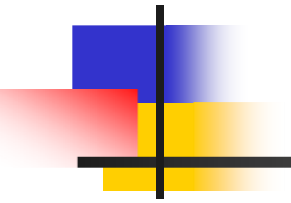


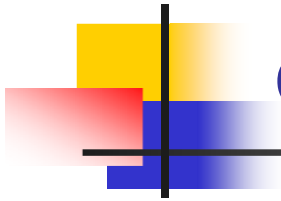
Operating Systems

(Operating System Structure)

Chapter 1-2

These lecture materials are modified from the lecture notes written by A. Silberschatz, P. Galvin and G. Gagne.

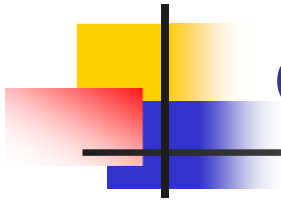
August, 2022



Outline

- What Operating Systems Do
- Computer-Systems Organization
- Computer-Systems Architecture
- Operating Systems Structure

What Operating Systems Do



Operating system (운영 체제)

❖ 단 한 가지로 표현되는 정의는 없음

- (1) 컴퓨터 하드웨어를 관리하는 소프트웨어
  - 컴퓨터 하드웨어에는 CPU, 메모리 및 I/O 장치와 스토리지가 포함
  - 운영 체제의 기본적인 책임은 리소스를 프로그램에 할당하는 것
  - 올바른 작동을 보장하고, 방해하지 않도록 프로그램을 관리 및 제어

- (2) 사용자와 컴퓨터 하드웨어 사이의 중개자
  - 사용자가 프로그램을 편리하게 사용할 수 있는 환경을 제공

The components of a computer system

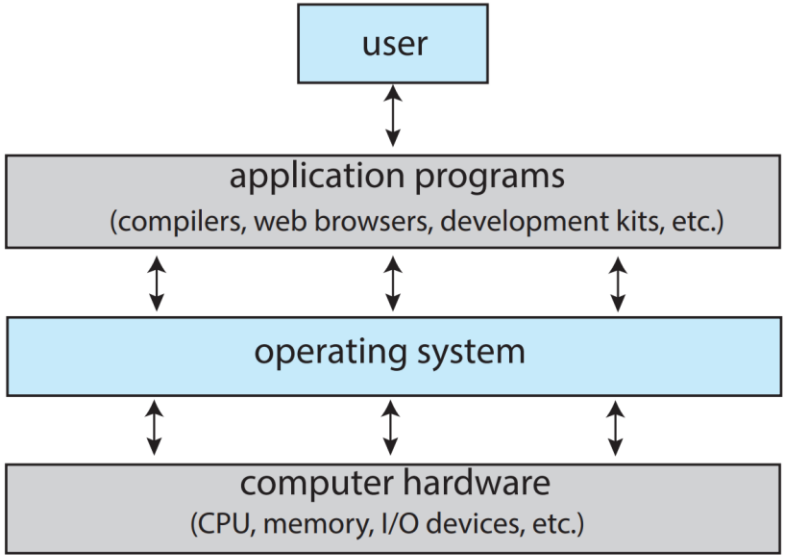


Figure 1.1 Abstract view of the components of a computer system.

- 사용자
- 하드웨어
  - 시스템의 기본 컴퓨팅 리소스를 제공
- 응용 프로그램
  - 워드 프로세서, 스프레드시트, 컴파일러 및 웹 브라우저 등
  - 하드웨어의 리소스를 사용함
- 운영체제
  - 다양한 사용자와 다양한 응용 프로그램 간의 리소스 사용을 조정

Operating system (운영 체제)

- 운영체제는 커널(Kernel)과 시스템 프로그램으로 구분됨

- OS가 수행되려면 소프트웨어로서 전원이 켜짐과 동시에 메모리에 올라가야 함.
  - 하지만, OS는 모두 메모리에 올라가기에는 너무 큼.
- OS 중 항상 필요한 부분만을 메모리에 상주시키고 그렇지 않은 부분은 필요할 때 메모리에 올려서 사용함.

- 이 때 메모리에 상주하는 OS의 일부분을 커널이라고 함.

Operating system (운영 체제)

- OS와 사용자들은 컴퓨터 시스템의 하드웨어 및 소프트웨어 리소스를 공유하기 때문에, 응용 프로그램이 함부로 시스템의 영역에 접근하지 못하도록 해야함.
- OS는 커널 모드(Kernel mode)와 유저 모드(User mode)로 나뉨. mode bit를 사용하여 이를 구분하며, 커널 모드는 0, 유저 모드는 1.
  - 응용 프로그램이 실행될 때 -> 유저 모드
  - 응용 프로그램이 시스템 콜을 통해 운영 체제에서 서비스를 요청하면 -> 커널 모드로 전환됨

Operating system (운영 체제)

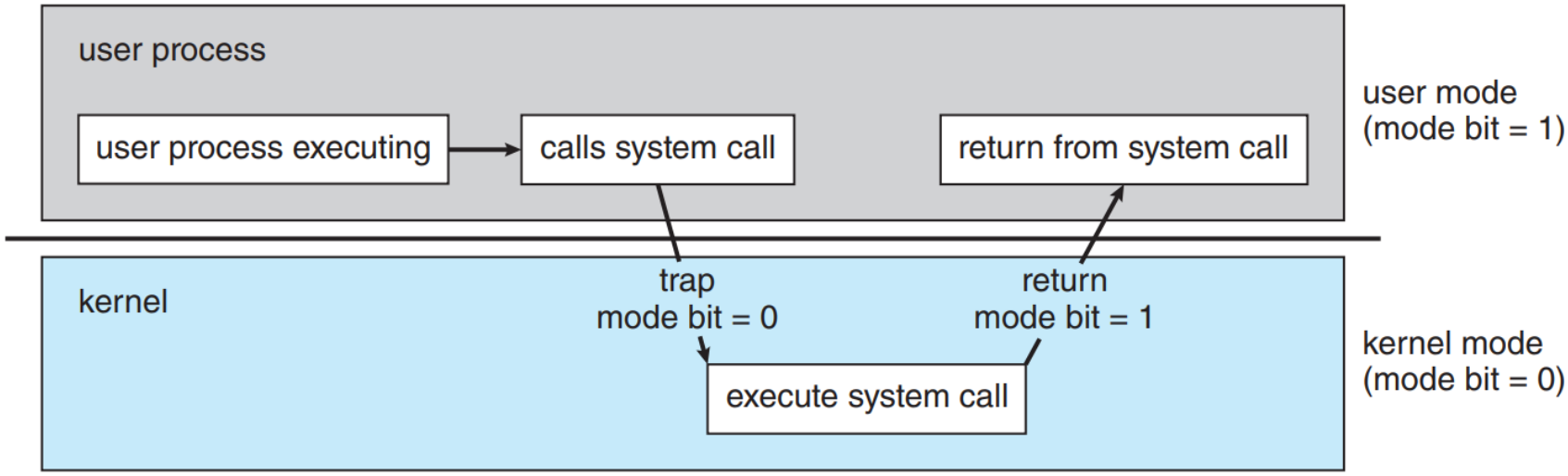
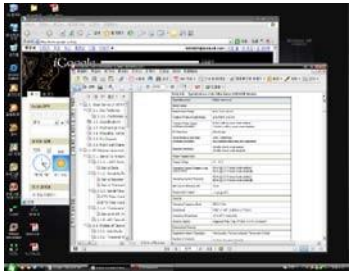


