Running Optcarrot (faster) on my own Ruby.

monochrome

 @s_isshiki1969

 sisshiki1969

monoruby

- https://github.com/sisshiki1969/monoruby 
- Ruby implementation with JIT compiler
- Written in Rust from (almost) scratch
- Only x86-64 / Linux is supported
- Motivation

    run optcarrot faster.
- Not intended to

    run Rails.

Compatibility

- Implementation stage: early alpha
- Aim to be compatible with CRuby(MRI).
- Not aim to be drop-in replacement of CRuby.
- Does NOT support
  - Native C extensions (has alternatives)
  - Native threads (Fiber is supported)
  - encoding: supports only UTF-8 and ASCII-8BIT
  - ObjectSpace, TracePoint, Refinements, ..

- Parser
    - recursive descent parser
    - hand-written
    - https://github.com/sisshiki1969/ruruby-parse

- Garbage collector
    - mark and sweep, stop-the-world
    - precise, non-moving
    - bitmap marking

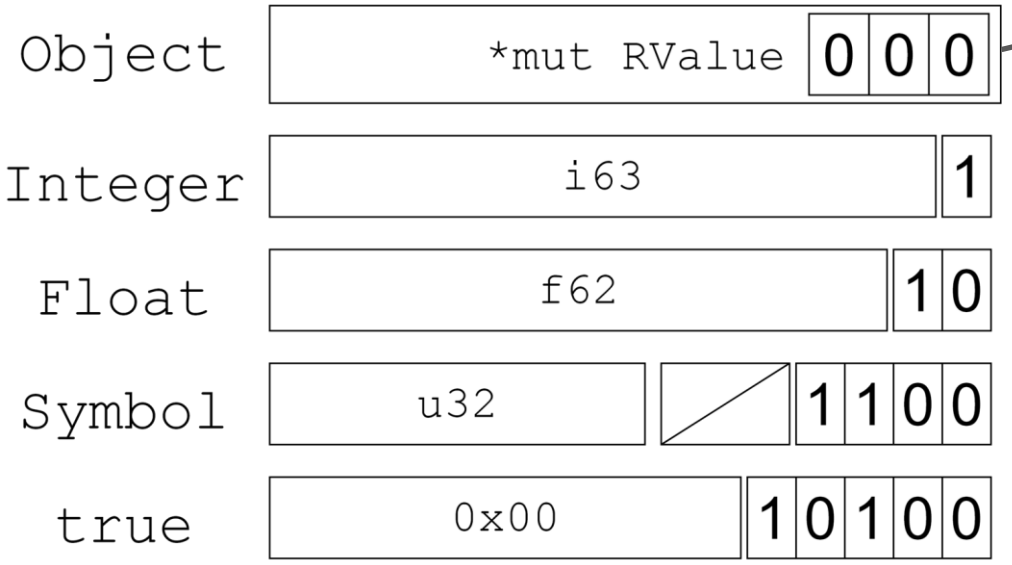
- Dynamic assembler
  runtime x86-64 assembler
  using proc_macro of Rust

https://github.com/sisshiki1969/monoasm

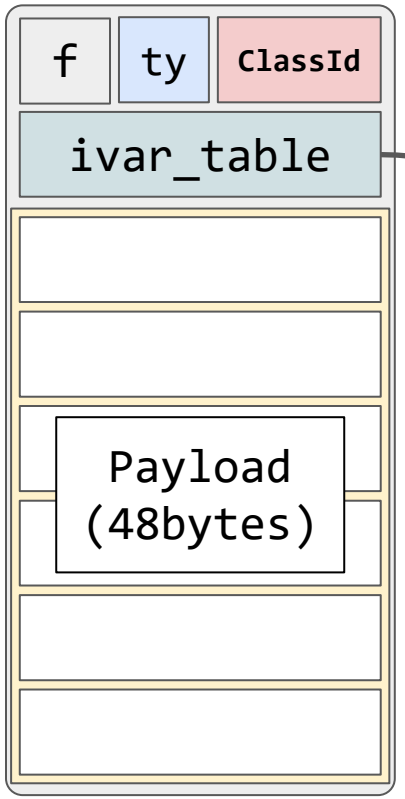
```
fn call_arg1(&mut self, dest: DestLabel, ret: u64) {
  monoasm!(self.jit,
    movq rdi, (42 + 100);
    call dest;
    movq R(ret + 12), rax;
  );
}
```

in-memory representation of Ruby objects

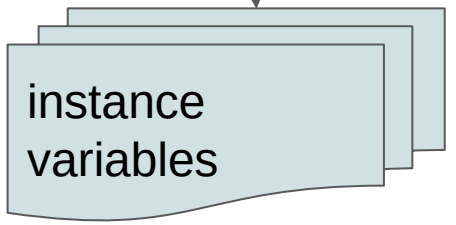
```
struct Value(std::num::NonZeroU64);
```



