# Integer division

## Introduction

Numerator / denominator = quotient … remainder

|  |  |  |  |
| --- | --- | --- | --- |
| Numerator | Denominator | Quotient | Remainder |
| 7 | 2 | 3 | 1 |
| -7 | 2 | 3 | -1 |
| 7 | -2 | -3 | 1 |
| -7 | -2 | 3 | -1 |

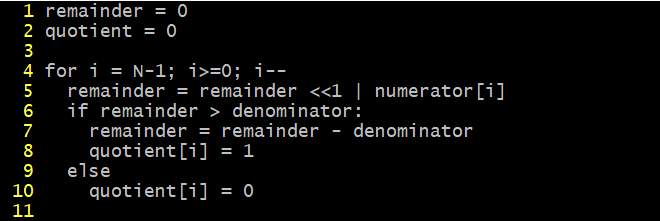
* 先把除数和被除数都转化为对应的绝对值运算，得到一个商和余数，根据符号规则加上符号就是最终运算结果
* Quotient: Numerator/denominator同号，商为正数；否则为负数
* Remainder: 和numerator同号

## Algorithm

**基2移位减**



对于N-bit有符号数的运算，需要N轮迭代减，每次的部分商为0/1算法如下：



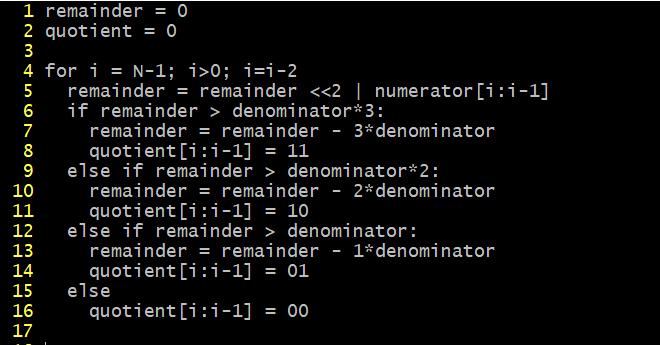
**算法优化**：我们在笔算十进制数的除法的原则是，除数被除数有效位（前导0为非有效位）高位对齐开始第一次减法运算，然后再依次查看被除数剩下的每一位,。如果被除数比除数短，那商一定是0。同样的原则也适用于2进制除法。对于32-bit的有符号运算，假设我们计算出来的Numerator的前导0的个数为N，denominator的前导0的个数为D，如果N>D，被除数比除数小，商为0， 被除数为余数。否则，numerator的前导0的每一位的迭代减就是可以省略的，numerator和denominator有效位高位对齐之前的迭代减也是可以省略的。

* Numerator leading zero saved subtraction time = N
* Numerator/denominator align saved subtraction time = 32 -D -1
* Saved subtraction: 31 +N –D
* Needful subtraction: 32 – (31 +N -D) = D –N + 1

**基4移位**



基4移位，就是将每2-bit看作一个4进制数，N-bit有符号数的运算需要N/2次迭代减法，每次的部分商为0/1/2/3中一个。



算法优化：同样的原则也适用于基4移位除法。对于32bit有符号数，不优化需要16次迭代减。同样，如果numerator的前导0为N个，denominator的前导0为D个：

* Numerator leading zero saved subtraction time = floor(N/2)
* Numerator/denominator align saved subtraction time = 16 –floor(D/2) -1
* Saved subtraction: 15 +floor(N/2) –floor(D/2)
* Needful subtraction:162 – (15 +floor(N/2) –floor(D/2)) = floor(D/2) –floor(N/2) + 1

## Implementation



Figure 0-1