Introduction to Virtual Machines with VMKit

Harris Bakiras,

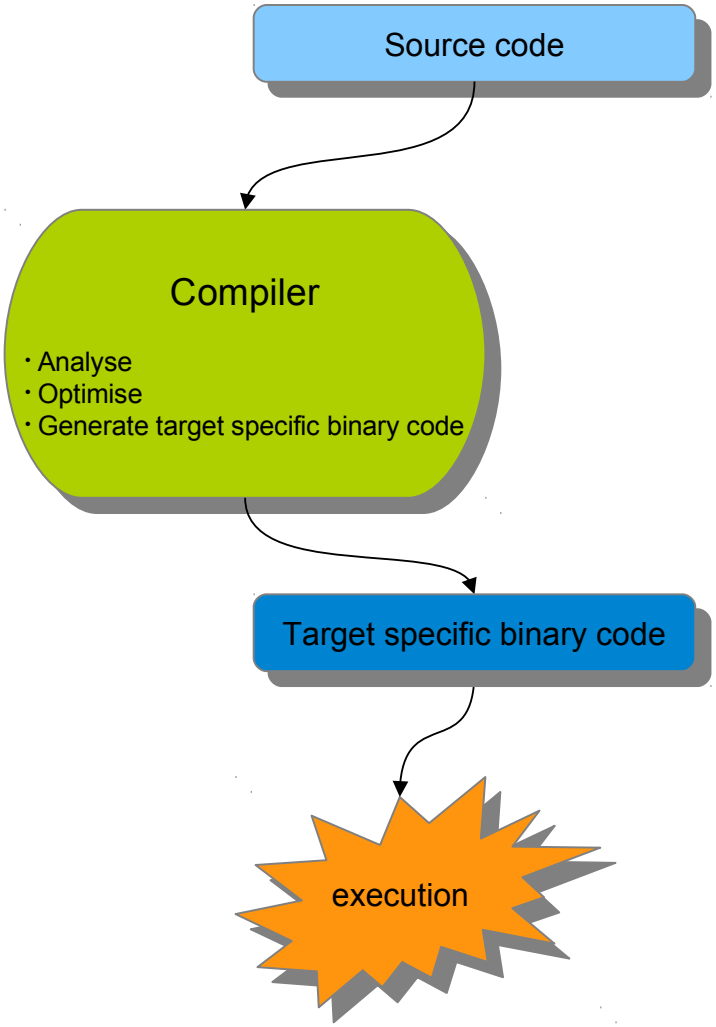
Supervisors : Gaël Thomas, Gilles Müller

LIP6 REGAL TEAM – INRIA (Paris/France)

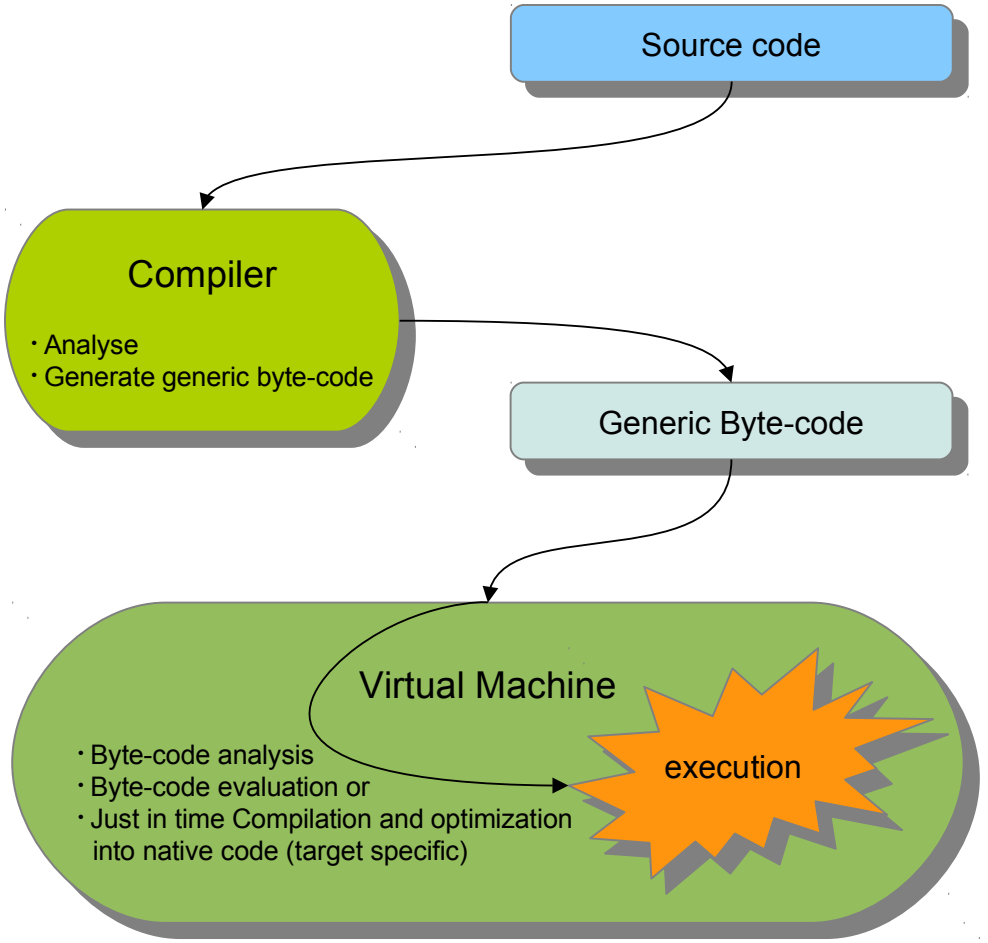
VMKit
a virtual machine substrate

Execution environments

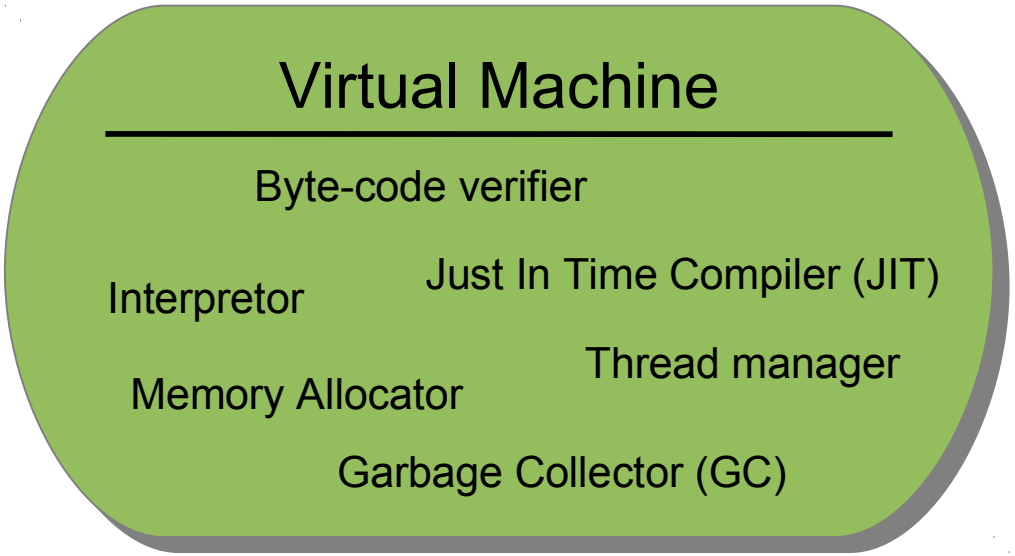
Native code execution (C, C++)



Virtual machine execution (Java, C#)



Virtual Machines



Problems

Developing time is extremely long !

How to test an idea with different languages ?

How to implement a new efficient virtual machine for new languages ?

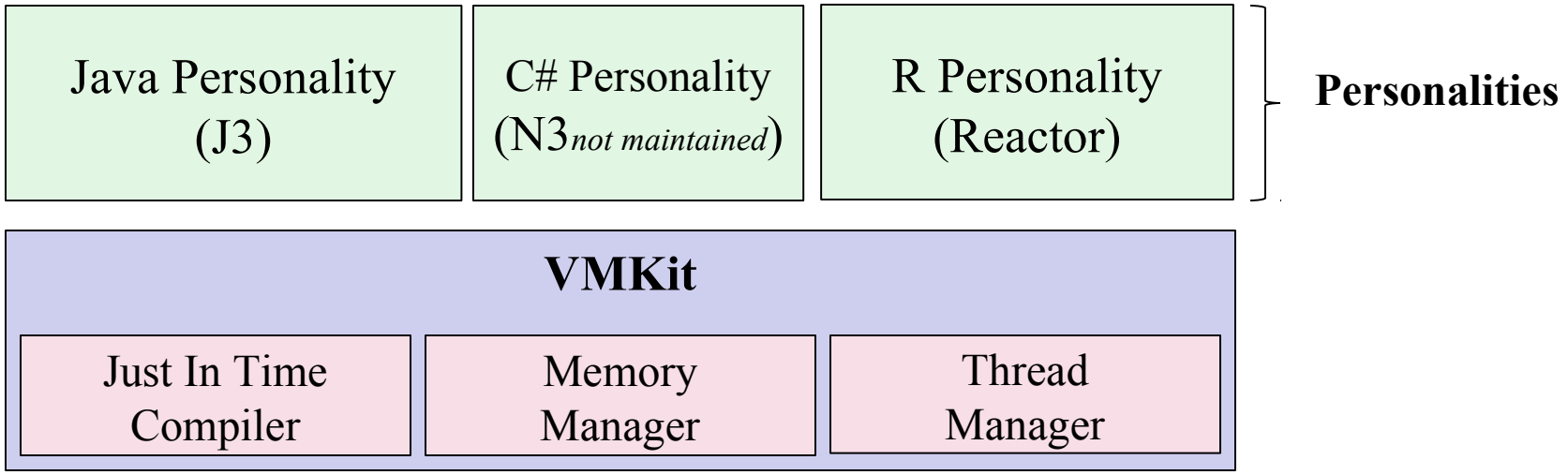
How to quickly extend existing languages ?

VMKit : a substrate of virtual machine

VMKit's goal : help experiments on VM

Objective : factorize VM's common components

- Just In Time Compiler : native code generation on the fly
- Memory manager : allocates and collects automatically free memory
- Thread manager: creates and synchronizes threads

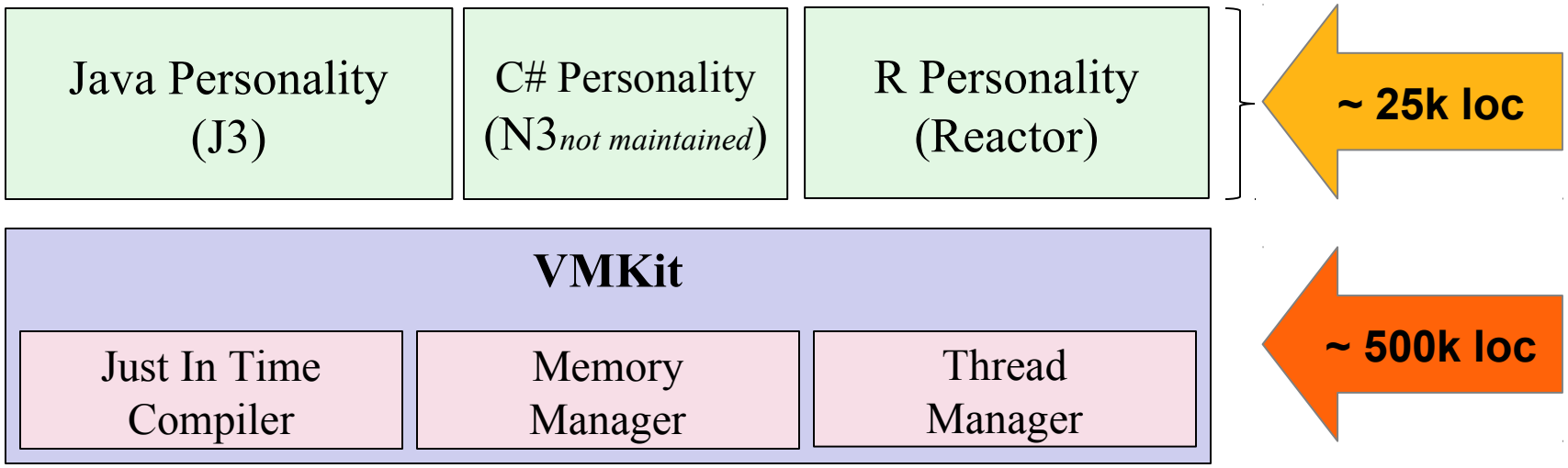


VMKit : a substrate of virtual machine

VMKit's goal : help experiments on VM

Objective : factorize VM's common components

- Just In Time Compiler : native code generation on the fly
- Memory manager : allocates and collects automatically free memory
- Thread manager: creates and synchronizes threads

