ch8 입출력 장치

08-1 장치 컨트롤러와 장치 드라이버

장치 컨트롤러

- 입출력장치는 앞서 학습한 CPU, 메모리보다 다루기 더 까다로움.왜?

  1. 입출력장치의 종류가 너무 다양
     - 장치가 다양하면 장치마다 속도, 데이터 전송 형식 등 다양
     - 다양한 입출력장치와 정보를 주고받는 방식을 규격화하기 어려움
  2. 일반적으로 CPU와 메모리의 데이터 전송률은 높지만 입출력장치의 데이터 전송률은 낮음
     - **전송률(transfer rate) :** 데이터를 얼마나 빨리 교환할 수 있는지를 나타내는 지표
- **이러한 이유로 입출력장치는**
  - **장치 컨트롤러를 통해 컴퓨터 내부와 정보를 주고받음**

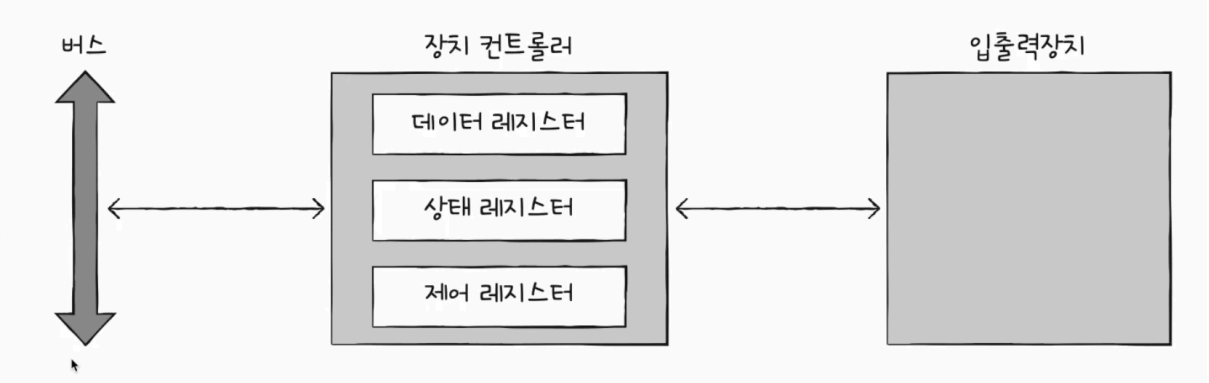
장치 컨트롤러의 역할

1. CPU와 입출력장치 간의 통신 중개
2. 오류 검출
3. 데이터 버퍼링

- 버퍼링?
  - 전송률이 높은 장치와 낮은 장치 사이에 주고받는 데이터를 버퍼라는 임시 저장 공간에 저장하여 전송률을 비슷하게 맞추는 방법

장치 컨트롤러의 구조

- 버스에 연결되어 데이터, 상태, 제어 정보를 주고받음



- **데이터 레지스터 :** CPU와 입출력장치 사이에 주고받을 데이터가 담기는 레지스터 (버퍼)

    - RAM을 사용하기도 함
- **상태 레지스터 :**상태 정보 저장
    - 입출력장치가 입출력 작업을 할 준비가 되었는지
    - 입출력 작업이 완료되었는지
    - 입출력장치에 오류는 없는지 등의 상태 정보
- **제어 레지스터 :** 입출력장치가 수행할 내용에 대한 제어 정보

장치 드라이버

: 장치 컨트롤러의 동작을 감지하고 제어하는 프로그램

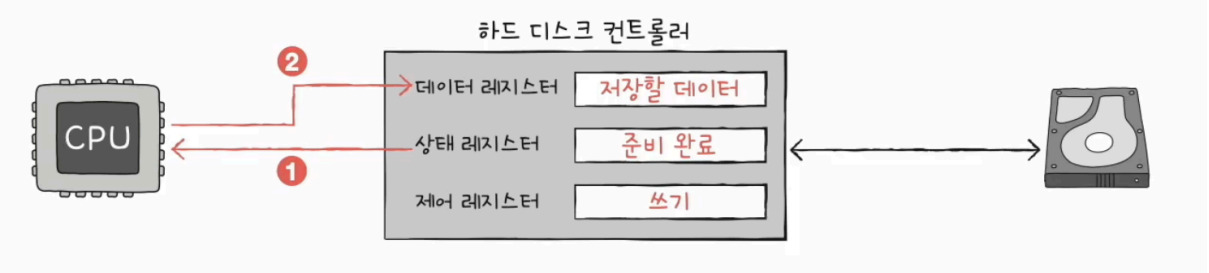
- 장치 컨트롤러가 컴퓨터 내부와 정보를 주고받을 수 있게 하는 프로그램
- **장치 컨트롤러**가 하드웨어적인 통로라면,
- **장치 드라이버**는 소프트웨어적인 통로
    - 장치 드라이버가 설치되어 있지 않다면 해당 입출력장치를 사용할 수 없음
    - **운영체제**가 연결된 장치의 드라이브를 인식하고 실행할 수 있다면 컴퓨터 내부와 정보를 주고받을 수 있음

