1. Introduction to Operating Systems

2. System Structure & Program Execution

2023.06.15 **김하림 신석철 임수형**

1. Introduction to Operating Systems

01 컴퓨터 시스템

02 운영체제란?

03 운영체제의 종류

01 컴퓨터 시스템

컴퓨터 시스템의 요소는 4가지가 있다.

1. 하드웨어 - CPU, Memory, I/O device : 기본 컴퓨팅 자원

2. 운영체제 - 하드웨어 제어, 응용프로그램간의 컴퓨팅 자원 사용 조정.

3. 어플리케이션 - word processors, spreadsheets, compilters :
   자원으로 compting problems를 해결하는 방법이 정의된 프로그램.

4. 사용자

운영체제(Operating System)는 컴퓨터의 하드웨어를 관리하고,
하드웨어와 소프트웨어, 사용자 매개하는 프로그램이다.

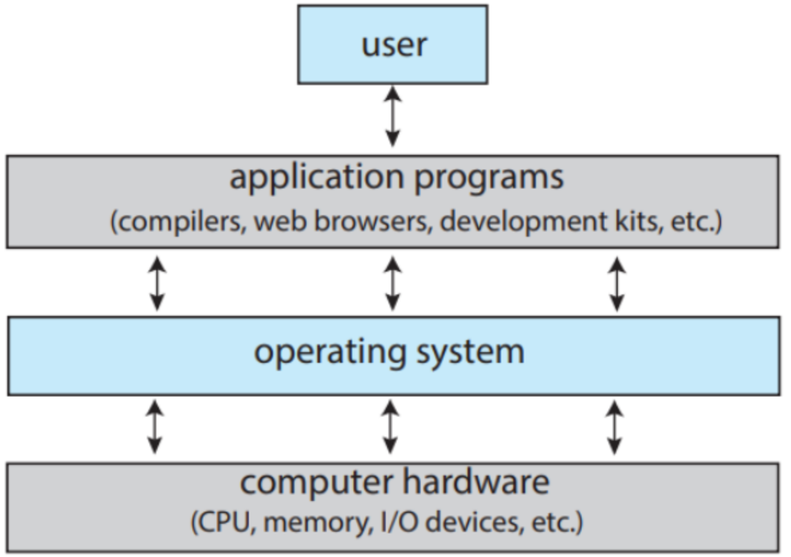


Figure 1.1 Abstract view of the components of a computer system.

02 운영체제의 목적, 정의

- 운영체제의 목적

1. 컴퓨터 시스템의 한정된 자원을 **효율성, 형평성**을 고려하여 관리.
   -> Resource Allocator

2. 컴퓨터 시스템을 편리하게 사용할 수 있는 환경 제공.
   -> ease of use

- 운영체제의 정의

운영체제에 완벽한 정의는 없다. 하지만 일반적으로 **운영체제는 컴퓨터에서 항상 실행되는 프로그램(kernel)**이다.

프로그램은 다음과 같이 구분할 수 있다.
1. 커널
2. 시스템 프로그램(유틸리티 : 파일관리, 프로그램 적재, 통신 ...)
3. 응용 프로그램

보다 넓은 의미로, 운영체제는 다음 3가지를 포함한다.

커널,
프로그램 개발을 용이하게 하고 기능을 제공하는 미들웨어 프레임워크,
실행 중인 시스템을 관리하는 데 도움이 되는 시스템 프로그램

미들웨어
응용 프로그램과 운영 체제 또는 다른 응용 프로그램 간의 상호 작용을 돕는 소프트웨어 계층 미들웨어는 서로 다른 시스템, 애플리케이션 또는 서비스 간의 통신, 데이터 교환, 데이터 변환 등을 관리.

시스템 프로그램
컴퓨터 시스템의 핵심 기능을 제공하기 위해 설계된 소프트웨어. 주로 운영 체제와 관련된 작업을 수행하며, 시스템 자원 (예: 메모리, 파일 시스템, 입출력 장치 등)을 관리하고 제어.

