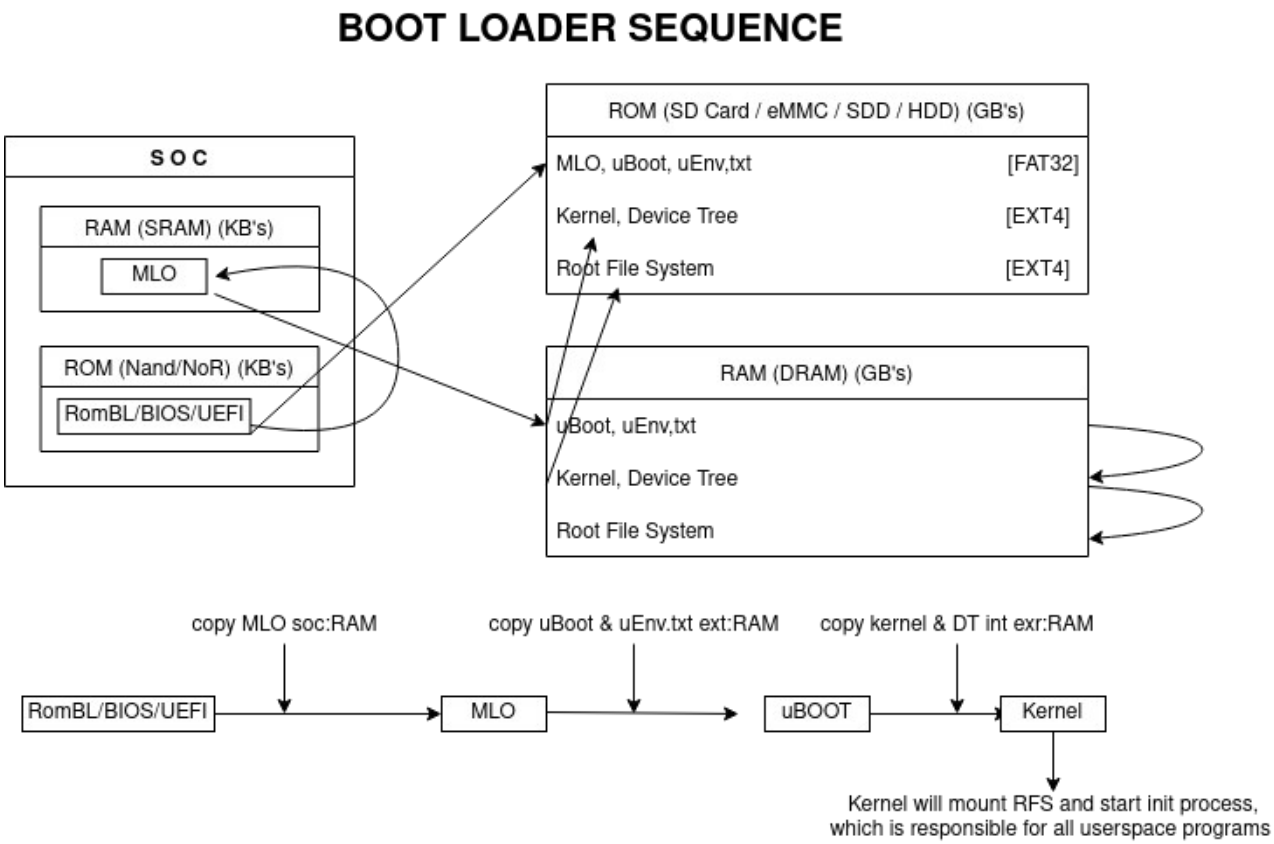

[GENERIC LINUX BOOT SEQUENCE]

components : [HW] soc:ROM, soc:RAM, ext:ROM, ext:RAM
components : [SW] ROM BootLoader (boot firmware = BIOS = UEFI), MLO, uBoot, uEnv.txt, Kernel, Device Tree, Root File System

soc:ROM & soc:RAM size usally will be in KB's. (soc:RAM will be SRAM) (soc:ROM will be NOR/NAND Flash)
ext:ROM & soc:RAM size usally will be in GB's. (ext:RAM will be DRAM) (ext:ROM will be SSD or MMC or eMMC or SD Card or HDD)
SSD, MMC, eMMC, SD Card & Pendrive all comes under flash drives (no moving compoents) and all uses NAND Flash inside, there speeds depends on there architectures.
HDD comes under hard drive (contains moving comonents) uses magnetic disk inside (not NAND Flash).

ROM BootLoader (boot firmware = BIOS = UEFI) size will be in KB's, this will be flashed by chip vendor, usally we execute this code as XIP execution in place. (will be executed from soc:ROM itself XIP)
MLO size will be in KB's, This file knews how to configure and set ext:RAM to functioning state. (will be copied and executed from soc:RAM)
uBoot size will be in MB's, knows how to load and config kernel and device tree, (will be copied and executed from ext:RAM)

1. Upon Reset some registers i.e CS, eIP, will be reset to some fixed address (0xffffff/ 0x0000) & CPU will start executing instructions strting from that address, this is the place where ROMBootoader (BIOS/UEFI/Firmware) code resides (this will in size of KB's)

2. ROMBootLoader is an application of its owm, it will execute in XIP mode (eXecution In Place) (Real Addressing mode) , means this will get executed fom soc:ROM itself using soc:RAM help.

3. RomBL will search for FAT32 partion in SD card and looks for MLO file and copies it into soc:RAM and pass control to MLO

4. Now MLO has the capability to configure and set ext:RAM into fuctioning.

5. Then MLO looks for uBoot and uEnv.txt and copies it into ext:RAM and pass control to uBoot.

6. uBOOT will look for Kernel partion (primary partion) and copies it to ext:RAM, as well as device tree based on uEnv.txt file, then it passe BOOTARGS to kernel and pass control to kernel.

7. Now Kernel will mount Root File system using BOOTARGS and later start init process (PID=0) which is responsible to start all userspace programs based on its config files.

Note:
MMC/SD Read Sector has two mode **Raw Mode** and **FAT mode**

In Raw mode all the boot images are plced in fixed address.

In FAT mode devicesxmay hold fat file system , so rom code will look for MLO file.

Note:
There two different kind of boot procedures BIOS and UEFI mode

BIOS is a legacy mode where ROM Code -> MLO -> Uboot -> kernel this seq exist
whole ROMBL will be present in ROM .

Whereas in UEFI mode is an advanced version in which data related to initilazion and startup will be stoed in .efi file instead of hard coding in rom (so update is possible)

Can a PC has both Bios and UEFI ?
UEFI replaces BIOS on PC , so there is noway to switch from BIOS to UEFI on an existing PC. You need additional HW that support and include UEFI, as most of the modern computers do.

Basically UEFI is a tiny OS that runs on top of the PC firmware and it can do alot more that bios, it may be stored in flah memory on the mother board or it can be loadded from haed drive or network share at boot time.