SVE/SVE2 support in LLVM and GNU toolchains

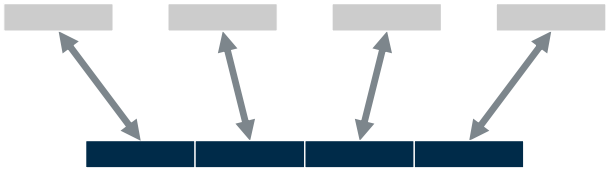
Linaro Connect SAN19-206

Ashok Bhat, Sr Product Manager

Sep 2019

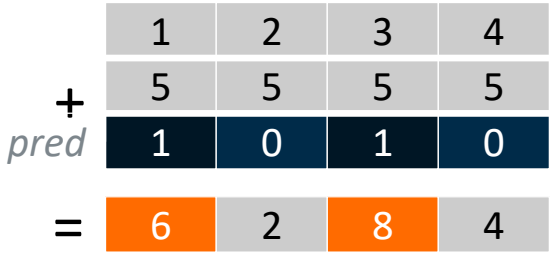
Scalable Vector Extension (SVE) - Recap

An optional vector extension to the Armv8-A architecture targeting HPC



Gather-load and scatter-store

Loads a single register from several non-contiguous memory locations.



Per-lane predication

Operations work on individual lanes under control of a predicate register.

```
for (i = 0; i < n; ++i)
```

INDEX i	n-2	n-1	n	n+1
CMPLT n	1	1	0	0

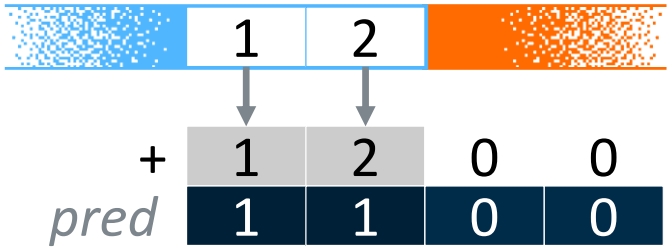
Predicate-driven loop control and management

Eliminate scalar loop heads and tails by processing partial vectors.

arm

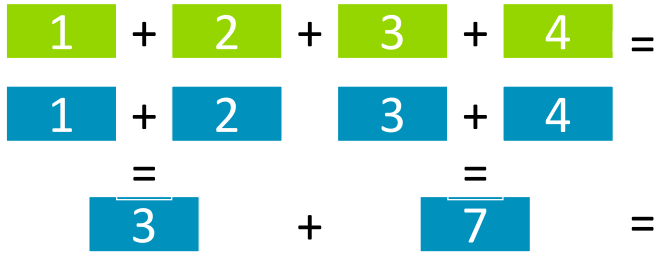
Scalable Vector Extension (SVE) - Recap

An optional vector extension to the Armv8-A architecture targeting HPC



Vector partitioning and software-managed speculation

First Faulting Load instructions allow memory accesses to cross into invalid pages.



Extended floating-point horizontal reductions

In-order and tree-based reductions trade-off performance and repeatability.

arm

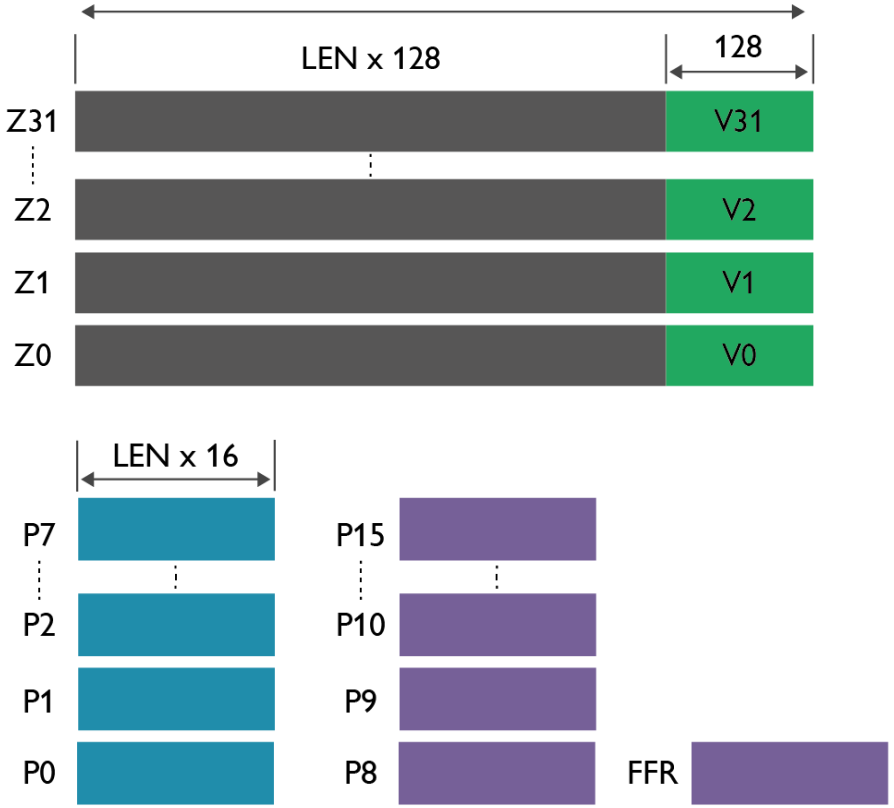
SVE registers

Scalable vector registers

- Z0-Z31 extending NEON's 128-bit V0-V31.
- Packed DP, SP & HP floating-point elements.
- Packed 64, 32, 16 & 8-bit integer elements.

