Proposal WebAssembly SIMD Modification

Currently as proposed there is an instruction defined in the WASM SIMD ISA as follows.

Instruction – i8x16.mul which is a register to register operation that takes 16 8 bit integers and multiplies them together resulting in an 8 bit value. If the distribution of the integers it flat this would result in approximately 50 percent of the instructions with overflow. This instruction is not too useful in real life applications that is known. This instruction could be left in the ISA mix but for IA would require a significant amount of instructions and would have to require overflow handling and specifications to achieve cross platform compatibility. This really is messy for no known applications. This overflow handling is currently not defined in the WASM SIMD specification.

Proposal is to remove this instruction as known applications don't use i8x16.mul functionality and add the widening load integer instructions. This was previously proposed as a candidate in bullet two back in November last year.

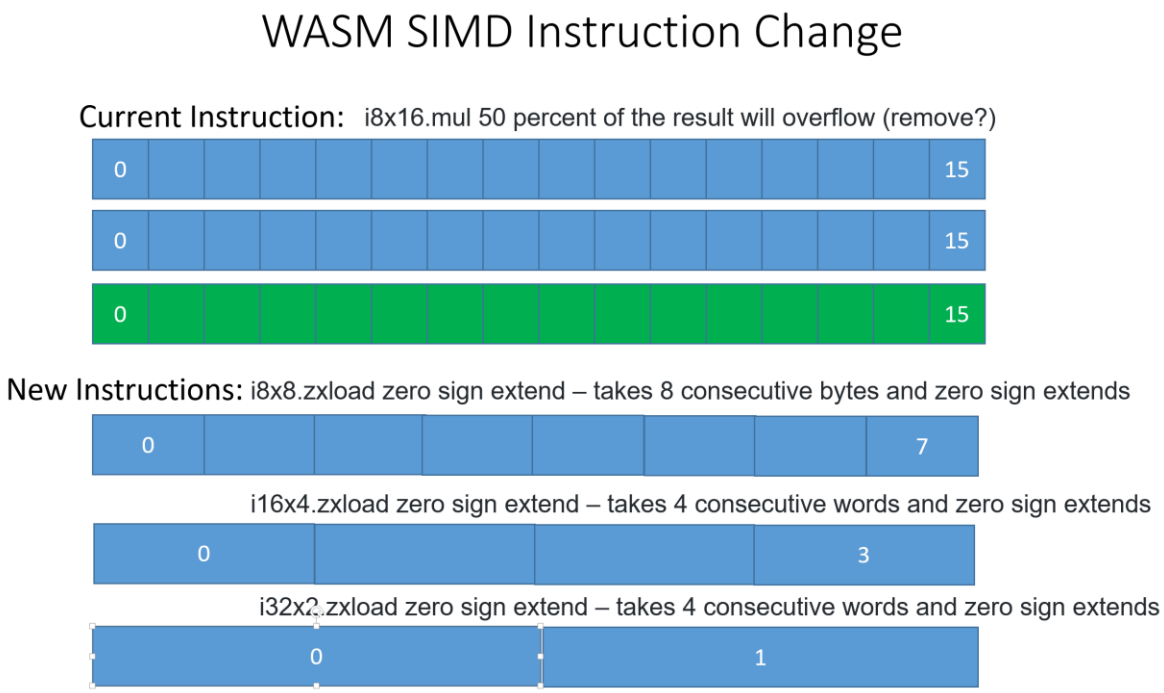
Candidates

- Horizontal pairwise addition, integer and floating point
- Widening integer conversion, signed and unsigned
- Saturating narrowing integer conversion

Proposed new instructions:

In conjunction with the potential change of the i8x16.mul removal it is proposed to create an instruction that would have six new load instructions to make integer multiplies easier. The first is i8x16. This would make i8, i16, i32 multiplies useful and more practical for applications such as machine learning, image compression and video and rendering data processing.

The new instructions would take consecutive integers of the corresponding size and zero sign extend and sign extend the consecutive bytes, words or dword to the promoted size of signed or unsigned data. An example of zero sign extend is shown below:



Intel and ARM both have this capability by doing the following:

movzxbw,

movzxwd

movzxdq

movsxbw

movsxwd

movsxdq

ARM:

Example instruction	Description
`LDR X0, [X1]`	Load from the address in X1
`LDR X0, [X1, #8]`	Load from address X1 + 8
`LDR X0, [X1, X2]`	Load from address X1 + X2
`LDR X0, [X1, X2, LSL, #3]`	Load from address X1 + (X2 << 3)
`LDR X0, [X1, W2, SXTW]`	Load from address X1 + sign extend(W2)
`LDR X0, [X1, W2, SXTW, #3]`	Load from address X1 + (sign extend(W2) << 3)

So the new instructions for WASM would be defined as follows:

i8x8.zxload, i16x4.zxload, i32x2.zxload, i8x8.sxload, i16x4.sxload, i32x2.sxload

As a result of these new instructions a multiply can now be done without worrying about signed and unsigned overflow on 50 percent of the data it operates on.