

Week 12.b  
CS 5600  
03/29 2023

- 0. Last time ↙
  - 1. Disks
  - 2. SSDs
  - 3. Intro to file systems
- 

CPU-I/O interactions, four approaches:

- \* port-mapped I/O → Keyboard, set cursor
  - \* memory-mapped I/O → Console putc
  - \* interrupts
  - \* via memory
- DMA
- MMIO
- PMIO

- { polling, interrupt } X { Programmed IO, DMA }
- drivers,

Q: NVIDIA GPU driver → AMD GPU?

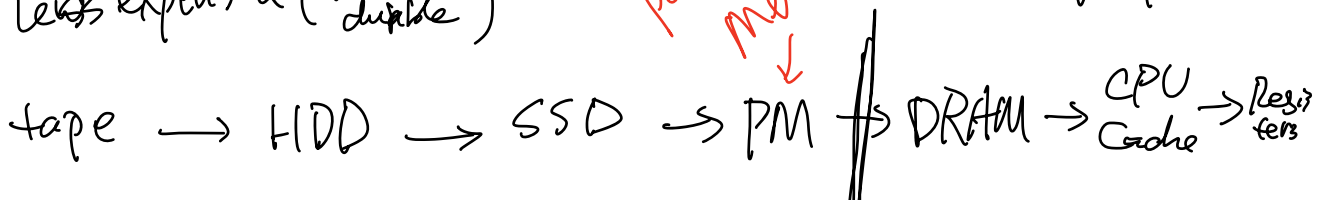
Q: NVIDIA GPU windows driver → Linux?

• Disk

less expensive (more durable)

Persistent Memory

Performance



Persistent ← → Volatile

### Hard disk drive organization

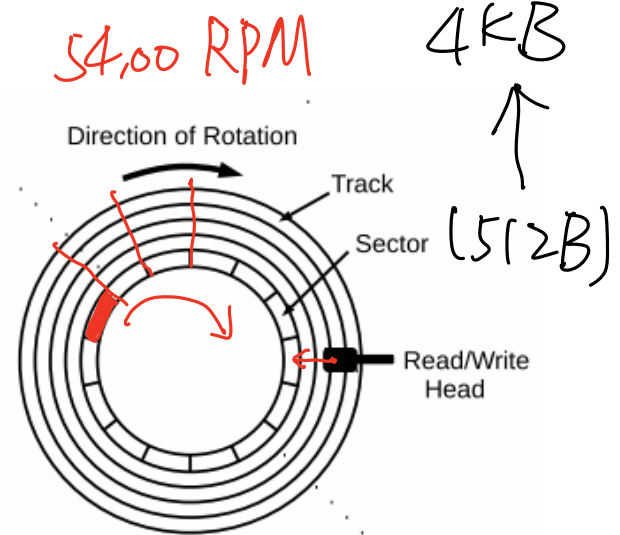
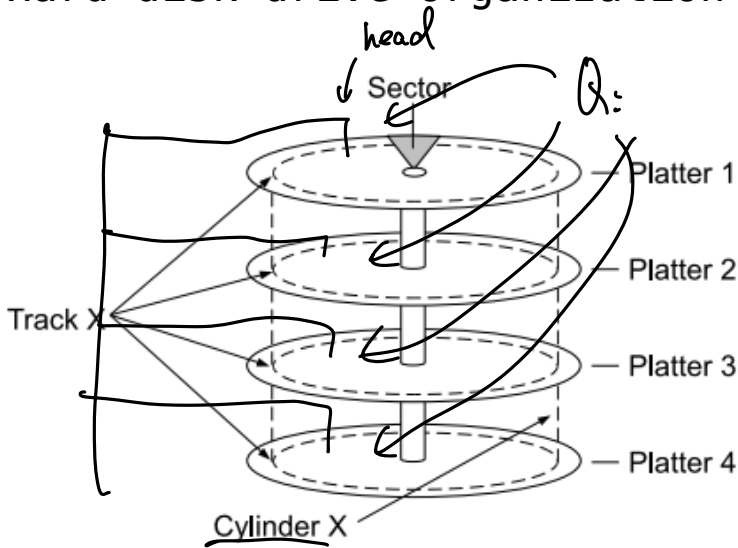


Fig 5.11, 5.12 from <http://www.ccs.neu.edu/~pjd/x600-book-v0903.pdf>

Q: Read vs. Write?

