1과목 · 데이터베이스

핵심 001 정보 시스템

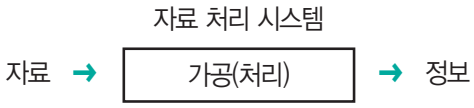
12.3, 07.9, 06.3, 05.3, 03.8, 00.5, 99.6

정보 시스템

- 조직체에 필요한 Data를 수집, 저장해 두었다가 필요시에 처리해서 의사 결정에 유용한 정보를 생성하고 분배하는 수단이다.
- 사용하는 목적에 따라 경영 정보 시스템, 군사 정보 시스템, 인사 행정 정보 시스템, 의사 결정 지원 시스템 등으로 사용된다.

정보와 자료

- 자료(Data) : 현실 세계에서 관찰이나 측정을 통해 수집한 단순한 사실이나 결과값으로, 가공되지 않은 상태
- 정보(Information) : 의사 결정에 도움을 줄 수 있는 유용한 형태로, 자료를 가공(처리)해서 얻는 결과물



- 자료 처리 시스템 : 정보 시스템이 사용할 자료를 처리하는 정보 시스템의 서브 시스템으로, 처리 형태에 따라 일괄 처리 시스템, 온라인 실시간 처리 시스템, 분산 처리 시스템으로 분류
- 데이터웨어 하우스(DataWare House) : 조직이나 기업체의 중심이 되는 주요 업무 시스템에서 추출되어 새로이 생성된 데이터베이스로서 의사결정 지원 시스템을 지원하는 주체적, 통합적, 시간적 데이터의 집합체

핵심 002 데이터베이스의 정의

14.8, 14.3, 13.6, 13.3, 12.8, 12.3, 11.6, 10.9, 09.3, 08.9, 08.3, 07.5, 07.3, 05.4, 04.5, 99.4

- 통합된 데이터(Integrated Data) : 자료의 중복을 배제한 데이터의 모임
- 저장된 데이터(Stored Data) : 컴퓨터가 실시간으로 접근할 수 있는 저장 매체에 저장된 자료
- 운영 데이터(Operational Data) : 조직의 고유한 업무를 수행하는 데 있어서 존재 가치가 확실하고 없어서는 안 될 반드시 필요한 자료
- 공용 데이터(Shared Data) : 여러 응용 시스템들이 공동으로 소유하고 유지하는 자료

핵심 003 데이터베이스의 특성

14.5, 14.3, 13.8, 13.3, 12.8, 12.5, 11.8, 11.3, 10.5, 10.3, 09.8, 09.3, 08.5, 07.5, 07.3

- 실시간 접근성(Real Time Accessibility) : 수시적이고 비정형적인 질의(조회)에 대하여 실시간 처리(Real-Time Processing)에 의한 응답이 가능함
- 계속적인 변화(Continuous Evolution) : 새로운 데이터의 삽입(Insertion), 삭제(Deletion), 갱신(Update)으로 항상 최신의 데이터를 유지함
- 동시 공유(동시 공용)(Concurrent Sharing) : 여러 사용자가 동시에 자기가 원하는 데이터를 이용할 수 있음
- 내용에 의한 참조(Content Reference) : 데이터베이스에 있는 데이터를 참조할 때 데이터 주소나 위치에 의해서가 아니라 사용자가 요구하는 데이터 내용으로 데이터를 찾음

핵심 004 기존의 파일 처리 방식에서의 문제점

09.5, 02.3, 01.3, 99.10

종속성으로 인한 문제점

- 종속성이란 응용 프로그램과 데이터 파일이 상호 의존적인 관계를 말한다.
- 데이터 파일이 보조기억장치에 저장되는 방법이나 저장된 데이터의 접근 방법을 변경할 때는 응용 프로그램도 같이 변경해야 한다.

중복성으로 인한 문제점

- 일관성 : 중복된 데이터 간에 내용이 일치하지 않는 상황이 발생하여 일관성이 없어짐
- 보안성 : 중복되어 있는 모든 데이터에 동등의 보안 수준을 유지하기가 어려움
- 경제성 : 저장 공간의 낭비와 동일한 데이터의 반복 작업으로 인한 비용의 증가
- 무결성 : 제어의 분산으로 인해 데이터의 정확성을 유지할 수 없음

핵심 005 DBMS의 정의 및 필수 기능

14.8, 13.6, 10.9, 10.5, 07.9, 06.9, 06.5, 05.5, 04.9, 04.5, 03.3, 02.9, 02.5, 02.3, 01.9, 01.6, 00.5, 00.3, 99.10

DBMS란 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해 주고, 데이터베이스를 관리해 주는 소프트웨어로 다음과 같은 3가지의 필수 기능이 있다.

- 정의(조직)(Definition) 기능 : 데이터의 형(Type)과 구조, 데이터가 DB에 저장될 때의 제약 조건 등을 명시하는 기능
- 조작(Manipulation) 기능 : 데이터 검색, 갱신, 삽입 삭제 등을 체계적으로 처리하기 위해 데이터 접근 수단 등을 정하는 기능
- 제어(Control) 기능
  - 데이터베이스를 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되어 데이터의 **무결성**이 유지되도록 제어해야 한다.
  - 정당한 사용자가 허가된 데이터만 접근할 수 있도록 **보안(Security)**을 **유지**하고 **권한(Authority)**을 **검사**할 수 있어야 한다.
  - 여러 사용자가 데이터베이스를 동시에 접근하여 데이터를 처리할 때 처리 결과가 항상 정확성을 유지하도록 **병행 제어**(Concurrency Control)를 할 수 있어야 한다.

핵심 006 DBMS의 장 · 단점

14.8, 12.5, 12.3, 11.3, 06.5, 05.9, 05.4, 05.3, 04.9, 02.9, 02.5, 00.10, 00.3

장점	단점
• 데이터의 중복을 피할 수 있음 • 저장된 자료를 공동으로 이용할 수 있음 • 데이터의 일관성을 유지할 수 있음 • 데이터의 무결성을 유지할 수 있음 • 보안을 유지할 수 있음 • 데이터를 표준화할 수 있음 • 데이터를 통합하여 관리할 수 있음 • 항상 최신의 데이터를 유지함 • 데이터의 실시간 처리가 가능함 • 데이터의 논리적 · 물리적 독립성이 보장됨	• 데이터베이스 전문가 부족 • 전산화 비용이 증가함 • 대용량 디스크로의 집중적인 Access로 과부하(Overhead)가 발생함 • 파일의 예비(Backup)와 회복(Recovery)이 어려움 • 시스템이 복잡함 • 파일 시스템에 비해 자료 처리 방법이 복잡함

논리적 독립성과 물리적 독립성

- **논리적 독립성** : 응용 프로그램과 데이터베이스를 독립시킴으로써, 데이터의 논리적 구조를 변경시키더라도 응용 프로그램은 변경되지 않음
- **물리적 독립성** : 응용 프로그램과 보조기억장치 같은 물리적 장치를 독립시킴으로써, 데이터베이스 시스템의 성능 향상을 위해 새로운 디스크를 도입하더라도 응용 프로그램에는 영향을 주지 않고 데이터의 물리적 구조만을 변경함

백업(Backup)

장비 고장 등의 비상사태에도 데이터베이스가 보존되도록 복사해 놓는 작업

핵심 007 스키마(Schema)의 정의

13.6, 10.5, 10.3, 08.5, 08.3, 06.9, 02.9, 00.10, 00.3, 99.10

- 데이터베이스의 구조와 제약 조건에 관한 전반적인 명세(Specification)를 기술(Description)한다.
- 데이터베이스를 구성하는 데이터 개체(Entity), 속성(Attribute), 관계(Relationship) 및 데이터 조작 시 데이터 값들이 갖는 제약 조건 등에 관해 전반적으로 정의한다.
- 스키마는 사용자의 관점에 따라 외부 스키마, 개념 스키마, 내부 스키마로 나누어진다.
- 스키마(Schema)는 데이터 사전에 저장되며, 다른 이름으로 메타데이터(Meta-data) 라고도 한다.

핵심 008 스키마(Schema)의 3계층

14.5, 11.8, 11.6, 11.3, 09.8, 09.3, 08.9, 08.3, 07.9, 07.3, 06.5, 06.3, 05.9, 05.3, 04.5, 03.5, 03.3, 02.9, 02.5, 01.9, 01.6, 01.3

외부 스키마(External Schema) = 서브 스키마 = 사용자 뷰(View)

- 사용자나 응용 프로그래머가 각 개인의 입장에서 필요로 하는 데이터베이스의 논리적 구조를 정의한다.
- 전체 데이터베이스의 한 논리적인 부분으로 볼 수 있으므로 서브 스키마(Subschema)라고도 한다.
- 하나의 데이터베이스 시스템에는 여러 개의 외부 스키마가 존재할 수 있으며, 하나의 외부 스키마를 여러 개의 응용 프로그램이나 사용자가 공용할 수 있다.
- 같은 데이터베이스에 대해서도 서로 다른 관점을 정의할 수 있도록 허용한다.
- 일반 사용자는 질의어(SQL)를 사용하여 DB를 사용한다.

개념 스키마(Conceptual Schema) = 전체적인 뷰(View)

- 데이터베이스의 전체적인 논리적 구조로서, 모든 응용 프로그램이나 사용자들이 필요로 하는 데이터를 종합한 조직 전체의 데이터베이스로 하나만 존재한다.
- 개념 스키마는 개체 간의 관계와 제약 조건을 나타내고 데이터베이스의 접근 권한, 보안 및 무결성 규칙에 관한 명세를 정의한다.

- 데이터베이스 파일에 저장되는 데이터의 형태를 나타낸 것으로 단순히 스키마(Schema)라고 하면 개념 스키마를 의미한다.
- 기관이나 조직체의 관점에서 데이터베이스를 정의한 것이다.
- 데이터베이스 관리자에 의해서 구성된다.

내부 스키마(Internal Schema)

- 물리적 저장장치의 입장에서 본 데이터베이스 구조로, 물리적인 저장장치와 밀접한 계층이다.
- 실제로 데이터베이스에 저장될 레코드의 물리적인 구조를 정의하고, 저장 데이터 항목의 표현 방법, 내부 레코드의 물리적 순서 등을 나타낸다.
- 시스템 프로그래머나 시스템 설계자가 보는 관점의 스키마이다.

핵심 009 데이터베이스 언어(Database Language)

13.8, 09.5, 05.4, 05.3, 03.8, 03.5, 02.5, 02.3, 01.6, 00.7, 00.5, 99.10

데이터 정의 언어(DDL; Data Definition Language)

- DB 구조, 데이터 형식, 접근 방식 등 DB를 구축하거나 수정할 목적으로 사용하는 언어이다.
- 번역한 결과가 데이터 사전(Data-Dictionary)이라는 특별한 파일에 여러 개의 테이블로 저장된다.
- 데이터 정의 언어의 기능
  - 외부 스키마 명세 정의
  - 데이터베이스 정의 및 수정
  - 스키마에 사용되는 제약 조건에 대한 명세 정의
  - 데이터의 물리적 순서 규정

데이터 조작 언어(DML; Data Manipulation Language) = 서브 언어

- 사용자로 하여금 데이터를 처리할 수 있게 하는 도구로서 사용자(응용프로그램)와 DBMS 간의 인터페이스를 제공한다.
- 응용 프로그램을 통하여 사용자가 DB의 데이터를 실질적으로 조작할 수 있도록 하기 위해 C, COBOL 등의 호스트 언어에 DB 기능을 추가시켜 만든 언어이다.
- 대표적인 데이터 조작어(DML)에는 질의어가 있으며, 질의어는 터미널에서 주로 이용하는 비절차적(Non procedural) 데이터 언어이다.

데이터 제어 언어(DCL; Data Control Language)

- 무결성, 보안 및 권한 제어, 회복 등을 하기 위한 언어이다.
- 데이터를 보호하고 데이터를 관리하는 목적으로 사용된다.
- 데이터 제어 언어의 기능
  - 불법적인 사용자로부터 데이터를 보호하기 위한 데이터 보안(Security)
  - 데이터의 정확성을 위한 무결성(Integrity) 유지
  - 시스템 장애에 대비한 데이터 회복과 병행수행 제어

핵심 010 기타 데이터베이스 사용자

14.3, 13.8, 09.5, 09.3, 08.9, 07.5, 06.9, 06.5, 05.9, 05.4, 04.9, 04.5, 03.8, 03.5, 02.9, 01.9, 01.6, 00.7

DBA(DataBase Administrator)

데이터베이스 시스템의 모든 관리와 운영에 대한 책임을 지고 있는 사람이나 그룹을 의미한다.

- 데이터베이스 구성 요소 결정
- 개념 스키마 및 내부 스키마 정의
- 데이터베이스의 저장 구조 및 접근 방법 정의
- 보안 및 데이터베이스의 접근 권한 부여 정책 수립
- 장애에 대비한 예비(Back Up) 조치와 회복(Recovery)에 대한 전략 수립
- 무결성을 위한 제약 조건의 지정
- 데이터 사전의 구성과 유지 관리
- DBMS의 선택, 보완, 평가에 대한 책임
- 사용자의 요구와 불평의 청취 및 해결
- 변화 요구에 대한 적응과 성능 향상에 대한 감시
- 시스템 감시 및 성능 분석
- 데이터 사용 추세, 이용 형태 및 각종 통계 등을 종합, 분석

응용 프로그래머

- 응용 프로그래머는 일반 호스트 언어로 프로그램을 작성할 때 데이터 조작어를 삽입해서 일반 사용자가 응용 프로그램을 사용할 수 있게, 인터페이스를 제공할 목적으로 데이터베이스를 접근하는 사람들이다.
- 응용 프로그래머는 C, COBOL, PASCAL 등의 호스트 언어와 DBMS가 지원하는 데이터 조작어에 능숙한 컴퓨터 전문가이다.

일반 사용자

일반 사용자는 보통 터미널을 이용하여 데이터베이스에 있는 자원을 활용할 목적으로 질의어나 응용 프로그램을 사용하여 데이터베이스에 접근하는 사람들이다.

핵심 011 데이터 모델의 정의

05.3, 03.8, 02.5, 00.7, 99.10

- 현실 세계의 정보들을 컴퓨터에 표현하기 위해서 단순화, 추상화하여 체계적으로 표현한 개념적 모형이다.
- 데이터, 데이터의 관계, 데이터의 의미 및 일관성, 제약조건 등을 기술하기 위한 개념적 도구들의 모임이다.
- 현실 세계를 데이터베이스에 표현하는 중간 과정, 즉 데이터베이스 설계 과정에서 데이터의 구조를 논리적으로 표현하기 위해 사용되는 도구이다.

핵심 012 데이터 모델의 종류

04.9, 01.6, 00.5, 99.10

개념적 데이터 모델

- 현실 세계에 대한 인간의 이해를 돕기 위하여 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정이다.
- 속성들로 기술된 개체 타입과 이 개체 타입들 간의 관계를 이용하여 현실 세계를 표현하는 방법이다.
- 현실 세계에 존재하는 개체를 인간이 이해할 수 있는 정보 구조로 표현하기 때문에 정보 모델이라고도 한다.
- 대표적으로 개체-관계(E-R) 모델이 있다.

논리적 데이터 모델

- 개념적 모델링 과정에서 얻은 개념적 구조를 컴퓨터가 이해하고 처리할 수 있는 컴퓨터 세계의 환경에 맞도록 변환하는 과정이다.
- 필드로 기술된 데이터 타입과 이 데이터 타입들 간의 관계를 이용하여 현실 세계를 표현하는 방법이다.
- 단순히 데이터 모델이라고 하면 논리적 데이터 모델을 의미한다.
- 논리적 데이터 모델은 데이터 간의 관계를 어떻게 표현하느냐에 따라 관계형 모델, 계층형 모델, 네트워크형 모델, 객체지향 모델로 구분한다.

핵심 013 데이터 모델에 표시할 사항

14.8, 13.6, 12.5, 11.3, 08.5, 08.3, 06.9, 06.3, 05.9, 04.3

- 구조(Structure) : 논리적으로 표현된 개체들 간의 관계를 표시함
- 연산(Operation) : 데이터베이스에 저장된 실제 데이터를 처리하는 방법을 표시하는 것으로서, 데이터베이스를 조작하는 기본 도구임
- 제약 조건(Constraint) : 데이터베이스에 저장될 수 있는 실제 데이터의 논리적인 제약 조건을 표시함

핵심 014 데이터 모델의 구성 요소

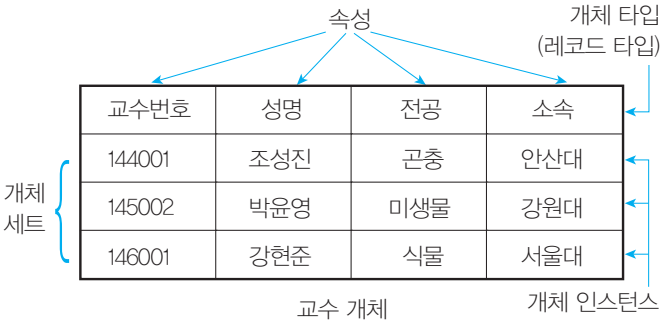
14.5, 14.3, 12.8, 12.5, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.3, 07.9, 05.5, 05.3, 04.9, 00.10, 00.3

개체(Entity)

- 데이터베이스에 표현하려는 것으로, 사람이 생각하는 개념이나 정보 단위 같은 현실 세계의 대상체이다.
- 유형, 무형의 정보로서 서로 연관된 몇 개의 속성으로 구성된다.
- 파일 시스템의 레코드에 대응하는 것으로, 어떤 정보를 제공하는 역할을 수행한다.
- 독립적으로 존재하거나 그 자체로서도 구별이 가능하다.

속성(Attribute)

- 데이터의 가장 작은 논리적 단위로서 파일 구조의 데이터 항목 또는 데이터 필드에 해당된다.
- 개체를 구성하는 항목이다.

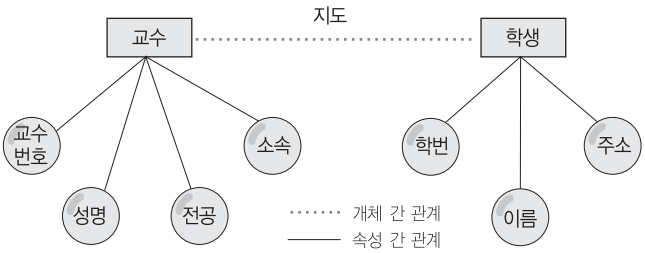


- 교수 개체의 구성 요소
  - 속성 : 개체가 가지고 있는 특성, 교수번호, 성명, 전공, 소속
  - 개체 타입 : 속성으로만 기술된 개체의 정의

- 개체 인스턴스 : 개체를 구성하고 있는 각 속성들이 값을 가져 하나의 개체를 나타내는 것으로 개체 어커런스(Occurence)라고도 함
- 개체 세트 : 개체 인스턴스의 집합

관계(Relationship)

- 개체 간의 관계 또는 속성 간의 관계



- 위 그림의 관계는 교수가 학생을 지도하는 관계이다.

기호	기호 이름	의미
	사각형	개체(Entity) 타입
	마름모	관계(Relationship) 타입
	타원	속성(Attribute)
	밑줄 타원	기본키 속성
	복수 타원	복합 속성 예 성명은 성과 이름으로 구성
n m	관계	1:1, 1:N, N:M 등의 개체 관계에 대해 선 위에 대응수 기술
	선, 링크	개체 타입과 속성 연결

핵심 015 개체-관계(Entity-Relationship) 모델

11.8, 09.8, 09.5, 07.9, 06.9, 05.4, 04.3, 03.8, 02.3

- 개념적 데이터 모델의 가장 대표적인 것으로, 1976년 Peter Chen에 의해 제안되었다.
- 개체 타입(Entity Type)과 이들 간의 관계 타입(Relationship Type)을 이용해 현실 세계를 개념적으로 표현한다.
- 데이터를 개체(Entity), 관계(Relationship), 속성(Attribute)으로 묘사한다.
- 특정 DBMS를 고려한 것은 아니다.
- E-R 다이어그램으로 표현하며, 1:1, 1:N, N:M 등의 관계 유형을 제한 없이 나타낼 수 있다.

핵심 016 E-R 다이어그램

02.5, 02.3, 02.5, 02.3
14.5, 13.8, 12.8, 12.3, 08.9, 08.5, 08.3, 07.9, 07.5, 07.3, 06.9, 06.5, 06.3, 05.5, 05.3, 04.9, 04.5, 04.3, 03.5, 02.9

- E-R 모델의 기본적인 아이디어를 시각적으로 표현하기 위한 도구이다.
- 개체 간의 관계는 물론 시스템 내의 역할을 하는 모든 개체들, 즉 조직, 부서, 사용자, 프로그램, 데이터를 모두 표시한다.

핵심 017 관계형 데이터 모델

05.9, 99.10

- 계층 모델과 망 모델의 복잡한 구조를 단순화시킨 모델이다.
- 2차원적인 표(Table)를 이용해서 데이터 상호 관계를 정의하는 DB 구조를 말하는데, 파일 구조처럼 구성한 테이블들을 하나의 DB로 묶어서 테이블 내에 있는 속성들 간의 관계(Relationship)를 설정하거나 테이블 간의 관계를 설정하여 이용한다.
- 데이터 간의 관계를 기본키(Primary Key)와 이를 참조하는 외래키(Foreign Key)로 표현한다.
- 대표적인 DBMS : Oracle, MS-SQL, Informix 등
- 1:1, 1:N, M:N 관계를 자유롭게 표현할 수 있다.
- 장점 : 간결하고, 보기 편리하며, 다른 데이터베이스로의 변환이 용이함
- 단점 : 성능이 다소 떨어짐

핵심

14.3, 07.9, 03.8, 02.3, 01.6, 99.8

018 계층형 데이터 모델

- 데이터의 논리적 구조도가 트리 형태이며, 개체가 트리를 구성하는 노드 역할을 한다.
- 개체 집합에 대한 속성 관계를 표시하기 위해 개체를 노드로 표현하고 개체 집합들 사이의 관계를 링크로 연결한다.
- 개체 간의 관계를 부모와 자식 간의 관계로 표현한다.
- 개체 타입 간에는 상위와 하위 관계가 존재하며, 일 대 다(1:N) 대응 관계만 존재한다.
- 레코드 삭제 시 연쇄 삭제(Triggered Delete)가 된다.
- 개체 타입들 간에는 사이클(Cycle)이 허용되지 않는다.
- 계층형 모델에서는 개체(Entity)를 세그먼트(Segment)라 부른다.
- 대표적인 DBMS : IMS 등

핵심

14.5, 14.3, 13.8, 13.6, 11.8, 11.6, 11.3, 10.3, 09.8, 08.5, 05.9, 03.3, 00.3, 99.6

019 망(그래프, 네트워크)형 데이터 모델

- CODASYL이 제안한 것으로, CODASYL DBTG 모델이라고도 한다.
- 그래프를 이용해서 데이터 논리 구조를 표현한 데이터 모델이다.
- 상위와 하위 레코드 사이에서 다 대 다(N:M) 대응 관계를 만족하는 구조이다.
- 상위의 레코드를 Owner, 하위의 레코드를 Member라 하여 Owner-Member 관계라고도 한다.
- 레코드 타입 간의 관계는 1:1, 1:N, N:M이 될 수 있다.
- 대표적인 DBMS : DBTG, EDBS, TOTAL 등

06.3, 05.4, 05.3, 03.5, 03.3, 00.10, 00.7, 00.5, 99.8

핵심

14.8, 14.3, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.3, 07.5, 07.3, 06.9,

020 데이터베이스 설계

개념적 설계(정보 모델링, 개념화)

- 정보의 구조를 얻기 위하여 현실 세계의 무한성과 계속성을 이해하고, 다른 사람과 통신하기 위하여 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정이다.
- 개념 스키마 모델링과 트랜잭션 모델링을 병행하여 수행한다.
- 요구 분석 단계에서 나온 결과(요구 조건 명세)를 DBMS에 독립적인 E-R 다이어그램(개체 관계도)으로 작성한다.
- DBMS에 독립적인 개념 스키마를 설계한다.

논리적 설계(데이터 모델링)

- 현실 세계에서 발생하는 자료를 컴퓨터가 처리할 수 있는 물리적 저장장치에 저장할 수 있도록 변환하기 위해 특정 DBMS가 지원하는 논리적 자료 구조로 변환시키는 과정이다.
- 개념 세계의 데이터를 필드로 기술된 데이터 타입과 이 데이터 타입들 간의 관계로 표현되는 논리적 구조의 데이터로 모델화 한다.
- 개념적 설계가 개념 스키마를 설계하는 단계라면 논리적 설계에서는 개념 스키마를 평가 및 정제하고 DBMS에 따라 서로 다른 논리적 스키마를 설계하는 단계이다.
- 트랜잭션의 인터페이스를 설계한다.
- 관계형 데이터베이스라면 테이블을 설계하는 단계이다.

물리적 설계(데이터 구조화)

- 논리적 설계 단계에서 논리적 구조로 표현된 데이터를 디스크 등의 물리적 저장장치에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정이다.
- 물리적 설계 단계에서는 다양한 데이터베이스 응용에 대해 처리 성능을 얻기 위해 데이터베이스 파일의 저장 구조 및 액세스 경로를 결정한다.
- 저장 레코드의 형식, 순서, 접근 경로와 같은 정보를 사용하여 데이터가 컴퓨터에 저장되는 방법을 묘사한다.
- 물리적 설계 단계에 꼭 포함되어야 할 것은 저장 레코드의 양식 설계, 레코드 집중(Record Clustering)의 분석 및 설계, 접근 경로 설계 등이 있다.
- 물리적 데이터베이스 구조의 기본적인 데이터 단위는 저장 레코드(Stored Record)이다.
- 물리적 데이터베이스 구조는 여러 가지 타입의 저장 레코드 집합이라는 면에서 단순한 파일과 다르다.
- 물리적 데이터베이스 구조는 데이터베이스 시스템의 성능에 중대한 영향을 미친다.
- 물리적 설계 옵션 선택 시 고려 사항
  - 반응 시간(Response Time) : 트랜잭션 수행을 요구한 시점부터 처리 결과를 얻을 때까지의 경과(응답) 시간

- 공간 활용도(Space Utilization) : 데이터베이스 파일과 액세스 경로 구조에 의해 사용되는 저장 공간의 양
- 트랜잭션 처리량(Transaction Throughput) : 단위 시간 동안 데이터베이스 시스템에 의해 처리될 수 있는 트랜잭션의 평균 개수

핵심 021 데이터베이스 설계 순서

13.8, 13.6, 12.5, 10.3, 08.5, 08.3, 07.5, 07.3, 06.9, 06.5, 05.9, 05.4, 04.5, 03.8, 03.5, 03.3, 02.9, 01.9

요구 분석 → 요구 조건 명세서 작성

↓

개념적 설계 → 개념 스키마, 트랜잭션 모델링, E-R 모델

↓

논리적 설계 → 목표 DBMS에 맞는 스키마 설계, 트랜잭션 인터페이스 설계

↓

물리적 설계 → 목표 DBMS에 맞는 물리적 구조의 데이터로 변환

↓

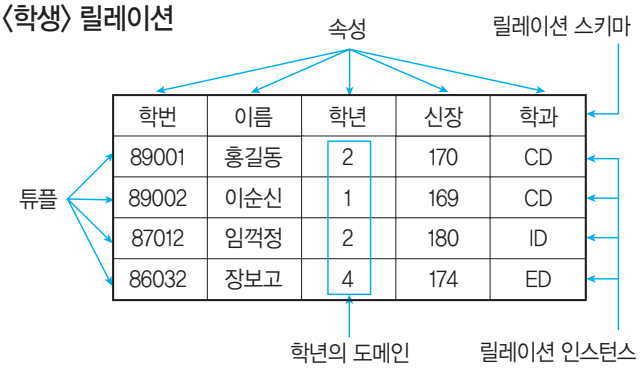
구현 → 특정 DBMS의 DDL로 데이터베이스 생성, 트랜잭션 작성

핵심 022 관계 데이터베이스의 Relation 구조

02.9, 02.3, 01.9, 01.6, 00.10
14.5, 13.6, 12.8, 12.5, 12.3, 10.9, 09.5, 09.3, 07.9, 07.3, 06.9, 06.5, 06.3, 05.9, 05.5, 05.3, 04.9, 04.5, 04.3, 03.5, 03.3,

릴레이션은 데이터들을 표(Table)의 형태로 표현한 것으로, 구조를 나타내는 릴레이션 스키마와 실제 값들인 릴레이션 인스턴스로 구성된다.

〈학생〉 릴레이션



튜플(Tuple)

- 릴레이션을 구성하는 각각의 행
- 속성의 모임으로 구성된다.
- 파일 구조에서 레코드와 같은 의미이다.
- 튜플의 수 = 카디널리티(Cardinality) = 기수 = 대응수

속성(Attribute, 애트리뷰트)

- 데이터베이스를 구성하는 가장 작은 논리적 단위
- 파일 구조상의 데이터 항목 또는 데이터 필드(열)에 해당된다.
- 개체의 특성을 기술한다.
- 속성의 수 = 디그리(Degree) = 차수

도메인(Domain)

- 하나의 애트리뷰트가 취할 수 있는 같은 타입의 원자(Atomic)값들의 집합
- 실제 애트리뷰트 값이 나타날 때 그 값의 합법 여부를 시스템이 검사하는 데에도 이용된다.

릴레이션 스키마

- 한 릴레이션의 논리적 구조를 기술한 것이다.
- 릴레이션 스키마는 정적인 성질을 가지며, 릴레이션 인스턴스는 동적인 성질을 가진다.

릴레이션 인스턴스(Relation Instance)

데이터 개체를 구성하고 있는 속성들에 데이터 타입이 정의되어 구체적인 데이터 값을 갖고 있는 것을 말한다.

예 〈학생〉 릴레이션의 인스턴스

핵심 023 릴레이션의 특징

05.5, 05.3, 04.9, 04.5, 99.10
14.8, 14.3, 13.8, 13.3, 12.8, 12.5, 11.6, 10.9, 10.5, 10.3, 09.5, 09.3, 08.9, 08.5, 08.3, 07.9, 07.5, 07.3, 06.5, 06.3, 05.9,

〈학생〉 릴레이션

학번	이름	학년	신장	학과
89001	홍길동	2	170	CD
89002	이순신	1	169	CD
87012	임꺽정	2	180	ID
86032	장보고	4	174	ED

- 한 릴레이션에 포함된 튜플들은 모두 상이하다.

  예 〈학생〉 릴레이션을 구성하는 홍길동 레코드는 홍길동에 대한 학적사항을 나타내는 것으로 〈학생〉 릴레이션 내에서는 유일하다.

- 한 릴레이션에 포함된 튜플 사이에는 순서가 없다.

  예 〈학생〉 릴레이션에서 홍길동 레코드와 임꺽정 레코드의 위치가 바뀌어도 상관없다.

- 튜플들의 삽입, 삭제 등의 작업으로 인해 릴레이션은 시간에 따라 변한다.
  - 예 〈학생〉 릴레이션에 새로운 학생의 레코드를 삽입하거나, 기존 학생에 대한 레코드를 삭제함으로써 테이블은 내용 면에서나 크기 면에서 변하게 된다.
- 릴레이션 스키마를 구성하는 속성들 간의 순서는 중요하지 않다.
  - 예 학번, 이름 등의 속성을 나열하는 순서가 이름, 학번 순으로 바뀌어도 데이터 처리에는 아무런 영향을 미치지 않는다.
- 속성의 유일한 식별을 위해 속성의 명칭은 유일해야 하지만, 속성을 구성하는 값은 동일한 값이 있을 수 있다.
  - 예 각 학생의 학년을 기술하는 속성인 '학년'은 다른 속성명들과 구분되어 유일해야 하지만 '학년' 속성에는 2, 1, 2, 4 등이 입력된 것처럼 동일한 값이 있을 수 있다.
- 릴레이션을 구성하는 튜플을 유일하게 식별하기 위해 속성들의 부분집합을 키(Key)로 설정한다.
  - 예 〈학생〉 릴레이션에서는 '학번'이나 '이름'이 튜플들을 구분하는 유일한 값인 키가 될 수 있다.
- 속성은 더 이상 쪼갤 수 없는 원자값만을 저장한다.
  - 예 '학년'에 저장된 1, 2, 4 등은 더 이상 세분화할 수 없다.

01.3, 99.10, 99.4
14.5, 14.3, 13.3, 12.8, 12.3, 11.8, 11.3, 10.5, 10.3, 09.8, 09.5, 08.5 08.3, 07.9, 06.5, 05.9, 05.4, 04.9, 03.3, 02.5, 02.3,

핵심 024 키(Key)의 개념 및 종류

키(Key)는 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 다른 튜플들과 구별할 수 있는 유일한 기준이 되는 애트리뷰트(속성)이다.

〈학생〉 릴레이션

학번	주민번호	성명	성별
1001	810429-1231457	김형석	남
1002	800504-1546781	김현천	남
1003	811215-2547842	류기선	여
1004	801245-2201248	홍영선	여

〈수강〉 릴레이션

학번	과목명
1001	영어
1001	전산
1002	영어
1003	수학
1004	영어
1004	전산

후보키 (Candidate Key)	• 릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별하기 위해 사용하는 속성들의 부분집합, 즉 기본키로 사용할 수 있는 속성들을 말함 • 릴레이션에 있는 모든 튜플에 대해서 유일성과 최소성을 만족시켜야 함 예 〈학생〉 릴레이션에서 '학번'이나 '주민번호'는 다른 레코드를 유일하게 구별할 수 있는 기본키로 사용할 수 있으므로 후보키이다.
기본키 (Primary Key)	• 후보키 중에서 선택한 주키(Main Key) • 한 릴레이션에서 특정 튜플을 유일하게 구별할 수 있는 속성 • Null 값을 가질 수 없음 • 기본키로 정의된 속성에는 동일한 값이 중복되어 저장될 수 없음 예 〈학생〉 릴레이션에서는 '학번'이나 '주민번호'가 기본키가 될 수 있고, 〈수강〉 릴레이션에서는 '학번'+'과목명'으로 조합해야 기본키가 만들어진다. 예 '학번'이 〈학생〉 릴레이션의 기본키로 정의되면 이미 입력된 '1001'은 다른 튜플의 '학번' 속성의 값으로 입력할 수 없다.
대체키 (Alternate Key)	• 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키들을 말하며, 보조키라고도 함 예 〈학생〉 릴레이션에서 '학번'을 기본키로 정의하면 '주민번호'는 대체키가 된다.
슈퍼키 (Super Key)	• 슈퍼키는 한 릴레이션 내에 있는 속성들의 집합으로 구성된 키로서 릴레이션을 구성하는 모든 튜플 중 슈퍼키로 구성된 속성의 집합과 동일한 값은 나타나지 않음 • 릴레이션을 구성하는 모든 튜플에 대해 유일성은 만족시키지만, 최소성은 만족시키지 못함 예 〈학생〉 릴레이션에서는 '학번', '주민번호', '학번'+'주민번호', '주민번호'+'성명', '학번'+'주민번호'+'성명' 등으로 슈퍼키를 구성할 수 있다.
외래키 (Foreign Key)	• 관계(Relationship)를 맺고 있는 릴레이션 R1, R2에서 릴레이션 R1이 참조하고 있는 릴레이션 R2의 기본키와 같은 R1 릴레이션의 속성 • 관계형 데이터 모델에서 외래키는 참조되는 릴레이션의 기본키와 대응되어 릴레이션 간에 참조 관계를 표현하는데 중요한 도구로 사용됨 • 외래키로 지정되면 참조 테이블의 기본키에 없는 값은 입력할 수 없음 예 〈수강〉 릴레이션이 〈학생〉 릴레이션을 참조하고 있으므로 〈학생〉 릴레이션의 '학번'은 기본키이고, 〈수강〉 릴레이션의 '학번'은 외래키이다. 예 〈수강〉 릴레이션의 '학번'에는 〈학생〉 릴레이션의 '학번'에 없는 값은 입력할 수 없다.

잠깐만요

- **널 값(NULL Value)** : 데이터베이스에서 아직 알려지지 않았거나 모르는 값으로서 "해당 없음" 등의 이유로 정보 부재를 나타내기 위해 사용하는, 이론적으로 아무것도 없는 특수한 데이터
- **최소성과 유일성** : '학번'+'주민번호'를 사용하여 슈퍼키를 만들면 다른 튜플들과 구분할 수 있는 유일성은 만족하지만, '학번'이나 '주민번호' 하나만 가지고도 다른 튜플들을 구분할 수 있으므로 최소성은 만족시키지 못함

00.10, 00.7, 00.5, 00.3, 99.10, 99.8, 99.6
13.6, 13.3, 12.8, 11.8, 10.3, 09.8, 09.3, 08.5, 08.3, 07.9, 07.5, 05.4, 05.3, 04.5, 03.8, 03.3, 02.9, 02.3, 01.9, 01.6,

핵심 025 무결성(Integrity)

- **개체 무결성** : 릴레이션에서 기본키를 구성하는 속성은 널(NULL) 값이나 중복값을 가질 수 없음
  - 예 〈학생〉 릴레이션에서 '학번'이 기본키로 정의 되면 튜플을 추가할 때 '주민번호'나 '성명' 필드에는 값을 입력하지 않아도 되지만 '학번' 속성에는 반드시 값을 입력해야 한다. 또한 '학번' 속성에는 이미 한 번 입력한 속성 값을 중복하여 입력할 수 없다.
- **참조 무결성** : 외래키 값은 NULL이거나 참조 릴레이션의 기본키 값과 동일해야 함, 즉 릴레이션은 참조할 수 없는 외래키값을 가질 수 없음
  - 예 〈수강〉 릴레이션의 '학번' 속성에는 〈학생〉 릴레이션의 '학번' 속성에 없는 값은 입력할 수 없다.
- **도메인 무결성** : 특정 속성의 값이, 그 속성이 정의된 도메인에 속한 값이어야 한다는 규정
  - 예 성별 속성의 도메인은 '남' 과 '여'로, 그 외의 값은 입력할 수 없다.

12.8, 12.5, 11.8, 09.8, 05.3

핵심 026 관계대수

- 관계형 데이터베이스에서 원하는 정보와 그 정보를 어떻게 유도하는가를 기술하는 절차적인 언어이다.
- 릴레이션을 처리하기 위해 연산자와 연산규칙을 제공하는 언어로 피연산자가 릴레이션이고, 결과도 릴레이션이다.
- 질의에 대한 해를 구하기 위해 수행해야 할 연산의 순서를 명시한다.
- **순수 관계 연산자** : Select, Project, Join, Division
- **일반 집합 연산자** : UNION(합집합), INTERSECTION(교집합), DIFFERENCE (차집합), Cartesian Product(교차곱)

14.8, 14.5, 12.3, 10.9, 10.3, 09.5, 08.9, 08.5, 03.8, 03.5, 99.8

핵심 027 순수 관계 연산자

관계 데이터베이스에 적용할 수 있도록 특별히 개발한 관계 연산자

연산자	특징
Select	• 릴레이션에 존재하는 튜플 중에서 선택 조건을 만족하는 튜플의 부분집합을 구하여 새로운 릴레이션을 만듦 • 릴레이션의 행(가로)에 해당하는 튜플을 구하는 것이므로 수평 연산이라고도 함 • 연산자의 기호는 그리스 문자 시그마(σ)를 사용함
Project	• 주어진 릴레이션에서 속성 List에 제시된 Attribute만을 추출하는 연산 • 릴레이션의 열(세로)에 해당하는 Attribute를 추출하는 것이므로 수직 연산자라고도 함 • 연산자의 기호는 그리스 문자 파이(π)를 사용함
Join	• 공통 속성을 중심으로 2개의 릴레이션을 하나로 합쳐서 새로운 릴레이션을 만드는 연산 • 연산자의 기호는 ⋈를 사용함 • 조인 조건이 '='일 때 동일한 속성이 두 번 나타나게 되는데, 이 중 중복된 속성을 제거하여 같은 속성을 한 번만 표기하는 방법을 자연(NATURAL) 조인이라고 함
Division	X ⊃ Y인 2개의 릴레이션에서 R(X)와 S(Y)가 있을 때, R의 속성이 S의 속성값을 모두 가진 튜플에서 S가 가진 속성을 제외한 속성만을 구하는 연산

14.5, 14.3, 12.8, 12.5, 11.6, 10.9, 10.5, 09.5, 09.3, 07.9, 06.3, 04.3, 00.5

핵심 028 관계해석

- 코드(E. F. Codd)가 수학의 Predicate Calculus(술어해석)에 기반을 두고 관계 데이터베이스를 위해 제안했다.
- 관계해석은 원하는 정보가 무엇이라는 것만 정의하는 비절차적 특성을 지닌다.
- 원하는 정보를 정의할 때는 계산 수식을 사용한다.
- 튜플 관계해석과 도메인 관계해석이 있다.
- 기본적으로 관계해석과 관계대수는 관계 데이터베이스를 처리하는 기능과 능력 면에서 동등하다.
- 질의어로 표현한다.

핵심 14.5, 13.3, 10.5, 09.5, 05.4, 04.3, 00.3

029 정규화(Normalization)

정규화의 개요

- 함수적 종속성 등의 종속성 이론을 이용하여 잘못 설계된 관계형 스키마를 더 작은 속성의 세트로 쪼개어 바람직한 스키마로 만들어 가는 과정이다.
- 정규형에는 제1정규형, 제2정규형, 제3정규형, BCNF형, 제4정규형, 제5정규형이 있으며, 차수가 높아질수록 만족시켜야 할 제약 조건이 늘어난다.
- 정규화는 데이터베이스의 개념적 설계 단계와 논리적 설계 단계에서 수행한다.
- 정규화는 논리적 처리 및 품질에 큰 영향을 미친다.

정규화의 목적

- 데이터 구조의 안정성을 최대화한다.
- 어떠한 릴레이션이라도 데이터베이스 내에서 표현 가능하게 만든다.
- 효과적인 검색 알고리즘을 생성할 수 있다.
- 중복을 배제하여 삽입, 삭제, 갱신 이상의 발생을 방지한다.
- 데이터 삽입 시 릴레이션을 재구성할 필요성을 줄인다.

정규화의 원칙

- 정보의 무손실 표현, 즉 하나의 스키마를 다른 스키마로 변환할 때 정보의 손실이 있어서는 안 된다.
- 분리의 원칙, 즉 하나의 독립된 관계성은 하나의 독립된 릴레이션으로 분리시켜 표현해야 한다.
- 데이터의 중복성이 감소되어야 한다.

핵심 11.6, 11.3, 07.3, 05.4, 02.5, 02.3, 01.3, 00.5, 99.8, 99.4

030 Anomaly(이상)의 개념 및 종류

- 이상(Anomaly)의 개념 : 정규화(Normalization)를 거치지 않은 데이터베이스 내에 데이터들이 불필요하게 중복되어 릴레이션 조작 시에 발생하는 예기치 못한 곤란한 현상
- 이상의 종류

삽입 이상 (Insertion Anomaly)	릴레이션에 데이터를 삽입할 때 의도와는 관계없이 원하지 않은 값들도 함께 삽입되는 현상
삭제 이상 (Deletion Anomaly)	릴레이션에서 한 튜플을 삭제할 때 의도와는 관계없는 값들도 함께 삭제되는 연쇄 삭제 현상
갱신 이상 (Update Anomaly)	릴레이션에서 튜플에 있는 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상

핵심 03.5, 01.6, 01.3, 00.10 14.8, 14.5, 13.8, 13.6, 13.3, 12.8, 12.3, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 07.9, 07.5, 06.9, 06.5, 05.4, 04.3,

031 정규화 과정

비정규 릴레이션
↓ 도메인이 원자값
1NF
↓ 부분적 함수 종속 제거
2NF
↓ 이행적 함수 종속 제거
3NF
↓ 결정자이면서 후보키가 아닌 것 제거
BCNF
↓ 다치 종속 제거
4NF
↓ 조인 종속성 이용
5NF

정규화 단계 암기 요령

정규화라는 출소자가 말했다.

두부이겨다줘≒도부이결다조

도메인이 원자값

부분적 함수 종속 제거

이행적 함수 종속 제거

결정자이면서 후보키가 아닌 것 제거

다치 종속 제거

조인 종속성 이용

함수적 종속 관계

〈수강〉 릴레이션이 (학번, 이름, 과목명)으로 되어 있을 때, '학번'이 결정되면 '과목명'에 상관없이 '학번'에는 항상 같은 이름이 대응된다. '학번'에 따라 '이름'이 결정될 때 '이름'을 '학번'에 함수 종속적이라고 하며 '학번 → 이름'과 같이 쓴다.

완전 함수적 종속 관계

속성 A가 다른 속성들의 집합 B 전체에 대해서 함수적 종속 관계를 갖지만 집합 B의 진부분집합에 대해서는 종속 관계를 갖지 않으면, 속성 A는 집합 B의 속성들에 대해 완전 함수적 종속 관계에 있다고 한다.

예 〈수강〉 릴레이션이 (학번, 과목명, 성적)으로 되어 있을 때, '성적'은 (학번, 과목명)에는 함수 종속이지만, '학번'이나 '과목명'에는 함수 종속이 아니다.

이행적 종속 관계

A → B이고 B → C일 때 A → C를 만족하는 관계

00.10, 99.4

핵심 032 SQL의 분류

14.8, 14.5, 14.3, 13.6, 13.3, 12.8, 11.6, 11.3, 10.9, 10.3, 09.8, 08.9, 08.5, 08.3, 07.9, 07.5, 07.3, 06.5, 05.5, 04.3, 02.5,

DDL(데이터 정의어)

- SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의하거나 변경 또는 삭제할 때 사용하는 언어이다.
- 데이터베이스 관리자나 데이터베이스 설계자가 사용한다.
- 데이터 정의어(DDL)의 3가지 유형

명령어	기능
CREATE	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의함
ALTER	TABLE에 대한 정의를 변경하는 데 사용함
DROP	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 삭제함

DML(데이터 조작어)

- 데이터베이스 사용자가 응용 프로그램이나 질의어를 통하여 저장된 데이터를 실질적으로 처리하는 데 사용하는 언어이다.
- 데이터베이스 사용자와 데이터베이스 관리 시스템 간의 인터페이스를 제공한다.
- 데이터 조작어(DML)의 4가지 유형

명령어	기능
SELECT	테이블에서 조건에 맞는 튜플을 검색함
INSERT	테이블에 새로운 튜플을 삽입함
DELETE	테이블에서 조건에 맞는 튜플을 삭제함
UPDATE	테이블에서 조건에 맞는 튜플의 내용을 변경함

DCL(데이터 제어어)

- 데이터의 보안, 무결성, 데이터 회복, 병행수행 제어 등을 정의하는 데 사용하는 언어이다.
- 데이터베이스 관리자가 데이터 관리를 목적으로 사용한다.
- 데이터 제어어(DCL)의 종류

명령어	기능
COMMIT	명령에 의해 수행된 결과를 실제 물리적 디스크로 저장하고, 데이터베이스 조작 작업이 정상적으로 완료 되었음을 관리자에게 알려줌
ROLLBACK	데이터베이스 조작 작업이 비정상적으로 종료되었을 때 원래의 상태로 복구함
GRANT	데이터베이스 사용자에게 사용 권한을 부여함
REVOKE	데이터베이스 사용자의 사용 권한을 취소함

핵심 033 Select문

14.3, 13.8, 12.5, 12.3, 10.5, 09.5, 09.3, 08.9, 06.9, 05.9, 05.5, 05.4, 04.9, 04.3, 03.5, 02.5, 00.7, 99.10

테이블을 구성하는 튜플(행)들 중에서 전체 또는 조건을 만족하는 튜플(행)을 검색하여 주기억장치 상에 임시 테이블로 구성시키는 명령문이다.

```
SELECT Predicate [테이블명.]속성명1, [테이블명.]속성명2,…
FROM 테이블명1, 테이블명2,…
[WHERE 조건]
[GROUP BY 속성명1, 속성명2,…]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]];
```

1. SELECT절

- Predicate : 불러올 튜플 수를 제한할 명령어를 기술함
  - ALL : 모든 튜플을 검색할 때 지정하는 것으로, 주로 생략함
  - DISTINCT : 중복된 튜플이 있으면 그 중 첫 번째 한 개만 검색함
  - DISTINCTROW : 중복된 튜플을 검색하지만 선택된 속성의 값이 아닌, 튜플 전체를 대상으로 함
- 속성명 : 검색하여 불러올 속성(열) 및 수식들을 지정함
  - 기본 테이블을 구성하는 모든 속성을 지정할 때는 '*' 를 기술한다.
  - 두 개 이상의 테이블을 대상으로 검색할 때는 반드시 테이블명.속성명으로 표현해야 한다.