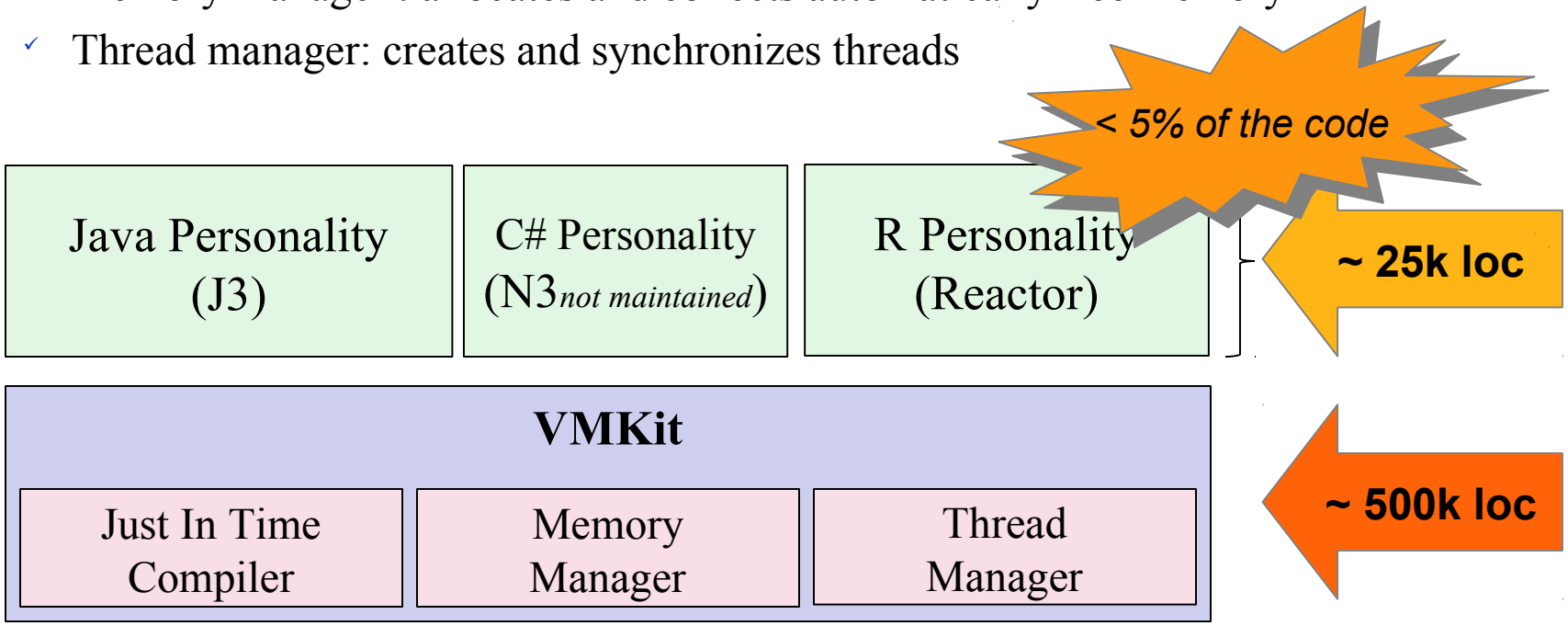


VMKit : a substrate of virtual machine

VMKit's goal : help experiments on VM

Objective : factorize VM's common components

- Just In Time Compiler : native code generation on the fly
- Memory manager : allocates and collects automatically free memory
- Thread manager: creates and synchronizes threads



VMKit
from a technical point of view

VMKit's Implementation

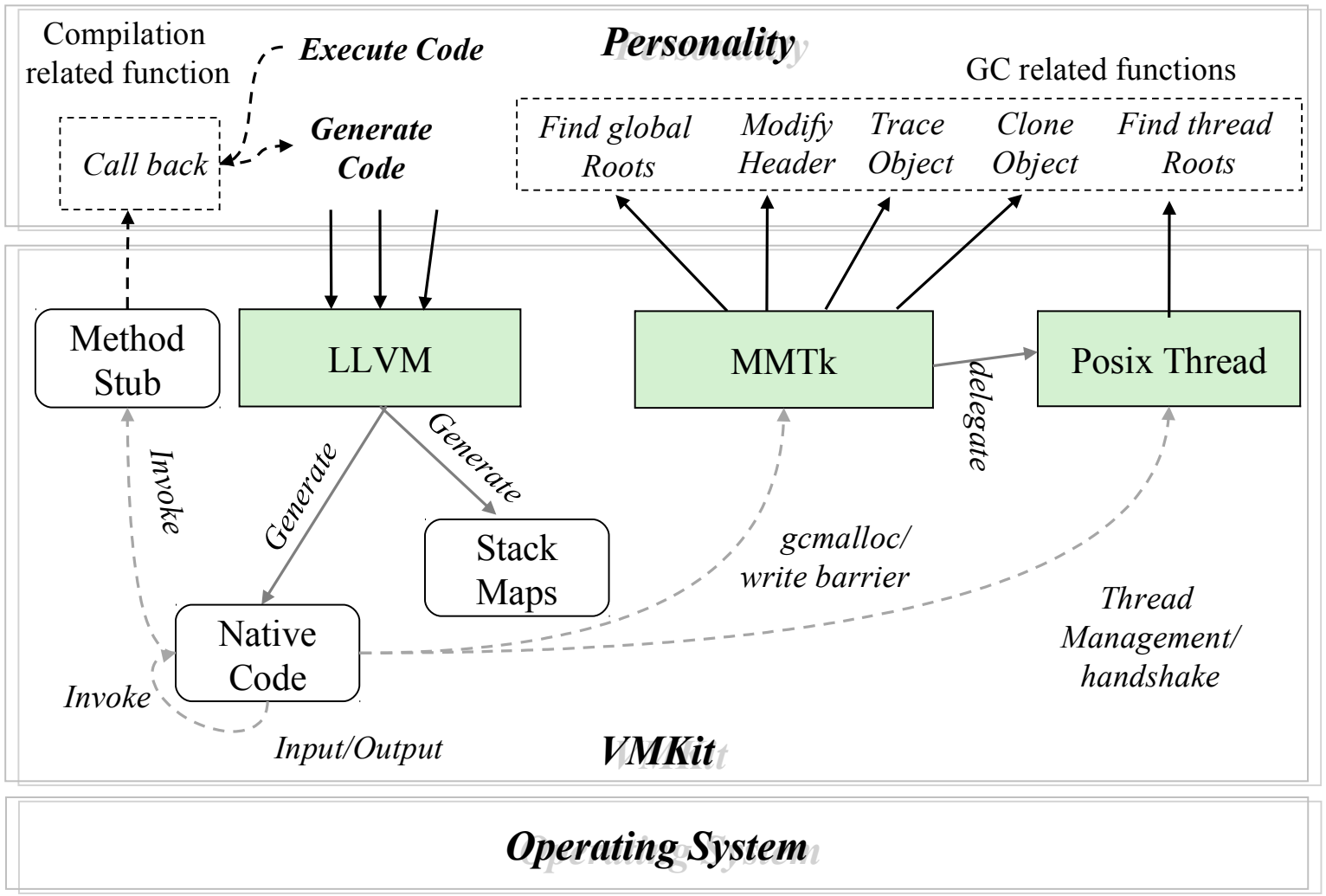
Implementation choice : **relies on external state of the art components**

- ✓ Just In Time Compiler (JIT) : LLVM [Lattner & Adve – CGO'04]
- ✓ Memory manager : MMTk [Blackburn et Al. – ICSE'04]
- ✓ Thread manager : Posix

VMKit = glue between the different components

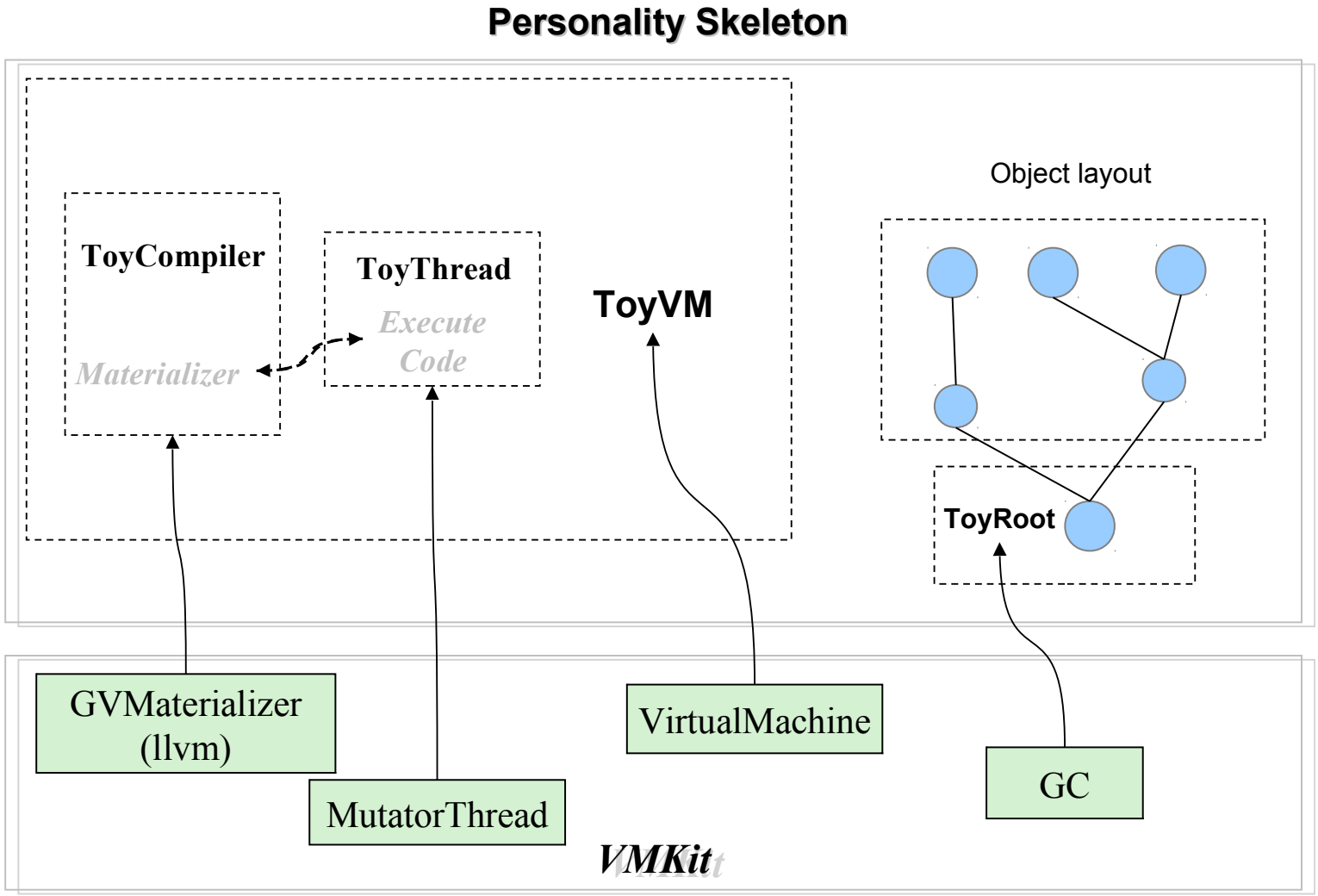
- Between JIT-C et memory manager = precise GC
- Between Thread and Memory managers = multi-threaded GC