03 운영체제의 종류

1. 동시작업 여부
single tasking : MS-DOS 프롬프트. 한 명령수행을 끝내기 전에 다른 명령 수행 불가.
multi tasking : UNIX, MS Windows

2. 사용자의 수
single user : MS-DOS, MS Windows
multi user : UNIX, NT server

3. 처리 방식
batch processing : 작업 요청의 일정량 모아서 한꺼번에 처리 작업이 완전 종료될 때까지 대기
time sharing :
  여러 작업을 수행할 때 컴퓨터 처리 능력을 일정한 시간 단위로 분할. batch에 비해 짧은 응답 시간
  Interactive한 방식 (유저가 느끼기에 interactive) 쿨럭/시간 에 비해서 사람의 반응속도
Realtime OS:
  정해진 시간 안에 반드시 종료됨이 보장.
  원자로/공장, 미사일, 반도체 장비 ..
  Hard/Soft Realtime system

대표적으로 Unix의 특징은 다음과 같다.
멀티 태스킹, 다중사용자 방식, 시분할 운영체제

2. System Structure & Program Execution

01 Computer System Structure

02 Interrupt

03 Mode Bit

04 Timer

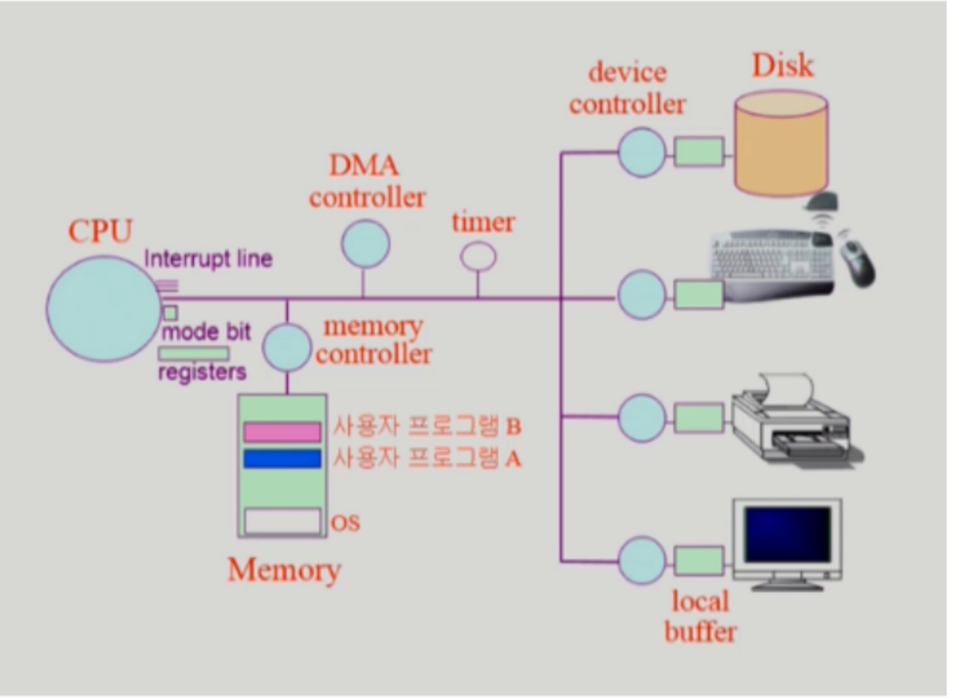
05 I/O Device Controller & I/O Execution

06 System Call

01 Computer System Structure

1. **CPU**: 매 clock cycle마다 memory에서 instruction을 읽어 실행
   - **mode bit**: kernel mode / user mode
   - **interrupt line**: instruction 한 줄 수행 후 interrupt line을 확인해 들어온 interrupt가 있는지 확인
   - **registers**: memory보다 빠르면서 정보를 저장할 수 있는 작은 공간
2. **Memory**: CPU의 작업 공간
3. **I/O Device**: disk, keyboard, ...
   - 각각 **device controller**와 **local buffer**가 존재
   - device controller: I/O Device의 작은 CPU 역할, 작업 완료 후 CPU에 interrupt
   - local buffer: device controller의 작업 공간
4. **timer**: 특정 프로그램이 CPU를 독점하는 것을 막음

