Overview

Prof. Yongtae Kim

Computer Science and Engineering
Kyungpook National University

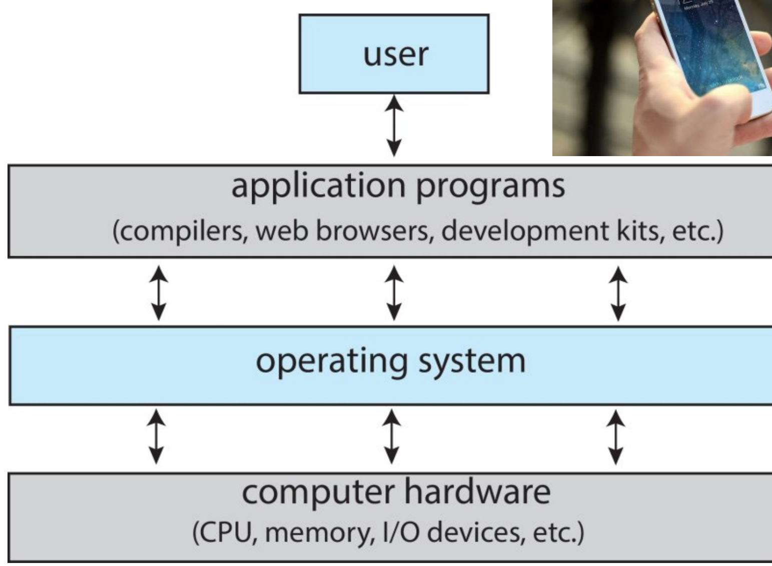
Introduction

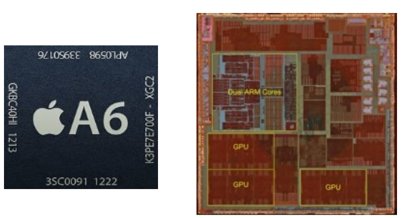
- **Operating system (OS) is software that manages a computer's hardware**
  - A program that acts as an intermediary between the computer user and the computer hardware
- **Operating systems are everywhere,**
  - from cars and home appliances that includes Internet of Things (IoTs) devices to smart phones, personal/enterprise computers, and cloud computing
- **The goals of operating systems**
  - Execute user pgroams and make solving user problems easier
  - Make the computer system convinent to use
  - Use the computer hareware in an efficient manner

Computer System

App
OS
HW

- **A computer system can be divided into four components: hardware, operating system, application programs, a user**
  - Hardware: central processing unit (CPU), memory, input/output (I/O) devices
  - Application programs: define the ways where these resources are used to solve users' computing problems (e.g. word processors, web browsers, etc)
  - Operating systems: control the hardware and coordinate its use among various application programs for users
  - Users: people, machines, other computers





What Operating Systems Do

- **An operating system is similar to a government**
  - Like a government, it performs no useful function by itself
  - It provides an environment within which other programs can do useful work
- **In user view, the OS is designed for ease of use without considering resource utilization**
  - The user interface for mobile devices (e.g., smartphones) features touch screens, voice recognitions
  - Some computers have little or no user view, such as embedded computers, and their OS is designed to run without user intervention
- **In system view, the OS can be considered as a resource allocator and a control program**
  - The OS acts as the manager of computing system resources and a control program that manages the execution of user programs to prevent errors

Defining Operating Systems

- **The OS covers many roles and functions and, in general, we have no completely adequate definition of an OS**
  - “Everything a vendor ships when you order an OS” can be a rough definition but this varies greatly across systems (less than a MB ~ GBs)

- **A more common definition is that “the one program running all time on the computer”, which is called kernel, part of OS**

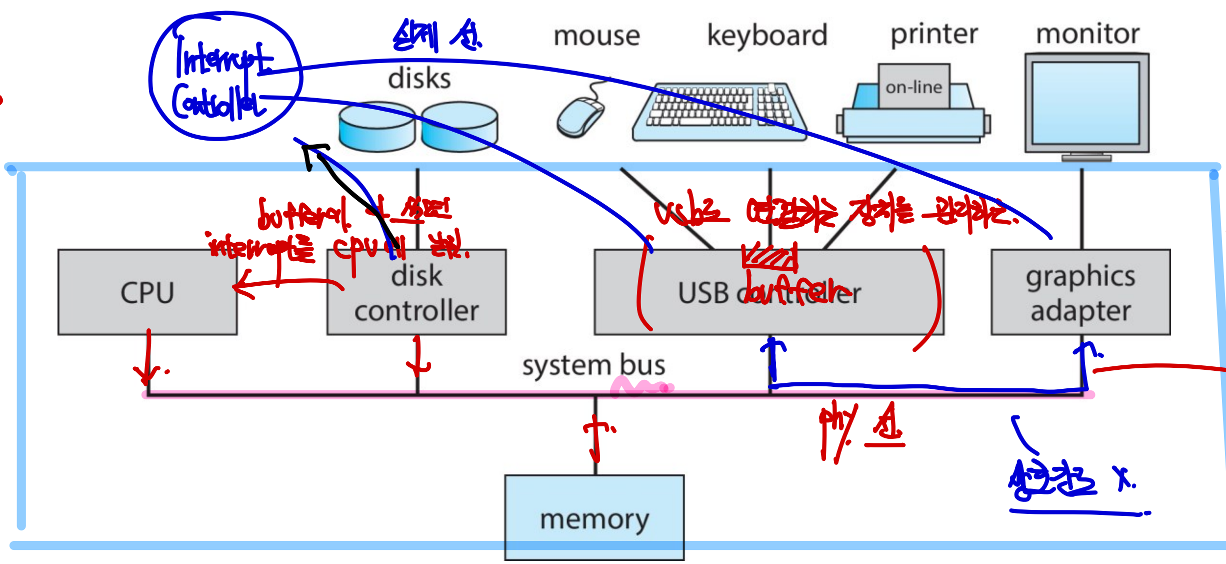
- **Along with the kernel, there are two other types of program:**
  1) System programs that ships with OS but not part of the kernel
  2) Application programs that include all programs not associated with OS

