# 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

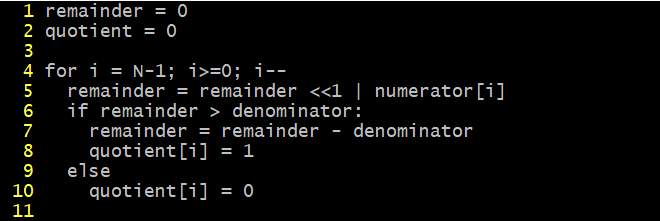
首先要将操作数转化为绝对值，并根据除数被除数的符号得出商和余数的符号。正数的绝对值是它本身，复数的绝对值等于它的二的补码表示法的数字按位取反加1。

然后用于计算的numerator和denominator就都是正数了。这里以7/2为例。

**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：



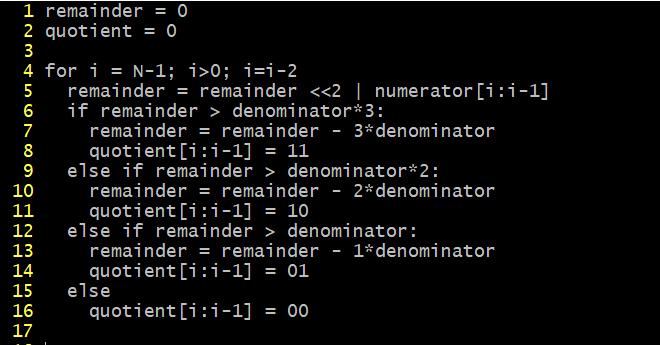
**算法优化**：我们在笔算十进制数的除法的原则是，除数被除数有效位（前导0为非有效位）高位对齐开始第一次减法运算，然后再依次查看被除数剩下的每一位,。如果被除数比除数短，那商一定是0。同样的原则也适用于2进制除法。对于32-bit的有符号运算，假设我们计算出来的Numerator的前导0的个数为N，denominator的前导0的个数为D，如果N>D，被除数比除数小，商为0， 被除数为余数。否则，numerator的前导0的每一位的迭代减就是可以省略的，numerator和denominator有效位高位对齐之前的迭代减也是可以省略的。

* 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=floor(D-N+2)

## Implementation

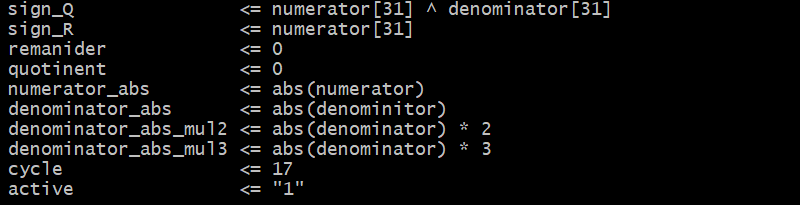
基4移位除法器比基2除法器的运算时间少一半，硬件资源的增加几乎可以忽略，这里是一个基4移位除法器的实现。以下是除法器实现的主要结构，包含numerator前导0的检测和denominator前导0的检测来减少迭代周期数。



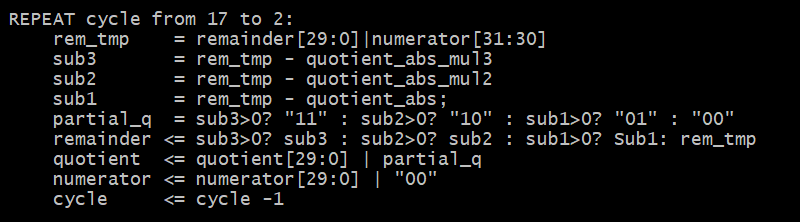
Figure - main structure of division



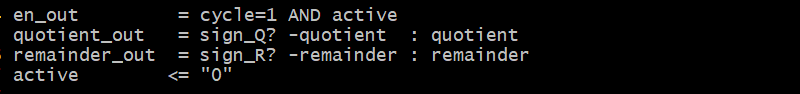
* 默认硬件除法从操作数输入到得出运算结果需要18个周期，cycle18为数据准备周期



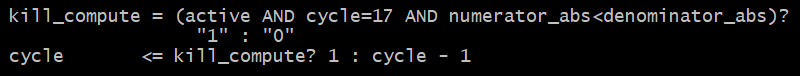
* 然后cycle17-cycle2是16个周期的迭代运算



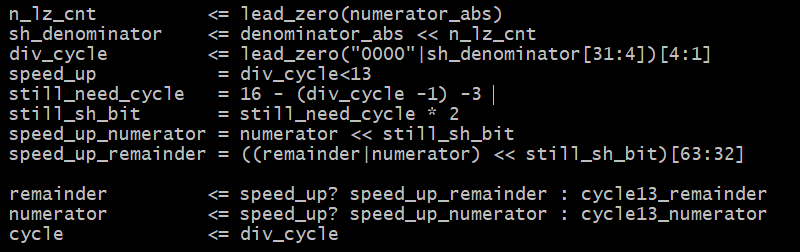
* Cycle1为1的时候输出最终运算结果



* Cycle17的时候如果检查到numerator\_abs < denominator\_abs，直接跳过迭代运算，cycle值直接改为1，下周就可以结束运算



* Numerator的前导0和denominator的有效位也可以加速迭代，但是计算出到底可以节约几拍需要3个周期，将可以节约的numerator\_abs进行移位又需要多一拍，所有cycle114的时候检测到节约的周期数大于4的时候，就可以启动speed\_up， 用div\_cycle和移位后的新的remainder & numerator替代当前的值。



* DIV实际需要的need\_cycle= floor(D-N+2)，但cycle寄存器存入的值比该值大1， 所以硬件要计算出来并写入cycle的div\_cycle= floor(D-N+4)，硬件实现就是先计算numerator的前导0（N）， 然后denominator左移N位（-N）， 然后计算移位且前面拼4个0的denominator的前导0，得出的结果就是（D-N+4）， 然后去掉最低位，就是floor。
* Cycle14的时候，numerator已经移动过3次，所有要用总省略的次数减去已经移动的3次才是还需要移动的次数。

下表是不同情况下除法执行所需要的周期数：



* 因为除法是多周期的，在执行过程中可能会遇到被flush的情形，此时active需要置为0， 所有内部寄存器不再翻转。