수험생의 마음으로 만든 책! 시나공 시리즈

2014 시나공

정답 및 해설

정보처리기사 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 프로그램이 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 기본키(Primary Key)와 이를 참조하는 외래키(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 기본키를 개체 B의 외래키로 추가하거나, 개체 B의 기본키를 개체 A의 외래키로 추가하여 표현한다.
- Y가 1:N 관계이면 개체 A의 기본키를 개체 B의 외래키로 추가하여 표현하거나 별도의 테이블로 표현한다.
- Y가 N:M 관계이면 개체 A와 B의 기본키를 모두 포함한 별도의 테이블로 표현한다.
- 기본키들은 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 기본키다.

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 외래키에 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

- $\sigma_{학과='국문과'}(학생)$: 학생 릴레이션에서 국문과 학생들을 Select한다.
- $\pi_{학번,이름}$: 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), 기본키 : {A, C}, 외래키 : C, 참조 : R2
- R2(C, B) 기본키 : 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 후보키인 관계형이다.
- 3NF에서 후보키가 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

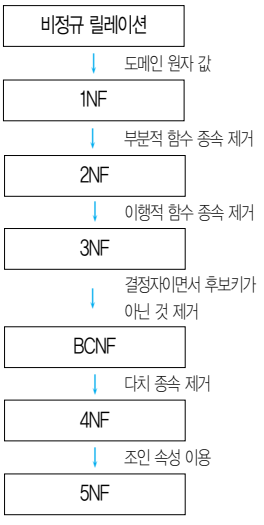
정규화

- 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 기본키의 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 (기본키_속성명),
        UNIQUE (대체키 속성명, … ),
        FOREIGN KEY (외래키_속성명, … )
            REFERENCES 참조테이블(기본키_속성명)
        CHECK (조건식) );
```

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

※ 기본키나 대체키에 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 정보를 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 기본키를 포함하는 릴레이션이 참조되는 릴레이션이 된다.

83. Section 014

릴레이션의 기본키와 대응되어 릴레이션 간의 참조 무결성 제약조건을 표현하는데 사용되는 중요한 도구는 외래키이다.

84. Section 014

기본키는 한 릴레이션에서 특정 튜플을 유일하게 구별할 수 있는 속성으로, 기본키로 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 — 데이터베이스 고급 기능

1.④	2.④	3.③	4.②	5.①	6.①	7.③	8.④	9.④	10.④	11.③	12.④	13.①	14.②	15.①
16.②	17.②	18.④	19.①	20.④	21.①	22.③	23.①	24.②	25.③	26.①	27.④	28.①	29.②	30.④
31.④	32.②	33.①	34.②	35.③	36.④	37.④	38.④	39.③	40.①	41.③	42.①	43.②	44.③	45.③
46.②														

1. Section 024

트랜잭션의 특징 중 원자성은 트랜잭션의 연산이 데이터베이스에 모두 반영되든지 아니면 전혀 반영되지 않아야 한다는 성질이다. 즉 일부만 반영되어서는 안 된다는 것이다.

2. Section 024

Isolation(독립성, 격리성, 순차성)

- 둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행중에 다른 트랜잭션의 연산이 끼어들 수 없다.
- 수행중인 트랜잭션은 완전히 완료될 때까지 다른 트랜잭션에서 수행 결과를 참조할 수 없다.

4. Section 024

해석 : 다음 중 데이터의 무결성을 책임지는 트랜잭션의 특징이 아닌 것은?

5. Section 024

트랜잭션

- 트랜잭션(Transaction)은 데이터베이스에서 하나의 논리적 기능을 수행하기 위한 작업의 단위이다.
- 트랜잭션은 데이터베이스 시스템에서 복구 및 병행 시행 시 처리되는 작업의 논리적 단위이다.
- 하나의 트랜잭션은 Commit되거나 Rollback된다.
- 트랜잭션은 일반적으로 회복의 단위가 된다.

6. Section 024

ROLLBACK 연산

- ROLLBACK 연산은 하나의 트랜잭션 처리가 비정상적으로 종료되어 데이터베이스의 일관성을 깨뜨렸을 때, 이 트랜잭션의 일부가 정상적으로 처리되었더라도 트랜잭션의 원자성을 구현하기 위해 이 트랜잭션이 행한 모든 연산을 취소(UNDO)시키는 연산이다.

7. Section 024

ROLLBACK 시의 조치

- 해당 트랜잭션을 재시작한다.
- 해당 트랜잭션을 폐기시킨다.

8. Section 024

트랜잭션의 상태

- Active(활동) : 트랜잭션이 실행중에 있는 상태
- Failed(실패) : 트랜잭션 실행에 오류가 발생하여 중단된 상태
- Aborted(철회) : 트랜잭션이 비정상적으로 종료되어 ROLLBACK 연산을 수행한 상태
- Partially Committed(부분 완료) : 트랜잭션의 마지막 연산까지 실행했지만, COMMIT 연산이 실행되기 직전의 상태
- Committed(완료) : 트랜잭션이 성공적으로 종료되어 COMMIT 연산을 실행한 후의 상태

9. Section 026

로킹은 병행제어의 문제점을 해결하기 위한 방법이다.

10. Section 025

장애의 유형

- 트랜잭션 장애 : 입력 데이터 오류, 불명확한 데이터, 시스템 자원 요구의 과다 등 트랜잭션 내부의 비정상적인 상황으로 인하여 프로그램 실행이 중지되는 현상
- 시스템 장애 : 데이터베이스에 손상을 입히지는 않으나 하드웨어 오동작, 소프트웨어의 손상, 교착상태 등에 의해 모든 트랜잭션의 연속적인 수행에 장애를 주는 현상
- 미디어 장애 : 저장장치인 디스크 블록의 손상이나 디스크 헤드의 충돌 등에 의해 데이터베이스의 일부 또는 전부가 물리적으로 손상된 상태

11. Section 025

- 무결성 검사기(Integrity Checker) : 데이터베이스에 대해 의미상의 정확성(일관성)을 통제하는 무결성 제약 조건들이 위배되는지를 검사하는 서브시스템
- 질의 처리기(Query Processor) : 질의어를 DBMS가 이해하는 최적화된 저수준 명령어로 변환하는 서브시스템
- 트랜잭션 관리기 : 다중 사용자 환경에서 평행적이고 동시에 일어나는 트랜잭션 문제를 없애서 각각의 사용자가 데이터베이스의 자원을 배타적으로 이용할 수 있도록 관리해 주는 서브시스템

12. Section 025

DBMS의 구성 요소인 회복 관리기에 의해서 미디어의 물리적 손상을 회복시키는 것은 불가능하다. 이런 경우에 대비해 별도의 미디어에 수시로 Backup해 두는 것이 중요하다. 물리적 손상을 입으면 그 자체로는 회복시킬 수 없기 때문에 Backup해 둔 것을 재저장(Restore)시킴으로써 회복할 수 있다. 그러나 Backup한 내용

까지만 회복이 가능하다.

13. Section 025

회복 조치 방법

- 재시도(REDO) : 덤프와 로그를 이용하여 데이터베이스를 원래 상태로 회복시킨 후 트랜잭션을 재실행시킴
- 취소(UNDO) : Log에 보관한 정보를 이용하여 가장 최근에 변경된 내용부터 거슬러 올라가면서 원상 복구시켜 트랜잭션 작업을 취소시킴

14. Section 025

연기 갱신 기법

- 트랜잭션이 성공적으로 수행되고 완료 시점에 도달할 때까지 데이터베이스에 대한 실질적인 갱신을 연기하는 것이다.
- 트랜잭션 수행 동안 갱신된 내용은 트랜잭션 작업 공간에 있는 로그에만 기록한다.
- 트랜잭션이 완료 시점에 도달하고 로그가 디스크에 저장된 후 갱신된 내용이 데이터베이스에 기록된다.
- 트랜잭션이 완료 시점에 도달하기 전에 트랜잭션의 실패가 발생하면 트랜잭션이 데이터베이스에 어떤 영향도 미치지 않았기 때문에 어떠한 수행 취소 작업도 필요하지 않다.

15. Section 025

트랜잭션이 부분 완료되기 전에 장애가 발생하여 트랜잭션이 ROLLBACK되면 트랜잭션이 실제 데이터베이스에 영향을 미치지 않았기 때문에 어떠한 갱신 내용도 취소(UNDO)시킬 필요 없이 무시하면 된다. 따라서 REDO 작업만 한다.

16. Section 025

트랜잭션이 실행되기 시작할 때 그림자 페이지 테이블을 현재의 페이지 테이블에 복사하는 것이 아니고, 장애가 발생하여 ROLLBACK 연산을 할 때 그림자 페이지 테이블을 현재의 페이지 테이블에 복사한다.

그림자 페이징 기법

- 갱신 이전의 데이터베이스를 일정 크기의 페이지 단위로 구성하여 각 페이지마다 복사본인 그림자 페이지로 별도 보관해 놓고, 실제 페이지를 대상으로 트랜잭션에 의한 갱신 작업을 하다가 장애가 발생하여 트랜잭션 작업을 ROLLBACK시킬 때 갱신된 이후의 실제 페이지 부분에 그림자 페이지를 대체하여 회복시키는 기법
- 로그, UNDO 및 REDO 알고리즘이 필요 없다.

17. Section 026

병행제어 기법의 종류

- 연산 지연 또는 트랜잭션 취소를 이용하는 기법
  - 로킹(2단계 로킹)
  - 타임 스탬프 순서
- 다중 버전 타임 스탬프 기법
- 검증 방법(낙관적 병행제어 기법)

18. Section 026

병행수행과 직렬성

- 다중 프로그램 환경에서 여러 개의 트랜잭션을 병행수행한다는 것은 같은 시간에 여러 개의 명령을 동시에 실행할 수 있다는 것이 아니라, 시분할이나 입 · 출력 인터럽트 기법 등을 이용하여 일정한 시간 내에 각 트랜잭션에 있는 명령들이 시간적으로 번갈아 실행된다는 것이다.
- 병행수행된 각각의 트랜잭션 결과는 각 트랜잭션을 독자적으로 수행시켰을 때의 결과와 같아야 하는데, 이를 직렬성(Serializability) 또는 직렬화 가능성이라 한다.

19. Section 026

제어 없는 병행수행의 문제점

문제점	의 미
갱신 분실 (Lost Update)	두 개 이상의 트랜잭션이 같은 자료를 공유하여 갱신할 때 갱신 결과의 일부가 없어지는 현상
비완료 의존성 (Uncommitted Dependency)	• 임시 갱신이라고도 함 • 하나의 트랜잭션 수행이 실패한 후 회복되기 전에 다른 트랜잭션이 실패한 갱신 결과를 참조하는 현상
모순성 (Inconsis-tency)	• 불일치 분석(Inconsistent Analysis)이라고도 함 • 두 개의 트랜잭션이 병행수행 될 때 원치 않는 자료를 이용함으로써 발생하는 문제
