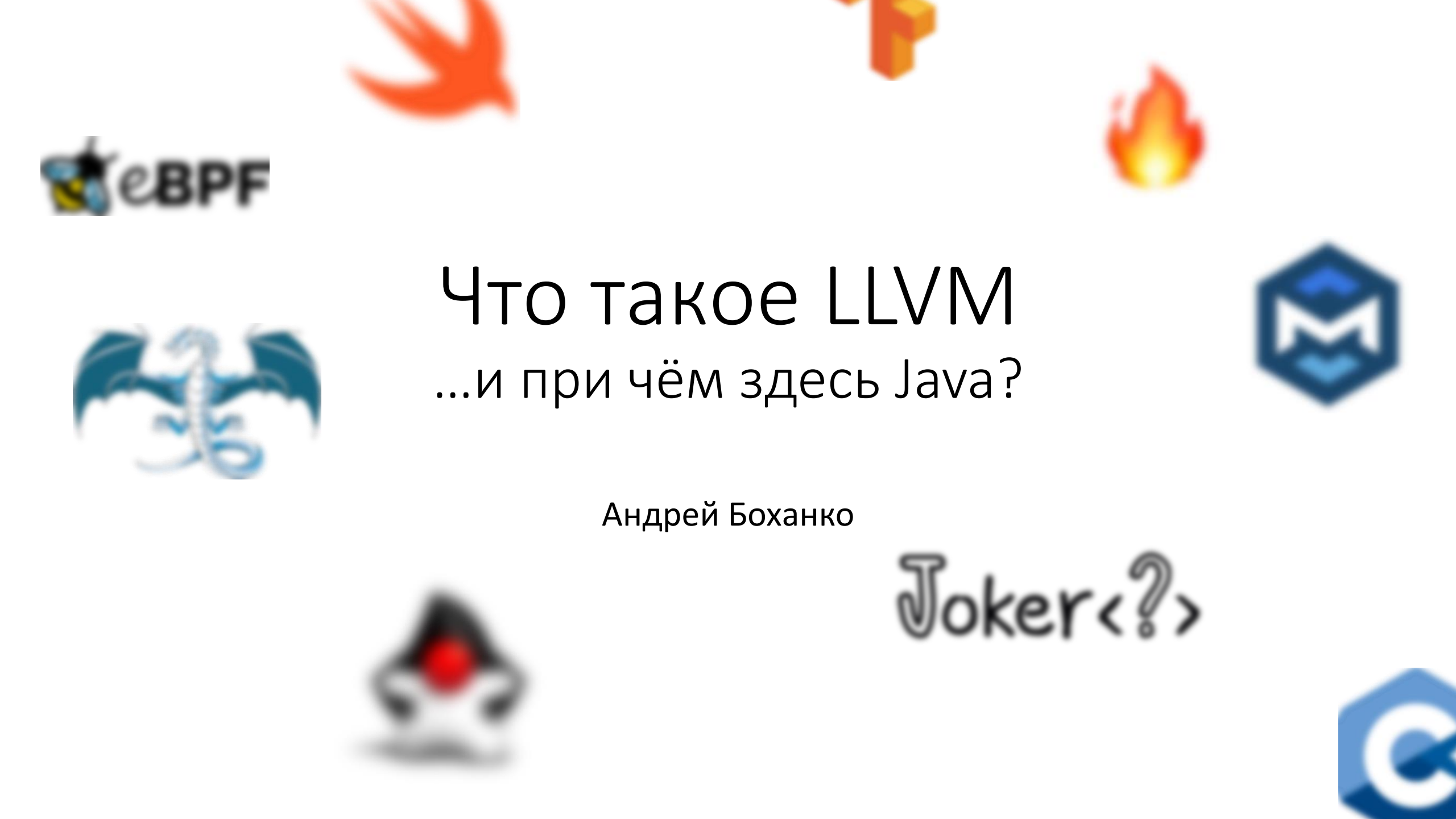


Что такое LLVM

…и при чём здесь Java?

Андрей Боханко

О чём этот доклад?

Что такое LLVM?

Что означают эти буквы? История и философия.

Как LLVM влияет на мир вокруг нас?

(да, и на каждого из вас тоже!)

При чём здесь Java?

