：
2. 10001001　00110100101000000000000

符号位（1位）：0  
指数位（8位）：10001001  
仮数位（23位补0）：00110100101000000000000