- **Today’s OS for general purpose and mobile computing also include middleware**
  - A set of software frameworks that provide additional services to application developers, such as databases, multimedia, and graphics

Computer System Organization

- A computer includes CPUs and device controllers connected through **common bus** providing access to shared memory
  - Each controller is in charge of a specific type of device and maintains some local buffer and special-purpose registers → CPU의 register와는 조금 다름. / CPU가 command를 인식하기 위한 것 = CPU와 소통하기 위함임.
  - The controller is responsible for moving data between devices and its buffer
- OSes have a **device driver** for each device controller, which provides the rest of OS with uniform interface to device
  → device 컨트롤러를 위한 인터페이스 → 메모리 사이클 내에 경쟁
  - CPU and controllers can execute in parallel, competing for memory cycles

CPU → 마우스, → 키보드) 불가능



→ 약 열두개.
약 한개 밖에 없음
마치 동시에 하는 것처럼 OS가 수행

Interrupts

→원래는 polling 방식 → Interrupts

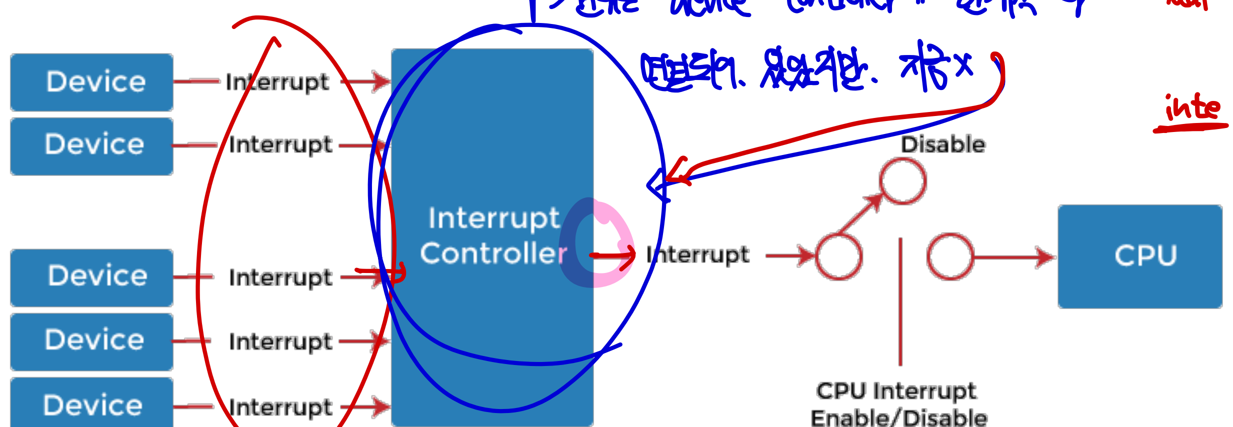
controller

driver

- **The device controller informs the CPU that it has finished its operation by causing an interrupt**

1) To start an ~~I/O operation~~, the device driver loads the appropriate registers in the device controller의 register에 접근.
2) The device controller, in turn, examines the contents of these registers to determine what action to take을 해야 할지 결정
3) The controller starts transfer of data between the device and its local buffer
4) Once the transfer is complete, the device controller issues an interrupts
5) The driver gives control to other parts of the OS

→원래는 device controller의 한계였기 때문 local buffer에 담겨야!