Обо мне

МЦСТ ЭЛЬБРУС

intel

NXP

align

HUAWEI

2012
В первый раз посетил LLVM DevMeeting

2013
Первый внутренний проект на основе LLVM

Май 2015
Первый патч, принятый в апстрим

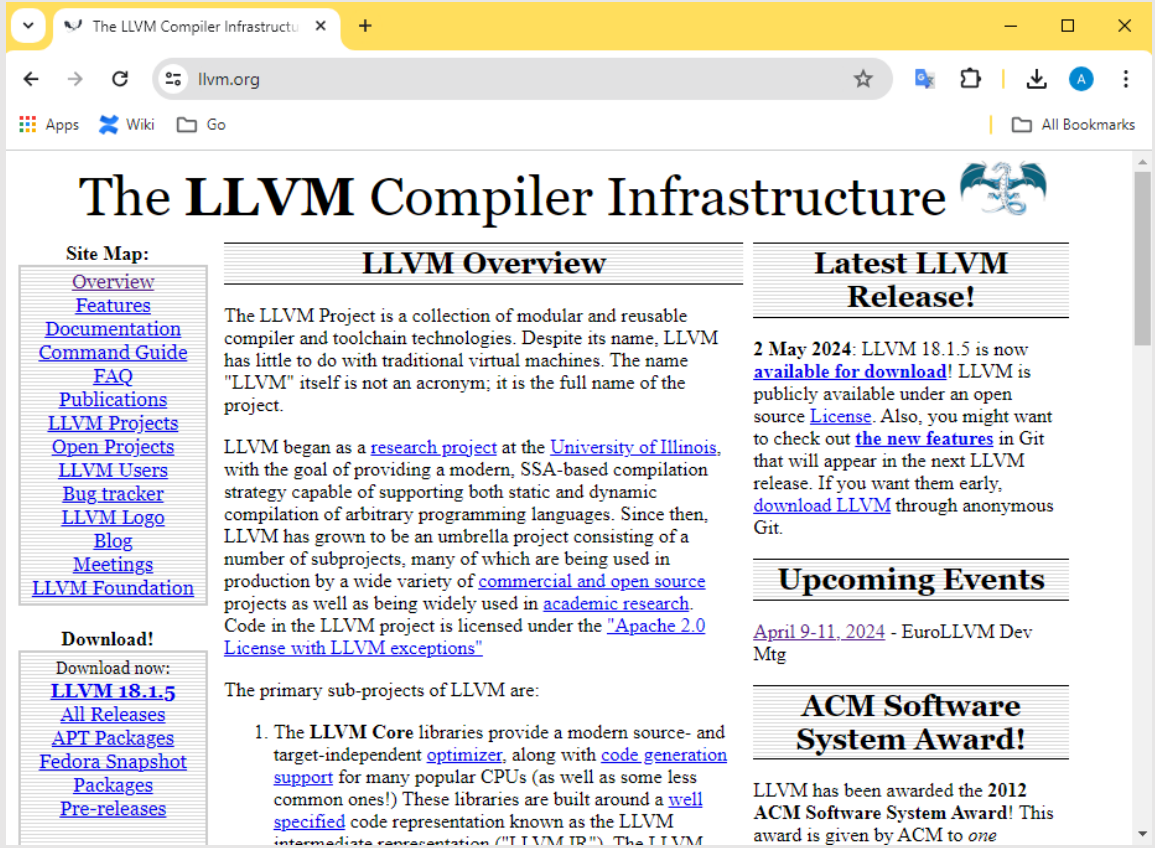
Июль 2015
Commit access

2017
Впервые включён в программный комитет LLVM DevMeeting

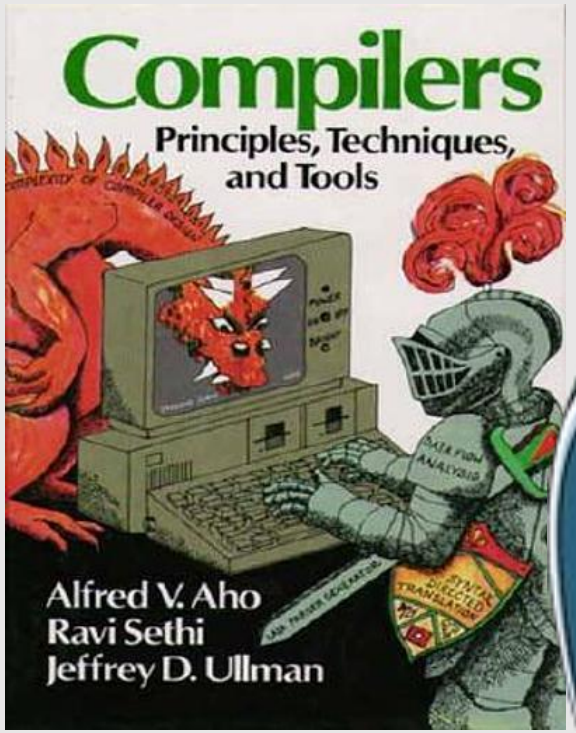
2024
Председатель программного комитета EuroLLVM'24

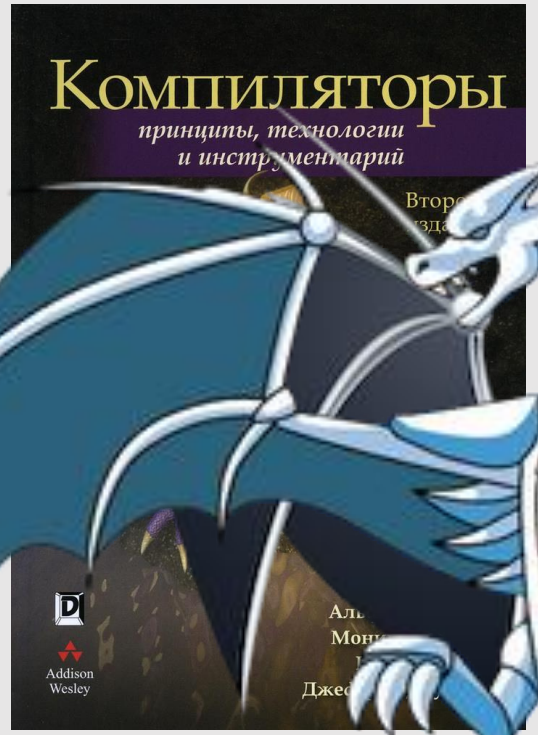
Что означает “LLVM”?

- Ничего! Это просто название.
  - "Despite its name, LLVM has little to do with traditional virtual machines. The name "LLVM" itself is not an acronym; it is the full name of the project."
- Собрание компонент для создания компиляторов и тулчейнов
  - “The LLVM Project is a collection of modular and reusable compiler and toolchain technologies.”