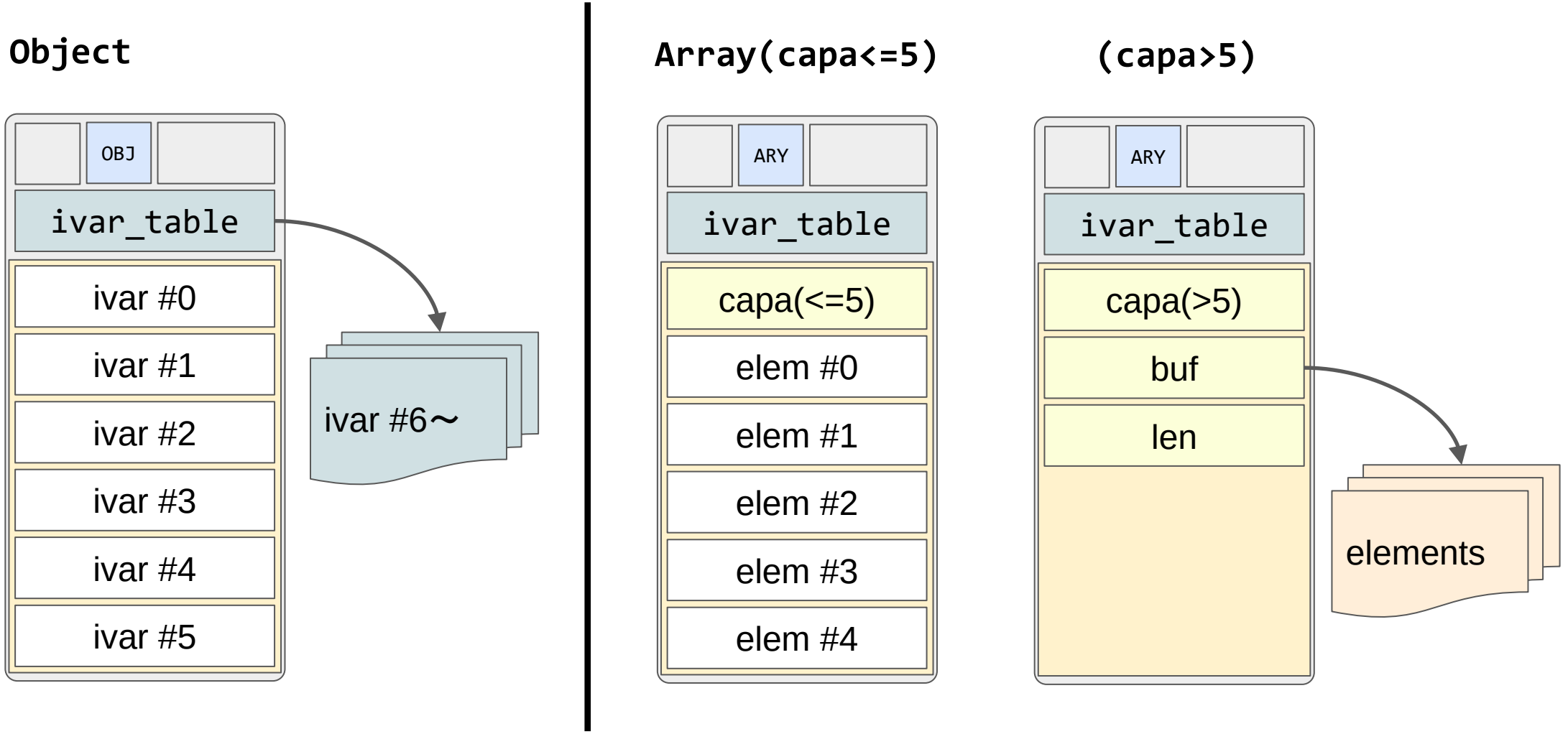
`struct RValue` 64 bytes



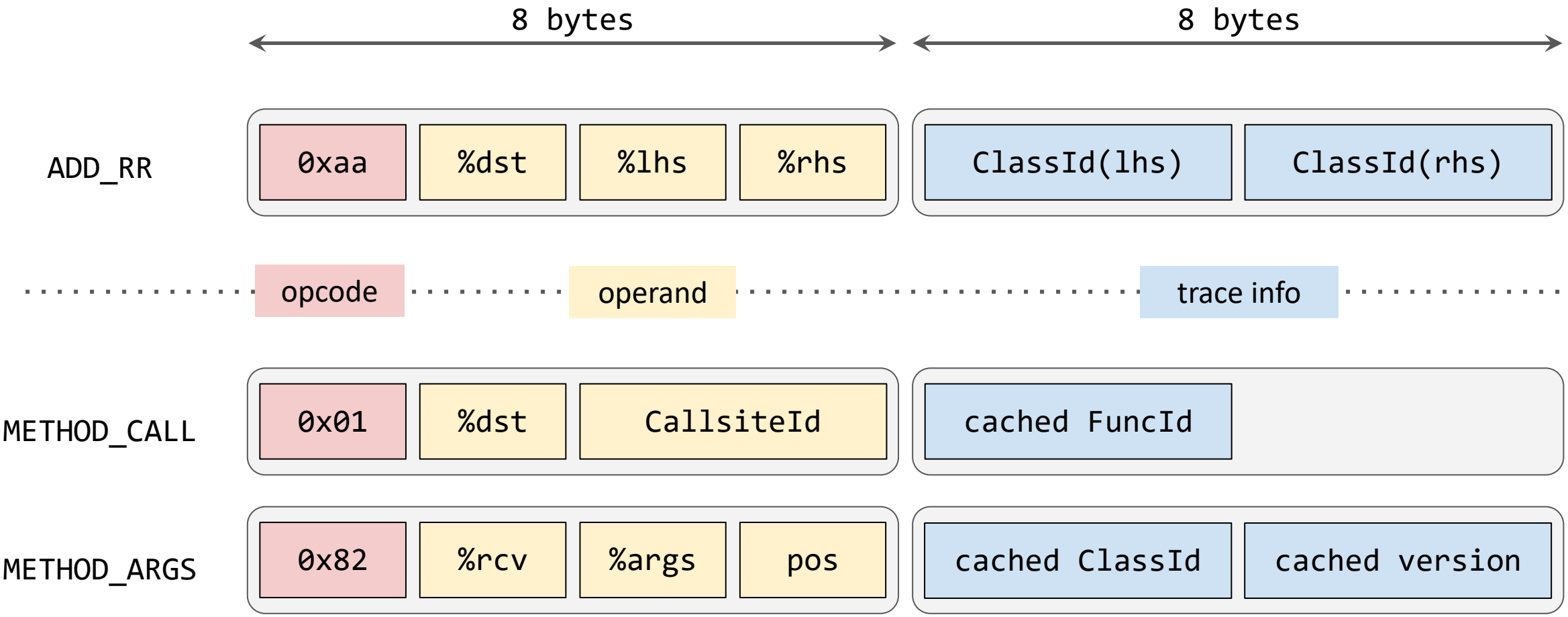
f: flags (frozen, etc)
ty: payload type



in-memory representation of Ruby objects



Bytecode (Virtual Machine instruction)



Summary: Interpreter

- A register machine VM.
- Collect and stores trace info in the bytecode.
- Optimizations

  global method cache, inline method cache, inline constant cache

  Embedding elements in RVALUE (Array, String)

  index access for instance variables
- Written in Assembly.

Summary: JIT compiler

- Method-based JIT compiler

  supports compilation of methods and loops.

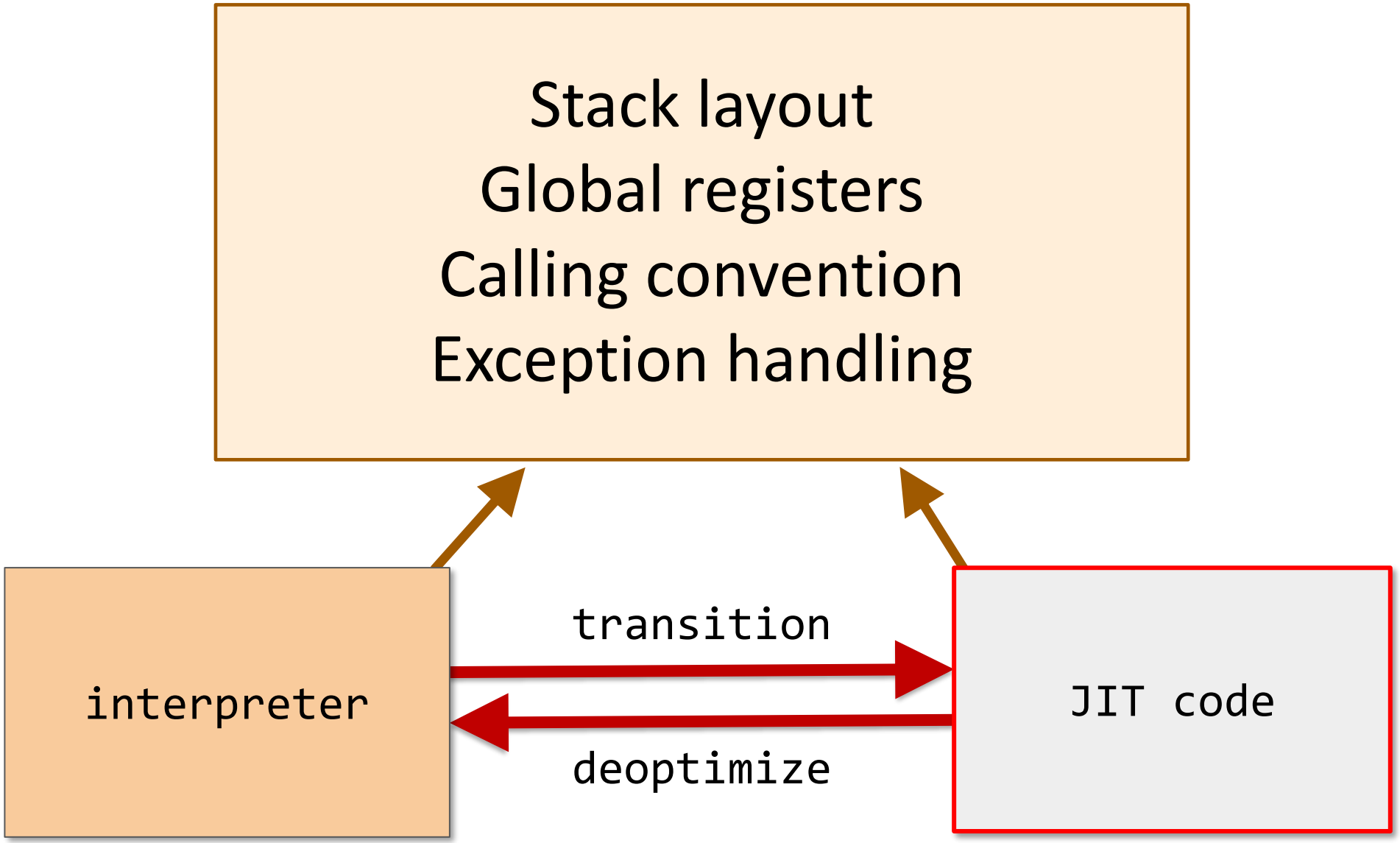
- Use trace info in the bytecode.

- Track class info of the registers and utilize for optimization.