연쇄 복귀 (Cascading Rollback)	병행수행되던 트랜잭션들 중 어느 하나에 문제가 생겨 ROLLBACK하는 경우 다른 트랜잭션도 함께 ROLLBACK되는 현상

21. Section 026

로킹(Locking)과 로킹 단위(Locking Granularity)

- 로킹(Locking)
  - 로킹은 주요 데이터의 액세스를 상호 배타적으로 하는 것이다.
  - 트랜잭션들이 어떤 로킹 단위를 액세스하기 전에 Lock(잠금)를 요청해서 Lock가 허락되어야만 그 로킹 단위를 액세스할 수 있도록 하는 기법이다.
- 로킹 단위(Locking Granularity)

– 병행제어에서 한꺼번에 로킹할 수 있는 단위이다.
– 데이터베이스, 파일, 레코드, 필드 등은 로킹 단위가 될 수 있다.
– 로킹 단위가 크면 로크 수가 작아 관리하기 쉽지만 병행성 수준이 낮아지고, 로킹 단위가 작으면 로크 수가 많아 관리하기 복잡하지만 병행성 수준이 높아진다.

22. Section 026

2단계 로킹(Two Phase Locking) 규약은 직렬성을 보장하는 대표적인 로킹 규약으로서 직렬성을 보장하는 장점은 있지만 교착상태를 예방할 수 없다는 단점이 있다.

23. Section 026

병행제어 기법의 종류

- **로킹(Locking) 기법** : 한 트랜잭션이 데이터 항목을 액세스하는 동안 다른 트랜잭션이 데이터 항목에 대한 로크를 소유한 경우에만 그 데이터 항목을 액세스할 수 있도록 하는 것
- **타임 스탬프(Time Stamp) 기법** : 트랜잭션의 실행 순서를 시스템이 부여한 타임 스탬프 순서에 따르게 하는 방법이다. 타임 스탬프 부여 방법으로 시스템 시계나 논리 계수기를 사용함
- **검증 기법** : 판독 단계, 검증 단계, 기록 단계에 따라 직렬성을 보장받음
- **다중 버전 기법** : Write 연산은 새로운 버전을 생성하고, Read 연산을 처리할 때 시스템은 새로운 버전을 선택함

24. Section 026

검증 기법

- 최적 병행수행 기법으로, 확인 기법 또는 낙관적 기법이라고도 한다.
- 병행수행하고자 하는 대부분의 트랜잭션이 판독 전용(Read Only) 트랜잭션일 경우 트랜잭션 간의 충돌률이 매우 낮아서 병행제어 기법을 사용하지 않고 실행되어도 이 중의 많은 트랜잭션은 시스템의 상태를 일관성 있게 유지한다는 점을 이용한다.
- 트랜잭션을 판독, 검증, 기록 단계로 수행하면서 트랜잭션 간의 충돌 정도를 확인하기 위해 검증(평가) 단계에 중점을 둔다.

27. Section 027

판독, 검증, 기록 단계로 수행하는 것은 병행제어 기법 중 하나인 검증 기법으로, 무결성과는 관계없다.

28. Section 027

무결성의 종류 : 널 무결성, 고유 무결성, 참조 무결성, 도메인 무결성, 키 무결성, 관계 무결성, 개체 무결성

30. Section 028

보안 기법의 종류

- **대조 확인(Authentication)** : 패스워드, 키(key), 목소리나 지문으로 대조 확인
- **권한 부여 규정** : 사용자 프로필(User Profile)의 유지
- **암호화** : 암호키와 해독키 사용

※ 시스템 체크 포인트는 장애 발생 시 회복을 위해 갱신 내용이나 시스템 상황 등에 관한 정보를 보관하는 지점이다.

31. Section 028

- **뷰 기법** : 뷰(View)에 권한을 명시하는 기법
- **GRANT/REVOKE 기법** : DB2의 SQL에서의 권한 부여 기법으로, DBA에 의해 GRANT/REVOKE 명령으로 권한을 부여하고 취소시키는 방법
- **DEFINE PERMIT 기법** : INGLES의 QUEL에서의 권한 부여 기법

32. Section 028

사용자 등급의 종류

- DBA : 데이터베이스 관리 책임자
- RESOURCE : 데이터베이스 및 테이블 생성 가능자
- CONNECT : 단순 사용자

33. Section 028

사용자 등급의 종류

- DBA : 데이터베이스 관리 책임자
- RESOURCE : 데이터베이스 및 테이블 생성 가능자
- CONNECT : 단순 사용자

34. Section 029

분산 데이터베이스의 목표

- **위치 투명성(Location Transparency)** : 액세스하려는 데이터베이스의 실제 위치를 알 필요 없이 단지 데이터베이스의 논리적인 명칭만으로 액세스할 수 있음
- **중복 투명성(Replication Transparency)** : 동일 데이터가 여러 곳에 중복되어 있더라도 사용자는 마치 하나의 데이터만 존재하는 것처럼 사용하고, 시스템은 자동으로 여러 자료에 대한 작업을 수행함

- 병행 투명성(Concurrency Transparency) : 분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실현되더라도 그 트랜잭션의 결과는 영향을 받지 않음
- 장애 투명성(Failure Transparency) : 트랜잭션, DBMS, 네트워크, 컴퓨터 장애에도 불구하고 트랜잭션을 정확하게 처리함

35. Section 029

분산 데이터베이스의 목적

- 원격 자원과 데이터를 사용함으로써 사용자의 생산성 향상과 시스템 가용성 증대
- 지역적 정보 처리의 효율성 증진
- 시스템의 확장성과 변경 용이성
- 신뢰도 증진

36. Section 029

데이터 분산의 문제점

- 질의 처리가 잘못될 경우 엄청난 양의 데이터 전송을 유발한다.
- 갱신 트랜잭션들의 상호 간섭과 모순된 판독이 일어날 수 있다.
- 교착상태의 위험이 있다.
- 프로토콜 부하를 초래할 수 있다.
- 감사가 용이하지 않다.
- 보안과 보호가 용이하지 않다.
- 회복 절차가 복잡해진다.

※ 데이터 분산 시 신뢰도는 높아진다.

37. Section 029

데이터의 실제 위치는 알 필요 없이 단지 데이터의 논리적 명칭만 가지고 접근할 수 있어야 한다. 이것을 위치 투명성이라고 하며, 분산 데이터베이스 시스템이 제공해야 할 가장 중요한 특성이다.

38. Section 029

분산 데이터베이스 시스템의 구성 요소

- 분산 처리기(Distributed Processors)
- 통신 네트워크(Communication Network)
- 분산 데이터베이스(Distributed Database)
- 분산 데이터베이스 관리 시스템(Distributed Data-base Management System)

39. Section 029

분산 DBMS

- SDD-1 : CCA 사에 의해 개발된 분산 DBMS로, ARPA 네트워크를 통신 매체로 하고 있음
- R* : System R을 확장시킨 것으로, IBM의 MVS 운영체제와 통신을 맡고 있는 소프트웨어인 CICS 환경하에서 개발되었음

40. Section 029

미들웨어(MiddleWare)

- 분산 환경에서 구성원들을 연결하고 구성원들 간의 차이를 극복하도록 범용으로 개발된 소프트웨어이다.
- 클라이언트와 서버 사이에 존재하면서 다중 통신, 데이터 액세스 프로토콜과 인터페이스 등을 지원한다.

41. Section 029

미들웨어의 종류

- 통신 미들웨어 : NOS(Network Operating System)
- 데이터베이스 미들웨어 : ODBC
- 분산 객체 미들웨어 : CORBA, DCOM

43. Section 027

DBMS는 두 사람 이상이 동시에 같은 레코드를 갱신하는 것을 허가하지 않으므로 데이터베이스의 무결성(Integrity)을 유지할 수 있다.

44. Section 024

해석 : 단일 사용자나 응용 프로그램이 데이터베이스의 내용을 접근하거나 변경하기 위해 실행되는 동작 또는 동작들의 모임이다.

45. Section 024

해석 : 트랜잭션은 활동 또는 요청이다. 주문, 구매, 교환, 추가 및 삭제는 컴퓨터에 저장된 전형적인 업무용 트랜잭션이다.

46. Section 026

로킹(Locking)

- 로킹은 주요 데이터의 액세스를 상호 배타적으로 하는 것이다.
- 트랜잭션들이 어떤 로킹 단위를 액세스하기 전에 Lock(잠금)을 요청해서 Lock이 허락되어야만 그 로킹 단위를 액세스할 수 있도록 하는 기법이다.
- 로킹 단위(Locking Granularity)
  - 병행제어에서 한꺼번에 로킹할 수 있는 단위로 로킹할 수 있는 객체의 크기를 의미한다
  - 데이터베이스, 파일, 레코드, 필드 등은 로킹 단위가 될 수 있

다.
- 로킹 단위가 크면 로크 수가 작아 관리하기 쉽지만 병행성 수준이 낮아지고, 로킹 단위가 작으면 로크 수가 많아 관리하기 복잡해 오버헤드가 증가하지만 병행성 수준이 높아진다.

5장 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 031

배열은 선형 구조로서 연속적으로 저장되므로 Sequential Memory Allocation이 가장 적합하다.

2. Section 031

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 031

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

4. Section 031

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 031

연결 리스트에서의 노드 구성 : | 데이터 | 링크 |

6. Section 031

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

7. Section 032

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 032

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 032

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 033

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

11. Section 033

- 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 034

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

13. Section 034

- 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 034

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 035

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 035

- 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 042

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 036

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 036

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

21. Section 036

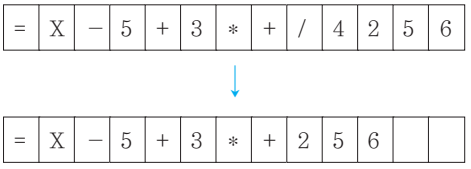