2. **FROM절** : 질의에 의해 검색될 데이터들을 포함하는 테이블명을 기술함
3. **WHERE절** : 검색할 조건 기술
4. **GROUP BY절**
   - 특정 속성을 기준으로 그룹화하여 검색할 때 그룹화할 속성을 지정한다.
   - 일반적으로 GROUP BY절은 그룹 함수와 함께 사용된다.
   - **그룹 함수의 종류**
     - COUNT(속성명) : 그룹별 튜플 수를 구하는 함수
     - MAX(속성명) : 그룹별 최대값을 구하는 함수
     - MIN(속성명) : 그룹별 최소값을 구하는 함수
     - SUM(속성명) : 그룹별 합계를 구하는 함수
     - AVG(속성명) : 그룹별 평균을 구하는 함수
5. **HAVING절** : GROUP BY와 함께 사용되며, 그룹에 대한 조건을 지정함
6. **ORDER BY절** : 특정 속성을 기준으로 정렬하여 검색할 때 사용함
   - **속성명** : 정렬의 기준이 되는 속성명을 기술함
   - [ASC|DESC] : 정렬 방식으로서 'ASC'는 오름차순, 'DESC'는 내림차순임, 생략하면 오름차순으로 지정됨

핵심 034 삽입, 삭제, 갱신문

13.8, 11.6, 07.5, 07.3, 06.3, 05.9, 05.5, 05.4, 05.3, 03.5, 03.3, 02.3, 00.3

삽입문(INSERT INTO ~)

- 기본 테이블에 새로운 튜플을 삽입할 때 사용한다.

```
INSERT
INTO 테이블명(속성명1, 속성명2,… )
VALUES (데이터1, 데이터2,… );
```

- 대응하는 속성과 데이터는 개수와 데이터 형식이 일치해야 한다.
- 기본 테이블의 모든 속성을 사용할 때는 속성명을 생략할 수 있다.
- SELECT문을 사용하여 다른 테이블의 검색 결과를 삽입할 수 있다.

삭제문(DELETE FROM ~)

- 기본 테이블에 있는 튜플들 중에서 특정 튜플을 삭제할 때 사용한다.

```
DELETE
FROM 테이블명
WHERE 조건;
```

- 모든 레코드를 삭제할 때는 WHERE절을 생략한다.
- 모든 레코드를 삭제하더라도 테이블 구조는 남아 있기 때문에 디스크에서 테이블을 완전히 제거하는 DROP과는 다르다.

갱신문 (UPDATE ~ SET ~)

- 기본 테이블에 있는 튜플들 중에서 특정 튜플의 내용을 변경할 때 사용한다.

```
UPDATE 테이블명
SET 속성명 = 데이터[, 속성명=데이터]
WHERE 조건;
```

핵심 035 내장 SQL(Embedded SQL)

06.3, 05.5, 04.5, 04.3, 03.8, 03.3, 01.9, 01.6, 01.3, 00.5

- 응용 프로그램이 실행될 때 함께 실행되도록 호스트 프로그램 언어로 만든 프로그램에 삽입된 SQL이다.
- 내장 SQL 실행문은 호스트 언어에서 실행문이 나타날 수 있는 곳이면 프로그램의 어느 곳에서나 사용할 수 있다.
- 일반 SQL문은 수행 결과로 여러 개의 튜플을 반환하는 반면, 내장 SQL은 단 하나의 튜플만을 반환한다.
- 내장 SQL문에 의해 반환되는 튜플은 일반 변수를 사용하여 저장할 수 있다.
- Host Program의 컴파일 시 내장 SQL문은 선행 처리기에 의해 분리되어 컴파일된다.
- 호스트 변수와 데이터베이스 필드의 이름은 같아도 된다.
- 내장 SQL문에 사용된 호스트 변수의 데이터 타입은 이에 대응하는 데이터베이스 필드의 SQL 데이터 타입과 일치하여야 한다.
- 내장 SQL문이 실행되면 SQL의 실행 상태가 SQL 상태 변수에 전달된다.
- **호스트 언어의 실행문과 SQL문을 구분시키는 방법**
  - 명령문의 구분 : C/C++에서 내장 SQL문은 $와 세미콜론(;) 문자 사이에 기술하고, Visual Basic에서는 내장 SQL문 앞에 'EXEC SQL'을 기술함
  - 변수의 구분 : 내장 SQL에서 사용하는 호스트 변수는 변수 앞에 콜론(:) 문자를 붙임

07.3, 06.9, 06.5, 06.3, 05.4, 04.3, 03.8, 03.4, 03.3, 02.9, 02.5, 01.9, 01.6, 00.10, 00.7, 00.3
14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.5, 08.3, 07.9,

핵심 036 뷰(View)

- 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된 가상 테이블이다.
- 저장장치 내에 물리적으로 존재하지 않지만, 사용자에게는 있는 것처럼 간주된다.
- 데이터 보정작업, 처리과정 시험 등 임시적인 작업을 위한 용도로 활용된다.

뷰(View)의 특징

- 기본 테이블로부터 유도된 테이블이기 때문에 기본 테이블과 같은 형태의 구조를 가지며, 조작도 기본 테이블과 거의 같다.
- 가상 테이블이기 때문에 물리적으로 구현되어 있지 않다.
- 필요한 데이터만 뷰로 정의해서 처리할 수 있기 때문에 관리가 용이하고 명령문이 간단해진다.
- 조인문의 사용을 최소화하여 사용상의 편의성을 최대화한다.
- 뷰를 통해서만 데이터에 접근하게 하면 뷰에 나타나지는 데이터를 안전하게 보호할 수 있다.
- 기본 테이블의 기본키를 포함한 속성(열) 집합으로 뷰를 구성해야만 삽입, 삭제, 갱신 연산이 가능하다.
- 정의된 뷰는 다른 뷰의 정의에 기초가 될 수 있다.
- 하나의 뷰를 삭제하면 그 뷰를 기초로 정의된 다른 뷰도 자동으로 삭제된다.

뷰의 장점

- 논리적 데이터 독립성을 제공한다.
- 동일 데이터에 대해 동시에 여러 사용자의 상이한 응용이나 요구를 지원해준다.
- 사용자의 데이터 관리를 간단하게 해준다.
- 접근 제어를 통한 자동 보안이 제공된다.

뷰의 단점

- 독립적인 인덱스를 가질 수 없다.
- 뷰의 정의를 변경할 수 없다.
- 뷰로 구성된 내용에 대한 삽입, 삭제, 갱신 연산에 제약이 따른다.

뷰 정의문

```
CREATE VIEW 뷰이름[(속성이름[,속성이름])]
AS SELECT문;
```

- SELECT문을 부질의로 사용하여 SELECT문의 결과로서 뷰를 생성한다.
- 부질의로서의 SELECT문에는 UNION이나 ORDER BY절을 사용할 수 없다.
- 속성 이름을 기술하지 않으면 SELECT문의 속성 이름이 자동으로 사용된다.

뷰 삭제문

```
DROP VIEW 뷰이름 {RESTRICT | CASCADE};
```

- RESTRICT : 뷰를 다른 곳에서 참조하고 있으면 삭제가 취소됨
- CASCADE : 뷰를 참조하는 다른 뷰나 제약 조건까지 모두 삭제됨

14.8, 13.8, 13.3, 12.5, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 08.3, 05.5, 03.3, 01.9, 01.6, 01.3, 00.7, 00.5, 00.3, 99.8

핵심 037 시스템 카탈로그

- 시스템 그 자체에 관련이 있는 다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스이다.
- 데이터베이스에 포함되는 모든 데이터 객체에 대한 정의나 명세에 관한 정보를 유지관리하는 시스템 테이블이다.
- 데이터 정의어의 결과로 구성되는 기본 테이블, 뷰, 인덱스, 패키지, 접근 권한 등의 데이터베이스 구조 및 통계 정보를 저장한다.
- 카탈로그들이 생성되면 자료 사전(Data Dictionary)에 저장되기 때문에 좁은 의미로는 카탈로그를 자료 사전이라고도 한다.
- 카탈로그에 저장된 정보를 메타 데이터(Meta-Data)라고 한다.

시스템 카탈로그의 특징

- 카탈로그 자체도 시스템 테이블로 구성되어 있어 일반 이용자도 SQL을 이용하여 내용을 검색해 볼 수 있다.
- INSERT, DELETE, UPDATE문으로 갱신하는 것은 허용하지 않는다.
- DBMS가 스스로 생성하고 유지한다.
- 카탈로그는 사용자가 SQL문을 실행시켜 기본 테이블, 뷰, 인덱스 등에 변화를 주면 시스템이 자동으로 갱신한다.

핵심 038 트랜잭션

14.8, 11.3, 10.9, 08.9, 06.3, 03.8, 01.9, 00.7, 00.3, 99.6

- 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미한다.
- 응용 프로그램이나 사용자가 데이터베이스의 내용을 접근하거나 변경하기 위해 실행되는 동작 또는 동작들의 모임이다.
- 데이터베이스 시스템에서 복구 및 병행 수행 시 처리되는 작업의 논리적 단위이다.

Atomicity (원자성)	• 트랜잭션의 연산은 데이터베이스에 모두 반영되든지 아니면 전혀 반영되지 않아야 함 • 트랜잭션 내의 모든 명령은 반드시 완벽히 수행되어야 하며, 모두가 완벽히 수행되지 않고 어느 하나라도 에러가 발생하면 트랜잭션 전부가 취소되어야 함
Consistency (일관성)	• 트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환함 • 시스템이 가지고 있는 고정 요소는 트랜잭션 수행 전과 트랜잭션 수행 완료 후의 상태가 같아야 함
Isolation (독립성, 격리성)	• 둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행중에 다른 트랜잭션의 연산이 끼어들 수 없음 • 수행중인 트랜잭션은 완전히 완료될 때까지 다른 트랜잭션에서 수행 결과를 참조할 수 없음
Durability (영속성, 지속성)	성공적으로 완료된 트랜잭션의 결과는 영구적으로 반영되어야 함

핵심 039 자료 구조의 분류

05.9, 05.4, 04.5, 04.3, 03.3, 02.5, 02.3, 01.9, 01.6, 01.3, 00.7, 00.3, 99.6, 99.4
14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.3, 11.8, 11.6, 11.3, 10.3, 09.3, 08.9, 08.5, 08.3, 07.9, 07.3, 06.9, 06.5, 06.3,

- 선형 구조 : 선형 리스트(배열), 연결 리스트, 스택, 큐, 데크
- 비선형 구조 : 트리, 그래프

핵심 040 연결 리스트(Linked List)

11.6, 08.3, 06.3, 05.4, 00.5

- 연결 리스트는 자료들을 임의의 기억공간에 기억시키되, 자료 항목의 순서에 따라 노드의 포인터 부분을 이용하여 서로 연결시킨 자료 구조이다.
- 노드의 삽입, 삭제 작업이 용이하다.
- 기억 공간이 연속적으로 놓여 있지 않아도 저장이 가능하다.
- 연결을 위한 링크(포인터) 부분이 필요하기 때문에 순차 리스트에 비해 기억 공간의 이용 효율이 좋지 않다.
- 접근 속도가 느리다.
- 희소 행렬을 링크드 리스트로 표현하면 기억 장소가 절약된다.
- 트리를 표현하기에 적합하다.

잠깐만요 ! 희소 행렬(Sparse Matrix)
행렬의 요소 중 많은 항들이 0으로 되어 있는 형태로, 기억장소를 절약하기 위해 링크드 리스트를 이용하여 저장합니다.

핵심 041 스택(Stack)

04.3, 03.8, 03.3, 02.3
14.8, 12.5, 12.3, 11.8, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 07.9, 07.5, 06.9, 06.5, 06.3, 05.5, 05.4, 04.9, 04.5,

- 리스트의 한쪽 끝으로만 자료의 삽입, 삭제 작업이 이루어지는 자료 구조이다.
- 가장 나중에 삽입된 자료가 가장 먼저 삭제되는 후입선출(LIFO; Last-In, First-Out) 방식으로 자료를 처리한다.
- TOP : Stack으로 할당된 기억공간에 가장 마지막으로 삽입된 자료가 기억된 위치를 가리키는 요소, 스택 포인터라고도 함
- Bottom : 스택의 가장 밑바닥임

Stack의 용도

- 부 프로그램 호출 시 복귀주소를 저장할 때
- 함수 호출의 순서 제어
- 인터럽트가 발생하여 복귀주소를 저장할 때
- 후위 표기법(Postfix Notation)으로 표현된 산술식을 연산할 때
- 0 주소지정방식 명령어의 자료 저장소
- 재귀(Recursive) 프로그램의 순서 제어
- 컴파일러를 이용한 언어 번역 시

핵심 042 스택의 삽입(Push)과 삭제(Pop)

13.8, 13.6, 13.3, 12.5, 11.8, 11.6, 09.8, 09.3, 07.9, 07.5, 07.3, 06.9

삽입(Push)

Top = Top + 1
If Top 〉 M Then
Overflow
Else
X(Top) ← Item
	스택 포인터(Top)를 1 증가시킨다.
	스택 포인터가 스택의 크기보다 크면 Overflow
	
	그렇지 않으면 Item이 가지고 있는 값을 스택의 Top 위치에 삽입한다.

- M : 스택의 크기
- Top : 스택 포인터
- X : 스택의 이름
- Overflow : 스택으로 할당받은 메모리 부분의 마지막 주소가 M번지라고 할 때, Top Pointer의 값이 M보다 커지면 스택의 모든 기억장소가 꽉 채워져 있는 상태이므로 더 이상 자료를 삽입할 수 없어 Overflow를 발생시킴

삭제(Pop)

If Top = 0 Then
  Underflow
Else
  Item ← X(Top)
  Top = Top−1
	스택 포인터가 0이면 스택의 바닥이어서 더 이상 삭제할 자료가 없으므로 Underflow를 처리한다.
	그렇지 않으면
	Top 위치에 있는 값을 Item으로 옮기고
	스택 포인터를 1 감소시킨다.

- Underflow : Top Pointer가 주소 0을 가지고 있다면 스택에는 삭제할 자료가 없으므로 Underflow를 발생시킴

잠깐만요 ! Stack에 기억되어 있는 자료를 삭제시킬 때는 제일 먼저 삭제할 자료가 있는지 없는지부터 확인해야 합니다.

02.3, 00.3, 99.6
핵심 14.8, 14.5, 14.3, 13.8, 13.6, 12.8, 12.5, 11.8, 11.6, 08.9, 08.5, 08.3, 07.3, 06.5, 05.5, 05.3, 04.3, 03.8, 03.5, 02.9, 02.5,

043 큐(Queue)

- 선형 리스트의 한 쪽에서는 삽입 작업이 이루어지고 다른 한 쪽에서는 삭제 작업이 이루어지도록 구성한 자료구조이다.
- 가장 먼저 삽입된 자료가 가장 먼저 삭제되는 선입선출(FIFO; First-In, First-Out) 방식으로 처리한다.
- 시작과 끝을 표시하는 두 개의 포인터를 갖는다.
  - 프런트(F, Front) 포인터 : 가장 먼저 삽입된 자료의 기억공간을 가리키는 포인터로, 삭제 작업을 할 때 사용함
  - 리어(R, Rear) 포인터 : 가장 마지막에 삽입된 자료가 위치한 기억장소를 가리키는 포인터로, 삽입 작업을 할 때 사용함

Queue를 이용하는 예

- 창구 업무처럼 서비스 순서를 기다리는 등의 대기 행렬의 처리에 사용한다.
- 운영체제의 작업 스케줄링에 사용한다.

07.5, 04.9, 02.9, 01.9, 00.10, 99.10, 99.4
핵심

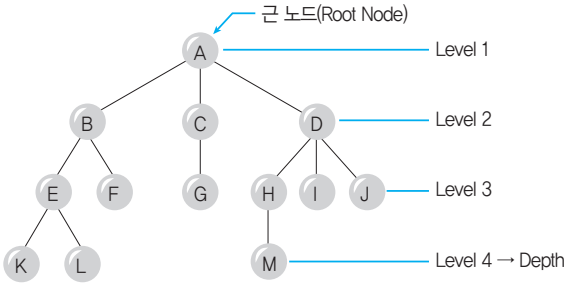
044 데크(Deque)

- 삽입과 삭제가 리스트의 양쪽 끝에서 모두 발생할 수 있는 자료 구조이다.
- Double Ended Queue의 약자이다.
- Stack과 Queue의 장점만 따서 구성한 것이다.
- 입력이 한 쪽에서만 발생하고 출력은 양쪽에서 일어날 수 있는 입력 제한과 입력은 양쪽에서 일어나고 출력은 한 곳에서만 이루어지는 출력 제한이 있다.
  - 입력 제한 데크 : Scroll
  - 출력 제한 데크 : Shelf

14.5, 14.3, 13.6, 12.3, 11.6, 11.3, 10.9, 10.3, 09.8, 08.5, 05.4, 04.5, 00.10, 99.8
핵심

045 트리(Tree)

정점(Node, 노드)과 선분(Branch, 가지)을 이용하여 사이클을 이루지 않도록 구성한 Graph의 특수한 형태로 가족의 계보(족보), 연산 수식, 회사 조직 구조도, 히프(Heap) 등을 표현하기에 적합하다.



- 노드(Node) : 트리의 기본 요소로서 자료 항목과 다른 항목에 대한 가지(Branch)를 합친 것
  예 A, B, C, D, E, F, G, H, I, J, K, L, M
- 근 노드(Root Node) : 트리의 맨 위에 있는 노드 예 A
- 디그리(Degree, 차수) : 각 노드에서 뻗어나온 가지의 수
  예 A = 3, B = 2, C = 1, D = 3
- 트리의 디그리 : 노드들의 디그리 중에서 가장 많은 수
  예 노드 A나 D가 3개의 디그리를 가지므로 위 트리의 디그리는 3이다.
- 단말 노드(Terminal Node) = 잎 노드(Leaf Node) : 자식이 하나도 없는 노드, 즉 디그리가 0인 노드
  예 K, L, F, G, M, I, J
- 비단말 노드(Non-Terminal Node) : 자식이 하나라도 있는 노드, 즉 디그리가 0이 아닌 노드
  예 A, B, C, D, E, H

- 자식 노드(Son Node) : 어떤 노드에 연결된 다음 레벨의 노드들

  예 D의 자식 노드 : H, I, J
- 부모 노드(Parent Node) : 어떤 노드에 연결된 이전 레벨의 노드

  예 E, F의 부모 노드는 B
- 형제 노드(Brother Node, Sibling) : 동일한 부모를 갖는 노드들

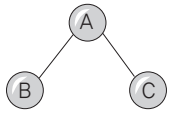
  예 H의 형제 노드는 I, J
- Level : 근 노드의 Level을 1로 가정한 후 어떤 Level이 L이면 자식 노드는 L+1이다.

  예 H의 레벨은 3
- 깊이(Depth, Height) : 어떤 Tree에서 노드가 가질 수 있는 최대의 레벨

  예 위 트리의 깊이는 4

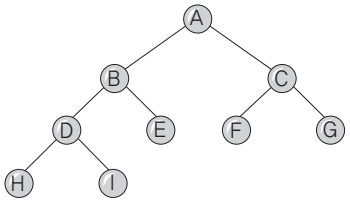
01.9, 01.6, 00.5
14.8, 14.5, 13.8, 13.6, 13.3, 12.8, 12.5, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.3, 06.5, 05.9, 05.4, 04.3, 03.8, 02.9, 02.5,

핵심 046 이진 트리의 운행법



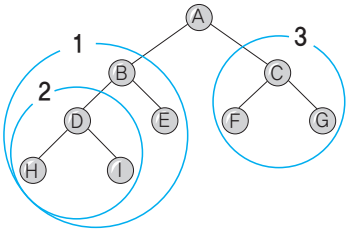
- 전위(Preorder) 운행 : Root → Left → Right 순으로 운행함 A, B, C
- 중위(Inorder) 운행 : Left → Root → Right 순으로 운행함 B, A, C
- 후위(Postorder) 운행 : Left → Right → Root 순으로 운행함 B, C, A

  예 다음 트리를 Inorder, Preorder, Postorder 방법으로 운행했을 때 각 노드를 방문한 순서는?



〈Preorder 운행법의 방문 순서〉

서브 트리를 하나의 노드로 생각할 수 있도록 다음 그림과 같이 서브 트리 단위로 묶는다. Preorder, Inorder, Postorder 모두 공통으로 사용한다.



❶ Preorder는 Root → Left → Right이므로 A13이 된다.

❷ 1은 B2E이므로 AB2E3이 된다.

❸ 2는 DHI이므로 ABDHIE3이 된다.

❹ 3은 CFG이므로 ABDHIECFG가 된다.

∴ 방문 순서 : ABDHIECFG

〈Inorder 운행법의 방문 순서〉

❶ Inorder는 Left → Root → Right이므로 1A3이 된다.

❷ 1은 2BE이므로 2BEA3이 된다.

❸ 2는 HDI이므로 HDIBEA3이 된다.

❹ 3은 FCG이므로 HDIBEAFCG가 된다.

∴ 방문 순서 : HDIBEAFCG

〈Postorder의 방문 순서〉

❶ Postorder는 Left → Right → Root이므로 13A가 된다.

❷ 1은 2EB이므로 2EB3A가 된다.

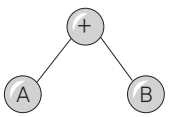
❸ 2는 HID이므로 HIDEB3A가 된다.

❹ 3은 FGC이므로 HIDEBFGCA가 된다.

∴ 방문 순서 : HIDEBFGCA

14.3, 12.5, 12.3, 09.8, 09.5, 08.9, 07.9, 07.5, 07.3, 05.5, 05.3, 04.3, 03.5,03.3, 99.4

핵심 047 수식의 표기법



- 전위 표기법(Prefix) : 연산자 → Left → Right, +AB
- 중위 표기법(Infix) : Left → 연산자 → Right, A+B
- 후위 표기법(Postfix) : Left → Right → 연산자, AB+

Infix 표기를 Postfix로 바꾸기

Infix로 표기된 수식에서 연산자를 해당 피연산자 두 개의 뒤(오른쪽)에 오도록 이동하면 Postfix가 된다.

X = A / B * (C + D) + E → X A B / C D + * E + =

❶ 연산 우선순위에 따라 괄호로 묶는다.

(X = (((A / B) * (C + D)) + E))

❷ 연산자를 해당 괄호의 뒤로 옮긴다.

X = (((A / B) * (C + D)) + E))

(X (((AB) / (CD) +) * E) +) =

❸ 괄호를 제거한다.

X A B / C D + * E + =

Infix 표기를 Prefix로 바꾸기

Infix로 표기된 수식에서 연산자를 해당 피연산자 두 개의 앞(왼쪽)에 오도록 이동하면 Prefix가 된다.

❶ 연산 우선순위에 따라 괄호로 묶는다.

(X = (((A / B) * (C + D)) + E))

❷ 연산자를 해당 괄호의 앞으로 옮긴다.

(X = (((A / B) * (C + D)) + E))

=(X + (* (/ (A B) + (C D)) E))

❸ 괄호를 제거한다.

=X + * / A B + C D E

Postfix를 Infix로 바꾸기

Postfix는 Infix 표기법에서 연산자를 해당 피연산자 2개의 뒤(오른쪽)로 이동한 것이므로 연산자를 다시 해당 피연산자 2개의 가운데로 옮기면 된다.

A B C − / D E F + * + → A / (B − C) +D * (E + F)

❶ 먼저 인접한 피연산자 2개와 오른쪽의 연산자를 괄호로 묶는다.

((A (B C −) /) (D (E F +) *) +)

❷ 연산자를 해당 피연산자의 가운데로 이동시킨다.

((A (B C −) /) (D (E F +) *) +)

((A / (B − C)) + (D * (E + F)))

❸ 필요 없는 괄호를 제거한다.

A / (B − C) + D * (E + F)

14.5, 13.3, 06.9, 05.4, 03.8, 03.5, 02.9, 02.5, 01.9, 01.3, 00.10, 00.7, 00.3, 99.10, 99.8

핵심 048 정렬(Sort)

정렬(Sort)은 파일을 구성하는 각 레코드들을 특정 키 항목을 기준으로 오름차순(Ascending) 또는 내림차순(Descending)으로 재배열하는 작업이고, 이 과정을 Sorting이라 한다.

내부 정렬

- 소량의 데이터를 주기억장치에만 기억시켜서 정렬하는 방식이다.
- **종류** : 히프 정렬, 삽입 정렬, 버블 정렬, 셸 정렬, 선택 정렬, 퀵 정렬, 2-Way Merge 정렬, 기수 정렬(=Radix Sort)

외부 정렬

- 대량의 데이터를 보조기억장치에 기억시켜서 정렬하는 방식으로, 대부분 병합 정렬(Merge Sort) 기법으로 처리한다.
- **종류** : 밸런스 병합 정렬, 케스캐이드 병합 정렬, 폴리파즈 병합 정렬, 오실레이팅 병합 정렬

정렬 알고리즘 선택 시 고려 사항

- 데이터의 양
- 초기 데이터의 배열 상태
- 키 값들의 분포 상태
- 소요공간 및 작업시간
- 사용 컴퓨터 시스템의 특성

07.3, 06.9, 05.9, 05.3

14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.5, 08.3, 07.9,

핵심 049 주요 정렬 알고리즘의 이해

삽입 정렬

예제 8, 5, 6, 2, 4를 삽입 정렬로 정렬하시오.

- 초기 상태 :

8	5	6	2	4

- 1회전 :

8	5	6	2	4

→

5	8	6	2	4

두 번째 값 5를 첫 번째 값과 비교하여 첫 번째 자리에 삽입하고 8을 한 칸 뒤로 이동시킨다.

- 2회전 :

5	8	6	2	4

→

5	6	8	2	4

세 번째 값 6을 첫 번째, 두 번째 값과 비교하여 8자리에 삽입하고 8은 한 칸 뒤로 이동시킨다.

• 3회전 : 5 6 8 2 4 → 2 5 6 8 4

네 번째 값 2를 처음부터 비교하여 맨 처음에 삽입하고 나머지를 한 칸씩 뒤로 이동시킨다.

• 4회전 : 2 5 6 8 4 → 2 4 5 6 8

다섯 번째 값 4를 처음부터 비교하여 5자리에 삽입하고 나머지를 한 칸씩 뒤로 이동시킨다.

버블 정렬

예제 8, 5, 6, 2, 4를 버블 정렬로 정렬하시오.

• 초기 상태 : 8 5 6 2 4

• 1회전 : 5 8 6 2 4 → 5 6 8 2 4 → 5 6 2 8 4 → 5 6 2 4 8

• 2회전 : 5 6 2 4 8 → 5 2 6 4 8 → 5 2 4 6 8

• 3회전 : 2 5 4 6 8 → 2 4 5 6 8

• 4회전 : 2 4 5 6 8

선택 정렬

예제 8, 5, 6, 2, 4를 선택 정렬로 정렬하시오.

• 초기 상태 : 8 5 6 2 4

• 1회전 : 5 8 6 2 4 → 5 8 6 2 4 → 2 8 6 5 4 → 2 8 6 5 4

• 2회전 : 2 6 8 5 4 → 2 5 8 6 4 → 2 4 8 6 5

• 3회전 : 2 4 6 8 5 → 2 4 5 8 6

• 4회전 : 2 4 5 6 8

2-Way 합병 정렬(Merge Sort)

예제 71, 2, 38, 5, 7, 61, 11, 26, 53, 42를 2-Way 합병 정렬로 정렬하시오.

• 1회전 : 2개씩 묶은 후 각각의 묶음 안에서 정렬합니다.

(71, 2) (38, 5) (7, 61) (11, 26) (53, 42)

↓

(2, 71) (5, 38) (7, 61) (11, 26) (42, 53)

• 2회전 : 묶여진 묶음을 2개씩 묶은 후 각각의 묶음 안에서 정렬합니다.

((2, 71) (5, 38)) ((7, 61) (11, 26)) (42, 53)

↓

(2, 5, 38, 71) (7, 11, 26, 61) (42, 53)

• 3회전 : 묶여진 묶음을 2개씩 묶은 후 각각의 묶음 안에서 정렬합니다.

((2, 5, 38, 71) (7, 11, 26, 61)) (42, 53)

↓

(2, 5, 7, 11, 26, 38, 61, 71) (42, 53)

• 4회전 : 묶여진 묶음 2개를 하나로 묶은 후 정렬합니다.

((2, 5, 7, 11, 26, 38, 61, 71) (42, 53))

↓

2, 5, 7, 11, 26, 38, 42, 53, 61, 71

핵심

12.8, 09.5, 06.5, 05.9, 05.3, 04.3, 02.5, 00.5

050 이분 검색(이진 검색)

- 제어 검색의 일종인 이분 검색은 반드시 순서화된 파일이어야 검색할 수 있다.
- 전체 파일을 두 개의 서브 파일로 분리해 가면서 Key 레코드를 검색한다.
- 찾고자 하는 Key 값을 파일의 중간 레코드 Key 값과 비교하면서 검색한다.
- 중간 레코드 번호(M) : $\frac{F+L}{2}$(단, F : 첫 번째 레코드 번호, L : 마지막 레코드 번호)

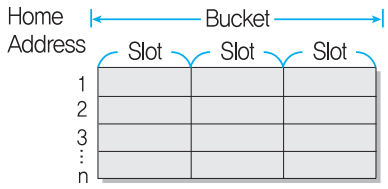
핵심 051 해싱(Hashing)

14.8, 14.5, 13.3, 12.5, 12.3, 11.8, 10.3, 09.8, 08.9, 08.3, 06.9, 06.3, 04.5, 04.3, 02.3, 01.9, 01.6, 00.10, 00.5

- Hash Table이라는 기억공간을 할당하고, 해시 함수(Hash Function)를 이용하여 레코드 키에 대한 Hash Table 내의 Home Address를 계산한 후 주어진 레코드를 해당 기억장소에 저장하거나 검색 작업을 수행하는 방식이다.
- DAM(직접접근방법) 파일을 구성할 때 해싱이 사용되며, 접근 속도는 빠르지만 기억공간이 많이 요구된다.
- 여러가지 검색 방식 중 검색 속도가 가장 빠르다.
- 삽입, 삭제 작업의 빈도가 많을 때 유리한 방식이다.
- 키-주소 변환 방법이라고도 한다.

해시 테이블(Hash Table)

- 레코드를 1개 이상 보관할 수 있는 Home Bucket들로 구성한 기억공간으로, 보조기억장치에 구성할 수도 있고 주기억장치에 구성할 수도 있다.



* n크기의 3개의 슬롯으로 구성된 버킷을 가진 해시표

- 버킷(Bucket) : 하나의 주소를 갖는 파일의 한 구역을 의미하며, 버킷의 크기는 같은 주소에 포함될 수 있는 레코드 수를 의미함
- 슬롯(Slot) : 1개의 레코드를 저장할 수 있는 공간으로 n개의 슬롯이 모여 하나의 버킷을 형성함
- Collision(충돌 현상) : 서로 다른 2개 이상의 레코드가 같은 주소를 갖는 현상
- Synonym : 같은 Home Address를 갖는 레코드들의 집합
- Overflow
  - 계산된 Home Address의 Bucket 내에 저장할 기억공간이 없는 상태
  - Bucket을 구성하는 Slot이 여러 개일 때는 Collision은 발생해도 Overflow는 발생하지 않을 수 있음

핵심 052 해시 함수(Hash Function)

08.5, 08.3, 06.3, 05.3, 04.3, 99.10

제산(Divide)법	코드 키(K)를 해시표(Hash Table)의 크기보다 큰 수 중에서 가장 작은 소수(Prime, Q)로 나눈 나머지를 홈 주소로 삼는 방식, 즉 h(K) = K mod Q임
제곱(Mid-Square)법	레코드 키값(K)을 제곱한 후 그 중간 부분의 값을 홈 주소로 삼는 방식
폴딩(Folding)법 (접지법)	레코드 키값(K)을 여러 부분으로 나눈 후 각 부분의 값을 더하거나 XOR(배타적 논리합)한 값을 홈 주소로 삼는 방식
기수(Radix) 변환법	키 숫자의 진수를 다른 진수로 변환시켜 주소 크기를 초과한 높은 자릿수는 절단하고, 이를 다시 주소 범위에 맞게 조정하는 방식
대수적 코딩(Algebraic Coding)법	키값을 이루고 있는 각 자리의 비트수를 한 다항식의 계수로 간주하고, 이 다항식을 해시표의 크기에 의해 정의된 다항식으로 나누어 얻은 나머지 다항식의 계수를 홈 주소로 삼는 방식
계수 분석(Digit Analysis)법 (숫자 분석법)	주어진 모든 키 값들에서 키값을 이루는 숫자의 분포를 분석하여 비교적 고른 자리를 필요한 만큼 택해서 홈 주소로 삼는 방식
무작위(Random)법	난수(Random Number)를 발생시켜 나온 값을 홈 주소로 삼는 방식

핵심 053 순차 파일(Sequential File) = 순서 파일

14.3, 11.6, 10.5, 09.5, 07.5, 07.3, 05.3

- 입력되는 데이터들을 논리적인 순서에 따라 물리적 연속 공간에 순차적으로 기록하는 방식이다.
- 급여 관리 등과 같이 변동 사항이 크지 않고 기간별로 일괄 처리를 주로 하는 경우에 적합하다.
- 주로 순차 접근이 가능한 자기 테이프에서 사용된다.

순차 파일의 장점

- 기록 밀도가 높아 기억공간을 효율적으로 사용할 수 있다.
- 레코드가 키 순서대로 편성되어 취급이 용이하다.
- 매체 변환이 쉬워 어떠한 매체에도 적용할 수 있다.
- 레코드를 기록할 때 사용한 키 순서대로 레코드를 처리하는 경우, 다른 편성법보다 처리 속도가 빠르다.

순차 파일의 단점

- 파일에 새로운 레코드를 삽입, 삭제, 수정하는 경우 파일 전체를 복사해야 하므로 시간이 많이 소요된다.
- 데이터 검색 시 처음부터 순차적으로 하기 때문에 검색 효율이 낮다.

핵심 14.8, 14.3, 13.8, 13.6, 10.9, 10.5, 07.9, 07.5, 05.4, 03.8, 03.3, 02.9, 00.7, 00.5, 99.10, 99.8, 99.6

054 색인 순차 파일(Indexed Sequential File)

- 순차 처리와 랜덤 처리가 모두 가능하도록 레코드들을 키 값 순으로 정렬(Sort)시켜 기록하고, 레코드의 키 항목만을 모은 색인을 구성하여 편성하는 방식이다.
- 색인을 이용한 순차적인 접근 방법을 제공하여 ISAM (Index Sequential Access Method)이라고도 한다.
- 레코드를 참조하는 경우 색인을 탐색한 후 색인이 가리키는 포인터(주소)를 사용하여 직접 참조할 수 있다.
- 일반적으로 자기 디스크에 많이 사용되며, 자기 테이프에서는 사용할 수 없다.

색인 순차 파일의 구성

- 기본 구역(Prime Data Area) : 실제 레코드들을 기록하는 부분으로, 각 레코드는 키 값 순으로 저장됨
- 색인 구역(Index Area) : 기본 구역에 있는 레코드들의 위치를 찾아가는 색인이 기록되는 부분으로, 트랙 색인 구역, 실린더 색인 구역, 마스터 색인 구역으로 구분할 수 있음
- 오버플로 구역(Overflow Area) : 기본 구역에 빈 공간이 없어서 새로운 레코드의 삽입이 불가능할 때를 대비하여 예비적으로 확보해 둔 부분

실린더 오버플로 구역(Cylinder Overflow Area)	각 실린더마다 만들어지는 오버플로 구역으로, 해당 실린더의 기본 구역에서 오버플로된 데이터를 기록함
독립 오버플로 구역 (Independent Overflow Area)	실린더 오버플로 구역에 더 이상 오버플로된 데이터를 기록할 수 없을 때 사용할 수 있는 예비 공간으로, 실린더 오버플로 구역과는 별도로 만들어짐

색인 순차 파일의 장점

- 순차 처리와 랜덤 처리가 모두 가능하므로, 목적에 따라 융통성 있게 처리할 수 있다.
- 효율적인 검색이 가능하고 레코드의 삽입, 삭제, 갱신이 용이하다.

색인 순차 파일의 단점

- 색인 구역과 오버플로 구역을 구성하기 위한 추가 기억 공간이 필요하다.
- 파일이 정렬되어 있어야 하므로 추가, 삭제가 많으면 효율이 떨어진다.
- 색인을 이용한 액세스를 하기 때문에 액세스 시간이 랜덤 편성 파일보다 느리다.

2과목 · 전자계산기 구조

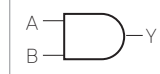
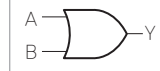
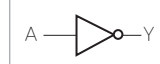
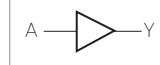
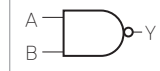
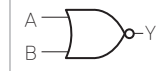
핵심 14.3, 13.6, 12.5, 11.3, 10.9, 07.3, 06.5, 05.5, 04.9, 04.3, 03.8, 02.3

055 불 대수의 기본 공식

- 교환법칙 : $A + B = B + A$, $A \cdot B = B \cdot A$
- 결합법칙 : $A + (B + C) = (A + B) + C$, $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
- 분배법칙 : $A \cdot (B+C) = A \cdot B + A \cdot C$, $A+B \cdot C = (A+B) \cdot (A+C)$
- 멱등법칙 : $A + A = A$, $A \cdot A = A$
- 보수법칙 : $A + \overline{A} = 1$, $A \cdot \overline{A} = 0$
- 항등법칙 : $A + 0 = A$, $A + 1 = 1$, $A \cdot 0 = 0$, $A \cdot 1 = A$
- 드모르강 : $\overline{A} + \overline{B} = \overline{(A \cdot B)}$, $\overline{A \cdot B} = (\overline{A}+\overline{B})$
- 복원법칙 : $A = A$

핵심 14.5, 13.8, 12.8, 07.9, 06.5, 05.5, 04.9, 03.8, 02.5, 02.3, 01.9, 01.6, 00.10, 00.3

056 논리 게이트

게이트	기호	의미	진리표	논리식
AND		입력신호가 모두 1일 때 1 출력	A B | Y 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	$Y = A \cdot B$ $Y = AB$
OR		입력신호 중 1개만 1이어도 1출력	A B | Y 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1	$Y = A + B$
NOT		입력된 정보를 반대로 변환하여 출력	A | Y 0 | 1 1 | 0	$Y = A'$ $Y = \overline{A}$
BUFFER		입력된 정보를 그대로 출력	A | Y 0 | 0 1 | 1	$Y = A$
NAND		NOT + AND, 즉 AND의 부정	A B | Y 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0	$Y = \overline{A \cdot B}$ $Y = \overline{AB}$
NOR		NOT + OR, 즉 OR의 부정	A B | Y 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0	$Y = \overline{A + B}$
XOR		입력신호가 모두 같으면 0, 한 개라도 틀리면 1출력	A B | Y 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0	$Y = A \oplus B$ $Y = \overline{A}B + A\overline{B}$
XNOR		NOT + XOR, 즉 XOR의 부정	A B | Y 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1	$Y = A \odot B$ $Y = \overline{A \oplus B}$ $Y = AB + \overline{A}\overline{B}$

핵심 057 조합논리회로와 순서논리회로

13.8, 12.8, 12.3, 08.9, 05.3, 02.3

- 조합논리회로는 임의의 시간에서의 출력이 이전의 입력에는 관계없이 현재의 입력 조합(0 또는 1)으로부터 직접 결정되는 논리회로이다. 이해 반해 순서논리회로는 외부로부터의 입력과 현재 상태에 따라 출력이 결정된다.
- **조합논리회로의 종류** : 반가산기, 전가산기, 병렬가산기, 반감산기, 전감산기, 디코더, 인코더, 멀티플렉서, 디멀티플렉서, 다수결회로, 비교기 등
- **순서논리회로의 종류** : 플립플롭, 레지스터, 카운터, RAM, CPU 등

핵심 058 반가산기 (HA; Half Adder)

12.3, 11.6, 10.5, 08.9, 07.5, 05.9, 04.5, 02.9, 01.6

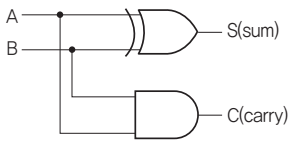
1Bit짜리 2진수 2개를 덧셈한 합(S)과 자리올림 수(C)를 구하는 회로이다.

진리표

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

논리식 : $Carry = A \cdot B \quad Sum = \overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$

논리회로



핵심 059 전가산기 (FA; Full Adder)

14.5, 11.8, 11.6., 10.5, 08.5, 06.3, 99.6

자리올림 수(C_i)를 포함하여 1Bit 크기의 2진수 3자리를 더하여 합(Sum)과 자리올림 수(Carry)를 구하는 회로이다.

진리표

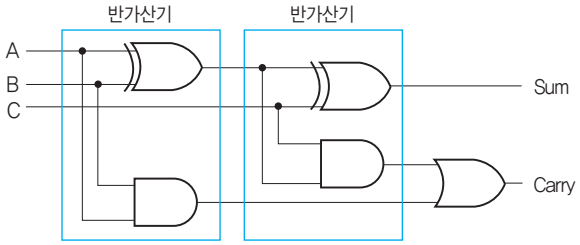
A	B	C_i	Sum	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

논리식

- 합계(Sum) = $(A \oplus B) \oplus C_i$
- 자리올림(Carry) = $(A \oplus B)C_i + AB$

회로

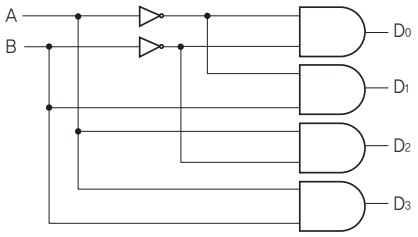
전가산기는 2개의 반가산기(HA)와 1개의 OR Gate로 구성된다.



핵심 060 디코더(Decoder)

09.8, 07.9, 07.5, 06.3

- n Bit의 Code화된 정보를 그 Code의 각 Bit 조합에 따라 2^n개의 출력으로 번역하는 회로이다.
- 명령어의 명령부나 번지를 해독할 때 사용하며, 주로 AND 게이트로 구성된다.
- **회로**



핵심 061 플립플롭

13.6, 13.3, 10.9, 08.9, 08.3, 06.5, 06.3, 05.3, 02.9

- 플립플롭은 전원이 공급되고 있는 한, 상태의 변화를 위한 신호가 발생할 때까지 현재의 상태를 그대로 유지하는 대표적인 순서 논리회로이다.
- 플립플롭 1개가 1Bit를 구성하는 2진 셀(Binary Cell)이 된다.
- 반도체 기억장치에서 2진수 1자리값을 기억하는 메모리 소자이다.
- 플립플롭은 레지스터를 구성하는 기본 소자이다.
- 기본적인 플립플롭은 2개의 NAND 또는 NOR 게이트를 이용하여 구성한다.

플립플롭	특징
RS	플립플롭의 기본으로, S와 R선의 입력을 조절하여 임의의 Bit 값을 그대로 유지시키거나, 무조건 0 또는 1의 값을 기억시키기 위해서 사용됨
JK	• RS FF에서 S=R=1일 때 동작되지 않는 결점을 보완한 플립플롭 • RS FF의 입력선 S와 R에 AND 게이트 2개를 추가하여 JK FF의 입력선 J와 K로 사용함 • 모든 플립플롭의 기능을 포함함
D	• RS FF의 R선에 인버터(Inverter)를 추가하여 S선과 하나로 묶어서 입력선을 하나만 구성한 플립플롭 • 입력하는 값을 그대로 저장하는 기능을 수행함
T	• JK FF의 두 입력선을 묶어서 한 개의 입력선으로 구성한 플립플롭 • T=0인 경우는 변화가 없고, T=1인 경우에 현재의 상태를 토글(Toggle)시킴. 즉 원 상태와 보수 상태의 2가지 상태로만 서로 전환됨
마스터-슬레이브 (M/S)	• 출력측의 일부가 입력측에 궤환(FeedBack)되어 유발되는 레이스 현상을 없애기 위해 고안된 플립플롭 • 2개의 플립플롭으로 구성되는 데, 한쪽 회로가 마스터이고 다른 한쪽이 슬레이브의 위치에 있어 마스터-슬레이브 플립플롭이라 함

특성 표

〈RS 플립플롭〉

S	R	$Q_{(T+1)}$	암기
0	0	$Q_{(T)}$	무(상태 변화 없음)
0	1	0	공(항상 0)
1	0	1	일(항상 1)
1	1	동작 안 됨	불(불가)

〈JK 플립플롭〉

J	K	$Q_{(T+1)}$	암기
0	0	$Q_{(T)}$	무(상태 변화 없음)
0	1	0	공(항상 0)
1	0	1	일(항상 1)
1	1	보수	보(보수)

잠깐만요 ! JK 플립플롭은 J와 K에 모두 1이 입력될 때 보수가 출력되는 것이 RS 플립플롭과 다릅니다.

핵심 062 자료 구성의 단위

07.5, 07.3, 06.5, 01.9, 01.3, 00.10, 00.3, 99.10, 99.8, 99.4

비트(Bit, Binary Digit)	• 자료(정보) 표현의 최소 단위 • 2가지 상태(0과 1)를 표시하는 2진수 1자리
니블(Nibble)	• 4개의 비트(Bit)가 모여 1개의 Nibble을 구성함 • 4비트로 구성되며 16진수 1자리를 표현하기에 적합함
바이트(Byte)	• 문자를 표현하는 최소 단위로, 8개의 비트(Bit)가 모여 1Byte를 구성함 • 1Byte는 256(2^8)가지의 정보를 표현할 수 있음 • 주소 지정의 단위로 사용됨
워드(Word)	• CPU가 한 번에 처리할 수 있는 명령 단위 • 반워드(Half Word) : 2Byte • 풀워드(Full Word) : 4Byte • 더블워드(Double Word) : 8Byte
필드(Field)	• 파일 구성의 최소 단위 • 의미 있는 정보를 표현하는 최소 단위
레코드 (Record)	• 하나 이상의 관련된 필드가 모여서 구성 • 컴퓨터 내부의 자료 처리 단위로서, 일반적으로 레코드는 논리 레코드(Logical Record)를 의미함
블록(Block) 물리 레코드 (PhysicalRecord)	• 하나 이상의 논리 레코드가 모여서 구성 • 각종 저장 매체와의 입·출력 단위를 의미하며, 일반적으로 물리 레코드(Physical Record)라고 함
파일(File)	프로그램 구성의 기본 단위로, 여러 레코드가 모여서 구성됨
데이터베이스 (Database)	여러 개의 관련된 파일(File)의 집합

핵심 063 진법 변환

14.8, 14.5, 14.3, 12.3, 10.3, 09.8, 09.5, 08.3, 07.5, 07.3, 06.9, 06.3, 05.9, 05.3, 04.5, 04.3, 02.5, 00.3, 99.8

10진수를 2진수, 8진수, 16진수로 변환

- 정수 부분 : 10진수의 값을 변환할 진수로 나누어 더 이상 나눠지지 않을 때까지 나누고, 나머지를 역순으로 표시함
- 소수 부분 : 10진수의 값에 변환할 진수를 곱한 후 결과의 정수 부분만을 차례대로 표기하되, 소수 부분이 0 또는 반복되는 수가 나올 때까지 곱하기를 반복함

예 $(47.625)_{10}$를 2진수, 8진수, 16진수로 변환하기

〈정수 부분〉

2진수	8진수	16진수
2) 47 2) 23 … 1 2) 11 … 1 2) 5 … 1 2) 2 … 1 1 … 0	8) 47 5 … 7 →	16) 47 2 … 15(F) →
$(47)_{10} = (101111)_2$	$(47)_{10} = (57)_8$	$(47)_{10} = (2F)_{16}$

〈소수 부분〉

2진수	8진수	16진수
0.625 × 2 = 1.250 → 0.250 × 2 = 0.5 → 0.5 × 2 = 1.0	0.625 × 8 = 5.000	0.625 × 16 = 10(A).000
$(0.625)_{10} = (0.101)_2$	$(0.625)_{10} = (0.5)_8$	$(0.625)_{10} = (0.A)_{16}$
$(47.625)_{10} \rightarrow (101111.101)_2$	$(47.625)_{10} \rightarrow (57.5)_8$	$(47.625)_{10} \rightarrow (2F.A)_{16}$

2진수, 8진수, 16진수를 10진수로 변환

정수 부분과 소수 부분의 각 자리를 분리하여 변환하려는 각 진수의 자리값과 자리의 지수승을 곱한 결과값을 모두 더하여 계산한다.

