Санкт-Петербургский государственный университет

Кафедра системного программирования

Интринсики с использованием векторного расширения RISC-V “V” для компилятора OCaml

Азат Райнурович Габдрахманов, группа 21.Б07-мм

Научный руководитель: Д. С. Косарев, ассистент кафедры системного программирования

Санкт-Петербург
2024

Введение

- В OCaml нет векторизации
- Jane Street делает векторизацию для AMD и ARM
- Итак дорогая стоимость арифметических операций
- Векторное расширение RISC-V “V”

Существующие решения

Jane Street (OCaml)[1]**:**

- SSE для AMD
- NEON для ARM

C:

- RISC-V RVV[2]

[1] https://github.com/ocaml-flambda/flambda-backend

[2] https://github.com/riscv-non-isa/rvv-intrinsic-doc

Постановка задачи

Целью работы является внедрение в компилятор OCaml интринсиков, специфичных под RISC-V

Задачи:

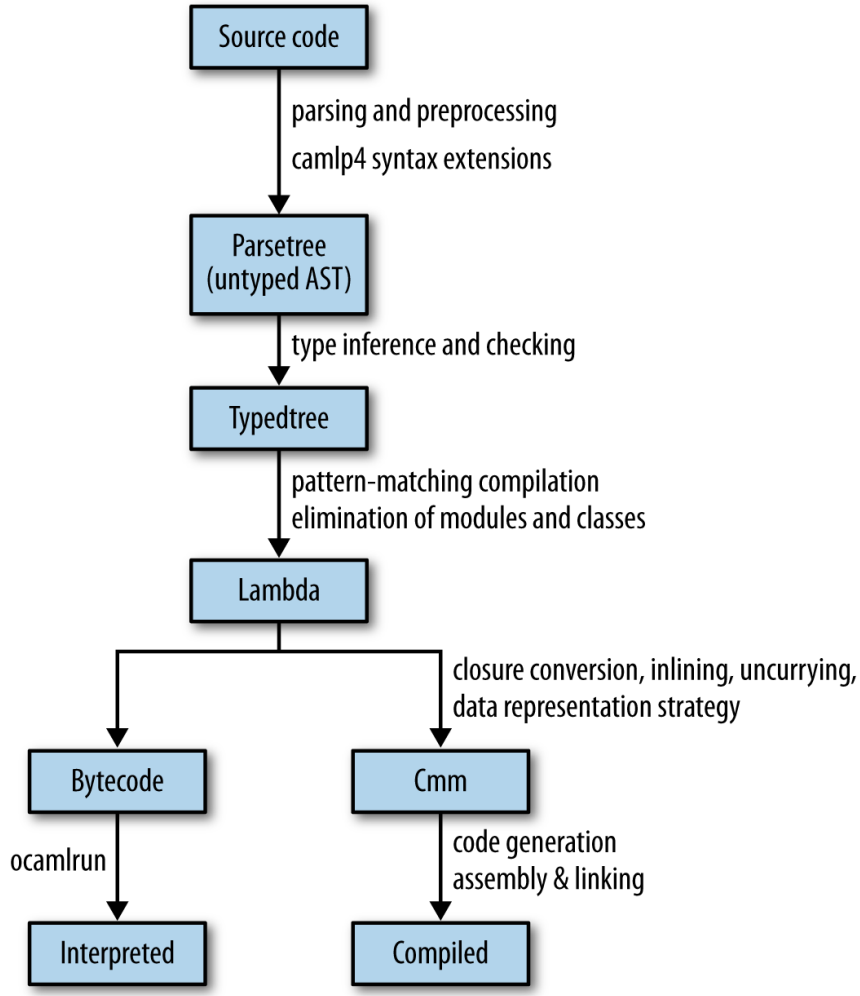
- Реализовать векторные интринсики
- Провести замеры

Векторизация

- В современных процессорах есть поддержка SIMD
- Векторизация — это хорошо
- Особенности OCaml

