Scalable vector multimedia optimisations
RISC-V V and ARM SVE2 extensions introduction

Rémi Denis-Courmont
Remlab Tmi
Ixelles, Belgium, 4th February 2023
1. History

2. From fixed-sized to variable-length

3. ARM Scalable Vector Extension

4. RISC-V Vectors
Attendees advisory

Disclaimer
The opinions expressed therein solely represent the personal views of the author.
Attendees advisory

Disclaimer

The opinions expressed therein solely represent the personal views of the author.

- I speak fast.
- I do not articulate well.
Attendees advisory

Disclaimer

The opinions expressed therein solely represent the personal views of the author.

- I speak fast.
- I do not articulate well.

If you did not understand...

Do interrupt me if needed!
Who am I?

- 16th FOSDEM attendance (since 2004)
Who am I?

- 16th FOSDEM attendance (since 2004)...
- 1st FOSDEM presentation!
- Not relevant to this presentation.
Outline

1. History
2. From fixed-sized to variable-length
3. ARM Scalable Vector Extension
4. RISC-V Vectors
What is this?
You may know if older than me.
Planet of Death

You may know if my age.
Single Instruction Multiple Data

- x86
  - 64 bits: MMX (1997)
Single Instruction Multiple Data

- x86
  - 64 bits: MMX (1997)
  - 128 bits: SSE (1999)
Single Instruction Multiple Data

- **x86**
  - 64 bits: MMX (1997)
Single Instruction Multiple Data

- **x86**
  - 64 bits: MMX (1997)
  - 256 bits: AVX2 (2011)
Single Instruction Multiple Data

- x86
  - 64 bits: MMX (1997)
  - 256 bits: AVX2 (2011)
Single Instruction Multiple Data

- **x86**
  - 64 bits: MMX (1997)
  - 256 bits: AVX2 (2011)

- **ARM**
Single Instruction Multiple Data

- **x86**
  - 64 bits: MMX (1997)
  - 256 bits: AVX2 (2011)

- **ARM**
  - 128 bits: ARMv8 A64 AdvSIMD, also a.k.a. NEON (2012)

- **RISC-V**
  - **ENOSYS**

Need to rewrite assembler every time.
Outline

1. History
2. From fixed-sized to variable-length
3. ARM Scalable Vector Extension
4. RISC-V Vectors
Dear CPU, what is your vector length?

```assembly
csrr t0, vlenb /* Vector LENGTH in Bytes */
```
Vector length

Dear CPU, what is your vector length?
```
csrr t0, vlenb /* Vector LENgth in Bytes */
```

Dear CPU, how many elements can you process?
```
csrr t0, vlenb
slri t0, t0, #2 /* 32-bit elements */
```
Vector length

Dear CPU, what is your vector length?

csrr t0, vlenb /* Vector LENgth in Bytes */

Dear CPU, how many elements can you process?

csrr t0, vlenb
slri t0, t0, #2 /* 32-bit elements */

1. Write main loop.
2. Unroll main loop.
3. Deal with edges.

That is how Clang vectorisation does it...
Vector length

Possible answers

- A power of two!

1 except embedded RISC-V
Vector length

Possible answers

- A power of two!
- 128 bits: guaranteed minimum\(^1\).
- 256, 512 bits: silicon designs announced, yet to ship.
- 1024 bits, even 4096 proposed in (RISC-V) simulations.
- 65536 bits: syntactic maximum (RISC-V).

\(^1\)except *embedded* RISC-V
Predication

- Not *completely* new concept
- Essential to variable vector length programming model
Predication

- Not *completely* new concept
- Essential to variable vector length programming model
- Vector of boolean
- Selects loaded/modified/stored elements

**ARMv9 example**

```assembly
MOV  x10, xzr
B    2f
1:
    ...
2:  WHILELT p0.s, x10, x0
    B.FIRST 1b
```
Unrolling

- Ill fit with predication
- Vector processing ≠ SIMD
- Just don’t unroll...
Unrolling

- Ill fit with predication
- Vector processing $\neq$ SIMD
- Just don’t unroll...
- ARM: ”SVE streaming mode”
  - Higher latency
  - Larger vectors (potentially)
  - Higher throughput
- No over-alignment required! Yay!
Outline

1. History
2. From fixed-sized to variable-length
3. ARM Scalable Vector Extension
4. RISC-V Vectors
Original SVE pretty useless for multimedia.
Original SVE pretty useless for multimedia.
SVE2 copies most NEON mnemonics.
Just insert the predicate register operand!
Famous last words.
Pick:

1. 1 of 10 WHILEExx instruction: WHILELT, WHILELO, ...
2. a predicate register,
3. the element size: \( B, H, S \) or \( D \).
4. a branch condition: B.FIRST, B.LAST...
Pick:

1. 1 of 10 WHILEExx instruction: WHILELT, WHILELO, ...
2. a predicate register,
3. the element size: $B$, $H$, $S$ or $D$.
4. a branch condition: B.FIRST, B.LAST...

- Remaining elements $\rightarrow$ Predicate register
- Predicate register $\rightarrow$ Condition flags
- Subtracted count $\rightarrow$ Output GP register
SVE

Pick:

1. 1 of 10 WHILEExx instruction: WHILELT, WHILELEO, ...
2. a predicate register,
3. the element size: $B$, $H$, $S$ or $D$.
4. a branch condition: B.FIRST, B.LAST...

- Remaining elements $\rightarrow$ Predicate register
- Predicate register $\rightarrow$ Condition flags
- Subtracted count $\rightarrow$ Output GP register

Stop pretending AArch64 is a RISC.
Processor feature detection
It would be too easy without it.

