Roadmap

- CPU management
  - Process, thread, synchronization, scheduling
- Memory management
  - Virtual memory, demand paging
- **Disk management**
  - **I/O**
  - **Filesystem**
- Other topics

KU THE UNIVERSITY OF KANSAS

I/O

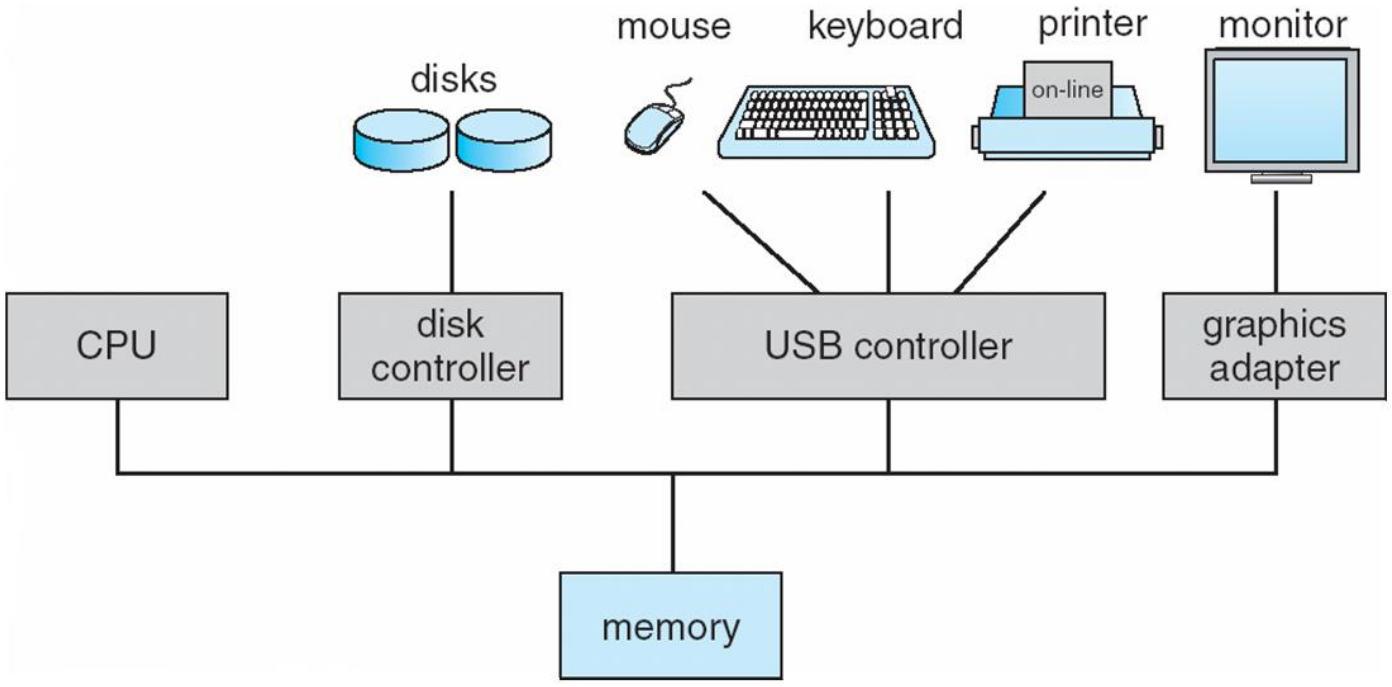
Heechul Yun

Disclaimer: some slides are adopted from book authors' slides with permission

Concepts to Learn

- I/O subsystems
- Blocking, non-blocking, asynchronous I/O
- Memory-mapped I/O
- Programmed I/O vs. DMA
- Disk

Input/output (I/O) Subsystems



I/O Subsystems: the Goal

- Provide easy to use **standardized** interfaces
  - This code works for many different devices

```
int fd = open("/dev/somedev", O_RDWR);
char buf[80];
for (int i = 0; i < 10; i++) {
        sprintf(buf, "i: %d\n", i);
        write(fd, buf, 80);
}
close(fd);
```

  - Hide the details of each device to users

Standard Device Types

- Block devices
  - E.g., disk, cd-rom, USB stick
  - High speed, block (sector) level accesses
- Character devices
  - E.g., keyboard, mouse, joystick
  - Low speed, character level accesses
- Network devices
  - E.g., ethernet, wifi, bluetooth
  - Socket interface

Types of I/O Operations

page 602

- Blocking I/O
  - Wait (i.e., the calling process is put to sleep) until the data is ready
- Non-blocking I/O
  - Immediately return to the caller no matter what.
  - I/O may not be completed
- Asynchronous I/O
  - Notify later when the I/O is completed (via callback or interrupts)

