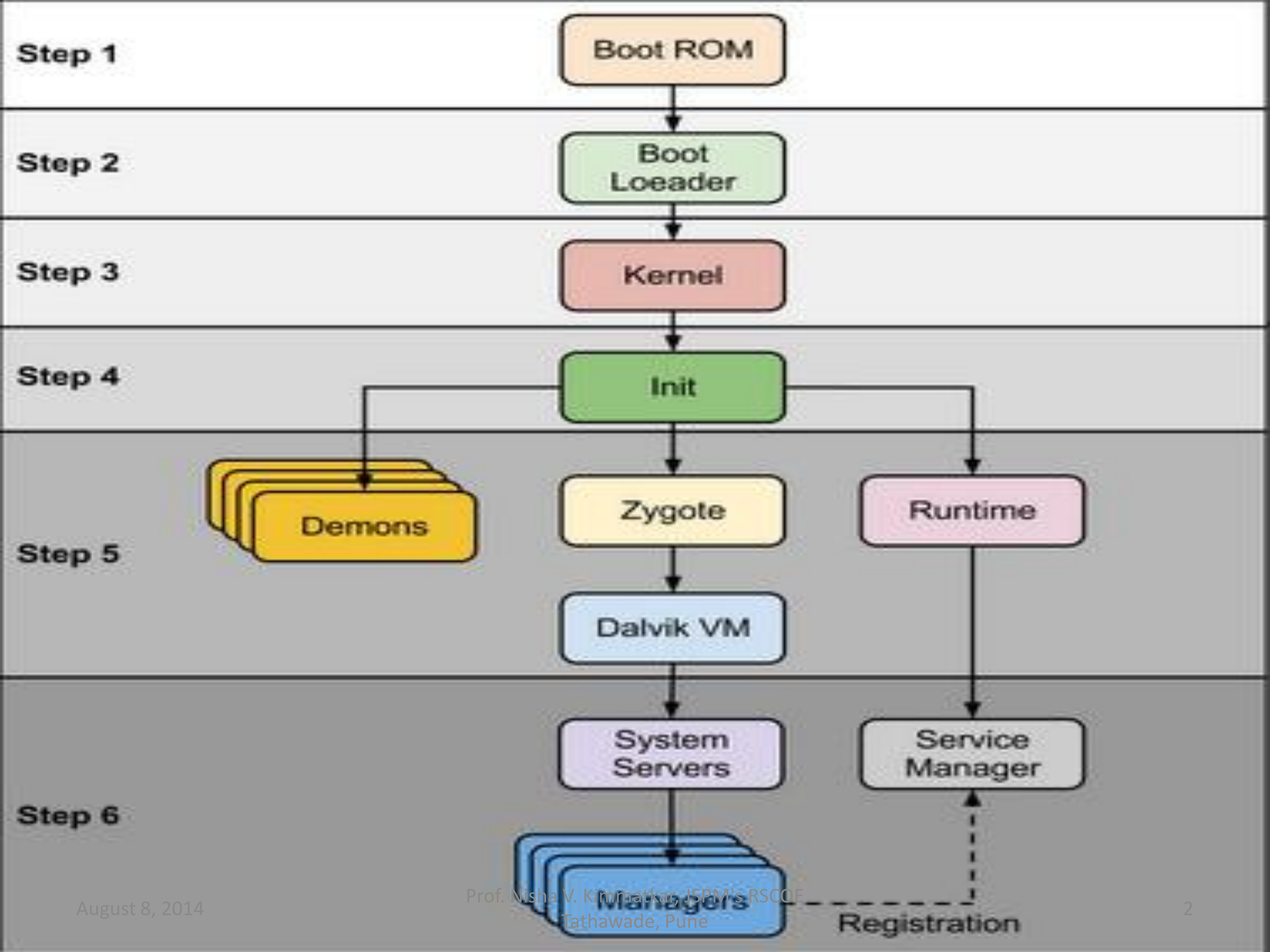
Android mobile booting sequence


Step 1
Boot ROM
Step 2
Boot Loeader
Step 3
Kernel
Step 4
Init
Step 5
Demons
Zygote
Runtime
Dalvik VM
Step 6
System Servers
Service Manager
Managers
Registration

Step 1 : Power On and System Startup

- When power start Boot ROM code start execution from pre defined location which is hardwired on ROM.
- It load Bootloader into RAM and start execution.

