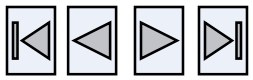
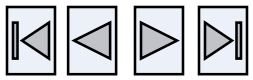
Android Platform Architecture

CSCI3310 Mobile Computing & Application Development



Outline

- Overview of Android
- Android OS Layers
- Dalvik Virtual Machine
- JIT vs AOT



What is Android?

- **Android** is a *Linux-based* platform for *mobile devices* …
  - *Operating System*
  - *Middleware*
  - *Applications*
  - *Software Development Kit* (**SDK**)
- Which kind of **mobile devices** … (examples)

SMARTPHONES

TABLETS

ANDROID TV

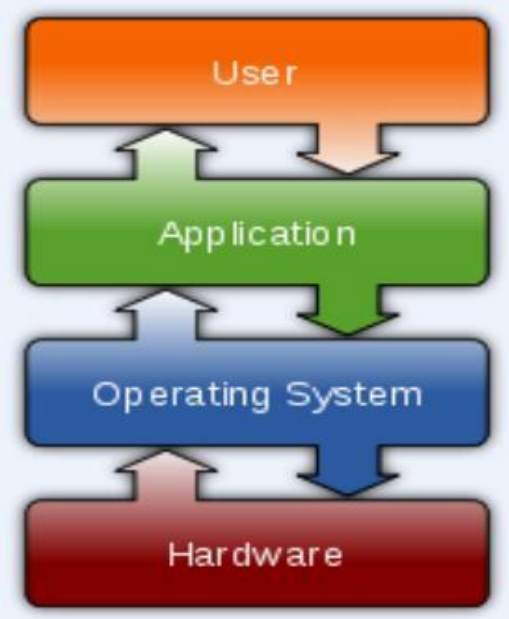
ANDROID Auto

Wear OS

ANDROID Things

What is Mobile OS?

- An operating system (OS) is an interface between hardware and user. It manages hardware and software resources of the system
- Mobile OS combines features of a PC OS with other features for handheld use
  - Touchscreen
  - Wireless connections
  - GPS
  - Camera
  - Speech recognition
  - …



Android History

- October 2003 - **Android Inc.** founded by Andy Rubin, Rich Miner, Nick Sears and Chris White
- August 2005 - **Google acquired** Android Inc.
- November 2007 - **OHA formed**
- September 2008 - Android 1.0 released

Open Handset Alliance (OHA)

Android sweet names


Alpha


Beta


Cupcake


Donut

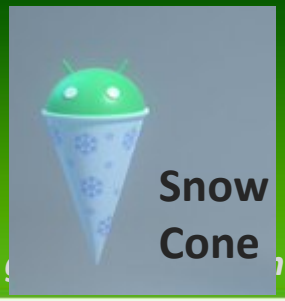

Eclair


Froyo


Gingerbread


Quince Tart


Red Velvet Cake


Snow Cone


Honeycomb


Ice Cream Sandwich


Jelly Bean


KitKat


Lollipop


Marshmallow

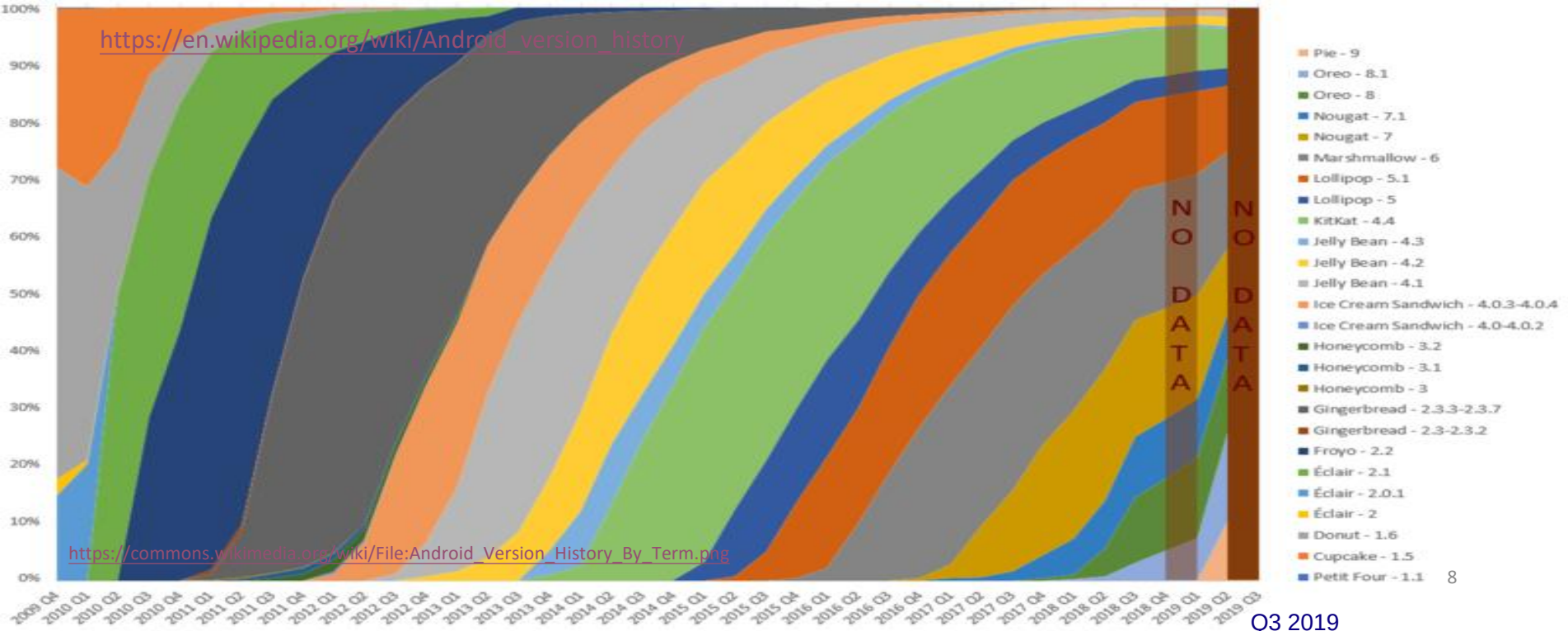

Nougat


Pie


Oreo

Android

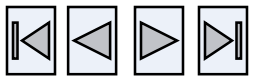
- Most updated Android 12 (Oct 2021)
- As of 2021, Android has over 2.5 billion monthly active users - the largest installed base of any OS



