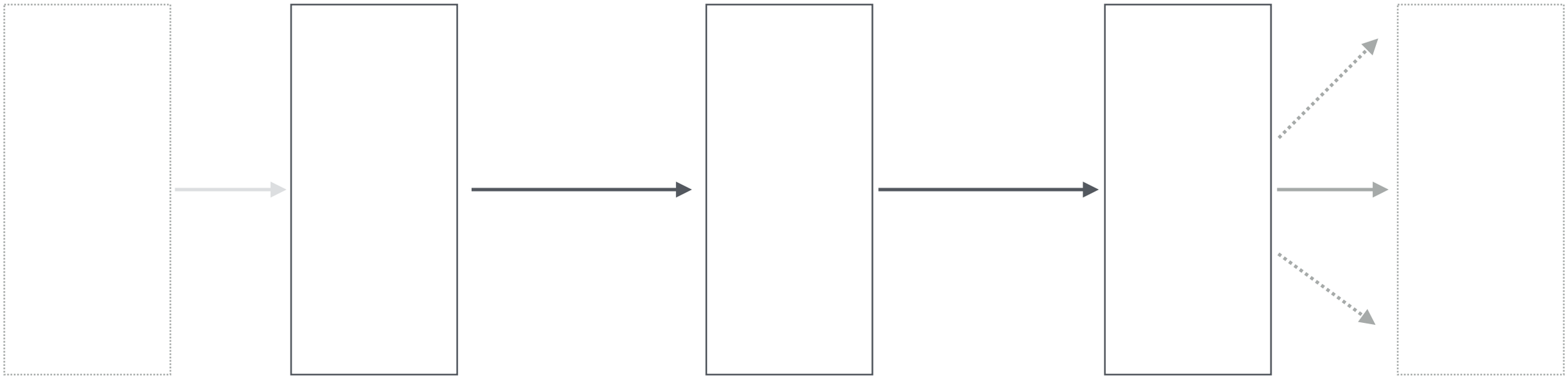
IR Deserts, Decompilation Swamps and radeco - Part II

- sushant94

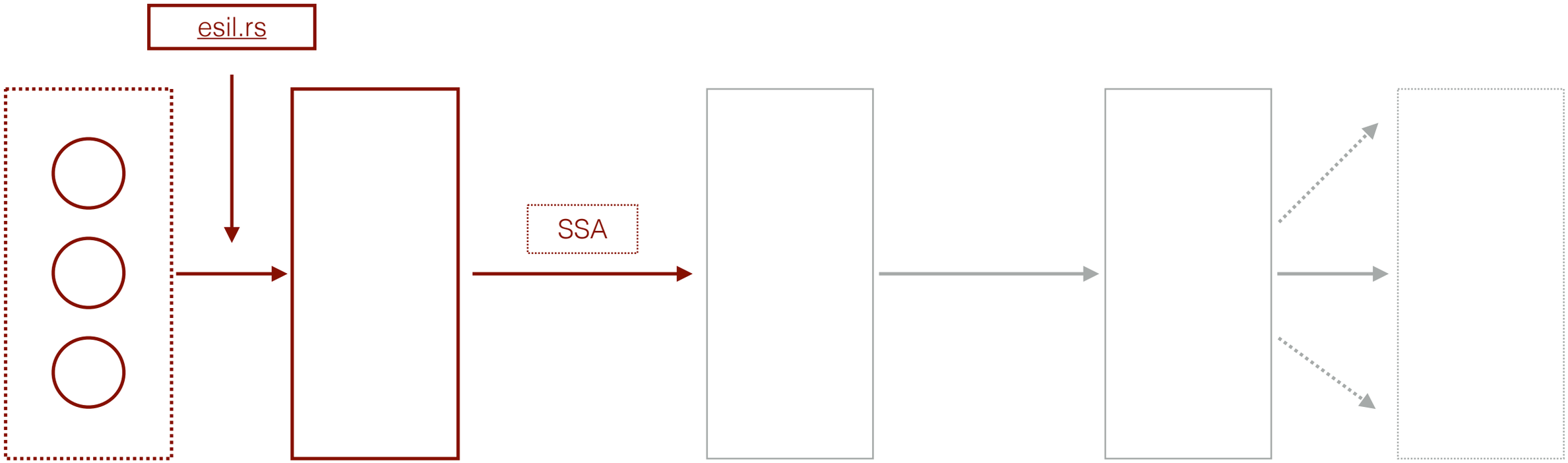
Design Overview

- radeco-lib + radeco (interactive tool)
- Written in rust
- Minimalistic and Generic
- Support multiple implementations
- Iterative and Interactive

