C语言标准

C90

● 信任程序员；● 不要妨碍程序员做需要做的事；● 保持语言精练简单；● 只提供一种方法执行一项操作；● 让程序运行更快，即使不能保证其可移植性。(作为实现，应该针对目标计算机来定义最合适的某特定操作，而不是强加一个抽象、统一的定义。)

C99

国际化、弥补缺陷和提高计算的实用性

C11

C是编译型语言

计算机系统是由硬件和系统软件组成的，它们共同工作来运行应用程序。

ASCII标准

h e 1 1 o 程 序 的 生 命 周 期 是 从 一 个 源 程 序 ( 或 者 说 源 文 件 )开 始 的 ， 即 程 序 员 通 过 编 辑 器 创 建 并 保 存 的 文 本 文 件 ， 文 件 名 是 h e 1 1 o . c 。 源 程 序 实 际 上就 是 一 个 由 值 。和 1 组 成 的 位 ( 又 称 为比特)序列，8 个位被组织成一组，称为宇节。每个字节表示程序中的某些文本字符。

文本文件，二进制文件

系统中所有的信息—包括磁盘文件、内 存中的程序、内存中存放的用户数据以及网络上传送的数据，都是由一串比特表示的。区分不同数据对象的唯一方法就是我们读到这些数据对象时的上下文。

可执行目标程序，可执行目标文件

编译系统

源文件—》预处理器（cpp）--》编译器（ccl）--》汇编器（as） --》链接器（ld）

1. // hello.c
2. #include <stdio.h>
4. **int** main()
5. {
6. printf("Hello World\n");
7. **return** 0;
8. }

计算机将信息编码为位/比特(bit)，通常组织成字节(byte)序列。

1 bit表示一个二进制数字（0或1），1 byte = 8 bit，是最小的可寻址的内存单位。

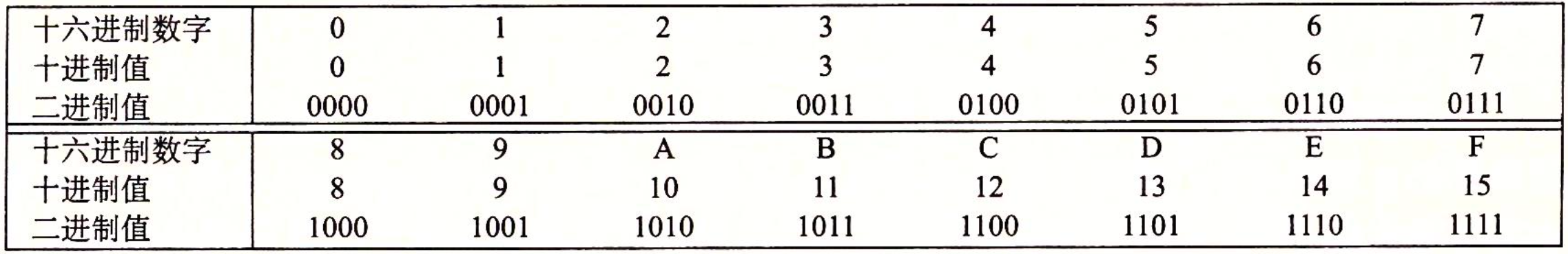
机器级程序将内存视为一个非常大的字节数组，称为虚拟内存(virtual memory)。

内存中的每个字节都由一个唯一的数字来标识，称为它的地址(address)，所有可能地址的集合称为虚拟地址空间(virtual address space)。

程序对象(program object)存放在存储器空间中，包括程序数据、指令和控制信息。每个程序对象可以简单地视为一个字节块，程序本身就是一个字节序列。

计算机都有一个字长(word size)，指明指针数据的标称大小(nominal size)。虚拟地址是以一个字来编码的，字长决定了虚拟地址空间的最大大小，一个字长为位的机器，虚拟地址的范围是，程序最多访问个字节。32位的虚拟地址空间约4GB，而64位的虚拟地址空间约为16EB。