Blocking I/O

```
int main(int argc, char *argv[])
{
      int src_fd, dst_fd; char buf[4100];    int nread, nwrite;   char *ptr;

      src_fd = open(argv[1], O_RDONLY);
      dst_fd = open(argv[2], O_WRONLY);
      nread = read(src_fd, buf, sizeof(buf));   ptr = buf;
      while (nread > 0) {
            errno = 0;
            nwrite = write(dst_fd, ptr, nread);
            fprintf(stderr, "nwrite = %d, errno = %d (%s)\n", nwrite, errno, strerror(errno));
            if (nwrite > 0 ) {
                  ptr   += nwrite; nread -= nwrite;
            }
      }
}
```

```
$ sudo ./copyfile /dev/zero /dev/ttyS0
nwrite = 4100, errno = 0 (Success)
```

Non-Blocking I/O

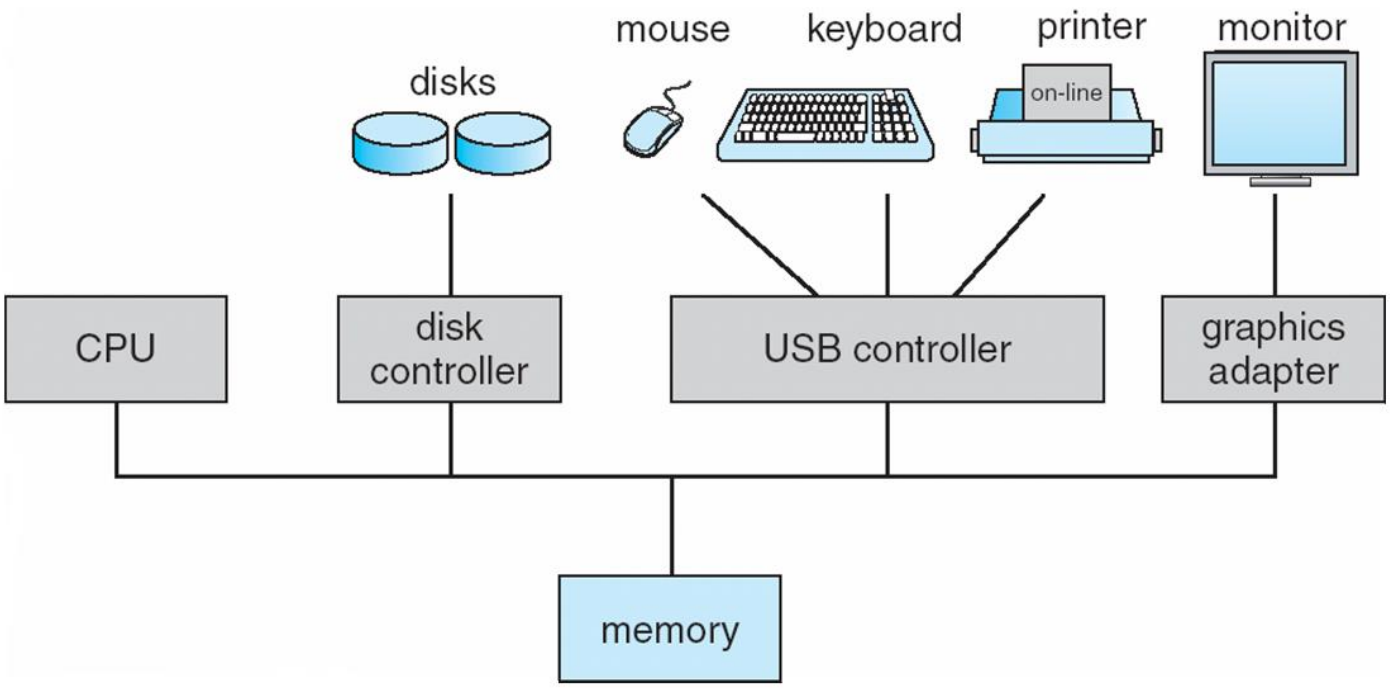
```
int main(int argc, char *argv[])
{
        int src_fd, dst_fd; char buf[4100];    int nread, nwrite;   char *ptr;

        src_fd = open(argv[1], O_RDONLY);
        dst_fd = open(argv[2], O_WRONLY|O_NONBLOCK);
        nread = read(src_fd, buf, sizeof(buf));   ptr = buf;
        while (nread > 0) {
                errno = 0;
                nwrite = write(dst_fd, ptr, nread);
                fprintf(stderr, "nwrite = %d, errno = %d (%s)\n", nwrite, errno, strerror(errno));
                if (nwrite > 0 ) {
                        ptr   += nwrite; nread -= nwrite;
                }
        }
}
```

```
$ sudo ./copyfile /dev/zero /dev/ttyS0
nwrite = 4095, errno = 0 (Success)
nwrite = -1, errno = 11 (Resource temporarily unavailable)
nwrite = -1, errno = 11 (Resource temporarily unavailable)
nwrite = 5, errno = 0 (Success)
```

How Does CPU Talk to Devices?

