RISC-V Cycle and Instret Privilege Mode Filtering (Smcntrpmf)

Authors: Beeman Strong

Version 1.0, 2023-11-30: Ratified

Table of Contents

Preamble . . . 1
Copyright and license information . . . 2
Contributors . . . 3
1. Introduction . . . 4
2. CSRs . . . 5
    2.1. Machine Counter Configuration Registers (mcyclecfg, minstretcfg) . . . 5
3. Counter Behavior . . . 6

Preamble

This document is in the Ratified state

No changes are allowed. Any desired or needed changes can be the subject of a follow-on new extension. Ratified extensions are never revised.

Copyright and license information

Contributors

This RISC-V specification has been contributed to directly or indirectly by:

- Beeman Strong <beeman@rivosinc.com>
- Greg Favor <gfavor@ventanamicro.com>
- John Hauser <jh.riscv@jhauser.us>

Chapter 1. Introduction

The cycle and instret counters serve to support user mode self-profiling usages, wherein a user can read the counter(s) twice and compute the delta(s) to evaluate user software performance and behavior. Currently, these counters are not filtered by privilege mode, and thus they continue to increment while traps (e.g., page faults or interrupts) to more privileged code are handled. This causes two problems:

- It introduces unpredictable noise to the counter values observed by the user.
- It leaks information about privileged software execution to user mode.

This proposal remedies these issues by introducing privilege mode filtering for the cycle and instret counters.

Chapter 2. CSRs

2.1. Machine Counter Configuration Registers (mcyclecfg, minstretcfg)

mcyclecfg and minstretcfg are 64-bit registers that configure privilege mode filtering for the cycle and instret counters, respectively.

63	62	61	60	59	58	57:0
0	MINH	SINH	UINH	VSINH	VUINH	*WPRI*

Field	Description
MINH	If set, then counting of events in M-mode is inhibited
SINH	If set, then counting of events in S/HS-mode is inhibited
UINH	If set, then counting of events in U-mode is inhibited
VSINH	If set, then counting of events in VS-mode is inhibited
VUINH	If set, then counting of events in VU-mode is inhibited

When all *x*INH bits are zero, event counting is enabled in all modes.

For each bit in 61:58, if the associated privilege mode is not implemented, the bit is read-only zero. Bits 57:56 are reserved for possible future modes.

For RV32, bits 63:32 of mcyclecfg can be accessed via the mcyclecfgh CSR, and bits 63:32 of minstretcfg can be accessed via the minstretcfgh CSR.

The CSR numbers are 0x321 for mcyclecfg, 0x322 for minstretcfg, 0x721 for mcyclecfgh, and 0x722 for minstretcfgh.

The content of these registers may be accessible from Supervisor level if the Smcdeleg/Ssccfg extensions are implemented.

The more natural CSR number for mcyclecfg would be 0x320, but that was allocated to mcountinhibit.

This register format matches that specified for programmable counters by Sscofpmf. The bit position for the OF bit (bit 63) is read-only 0, since these counters do not generate local counter overflow interrupts on overflow.

Chapter 3. Counter Behavior

The fundamental behavior of cycle and instret is modified in that counting does not occur while executing in an inhibited privilege mode. Further, the following defines how transitions between a non-inhibited privilege mode and an inhibited privilege mode are counted.

The cycle counter will simply count CPU cycles while the CPU is in a non-inhibited privilege mode. Mode transition operations (traps and trap returns) may take multiple clock cycles, and the change of privilege mode may be reported as occurring in any one of those cycles (possibly different for each occurrence of a trap or trap return).

> *The RISC-V ISA has no requirement that the number of cycles for a trap or trap return be the same for all occurrences. Implementations are free to determine the extent to which this number may be consistent and predictable (or not), and the same is true for the specific cycle in which privilege mode changes.*

For the instret counter, most instructions do not affect mode transitions, so for those the behavior is clear: instructions that retire in a non-inhibited mode increment instret, and instructions that retire in an inhibited mode do not. There are two types of instructions that can affect a privilege mode change: instructions that cause synchronous exceptions to a more privileged mode, and xRET instructions that return to a less privileged mode. The former are not considered to retire, and hence do not increment instret. The latter do retire, and should increment instret only if the originating privilege mode is not inhibited.

> *The instret definition above is intended to ensure that the counter increments in a predictable fashion. For example, consider a scenario where minstretcfg is configured such that all modes other than U-mode are inhibited. A user mode load should increment only once, even if it takes a page fault or other exception. With this definition, the faulting execution of the load will not increment (it does not retire), the handler instructions will not increment (they execute in an inhibited mode), including the xRET (it arguably retires in a non-inhibited mode, but it originates in an inhibited mode). Only once the load is re-executed and retires will it increment instret.*
>
> *In cases where an instruction is emulated by software running in a privilege mode that is inhibited in minstretcfg, the emulation routine must emulate the instret increment.*