ARMv8-A Scalable Vector Extension (SVE) Simulation and Evaluation

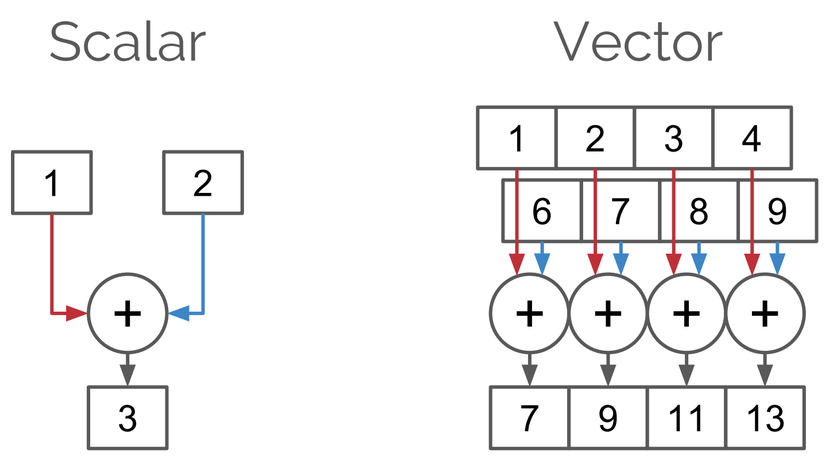
Hal Jones, supervised by Simon McIntosh-Smith

University of Bristol, Department of Computer Science

Introduction

Conventional (**scalar**) processor instructions perform a single calculation each. Multiple calculations require multiple instructions, and thus cycles.

Vector instructions allow a processor to repeat a mathematical operation across a set of data, in a single cycle. This technique is also known as "Single Instruction, Multiple Data" (**SIMD**).

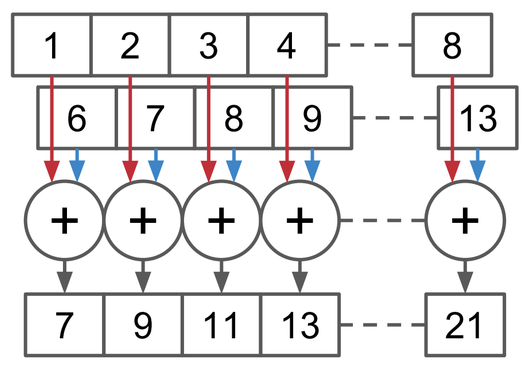


Scalable Vector Extension (SVE)

Current SIMD implementations have fixed vector lengths. Increasing the vector length requires a significant number of new instructions.

SVE is an extension to the ARMv8-A architecture that allows for hardware designers to specify the size of their vector registers. Instructions are **Vector-Length Agnostic**, and perform an operation over any vector length.

There are currently no production processors implementing SVE, so evaluating the performance of this extension is not easily possible.



Research Goals

Simulate SVE

Create a **64-bit ARM processor simulator** capable of running compiled AArch64 binaries, with SVE support

Benchmark Support

Progressive implementation of simulator features will make it possible to execute **high-performance benchmarks** such as BUDE, TeaLeaf and CloverLeaf, compiled to include SVE instructions.

Performance Evaluation

Configuring the simulator to match existing an high-performance ARM processor and **measuring benchmark performance** will allow for an approximation of SVE benefits. Varying simulator properties such as cache sizes and vector length can be used to explore the benefits and limitations of SVE across a range of hardware.

