수험생의 마음으로 만든 책! 시나공 시리즈

2014 시나공

정답 및 해설

정보처리산업기사 필기

길벗R&D 지음

길벗
독자의 1초까지 아껴주는
정성을 만나 보세요.

Contents

예상문제은행 정답 및 해설

예상문제은행

정답 및 해설

1과목 데이터베이스

1장 정답 및 해설 — 데이터베이스의 개념

1.③	2.②	3.④	4.②	5.④	6.④	7.③	8.②	9.③	10.④	11.②	12.③	13.②	14.④	15.①
16.④	17.③	18.③	19.②	20.②	21.②	22.④	23.①	24.②	25.②	26.④	27.②	28.③	29.③	30.③
31.①	32.②	33.①	34.②	35.①	36.①	37.④	38.③	39.④						

1. Section 001

③번은 정보 시스템에 대한 설명이다.

정보 시스템의 정의

- 정보 시스템이란 조직체에 필요한 DATA를 수집, 저장해 두었다가 필요 시에 처리해서 의사결정에 유용한 정보를 생성하고 분배하는 수단을 말한다.
- 정보 시스템은 사용하는 목적에 따라 경영 정보 시스템, 군사 정보 시스템, 인사 행정 정보 시스템, 의사 결정 지원 시스템 등으로 구분되어 사용된다.

2. Section 001

- **자료 처리 시스템** : 정보 시스템이 사용할 자료를 처리하는 정보 시스템의 서브 시스템
- **전문가 시스템** : 전문지식을 컴퓨터에 데이터베이스화하여 비전문가의 질문에 대한 답을 컴퓨터가 제시하는 시스템
- **응용 시스템** : 조직에서 특정한 한 부서에 정보를 제공하기 위해서 운영되는 정보 시스템의 서브 시스템, 이 응용 시스템을 운영하기 위해 구현된 프로그램을 응용 프로그램이라함

3. Section 003

- **종속성으로 인한 문제점**
  - 종속성이란 응용 프로그램과 데이터 파일이 상호 의존적인 관계를 말한다.
  - 응용 프로그램과 데이터 파일이 상호 의존적인 관계에서는 데이터 파일이 보조기억장치에 저장되는 방법이나 저장된 데이터의 접근 방법을 변경할 때는 응용 프로그램도 같이 변경하여야 한다.
- **중복성으로 인한 문제점**
  - **일관성** : 중복된 데이터 간에 내용이 일치하지 않는 상황이 발생하여 일관성이 없어짐
  - **보안성** : 중복되어 있는 모든 데이터에 동등한 보안 수준을 유지하기가 어려움
  - **경제성** : 저장공간의 낭비와 동일한 데이터의 반복 작업으로 인해 비용이 증가함
  - **무결성** : 제어의 분산으로 인해 데이터의 정확성을 유지할 수 없음

4. Section 003

기존 파일 처리 방식이 데이터베이스를 이용하는 것에 비해 처리 속도가 느린 것은 아니다. 문제점은 3번 해설을 참조할 것

5. Section 002

데이터베이스의 정의

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

※ 파일 캐비닛의 데이터는 데이터베이스에 포함되지 않는다.

6. Section 003

DBMS의 필수 기능

- **정의(조직)(Definition) 기능** : 모든 응용 프로그램들이 요구하는 데이터 구조를 지원하기 위해 데이터베이스에 저장될 데이터의 형(Type)과 구조에 대한 정의, 이용 방식, 제약 조건 등을 명시

하는 기능

- 조작(Manipulation) 기능 : 데이터 검색, 갱신, 삽입, 삭제 등을 체계적으로 처리하기 위해 사용자와 데이터베이스 사이의 인터페이스 수단을 제공하는 기능
- 제어(Control) 기능
  - 데이터베이스를 접근하는 갱신, 삽입, 삭제 작업이 정확하게 수행되어 데이터의 무결성이 유지되도록 제어해야 한다.
  - 정당한 사용자가 허가된 데이터만 접근할 수 있도록 보안(Security)을 유지하고 권한(Authority)을 검사할 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.

7. Section 002

단위 프로그램의 자료를 독립적으로 관리하기 위한 것은 파일 시스템이다. 데이터베이스는 모든 응용 프로그램들이 공동으로 사용할 수 있도록 통합된 데이터를 관리한다.

8. Section 003

해석 : 백업은 데이터베이스가 장비 고장 또는 다른 비상시에 보존되도록 복사하는 활동이다.

9. Section 004

해석 : 이것은 데이터가 기억장치에 배치되는 방법을 정의한다. 이것은 시스템 프로그래머나 시스템 디자이너 관점에서 데이터베이스의 물리적인 저장 구조를 묘사한다.

10. Section 003

논리적 독립성이란 응용 프로그램과 데이터베이스를 독립시킴으로써, 데이터의 논리적 구조를 변경시키더라도 응용 프로그램은 변경시키지 않아도 된다는 것과 그 반대의 경우를 의미한다.

11. Section 003

물리적 독립성은 응용 프로그램과 보조기억장치 같은 물리적 장치를 독립시킴으로써, 데이터베이스 시스템의 성능 향상을 위해 새로운 디스크를 도입하더라도 응용 프로그램에는 영향을 주지 않고 데이터의 물리적 구조만을 변경시키는 것을 의미한다.

12. Section 003

데이터베이스의 장점

- 데이터의 중복을 피할 수 있다.
- 저장된 자료를 공동으로 이용할 수 있다.
- 데이터의 일관성을 유지할 수 있다.
- 데이터의 무결성을 유지할 수 있다.
- 보안을 유지할 수 있다.
- 데이터를 표준화할 수 있다.
- 데이터를 통합하여 관리할 수 있다.
- 항상 최신의 데이터를 유지한다.
- 데이터의 실시간 처리가 가능하다.
- 데이터의 논리적, 물리적 독립성이 보장된다.

※ 특정한 응용 업무 하나만을 위한다면 해당 업무에 최적화된 파일 시스템이 더 적합하다.

13. Section 003

데이터베이스의 단점

- 데이터베이스의 전문가가 부족하다.
- 전산화 비용이 증가한다.
- 대용량 디스크로의 집중적인 Access로 과부하(Overhead)가 발생한다.
- 파일의 예비(Backup)와 회복(Recovery)이 어렵다.
- 시스템이 복잡하다.

※ 데이터베이스는 중복된 데이터를 제거함으로써 기억공간을 효율적으로 사용할 수 있다.

14. Section 002

데이터베이스 시스템의 구성 요소

- 데이터베이스
- 스키마
- DBMS(데이터베이스 관리 시스템)
- 데이터베이스 언어
- 데이터베이스 컴퓨터
- 데이터베이스 사용자

15. Section 003

DBMS의 필수 기능

- 정의(조직) 기능 : 데이터베이스에 저장될 데이터의 형과 구조에 대한 정의, 이용 방식, 제약 조건 등을 명시하는 기능
- 조작 기능 : 데이터의 검색, 갱신, 삽입, 삭제 등을 체계적으로 처리하기 위해 데이터 접근 수단 등을 정하는 기능
- 제어 기능 : 데이터의 정확성과 안전성을 유지하기 위한 무결성, 보안 및 권한 검사, 병행 수행 제어 등의 기능을 정하는 기능

※ 응용 프로그램 유지 관리는 DBMS의 역할이 아니고 데이터베이스 관리자가 할 업무이다.

16. Section 003

복구(Recovery) 기능의 두 가지 형태

- 하나는 디스크에 저장된 파일이 손상을 입었을 때 예비(Backup)시켜 두었던 파일을 재저장시켜서 복구하는 것인데, 이것은 운영체제의 파일 시스템이 하는 기능이다.
- 다른 하나는 트랜잭션들의 병행수행에서 문제가 발생했을 때 병행 수행된 모든 트랜잭션들이 갱신한 내용들을 취소시키고 원래의 상태로 회복시키는 기능이다.

※ 따라서 이 문제는 엄격히 말하면 정답이 없다. 굳이 정답을 정하려면 무결성 유지, 보안과 권한 검사, 병행제어 기능은 DBMS만의 기능이지만, 복구는 두 가지 형태 중 파일 시스템의 기능으로 간주하여 ④번을 답으로 할 수 있다.

17. Section 003

DBMS의 장점

- 데이터의 논리적, 물리적 독립성이 보장된다.
- 데이터의 중복을 피할 수 있다.
- 저장된 자료를 공동으로 이용할 수 있다.
- 데이터의 일관성을 유지할 수 있다.
- 데이터의 무결성을 유지할 수 있다.
- 보안을 유지할 수 있다.
- 데이터를 표준화할 수 있다.
- 데이터를 통합하여 관리할 수 있다.
- 항상 최신의 데이터를 유지한다.
- 데이터의 실시간 처리가 가능하다.

DBMS의 단점

- 데이터베이스의 전문가가 부족하다.
- 전산화 비용이 증가한다.
- 대용량 디스크로의 집중적인 Access로 과부하(Overhead)가 발생한다.
- 파일의 예비(Backup)와 회복(Recovery)이 어렵다.
- 시스템이 복잡하다.

※ DBMS는 파일 시스템에 비해 구축 비용 및 시스템 운영 비용이 많이 든다.

18. Section 003

DBMS(DataBase Management System)의 정의

- DBMS란 사용자와 데이터베이스 사이에서 사용자의 요구에 따라 정보를 생성해 주고, 데이터베이스를 관리해 주는 소프트웨어이다.
- DBMS는 기존의 파일 시스템이 갖는 데이터의 종속성과 중복성의 문제를 해결하기 위해 제안된 시스템으로, 모든 응용 프로그램들이 데이터베이스를 공용할 수 있도록 관리해 준다.
- DBMS는 데이터베이스의 구성, 접근 방법, 관리 유지에 대한 모든 책임을 진다.

※ 데이터베이스 스키마를 데이터베이스 파일로 생성하는 것은 DBMS의 역할이지만, 데이터 모델링을 수행하고 데이터베이스 스키마를 생성하는 것은 사람(DBA)의 역할이다.

20. Section 003

해석 : 데이터베이스를 관리하기 위한 검색 프로그램과 저장소의 모임이다. 이것은 사용자의 질의에 대한 응답으로 데이터베이스로부터 조직하고, 처리하고, 데이터 요소들을 선택하여 보여준다.

22. Section 004

스키마

- 스키마는 데이터베이스의 구조와 제약 조건에 관한 전반적인 명세(Specification)를 기술(Description)한다.
- 스키마는 데이터베이스를 구성하는 데이터 개체(Entity), 속성(Attribute), 관계(Relationship) 및 데이터 조작 시 데이터 값들이 갖는 제약 조건 등에 관해 전반적으로 정의한다.
- 데이터베이스 내에 있는 데이터의 논리적 단위 사이의 관계성을 표현한다.
- 다른 이름으로 메타 데이터(Meta-Data)라고도 한다.

※ 데이터베이스 스키마에 자료를 처리할 응용 프로그램 구조를 표현하지는 않는다.

23. Section 004

- **외부 스키마와 개념 스키마 간의 접속** : 응용 인터페이스
- **내부 스키마와 개념 스키마 간의 접속** : 저장 인터페이스

25. Section 004

저장된 레코드 및 필드의 순서, 색인, 해시 주소, 포인터 등의 상세한 사항은 내부 스키마에 기술한다.

26. Section 004

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

- 외부 스키마는 사용자나 응용 프로그래머가 각 개인의 입장에서 필요로 하는 데이터베이스의 논리적 구조를 정의한 것이다.
- 외부 스키마(External Schema)는 전체 데이터베이스의 한 논리적인 부분으로 볼 수 있으므로 서브 스키마(Subschema)라고도 한다.
- 사용자가 개별적으로 사용하는 뷰를 나타내는 것으로, 전체 데이터베이스의 일부일 수도 있고 전체일 수도 있다.
- 같은 데이터베이스에 대해서도 서로 다른 관점을 정의할 수 있도록 허용한다.
- 일반 사용자는 질의어(SQL)를 이용하여 DB를 쉽게 사용할 수 있다.
- 응용 프로그래머는 COBOL, C 등의 언어를 사용한다.

※ 보안성 검사, 무결성 검사와 같은 부가적인 특징을 포함하는 스키마는 개념 스키마이다.

27. Section 004

논리적인 데이터베이스 전체의 구조를 나타내며, 데이터베이스 파일(File)에 저장되어 있는 레코드(Record)와 데이터 항목(Item)의 이름을 부여하고 그들 사이에 관계의 구조를 나타내는 스키마(Schema)는 개념 스키마이다.

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

- 개념 스키마는 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.
- 단순히 스키마(Schema)라고하면 개념 스키마를 의미한다.
- 기관이나 조직체의 관점에서 데이터베이스를 정의한 것이다.
- 데이터베이스 관리자(DBA)에 의해서 구성된다.

28. Section 005

단말 사용자가 데이터베이스(DB)에 대한 정보를 요구하기 위해 사용하는 언어는 질의어이다.

- DDL : 테이블이나 뷰를 구축 또는 삭제하는 기능을 가진 언어
- DCL : 무결성(Integrity) 유지, 보안(Security)과 권한(Authority) 검사, 트랜잭션의 회복(Recovery), 병행제어(Concurrency Control) 등의 기능을 지원하는 언어
- DML : 응용 프로그램을 통하여 사용자가 DB의 데이터를 실질적으로 조작할 수 있는 기능을 가진 언어

29. Section 005

데이터 사전에는 단순 데이터가 아닌 DB 구조, 데이터 형식, 접근 방식 등 DB 구축에 관한 자료가 저장되어 있다.

30. Section 005

데이터 조작어의 조건

- 사용하기 쉽고 자연 언어에 가까워야 한다.
- 데이터에 대한 연산뿐만 아니라 뷰 내의 데이터나 데이터 간의 관계를 정확하고 완전하게 명시할 수 있어야 한다.
- 데이터 언어의 효율적인 구현을 지원해야 한다. 즉 데이터 언어의 구문이 DBMS가 제공하는 기본적인 연산과 관련을 갖도록 해야 한다.

31. Section 005

해석 : 다음 중 사용자가 데이터베이스를 만들고 고유의 스키마를 생성할 수 있도록 해주는 언어는 무엇인가?
사용자가 데이터베이스를 만들고, 고유의 스키마를 생성할 수 있도록 해주는 언어는 데이터 정의어이다.

32. Section 005

질의어는 터미널에서 주로 이용하는 비절차적 데이터 언어이다.

33. Section 006

- ② : 단말 사용자
- ③,④ : DBA(데이터베이스 관리자)

34. Section 006

데이터베이스 관리 시스템(DBMS)은 데이터베이스를 관리하는 소프트웨어로 DBA는 DBMS를 이용하여 데이터베이스를 설계하는 것이지 DBA가 DBMS를 설계하는 것은 아니다.

35. Section 006

데이터베이스 관리자(DBA)는 주로 데이터 제어어(DCL)를 이용하여 데이터베이스의 무결성을 유지한다.

36. Section 006

해석 : DBMS를 사용하는 중요한 이유 중 하나는 데이터와 프로그

램이 그들의 데이터에 접근할 수 있도록 중앙에서 제어하기 위해서이다.

DBA의 역할

- 데이터베이스 설계와 조작에 대한 책임
  - 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(Backup) 조치와 회복(Recovery)에 대한 전략 수립
  - 무결성을 위한 제약 조건의 지정
  - 데이터 사전의 구성과 유지 관리
  - 사용자의 변화 요구와 성능 향상을 위한 데이터베이스의 재구성
- 행정 책임
  - 사용자의 요구와 불평의 청취 및 해결
  - 데이터 표현 방법의 표준화
  - 문서화에 대한 기준 설정
- 시스템 감시 및 성능 분석
  - 변화 요구에 대한 적응과 성능 향상에 대한 감시
  - 시스템 감시 및 성능 분석
  - 자원의 사용도와 병목 현상 조사
  - 데이터 사용 추세, 이용 형태 및 각종 통계 등을 종합, 분석

37. Section 006

해석 : 응용 프로그래머가 새로운 형식의 레코드를 생성하거나 이전 레코드에 새로운 항목을 추가 또는 확장하려면 데이터베이스 관리자에게 허가를 받아야 한다.

38. Section 006

응용 프로그램과 데이터베이스 사이에서 중재자로서의 역할을 담당하는 것은 DBMS이다.

39. Section 006

해석 : DBA는 데이터베이스 시스템을 관리하기 위한 개인 또는 개인 책임의 그룹이다. DBA의 임무는 다음과 같다 : 설계, 데이터베이스 시스템의 구현과 유지, 데이터베이스 시스템 이용, 그리고 데이터베이스 시스템 사용에 관한 직원들 교육

2장 정답 및 해설 — 데이터 모델링 및 설계

1.④	2.④	3.④	4.②	5.①	6.②	7.④	8.③	9.①	10.③	11.②	12.③	13.③	14.④	15.④
16.④	17.②	18.①	19.④	20.③	21.④	22.④	23.④	24.③	25.①	26.②	27.②	28.③	29.②	30.④
31.③	32.③	33.③	34.③											

1. Section 007

데이터 모델

- 데이터 모델은 현실 세계의 정보들을 컴퓨터에 표현하기 위해서 단순화, 추상화 형태로 체계적으로 표현한 개념적 모형이다.
- 현실 세계를 데이터베이스에 표현하는 중간 과정, 즉 데이터베이스 설계 과정에서 데이터의 구조를 표현하기 위해 사용되는 도구이다.
- 데이터의 구조(Schema)를 논리적으로 묘사하기 위해 사용되는 지능적 도구이다.

2. Section 007

데이터베이스 사용자의 관심 밖에 존재하는 저장소의 상세한 내용들은 물리적 데이터 모델에 대한 내용으로 데이터 모델에 물리적 모델까지 표현하지는 않는다.

3. Section 007

- 캡슐화(Capsulation), 상속(Inheritance), 다형성(Polymorphism) 등은 객체지향 프로그램 언어의 특징이며, 객체지향 프로그램 개념에 기반을 두고 있는 데이터 모델은 객체지향 데이터 모델이다.
- 객제지향 데이터 모델(Object Oriented Data Model)은 객체 및 객체 식별자, 애트리뷰트와 메소드, 클래스, 클래스 계층 및 계승 그리고 복합 개체 등의 객체지향 개념을 지원하는 데이터

모델이다.

5. Section 007

개체(Entity)

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

6. Section 007

해석 : 각각의 개체는 고유한 특성을 가지고 있다. 이것을 무엇이라 부르는가?

7. Section 012

해석 : 데이터베이스 설계 과정 중 DBMS의 모델에 맞게 구현된 데이터베이스 스키마가 결과로 산출되는 단계는 무슨 단계인가?

8. Section 007

망 모델은 그래프형으로 조직된다.

9. Section 012

개념적 설계

- 정보의 구조를 얻기 위하여 현실 세계의 무한성과 계속성을 이해하고, 다른 사람과 통신하기 위하여 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정이다.
- Attribute들로 기술된 Entity 타입과 이 Entity 타입들 간의 Relationship을 이용하여 현실 세계의 객체(Object)를 개념 세계의 개체(Entity)로 모델화한다.
- 개체-관계 도표(E-RD)로 작성한다.

10. Section 007

현실 세계를 컴퓨터에 표현하기 위해 논리적 구조로 변환하는 데 이용하는 것은 논리적 데이터 모델이다.

11. Section 007

논리적 설계(데이터 모델링)

- 논리적 데이터 모델은 필드로 기술된 데이터 타입과 이 데이터 타입들 간의 관계를 이용하여 현실 세계를 표현하는 방법이다.
- 단순히 데이터 모델이라고 하면 논리적 데이터 모델을 의미한다.
- 특정 DBMS는 특정 모델 하나만 선정하여 사용한다.
- 논리적 데이터베이스 모델은 데이터 간의 관계를 어떻게 표현하느냐에 따라 관계 모델, 계층 모델, 네트워크 모델로 구분한다.

※ Attribute들로 기술된 Entity 타입과 이 Entity 타입들 간의 Relation을 이용하여 현실 세계의 객체를 개념 세계의 개체로 모델화하는 것은 개념적 설계에 대한 설명이다.

12. Section 008

E-R 다이어그램에서 개체 타입은 사각형, 관계 타입은 마름모, 속성은 타원으로 표현한다.

E-R 도형

- **다이아몬드(마름모)** : 관계(Relationship) 타입
- **사각형** : 개체 집합
- **타원** : 속성(Attribute)
- **밑줄 타원** : 기본 키 속성
- **선, 링크** : 개체 타입과 속성을 연결

13. Section 008

Entity란 실세계에 존재하는 객체에 대해 사람이 생각하는 개념이나 정보 단위를 말한다.

14. Section 008

구성 원소들을 연결하는 링크에는 레이블을 부여할 수 없고, 1:1 관계 등의 관계 유형을 표현할 수 있다.

15. Section 008

- 관계 유형은 연결선 위에 기술한 대응수 중에서 최대 대응수끼리의 관계로 나타내므로, (1, 1)과 (1, n)에서 1:n 관계임을 알 수 있다.
- 학과는 하나의 대학에 소속되어 있다.
- 대학은 최소 1개, 최대 n개의 학과로 구성되어 있다.

16. Section 009

- **관계형 DBMS** : SQL, ORACLE, dBASE III
- **계층형 DBMS** : IMS

- 망형 DBMS : DBTG, EDBS, TOTAL

17. Section 009

관계형 데이터 모델

- 계층 모델과 망 모델의 복잡한 구조를 단순화시킨 모델
- 표(Table)를 이용해서 데이터 상호관계를 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 기본 키(Primary Key)와 이를 참조하는 외래 키(Foreign Key)로 표현한다.
- 관계 모델의 대표적인 언어는 SQL이다.
- 1:1, 1:N, M:N 관계를 자유롭게 표현할 수 있다.

18. Section 009

E-R 모델을 관계 테이블로 변환하는 방법

- 개체는 독립적인 관계로 표현한다.
- Y가 1:1 관계이면 개체 A의 기본 키를 개체 B의 외래 키로 추가하거나, 개체 B의 기본 키를 개체 A의 외래 키로 추가하여 표현한다.
- Y가 1:N 관계이면 개체 A의 기본 키를 개체 B의 외래 키로 추가하여 표현하거나 별도의 테이블로 표현한다.
- Y가 N:M 관계이면 개체 A와 B의 기본 키를 모두 포함한 별도의 테이블로 표현한다.
- 기본 키들은 밑줄을 친다.

19. Section 010

계층 데이터 모델에서는 m:n 관계를 직접 표현할 수 없으므로 두 개의 1:n 관계로 표현한다.

20. Section 010

- 계층형 데이터 모델에서 링크는 한 방향으로 완전한 함수 관계를 이룬다.
- 링크는 계층 정의 트리라는 순서 트리를 이룬다.

21. Section 010

계층 데이터 모델의 단점

- 대칭적 형태의 질의어를 대칭적인 방식으로 표현할 수 없다.
- DB에 대한 뷰가 스키마로부터 영향을 매우 많이 받는다.
- 데이터 상호 간의 유연성이 부족하다.
- 검색 경로가 한정되어 있다.
- 삽입과 삭제 연산이 매우 복잡하다.
- 다 대 다 관계를 처리하기 어렵다.

23. Section 007

- 레코드 : 파일 시스템에서 개체를 나타내는 용어
- 튜플 : 관계 DB에서 개체를 나타내는 용어
- 세그먼트 : 계층 DB에서 개체를 나타내는 용어
- 속성 : 개체를 구성하는 항목

24. Section 011

오너와 멤버가 하나의 세트를 구성하는 DBMS는 네트워크 DBMS이다.

DBMS 종류

- 관계형 DBMS : DB2, Sybase, Oracle, dBASE III
- 계층형 DBMS : IMS
- 망형 DBMS : TOTAL, DBTG, EDBS

25. Section 011

실제 의미	관계 DB	계층 DB	망 DB
관계성 표현 구조	테이블	트리	그래프
개체의 관계성	내부 상관 관계성	부모-자식 관계	세트(오너-멤버 관계)
개체 집합	테이블, 릴레이션	세그먼트 타입	레코드 타입
함수관계	1:1, 1:N, N:M	1:N	1:1, 1:N, N:M
개체(레코드)	튜플	세그먼트 오커런스	레코드 오커런스
항목(필드)	속성	필드	데이터 항목
항목값	속성 값	필드 값	데이터 항목 값

28. Section 012

물리적 설계의 개념 및 특징

- 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 대해서 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 포함시켜야 할 것은 저장 레코드의 양식

설계, 레코드 집중(Record Clustering)의 분석 및 설계, 접근 경로 설계 등이다.

- 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.

29. Section 012

해석 : 데이터베이스 설계 측면에서 볼 때 같은 개념이 아닌 것은?
①, ③, ④번은 개념적 데이터 모델과 같은 의미이다.

30. Section 012

물리적 데이터베이스 설계 시 고려 사항

- 인덱스의 구조
- 레코드 크기
- 파일에 존재하는 레코드 개수
- 파일에 대한 트랜잭션의 갱신과 참조 성향
- 시스템 운용 시 파일 크기의 변화 가능성
- 데이터의 무결성, 일관성, 효율성, 보안, 회복, 데이터베이스의 확장성

31. Section 009

해석 : 관계형 데이터베이스 구조의 측면에서 데이터를 처리하는 데이터베이스 관리 시스템은 사용자의 파일에 있는 레코드들의 논리적 관계를 나타내는 일련의 2차원적 테이블(표)을 사용한다.

32. Section 012

수행될 질의를 안다는 것은 요구사항 분석 이후의 내용이므로 개념적 설계 단계에서 고민할 문제이다.

33. Section 012

데이터베이스 구현

- 구현 단계는 논리적 설계 단계와 물리적 설계 단계에서 도출된 데이터베이스 스키마를 파일로 생성하는 단계이다.
- 사용하려는 특정 DBMS의 DDL을 이용하여 데이터베이스 스키마를 기술한 후 컴파일하여 빈 데이터베이스 파일을 생성한다.
- 생성된 빈 데이터베이스 파일에 데이터를 입력한다.
- 응용 프로그램을 위한 트랜잭션을 작성한다.
- 데이터베이스 접근을 위한 응용프로그램을 작성한다.

34. Section 009, 010, 011

관계 데이터 모델, 계층 데이터 모델, 네트워크 데이터 모델의 가장 큰 차이점은 관계의 표현 방법이다.

3장 정답 및 해설 — 관계 데이터베이스 모델과 언어

1.①	2.③	3.④	4.②	5.④	6.①	7.③	8.②	9.④	10.①	11.③	12.③	13.③	14.②	15.④
16.②	17.③	18.③	19.②	20.①	21.①	22.③	23.②	24.③	25.④	26.②	27.③	28.②	29.④	30.④
31.③	32.②	33.①	34.④	35.③	36.④	37.②	38.①	39.①	40.③	41.④	42.④	43.②	44.②	45.③
46.①	47.④	48.③	49.③	50.③	51.②	52.④	53.②	54.④	55.③	56.②	57.③	58.①	59.②	60.③
61.①	62.①	63.④	64.③	65.①	66.①	67.②	68.④	69.③	70.①	71.②	72.④	73.③	74.②	75.①
76.②	77.①	78.③	79.④	80.②	81.③	82.③	83.①	84.①	85.③					

1. Section 013

튜플은 릴레이션을 구성하는 각각의 행을 말하는 것으로, 이 문제의 릴레이션에서 튜플의 수는 4이다.

2. Section 013

관계형 데이터베이스 관련 용어 중 행은 튜플(Tuple)이라고 불리며, 열은 속성(Attribute)이라고 불린다. 그리고 테이블(Table)은 릴레이션(Relation)이라고 불린다.

3. Section 014

해석 : 테이블에서 두 개 이상의 중복값을 허락하지 않는, 테이블 내의 유일한 구분자는 기본 키다.

4. Section 013

관계형 데이터베이스에서의 튜플을 레코드라고 부른다.

5. Section 015

교차곱(Cartesian Product)

- 카티션 프로덕트(교차곱)는 두 릴레이션에 있는 튜플들의 순서쌍을 구하는 연산이다.
- $R \times S = \{r \cdot s \mid r \in R \wedge s \in S\}$
  - $r = \langle a_1, a_2, \cdots, a_n \rangle$, $s = \langle b_1, b_2, \cdots, b_n \rangle$
  - r은 R에 존재하는 튜플이고, s는 S에 존재하는 튜플이다.
- 교차곱의 카디널리티는 두 릴레이션의 카디널리티를 곱한 것과 같다.
  - $|R \times S| = |R| \times |S|$

6. Section 014

①번은 외래 키에 대한 설명이다.

8. Section 014

외래 키와 참조하려는 테이블의 기본 키는 도메인과 속성 개수가 같아야 하지만 속성명이 동일할 필요는 없다.

9. Section 015

합집합은 두 릴레이션을 합친 후 공통적인 것 중 하나는 제거하고, 교집합은 두 릴레이션에서 공통적인 것만 구하고, 차집합은 두 릴레이션에서 공통적인 것을 제외한 나머지를 구하는 것이므로 모두 공통 튜플 수와 관계가 있다. 하지만 교차곱은 두 릴레이션의 모든 카디널리티를 곱하는 연산이므로 두 릴레이션의 공통 튜플 수와 관계가 없다.

10. Section 015

"연산의 인수로 주어진 릴레이션에서 어떤(확실한) 속성들을 산출하는 단항 연산이다. 릴레이션은 집합이므로 중복된 행들이 제거된다."라고 해석되는 좀 난해한 문장이다. 그러나 보기에 주어진 연산 중 릴레이션에서 속성들을 추출하는 단항 연산은 Project뿐이고, 추출된 릴레이션의 특성이 집합이기 때문에 중복된 행들이 제거되는 것도 Project의 특징이다. 중복된 행들이 제거된다는 의미는, 예를 들어 1-79쪽 〈성적〉 테이블에 이름이 중복되어 저장되어 있을 경우 이름 속성만 추출한다면 동일한 이름이 여러 개 추출되는 것은 의미가 없다. 그러므로 중복된 이름들을 제거하게 되는 것이다.

11. Section 015

일반 집합 연산 시 카디널리티

- 합집합($|R \cup S|$) : $\leq |R| + |S|$
- 교집합($|R \cap S|$) : $\leq MIN\{|R|, |S|\}$
- 차집합($|R - S|$) : $\leq |R|$
- 교차곱($|R \times S|$) : $= |R| \times |S|$

12. Section 015

두 개의 릴레이션을 이용하는 관계대수 연산으로서, ㉠과 ㉡으로 세 번째 릴레이션 ㉢을 만드는데, 이 세 번째 릴레이션 ㉢에는 ㉠ 릴레이션의 모든 행과 ㉡ 릴레이션의 모든 행이 연결된 형태로 들어 있다.

14. Section 015

조인 조건이 '='일 때 동일한 속성이 두 번 나타나게 되는데, 이 중 중복된 속성을 제거하여 같은 속성을 한 번만 표기하는 방법을 자연(Natural) 조인이라고 한다.

15. Section 015

디비전 : $R \div S$

16. Section 015

조건이 PRICE=COST이므로 PRICE와 COST 값이 동일한 튜플만 CODE, COST, PRICE 순으로 나타낸다. 따라서 ②번의 경우처럼 COST와 PRICE가 다른 것은 구해지지 않는다.

17. Section 015

- $\sigma_{\text{학과='국문과'}}(\text{학생})$: 학생 릴레이션에서 국문과 학생들을 Select한다.
- $\pi_{\text{학번,이름}}$: Select된 결과를 대상으로 이름과 학번만 새로운 테이블로 나타낸다.

18. Section 013

- 해석 : 속성은 더 이상 쪼개질 수 없는 값이라 하여 무엇이라고 불리는가?
- 속성은 더 이상 쪼개질 수 없다 하여 원자적인 속성이라 부른다.

19. Section 015

관계대수는 원하는 정보와 그 정보를 어떻게 유도하는가에 대한 연산의 순서를 기술하는 절차적인 언어인 데 비해, 관계해석은 원하는 정보에 대한 내용만 형식을 갖추어 정의한다.

21. Section 016

- 정규화란 관계형 데이터 모델에서 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.

22. Section 016

문제에 주어진 함수적 종속 관계는 기본키(A, B)에 속하지 않으면서 키에 완전 종속이 아닌 속성도 없고 또 이행적 종속도 없으므로 2정규형이면서 3정규형이다. 하지만 결정자 C가 후보키로 취급되지 않았기 때문에 BCNF는 아니다. 문제의 릴레이션 R을 다음과 같이 분리하여 BCNF 정규형으로 만들 수 있다.

- R1(A, C), 기본 키 : {A, C}, 외래 키 : C, 참조 : R2
- R2(C, B) 기본 키 : B

23. Section 016

3NF는 정규형에서 모든 이행(Transitive) 종속을 제거해야 한다.

24. Section 016

정규화 하는 것은 테이블을 결합하여 종속성을 제거하는 것이 아니고, 더 작은 테이블로 분해해 가면서 종속성을 제거하는 것이다.

25. Section 016

이행적 종속 관계

A → B이고 B → C일 때 A → C를 만족하는 관계

① B → E : B → D, D → E
② C → D : C → B, B → D
③ B → F : B → D, D → E, E → F
※ A → C는 성립되지 않는다.

26. Section 016

BCNF(Boyce-Codd 정규형)

- 릴레이션 R에서 결정자가 모두 후보 키인 관계형이다.
- 3NF에서 후보 키가 많고 서로 중첩되는 경우에 적용하는, 강한 제3정규형이라고도 한다.
- 모든 BCNF(Boyce-Codd Normal Form)가 종속성을 보존하는 것은 아니다.
- BCNF의 제약 조건
  - 키가 아닌 모든 속성은 각 키에 대하여 완전 종속되어야 한다.
  - 키가 아닌 모든 속성은 그 자신이 부분적으로 들어가 있지 않은 모든 키에 대하여 완전 종속되어야 한다.
  - 어떤 속성도 키가 아닌 속성에 대해서는 완전 종속할 수 없다.

※ BCNF는 복합 속성을 허용하며, 릴레이션의 모든 결정자가 후보키라는 개념에 기본을 두고 있다.

27. Section 016

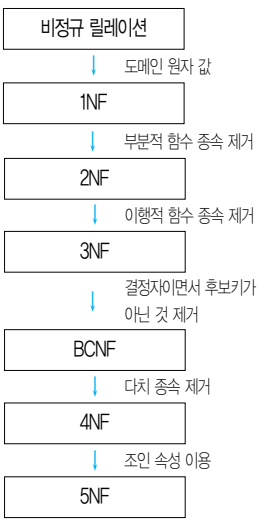
정규화

- 정규화란 관계형 데이터 모델에서 함수적 종속성 등의 종속성 이론을 이용하여 잘못 설계된 관계형 스키마를 더 작은 속성의 세트로 쪼개어 바람직한 스키마로 만들어 가는 과정이다.
- 정규형에는 제 1정규형, 제 2정규형, 제 3정규형, BCNF형, 제 4정규형, 제 5정규형이 있다.
- 정규화는 데이터베이스의 논리적 설계 단계에서 수행된다.

28. Section 016

A → B이고 B → C일 때 A → C를 만족하는 종속 관계는 이행적 종속 관계이다. 이행적 종속을 제거하는 것은 제3정규형을 만드는 것이다.

정규화 과정 정리



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

29. Section 016

어떤 애트리뷰트 Y가 다른 복합 애트리뷰트 X에 종속되고 X의 부분집합에는 종속되지 않는 경우를 완전 함수적 종속이라 하고, 부분집합에도 종속되는 경우를 부분 함수적 종속이라고 한다.

30. Section 016

기본키가 (A, B)이고 B→D이면 기본 키의 일부인 B에 의해 D가 결정되므로 부분 함수적 종속이 존재하는 것이다. 이 부분 함수적 종속을 제거해야 2NF가 되므로 현재는 1NF이다.

31. Section 017

SQL(Structured Query Language)

- 1974년 IBM 연구소에서 개발한 SEQUEL에서 유래한다.
- IBM 외에도 많은 회사에서 관계형 데이터베이스를 지원하는 언어로 채택하고 있다.
- 관계대수와 관계해석을 기초로 한 혼합 데이터 언어이다.
- 질의어이지만, 질의 기능만 있는 것이 아니라 데이터 구조의 정의, 데이터 조작, 데이터 제어 기능을 모두 갖추고 있다.

32. Section 017

SQL에서 사용하는 테이블의 종류

- 기본 테이블(Base Table) : 이름을 가지고 있으며 독자적으로 존재함
- 뷰 테이블(View Table) : 독자적으로 존재하지 못하고, 기본 테이블로부터 유도된 이름을 가진 가상 테이블
- 임시 테이블(Temporary Table) : 질의문 처리 결과로 만들어진 테이블로서, 이름을 가지지 않음

33. Section 017

질의 최적화는 질의를 실행하는 전략을 개선시키는 것이다.

34. Section 017

질의의 모호성

- SQL에서는 다른 테이블에 있는 두 개의 속성에 대해 같은 이름을 사용하는 것을 허용한다.
- 질의에서 다른 테이블에 존재하면서 같은 이름을 사용하는 두 개의 속성을 참조하면 질의가 모호해진다.
- 모호성을 제거하기 위해서 테이블의 이름을 같이 사용함으로써 속성 이름의 모호성을 제거한다.

중첩 질의의 모호성

- 외부 질의에 사용되는 테이블과 내부 질의에 사용되는 테이블이 같은 속성을 포함하고 있을 때 질의가 모호해지는 경우가 존재한다.
- 별명 달기 방법을 이용하여 해결한다.

※ 두 개의 속성이 같은 데이터형을 참조하는 경우에는 질의의 모호성이 발생하지 않는다.

35. Section 018

스키마 정의문 형식

```
CREATE SCHEMA 스키마_이름 AUTHORIZATION 사용자_id;
```

36. Section 018

도메인(사용자 정의 Data_Type) 정의문

```
CREATE DOMAIN 도메인_이름 data_type
[DEFAULT 묵시값_정의]
[CONSTRAINT VALID-도메인_이름
CHECK (범위값)];
```

37. Section 018

데이터 형(Data Type)

- DECIMAL(m,n) : 10진 소수
- INTEGER : 4Byte 정수
- SMALLINT : 2Byte 정수
- FLOAT : 부동 소수점 수
- CHAR(n) : 문자의 수가 n인 스트링
- VARCHAR(n) : 문자의 수가 최대 n인 스트링

38. Section 018

도메인(사용자 정의 Data_Type) 정의문

```
CREATE DOMAIN 도메인_이름 data_type
    [DEFAULT 묵시값_정의]
    [CONSTRAINT VALID-도메인_이름
    CHECK (범위값)]
```

39. Section 018

※ 일반적으로 테이블 생성 명령어에 스키마를 생성하지 않으면 현재 명령어가 실행되는 환경에 있는 스키마에 명령이 적용된다.

CREATE TABLE 명령어

- 새로운 테이블을 생성하기 위해서 사용한다.
- 테이블의 이름, 속성, 제약을 명시한다.
- 명령을 적용할 스키마를 명시할 수 있다.
- 명령을 적용할 스키마를 명시하는 경우에는 스키마와 생성할 테이블 이름을 '.' 문자로 구분한다.

40. Section 018

```
CREATE TABLE 기본테이블_이름
        (속성명  data_type  [Not Null], …,
        PRIMARY KEY (기본 키_속성명),
        UNIQUE (대체 키 속성명, … ),
        FOREIGN KEY (외래 키_속성명, … )
            REFERENCES 참조테이블(기본 키_속성명)
        CHECK (조건식) );
```

- 속성명 : 기본 테이블에 포함될 모든 속성에 대하여 속성명과 그 속성의 data_type, Not NULL 여부를 지정함
- PRIMARY KEY : 기본 키 속성을 지정함
- UNIQUE : 대체 키로 사용할 속성명들을 지정함
- FOREIGN KEY~REFERENCES~
  - 참조할 다른 테이블과 그 테이블을 참조할 때 사용할 외래 키 속성을 지정한다.
  - 외래 키가 지정되면 참조 무결성의 CASCADE 법칙이 적용된다.
- CHECK : 제약 조건의 정의

※ 기본 키나 대체 키에 대한 인덱스를 생성할 때는 UNIQUE 옵션을 사용한다.

41. Section 016

정규화는 중복과 종속성을 제거하기 위해 릴레이션을 분해하는 것이므로, 바람직하지 못한 릴레이션을 어떻게 합쳐야 하는지에 관한 것이 아니라 어떻게 분할할 것인지에 대한 구체적인 판단 기준을 제공한다.

42. Section 018

기본 data_type에는 SEX가 없기 때문에 CREATE DOMAIN으로 사용자가 정의한 data_type이라고 보아야 한다.

43. Section 018

```
CREATE [UNIQUE] INDEX 인덱스_이름
        ON  기본테이블_이름({속성_이름  [ASC  |
        DESC],})
        [CLUSTER];
```

- 정렬 여부 지정
  - ASC : 오름차순 정렬
  - DESC : 내림차순 정렬
- CLUSTER 옵션 : 동일 인덱스 값을 갖는 튜플들을 그룹으로 묶을 때 사용

44. Section 018

ALTER TABLE 일반 형식

```
ALTER TABLE 기본테이블_이름 ADD 속성_이름 data-
    type [DEFAULT '기본값'];
ALTER TABLE 기본테이블_이름 ALTER 속성_이름 [SET
    DEFAULT '기본값'];
ALTER TABLE 기본테이블_이름 DROP 속성_이름
    [CASCADE];
```

- ADD : 새로운 속성(열)을 추가할 때 사용
- ALTER : 특정 속성의 Default 값을 변경할 때 사용
- DROP : 특정 속성을 삭제할 때 사용

※ 열은 속성을 말한다.

45. Section 018

DROP 명령어

- SCHEMA, DOMAIN, TABLE, VIEW를 제거하는 데 사용된다.
- CASCADE와 RESTRICT의 두 개의 옵션이 존재한다.
- CASCADE 옵션을 사용하면 제거될 테이블을 참조하는 모든 제약과 뷰도 자동으로 스키마로부터 삭제된다.
- RESTRICT 옵션이 사용되면 테이블이 제약이나 뷰로부터 참조되지 않는 경우에만 삭제된다.

46. Section 019

SELECT문의 일반 형식

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

※ SELECT절은 질의 결과에 포함될 데이터 열(속성)들을 기술하며, 이는 데이터베이스로부터 데이터 열 또는 계산 열이 될 수 있다.

47. Section 019

UNION은 중복을 제거하여 두 개의 테이블을 통합하는 명령으로, 두 개의 테이블에서 A 속성을 합치면 1, 2, 2, 3, 3, 4이지만 여기서 중복을 제거하면 1, 2, 3, 4가 된다.

48. Section 019

HAVING절은 GROUP BY와 함께 사용되는 것으로, 그룹에 대한 조건을 지정한다.

49. Section 019

WHERE 조건에 지정된 하위 질의의 SELECT문을 먼저 검색한 후 검색 결과를 본 질의의 조건에 있는 사원번호 속성과 비교한다.

❶ 〈인사〉 테이블에서 성명 속성의 값이 "오영우"와 같은 튜플의 '사원번호' 속성의 값을 검색한다. 결과는 43이다.

❷ 〈차량〉 테이블에서 '사원번호' 속성의 값이 43과 같은 튜플의 '종류' 속성의 값을 검색한다. 결과는 "C"이다.

50. Section 019

정렬을 나타내는 "ORDER BY 지원학과, 점수 DESC"를 통해 지원학과 순으로 오름차순 정렬되고, 지원학과가 같은 경우는 점수를 기준으로 내림차순 정렬됨을 알수 있다. 또한 이 모든 정렬은 WHERE 점수 〉 59에 의해 점수가 60점 이상인 지원자만을 대상으로 함을 알 수 있다.

51. Section 020

INSERT의 일반형식

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

※ 기본 테이블의 모든 속성을 사용하여 데이터를 삽입할 경우에는 속성명을 생략할 수 있다.

52. Section 019

LIKE : (속성 LIKE "부분 문자%") 형식을 사용하여 지정된 속성에서 부분 문자가 들어 있는 튜플을 대상으로 검색시킬 조건을 표현할 때 사용한다.

53. Section 019

LIKE는 문자열의 패턴을 비교할 때 사용하는 연산자이고, '%'는 모든 문자를 의미하는 와일드카드 문자이므로 'WHERE SNAME LIKE '홍%'은 'SNAME' 속성의 값이 "홍"으로 시작하는 모든 튜플을 의미한다.

54. Section 019

SQL의 내장 함수

- COUNT(속성명) : 튜플 수를 구하는 함수
- MAX(속성명) : 최대값을 구하는 함수
- MIN(속성명) : 최소값을 구하는 함수
- SUM(속성명) : 합계를 구하는 함수
- AVG(속성명) : 평균을 구하는 함수

55. Section 019

그룹화 속성을 명시하기 위해서는 GROUP BY절을 사용해야 한다.

56. Section 019

정렬

- 질의를 수행해서 얻은 튜플을 사용자가 정렬할 수 있도록 한다.
- ORDER BY절을 사용한다.
- 기본 정렬 방식은 오름차순이다.
- ASC, DESC 키워드를 사용하여 사용자가 각각 오름차순 정렬, 내림차순 정렬 방식을 지정할 수 있다.

57. Section 019

```
SELECT 〈속성 리스트〉
FROM 〈테이블 리스트〉 WHERE 〈조건〉
```

- 〈속성 리스트〉 : 직원의 '이름(name)'과 그들이 사는 '도시(city)'를 찾으라고 하였으므로 속성 name과 city를 기술한다. 단, 〈테이블 리스트〉에서 첫 번째로 기술한 테이블에 대한 속성은 소속 표시를 하지 않아도 된다.
- 〈테이블 리스트〉 : 직원들에 대한 테이블 works와 그들이 사는 위치에 대한 테이블 lives를 기술한다.
- 〈조건〉 : company가 "제일은행"인 튜플을 찾고, 찾은 튜플의 직원(name)이 사는 도시를 찾도록 조건을 지정한다. 즉 works 테이블에서 company가 "제일은행"이고 그 직원의 이름(name)과 일치하는 이름을 가진 튜플을 lives에서 찾도록 표현한다. 따라서 works.company = '제일은행' and works.name = lives.name으로 기술한다.

58. Section 020

명령 하나로 한 개의 테이블에만 삽입시킬 수 있다.

59. Section 020

※ 특별히 선별할 레코드가 없다면 조건을 안 써도 되지만 쓰려면 반드시 WHERE 명령 뒤에 써야 한다.

DELETE 명령어

- 삭제될 튜플에 대한 조건을 WHERE절에 기술한다.
- 모든 레코드를 삭제할 때는 WHERE절을 생략한다.
- 모든 레코드를 삭제하더라도 테이블 구조는 남아 있기 때문에 디스크에서 테이블을 완전히 제거하는 DROP과는 다르다.
- 참조 무결성 제약이 존재하면 다른 테이블에 존재하는 튜플도 삭제될 수 있다.
- 한 개의 명령은 하나의 테이블만 대상으로 삭제시킬 수 있다.
- 조건절에 부속 질의어를 사용할 수 있다.

60. Section 020

※ 변경할 속성과 그들의 새로운 값을 명시하기 위해서는 SET절이 사용된다.

UPDATE문의 일반 형식

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

61. Section 020

UPDATE 명령은 WHERE에서 정의된 조건에 따라 SET에서 정의된 식으로 갱신한다.

62. Section 020

한 개의 Delete문에는 한 개의 테이블명만 사용할 수 있다.

63. Section 021

내장 SQL문의 호스트 변수의 데이터 타입은 이에 대응하는 데이터베이스 필드의 SQL 데이터 타입과 일치하여야 한다.

64. Section 021

내장 SQL 문장 끝은 사용하는 호스트 언어에 따라 문장의 끝을 처리하는 방법이 다르다. C 언어와 같이 문장이 ;(세미콜론)으로 끝나는 호스트 언어는 내장 SQL도 ;(세미콜론)으로 끝내고, Visual Basic과 같이 특별한 기호 없이 끝나는 언어는 내장 SQL도 기호 없이 종료한다.

66. Section 021

커서(Cursor)

- 커서(Cursor)는 내장 SQL문의 수행 결과로 반환될 수 있는 복수의 튜플들을 액세스할 수 있도록 해주는 개념이다.
- 질의 수행 결과로 반환되는 첫 번째 튜플에 대한 포인터로 생각할 수 있다.
- 커서를 사용하여 질의 결과로 반환될 수 있는 튜플들을 한 번에 하나씩 차례로 처리할 수 있다.

67. Section 021

FETCH : 질의 결과의 튜플들 중 현재의 다음 튜플로 커서를 이동시키는 명령

69. Section 022

뷰는 일반 테이블과 마찬가지로 CREATE로 정의한다.

71. Section 023

시스템 카탈로그(System Catalog)의 의미

- 시스템 카탈로그는 시스템 그 자체에 관련이 있는 다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스이다.
- 시스템 카탈로그는 데이터베이스에 포함되는 모든 데이터 객체에 대한 정의나 명세에 관한 정보를 유지 관리하는 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(MetaData)라고 한다.

72. Section 023

시스템 카탈로그는 데이터 정의어의 결과로 구성되는 기본 테이블, 뷰, 인덱스, 패키지, 접근 권한 등의 데이터베이스 구조 및 통계 정보를 저장한다.

※ 개체는 데이터베이스에 저장되는 일반적인 데이터로서 일반 테이블에 저장된다.

73. Section 023

Data Directory = 사전 관리기

- 데이터 사전에 수록된 데이터를 실제로 접근하는 데 필요한 정

보를 관리 유지하는 시스템이다.

- 시스템 카탈로그는 사용자와 시스템 모두 접근할 수 있지만 데이터 디렉터리는 시스템만 접근할 수 있다.

74. Section 023

해석 : 많은 기관들은 현재 데이터베이스 시스템용으로 ( )를 관리하기 위해 미니 DBMS인 데이터 사전 시스템을 사용하고 있다. 이것은 데이터베이스의 구조, 제약, 응용, 사용 권한 등을 기술하는 데이터이다.

75. Section 023

스키마를 정의하는 언어는 DDL이며, DDL로 정의한 스키마는 데이터 사전에 수록된다.

76. Section 016

Anomaly(이상)의 개념 및 종류

- 정규화(Normalization)를 거치지 않으면 데이터베이스 내에 데이터들이 불필요하게 중복되어 릴레이션 조작 시 예기치 못한 곤란한 현상이 발생하는데 이를 이상(Anomaly)이라 하며, 이상에는 삽입 이상, 삭제 이상, 갱신 이상이 있다.
- 삽입 이상(Insertion Anomaly) : 릴레이션에 데이터를 삽입할 때 의도와는 상관없이 원하지 않은 값들도 함께 삽입되는 현상
- 삭제 이상(Deletion Anomaly) : 릴레이션에서 한 튜플을 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는, 연쇄 삭제 현상이 일어나는 현상
- 갱신 이상(Update Anomaly) : 릴레이션에서 튜플에 있는 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상

77. Section 023

시스템 카탈로그는 시스템 그 자체에 관련이 있는 다양한 객체들에 관한 정보를 저장하는 시스템 데이터베이스로 물리적으로 존재한다.

78. Section 023

질의 최적화기

사용자의 요구를 효율적인 형태로 변환하고 질의를 처리하는 좋은 전략을 모색한다.

79. Section 023

시스템 카탈로그는 DBMS가 스스로 생성하고, 유지하는 데이터베이스 내의 특별한 테이블의 집합체이다.

80. Section 023

데이터 사전은 테이블(시스템 테이블)로 구성되어 있어 일반 사용자도 SQL을 이용하여 내용을 검색해 볼 수 있다. 하지만 데이터 사전의 내용을 변경할 수는 없다.

81. Section 014

외래키를 포함하는 릴레이션이 참조하는 릴레이션이 되고, 대응되는 기본 키를 포함하는 릴레이션이 참조되는 릴레이션이 된다.

83. Section 014

릴레이션의 기본 키와 대응되어 릴레이션 간의 참조 무결성 제약조건을 표현하는데 사용되는 중요한 도구는 외래 키이다.

84. Section 014

기본 키는 한 릴레이션에서 특정 튜플을 유일하게 구별할 수 있는 속성으로, 기본 키로 지정된 속성에는 동일한 값이 존재할 수 없다.

85. Section 023

해석 : 기본적인 데이터베이스의 구조를 기술하는데 사용하는 용어로, 카탈로그에 저장된 정보를 메타 데이터라고 한다.

4장 정답 및 해설 — 자료 구조의 기본

1.②	2.④	3.①	4.④	5.③	6.①	7.④	8.②	9.①	10.①	11.①	12.④	13.③	14.②	15.③
16.④	17.④	18.④	19.④	20.①	21.①	22.②	23.③	24.③	25.④	26.④	27.④	28.③	29.①	30.③
31.③	32.③	33.①	34.①	35.④	36.①	37.④	38.③	39.④	40.④	41.③	42.③	43.③	44.①	45.③
46.③	47.③	48.①	49.④	50.③	51.②	52.①	53.①	54.④	55.④	56.④				

1. Section 025

배열은 선형 구조로서 연속적으로 저장되므로 Sequential Memory Allocation이 가장 적합하다.

2. Section 025

배열의 크기는 각 차원 크기의 곱이다.

- 행의 개수 = 2 − (−3) + 1 = 6
- 열의 개수 = 5 − 2 + 1 = 4
- 배열의 크기 = 행의 개수 × 열의 개수 = 6 × 4 = 24

※ −3:2 = −3, −2, −1, 0, 1, 2

※ 2:5 = 2, 3, 4, 5

3. Section 025

환상형 링크드 리스트는 맨 마지막의 포인터가 처음의 노드를 지시하는 것으로, 널(Nil) 포인터가 존재하지 않는다.

4. Section 025

연결 리스트(Linked List)

- 물리적인 순서에 의해서가 아니라 포인터를 사용하여 논리적인 순서에 따라 저장한다.
- 장점 : 중간 노드의 삽입, 삭제 시 효율적이고, 기억공간이 독립적임
- 단점 : 액세스 시간이 느리고, 포인터가 차지하는 만큼의 기억공간이 추가로 필요함

5. Section 025

연결 리스트에서의 노드 구성 : | 데이터 | 링크 |

6. Section 025

연속 배열에서의 삽입은 $\frac{n+1}{2}$ 이고, 삭제 시에는 $\frac{n-1}{2}$ 이다.

7. Section 026

Stack의 응용 분야

- 부 프로그램 호출 시 복귀주소를 저장할 때
- 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

※ 스풀(Spool)은 먼저 작업이 할당된 자료를 먼저 처리하는 것이므로 큐(Queue)를 사용한다.

8. Section 026

자료의 삽입(Push)

```
TOP=TOP + 1          ← 스택 포인터(TOP)를 1 증가시키다.
IF TOP >M THEN       ← 스택 포인터가 스택의 크기보다 크
   OVERFLOW            면 OVERFLOW
ELSE                 ← 그렇지 않으면 ITEM이 가지고 있는
   X(TOP) ← ITEM       값을 스택의 TOP 위치에 삽입한다.
```

- M : 스택의 크기
- TOP : 스택 포인터
- X : 스택의 이름
- OVERFLOW : 스택으로 할당받은 메모리 부분의 마지막 주소가 M번이라 할 때, Top Pointer에 M 주소가 저장되어 있다면 스택의 모든 기억장소가 꽉 채워져 있는 상태이므로 더이상의 자료를 삽입할 수 없어 Overflow를 발생시킴

※ ① if Top ≥ n then underflow, end를 ① if Top ≥ n then overflow, end로 고쳐야 한다.

9. Section 026

Stack Pop(제거) 연산의 순서

- 스택이 빈 상태인지를 검사하여 참이면 Underflow가 발생한 것이므로 종료한다.

- 스택 포인터가 가리키는 원소를 제거한다.
- 스택 포인터를 하나 감소한다.
- 연산 수행 결과 스택이 빈 상태인지를 검사하여 참이면 스택 포인터를 Nil로 설정한다.

10. Section 027

중위 표기식의 후위 표기 변환, 함수 호출과 리턴, 이진 트리의 중위 순회 등은 스택의 응용 분야이다.

11. Section 027

- Stack : 가장 나중에 입력된 데이터가 가장 먼저 출력되는 LIFO(Last In First Out) 구조
- Deque
  - 스택과 큐의 복합 형태로, 선형 구조 중 가장 일반적인 구조
  - 입 · 출력이 양쪽에서 가능
  - 스크롤(Scroll) : 입력 제한 데크
  - 셸프(Shelf) : 출력 제한 데크

12. Section 028

형제 노드(Sibling)는 트리에서 같은 부모(Parent) 노드를 갖는 노드들을 의미한다.

13. Section 028

- 트리의 차수 : 전체 디그리 중에서 최대 디그리를 트리의 차수로 함
- 터미널 노드(Terminal Node) : 디그리가 0인 노드로, ⓓ ⓔ ⓘ ⓙ ⓚ ⓗ를 의미
- 간 노드(Non-terminal Node) : 디그리가 0이 아닌 노드

14. Section 028

트리(Tree)에서 임의의 노드 N에 연결된 다음 레벨 (Level)의 노드를 Children node라고 한다.

