Jan 5, 2023 | RISC-V Control Transfer History TG Meeting

Attendees: tech.meetings@riscv.org Beeman Strong Bruce Ableidinger

Notes

- **Attendees**: Beeman, Snehasish, JohnS, Bruce, StasN, GregF
- **Slides** here (same as last time, but with new last slide)
    - Had an issue with zoom where I couldn't record, sorry
- Resumed on slide 12
- Full context switch only employed with call-stack mode, otherwise clear records on context switch
- Reviewing Stephane Eranian's input from Oct 2022
- Want to minimize overhead of use by making accesses fast
    - Bruce: sampling overhead can be minimized by adjusting sample rate
    - Beeman: true, but want to making accesses fast reduces need to reduce sample rate when using CTR
- AutoFDO samples on branches that are recorded. Intel supports BR_RETIRED events per branch-type, to allow alignment with LBR config. Also supports LBR_INSERTS event to simply count any transfers that update LBRs.
- When to freeze CTR will depend somewhat on the implementation. What branch/stack history leading up to sample PC, which may mean freeze on overflow (if the PC at that point is saved somehow) or on interrupt (if the sample PC will simply be that of the inst before the interrupt).
- Known, modern implementations support 32 entries
    - Save for AMD BRS, which has 16
    - 16 gets most of value, but 32 better
    - Snehashish: 16 entries means have to sample more, take more interrupts
- Call-stack mode is nice, but not as nice as full shadow stack since don't have path back to main
    - Software could preload CTR call-stack when enabling, but doing one stack unwind
    - Bruce: code doesn't always use calls & returns, which messes up call-stack mode
        - E-trace has transfer type defs that include tail-calls and co-routine swaps. If we leverage those, could be smart about how they update the stack.
- Bruce: what does VM support entail?
    - Beeman: not sure we discussed this with Stephane, but believe it would mean priv mode filtering, and ifc that would allow VM to configure CTR without the ability to record more privileged code
- 32 entries good, 16 increases overhead, no need for config yet. >32 helpful.
    - Beeman: configurable depth may be more useful if there are implementations with >32 entries, to allow limiting of context switch overhead. Also useful for VM

migration in heterogeneous datacenters, to defeature an implementation with more CTRs to match one with less.
    - Google not as concerned with VM migration, but that's for Google's own use. Would be more problematic if a tenant was using CTR and seeing changes on migration.
    - Does Google expose LBR/etc to tenants directly? No, but believe MS Azure/Hyper-V does.
- ARM supports up to 64 entries, with banks of 32
- Intel had implementations where a macrofused cmp+jcc would record the PC of the cmp rather than the jcc, which caused problems
- **Requirements Brainstorm**
    - Number of entries
        - Snehasish: recommend min of 32, don't want embedded to opt for something lower
        - Greg: concerned that min of 32 might be too high for more minimal implementations
        - *Conclusion*: let's make 16 min, recommend 32+, and may require 32 in some profiles/platform specs
    - CSR vs MMIO ifc
        - Greg: MMIO seems slower, unless HW is turning it into reg accesses (which they don't like). MMIO has to be non-spec, whereas CSR reads could be non-spec.
        - Greg: third answer could be to record internally then push to cacheable memory
            - Questions then about how to write the HW records, or whether that's necessary
        - This is a complicated topic, propose we consider other requirements first, then spend a meeting going over pros & cons of this one
- Out of time, will resume requirements discussion next meeting

Action items

- [ ]