예 $(101111.101)_2$를 10진수로 변환하기

$$
\begin{aligned}
&(1\ 0\ 1\ 1\ 1\ 1\ .\ 1\ 0\ 1)_2\\
&\quad \times\ \times\ \times\ \times\ \times\ \times\ \ \times\ \times\ \times\\
&= 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0\ .\ 2^{-1}\ 2^{-2}\ 2^{-3}\\
&= 32 + 0 + 8 + 4 + 2 + 1\ .\ 0.5 + 0 + 0.125\\
&= 47.625
\end{aligned}
$$

예 $(57.5)_8$를 10진수로 변환하기

$$
\begin{aligned}
&(5\ \ 7\ .\ 5)_8\\
&\quad \times\ \ \times\ \ \times\\
&= 8^1\ \ 8^0\ .\ 8^{-1}\\
&= 40 + 7\ .\ 0.625\\
&= 47.625
\end{aligned}
$$

예 $(4F.2)_{16}$를 10진수로 변환하기

$$
\begin{aligned}
&(4\ \ F\ .\ \ 2)_{16}\\
&\quad \times\ \ \times\ \ \ \times\\
&= 16^1 + 16^0\ .\ 16^{-1}\\
&= 64 + 15\ .\ 0.125\\
&= 79.125
\end{aligned}
$$

2진수, 8진수, 16진수 상호 변환

- **2진수를 8진수로** : 정수 부분은 소수점을 기준으로 왼쪽 방향으로 3자리씩, 소수 부분은 소수점을 기준으로 오른쪽 방향으로 3자리씩 묶어서 변환함
- **2진수를 16진수로** : 정수 부분은 소수점을 기준으로 왼쪽방향으로 4자리씩, 소수 부분은 소수점을 기준으로 오른쪽 방향으로 4자리씩 묶어서 변환함
- **8진수, 16진수를 2진수로** : 8진수 1자리는 2진수 3비트로, 16진수 1자리는 2진수 4비트로 풀어서 변환함
- **8진수를 16진수로** : 8진수를 2진수로 변환한 뒤 2진수를 16진수로 변환함
- **16진수를 8진수로** : 16진수를 2진수로 변환한 뒤 2진수를 8진수로 변환함

핵심 064 보수

05.9, 05.3, 04.3, 03.3, 02.9, 01.9, 99.10, 99.8

컴퓨터가 기본적으로 수행하는 덧셈 회로를 이용하여 뺄셈을 수행하기 위해 사용한다.

r의 보수	• 10진법에는 10의 보수가 있고, 2진법에는 2의 보수가 있음 • 보수를 구할 숫자의 자릿수만큼 0을 채우고 가장 왼쪽에 1을 추가하여 기준을 만듦 예 33의 10의 보수는? 33 + X = 100 → X = 100 − 33 → X = 67 예 10101의 2의 보수는? 10101 + X = 100000 → X = 100000 − 10101 → X = 01011
r−1의 보수	• 10진법에는 9의 보수가 있고, 2진법에는 1의 보수가 있음 • 10진수 N에 대한 9의 보수는 주어진 숫자의 자릿수 만큼 9를 채워 기준을 만듦 예 33의 9의 보수는? 33 + X = 99 → X = 99 − 33 → X = 66 • 2진수 N에 대한 1의 보수는 주어진 숫자의 자릿수 만큼 1을 채워 기준을 만듦 예 10101의 1의 보수는? 10101 + X = 11111 → X = 11111 − 10101 → X = 01010

01.6, 0.5, 99.6

핵심 065 2진 연산

14.5, 13.8, 12.8, 11.8, 11.6, 11.3, 09.5, 08.9, 07.5, 07.3, 06.9, 06.3, 05.5, 05.3, 04.9, 04.5, 04.3, 03.8, 02.9, 02.5, 02.3,

- 정수값을 2진수로 변환하여 표현하는 방식이다.
- 표현할 수 있는 범위는 작지만 연산 속도가 빠르다.
- n Bit 크기의 워드가 있을 때 맨 처음 1Bit는 부호(Sign) 비트로 사용되고 나머지 n-1 Bit에 2진수로 표현된 정수값이 저장된다.
- **양수** : 부호 비트에 0을 넣고, 변환된 2진수 값을 Data Bit의 오른쪽에서 왼쪽 순으로 차례로 채우고 남는 자리에 0을 채움
- **음수** : 음수를 표현할 때는 다음과 같은 3가지 방법을 사용함

종류	표현 방법	비고
부호화 절대치법 (Signed Magnitude)	양수 표현에 대하여 부호 Bit의 값만 0을 1로 바꿈	2가지 형태의 0 존재(+0, -0)
부호화 1의 보수법 (Signed 1's Comple-ment)	양수 표현에 대하여 1의 보수를 취함	
부호화 2의 보수법 (Signed 2's Comple-ment)	양수 표현에 대하여 2의 보수를 취함	한 가지 형태의 0만 존재 (+0)

표현 범위

종류	범위	n=8	n=16	n=32
부호화 절대치법	$-2^{n-1}+1 \sim +2^{n-1}-1$	-127 ~ +127	-32767 ~ +32767	$-2^{31}+1 \sim +2^{31}-1$
부호화 1의 보수법				
부호화 2의 보수법	$-2^{n-1} \sim +2^{n-1}-1$	-128 ~ +127	-32768 ~ +32767	$-2^{31} \sim +2^{31}-1$

핵심 066 10진 연산

12.8, 11.6, 10.9, 09.8, 09.5, 08.5, 05.9, 05.4, 00.7, 99.4

10진수 1자리를 2진수 4자리로 표현하는 방식으로, 언팩(Unpack) 연산과 팩(Pack) 연산이 있다.

언팩(Unpack) 연산

- 존(Zone)형 10진 연산이라고도 한다.
- 연산이 불가능하고, 데이터의 입 · 출력에 사용된다.
- 1Byte로 10진수 1자리를 표현한다.
- 4개의 존(Zone) 비트와 4개의 숫자(Digit) 비트를 사용한다.
- 최하위(가장 오른쪽) 바이트의 존(Zone) 부분을 부호로 사용한다.

Zone	Digit	Zone	Digit	Zone	Digit	···	Sign	Digit

- **Zone 부분** : 무조건 1111을 넣음
- **Digit 부분** : 10진수 1자리를 4Bit 2진수로 표현함
- **Sign 부분** : 양수는 C(1100_2), 음수는 D(1101_2), 부호 없는 양수는 F(1111_2)로 표현함

팩(Pack) 연산

- 연산이 가능하고, 데이터의 입 · 출력이 불가능하다.
- 1Byte로 10진수 2자리를 표현한다.
- 최하위(가장 오른쪽) 바이트의 4Bit 부분을 부호로 사용한다.

Digit	Digit	Digit	Digit	Digit	Digit	···	Digit	Sign

- **Digit 부분** : 10진수 1자리를 4Bit 2진수로 표현함
- **Sign 부분** : 양수는 C(1100_2), 음수는 D(1101_2), 부호 없는 양수는 F(1111_2)로 표현함

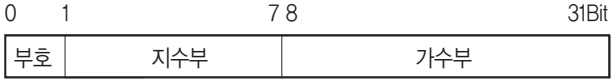
핵심 067 부동 소수점 표현

14.3, 13.6, 11.8, 11.6, 09.8, 08.3, 06.9, 05.9, 05.5, 02.3, 01.6

부동 소수점 방식은 소수점이 포함된 실수 데이터의 표현과 연산에 사용하는 방식이다.

부동 소수점 방식의 특징

- 고정 소수점 방식으로 표현하는 것보다 매우 큰 수나 작은 수, 매우 정밀한 수를 적은 비트로 표현할 수 있다.
- 과학이나 공학 또는 수학적인 응용에 주로 사용된다.
- 고정 소수점 방식에 비해 연산 시간이 많이 걸린다.
- 지수부와 가수부를 분리하는 정규화 과정이 필요하다.
- 정규화의 목적은 유효 자릿수를 최대로 하여 수의 정밀도를 높이기 위한 것이다.
- 4Byte를 사용하는 단정도와 가수부를 4Byte 추가하여 좀더 정밀하게 표현할 수 있는 8Byte 배정도 표현법이 있다.



부동 소수점 수의 연산 방법

• 덧셈, 뺄셈

❶ 0인지의 여부를 조사한다.

❷ 가수의 위치 조정 : 두 자료의 지수를 비교한 후 소수점의 위치를 이동하여 지수가 큰 쪽에 맞춘다.

❸ 가수부 값끼리 더하거나 뺀다.

❹ 결과를 정규화한다.

• 곱셈

❶ 0인지의 여부를 조사한다.

❷ 지수를 더한다.

❸ 가수를 곱한다.

❹ 결과를 정규화한다.

• 나눗셈

❶ 0인지의 여부를 조사한다.

❷ 부호를 결정한다.

❸ 피제수가 제수보다 작게 피제수의 위치를 조정한다.

❹ 지수의 뺄셈을 한다.

❺ 가수의 나눗셈을 한다.

핵심 068 자료의 외부적 표현

12.5, 08.5, 07.3, 06.5, 05.5, 05.3, 04.9, 03.8, 03.5, 02.9, 01.6, 99.10

BCD (Binary Coded Decimal, 2진화 10진 코드)	• 6Bit 코드로 IBM 사에서 개발 • 1개의 문자를 2개의 Zone 비트와 4개의 Digit 비트로 표현함 • 6Bit는 2^6개를 표현할 수 있으므로 64개의 문자를 표현할 수 있음 • 1Bit의 Parity Bit를 추가하여 7Bit로 사용함 • 영문 소문자를 표현하지 못함
ASCII 코드(American Standard Code for Information Interchange)	• 7Bit 코드로 미국 표준협회에서 개발 • 1개의 문자를 3개의 Zone 비트와 4개의 Digit 비트로 표현함 • 2^7=128가지의 문자를 표현할 수 있음 • 1Bit의 Parity Bit를 추가하여 8Bit로 사용함 • 통신 제어용 및 마이크로 컴퓨터의 기본 코드임
EBCDIC (Extended BCD Interchange Code, 확장 2진화 10진 코드)	• 8Bit 코드로 IBM에서 개발 • 1개의 문자를 4개의 Zone 비트와 4개의 Digit 비트로 표현함 • 2^8=256가지의 문자를 표현할 수 있음 • 1Bit의 Parity Bit를 추가하여 9Bit로 사용함 • 대형 기종의 컴퓨터에서 사용함

02.5, 01.3, 01.9, 00.10, 99.10, 99.8, 99.6

핵심 069 기타 자료의 표현 방식

13.8, 13.3, 11.6, 11.3, 10.3, 09.8, 09.3, 07.9, 07.5, 07.3, 06.9, 06.5, 06.3, 05.9, 05.5, 05.3, 04.5, 04.3, 03.8, 03.5, 03.3,

BCD 코드	• 10진수 1자리의 수를 2진수 4Bit로 표현함 • 4Bit의 2진수 각 Bit가 8(2^3), 4(2^2), 2(2^1), 1(2^0)의 자리값을 가지므로 8421 코드라고도 함 • 대표적인 가중치 코드 • 문자 코드인 BCD에서 Zone 부분을 생략한 형태임 • 10진수 입 · 출력이 간편함
Excess-3 코드 (3초과 코드)	• BCD + 3, 즉 BCD 코드에 3을 더하여 만든 코드임 • 대표적인 자보수 코드이며, 비가중치 코드임
Gray 코드	• BCD 코드의 인접하는 비트를 X-OR 연산하여 만든 코드 • 입 · 출력장치, A/D 변환기, 주변장치 등에서 숫자를 표현할 때 사용 • 1Bit만 변화시켜 다음 수치로 증가시키기 때문에 하드웨어적인 오류가 적음
패리티 검사 코드	• 코드의 오류를 검사하기 위해서 데이터 비트 외에 1Bit의 패리티 체크 비트를 추가하는 것으로 1Bit의 오류만 검출할 수 있음 • Odd(기수) Parity : 코드에서 1인 Bit의 수가 홀수가 되도록 0이나 1을 추가함 • Even(우수) Parity : 코드에서 1인 Bit의 수가 짝수가 되도록 0이나 1을 추가함
해밍 코드	• 오류를 스스로 검출하여 교정이 가능한 코드 • 1Bit의 오류만 교정할 수 있음 • 데이터 비트 외에 에러 검출 및 교정을 위한 잉여 비트가 많이 필요함 • 해밍 코드 중 1, 2, 4, 8, 16 …… 2^n 번째 비트는 오류 검출을 위한 패리티 비트임
허프만 코드	사용되는 문자의 빈도수에 따라 코드의 길이가 달라짐

핵심 070 그레이 코드 변환

14.3, 10.3, 09.3, 06.3, 05.9, 04.9, 04.5, 03.3, 02.5, 01.6, 01.3

2진수를 Gray Code로 변환하는 방법

❶ 첫 번째 그레이 비트는 2진수의 첫 번째 비트를 그대로 내려쓴다.

❷ 두 번째 그레이 비트부터는 변경할 2진수의 해당 번째 비트와 그 왼쪽의 비트를 XOR 연산하여 쓴다.

예 2진수 1001을 Gray Code로 변환하시오.

2진수 : 1 ⊕ 0 ⊕ 0 ⊕ 1

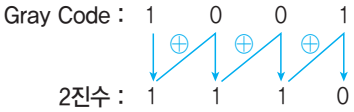
그레이 코드 : 1 1 0 1

Gray Code를 2진수로 변환하는 방법

❶ 첫 번째 2진수 비트는 그레이 코드의 첫 번째 비트를 그대로 내려쓴다.

❷ 두 번째 2진수 비트부터는 왼쪽에 구해 놓은 2진수 비트와 변경할 해당 번째 그레이 비트를 XOR 연산하여 쓴다.

예 Gray Code 1001을 2진수로 변환하시오.



핵심 071 코드의 분류

13.6, 12.3, 08.5, 08.3, 04.9, 02.5, 00.10, 99.8

분류	코드 종류
가중치 코드 (Weight Code)	BCD(8421), 2421, 84-2-1, Biquinary(5043210), 51111, Ring-Counter(9876543210)
비가중치 코드 (Non-Weight Code)	3 초과(Excess-3), Gray, Jonson, 2-Out-of-5, 3-Out-of-5
자보수 코드 (Self-Complement Code)	Excess-3, 2421, 51111, 84-2-1
오류 검출용 코드	해밍 코드, 패리티 검사 코드, Biquinary, Ring-Counter, 2-Out-of-5, 3-Out-of-5

핵심 072 중앙처리장치의 구성 요소

14.5, 14.3, 13.8, 12.5, 11.3, 10.3, 08.5, 05.3, 04.5, 03.8, 03.3, 02.9, 02.5, 02.3, 01.9, 01.6, 00.7, 00.3, 99.10, 99.4

제어 장치

- 컴퓨터에 있는 모든 장치들의 동작을 지시하고 제어하는 장치
- 주기억장치에서 읽어 들인 명령어를 해독하여 해당하는 장치에게 제어 신호를 보내 정확하게 수행하도록 지시함
- 제어장치의 구성 요소
  - 명령 레지스터 : 현재 실행중인 명령어의 내용을 기억하고 있는 레지스터
  - 명령 해독기(Decoder) : 명령 레지스터에 있는 명령어를 해독하는 회로
  - 제어신호 발생기, 부호기(Encoder) : 해독된 명령에 따라 각 장치로 보낼 제어 신호를 생성하는 회로
  - 제어 주소 레지스터(CAR) : 다음에 실행할 마이크로명령어의 주소를 저장하는 레지스터로, Mapping의 결과값, 주소필드, 서브루틴 레지스터의 내용들이 적재되어 있음
  - 제어 버퍼 레지스터(CBR) : 제어 기억장치로부터 읽혀진 마이크로명령어 비트들을 일시적으로 저장하는 레지스터
  - 제어 기억장치 : 마이크로명령어들로 이루어진 마이크로프로그램을 저장하는 내부 기억장치
  - 순서 제어 모듈 : 마이크로명령어의 실행 순서를 결정하는 회로들의 집합
  - 순차 카운터(Sequence counter) : 디코더에 의해 선택된 번호에 해당하는 타이밍 신호를 생성
- 제어장치에 입력되는 항목 : 명령어 레지스터, 플래그, 클록

연산 장치 (ALU)

- 제어장치의 명령에 따라 실제로 연산을 수행하는 장치
- 산술연산, 논리연산, 관계연산, 이동(Shift) 등의 연산을 수행함
- 가산기, 누산기(AC; Accumulator), 보수기, 데이터 레지스터, 오버플로 검출기, Shift Register 등으로 구성되어 있음

레지스터

- CPU 내부에서 처리할 명령어나 연산의 중간 결과값 등을 일시적으로 기억하는 임시 기억장소
- 플립플롭(Flip-Flop)이나 래치(Latch)들을 병렬로 연결하여 구성함
- 메모리 중에서 가장 속도가 빠름
- 레지스터의 크기는 워드를 구성하는 비트 개수만큼의 플립플롭으로 구성되며, 여러 개의 플립플롭은 공통 클록의 입력에 의해 동시에 여러 비트의 자료가 저장됨
- 레지스터를 구성하는 플립플롭은 저장하는 값을 임의로 설정하기 위해 별도의 입력 단자를 추가할 수 있으며, 저장값을 0으로 하는 것을 설정해제(CLR)라 함
- 레지스터 간의 자료 전송
  - 직렬 전송 : 직렬 시프트 마이크로 오퍼레이션을 뜻하며, 병렬 전송에 비해 전송속도가 느림
  - 병렬 전송 : 하나의 클록 펄스 동안에 레지스터 내의 모든 비트, 즉 워드가 동시에 전송되는 전송 방식
  - 버스 전송 : 모든 레지스터들이 공통으로 이용하는 경로로, 병렬 전송에 비해 결선의 수를 줄일 수 있다는 장점이 있음

핵심 073 주요 레지스터

01.3, 00.10, 00.7, 00.3, 99.10, 99.10, 99.8
14.5, 14.3, 13.8, 13.6, 13.3, 12.5, 12.3, 11.6, 10.3, 09.5, 09.3, 08.9, 08.3, 07.3, 06.9, 06.3, 05.9, 04.5, 02.9, 02.5, 01.6,

레지스터	기능
프로그램 카운터, 프로그램 계수기 (PC; Program Counter)	• 다음 번에 실행할 명령어의 번지를 기억하는 레지스터 • 분기 명령이 실행되는 경우 그 목적지 주소로 갱신됨
명령 레지스터 (IR, Instruction Register)	현재 실행중인 명령의 내용을 기억하는 레지스터
누산기 (AC; Accumulator)	연산된 결과를 일시적으로 저장하는 레지스터로 연산의 중심임
• 상태 레지스터(Status Register) • PSWR(Program Status Word Register) • 플래그 레지스터	• 시스템 내부의 순간순간의 상태가 기록된 정보를 PSW라고 함 • 오버플로, 언더플로, 자리올림, 계산 상태(0, -, +), 인터럽트 등의 PSW를 저장하고 있는 레지스터
메모리 주소 레지스터 (MAR; Memory Address Register)	읽기 동작이나 쓰기 동작을 수행할 때 필요한 주기억 장소의 주소를 저장하는 주소 저장용 레지스터
베이스 레지스터 (Base Register)	명령이 시작되는 최초의 번지, 즉 시작 주소를 기억하는 레지스터
메모리 버퍼 레지스터(MBR; Memory Buffer Register)	기억장치를 출입하는 데이터가 잠시 기억되는 레지스터로, CPU가 데이터를 처리하기 위해서는 반드시 거쳐야 함

인덱스 레지스터 (Index Register)	• 주소의 변경, 서브루틴 연결 및 프로그램에서의 반복 연산의 횟수를 세는 레지스터 • 명령어 실행 과정에서 명령어가 지정한 번지를 수정하기 위해 사용됨 • 사용자가 내용을 변경할 수 있음
데이터 레지스터 (Data Register)	연산에 사용될 데이터를 기억하는 레지스터
시프트 레지스터 (Shift Register)	• 저장된 값을 왼쪽 또는 오른쪽으로 1Bit씩 자리를 이동시키는 레지스터 • 2배 길이 레지스터라고도 함 • 데이터의 직렬 전송에 사용
메이저 스테이터스 레지스터 (Major Status Register)	CPU의 메이저 상태를 저장하고 있는 레지스터

핵심 074 버스

10.3, 08.3, 04.3, 00.3

- CPU, 메모리, I/O 장치 등과 상호 필요한 정보를 교환하기 위해 연결하는 공동의 전송선이다.
- 컴퓨터 내부 회로에서 버스를 사용하는 가장 큰 목적은 결선의 수를 줄이기 위해서이다.
- 버스의 종류

전송하는 정보에 따른 버스의 분류	• 주소 버스(Address Bus) : CPU가 메모리나 입·출력 기기의 번지를 지정할 때 사용하는 단방향 전송선 • 자료 버스(Data Bus) : CPU와 메모리 또는 입·출력 기기 사이에서 데이터를 전송하는 양방향 전송선 • 제어 버스(Control Bus) : CPU의 현재 상태나 상태 변경을 메모리 또는 입·출력에 알리는 제어신호를 전송하는데 사용하는 양방향 전송선
위치에 따른 버스의 분류	• 내부 버스 : CPU 및 메모리 내에 구성된 Bus • 외부 버스 : 주변 입·출력장치에 구성된 Bus

핵심 075 명령어의 구성

13.6, 13.3, 12.5, 11.8, 11.3, 10.5, 10.3, 08.9, 08.3, 07.9, 07.3, 05.9, 05.4, 05.3, 04.5, 03.8, 03.5, 03.3, 02.3

Operation Code, 연산자부	모드(Mode)부	Operand, 자료부

연산자부(Operation Code부)

- 수행해야 할 동작에 맞는 연산자를 표시하며, 흔히 OP-Code부라고 한다.
- 연산자부의 크기(비트 수)는 표현할 수 있는 명령의 종류를 나타내는 것으로, n Bit면 최대 2^n개의 명령어를 사용할 수 있다.
- 모드(Mode)부 : 주소부의 유효 주소가 결정되는 방법을 지정하며, 모드 비트가 0이면 직접, 1이면 간접임

자료부(Operand부), 주소부

- 실제 데이터에 대한 정보를 표시하는 부분이다.
- 기억장소의 주소, 레지스터 번호, 사용할 데이터 등을 표시한다.
- 주소부의 크기는 메모리의 용량과 관계가 있다.
- 자료부의 길이가 n Bit라면 최대 2^n개의 기억장소를 주소로 지정할 수 있다.

핵심 076 연산자(Operation Code)의 기능

04.3, 03.8, 03.5, 02.9, 01.9, 01.6, 01.3, 00.10
14.8, 14.3, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.3, 10.9, 10.5, 10.3, 09.8, 09.3, 06.9, 06.5, 06.3, 05.9, 05.4, 04.9, 04.5,

함수 연산 기능	산술 연산 : ADD, SUB, MUL, DIV, 산술 SHIFT 등 논리 연산 : NOT, AND, OR, XOR, 논리적 SHIFT, ROTATE, COMPLEMENT, CLEAR, SET등
자료 전달 기능	CPU와 기억장치 사이에서 정보를 교환하는 기능 • Load : 기억장치에 기억되어 있는 정보를 CPU로 꺼내오는 명령 • Store : CPU에 있는 정보를 기억장치에 기억시키는 명령 • Move : 레지스터 간에 자료를 전달하는 명령 • Push : 자료를 스택에 저장하는 명령 • Pop : 스택에서 자료를 꺼내오는 명령
제어 기능	명령어의 실행 순서를 변경시킬 때 사용하는 기능 • 무조건 분기 명령 : GOTO, Jump(JMP) 등 • 조건 분기(Branch) 명령 : IF 조건, SPA, SNA, SZA 등 • Call : 부 프로그램 호출 • Return : 부 프로그램에서 메인 프로그램으로 복귀
입·출력 기능	CPU와 I/O장치, 또는 메모리와 I/O 장치 사이에서 자료를 전달하는 기능 • INPUT : 입·출력장치의 자료를 주기억장치로 입력하는 명령 • OUTPUT : 주기억장치의 자료를 입·출력장치로 출력하는 명령

핵심 077 피연산자의 수에 따른 연산자의 분류

14.8, 14.3, 12.3, 08.5, 07.9, 06.9, 05.9, 05.5, 05.3, 04.9, 04.3, 00.5, 99.10, 99.6

NOT A처럼 피연산자가 1개만 필요한 연산자를 단항 연산자라 하고, A+B처럼 피연산자가 2개 필요한 연산자를 이항 연산자라 한다.

단항 연산자 (Unary Operator)	NOT, COMPLEMENT, SHIFT, ROTATE, MOVE, CLEAR 등
이항 연산자 (Binary Operator)	사칙 연산, AND, OR, XOR, XNOR

03.3, 01.9, 01.3, 00.7, 00.5, 00.3, 99.4
14.8, 14.5, 13.8, 13.6, 13.3, 12.8, 12.3, 11.6, 11.3, 10.5, 10.3, 09.5, 09.3, 08.5, 08.3, 07.3, 06.3, 05.9, 05.5, 05.3, 04.3,

핵심 078 연산

AND (Masking Operation)	• 특정 문자 또는 특정 Bit를 삭제(Clear)시키는 명령으로 Masking 명령이라고도 함 • 삭제할 부분의 Bit를 0과 AND시켜서 삭제하는데, 대응시키는 0인 Bit를 Mask Bit라고 함
OR (Selective Set)	• 특정 문자를 삽입하거나 특정 Bit에 1을 세트시키는 명령으로 Selective Set 연산이라고도 함 • 삽입하거나 세트시킬 Bit에 삽입할 문자 코드 또는 1을 OR 연산시킴
XOR (Compare, 비교)	• 2개의 데이터를 비교하거나 특정 비트를 반전시킬 때 사용함 • 2개의 데이터를 XOR 연산하여 결과에 1Bit라도 1이 있으면 서로 다른 데이터임 • 반전시킬 때는 반전시킬 비트와 1을 XOR 시킴
NOT (Complement, 보수)	각 비트의 값을 반전시키는 연산으로 보수를 구할 때 사용함
논리 Shift	• 왼쪽 또는 오른쪽으로 1Bit씩 자리를 이동시키는 연산으로 데이터의 직렬 전송(Serial Transfer)에 사용함 • 삽입되는 자리는 무조건 0임
Rotate	• Shift에서 밀려 나가는 비트의 값을 반대편 값으로 입력하는 연산 • 문자 위치를 변환할 때 이용
산술 Shift	• 부호(Sign)를 고려하여 자리를 이동시키는 연산으로, 2^n으로 곱하거나 나눌 때 사용함 • 왼쪽으로 n Bit Shift하면 원래 자료에 2^n을 곱한 값과 같음 • 오른쪽으로 n Bit Shift하면 원래 자료를 2^n으로 나눈 값과 같음 • 홀수를 오른쪽으로 한 번 Shift하면 0.5의 오차가 발생함

14.3, 07.5, 07.3, 05.3

핵심 079 연산자의 우선순위

계산식에 연산자가 혼합되어 나오면 우선순위는 '산술 〉 관계 〉 논리' 순이며 각각은 다음과 같다.

① 산술 연산자 : ^(거듭제곱) 〉 ×(곱셈), ÷(나눗셈) 〉 +, −

② 관계 연산자 : =, ≠, >, <, ≥, ≤ (우선순위가 모두 같음)

③ 논리 연산자 : NOT 〉 AND 〉 OR

※ 우선순위가 같은 연산자가 혼합되어 나오면 연산순서는 왼쪽에서 오른쪽으로 진행된다.

02.5, 02.3, 01.9, 01.6
14.8, 13.6, 12.8, 11.8, 10.9, 10.5, 10.3, 09.8, 09.5, 08.9, 08.3, 06.9, 05.9, 05.5, 05.3, 04.5, 04.3, 03.8, 03.5, 03.3, 02.9,

핵심 080 명령어 형식

3 번지 명령어	• Operand부가 3개로 구성되는 명령어 형식으로 여러 개의 범용 레지스터(GPR)를 가진 컴퓨터에서 사용함 • 연산의 결과는 Operand 3에 기록됨 • 연산 시 원시 자료를 파괴하지 않음 • 다른 형식의 명령어를 이용하는 것보다 프로그램 전체의 길이를 짧게 할 수 있음 • 전체 프로그램 실행 시 명령 인출을 위하여 주기억장치를 접근하는 횟수가 줄어들어 프로그램 실행 속도를 단축시킴
2 번지 명령어	• Operand부가 두 개로 구성되는, 가장 일반적으로 사용되는 명령어 형식임 • 여러 개의 범용 레지스터를 가진 컴퓨터에서 사용함 • 실행 속도가 빠르고 기억 장소를 많이 차지하지 않음 • 3주소 명령에 비해 명령어의 길이가 짧음 • 계산 결과가 기억장치에 기억되고 중앙처리장치에도 남아 있어서 계산 결과를 시험할 필요가 있을 때 시간이 절약됨 • 단점 – 연산의 결과는 주로 Operand1에 저장되므로 Operand 1에 있던 원래의 자료가 파괴됨 – 전체 프로그램의 길이가 길어짐
1 번지 명령어	• Operand부가 1개로 구성되어 있음 • AC(Accumulator; 누산기)를 이용하여 명령어를 처리함
0 번지 명령어	• Operand부 없이 OP-Code부만으로 구성됨 • 모든 연산은 Stack 메모리의 Stack Pointer가 가리키는 Operand를 이용하여 수행함 • 수식을 계산하기 위해서는 우선, 수식을 Postfix(역 Polish) 형태로 변경해야 함 • 모든 연산은 스택에 있는 자료를 이용하여 수행하기 때문에 스택 머신(Stack Machine)이라고도 함 • 원래의 자료가 남지 않음

04.9, 03.8, 03.5
14.5, 14.3, 13.8, 13.6, 12.8, 12.5, 11.8, 11.3, 10.5, 10.3, 09.8, 09.5, 08.5, 08.3, 07.9, 07.5, 07.3, 06.3, 05.9, 05.4, 05.3,

핵심 081 주소지정방식의 종류

암시적 주소 지정방식 (Implied Mode)	• 명령 실행에 필요한 데이터의 위치를 지정하지 않고 누산기나 스택의 데이터를 묵시적으로 지정하여 사용함 • 오퍼랜드가 없는 명령이나 'PUSH R1'처럼 오퍼랜드가 1개인 명령어 형식에 사용됨

즉치(즉시)적 주소지정방식 (Immediate Mode)	• 명령어 자체에 오퍼랜드(실제 데이터)를 내포하고 있는 방식 • 별도의 기억장소를 액세스하지 않고 CPU에서 곧바로 자료를 이용할 수 있어서 실행 속도가 빠르다는 장점이 있음 • 명령어의 길이에 영향을 받으므로 표현할 수 있는 데이터 값의 범위가 제한적임
직접 주소 지정방식 (Direct Mode)	• 명령의 주소부(Operand)가 사용할 자료의 번지를 표현하고 있는 방식 • 명령의 Operand부에 표현된 주소를 이용하여 실제 데이터가 기억된 기억장소에 직접 사상시킬 수 있음 • 주소 부분에 실제 사용할 데이터의 유효 주소를 적기 때문에 주소 길이에 제약을 받음 • 기억 용량이 2^n개의 Word인 메모리 시스템에서 주소를 표현하려면 n비트의 Operand부가 필요함 • 명령의 Operand부에 데이터를 가지고 있는 레지스터의 번호를 지정하면 레지스터 모드라고 함
간접 주소 지정방식 (Indirect Mode)	• 명령어에 나타낼 주소가 명령어 내에서 데이터를 지정하기 위해 할당된 비트(Operand 부의 비트) 수로 나타낼 수 없을 때 사용하는 방식 • 명령의 길이가 짧고 제한되어 있어도 긴 주소에 접근 가능함 • 명령어 내의 주소부에 실제 데이터가 저장된 장소의 번지를 가진 기억장소의 번지를 표현하므로, 최소한 주기억장치를 2번 이상 접근하여 데이터가 있는 기억장소에 도달함 • 명령의 Operand부에 데이터의 주소를 가지고 있는 레지스터의 번호를 지정하면 레지스터 간접 모드라고 함
계산에 의한 주소지정방식	• Operand부와 특정 레지스터의 값이 더해져서 유효주소를 계산하는 방식 • 상대(Relative) 주소지정방식 – 명령어의 주소 부분 + PC – 분기 명령에 많이 사용됨 • Base Register Mode : 명령어의 주소 부분 + Base Register • Index Register Mode : 명령어의 주소 부분 + Index Register ※ 계산에 의한 주소지정방식은 전체 기억장치의 주소를 사용해야 하는 일반적인 주소지정방식에 비해 적은 수의 비트를 사용하고, 레지스터 지정필드 없이 묵시적으로 레지스터를 지정하여 사용하기 때문에 데이터의 주소를 분류할 때 약식 주소라고 함

핵심

082 마이크로 오퍼레이션(Micro Operation)의 정의

14.8, 14.3, 13.8, 12.8, 11.3, 10.5, 08.9, 07.5, 06.3, 05.9, 05.5, 05.3, 04.9, 04.3, 02.9, 01.9, 00.10, 99.6

- Instruction을 수행하기 위해 CPU 내의 레지스터와 플래그가 의미 있는 상태 변환을 하도록 하는 동작이다.
- 기억장치로부터 인출하여 해독된 명령어를 실행하기 위해 제어 신호를 발생시키는 각 단계의 세부 동작을 말한다.
- 레지스터에 저장된 데이터에 의해 이루어지는 동작이다.
- 한 개의 Clock 펄스 동안 실행되는 기본 동작이다.
- 마이크로 오퍼레이션의 순서를 결정하기 위하여 제어장치가 발생하는 신호를 제어 신호라고 한다.
- 마이크로 오퍼레이션은 Instruction 실행과정에서 한 단계씩 이루어지는 동작으로, 한 개의 Instruction은 여러 개의 Micro Operation이 동작되어 실행된다.
- Micro Cycle Time
  – 한 개의 Micro Operation을 수행하는 데 걸리는 시간
  – CPU Cycle Time 또는 CPU Clock Time이라고도 하며, CPU 속도를 나타내는 척도로 이용됨

제어 워드와 마이크로 프로그램

- **제어 워드** : 레지스터의 선택과 산술 논리 연산장치의 역할을 결정하고, 어떤 마이크로 연산을 할 것인가를 결정하는 비트의 모임을 제어 워드라고 함. 제어 워드는 마이크로 명령어라고도 함
- **마이크로 프로그램** : 어떤 명령을 수행할 수 있도록 구성된 일련의 제어 워드가 특수한 기억장치 속에 저장될 때 이를 마이크로 프로그램이라고 함

핵심

083 Micro Cycle Time 부여 방식

14.8, 14.5, 13.6, 13.3, 12.5, 11.8, 11.3, 10.9, 10.3, 09.5, 08.5, 06.5, 05.4, 00.3

- 한 개의 Micro Operation을 수행하는 데 걸리는 시간을 Micro Cycle Time이라 한다.
- Micro Cycle Time은 CPU Cycle Time 또는 CPU Clock Time이라고도 하며, CPU 속도를 나타내는 척도로 이용한다.

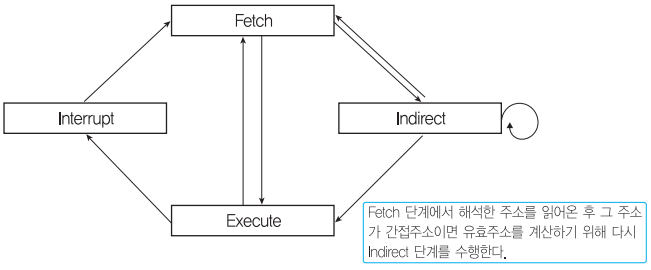
동기 고정식 (Synchronous Fixed)	• 모든 마이크로 오퍼레이션의 동작 시간이 같다고 가정하여 CPU Clock의 주기를 Micro Cycle Time과 같도록 정의하는 방식 • 모든 마이크로 오퍼레이션 중에서 동작 시간이 가장 긴 마이크로 오퍼레이션의 동작 시간을 Micro Cycle Time으로 정함 • 모든 마이크로 오퍼레이션의 동작 시간이 비슷할 때 유리한 방식임 • **장점** : 제어기의 구현이 단순함 • **단점** : CPU의 시간 낭비가 심함
동기 가변식 (Synchronous Variable)	• 동작 시간이 유사한 Micro Operation들끼리 그룹을 만들어, 각 그룹별로 서로 다른 Micro Cycle Time을 정의하는 방식 • 동기 고정식에 비해 CPU 시간 낭비를 줄일 수 있는 반면, 제어기의 구현은 조금 복잡함 • 마이크로 오퍼레이션의 동작 시간이 현저하게 차이 날 때 유리함(정수배)

비동기식 (Asynchron-ous)	• 모든 마이크로 오퍼레이션에 대하여 서로 다른 Micro Cycle Time을 정의하는 방식 • CPU의 시간 낭비는 전혀 없으나 제어기가 매우 복잡해지기 때문에 실제로는 거의 사용되지 않음

핵심 084 메이저 스테이트

14.8, 07.3, 06.9, 05.5, 04.9, 04.5, 03.5, 03.3, 00.5, 99.4

- 현재 CPU가 무엇을 하고 있는가를 나타내는 상태로서 Fetch, Indirect, Execute, Interrupt 이렇게 4개의 상태가 있다.
- CPU는 메이저 스테이트의 4가지 단계를 반복적으로 거치면서 동작을 수행한다.
- 메이저 스테이트는 메이저 스테이트 레지스터를 통해서 알 수 있다.
- Major Cycle 또는 Machine Cycle이라고도 한다.
- 메이저 스테이트의 변천 과정



핵심 085 인출 단계(Fetch Cycle)

14.3, 13.6, 11.3, 10.5, 09.5, 08.3, 07.9, 07.3, 07.5, 06.9, 05.9, 04.3, 03.3, 02.9, 02.3, 01.9, 01.3, 99.8, 99.6

- 명령어를 주기억장치에서 중앙처리장치의 명령 레지스터로 가져와 해독하는 단계이다.
- 읽어와 해석된 명령어가 1Cycle 명령이면 이를 수행한 후 다시 Fetch Cycle로 변천한다.
- 1Cycle 명령이 아니면, 해석된 명령어의 모드 비트에 따라 직접주소와 간접주소를 판단한다.
- 동작 순서

Micro Operation	의미
MAR ← PC	PC에 있는 번지를 MAR에 전송시킴
MBR ← M[MAR], PC ← PC + 1	• 메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송함 • 다음에 실행할 명령의 위치를 지정하기 위해 PC의 값을 1 증가시킴
IR ← MBR[OP], I ← MBR[I]	• 명령어의 OP-Code 부분을 명령 레지스터에 전송함 ※현재 MBR에는 주기억장치에서 읽어온 명령이 들어 있음 • 명령어의 모드 비트를 플립플롭 I에 전송함
F ← 1 또는 R ← 1	I가 0이면 F 플립플롭에 1을 전송하여 Execute 단계로 변천하고, I가 1이면 R 플립플롭에 1을 전송하여 Indirect 단계로 변천함

핵심 086 간접 단계(Indirect Cycle)

10.3, 09.8, 05.4, 99.10, 99.4

- Fetch 단계에서 해석된 명령의 주소부가 간접주소인 경우 수행된다.
- Fetch 단계에서 해석한 주소를 읽어온 후 그 주소가 간접주소이면 유효주소를 계산하기 위해 다시 Indirect 단계를 수행한다.
- 간접주소가 아닌 경우에는 명령어에 따라 Execute 단계 또는 Fetch 단계로 이동할지를 판단한다.
- 동작 순서

Micro Operation	의미
MAR ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 MAR에 전송함
MBR ← M[MAR]	메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송함
No Operation	동작 없음
F ← 1, R ← 0	F에 1, R에 0을 전송하여 Execute 단계로 변천함

핵심 087 인터럽트 단계(Interrupt Cycle)

13.8, 12.3, 09.3, 08.3, 05.5, 03.8, 00.7

- 인터럽트 발생 시 복귀주소(PC)를 저장시키고, 제어 순서를 인터럽트 처리 프로그램의 첫 번째 명령으로 옮기는 단계이다.
- 인터럽트 단계를 마친 후에는 항상 Fetch 단계로 변천한다.
- Interrupt가 발생할 때만 실행되어 다른 일을 처리하고 돌아오기 때문에 하드웨어로 실현되는 서브루틴(부 프로그램)의 호출이라고도 한다.

• 동작 순서

Micro Operation	의미
MBR[AD] ← PC, PC ← 0	• PC가 가지고 있는, 다음에 실행할 명령의 주소를 MBR의 주소 부분으로 전송함 • 복귀주소를 저장할 0번지를 PC에 전송함
MAR ← PC, PC ← PC + 1	• PC가 가지고 있는 값 0번지를 MAR에 전송함 • 인터럽트 처리 루틴으로 이동할 수 있는 인터럽트 벡터의 위치를 지정하기 위해 PC의 값을 1 증가시켜 1로 세트시킴
M[MAR] ← MBR, IEN ← 0	• MBR이 가지고 있는, 다음에 실행할 명령의 주소를 메모리의 MAR이 가리키는 위치(0 번지)에 저장함 • 인터럽트 단계가 끝날 때까지 다른 인터럽트가 발생하지 않게 IEN에 0을 전송함
F ← 0, R ← 0	F에 0, R에 0을 전송하여 Fetch 단계로 변천함

핵심 088 주요 명령의 마이크로 오퍼레이션

13.8, 11.8, 06.3, 05.5, 03.8, 02.3, 99.8

ADD : AC ← AC + M[AD]

Micro Operation	의미
MAR ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 MAR에 전송함
MBR ← M[MAR]	메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송함
AC ← AC + MBR	누산기의 값과 MBR의 값을 더해 누산기에 전송함

LDA(Load to AC) : AC ← M[AD]

Micro Operation	의미
MAR ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 MAR에 전송함
MBR ← M[MAR] AC ← 0	• 메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송함 • AC에 0을 전송하여 AC를 초기화함
AC ← AC + MBR	• 메모리에서 가져온 MBR과 AC를 더해 AC에 전송함 ※ 초기화된 AC에 더해지므로 메모리의 값을 AC로 불러오는 것이 됨

STA(Store AC) : M[AD] ← AC

Micro Operation	의미
MAR ← MBR[AD]	MBR에 있는 명령어의 번지 부분을 MAR에 전송함
MBR ← AC	AC의 값을 MBR에 전송함
M(MAR) ← MBR	MBR의 값을 메모리의 MAR이 지정하는 위치에 전송함

핵심 089 제어장치의 비교

11.6, 10.5, 10.3, 08.3, 02.5, 00.7, 99.10

제어장치는 필요한 마이크로 연산들이 연속적으로 수행할 수 있도록 제어 신호를 보내는 역할을 한다.

구분	고정배선 제어장치	마이크로 프로그래밍 기법
반응 속도	고속	저속
회로 복잡도	복잡	간단
경제성	비경제적	경제적
융통성	없음	있음
구성	하드웨어	소프트웨어

핵심 090 입 · 출력장치의 구성

14.8, 14.5, 04.5, 03.8, 02.3, 01.6, 00.10

입 · 출력 제어장치

• 입 · 출력장치와 컴퓨터 사이의 자료 전송을 제어하는 장치로, 데이터 버퍼 레지스터를 이용하여 두 장치 간의 속도 차를 조절한다.

• DMA, 채널, 입 · 출력 프로세서, 입 · 출력 컴퓨터 등이 입 · 출력 제어장치에 해당된다.

입 · 출력 인터페이스 : 동작 방식이나 데이터 형식이 서로 다른 컴퓨터 내부의 주기억장치나 CPU의 레지스터와 외부 입 · 출력장치 간의 2진 정보를 원활하게 전송하기 위한 방법을 제공함

입 · 출력 버스

• 주기억장치와 입 · 출력장치 사이의 데이터 전송을 위해 모든 주변장치의 인터페이스에 공통으로 연결된 버스이다.

• 입 · 출력 버스는 데이터 버스, 주소 버스, 제어 버스로 구성된다.

입 · 출력장치의 종류

입력장치	키보드, 마우스, 스캐너, OMR, OCR, MICR, BCR(Bar Code Reader), 마이크로 필름 입력장치(CIM: Computer Input Microfilm), 라이트 펜, 터치스크린, 디지타이저 등

출력장치	모니터, 프린터, 플로터, 마이크로 필름 출력장치(COM; Computer Output Microfilm) 등
보조기억장치 (입·출력 겸용 장치)	자기 디스크, 자기 테이프, 자기 드럼, 하드디스크, 플로피디스크 등

핵심 091 기억장치와 입·출력장치의 동작 차이

14.3, 12.8, 11.6, 11.3, 10.5, 09.3, 03.5, 03.3, 00.3

기억장치는 처리 속도가 nano(10^{-9}) 단위인 전자적인 장치이고, 입·출력장치는 milli(10^{-3})의 단위인 기계적인 장치이므로 동작 방식에는 많은 차이가 있다.

비교 항목	입·출력장치	기억장치
동작의 속도	느리다	빠르다
동작의 자율성	타율/자율	타율
정보의 단위	Byte(문자)	Word
착오 발생률	많다	적다

핵심 092 스풀링(SPOOLING)

12.3, 09.5, 08.5, 05.3, 03.3, 02.3, 00.10, 99.10, 99.8, 99.6

- Simultaneous Peripheral Operation On-Line의 약자이다.
- 다중 프로그래밍 환경하에서 용량이 크고 신속한 액세스가 가능한 디스크를 이용하여 각 사용자 프로그램이 입·출력할 데이터를 직접 I/O 장치로 보내지 않고 디스크에 모았다가 나중에 한꺼번에 입·출력함으로써 입·출력장치의 공유 및 상대적으로 느린 입·출력장치의 처리 속도를 보완하는 기법이다.
- 스풀링은 고속의 CPU와 저속의 입·출력장치가 동시에 독립적으로 동작하게 하여 높은 효율로 여러 작업을 병행 작업할 수 있도록 해줌으로써 다중 프로그래밍 시스템의 성능 향상을 가져올 수 있다.
- 스풀링은 디스크 일부를 매우 큰 버퍼처럼 사용한다.

스풀링과 버퍼링의 비교

버퍼링도 입·출력장치와 CPU 간의 속도 차이를 해결하기 위해 사용하는 목적은 같지만 다음과 같은 점이 스풀링과 다르다.

구분	버퍼링	스풀링
저장 위치	주기억장치	보조기억장치
운영 방식	단일 작업	다중 작업
구현 방식	하드웨어	소프트웨어

핵심 093 입·출력(Input-Output) 제어 방식

14.8, 12.8, 12.5, 10.9, 10.5, 09.8, 09.5, 09.3, 08.9, 07.5, 07.3, 05.9, 05.4, 05.3, 03.5, 03.3, 01.6, 00.7, 00.3,99.10, 99.4

Programmed I/O(폴링)	• 원하는 I/O가 완료되었는지의 여부를 검사하기 위해서 CPU가 상태 Flag를 계속 조사하여 I/O가 완료되었으면 MDR(MBR)과 AC 사이의 자료 전송을 CPU가 직접 처리하는 I/O 방식 • 입·출력에 필요한 대부분의 일을 CPU가 해주므로 인터페이스는 MDR, Flag, 장치번호 디코더로만 구성하면 됨 • I/O 작업 시 CPU는 계속 I/O 작업에 관여해야 하기 때문에 다른 작업을 할 수 없다는 단점이 있음
Interrupt I/O	• 입·출력을 하기 위해 CPU가 계속 Flag를 검사하지 않고, 데이터를 전송할 준비가 되면 입·출력 인터페이스가 컴퓨터에게 알려 입·출력이 이루어지는 방식 • 입·출력 인터페이스는 CPU에게 인터럽트 신호를 보내 입·출력이 있음을 알림 • CPU가 계속 Flag를 검사하지 않아도 되기 때문에 Programmed I/O보다 효율적임
DMA (Direct Memory Access)에 의한 I/O	• 입·출력장치가 직접 주기억장치를 접근(Access)하여 Data Block을 입·출력하는 방식으로 입·출력 전송이 CPU의 레지스터를 경유하지 않고 수행됨 • CPU는 I/O에 필요한 정보를 DMA 제어기에 알려서 I/O 동작을 개시시킨 후 I/O 동작에 더 이상 간섭하지 않고 다른 프로그램을 할당하여 수행함 • 입·출력 자료 전송 시 CPU를 거치지 않기 때문에 CPU의 부담 없이 보다 빠른 데이터의 전송이 가능함 • 인터럽트 신호를 발생시켜 CPU에게 입·출력 종료를 알림 • Cycle Steal 방식을 이용하여 데이터를 전송함 • CPU에서 DMA 제어기로 보내는 자료 – DMA를 시작시키는 명령 – 입·출력 하고자 하는 자료의 양 – 입력 또는 출력을 결정하는 명령 • DMA의 구성 요소 – 인터페이스 회로 : CPU와 입·출력 장치와의 통신 담당 – 주소 레지스터 : 기억장치의 위치 지정을 위한 번지 기억 – 워드 카운트 레지스터 : 전송되어야 할 워드의 수를 표시함 – 제어 레지스터 : 전송 방식 결정 – 데이터 버스 버퍼, 주소 버스 버퍼 : 전송에 사용할 자료나 주소를 임시로 기억함

Channel에 의한 I/O	• I/O를 위한 특별한 명령어를 I/O 프로세서에게 수행토록 하여 CPU 관여 없이 주기억장치와 입·출력장치 사이에서 입·출력을 제어하는 입·출력 전용 프로세서(IOP) • DMA 방법으로 입·출력을 수행하므로 DMA의 확장된 개념으로 볼 수 있음 • DMA 제어기의 한계를 극복하기 위하여 고안된 방식임 • 채널 제어기는 채널 명령어로 작성된 채널 프로그램을 해독하고 실행하여 입·출력 동작을 처리함 • CPU로부터 입·출력 전송을 위한 명령어를 받으면 CPU와는 독립적으로 동작하여 입·출력을 완료함 • 주기억장치에 기억되어 있는 채널 프로그램의 수행과 자료의 전송을 위하여 주기억장치에 직접 접근함 • I/O 장치는 제어장치를 통해 채널과 연결됨 • I/O 채널은 CPU의 I/O 명령을 수행하지 않고 I/O 채널 내의 특수목적 명령을 수행함 • CPU와 인터럽트로 통신함 • **채널의 종류** – Selector Channel : 고속 입·출력장치(자기 디스크, 자기 테이프, 자기 드럼) 1개와 입·출력하기 위해 사용함 – Multiplexer Channel : 저속 입·출력장치(카드리더, 프린터) 여러 개를 동시에 제어하는 채널 – Block Multiplexer Channel : 동시에 여러 개의 고속 입·출력장치를 제어함

잠깐만요 ! Cycle Steal

- 데이터 채널(DMA 제어기)과 CPU가 주기억장치를 동시에 Access할 때 우선순위를 데이터 채널에게 주는 방식입니다.
- Cycle Steal은 한 번에 한 데이터 워드를 전송하고 버스의 제어를 CPU에게 돌려줍니다.
- Cycle Steal을 이용하면 입·출력 자료의 전송을 빠르게 처리해 주는 장점이 있습니다.
- Cycle Steal 시 중앙처리장치는 메모리 참조가 필요 없는 오퍼레이션을 계속 수행합니다.