- 부모 노드(Parent Node) : 어떤 노드에 연결된 이전 레벨의 노드들
- 형제 노드(Brother Node, Sibling) : 동일한 부모를 갖는 노드들
- 단말 노드(Terminal Node) = 잎 노드(Leaf Node) : 자식이 하나도 없는 노드, 즉 Degree가 0인 노드

15. Section 029

레벨 i란 전체 레벨 수에서 특정 레벨을 의미한다.

레벨(Level)과 노드 수와의 관계 : i 수준에 존재할 수 있는 노드의 최대 개수는 2^{i-1}

16. Section 029

- 전이진 트리(Complete Binary Tree) : 순서대로 들어온 트리로, 정이진 트리가 되기 직전의 트리
- 정이진 트리(Full Binary Tree) : 깊이가 k인 트리에서 전체 노드의 개수가 2^k-1을 만족하는 이진 트리
- 사향 이진 트리(Skewed Binary Tree) : 왼쪽이나 오른쪽으로 치우쳐진 트리

17. Section 036

차수가 m인 B 트리의 특성

- 루트 노드는 적어도 두 개의 자식 노드를 가져야 한다.
- 루트 노드와 리프 노드를 제외한 모든 노드는 적어도 $\frac{m}{2}$의 자식 노드를 가져야 한다.
- 모든 리프 노드의 수준(Level)이 같아야 한다.

18. Section 030

스레드(Thread) 2진 트리

- 닐 포인터(Nil Pointer)를 트리의 운행에 이용하는 것이다.
- Ⓔ의 왼쪽 링크는 프리오더로 운행할 시 정상적인 방문 순서에 따라 H를 지정한다.
- Ⓔ의 오른쪽 링크는 닐 포인터이므로 프리오더로 운행한 다음, 바로 다음 노드를 지칭하도록 포인터를 설정한다.
- Preorder로 운행한 경우 : ABDEHCFGIJ
  ∴ Ⓔ의 왼쪽 포인터 : Ⓗ, 오른쪽 포인터 : Ⓗ

20. Section 030

AVL 트리는 단노드들의 레벨 차이가 1 이하인 트리로서 탐색 시간이 빠르다.

21. Section 030

연산자의 우선순위에 맞게 괄호로 묶은 후 연산자를 해당 연산의 괄호 뒤(오른쪽)로 옮긴다.

22. Section 030

후위 순서로 운행하면 D, E, C, B, A이다.

23. Section 030

스택에 Prefix로 저장된 연산식은 피연산자, 피연산자, 연산자 순서로 꺼냈을 때 연산이 된다. 그러므로 6, 5를 꺼낸 후 2, 4, /를 꺼내서 4/2를 연산한 후 결과 2를 넣고 다시 5, 6을 넣는다.

=	X	−	5	+	3	*	+	/	4	2	5	6

↓

=	X	−	5	+	3	*	+	2	5	6		

6, 5, 2, +를 꺼내서 2+5의 결과와 6을 다시 넣는다.

=	X	−	5	+	3	*	7	6				

↓

=	X	−	5	+	3	42						

=	X	−	5	45								

↓

=	X	-40										

∴ X = −40

24. Section 030

Tree 구조로 나타낸 산술식을 원래의 수식으로 표현하려면 인오더로 운행한다.

- 인오더 운행 결과 : A * B − C + D / E
- 부 노드에 해당하는 연산자와 자 노드에 해당하는 피연산자를 괄호로 묶으면 우선순위를 알수 있다.
  ((A * (B − C)) + (D / E))

25. Section 029

우측 사향 이진 트리는 노드가 우측으로 치우친 것으로 프리오더, 인오더로의 운행 결과가 같고, 좌측 사향 이진 트리는 인오더와 포스트오더의 운행 결과가 같다.

26. Section 031

완전 그래프의 조건

- 비방향성 그래프 : 전체 간선의 수가 $\frac{n(n-1)}{2}$개
- 방향성 그래프 : 전체 간선의 수가 n(n−1)개

27. Section 032

해석 : ( )는 보통 순서가 없는 아이템들 또는 레코드들을 각 레코드의 내용을 근거로 한 특정 조건에 따라서 순서적으로 배열하는 것이다.

28. Section 033

기수 정렬(Radix Sort)

- 정렬할 데이터의 키워드를 사용하여 정렬을 수행하는 방식이다.
- 키워드는 해당 데이터를 위치시킬 주소를 결정하는 데 사용한다.

29. Section 032

- 외부 정렬 : 정렬할 데이터가 많아 모두 주기억장치에 올려질 수 없는 경우에 유리하며, 보조기억장치에 있는 데이터 중 일부만을 주기억장치에 올려가면서 전체 데이터를 정렬하는 방식
- 내부 정렬 : 정렬할 모든 데이터를 주기억장치(Main Memory)에 올려 놓고 정렬하는 방식으로, 정렬할 데이터가 적은 경우에 유리함

30. Section 033

- 셀렉션 정렬은 앞에서부터 정렬되기 시작한다. 즉 1단계를 마치고 나면 가장 작은 값이 맨 앞으로 오며, 2단계를 마치면 두 번째로 작은 값이 두 번째로 온다.
- 버블 정렬은 뒤에서부터 정렬되기 시작한다. 즉 1단계를 마치고 나면 가장 큰 값이 맨 뒤로 가고 2단계를 마치면 두 번째로 큰 값이 뒤에서 두 번째에 위치한다.
- 인서션 정렬은 1단계를 마치면 첫 번째와 두 번째 값만 비교하여 교환한다.

31. Section 033

Quick Sort

- 퀵 정렬은 레코드의 많은 자료 이동을 없애고 하나의 파일을 부분적으로 나누어 가면서 정렬하는 방법으로 키를 기준으로 작은 값은 왼쪽에, 큰 값은 오른쪽 서브파일로 분해시키는 방식으로 정렬한다.
- 정렬 방식 중에서 가장 빠른 방식이며, 프로그램에서 되부름을 이용하기 때문에 스택(Stack)이 필요하다.

32. Section 033

Control Key를 중심으로 작은 값은 왼쪽에, 큰 값은 오른쪽에 놓으면서 정렬하는 방식은 퀵 정렬이다.

33. Section 034

이분 검색은 데이터가 정렬되어 있음을 전제로 하며, 데이터가 많을수록 효과적이다.

34. Section 034

보간 검색(Interpolation Search)

- 보간 검색은 찾으려는 레코드가 있음직한 부분의 키를 택하여 검색하는 방식이다.
- 선정 레코드 번호 = $\frac{\text{찾으려는 키 값(추측)} - \text{최소키 값}}{\text{최대키 값} - \text{최소키 값}} \times$ 레코드 수
- 찾으려는 레코드 근처에서부터 찾아가기 때문에 검색시간이 빠르지만, 예측을 해야 하므로 실제로는 프로그래밍이 불가능하다.

35. Section 034

제어 검색의 종류 : Interpolation 검색, Fibonacci 검색, 이분 검색

36. Section 034

이진 검색 과정

- **첫 번째 검색** : 시작 위치 = 1, 끝 위치 = 11

  $\frac{(\text{시작 위치} + \text{끝 위치})}{2} = \frac{(1+11)}{2}$ = 여섯 번째 위치의 값이 찾는 값인지 비교
- 여섯 번째 데이터 38은 21보다 크다. 따라서 끝 위치를 6-1=5로 해서 2번째 검색

  $\frac{(\text{시작 위치} + \text{끝 위치})}{2} = \frac{(1+5)}{2}$ = 세 번째 위치의 값이 찾는 값인지 비교

→ 세 번째 위치의 값이 찾는 데이터이므로 두 번째 수행에서 검색 완료

37. Section 034

검색 방법

- **키(Key)에 의한 검색** : 선형 검색, 제어 검색(이분 검색, 피보나치 검색, 보간 검색), 블록 검색, 이진 트리 검색 등
- **계수적 성질에 의한 검색** : 해싱(Hashing)

38. Section 035

해싱 기법

- 해싱은 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(직접 접근) 파일을 구성할 때 해싱이 사용되며, 접근 속도는 빠르나 기억공간이 많이 요구된다.
- 검색 속도가 가장 빠르다.
- 삽입, 삭제 작업의 빈도가 많을 때 유리한 방식이다.
- **버킷(Bucket)** : 하나의 주소를 갖는 파일의 한 구역을 의미하며, 버킷의 크기는 같은 주소에 포함될 수 있는 레코드 수를 의미함
- **슬롯(Slot)** : 한 개의 레코드를 저장할 수 있는 공간으로 n개의 슬롯이 모여 하나의 버킷을 형성함
- **Collision(충돌 현상)** : 서로 다른 두 개 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 때 Collision은 발생해도 Overflow는 발생하지 않을 수 있음)

39. Section 035

- **선형 검색** : 식별자가 이미 꽉 찬 버킷에 삽입되려고 하는 경우 새로 삽입될 식별자는 빈 슬롯을 가지고 있는 가장 가까운 버킷에 삽입함
- **이중 해싱** : 충돌이 발생하면 다른 해싱 함수를 적용함
- **체이닝** : 식별자가 삽입되는 버킷에 이미 다른 식별자들의 리스트가 존재하면 새로 삽입되는 식별자는 리스트의 마지막 위치에 삽입함
- **직접 파일에서의 충돌(Collision) 해결 방법** : 선형 방법(=Linear Method=Open Address), Random Method(무작위 처리), Quadratic Method, Quadratic Quotient Method, Chain Method, Double Hashing

40. Section 035

충돌(Collision)이 발생할 때 다음 버킷이 있으면 저장하고, 없으면 오버플로에 저장한다.

41. Section 035

오버플로 처리(Overflow Handling)

- Linear Open Addressing : 식별자가 이미 꽉 찬 버킷에 삽입되려고 하는 경우 새로 삽입될 식별자는 빈 슬롯을 가지고 있는 가장 가까운 버킷에 삽입함

- Chaining : 해싱 테이블의 각 버킷이 유의어의 리스트를 유지할 수 있도록 함으로써 Linear Open Addressing 방식에서 필요로 하는 비교 연산의 대부분을 제거하고 식별자가 삽입되는 버킷에 이미 다른 식별자들의 리스트가 존재하면 새로 삽입되는 식별자는 리스트의 마지막 위치에 삽입함

42. Section 035

해싱(Hashing) : 검색할 때 계수적 성질을 이용, 계산표(Hashing Table)에 의하여 주소를 결정하여 기억공간에 레코드를 보관하거나 검색하는 방법(해싱 함수 : 주소를 계산하는 수식)

해싱 함수의 종류

- 제산법(Division) : 키를 어떤 정수(주로 소수 이용)로 나눈 나머지 값을 주소로 이용
- 제곱법(Mid-Square) : 키를 제곱하여 얻은 값의 중간 부분값으로 주소를 결정
- 접는 방법(Folding) : 키를 여러 부분으로 나누어 각 부분을 더하거나 XOR(배타적 논리합)을 이용하여 주소를 얻는 방법
- 기수(Radix) 변환 : 다른 진법의 변환으로 주소 계산
- 계수 분석법(Digit-Analysis) : 키 각각의 자릿수를 고려해서 몇 개의 자릿수를 선정하여 주소로 결정

43. Section 035

- 폴딩(Folding)법 : 레코드 키값(K)을 여러 부분으로 나눈 후 각 부분의 값을 더하거나 XOR(배타적 논리합)한 값을 홈 주소로 삼는 방식
- 제곱(Mid-Square) 법 : 레코드 키값(K)을 제곱한 후 그 중간 부분의 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

44. Section 035

역파일(Inverted File)

- 인덱스와 함께 키가 같이 저장되어 있는 구조이다.
- 인덱스의 길이가 일정하지 않으며 인덱스만 사용해서 레코드에 접근 가능하다.

다중 리스트 파일(Multi-List File)

- 인덱스에는 키값을 갖는 첫 번째 레코드에 대한 포인터만 저장되어 있고, 나머지 레코드는 리스트의 포인터를 따라 접근해야 하는 파일이다.
- 인덱스의 길이가 고정되어 있으며, 인덱스만 가지고는 모든 레코드를 액세스할 수 없다.

45. Section 033

해석 : 병합이란 결과적으로 생성된 파일이 두 개의 개별적인 파일과 같은 구성을 가지고 있도록 두 파일을 결합하는 것이다. 예를 들어, 이름이 알파벳순으로 된 두 파일을 병합한다면 모든 이름이 알파벳 순서로 된 하나의 큰 파일이 만들어진다.

46. Section 037

직접 접근 파일(Directed Access Method)의 특성

- 디스크의 임의의 위치에 레코드를 저장하는 방법이다.
- 데이터의 입 · 출력이 자주 일어나는 응용에 적합하다.
- 응답시간에 제한이 있을 때 적합하다.

47. Section 037

멀티리스트 파일

- 장점 : 인덱스의 길이가 고정되어 있으며 수정, 삭제, 검색 작업을 효율적으로 처리함
- 단점 : 인덱스만 가지고는 모든 레코드를 액세스할 수 없음

48. Section 037

ISAM 파일의 인덱스 영역

- 마스터 색인(Master Index) : 실린더의 색인이 길 경우 처리가 용이하도록 인덱스가 구성됨
- 실린더 색인(Cylinder Index) : 트랙 색인이 길 경우 처리가 용이하도록 인덱스가 구성됨
- 트랙 색인(Track Index) : 기본 영역에 존재하는 각 트랙에서 마지막 레코드의 색인값으로 구성됨

49. Section 037

- 색인 순차(ISAM) 파일의 실린더 색인이 너무 길 때 이를 효율적으로 하기 위해 마스터 인덱스를 설정한다.
- 트랙 인덱스는 여러 개가 존재하지만 실린더 인덱스와 마스터 인덱스는 한 개만 존재한다.

50. Section 031

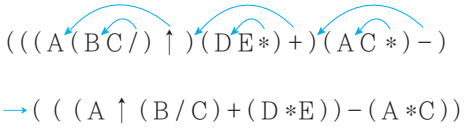
그래프에서 간선의 수는 노드를 연결하고 있는 선의 개수의 합이다.

51. Section 030

후위 표기 방식으로 표현된 수식은 피연산자 사이에 있어야 할 연산자가 연산해야 할 피연산자 뒤(오른쪽)로 가는 것을 말한다.

- 중위 표기 : A+B
- 전위 표기 : +AB
- 후위 표기 : AB+

즉 후위 식으로 표기된 식은 피연산자, 피연산자, 연산자 순서가 되어야 연산을 한다. 그러니까 A, B 다음에 연산자가 있으면 연산을 할 텐데, 연산자가 없고 피연산자인 C가 있으므로 A는 다음에 연산하고 먼저 BC/를 연산한다. BC/를 중위 식으로 표기하면 B/C이다. 후위 표기된 수식을 우리에게 익숙한 중위 표기로 변경하려면 연산자를 해당 피연산자의 사이로 이동시키면 된다.
중위 표기식으로 된 수식을 보면 어떤 연산이 가장 먼저 일어나는지 쉽게 알 수 있다.



52. Section 030

중위(Infix) 표기법은 연산자가 피연산자 사이에 있는 것으로 문제의 보기가 중위 표기법으로 표기된 것이다.

- 전위(Prefix) 표기법 : 연산자를 해당 피연산자들의 앞으로 이동시킨다.

- 후위(Postfix) 표기법 : 연산자를 해당 피연산자들의 뒤로 이동시킨다.

53. Section 037

네 가지 모두 응용에 적합한 파일 조직을 선택하는 데 영향을 주는 요인들이다.

54. Section 035

해싱 함수 선택 시 고려할 사항 중 하나는 오버플로의 최소화이다.

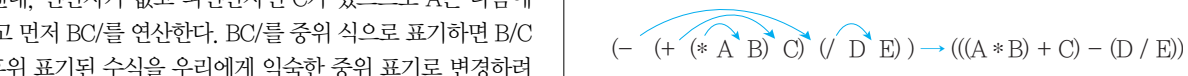
55. Section 030

Prefix(전위) 표기란 연산자가 해당 피연산자 2개의 앞(왼쪽)에 표기되어 있는 것을 말한다. 그러므로 인접한 피연자 2개와 왼쪽으로 인접한 연산자를 묶은 후 연산자를 피연산자 사이에 옮겨놓으면 된다.

❶ 피연산자 2개와 왼쪽으로 인접한 연산자 1개를 묶는다.

(− (+ (∗ A B) C) (/ D E))

❷ 연산자를 피연산자 사이로 이동시킨다.



❸ 불필요한 괄호를 제거한다.

(((A∗B)+C)−(D/E)) → A∗B+C−D/E

56. Section 037

순차 파일은 레코드가 논리적인 처리 순서에 따라 연속된 물리적 저장 공간에 차례대로 기록되어 있기 때문에 레코드의 삽입, 삭제 시 파일을 재구성해야 하므로 파일 전체를 복사해야 한다.

순차 파일(Sequential File) = 순서 파일

- 순차 파일은 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 접근만 가능한 자기 테이프에서 사용된다.
- 순차 파일의 장 · 단점

장점	단점
• 파일의 구성이 용이하고, 순차적으로 읽을 수 있으므로 기억 공간의 이용 효율이 높음 • 레코드만 저장하고 부가적인 정보는 저장하지 않으므로 기억 공간의 낭비를 방지할 수 있음 • 물리적으로 연속된 공간에 저장되므로 접근 속도가 빠름 • 어떠한 기억 매체에서도 실현 가능함	• 파일에 새로운 레코드를 삽입하거나 삭제하는 경우 파일 전체를 복사한 후 수행해야 하므로 시간이 많이 걸림 • 파일의 특정 레코드를 검색하려면 순차적으로 모든 파일을 비교하면서 검색해야 하므로 검색 효율이 낮고, 접근 시간 및 응답시간이 느림

2과목 전자계산기 구조

1장 정답 및 해설 — 논리회로

1.①	2.④	3.③	4.①	5.②	6.③	7.③	8.①	9.④	10.①	11.①	12.④	13.②	14.③	15.③
16.②	17.④	18.②	19.④	20.③	21.③	22.③	23.②	24.①	25.①	26.③	27.③	28.②	29.④	30.④
31.①	32.③	33.③	34.④	35.④	36.②	37.④	38.②	39.②	40.②	41.②	42.③	43.②	44.①	45.③
46.④	47.④	48.③	49.②	50.④	51.③	52.④	53.③	54.②	55.②	56.②				

1. Section 038

드모르강의 법칙을 적용한다. 드모르강의 법칙은 변수의 개수에 상관없이 적용된다.

$\overline{AB} = \overline{A}+\overline{B}$, $\overline{ABC} = \overline{A}+\overline{B}+\overline{C}$

2. Section 038

①: $A+\overline{A}$는 1이므로 $F(A_1,A_2,\cdots A_n)$의 결과에 관계없이 1이 된다. $A+1=1$을 적용한다.

②: $A\overline{B}+B=(A+B)\cdot(\overline{B}+B)=(A+B)\cdot 1=A+B$

③: $1\oplus 1=0$, $0\oplus 0=0$, 즉 같은 값을 XOR시키면 0이 출력된다.

④: 콘센서스의 법칙에 따라 $\overline{A}\cdot\overline{B}+\overline{B}\cdot\overline{C}+\overline{C}\cdot A=\overline{A}\cdot\overline{B}+\overline{C}\cdot A$

3. Section 038

③: $\overline{A+B} = \overline{A}\cdot\overline{B}$

4. Section 038

$(A+B)\overline{(A\cdot B)}$

$= (A+B)(\overline{A}+\overline{B})$

$= A\overline{A}+A\overline{B}+\overline{A}B+B\overline{B}$ ← $A\overline{A}=0$, $B\overline{B}=0$

$= A\overline{B}+\overline{A}B$

5. Section 038

$Y = \overline{A}\,\overline{B}\,\overline{C}+A\,\overline{B}C+\overline{A}\,B\,C+\overline{A}\,\overline{B}\,\overline{C}$

$= \overline{A}\,\overline{B}\,(\overline{C}+C)+A\overline{B}\,(C+\overline{C})$ ← $A+\overline{A}=1$

$= \overline{A}\,\overline{B}(1)+A\overline{B}\,(1)$ ← $A\cdot 1=A$

$= \overline{A}\,\overline{B}+A\overline{B}$

$= \overline{B}\,(\overline{A}+A)$ ← $A+\overline{A}=1$

$= \overline{B}\,(1)$

$= \overline{B}$ ← $A\cdot 1=A$

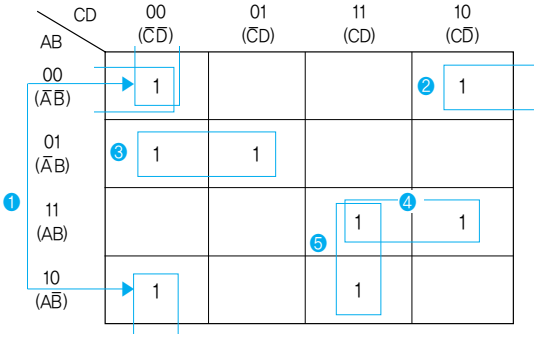
6. Section 038

❶ 4개의 변수에 해당하는 카르노 맵을 그리고 해당하는 위치(0, 2, 4, 5, 8, 11, 14, 15)에 1을 입력한다.

CD / AB	00 ($\overline{C}\,\overline{D}$)	01 ($\overline{C}D$)	11 (CD)	10 ($C\overline{D}$)
00 ($\overline{A}\,\overline{B}$)	0000 : 0 ($\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$) 1	0001 : 1 ($\overline{A}\,\overline{B}\overline{C}D$)	0011 : 3 ($\overline{A}\,\overline{B}CD$)	0010 : 2 ($\overline{A}\,\overline{B}C\overline{D}$) 1
01 ($\overline{A}B$)	0100 : 4 ($\overline{A}B\overline{C}\,\overline{D}$) 1	0101 : 5 ($\overline{A}B\overline{C}D$) 1	0111 : 7 ($\overline{A}BCD$)	0110 : 6 ($\overline{A}BC\overline{D}$)
11 (AB)	1100 : 12 ($AB\overline{C}\,\overline{D}$)	1101 : 13 ($AB\overline{C}D$)	1111 : 15 ($ABCD$) 1	1110 : 14 ($ABC\overline{D}$) 1
10 ($A\overline{B}$)	1000 : 8 ($A\overline{B}\,\overline{C}\,\overline{D}$) 1	1001 : 9 ($A\overline{B}\overline{C}D$)	1011 : 11 ($A\overline{B}CD$) 1	1010 : 10 ($A\overline{B}C\overline{D}$)

A는 0, B는 1, C는 0, D는 0이므로 0100이된다. 1은 참, 0은 거짓이므로 $\overline{A}B\overline{C}\,\overline{D}$가 된다.

❷ 1이 입력되어 이웃하는 칸을 최대 2^i(1, 2, 4, 8, 16 …)개로 묶는다. 한번 묶인 칸이 다른 묶음에 또 묶여도 되며, 1묶음에 묶여지는 칸이 많을수록, 그리고 묶음의 개수가 적을수록 간소화된다.



※ ❶번 묶음은 위와 아래를 합쳐 2개를 한 묶음으로 묶은 것이다.

❸ 묶여진 묶음을 1개로 간주하고 불 함수를 읽는다. 1개의 묶음에 속하는 변수들은 AND 연산시키고, 다른 묶음과는 OR 연산시킨다. 묶음이 0과 1에 모두 속해 있는 변수는 0과 1 아무거나 입력되어도 상관없으므로 무시한다.

❶번 묶음 :
- 변수 A에 대해서는 1, 0에 모두 속하므로 무시한다.
- 변수 B는 0에만 속하므로 $\overline{B}$
- 변수 C는 0 그대로 이므로 $\overline{C}$
- 변수 D는 0 그대로 이므로 $\overline{D}$
- AND로 합치면 $\overline{B}\,\overline{C}\,\overline{D}$이다.

❷번 묶음 :
- 변수 A는 0 그대로 이므로 $\overline{A}$
- 변수 B는 0 그대로 이므로 $\overline{B}$
- 변수 C는 1, 0에 모두 속하므로 무시한다.
- 변수 D는 0에만 속하므로 $\overline{D}$
- AND로 합치면 $\overline{A}\,\overline{B}\,\overline{D}$이다.

❸번 묶음 :
- 변수 A는 0 그대로 이므로 $\overline{A}$
- 변수 B는 1 그대로 이므로 B
- 변수 C는 0에만 속하므로 $\overline{C}$
- 변수 D는 1, 0에 모두 속하므로 무시한다.
- AND로 합치면 $\overline{A}B\overline{C}$이다.

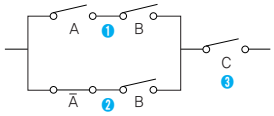
❹번 묶음 :
- 변수 A는 1 그대로 이므로 A
- 변수 B는 1 그대로 이므로 B
- 변수 C는 1에만 속하므로 C
- 변수 D는 1, 0에 모두 속하므로 무시한다.
- AND로 합치면 ABC이다.

❺번 묶음 :
- 변수 A는 1에만 속하므로 A
- 변수 B는 1, 0에 모두 속하므로 무시한다.
- 변수 C는 1 그대로 이므로 C
- 변수 D는 1 그대로 이므로 D
- AND로 합치면 ACD이다.

이어서 ❶, ❷, ❸, ❹, ❺번을 OR로 묶으면 된다.

7. Section 039

스위칭 회로에서 직렬은 AND로, 병렬은 OR로 표현한다.



❶은 직렬이므로 AB이고 ❷도 직렬이므로 $\overline{A}B$이다. ❶과 ❷는 병렬이므로 $AB+\overline{A}B$가 되고 ❸과는 직렬로 연결되므로 $(AB+\overline{A}B)C$가 된다.

$(AB+\overline{A}B)C$
$= (A+\overline{A})BC$
$= 1 \cdot BC$ ← $A+\overline{A}=1$
$= BC$ ← $A \cdot 1=A$

8. Section 038

$(\overline{A}+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})$의 보수는 $\overline{(\overline{A}+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})}$이다.

$\overline{(\overline{A}+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})}$
$= \overline{(\overline{A}+\overline{B}+\overline{C})}+\overline{(\overline{A}+B+\overline{C})}$ ← $\overline{(A \cdot B)}=(\overline{A}+\overline{B})$
$= (\overline{\overline{A}} \cdot \overline{\overline{B}} \cdot \overline{\overline{C}}) + (\overline{\overline{A}} \cdot \overline{B} \cdot \overline{\overline{C}})$ ← $\overline{(A+B)}=(\overline{A} \cdot \overline{B})$
$= A \cdot B \cdot C + A \cdot \overline{B} \cdot C$ ← $\overline{\overline{A}}=A$
$= AC(B+\overline{B})$
$= AC \cdot 1$ ← $A+\overline{A}=1$
$= AC$ ← $A \cdot 1=A$

9. Section 039

논리회로군(Logic Circuit Families)의 성능 평가 요소

- Fan-Out(출력단자 연결회로 수) : 장치의 출력단자로부터 출력되는 신호를 입력으로 받을 수 있는 회로의 수로서, 논리회로의 출력 측에서 다수의 신호가 분배되어 부채꼴로 퍼져나가는 것 같이 보여서 팬 아웃이라함
- Power-Dissipation(전력 소실)
- Propagation Delay(전파 지연) : 논리회로에서 입력된 신호가 출력으로 전파되는 데 걸리는 평균 전이 지연시간으로, 동작 속도는 지연시간에 반비례함

※ Turn Around Time은 컴퓨터가 문제를 처리하여 결과를 반환하는 데 걸리는 반환시간으로, 운영체제를 비롯한 시스템의 성능 평가 기준이다.

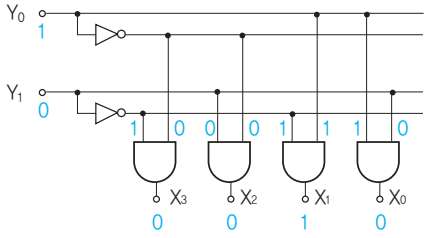
10. Section 039

Propagation Delay(전파 지연)란 논리회로에서 입력된 신호가 출력으로 전파되는 데 걸리는 평균 전이 지연시간이다. 동작 속도는

지연시간에 반비례한다.

11. Section 039

이런 문제는 값을 직접 대입해서 풀면 된다.



13. Section 039

인버터는 NOT 게이트의 다른 이름이다. 즉 0이 입력되면 1을 출력하고 1이 입력되면 0을 출력한다.

①: NAND 게이트는 입력되는 값이 모두 1일 때만 0을 출력한다.

A	B	A NAND B
0	0	1
1	1	0

NAND 게이트의 두 입력단자를 연결하면 입력되는 두 변수에 항상 같은 값이 입력된다. 0을 입력하면 0과 0이 입력되어 1을 출력하고, 1을 입력하면 1, 1이 되어 0을 출력한다.

②: Exclusive NOR는 입력되는 신호가 모두 같을 때만 1을 출력하는 게이트다.

A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

게이트의 한 선이 1로 세트된 상태에서는 0을 입력하면 0이 출력되고, 1을 입력하면 1이 출력되므로 인버터의 역할을 하지 못한다.

③: Exclusive OR는 입력되는 신호가 모두 같을 때만 0을 출력하는 게이트다.

A	B	A NOR B
0	0	0
0	1	1
1	0	1
1	1	0

게이트의 한 선이 1로 세트된 상태에서는 0을 입력하면 1이 출력되고, 1을 입력하면 0이 출력된다.

④: NOR는 입력되는 신호가 모두 0일 때만 1을 출력하는 게이트다.

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

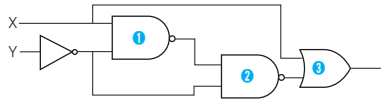
게이트의 한 선이 0으로 세트된 상태에서는 0을 입력하면 1이 출력되고, 1을 입력하면 0이 출력된다.

14. Section 039

- 1의 보수는 0은 1로, 1은 0으로 바꾸어 출력하면 되므로 NOT Gate로 쉽게 구할 수 있다.

※ 1의 보수는 Section 051에서 학습합니다.

15. Section 039



문제의 논리회로를 논리식으로 표현하면, ❶은 $X\overline{Y}$이고 ❷는 ❶$\overline{Y}$이며 ❸은 X+❷이므로

$X+\overline{X\overline{Y}\overline{Y}}$가 되며, 간략화하면 다음과 같다.

$X+\overline{X\overline{Y}\overline{Y}}$

$= X+\overline{X\overline{Y}}+\overline{\overline{Y}}$

$= X+X\overline{Y}+Y$

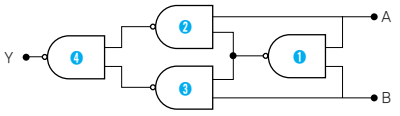
$= X(1+\overline{Y})+Y$

$= X(1)+Y$

$= X+Y$

16. Section 039

문제에 제시된 그림은 X-OR를 표현한 논리회로이다. 논리회로에 사용된 각 논리 게이트를 분리하여 논리식으로 표현한 후 1개의 논리식으로 합치면 된다.



❶ : $\overline{AB}$

❷ : $\overline{A\ ❶} = \overline{(A)(\overline{AB})}$

❸ : $\overline{❶\ B} = \overline{(\overline{AB})(B)}$

❹ : $\overline{❷ \cdot ❸} = \overline{\overline{(A)(\overline{AB})}\,\overline{(\overline{AB})(B)}}$

$\overline{\overline{(A)(\overline{AB})}\,\overline{(\overline{AB})(B)}}$

$= \overline{A(\overline{AB})} + \overline{(\overline{AB})B}$

$= A(\overline{A}+\overline{B}) + (\overline{A}+\overline{B})B$

$= A\overline{A} + A\overline{B} + \overline{A}B + \overline{B}B$ ← $A\overline{A}=0$, $B\overline{B}=0$

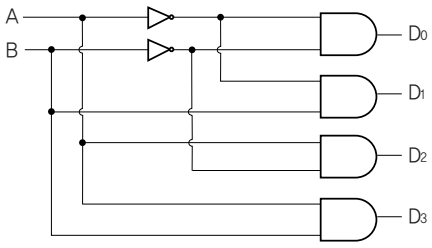
$= 0 + A\overline{B} + \overline{A}B + 0$

$= A\overline{B} + \overline{A}B = A \oplus B$

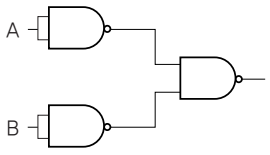
17. Section 041

디코더는 주로 AND 게이트로 이루어져 있다. 2×4 디코더의 경우 2개의 NOT 게이트와 4개의 AND 게이트로 구성된다.

2×4 디코더



18. Section 039



$\overline{A\overline{A}}\ \overline{B\overline{B}}$ ← $\overline{AB}=\overline{A}+\overline{B}$ 적용

$=\overline{A\overline{A}}+\overline{B\overline{B}}$ ← $\overline{\overline{A}}=A$ 적용

$=(AA)+(BB)$ ← $AA=A$ 적용

$=A+B$ 이므로 OR 게이트와 같다.

19. Section 039

출력이 1인 부분에 대한 입력 변수들을 AND 연산한 후 각각을 OR시킨 다음 간소화한다.

입력 ABC	출력 Y	
000	1	$\overline{A}\overline{B}\overline{C}$
001	0	
010	1	$\overline{A}B\overline{C}$
011	0	
100	1	$A\overline{B}\overline{C}$
101	0	
110	1	$AB\overline{C}$
111	0	

$Y=\overline{A}\overline{B}\overline{C}+\overline{A}B\overline{C}+A\overline{B}\overline{C}+AB\overline{C}$

$=\overline{A}\overline{C}(\overline{B}+B)+A\overline{C}(\overline{B}+B)$ ← $\overline{B}+B=1$

$=\overline{A}\overline{C}+A\overline{C}$

$=(\overline{A}+A)\overline{C}=\overline{C}$

20. Section 042

- **조합논리회로** : 반가산기, 전가산기, 병렬가산기, 반감산기, 전감산기, 디코더, 인코더, 멀티플렉서, 디멀티플렉서, 다수결회로, 비교기 등
- **순서논리회로** : 플립플롭, 카운터, 레지스터, RAM, CPU 등

21. Section 040

반가산기

〈진리표〉

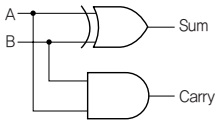
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

〈논리식〉

Carry=A·B

Sum=A⊕B

〈회로도〉



22. Section 040

출력(F)이 1인 것에 해당하는 입력 변수들을 AND 연산한 후 각각을 OR 연산자로 연산하여 표현한다(0이면 부정($\overline{A}$, $\overline{B}$), 1이면 긍정(A, B)).

A	B	F	
0	0	0	
0	1	1	❶
1	0	1	❷
1	1	0	

❶ $\overline{A}B$, ❷ $A\overline{B}$

F= ❶ + ❷ $=\overline{A}B+A\overline{B}$

24. Section 040

전가산기의 논리식

- 합(S) $=\overline{X}\overline{Y}Z+\overline{X}Y\overline{Z}+X\overline{Y}\overline{Z}+XYZ$
  $=(\overline{X}\overline{Y}+XY)Z+(\overline{X}Y+X\overline{Y})\overline{Z}$
  $=\overline{(X\oplus Y)}Z+(X\oplus Y)\overline{Z}$
  $=(X\oplus Y)\oplus Z$
- 자리올림(C) $=\overline{X}YZ+X\overline{Y}Z+XY\overline{Z}+XYZ$
  $=(\overline{X}Y+X\overline{Y})Z+XY(\overline{Z}+Z)$
  $=(X\oplus Y)Z+XY$

25. Section 041

Carry-L●●k-Ahead(자리올림수 예견) 회로

부분 가산기 등에 의해서 제공되는 확산, 발생 기호들로부터 최종 자리올림수를 예견하는 것으로, 이는 자리 올림수가 전달되는 전파 지연시간을 해소함으로써 2진 가산 속도를 현저히 빠르게 할 수 있다.

26. Section 041

산술 연산에서 Overflow가 발생하는 경우는 부호 Bit가 변경될 때이다. 따라서 Overflow를 검출하려면 연산 후 부호 비트(양수 0, 음수 1)가 제대로 유지되었는지 비교해 보면 알 수 있다. 유지해야 할 부호값과 연산 후의 부호 비트 값을 비교하여 다르면 1을 출력시켜 Overflow가 발생했음을 나타내야 하므로, 입력되는 값이 하나라도 다르면 1이 출력되는 XOR 게이트가 사용된다.

27. Section 039

일치 여부를 비교하여 일치하지 않으면 0, 일치하면 1을 출력시키기 위해서는 XNOR 게이트를 사용한다.

A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

28. Section 041

병렬 가산기는 nBit로 된 2진수를 더하기 위해 n개의 전가산기를 사용하는 실질적인 가산기다. 이런 문제는 임의의 입력값을 직접 대입하여 계산해 보면 쉽게 풀린다. 다른 입력 자료는 고정되어 있으므로 A만 임의의 숫자로 지정하면 된다.

A에 '0101' 이 입력되었다고 가정하면,

A : 0 1 0 1
0 : 0 0 0 0
C : 1
0 1 1 0

즉 A의 값이 5에서 6으로 1 증가하는 결과가 되었다.

※ C는 자리올림수이므로 1Bit만 입력된다.

29. Section 041

- 문제에 주어진 그림은 디코더이다.
- 디코더는 주로 번역기로 사용하지만, 장치번호 번역기는 입 · 출력장치 측의 인터페이스에 포함되어 있는 요소로서, 단순히 CPU가 보내준 장치번호가 해당 입 · 출력장치의 번호인지 아닌지만 판단하기 때문에 Decoder 회로를 사용하지 않는다.

30. Section 041

- 디코더 : $n \rightarrow 2^n$
- 인코더 : $2^n \rightarrow n$
- 디멀티플렉서 : $1 \rightarrow 2^n$
- 멀티플렉서 : $2^n \rightarrow 1$

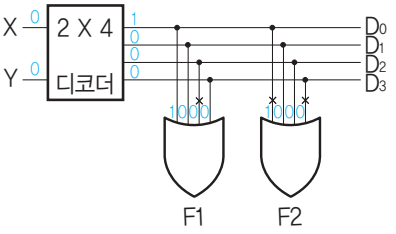
※ 그림은 2^n개의 입력 중 1개의 입력 선을 선택하여 출력하기 위해 n개의 선택 선을 사용하는 멀티플렉서이다. 출력에 Q와 $\overline{Q}$가 있어 두 개의 선이 있는 것이 아니다. $\overline{Q}$는 Q의 부정을 의미한다.

31. Section 041

그림에 맞게 F1과 F2로 입력되는 4개의 디코더 출력선을 구해보면 다음과 같다.

	X	Y	출력선	출력선	F1	F2
❶	0	0	D_0	$\overline{X}\,\overline{Y}$	1	0
	0	1	D_1	$\overline{X}\,Y$	1	1
	1	0	D_2	$X\,\overline{Y}$	0	1
❷	1	1	D_3	$X\,Y$	1	0

- ❶ X와 Y에 0이 입력되면 디코더의 출력은 0번 선, 즉 D_0만 1이 되고 나머지는 0이 된다. 그러면 F1으로 입력되는 4개의 선 중 맨 왼쪽의 선에만 1이 입력되지만 OR 게이트는 입력값 중 한 개라도 1이 있으면 1이 출력되므로 F1은 1을 출력한다. F2도 입력되는 4개의 선 중 맨 왼쪽의 선에만 1이 입력되지만 선이 절단되었으므로 1이 입력되는 선이 하나도 없게 된다. 결국 F2는 0을 출력한다. 그림으로 보면 다음과 같다.



- ❷ 이제 X와 Y에 1이 입력될 때를 살펴보자. X와 Y에 모두 1이 입력되면 디코더의 출력은 3번 선, 즉 D_3만 1이 되고 나머지는 0이 된다. 그러면 F1으로 입력되는 4개의 선 중 왼쪽에서 네 번째 선에만 1이 입력되지만 OR 게이트는 입력값 중 한 개라도 1이 있으면 1이 출력되므로 F1은 1을 출력한다. F2도 입력되는 4개의 선 중 왼쪽에서 네 번째 선으로만 1이 입력되지만 선이 절단되었으므로 1이 입력되는 선이 하나도 없다. 결국 F2는 0을 출력한다.

32. Section 041

$Y=A\cdot B+\overline{A}\cdot\overline{B}=A\odot B$ 즉, XNOR 기능을 이용하는 회로로서 비교기, 홀수 패리티 검사기 등에서 사용한다.

33. Section 042

조합논리회로와 순차논리회로의 차이점

구분	조합논리회로	순서논리회로
기억 기능	없다	있다
구성 요소	논리소자	논리소자, 플립플롭(FF)
출력 신호	입력 신호에 의해서만 결정	입력 신호와 현재의 상태에 의해서 결정
종류	반가산기, 전가산기, 병렬가산기, 반감산기, 전감산기, 디코더, 인코더, 디멀티플렉서, 멀티플렉서, 다수결회로, 비교기(일치회로, 반일치회로)	기억기능을 가지는 모든 회로(2진 카운터, 시프트 레지스터, RAM 등)

34. Section 042

순서논리회로는 조합회로와 Flip-Flop 같은 기억 기능이 있는 소자로 구성된다.

36. Section 042

동기식 동작(Synchronous Operation)

각 사건이나 동작의 수행이 한 Clock에서 발생하는 제어 신호의 결과로 동시에 일어나는 것을 말한다.

37. Section 042

RS 플립플롭에 S=1, R=1이 입력되면 동작되지 않는다.

S	R	$Q_{(t+1)}$
0	0	$Q_{(t)}$상태가 변화 없음
0	1	0
1	0	1
1	1	동작 안됨

※ 주의 : 문제에는 S와 R이 바뀌어 있다.

38. Section 042

D 플립플롭

- D플립플롭은 JK 플립플롭의 하나의 선에 Inverter를 추가하여 다른 선과 묶어 입력선을 한 개로 만든 플립플롭이다.
- Clock이 발생하면 입력선으로 입력된 값을 그대로 저장하는 기능을 수행한다.

D	$Q_{(t+1)}$
0	0
1	1

D 플립플롭의 특성표

J	K	$Q_{(t+1)}$
0	0	$Q_{(t)}$
0	1	0
1	0	1
1	1	$\overline{Q}_{(t)}$

JK 플립플롭의 특성표

입력선으로 0을 입력하면 J = 0, K = 1이고, 1을 입력하면 J = 1, K = 0인 두 가지 상태만 발생하므로 D 플립플롭과 같은 기능을 한다.

※ Toggle 기능을 가지고 있으며, 카운터에 이용되는 플립플롭은 T 플립플롭이다.

39. Section 042

RS 플립플롭에 AND 회로를 연결하여 만든 플립플롭은 JK 플립플롭이다.

J	K	$Q_{(t+1)}$
0	0	$Q_{(t)}$
0	1	0
1	0	1
1	1	$\overline{Q}_{(t)}$

입력되는 두 선 중 하나만 1이면 Q_{n+1}은 0이나 1이다.

40. Section 042

RS 플립플롭에서 S=R=1일 때 동작되지 않는 단점을 해소한 플립플롭은 JK 플립플롭이다. JK 플립플롭은 J=K=1일 때 Q_{t+1}의 상태는 보수이다.

41. Section 041

이 문제는 출력식이 간단하므로 카르노 맵을 이용하는 것보다 진리표를 그려서 푸는 것이 이해하기 쉽다. $F=\overline{B}+(\overline{AC})$에 대한 진리표를 그려서 1이 출력되는 입력 번호를 적으면 된다.

❶ A, B, C 세 변수에 대한 빈 진리표를 그린다.

A	B	C	F	일련번호
0	0	0		0
0	0	1		1
0	1	0		2
0	1	1		3
1	0	0		4
1	0	1		5
1	1	0		6
1	1	1		7

❷ $F=\overline{B}+(\overline{AC})=\overline{B}+\overline{A}+\overline{C}$와 같고 각각의 변수가 OR로 결합되어 있으므로 $\overline{A}$ 또는 $\overline{B}$ 또는 $\overline{C}$가 입력되면 1이 출력된다. 먼저 $\overline{A}$에 해당하는 부분에 1을 표시한다. A=1, $\overline{A}$=0이므로 A가 0인 부분에 1을 표시한다.

※ $(\overline{AC})$는 드모르강의 정리에 의해 $\overline{A}+\overline{B}$이다.

A	B	C	F	일련번호
0	0	0	1	0
0	0	1	1	1
0	1	0	1	2
0	1	1	1	3
1	0	0		4
1	0	1		5
1	1	0		6
1	1	1		7

❸ $\overline{B}$에 해당하는 부분에 1을 표시한다. B=1, $\overline{B}$=0이므로 B가 0인 부분에 1을 표시한다. 중복되는 부분은 그대로 둔다.

A	B	C	F	일련번호
0	0	0	1	0
0	0	1	1	1
0	1	0	1	2
0	1	1	1	3
1	0	0	1	4
1	0	1	1	5
1	1	0		6
1	1	1		7

❹ $\overline{C}$에 해당하는 부분에 1을 표시한다. C=1, $\overline{C}$=0이므로 C가 0인 부분에 1을 표시한다. 중복되는 부분은 그대로 둔다.

A	B	C	F	일련번호
0	0	0	1	0
0	0	1	1	1
0	1	0	1	2
0	1	1	1	3
1	0	0	1	4
1	0	1	1	5
1	1	0	1	6
1	1	1		7

1이 표시된 0, 1, 2, 3, 4, 5, 6에 입력이 있으면 $F=\overline{B}+(\overline{AC})$의 함수식에 맞는 결과가 출력된다.

42. Section 042

T 플립플롭

- T 플립플롭은 JK FF의 두 입력선을 묶어서 한 개의 입력선으로 구성한 플립플롭이다.
- T=0인 경우는 변화가 없고, T=1인 경우 현재의 상태를 토글(Toggle)시킨다. 즉 원 상태와 보수 상태의 두 가지 상태로만 서로 전환된다.

T	$Q_{(t+1)}$
0	$Q_{(t)}$
1	$\overline{Q_{(t)}}$

T 플립플롭의 특성표

J	K	$Q_{(t+1)}$
0	0	$Q_{(t)}$
0	1	0
1	0	1
1	1	$\overline{Q_{(t)}}$

JK 플립플롭의 특성표

JK 플립플롭의 두 입력선을 묶어서 입력선으로 0을 입력하면 J = 0, K = 0, 1을 입력하면 J = 1, K = 1인 두 가지 상태만 발생하므로 T 플립플롭과 같은 기능을 한다.

43. Section 042

※ 트리거 플립플롭(Trigger Flip-Flop)은 T 플립플롭을 말한다.

T 플립플롭

- T 플립플롭은 JK FF의 두 입력선을 묶어서 한 개의 입력선으로 구성한 플립플롭이다.
- T=0인 경우는 변화가 없고, T=1인 경우에 현재의 상태를 토글(Toggle)시킨다. 즉 원 상태와 보수 상태의 두 가지 상태로만 서로 전환된다.

T	$Q_{(t+1)}$
0	$Q_{(t)}$
1	$\overline{Q_{(t)}}$

44. Section 042

Toggle

두 가지 상태에 대해서 입력이 이루어질 때마다 서로 반대 상태로 전환시키는 것을 말하는 것으로, JK 플립플롭에서는 J, K선 모두 1일 때 이루어지고, T 플립플롭에서는 T선이 1일 때 이루어진다.

45. Section 040

플립플롭 한 개가 1비트를 나타내므로 3비트 레지스터이며, Clock Pulse가 한 번 발생할 때마다 다음과 같이 0~7까지 일정하게 변하므로 8진 카운터이다.

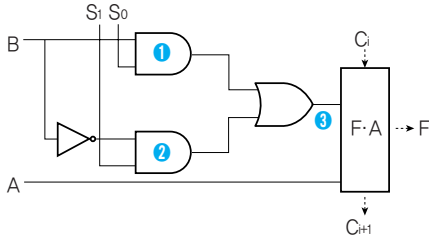
		$A_2\,A_1\,A_0$
초기상태		0 0 0
클록 순서	t_1	1 1 1
	t_2	1 1 0
	t_3	1 0 1
	t_4	1 0 0
	t_5	0 1 1
	t_6	0 1 0
	t_7	0 0 1
	t_8	0 0 0

※ 이런 문제는 값을 대입하여 추적하기보다는 3Bit로 나타낼 수 있는 수가 0~7이라는 것을 염두에 두고 문제를 해결하는 것이

좋을 것 같다.

46. Section 040

F·A는 전가산기를 말한다. 전가산기의 출력은 F=(A⊕B)⊕C_i이다. C_i는 1, A는 A 그리고 B는 다음과 같이 계산한다.



- ❶은 BS_0, ❷는 $\overline{B}S_1$이므로 ❸에 해당하는 F · A의 입력은 $BS_0 + \overline{B}S_1$인데 S_0와 S_1이 1이므로 B · 1+$\overline{B}$ · 1이 된다. B · 1+$\overline{B}$ · 1=B+$\overline{B}$=1
- B에 해당하는 입력값은 1이므로 F=(A⊕1)⊕1=$\overline{A}$⊕1=A

※ A⊕1에서 A에 대입될 수 있는 값이 0이나 1이라는 거는 알죠? A에 값을 대입해 보면 1⊕1=0, 0⊕1=1이 됩니다. 즉 A값에 대한 반대값이 결과로 출력되므로 A⊕1의 출력은 $\overline{A}$입니다.

47. Section 041

멀티플렉서(MUX, Multiplexer)

- 멀티플렉서는 2^n의 입력선 중 한 개를 선택하여 그 선으로부터 입력되는 값을 한 개의 출력선으로 출력시키는 회로이다.
- 2^n개의 입력선 중 한 개의 선을 선택하기 위해 n개의 선택선(Select Line)을 이용한다.

48. Section 042

결선 게이트란 각 게이트들의 출력 단자들을 직접 선으로 연결하여 논리 기능을 발휘하도록 하는 게이트로서 많은 논리 기능을 부여할 수 있다.

49. Section 038

F=XY+$\overline{X}$Z에 대한 진리표를 그려서 1이 출력되게 하는 입력 번호를 적으면 된다.

❶ X, Y, Z 세 변수에 대한 빈 진리표를 그린다.
XY와 $\overline{X}$Z가 OR로 결합되어 있으므로 XY가 입력되거나 $\overline{X}$Z가 입력되면 F는 1이 출력된다. 먼저 XY에 해당하는 부분에 1을 표시한다. X=1, $\overline{X}$=0, Y=1, $\overline{Y}$=0이므로 X, Y 모두 1인 부분에 1을 표시한다.

X	Y	Z	F	일련번호
0	0	0		0
0	0	1		1
0	1	0		2
0	1	1		3
1	0	0		4
1	0	1		5
1	1	0	1	6
1	1	1	1	7

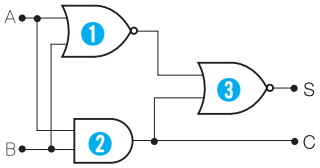
❷ $\overline{X}$Z에 해당하는 부분에 1을 표시한다. X=1, $\overline{X}$=0, Z=1, $\overline{Z}$=0이므로 X가 0이고 Z가 1인 부분에 1을 표시한다.

X	Y	Z	F	일련번호
0	0	0		0
0	0	1	1	1
0	1	0		2
0	1	1	1	3
1	0	0		4
1	0	1		5
1	1	0	1	6
1	1	1	1	7

❸ 1이 표시된 1, 3, 6, 7부분이 최소항의 합이고 다음과 같이 표시한다.

F(X, Y, Z) = Σ(1, 3, 6, 7)

50. Section 040



❶은 $\overline{(A+B)}$이고, ❷는 AB이므로 ❸은 $\overline{(❶+❷)}$, 곧 $\overline{(\overline{(A+B)}+AB)}$이다. 간략화하면 다음과 같다.

S = $\overline{(\overline{(A+B)}+AB)}$

= $\overline{(A+B)}$ · $\overline{(AB)}$

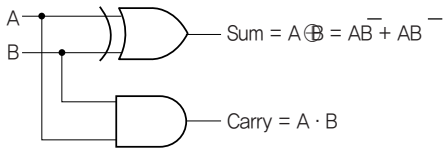
= (A+B) · $\overline{(AB)}$

= (A+B) · ($\overline{A}$+$\overline{B}$)

= A⊕B

C = AB

※ 간략화한 회로를 다시 그리면 반가산기가 그려진다.



51. Section 040

전가산기는 자리올림(C)과 이진수 두 비트(A, B), 즉 이진수 세 비트를 더해 합과 캐리(자리올림)를 구하는 논리회로이다. 다음은 전가산기의 진리표이다.

A	B	C	Sum	Carry
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

진리표를 참조하지 않더라고 A, B, C 에 입력되는 값을 그대로 더해서 합과 캐리를 구하면 된다. A=1, B=0, C=1 이므로 결과는 다음과 같다.

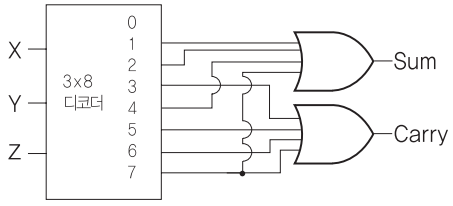
$$\begin{array}{r} 1 \\ 0 \\ +\,1 \\ \hline 10 \end{array}$$

합(S_0)은 0이 되고 자리올림(C_0)은 1이 된다.

52. Section 041

전가산기 진리표에서 Sum과 Carry에 대해 각각 1이 출력되는 번호와 같은 번호를 3×8 디코더에서 선택하여 Sum과 Carry에 해당하는 OR 게이트의 입력에 연결시키면 된다.

X	Y	Z	Sum	Carry	일련번호
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	2
0	1	1	0	1	3
1	0	0	1	0	4
1	0	1	0	1	5
1	1	0	0	1	6
1	1	1	1	1	7



53. Section 039

입력이 2개인 NAND 게이트만으로 회로를 구성하라고 했으니 논리식의 변수가 2개가 되도록 간략화해야 한다.

$F = \overline{A}\,\overline{B}C + A\overline{B}C + ABC$

$= \overline{A}\,\overline{B}C + AC(\overline{B}+B)$ ← $\overline{B}+B=1$

$= \overline{A}\,\overline{B}C + AC(1)$ ← $A \cdot 1 = A$

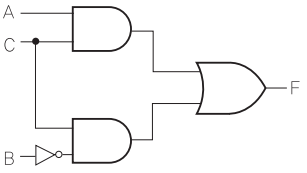
$= C(\overline{A}\,\overline{B}+A)$

$= C(\,(\overline{A}+A)\,(\overline{B}+A)\,)$ ← $\overline{A}+A=1$

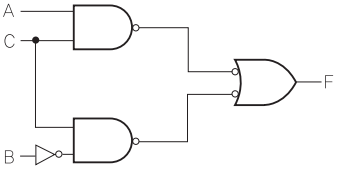
$= C(\,(1)(\overline{B}+A)\,)$ ← $A \cdot 1 = A$

$= \overline{B}C + AC$

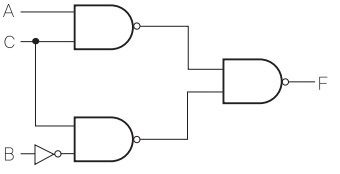
$F = AC + \overline{B}C$를 그대로 논리회로로 그리면 다음과 같다.



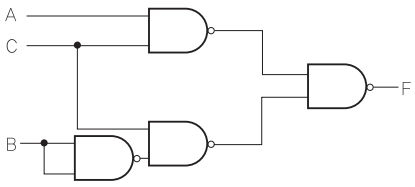
부정의 부정은 원래의 값과 같은 값을 출력하니 각각에 대해 NOT 게이트 두 개를 붙여 같은 결과를 출력하는 NAND 게이트로 변경할 수 있다.

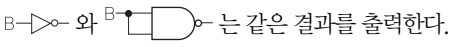


드모르강 법칙에 의하면 $\overline{A}+\overline{B}=\overline{(AB)}$이 되므로 다음과 같이 변경할 수 있다.



그런데 문제는 2개의 입력을 받는 NAND만을 사용하라 했으니 $\overline{C}$를 다음과 같이 2입력의 NAND로 변경해야 한다.





와 는 같은 결과를 출력한다.

∴ 결론적으로 논리식 F=AC+$\overline{B}$C는 총 4개의 NAND 게이트를 사용하여 회로를 구성할 수 있다.