3 Boxes …



Frontend



Source

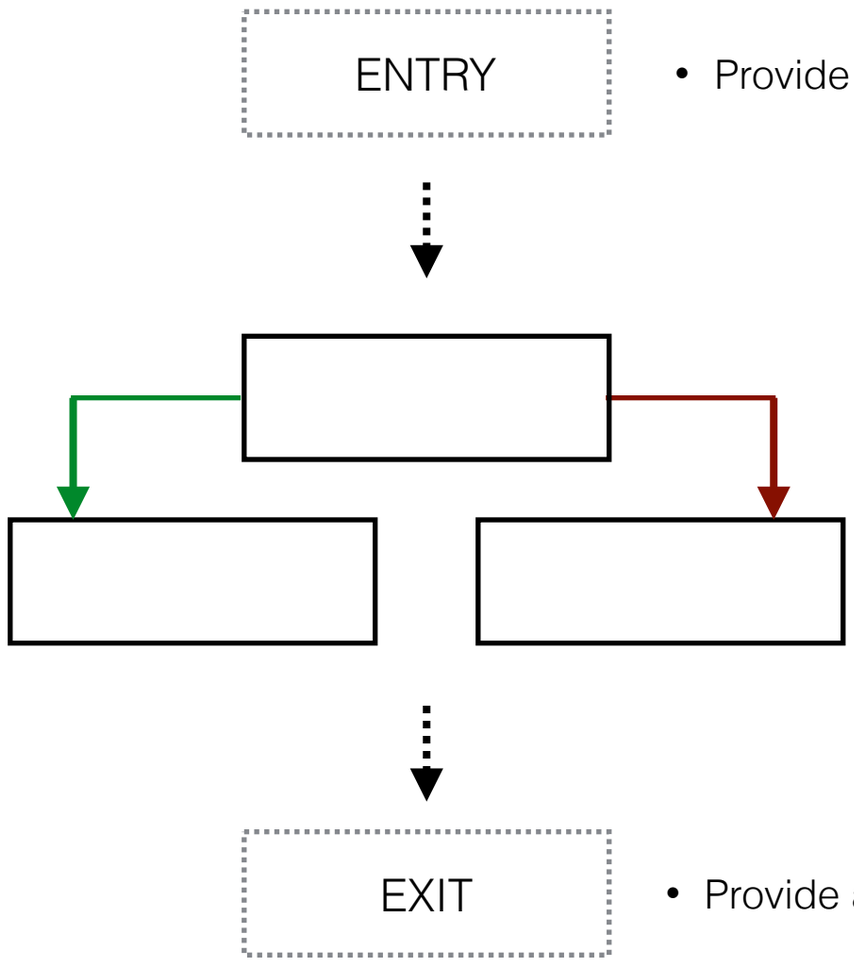
- radare2
- File
- Others

Frontend

- Structs and algorithms to convert source to SSA.
- Module and Function containers