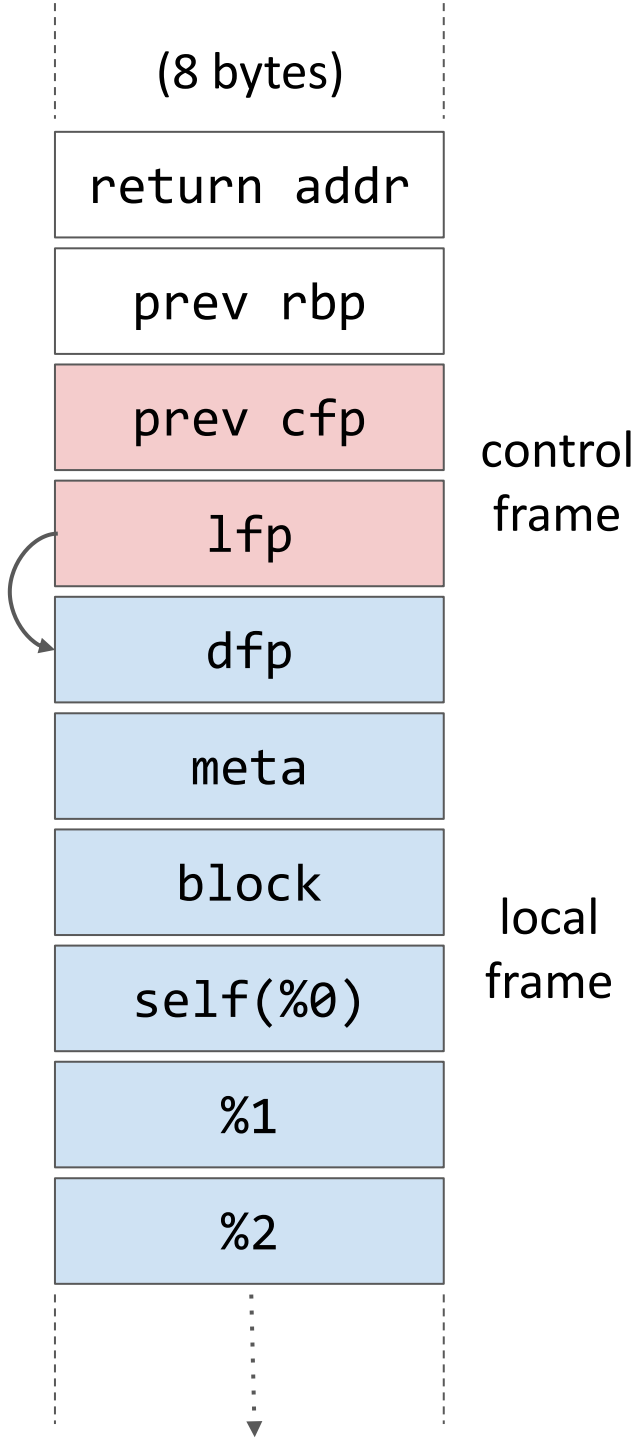
  reduce memory access

  omit unnecessary guards

  reduce Float <-> f64 conversion


Stack layout
Global registers
Calling convention
Exception handling
interpreter
transition
deoptimize
JIT code

stack frame structure

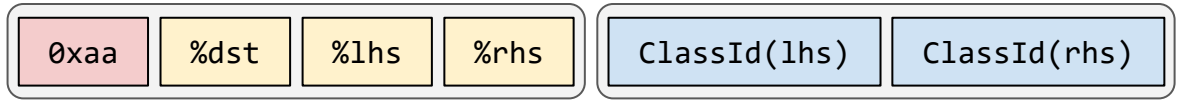



```
cfp: control frame pointer
prev cfp: link to the control frame of the caller
lfp: local frame pointer
dfp: link to the lfp of the outer scope
meta: metadata of this method/block
      function ID,
      the number of the registers,
      other flags
block: the given block
```

a + b : Interpreter

```
movsx  rsi,WORD PTR [r13-0x10]
movzx  rdi,WORD PTR [r13-0xe]
movzx  r15,WORD PTR [r13-0xc]
neg    rdi
mov    rdi,QWORD PTR [r14+rdi*8-0x30]
neg    rsi
mov    rsi,QWORD PTR [r14+rsi*8-0x30]
neg    r15
lea    r15,[r14+r15*8-0x30]
test   rdi,0x1
je     slow_path
test   rsi,0x1
je     slow_path
mov    DWORD PTR [r13-0x8],0x6
mov    DWORD PTR [r13-0x4],0x6
mov    rax,rdi
sub    al,0x1
add    rax,rsi
jo     slow_path
mov    QWORD PTR [r15],rax
movabs r15,0x561fe2169000
movzx  rax,BYTE PTR [r13+0x6]
add    r13,0x10
jmp    QWORD PTR [r15+rax*8]
```

ADD_RR



1) load objects from stack

2) guard for Fixnum
   goto slow_path if not Fixnum

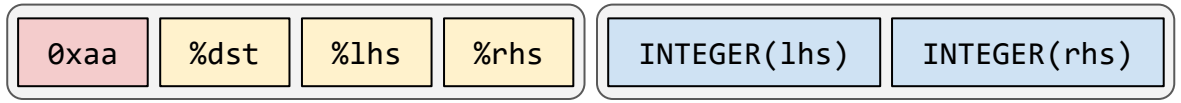
3) store trace info in bytecode

4) execute and store
   goto slow_path if overflow

5) fetch & dispatch

a + b : JIT compiled code

ADD_RR

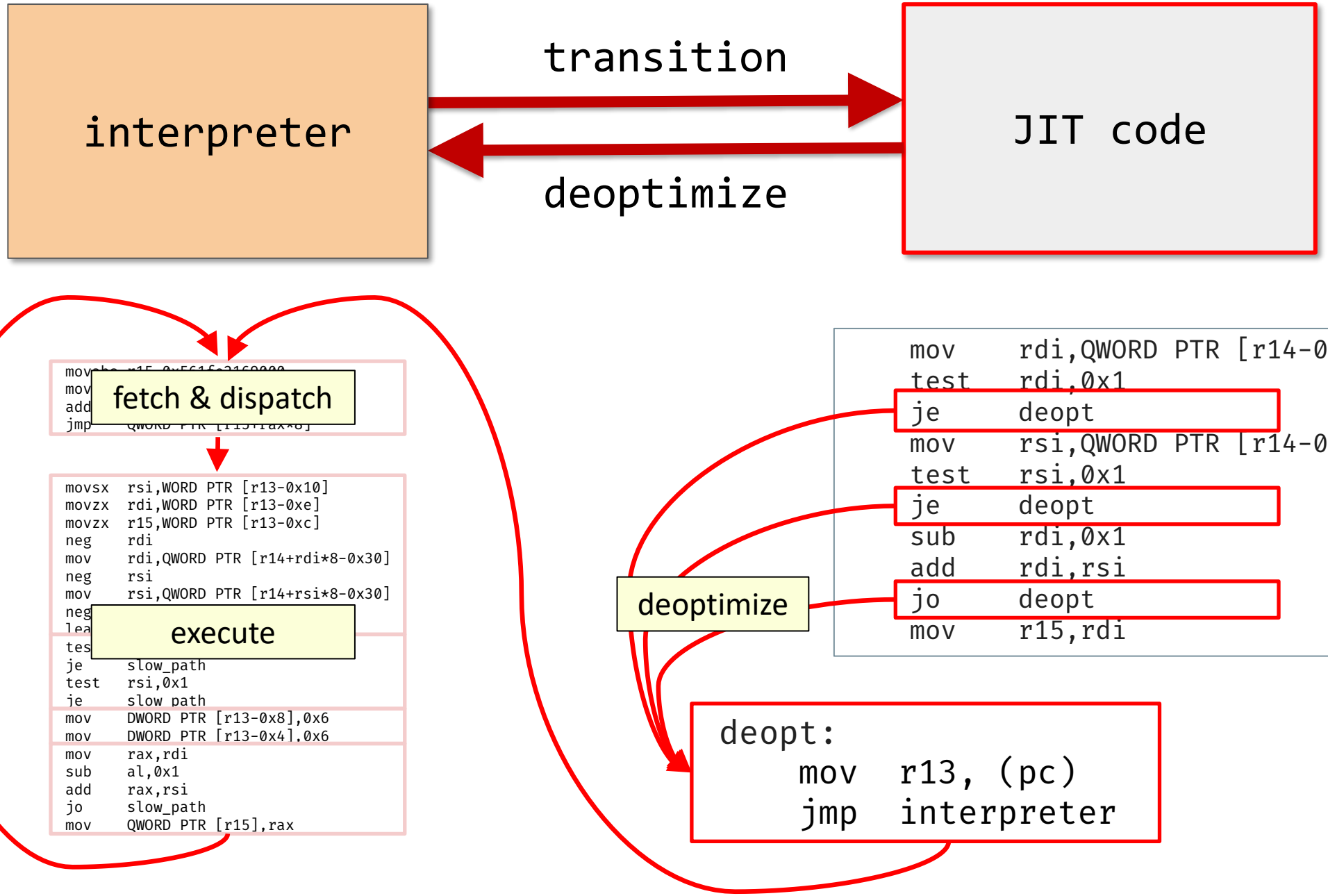



```
  mov     rdi,QWORD PTR [r14-0x38]
  test    rdi,0x1
  je      deopt
  mov     rsi,QWORD PTR [r14-0x40]
  test    rsi,0x1
  je      deopt
  sub     rdi,0x1
  add     rdi,rsi
  jo      deopt
  mov     r15,rdi
```