十六进制(hexadecimal，简写hex)数



C语言中，以0x或0X开头的数字常量被认为是十六进制数，例如0xFA1D37B。

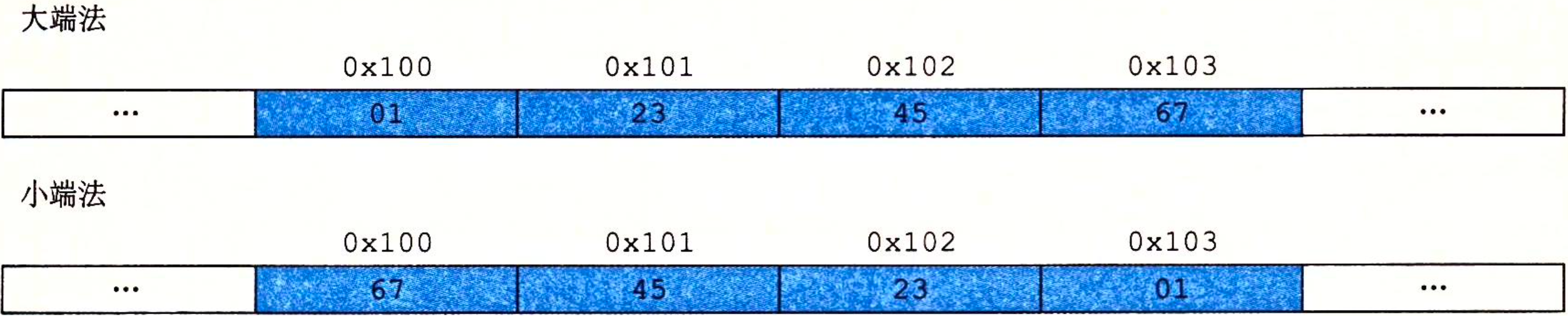
C语言中，整数、指针和浮点数等数据类型。



对于跨越多字节的程序对象，需要建立两个规则：对象的地址是什么、内存中如何排列这些字节。

多字节对象通常被存储为连续的字节序列，对象的地址为所使用字节中最小的地址。以int类型变量为例，假设的值为0x100，那么的4个字节将被存储在0x100、0x101、0x102、0x103位置。

排列表示一个对象的字节有两个通用的规则：小端法(little endian)，最低有效字节在最前面的方式；大端法(big endian)，最高有效字节在最前面的方式。在0x100 ~ 0x103的字节排序：



信息存储

位/比特(bit)，一个二进制数字，0或者1。

数字表示：

-无符号（unsigned）编码，基于传统的二进制表示法，表示大于或者等于0的数字；

-补码（two’s-complement）编码，表示有符号整数；

-浮点数（floating-point）编码，表示实数的科学记数法的以2为基数的版本。

溢出（overflow）

整数的表示虽然只能编码 一个相对较小的数值范围，但是这种表示是精确的;而浮点数虽然可以编码一个较大的数值范围，但是这种表示只是近似的。

字节（byte）=8个bit，是最小的可寻址的内存单位。

机器级程序将内存视为一个非常大的字节数组，称为虚拟内存（virtual memory）。

程 序 对 象 ( p r o g t a m o b j e c t )， 即 程 序 数 据 、 指 令 和 控 制 信 息 。

小结

计算机将信息编码为位( 比特)，通常组织成字节序列。有不同的编码方式用来表示整数、实数和宁 符串。不同的计算机模型在编码数字和多字节数据中的字节顾序时使用不同的约定。 C语言的设计可以包容多种不同字长和数字编码的实现。64 位字长的机器逐渐普及，并正在取代统治 市场长达 30 多年的 32 位机器。由于64 位机器也可以运行为 32 位机器编译的程序，我们的重点就放在区 分32位和64位程序，而不是机器本身。64位程序的优势是可以突破32位程序具有的4GB地址限制。 大多数机器对整数使用补码编码，而对浮点数使用IEEE 标准 754 编码。在位级上理解这些编码，并 且理解算术运算的数学特性，对于想使编写的程序能在全部数值范围上正确运算的程序员来说，是很重要的。 在相同长度的无符号和有符号整数之间进行强制类型转换时，大多数C 语言实现遵循的原则是底层 的 位 模 式 不 变 。 在 补 码 机 器 上 ， 对 于一 个 w 位 的 值 ， 这 种 行 为 是 由 两 数 T 2 U 。 和 U 2 丁。 来 描 述 的 。 C 语 言隐式的强制类型转换会出现许多程序员无法预计的结果，常常导致程序错误。 由于编码的长度有限，与传统整数和实数运算相比，计算机运算具有非常不同的屈性。当超出表示 范围时，有限长度能够引起数值滥出。当浮点数非常接近手0. 0，从而转换成零时，也会下溢。 和大多数其他程序语言一样，C 语言实现的有限整数运算和真实的整数运算相比，有一些特殊的属 性 。 例 如 ， 由 于溢 出 ， 表 达 式 x x x 能 够 得 出 负 数 。 但 是 ， 无 符 号 数 和 补 码 的 运 算 都 满 足 整 数 运 算 的 许 多 其他属性，包括结合律、交换律和分配律。这就允许编译器做很多的优化。例如，用(x<<3)- ×取代表达 式7\*x 时，我们就利用 了结合律、交换律和分配律的属性，还利用了移位和乘以2 的我之间的关系。 我们已经看到了几种使用位级运算和算术运算组合的聪明方法。例如，使用补码运算，~ x+ 1 等价 于 - x。另外一个例子，假设我们想要一个形妍[0，...，0，1，...，1]的位模式，由w一k 个。后面祭跟着人个1组成。这些位模式有助于掩码运算。这种模式能够通过C 表达式(1<<k)- 1生成，利用的是这样一个 属性，即我们想要的位模式的数值为2\*一1。例如，表达式(1<<8)- 1将产生位模式OxFF。 浮点表示通过將数字编码为z×2° 的形式来近似地表示实数。最常见的浮点表示方式是由IEEE枋 淮754定义的。它提供了几种不同的精度，最常见的是单精度(32 位)和双精度(64位)。IEEE浮点也能 够 表 示 特 殊 值 十∞ 、 一 ∞ 和 N a N 。 必须非常小心地使用浮点运算，因为浮点运算只有有限的范国和精度，而且并不遊守普避的算术屈 性，比如结合性。