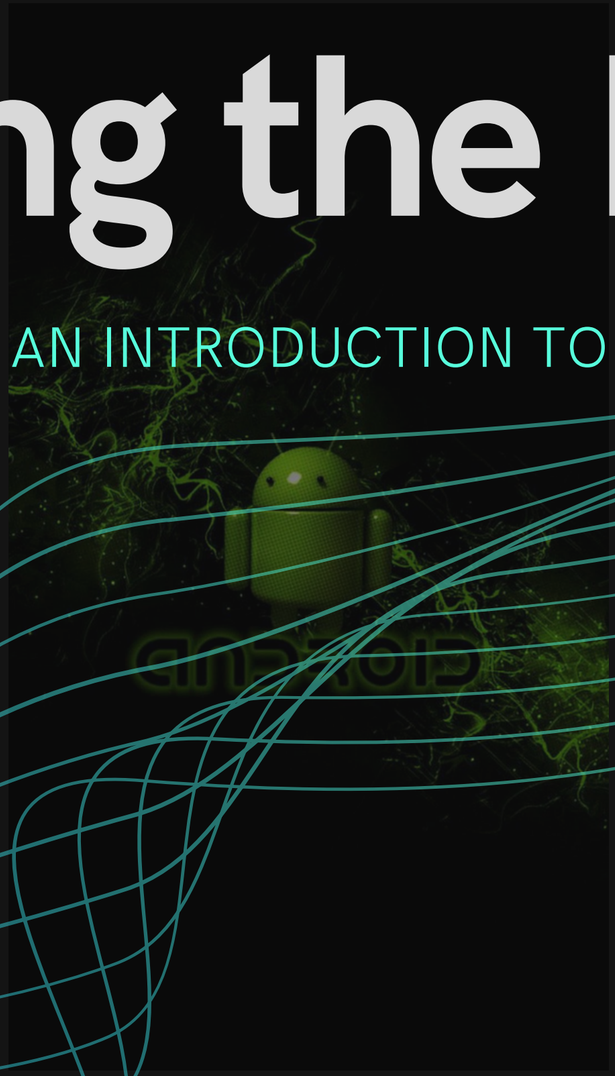
Dissecting the DEX Format

AN INTRODUCTION TO KUNAI LIB



Eduardo Blázquez

COMPILER ENGINEER - QUARKSLAB

- Almost PhD at University Carlos III of Madrid
- Developer of Kunai Lib for DEX analysis
- Previously Researcher at UC3M-COSEC Lab

What will we see today?

ANDROID

Is there anyone who doesn’t know about it yet?

ANDROID APP EXECUTION

Brief introduction to the execution of an app.

APK & DEX

Presentation of the APKs and the Dalvik EXecutable format

KUNAI LIB

Introduction to Kunai Library

DEMO

Small demo of simple programs written with Kunai

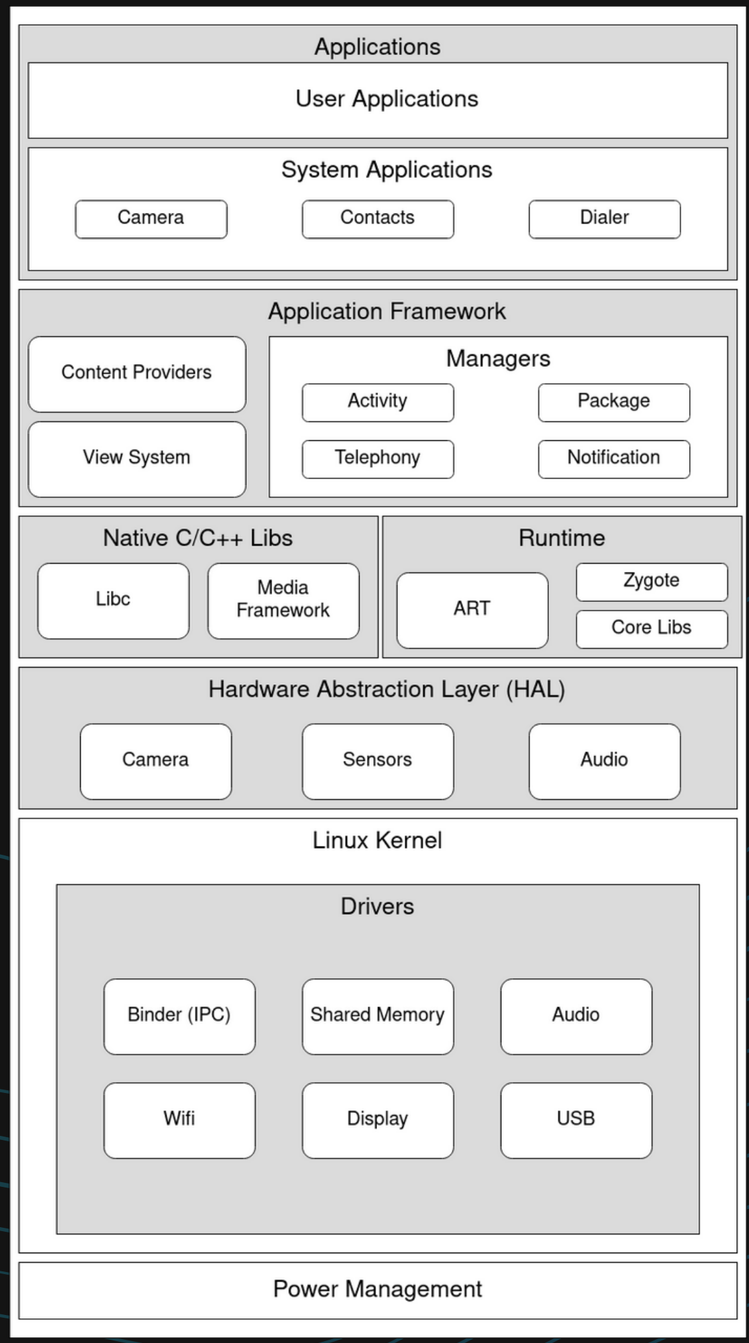
Android

- Operating System designed by Android inc., started as an OS for cameras, but quickly moved to smartphones
- Android inc. was acquired by Google inc. in 2005
- First version released in 23 September 2008
- Actually is the most used OS with more than 30% of market share
- The Android Open Source Project is available under Apache License

picture from: https://www.xatakandroid.com/sistema-operativo/todas-versiones-android-historia

Android “Internals”

- **Android is a Linux Kernel based OS**
- **Above a HAL that abstracts from the kernel drivers implementation**
- **Native libs used by the system, for example an implementation of libc called *Bionic***
- **A runtime system for the Android applications (ART)**
- **Java framework available to developers**
- **System/User applications**



Android app execution

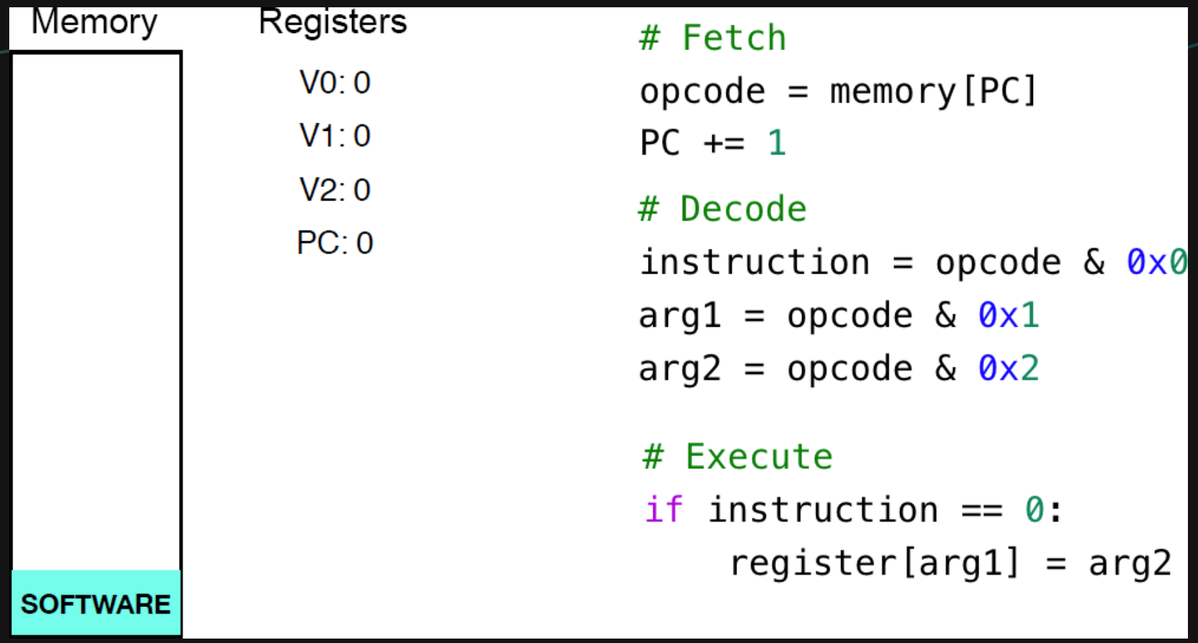
DALVIK VIRTUAL MACHINE (DVM)

“In the beginning was Dalvik, and Dalvik was with Android, and Dalvik was Android...”