radeco IR

radeco IR

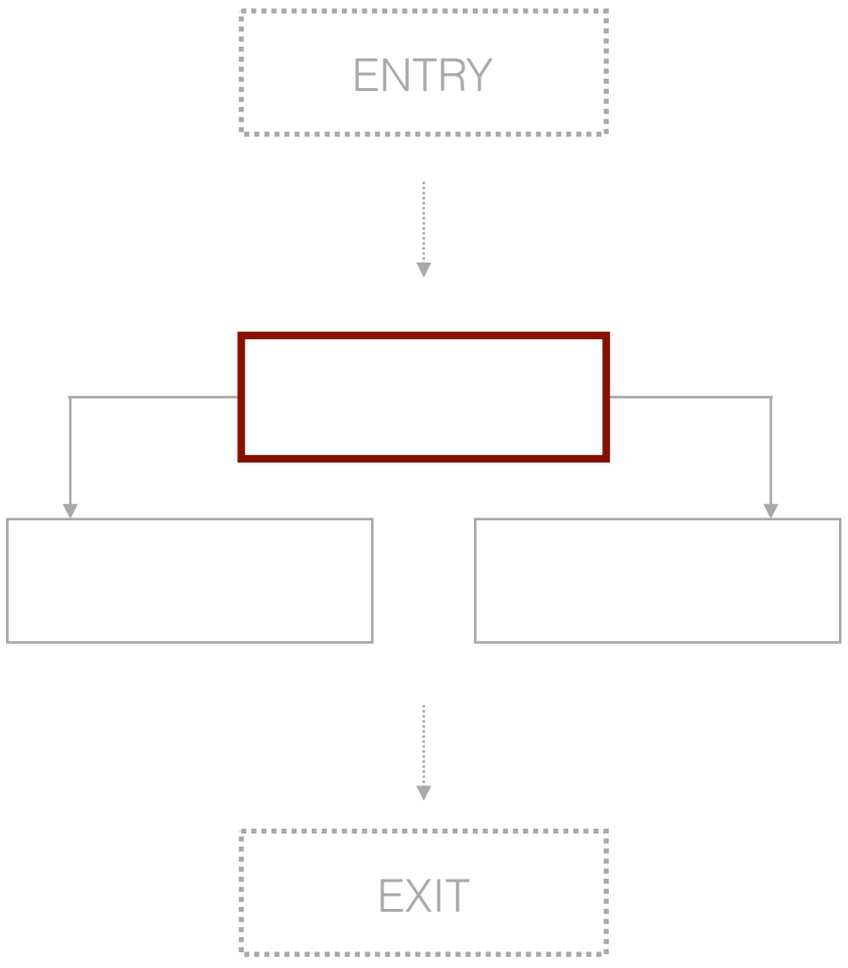


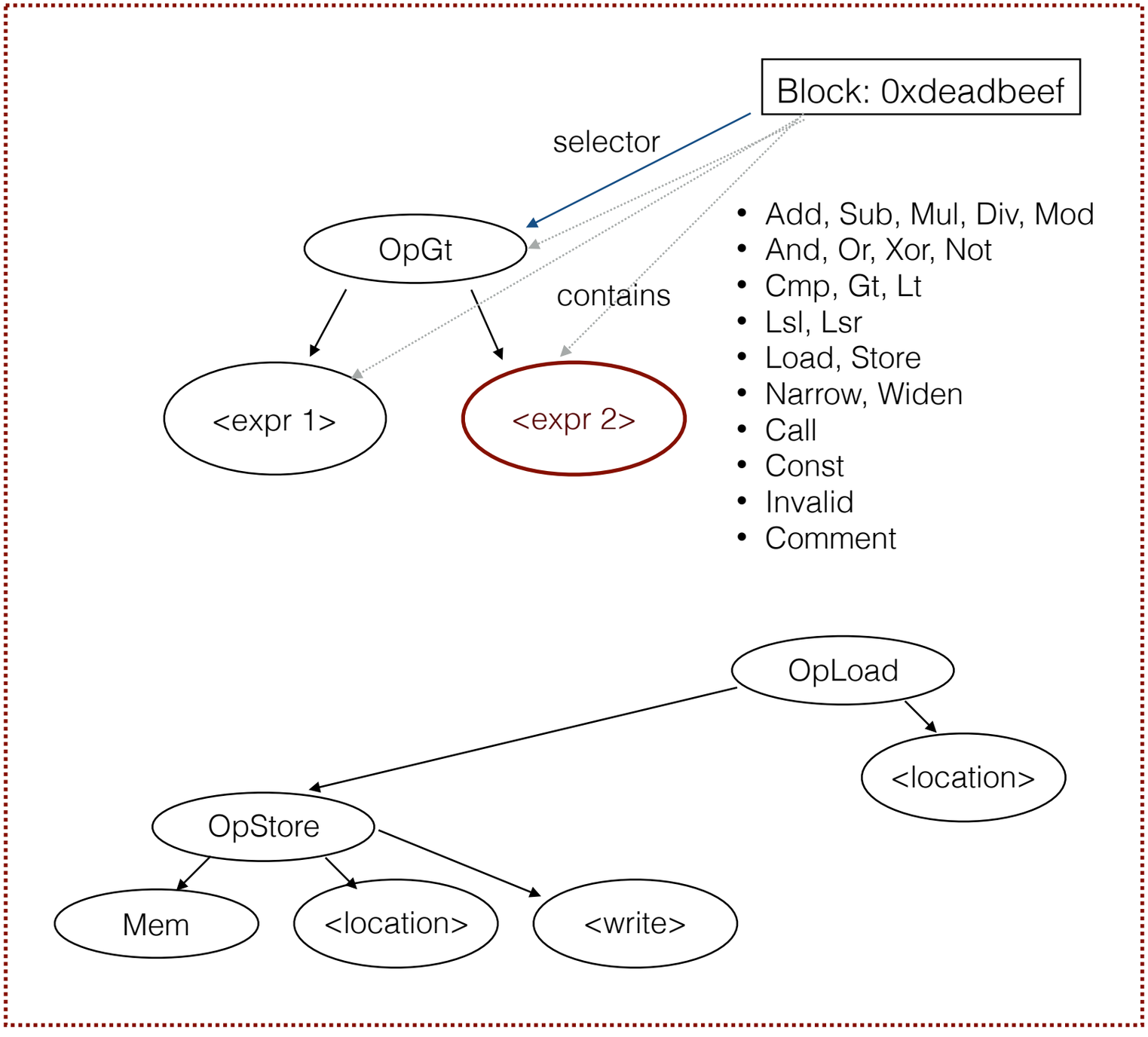
- Provide definitions to registers, flags and memory.

