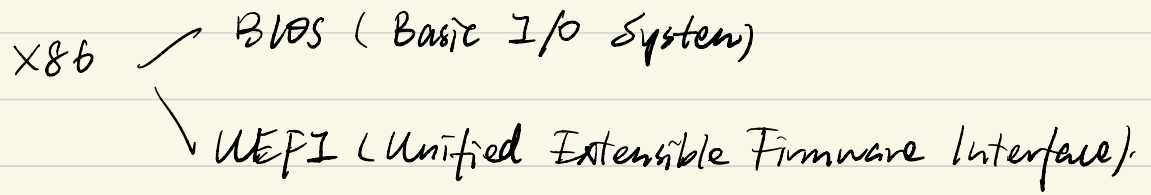
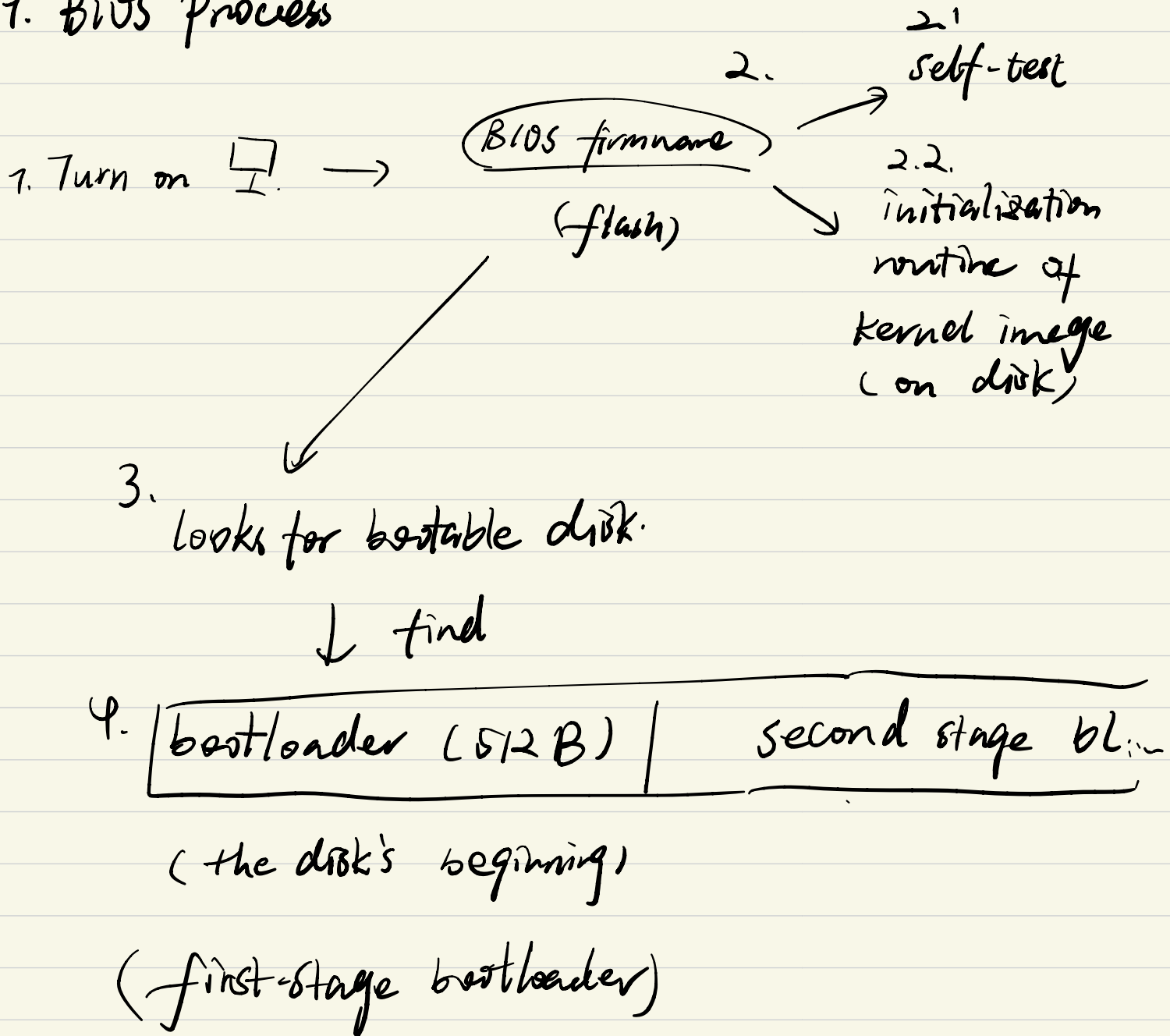


< Ch2. Minimal Rust Kernel >



- 2.1. BIOS Process



boot loader
(assembly magic number).

4.1. determine the loc of kernel img (disk)

5.1. load into memory

4.2. switch CPU: 16-bit real mode

↓
32-bit protected mode

5.2. ↓
64-bit long mode.

(64-bit regs & complete main mem)
are available

4.3. Query certain info from BIOS,

(e.g. memory map)

and pass it to the OS kernel.

(补充阅读: { Multiboot - e.g. GNU GRUB }
UEFI)

2.2 Minimal Kernel.

Goal: create a disk img ("Hello World").

↑
extend freestanding Rust binary (ch1).

'Cargo bootimage.'

- ① compiles our kernel to an ELF file
- ② compiles the bootloader dependency as a standalone executable
- ③ links the bytes of the kernel ELF file to the bootloader.

(booted)
bootloader

- ① reads & parses the appended ELF
- ② maps the program fragments to virt addresses in the page table
- ③ zeroes the '.bss' section.
- ④ sets up stack
- ⑤ reads the entry point address (-start) and jumps to it.