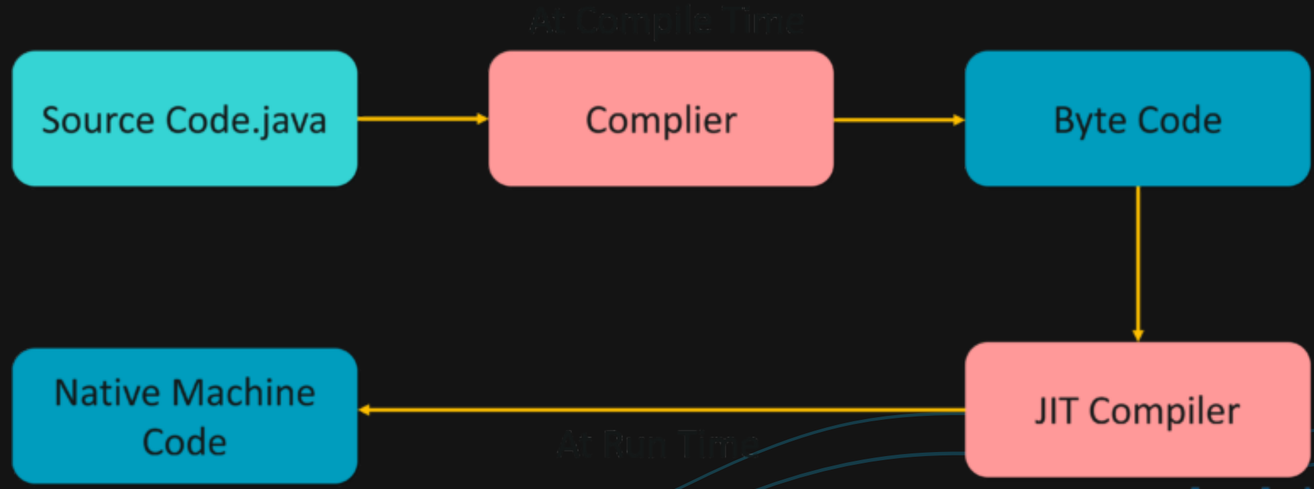
- **First runtime from Android**
- **Virtual Machine register based (in opposite to JVM)**
- **More than 200 bytecodes supported**
- **Support for JIT of DEX files (since Android 2.2 “Froyo”)**
- **Replaced by Android ART since Android Lollipop**

Android app execution

DALVIK VIRTUAL MACHINE (DVM)



Virtual Machine Execution

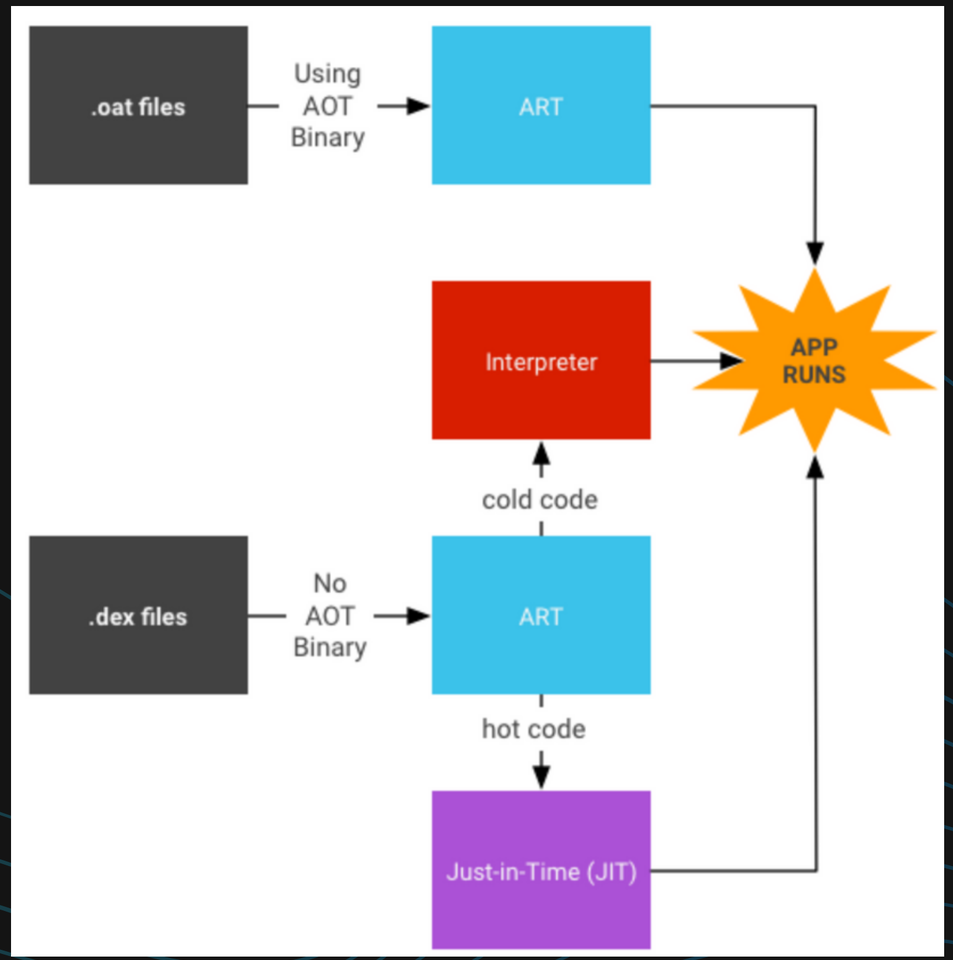


Just In Time Compilation

Android app execution

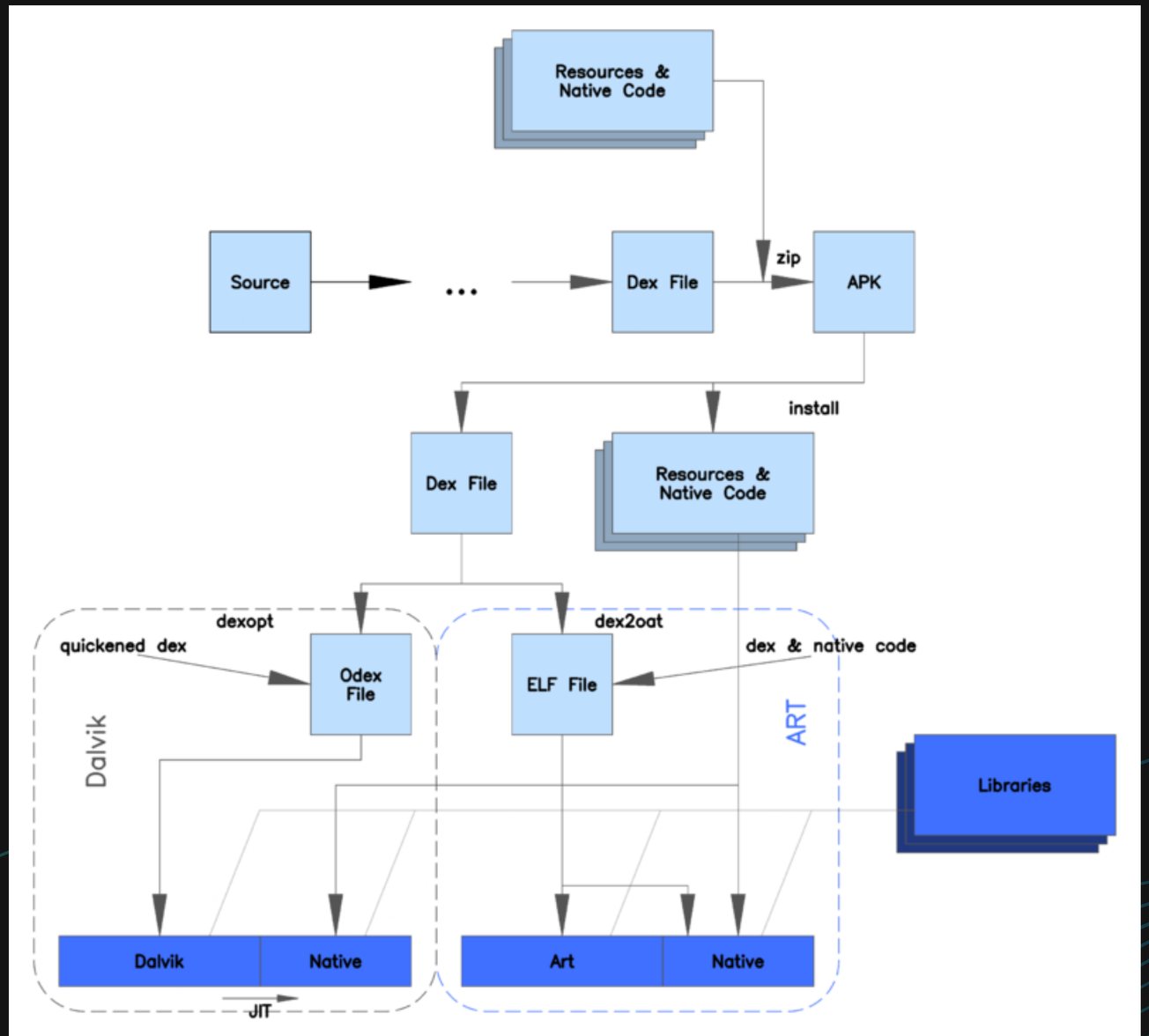
ANDROID RUNTIME (ART)

- **Runtime environment that replaced DVM-JIT in Android Lollipop**
- **Replace JIT with Ahead-of-time compilation**
- **Compilation of DEX at installation time**
- **It keeps the DEX interpreter and part of JIT**



