Signed Integer Division#

by Robert L. Smith

Originally appearing in Dr. Dobb's Journal September 1983

Not all methods of integer division produce a uniform result when the dividend and divisor have opposite signs. This may not be so important in the area of commerce where negative values are perhaps used less. When dealing with measurement and control, however, uniformity becomes more significant. The Forth-83 Standard adopts a method for signed integer division called "floored" division. While APL has used this method for the RES function for years and Small-Talk provides it as one of three methods for integer division, acceptance of the Standard makes Forth the first "popular" language to embrace this method based on its theoretical merits. The problem is one of mathematical purity versus user expectation. This article will attempt to clarify some of the issues involved.

Integer division is a mathematical function of two integers (a dividend and a divisor) that yields an integer quotient and an integer remainder. That appears to be a fairly straightforward operation, but there is not universal agreement of the desired results when one or both arguments are negative. When an ingeger quotient is used in plotting or machine control, the desired function is usually _not_ the quotient given by the majority of computers.

Most computers with a divide function produce a quotient that has a property of symmetry around zero when plotted as a function of the dividend, due to the fact that the quotient is rounded toward zero. Speaking mathematically, the property is actually one of anti-symmetry, where the sign of the quotient is reversed when the sign of the dividend (or numerator) is reversed. For integer division, this "symmetric" property leads to a sort of discontinuity around zero. In this case, the remainder is either zero or it takes the sign of the dividend. Figure 1a illustrates the quotient q as a function of a variable dividend, and a constant divisor 3. We readily see the discontinuity near zero. This may

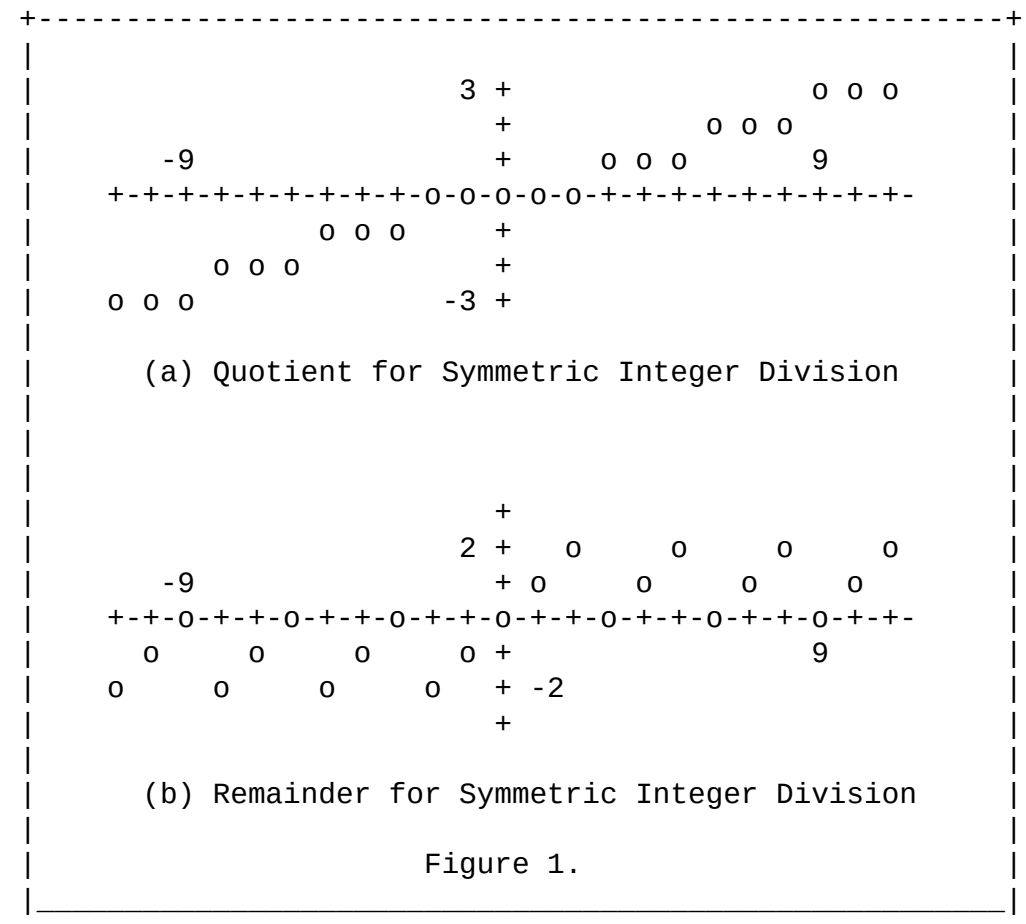

Figure 1.

be reasonably serious when this quotient function is used for plotting or moving robot arms. The integer quotient needs an associated remainder:

```
r = n - q * d
```

where n is the numerator or dividend, d is the denominator or divisor, q is the quotient, and r is the remainder. The remainder function for the constant divisor 3 is illustrated in Figure 1b. If we look at the case of positive dividends and divisors, we observe the cyclic property that

```
r(n+d) = r(n)
```

In other word, the remainder usually has a repeating or cyclical property as the dividend changes. For the remainder shown in Figure 1b, we see that this simple property is not maintained for dividends between -d and 0.

If we require that the remainder be cyclical, then the quotient no longer has any unusual discontinuities. There are a number of possible choices here. One obvious choice is to make the remainder the same as the modulus or residue function (1). In this case the quotient is rounded toward minus infinity. This rounding procedure is called the "floor" function. Figure 2 shows the floored quotient and its related modulus for the same arguments used in Figure 1. Notice the quotient behaves in a more nearly continuous fashion around zero. This is the form used in the Forth-83 Standard, as well as some of the older versions of Forth. The National 16032 microprocessor produces floored division in addition to the older "rounded toward zero" variety. The modulus function is called MOD in Forth-83 and in the National 16032. It is called RES in APL.

