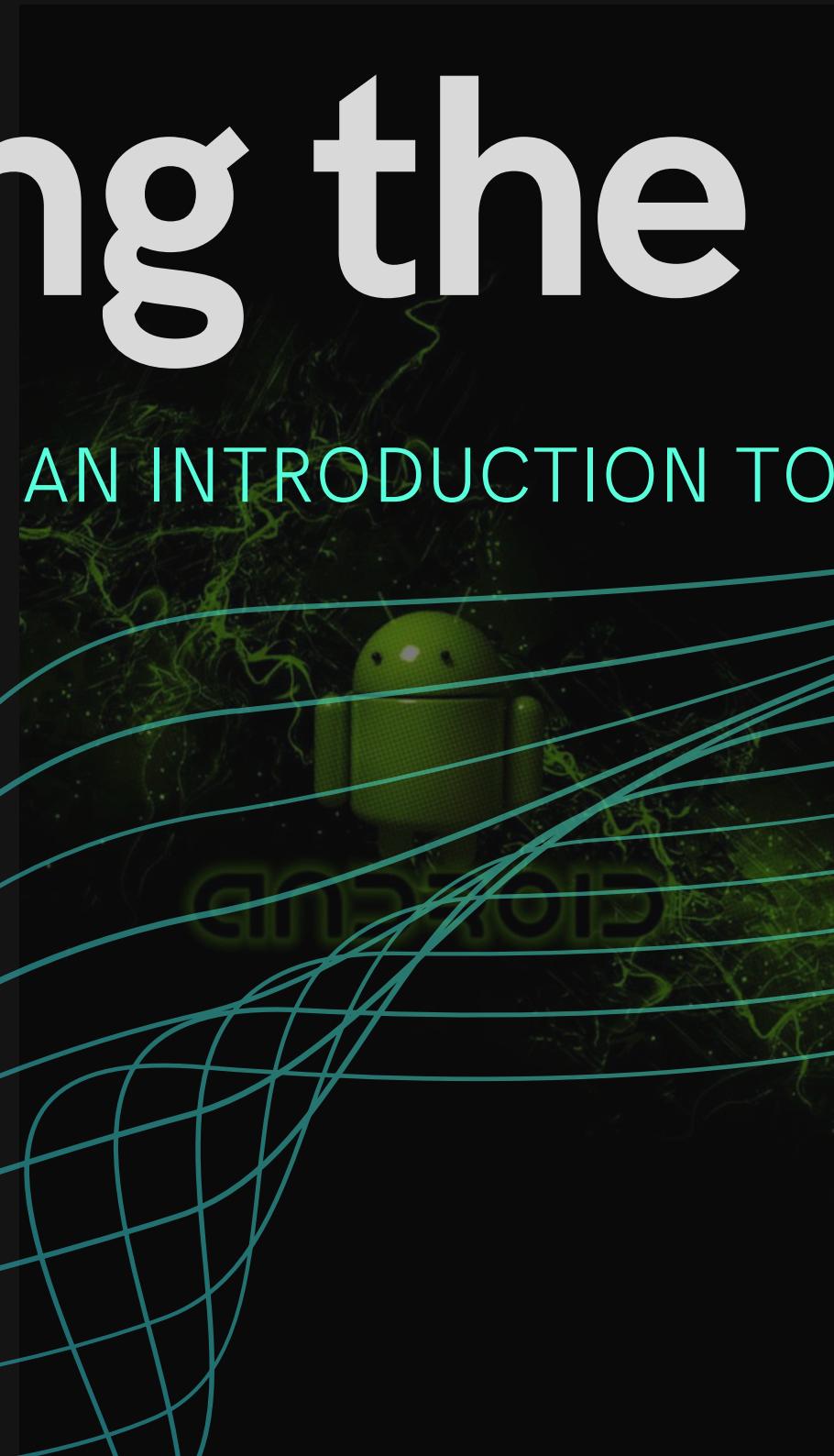


# Dissecting the DEX Format



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?

03

## 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

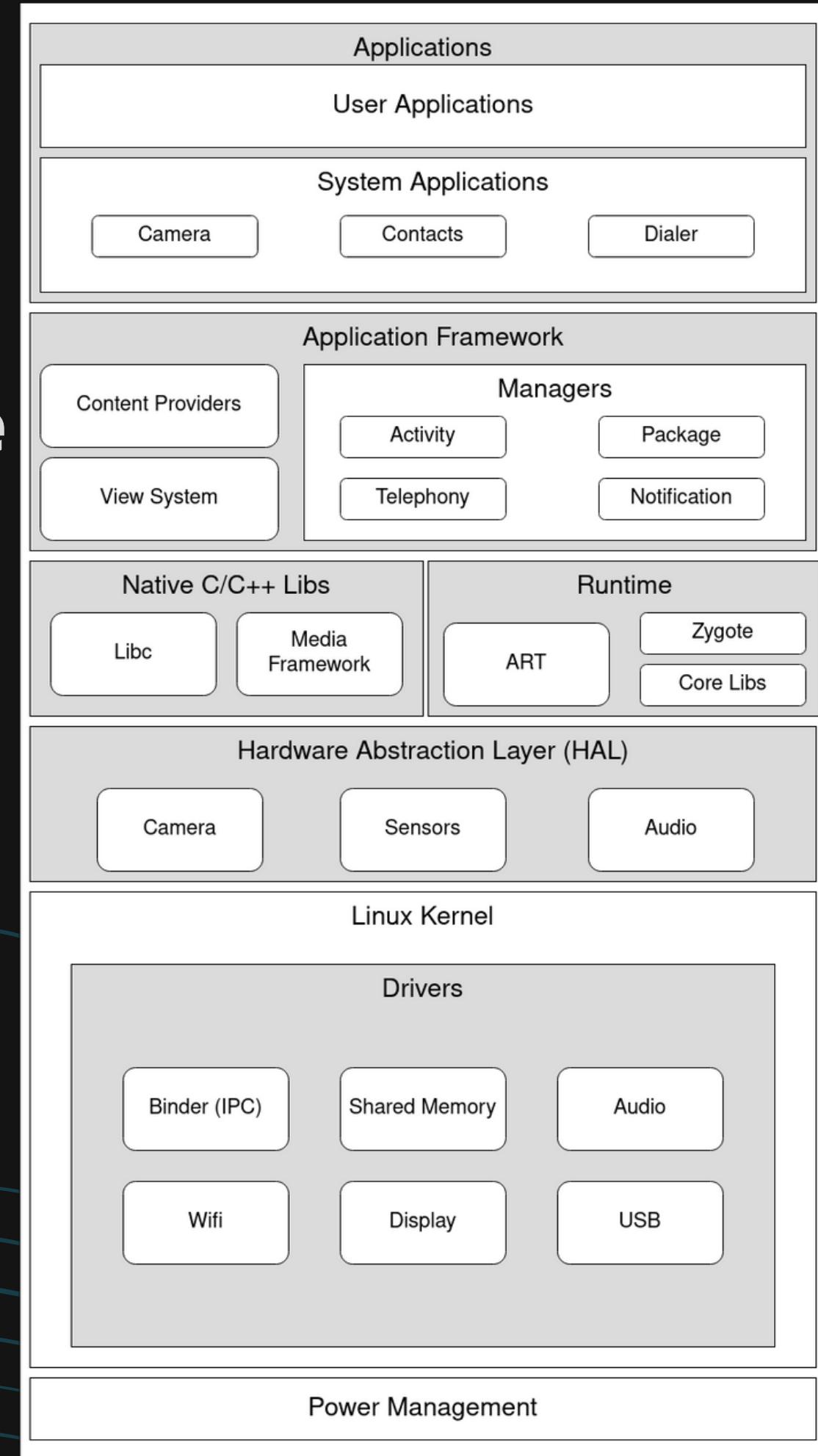
04



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..”*

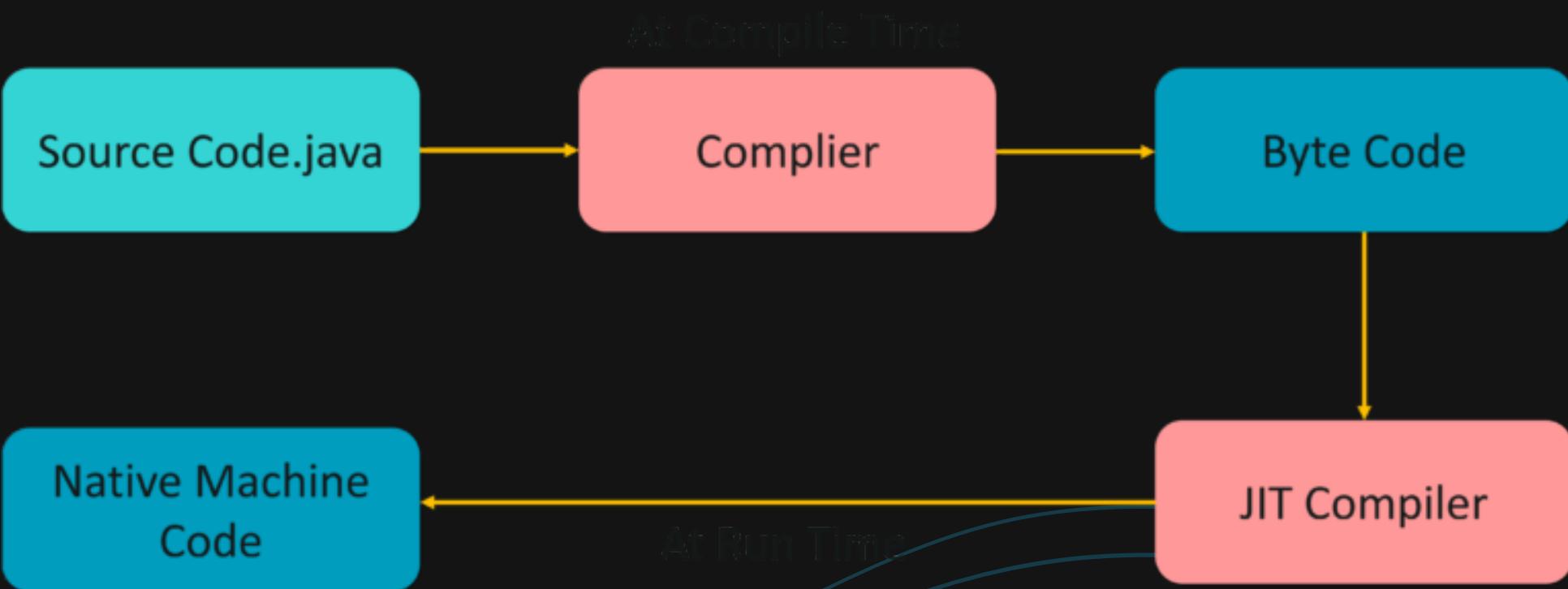
- First runtime from Android
- Virtual Machine register based (in opposite to JVM)
- More than 200 bytecodes supported
- Support for JIT of DEX files (since Android 2.2 “Froyo”)
- Replaced by Android ART since Android Lollipop

