

计算机系统

1. 信息表示

华东师范大学 数据科学与工程学院

2023年09月13日

钱卫宁，赵明昊

{wnqian,mhzhao}@dase.ecnu.edu.cn

位 (bit) : “数字化”的最小单位

0 或者 1

可以（通过组合）来表示数字，集合，字符串，操作（指令）...

用电子元器件实现：可用双稳态元器件实现，可在有噪声的不准确线路上可靠传输

计数

以2为基的数字表示

- $15213_{10} = 11101101101101_2$
- $1.20_{10} = 1.0011001100110011[0011] \dots_2$
- $1.5213 \times 10^4 = 1.11011011011012 \times 2^{13}$

十进制 \Rightarrow 二进制：整数部分：除二取余，顺序排列

十进制 \Rightarrow 二进制：小数部分：乘二取整，顺序排列

字节 (byte)

1 byte = 8 bits

二进制: $00000000_2 \sim 11111111_2$

十进制: $0_{10} \sim 255_{10}$

十六进制: $00_{16} \sim FF_{16}$

($FA1D37B_{16}$ C 语言表示: $0xFA1D37B$ 或者 $0xfa1d37b$)

二进制与十六进制的转换?

典型的数据类型及其表示位数

C数据类型	32位机器	64位机器	x86-64
char	1	1	1
short	2	2	2
int	4	4	4
long	4	8	8
float	4	4	4
double	8	8	8
指针	4	8	8

位操作：布尔代数 (Boolean Algebra)

逻辑的代数表示方法 (True: 1, False: 0)

- 按位与 (and) : $A \& B = 1$ 当且仅当 $A = 1$ 且 $B = 1$;
- 按位或 (or) : $A | B = 1$ 当且仅当 $A = 1$ 或者 $B = 1$;
- 按位非 (not) : $\sim A = 1$ 当且仅当 $A = 0$;
- 按位异或 (xor) : $A \wedge B = 1$ 当且仅当 $A = 1$ 或 $B = 1$ 但两者不同时为1。

通用布尔代数

位向量操作

- $01101001 \& 01010101 = 01000001$
- $01101001 | 01010101 = \text{????????}$
- $01101001 \wedge 01010101 = \text{????????}$
- $\sim 01010101 = \text{????????}$

表示集合

宽度为 w 的位向量可以表示 $\{0, \dots, w - 1\}$ 的子集

$$(a_i = 1 \text{ 当且仅当 } i \in A)$$

- $\&$: \cap
- $|$: \cup
- \wedge : 对称差
- \sim : 补

C 语言中的按位操作

可作用于所有整数数据类型：int, long, short, char, unsigned

- $\sim 0x41 = ?$
- $\sim 0x00 = ?$
- $0x69 \& 0x55 = ?$
- $0x69 | 0x55 = ?$

C 语言中的逻辑操作

与，或，非 (&&, ||, !)

- `!0x41 = ?`
- `!0x00 = ?`
- `!!0x41 = ?`
- `0x69 && 0x55 = ?`
- `0x69 || 0x55 = ?`
- `p && *p = ?`
- `p && *p++ = ?`

注意 C 语言中的 **early termination (lazy)**

移位操作

左移: $x \ll y$

右移: $x \gg y$

- 逻辑 (logical) 右移: 左侧补0
- 算术 (arithmetic) 右移: 左侧补最高位 (significant bit)

如果 $y < 0$ 或者 $y >$ 字长, 则行为无定义

整数

回顾我们学习算术的过程

(数字, 进制/整数, 四则运算, 有理数, 实数, ...)

整数

- 无符号整数 (unsigned)

$$B2U(X) = \sum_{i=0}^{w-1} x_i \times 2^i$$

- 有符号整数——基为 2 的补码 (two's complement)，或简称为“补码”

