Defang and deprecate memory_order::consume

Doc. No: P3475R0
Contact: Hans Boehm (hboehm@google.com)
Audience: SG1
Date: Oct 15, 2024
Target: C++26
(Arguably a revision of P0371R0. But P0371R1 took a different direction and was accepted.)

Abstract

We again propose to deprecate `memory_order_consume`. We suggested a different variant of this about 8 years ago in P0371R0, which SG1 did not like. Circumstances have changed, and SG1 seemed much more amenable to this at the St. Louis meeting.

Rationale

It is widely accepted that the current definition of `memory_order::consume` in the standard is not useful. All current compilers essentially map it to `memory_order::acquire`. The difficulties appear to stem both from the high implementation complexity, from the fact that the current definition uses a fairly general definition of "dependency", thus requiring frequent and inconvenient use of the `kill_dependency()` call, and from the frequent need for `[[carries_dependency]]` annotations.

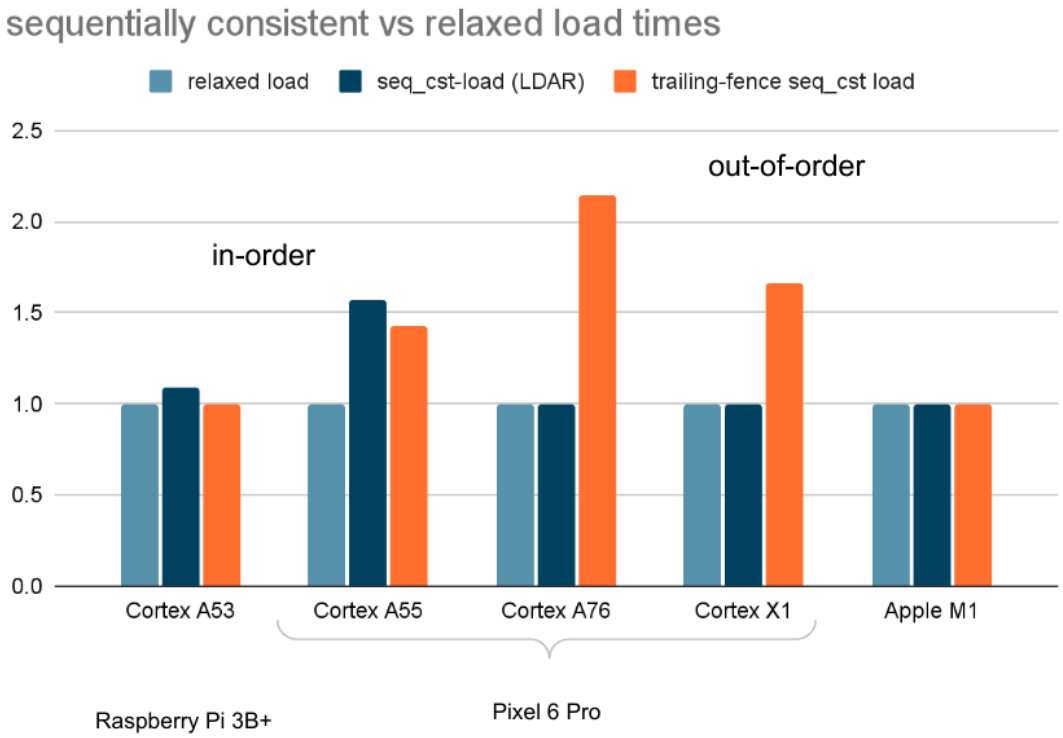
It is also widely accepted that `memory_order::consume` has frequent and important use cases in many large code bases. For example, the Linux kernel makes extensive use of RCU. To avoid over-constraining memory order for something like RCU on architectures like ARM and Power. something like `memory_order::consume` is required. Some core Android code similarly relies on dependency-based ordering.

On the other hand, there are strong arguments that we are not likely to introduce a new facility that can benefit significantly from the existing wording, and we should remove it:

1. This problem has been recognized for around a decade. We have had much discussion around it and possible replacements (e.g. P0098, P0190, P0750) We have not agreed on a replacement. We have not fixed even serious issues with its current specification, discouraging its use instead.
2. It complicates the memory model appreciably. More mathematical academic work tends to ignore its existence, since it is known to be broken. This makes it harder to translate

such work into the standard. The memory model is in some need of tricky repair. It would be nice to avoid this unused complication.

3. The most widely-used CPU architectures no longer really require `memory_order::consume`. It was never very beneficial on X86. ARM provides hardware support for `memory_order::acquire` (and `memory_order::seq_cst`) loads that, for common micro-architectures and use-cases, perform similarly to `memory_order_relaxed`. RISC-V currently lacks this support, but appears to be moving in the same direction with the proposed Zalasr extension. The main architectures that could still benefit are Power and GPUs.
   Here is a comparison of microbenchmark `relaxed` vs `seq_cst` load times on some ARM implementations. (All versions include an indirect function call. so differences are understated, but not hugely so. The dark blue LDAR version uses the hardware `seq_cst` load instruction.)



4. In practice, non-portable, very careful, abuse of `memory_order_relaxed` seems to have carried the day for the remaining use cases.

Outline of wording changes

1. Remove [intro.races] p7 and p8, which define “carries a dependency” and “dependency-ordered before”.
2. Remove [intro.races] p9, the definition of “inter-thread happens before”.

3. Remove [intro.races] p10, the definition of “happens before”.
4. Modify [intro.races] to rename “simply happens before” to “happens before”
5. Remove [dcl.attr.depend], which defines `[[carries_dependency]]`. There shouldn’t be a reason to deprecate it, since remaining occurrences in code will presumably be ignored anyway.
6. In [atomics.order], perhaps we can just delete 1.3, which defines `consume`, but leave `consume` in the `enum class` as a placeholder, defining it in Annex D. I don’t believe we can remove it, since we want to preserve the mapping to the underlying type.
7. Remove mentions of `memory_oder::consume` from the many lists of acceptable memory orders in *33: Concurrency Support Library*.
8. Remove all mentions of `kill_dependency`, including the definition in the final paragraphs of [atomics.order].
9. Add a new section to Annex D:

> D.? memory_order_consume [depr.consume]
>
> `memory_order::consume` is equivalent to `memory_order::acquire`.
>
> ```
> template<class T> T kill_dependency(T y) noexcept;
> ```
> *Returns*: y.