- CPU talks to device controllers
  - Via I/O instructions or **memory mapped I/O**



Memory Mapped I/O

Base Address	Limit Address	Size	Description
0x0000_0000	0x0001_0000	64 KB	iROM
0x0200_0000	0x0201_0000	64 KB	iROM (mirror of 0x0 to 0x10000)
0x0202_0000	0x0206_0000	256 KB	iRAM
0x0300_0000	0x0302_0000	128 KB	Data memory or general purpose of Samsung Reconfigurable Processor SRP.
0x0302_0000	0x0303_0000	64 KB	I-cache or general purpose of SRP.
0x0303_0000	0x0303_9000	36 KB	Configuration memory (write only) of SRP
0x0381_0000	0x0383_0000	–	AudioSS's SFR region
0x0400_0000	0x0500_0000	16 MB	Bank0 of Static Read Only Memory Controller (SMC) (16-bit only)
0x0500_0000	0x0600_0000	16 MB	Bank1 of SMC
0x0600_0000	0x0700_0000	16 MB	Bank2 of SMC
0x0700_0000	0x0800_0000	16 MB	Bank3 of SMC
0x0800_0000	0x0C00_0000	64 MB	Reserved
0x0C00_0000	0x0CD0_0000	–	Reserved
0x0CE0_0000	0x0D00_0000	–	SFR region of Nand Flash Controller (NFCON)
0x1000_0000	0x1400_0000	–	SFR region
0x4000_0000	0xA000_0000	1.5 GB	Memory of Dynamic Memory Controller (DMC)-0
0xA000_0000	0x0000_0000	1.5 GB	Memory of DMC-1

USB, SD/MMC, Timer, …

DRAM

Samsung Exynos 4412 (ARM) Processor **Physical** Address Map

Memory Mapped I/O

- Parts of physical memory space are mapped to hardware controllers
  - Mapped to control registers and buffers
- Reading/writing from/to the memory mapped regions in device specific ways
  - Device drivers' job

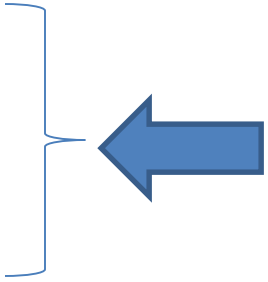
Example

```
#define CTRL_BASE_ADDR 0xCE000000

int *io_base = (int *)ioremap_nocache(CRTL_BASE_ADDR, 4096);

// initialize the device (by writing some values to h/w regs)
*io_base = 0x1;
*(io_base + 1) = 0x2;
*(io_base + 2) = 0x3;
…
// wait until the device is ready (bit31 = 0)
while (*io_base & 0x80000000);

// send data to the device
for (i = 0; i < sizeof(buffer); i++) {
    *(io_base + 0x10) = buffer[i];
     while (*io_base & 0x80000000);
}
```