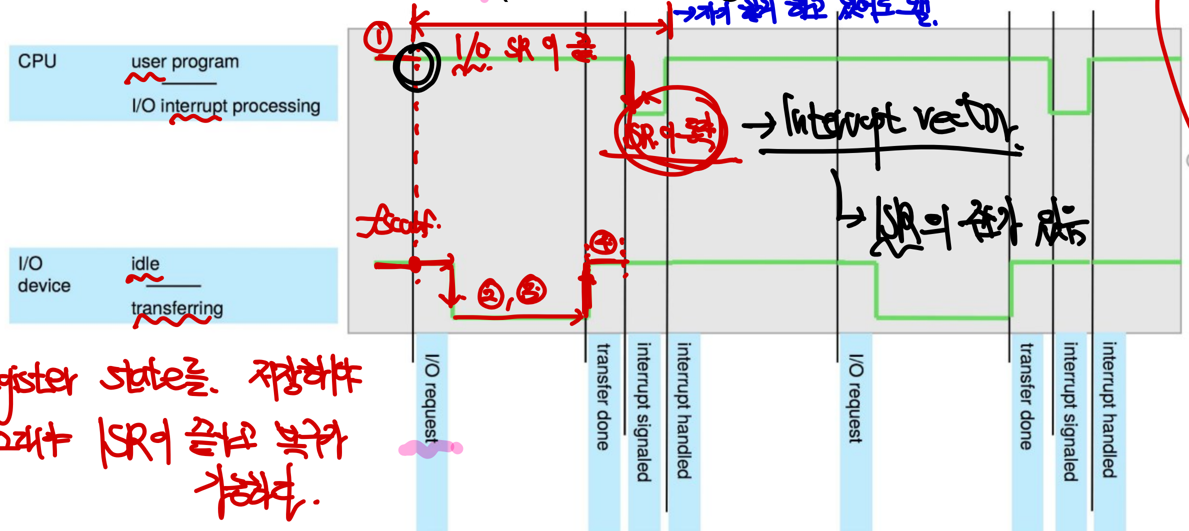


↳ priority Queue.

Interrupt Timeline

- **Interrupts are a key part of how the OS and hardware interact, and the OS works in interrupt driven way**
  - When CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location (i.e. starting address where the service routine)

ISR로 감.



table

K.I
M.I.
여기서 얻을 해야 할 일

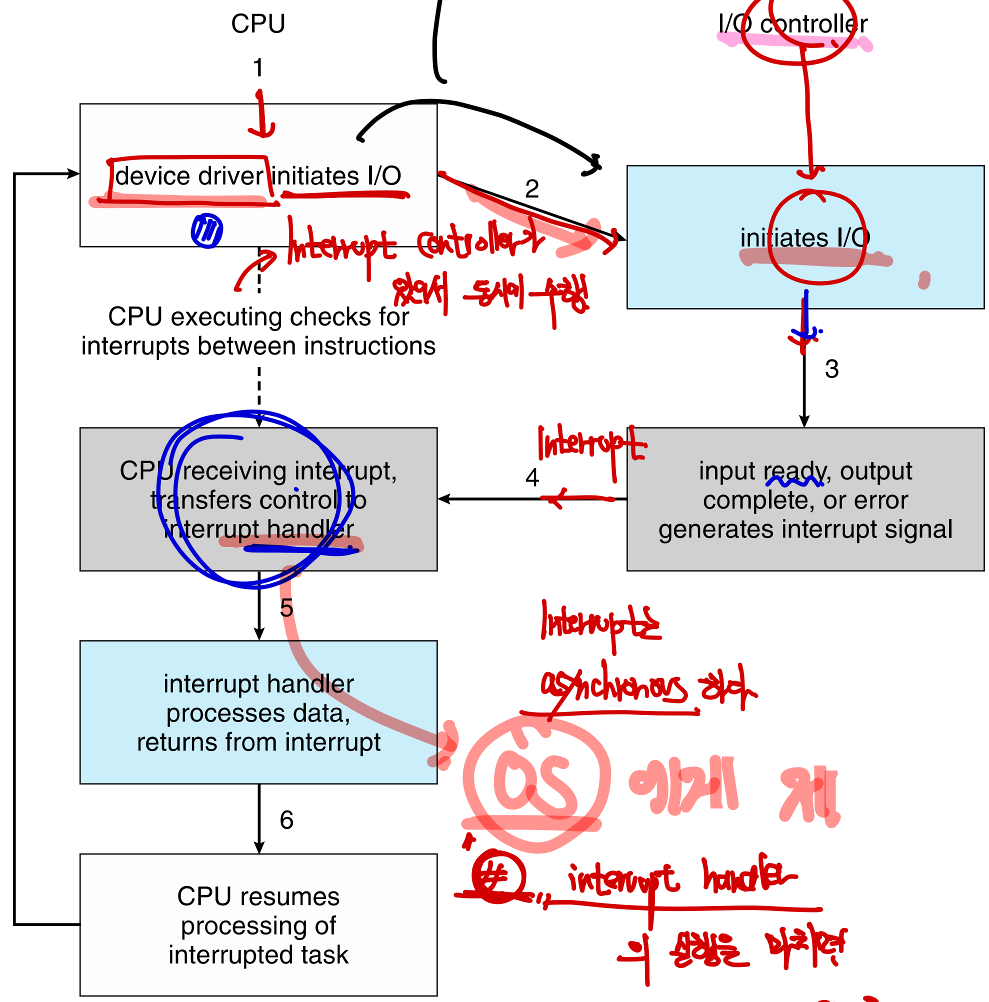
이전의 register state를 저장해야 한다. 그래야 ISR이 끝나고 복귀가 가능하다.