1) load and guard for Fixnum
   deoptimize if not Fixnum

2) execute and store
   deoptimize if overflow

```
deopt:
    mov     r13, (pc)
    jmp     interpreter
```


interpreter
transition
deoptimize
JIT code
fetch & dispatch
movsx rsi,WORD PTR [r13-0x10]
movzx rdi,WORD PTR [r13-0xe]
movzx r15,WORD PTR [r13-0xc]
neg rdi
mov rdi,QWORD PTR [r14+rdi*8-0x30]
neg rsi
mov rsi,QWORD PTR [r14+rsi*8-0x30]
execute
je slow_path
test rsi,0x1
je slow_path
mov DWORD PTR [r13-0x8],0x6
mov DWORD PTR [r13-0x4],0x6
mov rax,rdi
sub al,0x1
add rax,rsi
jo slow_path
mov QWORD PTR [r15],rax
deoptimize
mov rdi,QWORD PTR [r14-0x38]
test rdi,0x1
je deopt
mov rsi,QWORD PTR [r14-0x40]
test rsi,0x1
je deopt
sub rdi,0x1
add rdi,rsi
jo deopt
mov r15,rdi
deopt:
mov r13, (pc)
jmp interpreter

Wait a minute…
Is 1 + 1 always 2 in Ruby?

```ruby
class Integer
  def +(other)
    42
  end
end
```

Disaster control

The problems are:

- The interpreter was generated on the assumption that “1+1=2”.
- The JIT compiler have been generated codes on the assumption that “1+1=2”.

So, we must:

- Generate a new interpreter with no “1+1=2” assumption.
- Invalidate all generated JIT codes so far.
- Prohibit JIT compilation from now on.
- AND we must invalidate JIT codes on the call stack.

Invalidate JIT codes on the call stack

```
:00001 %1 = %0.a()
mov     rdi,QWORD PTR [r14-0x30]
mov     eax,DWORD PTR [rip+0x1fff65c6]
cmp     DWORD PTR [rip+0x1fff6408],eax
jne     0xfff7458
mov     r13,rdi
cmp     DWORD PTR [rip+0x1fff6410],0x0
jne     0xfff74b8
sub     rsp,0x20
xor     rax,rax
push    rax
movabs  rax,0x10000002000001af
push    rax
xor     rax,rax
push    rax
push    r13
add     rsp,0x40
lea     r14,[rsp-0x10]
mov     QWORD PTR [r14-0x10],r14
mov     rdi,QWORD PTR [rbx]
lea     rsi,[rsp-0x18]
mov     QWORD PTR [rsi],rdi
mov     QWORD PTR [rbx],rsi
movabs  r13,0x5632ca8ece50
call    0xffffff68
lea     r14,[rbp-0x8]
mov     QWORD PTR [rbx],r14
mov     r14,QWORD PTR [rbp-0x10]
test    rax,rax
je      0xfff7449
mov     r15,rax
jmp   deopt
sub     DWORD PTR [rip+0x1fff67a5],0x1
jmp     0xfff7802
```

1) guards for IMC

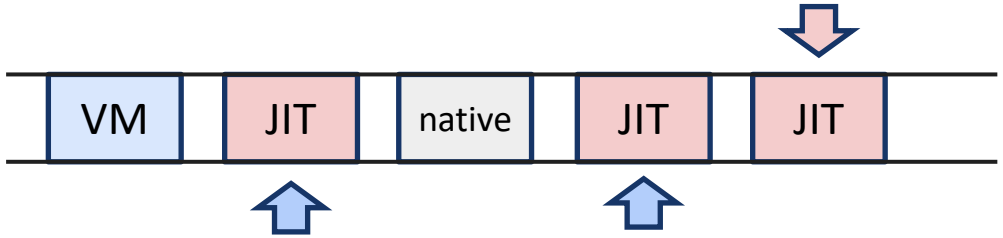
2) push frame

3) call

4) pop frame

5) check exception

call stack

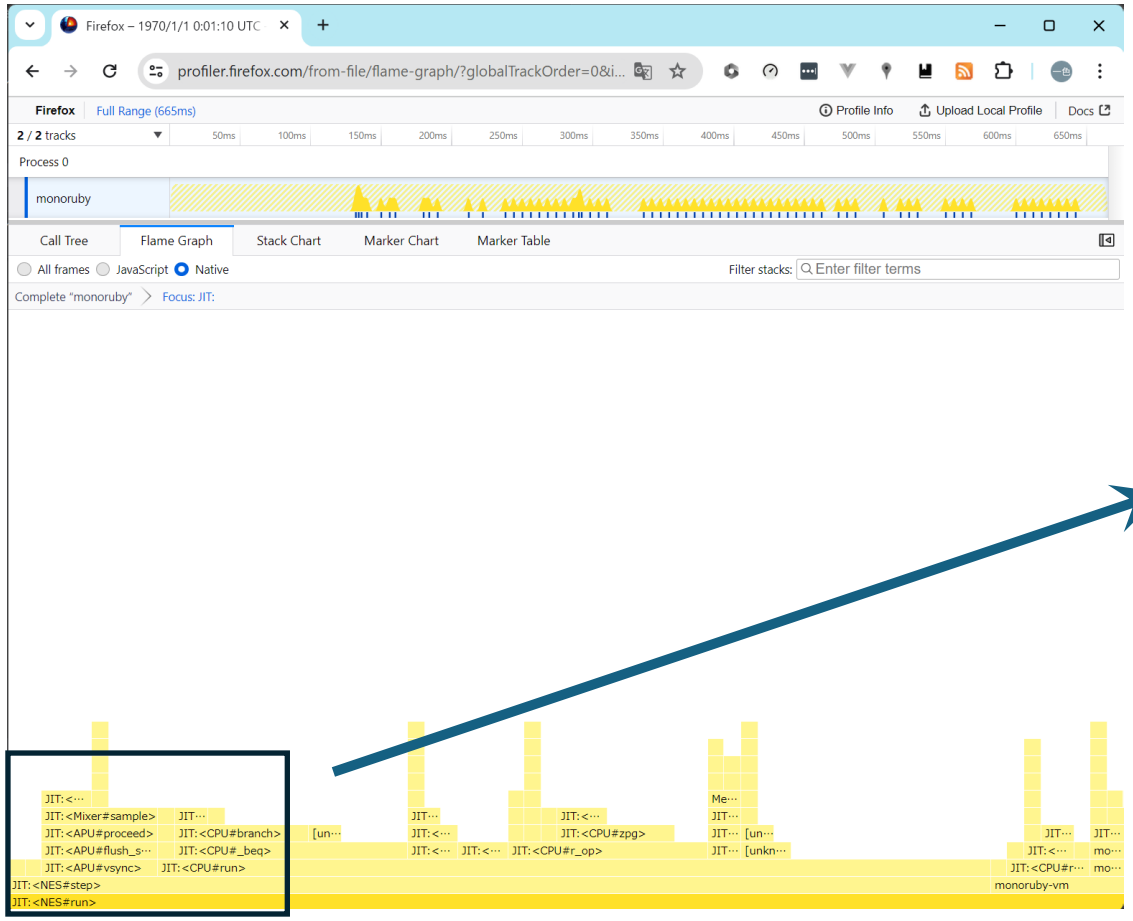


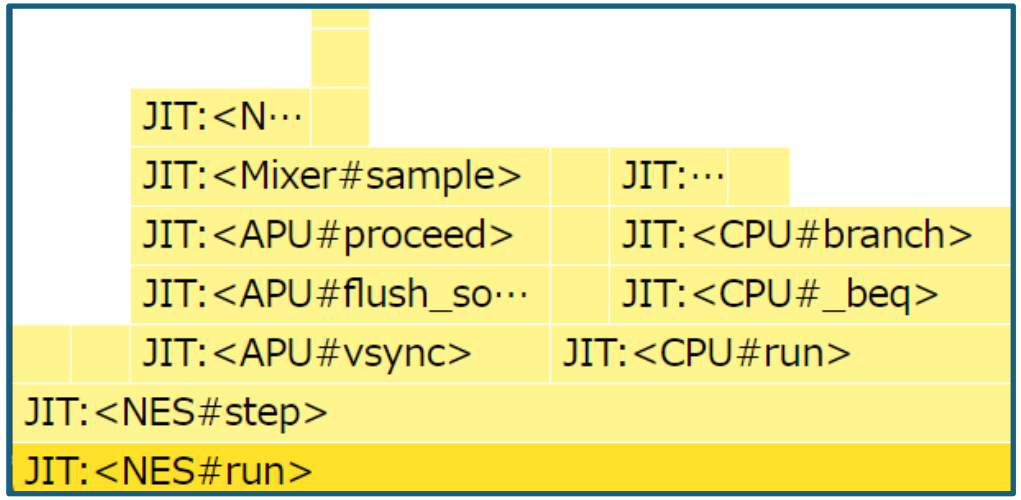

```
deopt:
    mov   r13, (pc)
    jmp   interpreter
```