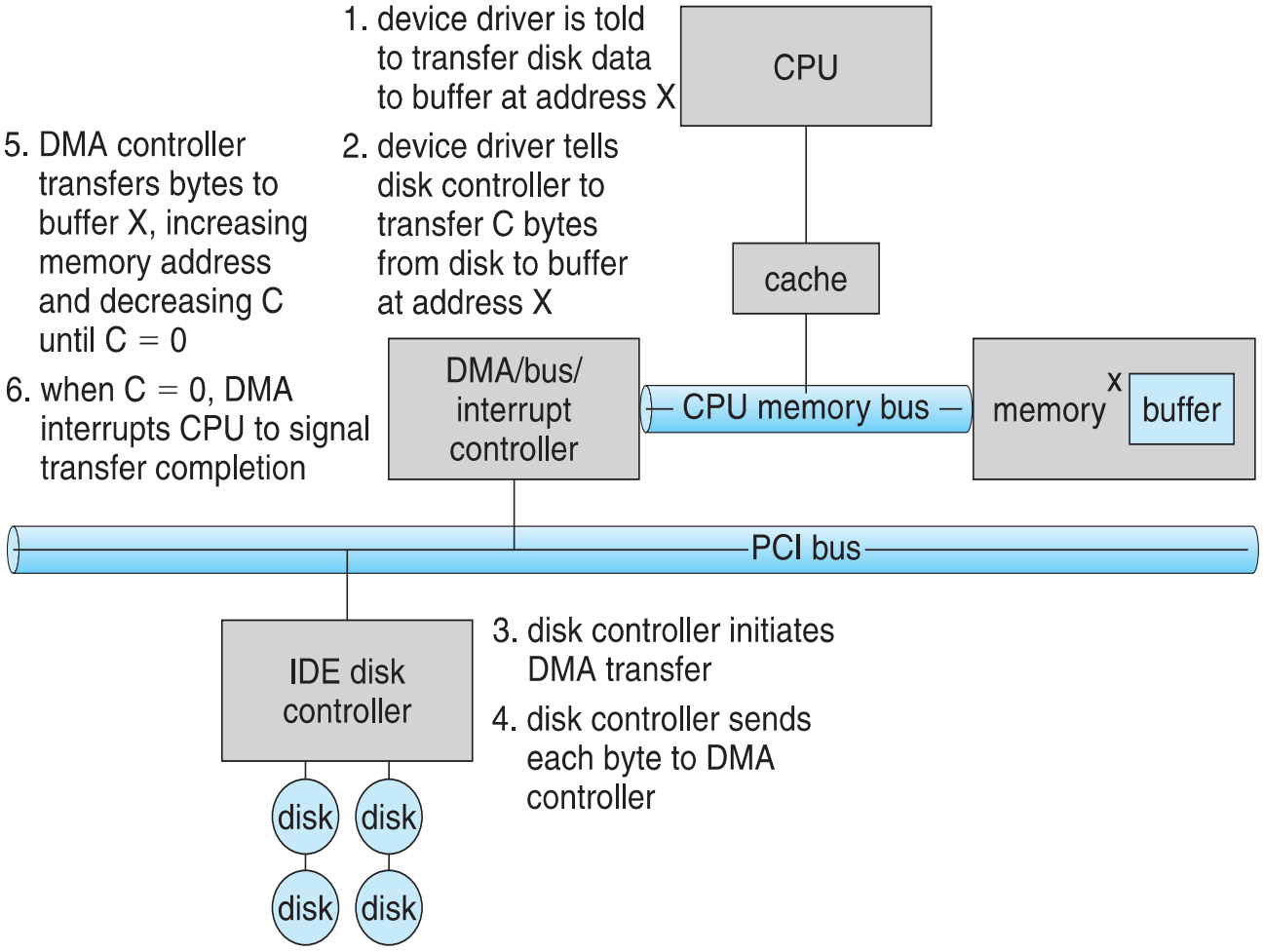
Programmed I/O (PIO)
Polling

Data Transfer Methods

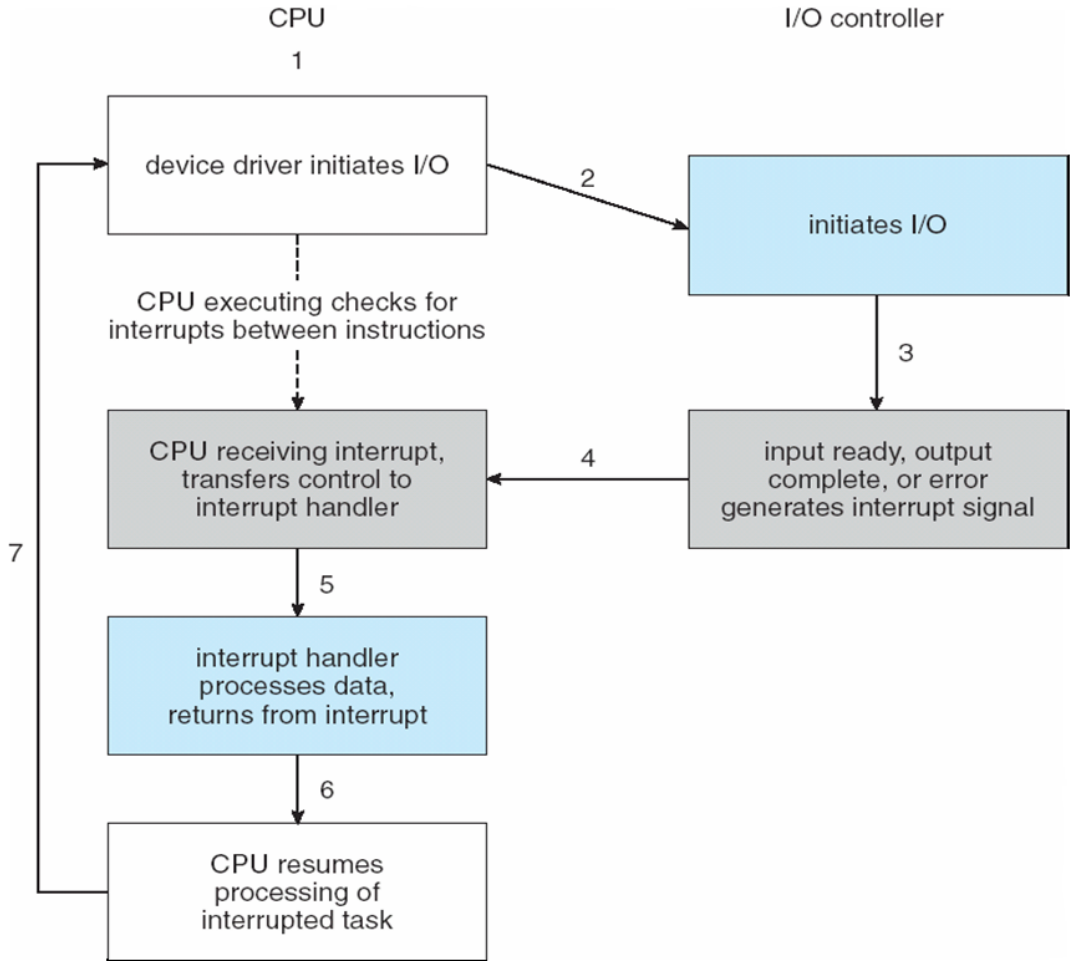
- Programmed I/O (PIO)
  - Via CPU's load/store instructions
  - Simple h/w, but high CPU load
- Direct Memory Access (DMA)
  - Controllers directly read/write from/to DRAM
  - Interrupts the CPU on the completion of I/O ops.
  - Complex h/w, but low CPU overhead