연산자의 우선순위에 맞게 괄호로 묶은 후 연산자를 해당 연산의 괄호 뒤(오른쪽)로 옮긴다.

22. Section 036

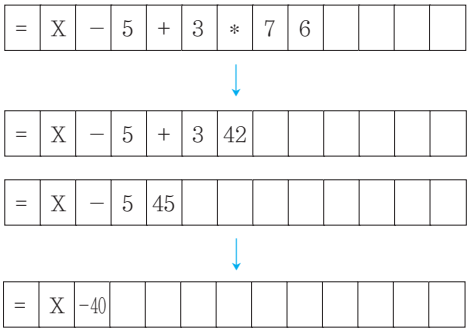
후위 순서로 운행하면 D, E, C, B, A이다.

23. Section 036

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.



6, 5, 2, +를 꺼내서 2+5의 결과와 6을 다시 넣는다.



∴ X = −40

24. Section 036

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 알 수 있다.
  ((A * (B – C)) + (D / E))

25. Section 035

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

26. Section 037

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 038

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

28. Section 039

기수 정렬(Radix Sort)

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

29. Section 038

- **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 039

- 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 039

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 039

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

33. Section 040

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

34. Section 040

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 040

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

36. Section 040

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 040

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 041

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 041

- **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 리스

트가 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 041

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

41. Section 041

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 041

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 041

- **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 043

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 039

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 043

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 043

멀티리스트 파일

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

48. Section 043

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 043

- 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 037

그래프에서 간선의 수는 노드를 연결하고 있는 선의 개수의 합이다.

51. Section 036

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.

→ (((A ↑ (B / C) + (D * E)) - (A * C))

52. Section 036

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 043

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

54. Section 041

해싱 함수 선택 시 고려할 사항 중 하나는 오버플로의 최소화이다.

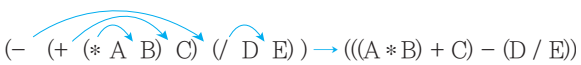
55. Section 036

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 043

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

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 044

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 044

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 044

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

4. Section 044

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

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

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

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

5. Section 044

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

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

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

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

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

$= \overline{B}(1)$

$= \overline{B}$ ← $A\cdot 1=A$

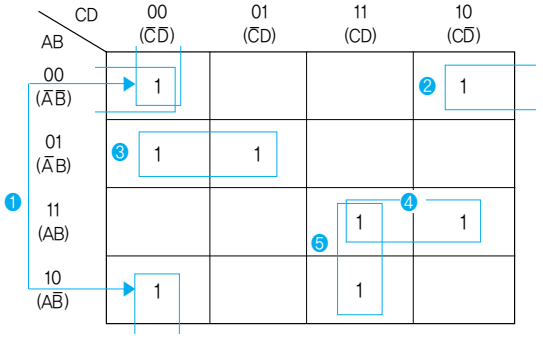
6. Section 044

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.

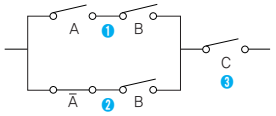
❹번 묶음 :
- 변수 A는 1 그대로 이므로 A
- 변수 B는 1 그대로 이므로 B
- 변수 C는 1에만 속하므로 C
- 변수 D는 1, 0에 모두 속하므로 무시한다.
- AND로 합치면 ABC이다.

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 045

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 044

$(\overline{A}+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})$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 045

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.

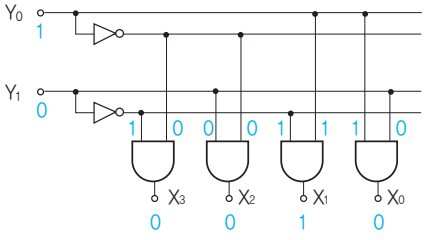
10. Section 045

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

지연시간에 반비례한다.

11. Section 045

이런 문제는 값을 직접 대입해서 풀면 된다.



13. Section 045

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

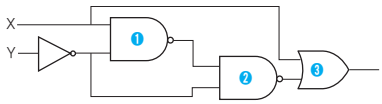
게이트의 한 선이 0으로 세트된 상태에서는 0을 입력하면 1이 출력되고, 1을 입력하면 0이 출력된다.

14. Section 045

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

※ 1의 보수는 Section 051에서 학습합니다.

15. Section 045



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

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

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

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

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

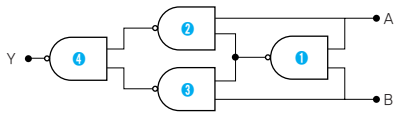
$= X(1+\overline{Y})+Y$

$= X(1)+Y$

$= X+Y$

16. Section 045

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

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

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

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

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

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

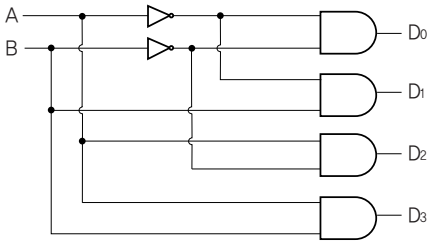
$= A\overline{B} + \overline{A}B = A \oplus B$

17. Section 047

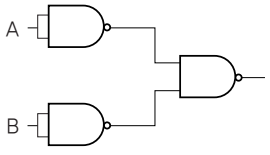
디코더는 주로 AND 게이트로 이루어져 있습니다. 2×4 디코더의

경우 2개의 NOT 게이트와 4개의 AND 게이트로 구성된다.

2×4 디코더



18. Section 045



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

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

