Влетаем с двух ног в JS на Wasm

Дмитрий Бежецков

HOLY JS

igalia

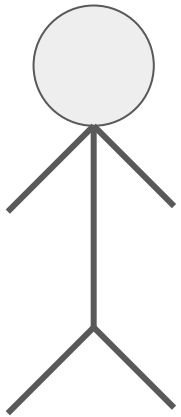
Дисклеймер

- Доклад не для тех, кто хочет послушать что такое Wasm
- Будем обсуждать как скомпилировать JS в Wasm и какое это решение показывает цифры

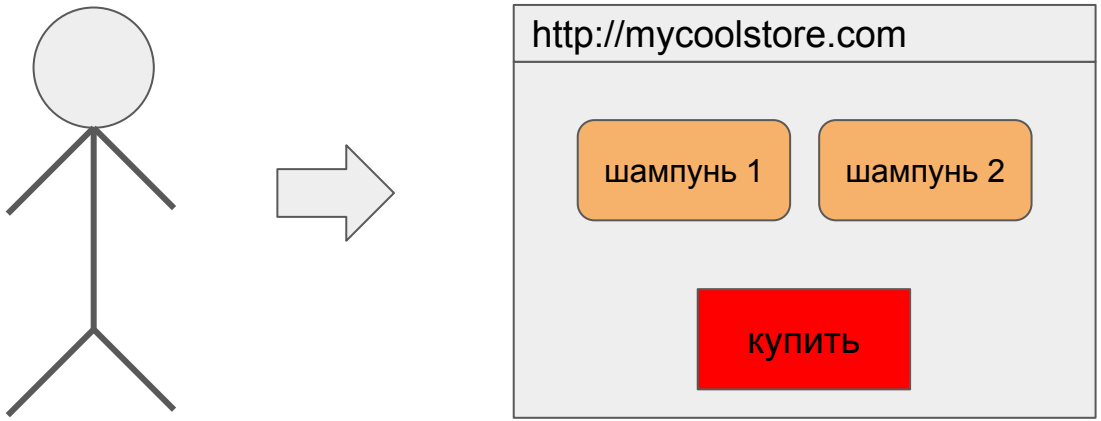
Wasm набирает обороты

1. В вебе: ML, обработка изображений, эмуляторы, игры и т.п.
2. В облаках: запуск пользовательского кода, дешевые контейнеры
3. Как язык для расширений: nginx, libsql

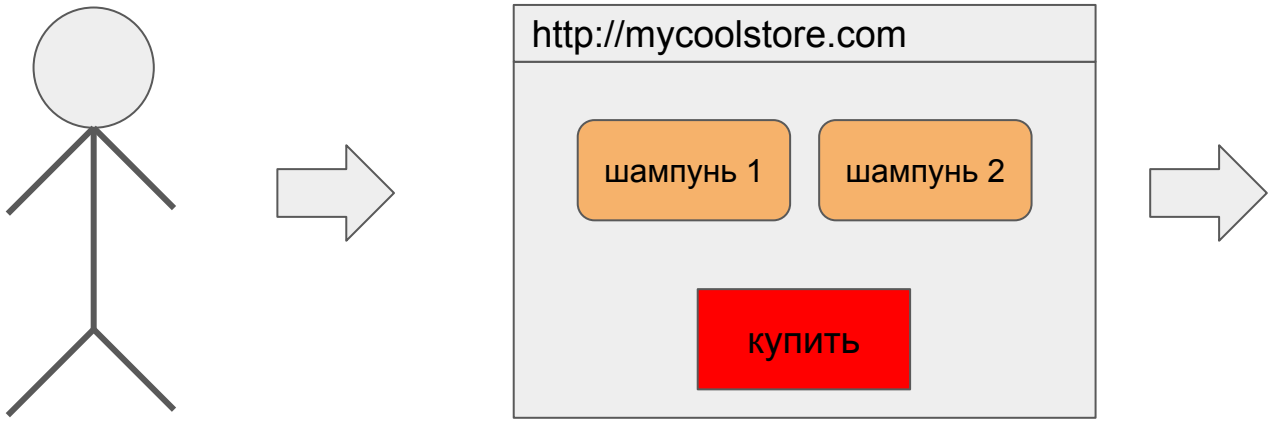
Пример



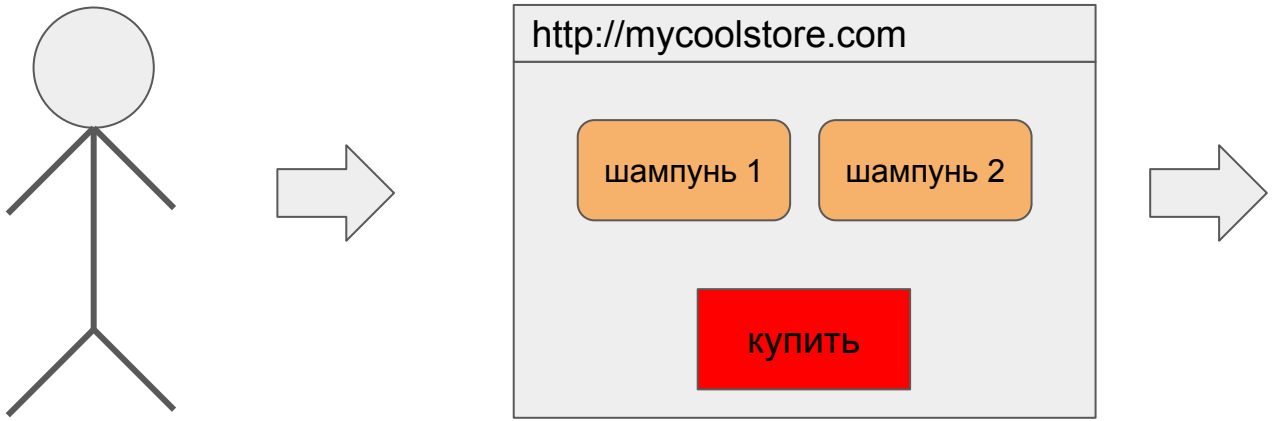
Пример



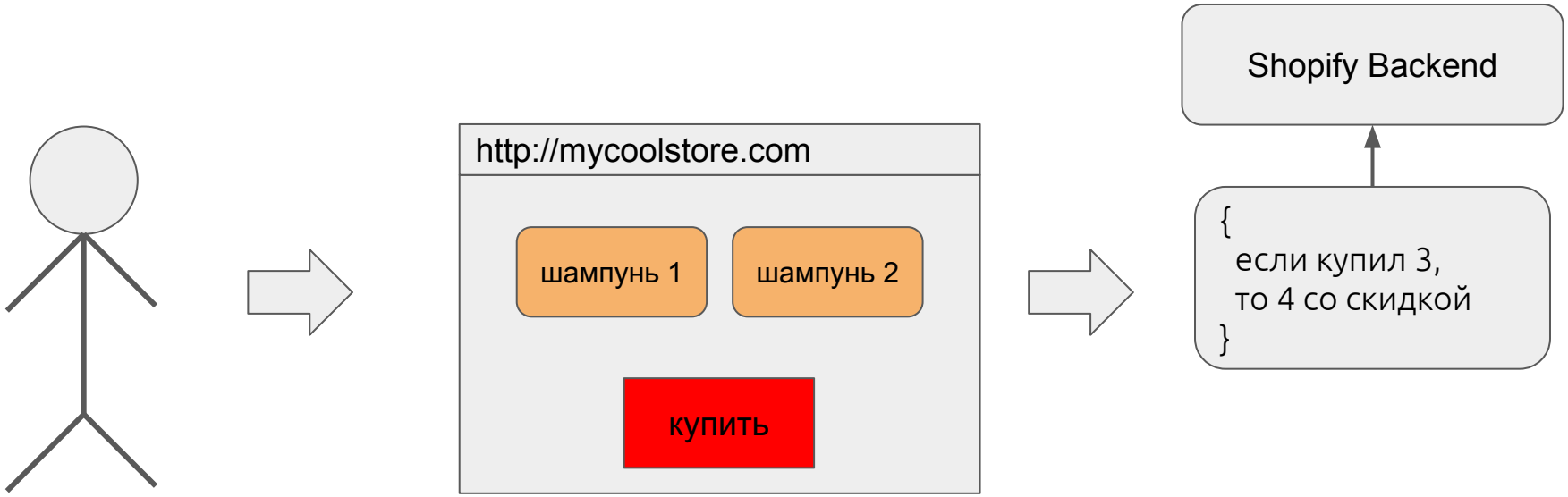
Пример



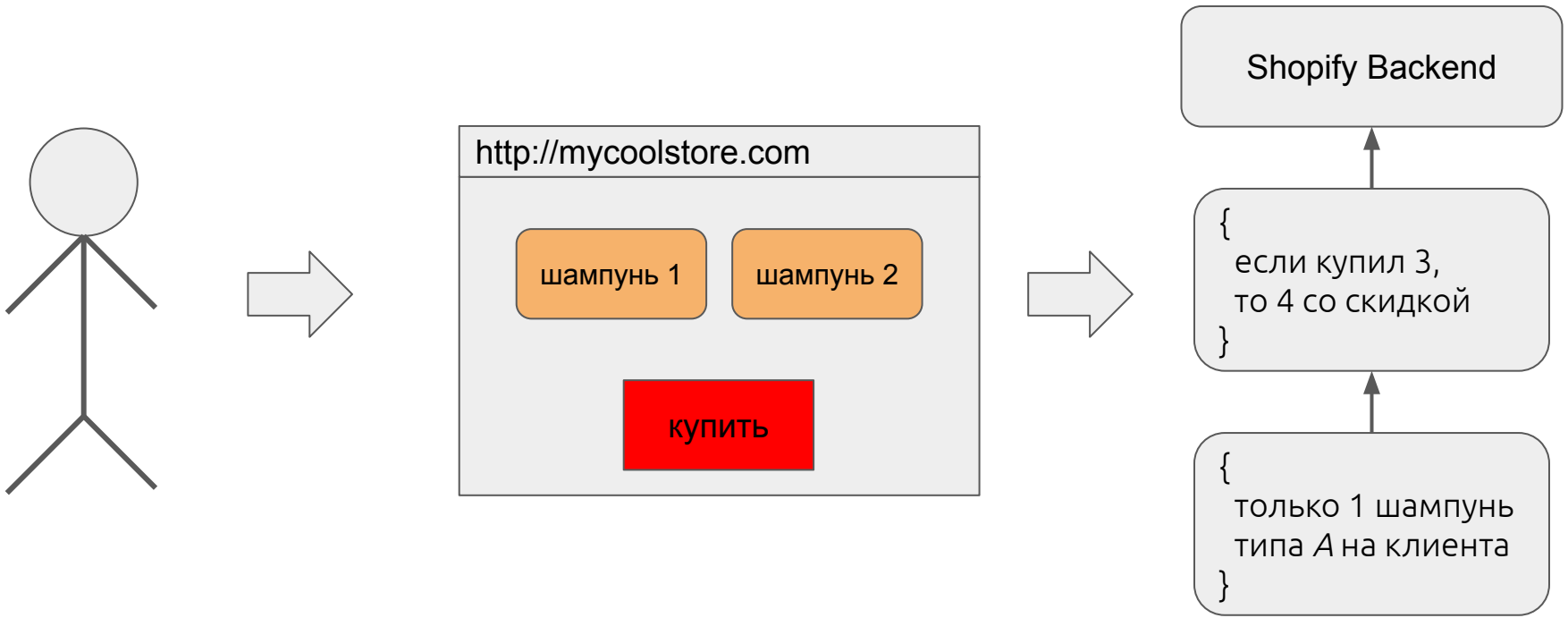
Пример


Shopify Backend
http://mycoolstore.com
шампунь 1
шампунь 2
купить

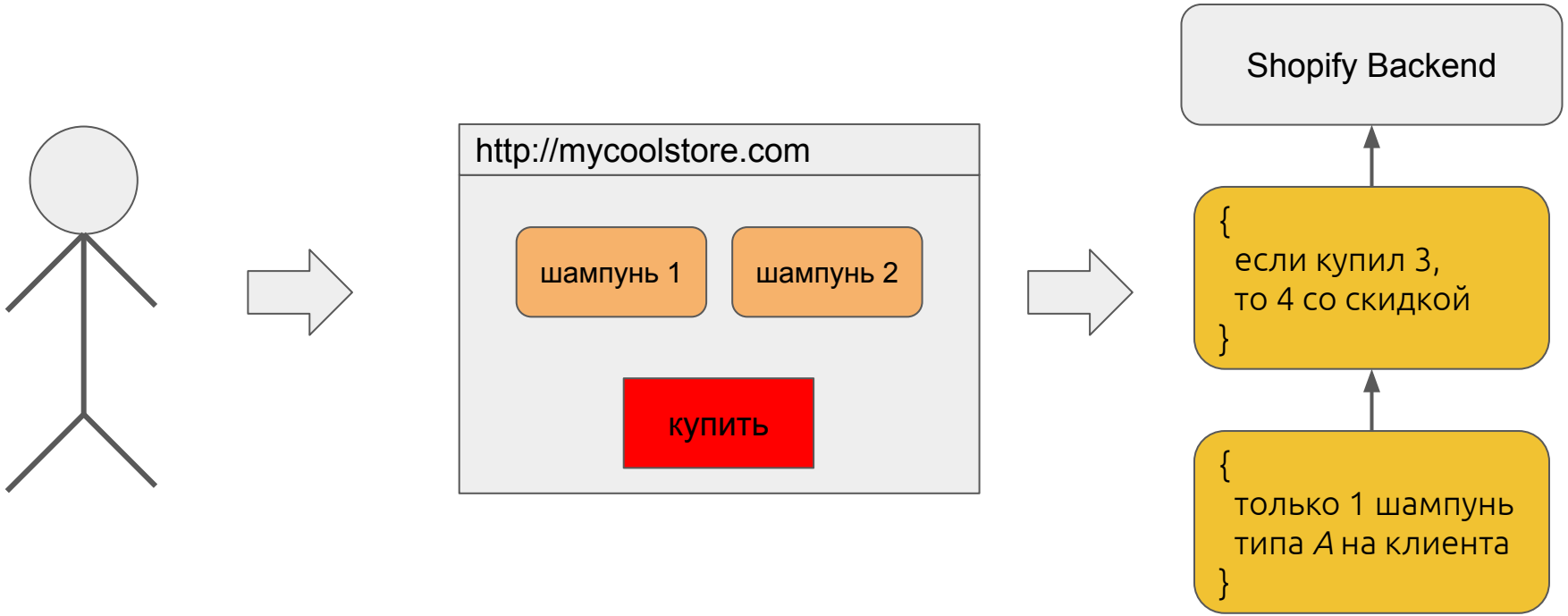
Пример


http://mycoolstore.com
шампунь 1
шампунь 2
купить
Shopify Backend
{
если купил 3,
то 4 со скидкой
}

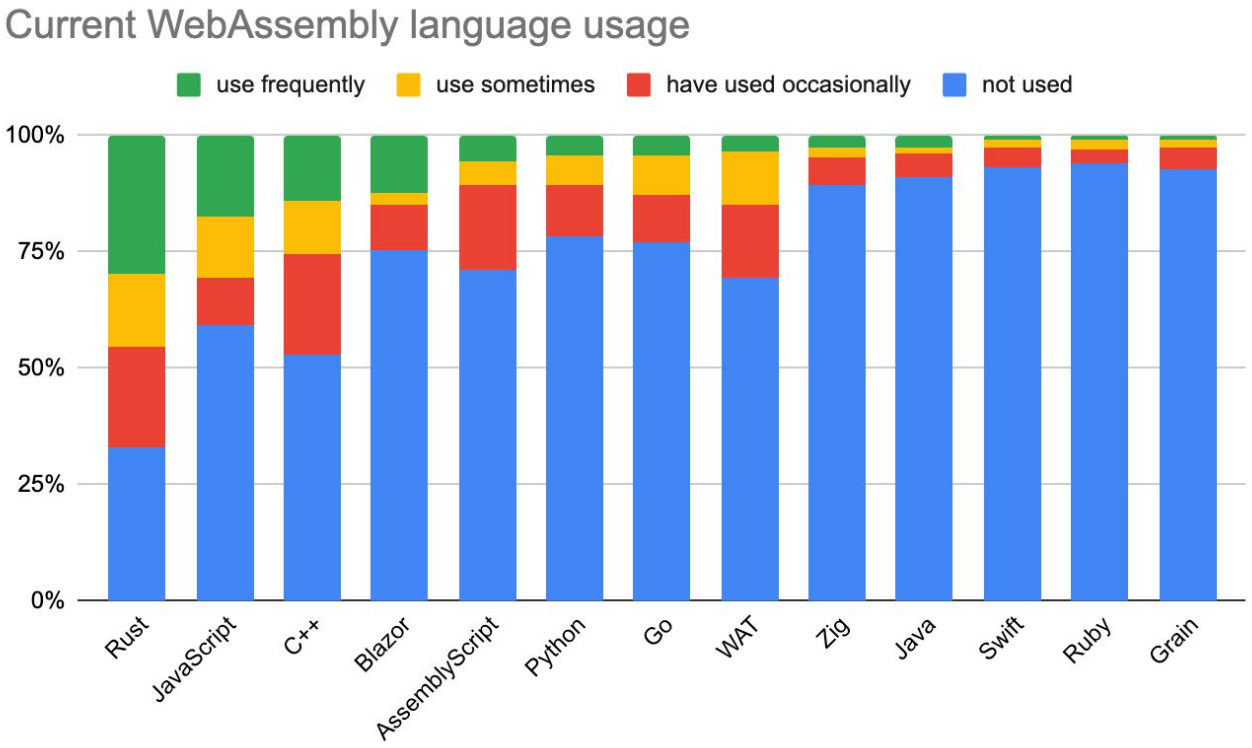
Пример


http://mycoolstore.com
шампунь 1
шампунь 2
купить
Shopify Backend
{
если купил 3,
то 4 со скидкой
}
{
только 1 шампунь
типа A на клиента
}

Пример


http://mycoolstore.com
шампунь 1
шампунь 2
купить
Shopify Backend
{
если купил 3,
то 4 со скидкой
}
{
только 1 шампунь
типа A на клиента
}

Самые популярные языки для wasm



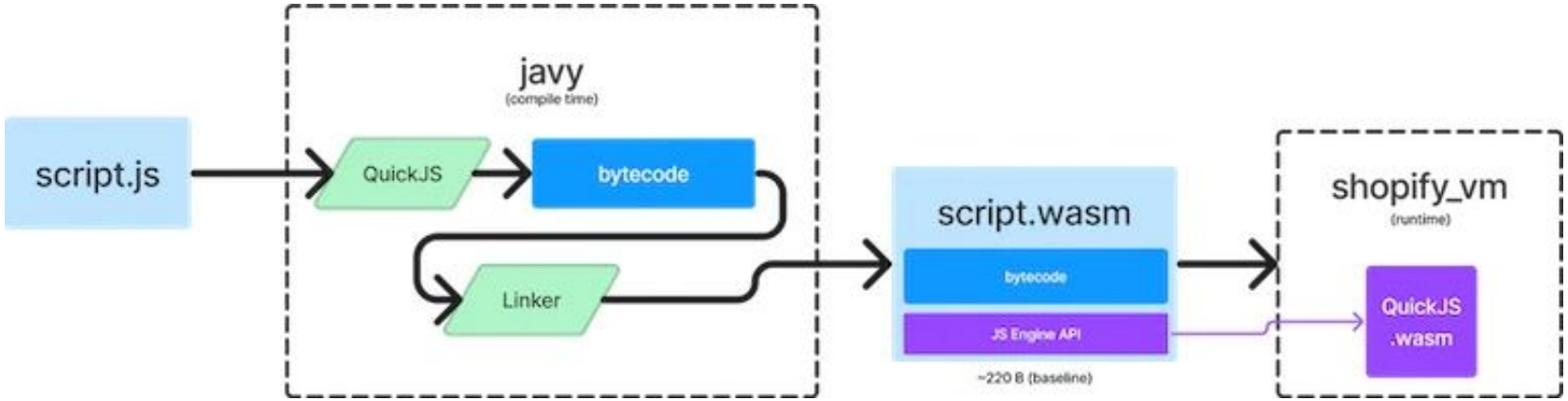
Текущее состояние дел с JS

SpiderMonkey

Обгоняем бугатти на самокате: quickjs

1. Маленький, написан одним человеком
2. Только интерпретатор
3. Очень хорошо оптимизирован
4. Компилируется в wasm размером в ~800Kb

Схема в Javy


javy
(compile time)
script.js
QuickJS
bytecode
Linker
script.wasm
bytecode
JS Engine API
~220 B (baseline)
shopify_vm
(runtime)
QuickJS
.wasm

Проблемы легковесного интерпретатора

- Это интерпретатор

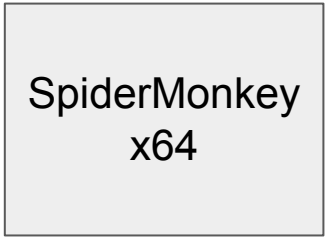
Wasm Jit

SpiderMonkey в качестве VM

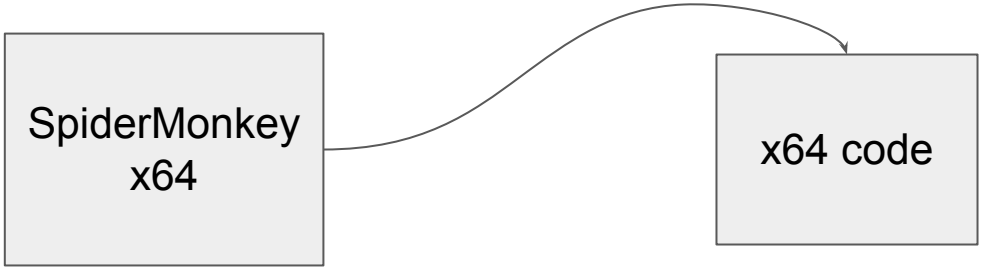
SpiderMonkey в качестве VM

1. Уже умеет компилироваться в wasm (https://habr.com/ru/articles/569492/)
2. Есть интерпретатор написанный на C++ в отличие от V8/JSC
3. К тому же она получается небольшой, около 5Mb