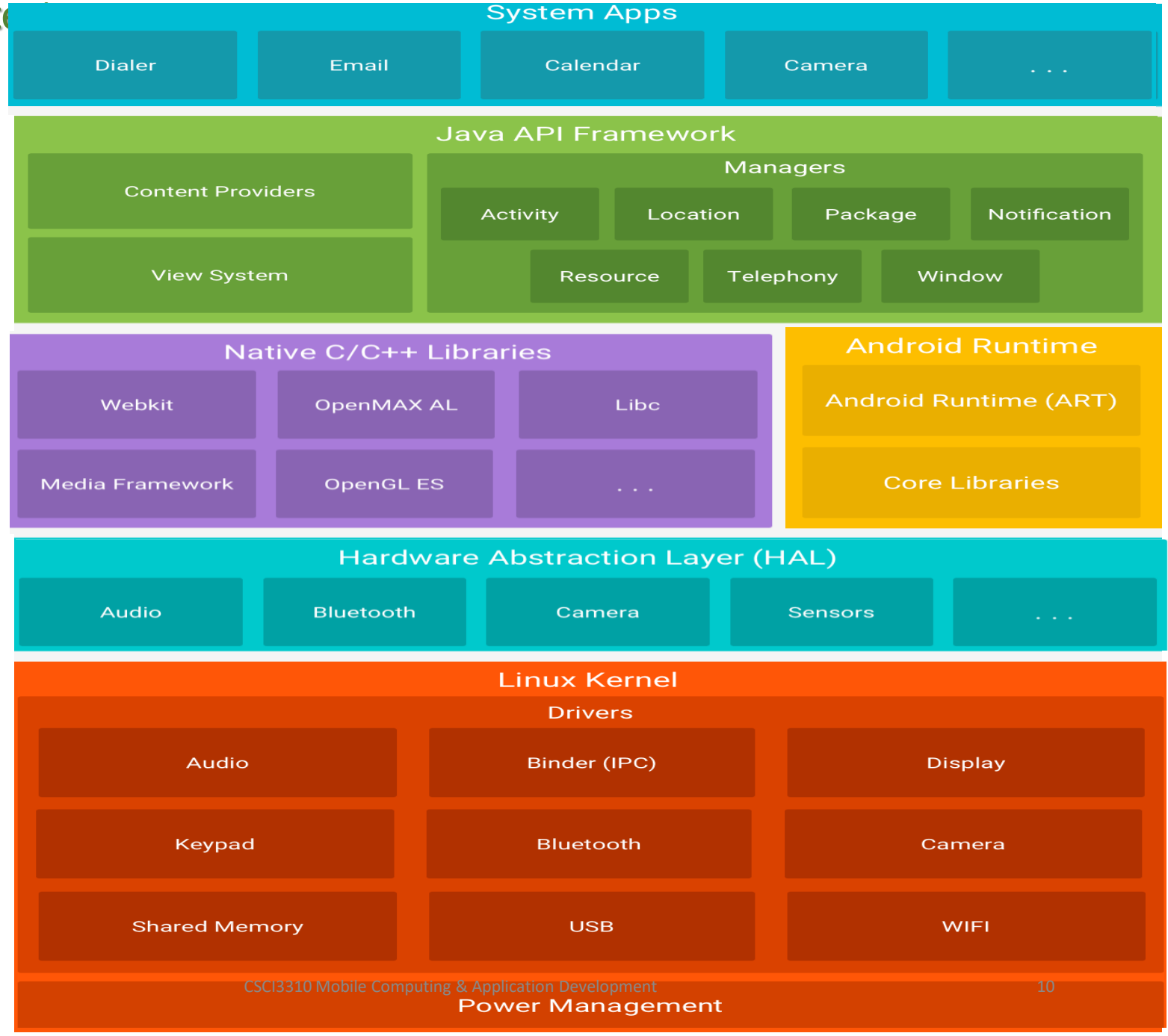
Android

- Google has released most of the Android code under the **Apache License**
  - *vendors can add proprietary extensions without submitting those back to the open source community*
- Operating system based on Linux kernel with a **Java / C++** programming interface
- Every Android app is isolated from other running applications during execution know as *Application Sandbox*

Moto "All applications are created equal"



Android Archite

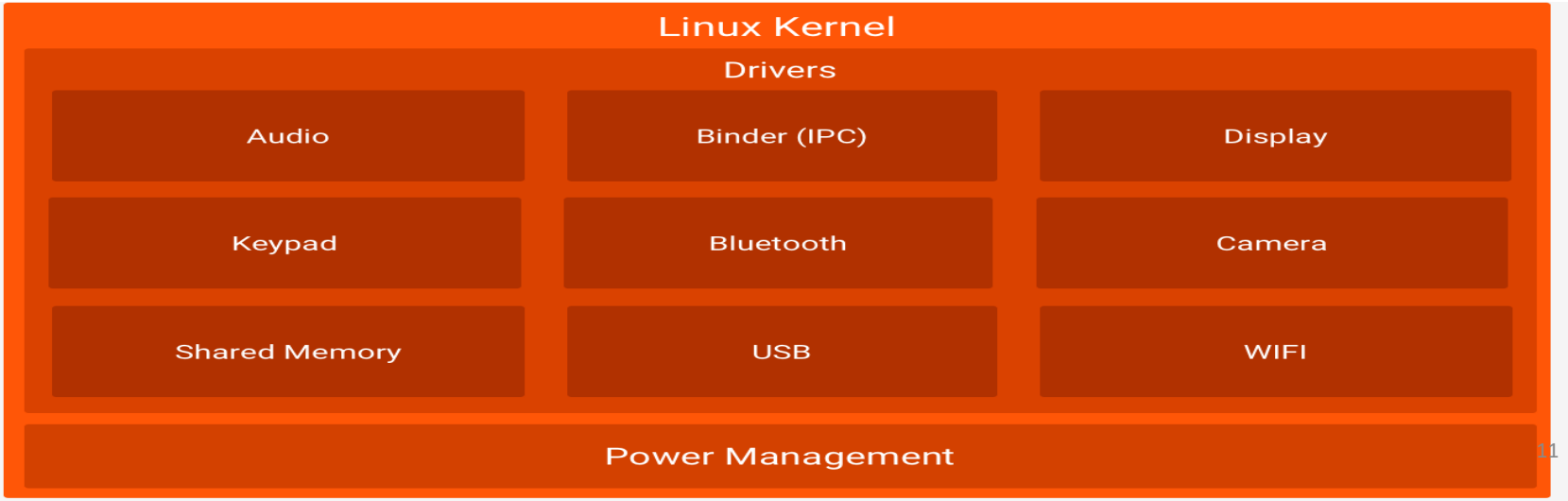


Linux Kernel

- Android is Linux kernel based, but not the traditional Unix-like:
  - Does not include the GNU C Library
  - Uses Bionic for its smaller runtime footprint and optimization for low-frequency CPUs.

Code name	Version#	API level	Linux kernel
	1	1	?
No codename	1.1	2	2.6
Cupcake	1.5	3	2.6.27
Donut	1.6	4	2.6.28
Eclair	2.0 – 2.1	5 – 7	2.6.29
Froyo	2.2 – 2.2.3	8	2.6.32
Gingerbread	2.3 – 2.3.7	9 – 10	2.6.35
Honeycomb	3.0 – 3.2.6	11 – 13	2.6.36
Ice Cream Sandwich	4.0 – 4.0.4	14 – 15	3.0.1
Jelly Bean	4.1 – 4.3.1	16 – 18	3.0.31 to 3.4.39
KitKat	4.4 – 4.4.4	19 – 20	3.10
Lollipop	5.0 – 5.1.1	21 – 22	3.16
Marshmallow	6.0 – 6.0.1	23	3.18
Nougat	7.0 – 7.1.2	24 – 25	3.18 to 4.4
Oreo	8.0 – 8.1	26 – 27	3.18 to 4.9
Pie	9	28	4.4 to 4.14
Android 10	10	29	4.9 to 4.19
Android 11	11	30	4.14 to 4.19
Android 12	12	31	4.19 to 5.10