Direct Memory Access

考点



Interrupt Driven I/O Cycle



DMA Example

```
int dad_transfer(struct dad_dev *dev, int write, void *buffer,
                 size_t count)
{
    dma_addr_t bus_addr;

    /* Map the buffer for DMA */
    dev->dma_dir = (write ? DMA_TO_DEVICE : DMA_FROM_DEVICE);
    dev->dma_size = count;
    bus_addr = dma_map_single(&dev->pci_dev->dev, buffer, count,
                              dev->dma_dir);
    dev->dma_addr = bus_addr;

    /* Set up the device */

    writeb(dev->registers.command, DAD_CMD_DISABLEDMA);
    writeb(dev->registers.command, write ? DAD_CMD_WR : DAD_CMD_RD);
    writel(dev->registers.addr, cpu_to_le32(bus_addr));
    writel(dev->registers.len, cpu_to_le32(count));

    /* Start the operation */
    writeb(dev->registers.command, DAD_CMD_ENABLEDMA);
    return 0;
}
```

Virtual Address

Physical address

From Jonathan Corbet, *Linux Device Drivers*, p453

Disk

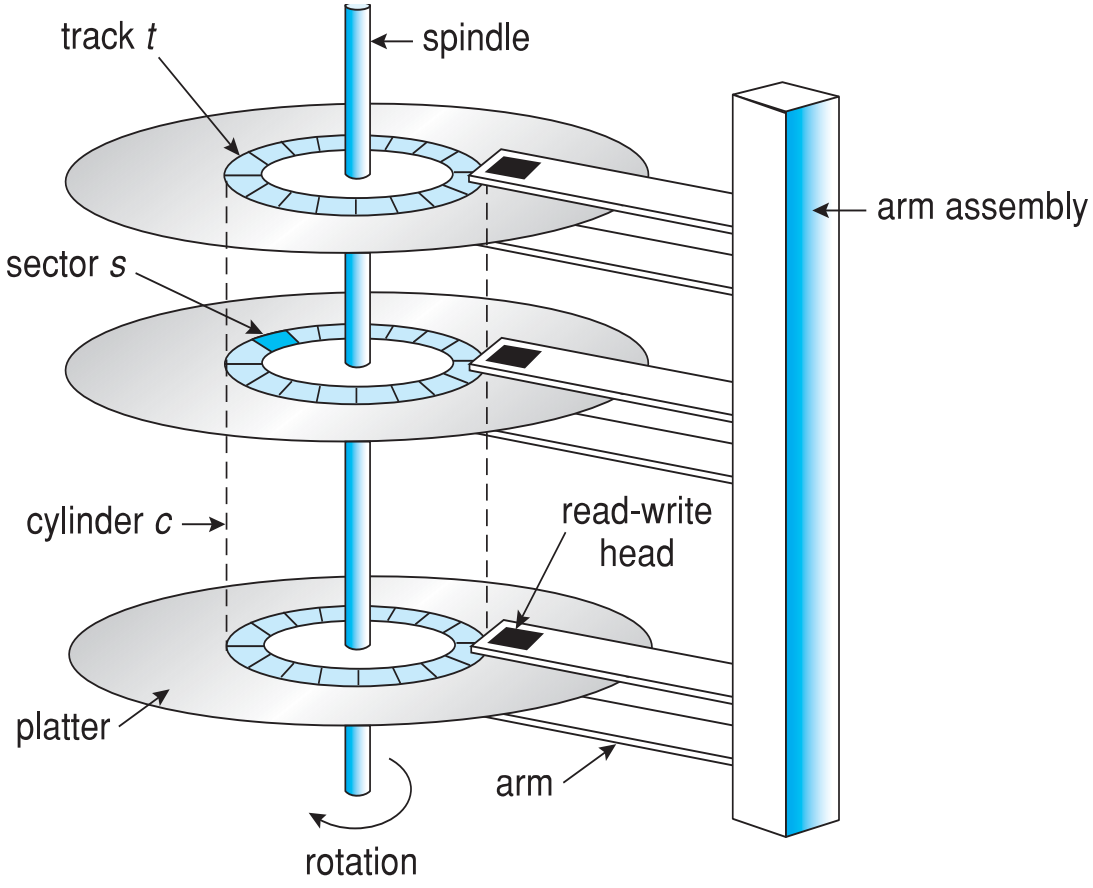
- Magnetic disks (HDD)
  - Still used as the main storage device on many computers
  - Mechanical device (moving parts)
  - Cheap but slow
- Solid-state disks (SSD)
  - All smartphones and tables, many notebooks
  - No moving parts, use NAND flash chips
  - Still a bit expensive but faster

