x64 Kernel Boot Process

Stage 1: BIOS and MBR

When the system powers on, the CPU starts executing in **16-bit real mode**.
The BIOS or UEFI Compatibility Support Module (CSM) loads the first **512 bytes** from the boot device (the **Master Boot Record**, or MBR) into memory at **physical address 0x7C00** and jumps there.

The MBR contains code that locates and loads a more capable **second-stage bootloader**.
Real mode provides access to BIOS interrupts for disk I/O and basic hardware control.

Sources:

- OSDev Wiki: MBR (x86)
- OSDev Wiki: Boot Sequence
- Intel SDM Vol. 3A, Chapter 9

Stage 2: Relocation and Second Stage Loader

The 512-byte limit of the MBR requires a second stage.
The MBR code **relocates itself** to a lower address (e.g., `0x0600`) to free up `0x7C00`,
then reads additional sectors from disk using **BIOS INT 13h**.

The second-stage bootloader can be larger and more feature-rich — it initializes hardware, memory maps, and prepares for protected mode.

Sources:

- OSDev Wiki: System Initialization (x86)
- OSDev Wiki: Bootloader FAQ

Stage 3: Enabling the A20 Line and Entering Protected Mode

Before switching to protected mode, the **A20 line** must be enabled to access memory beyond 1MB.
Then, a **Global Descriptor Table (GDT)** is constructed to define flat 4GB memory segments for code and data.
After loading GDTR, the bootloader sets `CR0.PE = 1` to enable protected mode and performs a **far jump** to 32-bit code.

Sources:

- OSDev Wiki: A20 Line
- OSDev Wiki: GDT Tutorial
- OSDev Wiki: Protected Mode

Stage 4: Setting Up Paging and Entering Long Mode (x64)

Long mode is the **64-bit mode** of the x86 architecture.
It requires **paging** and **Physical Address Extension (PAE)** to be enabled.

Steps to enter long mode:

1. Enable PAE by setting `CR4.PAE = 1`.
2. Load page tables (PML4 → PDP → PD → PT).
3. Set the Long Mode Enable (LME) bit in the `EFER` MSR.
4. Enable paging via `CR0.PG = 1`.

Finally, a **far jump** to a 64-bit code segment activates long mode.

Sources:

- OSDev Wiki: Long Mode
- Wikipedia: Long Mode
- AMD64 Architecture Programmer’s Manual Vol. 2

Stage 5: Loading and Jumping to the Kernel

Once in long mode, the bootloader loads the **kernel image** (ELF or flat binary) into memory,
sets up stack and page tables if needed, and jumps to the kernel’s **entry point**.

The kernel then initializes its environment: interrupts, higher-half mapping, drivers, etc.

Sources:

- OSDev Wiki: ELF
- OSDev Wiki: Loading a Higher Half Kernel
- OSDev Wiki: Bare Bones (x86-64)

Further Reading

- OSDev Wiki Main Page
- UEFI Specification
- AMD Developer Guides and Manuals
- Intel® 64 and IA-32 Architectures Software Developer’s Manual