Q: Sequential Read/write vs. Random R/W?

solid state drive

### A Flash-based SSD (logical diagram)

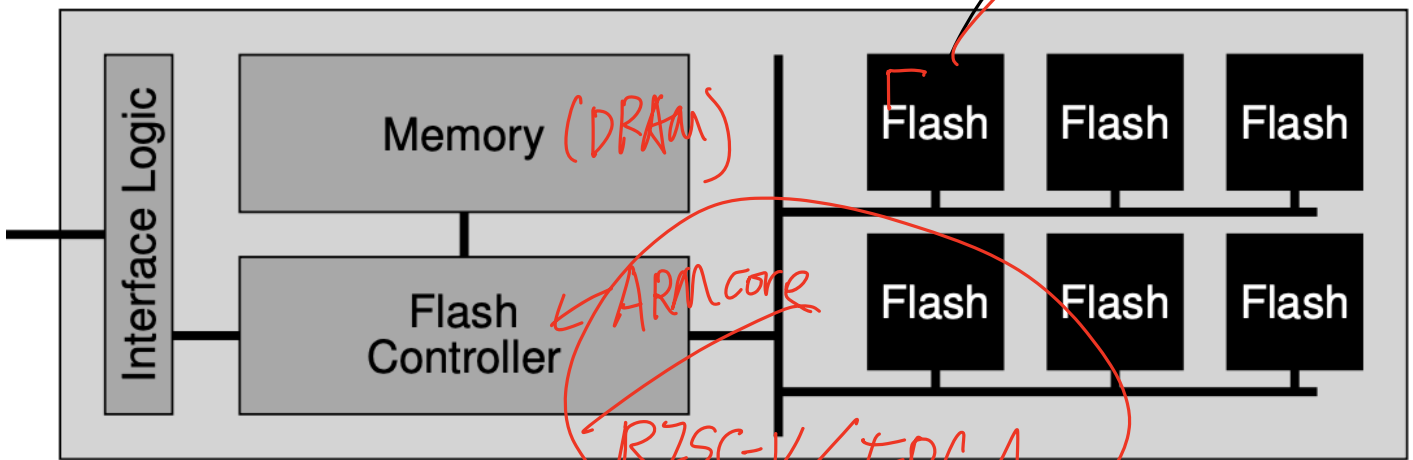
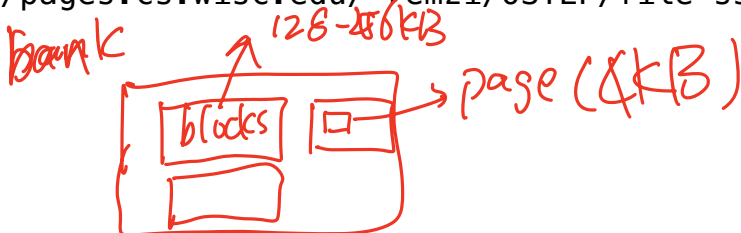


Fig 44.3 from <https://pages.cs.wisc.edu/~remzi/OSTEP/file-ssd.pdf>

Smart SSD



Common #s and performance

- \* capacity: Several TBs, ( $\rightarrow$  20TBs)
- \* platters: 8
- \* number of cylinders:  $\sim 100,000$
- \* sectors per track:  $\sim 1000$
- \* RPM: 5400  $\rightarrow$  15000
- \* transfer rate: 50  $\sim$  200 MB/s

question]

\* mean time between failures:

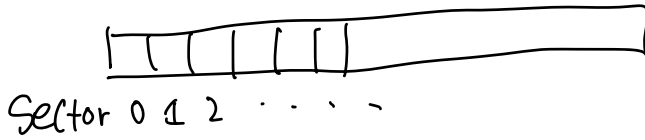
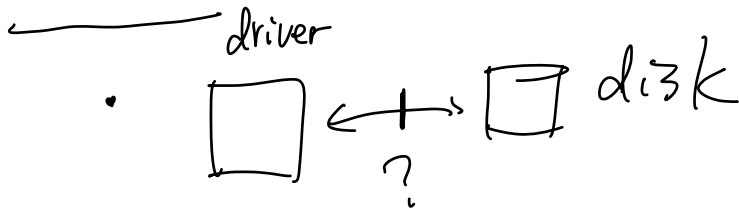
$\sim 1M$  hrs

$\hookrightarrow$  60 yrs.

25,233 hrs.

$\uparrow$   $<$  3 yrs.

