운영체제: 강의노트 02

A. Silberschatz, P.B. Galvin, G. Gagne
*Operating System Concepts*,
Sixth Edition, John Wiley & Sons, 2003.

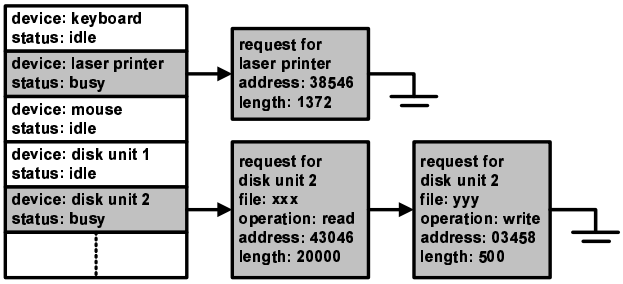
Part I. Overview

# 2 컴퓨터 시스템 구조

## 2.1 컴퓨터 시스템의 동작

- 현대의 범용 컴퓨터는 공유된 주기억장치에 접근을 제공하는 공통 버스에 의해 연결된 CPU와 여러 개의 장치 제어기(device controller)로 구성되어 있다.
  - 장치 제어기: 각 장치(디스크, 오디오 장치, 비디오 디스플레이)를 관리한다.
  - 장치 제어기와 CPU는 병행으로 수행되므로 이들은 주기억장치 접근에 대해 경쟁한다. 주기억장치 제어기는 이들의 접근을 동기화해준다.
- 컴퓨터가 처음 구동되면 초기에 실행될 프로그램이 필요하다. 이 프로그램을 **부트스트랩 프로그램(bootstrap program)**이라 한다. 이 프로그램은 보통 컴퓨터 하드웨어 내에 ROM(Read-Only Memory)에 저장되어 있다.
  - 부트스트랩 프로그램은 모든 하드웨어를 초기화하고 운영체제 커널을 주기억장치에 적재한 후에 커널을 실행한다.
- 컴퓨터에서 사건의 발생은 **인트럽트(interrupt)** 신호 또는 **트랩(trap)**을 통해 운영체제에 통보된다. 트랩은 다른 말로 예외(exception)라 한다.
  - 인트럽트는 하드웨어가 시스템의 수행 흐름을 바꾸기 위해 발생하는 것이고, 트랩은 소프트웨어가 발생하는 인트럽트이다.
  - 하드웨어는 CPU에 특정 신호를 보내어 인트럽트의 발생을 알린다.
  - 소프트웨어는 **시스템 호출(system call)**이라는 특정 연산을 실행하여 일부로 발생시키거나 오류(0 나누기, 부적합한 주기억장치 접근) 때문에 자발적으로 발생된다.
  - 인터럽트가 발생되면 CPU는 현재 수행중인 작업을 멈추고, 운영체제 내에 있는 특정 코드를 실행한다. 이 실행이 끝나면 다시 멈춘 작업을 재개한다.
  - 기본적인 인터럽트 동작원리

    (1) 현재 작업을 멈추고, 현재 상태를 보관




<그림 2.1> 장치 상태 테이블

    (2) 인터럽트의 종류 분석

    (3) 특정 인터럽트 수행(인터럽트 벡터 활용)

    (4) 보관된 상태를 원상복귀하고 멈춘 작업을 재개

## 2.2 I/O 구조

- 장치 제어기에 따라 하나 이상의 장치가 제어기에 연결될 수 있다.
- 장치 제어기는 지역 버퍼와 몇 개의 특수 목적 레지스터를 유지한다.
- 장치 제어기는 연결된 주변장치와 지역 버퍼 간에 데이터 이동을 책임진다. 이 버퍼의 크기는 주변장치에 따라 다르다.

### 2.2.1 I/O 인터럽트

- I/O을 시작하기 위해 CPU는 장치 제어기 내에 있는 레지스터에 적절한 데이터를 적재한다. 제어기는 이것을 검사하여 어떤 행동을 취할지 결정한다. 입출력의 수행이 끝나면 보통 인터럽트를 통해 CPU에 그 사실을 통보한다.
- 입출력의 두 가지 형태
  - 동기식 입출력(synchronous I/O): 입출력이 시작되면 요청한 프로세서는 입출력이 완료될 때까지 기다린다.
  - 비동기식 입출력(asynchronous I/O): 요청한 프로세서는 입출력이 완료될 때까지 기다리지 않고 계속 다른 작업을 수행한다.
- 입출력의 완료를 기다리는 방법
  - 특수한 명령어 사용
  - 대기 루프 사용
- 만약 CPU가 입출력 완료를 항상 기다리면 한번에 한 입출력만 가능하다. 이것은 바람직하지 않다. 여러 입출력을 병행으로 수행할 수 있고, 입출력과 계산을 병행할 수 있어야 시스템의 효율을 높일 수 있다.
- 운영체제는 여러 개의 입출력 요청을 관리하기 위해 그림 2.1과 같은 장치 상태 테이블(device-status table)을 유지한다. 또한 각 장치마다 대기 큐를 유지한다.

### 2.2.2 DMA 구조

- 속도가 느린 입출력 장치는 하나의 입력을 받은 후에 다음 입력까지 CPU는 다른 유용한 작업을 할 수 있다. 그러나 반대로 입출력 장치의 속도가 빠르면 인트럽트가 너무 빈번하게 발생하여 CPU가 다른 유용한 작업을 할 시간이 적다.
- 이것을 해결하기 위해 사용하는 기법이 DMA (Direct Memory Access)이다. DMA 방식에서 장치 제어기는 데이터 블록을 CPU의 관여없이 직접 주기억장치로 이동하며, 인터럽트는 바이트 단위가 아닌 블록 단위로 발생한다.

## 2.3 저장 구조

- 컴퓨터 프로그램이 실행되기 위해서는 주기억장치에 적재되어야 한다. 주기억장치를 다른 말로 임의접근 메모리(random-access memory)라 한다.
- CPU가 직접 접근할 수 있는 기억장치는 주기억장치뿐이다.
- 주기억장치는 보통 동적 임의접근 메모리(dynamic RAM)라고 하는 반도체 메모리를 사용한다.
- 주기억장치의 한 구성 단위를 워드(word)라 하며, 각 워드는 독특한 주소를 가진다.
- 이상적으로는 모든 프로그램과 데이터를 주기억장치에 영구적으로 저장하고 싶지만 다음 두 가지 이유 때문에 가능하지 않다.
  - 이유 1. 주기억장치의 크기
  - 이유 2. 주기억장치의 휘발성

  이 문제를 해결하기 위해 보조 기억장치를 사용한다. 보조 기억장치는 많은 양의 데이터를 영구 보관할 수 있어야 한다.
- 가장 널리 사용하는 보조 기억장치는 자기 디스크(magnetic disk)이다.

### 2.3.1 주기억장치

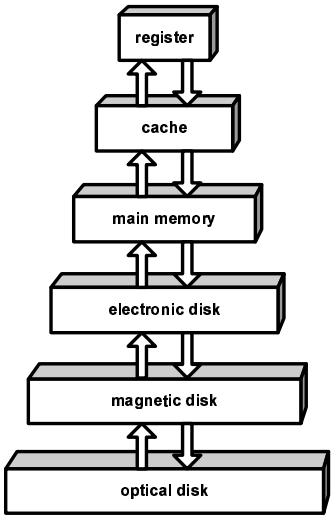
- Memory-mapped I/O: 주기억장치의 일부 주소가 입출력을 위해 예약되어 있으며, 이 주소에서 읽거나 쓰면 장치 레지스터로부터 데이터를 읽거나 쓰는 결과가 된다. 이 방법은 고속 응답시간을 요구하는 장치에 적합하다.
- 이 방법은 직렬포트 또는 병렬포트에 연결된 장치와 입출력할 때도 사용된다. CPU는 I/O 포트를 통해 이와 같은 장치와 데이터를 교환한다. 데이터 교환 방식은 다음과 같은 두 가지 방식이 있다.
  - 프로그램된 I/O(programmed I/O): CPU가 계속 장치의 상태를 검사(polling)하는 방식
  - 인터럽트 기반: 다음 데이터를 처리할 준비가 되면 장치 제어기는 인터럽트를 통해 그 사실을 CPU에 알리는 방식
- 주기억장치와 CPU의 속도 차이를 극복하기 위해 주기억장치와 CPU 사이에 캐시(cache)라고 하는 고속 메모리 버퍼를 사용한다.

### 2.3.2 자기 디스크

- 자기 디스크는 플래터(platter)라고 하는 여러 개의 원형 판으로 구성되어 있으며, 이 플래터는 다시 원형 모양의 트랙(track)으로 나뉘어진다. 트랙은 다시 여러 개의 섹터로 나뉘어진다.
- 같은 위치에 있는 트랙의 모음을 실린더(cylinder)라 한다.
- 디스크의 속도는 디스크에서 컴퓨터로 데이터를 전송하는 비율인 전송률(transfer rate)과 임의접근 시간(random-access time)이라고 하는 위치결정 시간(positioning time)에 의해 결정된다.
- 임의접근 시간의 구성요소
  - 탐색 시간(seek time): 디스크 암을 원하는 실린더 위치로 옮기는데 걸리는 시간
  - 회전 지연(rotation latency): 디스크가 회전하여 원하는 섹터가 디스크 암 위치에 놓이는데 걸리는 시간
- 보통 플래터는 얇은 보호막으로 덮여 있지만 헤드가 자기 표면을 손상시킬 수 있으며, 이 경우에는 복구할 수 없다.
- 디스크 드라이브는 I/O 버스를 통해 컴퓨터에 연결되어 있으며, 현재 널리 사용되는 방식은 다음과 같다.
  - EIDE (Enhanced Integrated Drive Electronics)
  - ATA (Advanced Technology Attachment)
  - SCSI (Small Computer-Systems Interface)
- 데이터의 교환은 특수한 제어기를 통해 이루어진다. 컴퓨터 연결 끝에는 호스트 제어기가 있으며, 디스크 자체 내에는 디스크 제어기가 있다.
- 디스크 제어기는 자체적으로 캐시를 가지고 있다.
- 실제 데이터는 디스크 제어기에 의해 디스크에서 캐시로 옮겨지고, 호스트 제어기는 캐시에 있는 데이터를 주기억장치로 옮긴다.

## 2.4 저장장치의 계층구조

- 저장장치의 계층구조에 위에 위치할 수록 속도는 빠르지만 고가이다. 또한 위에 위치할 수록 휘발성이다.
- 두 저장장치의 속도 차이는 중간에 빠른 캐시를 설치하여 극복할 수 있다.
- 시스템을 구성할 때 저장장치의 계층구조를 균형있게 잘 구성하면 저렴한 가격에 높은 성능을 얻을 수 있다.




<그림 2.2> 저장장치의 계층구조

### 2.4.1 캐싱

- CPU가 데이터를 필요하면 먼저 캐시에 그 데이터가 있는지 검사한다. 만약 있으면 캐시에서 바로 사용하고 없으면 주기억장치에 있는 데이터를 사용하지만 이 데이터의 복사본을 캐시에 보관한다. 이것은 이 데이터를 곧 또 다시 사용할 확률이 높기 때문이다.
- 명령어 캐시처럼 전적으로 하드웨어를 이용하여 구현된 캐시도 있다. 이것은 운영체제가 관여하지 못하므로 이 과목에서는 다루지 않는다.
- 캐시의 크기는 제한되어 있으므로 이것을 잘 관리하여야 시스템의 성능을 높일 수 있다. 캐시의 크기와 교체 정책(replacement policy)을 잘 선택하면 원하는 데이터가 캐시에 있을 확률을 80에서 99%까지 높일 수 있다.
- 주기억장치는 CPU와 보조기억장치 사이에 있는 캐시로 사용될 수 있다.

### 2.4.2 일관성

- 저장장치의 계층구조를 사용하면 같은 데이터가 여러 레벨에 존재할 수 있다.
- 한번에 하나의 프로세스만 동작하면 이것은 아무 문제가 되지 않는다. 그러나 여러 프로세스가 같은 데이터를 접근하고자 하면 모든 프로세스가 최신의 데이터를 얻을 수 있도록 해야 한다. 이 문제는 다중프로세서 시스템에서 더욱 심각하다.
  - 다중프로세서 시스템에서 여러 캐시에 같은 데이터가 동시에 존재할 때 하나의 캐시에서 그 값이 변경되면 다른 캐시도 변경이 반영되어야 한다. 이와 같은 문제를 캐시 일관성(cache coherency) 문제라 한다. 이 문제는 보통 하드웨어 문제이다.
- 분산 환경에서는 여러 파일의 복사본이 여러 컴퓨터에 분산되어 있을 수 있다. 따라서 하나의 복사본에 대한 갱신이 이루어지면 다른 복사본도 갱신되도록 하여야 한다.

## 2.5 하드웨어 보호

- 다중 프로그래밍 환경에서는 하나의 프로세스의 오류가 다른 프로세스에게도 영향을 줄 수 있다.
- 따라서 운영체제는 이런 오류가 다른 프로세스에 영향을 주지 않도록 해야 한다.
- 보통 오류가 발생하면 하드웨어는 트랩을 발생하며, 운영체제는 트랩을 발생시킨 프로세스를 강제로 종료시키고 적절한 오류 메시지를 출력하여 준다.

### 2.5.1 이중모드 동작

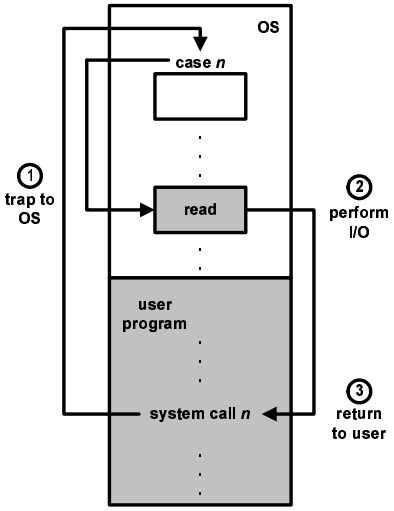
- 여러 오류로부터 프로세스를 보호하기 위해 보통 다음과 같은 두 가지 동작 모드를 제공한다. 현재 모드를 나타내기 위해 하드웨어에 이것을 나타내는 모드 비트(mode bit)가 있다.
  - 사용자 모드: 사용자 프로세스는 이 모드에서 수행되다가 인트럽트나 트랩이 발생하면 모니터 모드로 전환된다.
  - 모니터 모드(monitor mode): 다른 말로 슈퍼바이저 모드(supervisor mode), 시스템 모드, 특권 모드(privileged mode)라 한다.
- 문제를 일으킬 소지가 있는 명령어는 특권 명령으로 분류하고 이 명령은 모니터 모드에서만 수행되도록 하여 프로세스를 보호한다.
- 사용자는 운영체제만이 수행할 수 있는 특권 명령의 수행을 부탁하여 운영체제와 상호작용한다. 이런 요청을 보통 시스템 호출이라 하며, 하드웨어는 이런 시스템 호출을 소프트웨어 인터럽트로 간주한다.

### 2.5.2 입출력 보호

- 사용자가 불법적인 입출력 요청을 할 수 없도록 모든 입출력 명령은 특권 명령으로 분류한다.
- 사용자는 그림 2.3과 같이 시스템 호출을 이용하여 입출력 명령을 실행한다.

### 2.5.3 주기억장치 보호

- 운영체제가 사용하는 주기억장치 영역은 사용자 프로그램으로부터 보호되어야 하며, 사용자 프로그램이 사용하는 주기억장치 영역은 다른 사용자 프로그램으로부터 보호되어야 한다.
- 주기억장치를 보호하기 위해서는 특정 프로그램이 접근할 수 있는 주기억장치 영역을 결정할 수 있어야 한다. 보통 기준 레지스터(base register)와 한계 레지스터(limit register)를 이용하여 이 영역




<그림 2.3> 시스템 호출을 이용한 입출력 수행 과정

을 나타낸다. CPU는 주기억장치 접근을 허용하기 전에 이 두 레지스터를 이용하여 허용여부를 결정한다.

### 2.5.4 중앙처리장치 보호

- 운영체제가 항상 제어권을 확보할 수 있도록 해야 한다. 사용자 프로그램이 무한루프에 빠져 제어권을 다시 운영체제에 넘기지 않을 수 있다.
- 이 문제를 해결하기 위해 타이머를 사용한다. 일정 시간이 경과되면 타이머는 인터럽트를 발생하여 운영체제에 제어권을 넘긴다.
- 기간은 고정되어 있을 수 있고, 가변적일 수도 있다.
- 시분할을 구현하기 위해서도 타이머를 사용한다. 이 경우 시간 구획(time slice)이라고 하는 고정된 기간마다 인터럽트를 발생한다.
  - 시분할에서 기간이 되어 인터럽트가 발생하면 운영체제는 현재 수행중인 프로세스의 상태를 보관하고, 다음 차례의 프로세스의 상태를 적재한 후에 실행한다. 이 과정을 문맥 전환(context switch)이라 한다.