핵심 **094** 14.3, 10.3, 09.3, 08.9, 08.5, 07.3,06.5, 05.5, 05.4, 05.3, 04.9, 03.8, 03.3, 02.3, 01.6, 00.7, 99.6

인터럽트의 종류 및 발생 원인

- 프로그램을 실행하는 도중에 예기치 않은 상황이 발생할 경우 현재 실행중인 작업을 즉시 중단하고, 발생된 상황을 우선 처리한 후 실행중이던 작업으로 복귀하여 계속 처리하는 것, 일명 "끼어들기"라고도 한다.
- 인터럽트 서비스 루틴을 실행할 때, 인터럽트 플래그(IF)를 0으로 하면 인터럽트 발생을 방지할 수 있다.
- 인터럽트는 외부 인터럽트, 내부 인터럽트, 소프트웨어 인터럽트로 분류하는데, 외부나 내부 인터럽트는 CPU의 하드웨어에서의 신호에 의해 발생하고 소프트웨어 인터럽트는 명령어의 수행에 의해 발생한다.

외부 인터럽트	• 전원 이상 인터럽트(Power Fail Interrupt) : 정전이 되거나 전원 이상이 있는 경우 • 기계 착오 인터럽트(Machine Check Interrupt) : CPU의 기능적인 오류 동작이 발생한 경우 • 외부 신호 인터럽트(External Interrupt) – 타이머에 의해 규정된 시간(Time Slice)을 알리는 경우 – 키보드로 인터럽트 키를 누른 경우 – 외부장치로부터 인터럽트 요청이 있는 경우 • 입·출력 인터럽트(Input-Output Interrupt) – 입·출력 Data의 오류나 이상 현상이 발생한 경우 – 입·출력장치가 데이터의 전송을 요구하거나 전송이 끝났음을 알릴 경우
내부 인터럽트	• 잘못된 명령이나 데이터를 사용할 때 발생하며, 트랩(Trap)이라고도 부름 • 프로그램 검사 인터럽트(Program Check Interrupt) – 0으로 나누기(Divide by zero)가 발생한 경우 – Overflow 또는 Underflow가 발생한 경우 – 프로그램에서 명령어를 잘못 사용한 경우 – 부당한 기억장소의 참조와 같은 프로그램의 오류
소프트웨어 인터럽트	• 프로그램 처리 중 명령의 요청에 의해 발생하는 것으로, 가장 대표적인 형태는 감시 프로그램을 호출하는 SVC(SuperVisor Call) 인터럽트가 있음 • SVC(SuperVisor Call) 인터럽트 – 사용자가 SVC 명령을 써서 의도적으로 호출한 경우 – 복잡한 입·출력 처리를 해야 하는 경우 – 기억장치 할당 및 오퍼레이터와 대화를 해야 하는 경우

핵심 **095** 12.8, 05.4, 04.3, 01.9, 01.3

인터럽트 발생 시 CPU가 확인할 사항

- 프로그램 카운터의 내용
- 사용한 모든 레지스터의 내용
- 상태 조건의 내용(PSW)

핵심 **096** 12.5, 12.3, 11.6, 09.8, 09.5, 08.3, 07.9, 06.9, 04.5, 04.3, 02.9, 02.5, 01.9, 01.6, 01.3, 00.5, 00.3, 99.8

인터럽트의 동작 순서

❶ 인터럽트 요청 신호 발생

❷ **프로그램 실행을 중단함** : 현재 실행중이던 명령어(Micro Instruction)는 끝까지 실행함

❸ **현재의 프로그램 상태를 보존함** : 프로그램 상태는 다음에 실행할 명령의 번지로서 PC가 가지고 있음

❹ **인터럽트 처리 루틴을 실행함** : 인터럽트를 요청한 장치를 식별함

❺ 인터럽트 서비스(취급) 루틴을 실행함 : 실질적인 인터럽트를 처리함

❻ 상태 복구 : 인터럽트 요청 신호가 발생했을 때 보관한 PC의 값을 다시 PC에 저장함

❼ 중단된 프로그램 실행 재개 : PC의 값을 이용하여 인터럽트 발생 이전에 수행중이던 프로그램을 계속 실행함

잠깐만요!

프로그램의 상태 보존
프로그램의 상태 보존이 필요한 이유는 인터럽트 서비스를 완료하고 원래 수행 중이던 프로그램으로 복귀하기 위해서입니다.

인터럽트 발생 시 PC의 값 보관 방법
인터럽트 발생 시 상태를 보존하기 위한 PC의 값은 메모리의 0번지, 스택, 인터럽트 벡터 중의 한 곳에 저장합니다.

인터럽트 반응 시간(Interrupt Response Time)
인터럽트 요청 신호를 발생한 후부터 인터럽트 취급 루틴의 수행이 시작될 때까지의 시간

인터럽트 벡터
- 중앙처리장치는 인터럽트가 발생한 장치번호를 받은 후에는 해당되는 인터럽트 서비스(취급) 루틴으로 분기하게 됩니다.
- 이때 기억장치 내의 특정한 곳에는 인터럽트 취급 루틴으로 분기하는 명령어들만을 기억하는 영역이 있는데, 이를 인터럽트 벡터라고 합니다.
- 인터럽트 벡터에는 인터럽트가 발생했을 때 프로세서의 인터럽트 서비스가 특정의 장소로 점프하도록 점프할 분기번지가 기억되어 있습니다.

핵심 097 인터럽트 우선순위

14.8, 10.9, 09.8, 03.3, 02.9, 02.5, 00.10

- 목적 : 여러 장치에서 동시에 인터럽트가 발생하였을 때 먼저 서비스할 장치를 결정하기 위해서임
- 기능
  - 각 장치에 우선순위를 부과하는 기능
  - 인터럽트를 요청한 장치의 우선순위를 판별하는 기능
  - 우선순위가 높은 것을 먼저 처리할 수 있는 기능
- 우선순위(높음 〉 낮음) : 전원 이상(Power Fail) 〉 기계 착오(Machine Check) 〉 외부 신호(External Signal) 〉 입·출력(I/O) 〉 명령어 잘못 〉 프로그램 검사(Program Check) 〉 SVC(SuperVisor Call)

잠깐만요! 마스크 불가 인터럽트(NMI; Non-Maskable Interrupt)
심각한 기억장치 오류나 정전 사태와 같은 급박한 상황에서 발생하는 것으로, 다른 어떤 서비스 요구에도 방해를 받지 않고 CPU에 전달됩니다.

핵심 098 인터럽트 우선순위 판별 방법

03.8, 03.3, 02.5, 01.6, 01.3, 00.3
14.5, 14.3, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 10.5, 10.3, 07.9, 07.3, 06.9, 06.5, 06.3, 05.9, 05.5, 05.4, 04.9, 04.5, 04.3,

소프트웨어적인 방법 : Polling	• Interrupt 발생 시 가장 높은 우선순위의 인터럽트 자원(Source)부터 차례로 검사해서, 우선순위가 가장 높은 Interrupt 자원(Source)를 찾아내어 이에 해당하는 인터럽트 서비스 루틴을 수행하는 방식 • 소프트웨어적인 방식을 폴링이라고 함 • 우선순위 변경이 쉬우며, 자기디스크와 같이 속도가 빠른 장치에 높은 등급을 부여함 • 많은 인터럽트가 있을 경우 그들을 모두 조사하는데 많은 시간이 걸려 반응 시간이 느리다는 단점이 있음 • 회로가 간단하고 융통성이 있으며, 별도의 하드웨어가 필요 없으므로 경제적임
하드웨어적인 방법 : Vectored Interrupt	• CPU와 Interrupt를 요청할 수 있는 장치 사이에 장치 번호에 해당하는 버스를 병렬이나 직렬로 연결하여 요청 장치의 번호를 CPU에 알리는 방식 • 벡터 인터럽트 방식에서는 인터럽트를 발생한 장치가 프로세서에게 분기할 곳에 대한 정보를 제공하는데, 이 정보를 인터럽트 벡터라 함 • 벡터 인터럽트를 하드웨어 신호에 의하여 수행되는 서브루틴이라고도 함 • 장치 판별을 위한 별도의 프로그램 루틴이 없어 응답 속도가 빠름 • 회로가 복잡하고 융통성이 없으며 추가적인 하드웨어가 필요하므로 비경제적임 • **직렬(Serial) 우선순위 부여 방식** : 데이지 체인(Daisy-Chain) 방식 – 인터럽트가 발생하는 모든 장치를 한 개의 회선에 직렬로 연결함 – 우선순위가 높은 장치를 선두에 위치시키고 나머지를 우선순위에 따라 차례로 연결함 • **병렬(Parallel) 우선순위 부여 방식** – 인터럽트가 발생하는 각 장치를 개별적인 회선으로 연결함 – 우선순위는 Mask Register의 비트 위치에 의해서 결정됨 – 마스크 레지스터는 우선순위가 높은 것이 서비스 받고 있을 때 우선순위가 낮은 것을 비활성화 시킬 수 있음 – 우선순위가 높은 Interrupt는 낮은 Interrupt가 처리되는 중에도 우선 처리됨

핵심 099 기억장치의 분류

10.3, 10.3, 06.5, 04.9, 03.5, 02.9, 99.6

주기억장치 (Main Storage)	RAM	• SRAM • DRAM
	ROM	• Mask ROM • PROM • EPROM • EEPROM
보조기억장치	• 자기 디스크 • 하드디스크 • 자기 테이프	• 자기 드럼 • 플로피디스크
특수 기억장치	• 복수 모듈 기억장치 • 캐시 기억장치	• 연관 기억장치 • 가상 기억장치

핵심 100 기억장치의 특성을 결정하는 요소

12.5, 11.8, 11.3, 05.9, 04.9, 01.3, 99.6

기억 용량	기억장치는 무조건 기억 용량이 큰 것을 사용한다고 해서 좋은 것이 아니라, 사용 목적에 따라 성능당 경비 비율이 적은 것을 사용하는 것이 바람직함
Access Time	• 기억장치에 읽기 요청이 발생한 시간부터 요구한 정보를 꺼내서 사용 가능할 때까지의 시간 • 한 Word 단위의 정보를 읽거나 기록하는 데 걸리는 시간 • Access Time = Seek(탐색) Time + Latency(대기) Time(또는 Search Time) + Transmission(전송) Time
Cycle Time	• 기억장치에 읽기 신호를 보낸 후 다시 읽기 신호를 보낼 수 있을 때까지의 시간 간격 • Cycle Time ≥ Access Time
Bandwidth (대역폭, 전송률)	• 메모리로부터 또는 메모리까지 1초 동안 전송되는 최대한의 정보량으로 기억장치의 자료 처리 속도를 나타내는 단위 • 하드웨어의 특성상 주기억장치가 제공할 수 있는 정보전달 능력의 한계를 의미함 • 메모리 워드의 길이가 작을수록 대역폭이 좋음

핵심 101 ROM(Read Only Memory)

14.8, 14.5, 13.6, 11.8, 10.5, 10.3, 09.3, 08.5, 08.3, 07.3, 06.9, 06.5, 03.8, 03.3, 02.9, 00.10, 99.10, 99.8

- 기억된 내용을 읽을 수만 있는 기억장치로서 일반적으로 쓰기는 불가능하다.
- 전원이 꺼져도 기억된 내용이 지워지지 않는 비휘발성 메모리이다.
- 실제로 ROM은 주기억장치로 사용하기보다는 주로 기본 입·출력 시스템(BIOS), 자가 진단 프로그램(POST) 같은 변경 가능성이 희박한 시스템 소프트웨어를 기억시키는 데 이용한다.
- ROM의 종류와 특징

종류	특징
Mask ROM	제조 공장에서 프로그램화하여 생산한 ROM으로, 사용자가 내용을 변경시킬 수 없음
PROM(Programmable ROM)	PROM 프로그램 장치라는 특수 장비를 이용하여 비어 있는 ROM에 사용자가 한 번만 내용을 기입할 수 있으며, 이후엔 읽기만 가능함
EPROM (Erasable PROM)	• 자외선을 쏘여서 기록한 내용을 지울 수 있고, PROM 프로그램 장치로 기록할 수도 있음 • 사용자가 여러 번 반복해서 지우거나 기록할 수 있음 • UVEPROM이라고도 함
EAROM(Erasable Alterable ROM)	전기적 특성을 이용하여 기록된 정보의 일부를 바꿀 수 있는 ROM
EEPROM(Electronic EPROM)	전기적인 방법을 이용하여 기록된 내용을 여러 번 수정하거나 새로운 내용을 기록할 수 있는 ROM

잠깐만요 ! 플래시 메모리

- EEPROM의 일종으로 전기적으로 데이터를 지우고 다시 기록할 수 있습니다.
- 전체가 아닌 일부 블록에 대해서 지우고 쓸 수 있다는 점이 EEPROM과 다릅니다.
- MP3 플레이어, 휴대전화, 디지털 카메라, PC 등에 널리 사용됩니다.
- 플래시 메모리를 이용한 디스크 드라이브를 하이브리드 드라이브라고 합니다.

핵심 102 RAM(Random Access Memory)

14.8, 11.8, 11.6, 09.5, 07.5, 05.5, 02.5, 01.9, 00.7

- 자유롭게 읽고 쓸 수 있는 기억장치로, RWM(Read Write Memory)이라고도 한다.
- RAM에는 현재 사용중인 프로그램이나 데이터가 저장되어 있다.
- 전원이 꺼지면 기억된 내용이 모두 사라지는 휘발성 메모리이다.
- 일반적으로 '주기억장치' 또는 '메모리'라고 하면 램을 의미한다.
- 정보가 저장된 위치는 주소로 구분한다.

• DRAM/SRAM의 특징

구분	동적 램(DRAM)	정적 램(SRAM)
구성 소자	콘덴서	플립플롭
특징	전원이 공급되어도 일정 시간이 지나면 전하가 방전되므로 주기적인 재충전(Refresh)이 필요함	전원이 공급되는 동안에는 기억 내용이 유지됨
전력 소모	적음	많음
접근 속도	느림	빠름
집적도(밀도)	높음	낮음
가격	저가	고가
용도	일반적인 주기억장치	캐시 메모리

03.5, 02.9, 01.9, 00.10, 99.6

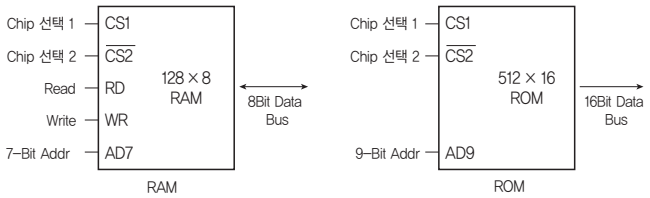
핵심

103 반도체 기억소자의 구성

14.8, 14.3, 13.6, 12.8, 12.5, 12.3, 11.8, 11.6, 11.3, 10.5, 09.5, 08.5, 07.9, 07.3, 06.9, 06.5, 05.5, 05.3, 04.9, 04.5, 03.8,

RAM/ROM의 구성

RAM은 칩 선택선, 읽기 쓰기 선택선, 주소선, 양방향 데이터 버스가 있지만 ROM은 읽기만 할 수 있도록 정해져 있으므로 읽기 쓰기 선택선이 없고, 데이터 버스도 단방향이다.



- CS1, CS2 : 칩 선택선
- RD : 입력 신호선
- WR : 출력 신호선
- AD : 주소선, 주소선의 수는 지정할 수 있는 워드의 수와 관계 있다. 주소선이 n개라면 2^n개의 워드를 지정할 수 있음
- Data Bus : 워드의 크기를 나타냄. 데이터 버스가 8Bit라면 워드의 크기가 8Bit임

RAM/ROM의 용량 계산

- 위 그림에서 RAM은 주소선이 7개이고, Data Bus가 8Bit이므로 128(2^7)×8Bit의 용량이고, ROM은 주소선이 9개이고, Data Bus가 16Bit이므로 512(2^9)×16Bit의 용량이다.
- 주소선의 수는 주소를 지정하는 MAR, 그리고 다음 실행할 명령의 주소를 가지고 있는 PC의 크기와 같고, Data Bus의 비트 수는 읽어온 또는 저장할 자료를 잠시 보관하는 MBR, 그리고 읽어온 명령어를 저장하는 IR의 크기와 같다.
- 주소선의 수 = MAR = PC
- Data Bus의 비트 수 = MBR = IR

13.3, 12.8, 12.3, 11.3, 10.9, 09.8, 08.9, 08.3, 07.3, 06.5, 05.4, 04.9, 03.3, 02.3, 00.5, 99.6

핵심

104 보조기억장치

보조기억장치의 특징

- 주기억장치에 비해 속도는 느리지만 저장 용량이 크다.
- 전원이 차단되어도 내용이 그대로 유지된다.
- 중앙처리장치와 직접 자료 교환이 불가능하다.
- 일반적으로 주기억장치에 데이터를 저장할 때는 DMA 방식을 사용한다.

보조기억장치의 종류

자기 테이프 (Magnetic Tape)	• 순차 처리(SASD)만 할 수 있는 대용량 저장 매체 • 가격이 저렴하고 용량이 커서 자료의 백업용으로 많이 사용함 • 자성 물질이 코팅된 얇은 플라스틱 테이프를 동그란 릴에 감아 놓은 형태 • 테이프의 시작과 끝 부분을 알리는 은박지 사이의 정보 저장 부분을 7~9트랙으로 구성함 • 트랙별로 1비트를 저장하므로 동시에 7~9비트를 읽거나 쓸 수 있다. • 블록 단위로 데이터를 전송함
자기 디스크 (Magnetic Disk)	• 자성 물질을 입힌 금속 원판을 여러 장 겹쳐서 만든 기억 매체로 용량이 크고 접근 속도가 빠름 • 순차, 비순차(직접) 처리가 모두 가능한 DASD (Direct Access Storage Device) 방식으로 데이터를 처리함 • 트랙(Track) : 디스크 표면에서 회전축(스핀들 모터)을 중심으로 데이터가 기록되는 동심원 • 섹터(Sector) : Track들을 일정한 크기로 구분한 부분이며, 정보 기록의 기본 단위임 • 실린더(Cylinder) : 서로 다른 면들에 있는 동일 위치의 Track들의 모임으로, 실린더의 수는 한 면의 트랙 수와 동일함 • 자기 디스크 장치의 3요소 : 디스크(Disk), 액세스 암(Access Arm), 헤드(Head)

잠깐만요 DASD / SASD

DASD(Direct Access Storage Device, 직접 접근 저장 매체)
- 기억공간의 필요한 위치를 직접 접근할 수 있으므로, 임의 접근, 순차 접근이 모두 가능합니다.
- 테이프 장치를 제외한 장치는 모두 DASD입니다.

SASD(Sequential Access Storage Device, 순차 접근 저장 매체)
- 기억공간의 필요한 위치를 접근하기 위해서는 처음부터 순서대로 접근해야만 합니다.
- 대표적인 매체에는 자기 테이프가 있습니다.

핵심 105 07.9, 04.9, 01.3, 00.10, 99.6

블로킹(Blocking)

블로킹이란 1개 이상의 논리적 레코드를 묶어서 테이프에 기록하는 방식이다.

비블로킹(Unblocking)

IRG	논리 레코드	IRG	논리 레코드	IRG	논리 레코드	IRG	논리 레코드	IRG

블로킹(Blocking)

IBG	논리 레코드	논리 레코드	논리 레코드	IBG	논리 레코드	논리 레코드	논리 레코드	IBG

- 하나의 블록을 구성하는 논리 레코드의 개수를 블록화 인수(BF; Blocking Factor)라고 한다.
- 블로킹을 하면 블로킹을 하지 않았을 때에 비해 IRG의 수가 줄어들기 때문에 다음과 같은 장점이 있다.
  - 기억공간의 낭비가 줄어든다.
  - Access Time이 감소한다.
  - 입 · 출력 횟수가 감소한다.

핵심 106 13.8, 12.3, 08.9, 07.5, 06.5, 05.9, 05.5, 05.4, 04.3, 03.5, 01.6, 01.3

연관 기억장치(Associative Memory)

- 기억장치에서 자료를 찾을 때 주소에 의해 접근하지 않고, 기억된 내용의 일부를 이용하여 Access할 수 있는 기억장치로, CAM(Content Addressable Memory)이라고도 한다.
- 주소에 의해서만 접근이 가능한 기억장치보다 정보 검색이 신속하다.
- 캐시 메모리나 가상 메모리 관리 기법에서 사용하는 Mapping Table에 사용된다.
- 외부의 인자와 내용을 비교하기 위한 병렬 판독 논리회로를 갖고 있기 때문에 하드웨어 비용이 증가한다.

핵심 107 13.6, 03.3, 01.3, 99.4

메모리 인터리빙(Memory Interleaving)

- 여러 개의 독립된 모듈로 이루어진 복수 모듈 메모리와 CPU 간의 주소 버스가 한 개로만 구성되어 있으면 같은 시각에 CPU로부터 여러 모듈들로 동시에 주소를 전달할 수 없기 때문에, CPU가 각 모듈로 전송할 주소를 교대로 분산 배치한 후 차례대로 전송하여 여러 모듈을 병행 접근하는 기법이다.
- 메모리 인터리빙 기법을 사용하면 기억장치의 접근 시간을 효율적으로 높일 수 있으므로 캐시 기억장치, 고속 DMA 전송 등에서 많이 사용된다.

잠깐만요 메모리 인터리빙, 인터리빙, 디스크 인터리빙으로 혼용되어 사용됩니다.

핵심 108 14.5, 13.8, 12.8, 11.6, 10.9, 08.5, 08.3, 05.9, 05.3, 04.3, 03.8, 02.5, 02.3, 01.3, 00.10, 00.5, 99.8

캐시 메모리(Cache Memory)

- CPU의 속도와 메모리의 속도 차이를 줄이기 위해 사용하는 고속 Buffer Memory이다.
- 캐시는 주기억장치와 CPU 사이에 위치한다.
- 캐시 메모리는 메모리 계층 구조에서 가장 빠른 소자이며, 처리 속도가 거의 CPU의 속도와 비슷할 정도이다.
- 캐시를 사용하면 기억장치를 접근(Access)하는 횟수가 줄어들기 때문에 컴퓨터의 처리 속도가 향상된다.
- 최근에는 명령어와 데이터를 따로 분리하여 각각의 캐시 메모리에 저장하는 분리 캐시를 운용하기도 한다. 분리 캐시를 사용하면 적중률은 떨어지지만 캐시 접근 시 충돌을 방지할 수 있다.
- 명령어나 자료를 찾기 위하여 캐시 메모리에 접근하는 경우, 원하는 정보가 캐시 메모리에 기억되어 있을 때 적중(Hit)되었다고 하고, 기억되어 있지 않으면 실패했다고 한다.

$$\text{적중률} = \frac{\text{적중 횟수}}{\text{총 접근 횟수}}$$

- 매핑(Mapping Process) 프로세스
  - 주기억장치로부터 캐시 메모리로 데이터를 전송하는 방법으로 주로 연관기억장치(CAM, Associative Memory)를 사용한다.
  - 종류 : 직접 매핑, 어소시에이티브 매핑, 세트-어소시에이티브 매핑

– 직접 매핑은 같은 인덱스를 가졌지만 다른 tag를 가진 두개 이상의 워드가 반복 접근할 경우 적중률이 낮아질 수 있다.

핵심 109 가상 기억장치(Virtual Memory)

14.5, 13.3, 11.6, 10.9, 09.5, 08.3, 07.3, 06.3, 03.3, 02.5, 00.5

- 기억 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용할 수 있도록 하는 운영체제의 메모리 운영 기법이다.
- 가상 기억장치의 목적은 주기억장치의 용량 확보이다.
- 가상 기억장치는 하드웨어적으로 실제로 존재하는 것이 아니고 소프트웨어적인 방법으로 보조기억장치를 주기억장치처럼 사용하는 것이다.
- 사용자 프로그램을 여러 개의 작은 블록으로 나누어서 보조기억장치 상에 보관해 놓고 프로그램 실행 시 필요한 부분들만 주기억장치에 적재한다.
- 가상 기억장치에서 주기억장치로 자료의 페이지를 옮기기 위해서 번지를 조정해 주어야 하는데, 이를 매핑(Mapping, 사상)이라 한다.
- 주기억장치의 이용률과 다중 프로그래밍의 효율을 높일 수 있다.
- 가상 기억장치 기법에서 사용하는 보조기억장치는 디스크 같은 DASD 장치이어야 한다.
- 주소의 사용
  – 가상 기억장치 기법에서는 보조기억장치에 저장된 사용자 프로그램을 블록으로 나누어 블록에 대한 주소를 주기억장치와는 별도의 주소로 표현하여 필요시 해당 블록만을 주기억장치에 적재한다.
  – 가상주소(논리주소) : 보조기억장치상의 주소로, 이들 주소의 집합을 주소 공간이라고 하며, 교체 단위는 페이지를 사용함
  – 실기억주소(물리적 주소) : 주기억장치상의 주소로 물리적 주소라고도 하며, 이들 주소의 집합을 메모리 공간 또는 기억 공간이라 함. 교체 단위는 블록을 사용함
- 페이지 부재(Page Fault)
  – CPU가 액세스한 가상 페이지가 주기억장치에 없는 경우를 말한다.
  – Page Fault가 발생하면 요구된 Page가 주기억장치로 옮겨질 때까지 프로그램 수행이 중단된다.

3과목 · 시스템 분석 및 설계

07.5, 07.3, 06.9, 06.5, 06.3, 05.5, 05.4, 04.9, 04.5, 04.3, 03.8, 02.9, 02.5, 01.6

핵심 110 시스템의 개요

14.8, 14.5, 4.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.5, 08.3, 07.9,

- 정의 : 공통의 목적을 달성하기 위하여 여러 가지 상호 관련된 요소들을 유기적으로 결합한 것
- 특성 : 목적성, 자동성, 제어성, 종합성

목적성	서로 다른 기능을 가지고 있는 시스템의 각 구성 요소들이 어떤 하나의 공통된 최종 목표에 도달하고자 하는 특성
자동성	어떤 조건이나 상황의 변화에 대응하여 스스로 대처할 수 있는 특성
제어성	정해진 목표를 달성하기 위해 시스템에 오류가 발생했는지 감시하고 바르게 진행 되도록 하는 특성
종합성	항상 관련된 다른 시스템과 상호 의존 관계로 통합되는 특성

- 기본 요소

입력(Input)	처리할 데이터, 처리 방법, 처리 조건을 시스템에 투입하는 것
처리(Process)	입력된 데이터를 처리 방법과 조건에 따라 변환하거나 가공하는 것
출력(Output)	처리된 결과를 시스템에서 산출하는 것
제어(Control)	자료가 입력되어 출력될 때까지의 처리 과정이 올바르게 행해지는지 감독하는 것
피드백(Feedback)	출력된 결과가 예정된 목적을 만족시키지 못한 경우 목적 달성을 위해 반복 처리하는 것

핵심 111 시스템 분석가(SA)

11.8, 07.3, 05.5, 04.5, 02.9, 02.5, 01.3, 99.8, 99.4

- 시스템의 전반적인 흐름과 사용자들의 요구 사항을 파악하고 해결책을 마련하는 사람이다.
- 기업의 목적과 현행 시스템의 문제점을 정확히 이해하고 해결책을 제시할 수 있어야 한다.
- 업무 내용이나 시스템에 대한 분석 능력이 있어야 한다.
- 컴퓨터 기술과 관리 기법을 알아야 한다.
- 시간 배정과 계획 등을 빠른 시간 내에 파악할 수 있어야 한다.
- 컴퓨터 하드웨어와 소프트웨어에 대한 전반적인 지식을 가져야 한다.
- 업계의 동향 및 관계 법규 등을 파악할 수 있어야 한다.

- 창조력, 응용력, 현장 분석 경험이 있어야 한다.
- 사용자와 프로그래머, 경영진 간의 의사소통을 원활히 하는 해결사 역할을 수행할 수 있어야 한다.

핵심 112 시스템 개발 생명 주기(SDLC)

12.8, 09.8, 09.3, 08.9, 07.9, 07.5, 06.9, 06.5, 06.3, 05.4, 05.3, 04.9, 03.8, 02.9, 02.5, 02.3, 01.9, 00.10, 00.5

- 시스템을 개발하는 과정에서 공통적으로 반복되는 단계를 말하는 것이다.
- **시스템 개발 생명 주기의 순서** : 시스템 조사 → 시스템 분석 → 시스템 설계 → 시스템 구현 → 테스트 → 시스템 운용 → 유지보수

시스템 조사	현행 시스템의 상태와 문제점을 파악하고 해결 방안을 제안하는 단계로, 예비 조사와 기초조사로 나뉨
시스템 분석	• 조사 단계에서 조사된 사용자의 요구 사항과 현행 시스템의 문제점을 명확히 파악하여 요구 분석 명세서를 작성하는 과정으로, 기능 분석, 예비 설계, 비용 효과 분석 순으로 진행됨 • 기업 환경 조사, 현장 조사, 기업이 필요로 하는 기능과 활동을 조사하고, 자료 흐름도, 자료 사전, 소단위 명세서 등 기능 분석을 위한 도구를 사용하여 모델을 설계함
시스템 설계	시스템 분석에 의해 정의된 시스템 요구 분석 명세서를 토대로 하여 새로운 시스템을 구체화하는 단계 • 기본 설계 : 분석 결과에 따라 사용자 입장에서 시스템 전체를 개괄적으로 설계하는 단계 • 상세 설계 : 각 기능의 논리적인 절차를 확정하고, 구체적인 입 · 출력 내용, 코드 설계, 파일의 구체적 사양을 결정하기 위한 단계
시스템 구현	설계 단계에서 산출된 설계 사양에 따라 프로그래밍 언어를 이용하여 원시 코드를 작성하는 단계로, 프로그래밍(Programming) 또는 코딩(Coding)이라고도 함
테스트	사용자의 요구에 따라 시스템이 구현되었는지 검증하는 단계로, 테스트의 종류에는 통합 테스트, 시스템 테스트, 인수 테스트가 있음
시스템 운용 (이행)	개발된 시스템을 실제 업무 처리에 적용하여 활용하는 단계
유지보수	• 시스템 개발 단계 중 가장 많은 비용이 투입되는 단계 • 종류 : 수정 유지보수, 적응 유지보수, 완전 유지보수, 예방 유지보수

- **T.M.Ho의 시스템 개발 주기 순서** : 목적 설정 → 상황 조사 → 현행 시스템의 연구 → 사용자 요구 사항 분석 → 대안 평가 → 새로운 하드웨어와 소프트웨어 선택 → 새 시스템의 설계 → 새 시스템의 구축 → 새 시스템의 인도

06.5, 05.9, 05.5, 05.4, 05.3, 04.9, 04.5, 04.3, 03.8, 03.5, 02.9, 02.5, 02.3, 01.9, 14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.6, 11.3, 10.9, 10.5, 09.8, 09.5, 09.3, 08.9, 08.3, 07.5, 07.3, 06.9,

핵심 113 코드 설계

- 코드는 컴퓨터를 이용하여 자료를 처리하는 과정에서 분류 · 조합 및 집계를 용이하게 하고, 특정 자료의 추출을 쉽게 하기 위해서 사용하는 기호이다. 또한 어떤 단위별 수치를 알거나 파일을 체계화하기 위해서 사용된다.
- 코드의 기능

3대 기능	분류, 식별, 배열
그 밖의 기능	간소화, 표준화, 암호화, 단순화, 연상(표의성), 오류 검출, 구별, 추출

- **코드 설계 순서** : 코드화 대상 선정 → 코드화 목적의 명확화 → 코드 부여 대상 수 확인 → 사용 범위 결정 → 사용 기간 결정 → 코드화 대상의 특성 분석 → 코드 부여 방식의 결정 → 코드의 문서화

코드화 대상 선정	• 어떤 항목을 대상으로 코드화할 것인지를 결정하는 단계 • 정보의 체계화 유무, 정보처리 효율성 유무, 정보 호환성 유무, 정보 표준화 유무 등을 고려하여 대상 항목을 결정함
코드화 목적의 명확화	무엇을 위해 코드가 필요한지 목적을 명확하게 밝히고, 코드에 부여할 기능을 결정함
코드 부여 대상 수 확인	현장 조사 자료를 근거로 코드화 대상을 확인하되, 현재의 자료량과 이후 동향에 대해서도 고려함
사용 범위 결정	코드 대상 항목에 대하여 설계된 코드의 사용이 컴퓨터 처리에 한정되는가, 해당 업무에만 한정되는가, 관련 부문의 업무에 공통으로 사용되는가, 기업 전체에 사용되는가, 관련 있는 타기업 또는 공공 기관이 공통으로 사용할 것인지 등을 결정함
사용 기간 결정	만든 코드를 영구적으로 사용할 것인지, 한시적으로 사용할 것인지를 결정함
코드화 대상의 특성 분석	코드화 대상 항목의 사용 경로, 코드 변경 유무, 코드의 추가 및 삭제의 빈도율 등을 면밀히 분석함
코드 부여 방식의 결정	자료의 특성에 따라 코드 체계, 체크 디지트의 사용 여부, 코드 자릿수, 코드 부여 요령 등을 결정함
코드의 문서화	코드를 실제 사용할 프로그래머가 이해할 수 있도록 코드와 관련한 전반적인 내용을 문서화하여 작성함

- **코드 설계시 유의 사항** : 기계 처리의 용이성, 취급의 용이성, 분류의 편리성(공통성, 체계성), 확장성, 단순성, 고유성, 표의성, 함축성

07.3, 06.9, 06.5, 06.3, 05.9, 05.5, 05.4, 05.3, 04.5, 03.8, 03.5, 03.3, 02.9, 02.5, 02.3
14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.3, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.5, 08.3, 07.9, 07.5,

핵심 114 코드의 종류

순서 코드 (Sequence Code)	• 자료의 발생 순서, 크기 순서 등 일정 기준에 따라서 최초의 자료부터 차례로 일련 번호를 부여하는 방법(순차 코드, 일련 번호식 코드) • 항목수가 적고, 변경이 적은 자료에 적합함 • 일정 순서대로 코드를 할당하기 때문에 기억 공간의 낭비가 없고 자릿수가 가장 짧음 • 단순 · 명료하고, 이해하기 쉬움 • 발생 순서에 따른 코드인 경우 추가가 매우 편리함(확장성 용이) • 고유성이 있으므로 기억이 용이함 • 코드 중간에 누락된 자료를 삽입하기 어려움(융통성이 적음)
구분 코드 (Block Code)	• 코드화 대상 항목 중에서 공통성이 있는 것끼리 블록으로 구분하고, 각 블록 내에서 일련 번호를 부여하는 방법 • 자릿수가 비교적 짧고, 블록별로 식별과 분류가 쉬움 • 블록마다 여유 코드를 두어 코드의 추가를 쉽게 할 수 있지만, 여유 코드는 코드 낭비의 요인이 되기도 함 • 기계 처리가 어려움
그룹 분류식 코드(Group Classification Code)	• 코드화 대상 항목을 일정 기준에 따라 대분류, 중분류, 소분류 등으로 구분하고, 각 그룹 안에서 일련 번호를 부여하는 방법 • 분류 기준이 명확한 경우에 이용도가 높으며, 기계 처리에 가장 적합함 • 각 자리가 특정한 의미를 가지고 있어 구분별 분류와 집계가 편리하며 그 의미가 명확함 • 여유 부분이 있어 자료 추가를 쉽게 할 수 있음 • 자릿수가 길어질 수 있음
10진 코드 (Decimal Code)	• 코드화 대상 항목을 0~9까지 10진 분할하고, 그 각각에 대하여 다시 10진 분할하는 방법을 필요한 만큼 반복하는 코드(도서 분류식 코드) • 도서관에서 도서 정리 목적으로 널리 사용함 • 코드 체계가 명확하고, 무한대로 확장이 가능함 • 삽입 및 추가가 용이하고, 배열이나 집계가 가능함 • 분류 항목이 10개 이상일 때는 비효율적임 • 자릿수가 길어질 수 있고, 기계 처리가 어려움
표의 숫자 코드 (Significant Digit Code)	• 코드화 대상 항목의 성질, 즉 길이, 넓이, 부피, 지름, 높이 등의 물리적 수치를 그대로 코드에 적용시키는 방법(유효 숫자 코드) • 코드에 대상체의 성질을 그대로 표시하므로 기억하기 쉬움 • 코드의 추가 및 삭제가 용이함 • 자릿수가 길어질 수 있고, 기계 처리에 불편함
연상 코드 (Mnemonic Code)	• 코드화 대상 항목의 명칭이나 약호와 관계있는 숫자나 문자, 기호를 이용하여 코드를 부여하는 방법(기호 코드) • 코드만 보고도 대상 품목을 쉽게 연상할 수 있음 • 지명, 물건명, 상호명에 많이 적용함 • 자릿수가 길어질 수 있음

00.10, 99.10, 99.6
13.6, 12.8, 11.8, 10.3, 09.3, 08.9, 08.5, 08.3, 07.9, 07.3, 06.9, 06.5, 06.3, 05.9, 05.5, 04.3, 03.8, 02.5, 02.3, 01.9, 01.6,

핵심 115 코드의 오류 발생 형태

- 필사 오류(Transcription Error) : 입력 시 임의의 한 자리를 잘못 기록한 경우 발생(오자 오류)
- 전위 오류(Transposition Error) : 입력 시 좌우 자리를 바꾸어 기록한 경우 발생
- 이중 오류(Double Transposition Error) : 전위 오류가 2개 이상 발생한 경우
- 생략 오류(Omission Error) : 입력 시 한 자리를 빼놓고 기록한 경우 발생
- 추가 오류(Addition Error) : 입력 시 한 자리를 더 추가하여 기록한 경우 발생
- 임의 오류(Random Error) : 위의 오류가 2가지 이상 결합하여 발생한 경우

07.3, 06.9, 06.5, 06.3, 05.9, 05.5, 05.4, 04.9 04.5, 04.3, 03.8, 03.5, 03.3, 02.5, 02.3
14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.3, 07.9, 07.5,

핵심 116 입력 설계

- 입 · 출력 설계의 표준화 : 방식의 표준화, 매체의 표준화, 형식의 표준화, 등록의 표준화, 코드의 표준화
- 입력 설계 순서 : 입력 정보의 발생 → 입력 정보의 수집 → 입력 정보의 매체화 → 입력 정보의 투입 → 입력 정보의 내용

입력 정보의 발생에 대한 설계	• 입력 정보의 명칭과 작성 목적, 입력 정보의 발생자와 발생 장소, 입력 정보의 발생 방법 및 발생 형태, 입력 정보의 발생 주기 및 시기와 발생 건수, 오류 검사 방법 결정 • 발생한 정보를 언제, 어디서, 누가, 무슨 용도로 사용하냐에 따라 다르게 설계됨
입력 정보의 수집에 대한 설계	수집 담당자와 수집 장소, 수집 방법과 형태, 수집 경로와 수집 주기 및 시기, 수집 시의 오류 검사 방법 결정
입력 정보의 매체화에 대한 설계	매체화 담당자와 장소, 매체화 장치, 레코드의 길이와 형식, 매체화 주기 및 시기, 매체화 시의 오류 검사 방법, 입력 정보의 형태 결정
입력 정보의 투입에 대한 설계	입력 매체의 모양과 서식, 입력 정보의 투입 주기 및 시기, 투입 시의 오류 검사 방법 결정
입력 정보의 내용에 대한 설계	입력 항목의 배열 순서와 항목명, 입력 항목의 자릿수와 문자 구분, 입력 정보의 오류 검사 방법 결정

잠깐만요 ! 입력 매체 장치 선택 시 검토 사항
시스템의 이행 방법 및 운용 비용, 입력 정보 발생 분야에서의 업무 특성, 입력 매체와 매체화 장치의 특성, 출력 정보를 이용할 시점에 맞게 투입

• 데이터 입력 방식

집중 매체화 시스템	발생한 데이터를 전표 상에 기록하고, 일정 시간 단위로 일괄 수집하여 입력 매체에 수록하는 방식
분산 매체화 시스템	데이터를 발생한 장소에서 매체화하여 처리하는 방식
턴 어라운드 시스템	• 출력 시스템과 입력 시스템이 일치된 방식 • 입력된 자료가 처리되어 일단 출력된 후 이용자를 경유하여 다시 재입력되는 방식으로, 공과금, 보험료 징수 등의 지로 용지를 처리하는 데 사용됨
일괄 처리 시스템	일정 시간 동안 수집된 변동 자료를 컴퓨터의 입력 자료로 만들었다가 필요한 시점에서 이 자료들을 입력하여 실행한 후 그 결과를 출력시키는 방식의 시스템

잠깐만요 ! 대화형 입 · 출력 방식

- **프롬프트 방식** : 프롬프트가 위치한 곳에 사용자가 직접 명령어나 자료를 입력하는 방식으로 명령어 처리 방법의 구현이 쉽고, 복잡한 명령도 사용 가능함
- **메뉴 방식** : 시스템에서 사용되는 명령어나 선택 사항을 메뉴로 구성하여 화면에 진열하는 방식으로 배우기 쉽고 편리함
- **항목 채우기 방식** : 정보를 직접 입력할 수 없는 메뉴 방식의 단점을 개선한 방식으로, 화면에 항목의 이름과 함께 입력 영역을 같이 진열하여 사용자가 입력 영역에 값을 직접 입력하는 방식
- **아이콘 방식** : 화면에 여러 개의 항목을 진열하고 그 중의 하나를 선택 도구로 지정하여 직접 실행하는 방식

핵심

117 원시 전표 설계 시 고려할 사항

05.5, 05.3, 03.3, 00.10, 99.6

원시 전표는 현장에서 발생한 자료를 최초로 기록한 종이를 의미한다.

기입의 측면

- 빠르고, 정확하고, 쉽게 기입할 수 있어야 한다.
- 기입 항목은 가능한 한 적게 하고, 간단하게 적을 수 있어야 한다.
- 전표 번호나 발행 주문과 같은 고정 항목은 미리 인쇄하거나 선택할 수 있게 한다.
- 일정한 순서대로 기입할 수 있어야 한다.
- 혼란을 초래할 우려가 있는 것은 기입 요령을 명시해야 한다.

취급 및 관리의 측면

- 전표의 크기를 표준화하고, 종이의 질이나 두께 등을 규격화한다.
- 전표 종류에 따라 구별하기 쉽도록 색상을 다르게 선택한다.
- 철(Filing)하거나 묶기 편하도록 여백과 구멍의 위치 등을 고려한다.

입력의 측면

- 전표를 읽는 순서와 입력하는 순서가 같도록 한다.
- 입력할 항목과 입력하지 않을 항목을 명확히 구분할 수 있도록 한다.
- 숫자 항목인 경우 자릿수가 구분되도록 한다.
- 관련 있는 항목끼리 위치를 통일시킨다.

핵심

118 출력 설계/COM 시스템

05.3, 04.3, 03.8, 03.5, 02.9, 01.9, 01.6, 00.10, 00.5, 00.3, 99.8, 99.4
14.8, 14.5, 13.6, 13.3, 12.8, 12.5, 12.3, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.5, 08.3, 07.9, 06.9, 06.5,

- 출력 설계 순서 : 출력 정보의 내용 → 출력 정보의 매체화 → 출력 정보의 분배 → 출력 정보의 이용

출력 정보 내용에 대한 설계	출력할 항목과 명칭, 출력 항목의 배열 순서, 크기, 자릿수, 출력 항목의 문자 표현 방법, 출력 항목에 대한 집계 방법, 출력 정보의 오류 검사 방법을 결정함
출력 정보 매체화에 대한 설계	출력 형식, 출력 매체 및 장치, 출력 정보의 양과 출력 복사의 매수, 출력 장소와 출력 시기 및 주기, 출력 배열 순서, 장치의 특성, 작동의 용이성을 결정함
출력 정보 분배에 대한 설계	분배 책임자, 분배 방법 및 형태, 분배 경로, 분배 주기 및 시기에 대해 결정함
출력 정보 이용에 대한 설계	출력 정보명과 출력 정보의 이용 목적, 출력 정보의 이용자와 이용 경로, 이용 주기 및 시기, 기밀성의 유무와 보존에 대해 결정함

- COM 시스템 : 출력 정보를 마이크로 필름에 수록하는 것으로, 축소 보관과 반영구적 보존이 가능하고, 지도, 설계 도면, 학적부, 병원의 기록 등을 보존, 검색, 관리하기에 적합한 방식

핵심

119 물리 레코드의 형식

05.9, 00.10, 00.7, 99.8, 99.6

- 비블록화 고정 길이 레코드 : 하나의 논리 레코드를 그대로 전송하는 방식으로, 레코드의 수만큼 입 · 출력이 발생하므로 시간 낭비가 많고, 경제성이 좋지 않음
- 블록화 고정 길이 레코드 : 길이가 동일한 여러 개의 논리 레코드를 묶어 하나의 블록으로 구성한 형태로, 프로그램의 작성이 쉽고, 속도가 빠르며, 경제성이 좋음

- **비블록화 가변 길이 레코드** : 길이가 다른 하나의 논리 레코드를 그대로 전송하는 방식으로, 레코드의 길이 정보를 반드시 표시해야 하며, 프로그램의 작성이 어렵고, 시간이 낭비되며, 경제성이 좋지 않음
- **블록화 가변 길이 레코드** : 길이가 서로 다른 여러 개의 논리 레코드를 묶어 하나의 블록으로 구성한 형태로, 각각의 레코드와 블록의 길이 정보를 반드시 표시해야 하며 프로그램의 작성이 어렵지만 처리 속도가 빠르고 경제성이 좋음

04.5, 03.8, 02.3, 01.9, 01.6, 01.3, 00.5, 00.3, 99.4

핵심 **120 데이터 파일의 종류**

13.6, 13.3, 12.8, 12.3, 11.6, 11.3, 10.9, 10.5, 09.8, 08.9, 08.5, 08.3, 07.3, 06.9, 06.5, 06.3, 05.9, 05.5, 05.4, 05.3, 04.9,

- **마스터 파일(Master File)** : 전표 처리에서의 원장 또는 대장에 해당하는 파일로, 자료 관리의 중추적 역할을 담당하며 기본이 되는 파일임. 트랜잭션 파일에 의해 갱신됨
- **트랜잭션 파일(Transaction File)** : 거래 내역이나 변동 내용 등 일시적인 성격을 지닌 정보를 기록하는 파일로, 마스터 파일을 갱신하거나 조회할 때 사용함
- **요약 파일(Summary File)** : 다른 파일의 중요 내용이나 합계를 요약해 놓은 파일로, 집계용으로 많이 사용됨
- **히스토리 파일(History File)** : 후일 통계 처리에 사용할 자료나 사고 발생 시 마스터 파일 등을 원상 복구시키기 위한 자료를 보존한 파일로, 현재까지 변화된 정보를 포함함
- **트레일러 파일(Trailer File)** : 마스터 파일을 목적에 따라 여러 개의 파일로 나누었을 때 가장 끝부분에 해당하는 파일
- **원시 파일(Source File)** : 입력 데이터를 알맞은 매체에 맞게 변환하여 만든 파일
- **백업 파일(Backup File)** : 만일의 사고에 대비하여 마스터 파일을 백업해 놓은 파일

14.8, 11.8, 10.5, 09.3, 07.9, 06.3, 05.9, 03.5

핵심 **121 순차 편성(파일)**

- 입력되는 데이터들을 논리적인 순서에 따라 물리적 연속 공간에 순차적으로 기록하는 방식으로, 주로 자기 테이프에 사용된다.
- 급여 관리 등과 같이 변동 사항이 크지 않고 기간별로 일괄 처리를 주로 하는 경우에 적합하다.
- 기억공간을 효율적으로 사용할 수 있지만 검색 효율은 낮다.
- 매체 변환이 쉬워 어떠한 매체에도 적용할 수 있다.
- 파일 중간에서 레코드를 삽입 · 삭제하는 경우 파일 전체를 복사해야 하므로 시간이 많이 소요된다.

