idu 인덕대학교

입출력시스템
- 입출력 버스 구조 및 버퍼링

컴퓨터소프트웨어학과

김병국 교수

학습목표

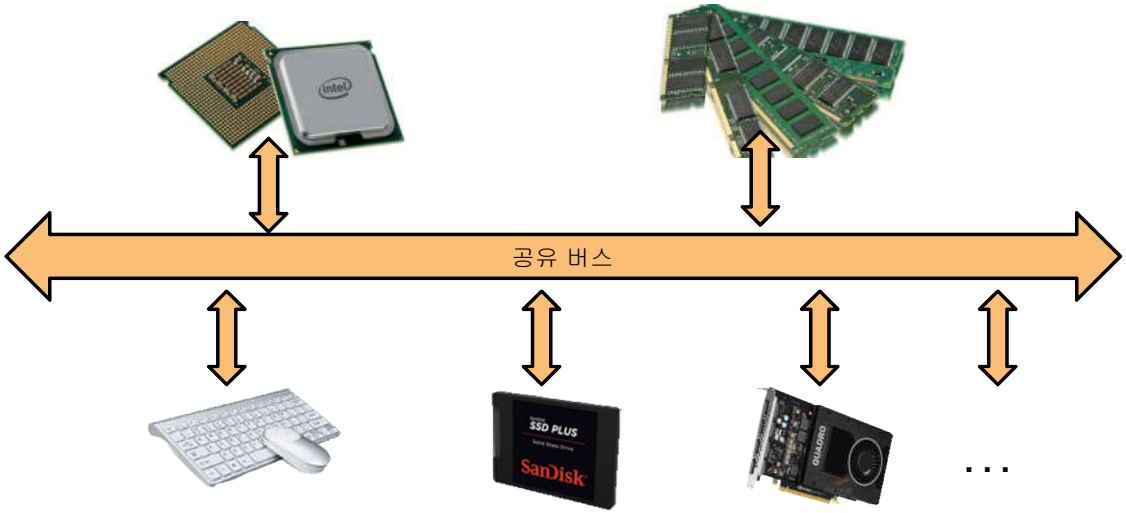
- 입출력 버스의 구조를 안다.
- 버퍼링의 개념을 이해한다.
- 실습을 통한 버퍼링의 장점을 익힌다.
- 캐싱과 스풀링을 이해한다.

목차

1. 입출력 버스 구조 (1/3)

❑ 초기 구조

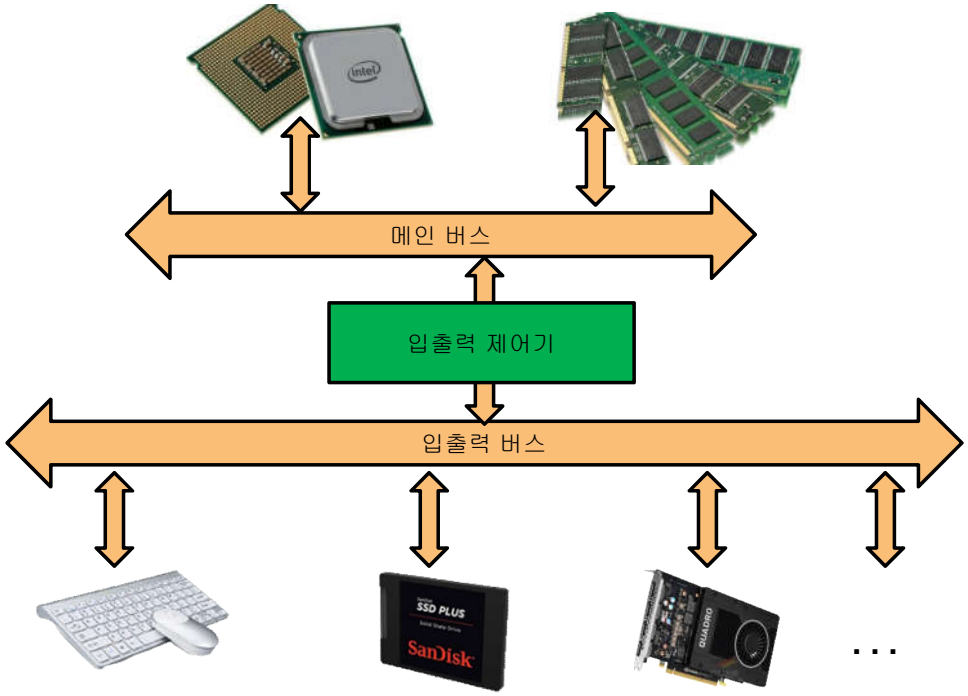
- 모든 장치가 하나의 버스로 연결
- CPU가 입출력장치로부터 데이터를 가져오기 위해 **폴링**(polling) 방식 이용