Computer System Structure



02 Interrupt

interrupt 당한 시점의 register와 program counter(PC)를 save한 후 CPU의 제어를 interrupt handler에 넘긴다

Interrupt(hardware interrupt): 하드웨어(키보드, 하드디스크..)가 발생시킨 interrupt

Trap(software interrupt): Exception, System call(user program이 커널 함수 호출)

interrupt 관련 용어

- **interrupt vector**: 해당 interrupt handler의 주소를 가지고 있음
- **interrupt handler** = interrupt service routine: 해당 interrupt를 처리하는 kernel 함수

03 Mode bit

1. user program의 잘못된 수행으로 다른 프로그램 및 운영체제에 피해가 가지 않도록 하기 위한 보호 장치
2. mode bit = 0: **Kernel mode** - previleged instruction 수행
    - Interrupt나 Exception 발생 시 하드웨어가 mode bit을 0으로 바꿈
    - user program에게 CPU를 넘기기 전에 mode bit을 1로 set
3. mode bit = 1: **User mode** - 제한된 instruction 수행

04 Timer

- 정해진 시간이 흐른 뒤 OS에게 제어권이 넘어가도록 interrupt를 발생시킴
- 타이머 값이 매 clock 1씩 감소하다가 0이 되면 timer interrupt 발생
- **특정 프로그램이 CPU를 독점하는 것을 막음**
- **time sharing** 구현, 현재 시간 계산 등을 위해 사용

05 I/O Device Controller

- 해당 I/O device를 관리하는 일종의 작은 CPU
- control register, status register
- local buffer = 일종의 data register
- I/O는 실제 device와 local buffer 사이에서 일어남
- I/O가 끝났을 경우 interrupt로 CPU에 알림**(hardware interrupt)**
- **device driver**: OS 코드 중 각 장치별 처리 루틴(handler) → Software
- **device controller**: 각 장치를 통제하는 작은 CPU → Hardware

05 I/O Execution

- user program은 어떻게 I/O 을 실행하는지?

1. **system call**: OS에게 I/O request를 보냄(Trap - software interrupt)

2. **trap**을 사용하여 interrupt vector의 특정 위치로 이동

3. 제어권이 interrupt vector가 가리키는 interrupt handler로 이동

4. 올바른 I/O request인지 확인 후 I/O 진행

5. I/O 완료 시 제어권을 system call 다음 instruction으로 넘김

06 System Call

- user program이 OS의 서비스를 받기 위해 커널 함수를 호출
- CPU 제어권이 OS에게 넘어감

07 동기식 입출력

- 운영체제(커널)에 I/O 요청이 들어오면 입출력 작업이 완료된 후에야 CPU 제어권이 유저 프로그램에 넘어감

구현 방법 1

I/O가 끝날 때까지 프로세스가 CPU를 가지고 있는 방법 (CPU 낭비)

