# 2️⃣ 컴퓨터시스템의 구조

| 태그 | 1주차 | CSS |
| --- | --- | --- |

## ✨ 운영체제

### 📌 의미

- 컴퓨터 하드웨어 바로 위에 설치되는 소프트웨어 계층
- 좁은 의미: 커널
  - 메모리의 상주하는 부분
- 넓은 의미
  - 각종 주변 유틸리티(파일 복사 등)를 포함한 개념

### 📌 목적

- 컴퓨터 시스템을 편리하게 사용할 수 있도록
- 하드웨어/소프트웨어 자원을 효율적으로 관리하는 (최대한 성능)
- 통치자, 프로그램에 맞게 메모리를 분배 → 컴퓨터를 훨씬 효율적으로 만듦

### 📌 분류

#### ✅ 동시 작업 가능 여부

- 단일 작업
  - 예) MS-DOS(초창지 OS) 프롬프트 상 명령 수행 끝나기 전 다른 명령 수행 X
- 다중 작업
  - 예) UNIX, Windows 등에서는 한 명령이 끝나기 전에 다른 명령 수행 가능

#### ✅ 사용자의 수

- 단일 사용자(single user)
  - MS-DOS, Windows
- 다중 사용자(multi user)
  - 한 대의 서버에 여러 명이 접속해서 쓸 수 있음
  - Linux, NT server

#### ✅ 처리 방식

- 일괄 처리(batch processing)
  - 일정량 모아서 한꺼번에 처리
  - 작업이 완전 종료될 때까지 기다려야 함
  - 예) 초기 Punch Card(작업 카드) 처리 시스템
- 시분할(time sharing) → 우리가 사용하는 OS
  - 여러 작업을 수행할 때 일정 시간 단위로 분할하여 사용
  - 일괄 처리에 비해 짧은 응답 시간(UNIX)
  - interactive ↔ 일괄 처리
  - 현대 범용 OS: best effort, 노력은 하지만 보장은 못해준다.
- 실시간(realtime)
  - 데드라인을 만족해야하는 시스템

        - 우리 OS는 안 그래도 됨
    - Hard realtime system 예) 원자로/공장, 미사일, 로보트, 반도체 장비 제어 등
    - Soft realtime system 예) 실시간 동영상 스트리밍

### 📌 컴퓨터에서 여러 작업을 동시에 수행하는 것을 뜻하는 몇 가지 용어

- multitasking
- multiprogramming: 메모리의 여러 프로그램이 올라가 있음을 강조
- time sharing: cpu 시간 분할하는 것을 강조
- multiprocess

— — — — — — — — — — — — — — — — — — — — — — — — — — — — — —

- multiprocessor → 컴퓨터에 cpu가 여러 개
- 우리 수업은 cpu가 한 개

### 📌 예

#### ✅ UNIX

- 대부분 C언어로 작성 (그 전 어셈블리어)
- 높은 이식성, 확장 용이
- 오픈 소스

#### ✅ DOS

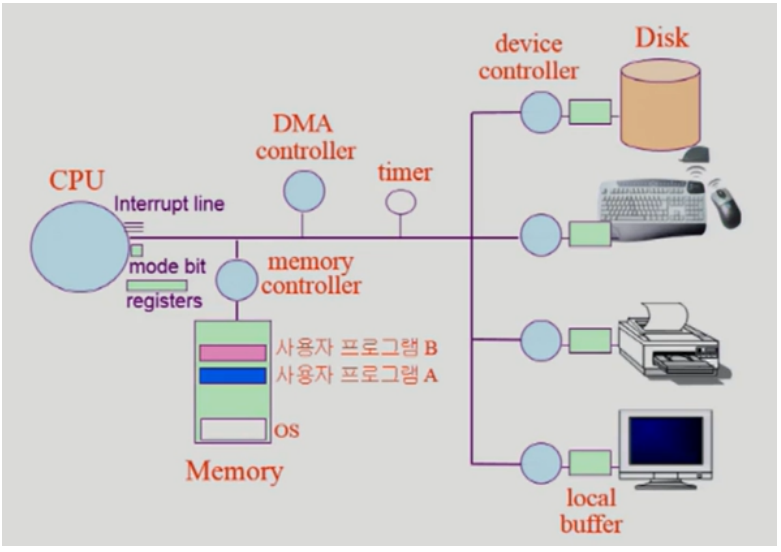
- MS
- 단일 사용자, 프로그램 하나

#### ✅ Windows

- MS
- 여러 프로그램

## ✨ 컴퓨터 시스템 구조

### 📌 사진




- 전원을 켜면 OS 메모리에 항상 상주

- 프로그램을 실행시키면(마우스 더블 클릭 등) 메모리에 올라가 프로세스가 됨
- controller: 작은 cpu, hw
- 메모리는 cpu의 작업 공간, 메모리에는 기계어
- cpu가 사용자 프로그램으로 넘어가면 os가 할 수 있는 것은 없음

