Byte and Halfword Atomic Memory Operations (Zabha)

Version v1.0.0, 2024-04-25: Ratified

Table of Contents

Preamble . . . 1
Copyright and license information . . . 1
Contributors . . . 1
1. Introduction . . . 2
2. Byte and Halfword Atomic Memory Operations (Zabha) . . . 3
Bibliography . . . 4

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: Ved Shanbhogue, Andrew Waterman, Gianluca Guida, Hans Boehm

Chapter 1. Introduction

The A-extension offers atomic memory operation (AMO) instructions for *words*, *doublewords*, and *quadwords* (only for `amocas`). The absence of atomic operations for subword data types necessitates emulation strategies. For bitwise operations, this emulation can be performed via word-sized bitwise AMO* instructions. For non-bitwise operations, emulation is achievable using word-sized `LR`/`SC` instructions.

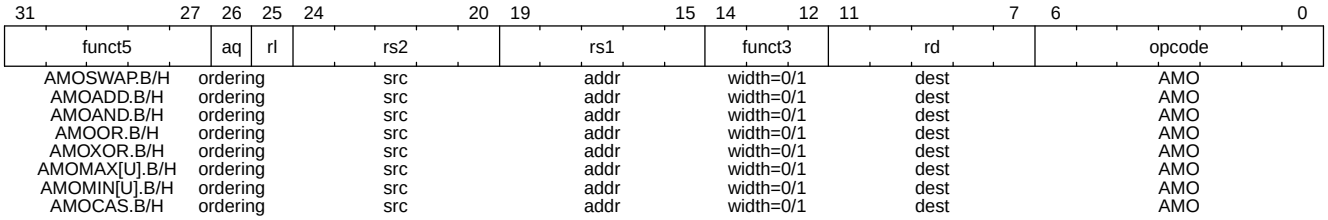
Several limitations arise from this emulation approach:

1. In systems with large-scale or Non-Uniform Memory Access (NUMA) configurations, emulation based on `LR`/`SC` introduces issues related to scalability and fairness, particularly under conditions of high contention.
2. Emulation of narrower AMOs through wider AMO* instructions on non-idempotent IO memory regions may result in unintended side effects.
3. Utilizing wider AMO* instructions for emulating narrower AMOs risks activating extraneous breakpoints or watchpoints.
4. In the absence of native support for subword atomics, compilers often resort to inlining code sequences to provide the required emulation. This practice contributes to an increase in code size, with consequent impacts on system performance and memory utilization.

The Zabha extension aims to address these limitations by adding support for *byte* and *halfword* atomic memory operations to the RISC-V Unprivileged ISA [1]. The Zabha extension depends upon the Zaamo standard extension.

Chapter 2. Byte and Halfword Atomic Memory Operations (Zabha)

Zabha extension provides the `AMO[ADD|AND|OR|XOR|SWAP|MIN[U]|MAX[U]].[B|H]` instructions. If Zacas [2] extension is also implemented, Zabha further provides the `AMOCAS.[B|H]` instructions.



Byte and halfword AMOs always sign-extend the value placed in `rd`, and ignore the $XLEN-1:2^{(width+3)}$ bits of the original value in `rs2`. The `AMOCAS.[B|H]` instructions similarly ignore the $XLEN-1:2^{(width+3)}$ bits of the original value in `rd`.

Similar to the AMOs specified in the A extension, the Zabha extension mandates that the address contained in the `rs1` register must be naturally aligned to the size of the operand. The same exception options as specified in the A extension are applicable in cases where the address is not naturally aligned.

Similar to the AMOs specified in the A and Zacas extensions, the AMOs in the Zabha extension optionally provide release consistency semantics, using the `aq` and `rl` bits, to help implement multiprocessor synchronization.

Zabha omits byte and halfword support for `LR` *and* `SC` *due to low utility.*

Bibliography

[1] “RISC-V Instruction Set Manual, Volume I: Unprivileged ISA .” [Online]. Available: github.com/riscv/riscv-isa-manual.

[2] “Atomic Compare-and-Swap (CAS) instructions.” [Online]. Available: github.com/riscv/riscv-zacas.