Benchmarking

- Tool
  - yjit-bench: https://github.com/Shopify/yjit-bench 
  - optcarrot: https://github.com/mame/optcarrot 
- Target
  - CRuby 3.3.1 (±YJIT)
  - TruffleRuby (truffleruby 24.0.1, GraalVM JVM/Native)
  - monoruby

perf


Firefox – 1970/1/1 0:01:10 UTC
profiler.firefox.com/from-file/flame-graph/?globalTrackOrder=0&i...
Firefox
Full Range (665ms)
Profile Info
Upload Local Profile
Docs
2 / 2 tracks
50ms
100ms
150ms
200ms
250ms
300ms
350ms
400ms
450ms
500ms
550ms
600ms
650ms
Process 0
monoruby
Call Tree
Flame Graph
Stack Chart
Marker Chart
Marker Table
All frames
JavaScript
Native
Filter stacks:
Enter filter terms
Complete "monoruby"
Focus: JIT:
JIT:<···
JIT:<Mixer#sample>
JIT:···
JIT:<APU#proceed>
JIT:<CPU#branch>
JIT:<APU#flush_s···
JIT:<CPU#_beq>
JIT:<APU#vsync>
JIT:<CPU#run>
JIT:<NES#step>
JIT:<NES#run>
JIT:<CPU#zpg>
JIT:<CPU#r_op>
monoruby-vm


JIT:<N···
JIT:<Mixer#sample>
JIT:···
JIT:<APU#proceed>
JIT:<CPU#branch>
JIT:<APU#flush_so···
JIT:<CPU#_beq>
JIT:<APU#vsync>
JIT:<CPU#run>
JIT:<NES#step>
JIT:<NES#run>

profiling

```
monoruby on  binding via 💎 v3.3.0 via 🦀 v1.78.0-nightly took 17s
❯ cargo run --features profile -- ../optcarrot/bin/optcarrot -b ../optcarrot/examples/Lan_Master.nes
   Compiling monoruby v0.3.0 (/home/monochrome/monoruby/monoruby)
    Finished dev [optimized + debuginfo] target(s) in 8.33s
     Running `target/debug/monoruby ../optcarrot/bin/optcarrot -b ../optcarrot/examples/Lan_Master.nes`
fps: 355.8979205684173
checksum: 59662

deoptimization stats (top 20)
func name                                FuncId [index]     count
-------------------------------------------------------------------------------------------------------
CPU#fetch                                   546 [00003]        9483   %2 = %2.[%1]                        [Method][Integer]
block in Parser#find_option                1032 [00001]         166   %2 = %1.to_s()                      [NilClass]
block in #<Class:Optcarrot::CPU>#op         675 [00003]          60   %6 = %6.is_a?(%7;1)                 [Array]
Oscillator#active?                          782 [00010]          18   %1 = @envelope: Optcarrot::APU::Triangle[IvarId(5)]
Noise#sample                                814 [00070]          12   %3 = @shifter: Optcarrot::APU::Noise[IvarId(11)]
PPU#main_loop                               914 [00028]          12   %3 = @sp_ram: Optcarrot::PPU[IvarId(53)]
CPU#add_mappings                            544 [00018]           9   %4 = %1.is_a?(%4;1)                 [Range]
Triangle#active?                            802 [00005]           6   %2 = @linear_counter: Optcarrot::APU::Triangle[IvarId(14)]
Pulse#active?                               793 [00005]           6   %1 = @valid_freq: Optcarrot::APU::Pulse[IvarId(10)]
block in PPU#main_loop                      931 [00031]           6   %4 = @hclk: Optcarrot::PPU[IvarId(11)]
block in PPU#main_loop                      930 [00003]           6   %1 = @hclk: Optcarrot::PPU[IvarId(11)]
APU#do_clock                                742 [00012]           6   %1 = @cpu: Optcarrot::APU[IvarId(1)]
APU#clock_dmc                               749 [00006]           6   %2 = @dmc_clock: Optcarrot::APU[IvarId(18)]
Pulse#sample                                799 [00008]           6   %4 = @timer: Optcarrot::APU::Pulse[IvarId(6)]
CPU#do_clock                                668 [00007]           6   %1 = @clk_frame: Optcarrot::CPU[IvarId(7)]
Triangle#sample                             808 [00011]           6   %3 = const[WAVE_FORM]               [[0, 1, 2, 3 .. ]]
Noise#sample                                814 [00007]           6   %3 = @timer: Optcarrot::APU::Noise[IvarId(6)]
PPU#wait_one_clock                          912 [00008]           6   %1 = const[Fiber]                   [Fiber]
Triangle#sample                             808 [00062]           6   %3 = @amp: Optcarrot::APU::Triangle[IvarId(8)]
PPU#sync                                    854 [00007]           6   %2 = const[RP2C02_CC]               [4]
```