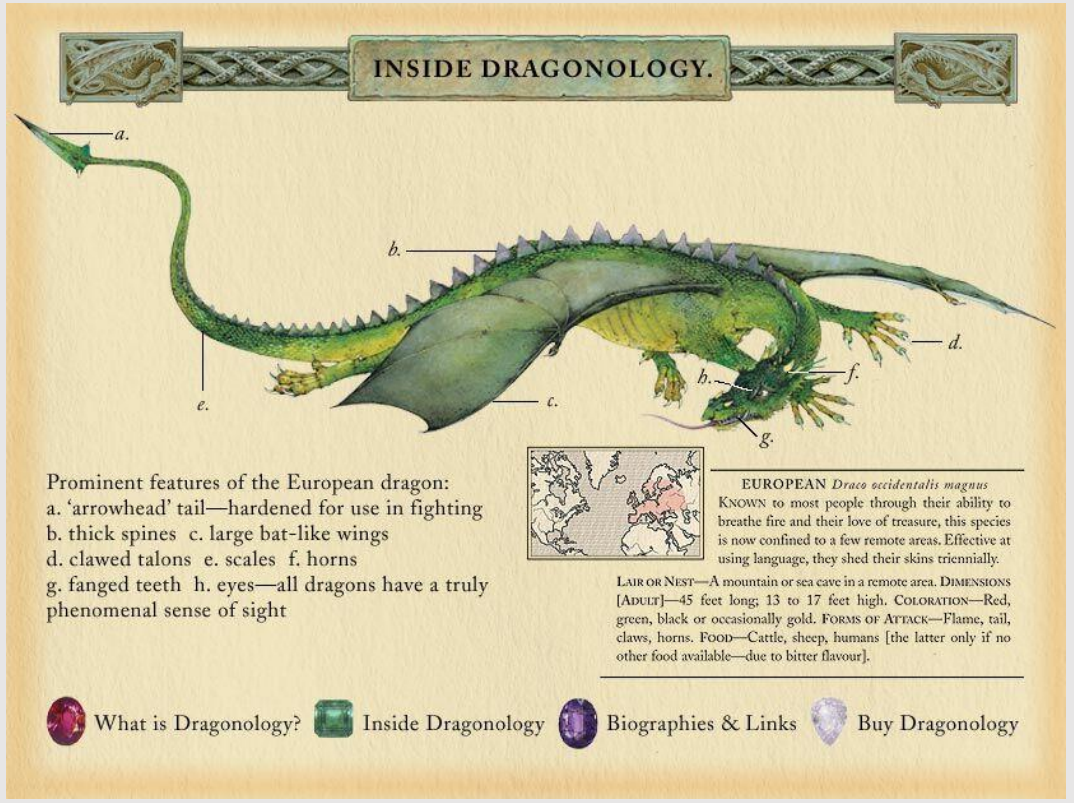


При чём здесь дракон?





История LLVM



Начало

LLVM: AN INFRASTRUCTURE FOR MULTI-STAGE OPTIMIZATION

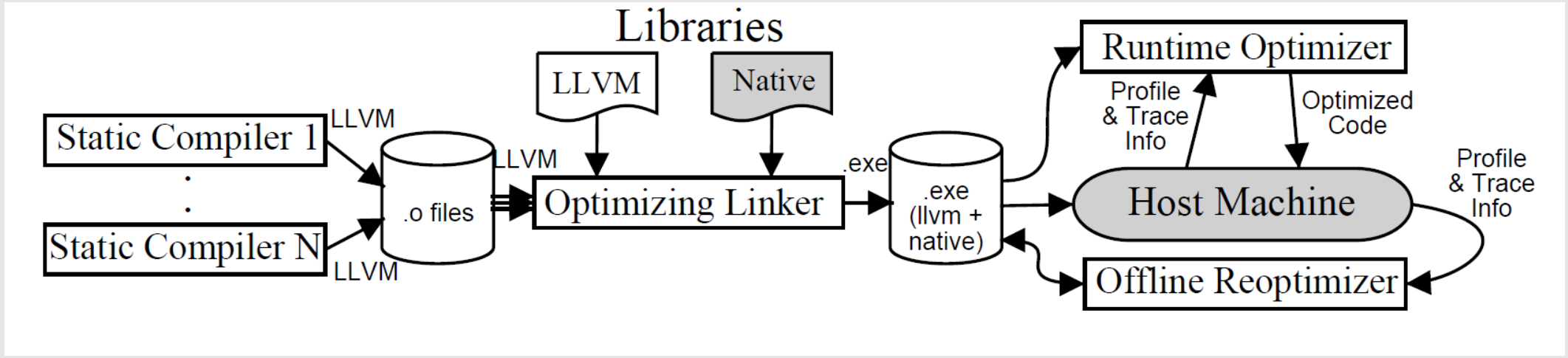
BY

CHRIS ARTHUR LATTNER

B.S., University of Portland, 2000

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2002



Apple

- В 2005 году Крис Латтнер начал работать в Apple
- LLVM стал статическим компилятором
- Почему Apple выбрала LLVM на замену GCC?
  - Лицензия (Apple не нравится GPLv3)
  - Полный контроль
  - Печальный опыт с компиляторами C++ для PowerPC

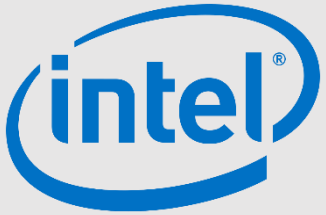
Clang

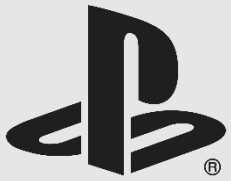
- К-ланг, а не Си-ланг!
- Проект начат в 2006 году Крисом Латтнером, Дагом Грегором и другими в Apple
  - Компилятор для “семейства языков C”: C, C++ и Objective-C
  - Изменения в Xcode: gcc -> llvm-gcc4 -> clang
- Через 10 лет Clang “съел” бизнес C++ компиляторов

Google

- Google начала использовать LLVM в ~2007
- Почему Google выбрала LLVM на замену GCC?
  - Один компилятор для всех операционных систем (Windows, Linux, OS X)
  - Security (компиляторные диагностики и санитайзеры)
  - Автоматический рефакторинг
  - Влияние в коммьюнити
- Google добавила поддержку Windows
  - Всего для двух приложений: Chrome и Google Earth
- Android полностью перешёл с GCC на Clang в 2016
  - "It’s time to have just one compiler for Android. One that can help find (and mitigate) security problems."

Все остальные





Секрет успеха

Permissive License

(разрешительная лицензия?)

- Главная причина успеха
  - LLVM использует лицензию Apache 2.0
  - GCC использует GPLv3
  - Все боятся “GCC вируса”!

- Закон Андрея Боханко
  - Число основных проектов в каждой категории ПО = числу основных типов open-source лицензий