54. Section 038

❶ 네 변수에 해당하는 카르노 맵을 그리고 F 함수의 해당 위치에 1을 입력한다.

wx \ yz	00 ($\overline{y}\overline{z}$)	01 ($\overline{y}z$)	11 (yz)	10 ($y\overline{z}$)
00 ($\overline{w}\overline{x}$)	0000 : 0 ($\overline{w}\overline{x}\overline{y}\overline{z}$)	0001 : 1 ($\overline{w}\overline{x}\overline{y}z$) 1	0011 : 3 ($\overline{w}\overline{x}yz$) 1	0010 : 2 ($\overline{w}\overline{x}y\overline{z}$)
01 ($\overline{w}x$)	0100 : 4 ($\overline{w}x\overline{y}\overline{z}$)	0101 : 5 ($\overline{w}x\overline{y}z$)	0111 : 7 ($\overline{w}xyz$) 1	0110 : 6 ($\overline{w}xy\overline{z}$)
11 (wx)	1100 : 12 ($wx\overline{y}\overline{z}$)	1101 : 13 ($wx\overline{y}z$)	1111 : 15 ($wxyz$) 1	1110 : 14 ($wxy\overline{z}$)
10 ($w\overline{x}$)	1000 : 8 ($w\overline{x}\overline{y}\overline{z}$)	1001 : 9 ($w\overline{x}\overline{y}z$)	1011 : 11 ($w\overline{x}yz$) 1	1010 : 10 ($w\overline{x}y\overline{z}$)

❷ 이어서 don't care 조건인 d 함수의 해당 위치에 X를 입력한다.

wx \ yz	00 ($\overline{y}\overline{z}$)	01 ($\overline{y}z$)	11 (yz)	10 ($y\overline{z}$)
00 ($\overline{w}\overline{x}$)	0000 : 0 ($\overline{w}\overline{x}\overline{y}\overline{z}$) ×	0001 : 1 ($\overline{w}\overline{x}\overline{y}z$) 1	0011 : 3 ($\overline{w}\overline{x}yz$) 1	0010 : 2 ($\overline{w}\overline{x}y\overline{z}$) ×
01 ($\overline{w}x$)	0100 : 4 ($\overline{w}x\overline{y}\overline{z}$)	0101 : 5 ($\overline{w}x\overline{y}z$) ×	0111 : 7 ($\overline{w}xyz$) 1	0110 : 6 ($\overline{w}xy\overline{z}$)
11 (wx)	1100 : 12 ($wx\overline{y}\overline{z}$)	1101 : 13 ($wx\overline{y}z$)	1111 : 15 ($wxyz$) 1	1110 : 14 ($wxy\overline{z}$)
10 ($w\overline{x}$)	1000 : 8 ($w\overline{x}\overline{y}\overline{z}$)	1001 : 9 ($w\overline{x}\overline{y}z$)	1011 : 11 ($w\overline{x}yz$) 1	1010 : 10 ($w\overline{x}y\overline{z}$)

❸ 1이나 X가 입력되어 이웃하는 칸을 최대 2ⁱ(1, 2, 4, 8, 16 …)개로 묶는다. 한번 묶인 칸이 다른 묶음에 또 묶여도 된다. 1묶음에 묶여지는 칸이 많을수록, 그리고 묶음의 개수가 적을수록 더 간소화된다. X는 신경 쓰지 않아도 되는 변수로 많은 수로 묶기 위해 필요할 때만 사용하면 된다.

wx \ yz	00 ($\overline{y}\overline{z}$)	01 ($\overline{y}z$)	11 (yz)	10 ($y\overline{z}$)
00 ($\overline{w}\overline{x}$)	× ❶	1	1 ❷	×
01 ($\overline{w}x$)		×	1	
11 (wx)			1	
10 ($w\overline{x}$)			1	

❹ 묶여진 묶음을 1개로 간주하고 불 함수를 읽는다. 한 개의 묶음에 속하는 변수들은 AND 연산시키고, 다른 묶음과는 OR 연산시킨다. 묶음이 0과 1에 모두 속해 있는 변수는 0과 1 아무거나 입력되어도 상관없으므로 무시한다.

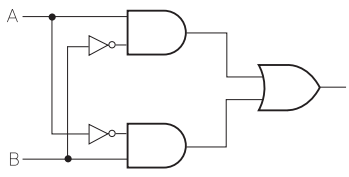
❶ 번 묶음
- 변수 w는 0에만 속하므로 $\overline{w}$
- 변수 x는 1, 0에 모두 속하므로 무시한다.
- 변수 y는 1, 0에 모두 속하므로 무시한다.
- 변수 z는 1에만 속하므로 z
- AND로 합치면 $\overline{w}z$이다.

❷ 번 묶음
- 변수 w는 1, 0에 모두 속하므로 무시한다.
- 변수 x는 1, 0에 모두 속하므로 무시한다.
- 변수 y는 1에만 속하므로 y
- 변수 z는 1에만 속하므로 z
- AND로 합치면 yz이다.

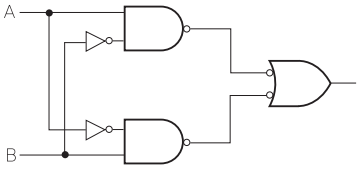
이어서 ❶과 ❷번을 OR로 묶으면 $\overline{w}z+yz$이 된다.

55. Section 041

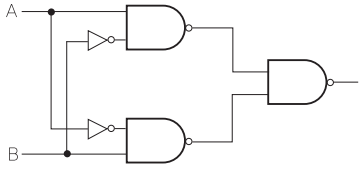
EX-OR 회로의 논리식 A⊕B= $A\overline{B}+\overline{A}B$를 AND, OR, NOT 게이트를 이용해서 논리회로로 그리면 다음과 같다.



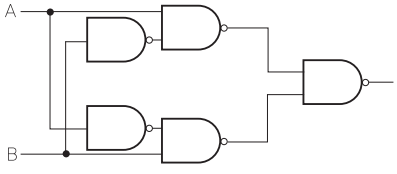
이것을 NAND 게이트만 이용하여 같은 기능을 수행하게 하려면, 부정의 부정이 원래의 값과 같은 값을 출력하는 원리를 이용하여 각각에 대해 NOT 게이트 두 개를 붙여 같은 결과를 출력하는 NAND 게이트로 변경한다.



드모르강 법칙에 의하면 $\overline{A}+\overline{B}=\overline{(AB)}$이 되므로 다음과 같이 변경할 수 있다.



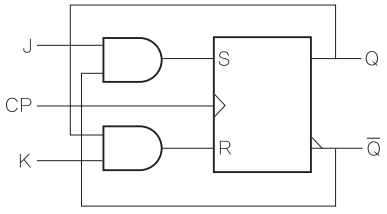
그런데 문제는 NAND만을 사용하라 했으니 Not 대신 입력이 하나인 NAND로 변경하면 된다.



∴ 결론적으로 논리식 EX-OR 회로는 총 5개의 NAND 게이트를 사용하여 같은 동작을 하도록 구현할 수 있다.

56. Section 042

다음과 같이 RS FF의 입력선 S와 R에 AND 게이트 2개를 추가하여 JK FF의 입력선 J와 K로 사용한다.



JK 플립플롭

- RS FF에서 S = R = 1일 때 동작되지 않는 결점을 보완한 플립플롭이다.
- RS FF의 입력선 S와 R에 AND 게이트 2개를 추가하여 JK FF의 입력선 J와 K로 사용한다.
- 다른 모든 플립플롭의 기능을 대용할 수 있으므로 응용 범위가 넓고 집적 회로화 되어 가장 널리 사용된다.
- 특성표

J	K	$Q_{(t+1)}$	상태
0	0	$Q_{(t)}$	상태 변화 없음(무)
0	1	0	Reset(공)
1	0	1	Set(일)
1	1	$\overline{Q}_{(t)}$	반전(보)

2장 정답 및 해설 — 자료의 표현

1. ④	2. ②	3. ②	4. ②	5. ③	6. ③	7. ④	8. ③	9. ③	10. ①	11. ②	12. ①	13. ②	14. ①	15. ①
16. ③	17. ①	18. ②	19. ②	20. ③	21. ①	22. ②	23. ④	24. ④	25. ②	26. ③	27. ③	28. ①	29. ①	30. ③
31. ③	32. ④	33. ②	34. ④	35. ①	36. ③	37. ①	38. ②	39. ④	40. ②	41. ①	42. ①	43. ②	44. ②	45. ③
46. ①	47. ②	48. ④	49. ②											

1. Section 049

2진화 10진수는 10진수를 4비트의 2진수로 표시하는 것으로 10진수가 0~9까지 10개의 숫자를 사용하므로 표현할 수 있는 가짓수는 10개이고, 2진화 16진수는 16진수를 4비트의 2진수로 표시하는 것으로 16진수가 0~F(15)까지 16개의 숫자를 사용하므로 표현할 수 있는 가짓수는 16개이다. 즉 둘의 차이는 6이다.

일련 번호	10진수	2진화 10진수	16진수	2진화 16진수
1	0	0000	0	0000
2	1	0001	1	0001
3	2	0010	2	0010
4	3	0011	3	0011
5	4	0100	4	0100
6	5	0101	5	0101
7	6	0110	6	0110
8	7	0111	7	0111
9	8	1000	8	1000
10	9	1001	9	1001
11			A (10)	1010
12			B (11)	1011
13			C (12)	1100
14			D (13)	1101
15			E (14)	1110
16			F (15)	1111

2. Section 043

- Full Word = 4Byte
- Half Word = 2Byte = Full Word의 반
- Double Word = 8Byte = Full Word의 배

3. Section 043

정보의 최소 단위인 1개의 0 또는 1의 정보를 표현하기 위해서는 1Bit를 사용하지만, 문자를 표시하기 위해서는 최소한 8Bit가 모여서 구성되는 Byte가 필요하다.

4. Section 044

2 | 6
2 | 3 … 0
1 … 1

6=110
0.125=0.001
6.125=110.001

0.125 × 2 = 0.250
0.250 × 2 = 0.500
0.500 × 2 = 1.000

5. Section 044

1 0 1 1 . 1 1 0 1

$2^3 + 2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4}$

8 0 2 1 0.5 0.25 0 0.0625

8+2+1+0.5+0.25+0.0625 = 11.8125

6. Section 044

오른쪽에서 왼쪽으로 2진수 3자리를 묶어서 8진수 1자리로 표현한다.

1 0 1 1 1 1 1 1 0
5 7 6

7. Section 044

8진수 1자리를 2진수 3자리로 표현한 후 2진수 4자리를 묶어 16진수 1자리로 표현한다.

2 6 5 ←8진수
0 1 0 1 1 0 1 0 1 ←2진수
0 11(B) 5 ←16진수

8. Section 044

10진수 12를 3진수로 변환하려면 3으로 나누어서 몫과 나머지를 구한다. 나머지가 3보다 작을 때까지 반복한 후 뒤에서부터 나머지를 적는다.

3 | 12
3 | 4 … 0
1 … 1

9. Section 045

1의 보수의 특징

- 음수의 표현이 간단하다(보수를 만들기 쉽다) : 0은 1로, 1은 0으로 바꾸어준다.
- +0과 −0, 즉 두 가지 형태의 0이 있다.
- 덧셈의 경우 최상위 비트에서 오버플로가 생기면 그때마다 1을 더해주어야 하므로 연산 과정이 복잡하다.

10. Section 045

고정 소수점 숫자에는 소수점이 포함되어 있지 않으므로 소수점을 표시하기 위해 사용하는 비트는 없다. 이는 소수점이 포함된 숫자인 부동 소수점 방식에서도 마찬가지다. 부동 소수점 방식에서는 소수점에 대한 정보 없이 8비트 이하의 비트를 모두 소수 이하의 수로 간주한다.

11. Section 046

2의 보수로 표현된 수치는 먼저 10진수로 변경하여 계산한 후 2의 보수로 변환하면 된다.

❶ 첫 번째 비트가 1이므로 음수값이다. 음수값은 다시 2의 보수를 취해야만 그 값의 크기를 알 수 있다. 101011과 100110에 대한 2의 보수를 구한 후 10진수로 변환한다.
101011 → 010101 → −21

100110 → 011010 → −26

❷ 계산한다.

−21 − (−26) = −21 + 26 = 5

❸ 이진수로 변환한다.

5 → 000101

12. Section 046

고정 소수점 방식이란 표현된 수치 자료의 맨 오른쪽에 소수점이 고정되어 있다고 가정하고 정수만 표현하는 방법이다. 반면 부동(浮動) 소수점 방식이란 소수점의 위치를 이동(부동(浮動))시켜 정규화하는 방법으로, 소수점이 포함된 수치 자료를 표현할 때 사용한다. 즉 고정 소수점 수는 부호와 수만 있다.

13. Section 046

양수 A와 B에 대해 2의 보수 표현 방식을 사용하여 A−B를 수행하였을 때 최상위비트에서 캐리(Carry)가 발생하였다면 B−A를 수행하면 최상위비트에서 캐리가 발생하지 않는다.

6−5=1을 예로 들어 확인해보자. 2의 보수 표현일 경우, A−B는 B의 2의 보수를 구해 A와 더하면 된다.

A=6=110

B=5=101

B의 2의 보수 = 011

```
  1 1 0
+ 0 1 1
-------
1 0 0 1
↑
자리올림수
```

자리올림수를 버리면 되므로 결과는 1이다.

이제 B−A, 즉 5−6=−1을 해보자. 마찬가지로 B−A는 A의 2의 보수를 구해 B와 더하면 된다.

B=5=101

A=6=110

A의 2의 보수 = 010

```
  1 0 1
+ 0 1 0
-------
  1 1 1
↑
자리올림수가 발생하지 않음
```

자리올림이 발생하지 않았다. 111을 10진수로 변환하기 위해 다시 2의 보수를 취하면 001, 그리고 111의 첫 번째 비트가 1이므로 −를 붙이면 결과는 −1이다.

이상을 정리하면 다음과 같다.

① A(6)−B(5)가 캐리가 발생했을 때 A가 B보다 큰 수이다.

② B(5)−A(6)를 수행하면 최상위비트에서 캐리가 발생하지 않는다.

③ A(6)+B(5)를 수행하면 최상위비트에서 캐리가 발생한다.

```
  6 = 1 1 0
+ 5 = 1 0 1
-----------
  1 0 1 1
↑
자리올림수
```

④ A(6)−B(5)의 결과에 캐리를 제거하고 1을 더해 올바른 결과가 나올 때는 1의 보수를 이용할 때이다.

A=6=110, B=5=101, B의 1의 보수 = 010

```
  1 1 0
+ 0 1 0
-------
1 0 0 0
+     1
-------
      1
```

14. Section 046

정수는 고정 소수점 방식으로 표현하며, 각 표현법에 따른 정수 범위는 다음과 같다.

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

※ 부호화 2의 보수법이 다른 표현 방식에 비해 1을 더 표현할 수 있다.

15. Section 046

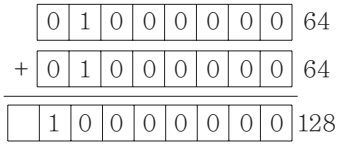
- 2의 보수를 사용하고, 크기가 1바이트면 수치가 −128~+127의 범위를 벗어나면 오버플로이다.
- 오버플로는 덧셈에 참여하는 두 수가 모두 양수이거나 음수일 때만 발생한다.
- 1의 보수나 2의 보수법에서는 부호를 포함하여 더하며, 부호가 반전되면 오버플로이다.
- 2의 보수법의 덧셈에서는 캐리가 발생하면 버린다.

① 양수끼리 더할 때 MSB에서 자리올림이 발생해야 오버플로이다.

예 64+63=127 : 오버플로 아님

※ MSB에서 자리올림이 발생하지 않아 부호의 변동이 없다. 즉 오버플로가 아니다.

64+64=128 : 오버플로



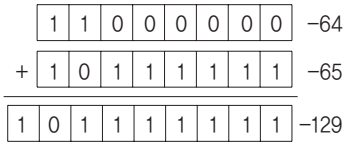
※ MSB에서 자리올림이 발생하여 부호가 반전되었다. 즉 오버플로가 발생했다.

② 음수끼리 더할 때 MSB에서 자리올림이 발생하지 않으면 Overflow가 일어난다.

• −64 + −64 = −128 : 오버플로 아님

※ MSB에서 자리올림이 발생하여 carry bit가 생겼지만 부호는 반전되지 않았다. 즉 오버플로가 발생하지 않았다.

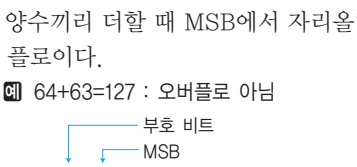
• −64 + −65 = −129 : 오버플로



※ MSB에서 자리올림이 발생하지 않았지만 carry bit가 생기고 부호가 반전되었다. 즉 오버플로가 발생했다.

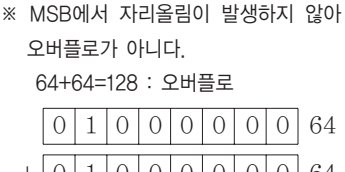
③ 부호 bit로 들어온 자리올림이 carry bit로 나가지 못하면 Overflow가 일어난다.

①번의 경우이다. 양수일 경우 MSB에서 자리올림이 발생하면 부호 비트인 0과 더해지므로 1이 되어 부호는 반전되며 carry bit로 나가지 못한다.



④ 부호 bit로 들어온 자리올림이 없는데 carry가 발생하면 Overflow가 일어난다.

②번의 경우이다. 음수의 경우 MSB에서 자리올림이 발생하지 않으면 부호 비트끼리 더하여 carry가 발생하는 데, 이런 경우는 오버플로일 경우에 그렇다.



16. Section 046

구분	부호화 절대치법 (Signed Magnitude)	부호화 1의 보수법 (Signed 1's Complement)	부호화 2의 보수법 (Signed 2's Complement)
표현 방법	양수 표현에 대하여 부호 비트의 값만 0을 1로 바꾼다.	양수에 대하여 1의 보수를 취한다.	양수에 대하여 2의 보수를 취한다.
비고	두 가지 형태의 0이 존재(+0, −0)		한 가지 형태의 0만 존재(+0)
	부호 비트 : 1 = 음수, 0 = 양수		

17. Section 046

2의 보수법에서는 덧셈 연산 시 발생하는 자리올림수를 무시하므로 다른 방법에 비해 처리가 간단하다.

18. Section 046

부호화 절대치 표현 방식에서 맨 앞의 비트로 부호를 판별하고 나머지 비트는 10진수로 변경하면 된다.

1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0

− 32 + 16 = −48

19. Section 046

- 2의 보수로 표현된 수는 부호 비트가 1인 수가 음수이므로 일단 음수만 계산해 본다.
- 음수로 표현된 수는 양수로 변환해야 그 크기의 절대값을 알 수 있다. 2의 보수로 표현된 음수는 다시 2의 보수를 구하면 된다.

1 0 0 1

↓

0 1 1 1

2^3 2^2 2^1 2^0

∴ 0 + 4 + 2 + 1 = 절대 크기는 7, 즉 −7이다.

1 1 1 1

↓

0 0 0 1

2^3 2^2 2^1 2^0

∴ 0 + 0 + 0 + 1 = 절대 크기는 1, 즉 −1이다.

※ 2의 보수로 표현된 수는 데이터 비트들 중에서 앞부분에 0이 많을수록 작은 값이 된다는 걸 알 수 있다.

20. Section 046

❶ 먼저 +15를 보기의 비트에 맞게 2진수로 표현한다.

0000 0000 0000 1111

❷ 1의 보수를 취한다.

1111 1111 1111 0000

21. Section 047

❶ 2진수로 변환한다.

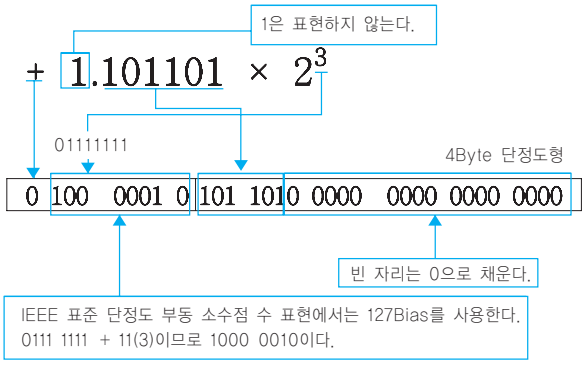
$(13.625)_{10} \rightarrow (1101.101)_2$

❷ 소수점의 위치를 조정하여 지수부와 가수부를 분리한다.

$1101.101 \rightarrow 1.101101 \times 2^3$

IEEE 표준에서는 정규화 시 가수부를 $1.XXXX\cdots \times 2^x$으로 표현한다. 그리고 1.XXXX…에서 1은 항상 1로 고정되기 때문에 표현하지 않는다.

❸ 정규화된 수치를 부동 소수점 방식에 맞게 표현한다.



22. Section 046

수치가 작아 4Byte로 표현하나 8Bit로 표현하나 그 형태는 동일하므로 1과 −1을 8Bit로 표현하면 0000 0001과 1111 1111이 된다. 두 수를 다음과 같이 더하면 합은 0이 되고 Carry가 발생한다. 2의 보수법을 이용하여 덧셈을 할 때 최상위 비트에서 발생하는 캐리는 무시한다.

1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 1

+ 1 0 0 0 0 0 0 0 0

23. Section 046

정수 곱셈 과정에서는 지수의 자릿수를 맞추는 정규화(Normalize) 과정은 필요하지 않다.

24. Section 046

2의 보수법의 연산 속도는 1의 보수에 비해 최소한 같거나 빠르다. 그리고 연산 결과를 비교하는 것은 비교기를 이용하기 때문에 같다고 보아야 한다.

25. Section 046

고정 소수점(Fixed Point)으로 덧셈이나 뺄셈을 할 때는 제일 먼저 두 수의 부호를 판단한다.

26. Section 046

감산 시 2의 보수법이 많이 쓰이는 이유는 보수를 취하기 쉬워서가 아니라 맨 끝단의 캐리를 무시하면 되기 때문에 최대 한 번만 더하면 되기 때문이다.

27. Section 046

뺄셈은 큰 수에서 작은 수를 빼기 때문에 A−B〉0인 경우는 A〉B이다. 뺄셈에서 오버플로가 발생하려면 두 수를 더해야 하는 경우다. 즉 A−(−B)는 A+B이므로 A−B에서 오버플로가 발생했다면 A는 양수, B는 음수이다.

28. Section 046

존 형식은 연산이 불가능하며, 입 · 출력에만 이용할 수 있다.

29. Section 047

Underflow는 더 이상 뺄 자료가 없어서 발생하는 것으로 소수 이하의 자리를 표현할 때 발생한다. 지수 부분의 bias가 64일 때 표현할 수 있는 이진수의 표현 범위는 2^{63}~2^{-64}이다. 그러므로 2^{-65}를 표현할 때 Underflow가 발생한다.

Overflow와 Underflow가 발생하는 이유

bias가 64라는 것은 100 0000을 기본으로 해서 소수 이상은 100 0000에 지수 승을 더하고, 소수 이하는 100 0000에서 지수승에 대한 숫자를 빼는 것을 말한다.

예

2^5 → 100 0000 + 101 → 100 0101

2^{63} → 100 0000 + 11 1111 → 111 1111

2^{64} → 100 0000 + 100 0000 → 1000 0000 자릿수가 늘어나므로 표현 불가 Overflow

2^{-5} → 100 0000 − 101 → 011 1011

2^{-64} → 100 0000 − 100 0000 → 000 00000

2^{-65} → 100 0000 − 100 1001 → 뺄 수 없으므로 Underflow 발생

30. Section 047

이 문제는 표현법에 관계없이 $(1.110\times10^{10}) \times (9.200\times10^{-5})$를 계산한 다음 유효자리에는 4자리, 지수에는 2자리에 맞춰 답을 구해야 한다. 즉 11,100,000,000 × 0.000092 = 1,021,200 = 1.0212×10^6 이지만 유효자리가 4자리라고 했으니 1.021×10^6이 된다.

32. Section 048

8Bit 중에서 실제 자료 비트는 7Bit이다.

∴송신 효율은 $\frac{7}{8} = 0.875$

33. Section 048

※ EBCDIC 코드는 확장된 BCD(8421) 코드이다.

EBCDIC(Extended BCD Interchange Code, 확장 2진화 10진 코드)

- 8Bit 코드로 IBM에서 개발하였다.
- 1개의 문자를 4개의 Zone 비트와 4개의 Digit 비트로 표현한다.
- 2^8=256가지의 문자를 표현할 수 있다.
- 1Bit의 Parity Bit를 추가하여 9Bit로 사용한다.
- 대형 기종의 컴퓨터에서 사용한다.

34. Section 048

7Bit 또는 8Bit로 한 문자를 표시하기도 하며 통신의 시작과 종료 등의 제어 조작에 편리한 코드는 아스키 코드이다.

ASCII 코드(American Standard Code for Information Interchange)

- 7Bit 코드로 미국 표준협회에서 개발하였다.
- 1개의 문자를 3개의 Zone 비트와 4개의 Digit 비트로 표현한다.
- 2^7=128가지의 문자를 표현할 수 있다.
- 1Bit의 Parity Bit를 추가하여 8Bit로 사용한다.
- 통신 제어용 및 마이크로컴퓨터에서 사용한다.

35. Section 049

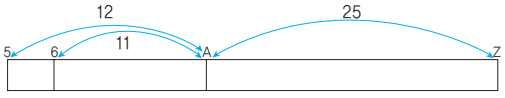
한글 2바이트 조합형 코드에서 한글과 영문을 구분하기 위해서 1비트를 사용한다.

36. Section 049

BCD 코드는 10진수 1자리를 2진수 4자리로 풀어서 사용하는 코드로서 10진수 입 · 출력이 간편하다.

37. Section 049

ASCII 코드 값으로 "A"는 65, "5"는 53이므로 두 값의 차이는 12이다. ASCII 코드 값으로 "Z"는 90, "6"은 54이므로 두 값의 차이는 36이다. 그러나 이 문제는 다음과 같이 그림을 그려보면 간단하게 풀 수 있다.



"5"와 "A"의 차이가 12이므로 "6"과 "A"의 차이는 11이다.

"A"와"Z"의 차이는 25이므로 "6"과 "Z"의 차이는 36이다.

38. Section 049

Gray 코드

- BCD 코드의 인접하는 비트를 X-OR 연산하여 만든 코드이다.
- 입 · 출력장치, A/D 변환기, 주변장치 등에서 숫자를 표현할 때 사용한다.
- 1Bit만 변화시켜 다음 수치로 증가시키기 때문에 하드웨어적인 오류가 적다.

39. Section 049

Excess-3 Code는 자기 보수 코드라서 다른 코드에 비해 보수를 구하기 편리하므로 연산에 유리하다.

40. Section 047

① IEEE 754는 부동 소수점 표현에 대한 국제 표준이다.

② 가수가 M이고, 지수 E라면 $1.M \times 2^E$ 형태를 취한다.

④ 64비트 복수 정밀도 형식의 경우 지수는 11비트이다.

41. Section 049

BCD 코드는 8421 코드라고도 하며, 각 자리값을 8421로 표시하면 1001이다.

- Excess-3 : 9 + 3 → 12 → 1100
- BCD 코드는 8421과 같다.
- 2421 Code : 1111 → 1×2 + 1×4 + 1×2 + 1×1 = 9

42. Section 049

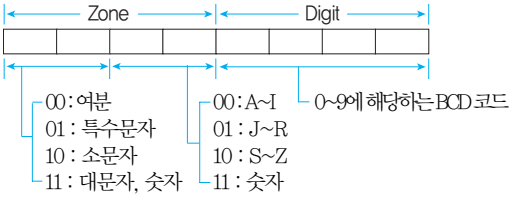
에러 검출 코드 : 해밍 코드, 비퀴너리 코드, 2 out-of 5 code(5 중 2 코드), 3 out-of 5 code(5 중 3 코드), 패리티 비트, 링카운트 코드

43. Section 049

해밍 코드에서 패리티 비트의 위치는 1, 2, 4, 8, … 2^n이다. 즉 7Bit라면 패리티 체크 비트의 위치는 C_1, C_2, 8, C_3, 4, 2, 1이다.

44. Section 049

EBCDIC 코드



예 a : 10 00 0001, b : 10 00 0010
A : 11 00 0001, B : 11 00 0010
1 : 11 11 0001, 5 : 11 11 0101

45. Section 047

FLOPS는 컴퓨터의 연산 속도를 나타내는 단위로 FLoating-point Operations Per Second의 약자이다. 즉 1초당 부동 소수점 연산 명령을 몇 번 실행할 수 있는지를 말한다. MFLOPS에서 M은 Mega를 말하는 것으로 1초에 부동 소수점 연산을 백만 번 수행함을 의미한다.

※ Mega = $2^{20} \fallingdotseq 10^6$, Giga = $2^{30} \fallingdotseq 10^9$

46. Section 046

팩 형식의 10진 표현은 부호가 표현되는 마지막 바이트의 디지트 부분(4비트)을 제외한 부분에는 4비트씩 끊어서 10진수 1자리가 표현된다. 그리고 마지막 바이트의 디지트 부분에 표현된 값이 C면 양수이고, D면 음수를 나타낸다.

-426은 426D이고 이것을 BCD로 표현하면

4 2 6 D

0100 0010 0110 1101이다.

47. Section 045

컴퓨터가 2의 보수로 표현된 수에 대해 뺄셈 연산을 할 때는 B의 보수를 구해 A에 더한 다음 자리올림수를 버리는 과정을 거친다. 그러나 이 문제는 과정을 요구하는 것이 아니라 결과만 구하면 되는 문제이므로 그냥 16진수 뺄셈을 하면 된다.

```
  F F F F   F F 6 1
- 0 0 0 0   0 0 4 F
-------------------
```

↓

```
  F F F F   F F 5 17
- 0 0 0 0   0 0 4 15
-------------------
  F F F F   F F 1 2
```

※ F는 10진수로 15이고, 16진수에서 윗자리를 빌려오면 16이 더해지므로 17-15=2가 된다.

48. Section 045

컴퓨터는 덧셈 연산을 기본으로 하여 사칙연산을 수행하므로 덧셈이 가장 빠르다.

- **뺄셈** : 감수에 대한 보수를 구한 다음 덧셈 연산을 한다. 5-2는 2에 대한 보수를 구한 다음 5와 더한다.
- **곱셈** : 덧셈 연산의 반복이다. 5×3은 5를 세 번 더한다.
- **나눗셈** : 뺄셈 연산의 반복이다. 15/5는 15에서 5를 세 번 뺀다.

49. Section 046

2의 보수법에서 부호를 제외하고 5비트로 표현할 수 있는 수의 범위는 31 ~ -32이다. 다음은 부호를 제외하고 덧셈한 결과다. 결과가 31 ~ -32을 넘으면 오버플로다.

① 10010=18
+ 00111=7
11001=25, 31보다 크지 않으므로 오버플로가 아니다.

② 10010=18
+ 01111=15
100001=33, 31보다 크므로 오버플로다.

③번은 부호 비트가 1이므로 음수이다.

③ 10010=-14
+ 11001= -7
101011=-21, -32보다 작지 않으므로 오버플로가 아니다.

④ 10010=18
+ 01011=11
11101=29, 31보다 크지 않으므로 오버플로가 아니다.

3장 정답 및 해설 — 프로세서

1.④	2.③	3.④	4.①	5.①	6.④	7.④	8.③	9.③	10.②	11.④	12.②	13.③	14.①	15.①
16.②	17.④	18.②	19.④	20.②	21.①	22.①	23.②	24.②	25.④	26.④	27.①	28.②	29.③	30.②
31.①	32.③	33.④	34.①	35.①	36.③	37.④	38.②	39.④	40.④	41.②	42.④	43.①	44.①	45.①
46.④	47.③	48.③	49.④	50.④	51.④	52.④	53.①	54.②	55.①	56.②	57.②	58.①	59.②	60.③
61.③	62.①	63.③	64.①	65.③										

1. Section 050

- 중앙처리장치의 기능은 제어, 연산, 기억, 전달 기능이다.
- 입력 기능은 주변장치인 입력장치의 기능이다.

2. Section 050

DMA는 제어장치의 제어를 받지 않고 자율적인 동작으로 메모리에 접근하여 입 · 출력을 수행하는 입 · 출력 제어기이다.

3. Section 050

연산의 중심이 되는 레지스터는 누산기 레지스터이다.

- **인덱스 레지스터** : 주소의 변경이나 프로그램에서의 반복 연산의 횟수를 세는 레지스터
- **데이터 레지스터** : 연산에 사용될 데이터를 기억하는 레지스터
- **명령 레지스터** : 현재 실행중인 명령의 내용을 기억하는 레지스터

4. Section 050

두 배 길이 레지스터는 주어진 컴퓨터 시스템에서 데이터 또는 저장장치의 한 단위 길이가 정상적인 것의 두 배인 것으로 시프트 레지스터가 해당된다.

- **시프트 레지스터** : 저장된 값을 왼쪽 또는 오른쪽으로 1Bit씩 자리 이동시키는 역할을 하는 것으로, 자료의 병렬 전송을 직렬 전송으로 변환하는 데 적합함
- **어드레스 레지스터(MAR)** : 기억장치를 출입하는 데이터의 번지를 기억하는 레지스터
- **AC 레지스터(누산기)** : 연산된 결과를 일시적으로 저장하는 레지스터로, 연산의 중심
- **버퍼 레지스터(MBR)** : 기억장치를 출입하는 데이터가 잠시 기억되는 레지스터

5. Section 050

버스 경합이란 여러 개의 자원이 하나의 버스를 점유하기 위해 벌이는 경쟁 현상으로 버스 경합을 줄이기 위해서는 버스의 고속화, 캐시 메모리의 사용, 다중 버스 사용 등의 방법이 사용된다.

6. Section 050

프로그램 카운터는 다음에 실행할 명령이 들어 있는 곳의 주소를 가지고 있는 레지스터이다.

- ② : MBR의 역할이다.
- ③ : IR의 역할이다.

7. Section 050

명령 코드가 명령을 수행할 수 있게 필요한 제어 기능을 제공해 주는 것은 CPU에 있는 제어장치이다.

- **레지스터** : CPU 내부에서 처리할 명령어나 연산의 중간 결과값 등을 일시적으로 기억하는 임시 기억장소
- **누산기** : 연산된 결과를 일시적으로 저장하는 레지스터로, 연산의 중심
- **스택** : 자료의 삽입 · 삭제 작업이 한쪽 방향에서만 가능할 수 있

도록 할당한 메모리의 일부로, 0 주소 명령의 데이터 저장소, 인터럽트의 복귀 주소 저장 등에 사용함

8. Section 052

1의 보수일 경우 왼쪽으로 Shift하면 양수인 경우 패딩 비트(빈 공간에 채워지는 비트)로 0이 들어오지만 음수인 경우 1, 즉 Sign Bit가 들어온다.

산술 Shift

산술 Shift는 부호(Sign)를 고려하여 자리를 이동시키는 연산으로, 2^n으로 곱하거나 나눌 때 사용한다.

- 왼쪽으로 n Bit Shift하면 원래 자료에 2^n을 곱한 값과 같다.
- 오른쪽으로 n Bit Shift하면 원래 자료를 2^n으로 나눈 값과 같다.
- 홀수를 오른쪽으로 한 번 Shift하면 0.5의 오차가 발생한다.
- 산술 Shift는 정수 표현 방식에서만 가능한 방법으로, 정수의 수치 표현 방법에 따라서 표현이 조금씩 다르다.

<table>
<tr><th>Shift</th><th>수치 표현법</th><th>-43</th><th>+43</th></tr>
<tr><td rowspan="3">Shift Left</td><td>부호화 절대치</td><td>• Padding Bit : 0
10101011 → 11010110
-43×2¹, 즉 -86이 된다.</td><td rowspan="3">양수는 모두 같다.
• Padding Bit : 0
00101011 → 01010110
43×2¹, 즉 86이 된다.</td></tr>
<tr><td>1의 보수법</td><td>• Padding Bit : 1
11010100 → 10101001
-43×2¹, 즉 -86이 된다.</td></tr>
<tr><td>2의 보수법</td><td>• Padding Bit : 0
11010101 → 10101010
-43×2¹, 즉 -86이 된다.</td></tr>
<tr><td rowspan="3">Shift Right</td><td>부호화 절대치</td><td>• Padding Bit : 0
• 오차 발생 : 0.5 증가
10101011 → 10010101
-43÷2¹ → -21.5 → -21</td><td rowspan="3">양수는 모두 같다.
• Padding Bit : 0
00101011 →
00010101 → 21
43÷2¹, 즉 21.5가 되어야 하지만 오차가 발생하여 0.5가 감소한다.</td></tr>
<tr><td>1의 보수법</td><td>• Padding Bit : 1
• 오차 발생 : 0.5 증가
11010100 → 11101010
-43÷2¹ → -21.5 → -21</td></tr>
<tr><td>2의 보수법</td><td>• Padding Bit : 1
• 오차 발생 : 0.5 감소
11010101 → 11101010
-43÷2¹ → -21.5 → -22</td></tr>
</table>

9. Section 050

실행될 명령어는 IR(명령 레지스터)에 기억된다.

10. Section 050

부호기(Encoder)

명령어 해독기가 명령을 번역하여 부호기에 전달하면, 부호기는 그 명령을 수행할 각 장치로 제어 신호를 발생시켜 동작시킨다.

11. Section 050

중앙처리장치에서 사용하고 있는 버스(BUS)는 제어 버스(Control Bus), 주소 버스(Address Bus), 데이터 버스(Data Bus) 세 가지이다.

12. Section 053

2주소 형식에서는 계산된 결과가 레지스터(CPU)에 저장되고, 그 레지스터의 값이 주기억장치에 저장되기 때문에 두 곳에 모두 결과가 남는다.

13. Section 050

CPU의 네 가지 기능 중 기억 기능을 수행하는 것이 레지스터인데, 레지스터의 종류는 다음과 같다.

- **프로그래머가 기억된 내용을 직접 프로그램 할 수 있는 것** : 연산용 레지스터, 인덱스 레지스터
- **프로그래머가 직접 변경할 수 없는 것** : 명령어 레지스터(IR), 프로그램 카운터(PC)
- **기억장치와의 자료 교환용** : MAR, MBR

15. Section 050

범용 레지스터가 많으면 자료를 얻기 위해 처리 속도가 느린 메모리에 접근하는 횟수를 줄일 수 있으므로 실행 속도가 빨라진다.

16. Section 051

컴퓨터 기종에 따라 인스트럭션의 형태와 주소지정방식이 다르다.

17. Section 051

모드 필드는 직접 번지와 간접 번지를 나타내는 비트를 포함하고 있으므로 모드 필드에 따라서 유효 번지가 결정된다.

18. Section 051

다음에 실행할 명령의 위치는 프로그램 카운터가 가지고 있기 때문에 Instruction Code에 반드시 다음 Instruction의 위치를 알리는 번지가 있을 필요는 없다.

19. Section 051

오퍼랜드로 지정될 수 있는 곳은 연산에 사용할 자료를 저장하고 있는 위치이다. 제어장치에는 사용할 자료를 저장하고 있지 않다.

20. Section 051

연산자의 수는 OP-Code(연산자 부)의 길이와 관계가 있다. OP-Code가 n 비트이면 사용할 수 있는 연산자의 종류는 2^n개이다.

- 첫 번째 명령의 OP-Code가 3비트이므로 $2^3 = 8$개의 연산자를 사용할 수 있다.
- 두 번째 명령의 OP-Code가 6비트이므로 $2^6 = 64$개의 연산자를 사용할 수 있다.

21. Section 051

인스트럭션 세트의 효율성을 높이기 위하여 고려할 사항에는 기억 공간, 사용 빈도, 주기억장치 밴드 폭 이용 등이 있다.

22. Section 051

논리 연산은 비수치적인 연산을 말한다.

- **논리 연산** : MOVE, NOT, AND, OR, 논리 SHIFT, ROTATE, COMPLEMENT, EXCLUSIVE OR 등
- **산술 연산** : ADD, SUBTRACT, MULTIPLY, DIVIDE, 산술 SHIFT 등

23. Section 052

왼쪽으로 2Bit 이동시키는 기능은 원래의 R1 값에 2^2을 곱하는 결과가 된다.

24. Section 051

오피코드	모드비트	레지스터 지정	기억장소주소
32개 명령	직접, 간접	4개	4K

명령어의 크기는 위의 요소들을 모두 지정할 수 있는 크기면 된다.

- **OP-Code** : 32개의 명령이므로 5비트($2^5 = 32$)
- **모드비트** : 직접/간접만 구분하면 되므로 1비트($2^1 = 2$)
- **레지지터 지정** : 4개 이므로 2비트($2^2 = 4$)
- **기억장소 주소** : 4K이므로 12비트($4K = 4 \times 1024 = 4096 = 2^{12}$)

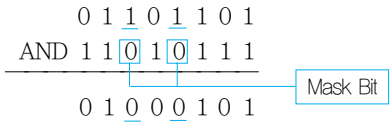
∴ 명령어 크기는 5+1+2+12 = 20

25. Section 058

AND(Masking Operation)

- AND 연산은 특정 문자 또는 특정 비트를 삭제(Clear)시키는 연산으로, Masking 연산이라고도 한다.
- AND 연산은 삭제할 부분의 비트를 0과 AND시켜서 삭제하는데, 대응시키는 0인 비트를 Mask Bit라 한다.

예 01101101에서 3번, 5번 비트 값을 Clear 시키는 경우



26. Section 050

제어장치가 명령을 해독하려면 주기억장치에 있는 명령어를 레지스터로 가져와야 한다. 가상 메모리에 있는 내용이라면 먼저 주기억장치로 가져온 다음 레지스터로 가져와야 한다.

제어장치(Control Unit)

- 제어장치(Control Unit)는 컴퓨터에 있는 모든 장치들의 동작을 지시하고 제어하는 장치이다.
- 제어장치는 주기억장치에서 읽어 들인 명령어를 해독하여 해당하는 장치에게 제어신호를 보내 정확하게 수행하도록 지시한다.

제어장치의 구성 요소

- **프로그램 카운터(Program Counter)** : 다음 번에 실행할 명령어의 번지를 기억하는 레지스터
- **명령 레지스터(Instruction Register)** : 현재 실행중인 명령의 내용을 기억하는 레지스터
- **명령 해독기(Decoder)** : 명령 레지스터에 있는 명령어를 해독하는 회로
- **번지 해독기** : 명령 레지스터에 있는 명령어가 가지고 있는 번지(직접, 간접 번지 등)를 해독하는 회로
- **부호기(Encoder)** : 해독된 명령에 따라 각 장치로 보낼 제어 신호를 생성하는 회로

27. Section 051

- 명령어의 길이 = OP 코드의 길이 + 주소지정방식 모드 비트 + 오퍼랜드의 길이
- 16 = 5 + 2 + '오퍼랜드의 길이' 이므로, 주소의 범위를 나타내는 오퍼랜드의 길이는 9Bit이다.

- 오퍼랜드의 길이가 n Bit이면 2^n개의 번지를 지정할 수 있으므로 2^9(512)개의 번지, 즉 0~511번까지의 범위를 번지로 지정할 수 있다.

28. Section 051

- 주기억장치의 용량 32K = $2^5 \times 2^{10} = 2^{15}$이므로 번지 비트수는 15비트이다.
- 워드 크기 24 = 오퍼레이션 비트 + 간접 비트(1) + 레지스터 선정 비트(2) + 주기억장소 선정 비트(15) 이므로 오퍼레이션 비트는 6(24-18)비트이다.
- 오퍼레이션 수는 2^6=64개이다. 그리고 주소부의 길이(15)는 MAR, PC와 같으며, MBR은 워드 길이(24)와 같다.

29. Section 051

NOP 명령은 아무런 동작을 하지 않고 Clock Pulse만 낭비하기 때문에 실행 속도를 느리게 한다.

30. Section 051

주기억장치의 각 기억장소를 구성할 때 기억장소의 주소를 Byte마다 부여하여 필요한 Byte 수만큼 조합하여 가변적으로 기억장소를 구성하느냐, Word 단위마다 주소를 부여하여 구성하느냐에 따라 전자를 '바이트 머신', 후자를 '워드 머신'이라 한다.

31. Section 052

10진 연산은 레지스터를 사용하지 않고 연산을 수행한다.

32. Section 052

연산 우선 순위

① 산술 연산자(∧(거듭제곱) → ×(곱셈), ÷(나눗셈) → +, −)
② 관계 연산(=, ≠, >, <, ≧, ≦)
③ 논리 연산자(NOT → AND → OR)
④ 동일 순위일 때는 왼쪽의 연산을 먼저 처리한다.

33. Section 051

산술 연산의 기본 연산자

- 슈퍼 컴퓨터를 제외한 일반적인 컴퓨터는 뺄셈기, 곱셈기, 나눗셈기가 별도로 없다. 그래서 덧셈기로만 산술 연산을 수행하는데, 모든 산술 연산은 기본 연산자를 이용하여 수행된다.
- **기본 연산자의 종류** : 덧셈, Shift, 보수
- **덧셈** : 덧셈
- **뺄셈** : 덧셈, 보수
- **곱셈** : 덧셈, Shift
- **나눗셈** : 덧셈, 보수, Shift

34. Section 054

암시적(묵시적) 주소지정방식(Implied Mode)

- 명령 실행에 필요한 데이터의 위치를 지정하지 않고 누산기나 스택의 데이터를 묵시적으로 지정하여 사용한다.
- 오퍼랜드가 없는 명령이나 PUSH R1처럼 오퍼랜드가 1개인 명령어 형식에 사용된다.

  예 SHL : 누산기의 내용을 좌측으로 1Bit 이동한다.
  PUSH R1 : R1의 내용을 스택의 최상위에 저장한다.

35. Section 052

대부분의 명령이 주기억장치에서 자료를 꺼내오고 계산된 결과를 주기억장치에 저장하기 때문에 주기억장치와의 자료 전달 명령이 가장 많다.

36. Section 052

과학적인 응용 및 상업적인 응용은 비수치적인 논리 연산보다 수치적인 자료의 산술 연산이 사용되는 분야이다.

37. Section 050

버스

- 버스는 CPU, 메모리, I/O 장치 등과 상호 필요한 정보를 교환하기 위해 연결하는 공동의 전송이다.
- 컴퓨터 내부 회로에서 버스 선(Bus Lines)을 사용하는 목적은 결선의 수를 줄이기 위해서이다.

38. Section 053

3주소 명령어

- 3주소 명령어는 Operand부가 3개로 구성되는 명령어 형식으로 여러 개의 범용 레지스터(GPR)를 가진 컴퓨터에서 사용한다.
- 연산의 결과는 Operand 1에 기록된다.
- 장점
  - 연산 시 원래의 자료를 파괴하지 않는다.
  - 다른 형식의 명령어를 이용하는 것보다 프로그램 전체의 길이를 짧게 할 수 있다.
  - 전체 프로그램 실행 시 명령 인출을 위하여 주기억장치를 접근하는 횟수가 줄어든다.

• 단점
  – 명령어 한 개의 길이가 너무 길어진다.
  – 하나의 명령을 수행하기 위해 최소한 네 번 기억장소에 접근해야 하므로 수행시간이 길어진다.

39. Section 052

부 프로그램과 매크로(Macro)의 공통점은 여러 번 중복되는 부분을 별도로 작성하여 사용하는 것이다.

- **부 프로그램** : 반복적으로 사용되는 코드를 별도의 프로그램으로 작성하여 필요할 때 호출하여 사용할 수 있도록 제작한 프로그램
- **매크로** : 반복적으로 사용되는 코드를 프로그램 내에 삽입하여 해당 프로그램의 어디서나 호출하여 사용할 수 있게 만들어 놓은 프로그램 코드 모임

40. Section 052

스왑(Swap) 또는 스와핑(Swapping)은 메모리 관리 기법의 하나로서, 프로그램 디버깅과는 무관하다.

41. Section 052

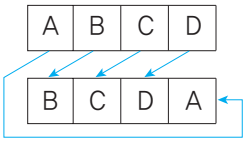
이터레이션은 해당 루틴이 정확한 결과를 산출할 때까지 해당 루틴에서 발생한 결과를 가지고 처음에 사용한 자료를 다시 수정하여 계산 작업을 반복적으로 수행하는 것을 말한다.

43. Section 054

상대 주소는 명령어 자신의 주소 부분과 프로그램 카운터의 내용의 더해서 계산하는 방식이다. 프로그램 카운터는 다음에 실행할 명령의 번지를 가지고 있는 것으로, JUMP 같은 분기 명령을 수행할 때는 프로그램 카운터의 값만 변경하면 된다. 이처럼 상대 주소 방식은 분기 명령을 간단하게 처리할 수 있으므로 분기 명령어와 자주 쓰인다.

44. Section 052

Rotate



45. Section 052

한 비트 오른쪽으로 산술 시프트하면 2로 나눈 것과 같고, 왼쪽으로 시프트하면 2로 곱한 것과 같으므로 산술적 Shift는 곱셈과 나눗셈을 위한 용도로 사용된다.

46. Section 053

0 주소 형식 명령어는 주소의 사용 없이 스택에 연산자와 피연산자를 넣었다 꺼내어 연산한 후 결과를 다시 스택에 넣으면서 연산하기 때문에 원래의 자료가 남지 않는다.

47. Section 053

지정할 기억장소가 고정되어 있는 경우 굳이 주소를 지정할 필요가 없다. 만약 지정하면 명령어 길이가 길어지고 대역폭도 비효율적이게 된다.

48. Section 053

피연산자를 기억시킬 레지스터의 종류에 따라 컴퓨터 구조를 분류하면 GPR(범용 레지스터) 컴퓨터 구조, ACC(누산기) 컴퓨터 구조, 스택(Stack) 컴퓨터 구조가 있다.

- 3주소 명령어 : GPR(범용 레지스터)
- 2주소 명령어 : GPR(범용 레지스터)
- 1주소 명령어 : ACC(누산기)
- 0주소 명령어 : 스택(Stack)

49. Section 053

스택 메모리가 사용되는 경우

- 인터럽트가 받아들여졌을 때의 복귀 주소
- 0-주소지정방식에서의 자료 저장소
- 재귀 프로그램(Recursive Program)의 수행 순서
- Postfix로 표현된 수식 연산
- 부 프로그램 호출 시 복귀 주소

50. Section 051

연산자(OP Code)의 수행에 필요한 자료는 주기억장치에서 가져와야 한다. 마그네틱(자기) 디스크는 보조기억장치로서 연산자(OP Code)의 수행에 필요한 자료를 보관시켜 놓기에는 적절하지 않다.

51. Section 054

0-주소 명령어는 주소를 지정하는 Operand부 없이 OP-Code부만으로 구성되어 있으며, Stack의 Top 포인터가 가리키는 Operand를 자료로 사용한다. 그러므로 자료의 주소를 지정할 필요가 없는 것은 0-주소이다.

52. Section 054

Implied Mode

0번지 명령어에서 Stack의 SP가 가리키는 Operand를 암시하여 이용한다.

53. Section 054

즉치(즉시)적 주소지정방식(Immediate Mode)

- 즉치적 주소지정방식은 명령어 자체에 오퍼랜드(실제 데이터)를 내포하고 있는 방식이다.
- 별도의 기억장소를 액세스하지 않고 CPU에서 곧바로 자료를 이용할 수 있어서 실행 속도가 빠르다는 장점이 있다.
- 명령어의 길이에 영향을 받으므로 표현할 수 있는 데이터 값의 범위가 제한적이다.

직접 주소지정방식(Direct Mode)

- 직접 주소지정방식은 명령의 주소부(Operand)가 사용할 자료의 번지를 표현하고 있는 방식이다.
- 명령의 Operand부에 표현된 주소를 이용하여 실제 데이터가 기억된 기억장소에 직접 사상시킬 수 있다.
- 기억 용량이 2^n개의 Word인 메모리 시스템에서 주소를 표현하려면 n Bit의 Operand부가 필요하다.
- 직접 주소지정방식에서 명령의 주소부에 데이터를 가지고 있는 레지스터의 번호를 지정하면 레지스터 모드라고 한다.

54. Section 054

간접주소방식은 명령어 내에 주소지정방식을 나타내는 별도의 Mode Bit(I)를 둬야 한다.

55. Section 054

간접 주소지정방식(Indirect Mode)

- 간접 주소지정방식은 명령어에 나타낼 주소가 명령어 내에서 데이터를 지정하기 위해 할당된 비트(Operand부의 비트) 수로 나타낼 수 없을 때 사용하는 방식이다.
- 명령의 길이가 짧고 제한되어 있어도 용량이 큰 컴퓨터의 긴 주소에 접근이 가능한 방식이다.
- 명령어 내의 주소부에 실제 데이터가 저장된 장소의 번지를 가진 기억장소의 번지를 표현함으로써, 최소한 주기억장치를 두 번 이상 접근하여 데이터가 있는 기억장소에 도달한다.
- 간접 주소지정방식에서 명령의 주소부에 데이터의 주소를 가지고 있는 레지스터의 번호를 지정하면 레지스터 간접 모드라고 한다.

56. Section 050

제어 주소 레지스터(CAR)는 다음에 실행할 마이크로 명령어의 주소를 저장하는 레지스터로, Mapping의 결과값, 주소 필드, 서브루틴 레지스터의 내용이 적재되어 있다.

57. Section 054

상대주소(Relative Address)

주소 부분의 값이 어떤 기준주소와 실제 데이터가 기억되어 있는 주소의 변위를 표시한다.

58. Section 054

수행할 명령어 자신이 들어 있는 기억장소를 가지고 있는 레지스터는 PC이다.

59. Section 054

베이스 레지스터 주소지정방식

- 베이스 레지스터에 저장된 기준 번지를 이용하는 주소지정방식이다.
- 유효번지 = 베이스 레지스터에 저장된 기준 번지 + 변위 = 2048 + 1022 = 3070

60. Section 050

T 플립플롭은 현재 상태의 값을 반대로 만들기 때문에 외부 입력을 그대로 저장하기에 적당하지 않다.

61. Section 054

간접 주소지정방식(Indirect Mode)

- 간접 주소지정방식은 명령어에 나타낼 주소가 명령어 내에서 데이터를 지정하기 위해 할당된 비트(Operand부의 비트) 수로 나타낼 수 없을 때 사용하는 방식이다.
- 명령의 길이가 짧고 제한되어 있어도 용량이 큰 컴퓨터의 긴 주소에 접근이 가능한 방식이다.
- 명령어 내의 주소부에 실제 데이터가 저장된 장소의 번지를 가진 기억장소의 번지를 표현함으로써, 최소한 주기억장치를 두 번 이상 접근하여 데이터가 있는 기억장소에 도달한다.
- 간접 주소지정방식에서 명령의 주소부에 데이터의 주소를 가지고 있는 레지스터의 번호를 지정하면 레지스터 간접 모드라고 한다.

62. Section 053

2-주소 명령어 형식에서는 주소필드1에 연산결과가 저장되므로

최종적인 계산 결과는 R1에 저장된다. 즉 마지막 Micro Operation은 R1의 값을 Y에 옮기면 된다.

연산코드	주소필드1	주소필드2	해석
MOV	R1	A	A의 값을 R1에 저장한다. R1 ← A
ADD	R1	B	B의 값을 R1에 더한다. R1 ← R1+B
MOV	R2	C	C의 값을 R2에 저장한다. R2 ← C
ADD	R2	D	D의 값을 R2에 더한다. R2 ← R2 + D
MUL	R1	R2	R1과 R2를 곱한다. R1 ← R1 * R2
MOV	Y	R1	R1의 값을 Y에 저장한다. Y ← R1

63. Section 053

순환 프로그램은 순환하는 만큼 반복하여 실행하면 결과를 알 수 있다.

① n=5일 때 : R=5+R(4)
② n=4일 때 : R=4+R(3)
③ n=3일 때 : R=3+R(2)
④ n=2일 때 : R=2+R(1)
⑤ n=1일 때 : R=R(1)
⑤번의 결과로 R은 1을 가지고 ④번으로 간다.
④는 R=2+1이므로 3을 가지고 ③번으로 돌아간다.
③은 R=3+3이므로 6을 가지고 ②번으로 돌아간다.
②는 R=4+6이므로 10을 가지고 ①번으로 돌아간다.
①은 R=5+10이므로 결과 15를 반환한다.

64. Section 054

사상 함수

가상기억장치에 있는 프로그램이 주기억장치에 적재되어 실행될 때 논리적인 가상주소를 물리적인 실기억주소로 변환하는 것을 주소 사상 또는 주소 매핑(Mapping)이라고 하며, 이때 실기억주소를 계산하는 함수를 사상 함수(Mapping Function)라고 한다.

65. Section 054

베이스 레지스터(Base Register Mode)

프로그램의 실행을 위하여 보조기억장치에 보관중이던 프로그램이 주기억장치로 Load될 때는 주기억장치 내의 적절한 빈 영역을 선택하여 Load하므로 Load될 때마다 할당되는 영역이 달라질 수 있다. 따라서 프로그램 상에서 명령어의 주소를 표현할 때는 절대주소로 나타내지 않고, Load될 때 할당되는 영역의 시작주소를 기준으로 재조정하여 실제 기억장소의 주소를 사용할 수 있도록 변위값으로 표현한다. 이때 명령의 시작주소를 가지고 있는 레지스터가 베이스 레지스터이며, 베이스 레지스터의 값과 명령어에 포함된 변위값을 더해 접근하고자 하는 기억장소의 유효주소를 얻는 것을 재배치라 한다. 또한 이러한 재배치 기법에 의해 Operand가 있는 기억장소를 지정하는 방법을 Base Register Addressing Mode라 한다.

4장 정답 및 해설 — 명령 실행과 제어

1.③	2.④	3.④	4.②	5.①	6.③	7.④	8.①	9.②	10.①	11.④	12.③	13.②	14.③	15.④
16.①	17.②	18.④	19.②	20.④	21.③	22.④	23.①	24.②	25.③	26.①	27.③	28.②	29.②	30.②
31.②	32.④	33.④	34.③	35.①	36.②	37.①	38.④	39.①	40.③					

1. Section 056

메이저 스테이트에서 인출 단계는 명령어를 주기억장치에서 중앙처리장치의 명령 레지스터로 가져와 해독하는 단계다. 즉 읽어온 내용(MBR)이 IR(명령 레지스터)로 이동해야 한다. MBR+AC → AC는 실행 사이클의 상태다.

