OPERATING SYSTEM

운영체제 기본 지식

김선우, 신동화, 이대은

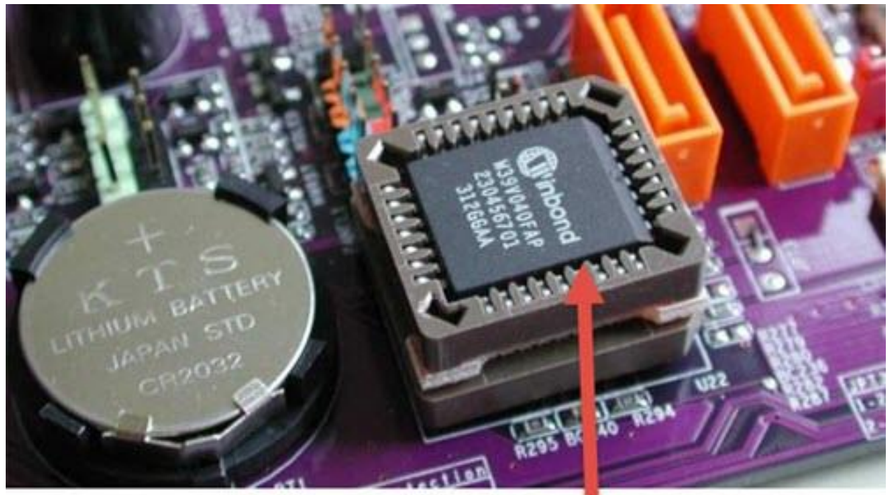
OPERATING SYSTEM

INDEX

0 컴퓨터의 부팅 과정

컴퓨터 부팅 과정
1. 전원을 켠다
2. 메인보드에 전원이 공급된다.
3. **ROM BIOS에도 전원이 공급.**

ROM BIOS는 수은 건전지(CMOS) 근처에 잘 찾아보면 있다.



0 컴퓨터의 부팅 과정

BIOS란?

1. BIOS (Basic Input/Output System)란 **기본 입출력 시스템**이다.
2. BIOS는 운영체제와 하드웨어 사이의 기본 입출력을 담당하기 위해 저 수전의 소프트웨어와 드라이버로 이루어진 **시스템 펌웨어**를 말한다.
3. 메인보드의 BIOS는 ROM BIOS라고 부른다.

펌웨어란?

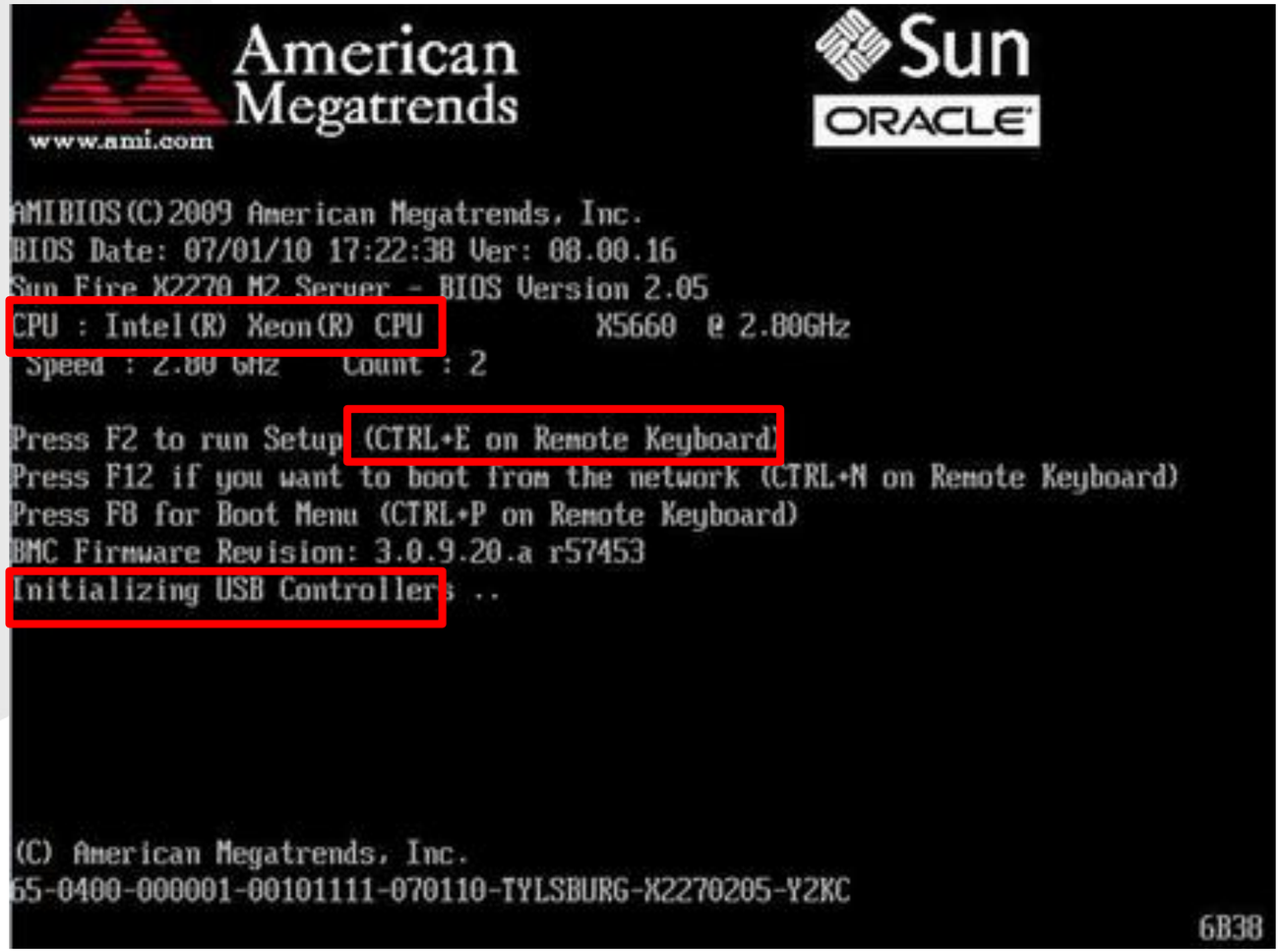
1. 비휘발성 메모리에 내장된 소프트웨어이다.
   (전원이 없어도 그 안에 데이터는 항상 남아있다.)

0 컴퓨터의 부팅 과정

BIOS는 부팅 후 **POST를 수행**한다.

POST란
Power 전원을
On 키고
Self 자신 시스템의
Test 오류를 진단

하는 것을 말한다.



0 컴퓨터의 부팅 과정

POST에서 하는 것들

1. 시스템 버스 테스트
2. 그래픽 카드 테스트
3. 메모리 테스트
4. 키보드 테스트
5. 디스크 테스트
6. P&P기능 동작
7. CMOS 내용 확인
8. DMI기능 동작

0 컴퓨터의 부팅 과정

비프음 횟수	의미	해결방법
2번	패러티 체크(Parity Check) 실패	메모리를 분리했다가 다시 제대로 꽂아봅니다. 그래도 문제가 발생한다면 메모리를 다른 소켓에 꽂아봅니다. 그래도 비프음이 계속 들리고 해당 메모리가 다른 컴퓨터에 꽂았을때 정상적으로 작동한다면 메인보드 불량입니다. A/S를 받으시거나 새 메인보드로 교체하셔야 합니다.
3번	메모리 점검 필요	
4번	시스템 타이머 문제	메인보드 백업 배터리의 문제입니다. A/S를 받으셔야 합니다.
5번	프로세서 오류	CPU 연결 상태 불량 혹은 CPU 오버클럭킹으로 인한 부작용입니다. CPU 클럭수를 다시 낮추거나 새로운 CPU로 교체하셔야 합니다.
6번	키보드 컨트롤러/GATE A20 이상	키보드 이상입니다. 키보드를 다른 것으로 교체하셔야 합니다.
7번	프로세스 인터럽트 오류	사용중인 PC의 바이오스가 지원하지 않는 CPU를 장착했을 때 혹은 CPU 불량일 경우 나는 오류입니다. CPU를 교체하셔야 합니다.
8번	그래픽카드 오류	그래픽 카드가 제대로 연결되어 있지 않거나 불량인 경우입니다. 다른 컴퓨터에 그래픽 카드 불량 유무를 확인 후 메인보드 혹은 그래픽카드 둘 중 하나를 교체해주셔야 합니다.