# Android app execution

DALVIK VIRTUAL MACHINE (DVM)

Memory	Registers	
	V0: 0	# Fetch
	V1: 0	opcode = memory[PC]
	V2: 0	PC += 1
	PC: 0	# Decode
		instruction = opcode & 0x0
		arg1 = opcode & 0x1
		arg2 = opcode & 0x2
		# Execute
SOFTWARE		if instruction == 0:
		register[arg1] = arg2

## Virtual Machine Execution

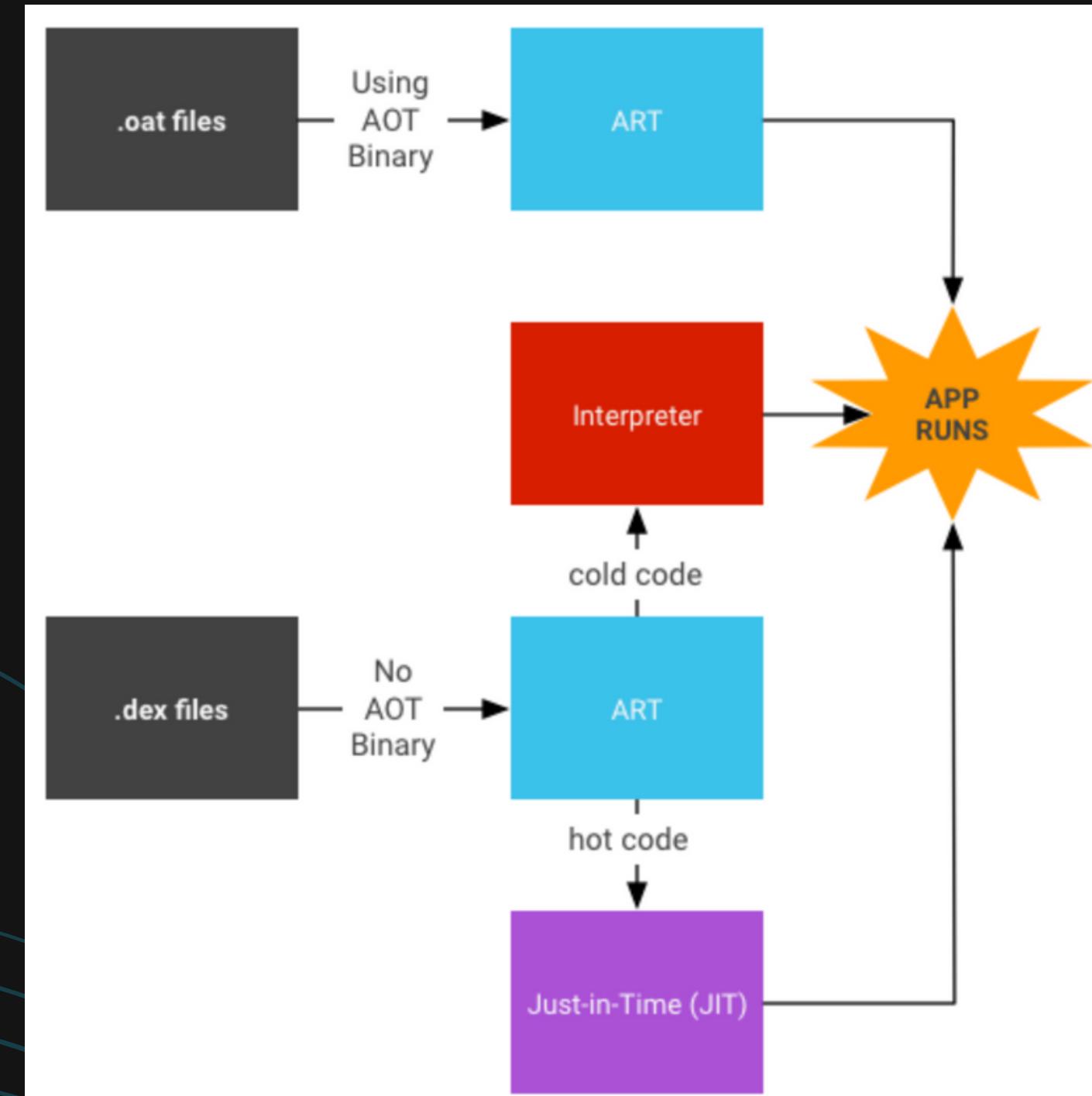


## Just In Time Compilation

# Android app execution

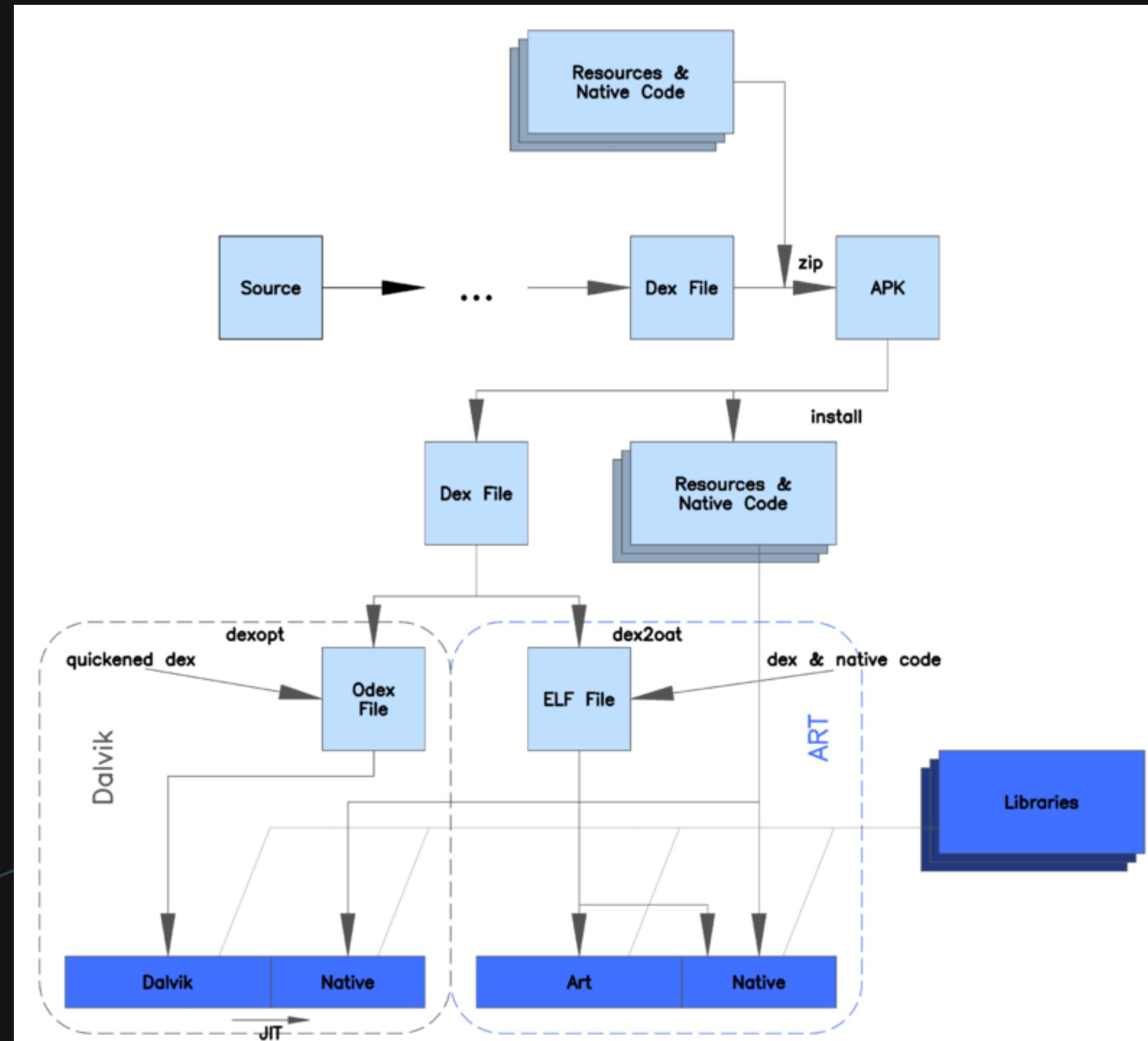
## ANDROID RUNTIME (ART)