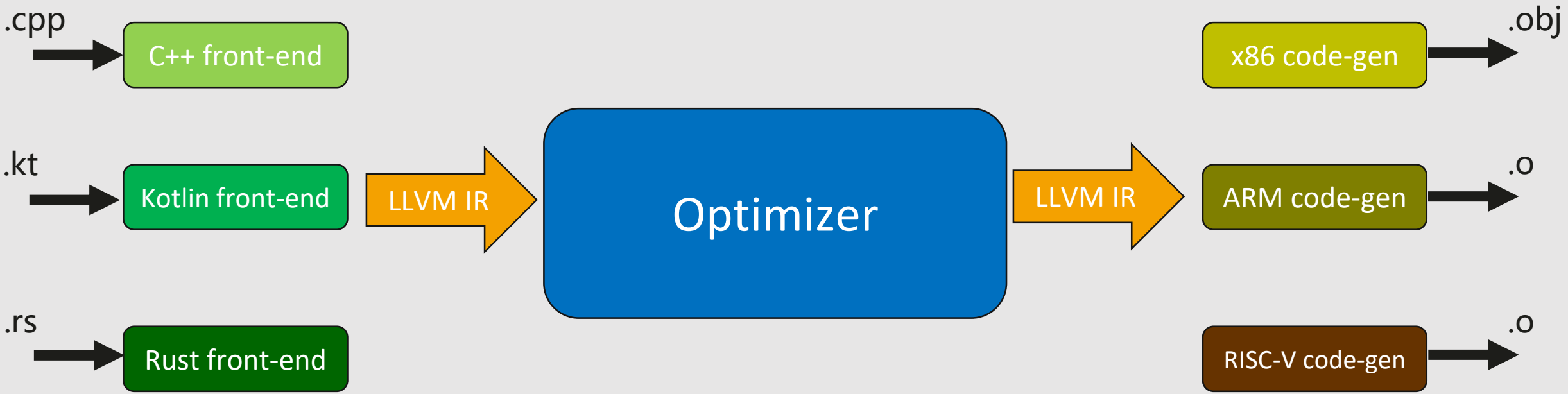
Benevolent Dictator

(доброжелательный диктатор?)