Figure 1.13 Transition from user to kernel mode.

User View

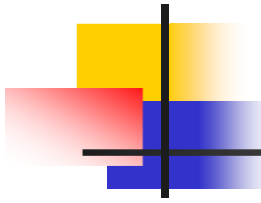
- User view의 목표
  - 사용자가 프로그램을 실행함으로써 문제를 쉽게 해결해야함
  - 사용자가 컴퓨터 시스템을 사용하기 편리하게 해야함

DOS



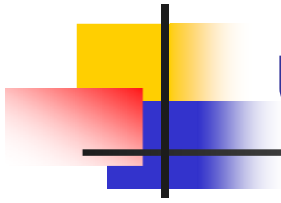
windows

iPAD



System View

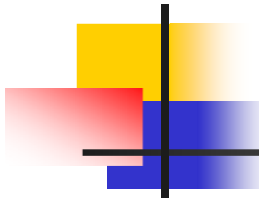
- System view의 목표
  - 컴퓨터 시스템을 효율적으로 운영하기 위한 리소스 관리자
  - 다양한 I/O 장치와 사용자 프로그램을 제어하여 오류 및 부적절한 사용을 방지



User View / System View

- 사용자에게 제공되는 유용한 기능
  - User interface
  - Program execution
  - I/O operations
  - File-system manipulation
  - Communications
  - Error detection

- 시스템의 효율적인 작동을 보장하기 위한 동작
  - Resource allocation
  - Logging
  - Protection and security



Operating system (운영 체제)

- 왜 공부해야 하는가?

  - 모든 코드는 OS 위에서 실행됨

  - OS에 대한 지식을 통해, 프로그래머는 작성한 코드의 성능을 효과적으로 개선할 수 있음

Computer-Systems Organization

범용 컴퓨터 시스템 구성

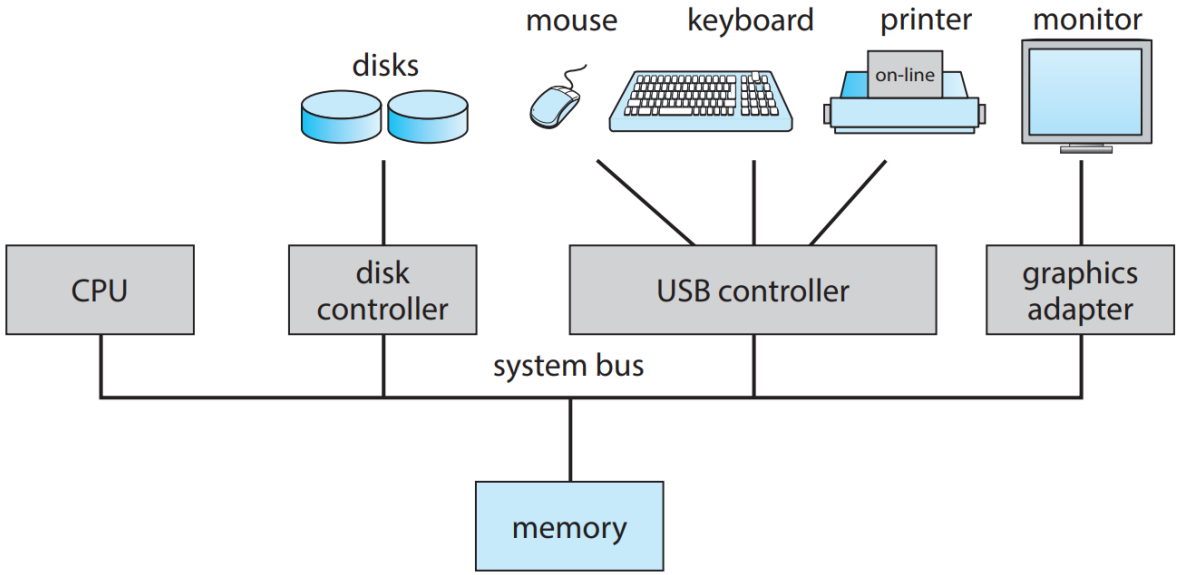


Figure 1.2 A typical PC computer system.

- 하나 이상의 CPU와 I/O device controller로 구성되며, 다른 controller와 메모리에 대한 접근을 제공하는 System bus를 통해 연결됨.
- CPU와 장치 컨트롤러는 메모리 cycle을 얻기위해 경쟁함