- **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**



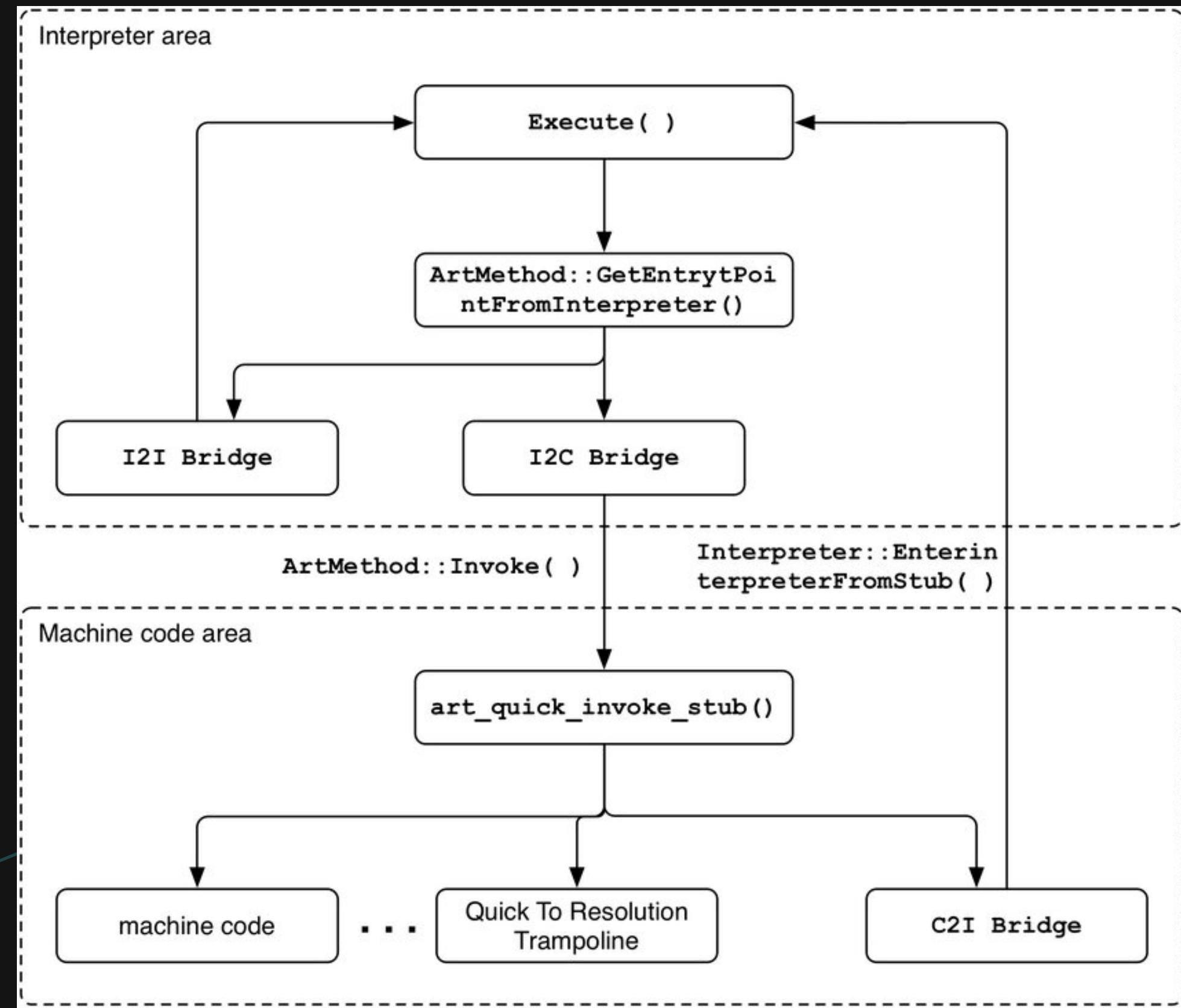
# Android app execution

## DALVIK VIRTUAL MACHINE VS ANDROID RUNTIME (ART)



# Android app execution

## ANDROID RUNTIME (ART) - NATIVE & INTERPRETER



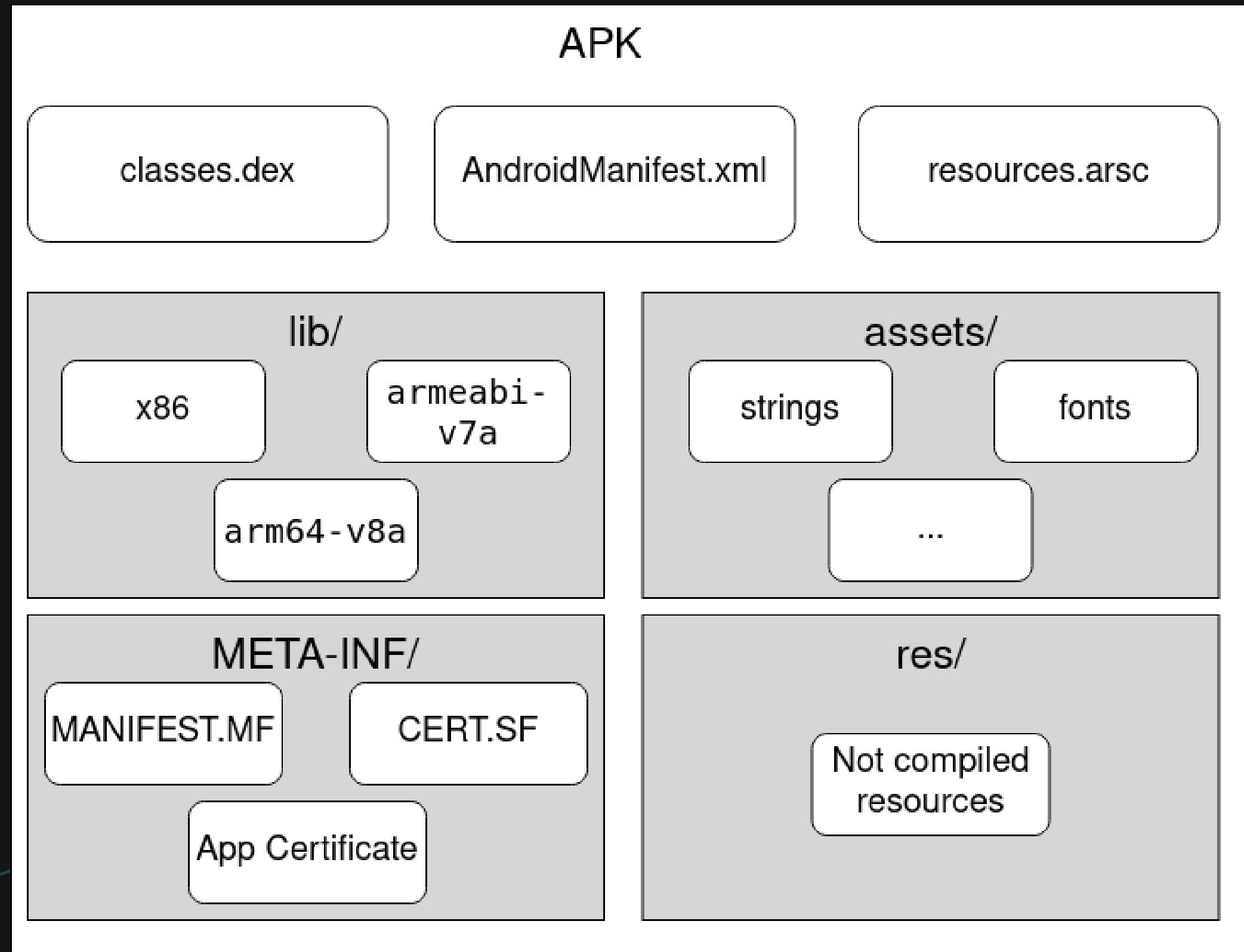
# APK & DEX

## 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)



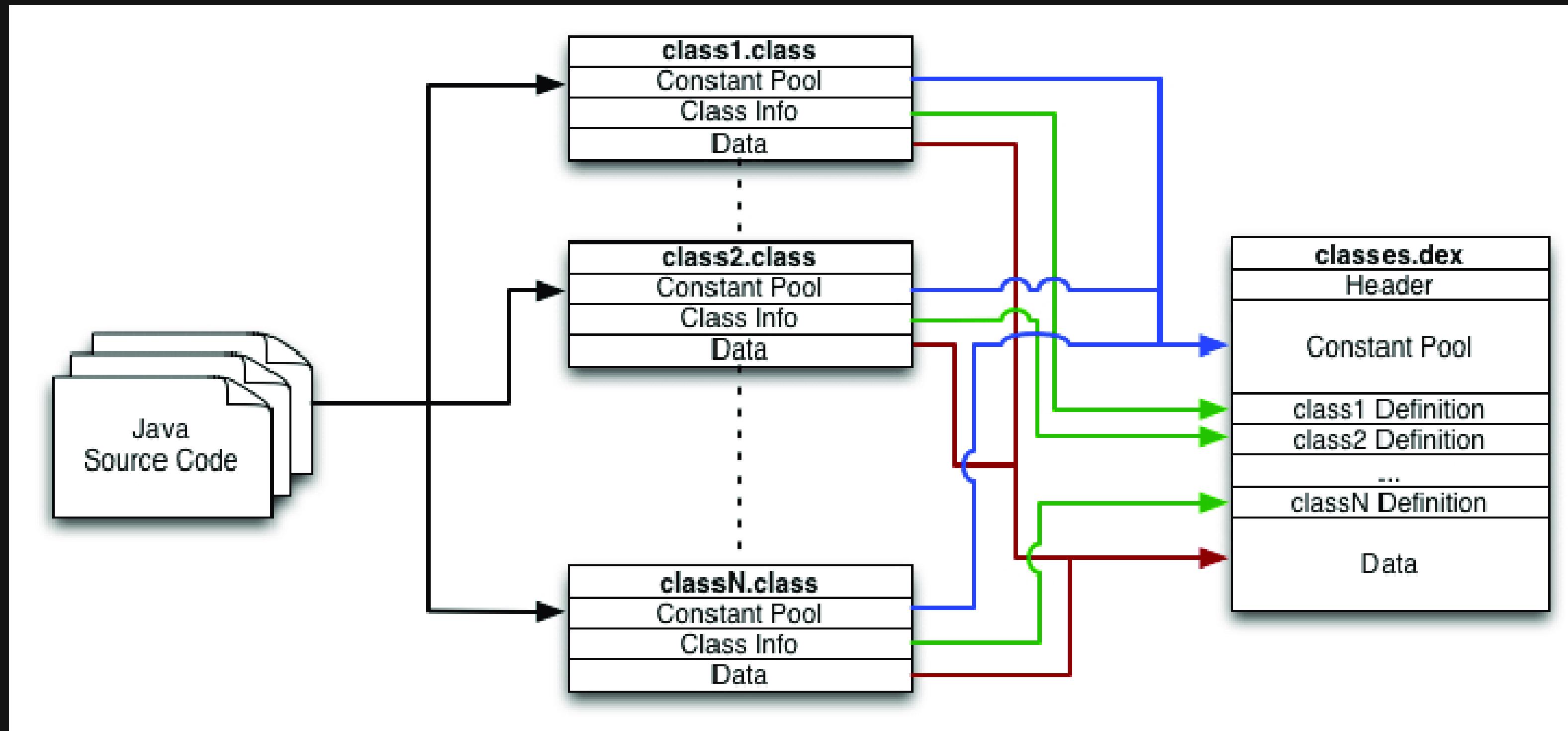
# APK & DEX

## 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)

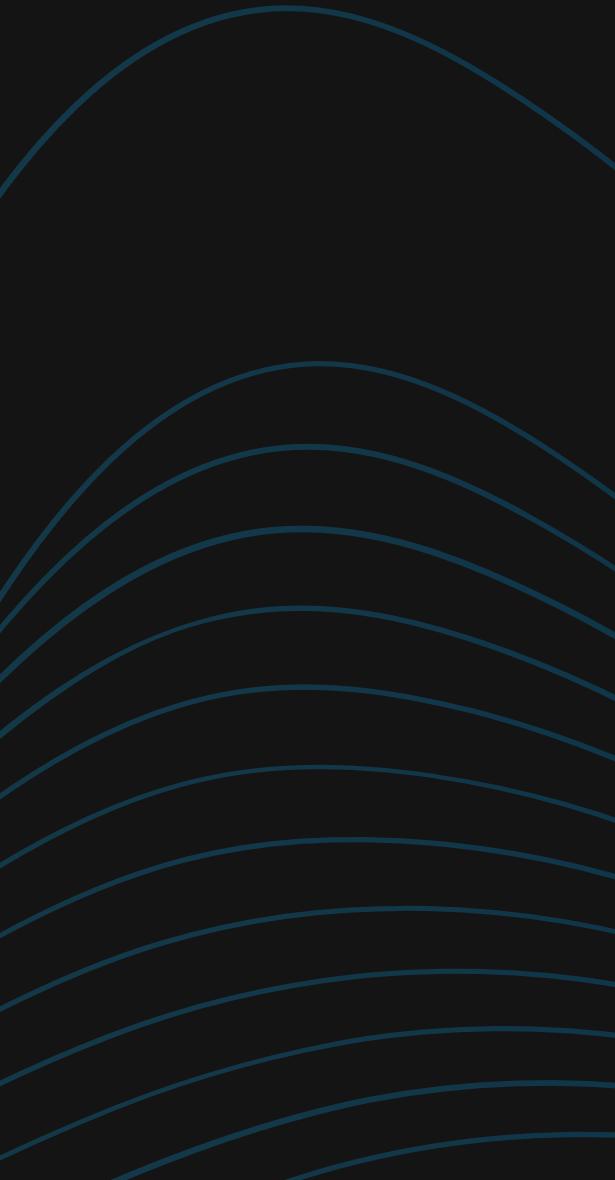


# APK & DEX

DALVIK EXECUTABLE (DEX)

**DEX,**  
WHAT PEOPLE THINK IT IS

15



# APK & DEX

## DALVIK EXECUTABLE (DEX)

16

# DALVIK EXECUTABLE

```
>adb shell dalvikvm -cp /data/hw.zip hw  
Hello World!
```

```
0 1 2 3 4 5 6 7 8 9 A B C D E F  
000: .d .e .x 0A .0 .3 .5 00 6F 53 89 BC 1E 79 B2 4F  
010: 1F 9C 09 66 15 23 2D 3B 56 65 32 C3 B5 81 B4 5A  
020: 70 02 00 00 70 00 00 00 78 56 34 12 00 00 00 00  
030: 00 00 00 00 DC 01 00 00 0C 00 00 00 70 00 00 00  
040: 07 00 00 00 A0 00 00 00 02 00 00 00 BC 00 00 00  
050: 01 00 00 00 D4 00 00 00 02 00 00 00 DC 00 00 00  
060: 01 00 00 00 EC 00 00 00 64 01 00 00 0C 01 00 00  
070: A6 01 00 00 3A 01 00 00 8A 01 00 00 40 01 00 00  
080: B4 01 00 00 76 01 00 00 54 01 00 00 6C 01 00 00  
090: 57 01 00 00 70 01 00 00 A1 01 00 00 C8 01 00 00  
0A0: 01 00 00 00 02 00 00 00 03 00 00 00 04 00 00 00  
0B0: 05 00 00 00 06 00 00 00 08 00 00 00 07 00 00 00  
0C0: 05 00 00 00 34 01 00 00 07 00 00 00 05 00 00 00  
0D0: 2C 01 00 00 04 00 01 00 0A 00 00 00 00 00 01 00  
0E0: 09 00 00 00 01 00 00 00 0B 00 00 00 00 00 00 00  
0F0: 01 00 00 00 02 00 00 00 00 00 00 00 FF FF FF FF  
100: 00 00 00 00 D1 01 00 00 00 00 00 00 02 00 01 00  
110: 02 00 00 00 00 00 00 00 08 00 00 00 62 00 00 00  
120: 1A 01 00 00 6E 20 01 00 10 00 0E 00 01 00 00 00  
130: 06 00 00 00 01 00 00 00 03 00 00 04 .L .h .w ; 00  
140: 12 .L .j .a .v .a ./ .1 .a .n .g ./ .0 .b .j .e  
150: .c .t .; 00 01 .V 00 13 .[ .L .j .a .v .a ./ .1  
160: .a .n .g ./ .S .t .r .i .n .g .; 00 02 .V .L 00  
170: 04 .m .a .i .n 00 12 .L .j .a .v .a ./ .1 .a .n  
180: .g ./ .S .y .s .t .e .m .; 00 15 .L .j .a .v .a  
190: ./ .i .o ./ .P .r .i .n .t .S .t .r .e .a .m .;  
1A0: 00 03 .o .u .t 00 0C .H .e .l .1 .0 20 .W .o .r  
1B0: .1 .d .! 00 12 .L .j .a .v .a ./ .1 .a .n .g ./  
1C0: .S .t .r .i .n .g .; 00 07 .p .r .i .n .t .l .n  
1D0: 00 00 00 01 00 00 09 8C 02 00 00 00 0C 00 00 00  