2. Section 055

마이크로 오퍼레이션

한 개의 명령(Instruction)을 실행하기 위해서는 그 명령의 위치를 파악하고, 그 곳을 찾아가 명령을 꺼내 와서 무슨 명령인지 번역하고, 또 그 명령 기능을 처리할 장치를 동작시키는 등 여러 동작 과정을 거치게 된다. 이때의 작은 동작 하나하나를 Micro

Operation이라 하는데, 이는 컴퓨터 기종별로 다르다.

3. Section 055

F는 처리기를 의미하는 것으로, F(R,R) → R 마이크로 오퍼레이션은 자료가 처리되어 다른 레지스터로 자료가 옮겨지는 마이크로 오퍼레이션을 나타낸다.

4. Section 055

타이밍 신호는 제어장치에 있는 순차 카운터와 디코더에 의해 발생한다. 만약 4Bit의 순차 카운터가 있다면, 이 카운터의 출력이 4 × 16 구조의 디코더로 입력되어 T_0~T_{15}까지 16개의 타이밍 신호를 생성한다.

5. Section 055

동기 고정식(Synchronous Fixed)

- 모든 마이크로 오퍼레이션의 동작시간을 같다고 가정하여 CPU Clock의 주기를 Micro Cycle Time과 같도록 정의하는 방식이다.
- 모든 마이크로 오퍼레이션의 수행 시간이 유사한 경우에 사용한다.
- 모든 마이크로 오퍼레이션 중에서 수행시간이 가장 긴 것을 Micro Cycle Time으로 정한다.
- **장점** : 제어기의 구현이 단순함
- **단점** : CPU의 시간 낭비가 심함

6. Section 055

인덱스 레지스터는 프로그램으로 레지스터의 내용을 변경할 수 있지만 연산 레지스터의 내용은 변경할 수 없다.

7. Section 056

메이저 스테이트의 정의

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

8. Section 056

메이저 스테이트

인출 단계 (Fetch Cycle)	• 명령어를 주기억장치에서 중앙처리장치의 명령 레지스터로 가져와 해독하는 단계이다. • 읽어와 해석된 명령어가 1 Cycle 명령이면 이를 수행한 후 다시 Fetch Cycle로 변천한다.
간접 단계 (Indirect Cycle)	• Fetch 단계에서 해석된 명령의 주소부가 간접주소인 경우 수행된다. • 이 사이클에서는 Fetch 단계에서 해석한 주소를 읽어온 후 그 주소가 간접주소이면 유효주소를 계산하기 위해 다시 Indirect 단계를 수행한다.
실행 단계 (Execute Cycle)	• Fetch 단계에서 인출하여 해석한 명령을 실행하는 단계이다. • 플래그 레지스터의 상태 변화를 검사하여 Interrupt 단계로 변천할 것인지를 판단한다. • Interrupt 요청 신호를 나타내는 플래그 레지스터의 변화가 없으면 Fetch 단계로 변천한다.
인터럽트 단계 (Interrupt Cycle)	• 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.

9. Section 056

페치 사이클에서는 Program Counter(PC), MAR, MBR이 사용된다.

10. Section 056

메이저 상태 중에서 명령어를 해독하여 인스트럭션의 종류에 대한 판단이 이루어지는 상태는 Fetch이다.

- Execute : Fetch 단계에서 인출하여 해석한 명령을 실행하는 단계
- Interrupt : 인터럽트 발생 시 복귀주소(PC)를 저장시키고, 제어순서를 인터럽트 처리 프로그램의 첫 번째 명령으로 옮기는 단계
- Indirect : Fetch 단계에서 해석된 명령의 주소부가 간접주소인 경우 수행됨

11. Section 056

Interrupt Major State는 인터럽트 발생 시에만 복귀주소를 저장시키고 제어 순서를 인터럽트 처리 프로그램의 첫 번째 명령으로 옮기는 단계로서 다른 메이저 스테이트에 비해 상대적으로 인스트럭션의 수행과 무관하다고 할 수 있다.

12. Section 055

서로 다른 레지스터인 R1, R2, R3 3개를 사용하는 3주소 명령이다.

13. Section 056

인터럽트 사이클은 인터럽트 발생 시 복귀주소(PC)를 저장시키고, 제어 순서를 인터럽트 처리 프로그램의 첫 번째 명령으로 옮기는 단계이다.

14. Section 056

분기 혹은 점프 명령문은 다음에 실행할 명령의 주소를 가지고 있는 PC의 값을 수정함으로써 이루어진다.
예 JMP 300 명령은 'PC ← 300'으로 처리된다.

15. Section 056

- 인출 과정에서 꺼낸 명령어에 오퍼랜드(실제 데이터)가 포함되어 있거나 직접주소이면 곧바로 실행 단계(Execute Cycle)로 변이한다.
- 메모리로부터 읽은 워드가 오퍼랜드의 주소일 경우 간접 사이클로 변이한다.

17. Section 056

Fetch Cycle의 동작 순서

제어 신호	Micro Operation	의미
C_{0t0}	MAR ← PC	PC에 있는 번지를 MAR에 전송시킨다.
C_{0t1}	MBR ← M[MAR], PC ← PC + 1	• 메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송한다. • 다음에 실행할 명령의 위치를 지정하기 위해 PC의 값을 1 증가시킨다.
C_{0t2}	IR ← MBR[OP], I ← MBR[I]	• 명령어의 OP-Code 부분을 명령 레지스터에 전송한다. • 명령어의 모드 비트를 플립플롭 I에 전송한다.
C_{0t3}	F ← 1 또는 R ← 1	I가 0이면 F 플립플롭에 1을 전송하여 Execute 단계로 변천하고, I가 1이면 R 플립플롭에 1을 전송하여 Indirect 단계로 변천한다.

18. Section 058

INTERRUPT State를 마친 다음에는 데이터 없이 항상 FETCH 단계로 변천한다.

제어 데이터

- 제어장치가 제어 신호를 발생하기 위한 자료로서, CPU가 특정한 메이저 상태와 타이밍 상태에 있을 때 제어 자료에 따른 제어 규칙에 의해 제어 신호가 발생한다.
- **제어 데이터는 종류**
  - 메이저 스테이트 사이의 변천을 제어하는 데이터
  - 중앙처리장치의 제어점을 제어하는 데이터
  - 인스트럭션의 수행 순서를 결정하는 데 필요한 제어 데이터

구분	Fetch	Indirect	Execute	Interrupt
State 간 변이용	명령어 종류 주소지정방식	주소지정방식	인터럽트 요청 신청	없음
제어점 제어용	명령어	유효주소	명령어의 연산자	Interrupt 체제에 따라 달라짐
수행 순서 제어용	PC	없음	PCI	nterrupt 체제에 따라 달라짐

19. Section 056

- ②번은 MBR ← PC, PC ← 0이어야 한다.
- **Interrupt Cycle의 동작 순서**

제어 신호	Micro Operation	의미
C_{3t0}	MBR[AD] ← PC, PC ← 0	• PC가 가지고 있는 다음에 실행할 명령의 주소를 MBR의 주소 부분으로 전송한다. • 복귀 주소를 저장할 0번지를 PC에 전송한다.
C_{3t1}	MAR ← PC, PC ← PC + 1	• PC가 가지고 있는 값 0번지를 MAR에 전송한다. • 인터럽트 처리 루틴으로 이동할 수 있는 인터럽트 벡터의 위치를 지정하기 위해 PC의 값을 1 증가시켜 1로 세트시킨다.
C_{3t2}	M[MAR] ← MBR, IEN ← 0	• MBR이 가지고 있는 다음에 실행할 명령의 주소를 메모리의 MAR이 가리키는 위치(0번지)에 저장한다. • 인터럽트 단계가 끝날 때까지 다른 인터럽트가 발생하지 않게 IEN에 0을 전송한다.
C_{3t3}	F ← 0, R ← 0	• F에 0, R에 0을 전송하여 Fetch 단계로 변천한다.

20. Section 056

실시간 처리와 인터럽트

실시간 처리를 하려면 처리 속도가 느린 입 · 출력장치를 이용해야 하기 때문에 명령 실행 속도가 느린 입 · 출력 명령을 실행해야 한다. 이때 입 · 출력 명령의 실행을 입 · 출력장치 측의 인터페이스

에 넘기고 CPU는 실시간 처리를 요구하는 프로그램을 실행하게 하려면 인터럽트가 필요하다. 이 외에도 Time Sharing System을 이용하는 기법도 있다.

21. Section 057

LOAD는 '꺼내오라'는 명령이므로, A 레지스터에 신호를 주어서 A가 B로부터 꺼내오게 한다.

22. Section 057

BUN(Branch UNconditionally) 명령은 무조건 분기 명령으로, BUN α에서 α 번지로 분기시키기 위해 α 번지를 다음에 실행할 명령의 주소를 가지고 있는 PC에 저장시킨다. 즉 MBR(AD) → PC가 된다.

23. Section 057

- BSA는 복귀주소를 저장하고 부 프로그램을 호출(Call)하는 명령이다.
- **BSA 명령의 실행 순서**

제어 신호		Micro Operation	의미
C_{2t_0}		MAR ← MBR[AD], MBR[AD] ← PC, PC ← MBR[AD]	• MBR에 있는 명령어의 번지 부분을 MAR에 전송한다. • PC의 값(복귀 주소)을 MBR의 주소 부분으로 전송한다. • MBR의 주소 부분을 PC로 전송한다.
C_{2t_1}		M[MAR], ← MBR[AD]	• MBR에 있는 명령어의 번지 부분을 메모리의 MAR이 가리키는 위치에 전송한다.
C_{2t_2}		PC ← PC + 1	PC의 값을 1 증가시킨다.
C_{2t_3}	IEN'	F ← 0	F에 0을 전송하면 F=0, R=00이 되어 Fetch 단계로 변천한다.
	IEN	R ← 1	R에 1을 전송하면 F=1, R=10이 되어 Interrupt 단계로 변천한다.

24. Section 055

마이크로 오퍼레이션이란 하나의 Clock 펄스 동안 실행되는 기본 동작으로, 모든 마이크로 오퍼레이션은 CPU의 Clock에 맞추어 실행된다.

25. Section 057

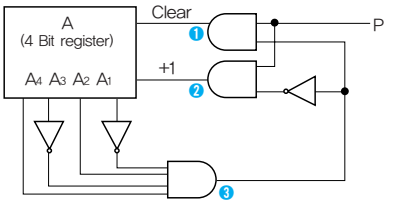
ORG는 프로그램이 시작되는 위치를 알리는 의사(Pseudo) 명령어로 프로그램이 100번지에서 시작됨을 알린다. 프로그램의 각 명령어는 다음과 같은 순서로 메모리에 위치한다.

```
        ORG   100
100     LDA   SUB
101     CMA
102     INC
103     ADD   MIN
104     STA   DIF
105     HLT
106 MIN, DEC   83
107 SUB, DEC   -23
108 DIF, HEX   0
        END
```

위 프로그램은 83-23을 수행하는 것이다.

- 100 : LDA SUB(Load AC from SUB)
  - 메모리의 SUB 위치의 값을 AC로 가져온다(-23을 읽어옴).
- 101 : CMA(Complement AC)
  - AC의 보수를 구한다(10진수로 76이 구해짐).
- 102 : INC(Increment AC)
  - AC의 값을 1 증가시킴(1의 보수를 2의 보수로 만들기 위함, 10진수로 77이됨)
- 103 : ADD MIN(Add MIN to AC)
  - MIN의 값을 AC에 더함(83+77=160, 2의 보수법에서는 자리 올림수를 버리므로 결과는 60이다.)
- 104 : STA DIF(Store AC in DIF)
  - AC의 값을 DIF에 저장함(60이 108번지에 저장됨)

26. Section 057



❸의 출력이 1이 되어야 ❶의 출력이 1이 되어 클리어되므로 클리어 동작은 $A4 \cdot \overline{A3} \cdot A2 \cdot \overline{A1}$이고, 1 증가 동작은 ❸의 출력이 0이 되어야 ❷의 출력이 1이 되어 1 증가 하므로 증가 동작은 $\overline{(A4 \cdot \overline{A3} \cdot A2 \cdot \overline{A1})}$이다. 예를 들어 설명하면, P에는 계속 1이 입력되고 있다고 가정하고 A레지스터가 10진수 10이 되면, 즉 다음과 같이

A4	A3	A2	A1
1	0	1	0

되면 ❸의 출력이 1이되고 ❶의 입력은 1과 1이 되어 출력이 1이므로 A는 0으로 클리어 된다. 동시에 ❷의 출력은 0이 되어 A를 1 증가시키지 않는다.
만약 다음과 같이 A가 8이었다면

A4	A3	A2	A1
1	0	0	0

❸의 출력이 0이 되고 ❶의 입력은 0과 1이므로 출력이 0이 되어 A는 클리어 되지 않는다. 반면 ❷의 출력이 1이 되어 A를 1 증가시켜 A는 1001(9)가 된다.
즉, 위의 제어장치는 0~9까지 증가하다가 10이 되면 다시 0으로 클리어되는 10진 카운터이다.

27. Section 055

- 메모리 한 번 접근 하는데 1사이클
- 연산하는데 1 사이클
- 두 개의 데이터를 가져오고(2 사이클), 연산하고(1 사이클), 메모리에 저장(1 사이클)하므로 총 4사이클이 필요하다.
- 1사이클에 4클록이 소요되면 총 16클록이 필요하다.
- 10Mhz는 1초에 10×1,000,000의 클록이 있다는 의미이다.

∵ 곧 16클록에 필요한 시간은 16 / 10,000,000 = 0.0000016 = 1.6μs이다.
※ 1μs = 10^{-6} = 0.000001

28. Section 058

- 총 메모리 용량 = 제어 메모리 용량 + 나모 메모리 용량
- 제어 메모리의 크기는 나노 메모리에 저장된 마이크로 명령어를 지정할 수 있는 만큼의 크기이다.
  - 제어 레지스터가 10Bit이므로 제어 메모리는 1024개의 워드가 필요하다.
  - 각 워드의 크기는 마이크로 명령어의 종류가 256(=2^8)개이므로 8Bit가 필요하다.
  - 용량 : 2^{10} × 8(256=2^8) = 8K
- 나노 메모리 = 명령어의 개수 × 명령어의 크기 = 256 × 200 = 512000 ≒ 50K

∴ 총 메모리 용량 = 8K + 50K = 58K

29. Section 055

보기 중 하나를 골라야 하니 동기 고정식으로 클록 사이클 타임이 부여된다고 가정하면 마이크로 사이클 타임이 가장 긴 100ns에 지연시간 10ns를 더한 110ns로 결정되어야 한다.

30. Section 058

나노 프로그램을 위한 제어 메모리의 크기는 '마이크로 프로그램의 크기×명령어의 수를 나타낼 수 있는 비트 수' 이다. 비트 수는 128 = 2^7이므로 2,048×7비트의 제어 메모리가 필요하다.

31. Section 058

마이크로 프로그래밍된 제어기의 구성 요소

- Mapping : 제어 메모리에 기계 명령의 마이크로 명령들이 기억되어 있는 시작주소로 변환
- 제어 메모리 : 마이크로 명령들을 저장하고 있는 CPU 내의 메모리
  - 기억 용량 = 2^n×k Bit Word
    단, k = 마이크로 명령의 Bit수 = 제어 워드 길이,
    n = 제어 번지 레지스터의 비트 수
- 주소 결정 회로 : Mapping Table, 제어 번지 레지스터, 다음 번지 생성기
- 기억장치 데이터 레지스터 : 마이크로 명령이 인출되어 저장된 제어 데이터 레지스터

32. Section 058

수평 마이크로 명령(Horizontal Micro Instruction)

- 마이크로 명령의 한 비트가 한 개의 마이크로 동작을 관할하는 명령
- 구성
  - μ-Operation부가 m Bit일 때 m개의 마이크로 동작을 표현한다.
  - Address부의 주소에 의해 다음 마이크로 명령의 주소를 결정한다.
  - 각 비트가 하나의 마이크로 동작을 제어하기 때문에 제어 비트를 디코딩할 필요가 없고, 마이크로 명령 한 개로 여러 개의 하드웨어 구성 요소를 동시에 동작시킬 수 있다.
  - 제어 워드의 비트들이 충분히 활용되지 못하며, 제어 워드의 길이가 길어지기 때문에 비용이 많이 든다.

33. Section 058

μ-Operation부가 m Bit일 때 m개의 마이크로 동작을 표현한다.

34. Section 058

수직 마이크로 명령은 한 개의 마이크로 명령으로 한 개의 마이크로 동작만 제어할 수 있는데, 문제에 제시된 명령어는 마이크로 오퍼레이션 필드가 3개 있으므로 최대 3개의 제어신호를 동시에 발생할 수 있다.

35. Section 058

①번은 수직 마이크로 명령에 대한 설명이다.

36. Section 055

Micro Cycle Time 부여 방식

Micro Cycle Time은 CPU 클록 주기와 Micro Cycle Time의 관계에 따라 동기 고정식, 동기 가변식, 비동기식으로 구분된다.

동기 고정식(Synchronous Fixed)

- 모든 마이크로 오퍼레이션의 동작시간이 같다고 가정하여 CPU Clock의 주기를 Micro Cycle Time과 같도록 정의하는 방식이다.
- 모든 마이크로 오퍼레이션 중에서 수행시간이 가장 긴 마이크로 오퍼레이션의 동작시간을 Micro Cycle Time으로 정한다.
- 모든 마이크로 오퍼레이션의 동작시간이 비슷할 때 유리한 방식이다.

동기 가변식(Synchronous Variable)

- 수행시간이 유사한 Micro Operation끼리 그룹을 만들어, 각 그룹별로 서로 다른 Micro Cycle Time을 정의하는 방식이다.
- 동기 고정식에 비해 CPU 시간 낭비를 줄일 수 있는 반면, 제어기의 구현은 조금 복잡하다.
- 마이크로 오퍼레이션의 동작시간이 차이가 날 때 유리하다(정수배).

비동기식(Asynchronous)

- 모든 마이크로 오퍼레이션에 대하여 서로 다른 Micro Cycle Time을 정의하는 방식이다.
- CPU의 시간 낭비는 전혀 없으나, 제어기가 매우 복잡해지기 때문에 실제로는 거의 사용되지 않는다.

37. Section 057

순서에 맞게 대입해 보면 된다.

- B ← $\overline{B}$: B는 B의 보수인 $\overline{B}$이 된다.
- B ← B+1 : $\overline{B}$에 1을 더했으므로 B는 $\overline{B}$+1이 된다.
- A ← A+B : B는 $\overline{B}$+1이므로 결과는 A+$\overline{B}$+1과 같다.

38. Section 055

인스트럭션의 성능 = 수행시간/(패치시간+준비시간) = 10/(5+3) = 1.25

39. Section 055

- 하나의 명령 사이클을 실행하는 데 2개의 머신 사이클이 필요하고 각 머신 사이클은 5개의 머신 스테이트로 되어 있다고 했으니 1개의 명령 사이클을 실행하는 데 필요한 클록은 총 10클록이다.
- 10MHz는 1초에 10×1,000,000의 클록이 있다는 의미이다.
- 그러므로 1클록에 필요한 시간은 1 / 10,000,000 = 0.0000001 = 0.1μs이다.

∴ 10클록에 필요한 시간은 0.1μs × 10 = 1μs이다.

※ M = 10^6

※ 1μs = 10^{-6} = 0.000001

40. Section 055

하나의 인스트럭션을 수행하기 위한 동작 하나 하나가 마이크로 연산이다. 즉 마이크로 연산이 여러 개 모이면 하나의 명령(Instruction)이 되는 것이고, 그러한 명령이 여러 개 모여 저장장치에 저장되어 있는 것을 마이크로 프로그램이라고 한다. 마이크로 프로그래밍은 마이크로 프로그램을 만드는 작업을 말한다.

5장 정답 및 해설 — 입력 및 출력

1.④	2.③	3.③	4.③	5.④	6.④	7.②	8.④	9.②	10.④	11.④	12.①	13.③	14.④	15.②
16.④	17.①	18.①	19.②	20.②	21.②	22.②	23.②	24.④	25.④	26.③	27.③	28.③	29.②	30.①
31.④	32.①	33.④	34.②	35.③	36.②	37.③	38.④	39.①	40.③	41.②	42.④	43.②	44.③	45.④
46.④	47.④													

1. Section 060

채널은 CPU로부터 입 · 출력 명령을 받으면 주기억장치에서 채널 프로그램을 읽어와 명령을 해독하고 코드를 변환하여 입 · 출력을 직접 수행하는 장치로서 신호를 변 · 복조하는 MODEM의 기능은 없다.

2. Section 059

입 · 출력장치의 동작은 중앙처리장치의 제어기가 제어하는 경우도 있고, 입 · 출력장치의 제어기에 의해 자율적으로 동작하는 경우도 있다.

3. Section 059

- 입 · 출력 처리 시스템은 입 · 출력 동작에 대한 제어 및 관리를 하는 것으로, 가상기억장치와는 관계없다.
- 블로킹 : 자기 테이프에 자료를 입 · 출력할 때 레코드 간에 생기는 갭(Gap)을 줄이기 위해 한 개 이상의 논리 레코드를 묶어 한 개의 물리(불록) 레코드로 만드는 것을 말함

5. Section 059

속도 차이를 해결하기 위한 요소

- Data Register(MDR, MBR) : 속도 차이를 해결하기 위해 I/O 자료를 일시적으로 저장하는 레지스터
- Flag(Status Register) : 속도 차이를 해결하기 위해 MDR에 자료가 차 있는지, 비어 있는지의 상태를 저장하는 Register

6. Section 059

- Channel : Channel은 CPU를 대신하여 주기억장치와 입 · 출력장치 사이에서 입 · 출력을 제어하는 입 · 출력 전용 프로세서(IOP)
- Handshaking : 컴퓨터와 주변장치 상에 Data 전송을 할 때 입 · 출력의 준비나 완료를 나타내는 신호(RDY, ADK)가 필요한 비동기식 병렬 입 · 출력 시스템에 널리 쓰이는 방식
- Interrupt I/O : Interrupt I/O 방식은 입 · 출력을 하기 위해 CPU가 계속 Flag를 검사하지 않고, 데이터를 전송할 준비가 되면 입 · 출력 인터페이스가 컴퓨터에게 알려 입 · 출력이 이루어지는 방식

※ Emulation은 구조가 다른 컴퓨터 시스템의 작동을 흉내내어 같은 일을 처리할 수 있게 만든 것을 말한다.

7. Section 059

입 · 출력장치의 종류

입력장치	키보드, 마우스, 스캐너, OMR, OCR, MICR, BCR(Bar Code Reader), 마이크로 필름 입력장치(CIM, Computer Input Microfilm), 라이트펜, 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

8. Section 060

DMA를 사용할 때, 평균 전송량이 8KB일 때 디스크가 전송에 100% 사용된다고 했으니 4MB를 전송하려면 500번의 DMA가 있어야 한다. 그리고 한 번의 DMA에 1500클록이 사용된다고 했으니 500번의 DMA를 수행하려면 750,000번의 클록이 사용된다.

4MB/8KB = 500

(1000+500) × 500 = 750,000

※ MB = 1,000,000 KB = 1,000

9. Section 059

- 버퍼링(Buffering) : 속도 차이가 심한 두 개 이상 전송장치 간의 속도 차이를 줄이기 위해 전송 또는 수신하는 데이터를 버퍼에 저장하는 동작. 스풀링도 버퍼링의 한 종류로 볼 수 있음
- 다중 프로그래밍(Multiprogramming) : 한 개의 시스템을 이용하

여 동시에 여러 개의 프로그램을 처리하는 방식

- **시분할 시스템(Time Sharing System)** : 각 사용자가 자신의 단말기를 통해 동시에 운영체제를 사용하면서 각자의 프로그램을 실행하는 방식. 시간을 잘게 쪼개 각 사용자에게 연속적으로 할당하므로 각 사용자는 자신이 시스템을 독점하여 사용하는 것으로 생각함

10. Section 059

스풀링과 버퍼링은 모두 큐 방식으로 입·출력을 수행한다.

11. Section 059

스트로브 제어(Stobe Control) 방식의 단점

전송을 시작한 송신장치는 수신장치가 데이터를 받았는지를 알 수 없기 때문에, 핸드셰이킹 방식보다 융통성과 신뢰성이 낮다.

12. Section 059

스트로브 제어 방식에서는 전송을 시작한 송신장치가 버스에 놓인 데이터를 수신 장치가 받아 들였는지 여부를 알 수 없다.

13. Section 060

CPU의 상태 보존이 필요한 것은 인터럽트를 이용한 입·출력 방식이다.

DMA(Direct Memory Access)에 의한 I/O

- DMA는 입·출력장치가 직접 주기억장치를 접근(Access)하여 Data Block을 입·출력하는 방식으로, 입·출력 전송이 CPU의 레지스터를 경유하지 않고 수행된다.
- CPU는 I/O에 필요한 정보를 DMA 제어기에 알려서 I/O 동작을 개시시킨 후 I/O 동작에 더 이상 간섭하지 않고 다른 프로그램을 할당하여 수행한다.
- DMA 방식은 입·출력 자료 전송 시 CPU를 거치지 않기 때문에 CPU의 부담 없이 보다 빠른 데이터의 전송이 가능하다.
- DMA는 인터럽트 신호를 발생시켜 CPU에게 입·출력 종료를 알린다.
- DMA는 Cycle Steal 방식을 이용하여 데이터를 전송한다.

16. Section 066

동시에 고속의 여러 개의 장치를 제어할 수 있는 채널은 블록 멀티플렉서 채널이다.

채널의 종류

Selector Channel (선택 채널)	• 고속 입·출력장치(자기 디스크, 자기 테이프, 자기 드럼)와 입·출력하기 위해 사용한다. • 특정한 한 개의 장치를 독점하여 입·출력한다.
Multiplexer Channel (다중 채널)	• 저속 입·출력장치(카드리더, 프린터)를 제어하는 채널 • 동시에 여러 개의 입·출력장치를 제어한다.
Block Multiplexer Channel	• 고속 입·출력장치를 제어하는 채널 • 동시에 여러 개의 입·출력장치를 제어한다.

17. Section 060

Programmed I/O

- Programmed I/O 방식은 원하는 I/O 작업이 완료되었는지의 여부를 검사하기 위해 CPU가 상태 Flag를 계속 조사하여 I/O 작업이 완료되었으면 MDR(MBR)과 AC 사이의 자료전송도 CPU가 직접 처리하는 I/O 방식이다.
- 입·출력에 필요한 대부분의 일을 CPU가 해주므로 Interface는 MDR, Flag, 장치 번호 디코더로만 구성하면 된다.
- I/O 작업 시 CPU는 계속 I/O 작업에 관여해야 하기 때문에 다른 작업을 할 수 없다는 단점이 있다.

18. Section 062

인터럽트 우선순위 체제에서 인터럽트를 동시에 처리할 수 있도록 요청하는 기능은 없다.

19. Section 060

인터럽트에 의한 입·출력 방식

CPU가 프로그램의 명령들을 순차적으로 실행하던 중 입·출력 명령을 실행할 차례가 되면 입·출력 동작을 개시시킨 후 그 입·출력 명령이 소속된 프로그램에 대하여 CPU에서의 실행을 중단시키고 CPU는 다른 프로그램을 할당하여 실행하게 된다. CPU가 새로운 프로그램을 할당하여 실행하는 동안 입·출력 명령에 대한 처리는 인터페이스에서 수행되고 있다.

※ ②번은 Programmed I/O에 대한 설명이다.

20. Section 060

데이터 대량 전송(Burst Transfer) 및 사이클 스틸링(Cycle Stealing)과 관계 있는 항목은 DMA에 의한 전송이다.

DMA(Direct Memory Access)에 의한 I/O

- DMA는 입·출력장치가 직접 주기억장치를 접근(Access)하여 Data Block을 입·출력하는 방식으로, 입·출력 전송이 CPU

의 레지스터를 경유하지 않고 수행된다.

- CPU는 I/O에 필요한 정보를 DMA 제어기에 알려서 I/O 동작을 개시시킨 후 I/O 동작에 더 이상 간섭하지 않고 다른 프로그램을 할당하여 수행한다.
- DMA 방식은 입·출력 자료 전송 시 CPU를 거치지 않기 때문에 CPU의 부담 없이 보다 빠른 데이터의 전송이 가능하다.
- DMA는 인터럽트 신호를 발생시켜 CPU에게 입·출력 종료를 알린다.
- DMA는 Cycle Steal 방식을 이용하여 데이터를 전송한다.

21. Section 060

Interrupt와 Cycle Steal의 차이점

Interrupt	Cycle Steal
• 수행하고 있던 Program은 정지되지만, 인터럽트 처리 루틴의 명령을 실행하기 위하여 CPU는 수행 상태에 있게 된다. • CPU의 상태 보존이 필요하다.	• CPU는 Steal된 Cycle 동안 완전히 대기 상태, 즉 아무런 동작을 하지 않고 DMA 제어기의 메모리 접근이 완료되기를 기다린다. • CPU의 상태 보존이 필요없다.

22. Section 060

Multiplexer 채널은 저속 입·출력장치용이고, Selector 채널은 고속 입·출력장치용이다.

23. Section 060

시작 번지는 DMA의 구성 요소가 아니라 데이터가 존재하거나 저장될 메모리 블록의 시작 주소다. 시작 번지는 CPU가 DMA 제어기를 초기화하기 위해 데이터 버스를 통해 보내는 정보로서 주소 레지스터에 저장된다.

DMA의 구성요소

- **인터페이스 회로** : CPU와 입·출력장치와의 통신 담당
- **주소 레지스터 및 주소 라인** : 기억장치의 위치 지정을 위한 번지 기억 및 전송
- **워드 카운트 레지스터** : 전송되어야 할 워드의 수 기억
- **제어 레지스터** : 전송 방식 결정
- **데이터 버스 버퍼, 주소 버스 버퍼** : 전송에 사용할 자료나 주소를 임시로 기억함

24. Section 060

채널은 CPU의 간섭 없이 독자적으로 동작하는 입·출력 전용 제어장치로서, 자체적으로 채널 프로그램을 해독 실행하여 자율적인 입·출력 동작을 한다.

25. Section 060

CPU에서 DMA 제어기로 보내는 자료

- DMA를 시작시키는 명령
- 입·출력하고자 하는 자료의 양
- 입력 또는 출력을 결정하는 명령

26. Section 061

- **교착상태** : 여러 프로세스가 서로 다른 프로세스가 사용하는 자원(가능하지 못한 자원)을 요구하며 무한정 기다리는 상태
- **무한 연기** : 사용이 가능한 자원을 기다리는 상태
- Ctrl+Alt+Delete나 RESET을 눌러 시스템을 시작하는 것은 재시작 인터럽트이다.

27. Section 061

인터럽트의 필요성은 CPU 실행과 입·출력의 순차적인 실행에 있는 것이 아니고 예기치 못한 일이 발생하거나 무작위 순서로 작업을 처리하기 위해서이다.

28. Section 059

프로그램 제어 방식은 원하는 I/O가 완료되었는지의 여부를 검사하기 위해 CPU가 상태 Flag를 계속 조사하여 I/O가 완료되었으면, MDR(MBR)과 AC 사이의 자료 전송도 CPU가 직접 처리하는 I/O 방식으로서 전용장치 제어 방식과는 관계가 없다.

29. Section 061

인터럽트를 처리할 때 복귀주소는 스택에 저장된다.

30. Section 061

- 인터럽트는 실제로 인터럽트 원인이 되는 사건이 일어날 때만 발생하는 것이지, 하고자 할 때라든지 원인 발생 전에는 인터럽트가 발생하지 않는다.
- 입·출력장치 동작에 CPU의 기능이 요청될 때를 입·출력 인터럽트라 한다.

31. Section 061

프로그램 실행중 0으로 나누기할 경우, 금지된 영역이나 레지스터에 접근하려 할 경우, 정의되지 않는 불법 명령을 사용한 경우에는 프로그램 오류로 인해 프로그램 실행이 종료되므로 이전 프로그램의 상태 보존이 필요 없다. 더 이상 진행되지 않는다는 것이다. 하지만 Cache Memory에서 캐시 Miss나 가상 메모리 시스템에서

Page Fault가 발생한 경우는 프로그램의 상태를 저장한 후 필요한 블록으로 이동하거나 Page를 주기억장치로 이동시킨 후 실행을 재개하면 되므로 이전 프로그램의 상태 보존이 필요하다.

32. Section 061

내부 인터럽트를 트랩이라 한다. 내부 인터럽트의 원인은 다음과 같다.

- 명령 처리중 오버플로(Overflow) 또는 언더플로(Underflow)가 발생했을 경우
- 0으로 나누는 명령이 수행될 경우

33. Section 061

External Interrupt와 Internal Interrupt는 CPU의 하드웨어 신호에 의해 발생하고, Software Interrupt는 명령어의 수행에 의해 발생한다. Trap은 내부 인터럽트에 대한 다른 표현이다.

34. Section 061

②번은 프로그램 인터럽트에 대한 설명이다. 재시작 인터럽트는 외부에서 키보드로 인터럽트 키를 눌러 발생하는 것으로, 외부 인터럽트에 해당된다.

35. Section 061

인터럽트 서비스 루틴이란 실질적으로 인터럽트를 처리하는 루틴으로, 서비스 루틴에서는 상대적으로 낮은 레벨의 마스크 레지스터를 클리어한다.

36. Section 061

인터럽트 동작 원리

❶ 인터럽트 요청 신호 발생
❷ **프로그램 실행을 중단** : 현재 실행중이던 명령어는 끝까지 실행
❸ **현재의 프로그램 상태를 보존** : 프로그램 상태는 다음에 실행할 명령의 번지를 말하는 것으로서 PC(프로그램 카운터)가 가지고 있다. PC의 값을 메모리의 0번지에 보관
❹ **인터럽트 처리 루틴을 실행** : 인터럽트 처리 루틴을 실행하여 인터럽트를 요청한 장치를 식별
❺ **인터럽트 서비스(취급) 루틴을 실행** : 실질적인 인터럽트를 처리
❻ **상태 복구** : 인터럽트 요청 신호가 발생했을 때 보관한 PC의 값을 다시 PC에 저장
❼ **중단된 프로그램 실행 재개** : PC의 값을 이용하여 인터럽트 발생 이전에 수행중이던 프로그램을 계속 실행

37. Section 061

프로그램 수행중에 인터럽트가 발생하면 현재 수행중인 인스트럭션(명령)을 끝내고, 다음에 수행할 명령의 위치를 저장한 후 인터럽트 처리 루틴으로 분기한다. 프로그램은 수많은 인스트럭션으로 구성되어 있다.

38. Section 061

마스크 레지스터는 우선순위가 높은 것이 서비스 받고 있을 때 우선순위가 낮은 것이 CPU에 인터럽트를 요청할 수 없도록 한다.

39. Section 061

제어 프로그램 호출(SVC, Supervisor Call) 인터럽트

사용자가 프로그램 내에서 SVC 명령을 써서 의도적으로 호출한 경우

40. Section 062

프로그램 카운터(PC)의 내용(복귀주소)은 메모리의 0번지나 스택에 보관한다.

41. Section 062

인터럽트 우선순위

정전 → 기계 고장 → 외부신호 → 입·출력 → 프로그램 오류 → SVC

※ 프로그램의 연산자나 주소지정방식의 잘못으로 인한 인터럽트는 프로그램 오류 인터럽트이다.

42. Section 062

인터럽트 서비스 루틴은 실질적인 인터럽트에 대한 조치를 취하는 단계이다.

43. Section 062

하드웨어적인 방식인 벡터 방식이 소프트웨어 방식인 폴링보다 빠르고, 복합적인 방식보다는 단일 방식이 빠르다.

44. Section 062

실시간 응용 시스템이란 여러 사용자의 요구에 즉각 응답하는 시스템으로, 여러 개의 장치에서 동시에 인터럽트가 발생할 수 있으므로 인터럽트 우선 체제는 특히 중요하다.

45. Section 062

- 병렬 우선순위 부여 방식은 인터럽트가 발생하는 각 장치를 개별적인 회선으로 연결하는데 다수의 인터럽트 요청이 있는 경

우, 특정 인터럽트에 대해 인터럽트가 금지되도록 하는 방법을 Mask라 한다.

- 각 장치의 인터럽트 요청에 따라 각 비트가 개별적으로 Set될 수 있는 Mask Register를 사용한다.
- 마스크 레지스터는 우선순위가 높은 것이 서비스받고 있을 때 우선순위가 낮은 것을 비활성화시킬 수 있다.

46. Section 061

페이지 폴트는 가상 메모리 시스템에서 CPU가 액세스한 가상 페이지가 주기억장치에 없는 경우로서 프로그램 오류와 관계없이 발생한다.

프로그램 오류로 발생하는 인터럽트

- 0으로 나눌 때
- 프로그램에서 명령어나 연산자를 잘못 사용한 경우
- Overflow 또는 Underflow가 발생한 경우
- 금지된 자원의 접근

47. Section 061

- ④번은 외부 인터럽트가 발생하는 이유이다.

프로그램 검사 인터럽트(Program Check Interrupt)

- 0으로 나누기(Divide by zero)가 발생한 경우
- Overflow 또는 Underflow가 발생한 경우
- 프로그램에서 명령어를 잘못 하용한 경우
- 부당한 기억장소의 참조와 같은 프로그램의 오류가 발생한 경우

6장 정답 및 해설 — 기억장치

1. ④	2. ④	3. ③	4. ③	5. ③	6. ③	7. ①	8. ④	9. ①	10. ①	11. ③	12. ①	13. ③	14. ①	15. ③
16. ①	17. ②	18. ①	19. ③	20. ②	21. ④	22. ①	23. ④	24. ①	25. ①	26. ④	27. ①	28. ②	29. ②	30. ④
31. ①	32. ④	33. ③	34. ①	35. ①	36. ③	37. ④	38. ②	39. ①	40. ③	41. ②	42. ①	43. ④	44. ①	45. ③
46. ①	47. ①	48. ③	49. ①	50. ④	51. ①	52. ①	53. ①	54. ②	55. ①	56. ②	57. ③	58. ④	59. ④	60. ④
61. ④	62. ③	63. ①	64. ③	65. ③	66. ①									

1. Section 063

MUX는 2^n 개의 입력 중 한 개 선을 선택할 때 사용하는 조합논리 회로로서 기억장치와 직접적인 관계는 없다.

- DMA : 중앙처리장치를 거치지 않고 주기억장치를 직접 접근하는 입 · 출력 방식
- MAR : 데이터가 저장된 또는 저장할 기억장소의 주소를 일시적으로 기억하는 레지스터
- MBR : 주기억장치에 저장할 자료나 불러온 자료를 일시적으로 기억하는 레지스터

2. Section 063

기억장치의 특성을 결정하는 요소(특성량)

- 기억 용량
- Access Time
- Cycle Time
- Bandwidth(대역폭)

3. Section 063

- 자기 테이프는 순차처리만 할 수 있기 때문에 평균 접근시간이 가장 길다.
- 기억장치들의 빠르기 순서(빠름 → 느림)
  Register → Cache Memory → Associative Memory → Main Memory(RAM → ROM → 자기코어) → 자기 드럼 → 자기 디스크 → 자기 테이프

4. Section 063

Access Time은 기억장치에 읽기 요청이 발생한 시간부터 요구한 정보를 꺼내서 사용 가능할 때까지의 시간이고, Cycle Time은 기억장치에 읽기 신호를 보낸 후 다시 읽기 신호를 보낼 수 있을 때

까지의 시간 간격을 말한다. 반도체 메모리는 접근 시간과 사이클 시간이 같지만 자기 코어 같은 파괴 메모리에서는 재 저장 시간이 필요하기 때문에 사이클 타임이 액세스 시간보다 길다.

5. Section 063

기억장소의 위치에 상관없이 접근 시간이 동일하게 Access하는 방법을 Random Access라 하며, RAM이나 ROM 같은 주기억장치의 Access 방법이 이에 해당한다.

6. Section 063

전원이 공급되지 않으면 그 내용이 증발되는 메모리를 휘발성(Volatile) 메모리 또는 소멸성 메모리라고 한다. 우리가 알고 있는 것 중 RAM이 여기에 해당되고 나머지는 모두 비휘발성 메모리이다.

7. Section 063

Computer 내부에 있는 주기억장치를 Main Storage 또는 Main Memory라고 부른다. 보통 Memory라고만 해도 주기억장치인 RAM을 의미한다.

- Accumulator : 연산의 결과를 일시적으로 저장하는 레지스터
- Magnetic Memory : 자기(자석) 메모리를 뜻함
- Register : CPU 내부에서 명령이나 데이터를 일시적으로 기억하는 고속의 임시 메모리

8. Section 064

캐시 메모리에 주로 사용 되는 것은 SRAM이다.

9. Section 063

사이클 타임(Mt)이 액세스 타임(At)보다 큰 경우는 자기 코어 같은 파괴 메모리에서 재저장 시간이 필요하기 때문이다. DRO(Destructive Read Out) Memory는 파괴 메모리를 의미한다.

10. Section 064

전류 일치 기술이란 자기 코어 기억장치에서 둘 또는 그 이상의 서로 다른 전류를 동시에 흘려서 판독/기록할 자기 코어를 선택하는 것을 말한다.

11. Section 064

코어는 한 개의 워드를 구성하는 비트 수만큼 Core Plain을 겹쳐 쌓는다. 즉 16장이므로 16Bit가 1워드이다.
가로 세로 각 32개이므로 32×32 = 1024, 즉 1K이다.

12. Section 064

주기억장치는 크게 운영체제가 상주하는 시스템 프로그램 영역과 일반 응용 프로그램이 상주하는 사용자 프로그램 영역으로 구분된다.

13. Section 064

(4096×16)/(256×4) = 64개의 256×4비트의 구성을 갖는 메모리 IC가 필요하다.

14. Section 064

전체 기억 용량을 메모리 1개의 크기로 나누면 메모리 수가 되고, Address Line은 계산된 메모리 수를 모두 접근할 수 있는 비트 수가 있으면 된다.

- 메모리 수
  - 메모리 1개의 용량이 1Byte이므로 64KByte/1Byte = 64K개
  - K = 1024 = 2^{10}이고, 64 = 2^6이므로 64K는 2^{16}이다.

※ 64K는 정확히 65,536이다.

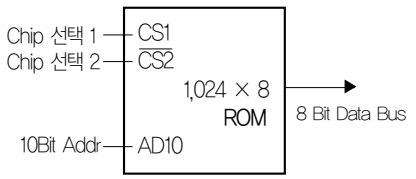
- 주소선 수
  - 주소선이 n개라면 2^n개의 메모리를 지정할 수 있다.
  - 2^{16}개의 메모리라면 16개의 주소선이 필요하다.

15. Section 064

주소선의 수는 워드의 수이고, 데이터 선의 수는 워드의 크기다. 주소선의 수가 n개라면 2^n개의 워드를 나타낼 수 있고, 데이터 선의 개수가 8이므로 워드의 크기는 1Byte이다.
즉 워드의 수는 2^{12} = 4096 = 4K이고, 워드의 크기는 8이므로 4K×8과 같이 표기한다.
※ 1K = 1024 = 2^{10}

17. Section 064

- 핀(Pin)의 수는 회로에 연결되는 선의 개수를 의미한다.
- 1,024개의 워드를 지정할 수 있는 10Bit의 주소선(2^{10} = 1024)과 8Bit의 워드를 실어 나를 수 있는 8개의 데이터 선이 필요하다.



1,024×8 ROM의 구조

18. Section 064

Meta Bit : Tag Bit라고도 하며, 주기억장치에 기억되는 명령과 자료를 구분하기 위해 모든 기억장소마다 별도로 둔 비트를 말함

19. Section 064

모니터 프로그램

사용자 프로그램의 동작을 제어하고 여러 가지 하드웨어, 소프트웨어의 활동을 총괄하는 기억장치 내에 상주하는 프로그램으로 보통 마이크로 프로그램으로 제작되어 ROM에 저장된다.

20. Section 064

- 정적 RAM과 동적 RAM 모두 전원을 끊는 순간 기억된 데이터가 소멸된다.
- 매 밀리 초마다 셀에 재생 신호를 가하는 것은 동적 램이다.

SRAM과 DRAM의 비교

구분	정적 램(SRAM, Static RAM)	동적 램(DRAM, Dynamic RAM)
특징	• 플립플롭(Flip-Flop)으로 제작 • 전원이 공급되는 한 기억된 내용이 소멸되지 않는다. • 고속 처리가 필요한 특수 메모리 구성에 사용한다.	• 전하 충전을 이용하는 콘덴서로 제작 • 기억된 정보를 유지하기 위해 일정 시간간격으로 재생(Refresh) 동작을 가해야 한다. • 일반적인 주기억장치로 사용한다.
장점	• 원하는 시점에 바로 접근이 가능하다. • 속도가 빠르다.	• 소비 전력이 적다. • 집적도가 높다(고밀도 집적회로). • 기억 용량이 크다. • 가격이 싸다.
단점	• 소비 전력이 많다. • 집적도가 낮다(저밀도 집적회로). • 기억 용량이 작다. • 가격이 비싸다.	• 재생 동작이 이루어지는 시간 때문에 원하는 시점에 바로 접근할 수 없다. • 속도가 느리다.

21. Section 064

ROM(Read Only Memory)의 종류

종류	특징
Mask ROM	제조공장에서 프로그램화하여 생산한 ROM으로, 사용자가 내용을 변경시킬 수 없다.
PROM (Programmable ROM)	PROM 프로그램 장치라는 특수장치를 이용하여 비어 있는 ROM에 사용자가 한 번만 내용을 기입할 수 있으며, 이후엔 읽기만 가능하다.
EPROM (Erasable PROM)	• 자외선을 쏘여서 기입한 내용을 지울 수도 있고, PROM 프로그램 장치로 내용을 기입할 수도 있다. • 사용자가 여러 번 반복해서 지우거나 기입할 수 있다.
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

22. Section 064

데이터의 연산은 연산장치의 기능이다.

23. Section 064

RAM에서 Access 회로(번지 해독기의 출력선 수)를 줄이기 위해 주로 배열 형태로 구성한다.

24. Section 065

액세스 암은 자기 디스크에서 읽기 쓰기 헤드를 이동하는 장치로 자기 테이프에는 없다.

25. Section 065

보조기억장치는 입력장치인 동시에 출력장치로도 사용할 수 있다.

- **보조기억장치** : 자기 디스크, 자기 테이프, 자기 드럼 등

26. Section 065

한 개의 셀이 1Bit를 저장하므로 6,000×30 = 180,000Bit를 저장할 수 있다.

27. Section 067

캐시의 용량이 주기억장치에 비해 적기 때문에 캐시의 각 주소 라인은 여러 개의 주기억장치 블록들에 의해 공유된다. 그렇기 때문

에 각 캐시에 기억시키는 블록의 주소에는 데이터 블록 외에도 현재 자신을 공유하는 블록들 중 어느 것이 적재되어 있는지를 구분해주는 태그 주소가 저장되어 있어야 한다.

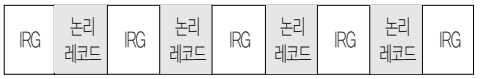
28. Section 064

RAM은 칩 선택선, 쓰기 신호선, 읽기 신호선이 필요하므로 최소한 3개의 입력 단자가 있어야 한다.

29. Section 065

- 블로킹이란 한 개 이상의 논리적 레코드를 묶어서 테이프에 기록하는 방식이다.

비 블로킹(Unblocking)



블로킹(Blocking)

- 하나의 블록을 구성하는 논리 레코드의 개수를 블록화 인수(BF, Blocking Factor)라고 한다.

블로킹의 장점

- 블로킹을 하면 블로킹을 하지 않았을 때에 비해 IRG의 수가 줄어들기 때문에 다음과 같은 장점이 있다.
  - 기억 공간의 낭비가 줄어든다.
  - Access Time이 감소한다.
  - 입 · 출력 횟수가 감소한다.

※ 블로킹을 한다고 해서 에러 발생률이 적어지는 것은 아니다.

30. Section 065

- 800BPI의 기록밀도는 1인치당 800문자(Byte)를 저장할 수 있는 것으로, 1200×12×800 = 11,520,000, 약 12M 문자를 기록할 수 있다.

※ 자기 테이프에서는 트랙의 수와 기록할 수 있는 양은 관계가 없다.

32. Section 065

- 문제에 주어진 자료는 다음과 같은 형태이다.

- 테이프의 길이 = (블록의 길이 + IBG) × 블록의 개수

① **블록의 길이** : IBG와 IBG 사이의 논리 레코드들의 길이의 합
  - 800BPI는 1Inch의 길이에 800자가 기록된다는 의미이다.
  - 1레코드의 길이에 대한 언급은 없지만 종이 카드 한 장은 80컬럼, 즉 80자이다.
  - 1레코드의 길이는 $\frac{80}{800}$, 즉 0.1 Inch이다.
  - 1블록에 들어가는 논리 레코드의 수가 1이므로 1블록의 길이는 0.1inch이다.

② **IBG** : 0.75 Inch

③ **블록의 개수** : 12,000장의 카드가 있고, 블록당 1개의 레코드가 기록되므로 12,000블록

$\therefore (0.1 + 0.75) \times 12{,}000 = 10{,}200$ Inch

$= \frac{10{,}200}{12} = 850$ Feet(1피트는 12인치)

33. Section 067

분리 캐시란 명령어와 데이터를 따로 분리하여 각각의 캐시 메모리에 저장하는 것으로 적중률은 떨어지지만 캐시 접근 시 충돌을 방지할 수 있다.

34. Section 065

디스크의 가장 윗면과 가장 아랫면은 사용하지 않는다. 따라서 디스크가 6매이면 총 12면 중 윗면과 아랫면을 제외한 10면을 사용한다.

36. Section 065

- 디스크 카트리지는 디스크 팩에서 양면을 사용할 수 있는 한 장의 디스크를 말한다.
- 네 개의 섹터라는 것은 한 개의 트랙에 네 개의 섹터가 있다는 의미이다.
- **디스크의 기억 용량**

  = 2(면)×200(트랙)×4(섹터)×320(워드)

  = 512,000

37. Section 065

자기 테이프는 연속적인 기록과 순차 처리만 가능하기 때문에 주소가 필요 없다.

38. Section 065

라이브러리 프로그램은 제곱근을 구하는 것과 같이 프로그램 작성 시 자주 이용하는 프로그램 루틴의 모임이기 때문에 기억 용량이

크고, 접근시간이 짧을 뿐만 아니라 순차 처리는 물론 직접 접근도 가능한 자기 디스크에 저장하는 것이 바람직하다.

39. Section 066

CAM(Content Addressable Memory)

- 연관기억장치(Associative Memory)라고도 한다.
- 주소에 의해 접근하지 않고, 기억된 내용의 일부를 이용하여 Access할 수 있는 기억장치이다.
- Mapping Table(사상 테이블) 구성에 주로 사용한다.
- 병렬 판독 회로가 있어야 하므로 하드웨어의 비용이 크다.
- CAM을 주기억장치로 사용하는 컴퓨터를 연관(Associative) 컴퓨터라고 부른다.

※ 파괴적(Destructive)으로 읽는다는 것은 자기 코어에서처럼 읽은 후 내용이 지워져 재저장 시간이 필요한 것을 말한다. 파괴적 메모리는 내용을 읽은 후 재저장 시간만큼 사이클 타임이 길어지므로 CAM의 기억소자로는 비효율적이다.

40. Section 066

연관 메모리(Associative Memory)는 인수 레지스터, 키 레지스터로, 매치 레지스터로 구성되어 있다.

- **인수 레지스터** : 찾고자 하는 내용의 일부를 기억하는 레지스터, 데이터 레지스터라고도 함
- **키 레지스터**
  - 인수 레지스터에 기억된 내용 중 검색에 사용할 비트를 결정하는 레지스터이다.
  - 인수 레지스터에서 검색에 사용할 Bit와 동일한 위치의 Bit에 1을 Set시켜 놓는다.
  - 1이 Set되어 있는 Bit들을 Mask Bit라 하여 키 레지스터를 마스크 레지스터라고도 한다.
- **매치 레지스터** : 인수 레지스터의 검색 비트를 포함하고 있는 워드를 찾은 경우 찾았음을 표시하기 위해 사용한다. 일치지시기라고도 함

41. Section 066

연관기억장치는 키 레지스터의 Mask Bit와 대응하는 일부 내용에 의해 액세스한다.

42. Section 066

연관기억장치는 액세스 속도를 빠르게 하는 것이 목적이다.

44. Section 064

ROM은 기록할 수 없고 오직 읽기만 가능한 기억장치이므로 쓰기 신호가 필요하지 않다.

45. Section 067

캐시 설계 시 고려할 사항

- 캐시의 크기(Cache Size)
- 전송 Block Size
- 교체 알고리즘(Replacement Algorithm)

46. Section 067

캐시 기억장치를 사용하면 사용하는 명령이나 데이터가 캐시 기억장치에도 기억되므로 캐시에는 현재 실행 중인 코드가 저장되어 있다고 할 수 있다.

47. Section 067

캐시 기억장치(Cache Memory)

- CPU의 속도와 메모리의 속도 차이를 줄이기 위해 사용하는 고속 Buffer Memory이다.
- 처리 속도가 거의 CPU의 속도와 비슷할 정도로 고속 처리를 하는 소용량 기억장치이다.
- 메모리 참조의 국소성(Locality of Reference)을 이용하여 현재 CPU에 의해 지속적으로 참조되는 프로그램의 일부를 미리 옮겨 놓고 CPU가 주기억장치 대신 이 곳을 접근하도록 함으로써 프로그램의 전체 실행시간을 단축시키는 기법이다.
- Cache의 성능을 나타내는 척도를 적중률(Hit Ratio)이라 한다.
- 적중률은 CPU가 캐시를 접근 시도한 전체 횟수와 적중(내용이 캐시에 있어서 참조 가능한 상태) 횟수의 비로 나타낸다.

※ 캐시는 고속 처리를 할 수 있는 소용량 기억장치이다.

48. Section 066

CAM(Content Addressable Memory)은 데이터를 병렬 탐색하기에 알맞도록 되어 있다.

49. Section 067

캐시 메모리에서 자료를 찾기 위한 매핑 테이블로 Associative Memory를 이용하는 것이 가장 효율적이다.

50. Section 067

적중률(Hit Ratio) : 주기억장치 참조 횟수에 대한 캐시 적중의 비

- 액세스 시간 = 캐시에서 찾는 데 걸리는 시간 + 주기억장치에서 찾는 데 걸리는 시간
- 캐시에서 찾는 데 걸리는 시간 = 80ns×0.9 = 72ns
- 주기억장치에서 찾는 경우는 캐시에서 못 찾았을 때이므로 캐시를 찾는 데 걸린 시간과 주기억장치에서 찾는 시간을 더해야 한다.
  (80ns × 0.1) + (1㎲ × 0.1) = (80ns × 0.1) + (1000ns×0.1) = 108(적중률이 0.9이므로 캐시에 없을 확률은 0.1이다.)
- 72 + 108은 180ns이다.

※ ㎲ = 마이크로 초(10^{-6}), ns = 나노 초(10^{-9}), 1㎲ = 1,000ns

51. Section 068

- 가상기억장치란 하드웨어적으로 실제로 존재하는 것이 아니라 기억용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용할 수 있도록 하는 운영체제의 운영 기법이다.
- 가상기억장치의 목적은 속도가 아니라 기억공간의 확보이다. 가상기억장치로 사용하는 보조기억장치에 자주 접근하게 되면 컴퓨터 시스템의 처리 효율이 저하된다.

52. Section 068

메모리 어드레스 Mapping Table은 가상 메모리 체계에서 보조기억장치인 가상주소를 실주소인 주기억장치의 주소로 변환할 때 사용하는 주소 변환 테이블이다.

53. Section 065

등각속도 방식에서는 트랙의 저장 밀도가 다르다.

등각속도(等角速度, Constant Angular Velocity)

- 등각속도란 디스크 저장 매체에서 디스크 회전 속도를 일정하게 하고 디스크의 회전각에 따라 데이터를 저장하는 방식이다.
- 디스크 내곽과 외곽의 회전 속도 차이로 생기는 데이터의 밀도가 달라 외곽에 저장공간의 낭비가 생기는 단점이 있으나, 헤드의 위치에 따라 디스크 회전 속도를 조절하는 상수선형속도(CLV)에 비해 데이터 접근 속도가 빠르다.

54. Section 068

가상기억장치는 보조기억장치의 일부를 주기억장치처럼 사용하는 메모리 관리 기법으로, 가상기억장치를 사용하면 주기억장치의 이용률과 다중 프로그램의 효율을 높일 수는 있지만 가상기억장치를 채택하지 않는 시스템에서에 비해 실행 속도가 빠르지는 않다.