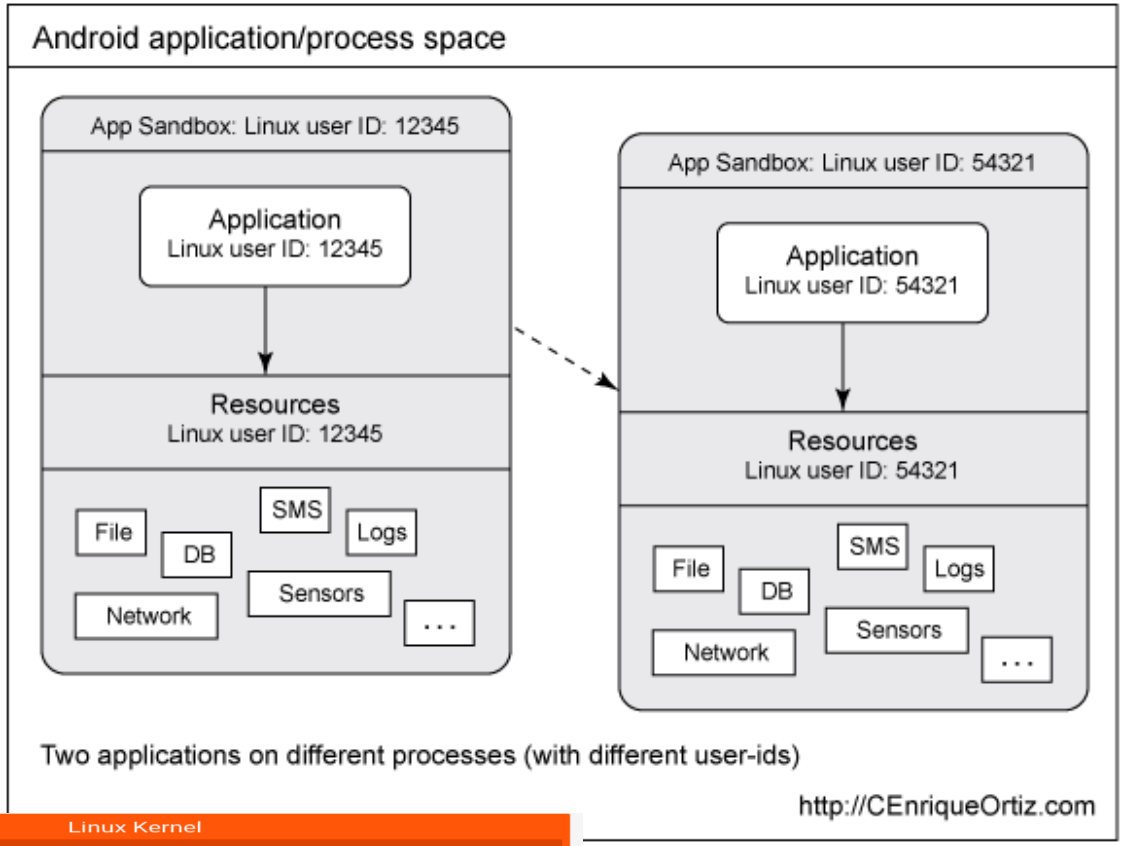
https://source.android.com/devices/architecture/kernel/android-common

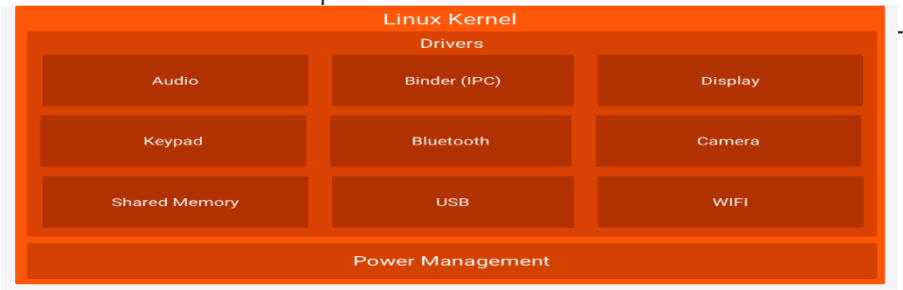


Linux Kernel

use of linux user in application sandbox

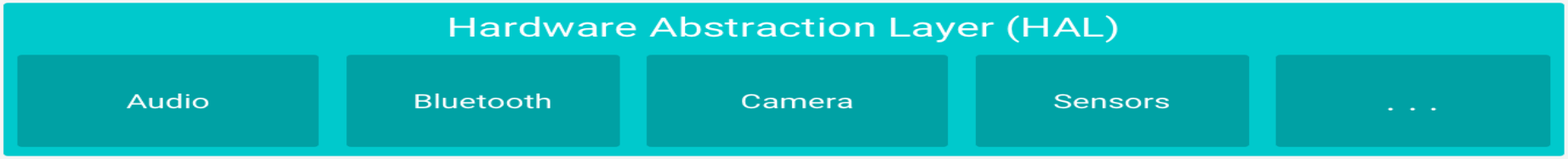
- Extensible mechanism for secure IPC
- Process isolation





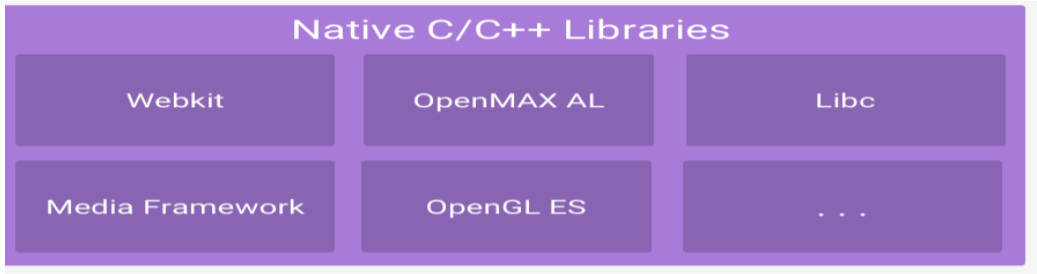
Hardware Abstraction Layer

- provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework.
- consists of multiple library modules, implementing an interface specific types of **hardware component**, e.g. camera or bluetooth
- when API access device hardware, system loads the library module for that hardware component.



Native C/C++ Libraries

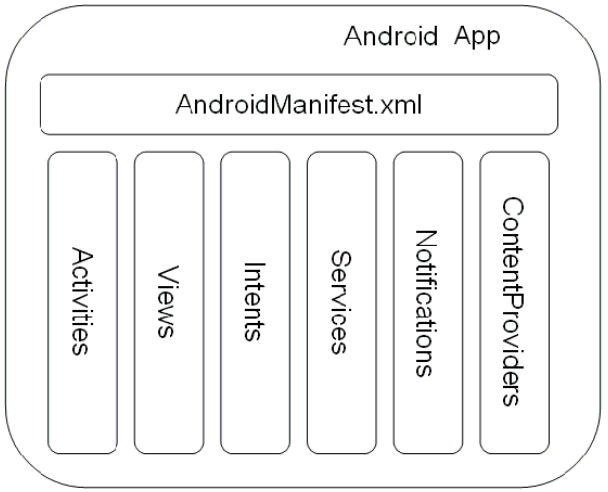
- Many core Android system components and services, such as ART and HAL, are built from native code that require native libraries written in C/C++
- Android's Java APIs to expose the functionality of some native libraries, e.g. OpenGL ES to **apps supporting 2D/3D graphics**
- To develop app requiring C/C++, use the **Android NDK** to access some of these native platform libraries directly from your native code.