1E0: 00 00 00 01 00 00 00 00 00 00 00 01 00 00 00  
1F0: 0C 00 00 00 70 00 00 00 02 00 00 00 07 00 00 00  
200: A0 00 00 00 03 00 00 00 02 00 00 00 BC 00 00 00  
210: 04 00 00 00 01 00 00 00 D4 00 00 00 05 00 00 00  
220: 02 00 00 00 DC 00 00 00 06 00 00 00 01 00 00 00  
230: EC 00 00 00 01 20 00 00 01 00 00 00 0C 01 00 00  
240: 01 10 00 00 02 00 00 00 2C 01 00 00 02 20 00 00  
250: 0C 00 00 00 3A 01 00 00 00 20 00 00 01 00 00 00  
260: D1 01 00 00 00 10 00 00 01 00 00 00 DC 01 00 00
```

ANGE ALBERTINI  
<http://pics.corkami.com>

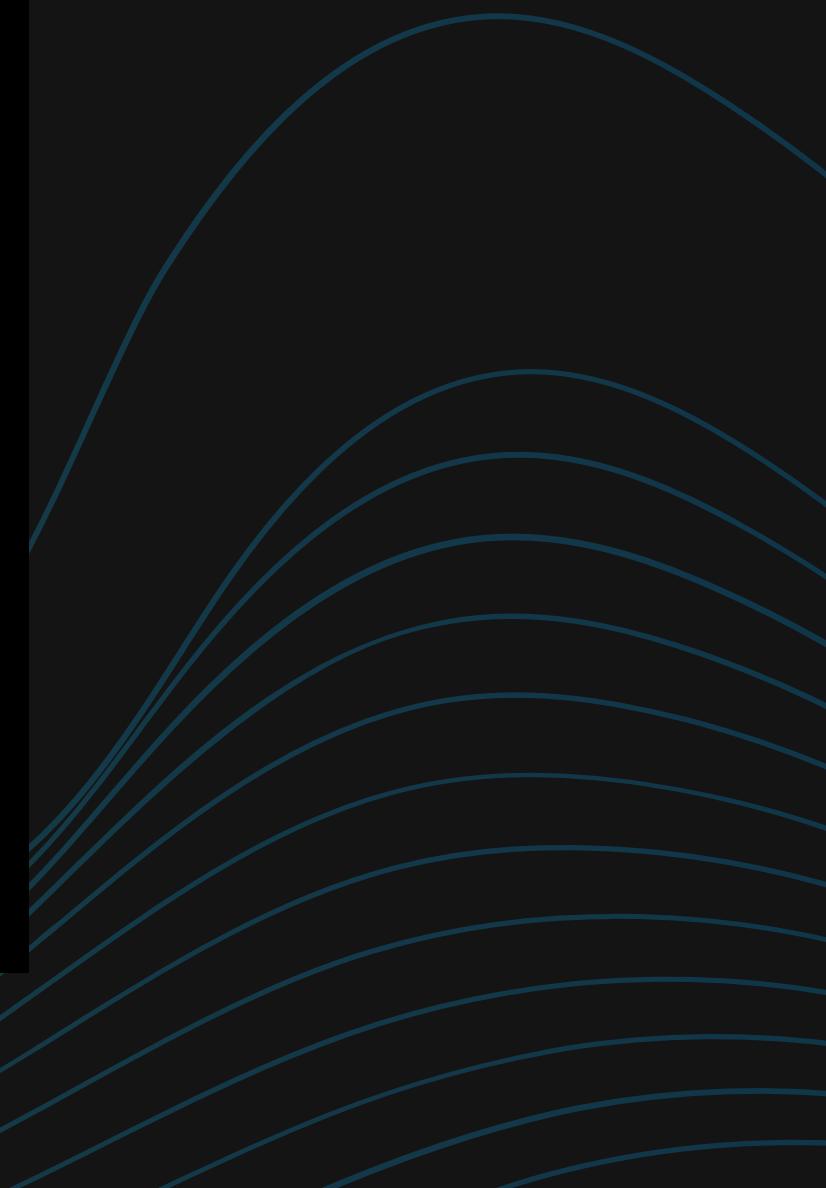


magic	"dex\n035\0"
adler32	0x8C89536F 1e79034f9c99961523 23999242c20581045a
sha1	
file_size	0x270
header_size	0x70
endian_tag	0x12345678 (little endian)
map offset	0x1DC
size / offsets	0x00C/0x070
strings_ids	0x007/0x0A0
type_ids	0x002/0x00C
proto_ids	0x001/0x004
field_ids	0x002/0x00C
method_ids	0x002/0x00C
class_defs	0x001/0x0EC
data	0x164/0x10C
HEADER	
STRING IDS (A-Z ORDER)	
TYPE IDs (STRING LIST INDEXES)	1 2 3 4 5 6 8
PROTO IDS	string_id return_type offset descriptor 7 5 0x134 7 5 0x12C
FIELD IDS	class 0x4 (Ljava/lang/System;) type 0x1 ([Ljava/io/PrintStream;) name 0xA ('out')
METHOD IDS	class 0x0 ("Lhw;") prototype 0x1 ([Ljava/lang/String;) name 0x9 ("main") class 0x1 (Ljava/io/PrintStream;) prototype 0x0 ([Ljava/lang/String;) name 0xB ("println")
CLASS DEFS	class 0x0 ("hw") access_flag 0x1 (PUBLIC) superclass 0x2 ([Ljava/lang/Object;) source 0xFFFFFFFF (none) data_offset 0x101
CODE	registers 2 in args 2 (words) out args 2 (words) instructions 8 (words) sgt-object v0, :out:[Ljava/io/PrintStream; const-string v1, "Hello World!" invoke-virtual {v0, v1}, :println([Ljava/lang/String;)v return-void
TYPE LIST	size 1 type 6 ([Ljava/lang/String;) size 1 type 3 ([Ljava/lang/String;)
STRING DATA (MUTF-8)	len / string 04 "Lhw;" 18 "[Ljava/lang/Object;" 1 "V" 19 "[Ljava/lang/String;" 2 "VL" 4 "main" 18 "[Ljava/lang/System;" 21 "[Ljava/io/PrintStream;" 3 "out" 12 "Hello World!" 18 "[Ljava/lang/String;" 7 "println"
CLASS DATA	direct methods 1 index diff 0x0 flags 0x9 (PUBLIC STATIC) code offset 0x020C (0x10C, encoded in uleb128)
MAP	count 12 0x0000 (HEADER) 1 0x000 0x0001 (STRING) 12 0x070 0x0002 (TYPE) 7 0x0A0 0x0003 (PROTO) 2 0x0C 0x0004 (FIELD) 1 0x004 0x0005 (METHOD) 2 0x00C 0x0006 (CLASS) 1 0x0EC 0x2001 (CODE) 1 0x10C 0x1001 (TYPE_LIST) 2 0x12C 0x2002 (STRING_DATA) 12 0x13A 0x2000 (CLASS_DATA) 1 0x101 0x1000 (MAP_LIST) 1 0x10C