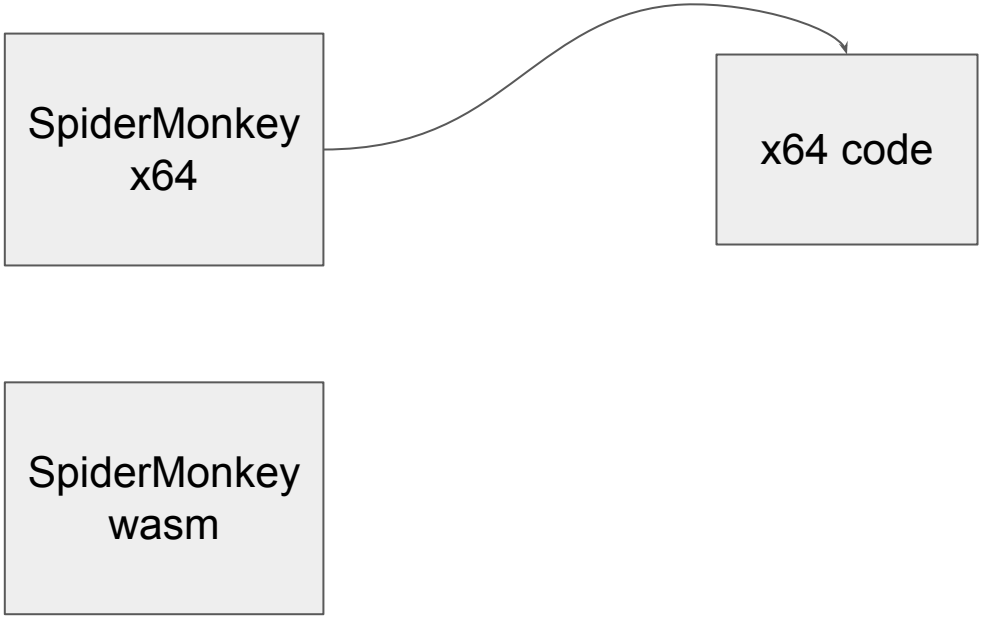
JIT для Wasm



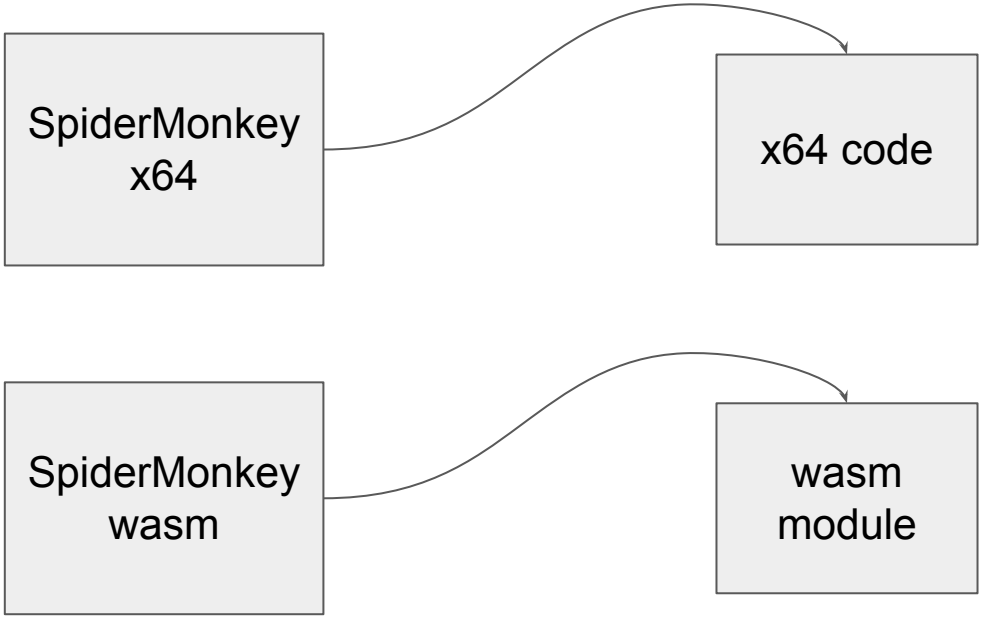
JIT для Wasm



JIT для Wasm

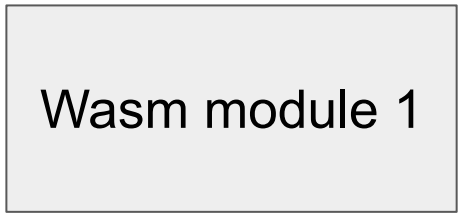


JIT для Wasm

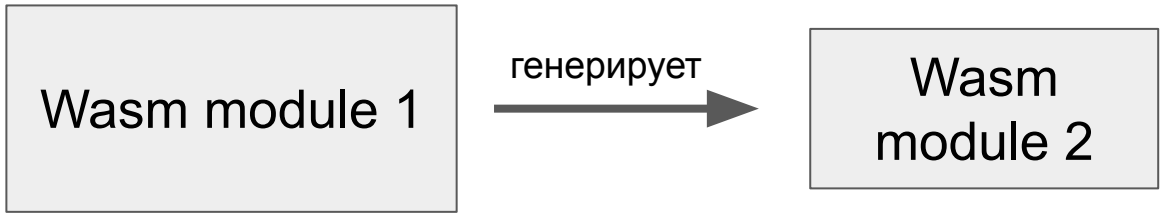


Wasm не поддерживает JIT

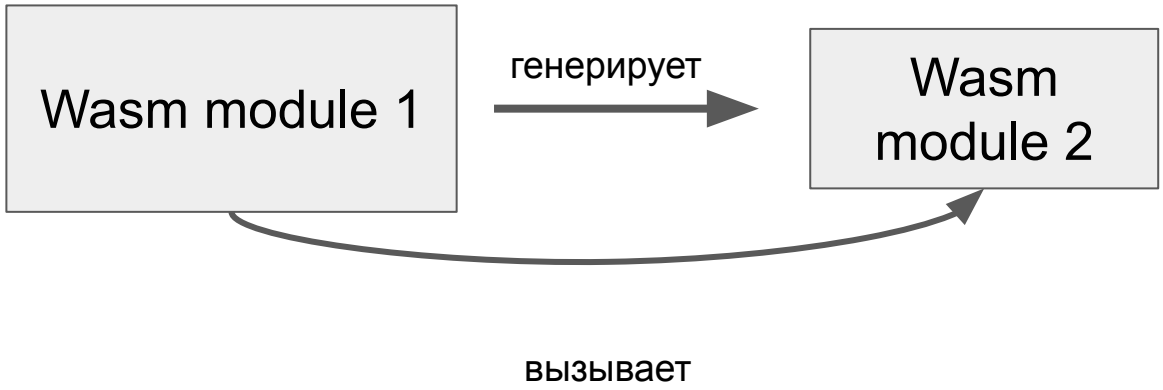
Wasm не поддерживает JIT



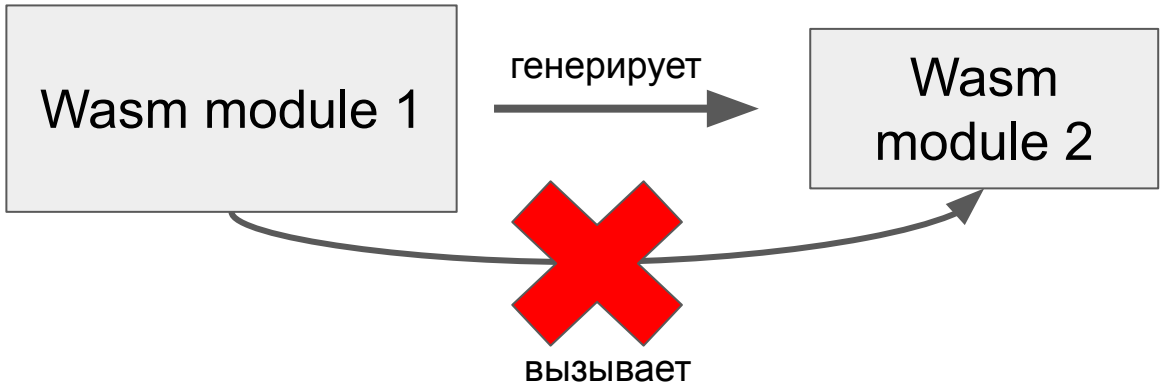
Wasm не поддерживает JIT



Wasm не поддерживает JIT



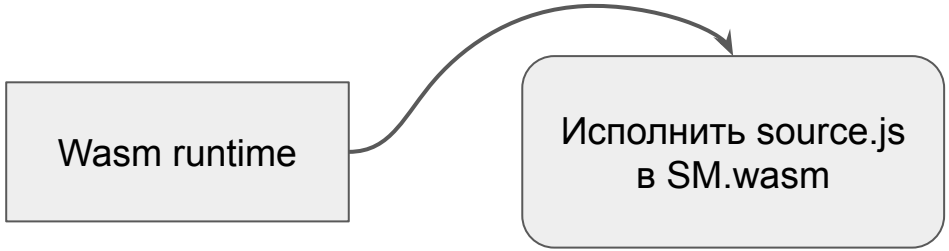
Wasm не поддерживает JIT



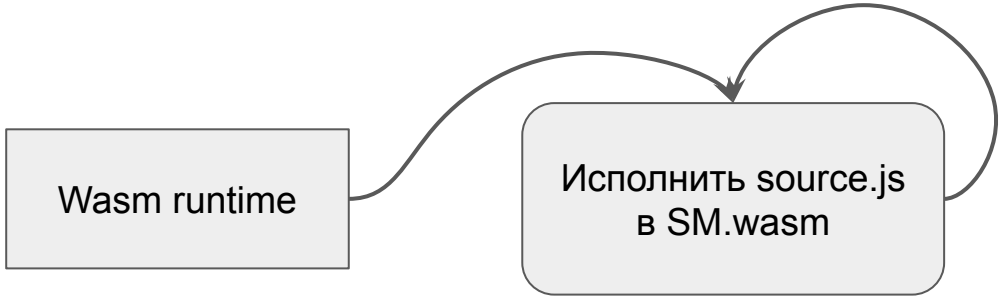
Асинхронный JIT

Асинхронный JIT

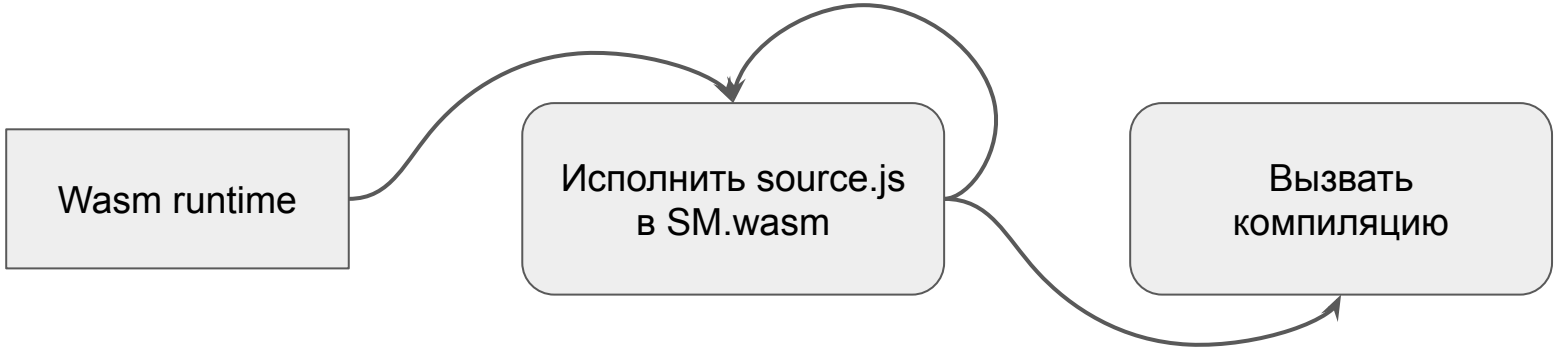
Асинхронный JIT



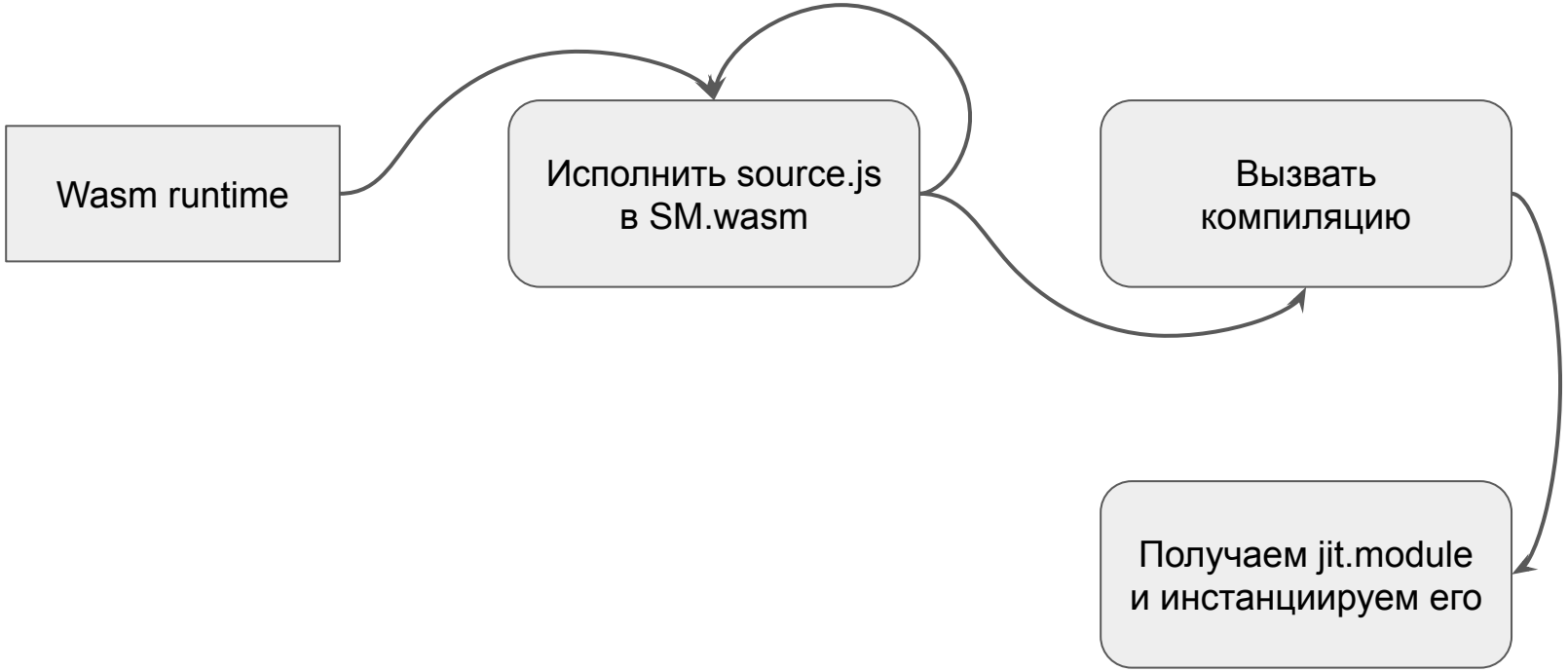
Асинхронный JIT



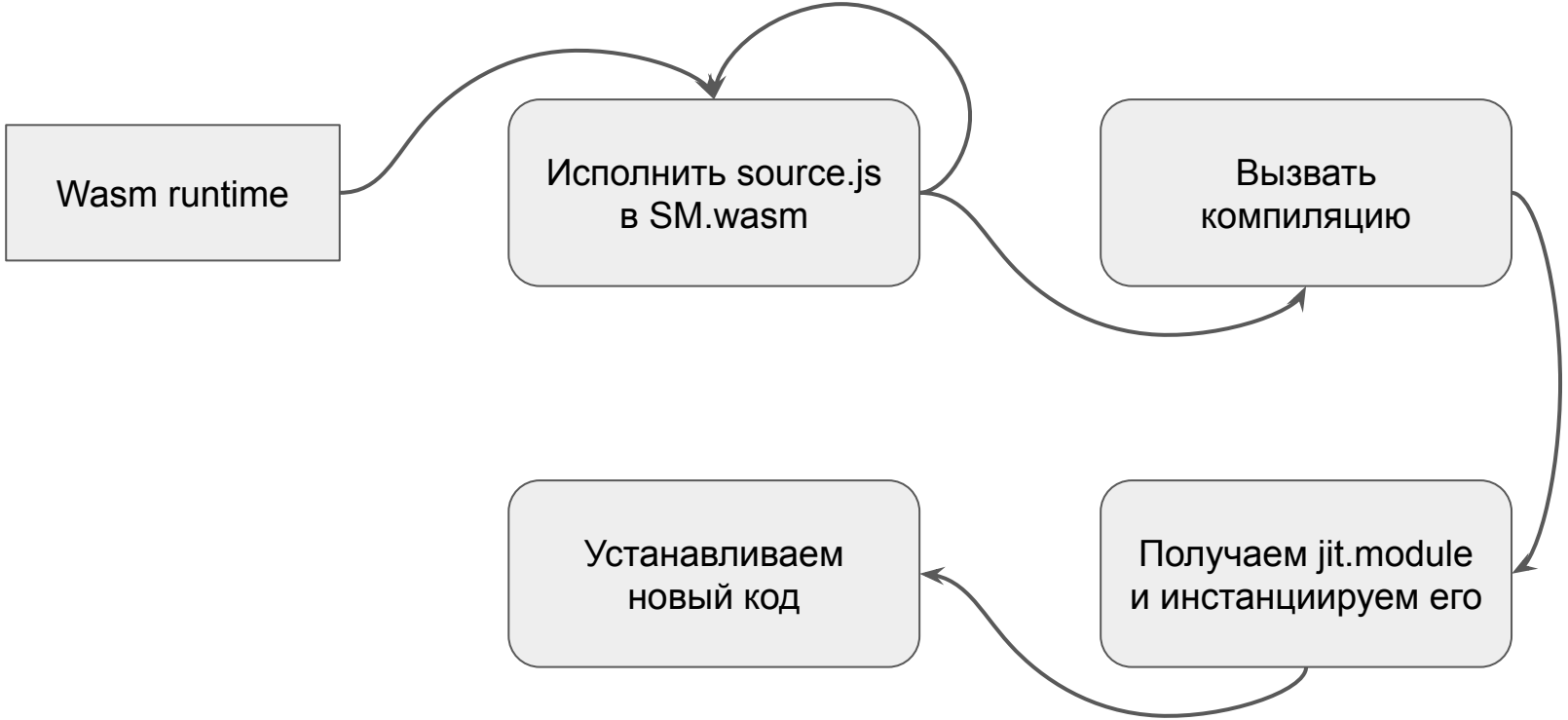
Асинхронный JIT



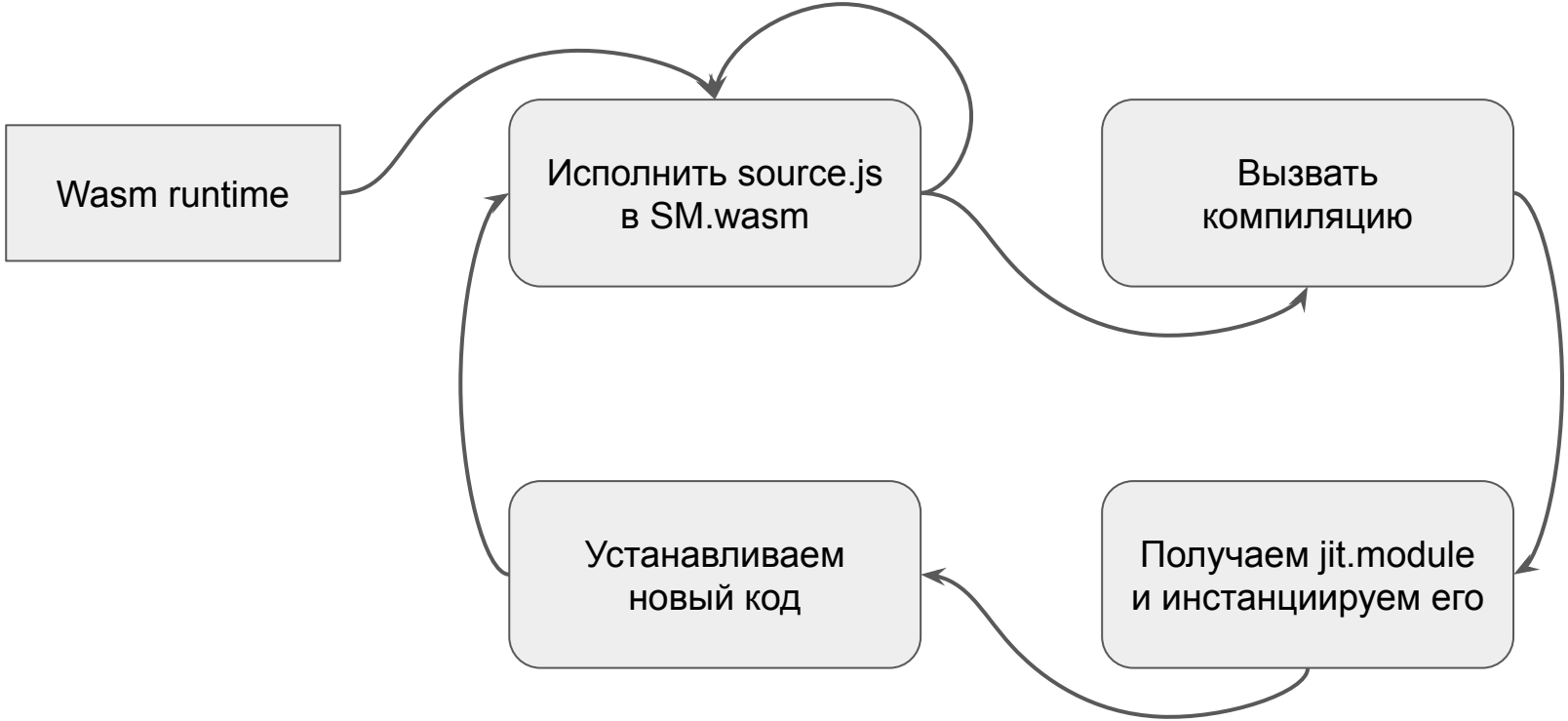
Асинхронный JIT



Асинхронный JIT

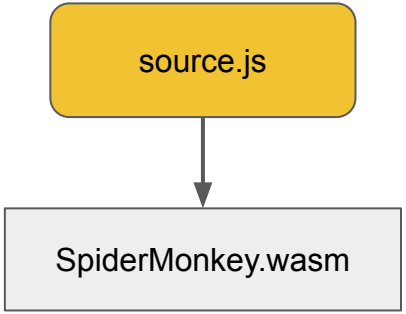


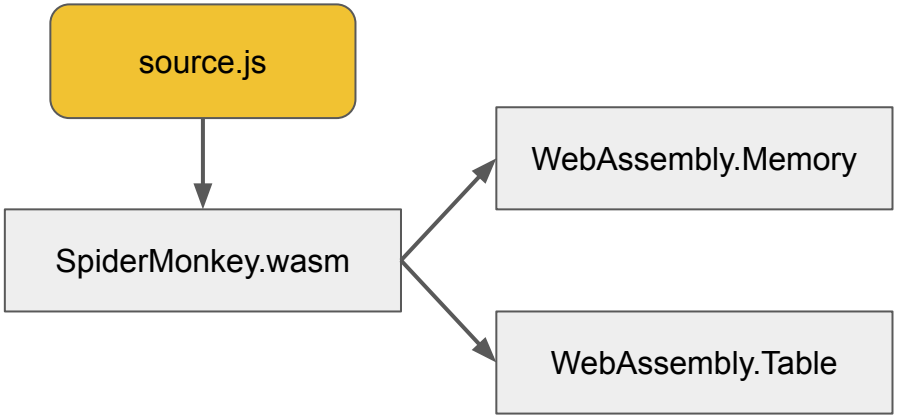
Асинхронный JIT

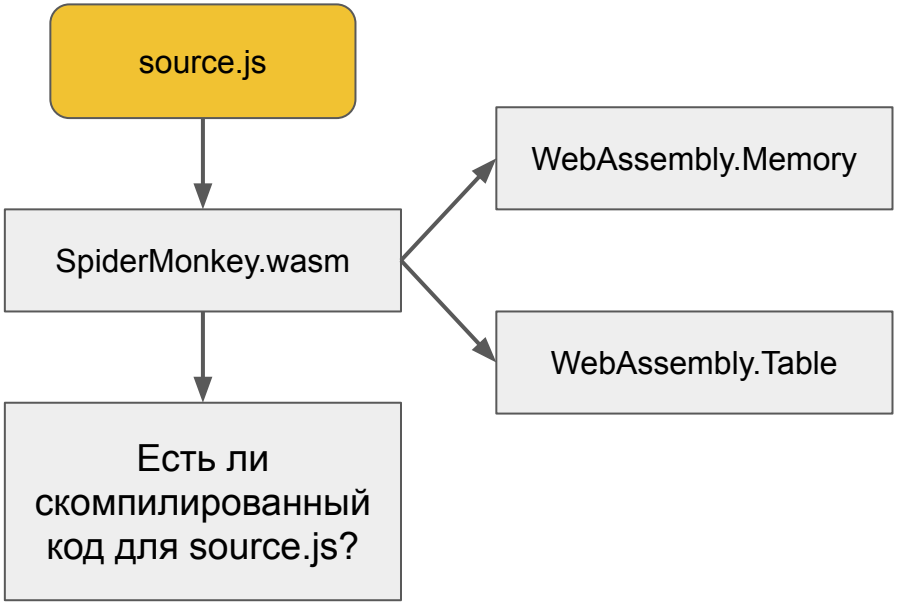


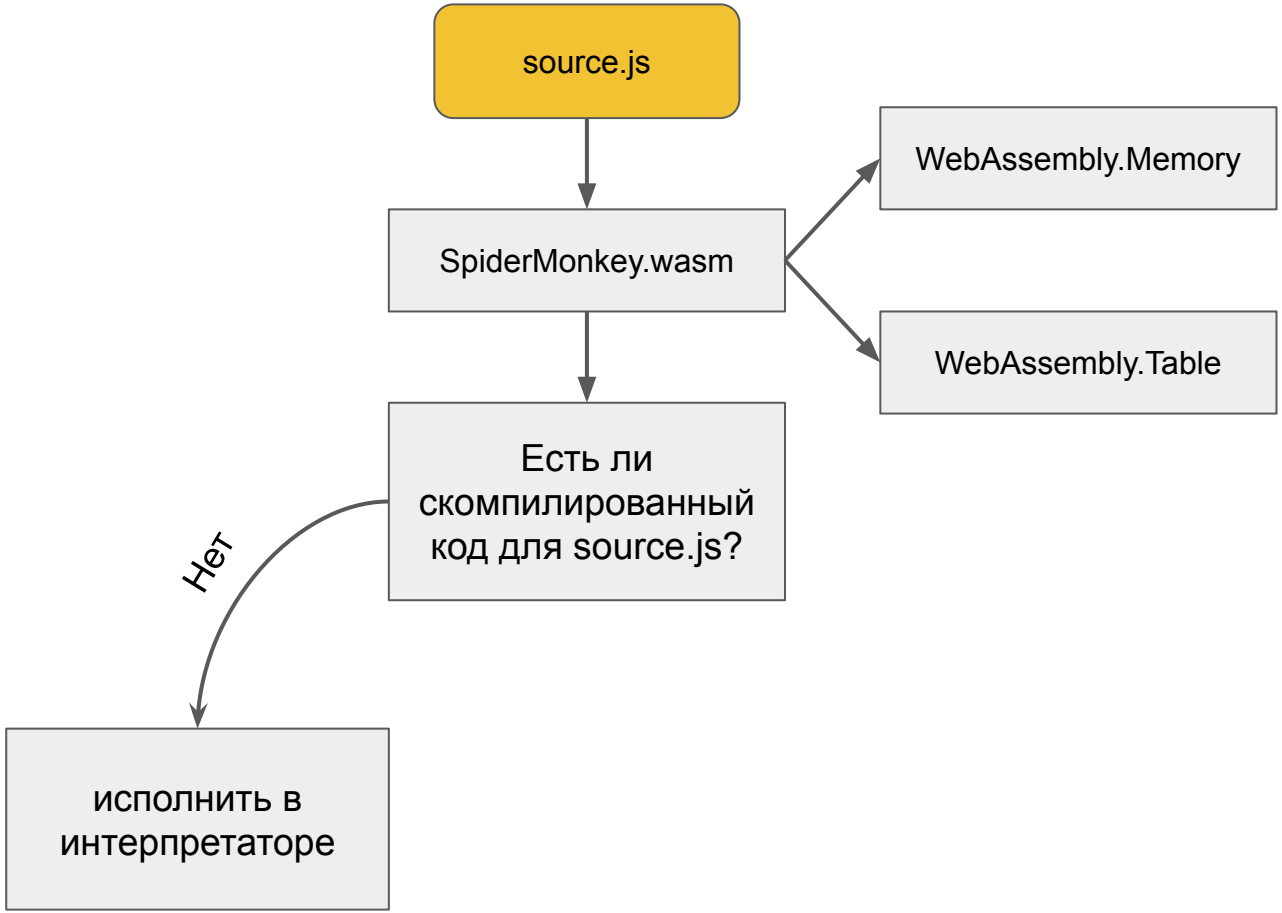

Теперь ты решаешь, когда
переключить передачу
BACK
OPTION

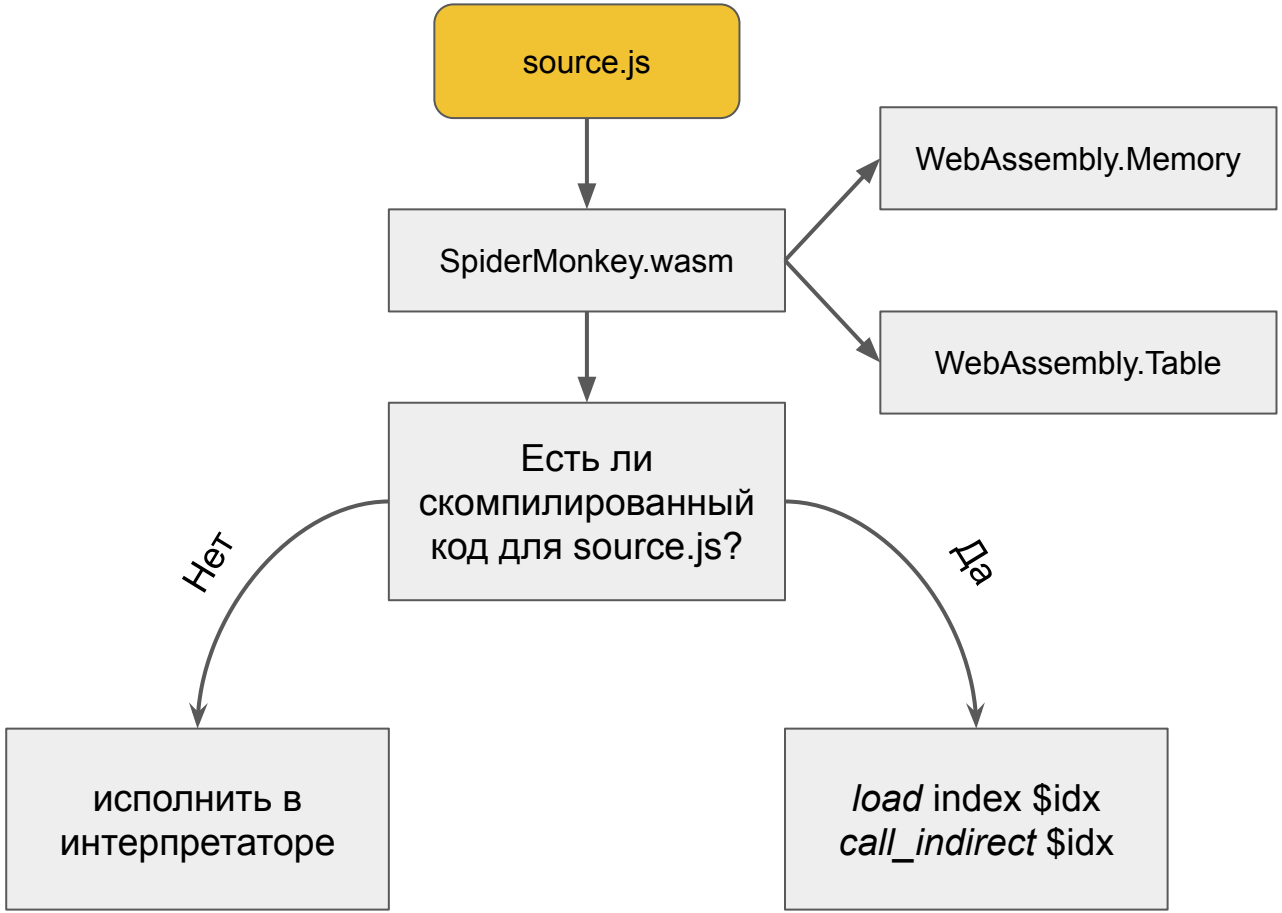
Установка нового кода