범용 컴퓨터 시스템 구성

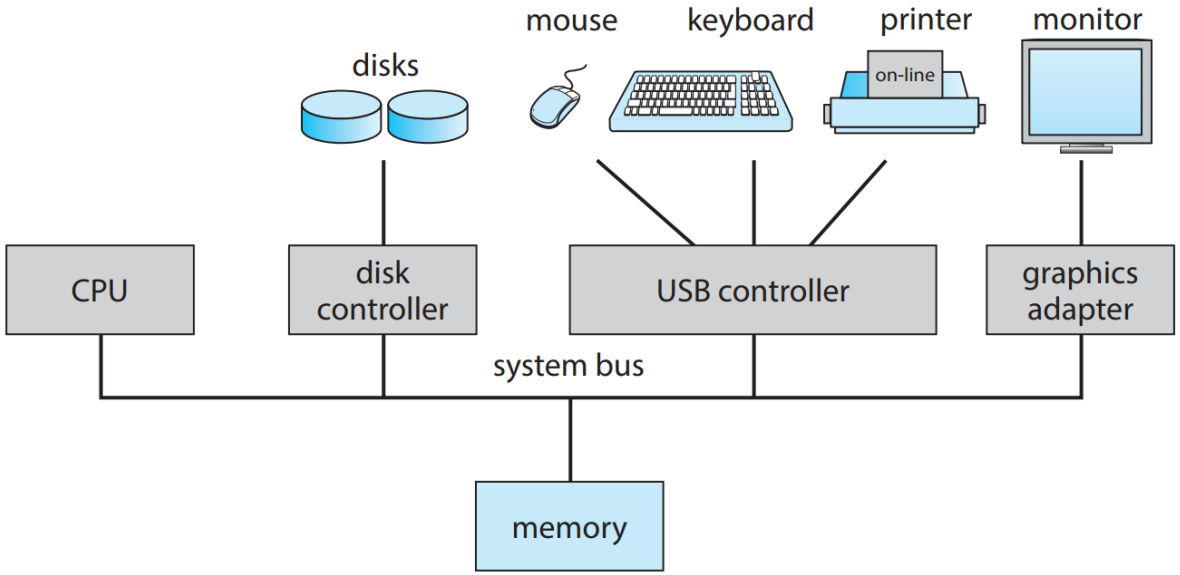
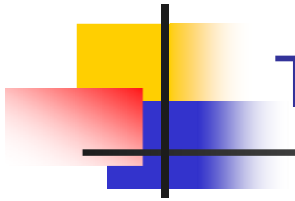


Figure 1.2 A typical PC computer system.

- 각 device controller를 위한 device driver가 있음
  - 보통 장비 제조사에서 제공
- OS는 driver를 통해 device controller에 명령을 내리고 device controller와의 인터페이스를 제공함



Three key aspects of the system

- **(1) Interrupt**
- (2) Storage structure
- (3) I/O structure

Interupt

- 운영 체제와 하드웨어가 상호 작용하는 방식의 핵심
- CPU가 인터럽트되면 하던 일을 멈추고 즉시 실행을 고정된 위치로 옮김. 인터럽트 서비스 루틴이 실행 후, 완료되면 CPU는 중단 된 일을 재개함.

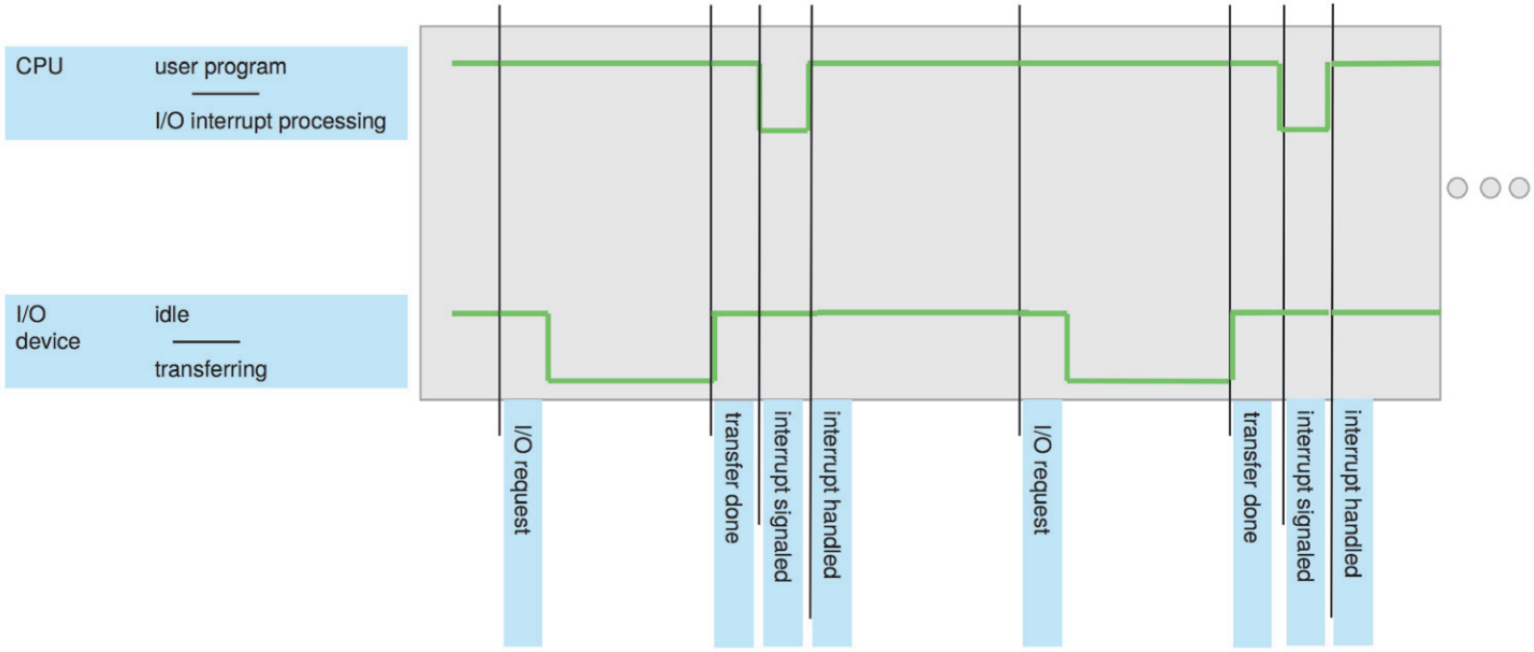
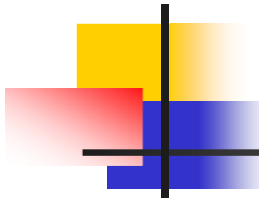


Figure 1.3 Interrupt timeline for a single program doing output.