# APK & DEX

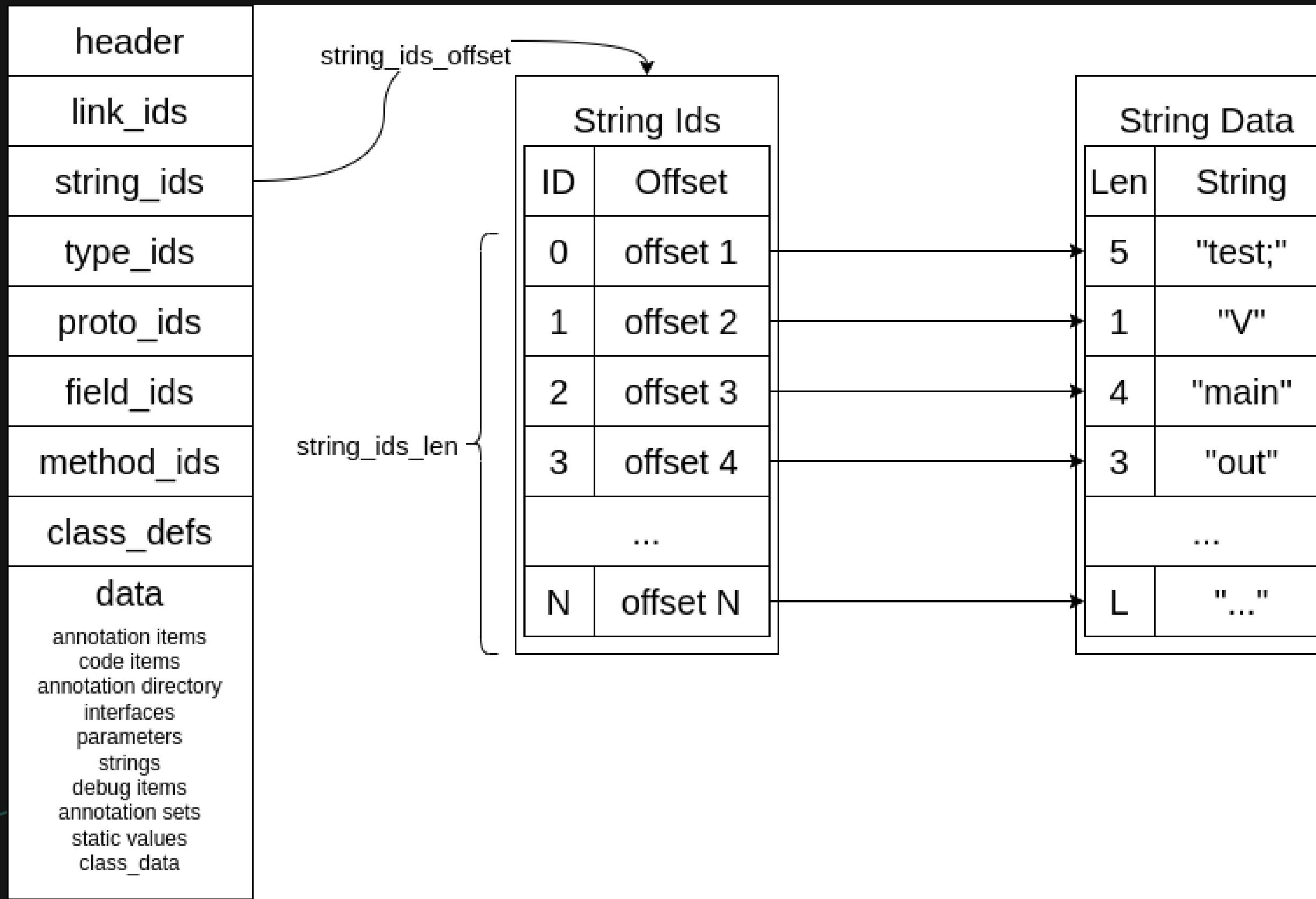
## DALVIK EXECUTABLE (DEX)