가상기억장치

- 기억 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용할 수 있도록 하는 운영체제의 메모리 운영 기법이다.
- 가상기억장치의 목적은 보조기억장치를 이용한 주기억장치의 용량 확보이다.
- 가상기억장치는 하드웨어적으로 실제로 존재하는 것이 아니고 소프트웨어적인 방법으로 보조기억장치를 주기억장치처럼 사용하는 것이다.
- 사용자 프로그램을 여러 개의 작은 블록으로 나눠 보조기억장치 상에 보관해놓고 프로그램 실행 시 필요한 부분들만 주기억장치에 적재한다.
- 주기억장치의 이용률과 다중 프로그래밍의 효율을 높일 수 있다.
- 오버레이(Overlay) 문제가 자동적으로 해결된다.
- 가상기억장치 기법에서 사용하는 보조기억장치는 디스크 같은 DASD 장치이어야 한다.

55. Section 068

가상 메모리 기법을 사용하면 실행할 수 있는 프로그램의 크기에 제약을 받지 않는다.

56. Section 065

SSD(Solid State Drive)

디스크 드라이브(HDD)와 비슷하게 동작하면서 HDD와는 달리 기계적 장치가 없는 반도체를 이용하여 정보를 저장하는 컴퓨터 보조기억장치이다. 기억 매체로는 플래시메모리나 DRAM을 사용하는데 DRAM은 전원 공급이 중단되면 저장된 내용이 모주 지워지는 단점이 있어 많이 사용하지는 않는다.

SSD의 장점

- 고속으로 데이터를 입 · 출력할 수 있다.
- 기계적 지연이나 실패율이 거의 없다.
- 외부 충격으로 인한 데이터의 손상이 없다.
- 발열 · 소음과 전력 소모가 적으며, 소형화 · 경량화 할 수 있다.

57. Section 068

단편화를 수집해서 커다란 빈 공간으로 만들기 위해서는 쓰레기 수집, 통합, 압축이 사용되는데, 이 과정에서 프로그램들을 한쪽으로 모으면서 프로그램의 주소를 지정해 주는 작업을 재배치라고 한다.

58. Section 068

비트수는 지정할 수 있는 기억장소의 개수를 말한다. 세그먼트를 지정할 수 있는 비트수가 4비트면 2^4개의 세그먼트를 지정할 수 있고, 각 세그먼트는 2^8개의 페이지를 지정할 수 있으며, 각각의 페이지는 2^8개의 워드를 지정할 수 있다. 즉 $2^4 \times 2^8 \times 2^8 = 2^{20}$개의 워드를 지정할 수 있다.

59. Section 068

가상기억장치 관리 기법에서 기억장치의 사용자 관점이란 사용자가 프로그램을 작성할 때 내용으로 구분한 영역 단위를 교체의 단위로 사용하는 것을 말한다.

Segmentation

- **세그먼트** : 고객관리를 하는 프로그램을 주 메뉴 프로그램과 고객 등록 프로그램, 고객 조회 프로그램 등으로 나누어 작성하는 것처럼, 하나의 작업 처리에 관련된 기능들에 대해 기능별로 독립시켜 작성한 부 프로그램 단위의 프로그램들을 의미하는 것
- 세그먼트 기법은 기억장치의 사용자 관점을 보존하는 기억장치 관리 기법이다.
- 세그먼트화하는 근본 이유는 기억공간의 절약이다.

※ 페이징 시스템은 논리적인 세그먼트들의 특성을 무시한 채 프로그램을 일률적인 페이지 크기로 나누어 실기억장치에 비연속적으로 기억시킨다.

60. Section 068

Segment 기법은 주기억장치의 기억공간 확대를 목적으로 하는 가상기억장치에 사용되는 기법이므로, 프로그램을 세그먼트 단위로 나누는 근본 목적 또한 기억공간을 늘린 것처럼 활용할 수 있도록 단편화(낭비되는 기억공간 조각)를 최소화하는 것이다.

61. Section 064

$K = 2^{10} = 1024$이므로 $4K = 4 \times 1024 = 4096$개이다.

62. Section 068

- 다중 프로그래밍에서는 여러 개의 프로그램이 동시에 병렬로 실행된다. 이때 어떤 프로그램에 의해 다른 프로그램의 결과가 잘못 쓰여지지 않도록 기억보호를 한다.
- 기억보호(Memory Protection)는 메모리의 각 블록에 허락할 수 있는 접근 형태를 지정하는 보호 비트(Protection Bit)를 둠으로써 이루어진다.

63. Section 068

가상 메모리를 사용한 컴퓨터에서 Page Fault가 발생하면 요구된 Page가 주기억장치로 옮겨질 때까지 프로그램 수행이 중단된다. 이때 교체할 페이지를 결정해서 보조기억장치의 이전 위치에 기억시키고 새로운 페이지를 교체한 페이지의 위치에 놓는 것을 스테이징이라고 한다.

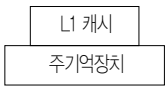
64. Section 064

- 이 문제는 어려워 보이지만 간단한 문제이다. 문제의 중간부에 있는 "하나의 Indirect Mode Bit, Operation Code, Processor Register를 나타내는 2비트와 Address Part~"는 문제 풀이와 전혀 관계가 없다.
- 기억장치에 있는 단어가 65,536개이므로 PC와 MAR은 65,536개를 지정할 수 있는 비트가 필요하다. $65,536 = 2^{16}$이므로 16비트이다.
- MBR은 한 단어의 크기와 같으므로 25비트가 필요하다.

65. Section 067

L1만 사용할 때와 L1, L2를 사용할 때의 액세스 시간을 계산하여 비교하면 된다.

L1만 사용할 때의 액세스 시간



찾는 자료가 L1 캐시에 없을 경우 주기억장치에서 자료를 찾으므로 액세스 시간은 다음과 같다.

- 메모리 액세스 시간 = L1 히트 시간 + L1 미스율 × L1 미스 패널티
  $= 1 + 0.05 \times 100 = 6$사이클
  - L1 히트 시간 : L1 캐시에서 자료를 찾는 데 걸리는 시간
  - L1 미스율 : L1 캐시에 자료가 없을 확률로 주기억장치에서 자료를 찾아야 함
  - L1 미스 패널티 : L1 캐시에 자료가 없을 경우 주기억장치를 액세스 하는 데 걸리는 시간, 문제에 주어진 L2 미스 패널티가 주기억장치를 액세스 하는 데 걸리는 시간이므로 L1 미스 패널티로 사용하면 됨

※ L2 캐시가 없을 경우 L1의 미스 패널티와 L1, L2 캐시를 사용하는 시스템에서의 L2 미스 패널티는 같다.

L1, L2 캐시를 사용할 때의 액세스 시간

찾는 자료가 L1 캐시에 없을 경우 L2 캐시를 액세스 한다. L2 캐시에도 자료가 없을 경우 주기억장치를 액세스 한다. L1 캐시에 자료가 없을 경우 L2 캐시를 액세스 하는데 걸리는 시간이 L1 미스 패널티이고, L2 캐시에 자료가 없을 경우 주기억장치를 액세스 하는데 걸리는 시간이 L2 미스 패널티다. 즉 L2 미스 패널티를 이용하여 L1 미스 패널티를 구한 후 전체에 대한 액세스 시간을 계산하면 된다.

- L1 미스 패널티 = L2 히트 시간 + L2 미스율 × L2 미스 패널티
  = 4 + 0.2 × 100
  = 24 사이클
- 메모리 액세스 시간 = L1 히트 시간 + L1 미스율 × L1 미스 패널티
  = 1 + 0.05 × 24
  = 2.2 사이클

∴ 6/2.2 = 2.73, 약 2.7배 액세스 시간이 향상된다.

66. Section 068

- 전체 페이지 테이블의 크기는 '페이지 수 × 페이지 테이블 엔트리의 크기'인데, 페이지 테이블 엔트리의 크기가 4바이트라고 주어졌으니 페이지 수만 계산하면 된다.
- 페이지 수는 주어진 가상 기억장소에 사용될 수 있는 페이지의 수를 의미하는 것으로 '가상 기억장소의 크기 / 페이지의 크기'다.
  - 가상 주소로 32비트를 사용한다고 했으니 가상 기억장소의 크기는 2^{32}바이트이다. 그리고 페이지의 크기는 4KB(2^{12})로 주어졌다.
  - 페이지 수 = $2^{32} / 2^{12} = 2^{20}$개다.
- 전체 페이지 테이블의 크기 = 페이지 수 × 페이지 테이블 엔트리의 크기

  = $2^{20} \times 4$

  = 1,048,576 × 4

  = 4,194,304

  = 4M바이트이다.

※ K = 2^{10} = 1,024

※ 4KB = $4 \times 2^{10} = 2^2 \times 2^{10} = 2^{12}$ = 4,096

※ M = $2^{20} = 2^{10} \times 2^{10}$ = K×K = 1,048,576

※ 4M = 4 × 1,048,576 = 4,194,304

3과목 운영체제

1장 정답 및 해설 — 운영체제의 개요

1.①	2.③	3.④	4.①	5.④	6.②	7.①	8.④	9.④	10.④	11.④	12.④	13.④	14.④	15.④
16.①	17.①	18.④	19.②	20.④	21.①	22.④	23.④	24.④	25.③	26.④	27.①	28.②	29.③	30.②
31.②	32.②	33.②	34.④	35.①	36.②	37.③	38.①	39.①	40.①					

1. Section 070

고급 언어로 작성된 프로그램을 컴파일하여 기계어로 만들어 주는 것은 언어 번역 프로그램(컴파일러, 인터프리터 등)이다.

3. Section 070

운영체제는 시스템 사용 도중 발생하는 내부, 외부적인 오류로부터 시스템을 보호하는 기능을 제공한다.

4. Section 070

시스템 성능 평가 기준 : 처리 능력(Throughput), 반환 시간(Turn Around Time), 사용 가능도(Availability), 신뢰도(Reliability)

5. Section 070

원시 프로그램을 목적 프로그램으로 변환하는 것은 언어 번역 프로그램의 역할이다.

6. Section 070

링킹(Linking)은 목적 프로그램과 또 다른 목적 프로그램, 라이브러리 함수 등을 결합하여 실행 가능한 모듈을 만드는 것으로, 처리 프로그램의 링커(Linker)가 이를 담당한다.

7. Section 070

VP400은 컴퓨터 시스템의 명칭이다.

8. Section 070

문서 작성기는 운영체제 기반에서 동작하는 응용 프로그램의 하나이다.

9. Section 070

시스템 자원이라 함은 운영체제의 관리를 받거나 사용 되는 것들을 말하는데, 언어 매뉴얼은 프로그램을 위한 도구이다.

10. Section 069

언어는 프로그래머에 의해 시스템에 가장 적합한 특성을 지닌 것을 선택하여 사용하게 된다.

11. Section 070

- 펌웨어(Firmware)는 ROM에 저장시켜서 사용하는 마이크로 프로그램들을 말하며, 변경되지 않는 내용을 저장하여 계속적으로 사용한다. 이러한 특성을 지닌 것은 운영체제의 핵심 부분이다.

※ 마이크로 프로그램은 ROM이나 PROM에 영구히 기록되는 프로그램을 의미한다.

12. Section 075

부트 로더(Boot Loader)

- 운영체제를 적재할 수 있도록 하는 것으로, ROM에 저장되어 있으며 메모리가 비어 있는 상태에서 처음에 실행되는 프로그램을 말한다.
- Reset 스위치를 누르면 다시 실행된다.
- 부트 스트랩 로더라고도 하며, 이를 통해 부팅(부트 스트랩, BootStrap)이 수행된다.

13. Section 071

문제는 하나의 CPU와 주기억장치를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식인 다중 프로그래밍에 대한 설명이다. 다중 프로그래밍은 한 프로그램이 입 · 출력하는 동안 다른 프로그램이 CPU를 사용할 수 있으므로 시스템의 효율과 CPU의 처리량을 증가시킬 수 있다.

14. Section 071

운영체제 발전 과정은 '일괄 처리 시스템 → 다중 프로그래밍 시스템, 다중 처리 시스템, 시분할 시스템, 실시간 처리 시스템 → 다중 모드 → 분산 처리 시스템' 순이다.

15. Section 071

시스템의 효율을 위하여 작업량을 일정 수준 모아두었다가 한꺼번에 처리하는 것은 일괄 처리 시스템이다.

16. Section 071

- 다중 프로그래밍은 여러 개의 작업을 효율적으로 병행하기 위해서 사용하는 것으로, 하나의 주기억장치를 공유, 분할하여 사용하게 되므로 메모리의 관리와 스케줄링, 보호는 반드시 이루어져야 하는 기능이다.
- Off-line 처리의 경우는 모아서 한 번에 처리하게 되므로, 병행처리를 하여 반응 속도를 빠르게 하는 것과는 관련이 적다.

17. Section 071

실시간 처리 시스템(Real Time Processing System)

- 데이터 발생 즉시, 또는 데이터 처리 요구가 있는 즉시 처리하여 결과를 산출하는 방식이다.
- 우주선 운행이나 레이더 추적기, 핵물리학 실험 및 데이터 수집, 전화교환장치의 제어, 은행의 On-line 업무 등 시간에 제한을 두고 수행되어야 하는 작업에 사용된다.

18. Section 075

- Loading : 보조기억장치에서 주기억장치로 프로그램을 가져오는 것
- Searching : 원하는 정보를 찾는 것
- Overapping : 주기억장치보다 큰 프로그램을 기억장치로 적재할 때 사용하는 방식 중 한 가지

19. Section 074

매크로와 부 프로그램의 비교

구분	매크로	부 프로그램
다른 이름	개방 서브루틴 (Opened Sub-routine)	폐쇄 서브루틴 (Closed Sub-routine)
처리 방식	주 프로그램의 매크로 호출 명령이 있는 위치마다 매크로 내용을 삽입하여 확장된 프로그램을 만들어 놓고 연속적으로 실행함	부 프로그램이 호출될 때마다 제어가 부 프로그램으로 넘어갔다가 다시 주 프로그램으로 복귀됨
특징	• 코딩이 간편해짐 • 부 프로그램은 매크로에 비해 프로그램 크기가 작아지고, 기억장소가 절약되지만 실행시간은 약간 느려짐	

20. Section 074

매크로는 프로그램 작성 시 한 프로그램 내에서 동일한 코드가 반복될 경우 반복되는 코드를 한 번만 작성하여 특정 이름으로 정의한 후 그 코드가 필요할 때마다 정의된 이름을 호출하여 사용하는 것으로, 반복되는 코딩을 간편하게 사용할 수 있다.

21. Section 074

- 매크로를 개방 서브루틴, 서브프로그램을 폐쇄 서브루틴이라고 한다.
- Function은 '함수'를 의미하며, 어떤 일을 수행하도록 지시하는 프로시저를 의미한다.

22. Section 074

④는 매크로, 즉 Opened 서브루틴에 대한 설명이다.

23. Section 073

- **매크로 1 Pass에서 사용되는 것** : 매크로 원본, PC, ST, MOT, POT, LT 등
- **매크로 2 Pass에서 사용되는 것** : 매크로 원시 프로그램 사본, PC, ST, LT, MOT, POT, 베이스 레지스터 테이블, Print Line, 목적 프로그램

24. Section 074

매크로 프로세서의 기능 : 매크로 정의 인식, 매크로 정의 저장, 매크로 호출 인식, 매크로 확장과 인수 치환

25. Section 074

매크로 프로세서는 컴파일러(Compiler)와도 같이 사용될 수 있다.

26. Section 069

- Control Program : 운영체제의 제어 프로그램, 즉 감시 프로그램, 작업 제어 프로그램, 데이터 관리 프로그램을 의미함
- Assembler : 어셈블리어로 작성된 원시 프로그램을 목적 프로그램으로 번역하는 프로그램
- Scheduler : 스케줄링(Scheduling)은 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업을 의미하며, 스케줄러는 이런 스케줄링 작업을 수행하는 것을 의미함

- Bench Mark Program : 컴퓨터 속도 등의 성능을 평가하기 위한 프로그램을 이용해 사용자가 준비한 기본 문제들을 처리하여 문제 해결의 결과로서 성능을 판정하는 방법

27. Section 072

- 언어 번역기로는 Compiler, Interpreter, Assembler가 있다.
- PASCAL, COBOL, FORTRAN은 프로그래밍 언어이다.

28. Section 070

운영체제는 사용자가 컴퓨터를 편리하고 효과적으로 사용하도록 환경을 제공하는 것이므로 일반적으로 사용되는 시스템에서 범용적으로 사용할 수 있도록 설계되어야 한다.

29. Section 073

- 어셈블러가 어셈블하는 방법은 원시 프로그램의 첫 줄에 있는 명령부터 차례로 의사 명령어 테이블을 검색하여 해당 내용이 있으면 의사 명령에 대한 실행 루틴의 시작 주소를 가지고 나와 그 루틴을 실행시켜서 어셈블에 필요한 정보를 어셈블러에게 제공하고, 없으면 집행 명령으로 인식하여 기계 명령어 테이블을 검색한다.
- 의사 명령어 테이블(POT, Psuedo Operation Table) : 의사 명령과 그 명령을 처리하는 실행 루틴이 있는 주소를 가지고 있는 테이블
- 기계 명령어 테이블(MOT, Machine Operation Table) : 어셈블리어의 실행 명령에 대응하는 기계어에 대한 정보를 가지고 있는 테이블

30. Section 073

Pass-1의 기능

- 기계 명령어의 길이 정의
- 위치 계수기(PC) 관리
- 기호들의 값을 ST에 기억
- 몇 가지 가연산자(의사 명령어) 처리
- 리터럴들을 LT에 기억

31. Section 073

어셈블리어는 프로그래밍과 오류 검증이 힘들고, 기계와 종속적인 프로그램을 만들게 되지만 메모리의 이용률이 좋은 프로그램을 작성할 수 있다.

32. Section 073

어셈블리어는 기계어를 기호화해 놓은 것으로, 작성한 CPU마다 어셈블리어가 다를 수 있다.

33. Section 075

재배치 로더(Relocation Loader)의 수행 순서 : 주기억장치 할당(Allocation) → 연결(Linking) → 재배치(Relocation) → 적재(Loading)

34. Section 075

- 로더는 기억장치 할당, 연결, 재배치, 적재의 기능을 하지만 최적화 기능을 수행하는 것은 아니다.
- 명령을 최적화하려면 명령어 최적화기 등을 이용한다.

35. Section 075

연결 편집기(Linkage Editor)는 언어 번역 프로그램에 의해 독립적으로 번역된 객체 모듈, 라이브러리, 또 다른 로드 모듈들을 연결하여 실행 가능한 로드 모듈을 만드는 시스템 소프트웨어이다.

36. Section 072, 075

원시 프로그램이 수행되기까지의 순서

원시 프로그램은 번역 프로그램을 통해 번역된 후 링킹을 통해 각 목적 프로그램이나 로드 모듈을 연결한 다음 적재기에 의해 주기억장치에 적재된다.

37. Section 071

- 시분할 처리 방식이란 CPU의 전체 사용 시간을 작은 시간 단위(Time Slice)로 쪼개어 각 단말기에 할당하여 그 시간량 동안만 처리하게 하는 방식으로, 매우 빠르게 사용자가 교대되어 돌아오기 때문에 사용자는 마치 자기 혼자만 사용하는 것처럼 느낀다.
- 시분할 처리 시스템의 단말장치로는 빠른 입 · 출력이 가능한 영상표시장치(모니터)가 적합하다.

39. Section 070

② 운영체제는 제어 프로그램과 처리 프로그램으로 구성되어 있다.

③ 자원 할당 측면에서 운영체제의 주된 기능은 프로세서, 기억장치, 입 · 출력장치, 파일 및 정보 등의 자원 관리 등이 있다.

④ 운영체제는 시스템 전체의 움직임을 감시, 감독 관리 및 지원하는 제어 프로그램과 주어진 문제를 응용 프로그램 감독하에 실제 데이터 처리를 하는 처리 프로그램으로 구성된다.

40. Section 072

- 프로그래밍 언어는 저급 언어와 고급 언어(컴파일러 언어)로 분

류하고, 다시 저급 언어는 기계어와 어셈블리어로 분류한다.

- 레지스터(Register) : CPU 내부에서 처리할 명령어나 연산의 중간 결과값 등을 일시적으로 기억하는 임시 기억장소

2장 정답 및 해설 — 프로세스 관리

1.①	2.④	3.③	4.④	5.②	6.②	7.②	8.④	9.③	10.④	11.③	12.②	13.③	14.②	15.③
16.④	17.④	18.③	19.①	20.③	21.①	22.④	23.③	24.①	25.③	26.③	27.①	28.③	29.①	30.②
31.①	32.③	33.④	34.③	35.①	36.②	37.②	38.①	39.④	40.③	41.③	42.③	43.③	44.①	45.②
46.①	47.③	48.①	49.①	50.④	51.③									

1. Section 076

프로세스 : PCB를 가진 프로그램, 실기억장치에 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

2. Section 076

프로세스가 생성될 때마다 해당 프로세스에 대한 고유의 PCB가 생성되고, 프로세스가 종료(소멸)되면 해당 PCB도 제거된다.

3. Section 076

- 워킹 셋(Working Set) : 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합
- PCB(Process Control Block, 프로세스 제어 블록) : 운영체제가 프로세스에 대한 중요한 정보를 저장해 놓는 곳
- 세그먼테이션(Segmentation) 기법 : 프로그램을 가변적인 크기로 나눈 단위를 세그먼트라 하며, 이 세그먼트를 블록으로 사용하는 기법

4. Section 076

PCB에는 프로세스의 현재 상태, 부모 및 자식 프로세스에 대한 포인터, 프로세스가 위치한 메모리에 대한 포인터, 할당된 자원에 대한 포인터, 프로세스의 고유 식별자, 스케줄링 및 프로세스의 우선순위, CPU 레지스터 정보 등이 저장되어 있다.

5. Section 081

은행원 알고리즘은 교착상태 해결 방법 중 회피 기법(Avoidance)에 해당한다.

6. Section 076

- 디스패치(Dispatch) : 준비 상태에서 실행 상태로 전이되는 것으로 CPU 스케줄러에 의해 수행됨
- 블록(Block) : 스스로 CPU 사용권을 양도하고 입 · 출력 처리가 완료될 때까지 대기함
- Timer Run Out : 프로세스가 자신에게 주어진 시간 할당량을 모두 사용한 경우 강제로 인터럽트에 의해 실행 상태에서 준비 상태로 전이됨
- Wake Up : 입 · 출력 지시 후 종료되었을 때 대기 상태에서 다시 준비 상태로 전이되는 것

7. Section 080

임의의 작업 순서를 지정하여 최소 평균 대기 시간을 구하려면 실행 시간이 가장 적은 것부터 차례로 수행하면 된다. 그러므로 수행 순서는 P3 → P2 → P1 순이다. 만약 최대 평균 대기 시간을 구하려면 실행 시간이 가장 긴 것부터 차례로 수행하면 된다.

8. Section 078

'(대기시간+서비스시간)/서비스시간'은 HRN 기법에서 우선순위를 계산하는 식이다.

9. Section 079

Round-Robin 스케줄링 방식은 적절한 응답시간이 보장되므로 일괄처리 시스템이 아닌 실시간 처리 시스템에 유용하다.

10. Section 077

스케줄링은 응답시간, 반환시간, 대기시간을 줄이고, CPU 이용률을 늘려서 많은 프로세스가 CPU나 자원을 효율적으로 사용하는 것을 목적으로 한다.

11. Section 076

할당되지 않은 주변 기기들의 상태 정보는 프로세스 제어 블록(PCB)에 포함되지 않고, 할당된 주변장치들에 대한 상태 정보가 포함된다.

12. Section 078

HRN 스케줄링 기법의 우선순위 계산식은 '(대기 시간 + 서비스 시간)/서비스 시간' 이며, 이 계산식으로 우선순위를 계산하면 (45 + 5) / 5 = 10이다.

15. Section 079

라운드 로빈(Round-Robin) 방식은 FCFS 기법과 같이 준비상태 큐에 먼저 들어온 프로세스가 먼저 CPU를 할당받지만 각 프로세스는 시간 할당량(Time Slice, Quantum) 동안만 실행한 후 실행이 완료되지 않으면 다음 프로세스에게 CPU를 넘겨주고 준비상태 큐의 가장 뒤로 배치된다. 작업 순서를 표로 표시하면 다음과 같다.

진행시간 0	3	6	9	12	14	17	20	23	24	27	30
작업순서	A	B	C	A	B	C	A	C	A	C	C
실행시간	3	3	3	3	2	3	3	3	1	3	3

※ ●는 각 작업이 종료되는 시점을 의미한다.
그러므로 작업 순서는 A, B, C, A, B, C, A, C, A, C, C 순이다.

16. Section 080

- 상호 배제 기법을 구현하기 위한 방법에는 소프트웨어적 구현과 하드웨어적 구현이 있는데, 소프트웨어적 구현 방법에는 데커(Deker) 알고리즘, 피터슨(Peterson) 알고리즘, Lamport의 빵집 알고리즘이 있고, 하드웨어적 구현 방법에는 Test_And_Set 기법과 Swap 명령어 기법이 있다.
- 은행원 알고리즘은 교착상태가 발생할 가능성을 배제하지 않고 교착 상태가 발생하면 적절히 피해나가는 교착 상태 회피 기법이다.

17. Section 077

바인딩 시간(Binding Time) : 프로그램에서 어떤 요소의 이름을 그것이 나타내는 실제의 대상물과 연결하는 시간으로, 스케줄링 알고리즘과는 무관함

18. Section 080

병행 프로세스는 한정된 컴퓨터 하드웨어나 자원을 공유하고, 동시에 작업을 수행하기 위해 사용하는 개념이다.

19. Section 079

- 대화형 시스템에 적당한 것은 선점 기법이므로, 선점 기법이 아닌 것을 찾아야 한다.
- 선점 스케줄링 기법 : RR, SRT, MFQ, MQ, 선점 Priority
- 비선점 스케줄링 기법 : FCFS, SJF, HRN, Deadline, Priority

20. Section 077

TAT(Turn Around Time)는 프로세서 스케줄링의 성능을 측정하기 위해 사용하는 것으로, 프로세스를 제출한 시간부터 실행이 완료될 때까지 걸리는 시간을 의미한다.

21. Section 079

평균 대기 시간 = 대기 시간의 총합 / 대기 작업의 수
= (0 + 32 + 20 + 23 + 40) / 5 = 23

22. Section 079

평균 반환 시간 계산

0	10	20	23	30	40	50	52	61
JOB 1	JOB 2	JOB 3	JOB 4	JOB 5	JOB 2	JOB 5	JOB 2	
10	10	3	7	10	10	2	9	

	JOB 1	JOB 2	JOB 3	JOB 4	JOB 5
반환 시간	10	61	23	30	52
도착 시간	0	10	15	16	20
반환 시간	10(10-0)	51(61-10)	8(23-15)	14(30-16)	32(52-20)

평균 반환 시간 : (10 + 51 + 8 + 14 + 32) / 5 = 115 / 5 = 23

※ ●는 각 작업이 종료되는 시점을 의미한다.

24. Section 079

RR에서 사용되는 일정한 시간량을 Time Slice 또는 Quantum이라고 한다.

25. Section 078, 079

- SRT : SJF 알고리즘을 선점 형태로 변경한 것으로, 현재 실행 중인 프로세스의 남은 시간과 준비상태 큐에 새로 도착한 프로세스의 실행 시간을 비교하여 가장 짧은 실행 시간을 요구하는 프로세스에게 CPU를 할당하는 기법
- SJF : 준비상태 큐에서 기다리고 있는 프로세스들 중에서 실행 시간이 가장 짧은 프로세스에 먼저 CPU를 할당하는 기법
- HRN : 실행 시간이 긴 프로세스에는 불리한 SJF 기법을 보완하기 위한 것으로, 서비스(실행) 시간과 대기 시간을 이용하는 기법

26. Section 078

SJF(Shortest Job First)는 준비상태 큐에서 기다리고 있는 프로세스들 중에서 실행 시간이 가장 짧은 프로세스에 먼저 CPU를 할당하는 기법으로, 평균 대기 시간이 가장 적게 걸린다.

27. Section 081

교착 상태는 둘 이상의 프로세스들이 자원을 점유한 상태(할당)에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 상태(할당된 자원이 해제되길 기다림)를 의미한다.

28. Section 079

- FCFS : 준비상태 큐에 도착한 순서에 따라 차례로 CPU를 할당하는 기법
- HRN : 실행 시간이 긴 프로세스에는 불리한 SJF 기법을 보완하기 위한 것으로, 서비스(실행) 시간과 대기 시간을 이용하는 기법
- 다단계 피드백 큐 : 특정 준비상태 큐에 들어간 프로세스가 다른 준비상태 큐로 이동할 수 없는 다단계 큐 기법을 준비상태 큐 사이를 이동할 수 있도록 개선한 기법으로, 각 준비상태 큐마다 할당된 시간을 부여하여 그 시간 동안 완료하지 못한 프로세스는 다음 단계의 준비상태 큐로 이동함

29. Section 076, 079, 081

시간 할당량(Time Slice)은 프로세스가 CPU를 할당받아 사용하도록 지정된 시간으로, 각 프로세스에 동일한 시간 할당량이 배당되어 수행 시간이 긴 프로세스는 시간 할당량 안에 작업을 완료하지 못할 수도 있다.

30. Section 078

- 교착상태(Dead Lock) : 상호 배제에 의해 나타나는 문제점으로, 둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상
- 무한 연기(Indefinite Postponement) : 우선순위 스케줄링 기법 등의 문제점으로 낮은 순위의 프로세스가 높은 순위의 프로세스에 의해 무한정 기다리는 상태
- 세마포어(Semaphore) : E.J.Dijkstra가 제안한 방법으로, P와 V라는 연산에 의해 동기화를 유지시키고, 상호 배제의 원리를 보장하는 알고리즘
- 임계 구역(Critical Section) : 다중 프로그래밍 운영체제에서 한 순간에 여러 개의 프로세스에 의해 공유되는 데이터 및 자원에 대하여 어느 한 시점에서는 하나의 프로세스에 의해서만 자원 또는 데이터가 사용되도록 지정된 공유 자원(영역)

31. Section 080

- 세마포어 : 두 개 이상의 프로세스를 한 시점에서는 동시에 처리할 수 없으므로 각 프로세스에 대한 처리 순서를 결정하는 동기화 기법 중의 하나임
- 교착상태 : 둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상으로 예방 기법, 회피 기법, 발견 기법, 회복 기법을 사용하여 교착상태를 해결할 수 있음

32. Section 080

- 상호 배제(Mutual Exclusion) : 특정 프로세스가 공유 자원을 사용하고 있을 경우 다른 프로세스가 해당 공유 자원을 사용하지 못하게 제어하는 기법
- 교착상태(Dead Lock) : 상호 배제에 의해 나타나는 문제점으로, 둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상
- 동기화(Synchronization) : 두 개 이상의 프로세스를 한 시점에서는 동시에 처리할 수 없으므로 각 프로세스에 대한 처리 순서를 결정하는 것

33. Section 080

상호 배제 문제를 해결하기 위한 것에는 데커 알고리즘, 피터슨 알고리즘, 빵집 알고리즘, 세마포어, 모니터 등이 있다.

34. Section 080

모니터 내의 공유 자원은 순서대로 접근할 수 있으며 동시에 접근이 불가능하다.

35. Section 081

- ②는 점유와 대기(Hold and Wait), ③은 상호 배제(Mutual Exclusion), ④는 환형 대기(Circular Wait)에 대한 설명이다.
- 교착상태의 조건 중 비선점(Non-preemption)은 다른 프로세스에 할당된 자원을 사용이 끝날 때까지 강제로 빼앗을 수 없음을 의미한다.

36. Section 081

교착상태를 방지(예방)하는 방법에는 상호 배제 조건의 부정, 점유와 대기 조건의 부정, 비선점 조건의 부정, 환형 대기 조건의 부정이 있다.

37. Section 081

교착상태의 해결 방법에는 교착상태 예방 기법(Prevention), 교착상태 회피 기법(Avoidance), 교착상태 발견 기법(Detection), 교착상태 회복 기법(Recovery)이 있다.

38. Section 081

- **교착상태 회피 기법(Avoidance)** : 교착상태가 발생할 가능성을 배제하지 않고, 교착 상태가 발생하면 적절히 피해나가는 방법
- **교착상태 발견 기법(Detection)** : 시스템에 교착상태가 발생했는지 점검하고, 교착상태에 있는 프로세스와 자원을 발견하는 것
- **교착상태 회복 기법(Recovery)** : 교착상태를 일으킨 프로세스를 종료하거나 교착상태의 프로세스에 할당된 자원을 선점하여 자원을 회복하는 것

40. Section 081

- 은행원 알고리즘에서 각 프로세스에게 자원을 할당하여 교착상태가 발생하지 않으며 모든 프로세스가 완료될 수 있는 상태를 안전 상태, 교착상태가 발생할 수 있는 상태를 불안전 상태라고 한다.
- 불안전 상태는 교착상태가 발생할 수 있는 상태이긴 하지만 불안전 상태가 곧 교착상태인 것은 아니다.

41. Section 081

- 교착상태의 회복 기법에서 자원 선점은 교착상태의 프로세스가 점유하고 있는 자원을 선점하여 다른 프로세스에게 할당하며 해당 프로세스를 일시 정지시키는 방법이다.
- **우선 선점할 프로세스** : 프로세스의 우선순위가 낮은 것, 처리된 진행 상태가 적은 프로세스, 사용되는 자원이 적은 프로세스

42. Section 081

세마포어는 프로세스 동기화를 위해 사용하는 기법이다.

43. Section 076

각각의 스레드는 자신만의 독립적인 제어 흐름을 갖는다.

45. Section 077

문맥 교환(Context Switching)은 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 작업이다. 그러므로 프로그램이 중단되고, 인터럽트 처리가 수행되기 전 문맥 교환이 발생된다.

46. Section 076

- **준비(Read) 상태** : 프로세스가 프로세서를 할당받기 위해 기다리고 있는 상태
- **대기(Wait), 보류, 블록(Block) 상태** : 프로세스에 입 · 출력 처리가 필요하면 현재 실행중인 프로세스가 중단되고, 입 · 출력 처리가 완료될 때까지 대기하고 있는 상태
- **조건 만족(Wake Up) 상태** : 입 · 출력 작업이 완료되어 프로세스가 대기 상태에서 준비 상태로 전이되는 과정

47. Section 076

운영체제의 프로세스 관리 기능 : 프로세스 스케줄링 및 동기화 관리 담당, 프로세스 생성과 제거, 시작과 정지, 메시지 전달, 교착상태 처리를 위한 기능 등

48. Section 079

RR 스케줄링에서 할당된 자원과 처리기의 소유권은 운영체제의 제어 권한이다.

49. Section 080

세마포어(Semaphore)에서는 초기치 연산(Semaphore Initialize, 세마포어의 초기치를 결정해 주는 연산), P 연산, V 연산이 사용된다.

50. Section 080

임계 구역내로 프로세스가 진입하는 것을 허용하는 것은 운영체제의 권한이다.

51. Section 080

병행 프로세스들의 고려사항

- 병행 프로세스들은 프로그래머가 외부적으로 스케줄링이 가능하도록 해야 한다.
- 공유 자원을 상호 배타적으로 사용해야 한다.
- 병행 프로세스들 사이에는 협력 또는 동기화가 이루어져야 한다.
- 두 프로세스 사이에는 통신이 가능해야 한다.
- 병행 프로세스들은 실행 순서에 관계없이 항상 일정한 실행 결과가 보장되어야 한다.
- 교착상태를 해결해야 하며 병행 프로세스들의 병렬 처리도(동시에 수행되는 정도)를 극대화해야 한다.

3장 정답 및 해설 — 기억장치 관리

1.①	2.③	3.④	4.④	5.④	6.②	7.③	8.①	9.④	10.④	11.②	12.①	13.③	14.④	15.④
16.②	17.②	18.①	19.③	20.③	21.②	22.④	23.①	24.③	25.④	26.③	27.④	28.②	29.①	30.④
31.③	32.②	33.①	34.④	35.①	36.③	37.②	38.④	39.②	40.①	41.②	42.②	43.③	44.③	45.③
46.③	47.④	48.②												

1. Section 083

- 정적(고정) 분할(Static Partition) : 프로그램을 할당하기 전에 운영체제가 주기억장치의 사용자 영역을 여러 개의 고정된 크기로 분할하고, 준비상태 큐에서 준비중인 프로그램을 각 영역에 적재하는 기법
- 동적(가변) 분할(Dynamic Partition) : 고정 분할 할당 기법의 단편화를 줄이기 위한 것으로, 미리 주기억장치를 분할하는 것이 아니라 프로그램을 주기억장치에 적재하면서 필요한 만큼의 크기로 영역을 분할하는 기법
- 세그먼테이션(Segmentation) 기법 : 가상기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나누어 주기억장치에 적재시키는 기법

2. Section 083

- 고정(정적) 분할 : 프로그램을 할당하기 전에 운영체제가 주기억장치의 사용자 영역을 여러 개의 고정된 크기로 분할하고, 준비상태 큐에서 준비중인 프로그램을 각 영역에 적재하여 수행하는 기법으로 실행할 프로그램의 크기를 미리 알고 있어야 함
- 가변(동적) 분할 : 고정 분할 할당 기법의 단편화를 줄이기 위한 것으로, 미리 주기억장치를 분할하는 것이 아니라 프로그램을 주기억장치에 적재하면서 필요한 만큼의 크기로 영역을 분할하는 기법이며 단편화가 발생될 수 있음

3. Section 083

- 인덱스 레지스터 : 주소 변경을 위해 사용되는 레지스터
- 상태 레지스터 : 연산중에 발생하는 여러 가지 상태값을 기억하는 레지스터
- 베이스 레지스터 : 주기억장치가 분할된 영역으로 나뉘어 관리될 때 프로그램이 한 영역에서 다른 영역으로 옮겨지더라도 명령의 주소 부분을 바꾸지 않고, 정상적으로 수행될 수 있도록 하기 위한 레지스터

4. Section 084

- 교착상태(Dead Lock)는 둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상을 의미한다.

- ①은 압축, ②는 재배치, ③은 통합이다.

5. Section 084

- 압축(Compaction) : 주기억장치 내에 분산되어 있는 단편화된 빈 공간을 결합하여 하나의 큰 가용 공간을 만드는 작업
- 통합(Coalescing) : 주기억장치 내에 인접해 있는 단편화된 빈 공간을 하나의 공간으로 통합하는 작업
- Page : 프로그램을 일정한 크기로 나눈 단위

6. Section 084, 085

페이징 기법에서는 외부 단편화가 존재할 수 없다.

7. Section 084

- 통합(Coalescing) : 주기억장치 내에 인접해 있는 단편화된 빈 공간을 하나의 공간으로 결합하는 작업
- 쓰레기 수집(Garbage Collection) : 주기억장치 내에 분산되어 있는 단편화된 빈 공간을 결합하여 하나의 큰 가용 공간으로 만드는 것(압축)
- 교체(Swapping) : 하나의 프로그램 전체를 주기억장치에 할당하여 사용하다 필요에 따라 다른 프로그램과 교체하는 기법

8. Section 082

②는 최적 적합(Best-Fit), ③은 최악 적합(Worst -Fit)에 대한 설명이다.

9. Section 083

- 스와핑(Swapping) : 하나의 프로그램 전체를 주기억장치에 할당하여 사용하다가 필요에 따라 다른 프로그램과 교체하는 기법으로 가상기억장치의 페이징 기법으로 발전되었음
- 압축(Compaction) : 주기억장치 내에 분산되어 있는 단편화된 공간을 결합하여 하나의 큰 가용 공간을 만드는 작업
- 재배치(Relocation) : 여러 위치에 분산된 단편화된 공간을 주기억장치의 한쪽 끝으로 옮겨서 큰 가용 공간을 만드는 압축 기법을 수행할 때 주어진 분할 영역의 주소를 새롭게 지정해주는 것

10. Section 082

Best Fit은 기억장치 배치(Placement) 전략 중 하나로, 내부 단편화 공간이 가장 적게 발생되는 영역에 배치하는 기법이다.

11. Section 082

최적 적합(Best Fit)은 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 작게 남기는 분할 영역에 배치시키는 방법이므로 12K의 프로그램은 14K 공백에, 18K의 프로그램은 26K의 공백에 할당된다.

12. Section 085

가상 메모리를 사용하면 CPU의 처리율이 높아지며, 주기억장치보다 큰 프로그램을 실행할 수 있고, 가상기억장치의 페이징 기법이나 세그먼테이션 기법을 사용하므로 오버레이 기법을 구현할 필요가 없다.

13. Section 083, 085

교체(Swapping) 기법은 하나의 프로그램 전체를 주기억장치에 할당하여 사용하다가 필요에 따라 다른 프로그램과 교체하는 기법이다.

14. Section 085

페이지 테이블은 기억장치에 저장되기 때문에 페이지 테이블을 사용함으로써 기억장소를 낭비하게 된다.

15. Section 085

- 가상기억장치의 구현 방법에는 페이징 기법과 세그먼테이션 기법이 있다.
- 스래싱(Thrashing) : 프로세스의 처리 시간보다 페이지 교체 시간이 더 많아지는 현상
- 집약(Compaction) : 주기억장치 내에 분산되어 있는 단편화된 빈 공간을 결합하여 하나의 큰 가용 공간을 만드는 작업
- 모니터(Monitor) : 동기화를 구현하기 위한 특수 프로그램 기법으로 특정 공유 자원을 프로세스에게 할당하는 데 필요한 데이터와 이 데이터를 처리하는 프로시저로 구성됨
- 오버레이(Overlay) : 주기억장치보다 큰 사용자 프로그램을 실행하기 위한 기법

16. Section 085

1MB ≒ 1,000KB이므로 8MB의 주기억장치는 8,000KB를 1KB로 나눈 만큼의 페이지를 갖게 된다.

17. Section 086

- LRU : 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법
- FIFO : 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법
- NUR : 참조 비트와 변형 비트를 이용하여 최근에 사용하지 않은 페이지를 교체하는 기법

18. Section 085

세그먼트는 사용자의 메모리 보는 관점을 지원하는 메모리 관리 방식이다.

19. Section 085

세그먼트 기법은 서로 관련 있는 내용을 묶어 놓은 블록으로, 페이징 기법에 비해 액세스 제어 기능이 강하며 세그먼트 매핑 시에 사용하는 SMT(Segment Map Table)에 접근 제어에 관한 내용이 포함되어 있다.

20. Section 085

매핑(Mapping)

논리적인 가상주소를 물리적인 실기억주소로 변환하는 것이다.

21. Section 086, 087

- 최적 적합(Best Fit)이나 최악 적합(Worst Fit)은 새로운 프로그램이나 데이터를 주기억장치의 어디에 위치시킬지를 결정하는 배치(Placement) 전략에 해당된다.
- FIFO : 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법
- Working Set : 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합으로, 주기억장치에 상주킴으로써 페이지 부재 및 페이지 교체 현상을 줄임
- PFF(Page Fault Frequency) : 페이지 부재가 일어나는 횟수

22. Section 087

스래싱(Thrashing)

- 스래싱은 프로세스의 처리 시간보다 페이지 교체 시간이 더 많아지는 현상이다.
- 다중 프로그래밍 시스템이나 가상기억장치를 사용하는 시스템에서 하나의 프로세스 수행 과정 중 자주 페이지 부재가 발생함으로 인해 나타나는 현상으로, 전체 시스템의 성능이 저하된다.
- 다중 프로그래밍의 정도가 높아짐에 따라 CPU의 이용율은 어느 특정 시점까지는 높아지지만, 다중 프로그래밍의 정도가 더욱 커지면 스래싱이 나타나고, CPU의 이용율은 급격히 감소하게 된다.
- CPU 이용율을 높이고 스래싱 현상을 방지하려면, 다중 프로그래밍의 정도를 적정 수준으로 유지하고 페이지 부재 빈도(Page Fault Frequency)를 조절하여 사용하며, Working Set을 유지해야 한다.

23. Section 086

- LRU(Least Recently Used) : 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법
- LFU(Least Frequently Used) : 사용 빈도가 가장 적은 페이지를 교체하는 기법
- NUR(Not Used Recently) : LRU와 비슷한 알고리즘이며, 최근에 사용하지 않은 페이지를 교체하기 위해 참조 비트와 변형 비트를 사용하는 기법

24. Section 086

- FIFO : 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법
- LRU : 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법
- NUR : 참조 비트와 변형 비트를 이용하여 최근에 사용하지 않은 페이지를 교체하는 기법

25. Section 086

LRU(Least Recently Used)

- 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법이다.
- 가장 많이 사용되는 기법으로, 각 페이지마다 계수기나 스택을 두어 현 시점에서 가장 오랫동안 사용하지 않은, 즉 가장 오래전에 사용된 페이지를 교체한다.
- 계수기나 스택과 같은 별도의 하드웨어가 필요하며, 시간적인 오버헤드가 발생된다.

26. Section 086

- LRU는 페이지 교체 알고리즘으로 페이지 부재 시 사용한다.
- 버퍼(Buffer) : 두 개의 장치가 데이터를 주고받을 때 두 장치 간의 속도 차이를 해결하기 위해 중간에 데이터를 임시로 저장해 두는 공간
- 캐시 메모리(Cache Memory) : 중앙처리장치와 주기억장치 사

이에 위치하여 컴퓨터의 처리 속도를 향상시키는 역할을 함

- 연관 메모리(Associative Memory) : 주소를 참조하여 데이터를 읽어오는 방식이 아니라 저장된 내용의 일부를 이용하여 기억장치에 접근하여 데이터를 읽어오는 기억장치

27. Section 087

Semaphore는 병행 프로세스의 동기화를 구현하기 위해 사용된다.

28. Section 087

- 시간 구역성이 이루어지는 기억장소 : 순환(Looping), 스택, 부 프로그램, Counting이나 집계에 사용되는 변수
- 공간 구역성이 이루어지는 기억장소 : 배열 순회(Array Traversal), 프로그램의 순차적인 코드 실행, 관련된 변수들을 근처에 선언하는 것

29. Section 086

- Belady의 모순 현상 : FIFO 페이지 교체 알고리즘에서 발생하는 것으로, 일반적으로 페이지 프레임의 수가 증가하면 페이지 부재율이 감소해야 하는데 페이지 프레임을 더 많이 할당했음에도 페이지 부재율이 증가하는 현상을 의미함
- 교착상태(DeadLock) : 둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상
- 구역성(Locality) : 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다는 이론
- 스래싱(Thrashing) : 프로세스의 처리 시간보다 페이지 교체 시간이 더 많아지는 현상

30. Section 087

워킹 셋(Working Set)은 자주 참조하는 페이지들의 집합을 의미하는 것으로, 스래싱이 발생되지 않게 하기 위해서 워킹 셋을 계속하여 변경해 주기 때문에 프로세스 실행중에 워킹 셋의 크기가 변경될 수 있다.

31. Section 087

내부 단편화는 분할된 영역이 할당될 프로그램의 크기보다 크기 때문에 프로그램이 할당된 후 사용되지 않고 남아 있는 빈 공간이므로 페이지 크기가 크면 클수록 내부 단편화는 증가할 수 있다.

32. Section 087

- 시간 구역성이 이루어지는 기억장소 : 순환(Looping), 스택, 부 프로그램, Counting이나 집계에 사용되는 변수
- 공간 구역성이 이루어지는 기억장소 : 배열 순회(Array Traversal), 프로그램의 순차적인 코드 실행, 관련된 변수들을 근처에 선언하는 것

33. Section 087

워킹 셋(Working Set)

- 실행 중인 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합을 의미한다.
- Denning이 제안한 것으로, 프로그램의 Locality 특징을 이용한다.
- 자주 참조되는 워킹 셋을 주기억장치에 상주시킴으로써 페이지 부재 및 페이지 교체 현상을 줄인다.

34. Section 088

SSTF 스케줄링은 탐색 거리가 짧은 요청을 먼저 서비스하는 방식으로, 현재 위치의 헤드에서 멀리 떨어진 요청은 기아 상태가 발생하고 응답 시간의 편차가 크기 때문에 대화형 시스템에는 사용이 부적합하다.

35. Section 088

SSTF(Shortest Seek Time First)

- 탐색 거리가 가장 짧은 트랙에 대한 요청을 먼저 서비스하는 기법이다.
- 현재 헤드 위치에서 가장 가까운 거리에 있는 트랙으로 헤드를 이동시킨다.
- FCFS보다 처리량이 많고, 평균 탐색 시간이 짧다.
- 처리량이 많은 일괄 처리 시스템에 유용하다.
- 현재 서비스한 트랙에서 가장 가까운 트랙에 대한 서비스 요청이 계속 발생하는 경우, 먼 거리의 트랙에 대한 서비스는 무한정 기다려야 하는 기아 상태가 발생할 수 있다.

36. Section 088

- SCAN : 현재 헤드의 위치에서 진행 방향이 결정되면 탐색 거리가 짧은 순서에 따라 그 방향의 모든 요청을 서비스하고, 끝까지 이동한 후 역방향의 요청 사항을 서비하는 기법
  ① 안쪽으로 진행할 경우 : 53 → 37 → 18 → 0 → 65 → 67 → 98 → 122 → 124 → 183
  ② 바깥쪽으로 진행할 경우 : 53 → 65 → 67 → 98 → 122 → 124 → 183 → 199 → 37 → 18

- C-SCAN : 헤드는 항상 바깥쪽에서 안쪽으로 이동하면서 가장 짧은 탐색 거리의 요청을 처리하며, 안쪽 요청이 더 이상 없을 경우 가장 바깥쪽 끝으로 헤드를 옮긴 후 안쪽 방향으로 요청을 처리하는 방식
  53 → 37 → 18 → 0 → 199 → 183 → 124 → 122 → 98 → 67 → 65 순으로 처리한다.

37. Section 088

SCAN

- 현재 헤드의 위치에서 진행 방향이 결정되면 탐색 거리가 짧은 순서에 따라 그 방향의 모든 요청을 서비스하고, 끝까지 이동한 후 역방향의 요청 사항을 서비스한다.
- 53 트랙의 위치에서 50 방향이나 59 방향으로 이동해야 하므로 ①, ②, ③ 중 하나를 선택해야 되며, 방향이 결정되면 그 방향의 모든 요청을 서비스해야 하므로 ②가 정답이다.

※ 안쪽과 바깥쪽 끝이 지정되지 않았으므로 표시된 트랙의 요청만 사용한다.

38. Section 088

- Seek Time : 헤드가 지정된 트랙 또는 실린더까지 접근하는 데 소요되는 시간
- Rotational Delay Time, Search Time : 헤드가 정해진 트랙 또는 실린더를 찾은 후 원판이 회전하여 원하는 섹터의 읽기/쓰기가 시작될 때 까지의 시간
- Transmission Time : 읽은 데이터를 주기억장치로 보내는 데 걸리는 시간

39. Section 085

Backing Store는 보조기억장치로, 여기서는 가상기억장치를 의미한다.

40. Section 088

FCFS 스케줄링 기법은 준비상태 큐에 도착한 순서에 따라 차례로 CPU를 할당하는 기법이므로 이동 순서는 '22 → 10 → 40 → 55 → 35 → 9 → 22' 순이며, 트랙10은 첫 번째로 서비스를 받게 된다.

41. Section 088

- FIFO : 디스크 대기 큐에 가장 먼저 들어온 트랙에 대한 요청을 먼저 서비스하는 기법
- SCAN : 현재 헤드의 위치에서 진행 방향이 결정되면 탐색 거리가 짧은 순서에 따라 그 방향의 모든 요청을 서비스하고, 끝까지 이동한 후 역방향의 요청 사항을 서비스하는 기법
- C-SCAN : 항상 바깥쪽에서 안쪽으로 움직이면서 가장 짧은 탐색 거리를 갖는 요청을 서비스하는 기법

42. Section 088

문제에 제시된 방향이 없으므로 다음과 같이 두 가지 방법으로 모두 수행한 후 올바른 답을 선택해야 한다.

① 0이 안쪽일 경우 : 50 → 40 → 0 → 200 → 180 → 150 → 130 → 120 → 100 → 80 → 70으로 총 이동 거리는 380

② 200이 안쪽일 경우 : 50 → 70 → 80 → 100 → 120 → 130 → 150 → 180 → 200 → 0 → 40으로 총 이동 거리는 390

43. Section 088

- SSTF : 탐색 거리가 가장 짧은 트랙에 대한 요청을 먼저 서비스하는 기법
- FCFS : 디스크 대기 큐에 가장 먼저 들어온 트랙에 대한 요청을 먼저 서비스하는 기법

44. Section 085

요구 페이징 기법은 페이징 기법의 하나이며, 페이지 요구가 있을 때에만 프로세스를 주기억장치에 적재하는 방식이므로 모든 페이지가 적재될 공간이 없어도 수행이 가능하다.

45. Section 082

- 기억장치 관리 정책 중 CPU에 의해 실행되거나 참조되기 위해서 주기억장치로 적재할 다음 프로그램이나 자료를 언제 가져올 것인가를 결정하는 정책은 반입 정책(Fetch Strategic)이다.
- **배치(Placement) 정책** : 새로 반입되는 프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정하는 전략
- **교체(Replacement) 정책** : 주기억장치의 모든 영역이 이미 사용 중인 상태에서 새로운 프로그램이나 데이터를 주기억장치에 배치하려고 할 때, 이미 사용되고 있는 영역 중에서 어느 영역을 교체하여 사용할 것인지를 결정하는 전략

46. Section 086

참조 페이지가 페이지 프레임에 없을 경우는 페이지 결함(부재)이 발생된다. FIFO는 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법이므로, 참조 페이지 b를 참조할 때에는 c를 제거한 후 b를 가져오게 된다. 그러므로 총 페이지 결함 발생 수는 5번이다.

요청된 페이지 번호	c	d	e	b	d	e	c
페이지 프레임	c	c	c	b	b	b	b
		d	d	d	d	d	c
			e	e	e	e	e
부재 발생	●	●	●	●			●

※ : 페이지 부재 발생

47. Section 085

압축(Compaction) 기법은 주기억장치 내에 분산되어 있는 단편화된 빈 공간을 결합하여 하나의 큰 가용 공간을 만드는 작업이다.

48. Section 088

SSTF 기법은 탐색 거리가 가장 짧은 트랙에 대한 요청을 먼저 서비스하는 기법이므로 현재 70 트랙에서 가장 가까운 45 트랙으로 이동하게 된다.

4장 정답 및 해설 — 정보 관리

1.③	2.④	3.①	4.④	5.④	6.④	7.①	8.③	9.③	10.①	11.④	12.①	13.②	14.④	15.③
16.①	17.②	18.④	19.③	20.③	21.④	22.③	23.③	24.④	25.③	26.②	27.④	28.①	29.④	30.②

1. Section 089

- 파일 시스템 : 파일의 저장, 액세스, 공유, 보호 등 보조기억장치에서의 파일을 총괄하는 파일 관리 기술
- 장치 상태 테이블(Device Status Table) : 시스템에 있는 장치의 상태를 나타내는 테이블

2. Section 089

파일 디스크립터의 내용 : 파일 이름, 보조기억장치에서의 파일 위치, 파일 구조, 보조기억장치의 유형, 액세스 제어 정보, 파일 유형, 생성 날짜와 시간, 제거 날짜와 시간, 최종 수정 날짜와 시간, 액세스한 횟수

3. Section 089

- 자료의 입 · 출력 단위는 비블록화 레코드와 블록화 레코드로 나누어지는데, 비블록화 레코드(Unblocking Record)는 하나의 논리 레코드가 하나의 물리 레코드를 이루는 형식이고, 블록화 레코드(Blocking Record)는 하나 이상의 논리 레코드가 모여서 하나의 물리 레코드를 이루는 형식이다.
- Block화 작업을 하면 한 번에 많은 데이터를 묶어서 전송하게 되므로 데이터 전송 속도 증가, 전송 횟수 감소 등으로 처리 효율이 높아진다.

4. Section 089

Read는 파일 내의 레코드를 단위로 한 작업이고, Open, Create, Copy는 파일을 단위로 한 작업이다.

5. Section 089

- 파일을 복사하는 명령은 Copy이다.
- Destroy는 파일을 디스크에서 삭제하는 명령이다.

6. Section 089

CPU 스케줄링은 운영체제의 핵심 기능이다.

8. Section 089

파일 디스크립터는 파일을 관리하기 위한 정보를 가지고 있는 제어 블록으로, 파일 시스템이 관리하므로 사용자가 직접 참조할 수 없다.

9. Section 092

링크를 이용한 기법에서 레코드를 검색할 경우 파일이 속한 레코드를 링크를 이용하여 순차적으로 검색해야 하므로, 탐색 시간이 오래 걸리고 직접 접근이 불가능하다.

10. Section 092

연속 할당

- 파일을 디스크의 연속된 기억공간에 할당하는 방법으로, 생성되는 파일 크기만큼의 공간이 있어야 한다.

- 논리적으로 연속된 레코드들이 물리적으로 인접한 공간에 저장되기 때문에 접근 시간이 빠르다.
- 파일 크기에 알맞은 연속 공간이 없을 경우 파일이 생성되지 않는다.
- 파일의 크기가 시간에 따라 변경될 경우 구현하기가 어렵다.
- 파일의 생성과 삭제가 반복되면서 단편화가 발생된다.

