WebAssembly Flexible Vectors Operations
Развитие концепции кросс-платформ SIMD

Пётр Пензин

16 июля 2020 г.

Содержание

- Архитектура SIMD
- SIMD в WebAssembly
- Векторные операции для WebAssembly

Архитектура SIMD

- Многопоточные вычисления – одна и та же операция выполняется на нескольких элементам одновременно
- Используется для мультимедийных приложений

Архитектура SIMD

- Многопоточные вычисления – одна и та же операция выполняется на нескольких элементам одновременно
- Используется для мультимедийных приложений
- Примеры: MMX, SSE, AVX*, Neon
  - 64 бит: MMX
  - 128 бит: SSE, Neon, MSA, AltiVec
  - 256 бит: AVX
  - 512 бит: AVX-512

WebAssembly SIMD

- "Общий знаменатель"расширений SIMD в портебительских процессорах[1]
- SIMT и векторные операции сознательно не включены
- Достаточно успешно поддерживается в V8, SpiderMonkey, и нескольких автономных движках[2]

[1] Презентация, 05.2017
[2] Wasm SIMD Implementation Status

Дальнейшее развитие?

- Процессора PC поддерживают более производительные расширения SIMD, но есть проблема с совместимостью

Дальнейшее развитие?

- Процессора PC поддерживают более производительные расширения SIMD, но есть проблема с совместимостью
- Есть примеры решения
  - Highway
  - System.Numerics.Vector in .NET

Ограничения

- Один набор виртуальных команд на разных платформах
- Тривиальный выбор машинных инструкций
- Частичная совместимость с Wasm SIMD

Альтернатива – ещё одно расширение WebAssembly SIMD

- Более узкая поддержка в железе
- Плохо совместимо с общей философией выбора "общего"набора команд
- Сильно усложняет выбор машинных операций

Flexible Vectors

Основная идея – операции аналогичные Wasm *simd128*, но без установленной длинны вектора

- Операции с линейной памятью работают с соседними ячейками, как и в *simd128*
- Максимальная длинна вектора устанавливается рантаймом

Типы данных и инструкции

Новые типы данных и инструкции

- $vec.<type>$ – разные типы данных для разных типов элементов
  - $i8$, $i16$, $i32$, $i64$ – целые числа
  - $f32$, $f64$ – действительные числа
- $vec.<type>.length$ – возвращает количество элементов в соответствующем типе

Типы данных и инструкции

Расширения инструкций

- $vec.<type>.<op>$ – та же потоковая операция что и $simd128<op>$, примененная к вектору длинны $vec.<type>.length$

  Например, $vec.f32.mul$ – то же что и $f32x4.mul$ применимо к вектору длинны 4, $vec.i32.add$ – $i32x4.add$, и т.д.

Пример

Сложение, $c = a + b$, sz – размер

```
(block $loop
  (loop $loop_top
    (br_if $loop (i32.lt (get_local $sz) (vec.f32.length)))
    vec.f32.load (get_local $a)
    vec.f32.load (get_local $b)
    vec.f32.add
    vec.f32.store (get_local $c)
    ;; Decrement $sz and increment $a, $b, $c
    (br $loop_top)
  )
)
(block $scalar_loop ;; Finish the remaining elements
```

Выбор машинных инструкций

- Шаблоны аналогичные *simd128*
- Не требует логики или сохранения глобального состояния

Дальнейшее расширение: векторные операции

Одна операция:

- $vec.<type>.set\_length$ – установить количество элементов в соответствующем типе

  Минимум между параметром этой операции и тем что поддерживает процессор.

Пример векторных операций

Vector addition, $c = a + b$, sz is the size

```
local $len i32
(block $loop
  (loop $loop_top
    (br_if $loop (i32.eq (get_local $sz) (i32.const 0)))
    (set_local $len (vec.f32.set_length (get_local $sz)))
    vec.f32.load (get_local $a)
    vec.f32.load (get_local $b)
    vec.f32.add
    vec.f32.store (get_local $c)
    ;; Decrement $sz by $len; increment $a, $b, and $c by $len
    (br $loop_top)
  )
)
```

Дальнейшее расширение: векторные операции

Достоинства:

- Уменьшает размер скомпилированного модуля
- Может работать с машинным SIMD с поддержкой масок

Дальнейшее расширение: векторные операции

Достоинства:

- Уменьшает размер скомпилированного модуля
- Может работать с машинным SIMD с поддержкой масок

Недостатки:

- Потеря производительности на машинах без масок
- Изменяемое глобальное состояние

Пока что экспериментальное предложение.

Текущее состояние проекта

1. Репозитория на Github
2. Начальный набор команд
3. Первоочередная цель – перенести операции из Wasm SIMD

Спасибо