1. 입출력 버스 구조 (2/3)

❑ 입출력 제어기 적용 구조

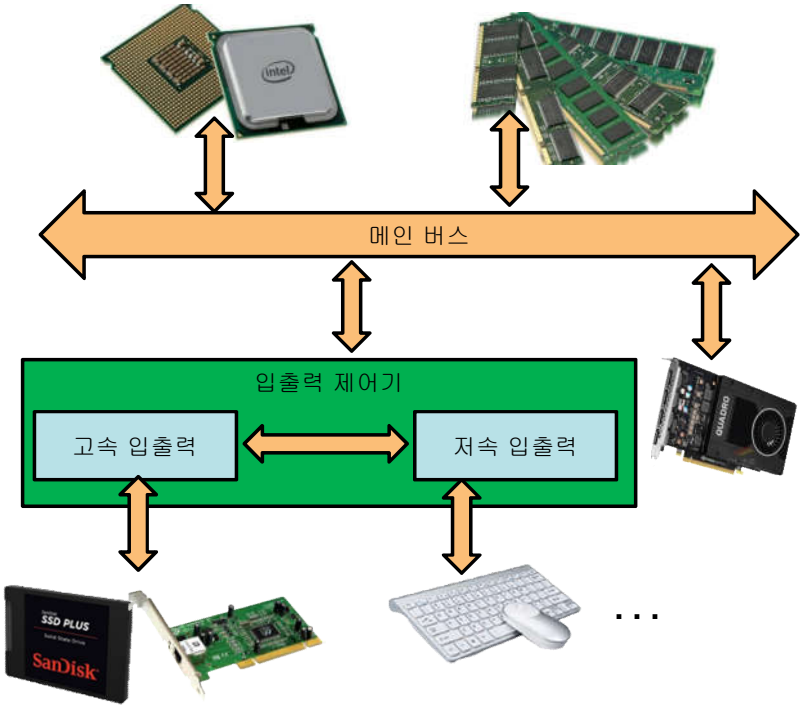
- 메인 버스와 입출력 버스의 2개의 채널로 구성
  - 메인 버스 : 고속으로 작동하는 CPU와 메모리가 사용
  - 입출력 버스 : 주변장치가 사용
- CPU와 메모리의 작업이 느려지는 것을 막을 수 있음
- 단, 주변장치들은 모두 늦은 속도와 동기를 맞춤



1. 입출력 버스 구조 (3/3)

❑ 입출력 버스를 분리

- 입출력 속도를 고려하여 버스를 분리 후 적용



[PC용 메인보드]

2. 버퍼링 (1/4)

❑ 버퍼(buffer)