- Provide a sink to write registers, flags and memory.

Basic Control Flow Graph (CFG)

radeco IR





Static Single Assignment Graph (SSA Graph)

- Superposition of CFG and Expression nodes.

radeco IR - Text representation

```
define-fun main(unknown) -> unknown {
    bb_0x4004E6.0000():
        %1: $Unknown64 = rsp - #x8
        %2: $Unknown0 = Store(mem, %1, rbp)
        %3: $Unknown64 = %1 - #x10
        %4: $Unknown64 = %1 - #x8
        %5: $Unknown0 = Store(%2, %4, #xffffffffffffffff)
        %6: $Unknown64 = %1 - #xc
        %7: $Unknown0 = Store(%5, %6, #x0)
        %8: $Unknown64 = %1 - #x8
        %9: $Unknown64 = Load(%7, %8)

                <… snip …>

        %29: $Unknown1 = %28 | %27
        JMP IF %29 0x400526.0000
    bb_0x40050D.0000():
        %30: $Unknown64 = %1 - #xc
        %31: $Unknown0 = Store(%11, %30, #x0)
    bb_0x400520.0000():

                <… snip …>

        %57 = Phi(rbx, rbx@0x40051B.0007)

                <… snip …>

    bb_0x400516.0000():

                <… snip …>

        call 0x4003c0(["%45", "%49", "%44", "%58", "%43", …])

                <… snip …>
```

Functions are defined using “define-fun”

radeco IR - Text representation

```
define-fun main(unknown) -> unknown {
    bb_0x4004E6.0000():
        %1: $Unknown64 = rsp - #x8
        %2: $Unknown0 = Store(mem, %1, rbp)
        %3: $Unknown64 = %1 - #x10
        %4: $Unknown64 = %1 - #x8
        %5: $Unknown0 = Store(%2, %4, #xffffffffffffffff)
        %6: $Unknown64 = %1 - #xc
        %7: $Unknown0 = Store(%5, %6, #x0)
        %8: $Unknown64 = %1 - #x8
        %9: $Unknown64 = Load(%7, %8)

                <… snip …>

        %29: $Unknown1 = %28 | %27
        JMP IF %29 0x400526.0000
bb_0x40050D.0000():
        %30: $Unknown64 = %1 - #xc
        %31: $Unknown0 = Store(%11, %30, #x0)
bb_0x400520.0000():

                <… snip …>

        %57 = Phi(rbx, rbx@0x40051B.0007)

                <… snip …>

bb_0x400516.0000():

                <… snip …>

        call 0x4003c0(["%45", "%49", "%44", "%58", "%43", …])

                <… snip …>
```

Blocks are labelled as bb_<addr>

radeco IR - Text representation

```
define-fun main(unknown) -> unknown {
    bb_0x4004E6.0000():
        %1: $Unknown64 = rsp - #x8
        %2: $Unknown0 = Store(mem, %1, rbp)
        %3: $Unknown64 = %1 - #x10
        %4: $Unknown64 = %1 - #x8
        %5: $Unknown0 = Store(%2, %4, #xffffffffffffffff)
        %6: $Unknown64 = %1 - #xc
        %7: $Unknown0 = Store(%5, %6, #x0)
        %8: $Unknown64 = %1 - #x8
        %9: $Unknown64 = Load(%7, %8)

                <… snip …>

        %29: $Unknown1 = %28 | %27
        JMP IF %29 0x400526.0000
bb_0x40050D.0000():
        %30: $Unknown64 = %1 - #xc
        %31: $Unknown0 = Store(%11, %30, #x0)
bb_0x400520.0000():

                <… snip …>

        %57 = Phi(rbx, rbx@0x40051B.0007)

                <… snip …>

bb_0x400516.0000():

                <… snip …>

        call 0x4003c0(["%45", "%49", "%44", "%58", "%43", …])

                <… snip …>
```