Detailed Architecture

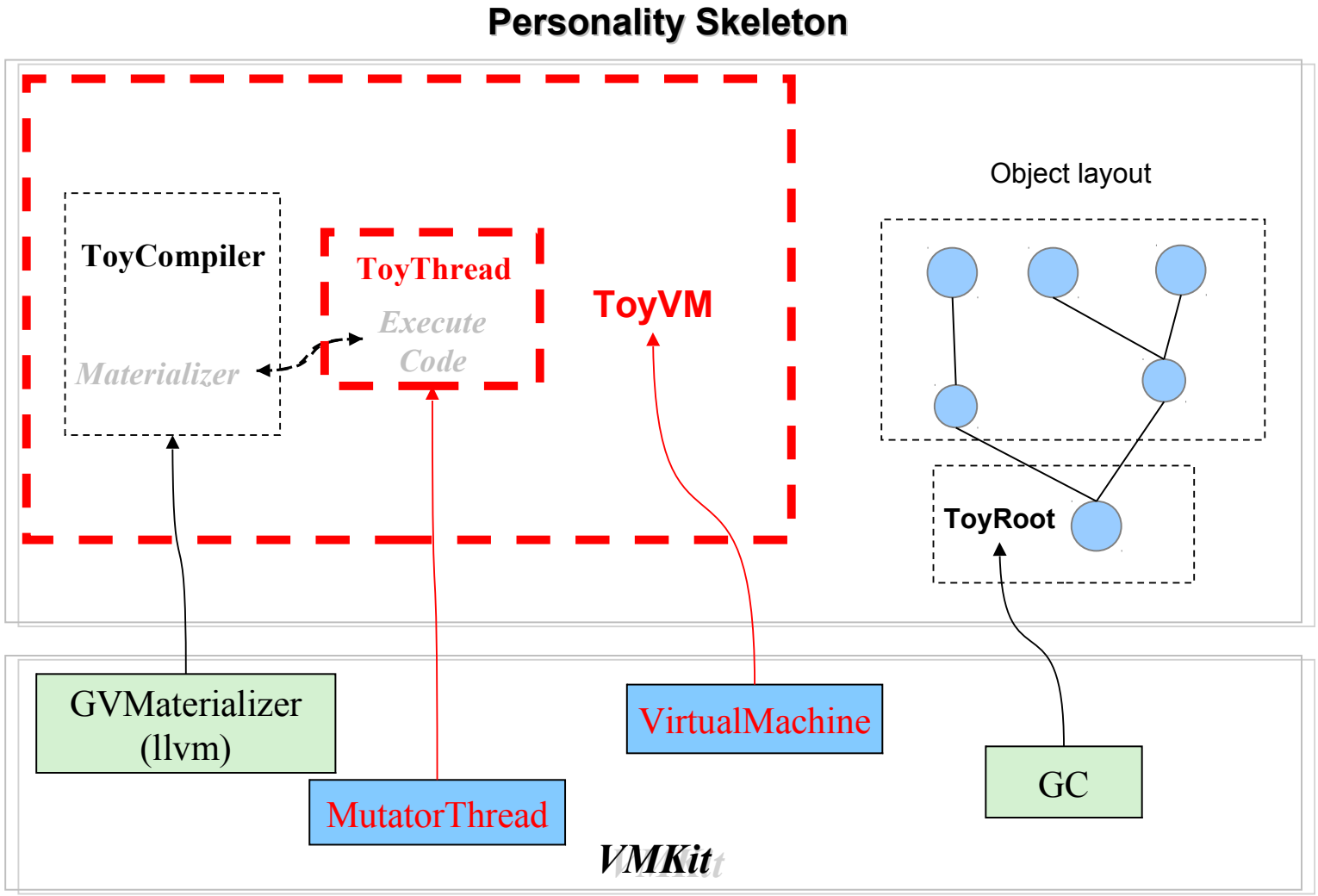


Minimal VM (ToyVM) for the tutorial

ToyVM's Architecture



ToyVM's Architecture



ToyVM and ToyThread

ToyVM ← VirtualMachine

- *Thread management*
- *Garbage collectors entry point*
- *Backtrace (execution stack browsing)*

} Provided

- Global variables tracing
- Exceptions management

} Developers charge

ToyThread ← MutatorThread ← Thread

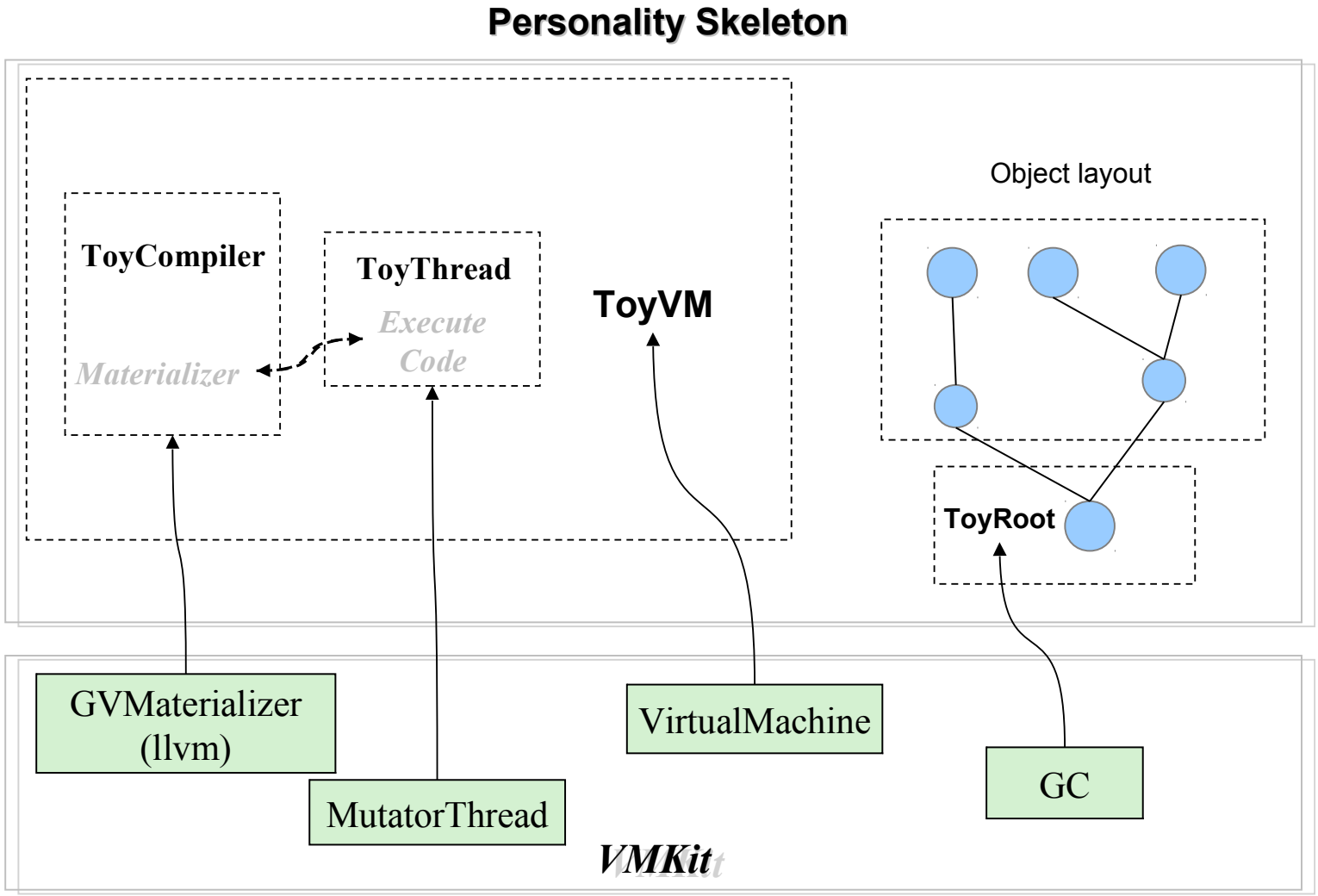
- *Garbage collectors thread synchronization*
- *Execution stack Scan during GC*

} Provided

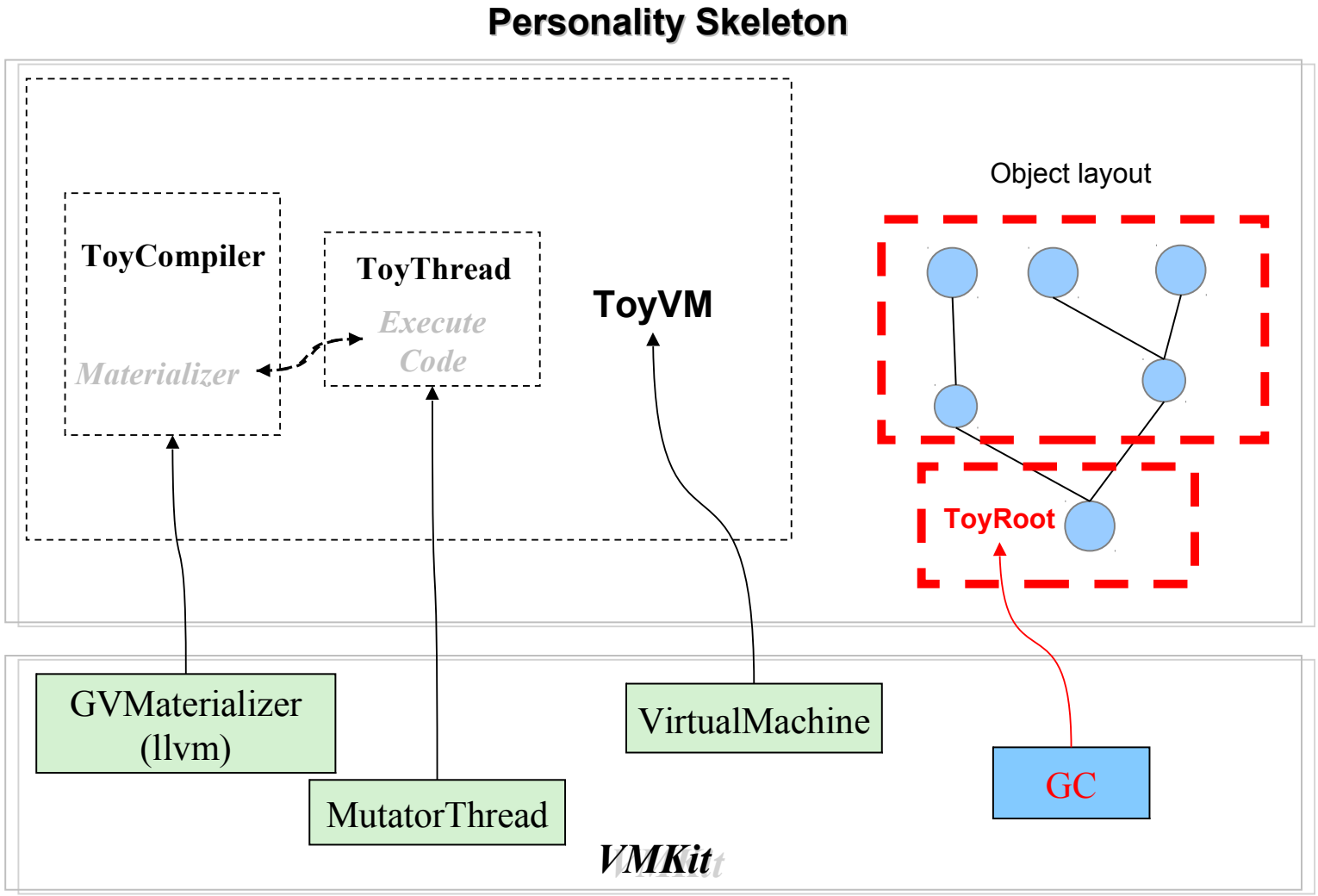
- Global variables tracing
- Main execution method

} Developers charge

ToyVM's Architecture



ToyVM's Architecture



ToyRoot (tag)

ToyRoot ← vmkit::gc

- Tag collectible objects (stack maps)

```
[ ……. ]

ToyRoot* F (ToyRoot* param) {
  TOY_PARAM(param) ;
  TOY_VAR(ToyRoot, val) ;
        [ init val … ]
  val = g(param, val) ;
  return val ;
}


ToyRoot* G (ToyRoot* a, ToyRoot* b){
  TOY_PARAM(a) ;
  TOY_PARAM(b) ;
  TOY_VAR(ToyRoot, res) ;
  res = a.doSomething(b) ;
  return res ;
}

[ ……. ]
```

ToyRoot (tag)

