

풀스택 양성과정

컴퓨터 기초 교육

Chap1. 운영체제

- 운영체제의 정의
- 운영체제의 역할과 목표
- 운영체제의 운용기법 및 컴퓨터 발달 과정
- 시스템 소프트웨어의 구성 및 분류(링커/로더/컴파일러/어셈블러)
- 스레드
- 프로세스 스케줄링
- 교착상태 발생 및 대책
- 기억장치 계층 구조 및 기억장치 관리 전략
- UNIX 시스템
- 분산처리 시스템

1. 운영체제의 정의

- 운영체제의 정의(OS, Operating System)

- 일반 컴퓨터, 노트북, 스마트폰의 전원을 켜면 가장 먼저 만나게 되는 소프트웨어
- H/W를 제어하는 S/W(하드웨어를 효율적으로 사용할 수 있도록 하는 프로그램)
- 응용 프로그램이나 사용자에게 컴퓨터 자원을 사용할 수 있는 인터페이스를 제공하고 그 결과를 돌려주는 시스템 소프트웨어
- 컴퓨터 시스템의 자원(프로세스, 기억장치, 파일 및 정보, 네트워크 및 보호)들을 효율적으로 관리
- 사용자가 컴퓨터를 편리하고 효율적으로 사용할 수 환경 제공



2. 운영체제의 역할 및 목표

- 운영체제의 역할

- 다른 응용 프로그램이 유용한 작업을 할 수 있도록 환경 마련
- 프로세서(CPU), 기억장치, 입출력 장치, 파일등의 **자원관리 및 자원 보호**
- 자원관리를 위해 **자원의 스케줄링** 기능을 제공(여러 사용자가 자원을 효율적으로 나누어 사용)
- 사용자와 시스템 간의 **편리한 인터페이스** 제공(내/외부 인터페이스 제공)
- **시스템**의 각종 하드웨어와 네트워크를 **관리/제어**
- 시스템의 내/외부적 **오류 검사 및 복구**

스케줄링 정의

여러 프로세스가 번갈아 사용하는 **공유 자원**을 **어떤 시점에 어떤 프로세스에 언제 할당할 것인지를 결정**
프로세스에 컴퓨터 자원을 적절히 배치하여 시스템 성능을 개선하는 역할

2. 운영체제의 역할 및 목표

- 운영체제의 주요 자원관리

프로세스 관리

- 프로세스 스케줄링 담당
- 프로세스 생성/제거, 시작과 정지, 메시지 전달 기능

기억장치 관리

- 메모리 할당 및 회수 관리 담당

주변장치 관리

- 입출력 장치 스케줄링 및 전반적인 관리 담당

파일 관리

- 파일의 생성과 삭제, 변경, 유지 등의 관리 담당

프로세스란?

- 컴퓨터에서 실행중인 프로그램

2. 운영체제의 역할 및 목표

- 운영체제의 성능평가 기준

처리능력
(Throughput)

- 일정 시간 내에 시스템이 처리하는 일의 양
- **처리량이 극대화**되어야 함

반환시간
(Turn Around Time)

- 시스템에 작업을 의뢰한 시간부터 처리가 완료될 때까지 걸린 시간
- **반환 시간이 최소화**되어야 함

사용 가능성
(Availability)

- 시스템의 자원을 사용할 필요가 있을 때 **즉시 사용 가능**한 정도

신뢰도
(Reliability)

- 시스템이 주어진 **문제를 정확하게 해결**하는 정도
- 처리량이 많아도 오류량이 많으면 안됨

3. 운영체제의 운용 기법 및 발달과정

① 일괄처리 시스템(Batch Processing)

- 초기의 컴퓨터 시스템(초기의 컴퓨터는 통신망으로 서로 연결하여 처리할 수 없어 자료가 생기는 즉시 처리 어려움)
- **일정량/일정 기간 동안 데이터를 모아서 한꺼번에 처리**
- 프로그램 실행 **중간에 사용자가 데이터를 입력하거나 수정하는 것이 불가능**한 시스템
- **일정 기간마다 주기적**으로 한꺼번에 처리할 필요가 있고, 그룹별로 분류 시킬 수 있는 성질의 업무처리에 사용
- 급여 계산, 지불 계산, 연말 결산 등의 업무에 사용

② 다중프로그래밍 시스템(Multiprogramming)

- **하나의 CPU로 여러 작업을 동시에 실행**하는 기술(처리량의 극대화)
- 한 번에 하나의 작업만 가능한 일괄 작업 시스템에 비해 효율성이 뛰어남
- **하나의 주기억 장치에 두 개 이상의 프로그램을 기억시켜 놓고 처리**
- 어떤 프로그램을 수행하다 입출력이 발생하면 해당 입출력이 처리되는 동안 CPU가 다른 프로그램을 수행함
- 어떤 프로그램을 선택할지는 '스케줄링' 을 통해 결정



다중프로그래밍 시스템

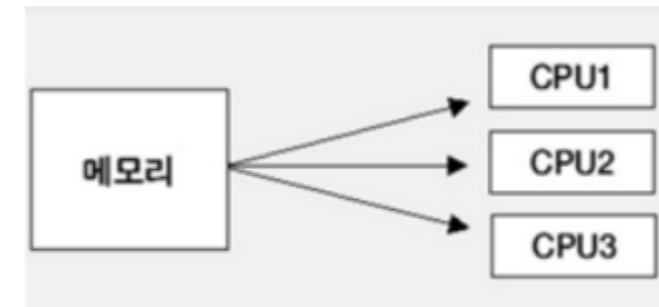
3. 운영체제의 운용 기법 및 발달과정

③ 시분할 시스템(Time Sharing System)

- 여러 명의 사용자가 사용하는 시스템에서 컴퓨터가 사용자들의 프로그램을 번갈아 가며 처리해 줌
- 일정 시간 단위(time slice)로 CPU를 한 사용자에서 다음 사용자로 신속하게 전환해 줌
- 같은 시점에 여러 개의 작업을 할 수 없음
- 실제로 자신만이 컴퓨터를 사용하고 있는 것처럼 사용할 수 있음
- 라운드 로빈(Round Robin)방식(하나의 작업에 할당된 시간을 Time Slice라고 함)
- 다중 프로그래밍 방식과 결합하여 모든 작업이 동시에 처리되는 것처럼 보임(대화식 처리 가능)

④ 다중처리 시스템(MultiProcessing)

- 여러 개의 CPU와 하나의 주기억 장치를 이용하여 여러 개의 프로그램을 동시에 처리
- 하나의 CPU가 고장나도 다른 CPU를 이용하여 업무 처리
- 여러 CPU는 하나의 메모리를 공유하며 단일 운영체제로 관리 됨



다중처리 시스템

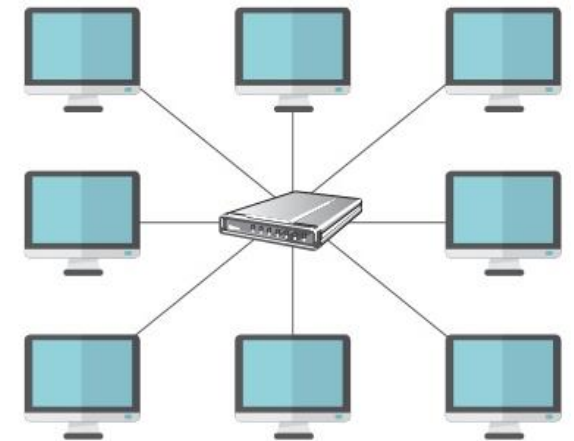
3. 운영체제의 운용 기법 및 발달과정

⑤ 실시간 처리(Real Time Processing)

- 사용할 수 있는 자원이 한정되어 있는 상황에서 **데이터 처리 요구가 있는 즉시 처리**
- 정해진 시간에 반드시 수행되어야 하는 작업에 적합
- 우주 왕복선, 레이더 추적기, 비행기 제어 시스템, 교통제어, 은행의 온라인 업무 등
- 실행 결과를 즉시 받아 볼 수 있어 응답 시간이 짧음

⑥ 분산처리 시스템(Distributed System)

- 하나의 대형 컴퓨터에서 수행하던 기능을 분산된 여러 컴퓨터에 분담 시킴
- 네트워크를 통하여 처리하는 방식
- 각 컴퓨터 시스템은 독립적인 CPU와 메모리를 사용
- **하나의 작업을 여러 개의 컴퓨터 시스템이 공동으로 작업**할 수 있음
- 통신 회선을 통하여 각 시스템에서 처리된 작업들을 모아 중앙 컴퓨터에서 처리
- 클라이언트/서버 운영체제임



(b) 분산 시스템

3. 운영체제의 운용 기법 및 발달과정

⑦ 클라우드 컴퓨팅(Cloud computing)

- 언제 어디서나 응용 프로그램과 데이터를 자유롭게 사용할 수 있는 컴퓨팅 환경
- PC, 핸드폰, 스마트 기기 등을 통하여 인터넷에 접속하고 다양한 작업을 수행
- 기기들 사이에 데이터 이동이 자유로움
- 하드웨어를 포함한 시스템이 구름에 가려진 것처럼 사용자에게 보이지 않는 컴퓨팅 환경이라는 의미

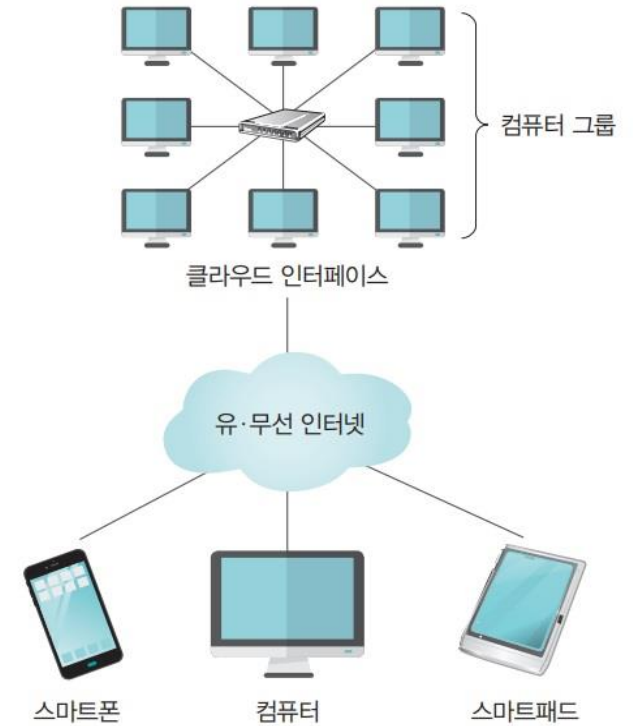


그림 1-18 클라우드 컴퓨팅 환경

⑧ 사물 인터넷(Internet of Things : IoT)

- 사물에 센서와 통신 기능을 내장하여 인터넷에 연결하는 기술
- 인터넷으로 연결된 사물들이 데이터를 주고 받아 스스로 분석하고 학습한 정보를 사용자에게 제공

4. 시스템 소프트웨어

• 시스템 소프트웨어 정의

- 컴퓨터 시스템을 효율적으로 운영하고 사용하기 위해 도와주는 소프트웨어
- 컴퓨터 시스템의 개별 하드웨어 요소를 직접 제어·통합·관리하는 기능을 수행
- 응용프로그램들을 지원하기 위해 개발된 SW로 시스템 전체를 작동시키는 프로그램
(ex) 운영체제, 언어번역프로그램(컴파일러/어셈블러), 링커, 로더, 유틸리티 등
- 사용자보다 H/W 위주의 S/W

곰플레이어는 모니터에게
'영상을 출력해' 라고 말할 권한이 없어
운영체제로 요청해야 함

• 시스템 소프트웨어의 계층적 분류

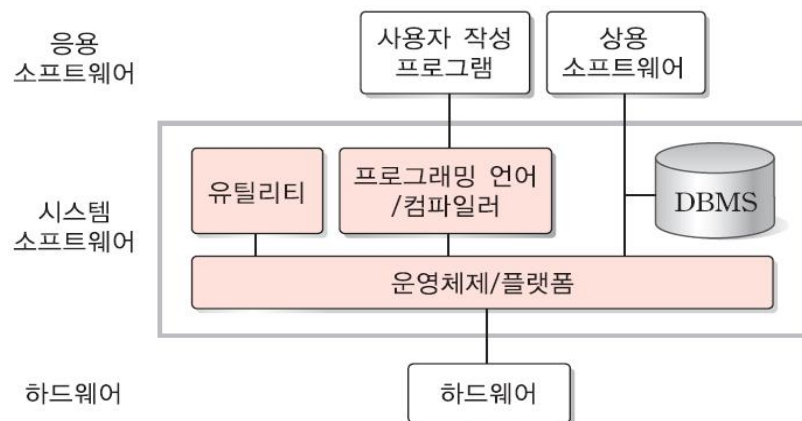
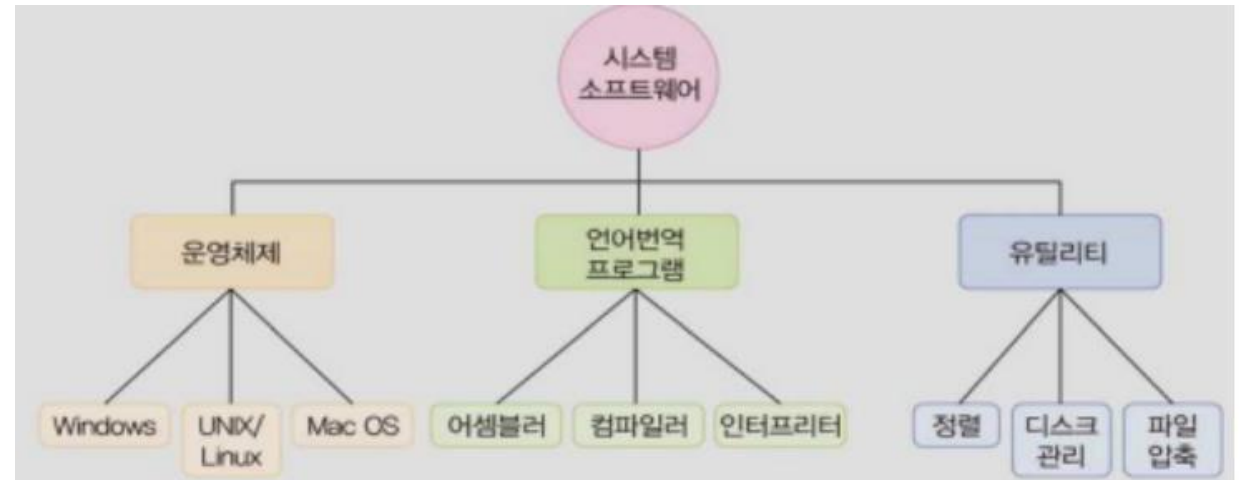


그림 5-16 시스템 소프트웨어의 계층적 분류

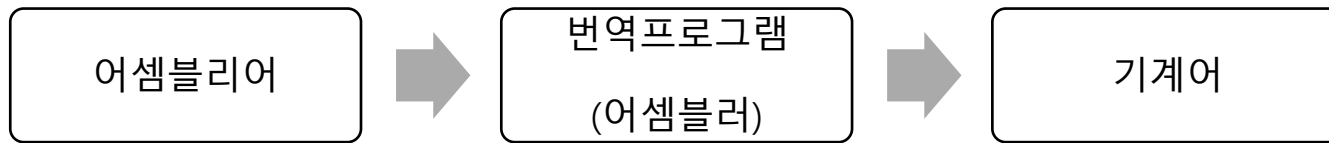


파일관리 : 파일 복사, 이동, 삭제, 디스크 공간 정리
보안 : 바이러스 검사, 스파이웨어 제거, 시스템 백업
시스템 최적화 : 시스템 성능을 향상시키기 위해 디스크 조각모음, 레지스트리 정리

4. 시스템 소프트웨어_어셈블러와 컴파일러/인터프리터

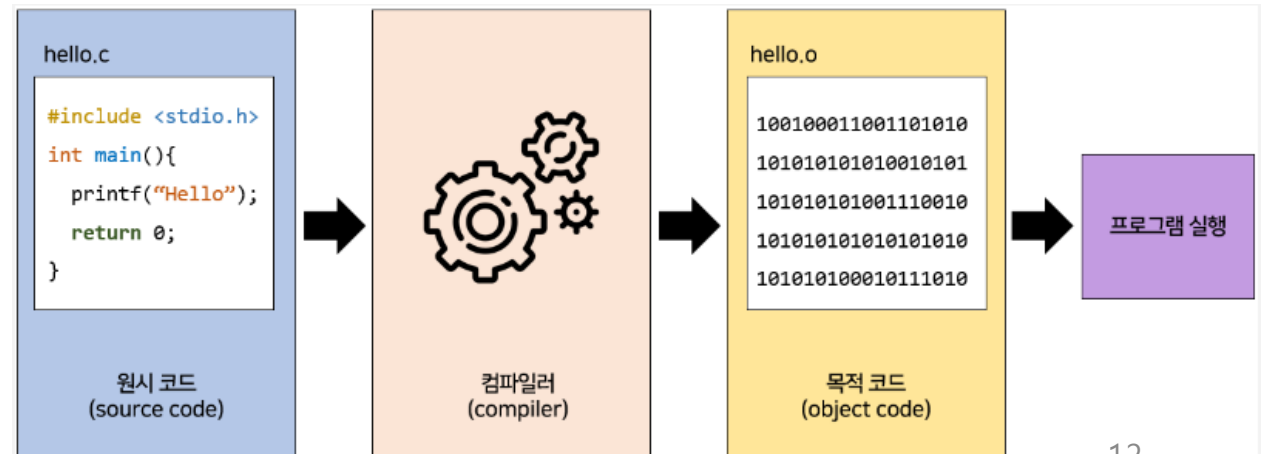
- 어셈블러(Assembler)

- 어셈블리어 : 사용자가 이해하기 어려운 기계어 대신 명령 기능을 쉽게 기호로 코드화 한 저급언어
- 인간이 이해하기 쉽도록 기계어와 거의 일대일로 대응하는 기호로 번역하는 프로그램



- 컴파일러(Compiler)

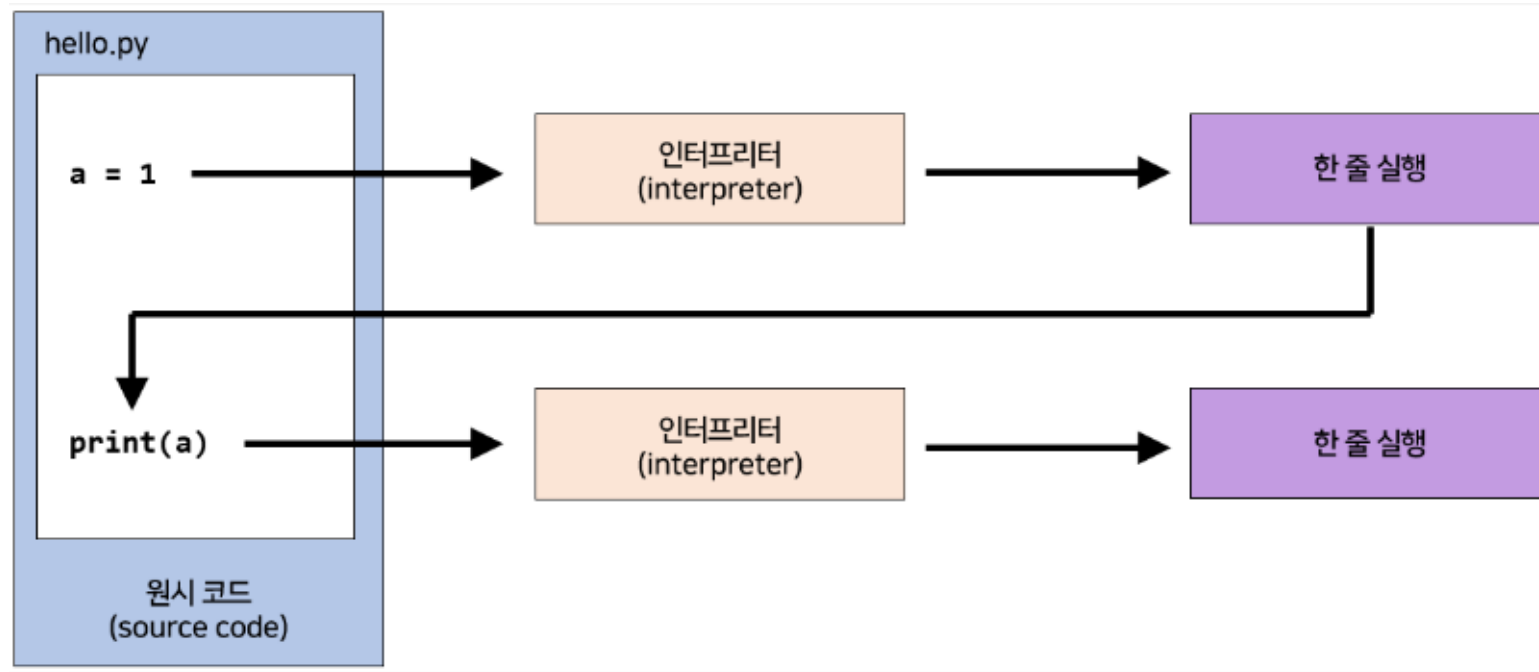
- 고급언어로 작성된 프로그램 전체를 목적프로그램으로 번역하는 프로그램
- 프로그램 전부를 실행하기 전에 한꺼번에 번역(번역 시간이 오래 걸림)
- 한번 번역 후 다시 번역하지 않으므로 실행 속도 빠름
- FORTTRAN, COBOL, PASCAL, C, C++ 등



4. 시스템 소프트웨어_어셈블러와 컴파일러/인터프리터

- 인터프리터(interpreter)