## 📌 Mode bit

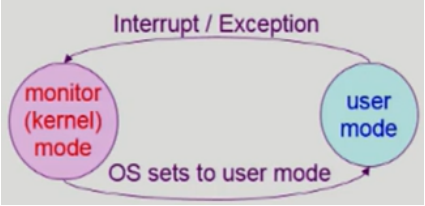
### ✅ 정의

- 사용자 프로그램으로부터 다른 프로그램이나 OS에 피해가 가지 않도록 보호하는 장치
- 기계어를 시작하기 전에 cpu가 mode bit을 보고 일

### ✅ 모드

- 1 사용자 모드: 사용자 프로그램 수행
  - 위험한 일은 하지 못하게, 제한된 기계어만
- 0 모니터 모드, 커널 모드, 시스템 모드: OS 코드 수행
  - 특권 명령: 보안을 해칠 수 있는 중요한 명령어는 0일 때만, 모든 I/O

### ✅ 넘어가는 방법




- 프로그램A 실행하다가 디스크에서 파일 읽어와야 하면 disk controller에게 일을 시킴
- disk controller가 일을 다 하면 cpu에게 **interrupt**를 검
- 운영체제에게 알리면 메모리에 해당 프로그램을 올림

## 📌 Registers 중 Program Counter

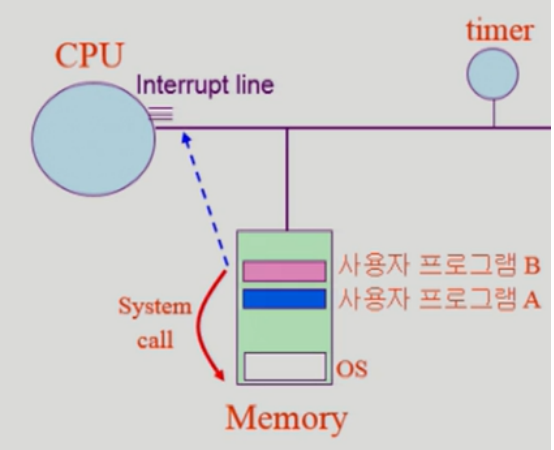
- 다음에 시작할 기계어의 메모리의 주소를 가지고 있음
- cpu가 pc를 참조

## 📌 timer

- cpu를 특정 프로그램이 독점하는 것을 막기 위한 장치
- 일정 시간 간격으로 인터럽트 발생시킴
- os가 cpu에 넘길 때 타이머와 함께 넘김
- 타이머 값이 0이 되면 cpu 제어권이 os에 넘어옴

## 📌 System Call

- 사용자 프로그램이 가진 권한으로 못하는 특권 명령의 기계어를 실행하고 싶을 때 운영체제에게 부탁
- 프로그램이 자신의 코드(SW)를 통해서 스스로 인터럽트를 검




## 📌 Interrupt

- 하드웨어가 거는 것
- Trap(sw interrupt)
  - Exception: 프로그램 오류
  - System call
- 인터럽트 벡터: 인터럽트 처리 루틴 주소
- 인터럽트 처리 루틴(인터럽트 핸들러): 인터럽트를 처리하는 커널 함수
- 현대 운영체제는 인터럽트가 들어올 때만 일한다. 구동된다.

## 📌 Device Controller

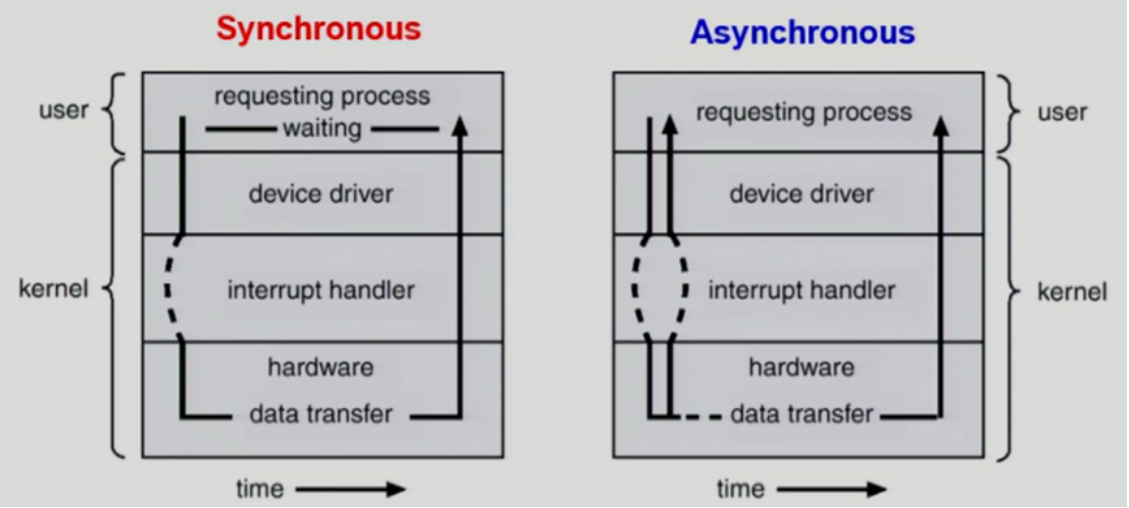
- I/O 장치를 관리하는 작은 cpu, hw
- 구분 → device driver: sw, 부탁을 하는 방법

