Flexible vectors update

Petr Penzin

Intel

June 20, 2023

Agenda

- Recap of the proposal
- Changes to the proposal since the previous update
- Phase 2?

Brief overview of the proposal

New types and instructions

- $vec. < type >$ – separate vector types for different lane types, size defaults to maximum supported by hardware
  - $i8$, $i16$, $i32$, $i64$ – integer
  - $f32$, $f64$ – floating point
- $vec. < type > . < op >$ – same lane-wise operation as in $simd128 < op >$, applied to vector of $vec. < type > .length$

  For example, $vec.f32.mul$ is identical to $f32x4.mul$ on a 4-lane vector, $vec.i32.add$ to $i32x4.add$, and so on
  - $vec. < type > .length$ gets number of elements in corresponding vector type length is fixed during execution and is multiple of 128 bits

Rationale

- Realizing substantial performance gains from tapping into more advanced SIMD instruction sets
- Certain algorithms' benefits are limited on 128-bit SIMD
- Instructions are meant to be direct translated to hardware SIMD, in spirit of existing SIMD standard

Proposal update since previous presentation

New instructions and semantics:

- Lane index bounds checking
- Shuffle operations

Lane index bounds checking

Old ’extract lane’ and ’replace lane’ were taking immediate arguments, which could be out of bounds.
Solution:

- Change existing ’extract’ and ’replace’ to take regular integer arguments, but trigger a bounds check at runtime
- Fallback operations with immediate indices within 128 bits (no bound checking)
- New ’extract’ and ’replace’ operations that take their argument modulo the lane count

Shuffle operations

New operations added (already had lane shift):

- Concat parts of input vectors
- Select odd lanes from one vector and even - from the other one
- Reverse lane order
- Duplicate odd lanes into lower even lanes

Phase 2 entry requirements

- Precise and complete overview document is available in a forked repo around which a reasonably high level of consensus exists.
- *Updates to the actual spec document, test suite, and reference interpreter are NOT yet required.*

Current status

- Added all classes of operations
- Expected some tweaks based on performance

Poll for Phase 2

Poll?

Thank you