11. Section 091

- **선형 리스트** : 데이터가 연속적으로 구성되어 있는 형태로 삽입, 삭제 시 재배치해야 하므로 작업이 복잡함
- **연결 리스트** : 하나의 노드가 데이터와 포인터로 구성된 형태로 포인터는 다음 자료가 있는 주소를 가지고 있다. 삽입과 삭제가 용이하지만 항상 포인터에 의해 자료를 검색하므로 검색 시간이 길어짐

12. Section 089

Update는 데이터를 갱신하는 것이고, 데이터 삽입은 Insert이다.

13. Section 090

- **색인 순차 파일** : 순차 파일과 직접 파일에서 지원하는 편성 방법이 결합된 형태
- **분할 파일** : 하나의 파일을 여러 개의 파일로 분할하여 저장하는 형태
- **직접 파일** : 파일을 구성하는 레코드를 임의의 물리적 저장공간에 기록하는 형태

14. Section 090

순차 파일은 레코드를 논리적인 처리 순서에 따라 연속된 물리적 저장공간에 기록하는 것으로, 파일의 특정 레코드를 검색하려면 순차적으로 모든 파일을 비교하면서 검색해야 하므로 검색 효율이 낮다.

15. Section 090

색인 순차 파일은 인덱스를 이용하여 참조하기 때문에 직접 참조하는 직접 파일보다 접근 시간이 느리다.

16. Section 090

- **순차 파일** : 레코드를 논리적인 처리 순서에 따라 연속된 물리적 저장공간에 기록하는 방법
- **색인 순차 파일** : 순차 파일과 직접 파일에서 지원하는 편성 방법이 결합된 형태

17. Section 090

- 직접 접근은 해싱 함수를 이용하여 주소를 계산해야 하므로 가변 길이 레코드에서는 사용하기 어렵다.
- **가변 길이 레코드** : 레코드마다 길이가 다른 것을 의미함

18. Section 095

- 정보 보안 기법에는 암호화 기법, 여분 정보 삽입 기법, 인증 교환 기법, 디지털 서명 기법, 접근 제어 기법 등이 있다.
- 사용자 감시는 보안 위험을 감소시키는 기법 중 하나이다.

19. Section 093

- **파일의 명명(Naming)** : 접근하고자 하는 파일 이름을 모르는 사용자를 접근 대상에서 제외시키는 기법
- **암호(Password)** : 각 파일에 판독 암호와 기록 암호를 부여하여 제한된 사용자에게만 접근을 허용하는 기법
- **접근 제어(Access Control)** : 사용자에 따라 공유 데이터에 접근할 수 있는 권한을 제한하는 방법, 즉 각 파일마다 접근 목록을 두고, 접근 가능한 사용자와 동작을 기록한 후 이를 근거로 접근을 허용하는 기법

20. Section 093

영역 3의 프로세스는 파일 2에 대해 실행이 가능, 파일 3에 대해 읽기/쓰기가 가능하다.

21. Section 093

전역 테이블은 크기 때문에 일반적으로 가상기억장치에 보관한다.

22. Section 093

파일 3 = { (영역2, {W}), (영역3, {R, W}) }

23. Section 094

보안과 신뢰도는 상관관계가 적다.

24. Section 094

보안 유지 방법에는 외부 보안, 내부 보안, 사용자 인터페이스 보안의 세 가지가 있으며 외부 보안에는 시설 보안과 운용 보안이 있다.

25. Section 095

- Reed-Solomon Code는 데이터 압축에 사용되는 알고리즘이다.
- FEAL(Fast Encryption ALgorithm) : 1987년 일본의 NTT(Nippon Telegraph and Telephone)에서 개발한 것으로, 고속 동작이 가능하도록 설계되었으며, DES와 유사한 구조를 가지고 있음

26. Section 094

백업(Backup)
원본 데이터의 손실을 대비하여 중요한 데이터를 외부 저장장치에 하나 더 만들어 두는 기능

27. Section 095

공용키 시스템에서 암호화키는 공개되고, 해독키는 보안되어야 한다.

28. Section 095

대칭키 암호화 기법으로 대표적인 것은 DES, 비대칭키 암호화 기법으로 대표적인 것은 RSA이다.

29. Section 095

- **디지털 서명(Digital Signature Mechanism)** : 손으로 쓴 서명과 같이 고유의 전자 서명으로, 송신자가 전자 문서 송신 사실을 나중에 부인할 수 없도록 하고, 작성 내용이 송 · 수신 과정에서 변조된 사실이 없다는 것을 증명할 수 있는 기법
- **인증 교환 기법(Authentication Exchange Mechanism)** : 수신자가 메시지 전송 도중에 변경되지 않았음을 확인할 수 있으며, 메시지가 정당한 상대방으로부터 전달된 것임을 확인할 수 있는 기법
- **접근 제어 기법(Access Control Mechanism)** : 데이터에 접근이 허가된 자에게만 데이터 사용을 허용하는 정책을 강화하기 위해 사용하는 기법

30. Section 090

- **활성률(Activity)** : 프로그램이 한 번 수행되는 동안 처리되는 레코드 수의 백분율(수행 레코드 수/전체 레코드 수 × 100)
- **크기(Size)** : 파일에 저장되어 있는 정보량

5장 정답 및 해설 — 분산 운영체제

1.④	2.④	3.③	4.③	5.③	6.②	7.④	8.②	9.②	10.②	11.③	12.③	13.①	14.④	15.③
16.③	17.③	18.①	19.②	20.①	21.④	22.④	23.④	24.①	25.②	26.④	27.①	28.④	29.④	30.②
31.④	32.③	33.③	34.④	35.②	36.④	37.④	38.①	39.④	40.④	41.①				

1. Section 096

분리 실행 처리기 구조
주/종 처리기의 비대칭성을 보완하여 각 프로세서가 독자적인 운영체제를 갖도록 구성한 구조로서, 각 프로세서에게 할당된 작업은 해당 프로세서가 모두 처리해야 되기 때문에 한 프로세서에 일이 밀려도 다른 프로세서는 유휴 상태가 발생될 수 있다.

2. Section 096

크로스바 교환 행렬은 시분할 및 공유 버스 구조에서 버스의 수를 기억장치 수만큼 증가시켜 연결한 방식으로, 장치의 연결이 복잡하다.

3. Section 097

다중 처리기 운영체제의 구조에는 주/종 처리기, 분리 실행 처리기, 대칭적 처리기가 있다.

4. Section 098

분산 시스템은 통신선을 이용하여 자원을 공유할 수 있다.

5. Section 098

- 분산 시스템의 설계 목적에는 신뢰도 향상, 자원 공유, 연산(계산) 속도 향상, 통신 등이 있다.

- 분산 시스템은 통신을 이용하기 때문에 보안에 문제가 발생할 수 있으므로 보안 문제는 분산 시스템의 단점에 해당된다.

6. Section 099

분산 운영체제

- 하나의 운영체제가 모든 시스템 내의 자원을 관리하는 것이다.
- 원격에 있는 자원을 마치 지역 자원인 것과 같이 쉽게 접근하여 사용할 수 있는 방식으로, 사용이 편리하고 시스템 간 자원 공유가 용이하다.

7. Section 098

분산된 시스템에서 프로세서 간의 통신은 필수적이다.

8. Section 099

분산 처리 시스템의 위상에 따른 분류에는 완전 연결형, 부분 연결형, 트리형, 스타형, 다중 접근 버스형 등이 있다.

9. Section 099

신뢰성 : 시스템 안의 한 사이트가 고장나더라도 다른 사이트들이 정상적으로 작동하는지의 여부

10. Section 099

기본 비용은 초기의 설치 비용을 의미하는 것으로, 완전 연결은 모든 사이트를 직접 연결해야 하므로 기본 비용이 많이 든다.

11. Section 099

- **스타(Star)형 연결** : 모든 사이트가 하나의 중앙 사이트에 직접 연결되어 있고, 그 외의 다른 사이트와는 연결되어 있지 않은 구조
- **부분적 연결(Partially Connection)** : 시스템 내의 일부 사이트들 간에만 직접 연결된 형태로, 직접 연결되지 않은 사이트는 연결된 다른 사이트를 통해 통신하는 구조
- **완전 연결(Fully Connection)** : 각 사이트들이 시스템 내의 다른 모든 사이트들과 직접 연결된 구조

12. Section 099

하나의 상위(부모) 사이트와 연결된 형제 사이트 간에는 상위(부모) 사이트를 통해서만 통신할 수 있다.

13. Section 099

스타형 연결 구조는 모든 사이트가 하나의 중앙 사이트에 직접 연결되어 있고, 그 외의 다른 사이트와는 연결되어 있지 않은 구조이다.

14. Section 099

링형 연결은 목적 사이트에 데이터를 전달하기 위해서 링형 구조에 따라 순환해야 하므로 통신 시간이 많이 걸릴 수 있다.

15. Section 099

다중 접근 버스 연결형은 시스템 내의 모든 사이트들이 공유 버스에 연결된 구조로, 한 사이트의 고장은 다른 사이트의 작동에 영향을 주지 않는다.

16. Section 099

광대역 통신망은 사이트 간의 거리가 멀기 때문에 에러 발생률이 높다.

17. Section 099

분산 운영체제에서 제공하는 이주 방법에는 데이터 이주, 연산 이주, 프로세서 이주가 있다.

18. Section 098

- 분산 파일 체제를 설계할 때 중요한 항목은 투명성이다.
- **투명성** : 사용자가 분산된 시스템에 위치한 여러 자원을 사용할 때 각 자원의 위치 정보를 알지 못하고 마치 하나의 커다란 시스템을 사용하는 것처럼 인식하는 것

19. Section 098

- LoCUS : 대규모 분산 운영체제를 구축하기 위해 LA의 캘리포니아 대학에서 개발한 시스템으로, UNIX와 호환이 가능하며 기존 시스템과는 전혀 다른 커널을 사용함
- Andrew : 카네기 멜론(Carnegie-Mellon) 대학에서 개발한 시스템으로, 클라이언트 머신과 서버 머신으로 구분되어 확장성이 좋음
- UNIX : 시분할 시스템을 위해 설계된 대화식 운영체제로, C 언어로 작성되어 이식성이 높고 크기가 작으며 이해하기 쉬움

20. Section 096

- **크로스바 교환 행렬** : 시분할 및 공유 버스 방식에서 버스의 수를 기억장치 수만큼 증가시켜 연결한 방식
- **시분할 및 공유 버스** : 프로세서, 주변장치, 기억장치 등의 각종 장치들 간을 버스라는 단일 경로로 연결한 방식

- **다중 포트 기억장치** : 시분할 및 공유 버스 방식과 크로스바 교환 행렬 방식을 혼합한 형태의 방식

21. Section 096

$2^n = 256$이므로 n = 8이 된다.

22. Section 097

다중 처리 시스템은 강결합 형태를 취하므로 기억장소를 공유하며 하나의 운영체제에 의해서 관리된다.

23. Section 099

서버(Server)란 서비스를 제공하는 시스템을 의미한다.

24. Section 099

'요청'과 '결과 제공'에 따라 클라이언트는 서버가 될 수 있고, 서버는 다시 클라이언트가 될 수 있다. 또한 서버는 다른 서버의 클라이언트가 될 수도 있다.

25. Section 098

이주(Migration) 투명성 : 사용자나 응용 프로그램의 동작에 영향을 받지 않고 시스템 내에 있는 자원을 이동할 수 있도록 함

26. Section 098

분산 처리 시스템은 시스템 설계가 복잡하다.

27. Section 097

약결합(Loosely Coupled) 시스템은 각 시스템마다 독립적인 운영체제와 기억장치를 가지므로 독립적으로 작동할 수 있다.

28. Section 097, 098

분산 및 병렬 처리 시스템의 장점 : 다수의 사용자들 간의 통신 용이, 자원 공유, 데이터 공유, 작업 과부하 감소, 신뢰도 향상, 사용 가능도 향상, 연산 속도 향상 등

29. Section 097, 098

분산 처리 시스템은 각 구성 요소의 추가나 제거가 용이하다.

30. Section 099

비분산 시스템은 중앙 집중식 시스템을 의미하는 것으로, 분산 응용의 설계는 중앙 집중식 시스템의 설계보다 어렵다.

32. Section 099

부분 연결형은 시스템 내의 일부 사이트들 간에만 직접 연결된 형태로, 각 사이트들이 시스템 내의 다른 모든 사이트들과 직접 연결된 완전 연결형보다 신뢰성이 낮다.

33. Section 098

분산 처리 시스템의 투명성에는 접근 투명성, 병행 투명성, 이주 투명성, 위치 투명성, 복제 투명성, 성능 투명성, 규모 투명성, 고장 투명성 등이 있다.

34. Section 098

분산 네트워크는 운영 조직이 복잡하다.

35. Section 098

다단계 클라이언트/서버 시스템

- 서로 다른 많은 수의 DBMS를 연결할 수 있으므로 확장성이 좋다.
- 서버의 부하가 증가할 경우 부하를 분배할 수 있으므로 부하 균등이 가능하다.
- 구현하기가 어렵고 개발 비용이 상대적으로 많이 든다.

※ 클라이언트/서버 환경에서 미들웨어는 클라이언트에서 서버에 있는 응용 프로그램이나 자원을 효율적으로 사용하기 위해서 클라이언트와 서버 사이에 놓이게 되는 중간 소프트웨어를 통칭한다.

36. Section 099

클라이언트/서버(Client/Server) 시스템은 정보를 제공하는 서버와 정보를 요구하는 클라이언트로 구성되어 있으며, 클라이언트(워크스테이션, PC 등)와 서버가 하나의 작업을 분산 협동처리(Distributed Cooperative Processing)하는 방식으로 개방된 시스템이다.

37. Section 098

분산 시스템은 독립적인 처리 능력을 가진 컴퓨터 시스템을 통신망으로 연결한 시스템으로, 중앙 집중형 시스템에 비해 보안 정책이 복잡해진다.

38. Section 099

- **링형(Ring) = 환형** : 시스템 내의 각 사이트가 인접하는 다른 두 사이트와만 직접 연결된 구조
- **트리(Tree) 또는 계층(Hierarchy)형** : 분산 처리 시스템의 가장 대표적인 형태로, 각 사이트들이 트리 형태로 연결된 구조

- 망형 – 완전 연결(Fully Connection)형 : 각 사이트들이 시스템 내의 다른 모든 사이트들과 직접 연결된 구조

39. Section 098

분산 처리 시스템은 사용자가 분산된 시스템에 위치한 여러 자원을 사용할 때 각 자원의 위치 정보를 알지 못하고 마치 하나의 커다란 시스템을 사용하는 것처럼 인식하도록 하는 투명성이란 개념을 사용하므로, 각 컴퓨터들이 어느 곳에 위치하는지 몰라도 상관이 없다.

40. Section 097

주/종(Master/Slave) 처리기 시스템에서는 주프로세서가 고장나면 전체 시스템이 다운된다.

41. Section 099

②는 스타(Star)형, ③은 다중 접근 버스 연결(Multi Access Bus Connection)형, ④는 링(Ring)형에 대한 설명이다.

6장 정답 및 해설 — 운영체제의 실제

1.④	2.④	3.④	4.③	5.③	6.②	7.②	8.②	9.③	10.③	11.②	12.④	13.④	14.③	15.④
16.④	17.①	18.③	19.③	20.①	21.④	22.②	23.③	24.①	25.④	26.③	27.④	28.①	29.④	30.④
31.①	32.③	33.②	34.①	35.③	36.③	37.①	38.①	39.③	40.①	41.②	42.③	43.③	44.③	45.③
46.③														

1. Section 100

플러그 앤 플레이(PnP, Plug & Play)는 컴퓨터 시스템에 하드웨어를 설치했을 때, 해당 하드웨어를 사용하는 데 필요한 시스템 환경을 운영체제가 자동으로 구성해 주는 것으로, UNIX에서는 PnP 기능을 제공하지 않는다.

2. Section 100

UNIX의 특징

- 대부분 C 언어로 작성되어 이식성이 높으며 장치, 프로세스 간의 호환성이 높다.
- 크기가 작고 이해하기가 쉽다.
- Multi-User(다중 사용자), Multi-Tasking(다중 작업)을 지원한다.
- 많은 네트워킹 기능을 제공하므로 통신망 관리용 운영체제로 적합하다.
- 트리 구조의 파일 시스템을 갖는다.

3. Section 100

- UNIX에서 명령어 해석기 기능을 수행하는 것은 Shell(쉘)이다.
- 커널(Kernel)은 UNIX의 핵심적인 부분으로, 프로세스(CPU 스케줄링) 관리, 기억장치 관리, 파일 관리, 입 · 출력 관리, 프로세스 간 통신, 데이터 전송 및 변환 등 여러 가지 기능을 수행한다.

4. Section 100

- 쉘(Shell) : 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기
- 유틸리티(Utility) : 일반 사용자가 작성한 응용 프로그램을 처리하는 데 사용함

5. Section 100

- ③은 커널(Kernel)에 대한 설명이다.
- 쉘(Shell)은 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기이다.

6. Section 100

- 커널(Kernel) : UNIX의 가장 핵심적인 부분으로, 하드웨어를 보호하고 프로그램과 하드웨어 간의 인터페이스 역할을 담당함
- 시스템 호출(System Call) : 프로세스가 커널에 접근하기 위한 인터페이스를 제공하는 명령어

7. Section 100

pipe는 프로세스 간에 단방향 통신 채널을 설정하여 프로세스 간에 통신이 가능하도록 하는 명령으로, 전송한 데이터는 FIFO 방식으로 상대에게 전달된다.

8. Section 100

- 도스에서 COMMAND.COM 파일은 명령을 해독하여 실행하는 명령어 해석기이므로 UNIX에서 쉘(Shell)과 같은 기능을 수행한다.
- 데몬(Demon) : 백그라운드 상태에서 실행하는 프로그램을 의미함
- 커널(Kernel) : UNIX의 핵심적인 부분으로, 프로세스(CPU 스케줄링) 관리, 기억장치 관리, 파일 관리, 입 · 출력 관리, 프로세스 간 통신, 데이터 전송 및 변환 등 여러 가지 기능을 수행함
- 유틸리티(Utility) : 일반 사용자가 작성한 응용 프로그램을 처리하는 데 사용함

11. Section 100

- exec : 프로세스 수행
- creat : 파일 생성
- mkfs : 파일 시스템 생성

12. Section 100

- 커널은 프로세스(CPU 스케줄링) 관리, 기억장치 관리, 파일 관리, 입 · 출력 관리, 프로세스 간 통신, 데이터 전송 및 변환 등의 기능을 수행한다.
- 데이터베이스를 관리하는 것은 DBMS(Data Base Management System)라고 한다.

13. Section 101

```
ls - l - rwxr - xr - x aaa bbb 98 Aug 7 19 : 16 ccc
```

- ls - l : ls는 파일의 목록을 나타내는 명령이고, -l은 파일 목록을 나타내되 허가 정보, 소유권, 변경 날짜 등과 같은 자세한 정보를 포함하여 나타내라는 옵션이다.
- 파일 정보 구조

①	파일 유형	-
②	소유권 권한	rwx
	그룹 권한	r - x
	전체 권한	r - x
③	링크 수	생략
④	그룹명	bbb
⑤	파일 크기	생략
⑥	최종 변경 일시	98 Aug 7 19 : 16
⑦	파일명	ccc

① 파일 유형 : 일반 파일은 '-', 디렉터리 파일은 'd', 입 · 출력 파일은 'c'로 표시한다.
② 소유권 · 그룹 · 전체 권한 : 소유권 권한, 그룹 권한, 전체 권한을 의미하며, 각 권한은 세 개의 문자로 표시한다.

r(read, 읽기 가능)	파일을 읽을 수만 있음
w(write, 쓰기 가능)	파일의 내용을 읽고, 수정할 수 있음
x(excute, 실행 가능)	파일을 실행시킬 수 있음

※ rwxr-xr-x : 소유자는 읽기, 쓰기, 실행이 가능, 그룹은 읽기, 실행만 가능, 전체 사용자는 읽기, 실행만 가능함
③ 링크 수 : 링크된 수를 표시한다.
④ 소유자명과 그룹명을 표시한다.
⑤ 파일 크기를 표시한다.
⑥ 최종적으로 변경된 일시를 표시한다.
⑦ 파일명을 표시한다.

14. Section 100

```
- rwxr - x - - x
```

- - : 파일 유형은 일반 파일이다.
- rwx : 소유자는 읽기, 쓰기, 실행이 가능하다.
- r - x : 그룹은 읽기, 실행이 가능하다.
- - - x : 전체 사용자는 실행만 가능하다.

15. Section 101

- | : 파이프 라인을 생성한다.
- [ ] : [a-z]와 같이 [ ] 사이에 문자의 범위를 지정한다.
- 〉〉 : 표준 입 · 출력장치의 방향을 바꾸어 주는 특수문자로, '〉〉' 다음에 파일이 존재하면 파일 끝에 데이터를 추가로 저장하고, 파일이 존재하지 않으면 새로운 파일을 만들어 데이터를 저장하라는 의미이다.

17. Section 101

UNIX는 파일과 디렉터리를 구분하지 않고 동일하게 취급한다.

18. Section 101

I-node에 포함된 정보 : 파일 소유자의 사용자 번호(UID) 및 그룹 번호(GID), 파일 크기, 파일 타입(일반 · 디렉터리 · 특수 파일 등), 생성 시기, 최종 변경 시기, 최근 사용 시기, 파일의 보호 권한, 파일 링크 수(파일 사용 횟수), 데이터가 저장된 블록의 시작 주소 등

19. Section 101

- 부트 블록 : 부팅 시 필요한 코드를 저장하고 있는 블록
- 슈퍼 블록 : 전체 파일 시스템에 대한 정보를 저장하고 있는 블록
- I-node 블록(Index-node) : 각 파일이나 디렉터리에 대한 모든 정보를 저장하고 있는 블록으로, 파일 소유자의 사용자 번호(UID) 및 그룹 번호(GID), 파일 크기, 파일 type(일반 · 디렉터리 · 특수 파일 등), 생성 시기, 최종 변경 시기, 최근 사용 시기, 파일의 보호 권한, 파일 링크 수, 데이터가 저장된 블록의 시작 주소 등이 저장되어 있음
- 데이터 블록 : 디렉터리별로 디렉터리 엔트리와 실제 파일에 대한 데이터가 저장된 블록

21. Section 101

I-node에 포함된 정보 : 파일 소유자의 사용자 번호 및 그룹 번호, 파일 크기, 파일 타입, 생성 시기, 최종 변경 시기, 최근 사용 시기, 파일의 보호 권한, 파일 링크 수, 데이터가 저장된 블록의 시작 주소 등

22. Section 101

UNIX 파일 시스템의 구조는 부트 블록(Boot Block), 슈퍼 블록(Super Block), I-node(Index-node) 블록, 데이터 블록으로 구성된다.

23. Section 101

- creat : 파일을 생성함
- ls : 현재 디렉터리 내의 파일 목록을 확인함
- mkfs : 파일 시스템을 생성함
- cat : 파일 내용을 화면에 표시함

24. Section 101

- mkfs : 파일 시스템을 생성함
- fsck : 파일 시스템을 검사하고 보수함
- mknod : 특수 파일을 생성함

25. Section 101

- mkfs : 파일 시스템을 생성함
- kill : 프로세스를 제거함
- mknod : 특수 파일을 생성함

26. Section 102

Windows 98의 특징

- Single-User
- 선점형 Multi-Tasking
- GUI(그래픽 사용자 인터페이스)
- 32Bit 파일 시스템 사용
- 공백을 포함하여 255자까지의 긴 파일 이름 사용 가능
- PnP(Plug and Play, 자동 감지 기능) 사용
- 네트워크 연결을 용이하게 하는 기능 지원
- 사운드, 동화상 등의 멀티미디어를 쉽게 사용할 수 있는 기능 지원
- CD-ROM의 Auto Display : CD-ROM 드라이브에 CD를 삽입하면 Autorun.inf 파일에 의해 자동 수행

27. Section 102

- 서류 가방 : 하나의 파일을 두 대 이상의 컴퓨터에서 옮겨가며 작업할 때 모든 컴퓨터에서 동일한 내용의 파일을 가지고 작업할 수 있게 관리함
- 내 컴퓨터 : 컴퓨터에 설치된 디스크 드라이브, 프린터, 제어판, 폴더, 파일 등을 표시 · 관리함
- 바탕 화면 : Windows 98의 기본적인 작업 공간을 의미함

28. Section 102

〈시작〉 단추를 클릭할 경우 나타나는 메뉴에는 프로그램, 즐겨찾기, 문서, 설정, 찾기, 도움말, 실행 등이 있다.

30. Section 103

①은 Autoexec.bat 파일, ②는 Config.sys 파일, ③은 Command.com 파일에 대한 설명이다.

31. Section 103

- 실행 가능한 파일을 나타내는 확장자 : COM, BAT, EXE
- OBJ : 번역 프로그램으로 번역된 목적 파일
- ASM : 어셈블리어를 이용하여 작성된 파일
- HWP : 혼글 파일

32. Section 103

한 화면씩 볼 경우는 more 기능을 사용하며, 내용을 출력할 때는 type 명령을 사용한다. 즉 type 명령 실행 후 결과를 파이프 기능을 이용하여 more에 넘겨주는 형태인 C:\>type text|more를 사용한다.

34. Section 103

- HIMEM.SYS : 멀티태스킹을 위해 연장 메모리를 사용할 수 있게 하는 파일
- EMM386.EXE : 386 이상의 기종에서 연장 메모리의 일부를 확장 메모리로 사용할 수 있게 하며, 상위 메모리를 관리하는 파일
- DRIVER.SYS : 드라이버를 제공하는 파일

※ 연장 메모리(Extended Memory) : CPU가 접근할 수 있는 1,024K 이상의 메모리 영역

35. Section 103

- DEL은 MS-DOS에서의 파일 삭제 명령이고, rmdir은 UNIX에서의 디렉터리 삭제 명령이다.
- 파일 삭제 명령은 MS-DOS에서는 DEL, UNIX에서는 rm이다.
- 디렉터리 삭제 명령은 MS-DOS에서는 RD, UNIX에서는 rmdir이다.

36. Section 101

사용 허가에 대해서는 r(읽기 가능), w(쓰기 가능), x(실행 가능) 중 하나를 지정할 수 있다.

37. Section 102

Windows 98은 선점형 멀티태스킹을 사용하므로 특정한 응용 프로그램에 문제가 발생되면 해당 프로그램만 종료시킬 수 있다.

38. Section 103

TYPE은 파일의 내용을 표시하는 내부 명령어이다.

39. Section 103

BAK 확장자는 백업 파일의 확장자이다.

40. Section 103

- CHKDSK : 디스크 상태를 점검함
- SYS : 시스템 파일을 복사함
- ATTRIB : 파일의 속성을 변경함

41. Section 101

- 부트 블록 : 부팅 시 필요한 코드를 저장하고 있는 블록
- I-node 블록(Index-node) : 각 파일이나 디렉터리에 대한 모든 정보를 저장하고 있는 블록

42. Section 101

파일 모드 값 640을 이진수로 변환하면 아래와 같이 표시할 수 있다.

```
 6    4    0
 ↓    ↓    ↓
110  100  000
```

첫 번째 값(6)은 소유자의 권한, 두 번째 값(4)은 그룹의 권한, 세 번째 값(0)은 전체 사용자의 권한을 나타낸다. 또한 각 권한은 읽기 가능, 쓰기 가능, 실행 가능 여부를 나타낸다. 그러므로 소유자는 읽기 · 쓰기(6 → 1(읽기)1(쓰기)0(실행))가 가능하고, 그룹은 읽기만 가능(4 → 100)하며, 전체 사용자는 읽기 · 쓰기 · 실행이 모두 불가능(0 → 000)하다. 1은 권한이 있음을, 0은 권한이 없음을 의미한다.

43. Section 100

①, ②, ④는 루트(Root)로부터 목적지까지의 경로를 나타내는 절대경로명에 해당하고, 다 번은 현재 위치한 디렉터리를 기준으로 한 목적지까지의 경로를 나타내는 상대경로명에 해당한다. ①, ②, ④의 처음 '/'는 루트 디렉터리를 나타내며, ③의 ..는 현재 디렉터리의 상위 디렉터리를 의미한다.

UNIX 시스템의 경로(Path)

- 원하는 디렉터리로 이동하기 위해 목적지까지의 디렉터리를 지정하는 것을 말한다.
- 경로명에는 절대경로명과 상대경로명이 있다.
- 루트(Root) 디렉터리는 처음에 '/'로 표시하고, 현재 디렉터리는 '.', 상위 디렉터리는 '..', 각 디렉터리의 구분은 "/"로 표시한다.
- 절대 경로명 : 루트(Root)로부터 목적지까지의 경로를 의미함
- 상대 경로명 : 현재 위치한 디렉터리를 기준으로 한 목적지까지의 경로를 의미함

44. Section 101

- open : 파일을 사용할 수 있는 상태로 준비시킴
- finger : 사용자 정보를 표시함

- chown : 소유자를 변경함

45. Section 101

UNIX 파일 시스템의 구조

- **부트 블록** : 부팅 시 필요한 코드를 저장하고 있는 블록
- **슈퍼 블록** : 전체 파일 시스템에 대한 정보를 저장하고 있는 블록
- **I-node 블록(Index-node)** : 각 파일이나 디렉터리에 대한 모든 정보를 저장하고 있는 블록으로, 파일 소유자의 사용자 번호(UID) 및 그룹 번호(GID), 파일 크기, 파일 type(일반 · 디렉토리 · 특수 파일 등), 생성 시기, 최종 변경 시기, 최근 사용 시기, 파일의 보호 권한, 파일 링크 수, 데이터가 저장된 블록의 시작 주소 등이 저장되어 있음
- **데이터 블록** : 디렉터리별로 디렉터리 엔트리와 실제 파일에 대한 데이터가 저장된 블록

46. Section 100

명령어 해독은 쉘(Shell)의 기능이다.

4과목 시스템 분석 및 설계

1장 정답 및 해설 — 시스템의 개요

1.③	2.③	3.④	4.③	5.②	6.③	7.④	8.③	9.①	10.④	11.④	12.②	13.①	14.③	15.④
16.②	17.②	18.①	19.③	20.②	21.④	22.①	23.③	24.④	25.②	26.③	27.④	28.①	29.③	

1. Section 104

- **목적성** : 서로 다른 기능을 가지고 있는 시스템의 각 구성 요소들이 어떤 하나의 공통된 최종 목표에 도달하고자 하는 특성
- **제어성** : 시스템이 오류 없이 그 기능을 발휘하기 위하여 정해진 규정이나 한계, 또는 궤도로부터 이탈되는 사태나 현상의 발생을 사전에 감지하여 그것을 바르게 수정해 가는 특성
- **자동성** : 어떤 조건이나 상황의 변화에 대응하여 스스로 대처할 수 있는 특성
- **종합성** : 항상 다른 관련 시스템과 상호 의존관계로 통합되는 특성

3. Section 104

시스템의 기본 요소 : 입력(Input), 처리(Process), 출력(Output), 제어(Control), 피드백(FeedBack)

4. Section 104

시스템의 기본 요소

- **입력(Input)** : 처리할 데이터, 처리 방법, 처리 조건을 시스템에 투입하는 것
- **처리(Process)** : 입력된 데이터를 처리 방법과 조건에 따라 변환 · 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

6. Section 104

시스템 분석가(SA)의 기본 조건

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

7. Section 104

시스템 분석가는 객관적인 입장에서 시스템을 분석해야 한다.

8. Section 105

- 시스템 운용은 테스트와 유지보수 사이에 들어간다.
- **시스템 개발 생명 주기(SDLC)** : 시스템 조사 → 시스템 분석 → 시스템 설계 → 시스템 구현 → 테스트 → 시스템 운용 → 유지보수

9. Section 105

시스템 개발 순서 : 목표 설정 → 현상 조사 분석 → 신시스템 설계 → 신시스템 이행 → 신시스템 실행 후 평가

10. Section 105

시스템 설계 순서 : 문제 제기 → 기업 환경 조사 → 현장 조사 → 기능 분석 → 기본 설계 → 상세 설계

11. Section 105

기본 설계 과정 : 코드 설계 → 출력 설계 → 입력 설계 → 파일 설계 → 프로세스 설계

12. Section 105

예비 조사(요구 사항 조사)
- 시스템 개발의 타당성을 조사하고 검토하는 단계
- 대상 시스템의 업무 처리 범위 및 기능을 조사한다.
- 기술적 타당성, 운영적 타당성, 경제적 타당성을 조사한다.
- 주로 관리자를 대상으로 한다.

13. Section 105

예비 설계 단계에서는 적용 업무가 아니라 적용 기법을 선정한다. 적용 업무의 선정은 기능 분석 단계에서 수행한다.

14. Section 105

시스템 분석 단계 : 기능 분석, 예비 설계, 비용 효과 분석

15. Section 105

비용 효과 분석에서의 작업
- 대체안의 작성
- 전산화 비용에 대한 견적
- 전산화 효과의 계수적 파악
- 대체안에 대한 평가

16. Section 105
- **시스템 조사** : 현행 시스템의 상태와 문제점을 파악하고, 해결 방안을 제안하는 단계
- **시스템 설계** : 시스템 분석에 의해 정의된 시스템 요구 사항 명세서를 토대로 하여 새로운 시스템을 구체화하는 단계
- **시스템 구현** : 설계 단계에서 산출된 설계 사양에 따라 프로그래밍 언어를 이용하여 원시 코드를 작성하는 단계

17. Section 105

제약 조건의 정리는 예비 설계 단계에서 수행한다.

기능 분석에서의 작업
- 현행 시스템에 대한 이해
- 요구와 기대 파악
- 문제점 및 특성 파악
- 환경 특성에 대한 파악
- 목적과 기능의 명확화
- 적용 업무의 선정

18. Section 105

적용 업무의 선정은 기능 분석 단계에서 수행한다.

19. Section 105

시스템 조사에는 예비 조사와 기초 조사가 있다.

20. Section 105

기본 설계 순서 : 코드 설계 → 출력 설계 → 입력 설계 → 파일 설계 → 프로세스 설계

21. Section 105

자료 수집 방법
- **면접법** : 작업과 관계되는 담당자를 직접 만나서 조사하는 방법(인터뷰 조사)
- **설문지법** : 질문 대상을 표본 추출하거나 전체를 대상으로 설문지를 이용하여 조사하는 방법(앙케트 조사)
- **현장 관찰법** : 실제 작업 현장에서 작업 처리 절차나 수행 과정을 직접 조사하는 방법
- **자료 조사법** : 시스템 개발에 필요한 서류나 문서 등을 수집하여 조사하는 방법

22. Section 105

시스템 개발 순서 : 예비 조사 → 업무 분석과 요구 정의 → 시스템 설계 → 프로그램 설계 → 프로그래밍 → 테스트와 디버깅

23. Section 105

시스템 구현
- 설계 단계에서 산출된 설계 사양에 따라 프로그래밍 언어를 이용하여 원시 코드를 작성하는 단계로 프로그래밍(Programming) 또는 코딩(Coding)이라고 한다.
- 시스템 구현 단계에서는 각 모듈의 코딩과 함께 디버깅이 이루어지고, 그 결과를 검증하는 단위 테스트를 실시한다.

24. Section 105

시스템 테스트의 종류
- **통합 테스트** : 모듈들을 통합하고, 시스템의 전반적인 흐름에 관한 오류를 점검하는 단계
- **시스템 테스트** : 구현 시스템이 원래의 요구 사항들을 만족시키는가를 알아보는 단계
- **인수 테스트** : 사용자의 요구 사항을 만족시키는가를 확인하기 위해 사용자에게 인수하는 과정에서 하는 테스트

25. Section 105

유지보수 단계는 시스템 개발 단계 중 가장 많은 비용이 투입되는 단계이다.

26. Section 104

- **면접 조사** : 작업과 관계되는 담당자를 직접 만나서 조사하는 방법(인터뷰 조사)
- **질문서(설문지, 앙케트 조사)** : 질문 대상을 표본 추출하거나 전체를 대상으로 설문지를 이용하여 조사하는 방법
- **현장 관찰 조사** : 실제 작업 현장에서 작업 처리 절차나 수행 과정을 직접 조사하는 방법
- **자료 조사** : 시스템 개발에 필요한 서류나 문서 등을 수집하여 조사하는 방법

27. Section 104

시스템은 공통된 목적을 달성하기 위하여 상호 관련이 없는 구성 요소가 아니라 여러 가지 상호 관련된 요소들을 유기적으로 결합한 것이다.

28. Section 104

시스템 분석가는 시스템을 사용하는 사용자들의 요구 사항을 파악하고 해결책을 마련하는 사람으로, 사람을 중심으로 분석을 수행해야 한다.

29. Section 105

①은 분석 단계, ②는 조사 단계, ④는 테스트 단계에 대한 설명이다.

2장 정답 및 해설 — 코드 설계

1.③	2.②	3.②	4.③	5.①	6.④	7.②	8.④	9.①	10.④	11.②	12.③	13.②	14.①	15.④
16.④	17.④	18.②	19.①	20.①	21.②	22.②	23.②	24.②	25.④	26.②	27.④	28.②	29.③	30.④
31.④														

1. Section 106

코드는 자료 처리의 효율을 높이기 위해 단순화시킨 기호이다. 자료의 내용을 쉽게 연상할 수 있는 코드가 있기는 하지만 자료의 내용을 쉽게 볼 수 있게 하는 것은 아니다.

2. Section 106

코드는 컴퓨터를 이용한 자료 처리 시 자료의 분류, 조합, 집계, 추출 등을 쉽게 하기 위해 사용하는 기호이다.

3. Section 106

코드의 기능

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

4. Section 106

코드는 자료를 간소화, 단순화시키는 기능을 한다.

5. Section 106

코드 설계 순서 : 코드 대상 선정 → 사용 범위와 사용 기간의 결정 → 코드 설계와 체크 → 코드의 번역, 코드표 작성 → 코드 파일 작성 → 코드 파일과 코드표 관리

6. Section 106

코드를 설계할 때는 다른 사람도 연상하기 쉽도록 코드화 대상 항목의 뜻과 의미가 내포되어야 한다.

7. Section 106

코드는 단순성이 있어야 한다.

8. Section 106

한 개인을 나타내는 코드는 하나만 부여해야 관리하기가 쉬워진다.

9. Section 106

코드 부여 대상의 추가, 수정, 삭제를 쉽게하는 코드의 성질은 확장성이다.

10. Section 107

- 그룹 분류식 코드(Group Classification Code) : 코드화 대상 항목을 일정 기준에 따라 대분류, 중분류, 소분류 등으로 구분하고, 각 그룹 안에서 일련 번호를 부여하는 방법
- 구분 코드(Block Code) : 코드화 대상 항목 중에서 공통성이 있는 것끼리 블록으로 구분하고, 각 블록 내에서 일련 번호를 부여하는 방법
- 10진 코드(Decimal Code) : 코드화 대상 항목을 0~9까지 10진 분할하고, 다시 그 각각에 대하여 10진 분할하는 방법을 필요한 만큼 반복함

11. Section 107

코드화 대상 항목을 학과, 입학년도, 번호에 따라 구분하고, 마지막 자리에 일련번호를 부여하는 방법을 사용하였으므로 그룹 분류식 코드이다.

12. Section 107

그룹 분류식 코드(Group Classification Code)는 코드화 대상 항목을 일정 기준에 따라 대분류, 중분류, 소분류 등으로 구분하고, 각 그룹 안에서 일련번호를 부여하는 방법이다.

13. Section 107

표의 숫자 코드(Significant Digit Code) = 유효 숫자 코드

- 코드화 대상 항목의 성질, 즉 길이, 넓이, 부피, 지름, 높이 등의 물리적 수치를 그대로 코드에 적용시키는 방법이다.
- 코드에 대상체의 성질을 그대로 표시하므로 기억하기 쉽다.
- 코드의 추가가 쉽다.
- 자릿수가 길어질 수 있고, 기계 처리에 불편하다.

14. Section 107

연상 코드(Mnemonic Code) = 기호 코드

- 코드화 대상 항목의 명칭이나 약호와 관계있는 숫자나 문자, 기호를 이용하여 코드를 부여하는 방법이다.
- 코드만 보고도 대상 품목을 쉽게 연상할 수 있다.
- 자릿수가 길어질 수 있다.
- 지명, 물건명, 상호명에 많이 적용한다.

15. Section 107

기호 코드(Mnemonic Code) = 연상 코드

- 코드화 대상 항목의 명칭이나 약호와 관계있는 숫자나 문자, 기호를 이용하여 코드를 부여하는 방법이다.
- 코드만 보고도 대상 품목을 쉽게 연상할 수 있다.
- 자릿수가 길어질 수 있다.
- 지명, 물건명, 상호명에 많이 적용한다.

16. Section 107

순차 코드(Sequence Code)

- 자료의 발생 순서, 크기 순서 등 일정한 기준에 따라 최초의 자료부터 차례로 일련 번호를 부여하는 방법이다.
- 항목수가 적고 변경이 적은 자료에 적합하다.
- 기억 공간의 낭비가 없고 자릿수가 가장 짧다.
- 단순 · 명료하며 이해하기 쉽다.
- 발생 순서에 따른 코드인 경우 추가가 매우 편리하다(확장성 용이).
- 고유성이 있으므로 기억이 용이하다.
- 명확한 분류 기준이 없어 코드에 따른 분류가 어렵다.
- 기계 처리가 어렵다.
- 코드의 중간에 새로운 자료나 누락된 자료를 삽입하기 어렵다(융통성이 적음).

17. Section 107

순차 코드는 명확한 분류 기준이 없어 코드에 따른 분류가 어려우므로 분류별 그룹화에 부적합하다고 할 수 있다.

18. Section 107

코드 중 '1'과 '3'의 순서가 바뀌었다. 입력 시 좌우 자리를 바꾸어 기록한 경우에 발생하는 오류는 전위 오류(Transposition Error)이다.

19. Section 107

- 표의 숫자 코드(Significant Digit Code) : 코드화 대상 항목의 성질, 즉 길이, 넓이, 부피, 지름, 높이 등의 물리적 수치를 그대로 코드에 적용시키는 방법
- 연상 코드(Mnemonic Code) : 코드화 대상 항목의 명칭이나 약호와 관계있는 숫자나 문자, 기호를 이용하여 코드를 부여하는 방법
- 합성 코드(Combined Code) : 필요한 기능을 하나의 코드로 수행하기 어려운 경우 두 개 이상의 코드를 조합하여 만드는 방법

20. Section 107

- 구분(블록) 코드(Block Code) : 코드화 대상 항목 중에서 공통성이 있는 것끼리 블록으로 구분하고, 각 블록 내에서 일련 번호를 부여하는 방법
- 10진 코드(Decimal Code) : 코드화 대상 항목을 0~9까지 10진 분할하고, 다시 그 각각에 대하여 10진 분할하는 방법을 필요한 만큼 반복
- 기호 코드(Mnemonic Code) : 코드화 대상 항목의 명칭이나 약호와 관계 있는 숫자나 문자, 기호를 이용하여 코드를 부여하는 방법(연상 코드)

21. Section 107

그룹 분류 코드는 그룹으로 분류한 다음 그룹내에서 또 다시 코드를 부여하기 때문에 기본적으로 소요되는 자릿수가 필요하다. 즉 적은 자릿수로 많은 그룹을 표현한다고 할 수 없다. 적은 자릿수로 많은 그룹을 구분할 수 있는 코드는 구분 코드이다.

22. Section 107

- 필사 오류(Transcription Error) = 오자 오류 : 입력 시 임의의 한 자리를 잘못 기록한 경우 발생
- 생략 오류(Omission Error) : 입력 시 한 자리를 빼놓고 기록한 경우 발생
- 추가 오류(Addition Error) : 입력시 한 자리를 더 추가하여 기록한 경우 발생

24. Section 107

10진 코드(Decimal Code)

- 코드화 대상 항목을 0~9까지 10진 분할하고, 다시 그 각각에 대하여 10진 분할하는 방법을 필요한 만큼 반복한다.
- 도서관에서 도서 정리를 목적으로 널리 사용된다.
- 코드 체계가 명확하고, 무한대로 확장이 가능하다.
- 삽입 및 추가가 용이하다.
- 10개 이상의 분류일 때는 비효율적이다.
- 자릿수가 길고, 기계 처리가 어렵다.

25. Section 107

12345가 1345로 표시된 것은 2가 생략된 형태로, 생략 오류(Omission Error)가 발생한 것이다.

- 필사(Transcription) 오류 : 입력 시 임의의 한 자리를 잘못 기록한 경우
- 전위(Transposition) 오류 : 입력 시 좌우 자리를 바꾸어 기록한 경우 발생
- 추가(Addition) 오류 : 입력 시 한 자리를 더 추가하여 기록한 경우 발생

26. Section 107

- 순차 코드(Sequence Code, 순서 코드) : 자료의 발생 순서, 크기 순서 등 일정 기준에 따라 최초의 자료부터 차례로 일련 번호를 부여하는 방법
- 그룹 분류식 코드(Group Classification Code) : 코드화 대상 항목을 일정 기준에 따라 대분류, 중분류, 소분류 등으로 구분하고, 각 그룹 안에서 일련 번호를 부여하는 방법
- 10진 코드(Decimal Code) : 코드화 대상 항목을 0~9까지 10진 분할하고, 다시 그 각각에 대하여 10진 분할하는 방법을 필요한 만큼 반복하는 방법

27. Section 107

강판의 두께와 폭, 길이를 그대로 코드에 적용시켰으므로 표의 숫자 코드에 해당한다.

- 구분 코드(블록 코드) : 코드화 대상 항목 중에서 공통성이 있는 것끼리 블록으로 구분하고 각 블록 내에서 일련 번호를 부여하는 방법
- 연상 코드(기호 코드) : 코드화 대상 항목의 명칭이나 약호와 관계 있는 숫자, 문자, 기호를 이용해 코드를 부여하는 방법
- 약자식 코드 : 코드화 대상 항목의 약자를 그대로 코드로 사용하는 방법

28. Section 106

코드와 데이터는 1:1의 대응관계가 되어야 한다.

29. Section 107

- 구분 코드(Block Code) : 코드화 대상 항목 중에서 공통성이 있는 것끼리 블록으로 구분하고, 각 블록 내에서 일련번호를 부여하는 방법
- 순차 코드(Sequence Code) : 자료의 발생 순서, 크기 순서 등 일정 기준에 따라서 최초의 자료부터 차례로 일련 번호를 부여하는 방법
- 합성 코드(Combined Code) : 필요한 기능을 하나의 코드로 수행하기 어려운 경우 두 개 이상의 코드를 조합하여 만드는 방법

30. Section 107

- 순서 코드(Sequence Code) : 자료의 발생 순서, 크기 순서 등 일정 기준에 따라서 최초의 자료부터 차례로 일련 번호를 부여하는 방법
- 그룹 분류식 코드(Group Classification Code) : 코드화 대상 항목을 일정 기준에 따라 대분류, 중분류, 소분류 등으로 구분하고, 각 그룹 안에서 일련 번호를 부여하는 방법
- 구분(블록) 코드(Block Code) : 코드화 대상 항목 중에서 공통성이 있는 것끼리 블록으로 구분하고, 각 블록 내에서 일련 번호를 부여하는 방법

31. Section 106

코드 설계의 목적에는 기계 처리의 용이성, 취급의 용이성, 분류의 편리성, 확장성, 단순성, 고유성, 표의성, 함축성 등이 있다.

3장 정답 및 해설 — 입 · 출력 설계

1.④	2.③	3.③	4.②	5.①	6.①	7.②	8.④	9.③	10.③	11.①	12.③	13.②	14.③	15.③
16.①	17.④	18.④	19.①	20.④	21.④	22.④	23.④	24.④	25.③	26.②	27.③	28.③		

1. Section 108

전표 번호나 발행 주문과 같은 고정 항목은 미리 인쇄하거나 선택할 수 있게 하여 원시 전표 작성의 효율성을 높인다.

2. Section 108

- 집중 매체화 시스템 : 발생한 데이터를 전표상에 기록하고, 일정 시간 단위로 일괄 수집하여 입력 매체에 수록하는 방식
- 분산 매체화 시스템 : 데이터를 발생한 장소에서 매체화하여 처리하는 방식
- 직접 입력 시스템 : OMR 카드 등과 같이 사람이 직접 손으로 써서 입력하는 방식

4. Section 108

①은 턴 어라운드 시스템, ③은 분산 매체화 시스템, ④는 타건 입력 시스템에 대한 설명이다.

5. Section 108

GIGO(Garbage In Garbage Out)는 '쓰레기가 들어가면 쓰레기가 나온다'는 의미이다. 아무리 정확한 컴퓨터라도 사람이 잘못 입력하면 컴퓨터도 잘못된 결과를 출력할 수 밖에 없으므로, 올바른 자료를 바르게 입력해야 한다는 입력의 중요성을 강조하는 용어이다.

6. Section 108

입력 정보의 설계 순서

입력 정보의 발생 → 입력 정보의 수집 → 입력 정보의 매체화 → 입력 정보의 투입 → 입력 정보의 내용

7. Section 108

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

8. Section 108

터치 스크린은 입 · 출력이 모두 가능한 장치로, 화면을 손가락으로 터치하여 원하는 자료를 입력하면 바로 화면을 통해 출력 결과를 확인할 수 있다.

9. Section 108

Point Of Sale(POS)

- 현장에 있는 POS용 단말기를 통해 판매 정보를 자동으로 관리해 주는 종합 판매 정보 시스템이다.
- 소프트웨어, POS용 단말기, 바코드 판독기가 필요하다.

10. Section 108

- 입력 정보 투입에 대한 설계 : 매체화된 정보를 정보 처리 과정

에 입력하는 방식을 설계하는 단계로, 입력장치와 투입 주기 및 시기, 오류 검사 방법에 대해 결정

- **입력 정보의 매체화에 대한 설계** : 입력 정보를 어떤 매체로 변환할 것인가에 대해 설계하는 단계로, 매체화 담당자와 장소, 매체화 장치, 레코드의 길이와 형식, 매체화 주기 및 시기, 매체화 시 오류 검사 방법에 대해 결정
- **입력 정보의 내용에 대한 설계** : 입력 정보의 목적을 달성하는 데 필요한 내용을 설계하는 단계로, 입력 항목의 배열 순서와 항목명, 입력 항목의 자릿수와 문자 구분, 투입 시 오류 검사 방법에 대해 결정

11. Section 108

입력 정보의 내용에 대한 설계

- 입력 항목의 배열 순서와 항목명에 대해 결정한다.
- 입력 항목의 자릿수와 문자 구분에 대해 결정한다.
- 입력 정보의 오류 검사 방법에 대해 결정한다.

12. Section 108

음성 입력의 단점

- 장치 가격이 비싸다.
- 사용 가능한 어휘가 한정되어 있다.
- 사용자의 음성이 변경되면 인식하지 못한다.

13. Section 101

- CAM(Content Addressable Memory) : 기억장치에서 자료를 읽을 때 주소에 의해 접근하지 않고 기억된 내용의 일부를 이용하여 접근(Access)할 수 있는 기억장치로, 연관기억장치 또는 연상기억장치라고 함
- ROM(Read Only Memory) : 기억된 내용을 읽을 수만 있는 기억장치로, 주로 기본 입 · 출력 시스템(BIOS), 글자 폰트, 자가진단 프로그램(POST) 등이 저장됨
- RAM(Random Access Memory) : 읽고 쓰기가 자유로운 기억장치로, 현재 사용중인 프로그램이나 데이터가 저장됨

14. Section 108

- **턴 어라운드 시스템** : 입력된 자료가 처리되어 일단 출력된 후 이용자를 경유하여 다시 재입력되는 방식으로, 공과금, 보험료 징수 등의 지로 용지를 처리하는 데 사용됨
- **집중 매체화 시스템** : 발생한 데이터를 전표 상에 기록하고, 일정 시간 단위로 일괄 수집하여 전산 부서에서 입력 매체에 수록하는 방식

15. Section 108

- **턴 어라운드 시스템** : 입력된 자료가 처리되어 일단 출력된 후 이용자를 경유하여 다시 재입력되는 방식으로, 공과금, 보험료 징수 등의 지로 용지를 처리하는 데 사용됨
- **분산 매체화 시스템** : 데이터를 발생한 장소에서 매체화하여 처리하는 방식

16. Section 108

입력 데이터가 기록될 매체에 대해 결정하는 것이므로, 입력 매체의 설계 단계에 속한다.

17. Section 108

입력 정보의 내용에 대한 설계

- 입력 항목의 배열 순서와 항목명
- 입력 항목의 자릿수와 문자 구분
- 입력 정보의 오류 검사 방법

※ 레코드 길이 및 형식은 입력 정보의 매체화에 대한 설계시 결정사항이다.

18. Section 108

- **매체의 표준화** : 입 · 출력 데이터의 성격, 양, 사용 목적에 따라 적절한 매체를 선택하여 표준화함
- **코드의 표준화** : 시스템 간의 호환성과 업무의 원활성을 위하여 한 조직 내의 여러 시스템에서 사용되는 코드를 표준화함
- **형식의 표준화** : 입 · 출력 매체의 특성과 기능을 고려하여 입 · 출력 형식을 표준화함으로써 컴퓨터의 효율성이 향상되며, 경비를 절약할 수 있음

19. Section 108

턴 어라운드 시스템은 입력된 자료가 처리되어 일단 출력된 후 이용자를 경유하여 다시 재입력되는 방식으로, 공과금 청구서, 보험료 징수, 수표 발행 등에 주로 사용된다.

20. Section 109

출력 설계 순서

출력 정보의 내용 → 출력 정보의 매체화 → 출력 정보의 분배 → 출력 정보의 이용

21. Section 109

출력 설계 시에는 오류 검증 절차가 아니라 출력 정보의 오류 검사 방법을 결정하는 작업을 수행해야 한다.

22. Section 109

기밀성의 유무와 보존에 대한 결정은 출력 정보의 이용에 대한 설계 단계에서 수행되는 작업이다.

23. Section 109

출력 보고서 설계 시 고려 사항

- 이용자
- 이용 목적
- 보고서의 양

24. Section 109

출력 주기 및 시기의 결정은 출력 정보 매체화의 설계 단계에서 수행한다.

25. Section 108

- **프롬프트 방식** : 프롬프트가 위치한 곳에 사용자가 직접 명령어나 자료를 입력하는 방식으로 명령어 처리 방법의 구현이 쉽고, 명령어를 복합적으로 활용하면 복잡한 명령도 사용 가능함
- **메뉴 방식** : 시스템에서 사용되는 명령어나 선택 사항을 메뉴로 구성하여 화면에 진열하는 방식으로, 배우기 쉽고 편리하며, 초보자도 쉽게 사용할 수 있어 사용자 층이 다양한 경우에 적합함
- **아이콘 방식** : 화면에 여러 개의 항목을 진열하고 그 중의 하나를 선택 도구로 지정하여 직접 실행하는 방식으로, 직접 조작 방식이라고도 함

26. Section 109

집계 구분의 결정은 출력 정보 내용의 설계 단계에서 수행한다.

27. Section 109

- **디지타이저** : 3차원 게임에 사용되는 캐릭터의 모형이나 2차원의 건축 설계도면 같은 데이터를 입력할 때 사용하는 장치
- **MICR(Magnetic Ink Character Reader)** : 자성을 띤 특수 잉크로 인쇄된 문자나 기호를 판독하는 장치
- **데이터 수집장치** : 여러 곳에 산재해 있는 여러 데이터를 수집하는 장치

28. Section 108

기입에 혼란을 가져올 수 있는 내용은 기입 요령을 명시해야 한다.

4장 정답 및 해설 — 파일 설계

1.③	2.④	3.③	4.②	5.④	6.①	7.③	8.②	9.④	10.③	11.②	12.③	13.③	14.④	15.④
16.④	17.①	18.①	19.③	20.①	21.③	22.④	23.④	24.④						

1. Section 112

파일의 설계 순서 중 가장 먼저 수행해야 할 사항은 파일의 성격 검토이다.

2. Section 112

파일 설계 순서

파일의 성격 검토 → 파일 항목의 검토 → 파일의 특성 조사 → 파일 매체의 검토 → 편성법 검토

3. Section 110

- **필드** : 파일 구성의 최소 단위, 의미 있는 정보를 표현하는 최소 단위
- **비트** : 자료 표현의 최소 단위
- **바이트** : 문자를 표현하는 최소 단위

4. Section 112

매체의 특성에 따라 업무 효율이 달라질 수 있으므로, 매체 선정 시 매체의 특성을 고려해야 한다.

5. Section 110

마스터 파일(Master File)

- 전표 처리에서의 원장 또는 대장에 해당되는 파일로서, 자료 관리의 중추적 역할을 담당하며 기본이 되는 파일이다.
- 트랜잭션 파일에 의해 갱신된다.