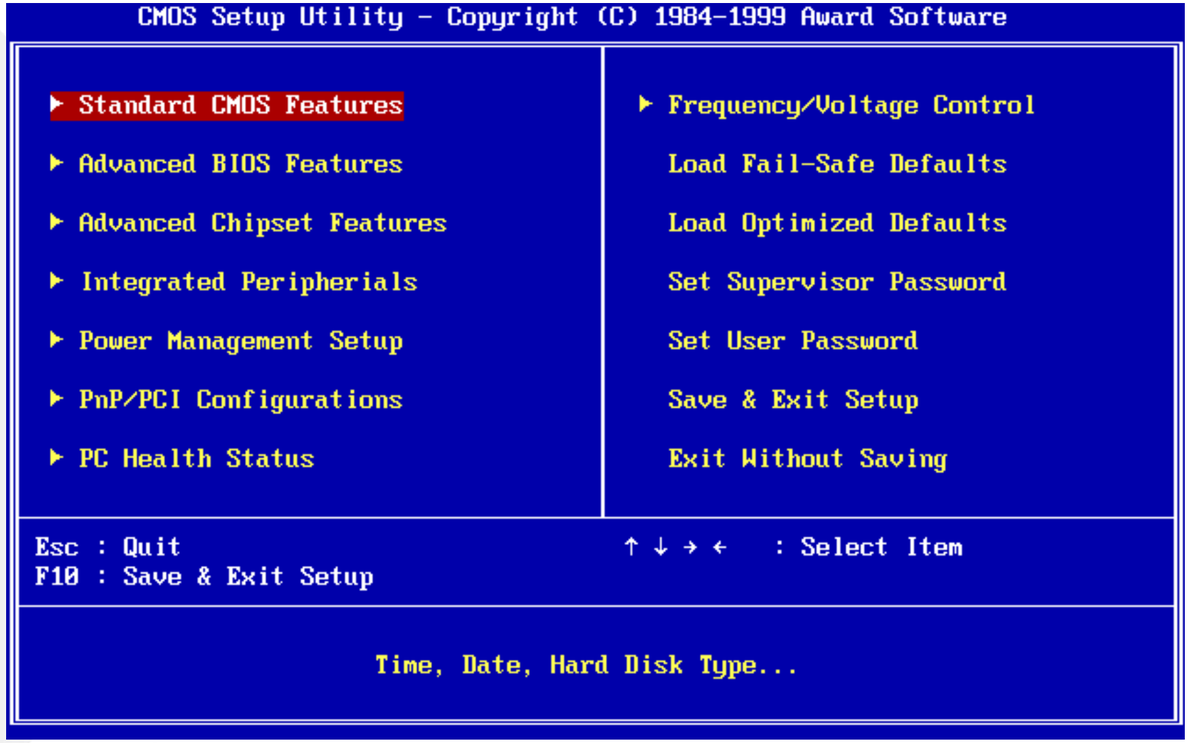
여담으로 BIOS에서 하드웨어 검사를 하기 때문에 문제가 있을 경우에는 비프음으로 하드웨어의 어디에 문제가 있는지 알 수 있다.

제조사 마다 비프음 코드는 다르다.

0 컴퓨터의 부팅 과정

BIOS는 컴퓨터가 꺼졌을 때 설정 값을 유지해야 한다.
그래서 필요한게 COMS이다.

(설정값 : 시스템의 날짜, 시간, 부팅 순서 등등)



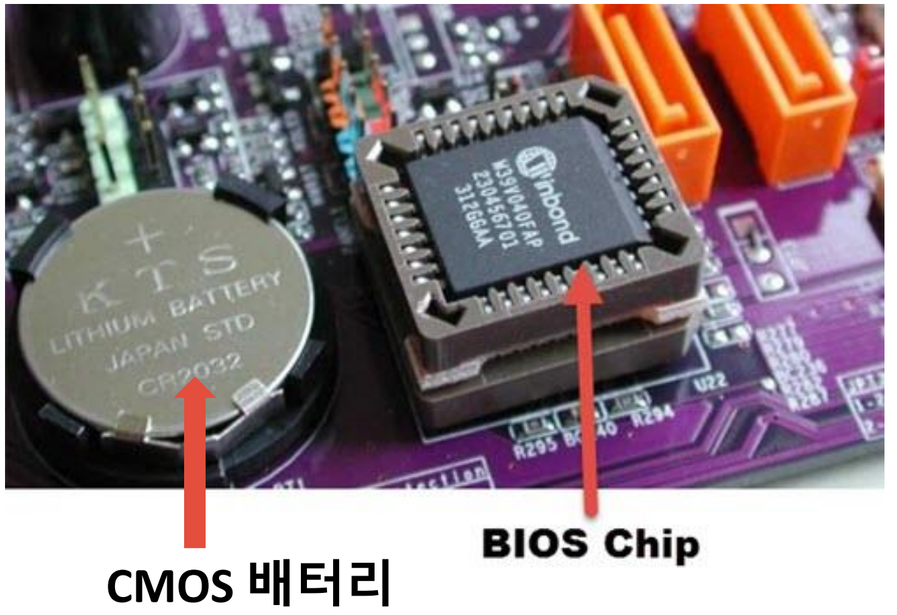
이런 파랑색 BIOS 화면은 Legacy BIOS라고 부른다.

CMOS == BIOS의 설정 값이 저장 되는 곳

0 컴퓨터의 부팅 과정

- CMOS 칩은 휘발성 메모리이며 컴퓨터가 꺼졌을 때 **BIOS 설정 값을 유지하기 위해서 전원(배터리)가 필요**하다.

- 이 수은 배터리가 전원 역할을 한다.



0 컴퓨터의 부팅 과정

- 만약 CMOS 배터리가 방전되거나 문제가 생기면 CMOS안에 있는 BIOS의 설정 값들은 다 날라간다.
- 초기화가 되면 BIOS 설정값을 다시 설정해야 한다.

- 그래서 만약 BIOS에 비밀번호가 걸려 있다면 수은 배터리를 뺏다가 껴서 해킹(?)을 하면 된다.

0 컴퓨터의 부팅 과정

최근의 UEFI는?

- **Legacy BIOS의 업그레이드 버전**이다.
- Legacy BIOS에서의 부족했던 안정성과 편의성을 제공.
  - Legacy BIOS에서 수은 배터리를 뺐다가 끼면 비밀번호 풀리던게 UEFI에서는 안 된다. -> 보안 강화
- 최근의 컴퓨터들은 Legacy BIOS가 아닌 UEFI 사용한다.

0 컴퓨터의 부팅 과정

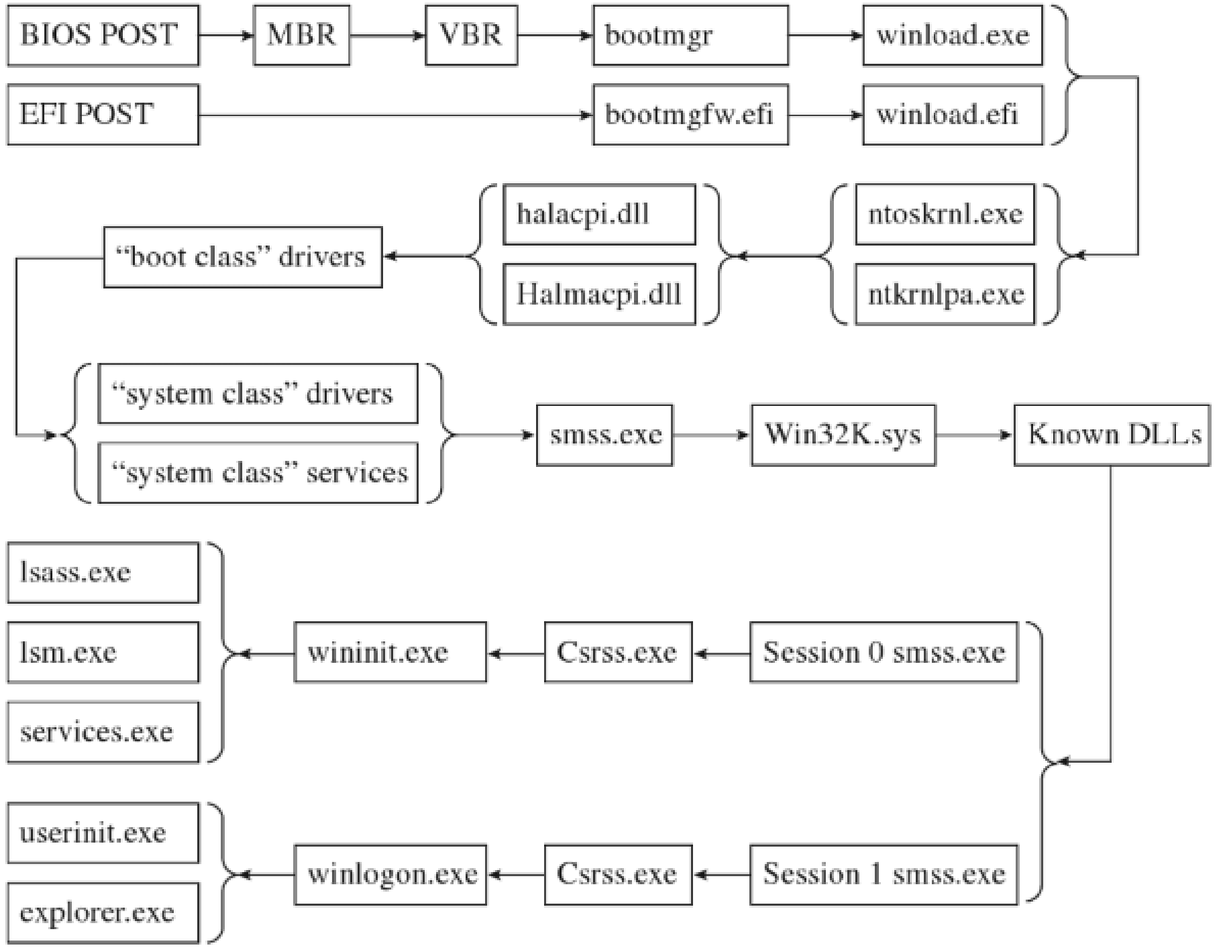
컴퓨터 부팅 과정 정리

1. **전원**을 켠다
2. 메인보드에 **전원이 공급**된다.
3. **부트 프로그램 실행**(BIOS)
4. BIOS에서 **하드웨어 검사**(POST)
5. 운영체제 로드
6. 운영체제 실행

