计算机系统与云计算

1. 信息表示 - 整数 cont'd

华东师范大学 数据科学与工程学院

2021年09月10日

钱卫宁

wnqian@dase.ecnu.edu.cn

符号扩展

w 位的有符号整数 X，转换为 $w+k$ 位的整数（**要求值不变**）

- 最高位复制 k 次：$X = x_{w-1}^{0}, \ldots, x_{w-1}^{k-1}, x_{w-1}, x_{w-2}, \ldots, x_0$
- $x_{w-1}^{i} = x_{w-1}$

为什么成立？？？

符号扩展

“小”整数类型转换为“大”整数类型

```
short int x = 15213;
int ix = (int) x;
short int y = -15213;
int iy = (int) y;
```

C语言此时会自动进行符号扩展

截断（truncation）

$w + k$ 位的有符号或无符号整数 X，转换为 w 位的整数（**要求对于“在范围内的X”，值不变**）

- 扔掉最高的 k 位

截断示例

截断不变号

- $2 = 00010 \rightarrow 0010\colon\ 2 \mod 16 = 2$
- $-6 = 11010 \rightarrow 1010\colon\ -6 \mod 16 = 26U \mod 16 = 10U = -6$

截断示例

截断变号

- $10 = 01010 \rightarrow -6 = 1010$: $10 \mod 16 = 10U \mod 16 = 10U = -6$
- $-10 = 10110 \rightarrow 6 = 0110$: $-10 \mod 16 = 22U \mod 16 = 6U = 6$

扩展与截断小结

扩展（e.g. short $\rightarrow$ int）

- 无符号：加 0
- 有符号：最高位扩展
- 结果可靠（可预期）

扩展与截断小结

截断（e.g. unsigned to unsigned short）

- 无论有无符号：去掉高位，结果重新解释
- 无符号：等同于取模运算
- 有符号：类似于取模运算
- 对于不太大的数，结果可靠（可预期）

In the Game of Civilization

https://www.zhihu.com/question/24830939/answer/29131258

加法

$$s = UADD_w(u, v) = u + v \mod 2^w$$

$$s = TAdd_w(u, v) =$$

- $u + v + 2^w,\ u + v < TMin_w$（负溢出）
- $u + v,\ TMin_w \leq u + v \leq TMax_w$
- $u + v - 2^w,\ TMax_w < u + v$（正溢出）

