Board Bringup

What to Expect?

- Board specific Details
- Understanding the target board
- How to play with the target board?
- Peeking into Vendor supplied Utilities, if any

Startup Doubts

- Typical doubts you may have, be it your desktop or board
  - What happens when you switch on the power?
  - Where does the processor starts executing on “Power on Reset”?
  - What code is there? Do you have access to that code?
  - How does the Operating System boot up?
  - How does the login prompt come?
  - Any many more

Startup Sequence

- To Decode all these, let's understand the Startup Sequence
  - Processor / Controller Startup
  - Bootloader or the Software Startup
  - Operating System Startup
  - Application Startup
- Though they may vary from board to board, we can have a generic overview
- And then, we shall get into your board specifics

Processor / Controller Startup

- Controller Internal Code
  - Mostly exists in Controllers (Embedded Systems)
- System Startup / Setup Code
  - Mostly exists for Processors in Desktops
  - Processors jump to a pre-designated address, typically Zero, to run these pre-programmed code
  - Referred as BIOS in the Desktop parlance
  - Needs to be programmed once, on virgin boards
- In both the cases, the Code looks for the Stage1 Bootloader at the designated places
  - Embedded: EEPROM, Flash, Serial Download, ...
  - Desktops: Floppy, CDROM, Hard Disk, Network, ...

Software Startup

- Stage 1 Bootloader
  - Initial Program Loader (IPL) in Embedded World
  - Master Boot Record (MBR) in Desktops
  - Constrained to limited Space. Can't do much
  - So, loads
- Stage 2 Bootloader
  - Have enough space to do luxurious stuff
  - Provides configurability and management features
  - Loads the Operating System
  - Passes arguments to the Operating System
  - Jumps to start executing the Operating System
  - For Desktops: LILO, GRUB, SYSLINUX, ...
  - In Embedded Systems: u-boot and others
    - Also called Secondary Program Loader (SPL)

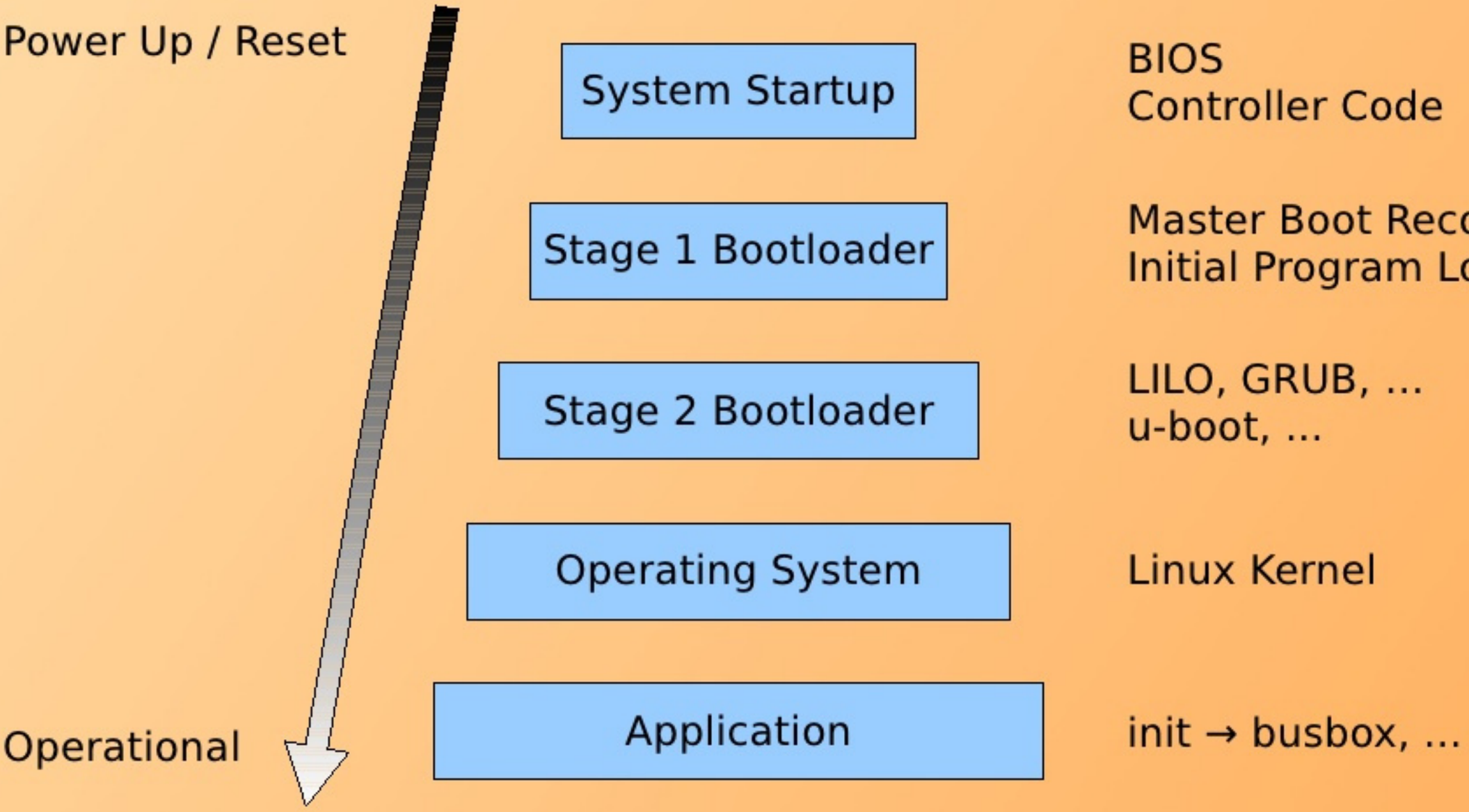
Operating System Startup

- Uncompresses the kernel, if compressed
- Configures itself based on the arguments from the Stage 2 Bootloader
- Setup the Kernel Space
- Jump execution to the first application
  - “init”
- Common to both Desktop & Embedded

Application Startup

- init
  - This could be the application binary itself. Or,
  - Could be a link to it
  - It typically starts the various daemons
  - And then does things as per the system's requirement
  - In Desktop, starts the login process to provide login prompts
  - In Embedded Systems, does the same at least during Development Cycle
  - Later, may do specific to customer requirement

Startup Sequence



Let's Startup the Board

Anatomy of Development Board

- Being a Embedded Developer, you should know your board in more detail
- Both
  - Hardware
    - Major components
    - Types of memories
  - Software
    - Manufactured State
    - Factory Restoration

Let's Browse the Board

Manufactured State

- State at which, it was when manufactured
- Also called the Virgin State
- Achieved by erasing all the memories
  - EEPROM
  - Flash – NOR, NAND, ...
- This, in principle erases
  - Bootloaders – Stage 1 & Stage 2
  - Kernel
  - File Systems

Factory Restoration

- So, this is to get the Board back to boot up
- In principle, populating back the memories erased with
  - Stage 1 & Stage 2 Bootloaders
  - Kernel
  - Root File System
  - Optionally, the Other File Systems
- Would need some special utilities
  - Provided by the board vendor (the factory guy)
  - Examples: RAM Monitor, Boot Monitor, ...
  - At times of nothing, these are good debug utilities, as well

What all have we done?

- Understood the target board
  - Switching it on
  - Accessing the stuff on it
  - Configuring it
- How to play with the target board?
  - Decoding the Hardware
  - Taking it to a virgin state
  - Restoring it to factory defaults

Any Queries?