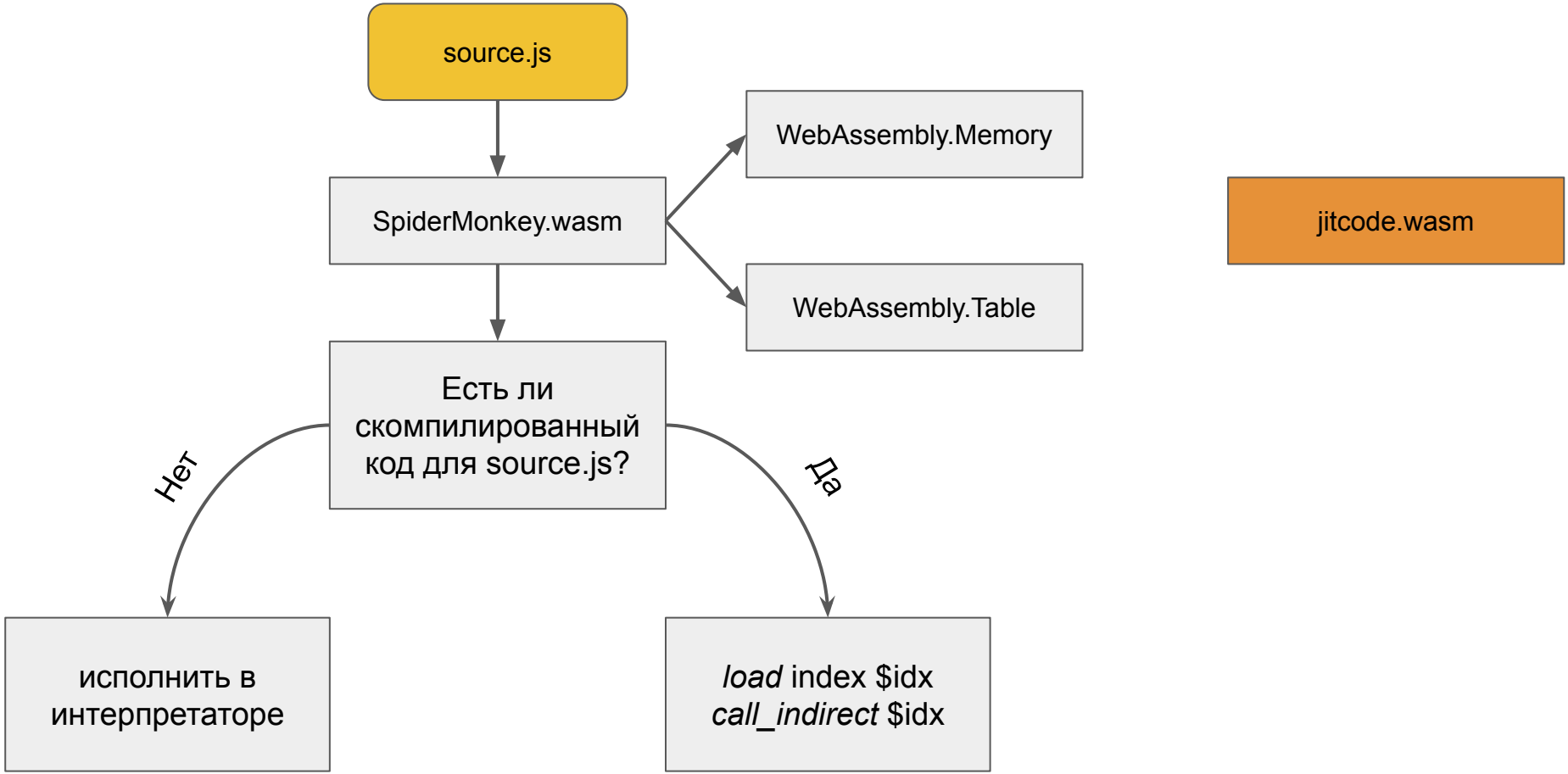

source.js
SpiderMonkey.wasm

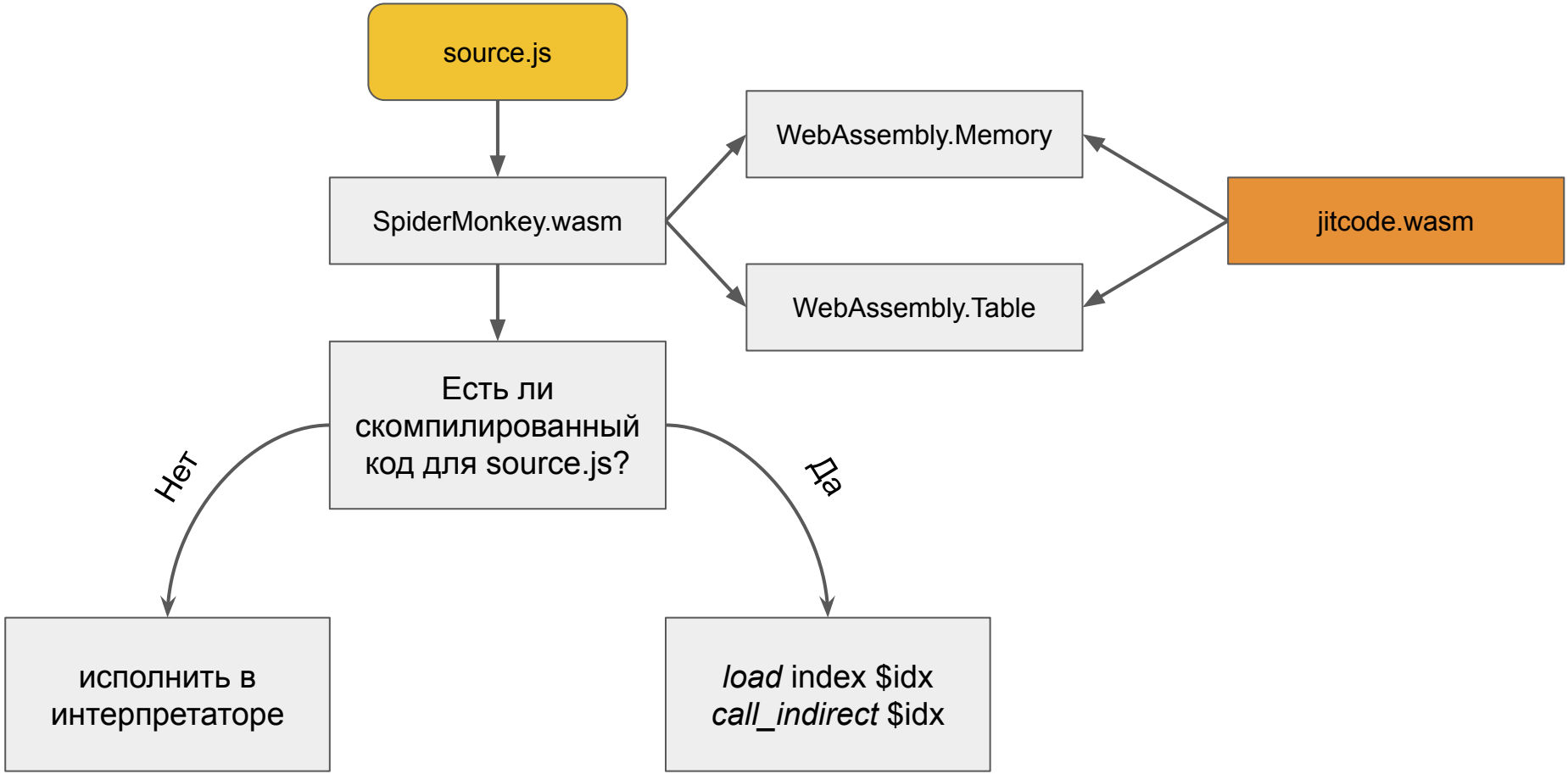

source.js
SpiderMonkey.wasm
WebAssembly.Memory
WebAssembly.Table

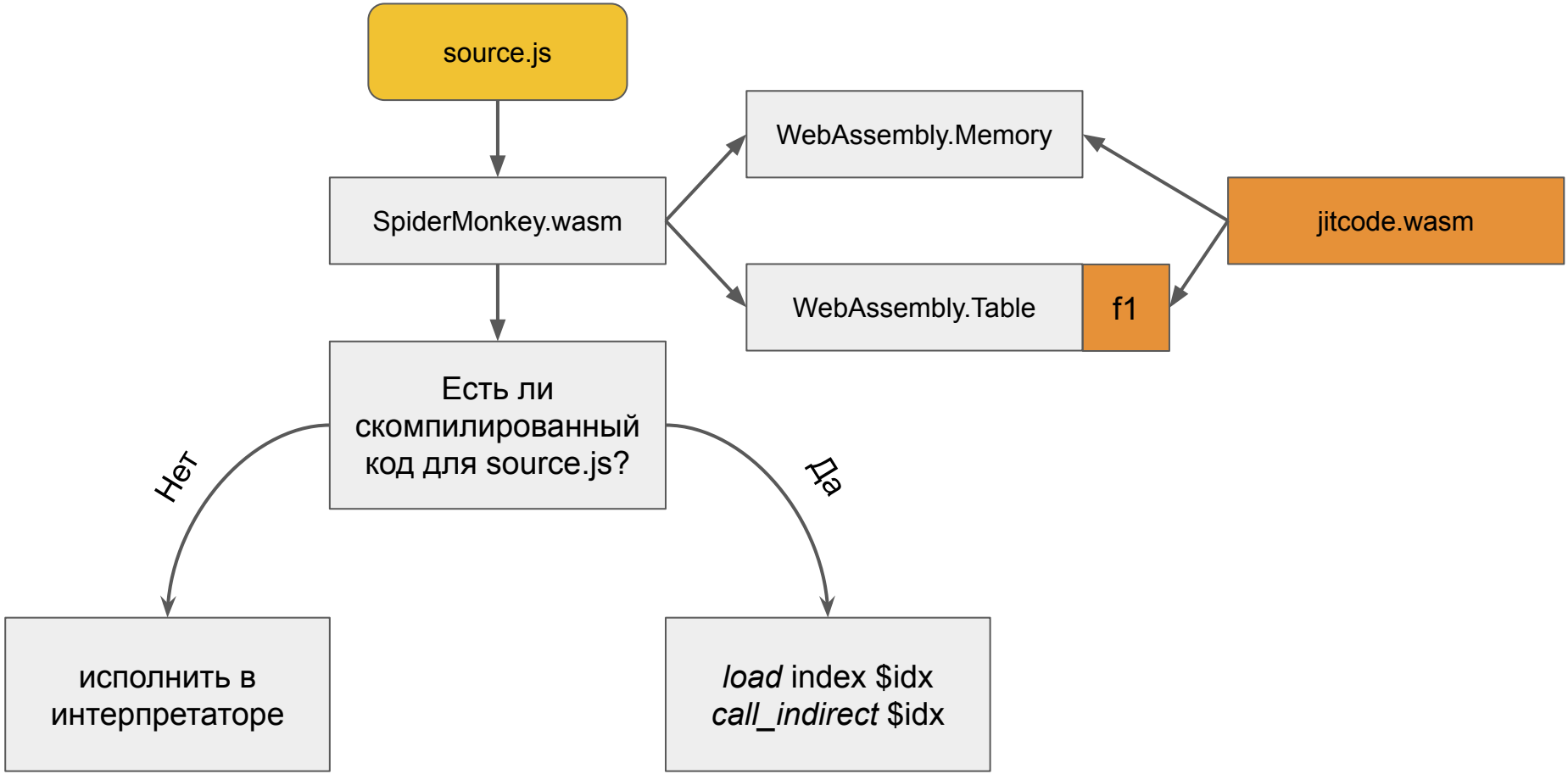

source.js
SpiderMonkey.wasm
WebAssembly.Memory
WebAssembly.Table
Есть ли скомпилированный код для source.js?

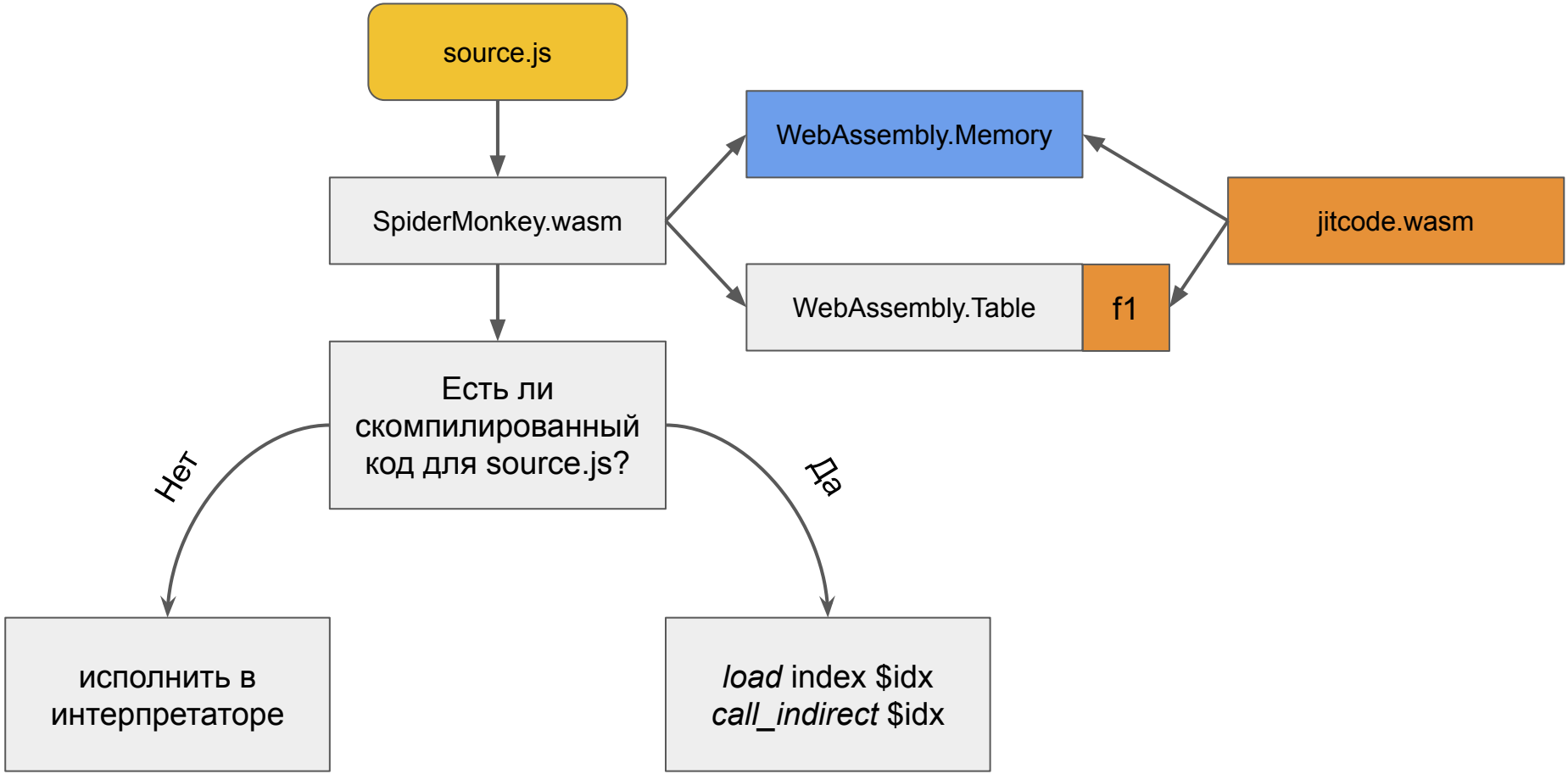

source.js
SpiderMonkey.wasm
WebAssembly.Memory
WebAssembly.Table
Есть ли скомпилированный код для source.js?
Нет
исполнить в интерпретаторе

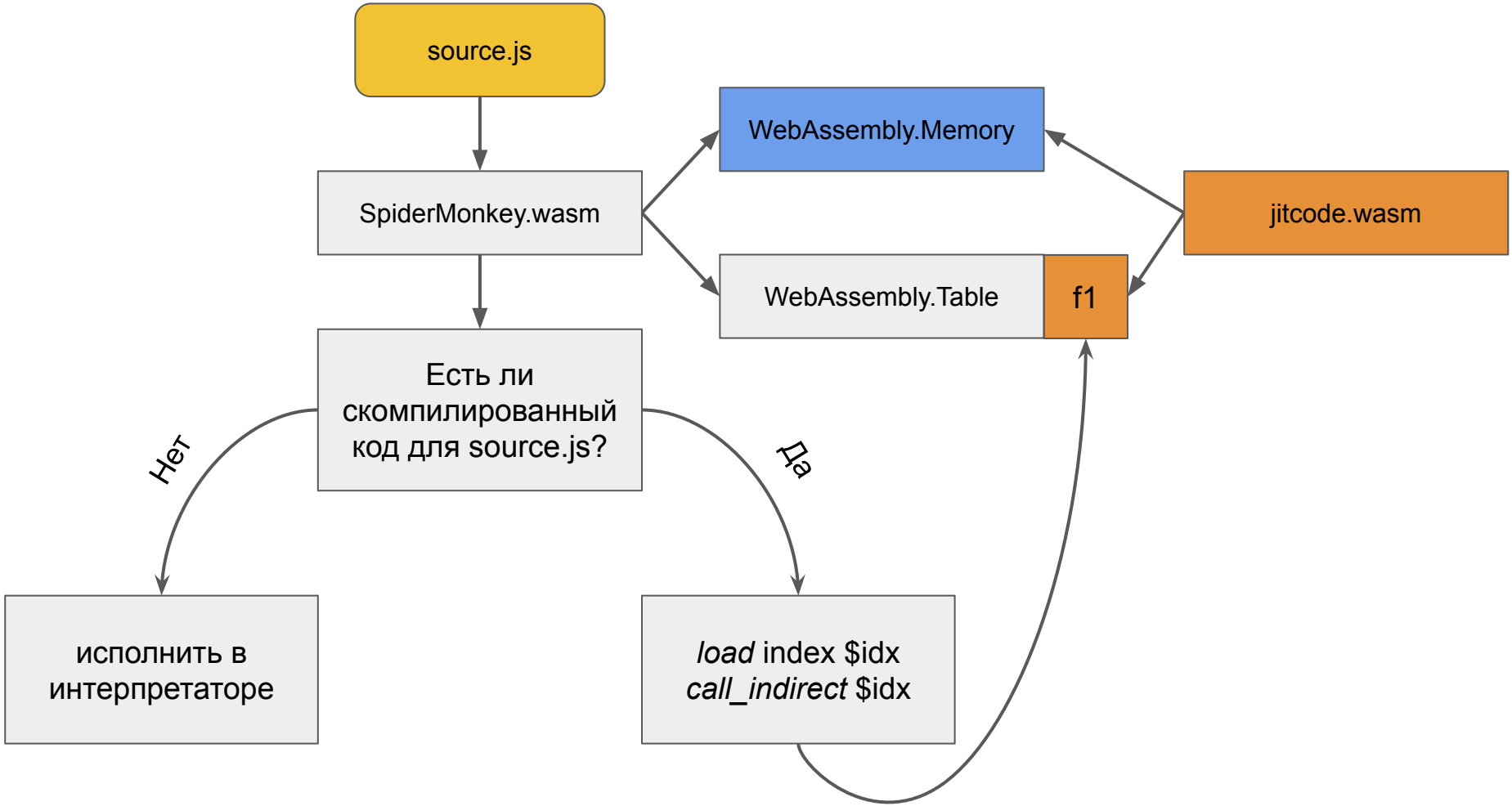

source.js
SpiderMonkey.wasm
WebAssembly.Memory
WebAssembly.Table
Есть ли скомпилированный код для source.js?
Нет
Да
исполнить в интерпретаторе
load index $idx
call_indirect $idx


source.js
SpiderMonkey.wasm
WebAssembly.Memory
WebAssembly.Table
jitcode.wasm
Есть ли скомпилированный код для source.js?
Нет
Да
исполнить в интерпретаторе
load index $idx
call_indirect $idx


source.js
SpiderMonkey.wasm
WebAssembly.Memory
WebAssembly.Table
jitcode.wasm
Есть ли скомпилированный код для source.js?
Нет
Да
исполнить в интерпретаторе
load index $idx
call_indirect $idx