1. I/O가 끝날 때까지 CPU를 낭비시킴
2. 매시점 하나의 I/O만 일어날 수 있음

구현 방법 2

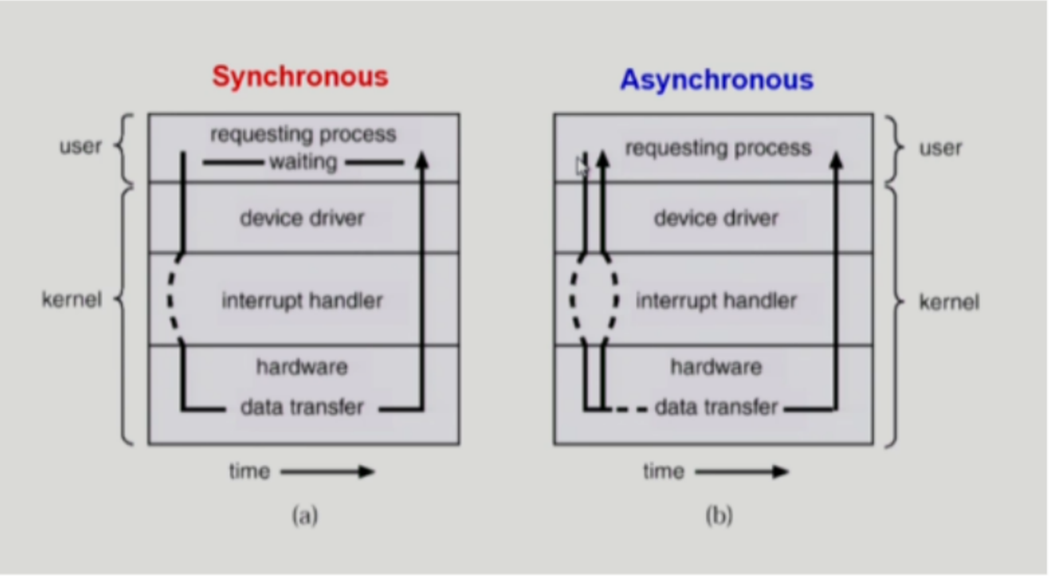
I/O가 완료될 때까지 다른 프로그램에게 CPU를 주는 방법

작업중에 오래걸리는 작업은 뒤로 미뤄두고 먼저 끝나는 작업부터 처리하는 방법

1. I/O가 완료될 때까지 해당 프로그램에게서 CPU를 빼앗음
2. I/O 처리를 기다리는 줄에 그 프로그램
3. 다른 프로그램에게 CPU를 줌

08 비동기식 입출력

- 운영체제(커널)에 I/O 요청만 해놓고 사용자 프로그램이 CPU 제어권을 즉시 돌려받아 다른 작업을 수행함
- 입출력 작업이 끝날때까지 기다리지 않음 (비순차적)
- (동기식, 비동기식) 입출력 모두 I/O의 완료는 인터럽트로 알려줌



09 Memory Mapped I/O

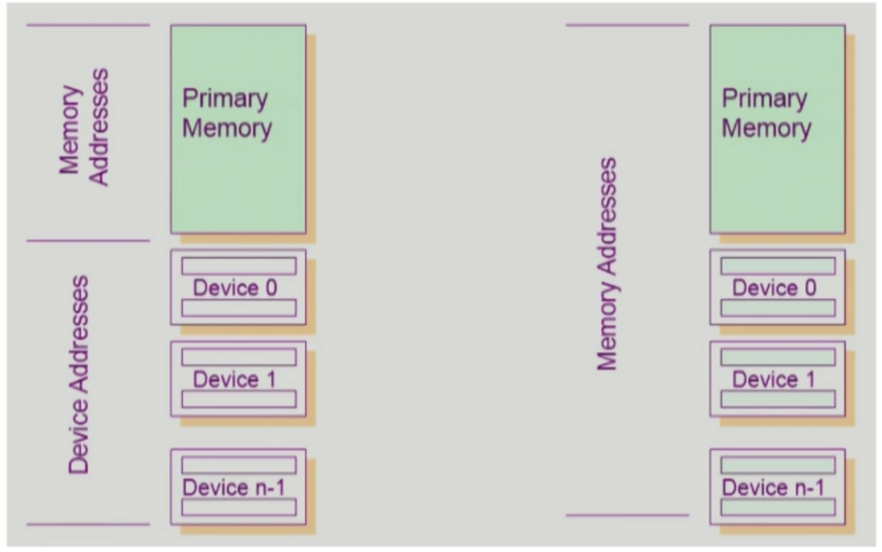
- (좌) 일반적인 I/O 처리 방법

메모리상에 I/O 처리를 위한 별도의 instruction이 존재. 해당 instruction을 실행하여 I/O 처리
일반적으로 입출력 장치와 메모리 간에 데이터를 주고받기 위해 별도의 명령어나 포트를 사용