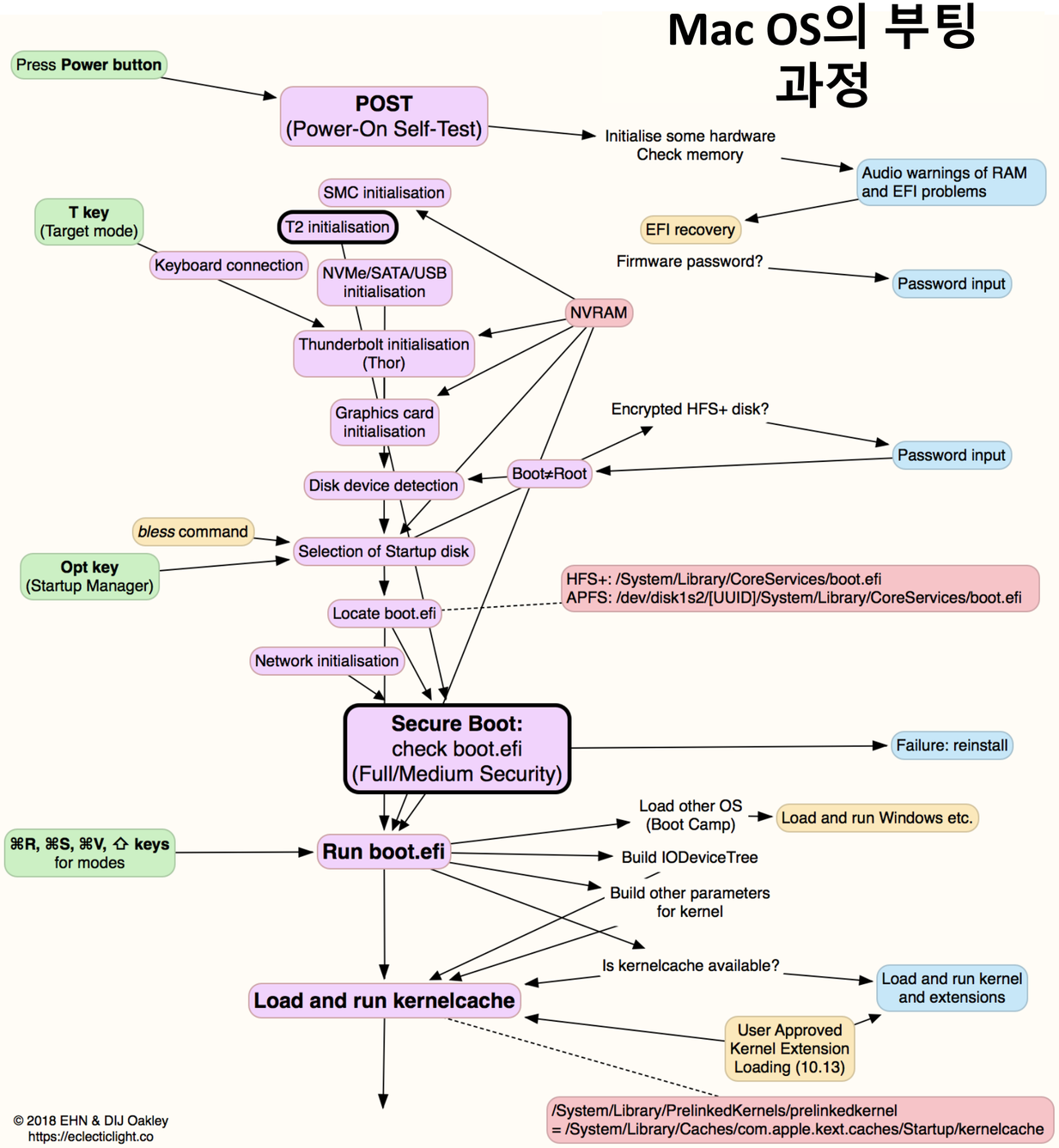
- 지금 까지가 맥, 리눅스, 윈도우들의 **공통적인 컴퓨터 부팅 과정**이다.
- 그 이후로는 각 디스크의 운영체제 구성에 맞게끔 로딩을 한다.

0 각 OS 부팅 과정

<윈도우 OS 부팅 과정>



Mac OS의 부팅 과정



OPERATING SYSTEM

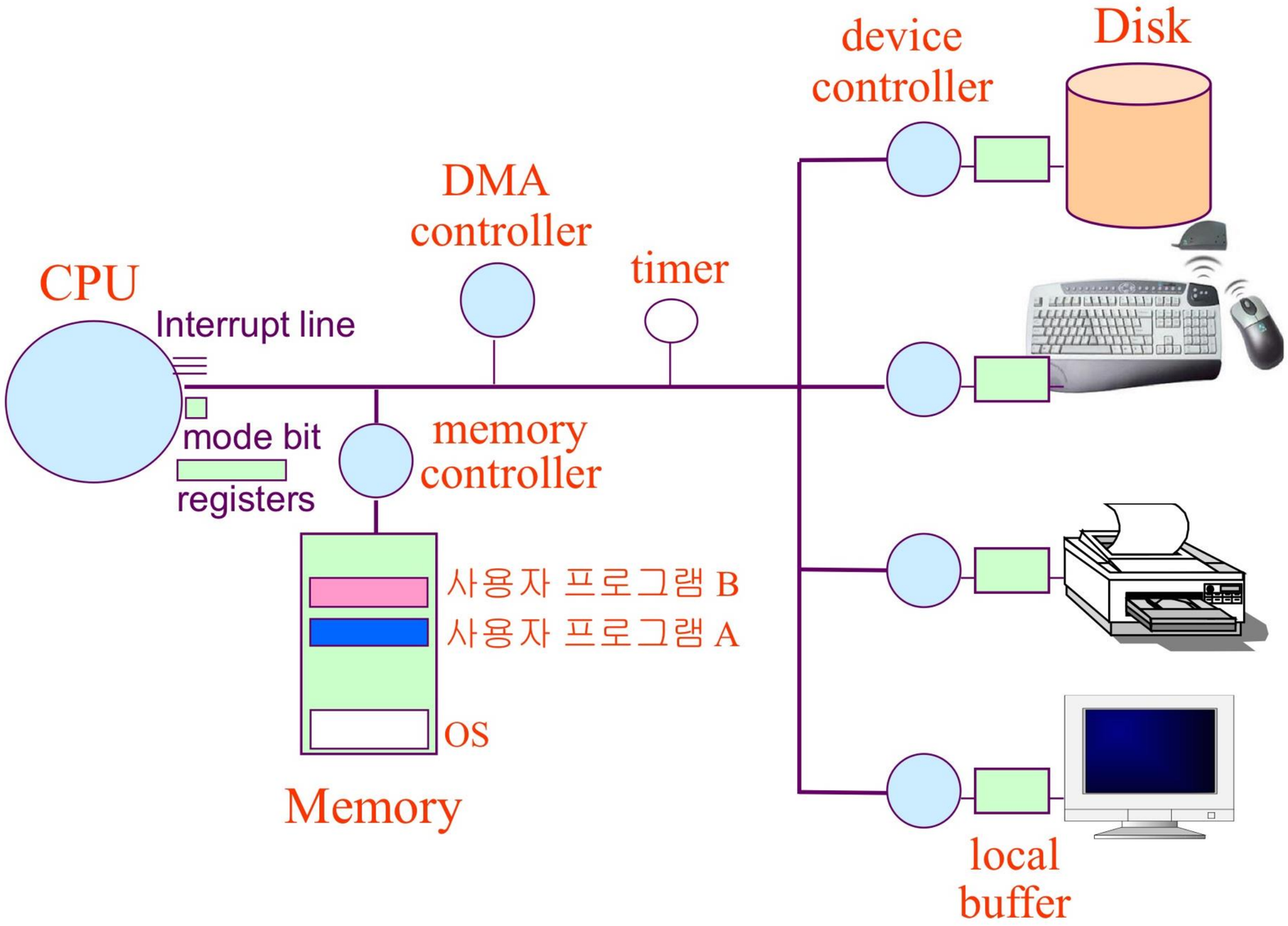
01

입출력 프로그래밍

1. 버퍼 (buffer)
2. 스풀링 (SPOOLing)
3. 채널 (channel)

