

Lecture 6: 知识 - 整数的表示与运算

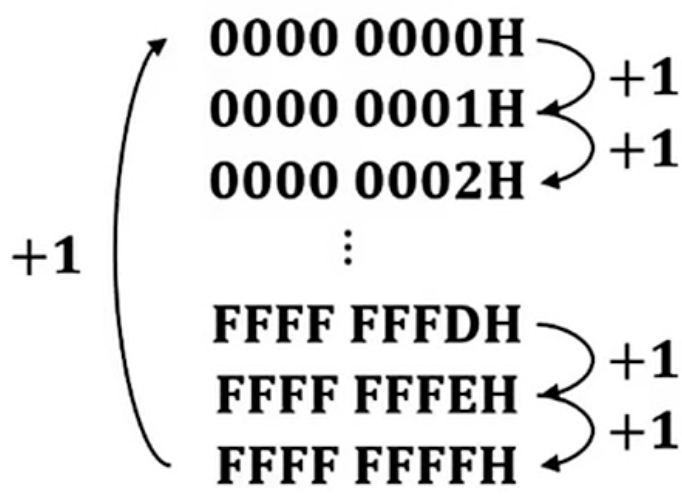
整数的表示

无符号整数

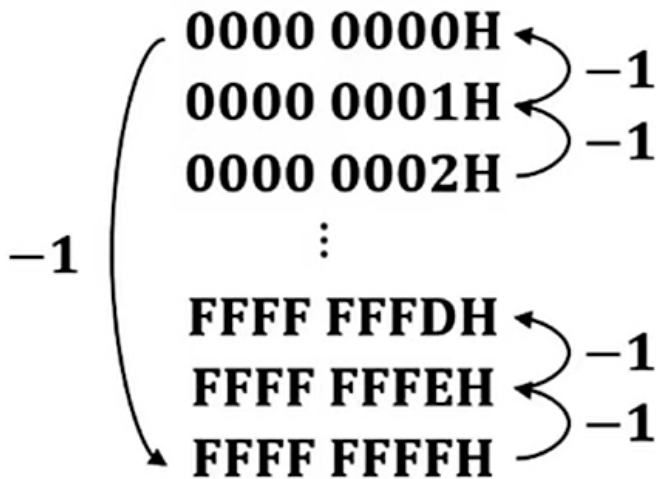
概念: 无符号整数

- 定义:
 - 当一个编码的全部**二进制位均为数值位而没有符号位**时, 该编码表示就是无符号整数, 简称无符号数;
 - 默认数的符号为正;
 - 因为无符号整数省略了一位符号位, 所以在字长相同的情况下, 它能表示的最大数比有符号整数能表示的大;
- 使用场合:
 - 一般在全部是正数运算且不出现负值结果的情况下, 使用无符号整数表示;
 - 例如, 可用无符号整数进行地址运算, 或用它来表示指针;
- 大小范围:
 - 8位无符号整数: $0 \sim 2^8 - 1$, 即 $0 \sim 255$
 - 8位有符号整数: $0 \sim 2^7 - 1$, 即 $-128 \sim 127$
- 补充: 无符号数循环取值

无符号32位整数的取值范围



无符号32位整数的取值范围



有符号整数

概念

- 定义：
 - 将符号数值化，并将符号位放在有效数字的前面，就组成了有符号整数；
- 实现：
 - 虽然前面介绍的原码、补码、反码和移码都可以用来表示有符号整数，但补码表示有其明显的优势；
 - (1) 与原码和反码相比，0 的补码表示唯一；
 - (2) 与原码和移码相比，补码运算规则比较简单，且符号位可以和数值位一起参加运算；
 - (3) 与原码和反码相比，补码比原码和反码多表示一个最小负数。
 - 计算机中的有符号整数都用 **补码** 表示；

范围：

- 范围：
 - **n** 位有符号整数的表示范围是：

$$-2^{n-1} \sim 2^{n-1} - 1$$

解释：

- **n-1**：因为正数和负数各占一半，所以是 **n-1**
- **-1**：因为 **0** 需要占一位；

补充：

★ 总结: short 的表示范围

- short 为 2B, 即 16 位;
- 无符号时:
 - 最大: 65535 = 1111, 1111, 1111, 1111
 - 最小: 0 = 0000, 0000, 0000, 0000
- 有符号时:
 - 最大: 32767 = 0111, 1111, 1111, 1111
 - 最小: -32768 = 1000, 0000, 0000, 0000

C 语言中的整数类型及类型转换

整型数据类型

概念: 整型数据类型

- 概念:
 - 字符型: `char`, 8 位, 1 字节
 - `char` 默认是无符号整数
 - 短整型: `short` 或 `short int`, 16 位, 2 字节;
 - `short`、`int`、`long` 等不指定 `signed/unsigned` 时都默认是有符号整数
 - 无符号整数 (`unsigned short/int/long`) 的全部二进制位均为数值位, 没有符号位, 相当于数的绝对值;
 - 整型: `int`, 32 位, 4 字节;
 - 长整型: `long` 或 `long int`, 在 32 位机器中为 32 位, 在 64 位机器中为 64 位;
- 补充:
 - `signed/unsigned` 整型数据都是按补码形式存储的;

数据类型转换

概念: `int` 型和 `float` 型占 4 字节, `double` 型占 8 字节;

概念: 有符号/无符号数转换

- 原理:
 - 强制类型转换的结果是保持位值不变, 仅改变了解释这些位的方式;
- 举例:
 - `short x=-4321; unsigned short y = (unsigned short)x;`
 - `//y = 61215;`
 - 各位保持不变, 按照无符号整数的方式解释、得到了 61215
- 注意:
 - 若同时有无符号数和有符号数参与运算, 则 C 语言标准规定按无符号数进行运算;

概念: 不同字长整数转换

- 原理：
 - 强制类型转换的结果是保持位值不变，仅改变了解释这些位的方式；
- 实现：
 - 大字长 `->` 小字长：
 - 系统把多余的高位部分直接截断，保留低位部分、直接赋值；
 - 小字长 `->` 大字长：
 - 其他情况：
 - 在位值相等的条件下还要**补充高位的符号位**，可以理解为数值的相等
 - 直接全部保留，高位部分**除开符号位**、其他补 `0`；
 - 整数：
 - 若原数字是无符号整数，则进行零扩展：
 - 扩展后的高位部分用 `0` 填充；
 - 若原数字是符号整数，进行符号扩展：
 - 扩展后的**高位部分用原数字符号位填充**；
 - 其实两种方式扩展的高位部分都可理解为**原数字的符号位**；
- 注意：
 - 小字长向大字长的转换，在位值相等的条件下还要补充高位的符号位，可以理解为数值的相等；
 - 注意，`char` 型为 8 位无符号整数，其在转换为 `int` 型时高位补 `0` 即可。
 - 无符号数 `->` 有符号数：需要进行 `0` 扩展，高位的部分用 `0` 填充；