Модульная структура

LLVM – коллекция переиспользуемых библиотек

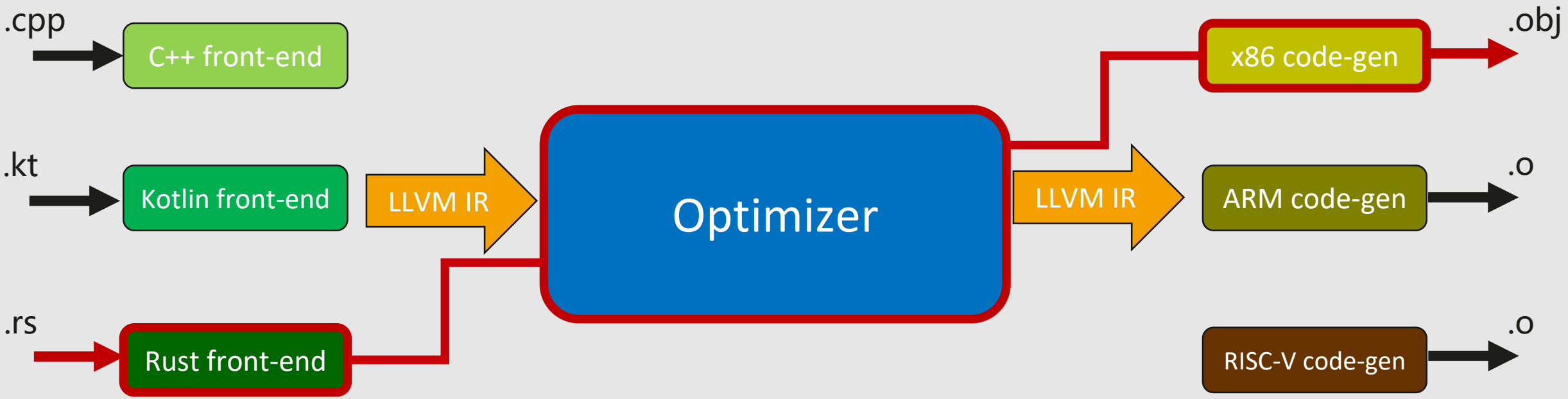
“Клей” между ними – LLVM IR



Модульная структура

LLVM – коллекция переиспользуемых библиотек

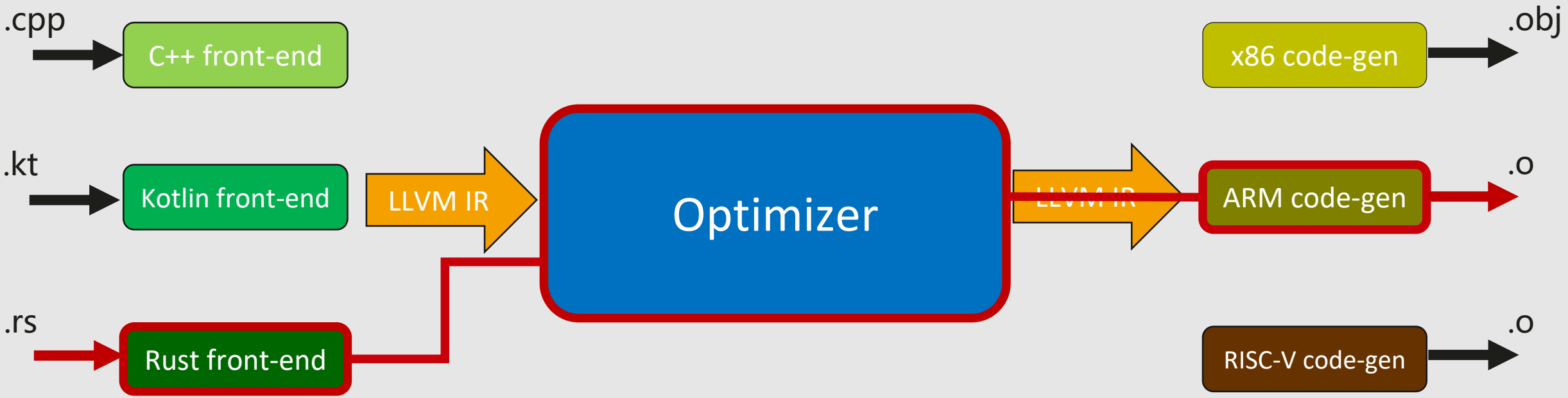
“Клей” между ними – LLVM IR



Модульная структура

LLVM – коллекция переиспользуемых библиотек

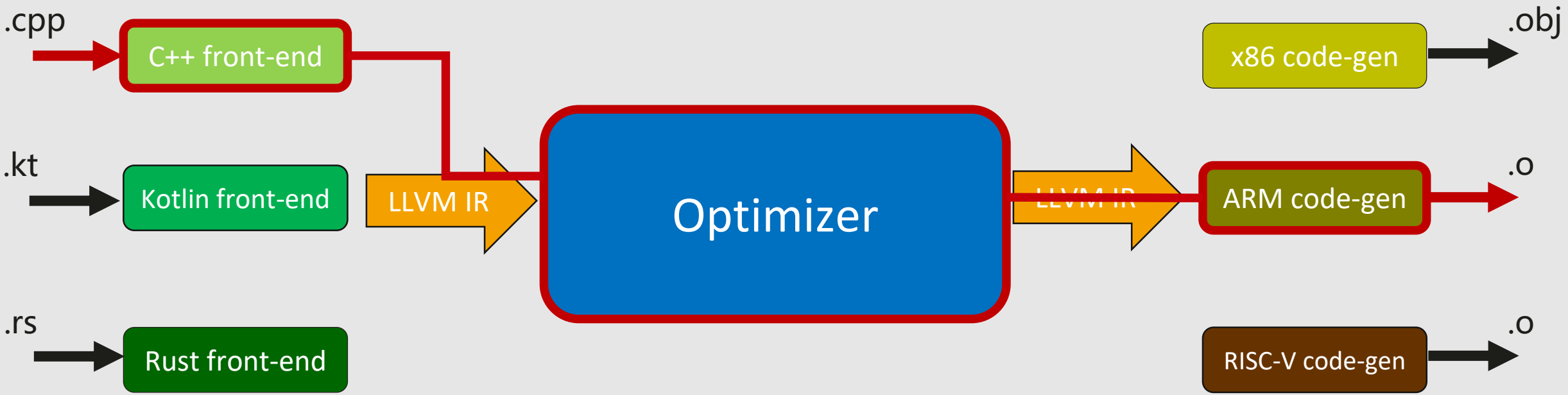
“Клей” между ними – LLVM IR



Модульная структура

LLVM – коллекция переиспользуемых библиотек

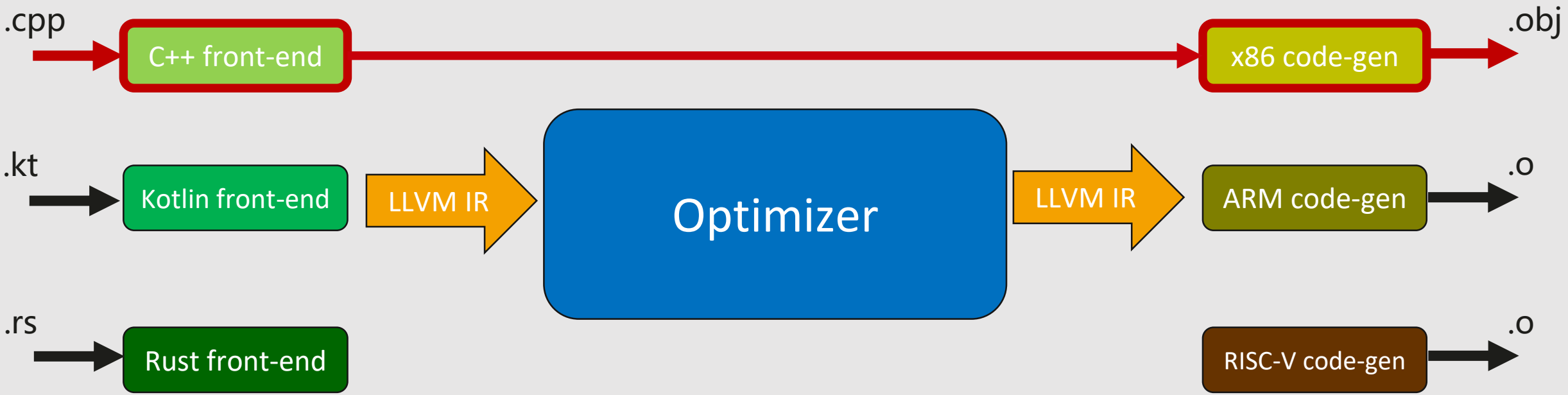
“Клей” между ними – LLVM IR



Модульная структура

LLVM – коллекция переиспользуемых библиотек

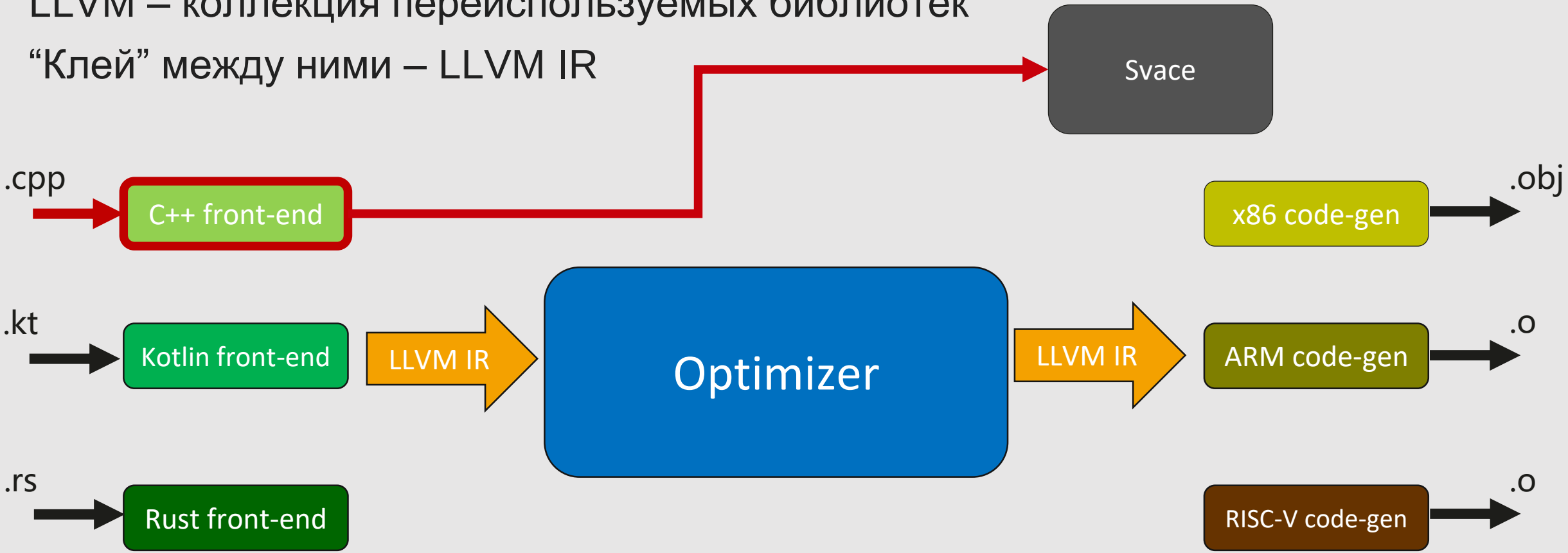
“Клей” между ними – LLVM IR



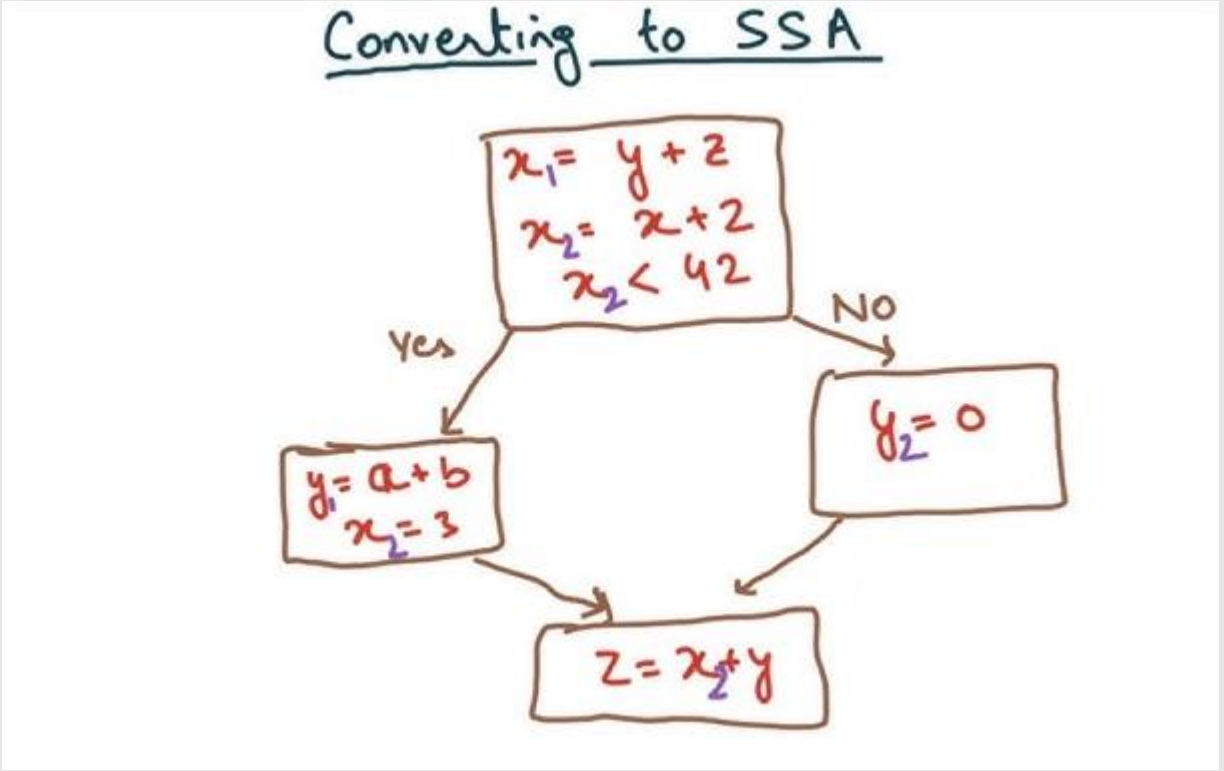
Модульная структура

LLVM – коллекция переиспользуемых библиотек

“Клей” между ними – LLVM IR



LLVM IR



“Hello World” на C

```
; ModuleID = 'hello.c'
source_filename = "hello.c"
target datalayout = "e-m:e-i64:64-f80:128-n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"

@.str = private unnamed_addr constant [14 x i8] c"Hello World!\0A\00", align 1

; Function Attrs: noinline nounwind optnone uwtable
define i32 @main() #0 {
  %1 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([14 x i8],
                                   [14 x i8]* @.str, i32 0, i32 0))
  ret i32 0
}

declare i32 @printf(i8*, ...) #1

...

!llvm.module.flags = !{!0}
!llvm.ident = !{!1}

!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{!"clang version 6.0.0-1ubuntu2 (tags/RELEASE_600/final)"}
```

“Hello World” на Kotlin

```