The First Commercial Disk Drive

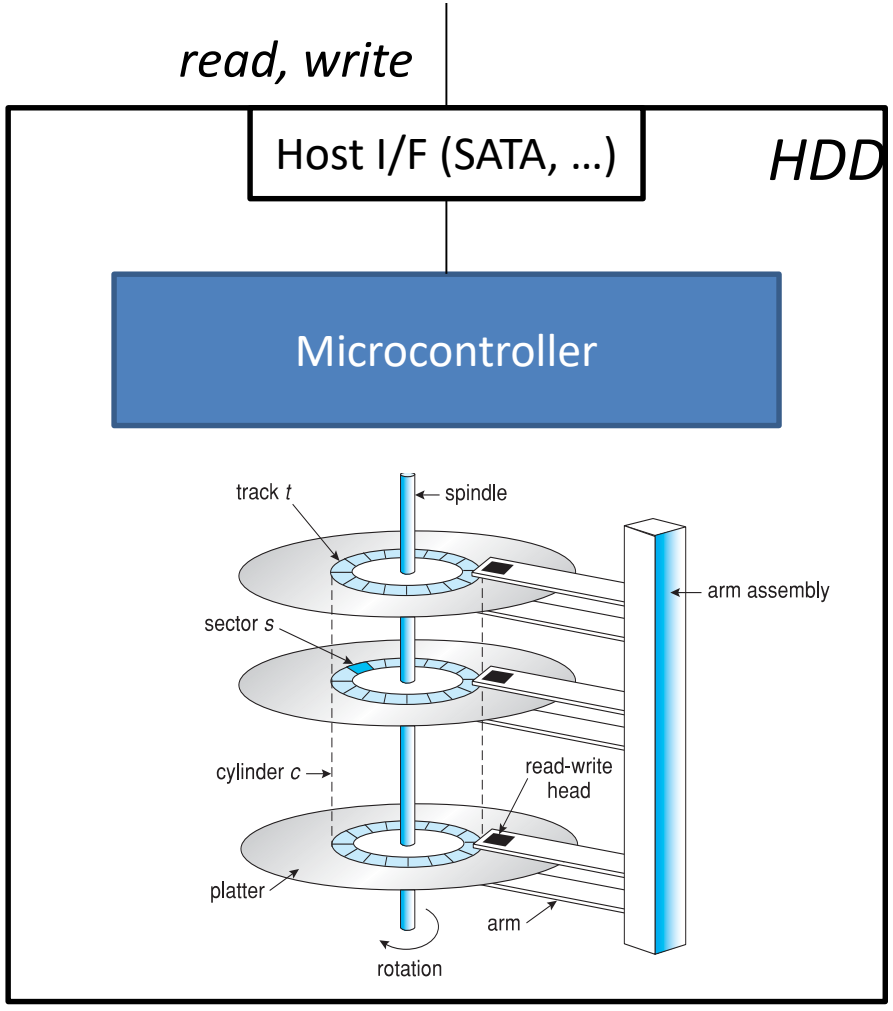
1956
IBM RAMDAC computer included the IBM Model 350 disk storage system