Interupt

- 기본 인터럽트 메커니즘
  - 1) OS는 I/O request가 발생하면 device driver를 통해서 해당 device controller에 명령을 내림
  - 2) Device controller가 명령받은 동작을 수행 후, 완료되면 interrupt 를 생성함.
  - 3) CPU가 Interrupt-request line으로 신호를 발생했음을 감지함
  - 4) CPU는 진행 중인 작업을 중단하고, 중단된 작업의 주소 및 상태에 대한 정보를 저장함
  - 5) 해당 인터럽트 번호를 인터럽트 서비스 루틴의 주소를 담고 있는 배열인 인터럽트 벡터에 대한 인덱스로 사용함
  - 6) 인터럽트 서비스 루틴은 인터럽트를 처리함
  - 7) 인터럽트 이전에 진행하던 작업의 주소 및 상태로 되돌림

Interupt

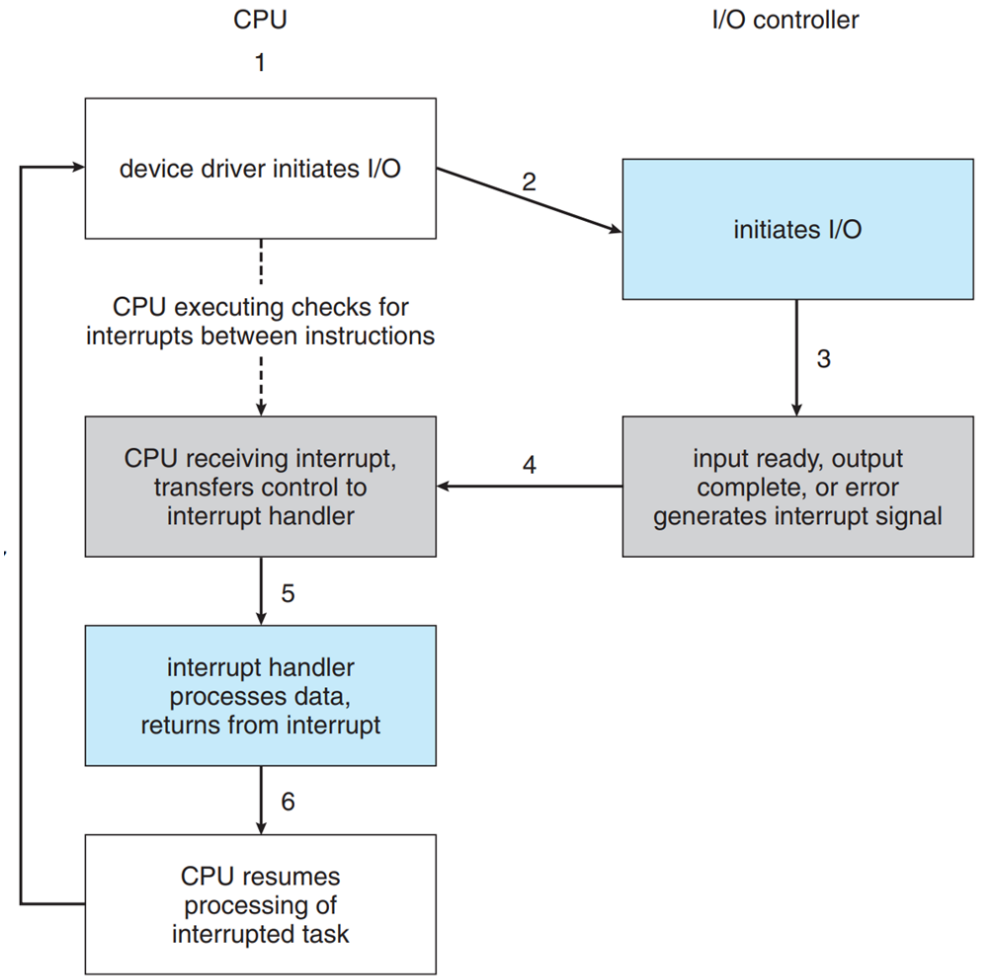


Figure 1.4 Interrupt-driven I/O cycle.

Interupt

- System call
  - OS에 서비스를 요청하기 위해 응용 프로그램에서 사용하는 메커니즘
    - Through API (Application Program Interface)

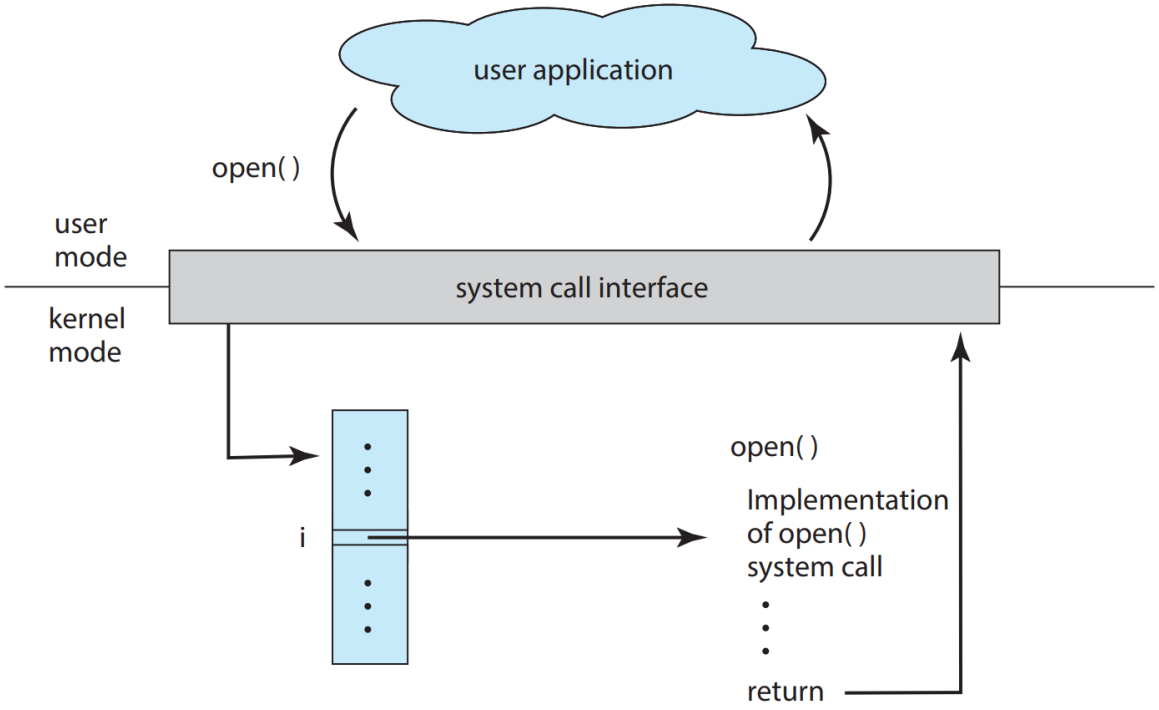


Figure 2.6 The handling of a user application invoking the `open()` system call.

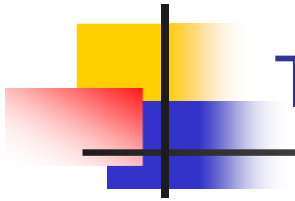
Interupt

- 내부 인터럽트 (Trap)
  - 소프트웨어에 의한 인터럽트
    - 내부 오류를 알림, 시스템 콜
  - 0에서 31까지의 이벤트
  - Synchronous

- 외부 인터럽트
  - 하드웨어에 의한 인터럽트
    - 장치에서 생성
  - 32에서 255까지의 이벤트
  - Asynchronous

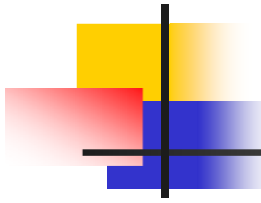
vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Figure 1.5 Intel processor event-vector table.