02.5, 01.9, 01.3, 00.10, 00.3

14.3, 13.8, 13.6, 12.8, 11.8, 10.9, 10.3, 08.9, 08.5, 07.5, 06.5, 06.3, 05.9, 05.4, 05.3, 04.9, 04.5, 04.3, 03.5, 03.3, 02.9,

핵심 **122 색인 순차 편성(파일)**

- 순차 처리와 랜덤 처리가 모두 가능하도록 레코드들을 키 값 순으로 정렬하여 기록하고, 레코드의 키 항목만을 모은 인덱스를 구성하여 편성하는 방식이다.
- 레코드를 참조하는 경우 인덱스를 탐색한 후 인덱스가 가리키는 포인터(주소)를 사용하여 참조하므로 랜덤 편성 파일에 비해 액세스 속도가 느리다.
- 순차 처리와 랜덤 처리가 모두 가능하다.
- 레코드 삽입 · 삭제 시 파일 전체를 복사할 필요가 없으므로 삭입 · 삭제가 용이하다.
- 일반적으로 자기 디스크에 많이 사용되며, 자기 테이프에서는 사용할 수 없다.
- 색인을 저장하기 위한 공간과 오버플로우 처리를 위한 별도의 공간이 필요하다.
- 파일이 정렬되어 있어야 하기 때문에 추가, 삭제가 많으면 파일을 재편성해야 하기 때문에 효율성이 떨어진다.
- **기본 데이터 구역** : 실제 레코드들을 기록하는 부분
- **인덱스 구역** : 기본 데이터 구역에 대한 인덱스가 기록되는 부분

트랙 인덱스	기본 데이터 구역의 한 트랙에 기록되어 있는 데이터 레코드들 중의 최대 키 값과 주소가 기록되는 인덱스로, 한 실린더당 하나씩 만들어짐
실린더 인덱스	각 트랙 인덱스의 최대 키 값과 해당 레코드가 기록된 실린더의 정보가 기록되는 인덱스로, 한 파일당 하나씩 만들어짐
마스터 인덱스	실린더 인덱스 구역의 정보가 많을 경우 그것을 일정한 크기의 블록으로 구성하는데, 이때 처리할 레코드가 어느 실린더 인덱스에 기록되어 있는지를 기록하는 인덱스

- **오버플로 구역** : 기본 데이터 구역에 빈 공간이 없어서 새로운 레코드의 삽입이 불가능할 때를 대비하여 예비로 확보해 둔 구역

실린더 오버플로(Cylinder Overflow)	각 실린더마다 오버플로 구역을 두어 기본 데이터 구역에서 오버플로된 데이터를 기록함
독립 오버플로(Independent Overflow)	실린더 오버플로 구역에 더 이상 오버플로된 데이터를 기록할 수 없을 때 사용할 수 있는 예비 공간으로, 실린더 오버플로 구역과는 별도로 만들어짐

00.5, 00.3, 99.4

핵심 14.8, 14.3, 13.8, 13.3, 12.3, 11.6, 11.3, 09.8, 09.5, 09.3, 08.9, 08.5, 08.3, 07.9, 07.5, 05.5, 05.3, 04.5, 03.8, 03.3, 00.10,

123 랜덤 편성(파일)

- 입력되는 정보를 기록 순서나 코드 순서와 같은 논리적 순서와 관계 없이 특정한 방법으로 키를 생성하여 임의의 위치에 보관하고 처리 시에도 필요한 장소에 직접 접근할 수 있도록 편성하는 방식이다.
- 처리하고자 하는 레코드를 주소 계산에 의하여 직접 처리할 수 있다.
- 은행의 온라인 시스템과 같은 대화식 처리에 가장 효율적인 방식이다.
- 접근 시간이 빠르고 레코드의 삽입, 삭제, 갱신이 용이하다.
- 어떤 레코드라도 평균 접근 시간 내에 검색이 가능하다.
- 충돌이 발생할 염려가 있으므로, 이를 위한 기억공간의 확보가 필요하다.
- 레코드의 주소 변환 과정을 위한 시간이 필요하다.
- 운영체제에 따라서는 키-주소 변환을 자동으로 하는 것도 있다.
- 주소 계산 방법에는 직접주소법과 디렉토리 조사법, 해싱 함수 이용법이 있다.

잠깐만요 ! 해싱 함수 이용법

- 해싱 함수를 이용하여 계산된 키 값(주소)에 해당하는 기억공간에 레코드를 보관하거나 보관된 레코드를 검색하는 방법입니다.
- **해싱 함수** : 레코드의 키 값에서 레코드가 저장되어 있는 기억장치의 주소를 계산해 내는 사상 함수
- **버킷(Bucket)** : 하나의 주소를 가지는 영역을 의미하는 것으로, 하나의 버킷은 하나 이상의 레코드를 포함할 수 있음
- **충돌(Collision)** : 2개의 서로 다른 레코드가 같은 기억공간(버킷)을 점유하려고 하는 현상
- **동거자(Synonym)** : 같은 버킷 주소를 갖는 레코드의 집합(유사어, 동의어)
- **해싱 함수 선택 시 고려 사항**
  - 오버플로(Overflow)의 최소화
  - 충돌(Collision)의 최소화
  - 계산 과정의 최소화(해싱 함수의 단순성)
  - 버킷의 크기
  - 키 변환 속도

05.4, 02.5, 99.10, 99.6

핵심

124 리스트 편성(파일)

- 레코드들을 일정한 규칙이나 제약 없이 기억공간에 자유롭게 기록하고, 각 레코드들은 다음 레코드의 주소를 가지고 있는 포인터를 통해 논리적인 순서로 연결하는 방식이다.
- 레코드는 데이터와 포인터로 구성되며, 포인터에는 다음 레코드의 주소가 저장된다.
- 앞뒤의 포인터 내용만 변경하여 레코드를 쉽게 삽입 · 삭제할 수 있다.
- 물리적으로 연속적인 공간을 확보하기 어려운 경우나 레코드의 수가 불규칙하게 변하는 경우에 효율적이다.
- 파일 구조가 복잡하므로 처리 효율이 떨어진다.
- 포인터로 인해 레코드의 크기가 커지고, 예비 영역이 필요하므로, 기억장소가 낭비된다.

06.9, 04.9, 04.3, 02.9, 01.9, 01.6, 00.7, 00.3, 99.4

핵심 14.8, 14.3, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.5, 08.3, 07.9, 07.5, 07.3,

125 파일 설계 순서

- 파일의 성격 검토 → 파일의 항목 검토 → 파일의 특성 조사 → 파일 매체의 검토 → 편성법 검토

파일의 성격 검토	검토 파일의 명칭, 작성 목적과 종류를 결정하고, 파일이 사용되는 적용 업무를 확인함
파일의 항목 검토	항목의 명칭과 저장 형식, 항목의 배열 순서와 자릿수, 레코드의 형식과 크기, 블록의 크기를 결정함
파일의 특성 조사	효율적인 파일의 처리 주기 및 처리 방식을 결정하고, 추가 · 수정 · 삭제의 발생 빈도와 처리 형태, 파일의 활동률을 확인함
파일 매체의 검토	• 기능 검토 사항 : 액세스 형태와 처리 방식, 처리 시간과 정보의 양, 작동의 용이성을 검토함 • 종합 검토 사항 : 저장 매체와 매체의 개수, 장치의 대수를 결정함
편성법 검토	순차 편성, 랜덤 편성, 색인 순차 편성, 리스트 편성 등 파일의 편성 방식을 결정함

05.5, 03.3, 01.9, 00.7, 00.5

핵심 14.8, 14.3, 13.8, 13.6, 13.3, 12.5, 12.3, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.5, 08.3, 07.9, 07.5, 05.9,

126 프로세스 설계

- 프로세스 설계는 입력 정보와 파일 정보를 가지고 출력 정보를 얻기까지의 업무 처리 절차와 흐름, 정보의 처리와 흐름을 명확하게 하는 것을 의미한다.

- 프로세스 설계 시 유의사항
  - 신뢰성과 정확성을 고려하여 처리 과정을 간결하고 명확히 표현한다.
  - 오류에 대비한 검사 시스템을 고려한다.
  - 시스템의 상태 및 구성 요소, 기능 등을 종합적으로 표시한다.
  - 새로운 시스템의 프로세스 설계뿐만 아니라 기존 시스템의 문제점 분석이 가능하도록 설계한다.
  - 정보의 흐름이나 처리 과정을 모든 사람이 이해할 수 있도록 표준화한다.
  - 오퍼레이터의 개입을 적게 해야 한다.
  - 사용자의 하드웨어와 프로그래밍에 관한 상식 수준을 고려한다.
  - 하드웨어와 프로그래머의 능력을 고려한다.
  - 분류 처리는 가능한 적게 하고, 조작을 간결화, 자동화하여 사용자의 수동 조작을 적게 한다.
  - 프로세스 전개의 사상을 통일하고, 하드웨어의 기기 구성, 처리 성능을 고려한다.
  - 운영체제를 중심으로 한 소프트웨어의 효율성을 고려한다.
- **프로세스 설계 순서** : 기본 사항 확인 → 처리 방식 설계 → 작업 설계

핵심

127 흐름도(Flowchart, 순서도)

14.5, 13.8, 12.8, 12.3, 11.8, 10.9, 10.5, 10.3, 08.3

- 시스템의 상태 및 구성 요소, 기능의 배열 순서와 총합 관계를 도형, 기호 등으로 표시한 것이다.
- **블록 차트(Block Chart)** : 시스템의 목적을 달성하는 데 필요한 모든 기능 및 각 부서를 블록으로 표시하는 차트로, 블록 다이어그램이라고도 함
- **시스템 흐름도(System Flowchart)** : 자료 발생부터 결과를 얻기까지 시스템의 전 과정을 나타내는 흐름도
- **프로세스 흐름도(Process Flowchart)** : 컴퓨터의 입력, 처리, 출력 과정을 나타내는 흐름도로, 오퍼레이터에게 처리 공정을 알려주고 컴퓨터의 전체적인 논리 구조의 파악, 컴퓨터의 사용 시간 계산 등에 사용됨
- **프로그램 흐름도(Program Flowchart)**
  - 시스템 흐름도 중에서 컴퓨터로 처리하는 부분을 중심으로 자료 처리에 필요한 모든 조작 순서를 표시하는 흐름도
  - 시스템 설계서에 따라 컴퓨터에 의한 처리 내용 및 조건, 입 · 출력 데이터의 종류와 출력 등을 컴퓨터의 기능에 맞게 정확하게 작성해야 함
  - 프로그램 흐름도 종류

개요 흐름도 (General Flowchart)	• 프로그램의 전체 내용을 개괄적으로 설명한 흐름도 • 컴퓨터로 처리하는 순서와 내용의 개요를 표시함
상세 흐름도 (Detail Flowchart)	• 코딩하기 직전에 작성되는 흐름도 • 각 처리 단계를 세분화하여 컴퓨터의 처리 순서와 자료의 이동 순서를 빠짐 없이 표시함

핵심

128 표준 처리 패턴

07.5, 07.3, 06.9, 06.5, 06.3, 05.5, 05.4, 05.3, 04.9, 04.5, 04.3, 03.8, 03.5, 03.3, 02.9, 02.5, 02.3
14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.5, 08.3, 07.9,

- **변환(Conversion)** : 입력 매체 상의 데이터에 대한 오류를 제거하고, 컴퓨터가 처리할 수 있는 형태로 편집하여 파일 매체로 변환(입력 변환)하고, 파일 매체에 저장된 내용을 사람이 확인할 수 있도록 출력 매체로 변환(출력 변환)하는 기능(매체 변환)
- **병합(Merge)** : 동일한 파일 형식을 갖는 2개 이상의 파일을 일정한 규칙에 따라 하나의 파일로 통합 처리하는 기능
- **갱신(Update)** : 마스터 파일의 내용을 변동 파일에 의해 추가, 삭제, 수정 등의 작업을 하여 새로운 내용의 마스터 파일을 생성하는 것
- **분배(Distribution)** : 하나의 파일 안에서 조건에 맞는 것과 그렇지 않은 것을 분리하는 기능
- **추출(Extract)** : 파일 안에서 특정 조건에 만족하는 데이터만을 골라내는 기능
- **대조(Matching)** : 2개의 파일을 대조시켜 그 기록 순서나 기록 내용을 검사하는 기능
- **조합(Collate)** : 레코드 형식이 서로 다른 2개 이상의 파일에서 조건에 맞는 것을 골라 새로운 레코드로 파일을 만드는 기능
- **생성(Generate)** : 파일을 읽어 들여서 데이터를 변형하여 입력 파일과 다른 형식의 새로운 파일을 작성하는 기능

07.3, 06.3, 05.9

핵심 129 오류 검사 시스템

14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.5, 07.9, 07.5,

• 컴퓨터 입력 단계에서의 검사 방법

체크 디지트 검사	코드를 설계할 때 본래의 코드에 검사를 위한 1자리의 숫자를 넣어줌으로써 컴퓨터에 의하여 자동으로 검사하며, 주민등록번호, 상품코드 등을 검사할 때 사용함
균형 검사 (Balance Check)	대차대조표에서 차변과 대변의 한계값을 검사하는 방법으로, 대차의 균형이나 가로, 세로의 합계가 일치하는가를 검사함
한계 검사 (Limit Check)	입력 데이터의 어떤 항목이 규정된 범위, 즉 상한 값, 하한 값 내에 있는지를 검사함
일괄 합계 검사 (Batch Total Check = Sum Check)	입력 데이터의 특정 항목에 대한 합계값을 미리 계산한 후 이것을 입력 데이터와 함께 입력하고, 컴퓨터 상에서 계산한 결과값과 수동 계산 결과값이 같은지를 검사함
데이터 수 검사 (Data Count Check)	컴퓨터로 처리할 데이터의 개수를 미리 파악해 두었다가 컴퓨터로 처리한 후 원래의 데이터 개수와 같은지 여부를 검사함
타당성 검사(Validity Check), 논리 검사	입력된 데이터에 논리적으로 오류가 있는지를 검사하는 방법
대조 검사(Matching Check)	입력 데이터와 시스템에 보관된 별도의 코드 표를 대조하여 그것이 일치하는지를 검사하는 방법
순차 검사(Sequence Check)	입력되는 데이터의 순서가 이미 정해진 순서와 일치하는지를 검사함

• 계산 처리 단계에서의 검사 방법

부호 검사(Sign Check = Plus-Minus Check)	계산 결과가 양수 또는 음수인지를 검사하는 방법
중복 레코드 검사(Double Record Check)	계산 처리하는 과정에서 동일한 레코드가 있는지를 검사하는 방법
불일치 레코드 검사(Unmatch Record Check)	마스터 파일과 트랜잭션 파일을 조합할 때 키 항목이 일치하는지의 여부를 검사하는 방법
오버플로 검사(Overflow Check)	계산된 결과가 규정된 자릿수 또는 한계를 초과하는지를 검사하는 방법
제로 균형 검사(Zero Balance Check)	계산 결과가 0이 되는지를 검사하는 방법
불능 검사(Impossible Check)	0으로 나누는 경우가 있는지를 검사하는 방법

07.3, 04.9, 04.5, 03.3, 02.9, 00.10, 00.7, 00.5, 99.6

핵심 130 프로그램 설계서

- 프로그래머의 업무 수행을 신속 · 정확하게 지원하는 작업 지시서의 역할을 하는 것이다.
- 시스템 분석가(SA) 또는 시스템 엔지니어(SE)가 작성한다.
- 프로그램 설계서 작성 효과
  - 프로그래머의 인사 이동 시 발생할 수 있는 결함을 방지할 수 있다.
  - 시스템의 수정, 변경, 보수, 운영, 관리가 용이하다.
  - 컴퓨터의 기종 변경 시 프로그램의 적용이 용이하다.
  - 공동 작업이 용이하여 생산성이 향상된다.
  - 교육 훈련의 참고 자료로 이용할 수 있다.
  - 장기 계획을 수립할 수 있으며, 비용을 절감할 수 있다.
  - 작업 부문별 담당자의 책임 한계가 명확해진다.
- 프로그램 설계서의 구성 : 시스템명 및 코드명, 설계 방침, 프로세스 흐름도, 코드표, 입 · 출력 설계표, 프로그래밍 지시서
- 프로그래밍 지시서의 구성 : 프로그램명, 설계서 작성자명, 프로그램의 작성 기간, 작성 비용, 작성 시기, 입 · 출력 일람, 처리 개요, 처리 명세, 프로그램 작성 후 제출할 사항, 참고 자료

06.3, 05.5, 05.4, 05.3

14.5, 14.3, 13.3, 12.5, 12.3, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.5, 08.3, 07.9, 07.5, 07.3, 06.9, 06.5,

핵심 131 시스템의 평가

- 시스템 평가의 목적 : 시스템의 성능과 유용도 판단, 처리 비용과 처리 효율 면에서 개선점 파악, 시스템 운용 관리의 타당성 파악, 다른 시스템 개발을 위한 참고 자료
- 정보 처리에 소요되는 시간의 견적 방법

입력에 의한 계산	프로세스 차트(작업 처리도)를 기초로 하여, 간단한 수식에 값을 대입하여 처리 시간을 계산하는 방법
컴퓨터에 의한 계산	처리 시간을 계산할 수 있는 응용 프로그램을 이용하는 방법으로, 파일 종류와 수량, 처리 순서, 기기 구성 등을 매개변수 형식으로 입력해 주면 각 작업에 대한 소요 시간 및 월 소요 시간 등이 자동으로 계산됨
추정에 의한 계산	시스템 설계자가 과거의 경험을 바탕으로 하여 처리 시간을 추정하는 방법으로, 시스템 설계자의 경험에 따라 정확도에 차이가 있으므로, 처리 시간을 정확하게 계산하기 어려움

• 시스템의 평가 항목

기능 평가	사용자가 요구했던 목표 및 목적 등의 기능을 정확하게 수행하는지를 평가하는 것
성능 평가	시스템이 운용 계획에서 마련한 운용 스케줄대로 수행하는지를 평가하는 것으로, 다음과 같은 기준에 따라 진행됨 • 응답 시간(Response Time)과 반환 시간(Turn-around Time) • CPU의 속도 및 기억 용량 • 파일 편성법과 액세스 방식 • 파일 장치 및 입 · 출력 장치의 속도 • 업무 프로그램의 구조와 사용 언어 • 업무 프로그램의 다중도 및 우선 순위
신뢰성 평가	시스템의 신뢰성은 시스템이 주어진 환경에서 주어진 시간 동안 오류 없이 작동할 확률을 의미하며, 신뢰성 평가를 위한 검토 항목은 다음과 같음 • 시스템 전체의 가동률 • 시스템을 구성하는 각 요소의 신뢰도 • 신뢰성 향상을 위해 시행한 처리의 경제적 효과

• 시스템의 신뢰성 측정

MTBF(Mean Time Between Failures)	• 평균 고장 간격. 수리가 가능한 시스템이 고장난 후부터 다음 고장이 날 때까지의 평균 시간 • MTBF = MTTF + MTTR
MTTF(Mean Time To Failures)	• 평균 가동 시간. 수리 불가능한 시스템의 사용 시점부터 고장이 발생할 때까지의 가동 시간 평균으로, 고장 평균 시간이라고도 함 • MTTF = $\frac{\text{가동중1} + \text{가동중2} + \cdots \text{가동중n}}{n}$
MTTR(Mean Time To Repair)	• 평균 수리 시간. 시스템에 고장이 발생하여 가동하지 못한 시간들의 평균 • MTTR = $\frac{\text{고장중1} + \text{고장중2} + \cdots \text{고장중n}}{n}$

• 신뢰도 : 시스템의 총 운용 시간 중 정상적으로 가동된 시간의 비율로, $\frac{MTTF}{MTTF + MTTR} \times 100\%$

$= \frac{MTBF}{MTBF + MTTR} \times 100\%$

$= \frac{MTTF}{MTBF} \times 100\%$ 로 구한다.

핵심 132 10.3, 09.5, 06.3, 01.3

소프트웨어 비용 산정

소프트웨어 비용 산정 요소

소프트웨어 비용 산정에 영향을 주는 요소에는 다음과 같은 것이 있다.

- 제품의 복잡도
- 제품의 크기
- 프로그래머의 자질
- 요구되는 신뢰도 수준
- 가용 시간
- 기술 수준

소프트웨어 비용 산정 방법

하향식 비용 산정 방법	• 시스템의 전체 비용을 산정한 후 각 구성 요소별로 비용을 세분화하는 방법 • 전문가 감정 : 조직 내에 있는 경험이 많은 두 명 이상의 전문가에게 비용 산정을 의뢰하는 방법 • 델파이식 : 조정자가 단계별로 여러 전문가들의 견해를 종합 · 조정하여 비용을 산정하는 방법
상향식 비용 산정 방법	• 시스템의 각 구성 요소에 대한 비용을 독립적으로 산정한 후 이들를 합산하는 방법 • 원시 코드 라인 수(LOC), 개발 단계별 인월 수, 수학적 산정 기법 등
COCOMO (COnstruc-tive COst MOdel)	• Boehm에 의해 고안된 수학적 비용 산정 방법으로, 상향식 기법에 속함 • 시스템 규모를 예측하고 정해진 식에 대입하여 소요 인원과 개발 인원을 예측하여 소프트웨어 개발 비를 산정함

핵심 133 05.4, 00.5, 99.10, 99.6

개발 단계에 따른 테스트 종류

- 단위 테스트 : 각각의 모듈에 대해 독립적인 환경에서 기본 데이터만 입력하여 테스트하는 방법
- 통합(결합) 테스트 : 시스템 모듈 간의 상호 인터페이스가 원활하게 수행되고 있는가를 확인하는 작업으로, 프로그램 단위별로 디버깅이 끝난 것을 모아 서로 연관된 프로그램군(Group)을 계통적으로 검사하며, 시스템 분석가에 의해 테스트 데이터가 작성됨(결합 테스트)
- 시스템 테스트 : 완성된 시스템이 초기의 목적을 만족시키는가를 확인하는 작업임
- 인수 테스트 : 사용자의 요구사항을 만족시키는가를 확인하는 작업으로, 개발 과정 이후에 사용자에게 인수하는 과정에서 수행함
- 테스팅 검토 : 테스트가 계획대로 수행되었는지를 확인하는 작업

04.9, 04.5, 04.3, 03.8, 03.5, 03.3, 02.9, 02.5, 02.3, 01.9, 01.6, 00.5, 00.3, 99.10, 99.8

핵심 14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.5, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.5, 07.5, 07.3, 05.9, 05.5,

134 문서화/문서화의 목적 및 효과

문서화

- 문서화는 시스템의 개발 요령과 순서 등 시스템 개발에 관련된 모든 행위를 문서로 만들어 두는 것이다.
- 문서화는 시스템 개발 과정의 한 작업이라고 할 수 있으며, 시스템 개발 과정의 각 단계마다 문서화를 수행해야 정확한 문서화가 이루어진다.
- 프로그램 내에서도 문서화를 할 수 있다.

문서화의 목적 및 효과

- 시스템 개발팀에서 운용팀으로 인수 인계가 용이하다.
- 개발 후에 시스템의 유지보수가 용이하다.
- 의사 소통이 잘돼 개발팀을 원활히 운용할 수 있다.
- 시스템 개발중의 추가 변경 또는 시스템 개발 후의 변경에 따른 혼란을 방지할 수 있다.
- 시스템 개발 방법과 순서를 표준화할 수 있어 효율적인 작업과 관리가 가능하다.
- 복수 개발자에 의한 병행 개발을 가능하게 한다.
- 타 업무 개발에 참고할 수 있다.
- 정보를 축적하고, 생산성을 향상시킬 수 있다.

핵심 14.5, 13.3, 12.8, 12.3, 08.5

135 소프트웨어 위기

- 여러 가지 원인에 의해 소프트웨어 개발 속도가 하드웨어 개발 속도를 따라 가지 못해 소프트웨어에 대한 사용자들의 요구 사항을 처리할 수 없는 문제가 발생함을 의미한다.
- 소프트웨어 위기의 현상
  - 개발 인력의 부족과 그로 인한 인건비 상승
  - 성능 및 신뢰성의 부족
  - 개발 기간의 지연 및 개발 비용의 증가
  - 유지보수의 어려움과 이에 따른 비용 증가
  - 소프트웨어의 생산성과 품질 저하
- 소프트웨어 위기의 발생 원인
  - 논리적인 소프트웨어의 특징을 이해하지 못했다.
  - 소프트웨어에 대한 관리 소홀로 효율적인 자원 통제가 이루어지지 못했다.
  - 소프트웨어의 품질이나 유지보수는 고려하지 않고, 프로그래밍에만 집착함으로써 다양하고 복잡해지는 요구 사항을 처리하지 못했다.

핵심 14.8, 14.3, 13.6, 12.3, 11.3, 10.3, 05.9, 04.3, 03.8, 03.3, 02.3, 01.9, 00.3, 99.4

136 소프트웨어 생명 주기 – 폭포수 모델

- 폭포수 모델(Waterfall Model)은 고전적(전통적) 생명주기 모델로서, 폭포수에서 한번 떨어진 물은 거슬러 올라갈 수 없듯이 소프트웨어 개발도 각 단계를 확실히 매듭짓고 그 결과를 철저하게 검토하여 승인 과정을 거친 후에 다음 단계를 진행하며, 이전 단계로 넘어갈 수는 없는 선형 순차적 모형이다.
- 가장 오랫동안 사용되어 온 모델로 적용 사례가 많다.
- 단계별 정의가 분명하며, 각 단계별로 산출물이 명확히 나온다.
- 개발 과정중에 발생하는 새로운 요구 사항을 반영하기 어려우므로 처음부터 모든 요구 사항들을 명확하게 제시해야 한다.
- 프로젝트 관리 및 자동화가 어렵다.
- 대규모 시스템에 적용하기 어렵다.
- 두 개 이상의 과정을 병행하여 수행할 수 없다.
- 개발 순서 : 타당성 조사 → 계획 → 요구사항 분석 → 기본 설계(개략 설계) → 상세 설계 → 구현(Coding) → 통합 시험 → 시스템 실행 → 유지보수

타당성 조사	시스템의 정의와 가능성 조사 및 다른 방법과 비교 조사하는 단계
계획	사용자 문제의 정의, 전체 시스템 차원의 기본 목표와 요구사항 결정, 추진 방안의 제시를 통해 시스템 개발 비용 및 소요 기간, 인력 등의 개발 계획을 수립하는 단계
요구사항 분석	소프트웨어에 요구되는 기능, 성능 그리고 인터페이스 등 사용자의 요구사항을 구체적으로 이해하는 단계
기본 설계 (=개략 설계)	개발될 소프트웨어에 대한 전체적인 하드웨어 및 소프트웨어 구조, 제어 구조, 자료 구조의 개략적인 설계를 작성하는 단계
상세 설계	각 단위 프로그램에 대한 사항을 상세히 기술하는 단계
구현 (Coding)	설계 단계에서 만들어진 설계 사양서를 바탕으로 프로그램을 작성하는 단계로, 코딩과 디버깅, 단위 테스팅을 수행하는 단계

통합 시험	단위 프로그램별로 구현된 것을 통합시키며 시험하는 단계
시스템 실행	전체 시스템이 정확하게 실행하는가를 확인하는 단계
유지보수	시스템의 사용중에 발생하는 여러 변경 사항에 대해 적응하고, 변화에 대비하는 과정

핵심 137 소프트웨어 생명 주기 – 기타

14.5, 11.8, 11.6, 10.5, 09.8, 09.5, 09.3, 05.3, 04.5, 03.5, 99.10

- 프로토타이핑 모델(Prototyping Model)
  - 사용자의 요구사항을 정확히 파악하기 위해 실제 개발될 소프트웨어에 대한 모형(Prototype, 시제품)을 만들어 의사소통의 도구로 삼으면서 개발하는 기법
  - 개발 순서 : 요구사항 분석 → 프로토타입 설계 → 프로토타입 개발(구축) → 고객의 평가 → 프로토타입 정제(조정) → 완제품 생산(구현)
- 나선형 모델
  - 폭포수 모델과 프로토타이핑 모델의 장점에 위험 분석 기능을 추가한 모델(점증적 모델)로, 초기에 위험 요소를 발견하지 못할 경우 위험 요소를 제거하기 위해서 많은 비용이 들 수 있다.
  - 복잡하고 규모가 큰 시스템의 소프트웨어 개발에 가장 현실적인 개발 모델로 대두되고 있다.
  - 시스템을 개발하면서 생기는 위험을 관리하고, 최소화하는 것이 주목적이다.
  - 개발 순서 : 계획 및 정의 → 위험 분석 → 공학적 개발 → 고객 평가
- 4세대 기법(4GT) : 자연어로 표현하는 4세대 언어를 이용하여 개발자가 조사한 요구사항을 자동으로 구현(Coding)시키는 비절차적 기법

핵심 138 IPT/HIPO

01.6, 01.3, 00.10, 00.5, 00.3
10.5, 09.5, 08.3, 07.9, 07.5, 07.3, 06.9, 06.5, 06.3, 05.9, 05.5, 05.4, 04.9, 04.5,04.3, 03.8, 03.5, 02.9, 02.3, 01.9,

IPT(Improved Programming Technique)

소프트웨어의 품질 개선과 생산성 향상을 위해 사용되는 프로그램 개발 기법

- 기술적 측면 : 구조적 설계(복합 설계), 구조적 코딩, 하향식 프로그래밍

※ IPT 보조 도구 : HIPO, 모듈 설계, 구조 도표, 의사 코드, N-S Chart

- 관리적 측면 : 책임 프로그래머 팀, 구조적 검토회의(Walkthrough), 검열(Inspection), 개발 지원 라이브러리
- IPT 기법의 도입 목적 : 보기 쉽고, 이해하기 쉬움, 생산성 향상, 유지보수 용이, 품질 향상

HIPO(Hierarchy Input Process Output)

- 시스템 실행 과정인 입력, 처리, 출력을 계층적으로 기술하는 방법이다.
- 시스템을 설계하거나 문서화하기 위한 도구이다.
- 체계화된 문서 작성이 가능하며, 보기 쉽고 알기 쉽다.
- 하향식(Top-Down) 방식을 사용하여 나타낸다.
- 개발 과정에서 문서화를 부산물로 얻을 수 있다.
- 도표 상에 기능 위주로 입력 내용, 처리 방법, 출력 내용이 제시되므로 시스템의 이해가 쉽다.
- 기능과 자료의 의존 관계를 동시에 표현할 수 있다.
- 유지보수 및 변경이 용이하다.
- HIPO의 구성

도식 목차 (Visual Table of Contents)	• HIPO에서 지정된 기능을 계층적으로 나타낸 도표 • 시스템의 구조와 각 기능의 관계를 도식화한 것으로, 특정 기능을 쉽게 찾을 수 있음
총괄 도표 (Overview Diagram)	• 시스템 또는 프로그램의 기능을 입력, 처리, 출력 관계로 도표화한 것 • 사용자의 관점에서 본 시스템 또는 프로그램의 기능과 처리 내용을 나타내는 것
상세 도표 (Detail Diagram)	총괄 도표에 나타난 기능을 구성하는 기본 요소들을 상세히 기술한 도표

핵심 139 구조적 분석

10.9, 07.9, 00.3, 99.6

구조적 분석의 개념

- 구조적 분석은 자료의 흐름과 처리를 중심으로 하는 요구 분석 방법이다.
- 도형 중심의 분석용 도구와 분석 절차를 이용하여 사용자의 요구 사항을 파악하고 문서화하는 체계적인 분석 기법이다.
- 구조적 분석용 도구에는 자료 흐름도(DFD), 자료 사전(DD), 소단위 명세서(Mini-Spec.), 개체 관계도(ERD), 상태 전이도(STD) 등이 있다.

구조적 분석의 효과

- 도형 중심의 문서화 도구를 사용함으로써 분석자와 사용자간 대화가 용이하다.
- 시스템 분석 시 사용자의 참여 기회를 확대하여 사용자의 요구 사항을 정확하게 작성할 수 있다.
- 시스템을 하향식으로 세분화하므로, 분석의 중복성을 배제할 수 있다.
- 사용자의 요구 사항을 논리적으로 표현하여 전체 시스템을 일관성 있게 이해할 수 있다.
- 시스템 개발의 모든 단계에서 필요한 명세서를 작성할 수 있다.

00.7, 00.5, 99.10

핵심

140 자료 흐름도(DFD)

14.5, 13.8, 13.6, 13.3, 12.5, 11.3, 10.5, 09.5, 09.3, 08.9, 06.9, 06.5, 06.3, 05.4, 05.3, 04.9, 03.5, 03.3, 02.9, 01.6, 00.10,

- 시스템의 처리 과정을 자료의 흐름에 중점을 두어 기술하는 분석용 도구로, 버블 차트라고도 한다.
- 시스템의 활동적인 구성 요소 및 그들 간의 연관 관계를 모형화하는 것으로, 논리적으로 일관성이 있어야 한다.
- 하향식 분할의 원리를 적용하여 그림(도형) 중심으로 표현한다.
- 기능별로 분할하여 다차원적으로 표현한다.
- 자료 흐름도의 구성 요소

구성 요소	의미	표기법
처리(Process)	입력된 자료를 출력으로 변환하는 것	◯
자료 흐름 (Data Flow)	발생지, 종착지, 처리 및 저장소 사이에서 자료의 흐름을 나타냄	⟶
자료 저장소 (Data Store)	시스템 상의 자료를 저장하기 위한 장소	═══
단말 (Terminator)	• 시스템에 필요한 자료가 입력되는 발생지와 시스템에서 처리된 자료가 출력되는 종착지를 나타냄 • 외부에 존재하는 사람이나 조직체 등	▭

14.8, 14.3, 12.8, 12.3, 11.8, 11.6, 10.3, 09.8, 08.5, 08.3, 03.8, 02.9, 02.5, 02.3, 01.9, 01.3, 00.7, 99.4

핵심

141 자료 사전(DD)

- 자료 흐름도에서 대상이 되는 시스템과 관련된 모든 자료에 대한 기본적인 사항들을 더 자세히 정의하기 위해 사용되는 도구로, 메타 데이터(Meta Data) 또는 데이터의 데이터라고도 한다.
- 이름을 이용해 정의를 쉽게 찾을 수 있어야 하며, 이름이 중복되어서는 안 된다.
- 갱신하기 쉬워야 하며, 정의하는 방식이 명확해야 한다.
- 중복된 정의가 없어야 한다.
- **자료 사전의 정의 대상** : 자료 흐름(Data Flow)을 구성하는 자료 항목, 자료 저장소(Data Store)를 구성하는 자료 항목, 자료에 대한 의미, 자료 요소(Data Element)의 단위 및 값
- 자료 사전의 기호

기호	의미	기호	의미
=	자료의 정의	{ }	자료의 반복
+	자료의 연결	|	대체 항목의 나열
( )	자료의 생략	* *	자료의 설명
[ ]	자료의 선택		

06.9, 04.3, 02.9, 02.5, 02.3, 00.7

핵심

142 구조적 프로그래밍의 규칙

구조적 프로그래밍은 Dijkstra가 제창한 것으로, 신뢰성 있는 소프트웨어의 생산과 코딩의 표준화 등을 위해 개발된 방법이다.

- 프로그램의 제어 흐름을 선형화한다.
- 단일 입구와 단일 출구만 가지게 하고, GOTO문은 사용하지 않는다.
- 구조화 이론의 세 가지 기본 논리 구조(순차, 선택(조건), 반복)만을 사용한다.

06.3, 05.3, 04.9, 00.5, 99.8, 99.6

핵심

143 N-S 차트

- 논리의 기술(절차)에 중점을 둔 도형식 표현 도구로, 프로그램의 논리적인 구조를 사각형 박스로 표시한다.
- 순서도의 대안으로 제시된 것으로, 상세 처리 과정의 표현 도구로 사용된다.

- 논리 구조의 기본 형태인 순차, 선택, 반복 구조를 시각적으로 표현한다.
- 하나의 입구와 출구를 가지며, 분기(GOTO) 명령이나 화살표를 사용하지 않는다.
- 배우기 쉽고 읽기 쉬우며, 원시 코드로의 변환이 용이하다.
- 프로그램의 구조를 쉽게 파악할 수 있다.

핵심 144 모듈(Module)

14.8, 14.5, 13.6, 12.8, 11.8, 11.6, 11.3, 10.9, 10.3, 09.8, 09.3, 08.9, 08.3

- 모듈은 소프트웨어 구조를 이루는 기본 단위로, 하나 또는 그 이상의 논리적 기능들을 수행하는 컴퓨터 지시어들의 집합이다.
- 하향식(Top-Down) 개념으로 구성한다.
- 모듈은 서로 결합되어 종속적으로 실행되지만, 각 모듈의 컴파일은 독립적으로 수행된다.
- 단계마다 컴파일하므로 수행 속도가 느리다.
- 모듈을 분담하여 독립적으로 작성할 수 있다.
- 모듈은 매개 변수를 이용하여 값을 전달할 수 있다.
- **모듈 설계 시 유의 사항 및 모듈화의 이점**

모듈 설계 시 유의 사항	• 적절한 크기로 설계되어야 함 • 모듈을 다른 곳에 재사용할 수 있도록 표준화해야 함 • 모듈 간의 결합도는 낮게 하고, 모듈 내의 응집도는 높게 하여 모듈의 독립성을 높임 • 보기 쉽고 이해하기 쉽도록 작성함 • 자료의 추상화와 정보 은닉 개념을 고려해야 함
모듈화의 이점	• 프로그램의 복잡도가 절감됨 • 시스템 개발 시 소프트웨어의 품질을 증대시킬 수 있음 • 시스템 개발 시 시간과 노력을 절감할 수 있음 • 시스템 개발 비용을 절감할 수 있음 • 프로그램의 신뢰도를 향상시킬 수 있음 • 시스템의 디버깅과 시험, 수정이 용이함 • 모듈마다 사용할 변수를 새로 정의하지 않고 상속하여 사용할 수 있음 • 변수의 선언을 효율적으로 하여 메모리를 유용하게 쓸 수 있음 • 업무 성격이 비슷한 처리에서는 모듈을 공통으로 사용할 수 있음

99.8, 99.6

핵심 145 결합도(Coupling)

13.8, 13.6, 12.5, 11.3, 10.5, 08.5, 07.5, 07.3, 05.9, 04.5, 04.3, 03.5, 03.3, 02.9, 02.5, 02.3, 01.3, 00.10, 00.7, 99.10,

- 두 모듈 간의 상호 의존도를 측정하는 것으로, 결합도가 낮을수록 모듈의 독립성은 높아진다.
- 결합도의 순서(강함 → 약함) : 내용 결합도 → 공통 결합도 → 외부 결합도 → 제어 결합도 → 스탬프 결합도 → 자료 결합도

내용 결합도 (Content Coupling)	한 모듈이 다른 모듈의 내부 자료를 직접적으로 참조하는 경우의 결합도로, 결합도 중 의존도가 가장 높고, 순서 변경이 다른 모듈에 영향을 주기 쉬움
공통 결합도 (Common oupling)	서로 다른 모듈들이 하나의 기억장소에 설정된 공통의 데이터 영역을 공유하는 경우의 결합도
외부 결합도 (Externul Coupling)	어떤 모듈에서 외부(External)로 선언한 자료(변수)를 다른 모듈에서 참조하는 경우의 결합도
제어 결합도 (Control Coupling)	서로 다른 모듈 간에 교환하는 매개변수(Para- meter)가 제어 정보인 경우의 결합도
스탬프 결합도 (Stamp Coupling)	서로 다른 모듈이 동일한 자료 구조를 참조하는 경우의 결합도
자료 결합도 (Data Coupling)	서로 다른 모듈 간에 매개변수 또는 인수를 통해 꼭 필요한 자료만을 교환하는 경우의 결합도로, 설계 품질이 가장 좋음

핵심 146 응집도(Cohesion)

13.3, 12.5, 07.9, 06.9, 05.4, 02.3, 00.10, 00.7, 99.8, 99.6

- 한 모듈 내에 있는 구성 요소의 기능적 관련성을 평가하는 기준으로서, 응집도가 높을수록 모듈의 독립성은 높아진다.
- 응집도의 순서(강함 → 약함) : 기능적 응집도 → 순차적 응집도 → 통신적 응집도 → 절차적 응집도 → 시간적 응집도 → 논리적 응집도 → 우연적 응집도

기능적 응집도	모듈 내부의 모든 기능 요소가 한 가지의 작업만을 수행하는 경우의 응집도
순차적 응집도	모듈 내부의 한 기능 요소에 의한 출력 데이터가 모듈 내의 다음 기능 요소의 입력 데이터로 제공되는 경우의 응집도
통신적 응집도	동일한 입 · 출력 데이터를 이용하여 서로 다른 기능을 수행하는 요소들로 구성되며, 요소 사이에 데이터를 주고받거나 데이터를 똑같이 공유함
절차적 응집도	일정한 순서에 의해 처리되어야 할 요소들을 하나의 모듈로 구성한 경우의 응집도

시간적 응집도	특정 시간에 처리되는 몇 개의 기능을 모아 하나의 모듈로 구성한 경우의 응집도
논리적 응집도	논리적으로 서로 관련 있는 요소들을 모아 하나의 모듈로 작성하여, 그 모듈의 기능이 매개변수에 따라 처리 내용이나 처리 루트가 달라지는 경우의 응집도
우연적 응집도	모듈 내부의 각 요소들이 서로 관계없는 것끼리 모인 경우의 응집도

02.9, 02.5, 01.9, 01.6, 01.3, 00.7, 00.5, 99.8

핵심

147 객체지향 시스템을 구성하는 기본 단위

14.5, 13.8, 12.8, 12.3, 11.8, 11.6, 10.5, 09.8, 09.5, 08.5, 07.5, 07.3, 06.9, 06.5, 05.5, 05.4, 05.3, 04.5, 04.3, 03.8, 03.5,

- 객체(Object)
  - 데이터(속성)와 이를 처리하기 위한 연산(메소드, 함수)을 결합시킨 실체이다.
  - 객체마다 각각의 상태(State)를 가지고 있다.
  - 행위(Behavior)에 대한 특징을 나타내며, 객체는 식별성을 가진다.
  - 모든 객체는 다른 객체들과 구별할 수 있는 이름을 가진다.
  - 일정한 기억장소를 가지고 있으며, 상태가 변할 수 있다.
- 속성(Attribute) : 한 클래스 내에 속한 객체들이 가지고 있는 데이터 값들을 단위별로 정의하는 것으로서 성질, 분류, 식별, 수량 또는 현재 상태 등을 표현함
- 메소드(Method) : 객체에 정의된 연산을 의미하며, 객체의 상태를 참조하거나 변경하는 수단이 됨
- 클래스(Class) : 2개 이상의 유사한 객체들을 묶어서 하나의 공통된 특성을 표현하는 요소, 즉 공통된 특성과 행위를 갖는 객체의 집합이라고 할 수 있음. 한 클래스를 기준으로 하여 그 기준 클래스의 상위 클래스를 슈퍼 클래스, 하위 클래스를 서브 클래스라고 함
- 인스턴스(Instance) : 하나의 클래스에 속하는 각각의 객체를 의미함
- 인스턴스화(Instantiation) : 클래스로부터 새로운 객체를 생성하는 행위
- 메시지(Message) : 외부로부터 하나의 객체에 전달되는 메소드(행위)의 요구

14.8, 11.3, 09.3, 07.9, 07.5, 06.5, 05.9, 04.9, 04.5, 04.3, 03.5, 03.3, 02.5, 02.3, 01.6, 01.3, 00.3, 99.10

핵심

148 객체의 특징

- 주체성 : 다른 객체들과 식별할 수 있는 속성
- 다형성 : 하나의 메시지에 대해 각 클래스가 가지고 있는 고유한 방법으로 응답할 수 있는 능력을 의미함. 즉 같은 연산자라도 각 클래스에 따라 다른 기능을 수행할 수 있음
- 분류성 : 동일 속성과 행위를 갖는 객체들을 하나의 클래스로 분류하는 속성
- 상속성(Inheritance) : 이미 정의된 상위 클래스의 메소드를 비롯한 모든 속성을 하위 클래스가 물려받을 수 있는 것
- 추상화(Abstraction) : 불필요한 부분을 생략하고 객체의 속성 중 가장 중요한 것에만 중점을 두어 개략화시키는 것으로, 데이터의 공통된 성질을 추출하여 슈퍼 클래스를 선정하는 개념임
- 캡슐화(Encapsulation) : 데이터와 데이터를 조작하는 연산들을 함께 묶어 하나의 모듈내에서 결합되도록 하는 것으로, 객체의 자료가 변조되는 것을 막으며 그 객체의 사용자들에게 내부적인 구현의 세부적인 내용들을 은폐 시키는 것
- 정보 은닉(Information Hidden) : 캡슐화된 정보를 외부에 감추는 것

14.5, 13.8, 13.6, 10.9, 10.3, 09.8, 08.9, 07.3, 05.9, 03.8, 02.3, 01.6, 00.7, 99.8

핵심

149 럼바우(Rumbaugh)의 분석 기법

- 객체 모델링 기법(OMT)이라고 하며 그래픽 표기법을 이용하여 모든 소프트웨어 구성 요소들의 객체를 모델링한다.
- 시스템의 무엇(객체 모델링)에서 언제(동적 모델링), 어떤 일(기능 모델링)이 일어나는가를 분석한다.
- 분석 절차 : 객체 모델링 → 동적 모델링 → 기능 모델링

객체 모델링 (Object Modeling)	실세계 문제 영역으로부터 시스템에 요구되는 객체와 클래스를 찾아내어 그들 간의 관계를 연관성, 집단화, 일반화 중심으로 규명하며, 객체 다이어그램으로 나타내는 것
동적 모델링 (Dynamic Modeling)	시간의 흐름에 따라 변하는 객체들 사이의 제어 흐름, 상호 작용, 연산 순서 등의 동적인 행위를 상태 다이어그램으로 나타내는 것
기능 모델링 (Function Modeling)	다수 프로세스 간의 데이터 흐름을 중심으로 처리 과정을 데이터 흐름도로 나타내는 것

핵심 150 01.3, 00.10, 00.5, 00.3, 99.4

코드(Coad)와 요돈(Yourdon) 기법/Booch기법

코드(Coad)와 요돈(Yourdon) 기법

- **분석 기법** : E-R 다이어그램(개체 관계도)을 사용하여 개체(Entity)의 활동들을 데이터 모델링하는 데 초점을 둔 기법
- **설계 기법** : 분석 사항을 하향식 방법으로 설계에 접근하여 프로토타입으로 개발하는 객체지향 설계 기법으로, 문제 영역 요소, 사람과 상호 작용 요소, 작업(Task) 관리 요소, 데이터 관리 요소로 구성됨

Booch 기법

- 1991년 Booch에 의해 발표된 객체지향 설계 기법으로, 설계 단계에 중점을 두어 클래스 및 객체의 식별과 그것들 간의 관계를 강조하였다.
- 데이터 흐름도(DFD)를 사용해서 객체를 분해하고, 객체들 간의 인터페이스를 찾아 Ada 프로그램으로 변환시키는 기법이다.
- 전체 시스템의 가시화와 실시간 처리(Real Time)에 유용하다.
- 설계를 위한 문서화 기법을 강조한 기법으로, 분석 단계와 구현의 세부 사항에 취약하다.