加法（无符号）

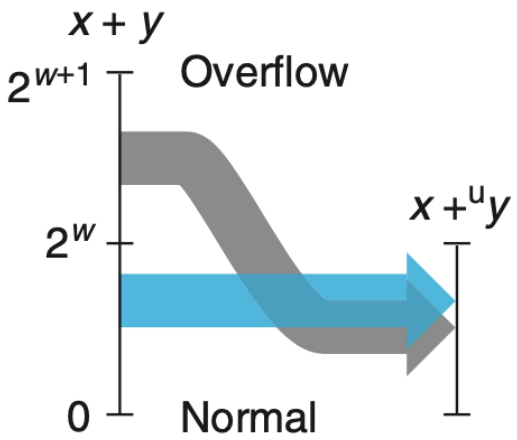
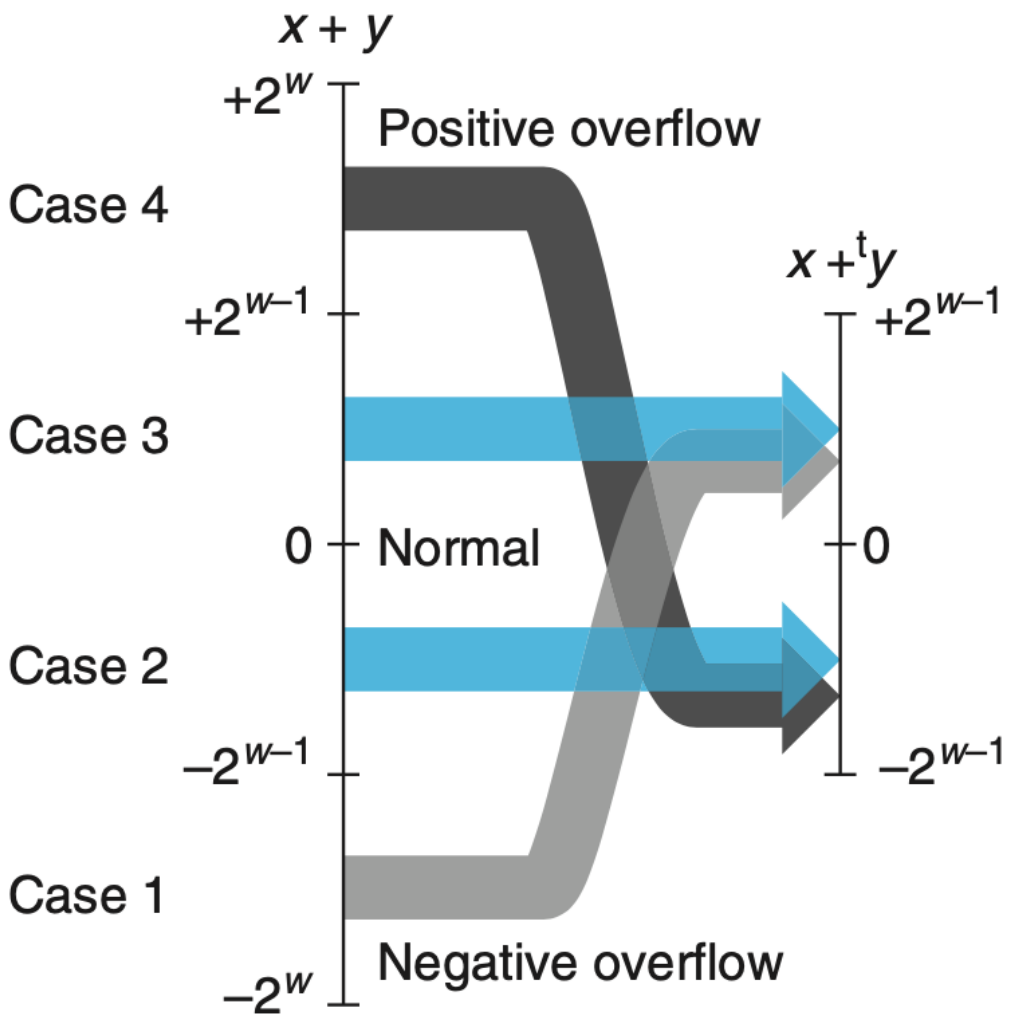


Figure 2.22 Relation between integer addition and unsigned addition. When $x + y$ is greater than $2^w - 1$, the sum overflows.

加法（补码）

Figure 2.24

Relation between integer and two's-complement addition. When $x + y$ is less than -2^{w-1}, there is a negative overflow. When it is greater than or equal to 2^{w-1}, there is a positive overflow.