Three key aspects of the system

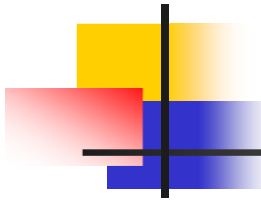
- (1) Interrupt
- **(2) Storage structure**
- (3) I/O structure



Storage structure

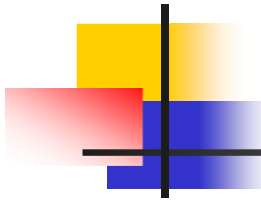
- 모든 형태의 메모리는 byte 배열을 제공함.
  - 각 byte에는 고유한 주소가 있음.

- CPU는 메모리에서만 명령을 로드할 수 있음
- 모든 프로그램을 실행하려면 먼저 메모리에 로드해야 함.
  - 메모리에서 명령어를 가져와서 해당 명령어를 명령어 레지스터에 저장.
  - 이 때 RAM이라고 하는 재기록 가능한 메모리에 로드함.
  - 일반적으로 DRAM(Dynamic Random-Access Memory)을 사용.



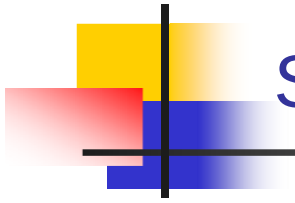
Storage structure

- 그러나 모든 프로그램과 데이터가 RAM에 영구적으로 상주하는 것은 **불가능**
  - 1) 필요한 모든 프로그램과 데이터를 영구적으로 저장하기에는 너무 작음
  - 2. RAM은 전원이 꺼지거나 다른 방식으로 손실되면 내용이 손실됨.
    - 예) Bootstrap 프로그램은 휘발되어선 안됨.
      - Bootstrap :컴퓨터 전원이 켜진 상태에서 운영 체제를 로드하기 위해 제일 먼저 실행하는 프로그램
      - 보통 ROM(Read-Only Memory)이나 EEPROM(Electrically Erasable Programmable Read-Only Memory)에 저장되어 있음.



Storage structure

- Secondary memory
  - 대용량, 비휘발성 메모리.
  - 대부분의 프로그램은 메모리에 로드될 때까지 보조 저장소에 저장됨
  - 메인 메모리보다 훨씬 느리므로, 적절한 관리가 필요함.
    - 대표적으로 HDDS(hard-disk drives: 마그네틱 기반)과 NVM(nonvolatile memory)가 있음.



Storage structure

- 다양한 스토리지 기술들의 차이점은 크게 **속도, 크기, 휘발성**
- 보통 가격 및 성능이 반비례함

Storage structure

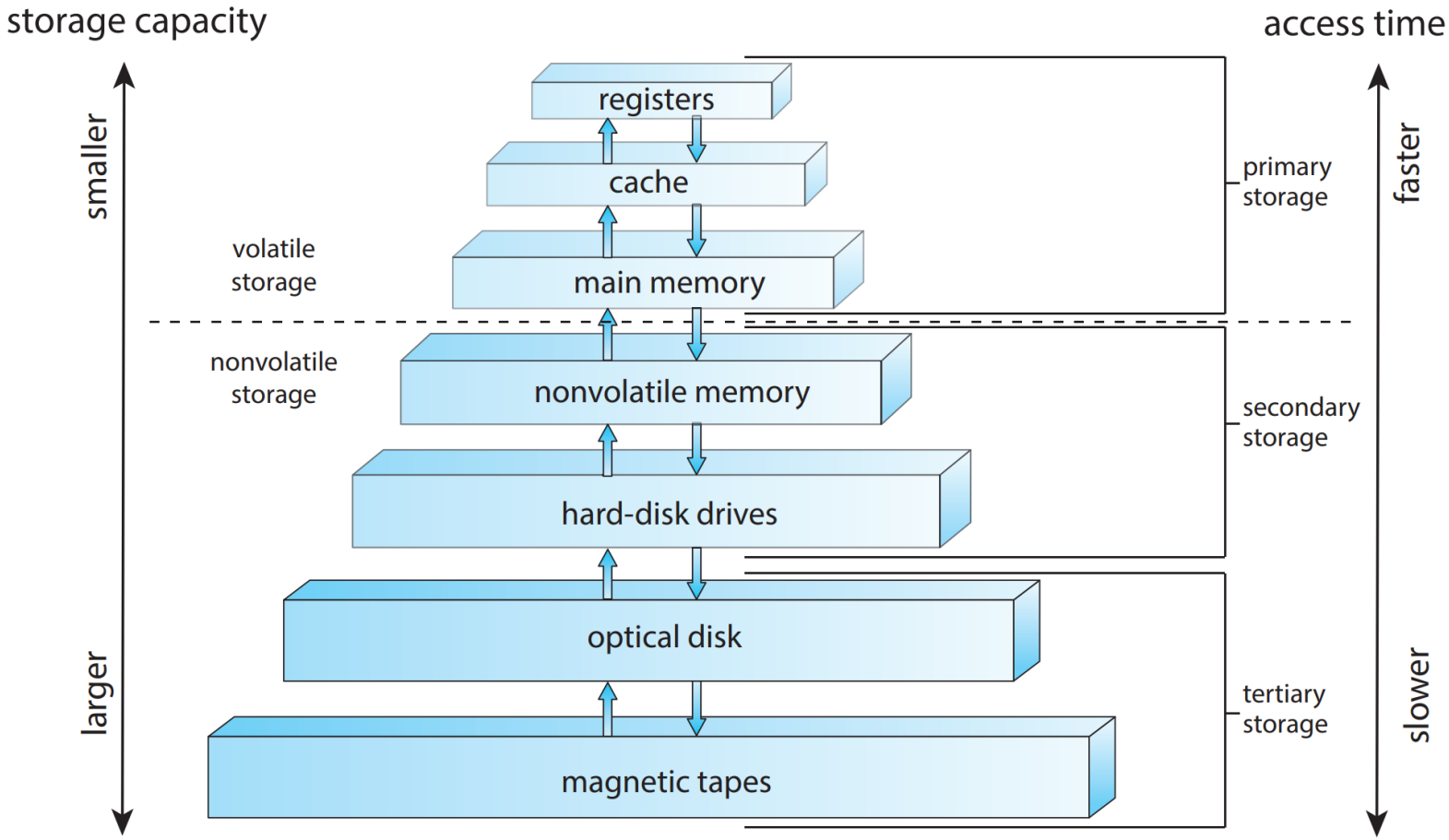
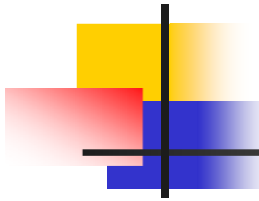
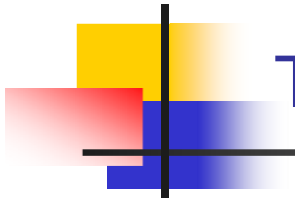


Figure 1.6 Storage-device hierarchy.