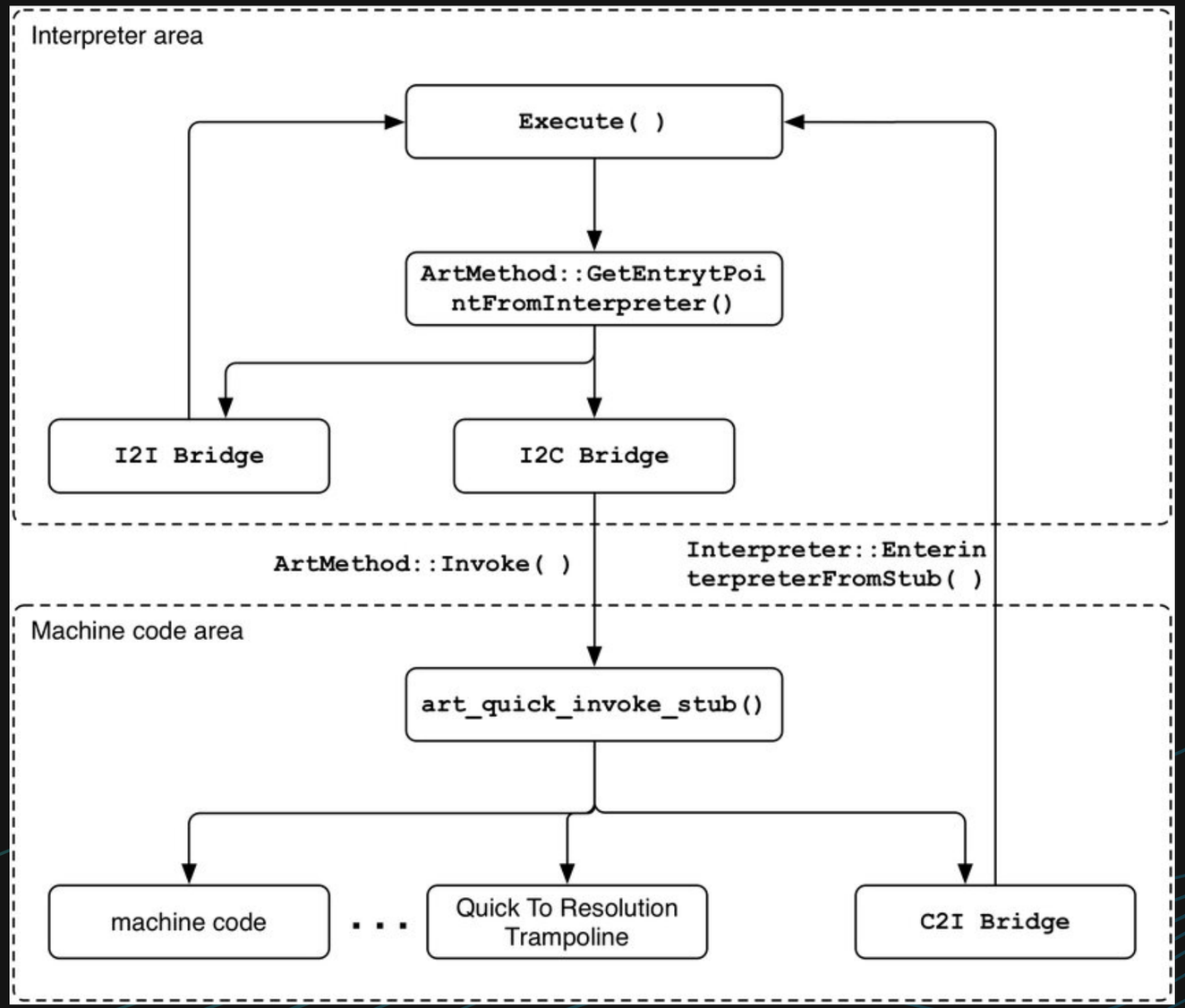
Android app execution

DALVIK VIRTUAL MACHINE VS ANDROID RUNTIME (ART)



Android app execution

ANDROID RUNTIME (ART) - NATIVE & INTERPRETER



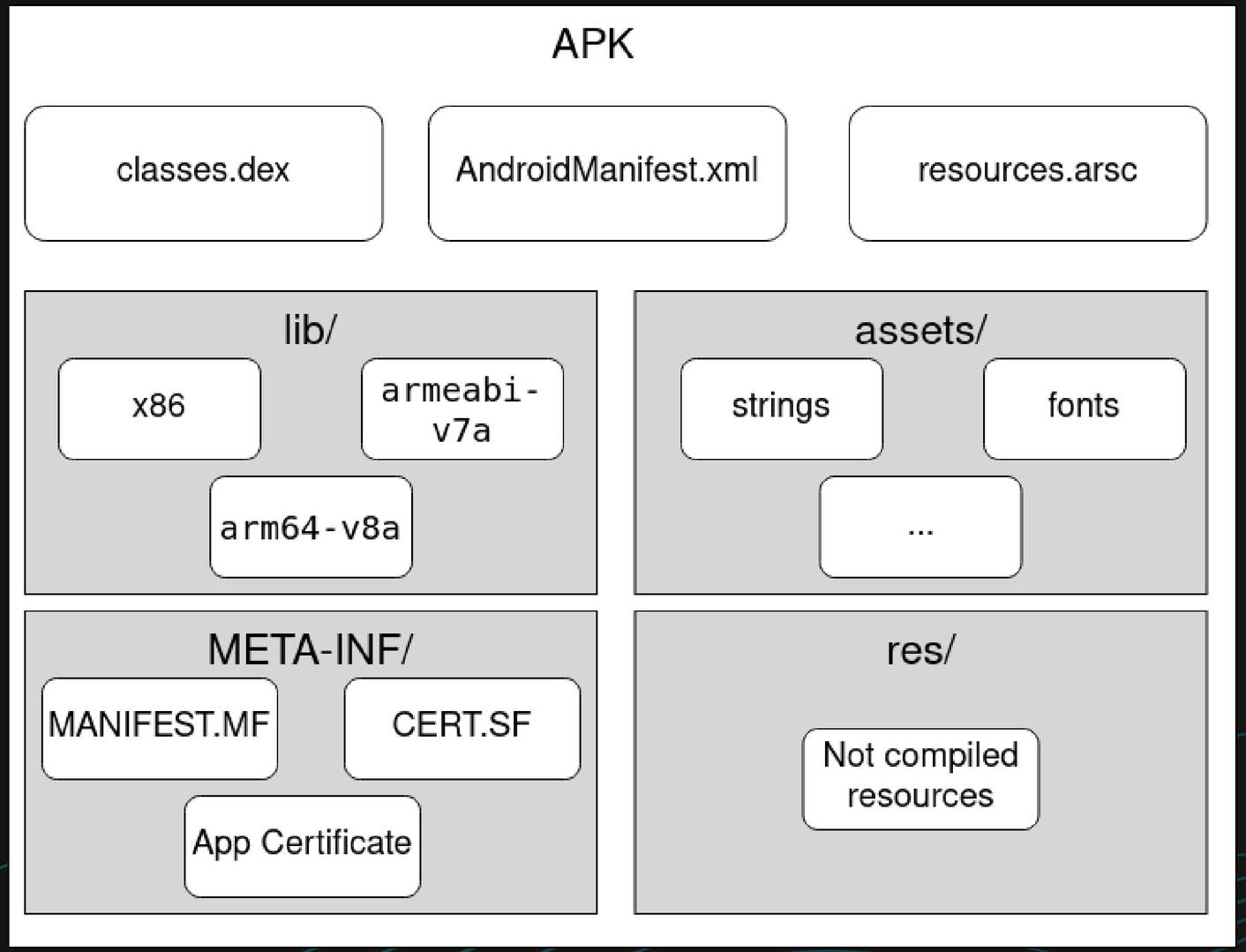
APK & DEX

ANDROID PACKAGE (APK)

- Archive file similar to Zip used to distribute applications
- Contains application’s code (dex and native), resources, assets, certificates, and manifest file
- AndroidManifest.xml specifying components from the app and used resources (permissions)
- To install it, you must sign with a developer certificate (no certificate authority, but google offers a signing service)
- If we want to check its content we can run the next command: `unzip -d <folder_name> <app_name>.apk`

APK & DEX

ANDROID PACKAGE (APK)