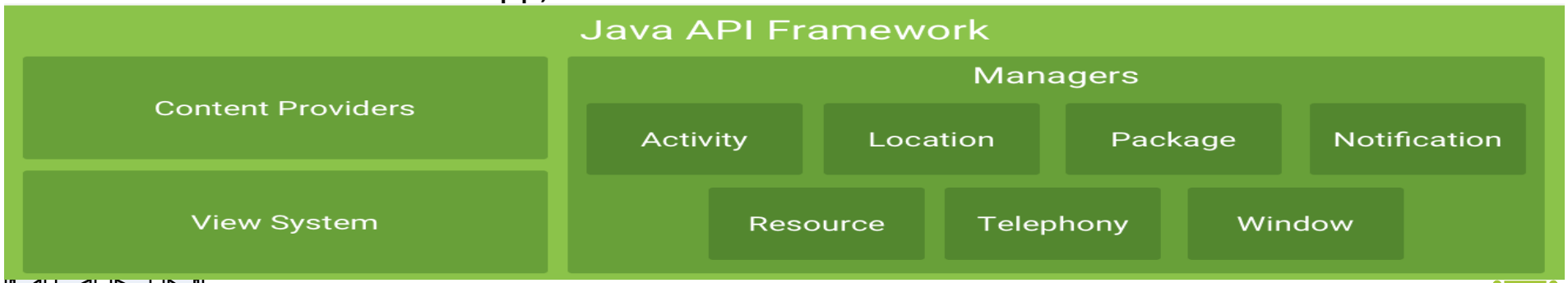


Java API Framework

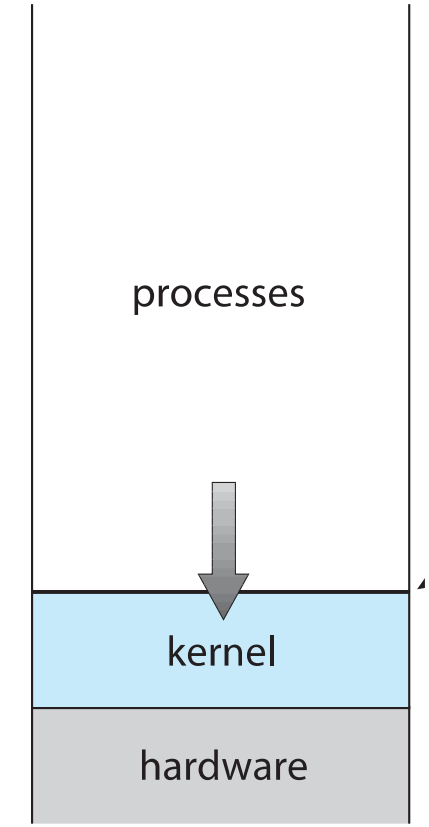
Android SDK provide feature-set of the Android in Java APIs:

- View System to build an app's UI e.g. lists, buttons etc.
- Resource Manager access to non-code resources e.g. graphics, and layout files
- Notification Manager to display alerts in the status bar
- Activity Manager to manage lifecycle of apps
- Content Providers to access data from other apps, such as Contacts app, or to share their own data

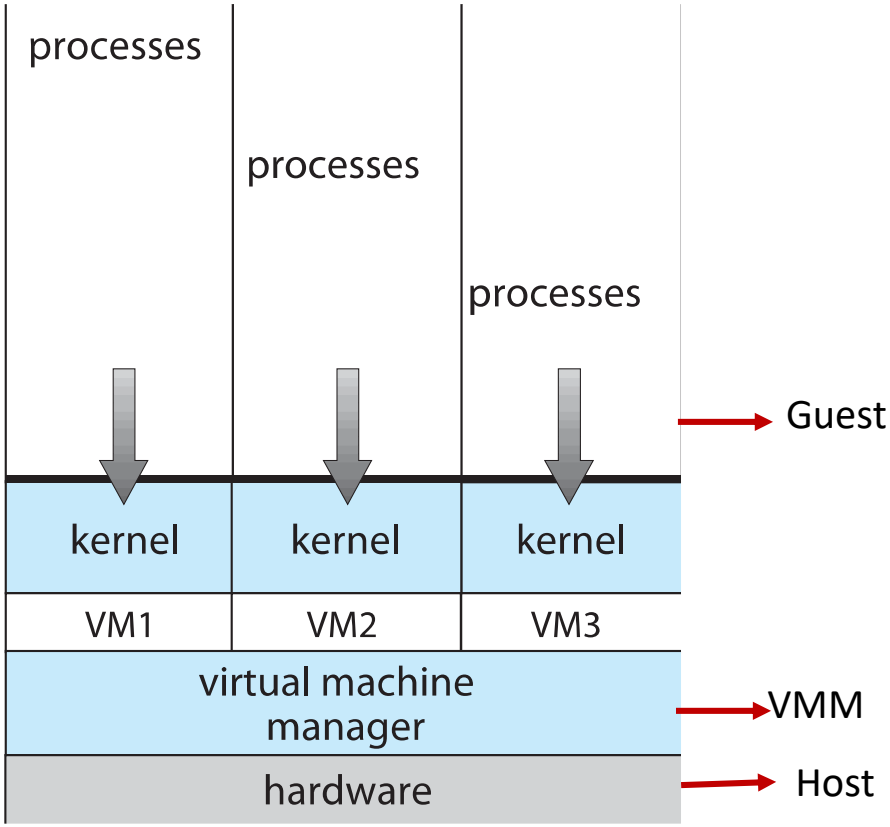




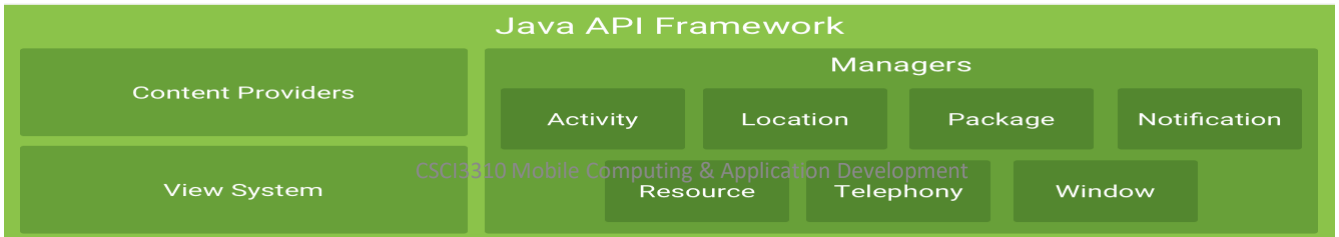
Virtual Machine (revision)



