# WebAssembly SIMD Sync 11/13/2020

**Attendees:**
Thomas Lively
Dan Weber
Arun Purushan
Lars Hansen
Andrew Brown
Rich Winterton
Jan Wassenberg
Marat Dukhan
Johnnie Birch

**Agenda:**

**1. Status updates**

TL: Feature detection to its own phase 1 proposal at the last CG meeting.

DW: Any news on Marat's question about register spilling from last time?

LH: We should lift this question and others to be issues in the new proposal repo.

**2. New integer sign extension operations (#395)**

[DW presenting Slides -- https://docs.google.com/presentation/d/1zceZRZFZbxZUIhAVKmnPMt74tq7hEQ09pZJiEibxRWM/edit?usp=sharing]

MD: What if we did twice-wide addition and subtraction instead? These have poor lowering on x64.

DW: No, that wouldn't work because color space conversion requires multiplying by -1, so can't even use widening multiplication, either.

MD: Another possibility would be shuffling that can set some lanes to 0.

DW: Yes, that would be useful.

TL: Would there be anything bad that comes out of relaxing the current shuffle to allow out of bounds indices to zero lanes?

JW, MD, LH: Yes, Spidermonkey could do this specialization of shuffles.

JW: These instructions are useful (except maybe 8->64)

DW:  I just added 8->64 for completeness.

AP: I would be opposed to adding that just for completeness, but would be ok adding it if we had a use case. What instruction set did you evaluate for x64?

DW: SSE3 and above, although everything except for 8->64 and 16->64 could be supported by SSE2. LLVM actually spills for those.

MD: XNNPACK has only one use case, loading from memory signed 8->32, so a little different. Loading from memory wouldn't be as nice on ARM.

… Looks at Godbolt LLVM-MCA analysis …

MD: Couldn't this be better [didn't catch how]

DW: Originally was going to add a multivalue version to return multiple widened vectors at once.

LH: No

TL: Dan, could you prepare a benchmark if we prototyped these?

DW: Yes.

TL: Sounds like we are skeptical of the 64 bit ops, what about others?

MD: Skeptical of all lowering on ARM

DW: Assuming masks are shared

MD: Probably not how V8 works

TL: We should ask Zhi (not present)

**3. Discuss further spec freezes**

- Note: interest in allowing changes for perf issues (#190, #189, #93, etc.)

MD: We should seal the proposal by Dec 31, except for some fixes.

Arun to make tracking issue for remaining issues lacking.

Marat to make tracking issue for looking at untested/missing data types for existing instructions.

LH, MD, TL: Not opposed to keeping what we have and moving forward (referring to issue about removing ne instructions).

DW: There doesn't seem to be a cost to keeping them.

AB: There is still a performance cost.

LH: It's about giving users who know what they're doing options.

AB: We could add a new pseudo variant of conversions.

MD: I wouldn't want to go through that again.

LH: Worried about getting hacks by the CG.

AP: Would want to try with the new variants and see what the CG thinks.

TL: I would prefer to document and ship in the meantime rather than do more work that would have to happen before shipping.

AB: Seems like there are three options:
- add new variants to non-performant instructions (a la min/max -> pmin/pmax)
- just document the non-performant instructions
- remove the non-performant instructions

JB: Why not just remove ne? Principled all or none approach?

TL: Yes, and no one wants to revisit all of them.

RW: Fine with just documenting

LH: Don't close the ne issue, rename it since it is important

DW: Renumbering?

TL: Maybe, you would have to pin your Emscripten version.

AP: Does feature detect support checking if certain operation is fast on a given arch? Without that developers will not have a way to avoid known inefficient instructions on their architectures of choice

TL: Technically is possible to do that, but will expose architectural information and CG might have opinions against it. Doest have to block the simd proposal, as feature detection has been spun off, it can continue to be discussed. Its an open topic