Step 2 : Bootloader

- Bootloader is small program which runs before Android operating system running.
- Bootloader is first program to run so It is specific for board and processor.
- Device manufacturer either use popular bootloaders like redboot,uboot, qi bootloader or they develop own bootloaders.

- **Bootloader perform execution in two stages,**

a. first stage It to detect external RAM and load program which helps in second stage,
b. In second stage bootloader setup network, memory, etc. which requires to run kernel, bootloader is able to provide configuration parameters or inputs to the kernel for specific purpose.

- Android bootloader can be found at

<Android Source>\bootable\bootloader\legacy\usbloaderlegacy

loader contain two important files that need to address here.

1. init.s - Initializes stacks, zeros the BSS segments, call _main() in main.c
2. main.c - Initializes hardware (clocks, board, keypad, console), creates Linux tags

Step 3: Kernel

1. Android kernel start similar way as desktop linux kernel starts.
2. As kernel launch it start setup cache, protected memory, scheduling, loads drivers.
3. When kernel finish system setup first thing it look for “init” in system files and launch root process or first process of system.

Step 4: init process

- init it very first process, we can say it is root process or grandmother of all processes.
- init process has two responsibilities

1. mount directories like /sys, /dev, /proc and

2. run init.rc script.

- init process can be found at init :

<android source>/system/core/init

- init.rc file can be found in source tree at <android source>/system/core/rootdir/init.rc
- readme.txt file can be found in source tree at <andorid source>/system/core/init/readme.txt

Step 5: Zygote and Dalvik

- In a Java, We know that separate Virtual Machine(VMs) instance will popup in memory for separate per app, In case of Android app should launch as quick as possible, If Android os launch different instance of Dalvik VM for every app then it consume lots of memory and time. so, to overcome this problem Android OS as system named "Zygote".
- Zygote enable shared code across Dalvik VM, lower memory footprint and minimal startup time.

- Zygote is a VM process that starts at system boot time as we know in previous step. Zygote preloads and initialize core library classes.
- Normally there core classes are read-only and part of Android SDK or Core frameworks. In Java VM each instance has it’s own copy of core library class files and heap objects.

Dalvik

- Dalvik is the process virtual machine (VM) in Google's Android operating system.
- It is the software that runs the applications on Android devices.
- Dalvik is thus an integral part of Android, which is typically used on mobile devices such as mobile phones and tablet computers as well as more recently on embedded devices such as smart TVs and media streamers.

- Programs are commonly written in Java and compiled to bytecode for the Java virtual machine, which is then translated to Dalvik bytecode and stored in .dex (Dalvik EXecutable) and .odex (Optimized Dalvik EXecutable) files, giving rise to the related terms odex and de-odex. The compact Dalvik Executable format is designed for systems that are constrained in terms of memory and processor speed.
- Dalvik is open-source software. It was originally written by Dan Bornstein, who named it after the fishing village of Dalvík, Iceland.

Step 6: System Service or Services

- After complete above steps, runtime request Zygote to launch system servers. System Servers are written in native and java both, System servers we can consider as process, The same system server is available as System Services in Android SDK. System server contain all system services.

Core Services:

1. Starting Power Manager
2. Creating Activity Manager
3. Starting Telephony Registry
4. Starting Package Manager
5. Set Activity Manager Service as System Process
6. Starting Context Manager
7. Starting System Context Providers
8. Starting Battery Service
9. Starting Alarm Manager
10. Starting Sensor Service
11. Starting Window Manager
12. Starting Bluetooth Service
13. Starting Mount Service

Other services

1. **Starting Status Bar Service**
2. **Starting Hardware Service**
3. **Starting NetStat Service**
4. **Starting Connectivity Service**
5. **Starting Notification Manager**
6. **Starting DeviceStorageMonitor Service**
7. **Starting Location Manager**
8. **Starting Search Service**
9. **Starting Clipboard Service**
10. **Starting Checkin Service**
11. **Starting Wallpaper Service**
12. **Starting Audio Service**
13. **Starting HeadsetObserver**
14. **Starting AdbSettingsObserver**

Step 7 : Boot Completed

- Once System Services up and running in memory, Android has completed booting process, At this time "ACTION_BOOT_COMPLETED" standard broadcast action will fire.