Temporaries begin with ‘%’ and are labelled serially

radeco IR - Text representation

```
define-fun main(unknown) -> unknown {
    bb_0x4004E6.0000():
        %1: $Unknown64 = rsp - #x8
        %2: $Unknown0 = Store(mem, %1, rbp)
        %3: $Unknown64 = %1 - #x10
        %4: $Unknown64 = %1 - #x8
        %5: $Unknown0 = Store(%2, %4, #xffffffffffffffff)
        %6: $Unknown64 = %1 - #xc
        %7: $Unknown0 = Store(%5, %6, #x0)
        %8: $Unknown64 = %1 - #x8
        %9: $Unknown64 = Load(%7, %8)

                <… snip …>

        %29: $Unknown1 = %28 | %27
        JMP IF %29 0x400526.0000
bb_0x40050D.0000():
        %30: $Unknown64 = %1 - #xc
        %31: $Unknown0 = Store(%11, %30, #x0)
bb_0x400520.0000():

                <… snip …>

        %57 = Phi(rbx, rbx@0x40051B.0007)

                <… snip …>

bb_0x400516.0000():

                <… snip …>

        call 0x4003c0(["%45", "%49", "%44", "%58", "%43", …])

                <… snip …>
```

Constants begin with a “#” followed by:

- x - Hex
- d - Decimal
- o - Octal
- b - Binary

radeco IR - Text representation

```
define-fun main(unknown) -> unknown {
    bb_0x4004E6.0000():
        %1: $Unknown64 = rsp - #x8
        %2: $Unknown0 = Store(mem, %1, rbp)
        %3: $Unknown64 = %1 - #x10
        %4: $Unknown64 = %1 - #x8
        %5: $Unknown0 = Store(%2, %4, #xffffffffffffffff)
        %6: $Unknown64 = %1 - #xc
        %7: $Unknown0 = Store(%5, %6, #x0)
        %8: $Unknown64 = %1 - #x8
        %9: $Unknown64 = Load(%7, %8)

                <… snip …>

        %29: $Unknown1 = %28 | %27
        JMP IF %29 0x400526.0000
bb_0x40050D.0000():
        %30: $Unknown64 = %1 - #xc
        %31: $Unknown0 = Store(%11, %30, #x0)
bb_0x400520.0000():

                <… snip …>

        %57 = Phi(rbx, rbx@0x40051B.0007)

                <… snip …>

bb_0x400516.0000():

                <… snip …>

        call 0x4003c0(["%45", "%49", "%44", "%58", "%43", …])

                <… snip …>
```

Types begin with a “$”

radeco IR - Text representation

```
define-fun main(unknown) -> unknown {
    bb_0x4004E6.0000():
        %1: $Unknown64 = rsp - #x8
        %2: $Unknown0 = Store(mem, %1, rbp)
        %3: $Unknown64 = %1 - #x10
        %4: $Unknown64 = %1 - #x8
        %5: $Unknown0 = Store(%2, %4, #xffffffffffffffff)
        %6: $Unknown64 = %1 - #xc
        %7: $Unknown0 = Store(%5, %6, #x0)
        %8: $Unknown64 = %1 - #x8
        %9: $Unknown64 = Load(%7, %8)

                <… snip …>

        %29: $Unknown1 = %28 | %27
        JMP IF %29 0x400526.0000
bb_0x40050D.0000():
        %30: $Unknown64 = %1 - #xc
        %31: $Unknown0 = Store(%11, %30, #x0)
bb_0x400520.0000():

                <… snip …>

        %57 = Phi(rbx, rbx@0x40051B.0007)

                <… snip …>

bb_0x400516.0000():

                <… snip …>

        call 0x4003c0(["%45", "%49", "%44", "%58", "%43", …])

                <… snip …>
```