加法图示（普通加法）

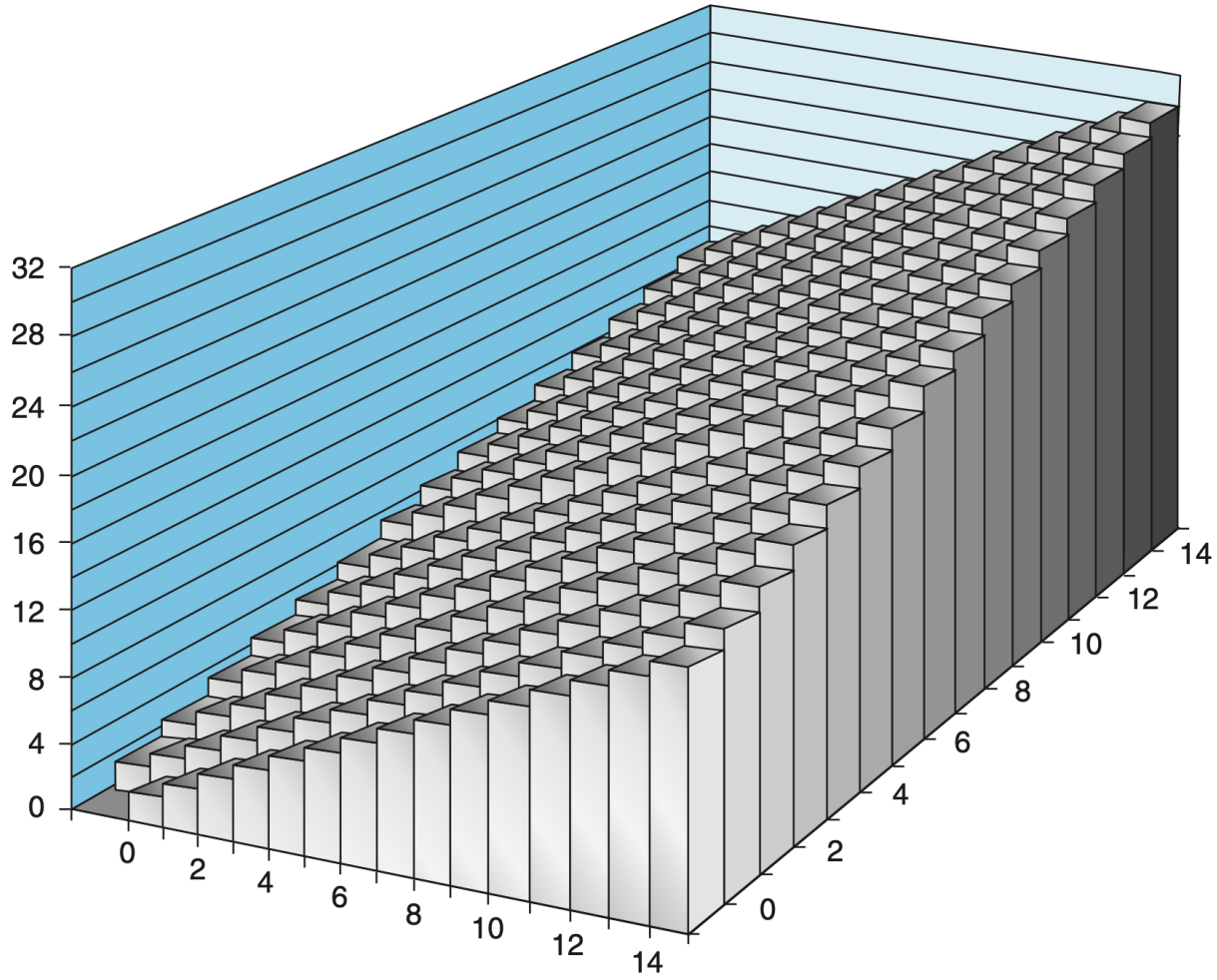


Figure 2.21 **Integer addition.** With a 4-bit word size, the sum could require 5 bits.

