Jan 19, 2023 | RISC-V Control Transfer History TG Meeting

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

Notes

- **Attendees**: Beeman, Bruce, Robert, JohnS, Snehasish, StasN
- **Slides** here
- Continuing discussion of CTR requirements
- Last time agreed to require at least 16 entries, recommend 32+
- Configurable depth: useful for minimizing context switch overhead, and for emulating homogeneous HW in a heterogeneous datacenter
    - **Will leave as optional**
    - SW could just use the lower entries? Yes, but if not switching upper entries then could have leakage across contexts in them
- Priv mode access
    - Should this mirror HPMs and be M-mode only by default, with opt-in to lower priv modes? Or just make it an S-mode feature by default?
    - All priority usages will be in S-mode
    - M-mode can always use an S-mode capability, it just can't forbid S-mode from using it
    - **Make it S-mode by default**
        - Could opt to make it readable from U-mode
    - Stas: could we start it from U-mode? If filtered to only capture U-mode?
        - Possible, would need to get further into definition
        - SW would have to ensure it's context switching, to avoid leakage
        - Save this discussion for later
- Metadata
    - Autofdo today uses only source/target PC
    - Some other usages use transfer type, cycles, mispredict
    - A minimal implementation may support only 2 regs (source/target)
        - Though we'll need to discuss whether entries should have a fixed ifc (e.g., 3 regs) or flexible (2, 3, 4+?). Former is nicer for SW, latter is more flexible for architecture.
        - But regardless of reg ifc, a minimal implementation (source/target only) could always report 0 for any unsupported/unimplemented regs
    - **Keep metadata optional, require source/target**
- Filtering
    - **Priv mode filtering should be required**
    - **Transfer type filtering optional**
        - Baseline implementation will just record all control flow transfers, which meets autofdo needs
        - Do we need it for call-stack mode? Not necessarily, but agreed that call-stack mode too is optional

            - Do we want the option to support recording not-taken branches?
                - Would be a natural extension of type filtering, since E-trace type encoding (which we will likely share) includes not-taken branches
                - Could be useful for identifying mispredicted not-taken branches precisely
                - But could be more expensive to implement, if an implementation limits taken transfers that can retire per cycle, but not not-taken transfers
                - Leave this as an option to discuss more later
        - Context filtering could match Sdtrig, filter by ASID/VMID or xcontext
            - **Context filtering optional**
- Modes: **call-stack mode optional**
    - Useful, but not required for baseline autofdo usage
- Freeze
    - Freeze is important for ensuring that transfers in skid window don't overwrite entries leading up to sample PC at time of counter overflow, if implementation supports recording the sample PC at overflow somehow
        - There is nothing standardized to do this, but implementations could store it to memory, trace, a register, etc
    - For a simple implementation, where the sample PC is simply the epc in the LCOFI handler, then freeze is needed only if priv mode filtering won't inhibit CTR recording upon entry to the LCOFI handler (e.g., because handler runs in the same mode as the code being profiled)
    - Suggest defining a freeze mechanism, but leave it up to implementations when to freeze
        - With guidance that it should be on the sample PC, whatever that is
    - Robert: could define multiple freeze bits (freeze on overflow, freeze on LCOFI, etc), and implementation could choose which one(s) to support
        - Agreed
    - Some disagreement on whether freeze should be optional, **pick up here next time**

Action items

- [ ]