


**旁注 十进制和十六进制间的转换**

较大数值的十进制和十六进制之间的转换，最好是让计算机或者计算器来完成。有大量的工具可以完成这个工作。一个简单的方法就是利用任何标准的搜索引擎，比如查询：

把 0xabcd 转换为十进制数

或

把 123 用十六进制表示。

 **练习题 2.4** 不将数字转换为十进制或者二进制，试着解答下面的算术题，答案要用十六进制表示。提示：只要将执行十进制加法和减法所使用的方法改成以 16 为基数。

- A.  $0x503c + 0x8 =$
- B.  $0x503c - 0x40 =$
- C.  $0x503c + 64 =$
- D.  $0x50ea - 0x503c =$

## 2.1.2 字数据大小

每台计算机都有一个字长(word size)，指明指针数据的标称大小(nominal size)。因为虚拟地址是以这样的一个字来编码的，所以字长决定的最重要的系统参数就是虚拟地址空间的最大大小。也就是说，对于一个字长为  $w$  位的机器而言，虚拟地址的范围为  $0 \sim 2^w - 1$ ，程序最多访问  $2^w$  个字节。

最近这些年，出现了大规模的从 32 位字长机器到 64 位字长机器的迁移。这种情况首先出现在为大型科学和数据库应用设计的高端机器上，之后是台式机和笔记本电脑，最近则出现在智能手机的处理器上。32 位字长限制虚拟地址空间为 4 千兆字节(写作 4GB)，也就是说，刚刚超过  $4 \times 10^9$  字节。扩展到 64 位字长使得虚拟地址空间为 16EB，大约是  $1.84 \times 10^{19}$  字节。

大多数 64 位机器也可以运行 32 位机器编译的程序，这是一种向后兼容。因此，举例来说，当程序 prog.c 用如下伪指令编译后

```
linux> gcc -m32 prog.c
```

该程序就可以在 32 位或 64 位机器上正确运行。另一方面，若程序用下述伪指令编译

```
linux> gcc -m64 prog.c
```

那就只能在 64 位机器上运行。因此，我们将程序称为“32 位程序”或“64 位程序”时，区别在于该程序是如何编译的，而不是其运行的机器类型。

计算机和编译器支持多种不同方式编码的数字格式，如不同长度的整数和浮点数。比如，许多机器都有处理单个字节的指令，也有处理表示为 2 字节、4 字节或者 8 字节整数的指令，还有些指令支持表示为 4 字节和 8 字节的浮点数。

C 语言支持整数和浮点数的多种数据格式。图 2-3 展示了为 C 语言各种数据类

C 声明		字节数	
有符号	无符号	32 位	64 位
[signed] char	unsigned char	1	1
short	unsigned short	2	2
int	unsigned	4	4
long	unsigned long	4	8
int32_t	uint32_t	4	4
int64_t	uint64_t	8	8
char *		4	8
float		4	4
double		8	8

图 2-3 基本 C 数据类型的典型大小(以字节为单位)。分配的字节数受程序是如何编译的影响而变化。本图给出的是 32 位和 64 位程序的典型值