鸽了两天，今天让我们来了解一个小知识点：

整数（Integer）在内存中的存储方式

整数在内存中通常采用二进制补码（Two's Complement）形式进行存储。以下是一些关键点：

补码表示法：

正整数：直接用二进制表示。例如，十进制的5在8位系统中表示为00000101。

负整数：负数用补码表示。补码是通过将数值的绝对值的二进制形式取反然后加1来获得。例如，十进制的-5在8位系统中表示为11111011。

内存布局：

整数通常占用固定的字节数，常见的是4字节（32位）或8字节（64位）。

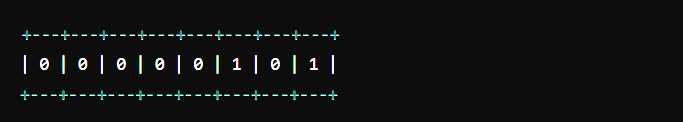
例如，32位整数范围是从-2,147,483,648到2,147,483,647，64位整数范围是从-9,223,372,036,854,775,808到9,223,372,036,854,775,807。

示例：

8位系统中，十进制的5表示为00000101，而-5表示为11111011。

在32位系统中，十进制的5表示为00000000 00000000 00000000 00000101，而-5表示为11111111 11111111 11111111 11111011。

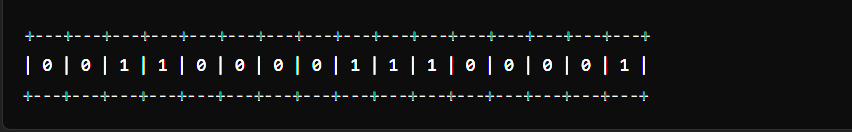
以8位二进制表示的正整数5为例：



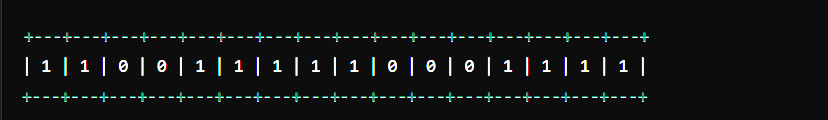
以8位二进制表示的负整数-5为例（补码表示法）：



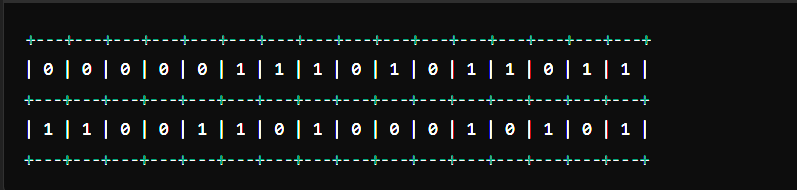
以16位二进制表示的正整数12345为例：



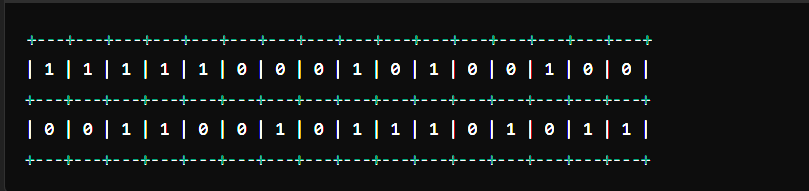
以16位二进制表示的负整数-12345为例（补码表示法）：



以32位二进制表示的正整数123456789为例：



以32位二进制表示的负整数-123456789为例（补码表示法）：



浮点数（Floating Point Number）在内存中的存储方式

浮点数在内存中采用IEEE 754标准表示，包括单精度（32位）和双精度（64位）两种常见形式。

IEEE 754标准：

浮点数表示分为三部分：符号位（Sign Bit），指数位（Exponent Bits），尾数位（Mantissa Bits）。

单精度浮点数（32位）：

符号位（1位）：0表示正数，1表示负数。