Storage structure

- Caching
  - 컴퓨터 시스템의 매우 중요한 원리
  - 정보를 더 빠른 스토리지에 저장시킴
    - 더 캐시를 먼저 확인하여 정보가 있는지 확인하고
      - 정보가 있을 경우 캐시의 정보 사용
      - 그렇지 않으면 데이터를 캐시에 복사하여 사용
  - 크기가 제한되어 있기 때문에 캐싱/교체 정책은 매우 중요함
    - 자주 쓰는 정보를 캐시에 저장하는 것이 효율적임
  - 캐시 크기 및 교체 정책을 고려해야 성능을 향상 시킬 수 있음



Three key aspects of the system

- (1) Interrupt
- (2) Storage structure
- **(3) I/O structure**

I/O structure

- 컴퓨터는 계산 혹은 I/O를 처리하는 일에 대부분의 시간을 사용
- OS는 I/O 장치와 I/O operation을 관리함.
- bus를 통해 CPU가 device controller I/O 명령을 내림.

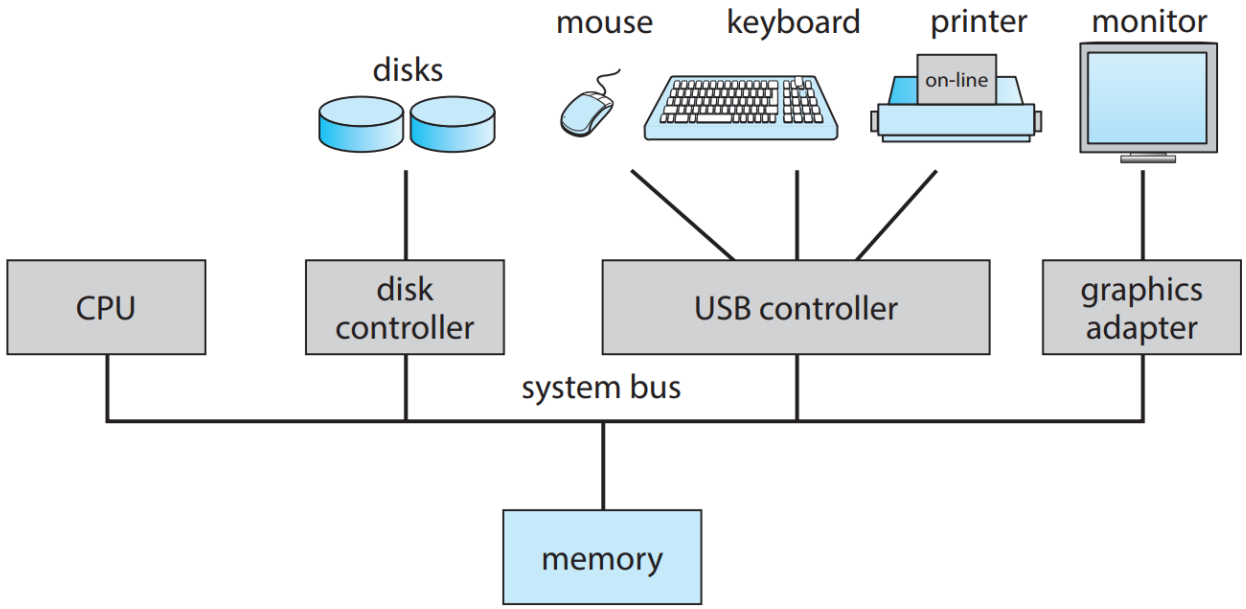
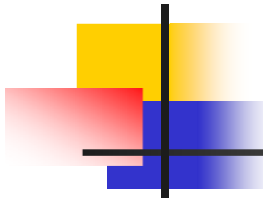
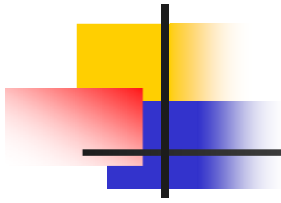


Figure 1.2 A typical PC computer system.



I/O structure

- Types of I/O
  - Polling(busy-waiting) : 종료할 때까지 반복해서 레지스터를 읽음
  - Interrupt
  - DMA : 컨트롤러에서 직접 I/O 작업을 처리하는 것을 의미
    대량 데이터 이동에 적합



I/O structure

- 범용 컴퓨터 시스템은 공통 버스를 통해 데이터를 교환하는 여러 장치로 구성되어 있음
- 인터럽트 구동 I/O 형식은 소량의 데이터를 이동하는 데 적합하지만, 대량 데이터 이동에 사용할 경우 높은 오버헤드를 생성할 수 있음

- **이 문제를 해결하기 위해 DMA(direct memory access)가 사용됨.**

I/O structure

- DMA(direct memory access)
  - 장치에 대한 버퍼, 포인터 및 카운터를 설정한 후, Device controller가 CPU의 개입 없이 전체 데이터 블록을 장치 및 주 메모리로 직접 전송.
  - 블록당 하나의 인터럽트만 생성