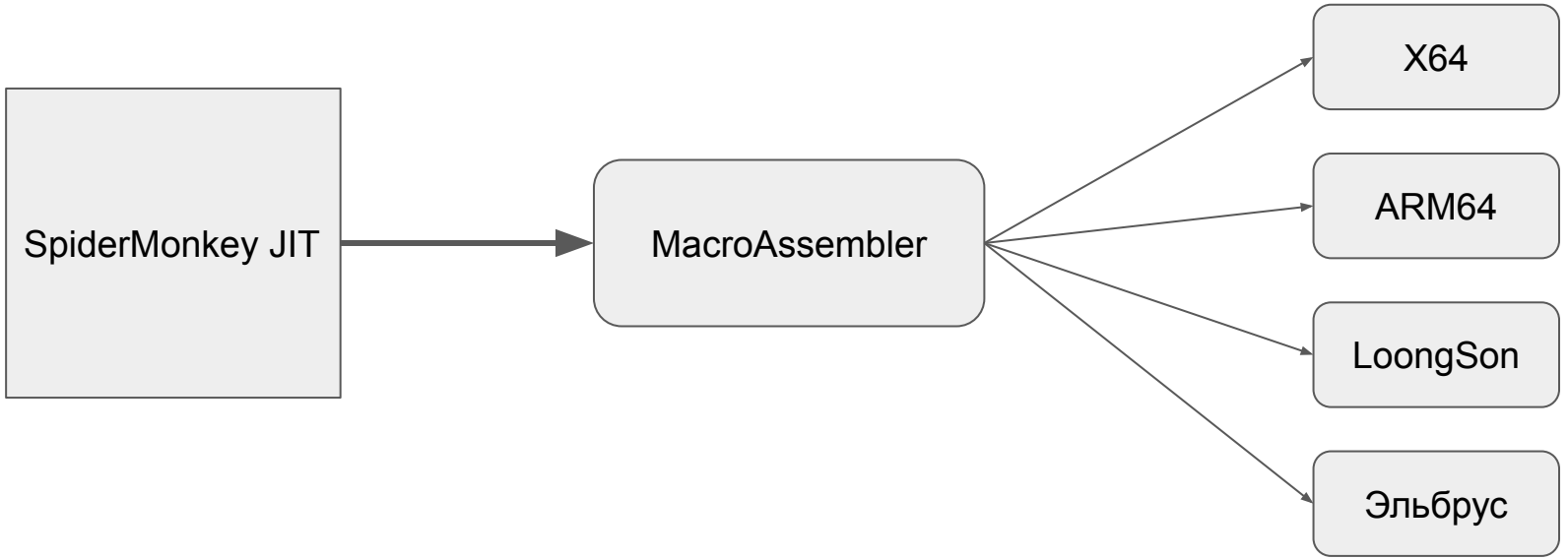

source.js
SpiderMonkey.wasm
WebAssembly.Memory
WebAssembly.Table
f1
jitcode.wasm
Есть ли скомпилированный код для source.js?
Нет
Да
исполнить в интерпретаторе
load index $idx
call_indirect $idx


source.js
SpiderMonkey.wasm
WebAssembly.Memory
WebAssembly.Table
f1
jitcode.wasm
Есть ли скомпилированный код для source.js?
Нет
Да
исполнить в интерпретаторе
load index $idx
call_indirect $idx

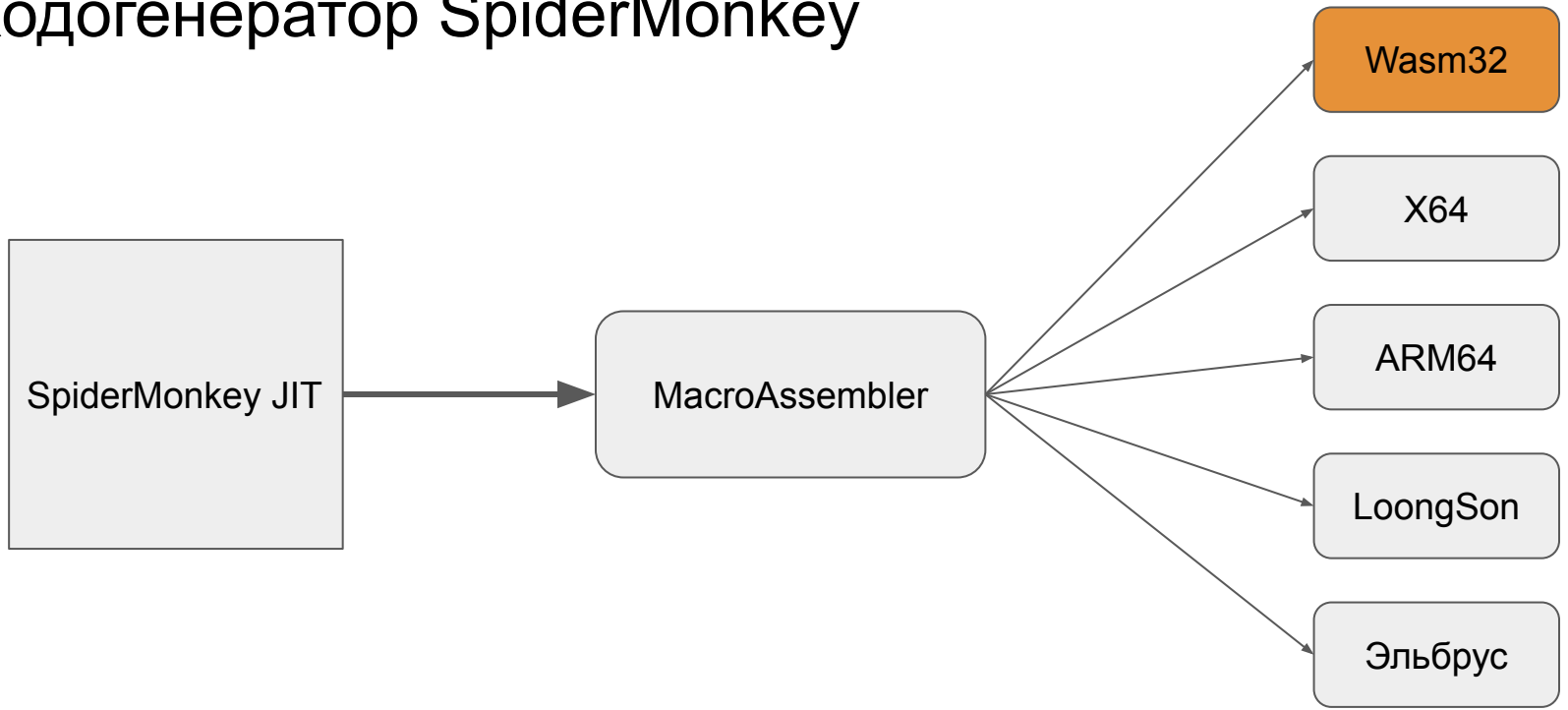

source.js
SpiderMonkey.wasm
WebAssembly.Memory
WebAssembly.Table
f1
jitcode.wasm
Есть ли скомпилированный код для source.js?
Нет
Да
исполнить в интерпретаторе
load index $idx
call_indirect $idx

Генерация кода

Кодогенератор SpiderMonkey



Кодогенератор SpiderMonkey



Ожидания SpiderMonkey от новой архитектуры

1. Процессор с регистрами
2. Наличие стека
3. Стандартный набор операций типа add, mul, div, mod, ld и т.п.
4. Лейблы, jump’ы и call’ы

Masm -> Wasm

Masm

Wasm

Masm

Wasm

Регистры

Masm

Регистры

Wasm

Локалы и глобалы

Masm

Регистры

Адресуемый стек

Wasm

Локалы и глобалы

Masm	Wasm
Регистры	Локалы и глобалы
Адресуемый стек	Не адресуемый стек

Masm	Wasm
Регистры	Локалы и глобалы
Адресуемый стек	Не адресуемый стек
Лейблы/Прыжки	

Masm	Wasm
Регистры	Локалы и глобалы
Адресуемый стек	Не адресуемый стек
Лейблы/Прыжки	Можно вызывать только функции и блоки

Эмуляция masm на wasm

Эмуляция masm на wasm

1. Каждую функцию в masm будет компилировать в отдельную функцию на wasm
2. Регистры представим как глобалы
3. Стек будет поддерживать в WebAssembly.memory
4. Уложим masm в cfg и напишем интерпретатор на базовых блоках

Схема компиляции

```
{
  Label done;
  Register scratch = R0.scratchReg();
  masm.loadScript(scratch);
  masm.branchTest32(Zero, scratch, scratch, &done);

  masm.unreachable();

  masm.bind(&done);
  masm.Push(Imm32(0));
}
```

Схема компиляции

```
{
  Label done;
  Register scratch = R0.scratchReg();
  masm.loadScript(scratch);
  masm.branchTest32(Zero, scratch, scratch, &done);

  masm.unreachable();

  masm.bind(&done);
  masm.Push(Imm32(0));
}
```

загрузить скрипт в scratch

Схема компиляции

```
{
  Label done;
  Register scratch = R0.scratchReg();
  masm.loadScript(scratch);
  masm.branchTest32(Zero, scratch, scratch, &done);

  masm.unreachable();

  masm.bind(&done);
  masm.Push(Imm32(0));
}
```

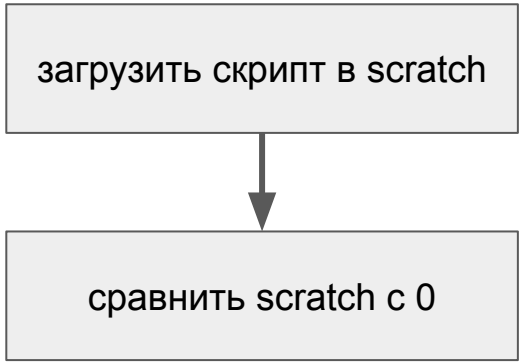


Схема компиляции

```
{
  Label done;
  Register scratch = R0.scratchReg();
  masm.loadScript(scratch);
  masm.branchTest32(Zero, scratch, scratch, &done);

  masm.unreachable();

  masm.bind(&done);
  masm.Push(Imm32(0));
}
```

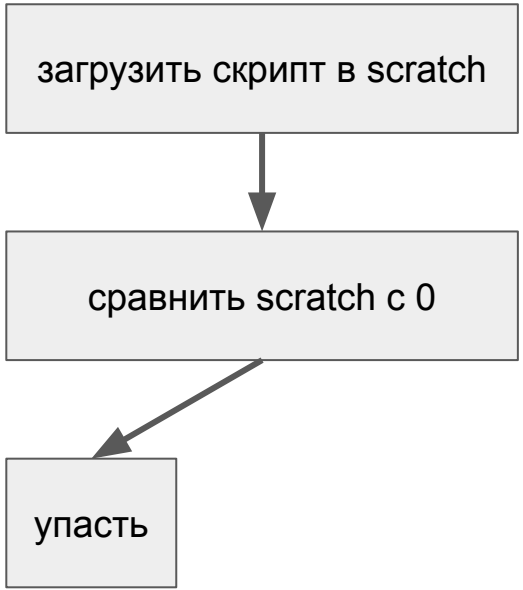


Схема компиляции

```
{
  Label done;
  Register scratch = R0.scratchReg();
  masm.loadScript(scratch);
  masm.branchTest32(Zero, scratch, scratch, &done);

  masm.unreachable();

  masm.bind(&done);
  masm.Push(Imm32(0));
}
```

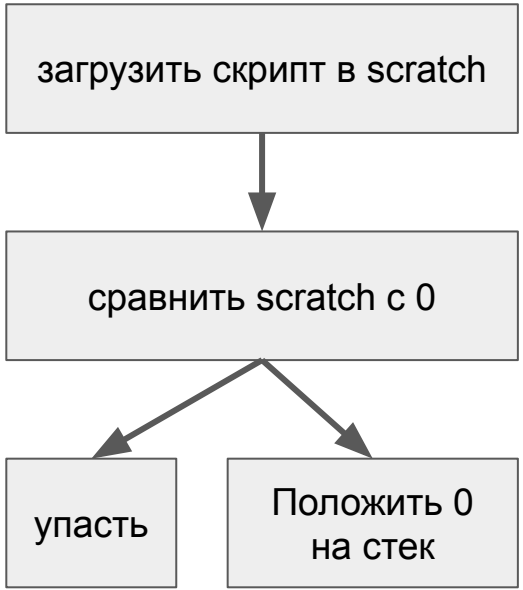
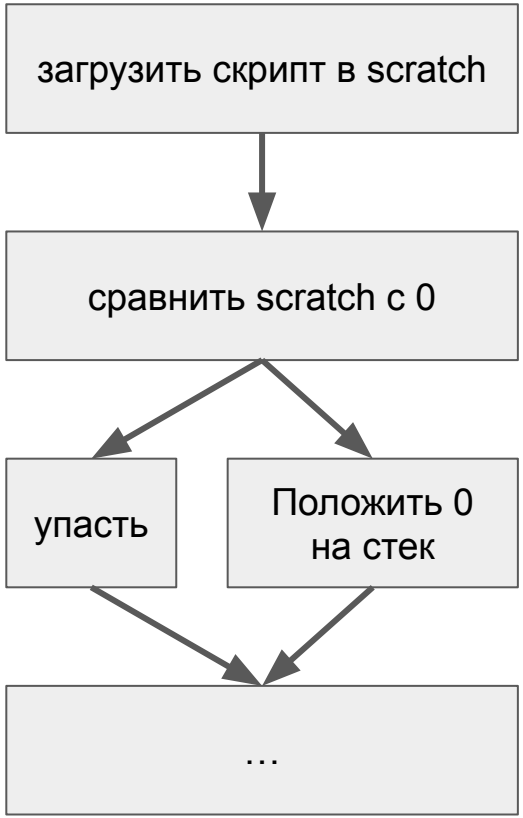
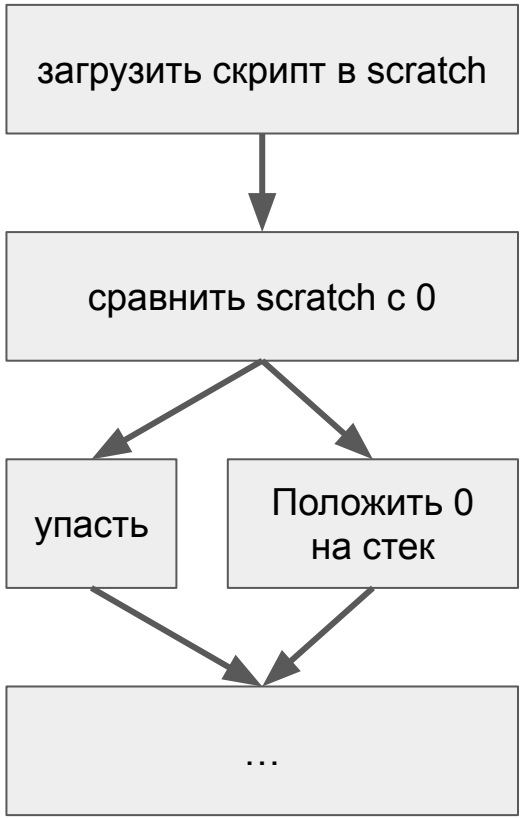
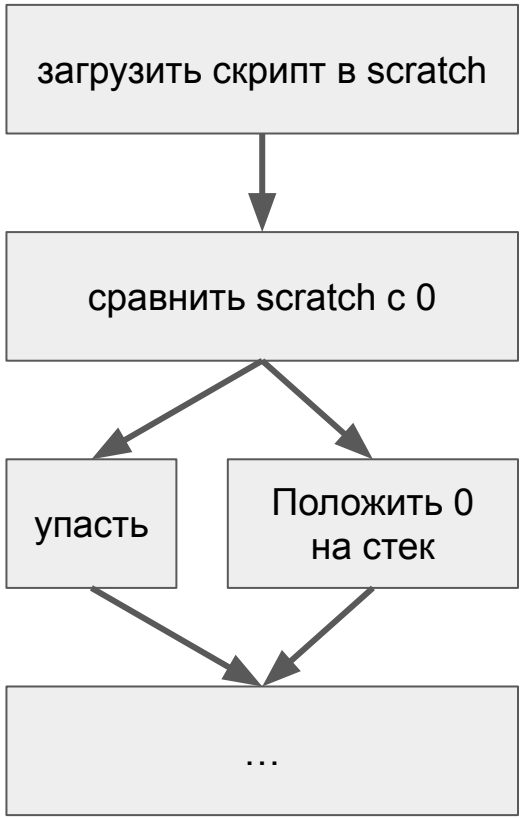


Схема компиляции

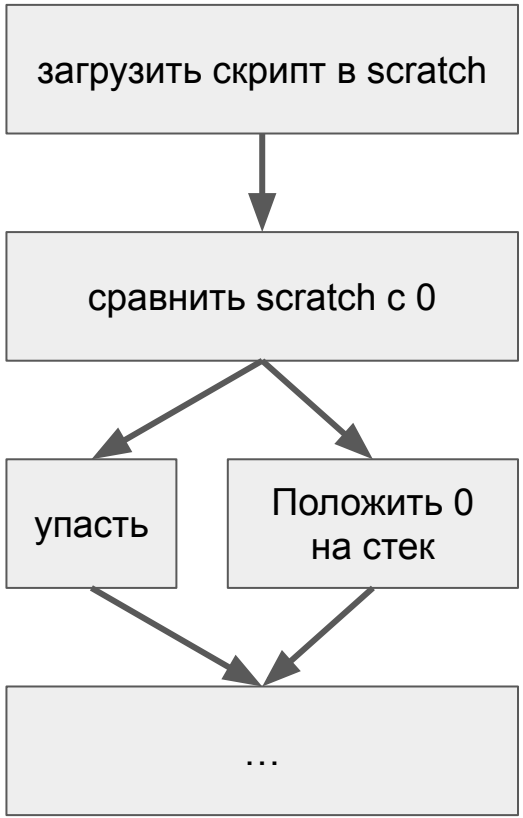
```
{
  Label done;
  Register scratch = R0.scratchReg();
  masm.loadScript(scratch);
  masm.branchTest32(Zero, scratch, scratch, &done);

  masm.unreachable();

  masm.bind(&done);
  masm.Push(Imm32(0));
}
```



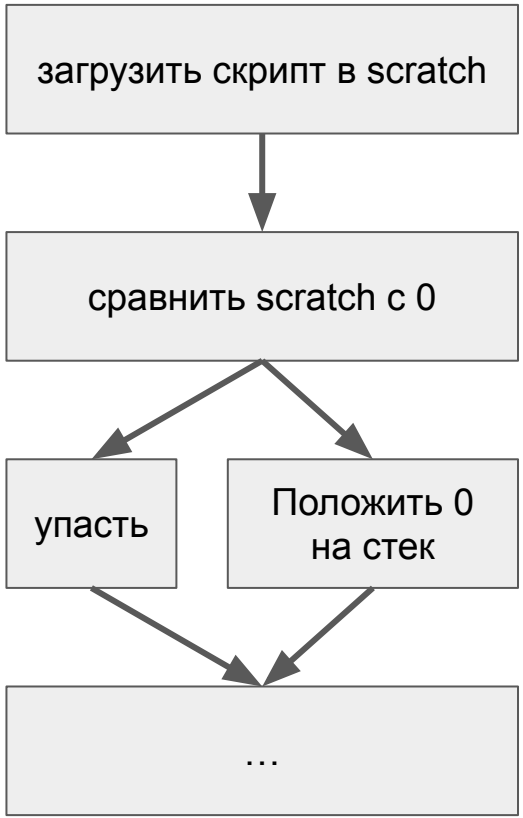

загрузить скрипт в scratch
сравнить scratch с 0
упасть
Положить 0
на стек
…



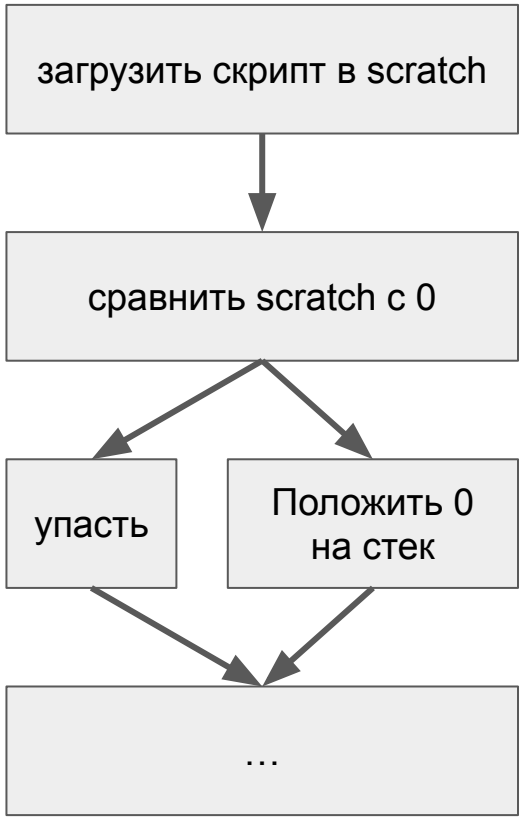

```
(loop $interpreter_loop
    (block
        (block
            (block
                (block
                    (block
                        (block
                            (i32.load (local.get $pc))
                            br_table 0 1 2 3 4
                        )

                        ;; Load script -> scratch
                        ++$pc
                        br $interpreter_loop
                    )

                    ;; Compare scratch against 0
                    if true => local.set $pc 2
                    else => local.set $pc 3
                    br $interpreter_loop
                )

                unreachable
                br $interpreter_loop
            )

            ;; Push(0)
            ++$pc
            br $interpreter_loop
        )
        br $interpreter_loop
)
```



