The Boot Process

When the computer is turned on, it begins executing firmware code that is stored in motherboard `ROM`. On `x86` there are two firmware standards: - The old Basic Input/Output System (`BIOS`). Outdated but simple and well-supported on any x86 machine since the 1980s. - The newer unified Extensible Firmware Interface (`UEFI`). More modern and with a lot more features but more complex to set up.

When it's loaded, the `BIOS`: 1. Performs a power-on self-test 2. Detects available `RAM` 3. Pre-initializes the `CPU` and hardware 4. Looks for a bootable disk. If it finds one, the control is transferred to its *bootloader*, a 512-byte portion of executable code stored at the disk's beginning.

The bootloader has three main jobs: 1. Determine the location of the kernel image on the disk. 2. Load it into memory and switch the `CPU` from the **16-bit real mode** first to the **32-bit protected mode** and then to the **64-bit long mode**, where 64-bit registers and the complete main memory are available. 3. Query information (such as a memory map) from the `BIOS` and pass it to the `OS` kernel.