deoptimize analysis

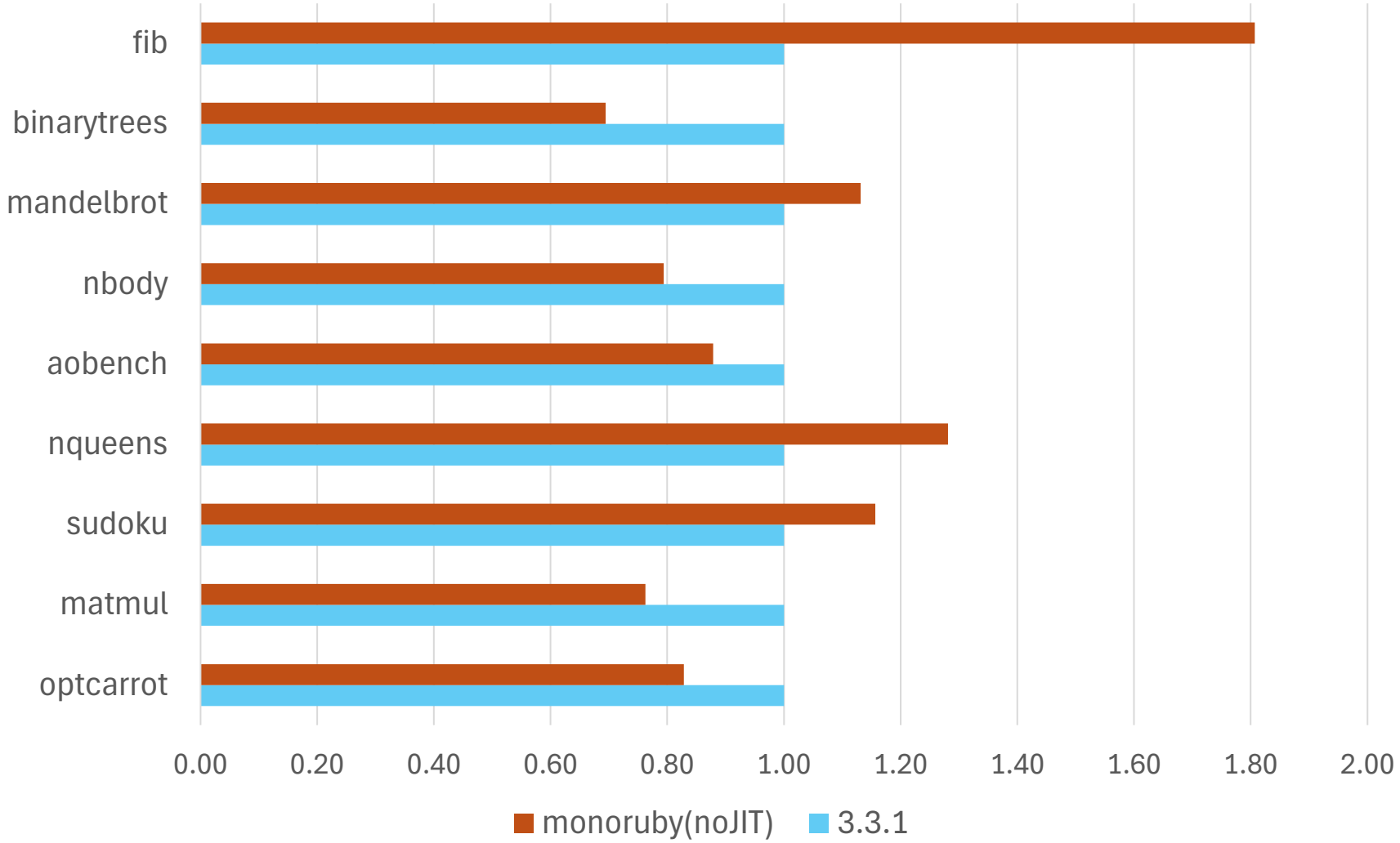
```
❯ cargo run --features deopt -- .quine/CML_quine.rb
   Compiling monoruby v0.3.0 (/home/monochrome/monoruby/monoruby)
    Finished dev [optimized + debuginfo] target(s) in 4.30s
     Running `target/debug/monoruby .quine/CML_quine.rb`
==> start whole compile: FuncId(439) <block in /main> self_class: #<Class:main> (eval):1
    total bytes(0):41526
    total bytes(1):4479
<== finished compile. elapsed:11.751µs
<-- deopt occurs in <block in /main> FuncId(440).    [00017] %4 = const[M]     [[[], [], [], [] .. ]]
<-- deopt occurs in <block in /main> FuncId(440).    [00017] %4 = const[M]     [[[], [], [], [] .. ]]
<-- deopt occurs in <block in /main> FuncId(440).    [00017] %4 = const[M]     [[[], [], [], [] .. ]]
<-- deopt occurs in <block in /main> FuncId(440).    [00017] %4 = const[M]     [[[], [], [], [] .. ]]
<-- non-traced branch in <block in /main> FuncId(450).    [00032] %6 = %1 * 2: i16    [<INVALID>][<INVALID>]
<-- non-traced branch in <block in /main> FuncId(450).    [00032] %6 = %1 * 2: i16    [Integer][Integer]
<-- non-traced branch in <block in /main> FuncId(450).    [00032] %6 = %1 * 2: i16    [Integer][Integer]
<-- non-traced branch in <block in /main> FuncId(450).    [00032] %6 = %1 * 2: i16    [Integer][Integer]
<-- non-traced branch in <block in /main> FuncId(450).    [00032] %6 = %1 * 2: i16    [Integer][Integer]
==> start whole compile: FuncId(450) <block in /main> self_class: #<Class:main> (eval):1
    total bytes(0):49432
    total bytes(1):20351
<== finished compile. elapsed:72.238µs
<-- deopt occurs in <block in /main> FuncId(451).    [00035] %7 = %7 - %8      [Float][Integer] caused by 0
<-- deopt occurs in <block in /main> FuncId(451).    [00035] %7 = %7 - %8      [Float][Integer] caused by 0
<-- deopt occurs in <block in /main> FuncId(451).    [00035] %7 = %7 - %8      [Float][Integer] caused by 0
<-- deopt occurs in <block in /main> FuncId(451).    [00035] %7 = %7 - %8      [Float][Integer] caused by 0

elapsed JIT compile time: 6.354501ms
```

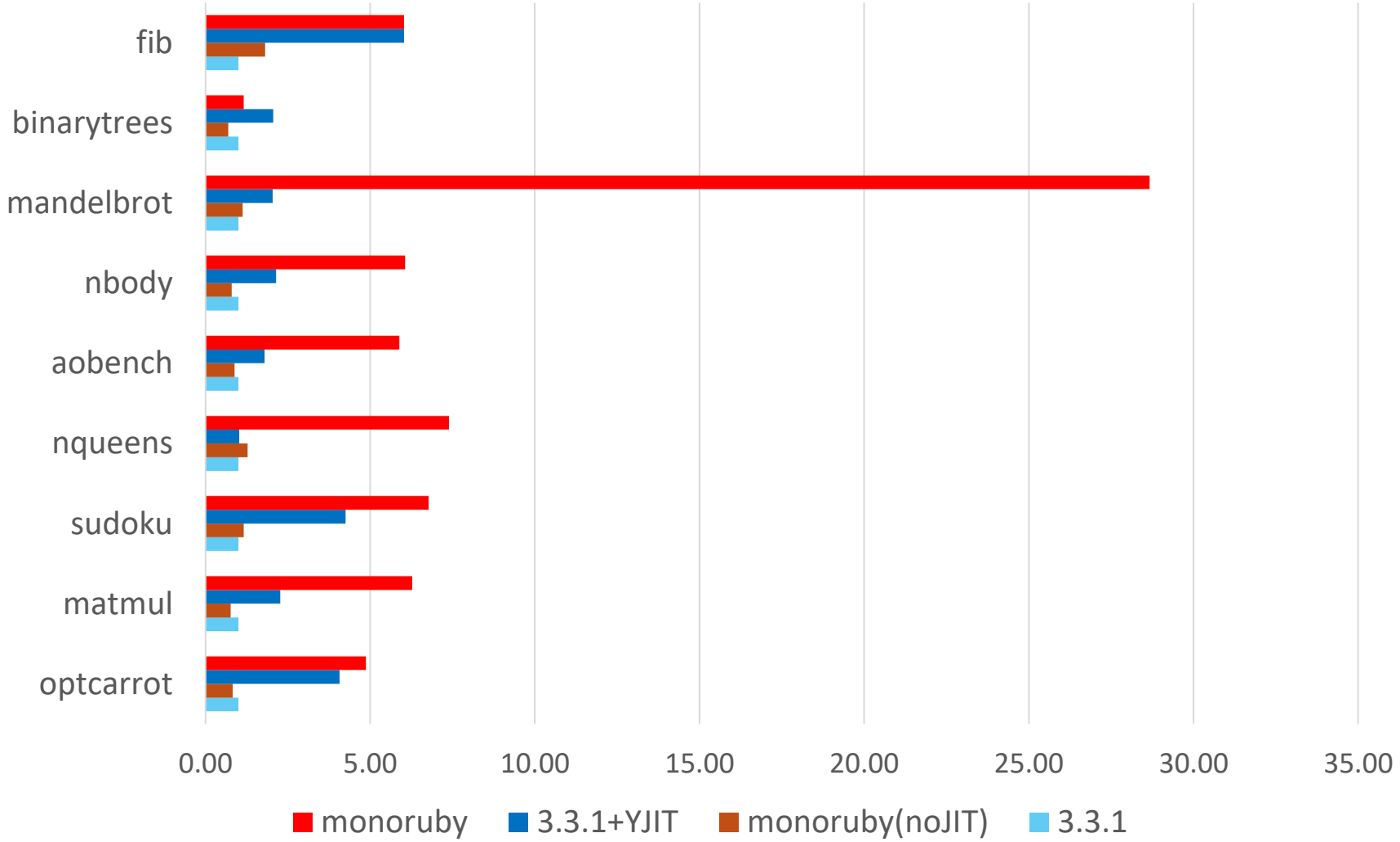
dump asm

```
❯ cargo run --features deopt -- .quine/CML_quine.rb
==> start whole compile: FuncId(439) <block in /main> self_class: #<Class:main> (eval):1
<== finished compile.
offset:Pos(41469) code: 57 bytes  data: 0 bytes
  00000: push   rbp
  00001: mov    rbp,rsp
  00004: sub    rsp,0x40
  00008: test   BYTE PTR [r14-0x19],0x80
  0000d: jne    0x1b
  00013: mov    QWORD PTR [r14-0x38],0x4
:00000 init_method reg:1 arg:0 req:0 opt:0 rest:false stack_offset:4
:00001 %1 = literal[[]]
  0001b: movabs rdi,0x7f5d47dbb580
  00025: movabs rax,0x56389278f240
  0002f: call   rax
  00031: mov    r15,rax
:00002 ret %1
  00034: mov    rax,r15
  00037: leave
  00038: ret
```