**DEX,  
WHAT IT REALLY IS**



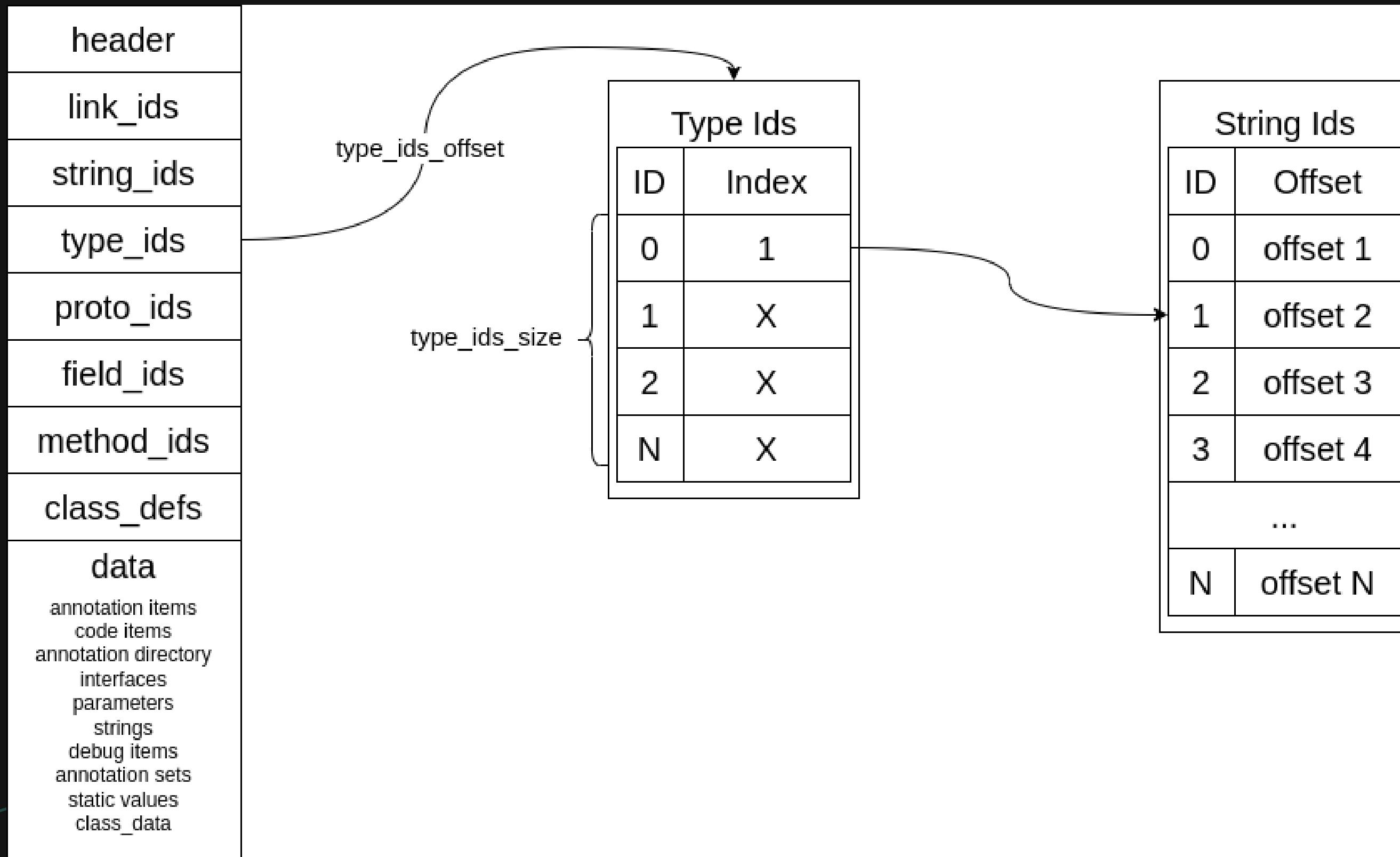
# APK & DEX

## DALVIK EXECUTABLE (DEX)



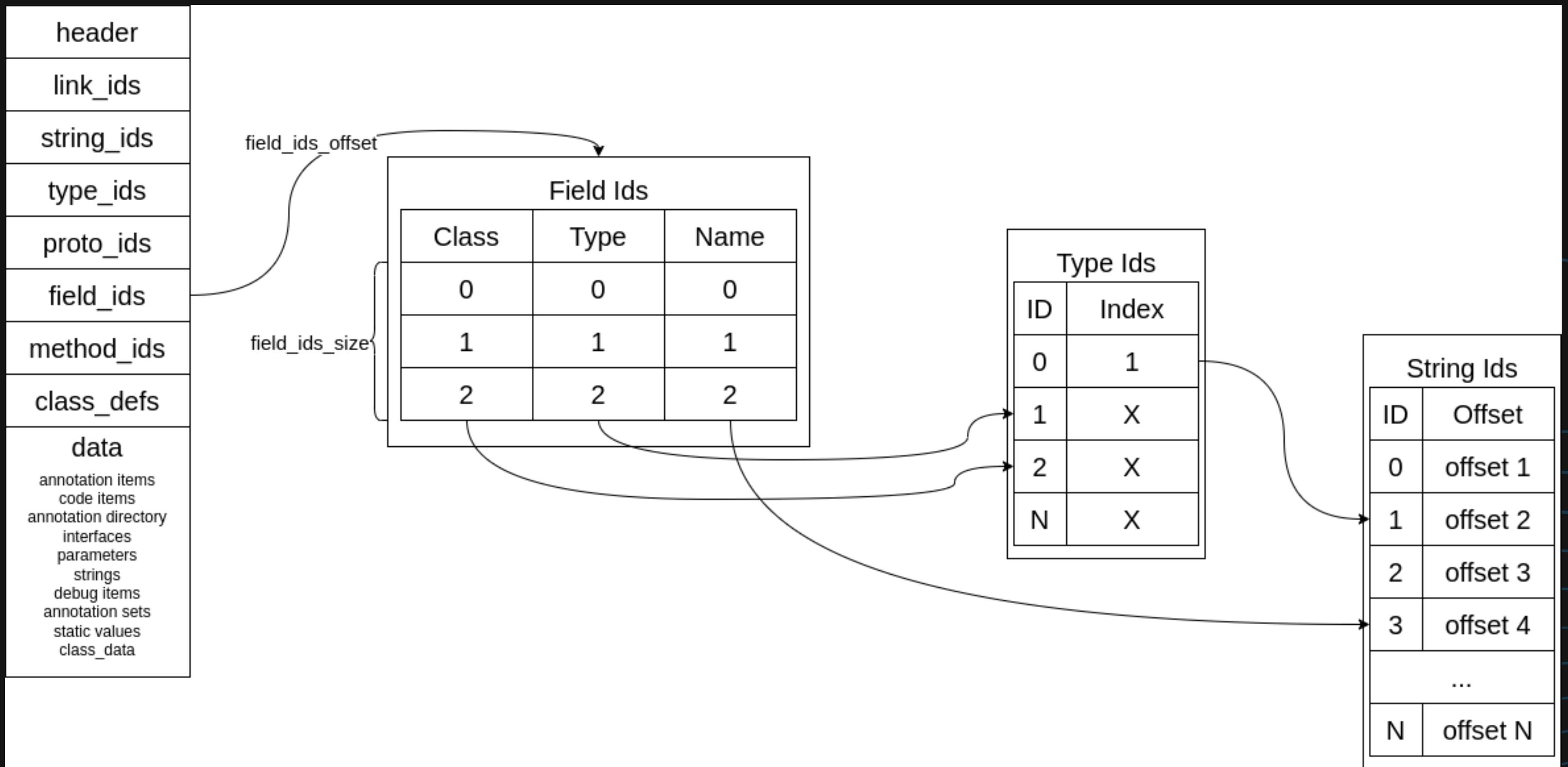
# APK & DEX

## DALVIK EXECUTABLE (DEX)



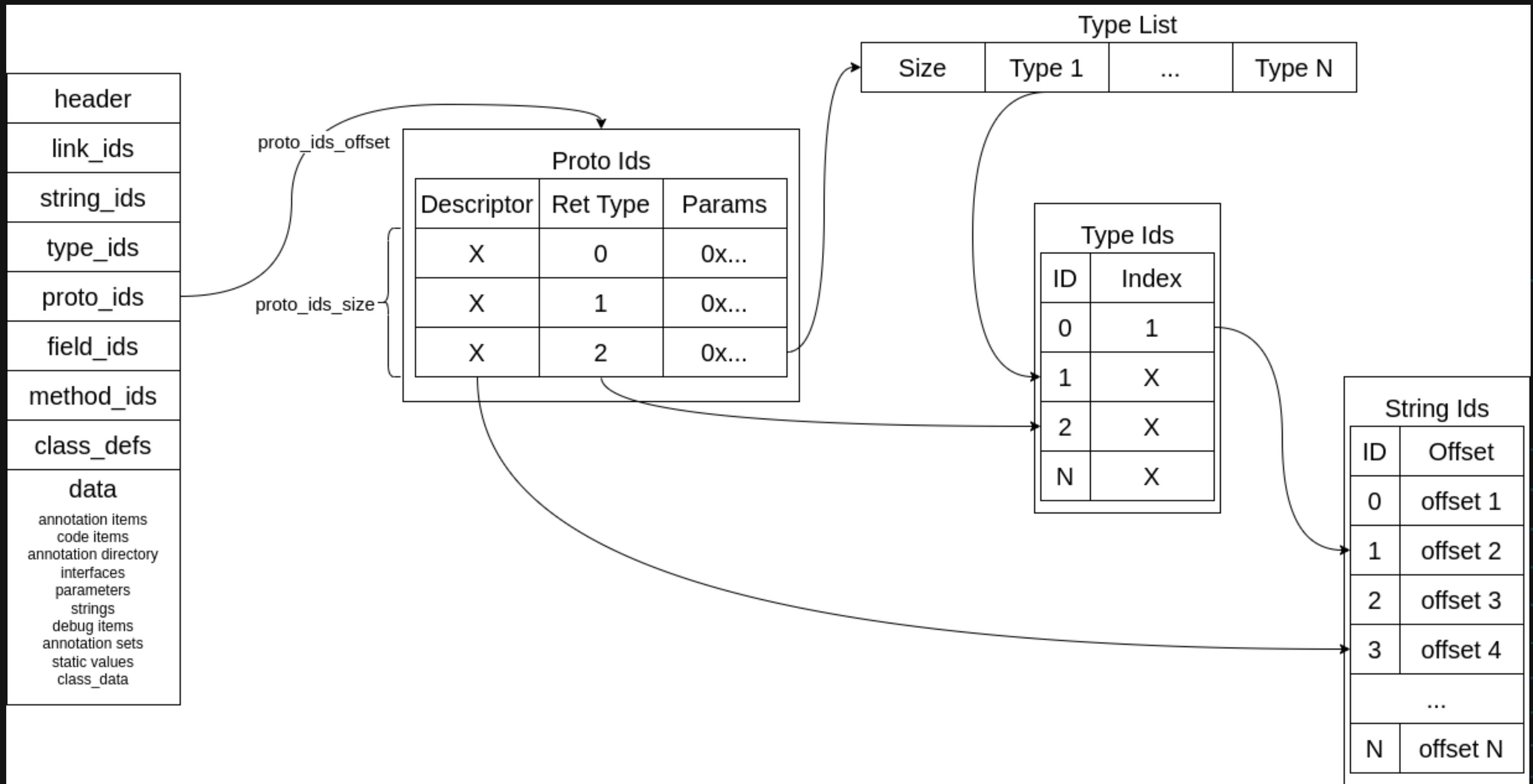
# APK & DEX

# DALVIK EXECUTABLE (DEX)



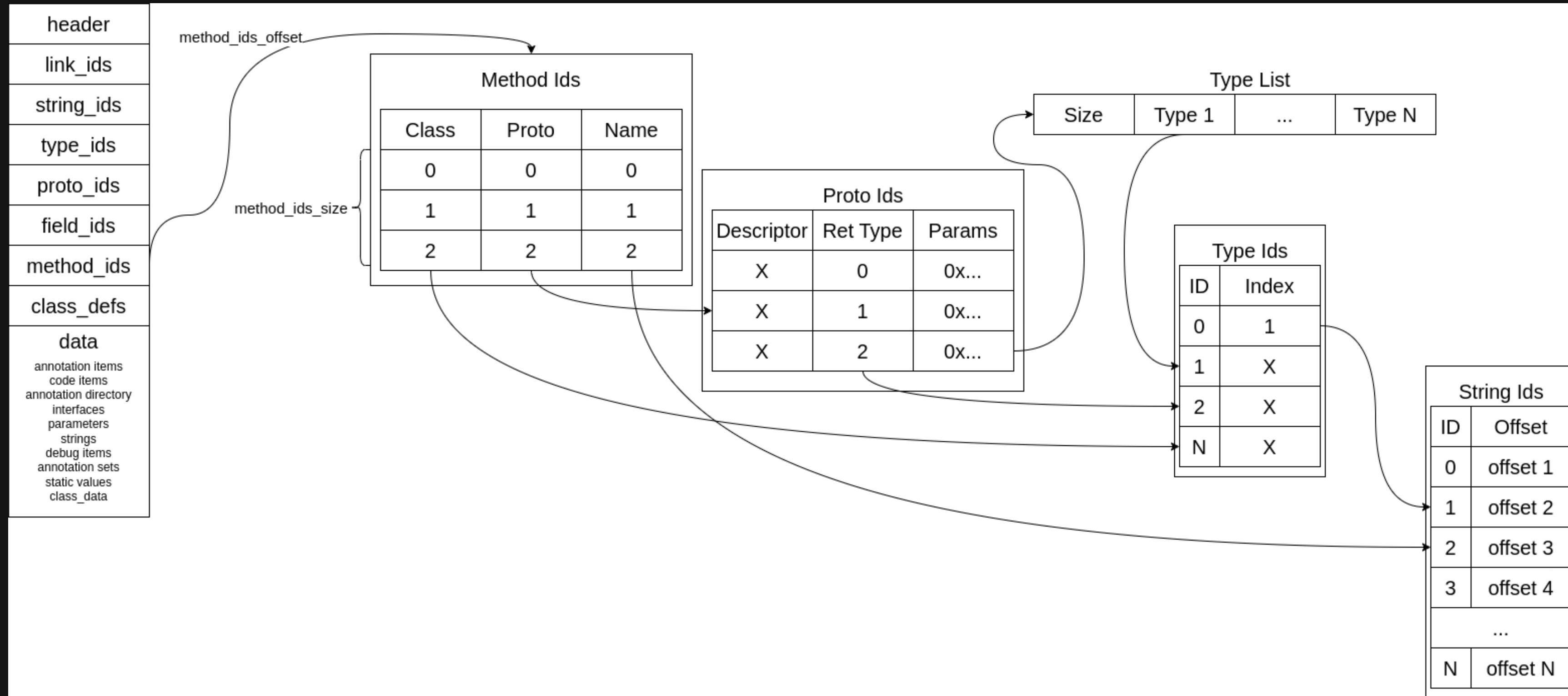
# APK & DEX

## DALVIK EXECUTABLE (DEX)



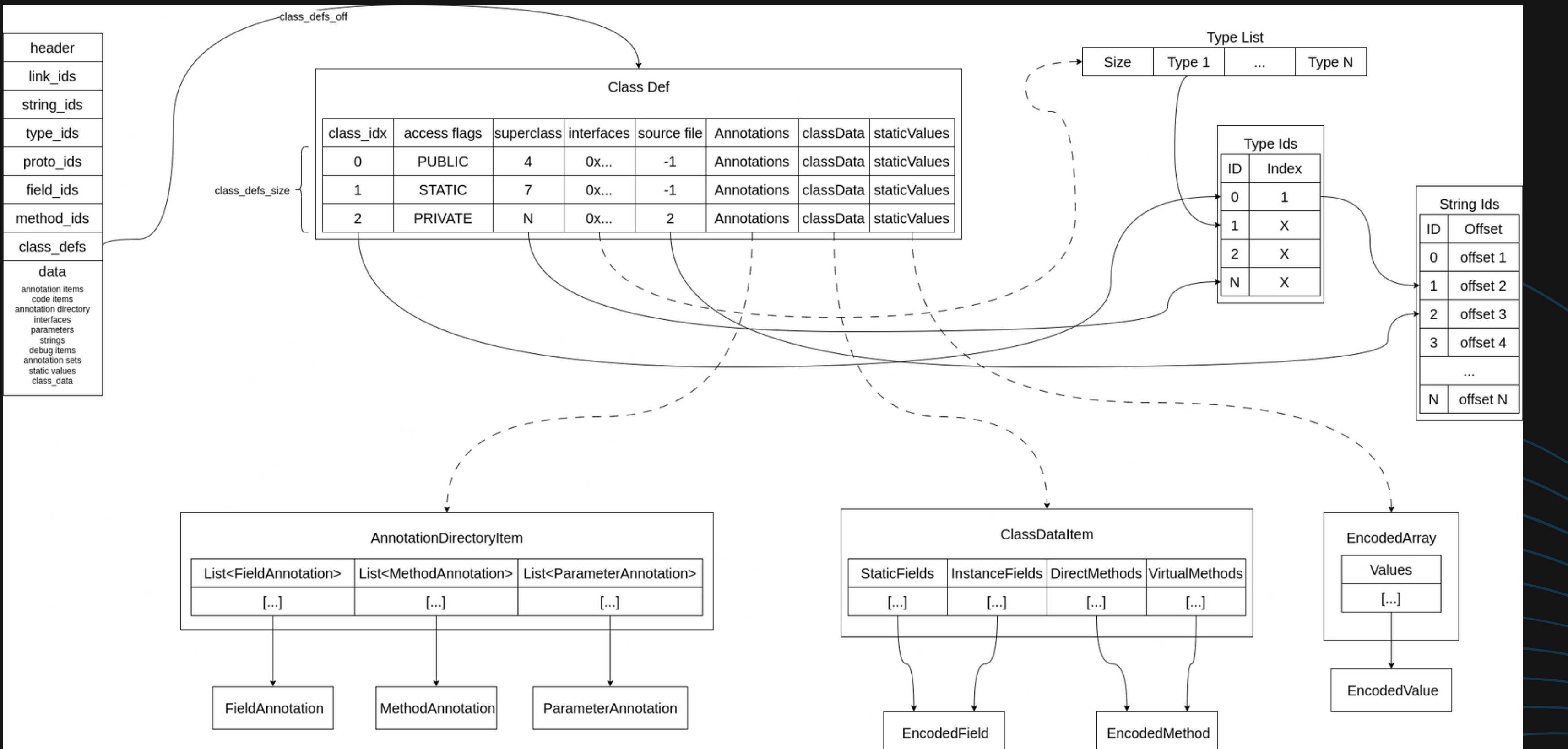
# APK & DEX

## DALVIK EXECUTABLE (DEX)



# APK & DEX

## DALVIK EXECUTABLE (DEX)



# APK & DEX

## DALVIK EXECUTABLE (DEX) - BYTECODE

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	<b>nop</b>	*	<b>move</b>	/from16 /16	*	<b>move-wide</b>	/from16 /16	*	<b>move-object</b>	/from16 /16	*	<b>move-result</b>	-wide -object	<b>move-exception</b>	-void	<b>return</b> *
1x	wide object	return-* /4 /16	<b>const</b>	<b>const</b>	<b>const</b>	<b>const-wide</b>	/high16 /16 /32 lit64 /high16	*	<b>const-string</b>	* -jumbo	<b>const-class</b>	<b>monitor</b>	enter exit	<b>check cast</b>		
2x	<b>instance of</b>	<b>array new</b>	<b>filled-new-array</b>	<b>fill</b>	<b>throw</b>	<b>goto</b>	*	/16 /32	<b>-switch</b>	packed sparse	<b>cmp*-float</b>	<b>cmp-double</b>				
3x	<b>cmp-double</b>	<b>cmp-long</b>	eq ne	lt ge	gt le	eqz	nez	ltz	gez	gtz	lez					
4x						<b>aget</b>	*	-wide -object -bool -byte -byte	-char -short	*	-wide -object -bool -byte -char	-short			<b>aput...</b>	
5x	<b>...aput</b>	-char -short	*	-wide -object	-bool	<b>iget</b>	-byte -char -short	*	-wide -object	-bool	<b>iput</b>	-byte -char	-short			
6x	*	-wide -object	-bool	-byte	-char	<b>sget</b>	-short	*	-wide -object	-bool	<b>sput</b>	-byte -char	-short	<b>virtual</b>	<b>super</b>	<b>invoke-...</b>