$$B2T(X) = -x_{w-1} \times 2^{w-1} + \sum_{i=0}^{w-2} x_i \times 2^i$$

整数

C 语言标准并不要求一定使用补码表示有符号整数

但是几乎所有机器上的实现都用补码

问题：请写出：15213 和 -15213 的补码形式

问题：如何判断一个补码表示的整数的正负？

动手练习

- $B2T(10) = ?$
- $B2T(-10) = ?$
- $B2T(15213) = ?$
- $B2T(-15213) = ?$

取值范围

- 无符号整数

$$UMin = 0 \quad (000 \dots 0)$$

$$UMax = 2^w - 1 \quad (111 \dots 1)$$

取值范围

- 补码

$$TMin = -2^{w-1} \quad (100 \dots 0)$$

$$TMax = 2^{w-1} - 1 \quad (011 \dots 1)$$

$$-1 \quad (111 \dots 1)$$

取值范围

$$|TMin| = TMax + 1$$

$$UMax = 2 * TMax + 1$$

```
#include <limits.h>
ULONG_MAX
LONG_MAX
LONG_MIN
```

关于无符号和有符号数补码表示的说明

- 非负数表示相同
- 每一种位组合表示不同的数（比如，不会存在+0和-0）
- 每一种可表示的整数有唯一的位串编码（当然如此！）
- 可逆

$$U2B(x) = B2U^{-1}(x)$$

$$T2B(x) = B2T^{-1}(x)$$

类型转换

补码 \rightarrow 无符号数

$$ux = B2U[T2B(x)]$$

无符号数 \rightarrow 补码

$$x = B2T[U2B(ux)]$$

保持位串不变，重新解释

类型转换

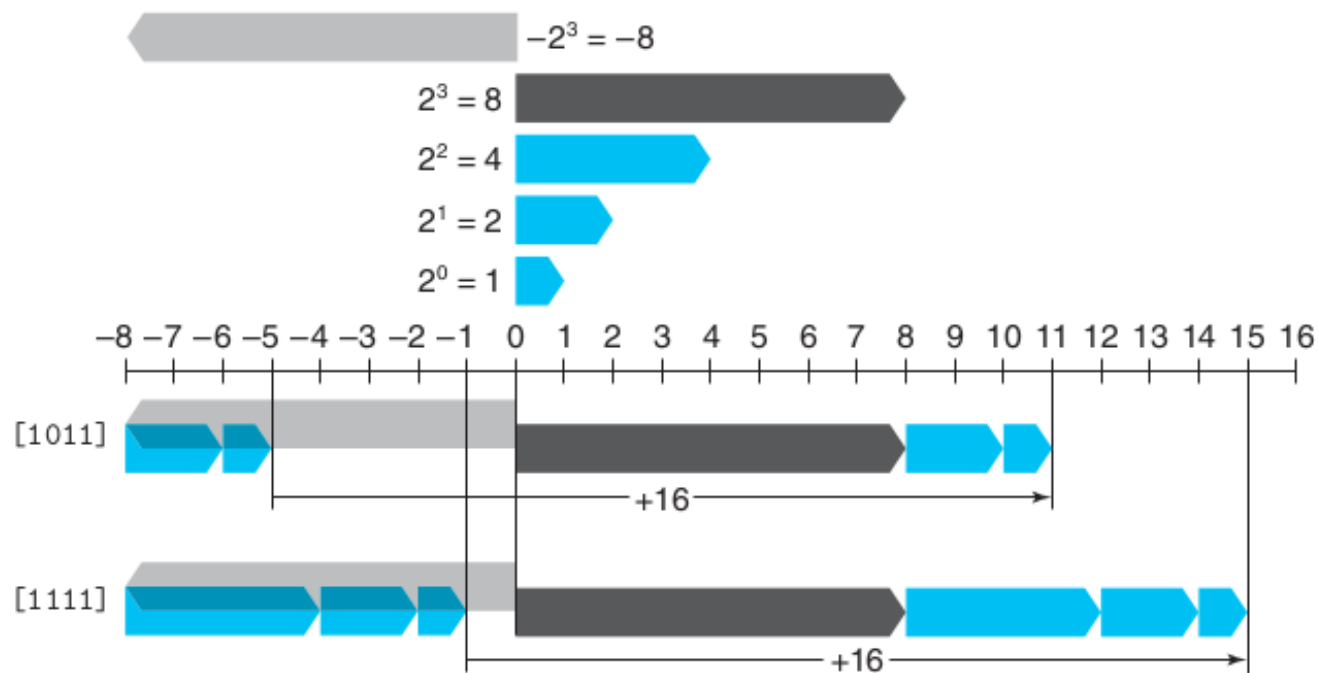


Figure 2.15 Comparing unsigned and two's-complement representations for $w = 4$. The weight of the most significant bit is -8 for two's complement, and $+8$ for unsigned, yielding a net difference of 16.

类型转换

Figure 2.16

Conversion from two's complement to unsigned.

Function *T2U* converts negative numbers to large positive numbers.