01

우리가 알고있는 컴퓨터의 구조



01 입출력 프로그래밍이 필요한 이유

레노버 씽크패드 L15 20U7S01000 (SSD 256GB)

노트북 / 운영체제(OS): 미포함(프리도스) / 일반유통상품 / 용도: 사무/인강용 /
화면정보 39.62cm(15.6인치) / 1920x1080(FHD) / 250nit / 눈부심 방지 / 광시야각 /
프로세서 AMD / 라이젠7 PRO-3세대 / 르누아르 / 4750U (1.7GHz) / 옥타코어 /
메모리 DDR4 / 메모리 용량: 8GB / 3200MHz / 메모리 교체: 가능 /
저장장치 M.2(NVMe) / 256GB /
그래픽 내장그래픽 / Radeon Graphics /
네트워크 무선랜: 802.11ax(Wi-Fi 6) / 유선랜: 지원 /
영상입출력 HDMI / 웹캠(HD) /

부품	사양
CPU	라이젠 8코어(**1.7GHz**)
메인보드	FSB(**?**)
메모리	8GB(**3,200MHz**)
하드디스크	256GB (**560MB/s**)?

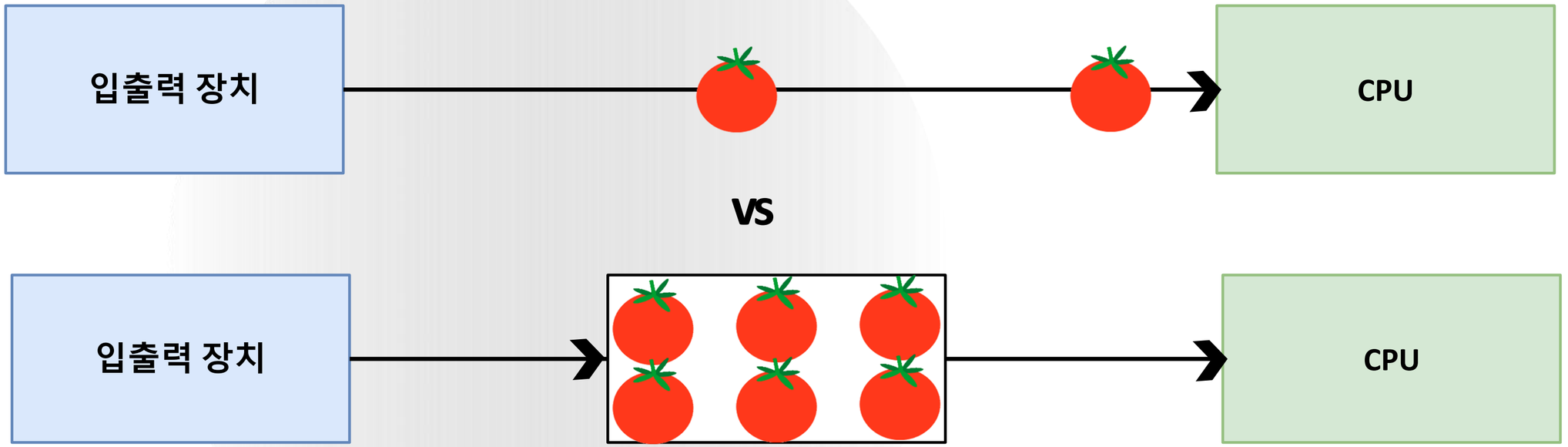
하드웨어들은 데이터 **처리 시간이 일정하지 않다.**
이로 인해 입출력 프로그래밍이 필요하다.

01 버퍼와 버퍼링의 개념

- **버퍼는 임시 저장 공간이다.**
  => 데이터를 한 곳에서 다른 한 곳으로 전송하는 동안 **일시적으로 그 데이터를 보관**하는 메모리의 영역

- **버퍼링**은 **버퍼를 활용하는 방식** 또는 **버퍼를 채우는 동작**을 말한다.

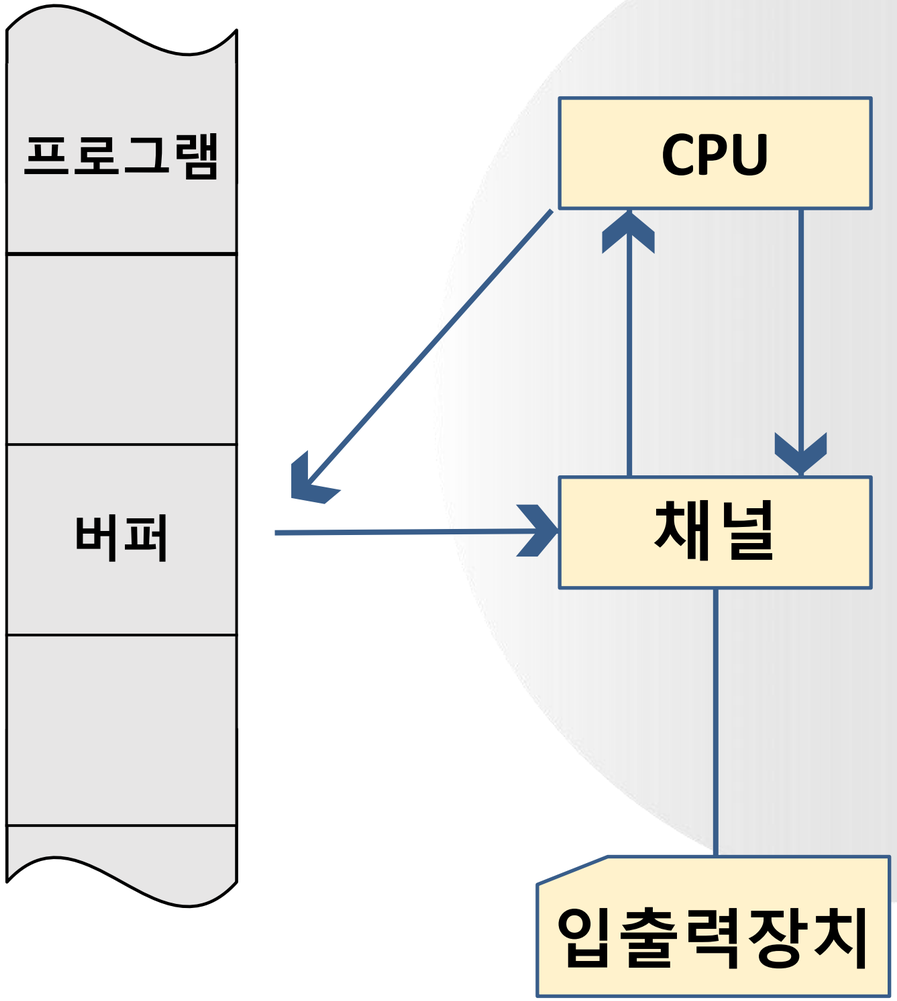
01 버퍼와 버퍼링의 개념



입출력 장치에서 데이터를 **하나씩 전송**하면 전송되는 **데이터 양이 적지만,**
일정량의 데이터를 **모아서 한번에 전송**하면 **많은 데이터를 옮길 수 있다.**
이 때 사용하는 장치가 버퍼다.

01

버퍼링 과정



1. CPU는 출력할 **데이터를 버퍼에 채운다.**
2. 버퍼가 다 채워지면 CPU는 **채널에게 출력을 지시**하고, **자신의 일을 계속**한다.
3. 채널은 출력을 수행함으로써 **버퍼를 비운다.**
4. 출력을 완료함으로써 버퍼를 다 비우면 CPU에게 알린다.