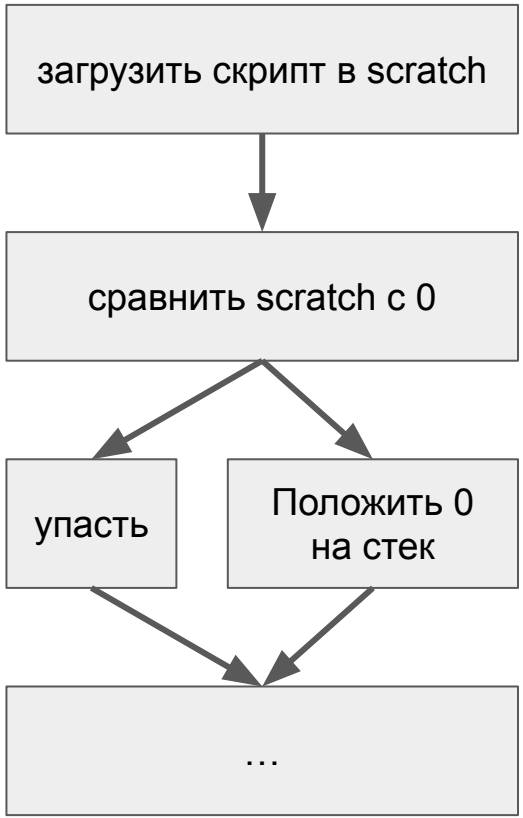

```
(loop $interpreter_loop
    (block
        (block
            (block
                (block
                    (block
                        (block
                            (i32.load (local.get $pc))
                            br_table 0 1 2 3 4
                        )

                        ;; Load script -> scratch
                        ++$pc
                        br $interpreter_loop
                    )

                    ;; Compare scratch against 0
                    if true => local.set $pc 2
                    else => local.set $pc 3
                    br $interpreter_loop
                )

                unreachable
                br $interpreter_loop
            )

            ;; Push(0)
            ++$pc
            br $interpreter_loop
        )
        br $interpreter_loop
)
```



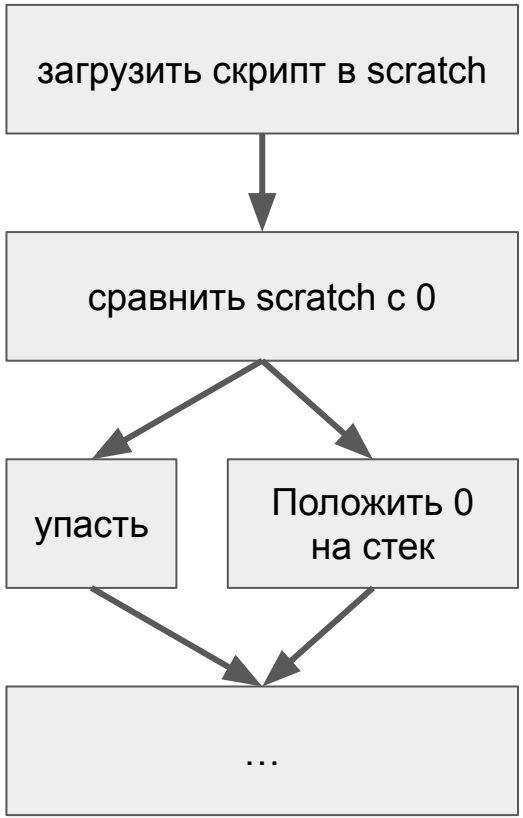

```
(loop $interpreter_loop
    (block
        (block
            (block
                (block
                    (block
                        (block
                            (i32.load (local.get $pc))
                            br_table 0 1 2 3 4
                        )

                        ;; Load script -> scratch
                        ++$pc
                        br $interpreter_loop
                    )

                    ;; Compare scratch against 0
                    if true => local.set $pc 2
                    else => local.set $pc 3
                    br $interpreter_loop
                )

                unreachable
                br $interpreter_loop
            )

            ;; Push(0)
            ++$pc
            br $interpreter_loop
        )
        br $interpreter_loop
)
```




```
(loop $interpreter_loop
    (block
        (block
            (block
                (block
                    (block
                        (block
                            (i32.load (local.get $pc))
                            br_table 0 1 2 3 4
                        )

                        ;; Load script -> scratch
                        ++$pc
                        br $interpreter_loop
                    )

                    ;; Compare scratch against 0
                    if true => local.set $pc 2
                    else => local.set $pc 3
                    br $interpreter_loop
                )

                unreachable
                br $interpreter_loop
            )

            ;; Push(0)
            ++$pc
            br $interpreter_loop
        )
        br $interpreter_loop
)
```



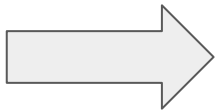

```
(loop $interpreter_loop
    (block
        (block
            (block
                (block
                    (block
                        (block
                            (i32.load (local.get $pc))
                            br_table 0 1 2 3 4
                        )

                        ;; Load script -> scratch
                        ++$pc
                        br $interpreter_loop
                    )

                    ;; Compare scratch against 0
                    if true => local.set $pc 2
                    else => local.set $pc 3
                    br $interpreter_loop
                )

                unreachable
                br $interpreter_loop
            )

            ;; Push(0)
            ++$pc
            br $interpreter_loop
        )
        br $interpreter_loop
)
```



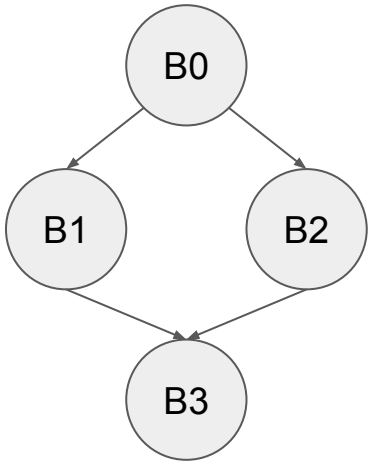

```
(loop $interpreter_loop
    (block
        (block
            (block
                (block
                    (block
                        (block
                            (i32.load (local.get $pc))
                            br_table 0 1 2 3 4
                        )

                        ;; Load script -> scratch
                        ++$pc
                        br $interpreter_loop
                    )

                    ;; Compare scratch against 0
                    if true => local.set $pc 2
                    else => local.set $pc 3
                    br $interpreter_loop
                )

                unreachable
                br $interpreter_loop
            )

            ;; Push(0)
            ++$pc
            br $interpreter_loop
    )
    br $interpreter_loop
)
```

Masm

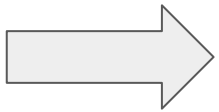
Masm



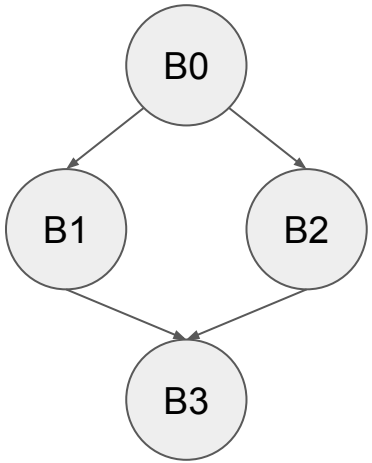
CFG

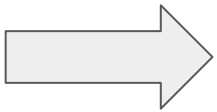


Masm



CFG

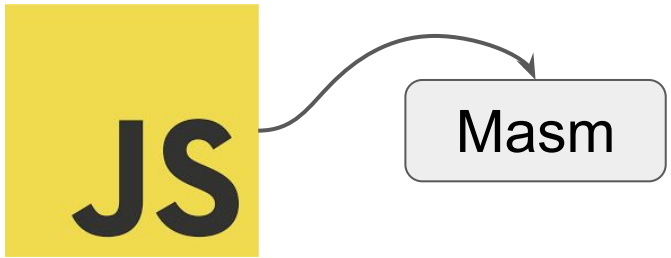


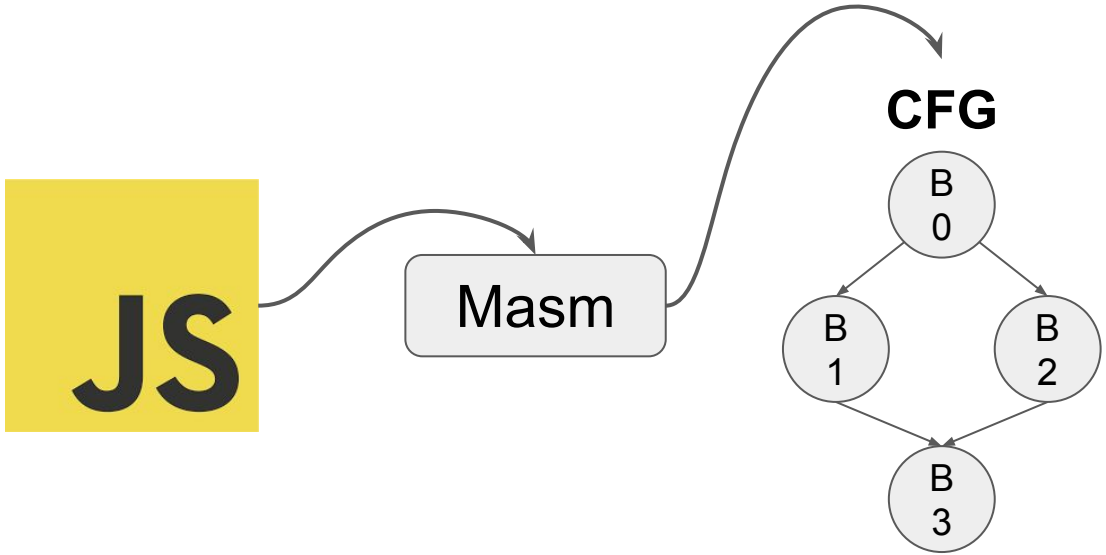


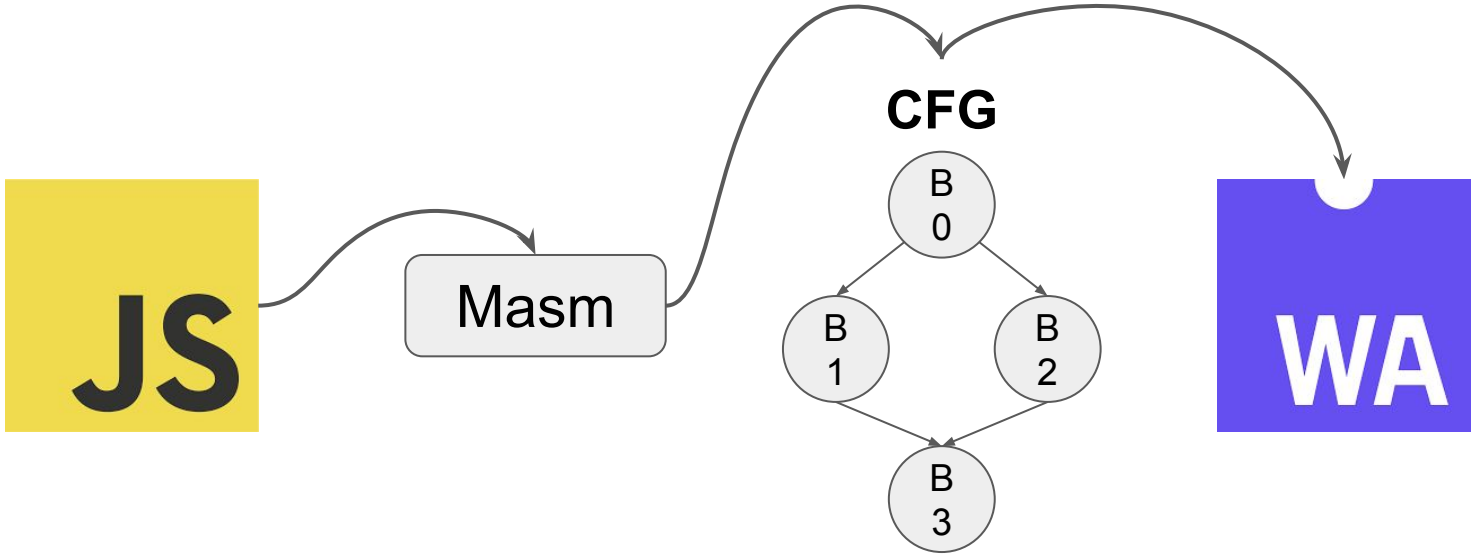
Wasm

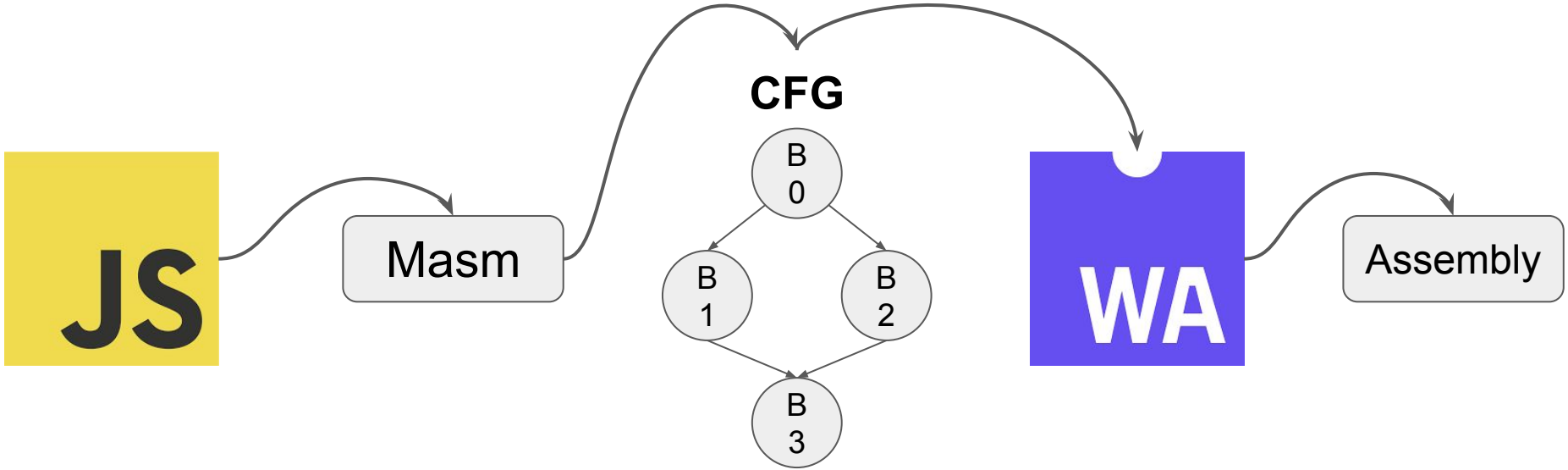
```
(loop
  (block
    (block
      (block
        (block
          (block
            …
          )
          B0
        )
        B1
      )
      B2
    )
    B3
  )
)
```

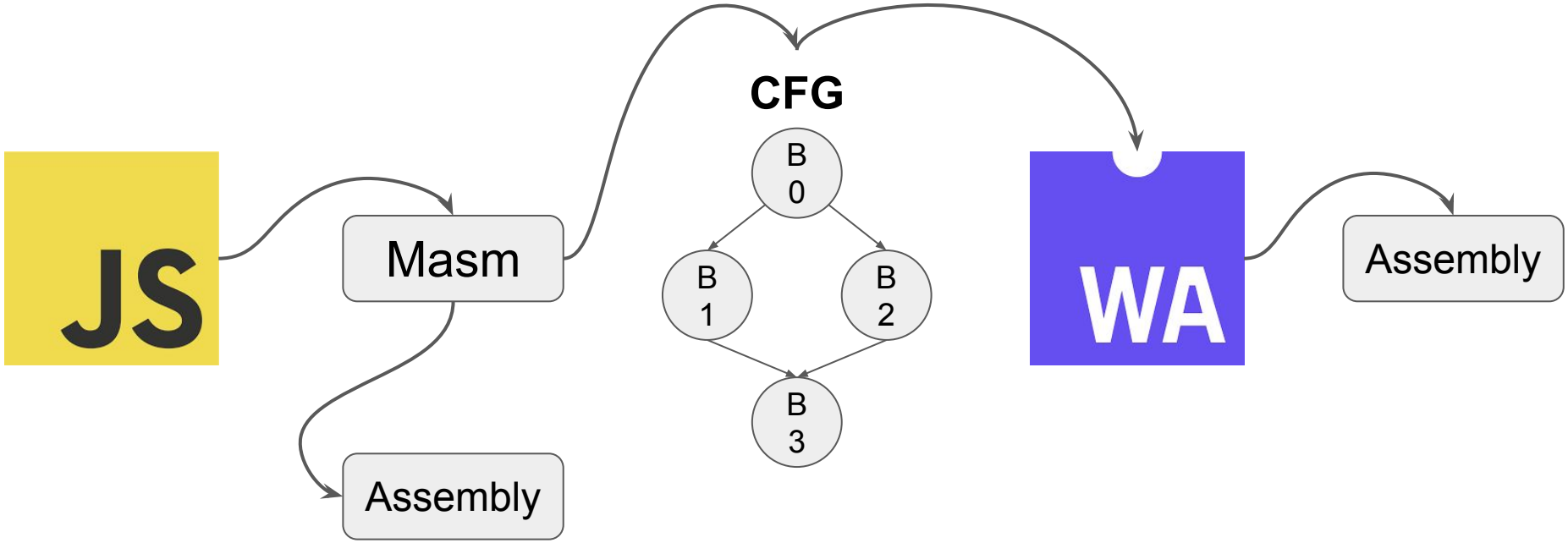

JS

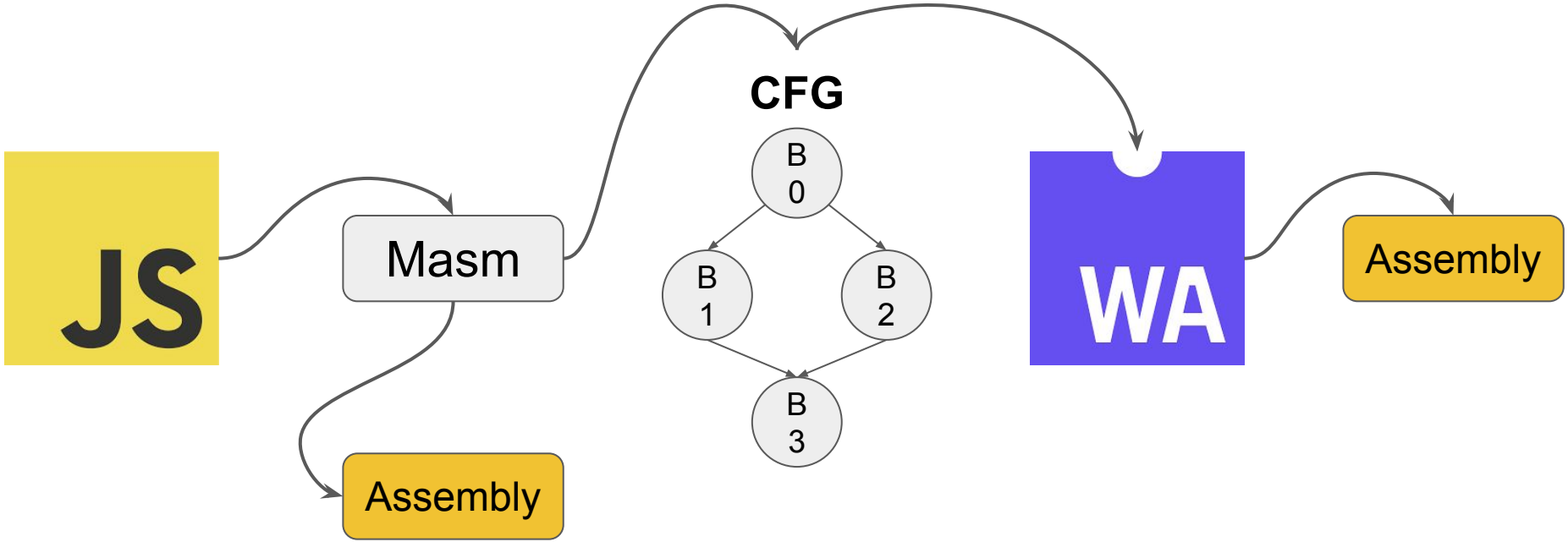

JS
Masm


JS
Masm
CFG
B
0
B
1
B
2
B
3

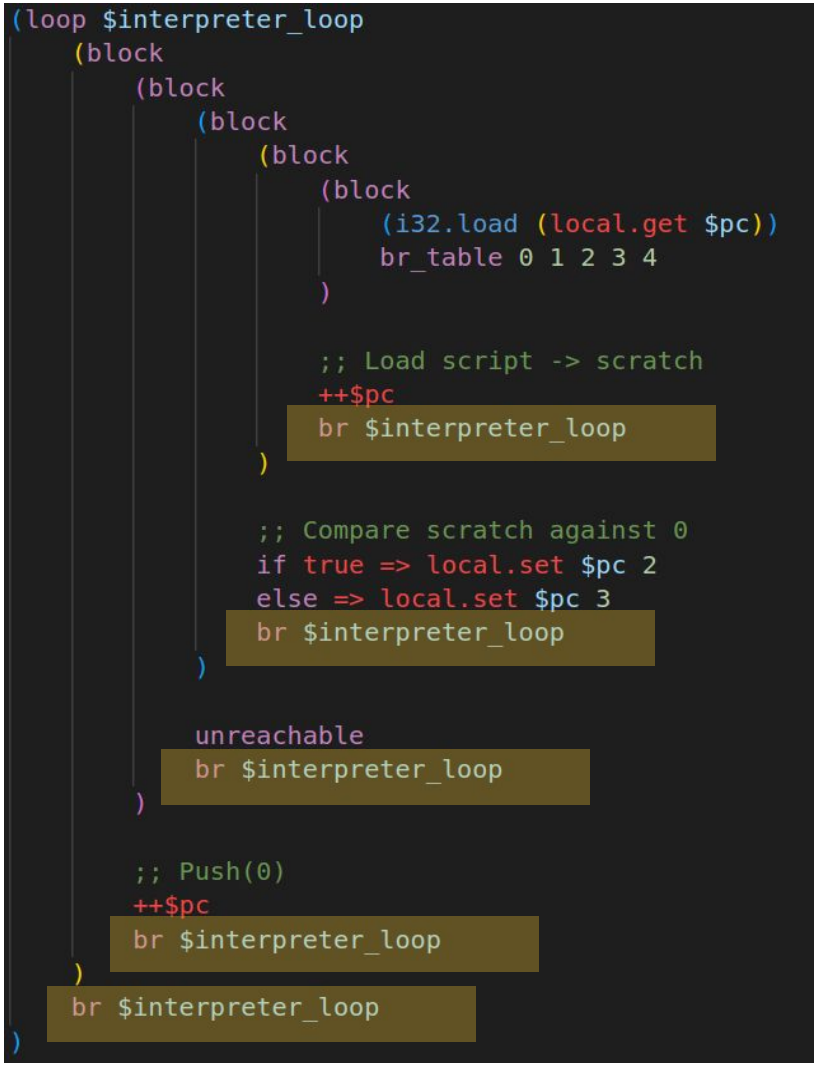

JS
Masm
CFG
B
0
B
1
B
2
B
3
WA


JS
Masm
CFG
B
0
B
1
B
2
B
3
WA
Assembly


JS
Masm
Assembly
CFG
B
0
B
1
B
2
B
3
WA
Assembly


JS
Masm
Assembly
CFG
B
0
B
1
B
2
B
3
WA
Assembly

```
(loop $interpreter_loop
    (block
        (block
            (block
                (block
                    (block
                        (block
                            (i32.load (local.get $pc))
                            br_table 0 1 2 3 4
                        )

                        ;; Load script -> scratch
                        ++$pc
                        br $interpreter_loop
                    )

                    ;; Compare scratch against 0
                    if true => local.set $pc 2
                    else => local.set $pc 3
                    br $interpreter_loop
                )

                unreachable
                br $interpreter_loop
            )

            ;; Push(0)
            ++$pc
            br $interpreter_loop
        )
        br $interpreter_loop
)
```


(loop $interpreter_loop
(block
(block
(block
(block
(block
(block
(i32.load (local.get $pc))
br_table 0 1 2 3 4
)
;; Load script -> scratch
++$pc
br $interpreter_loop
)
;; Compare scratch against 0
if true => local.set $pc 2
else => local.set $pc 3
br $interpreter_loop
)
unreachable
br $interpreter_loop
)
;; Push(0)
++$pc
br $interpreter_loop
)
br $interpreter_loop
)