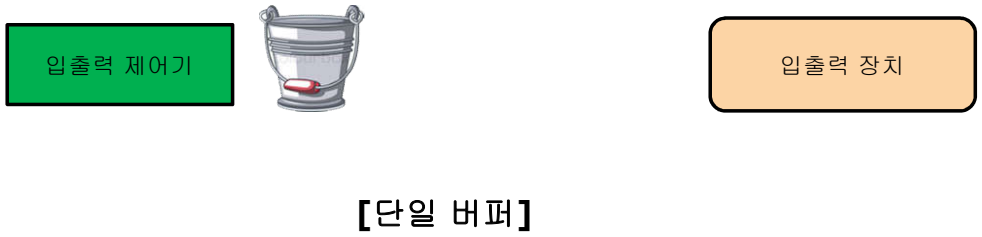
- 속도가 다른 두 장치 간 통신 시 대시 기간을 단축하기 위해 임시로 기록하기 위한 공간
- 장치의 일기/쓰기의 횟수를 감소 → 성능 향상
- 외부 저장장치를 포함한 대부분의 장치들은 버퍼 기술을 사용

버퍼

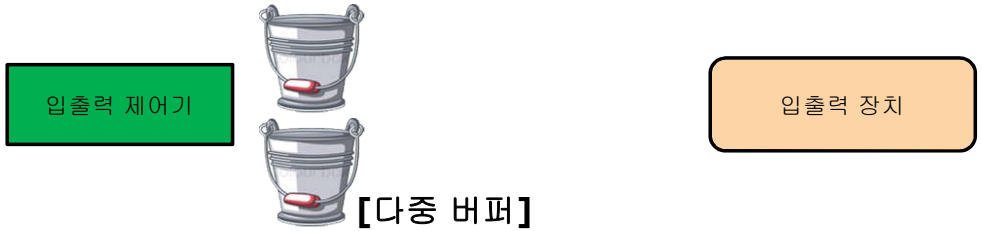
2. 버퍼링 (2/4)

❑ 버퍼의 종류

- 단일(Single) 버퍼
  - 버퍼의 내용이 처리되는 동안, 프로세서는 **대기상태에 진입**하는 문제 발생
- 다중(Multiple) 버퍼
  - 버퍼의 대기상태를 해결하기 위해, 여러 개의 버퍼를 둠 ← 비용 증가



[단일 버퍼]



[다중 버퍼]

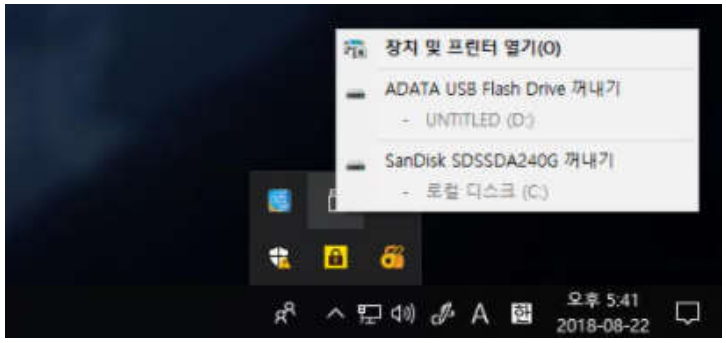
2. 버퍼링 (3/4)

❑ 안전한 하드웨어 제거(Plug Out) (1/2)

- 운영체제는 버퍼가 다 차지 않으면 버퍼가 다 찰 때까지 또는 지정된 시간(timeout) 동안 대기

- 대기상태에 따른 자료 전송이 지연
- 지연되는 상태에서 저장장치를 제거하면, 해당 데이터의 문제가 발생하게 됨

- 하드웨어를 안전하게 제거하기 위해서는 버퍼가 다 차지 않아도 강제로 버퍼의 내용을 저장장치로 옮기기 위한 수동 명령 필요(플러시: flush)



2. 버퍼링 (4/4)

❑ 안전한 하드웨어 제거(Plug Out) (2/2)

- 리눅스의 관련 명령
- 명령어: sync
  - 버퍼내 존재하는 모든 내용을 블록장치에 기록하기 위한 명령
- 명령어: umount [장치명]
  - 파일시스템에서 탈거하기 위한 명령어