1s, 0.01s, 1min, 2weeks.  
1month, 5 yrs,  $\infty$   
7 yrs, 11 yrs, 20 yrs, 50 yrs,



```

1 CS5600, Handout Week12.a
2
3 // Code snippets borrowed from WeensyOS boot loader
4 // They illustrate how kernel "talks" to a disk through programmed I/O:
5 // the bootloader reads in the kernel from the disk.
6 //
7 // See the functions boot_waitdisk() and boot_readsect(). Compare to
8 // Figures 36.5 and 36.6 in OSTEP.
9
10
11 // WeensyOS boot loader loads the kernel at address 0x40000 from
12 // the first IDE hard disk.
13 //
14 // A BOOT LOADER is a tiny program that loads an operating system into
15 // memory. It has to be tiny because it can contain no more than 510 bytes
16 // of instructions: it is stored in the disk's first 512-byte sector.
17
18 #define SECTORSIZE 512 ← ptr // 100
19
20 // boot_readsect(dst, src_sect)
21 // Read disk sector number `src_sect` into address `dst`.
22 static void boot_readsect(uintptr_t dst, uint32_t src_sect) {
23     // programmed I/O for "read sector"
24     boot_waitdisk();
25     outb(0x1F2, 1); // send `count = 1` as an ATA argument
26     outb(0x1F3, src_sect); // send `src_sect`, the sector number
27     outb(0x1F4, src_sect >> 8);
28     outb(0x1F5, src_sect >> 16);
29     outb(0x1F6, (src_sect >> 24) | 0xE0);
30     outb(0x1F7, 0x20); // send the command: 0x20 = read sectors
31
32     // then move the data into memory
33     boot_waitdisk();
34     insl(0x1F0, (void*) dst, SECTORSIZE/4); // read 128 words from the disk
35 }
36
37 // boot_waitdisk
38 // Wait for the disk to be ready.
39 static void boot_waitdisk(void) {
40     // Wait until the ATA status register says ready (0x40 is on)
41     // & not busy (0x80 is off)
42     while ((inb(0x1F7) & 0xC0) != 0x40) {
43         /* do nothing */
44     }
45 }
46
47

```

Handwritten annotations:

- Line 18:  $512 \leftarrow \text{ptr}$  and  $// 100$
- Line 22:  $\rightarrow$  pointing to the function signature
- Line 25:  $\uparrow$  pointing to `boot_waitdisk()`
- Line 26:  $\uparrow$  pointing to `outb(0x1F2, 1)`
- Line 27:  $\uparrow$  pointing to `outb(0x1F3, src_sect)`
- Line 28:  $\uparrow$  pointing to `outb(0x1F4, src_sect >> 8)`
- Line 29:  $\uparrow$  pointing to `outb(0x1F5, src_sect >> 16)`
- Line 30:  $\uparrow$  pointing to `outb(0x1F6, (src_sect >> 24) | 0xE0)`
- Line 31:  $\uparrow$  pointing to `outb(0x1F7, 0x20)`
- Line 34:  $\uparrow$  pointing to `boot_waitdisk()`
- Line 35:  $\uparrow$  pointing to `insl(0x1F0, (void*) dst, SECTORSIZE/4)`
- Line 36:  $\uparrow$  pointing to `insl(0x1F0, (void*) dst, SECTORSIZE/4)`
- Line 39:  $\uparrow$  pointing to `boot_waitdisk`
- Line 40:  $\uparrow$  pointing to `Wait for the disk to be ready.`
- Line 41:  $\uparrow$  pointing to `static void boot_waitdisk(void) {`
- Line 42:  $\uparrow$  pointing to `Wait until the ATA status register says ready (0x40 is on)`
- Line 43:  $\uparrow$  pointing to `& not busy (0x80 is off)`
- Line 44:  $\uparrow$  pointing to `while ((inb(0x1F7) & 0xC0) != 0x40) {`
- Line 45:  $\uparrow$  pointing to `/* do nothing */`
- Line 46:  $\uparrow$  pointing to `}`
- Line 47:  $\uparrow$  pointing to `}`

Other annotations:

- Line 26:  $\uparrow$  pointing to `1` with note "sect = 100"
- Line 27:  $\uparrow$  pointing to `src_sect` with note "PM20"
- Line 28:  $\uparrow$  pointing to `src_sect >> 8` with note "28bit"
- Line 35:  $\uparrow$  pointing to `SECTORSIZE/4` with note  $512/4 = 128$
- Line 44:  $\uparrow$  pointing to `polling`

BURNMED: wear-out

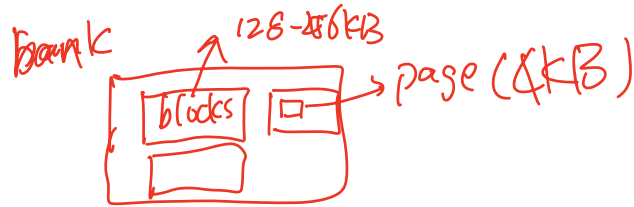
SSD

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

1 page → #erase → 10,000

Ops:

- read : 1 Page
- erase : 1 block, reset ALL bits to 1
- Program: 1 page, set some bits to 0  
(Cannot program a page twice w/o erasing)



Q: how to update a page?

$a \neq 1$