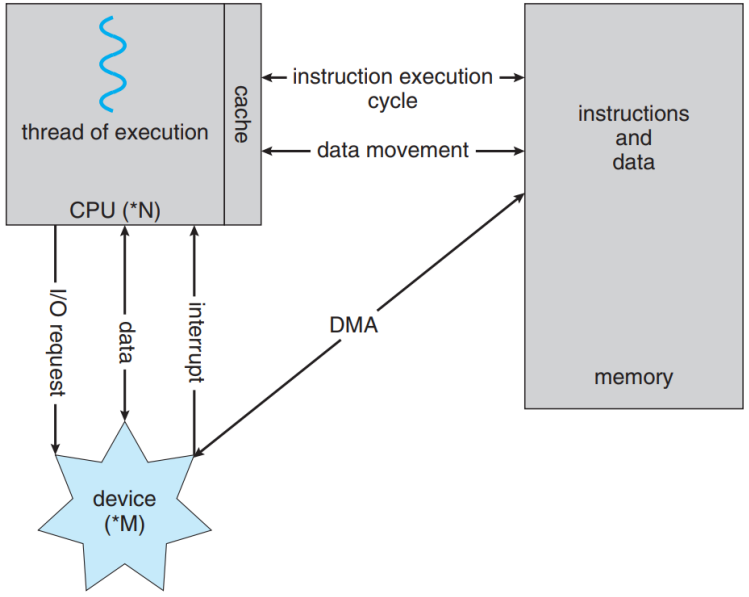
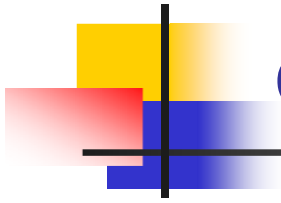


Figure 1.7 How a modern computer system works.

Computer-Systems Architecture

Computer systems architecture

- 구성요소 정의
  - CPU - 명령을 실행하는 하드웨어.
  - 프로세서 - 하나 이상의 CPU가 포함된 물리적 칩
  - 코어 – CPU의 기본 계산 단위
    명령을 실행하고 데이터를 로컬에 저장하기 위한 레지스터



Computer systems architecture

- Single-Processor Systems
  - 단일 코어 CPU가 하나만 있는 경우
  - 과거에 사용되었던 시스템
  - 그러나 현재의 컴퓨터 시스템이 단일 프로세서 시스템인 경우는 거의 없음.

Computer systems architecture

- Multiprocessor Systems
  - CPU가 2개 이상인 경우
  - 프로세서 수를 늘리면 더 짧은 시간에 더 많은 작업을 수행할 수 있음.
  - N개의 CPU가 있는 경우 N개의 프로세스가 실행될 수 있음
    - 그러나 N개의 프로세서의 성능 향상이 N배가 되는 것은 아님 그보다 작음
    - 여러 프로세서를 사용할 때 일정량의 오버헤드가 발생하고, 공유 리소스에 대한 프로세스 끼리의 경쟁이 일어나기 때문임
  - CPU 하나는 유휴 상태에 있고 다른 것은 과부하 상태일 수 있어 비효율적일 수 있음
    - 이를 위한 관리가 필수적임

Computer systems architecture

- Multiprocessor Systems

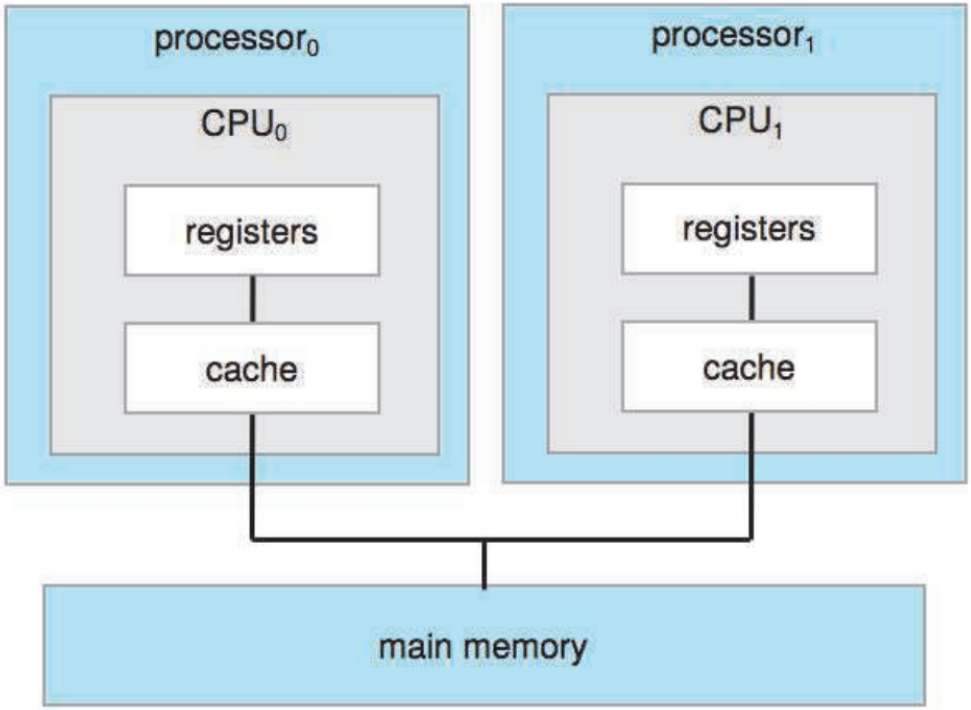


Figure 1.8 Symmetric multiprocessing architecture.

Computer systems architecture

- Multiprocessor Systems
  - 멀티프로세서의 정의는 시간이 지남에 따라 발전
  - 요즘은 멀티 프로세서의 정의에 단일 칩에 여러 컴퓨팅 코어가 있는 Multicore system이 포함됨.

- Multicore Systems
  - on-chip 통신이 between-chip 통신보다 빠르기 때문에, 멀티코어 시스템은 단일 코어가 있는 여러 프로세서보다 더 효율적임.
  - 여러 개의 코어가 있는 하나의 프로세서는 여러 개의 단일 코어 칩보다 훨씬 적은 전력을 사용