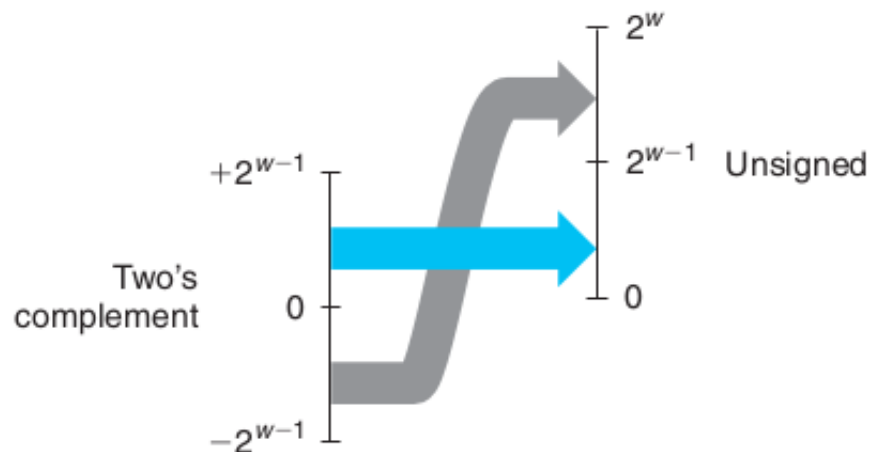
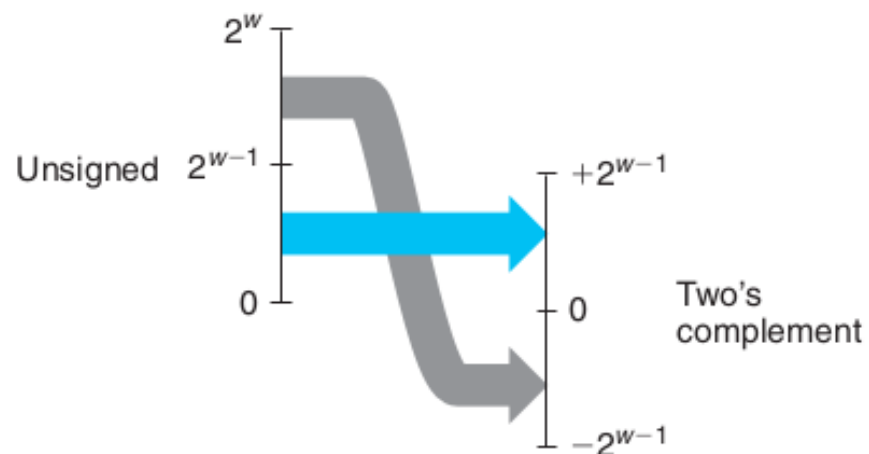


Figure 2.17

Conversion from unsigned to two's complement.

Function *U2T* converts numbers greater than $2^{w-1} - 1$ to negative values.



C 语言中的有符号和无符号整数

常数

- 如果无特别说明，为有符号数
- 后缀'U'表示无符号数，如0U，4294967259U

C 语言中的有符号和无符号整数

类型转换

- 显式类型转换：

```
int tx, ty;  
unsigned ux, uy;  
tx = (int) ux;  
uy = (unsigned) ty;
```


C 语言中的有符号和无符号整数

类型转换

- 隐式类型转换：

```
tx = ux;  
uy = ty;  
int fun(unsigned u);  
uy = fun(tx);
```

C 语言类型转换

表达式计算

- 一个表达式中同时有有符号和无符号数，有符号数将被隐式转换为无符号数
- 不仅四则运算、位操作如此，关系判断运算：<, >, ==, <=, >=也遵守此规则

$$W = 32$$

$$TMIN = -2,147,483,648; TMAX = 2,147,483,647$$

Constant ₁	Constant ₂	关系运算	计算时数据类型
-1	0U	>	unsigned
2147483647	-2147483647-1	>	signed
2147483647U	-2147483647-1	<	unsigned
-1	-2	>	signed
(unsigned)-1	-2	>	unsigned
2147483647	2147483648U	<	unsigned
2147483647	(int) 2147483648U	>	signed

易错的隐式转换

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

```
#define DELTA sizeof(int)  
int i;  
for (i = CNT; i-DELTA >= 0; i -= DELTA)  
    ...
```

有符号/无符号转换小结

位串保持不变

位串重新解读

可能导致意想不到的结果： $+/- 2^w$

同时含有符号和无符号整数的表达式，有符号数被转换为无符号数进行运算

预习要求

阅读至2.3结束

抽时间仔细/反复阅读第一章