pg22 浮点运算不可结合

由于表示的精度有限，浮点运算是不可结合的

> e.g：(3.14+1e20)-le20的值为0.0，而3.14+(1e20-1e20)的值是3.14

pg24 GCC

GCC（GNU Compiler Collection）：GNU编译器套装

pg27 32位程序/64位程序的区别

对于一个字长为w位的机器而言，虚拟地址的范围为0~2^w-1，程序最多访问2^w个字节

将程序称为“32位程序”或者“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

布尔运算

- pg36 `|` 和 `&` 有结合律、分配律

  > a & (b | c) = (a & b) | (a & c)
  >
  > a | (b & c) = (a | b) & (a | c)

- pg36 异或

  > a ^ 0 = a
  >
  > a ^ a = 0
  >
  > a ^ 全1 等价于 ~a

- 练习题2.12-B 位运算实现 `x==y`：`!(x ^ y)`
- 练习题2.10 位运算实现原地交换

```c
// 注意 这个技巧基本没啥用
// 参考：https://blog.csdn.net/solstice/article/details/5166912
void inplace_swap(int *x, int *y) {
    *y = *x ^ *y;
    *x = *x ^ *y;
    *y = *x ^ *y;
}
```

pg48 补码、反码

- 补码（$Two's\quad complement$）：对于非负数x，我们用2^w-x（这里只有一个1）表示$-x$
- 反码（$Ones'\quad complement$）：对于非负数x，我们用$[111...1]-x$（这里有很多个1）表示$-x$

溢出判断

- 练习题2.27 无符号加法判断溢出

```
int uadd_ok(unsigned x, unsigned y) {
    unsigned sum = x+y;
    return sum >= x;
}
```

- 练习题2.30 补码加法判断溢出

```
int tadd_ok(int x, int y) {
    int sum = x+y;
    int neg_over = x < 0 && y < 0 && x+y >= 0;
    int pos_over = x >= 0 && y >= 0 && x+y < 0;
    return !neg_over && !pos_over;
}
```

- 练习题2.35 补码乘法判断溢出

```
int tmul_ok(int x, int y) {
    int p = x*y;
    return !x || p/x == y;
}
```

乘以常数优化

TODO