4과목 · 운영체제

핵심 151 06.5, 05.3, 03.8, 01.3, 00.7

운영체제의 개요

정의	컴퓨터 시스템의 자원들을 효율적으로 관리하며, 사용자가 컴퓨터를 편리하고 효과적으로 사용할 수 있도록 환경을 제공하는 여러 프로그램의 모임으로, 제어 프로그램과 처리 프로그램으로 구분 • 제어 프로그램 : 시스템 전체의 작동 상태 감시, 작업의 순서 지정, 작업에 사용되는 데이터 관리 등의 역할 수행 – 감시 프로그램(Supervisor Program) – 작업 제어 프로그램(Job Control Program) – 자료 관리 프로그램(Data Management Program) • 처리 프로그램 : 제어 프로그램의 지시를 받아 사용자가 요구한 문제를 처리하기 위한 프로그램 – 언어 번역 프로그램(Language Translator Program) – 서비스 프로그램(Service Program) – 문제 프로그램(Problem Program) • 운영체제의 종류 : Windows, UNIX, LINUX, MS-DOS 등
목적 및 성능 평가 기준	• 처리 능력 및 신뢰도 향상, 사용 가능도 향상, 반환 시간의 단축 • 성능 평가 기준 – 처리 능력(Throughput) : 일정 시간 내에 시스템이 처리하는 일의 양 – 반환 시간(Turn Around Time) : 시스템에 작업을 의뢰한 시간부터 처리가 완료될 때까지 걸린 시간 – 사용 가능도(Availability) : 시스템의 각종 자원을 사용할 필요가 있을 때 즉시 사용 가능한 정도 – 신뢰도(Reliability) : 시스템이 주어진 문제를 정확하게 해결하는 정도
기능	• 프로세서, 기억장치, 입 · 출력장치, 파일 및 정보 등의 자원 관리 • 자원의 스케줄링 기능 제공 • 사용자와 시스템 간의 편리한 인터페이스 제공 • 시스템의 각종 하드웨어와 네트워크 관리 · 제어 • 시스템의 오류 검사 및 복구, 데이터 관리, 데이터 및 자원 공유, 시스템의 초기화 • 자원 보호 기능 제공 • 가상 계산기 기능 제공

핵심 152 00.3 14.8, 14.3, 12.5, 12.3, 10.9, 10.5, 09.8, 09.5, 08.9, 08.5, 08.3, 07.9, 05.4, 04.9, 04.3, 03.8, 02.5, 01.6, 00.10, 00.5,

운영체제 운용 기법 및 발달 과정

운영체제 운용 기법

일괄 처리(Batch Processing) 시스템	• 초기의 컴퓨터 시스템에서 사용된 형태로, 일정량 또는 일정 기간 동안 데이터를 모아서 한꺼번에 처리하는 방식 • 컴퓨터 시스템을 효율적으로 사용할 수 있음 • 사용자 측면에서는 반환(응답) 시간이 늦지만 하나의 작업이 모든 자원을 독점하므로 CPU 유휴 시간이 줄어듦 • 급여 계산, 지불 계산, 연말 결산 등의 업무에 사용
다중 프로그래밍(Multi-Programming) 시스템	• 하나의 CPU와 주기억장치를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식 • 하나의 주기억장치에 2개 이상의 프로그램을 기억시켜 놓고, 하나의 CPU와 대화하면서 동시에 처리함
시분할(Time Sharing) 시스템	• 여러 명의 사용자가 사용하는 시스템에서 컴퓨터가 사용자들의 프로그램을 번갈아 가며 처리해 줌으로써 각 사용자에게 독립된 컴퓨터를 사용하는 느낌을 주는 것이며 라운드 로빈(Round Robin) 방식이라고도 함 • 여러 사용자가 각자의 단말 장치를 통하여 동시에 운영체제와 대화하면서 각자의 프로그램을 실행함 • 하나의 CPU는 같은 시점에서 여러 개의 작업을 동시에 수행할 수 없기 때문에, CPU의 전체 사용 시간을 작은 작업 시간량(Time Slice)으로 나누어서 그 시간량 동안만 번갈아 가면서 CPU를 할당하여 각 작업을 처리함 • 다중 프로그래밍 방식과 결합하여 모든 작업이 동시에 진행되는 것처럼 대화식 처리가 가능함

다중 처리(Multi-Processing) 시스템	• 여러 개의 CPU와 하나의 주기억장치를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식 • 하나의 CPU가 고장나더라도 다른 CPU를 이용하여 업무를 처리할 수 있으므로 시스템의 신뢰성과 안정성이 높음
실시간 처리(Real Time Processing) 시스템	• 데이터 발생 즉시, 또는 데이터 처리 요구가 있는 즉시 처리하여 결과를 산출하는 방식 • 우주선 운행이나 레이더 추적기, 핵물리학 실험 및 데이터 수집, 전화교환장치의 제어, 은행의 온라인 업무, 좌석 예약 업무, 인공위성, 군함 등의 제어 업무 등 시간에 제한을 두고 수행되어야 하는 작업에 사용됨
다중 모드 처리(Multi-Mode Processing)	일괄 처리 시스템, 시분할 시스템, 다중 처리 시스템, 실시간 처리 시스템을 한 시스템에서 모두 제공하는 방식
분산 처리(Distributed Processing) 시스템	• 여러 개의 컴퓨터(프로세서)를 통신 회선으로 연결하여 하나의 작업을 처리하는 방식 • 각 단말 장치나 컴퓨터 시스템은 고유의 운영체제와 CPU, 메모리를 가지고 있음

발달 과정

일괄 처리 시스템 → 다중 프로그래밍, 다중 처리, 시분할, 실시간 처리 시스템 → 다중 모드 → 분산 처리 시스템

핵심 153 링커/로더

14.5, 10.9, 06.9, 06.5, 05.9, 05.3, 03.8, 03.5, 02.3, 01.3, 00.5

링커

- 언어 번역 프로그램이 생성한 목적 프로그램들과 라이브러리, 또 다른 실행 프로그램(로드 모듈) 등을 연결하여 실행 가능한 로드 모듈을 만드는 시스템 소프트웨어로 Linkage Editor라고도 한다.
- 연결 기능만 수행하는 로더의 한 형태로, 링커에 의해 수행되는 작업을 링킹(Linking)이라 한다.

로더

정의	컴퓨터 내부로 정보를 들여오거나 로드 모듈을 디스크 등의 보조기억장치로부터 주기억장치에 적재하는 시스템 소프트웨어
기능	• 할당(Allocation) : 실행 프로그램을 실행시키기 위해 기억장치 내에 옮겨놓을 공간을 확보하는 기능 • 연결(Linking) : 부 프로그램 호출 시 그 부 프로그램이 할당된 기억장소의 시작주소를 호출한 부분에 등록하여 연결하는 기능 • 재배치(Relocation) : 디스크 등의 보조기억장치에 저장된 프로그램이 사용하는 각 주소들을 할당된 기억장소의 실제 주소로 배치시키는 기능 • 적재(Loading) : 실행 프로그램을 할당된 기억공간에 실제로 옮기는 기능
종류	• Compile And Go 로더 : 별도의 로더 없이 언어 번역 프로그램이 로더의 기능까지 수행하는 방식(할당, 재배치, 적재 작업을 모두 언어 번역 프로그램이 담당) • 절대 로더(Absolute Loader) : 목적 프로그램을 기억 장소에 적재시키는 기능만 수행하는 로더(할당 및 연결은 프로그래머가, 재배치는 언어 번역 프로그램이 담당) • 직접 연결 로더(Direct Linking Loader) : 일반적인 기능의 로더로, 로더의 기본 기능 4가지를 모두 수행하는 로더 • 동적 적재 로더(Dynamic Loding Loader) : 프로그램을 한꺼번에 적재하는 것이 아니라 실행 시 필요한 일부분만을 적재하는 로더

05.5, 04.9, 04.5, 03.8, 02.5, 01.9, 01.3, 00.10, 00.5, 99.10, 99.8, 99.6, 05.3, 04.9, 14.8, 14.5, 14.3, 13.8, 13.6, 13.3,12.5, 12.3, 11.6, 11.3, 10.9, 10.5, 10.3, 9.8, 09.5, 09.3, 08.5, 08.3, 07.5, 07.3, 06.3, 05.9,

핵심 154 프로세스의 여러 가지 정의/PCB

프로세스의 여러 가지 정의

- 실행중인 프로그램, PCB를 가진 프로그램, 실기억장치(주기억장치)에 저장된 프로그램
- 프로세서가 할당되는 실체, 프로시저가 활동중인 것
- 비동기적 행위를 일으키는 주체, 지정된 결과를 얻기 위한 일련의 계통적 동작
- 목적 또는 결과에 따라 발생되는 사건들의 과정
- 운영체제가 관리하는 최소 실행 단위
- 프로세서 제어 블록의 존재로서 명시되는 것

PCB(Process Control Block)

- 운영체제가 프로세스에 대한 중요한 정보를 저장해 놓는 곳이다.
- 각 프로세스가 생성될 때마다 고유의 PCB가 생성되고 프로세스가 완료되면 PCB가 제거된다.
- PCB에 저장되어 있는 정보
  - 프로세스의 현재 상태
  - 포인터(부모 프로세스 · 자식 프로세스 · 프로세스가 위치한 메모리 · 할당된 자원에 대한 포인터)
  - 프로세스 고유 식별자
  - 스케줄링 및 프로세서의 우선순위
  - CPU 레지스터 정보
  - 주기억장치 관리 정보
  - 입 · 출력 상태 정보
  - 계정 정보

핵심 155 프로세스 상태 전이

05.9, 05.5, 05.3, 03.3, 00.10, 99.8, 99.4

프로세스 상태 전이는 프로세스가 시스템 내에 존재하는 동안 프로세스의 상태가 변하는 것을 의미한다.

프로세스의 주요 상태

준비(Read)	프로세스가 프로세서를 할당받기 위해 기다리고 있는 상태
실행(Run)	• 준비상태 큐에 있는 프로세스가 프로세서를 할당받아 실행되는 상태 • 프로세스 수행이 완료되기 전에 프로세스에게 주어진 프로세서 할당 시간이 종료(Time Run Out)되면 프로세스는 준비 상태로 전이됨 • 실행중인 프로세스에 입 · 출력(I/O) 처리가 필요하면 실행중인 프로세스는 대기 상태로 전이됨
대기(Wait), 보류, 블록(Block)	프로세스에 입 · 출력 처리가 필요하면 현재 실행중인 프로세스가 중단되고, 입 · 출력 처리가 완료될 때까지 대기하고 있는 상태

프로세스 상태 전이 관련 용어

Dispatch	준비 상태에서 대기하고 있는 프로세스 중 하나가 프로세서를 할당받아 실행 상태로 전이되는 과정
Wake-Up	입 · 출력 작업이 완료되어 프로세스가 대기 상태에서 준비 상태로 전이되는 과정

핵심 156 스레드(Thread)

14.3, 12.8, 12.3, 11.8, 10.9, 10.3, 09.5, 09.3, 07.9, 06.3, 00.10

- 스레드는 프로세스 내에서의 작업 단위이면서 시스템의 여러 자원을 할당받아 실행하는 프로그램의 단위이다.
- 하나의 프로세스에 하나의 스레드가 존재하는 경우에는 단일 스레드, 하나 이상의 스레드가 존재하는 경우에는 다중 스레드라고 한다.
- 프로세스의 일부 특성을 갖고 있기 때문에 경량(Light Weight) 프로세스라고도 한다.
- 자신만의 스택(Stack)과 레지스터(Register)를 가지며 독립된 제어 흐름을 갖는다
- 프로세스의 구성을 제어의 흐름 부분과 실행 환경 부분으로 나눌 때, 프로세스의 실행 부분을 담당함으로써 실행의 기본 단위가 된다.
- 운영체제의 성능을 개선하려는 소프트웨어적 접근 방법이다.

- 스레드의 분류

사용자 수준의 스레드	• 사용자가 만든 라이브러리를 사용하여 스레드를 운용함 • 속도는 빠르지만 구현이 어려움
커널 수준의 스레드	• 운영체제의 커널에 의해 스레드를 운용함 • 구현이 쉽지만 속도가 느림

- 스레드 사용의 장점
  - 하나의 프로세스를 여러 개의 스레드로 생성하여 병행성을 증진시킬 수 있다.
  - 하드웨어, 운영체제의 성능과 응용 프로그램의 처리율을 향상시킬 수 있다.
  - 응용 프로그램의 응답 시간(Response Time)을 단축시킬 수 있다.
  - 실행 환경을 공유시켜 기억 장소의 낭비가 줄어든다.
  - 프로세스들 간의 통신이 향상된다.
  - 스레드는 공통적으로 접근 가능한 기억장치를 통해 효율적으로 통신한다.

핵심 157 스케줄링

14.8, 07.9, 06.5, 03.8, 02.9, 01.6, 00.7, 00.5

- 정의 : 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업
- 목적 : 공정성, 처리율 증가, CPU 이용률 증가, 우선순위 제도, 오버헤드 최소화, 응답 시간 최소화, 반환 시간 최소화, 대기 시간 최소화, 균형 있는 자원의 사용, 무한 연기 회피
- 문맥 교환(Context Switching) : 하나의 프로세스에서 다른 프로세스로 CPU가 할당되는 과정에서 발생되는 것으로, 새로운 프로세스에 CPU를 할당하기 위해 현재 CPU가 할당된 프로세스의 상태 정보를 저장하고, 새로운 프로세스의 상태 정보를 설정한 후 CPU를 할당하여 실행되도록 하는 작업

핵심 158 프로세서(스) 스케줄링의 종류

09.5, 08.9, 07.9, 07.5, 06.9, 06.5, 05.9, 05.5, 05.3, 04.5, 04.3, 03.8, 03.5, 02.5, 02.3, 01.9, 01.6, 00.7, 99.10

비선점 (Non-preemptive) 스케줄링	• 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없는 스케줄링 기법 • 프로세스가 CPU를 할당받으면 해당 프로세스가 완료될 때까지만 CPU를 사용하므로 응답시간 예측이 용이함 • 모든 프로세스에 대한 요구를 공정하게 처리할 수 있음 • 일괄 처리 방식에 적합하며, 중요한(처리 시간이 짧은) 작업이 중요하지 않은(처리 시간이 긴) 작업을 기다리는 경우가 발생할 수 있음 • 종류 : FCFS(FIFO), SJF, 우선순위, HRN, 기한부
선점 (Preemptive) 스케줄링	• 하나의 프로세스가 CPU를 할당받아 실행하고 있을 때 우선순위가 높은 다른 프로세스가 CPU를 강제로 빼앗아 사용할 수 있는 스케줄링 기법 • 우선순위가 높은 프로세스를 빠르게 처리할 수 있음 • 주로 빠른 응답 시간을 요구하는 대화식 시분할 시스템, 온라인 응용 등에 사용 • 종류 : SRT, 선점 우선순위, Round Robin, 다단계 큐(MQ), 다단계 피드백 큐(MFQ)

핵심 159 비선점 스케줄링의 종류

07.5, 06.3

14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.6, 11.3, 10.9, 09.5, 09.3, 08.9, 10.5, 10.3, 09.8, 08.5, 08.3, 07.9,

FCFS(First Come First Service)	• 준비상태 큐에 도착한 순서에 따라 차례로 CPU를 할당하는 기법 • 먼저 도착한 것이 먼저 처리되어 공평성은 유지되지만 짧은 작업이 긴 작업을, 중요한 작업이 중요하지 않은 작업을 기다리게 됨
SJF (Shortest Job First)	• 실행 시간이 가장 짧은 프로세스에 먼저 CPU를 할당하는 기법 • 가장 적은 평균 대기 시간을 제공하는 최적 알고리즘
HRN(Hightest Response-ratio Next)	• 실행 시간이 긴 프로세스에 불리한 SJF 기법을 보완하기 위한 것으로, 대기 시간과 서비스(실행) 시간을 이용하는 기법 • 우선순위 계산 공식 = $\frac{\text{대기 시간} + \text{서비스 시간}}{\text{서비스 시간}}$
기한부 (Deadline)	• 프로세스에게 일정한 시간을 주어 그 시간 안에 프로세스를 완료하도록 하는 기법 • 시스템은 프로세스에게 할당할 정확한 시간을 추정해야 하며, 이를 위해서 사용자는 시스템이 요구한 프로세스에 대한 정확한 정보를 제공해야 함
우선순위 (Priority)	• 기다리는 각 프로세스마다 우선순위를 부여하여 그 중 가장 높은 프로세스에게 먼저 CPU를 할당하는 기법 • 우선순위의 등급은 내부적 요인과 외부적 요인에 따라 부여할 수 있음 • 각 작업마다 우선순위가 주어지며, 우선순위가 제일 높은 작업에게 먼저 프로세서가 할당됨 • 가장 낮은 순위를 부여받은 프로세스는 무한 연기 또는 기아 상태(Starvation)가 발생할 수 있음

핵심 160 에이징(Aging) 기법

09.3, 02.3, 00.10, 00.5, 00.3

- 시스템에서 특정 프로세스의 우선순위가 낮아 무한정 기다리게 되는 경우, 한번 양보하거나 기다린 시간에 비하여 일정 시간이 지나면 우선순위를 한 단계씩 높여 가까운 시간 안에 자원을 할당받도록 하는 기법이다.
- SJF나 우선순위 기법에서 발생할 수 있는 무한 연기 상태, 기아 상태를 예방할 수 있다.

핵심 161 선점 스케줄링의 종류

03.8, 03.5

14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 11.8, 11.6, 11.3, 10.9, 10.5, 09.8, 07.5, 07.3, 06.9, 06.5, 05.5, 05.4, 04.5, 04.3,

선점 우선순위	준비상태 큐의 프로세스들 중에서 우선순위가 가장 높은 프로세스에게 먼저 CPU를 할당하는 기법
SRT(Shortest Remaining Time)	비선점 기법인 SJF 알고리즘을 선점 형태로 변경한 기법으로, 현재 실행중인 프로세스의 남은 시간과 준비상태 큐에 새로 도착한 프로세스의 실행 시간을 비교하여 가장 짧은 실행 시간을 요구하는 프로세스에게 CPU를 할당하는 기법
RR(Round Robin)	• 시분할 시스템(Time Sharing System)을 위해 고안된 방식으로, FCFS 알고리즘을 선점 형태로 변형한 기법 • FCFS 기법과 같이 준비상태 큐에 먼저 들어온 프로세스가 먼저 CPU를 할당받지만 각 프로세스는 할당된 시간(Time Slice, Quantum) 동안만 실행한 후 실행이 완료되지 않으면 다음 프로세스에게 CPU를 넘겨주고 준비상태 큐의 가장 뒤로 배치됨 • 할당되는 시간이 클 경우 FCFS 기법과 같아지고, 할당되는 시간이 작을 경우 문맥 교환 및 오버헤드가 자주 발생됨
다단계 큐 (Multi level Queue)	프로세스들을 우선순위에 따라 시스템 프로세스, 대화형 프로세스, 일괄 처리 프로세스 등으로 상위, 중위, 하위 단계의 단계별 준비 큐를 배치하는 CPU 스케줄링 기법

다단계 피드백 큐 (Multi level Feedback Queue)	• 특정 그룹의 준비상태 큐에 들어간 프로세스가 다른 준비상태 큐로 이동할 수 없는 다단계 큐 기법을 준비상태 큐 사이를 이동할 수 있도록 개선한 기법 • 특정 큐에서 오래 기다린 프로세스나 I/O 작업 주기가 큰 프로세스 또는 Foreground 큐에 있는 프로세스는 우선순위가 높은 단계의 준비 큐로 이동시키고, CPU의 점유 시간이 긴 작업은 우선순위가 낮은 하위 단계의 준비 큐로 이동시킴 • 마지막 단계 큐에서는 작업이 완료될 때까지 RR 스케줄링 기법을 사용함

핵심 14.5, 13.3, 12.5, 11.8, 11.6, 11.3, 10.9, 10.3, 09.8, 08.9, 08.5, 08.3, 07.5, 06.9, 06.5, 05.5, 05.4, 04.9, 03.5

162 임계 구역/상호 배제

임계 구역(Critical Section)

- 다중 프로그래밍 운영체제에서 여러 개의 프로세스가 공유하는 데이터 및 자원에 대하여 어느 한 시점에서는 하나의 프로세스만 자원 또는 데이터를 사용하도록 지정된 공유 자원(영역)이다.
- 임계 구역에는 하나의 프로세스만 접근할 수 있으며, 해당 프로세스가 자원을 반납한 후에만 다른 프로세스가 자원이나 데이터를 사용할 수 있다.
- 임계 구역 내로 프로세스가 진입하는 것을 허용하는 것은 운영체제의 제어 권한이다.
- 특정 프로세스가 독점해서는 안되며, 임계 구역 내에서의 작업은 신속하게 진행되어야 한다.
- 임계 구역에서는 프로세스가 무한 루프에 빠지지 않도록 해야 한다.

상호 배제(Mutual Exclusion)

- 특정 프로세스가 공유 자원을 사용하고 있을 경우 다른 프로세스가 해당 공유 자원을 사용하지 못하게 제어하는 기법이다.
- 여러 프로세스가 동시에 공유 자원을 사용하려 할 때 각 프로세스가 번갈아가며 공유 자원을 사용하도록 하는 것으로 임계 구역을 유지하는 기법이다.
- 상호 배제 기법을 구현하기 위한 방법에는 소프트웨어적인 방법과 하드웨어적인 방법이 있다.

소프트웨어적 구현 방법	데커(Deker) 알고리즘, 피터슨(Peterson) 알고리즘, 빵집 알고리즘
하드웨어적 구현 방법	Test & Set 기법, Swap 명령어 기법

핵심 09.5, 06.3, 03.8, 03.3, 01.6, 01.3, 99.6

163 세마포어(Semaphore)

- 각 프로세스에 제어신호를 전달하여 순서대로 작업을 수행하도록 하는 기법이다.
- E.J.Dijkstra가 제안하였으며, 사용되는 연산에는 P와 V, 초기치 연산이 있고, 상호 배제의 원리를 보장한다.
- S는 P와 V 연산으로만 접근 가능한 세마포어 변수로, 공유 자원의 개수를 나타내며 0 또는 1이나 0 또는 양의 값을 가질 수 있다.
- P 연산 : 자원을 사용하려는 프로세스들의 진입 여부를 자원의 개수(S)를 통해 결정하는 것으로, Wait 동작이라고 함
- V 연산 : 대기중인 프로세스를 깨우는 신호(Wake Up)로서, Signal 동작이라고 함

핵심 12.3, 10.5, 09.3, 08.3, 06.5, 05.5, 04.5, 03.8, 02.5, 01.9, 01.6, 99.10

164 모니터(Monitor)

- 동기화를 구현하기 위한 특수 프로그램 기법으로 특정 공유 자원을 프로세스에게 할당하는 데 필요한 데이터와 이 데이터를 처리하는 프로시저로 구성된다.
- 자료 추상화와 정보 은폐 개념을 기초로 하며 공유 자원을 할당하기 위한 병행성 구조로 이루어져 있다.
- 모니터 내의 공유 자원을 사용하려면 프로세스는 반드시 모니터의 진입부를 호출해야 한다.
- 외부의 프로시저는 직접 액세스할 수 없으며, 모니터의 경계에서 상호 배제가 시행된다.
- 한 순간에 하나의 프로세스만 진입하여 자원을 사용할 수 있다.
- 모니터에서는 Wait와 Signal 연산이 사용된다.

핵심 12.8, 11.6, 11.3, 10.9, 08.9, 08.5, 07.3, 06.5, 04.9, 04.5, 04.3, 03.8, 03.5, 02.9, 02.3, 01.9, 01.6, 00.7, 00.5, 99.8, 99.4

165 교착 상태(Deadlock)

정의	상호 배제에 의해 나타나는 문제점으로, 둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상

필요 충분 조건	• 상호 배제(Mutual Exclusion) : 한 번에 한 개의 프로세스만이 공유 자원을 사용할 수 있어야 함 • 점유와 대기(Hold & Wait) : 최소한 하나의 자원을 점유하고 있으면서 다른 프로세스에 할당되어 사용되고 있는 자원을 추가로 점유하기 위해 대기하는 프로세스가 있어야 함 • 비선점(Non-preemptive) : 다른 프로세스에 할당된 자원은 사용이 끝날 때까지 강제로 빼앗을 수 없어야 함 • 환형 대기(Circular Wait) : 공유 자원과 공유 자원을 사용하기 위해 대기하는 프로세스들이 원형으로 구성되어 있어 자신에게 할당된 자원을 점유하면서 앞이나 뒤에 있는 프로세스의 자원을 요구해야 함

00.3, 99.10

14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.5, 12.3, 11.8, 10.5, 10.3, 09.8, 09.5, 08.3, 07.5, 05.9, 05.4, 04.9, 03.3, 02.3, 00.5,

핵심 166 교착 상태 해결 방법

• 예방 기법(Prevention) : 교착 상태가 발생되지 않도록 사전에 시스템을 제어하는 방법으로, 교착 상태 발생의 4가지 조건 중에서 어느 하나를 제거(부정)함으로써 수행됨

상호 배제 부정	한 번에 여러 개의 프로세스가 공유 자원을 사용할 수 있도록 하는 것으로 실현은 불가능함
점유 및 대기 부정	프로세스가 실행되기 전 필요한 모든 자원을 할당하여 프로세스 대기를 없애거나 자원이 점유되지 않은 상태에서만 자원을 요구하도록 함
비선점 부정	자원을 점유하고 있는 프로세스가 다른 자원을 요구할 때 점유하고 있는 자원을 반납하고, 요구한 자원을 사용하기 위해 기다리게 함
환형 대기 부정	자원을 선형 순서로 분류하여 고유 번호를 할당하고, 각 프로세스는 현재 점유한 자원의 고유 번호보다 앞이나 뒤 어느 한쪽 방향으로만 자원을 요구하도록 함

• 회피 기법(Avoidance) : 교착 상태가 발생할 가능성을 배제하지 않고, 교착 상태가 발생하면 적절히 피해나가는 방법으로, 주로 은행원 알고리즘(Banker's Algorithm)이 사용됨

은행원 알고리즘	• Dijkstra가 제안한 것으로, 은행에서 모든 고객의 요구가 충족되도록 현금을 할당하는 데서 유래한 기법 • 각 프로세스에게 자원을 할당하여 교착 상태가 발생하지 않으며 모든 프로세스가 완료될 수 있는 상태를 안전 상태, 교착 상태가 발생할 수 있는 상태를 불안전 상태라고 함

• 발견 기법(Detection) : 시스템에 교착 상태가 발생했는지 점검하여 교착 상태에 있는 프로세스와 자원을 발견하는 것으로, 교착 상태 발견 알고리즘과 자원 할당 그래프 등이 사용됨

• 회복 기법(Recovery) : 교착 상태를 일으킨 프로세스를 종료하거나 교착 상태의 프로세스에 할당된 자원을 선점하여 프로세스나 자원을 회복하는 것

07.3, 06.9, 06.5,

14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.5, 08.3, 07.5,

핵심 167 기억장치 관리 전략

• 반입(Fetch) 전략 : 보조기억장치에 보관중인 프로그램이나 데이터를 언제 주기억장치로 적재할 것인지를 결정하는 전략

요구 반입	실행중인 프로그램이 특정 프로그램이나 데이터 등의 참조를 요구할 때 적재하는 방법
예상 반입	실행중인 프로그램에 의해 참조될 프로그램이나 데이터를 미리 예상하여 적재하는 방법

• 배치(Placement) 전략 : 새로 반입되는 프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정하는 전략

최초 적합 (First Fit)	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치시키는 방법
최적 적합 (Best Fit)	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 적게 남기는 분할 영역에 배치시키는 방법
최악 적합 (Worst Fit)	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 많이 남기는 분할 영역에 배치시키는 방법

• 교체(Replacement) 전략 : 주기억장치의 모든 영역이 이미 사용중인 상태에서 새로운 프로그램이나 데이터를 주기억장치에 배치하려고 할 때, 이미 사용되고 있는 영역 중에서 어느 영역을 교체하여 사용할 것인지를 결정하는 전략으로, FIFO, OPT, LRU, LFU, NUR, SCR 등이 있음

핵심 168 주기억장치 할당 기법

04.5, 02.5, 00.10, 99.6, 99.4

단일 분할 할당 기법

- 주기억장치를 운영체제 영역과 사용자 영역으로 나누어 한 순간에는 오직 한 명의 사용자만이 주기억장치의 사용자 영역을 사용하는 기법
- **오버레이(Overlay) 기법** : 주기억장치보다 큰 사용자 프로그램을 실행하기 위한 기법으로, 보조기억장치에 저장된 하나의 프로그램을 여러 개의 조각으로 분할한 후 필요한 조각을 차례로 주기억장치에 적재하여 프로그램을 실행함
- **스와핑(Swapping) 기법** : 하나의 프로그램 전체를 주기억장치에 할당하여 사용하다 필요에 따라 다른 프로그램과 교체하는 기법

다중 분할 할당 기법

- **고정 분할 할당** : 프로그램을 할당하기 전에 운영체제가 주기억장치의 사용자 영역을 여러 개의 고정된 크기로 분할하고, 준비상태 큐에서 준비중인 프로그램을 각 영역에 할당하여 수행하는 기법
- **가변 분할 할당** : 고정 분할 할당 기법의 단편화를 줄이기 위한 것으로, 미리 주기억장치를 분할해 놓는 것이 아니라 프로그램을 주기억장치에 적재하면서 필요한 만큼의 크기로 영역을 분할하는 기법

핵심 169 단편화/단편화 해결 방법

10.9, 10.5, 09.3, 05.5, 03.8, 03.3, 02.9, 02.3, 00.10, 00.7, 00.5, 99.10

단편화

- 분할된 주기억장치에 프로그램를 할당하고 반납하는 과정을 반복하면서 사용되지 않고 남는 기억장치의 빈 공간 조각을 의미한다.
- **내부(Internal) 단편화** : 분할된 영역이 할당될 프로그램의 크기보다 크기 때문에 프로그램이 할당된 후 사용되지 않고 남아 있는 빈 공간
- **외부(External) 단편화** : 분할된 영역이 할당될 프로그램의 크기보다 작기 때문에 프로그램이 할당될 수 없어 사용되지 않고 빈 공간으로 남아 있는 분할된 전체 영역

단편화 해결 방법

- **통합(Coalescing) 기법** : 주기억장치 내에 인접해 있는 단편화된 공간을 하나의 공간으로 통합하는 작업
- **압축(Compaction, 집약) 기법, 쓰레기 수집** : 주기억장치 내에 분산되어 있는 단편화된 빈 공간을 결합하여 하나의 큰 가용 공간을 만드는 작업으로, 여러 위치에 분산된 단편화된 빈 공간을 주기억장치의 한쪽 끝으로 옮겨서 큰 기억 공간을 만듦

핵심 170 가상 기억장치

14.5, 14.3, 11.6, 11.3, 10.3, 08.5, 08.3, 05.9, 05.4, 05.3, 04.3, 02.5, 01.3, 99.10, 99.8

- 보조기억장치(하드디스크)의 일부를 주기억장치처럼 사용하는 것으로, 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법이다.
- 주기억장치의 용량보다 큰 프로그램을 실행하기 위해 사용한다.
- 가상 기억장치에 저장된 프로그램을 실행하려면 가상 기억장치의 주소를 주기억장치의 주소로 바꾸는 주소 변환 작업이 필요하다.
- 가상 기억장치에서 프로그램이 갖는 연속적인 주소가 실제 기억장치에서는 연속적일 필요는 없다.
- 블록 단위로 나누어 사용하므로 연속 할당 방식에서 발생할 수 있는 단편화를 해결할 수 있다.
- 주기억장치의 이용률과 다중 프로그래밍의 효율을 높일 수 있다.
- 가상 기억장치 구현 기법

페이징 (Paging) 기법	• 가상 기억장치에 보관되어 있는 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 나눠진 프로그램(페이지)을 동일하게 나눠진 주기억장치의 영역(페이지 프레임)에 적재시켜 실행하는 기법 • 주소 변환을 위해서 페이지의 위치 정보를 가지고 있는 페이지 맵 테이블이 필요함 • 동일한 크기로 나누므로 외부 단편화는 발생하지 않으나 내부 단편화는 발생할 수 있음
세그먼테이션 (Segmentation) 기법	• 가상 기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행시키는 기법 • 프로그램을 배열이나 함수 등과 같은 논리적인 크기로 나눈 단위를 세그먼트라고 하며, 각 세그먼트는 고유한 이름과 크기를 갖고 있음 • 다른 세그먼트에게 할당된 영역을 침범할 수 없으며, 이를 위해 기억장치 보호키(Storage Protection Key)가 필요함 • 주소 변환을 위해서 세그먼트의 위치 정보를 가지고 있는 세그먼트 맵 테이블이 필요함 • 기억장치의 사용자 관점을 보존하는 기억장치 관리 기법임 • 논리적인 크기로 나누므로 내부 단편화는 발생하지 않으나 외부 단편화는 발생할 수 있음

07.5, 07.3, 06.9

핵심 171 14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.6, 11.3, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.5, 08.3, 07.9,

171 페이지 교체 알고리즘

페이지 부재가 발생했을 때 가상 기억장치의 필요한 페이지를 주기억장치에 적재해야 하는데, 이때 주기억장치의 모든 페이지 프레임이 사용중이면 어떤 페이지 프레임을 선택하여 교체할지 결정하는 기법이다.

OPT(OPTimal replacement, 최적 교체)	• 앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법 • 각 페이지의 호출 순서와 참조 상황을 미리 예측해야 하므로 실현 가능성이 희박함
FIFO(First In First Out)	• 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법 • 이해하기 쉽고, 프로그래밍 및 설계가 간단함
LRU(Least Recently Used)	• 최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법 • 각 페이지마다 계수기나 스택을 두어 현 시점에서 가장 오랫동안 사용하지 않은, 즉 가장 오래 전에 사용된 페이지를 교체함
LFU(Least Frequently Used)	• 사용 빈도가 가장 적은 페이지를 교체하는 기법 • 프로그램 실행 초기에 많이 사용된 페이지가 그 후로 사용되지 않을 경우에도 프레임을 계속 차지할 수 있음
NUR (Not Used Recently)	• LRU와 비슷한 알고리즘으로, 최근에 사용하지 않은 페이지를 교체하는 기법 • 최근의 사용 여부를 확인하기 위해서 각 페이지마다 참조 비트(Reference Bit)와 변형 비트(Modified Bit, Dirty Bit)가 사용됨
SCR(Second Chance Replacement)	가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지의 교체를 방지하기 위한 것으로, FIFO 기법의 단점을 보완하는 기법

핵심 172 13.8, 13.6, 13.3, 12.8, 12.3, 10.9, 10.5, 09.8, 09.3, 08.9, 08.5, 07.5, 07.3, 04.9, 02.5, 99.6

172 페이지 크기

페이징 기법을 사용하면 프로그램을 동일한 크기로 나눈 페이지 단위로 나누게 되는데, 페이지의 크기에 따라 시스템에 미치는 영향이 다르다. 페이지 크기에 따른 특징은 다음과 같다.

페이지 크기가 작을 경우	• 페이지의 단편화가 감소되고, 한 개의 페이지를 주기억장치로 이동하는 시간이 줄어듦 • 프로세스(프로그램) 수행에 필요한 내용만 주기억장치에 적재될 확률이 높아 효율적인 워킹셋을 유지할 수 있음 • Locality(국부성)에 더 일치할 수 있기 때문에 기억장치 효율이 높아짐 • 페이지 정보를 갖는 페이지 맵 테이블의 크기가 커지고, 매핑 속도가 늦어짐 • 디스크 접근 횟수가 많아져서 전체적인 입 · 출력 시간은 늘어남 • 우수한 Working Set을 가질 수 있음 • 단편화가 줄어 공간 낭비가 줄어듦
페이지 크기가 클 경우	• 페이지 정보를 갖는 페이지 맵 테이블의 크기가 작아지고, 매핑 속도가 빨라짐 • 디스크 접근 횟수가 줄어들어 전체적인 입 · 출력의 효율성이 증가됨 • 페이지의 단편화가 증가되고, 한 개의 페이지를 주기억장치로 이동하는 시간이 늘어남 • 프로그램 수행에 불필요한 내용까지도 주기억장치에 적재될 수 있음 • 단편화가 늘어 공간 낭비가 늘어남

핵심 173 13.3, 12.5, 08.3, 07.5, 07.3, 06.5, 06.3, 05.5, 03.8, 03.5, 03.3, 02.9, 02.5, 02.3, 01.6, 00.7, 00.5, 00.3, 99.8

173 국부성(Locality, 구역성)

- 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다는 이론이다.
- 스래싱을 방지하기 위한 워킹 셋 이론의 기반이 된다.
- 프로세스가 집중적으로 사용하는 페이지를 알아내는 방법 중 하나로, 가상 기억장치 관리의 이론적인 근거가 된다.
- Locality 종류

시간 구역성 (Temporal Locality)	• 프로세스가 실행되면서 하나의 페이지를 일정 시간 동안 집중적으로 액세스하는 현상 • 시간 구역성이 이루어지는 기억장소 : Loop(반복, 순환), 스택(Stack), 부 프로그램(Sub Routine), Counting(1씩 증감), Totaling(집계)에 사용되는 변수(기억장소)
공간 구역성 (Spatial Locality)	• 프로세스 실행 시 일정 위치의 페이지를 집중적으로 액세스하는 현상 • 공간 구역성이 이루어지는 기억장소 : 배열 순회(Array Traversal), 순차적 코드의 실행, 프로그래머들이 관련된 변수(데이터를 저장할 기억장소)들을 서로 근처에 선언하여 할당되는 기억장소, 같은 영역의 변수 참조

핵심 174 14.5, 12.8, 11.8, 11.3, 10.3, 09.5, 08.5, 06.9, 05.3, 04.5, 02.3, 01.9, 00.10, 99.10, 99.6

174 워킹 셋/페이지 부재

- 워킹 셋(Working Set)
  - 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합이다.
  - 워킹 셋을 주기억장치에 상주시킴으로써 페이지 부재 및 페이지 교체현상을 줄일 수 있다.
  - 주기억장치 내의 워킹 셋을 감소시키면 주기억장치의 페이지 프레임 공간을 적게 차지하면서 워킹 셋을 사용할 수 있기 때문에 스래싱이 감소하게 된다.
  - 윈도우의 크기, 즉 워킹 셋의 설정 시간 간격이 증가하면 주기억장치에 유지되는 워킹 셋은 커지고, 설정 시간 간격이 감소하면 워킹 셋은 작아진다.
  - 워킹 셋으로 인해 오버헤드가 발생되므로 최소한의 워킹 셋을 올려놓아야 한다.
- 페이지 부재(Page Fault)는 프로세스 실행 시 참조할 페이지가 주기억장치에 없는 현상이며, 페이지 부재 빈도(PFF; Page Fault Frequency)는 페이지 부재가 일어나는 횟수를 의미한다.
- **페이지 부재 빈도 조절 방식** : 페이지 부재율(Page Fault Rate)에 따라 주기억장치에 있는 페이지 프레임의 수를 늘리거나 줄여 페이지 부재율을 적정 수준으로 유지하는 방식

핵심 175 12.8, 12.3, 09.5, 09.3, 08.3, 07.9, 06.9, 06.5, 05.4, 04.5, 03.8, 03.5, 03.3, 02.3, 01.9, 01.3, 00.7, 00.5, 00.3, 99.4

175 스래싱(Thrashing)

- 프로세스의 처리 시간보다 페이지 교체 시간이 더 많아지는 현상으로 페이지 부재가 계속 늘어나고, 기억장치 접근 시간이 지속된다.
- 다중 프로그래밍 시스템이나 가상 기억장치를 사용하는 시스템에서 하나의 프로세스 수행 과정중 자주 페이지 부재가 발생함으로 인해 나타나는 현상으로 전체 시스템의 성능이 저하된다.
- 다중 프로그래밍의 정도가 높아짐에 따라 CPU의 이용률은 어느 특정 시점까지는 높아지지만 다중 프로그래밍의 정도가 더욱 커지면 스래싱이 나타나고, CPU의 이용률은 급격히 감소된다.
- CPU 이용률을 높이고, 스래싱 현상을 방지하려면 다중 프로그래밍의 정도를 적정 수준으로 유지하고, 페이지 부재 빈도(Page Fault Frequency)를 조절하여 사용하며, Working Set을 유지해야 한다.

06.9, 06.5

핵심 176 14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 11.8, 11.6, 11.3, 10.9, 10.3, 09.8, 09.5, 08.9, 08.5, 08.3, 07.9, 07.5, 07.3,

176 디스크 스케줄링

사용할 데이터가 디스크 상의 여러 곳에 저장되어 있을 경우 데이터를 액세스하기 위해 디스크 헤드가 움직이는 경로를 결정하는 기법이다.

FCFS (First-Come First-Service)	• 가장 간단한 스케줄링으로, 디스크 대기 큐에 가장 먼저 들어온 트랙에 대한 요청을 먼저 서비스하는 기법 • 디스크 대기 큐에 들어온 순서대로 서비스하기 때문에 더 높은 우선순위의 요청이 입력되어도 순서가 바뀌지 않아 공평성이 보장
SSTF (Shortest Seek Time First)	• 탐색 거리가 가장 짧은 트랙에 대한 요청을 먼저 서비스하는 기법 • 현재 헤드 위치에서 가장 가까운 거리에 있는 트랙으로 헤드를 이동시킴 • FCFS보다 처리량이 많고, 평균 탐색 시간이 짧음 • 응답 시간의 편차가 크다. 즉 헤드에서 멀리 떨어진 요청은 기아 상태가 발생함 • 처리량이 많은 일괄 처리 시스템에 유용함
SCAN	• SSTF가 갖는 탐색 시간의 편차를 해소하기 위한 기법 • 현재 헤드의 위치에서 진행 방향이 결정되면 탐색 거리가 짧은 순서에 따라 그 방향의 모든 요청을 서비스하고, 끝까지 이동한 후 역방향의 요청 사항을 서비스함
C-SCAN (Circular SCAN)	• 항상 바깥쪽에서 안쪽으로 움직이면서 가장 짧은 탐색 거리를 갖는 요청을 서비스하는 것으로, 가장 안쪽과 가장 바깥쪽의 실린더에 대한 차별 대우를 없앤 기법 • 헤드는 트랙의 바깥쪽에서 안쪽으로 한 방향으로만 움직이며 서비스하여 끝까지 이동한 후, 안쪽에 더 이상의 요청이 없으면 헤드는 가장 바깥쪽의 끝으로 이동한 후 다시 안쪽으로 이동하면서 요청을 서비스함
N-step SCAN	SCAN 기법을 기초로 하며, 어떤 방향의 진행이 시작될 당시에 대기중이던 요청들만 서비스하고, 진행 도중 도착한 요청들은 한데 모아서 다음의 반대 방향 진행 때 서비스하는 기법

01.9, 01.3, 99.10

핵심 177 14.8, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 11.8, 11.3, 10.5, 10.3, 09.8, 09.3, 08.9, 08.5, 07.3, 06.9, 05.5, 05.4, 05.3, 02.5,

177 파일/파일 시스템/파일 디스크립터

파일

- 파일은 사용자가 작성한 서로 관련 있는 레코드의 집합체를 의미한다.
- 프로그램 구성의 기본 단위가 되며, 보조기억장치에 저장된다.

- 각 파일마다 위치, 크기, 작성 시기 등의 여러 속성을 가지고 있다.
- 파일 특성을 결정하는 기준
  - 소멸성(Volatility) : 파일을 추가하거나 제거하는 작업의 빈도수
  - 활성률(Activity) : 프로그램이 한 번 수행되는 동안 처리되는 레코드 수의 백분율((수행 레코드 수/전체 레코드 수) × 100)
  - 크기(Size) : 파일에 저장되어 있는 정보량

파일 시스템의 기능

- 파일 시스템은 파일의 저장, 액세스, 공유, 보호 등 보조기억장치에서의 파일을 총괄하는 파일 관리 기술이다.
- 사용자가 물리적 장치 이름 대신에 기호화된 이름을 사용할 수 있도록 한다.
- 사용자와 보조기억장치 사이에서 인터페이스를 제공한다.
- 사용자가 파일을 생성, 수정, 제거할 수 있도록 한다.
- 적절한 제어 방식을 통해 타인의 파일을 공동으로 사용할 수 있도록 한다.
- 사용자가 적합한 구조로 파일을 구성할 수 있도록 한다.
- 불의의 사태에 대비하여 파일의 예비와 복구 등의 기능을 제공한다.
- 파일을 안전하게 사용할 수 있도록 하고, 파일이 보호되어야 한다.
- 파일 공유를 위해서 판독만 허용, 기록만 허용, 수행만 허용 또는 이들을 여러 형태로 조합한 것 등 여러 종류의 액세스 제어 방법을 제공한다.
- 파일에 대한 암호화와 해독 기능을 제공한다.

파일 디스크립터(File Descriptor)

- 파일을 관리하기 위한 시스템(운영체제)이 필요로 하는 파일에 대한 정보를 갖고 있는 제어 블록으로, 파일 제어 블록(FCB)이라고도 한다.
- 보통 파일 디스크립터는 보조기억장치 내에 저장되어 있다가, 해당 파일이 Open될 때 주기억장치로 옮겨진다.
- 파일 시스템이 관리하므로 사용자가 직접 참조할 수 없다.
- 파일 디스크립터는 파일마다 독립적으로 존재하며, 시스템에 따라 다른 구조를 가질 수 있다.
- 파일 디스크립터의 정보 : 파일 이름, 보조기억장치에서의 파일 위치, 파일 구조, 보조기억장치의 유형, 액세스 제어 정보, 파일 유형, 생성 날짜와 시간, 제거 날짜와 시간, 최종 수정 날짜 및 시간, 액세스한 횟수 등

핵심 178 순차 파일(Sequential File), 순차 접근 방식

10.5, 05.4, 99.8

- 레코드를 논리적인 처리 순서에 따라 연속된 물리적 저장공간에 기록하는 것이다.
- 급여 관리 등과 같이 변동 사항이 크지 않고 기간별로 일괄 처리를 주로 하는 경우에 적합하다.
- 주로 순차 접근이 가능한 자기 테이프에서 사용한다.
- 장점
  - 파일의 구성이 용이하며 기억공간의 이용 효율 높다.
  - 레코드를 기록할 때 사용한 키 순서대로 레코드를 처리할 경우 다른 편성법보다 처리 속도가 빠르다.
  - 어떠한 기억 매체에서도 실현 가능하다.
- 단점 : 파일에 새로운 레코드를 삽입하거나 삭제하는 경우 시간이 많이 소요되며, 검색 효율이 낮음

핵심 179 직접 파일/색인 순차 파일

13.6, 09.8, 08.3, 06.9, 05.9, 05.3, 03.8, 02.9, 00.3

직접 파일(Direct File)

- 파일을 구성하는 레코드를 임의의 물리적 저장공간에 기록하는 것이다.
- 레코드에 특정 기준으로 키가 할당되며, 해싱 함수(Hashing Function)를 이용하여 이 키에 대한 보조기억장치의 물리적 상대주소를 계산한 후 해당하는 주소에 레코드를 저장한다.
- 레코드는 해싱 함수에 의해 계산된 물리적 주소를 통해 접근이 가능하다.
- 임의 접근이 가능한 자기 디스크나 자기 드럼을 사용한다.
- 장점 : 파일의 각 레코드에 직접 접근하거나 기록할 수 있음, 접근 시간이 빠르고, 레코드의 삽입, 삭제, 갱신이 용이함

- 단점 : 레코드의 주소 변환 과정이 필요하며, 이 과정으로 인해 시간이 소요됨, 기억공간의 효율이 저하됨

색인 순차 파일(Indexed Sequential File)

- 순차 파일과 직접 파일에서 지원하는 편성 방법이 결합된 형태이다.
- 각 레코드를 키 값 순으로 논리적으로 저장하고, 시스템은 각 레코드의 실제주소가 저장된 색인을 관리한다.
- 레코드를 참조하려면 색인을 탐색한 후 색인이 가리키는 포인터(주소)를 사용하여 직접 참조할 수 있다.
- 기본 영역, 색인 영역, 오버 플로 영역으로 구성되며, 색인 영역은 트랙 색인 영역, 실린더 색인 영역, 마스터 색인 영역으로 분류된다.
- 장점 : 순차 처리와 임의 처리가 모두 가능, 효율적인 검색 가능, 삭제, 삽입, 갱신이 용이함
- 단점 : 인덱스를 위한 별도의 기억공간이 필요하며, 접근 시간이 직접 파일보다 느림

핵심 180 디렉터리의 구조

14.8, 14.5, 12.8, 12.5, 12.3, 11.6, 11.3, 10.9, 09.5, 09.3, 09.9, 04.9, 04.5, 01.3, 00.3, 99.4

1단계 디렉터리	• 가장 간단하고, 모든 파일이 하나의 디렉터리 내에 위치하여 관리되는 구조 • 모든 파일들이 유일한 이름을 가지고 있어야 함 • 모든 파일이 같은 디렉터리 내에 유지되므로 이해가 용이하지만, 파일의 수나 사용자의 수가 증가하면 파일 관리가 복잡해짐
2단계 디렉터리	• 중앙에 마스터 파일 디렉터리가 있고, 그 아래에 사용자별로 서로 다른 파일 디렉터리가 있는 2계층 구조로, 디렉터리 탐색은 포인터에 의해 계층적으로 이루어짐 • 마스터 파일 디렉터리는 사용자 파일 디렉터리를 관리하고, 사용자 파일 디렉터리는 사용자별 파일을 관리함 • 서로 다른 디렉터리에서는 동일한 파일 이름을 사용할 수 있음
트리 디렉터리	• 하나의 루트 디렉터리와 여러 개의 종속(서브) 디렉터리로 구성된 구조 • DOS, Windows, UNIX 등의 운영체제에서 사용되는 디렉터리 구조로, 디렉터리 탐색은 포인터에 의해 계층적으로 이루어짐 • 서로 다른 디렉터리에서 동일한 이름의 파일이나 디렉터리를 생성할 수 있음 • 디렉터리의 생성과 파괴가 비교적 용이함
비순환 그래프 디렉터리	• 하위 파일이나 하위 디렉터리를 공동으로 사용할 수 있는 것으로, 사이클이 허용되지 않는 구조 • 하나의 파일이나 디렉터리가 여러 개의 경로 이름을 가질 수 있음 • 디스크 공간을 절약할 수 있으며 공유된 파일을 삭제할 경우 고아 포인터(Dangling Pointer)가 발생할 수 있음
일반적인 그래프 디렉터리	• 트리 구조에 링크(Link)를 첨가시켜 순환을 허용하는 그래프 구조 • 디렉터리와 파일 공유에 완전한 융통성이 있음 • 탐색 알고리즘이 간단하며, 원하는 파일에 접근하기가 용이함 • 불필요한 파일을 제거하여 사용 공간을 늘리기 위한 참조 계수기가 필요함

핵심 181 디스크 공간 할당 방법

05.5, 04.5, 00.7, 99.8, 99.6, 99.4

- 연속 할당 : 파일을 디스크의 연속된 기억공간에 할당하는 방법으로, 생성되는 파일 크기만큼의 공간이 있어야 하고, 액세스 시간이 감소되며 디렉터리 구현이 용이함
- 불연속 할당 : 파일의 크기가 변경될 경우 구현이 어려운 연속 할당의 단점을 보완하기 위한 것으로, 디스크 공간을 일정 단위로 나누어 할당하는 기법

섹터 단위 할당	하나의 파일이 디스크의 섹터 단위로 분산되어 할당되는 방법으로, 하나의 파일에 속하는 섹터들이 연결 리스트(Linked List)로 구성되어 있음
블록 단위 할당	• 하나의 파일이 연속된 여러 개의 섹터를 묶은 블록 단위로 할당되는 방법 • 블록 단위 할당에는 블록 체인 할당, 색인 블록 체인 할당, 블록 지향 파일 사상 기법이 있음

핵심 182 자원 보호 기법/파일 보호 기법

14.5, 14.3, 13.8, 13.6, 13.3, 12.3, 11.8, 11.6, 11.3, 10.3, 09.8, 08.5, 07.9, 06.3, 03.5, 01.6, 01.3, 99.6

자원 보호 기법

- 접근 제어 행렬(Access Control Matrix) : 자원 보호의 일반적인 모델로, 객체에 대한 접근 권한을 행렬로써 표시한 기법. 행(Row)은 영역(사용자, 프로세스), 열(Column)은 객체, 각 항은 접근 권한의 집합으로 구성됨
- 전역 테이블(Global Table) : 가장 단순한 구현 방법으로, 3개의 순서쌍인 영역, 객체, 접근 권한의 집합을 목록 형태로 구성한 기법
- 접근 제어 리스트(Access Control List) : 접근 제어 행렬에 있는 각 열, 즉 객체를 중심으로 접근 리스트를 구성한 기법

- 권한(자격) 리스트(Capability List) : 접근 제어 행렬에 있는 각 행, 즉 영역을 중심으로 권한 리스트를 구성한 기법으로, 각 영역에 대한 권한은 객체와 그 객체에 허용된 연산자로 구성됨

파일 보호 기법

- 파일의 명명(Naming) : 접근하고자 하는 파일 이름을 모르는 사용자를 접근 대상에서 제외시키는 기법
- 비밀번호(Password, 암호)
  - 각 파일에 판독 암호와 기록 암호를 부여하여 암호를 아는 사용자에게만 접근을 허용하는 기법이다.
  - 패스워드는 숫자와 문자를 혼합하여 사용하는 것이 바람직하며 추측 가능한 전화번호, 생년월일 등으로 구성하지 않는 것이 좋다.
  - 패스워드는 길게 작성하는 것이 좋다.
  - 패스워드는 자주 변경하는 것이 좋다.
- 접근 제어(Access Control) : 사용자에 따라 공유 데이터에 접근할 수 있는 권한을 제한하는 방법

핵심 183 보안 유지 기법

10.9, 08.3, 06.9, 05.5, 05.4, 04.5, 03.3, 01.6, 00.3

보안을 유지할 수 있는 기법에는 외부 보안, 사용자 인터페이스 보안, 내부 보안이 있다.