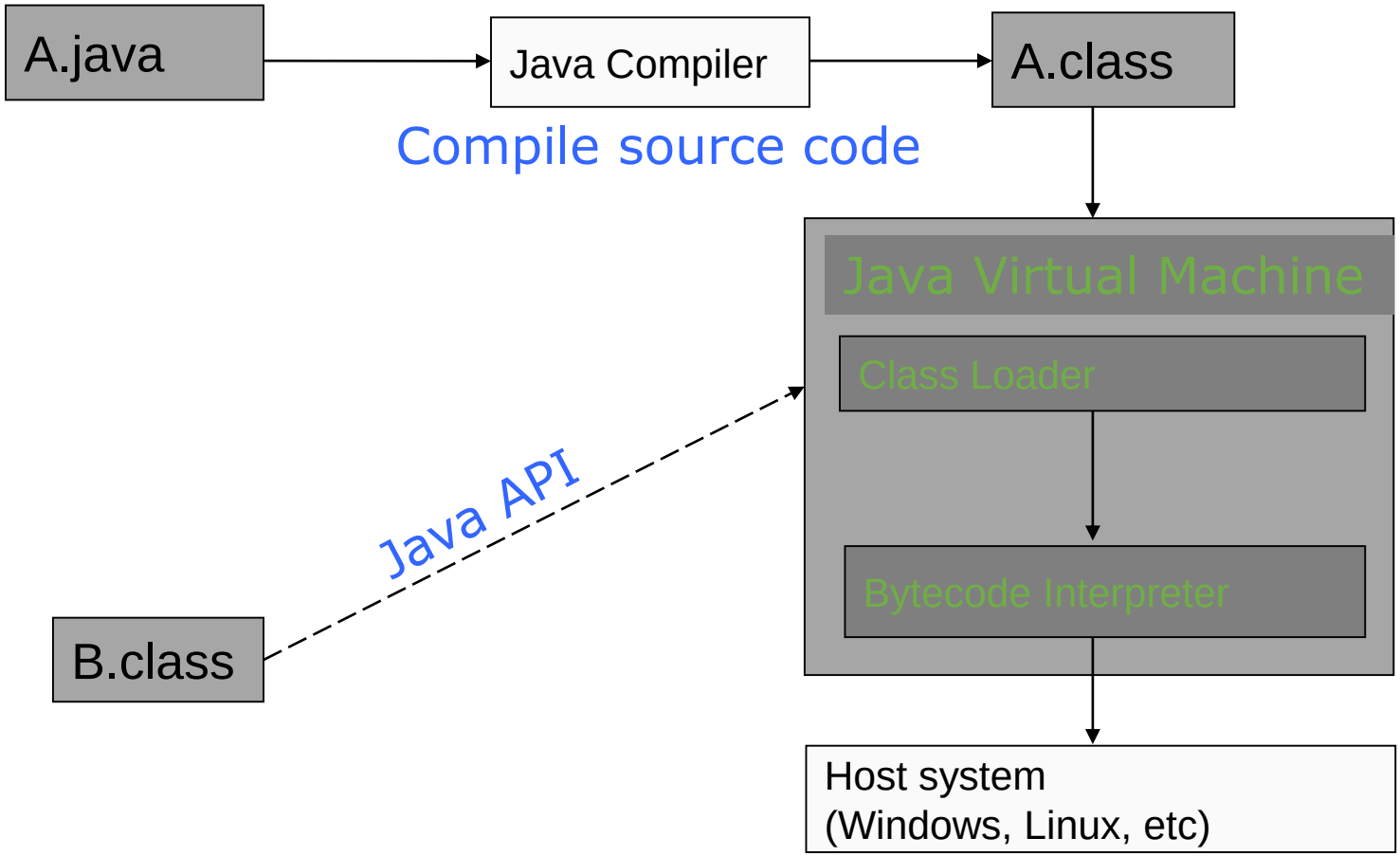
Non-virtual machine

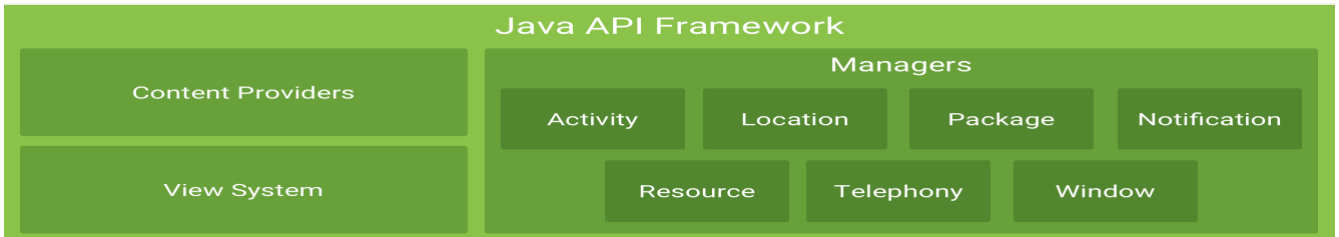


Virtual machine



JVM Workflow



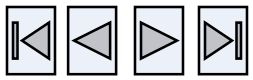


Dalvik Virtual Machine

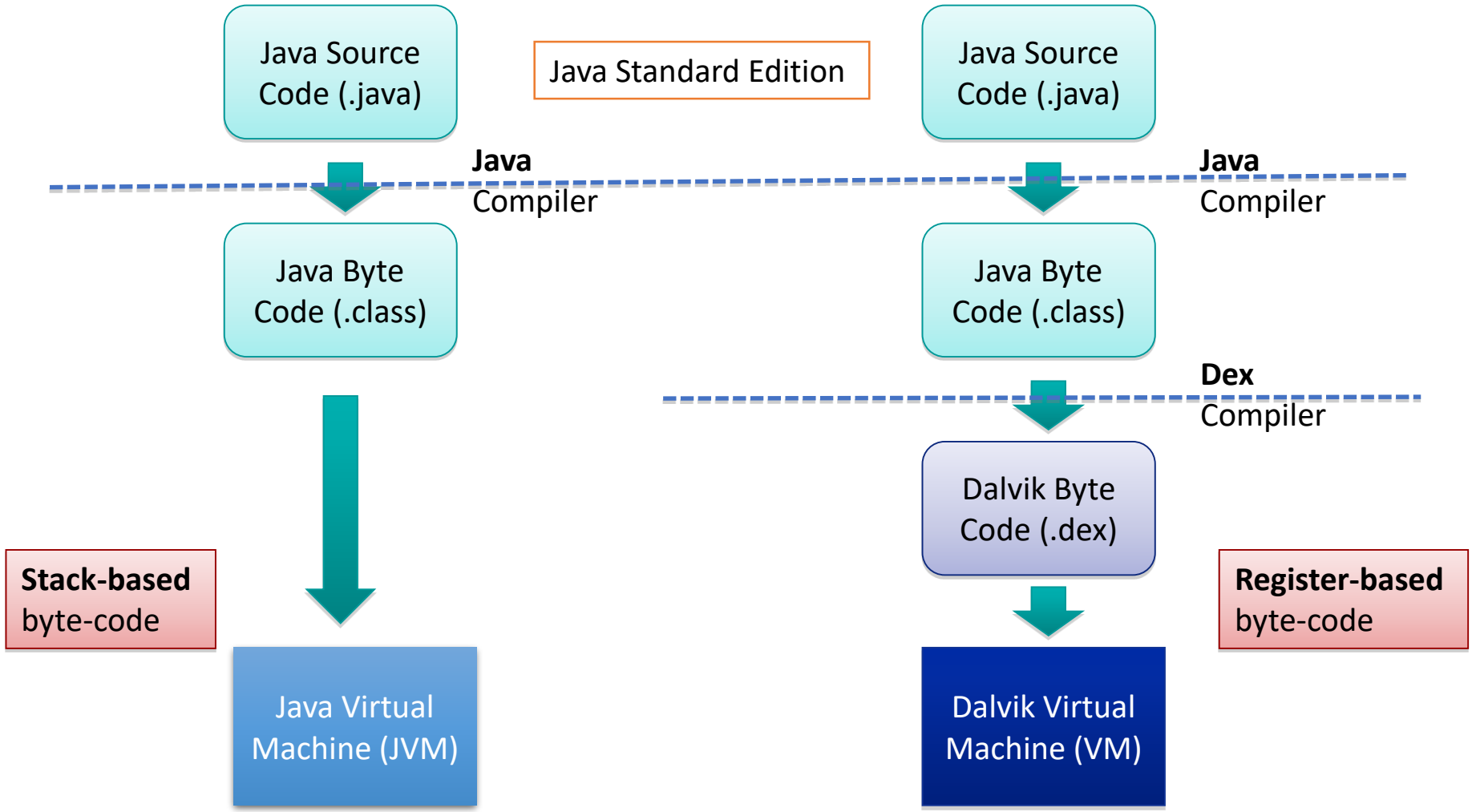
- A special process VM used by Android for versions before 4.4
- **optimized for battery-powered** mobile devices with limited memory and CPU
- **Standard Java bytecode cannot execute** on it, compact Dalvik Executable (.**dex**) format instead
- applications are packed into an **.apk** (Android Package) file and deployed
- Only reuse Java syntax, **no compatibility with Java** SE/ME
- After Android 2.2, Dalvik implemented just-in-time compiler (JIT) boasting further the performance

Apk file

- Apk file - Android Package (APK) file
  - a variant of the JAR format for the distribution and installation of bundled components.
  - contains the following folders:
    - META-INF
    - res
  - and files:
    - AndroidManifest.xml (application components definitions)
    - classes.dex (dex code for installation)
    - resources.arsc (android compiled resources)



Dalvik Virtual Machine vs JVM



Dalvik Bytecode VS Java Bytecode

Java source code

```
public int method( int i1, int i2 ) {
  int i3 = i1 * i2;
  return i3 * 2;
}
```

Java bytecode

```
method public method(II)I
        iload  _1
        iload  _2
        imul
        istore _3
        iload  _3
        iconst _2
        imul
        ireturn
.end method
```

Dalvik bytecode

```
method public method(II)I
        mul-int v0,v2,v3
        mul-int/lit-8 v0,v0,2
        return v0
.end method
```

Stack-based byte-code

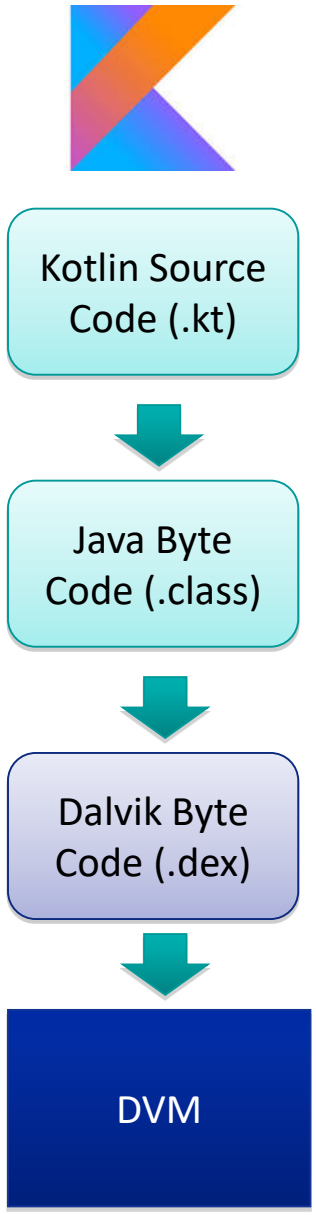
Register-based byte-code

Both use JIT (Just In Time) compiler to compile source code into machine code during runtime.