- 고급언어로 작성된 프로그램을 한 줄 단위로 받아들여 번역 후 즉시 실행 시키는 프로그램
- 줄 단위로 번역/실행되므로 시분할 시스템에 유용
- 번역 속도는 빠르지만 프로그램 실행시 매번 번역해야 하므로 실행 속도 느림
- BASIC, LISP, APL, SNOBOL 등



5. 스레드

- 스레드(Thread)의 정의

- 프로세스 내 작업단위로 프로세스 안에 스레드 개념이 포함됨
- 작은 단위의 일이 모여 하나의 작업(task) 단위가 됨 (집 짓기는 프로세스, 토지 구입, 설계, 토목공사 인테리어는 스레드)
- CPU가 처리하는 작업의 단위는 프로세스로부터 전달받은 스레드
 - 운영체제 입장에서의 작업 단위는 프로세스
 - CPU 입장에서의 작업 단위는 스레드
- 모든 프로세스는 한 개 이상의 스레드를 가지고 두 개 이상의 스레드를 사용하는 경우는 멀티스레드라 함

- 프로세스와 스레드의 차이

- 프로세스끼리는 약하게 연결되어 있는 반면 스레드끼리는 강하게 연결되어 있음
 - 프로세스는 자신만의 고유 공간을 할당 받아 사용
 - 스레드는 같은 프로세스 안의 다른 스레드들과 자원을 공유하며 사용

5. 스레드

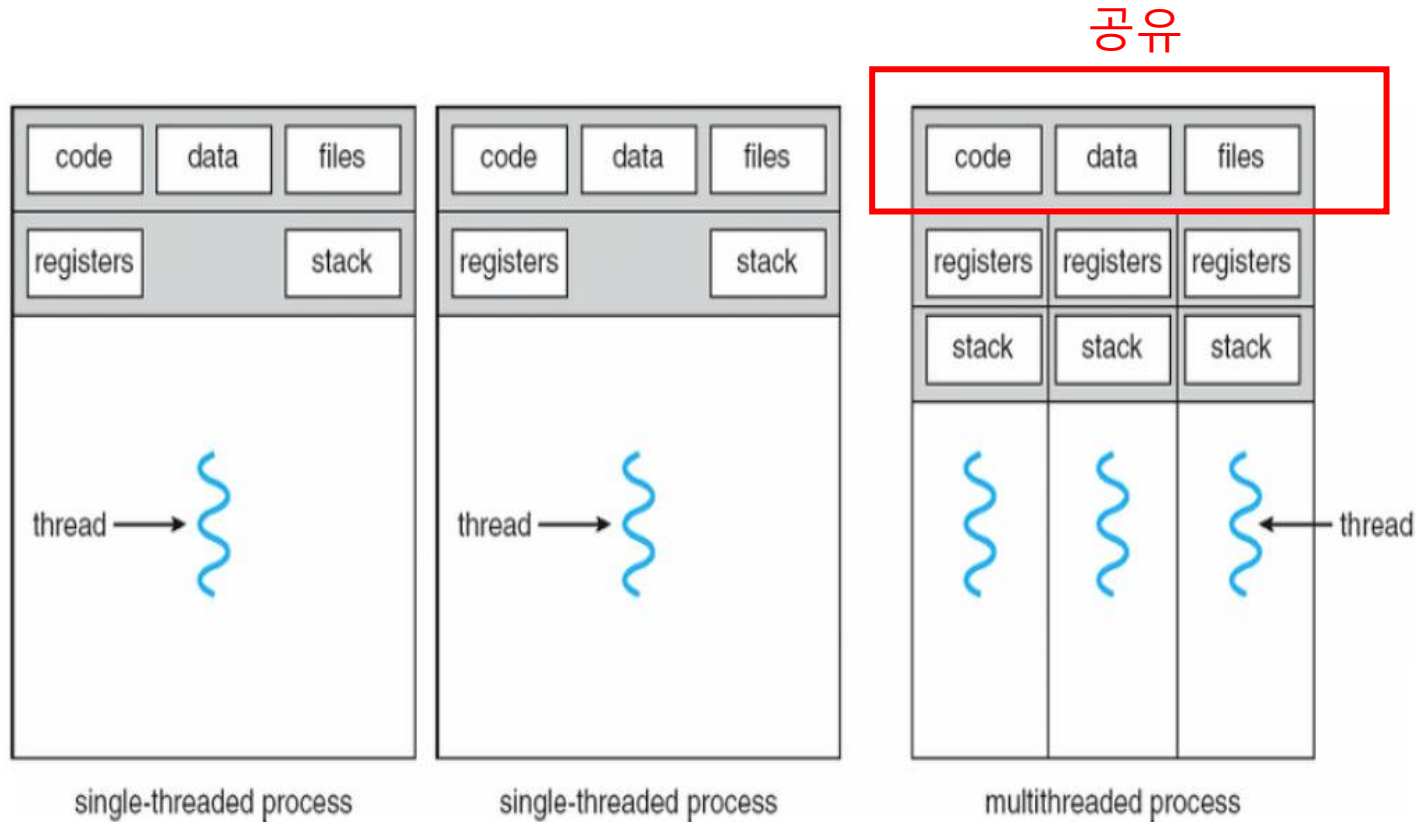
① 싱글스레드

- 한 프로세스가 하나의 스레드를 이용하여 한 번에 한 작업만 수행하는 것은 싱글 스레드(Single thread)
- 데드락과 같은 현상이 발생하지 않음
- 하나의 스레드에서 함수가 실행되는 동안 다른 함수의 실행을 확인할 수 없음

② 멀티스레드

- 하나의 프로세스에 여러 개의 스레드(작업)가 동시에 수행되는 것 (스레드간 통신)
- 프로세스 내 작업을 여러 개의 스레드로 분할함으로써 작업의 부담을 줄이는 프로세스 운영 기법
- 새로운 프로세스를 생성하는 것보다 같은 프로세스 내의 스레드를 생성하는 것이 더 빠름

5. 스레드



- 멀티 스레드의 장점

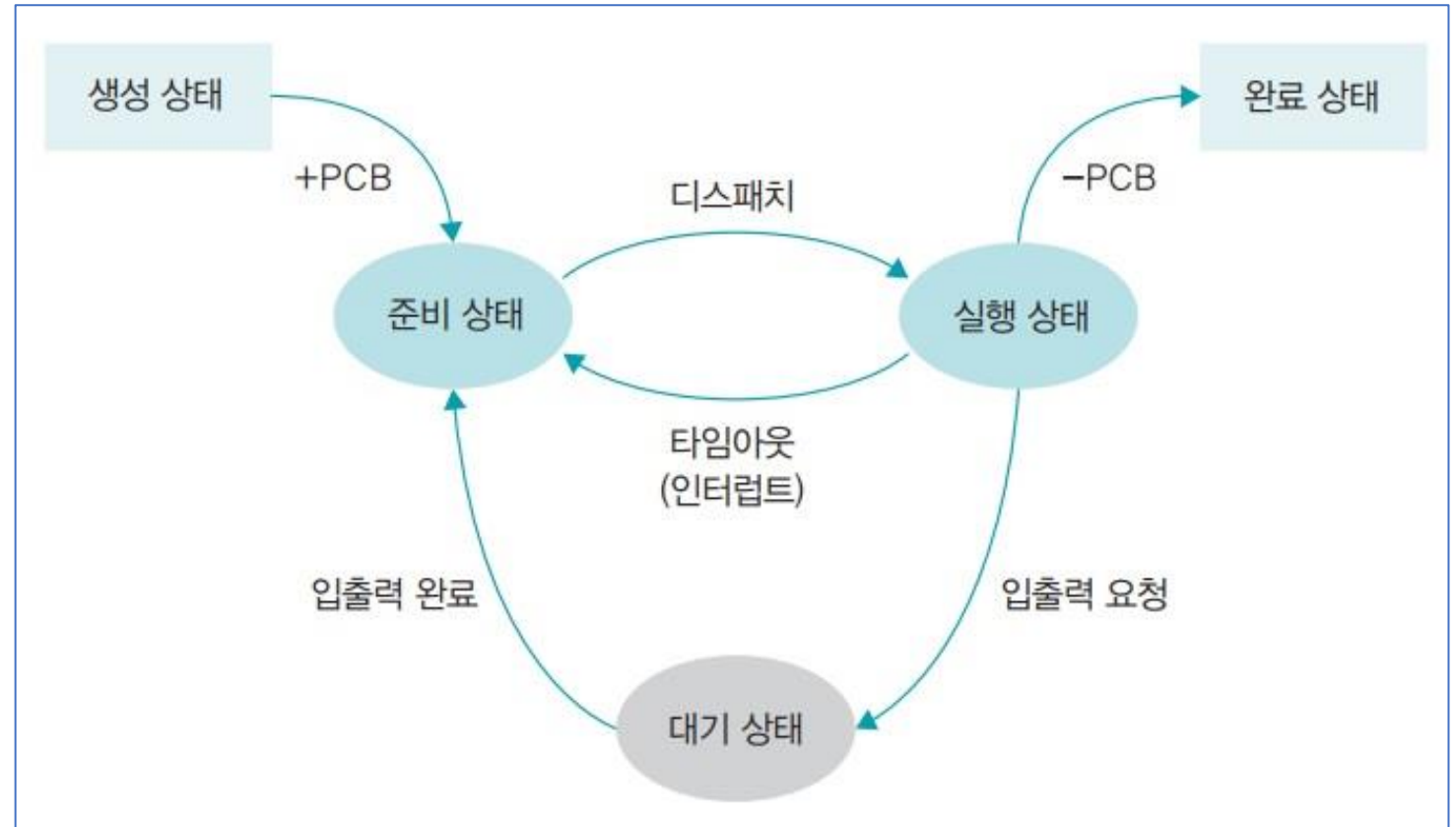
- 한 스레드가 입출력으로 인해 작업이 진행되지 않더라도 다른 스레드가 작업을 계속하므로 응답성 향상
- 프로세스가 가진 자원공유
- 여러 개의 프로세스를 생성하는 것과 달리 불필요한 자원의 중복을 막음으로써 시스템의 효율성 향상

- 멀티 스레드의 단점

- 모든 자원을 공유하기 때문에 한 스레드에 문제가 생기면 전체 프로세스에 영향을 미칠 수 있음

6. 프로세스 스케줄링

- 프로세스의 상태 변화
 - 생성 - 준비 - 실행 - 대기



프로세스(CPU) 스케줄러에 의해 모든 프로세스 상태가 관리됨

6. 프로세스 스케줄링

- 프로세스 스케줄링이란?
 - 여러 개의 프로세스가 시스템에 존재하지만 프로세스를 처리할 CPU는 한대 뿐
 - 자원을 할당 할 프로세스를 선택해야 함 => 스케줄링(Scheduling)
 - CPU나 자원을 효율적으로 사용하기 위한 정책
- 프로세스 스케줄링의 목표는 무엇일까?

모든 프로세스가 공유 자원을 골고루 배분 받고 공정성을 유지하며 안정적으로 작동해야 함

중요도에 따라 우선순위가 지정되어야 하며, 확장성을 고려해 갑작스런 변화에 대응함으로써 전체적인 시스템의 성능을 높여야 함.

6. 프로세스 스케줄링

- 프로세스 스케줄링의 목적

공평성	모든 프로세스가 자원을 공평하게 배정받아야 함. 자원 배정 과정에서 특정 프로세스가 배제되어서는 안 됨 어떤 프로세스도 무한 대기 상태가 되지 않아야 함
효율성	시스템 자원이 유휴 시간 없이 사용되도록 스케줄링을 해야 함. 유휴 자원을 사용하려는 프로세스에는 우선권 을 주어야 함
안정성	우선순위를 사용하여 중요 프로세스가 먼저 작동하도록 배정함 시스템 자원을 점유하거나 파괴하려는 프로세스로부터 자원을 보호 해야 함
확장성	프로세스가 증가해도 시스템이 안정적으로 작동하도록 조치해야 함
반응 시간 보장	응답이 없는 경우 사용자는 시스템이 멈춘 것으로 가정하기 때문에 시스템은 적절한 시간 안에 프로세스의 요구에 반응해야 함
무한 연기 방지	특정 프로세스의 작업이 무한히 연기되어서는 안 됨 - 대책 : 에이징(aging)방법

6. 프로세스 스케줄링

- 선점형 스케줄링

- 운영체제가 필요하다고 판단하면 실행 상태에 있는 프로세스의 작업을 중단시키고 새로운 작업을 시작할 수 있는 방식
- 하나의 프로세스가 CPU를 독점할 수 없기 때문에 빠른 응답 시간을 요구하는 대화형 시스템이나 시분할 시스템에 적합
- 높은 우선 순위를 가진 프로세스들이 빠른 처리를 요구하는 시스템에서 유용 함
- 문맥 교환(Context Switch)으로 인한 오버헤드를 초래할 수 있음

- 비선점형 스케줄링

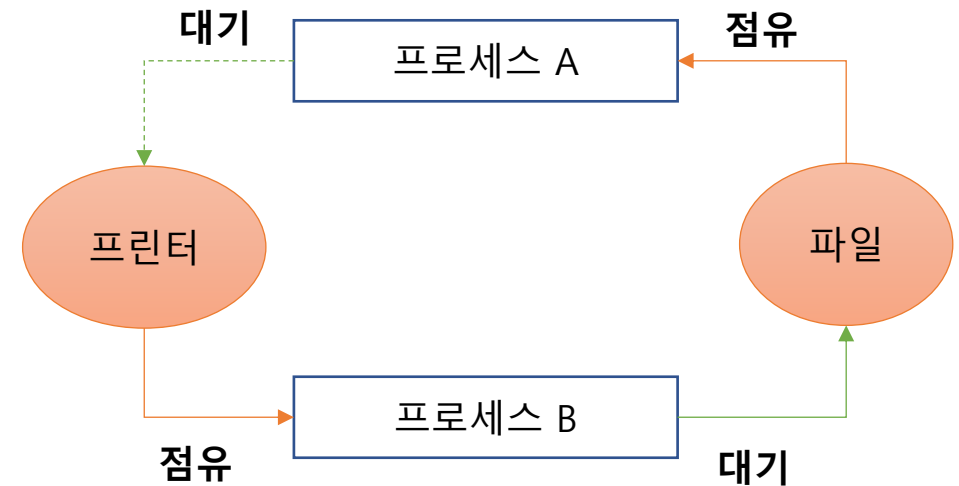
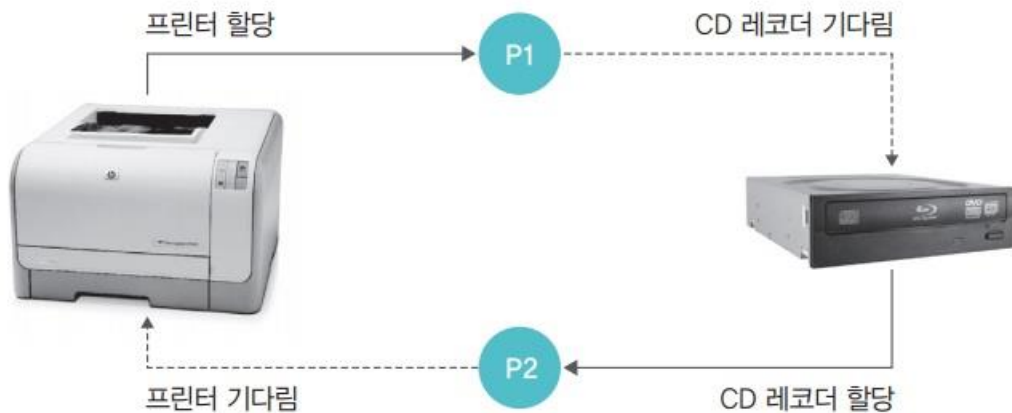
- 어떤 프로세스가 실행 상태에 들어가 CPU를 사용하면 그 프로세스가 종료되거나 자발적으로 대기 상태에 들어가기 전까지는 계속 실행되는 방식
- 선점형 스케줄링보다 스케줄러의 작업량이 적고 문맥 교환에 의한 낭비도 적음
- CPU 사용 시간이 긴 프로세스 때문에 CPU 사용 시간이 짧은 여러 프로세스가 오랫동안 기다리게 되어 전체 시스템의 처리율이 떨어짐
- 과거의 일괄 작업 시스템에서 사용하던 방식

7. 교착상태

- 교착상태 정의

2개 이상의 프로세스가 공유 자원을 사용하기 위해 다른 프로세스의 작업이 끝나기만 기다리며 작업을 더 이상 진행하지 못하는 상태

- 교착 상태는 다른 프로세스와 공유할 수 없는 자원을 사용할 때 발생



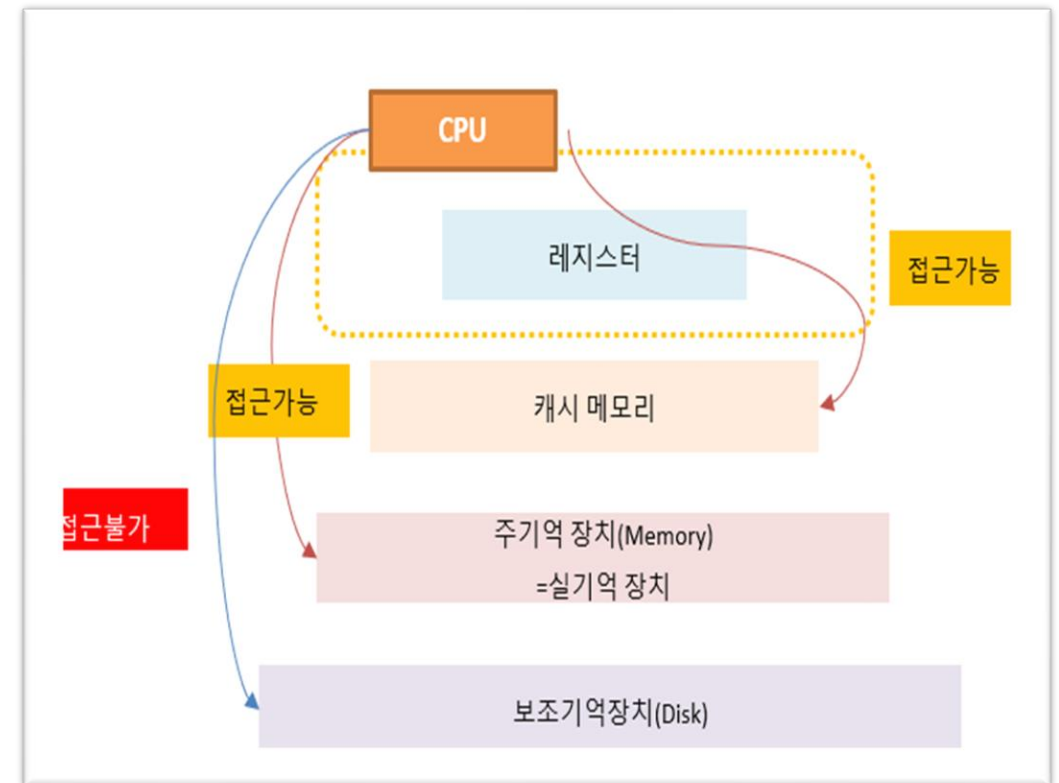
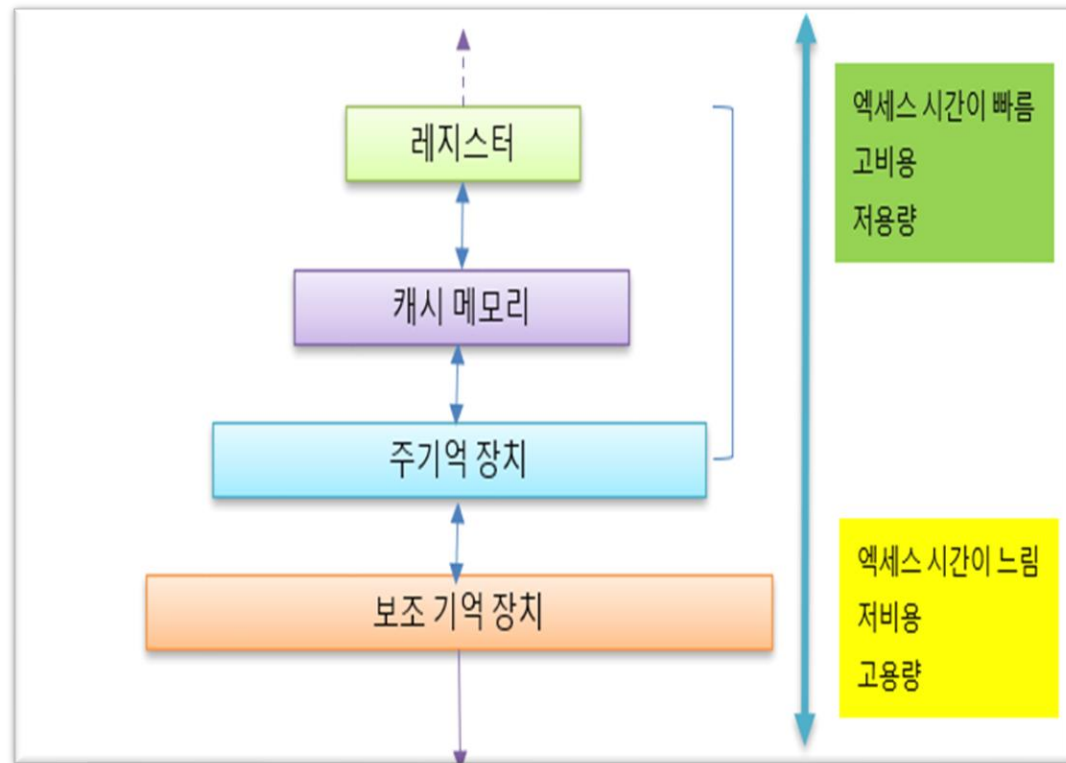
7. 교착상태

- 교착상태 필요조건
 - 다음 4가지 조건이 모두 발생해야만 교착상태 발생
 - 4가지 중 단 한가지라도 만족하지 않으면 교착상태가 발생하지 않음