- Preprocessor: defined(__ARM_FEATURE_SVE2)
- Bare metal: ID_AA64*EL1 register fields
Processor feature detection
It would be too easy without it.

- Preprocessor: defined(_ARM_FEATURE_SVE2)
- Bare metal: ID_AA64_*_EL1 register fields
- Linux: bits from AT_HWCAP2 auxillary vector entry
  - HWCAP2_SVE2 is probably what you want
  - HWCAP2_SVEPMULL
  - HWCAP2_SVEBITPERM
  - HWCAP2_SVE2P1

Examples
#include <sys/auxv.h>
(getauxval(AT_HWCAP2) & HWCAP2_SVE2)
Processor feature detection

It would be too easy without it.

- Preprocessor: defined(\_\_ARM\_FEATURE\_SVE2)
- Bare metal: ID\_AA64\_*\_EL1 register fields
- Linux: bits from AT\_HWCAP2 auxillary vector entry
  - HWCAP2\_SVE2 is probably what you want
  - HWCAP2\_SVEPMULL
  - HWCAP2\_SVEBITPERM
  - HWCAP2\_SVE2P1

Examples

```
#include <sys/auxv.h>
(getauxval(AT\_HWCAP2) & HWCAP2\_SVE2)
```

- Other OSes: lol
Availability

- Specifications
  - SVE (2016)
Availability

- Specifications
  - SVE (2016)... explicitly not intended for multimedia payloads
  - SVE2 (2019)
  - SME / Scalable Matrix Extension (2021)
  - Streaming SVE
Availability

- Specifications
  - SVE (2016)...explicitly not intended for multimedia payloads
  - SVE2 (2019)
  - SME / Scalable Matrix Extension (2021)
  - Streaming SVE

- Hardware
  - Cortex-X2, Cortex-A510, Cortex-A710
  - Arm DynamIQ-110 cluster (2022)
Availability

- Specifications
  - SVE (2016) . . . explicitly not intended for multimedia payloads
  - SVE2 (2019)
  - SME / Scalable Matrix Extension (2021)
  - Streaming SVE

- Hardware
  - Cortex-X2, Cortex-A510, Cortex-A710
  - Arm DynamIQ-110 cluster (2022)
  - Samsung Exynos 2200
  - Qualcomm SM8450 Snapdragon 8 Gen 1
Outline

1. History
2. From fixed-sized to variable-length
3. ARM Scalable Vector Extension
4. RISC-V Vectors
Predication
Not sure if simpler or more intricate

**Vector configuration**

```
vsetvli t0, a4, e16, m1, ta, ma
```

- \(a4\) = available elements (input)
- Output operand: \(t0\) = vector length (output)
- Element size: \(e16\) \(\leftrightarrow\) 16 bits
- Group size: \(m1\) \(\leftrightarrow\) 1 vector \(\leftrightarrow\) no grouping
- Tail mode: \(ta\) agnostic \(\leftrightarrow\) don’t care
- Mask mode: \(ma\) agnostic \(\leftrightarrow\) don’t care
Registers

- Prefer greatest power-of-two multiple-numbered vectors
  - grouping and segmentation require aligned numbers
- FP registers \(\neq\) Vectors
  - more registers for hybrid scalar/vector functions

**Warning**

Mind the FP calling conventions!
Bit shuffling

- Segmented loads & stores up to 8 structures (ARM can do up to 4 only)
- GP register-strided loads & stores
- ... including negative strides.

Example

```bash
# Load a column of 16-bit samples
# at [a0] with pitch a4 in vector v8.
vlse16.v v8, (a0), a4
```
Bit shuffling

- Segmented loads & stores up to 8 structures (ARM can do up to 4 only)
- GP register-strided loads & stores
- ... including negative strides.

Example

```c
# Load a column of 16-bit samples
# at [a0] with pitch a4 in vector v8.
vlse16.v v8, (a0), a4
```

- But... no vector↔vector transpose/zip
Processor feature detection

- **Preprocessor:**
  - Element size: `__riscv_v_elen_fp = 32` or `64`
  - `__riscv_vector` → `elen ≥ 64` bits
  - Vector length: `__riscv_zvl{32,64,128,...}b`
  - `__riscv_vector` → `VL ≥ 128` bits

- **Hardware:**
Processor feature detection

- Preprocessor:
  - Element size: \_\_riscv\_v\_elen\_fp = 32 or 64
  - \_\_riscv\_vector \Rightarrow \textit{elen} \geq 64 \text{ bits}
  - Vector length: \_\_riscv\_zvl\{32,64,128,...\}b
    - \_\_riscv\_vector \Rightarrow VL \geq 128 \text{ bits}
- Hardware: DeviceTree cpu node property
- Linux: bit 21 from AT\_HWCAP auxillary vector entry

Examples

```c
#include <sys/auxv.h>
(getauxval(AT\_HWCAP) \& (1\text{U} \ll (’V’ - ’A’)))
```
Availability

- Specifications
  - RISC-V "V" Vector extension version 1.0 (ratified 2021)
  - Not integrated in RISC-V unprivileged specification yet
Availability

- Specifications
  - RISC-V "V" Vector extension version 1.0 (ratified 2021)
  - Not integrated in RISC-V unprivileged specification yet

- Hardware
  - Open-source designs exist (but...)
  - T-Head (Alibaba): draft version 0.7.1 only so far
  - SiFive: several IPs announced, not sold yet
  - Andes: AX45, not sold yet
Further references

- Arm Architecture Reference Manual, ARMv8-A
- Arm SVE supplement
- Arm SME supplement
- RISC-V Vector extension version 1.0.
- FFmpeg source code.
Any questions?