리눅스 프로그래밍

Pre-Class
for 리눅스 프로그래밍

인공지능소프트웨어과
이혜정 교수

* 본 교안의 일부 그림과 텍스트는 '운영체제 – 그림으로 배우는 구조와 원리' (개정3판, 한빛미디어)으로 발췌하였음

목차

컴퓨터 하드웨어의 구성

- 프로세서, CPU
- 시스템버스
- 기억장치(메모리), 저장장치
- 입출력 장치

컴퓨터 하드웨어의 구성

- **컴퓨터 시스템**
  - 데이터를 처리하는 물리적인 기계장치인 하드웨어(HW)와 어떤 작업을 지시하는 명령어로 작성한 프로그램인 소프트웨어(SW)로 구성

- **컴퓨터 하드웨어**
  - 프로세서, 메모리(기억장치), 주변장치로 구성되고, 이들은 시스템 버스로 연결

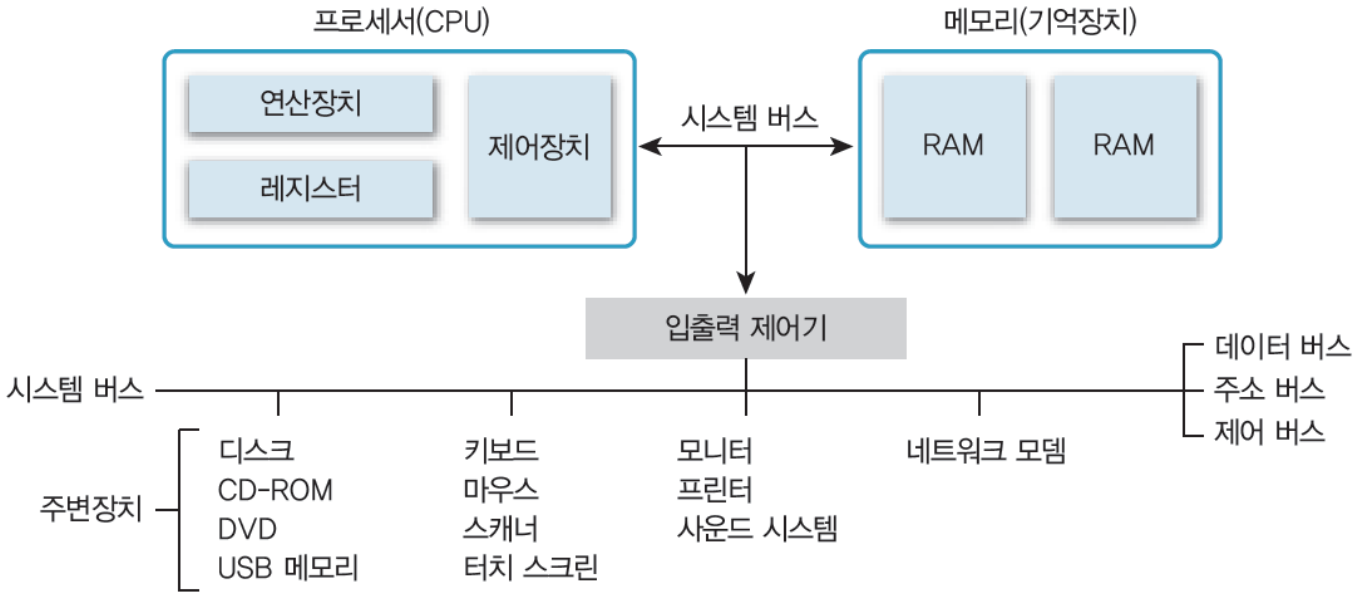
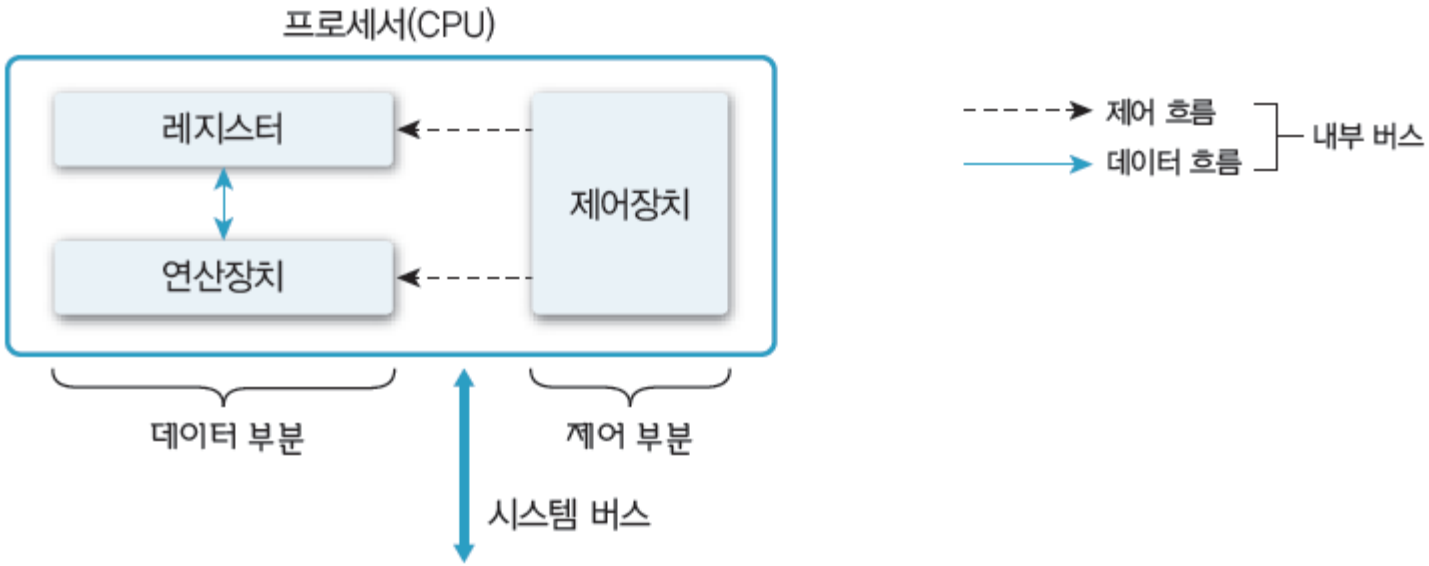


그림 1-1 컴퓨터 하드웨어의 구성

프로세서, CPU

- **프로세서(Processor), CPU(Central Processing Unit, 중앙처리장치)**
  - 컴퓨터를 구성하는 모든 장치의 동작을 제어하고 연산 수행

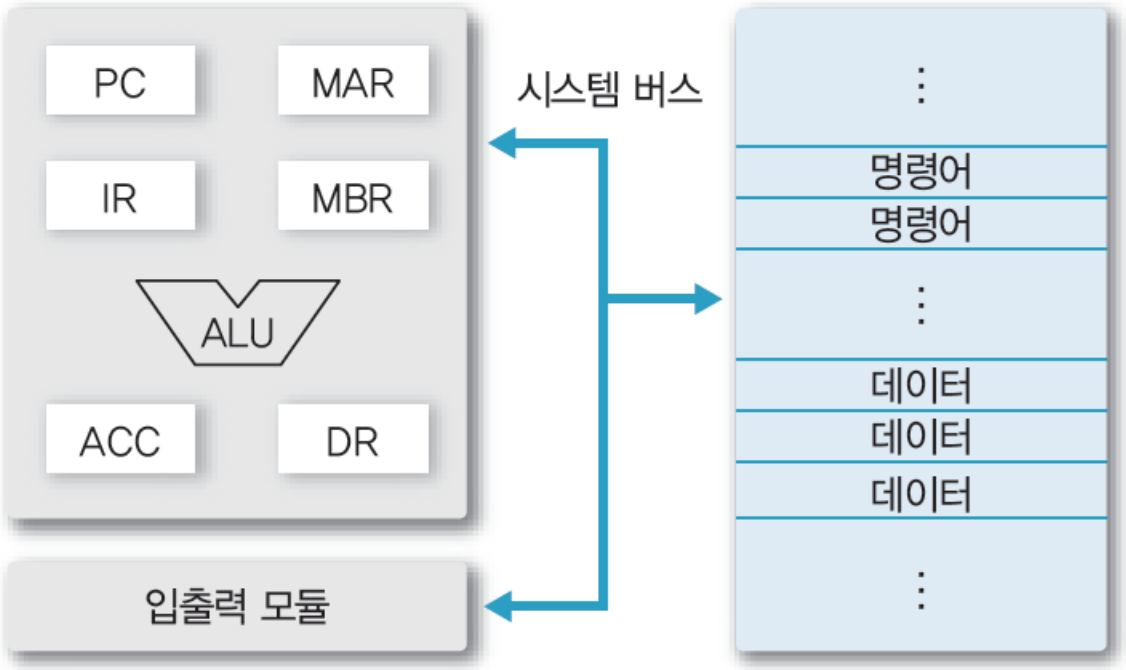


  - 구성요소
    - 레지스터 (Register) : 작업에 필요한 데이터를 CPU 내부에 저장
    - 연산장치 (ALU, Arithmetic & Logic Unit) : 산술 연산 , 논리 연산
    - 제어장치 (CU, Control Unit) : 프로세서에서 작업을 지시

레지스터

- 프로세서 내부에 있으며, 프로세서가 사용할 데이터를 보관하는 가장 빠른 메모리

- 프로세서의 기본 레지스터



• ALU Arithmetic Logic Unit : 산술 · 논리 연산장치

메모리, Memory

- **메모리 계층 구조** 자격증
  - 1950~1960년대 너무 비싼 메인 메모리의 가격 문제 때문에 제안한 방법
  - 메모리를 계층적으로 구성하여 비용, 속도, 용량, 접근시간 등을 상호 보완

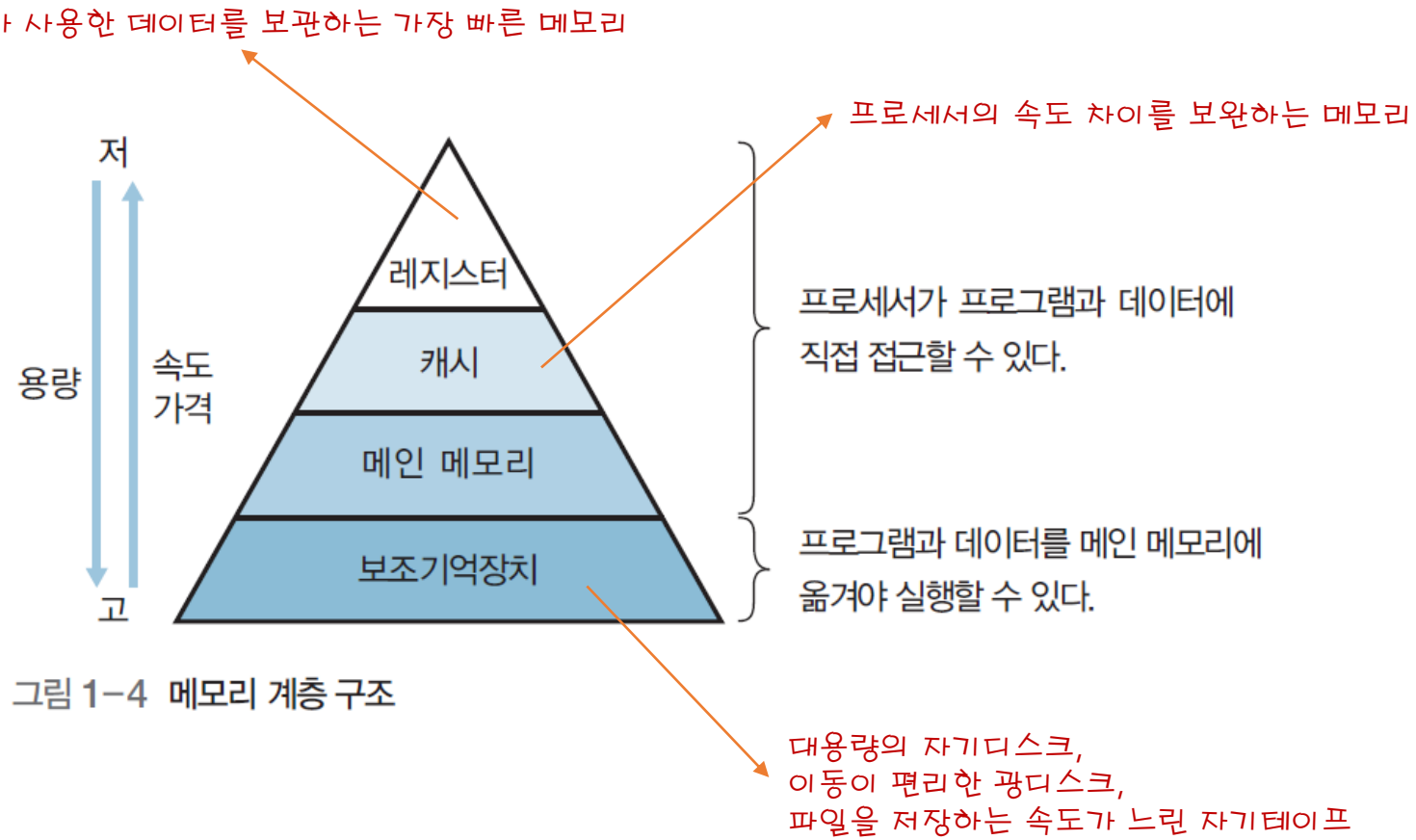
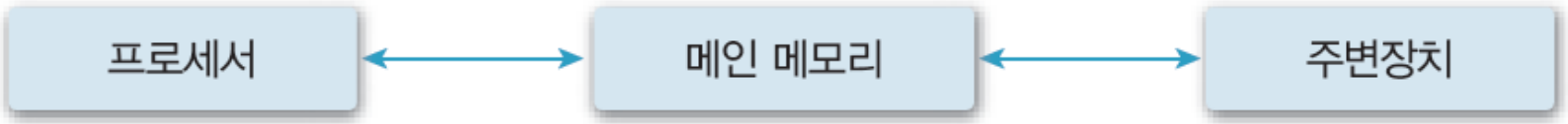


그림 1-4 메모리 계층 구조

메인 메모리

- Main Memory
- 주기억장치

- 프로세서 외부에 있으면서 프로세서에서 수행할 프로그램과 데이터를 저장하거나 프로세서에서 처리한 결과 저장



캐시(Cache)

- **프로세서 내부나 외부에 있으며, 처리 속도가 빠른 프로세서와 상대적으로 느린 메인 메모리의 속도 차이를 보완하는 고속 버퍼**
  - 메인 메모리에서 데이터를 블록 단위로 가져와 프로세서에 워드 단위로 전달하여 속도를 높임

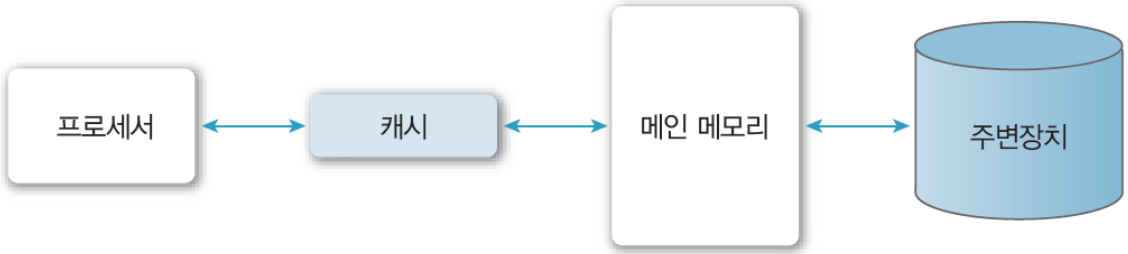


그림 1-9 메인 메모리의 역할 2 : 캐시 추가

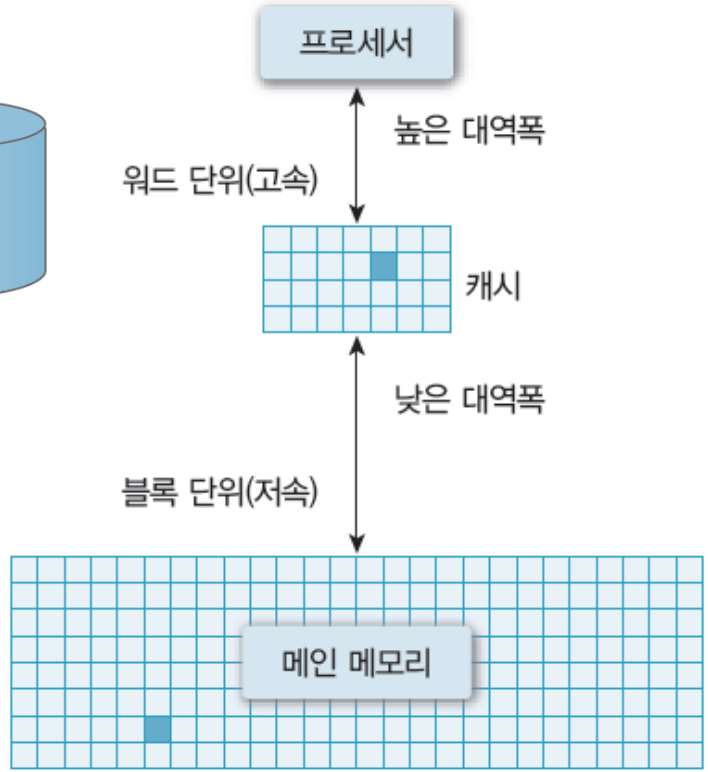
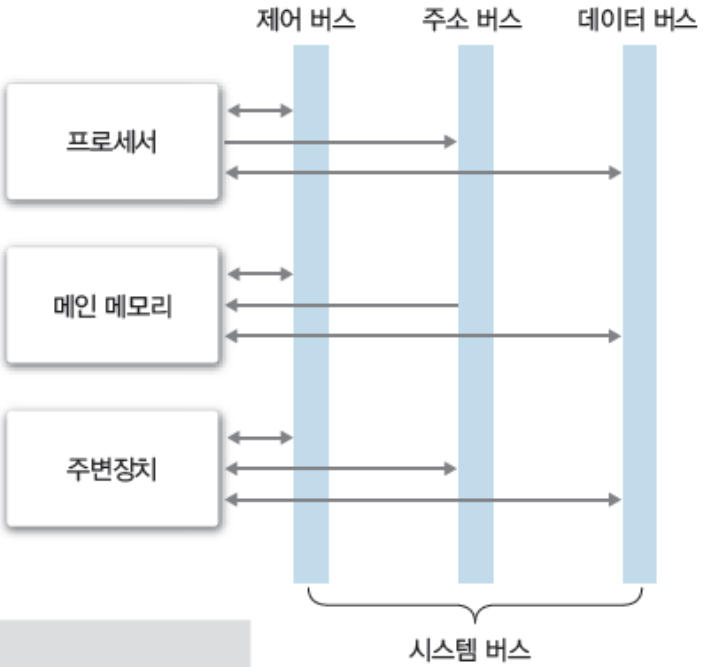


그림 1-10 캐시의 역할

시스템 버스(Bus)

- **하드웨어를 물리적으로 연결하여 서로 데이터를 주고받을 수 있게 하는 통로**
  - 컴퓨터 내부의 다양한 신호(데이터 입출력 신호, 프로세서 상태 신호, 인터럽트 요구와 허가 신호, 클록clock 신호 등)를 시스템 버스로 전달



- **종류 : 데이터 버스, 주소 버스, 제어 버스**

표 1-3 시스템 버스의 종류

종류	설명
데이터 버스	프로세서와 메인 메모리, 주변장치 사이에서 데이터를 전송한다. 데이터 버스를 구성하는 배선 수는 프로세서가 한 번에 전송할 수 있는 비트 수를 결정하는데, 이를 워드라고 한다.
주소 버스	프로세서가 시스템의 구성 요소를 식별하는 주소 정보를 전송한다. 주소 버스를 구성하는 배선 수는 프로세서와 접속할 수 있는 메인 메모리의 최대 용량을 결정한다.
제어 버스	프로세서가 시스템의 구성 요소를 제어하는 데 사용한다. 제어 신호로 연산장치의 연산 종류와 메인 메모리의 읽기나 쓰기 동작을 결정한다.

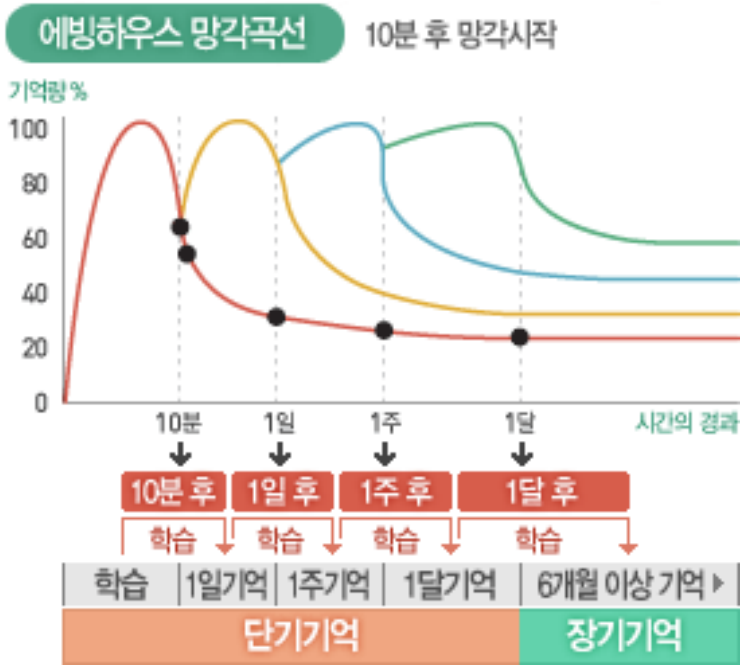
주변 장치

- **저장장치 (보조기억장치)**
  - 메인 메모리와 달리 거의 영구적으로 데이터를 저장하는 장치
  - 데이터를 입력하여 저장하며, 저장한 데이터를 출력하는 공간이므로 입출력장치에 포함하기도 함

- **입력장치**
  - 컴퓨터에서 처리할 데이터를 외부에서 입력하는 장치

- **출력장치**
  - 입력장치와 반대로 컴퓨터에서 처리한 데이터를 외부로 보내는 장치

망각 곡선 되살리기


에빙하우스 망각곡선
10분 후 망각시작
기억량 %
100
80
60
40
20
0
10분
1일
1주
1달
시간의 경과
10분 후
1일 후
1주 후
1달 후
학습
학습
학습
학습
학습
1일기억
1주기억
1달기억
6개월 이상 기억
단기기억
장기기억

컴퓨터 시스템의 동작

- 작업 처리 순서
- 명령어 구조 및 실행 방식
- 인터럽트 동작 방식

컴퓨터의 작업 처리 순서

- **컴퓨터 시스템으로 작업을 처리할 때는 다음 순서에 따라 동작, 제어장치(CU)가 이 동작을 제어**
  - ❶ 입력장치로 정보를 입력받아 메모리에 저장한다.
  - ❷ 메모리에 저장한 정보를 프로그램 제어에 따라 인출하여 연산장치에서 처리한다.
  - ❸ 처리한 정보를 출력장치에 표시하거나 보조기억장치에 저장한다.

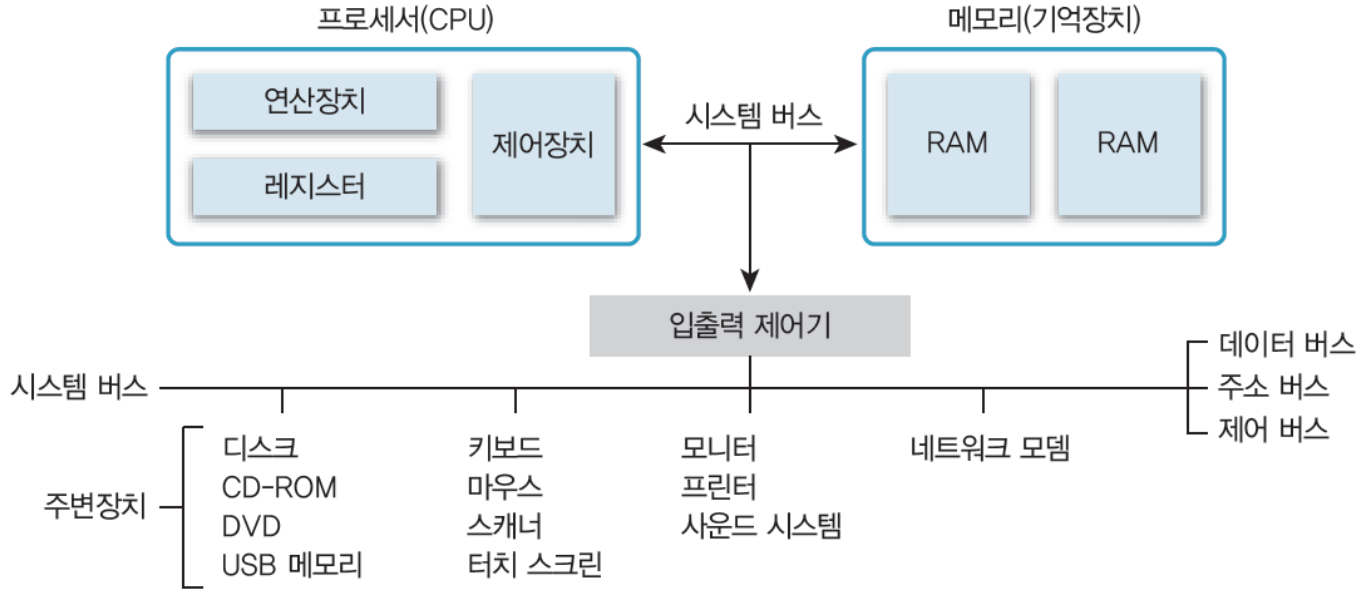


그림 1-1 컴퓨터 하드웨어의 구성

명령어의 구조

▪ 명령어의 기본 구조

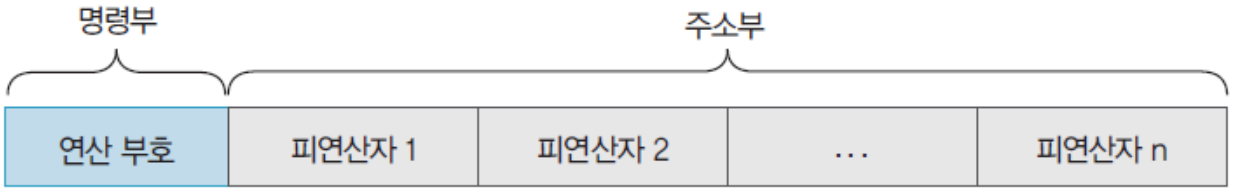


그림 1-13 명령어의 기본 구조

- 연산 부호OPcode, OPeration code (오피코드)
  - 프로세서가 실행할 동작인 연산 지정
  - 산술 연산(+, -, *, /), 논리 연산(AND, OR, NOT), 시프트shift, 보수 등 연산 정의
  - 연산 부호가 n비트이면 최대 2n개 연산이 가능
- 피연산자operand (오퍼랜드)
  - 연산할 데이터 정보 저장
  - 데이터는 레지스터나 메모리, 가상 기억장치, 입출력장치 등에 위치할 수 있는데 보통 데이터 자체보다는 데이터의 위치 저장

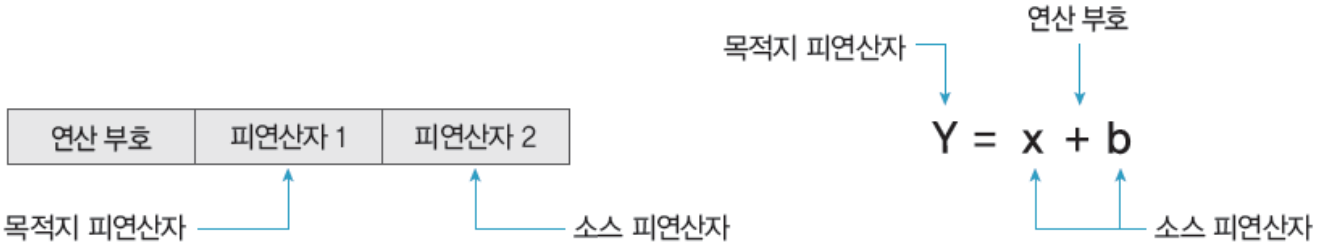


그림 1-14 소스 피연산자와 목적지 피연산자

명령어의 실행

▪ 명령어의 실행 과정

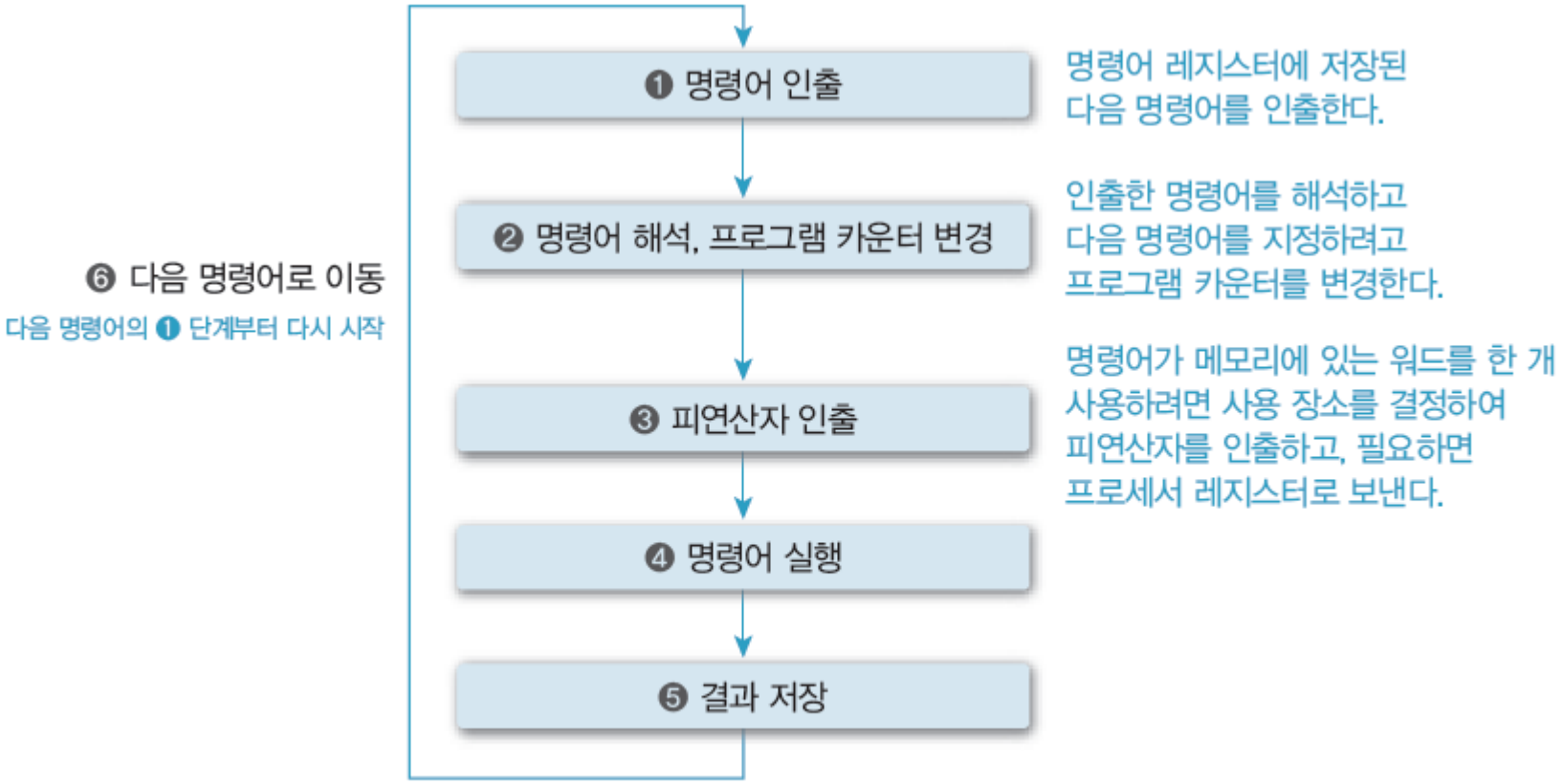
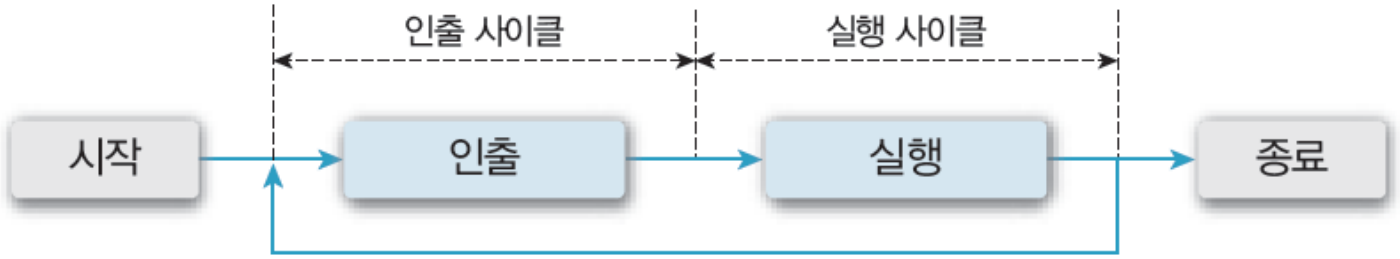


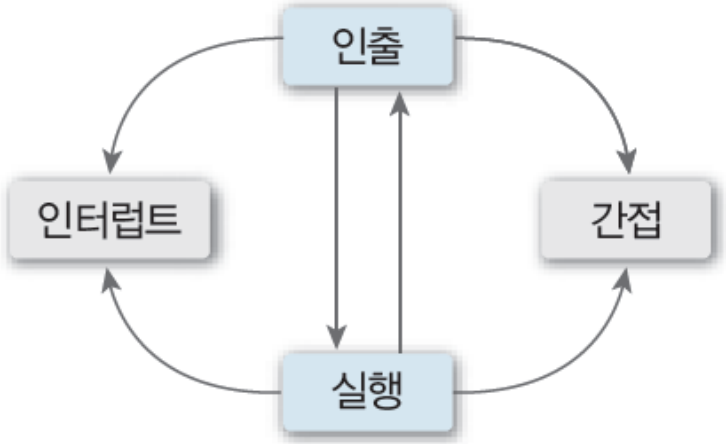
그림 1-18 명령어 실행 과정

명령어의 실행

▪ 명령어의 실행 사이클



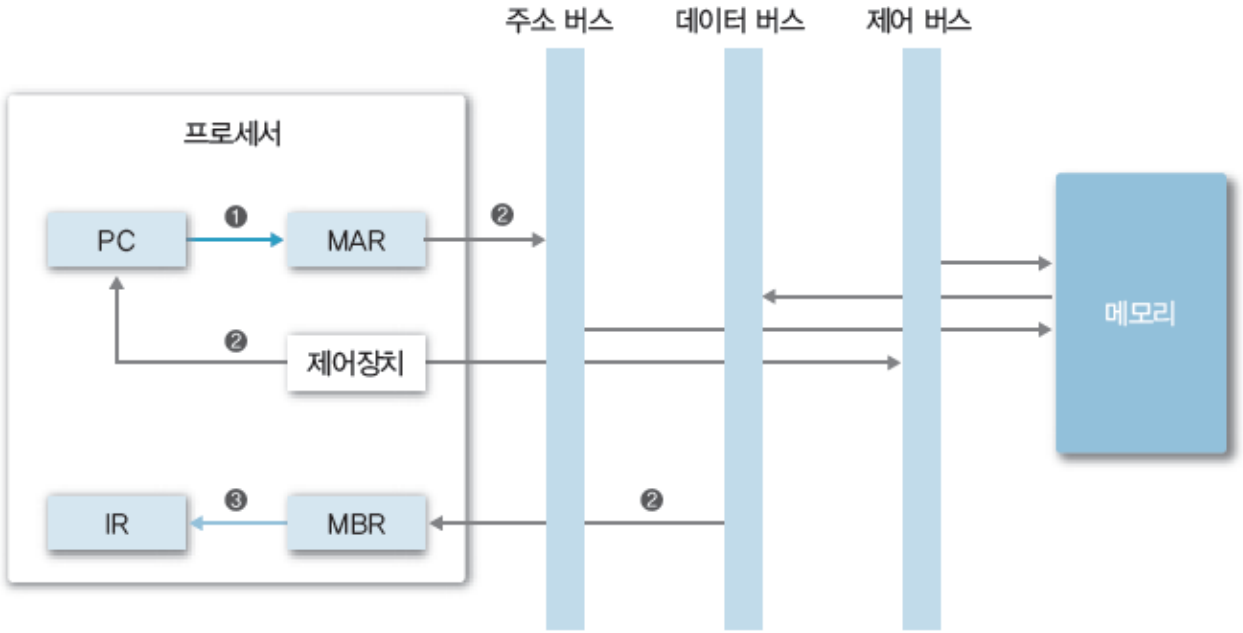
(a) 일반적인 명령어 사이클



(b) 세분화된 명령어 사이클

그림 1-19 명령어 실행 사이클

명령어의 실행 – 인출 사이클

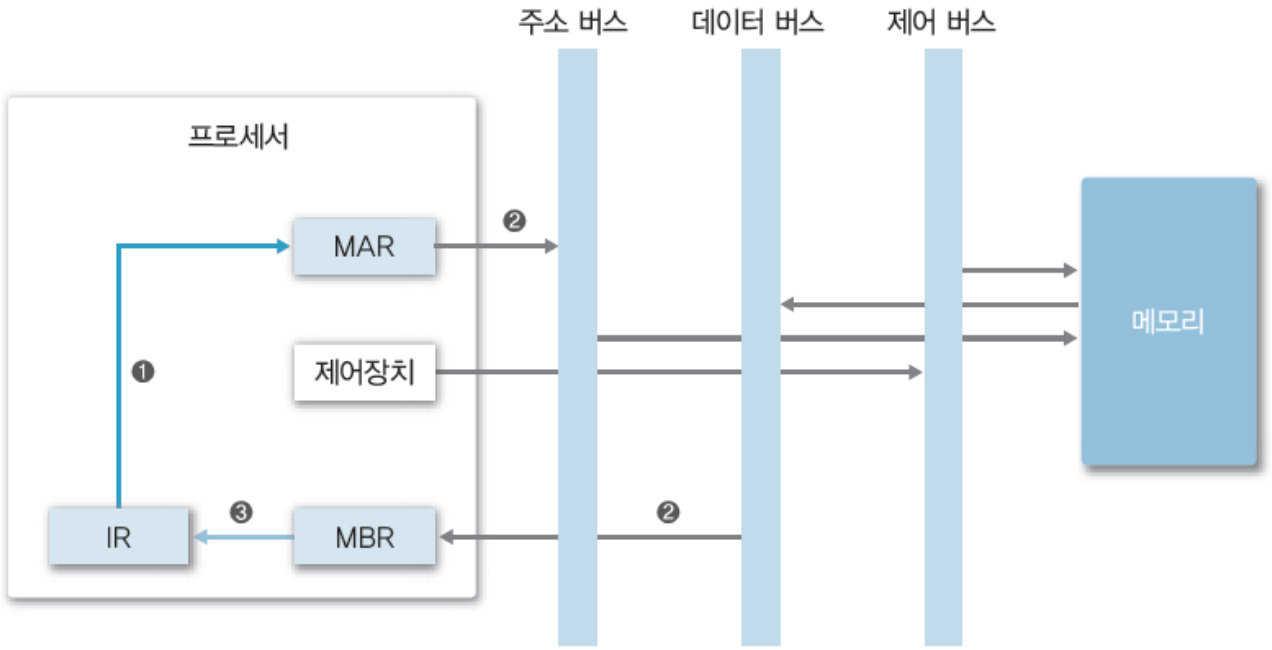


시간	레지스터 동작	설명
❶	PC → MAR	PC에 저장된 주소를 프로세서 내부 버스를 이용하여 MAR에 전달한다.
❷	Memory^{MAR} → MBR	MAR에 저장된 주소에 해당하는 메모리 위치에서 명령어를 인출한 후 이 명령어를 MBR에 저장한다. 이때 제어장치는 메모리에 저장된 내용을 읽도록 제어 신호를 발생시킨다.
	PC + 1 → PC	다음 명령어를 인출하려고 PC를 증가시킨다.
❸	MBR → IR	MBR에 저장된 내용을 IR에 전달한다.

- PC : 프로그램 카운터
- MBR : 메모리 버퍼 레지스터
- MAR : 메모리 주소 레지스터
- IR : 명령어 레지스터

그림 1-20 인출 사이클 과정

명령어의 실행 – 간접 사이클

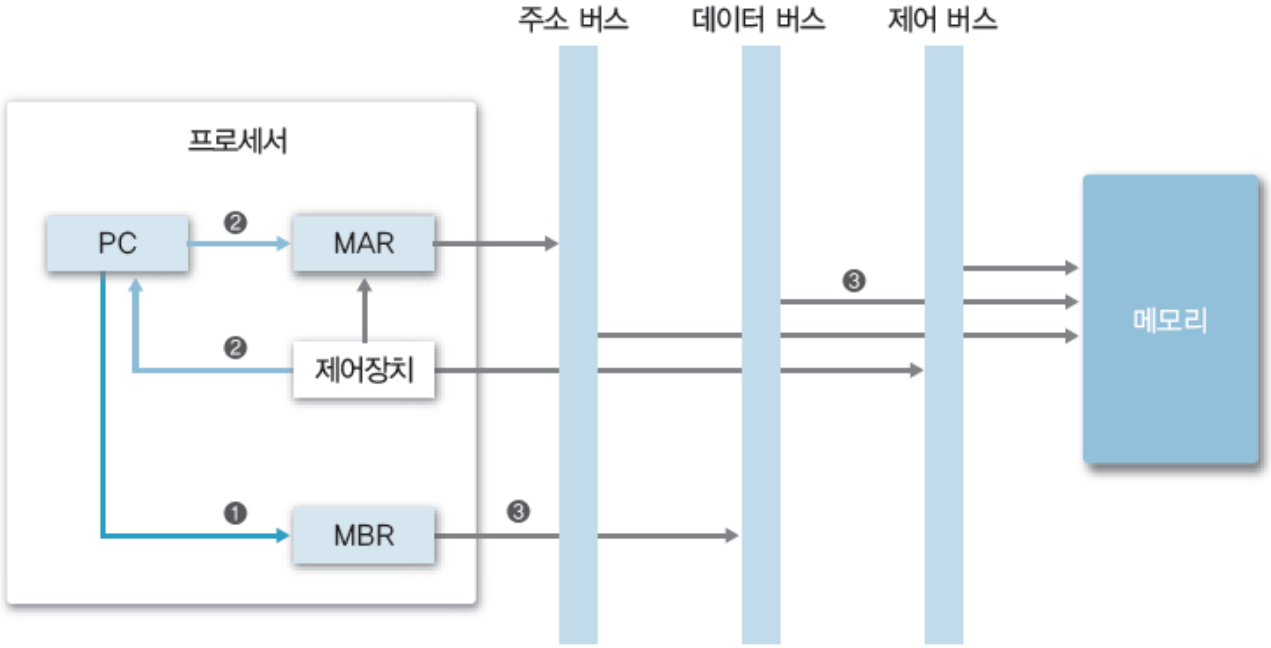


시간	레지스터 동작	설명
❶	$IR^{addr} \rightarrow MAR$	IR에 저장된 명령어의 피연산자(주소부)를 MAR에 전달한다.
❷	$Memory^{MAR} \rightarrow MBR$	MAR에 저장된 주소에 해당하는 메모리 위치에서 데이터를 인출한 후 이 데이터를 MBR에 저장한다. 이때 제어장치는 메모리에 저장된 내용을 읽도록 제어 신호를 발생시킨다.
❸	$MBR \rightarrow IR^{addr}$	MBR에 저장된 내용을 IR에 전달한다.

- IR : 명령어 레지스터
- MAR : 메모리 주소 레지스터
- MBR : 메모리 버퍼 레지스터

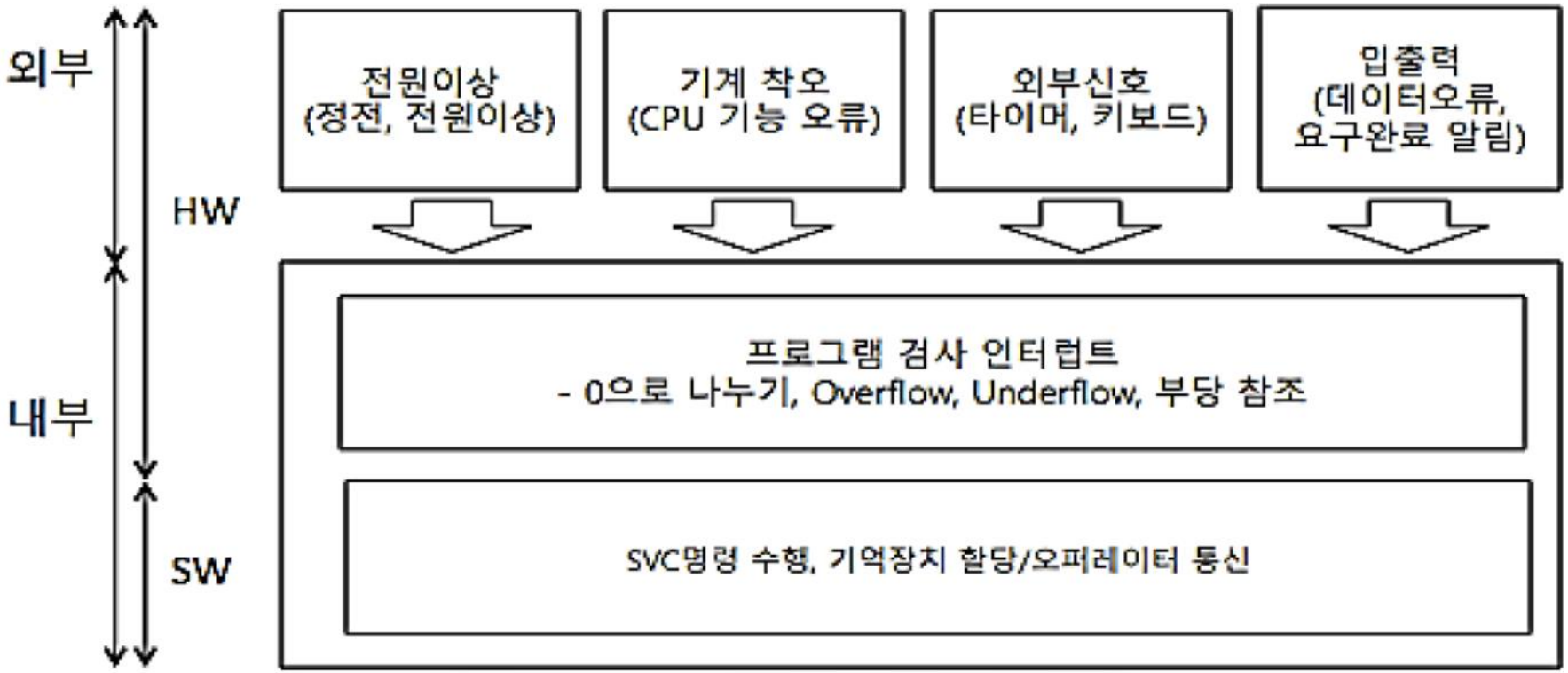
명령어의 실행 – 인터럽트 사이클

- **인터럽트**
  - 현재 실행 중인 프로그램을 중단하고 다른 프로그램의 실행을 요구하는 명령어

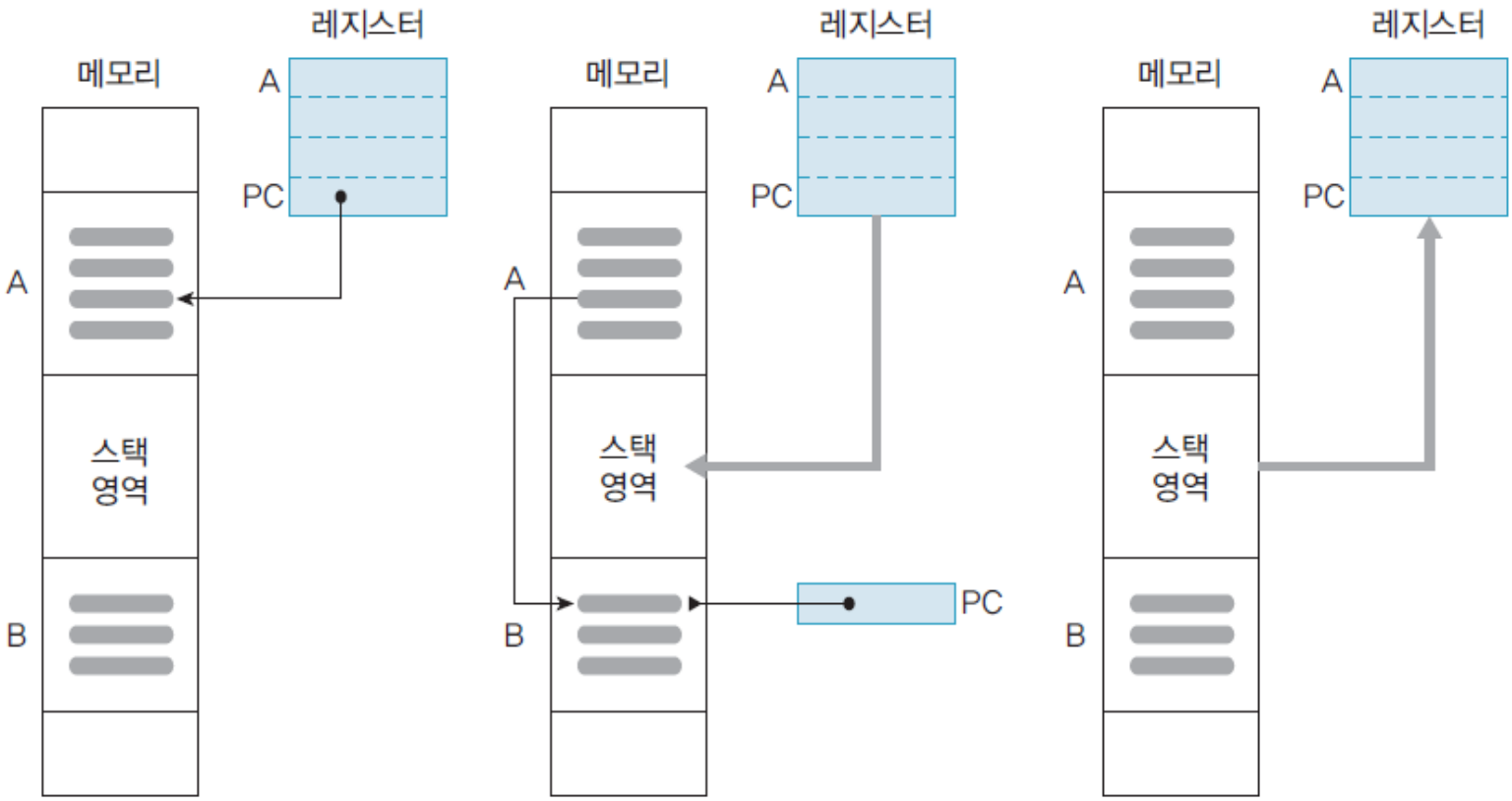


시간	레지스터 동작	설명
❶	PC → MBR	PC의 내용을 MBR에 저장한다.
❷	IntRoutine_Address → PC	인터럽트 루틴 주소를 PC에 저장한다.
	Save_Address → MAR	PC에 저장된 인터럽트 루틴 주소를 MAR에 저장한다.
❸	MBR → Memory^{MAR}	MBR의 주소에 있는 내용을 지시된 메모리 셀로 이동한다.

인터럽트 유형


외부
내부
HW
SW
전원이상
(정전, 전원이상)
기계 착오
(CPU 기능 오류)
외부신호
(타이머, 키보드)
입출력
(데이터오류,
요구완료 말림)
프로그램 검사 인터럽트
- 0으로 나누기, Overflow, Underflow, 부당 참조
SVC명령 수행, 기억장치 할당/오퍼레이터 통신

인터럽트 처리 과정



(a) 인터럽트 발생 전

프로그램 A를 실행, 프로그램 카운터PC는 현재 명령어를 가리킴

(b) 인터럽트 발생

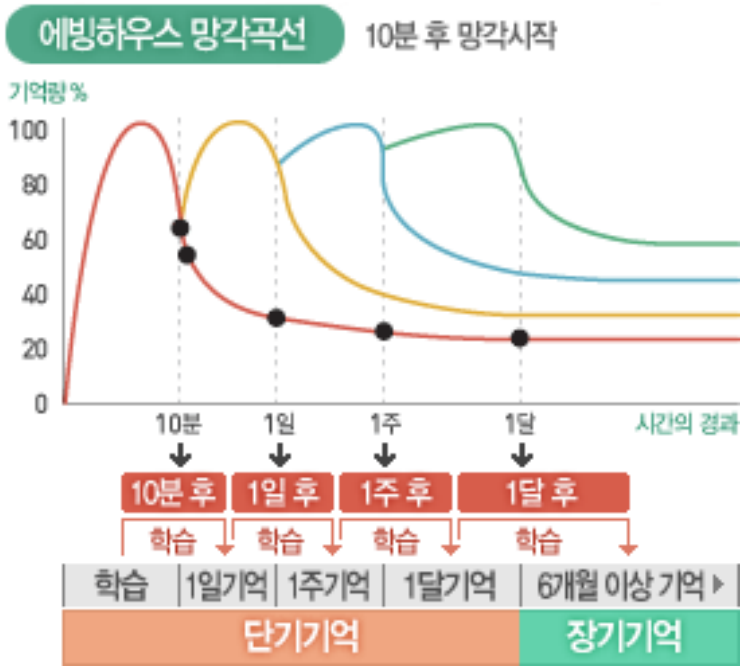
현재 명령어를 종료, 레지스터의 모든 내용을 스택 영역(또는 프로세스 제어 블록)에 보내고 pc에는 인터럽트 처리 프로그램(프로그램 B)의 시작 위치를 저장하고 제어를 넘긴

(c) 인터럽트 처리 후

스택 영역에 있던 내용을 레지스터에 다시 저장하며, 프로그램 A가 다시 시작하는 위치를 저장하고 중단했던 프로그램 A를 재실행

그림 1-24 인터럽트 처리 과정

망각 곡선 되살리기

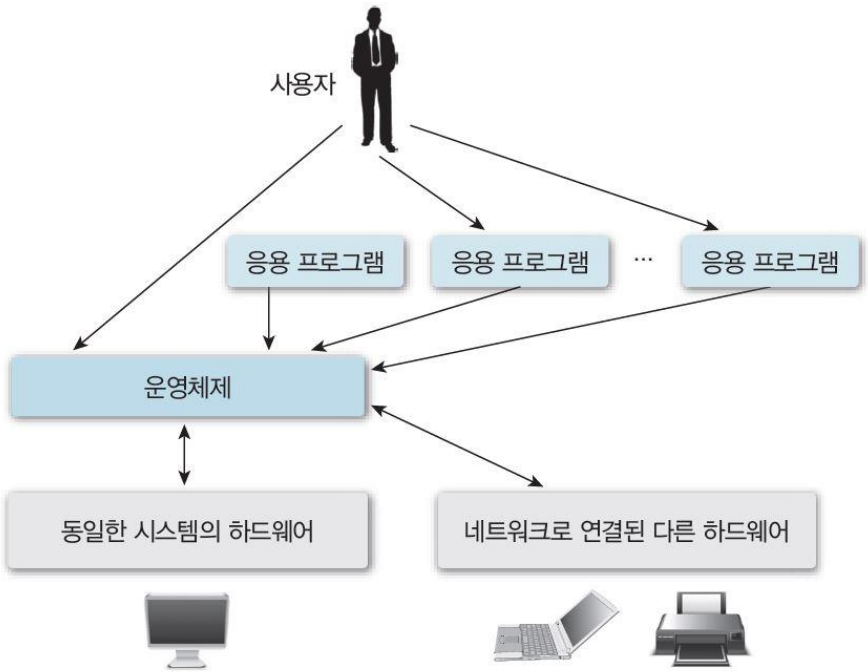

에빙하우스 망각곡선
10분 후 망각시작
기억량 %
100
80
60
40
20
0
10분
1일
1주
1달
시간의 경과
10분 후
1일 후
1주 후
1달 후
학습
학습
학습
학습
학습
1일기억
1주기억
1달기억
6개월 이상 기억
단기기억
장기기억

운영체제 개요

- 운영체제의 정의
- 운영체제의 기능
- 운영체제의 역할
- 운영체제의 목적

운영체제의 정의 (1)

- **사용자 및 응용 프로그램와 하드웨어 사이의 중간 매개체로 동작하는 시스템 소프트웨어**
  - 사용자가 응용 프로그램을 실행할 수 있는 기반 환경을 제공하여 컴퓨터를 편리하게 사용할 수 있도록 도와주고,
    하드웨어를 효율적으로 사용할 수 있도록 다양한 기능을 제공하는 소프트웨어
  - 사용자가 하드웨어에 접근할 수 있는 유일한 수단



운영체제의 정의 (2)

- **응용 프로그램의 실행을 제어하고, 자원을 할당 및 관리하며, 입출력 제어 및 데이터 관리와 같은 서비스를 제공하는 소프트웨어**
  - 하드웨어 및 사용자, 응용 프로그램, 시스템 프로그램 사이에서 인터페이스를 제공
  - 프로세서, 메모리, 입출력장치, 통신장치 등 컴퓨터 자원을 효과적으로 활용하려고 조정·관리
  - 메일 전송, 파일 시스템 검사, 서버 작업 등 높은 수준의 서비스를 처리하는 응용 프로그램을 제어
  - 다양한 사용자에게서 컴퓨터 시스템을 보호하려고 입출력을 제어하며 데이터를 관리

운영체제의 기능

그림 2-4 운영체제의 기능

운영체제의 역할

- **자원 관리**
  - 컴퓨터 시스템 자원을 응용 프로그램에 나눠주어 사용자가 원활히 작업하게 함
  - 자원 요청한 프로그램이 여러 개면 적당한 순서로 자원 배분하고 적절한 시점에 자원 회수하여 다른 응용 프로그램에 나눠줌
- **자원 보호**
  - 비정상적 작업으로부터 컴퓨터 자원 보호
- **하드웨어 인터페이스 제공**
  - 사용자가 복잡한 과정 없이 다양한 장치를 사용할 수 있게 하드웨어 인터페이스 제공
  - CPU, 메모리, 키보드, 마우스 같은 다양한 하드웨어를 일관된 방법으로 사용하도록 지원
- **사용자 인터페이스 제공**
  - 사용자가 운영체제를 편리하게 사용하도록 지원

    예) 윈도우의 그래픽 사용자 인터페이스(GUI)

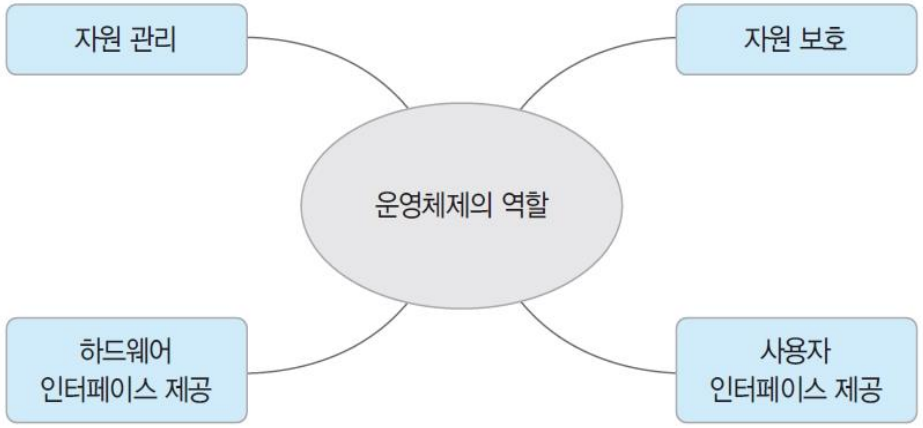


그림 1-6 운영체제의 역할

운영체제의 목표

- **효율성**
  - 같은 자원으로 더 많은 작업량 처리하거나, 같은 작업량 처리하는 데 더 적은 자원 사용
- **안정성**
  - 사용자와 응용 프로그램의 안전 문제와 하드웨어적인 보안 문제 처리
  - 시스템에 문제 발생시 이전으로 복구하는 결함 포용 기능 수행
- **확장성**
  - 다양한 시스템 자원을 컴퓨터에 추가/제거하기 편리한 것
- **편리성**
  - 사용자가 편리하게 작업할 수 있는 환경 제공하는 것

운영체제의 목표 (발전 목적)

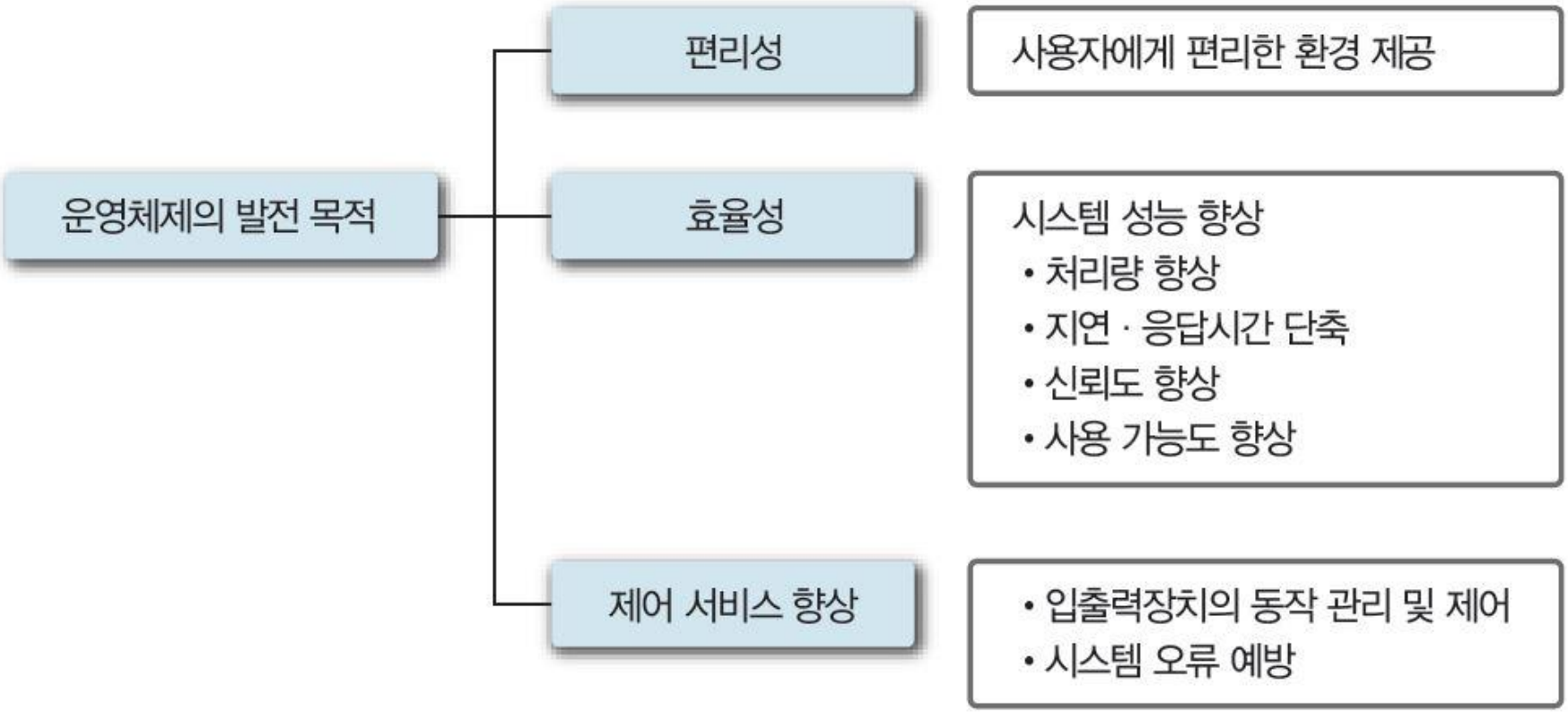
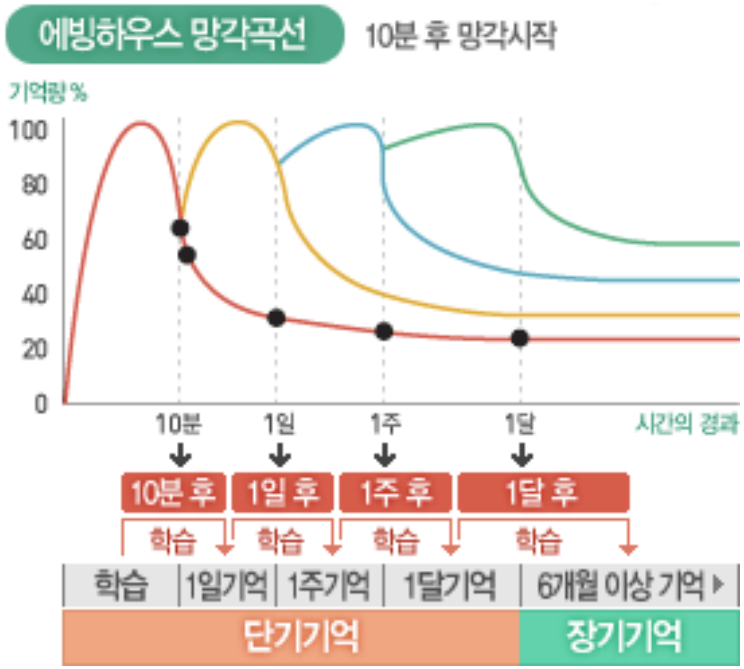


그림 2-3 운영체제의 발전 목적

망각 곡선 되살리기


에빙하우스 망각곡선
10분 후 망각시작
기억량 %
100
80
60
40
20
0
10분
1일
1주
1달
시간의 경과
10분 후
1일 후
1주 후
1달 후
학습
학습
학습
학습
학습
1일기억
1주기억
1달기억
6개월 이상 기억
단기기억
장기기억

운영체제 유형

운영체제 유형

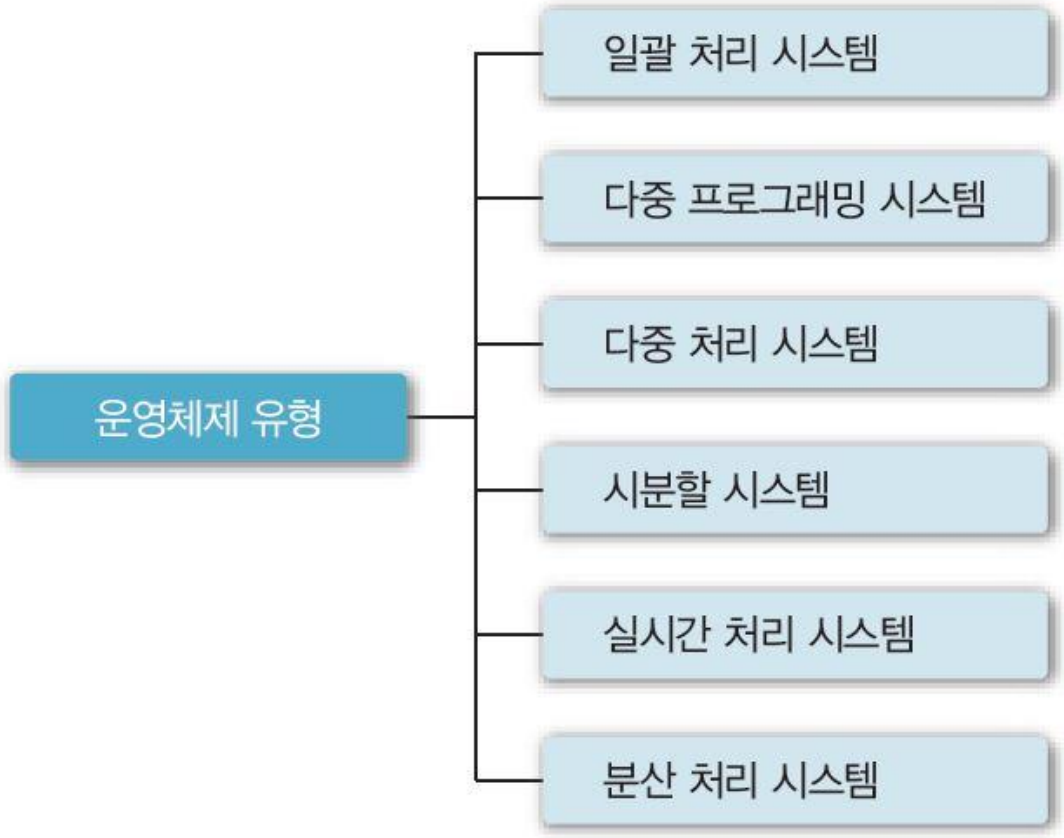


그림 2-11 운영체제의 유형

다중 프로그래밍 시스템 (1)

- **프로세스가 다른 작업 수행 시 입출력 작업 불가능하여 프로세서와 메인 메모리의 활용도 떨어지는 일괄 처리 시스템의 큰 문제를 다중 프로그래밍 도입하여 해결**
  - 프로세서가 유휴 상태일 때 실행 중인 둘 이상의 작업이 프로세서를 전환 (인터리빙)하여 사용할 수 있도록 동작

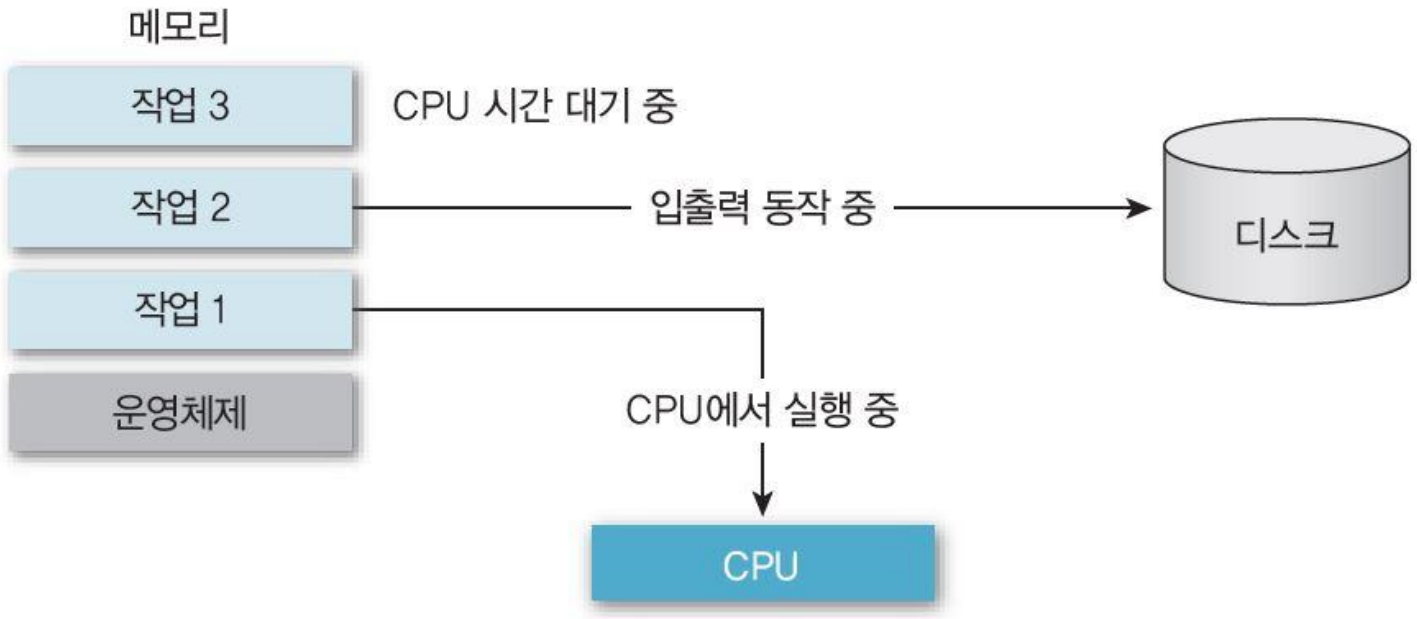


그림 2-12 다중 프로그래밍 시스템

다중 프로그래밍 시스템 (2)

- **다중 프로그래밍의 특징**
  - 높고 효율적인 프로세서 사용률(효율적인 운영) 증가
  - 많은 사용자의 프로그램이 거의 동시에 프로세서를 할당받는 듯한 느낌
  - 다중 프로그래밍 운영체제는 아주 복잡
  - 여러 작업을 준비 상태로 두려면 이를 메모리에 보관, 일정 형태의 메모리를 관리해야 함
  - 여러 작업이 수행할 준비를 갖추고 있으면, 이 중 하나를 선택하는 결정 방법 필요 (인터럽트 이용 수행하는 프로세서 스케줄링의 다중 프로그래밍으로, 현재 운영체제의 중심 주제)

그림 2-13 다중 프로그래밍 시스템의 처리 방법 예

시분할 시스템TSS, Time Sharing System (1)

- **여러 사용자에게 짧은 간격으로 프로세서 번갈아 할당, 마치 자기 혼자 프로세서를 독점하고 있는 양 착각하게 하여 여러 사용자가 단일 컴퓨터 시스템을 동시 사용 가능**
  - 다중 프로그래밍을 논리적으로 확장한 개념, 프로세서가 다중 작업을 교대로 수행
  - 다수의 사용자가 동시에 컴퓨터의 자원을 공유할 수 있는 기술
  - 1970년 초까지는 시분할 시스템 만들기가 아주 어렵고 비용도 많이 들어 일반화 못함

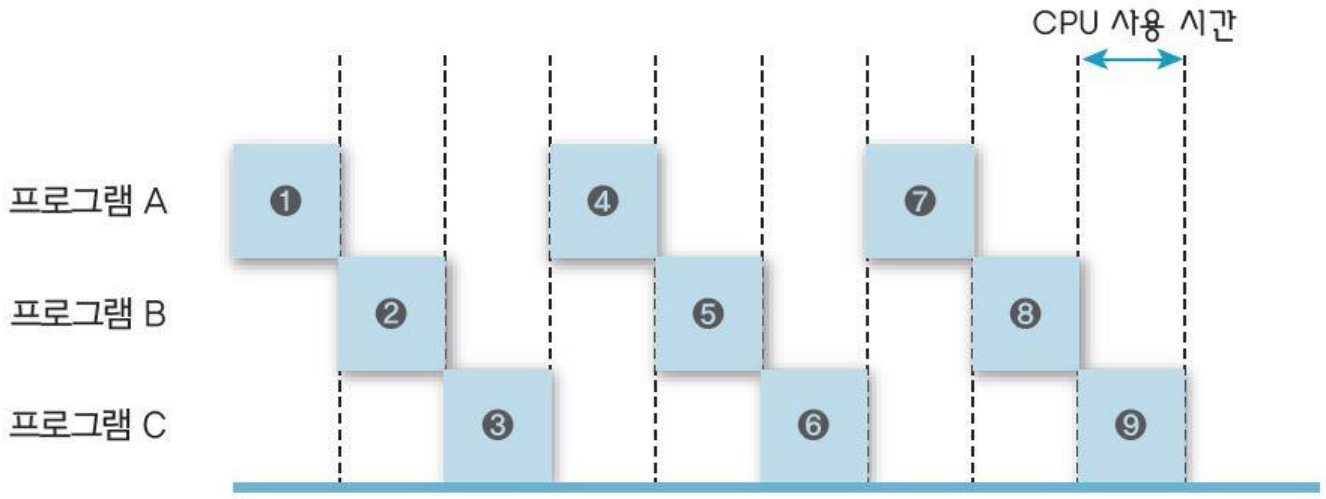


그림 2-14 시분할 시스템의 처리 방법 예

시분할 시스템TSS, Time Sharing System (2)

- **다중 프로그래밍 시스템과 시분할 시스템 특징**
  - 메모리에 여러 프로그램을 적재하므로 메모리 관리 필요
  - 어떤 프로그램을 먼저 실행할지 결정하는 스케줄링 개념 필요
  - 다중 프로그래밍 시스템의 목표 : 프로세서 사용 최대화
  - 시분할 시스템의 목표 : 응답시간 최소화

표 2-2 시분할 시스템의 장점과 단점

장점	• 빠른 응답 제공 • 소프트웨어의 중복 회피 가능 • 프로세서 유휴시간 감소
단점	• 신뢰성 문제 • 보안 의문 및 사용자 프로그램과 데이터의 무결성 • 데이터 통신의 문제

다중 처리multiprocessing 시스템

- **단일 컴퓨터 시스템 내에서 둘 이상의 프로세서 사용, 동시에 둘 이상의 프로세스 지원**
  - 여러 프로세서와 시스템 버스, 클록, 메모리와 주변장치 등 공유
  - 빠르고, 프로세서 하나가 고장 나도 다른 프로세서 사용하여 작업 계속, 신뢰성 높음
  - 프로세서 간의 연결, 상호작용, 역할 분담 등을 고려해야 함
  - 다중 처리 시스템을 구성하는 방법에는 비대칭(주종)적 구성과 대칭적 구성이 있음

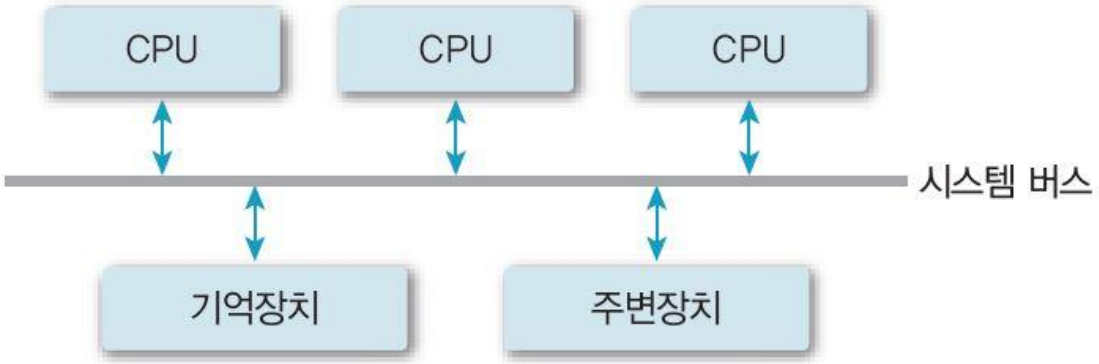


그림 2-15 다중 처리 시스템

실시간 처리 시스템real time processing system

- **요청 사항이 있을 때 즉시 응답하는 시스템**
  - 반응(응답)시간 : 입력 및 업데이트된 정보 요구 처리 후 디스플레이에 응답하는 시스템에 소요 시간
    - 프로세서에 이미 고정(반응시간이 온라인 처리에 비해 매우 짧음)
    - 고정 시간 제약을 잘 정의하지 않으면 시스템 실패.

- **실시간 처리 시스템의 두 가지 유형**
  - 경성 실시간 처리 시스템hard real time processing system
    - 작업의 실행 시작이나 완료에 대한 시간 제약 조건을 지키지 못할 때 시스템에 치명적인 영향을 주는 시스템
    - 무기 제어, 발전소 제어, 철도 자동 제어, 미사일 자동 조준 등이 이에 해당
    - 보장되는 컴퓨팅, 시간의 정확성과 컴퓨팅 예측성을 갖게 해야 함
  - 연성 실시간 처리 시스템soft real time processing system
    - 작업 실행에서 시간 제약 조건은 있으나, 이를 지키지 못해도 전체 시스템에 치명적인 영향을 미치지 않는 시스템
    - 동영상은 초당 일정 프레임frame 이상의 영상을 재생해야 한다는 제약이 있으나, 일부 프레임을 건너뛰어도 동영상을 재생 시스템에는 큰 영향을 미치지 않음

분산 처리 시스템distributed processing system

- **하나의 프로그램을 여러 프로세서에 분산하여 동시에 실행**
  - 사용자에게는 중앙집중식 시스템처럼 보이는데, 다수의 독립된 프로세서에서 실행
  - 시스템마다 독립적인 운영체제와 메모리로 운영, 필요 시 통신하는 시스템
  - 데이터를 여러 위치에서 처리·저장, 여러 사용자가 공유

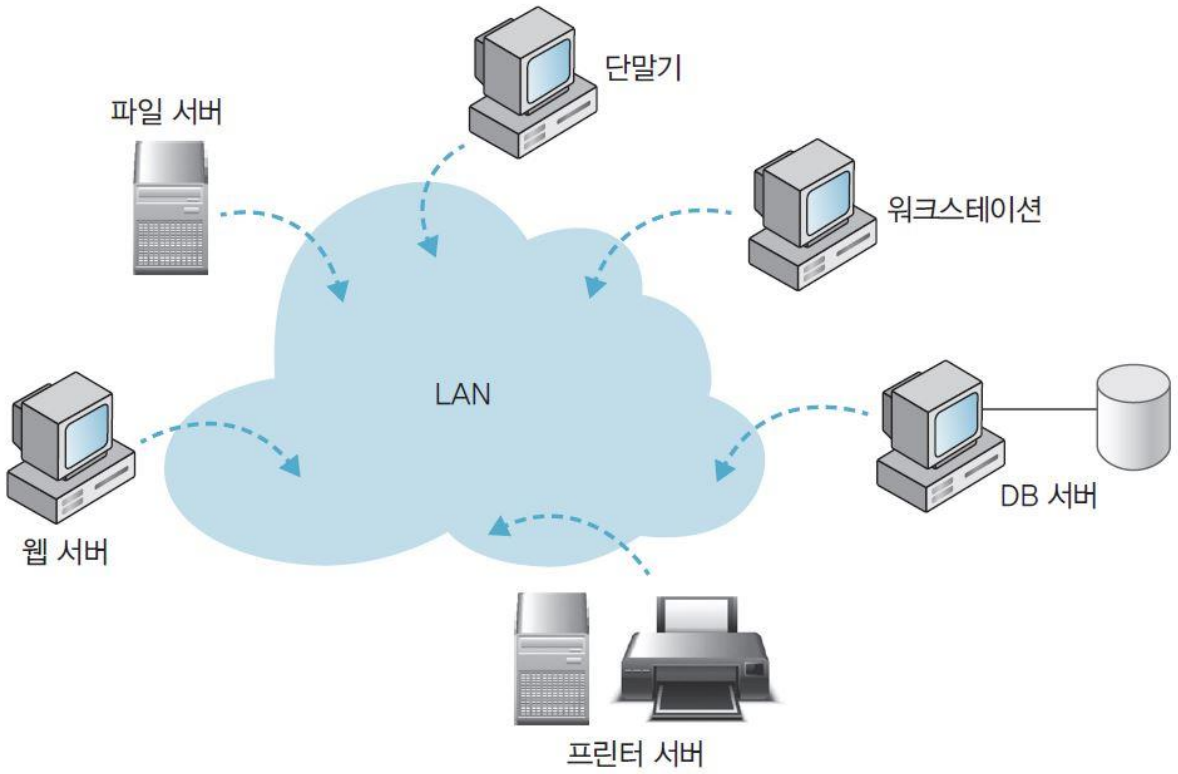
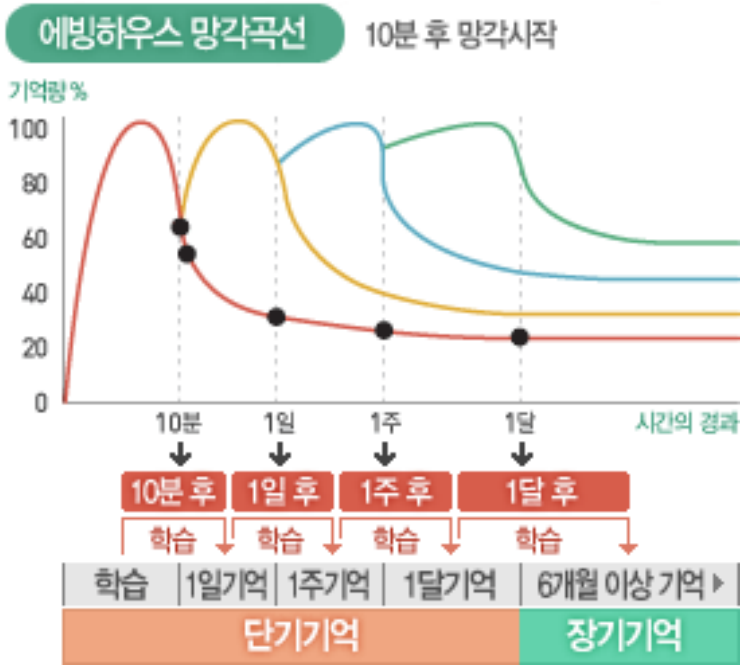


그림 2-16 분산 처리 시스템

망각 곡선 되살리기


에빙하우스 망각곡선
10분 후 망각시작
기억량 %
100
80
60
40
20
0
10분
1일
1주
1달
시간의 경과
10분 후
1일 후
1주 후
1달 후
학습
학습
학습
학습
학습
1일기억
1주기억
1달기억
6개월 이상 기억
단기기억
장기기억