

Bulk Memory Operations Implementation in WebAssembly

Роботу виконав : Сиротенко Олександр Романович

Науковий керівник : кандидат технічних наук Демківський Є.О.

Мета роботи

Дизайн та розробка підтримки так званих "масивних" операцій над пам'яттю у WebAssembly.

Проблематика

- Продуктивність сучасних веб-застосунків.
- Відносно мала швидкість обробки великих даних.
- Велика кількість атомарних викликів запису.
- Запис у пам'ять набагато дорожчий, ніж читання.

WebAssembly

“

WebAssembly – це бінарний формат інструкцій для стекової віртуальної машини.

Докладніше

- Спроектований як переносиме AS-дерево.
- Має два проміжні представлення.
- Декларується **швидке** та **безпечне** виконання коду.
- Кросплатформенний та відкритий [стандарт](#). [1]

Масивні операції над пам'яттю

Лінійний та швидкий запис великої кількості байт.

```
memset (*ptr, int value, size_t num);
```

```
memcpy (*dest, const *src, size_t num);
```

...у WebAssembly

- Специфікація WebAssembly *поки що* не визначає наявність масивних операцій.
- Існує офіційна [пропозиція](#) додати такі операції у стандарт. [2]
- Автор роботи є одним із розробників цієї пропозиції 😊
- Цільова віртуальна машина - **ChakraCore** від компанії Microsoft.

Очікуваний результат

- Підтримка швидких операцій над лінійною пам'яттю :

`memory.copy`, `memory.fill`, `memory.init`






- Підтримка швидких операцій над таблицею вказівників :

`table.copy`, `table.init`

- Пасивної преініціалізації сегментів пам'яті.

Деталі реалізації

Загальне

- Включити ключові слова в лексер, токени в парсер 
- Включити коди операцій для бінарного представлення 
- Генерувати нові intrinsic - інструкції для JIT-компілятора 
- Забезпечити перехоплення виключень  SOON
- Використання транзакційної пам'яті  SOON

Як обирався алгоритм пришвидшеного запису (1)

- ChakraCore вже має реалізацію таких операцій для JavaScript.
- **Проблема** : немає підтримки обробки пасток ОС (trap).

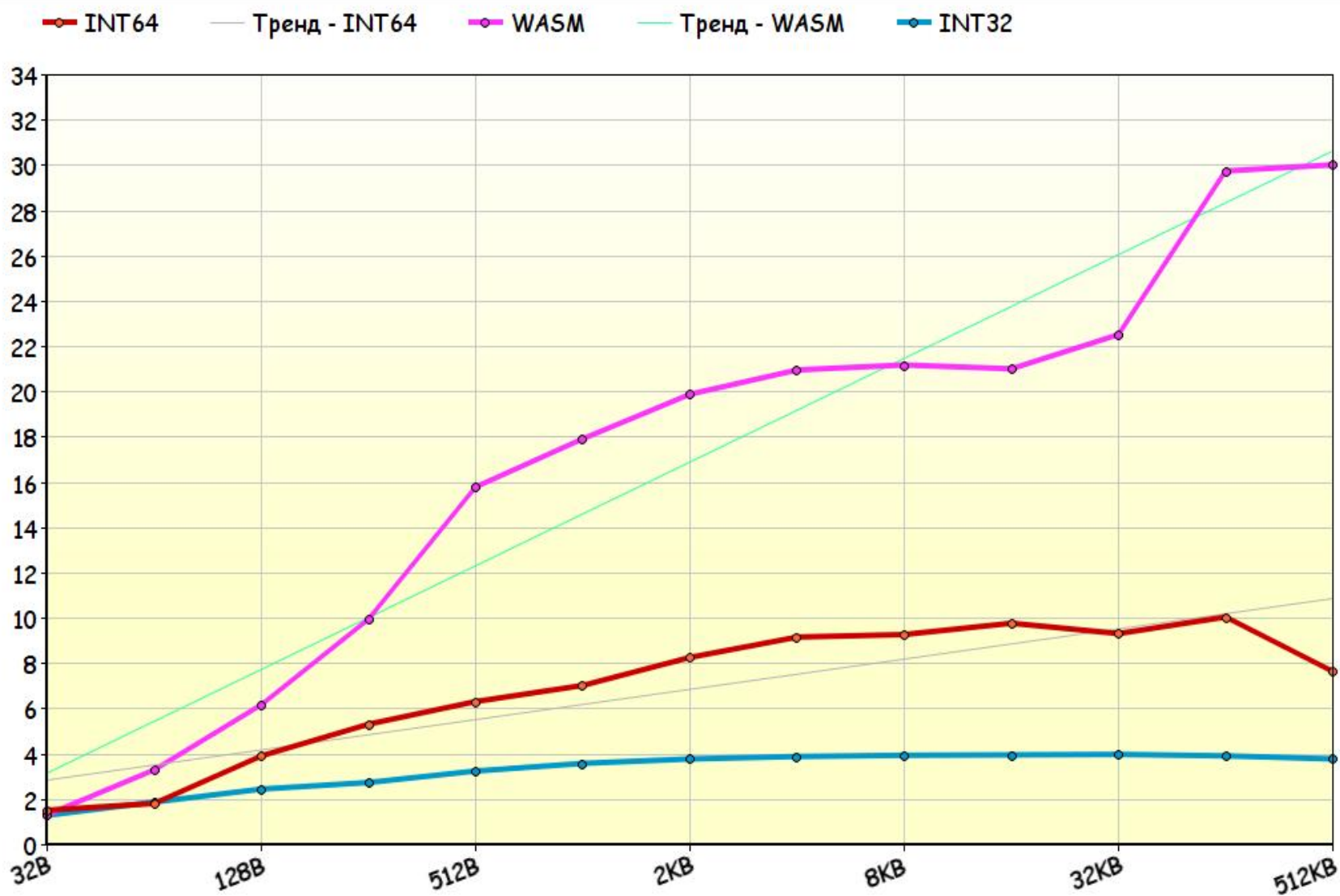
Як обирався алгоритм пришвидшеного запису (2)

- Перевіряємо необхідні умови : чи вдосталь пам'яті, чи доступна зараз пам'ять, звідки ми хочемо прочитати.
- Побайтово копіюємо в пам'ять призначення по вказівнику.
- Якщо трапляється помилка, передаємо її обробнику віртуальної машини.

Як обирався алгоритм пришвидшеного запису (3)

- Оптимізація, яка дозволяє минати загрузку значень у кеш.
- Додаткова перевірка на кратність кількості байт для запису до 128/256/512.
- Якщо так, то використовуємо інструкції `movaps` / `movntps` із набору SSE (SIMD).

THROUGHPUT CHART



ONE GIGABYTE OF INTEGERS



@FlyCreat1ve

Дякую за увагу!

Питання?