

Re: C2124 is most disrespectful of IEEE floating point arithmetic

Re: C2124 is most disrespectful of IEEE floating point arithmetic

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2006-08/msg01265.html>

- *From:* "Tom Widmer [VC++ MVP]" <tom_usenet@xxxxxxxxxxxxx>
 - *Date:* Thu, 31 Aug 2006 13:26:53 +0100
-

Dr Pizza wrote:

Abdo Haji-Ali wrote:

"Dr Pizza" <DrPizza@xxxxxxxxxxxxxxxxxxxxx> wrote in message news:xn0eqk4re2h581y00a@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Erm, if that is your interpretation then that part of the standard has no meaning. There is not one single "mathematics". For example, the mathematics of integers has different properties from the mathematics of reals (consider how for example things like division behave differently with integers as compared to reals). There are a number of peculiarities to the mathematics used for various types; the arithmetics for the three kinds basic numeric types (signed integers, unsigned integers, floating point) are all different.

Give me the name of the book that says 1.0/0.0 is defined under any "mathematics"...

It's not a book as such, but IEC 559 clearly states that 1.0 / 0.0 is defined to be +inf. And IEC 559 governs this situation because `is_iec559` is true for both float and double.

Actually, IEC 559 makes no such guarantee. 1.0/0.0 produces a division by zero exception, which may cause a trap depending on whether a trap handler is set up (which it apparently is in the compiler!). In addition, IEC 559 does not define how source code maps onto floating point operations, and nor does the C++ standard for that matter, so that's really up to the compiler (and indeed, some compilers use extended precision registers for

Re: C2124 is most disrespectful of IEEE floating point arithmetic

intermediate operations, etc.).

`is_iec559` is simply a flag that tells you how the CPU is treating the type. It's not a flag from the compiler but from the C++ library. You can't blame the compiler for its value

The compiler ships with the library, and Microsoft bundle them all together; VC++ is a hosted environment. If the library is lying (i.e. does not properly describe the environment) then MS's implementation is faulty. The library and the compiler go together; if you maintain that the compiler is correct then the library is incorrect and there is still a problem.

The program:

```
double d = 1.0/0.0;
```

```
int main(){ }
```

is ill-formed according to the standard, regardless of whether `has_infinity` or `is_iec559` is true or not. If MSVC compiled it without warning, it would be non-conforming.

Regarding the case where the double isn't a global (and therefore the initializer isn't a constant-expression), I agree that MS would be better to define their undefined behaviour as returning `+inf`, but at least the error forces you to be explicit, and use `infinity()` when you mean it.

Tom

.