- **The interrupt must transfer control to an appropriate routine**
  - Interrupts must be handled quickly, as they occur very frequently
  - The interrupt routine is called indirectly through a table called interrupt vector, which contains the addresses of all the interrupt service routine → interrupt가 발생했을때 실행되는 프로그램
  - Interrupt architecture must save a state information so it can restore

Interrupt Driven I/O Cycle

- **The basic interrupt mechanism works as follows:**
  - The device controller raises an interrupt by asserting a signal on interrupt request line (wire)
  - The CPU catches the interrupt and dispatches it to the interrupt handler
  - The handler clears the interrupt by servicing the device
- **Modern computer includes interrupt controller hardware to deal with sophisticated interrupt handling features**
  - e.g.) (interrupt prioritization)

메모리에서 데이터를 읽는 것 진행



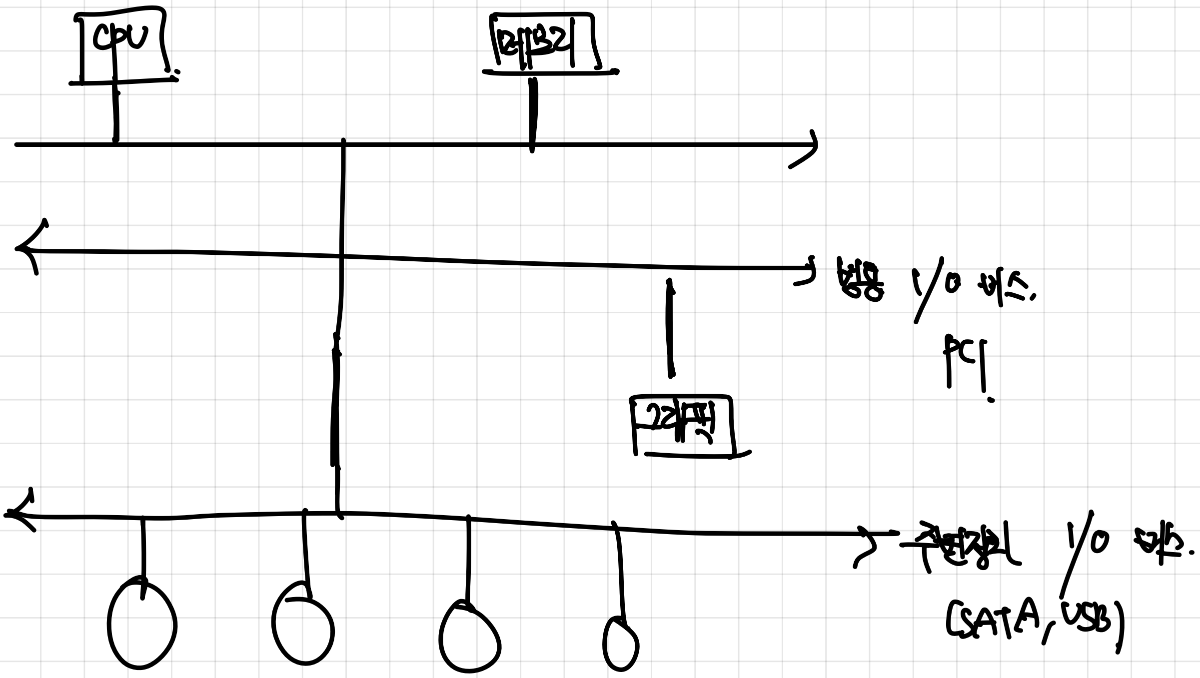
Interrupt controller가 있어서 동시에 수행

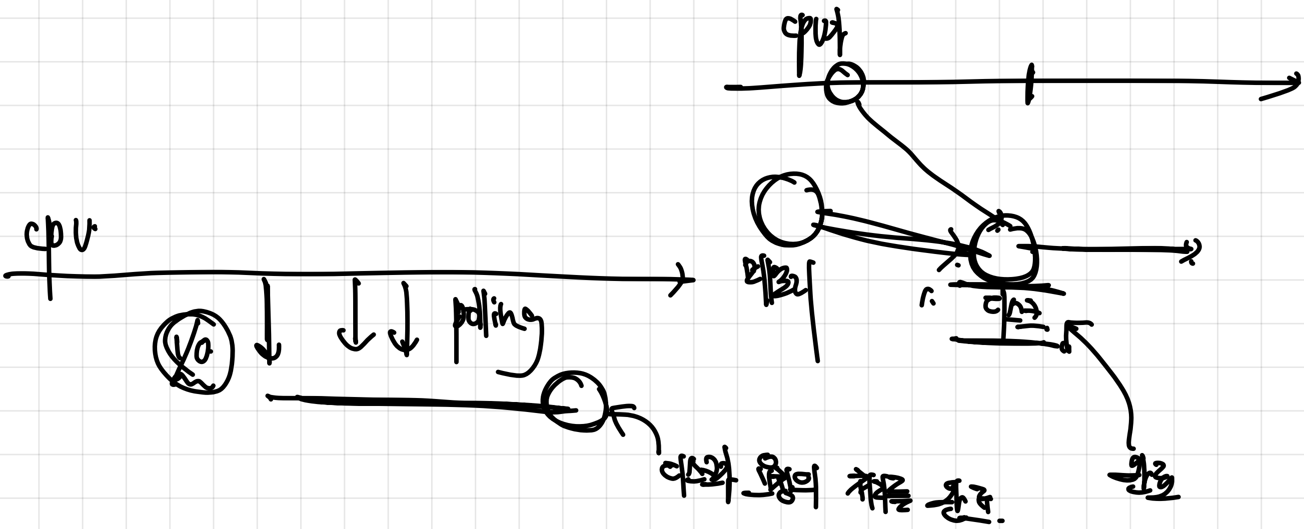
Interrupt

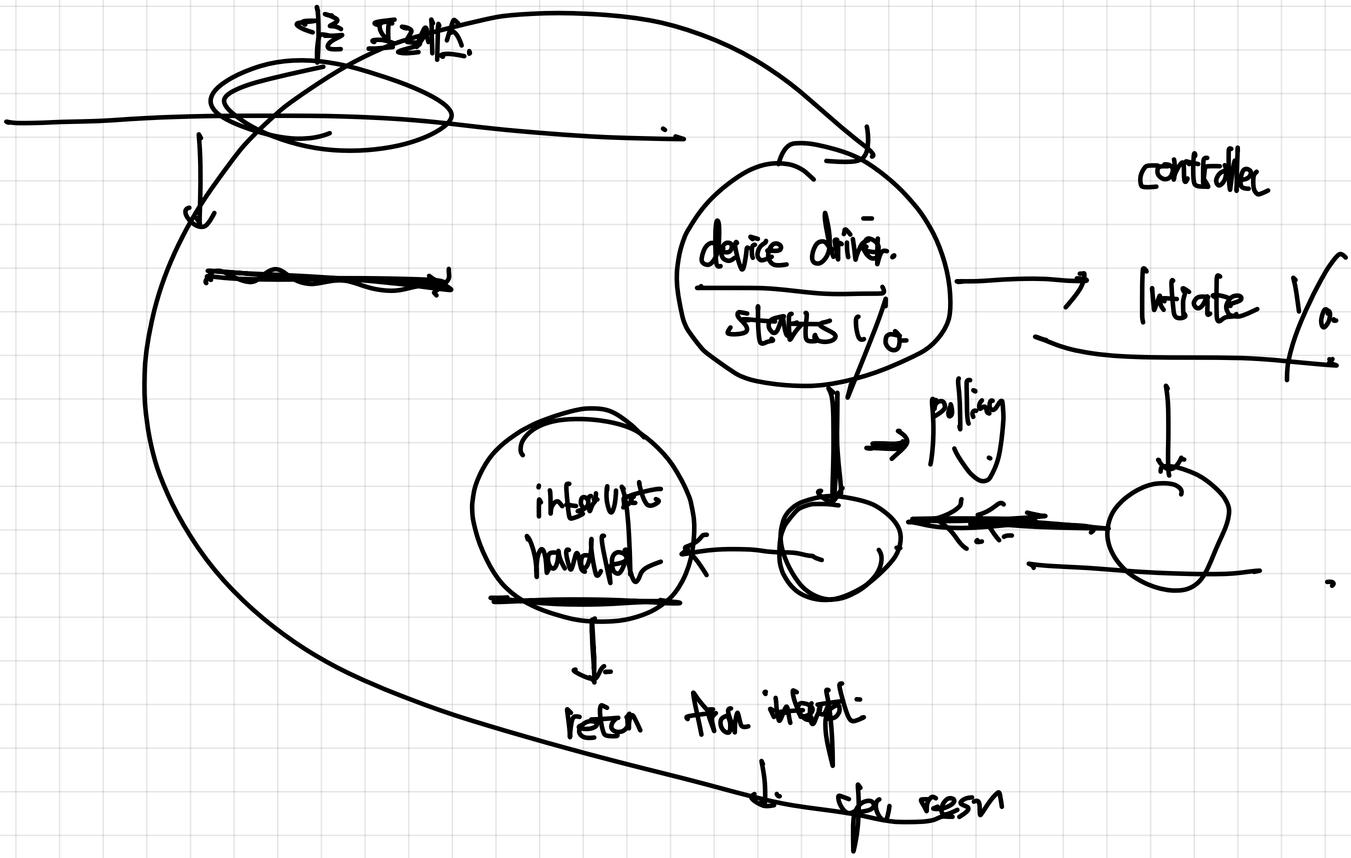
Interrupt는 asynchronous 하다

OS에게 제

interrupt handler의 실행을 마치면 OS에게 제어를 줌


CPU
메모리
범용 I/O 버스.
PCI
그래픽
주변장치 I/O 버스.
(SATA, USB)


CPU가
cpu
I/O
polling
메모리
디스크
디스크가 요청의 처리를 완료.
완료


다른 프로세스.
controller
device driver.
starts I/O
initiate I/O
polling
interrupt handler
return from interrupt
cpu resume

device driver → OS의 명령을 하드웨어가 이해하기 위한 인터페이스

Storage Structure

- **The CPU can load instruction only from memory**
  - Main memory is common implemented in dynamic random access memory (DRAM), which is a volatile memory that loses its contents when power off
    휘발성