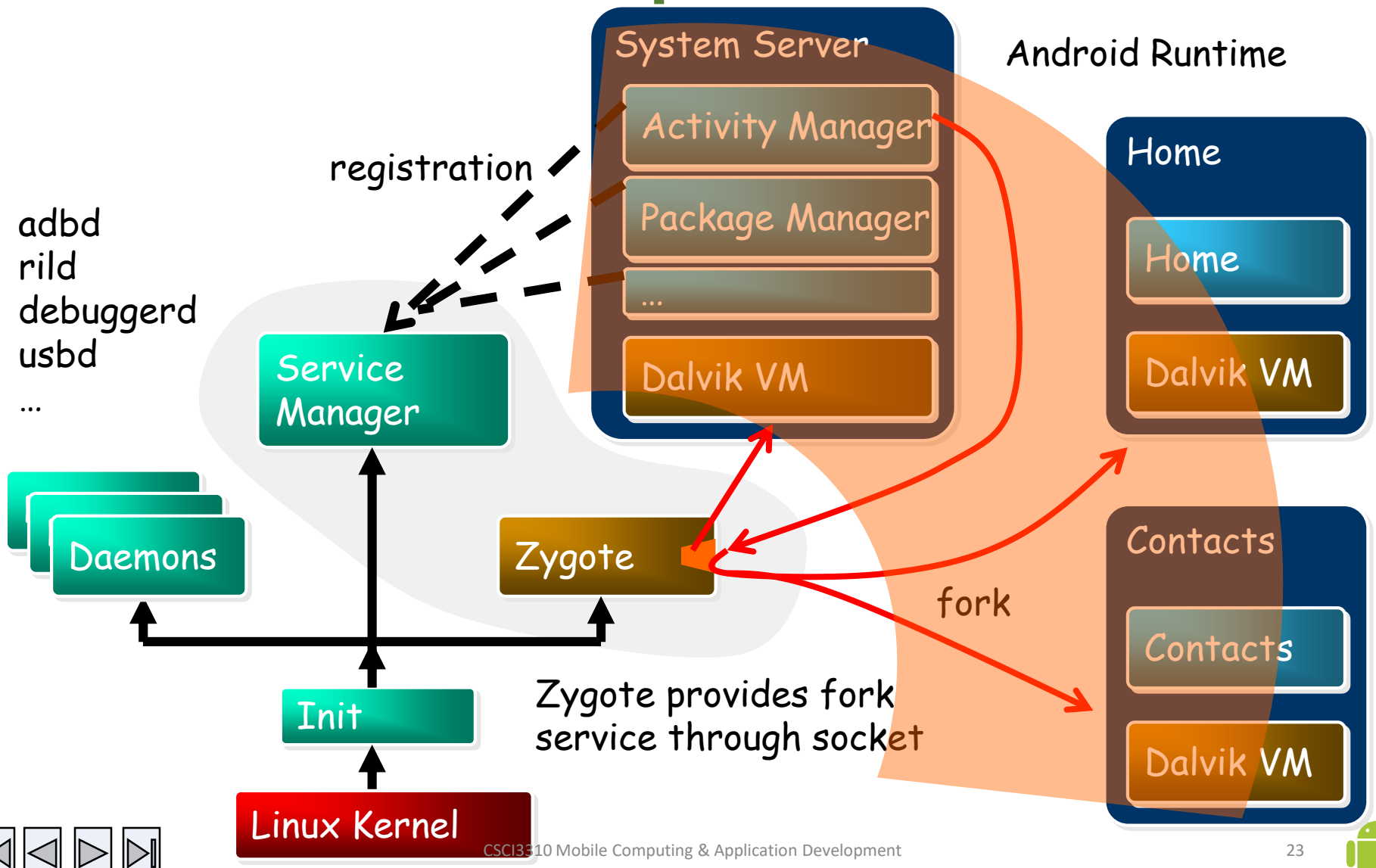
Kotlin for Android

Kotlin is officially supported by Android in May 2019

- JVM-language
  - can be compiled to Java bytecode (.class) that runs on the JVM
  - Essentially mean it can be converted to DEX that runs on the DVM
- 100% interoperable with the Java:
  - Call Java-based code from Kotlin, or call Kotlin from Java-based code.



Android Boot Sequence



Operating Environment

- Can run **multiple Dalvik virtual machines simultaneously**
- **Each Android application** executes **its Dalvik Executable** file(.dex) **on its own Dalvik VM**.
  - Dx : compiles Java file into dex file.

- Designed for embedded environment
  - Supports **multiple virtual machine** processes per device
  - Highly **CPU-optimized** bytecode interpreter
  - Efficiently using runtime **memory**

- Core Libraries
  - Core APIs for Java language provide a powerful, yet simple and familiar development platform

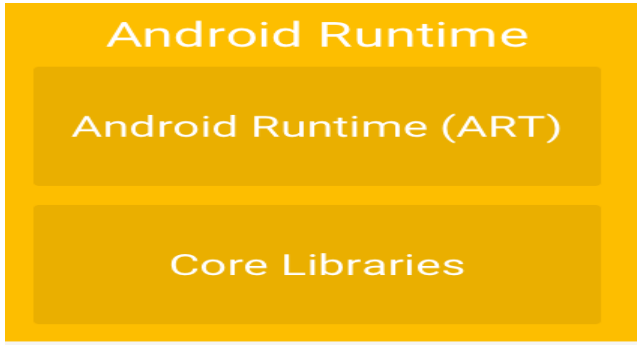
Dalvik Virtual Machine

Problems

1. **Byte code execution** is interpreted, causing **performance hit**

2. App requirements further complicated the hardware as well as platform, causing its performance **can't scale up** with multi-core architecture

Android Runtime (ART)



