Ada is a DoD trademark

Editor:

As a comment on the letters of Mr. Weiss and Lt. Col. Druffel (June 1981) regarding the trademarking of the name "Ada" when used in the context of a computer language, please note that the advertisement on the page following the letters used the name "Ada" more than 20 times, and as far as I can tell didn't mention that "Ada is a trademark of the US Department of Defense."

By the way, do I have to put such a notice at the bottom of this letter?

J. Richard Swenson
University of Toronto

No. We put it in the headline.  Ed.

Usual division wrong?

Editor:

In the article "Compilers and Computer Architecture," by William Wulf, July 1981, it is said that the familiar arithmetic shift instructions make an example of irregularity. This is not really true. It is true that the right shifts do not give the same result as the integer division implemented on most machines, maybe all, but the fault lies with the division, not the right shift.

The right shift works in the fashion of a function well known in number theory, the greatest-integer function. For a right shift it is true that if \( y \) is the result of shifting \( x \) right by \( n \) bits, then \( x \geq y \times 2^n \). For the usual division it is true that if \( y \) is the result of dividing \( x \) by \( w \), then \( |x| \geq |x| w \). From this it would already seem that the usual integer division is a more complex operation since the relation needs the absolute-value marks. I admit that it is not safe to conclude much from this, but it is both indicative and suggestive.

There is another way in which the usual integer division (let me write it with \( \div \)) is too complex. Consider all remainders possible when validly dividing \( x \) by \( w \). There are \( 2w - 1 \) of them. \( W \) is used when \( x \) and \( w \) have the same sign, and the rest along with 0 are used when \( x \) and \( w \) have different signs. This is to ensure that \( (x + w) \times w + x \text{ rem } w = x \). (\( X \text{ rem } w \) is the remainder on dividing \( x \) by \( w \).) It is right to guarantee this equality, but it can be guaranteed by so defining the remainder; and the remainder so defined is more useful, as may be seen below.

Now, if it happens, as it occasionally does, that one wishes to divide integers and round to the nearest integer, with the usual division one needs to do the division differently depending on whether the dividend and the divisor have the same sign:

1. \( x w \geq 0: (x + w \div 2) + w \)
2. \( x w \leq 0: (x - w \div 2) + w \)

If the division were greatest-integer division—that is, the quotient were the greatest integer less than or equal to the exact quotient—one would not need to bother to test the sign. Formula (1) would do.

Again, aside from the admittedly most common use of base conversion, the usual zero-centered division makes general base conversion troublesome. In the most common use this is no trouble because in converting to sign-magnitude notation one divides a positive number by a positive number. But in the more general case, troubles arise. Say we are converting a negative number to a positive base. The usual division will result in a string of not positive digits, where a positive number would result in a string of not negative digits. As a formalism for sign-magnitude notation this is not bad, but that is not what we are after. With greatest-integer division the digits would all be not negative, and the notation would be base-complement notation, the most usual in computers nowadays. With a negative base something like that would happen, but now with a mixing of positive and negative digits.

In all the foregoing the resulting number would be right, in some sense, but not really the thing sought. With greatest-integer division the resulting number would be truly right, as long as not negative digits are wanted; but even negative digits are easier to get consistently in the representation, if so wanted.

It might be objected that greatest-integer division makes false the equality

\[ (-x)/w = -(x/w), \]

which is true for real numbers. Well, integers are not real numbers, and this is not useful in integer algorithms anyhow. A variant related to the equality

\[ \lfloor x/w \rfloor = -\lfloor -(x/w) \rfloor \]

would be more useful.

Now, let

\[ z(v) = \begin{cases} v & \text{if } v \geq 0 \\ v + 1 & \text{if } v < 0 \end{cases} \]

\( Z \) can be defined in terms of \( \lfloor \rfloor \) and \( \lceil \rceil \), as may be seen, or \( \lfloor \rfloor \) and \(-\lceil \rceil \), or \(-\lfloor \rfloor \) and \( \lceil \rceil \). \( \lfloor \rfloor \) can be defined in terms of \( z \) thus:

\[ \lfloor v \rfloor = \begin{cases} z(v) & \text{if } z(v) \leq v \\ z(v) + 1 & \text{if } z(v) > v \end{cases} \]

Since \( v \) is really a ratio \( x/w \), the needed comparison is somewhat involved, and it is easier to define \( z \) in terms of \( \lceil \rceil \) and \(-\lfloor \rfloor \) than \( \lfloor \rfloor \) in terms of \( z \) and decrementing.

What all this is leading to is that it is better to change the usual division to match the arithmetic right shift, and use greatest-integer division, than to change the right shift, or even to think of it as deviant.

One may wonder if the usual division is really wrong, as I say it is, why it has come to be so standard. I would guess that the reason is that it is very little used in such a way that it makes a difference, unless neither dividend nor divisor is negative. This is quite different from floating-point, to which much attention has been paid for the sake of the scientific community. Even those who are interested in integers—mathematicians in number theory—care little about the remainder (other than whether it is 0) and pretty much ignore negative integers. But now that attention is being paid to it, it is time to correct integer division.

Sándor Halász
Ithaca, New York