Компиляция в OCaml

- Два компилятора
- ocamlc - быстрая компиляция
- ocamlopt - быстрое исполнение



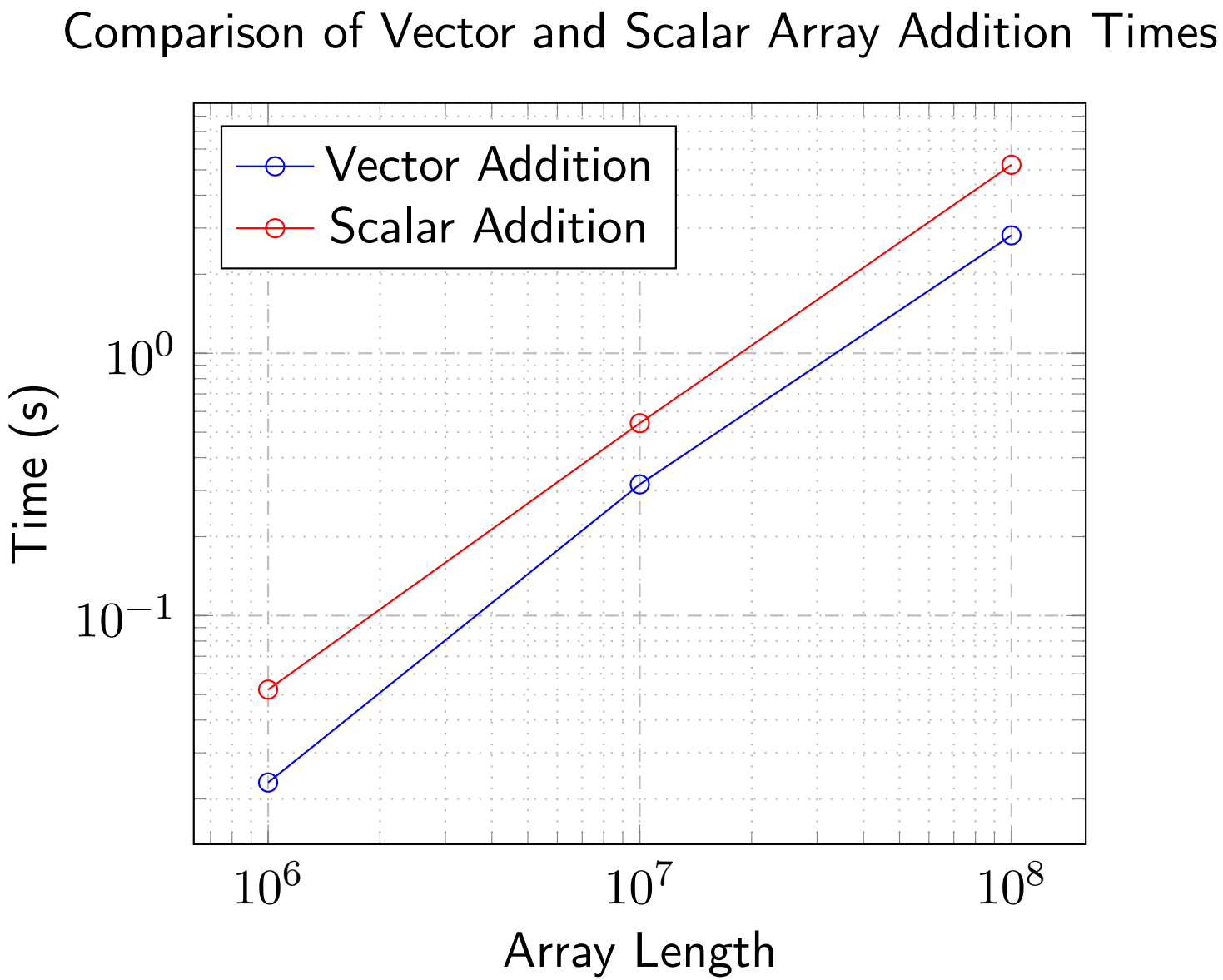
Подход

Интринсики из вызова внешней функции src $\rightarrow$ cmm $\rightarrow$ asm

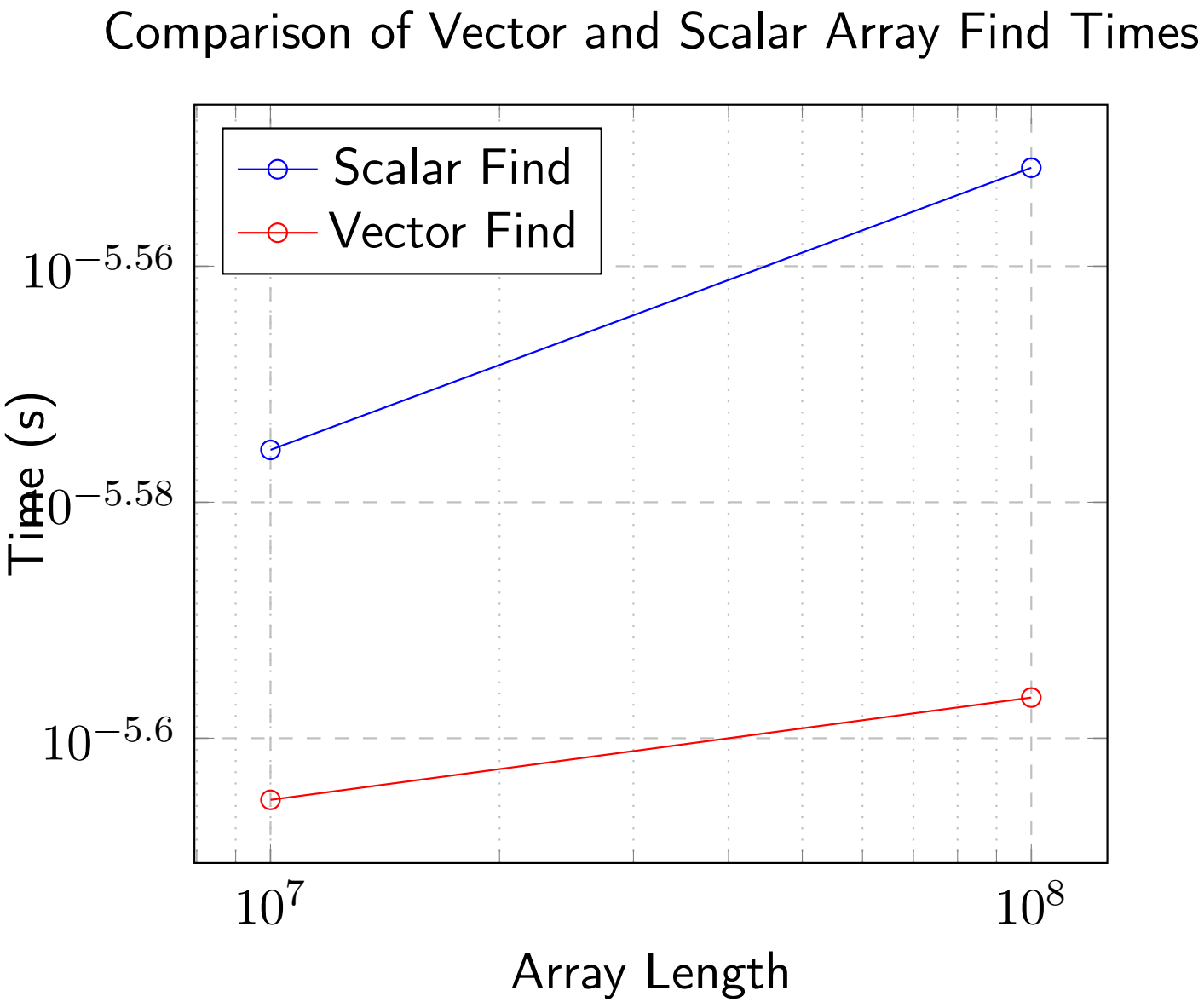
```
external vector_add : int array -> int array -> int array ->
 int -> int array = "vector_add"
let arr3 = vector_add arr1 arr2 arr3 (Array.length arr1)
```

```
(let  arr3/276
(extcall "vector_add"{vec.ml:13,11-34}
(load_mut val "camlVec")
    (load_mut val (+a "camlVec" 8)) 17 int,int,int -> val)
```

Замеры сложения



Замеры find


Comparison of Vector and Scalar Array Find Times
Scalar Find
Vector Find
$10^{-5.56}$
$10^{-5.58}$
$10^{-5.6}$
Time (s)
10^{7}
10^{8}
Array Length

Результаты

- Реализован интринсик для сложения векторов
- Реализован интринсик find
- Проведены замеры

