RISC-V Summit China 2023

RISC-V Vector Support on Valgrind

Wu Fei < fei2.wu@intel.com>
System Software Architect

Legal Notices and Disclaimers

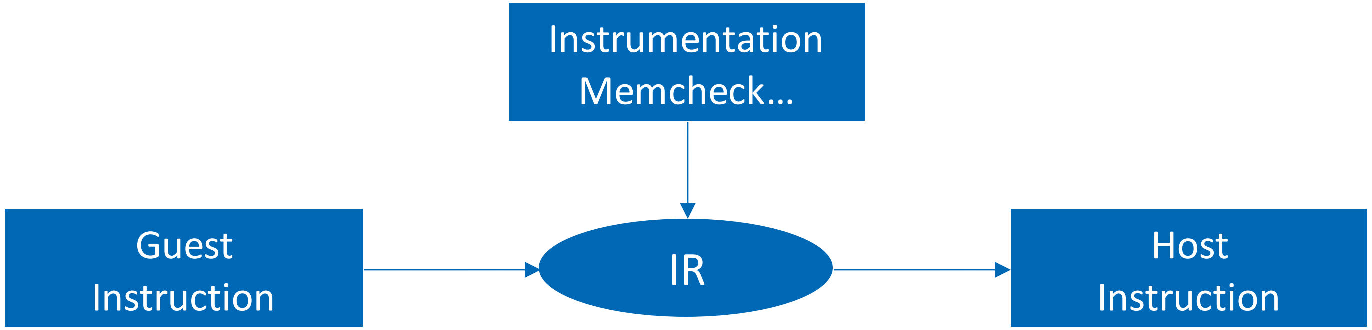
Statements in this document that refer to future plans or expectations are forward-looking statements. These statements are based on current expectations and involve many risks and uncertainties that could cause actual results to differ materially from those expressed or implied in such statements. For more information on the factors that could cause actual results to differ materially, see our most recent earnings release and SEC filings at www.intc.com.

All product plans and roadmaps are subject to change without notice. Any forecasts of goods and services needed for Intel's operations are provided for discussion purposes only. Intel will have no liability to make any purchase in connection with forecasts published in this document. Code names are often used by Intel to identify products, technologies, or services that are in development and usage may change over time. No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

This document contains information on products and/or processes in development.

Background

- Valgrind currently lacks support for the RISC-V Vector ISA, while it has already been enabled for RV64GC.
- There is no existing support on Valgrind for variable length vector instructions, a new design is required.
- Here is a simplified flow of how Valgrind works:



Design Choices

Method for RVV	Pro	Cons
Scalar emulation	• Leverage existing scalar IRs	• IR explosion • Hard to optimize
Dirty helper	• Easy for basic binary translation, e.g. tool=none	• Deviate from the design principle of Valgrind • Deal with the instrumentation tools such as Memcheck directly
Vector IR	• Standard way to extend IR • Generic design across different vector ISAs	• Requires brand new design

- The preferred way in descending order to enable new instruction on Valgrind
  - Using existing Iops, creating new Iops, clean helper, dirty helper

Challenges

- Generic framework and IRs for different vector ISA
- RVV LMUL and backend register allocation
  - No register group allocation yet
- Mask instruction efficiency
  - Inefficient to handle it element by element

Current Status

- A generic vector IR encoding mechanism
- A working prototype to run simple RVV testcases
  - A few instructions uses the new vector IR
  - Memcheck runs well on the prototype
  - Framework enhancement such as adding CPU state to TB
- The Vector IR design is still in review
- RVV Intel public repository:
  - https://github.com/intel/valgrind-rvv/tree/poc-rvv
- RV64GC repository:
  - https://github.com/petrpavlu/valgrind-riscv64.git

Next Step

- Get the vector IR design reviewed
- Design the code pattern for common features such as LMUL, mask
- Complete the full RVV support

RISE

RISC-V Software Ecosystem

- https://riseproject.dev

RISE is focused on positive and transparent collaborations with upstream projects to deliver commercial-ready software for various use cases

How: Align on highest priorities & avoid (accidental) duplication of work
Goal: Accelerate open source SW for RISC-V architecture

https://www.intel.com/content/www/us/en/developer/articles/community/rising-to-the-challenge-risc-v-software-readiness.html

Finding more interesting topics from Intel on RISC-V summit China 2023

Topic	When
RISC-V Vector Support on Valgrind	August 25 6pm
Best practice to optimize SW with vectorization on RISC-V	Poster
RISC-V firmware solution	August 24 4:30pm
Enhance UEFI on RISC-V	August 24 4:20pm
Enabling compliance test for RISC-V BRS	August 24 3pm
The ACRN/RISC-V project: embedded hypervisor design and status update	August 24 5pm

intel®