; Function Attrs: noinline
define void @"kfun:#main(){}"() #12 {
prologue:
  br label %locals_init

locals_init:                                      ; preds = %prologue
  %0 = phi %struct.ObjHeader** [ null, %prologue ]
  br label %stack_locals_init

stack_locals_init:                                ; preds = %locals_init
  call void @Kotlin_mm_safePointFunctionPrologue() #13
  br label %entry

entry:                                            ; preds = %stack_locals_init
  call void @"kfun:kotlin.io#println(kotlin.Any?){}"(%struct.ObjHeader* bitcast ({ %struct.
ArrayHeader, [13 x i16] }* @812 to %struct.ObjHeader*))
  br label %call_success

call_success:                                     ; preds = %entry
  br label %epilogue

epilogue:                                         ; preds = %call_success
  ret void
}
```

LLVM IR

- “Клей” всего проекта LLVM
- Обратная совместимость
- Независимость от языка
- Независимость от платформы
- Лингва франка?

MLIR

- Machine Learning IR?
- Multi Language IR?
- Mojo Language IR?
- Moore’s Law IR?
- Multi-Level IR?
- …

```
func @main(%arg0: tensor<i1>, %arg1: tensor<1xf32>, %arg2: tensor<1xf32>) -> tensor<1xf32> {
  %0 = "tfl.pseudo_input"(%arg0) : (tensor<i1>) -> tensor<i1>
  %1 = "tfl.pseudo_input"(%arg1) : (tensor<1xf32>) -> tensor<1xf32>
  %2 = "tfl.pseudo_input"(%arg2) : (tensor<1xf32>) -> tensor<1xf32>
  %3 = "tf.If"(%0, %1, %2) {
    else_branch = @testIfElse, then_branch = @testIfThen
  } : (tensor<i1>, tensor<1xf32>, tensor<1xf32>) -> tensor<1xf32>
  return %1 : tensor<1xf32>
}
```