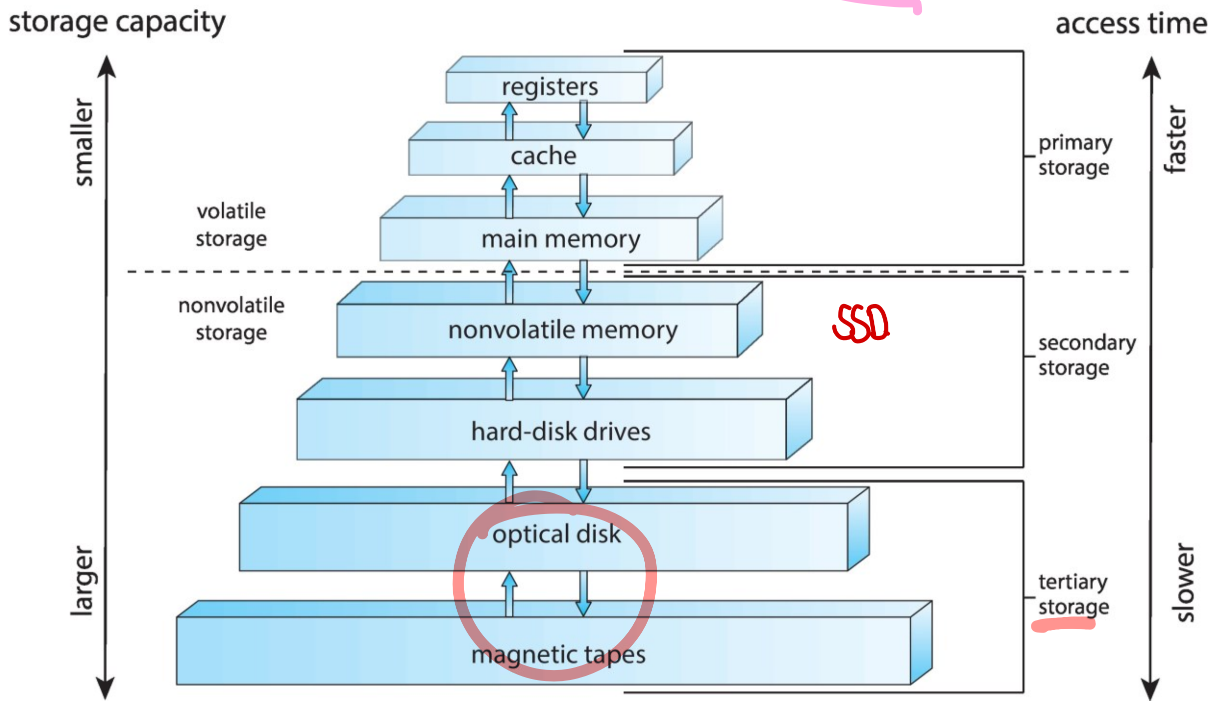
운영체제 하드웨어 초기화, 부팅

- **Computers use other forms of memory**
  (disk에 있으면 너무 느림., 메모리 → 지워짐)
  여기에 저장됨
  - Bootstrap program, which is loaded at computer power-up/reboot, is typically stored in electrically erasable programmable read only memory (EEPROM)
    ?
  - Generally known as firmware and initialize the system and load the OS kernel
  - Also known as basic input output system (BIOS) in computers
    or UEFI
    프로그래밍 가능 (특정 전압을 가해서)

- **Ideally, we want the programs and data to reside in main memory permanently, but not possible**
  - Main memory is usually too small to store all needed programs and data permanently
  - Main memory is volatile

Secondary Storage

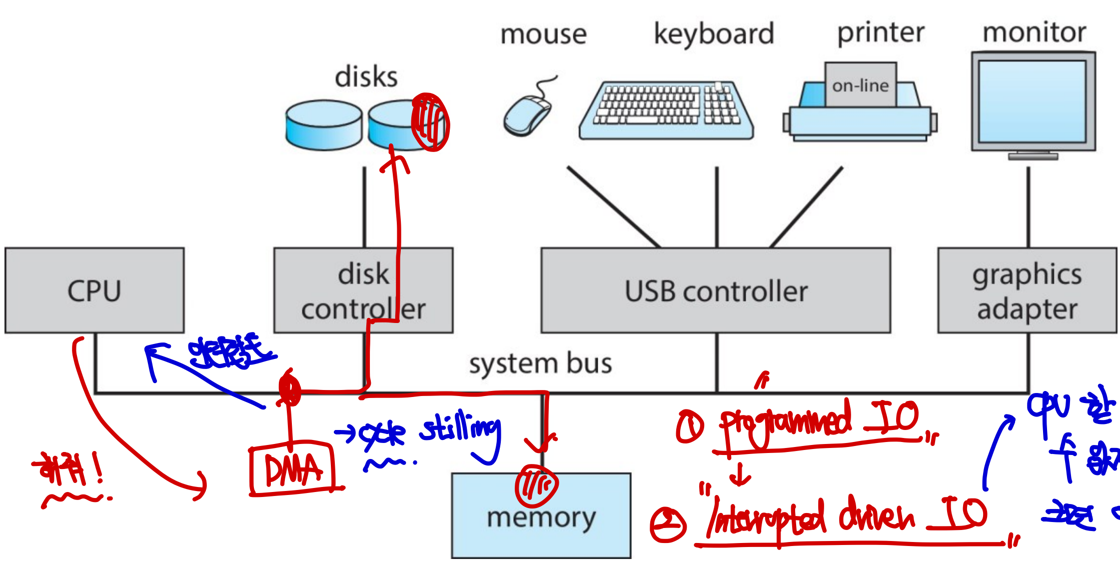
- **Most computer systems provides secondary storage as an extension of main memory**
  - Hard-disk drives (HDDs) and nonvolatile memory (NVM) devices are most common as the secondary storage
- **Tertiary storage is slow enough and large enough (e.g. blu-ray)**
  - They are used only for special purposes (e.g. backup)