加法图示（unsigned）

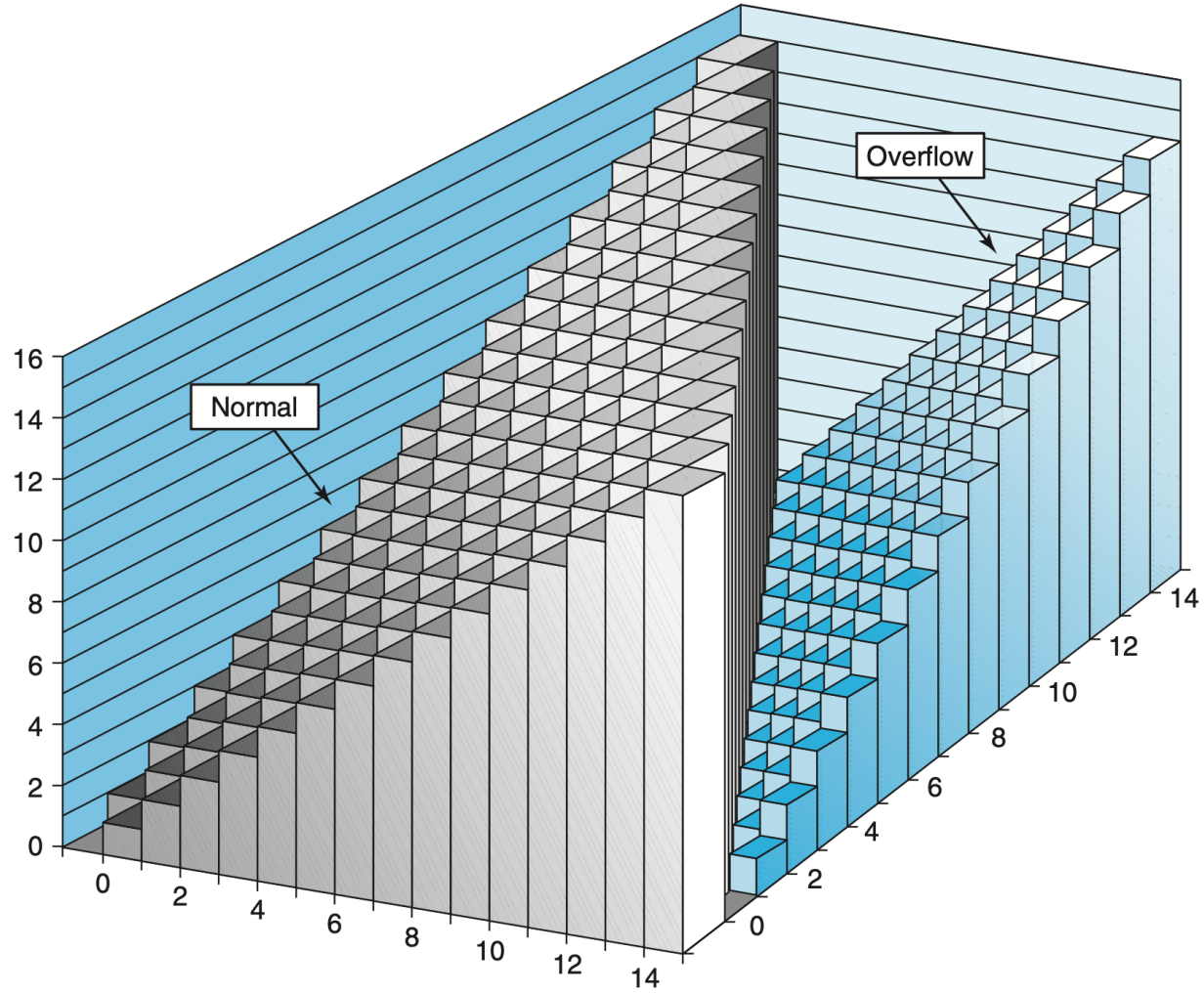


Figure 2.23 **Unsigned addition.** With a 4-bit word size, addition is performed modulo 16.