$=(AA)+(BB)$ ← $AA=A$ 적용

$=A+B$ 이므로 OR 게이트와 같다.

19. Section 045

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 048

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

21. Section 046

반가산기

〈진리표〉

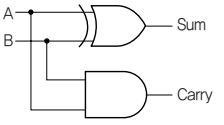
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

〈논리식〉

$Carry=A\cdot B$

$Sum=A\oplus B$

〈회로도〉



22. Section 046

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 046

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 047

Carry-Look-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 047

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 045

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 047

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 047

- 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 047

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

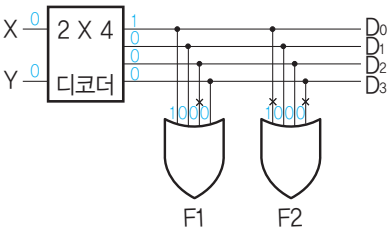
※ 그림은 2^n개의 입력 중 1개의 입력 선을 선택하여 출력하기 위해 n개의 선택 선을 사용하는 멀티플렉서이다. 출력에 Q와 $\overline{Q}$가 있어 두 개의 선이 있는 것이 아니다. $\overline{Q}$는 Q의 부정을 의미한다.

31. Section 047

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 047

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

33. Section 048

조합논리회로와 순차논리회로의 차이점

구 분	조합논리회로	순서논리회로
기억 기능	없다	있다
구성 요소	논리소자	논리소자, 플립플롭(FF)
출력 신호	입력 신호에 의해서만 결정	입력 신호와 현재의 상태에 의해서 결정
종 류	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 048

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

36. Section 048

동기식 동작(Synchronous Operation)

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

37. Section 048

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 048

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 048

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 048

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 047

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 048

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 048

※ 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 048

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 046

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

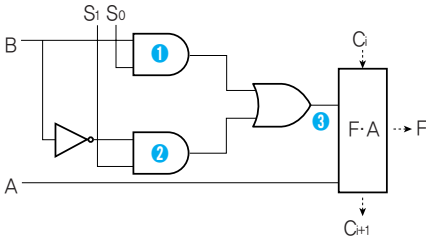
		$A_2\,A_1\,A_0$
초기상태		0 0 0
클록 순서	t_1	1 1 1
	t_2	1 1 0
	t_3	1 0 1
	t_4	1 0 0
	t_5	0 1 1
	t_6	0 1 0
	t_7	0 0 1
	t_8	0 0 0

※ 이런 문제는 값을 대입하여 추적하기보다는 3Bit로 나타낼 수

있는 수가 0~7이라는 것을 염두에 두고 문제를 해결하는 것이 좋을 것 같다.

46. Section 046

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 047

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 048

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

49. Section 044

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

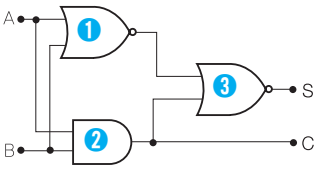
❷ $\overline{X}$Z에 해당하는 부분에 1을 표시한다. X=1, $\overline{X}$=0, Z=1, $\overline{Z}$=0이므로 X가 0이고 Z가 1인 부분에 1을 표시한다.

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 046



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

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

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

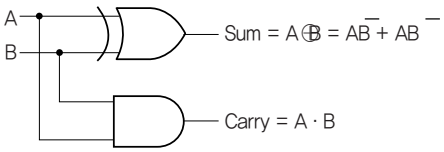
= (A+B) · $\overline{(AB)}$

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

= A⊕B

C = AB

※ 간략화한 회로를 다시 그리면 반가산기가 그려진다.



51. Section 046

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.

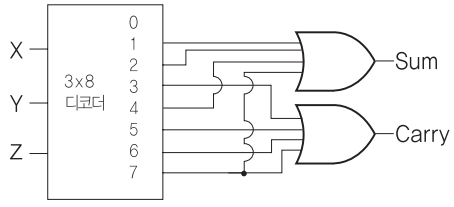
```
  1
  0
+ 1
----
 10
```

합(S_0)은 0이 되고 자리올림(C_0)은 1이 된다.

52. Section 047

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 045

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

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

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

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

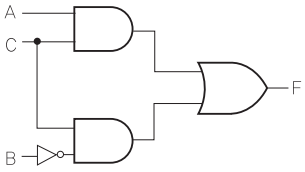
$= C(\overline{A}\,\overline{B}+A)$

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

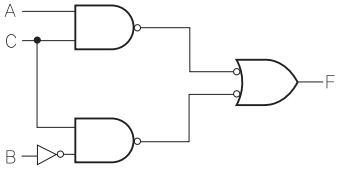
$= C(\,(1)(\overline{B}+A)\,)$ ← $A \cdot 1 = A$

$= \overline{B}C + AC$

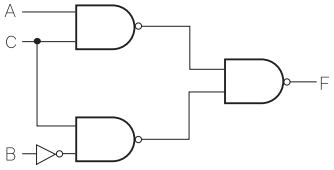
$F = AC + \overline{B}C$를 그대로 논리회로로 그리면 다음과 같다.



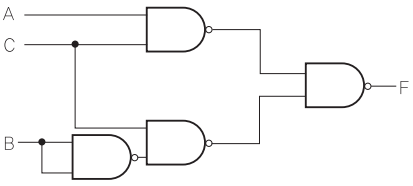
부정의 부정은 원래의 값과 같은 값을 출력하니 각각에 대해 NOT 게이트 두 개를 붙여 같은 결과를 출력하는 NAND 게이트로 변경할 수 있다.

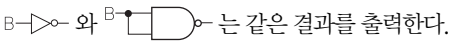


드모르강 법칙에 의하면 $\overline{A}+\overline{B}=\overline{(AB)}$이 되므로 다음과 같이 변경할 수 있다.



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 044

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 2^i(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.

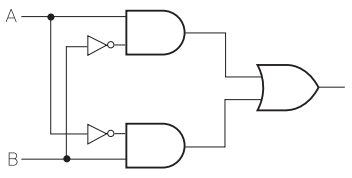
❶ 번 묶음
- 변수 w는 0에만 속하므로 $\overline{w}$
- 변수 x는 1, 0에 모두 속하므로 무시한다.
- 변수 y는 1, 0에 모두 속하므로 무시한다.
- 변수 z는 1에만 속하므로 z
- AND로 합치면 $\overline{w}$z이다.

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.

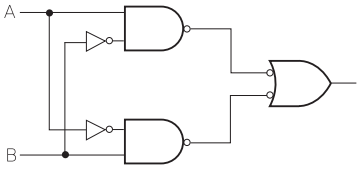
이어서 ❶과 ❷번을 OR로 묶으면 $\overline{w}$z+yz이 된다.

55. Section 047

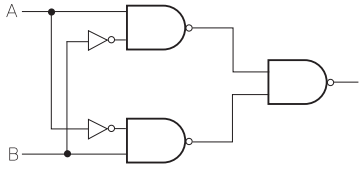
EX-OR 회로의 논리식 A⊕B= A$\overline{B}$+$\overline{A}$B를 AND, OR, NOT 게이트를 이용해서 논리회로로 그리면 다음과 같다.



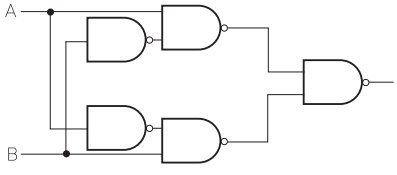
이것을 NAND 게이트만 이용하여 같은 기능을 수행하게 하려면, 부정의 부정이 원래의 값과 같은 값을 출력하는 원리를 이용하여 각각에 대해 NOT 게이트 두 개를 붙여 같은 결과를 출력하는 NAND 게이트로 변경한다.



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



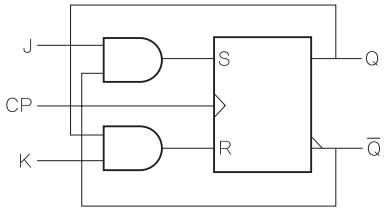
그런데 문제는 NAND만을 사용하라 했으니 Not 대신 입력이 하나인 NAND로 변경하면 된다.



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

56. Section 048

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 055

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 049

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

3. Section 049

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

4. Section 050