ToyRoot ← vmkit::gc

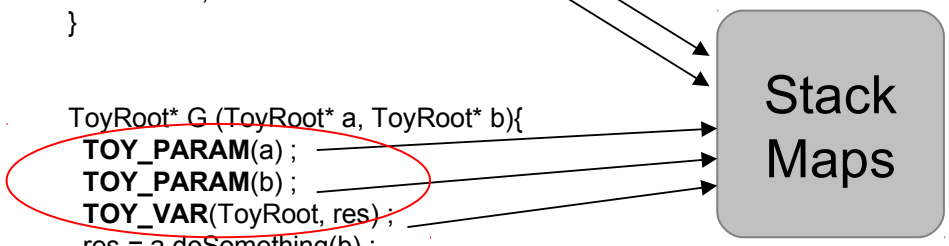
- Tag collectible objects (stack maps)

```
[ ……. ]

ToyRoot* F (ToyRoot* param) {
  TOY_PARAM(param) ;
  TOY_VAR(ToyRoot, val) ;
        [ init val … ]
  val = g(param, val) ;
  return val ;
}


ToyRoot* G (ToyRoot* a, ToyRoot* b){
  TOY_PARAM(a) ;
  TOY_PARAM(b) ;
  TOY_VAR(ToyRoot, res) ;
  res = a.doSomething(b) ;
  return res ;
}

[ ……. ]
```



ToyRoot (tag)

ToyRoot ← vmkit::gc

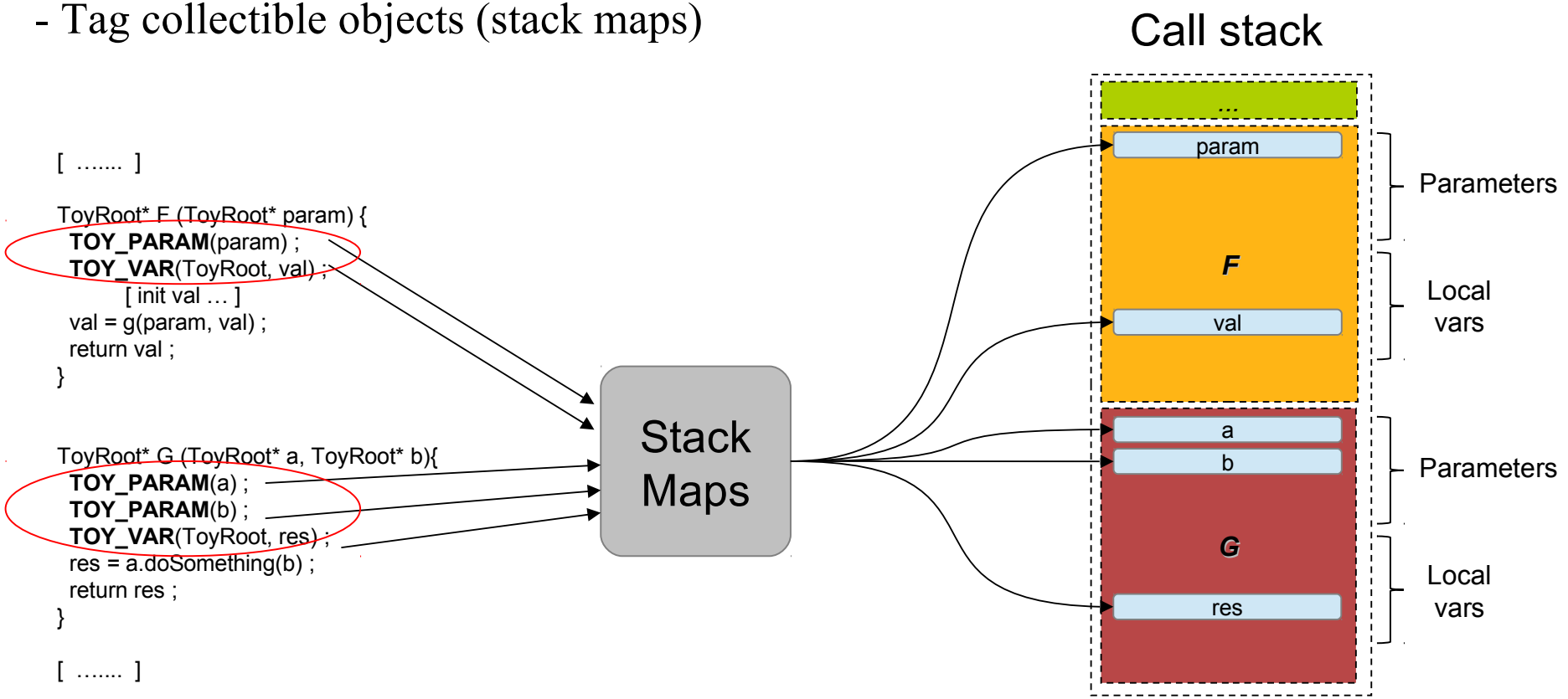
- Tag collectible objects (stack maps)

```
[ ……. ]

ToyRoot* F (ToyRoot* param) {
  TOY_PARAM(param) ;
  TOY_VAR(ToyRoot, val) ;
        [ init val … ]
  val = g(param, val) ;
  return val ;
}


ToyRoot* G (ToyRoot* a, ToyRoot* b){
  TOY_PARAM(a) ;
  TOY_PARAM(b) ;
  TOY_VAR(ToyRoot, res) ;
  res = a.doSomething(b) ;
  return res ;
}

[ ……. ]
```

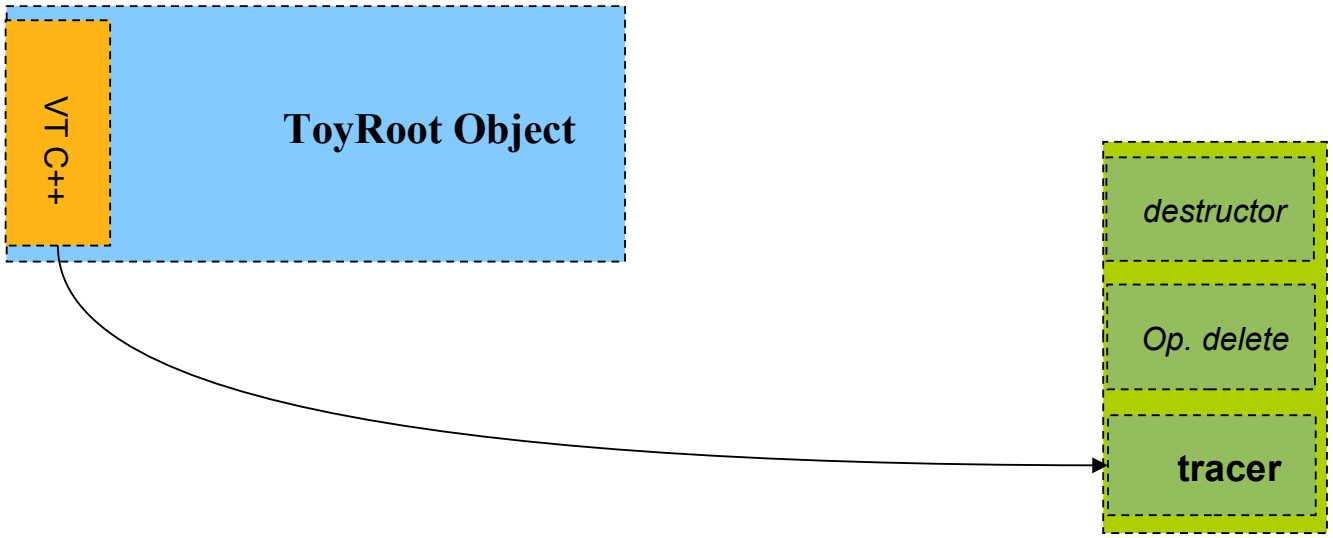


ToyRoot (tracer)

ToyRoot ← vmkit::gc

- Tag collectible objects (stack maps)

- Object tracing method

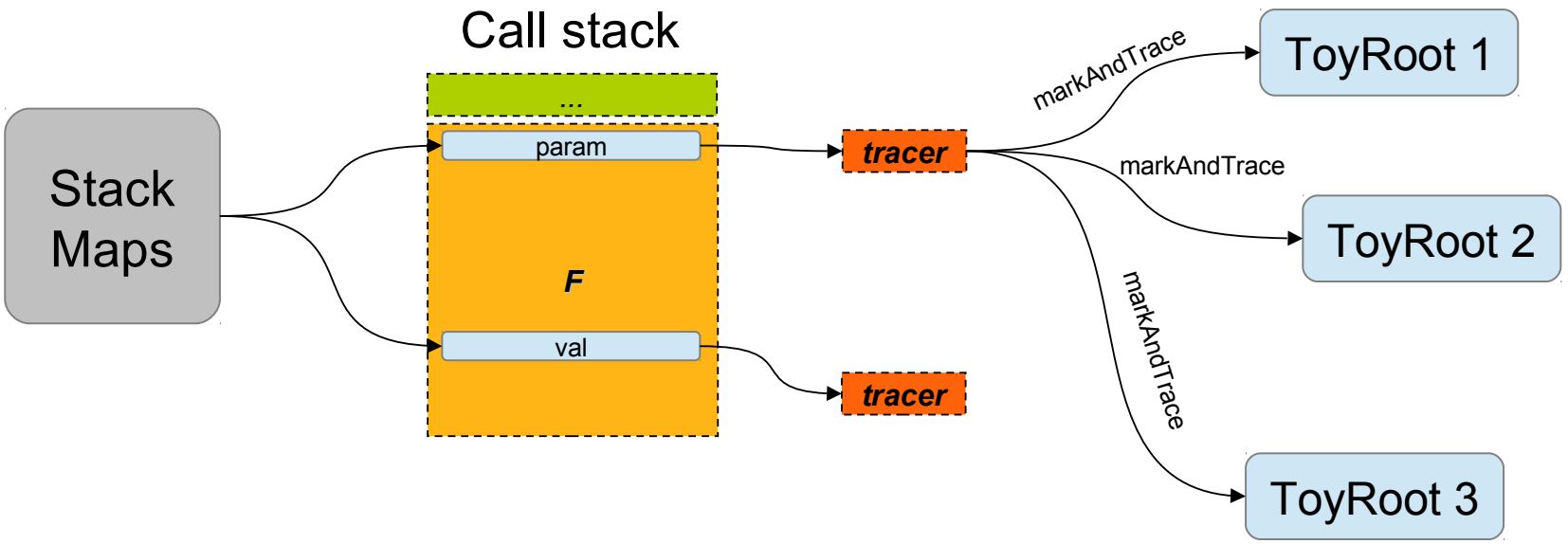


ToyRoot (tracer)

ToyRoot ← vmkit::gc

- Tag collectible objects (stack maps)

- Object tracing method



ToyRoot (allocation)

