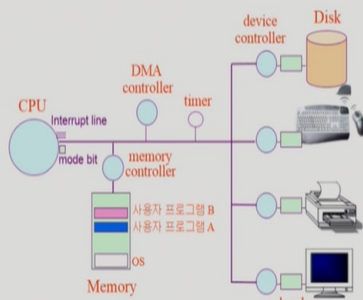


## 컴퓨터 시스템 구조



CPU : 4byte 짜리 처리. 메모리에 있는 가변인자 프로그램(비트스트림) (CPU 내부 있음.)  
가변인자를 읽어들인다. 4byte 증가시켜서

기계가 켜, 정지하는 인스트럭션은 인스턴트 Jump (if)  
Interrupt (line이 끊어지면, 그 주입수신시각 (interrupt vector) → 몇  
어떻게 행동할 지는 커널함수 결정되어 있다

mode bit : 0 : 모든 instruction 실행가능. (이때만 I/O 가능)  
1 : 한정된 instruction만 실행.

## 인터럽트 (Interrupt)

- 인터럽트
  - 인터럽트 당한 시점의 레지스터와 program counter를 save 한 후 CPU의 제어를 인터럽트 처리 루틴에 넘긴다
- Interrupt (넓은 의미)
  - Interrupt (하드웨어 인터럽트): 하드웨어가 발생시킨 인터럽트
  - Trap (소프트웨어 인터럽트)
    - Exception: 프로그램이 오류를 범한 경우
    - System call: 프로그램이 커널 함수를 호출하는 경우
- 인터럽트 관련 용어
  - 인터럽트 벡터
    - 해당 인터럽트의 처리 루틴 주소를 가지고 있음
  - 인터럽트 처리 루틴 (=Interrupt Service Routine, 인터럽트 핸들러)
    - 해당 인터럽트를 처리하는 커널 함수

Exception: mode bit이 1일 때도 이걸 넘겨줘야 하지만  
잘못된 행동을 하면 발생하는 interrupt.

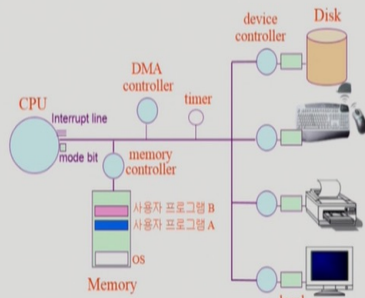
## 입출력(I/O)의 수행

- 모든 입출력 명령은 특권 명령
- 사용자 프로그램은 어떻게 I/O를 하는가?
  - 시스템콜(system call)
    - 사용자 프로그램은 운영체제에게 I/O 요청
  - trap을 사용하여 인터럽트 벡터의 특정 위치로 이동
  - 제어권이 인터럽트 벡터가 가리키는 인터럽트 서비스 루틴으로 이동
  - 올바른 I/O 요청인지 확인 후 I/O 수행
  - I/O 완료 시 제어권을 시스템콜 다음 명령으로 옮김

사용자 프로그램이 직접 입출력할 수 없으므로 system call.

의도적으로 interrupt line 끊어져서 CPU 제어를 운영체제에 넘겨줌.

## 컴퓨터 시스템 구조



timer가 interrupt 에 카운트.

소프트웨어 계속 사용을 방지하기 위해 timer를 설정하여 CPU 재귀를 방지함

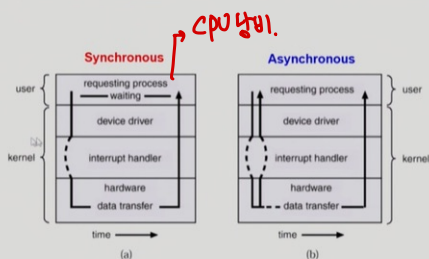
## 동기식 입출력과 비동기식 입출력

- 동기식 입출력 (synchronous I/O)
  - I/O 요청 후 입출력 작업이 완료된 후에야 재어가 사용자 프로그램에 넘어감
  - 구현 방법 1
    - I/O가 끝날 때까지 CPU를 낭비시킴
    - 매시점 하나의 I/O만 일어날 수 있음
  - 구현 방법 2
    - I/O가 완료될 때까지 해당 프로그램에게서 CPU를 빼앗음
    - I/O 처리를 기다리는 중에 그 프로그램을 줄 세움
    - 다른 프로그램에게 CPU를 줌
- 비동기식 입출력 (asynchronous I/O)
  - I/O가 시작된 후 입출력 작업이 끝나기를 기다리지 않고 재어가 사용자 프로그램에 즉시 넘어감

☆ 두 경우 모두 I/O의 완료는 인터럽트로 알려줌

동기식 입출력 :

## 동기식 입출력과 비동기식 입출력



Synchronous I/O : 요청된 I/O를 바탕으로 작업을 해야하는 경우.

Asynchronous I/O : 읽기든 데이터나 상관없이 작업 수행

↳ write는 보통 비동기 구현.

이화여자대학교  
EWHA WOMANS UNIVERSITY

### DMA (Direct Memory Access)

- DMA (Direct Memory Access)
  - 빠른 입출력 장치 메모리에 가까운 속도로 처리하기 위해 사용
  - CPU의 중재 없이 device controller가 device의 buffer storage의 내용을 메모리에 block 단위로 직접 전송
  - 바이트 단위가 아니라 block 단위로 인터럽트를 발생시킴

0:26:10 -31:15 1x 1080p

메모리 접근은 CPU만 가능..

but. DMA도 접근 가능하도록 설정

CPU만 접근할 경우

interrupt를 너무 자주 당하게 됨

이화여자대학교  
EWHA WOMANS UNIVERSITY

### 컴퓨터 시스템 구조

0:28:11 -29:14 1x 1080p

1byte 같은 작은 단위는 I/O 버퍼에 쌓은 후. 차면

1번 <sup>CPU가</sup> interrupt를 받음.

DMA가 작업하기, interrupt는 작업이 처리되었다고 알려줌.

이화여자대학교  
EWHA WOMANS UNIVERSITY

### 서로 다른 입출력 명령어

- I/O를 수행하는 special instruction에 의해
- Memory Mapped I/O에 의해

0:30:38 -26:47 1x 1080p

방법1. 메모리 접근하는 instruction이 있고,

I/O 접근하는 special instruction이 의해. I/O 처리 가능.

방법2 I/O 장치도 메모리 주소에 양력.

따라서 메모리 접근하면 I/O 실행.

이화여자대학교

YONSEI WOMANS UNIVERSITY

저장장치 계층 구조

CPU

Registers

Cache Memory

Main Memory

Magnetic Disk

Optical Disk

Magnetic Tape

Primary (Executable)

Secondary

속도, 용량, 비용, 휘발성

↑

↓

관련하면 비용 상승.

해결방법 비휘발성 메모리

이 부분은 변화 중.

★ Caching: copying information into faster storage system

재사용 목적.

0:33:23

↺

↻

↻

↻

-24:02

1x

1080p

⌵

Primary: CPU에서 작업 접근 시 처리 가능한 등. Executable (byte 단위) 접근 가능함

Hard disk: sector 단위로 읽기며 unExecutable!

CPU는 1코클당 1개의 Instruction을 처리하나, Main 메모리는 100코클 정도 소요. 그 속화이를 완충하기 위해 registers, Cache Memory 등.

이화여자대학교

YONSEI WOMANS UNIVERSITY

프로그램의 실행 (메모리 load)

Physical memory

File system

프로세스 B

프로세스 A

kernel

실행파일 B

실행파일 A

0:39:21

↺

↻

↻

↻

-18:04

1x

1080p

⌵

프로그램은 실행파일 형태로 하드디스크에 저장. 이를 실행하면 메모리로 올라가서 프로세스가 됨

이화여자대학교

YONSEI WOMANS UNIVERSITY

프로그램의 실행 (메모리 load)

Physical memory

Virtual memory

Swap area

stack

data

code

Address space

Kernel Address space

프로세스 B의 Address space

프로세스 A의 Address space

실행파일 B

실행파일 A

필요한 공간!

예전 2GB까지

커널 영역

가져 올릴 수 없음

키엔 영역 상부

(하드디스크)

전원 나기면 위키는 데이터.

메모리 용량의 한계로 담이두는 용도.

처리 후행부 자석이 설명

0:39:47

↺

↻

↻

↻

-17:38

1x

1080p

⌵

비로 메모리가 올라가는 것이 아니라 가상메모리로 넘어감 ⇒ 실제로 메모리에 올라간 단계는 아님.

독자적인 주소공간 형성. (code, data, stack)  
↓  
실행권. 자료의 양은 한

• 실행시 당장 필요한 코드만 올림.  
그 외는 Swap area에 내려놓는다.



이화여자대학교 EWHA WOMANS UNIVERSITY

## 커널 주소 공간의 내용

code: 커널 코드  
 o 시스템콜, 인터럽트 처리 코드  
 o 자원 관리를 위한 코드  
 o 편리한 서비스 제공을 위한 코드

data: Process A, Process B, PCB, PCB, Process Control Block, CPU, mem, disk

stack: Process A의 커널 스택, Process B의 커널 스택

Legend:  
 □ : Table (Data Structure)  
 ■ : Object (hardware or software)

0:46:09 -11:16 1x 1080p

인터럽트 코드: 인터럽트 왔을 때 어떻게 처리해야 하는가?  
 큰 작업이 되었을 때

이화여자대학교 EWHA WOMANS UNIVERSITY

## 사용자 프로그램이 사용하는 함수

- 함수(function)
  - 사용자 정의 함수
    - 자신의 프로그램에서 정의한 함수
  - 라이브러리 함수
    - 자신의 프로그램에서 정의하지 않고 갖다 쓴 함수
    - 자신의 프로그램의 실행 파일에 포함되어 있다
  - 커널 함수
    - 운영체제 프로그램의 함수
    - 커널 함수의 호출 = 시스템 콜

Handwritten notes:  
 라이브러리 함수: 컴파일된 코드를 실행파일에서 함수 호출  
 커널 함수: 시스템 콜 통해 커널로 들어갈 수 있음  
 커널 함수 호출: 커널 함수가 있는 곳 (호출할 때)

0:51:04 -6:21 1x 1080p

함수를 고수준 언어에서 사용하는 이유는...  
 모든 프로그램은 함수로 되어 있음.

컴파일러가 커널로 되더라도 함수를 필요로 함.

커널 함수 호출 이유.

- 메모리 주소를 프로그램 안에서 kernel로 점프하기  
 일반 Jump는 인스트럭션 블록과 다른 논리적인 주소로 점프  
 일반 Jump는 인스트럭션 블록과 다른 논리적인 주소로 점프
- 물리적 차원이며 사용자 프로그램이 운영체제 못 건드릴 때

이화여자대학교 EWHA WOMANS UNIVERSITY

## 프로그램의 실행

A가 CPU를 가지고 있는 환경에서 작성.  
 CPU 관리 미션 → A의 CPU를 가지고 실행을 → kernel 실행 → A 실행. → timer interrupt → kernel → B 실행 등...

Program begins → User mode → System call → kernel mode → Library function call → Return from kernel → user mode → Program ends

0:54:28 -2:57 1x 1080p

프로그램이 실행되는 단계.

A가 시작하여 종료될 때 까지 운용.