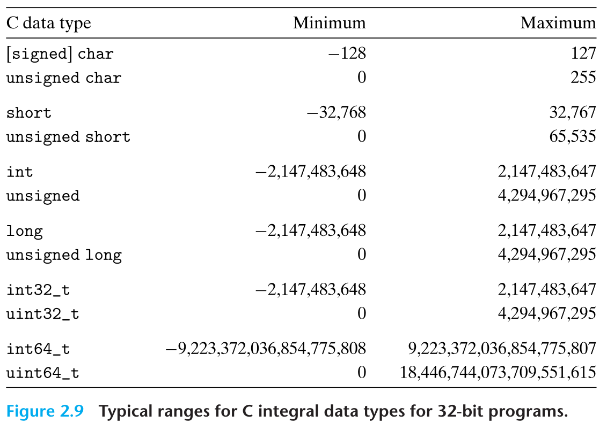
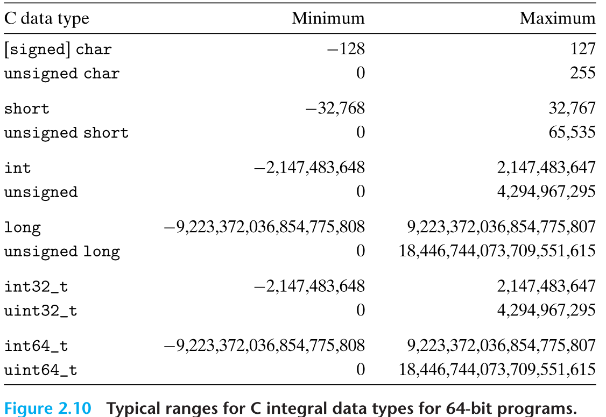
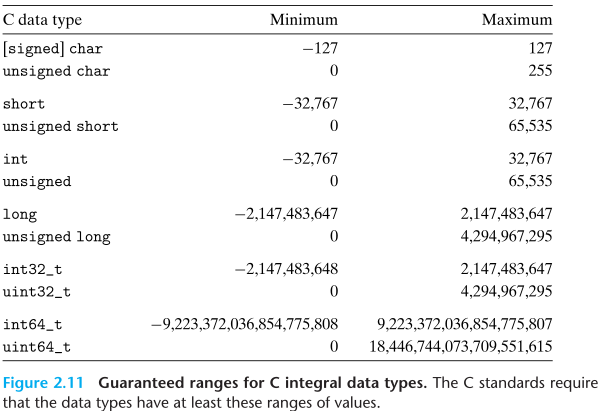
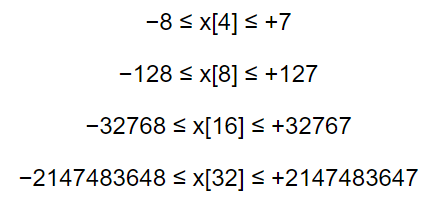
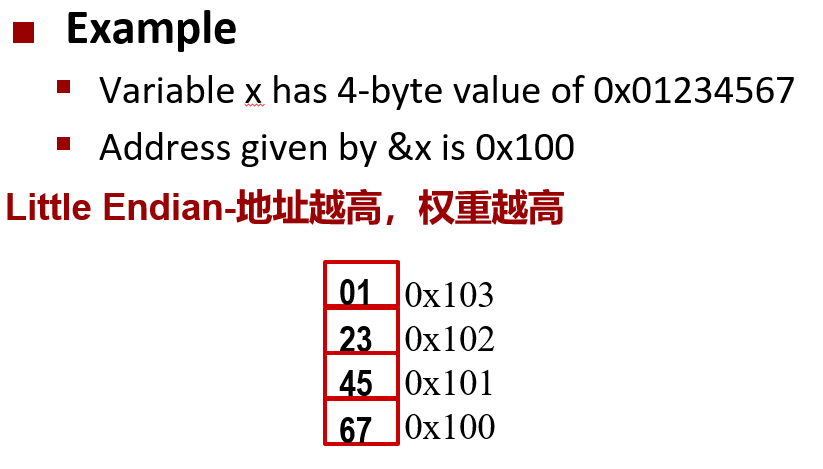
## 2 信息的表示和处理

