WG14 N3127
Meeting notes

C Floating Point Study Group Teleconference

2023-03-29
8 AM PST / 10 PM EST / 4 PM UTC

Attendees: Rajan, Jim, Fred, Damian, David H.

New agenda items (https://wiki.edg.com/pub/CFP/WebHome/CFP_meeting_agenda-20230329-update.pdf):
None.

Next Meeting(s):
May 10, 2023, 4PM UTC
ISO Zoom teleconference
Please notify the group if this time slot does not work.

New action items:
Jim: Comments for CD2: Item 8: Change "unsigned and unsigned zeros" -> "unsigned zeros and"
Jim: CD2 comment for float_t/double_t re floating type vs real floating type: Add "If the types are not real floating types, the behavior is implementation-defined."
All: Consider: {FLT/DEC}_EVAL_METHOD in TS-5: Instead of allowing them to change with the #pragma FENV_FLT_EVAL_METHOD, create new macros that can change with the pragma. The existing macros give the default evaluation method, and are available for #if/#elif, while the new macros have the same value as the original macros at the beginning but can change based on the #pragma FENV_FLT_EVAL_METHOD setting and are not valid for use in #if/#elif.
Damian: Rework the carg description to say phase instead of phase angle using the spreadsheet form.

C++ liaison:
Skipped.

C23 integration:
CD: https://wiki.edg.com/pub/CFP/WebHome/ISO-IEC_JTC_1-SC_22_N5777_Text_for_CD_9899.pdf

Carry-over action items results:
David H: Get an example for the scaled reduction functions (perhaps by asking Jason or Jim or looking into the IEEE references). - Not done.
See https://754r.ucbtest.org/background/traps-and-wraps.txt
David H: Get an example for the augmented arithmetic functions (perhaps by asking Jason or Jim or looking into the IEEE references). - Not done.

Action items results (from previous meeting):

Jim to submit [Cfp-interest 2684] as a CD2 comment. - Done. See Other issues entry 1 below.

Jim to submit [Cfp-interest 2686] as a CD2 comment. See “Comments for CD2” below. - Done. See Other issues entry 1 below.

Discussion:

None.

Other issues:

Comments for CD2

See [Cfp-interest 2728] possible NB comments for CD2; https://wiki.edg.com/pub/CFP/WebHome/CD2_comments-2D20230307.pdf

1) Looks OK.
2) OK.
3) OK.
4) OK.
5) OK.
6) OK.
7) OK.
8) ^Jim: Comments for CD2: Item 8: Change "unsigned and unsigned zeros" -> "unsigned zeros and"
9) OK.
10) Fred: 754 does not require complete subsets for the NaN payload?
Jim: No it does not.
David: This is OK.
Jim: This doesn't talk about SNaNs. Inclined to leave that to the implementation.
OK.
11) Typo. OK.

Definition of “floating types”

See [Cfp-interest 2654 and chain] definition of "floating types"

Jim: I would suggest the answer is No for having wide representations being types in some way. Evaluating the expression, the type is the semantic type. There still can be a wider representation but the type is the type.

Jim: For _t types, should they be real floating types from floating types? Classification macros can't use these and other things as well. The _t types are for arithmetic. If they are not real floating types, it is ambiguous.

Damian: Shouldn't they be real floating types? They need to remain so.

Jim: We don't say that yet (7.12#3). We can require them to be real floating types. A footnote can say they can be implementation extension types. An alternative is if they are not real floating types, the behavior is unspecified.

Rajan: No implementation has this as a non-real floating type right?

Fred: No, Microsoft has wide representation internally that is not in storage. Gives more precision.

Jim: True, but that is wide evaluation, not directly what is being talked about here. They could also make float_t be double.

Fred: That wouldn't work on x87 Microsoft. They use precision of double, but wider exponent range. And long double is the same as double for them.

Jim: HP had an internal register format with 2 extra exponent bits (82-bit format). It was made into an extension type for math library implementers (not end users). It had properties of a real floating type. That type could have been the definition of the _t types.

Fred: Is floating type defined anywhere?

Jim: It is pretty squishy. Generally types that represent floating point numbers.

Rajan: There has to be choices for unspecified, so I don't think that is the correct term for alternative 2.

Jim: It should be implementation defined.

Fred: I like it. Unspecified has to list the choices.

^OK to be a CD2 comment for float_t/double_t if not being real floating types, they are implementation defined. I.e. Add "If the types are not real floating types, the behavior is implementation-defined."

Jim: Nothing says if one has to be included in another, other than in Annex H.

Fred: It should be a constraint violation.

Jim: There are lots of examples but I can't come up with them now. Double double with float128 is an example.

Fred: I think it is better for portability to force the programmer to have a cast. If neither operand is a subset of the other, it is a constraint violation.

Jim: This would disallow converting both to a wider format that contains both.

Fred: Correct.

Jim: For wide evaluation, would you preclude it?

Rajan: Where would this be? What does this apply to? There are issues with usual arithmetic conversions, other types, etc.

Fred: I need to look at the standard to figure it out.

Jim: For intermediates, the extra range/precision may not correspond to any other type or be contained in another type or vice versa.

Review TS part 4 revision

See [Cfp-interest 2710] Re: post-C23 update for TS 18661-4

Review TS part 5 revision

See [Cfp-interest 2730] TS 18661-5 revision 20230327; https://wiki.edg.com/pub/CFP/WebHome/cfp5r-20230327.pdf

Issue 1) Looks like a good idea. Issues with sync, but still very useful.

Issue 2) Rajan: Can we do a new macro name for the changing methods? The existing ones are the default, and available for #if/#elif, while the new ones have the same default originally but can change based on the #pragma FENV_FLT_EVAL_METHOD and not be valid for use in #if/#elif.

Jim: Sounds like something that would work.

^All: Consider: {FLT/DEC}_EVAL_METHOD in TS-5: Instead of allowing them to change with the #pragma FENV_FLT_EVAL_METHOD, create new macros that can change with the pragma. The existing ones are the default, and available for #if/#elif, while the new ones have the same default originally but can change based on the #pragma FENV_FLT_EVAL_METHOD and not be valid for use in #if/#elif.

Issue 3) _Pragma may cause issues with prefix_FLT_EVAL_METHOD if there is no prefix.

Conversions of comparison macro arguments

See [Cfp-interest 2731 and chain] comparison macros and exceptions due to conversion to the semantic type

Jim: In the main standard, we leave it unspecified or implementation defined if the arguments are converted to the semantic type before the comparison is done. For Annex F, we say explicitly they are NOT converted. This allows the same values as the builtin comparison operators. For the main body, if overfloat/underflow occurs, does it signal? The specification says without SNaNs they should not signal.

Jim: I believe any conversions have to be prior to the comparison. If there is an overflow, it would be raised. Similar to function arguments.

Fred: Agreed.

Jim: CFP2733 has Vincent show how an implementation may miss an overflow/underflow.

Jim: Any issue with the proposed clarification in the email?

Rajan: From an implementation perspective, we should not bias one way over another.

Jim: Can go on Fred's list? Let's leave this for further thought and put it in Fred's list if we don't decide to put it into CD2.

Terminology in definition of carg

See [Cfp-interest 2734] Definition of the 'carg' function

Looks good. Needs to be in the right spreadsheet comment form.

Consensus not to have alternatives.

^Damian: Rework the carg description to say phase instead of phase angle using the spreadsheet form.

Regards,

Rajan Bhakta
z/OS XL C/C++ Compiler Technical Architect
ISO C Standards Representative (Canada, USA), INCITS/C Chair
C/C++ Compiler Development
rbhakta@us.ibm.com