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 050

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 050

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

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

7. Section 050

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 050

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 051

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 052

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 052

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 052

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 052

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 052

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 052

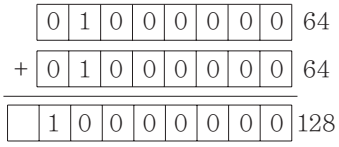
- 2의 보수를 사용하고, 크기가 1바이트면 수치가 −128~+127의 범위를 벗어나면 오버플로이다.
- 오버플로는 덧셈에 참여하는 두 수가 모두 양수이거나 음수일 때만 발생한다.
- 1의 보수나 2의 보수법에서는 부호를 포함하여 더하며, 부호가 반전되면 오버플로이다.
- 2의 보수법의 덧셈에서는 캐리가 발생하면 버린다.

① 양수끼리 더할 때 MSB에서 자리올림이 발생해야 오버플로이다.

예 64+63=127 : 오버플로 아님

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

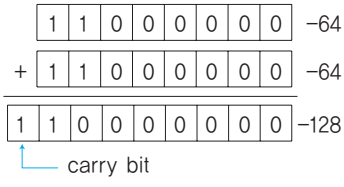
64+64=128 : 오버플로



※ MSB에서 자리올림이 발생하여 부호가 반전되었다. 즉 오버플로가 발생했다.

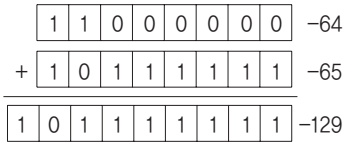
② 음수끼리 더할 때 MSB에서 자리올림이 발생하지 않으면 Overflow가 일어난다.

• −64 + −64 = −128 : 오버플로 아님



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

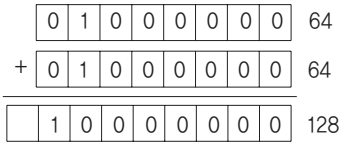
• −64 + −65 = −129 : 오버플로



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

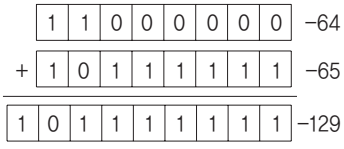
③ 부호 bit로 들어온 자리올림이 carry bit로 나가지 못하면 Overflow가 일어난다.

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 052

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 052

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

18. Section 052

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

− 32 + 16 = −48

19. Section 052

- 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 052

❶ 먼저 +15를 보기의 비트에 맞게 2진수로 표현한다.

0000 0000 0000 1111

❷ 1의 보수를 취한다.

1111 1111 1111 0000

21. Section 053

❶ 2진수로 변환한다.

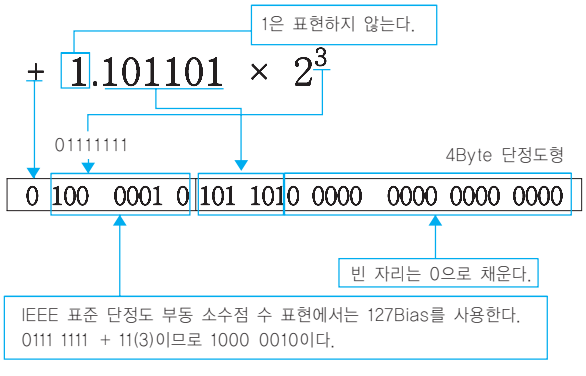
$(13.625)_{10} \rightarrow (1101.101)_2$

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 052

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 052

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

24. Section 052

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

25. Section 052

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

26. Section 052

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 052

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 052

존 형식은 연산이 불가능하며, 입 · 출력에만 이용할 수 있다.

29. Section 053

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 053

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 054

8Bit 중에서 실제 자료 비트는 7Bit이다.

∴송신 효율은 $\frac{7}{8} = 0.875$

33. Section 054

※ 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 054

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 055

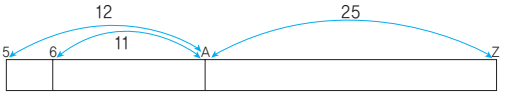
한글 2바이트 조합형 코드에서 한글과 영문을 구분하기 위해서 1비트를 사용한다.

36. Section 055

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

37. Section 055

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 055

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 055

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

40. Section 053

① IEEE 754는 부동 소수점 표현에 대한 국제 표준이다.
③ 가수가 M이고, 지수 E라면 $1.M \times 2^E$ 형태를 취한다.
④ 64비트 복수 정밀도 형식의 경우 지수는 11비트이다.

41. Section 055

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 055

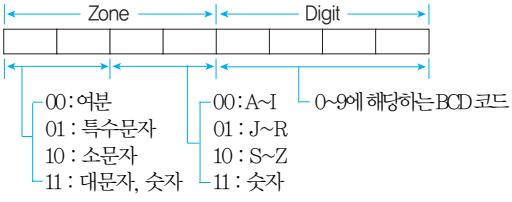
에러 검출 코드 : 해밍 코드, 비퀴너리 코드, 2 out-of 5 code(5 중 2 코드), 3 out-of 5 code(5 중 3 코드), 패리티 비트, 링카운트 코드

43. Section 055

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 055

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 053

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 052

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 051

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 051

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 052

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 056
- 중앙처리장치의 기능은 제어, 연산, 기억, 전달 기능이다.
- 입력 기능은 주변장치인 입력장치의 기능이다.

2. Section 056
DMA는 제어장치의 제어를 받지 않고 자율적인 동작으로 메모리에 접근하여 입 · 출력을 수행하는 입 · 출력 제어기이다.

3. Section 056
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 056
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 056
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 056
프로그램 카운터는 다음에 실행할 명령이 들어 있는 곳의 주소를 가지고 있는 레지스터이다.
- ② : MBR의 역할이다.
- ③ : IR의 역할이다.

7. Section 056
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 058

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.

Shift	수치 표현법	-43	+43
Shift Left	부호화 절대치	• Padding Bit : 0 10101011 → 11010110 -43×2^1, 즉 -86이 된다.	양수는 모두 같다. • Padding Bit : 0 00101011 → 01010110 43×2^1, 즉 86이 된다.
	1의 보수법	• Padding Bit : 1 11010100 → 10101001 -43×2^1, 즉 -86이 된다.	
	2의 보수법	• Padding Bit : 0 11010101 → 10101010 -43×2^1, 즉 -86이 된다.	
Shift Right	부호화 절대치	• Padding Bit : 0 • 오차 발생 : 0.5 증가 10101011 → 10010101 $-43\div2^1$ → -21.5 → -21	양수는 모두 같다. • Padding Bit : 0 00101011 → 00010101 → 21 $43\div2^1$, 즉 21.5가 되어야 하지만 오차가 발생하여 0.5가 감소한다.
	1의 보수법	• Padding Bit : 1 • 오차 발생 : 0.5 증가 11010100 → 11101010 $-43\div2^1$ → -21.5 → -21	
	2의 보수법	• Padding Bit : 1 • 오차 발생 : 0.5 감소 11010101 → 11101010 $-43\div2^1$ → -21.5 → -22	

9. Section 056

실행될 명령어는 IR(명령 레지스터)에 기억된다.

10. Section 056

부호기(Encoder)

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

11. Section 056

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

12. Section 059

