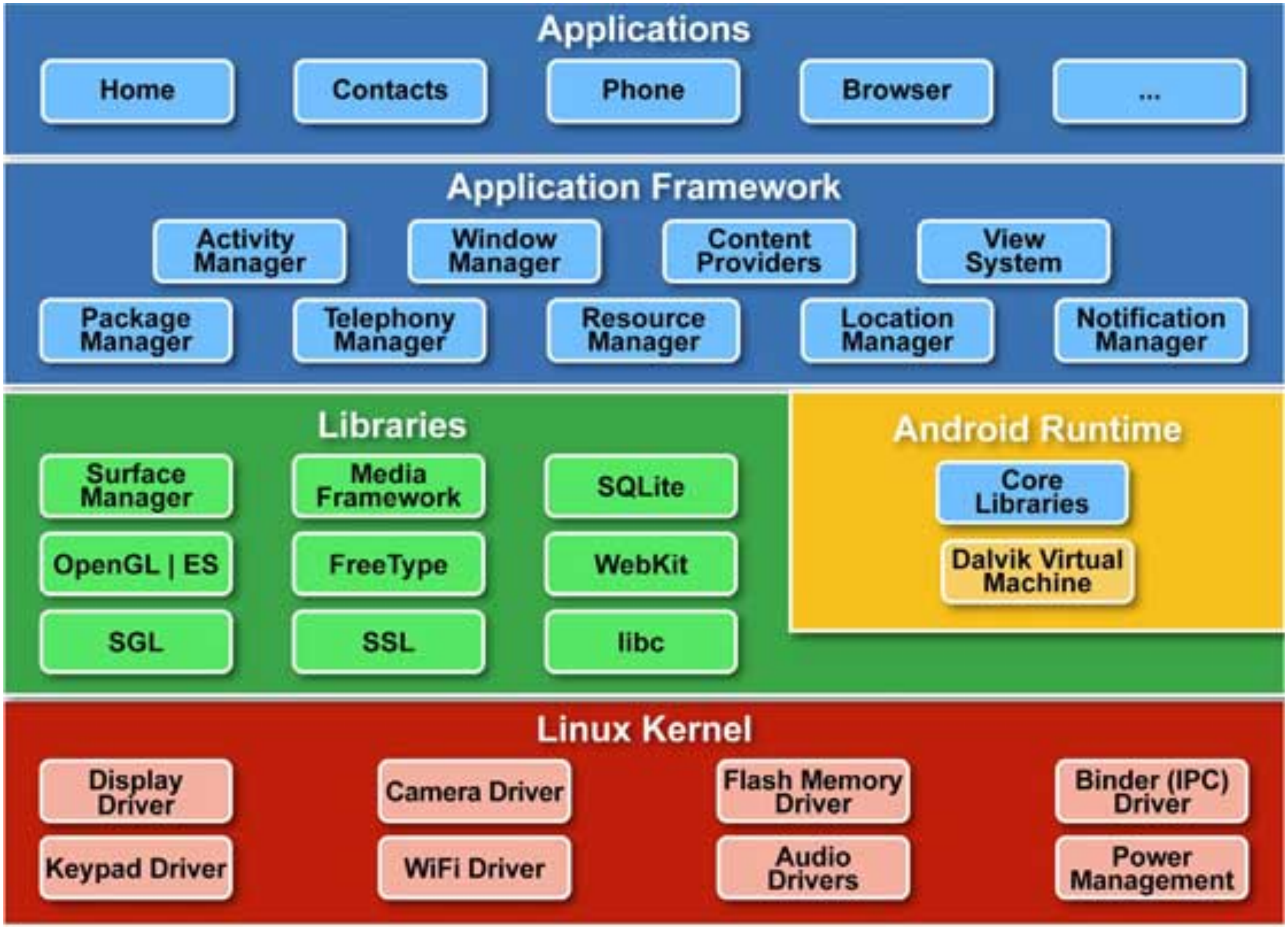
iOS Architecture

CS 4720 – Mobile Application Development

Remember This?