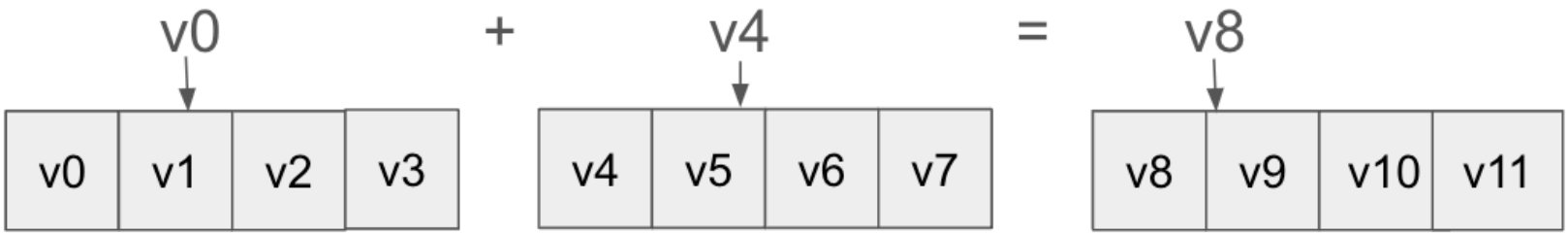
ссылка на репозиторий: `https://github.com/Azatic/ocaml`

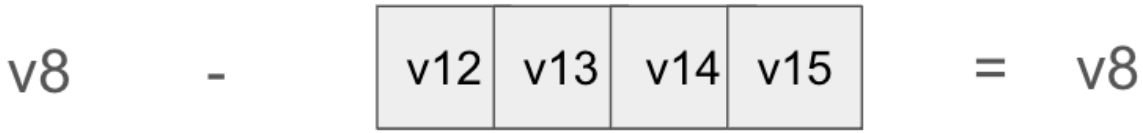
- Результаты связанные со сложением векторов были опубликованы в сборнике «Современные технологии в теории и практике программирования»

Накладные расходы в векторном случае

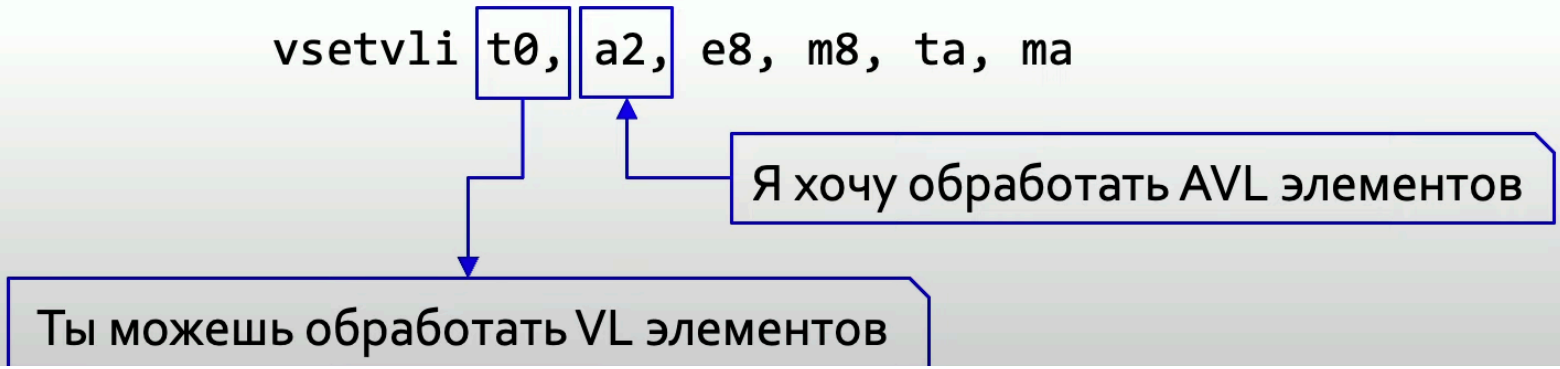
- x + y -> x + y - 1
- Аналогично для векторов:

```
vadd.vv v8, v4, v0
vsub.vv v8, v8, v12
```





Дополнительный слайд 2

Идея инструкций vset*, AVL и VL

```
vsetvli rd, rs1, vtype1   # AVL = x[rs1], x[rd] = vl
vsetivli rd, uimm, vtype1 # AVL = uimm, x[rd] = vl
vsetvl rd, rs1, rs2       # AVL = x[rs1], VT = x[rs2], x[rd] = vl

vset* rd, x0, ...         # AVL = VLMAX
```