08-2 다양한 입출력 방법

프로그램 입출력

: 프로그램 속 명령어로 입출력장치를 제어하는 방법

ex. 메모리에 저장된 정보를 하드 디스크에 백업 (= 하드 디스크에 새로운 정보 쓰기)



1. CPU는 하드 디스크 컨트롤러의 제어 레지스터에 쓰기 명령 보냄
2. 하드 디스크 컨트롤러는 하드 디스크의 상태 확인 후 상태 레지스터에 준비 완료 표시
3. CPU는 상태 레지스터를 주기적으로 읽어보며 하드 디스크의 준비 여부 확인 (폴링)
4. 하드 디스크가 준비되었다면 백업할 메모리의 정보를 데이터 레지스터에 쓰기
5. 백업 작업(쓰기 작업)이 끝나지 않았다면 1번부터 반복, 쓰기가 끝났다면 작업 종료

- CPU가 장치 컨트롤러의 레지스터 값을 읽고 씀으로써 이루어짐
    - CPU가 어떻게 레지스터들(입출력장치의 주소)을 아는가?
    - 명령어들이 어떻게 표현되고 어떻게 메모리에 저장되는가?

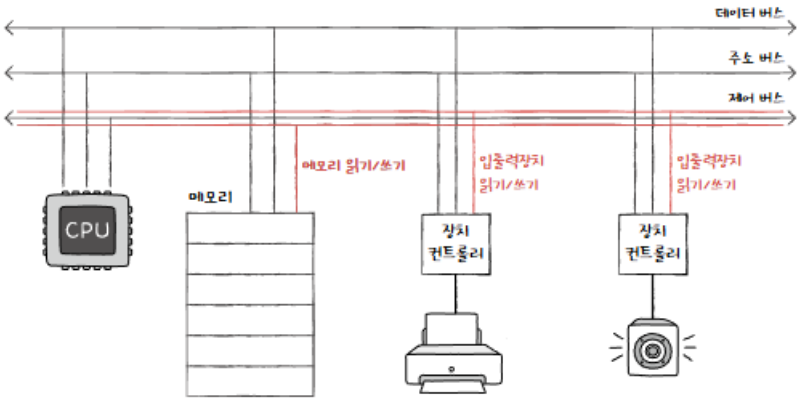
- **메모리 맵 입출력 ,고립형 입출력**

메모리 맵 입출력

: 메모리에 접근하기 위한 주소 공간과 입출력장치에 접근하기 위한 주소 공간을 하나의 주소 공간으로 간주
(메모리 접근 명령어 == 입출력장치 접근 명령어)

고립형 입출력

: 메모리를 위한 주소 공간과 입출력 장치를 위한 주소 공간을 분리하는 방법



- (입출력 읽기/쓰기 선을 활성화시키는) 입출력 전용 명령어 사용

메모리 맵 입출력	고립형 입출력
메모리와 입출력장치는 같은 주소 공간 사용	메모리와 입출력장치는 분리된 주소 공간 사용
메모리 주소 공간이 축소됨	메모리 주소 공간이 축소되지 않음
메모리와 입출력장치에 같은 명령어 사용 가능	입출력 전용 명령어 사용

인터럽트 기반 입출력

- 인터럽트 복습
  - (하드웨어) 인터럽트의 개념: 일종의 알람 역할
  - 플래그 레지스터 속 인터럽트 비트: 하드웨어 인터럽트 받을 수 있을지 없을지 여부
  - 인터럽트 요청 신호: CPU에게 입출력장치가 보내는 신호
  - 인터럽트 서비스 루틴: 인터럽트를 처리하기 위한 특별한 프로그램
- 하드웨어 인터럽트는 장치 컨트롤러에 의해 발생
  - CPU는 장치 컨트롤러에 입출력 작업을 명령하고, 장치 컨트롤러가 입출력장치를 제어하며 입출력을 수행하는 동안 CPU는 다른 일을 할 수 있음.
  - 장치 컨트롤러가 입출력 작업을 끝낸 뒤 CPU에게 인터럽트 요청 신호를 보내면 CPU는 하던 일을 잠시 백업하고 인터럽트 서비스 루틴 실

여러 입출력장치에서 동시 발생한 인터럽트

- 인터럽트 발생 순서대로 처리
  - 플래그 레지스터 속 인터럽트 비트를 비활성화한 채 인터럽트를 처리하는 경우
  - 하드웨어 인터럽트는 이렇게 처리할 수 있으나, non-maskable interrupt(NMI)는 인터럽트 비트가 비활성화되어 있어도 순차적으로 처리하지 않음
  - 현실적으로 모든 인터럽트를 순차적으로 처리할 수 없음
  - 인터럽트 중에서도 우선 순위가 높은 인터럽트가 존재