외부 보안	• 시설 보안 : 천재 지변이나 외부 침입자로부터의 보안 • 운용 보안 : 전산소 관리 및 경영자들의 정책과 통제에 의해 이루어지는 보안
사용자 인터페이스 보안	운영체제가 사용자의 신원을 확인한 후 권한이 있는 사용자에게만 시스템의 프로그램과 데이터를 사용할 수 있게 하는 보안 기법
내부 보안	하드웨어나 운영체제의 내장된 보안 기능을 이용하여 시스템의 신뢰성을 유지하고, 보안 문제를 해결하는 기법

핵심 184 암호화 기법

14.8, 05.5, 02.3, 01.9, 00.3, 99.10

- 데이터를 보낼 때 송신자가 지정한 수신자 이외에는 그 내용을 알 수 없도록 평문을 암호문으로 변환하는 것이다.
- 비밀키 시스템(Private Key System, 개인키 시스템) : 동일한 키로 데이터를 암호화하고, 해독(복호화)하는 대칭 암호화 기법으로, 대표적인 암호화 방식에는 DES가 있음
- 공용키 시스템(Public Key System, 공개키 시스템) : 서로 다른 키로 데이터를 암호화하고, 해독하는 비대칭 암호화 기법으로, 대표적인 암호화 방식에는 RSA가 있음

핵심 185 프로세서 연결 방식

13.3, 12.5, 00.3, 99.6

시분할 및 공유 버스	• 프로세서, 주변장치, 기억장치 등의 각종 장치들을 '버스'라는 단일 경로로 연결한 방식 • 장치 연결이 단순하고, 경제적일 뿐만 아니라 융통성이 있어 장치 추가가 용이함 • 한 시점에서는 한 장치가 버스를 점유하기 때문에 하나의 데이터만 전송할 수 있음 • 버스에 이상이 생기면 전체 시스템이 가동되지 않음 • 시스템 전체 전송량이 버스의 전송률에 의해 제한됨 • 둘 이상의 장치에서 버스를 사용하기 위해 경쟁하면 시스템 성능 효율이 저하됨
크로스바 교환 행렬	• 시분할 및 공유 버스 방식에서 버스의 수를 기억장치 수만큼 증가시켜 연결한 방식 • 각 기억장치마다 다른 경로를 사용할 수 있으며, 두 개의 서로 다른 기억장치를 동시에 참조할 수 있음 • 장치의 연결이 복잡해짐
하이퍼 큐브	• 다수의 프로세서들을 연결하는 방식으로 비교적 경제적인 방식임 • 4개의 프로세서를 2개씩 서로 이웃하게 연결한 사각형 모양의 2차원 하이퍼 큐브를 만들고, 2차원 하이퍼 큐브의 대응점을 각각 연결하여 3차원 하이퍼 큐브를 형성하고, 이런 형식으로 4차원, 5차원 … 하이퍼 큐브를 형성함 • 다수의 프로세서를 연결할 수 있으며, 확장성이 좋음 • 하나의 프로세서에 연결되는 다른 프로세서의 수(연결점)가 n개일 경우 프로세서는 총 2^n개가 필요함
다중 포트 기억장치	• 시분할 및 공유 버스 방식과 크로스바 교환 행렬 방식을 혼합한 형태의 방식 • 많은 수의 프로세서를 쉽게 연결할 수 있음 • 다양한 연결이 가능하며, 전송 시간이 비교적 느림

핵심 186 다중 처리기의 운영체제 구조

13.8, 13.6, 13.3, 12.3, 11.6, 11.3, 10.5, 09.8, 09.3, 08.3, 07.9, 05.9, 01.3, 99.8

주/종 처리기	• 하나의 프로세서를 Master(주프로세서)로 지정하고, 나머지들은 Slave(종프로세서)로 지정하는 구조 • 주프로세서가 고장나면 전체 시스템이 다운됨 • 주프로세서 : 입 · 출력과 연산 담당, 운영체제 수행 • 종프로세서 : 연산만 담당 • 주프로세서만 입 · 출력을 수행하는 비대칭 구조임
분리 실행 처리기	• 주/종 처리기의 비대칭성을 보완하여 각 프로세서가 독자적인 운영체제를 가지고 있도록 구성한 구조 • 각 프로세서에서 발생하는 인터럽트는 해당 프로세서에서 해결 • 각 프로세서가 독자적인 운영체제를 가지고 있기 때문에 한 프로세서가 고장나더라도 전체 시스템이 다운되지 않음
대칭적 처리기	• 여러 프로세서들이 완전한 기능을 갖춘 하나의 운영체제를 공유하여 수행하는 구조 • 가장 복잡한 구조를 가지고 있으나 가장 강력한 시스템임 • 여러 개의 프로세서가 동시에 수행될 수 있고, 시스템의 전반적인 정보를 통일적이고 일관성 있게 운영함

핵심 187 프로세서의 결합도

14.3, 12.8, 10.9, 09.5, 09.3, 08.5, 05.4, 04.5, 03.3, 02.5, 99.8

약결합 (Loosely Coupled) 시스템	• 각 프로세서마다 독립된 메모리를 가진 시스템으로, 분산 처리 시스템이라고도 함 • 둘 이상의 독립된 컴퓨터 시스템을 통신망(통신 링크)을 통하여 연결한 시스템 • 각 시스템마다 독자적인 운영체제를 가지고 있음 • 각 시스템은 독립적으로 작동할 수도 있고, 필요한 경우에는 상호 통신할 수도 있음 • 프로세서 간의 통신은 메시지 전달이나 원격 프로시저 호출을 통해서 이루어짐 • 각 시스템마다 독자적인 운영이 가능하므로 CPU 간의 결합력이 약함
강결합 (Tightly Coupled) 시스템	• 동일 운영체제하에서 여러 개의 프로세서가 하나의 메모리를 공유하여 사용하는 시스템으로, 다중(병렬) 처리 시스템이라고도 함 • 하나의 운영체제가 모든 프로세서와 시스템 하드웨어를 제어함 • 프로세서 간의 통신은 공유 메모리를 통해서 이루어짐 • 하나의 메모리를 사용하므로 CPU 간의 결합력이 강함

핵심 188 분산 처리 시스템의 목적/장 · 단점/투명성

01.9, 01.6, 01.3

14.8, 14.5, 14.3, 13.8, 12.5, 11.8, 11.6, 10.3, 09.8, 08.9, 08.5, 07.5, 07.3, 06.9, 05.5, 05.4, 05.3, 03.8, 02.9, 02.5, 02.3,

분산 처리 시스템의 목적/장 · 단점

- 목적 : 자원 공유, 연산 속도 향상, 신뢰도 향상, 컴퓨터 통신, 처리량 증가
- 장점 : 통신 용이, 장치 공유, 데이터 공유, 중앙 컴퓨터 과부하 줄임, 컴퓨터의 위치를 몰라도 자원 사용 가능, 시스템의 점진적 확장 가능 등
- 단점 : 중앙 집중형 시스템에 비해 소프트웨어의 개발이 어려움, 보안 문제 발생으로 보안 정책이 복잡함

분산 처리 시스템의 투명성

- 투명성(Transparency) : 어떠한 사실이 존재함에도 마치 투명하여 보이지 않는 것처럼, 사실의 존재 여부를 염두에 두지 않아도 되는 성질
- 여러 유형의 투명성을 통해 자원의 위치나 정보가 변경되더라도 사용자가 이를 인식하지 못하게 된다.
- 투명성의 종류

위치(Location) 투명성	사용자가 하드웨어나 소프트웨어와 같은 자원(정보 객체)의 물리적 위치를 모르더라도 자원에 접근할 수 있도록 함
이주(Migration) 투명성	사용자나 응용 프로그램의 동작에 영향을 받지 않고 시스템 내에 있는 자원을 이동할 수 있도록 함
복제(Replication) 투명성	자원의 복제를 사용자에게 통지할 필요 없이 자유로이 수행할 수 있음
병행(Concurrency) 투명성	자원의 위치를 모르더라도 다중 사용자들이 자원을 병행하여 처리하고, 공유할 수 있도록 함
접근(Access) 투명성	각 프로세서의 로그인 등과 같은 동작을 사용하여 지역이나 원격 자원에 접근할 수 있음
성능(Performance) 투명성	여러 부하에 대해 성능을 증가시키기 위하여 시스템을 재구성할 수 있도록 함
규모(Scaling) 투명성	시스템이나 응용 프로그램들이 시스템 구조나 응용 알고리즘에 대한 변경 없이 규모에 맞추어 확장할 수 있도록 함
고장(Failure) 투명성	사용자나 응용 프로그램이 하드웨어나 소프트웨어 구성 요소의 고장에도 불구하고 그들의 작업을 완료할 수 있도록 함

01.9, 01.6, 99.4

핵심 189 위상에 따른 분산 처리 시스템의 분류

14.8, 14.5, 13.6, 12.8, 12.3, 11.8, 11.3, 10.9, 10.5, 09.5, 08.3, 07.9, 07.5, 07.3, 06.5, 06.3, 04.9, 04.5, 03.5, 03.3, 02.9,

완전 연결 (Fully Connection)형	• 각 사이트들이 시스템 내의 다른 모든 사이트들과 직접 연결된 구조 • 사이트의 수가 n개이면 링크(연결) 수는 $\frac{n(n-1)}{2}$ 개임 • 기본 비용은 많이 들지만 통신 비용은 적게 들고, 신뢰성이 높음
부분 연결 (Partially Connection)형	• 시스템 내의 일부 사이트들 간에만 직접 연결된 형태로, 직접 연결되지 않은 사이트는 연결된 다른 사이트를 통해 통신하는 구조 • 기본 비용은 완전 연결형보다 적게 들고, 통신 비용은 완전 연결형보다 많이 소요됨 • 완전 연결형보다 신뢰성이 낮음
트리(Tree)/계층 (Hierarchy)형	• 분산 처리 시스템의 가장 대표적인 형태로, 각 사이트들이 트리 형태로 연결된 구조 • 기본 비용은 부분 연결형보다 적게 소요되고, 통신 비용은 트리의 깊이에 비례함 • 부모(상위) 사이트의 자식(하위) 사이트들은 그 부모 사이트를 통해 통신이 이루어짐 • 부모 사이트가 고장나면 그 자식 사이트들은 통신이 불가능함
스타(Star)형/ 성형	• 모든 사이트가 하나의 중앙 사이트에 Point-to-Point 형태로 연결되어 있고, 그 외의 다른 사이트와는 연결되어 있지 않은 구조 • 기본 비용은 사이트의 수에 비례하며, 통신 비용은 적게 소요됨 • 구조가 간단하고, 보수 및 관리가 용이함 • 중앙 사이트를 제외한 사이트의 고장이 다른 사이트에 영향을 미치지 않지만, 중앙 사이트가 고장날 경우 모든 통신이 단절됨 • 사이트의 증가에 따라 통신 회선도 증가함
링(Ring)형/환형	• 시스템 내의 각 사이트가 인접하는 다른 두 사이트와만 직접 연결된 구조 • 정보는 단방향 또는 양방향으로 전달될 수 있음 • 기본 비용은 사이트 수에 비례하고, 목적 사이트에 데이터를 전달하기 위해 링을 순환할 경우 통신 비용이 증가함
다중 접근 버스 연결(Multi Access Bus Connection)형	• 시스템 내의 모든 사이트들이 공유 버스에 연결된 구조 • 기본 비용은 사이트 수에 비례하고, 통신 비용은 일반적으로 저렴함 • 사이트의 고장은 다른 사이트의 통신에 영향을 주지 않지만, 버스의 고장은 전체 시스템에 영향을 줌 • 물리적 구조가 단순하고, 사이트의 추가, 삭제가 용이함 • 통신 회선 길이에 제한이 있음

05.9, 05.4, 05.3, 04.3

핵심 190 네트워크 운영체제/분산 운영체제

네트워크 운영체제	• 독자적인 운영체제를 가지고 있는 시스템을 네트워크로 구성한 것으로, 사용자가 원격 시스템으로 로그인하거나 원격 시스템으로부터 필요한 자원을 전달받아야 하는 방식 • 사용자는 시스템의 각 장치에 대해 알고 있어야 함 • 지역적으로 멀리 떨어져 있는 대규모 시스템에서 주로 사용함 • 설계와 구현이 쉽고, 자원 공유가 번거로움
분산 운영체제	• 하나의 운영체제가 모든 시스템 내의 자원을 관리하는 것으로, 원격에 있는 자원을 마치 지역 자원인 것과 같이 쉽게 접근하여 사용할 수 있는 방식 • 사용이 편리하고, 시스템 간 자원 공유가 용이함 • 하나의 운영체제가 시스템 전체를 관리해야 하므로 설계와 구현이 어려움 • 이주 – 데이터 이주(Data Migration) : 데이터를 요청한 사용자의 컴퓨터로 해당 데이터의 복사본을 전송시키는 방식 – 연산 이주(Computation Migration) : 요청한 데이터가 있는 컴퓨터에서 데이터를 처리하여 해당 결과를 요청한 컴퓨터에게 보내는 방식으로, 전송할 요청 데이터가 많을 경우 데이터를 전송시키는 것이 아니라 결과를 전송시키며 프로세스가 특정 사이트에 접근할 때 원격 프로시저 호출을 통해 이동함 – 프로세스 이주(Process Migration) : 프로세스의 전체 또는 일부를 다른 컴퓨터에서 실행되도록 하는 방식

14.8, 14.5, 14.3, 13.6, 12.3, 10.9, 10.3, 09.3, 08.3, 07.5, 06.5, 06.3, 04.5, 04.3, 01.6, 00.5, 99.10

핵심 191 UNIX의 특징

- 시분할 시스템을 위해 설계된 대화식 운영체제로, 소스가 공개된 개방형 시스템(Open System)이다.
- 대부분 C 언어로 작성되어 있어 이식성이 높으며 장치, 프로세스 간의 호환성이 높다.
- 크기가 작고 이해하기가 쉬우며, Multi-User(다중 사용자), Multi-Tasking(다중 작업)을 지원한다.
- 많은 네트워킹 기능을 제공하므로 통신망(Network) 관리용 운영체제로 적합하다.
- 트리 구조의 파일 시스템으로, 전문적인 프로그램 개발이 용이한다.
- 다양한 유틸리티 프로그램들이 존재한다.

04.9, 04.5, 04.3, 03.8, 02.9, 02.5, 02.3, 01.6, 01.3, 00.10, 00.7, 00.5, 99.10, 99.8, 99.6

핵심 14.8, 14.5, 13.8, 13.6, 13.3, 11.8, 11.3, 10.9, 10.5, 09.8, 09.5, 09.3, 08.5, 08.3, 07.9, 07.5, 07.3, 06.9, 06.5, 06.3, 05.3,

192 UNIX 시스템의 구성

커널 (Kernel)	• UNIX의 가장 핵심적인 부분으로, 주기억장치에 적재된 후 상주하면서 실행됨 • 하드웨어를 보호하고, 프로그램들과 하드웨어 간의 인터페이스 역할을 담당함 • 프로세스 관리, 기억장치 관리, 파일 관리, 입 · 출력 관리, 프로세스 간 통신, 데이터 전송 및 변환 등 여러 가지 기능을 수행함
쉘(Shell)	• 사용자의 명령어를 인식하여 프로그램을 호출하고, 명령을 수행하는 명령어 해석기 • 시스템과 사용자 간의 인터페이스를 담당함 • DOS의 COMMAND.COM과 같은 기능을 수행함 • 주기억장치에 상주하지 않고, 명령어가 포함된 파일 형태로 존재하며 보조기억장치에서 교체 처리가 가능함 • 공용 쉘이나 자신이 만든 쉘을 사용할 수 있음
유틸리티 (Utility)	• 일반 사용자가 작성한 응용 프로그램을 처리하는 데 사용함 • DOS에서의 외부 명령어에 해당됨

02.9, 02.5, 02.3, 01.9, 01.3, 00.3

핵심 14.3, 13.8, 12.8, 12.5, 12.3, 11.8, 11.6, 11.3, 10.5, 10.3, 09.8, 09.5, 08.9, 05.9, 05.5, 05.4, 04.9, 04.5, 04.3, 03.5, 03.3,

193 UNIX 파일 시스템

파일 시스템

- UNIX 파일 시스템의 디렉터리 구조는 트리 구조로 이루어져 있다.
- 디렉터리나 주변장치를 파일과 동일하게 취급한다.
- 파일 생성 및 삭제 기능, 보호 기능을 갖는다.
- 파일 형식은 일반 파일(Regular File), 디렉터리 파일(Directory File), 특수 파일(Special File)의 3가지 형식을 제공한다.

파일 시스템의 구조

- **부트 블록** : 부팅 시 필요한 코드를 저장하고 있는 블록
- **슈퍼 블록** : 전체 파일 시스템에 대한 종합적인 정보, 디스크 자체에 관련된 정보를 저장하고 있는 블록
- **Inode 블록** : 개개의 파일이나 디렉터리에 대한 모든 정보를 저장하고 있는 블록으로, 파일 소유자의 사용자 번호(UID) 및 그룹 번호(GID), 파일 크기, 파일 타입, 생성 시기, 최종 변경 시기, 최근 사용 시기, 파일의 보호 권한, 파일 링크 수, 데이터가 저장된 블록의 시작 주소 등에 대한 정보를 가지고 있음
- **데이터 블록** : 디렉터리별로 디렉터리 엔트리와 실제 파일에 대한 데이터가 저장된 블록

14.5, 14.3, 13.3, 12.8, 12.5 ,11.8, 11.6, 10.5, 08.9, 08.5, 07.9, 07.3, 06.9, 05.5, 05.3, 04.3, 03.8, 03.5, 03.3, 01.6, 00.10

핵심

194 UNIX의 주요 명령어

명령어	의미
fork	새로운 프로세스 · 자식 프로세스 생성, 하위 프로세스 호출, 프로세스 복제
exec	새로운 프로세스 수행
&	백그라운드 처리를 위해 명령의 끝에 입력
wait	fork 후 exec에 의해 실행되는 프로세스의 상위 프로세스가 하위 프로세스 종료 등의 event를 기다림
exit	프로세스 수행 종료
kill	프로세스 제거
cat	내용을 화면에 표시
chmod	파일의 사용 권한 허가 지정
mount	파일 시스템을 마운팅함
mkfs	파일 시스템 생성
chdir	현재 사용할 디렉터리 위치 변경
fsck	파일 시스템 검사 · 보수
rmdir	디렉터리 삭제
ls	현재 디렉터리 내의 파일 목록 확인
getpid	자신의 프로세스 아이디를 얻음
chown	소유자를 변경함
ps	프로세스 상태를 확인함

5과목 · 정보 통신 개론

14.8, 13.3, 11.8, 08.9, 08.3, 05.9, 05.5, 05.3, 04.9, 04.5, 04.3, 03.3, 02.9, 02.3, 01.6, 01.3, 00.10, 99.10, 99.8, 99.4

핵심

195 정보 통신의 개요

- **정보 통신** : 컴퓨터와 통신 기술의 결합에 의해 통신 처리 기능과 정보 처리 기능은 물론 정보의 변환, 저장 과정이 추가된 형태의 통신

> 정보 통신 = 전기 통신(정보 전송) + 컴퓨터(정보 처리)

- **통신의 3요소** : 정보원, 수신원, 전송 매체(=통신 회선)

• 데이터 통신 시스템의 주요 발달 과정

SAGE	• 최초의 데이터 통신 시스템 • 미 공군에 설치된 반자동 방공 시스템
SABRE	• 최초의 상업용 데이터 통신 시스템 • 미국 항공회사의 좌석 예약 시스템
ARPANET	인터넷의 효시가 된 통신 시스템
ALOHA	최초의 무선 패킷 교환 시스템, 회선 제어 방식 중 경쟁 방식의 모체
SNA	데이터 통신 시스템의 표준화가 시작

핵심

196 정보 통신 시스템의 특징

04.9, 01.6, 00.3, 99.6

- 고속 · 고품질의 전송이 가능하다.
- 고도의 오류 제어 방식으로 시스템의 신뢰도가 높다.
- 대형 컴퓨터와 대용량 파일을 공동으로 이용할 수 있다.
- 분산 처리가 가능하다.
- 통신 회선을 효율적으로 이용할 수 있다.
- 대용량 · 광대역 전송이 가능하다.
- 거리와 시간의 한계를 극복한다.
- 통신 비밀을 유지하기 위한 보안 시스템의 개발이 필요하다.

핵심

197 정보 통신 시스템의 기본 구성

13.6, 11.6, 10.5, 10.3, 08.3, 07.9, 06.5, 06.3, 05.4, 04.3, 03.3, 02.9, 02.5, 02.3, 01.3, 00.10, 00.7, 00.5, 99.6

데이터 전송계	단말장치, 데이터 전송회선(신호 변환장치, 통신회선), 통신 제어장치
데이터 처리계	컴퓨터(하드웨어, 소프트웨어)

통신 시스템의 구성 요소

- 통신 시스템의 4대 구성 요소 : 단말장치, 데이터 전송회선(신호 변환장치, 통신회선), 통신 제어장치(CCU), 컴퓨터
- 정보 통신 시스템의 3대 요소 : 단말장치, 전송장치(신호 변환장치, 통신회선), 컴퓨터
- 데이터 통신 시스템의 3대 요소 : 단말장치, 전송장치, 통신 제어장치
- 정보 통신망의 3대 구성 요소 : 단말장치, 교환장치, 전송장치

핵심

198 정보 통신 시스템의 처리 형태 및 응용

14.5, 14.3, 13.8, 10.9, 10.5, 06.9, 06.5, 05.5, 05.4, 03.3, 02.5, 99.4

정보 통신 시스템의 처리 형태

온라인 시스템	• 데이터가 발생한 단말장치와 데이터를 처리할 컴퓨터가 통신회선을 통해 직접 연결된 형태 • 데이터 송 · 수신 중간에 사람 혹은 기록 매체가 개입되지 않음 • 정보 통신 업무의 대부분을 차지하는 실시간 처리가 요구되는 작업에 주로 사용됨
실시간 처리 시스템	• 데이터가 발생한 즉시 처리하여 그 결과를 되돌려 주는 방식 • 은행 업무, 예약 업무, 각종 조회 업무 등에 사용
시분할 처리 시스템	컴퓨터를 사용할 수 있는 시간을 일정하게 나누어 여러 개의 단말장치가 정해진 시간(Time Slice) 동안 번갈아가며 컴퓨터의 자원을 공동으로 사용하는 방식

정보 통신 시스템의 응용

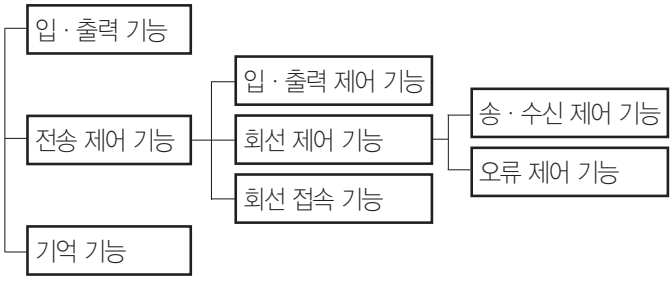
온라인 실시간 처리	거래 처리, 질의/응답, 메시지 교환
온라인 일괄 처리	데이터 수집과 입력, 원격 일괄 처리

핵심

199 단말장치(DTE)

14.8, 14.3, 13.8, 12.5, 10.5, 08.5, 04.9, 04.5, 00.7, 00.5, 00.3, 99.4

- 데이터 통신 시스템과 외부 사용자의 접점에 위치하여 최종적으로 데이터를 입 · 출력하는 데이터 회선 종단 장치이다.
- 기능



- 내장 프로그램의 유무(기능상)에 따른 분류

지능형(스마트) 단말장치	• CPU와 저장장치가 내장된 단말장치 • 프로그램을 설치하여 단독으로 일정 수준 이상의 작업 처리가 가능
비지능형(더미) 단말장치	• 입력장치와 출력장치로만 구성 • 단독으로 작업을 처리할 수 있는 능력이 없음

• 원격 일괄 단말장치(Remote Batch Terminal) : 멀리 떨어진 장소(원격지)에서 컴퓨터로 처리할 작업을 한꺼번에 모아서 처리(일괄 처리)하는 단말장치

핵심 200 통신 제어장치(CCU)

13.8, 13.6, 13.3, 12.3, 10.5, 10.3, 08.5, 07.3, 05.5, 01.9, 01.6, 99.8, 99.4

• 데이터 전송 회선과 주컴퓨터 사이에 위치하여, 컴퓨터가 데이터 처리에 전념할 수 있도록 컴퓨터를 대신해 데이터 송 · 수신에 관한 전반적인 제어 기능을 수행한다.

• 통신 제어장치의 기능

전송 제어	다중 접속 제어, 교환 접속 제어, 통신 방식 제어, 우회 중계 회선 설정(경로 설정)
동기 및 오류 제어	동기 제어, 흐름 제어, 오류 검출 및 제어, 회선의 감시 및 제어, 응답 제어, 정보 전송 단위의 정합, 데이터 신호의 직 · 병렬 변환(조립 및 분해), 투과성, 정보 표시 형식의 변환, 우선권 제어
그 밖의 기능	제어 정보 식별, 기밀 보호, 관리 기능

잠깐만요 ! 동기 제어 : 컴퓨터의 처리 속도와 통신 회선상의 전송 속도 차이를 조정하기 위해 송 · 수신 타이밍을 맞추는 기능

핵심 201 신호 변환장치(DCE)

00.7, 99.10, 99.6, 99.4
13.6, 13.3, 12.8, 12.5, 11.8, 11.6, 10.9, 09.5, 08.9, 07.5, 06.9, 06.5, 06.3, 05.5, 05.3, 04.9, 04.3, 03.8, 02.9, 01.6, 01.3,

컴퓨터나 단말장치의 데이터를 통신회선에 적합한 신호로 변경하거나 통신회선의 신호를 컴퓨터나 단말장치에 적합한 데이터로 변경하는 신호 변환 기능을 수행한다.

모뎀 (MODEM)	• 컴퓨터나 단말장치로부터 전송되는 디지털 데이터를 아날로그 회선에 적합한 아날로그 신호로 변환하는 변조(MOdulation) 과정과 그 반대의 복조(DEModulation) 과정을 수행함 • 디지털 데이터를 공중 전화 교환망(PSTN)과 같은 아날로그 통신망을 이용하여 전송할 때 사용함 • 기능 : 변 · 복조 기능, 자동 응답 기능, 반복 호출 기능, 자동 속도 조절 기능, 모뎀 시험 기능
DSU (Digital Service Unit)	• 컴퓨터나 단말장치로부터 전송되는 디지털 데이터를 디지털 회선에 적합한 디지털 신호로 변환하는 과정과 그 반대의 과정을 수행 • 신호의 변조 과정이 없이 단순히 유니폴라(단극성) 신호를 바이폴라(양극성) 신호로 변환하여 주는 기능만 제공하기 때문에 모뎀에 비하여 구조가 단순함 • 디지털 데이터를 공중 데이터 교환망(PSDN)과 같은 디지털 통신망을 이용하여 전송할 때 사용함 • 송 · 수신 기능과 타이밍 회복 기능을 DSU 자체에서 수행함 • 속도가 빠르고, 오류율이 낮음
코덱 (CODEC)	• 아날로그 데이터를 디지털 통신 회선에 적합한 디지털 신호로 변환하거나 그 반대의 과정을 수행함 • 펄스 코드 변조(PCM) 방식을 이용하여 데이터를 변환함

핵심 202 DTE/DCE 접속 규격

00.5, 99.10
14.5, 12.8, 11.8, 11.6, 10.9, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.5, 07.5, 07.3, 06.5, 05.5, 05.3, 04.3, 03.8, 03.5, 01.6,

• 단말장치(DTE)와 회선 종단장치(DCE)간의 접속을 정확하게 수행하기 위한 기계적(물리적), 전기적, 기능적, 절차적 조건을 사전에 정의해 놓은 규격으로, OSI 참조 모델의 물리 계층과 관계가 있다.

• 접속 규격 표준안

ITU-T	V 시리즈	공중 전화(아날로그 데이터) 교환망(PSTN)을 통한 DTE/DCE 접속 규격 • V.24 : 기능적, 절차적 조건에 대한 규정 • V.28 : 전기적 조건에 대한 규정
	X 시리즈	공중 데이터(디지털 데이터) 교환망(PSDN)을 통한 DTE/DCE 접속 규격 • X.20 : 비동기식 전송을 위한 DTE/DCE 접속 규격 • X.21 : 동기식 전송을 위한 DTE/DCE 접속 규격 • X.24 : DTE와 DCE간의 상호 접속 회로를 위한 DTE/DCE 접속 규격 • X.25 : 패킷 전송을 위한 DTE/DCE 접속 규격
EIA	RS-232C	공중 전화 교환망(PSTN)을 통한 DTE/DCE 접속 규격 • V.24, V.28, ISO2110을 사용하는 접속 규격과 기능적으로 호환성을 가지며, 현재 가장 많이 사용됨 • RS-232C에 의한 직접 접속(Null MODEM)에 사용되는 핀 : 보호용 접지(GND), 송신(TXD), 수신(RXD)
	RS-449	고속 데이터 통신을 위한 DTE/DCE 접속 규격 • RS-232C의 단점을 보완하기 위한 새로운 표준 • 거리에 제한이 없고, RS-232C에 비해 속도가 빠름

• RS-232C 커넥터 : 25핀으로 구성, 전송 거리는 15m 이하, 데이터 신호 속도는 최고 20Kbps이며, 전이중/반이중, 동기/비동기 모두에 대응함

핵심 14.5, 14.3, 13.6, 12.8, 12.5, 11.8, 11.6, 11.3, 10.9, 09.8, 08.9, 07.9, 04.5, 00.10, 99.6, 99.4

203 주파수 분할 다중화기(FDM)

- 통신회선의 주파수를 여러 개로 분할하여 여러 대의 단말장치가 동시에 사용할 수 있도록 한 것이다.

잠깐만요 ! 다중화(Multiplexing)
하나의 고속 통신 회선을 다수의 단말기가 공유할 수 있도록 하는 것으로, 다중화를 위한 장치에는 다중화기, 집중화기, 공동 이용기가 있습니다.

- 전송 신호에 필요한 대역폭보다 전송 매체의 유효 대역폭이 큰 경우에 사용한다.
- 다중화기 자체에 변 · 복조 기능이 내장되어 있어 모뎀을 설치할 필요가 없다.
- 시분할 다중화기에 비해 구조가 간단하고 가격이 저렴하다.
- 대역폭을 나누어 사용하는 각 채널들 간의 상호 간섭을 방지하기 위한 보호 대역(Guard Band)이 필요하다.
- 보호 대역(Guard Band) 사용으로 인한 대역폭의 낭비가 초래된다.
- 저속(1,200bps 이하)의 비동기식 전송, 멀티 포인트 방식, 아날로그 신호 전송에 적합하다.
- TV 방송이나 CATV등에 사용된다.

핵심 14.5, 14.3, 11.3, 06.3, 05.9, 00.5

204 시분할 다중화기(TDM)

- 통신회선의 대역폭을 일정한 시간 폭(Time Slot)으로 나누어 여러 대의 단말장치가 동시에 사용할 수 있도록 한 것이다.
- 대역폭의 이용도가 높아 고속 전송에 용이하다.
- 디지털 회선에서 주로 이용하며, 대부분의 데이터 통신에 사용된다.
- 다중화기의 내부 속도와 단말장치의 속도 차이를 보완하기 위한 버퍼가 필요하다.
- 모든 단말장치에 균등한(고정된) 시간폭(Time Slot)을 제공하는 동기식 시분할 다중화기(STDM)와 전송할 데이터가 있는 단말장치에만 시간폭(Time Slot)을 제공하는 비동기식 시분할 다중화(ATDM)기가 있다.

핵심 04.9, 03.3, 02.9, 00.3, 99.10, 99.8, 99.6

205 데이터 처리 시스템

- 정보 통신 시스템 중 데이터 처리 기능을 담당하는 컴퓨터를 의미한다.
- 통신 소프트웨어를 실행하고 송 · 수신되는 자료를 처리하는 등 데이터 통신 시스템 전체를 제어한다.
- 기계장치인 하드웨어와 하드웨어를 움직이는 소프트웨어로 구성된다.

하드웨어	• 중앙처리장치의 구성 – 제어장치 : 모든 장치들의 동작을 지시하고 제어 · 감독하는 장치 – 연산장치 : 제어장치의 명령에 따라 실제로 연산을 수행하는 장치 – 주기억장치 : 수행중인 프로그램과 데이터가 저장되는 장치
소프트웨어	• 운영체제 : 사용자의 편의를 도모하는 동시에 시스템의 생산성을 높이기 위한 프로그램의 모임으로, 컴퓨터의 전반적인 운영과 각종 컴퓨터 자원의 관리를 수행함 • 운영체제의 구성 – 제어 프로그램 : 컴퓨터 전체의 작동 상태 감시, 작업의 순서 지정, 작업에 사용되는 데이터 관리 등의 역할 수행 – 처리 프로그램 : 제어 프로그램의 지시를 받아 사용자가 요구한 문제를 해결하기 위한 역할 수행

핵심 14.8, 12.3, 11.8, 11.6, 11.3, 10.5, 10.3, 08.9, 07.5, 07.3, 06.9, 06.5, 06.3, 05.4, 04.3, 03.5, 03.3, 02.5, 01.9, 01.3,

206 광섬유 케이블(Optical Fiber Cable)

- 유리를 원료로 하여 제작된 가느다란 광섬유를 여러 가닥 묶어서 케이블의 형태로 만든 것이다.
- 데이터를 빛으로 바꾸어 빛의 반사(전반사) 원리를 이용하여 전송한다.
- 유선 매체 중 가장 빠른 속도와 높은 주파수 대역폭을 제공한다.
- 대용량, 장거리 전송이 가능하다.
- 도청이 어려워 보안성이 뛰어나다.
- 저손실성, 무유도, 무누화의 성질을 가진다.
- 감쇠율이 적어 리피터의 설치 간격이 넓으므로 리피터의 소요가 적다.
- 온도 변화에 안정적이고 신뢰성이 높다.
- 설치 비용은 비싸지만 단위 비용은 저렴하다.

- 광섬유 간의 연결이 어려워 설치 시 고도의 기술이 필요하다.
- 광섬유 케이블의 구성
  - 코어(Core) : 빛이 전파되는 영역으로, 클래드보다 높은 굴절률을 가짐
  - 클래드(Clad) : 코어보다 약간 낮은 굴절률을 가지므로 코어의 빛을 반사시켜 외부로 빠져나가지 못하게 하고, 코어를 외부의 압력으로부터 보호함
  - 재킷(Jacket) : 습기, 마모, 파손 등의 위험으로부터 내부를 보호함
- 광섬유 케이블의 전송 모드 : 단일 모드, 계단형 다중 모드, 언덕형 다중 모드

잠깐만요 ! 광통신의 3요소
- 발광기(LD; Laser Diode) : 전광 변환(전기 에너지 → 빛 에너지)을 수행하며, 송신 측 요소임
- 수광기(PD; Photo Diode) : 광전 변환(빛 에너지 → 전기 에너지)을 수행하며, 수신 측 요소임
- 광심선(광 케이블) : 중계부로, 유리를 원료로 하여 제작된 가느다란 광섬유로 구성됨

04.3, 03.8

핵심 **207 통신 속도와 통신 용량**

14.8, 14.3, 13.6, 13.3, 12.8, 12.3, 11.3, 10.9, 10.5, 09.8, 09.5, 09.3, 08.9, 08.5, 08.3, 07.5, 07.3, 06.9, 05.5, 05.3, 04.9,

- 통신 속도

변조 속도	• 1초 동안 몇 개의 신호 변화가 있었는가를 나타내는 것(단위 : Baud) • 변조 속도(Baud) = 데이터 신호 속도(Bps) / 변조 시 상태 변화 수
신호 속도	• 1초 동안 전송 가능한 비트의 수(단위 : Bps(Bit/Sec)) • 데이터 신호 속도(Bps) = 변조 속도(Baud) × 변조시 상태 변화 수
전송 속도	단위 시간에 전송되는 데이터의 양(문자, 블록, 비트, 단어 등)
베어러 속도	데이터 신호에 동기 문자, 상태 신호 등을 합한 속도(단위 : Bps(Bit/Sec))

- 변조 시 상태 변화 수 : 모노비트(Monobit) = 1Bit, 디비트(Dibit) = 2Bit, 트리비트(Tribit) = 3Bit, 쿼드비트(Quadbit)= 4Bit
- 통신 용량 : 단위 시간 동안 전송 회선이 최대로 전송할 수 있는 통신 정보량
- 샤논(Shannon)의 정의

$$C = W \cdot \log_2(1 + \frac{S}{N})[Bps]$$

C : 통신 용량, W : 대역폭(대역폭이 'Band Width'이므로 'W' 대신 'B'로도 사용), S : 신호 전력, N : 잡음 전력

- 전송로의 통신 용량을 늘리기 위한 방법 : 주파수 대역폭을 늘림, 신호 세력을 높임, 잡음 세력을 줄임

13.6, 09.3, 08.9, 06.5, 05.5, 05.4, 03.3, 02.5, 01.9, 00.7, 00.3

핵심 **208 신호 변환 방식의 종류**

아날로그 데이터 → 아날로그 신호	진폭 변조(AM), 주파수 변조(FM), 위상 변조(PM)
디지털 데이터 → 아날로그 신호	• 모뎀 이용 • 변조 방식 : 진폭 편이 변조(ASK), 주파수 편이 변조(FSK), 위상 편이 변조(PSK), 직교 진폭 변조(QAM)
아날로그 데이터 → 디지털 신호	• 코덱 이용 • 변조 방식 : 펄스 코드 변조(PCM)
디지털 데이터 → 디지털 신호	2진 데이터의 각 비트를 디지털 신호 요소로 변환하며, DSU를 이용함

14.8, 12.8, 12.5, 12.3, 11.6, 11.3, 10.5, 09.8, 09.5, 09.3, 08.9, 08.5, 06.3, 05.5, 05.4, 04.9, 03.8, 03.5, 01.9, 99.10

핵심 **209 신호 변환 방식 – 디지털 변조**

진폭 편이 변조(ASK)	• 2진수 0과 1을 서로 다른 진폭의 신호로 변조 • 신호 변동과 잡음에 약하여 데이터 전송용으로 거의 사용되지 않음
주파수 편이 변조(FSK)	• 2진수 0과 1을 서로 다른 주파수로 변조 • 1,200Bps 이하의 저속도 비동기식 모뎀에서 사용
위상 편이 변조(PSK)	• 2진수 0과 1을 서로 다른 위상을 갖는 신호로 변조 • 한 위상에 1비트(2위상), 2비트(4위상), 또는 3비트(8위상)를 대응시켜 전송하므로, 속도를 높일 수 있음 • 중 · 고속의 동기식 모뎀에 많이 사용됨
직교 진폭 변조(QAM) = 진폭 위상 변조, 직교 위상 변조	• 반송파의 진폭과 위상을 상호 변환하여 신호를 얻는 변조 방식 • 고속 전송 가능, 9,600Bps 모뎀의 표준 방식으로 권고됨

핵심 210 14.8, 14.3, 12.8, 12.3, 11.6, 10.5, 09.8, 09.5, 08.3, 07.5, 06.5, 05.4, 00.5, 99.6

신호 변환 방식 – 펄스 코드 변조(PCM)

- 화상, 음성, 동영상 비디오, 가상 현실 등과 같이 연속적인 시간과 진폭을 가진 아날로그 데이터를 디지털 신호로 변조하는 방식으로, CODEC을 이용하다.
- 펄스 변조 : 펄스파의 진폭, 폭, 위상 등을 변화시키는 변조 방식
- 펄스 코드 변조(PCM) 순서 : 송신 측(표본화 → 양자화 → 부호화) → 수신 측(복호화 → 여파화)

표본화 (Sampling)	• 음성, 영상 등의 연속적인 신호 파형을 일정 시간 간격으로 검출하는 단계 • 샤논(Nyquist Shanon)의 표본화 이론 : 어떤 신호 $f_{(t)}$가 의미를 지니는 최고 주파수보다 2배 이상의 주파수로 균일한 시간 간격 동안 채집된다면 이 채집된 데이터는 원래의 신호가 가진 모든 정보를 포함함 • 표본화에 의해 검출된 신호를 PAM 신호라고 하며, 아날로그 형태임 • 표본화 횟수 = 2배 × 최고 주파수 • 표본화 간격 = $\frac{1}{\text{표본화 횟수}}$
양자화 (Quantizing)	• 표본화된 PAM 신호를 유한 개의 부호에 대한 대표값으로 조정하는 과정 • 실수 형태의 PAM 신호를 반올림하여 정수형으로 만듦 • 양자화 잡음 : 표본 측정값과 양자화 파형과의 오차를 말하며, 주로 PCM 단국장치에서 발생 • 양자화 잡음은 양자화 레벨을 세밀하게 함으로써 줄일 수 있으나, 이 경우 데이터의 양이 많아지고 전송 효율이 낮아짐 • 양자화 레벨 : PAM 신호를 부호화할 때 2진수로 표현할 수 있는 레벨(양자화 레벨 = $2^{\text{표본당 전송 비트 수}}$)
부호화 (Encoding)	양자화된 PCM 펄스의 진폭 크기를 2진수(1 또는 0)로 표시하는 과정
복호화 (Decoding)	수신된 디지털 신호(PCM 신호)를 PAM 신호로 되돌리는 과정
여파화 (Filtering)	PAM 신호를 원래의 입력 신호인 아날로그 신호로 복원하는 과정

핵심 211 05.9, 04.5, 02.3, 00.7, 00.5

전송 방식

아날로그/디지털 전송

아날로그 전송	• 전송 매체를 통해 전달되는 신호가 아날로그 형태인 것 • 신호의 감쇠 현상이 심하므로 장거리 전송 시 증폭기에 의해 신호를 증폭하여 전송하며, 이때 신호에 포함된 잡음까지 같이 증폭되기 때문에 오류의 확률이 높음
디지털 전송	• 전송 매체를 통해 전달되는 신호가 디지털 형태인 것 • 장거리 전송 시 중계기(재생기, Repeater)에 의해 원래의 신호 내용을 다시 복원한 다음 전송하는 방식이기 때문에 잡음에 의한 오류율이 낮음 • 대역폭을 효율적으로 이용하여 더 많은 용량을 전송할 수 있음 • 아날로그나 디지털 정보의 암호화를 쉽게 구현할 수 있음 • 전송 매체로는 광섬유 케이블, 동축 케이블, UTP 케이블, 마이크로 웨이브(M/W) 등이 사용 • 기술의 발전으로 인한 전송 장비의 소형화, 가격의 저렴화가 가능함

직렬/병렬 전송

직렬 전송	• 정보를 구성하는 각 비트들이 하나의 전송 매체를 통하여 한 비트씩 순서적으로 전송되는 형태 • 전송 속도가 느림 • 원거리 전송에 적합하며 구성 비용이 적게 들어 대부분의 데이터 통신에 사용
병렬 전송	• 정보를 구성하는 각 비트들이 여러 개의 전송 매체를 통하여 동시에 전송되는 형태 • 전송 속도는 빠르지만 구성 비용이 많이 소요됨 • 근거리 전송에 적합하며 주로 컴퓨터와 주변기기 사이의 데이터 전송에 사용됨

핵심 212 14.8, 14.5, 13.8, 13.6, 13.3, 12.8, 09.3, 08.9, 05.4, 04.9, 03.8, 03.5, 02.9, 02.5, 01.6, 00.10, 00.5, 00.3, 99.8

통신 방식

- 단방향(Simplex) 통신 : 한쪽 방향으로만 전송이 가능한 방식(예 라디오, TV)
- 반이중(Half-Duplex) 통신 : 양방향 전송이 가능하지만 동시에 양쪽 방향에서 전송할 수 없는 방식(예 무전기, 모뎀을 이용한 데이터 통신)
- 전이중(Full-Duplex) 통신 : 동시에 양방향 전송이 가능한 방식으로, 전송량이 많고, 전송 매체의 용량이 클 때 사용(예 전화, 전용선을 이용한 데이터 통신)

핵심 213 비동기식 전송

14.3, 13.6, 13.3, 12.5, 11.6, 11.3, 08.5, 07.5, 07.3, 04.5, 03.8, 01.6, 00.10, 00.3, 99.8

- 한 문자를 나타내는 부호(문자 코드) 앞뒤에 Start Bit와 Stop Bit를 붙여서 Byte와 Byte를 구별하여 전송하는 방식이다.
- Stop Bit는 1Bit, $1\frac{1}{2}$Bit, 2Bit가 사용된다.
- 시작 비트, 전송 문자(정보 비트), 정지 비트로 구성된 한 문자를 단위로 하여 전송하며, 오류 검출을 위한 패리티 비트(Parity Bit)를 추가하기도 한다.
- 문자와 문자 사이의 휴지 시간(Idle Time)이 불규칙하다.
- 전송 속도 300~19200Bps의 비교적 저속, 단거리 전송에 사용된다.
- 문자마다 시작, 정지를 알리기 위한 비트가 2~3Bit씩 추가되므로, 전송 효율이 떨어진다.
- 실제 전송 시 잉여 Bit의 상승률이 커진다.

핵심 214 동기식 전송

05.5, 03.8, 00.10, 00.3, 99.8, 99.6

- 미리 정해진 수만큼의 문자열을 한 블록(프레임)으로 만들어 일시에 전송하는 방식이다.
- 프레임 단위로 전송하므로 전송 속도가 빠르다.
- 프레임(Frame) : 전송할 자료를 일정한 크기로 분리한 것으로, 데이터뿐만 아니라 행선지 코드, 동기를 위한 제어 문자, 오류 검출을 위한 패리티나 CRC 등의 추가 정보를 포함함
- 시작/종료 비트로 인한 오버헤드가 없고, 휴지 시간이 없으므로, 전송 효율이 좋다.
- 주로 원거리 전송에 사용한다.
- 단말기는 반드시 버퍼 기억장치를 내장하여야 한다.
- 비트 동기 방식과 블록 동기 방식이 있으며, 블록 동기 방식은 문자 위주 동기 방식과 비트 위주 동기 방식으로 나누어진다.