5M (7 bit) characters
50 x 24” platters
Access time = < 1 second

Magnetic Disk



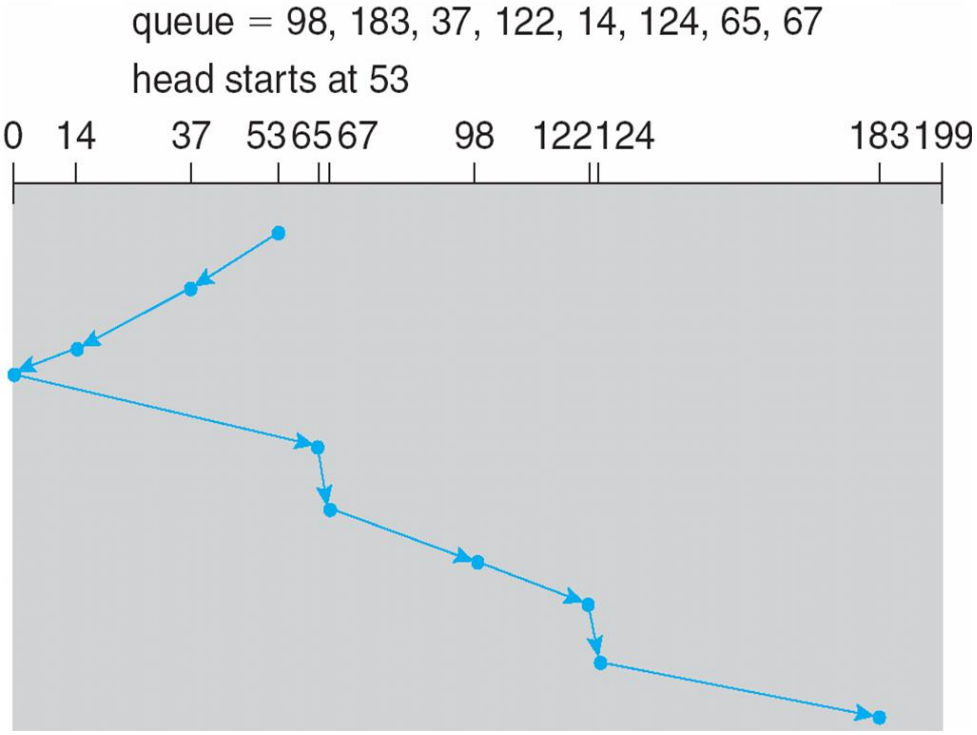
Hard Disk Drive (HDD)