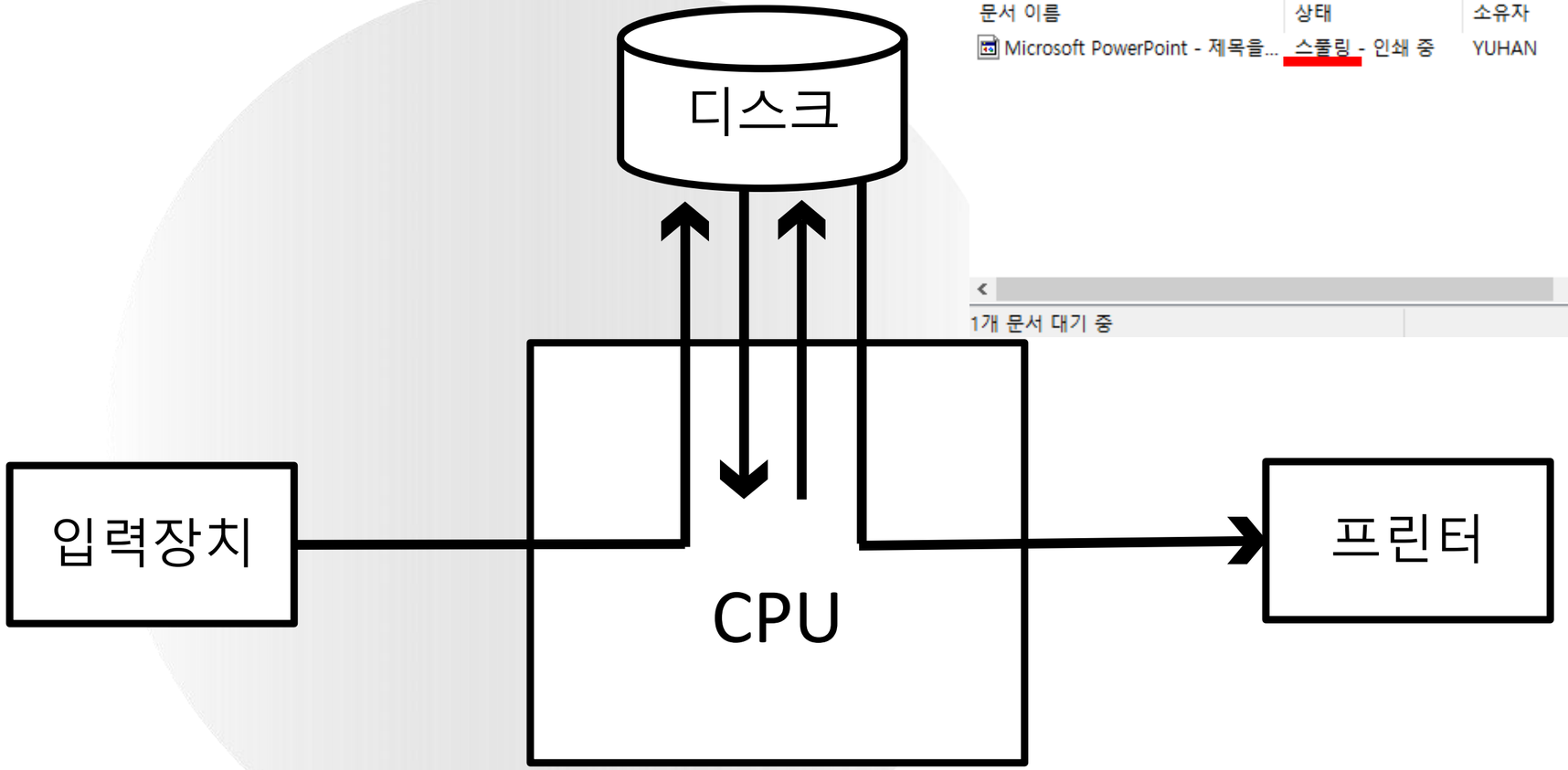
채널 : 주변장치에 대한 제어 권한을 CPU로부터 넘겨받아 CPU 대신 입출력을 관리하는 것.

01 스풀러의 개념

- **버퍼의 일종.**
- 차이점이라고 하면 버퍼는 메모리를
  임시 저장공간으로 사용하고,
  스풀러는 **디스크를 임시 저장공간으로 사용**한다.
- 프린터기를 생각하면 편하다.
  (일괄 작업 시스템하고도 비슷하다)

01

스풀링의 과정



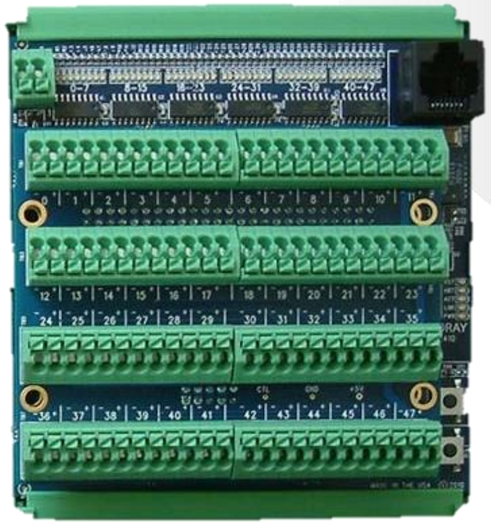
버퍼링을 이해했다면 이 그림을 쉽게 이해할 것입니다.

01 버퍼와 스풀링을 사용하는 이유

- 버퍼링은 주기억장치에서 하드웨어적으로 구현한 것이고 스풀링은 보조기억장치에서 소프트웨어적으로 구현하는 것이다.
- **버퍼링은** CPU와 입출력장치 둘 중에서 어느 하나가 먼저 종료해버리면 되기 때문에 **CPU와 입출력장치를 동시에 사용할 수 없는 반면**
- **스풀링은** 한 작업의 입출력과 다른 작업의 CPU 계산 **작업을 동시에 수행할 수 있다.**

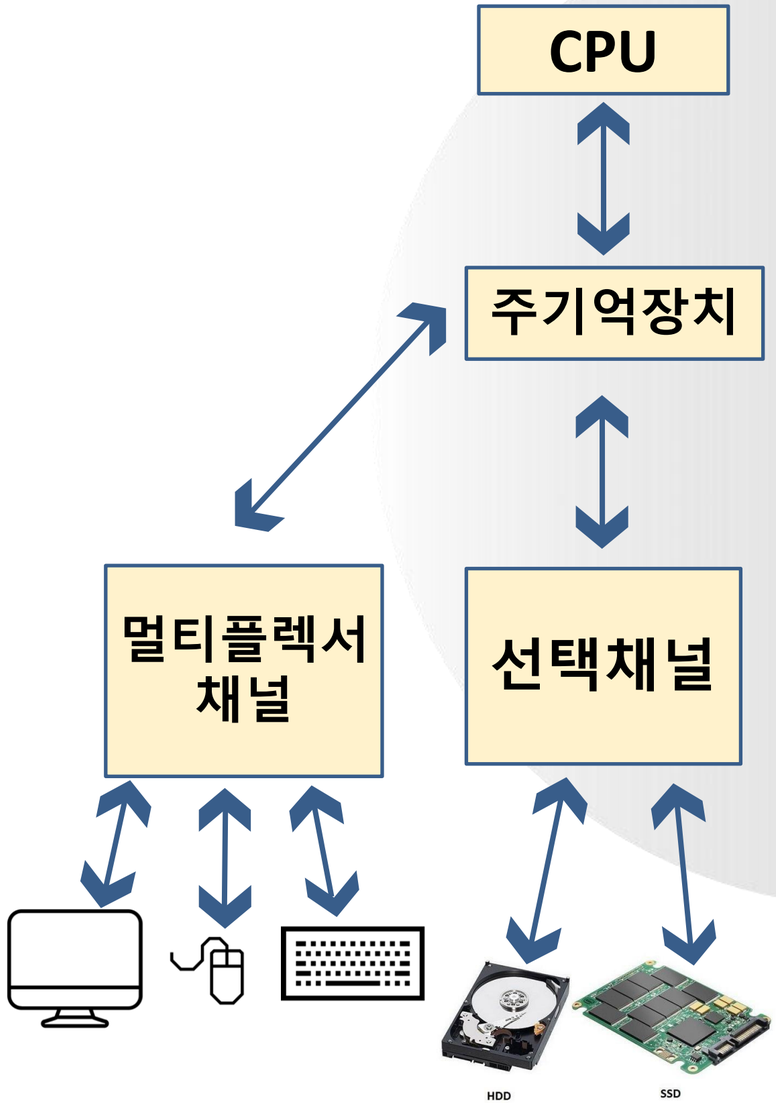
01 채널

- CPU는 입출력이 실제로 수행 완료될 때까지 기다리거나 수시로 입출력장치의 상태를 점검하면 비효율적이다.
- 채널은 중앙 처리 장치와 주변 장치의 **동작을 분리**시키고, 그 사이에서 중앙 처리 장치가 계속 쉬지 않고 다른 일을 할 수 있도록 하는 별도의 컴퓨터 같은 역할을 한다.



라즈베리파이에서는 이 부분이 하드웨어적으로 채널을 담당함

01 채널의 종류



- 선택 채널
  : 여러개의 입출력장치가 연결되어 있다 하더라도 **한 번에 단 하나의 입출력장치만을** 선택적으로 지원한다.
- 멀티플렉서 채널
  : 키보드나 프린터와 같은 비교적 **전송속도가 느린 입출력장치를 제어**하기 위한 채널, **시분할 형태로 제어**한다.