6. Section 110

- 트랜잭션 파일(Transaction File)은 거래 내역이나 변동 내용 등 일시적인 성격을 지닌 정보를 기록하는 파일로, 마스터 파일을 갱신하거나 조회할 때 사용한다.
- **트레일러 파일(Trailer File)** : 마스터 파일을 목적에 따라 여러 개의 파일로 나누었을 때 가장 끝부분에 해당하는 파일

7. Section 110

- **트랜잭션 파일(Transaction File)** : 거래 내역이나 변동 내용 등 일시적인 성격을 지닌 정보를 기록하는 파일로, 마스터 파일을 갱신하거나 조회할 때 사용함
- **히스토리 파일(History File)** : 후일 통계 처리에 사용할 자료나 사고 발생 시 마스터 파일 등을 원상 복구시키기 위한 자료를 보존한 파일
- **트레일러 파일(Trailer File)** : 마스터 파일을 목적에 따라 여러 개의 파일로 나누었을 때 가장 끝부분에 해당하는 파일

8. Section 110

- **트랜잭션 파일(Transaction File)** : 거래 내역이나 변동 내용 등 일시적인 성격을 지닌 정보를 기록하는 파일로, 마스터 파일을 갱신하거나 조회할 때 사용됨
- **마스터 파일(Master File)** : 전표 처리에서의 원장 또는 대장에 해당하는 파일로, 자료 관리의 중추적 역할을 담당하며 기본이 되는 파일. 트랜잭션 파일에 의해 갱신됨
- **요약 파일(Summary File)** : 다른 파일의 중요 내용이나 합계를 요약해 놓은 파일로, 집계용으로 많이 사용됨

9. Section 110

- **작업(Work) 파일** : 파일을 수행 내용에 따라 분류할 경우 프로그램 실행 중 일시적으로 발생하는 자료를 처리하기 위한 임시 파일
- **히스토리 파일(History File)** : 후일 통계 처리에 사용할 자료나 사고 발생 시 마스터 파일 등을 원상 복구시키기 위한 자료를 보존한 파일
- **요약 파일(Summary) 파일** : 다른 파일의 중요 내용이나 합계를 요약해 놓은 파일로, 집계용으로 많이 사용함

10. Section 110

비블록화 가변 길이 레코드는 전송하는 레코드의 길이가 모두 다르므로 레코드의 크기 정보를 표시해야 한다. 그러므로 프로그램의 작성이 어렵다. 또한 블록을 구성하지 않아 레코드의 수만큼 입 · 출력이 발생하므로 처리 속도도 느리다.

11. Section 111

색인 순차 편성의 구성

- **기본 데이터 구역** : 실제 레코드들을 기록하는 부분
- **인덱스 데이터 구역** : 기본 데이터 구역에 대한 인덱스가 기록되는 부분
- **오버플로 구역** : 기본 데이터 구역의 공간 부족에 대비하여 예비적으로 확보해 둔 영역

12. Section 111

리스트 편성 파일에서 레코드를 삽입하거나 삭제할 때는 삽입(삭제) 위치의 앞뒤 포인터 내용만 변경하면 되므로 레코드의 삽입 · 삭제 · 추가가 쉽다.

13. Section 111

- **순차 파일** : 입력되는 데이터들을 논리적인 순서에 따라 물리적 연속 공간에 순차적으로 기록하는 방식으로, 주로 자기 테이프에 사용됨
- **랜덤(직접) 파일(Random File)** : 입력되는 정보를 기록 순서나 코드 순서와 같은 논리적 순서와 관계없이 특정한 방법으로 키를 생성하여 임의로 보관하고, 처리 시에도 필요한 장소에 직접 접근할 수 있도록 편성하는 방식
- **리스트 파일(List File)** : 레코드들을 일정한 규칙이나 제약 없이 기억공간에 자유롭게 기록하며, 각 레코드들은 다음 레코드의 주소를 가지고 있는 포인터를 통해 논리적인 순서로 연결됨

14. Section 111

랜덤 편성법은 주소를 직접 계산하여 키 변환을 하는 방법을 사용하므로, 키 변환을 위한 처리 시간이 필요하다.

15. Section 111

해싱 함수 선택 시 고려 사항

- 오버플로(Overflow)의 최소화
- 충돌(Collision)의 최소화

- 계산 과정의 최소화(해싱 함수의 단순성)
- 키 변환 속도
- 버킷의 크기

16. Section 111

- 동거자(Synonym) : 같은 주소를 갖는 레코드의 집합(유사어, 동의어)
- Collision(충돌) : 두 개의 서로 다른 레코드가 같은 기억공간(버킷)을 점유하려고 하는 현상

17. Section 112

파일 설계 시 고려 사항

- 보조기억장치의 접근 횟수
- 파일에 접근해서 수행할 연산의 유형 : 생성, 갱신, 검색, 관리 유지
- 파일 활동률(File Activity Ratio) = 처리 대상 레코드 수/전체 레코드 수
- 트랜잭션이나 검색 요청에 대한 응답 시간(Response Time)

18. Section 112

파일 활동률(File Activity Ratio) = 처리 대상 레코드 수/전체 레코드 수

19. Section 112

매체 수의 산출은 파일 매체에 관한 설계 시 종합 검토 사항에 속한다.

20. Section 111

급여 업무와 같이 변동 사항이 크지 않고 기간별로 일괄 처리를 하는 경우에는 순차 파일을 사용하는 것이 가장 좋다.

- 색인 순차 파일(Indexed Sequential File) : 순차 처리와 랜덤 처리가 모두 가능하도록 레코드들을 키 값 순으로 정렬(Sort)시켜 기록하고, 레코드의 키 항목만을 모은 인덱스를 구성하여 편성하는 방식
- 랜덤 파일(Random File) : 입력되는 정보를 기록 순서나 코드 순서와 같은 논리적 순서와 관계없이 특정한 방법으로 키를 생성하여 임의로 보관하고, 처리 시에도 필요한 장소에 직접 접근할 수 있도록 편성하는 방식
- 리스트 파일(List File) : 레코드들을 일정한 규칙이나 제약 없이 기억공간에 자유롭게 기록하며, 각 레코드들은 다음 레코드의 주소를 가지고 있는 포인터를 통해 논리적인 순서로 연결됨

21. Section 111

순차 파일은 입력되는 데이터들을 논리적인 순서에 따라 물리적 연속 공간에 순차적으로 기록하는 방식으로, 데이터 검색 시 무조건 처음부터 순차적으로 검색하기 때문에 검색 효율이 낮다.

22. Section 111

해싱 함수와 관련 있는 파일은 랜덤 파일이다.

23. Section 111

해싱 함수 선택 시 고려 사항

- 오버플로(Overflow)의 최소화
- 충돌(Collision)의 최소화
- 계산 과정의 최소화(해싱 함수의 단순성)
- 키 변환 속도
- 버킷의 크기

24. Section 111

랜덤 편성은 논리적 순서와 관계없이 특정한 방법으로 키를 생성하여 임의의 위치에 보관하고, 처리시에도 필요한 장소에 직접 접근할 수 있도록 편성하는 방식으로, 충돌이 발생할 수 있으므로, 이를 위한 기억공간의 확보가 필요하다.

5장 정답 및 해설 — 프로세스와 프로그램 설계

1.③	2.②	3.②	4.①	5.④	6.③	7.①	8.②	9.③	10.②	11.④	12.②	13.④	14.②	15.③
16.②	17.④	18.③	19.①	20.①	21.③	22.①	23.②	24.②	25.①	26.②	27.④	28.①	29.④	30.④
31.③														

1. Section 113

프로세스 설계(Process Chart)

- 입력 자료를 출력 정보로 얻어내기 위한 처리 절차에 관한 설계이다.
- 자료와 처리의 흐름, 정보의 흐름 등을 밝히기 위한 목적으로 사용한다.

2. Section 113

새로운 시스템의 프로세스 설계뿐만 아니라 기존 시스템의 문제점 분석이 가능하도록 설계해야 한다.

3. Section 113

정보의 사용 목적 확인은 기본 사항 확인 단계에서 수행해야 한다.

4. Section 113

- **시스템 흐름도** : 자료 발생부터 결과를 얻기까지 시스템의 전 과정을 나타내는 순서도
- **블록 차트(Block Chart)** : 시스템의 목적을 달성하는 데 필요한 모든 기능 및 각 부서를 블록으로 표시하는 차트로, 각 기능 및 부서 상호 간의 순서와 관련 상태를 선으로 연결함으로써 상호 관련 상태를 정보의 흐름으로 표시하며, 업무를 개괄적으로 파악하는 데 사용함
- **프로그램 흐름도** : 시스템 흐름도 중에서 컴퓨터로 처리하는 부분을 중심으로 자료 처리에 필요한 모든 조작 순서를 표시하는 순서도로, 개요 순서도와 상세 순서도로 구분됨

5. Section 113

- ④의 내용은 상세 흐름도에 대한 설명이다.
- **개요 흐름도(General Flowchart)** : 프로그램의 전체 내용을 개괄적으로 설명한 흐름도로, 컴퓨터로 처리하는 순서와 내용의 개요를 표시함

6. Section 114

- **변환(Conversion)** : 입력 매체 상의 데이터 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 기능으로, 매체 변환이라고도 함
- **정렬(Sort, 분류)** : 레코드를 처리할 순서에 맞게 오름차순 또는 내림차순으로 재배치하는 기능
- **대조(Matching)** : 두 개의 파일을 대조시켜 그 기록 순서나 기록 내용을 검사하는 기능

7. Section 114

조건에 부합되는 것과 그렇지 않은 것을 분리하는 것은 분배(Distribution)에 해당한다.

8. Section 114

정렬(Sort, 분류)

- 레코드를 처리할 순서에 맞게 오름차순 또는 내림차순으로 재배치하는 기능이다.
- 정렬중에는 레코드의 중복을 검사할 수 있고, 정렬된 레코드는 시스템의 처리 효율을 높인다.

9. Section 114

병합(Merge) : 동일한 파일 형식을 갖는 두 개 이상의 파일을 일정한 규칙에 따라 하나의 파일로 통합 처리하는 기능

10. Section 114

- **추출(Extract)** : 파일 안에서 특정 조건에 만족하는 데이터만을 골라내는 기능으로, 정보 검색을 위하여 필수적인 기능
- **정렬(Sort, 분류)** : 레코드를 처리할 순서에 맞게 오름차순 또는 내림차순으로 재배치하는 기능

11. Section 114

- **추출(Extract)** : 파일 안에서 특정 조건에 만족하는 데이터만을 골라내는 기능으로, 정보 검색을 위하여 필수적인 기능

- 병합(Merge) : 동일한 파일 형식을 갖는 두 개 이상의 파일을 일정한 규칙에 따라 하나의 파일로 통합 처리하는 기능
- 보고서(Reporting) : 처리 결과를 출력하는 기능

12. Section 114

대조(Matching)

- 두 개의 파일을 대조시켜 그 기록 순서나 기록 내용을 검사하는 기능이다.
- 트랜잭션 파일을 이용하여 마스터 파일을 갱신하는 경우 반드시 두 파일의 대조 작업이 필요하다.

13. Section 114

분배(Distribution) : 하나의 파일 안에서 조건에 맞는 것과 그렇지 않은 것을 분리하는 기능

14. Section 115

- 균형 체크(Balanced Check) : 차변과 대변의 한계값을 검사하는 방법으로 대차의 균형이나 가로, 세로의 합계가 일치하는가를 검사
- 순차 체크(Sequence Check) : 입력되는 데이터의 순서가 이미 정해진 순서와 일치하는지를 검사하는 방법
- 형식 체크(Format Check) : 입력되는 데이터의 자릿수, 형식, 행, 열, 페이지 번호 등이 규정대로 되어 있는지 검사하는 방법

15. Section 115

①은 일괄 합계 검사(Batch Total Check), ②는 균형 검사(Balance Check), ④는 데이터 수 검사(Data Count Check)에 대한 설명이다.

16. Section 115

- 대조 검사 : 입력 데이터와 시스템에 보관된 별도의 코드표를 대조하여 그것이 일치하는지를 검사하는 방법
- 일괄 합계 검사 : 입력 데이터의 특정 항목 합계값을 미리 계산해서 이것을 입력 데이터와 함께 입력하고, 컴퓨터 상에서 계산한 결과값과 수동 계산 결과값이 같은지를 검사하는 방법
- 균형 검사 : 차변과 대변의 한계값을 체크하는 방법으로, 대차의 균형이나 가로, 세로의 합계가 일치하는가를 검사

17. Section 115

공란 검사는 공백이 포함되어 있으면 무조건 오류가 발생하는 것이 아니라 공백으로 있어야 할 필드에 확실히 공백으로 되어 있지 않은 경우 오류가 발생하는 것이다.

18. Section 115

체크 디지트는 오류를 검사하기 위해 넣어주는 1자리의 숫자를 의미한다. 코드의 각 자릿수에 가중치(Weight)를 곱하여 합산한 값을 Modulus(10 또는 11)로 나누어 나머지를 구하고, Modulus(10 또는 11)에서 나머지 값을 뺀 값이 체크 디지트가 된다.

19. Section 115

대조 검사(Matching Check)는 입력 데이터와 시스템에 보관된 별도의 코드표를 대조하여 그것이 일치하는지를 검사하는 방법이다.

20. Section 115

- Balance Check – 차변과 대변
- Batch Total Check – 수작업 합계 체크
- Format Check – 칼럼 수 밀림
- Limit Check– 상한값과 하한값
- Parity Check – 메모리 내부에서 체크

21. Section 115

- 2월의 날수가 30일로 기록된 것을 오류로 검사한 것은 한계 검사 기법을 사용한 것이다.
- 한계 검사(Limit Check) : 자료 항목이 주어진 조건 범위에 맞는지를 검사하는 것으로, 이때 최대치와 최소치가 사전에 준비되어 있어야 함

22. Section 115

데이터 수 검사(Data Count Check)

- 사전에 데이터 개수를 확인한 후 컴퓨터로 처리한 개수와 맞는지 비교하는 방법이다.
- 데이터가 누락되었는지, 중복되었는지를 검사하기 위한 방법이다.

23. Section 115

체크 디지트 검사(Check Digit Check)

- 코드를 설계할 때 본래의 코드에 검사할 수 있는 1자리의 숫자를 넣어줌으로써 컴퓨터에 의하여 자동으로 검사하는 방법

이다.

- 입력 항목이 코드인 경우에 적용하는 방법으로, 검사를 하기 위해서 넣어준 1자리의 숫자를 체크 디지트라 한다.
- 주민등록번호, 상품코드, 계좌번호 등을 검사할 때 사용한다.

24. Section 115

대조 검사(Matching Check)는 입력 데이터와 시스템에 보관된 별도의 코드표를 대조하여 그것이 일치하는지를 검사하는 방법이다.

25. Section 115

코드 값 : 5 3 2 7 4
× × × × ×
가중치 : 5 7 4 5 8

- 2 + 5 + 2 + 1 + 8 + 3 + 5 + 3 + 2 = 31
- 31 ÷ 10 = 1(나머지)
- 10 − 1 = 9(체크 디지트) → 532749

26. Section 115

반향 체크는 컴퓨터 입력 단계에서의 체크 방법이다.

- 중복 레코드 체크(Double Record Check) : 계산 처리 과정에서 동일한 레코드가 있는지를 검사하는 방법
- 플러스 마이너스 체크(Plus-Minus Check = 부호 검사(Sign Check)) : 계산 결과가 양수 또는 음수인지를 검사하는 방법
- 오버플로 체크(Overflow Check) : 계산된 결과가 규정된 자릿수 또는 한계를 초과하는지를 검사하는 방법

27. Section 116

프로그램 설계서의 구성 요소

시스템명 및 코드명, 설계 방침, 프로세스 흐름도, 코드표, 입 · 출력 설계표, 프로그래밍 지시서

28. Section 116

프로그래밍 지시서에 포함되는 내용

프로그램명, 설계서 작성자명, 프로그램의 작성 기간, 작성 비용, 작성 시기, 입 · 출력 일람, 처리 개요, 처리 명세, 프로그램 작성 후 제출할 사항, 참고 자료

29. Section 116

프로그램 설계서는 시스템 엔지니어 또는 시스템 분석가가 작성하며, 프로그래머가 시스템의 구조와 기능을 코드로 구현할 수 있을 정도로 상세하게 작성한다.

30. Section 116

시스템 운용 계획서는 새로 개발된 시스템을 실제 현장에서 운용하기 위한 것으로, 프로그램 설계서에 포함되지 않는다.

31. Section 115

코드 값 : 8 3 2 9 4
× × × × ×
가중치 : 8 7 6 5 4

- 64 + 21 + 12 + 45 + 16 = 158
- 158 ÷ 10 = 8(나머지)
- 10 − 8 = 2(체크 디지트) → 832942

6장 정답 및 해설 — 시스템 평가와 문서화

1.③	2.④	3.①	4.③	5.①	6.④	7.②	8.①	9.④	10.①	11.②	12.①	13.③	14.②	15.②
16.②	17.②	18.④	19.④	20.④	21.④	22.④	23.④	24.②	25.②	26.④	27.②	28.①	29.④	30.④
31.②	32.③	33.④	34.④											

1. Section 117

시스템 평가의 목적

- 시스템의 성능과 유용도를 판단할 수 있다.
- 처리 비용과 처리 효율 면에서 개선점을 파악할 수 있다.
- 다음 시스템을 개발할 때 원활한 진행을 위한 참고 자료가 될 수 있으며, 동일한 실수를 하지 않게 된다.
- 시스템 운영 관리의 타당성을 파악할 수 있다.

2. Section 117

시스템의 판정 기준 : 시스템의 능력성, 신뢰성, 유연성, 효율성, 편리성, 안정성, 생산성

3. Section 117

시스템의 평가

- **효율성** : 현재 사용 가능한 기기만으로도 시스템이 잘 수행되는지의 정도
- **신뢰성** : 시스템이 고장 없이 정해진 시간 내에 정확한 결과를 산출하는 정도
- **유연성** : 다른 하드웨어 환경에서도 소프트웨어의 수행이 무난한 성질

4. Section 117

- **성능 평가** : 시스템이 운영 계획에서 마련한 운영 스케줄대로 수행하는지를 평가하는 것
- **기능 평가** : 사용자가 요구했던 기능을 정확하게 수행하는지를 평가
- **신뢰성 평가** : 주어진 환경에서 주어진 시간 동안 오류 없이 작동할 확률을 평가

5. Section 117

시스템 평가의 항목 : 기능 평가, 성능 평가, 신뢰성 평가

6. Section 117

신뢰성 평가를 위한 검토 항목

- 시스템 전체의 가동률
- 시스템을 구성하는 각 요소의 신뢰도
- 신뢰성 향상을 위해 시행한 처리의 경제 효과

7. Section 117

먼저 MTBF와 MTTF를 같은 개념으로 사용하는지 확인해야 하는데, 이번 문제에서는 상호 인접한 고장 사이의 가동된 시간의 평균, 즉 평균 가동 시간에 대한 설명만 있으므로 MTBF와 MTTF를 같은 개념으로 보고 있는 것이다.

8. Section 117

신뢰성의 지표

- MTBF(Mean Time Between Failure) : 평균 고장 간격, MTBF = MTTF + MTTR
- MTTR(Mean Time To Repair) : 평균 수리 시간
- **신뢰도(Availability)** : 시스템의 총 운용 시간 중 정상적으로 가동된 시간의 비율로 가용도라고도 함

9. Section 117

평균 가동 시간이란 시스템이 고장 없이 가동된 시간들의 평균치이다.

10. Section 117

- MTBF : 평균 고장 간격, 수리가 가능한 시스템이 고장난 후부터 다음 고장이 날 때까지의 평균 시간
- **신뢰도(가용도)** : 시스템의 총 운용 시간 중 정상적으로 가동된 시간의 비율

11. Section 117

MTBF와 MTTF를 구분한다는 아무런 제시가 없으므로 MTBF와 MTTF를 같은 개념으로 보고 문제를 푼다.

$$\text{MTBF} = \frac{\text{가동중 1} + \text{가동중 2} + \text{가동중 3} + \cdots + \text{가동중 n}}{\text{n}}$$

$$= \frac{144 + 178 + 216 + 252}{4}$$

$$= \frac{790}{4} = 197.5$$

12. Section 117

시스템 성능 평가 기준

- 응답 시간(Response Time)
- 반환 시간(Turnaround Time)
- CPU 속도 및 기억 용량
- 파일 편성법과 액세스 방식
- 파일 장치 및 입 · 출력장치의 속도
- 업무 프로그램의 구조와 사용 언어
- 업무 프로그램의 다중도 및 우선순위

13. Section 118

1인당 평균 생산성이 월간 300 라인이므로, 한 사람이 30,000 라인을 개발하는 데는 100개월이 걸린다. 개발 인원이 5명이므로 결국 20개월(100개월/5) 정도가 소요된다.

14. Section 117

시스템 완성 후의 평가 방법

- 이용 부분의 만족도를 분석한다.
- 프로그램의 정확성과 효율성을 분석한다.
- 개발비와 운용 비용을 파악하고 분석한다.

15. Section 118

시스템 도입의 평가

- **정보 작성 원가의 계산** : 처리 시간 × 단위 시간당 비용
- **시스템 이행 계획의 수립** : 개요 설계가 끝난 후 시스템을 이행하기 위한 계획을 세우는 것
- **시스템 도입 효과의 평가** : 시스템 도입과 운용에 필요한 비용에 대한 평가와 시스템 도입에 따른 직 · 간접적인 효과에 대한 평가

16. Section 118

시스템 도입에 따른 효과

- 시스템을 도입함으로써 직접 또는 간접적으로 얻을 수 있는 효과에 대해 평가하는 것으로, 이익 요소에 해당한다.
- **직접 효과** : 인건비 절감, 사무비 절감, 재고 비용 절감, 작업 능률 향상, 작업 시간 단축, 원가 절감 등
- **간접 효과** : 관리 방식의 합리화, 사무의 표준화, 서비스 향상, 신용 증가, 이미지 향상 등

17. Section 119

목적에 의한 테스트(Test)의 종류

- **기능 테스트** : 주어진 입력에 대해 기대되는 결과가 출력되는지에 대한 테스트
- **성능 테스트** : 처리량, 응답 시간, 메모리 활용도, 처리 속도 등에 대한 테스트
- **강도 테스트** : 정보의 과부하 시 최저 조건 미달이나 물리적 충격 등에 대한 테스트
- **복잡도 테스트** : 논리 경로의 복잡도에 대한 테스트

18. Section 119

- **시스템 테스트** : 시스템이 초기의 목적을 만족시키는가를 확인하는 작업
- **단위 테스트** : 각각의 모듈에 대해 독립적인 환경에서 기본 데이터만 입력하여 테스트하는 작업
- **하향 테스트 방식** : 프로그램 구조의 상위 모듈로부터 하위 모듈 방향으로 순차적으로 테스트하는 방식

19. Section 119

원시 코드의 모든 문장을 한 번 이상 수행하는 것은 화이트 박스 테스트이다.

20. Section 120

시스템의 문서화는 시스템의 대한 이해와 개발 중 또는 개발 후의 관리를 위해서 필요한 것이다.

21. Section 120

문서화된 문서는 타 업무 개발 시 중요한 참고 자료가 될 수 있다.

22. Section 120

문서의 표준화는 문서 작성법이나 표기법을 일정한 기준에 따라 통일시키는 것으로, 표준화를 수행하면 정해진 도표나 표기법을 사용하므로 프로그램의 작성이 용이해진다.

23. Section 118

시스템 도입 시 고려사항으로는 컴퓨터 시스템의 호환성, 소요 예산 및 운영조직 확보, 기기 규모의 적정성 등이 있다.

24. Section 120

문서화의 목적 및 효과

- 시스템 개발팀에서 운용팀으로 인수 인계가 용이하다.
- 개발 후에 시스템의 유지보수가 용이하다.
- 개발 진척 관리의 지표를 삼을 수 있다.
- 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.
- 프로그램을 공유 재산화할 수 있다.
- 타 업무 개발에 참고할 수 있다.

25. Section 120

프로그램 표준화를 하면 프로그램 작성에 드는 시간과 비용을 절감하고, 효율적인 관리를 할 수는 있지만, 프로그램의 길이가 짧아지는 것은 아니다.

26. Section 120

프로그램을 표준화하면 업무 처리 내용 변경과 기종 변경에 따른 프로그램 변환 작업을 쉽게 할 수 있다.

27. Section 119

- 인스펙션(Inspection) : 검토 회의(Walk-Through)를 발전시킨 형태로, 소프트웨어 개발 단계에서 산출된 결과물의 품질을 평가하여 이를 개선시키는 데 사용됨
- 디버깅(Debugging) : 테스트에 의해 밝혀진 오류를 수정하는 작업
- 검사(Testing) : 소프트웨어의 오류를 찾아내기 위하여 시스템을 실행하여 평가하는 작업

28. Section 119

- 단위 테스트 방식 : 각각의 모듈에 대해 독립적인 환경에서 기본 데이터만 입력하여 테스트하는 방식
- 하향 테스트 방식 : 프로그램 구조의 상위 모듈로부터 하위 모듈 방향으로 순차적으로 테스트하는 방식
- 통합 테스트 방식 : 시스템 모듈 간의 상호 인터페이스가 원활하게 수행되고 있는가를 확인하는 방법으로, 프로그램 단위별로 디버깅이 끝난 것을 모아 서로 연관된 프로그램군을 계통적으로 테스트함

29. Section 119

문서화의 목적 및 효과

- 시스템 개발팀에서 운용팀으로 인수 인계가 용이하다.
- 개발 후에 시스템의 유지보수가 용이하다.
- 시스템을 쉽게 이해할 수 있다.
- 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.
- 프로그램을 공유 재산화할 수 있다.

30. Section 120

문서화는 시스템의 개발 요령과 순서 등 시스템 개발에 관련된 모든 행위를 문서로 만들어 두는 것으로, 시스템 개발 후에 시스템 수행 능력을 쉽게 파악할 수는 없다. 시스템의 수행 능력은 시스템 평가를 통해 파악할 수 있다.

31. Section 120

문서화는 시스템의 개발 요령과 순서 등 시스템 개발에 관련된 모든 행위를 문서로 만들어 두는 것으로, 문서화의 표준화는 이를 표준화시키는 것을 의미한다. ①, ③, ④는 문서화를 표준화할 경우 개발자 측면에서의 효과이고, ②는 관리자 측면에서의 효과이다.

32. Section 118

1인당 월평균 생산량이 1,000 라인이므로, 한 사람이 80,000 라인을 개발하는 데는 80개월이 걸린다. 개발 인원이 4명이므로 결국 20개월(80개월/4) 정도가 소요된다.

33. Section 117

처리 시간의 견적 방법

- 입력에 의한 계산 : 프로세스 작업(작업 처리도)을 기초로 하여, 간단한 수식에 값을 대입하여 처리 시간을 계산하는 방법
- 컴퓨터에 의한 계산 : 처리 시간을 계산할 수 있는 응용 프로그램을 이용하는 방법
- 추정에 의한 계산 : 시스템 설계자가 과거의 경험을 바탕으로 하여 처리 시간을 추정하는 방법

34. Section 117

- 기능 평가 : 사용자가 요구했던 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

7장 정답 및 해설 — 소프트웨어 공학과 IPT 기법

1.④ 2.② 3.① 4.② 5.① 6.③ 7.① 8.③ 9.② 10.② 11.② 12.① 13.② 14.③ 15.①
16.② 17.④ 18.④ 19.① 20.③ 21.②

1. Section 121

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

2. Section 121

소프트웨어의 특징

- 유용성(Usefulness) : 사용자가 요구하는 대로 동작해야 함
- 신뢰성(Reliability) : 일정 시간 내에 주어진 조건하에서 원하는 기능을 실행할 수 있어야 함
- 명확성(Clarity) : 애매모호함이 없이 처리 절차에 맞게 수행되어 정확한 결과가 산출되어야 함
- 효율성(Efficiency) : 하드웨어 자원을 효율적으로 이용할 수 있어야 함
- 비용 효과성 : 개발, 유지보수, 이용 시 비용 면에서 효과적이어야 함

3. Section 121

소프트웨어 개발 주기

- 프로토타이핑 모델(Protopying Model) : 요구 수집 및 평가 → 신속 설계 → 프로토타입 구축 → 사용자 평가 → 프로토타입 조정 → 구현
- 나선형 모델(Spiral Model) : 계획 및 정의 → 위험 분석 → 공학적 개발 → 고객 평가
- 4세대 기법(4GT Model) : 요구 사항 수집 → 설계 전략 → 구현 → 테스트

4. Section 121

폭포수 모델(Waterfall Model)의 순서

타당성 조사 → 계획 → 요구 분석 → 기본 설계(개략 설계, 개요 설계) → 상세 설계 → 구현(Coding) → 통합 시험(Test) → 시스템 실행 → 유지보수

5. Section 121

폭포수 모델

- 가장 오랫동안 폭넓게 사용되어 온 모델로, 적용 사례가 많다.
- 단계별 정의가 분명하며, 각 단계별로 산출물이 명확히 나온다.
- 개발 과정중에 발생하는 새로운 요구 사항을 시스템에 반영하기가 어려우므로 처음부터 사용자들이 모든 요구 사항을 정확하게 제시해야 한다.
- 프로젝트 관리 및 자동화가 어렵다.
- 대규모 시스템에 적용하기가 어렵다.

6. Section 121

- 타당성 조사 : 시스템의 정의와 가능성 조사 및 다른 방법과 비교 조사하는 단계
- 계획 : 사용자 문제의 정의, 전체 시스템 차원의 기본 목표와 요구 사항 결정, 추진 방안의 제시를 통해 시스템 개발 비용 및 소요 기간, 인력 등의 개발 계획을 수립하는 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

7. Section 121

- 나선형 모델(Spiral Model, 점증적 모델) : 폭포수 모델과 프로토타이핑의 장점에 위험 분석 기능을 추가한 모델
- 폭포수 모델(Waterfall Model) : 고전적(전통적) 생명 주기 모델로서, 폭포수에서 한번 떨어진 물이 거슬러 올라갈 수 없듯이 소프트웨어 개발도 각 단계를 확실히 매듭짓고 그 결과를 철저하게 검토하여 승인 과정을 거친 후에 다음 단계를 진행하며 이전 단계로 넘어갈 수 없는 방식

8. Section 121

- **폭포수 모델(Waterfall Model)** : 고전적(전통적) 생명 주기 모델로서, 폭포수에서 한번 떨어진 물이 거슬러 올라갈 수 없듯이 소프트웨어 개발도 각 단계를 확실히 매듭짓고 그 결과를 철저하게 검토하여 승인 과정을 거친 후에 다음 단계를 진행하며 이전 단계로 넘어갈 수 없는 방식
- **프로토타이핑 모델(Prototyping Model)** : 사용자의 요구 사항을 정확히 파악하기 위해 실제 개발될 소프트웨어에 대한 모형(Prototype, 시제품)을 만들어 의사 소통의 도구로 삼으면서 개발하는 기법
- **4세대 기법(4GT, 4th Generation Techniques)** : 자연어로 표현하는 4세대 언어(4GL)를 이용하여 개발자가 조사한 요구 사항을 자동으로 구현(Coding)시키는 비절차적 기법

9. Section 122

IPT 기법의 도입 목적

- 보기 쉽고 이해하기 쉬우며, 개발과 유지보수의 양면에서 경제적인 프로그램을 만들 수 있게 한다.
- 개발자의 생산성을 향상시킨다.
- 완성된 프로그램의 품질을 향상시켜 유지보수를 용이하게 한다.
- 프로그램의 표준화로 개인 격차를 해소하고, 개발 요원의 교대 근무를 용이하게 한다.

10. Section 122

IPT 기법의 기술적 측면

- **설계 분야** : 구조적 설계(Structured Design, 복합 설계)
- **코딩 분야** : 구조적 코딩(Structured Cording)
- **테스트 분야** : 하향식 프로그래밍(Top-Down Programming)
- **IPT 보조 도구** : HIPO, 의사 코드, N-S Chart, 구조 도표, 모듈 설계

11. Section 121

IPT 기법은 개발과 유지보수 양면에서 경제적인 프로그램을 만들 수 있게 한다.

12. Section 122

HIPO는 문서를 쓰는 방법이 표준화되어 있어 체계화된 문서 작성이 가능하다.

13. Section 122

HIPO는 시스템 실행 과정인 입력, 처리, 출력을 계층적으로 기술하는 방법으로, 시스템을 설계하거나 문서화할 때 사용되며, 도식 목차, 총괄 도표, 상세 도표로 구성된다.

- **복합 설계(Composite Design)** : 상세 설계의 초기 단계에서 프로그램의 구조를 설계하는 방식으로, 프로그램의 기능을 하향식으로 상세화해서 종속 기능을 모듈에 각각 대응시킴
- **N-S 차트(Nassi-Schneiderman Chart)** : 논리의 기술(절차)에 중점을 둔 도형식 표현 도구
- **구조적 코딩(Structured Coding)** : 코딩 기법의 표준화 방법으로 GOTO문을 배제하고 순차, 선택, 반복 구조만을 사용하여 신뢰성을 향상시킨 코딩 기법

14. Section 122

HIPO는 기능 중심의 도형 표현에 편리하다.

15. Section 122

HIPO는 하향식(Top-Down) 방식을 사용하여 나타낸다.

16. Section 122

HIPO는 하향식(Top-Down) 방법으로 문서화 또는 설계를 수행한다.

17. Section 122

- **총괄 도표(Overview Diagram)** : 시스템 또는 프로그램의 기능을 입력, 처리, 출력 관계로 도표화한 것으로, 사용자의 관점에서 본 시스템 또는 프로그램의 기능과 처리 내용을 나타냄
- **상세 도표(Detail Diagram)** : 총괄 도표에 나타난 기능을 구성하는 기본 요소들을 상세히 기술한 도표

18. Section 121

폭포수 모델은 각 단계를 확실히 매듭지은 후 다음 단계로 진행하며, 다음 단계로 넘어간 후에는 이전 단계로 넘어갈 수 없는 방식이므로 요구 사항 변경이 매우 어렵다.

19. Section 121

- **요구 분석** : 소프트웨어에 요구되는 기능, 성능, 그리고 인터페이스 등 사용자의 요구 사항을 구체적으로 이해하는 단계
- **코딩** : 설계 단계에서 만들어진 설계 사양서를 바탕으로 프로그램을 작성하는 단계

• **운영 및 유지보수** : 만들어진 시스템을 실행하고, 시스템 사용중에 발생하는 여러 변경 사항에 대해 적응 · 변화하는 단계

20. Section 121

'계획 수립 → 위험 분석 → 공학화 → 고객 평가' 의 순서로 진행되는 소프트웨어 개발 주기 모델은 나선형 모형이다.

21. Section 121

소프트웨어는 요구나 환경의 변화에 따라 적절히 변형할 수 있는 특성인 '순응성' 을 가지고 있다.

8장 정답 및 해설 — 구조적 분석과 설계

1.②	2.①	3.④	4.②	5.②	6.④	7.①	8.③	9.④	10.③	11.④	12.①	13.①	14.②	15.③
16.②	17.③	18.③	19.④	20.②	21.②	22.①	23.③	24.③	25.③	26.④	27.②	28.④	29.③	30.①
31.③	32.①	33.③	34.①	35.④	36.②	37.③	38.④	39.③	40.④	41.②				

1. Section 123

구조적 분석은 도형 중심의 문서화 도구를 사용함으로써 분석자와 사용자 간의 의사소통이 용이하다.

2. Section 123

구조적 분석 절차 : 현행 시스템의 물리적 모형화 → 현행 시스템의 논리적 모형화 → 새로운 시스템의 논리적 모형화 → 새로운 시스템의 물리적 모형화 → 대안 선택 → 구조적 명세서 작성

3. Section 124

구조적 분석 도구에는 자료 흐름도(DFD), 자료 사전(DD), 소단위 명세서(Mini-Spec.), 개체 관계도(ERD), 상태 전이도(STD) 등이 있다.

4. Section 123

시스템 분석 시 사용자의 요구 사항이나 의견은 중요한 자료가 되므로, 사용자의 참여 없이는 정확한 분석이 불가능하다.

5. Section 124

자료 흐름도의 구성 요소 : 처리(Process), 자료 흐름(Data Flow), 자료 저장소(Data Store), 단말(Terminal, 자료 발생지)

6. Section 124

- 단말은 ▭ 로 표시한다.
- ═ 는 자료 저장소를 의미한다.

7. Section 124

자료 흐름도의 구성 요소와 표기법

- 처리(Process) : ◯
- 자료 흐름(Data Flow) : ⟶
- 자료 저장소(Data Store) : ═
- 단말(Terminal) : ▭

8. Section 124

자료 흐름은 처리를 거쳐 변환될 때마다 새로운 이름을 부여한다.

9. Section 124

자료 흐름도는 자료의 흐름에 중점을 두는 분석용 도구로, 제어 흐름은 중요시하지 않는다.

10. Section 124

개체 관계도(ERD, Entity Relationship Diagram)

시스템에서 처리되는 개체(자료)와 개체의 구성과 속성, 개체 간의 관계를 표현하여 개체를 모델화하는 데 사용되는 것으로, 객체(Entity), 관계(Relationship), 속성(Atrribute) 등으로 구성된다.

11. Section 124

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

12. Section 124

자료 사전(DD, Data Dictionary)은 자료 흐름도의 처리 내용이 아니라 자료 흐름도의 대상이 되는 모든 자료에 대한 기본 사항들을 구체적으로 정의한다.

13. Section 124

'+'는 자료의 연결, '='는 자료의 정의, '[ ]'는 다중 택일, '* *'는 주석이다.

14. Section 124

대체 항목의 나열은 '|' 혹은 ';'이다.

15. Section 124

자료 사전 작성 시 고려 사항

- 이름을 가지고 정의를 쉽게 찾을 수 있어야 한다.
- 이름이 중복되어서는 안 된다.
- 갱신하기 쉬워야 한다.
- 정의하는 방식이 명확해야 한다.
- 중복된 정의가 없어야 한다.

16. Section 124

단계화된 DFD에서 최하위 처리의 입력 자료 흐름을 출력 자료 흐름으로 변환하는 과정을 명세화해 놓은 문서를 소단위 명세서라고 한다. 소단위 명세서 작성 도구에는 구조적 언어, 의사 결정표, 의사 결정도 등이 있다.

17. Section 124

Mini-Spec.(소단위 명세서)은 자료 흐름도의 최하위 처리 절차를 정밀하게 다룬다.

18. Section 124

소단위 명세서(Mini – Spec.)

- 자료 흐름도 상의 최하위 처리 절차를 상세하게 기술하는 데 사용하는 도구이다.
- 구조적 언어, 의사 결정표, 의사 결정도를 이용해 표현한다.

19. Section 124

구조적 방법에서의 세 가지 제어 구조

- **순차 구조** : 순서대로 수행되는 절차로 구성
- **선택 구조** : 조건에 맞는 내용으로 선택하는 형태로 구성
- **반복 구조** : 무조건 분기문을 제외한 조건에 의한 자동 반복문으로 구성

20. Section 124

자료 사전(DD, Data Dictionary)은 자료 흐름도의 대상이 되는 모든 자료에 대한 기본 사항들을 더 자세히 정의하기 위해 사용되는 도구로, 자료 사전의 정의 대상에는 자료 흐름(Data Flow), 자료 요소(Data Element), 자료의 의미, 자료 저장소(Data Store)가 있다.

21. Section 125

구조적 설계 절차 : 구조 도표 작성 → 구조 도표 평가 → 모듈 설계 → 데이터베이스 설계 → 패키징(Packaging)

22. Section 127

모듈의 독립성을 높이려면 결합성을 약하게 하고, 응집성은 강하게 해야 한다.

23. Section 125

- **테스팅(Testing)** : 오류를 발견하기 위해 프로그램이나 시스템을 수행하는 작업
- **디버깅(Debugging)** : 발견된 오류의 성질을 규명하고 수정하는 작업

24. Section 125

구조적 프로그래밍은 가능한 한 GOTO문을 사용하지 않는다.

25. Section 126

- **HIPO** : 시스템 실행 과정인 입력, 처리, 출력을 계층적으로 기술하는 방법
- **구조 도표(Structure Chart)** : 시스템의 구조와 설계를 구체적으로 나타내는 데 사용되는 도표로, 설계 구조도라고도 함
- **프로그램 기술 언어(PDL)** : 구조적으로 설계된 프로그램을 일정한 규칙에 맞게 자연어 형식으로 표현한 것으로, 의사코드 또는 구조적 영어(언어)라고도 함

26. Section 126

N-S Chart는 분기(GOTO) 명령을 사용하지 않는다.

27. Section 126

①은 의사 코드, ③은 N-S 차트, ④는 자료 흐름도에 대한 설명이다.

구조 도표(Structure Chart)

- 소프트웨어의 구조와 설계를 구체적으로 보여주는 역할을 한다.
- 시스템을 모듈이라는 몇 개의 고유 기능으로 분할하여 나타내고, 모듈 간의 인터페이스를 계층 구조로 표현한다.
- 시스템 기능을 모듈의 호출 관계와 인터페이스로 나타낸다.

28. Section 126

프로그램 기술 언어(PDL) = 의사 코드(Pseudo Code)

- 구조적으로 설계된 프로그램을 일정한 규칙에 맞게 자연어 형식으로 표현한 것이다.
- 모듈의 논리를 설계할 때 사고 과정을 체계화해 가는 방법으로, IPT의 배경이 되는 하향식 접근 방식으로 논리의 전체 흐름을 표현한다.

29. Section 126

의사 코드는 프로그래밍 언어와 유사한 서술적 표현에 의하여 작성하는 것으로, 특정 프로그램에 대한 지식이 없어도 작성할 수 있다.

30. Section 125

모듈은 분담하여 독립적으로 작성할 수 있다.

31. Section 125

모듈러 프로그래밍(Modular Programming)은 각각의 모듈의 기능을 어떻게 구현하는가보다는 전체 프로그램이 어떤 기능을 하는가를 중요시 여긴다.

32. Section 126

모듈의 결합도와 응집도를 고려하여 적당한 크기로 모듈화할 때 소프트웨어의 품질을 높일 수 있다.

33. Section 127

의사 결정 분리(Decision Split)란 구조의 인식 부분에서 필요한 데이터와 실행 부분에서 필요한 데이터가 서로 다른 모듈에 분리되어 있는 상태를 의미하는 것으로, 의사 결정 분리는 가능한 한 피하는 것이 좋다.

34. Section 127

모듈(Module)의 독립성을 높이기 위해서는 모듈 간의 결합도는 최소화하고, 모듈 내 요소들 간의 응집력은 최대화한다.

35. Section 127

- **내용 결합도** : 한 모듈이 다른 모듈의 내부 자료를 직접적으로 참조하는 경우의 결합도
- **외부 결합도** : 어떤 모듈에서 외부로 선언되어 있는 자료(변수)를 다른 모듈에서 참조하는 경우의 결합도
- **공통 결합도** : 서로 다른 모듈들이 하나의 기억장소에 설정된 공통의 데이터 영역을 공유하는 경우의 결합도

36. Section 127

가장 낮은 결합도는 자료 결합도이며, 내용 결합도는 결합도가 가장 높다.

37. Section 127

결합도의 순서(낮음 → 높음) : 자료 결합도 → 스탬프 결합도 → 제어 결합도 → 외부 결합도 → 공통 결합도 → 내용 결합도

38. Section 127

응집도가 높을수록 모듈의 독립성도 높아지므로 모듈의 독립성 관점에서는 응집도가 가장 높은 기능적 응집도를 갖는 것이 좋다.

- **우연적 응집도** : 모듈 내부의 각 요소들이 서로 관계없는 것끼리 모인 경우
- **시간적 응집도** : 특정 시간에 처리되는 몇 개의 기능을 모아 하나의 모듈로 구성한 경우
- **절차적 응집도** : 일정한 순서에 의해 처리되어야 할 요소들을 하나의 모듈로 구성한 경우로, 전달 데이터와 반환 데이터 사이에 관련이 없고 처리 기능보다는 실행 순서에 의해 연결됨
- **기능적 응집도** : 모듈 내부의 모든 기능 요소가 한 가지의 작업만을 수행하는 경우

39. Section 127

- **논리적 응집도** : 논리적으로 서로 관련있는 요소들을 모아 하나의 모듈로 작성하여, 그 모듈의 기능이 매개변수에 따라 처리 내용이나 처리 루트가 달라지게 되는 경우
- **통신적 응집도** : 동일한 입 · 출력 데이터를 이용하여 서로 다른 기능을 수행하는 요소들로 구성되며, 요소들 사이에 데이터를 주고받거나 데이터를 똑같이 공유함

- **순차적 응집도** : 모듈 내부의 한 기능 요소에 의한 출력 데이터가 다음 기능 요소에 입력 데이터로 제공되는 경우

40. Section 127

- **제어 결합도** : 서로 다른 모듈 간에 교환하는 매개변수(Parameter)가 제어 정보인 경우의 결합도
- **외부 결합도** : 어떤 모듈에서 외부(External)로 선언되어 있는 자료(변수)를 다른 모듈에서 참조하는 경우의 결합도
- **공통 결합도** : 서로 다른 모듈들이 하나의 기억장소에 설정된 공통의 데이터 영역을 공유하는 경우의 결합도

41. Section 127

- **자료 결합도(Data Coupling)** : 모듈 간의 인터페이스가 자료 요소로만 구성될 때의 결합도
- **제어 결합도(Control Coupling)** : 한 모듈에서 다른 모듈로 논리적인 흐름을 제어하는 데 사용하는 제어 요소(Function code, Switch, Tag, Flag)가 전달될 때의 결합도
- **외부 결합도(External Coupling)** : 어떤 모듈에서 외부로 선언한 데이터(변수)를 다른 모듈에서 참조할 때의 결합도

9장 정답 및 해설 — 객체지향 분석과 설계

1.②	2.①	3.③	4.②	5.①	6.②	7.②	8.③	9.④	10.④	11.④	12.①	13.②	14.③	15.②
16.③	17.④	18.①	19.②	20.③	21.③	22.③	23.④	24.②	25.④	26.②	27.②			

1. Section 128

두 개 이상의 공통된 특징을 갖는 객체들을 모아둔 것을 클래스라고 한다.

2. Section 128

- **클래스(Class)** : 두 개 이상의 유사한 객체들을 묶어서 하나의 공통된 특성을 표현하는 요소
- **메시지(Message)** : 외부로부터 하나의 객체에 전달되는 메소드(행위)의 요구

3. Section 128

- **인스턴스(Instance)** : 하나의 클래스에 속하는 각각의 객체
- **메소드(Method)** : 객체에 정의된 연산을 의미하며, 객체의 상태를 참조하거나 변경하는 수단이 됨

4. Section 128

- **속성(Attribute)** : 한 클래스 내에 속한 객체들이 가지고 있는 데이터 값들을 단위별로 정의하는 것으로서 성질, 분류, 식별, 수량 또는 현재 상태 등을 표현
- **메시지(Message)** : 외부로부터 하나의 객체에 전달되는 메소드(행위)의 요구
- **인스턴스(Instance)** : 하나의 클래스에 속하는 각각의 객체

5. Section 128

- **메소드(Method)** : 객체에 정의된 연산을 의미하며, 객체의 상태를 참조하거나 변경하는 수단이 됨
- **상속성(Inheritance)** : 이미 정의된 상위 클래스의 메소드를 비롯한 모든 속성을 하위 클래스가 물려받을 수 있는 것
- **인스턴스(Instance)** : 하나의 클래스에 속하는 각각의 객체

6. Section 128

- 문제의 내용은 캡슐화를 의미한다.
- 보기 ②의 상위 클래스의 메소드를 비롯한 모든 속성을 하위 클래스가 물려받을 수 있음을 의미하는 것은 상속성이다.

7. Section 128

캡슐화되고 정보 은닉된 객체는 객체 제공자가 그 내부를 변경하더라도 이용자의 원시 프로그램에는 영향을 주지 못하므로, 유지보수 시 일부 변경에 의한 부작용을 최소화할 수 있다.

8. Section 128

럼바우의 객체 모델링 기법(OMT)의 분석 절차 : 객체 모델링 → 동적 모델링 → 기능 모델링

9. Section 128

추상화(Abstraction)는 불필요한 부분을 생략하고 객체의 속성 중 가장 중요한 것에 중점을 두어 개략화하는 것으로, 데이터의 공통된 성질을 추출하여 슈퍼 클래스를 선정한다.

10. Section 128

객체지향 기법은 구조적 방법에서의 생산성 문제를 해결하기 위한 새로운 방법이다.

11. Section 128

Cohesion(응집도)은 구조적 방법에서의 설계도 평가 척도이다.

12. Section 128

- 메시지(Message) : 외부로부터 하나의 객체에 전달되는 메소드(연산)의 요구
- 상속성(Inheritance) : 이미 정의된 상위 클래스의 메소드를 비롯한 모든 속성을 하위 클래스가 물려받을 수 있는 것

13. Section 128

메시지(Message)는 외부로부터 하나의 객체에 전달되는 메소드(행위)의 요구를 의미하며, 목적지는 메시지를 받을 객체, 연산은 메시지를 받으려는 동작, 인자는 연산이 성공하는 데 필요한 정보를 의미한다.

14. Section 128

추상화(Abstraction)의 종류

- 기능(Function) 추상화 : 가시적인 루틴은 타 그룹과 통신하게 하고 구체적인 기능은 은폐
- 제어(Control) 추상화 : 효과는 정의하고 메커니즘은 은폐
- 자료(Data) 추상화 : 자료에 대한 연산은 정의하고 처리 내용은 은폐

15. Section 128

객체지향 분석 절차

① 객체 모델링 : 시스템에 요구되는 객체를 추출하고, 객체들 간의 관계를 정의하여 객체 다이어그램으로 나타냄

② 동적 모델링 : 시간의 흐름에 따라 변하는 객체들 사이의 동적인 행위를 상태 전이도로 나타냄

③ 기능 모델링 : 정보의 처리 과정을 데이터 흐름도로 나타냄

16. Section 128

- 클래스(Class) : 2개 이상의 유사한 객체들을 묶어서 하나의 공통된 특성을 표현하는 요소
- 메시지(Message) : 외부로부터 하나의 객체에 전달되는 메소드(연산)의 요구
- 상속성(Inheritance) : 이미 정의된 상위 클래스의 메소드를 비롯한 모든 속성을 하위 클래스가 물려받을 수 있는 것

17. Section 128

객체의 속성

- 주체성 : 다른 객체들과 식별할 수 있는 속성
- 분류성 : 동일 속성과 행위를 갖는 객체들을 하나의 클래스로 분류하는 속성
- 다형성 : 같은 연산자라도 각 클래스에 따라 다른 기능을 수행할 수 있는 속성
- 상속성 : 상위 클래스의 내용을 하위 클래스에 물려주는 성질

18. Section 129

객체 모델링 순서 : 객체 식별 → 객체의 속성 기술 → 연산의 정의 → 객체들 간의 관계 정의 → 클래스의 계층화 → 모듈화

19. Section 129

동적 모델링 순서 : 사건 시나리오 작성 → 사건 추적도 작성 → 사건 흐름도 작성 → 상태 전이도 작성

20. Section 129

기능 모델링 순서 : 입 · 출력 데이터의 정의 → 자료 흐름도 작성 → 기능의 정의 → 제약 조건 파악 → 최적화 기준 명세화

21. Section 128

인스턴스(Instance)는 하나의 클래스에 속하는 각각의 객체를 의미하며, 객체는 자료 구조와 이를 처리하기 위한 연산을 결합시킨 실체이다.

22. Section 128

객체지향 기법은 소프트웨어 개발 및 유지보수를 용이하게 하는 기법이다.

23. Section 128

추상화를 함으로써 실제 상황 모델의 복잡성을 감소시킬 수 있다.

24. Section 129

우선순위 결정은 시스템 설계 단계에서 수행하는 작업이다.

25. Section 129

외부와 상호 작용하기 위한 제어 방식을 구현하는 것은 객체 설계 단계에서 수행한다. 시스템 설계 단계에서는 시스템의 제어 방식을 결정한다.

26. Section 128

- **객체** : 데이터(속성)와 이를 처리하기 위한 연산(메소드)을 결합시킨 실체
- **다형성** : 하나의 메시지에 대해 각 클래스가 가지고 있는 고유한 방법으로 응답할 수 있는 능력. 즉 같은 연산자라도 각 클래스에 따라 다른 기능을 수행할 수 있는 특성
- **상속성** : 이미 정의된 상위 클래스의 메소드를 비롯한 모든 속성을 하위 클래스가 물려받을 수 있는 것

27. Section 128

객체지향 기법의 장 · 단점

- 상속을 통한 재사용과 시스템의 확장이 용이하다.
- 자연적인 모델링에 의해 분석과 설계를 쉽고 효율적으로 할 수 있다.
- 사용자와 개발자 사이의 이해를 쉽게 해준다.
- 대형 프로그램의 작성이 용이하다.
- 소프트웨어 개발 및 유지보수가 용이하다.
- 복잡한 구조를 단계적, 계층적으로 표현할 수 있으므로 생명주기 상에서 일관적으로 나타낼 수 있다.
- 프로그래밍 구현을 지원해 주는 정형화된 분석 및 설계 방법이 없다.
- 구현 시 처리 시간이 지연된다.
- 공통된 속성을 명백히 표현할 수 있다.
- 객체 간의 종속성을 극소화할 수 있다.

5과목 정보 통신 개론

1장 정답 및 해설 — 정보 통신의 기본

1.③ 2.② 3.③ 4.③ 5.① 6.① 7.④ 8.④ 9.② 10.③ 11.④ 12.② 13.③ 14.③ 15.④
16.① 17.③ 18.① 19.③ 20.② 21.② 22.②

1. Section 130

세계 최초의 상업용 데이터 통신 시스템은 SABRE이다. SAGE는 최초의 데이터 통신 시스템으로 군사용 시스템이다.

2. Section 130

- 유선 통신 : 꼬임선, 동축 케이블, 광섬유 케이블 등의 유선 매체를 이용한 통신
- 전령 통신 : 사람이 직접 도보나 역마차를 이용하여 정보를 전송하는 통신
- 무선 통신 : 지상 마이크로파, 위성 마이크로파 등의 무선 매체를 이용한 통신

3. Section 130

정보화란 컴퓨터 및 통신 기술을 이용하여 일상 생활에서 사용되는 각종 정보의 유용 가치를 높이는 활동을 의미한다.

4. Section 130

정보의 가치는 연속적인 정보 활동과 축적에 의해 계속적으로 높아진다.

5. Section 130

노동 경제성은 노동의 투자로 인해 얻는 이득의 정도를 나타내는 것이다. 정보 통신을 이용한 정보의 이용은 노동보다는 컴퓨터 및 통신 기술을 이용해 수행하는 것으로, 노동 경제성의 향상으로 볼 수 없다.

6. Section 130

데이터와 정보의 진화 과정 : 데이터(Data) → 정보(Information) → 지식(Knowledge) → 지능(Intelligence)

7. Section 130

정보 통신 시스템의 기능

- 거리와 시간의 한계를 극복
- 시분할 방식에 의한 대형 컴퓨터의 공동 이용
- 분산 처리 환경을 이용한 자원의 공유 및 작업의 분담

8. Section 131

컴퓨터는 데이터 처리계에 속한다.

9. Section 130

- TELNET : 멀리 떨어져 있는 컴퓨터에 접속하여 자신의 컴퓨터처럼 사용할 수 있도록 해주는 서비스
- SNA : IBM에서 발표한 컴퓨터 간 접속 네트워크 시스템 표준으로, 이로 인해 데이터 통신 시스템의 표준화가 시작되었다.
- ARPANET : 미 국방성에서 설치한 최초의 유선 패킷 교환 시스템으로, 인터넷의 효시가 된 시스템

10. Section 131

- 정보 통신 시스템의 기본 구성 요소는 단말장치, 전송 장치(신호 변환장치, 통신 회선), 통신 제어장치, 컴퓨터(정보 처리 시스템)이다.
- 다중화 장치는 하나의 고속 통신 회선을 다수의 단말장치가 공유할 수 있도록 하는 장치이다.

11. Section 130

통신의 3요소

- 정보원(Source) : 정보를 입력받아 전송하는 곳
- 수신원(Destination) : 전송된 정보를 수신하는 곳
- 전송 매체(Transmission Medium) : 정보원과 수신원을 연결하는 매체(통신 회선)

12. Section 131

- **신호 변환장치** : 단말장치나 통신 제어장치로부터 들어온 신호를 통신 회선에 적합한 신호로 변환하는 장치
- **다중화 장치** : 하나의 통신 회선을 여러 개의 단말장치가 동시에 접속하여 사용할 수 있도록 하는 장치
- **단말장치** : 데이터 통신 시스템과 외부 사용자의 접속점에 위치하여 최종적으로 데이터를 입 · 출력하는 장치

13. Section 131

단말장치에서 전송한 디지털 데이터를 아날로그 신호로 변환한 후 음성 전송용으로 설계된 전송로에 송신하는 신호 변환장치는 모뎀(MODEM)이다.

※ 광섬유 케이블에 대한 설명은 5-43쪽을 참고하세요.

14. Section 132

실시간 처리가 가능해진 것은 온라인 시스템의 발전에 의해서이다.

15. Section 132

저장 매체를 반드시 필요로 하는 시스템은 오프라인 시스템이다.