- Storage size
  - ~ 3TB
- Performance
  - B/W: ~1Gb/s
  - Seek time: 3-12ms

Disk Scheduling

- Goal: minimize seek time
- FCFS, SSTF (shortest seek time first), SCAN



SCAN scheduling

NAND Flash Memory Chip

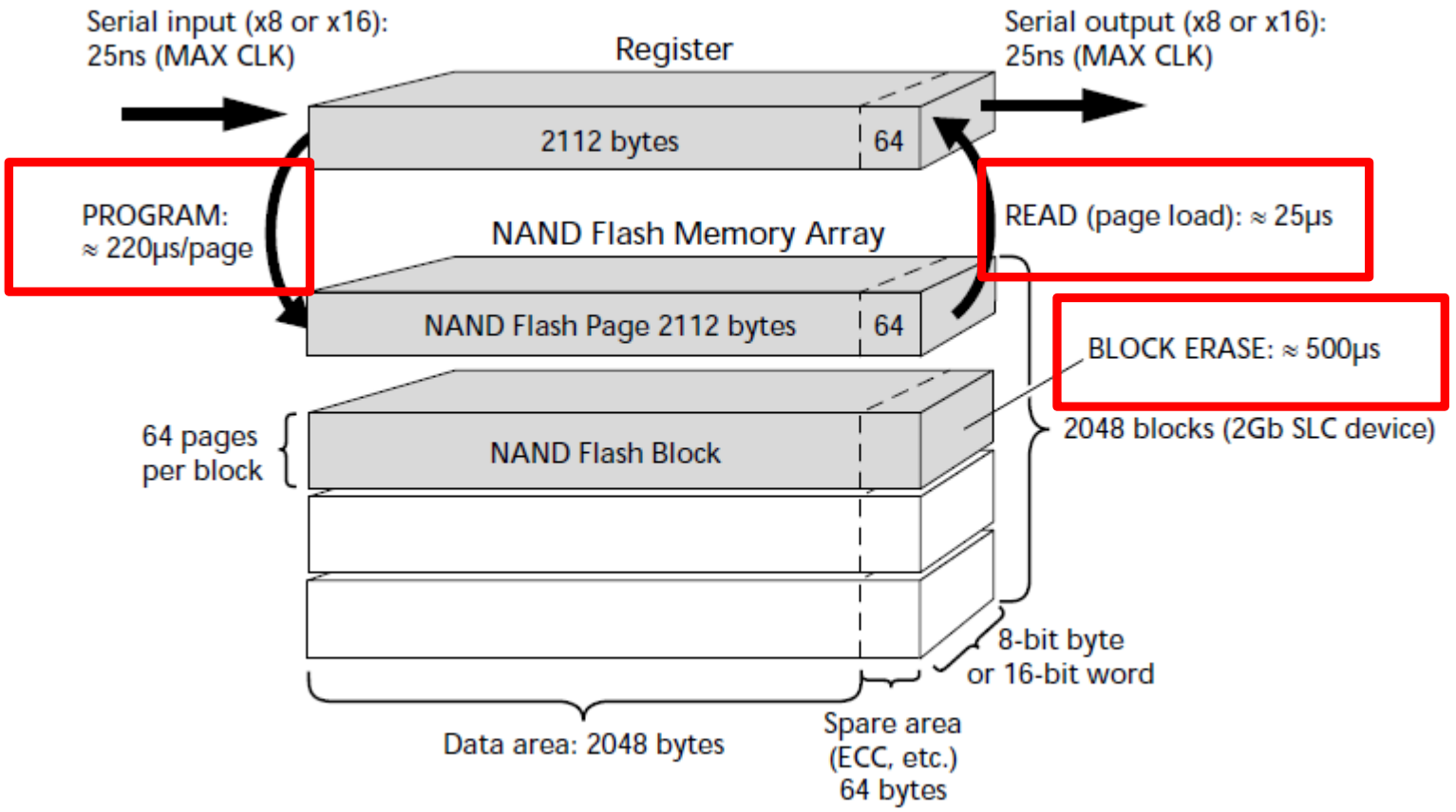
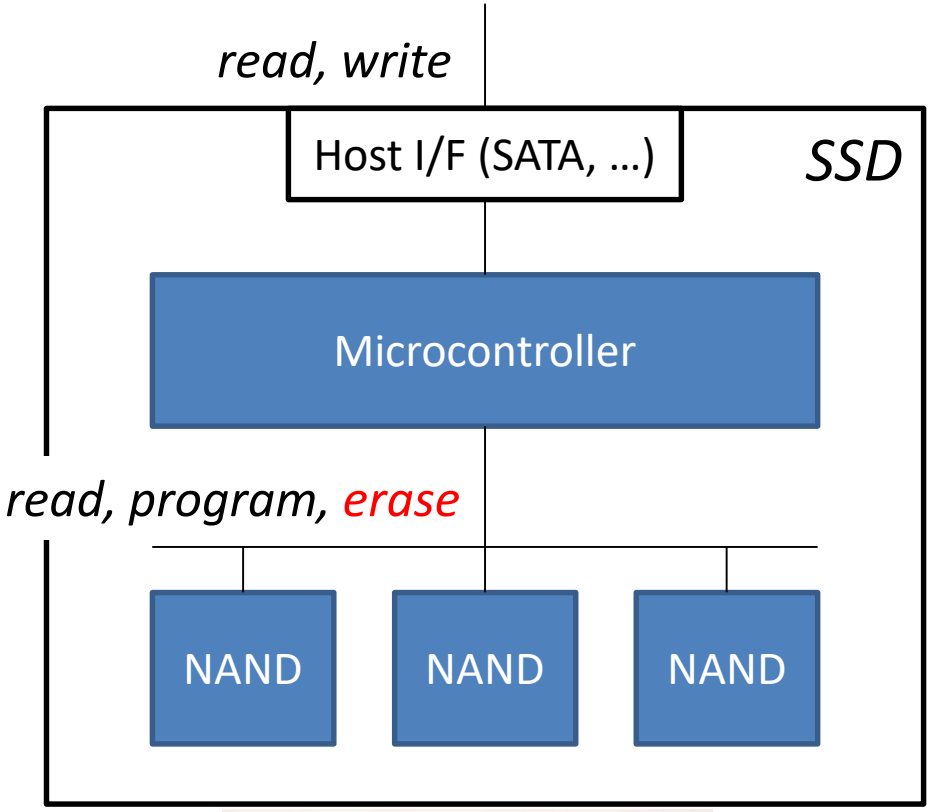


Figure source: Micron, "TN-29-19: NAND Flash 101", 2010

Solid-State Disk (SSD)



- Same I/f as HDD
  - SATA.
- Flash Translation Layer (FTL)
  - S/W running on the controller
  - Provides disk abstraction
- Storage size
  - ~1TB
- No seek time
- Bandwidth
  - SATA (6Gbps) is the bottleneck
  - Some use PCIe I/F

Summary

- I/O subsystems
  - Standardized interfaces to access various i/o devices
- I/O device types
  - Block, characters, network devices
- I/O mechanisms
  - Memory-mapped I/O, I/O instructions
  - Programmed I/O vs. DMA
- Disk
  - HDD vs. SSD