OPERATING SYSTEM

02

병렬처리(Flynn의 분류)

❶ 병렬처리 개념
❷ 병렬 프로그래밍의 어려움
❸ Flynn의 분류

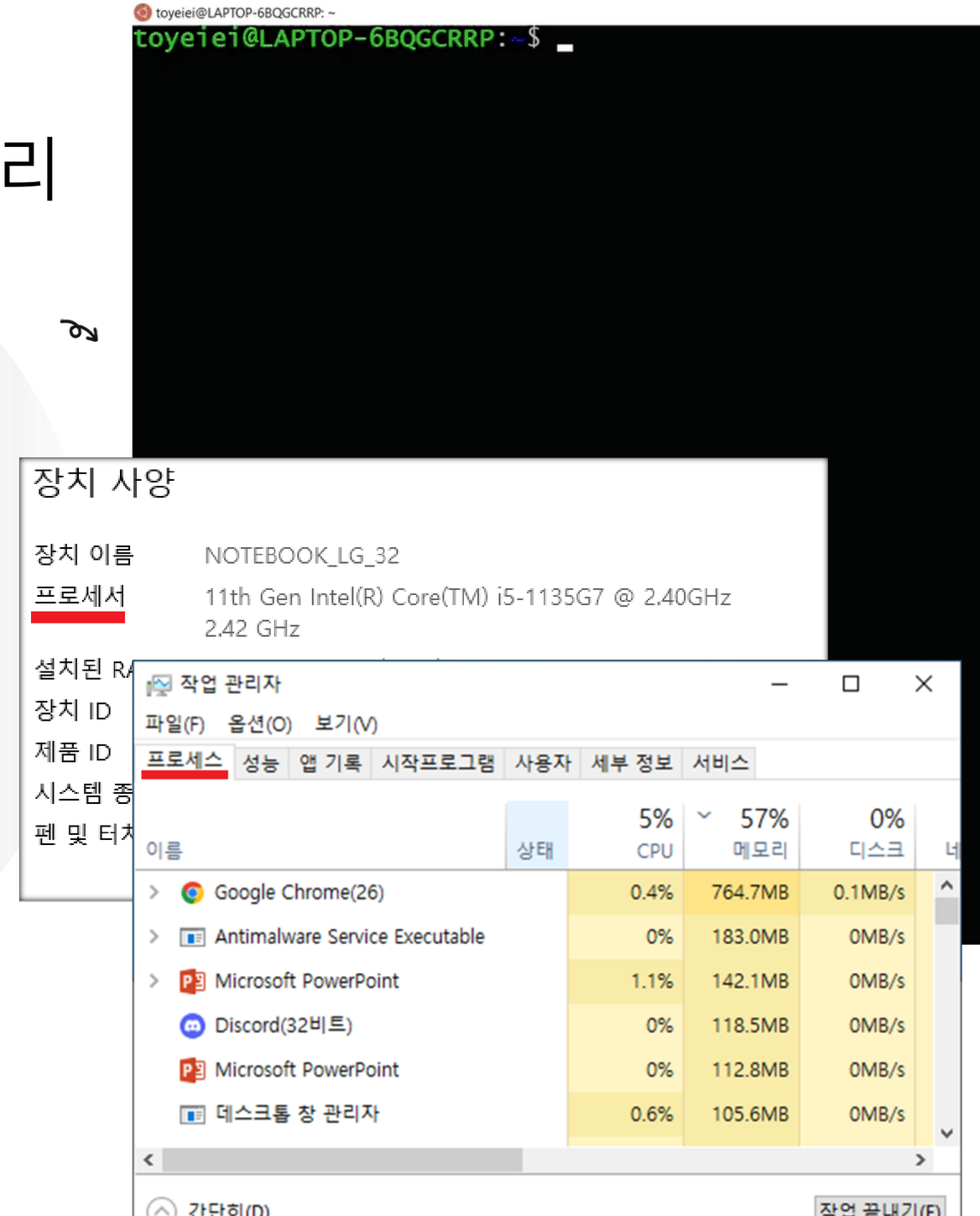
02 병렬처리를 하기에 앞서 용어 정리

병렬처리 기법의 파이프라인과 리눅스에서의 파이프는 전혀 다른거지만
이해를 위해 사진 가져옴.

- **파이프 라인**
  : 한 데이터 처리 단계의 출력이 다음 단계의 입력으로 이어지는 형태

- **프로세서(Processor)**
  : 프로세서 == CPU

- **프로세스(Process)**
  : 컴퓨터에서 실행중인 프로그램



02 병렬처리

컴퓨터 연산을 할 때 가능한 한 **최대의 처리율**을 얻기 위해 병렬성이 필요하다.
하지만 병렬성의 개념을 완벽하게 나타낼 수 있는 프로그램 개발은 현실적으로 어려움이 있다.

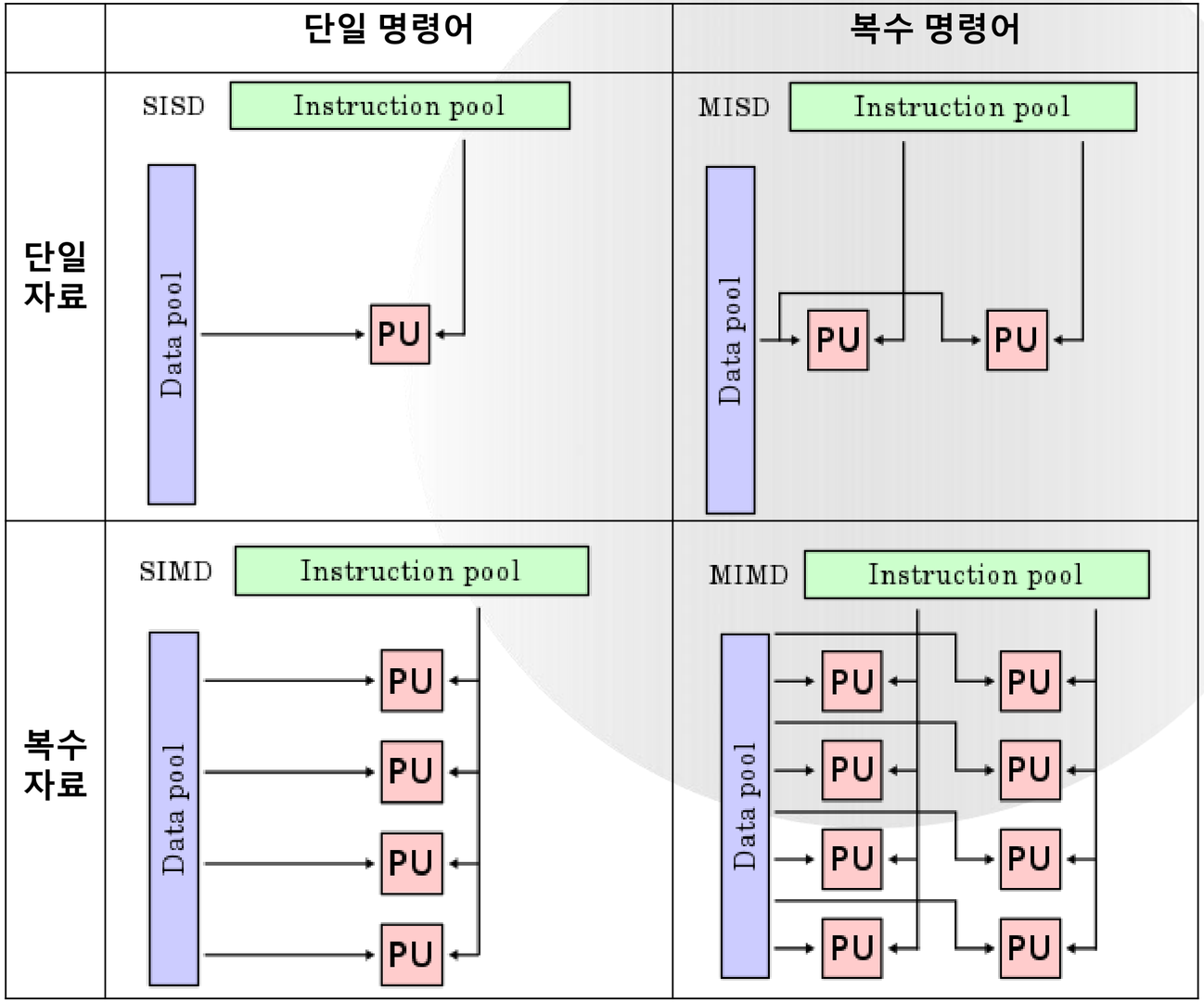
02 병렬성을 완벽하게 나타내는 프로그램 개발이 어려운 이유

- **사람들의 사고 형태는** 순차적으로 이루어지기 때문에 동시성을 가지고 **병렬적으로 생각한다는 것은 어렵다.**
- 병렬성을 효율적으로 표현할 수 있는 **언어가 아직 없다.**
- **컴퓨터 하드웨어는 순차적인 작동의 형태로 구성**되어 있다.
- 병렬 프로그램의 정확성에 대한 증명이 복잡하다.