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

13. Section 056

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 056

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

16. Section 057

컴퓨터 기종에 따라 인스트럭션의 형태와 주소지정방식이 다르다.

17. Section 057

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

18. Section 057

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

19. Section 057

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

20. Section 057

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 057

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

22. Section 057

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 058

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

24. Section 057

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=4×1024=4096=2^{12})

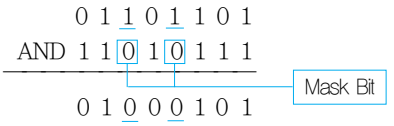
∴ 명령어 크기는 5 + 1 + 2 + 12 = 20

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 056

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 057

- 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 057

- 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 057

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

30. Section 057

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 058

10진 연산은 레지스터를 사용하지 않고 연산을 수행한다.

32. Section 058

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 057

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 060

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 058

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

36. Section 058

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

37. Section 056

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 059

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 접

근하는 횟수가 줄어든다.

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

39. Section 058

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 058

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

41. Section 058

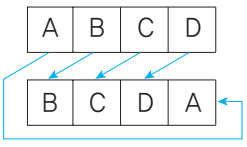
이터레이션은 해당 루틴이 정확한 결과를 산출할 때까지 해당 루틴에서 발생한 결과를 가지고 처음에 사용한 자료를 다시 수정하여 계산 작업을 반복적으로 수행하는 것을 말한다.

43. Section 060

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 058

Rotate



45. Section 058

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

46. Section 059

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 059

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

48. Section 059

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 059

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 057

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 060

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 060

Implied Mode

0번지 명령어에서 Stack의 SP가 가리키는 Operand를 암시하여 이용한다.

53. Section 060

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 060

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

55. Section 060

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 056

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

57. Section 060

상대주소(Relative Address)

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

58. Section 060

수행할 명령어 자신이 들어 있는 기억장소를 가지고 있는 레지스터는 PC이다.

59. Section 060

베이스 레지스터 주소지정방식

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

60. Section 056

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

61. Section 060

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 059

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 059

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 060

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 060

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 062

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 061

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 061

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

4. Section 061

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 061

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 061

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

7. Section 062

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 062

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 062

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

10. Section 062

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 062

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 061

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

13. Section 062

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

14. Section 062

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

15. Section 062

- 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 062

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 064

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

구 분	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 062

- 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 062

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 063

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

22. Section 063

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 063

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

제어 신호		Micro Operation	의미
C_{2t0}		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_{2t1}		M[MAR], ← MBR[AD]	• MBR에 있는 명령어의 번지 부분을 메모리의 MAR이 가리키는 위치에 전송한다.
C_{2t2}		PC ← PC + 1	PC의 값을 1 증가시킨다.
C_{2t3}	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 061

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

25. Section 063

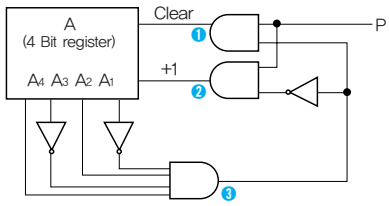
ORG는 프로그램이 시작되는 위치를 알리는 의사(Pseudo) 명령어로 프로그램이 100번지에서 시작됨을 알린다. 프로그램의 각 명령어는 다음과 같은 순서로 메모리에 위치한다.