Улучшенная схема компиляции

Улучшенная схема компиляции

1. В JavaScript нет goto

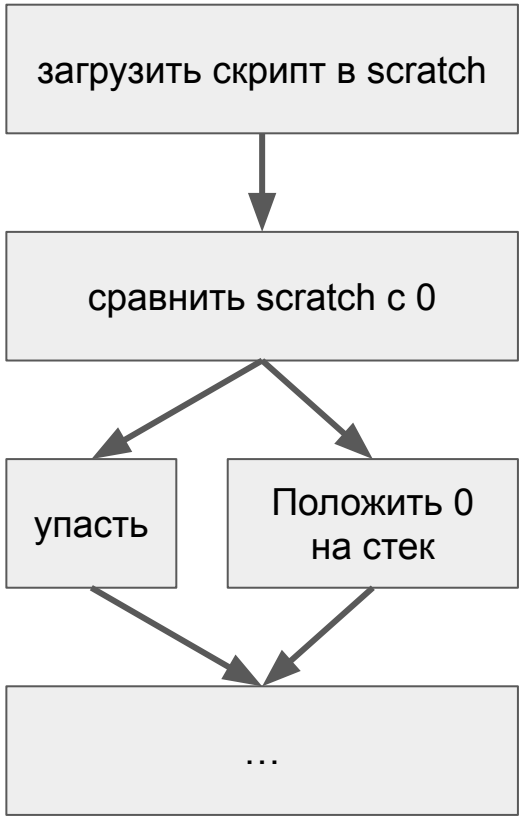
Улучшенная схема компиляции

1. В JavaScript нет goto
2. Masm генерирует structured control flow

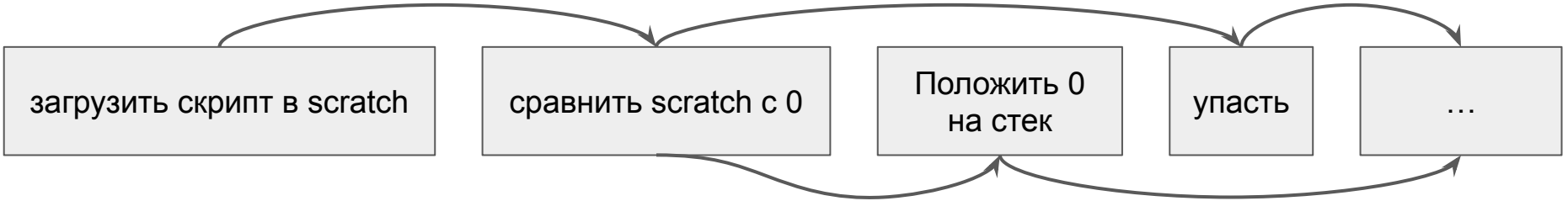
Улучшенная схема компиляции

1. В JavaScript нет goto
2. Masm генерирует structured control flow
3. Если отсортировать граф потока управления, то можно генерировать прямые переходы

Сортируем граф



Сортируем граф



```
(loop $interpreter_loop
    (block
        (block
            (block
                (block
                    (block
                        (block
                            (i32.load (local.get $pc))
                            br_table 0 1 2 3 4
                        )

                        ;; Load script -> scratch
                        ++$pc
                        br $interpreter_loop
                    )

                    ;; Compare scratch against 0
                    if true => local.set $pc 2
                    else => local.set $pc 3
                    br $interpreter_loop
                )

                unreachable
                br $interpreter_loop
            )

            ;; Push(0)
            ++$pc
            br $interpreter_loop
        )
        br $interpreter_loop
)
```

```
(loop $interpreter_loop
    (block
        (block
            (block
                (block
                    (block
                        (block
                            (i32.load (local.get $pc))
                            br_table 0 1 2 3 4
                        )

                        ;; Load script -> scratch
                        ++$pc
                        br $interpreter_loop
                    )

                    ;; Compare scratch against 0
                    if true => local.set $pc 2
                    else => local.set $pc 3
                    br $interpreter_loop
                )

                unreachable
                br $interpreter_loop
            )

            ;; Push(0)
            ++$pc
            br $interpreter_loop
        )
        br $interpreter_loop
)
```

```
(loop $interpreter_loop
    (block
        (block
            (block
                (block
                    (block
                        (i32.load (local.get $pc))
                        br_table 0 1 2 3 4
                    )

                    ;; Load script -> scratch
                    ++$pc
                    br $interpreter_loop
                )

                ;; Compare scratch against 0
                if true => local.set $pc 2
                else => local.set $pc 3
                br $interpreter_loop
            )

            unreachable
            br $interpreter_loop
        )

        ;; Push(0)
        ++$pc
        br $interpreter_loop
    )
    br $interpreter_loop
)
```

```
(block
    (block
        (block
            (block
                ;; Load script -> scratch
                ++$pc
                br 0
            )
            ;; Compare scratch against 0
            if true => br 2
            else => br 3
        )
        unreachable
    )
    ;; Push(0)
    br 0
)
```

```
(loop $interpreter_loop
    (block
        (block
            (block
                (block
                    (block
                        (i32.load (local.get $pc))
                        br_table 0 1 2 3 4
                    )

                    ;; Load script -> scratch
                    ++$pc
                    br $interpreter_loop
                )

                ;; Compare scratch against 0
                if true => local.set $pc 2
                else => local.set $pc 3
                br $interpreter_loop
            )

            unreachable
            br $interpreter_loop
        )

        ;; Push(0)
        ++$pc
        br $interpreter_loop
    )
    br $interpreter_loop
)
```

```
(block
    (block
        (block
            (block
                ;; Load script -> scratch
                ++$pc
                br 0
            )
            ;; Compare scratch against 0
            if true => br 2
            else => br 3
        )
        unreachable
    )
    ;; Push(0)
    br 0
)
```

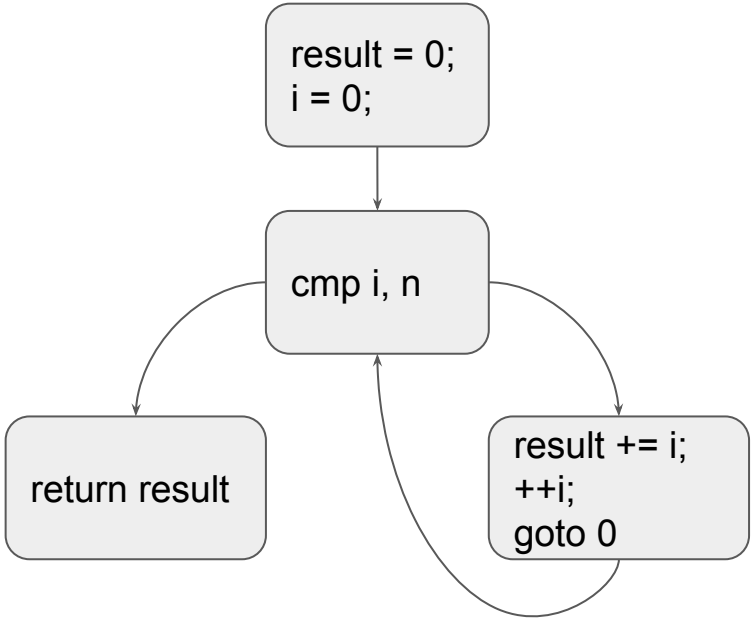
Но есть один нюанс

Но есть один нюанс

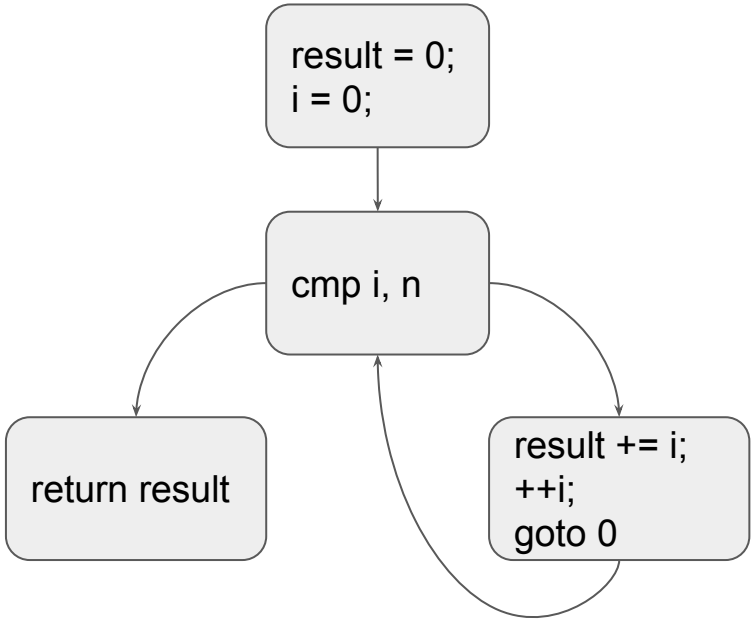
```
function sum(n) {
    let result = 0;
    for (let i = 0; i < n; ++i) {
        result += i;
    }
    return result;
}
```

Но есть один нюанс

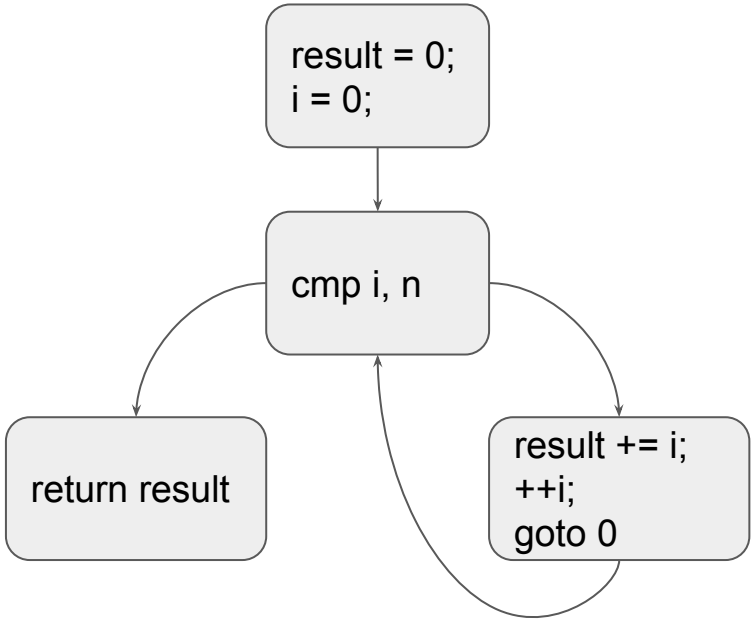
```
function sum(n) {
    let result = 0;
    for (let i = 0; i < n; ++i) {
        result += i;
    }
    return result;
}
```



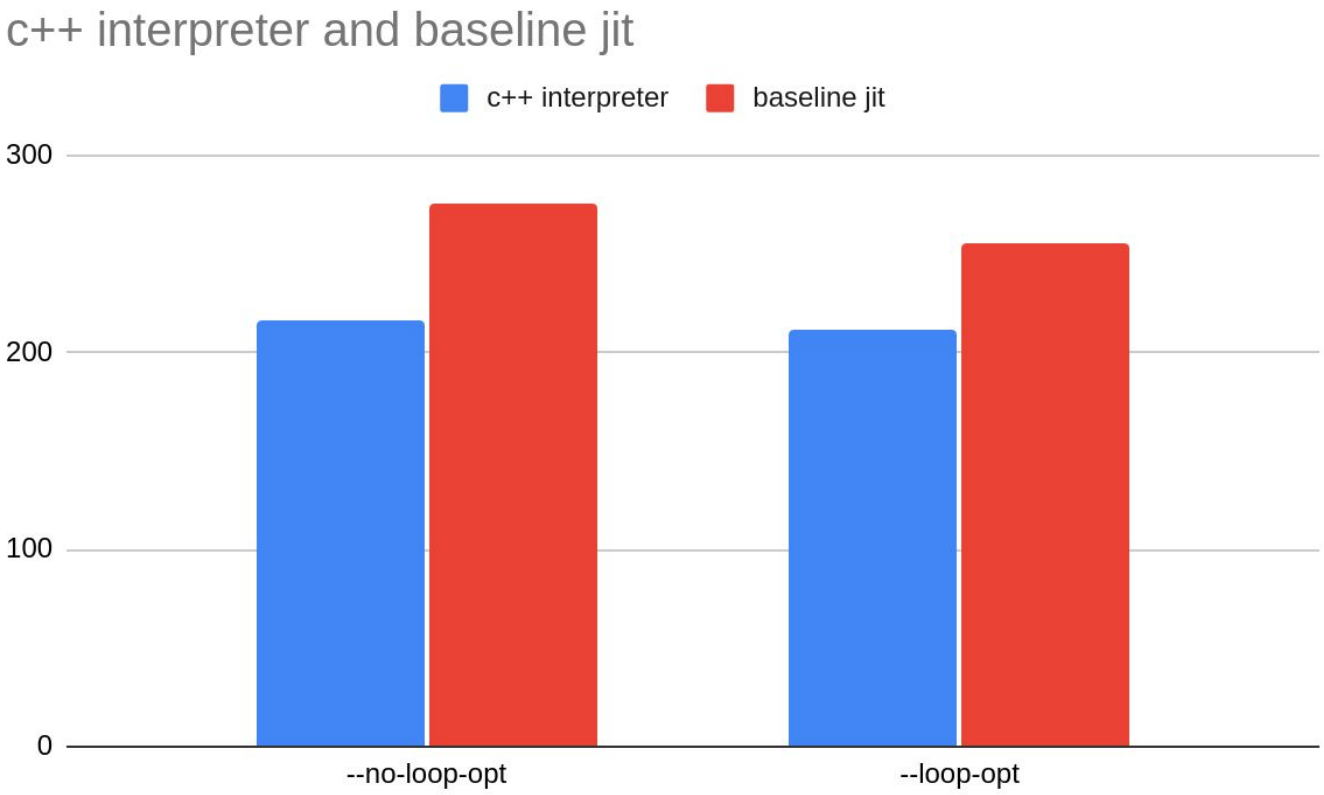
Отслеживаем циклы



Отслеживаем циклы




```
(block
    (block
        (block
            result = 0
            i = 0;
        )
        (loop
            (block
                cmp i, n
                br_if 1
            )
            result += i;
            ++i;
            br 0
        )
    )
    return result;
)
```

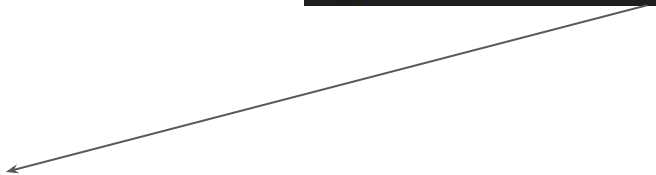

c++ interpreter and baseline jit
c++ interpreter
baseline jit
300
200
100
0
--no-loop-opt
--loop-opt

JavaScript сложный язык

```
function add(a, b) {
    return a + b;
}
```

JavaScript сложный язык

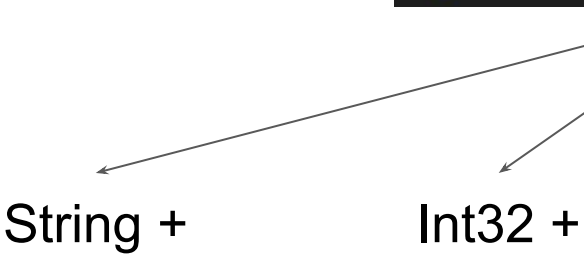
```
function add(a, b) {
    return a + b;
}
```



String +

JavaScript сложный язык

```
function add(a, b) {
    return a + b;
}
```

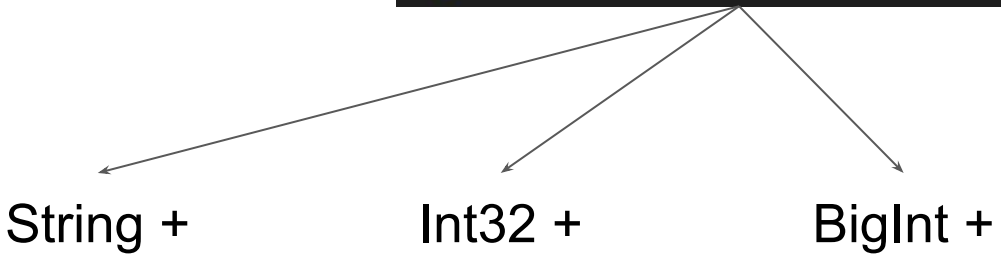


String +

Int32 +

JavaScript сложный язык

```
function add(a, b) {
    return a + b;
}
```



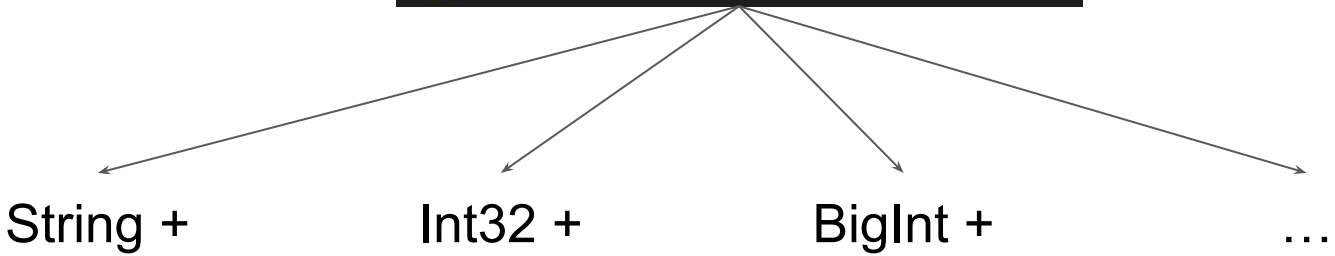
String +

Int32 +

BigInt +

JavaScript сложный язык

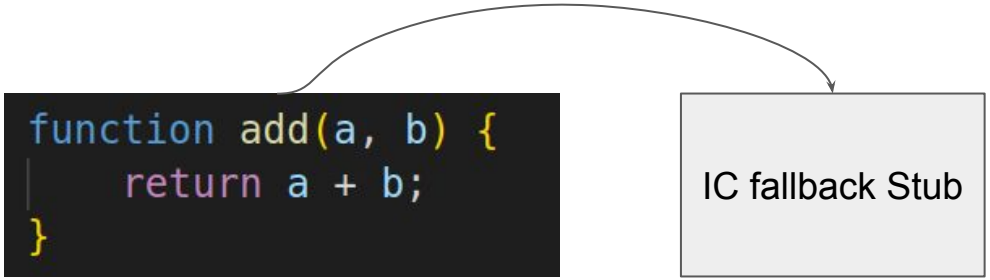
```
function add(a, b) {
    return a + b;
}
```



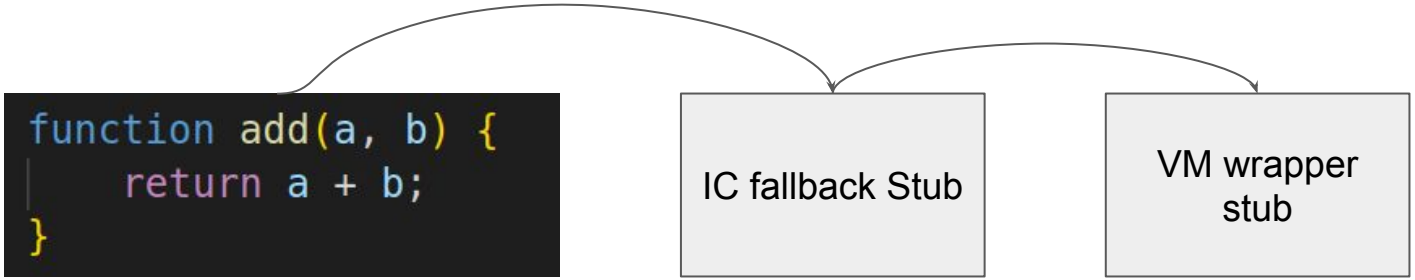
Инлайн кэши: текущая реализация

```
function add(a, b) {
    return a + b;
}
```