상호 배제(mutual exclusion)	한 프로세스가 사용하는 자원은 다른 프로세스와 공유할 수 없는 배타적인 자원이어야 함
비선점(non-preemptive)	한 프로세스가 사용 중인 자원은 중간에 다른 프로세스가 빼앗을 수 없는 비선점 자원이어야 함
점유와 대기(hold and wait)	프로세스가 어떤 자원을 할당 받은 상태에서 다른 자원을 기다리는 상태여야 함
원형 대기(circular wait)	점유와 대기를 하는 프로세스 간의 관계가 원을 이루어야 함

8. 기억장치 계층 구조

- 계층구조



8. 기억장치 관리 전략

- 기억장치 관리 전략의 정의
 - 보조기억장치의 프로그램을 주기억장치에 적재 시키는 시기, 적재 위치 등을 지정하는 것
 - 한정된 주기억장치의 공간을 효율적으로 사용하기 위한 것

반입전략 Fetch	배치전략 Placement	교체전략 Replacement
<ul style="list-style-type: none">언제 주기억 장치로 적재할 것인지를 결정하는 전략요구반입예상반입	<ul style="list-style-type: none">주기억장치의 어디에 위치 시킬 것인지 결정하는 전략최소적합(First Fit)최적적합(Best Fit)최악적합(Worst Fit)	<ul style="list-style-type: none">주기억장치의 모든 영역이 이미 사용중인 상태에서 어느 영역을 교체할 것인지 결정하는 전략FIFO, OPT, LRU, NUR, SCR 등

8. 기억장치 관리 전략

• 단편화의 정의

- 프로그램들의 크기와 주기억장치의 크기가 다름으로 발생하는 현상
- 사용되지 않고 남은 기억장치의 빈 공간 조각

내부 단편화

- 정해진 크기에 프로그램을 할당하고 남은 기억 공간
- 사용되지 못한 공간

외부 단편화

- 정해진 크기는 아니지만, 프로그램의 크기가 커서 기억할 수 없게 된 공간

• 단편화 해결방법

통합

Coalescing

- **인접한 공백**(기억공간)들을 더 큰 하나의 공백으로 만드는 과정

집약

Compaction

- **서로 떨어져 있는 여러 개의 낭비 공간**을 모아서 하나의 큰 공간을 만드는 과정(디스크 조각모음)

영역번호	영역크기	상태
1	5K	공백
2	14K	공백
3	10K	사용 중
4	12K	공백
5	16K	공백

First Fit : 첫번째 영역은 2번

Best Fit : 단편화를 가장 작게 남기는 영역은 4번

Worst Fit : 단편화를 가장 많이 남기는 영역은 5번

9. UNIX 시스템

- UNIX 시스템 특징
 - 시분할 시스템을 위해 설계된 대화식 운영체제
 - 사용자에게 명령어를 입력 받기 위해 쉘 화면에 명령어 기술
 - 대부분 코드가 C언어로 기술되어 있음
 - 소스를 누구나 볼 수 있도록 공개된 개방형 시스템
 - 다중 사용자, 다중 작업을 지원 함
 - 이식성이 좋음(PC에서 부터 슈퍼 컴퓨터까지 하드웨어의 종류에 상관없이 운영됨)
 - 계층적의 파일 시스템(트리구조)
 - 다양한 유틸리티 프로그램들이 존재 함
 - 파일 생성 및 삭제 기능, 보호 기능을 가지고 있음
 - 파일 소유자, 그룹 및 그 외 다른 사람으로부터 사용자를 구분하여 파일을 보호

9. UNIX 시스템

- UNIX 시스템 구성

- ① 커널

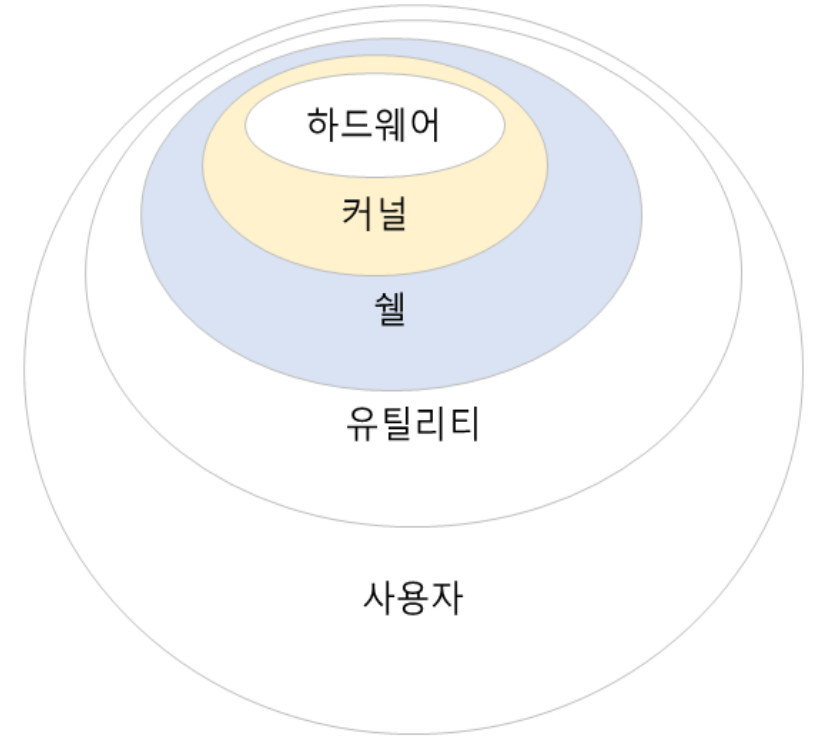
- UNIX의 가장 핵심적인 부분
 - 컴퓨터가 부팅될 때 주기억장치에 적재된 후 상주하면서 실행됨
 - 하드웨어를 보호하고 응용 프로그램에게 다양한 서비스를 제공
 - 프로세스 관리, 기억장치 관리, 파일 관리, 입출력 관리등을 수행함

- ② 셸(Shell)

- 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기
 - 시스템과 사용자간의 인터페이스를 담당

- ③ 유틸리티(Utility)

- 일반 사용자가 작성한 응용 프로그램을 처리하는데 사용
 - 운영체제가 지원하는 사용자 프로그램을 말함(문서편집기, 데이터베이스 관리, 컴파일러, 네트워크 응용프로그램)



10. 분산처리 시스템

- 분산처리 시스템의 특징

- 하나의 대형 컴퓨터에서 수행하던 일들을 지역적으로 분산된 여러 개의 소형 컴퓨터에 분담시켜 수행함
- 독립적인 처리 능력을 가진 컴퓨터 시스템을 통신망으로 연결한 시스템
- 원격에 있는 자원을 마치 지역 자원인 것처럼 쉽게 접근하여 사용할 수 있는 방식

- 분산 운영체제의 목적

자원 공유	• 각 시스템이 통신망을 통해 연결되어 있으므로 유용한 자원을 공유하여 사용할 수 있다.
연산 속도 향상	• 하나의 일을 여러 시스템에 분산시켜 처리함으로써 연산 속도가 향상된다.
신뢰도 향상	• 여러 시스템 중 하나의 시스템에 오류가 발생하더라도 다른 시스템은 계속 일을 처리할 수 있으므로 신뢰도가 향상된다.
컴퓨터 통신	• 지리적으로 떨어진 시스템에 통신망을 두어 정보를 교환할 수 있다.

10. 분산처리 시스템

- 분산처리 시스템의 장단점

- ① 장점

- CPU의 과부하를 줄일 수 있다(처리 능력 한계 극복)
 - 여러 사용자들이 데이터를 공유할 수 있다.
 - 제한된 장치를 여러 지역의 사용자가 공유할 수 있다.
 - 업무량 증가에 따라 시스템의 점진적인 확장이 용이하다.
 - 하나의 일을 여러 시스템이 처리함으로써 연산속도, 신뢰도, 사용가능도가 향상되고 결함 허용이 가능하다.

- ② 단점

- 중앙 집중형 시스템에 비해 소프트웨어 개발이 어렵다.
 - 중앙 집중형 시스템에 비해 보안 정책이 복잡해진다.
 - 에러 발생시 원인 파악이 어렵다.

Chap2. 컴퓨터 구조

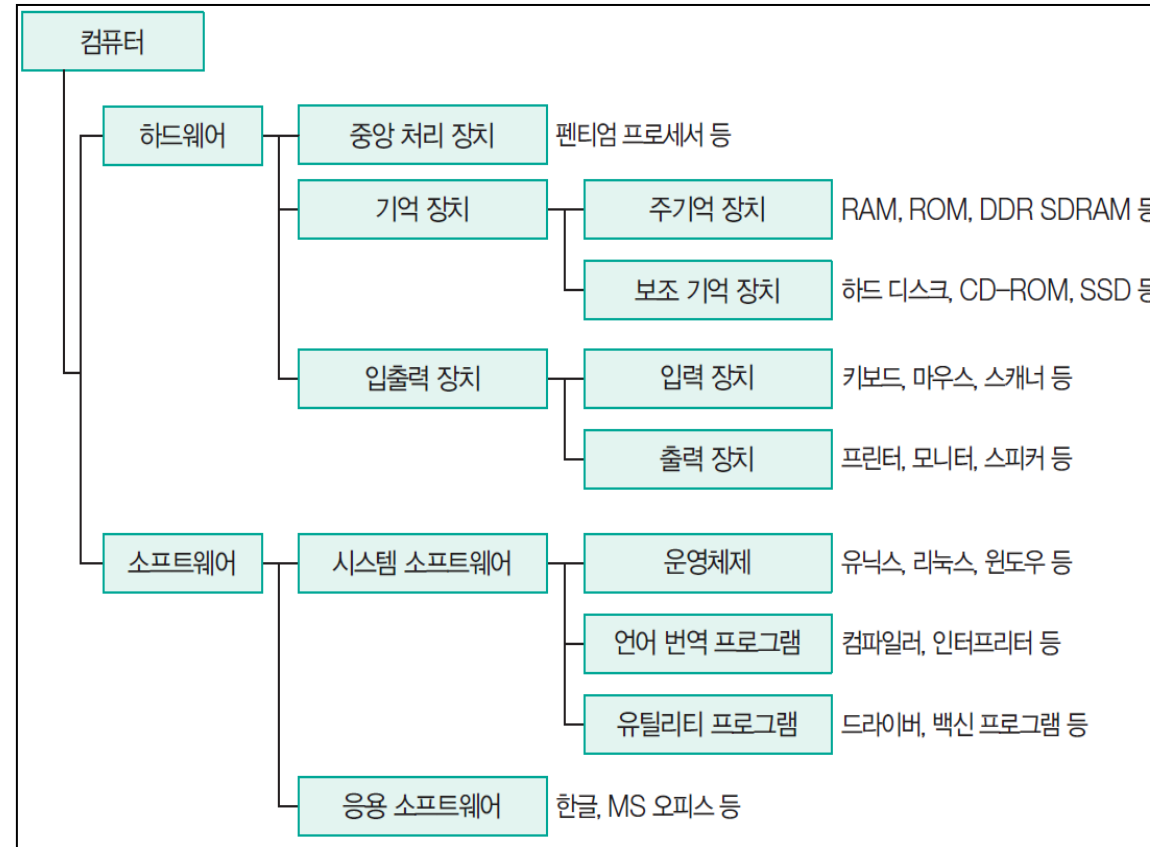
- 컴퓨터 시스템의 구성
- 컴퓨터의 분류
- 데이터 표현
- 진법변환
- 정수표현
- 데이터 코드 표현
- 중앙처리장치
- 명령어 사이클
- 기억장치

1. 컴퓨터 시스템의 구성

- 컴퓨터 시스템의 기본 구성

- 하드웨어 : 컴퓨터에서 각종 정보를 입력하여 처리하고 저장하는 동작이 실제 일어나게 해주는 물리적인 실체
- 소프트웨어 : 컴퓨터를 작동시키거나 이용하기 위한 프로그램과 기술을 모두 가리키는 명령(command)들의 집합

(프로그램)



1. 컴퓨터 시스템의 구성

- 하드웨어

- 중앙 처리 장치, 주기억 장치, 보조 기억 장치, 입력 장치, 출력 장치, 시스템 버스
- 컴퓨터의 기능을 수행하기 위해 각 구성 요소들은 **시스템 버스를 통해 상호 연결**되어 있음

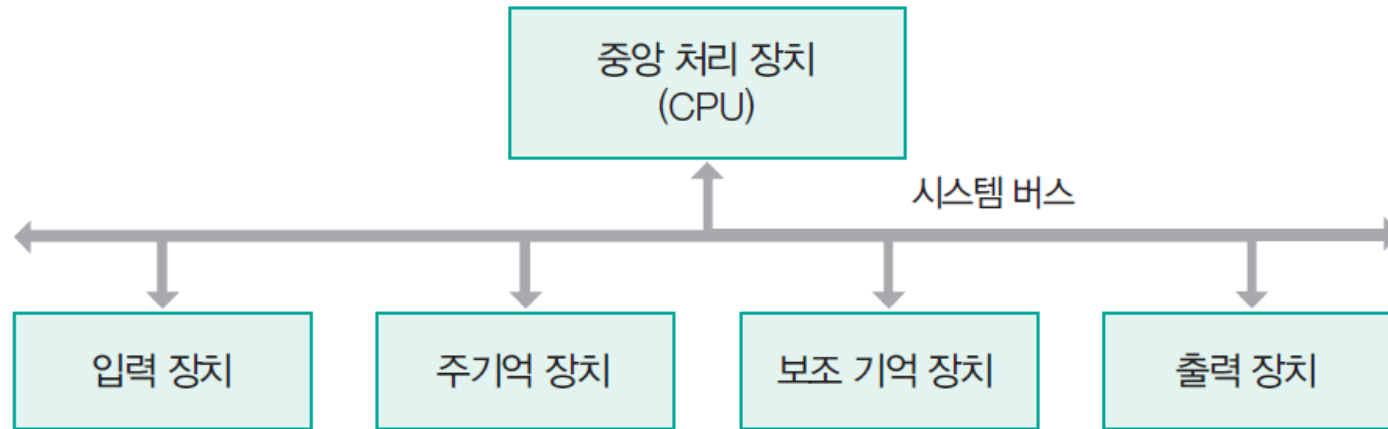


그림 1-2 컴퓨터의 하드웨어 구성

1. 컴퓨터 시스템의 구성

- 중앙 처리 장치(Central Processing Unit, CPU)
 - 컴퓨터의 핵심 기능인 프로그램 실행과 데이터 처리를 담당
 - 중앙 처리 장치를 프로세서(processor) 또는 마이크로프로세서(microprocessor)라고도 부름
- 중앙 처리 장치의 구성 요소
 - ① 산술 논리 연산 장치(Arithmetic and Logic Unit, ALU)
 - 산술 연산, 논리 연산, 보수 연산, 시프트 연산수행
 - ② 제어 장치(Control Unit, CU)
 - 프로그램의 명령어를 해독하여 명령어 실행에 필요한 제어 신호를 발생시키고 컴퓨터의 모든 장치를 제어
 - ③ 레지스터(register)
 - 중앙 처리 장치 내부에 있는 데이터를 일시적으로 보관하는 임시기억 장치로, 프로그램 실행 중에 사용되며 고속으로 액세스할 수 있음

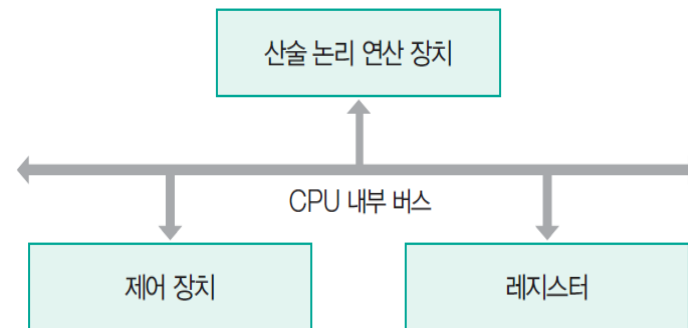


그림 1-3 중앙 처리 장치 내부 구조

1. 컴퓨터 시스템의 구성

- 기억장치

- ① 주기억 장치

- 반도체 칩으로 구성되어 고속으로 액세스가 가능
 - 프로그램 실행 중에 일시적으로만 사용되는 휘발성 메모리

- ② 보조 기억 장치

- 하드 디스크나 SSD, CD-ROM 같은 비휘발성 메모리
 - 저장 밀도가 높고 저가이지만 속도가 느림
 - 중앙 처리 장치에 당장 필요하지 않은 많은 양의 데이터나 프로그램을 저장하는 장치

- 입출력장치

- ① 입력 장치(input device)

- 데이터를 전자적인 2진 형태로 변환하여 컴퓨터 내부로 전달(키보드, 마우스 등)

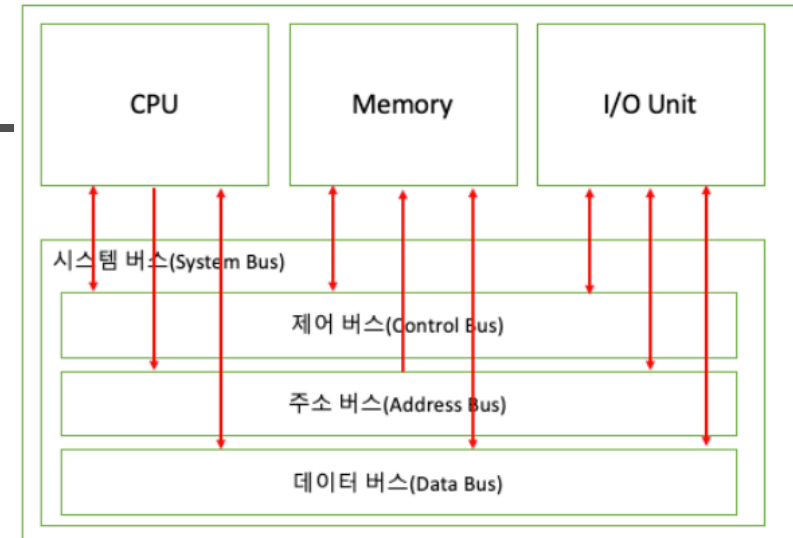
- ② 출력 장치(output device)

- 중앙 처리 장치가 처리한 전자적인 형태의 데이터를 사람이 이해할 수 있는 데이터로 변환하여 출력
 - LCD 모니터, 프린터, 스피커가 있음

1. 컴퓨터 시스템의 구성

• 시스템 버스

- CPU와 메인보드 사이에 데이터를 주고받는 통로
- 중앙 처리 장치와 기억 장치 및 입출력 장치 사이에 **정보를 교환하는 통로**
- 컴퓨터 시스템의 구성 요소들을 상호 연결해 주는 중심 통로
- 주소 버스, 데이터 버스, 제어 버스가 있음



주소 버스 (address bus)	<ul style="list-style-type: none"> • 기억 장치나 입출력 장치를 지정하는 주소 정보를 전송하는 신호 선들의 집합 • 주소는 중앙 처리 장치에서 발생되어 기억장치와 입출력 장치로 보내지는 정보 • 단방향 전송
데이터 버스 (data bus)	<ul style="list-style-type: none"> • 기억 장치나 입출력 장치 사이에 데이터를 전송하기 위한 신호선들의 집합. • 데이터선의 수는 중앙 처리 장치가 한 번에 전송할 수 있는 데이터 비트의 수를 결정 • 데이터를 읽기 / 쓰기 동작 모두를 지원해야 하므로 양방향 전송
제어 버스 (control bus)	<ul style="list-style-type: none"> • 중앙 처리 장치가 시스템 내의 각종 요소의 동작을 제어하는 데 필요한 신호선들의 집합 • 기억 장치 읽기와 쓰기 신호(memory read/write), 입출력 장치 읽기와 쓰기 신호(I/O read/write) 등이 있음 • 단방향