3. 프로그래밍 실습 (1/2)

❑논버퍼링 예

```c
#include <stdio.h>
#include <unistd.h>

int
main(void)
{
    for(int i=0; i<10000000; i++)
    {
        char c = 'a' + i%26;
        putchar(c);
        fflush(stdout);
    }

    return 0;
}
```

3. 프로그래밍 실습 (2/2)

❑버퍼링 예

```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int
main(void)
{
    char buffer[BUFSIZ];

    for(int i=0; i<10000000; i++)
    {
        char c = 'a' + i%26;
        buffer[i%BUFSIZ]=c;
        if(i≠0 && i%BUFSIZ==BUFSIZ-1)
        {
            printf(buffer);
            fflush(stdout);
            memset(buffer, 0, BUFSIZ);
        }
    }

    printf(buffer);
    fflush(stdout);

    return 0;
}
```

4. 캐싱

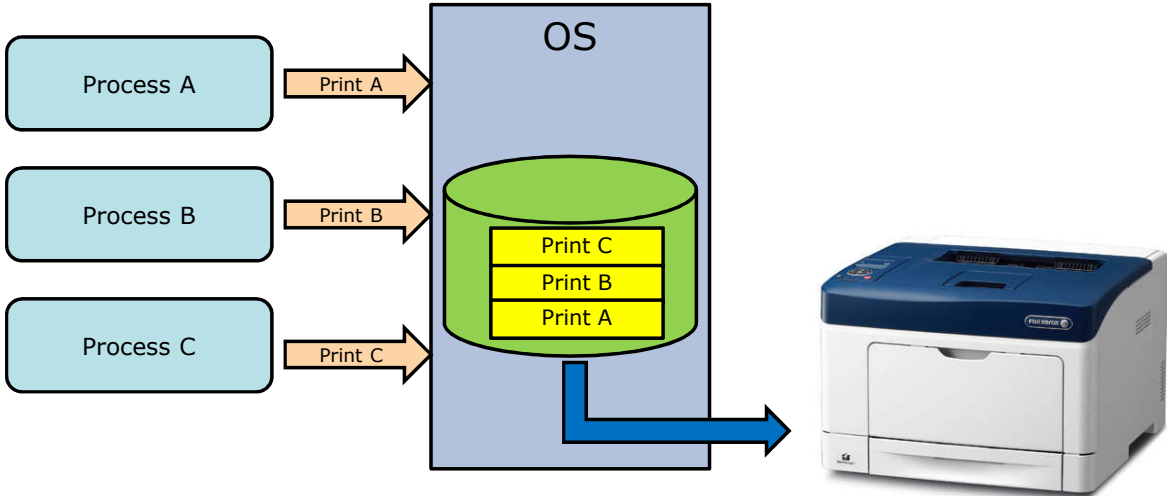
❑ 캐싱(Caching)

- 명령어와 데이터를 캐시에 일시적으로 저장
- 프로세서와 주 기억장치 간의 접근 속도 차이를 줄임 → 컴퓨터 성능 향상 방법
- 캐싱과 버퍼링은 서로 기능이 다름
- 버퍼와의 차이점
  - 캐시는 자주 사용할 자료를 미리 복사하여 저장
    → 빠른 메모리(SDRAM) 영역
  - 캐시 : 데이터의 **복사본**을 저장하는 장소
  - 버퍼 : 데이터가 위치하는 **유일한** 장소

5. 스풀링

❑ 스풀링(Spooling)

- 다수의 프로세스가 하나의 입출력 장치에 처리 기능을 요청할 수 있음(예: 프린팅)
- 운영체제가 요청 기능을 내부적으로 큐(Queue)로 관리하여 순차적으로 대신하여 처리
- 프로세스는 운영체제에 기능을 맡기고 다른 일을 수행할 수 있음



수고하셨습니다.