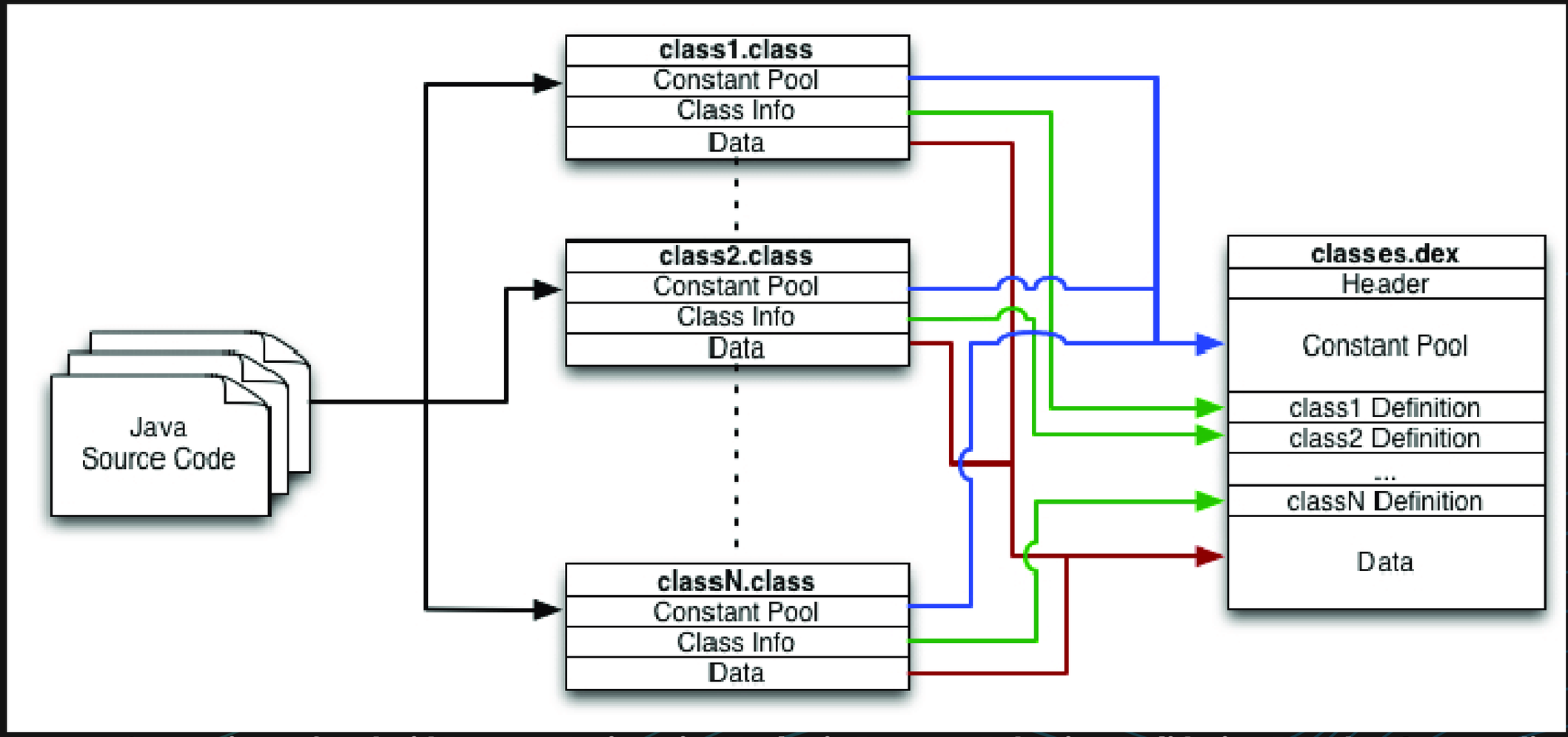
APK & DEX

DALVIK EXECUTABLE (DEX)

- File type that contains the structures for representing: classes, methods, fields, prototypes, types...
- It contains a big table with all the strings from dalvik code (including types, classes names, method names, etc).
- Each method contains structures for keeping the information about try/catch, annotations, even its own bytecode (easy to disassemble :D)
- The format is explained at the Android website, as well as other resources (e.g. at Lief website).

APK & DEX

DALVIK EXECUTABLE (DEX)



Source: A Comparison of Android Reverse Engineering Tools via Program Behaviors Validation Based on Intermediate Languages Transformation

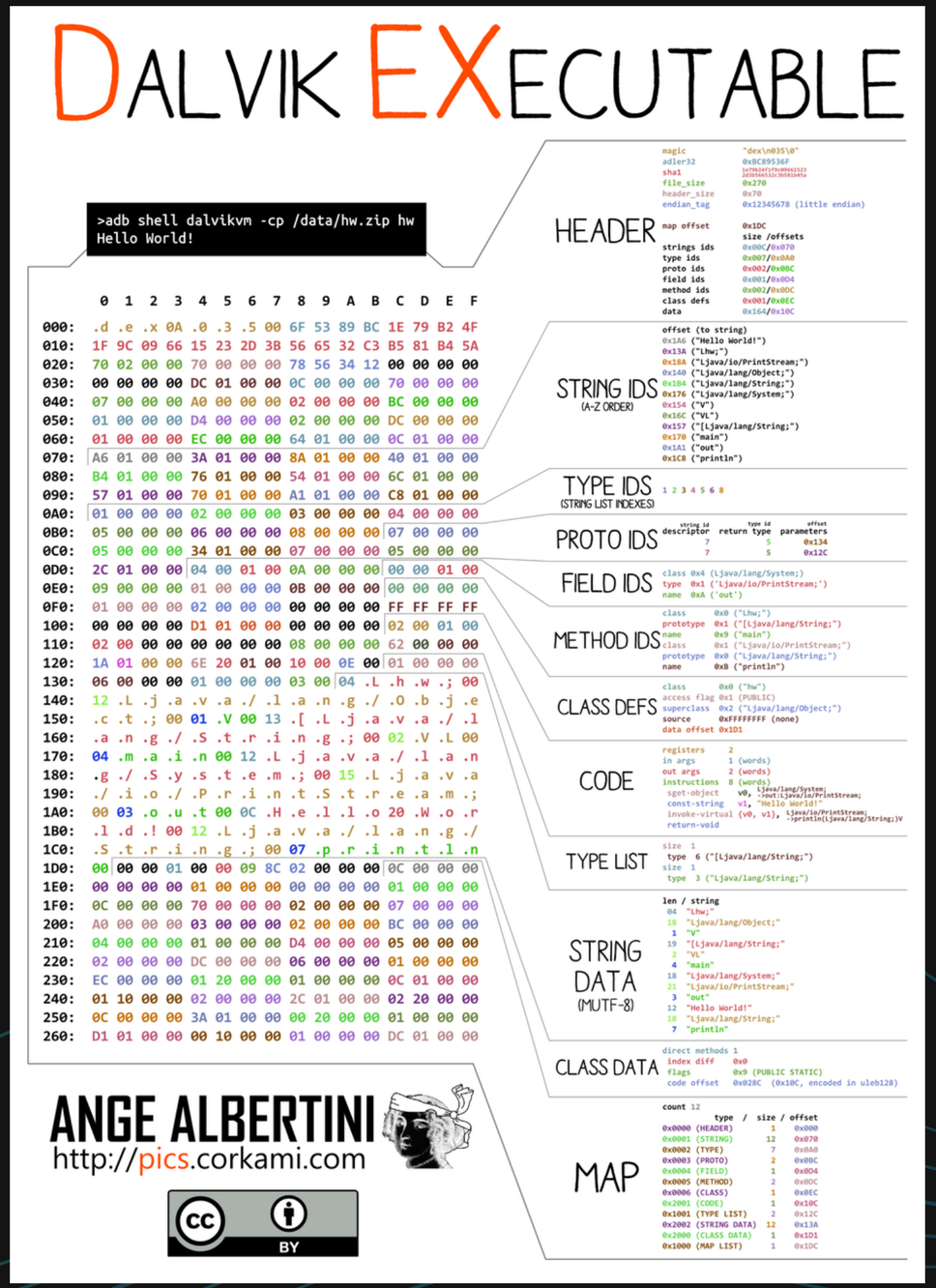
APK & DEX

DALVIK EXECUTABLE (DEX)

DEX,
WHAT PEOPLE THINK IT IS

APK & DEX

DALVIK EXECUTABLE (DEX)



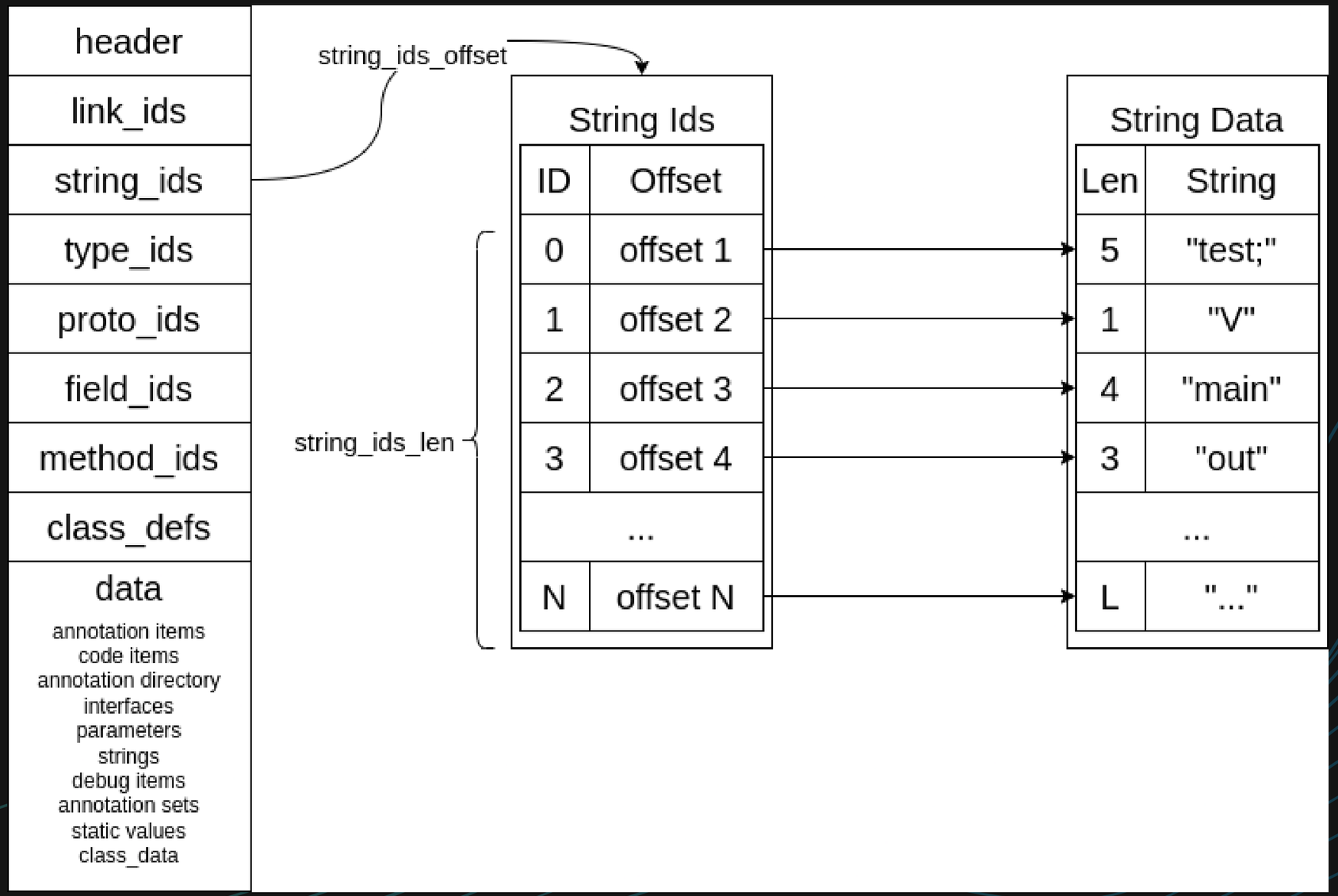
APK & DEX

DALVIK EXECUTABLE (DEX)

