WG14/N45
X3J11/95-05

Title: Signed Integer Division
Date: 23 August 1995

Author: Tom MacDonald
Cray Research, Inc.
655F Lone Oak Drive
Eagan MN 55121
USA
E-mail: tam@cray.com

Document Number: WG14 N452 (a.k.a. X3J11/95-053)

Abstract:

Currently signed integer division has implementation- defined semantics if either operand is negative. This proposal proposes to remove the implementation defined semantics and replace them with the Fortran rules.

Proposal: Change the following words in the current C Standard

6.3.5 Multiplicative Operators

From:

> When integers are divided and the division is inexact, if both operands are positive the result of the "/" operator is the largest integer less than the algebraic quotient and the result of the "%" operator is positive. If either operand is negative, whether the result of the "/" operator is the largest integer less than or equal to the algebraic quotient or the smallest integer greater than or equal to the quotient is implementation-defined, as is the sign of the result of the "%" operator. If the quotient "a/b" is representable, the expression "(a/b)*b + a%b" shall equal "a".

To:

> When integers are divided, the result of the "/" operator is the integer value closest to the mathematical quotient, and between zero and the mathematical quotient inclusively.
>
> If the quotient "a/b" is representable, the expression "a%b" shall equal "a-(a/b)*b".

```
Examples:  (-8) /   3  == (-2)
           (-8) %   5  ==  -3
             8  % (-5) ==   3
           (-8) % (-5) ==  -3
```

Comments:

The above wording is easier to understand, removes implementation-defined behavior from the standard, and is consistent with Fortran 90.

The LIA-1 Standard contains the following information:

> The ratio of two integers is not necessarily an integer. Thus, the result of an integer division may require rounding. Two rounding rules are in common use: round toward minus infinity, and round toward zero. Both are allowed by LIA-1. These rounding rules give identical results for divI(x,y) when "x" and "y" have the same sign, but produce different results when the signs differ.

Thus the current C Standard and this proposal conform to LIA-1.