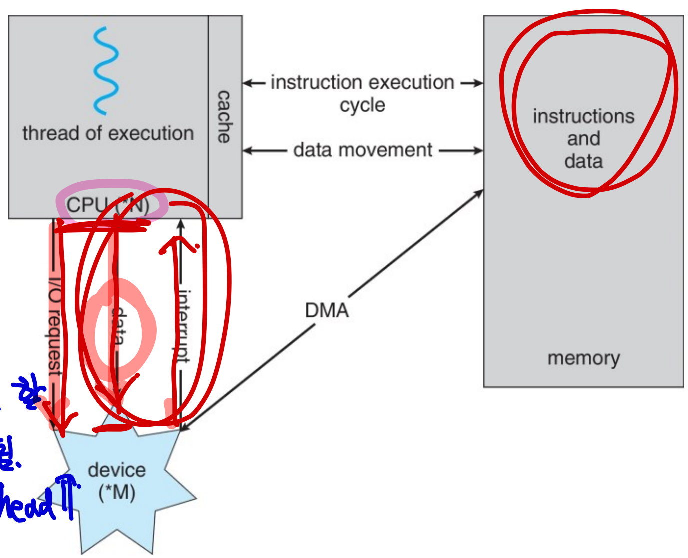


I/O Structure

CPU를 대신하여 I/O장치와 Memory 사이의 데이터전송을 담당하는 장치
→ CPU의 개입없이 주변장치와 주기억장치와의 데이터 전송이 이뤄짐.

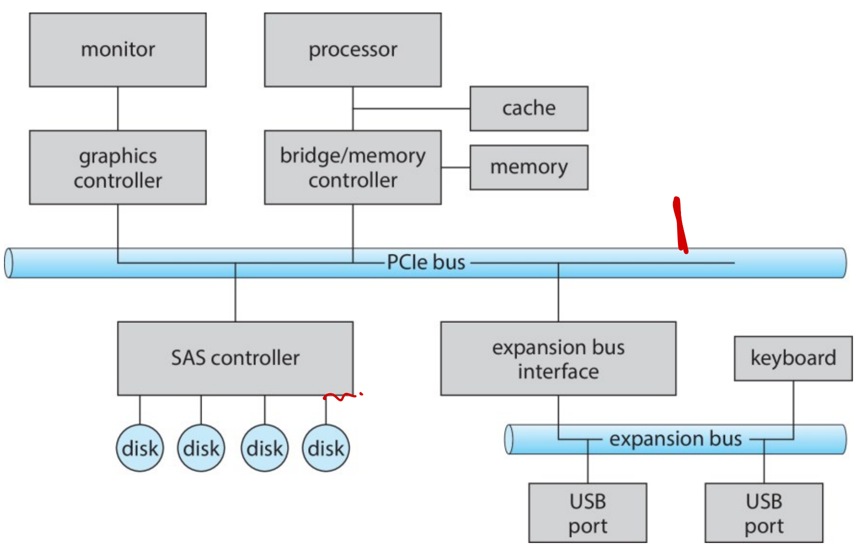
- **The form of interrupt-driven I/O in a common bus is fine for moving small amounts of data**
  - However, this can produce high overhead when used for bulk data movement
- **To resolve this problem, direct memory access (DMA) is used**
  → 인터럽트의 발생 횟수는 최소화.
  - The device controller transfers data directly to/from the device and main memory with no CPU intervention by the DMA controller
  - Only one interrupt is generated per block, rather than one interrupt per byte

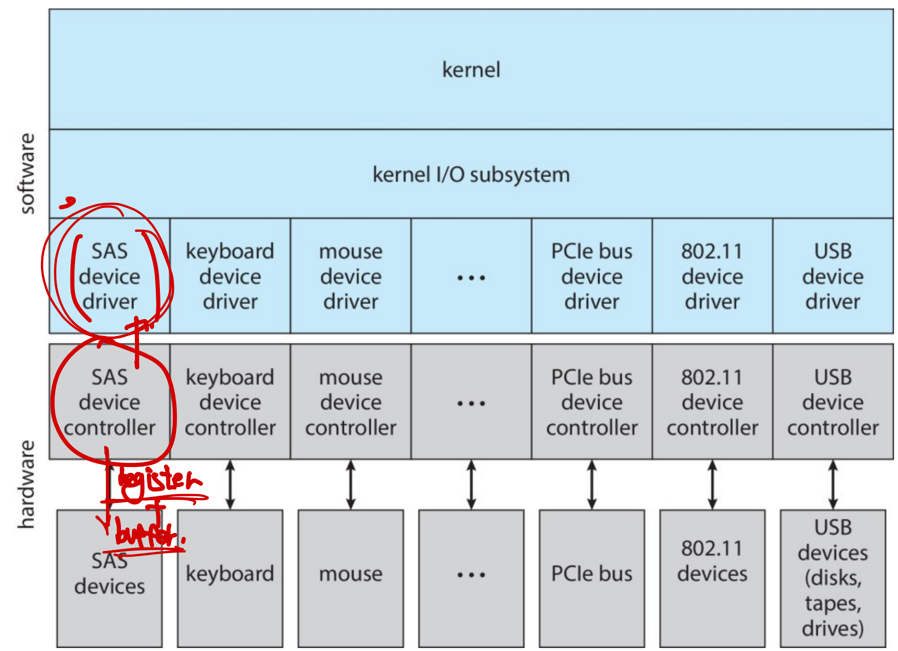




Typical PC I/O Structure

- **Computers operate a great many kinds of devices**
  - Storage devices, network devices, and human-interface devices
  - They are connected each other in a common bus (e.g. PCI bus)
- **Each OS has its own I/O subsystem structures and device driver frameworks**
  - The device drivers hide the details of device interactions from kernel