## 📌 운영체제한테 CPU가 넘어가는 경우

1. I/O interrupt
2. timer interrupt
3. 사용자 프로그램이 권한 명령 코드를 실행해야 할 때(System call, trap)
4. Exception
5. cpu를 쓸 의지가 없는 경우, i/o가 오래 걸릴 것 같은 경우 ?

→ 결국 interrupt

# ✨ 입출력




## 📌 동기식 입출력 synchronous I/O

### ✅ 개념과 예시

- 싱크로나이즈드 스위밍, 립싱크, 시간 동기화
- 일반적
- 파일 읽는 건 읽어졌는지 확인하고 다음 작업

## 📌 비동기식 입출력 asynchronous I/O

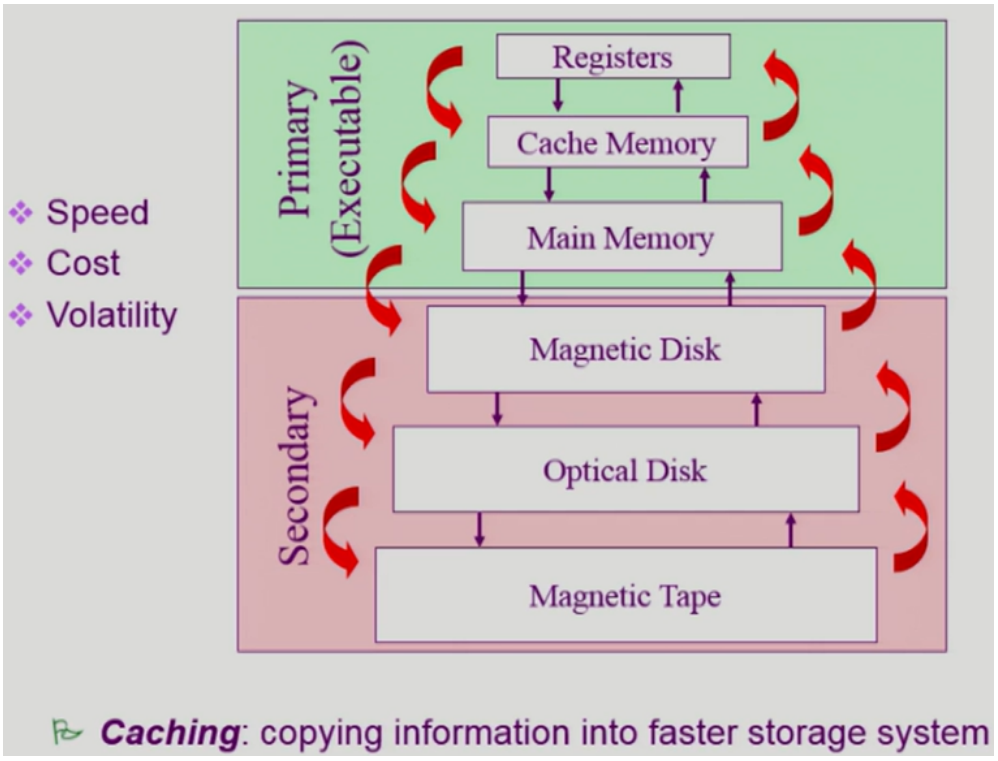
### ✅ 개념과 예시

- 파일에 써라는 요청, 쓰라고 하고 다음  할 수도 있음

## 📌 DMA (Direct Memory Access)

- 인터럽트도 일종의 오버헤드, 자주 걸리면 cpu한테도 비효율적
- 디바이스의 buffer storage의 내용을 직접 메모리에 일을 하고 block 단위(byte 단위 x)로 인터럽트를 걸어서 일이 끝났다고 알려줌
- 메모리에 접근할 수 있는 장치를 더 두는 것
- 덜 빈번하게 인터럽트를 발생시킬 수 있음

## 📌 저장장치 계층 구조




- caching: 재사용성 높임