이름	상호작용 유형	분리 유형	구현체 예시
액터 모델	비동기 메시지 전달	태스크	D 언어, Erlang, Scala, SALSA
벌크 동기적 병렬	공유 메모리	태스크	Apache Giraph, Apache Hama, BSPlib
순차적 프로세스 통신	동기 메시지 전달	태스크	Ada, Occam, VerilogCSP, Go 언어
Circuit	Message passing	태스크	Verilog, VHDL
Dataflow	메시지 전달	태스크	Lustre, TensorFlow, Apache Flink
함수적 프로그래밍	메시지 전달	태스크	Concurrent Haskell, Concurrent ML
LogP 머신	동기 메시지 전달	미정	없음
병렬 랜덤 액세스 머신	공유 메모리	데이터	Cilk, CUDA, OpenMP, Threading Building Blocks, XMTC

← 몇 가지 병렬 프로그래밍 언어는 있으나 이 중 어느 것도 완벽한 병렬 프로그래밍은 없다.

02 Flynn의 분류 Flynn's taxonomy

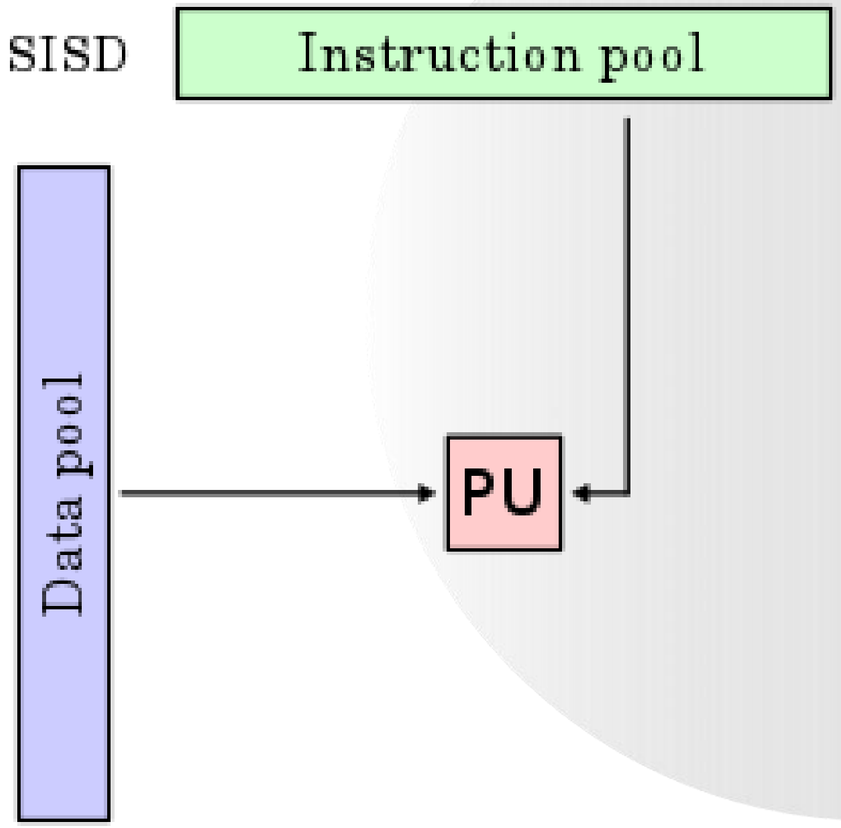


- 플린 분류는 마이클 플린이 1966년에 제안한 컴퓨터 **아키텍처 분류**이다.
- 4개의 아키텍처가 아래와 같이 시각적으로 표시된다.

02

SISD Single Instruction, Single Data

한 번에 데이터 하나를 명령어 하나로 처리하는 기법.

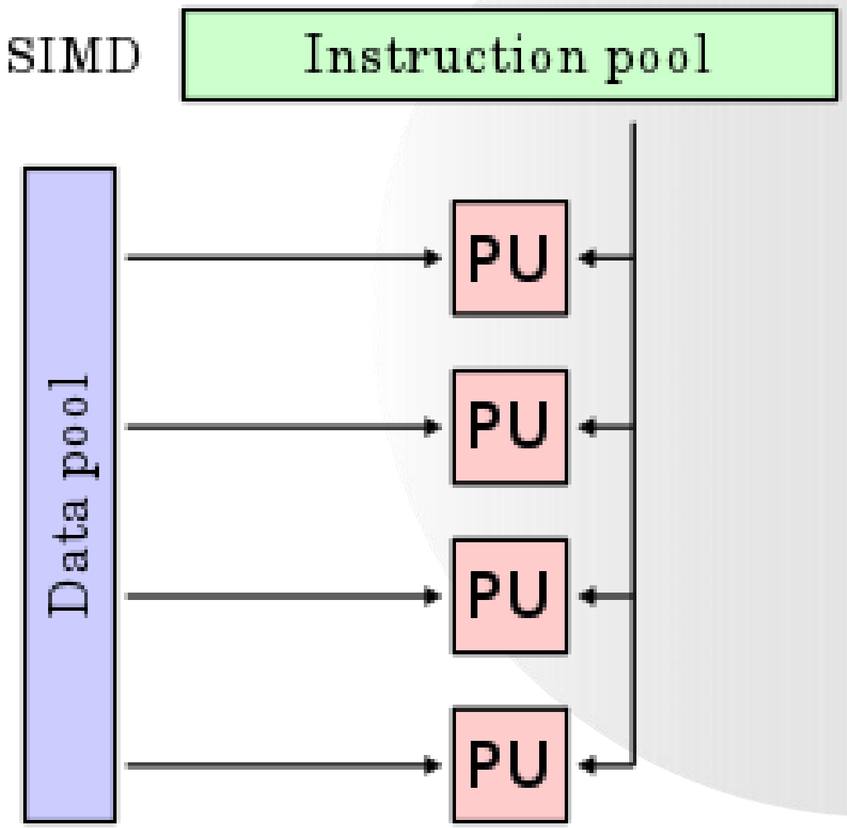


- 폰 노이만 구조를 생각하자.
- **한 번에 한 개씩**의 명령어를 처리하는 단일 프로세서 시스템
- 파이프라이닝과 슈퍼스칼라(superscalar) 구조를 이용

파이프라인을 여러 개 두어 명령어를 동시에 실행하는 기술

02 SIMD

Single Instruction Multiple Data
한 번에 데이터 여러 개를 명령어 하나로 처리하는 기법.

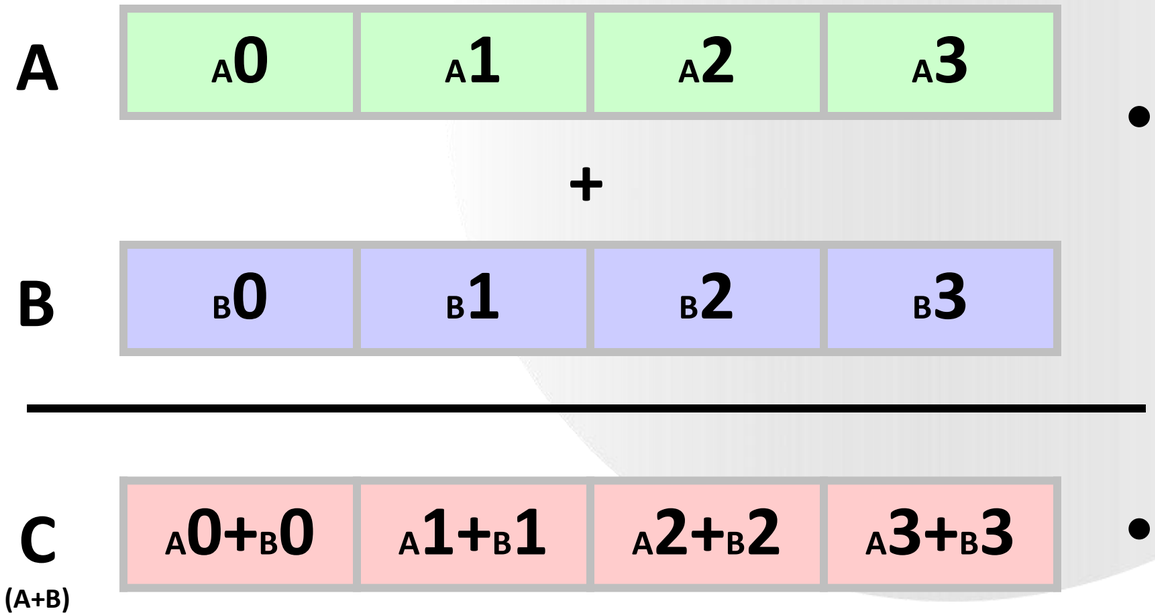


- 배열 프로세서와 파이프라인이 이 분류에 속한다.
- 모든 CPU가 같은 명령어를 수행하지만 **병렬적**으로는 다른 **데이터를 처리**하는 구조.
- 가장 일반적인 병렬 연산 기법

02 SIMD

Single Instruction Multiple Data
한 번에 데이터 여러 개를 명령어 하나로 처리하는 기법.