1. 컴퓨터 시스템의 구성

- CPU와 메모리간의 시스템 버스의 동작형태 예

- 주기억장치 주소 50에 데이터 20을 쓰기
- 주소 버스에 50을, 데이터 버스에 20을 전송, 제어 버스에 쓰기 제어 신호를 보냄

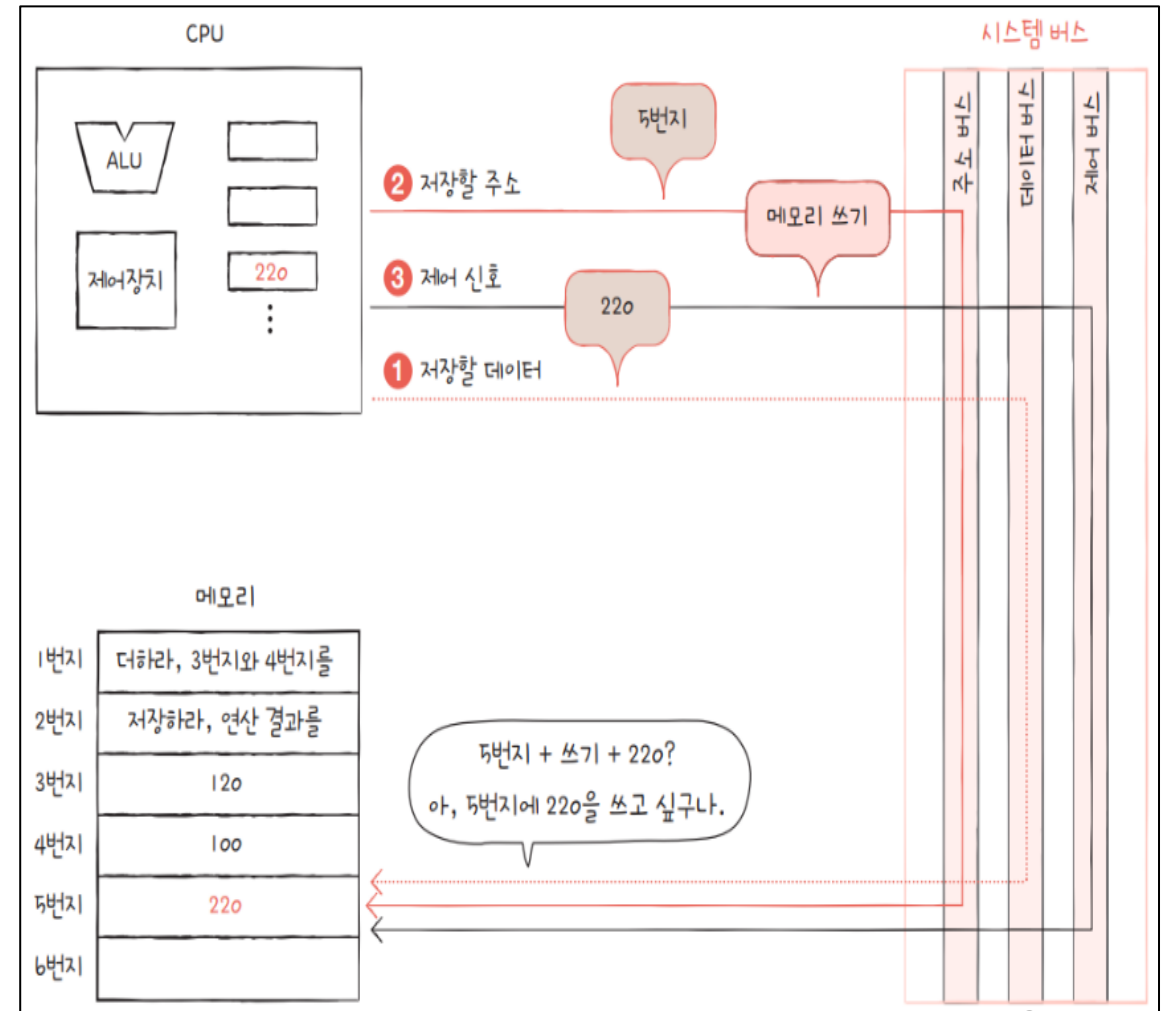
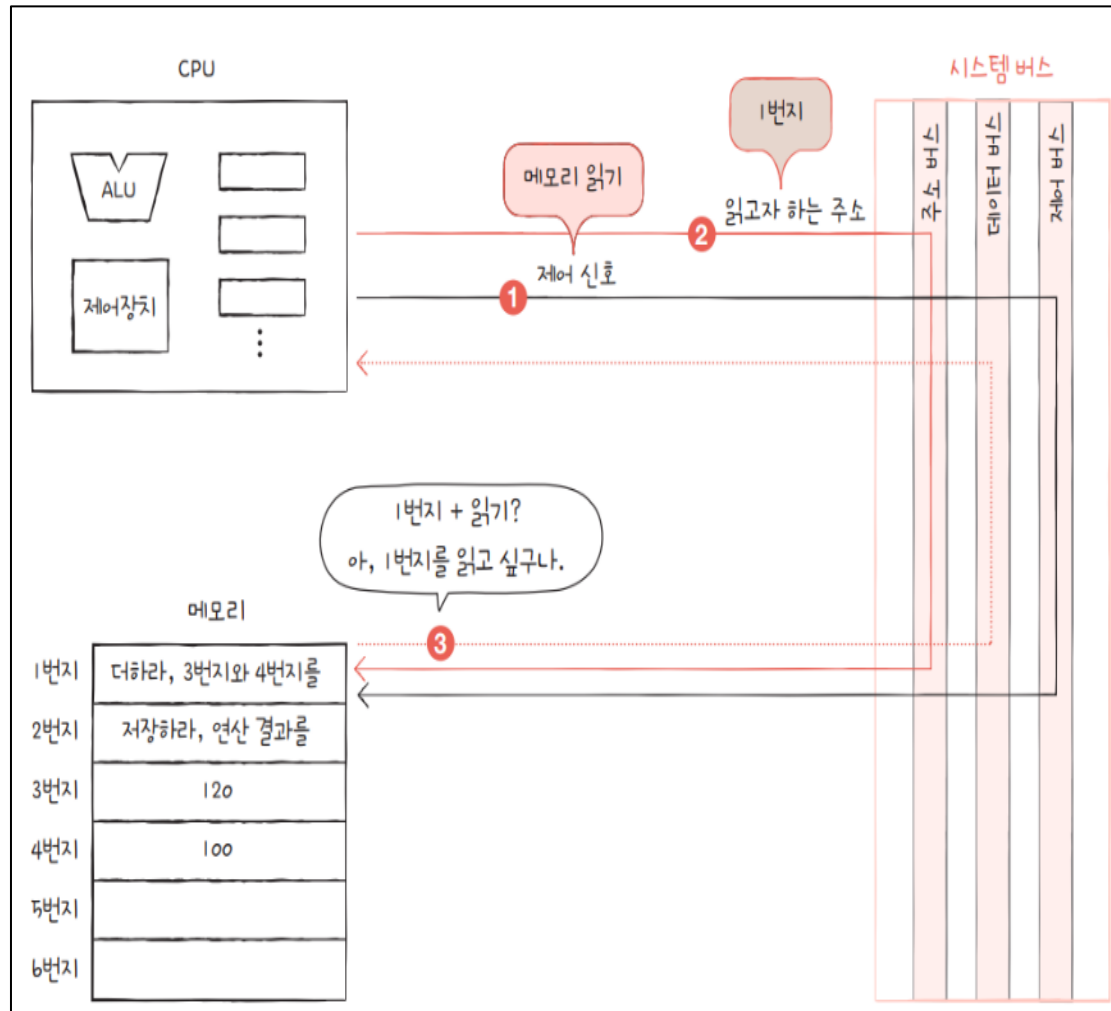


- 주기억장치 주소 60의 데이터를 읽기
- 주소 버스에 60을 전송하고, 제어 버스에 읽기 제어 신호를 보냄.



1. 컴퓨터 시스템의 구성

- CPU와 메모리간의 시스템 버스의 동작형태 예



1. 컴퓨터 시스템의 구성

- 소프트웨어

- ① 시스템 소프트웨어

- 하드웨어를 관리하고 응용 소프트웨어를 실행하는데 필요한 프로그램
 - 운영체제, 언어번역 프로그램, 링커, 로더, 유틸리티 프로그램(장치 드라이버, 백신 프로그램 등)등이 속함

- ② 응용 소프트웨어

- 컴퓨터 시스템을 일반 사용자들이 특정한 용도에 활용하기 위해 만든 프로그램
 - 사무용(한글, MS-Office), 그래픽용(포토샵, 일러스트레이터), 통신 및 네트워크용(크롬, 카카오톡)

1. 컴퓨터 시스템의 구성

- 프로그램 처리과정

- 프로그램의 정의

어떤 결과를 얻기 위해 컴퓨터가 받아들일 수 있는 형태의 명령어를 나열하여 구성한 문장
 고급언어(C, C++ 등)로 작성
 사람들이 이해하기 쉽지만 컴퓨터는 이해할 수 없음

- 고급언어로 작성한 프로그램 => 컴파일러를 이용해 컴퓨터가 알아들을 수 있는 언어(기계어)로 번역해야 함



고급언어	인간이 이해하기 쉬운 프로그래밍 언어 C, C++, 자바, 파이썬
저급언어	기계어나 어셈블리어를 말함

니모닉(mnemonic)
 LOAD, ADD, STOR

그림 1-5 고급 언어 프로그램의 변환 과정과 예

2. 데이터 표현

• 디지털 정보의 단위

① 비트(bit)

- 0과 1의 두 가지 상태만 저장되며 2진수로 데이터를 표현하는 단위
- 매우 단순한 정보만 표현(보통 정보를 표현 할 때 1byte를 사용)
- n 비트로 2진수 2^n 개를 표현할 수 있음
- 1nibble = 4bit

② 바이트(Byte)

- 정보를 표현하기 위한 비트의 집합(1byte = 8bit)
- 1byte = 1문자(character), 영어는 1byte로 한 문자 표현, 한글은 2byte가 필요

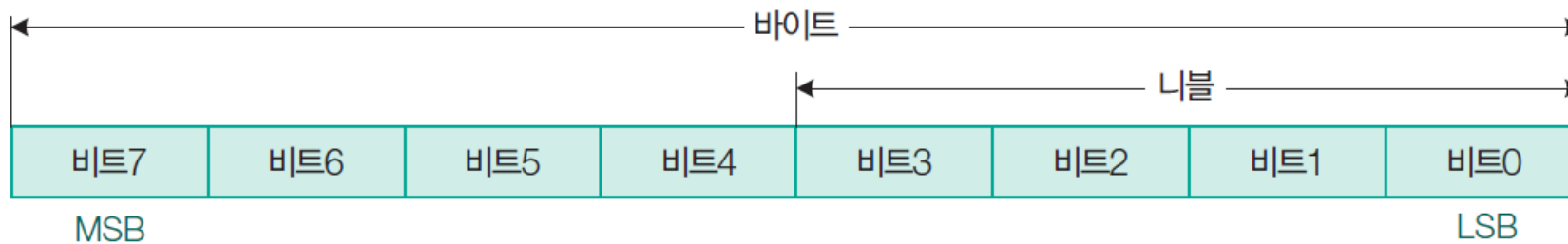


그림 2-1 비트, 니블, 바이트의 관계

MSB Most Significant Bit: 최상위 비트

LSB Least Significant Bit: 최하위 비트

2. 데이터 표현

기수 : 진법을 나타내는 기본 수

- 10진법(Decimal Notation)

- 기수가 10인 수
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9의 10개 수로 표현

$$\begin{aligned} 9345.35 &= 9 \times 1000 + 3 \times 100 + 4 \times 10 + 5 \times 1 + 3 \times 0.1 + 5 \times 0.01 \\ &= 9 \times 10^3 + 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 + 3 \times 10^{-1} + 5 \times 10^{-2} \end{aligned}$$

- 2진법(Binary Notation)

- 기수가 2인 수
- 0, 1 두 개의 수로 표현

$$\begin{aligned} 1010.1011_{(2)} &= 1 \times 1000_{(2)} + 0 \times 100_{(2)} + 1 \times 10_{(2)} + 0 \times 1_{(2)} \\ &\quad + 1 \times 0.1_{(2)} + 0 \times 0.01_{(2)} + 1 \times 0.001_{(2)} + 1 \times 0.0001_{(2)} \\ &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} \end{aligned}$$

2. 데이터 표현

- 8진법

- 0~7까지 8개의 수로 표현

$$\begin{aligned} 607.36_{(8)} &= 6 \times 100_{(8)} + 0 \times 10_{(8)} + 7 \times 1_{(8)} + 3 \times 0.1_{(8)} + 6 \times 0.01_{(8)} \\ &= 6 \times 8^2 + 0 \times 8^1 + 7 \times 8^0 + 3 \times 8^{-1} + 6 \times 8^{-2} \end{aligned}$$

- 16진법

- 0~9, A~F까지 16개의 기호로 표현

$$\begin{aligned} 6C7.3A_{(16)} &= 6 \times 100_{(16)} + C \times 10_{(16)} + 7 \times 1_{(16)} + 3 \times 0.1_{(16)} + A \times 0.01_{(16)} \\ &= 6 \times 16^2 + C \times 16^1 + 7 \times 16^0 + 3 \times 16^{-1} + A \times 16^{-2} \end{aligned}$$

- 8진수보다는 16진수를 사용하는 경우가 더 많은데 실제로 컴퓨터 구조나 어셈블리어에서는 16진수를 많이 사용함

2. 데이터 표현

- 2진수에 해당하는 8진수, 16진수, 10진수 표현

표 2-2 2진수에 해당하는 8진수, 16진수, 10진수 표현

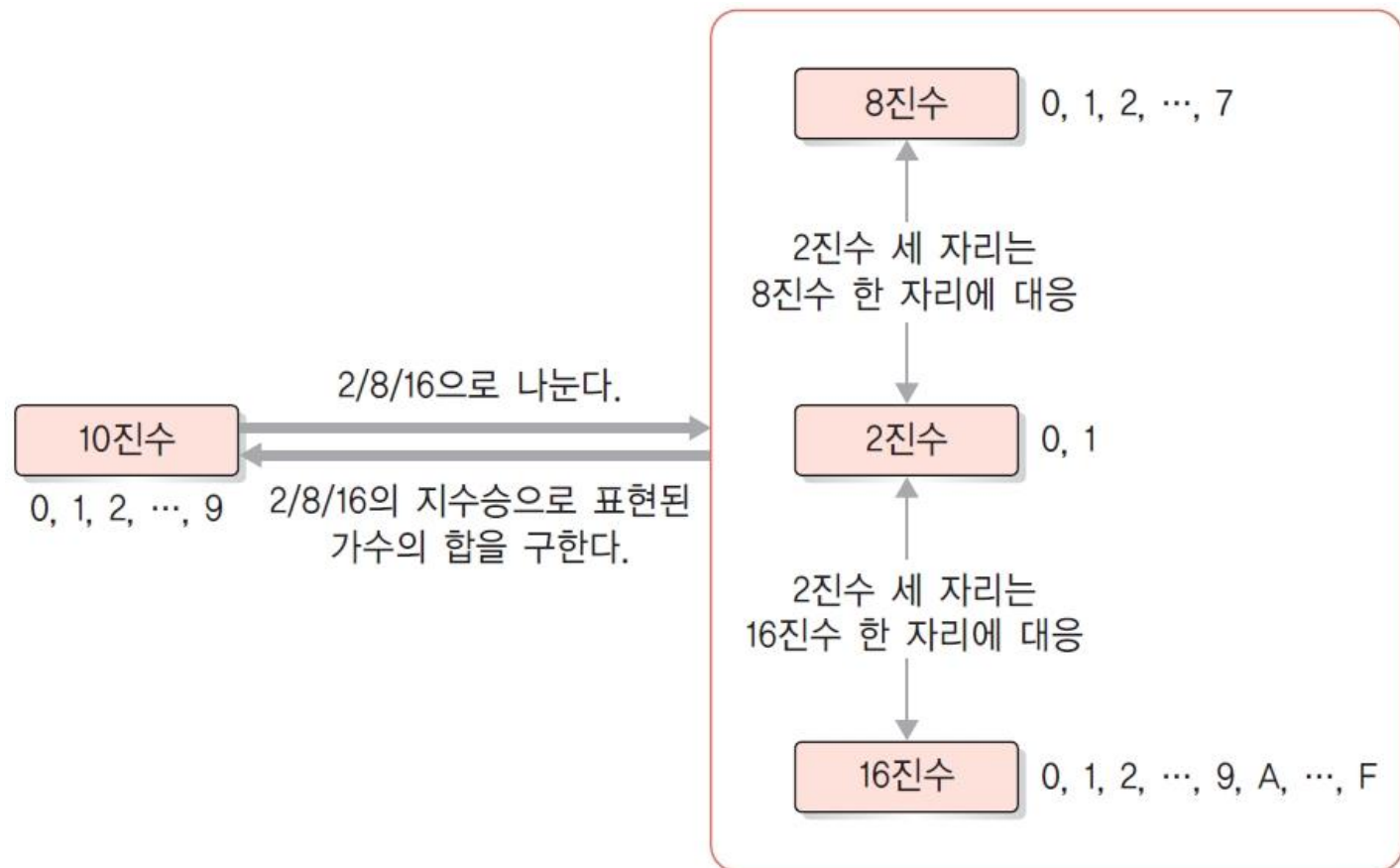
2진수	8진수	10진수	2진수	16진수	10진수	2진수	16진수	10진수
000	0	0	0000	0	0	1000	8	8
001	1	1	0001	1	1	1001	9	9
010	2	2	0010	2	2	1010	A	10
011	3	3	0011	3	3	1011	B	11
100	4	4	0100	4	4	1100	C	12
101	5	5	0101	5	5	1101	D	13
110	6	6	0110	6	6	1110	E	14
111	7	7	0111	7	7	1111	F	15

3. 진법변환

- 진법변환

- 각 진법의 지수는 기수의 지수승으로 표현 함

- $M = B^E$



3. 진법 변환

- 10진수 => 2진수로 변환
 - 정수부분과 소수부분으로 나누어 변환
 - 정수부분은 2로 나누고, 소수부분은 2를 곱함
- 10진수 75.6875를 2진수로 변환한 예

2 75	나머지	→	2진수
2 37 ... 1		→	1
2 18 ... 1		→	11
2 9 ... 0		→	011
2 4 ... 1		→	1011
2 2 ... 0		→	01011
2 1 ... 0		→	001011
0 ... 1		→	1001011
몫			

2진수	←	정수	소수	
		0.	6875	
		×	2	
0.1	←	1.	3750	곱셈 결과 정수를 적는다.
		×	2	
0.10	←	0.	7500	
		×	2	
0.101	←	1.	5000	
		×	2	
0.1011	←	1.	0	소수 부분이 0이 될 때까지 계산한다.

$$75.6875_{(10)} = 1001011.1011_{(2)}$$

3. 진법 변환

- 2진수, 8진수, 16진수를 10진수로 변환 => 각 자릿수에 기수(가중치)를 곱해서 더함

① 2진수를 10진수로 변환한 예

$$\begin{aligned} 101101.101_{(2)} &= 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\ &= 32 + 0 + 8 + 4 + 0 + 1 + 0.5 + 0 + 0.125 \\ &= 45.625_{(10)} \end{aligned}$$

② 8진수를 10진수로 변환한 예

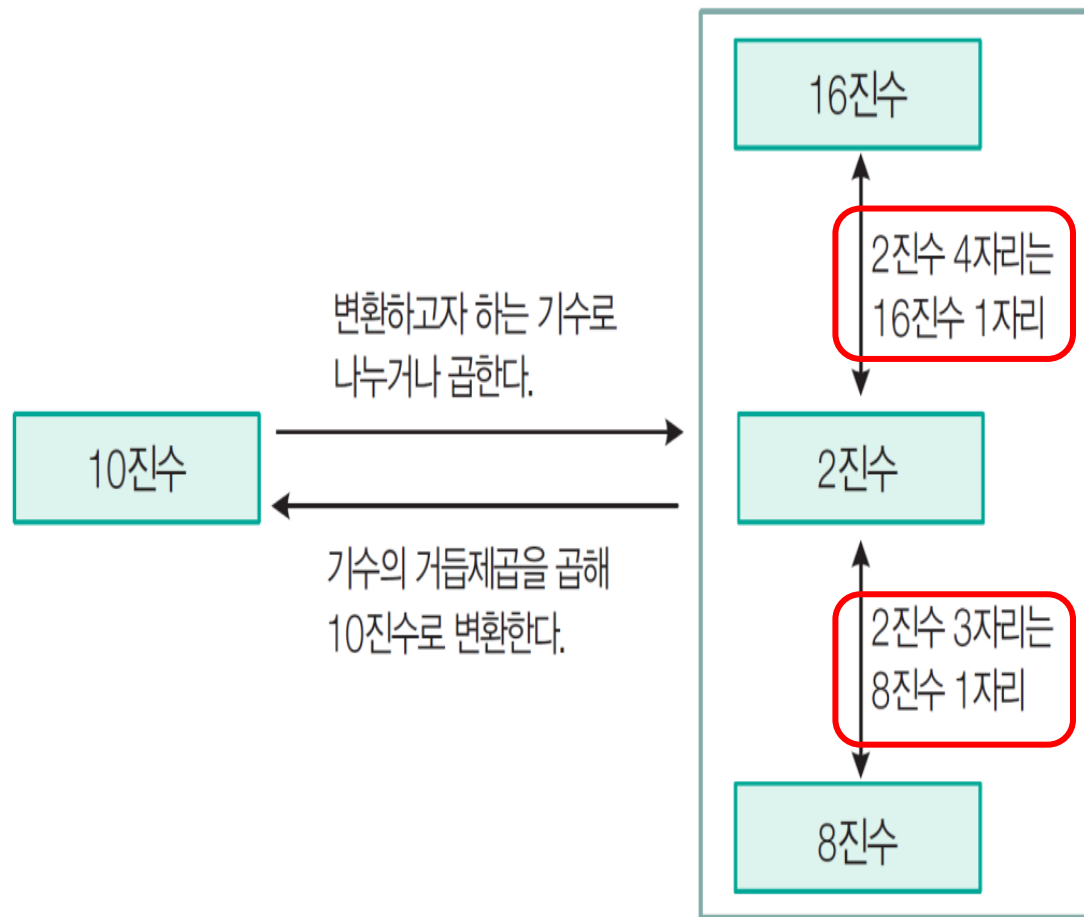
$$\begin{aligned} 364.35_{(8)} &= 3 \times 8^2 + 6 \times 8^1 + 4 \times 8^0 + 3 \times 8^{-1} + 5 \times 8^{-2} \\ &= 3 \times 64 + 6 \times 8 + 4 \times 1 + 3 \times 0.125 + 5 \times 0.015625 \\ &= 192 + 48 + 4 + 0.375 + 0.078125 \\ &= 244.453125_{(10)} \end{aligned}$$

③ 16진수를 10진수로 변환한 예

$$\begin{aligned} A3.D2_{(16)} &= 10 \times 16^1 + 3 \times 16^0 + 13 \times 16^{-1} + 2 \times 16^{-2} \\ &= 10 \times 16 + 3 \times 1 + 13 \times 0.0625 + 2 \times 0.00390625 \\ &= 160 + 3 + 0.8125 + 0.0078125 \\ &= 163.8203125_{(10)} \end{aligned}$$

3. 진법 변환

- 2진수 - 8진수 - 10진수 - 16진수 상호 변환



10진수	2진수	8진수	16진수
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

그림 2-2 2진수, 8진수, 10진수, 16진수 상호 변환 개념도

3. 진법 변환_Quiz

- 10진수를 2진수, 8진수로 변환
 - 25
 - 18
 - 10.5
- 2진수를 10진수로 변환
 - 1010110
 - 10011.011
 - 110.1
- 10진수 48.8125를 2진수, 8진수, 16진수로 변환

