整数与无符号整数之间的转换

无符号整数

- 某些应用不出现负数，整型数中有一半的数值范围被浪费。

无符号数：在各整数类型前加上关键词 unsigned

unsigned int	$0 \sim 2^{32}-1$
unsigned short	$0 \sim 2^{16}-1$ （65535）
unsigned long	$0 \sim 2^{32}-1$ (32 位机器)

无符号数的编码

长度为n+1的编码

$x = x_0x_1x_2\cdots x_n \qquad x_i=\{0,1\},\ 0\leqslant i\leqslant n$

数值表示:

$$x_0 2^n + x_1 2^{n-1} + \cdots + x_{n-1} 2^1 + x_n$$

数值范围: $0\leqslant x\leqslant 2^{n+1}-1$

模: 2^{n+1}

Signed vs. Unsigned

- 显式转换
  - **int tx, ty;**

    unsigned ux, uy;

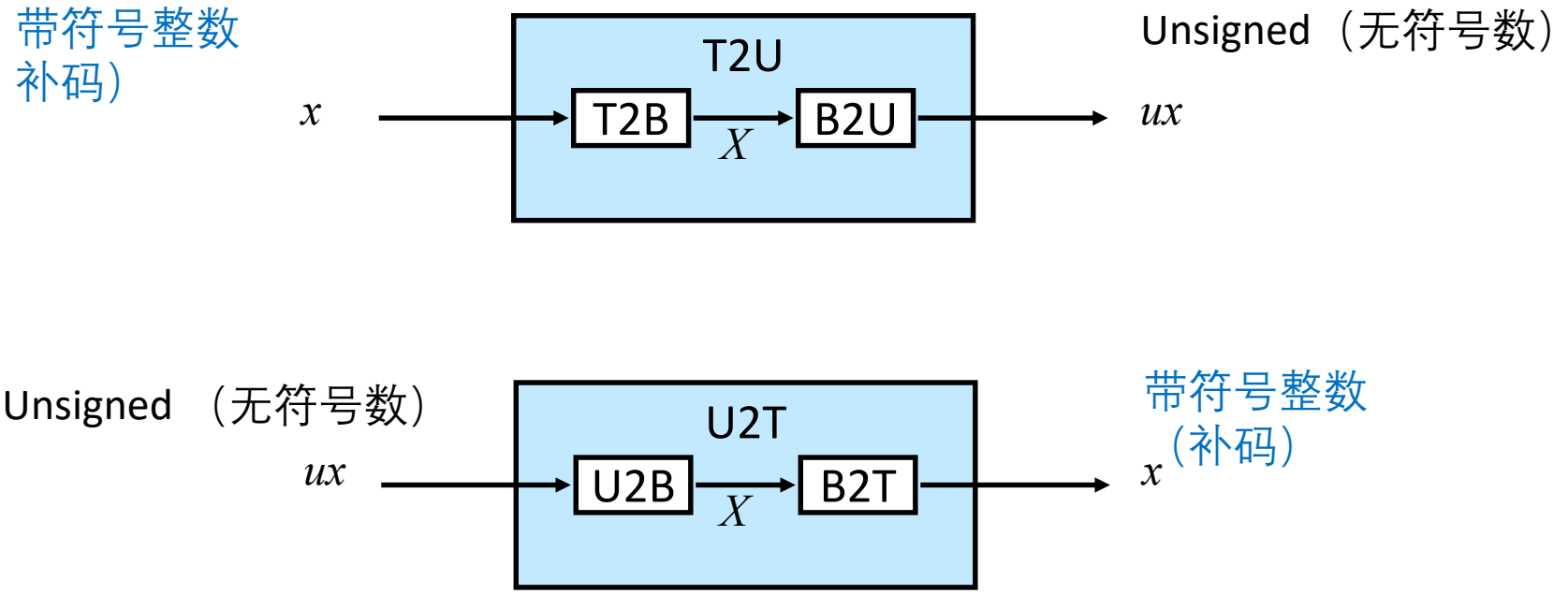
    tx = (int) ux;

    uy = (unsigned) ty;
- 隐式转换：赋值、过程调用时

    tx = ux;

    uy = ty;
  - 表达式中如果有unsigned，signed 会自动转换为 unsigned
  - **包括比较操作** <, >, ==, <=, >=