문자(위주) 동기 방식	• SYN 등의 동기 문자(전송 제어 문자)에 의해 동기를 맞추는 방식 • BSC 프로토콜에서 사용됨
비트(위주) 동기 방식	• 데이터 블록의 처음과 끝에 8비트의 플래그 비트(01111110)를 표시하여 동기를 맞추는 방식 • HDLC, SDLC 프로토콜에서 사용됨

핵심 215 회선 구성 방식

12.8, 11.6, 03.3, 02.5, 02.3

포인트 투 포인트 (Point-to-Point) 방식	• 중앙 컴퓨터와 단말기를 일 대 일 독립적으로 연결하는 방식 • 통신망을 성형(Star)으로 구성할 때 사용
멀티 드롭 방식 (Multi-Drop) = 멀티 포인트	• 여러 대의 단말기들을 한 개의 통신회선에 연결하는 방식 • 단말기는 주소 판단 기능과 버퍼를 가지고 있어야 함 • 회선 공유로 효용도가 높고, 가격도 저렴 • 선로의 속도, 단말기에 의해 생기는 교통량, 하드웨어와 소프트웨어의 처리 능력에 따라 연결할 수 있는 단말기의 수가 달라짐 • 통신망을 버스형(Bus)으로 구성할 때 사용
회선 다중 (Line Multiplexing) 방식	• 여러 대의 단말기들을 다중화 장치를 이용하여 중앙 컴퓨터와 연결하는 방식 • 중앙 컴퓨터와 다중화 장치 사이를 대용량 회선으로 연결하여 전송 속도 및 효율을 높임 • 통신 회선의 고장 시 고장 지점 이후의 터미널은 모두 운영 불능에 빠지는 단점이 있음

핵심 216 회선 제어 방식

14.5, 13.3, 12.3, 03.3, 01.9, 01.3, 00.10

경쟁 (Contention) 방식	• 회선 접속을 위해서 서로 경쟁하는 방식 • 데이터를 전송하고자 하는 모든 단말장치가 서로 대등한 입장에 있음 • 데이터 링크가 설정되면 정보 전송이 종료되기 전까지는 데이터 링크의 종결이 이루어지지 않고 독점적으로 정보를 전송함 • 대표적인 시스템으로는 ALOHA가 있음
폴링/셀렉션 (Polling/ Selection) 방식	• 주컴퓨터에서 송 · 수신 제어권을 가지고 있는 방식 • 폴링(Polling) : 주컴퓨터에서 단말기에게 전송할 데이터가 있는지를 물어 전송할 데이터가 있다면 전송을 허가하는 방식으로, 단말기에서 주컴퓨터로 보낼 데이터가 있는 경우에 사용 • 셀렉션(Selection) : 주컴퓨터가 단말기로 전송할 데이터가 있는 경우 그 단말기가 받을 준비가 되어 있는가를 묻고, 준비가 되어 있다면 주컴퓨터에서 단말기로 데이터를 전송하는 방식

핵심 217 전송 제어의 기본

13.6, 11.6, 07.9, 04.3, 01.6, 01.3, 00.10, 00.7

- 전송 제어 : 데이터의 원활한 흐름을 위하여 입 · 출력 제어, 회선 제어, 동기 제어, 오류 제어, 흐름 제어 등을 수행하는 것

- OSI 7 참조 모델의 데이터 링크 계층(2계층)에서 수행하는 기능이다.
- **전송 제어 절차** : 데이터 통신 회선의 접속 → 데이터 링크 설정(확립) → 정보 메시지 전송 → 데이터 링크 종결 → 데이터 통신회선의 절단

데이터 통신 회선의 접속	통신회선과 단말기를 물리적으로 접속하는 단계
데이터 링크 설정(확립)	송 · 수신 측간의 논리적 경로를 구성하는 단계
데이터(정보 메시지) 전송	• 설정된 데이터 링크를 통해 데이터를 수신 측에 전송하는 단계 • 오류 제어와 순서 제어를 수행함 • 확인 신호(ACK) 등을 보내고 문자 동기를 유지함
데이터 링크 종결(해제)	송 · 수신 측 간의 논리적 경로를 해제하는 단계
데이터 통신 회선의 절단	통신회선과 단말기 간의 물리적 접속을 절단하는 단계

핵심 218 BSC/전송 제어 문자

14.5, 13.8, 12.8, 11.8, 10.9, 09.8, 09.3, 08.5, 06.9, 05.5, 05.4, 00.7

BSC

- 문자(Character) 위주의 프로토콜로, 바이트(Byte) 단위로 구성되는 각 프레임에 전송 제어 문자를 삽입하여 전송을 제어한다.
- 문자 코드에 의존적이며, 사용할 수 있는 코드가 제한적이다.
- 통신하는 컴퓨터들이 사용하는 문자 코드 체계가 통일되어 있어야 한다.
- 반이중(Half Duplex) 전송만을 지원한다.
- 주로 동기식 전송을 사용하나 비동기 전송 방식을 사용하기도 한다.
- 포인트 투 포인트, 멀티 포인트 방식에서 주로 사용한다.
- 에러 및 흐름 제어를 위해 정지 대기(Stop-and-Wait) ARQ를 사용한다.
- 오류 검출이 어렵고, 전송 효율이 나쁘다.

전송 제어 문자

링크 관리, 프레임의 시작 및 끝의 구별과 오류 제어 등의 기능을 한다.

문자	기능
SYN(SYNchronous Idle)	문자 동기
SOH(Start Of Heading)	헤딩의 시작
STX(Start of TeXt)	본문의 시작 및 헤딩의 종료
ETX(End of TeXt)	본문의 종료
ETB(End of Transmission Block)	블록의 종료
EOT(End Of Transmission)	전송 종료 및 데이터 링크의 해제
ENQ(ENQuiry)	상대편에 데이터 링크 설정 및 응답 요구
DLE(Data Link Escape)	전송 제어 문자 앞에 삽입하여 전송 제어 문자임을 알림(문자의 투과성을 위해 삽입)
ACK(ACKnowledge)	수신된 메시지에 대한 긍정 응답으로, 아무런 에러가 없다는 의미임
NAK (Negative AcKnowledge)	수신된 메시지에 대한 부정 응답

핵심 219 HDLC

04.9, 02.5, 01.3, 00.7, 00.3, 00.2
14.8, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.6, 10.9, 09.8, 09.5, 08.5, 07.5, 07.3, 06.9, 06.5, 06.3, 05.4, 05.3,

- 비트(Bit) 위주의 프로토콜로, 각 프레임에 데이터 흐름을 제어하고 오류를 보정할 수 있는 비트 열을 삽입하여 전송한다.
- 포인트 투 포인트 및 멀티 포인트, 루프 방식에서 모두 사용 가능하다.
- 단방향, 반이중, 전이중 통신을 모두 지원하며 동기식 전송 방식을 사용한다.
- 에러 제어를 위해 Go-Back-N ARQ와 선택적 재전송(Selective Repeat) ARQ를 사용한다.
- 흐름 제어를 위해 슬라이딩 윈도우 방식을 사용한다.
- 전송 제어상의 제한을 받지 않고 자유로이 비트 정보를 전송할 수 있다(비트 투과성).
- 전송 효율과 신뢰성이 높다.
- HDLC 프레임 구조

플래그	주소부	제어부	정보부	FCS	플래그

– 플래그(Flag) : 프레임의 시작과 끝을 나타내는 고유한 비트 패턴(01111110)으로, 프레임의 시작과 끝을 구분, 동기 유지(통화로의 혼선을 방지하기 위해), 비트 투과성을 이용한 기본적인 오류 검출 등의 기능을 수행함

– 주소부(Address Field)

▶ 송 · 수신국을 식별하기 위해 사용

▶ 모든 수신국에 전송되는 방송용은 '11111111', 임의로 지정된 수신국에만 전송되는 시험용은 '00000000'

– 제어부(Control Field) : 프레임의 종류를 식별하기 위해 사용

▶ 정보 프레임(Information Frame) : 사용자 데이터를 전달

▶ 감독 프레임(Supervisory Frame) : 오류 제어와 흐름 제어를 수행

▶ 비(무)번호 프레임(Unnumbered Frame) : 회선의 설정, 유지 및 종결을 담당

– 정보부(Information Field) : 실제 정보 메시지가 들어 있는 부분

– FCS(프레임 검사 순서 필드) : 프레임 내용에 대한 오류 검출을 위해 사용되는 부분으로, 일반적으로 CRC 코드가 사용됨

– HDLC의 데이터 전송 모드 : 표준(정규) 응답 모드(NRM), 비동기 응답 모드(ARM), 비동기 균형(평형) 모드(ABM)

핵심 220 오류의 발생 원인

14.8, 14.3, 09.5, 09.3, 08.3, 04.9

- **감쇠(Attenuation)** : 전송 신호 세력이 전송 매체를 통과하는 과정에서 거리에 따라 약해지는 현상
- **지연 왜곡(Delay Distortion)** : 하나의 전송 매체를 통해 여러 신호를 전달했을 때 주파수에 따라 그 속도가 달라짐으로써 생기는 오류
- **백색 잡음(White Noise)** : 전송 매체 내부에서 온도에 따라 전자의 운동량이 변화함으로써 생기는 잡음으로, 가우스 잡음, 열 잡음이라고도 함
- **상호 변조(간섭) 잡음(Intermodulation Noise)** : 서로 다른 주파수들이 하나의 전송 매체를 공유할 때 주파수 간의 합(合)이나 차(差)로 인해 새로운 주파수가 생성되는 잡음
- **누화 잡음 = 혼선(Cross talk Noise)** : 인접한 전송 매체의 전자기적 상호 유도 작용에 의해 생기는 잡음
- **충격성 잡음(Impulse Noise)** : 번개와 같은 외부적인 충격 또는 통신 시스템의 결함이나 파손 등의 기계적인 충격에 의해 순간적으로 생기는 잡음으로, 디지털 데이터를 전송하는 경우 중요한 오류 발생 요인이 됨

핵심 221 전송 오류 제어 방식

05.5, 00.5

전진(순방향) 오류 수정(FEC)	• 재전송 요구 없이 오류 검출과 수정을 스스로 하는 방식 • 역채널이 필요 없고, 연속적인 데이터 흐름이 가능함 • 데이터 비트 이외에 오류 검출 및 수정을 위한 비트(잉여 비트)들이 추가로 전송되어야 하기 때문에 전송 효율이 떨어짐 • 오류 검출과 수정을 모두 수행해야 하므로, 후진(역방향) 오류 수정에 비해 기기와 코딩이 더 복잡함 • 해밍 코드, 상승 코드 방식이 있음
후진(역방향) 오류 수정(BEC)	• 데이터 전송 과정에서 오류가 발생하면 송신 측에 재전송을 요구하는 방식 • 패리티 검사, CRC, 블록합 방식 등을 사용하여 오류를 검출하고, 오류 제어는 자동 반복 요청(ARQ)에 의해 이루어짐

핵심 222 전송 오류 제어방식 – 자동 반복 요청(ARQ)

14.5, 14.3, 13.8, 13.3, 12.3, 11.6, 10.9, 09.5, 07.3, 05.9

- 자동 반복 요청은 오류 발생 시 수신 측은 오류 발생을 송신 측에 통보하고, 송신 측은 오류 발생 블록을 재전송하는 모든 절차를 의미한다.
- **종류**

정지–대기(Stop and Wait) ARQ	• 송신 측에서 한 개의 블록을 전송한 후 수신 측으로부터 응답을 기다리는 방식 • 구현 방법이 가장 단순하지만, 전송 효율이 떨어짐
연속(Continuous) ARQ	• 연속적으로 데이터 블록을 보내는 방식 • Go–Back–N ARQ : 오류가 발생한 블록 이후의 모든 블록을 재전송하는 방식 • 선택적 재전송(Selective Repeat) ARQ : 오류가 발생한 블록만을 재전송하는 방식

적응적 (Adaptive) ARQ	• 블록 길이를 채널의 상태에 따라 그때그때 동적으로 변경하는 방식으로, 전송 효율이 가장 좋음 • 제어 회로가 복잡하고, 비용이 많이 들어 현재 거의 사용되지 않음

핵심 223 오류 검출 방식

14.8, 14.5, 14.3, 13.8, 13.6, 12.5, 12.3, 10.3, 09.8, 08.3, 07.3, 05.5, 05.3, 04.5, 02.3, 02.5, 01.9, 01.6, 01.3, 00.7, 99.4

패리티 검사	• 데이터 블록에 1비트의 검사 비트인 패리티 비트(Parity Bit)를 추가하여 오류를 검출함 • 가장 간단한 방식이지만, 2개의 비트에 오류가 동시에 발생하면 검출이 불가능함 • 오류를 검출만 할 수 있고, 수정은 하지 못함 • 홀수/짝수 수직 패리티 체크와 홀수/짝수 수평 패리티 체크가 있음 • 짝수 = 우수, 홀수 = 기수
해밍 코드	• 수신측에서 오류가 발생한 비트를 검출한 후 직접 수정하는 방식 • 1비트의 오류만 수정이 가능하며, 정보 비트 외에 잉 여 비트가 많이 필요함 • 전송 비트 중 1, 2, 4, 8, 16, 32, 64, …, 2^n번째를 오류 검출을 위한 패리티 비트로 사용함
순환 중복(잉여) 검사(CRC)	• 프레임 단위로 오류 검출을 위한 다항식 코드(FCS)를 사용하여 오류를 검출하는 방식 • 동기식 전송에 사용되는 에러 검출 기법으로 데이터가 프레임 단위로 전송될 때 사용되는 방식임 • HDLC 프레임의 FCS(프레임 검사 순서 필드)를 만드는 방법으로 사용됨 • 집단 오류를 검출할 수 있고, 검출률이 높으므로 가장 많이 사용됨

핵심 224 통신 프로토콜 / 프로토콜의 전송 방식

01.3, 00.5, 99.10, 99.8, 99.6
14.8, 13.6, 13.3, 12.8, 12.3, 10.9, 09.8, 08.9, 08.5, 08.3, 07.9, 07.3, 05.9, 05.5, 05.4, 04.3, 03.8, 03.5, 03.3, 02.9, 02.3,

통신 프로토콜

- 정의 : 서로 다른 기기들 간의 데이터 교환을 원활하게 수행할 수 있도록 표준화시켜 놓은 통신 규약
- 기본 요소
  - 구문(Syntax) : 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정
  - 의미(Semantics) : 두 기기 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정
  - 시간(Timing) : 두 기기 간의 통신 속도, 메시지의 순서 제어 등을 규정
- 기능 : 단편화, 재결합, 캡슐화, 흐름 제어, 오류 제어, 동기화, 순서 제어, 주소 지정, 다중화, 경로 제어, 전송 서비스(우선순위, 서비스 등급, 보안성)
- 캡슐화(Encapsulation) : 전송할 데이터의 앞 부분과 뒷 부분에 헤더와 트레일러라는 제어 정보를 첨가하는 과정
- 캡슐화 할 때 제어 정보에 포함되는 것 : 송 · 수신지 주소, 오류 검출 코드, 프로토콜 제어 정보

프로토콜의 전송 방식

- 문자 전송 방식 : 전송 제어 문자를 사용하여 데이터 프레임의 시작과 끝을 나타내는 방식(예 BSC 등)
- 바이트 방식 : 데이터 프레임의 헤더(Header)에 전송 데이터 프레임의 문자 개수, 메시지 수신 상태 등의 제어 정보를 삽입하여 전송하는 방식(예 DDCM 등)
- 비트 방식 : 데이터 프레임의 시작과 끝을 나타내는 고유한 비트 패턴(플래그)을 삽입해서 전송하는 방식(예 HDLC, SDLC, LAPB, ADCCP 등)

핵심 225 OSI 참조 모델

13.8, 03.5, 01.9, 00.5

OSI 참조 모델의 목적

- 서로 다른 시스템 간을 상호 접속하기 위한 개념을 규정한다.
- 시스템 간의 상호 회선 교환뿐만 아니라 축적 교환을 위한 개념도 규정한다.
- OSI 규격을 개발하기 위한 범위를 정한다.
- 관련 규격의 적합성을 조정하기 위한 공통적 기반을 제공한다.

OSI 참조 모델의 기본 원칙

- 적절한 수의 계층으로 나누어 시스템의 복잡도를 최소화한다.
- 서비스 접점의 경계를 두어 상호 작용이 적어질 수 있도록 한다.
- 프로세스나 기술적인 면에서 명백히 다른 기능을 처리하도록 계층을 분리한다.
- 비슷한 기능은 하나의 계층으로 모은다.
- 인접한 상 · 하위 계층 간에는 인터페이스를 둔다.
- 한 계층을 수정할 때 다른 계층에 영향을 주지 않도록 한다.

06.3, 05.5, 05.4, 05.3, 04.9, 04.5, 03.3, 02.9, 02.5, 02.3, 01.6, 00.10, 00.3, 99.10
14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.5, 12.3, 11.8, 11.6, 11.3, 10.5, 10.3, 09.8, 09.5, 09.3, 08.9, 08.5, 07.3, 06.9, 06.5,

핵심 226 OSI 7계층의 기능

- 다른 시스템 간의 원활한 통신을 위해 ISO(국제 표준화 기구)에서 제안한 통신 규약(Protocol)이다.
- OSI 7계층 : 하위 계층(물리 계층 → 데이터 링크 계층 → 네트워크 계층) → 상위 계층(전송 계층 → 세션 계층 → 표현 계층 → 응용 계층)

물리 계층 (Physical Layer)	전송에 필요한 두 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성을 정의
데이터 링크 계층 (Data Link Layer)	• 두 개의 인접한 개방 시스템들 간에 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 함 • 흐름 제어, 프레임 동기화, 오류 제어, 순서 제어 • 논리 링크 제어 및 매체 액세스 제어를 담당함 • HDLC, PPP, LLC, ADCCP, LAPB, X.25 등의 프로토콜을 사용함 • 데이터 링크 연결의 설정과 해제를 담당함
네트워크 계층 (Network Layer, 망 계층)	• 개방 시스템들 간의 네트워크 연결 관리(네트워크 연결을 설정, 유지, 해제), 데이터의 교환 및 중계 • 경로 설정(Routing), 트래픽 제어, 패킷 정보 전송
전송 계층 (Transport Layer)	• 종단 시스템(End-to-End) 간에 투명한 데이터 전송을 가능하게 함 • 전송 연결 설정, 데이터 전송, 연결 해제 기능 • 주소 설정, 다중화, 에러 제어, 흐름 제어 • TCP, UDP 등의 프로토콜을 사용함
세션 계층 (Session Layer)	• 송 · 수신 측간의 관련성을 유지하고 대화 제어를 담당 • 프로세스들 간의 대화(회화) 구성 및 동기 제어, 데이터 교환 관리 기능 • 체크점(=동기점) : 오류가 있는 데이터의 회복을 위해 사용하는 것으로 소동기점과 대동기점이 있음
표현 계층 (Presentation Layer)	• 응용 계층으로부터 받은 데이터를 세션 계층에 맞게, 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하는 기능 • 코드 변환, 데이터 암호화, 데이터 압축, 구문 검색, 정보 형식(포맷) 변환, 문맥 관리 기능
응용 계층 (Application Layer)	사용자(응용 프로그램)가 OSI 환경에 접근할 수 있도록 서비스를 제공함

14.5, 14.3, 12.5, 11.8, 10.9, 09.5, 07.5, 06.9, 06.3, 05.9, 04.9, 02.3, 01.9, 00.10, 99.4

핵심 227 TCP/IP

- 인터넷에 연결된 서로 다른 기종의 컴퓨터들이 원활하게 데이터를 주고받을 수 있도록 하는 표준 프로토콜이다.
- TCP/IP는 TCP 프로토콜과 IP 프로토콜이 결합된 것을 의미한다.

TCP(Transmission Control Protocol)	• OSI 7계층의 전송 계층에 해당함 • 신뢰성 있는 연결형 서비스를 제공함 • 패킷의 다중화, 순서 제어, 오류 제어, 흐름 제어 기능을 제공함
IP(Internet Protocol)	• OSI 7계층의 네트워크 계층에 해당함 • 데이터그램을 기반으로 하는 비연결형 서비스를 제공함 • 패킷의 분해/조립, 주소 지정, 경로 선택 기능을 제공함

- TCP/IP 계층 구조
  - 응용 계층 : 응용 프로그램 간의 데이터 송 · 수신 제공(TELNET, FTP, SMTP, SNMP, E-Mail 등)
  - 전송 계층 : 호스트들 간의 통신 제공(TCP, UDP)
  - 인터넷 계층 : 데이터 전송을 위한 주소 지정, 경로 배정 제공(IP, ICMP, IGMP, ARP, RARP 등)
  - 네트워크 액세스 계층 : 실제 데이터(프레임)를 송 · 수신하는 역할(Ethernet, IEEE 802, HDLC, X.25, RS-232C 등)

잠깐만요 TCP/IP 계층 구조
네트워크 액세스 계층을 물리 계층과 데이터 링크 계층으로 세분화하여 물리 계층, 데이터 링크 계층, 인터넷 계층, 전송 계층, 응용 계층 이렇게 5계층으로 구분하기도 합니다.

06.3, 05.5, 04.3, 03.8, 02.5, 01.9, 01.3, 00.7

핵심 228 표준안 제정 기관

- 국제표준화기구(ISO), 국제전기통신연합(ITU), 국제전기표준협회(IEC), 미국표준기구(ANSI), 전자공업협회(EIA), 전기전자기술자협회(IEEE), 인터넷 아키텍처 위원회(IAB)가 있다.
- 주요 ITU-T 권고 시리즈

I 시리즈	ISDN에 관한 권고
X 시리즈	공중 데이터망(PSDN)을 통한 데이터 전송에 관한 권고
V 시리즈	공중 전화망(PSTN)을 통한 데이터 전송에 관한 권고
T 시리즈	텔레마틱 서비스를 위한 단말장치와 프로토콜에 관한 권고

• IETF(Internet Engineering Task Force) : IAB의 산하 조직으로 인터넷의 운영, 관리 및 기술적인 쟁점 등을 해결하기 위해 망 설계자, 관리자, 연구자, 망 사업자 등으로 구성된 조직으로, 변화하는 망 환경에 따라 새로운 기술을 제시하고 인터넷 표준안을 제정하기 위한 기술 위원회

핵심

14.5, 13.8, 13.6, 13.3, 11.6, 08.9, 06.3, 04.9, 03.8, 03.3, 02.9, 02.3, 01.9, 01.6, 01.3, 00.10, 00.5, 00.3, 99.8

229 망(Network)의 구성 형태

성형(Star, = 중앙 집중형)

• 중앙에 중앙 컴퓨터가 있고, 이를 중심으로 단말기들이 연결되는 중앙 집중식 네트워크 구성 형태이다.

• 단말기의 추가와 제거가 쉽고, 교환 노드의 수가 가장 적다.

링형(Ring, = 루프형)

• 컴퓨터와 단말기들을 서로 이웃하는 것끼리 포인트 투 포인트 방식으로 연결시킨 형태이다.

• 데이터는 단방향 또는 양방향으로 전송할 수 있으며, 단방향 링의 경우 컴퓨터, 단말기, 통신 회선 중 어느 하나라도 고장나면 전체 통신망에 영향을 미친다.

• 양방향 링의 경우 한 노드(Node)가 절단되어도 우회로를 구성하여 통신이 가능하다.

버스형(Bus)

• 한 개의 통신 회선에 여러 대의 단말기가 연결되어 있는 형태이다.

• 물리적 구조가 간단하고, 단말기의 추가와 제거가 용이하다.

계층형(Tree, = 분산형)

• 각 단말기(노드)가 계층적으로 구성되는 것으로 중앙 컴퓨터와 일정 지역의 단말기까지는 하나의 통신 회선으로 연결시키고, 이웃하는 단말기는 일정 지역 내에 설치된 중간 단말기로부터 다시 연결시키는 형태이다.

• 분산 처리 시스템을 구성하는 방식이다.

망형(Mesh)

• 모든 지점의 컴퓨터와 단말기를 서로 연결한 형태로, 노드의 연결성이 높다.

• 많은 단말기로부터 많은 양의 통신을 필요로 하는 경우에 유리하다.

• 공중 데이터 통신망에서 사용되며, 통신 회선의 총 경로가 가장 길다.

• 통신 회선 장애 시 다른 경로를 통하여 데이터를 전송할 수 있다.

• 모든 노드를 망형으로 연결할 때, 노드수가 n개라면 필요한 회선 수는 $\frac{n(n-1)}{2}$ 개이고, 각 장치당 포트 수는 n−1개이다.

핵심

12.8, 10.9, 09.5, 08.3, 06.9, 06.3, 05.9, 00.5

230 회선 교환 방식

• 통신을 원하는 두 지점을 교환기를 이용하여 물리적으로 접속시키는 방식이다.

• 데이터 전송 전에 먼저 물리적 통신 회선을 통한 연결이 필요하다.

• 접속이 되고 나면 그 통신 회선은 전용 회선에 의한 통신처럼 데이터가 전달된다(고정 대역 전송).

• 접속에는 긴 시간이 소요되나 일단 접속되면 회선 교환기 내에서의 전송 지연이 거의 없어 실시간 전송이 가능하다.

• 데이터 전송에 필요한 전체 시간이 축적 교환 방식에 비해 길다.

• 일정한 데이터 전송률을 제공하므로 동일한 전송 속도가 유지된다.

• 전송된 데이터의 오류 제어나 흐름 제어는 사용자에 의해 수행된다.

• 길이가 긴 연속적인 데이터 전송에 용이하다.

• 공간 분할 교환 방식과 시분할 교환 방식으로 나누어지고, 시분할 교환 방식에는 TDM 버스 교환 방식, 타임 슬롯 교환 방식, 시간 다중화 교환 방식이 있다.

• **통신 과정** : 호(링크) 설정 → 데이터 전송 → 호(링크) 해제

핵심

04.3, 03.3, 02.3, 01.9, 00.7, 99.10, 99.6
14.8, 14.5, 14.3, 13.8, 13.6, 13.3, 12.8, 12.3, 11.8, 11.6, 11.3, 10.5, 10.3, 09.5, 09.3, 08.9, 08.5, 06.9, 06.5, 05.9, 05.5,

231 패킷 교환 방식

• 메시지를 일정한 길이의 패킷으로 잘라서 전송하는 방식이다.

• **패킷(Packet)** : 전송 혹은 다중화를 목적으로, 메시지를 일정한 비트 수로 분할하여 송 · 수신 측 주소와 제어 정보 등을 부가하여 만든 데이터 블록

- PAD : 비패킷형 단말기들이 패킷 교환망에 접속할 수 있도록 데이터를 패킷으로 조립하거나 분해하는 역할을 함
- 응답 시간이 빠르므로, 대화형 응용이 가능하다.
- 음성 전송보다 데이터 전송에 더 적합하다.
- 코드 및 속도 변환이 가능하다.
- 패킷망 상호 간의 접속을 위한 프로토콜은 X.75이다.
- 하나의 회선을 여러 사용자가 공유할 수 있으므로 회선 이용률이 높다.
- 통신량의 제어를 통한 망의 안전성을 높일 수 있다.
- 전송 시 교환기, 회선 등에 장애가 발생하여도 다른 정상적인 경로를 선택하여 우회할 수 있다.
- 패킷 교환 방식은 트래픽 용량이 큰 경우, 즉 데이터 교환이 많은 경우 유리하다.
- 대량의 데이터 전송 시 전송 지연이 많아진다.
- 대역폭 설정이 유동적이다.
- **패킷 교환망의 기능** : 패킷 다중화, 경로 제어, 논리 채널, 순서 제어, 트래픽 제어, 오류 제어
- **패킷 교환 방식의 종류**

가상 회선 방식	• 단말기 상호간에 논리적인 가상 통신 회선을 미리 설정하여 송신지와 수신지 사이의 연결을 확립한 후에 설정된 경로를 따라 패킷들을 순서적으로 운반하는 방식 • 통신이 이루어지는 컴퓨터 사이의 데이터 전송의 안정, 신뢰성이 보장됨 • 패킷의 송 · 수신 순서가 같음 • 통신 과정 : 호 설정 → 데이터 전송 → 호 해제
데이터 그램 방식	• 연결 경로를 설정하지 않고 인접한 노드들의 트래픽(전송량) 상황을 감안하여 각각의 패킷들을 순서에 상관없이 독립적으로 운반하는 방식 • 패킷마다 전송 경로가 다르며, 송 · 수신 순서가 다를 수 있음 • 부하가 적거나 간헐적인 통신에 적합함

04.3, 03.8, 03.5, 02.9, 02.5, 01.6, 00.7, 00.3, 99.10, 99.8, 99.4
14.5, 14.3, 13.6, 12.5, 12.3, 11.8, 10.9, 09.8, 08.5, 08.3, 07.9, 07.5, 07.3, 06.5, 06.3, 05.9, 05.5, 05.4, 05.3, 04.9, 04.5,

핵심 232 LAN(근거리 통신망)

- 광대역 통신망과는 달리 학교, 회사, 연구소 등 한 건물이나 일정 지역 내에서 컴퓨터나 단말기들을 고속 전송 회선으로 연결하여 프로그램 파일 또는 주변장치를 공유할 수 있도록 한 네트워크 형태이다.
- 단일 기관의 소유, 제한된 지역 내의 통신이다.
- 광대역 전송 매체의 사용으로 고속 통신이 가능하다.
- 경로 선택이 필요 없고, 오류 발생률이 낮다.
- 전송 매체로 꼬임선, 동축 케이블, 광섬유 케이블 등을 사용한다.
- 전송 방식으로 베이스밴드와 브로드밴드 방식이 있다.
- 망의 구성 형태에 따라 스타형, 버스형, 링형, 트리형, 망형으로 분류할 수 있다.
- LAN의 계층 구조는 물리 계층과 데이터 링크 계층으로 나누어진다.

물리 계층	OSI 7계층의 물리 계층과 동일한 기능을 제공함
데이터 링크 계층	• 매체 접근 제어(MAC) 계층과 논리 링크 제어(LLC) 계층으로 나누어짐 • 매체 접근 제어(MAC) 방식의 종류 : CSMA, CSMA/CD, 토큰 버스, 토큰 링

- IEEE 802의 주요 표준 규격

802.1	전체의 구성	802.5	토큰 링 방식
802.2	논리 링크 제어(LLC)	802.6	도시형 통신망(MAN), DQDB(이중 버스 통신망)
802.3	CSMA/CD 방식		
802.4	토큰 버스 방식	802.11	무선 LAN

12.5, 12.3, 11.6, 10.9, 06.5, 05.9, 05.4, 05.3, 04.9, 03.5, 02.9, 01.9, 00.5, 00.3, 99.10

핵심 233 CSMA/CD 방식

- 통신 회선이 사용중이면 일정 시간 동안 대기하고, 통신 회선상에 데이터가 없을 때에만 데이터를 송신하며, 송신중에도 전송로의 상태를 계속 감시한다.
- 버스형 또는 성형 LAN에 가장 일반적으로 사용된다.
- Ethernet의 표준이다.
- IEEE 802.3의 표준 규약이다.
- 일정 길이 이하의 데이터를 송신할 경우 충돌을 검출할 수 없다.
- 전송량이 적을 때 매우 효율적이고 신뢰성이 높다.
- 전송량이 많아지면 채널의 이용률이 떨어지고 전송 지연 시간이 급격히 증가한다.
- 충돌 발생 시 다른 노드에서는 데이터를 전송할 수 없으며, 지연 시간을 예측하기 어렵다.
- 송신 도중 충돌이 발생하면 송신을 중지하고, 모든 노드에 충돌을 알리는 재밍 신호를 전송한 후 일정 시간이 지난 다음 데이터를 재송신한다.
- 모든 제어기는 동등한 액세스 권한이 있다.

핵심 13.3, 12.8, 11.6, 11.3, 10.3, 07.3, 05.9, 05.4, 05.3, 04.3, 00.10, 99.8

234 네트워크 관련 장비

- 허브(Hub) : 한 사무실이나 가까운 거리의 컴퓨터들을 연결하는 장치로, 각 회선을 통합적으로 관리하며 신호 증폭 기능을 하는 리피터의 역할도 함
- 리피터(Repeater) : 물리 계층의 장비로, 전송되는 신호를 재생해줌
- 브리지(Bridge) : 데이터 링크 계층의 장비로, LAN과 LAN을 연결하거나 LAN 안에서의 컴퓨터 그룹을 연결함
- 라우터(Router) : 네트워크 계층의 장비로, 동종의 LAN과 LAN의 연결 및 경로 선택, 서로 다른 LAN이나 LAN과 WAN을 연결함
- 게이트웨이(Gateway) : 프로토콜 구조가 전혀 다른 네트워크(망)의 연결을 수행하는 장비로, 세션 계층, 표현 계층, 응용 계층 간을 연결하여 데이터 형식 변환, 주소 변환, 프로토콜 변환 등을 수행함

핵심 09.3, 07.5, 06.3, 05.3, 03.5, 02.9, 02.5, 00.5, 99.10, 99.6

235 VAN(부가 가치 통신망)

- 공중 통신 사업자로부터 통신 회선을 임대하여 하나의 사설망을 구축하고 이를 통해 정보의 축적, 가공, 변환 처리 등 가치를 첨가한 후 불특정 다수를 대상으로 서비스를 제공하는 통신망이다.
- 계층 구조 : 기본 통신 계층, 네트워크 계층, 통신 처리 계층, 정보 처리 계층,
- 기능 : 전송 기능, 교환 기능, 통신 처리 기능, 정보 처리 기능
- 통신 처리 기능은 축적 교환 기능과 변환 기능으로 나누어진다.

축적 교환 기능	전자 사서함, 데이터 교환, 동보 통신, 정시 수집, 정시 배달
변환 기능	속도 변환, 프로토콜 변환, 코드 변환, 데이터 형식(Format) 변환, 미디어 변환

잠깐만요 ! **프로토콜 변환** : 서로 다른 네트워크 간에 또는 서로 다른 기종 간에 통신이 가능하도록 통신 절차를 변환하는 기능

핵심 14.3, 10.3, 05.5, 05.4, 05.3, 04.5, 04.3, 03.8, 02.9, 02.5, 02.3, 01.6, 01.3, 00.10, 99.4

236 ISDN(종합 정보 통신망)

- 음성, 문자, 화상 등의 다양한 통신 서비스를 하나의 디지털 통신망을 근간으로 하여 종합적으로 제공할 수 있도록 통합한 것이다.
- 통신 방식 및 전송로가 모두 디지털 방식이다.
- 단일 통신망으로 음성, 데이터, 영상 등의 다양한 서비스를 제공한다.
- 고속 통신이 가능하며, 확장성과 재배치성이 좋다.
- 64Kbps 1회선 교환 서비스를 기본으로 한다.
- ISDN의 통신 서비스

베어러 서비스	• 단말기가 전송하는 정보를 변형 없이 그대로 전달만 하는 서비스 • 회선 교환, 패킷 교환 등 하위 계층의 기능만을 제공함
텔레 서비스	• 베어러 서비스를 기본으로 고도의 기능을 부가하여 제공하는 서비스 • 실제로 단말기를 조작하고 통신하는 이용자 측에서 본 서비스 • 상위 계층의 전화, 팩스, 텔레텍스, 비디오텍스, 텔렉스, 원격 회의 등의 서비스와 하위 계층의 정보 전송, 액세스 서비스 등을 모두 제공함
부가 서비스	베어러 서비스나 텔레 서비스에 발신 번호 표시, 수신자 부담, 통화 대기 등의 기능을 부가하여 서비스 이용률을 높이는 서비스

핵심 14.5, 13.8, 09.8, 09.5, 09.3, 06.5, 06.3, 05.9, 05.5, 03.8, 03.5, 02.9, 02.5, 01.3, 00.3, 99.10, 99.8

237 ISDN의 구조 및 참조점

- 주요 채널의 종류

채널	채널 속도		용도
B	64Kbps		• 디지털 정보용 채널 • 사용자의 정보를 전송하기 위한 채널 • PCM화된 디지털 음성이나 회선 교환에 의한 제어 신호, 패킷 교환에 의한 정보의 전송에 이용됨
D	16Kbps 64Kbps		• 디지털 신호용 채널, 제어용 신호 메시지 전달 역할 • 16Kbps 이하 저속의 패킷 교환에 의한 정보 전송을 위해 이용함
H	H_0	384Kbps	• 고속의 디지털 정보용 채널 • B 채널에서 제공되는 모든 방식의 정보를 고속으로 전송
	H_{11}	1,536Kbps	
	H_{12}	1,920Kbps	

- 기본 속도 인터페이스(BRI) : 2B + D + 오버헤드 = 2 × 64 + 16 + 48 = 192Kbps

• **참조점** : ISDN을 구성하는 각 요소 간의 인터페이스를 구분하는 기능으로, 기준점, 접속점, 분계점이라고도 함. 정보 통신망 상호간을 연결할 때 시설, 운영 및 유지보수의 책임 한계를 구분하기 위한 접속점이 됨

핵심 238 인터넷(Internet)

12.8, 10.3, 09.5, 03.3, 01.9

• TCP/IP 프로토콜을 기반으로 하여 전세계 수많은 컴퓨터와 네트워크들이 연결된 광범위한 컴퓨터 통신망이다.

• ARPANET에서 시작되었으며, 유닉스 운영체제를 기반으로 한다.

• 인터넷에 연결된 모든 컴퓨터는 고유한 IP 주소를 갖는다.

• 컴퓨터 또는 네트워크를 서로 연결하기 위해서는 브리지, 라우터, 게이트웨이가 사용된다.

• **인터넷 서비스** : TCP/IP의 응용 계층에서 제공함

WWW	• 텍스트, 그림, 동영상, 음성 등 인터넷에 존재하는 다양한 정보를 거미줄처럼 연결해 놓은 종합 정보 서비스 • HTTP 프로토콜을 사용하는 하이퍼텍스트 기반으로 되어 있음
전자 우편 (E-mail)	• 인터넷을 통해 다른 사람과 편지뿐만 아니라 그림, 동영상 등 다양한 형식의 데이터를 주고받을 수 있도록 해주는 서비스 • 전자 우편에 사용되는 프로토콜 : SMTP, POP3, MIME
FTP	컴퓨터와 컴퓨터 또는 컴퓨터와 인터넷 사이에서 파일을 주고받을 수 있도록 하는 원격 파일 전송 프로토콜
Telnet	• 멀리 떨어져 있는 컴퓨터에 접속하여 자신의 컴퓨터처럼 사용할 수 있도록 해주는 서비스 • 프로그램을 실행하는 등 시스템 관리 작업을 할 수 있는 가상의 터미널(Virtual Terminal) 기능을 수행함
USENET	분야별로 공통의 관심사를 가진 인터넷 사용자들이 서로의 의견을 주고받을 수 있게 하는 서비스

핵심 239 위성 통신

11.3, 09.8, 09.3, 07.9, 06.9, 05.4, 05.3, 03.5, 02.9, 02.3, 00.3

• 지상에서 쏘아올린 마이크로 주파수를 통신 위성을 통해 변환, 증폭한 후 다른 주파수로 지상에 송신하는 방식이다.

• 위성 통신에서 사용하고 있는 주파수 대역은 3~30 GHz의 극초단파(SHF)이다.

• 통신 위성은 약 36,000km 정도의 정지 궤도 상에 위치하여 지구의 자전 속도로 운행한다.

• 한 대의 통신 위성은 지구 표면의 약 42.4% 지역을 커버할 수 있으므로 이론상 최소한 3개의 정지 위성만 있으면 극 지역을 제외한 지구상의 어느 지점과도 통신이 가능하다.

• 대역폭이 넓어 고속 · 대용량 통신이 가능하고, 통신 비용이 저렴하다.

• 오류율이 적어 고품질의 정보 전송이 가능하다.

• 통신 범위가 넓으며, 전송 비용이 거리에 관계없이 일정하다.

• 신호가 한 노드에서 다음 노드까지 도달하는데 걸리는 전송(전파) 지연 시간이 길며, 보안성이 취약하다.

• 위성 통신 시스템은 통신 위성, 지구국, 채널로 구성된다.

• **다중 접속 방식** : FDMA(주파수 분할 다중 접속), TDMA(시분할 다중 접속), CDMA(코드 분할 다중 접속)

핵심 240 셀룰러(Cellular) 시스템

13.8, 11.6, 10.9, 10.3, 08.5, 08.3, 07.9

• 서비스 지역을 셀(Cell)이라는 여러 개의 영역으로 나눈 후 각 셀마다 하나의 기지국을 설치하여 인접 셀 간에는 상호 간섭을 받지 않도록 하고, 어느 정도 떨어진 셀 간에는 동일 주파수 채널을 사용하도록 하는 방식이다.

• **셀룰러 시스템의 특징**

– 주파수 재사용(Frequency Reuse) : 인접하지 않는 셀은 같은 주파수를 사용함으로써 통화량을 늘리고, 회선의 사용을 극대화할 수 있음

– 핸드오프(Hand-off, Hand-over) : 가입자가 서비스 중인 기지국 영역을 벗어나 다른 기지국으로 이동할 때, 통화가 단절되지 않도록 통화 채널을 자동으로 전환하는 기능

– 로밍(Roaming) 서비스 : 가입자가 자신의 홈 교환국(서비스 지역)을 벗어나 타교환국에 있어도 서비스를 받을 수 있는 것을 의미함. 로밍은 한 사업자의 교환국 사이에서뿐만 아니라 사업자 간, 국가 간에도 가능함

• **셀룰러 시스템의 구성** : 공중 전화망(PSTN), 이동 전화 교환국(MTSO), 이동국(MS), 기지국(BS), 홈 가입자 위치 등록기(HLR), 방문자 위치 등록기(VLR)

핵심 241 14.8, 14.5, 12.5, 12.3, 10.9, 10.5, 09.3, 07.3, 06.3, 04.5, 02.5

241 광대역 종합 정보 통신망(B-ISDN)

광대역 종합 정보 통신망(B-ISDN)

- 광대역 전송 방식과 광대역 교환 방식을 통하여 문자, 음성, 영상 등 다양한 형태의 통신 서비스를 쌍방향으로 제공하는 광대역 ISDN이다.
- 고속 교환 기술을 통해 150 ~ 600Mbps 이상의 고속 전송이 가능하다.
- 전송 방식은 ATM(Asynchronous Transfer Mode; 비동기 전송 모드)을 사용한다.

ATM(비동기 전송 모드)

- 교환 전화 등에 쓰이는 회선 교환과 패킷 교환의 장점을 결합한 교환 및 다중화 기술이다.
- ATM은 모든 데이터를 셀(Cell)로 분할하여 비동기식 시분할 다중화 방식으로 전송한다.
- 셀의 크기는 53Byte(5Byte : 헤더(Header), 48Byte : 사용자 정보(Payload))이며, 고정 길이이다.

잠깐만요 ! 셀의 크기 단위로 Byte 대신 옥테드(Octet)를 사용하기도 합니다.

- ATM은 보낼 정보가 있을 때만 셀에 정보를 실어 전송하고 나머지는 Idle Cell을 보낸다. 이때 Idle Cell은 망에서 폐기되므로 회선의 낭비를 막을 수 있다.
- 음성, 문자, 영상 등 모든 정보를 통합화하여 서비스할 수 있다.
- 고속, 광대역, 멀티미디어 통신을 모두 수용할 수 있는 기술이다.
- 정보량에 따라 유동적으로 통신 채널의 대역폭을 바꿀 수 있다.
- 속도가 빠르고, 효율이 높다.
- 고정 길이의 셀 단위로 전송하므로 전송 지연 시간을 예측할 수 있고, 처리가 간단하며 신뢰성은 더 높다.
- 다양한 서비스를 실시간으로 지원한다.

핵심 242 13.3, 10.3, 09.3, 06.9, 06.5, 05.9, 05.4, 04.5, 03.5, 02.3, 01.9, 01.3

242 뉴미디어(New Media)

- '새로운'이란 뜻의 New와 '정보 전달 수단'이란 뜻의 Media의 합성어로, 최근의 정보 통신 기술의 발달로 새롭게 나타나게 되는 다양한 매체들을 의미한다.
- 특정 계층을 목표로 한 쌍방향성, 탈대중화, 비동시성, 정보의 다양성, 다채널성, 고속성, 대용량성, 광대역성의 특징을 갖는다.
- 기존의 미디어와 융합되어 발전한다.
- 유선계와 무선계로 분류

유선계	CATV, 비디오텍스, VRS, 원격 회의, ARS, 텔레비전 전화, 팩시밀리, 퍼스널 컴퓨터 통신, LAN, VAN, ISDN 등
무선계	위성 통신, 텔레텍스트, HDTV, PCM 음성 방송, 팩시밀리 방송, 개인 휴대 통신 등
독립계(패키지계)	비디오 디스크, 디지털 오디오 디스크, VTR, 광디스크 등

- 방송계와 통신계로 분류

방송계	CATV, PCM 음성 방송, 텔레텍스트, HDTV, 팩시밀리 방송 등
통신계	원격 회의, 비디오텍스, 텔레텍스, VRS, 개인 휴대 통신, 퍼스널 컴퓨터 통신, LAN, VAN, ISDN 등

핵심 243 14.8, 14.5, 13.8, 12.8, 12.5, 11.3, 10.9, 10.3, 09.8, 08.9, 06.9, 06.3, 05.5, 05.3, 04.9, 04.5, 03.8, 03.5, 02.9, 01.6, 99.10

243 주요 뉴미디어(New Media)

CATV	• 원래 난시청 해소를 목적으로 설치했던 공동 시청 안테나를 이용하여 수신한 TV 신호를 일정한 전송로를 통하여 사용자에게 제공함 • 양방향 통신이 가능함 • 사용자의 범위가 한정적임 • 다채널로서 방송뿐만 아니라 종합 정보 서비스가 가능함 • 전송로는 동축 케이블이나 광섬유 케이블을 사용함 • 기존 TV와 방송 방식이 동일하여 기존 TV를 단말장치로 사용할 수 있음 • 헤드앤드(Head-End), 전송로, 가입자 설비(단말장치)로 구성됨
비디오텍스 (Videotex)	• 각종 정보를 모아 데이터베이스(DB)를 구축하고, 전화망을 통해 TV나 단말장치에 접속하여 필요한 정보를 문자나 그림의 형태로 검색할 수 있도록 하는 서비스 • 정보 검색, 거래 처리, 메시지 전달, 예약 업무, 원격 감시 서비스 등이 있음 • 대화형 양방향 미디어로서, 요구하는 정보를 즉시 제공받을 수 있음 • 문자 및 도형 정보의 표현 형식 : 모자이크 방식, 지오메트릭 방식, 포토그래픽 방식, DRCS

HDTV(High Definition TV)	• 기존의 TV 주사선을 늘리고 주파수 대역폭을 확대하여 선명한 화상과 양질의 음성을 제공하는 TV • 위성 TV 방송, TV 회의 등의 새로운 매체의 단말장치로 사용됨
텔레텍스트 (Teletext)	• TV 전파의 빈틈을 이용하여 TV 방송과 함께 문자나 도형 정보를 제공하는 문자 다중 방송 • 일기예보, 프로그램 안내나 방송되는 프로의 세부 설명, 교통안내 등 방송국에서 제공하는 정보를 일방적으로 수신하는 형태
텔레텍스 (Teletex)	• 워드프로세서 전용기와 같이 문서 작성과 편집 기능을 갖는 기기에 통신 기능을 부가하여 공중 전화망이나 공중 데이터망을 통해서 문서를 교환하는 시스템 • 문서를 작성중이거나 부재중에도 수신이 가능
텔렉스 (Telex)	• 문자, 숫자 및 기호 등의 정보를 텔렉스 교환기를 사용해서 전송하는 시스템 • 최초의 문자 전송 시스템으로 가입 전신이라고도 함 • 사용자 부재중에도 통신이 가능하며, 사용하는 문자수에 제한이 있음
텔레미터링 시스템 (Tele-Metering System)	원거리에서 전기나 수도, 가스 등의 사용량을 수시로 검침하여 자동으로 계산해 주는 원격 검침 시스템
MHS (Message Handling System)	• 개인용 컴퓨터, 팩시밀리, 텔렉스 등 통신수단에 관계 없이 상대방의 통신수단별 번호만 알면 국내외 어디서나 메시지를 교환할 수 있는 시스템 • 문자, 도형, 화상, 음성 등 다양한 메시지의 생성, 전송, 축적, 수신에 대한 전반적인 서비스를 수행하는 전자우편(E-Mail) 시스템 • 사용자 처리기(UA), 메시지 전송 처리기(MTA), 메시지 축적기(MS)로 구성됨

핵심

11.3, 09.5, 05.4, 03.8, 02.5, 00.5, 99.8

244 멀티미디어(Multimedia)

- 다중 매체(정보 전달 수단)를 의미하는 것으로, 텍스트, 그래픽, 사운드, 동영상, 애니메이션 등의 다양한 매체를 디지털 데이터로 통합하여 전달한다.
- 특징 : 디지털화, 쌍방향성, 비선형성, 정보의 통합성
- JPEG : 정지 영상 압축의 국제 표준 방식
- MPEG : 동영상 압축을 위한 국제 표준 규격
- MPEG 규격

MPEG-1	CD-ROM과 같이 전송 속도가 약 1.5Mbps인 저장 매체에 가정용 VTR 수준의 동영상과 음향을 압축 · 저장하기 위한 것으로, CD-I나 비디오 CD 등이 이 규격을 따르고 있음
MPEG-2	차세대 텔레비전 방송이나 ISDN, 케이블망 등을 이용한 영상 전송을 위하여 제정되었으며, HDTV, 위성 방송, DVD 등이 이 규격을 따르고 있음
MPEG-4	통신, PC, 방송 등을 결합하는 복합 멀티미디어 서비스의 통합 표준을 위한 것으로, 공중망이나 무선 이동 통신 등을 대상으로 하며, 특히 IMT-2000 환경에서 영상 정보 압축 전송 시 필수적인 요소로 인정받고 있음
MPEG-7	멀티미디어 정보 검색이 가능한 동영상, 데이터 검색 및 전자상거래 등에 사용하도록 개발되었음
MPEG-21	위의 MPEG 기술들을 통합해 디지털 콘텐츠의 제작, 유통, 보안 등 전 과정을 관리할 수 있는 기술임