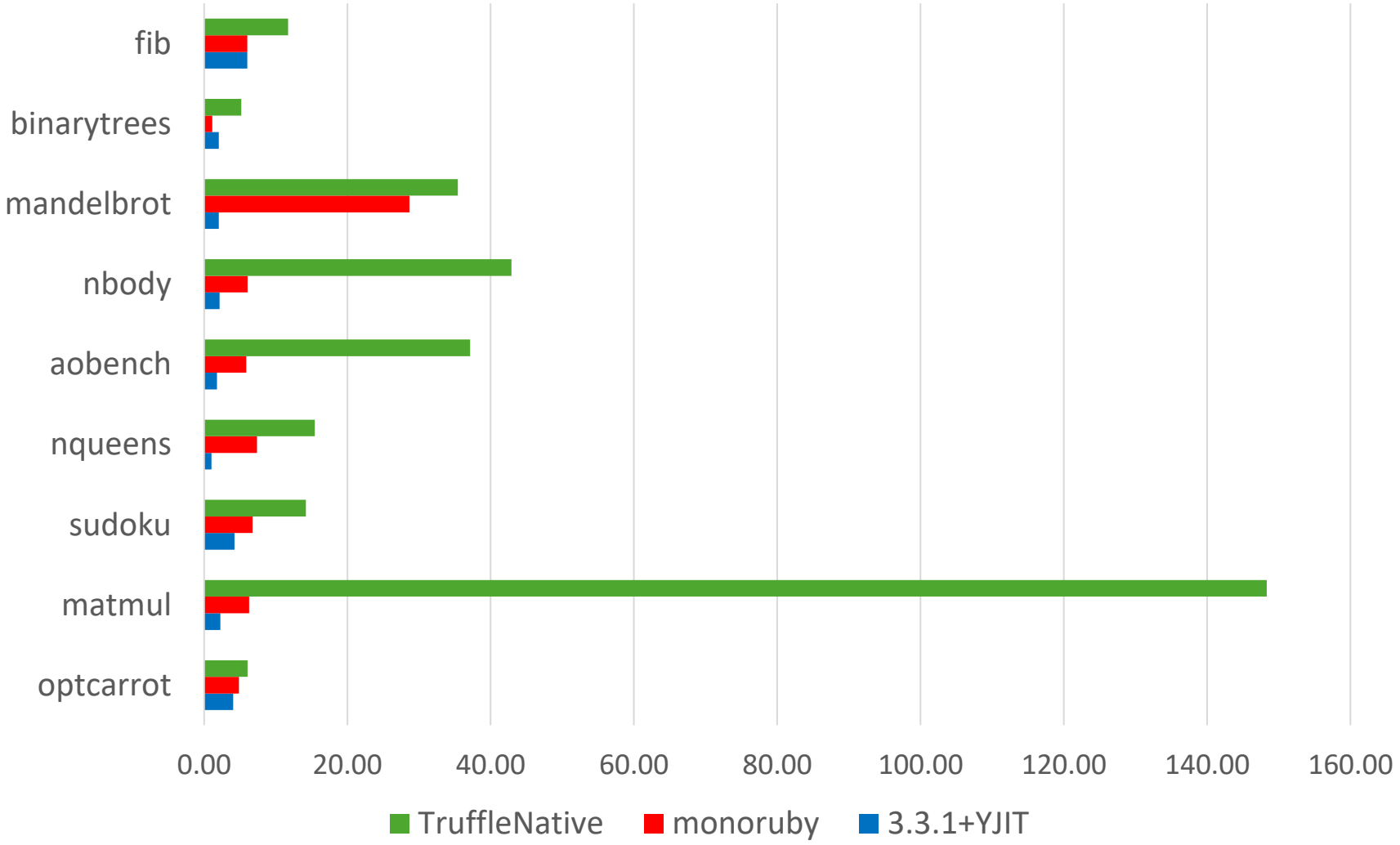
Benchmark results - interpreter (yjit-bench)



Benchmark results + JIT



Benchmark results + truffle

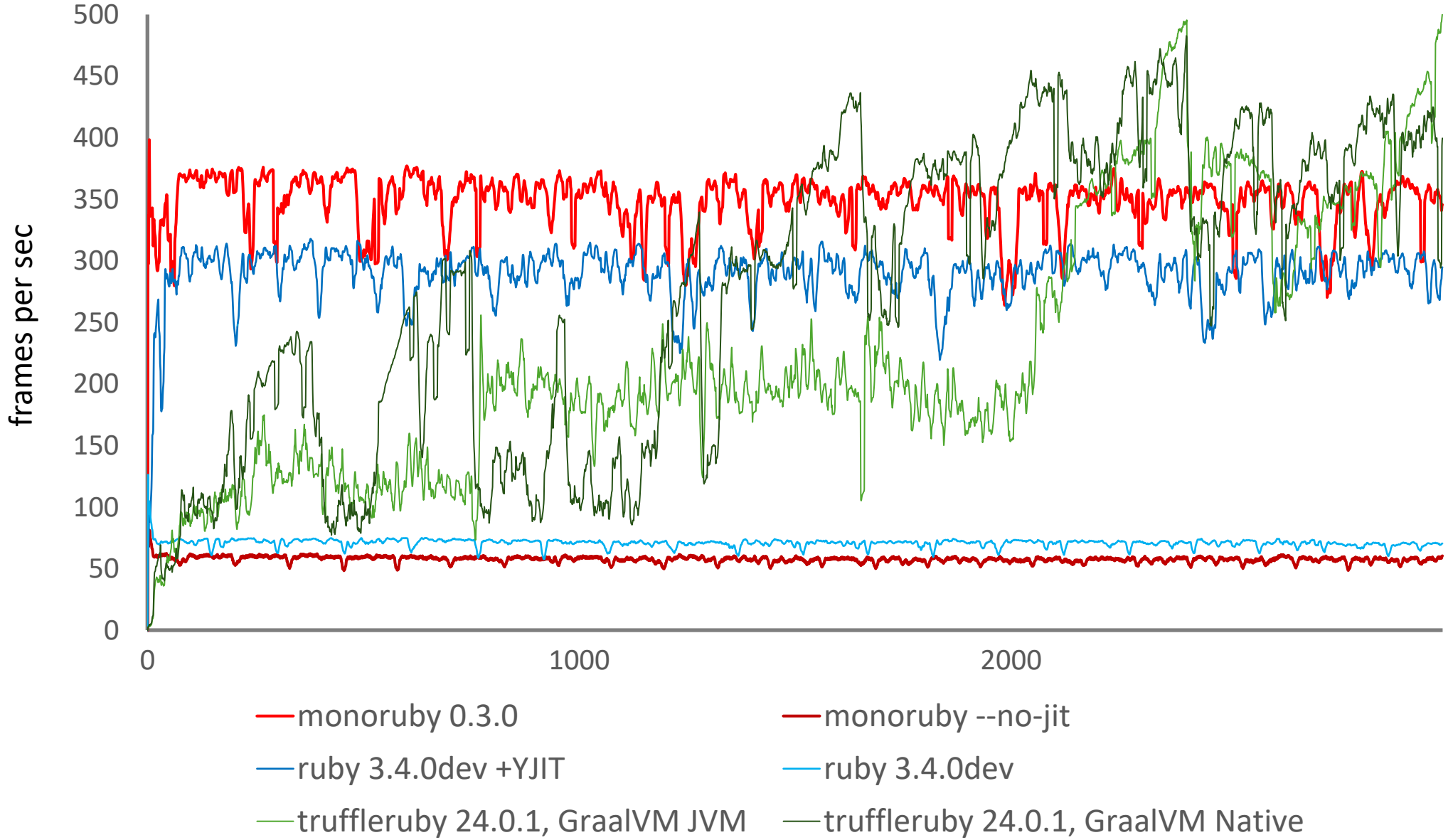


Memory footprint (RSS)

(MiB)

	3.3.1+YJIT	monoruby	TruffleRuby
fib	20.4	9.6	913.7
binarytrees	26.8	15.9	1233.3
mandelbrot	20.6	9.6	518.6
nbody	20.4	9.8	978.9
aobench	21.3	11.3	1515.7
sudoku	28.7	20.5	882.5
nqueens	20.8	11.2	1389.6
optcarrot	20.7	10.6	806.8
matmul	64.3	74.5	1483.7

Optcarrot benchmark fps history (up to 3000 frames)