Signed & Unsigned 之间的转换

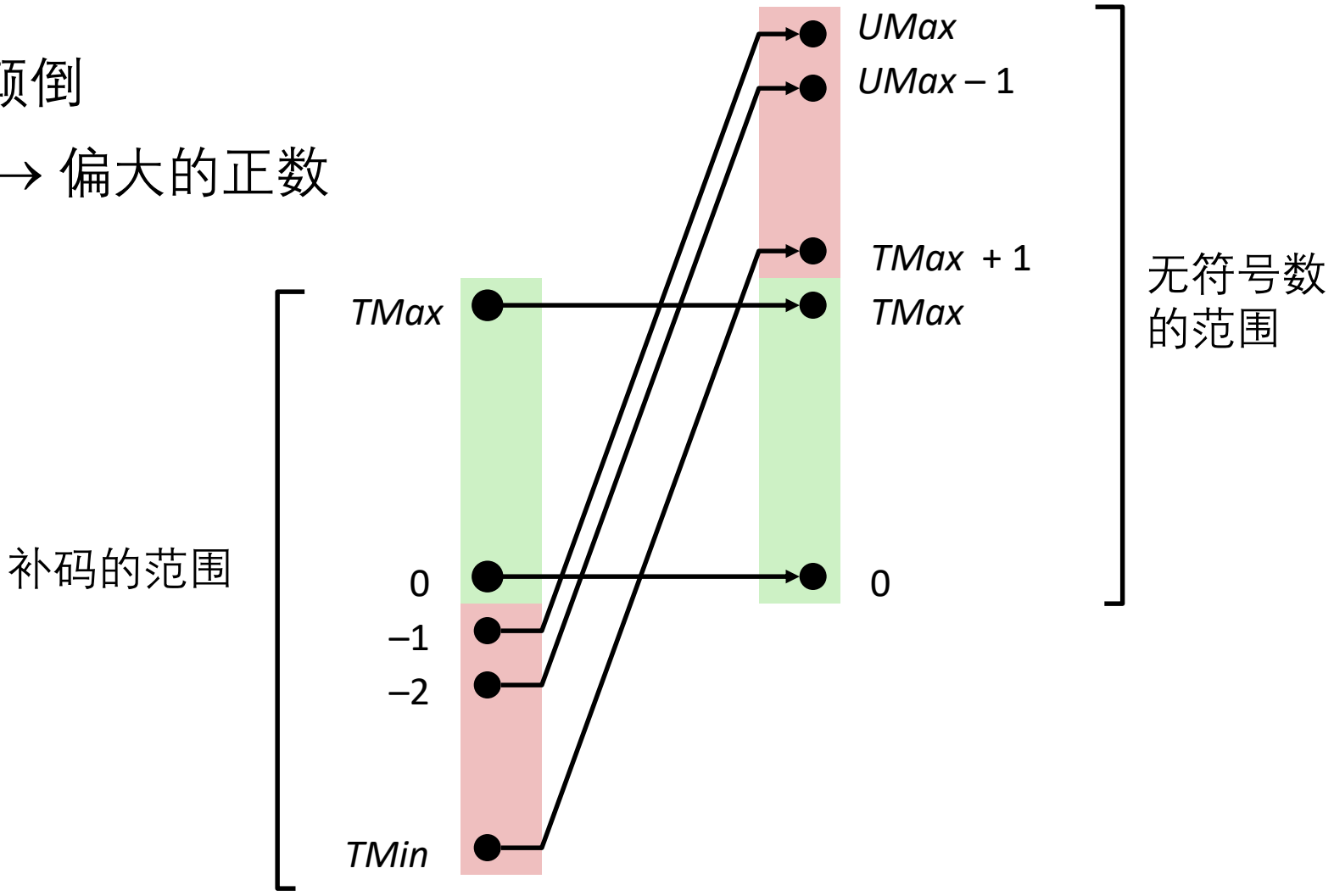


unsigned 与带符号整数 之间的转换：

- 维持机器数的格式不变，重新解释

有符号转换为无符号

- 顺序颠倒
- 负数 → 偏大的正数



举例

以下C语言代码：

```
short si=-8196;  // 1101 1111 1111 1100
unsigned short usi=si;  // 1101 1111 1111 1100
```

执行以上程序后，usi的值为：57340

举例

- 同时有无符号数和有符号数参加运算时，C编译器会隐含的将带符号整数强制类型转换为无符号数。
- 例如：以下程序试图求所有数组元素的和，元素的个数由length 表示。

```
1   /* WARNING: This is buggy code */
2   float sum_elements(float a[], unsigned length) {
3       int i;
4       float result = 0;
5
6       for (i = 0; i <= length-1; i++)
7           result += a[i];
8       return result;
9   }
```

当length 的参数值设置为0， 出现什么问题?

举例：

已知 f(n)= = 11…1 B，计算f(n)的C语言函数f 如下：

```
1       int f ( unsigned n)
2       {       int sum = 1 , power = 1 ;
3               for( unsigned i= 0;  i<= n - 1 ;  i + + )
4               {       power *= 2;
5                       sum + = power;
6               }
7               return sum;
8       }
```

当n = 0时，f 会出现死循环，为什么？

什么时候使用unsigned？

- 需要正整数的模运算系统时
  - C 语言的标准实现：unsigned 加法 类似于取模算术运算
    - 0 → 1 → ··· → *Umax* → 0
- 需要用数的各位编码来表示集合时

小结

- 无符号数： unsigned
- 编程时要注意有符号数和无符号数之间的转换
- C语言支持隐含的强制类型转换，可能会带来意想不到的问题