Scalable predicate registers

- P0-P7 governing predicates for load/store/arithmetic.
- P8-P15 additional predicates for loop management.
- FFR first fault register for software speculation.



arm

Scalable Vector Extension v2 (SVE2)

Scalable Data-Level Parallelism for more applications

Built on SVE

Improved scalability

Vectorization of more workloads

Built on the SVE foundation

- Scalable vectors with hardware choice from 128 to 2048 bits.
- Vector-length agnostic programming for “write once, run anywhere”.
- Predication and gather/scatter allows more code to be vectorized.
- Tackles some obstacles to compiler auto-vectorisation.

Scaling single-thread performance to exploit long vectors

- SVE2 adds NEON™-style fixed-point DSP/multimedia plus other new features.
- Performance parity and beyond with classic NEON DSP/media SIMD.
- Tackles further obstacles to compiler auto-vectorization.

Enables vectorization of a wider range of applications than SVE

- Multiple use cases in Client, Edge, Server and HPC.
  - DSP, Codecs/filters, Computer vision, Photography, Game physics, AR/VR, Networking, Baseband, Database, Cryptography, Genomics, Web serving.
- Improves competitiveness of Arm-based CPU vs proprietary solutions.
- Reduces s/w development time and effort.

SVE2 enhancements

NEON-style “DSP” instructions

- Trad NEON fixed-p, widen, narrow & pairwise ops
- Fixed-point complex dot product, etc. (LTE)
- Interleaved add w/ carry (wide multiply, BigNums)
- Multi-register table lookup (LTE, CV, shuffle)
- Enhanced vector extract (FIR, FFT)

Cross-lane match detect / count

- In-memory histograms (CV, HPC, sorting)
- In-register histograms (CV, G/S pointer de-alias)
- Multi-character search (parsers, packet inspection)

Non-temporal Gather / Scatter

- Explicit cache segregation (CV, HPC, sorting)

Bitwise operations

- PMULL32→64, EORBT, EORTB (CRC, ECC, etc.)
- BCAX, BSL, EOR3, XAR (ternary logic + rotate)

Bit shuffle

- BDEP, BEXT, BGRP (LTE, compression, genomics)

Cryptography

Optional

- AES, SM4, SHA3, PMULL64→128

Miscellaneous vectorisation

- WHILEGE/GT/HI/HS (down-counting loops)
- WHILEWR/RW (contiguous pointer de-alias)
- FLOGB (other vector trig)

arm

SVE/SVE2 support in the toolchain

What does it involve?

What needs to be supported?

- Asm/disasm support
- Intrinsic (ACLE) support
- Auto-vectorization

Which components need to support?

- Compiler, Assembler/Disassembler
- Debugger
- Libraries

SVE/SVE2 support in GNU toolchain

Assembler/Disassembler

- SVE supported in binutils.
- SVE2 support upstreamed, will appear in binutils 2.33 version (2019 H2 release).

Intrinsics (ACLE)

- Contentious due to sizeless type needed in C/C++ frontend. Not supported in GCC 9, planned for GCC10.

Compiler auto-vectorization

- SVE auto-vectorization first supported in GCC8, with further improvements in GCC9. More planned for GCC10.
- Basic SVE2 auto-vectorization planned for GCC 10.

Debug support

- SVE support since GDB 8.2 release in Sep 2018.
- SVE2 will be automatically supported after SVE2 has been enabled in binutils

Libraries

- Plan to add SVE vector math routines first. SVE String routines being considered.
- All routines will be first contributed to Arm Optimized Routines project and then to GLIBC

SVE/SVE2 support in LLVM toolchain

Assembler/Disassembler

- LLVM9 supports SVE and SVE2 assembly and disassembly.

Intrinsics (ACLE)

- Contentious due to sizeless type needed in C/C++ frontend. Currently under discussion in upstream clang project.

Compiler auto-vectorization

- Auto-vectorization is not supported in LLVM9. However, significant progress was made in LLVM9 as scalable vector type support was merged.