DEX,
WHAT IT REALLY IS

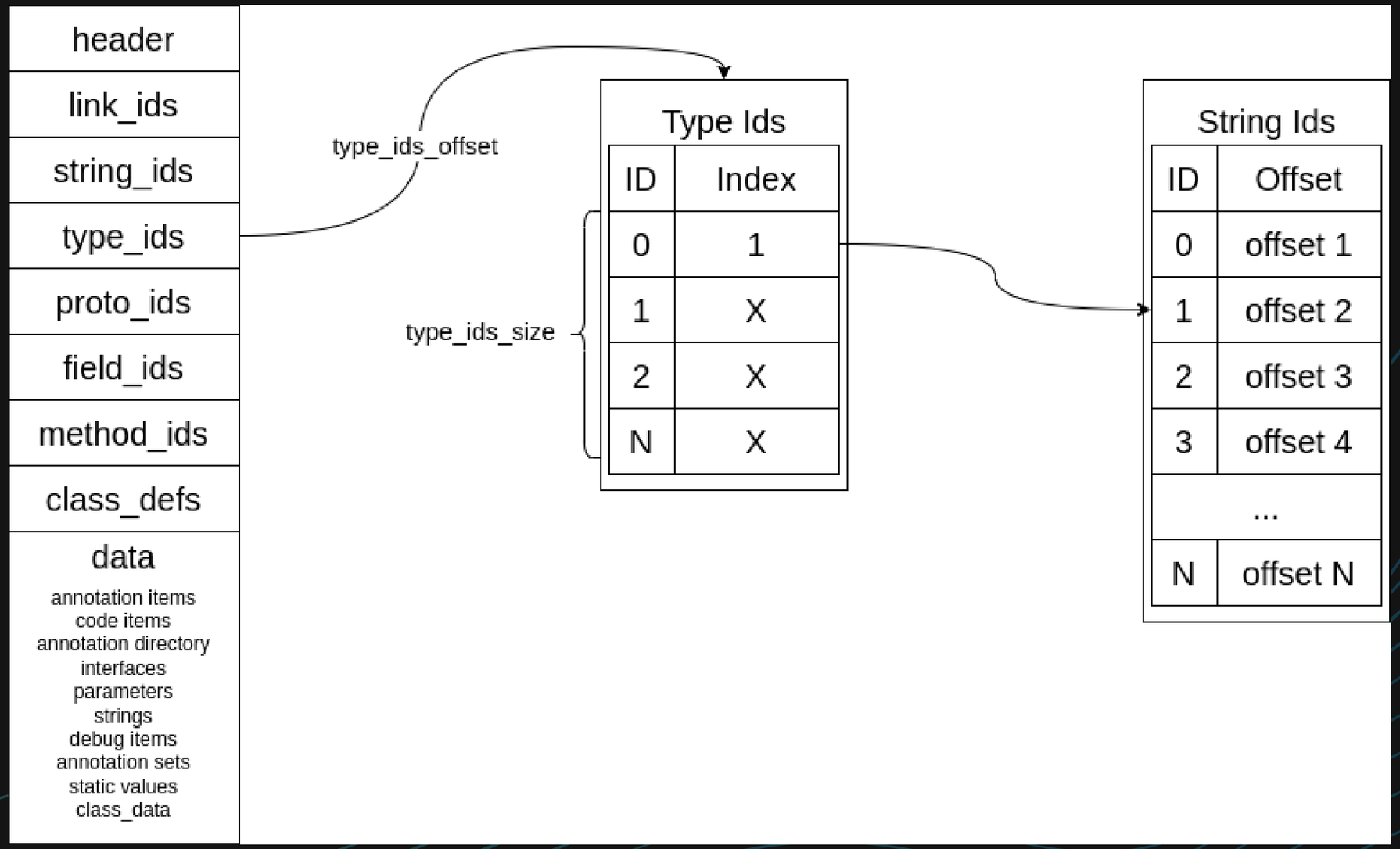
APK & DEX

DALVIK EXECUTABLE (DEX)



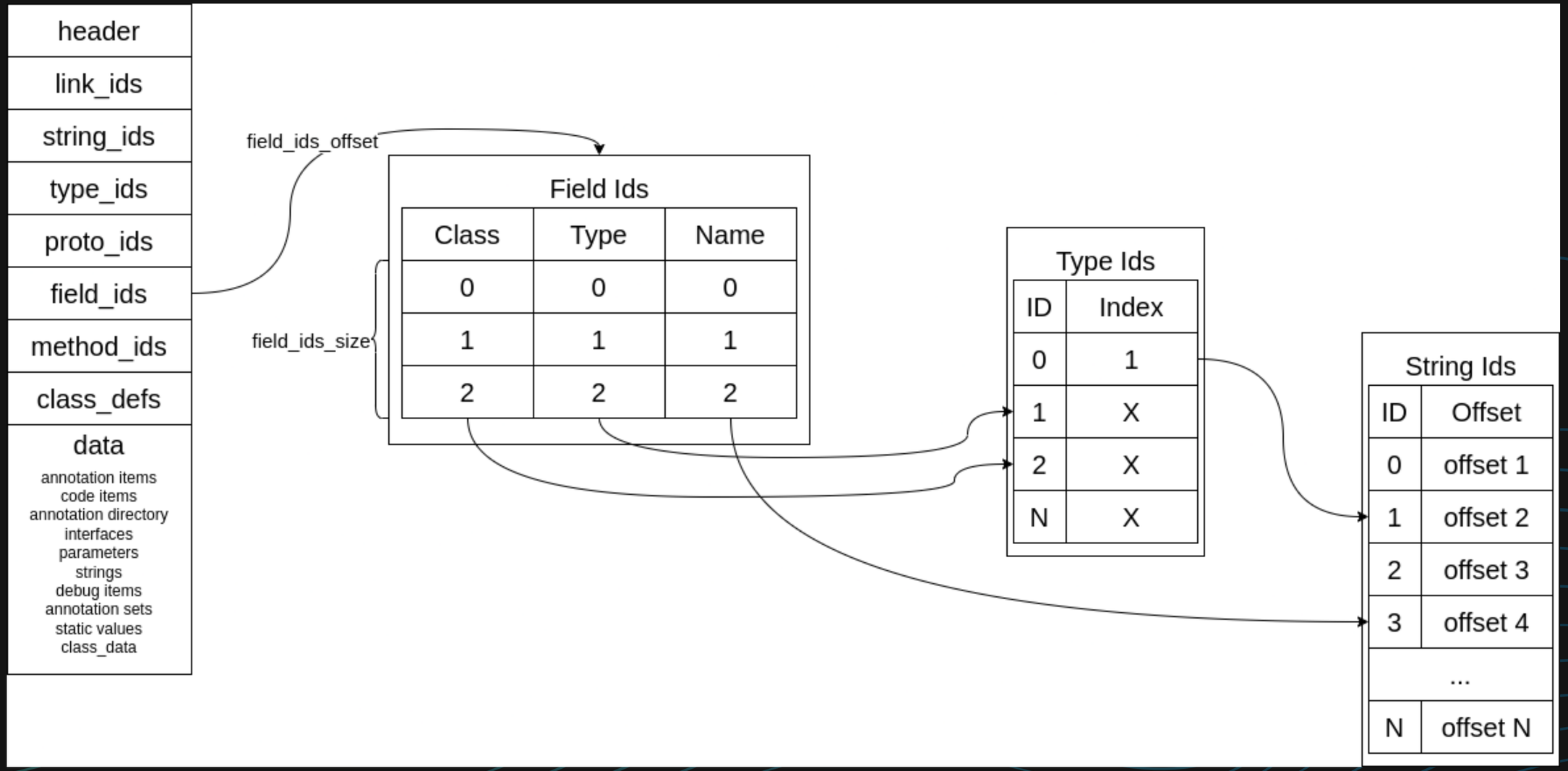
APK & DEX

DALVIK EXECUTABLE (DEX)



