What is a bootloader?

A bootloader is a small program (which is stored in an internal ROM(non volatile), flash memory) that runs when a computer or device is powered on. It is responsible for initializing hardware and loading the operating system or firmware into memory(RAM). It acts like a bridge hardware and the os.

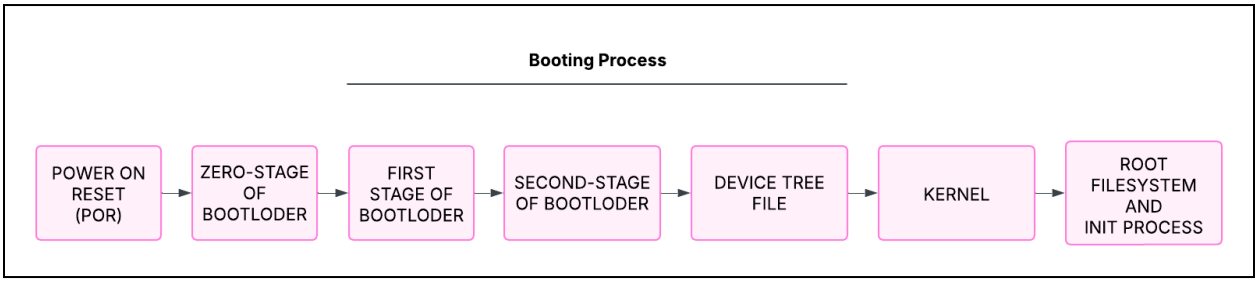
Its main work:

- Initialize Hardware(CPU,RAM,Disks).
- Load the OS kernel into memory.
- Hand over the control to the os.

For every os different bootloader are used like for windows use windows boot manager and linux used grub.

Booting Process :

We can see the booting process from scratch.



General Boot FlowDiagram

1.Power On Reset (POR)

When the power button press or the voltage apply to the board the following steps occour before the CPU execute the first instruction:

This power stabilisation and clock generation is important because when this gets set then the cpu can start executing the instruction.

1.1 Power Stabilization:

When the power supply to the board the that time we need to stabilize the flow of current for that we use following steps:

Voltage Rails: The power management Ic (PMIC) or voltage regulators activate and stabilize core voltages for bord on board devices.

Ex: On an ARM Cortex-A SOC the PMIC ensures 1.2v for the cpu and 1.8v for DDR memory power supply.

Power Sequencing: Voltages are applied to the board in specific order to avoid damage.

Ex: cores first, I/O devices second).

1.2 Clock Generation:

The clock signal provided by the crystal oscillator to CPU and the PLLs(Phase-Locked Loops): Lock the stable frequencies to drive CPU, memory and peripherals.

1.3 Reset Signal assertion

Once the power and clock frequency get stable the POR releases the reset line allowing the CPU to start executing.

The control transfer to the Zero- Stage.

2. Zero-stage of bootloader

In stage 0 the initial code executed immediately after the power-on.It is hardcoded into the chip's ROM by manufacturer and cannot be modified.its job is to initialize the bare minimum hardware, detect a valid boot source and load the next stage(eg. U-Boot,SPL).

Key components work in this stage:

2.1 Boot Rom: A small unmodified firmware also called EEPROM which contains a code to initialize critical hardware(clocks, memory controllers).

It checks the pre-configured boot sources(eg. eMMc,SD card,USB) it selects the boot source selection from SD card eMMC.

Example: BBB checks the gpio pins to decide whether to boot from SD card or USB.

2.2 Initial program loader or Secondary Program Loader (IPL / SPL)

Loads the next-stage bootloader U-boot from storage into on chip SRAM.

STAGE 0 BOOTFLOW:

1. After the power on the cpu starts executing code from the reset vector(a fixed address in boot ROM).
And minimal hardware initialises like clock, memory.
2.Scan the devices for boot source in order (eg. SD card, eMMc, Uart).

Know the Rom bootloader loads the first stage bootloader from the boot device into the internal soc memory and passes control to it.

3. First-Stage of bootloader.

In this stage SPL loaded by rom initialise hardware (eg. DRAM,Peripherals).loads the next stage(second stage).
Runs from on chip SRAM.

Responsibilities:

The Hardware initialization like DRAM configures memory.
Control passess to the second stage bootloder.

4. Second-Stage of bootloader:

The second stage bootloader reads its configuration settings (either statically embedded or external file) and finds and loads the linux kernel and the device tree binary into ram.
Its loads the kernel(zImage) and device tree(dtb) from an SD card and passes control to kernel.
Provides a user interface for configuration (eg. U-boot command line).
Support booting from multiple sources (network, USB, FLASH).

Workflow:

- Search for the kernel image into (eg. zImage, uImage) on storage.
- Copy the kernel binary from storage to DRAM.
- Lods the Device tree Blob(.dtb)
- Pass the arguments which is command line parameters to kernel (eg. root filesystem location, console setting).

5. Device tree blob:

The DTB is a binary file that describes hardware components (e.g., CPU, UART, GPIO) to the kernel. It is used in embedded systems where hardware is not self-descriptive.

6. Kernel:

After successfully loaded the device tree the control passed to the kernel.

1. kernel Decompression and Initialization:

- The compressed kernel image is decompressed into Ram

2. Kernel initializes:

- Memory Management Unit (MMU, paging, etc).
- CPU and scheduling system.
- Device drivers (UART, I2C, SPI, etc) using DTB data.

7. Step3:Kernel Starts the init Process:

The init process is the first user-space process started by the linux kernel after its hardware and driver initialization.

Key Responsibility:

1. System Initialization

2. Managing system services and Daemons:

- Start background services like cron, sshd and networking.

3.Runtime process.

4. Managing system logging.

Ensures logging services like syslog.

5. Handling User Sessions

- Starts the login prompt on the terminal or serial console.
- Manages user login process.
- Launches user shells.

BOOT MODES:

1.SD card

2.EMMc

3.Network(PXE/TFTP).

The boot modes are also categorised in primary and secondary.

1. Primary:

- Flash Boot: Boot from NOR/NAND Flash or eMMC.
- SD/MMC Boot: Boot from an SD card or eMMC.
- Network Boot: Boot over Ethernet using PXE/TFTP.

2. Secondary boot Modes:

- Recovery Mode: Boot into a minimal environment for firmware updates.
- Secure Boot: Verify firmware signatures using cryptographic keys.