字数据大小

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C声明 | | 字节数-以字节为单位 | | |
| 有符号 | 无符号 | 32位 | 64位 | X86-64 |
| [signed] char | unsigned char | 1 | 1 | 1 |
| short | unsigned short | 2 | 2 | 2 |
| int | unsigned | 4 | 4 | 4 |
| long | unsigned long | 4 | 8 | 8 |
| int32\_t | uint32\_t | 4 | 4 |  |
| int64\_t | uint64\_t | 8 | 8 |  |
| char\*-指针类型 |  | 4 | 8 | 8 |
| float |  | 4 | 4 | 4 |
| double |  | 8 | 8 | 8 |
| long double |  |  |  | 10/16 |

字长（word size）-指明指针数据的标称大小（nominal size）。字长决定的最重要的系统参数为虚拟地址空间的最大大小。字长为w位，则虚拟地址范围为0~2w-1。

字节序

书写字节序的最自然方式为最低位字节在左边，最高位字节在右边。

小头端实验见byteordering.c

Size\_t的取值范围为目标平台下最大可能的数组尺寸，不同于unsigned int.

使用ASCII码作为字符码的任何系统上都将得到相同的结果，与字节顺序和字大小规则无关。文本数据在大小端平台上显示一样。

计算机系统的一个基本概念：从机器的角度来看，程序仅仅是字节序列。

整数和浮点数及相关运算

对于计算机，常见的四则运算并不一定成立：

平方大于等于0-对于float成立，

对有符号整型不一定，对于无符号整型成立（无符号整型总是大于0）

加法的结合律-对于有/无符号整型成立，对于float不一定

布尔操作

对于二进制和数值进行布尔运算是不同的，且两者的操作符亦不同。

二进制布尔运算操作符及对应的集合操作

& AND 全1则1 集合的交集-intersection

| OR 存在1则1 集合的并集-union

~ NOT 若1则0，若0则1 集合的补集-complement

^ XOR(exclusive-OR) 同0异1 集合的对称差异（symmetric difference)

此些布尔运算符将按 位 进行操作，应用于任意“整型” long int short char unsigned

数值布尔运算操作符

&& AND

|| OR

！取反

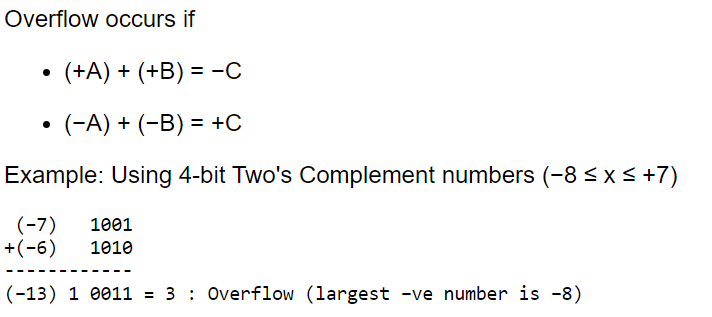
此些布尔运算符将：0为假，非零为真；只返回0或1；提前中断

整数溢出-乘法和除法溢出待续

加法溢出-两个补码相加，两者符号相同（同正或同负），当且仅当结果有相反符号时会发生溢出（实际为两者数值较大，相加时造成溢出-符号位进位，溢出后使得原有补码符号位取反）。当两个不同补码符号位的数值相加不会发生溢出。

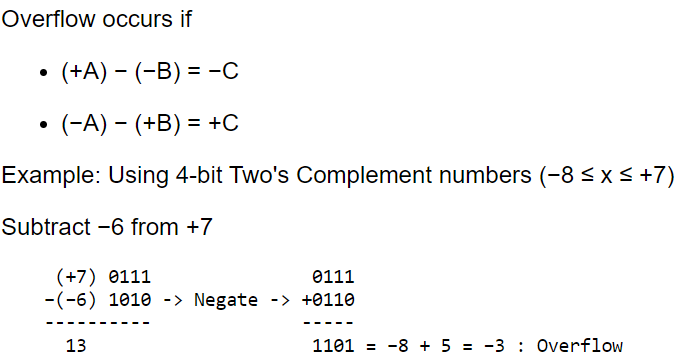
加法溢出时：

两个正数相加必产生负数

 两个负数相加必产生正数。

减法溢出

612-485=127；612为被减数（minuhend）,485为减数（subtrahend），127为结果（result）

 两个符号相反的补码相减，当且仅当结果符号与减数相同时发生溢出

左移和右移

左移-只存在一种，即逻辑右移（在右边添加0）。

右移-逻辑右移（左侧添加0）和算术右移（左侧添加符号位）

C语言对右移操作未进行明确定义。几乎所有编译器/机器组合都对有符号数使用算术右移，且许多程序员假设机器会使用这种右移。对于无符号数，右移必须为逻辑右移。