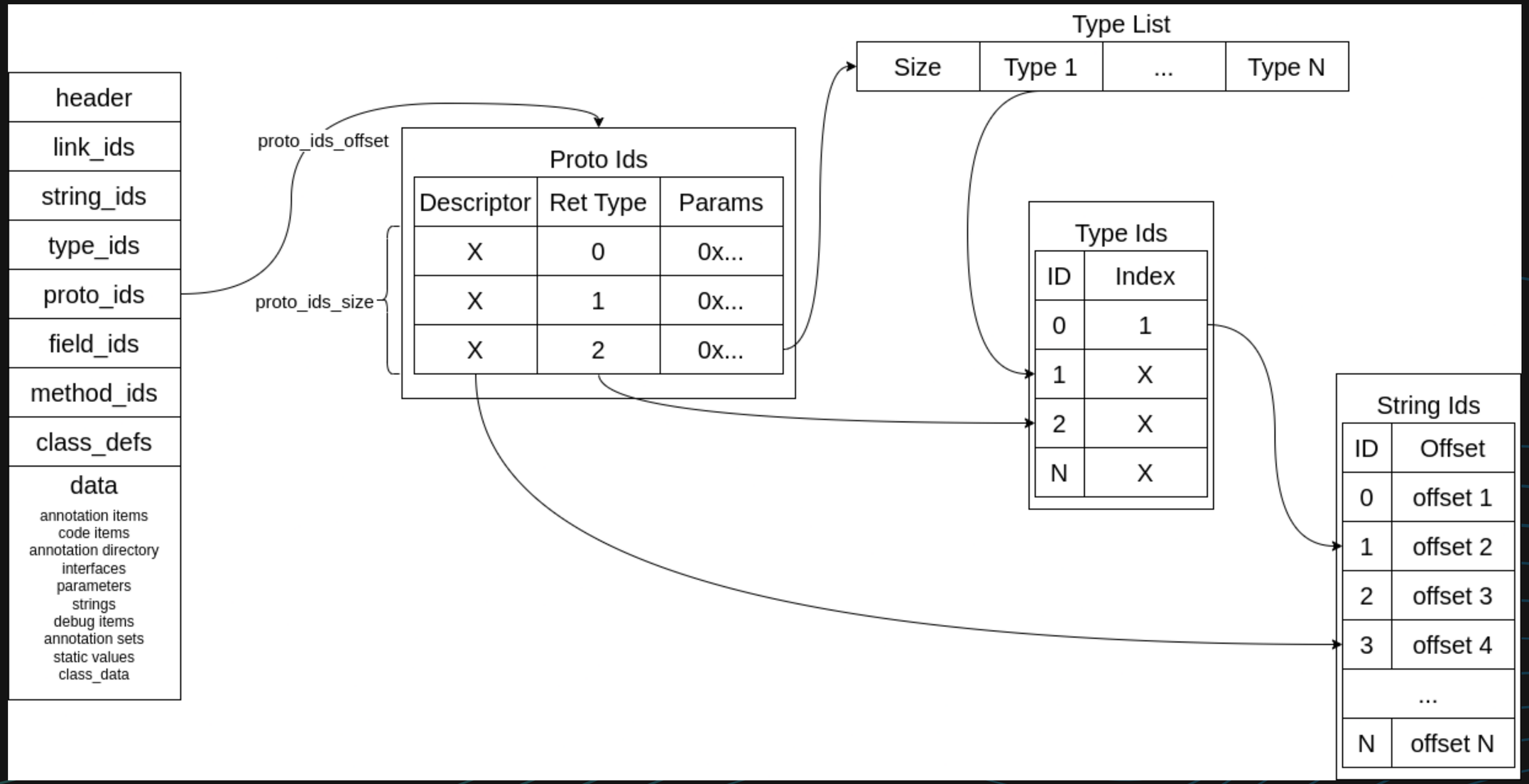
APK & DEX

DALVIK EXECUTABLE (DEX)



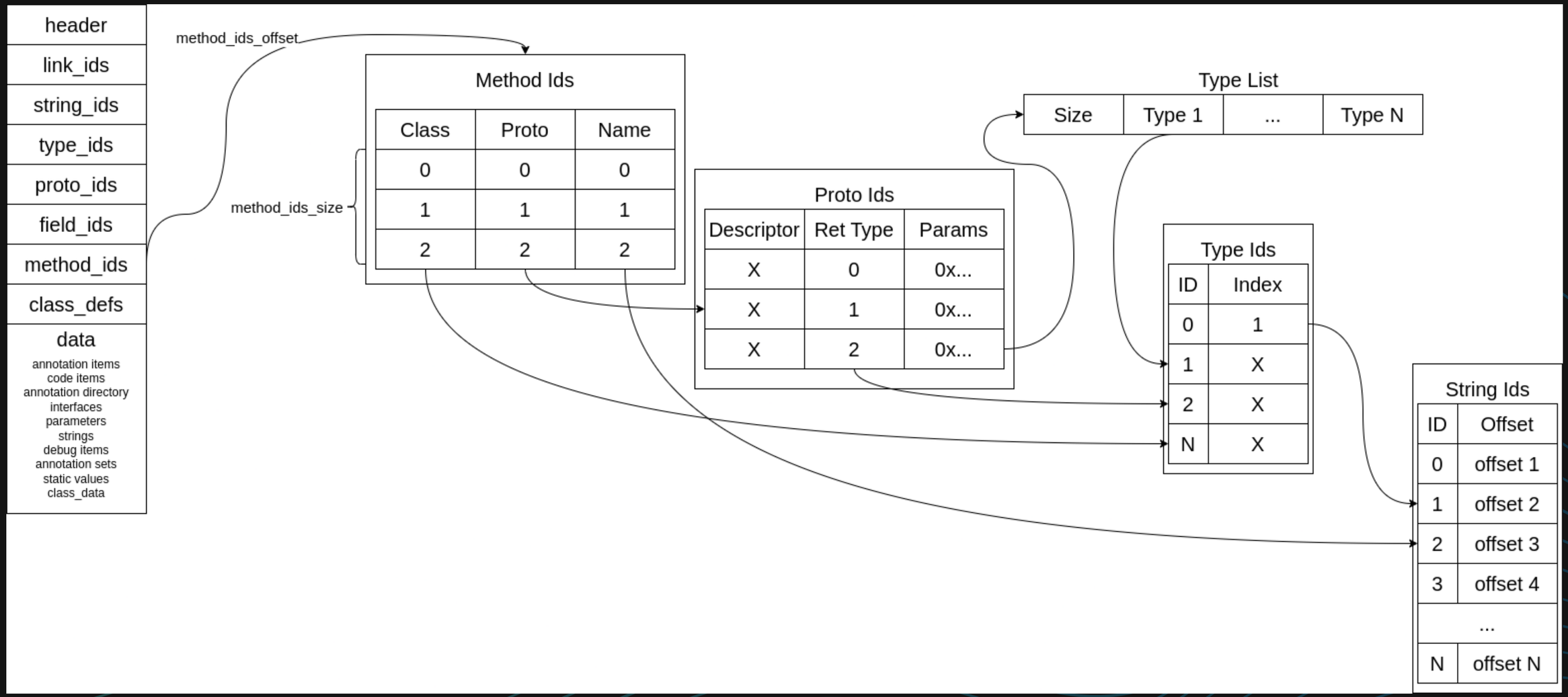
APK & DEX

DALVIK EXECUTABLE (DEX)



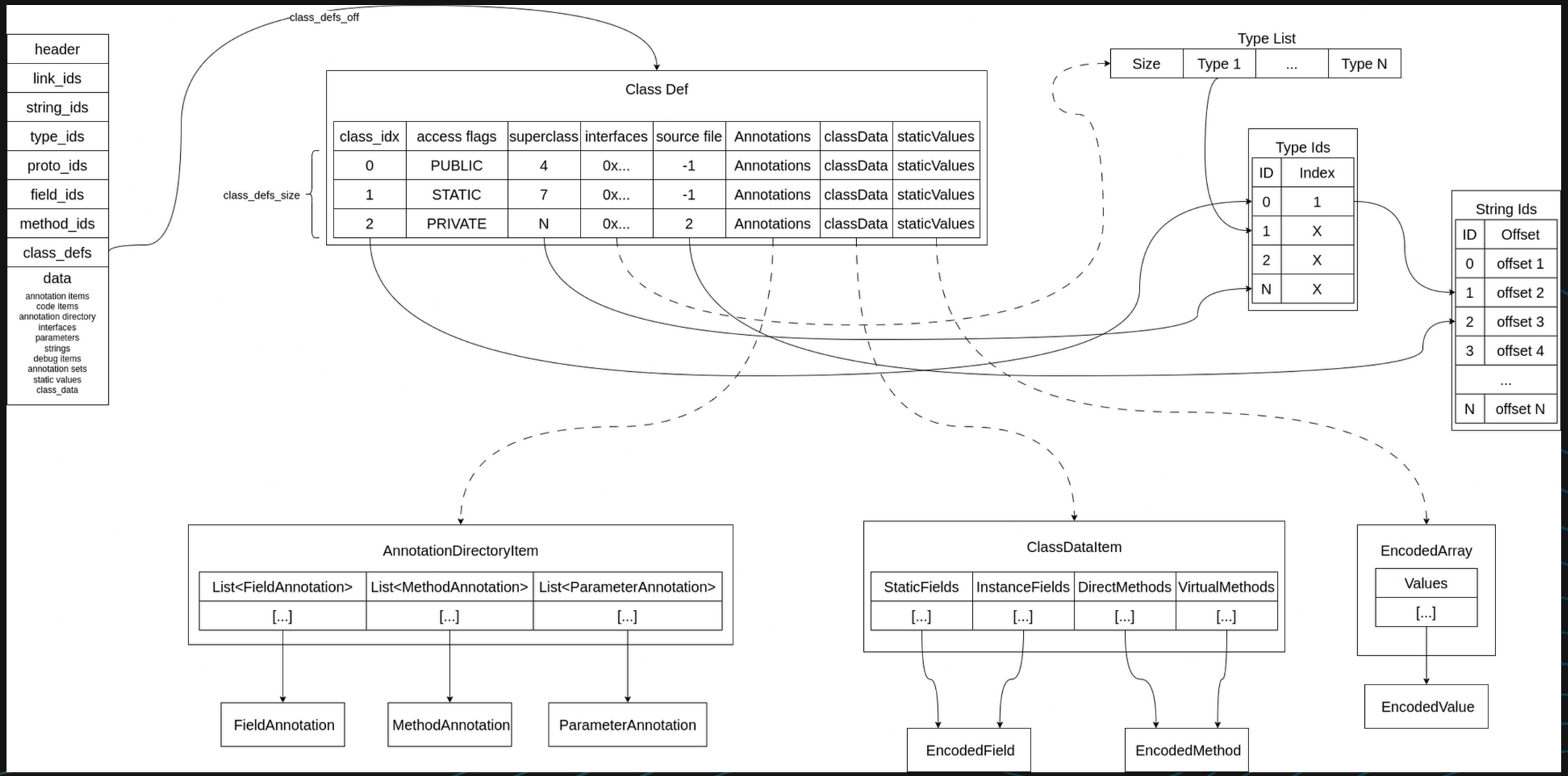
APK & DEX

DALVIK EXECUTABLE (DEX)