```
        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 자리올림수를 버리므로 결과는 60이다.)
- 104 : STA DIF(Store AC in DIF)
  - AC의 값을 DIF에 저장함(60이 108번지에 저장됨)

26. Section 063



3의 출력이 1이 되어야 1의 출력이 1이 되어 클리어되므로 클리어 동작은 $A4 \cdot \overline{A3} \cdot A2 \cdot \overline{A1}$이고, 1 증가 동작은 3의 출력이 0이 되어야 2의 출력이 1이 되어 1 증가 하므로 증가 동작은 $(A4 \cdot \overline{A3} \cdot A2 \cdot \overline{A1})$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 아래와 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 061

- 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 064

- 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 061

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 064

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 064

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 064

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 064

μ-Operation부가 m Bit일 때 m개의 마이크로 동작을 표현한다.

34. Section 064

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 064

①번은 수직 마이크로 명령에 대한 설명이다.

36. Section 061

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 063

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 061

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

39. Section 061

- 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 061

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 066

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 065

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

3. Section 065

- 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 065

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 065

- 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 065

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 066

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 065

- 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 065

스풀링과 버퍼링은 모두 큐 방식으로 입·출력을 수행한다.

11. Section 065

스트로브 제어(Strobe Control) 방식의 단점

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

12. Section 065

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

13. Section 066

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 066

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 068

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

19. Section 066

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 066

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 066

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 066

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

23. Section 066

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 066

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

25. Section 066

CPU에서 DMA 제어기로 보내는 자료

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

26. Section 067

- **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 067

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

28. Section 065

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 067

인터럽트를 처리할 때 복귀주소는 스택에 저장된다.

30. Section 067

- 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 067

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 067

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 067

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

34. Section 067

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

35. Section 067

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

36. Section 067

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 067

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 067

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

39. Section 067

제어 프로그램 호출(SVC, Supervisor Call) 인터럽트

사용자가 프로그램 내에서 SVC 명령을 써서 의도적으로 호출한 경우

40. Section 068

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

41. Section 068

인터럽트 우선순위

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

※ 프로그램의 연산자나 주소지정방식의 잘못으로 인한 인터럽트는 프로그램 오류 인터럽트이다.

42. Section 068

인터럽트 서비스 루틴은 실질적인 인터럽트에 대한 조치를 취하는 단계이다.

43. Section 068

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

44. Section 068

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

45. Section 068

- 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 067

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 067

- 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 069

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 069

기억장치의 특성을 결정하는 요소(특성량)

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

3. Section 069

- 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 069

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 069

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

6. Section 069

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 069

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 070

캐시 메모리에 주로 사용 되는 것은 SRAM이다.

9. Section 069

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 070

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

11. Section 070

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 070

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

13. Section 070

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

14. Section 070

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.