加法图示（有符号）

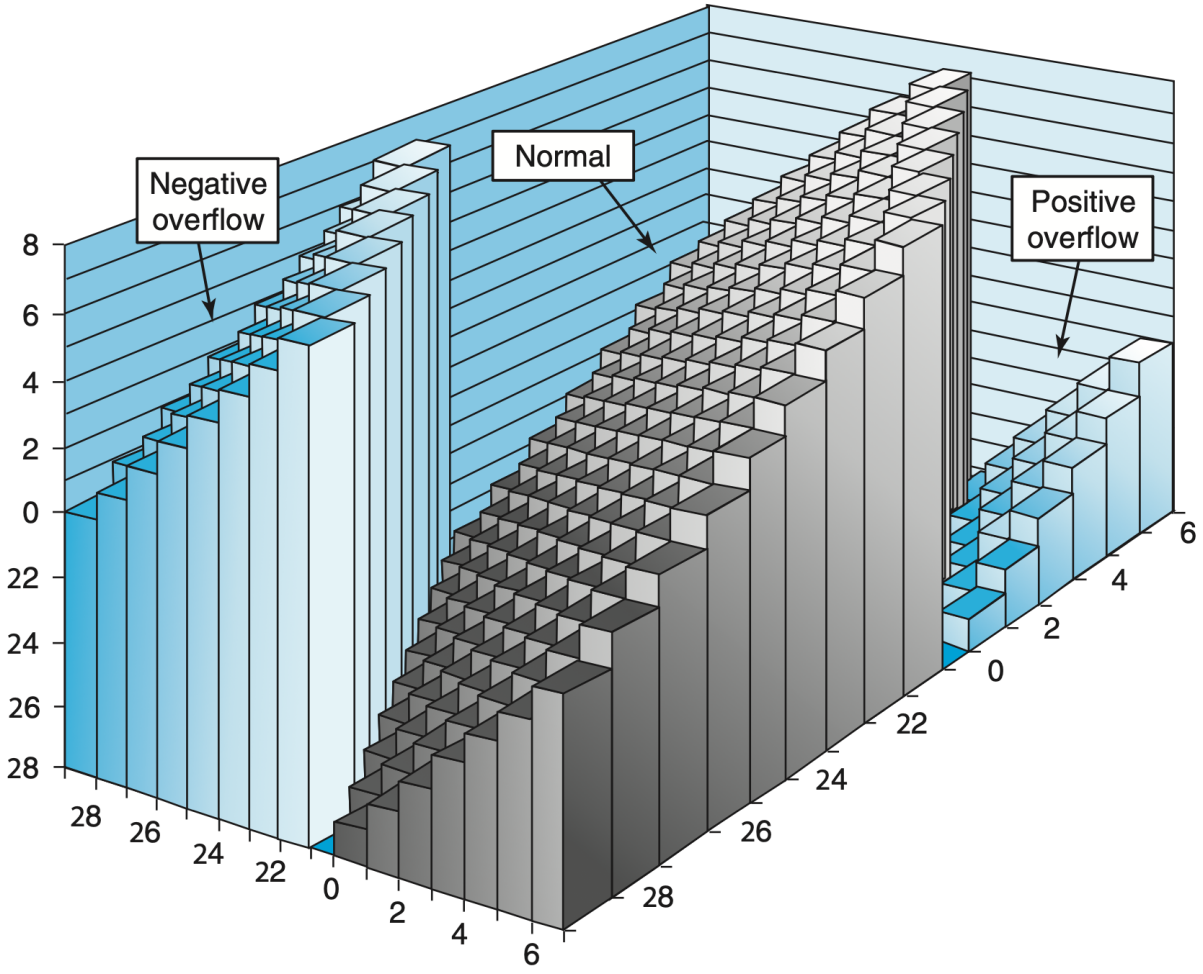


Figure 2.26 **Two's-complement addition.** With a 4-bit word size, addition can have a negative overflow when $x + y < -8$ and a positive overflow when $x + y \geq 8$.

$U\,Add$ vs $T\,Add$

位操作相同

- $x +_w^t y = U2T_w(T2U_w(x) +_w^u T2U_w(y))$

```
int s, t, u, v;
s = (int) ((unsigned) u + (unsigned) v);
t = u + v;

s==t
```

例子： $11101001 + 11010101$

乘法

w 位 x, y 相乘，结果（很可能）超过 w 位

- 无符号：最多 $2w$ 位：
  $0 \leq x * y \leq (2^w - 1)^2 = 2^{2w} - 2^{w+1} + 1$
- 补码最小值：最多 $2w - 1$ 位：
  $x * y \geq (-2^{w-1}) * (2^{w-1} - 1) = -2^{2w-2} + 2^{w-1}$
- 补码最大值：最多 $2w$ 位（当 $(TMin_w)^2$ 时达到）：
  $x * y \leq (-2^{w-1})^2 = 2^{2w-2}$

如需要保留“完整”结果，需要扩展字长。如需要，由软件实现。

乘法

$$UMult_w(u, v) = u \cdot v \mod 2^w$$

$$TMult_w(u, v) = U2T_w((u \cdot v) \mod 2^w)$$

例子： $11101001 * 11010101$

乘法示例

Mode	x		y		$x \cdot y$		Truncated $x \cdot y$	
Unsigned	5	[101]	3	[011]	15	[001111]	7	[111]
Two's complement	−3	[101]	3	[011]	−9	[110111]	−1	[111]
Unsigned	4	[100]	7	[111]	28	[011100]	4	[100]
Two's complement	−4	[100]	−1	[111]	4	[000100]	−4	[100]
Unsigned	3	[011]	3	[011]	9	[001001]	1	[001]
Two's complement	3	[011]	3	[011]	9	[001001]	1	[001]

Figure 2.27 Three-bit unsigned and two's-complement multiplication examples. Although the bit-level representations of the full products may differ, those of the truncated products are identical.