7x	<b>...invoke-</b>	-direct -static -interface		virtual	super	<b>invoke-* range</b>	-direct -static -interface				<b>neg</b>	<b>not</b>	<b>neg long</b>	<b>not long</b>	<b>neg float</b>	
8x	<b>neg</b>	double	<b>int-to-float</b>	long	float	<b>int</b>	<b>long-to-float</b>	double	<b>float-to</b>	int	<b>double-to</b>	long	float	<b>int-to-char</b>	<b>short</b>	
9x	<b>add-</b>	<b>sub-</b>	<b>mul-</b>	<b>div-</b>	<b>rem-</b>	<b>and-</b>	<b>or-</b>	<b>xor-</b>	<b>shl-</b>	<b>shr-</b>	<b>ushr-</b>	<b>add-</b>	<b>sub-</b>	<b>mul-</b>	<b>div-</b>	<b>rem-</b>
Ax	<b>...-long</b>	<b>and-</b>	<b>or-</b>	<b>xor-</b>	<b>shl-</b>	<b>shr-</b>	<b>ushr-</b>		<b>add</b>	<b>sub</b>	<b>mul</b>	<b>div</b>		<b>-double</b>		
Bx	<b>add-</b>	<b>sub-</b>	<b>mul-</b>	<b>div-</b>	<b>rem-</b>	<b>and-</b>	<b>or-</b>	<b>xor-</b>	<b>shl-</b>	<b>shr-</b>	<b>ushr-</b>	<b>add</b>	<b>sub</b>	<b>mul</b>	<b>div</b>	<b>rem</b>
Cx	<b>...-long/2addr</b>	<b>and-</b>	<b>or-</b>	<b>xor-</b>	<b>shl-</b>	<b>shr-</b>	<b>ushr-</b>		<b>add</b>	<b>sub</b>	<b>mul</b>	<b>div</b>		<b>-double/2addr</b>		
Dx	<b>add-</b>	<b>sub-</b>	<b>mul-</b>	<b>div-</b>	<b>rem-</b>	<b>and-</b>	<b>or-</b>	<b>xor-</b>	<b>add-</b>	<b>sub-</b>	<b>mul-</b>	<b>div-</b>	<b>rem-</b>	<b>and-</b>	<b>or-</b>	<b>xor-</b>
Ex				<b>-int/lit8</b>										<b>execute inline</b>		
Fx	<b>invokes-direct-empty</b>					<b>iget-*-quick</b>	*	wide object	<b>input-*-quick</b>	*	wide object	virtual	virtual/range	super super/range		
		<b>misc</b>		<b>object</b>		<b>conversion</b>										
		<b>moves</b>		<b>flow</b>		<b>arithmetic</b>										
		<b>method</b>		<b>conditional</b>												
		<b>literals</b>		<b>transfer</b>												
		<b>system</b>		<b>logical</b>		<b>undefined</b>										

Dalvik Virtual Machine (android)