Store creates a new instance of memory

radeco IR - Text representation

```
define-fun main(unknown) -> unknown {
    bb_0x4004E6.0000():
        %1: $Unknown64 = rsp - #x8
        %2: $Unknown0 = Store(mem, %1, rbp)
        %3: $Unknown64 = %1 - #x10
        %4: $Unknown64 = %1 - #x8
        %5: $Unknown0 = Store(%2, %4, #xffffffffffffffff)
        %6: $Unknown64 = %1 - #xc
        %7: $Unknown0 = Store(%5, %6, #x0)
        %8: $Unknown64 = %1 - #x8
        %9: $Unknown64 = Load(%7, %8)

                <… snip …>

        %29: $Unknown1 = %28 | %27
    JMP IF %29 0x400526.0000
bb_0x40050D.0000():
    %30: $Unknown64 = %1 - #xc
    %31: $Unknown0 = Store(%11, %30, #x0)
bb_0x400520.0000():

            <… snip …>

    %57 = Phi(rbx, rbx@0x40051B.0007)

            <… snip …>

bb_0x400516.0000():

            <… snip …>

    call 0x4003c0(["%45", "%49", "%44", "%58", "%43", …])

            <… snip …>
```

JMP IF <condition> <target_true> else <target_false>

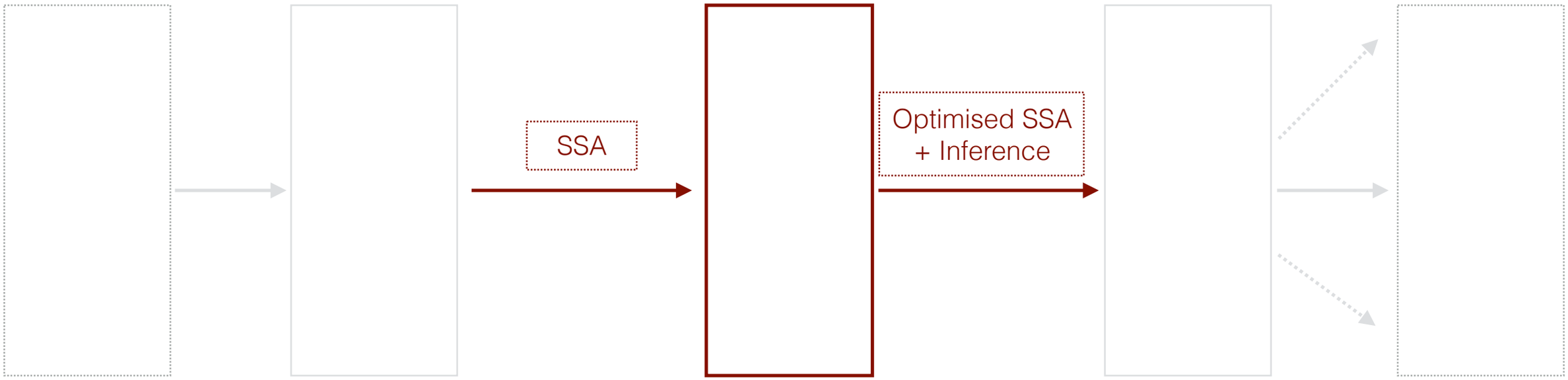
radeco IR - Text representation

```
define-fun main(unknown) -> unknown {
    bb_0x4004E6.0000():
        %1: $Unknown64 = rsp - #x8
        %2: $Unknown0 = Store(mem, %1, rbp)
        %3: $Unknown64 = %1 - #x10
        %4: $Unknown64 = %1 - #x8
        %5: $Unknown0 = Store(%2, %4, #xffffffffffffffff)
        %6: $Unknown64 = %1 - #xc
        %7: $Unknown0 = Store(%5, %6, #x0)
        %8: $Unknown64 = %1 - #x8
        %9: $Unknown64 = Load(%7, %8)

                <… snip …>

        %29: $Unknown1 = %28 | %27
        JMP IF %29 0x400526.0000
bb_0x40050D.0000():
        %30: $Unknown64 = %1 - #xc
        %31: $Unknown0 = Store(%11, %30, #x0)
bb_0x400520.0000():

                <… snip …>

        %57 = Phi(rbx, rbx@0x40051B.0007)

                <… snip …>

bb_0x400516.0000():

                <… snip …>

        call 0x4003c0(["%45", "%49", "%44", "%58", "%43", …])

                <… snip …>
```

Call passes every register and latest memory instance as arguments

Back to the Boxes!