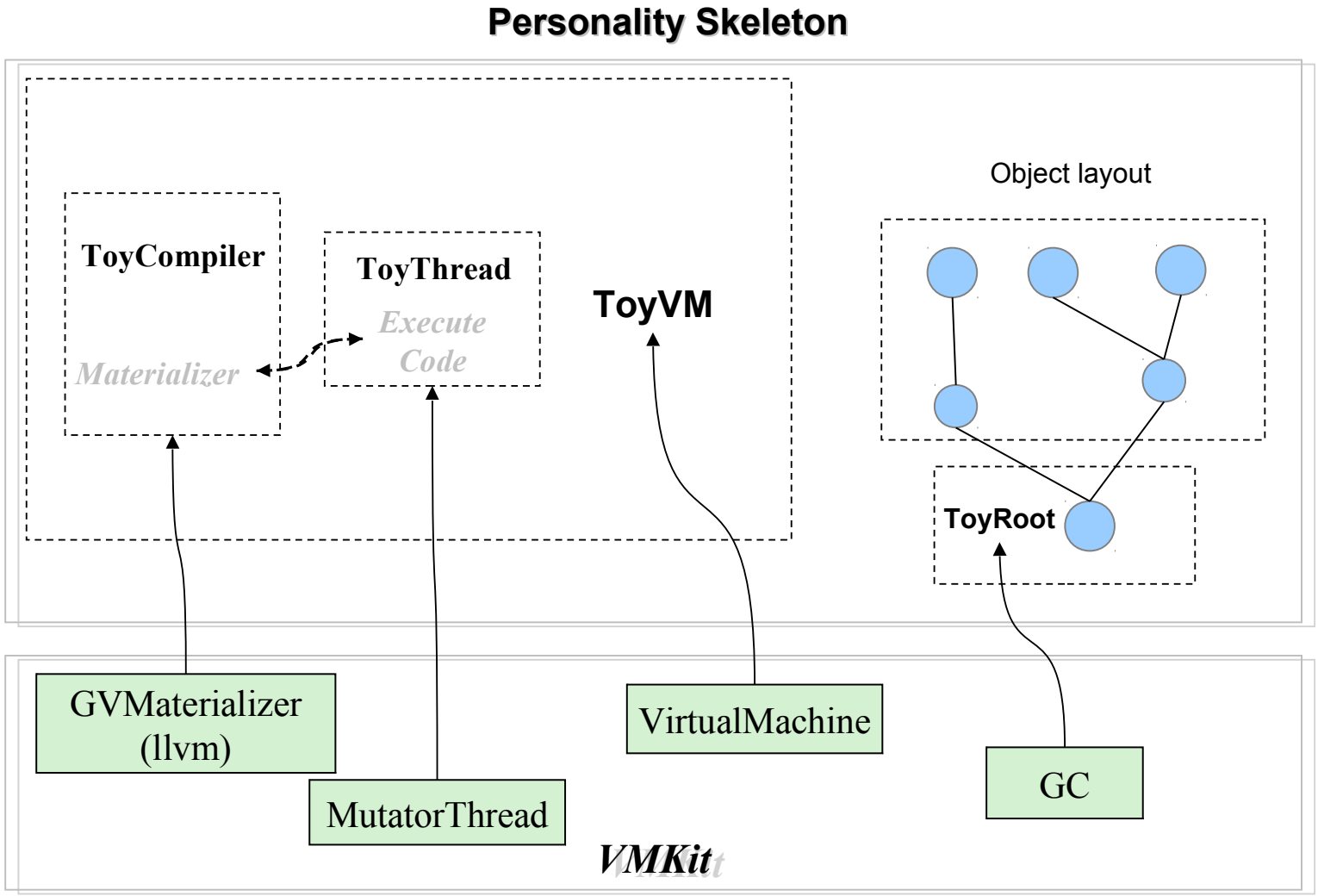
ToyRoot ← vmkit::gc

- Tag collectible objects (stack maps)

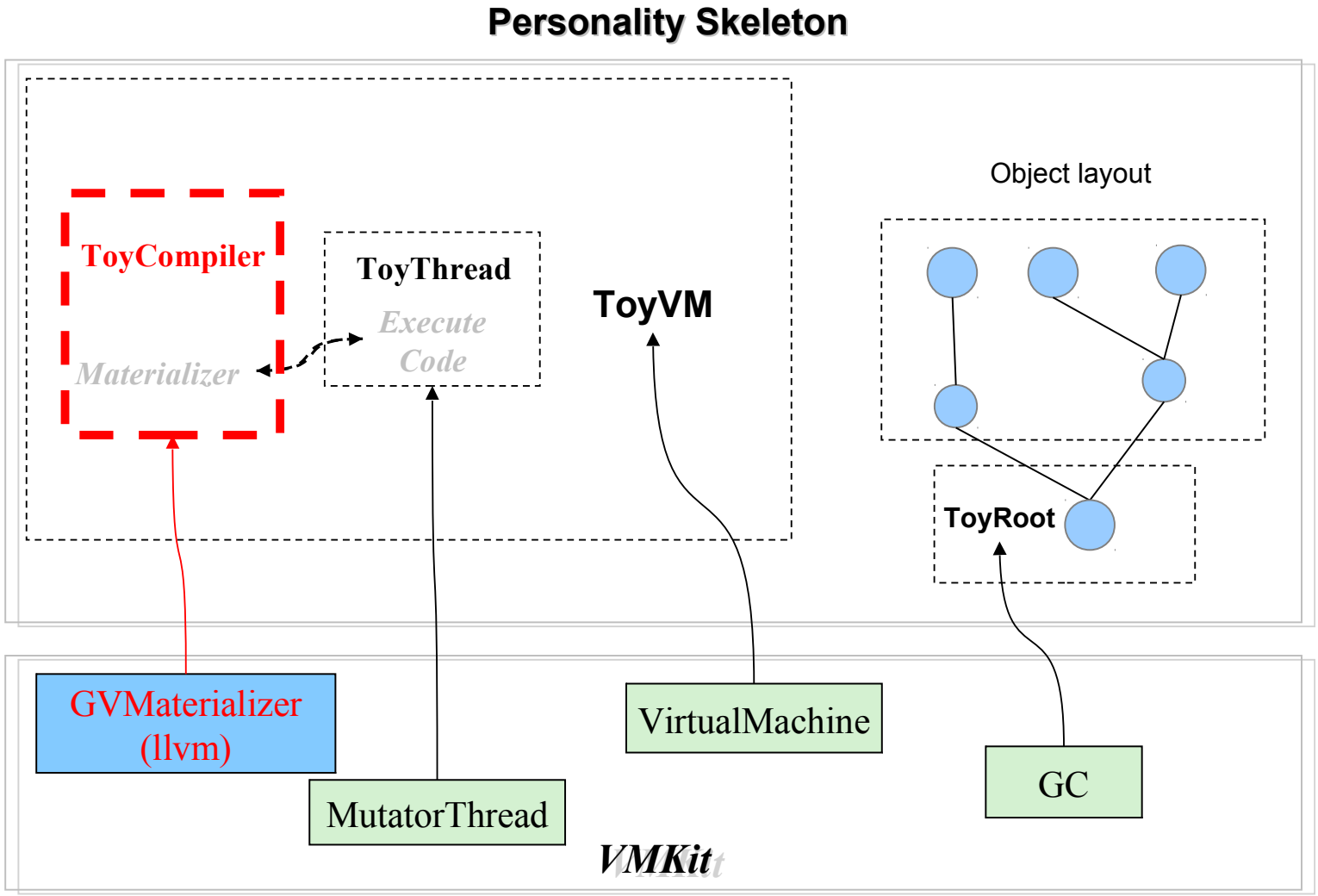
- Object tracing method

- GC objects allocation
  - Override *operator new*
  - Call to *new* forbidden for gc objects (opaque parameter)

ToyVM's Architecture



ToyVM's Architecture



ToyCompiler (load IR)

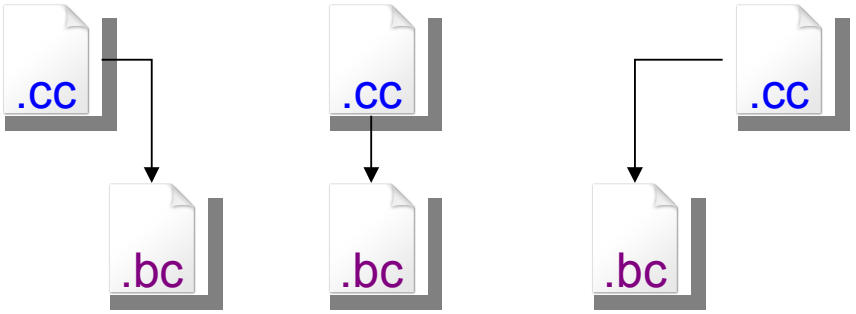
ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

ToyCompiler (load IR)

ToyCompiler ← GVMaterializer

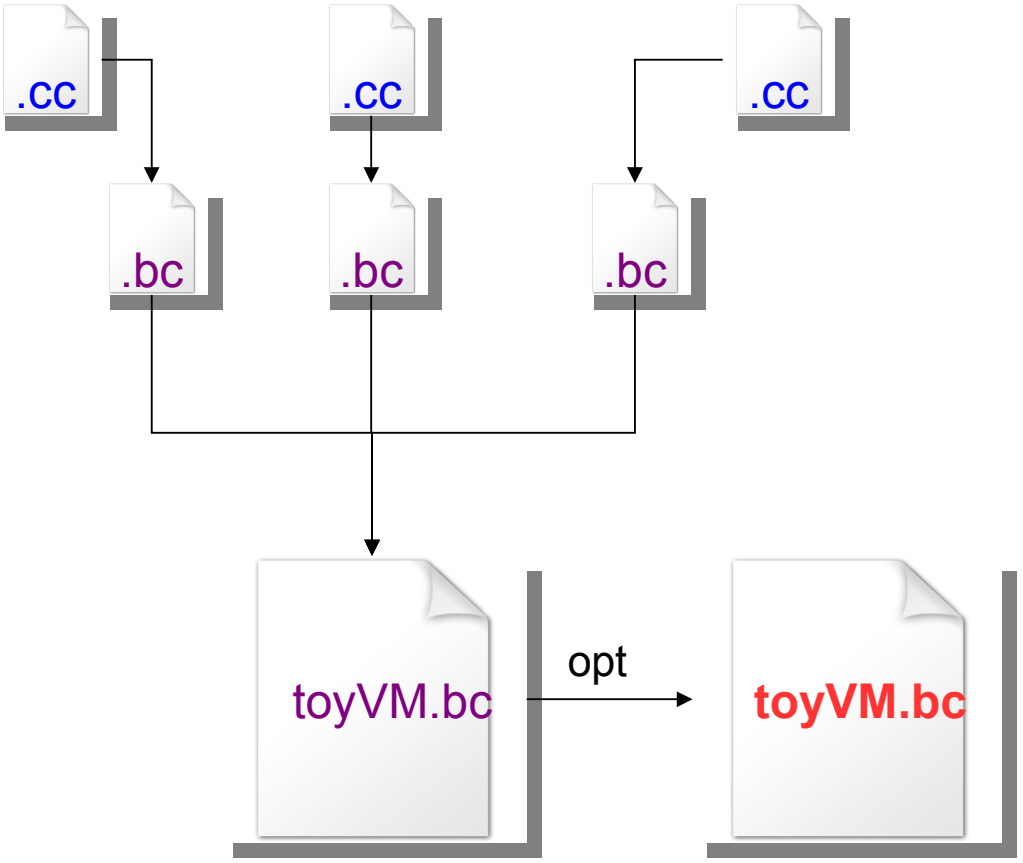
Retrieving ToyVM's IR (LLVM intermediate representation)



ToyCompiler (load IR)

ToyCompiler ← GVMaterializer

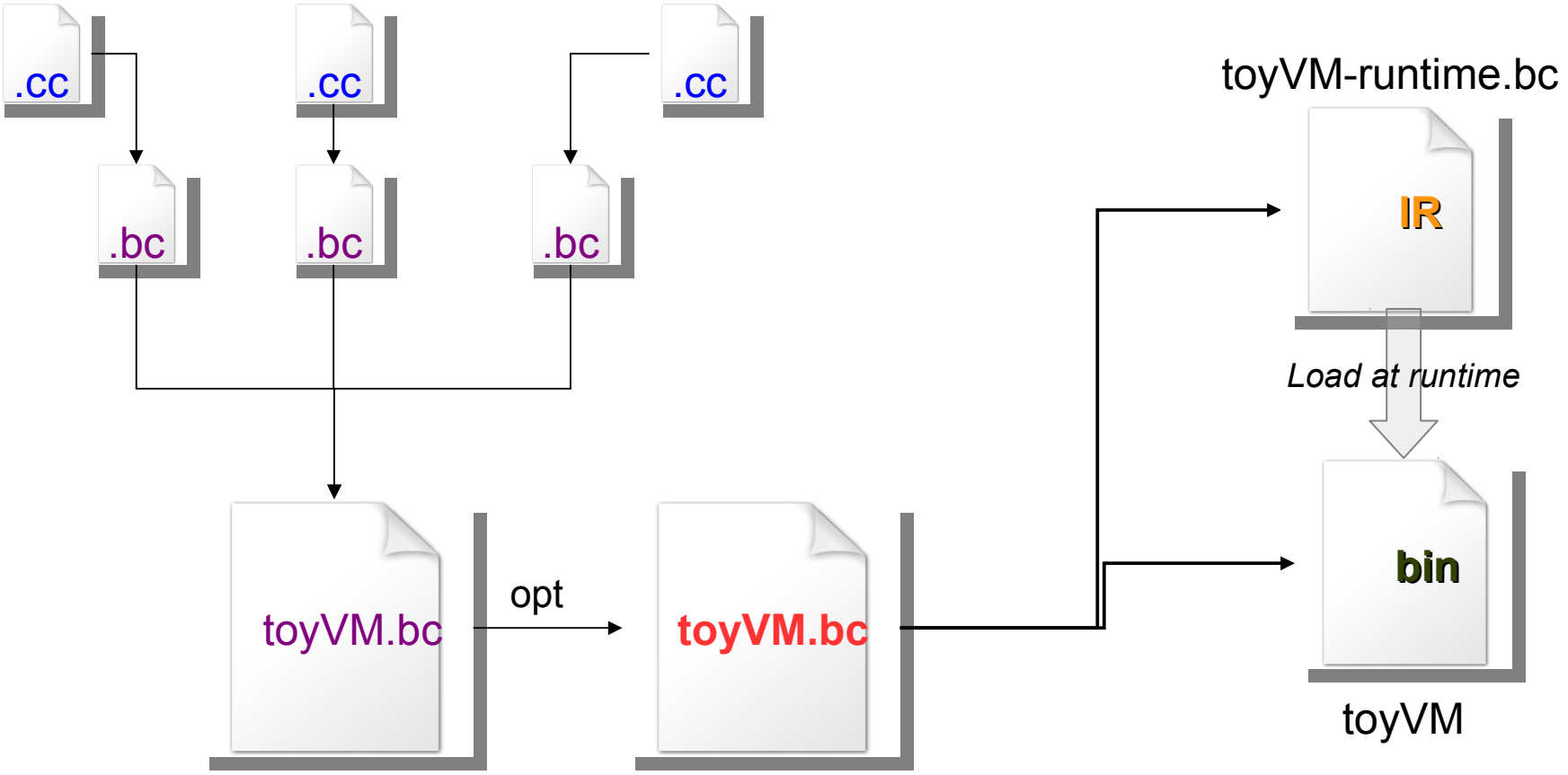
Retrieving ToyVM's IR (LLVM intermediate representation)