- (우) Memory Mapped I/O

I/O 처리를 위한 instruction을 메모리주소의 연장선상에 놓는 방법

메모리 맵드 I/O는 입출력 장치를 다루는 소프트웨어 코드를 단순화하고 다른 메모리 위치와 구별할 필요 없이 일관된 방식으로 접근할 수 있음



10 저장장치 계층 구조

- primary 매체

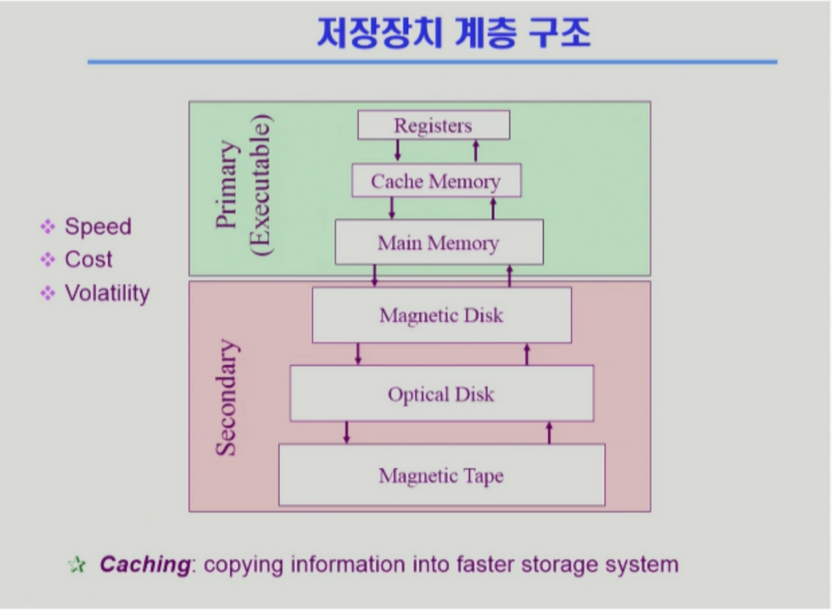
CPU에서 직접 접근. CPU가 byte 단위로 읽고 실행할 수 있는 경우를 말함
휘발성 매체이기 때문에 전원이 꺼지면 저장된 데이터가 모두 지워짐

- secondary 매체

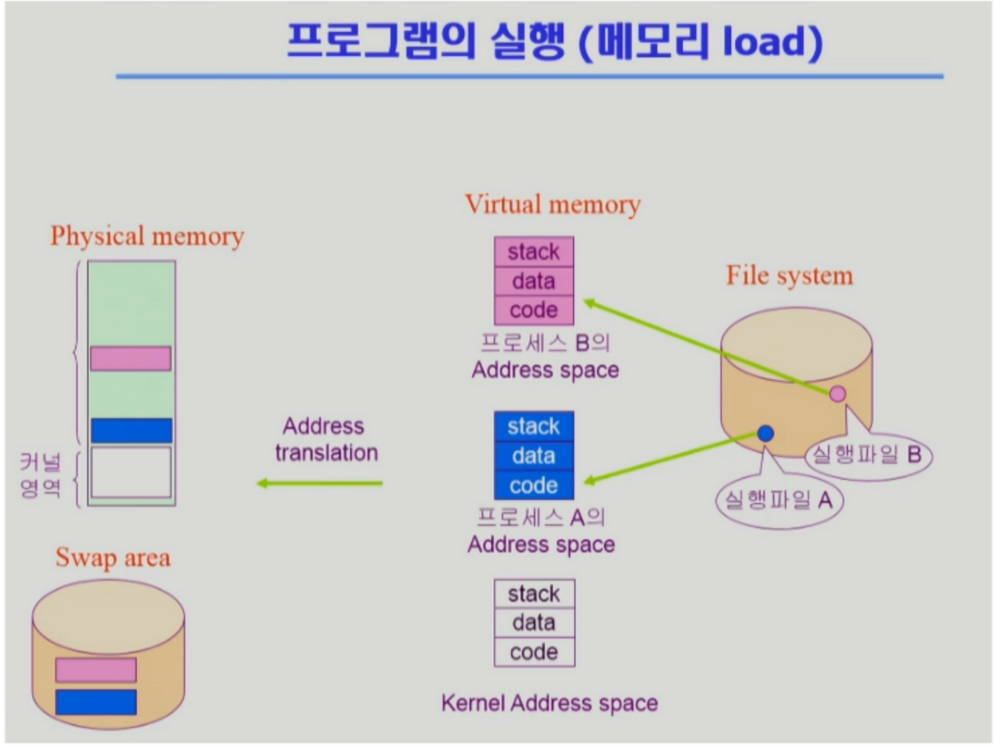
CPU에서 직접 접근하여 실행할 수 없음. 비휘발성 매체여서 전원이 꺼져도 데이터가 남음