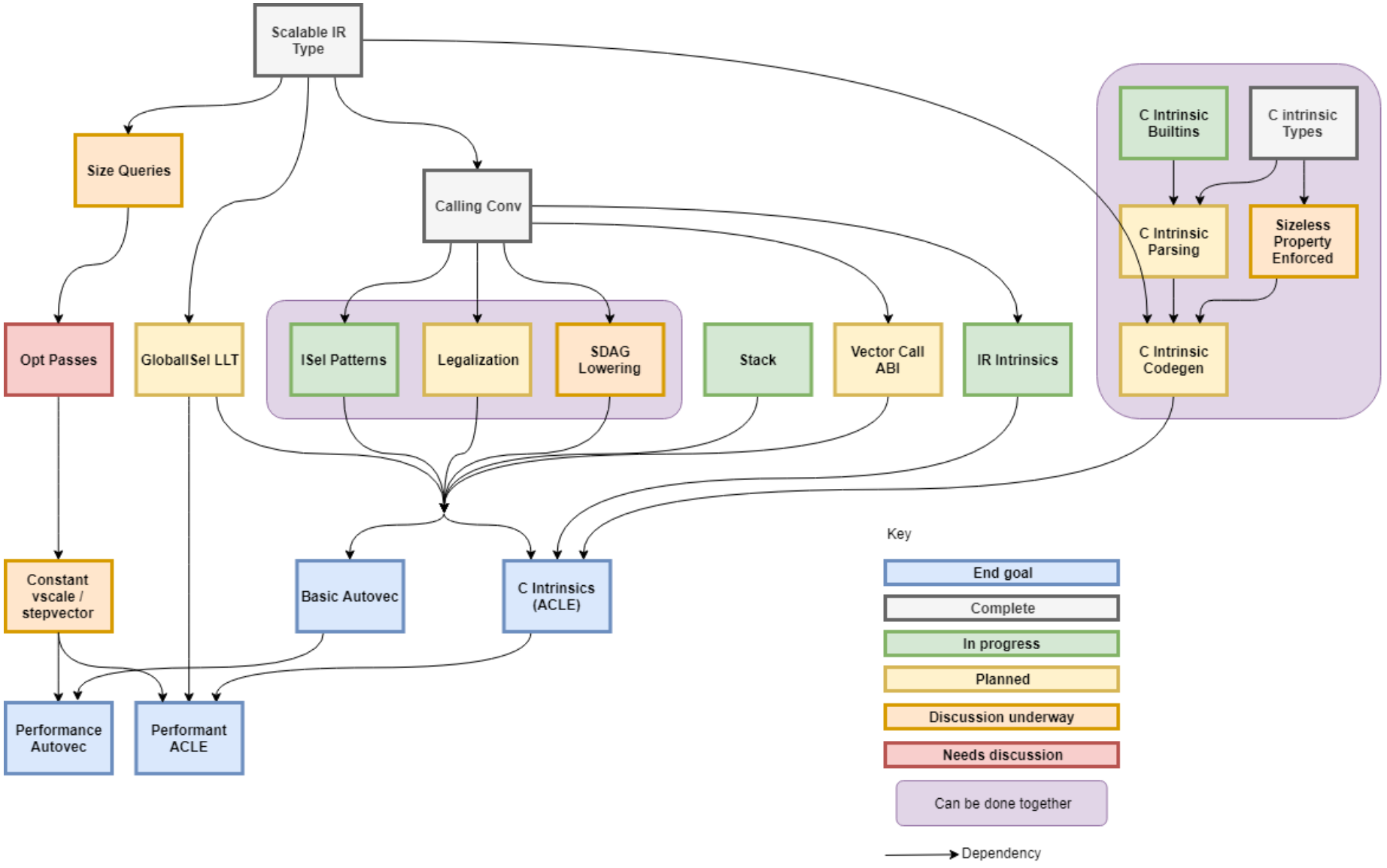
Debug support

- Support for SVE/SVE2 in LLDB is ongoing. More details in SAN19-204 - Road to SVE enablement in LLDB talk.

Libraries

- SVE routines in Arm Optimized Routines will allow commercial toolchains to use optimized SVE libraries. More details on Optimized Routines project in SAN19-511 - Optimized Routines talk.

LLVM SVE intrinsic and auto-vectorization status/plan



arm

SVE/SVE2 support in other projects/tools

Open source

- QEMU
  - SVE support since version 3.1
  - SVE2 support planned/work-in-progress by Linaro
- Valgrind
  - No support for SVE
  - Work being planned by Linaro
- DynamoRIO - Dynamic Instrumentation Tool Platform
  - SVE support in progress by Arm

Commercial tools

- Arm Fast Models (for bare-metal simulation)
- ArmIE (for process simulation)
- Arm Compiler (for bare metal use-case)
- Arm Compiler for Linux (for Linux user-space use-case)

SVE/SVE2 support in GNU and LLVM toolchains

Sufficient support now in the toolchains to get started

- Assembly/Disassembly: now supported in both toolchains.
- Auto-vectorization : GCC has support for SVE auto-vectorization since GCC8, while progress is being made in adding such a support to LLVM.
- Intrinsics (ACLE): Contentious due to sizeless type in C/C++ language frontend. Being worked on in both toolchains.

More progress expected in the future as SVE hardware becomes available

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה