idu 인덕대학교

컴퓨터 시스템 구조
– 메모리 및 성능향상 기법

컴퓨터소프트웨어학과

김병국 교수

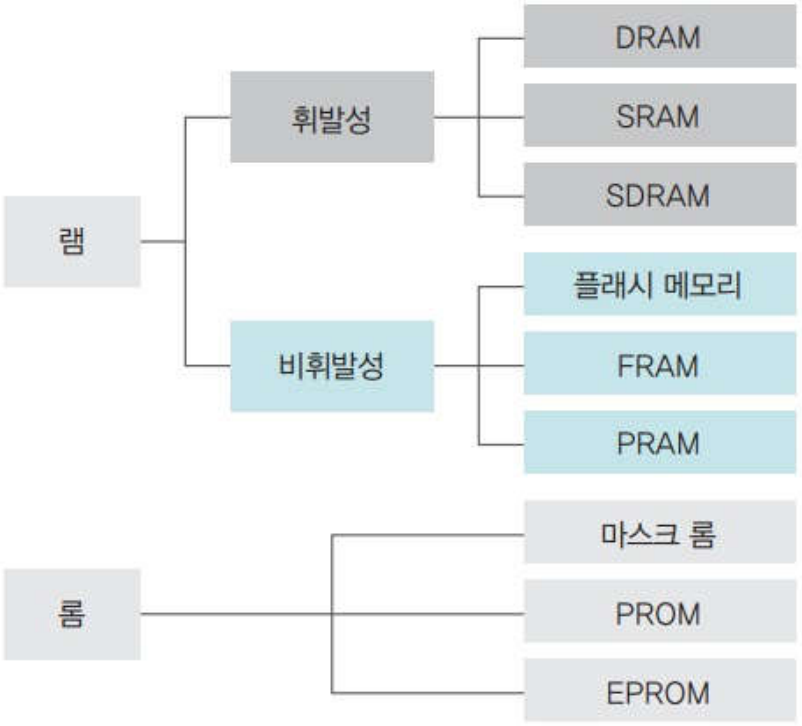
학습목표

- 메모리의 종류와 각 기능을 이해한다.
- 메모리 보호의 필요성을 익힌다.
- 컴퓨터의 성능을 향상하는 기술을 알아본다.
- 인터럽트를 이해한다.

목차

1. 메모리의 종류 (1/3)

❑메모리의 종류

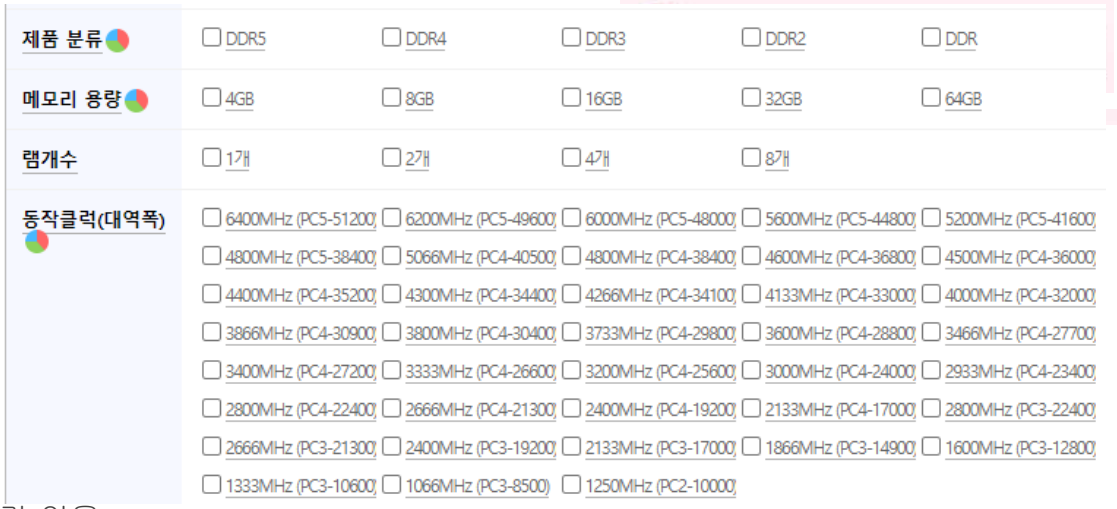


[메모리의 종류]

1. 메모리의 종류 (2/3)

❑ 휘발성 메모리

- DRAM(Dynamic RAM)
  - 저장된 0과 1의 데이터가 일정 시간이 지나면 사라지므로 일정 시간마다 다시 재생시켜야 함
- SRAM(Static RAM)
  - 전력이 공급되는 동안에는 데이터를 보관할 수 있어 재생할 필요가 없음
- SDRAM(Synchronous Dynamic Random Access Memory)
  - 클록틱(펄스)이 발생할 때마다 데이터를 저장하는 동기 DRAM



❑ 비휘발성 메모리

- 플래시 메모리(Flash Memory)
  - SD 카드, USB 드라이브같이 전력이 없어도 데이터를 보관할 수 있는 저장장치
- SSD(Solid State Drive)
  - 빠른 데이터 접근 속도, 저전력, 내구성이 HDD보다 좋음

1. 메모리의 종류 (3/3)

❑ 롬(ROM)의 종류

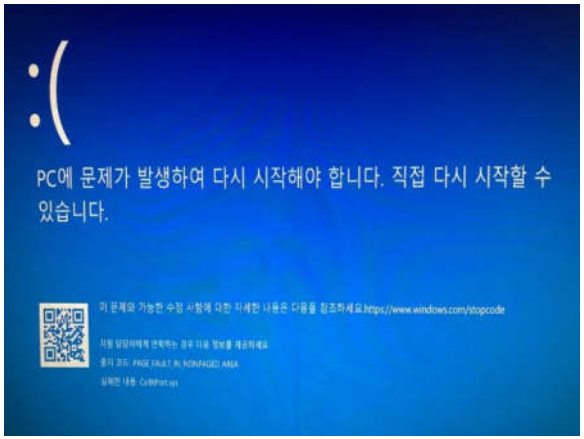
- 마스크 롬(Mask ROM)
  - 데이터를 지우거나 쓸 수 없음
- PROM(Programmable ROM)
  - 전용 기계를 이용하여 데이터를 한 번만 저장할 수 있음
- EPROM(Erasable Programmable ROM)
  - 데이터를 여러 번 쓰고 지울 수 있음

2. 메모리 보호 (1/2)

❑ 메모리 보호의 필요성

- 현대의 운영체제는 **시분할 기법을 사용**하여 여러 프로그램을 동시에 실행
- **사용자(응용프로그램) 영역이 여러 개의 작업 공간으로 나뉨**

- 메모리가 **보호되지 않으면 다른 작업의 영역을 침범**
  - 오류 발생(정상동작 불가)
  - 프로그램을 파괴하거나 데이터를 지울 수 있음
  - 최악의 경우 운영체제 영역을 침범하여 시스템 마비도 가능



[블루 스크린]

2. 메모리 보호 (2/2)

❑ 메모리 보호 방법

① 작업의 메모리 **시작 주소를 경계 레지스터에 저장** 후 작업

② 작업이 차지하고 있는 메모리의 크기, 즉 **마지막 주소까지의 차이를 한계 레지스터에 저장**

③ 사용자의 작업이 진행되는 동안 이 두 레지스터의 **주소 범위를 벗어나는지 하드웨어적으로 점검**

④ **두 레지스터의 값을 벗어 나면 메모리 오류**와 관련된 인터럽트가 발생

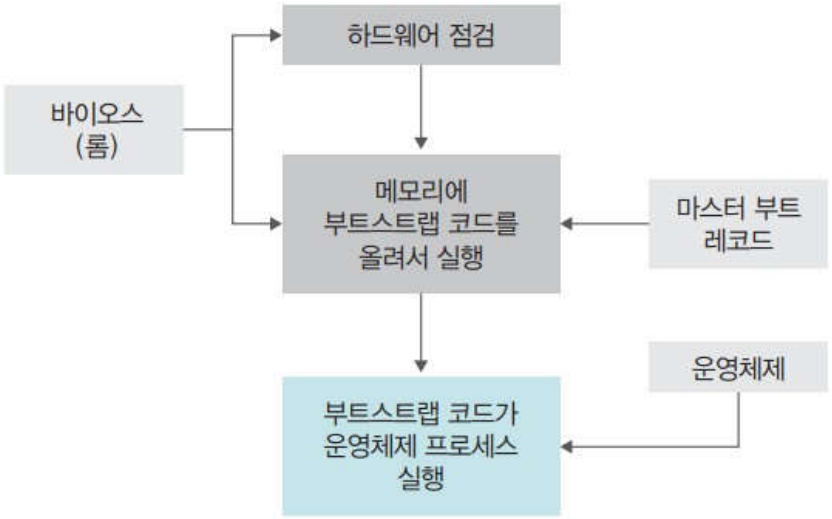
⑤ 메모리 영역을 벗어나서 발생한 인터럽트의 경우 **운영체제가 해당 프로그램을 강제 종료**

[메모리 보호]

3. 부팅

❑ 부팅

- 컴퓨터에 전원을 인가
- 운영체제를 메모리에 적재(Loading)
- 운영체제의 초기화 수행
  - 하드웨어 점검
  - 각종 운영체제 모듈의 기능 구성

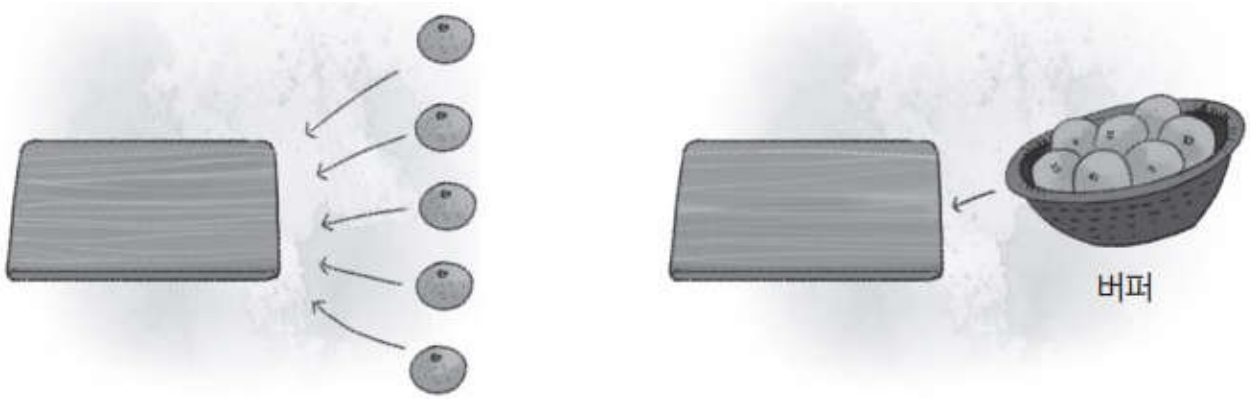


[부팅 과정]

4. 성능 향상 기법 (1/7)

❑ 버퍼(Buffer)

- 속도에 차이가 있는 두 장치 사이에서 그 차이를 완화하는 역할을 하는 장치
- 일정량의 데이터를 모아 옮김으로써 속도의 차이를 완화

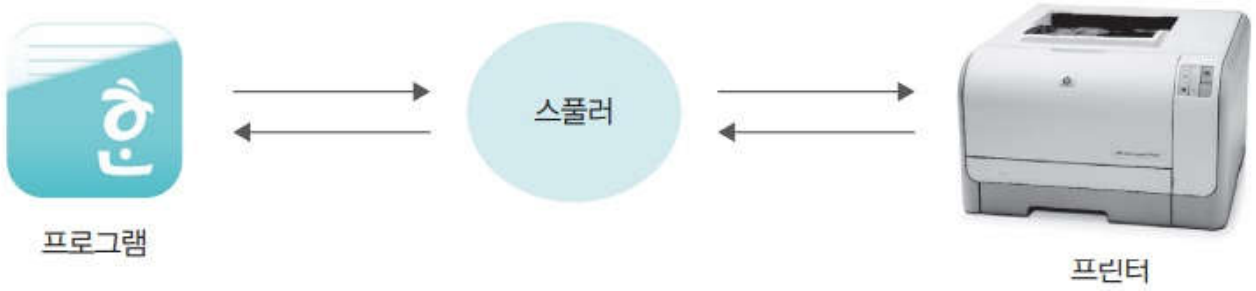


[버퍼의 개념]

4. 성능 향상 기법 (2/7)

❑ 스풀(Spool)

- CPU와 입출력장치가 독립적으로 동작하도록 고안된 소프트웨어적인 버퍼
- [예] 프린트 스풀러
  - 인쇄할 내용을 순차적으로 출력하는 소프트웨어로 출력 명령을 내린 프로그램과 독립적으로 동작
  - 인쇄물이 완료될 때까지 다른 인쇄물이 끼어들 수 없으므로 프로그램 간에 배타적임



[스풀러의 동작 원리]

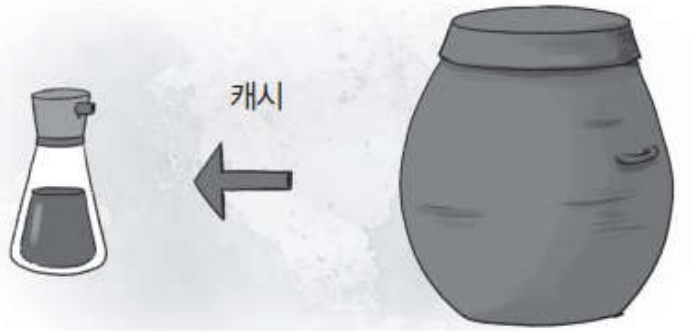
4. 성능 향상 기법 (3/7)

❑ 캐시(Cache) (1/4)

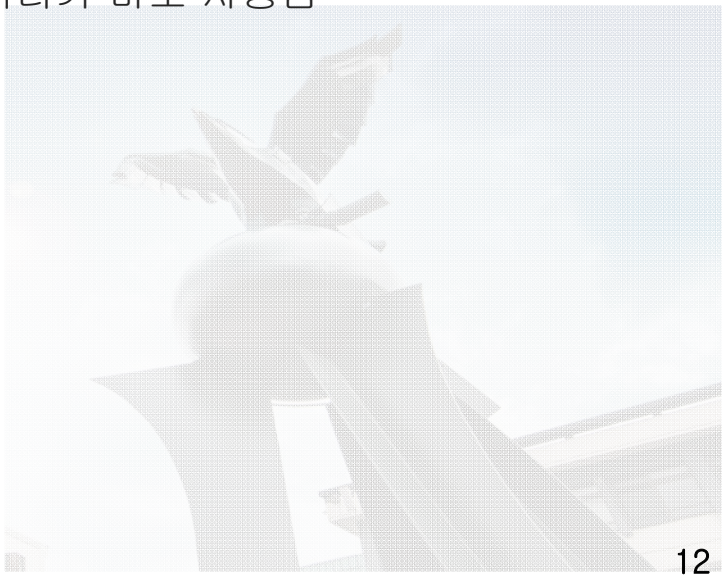
- 메모리와 CPU 간의 속도 차이(BSB와 FSB의 속도 차이)를 완화하기 위해 메모리의 데이터를 미리 가져와 저장해 두는 임시 장소

- 필요한 데이터를 모아 한꺼번에 전달하는 버퍼의 일종으로 CPU가 앞으로 사용할 것으로 예상되는 데이터를 미리 가져다 놓음

- CPU가 특정 메모리 주소에 접근할 때, 캐시에 해당 데이터가 있으면 그 데이터가 바로 사용됨



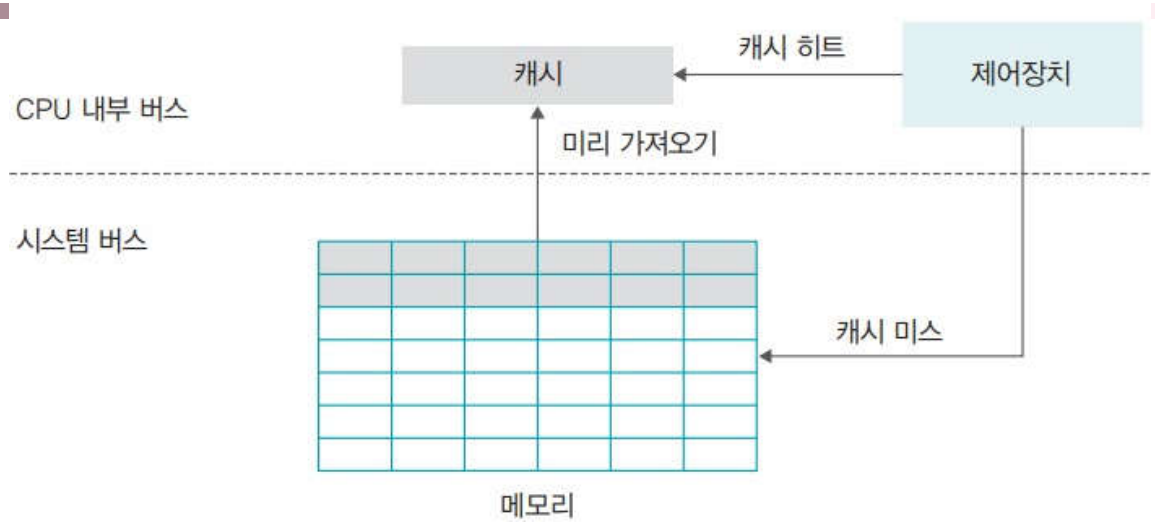
[캐시의 개념]



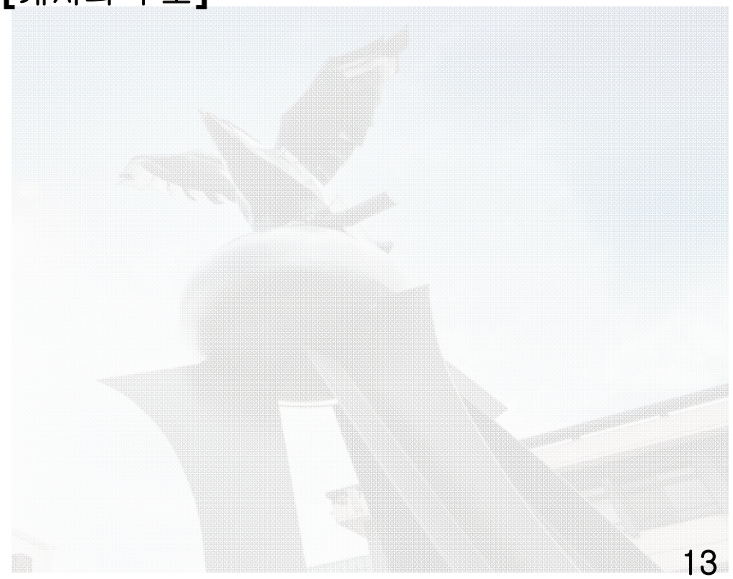
4. 성능 향상 기법 (4/7)

❑ 캐시 (2/4)

- 캐시 히트(Cache Hit)
  - 캐시에 원하는 데이터를 찾음
  - 그 데이터를 바로 사용
- 캐시 미스(Cache Miss)
  - 원하는 데이터가 캐시에 없을 때
  - 메모리로 가서 데이터를 갖고 옴
- 캐시 적중률(Cache Hit Ratio)
  - 캐시 히트가 되는 비율
  - 캐시 적중률이 높을 수록 더 빠른 처리가 가능



[캐시의 구조]



4. 성능 향상 기법 (5/7)

❑ 캐시 (3/4)

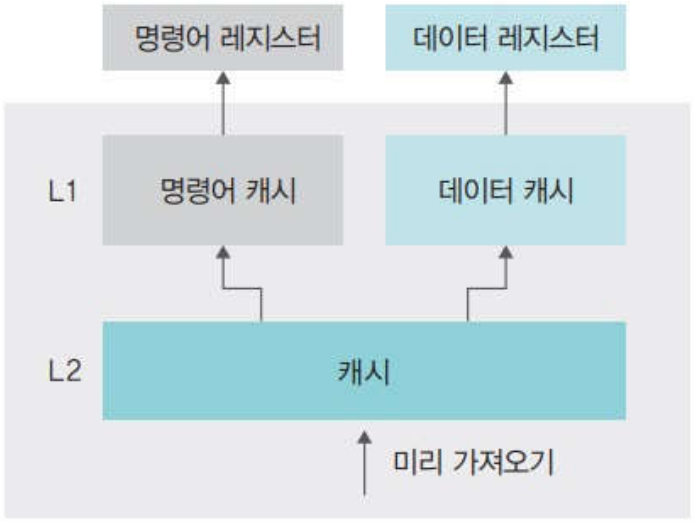
- 즉시 쓰기(Write Through)
  - 캐시의 데이터가 변경되면 이를 즉시 메모리에 반영
  - **메모리와의 빈번한 데이터 전송 → 성능 저하 발생**
  - 메모리에 **최신 값이 유지**되는 장점

- 지연 쓰기(Write Back)
  - **주기적으로 변경된 내용을 메모리에 반영**
  - **카피백(Copy Back)**이라고도 함
  - 메모리와의 데이터 전송 횟수가 줄어들어 **시스템의 성능이 향상됨**
  - 메모리와 캐시 데이터 사이의 **데이터 불일치 가능성 존재**

4. 성능 향상 기법 (6/7)

❑ 캐시 (4/4)

- L1 캐시와 L2캐시
  - 일반 캐시
    - 명령어와 데이터의 구분 없이 모든 자료를 가져옴
    - 메모리와 연결되기 때문에 L2(Level 2) 캐시라고 부름
  - 특수 캐시
    - 명령어와 데이터를 구분하여 가져옴
    - CPU 레지스터에 직접 연결되기 때문에 L1(Level 1) 캐시라고 부름

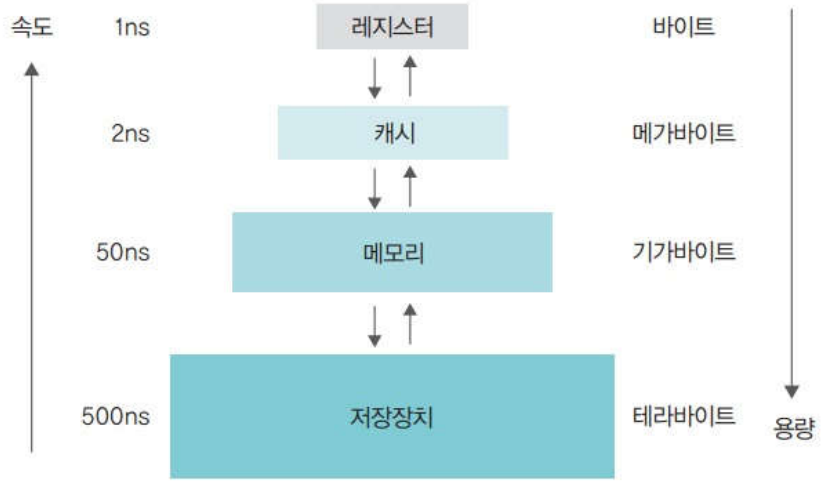


[레벨별 캐시의 구조]

4. 성능 향상 기법 (7/7)

❑ 저장장치의 계층 구조

- 개념
  - 용량이 작고 빠른 저장장치를 CPU 가까운 쪽에 배치
  - 용량이 크고 느린 저장장치를 반대쪽에 배치
  - 효율적인 가격에 적당한 속도와 용량을 동시에 얻는 방법



[저장장치의 계층 구조]

- 이점
  - CPU와 가까운 쪽에 레지스터나 캐시를 배치
    → CPU가 작업을 빨리 진행
  - 메모리에서 작업한 내용을 하드디스크와 같이 저렴하고 용량이 큰 저장장치에 영구적으로 저장할 수 있음

5. 인터럽트 (1/7)

❑ 폴링 방식(Polling)

- CPU가 직접 입출력장치에서 데이터를 가져오거나 내보내는 방식
- CPU가 입출력장치의 상태를 주기적으로 검사
  - 일정한 조건을 만족할 때 데이터를 처리
  - 반복적인 모니터링 작업으로 인해 작업 효율이 떨어짐

[조건을 계속 점검하는 폴링방식]

5. 인터럽트 (2/7)

❑ 인터럽트 방식(Interrupt)

- 입출력 관리자가 대신 입출력을 해주는 방식
- CPU의 작업과 저장장치의 데이터 이동을 독립적으로 운영
  - 시스템의 효율을 높임
- 데이터의 입출력이 이루어지는 동안 CPU가 다른 작업을 할 수 있음

[입출력 관리자를 통해 이벤트를 처리]

5. 인터럽트 (3/7)

❑ 인터럽트

- 입출력 관리자가 CPU에 보내는 이벤트 신호

❑ 인터럽트 번호

- 많은 주변장치 중 어떤 것에 이벤트가 발생되었는지를 CPU에 알려주기 위해 사용하는 번호
- 윈도우 운영체제에서는 IRQ라 부름

❑ 인터럽트 벡터

- 여러 개의 입출력 작업을 한꺼번에 처리하기 위해 여러 개의 인터럽트를 하나의 배열로 만든 것

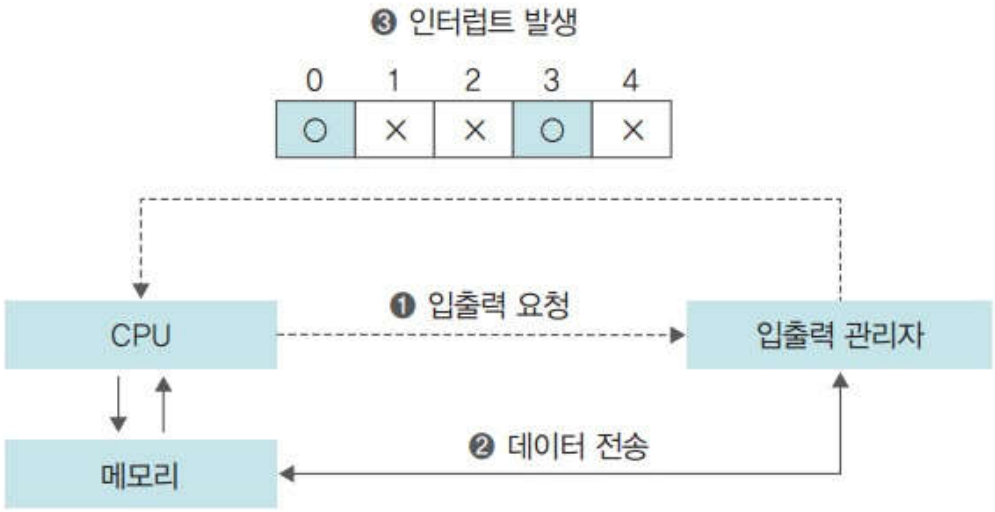
5. 인터럽트 (4/7)

❑ 인터럽트 방식의 동작 과정

① CPU가 입출력 관리자에게 입출력 명령을 보낸다.

② 입출력 관리자는 명령받은 데이터를 메모리에 가져다 놓거나 메모리에 있는 데이터를 저장장치로 옮긴다.

③ 데이터 전송이 완료되면 입출력 관리자는 완료 신호를 CPU에 보낸다.

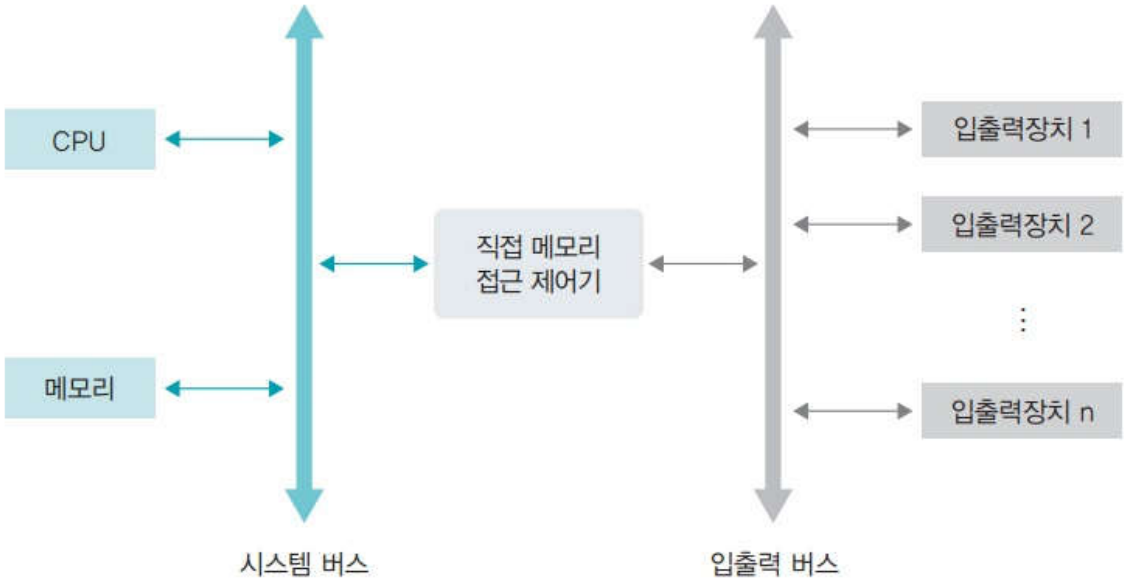


[인터럽트 처리 과정]

5. 인터럽트 (5/7)

❑ 직접 메모리 접근(DMA: Direct Memory Access)

- 입출력 관리자가 CPU의 허락 없이 메모리에 접근할 수 있는 권한
- 메모리는 CPU의 작업 공간이지만, 데이터 전송을 지시받은 입출력 관리자는 직접 메모리 접근 권한이 있어야만 작업을 처리할 수 있음

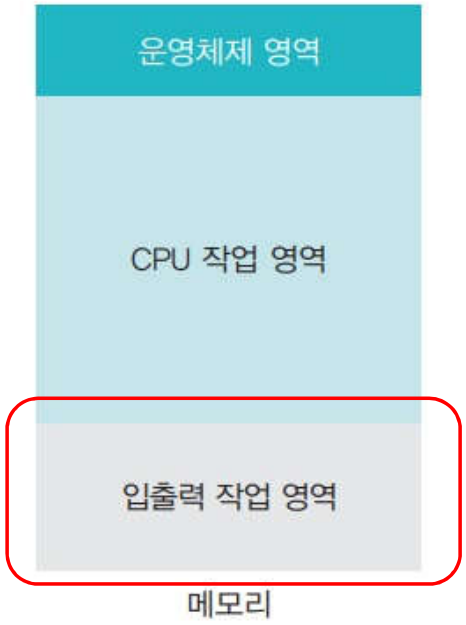


[DMA 제어기 구조]

5. 인터럽트 (6/7)

❑ 메모리 매핑 입출력(MMIO: Memory Mapped I/O)

- 메모리의 일정 공간을 입출력에 할당하는 기법



[메모리 매핑 입출력]

❑사이클 훔치기

- CPU와 직접 메모리 접근(DMA)이 동시에 메모리에 접근하면 → CPU가 양보함
- CPU의 작업 속도보다 입출력장치의 속도가 느리기 때문에 직접 메모리 접근에 양보하는 것 ← 이러한 상황 : **사이클 훔치기(Cycle Stealing)**

수고하셨습니다.