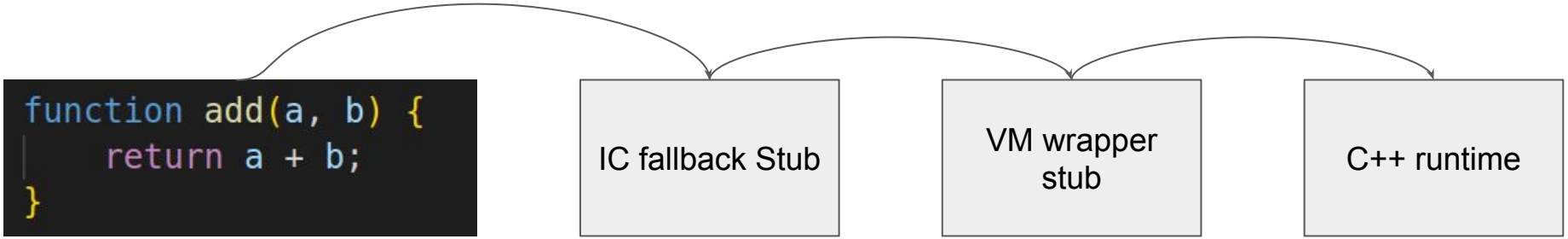
Инлайн кэши: текущая реализация



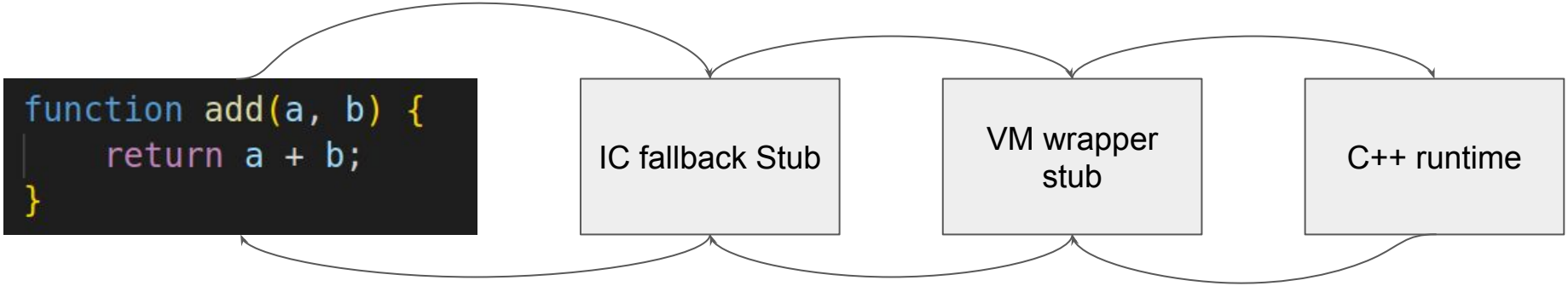
Инлайн кэши: текущая реализация



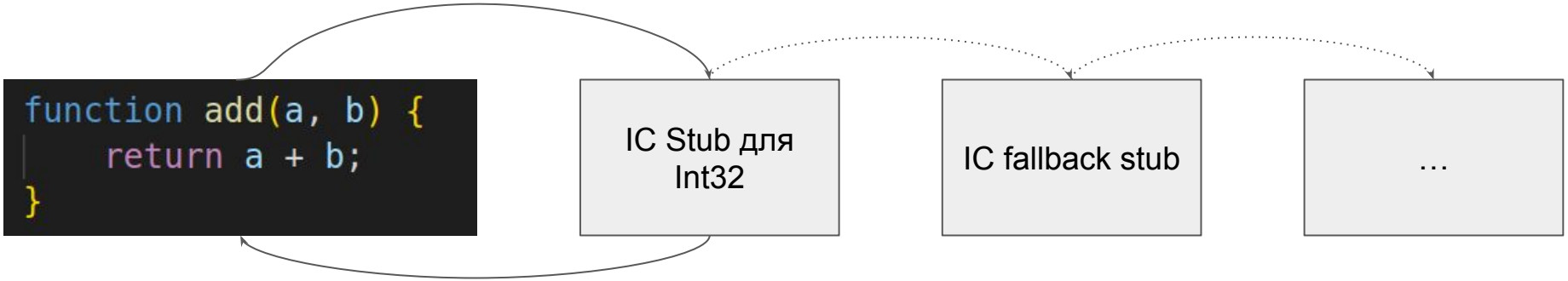
Инлайн кэши: текущая реализация



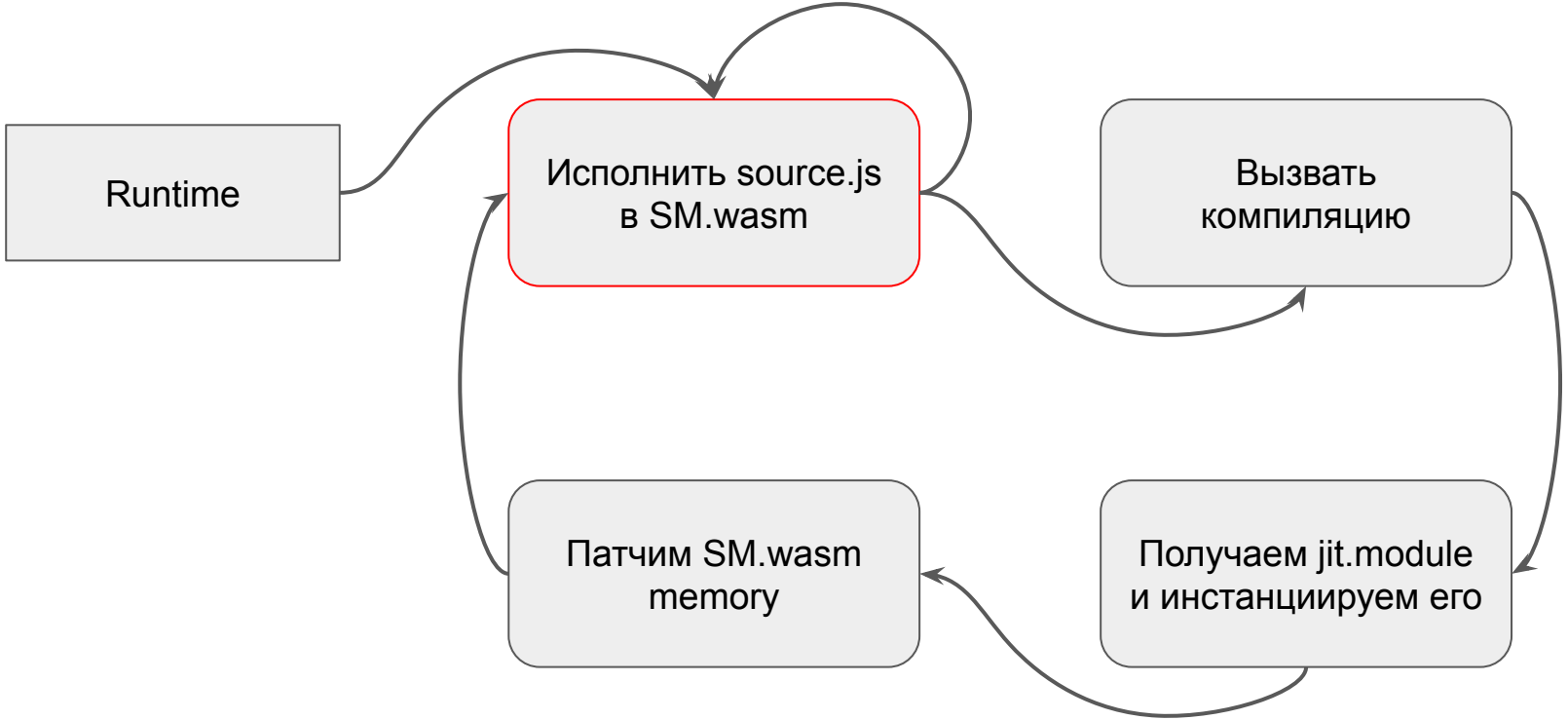
Инлайн кэши: текущая реализация



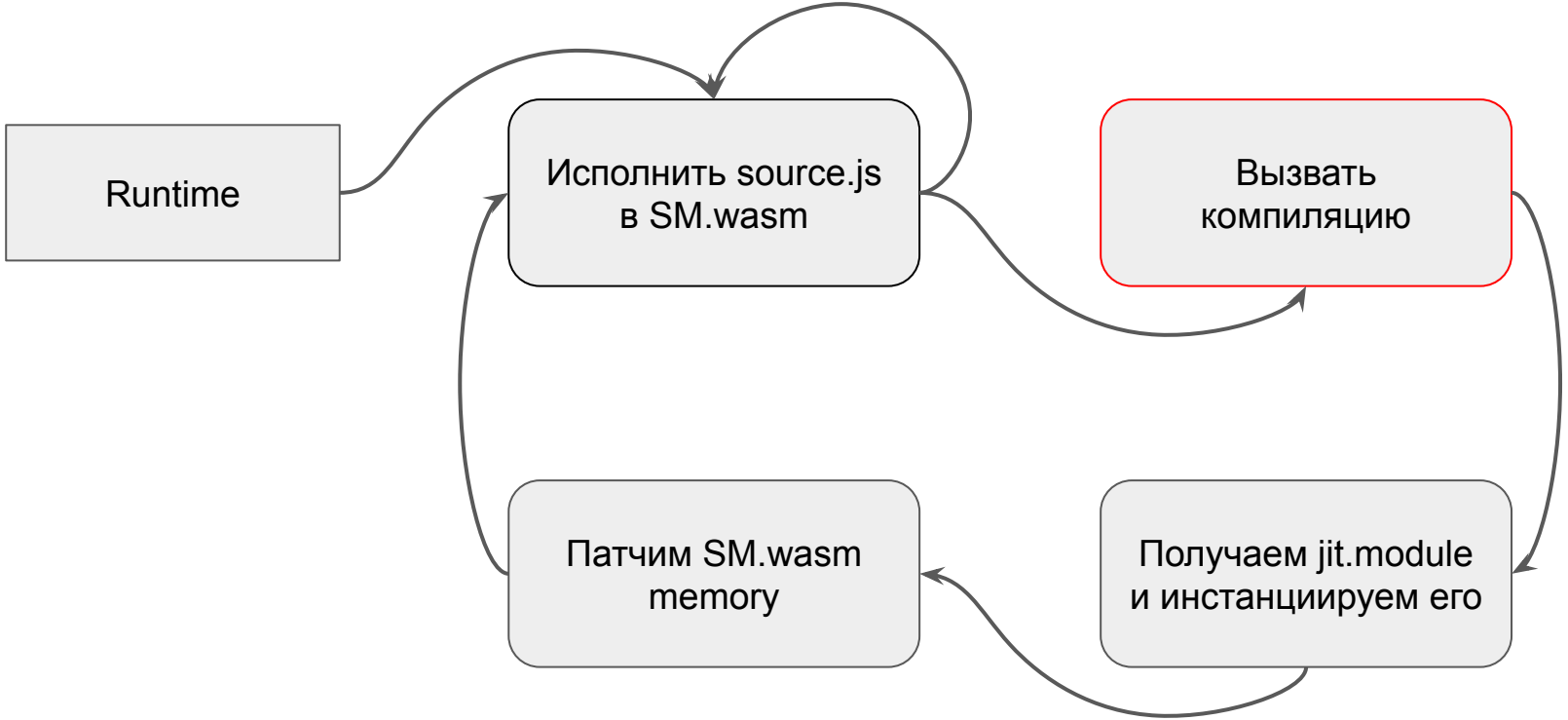
Инлайн кэши: как хочется



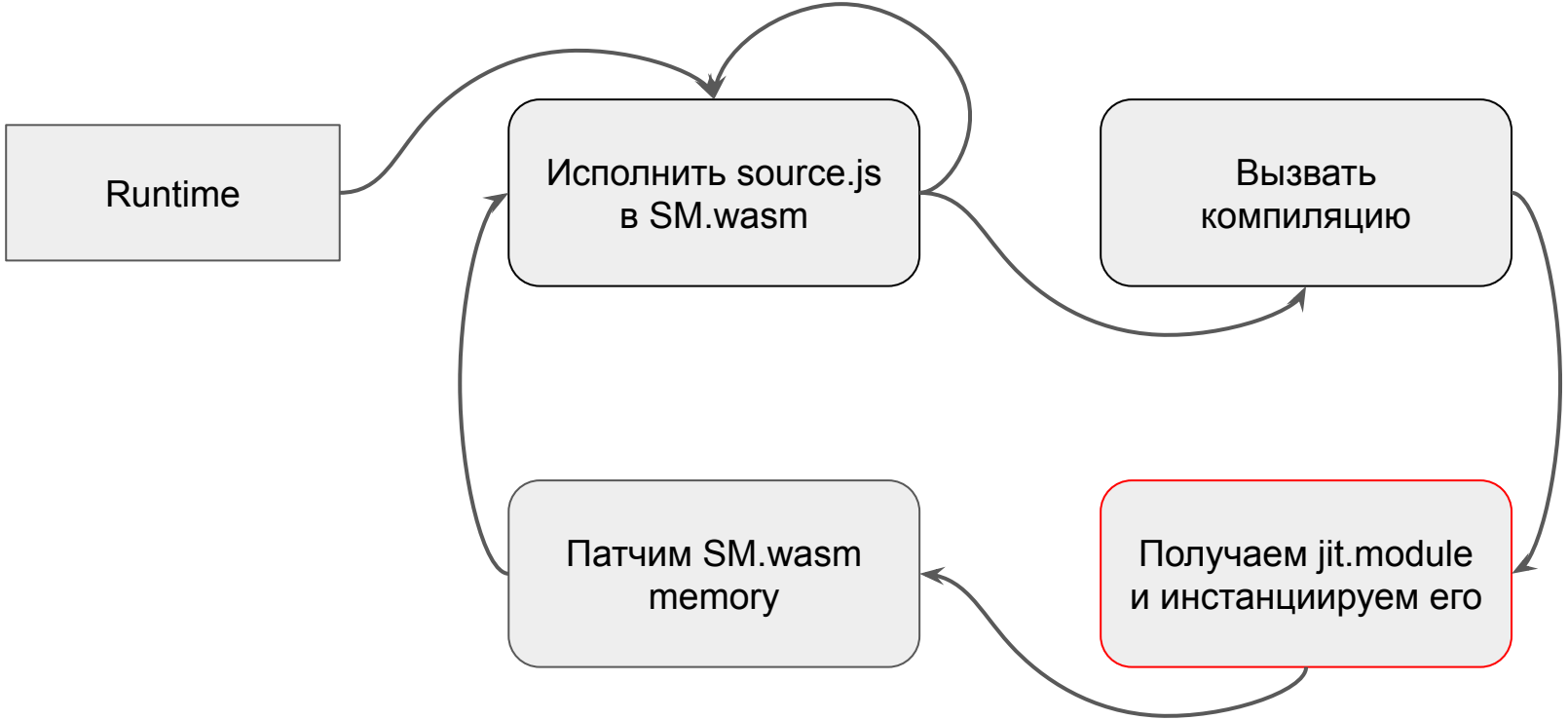
Асинхронный JIT еще раз



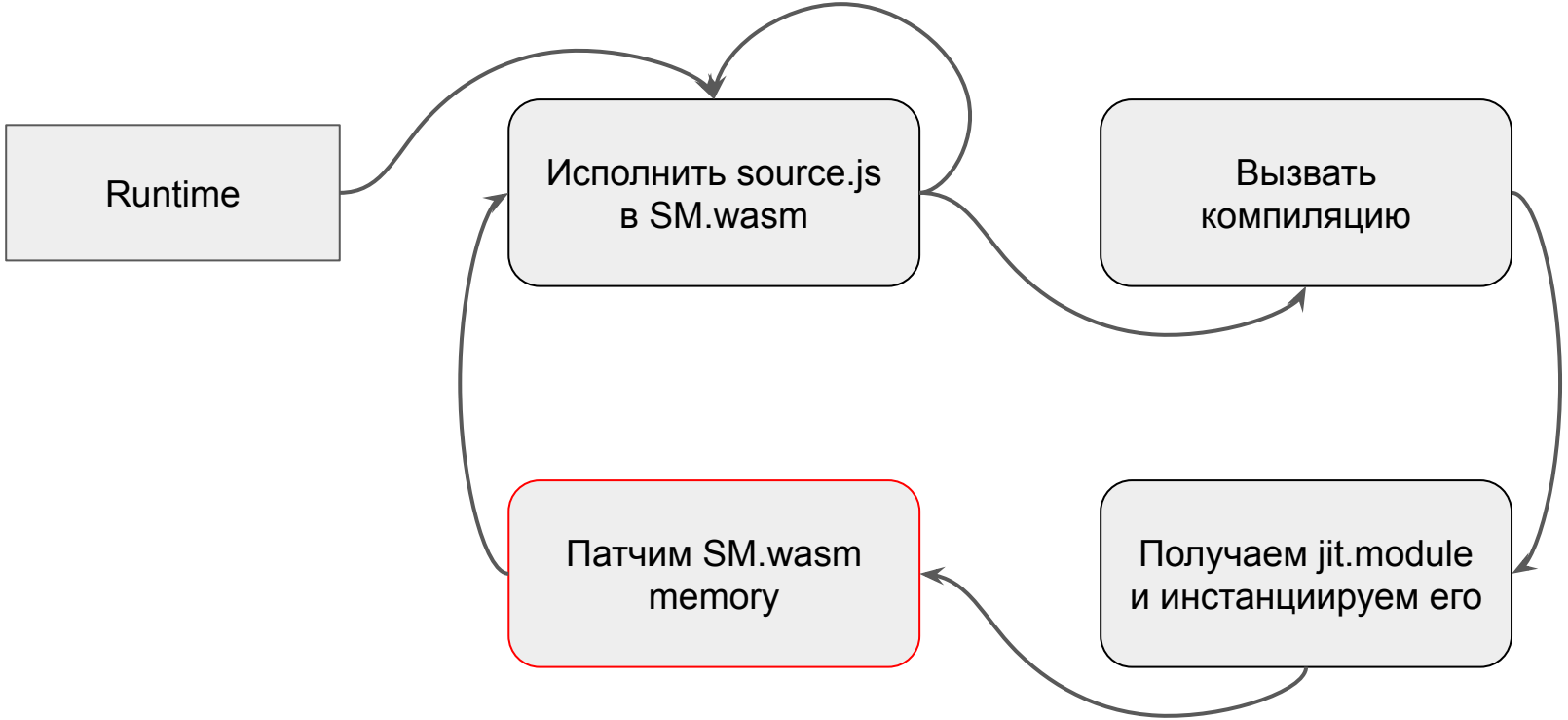
Асинхронный JIT еще раз



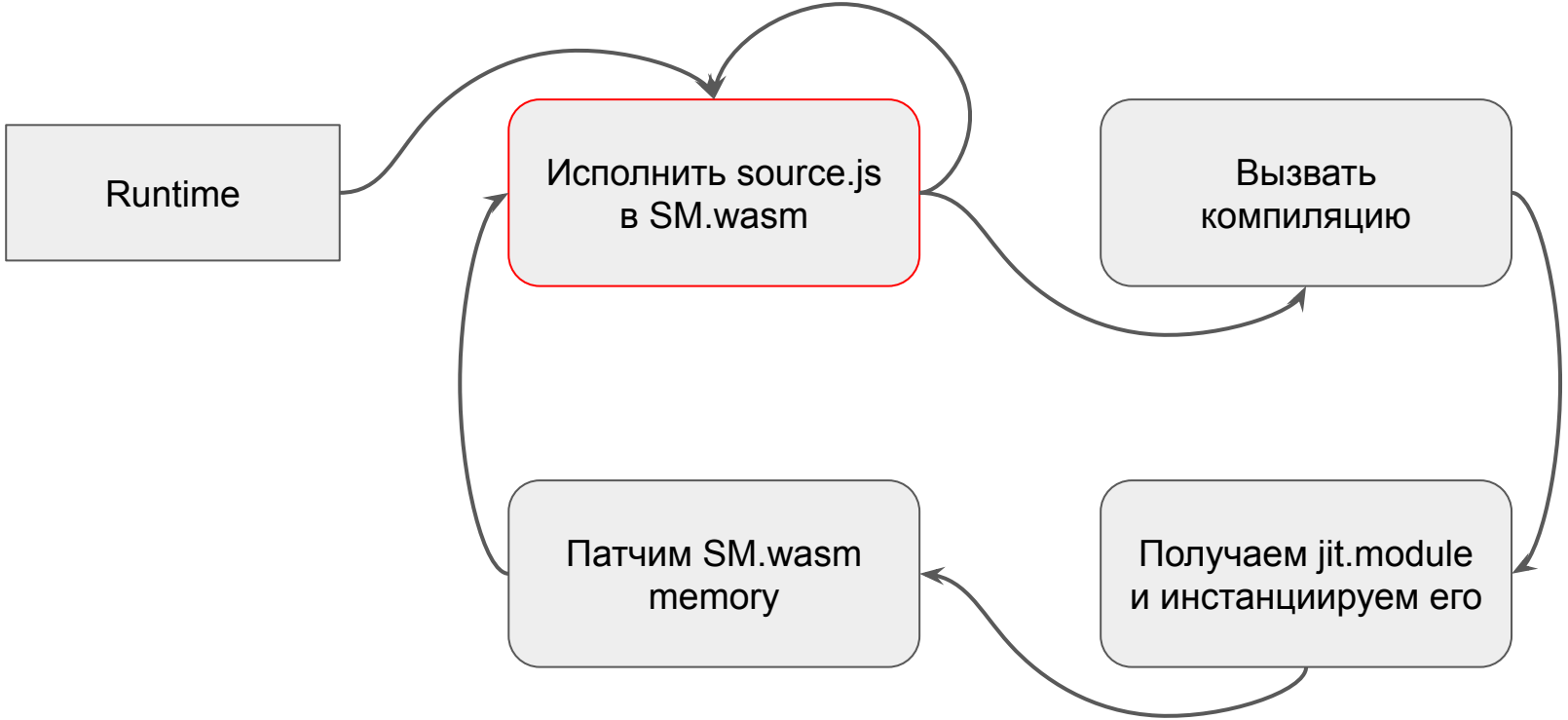
Асинхронный JIT еще раз



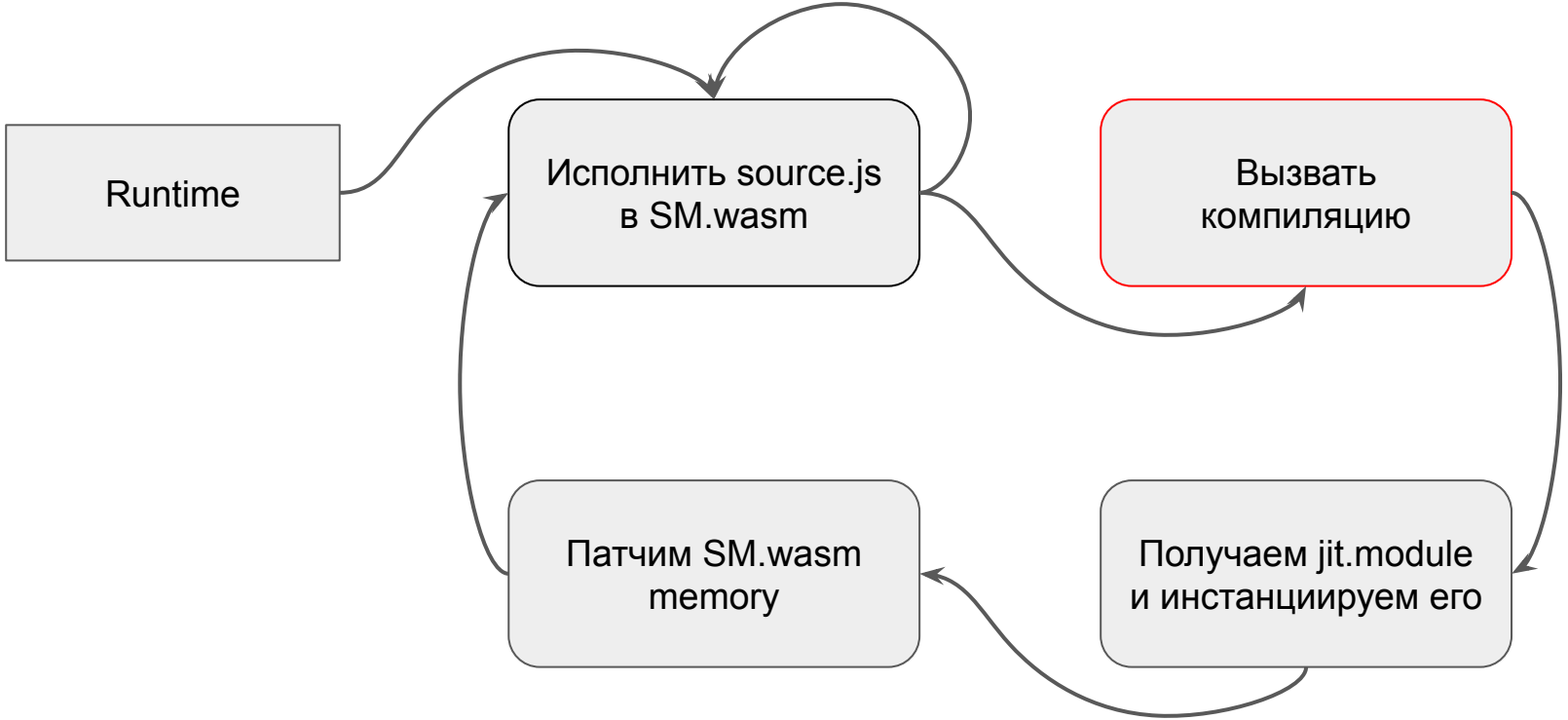
Асинхронный JIT еще раз



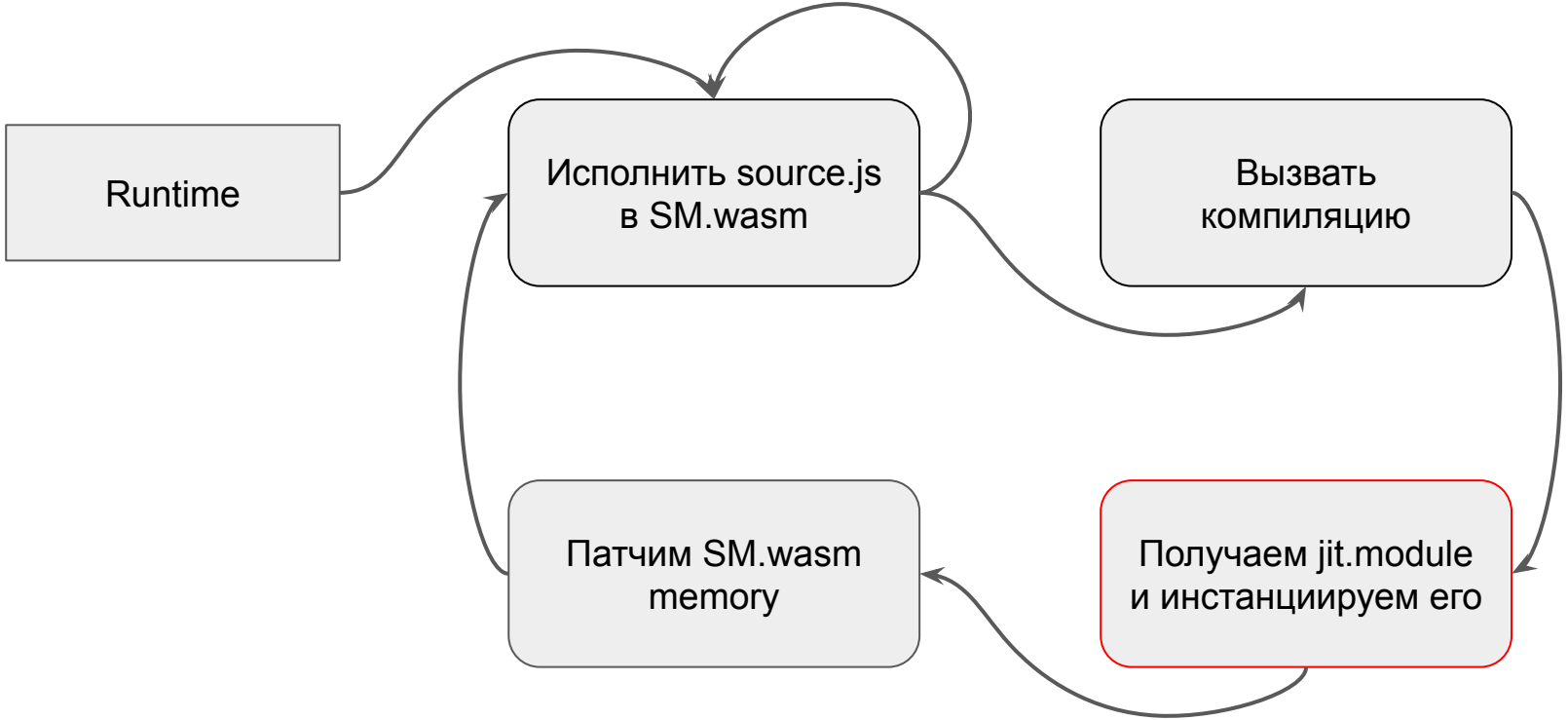
Асинхронный JIT еще раз



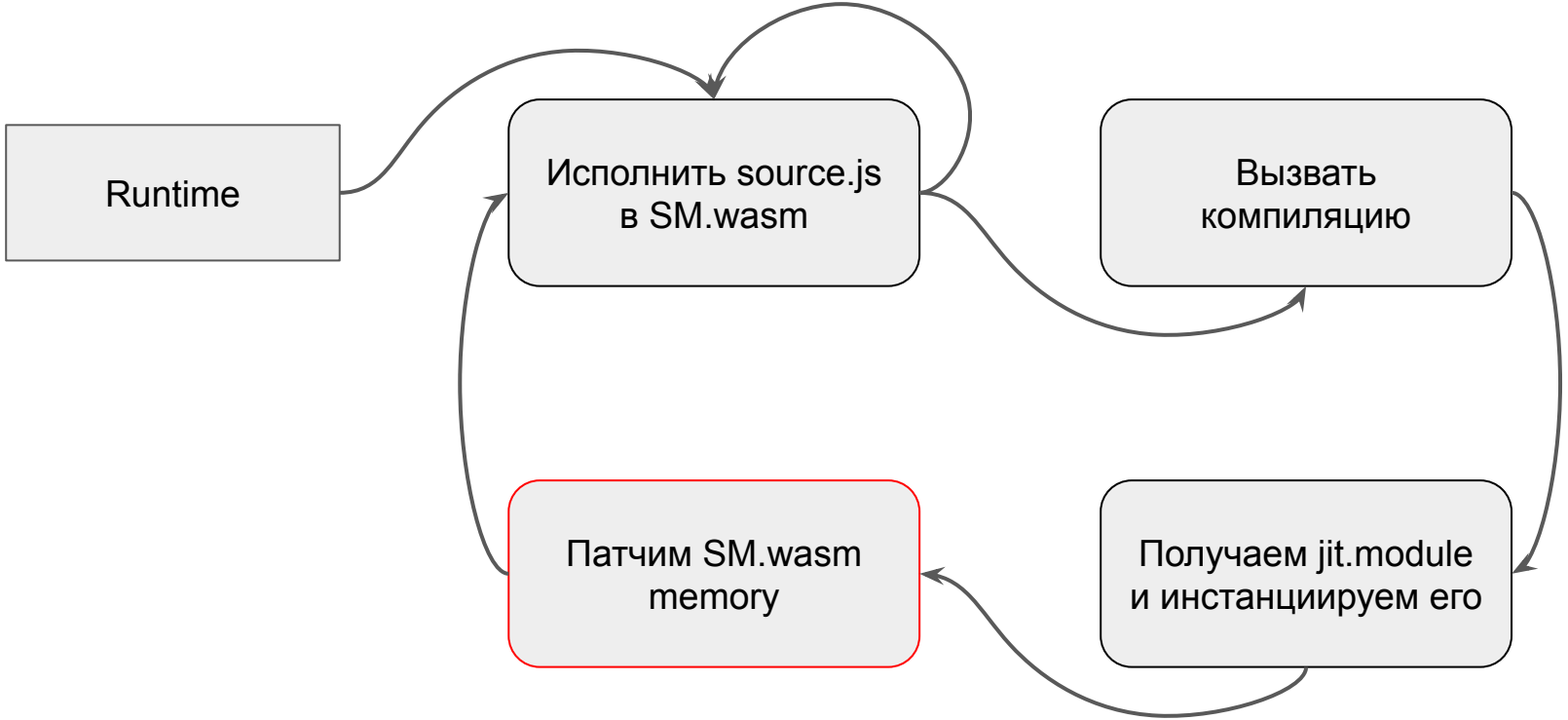
Асинхронный JIT еще раз



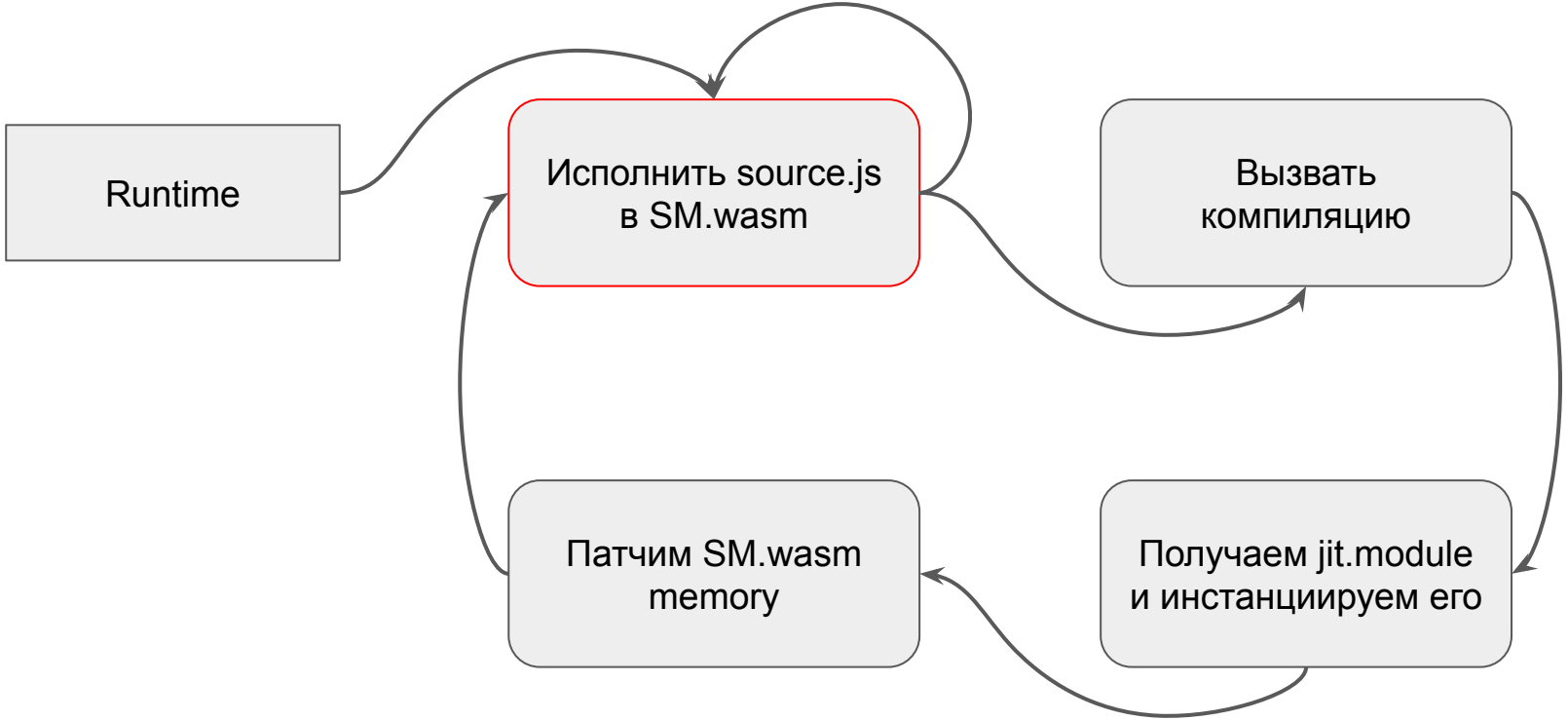
Асинхронный JIT еще раз



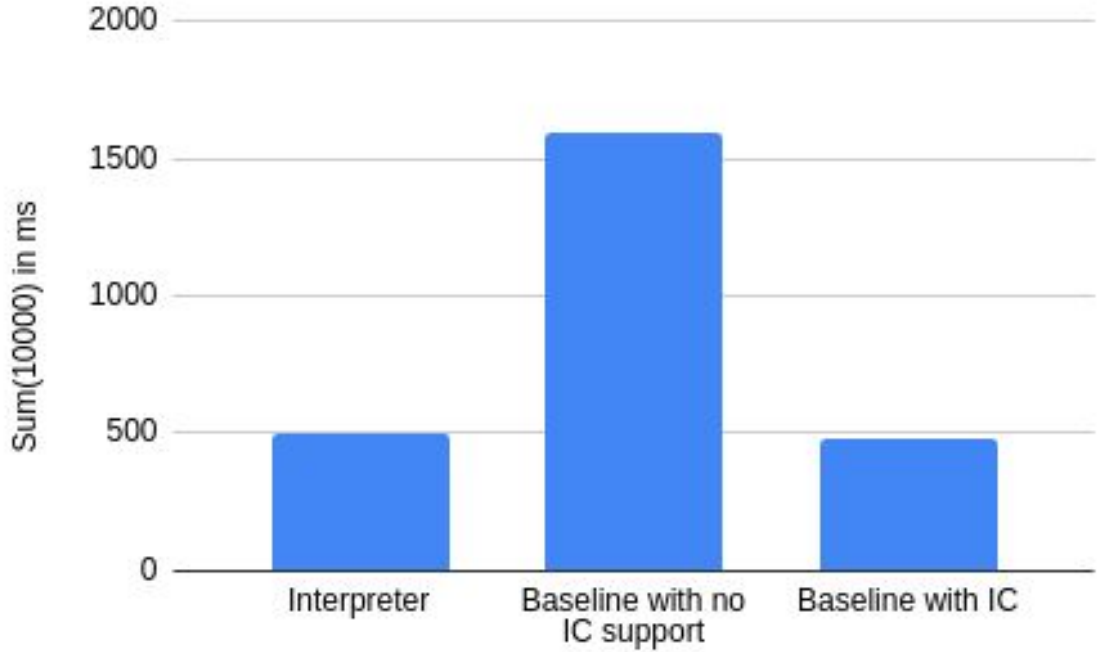
Асинхронный JIT еще раз



Асинхронный JIT еще раз



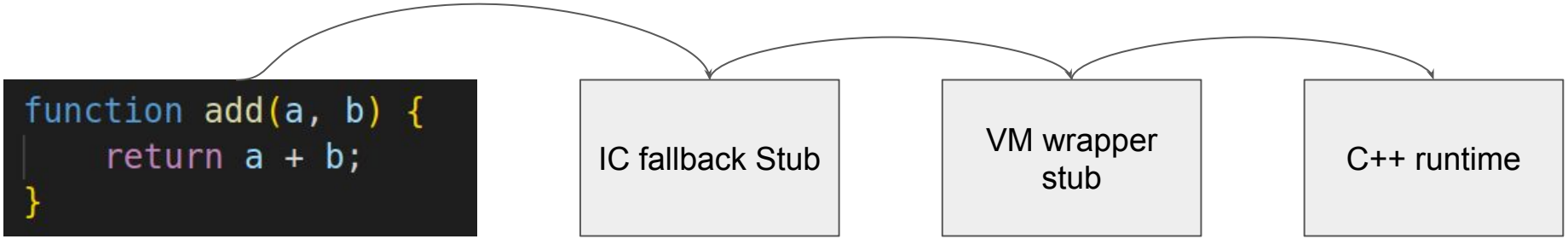
Результаты


2000
1500
1000
500
0
Sum(10000) in ms
Interpreter
Baseline with no IC support
Baseline with IC

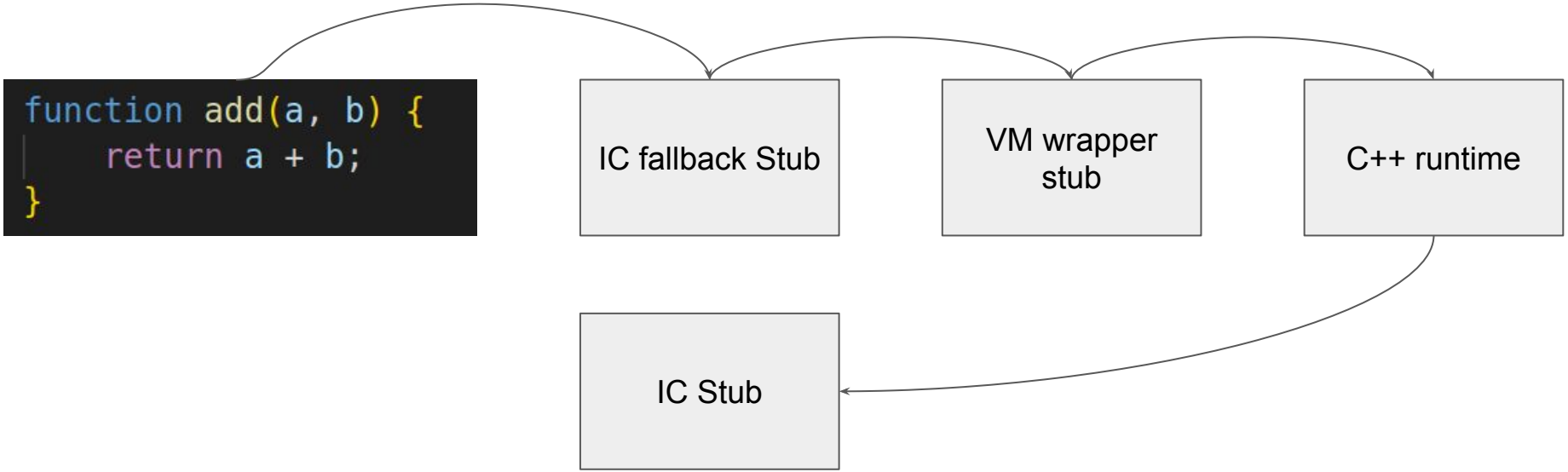
Проблемы

1. Регистры как глобалы - не очень хорошая идея оказалась
2. Вызов IC через *call_indirect* в wasmtime генерирует очень много оверхеда

Генерация прямых вызовов и инлайнинг ICs

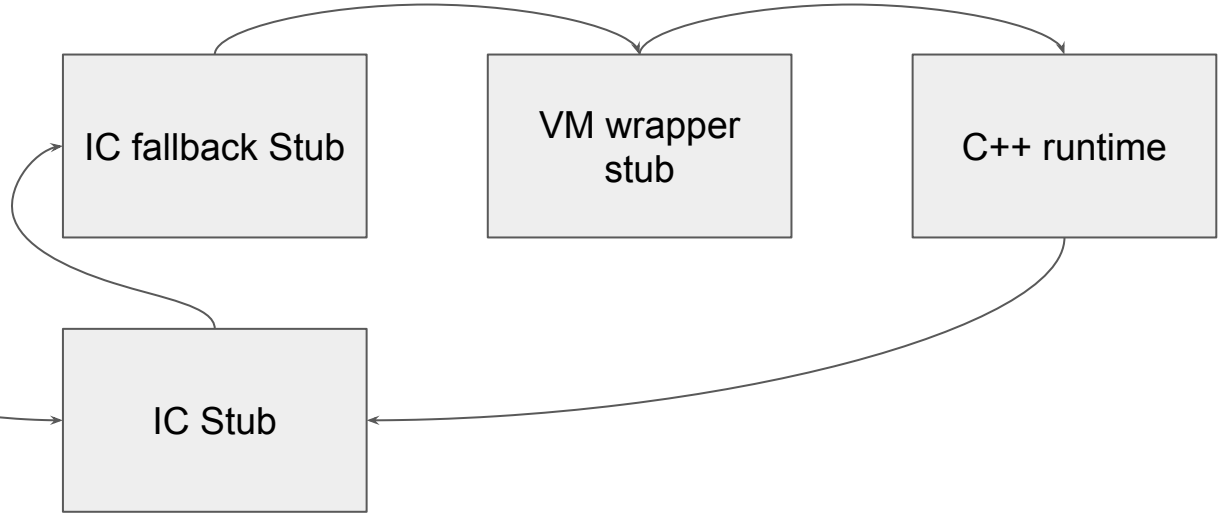


Генерация прямых вызовов и инлайнинг ICs



Генерация прямых вызовов и инлайнинг ICs

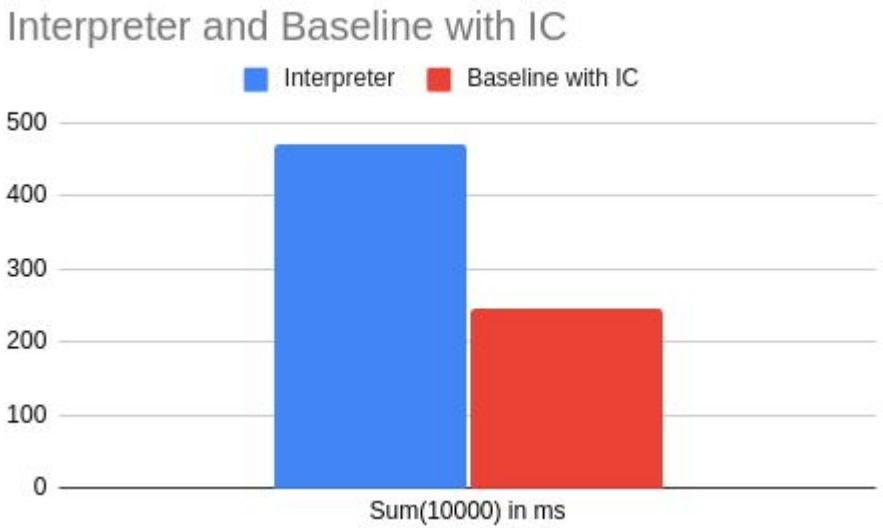
```
function add(a, b) {
    return a + b;
}
```



Генерация прямых вызовов и инлайнинг ICs

- Мы не инстанциируем новых IC стабов в рантайме, мы только откладываем их на потом
- Мы контролируем вызовы компиляции