- 10진수를 2진수
 - $25 \Rightarrow 11001$
 - $18 \Rightarrow 10010$
 - $10.5 \Rightarrow 1010.1$
- 10진수를 8진수
 - $25 \Rightarrow 31$
 - $18 \Rightarrow 22$
 - $10.5 \Rightarrow 12.4$
- 2진수를 10진수로
 - $1010110 \Rightarrow 86$
 - $10011.011 \Rightarrow 19.375$
 - $110.1 \Rightarrow 6.5$
- $110000.1101_{(2)}$
- $60.64_{(8)}$
- $30.D_{(16)}$

4. 데이터 표현 코드_숫자코드

- BCD 코드(8421코드)
 - 이진수 네 자리를 묶어 십진수 한 자리로 사용하는 기수법
 - 십진수 0~9까지 숫자를 4비트의 2진수로 표현 (십진수를 이진수로 변환하는 것과 다름)
 - 십진수 각각의 자리에 해당하는 숫자만 4비트 2진수로 변환

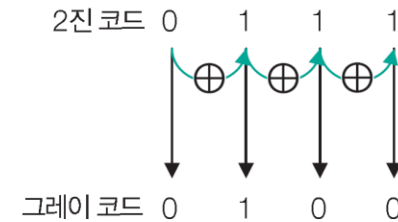
10진수	BCD 코드	10진수	BCD 코드	10진수	BCD 코드
0	0000	10	0001 0000	20	0010 0000
1	0001	11	0001 0001	31	0011 0001
2	0010	12	0001 0010	42	0100 0010
3	0011	13	0001 0011	53	0101 0011
4	0100	14	0001 0100	64	0110 0100
5	0101	15	0001 0101	75	0111 0101
6	0110	16	0001 0110	86	1000 0110
7	0111	17	0001 0111	97	1001 0111
8	1000	18	0001 1000	196	0001 1001 0110
9	1001	19	0001 1001	237	0010 0011 0111

4. 데이터 표현 코드_숫자코드

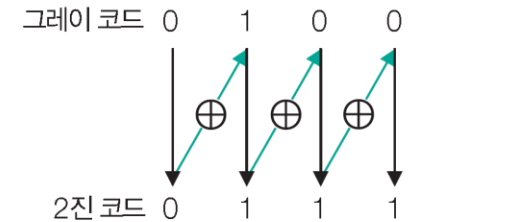
• 그레이 코드

- 연속되는 코드들 간에 하나의 비트만 변경(한 번에 하나의 비트만 변화)
- 회로 설계나 데이터 변환시에 유용하게 활용 됨

10진 코드	2진 코드	그레이 코드	10진 코드	2진 코드	그레이 코드
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000



(a) 2진 코드를 그레이 코드로 변환하는 방법



(b) 그레이 코드를 2진 코드로 변환하는 방법

그림 2-7 2진 코드와 그레이 코드의 상호 변환 방법

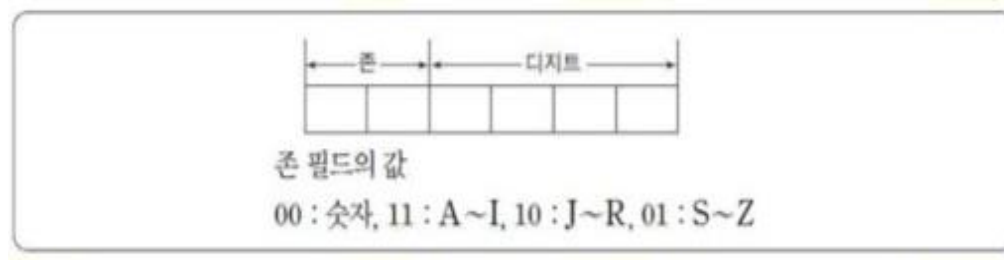
XOR의 진리표

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

$$F = A \oplus B$$

4. 데이터 표현 코드_영숫자 코드

- 표준 BCD 코드
 - 6비트로 한 문자를 표현(최대 64문자까지 표현 가능)
 - 상위 2비트 : 존 비트
 - 하위 4비트 : 디지털 비트



- zone 비트 : 영문자, 숫자, 특수문자를 구분하는 역할
- 나머지 4비트 : 일련번호 역할

존 비트		디지털 비트			
5	4	3	2	1	0
1	1	영문자 A~I(0001~1001)			
1	0	영문자 J~R(0001~1001)			
0	1	영문자 S~Z(0010~1001)			
0	0	숫자 0~9(0001~1010)			
혼용		특수문자 및 기타문자			

4. 데이터 표현 코드_영숫자 코드

- ASCII 코드

- 미국 국립 표준 연구소(ANSI)가 제정한 정보 교환용 미국 표준 코드(7bit로 표현)
- 7비트 128가지의 문자를 표현 가능
- 데이터 통신용 코드
- 대소문자 구별

zone bit			digit bit			
6	5	4	3	2	1	0
1	0	0	영문자 A~O(0001~1111)			
1	0	1	영문자 P~Z(0000~1010)			
0	1	1	숫자 0~9(0000~1001)			

4. 데이터 표현 코드_영숫자 코드

- EBCDIC 코드

- 대형 컴퓨터와 IBM 계열 컴퓨터에서 많이 사용되고 있는 8비트 코드(IBM에서 개발)
- 7비트의 표준 BCD 코드를 8비트로 확장한 코드
- 256종류의 문자 코드를 표현할 수 있는 영숫자 코드

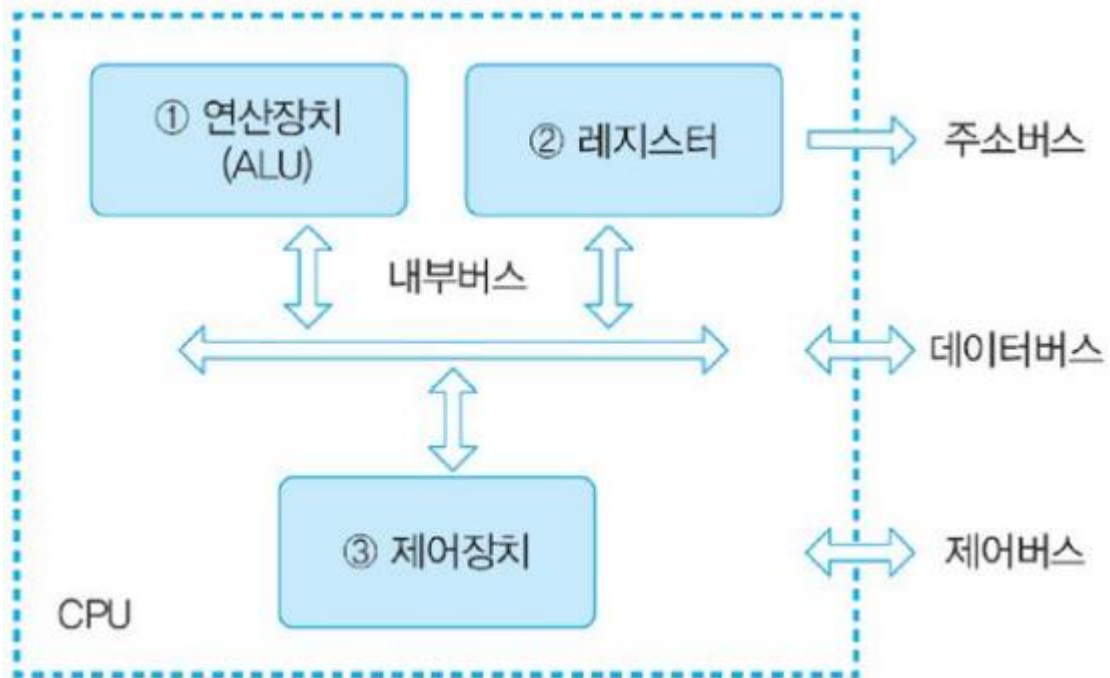
b_9	$b_8 b_7 b_6 b_5$	$b_4 b_3 b_2 b_1$
패리티	존 비트	디지털 비트
1	4	4

4. 데이터 표현 코드_영숫자 코드

- 유니코드
 - 2byte코드(16비트, 65,536자까지)
 - ASCII 코드의 한계성을 극복하기 위하여 개발된 인터넷 시대의 표준
 - 다양한 운영체제와 최신의 모든 웹 브라우저 및 기타 많은 제품에서 유니코드를 지원
 - 미국, 유럽, 동아시아, 아프리카, 아시아 태평양 지역 등의 주요 언어들에 적용될 수 있음
 - 유니코드는 유럽, 중동, 아시아 등 거의 대부분의 문자를 포함하고 있으며, 10만개 이상의 문자로 구성되어 있음
 - 구두표시, 수학기호, 전문기호, 기하학적 모양, 덩벙 기호 등을 포함

5. 중앙처리장치

- 프로세서(Processor)
 - 컴퓨터 두뇌에 해당
 - 메모리에 저장된 명령어를 가져와 해독하고 실행하는 기능
 - 프로세서 내부의 장치 : 내부 버스로 연결



5. 중앙처리장치

- 프로세서 3가지 구성 필수 구성요소

- ① 산술 논리 연산 장치(Arithmetic Logic Unit, ALU)

- 산술 및 논리 연산 등 기본 연산을 수행
- 그 결과를 누산기 (Accumulator, AC)에 저장

- ② 제어 장치 (Control Unit, CU)

- 메모리에서 명령어를 가져와 해독하고 실행에 필요한 장치들을 제어하는 신호를 발생

- ③ 레지스터 세트(register set)

- 프로세서 내에 존재하는 용량은 작지만 매우 빠른 메모리
- ALU의 연산과 관련된 데이터를 일시 저장하거나 특정 제어 정보 저장
- 각 레지스터는 하나의 값을 저장할 수 있음
- 고속으로 읽고 쓸 수 있음
- 목적에 따라 특수 레지스터와 범용 레지스터로 분류

- 범용레지스터

- 연산에 필요한 데이터나 연산 결과를 임시로 저장하는 레지스터

- 특수 목적 레지스터

- 특별한 용도로 사용하는 레지스터로, 용도와 기능에 따라 구분되는 레지스터

5. 중앙처리장치

- 산술 논리 연산 장치(Arithmetic Logic Unit, ALU)
 - 산술 연산 + 논리 연산
 - 각종 산술 연산들과 논리 연산들을 수행하는 회로들로 이루어진 하드웨어 모듈
 - 주로 정수 연산을 처리
 - 산술 연산
 - 가산기를 기반으로 동작 덧셈, 뺄셈(2의 보수), 곱셈, 나눗셈(덧셈과 시프트를 반복하여 계산), 증가, 감소, 보수
 - 논리 연산
 - AND, OR, NOT, XOR, 시프트(shift)

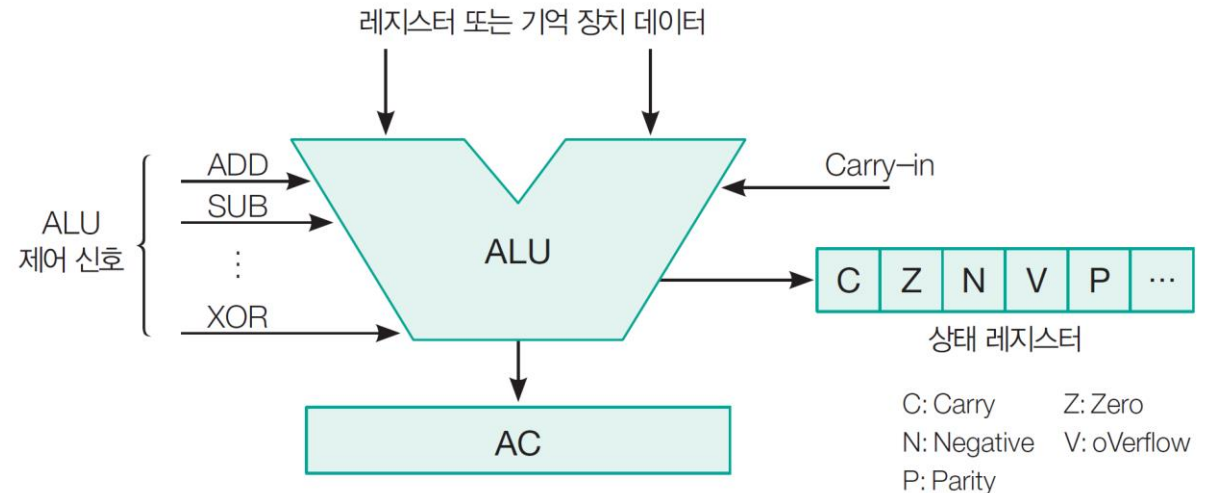


그림 4-4 ALU의 동작

5. 중앙처리장치

- 산술 논리 연산 장치(Arithmetic Logic Unit, ALU)
 - 산술 연산 + 논리 연산
 - 각종 산술 연산들과 논리 연산들을 수행하는 회로들로 이루어진 하드웨어 모듈
 - 주로 정수 연산을 처리
 - 산술 연산
 - 가산기를 기반으로 동작 덧셈, 뺄셈(2의 보수), 곱셈, 나눗셈(덧셈과 시프트를 반복하여 계산), 증가, 감소, 보수
 - 논리 연산
 - AND, OR, NOT, XOR, 시프트(shift)

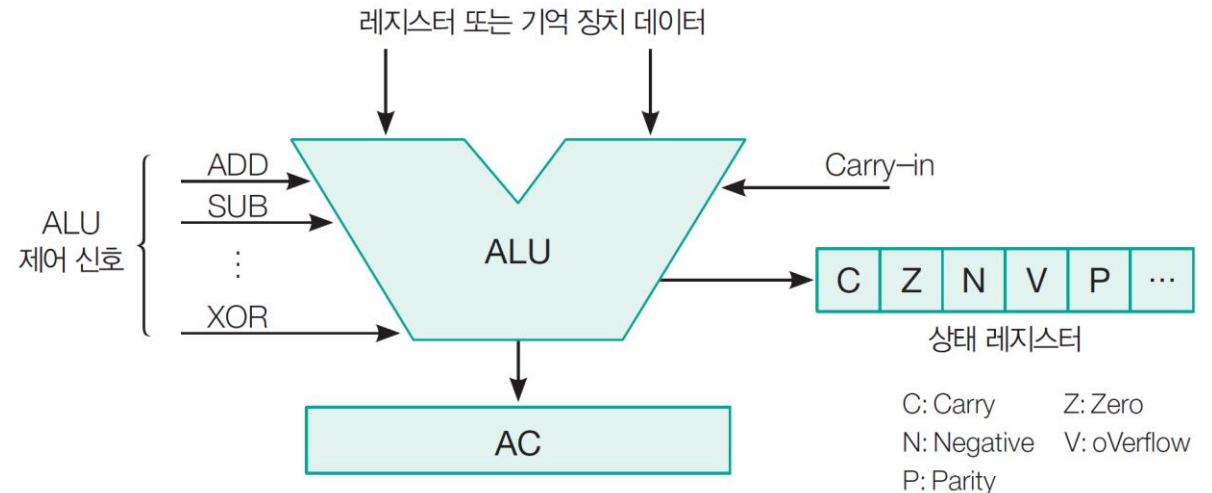


그림 4-4 ALU의 동작

5. 중앙처리장치

- 산술 논리 연산 장치(Arithmetic Logic Unit, ALU)

- 논리연산

- AND, OR, NOT, XOR

A	B	NOT A	NOT B	A AND B	A OR B	A XOR B
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	1	0

- 산술연산

- 덧셈, 뺄셈, 곱셈, 나눗셈 등

5. 중앙처리장치

- 레지스터 세트
 - 레지스터는 CPU가 사용하는 데이터와 명령어를 신속하게 읽어 오고 저장하고 전송하는 데 사용
 - 주기억 장치에서 데이터를 읽어와 레지스터에 저장
 - 처리를 마치면 주기억 장치에 그 결과를 다시 저장
 - 레지스터의 개수는 시스템의 크기나 용도에 따라 다름
 - CPU의 임시 기억 장소로 명령, 주소, 데이터 등을 일시적으로 담아둠
 - 레지스터는 메모리 계층의 최상위에 있으며 시스템에서 가장 빠른 메모리
- 레지스터 용도에 따른 종류
 - 누산기(Accumulator, AC)
 - 프로그램 카운터(Program Counter, PC)
 - 명령 레지스터(Instruction Register, IR)
 - 인덱스 레지스터(Index Register, IX)
 - 스택 포인터(Stack Pointer, SP)
 - 메모리 데이터 레지스터(Memory Data Register : MDR, Memory Buffer Register : MBR)
 - 메모리 주소 레지스터(Memory Address Register, MAR)

5. 중앙처리장치

• 제어장치

- 컴퓨터의 모든 동작을 제어하는 CPU의 핵심 장치
- 프로그램에서 주어진 명령을 '해석','실행' 하기 위해 제어신호를 만듦
- ALU, I/O 장치에 프로세서가 전송한 명령어 수행
- 주기억 장치의 명령어를 읽어 CPU의 명령 레지스터 IR로 가져오고, 제어 신호를 발생

• 제어 장치의 기본 기능

- CPU에 접속된 장치들에 대한 데이터 이동 순서 조정
- 명령어 해독
- CPU 내 데이터 흐름 제어
- 외부 명령을 받아 일련의 제어 신호 생성
- 실행 장치(예를 들어 ALU, 데이터 버퍼, 레지스터) 제어
- 명령어 인출, 명령어 해독, 명령어 실행 등을 순서에 맞추어 처리

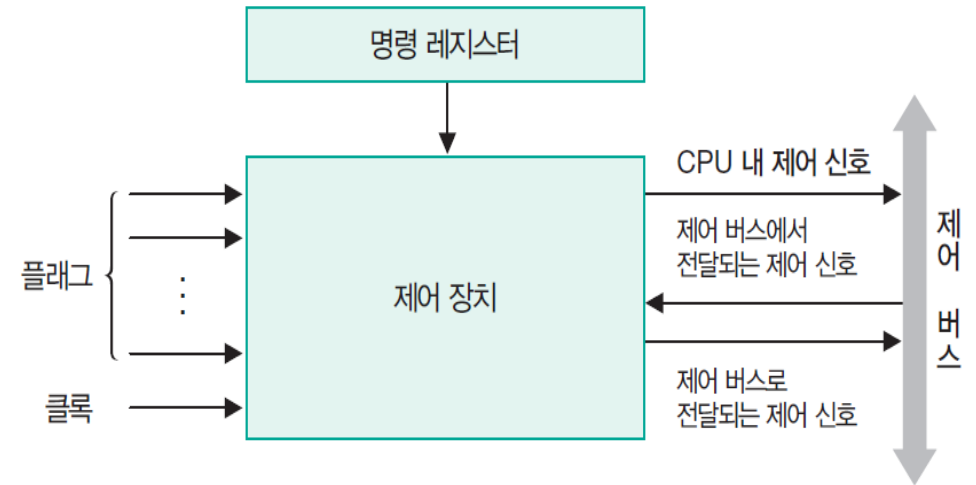


그림 5-1 제어 장치의 기능

6. 기억장치

- 기억장치 위치에 따른 분류
 - 위치는 컴퓨터 내부와 외부로 구분하는 기준
 - CPU 내부의 레지스터와 주기억 장치는 내부 기억 장치이고, 자기 디스크와 자기 테이프는 외부 기억 장치
- 용량에 따른 분류
 - 용량(capacity) : 기억 장치가 저장할 수 있는 데이터의 총량으로 바이트나 워드로 나타냄.