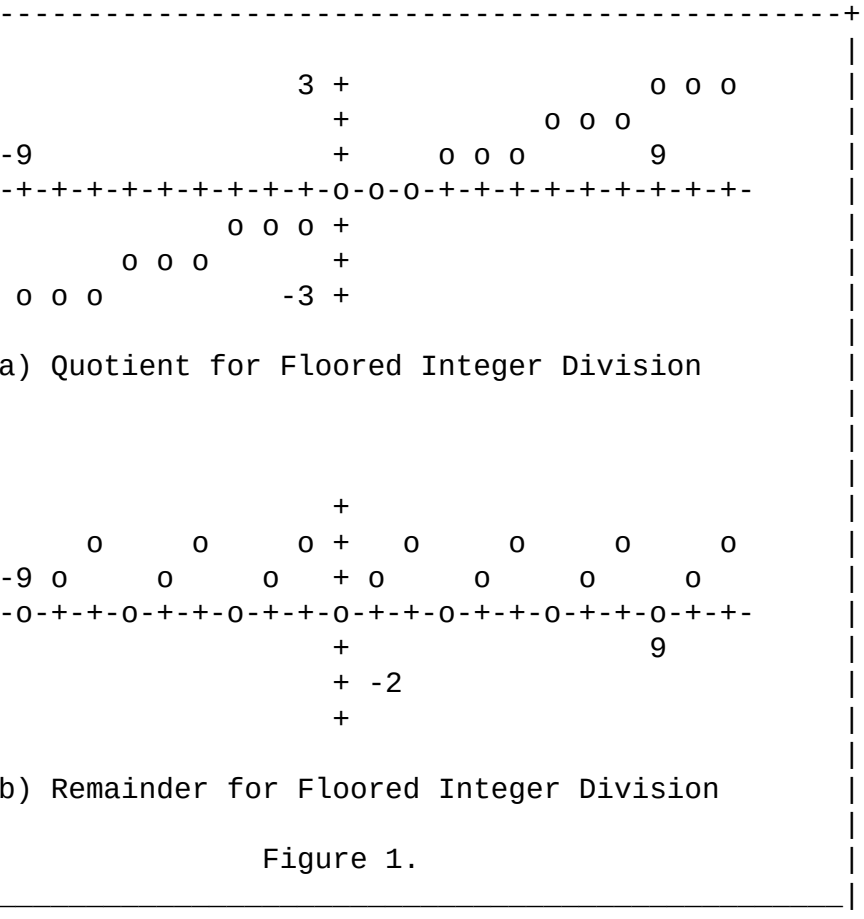

Figure 1.

The "floored" quotient shown in Figure 2 is not anti-symmetric around zero. However, for odd divisors one may easily obtain a symmetric result by adding a correction factor to the dividend prior to division. Although the quotient is generally not defined when the divisor is zero, the modulus is usually defined to take the value of the dividend for this case. If infinities are not allowed in computer representations, and the product of any number and zero is always zero, then this definition preserves the equation

```
n = q * d + r
```

for all values of d, including zero.

Alternative remainder functions include a positive modulus and a remainder that takes the sign of the quotient (2). Some other possibilities have the undesirable feature of negative remainders when the dividend and divisor are both positive.

Floored division is simply more useful in the majority of applications programs. The major objection is that the results are not what most people expect: -1 divided by 4 gives 0 in the rounded-toward-zero division case, but -1 for floored division. Both cases give the same results when the dividend and divisor have the same sign. Timing efficiencies may play a small role in deciding which form of division to use, but generally the division process is sufficiently slow that additional tests for different forms of rounding take only a little extra time. Indeed, for some processors with built-in signed and unsigned divide functions, it may be faster in the common case of positive arguments to test signs and use the unsigned division than to just use the signed division function. If you have an older Forth system (such as 79-Standard or fig-FORTH), the screen in Figure 3 shows a high-level conversion from the older form of /MOD to the newer version. For those unfamiliar with Forth, /MOD takes two arguments, the dividend and the divisor, and returns two results: the quotient and the modulus, or remainder. The quotient is returned as the most accessible element on the stack.

The appearance of floored division in some of the newer processor chips and languages indicates the increasing awareness of its utility. We might note in passing that even floating-point division will probably be different in the future than it was in the past due to the new Floating-Point Standard, which will require proper rounding of the quotient.

References#

(1) Donald K. Knuth, _The Art of Computer Programming: Volume I, Fundamental Algorithms_, Second Edition, Menlo Park: Addison-Wesley, 1973, p. 127.

(2) Robert Berkey, "Integer Division, Rounding and Remainders", 1982 FORML Conference Proceedings, San Jose, California: Forth Interest Group, 1983, pp. 13-23.

```
( Define 83-Standard  /MOD  in terms of old  /MOD )
: /MOD   ( num den -- mod quot )
   OVER OVER XOR 0<       ( test signs of arguments )
   IF                     ( signs are different )
        >R R@ /MOD OVER   ( divide and examine remainder )
        IF                ( non-zero remainder )
             1- SWAP R> + SWAP   ( adjust results )
        ELSE                     ( zero remainder )
             R> DROP             ( discard old den )
        THEN
   ELSE                          ( signs just the same )
        /MOD                     ( just divide normally )
   THEN ;                        ( end of definition )
```

Figure 3.
High Level Forth Code to Convert to Floored Division