# Kunai Lib

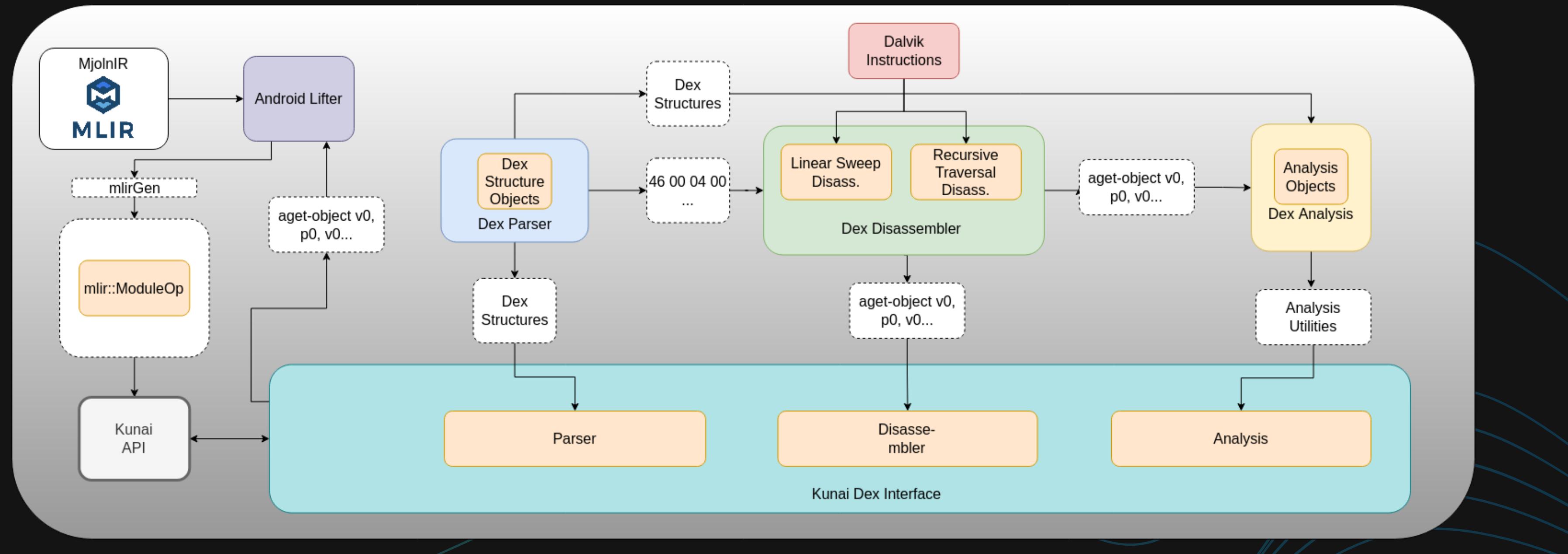
## INTRODUCTION TO KUNAI

- **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.**



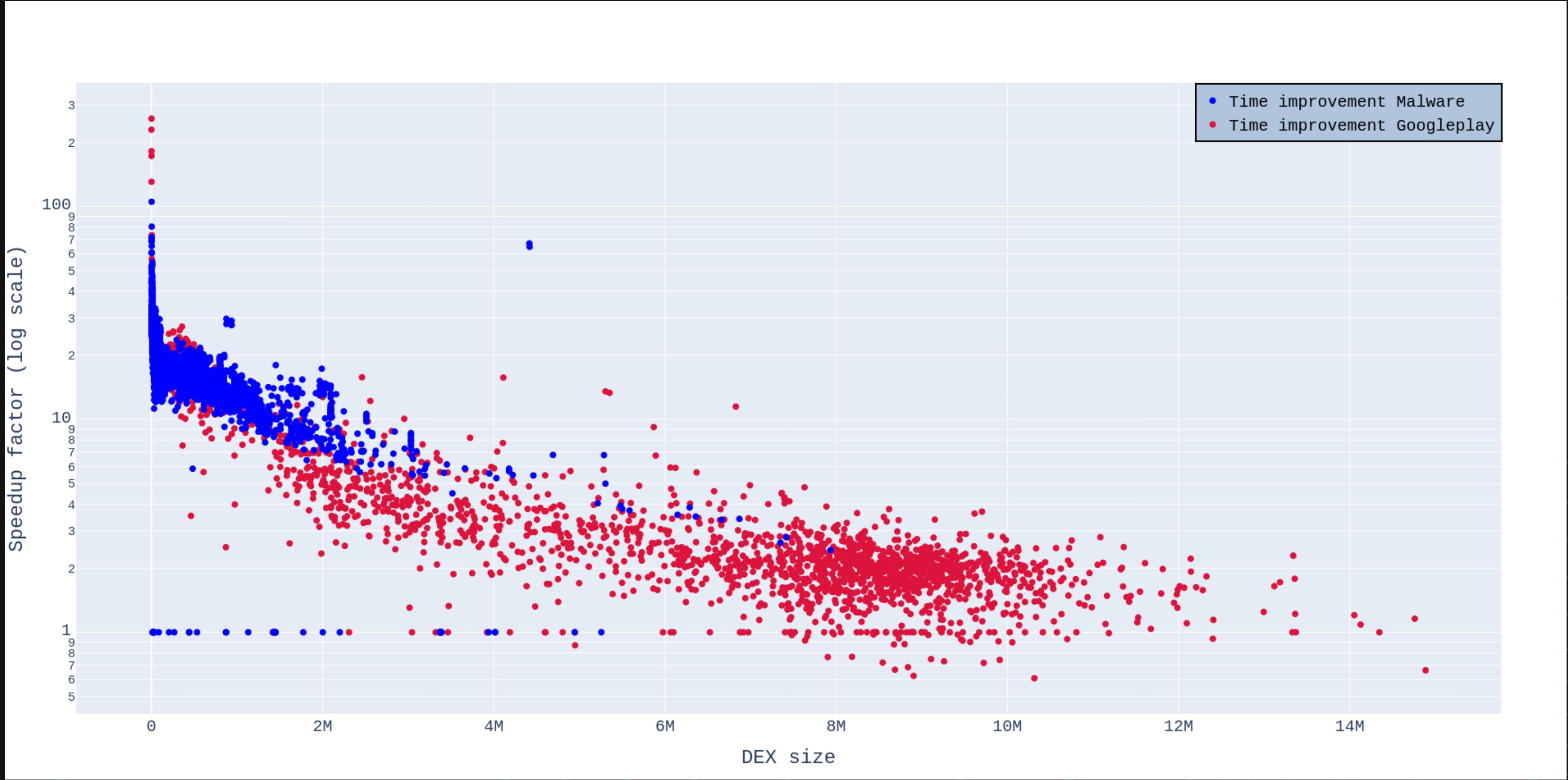
# Kunai Lib

## INTRODUCTION TO KUNAI



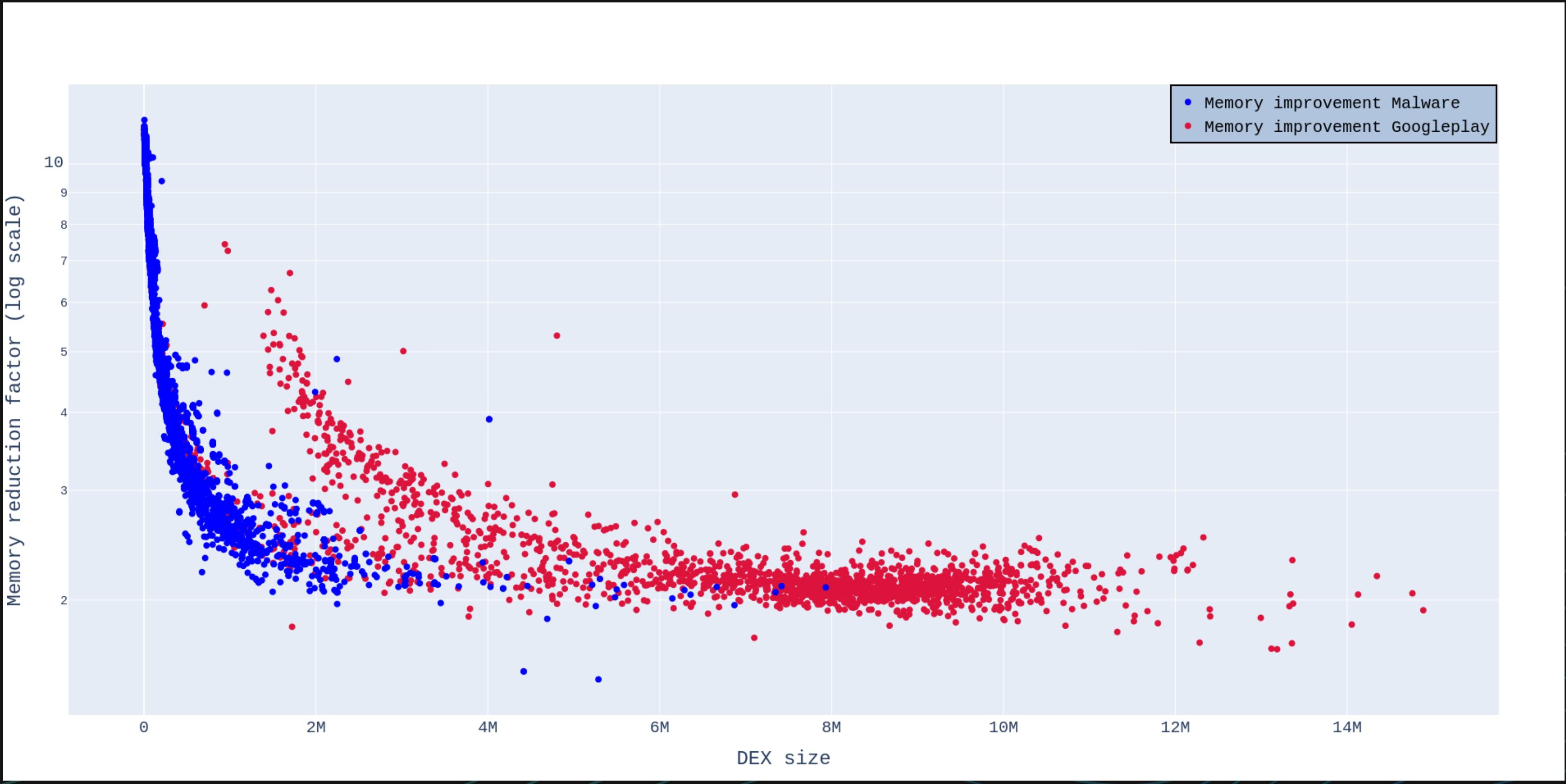
# Kunai Lib

## INTRODUCTION TO KUNAI



# Kunai Lib

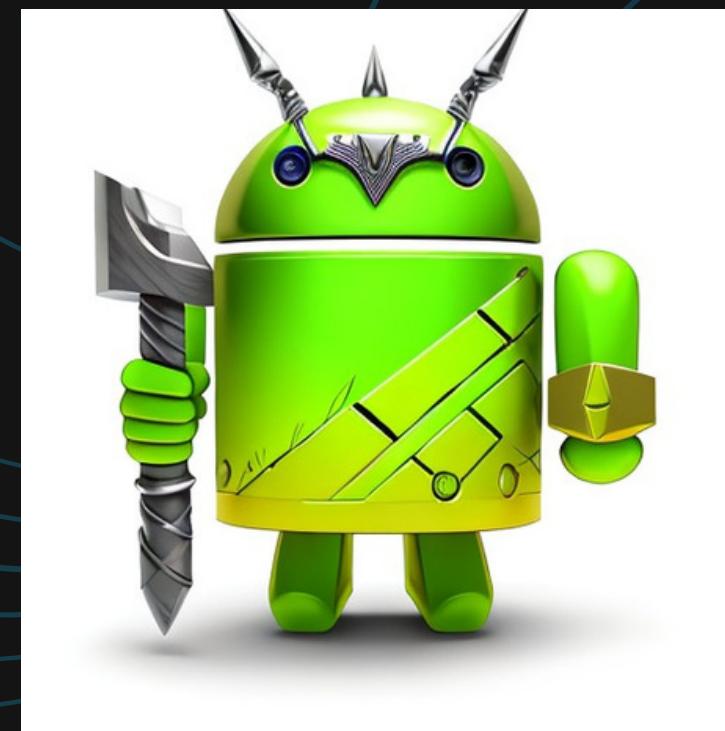
## INTRODUCTION TO KUNAI



# 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

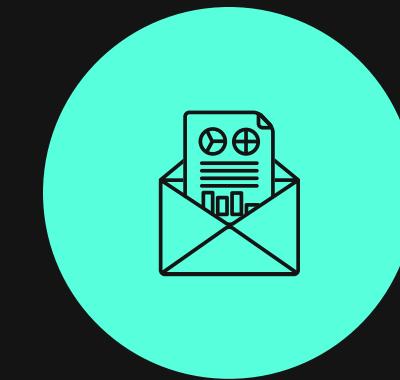
## MJOLNIR

```
"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<"Ljava/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>