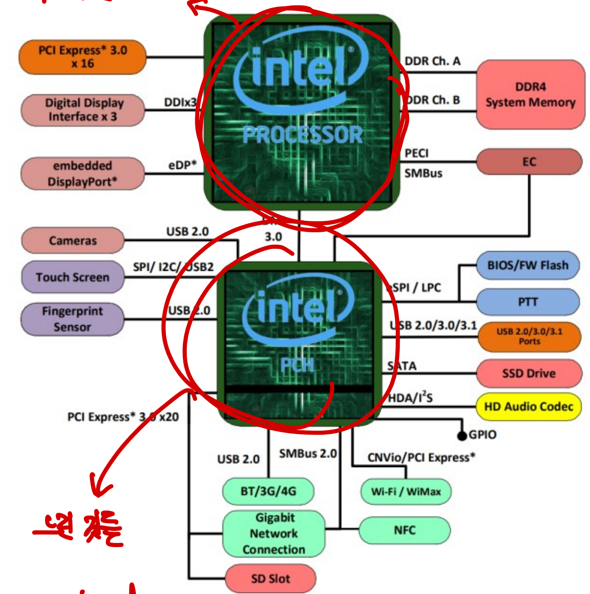
Disk I/O Example

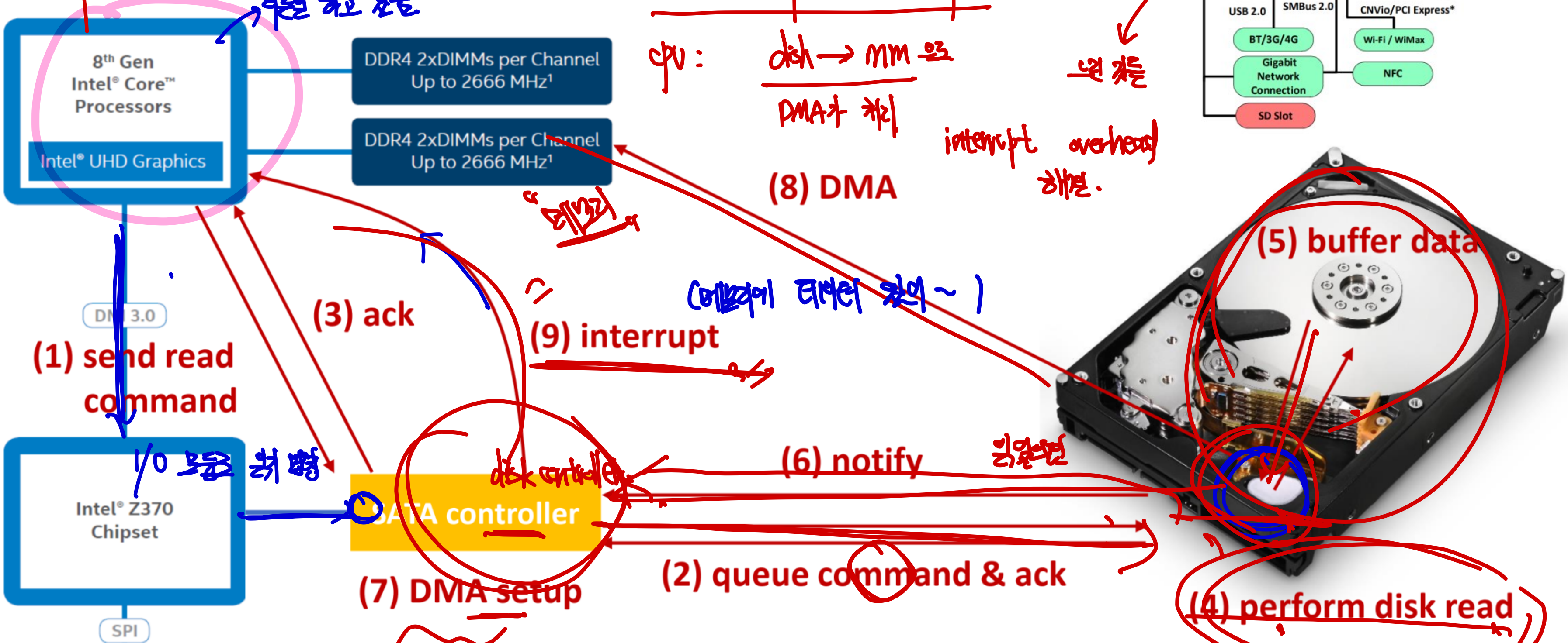
- **SATA disk read operation using DMA**
  - Intel's processor platform point of view
  - The CPU initiates a read command and forget
  - Once the DMA transfer is done, the CPU gets notified by an interrupt

끝마치면. CPU에게 interrupt

CPU : disk → mm 으로 DMA가 처리

interrupt overhead 해결.





Courtesy of Prof. Jin-Soo Kim @ SNU