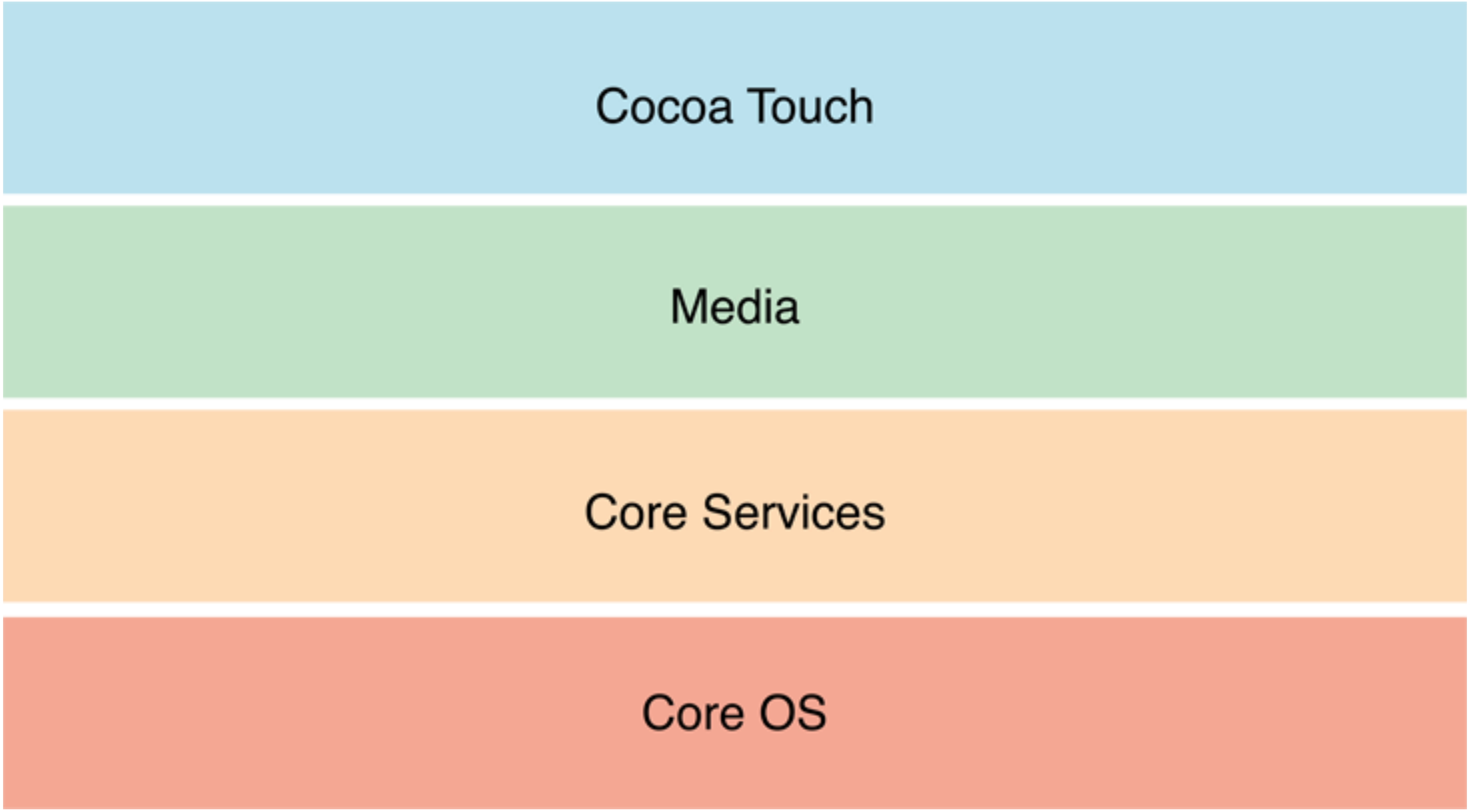
Remember This?

- Android looks like it has a lot of layers
- A lot of this comes from its design mentality
- Plus Java on top of Linux
- But it really does just boil down to:
  - User Apps on top of
  - The main frameworks on top of
  - The main libraries (working with Java) on top of
  - The Linux kernel

iOS Architecture

- iOS is also *nix based
- iOS comes directly from the development path of OS X
- Cocoa became Cocoa Touch – the main API for apps to interface with the OS
- iOS has similar layers, but they are not as “rigid”

iOS Architecture



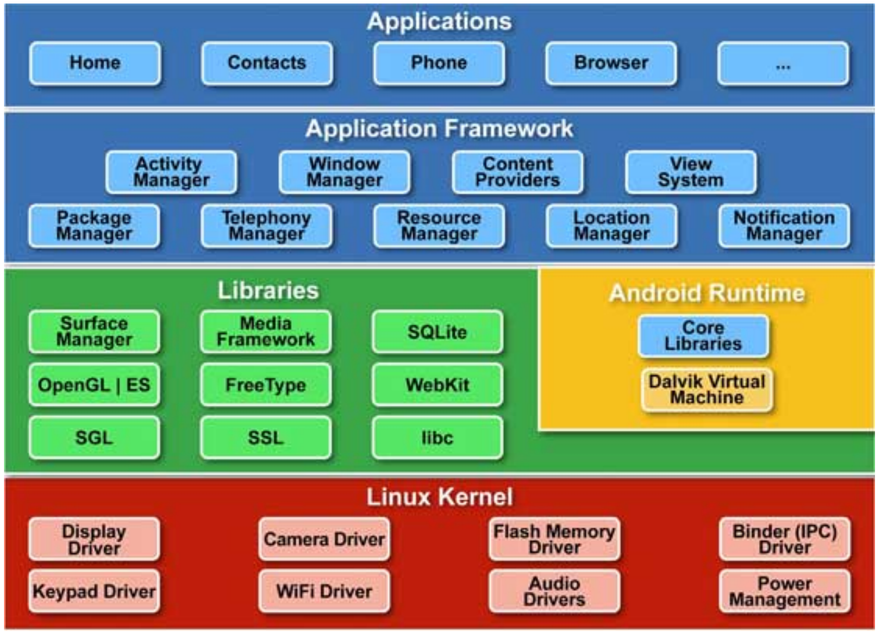
Cocoa Touch

Media

Core Services

Core OS

Let’s Compare!