指数位（8位）：表示指数部分，使用偏移量为127的移码（biased exponent）。

尾数位（23位）：表示尾数（实际小数部分），隐含了一个1，即1.xxx。

双精度浮点数（64位）：

符号位（1位）：0表示正数，1表示负数。

指数位（11位）：表示指数部分，使用偏移量为1023的移码。

尾数位（52位）：表示尾数（实际小数部分），隐含了一个1，即1.xxx。

内存布局：

单精度：1位符号位 + 8位指数位 + 23位尾数位。

双精度：1位符号位 + 11位指数位 + 52位尾数位。

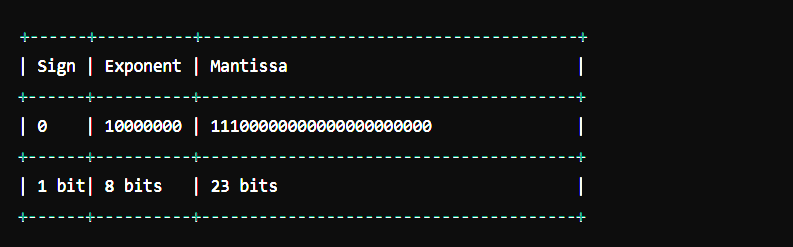
示例：

单精度浮点数：

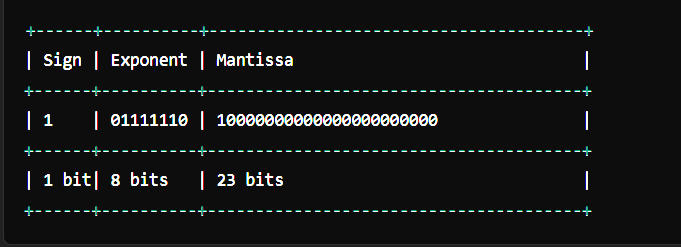
十进制的3.75表示为：0 10000000 11100000000000000000000。

符号位是0（正数），指数位是10000000（表示128，实际指数是1），尾数位是11100000000000000000000（表示1.75，隐含的1）。

单精度浮点数（32位）（3.75）：



（-0.75）：

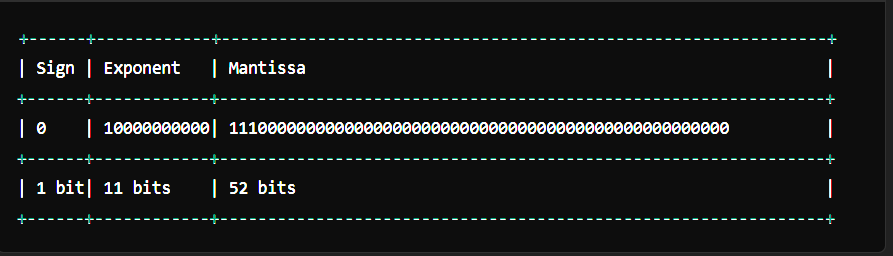


双精度浮点数：

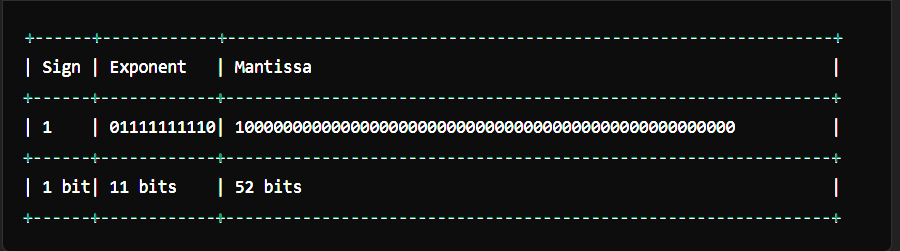
十进制的3.75表示为：0 10000000000 1110000000000000000000000000000000000000000000000000。

符号位是0（正数），指数位是10000000000（表示1024，实际指数是1），尾数位是1110000000000000000000000000000000000000000000000000（表示1.75，隐含的1）。

双精度浮点数（64位）（3.75）：



（-0.75）：



解释补充：

1.整数存储方式：

正数直接转换为二进制表示，负数则通过补码表示（即对其绝对值取反加1）。

2.浮点数存储方式：

按照IEEE 754标准进行存储。

单精度（32位）由1位符号位、8位指数位和23位尾数位组成。

双精度（64位）由1位符号位、11位指数位和52位尾数位组成。

符号位0表示正数，1表示负数；指数位使用移码（偏置值）表示；尾数位表示有效数字部分。

这里对整数和浮点数的内存仅需了解，在日后出问题时方便我们找到问题的根源不需要完全记住。