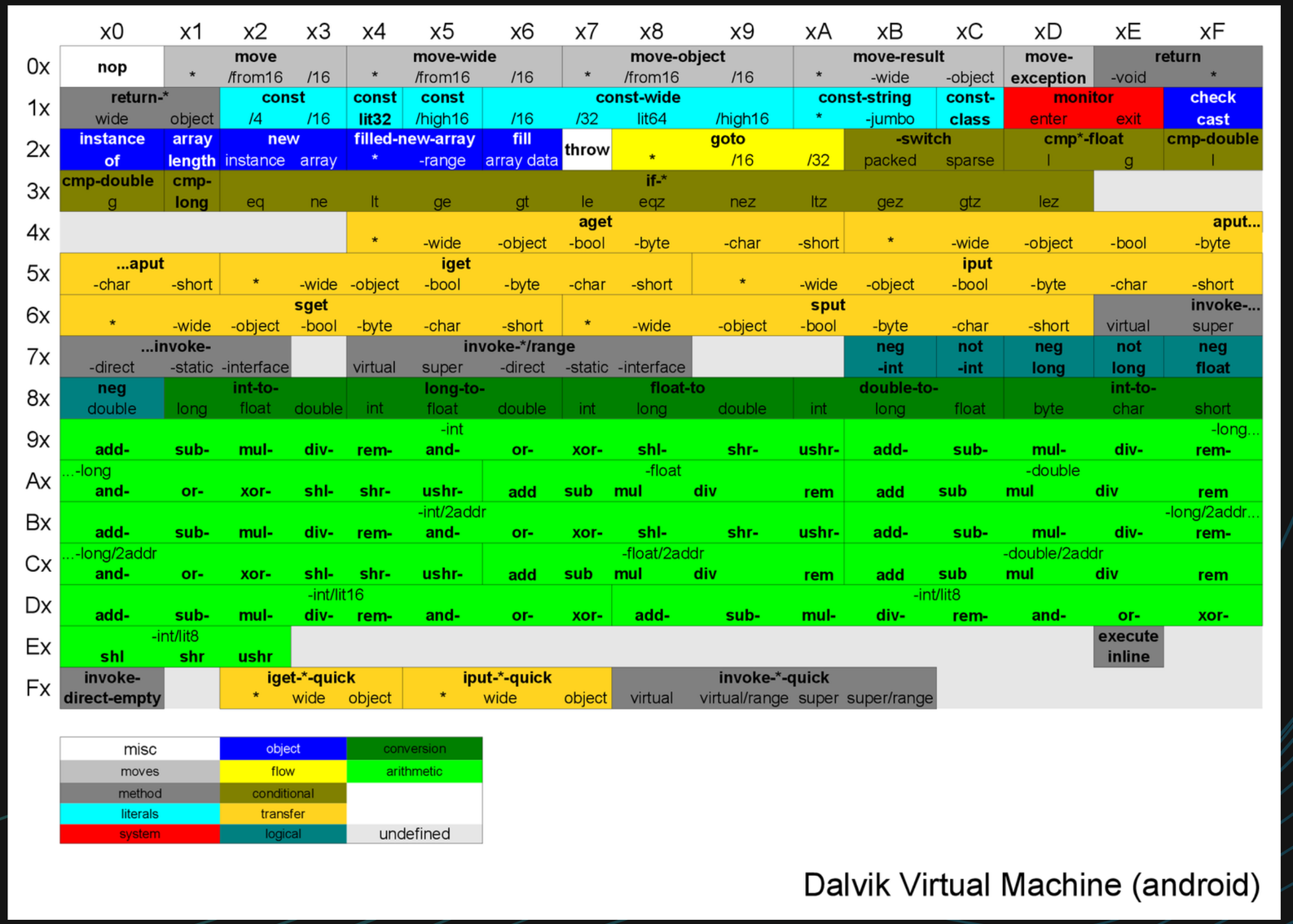
APK & DEX

DALVIK EXECUTABLE (DEX)



APK & DEX

DALVIK EXECUTABLE (DEX) - BYTECODE



Dalvik Virtual Machine (android)

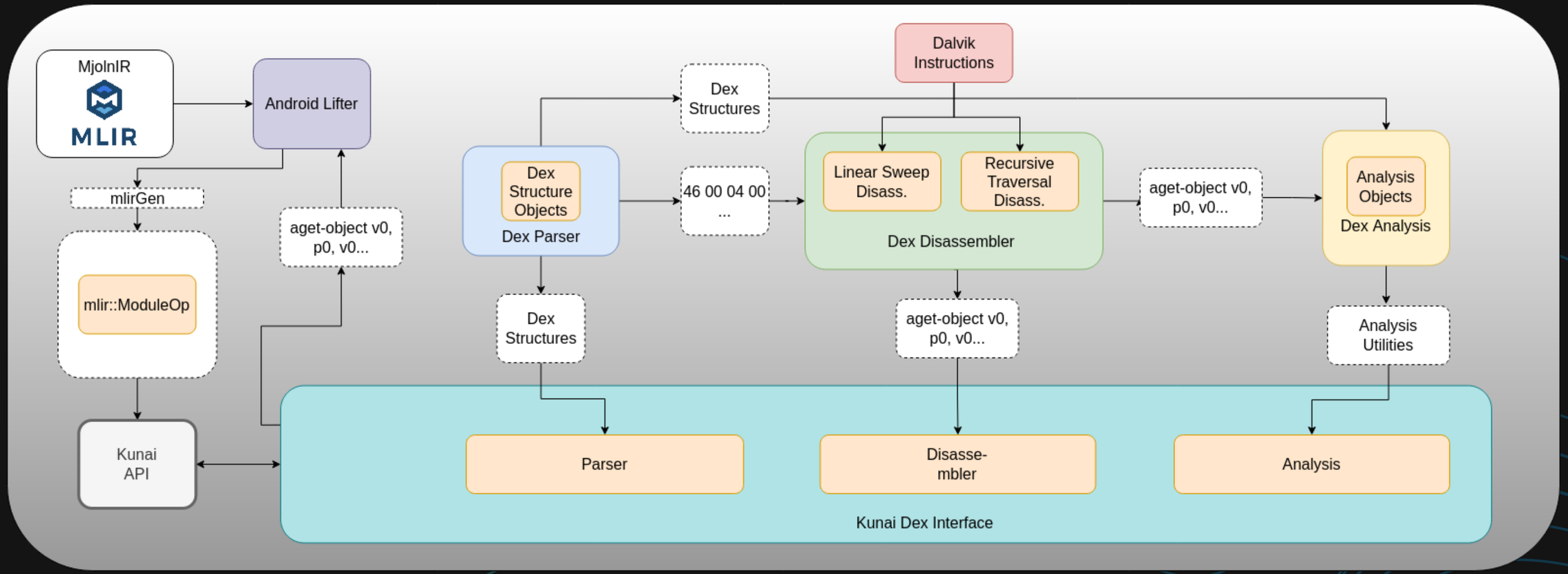
Kunai Lib

INTRODUCTION TO KUNAI

- **Library written in C++ for Dalvik Executable Analysis**
- **APK not supported yet but planned!!!**
- **It offers a parser for DEX format, two algorithms of disassembly, analysis utilities for creating CFG and x-refs**
- **A WIP for an Intermediate Representation (MjolnIR)**
- **The project started as a faster version of Androguard for using it during my PhD.**

