Contribution ID: **86** Type: **not specified**

The never-ending saga of control dependencies

Friday, 24 September 2021 08:50 (30 minutes)

The Linux kernel continues to rely on control dependencies as a cheap mechanism to enforce ordering between a prior load and a later store on some of its hottest code paths. However, optimisations by both the compiler and the CPU hardware can potentially defeat this ordering and introduce subtle, undebuggable failures which may only manifest on some systems.

Improving the robustness of control dependencies is therefore a hotly debated topic, with proposals ranging from limiting their usage, inserting conditional branches, introducing compiler support and using memory barriers instead. The scope of possible solutions has resulted in somewhat of a deadlock, so this session aims to cover the following in the interest of progressing the debate and soliciting opinions from others:

- What are control dependencies?
- How can they be broken by the compiler?
- How can they be broken by the CPU? (specifically, arm64)
- volatile_if() and a potential compiler __builtin
- A better barrier() macro
- Upgrading READ_ONCE() and relaxed atomics to have acquire semantics

LKML mega-thread: https://lore.kernel.org/r/YLn8dzbNwvqrqqp5@hirez.programming.kicks-ass.net

I agree to abide by the anti-harassment policy

I agree

Primary authors: DEACON, Will; ZIJLSTRA, Peter (Intel OTC); MCKENNEY, Paul (Facebook); ALGLAVE, Jade (Arm)

Presenters: DEACON, Will; ZIJLSTRA, Peter (Intel OTC); MCKENNEY, Paul (Facebook); ALGLAVE, Jade (Arm)

Session Classification: Toolchains and Kernel MC

Track Classification: Toolchains and Kernel MC