Applications

Home

Contacts

Phone

Browser

...

Application Framework

Activity Manager

Window Manager

Content Providers

View System

Package Manager

Telephony Manager

Resource Manager

Location Manager

Notification Manager

Libraries

Surface Manager

Media Framework

SQLite

OpenGL | ES

FreeType

WebKit

SGL

SSL

libc

Android Runtime

Core Libraries

Dalvik Virtual Machine

Linux Kernel

Display Driver

Camera Driver

Flash Memory Driver

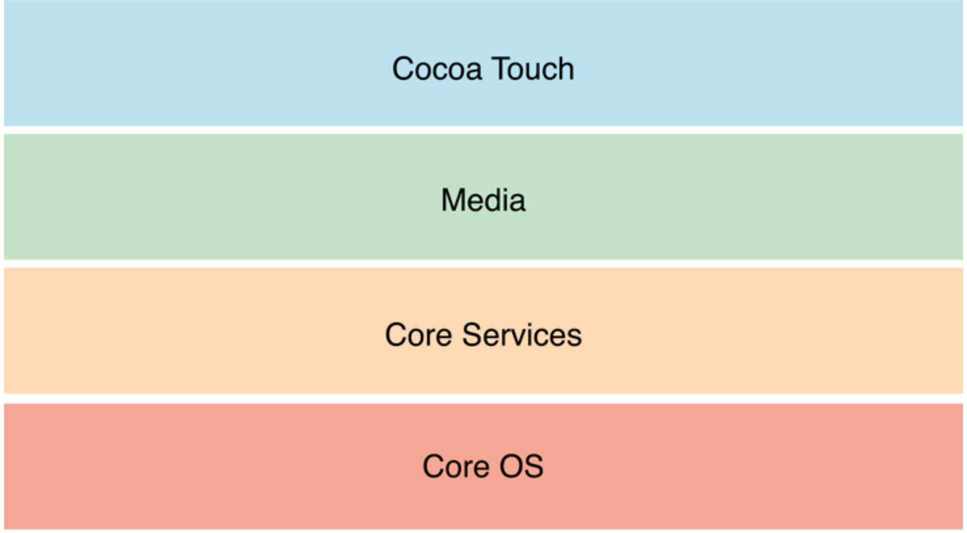
Binder (IPC) Driver

Keypad Driver

WiFi Driver

Audio Drivers

Power Management



Cocoa Touch

Media

Core Services

Core OS

What's Different?

- The layers are actual more flat
- The hierarchy here comes more from the higher levels making use of the lower levels
- All levels are accessible in similar ways
- Some have to imported more explicitly

Cocoa Touch

- Drives the UI
  - Provides the Controllers, Widgets, etc.
- Provides access to main system functions
  - Contacts, Camera, touch input, share with other apps, push notifications, etc.
- Should be the first place you look for any features you want to use in your app

Media

- The Graphics libraries live here
  - Core Graphics (Quartz), OpenGL, Metal, Photos Library, Animation
- Audio
  - Media player, OpenAL, Core Audio
- AirPlay

Core Services

- Gives access to fundamental resources needed for app
- Built on Core Foundation and Foundation frameworks
  - These are the basis for NeXTSTEP
- Networking, iCloud, Encryption, SQLite
- GPS, Telephony, SMS
- Facebook, Twitter

Core OS

- You’re probably not going to use these in your app
- Bluetooth
- USB and other accessories
- Kernel operations

The App Bundle

- In iOS (and macOS), a Bundle is actually a directory that groups a program’s resources together
- This is the .app file you find in macOS and iOS
- An .app file is zipped up in a .ipa file (this is the iOS equivalent of a .apk file)
- Let’s look at a .ipa!

Inside an iOS App Bundle

- The executable
- Information property list - Info.plist
  - Kind of like AndroidManifest, but with a bit less info
- App icons – for the home screen
- Storyboards – the layout of your screens
- Settings.bundle – the file needed to have options in Settings
- All assets go in the root folder (localized assets go in subfolders)

Main Components

- For Android, we have:
  - Activities: each screen/feature
  - Intents: messages to be passed
  - Services: background activities
  - Content Providers: data sources
  - Layouts: .xml files that represent the UI

Main Components

- Activity -> ViewController
- Intents -> Segues (kinda…), `present` other ViewControllers
- Service -> "Background Mode" + specific API calls
- Content Provider -> CoreData
- Layouts -> Storyboards and Scenes