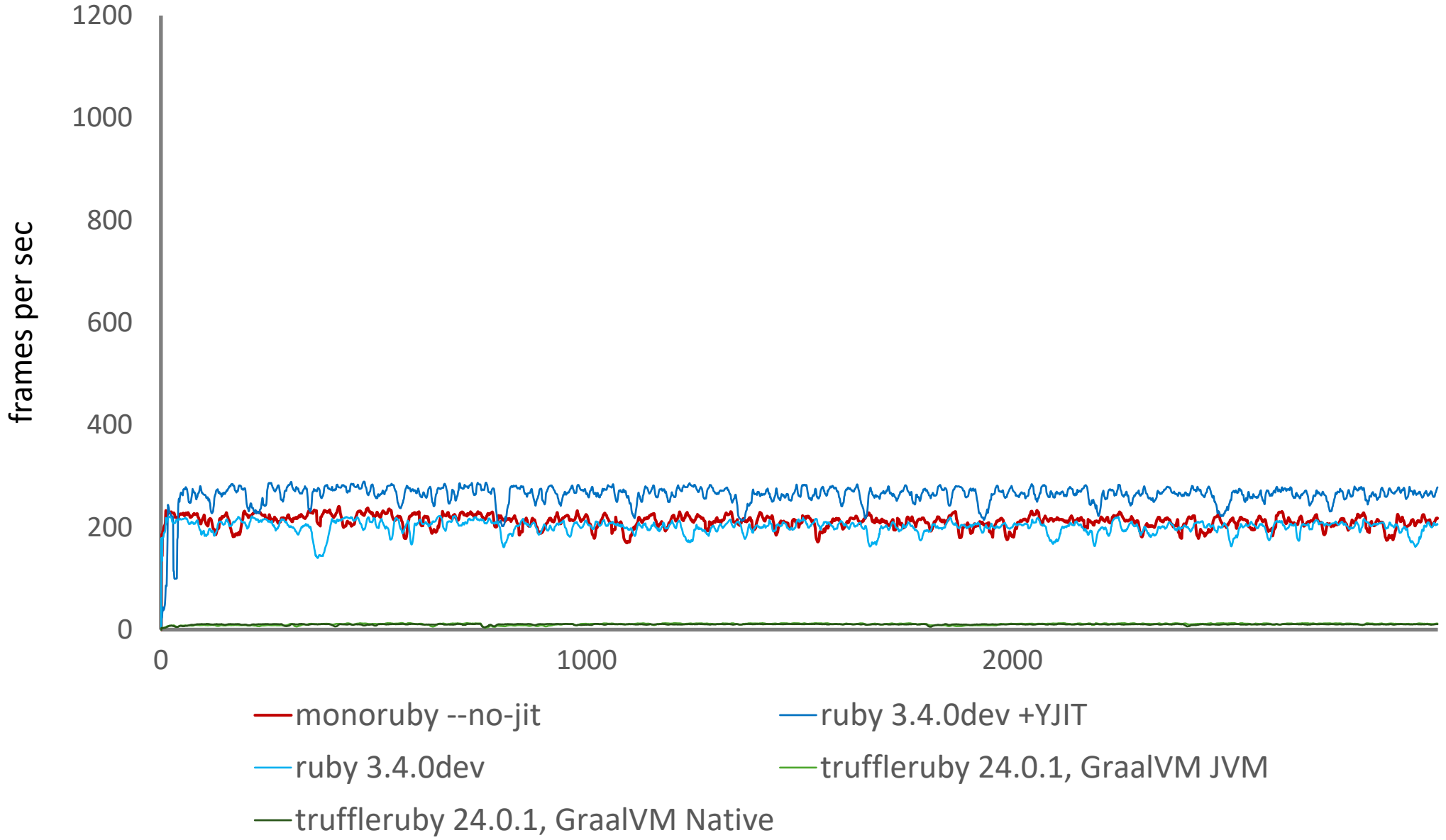


optcarrot –opt

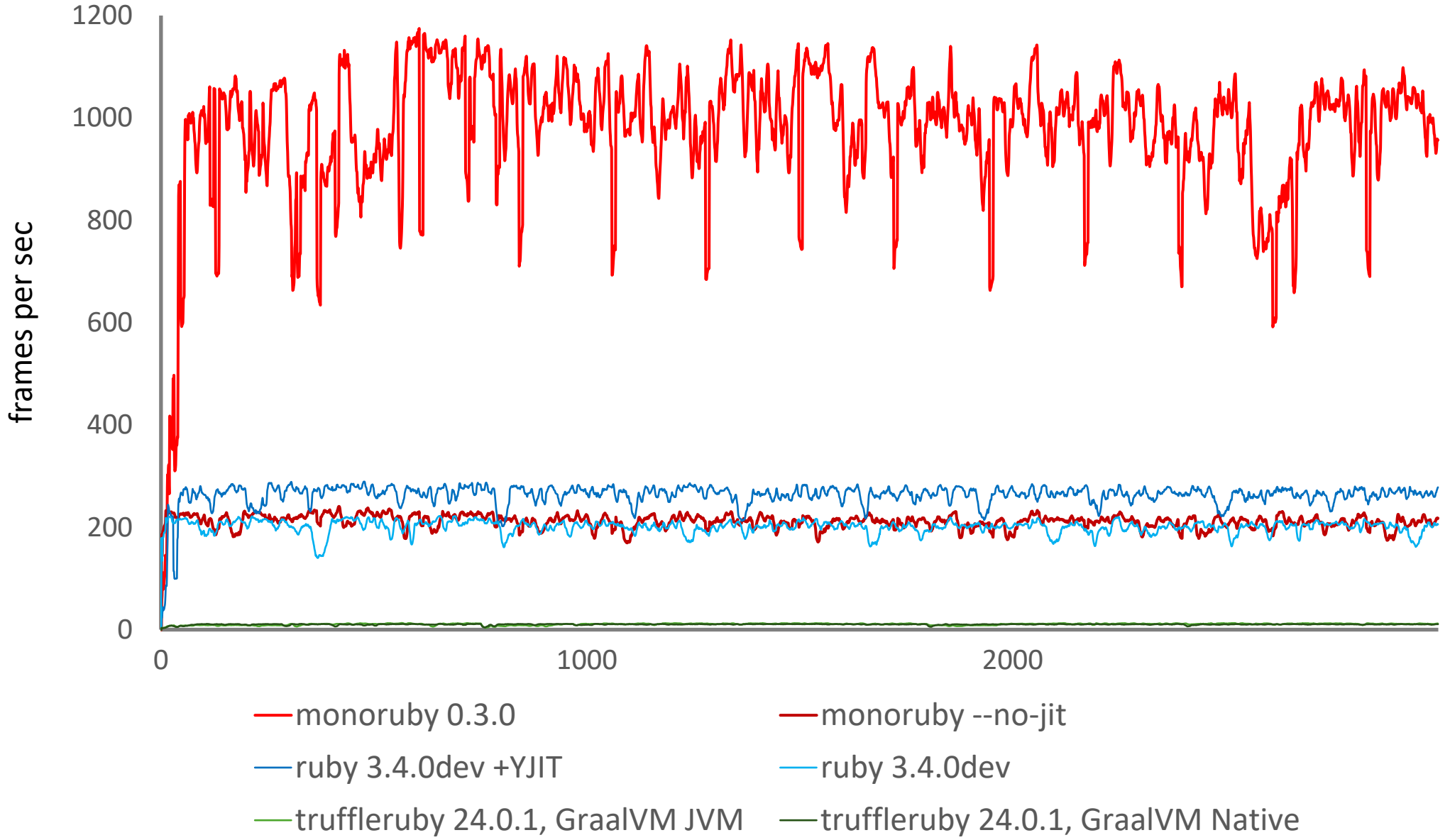
```
TRIVIAL_BRANCH_RE = /
  ^(¥ *)(if|unless)¥ (true|false)¥n
  ^((?:¥1¥ +.*¥n|¥n)*)
   (?:
     ¥1else¥n
     ((?:¥1¥ +.*¥n|¥n)*)
   )?
  ^¥1end¥n
/x
# remove "if true" or "if false"
def remove_trivial_branches(code)
  code = code.dup
  nil while
    code.gsub!(TRIVIAL_BRANCH_RE) do
      if ($2 == "if") == ($3 == "true")
        indent(-2, $4)
      else
        $5 ? indent(-2, $5) : ""
      end
    end
  code
end
```

Optcarrot: A pure-ruby NES emulator https://youtu.be/oD35gcGPGQc?si=BvJwiOUzvyt1LTt6

Optcarrot benchmark fps history with –opt (up to 3000 frames)



Optcarrot benchmark fps history with –opt (up to 3000 frames)



Run XXX on your own Ruby.

Special thanks:
@ko1, @k0kubun,@mametter
@yhara, @raviqqe