16. Section 132

- **온라인 처리 시스템** : 데이터가 발생한 단말장치와 데이터를 처리할 컴퓨터가 통신 회선을 통해 직접 연결된 형태
- **일괄 처리 시스템** : 일정 양 또는 일정 기간 동안 데이터를 모아서 한꺼번에 처리하는 방식
- **다중 처리 시스템** : 여러 개의 CPU와 하나의 주기억장치를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식

17. Section 132

- Simplex System : 시스템을 구성하는 구성 요소들이 각각 한 개씩만 설치되어 있는 시스템
- Duplex System : 두 대의 CPU를 사용하되 한 쪽의 CPU가 가동중일 때에는 다른 한 CPU는 대기하게 되며, 가동 중인 CPU가 고장나면 즉시 대기중인 한쪽 CPU가 가동되어 시스템이 안전하게 작동되도록 운영하는 방식
- Multiprocessor System : 프로세서를 여러 개 두어 업무를 분담하여 처리할 수 있는 방식

19. Section 132

온라인 시스템(On-Line System)

- 데이터가 발생한 단말장치와 데이터를 처리할 컴퓨터가 통신 회선을 통해 직접 연결된 형태로, 데이터 송 · 수신 중간에 사람 혹은 기록 매체가 개입되지 않는다.
- 정보 통신 업무의 대부분을 차지하는 실시간 처리가 요구되는 작업에 주로 사용된다.
- 단말장치, 중앙처리장치, 통신 제어장치, 통신 회선 등으로 구성된다.

20. Section 131

정보 통신 시스템의 3대 구성 요소는 단말장치, 전송장치(전송 회선, 통신 제어장치), 컴퓨터이다. 교환장치는 정보 통신망의 3대 요소에 해당한다.

21. Section 130

정보 통신에 사용되는 정보의 형태는 문자나 부호뿐만이 아니라 음성, 그림, 영상 등을 모두 포함한다.

22. Section 130

이중화된 장치나 선로를 사용하면 위험으로부터 데이터를 안전하게 보호하여 오류 발생을 최소화할 수 있다. 정보통신 시스템은 이러한 에러 제어 방식을 사용하므로 신뢰도가 높다.

2장 정답 및 해설 — 정보 통신 기기

1.③	2.①	3.②	4.④	5.①	6.③	7.④	8.④	9.④	10.②	11.③	12.④	13.③	14.④	15.③
16.③	17.③	18.④	19.④	20.③	21.②	22.④	23.③	24.②	25.③	26.④	27.④	28.④	29.④	30.③
31.②	32.③	33.②	34.①	35.③	36.②	37.②	38.②	39.④	40.②					

1. Section 133

단말장치의 전송 제어 기능에는 입 · 출력 제어 기능, 회선 제어 기능, 회선 접속 기능이 있다.

2. Section 133

중앙 컴퓨터에 모든 처리를 의존하는 종속적인 단말장치는 비지능형(더미) 단말장치이다.

3. Section 134

통신 제어장치는 데이터 전송 회선(통신 회선, 신호 변환장치)과 컴퓨터 사이에 위치한다. 정확한 위치를 말하자면 통신 회선 → 신호 변환장치(DCE) → 통신 제어장치(CCU) → 컴퓨터 순이다.

4. Section 134

컴퓨터에서 처리된 데이터를 통신 회선에 적합한 신호로 변경하는 것은 신호 변환장치의 기능이다.

5. Section 134

통신 회선을 통해 송 · 수신되는 데이터를 저장하고 처리하는 장치는 컴퓨터이다.

6. Section 134

사용자가 컴퓨터와 대화할 수 있는 창구 역할을 하는 것은 DTE(단말장치)이다.

7. Section 134

단말장치와 컴퓨터를 물리적으로 연결하는 것은 통신 회선이다.

8. Section 135

DSU(Digital Service Unit)는 디지털 데이터를 공중 데이터 교환망(PSDN)을 이용하여 전송할 때 사용한다.

9. Section 135

- 동기식 모뎀은 위상 편이 변조(PSK) 또는 직교 진폭 변조(QAM) 방식을 사용한다.
- 주파수 편이 변조(FSK) 방식을 사용하는 것은 비동기식 모뎀이다.

10. Section 135

DSU는 신호의 변조 과정 없이 단순히 유니폴라 신호를 바이폴라 신호로 변환해 주는 기능만 제공하기 때문에 모뎀에 비하여 설계가 단순하다.

11. Section 135

아날로그 데이터를 디지털 신호로 변환하는 것을 코더(Coder), 디지털 데이터를 아날로그 신호로 변환하는 것을 디코더(Decoder)라고 한다.

12. Section 135

기존의 음성 통신망을 이용하여 디지털 데이터를 전송하기 위해서는 모뎀(MODEM)이 사용된다.

13. Section 136

RS-232C

- DTE와 DCE 사이의 접속 규격
- 25핀으로 구성된 커넥터로, 전송 거리는 15m 이하이다.
- 데이터 신호 속도는 최고 20Kbps이다.
- 전이중/반이중, 동기/비동기 모두에 대응한다.

14. Section 136

RS-232C의 데이터 신호 속도는 20Kbps 이하이고, 전이중/반이중, 동기/비동기 모두에 대응한다.

15. Section 136

V.24, V.28, RS-232C는 아날로그 통신에 관한 표준 권고안이다.

16. Section 136

RS-232C는 데이터의 전송 속도가 20Kbps 이하로 느리고, 전송 거리는 15m 이하로 제한된다.

17. Section 136

X.21은 공중 데이터 교환망에서의 동기식 전송을 위한 DTE/DCE 간의 인터페이스 규격이다.

18. Section 136

- TXD(Transmitted Data) : 데이터를 송신하는 기능
- RTS(Request To Send) : DTE에서 DCE한테 송신을 요청하는 기능
- DSR(Data Set Ready) : DCE의 동작 상태를 알리는 기능

19. Section 136

DTE와 DCE 간의 절차적 조건은 신호의 전송 절차 등에 관한 특성을 규정한다.

20. Section 136

모뎀을 단말장치에 접속할 때이므로 공중 전화 교환망을 통한 표준안을 의미하는 것이다. X.25는 공중 데이터 교환망을 통한 표준안이므로, 모뎀이 아닌 DSU를 단말장치에 접속할 때 사용한다.

21. Section 136

- DCD(Data Carrier Detect) : DCE가 선로 쪽으로부터 감지할 수 있는 크기의 신호를 수신하고 있음을 DTE에게 알림
- DTR(Data Terminal Ready) : DTE가 정상적인 동작 상태에 있음을 DCE에게 알림
- CTS(Clear To Send) : DCE가 송신 준비를 완료했음을 DTE에 알림

22. Section 137

주파수 분할 다중화기 자체에는 변 · 복조 기능이 내장되어 있으므로 별도의 모뎀이 필요하지 않다.

23. Section 137

다중화(Multiplexing)는 하나의 통신 회선을 여러 개의 단말장치가 동시에 접속하여 사용할 수 있도록 하는 기능으로, 통신 회선을 공유함으로써 전송 효율을 높일 수 있다.

24. Section 137

CSMA는 매체 접근 제어(MAC) 방식에 속한다.

25. Section 137

동기식 시분할 다중화는 모든 단말장치에 균등한(고정된) 시간폭(Time Slot)을 제공하므로 전송할 데이터가 없는 경우에는 시간폭(Time Slot)이 낭비된다.

26. Section 137

- FDM(주파수 분할 다중화 방식)은 아날로그 신호 전송에 사용한다.
- 반송 주파수란 통신 회선의 특성에 알맞은 신호로 변조하기 위하여 사용되는 기준 파형을 말한다.

27. Section 137

- **코드 분할 다중화** : 스펙트럼 확산 방식을 이용하여 넓은 주파수 대역에 다수의 단말장치가 서로 다른 코드를 사용함으로써 동일한 주파수로 동시에 데이터를 전송하는 방식으로, 동일한 주파수로 동시에 데이터를 전송할 수 있으므로 주파수 이용 효율이 좋으며, 서로 다른 코드를 이용할 수 있으므로 보안이 우수함
- **주파수 분할 다중화** : 통신 회선의 주파수를 여러 개로 분할하여 여러 대의 단말 장치가 동시에 사용할 수 있도록 하는 다중화 기법
- **동기식 시분할 다중화** : 모든 단말장치에 균등한 시간폭을 제공

28. Section 137

④는 동기식 시분할기에 대한 설명이다.

29. Section 137

통계 시분할 다중화기는 회선의 사용 효율이 높다는 장점이 있지만, 트래픽이 모든 단말기에서 연속적으로 발생하는 경우 오히려 성능 저하를 가져올 수 있고 전송 지연이 발생할 수 있다.

30. Section 137

지능 다중화기는 기능이 복잡하므로 비용이 비싸고, 접속에 소요되는 시간이 길다.

31. Section 137

역다중화기(Inverse Multiplexer)는 광대역 통신 회선을 사용하지 않고 음성 대역 회선을 이용하여 9,600 Bps 이상의 광대역 속도를 얻을 수 있으므로, 비용을 절감할 수 있다.

32. Section 137

집중화기(Concentrator)는 불규칙적인 전송에 적합하다.

33. Section 131

송신 측 터미널의 데이터는 신호 변환장치(모뎀 등)에 의해 데이터 전송로에 적합한 신호로 변환되어 전송되고, 수신 측의 신호변환장치(모뎀 등)에 의해 다시 수신 측 컴퓨터에 적합한 데이터로 변환된다. 그리고 전송로를 통해 전송된 데이터는 통신 제어장치에서 컴퓨터가 처리하기 적합한 단위로 조립 및 분해된다.

34. Section 137

②, ③, ④는 주파수 분할 다중화기(FDM)의 특징이다.

35. Section 135

전화기에서 음성을 전기 신호로 변환시켜 주는 장치는 송화기이고, 전기 신호를 다시 음성으로 변환시켜주는 장치는 수화기이다.

36. Section 136

X 시리즈는 공중 데이터 교환망(PSDN)을 통한 DTE/DCE 접속 규격이다.

37. Section 137

다중화(Multiplexing)란 하나의 고속 통신 회선을 다수의 단말장치가 공유할 수 있도록 하는 기술이다.

38. Section 137

- 시분할 다중화 방식 : 통신 회선의 대역폭을 일정한 시간 폭으로 나누어 여러 대의 단말장치가 동시에 사용할 수 있도록 한 방식
- 코드 분할 다중화 방식 : 스펙트럼 확산 방식을 이용하여 넓은 주파수 대역에 다수의 단말장치가 서로 다른 코드를 사용함으로써 동일한 주파수로 동시에 데이터를 전송하는 방식
- 공간 분할 다중화 방식 : 공간적으로 분리된 여러 개의 물리적인 채널을 하나의 논리적인 채널로 다중화하는 방식

39. Section 137

① 가드 밴드는 채널의 증가가 아니라 각 채널들 간의 상호 간섭을 방지하기 위해 사용한다.
② 대역 확산 방식을 이용한 다중화 방식은 코드 분할 다중화 방식이다.
③ 각 채널에 고정된 Time Slot을 할당하는 방식은 동기식 시분할 다중화 방식이다.

40. Section 136

- X.20 : 비동기식 전송을 위한 DTE/DCE 접속 규격
- X.22 : 다중 DTE/DCE 접속 규격
- X.25 : 패킷 전송을 위한 DTE/DCE 접속 규격

3장 정답 및 해설 — 정보 전송 기술

1.①	2.②	3.③	4.②	5.④	6.④	7.①	8.④	9.②	10.④	11.③	12.①	13.④	14.①	15.③
16.②	17.④	18.①	19.③	20.①	21.④	22.②	23.③	24.④	25.①	26.④	27.①	28.③	29.②	30.④
31.②	32.③	33.①	34.③	35.④	36.②	37.③	38.④	39.①	40.③	41.③	42.③	43.④	44.④	45.①
46.③	47.④	48.④	49.②	50.④										

1. Section 140

광섬유 케이블은 동축 케이블보다 신호 감쇠 현상이 적다.

2. Section 140

유선 매체 중 가장 빠른 속도와 높은 주파수 대역폭을 제공하는 것은 광섬유 케이블이다.

3. Section 140

동축 케이블(Coaxial Cable)

- 중심 도체를 플라스틱 절연체를 이용하여 감싸고, 이를 다시 외부 도체를 이용하여 감싸는 형태로 구성된다.
- 아날로그와 디지털 신호 전송에 모두 사용한다.
- 주파수 범위가 넓어서 데이터 전송률이 높다.
- 외부 간섭과 누화에 둔감하다.
- 고주파 특성이 양호하며, 광대역 전송에 적합하다.
- CATV, 근거리 통신망 등 용도가 다양하다.
- 신호의 감쇠 현상을 막기 위해 일정 간격마다 중계기를 설치해야 한다.

4. Section 140

광섬유 케이블은 크기가 작고 무게가 가볍지만 설치 시 고도의 기술이 필요하고, 광섬유 케이블 자체가 고가이므로 초기 비용이 많이 든다.

5. Section 150

HDLC 프레임 주소부에서 모든 스테이션에게 프레임을 전송하는 방송용 값은 '11111111' 이고 시스템에 의해 임의로 수신국이 지정되는 시험용 값은 '00000000' 이다.

6. Section 141

bps = baud × 변조 가능 비트 수

- 변조 가능 비트 수 = 트리비트(Tribit) = 3Bit
- bps = 2,400 × 3 = 7,200

7. Section 141

- 위상이 4개면 한번에 4개의 서로 다른 데이터를 보낼 수 있다는 의미이고, 4개의 데이터라면 한번에 2진수 2비트로 표현할 수 있다($2^2=4$).
- Baud = 9,600/2 = 4,800

8. Section 141

베어러 속도는 데이터 신호에 동기 문자, 상태 신호 등을 합한 속도로, 단위는 bps(bit/sec)를 사용한다.

9. Section 142

디지털 신호는 2진수 0과 1에 대한 전압 펄스로, 전압의 높낮이가 일정하다.

10. Section 143

디지털 변조 방식

- **진폭 편이 변조(ASK)** : 2진수 0과 1을 서로 다른 진폭의 신호로 변조하는 방식
- **주파수 편이 변조(FSK)** : 2진수 0과 1을 서로 다른 주파수로 변조하는 방식
- **위상 편이 변조(PSK)** : 2진수 0과 1을 서로 다른 위상을 갖는 신호로 변조하는 방식
- **직교 진폭 변조(QAM)** : 진폭과 위상을 상호 변환하여 신호를 얻는 변조 방식

11. Section 142

- **표본화** : 음성, 영상 등의 연속적인 신호 파형을 일정 시간 간격으로 검출하는 단계
- **양자화** : 표본화된 PAM 신호를 유한 개의 부호에 대한 대표값으로 조정하는 과정

12. Section 143

진폭 편이 변조(ASK)

- 2진수 0과 1을 서로 다른 진폭의 신호로 변조하는 방식이다.
- 이 방식을 사용하는 모뎀은 구조가 간단하고, 가격이 저렴하다.
- 신호 변동과 잡음에 약하여 데이터 전송용으로 거의 사용되지 않는다.

13. Section 145

베이스밴드(Base Band) 전송 방식은 감쇠 현상이 심하여 원거리 전송에 부적합하다.

14. Section 144

- 1μs는 1/1,000,000s이다.
- 표본화 간격은 1/표본화 횟수인데, 표본화 횟수는 최대 주파수의 2배이므로 1/2,000 = 0.0005이다.
  ∴ 표본화 주기는 0.0005×1,000,000 = 500[μs]이다.

15. Section 144

- 펄스 변조 방식 중 연속 레벨 변조 방식은 아날로그 변조 방식으로 PAM, PWM, PPM이 있다.
- PCM은 펄스 부호 변조라고 하며, 불연속 레벨 변조에 속한다.

16. Section 144

- **부호화** : 양자화된 PCM 펄스의 진폭 크기를 2진수(1과 0)로 표시하는 과정
- **여파화** : PAM 신호를 원래의 입력 신호인 아날로그 신호로 복원하는 과정
- **양자화** : 표본화된 PAM 신호를 유한 개의 부호에 대한 대표값으로 조정하는 과정

17. Section 144

수신된 디지털 신호(PCM 신호)를 PAM 신호로 되돌리는 것을 복호화라고 한다.

18. Section 146

디지털 전송 시에도 감쇠 현상은 나타나지만 중계기에 의해 원래의 신호 내용을 복원한 다음 전송하는 방식이기 때문에 잡음에 의한 오류율이 낮다.

19. Section 146

병렬 전송 방식은 전송 속도는 빠르지만 구성 비용이 비싸므로, 컴퓨터와 프린터 같이 가까운 거리를 연결할 때 사용된다.

20. Section 146

라디오나 TV 방송과 같이 한쪽에서는 송신만 하고 다른 한쪽에서는 수신만 하는 방식을 단방향(단향) 통신이라고 한다.

21. Section 146

①, ②는 반이중(Half-Duplex) 통신, ③은 단방향(Simplex) 통신에 대한 설명이다.

22. Section 146

전송량이 많고, 전송 매체의 용량이 클 때는 전이중 통신 방식을 사용한다.

전이중(Full-Duplex) 통신

- 동시에 양방향 전송이 가능한 방식이다.
- 4선식 선로를 사용하며, 주파수 분할을 이용할 경우 2선식도 가능하다.
  예 전화, 전용선을 이용한 데이터 통신

23. Section 152

- **순환 중복 검사(CRC)** : 다항식 코드를 사용하여 오류를 검출하는 방식
- **자동 연속 방식** : 수신 측에서 동일한 데이터를 두 번 이상 전송하면 수신 측에서 두 데이터를 비교해 이상 유무를 판별한 후 오류 발생 시 이를 수정하는 방식
- **궤환 전송 방식(Echo Check)** : 수신 측에서 받은 데이터를 송신 측으로 되돌려 보내어, 원본 데이터와 비교하여 오류가 있는 경우 재전송하는 방식

24. Section 146

비동기식 전송은 전송되는 비트의 양이 많거나 한 번에 보내지는 비트의 길이가 길어지는 경우 송신 측과 수신 측의 샘플링(Sampling) 시점이 달라서 발생하는 프레이밍 에러의 가능성이 높아진다.

25. Section 147

폴링/셀렉션(Polling/Selection) 방식

- 컴퓨터에서 송 · 수신 제어권을 가지고 있는 방식이다.
- 트래픽이 많은 멀티 포인트(Multi-Point) 방식으로 연결된 회선에서 사용한다.

26. Section 147

④는 폴링/셀렉션 방식에 대한 설명이다.

27. Section 150

HDLC 프로토콜은 비트 방식 프로토콜이다.

28. Section 149

- SOH(Start of Heading) : 헤딩의 시작을 의미
- STX(Start of Text) : 본문의 시작 및 헤딩의 종료

29. Section 146

시작 비트와 종료 비트를 이용하여 동기화하는 것은 비동기식 전송 방식이다.

30. Section 146

데이터 블록의 처음과 끝에 플래그(Flag)를 표시하여 동기를 맞추는 방식은 비트 위주 동기 방식이다.

31. Section 147

포인트 투 포인트(Point-to-Point) 방식은 주 컴퓨터와 단말기를 일 대 일 독립적으로 연결하므로 고장 발생 시 보수가 쉽다.

32. Section 150

플래그(Flag)의 역할

- 프레임의 시작과 끝을 구분
- 동기 유지(통화로의 혼선을 방지하기 위해)
- 비트 투과성을 이용한 기본적인 오류 검출

33. Section 149

BSC는 동기식 전송 방식에 사용되며, 반이중 통신만을 지원한다.

34. Section 151

- 문제의 하드와이어 전송 매체는 유선 매체를 의미한다.
- **감쇠(Attenuation)** : 전송 신호 세력이 전송 매체를 통과하는 과정에서 거리에 따라 약해지는 현상
- **누화 잡음** : 인접한 전송 매체의 전자기적 상호 유도 작용에 의해 생기는 오류

35. Section 151

블록을 전송할 때마다 수신 측의 응답을 기다리는 것은 정지-대기(Stop-and-Wait) ARQ이다.

적응적(Adaptive) ARQ

- 전송 효율을 최대로 하기 위해서 블록 길이를 채널의 상태에 따라 그때그때 동적으로 변경하는 방식이다.
- 전송 효율이 제일 좋지만, 제어 회로가 매우 복잡하므로 현재 거의 사용되지 않는다.

36. Section 152

CRC(Cyclic Redundancy Check, 순환 중복 검사)

- 다항식 코드를 사용하여 오류를 검출하는 방식이다.
- 동기식 전송에서 주로 사용된다.
- HDLC 프레임의 FCS(프레임 검사 순서 필드)에 사용되는 방식이다.
- 집단 오류를 검출할 수 있고, 검출률이 높으므로 가장 많이 사용된다.

37. Section 151

전진 오류 수정(FEC, Forward Error Correction)에서 오류를 수정하기 위한 방식에는 해밍 코드 방식과 상승 코드 방식이 있다.

- 압축(Compression) : 디스크 공간을 효율적으로 사용하거나 통신 시 파일 전송 시간 및 비용의 절감을 위해 중복되는 데이터를 이용하여 파일의 크기를 줄이는 것
- 패리티 비트(Parity Bit) : 데이터 전송 시 오류를 검출하기 위하여 추가하는 오류 검출 비트
- Huffman Coding : 데이터 압축 기법의 한 가지로, 압축하려는 정보 내의 각 문자에 대한 발생 빈도를 조사해 자주 나타나는 문자에는 짧은 부호를 할당하고, 잘 나타나지 않는 문자에는 긴 부호를 할당하는 방식

40. Section 143

- ASK(Amplitude Shift Keying) : 2진수 0과 1을 서로 다른 진폭의 신호로 변조하는 방식
- DM(Delta Modulation) : 펄스 변조 중 불연속 레벨(디지털) 변조에 속하는 것으로 바로 앞의 신호보다 현재 신호의 진폭이 작으면 음의 펄스로, 반대로 크면 양의 펄스로 바꾸는 방식
- ADPCM(Adaptive Differential PCM) : 음향이나 아날로그 정보를 2진수 0 또는 1로 변환시키는 적응 차등 펄스 부호 변조 방식

41. Section 144

샤논의 정의인 $C = W \cdot \log_2(1+S/N)$[bps]에 대입하면 $3000 \cdot \log_2(1+15) = 3000 \cdot \log_2(16) = 3000 \cdot 4 = 12000$[bps]가 된다.

42. Section 150

프레임의 구조 중 오류 검출을 위한 FCS는 16Bit 또는 32Bit로 구성된다.

43. Section 150

- SDLC는 HDLC와 같은 프레임 구조를 갖는다.
- 프레임 구조를 순서대로 나열하면 '플래그, 주소부, 제어부, 정보부, FSC, 플래그' 순이다.

44. Section 146

항상 한 묶음으로 구성된 문자 사이의 휴지 간격이 존재하는 것은 비동기식 전송 방식이다.

45. Section 152

순환 중복 검사(CRC) 방식은 동기식 전송에 사용되는 에러 검출 기법으로, 문자 단위가 아니라 프레임 단위로 전송될 때 사용된다.

46. Section 140

꼬임선은 동축 케이블이나 광섬유 케이블에 비해 거리, 대역폭, 전송속도 면에서 상대적으로 많은 제약을 갖는다.

47. Section 146

HDLC에서 사용되는 프레임에는 정보(Information) 프레임, 감독(Supervisor) 프레임, 비(무)번호(Unnumbered) 프레임이 있다.

48. Section 146

주로 장거리 전송에 이용되며 고속 전송에 적합한 것은 동기식 전송 방식이다.

49. Section 151

- 흐름 제어 : 데이터가 효율적인 속도로 처리될 수 있도록 데이터의 흐름을 제어하는 기능
- 링크 제어 : 물리적인 전송 매체를 통해 데이터가 안정적으로 전송될 수 있도록 에러 검출 및 교정, 흐름 제어 등을 수행하는 기능
- 회선 제어 : 각 장치들이 회선을 통해 데이터를 전송할 때 충돌이 발생하지 않도록 송 · 수신 시 필요한 규칙을 정해 이를 기준으로 제어하는 기능

50. Section 140

광을 검출한다는 것은 전송되어 온 빛(신호)을 검출한다는 것으로 수신 측 요소를 의미한다. APD는 PD의 종류 중 하나이다.

4장 정답 및 해설 — 통신 프로토콜

1.③	2.④	3.②	4.②	5.②	6.①	7.①	8.③	9.②	10.②	11.②	12.②	13.③	14.③	15.③
16.①	17.③	18.②	19.③	20.①	21.③	22.②	23.④	24.③	25.①	26.③	27.①	28.④	29.①	30.①
31.④	32.③	33.④	34.④	35.③										

1. Section 153

통신 프로토콜(Communication Protocol)은 서로 다른 기기들 간의 데이터 교환을 원활하게 수행할 수 있도록 표준화한 통신 규약이다.

2. Section 153

프로토콜의 주요 기능

- 단편화와 재결합
- 캡슐화
- 흐름 제어
- 오류 제어
- 동기화
- 순서 제어
- 주소 지정
- 다중화
- 경로 제어
- 전송 서비스(우선순위, 서비스 등급, 보안성)

3. Section 153

- 단편화(Fragmentation) : 송신 측에서 전송할 데이터를 전송하기에 알맞은 일정 크기의 작은 블록으로 자르는 작업
- 다중화(Multiplexing) : 한 개의 통신 회선을 여러 가입자들이 동시에 사용하도록 하는 기능
- 동기화(Synchronization) : 송 · 수신 측이 같은 상태를 유지하도록 타이밍(Timing)을 맞추는 기능

4. Section 153

프로토콜의 기본 요소는 구문(Syntax), 의미(Semantics), 시간(Timing)이다.

5. Section 153

- 수신 측에서 단편화된 블록을 원래의 데이터로 모으는 것을 재결합(Reassembly)이라고 한다.
- 한 개의 통신 회선을 여러 가입자들이 동시에 사용하도록 하는 기능을 다중화(Multiplexing)라고 한다.

6. Section 154

OSI 참조 모델은 적절한 수의 계층으로 나누어 시스템의 복잡도를 최소화하였다.

7. Section 154

- 사용자가 OSI 환경에 접근할 수 있도록 서비스를 제공하는 것은 응용 계층이다.
- 전송 계층은 논리적 안정과 균일한 데이터 전송 서비스를 제공함으로써 종단 시스템(End-to-End) 간에 투명한 데이터 전송을 가능하게 한다.

8. Section 154

OSI 7계층 참조 모델

- **하위 계층** : 물리 계층, 데이터 링크 계층, 네트워크 계층
- **상위 계층** : 전송(트랜스포트) 계층, 세션 계층, 표현 계층, 응용 계층

9. Section 155

- **물리 계층** : 전송에 필요한 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성을 정의함
- **네트워크 계층** : 개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터 교환 및 중계 기능, 경로 설정 기능을 함
- **트랜스포트(전송) 계층** : 논리적 안정과 균일한 데이터 전송 서비스를 제공함으로써 종단 시스템(End-to-End) 간에 투명한 데이터 전송을 가능하게 함

10. Section 153

슬라이딩 윈도우(Sliding Window)는 수신 측이 수신할 수 있는 최대 패킷의 수를 정한 후 수신 측으로부터 확인 신호를 받지 않더라도 연속적으로 패킷을 전송할 수 있으므로 한 번에 여러 개의 프

레임을 전송할 경우 효율적이다.

11. Section 155

트랜스포트(전송) 계층은 네트워크의 유형(Type)을 A형, B형, C형 3개로 나누고, 서비스의 등급(Class)을 0~4까지 5개로 나누어, 네트워크 형에 따라 다양한 품질의 서비스(QoS)를 제공한다.

12. Section 155

데이터 링크 계층의 주요 기능은 프레임 동기, 오류 제어, 흐름 제어이다.

13. Section 155

X.25와 IP는 네트워크 계층의 대표적인 프로토콜이다.

14. Section 155

표현 계층

- 응용 계층으로부터 받은 데이터를 세션 계층에 보내지기 전에 통신에 적당한 형태로 변환하고, 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하여, 서로 다른 기종의 컴퓨터 사이의 데이터 전송을 가능하게 한다.
- 코드 변환, 구문 검색, 암호화, 압축, 정보 형식(포맷) 변환, 접속 설정, 문맥 관리, 정보 전송 기능을 한다.

15. Section 155

응용 계층은 사용자와 바로 접해 있는 계층으로 사람이 이해할 수 있는 형태의 정보를 다루고, 표현 계층은 사용자의 정보를 컴퓨터가 인식할 수 있는 형태로 전환한다.

16. Section 155

전송 계층(Transport Layer)

- 논리적 안정과 균일한 데이터 전송 서비스를 제공함으로써 종단 시스템(End-to-End) 간에 투명한 데이터 전송을 가능하게 한다.
- OSI 7계층 중 하위 3계층과 상위 3계층의 인터페이스를 담당한다.
- 종단 시스템(End-to-End) 간의 전송 연결 설정과 해제, 데이터 전송 기능을 하며, 주소 지정, 다중화, 역다중화, 오류 제어와 흐름 제어를 수행한다.

17. Section 155

경로 설정 기능은 네트워크 계층의 서비스이다.

18. Section 156

X.25는 통신을 원하는 두 단말장치가 패킷 교환망을 통해 패킷을 원활히 전달하기 위해서, DCE와 DTE 간의 통신 절차를 규정하는 프로토콜이다.

19. Section 156

- **물리 계층** : 패킷 교환기와 단말장치 간의 물리적 접속에 관한 인터페이스를 정의
- **프레임 계층** : 패킷의 원활한 전송을 위해 데이터 링크 제어를 수행
- **패킷 계층** : OSI 7계층의 네트워크 계층에 해당하는 것으로, DTE와 DCE 간에 가상 회선을 제공하여 데이터의 안정된 전송을 지원

20. Section 154

다중화는 하나의 고속 통신 회선을 다수의 터미널이 공유할 수 있도록 하는 기능이다. 즉 정확한 전송보다는 고속의 통신 회선을 좀 더 효율적으로 이용하기 위한 기능이다.

21. Section 156

X.25 프로토콜은 프레임 계층과 패킷 계층의 두 계층에서 오류 제어 기능을 제공하므로, 신뢰성이 좋다.

22. Section 156

프레임 계층

- 패킷의 원활한 전송을 위해 데이터 링크 제어를 수행한다.
- OSI 7계층의 데이터 링크 계층에 해당한다.
- 전송 제어를 위해 HDLC 프로토콜의 변형인 LAPB를 사용한다.
- 다중화, 순서 제어, 오류 제어, 흐름 제어 기능 등을 한다.

23. Section 156

X.25는 한 회선에 장애가 발생한 경우 정상적인 통신 경로를 선택하여 우회 전송이 가능하다.

24. Section 158

- 전자공업협회(EIA, Electronic Industries Association)는 OSI 계층의 물리 계층에 관한 연구를 주로 하며, RS-232C 표준안을 제정하였다.
- 전기전자기술자협회(IEEE, Institute of Electric and Electronic Engineers)는 802.3(CSMA/CD), 802.5(토큰링) 등

근거리 통신망(LAN)에 관한 표준화(IEEE 802)로 널리 알려져 있다.

25. Section 155

암호화와 형식 변환의 기능을 제공하는 계층은 표현 계층이다.

26. Section 157

- 인터넷 계층의 프로토콜에는 IP, ICMP, IGMP, ARP, RARP 등이 있다.
- TCP, UDP는 전송 계층의 프로토콜이다.

27. Section 157

- RARP(Reverse Address Resolution Protocol) : 물리적 주소를 IP 주소로 변환하는 기능
- ICMP(Internet Control Message Protocol) : IP에서 발생하는 문제를 처리하기 위한 프로토콜로, 버퍼 공간의 부족이나 주소 인식 불가 등 패킷 수신 시 송신 측에 문제 발생을 알리는 역할을 담당하여 네트워크의 문제 발생을 막는다.
- IGMP(Internet Group Management Protocol) : 인터넷 그룹 관리 프로토콜. 인터넷 컴퓨터가 멀티캐스트 그룹 구성원이 인접해 있는지를 인근의 라우터들에게 알리는 기능

28. Section 153

통신 프로토콜에서 실체(Entity)에 해당하는 것은 사용자 응용 프로그램, 파일 전송 패키지, DB 관리 시스템이다.

29. Section 154

다중화, 오류 검출, 회복 등의 기능을 수행하는 계층은 전송 계층이다.

30. Section 154

서비스 프리미티브(Primitive)는 계층 간 표현되는 동작으로 요구, 지시, 응답, 확인 이렇게 4가지가 있다.

31. Section 154

OSI 참조 모델은 어떤 시스템이라도 상호 연결될 수 있도록 하는 표준을 제공하는 것으로 개별적인 독립된 시스템의 특수성 유지는 OSI 참조 모델과 무관하다.

32. Section 155

통신망, 즉 Network와 관련된 계층은 네트워크 계층이고, 다중화는 전송 계층의 기능이다.

33. Section 154

계층화 구조의 기본 구성 요소는 개체(Entity), 데이터 단위(Data Unit), 접속(Connection)이다.

34. Section 155

- 응용 간의 대화 제어, 동기 제어를 담당하는 계층은 세션 계층이다.
- 물리 계층(Physical Layer) : 전송에 필요한 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성을 정의함
- 네트워크 계층(Network Layer, 망 계층) : 개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터 교환 및 중계 기능, 경로 설정 기능을 함
- 데이터 링크 계층(Data Link Layer) : 두 개의 인접한 개방 시스템들 간에 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 하며, 프레임 동기, 오류 제어, 흐름 제어 기능을 함

35. Section 157

TCP는 신뢰성 있는 연결형 서비스를 제공한다.

5장 정답 및 해설 — 정보 통신망 기술

1.④	2.④	3.①	4.②	5.③	6.③	7.③	8.②	9.④	10.②	11.②	12.②	13.②	14.②	15.①
16.①	17.③	18.④	19.④	20.③	21.①	22.②	23.③	24.②	25.④	26.③	27.③	28.②	29.②	30.①
31.①	32.①	33.①	34.①	35.②	36.④	37.①	38.④	39.④	40.④	41.③	42.③	43.②	44.③	45.③
46.③	47.②	48.③	49.②	50.①										

1. Section 159

정보 통신망은 정보를 제공 또는 수신하는 단말장치와 직접 정보를 전달하는 전송장치, 그리고 정보를 식별하여 수신지로 전송하는 교환장치로 구성된다.

2. Section 159

성형(Star)

- 중앙에 주 컴퓨터가 있고, 이를 중심으로 단말장치들이 연결되는 중앙 집중식 네트워크의 구성 형태이다.
- 포인트 투 포인트(Point-to-Point) 방식으로 회선을 연결한다.
- 각 단말장치들은 주 컴퓨터를 통하여 데이터를 교환한다.
- 단말장치의 추가와 제거가 쉽다.
- 하나의 단말장치가 고장나더라도 다른 단말장치에는 영향을 주지 않지만, 주 컴퓨터가 고장나면 전체 통신망의 기능이 정지된다.
- 중앙 집중식이므로 교환 노드의 수가 가장 적다.

3. Section 163

"10 Base 5"에서 10은 전송 속도가 10Mbps, BASE는 베이스밴드 방식, 5는 한 세그먼트의 최장 거리가 500m임을 나타낸다. "10 Base 5" 네트워크에서 사용되는 표준 전송 매체는 굵은 동축 케이블(Coaxial Cable)이다.

4. Section 159

루프형은 컴퓨터와 단말장치들을 서로 이웃하는 것끼리 포인트 투 포인트(Point-to-Point) 방식으로 연결된 형태로, 단말장치의 추가와 제거가 어렵다.

5. Section 159

망형(Mesh)

- 모든 지점의 컴퓨터와 단말장치를 서로 연결한 형태로, 노드의 연결성이 높다.
- 많은 단말장치로부터 많은 양의 통신을 필요로 하는 경우에 유리하다.
- 보통 공중 데이터 통신 네트워크에서 주로 사용되며, 통신 회선의 총 경로가 가장 길다.

6. Section 159

정보의 전달 체계는 정보 통신망이고 정보 통신망 내에서 사용되는 실질적인 장비들이 단말장치, 교환장치, 컴퓨터 등이다.

7. Section 159

정보 통신망의 3대 동작 기능은 전달 기능, 신호 기능, 제어 기능이다.

8. Section 161

목적지의 주소, 제어 정보 등의 추가로 인해 오버헤드가 존재하는 것은 패킷 교환 방식의 단점이다.

9. Section 161

패킷 교환망의 주요 기능 : 패킷 다중화, 경로 제어, 논리 채널, 순서 제어, 트래픽 제어, 오류 제어

10. Section 161

가상 회선 방식과 회선 교환 방식은 전송 전에 먼저 송신지와 수신지 사이의 연결을 확립한다.

11. Section 161

패킷 교환 방식은 데이터를 패킷 교환기에 저장하였다가 전송 경로를 결정(선택)한 후 전송하는 것으로, '저장 → 선택 → 전달' 이라고 할 수 있다.

12. Section 162

- WAN : 일반적으로 제3자에 의해 제공되고 운영되는 공중망(Public Network)으로, LAN을 여러 개 모아 그들 간에 고속 전송이 가능한 전용 회선으로 연결함
- VAN : 공중 통신 사업자로부터 통신 회선을 임대하여 하나의 사설망을 구축하고 이를 통해 정보의 축적, 가공, 변환 처리 등 부가 가치를 첨가한 후 불특정 다수를 대상으로 서비스를 제공하는 통신망
- ISDN : 음성 및 데이터, 화상 등의 다양한 통신 서비스를 하나의 디지털 통신망을 근간으로 해서 종합적으로 제공할 수 있도록 통합한 것

13. Section 162

LAN은 하나의 건물 또는 공장, 학교, 등의 일정 지역으로 통신 거리가 제한된다.

14. Section 162

LAN은 특정 정보의 고속 처리보다는 고속 전송에 중점을 둔 정보 전송을 위한 네트워크로, ②의 복잡한 과학 기술 계산의 고속 처리는 컴퓨터를 통해 얻을 수 있는 효과이다.

15. Section 162

IEEE에 의한 LAN은 OSI 7계층의 하위 2계층인 물리 계층과 데이터 링크 계층에 해당한다.

16. Section 162

블루투스(Bluetooth)는 근거리에서 무선으로 데이터 통신을 가능하게 해주는 표준 기술이다. 블루투스를 이용하면 핸드폰, PDA, 노트북과 같은 휴대 가능한 장치들 간의 양방향 근거리 통신을 복잡한 전선없이 저렴한 가격으로 구현할 수 있다.

17. Section 160

- **단말장치** : 데이터 통신 시스템과 외부 사용자의 접속점에 위치하여 최종적으로 데이터를 입 · 출력하는 장치
- **변 · 복조장치** : 단말장치로부터 전송되는 디지털 데이터를 아날로그 회선에 적합한 아날로그 신호로 변환하는 변조(MOdulation) 과정과 그 반대의 복조(DEModulation) 과정을 수행
- **다중화장치** : 하나의 통신 회선에 여러 개의 단말 장치가 동시에 접속하여 사용할 수 있도록 하는 장치

18. Section 163

고속 이더넷의 전송 속도는 100Mbps이다.

19. Section 163

CSMA/CD는 모든 단말장치가 공평하게 매체에 접근하게 하는 경쟁 방식의 매체 접근 기법이다. ④는 액세스 권리가 공평하게 부여되지 않는다는 의미이므로 잘못된 설명이다.

20. Section 165

부가 가치 통신망(VAN)의 기능에는 전송 기능, 교환 기능, 통신 처리 기능, 정보 처리 기능이 있다.

21. Section 165

10 Base T는 전송속도, 전송방식, 전송매체를 표현한 것으로 전송속도가 10Mbps, 전송방식이 Baseband, 전송매체가 Twisted Pair Cable임을 의미한다.

22. Section 161

①, ③, ④는 공중 데이터 교환망(PSDN)에 대한 설명이다.

23. Section 165

VAN은 패킷 교환망을 이용한 교환 서비스를 한다.

24. Section 166

- ISDN은 OSI의 통신 계층 구조를 따르며, 하위 계층 기능만 제공하는 베어러 서비스와 상위/하위 계층 기능을 모두 제공하는 텔레 서비스로 나누어진다.
- ②는 VAN의 기능이다.

25. Section 166

ISDN의 기본 속도 인터페이스(BRI)는 2B + D + 오버헤드 = 2 × 64(B) + 16(D) + 48(오버헤드) = 192Kbps이다.

26. Section 166

ISDN의 기본 속도 구조는 2B + D, 즉 두 개의 B(정보 전송용) 채널과 한 개의 D(제어 신호용) 채널로 구성된다.

27. Section 166

TA(Terminal Adapter)는 비 ISDN 단말장치를 ISDN에 접속하는 역할(프로토콜 및 속도 변환)을 한다.

참조점(기준점, 접속점, 분계점)

- U(User) : 내부망과 외부망을 구분
- T(Terminal) : 사용자 영역과 네트워크 영역을 구분
- S(System) : 사용자 장비와 네트워크 장비를 구분
- R(Rate) : 비 ISDN 단말장치와 ISDN 장비를 구분

28. Section 166

ISDN의 기본 속도 구조(BRI, Basic Rate Interface)

- 가정용 ISDN 구조
- 2B + D, 즉 두 개의 B(정보 전송용) 채널과 한 개의 D(제어 신호용) 채널로 구성
- D 채널은 16Kbps를 사용
- 속도 : 2B+D+오버헤드 = 2×64+16+48 = 192Kbps

29. Section 164

브리지(Bridge)는 두 개의 LAN을 연결하는 기능을 하며, 통신량을 조절하여 네트워크 상의 많은 단말장치들에 의해 발생되는 트래픽 병목 현상을 줄일 수 있다.

30. Section 162

FDDI(Fiber Distributed Data Interface)는 LAN과 LAN 사이 혹은 컴퓨터와 컴퓨터 사이를 광섬유 케이블로 연결하는 고속 통신망 구조로 이중 링 구성을 통해 통신망이 한꺼번에 단절되는 것을

방지한다.

31. Section 167

문자로 된 도메인 네임을 컴퓨터가 이해할 수 있는 IP 주소로 변환하는 역할을 하는 시스템을 DNS(Domain Name System)라고 한다.

32. Section 167

- FTP는 컴퓨터와 컴퓨터 또는 컴퓨터와 인터넷 사이에서 파일을 주고받을 수 있도록 하는 원격 파일 전송 프로토콜이다.
- 일정한 주제를 놓고 여러 사람이 토론을 벌이는 인터넷 서비스는 유즈넷(Usenet)이다.

33. Section 167

- Repeater : 전송되는 신호가 전송 선로의 특성 및 외부 충격 등의 요인으로 인해 원래의 형태와 다르게 왜곡되거나 약해질 경우 원래의 신호 형태로 재생하여 다시 전송하는 역할을 수행
- Gateway : 프로토콜 구조가 전혀 다른 네트워크의 연결을 수행
- Bridge : LAN과 LAN을 연결하거나 LAN 안에서의 컴퓨터 그룹(세그먼트)을 연결하는 기능을 수행

34. Section 169

이동 통신의 통신 내용에는 음성뿐만 아니라 신호, 문자, 영상 정보 등을 모두 포함한다.

35. Section 169

시분할 다중 접속(TDMA) 방식은 주파수 다중 접속(FDMA) 방식에 비해 동일한 주파수 대역에서 3배 이상 가입자를 증가시킬 수 있다.

36. Section 169

셀룰러 시스템은 주파수 대역을 800MHz를 이용하고, PCS 시스템은 1.8GHz를 이용한다.

37. Section 160

회선 교환 방식은 전송된 데이터의 오류 제어나 흐름 제어를 사용자가 직접 제어해야 한다.

38. Section 161

X.75는 패킷망 상호 간의 접속을 위한 프로토콜이다.

39. Section 163

- LAN에서 사용하는 매체 액세스 제어 방식 중 CSMA/CD 방식은 전송 도중 충돌이 감지되면 즉시 전송을 멈추고 다른 스테이션에 충돌을 알리는 재밍 신호를 전송한 후 일정 시간이 지난 다음 데이터를 재송신한다.
- 재밍(Jamming) 신호란 어떤 주파수의 신호를 방해할 목적으로 보내는 주파수를 말한다.

40. Section 164

라우터(Router)는 브리지와 같이 LAN과 LAN의 연결 기능에 데이터 전송의 최적 경로를 선택할 수 있는 기능이 추가된 것으로, 서로 다른 LAN이나 LAN과 WAN을 연결하는 기능도 수행한다.

41. Section 169

흐름 제어(Flow Control)는 패킷 교환망에 흐르는 패킷 수를 적절히 조절하여 전체 시스템의 안전성을 기하고 서비스의 품질 저하를 방지하는 기능이다.

42. Section 169

CDMA에서 셀 분할은 정적 분할뿐만 아니라 동적 분할 방법도 사용한다.

43. Section 169

패킷(Packet)은 전송 혹은 다중화를 목적으로 메시지를 일정한 비트 수로 분할하여 송 · 수신 측 주소와 제어 정보 등을 부가하여 만든 데이터 블록이다.

44. Section 166

시간적으로 양방향 신호를 제어하는 방식은 TCM(시간 압축 다중화 방식, Time Compression Multiplexing)이다.

- TCM(시간 압축 다중화, Time Compression Multiplexing) 방식 : 2선식 선로에 전이중 전송을 위해서 전송시간을 분할하여 송신과 수신을 교대로 실현하는 방식
- ECH(반향 제거, Echo Cancellation Hybrid) 방식 : 한 개의 전송 선로의 동일 대역폭 내, 동일 시간에 양방향 전송을 가능하게 하는 기술로, 송신 신호가 수신 측으로 누설되어 수신 신호에 혼합되면 반향 제거기는 송신 신호 성분이 누설되는 경로의 전달함수를 예측하여 반향 성분을 제거함

45. Section 169

가상 회선 방식은 송신지로부터 수신지까지 패킷들을 순서적으로

운반하는 방식이고, 데이터그램 방식은 각각의 패킷들을 순서에 상관없이 운반하는 방식이다. 외부 서비스에서 데이터그램 방식으로 순서 없이 운반된 패킷을 내부 서비스에서 순서적으로 운반한다는 것은 무의미하다.

46. Section 167

인터넷의 기본 프로토콜은 TCP/IP이다.

47. Section 170

ATM 헤더의 구조

- GFC(Generic Flow Control) : 가입자망 내에서 ATM 망 쪽으로의 흐름 제어를 수행하거나 각 단말이 공평하게 전송 매체를 사용하도록 함
- VPI(Virtual Path Identifier) : 셀이 ATM 망에서 실제로 전달되는 가상 경로(Path)를 배정하는 주소
- VCI(Virtual Channel Identifier) : Virtual Path 내에서 가상 채널을 구분하기 위하여 사용되는 주소
- PT(Payload Type) : 첫 비트는 사용자 데이터를 구분하는데 사용되고, 프레임 경계 식별 등에 이용됨
- CLP(Cell Loss Priority) : 망에서 과도한 트래픽 발생 시 우선적으로 버려도 되는 셀을 표시함
- HEC(Header Error Control) : 셀 헤더 내의 에러 검출 및 한 Bit의 에러 복구 기능을 위하여 사용됨

48. Section 170

ATM 프로토콜의 구조

- 평면(Plane)
  - 관리 평면(Management Plane) : 평면 관리 및 계층 관리, 망 감시 기능을 수행
  - 제어 평면(Control Plane) : 호 제어와 연결 제어 기능을 수행
  - 사용자 평면(User Plane) : 사용자 정보 전송을 지원함
- 계층(레이어, Layer)
  - 상위 계층(Higher Layer) : 애플리케이션을 제공하는 계층으로 제어 상위 계층(Control Higher Layer), 사용자 정보 상위 계층(User Information Higher Layer)으로 구성
  - ATM 적응 계층(ATM Adaptation Layer) : 페이로드(Payload)를 만들기 위해 48바이트 이내로 데이터를 잘라내는 역할을 함
  - ATM 계층(ATM Layer) : 프레임(Frame)을 생성하고 헤더를 정의함
  - 물리 계층 (Physical Layer) : 전송 매체와 신호 부호화 방법에 대한 명세를 포함함

49. Section 170

광대역 종합정보통신망은 회선 교환이나 패킷 교환 방식을 사용하는 것이 아니라 회선 교환과 패킷 교환 방식의 장점을 결합한 교환 및 전송 기술인 ATM 방식을 사용한다.

50. Section 170

데이터 교환 방식에는 회선 교환 방식과 축적 교환 방식이 있으며, 축적 교환 방식에는 메시지 교환 방식과 패킷 교환 방식이 있다.

6장 정답 및 해설 — 뉴미디어/멀티미디어

1.③	2.②	3.②	4.④	5.①	6.②	7.③	8.③	9.②	10.④	11.①	12.③	13.②	14.③	15.①
16.②	17.④	18.③	19.①	20.③	21.①	22.②	23.④	24.②	25.②	26.②				

1. Section 171

네트워크화는 지역적 공간에 구애받지 않는 광대역성을 제공한다.

2. Section 171

뉴 미디어의 분류

- 유선계 : CATV, 비디오텍스, VRS, 원격 회의(Teleconference), ARS, 텔레비전 전화, 팩시밀리 통신, 퍼스널 컴퓨터 통신, LAN, VAN, ISDN 등
- 무선계 : 위성 통신, 텔레텍스트, HDTV, PCM 음성 방송, 팩시밀리 방송, 개인 휴대 통신 등

- **패키지계** : 비디오 디스크, 디지털 오디오 디스크, VTR, 광 디스크 등

3. Section 171

텔레텍스트는 TV 화면과 화면 사이의 수직 귀선 시간(전파 간격)을 이용하여 정보를 한쪽 방향으로만 전송하는 형태로, 문자 다중 방송이라고도 한다. 방송의 형태이므로 무선 형태(패킷 라디오 방식)로 제공된다.

4. Section 171

비디오텍스는 유선계에 속한다.

5. Section 171

뉴 미디어를 방송계와 통신계로 분류

- **방송계** : CATV, PCM 음성 방송, 텔레텍스트, HDTV 등
- **통신계** : 원격 회의(Teleconference), 비디오텍스, 텔레텍스, VRS, 개인 휴대 통신, PC 통신 , LAN, VAN, ISDN 등

6. Section 171

- HDTV : 기존의 TV 주사선을 늘리고 주파수 대역폭을 확대하여 선명한 화상과 양질의 음성을 제공하는 TV
- CCTV : 화상 정보를 특정의 목적으로 특정의 대상자에게 전달하는 시스템
- NTSC : 한국, 미국, 일본 등지에서 사용하는 컬러 TV 표준 방식

7. Section 171

CATV(Cable Television)

- 원래 난시청 해소를 목적으로 설치했던 공동 시청 안테나를 이용하여 수신한 TV 신호를 일정한 전송로를 통하여 사용자에게 제공한다.
- 양방향 통신이 가능하다.
- 사용자의 범위가 한정적이다.
- 다채널로서 방송뿐만 아니라 종합 정보 서비스가 가능하다.
- 전송로는 동축 케이블이나 광섬유 케이블을 사용한다.
- 기존 TV와 방송 방식이 동일하여 기존 TV를 단말기로 사용한다.

8. Section 171

- **화상 응답 시스템(VRS, Video Response System)** : 기본적인 개념은 비디오텍스와 같지만 정지 화상, 동화상, 음성 등을 송 · 수신할 수 있는 시스템
- **텔레텍스트(Teletext)** : TV 전파의 빈틈을 이용하여 TV 방송과 함께 문자나 도형 정보를 제공하는 문자 다중 방송
- **자동 응답 시스템(ARS, Automatic Response System)** : 각종 정보를 음성으로 저장하여 두고 사용자가 전화를 이용하여 원하는 정보를 검색할 수 있도록 하는 시스템

9. Section 171

- **텔렉스(Telex)** : 문자, 숫자 및 기호 등의 정보를 텔렉스 교환기를 사용해서 전송하는 시스템
- **팩시밀리(FAX, Facsimile)** : 종이 위의 문자나 도형 정보를 화상 정보로 분해하여 전기적 신호로 전송하고, 수신 측에서는 원래의 정보로 재생하는 방식
- **텔레텍스트(Teletext)** : TV 전파의 빈틈을 이용하여 TV 방송과 함께 문자나 도형 정보를 제공하는 문자 다중 방송

10. Section 171

문자 및 도형 정보의 표현 방식

- **모자이크(Mosaic) 방식** : 그림을 정해진 크기의 블록으로 분할하여 미리 정해져 있는 모양의 블록(모자이크 블록)에 그림을 맞추듯이 표현하는 방식
- **지오메트릭(Geometric) 방식** : 그림의 윤곽을 직선, 원호, 점, 사각형, 다각형의 5종류로 분해한 후 조합하여 표현하는 방식
- **포토그래픽(Photographic) 방식** : 도형을 도트(화소) 단위로 분해하여 표현하는 방식
- DRCS(Dynamically Redefinable Character Set) : 동적으로 재정의가 가능한 문자 집합의 조합으로 표현하는 방식

11. Section 171

CATV(Cable Television)는 양방향 통신이 가능하다.

12. Section 172

멀티미디어는 데이터가 일정한 방향으로 순서적으로 처리되는 것이 아니라 사용자의 선택에 따라 다양한 방향으로 처리된다.

13. Section 172

MIDI는 오디오 파일 형식 중 하나이다.

14. Section 172

VHF 튜너는 일반 텔레비전을 수신하기 위한 장치의 한 부분으로,

멀티미디어 시스템에 두루 쓰이는 장비라고 보기는 어렵다.

15. Section 172

JPEG(Joint Photographer's Experts Group)은 정지 영상 압축의 국제 표준 방식으로, 주로 손실 압축 방식을 사용하여 압축하며, 인터넷에서 그림을 전송할 때 많이 사용된다.

16. Section 172

MP3(MPEG Audio Player-3)는 고음질 오디오 압축의 표준 형식으로, MPEG에서 규정한 MPEG-1의 압축 기술을 이용하여 음반 CD 수준의 음질을 유지하면서 용량을 1/12까지 압축할 수 있다.

17. Section 172

MPEG는 동영상 전문가 그룹에서 제정한 동영상 압축을 위한 국제표준규격으로, 손실 압축 기법을 사용한다.

18. Section 172

MPEG은 동영상 압축 기술이다.

19. Section 171

- 비디오텍스 : 각종 정보를 모아 데이터베이스를 구축하고 전화망을 통해 TV나 단말장치에 접속하여 필요한 정보를 문자나 그림의 형태로 검색할 수 있도록 하는 서비스
- CATV : 원래 난시청 해소를 목적으로 설치했던 공동 시청 안테나를 이용하여 수신한 TV 신호를 일정한 전송로를 통하여 사용자에게 제공
- HDTV : 기존의 TV 주사선을 늘리고 주파수 대역폭을 확대하여 선명한 화상과 양질의 음성을 제공하는 TV

20. Section 171

뉴미디어는 디지털 기술의 발전과 관계가 있다.

21. Section 171

- 텔레텍스트(Teletext) : TV 전파의 빈틈을 이용하여 TV 방송과 함께 문자나 도형 정보를 제공하는 문자 다중 방송
- 팩시밀리(Fax) : 종이 위의 문자나 도형 정보를 화상 정보로 분해하여 전기적 신호로 전송하고, 수신 측에서는 원래의 정보로 재생하는 방식
- 텔렉스(Telex) : 문자, 숫자 및 기호 등의 정보를 텔렉스 교환기를 사용해서 전송하는 시스템

22. Section 171

Teletext(텔레텍스트)는 TV 전파의 빈틈을 이용하여 TV 방송과 함께 문자나 도형 정보를 제공한다.

23. Section 171

HDTV는 기존의 TV 주사선을 늘리고 주파수 대역폭을 확대하여 선명한 화상과 양질의 음성을 제공하는 TV로 뉴미디어의 한 형태이다.

아날로그 컬러 TV 방식

- NTSC : 미국에서 개발된 방식으로 수평 주사선은 525라인이지만, 이 중 487라인만 실제 화면 구성에 사용되며 나머지는 기타 정보를 표현하기 위해 사용됨
- PAL : 독일에서 개발된 방식으로 NTSC 방식과 유사하나 주사선이 625라인이며 신호 전송계에 따른 색 변형이 적고 방송 설비에 고도의 규격이 필요 없다는 이점이 있음
- SECAM : 프랑스에서 개발된 방식으로 주사선이 625라인이며 두 가지 색차 신호를 NTSC 및 PAL 방식의 경우와 같이 동시에 보내는 것이 아니라 순차적으로 교체하면서 보내는 방식

24. Section 171

CATV 시스템은 헤드엔드와 중계 전송망(전송로), 가입자 설비(단말장치)로 구성된다.

25. Section 171

- 텔레마틱(Telematics)이란 정보 통신 기술의 발달로 새롭게 개발된 다양한 매체들을 의미하는 뉴미디어의 여러 가지 형태를 의미한다.
- 텔레텍스트는 TV 전파의 빈틈을 이용하여 TV 방송과 함께 문자나 도형 정보를 제공하는 문자 다중 방송으로, 전송되는 정보를 일방적으로 수신만 할 수 있는 단방향 형태로 제공된다.

26. Section 171

Radio, Teletext는 방송국에서 송신하는 정보를 수신만 할 수 있는 단방향 형태이고, CCTV는 특정 목적으로 촬영한 화상 정보를 특정의 수신자에게 전달만 하는 시스템이다.