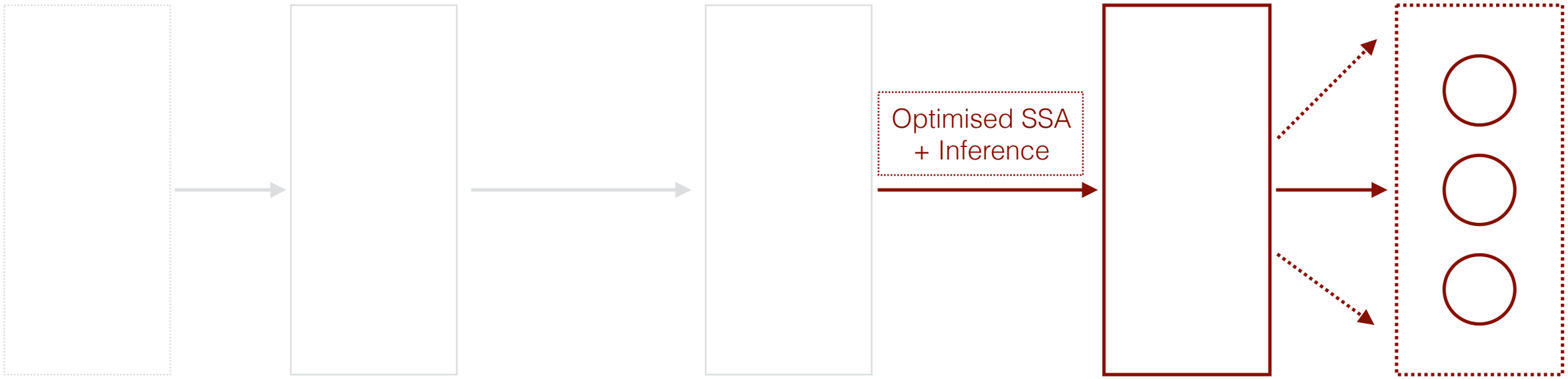
Analysis



Analysis

- Constant Propagation (SCCP)
- Dominator Tree (DOM)
- Constant Subexpression Elimination (CSE)
- Subgraph matching (Grep and Replace)
- Inter procedural Analysis
- Value Set Analysis (VSA)
- Type Inference (TIE / ReTypd)

Backend



Backend

- Pseudo C-Writer

Result

C Output
Recompiled Bin
Symbolic Execution
Emulation

How is the code organized?

Code Organization - Directory Overview

- radeco-lib/
  - src/
    - analysis/
    - frontend/
    - middle/
    - backend/
    - utils/

Code Organization - Frontend

- radeco-lib/
  - src/
    - analysis/
    - frontend/
      - bindings.rs - Structs and definitions for declaring and manipulating variable bindings.
      - containers.rs - Organizes instructions into Module and Function. Holds information between analysis.
      - source.rs - Defines the trait `Source`. Implements `Source` for r2 and reading from File.
      - ssaconstructor.rs - Constructs SSA from a given `Source`.
  - middle/
  - backend/
  - utils/

Code Organization - Middle (lib-ir)

- radeco-lib/
  - src/
    - analysis/
    - frontend/
    - middle/
      - ssa/ - Contains definitions to manipulate and build the SSA Form.
      - dce.rs - Dead Code Elimination.
      - dot.rs - Converts internal IR to graph using dot.
      - ir_writer.rs - Writes out IR to a human readable, LLVM IR like output.
      - phiplacement.rs - Used by ssaconstructor.rs to place phis at merge points.
      - regfile.rs - Parses and provides register definitions.
  - backend/
  - utils/

Code Organization - Analysis

- radeco-lib/
  - src/
    - analysis/
      - cse/ - Common Subexpression Elimination.
      - dom/ - Dominator Tree Construction.
      - interproc/ - Framework for inter procedural analysis.
      - matcher/ - Grep-and-Replace.
      - sccp/ - Sparse Conditional Constant Propagation.
      - tie/ - Type Inference.
      - valueset/ - Value Set Analysis (VSA).
    - frontend/
    - middle/
    - backend/
    - utils/

How can you contribute?

- Beginner issues
- Tests
- Documentation
- Bug Reports
- Feedback

Looking Forward …

- Regular release cycles
- Full Type Inference
- Pseudo C Output
- radeco tool and better integration with r2

Questions?