- **우선순위를 반영하여 인터럽트 처리**
  - **언제 ??**
    - **NMI가 발생한 경우**, 혹은 플래그 레지스터 속 인터럽트 비트를 활성화한 채 인터럽트를 처리하는 경우
  - **PIC(Programmable Interrupt Controller)사용해서 처리!**
    - 여러 장치 컨트롤러에 연결,
    - 장치 컨트롤러의 하드웨어 인터럽트의 우선순위 판단,
    - CPU에게 지금 처리해야 하는 인터럽트가 무엇인지 판단,

    cf)...NMI 우선순위까지 판단하지는 않음

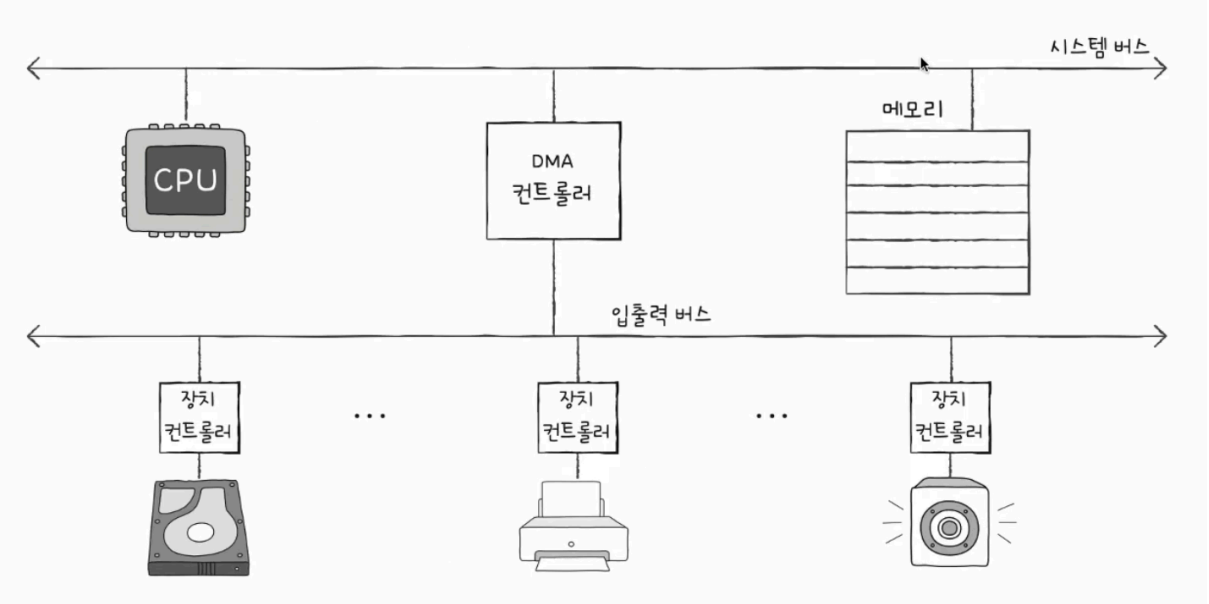
DMA 입출력

- 프로그램 입출력 &인터럽트 기반 입출력의 **공통점**
  - 입출력장치와 메모리 간의 데이터 이동은 CPU가 주도
  - 이동하는 데이터도 반드시 CPU를 거침
- **DMA (Direct Memory Access)**
  - CPU를 거치지 않고 입출력장치가 메모리에 직접적으로 접근하는 기능

DMA 입출력 과정

1. CPU는 DMA 컨트롤러에 입출력 작업을 명령
2. DMA 컨트롤러는 CPU 대신 장치 컨트롤러와 상호작용하며 입출력 작업을 수행
   - 이때 DMA 컨트롤러는 필요한 경우 메모리에 직접 접근
3. 입출력 작업이 끝나면 DMA 컨트롤러는 인터럽트를 통해 CPU에 작업이 끝났음을 알림
   - CPU는 입출력 작업의 시작과 끝만 관여
   - DMA 과정에서 시스템 버스를 사용하는데,
     시스템 버스는 공용 자원이기에 동시 사용 불가능
   - 즉, CPU가 시스템 버스를 사용할 때 DMA 컨트롤러는 시스템 버스를 사용할 수 없고, 반대도 마찬가지
4. DMA 컨트롤러는 CPU가 시스템 버스를 이용하지 않을 때마다 조금씩 시스템 버스를 사용하거나, CPU가 일시적으로 시스템 버스를 이용하지 않도록 허락을 구하고 시스템 버스 이용

입출력 버스



- 시스템 버스를 (불필요하게) 두 번 이용하는 DMA 컨트롤러
  - 입출력 버스를 통해 시스템 버스의 이용 빈도 낮춤
- e.x. PCI 버스, PCI express (PCIe) 버스