Ahead-of-time (AOT) compilation

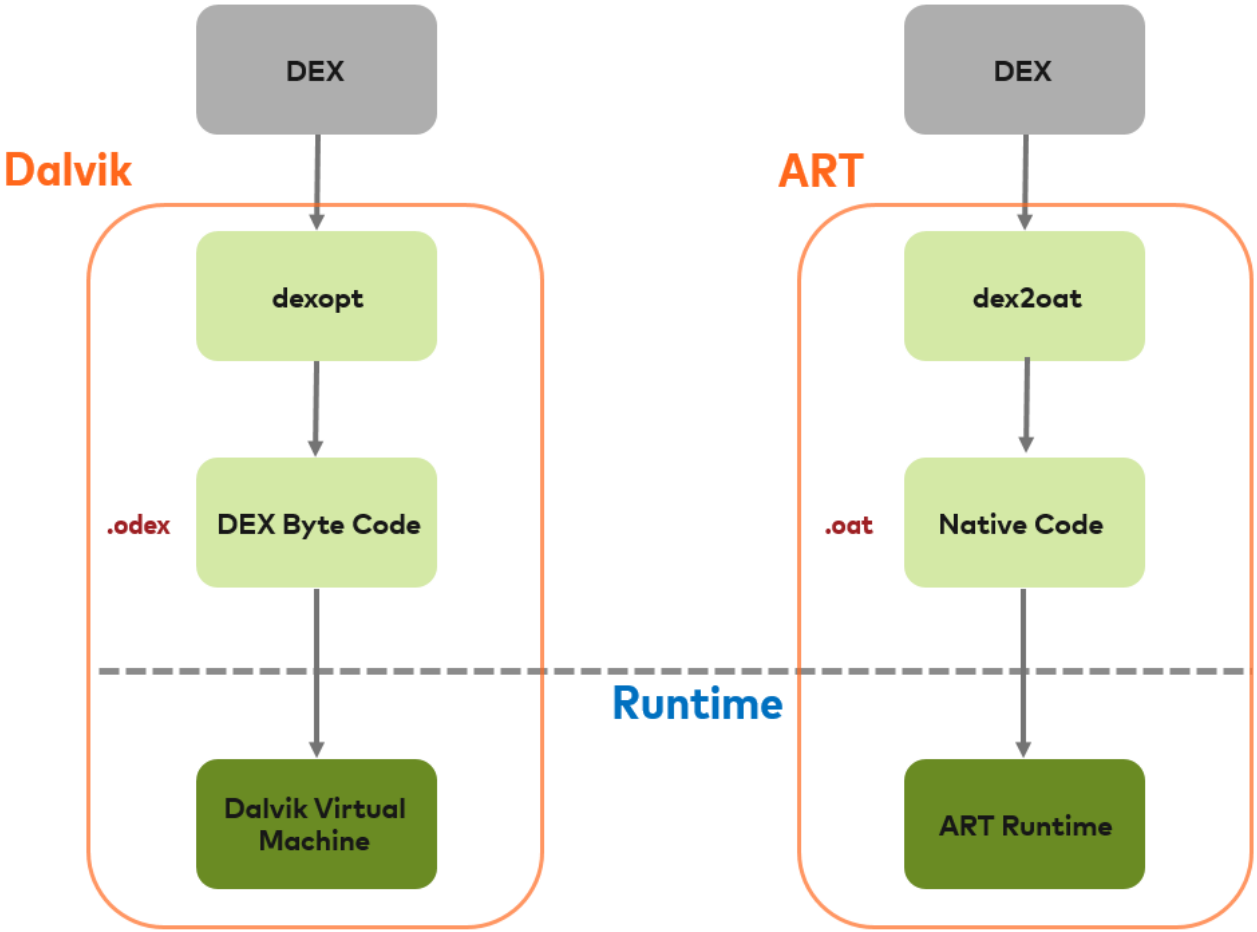
- Generates a compiled app executable (entire) for the target device by on-device **dex2oat** tool
- Introduced in the 4.4 & later to replace direct use of Dalvik VM

Pros:

- Improve app performance: skips bytecode interpretation
- Smaller memory footprint: about twice less space occupying compared to DVM
- Improved garbage collection

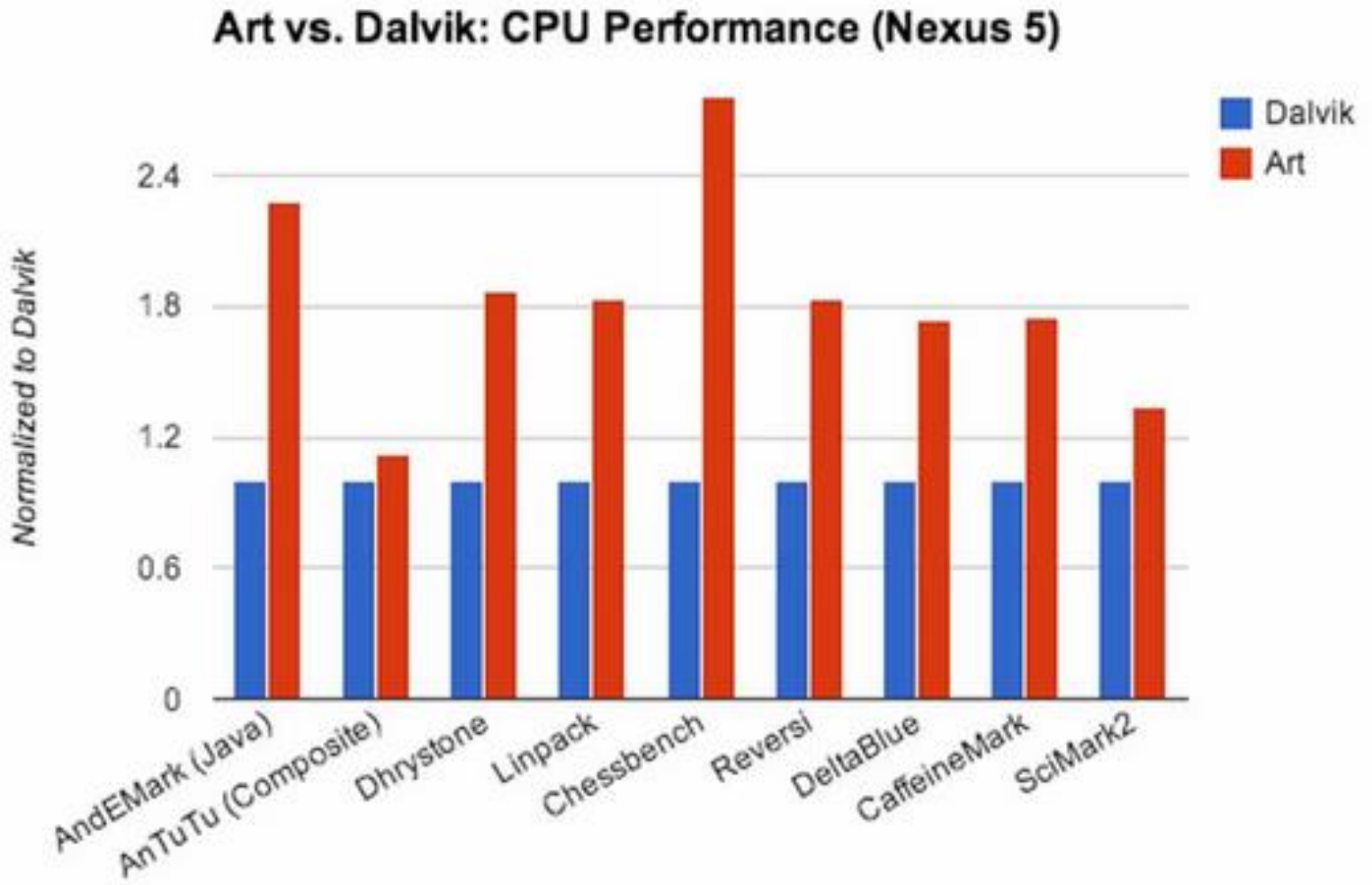
Cons:

- One-time compilation takes more time to complete
- Take more time in device's first boot and an app's first start-up