移位：乘以 2 的幂

$$u << k = u * 2^k$$

- $u << 3 = u * 8$
- $(u << 5) - (u << 3) = u * 24$

多数计算的移位操作快于乘法，编译器会自动用移位替换乘法

移位：除以 2 的幂

无符号数：$u >> k = u/2^k$

- 无符号数的右移是**逻辑右移**

有符号数：$x >> k = x/2^k$

- 有符号数的右移是**算术右移**

注意：有符号数这样做除法以后，正数和负数的舍入方向是不同的

$x = -15213, y = 15213$，$x >> 4$ **和** $y >> 4$ **分别等于多少?**

向上舍入

$$(x + 2^k - 1)/2^k$$

```
(x+(1<<k)-1) >> k
```

- 如果无舍入：$2^k - 1$ 正好被右移去掉
- 负数，有舍入：最后结果增加 1（由向 $-\infty$ 变为向 0）

向 0 舍入

```
(x<0 ? x+(1<<k)-1 : x) >> k
```

相反数

$$-x ==\sim x + 1$$

运算小结

加法

- 有符号和无符号：普通运算，然后截断高位，**两者在位表示级别完全相同**

乘法

- 有符号和无符号：普通运算，然后截断高位，**两者在位表示级别完全相同**

关于无符号数

错：

```
unsigned i;
for (i = cnt-2; i >= 0; i--)
  a[i] += a[i+1];
```

关于无符号数

正确：

```
size_t i;
for (i = cnt-2; i < cnt; i--)
  a[i] += a[i+1];
```

$$0 - 1 \rightarrow UMax$$

Robert Seacord, Secure Coding in C and C++ (2nd Ed.). Addison-Wesley 2013

Why should I use unsigned?

- Do Use When Performing Modular Arithmetic
  - Multiprecision arithmetic
- Do Use When Using Bits to Represent Sets
  - Logical right shift, no sign extension
- Do Use In System Programming
  - Bit masks, device commands,...

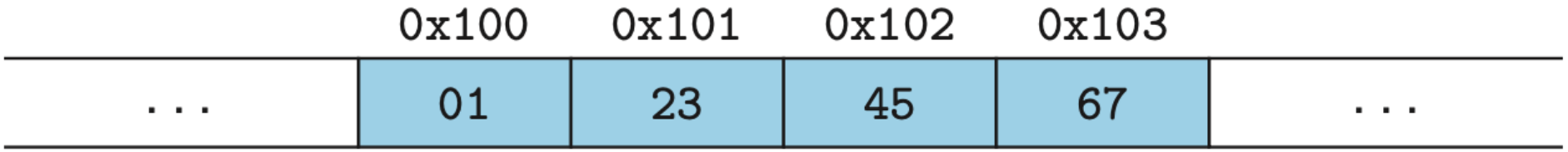
数据表示的重要概念

- 内存（memory），地址空间（address space）
- 字（word），字长（word size）
- 地址（address，字的低位）
- 字节序（byte ordering）
  - big endian：Sun (Oracle SPARC), PPC Mac, Internet
  - little endian：x86, ARM processors running Android, iOS, and Linux

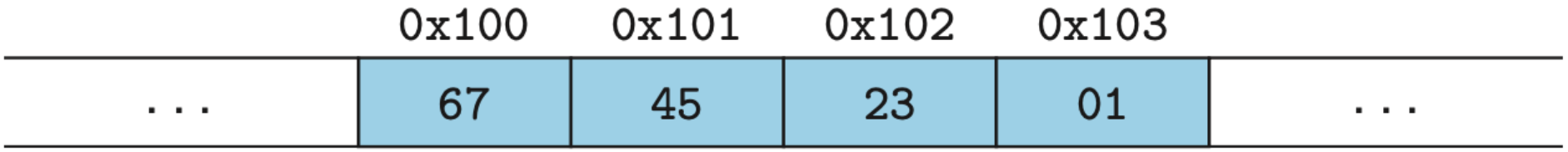
Byte ordering

0x01234567

Big endian



Little endian



Byte ordering

你能写一个显示计算机上字节序的程序（函数）么？

```
void show_bytes(pointer start, size_t len);
```

字符串

字符数组，以 ‘\0’ 结尾

字符以 ASCII 编码

```
> man ascii
```

与字节序无关

汇编中的逆序字节列表

Address	Instruction Code	Assembly Rendition
8048365:	5b	pop %ebx
8048366:	81 c3 ab 12 00 00	add $0x12ab,%ebx
804836c:	83 bb 28 00 00 00 00	cmpl $0x0,0x28(%ebx)

预习要求

阅读至2.4结束

抽时间仔细/反复阅读第一章