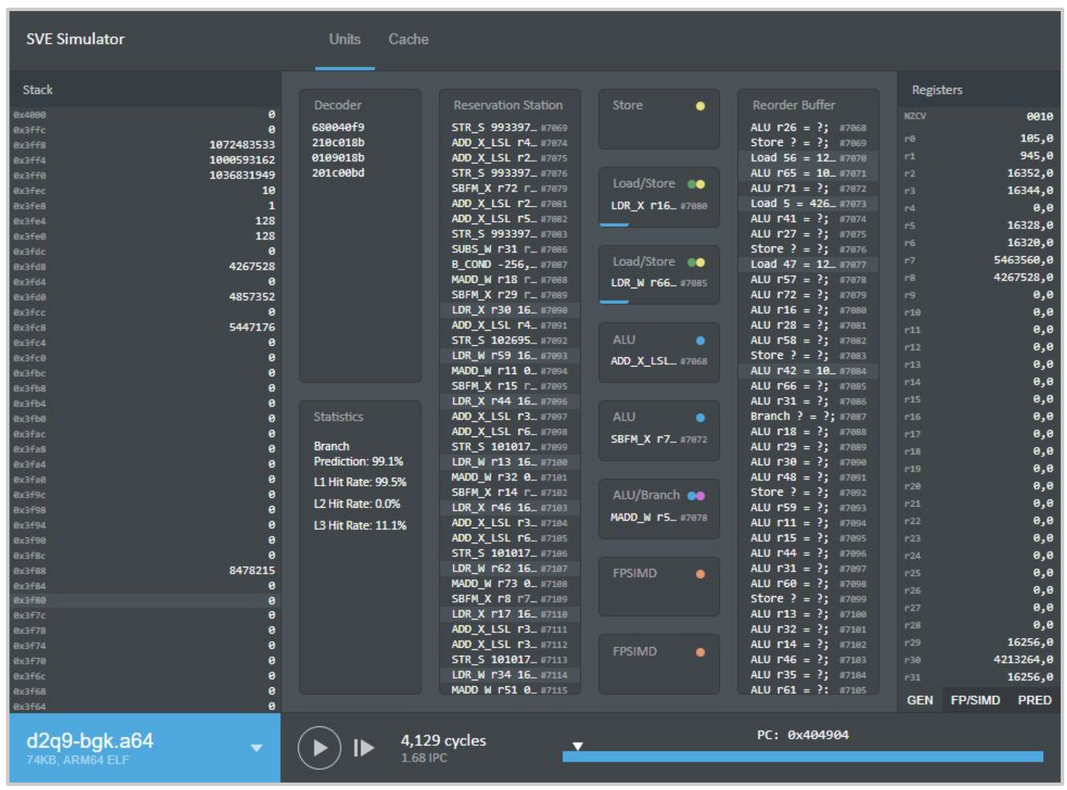
Project Progress

Processor Simulator

- Superscalar out-of-order with speculative execution
- Capable of **decoding and executing AArch64 binaries**
- Configurable multi-level cache
- Advanced SIMD (**ASIMD**) and SVE support
- Graphical processor/cache visualiser for debugging and execution inspection
- Configured to match a **Cavium ThunderX2** core

Supported Benchmarks

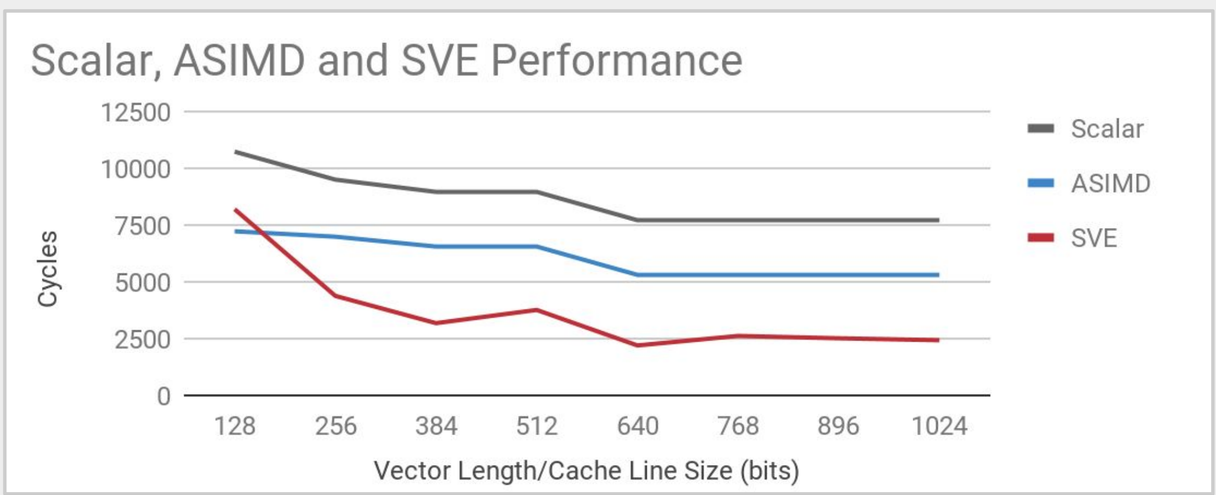
- Supports several benchmarks such as BUDE and Lattice Boltzmann, at various optimisation levels
- Designed to be rapidly extensible to support further benchmarks



Preliminary Results

With the default 128-bit vector length, SVE performs marginally worse than ASIMD in most test benchmarks, likely due to vector predication overhead.

With a vector (and cache line) size increase, SVE code performs **significantly better** than ASIMD, with a 60% performance difference at 256 bits in the test case below. Further increasing the vector size sees diminishing returns.



Above: 1000 iterations of the *Daxpy* algorithm, compiled using *armclang*. Fewer cycles is better.

University of BRISTOL