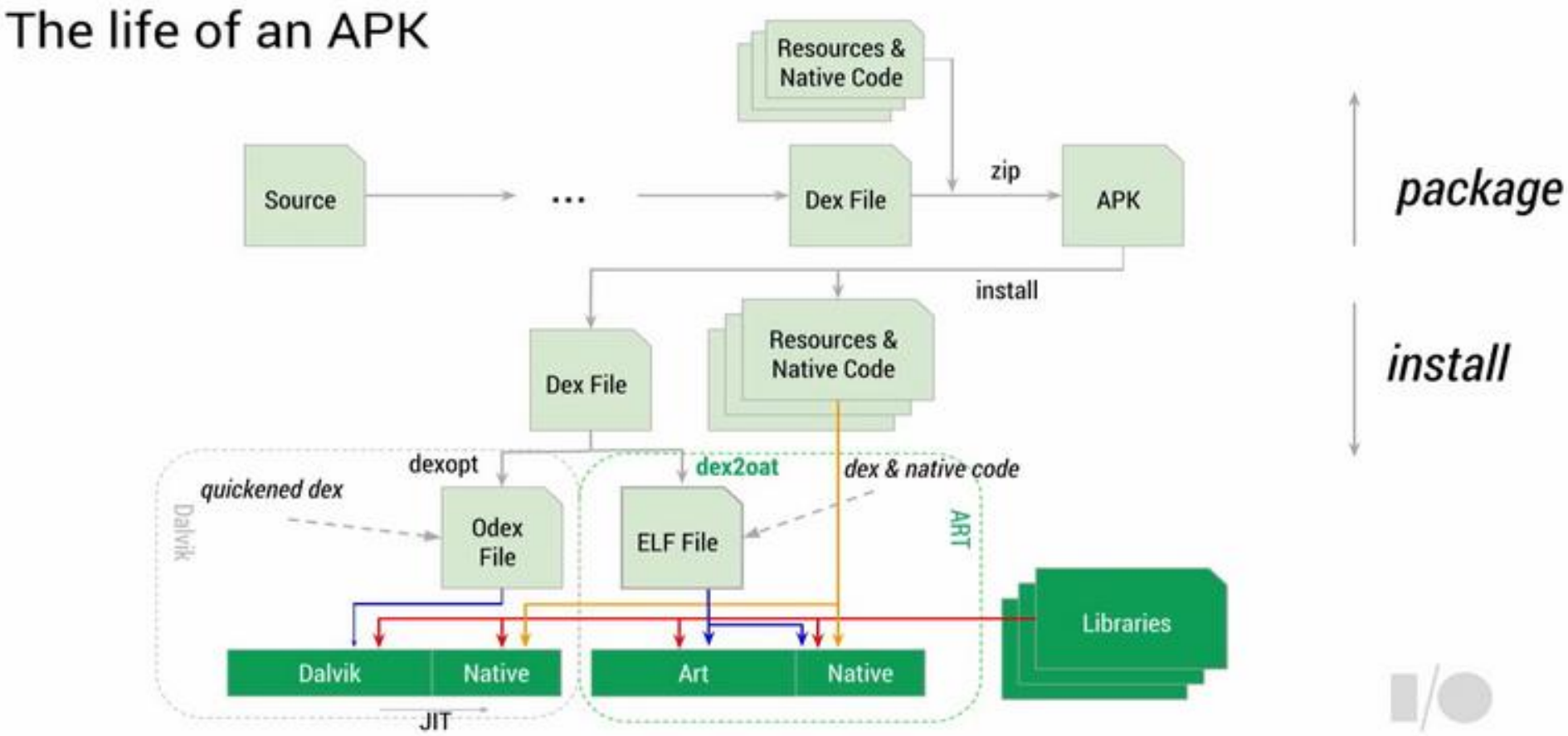


Dalvik vs ART

ART


Art vs. Dalvik: CPU Performance (Nexus 5)
Dalvik
Art
Normalized to Dalvik
2.4
1.8
1.2
0.6
0
AndEMark (Java)
AnTuTu (Composite)
Dhrystone
Linpack
Chessbench
Reversi
DeltaBlue
CaffeineMark
SciMark2

ART

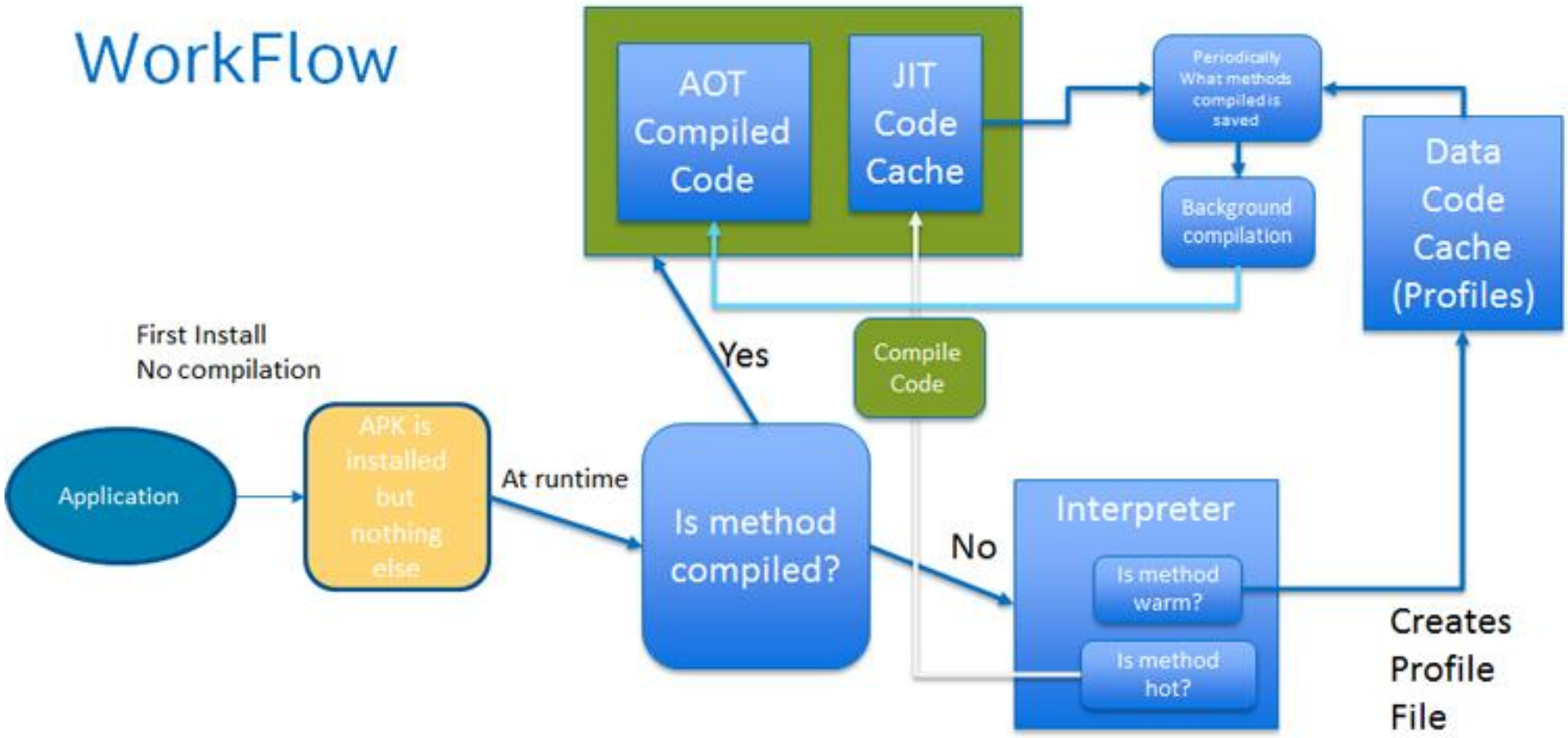

The life of an APK
Resources & Native Code
Source
...
Dex File
zip
APK
package
install
Dex File
Resources & Native Code
install
dexopt
dex2oat
dex & native code
quickened dex
Dalvik
Odex File
ELF File
ART
Libraries
Dalvik
Native
Art
Native
JIT

Reintroducing JIT

- Android 7.0 reintroduced JIT compilation along with AOT, and an interpreter in the ART
  - The new JIT constantly does profiling and improves applications as they run.
  - Also tackle initial installation time and memory issues.

- In **Hybrid Runtime**,
  - No compilation during install: apps can be started right away when bytecode is interpreted.
  - When the device is idle, a daemon runs **AOT-compile** frequently used code based on a profile generated during the first runs.

Hybrid AOT/JIT Runtime



Reference

- Android Platform Architecture
  https://developer.android.com/guide/platform/index.html

- JVM Instruction Set & Dalvik-bytecode
  https://docs.oracle.com/javase/specs/jvms/se7/html/jvms-6.html
  https://source.android.com/devices/tech/dalvik/dalvik-bytecode