- Caching(캐싱)

secondary 매체로부터 primary 매체인 Cache memory에 데이터를 복사해오는 것
데이터의 재사용을 목적



11 메모리 주소



- Virtual memory

프로그램을 실행하는 시점에 해당 프로그램만의 독자적인 Address space(메모리상의 주소 공간)를 생성하는 것
- stack — 함수를 호출하거나 리턴할때 사용하는 영역
- data — 변수 등 프로그램이 사용하는 데이터들을 담고 있는 영역
- code — CPU에서 실행할 기계어 코드를 담고 있는 영역

- Physical memory

Address Space 중에서 당장 필요한 부분이 물리적 메모리에 남음
당장 필요하지 않은 것은 swap area로 내려감

- Kernal Address Space

- code — CPU 들이 수행할 기계어들이 모인 부분. 실행 파일에서의 코드들이 올라옴
- data — 전역 변수, 정적 변수, 상수 등의 데이터가 저장
- stack — 함수 호출 및 임시 데이터 저장에 사용

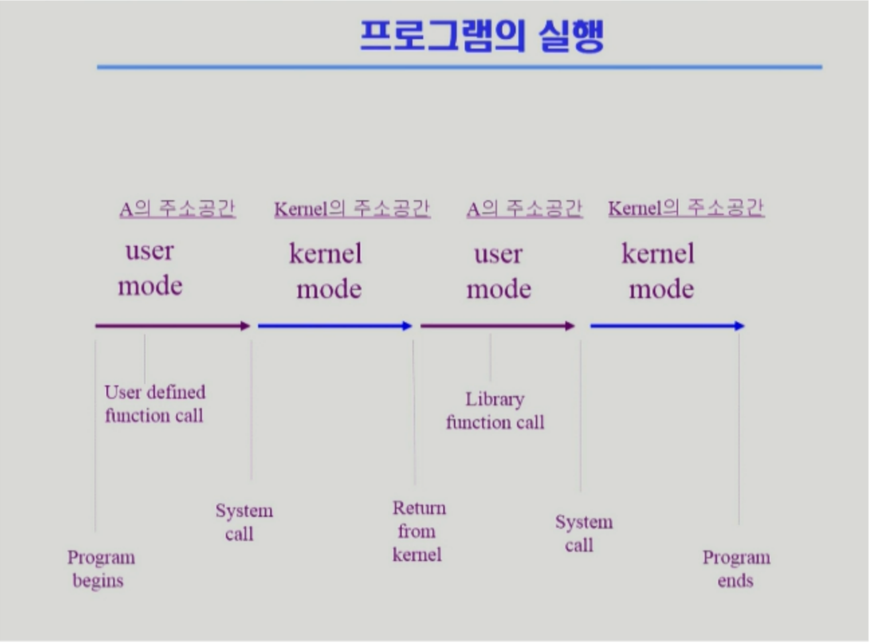
12 유저 프로그램이 사용하는 함수

- 사용자 정의 함수 : 내가 프로그램에 정의한 함수
- 라이브러리 함수 : 다른사람이 만들어 놓은 함수이지만 내 프로그램의 실행 파일에 포함되어 있는 함수
- 커널 함수 : 운영체제 프로그램의 함수, 커널 함수의 호출 = System call

사용자 정의 함수든 라이브러리 함수든 컴파일해서 실행파일을 만들게 되면 내 프로그램 안에 들어있는 함수이기 때문에 언제든 자유롭게 실행할 수 있음

반면 커널함수의 경우 내 프로그램의 함수가 아니라 커널코드에 포함된 함수이기 때문에 시스템 콜을 통해서 CPU 제어권을 넘겨야만 실행이 가능

13 프로그램 실행 단계



하나의 프로그램이 실행되는 동안 운영체제는 user mode ↔ kernel mode 전환을 반복

프로그램이 CPU 제어권을 가지고 있으면 user mode
이때 syetem call을 하게 되면 CPU 제어권이 커널로 넘어가면서 kernel mode로 전환하게 되고 커널의 주소 공간에 있는 커널함수를 실행