ToyCompiler (load IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

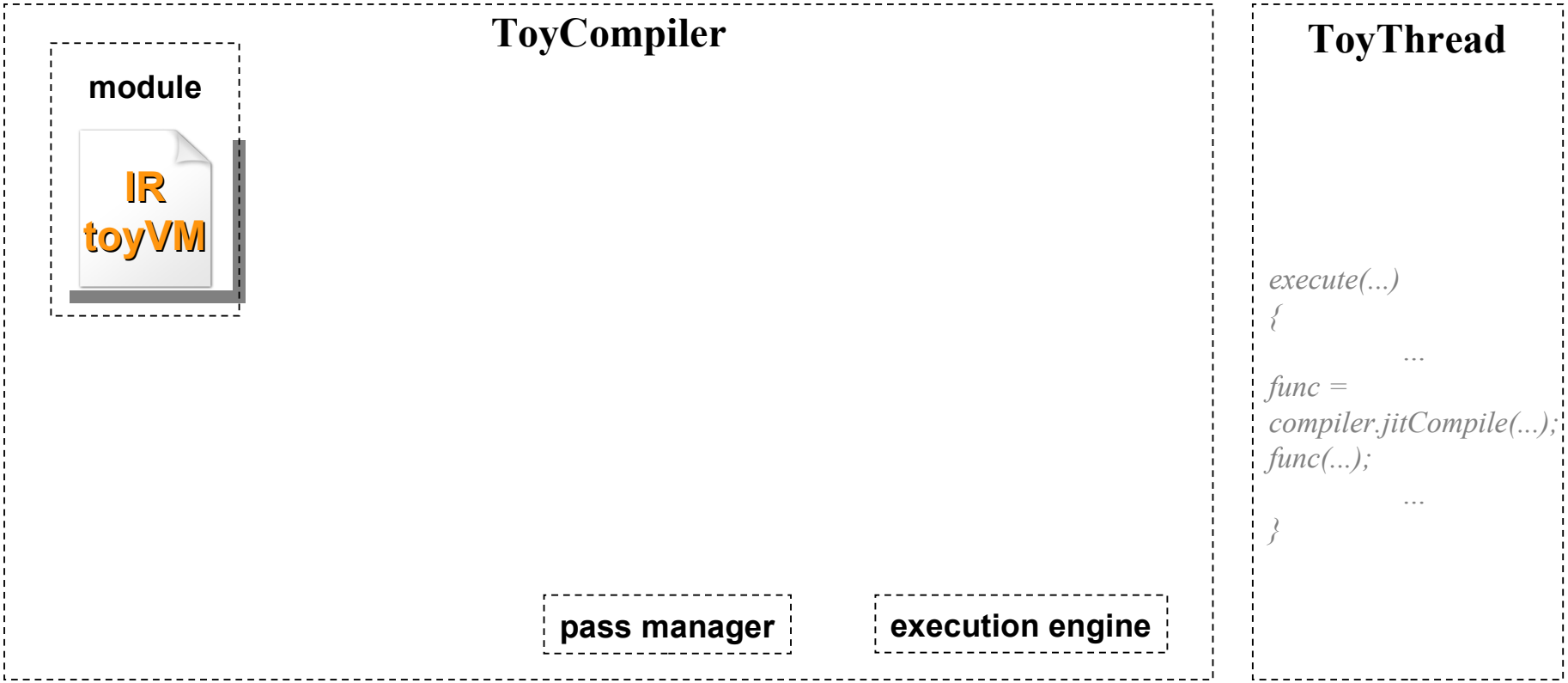


ToyCompiler (generate IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- ✓ JIT compiling