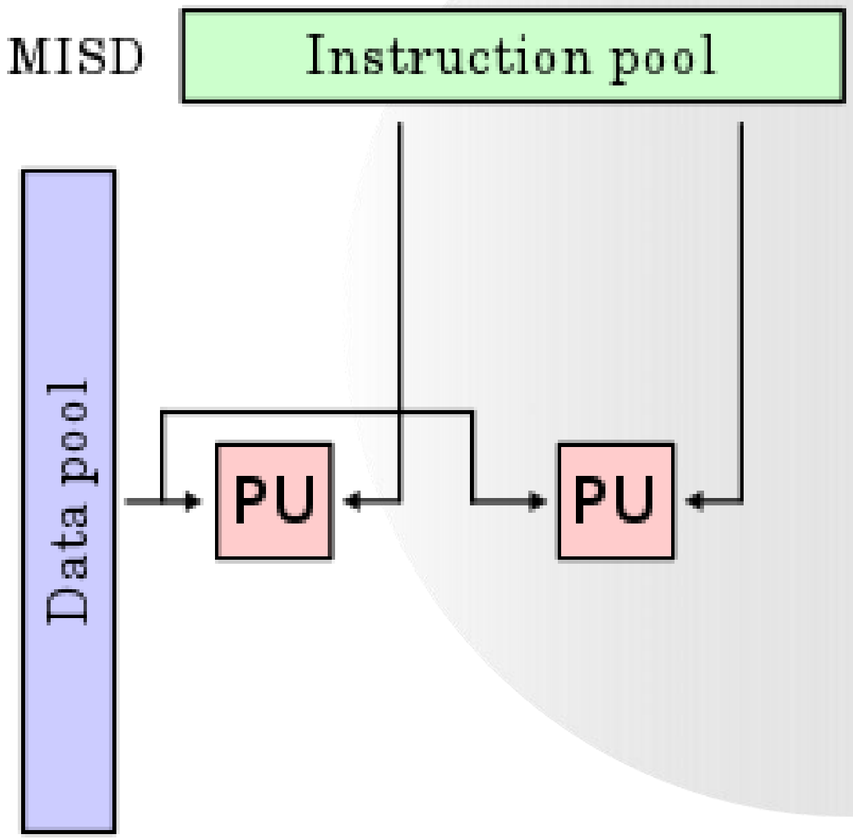
```
for(i=0; i<4; i++)
{
    C[i] = A[i] + B[i]
}
```



- 배열 프로세서와 파이프라인이 이 분류에 속한다.
- 모든 CPU가 같은 명령어를 수행하지만 **병렬적**으로는 다른 **데이터를 처리**하는 구조.
- 가장 일반적인 병렬 연산 기법

02 MISD

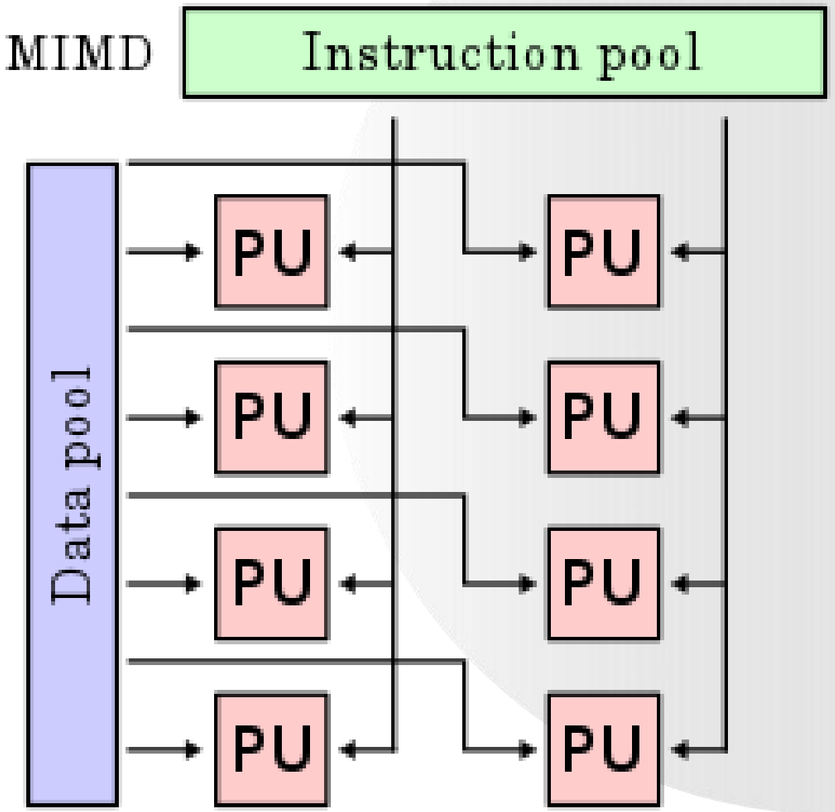
Multiple **I**nstruction, **S**ingle **D**ata
한 번에 데이터 한 개를 여러 명령어로 처리하는 기법.



- 여러 개의 명령어로 하나의 데이터를 처리하는 방식
- 이론적으로만 존재.

02 MIMD

Multiple **I**nstruction, **M**ultiple **D**ata
한 번에 데이터 여러 개를 명령어 하나로 처리하는 기법.

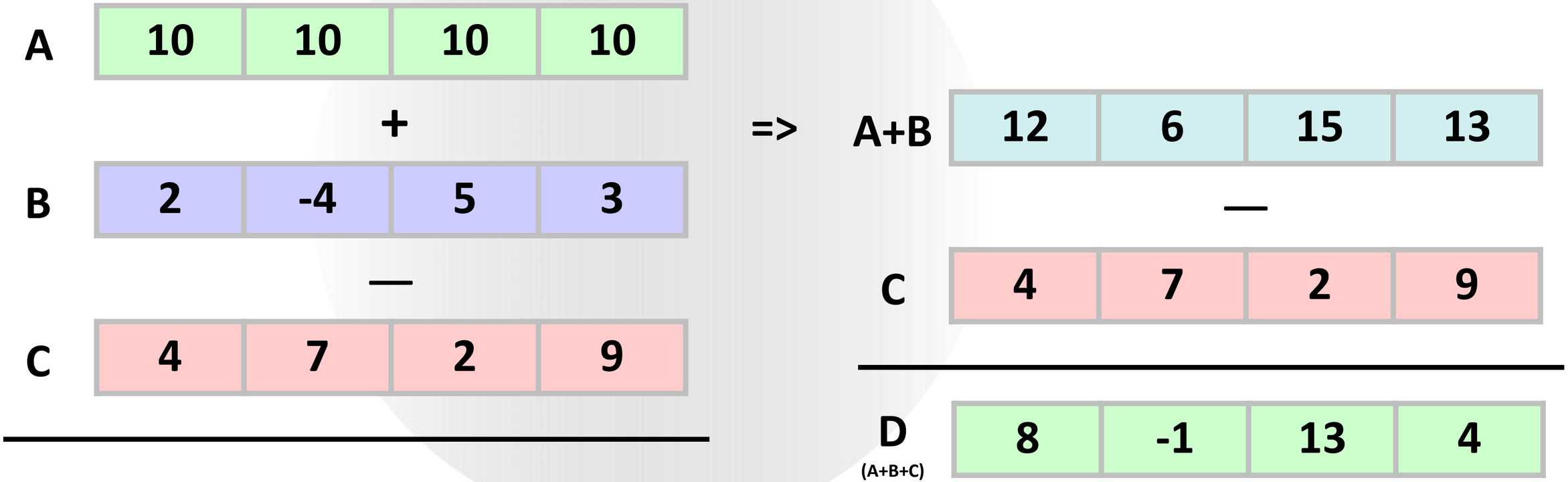


- 대부분의 다중 프로세서 시스템과 다중 컴퓨터 시스템이 이 분류에 속함
- **여러 개**의 CPU들이 **서로 다른 명령어**와 **다른 데이터**를 처리하는 구조
- 슈퍼스칼라를 지원하는 멀티코어 프로세서나 슈퍼컴퓨터들에서 사용.

02 MIMD

Multiple **I**nstruction, **M**ultiple **D**ata
한 번에 데이터 여러 개를 명령어 하나로 처리하는 기법.

<SIMD 연산 예시>



02 MIMD

Multiple **I**nstruction, **M**ultiple **D**ata
한 번에 데이터 여러 개를 명령어 하나로 처리하는 기법.

<SIMD연산 예시>
(10, 10, 10, 10) → (2, -4, 5, 3) → (4, 7, 2, 9)
A + B - C
(12, 6, 15, 13) → (8, -1, 13, 4)

<MIMD 연산 예시>

A	10	10	10	10
	+	-	*	%
B	2	-4	5	3
	-	+	%	*
C	4	7	2	9

=>

α	12	14	50	1
	-	+	%	*
C	4	7	2	9
β	8	13	0	9

=> **여러 개**의 CPU들이 **서로 다른 명령어**와 **다른 데이터**를 처리하는 구조

OPERATING SYSTEM

감사합니다.