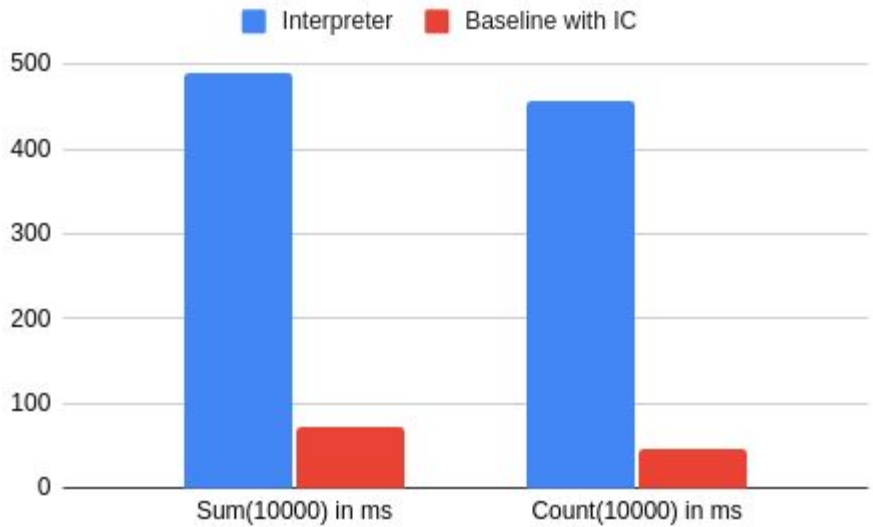
Результаты


Interpreter and Baseline with IC
Interpreter
Baseline with IC
500
400
300
200
100
0
Sum(10000) in ms

Конвенция вызовов и представление регистров

1. Доступ к глобалам не оптимизируется аллокатором регистров
2. IC вызовы используют свой кастомный ABI, который мы не можем выразить в wasm
3. Поэтому мы решили сделать свою конвенцию вызовов:

   f: (fp: i32, R0: i64, R1: i64, ICStubReg: i64) -> i64


Interpreter
Baseline with IC
500
400
300
200
100
0
Sum(10000) in ms
Count(10000) in ms

Дальнейшие шаги

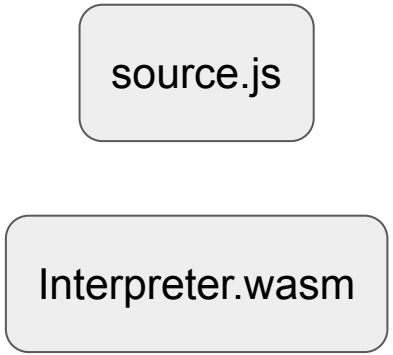
1. Сборка всех промежуточных модулей в один, т.е. source.js -> source.wasm.
2. Оптимизируем размер получаемого source.wasm.
3. Мы все заапстримим в mozilla к концу года

Заключение

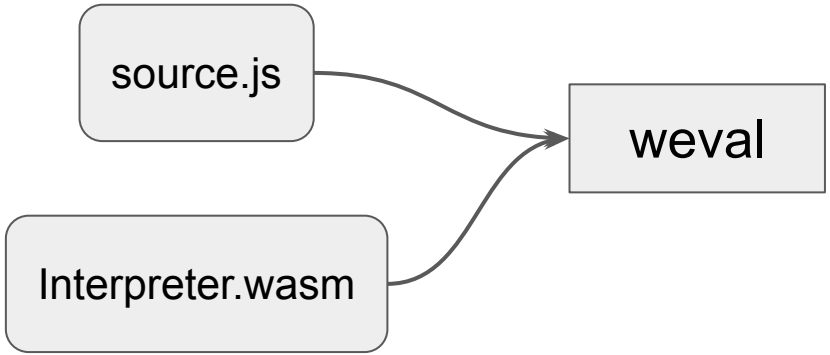
1. Теперь есть решение быстрее в ~9 раз чем quickjs
2. Компиляция в wasm это просто, но добиться хорошей производительности сложно
3. JS теперь запускать в облаках через wasm стало еще приятнее

Bonus: что делать если я хочу примерно тоже самое, но для python/php/ruby

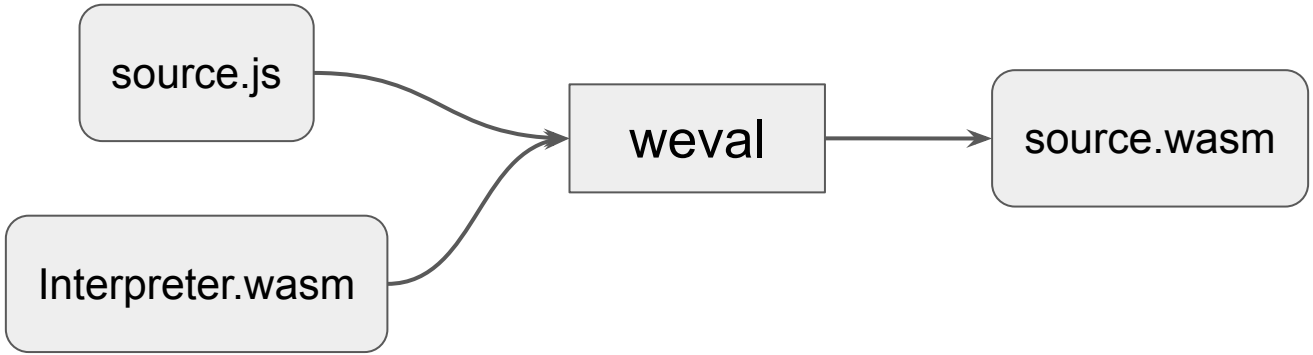
Bonus: что делать если я хочу примерно тоже самое, но для python/php/ruby



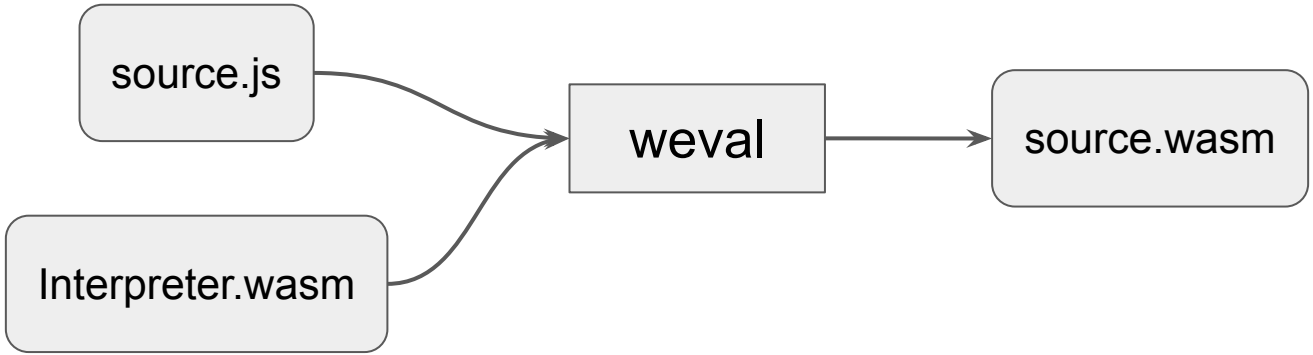
Bonus: что делать если я хочу примерно тоже самое, но для python/php/ruby



Bonus: что делать если я хочу примерно тоже самое, но для python/php/ruby



Bonus: что делать если я хочу примерно тоже самое, но для python/php/ruby



Partial evaluator на wasm: https://github.com/cfallin/weval

Extra bonus: AOT для JS

https://blogs.igalia.com/compilers/2023/05/10/compiling-bigloo-scheme-to-webassembly/

https://www.youtube.com/watch?v=iY1EXHQ6IeQ

Спасибо за внимание