“Зонтик” LLVM

LLVM это... компиляторы

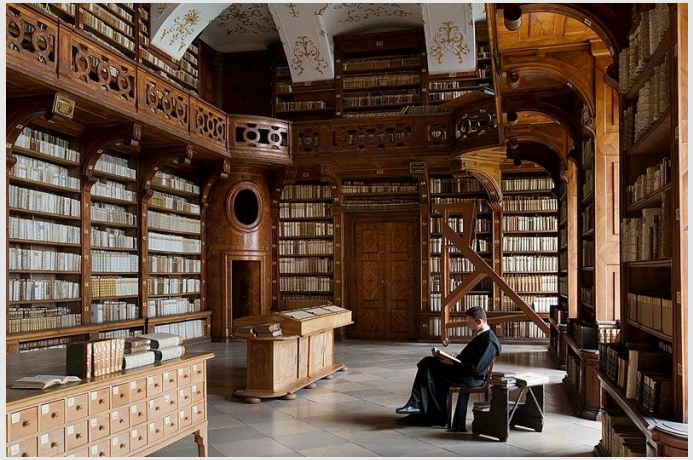
LLVM это… библиотеки

C++ Standard Library (libc++)

C Standard Library (llvm-libc)

OpenMP Library (libomp)

Coverage-guided fuzz testing Library (libFuzzer)



LLVM это… инструменты

Clang Static Analyzer

Автозавершение (в VSCode, XCode, CLion, QtCreator)

Санитайзеры (Address, Memory, Thread, UB, …)

lld (линкер)

ldb (отладчик)

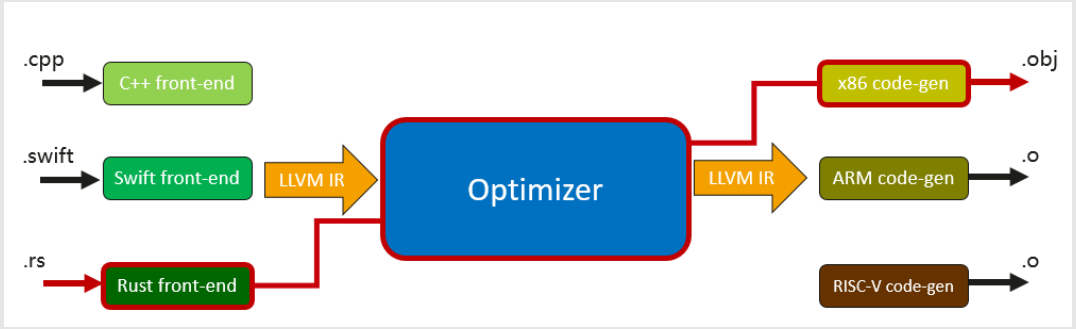
LLVM это... ML



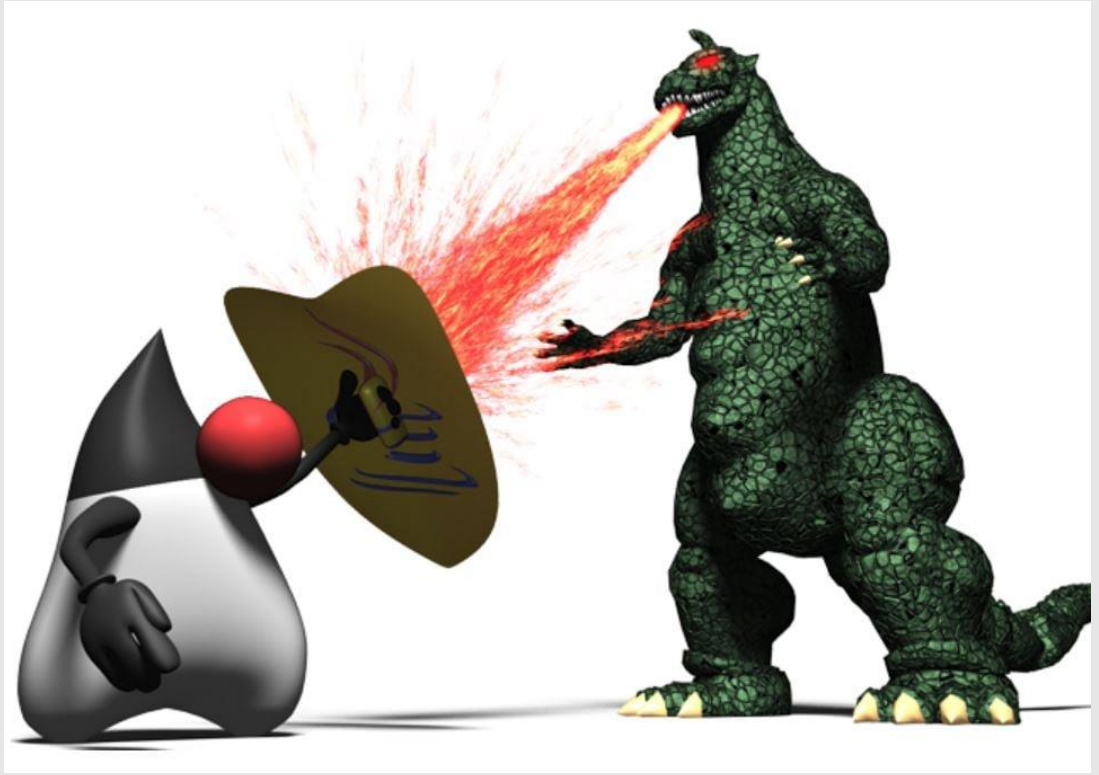


Почему?


WHAT'S IN THE SACK?
LETTERS OF SUPPORT FROM MY FANS, AND FANS OF THE FSF
OH. WELL I GET LETTERS OF SUPPORT TOO YOU KNOW. AND THE OSI.
OH YES?
LOOK, I HAVE SOME EXAMPLES HERE IN MY WALLET...
THIS ONE IS FROM SUN MICROSYSTEMS. THIS IS FROM VA LINUX.
THOSE ARE HUNDRED DOLLAR BILLS, ERIC.
OOH, THIS DOOSEY IS FROM NETSCAPE!


.cpp
C++ front-end
.swift
Swift front-end
.rs
Rust front-end
LLVM IR
Optimizer
LLVM IR
x86 code-gen
ARM code-gen
RISC-V code-gen
.obj
.o
.o

Где же Java?!



Компилятор для JDK

https://openjdk.org/groups/build/doc/building.html

Native Compiler (Toolchain) Requirements

Large portions of the JDK consists of native code, that needs to be compiled to be able to run on the target platform. In theory, toolchain and operating system should be independent factors, but in practice there's more or less a one-to-one correlation between target operating system and toolchain.