Computer systems architecture

- Multicore Systems

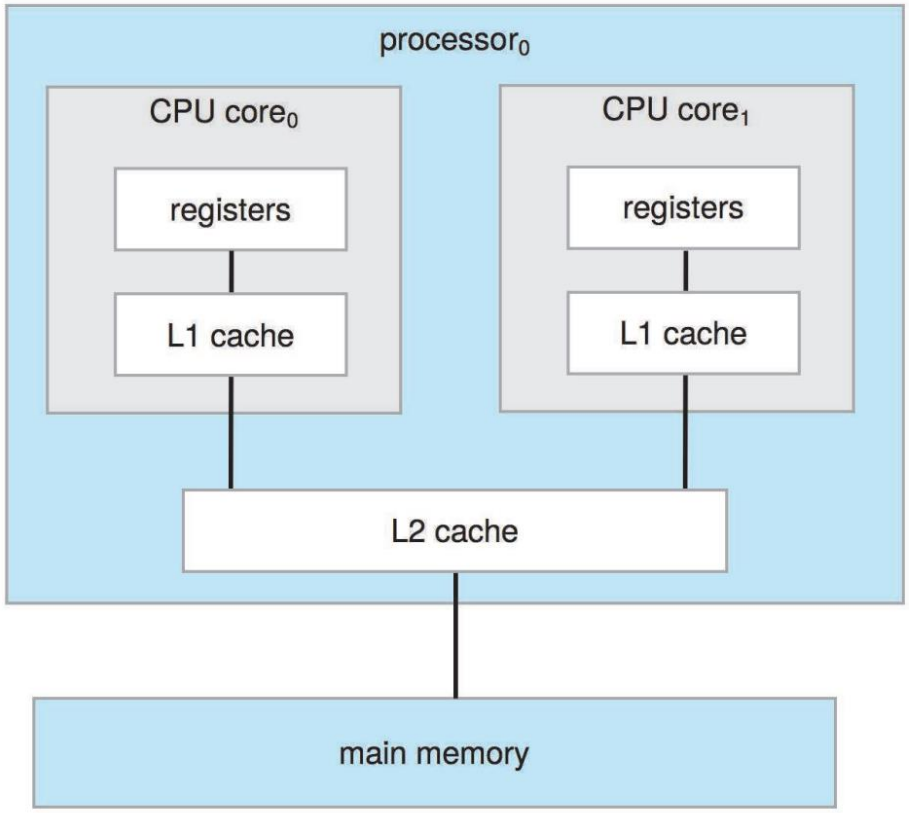
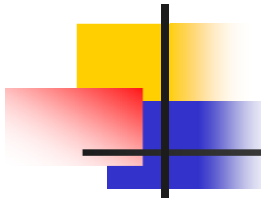


Figure 1.9 A dual-core design with two cores on the same chip.



Computer systems architecture

- Multiprocessor Systems은 두 가지 유형으로 구분됨
  - Asymmetric multiprocessing
    - 하나의 프로세서가 master, 나머지 프로세서는 slave
    - Master가 slave를 제어함.
    - 마스터 프로세서에서만 OS가 수행되어, slave 프로세서에 프로세스를 할당하거나, 데이터 및 I/O 관리를 수행.
  - Symmetric multiprocessing
    - 프로세서는 모두 평등함.
    - 각 프로세서는 메모리를 공유하고 서로 직접 상호 작용

Computer systems architecture

- Multiprocessor Systems
  - 여러개의 컴퓨터 시스템이 네트워크로 연결되어 있는 구조
  - 고성능 컴퓨팅 환경 제공
  - 단점: 유지보수의 어려움, 성능이 네트워크 환경에 매우 영향을 받음

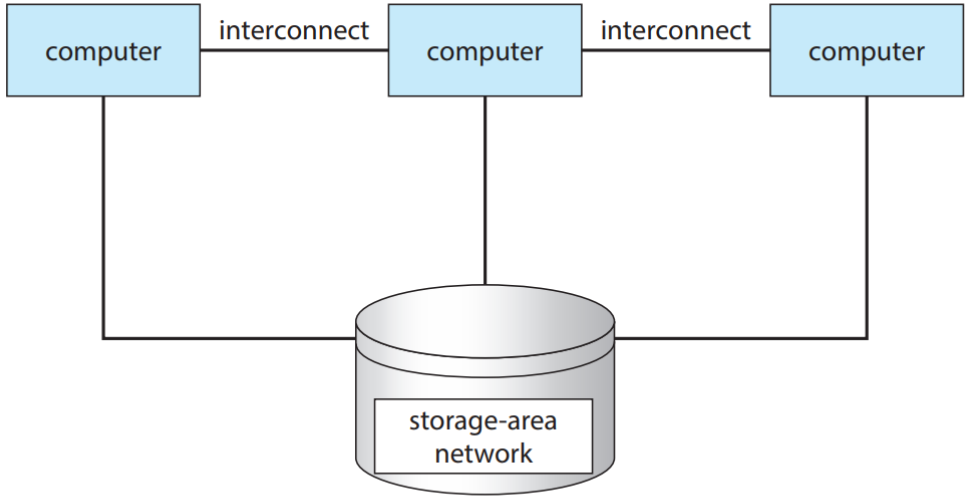


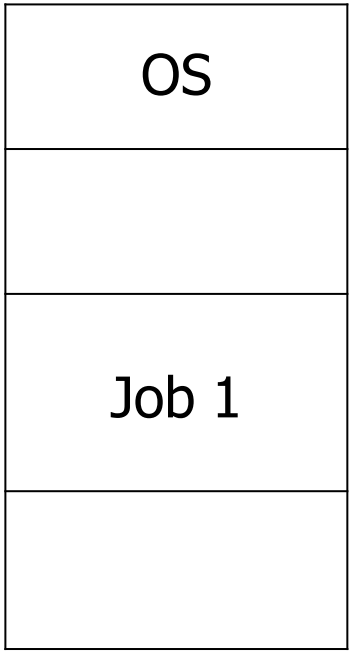
Figure 1.11 General structure of a clustered system.

Operating Systems Structure

Operating systems structure

- Multiprogramming
  - 여러 프로그램을 메모리에 로드해 둠.
  - 하나의 프로그램을 선택하여 실행
  - 한 프로세스가 대기 상태가 되면 다른 프로세스를 수행

  - CPU의 사용 효율을 높일 수 있음



Single-programming
(MS-DOS)

Multiprogramming

Operating systems structure

- Multitasking
  - 이는 프로세스마다 작업 시간을 정해두고 번갈아가면서 작업하는 방식
  - 프로세스들이 빠르게 번갈아가며 메모리를 사용하면 사용자 입장에서는 마치 동시에 작동하는 것처럼 보이게 됨
  - 반응 시간(Response time)을 줄이는 것이 중요
  - Time sharing이라고도 부름.

  - 예) 키보드 입력 시 : 타이핑 input이 컴퓨터에게는 너무 느리므로, OS는 CPU를 유휴 상태로 두는 대신 CPU를 다른 프로세스의 작업으로 전환

Operating systems structure

- 여러 프로세스가 동시에 실행될 준비가 되면 시스템은 다음에 실행할 프로세스를 선택해야 함
  - -> CPU 스케줄링

- 동시에 메모리에 여러 프로세스를 보유해야함
  - -> 메모리 관리

- 그 외에도 OS에서 중요하게 다뤄야할 기능은 더 많으나,
  이 강의에서는 비전공자 대상임을 고려하여, 우분투 사용 실습도 같이 진행할 예정이므로 이 두 파트 위주로 수업을 진행할 예정임.

Chapter 1-2
Finish