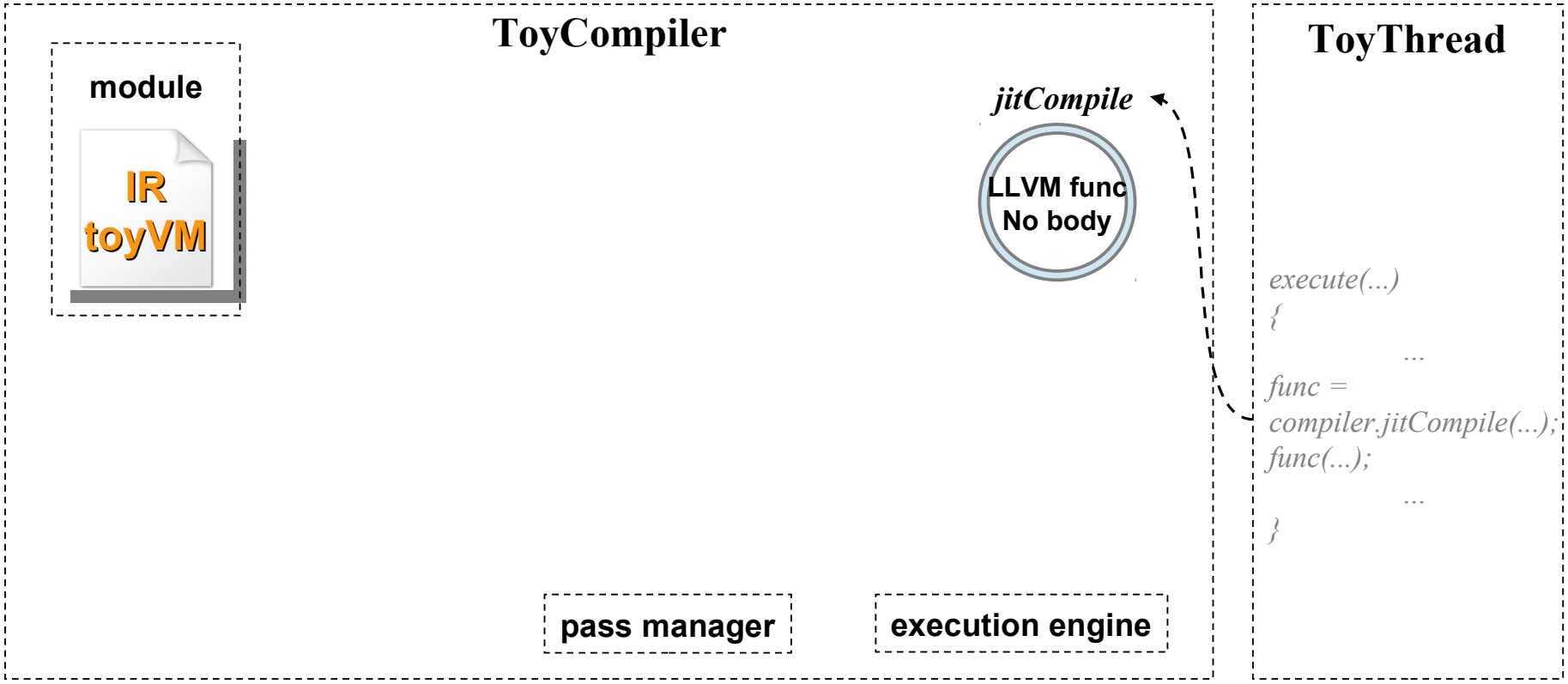


ToyCompiler (generate IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling

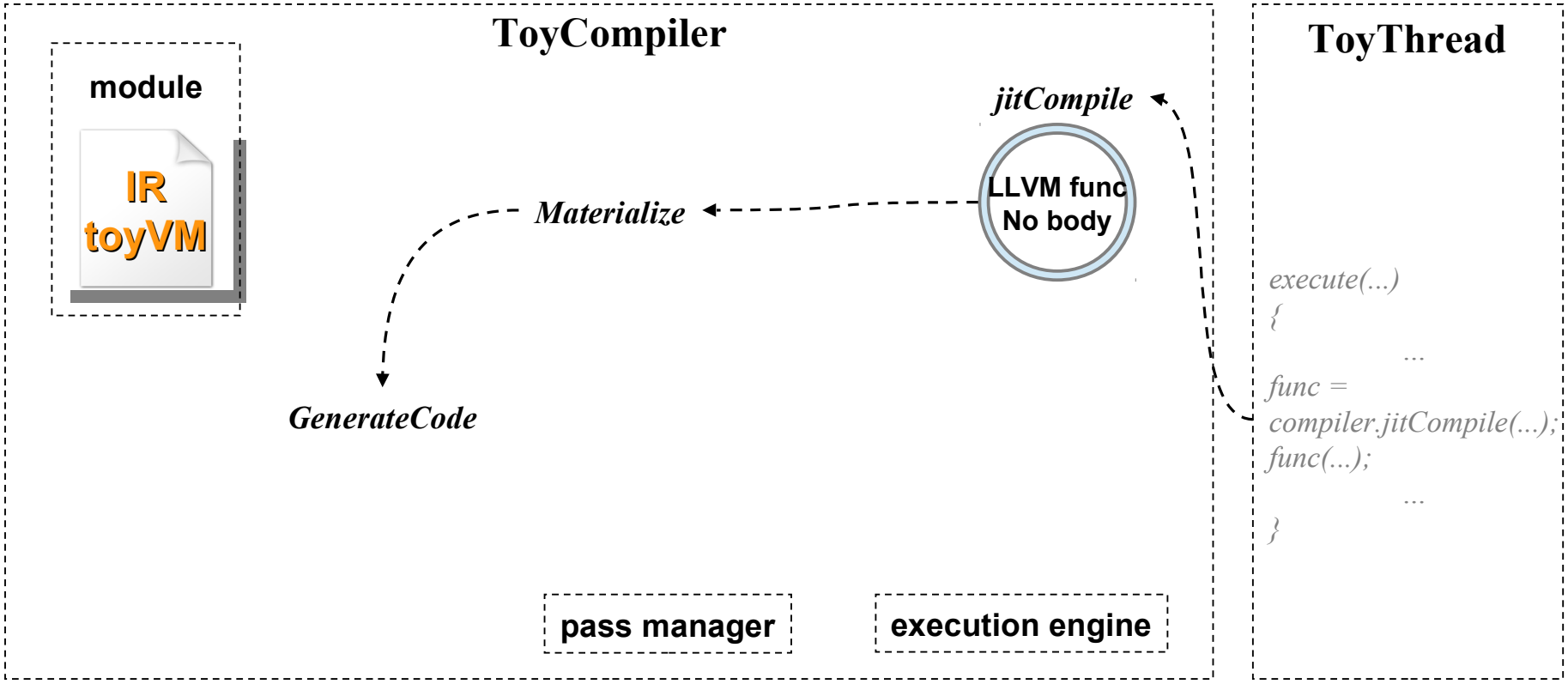


ToyCompiler (generate IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling

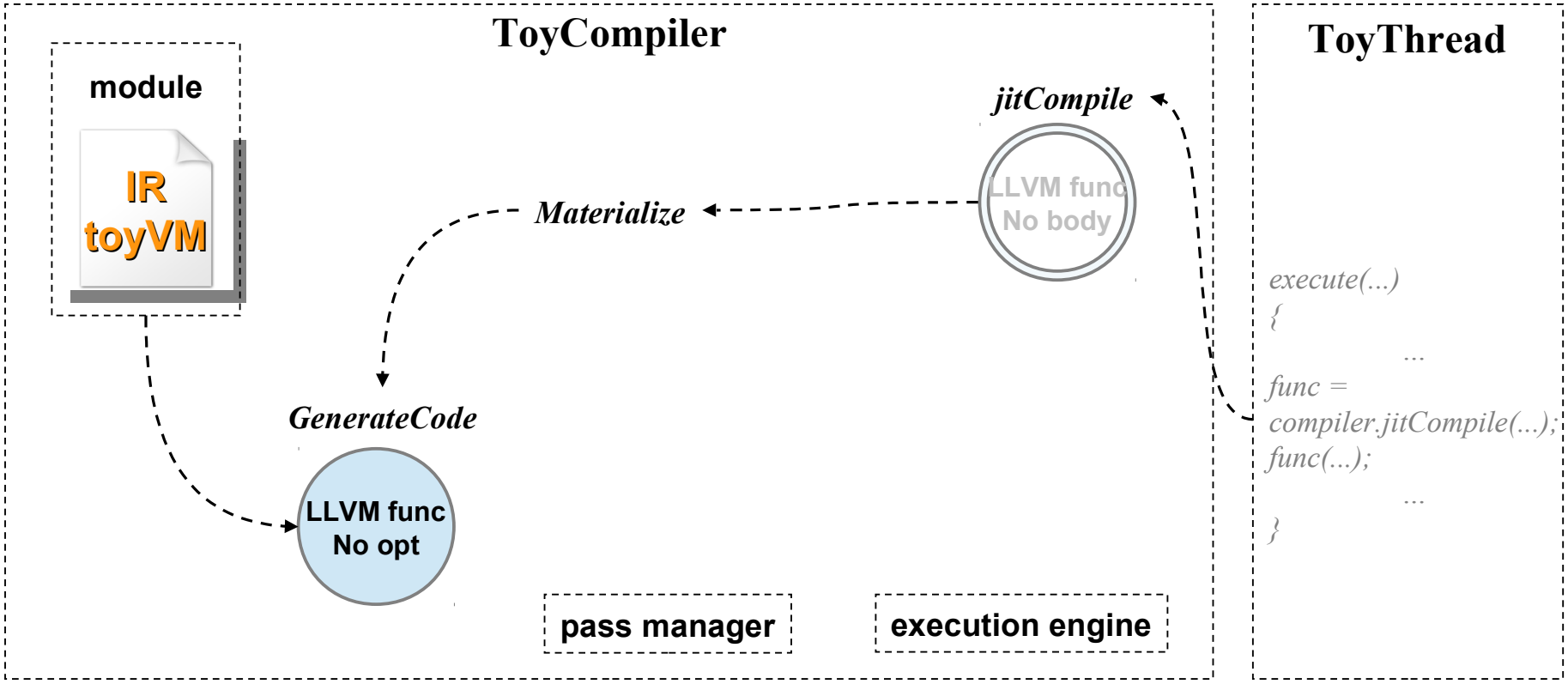


ToyCompiler (generate IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling

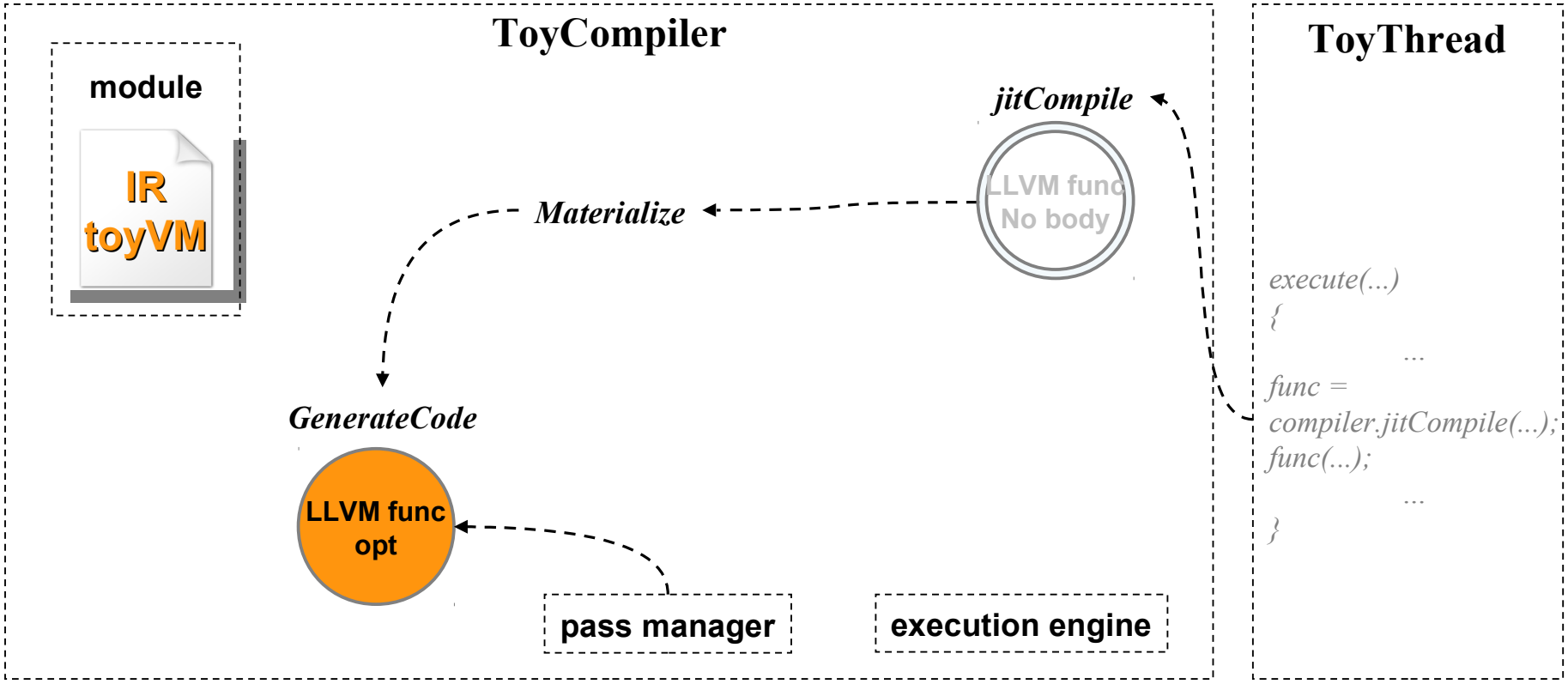


ToyCompiler (optimize IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling

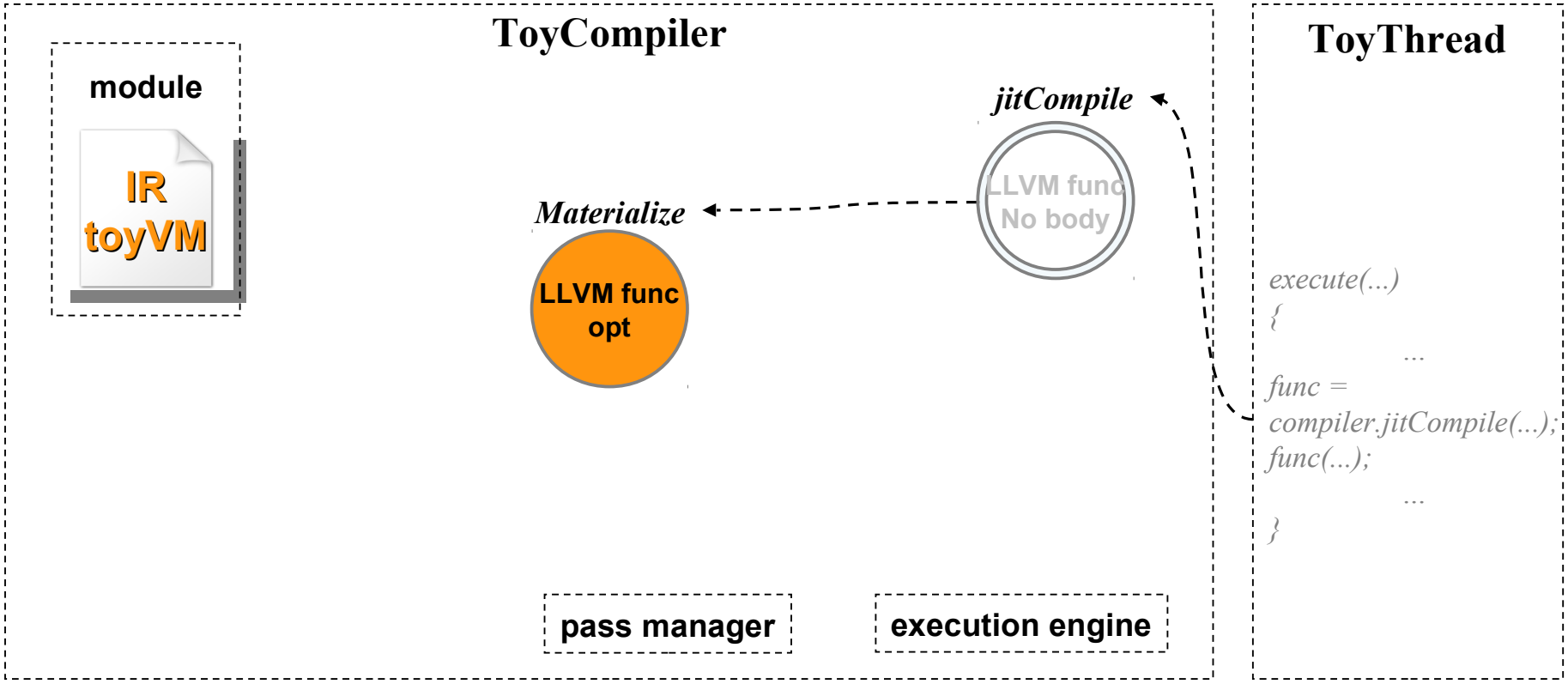


ToyCompiler (optimize IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling

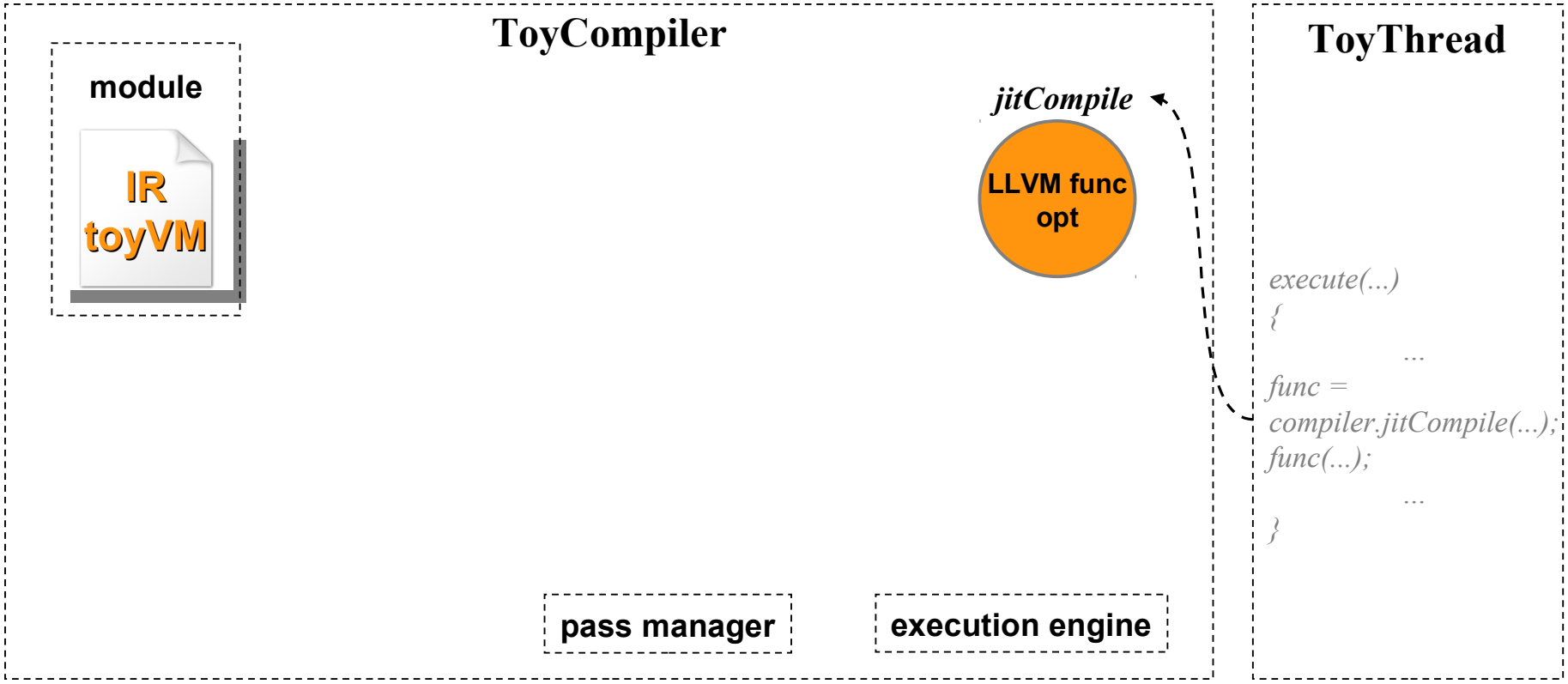


ToyCompiler (optimize IR)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling

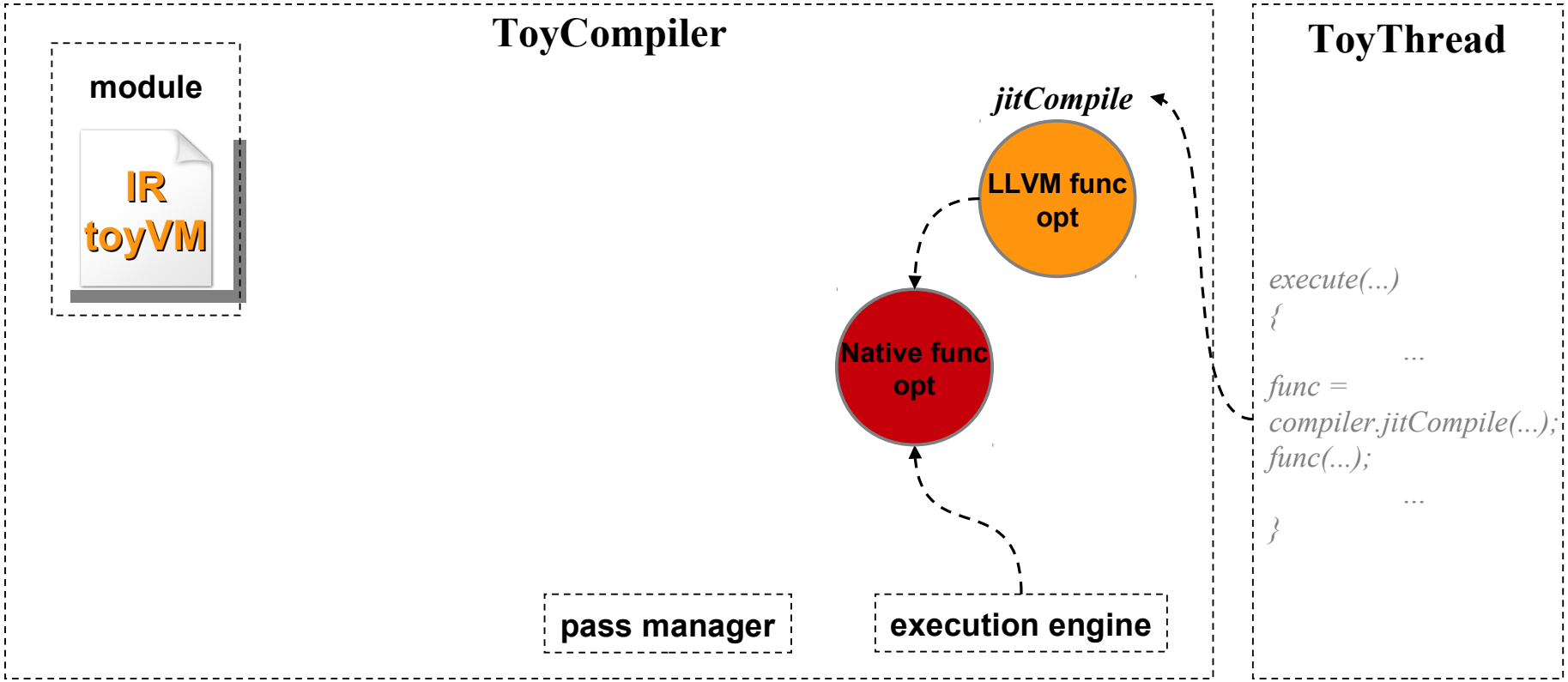


ToyCompiler (IR to native)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling

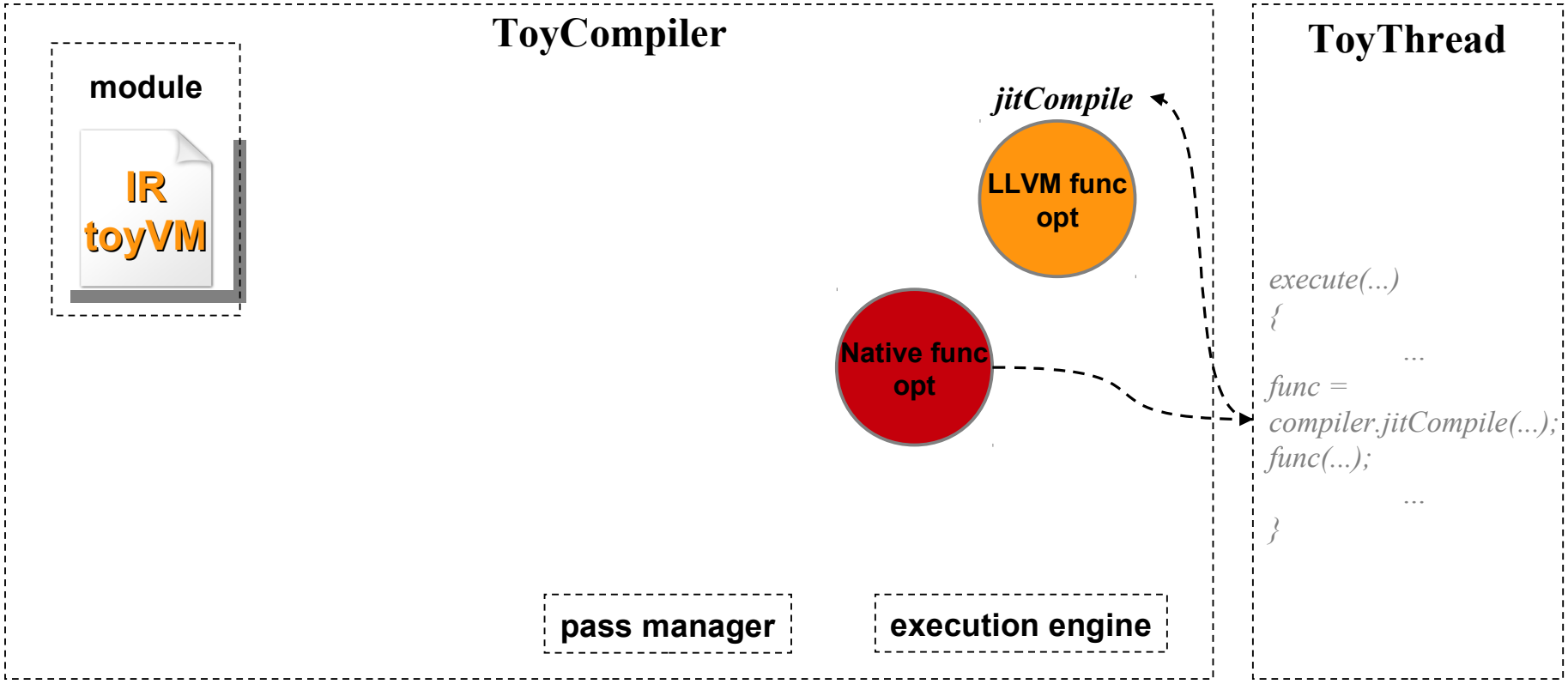


ToyCompiler (IR to native)

ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling



ToyCompiler

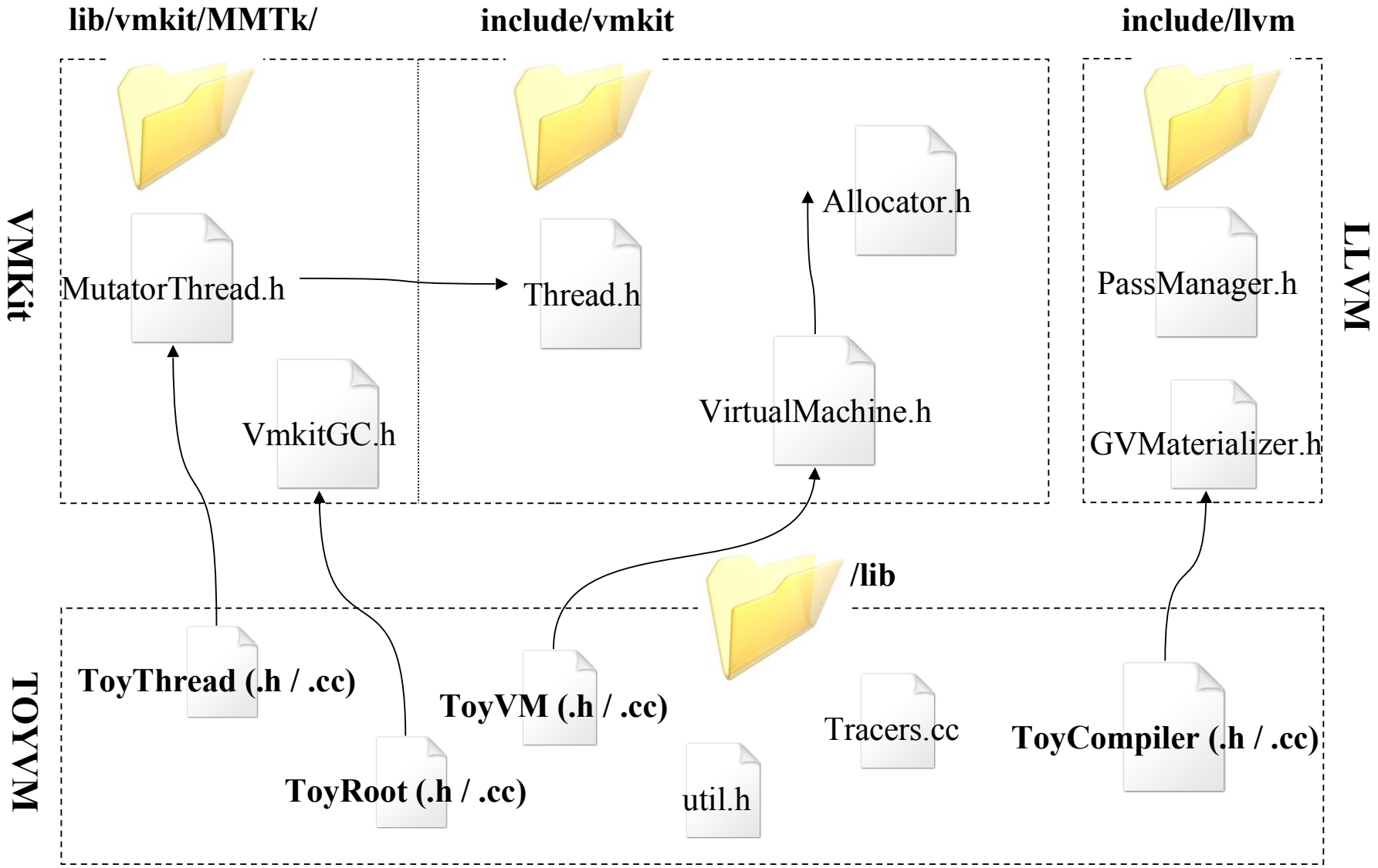
ToyCompiler ← GVMaterializer

Retrieving ToyVM's IR (LLVM intermediate representation)

- JIT compiling
  - Generation of IR
  - Optimization of IR
  - Convertion of IR into native code

Questions ?

Let's code !


lib/vmkit/MMTk/
include/vmkit
include/llvm
VMKit
LLVM
MutatorThread.h
Thread.h
Allocator.h
PassManager.h
VmkitGC.h
VirtualMachine.h
GVMaterializer.h
/lib
TOYVM
ToyThread (.h / .cc)
ToyVM (.h / .cc)
Tracers.cc
ToyCompiler (.h / .cc)
ToyRoot (.h / .cc)
util.h