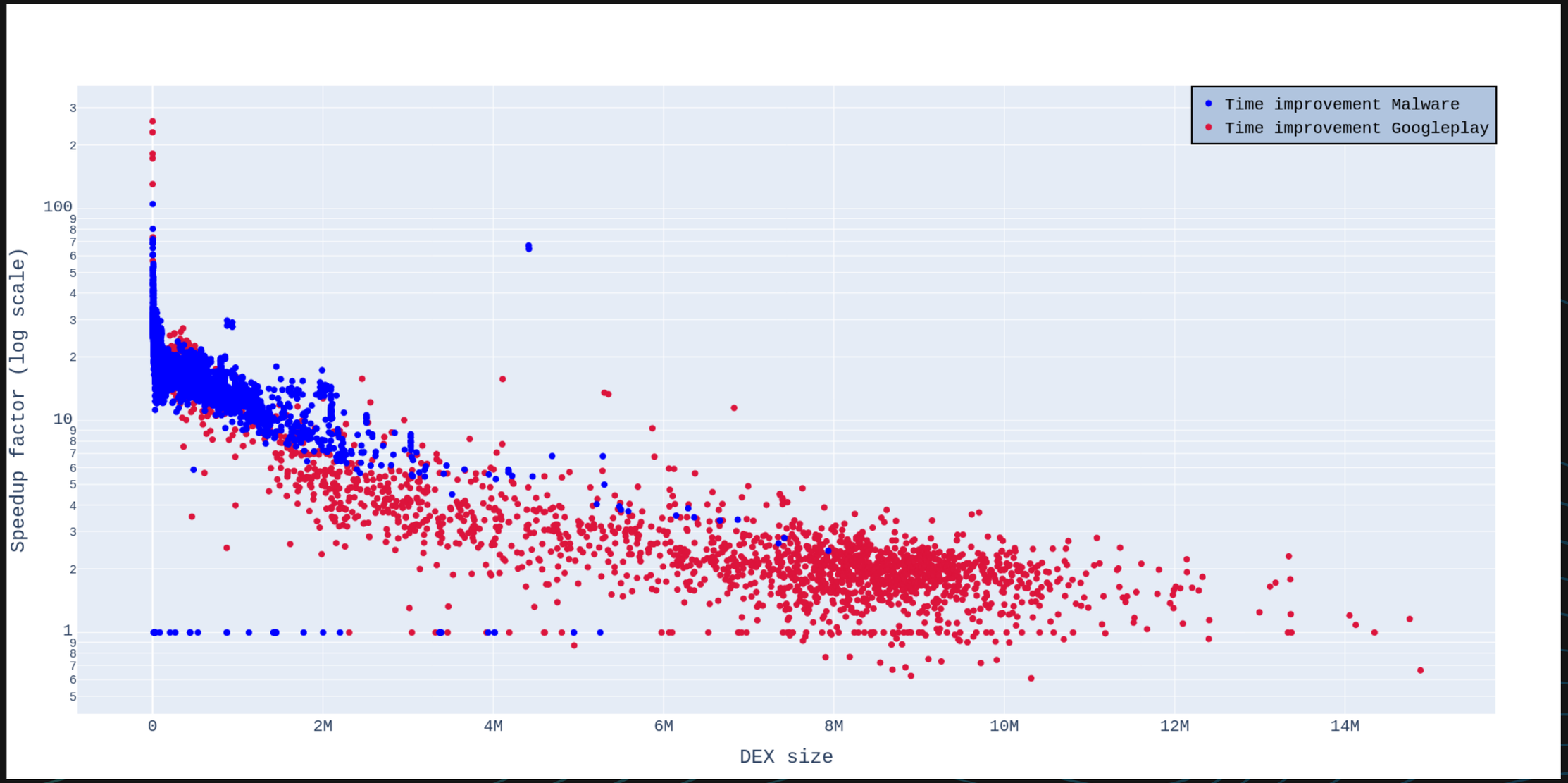
Kunai Lib

INTRODUCTION TO KUNAI



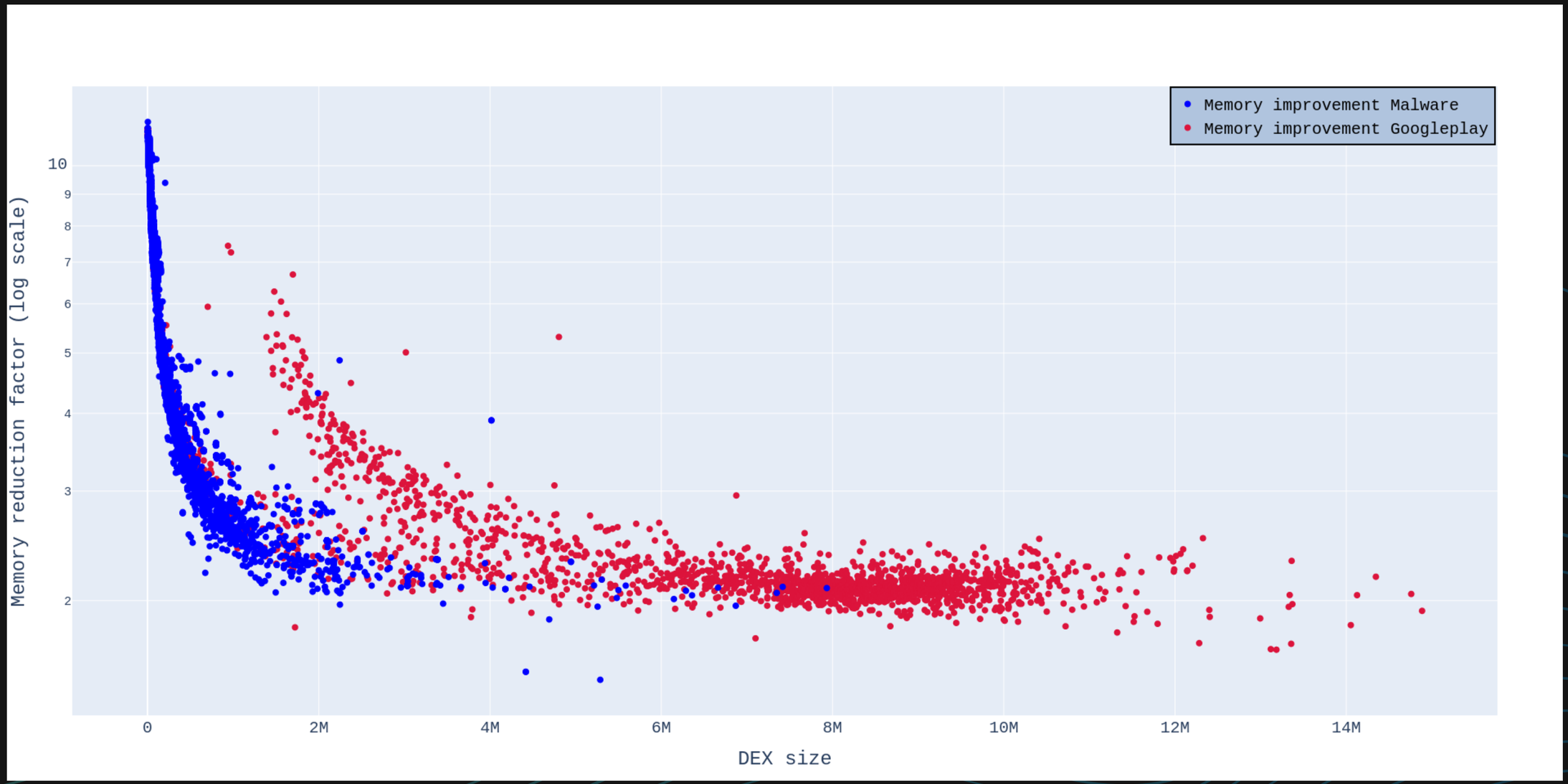
Kunai Lib

INTRODUCTION TO KUNAI



Kunai Lib

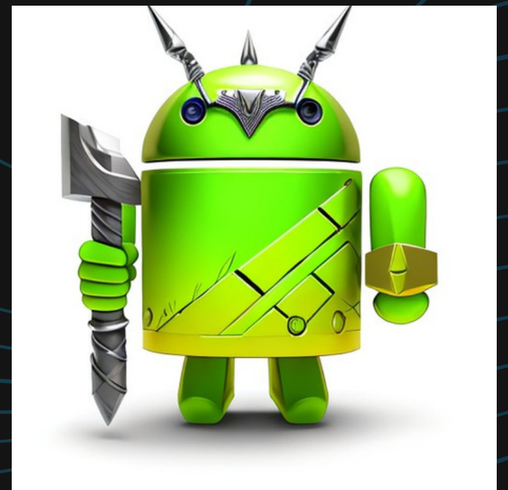
INTRODUCTION TO KUNAI



Kunai Lib

MJOLNIR

- Work in progress Intermediate Representation for Kunai
- Written using MLIR (part of LLVM project for writing new IRs)
- Dialects used:
  - ::mlir::cf::ControlFlowDialect
  - ::mlir::arith::ArithDialect
  - ::mlir::func::FuncDialect
  - ::mlir::KUNAI::MjolnIR::MjolnIRDialect
- MLIR imposes the use of the SSA format as well as the usage of block parameters instead of phi-nodes.



Kunai Lib

MJOLNIR

```
"builtin.module"() <{sym_name = "LMain;"}> ({
  "func.func"() <{function_type = (i32) -> i32, sym_name = "test_if"}> ({
  ^bb0(%arg0: !MjolnIR.object<"LMain;">, %arg1: i32):
    "MjolnIR.nop"() : () -> ()
    "MjolnIR.nop"() : () -> ()
    %0 = "arith.constant"() <{value = 0 : i32}> : () -> i32
    %1 = "arith.cmpi"(%arg1, %0) <{predicate = 1 : i64}> : (i32, i32) -> i1
    "cf.cond_br"(%1)[^bb2, ^bb1] <{odsOperandSegmentSizes = array<i32: 1, 0, 0>}> : (i1) -> ()
  ^bb1:  // pred: ^bb0
    %2 = "arith.constant"() <{value = 5 : i32}> : () -> i32
    %3 = "arith.constant"() <{value = 20 : i32}> : () -> i32
    "cf.br"(%2, %3)[^bb3] : (i32, i32) -> ()
  ^bb2:  // pred: ^bb0
    %4 = "MjolnIR.loadfield"() {access = 0 : i32, fieldClass = @"Ljava/lang/System;", fieldName = @out, fieldRef =
0 : ui32} : () -> !MjolnIR.object<"Ljava/io/PrintStream;">
    %5 = "MjolnIR.load-string"() {string = @"\22This is a test!\22", stringRef = 9 : ui32} : () -> !MjolnIR.object<
"Ljava/lang/String;">
    "MjolnIR.invoke"(%4, %5) {callee = @println, isStatic = 0 : ui32, methodRef = 2 : ui32} : (!MjolnIR.object<"Lja
va/io/PrintStream;">, !MjolnIR.object<"Ljava/lang/String;">) -> ()
    %6 = "arith.constant"() <{value = 30 : i32}> : () -> i32
    %7 = "arith.constant"() <{value = 2 : i32}> : () -> i32
    "cf.br"(%7, %6)[^bb3] : (i32, i32) -> ()
  ^bb3(%8: i32, %9: i32):  // 2 preds: ^bb1, ^bb2
    %10 = "arith.addi"(%8, %9) : (i32, i32) -> i32
    %11 = "arith.constant"() <{value = 33 : i32}> : () -> i32
    %12 = "arith.addi"(%10, %11) : (i32, i32) -> i32
    "func.return"(%12) : (i32) -> ()
  }) : () -> ()
}) : () -> ()
```

DEMO TIME!!!

Finished!
Questions?

EMAIL

kunai.static.analysis@gmail.com

WEBSITE

https://farena.in