6. 기억장치

- RAM(Read Access Memory)
 - 전원이 꺼지면 저장 내용이 지워지는 휘발성 메모리
 - RAM은 휘발성이어서 사용하려면 전원을 계속 공급해야 하므로 일시적인 저장 장치로만 활용됨
 - RAM은 데이터의 읽기와 쓰기가 모두 가능함
 - CPU가 지정하는 주소에 있는 정보를 직접 액세스할 수 있어 메모리의 위치에 관계없이 액세스 시간이 동일함
- ROM(Read Only Memory)
 - 저장된 데이터를 읽을 수는 있으나, 별도의 장치 없이는 변경할 수 없음
 - 전원이 꺼져도 정보가 없어지지 않는 비휘발성 메모리
 - 변경할 필요가 없는 프로그램이나 데이터를 저장하는 데 사용됨
 - 운영체제를 가동하기 전에 컴퓨터의 각 구성요소를 점검하기 위한 기본 정보들이 들어 있음(BIOS 정보)
 - 컴퓨터 시스템에서는 RAM과 함께 주기억 장치의 일부분으로 ROM을 사용하고 있음

6. 기억장치

- 플래시메모리(Flash memory)
 - 전원이 꺼져도 저장된 정보가 사라지지 않는 비휘발성 메모리
 - ROM의 장점과 정보의 입출력이 자유로운 RAM의 장점을 동시에 지닌 반도체 메모리
 - 속도가 빠르며 전력소모가 적음
 - 2001년부터 USB 드라이브 이름으로 소개되었으며, 이후 디지털 캠코더, 휴대폰, 디지털 카메라 등의 휴대용 디지털 기기에 사용되면서 사용량이 급격히 증가하였음
- SSD(Solid State Drive)
 - HDD와 달리 자기디스크가 아닌 반도체를 이용해 데이터를 저장
 - HDD보다 빠른 속도로 데이터의 읽기나 쓰기가 가능
 - 물리적으로 움직이는 부품이 없기 때문에 작동 소음이 없으며 전력소모도 적음

6. 기억장치

• 캐시메모리

❖ 캐시의 사용 목적

- CPU와 주기억 장치의 속도 차이로 인한 CPU 대기 시간을 최소화 시키기 위하여 CPU와 주기억 장치 사이에 설치하는 고속 반도체 기억장치
- 주기억 장치의 호출 시간을 단축하고, 중앙 처리 장치(CPU)의 처리 능력을 향상시키기 위한 소용량이며 고속인 기억장치

❖ 캐시의 특징

- 주기억 장치보다 액세스 속도가 높은 칩 사용
- 가격 및 제한된 공간 때문에 용량이 적음

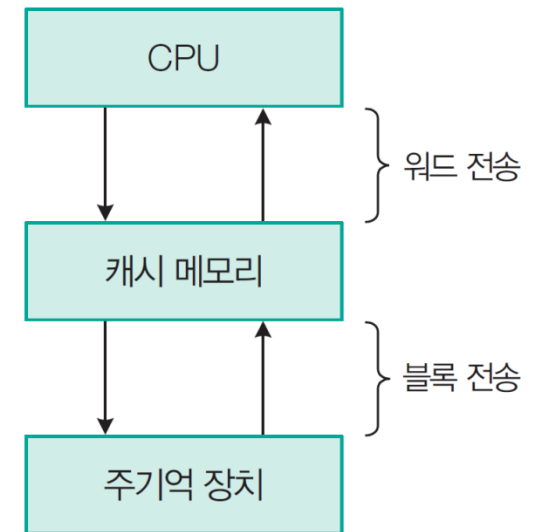


그림 6-20 캐시 위치

6. 기억장치

- 캐시의 기본적인 동작 흐름도

- CPU가 기억 장치에서 어떤 정보(명령어 또는 데이터)를 읽으려는 경우 먼저 해당 정보가 캐시에 있는지 검사
- 정보가 캐시기억장치에 있을 경우 해당 정보를 즉시 읽어 들이고, 없다면 해당 정보를 주기억 장치에서 캐시로 적재한 후 읽어 들임

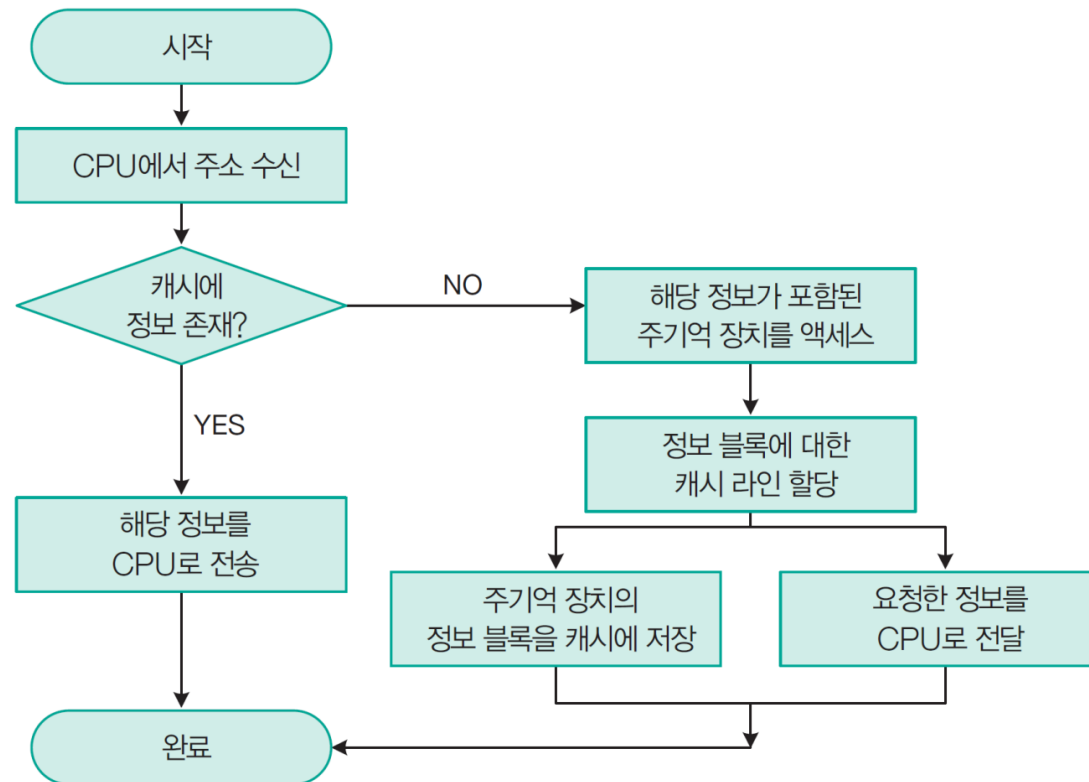


그림 6-21 캐시 읽기 동작 순서

Chap3. 정보통신

- 정보통신의 정의 및 특징
- 정보시스템의 구성요소
- 정보전송 방식과 기술
- 오류처리 방식
- 통신망 형태
- 네트워크 장비
- 4차 산업혁명 시대의 정보통신 기술과 서비스

1. 정보통신의 정의 및 특징

- 정보통신 기술(ICT)

- 정보처리기술(IT) + 통신 기술(CT)
- 여러 단말장치를 통신망에 접속하여 데이터를 전송하고 처리하며 교환하는 통신 체계
- 초기에는 독립적으로 발전, 점차 두 기술이 융합되어 발전됨

- ① 정보처리 기술

- 데이터를 의미 있는 정보로 변환하거나 데이터를 지식으로 변환하기 위해 컴퓨터를 이용하여 처리하는 것
- 데이터 : 현실세계를 단순히 관찰, 측정하여 수집하거나 생산한 사실이나 측정치
- 정보 : 데이터를 가공하거나 변환하여 얻은 결과물로 의사결정을 하는데 도움이 되는 가치 있는 데이터



- ② 통신기술

- 멀리 떨어진 정보원과 정보 목적지 사이에서 정보를 전송하는 기술
- 정보원 / 목적지 : 컴퓨터, 스마트폰, 가전제품 등
- 통신회선(유선/무선)을 이용하여 컴퓨터와 컴퓨터 사이에서 정보를 교환하며, 컴퓨터끼리 다양한 망으로 연결

2. 정보통신 시스템의 구성요소

• 정보통신 시스템의 정의

정보(문자, 음향, 영상 등)를 효과적으로 수집·가공하여 전송하기 위한 기기 및 조직화된 시스템
멀리 떨어진 정보원과 정보 목적지 사이에서 정보를 전송·처리하려고 여러 구성요소를 상호 유기적으로 결합한 시스템

• 정보통신 시스템의 특징

- 통신 회선의 효율성 제공하여 비용 절감
- 고속 전송과 고품질 전송
- 오류 제어를 통해 신뢰도를 높임
- 대형 컴퓨터와 대용량 파일 공동 이용 할 수 있음
- 분산처리 가능
- 대용량/광대역 전송 가능
- 거리와 시간의 한계 극복
- 비밀 유지를 위한 보안 시스템 개발 필요

• 정보통신 서비스 예

- 전자우편
- 음성 메일
- 팩스
- TV방송, 영상회의
- 원격회의, 원격교육, 스마트폰 통신 등

3. 정보전송 방식과 기술

- 통신 회선의 접속 방식
 - 컴퓨터와 단말기를 연결하는 방식에 따른 분류

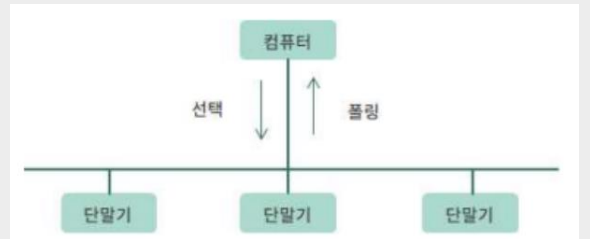
포인트 투 포인트 방식

컴퓨터 시스템과 단말기, 또는 단말간 일대일로 독립적으로 연결하는 방식
응답속도가 빨라 주로 고속 처리에 이용
전송할 양이 많을 때 적합



멀티 투 포인트 방식

컴퓨터 시스템에 연결된 전송회선 1개에 여러 대의 단말기들을 연결하는 방식
회선 공유로 효율도가 높음
비교적 전송량이 적을 때 사용하고 가격이 저렴





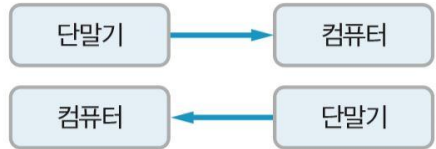
회선 다중 방식

여러 대의 단말기들을 다중화 장치를 이용하여 중앙 컴퓨터와 연결하는 방식
다중화 장치와 컴퓨터 사이에는 대용량 회선으로 연결



3. 정보전송 방식과 기술

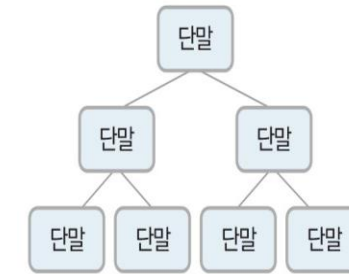
- 통신회선의 이용방식
 - 통신회선을 이용하는 방식에 따른 분류

단방향 통신 Simplex	데이터를 한쪽 방향으로만 전송이 가능한 방식 라디오, TV	
반이중 통신 Half-Duplex	양방향 전송이 가능한 방식 동시에 양쪽 방향에서 전송할 수 없는 방식 무전기, 모뎀을 이용한 데이터 통신	
전이중 통신 Full-Duplex	동시에 양방향 전송이 가능한 방식 고속으로 처리할 수 있음 전송량이 많고, 전송 매체의 용량이 클 때 사용 전화	

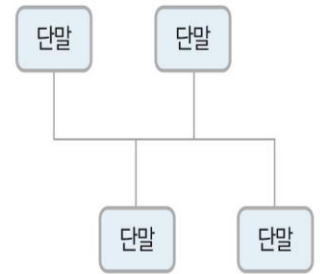
3. 정보전송 방식과 기술

- 통신회선망의 구성 형태
 - 단말기를 컴퓨터와 서로 밀접하게 결합한 형태(단말기 또는 컴퓨터)
 - 통신 네트워크라고도 함

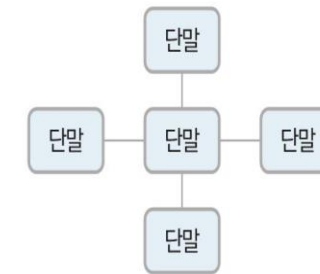
성형 (중앙집중형)	중앙에 컴퓨터가 있고, 이를 중심으로 단말기들이 연결되어 있는 중앙 집중식 형태 단말기의 추가와 제거가 쉽고, 교환 노드의 수가 가장 적음 중앙 단말에 오류 등 장애가 발생하면 전체 시스템에 영향을 미침
링형	컴퓨터와 단말기들을 서로 이웃하는 것끼리 포인트 투 포인트 방식으로 연결 컴퓨터, 단말기, 통신 회선 중 어느 하나라도 고장나면 전체 통신망에 영향을 미침
버스형	한 개의 통신 회선에 여러 대의 단말기가 연결되어 있는 형태 물리적 구조가 간단, 단말기의 추가와 제거가 용이
계층형	각 단말기가 계층적으로 구성되는 것 중앙 컴퓨터와 일정 지역의 단말기까지 하나의 통신 회선으로 연결 분산처리 시스템에 효율적
망형	모든 지점의 컴퓨터와 단말기를 서로 연결한 형태 많은 단말기로부터 많은 양의 통신을 필요로 하는 경우 유리 통신 회선 장애 시 다른 경로를 통하여 데이터를 전송



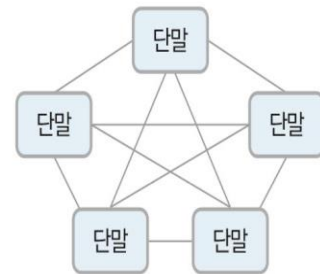
(a) 트리형



(b) 버스형



(c) 성형



(d) 망형

3. 정보전송 방식과 기술

• 데이터 전송방식

- 데이터를 보내는 방식에 따른 분류
- 컴퓨터는 정보를 전송할 때 1byte나 그 정수 배의 단위로 전송

① 직렬전송방식

- 하나의 전송 매체를 통해 1byte를 한 비트씩 순서적으로 전송되는 형태
- 순차적 전송으로 인해 전송 속도가 느림
- 바이트 문자열을 비트열로 바꿔주는 병렬-직렬 변환기
- 수신할 때는 비트열을 바이트 문자열로 바꿔주는 직렬-병렬 변환기 필요
- 설치비용 저렴 및 설치방법 간단
- 원거리 전송에 적합

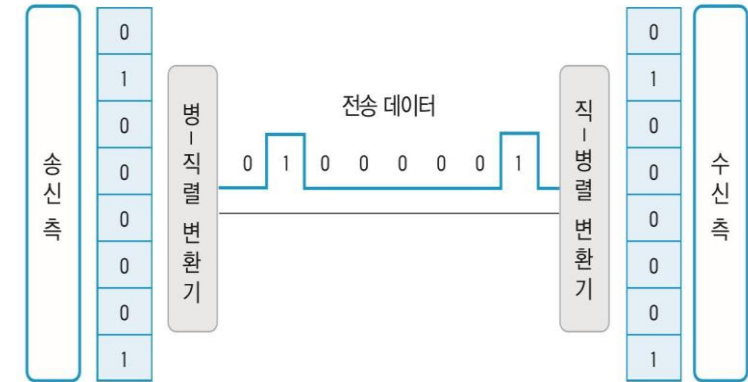


그림 4-16 직렬 전송 방식

② 병렬전송방식

- 문자를 구성하는 각 비트를 전송선로 7~8개를 이용해 동시에 전송하는 방식
- 전송속도가 빠르고 단말기와도 쉽게 연결할 수 있어 편리
- 근거리 전송에 적합
- 컴퓨터와 하드 디스크를 연결시 사용
- 직렬과 병렬 변환 회로가 필요 없음

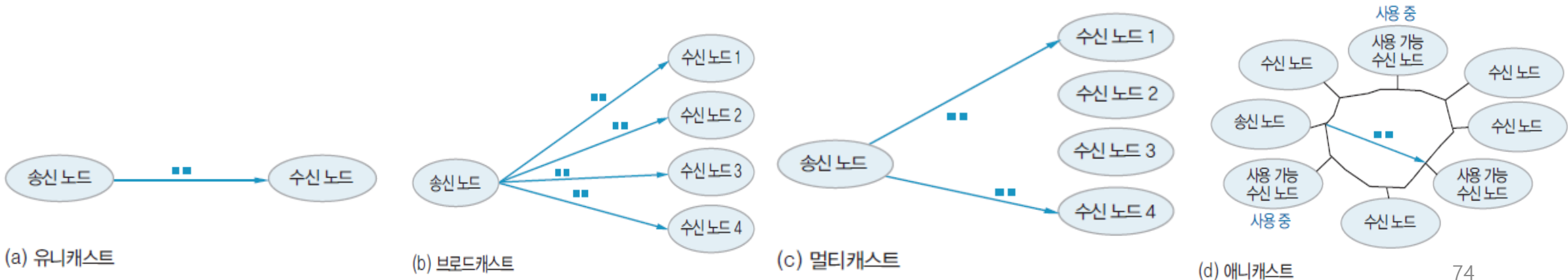


그림 4-17 병렬 전송 방식

3. 정보전송 방식과 기술

- 캐스팅 모드의 전송방식

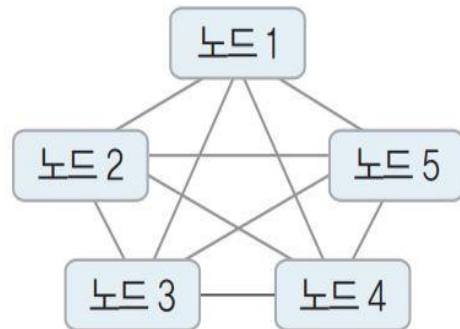
- 통신에 참여하는 송신자와 수신자의 수에 따른 전송 방식
- 유니캐스트 (Unicast)
 - 정보를 송수신할 때 송신 노드와 수신 노드가 하나인 일-대-일 방식
- 브로드캐스트 (Broadcast)
 - 하나의 송신 노드 하나가 네트워크에 연결된 수신 가능한 모든 노드에 데이터를 전송하는 방식(라디오/TV)
- 멀티캐스트(Multicast)
 - 하나의 송신 노드가 네트워크에 연결된 하나 이상의 특정 수신 노드(미리 정해 둠)에 데이터를 전송하는 일-대-다 방식
 - 전자우편 서비스에서 주소록에 등록된 사람에게 한번에 메일 전송
- 애니캐스트(Anycast)
 - 송신 노드가 네트워크에 연결된 수신 가능한 노드 중에서 한 노드에만 데이터를 전송하는 방식
 - 프린터가 여러 대 연결된 네트워크의 경우 현재 프린트 중인 프린트 서버를 피해 다른 프린트를 사용하는 경우



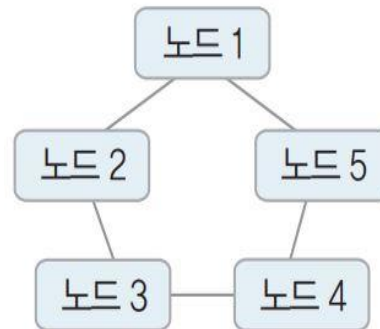
5. 통신망 형태

• 정보통신망의 개념

- 정보통신 시스템에서 정보(텍스트, 이미지, 음성 등)를 효율적으로 전송하기 위해 통신장비(컴퓨터 시스템, 단말기 등)를 상호 유기적으로 결합한 것
- 하나의 회선에 여러 시스템을 연결하거나 몇 개의 회선을 공유하는 방식으로 구성됨
- 통신 비용을 절감하고 정보를 효율적으로 전송하는게 주요 목적임



(a) 기본적인 형태의 통신망



(b) 링 형태의 통신망

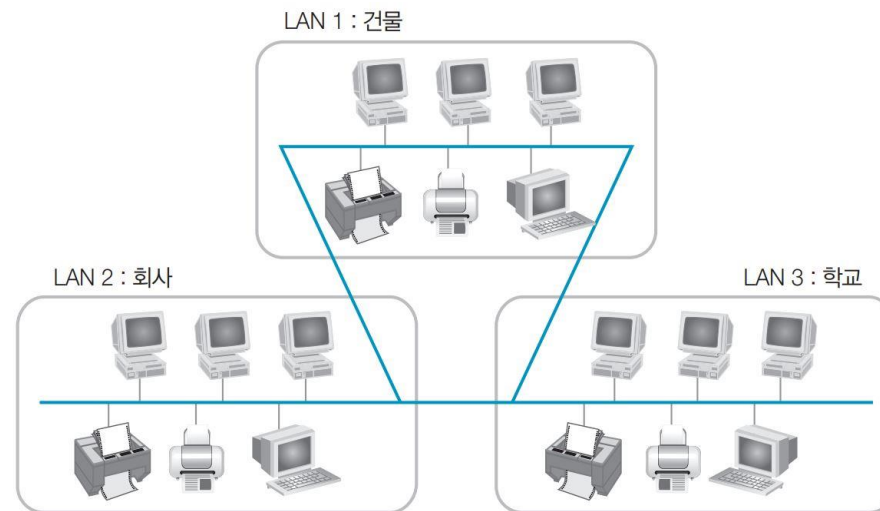


비용, 통신망의 목적을
충분히 고려하여 형태를 적용

그림 6-1 통신망의 다양한 형태

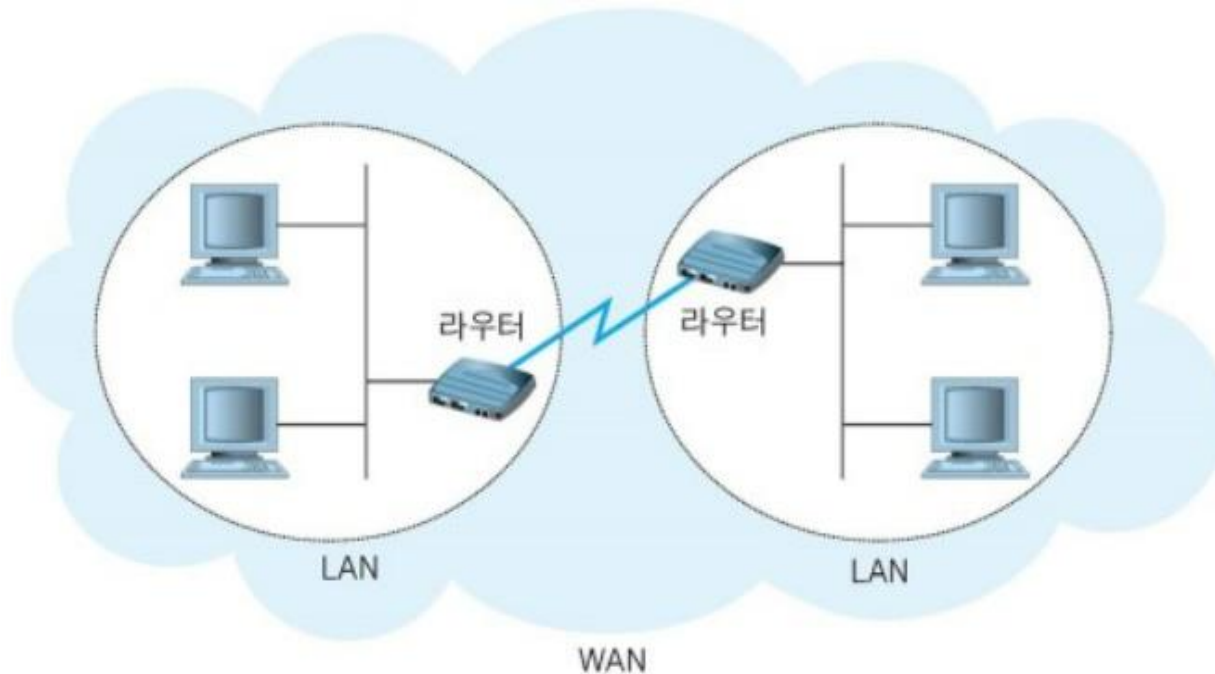
5. 통신망 형태

- LAN(Local Area Network, 근거리 통신망)
 - 가까운 거리에 있는 컴퓨터와 통신장비들을 연결하여 고속통신이 가능한 통신망
 - 컴퓨터와 그 주변 장치, 전화기, 팩시밀리, TV 수상기 등 통신 기능이 있는 기기는 모두 연결 가능(자원 공유)
 - 통신에 적합한 지역에서만 제한적으로 사용 가능(1~20km)
 - 데이터 전송속도가 매우 빠른 통신매체로 구성(1~20Mbps)
 - 전송 특성이 좋은 매체를 사용하며, 오류 발생률이 매우 낮아 신뢰성 있는 정보 전송이 가능
 - 하나의 통신망을 이용하여 텍스트, 음성, 이미지를 모두 전송하여 정보를 종합적으로 처리 가능



5. 통신망 형태

- WAN(Wide Area Network, 광역 통신망)
 - 둘 이상의 LAN이 넓은 지역에 걸쳐 연결되어 있는 네트워크
 - 도시와 도시 간, 국가와 국가 간 등 원격지 사이를 연결하는 통신망
 - 범위는 보통 10km 이상



6. 네트워크 장비

• 허브

- 가까운 거리의 컴퓨터들을 연결하는 장치
- 한 가운데에 있는 제어장치를 중심으로, DTE가 있는 지점 간에 트리 구조로 연결하는 장비
- LAN끼리 연결하는 경우에도 사용
- 수신한 신호를 정확히 재생하여 다른 쪽으로 내보내는 역할
- 초기의 허브는 신호를 증폭하고 재생하며 각 노드를 집중화 시켜주는 역할(초기 더미 허브)
- 최근에는 망을 관리할 수 있는 수준인 지능형 허브로 발전(스위칭 허브)

• 리피터

- 단말기 사이의 거리가 멀어질수록 감쇄되는 신호를 재생시키는 장비
- 서로 분리된 동일한 LAN에서 네트워크의 거리를 연장할 때 사용

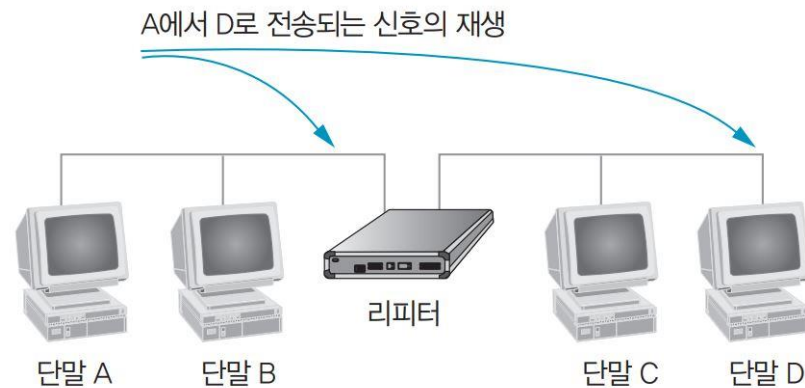


그림 6-16 리피터의 연결 예

6. 네트워크 장비

- 라우터

- 서로 다른 네트워크를 연결해주는 장치(각각의 네트워크는 통신방법이나 신호가 다르기 때문에 여러가지 네트워크들이 정보를 주고받기 위해 필요)
- 임의의 외부 네트워크와 내부 네트워크를 연결해 주는 장비
- 구조가 다른 망끼리도 연결할 수 있어 근거리 통신망과 광역통신망을 연결하는 데 주로 사용
- 데이터가 전송될 수 있는 여러 경로 중 가장 적절한 전송 경로를 선택하는 역할

- 게이트웨이

- 한 네트워크에서 다른 네트워크로 이동하기 위해 거쳐야 하는 지점
- 이기종 네트워크를 연결하고 프로토콜 구조가 전혀 다른 네트워크의 프로토콜이 다를 경우 중재 역할을 해줌

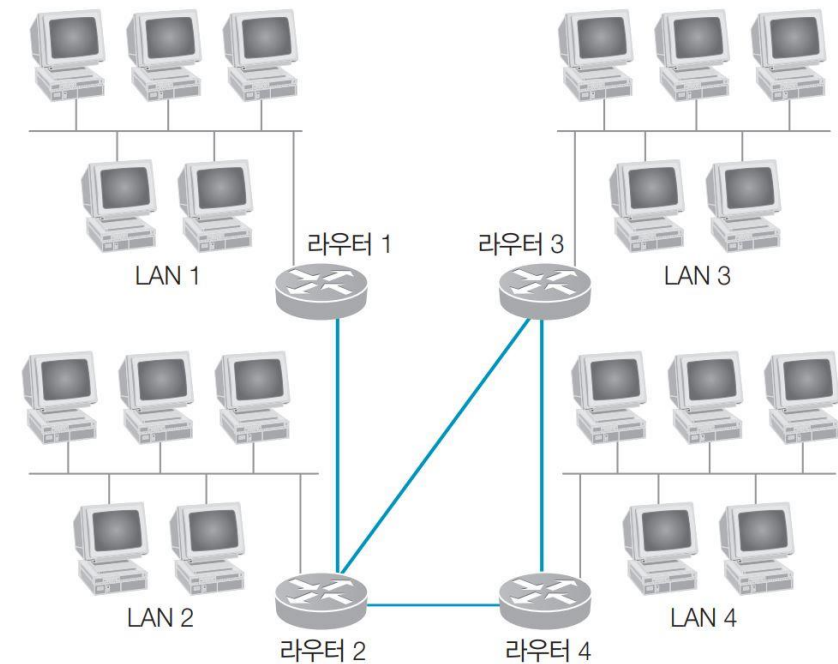


그림 6-20 라우터의 연결 예

Chap4. 네트워크

- 컴퓨터 네트워크 기초
- OSI 7Layer
- TCP/IP 4계층 모델
- 주소의 표현
- IPv4, IPv6

1. 컴퓨터 네트워크 기초

- 컴퓨터 네트워크 정의
 - 다수의 시스템을 전송매체로 연결해 구성한 시스템들의 집합체
 - 컴퓨터들 간에 정보 또는 데이터를 전달하기 위해 컴퓨터들을 서로 연결한 것
 - 다른 여러 호스트를 서로 연결해서 통신하기 위해 연결방식의 표준화가 필요함
- 국제 표준화 단체인 ISO에서 OSI7Layer(Open System Interconnection)모델을 제안
 - 표준 프로토콜을 사용하여 다양한 통신 시스템이 통신할 수 있도록 국제표준화기구에서 만든 개념 모델
 - 이기종 컴퓨터 시스템이 서로 통신할 수 있는 표준을 제공
 - 네트워크에 연결된 시스템이 갖추어야 할 기능을 상세히 정의하고 있음
 - 각 계층은 독립적인 고유 기능을 수행

1. 컴퓨터 네트워크 기초

- 여러 시스템이 전송 매체로 연결되어 네트워크를 구성한 예

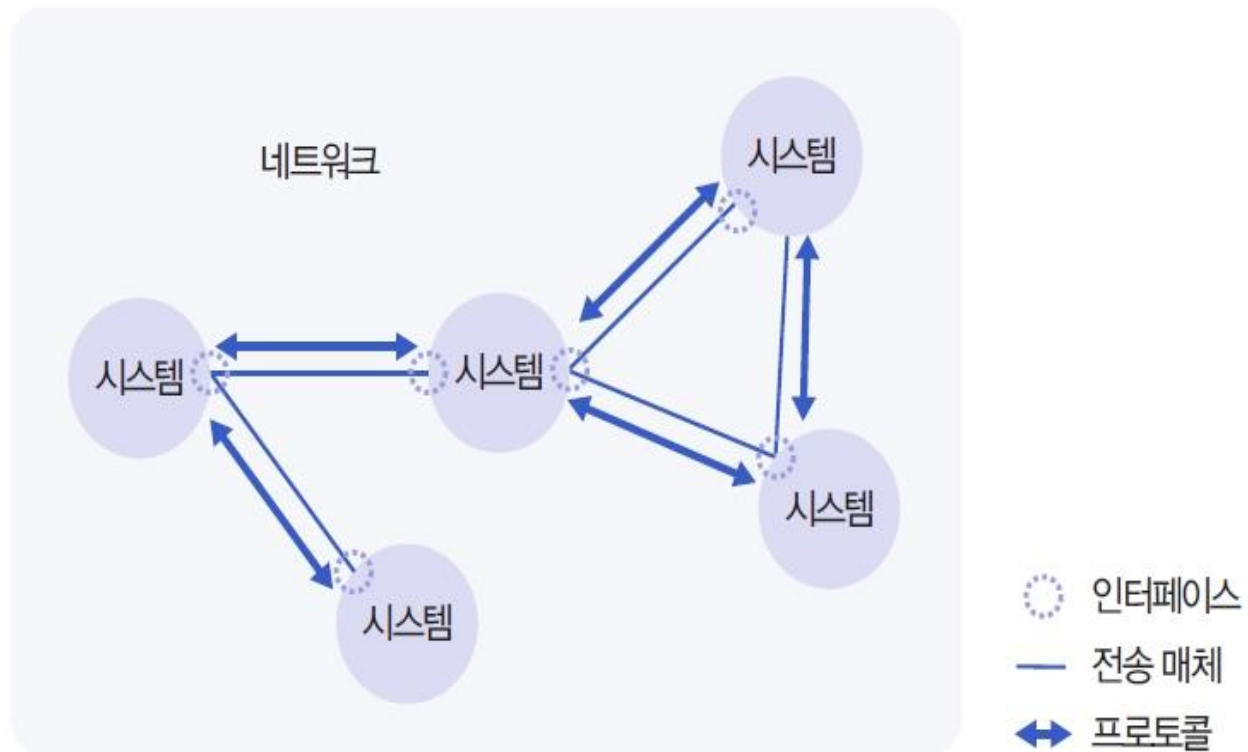


그림 1-1 네트워크의 구성

시스템이 전송매체를 통해 데이터를 교환하기 위해

표준화된 프로토콜을 사용해야 함

(ex)인터넷은 IP라는 프로토콜을 사용하는 네트워크 집합체

네트워크 = 시스템 + 전송매체 + 인터페이스+프로토콜

네트워크를 이해하기 위해 선행되어야 하는 개념

- 시스템
- 인터페이스
- 전송매체
- 프로토콜
- 네트워크
- 인터넷

1. 컴퓨터 네트워크 기초

- 네트워크 기초 용어

- 네트워크

- 전송 매체를 매개로 서로 연결되어 데이터를 교환하는 시스템의 모음
 - 여러 시스템이 프로토콜을 사용하여 데이터를 주고 받는 것을 의미
 - 네트워크끼리 라우터라는 중개 장비를 이용해 연결

- 시스템

- 능동적으로 동작하는 대상
 - 시스템에 필요한 외부 입력과 결과물인 출력이 있을 수 있으며, 물리적으로 공유하는 전송 매체에 의해 서로 연결됨

- 인터페이스

- 시스템과 전송 매체의 연결 지점에 대한 규격
 - 예: RS-232C, USB(잭의 크기와 같은 규격이 표준화 되어야 함)

- 전송 매체

- 시스템끼리 정해진 인터페이스를 연동해 데이터를 전달할 때 필요한 물리적인 전송 수단
 - 유선매체 : 물리적 경로를 따라 전달(꼬임선, 동축케이블, 광섬유 케이블)
 - 무선매체 : 물리적 경로를 따르지 않고 공간을 매개로 전달(라디오파, 마이크로파-휴대전화, 위성통신, 무선LAN)

1. 컴퓨터 네트워크 기초

- 네트워크 기초 용어

- 프로토콜

- 전송 매체를 통해 데이터를 교환할 때의 표준화된 통신 규칙
 - 컴퓨터나 원거리 통신 장비 사이에서 메시지를 주고 받는 양식과 규칙 체계

- 인터넷(Internet)

- 전세계의 네트워크가 유기적으로 연결되어 동작하는 통합 네트워크
 - IP(Internet Protocol)를 사용

2. OSI 7Layer

• 계층 구조

- 서로 다른 기종의 호스트들이 서로 연결해서 통신하기 위한 표준화 방식인 OSI(Open System Interconnection) 7계층 모델을 제안함
- OSI 7계층 모델은 네트워크에 연결된 시스템이 갖추어야 할 기본 구조와 기능을 상세히 정의(포트, 프로토콜 등)
- 문제 발생 시 각 계층별로 원인이 어디인지를 쉽게 파악할 수 있음
- 일반 사용자는 OSI 7계층 맨 위에 있는 응용 계층을 통해 데이터의 송수신을 요청
- 요청은 하위 계층에 순차적으로 전달되어 맨 아래에 있는 물리 계층을 통해 상대 호스트에 전송
- 데이터를 수신하는 호스트에서는 송신 호스트와는 반대 방향으로 처리가 이루어짐

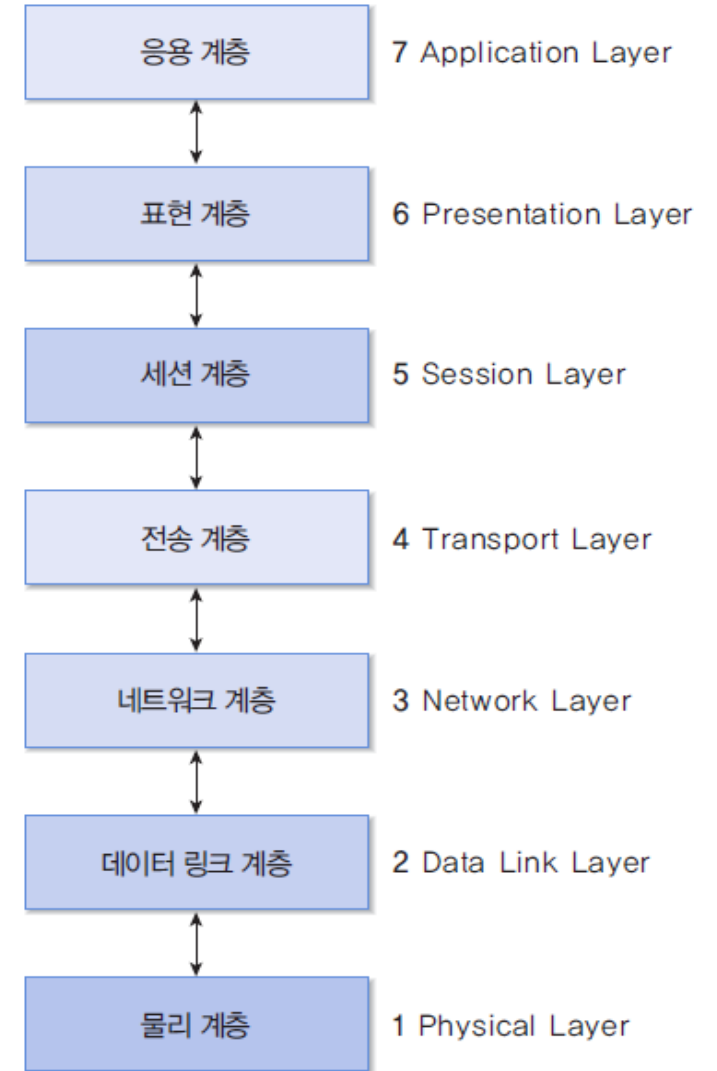
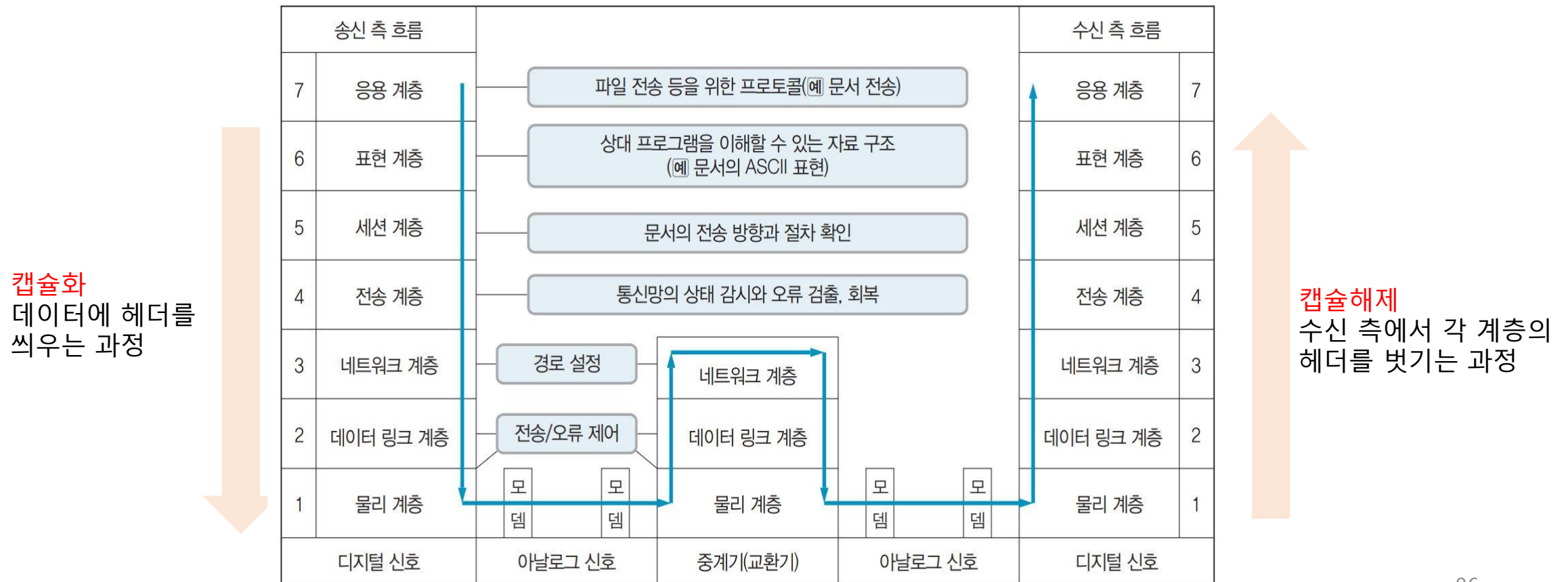


그림 1-4 OSI 7계층 모델

2. OSI 7Layer

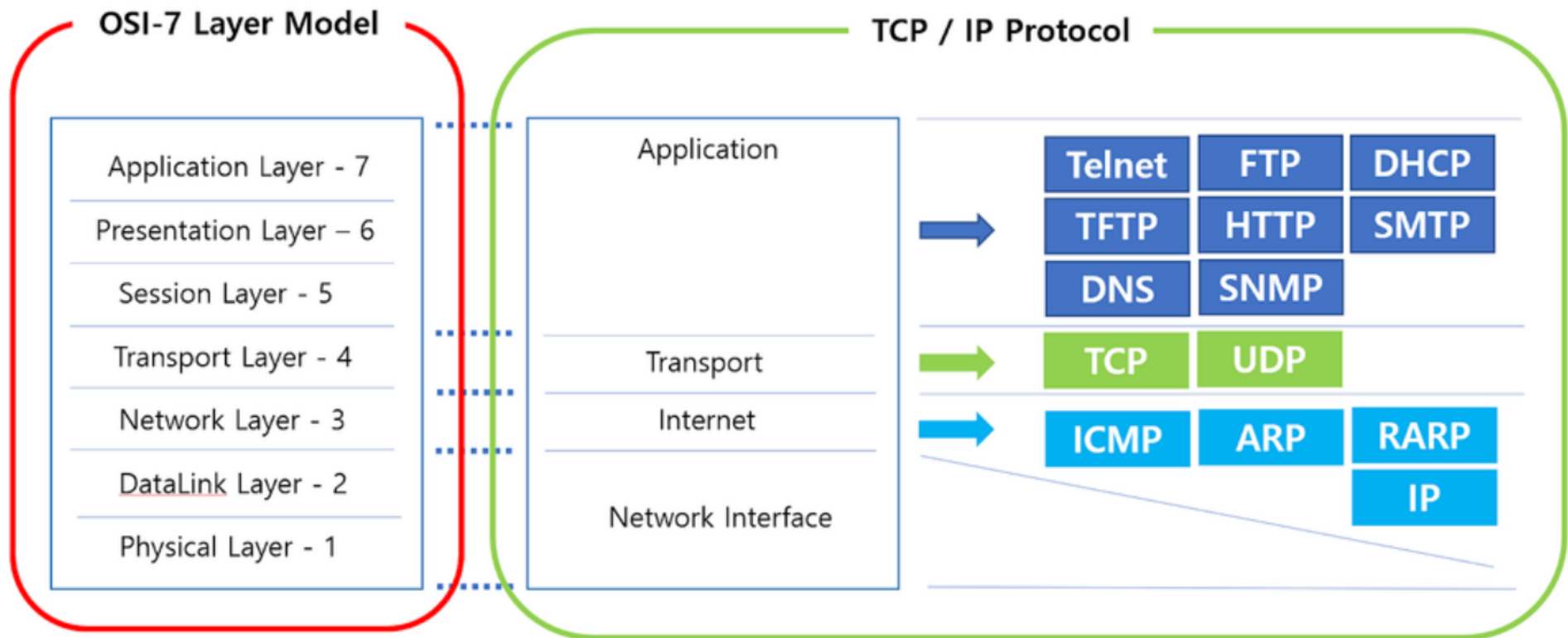
- OSI 7계층 상호 간의 데이터 전달 원리

- 캡슐화와 캡슐해제 과정을 거쳐 송신 측의 최상위 계층에서 보낸 원래의 데이터(헤더가 없는 데이터)를 수신 측의 최상위 계층(7계층 : 응용 계층)으로 정확하게 전달



3. TCP/IP 참조모델

- TCP/IP참조 모델
 - OSI7 Layer 모델을 기반으로 실무적으로 이용할 수 있도록 단순화한 모델



4. 주소의 표현

- 네트워크에 존재하는 다양한 주소
 - 네트워크 계층 : IP 주소
 - 데이터링크 계층 : MAC주소
 - 전송계층 : Port번호
 - 도메인네임
 - 이메일주소

4. 주소의 표현

① 논리적 주소(IP주소)

- 네트워크 계층의 기능을 수행하는 IP 프로토콜이 호스트를 구분하기 위해 사용하는 주소 체계
- 인터넷에서 IP 주소는 패킷의 경로를 결정하는 데 중요한 역할을 함
- IP 주소는 32비트의 이진 숫자로 구성, 보통 8비트씩 네 부분으로 나누어 십진수로 표현

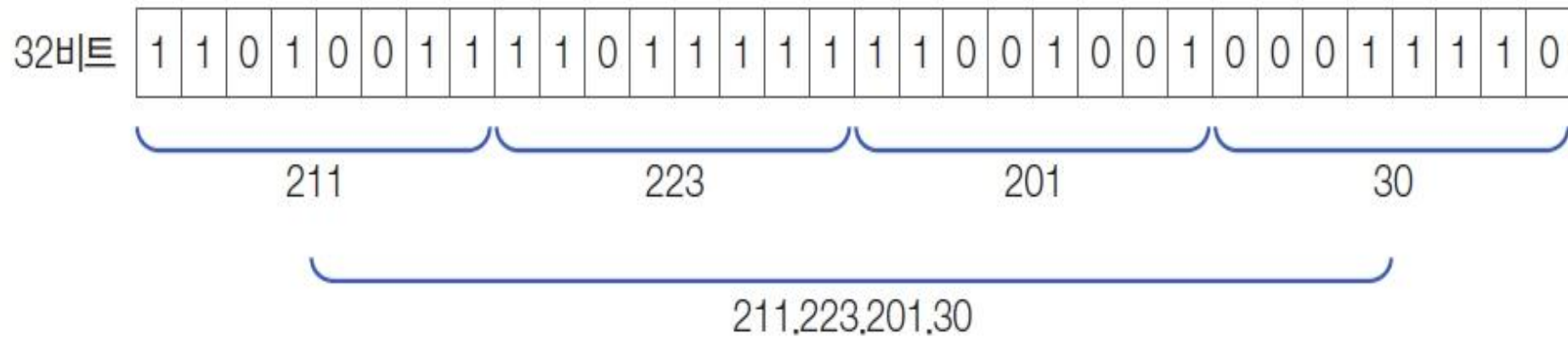


그림 1-9 IP 주소의 표현

4. 주소의 표현

② 도메인 네임

- IP주소를 쉬운 문자로 나타낸 것
- 호스트 이름
 - <호스트>.<단체 이름>.<단체 종류>.<국가 도메인>
 - 예: zebra.korea.co.kr
- 도메인 이름을 IP주소로 변환하는 작업이 필요
- DNS(Domain Name System)
 - 주소와 이름 정보를 자동으로 유지하고 관리하는 시스템
 - 호스트 주소와 이름 정보는 네임서버라는 특정한 관리 호스트가 유지
 - 주소 변환 작업이 필요한 클라이언트는 네임 서버에 요청해서 IP주소를 얻음

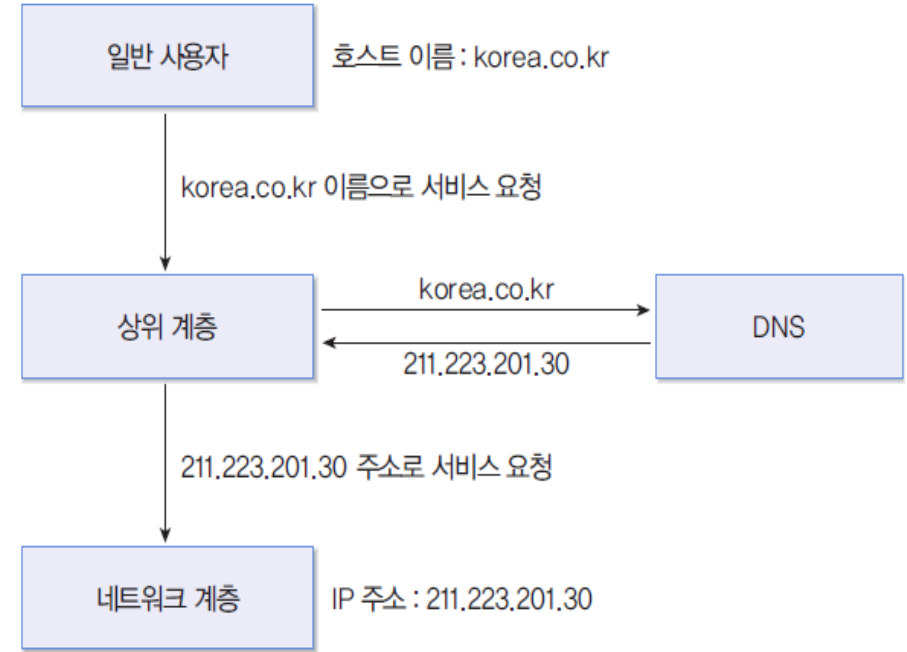


그림 1-11 호스트 이름과 IP 주소의 변환

표 1-2 국가 도메인

국가 도메인	해당 국가명
kr	한국
jp	일본
us	미국

표 1-3 단체 종류

단체 종류	기관 성격
co ^{company}	회사
ac ^{academy}	교육기관
go ^{government}	정부 소속 기관

4. 주소의 표현

③ MAC 주소

- 일반적으로 LAN카드에 내장되어 있음
- 물리 계층을 통해 데이터를 전송할 때 MAC주소를 이용해 호스트를 구분
- IP주소를 MAC 주소로 변환해야 함(ARP, RARP)
- 총48bit로 구성되고 16진수로 표현됨

④ 포트 주소(0~65,535)

- 전송계층에서 사용
- 호스트에서 실행되는 프로세스를 구분해 줌
- 하나의 포트는 하나의 프로세스만 사용 가능

(Well-Known Port)

서비스	포트번호
FTP	21
Telnet	23
SMTP	25
HTTP	80

⑤ 메일 주소

- 응용계층의 메일 시스템에서 사용자를 구분하려고 사용
- 사용자 이름과 호스트 이름을 @문자로 구분하여 표기

5. IPv4/IPv6

• IPv4 개요

- 전 세계 모든 컴퓨터에 부여된 고유의 식별 주소
- 10진수 4개와 .(점)으로 표현하며, 실제로는 32비트로 구성
- 네트워크 번호와 그 네트워크에 접속해서 부여하는 호스트 번호로 구성
- 인터넷을 이용해 발신지에서 목적지까지 전송할 수 있도록 라우팅 기능을 수행
- 차세대 버전으로 IPv6가 있으며, 보통 IP주소는 IPv4를 지칭

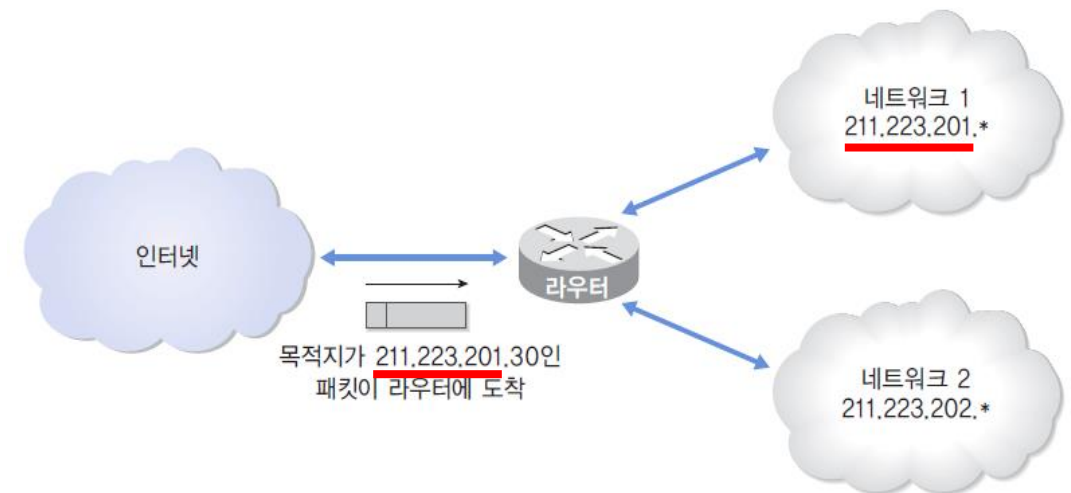


그림 1-10 라우터의 경로 선택

5. IPv4/IPv6

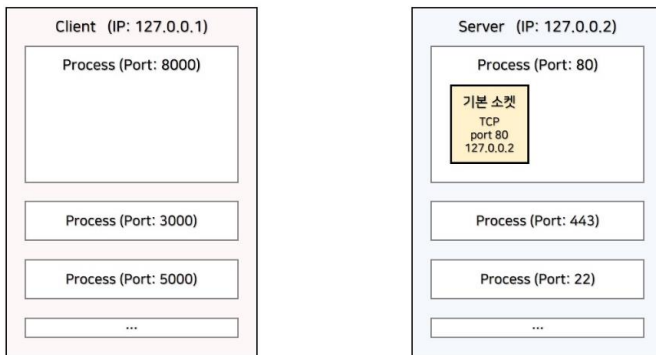
- IPv6 개요

- 현재 사용하고 있는 IP주소 체계인 IPv4의 주소 부족 문제를 해결하기 위해 개발
- IPv4에 비해 자료 전송 속도가 빠름
- 16진수로 표현되고 128bit의 주소체계
- 인증성, 기밀성, 데이터 무결성의 지원으로 보안 문제 해결
- 기본 헤더 뒤에 확장 헤더를 더함으로써 더욱 다양한 정보의 저장이 가능해짐

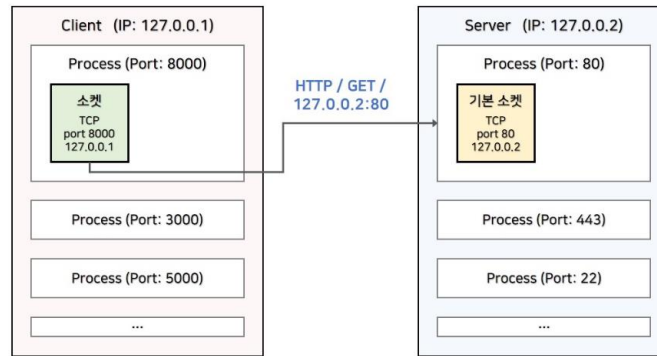
6. 소켓(Socket)

- 클라이언트와 서버의 각 프로세스가 통신으로 데이터를 주고받기 위해서 소켓이 필요함
- 클라이언트 소켓 : 클라이언트 프로세스는 반드시 소켓을 열어 소켓에 데이터를 써서 서버에 전송
클라이언트 프로세스마다 고유한 포트번호를 가짐 => 여러 개의 소켓을 가질 수 있음
- 서버 소켓 : 서버 프로세스는 소켓으로부터 받은 데이터를 읽어 사용함

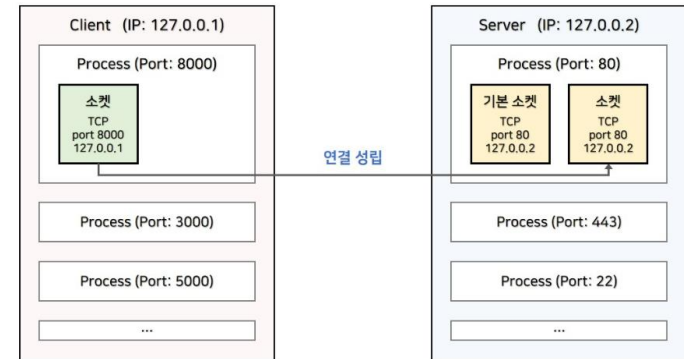
1 서버의 각 프로세스는 기본적으로 연결 요청을 듣고있는 하나의 소켓(Listen)을 소유



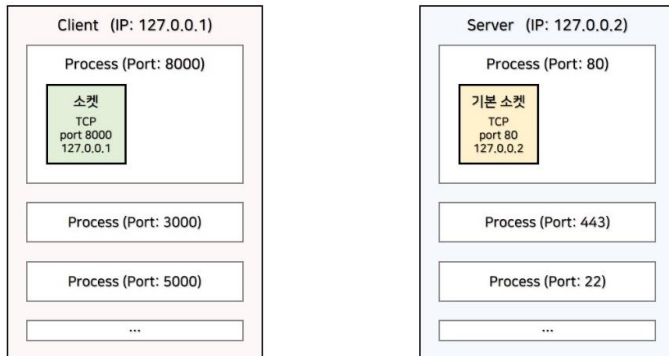
3 생성된 소켓에서 서버(127.0.0.2)의 80번 포트에 네트워크 연결 요청



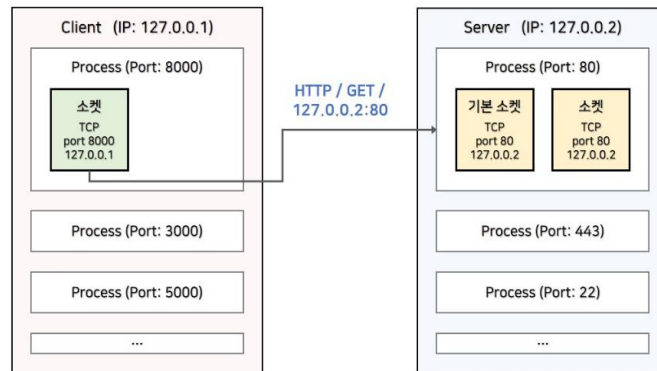
5 새로 만들어진 서버의 소켓과 클라이언트의 소켓이 연결을 성립



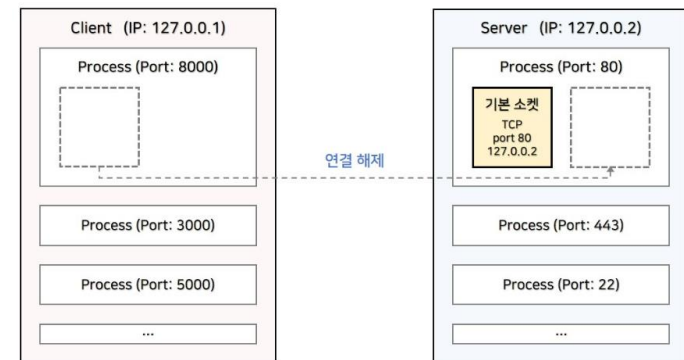
2 클라이언트(127.0.0.1)의 8000번 포트를 사용하는 프로세스에서 TCP 통신을 위해 소켓을 생성



4 서버에 기본적으로 존재하던 소켓이 요청을 받아 조건을 확인한 후 본인을 복제한 새로운 소켓을 생성

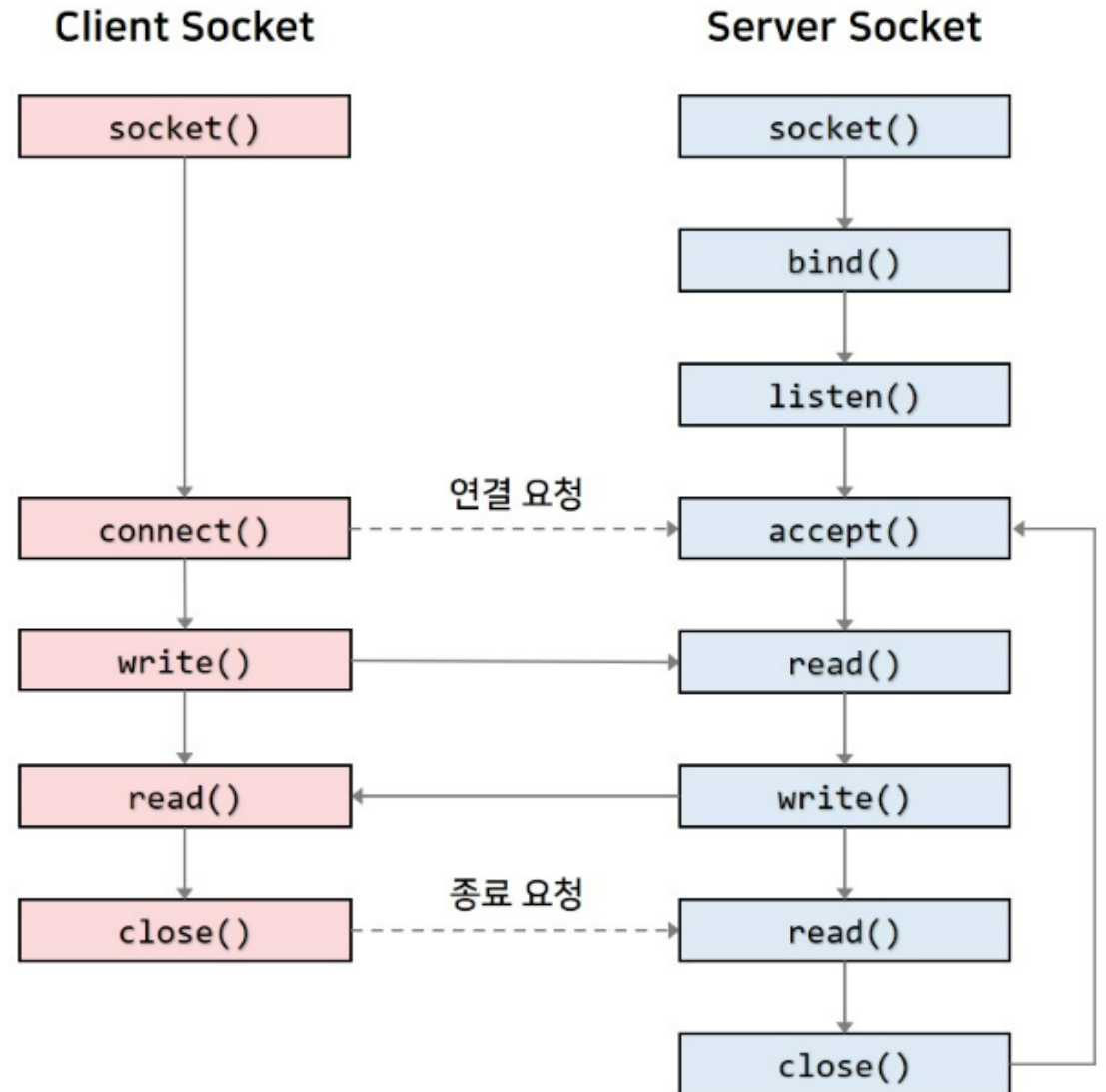


6 통신이 종료되면 통신을 위해 만들어진 소켓들은 모두 소멸



6. 소켓(Socket) 프로그래밍

- 소켓을 만들고, 통신하고, 소켓을 없애는 과정을 말함
- 클라이언트측과 서버측의 과정이 다름



6. 소켓(Socket)- 서버

- socket() : 서버의 소켓 생성

```
int    server_socket;  
server_socket = socket( PF_INET, SOCK_STREAM, 0);
```

- bind() : 클라이언트가 서버의 위치를 알기 위한 IP와 Port의 등록 작업, sockaddr_in 구조체 이용

```
struct sockaddr_in server_addr;  
memset( &server_addr, 0, sizeof( server_addr));  
server_addr.sin_family    = AF_INET;           // IPv4 인터넷 프로토콜  
server_addr.sin_port      = htons( 4000);      // 사용할 port 번호는 4000  
server_addr.sin_addr.s_addr = htonl( INADDR_ANY); // 32bit IPV4 주소  
  
if( -1 == bind( server_socket, (struct sockaddr*)&server_addr, sizeof( server_a  
ddr) ) )  
{  
    printf( "bind() 실행 에러\n");  
    exit( 1);  
}
```


6. 소켓(Socket)- 서버

- listen() : 클라이언트의 connect()함수의 접속 요청을 '확인' 하도록 요청

```
listen( server_socket, 5)
```

- accept() : 클라이언트의 요청에 대한 처리, 접속된 클라이언트와 대화하기 위한 소켓을 생성

```
int client_addr_size;  
client_addr_size = sizeof( client_addr);  
  
client_socket = accept( server_socket, (struct sockaddr*)&client_addr,&client_a  
ddr_size);
```

- recv() : 클라이언트가 send()함수를 통해 데이터를 전송한 것을 받음

```
int recv(int client_socket, void *buf, size_t len, int flags);
```

- closesocket() : 함수를 통해 서버와 클라이언트의 소켓 연결을 종료

```
close( client_socket);
```

6. 소켓(Socket)- 클라이언트

- socket() : 클라이언트의 소켓 생성

```
int    client_socket;  
client_socket = socket( PF_INET, SOCK_STREAM, 0);
```

- connect() : listen()상태의 서버에게 접속 요청을 시도, 서버의 주소와 포트번호를 지정하기 때문에 지정된 서버의 주소와 포트번호로 연결을 시도

```
struct sockaddr_in  server_addr;  
memset( &server_addr, 0, sizeof( server_addr));  
server_addr.sin_family = AF_INET;  
server_addr.sin_port = htons( 4000);  
server_addr.sin_addr.s_addr= inet_addr( "127.0.0.1"); // 서버의 주소  
if( -1 == connect( client_socket, (struct sockaddr*)&server_addr, sizeof( server_addr) ) )  
{  
    printf( "접속 실패\n");  
    exit( 1);  
}
```

6. 소켓(Socket)- 클라이언트

- recv() : 서버에서 전송하는 데이터를 받거나 send()함수를 통해서 데이터를 전송할 수 있음

```
int recv(int client_socket, void *buf, size_t len, int flags);
```

- closesocket() : 함수를 통해 서버와 클라이언트의 소켓 연결을 종료

```
close( client_socket);
```

감사합니다.