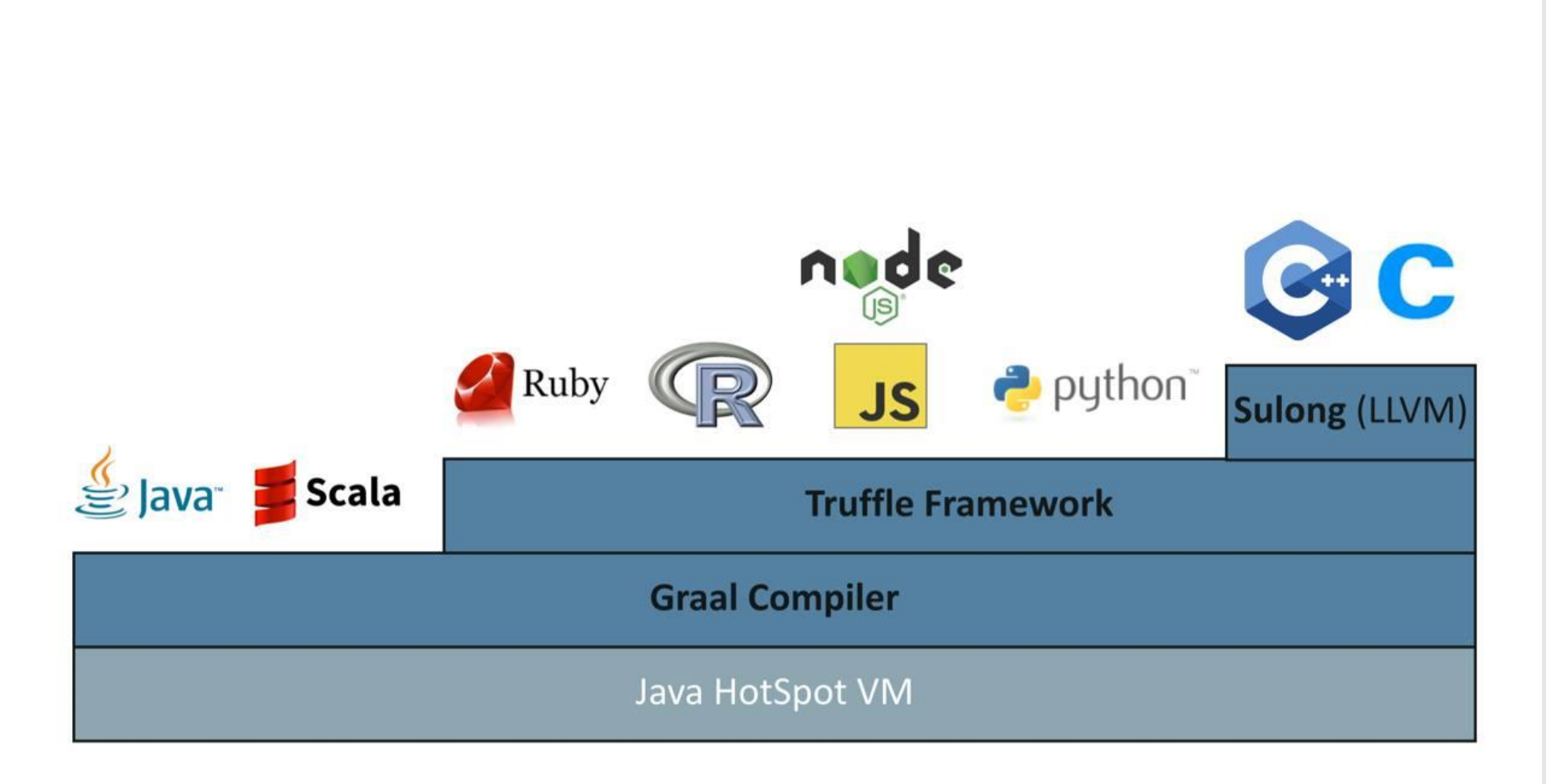
Operating system	Supported toolchain
Linux	gcc, clang
macOS	Apple Xcode (using clang)
AIX	IBM Open XL C/C++
Windows	Microsoft Visual Studio

Please see the individual sections on the toolchains for version recommendations. As a reference, these versions of the toolchains are used, at the time of writing, by Oracle for the daily builds of the JDK. It should be possible to compile the JDK with both older and newer versions, but the closer you stay to this list, the more likely you are to compile successfully without issues.

Operating system	Toolchain version
Linux	gcc 13.2.0
macOS	Apple Xcode 14.3.1 (using clang 14.0.3)
Windows	Microsoft Visual Studio 2022 version 17.6.5

All compilers are expected to be able to handle the C11 language standard for C, and C++14 for C++.

GraalVM


node
C++
C
Ruby
R
JS
python
Sulong (LLVM)
Java
Scala
Truffle Framework
Graal Compiler
Java HotSpot VM

Azul “Falcon” JIT

LLVM – The Technology Behind Falcon

The Falcon JIT compiler is based on technology from LLVM, the popular compiler infrastructure project with active involvement from leading universities and dozens of corporate contributors including Adobe, Apple, Google, NVIDIA, and Intel.

Today LLVM is used in a variety of programming languages and frameworks outside Java and the JVM, including newer platforms like Swift and Rust as well as many others including C/C++, Objective-C, Swift, Clang, OpenCL, CUDA and many more.

LLVM is considered the state-of-the-art in compiler technology, and the LLVM community is continually incorporating new optimization techniques and support for new processor instruction sets and hardware platforms.

Kotlin Native Compiler

Kotlin/Native

Kotlin/Native is an LLVM backend for the Kotlin compiler, runtime implementation, and native code generation facility using the LLVM toolchain.

Kotlin/Native is primarily designed to allow compilation for platforms where virtual machines are not desirable or possible (such as iOS or embedded targets), or where a developer is willing to produce a reasonably-sized self-contained program without the need to ship an additional execution runtime.

- LLVM повсюду и влияет на всех нас!
  - …и даже на Java
  - Есть смысл узнать про LLVM больше ✓
  - Давайте влиять на LLVM!
- Если вам нужно сделать новый язык или инструмент
  - …вы знаете ответ (и в нём четыре буквы)
- Нужна помощь?
  - andreybokhanko@gmail.com

Q&A