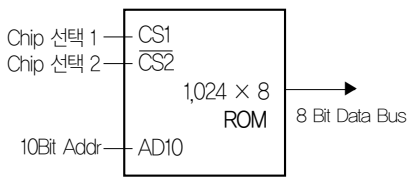
- 주소선 수
  - 주소선이 n개라면 2^n개의 메모리를 지정할 수 있다.
  - 2^{16}개의 메모리라면 16개의 주소선이 필요하다.

15. Section 070

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 070

- 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 070

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

19. Section 070

모니터 프로그램

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

20. Section 070

- 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 070

ROM(Read Only Memory)의 종류

종 류	특 징
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 070

데이터의 연산은 연산장치의 기능이다.

23. Section 070

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

24. Section 071

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

25. Section 071

보조기억장치는 입력장치인 동시에 출력장치로도 사용할 수 있다.

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

26. Section 071

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

27. Section 073

캐시의 용량이 주기억장치에 비해 적기 때문에 캐시의 각 주소 라

인은 여러 개의 주기억장치 블록들에 의해 공유된다. 그렇기 때문에 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 070

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

29. Section 071

- 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 071

- 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 071

- 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 073

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

34. Section 071

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

36. Section 071

- 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 071

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

38. Section 071

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 072

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 072

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 072

연관기억장치는 키 레지스터의 Mask Bit와 대응하는 일부 내용에 의해 액세스한다.

42. Section 072

연관기억장치는 액세스 속도를 빠르게 하는 것이 목적이다.

44. Section 070

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

45. Section 073

캐시 설계 시 고려할 사항

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

46. Section 073

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

47. Section 073

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 072

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

49. Section 073

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

50. Section 073

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) + (1μs × 0.1) = (80ns × 0.1) + (1000ns×0.1) = 108(적중률이 0.9이므로 캐시에 없을 확률은 0.1이다.)
- 72 + 108은 180ns이다.

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

51. Section 074

- 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 074

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

53. Section 071

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 074

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 074

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

56. Section 071

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 074

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 074

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 074

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 074

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 070

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

62. Section 074

- 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 074

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.

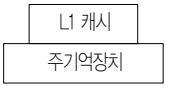
64. Section 070

- 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 073

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 074

- 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

$= 2^{20} \times 4$

$= 1,048,576 \times 4$

$= 4,194,304$

$= 4M$바이트이다.

※ $K = 2^{10} = 1,024$

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

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

※ $4M = 4 \times 1,048,576 = 4,194,304$

7장 정답 및 해설 — 병렬 컴퓨터

1.④	2.④	3.①	4.③	5.③	6.①	7.③	8.③	9.③	10.②	11.②	12.①	13.②	14.②	15.④
16.③	17.①	18.①	19.③	20.②	21.①									

1. Section 075

Flynn의 컴퓨터 구조 분류

구분	구조 및 기능
SISD	• 명령 하나가 자료 하나를 처리하는 구조 • Pipeline에 의한 시간적 병렬 처리
SIMD	• 한 개의 명령으로 여러 Data를 동시에 처리하는 구조 • Array Processor에 의한 동기적 병렬 처리(Synchronous Parallelism)
MISD	• 다수의 처리기에 의해 각각의 명령들이 하나의 Data를 처리하는 구조 • 실제로 사용되지 않는 구조 • Pineline에 의한 비동기적 병렬 처리
MIMD	• 다수의 처리기가 각각 다른 명령 흐름과 자료 흐름을 가지고 여러 개의 자료를 처리하는 구조 • Multi Processor에 의한 비동기적 병렬 처리(Asynchronous Parallelism)

3. Section 075

- 배열 처리기는 복수 모듈로 구성된다.
- Cm*, Cmmp : 카네기 멜론 대학에서 개발한 다중 처리기 시스템으로, Cm*은 50개의 LSI-11 마이크로 프로세스들로 구성하며, Cmmp는 P에-11/40 미니 컴퓨터들로 구성된 16개의 프로세서를 갖는 시스템

4. Section 075

병렬 처리 시스템의 종류

- Pipeline에 의한 시간적 병렬 처리
- Vector Processor : 파이프라인화된 벡터 프로세서, 시스톨릭 프로세서
- Array Processor에 의한 동기적 병렬 처리(Synchronous Parallelism)
- Data Flow Computer(데이터 흐름 컴퓨터)
- Multi Processor에 의한 비동기적 병렬 처리(Asynchronous Parallelism)
- 기타 : Overlay, Multi Function Unit, Memory Interlea ving

5. Section 075

마이크로 프로그래밍은 특정 명령을 수행하기 위해 마이크로 명령어를 이용하여 ROM 같은 특수한 기억장치에 프로그램으로 저장하는 것을 말한다.

9. Section 076

명령 파이프라인은 다른 프로세서 사이클을 포함하도록 확장할 수 있다.

10. Section 076

SIMD

- 한 개의 명령으로 여러 Data를 동시에 처리하는 구조
- 다수의 처리기가 한 개의 제어장치에 의해 제어된다.
- 배열 처리기(Array Processor)에 의한 동기적 병렬 처리(Synchronous Parallelism)가 해당된다.

11. Section 076

Vector Processor

- 산술 및 논리 연산, 비교, 내적 연산, 최대 · 최소값 구하기 등의 벡터 연산 명령을 빠르고 효율적으로 수행하도록 구성된 처리기이다.
- 구성 형태에 따른 종류 : 파이프라인화된 벡터 프로세서(Pipelined Vector Processor), 시스톨릭 프로세서(Systolic Processor)

12. Section 076

시스톨릭 처리기(Systolic Processor)

- 데이터 흐름과 제어 흐름이 규칙적인 특징을 갖는 시스톨릭 알고리즘을 이용하여 수행하는 처리기이다.
- 시스톨릭 알고리즘은 파이프라인화된 벡터 프로세서와 배열 프로세서의 특징을 결합한 것으로, VLSI(초고밀도 집적회로) 기법을 이용하여 구현한다.
- 신호, 화상 처리와 같은 응용에 사용하기 위해 개발된 것으로, 비용이나 성능 면에서 우수하지만 응용의 한계성과 프로그램화의 어려움이 따른다.

14. Section 076

배열 처리기(Array Processor)

- 배열 처리기는 PE(Processing Element)라고 불리는 다수의 연산기를 갖는 동기적 병렬 처리기이다.
- 명령 해독 및 제어는 제어장치가 하고, PE들은 명령 해독 능력이 결여된 수동적 장치로서 명령 처리만 한다.
- 각 PE들은 데이터 운행 연결망에 의해 상호 연결되어 PE(ALU)들을 중복해서 이용함으로써 공간적 병렬성을 얻을 수 있다.
- 벡터 계산이나 행렬 계산에 적합하다.

15. Section 076

신호, 화상 처리와 같은 응용에 사용하기 위해 개발된 것은 시스톨릭 처리기이다.

16. Section 076

벡터 프로세서와 배열 프로세서의 비교

- 공통점
  - 하나의 명령 흐름(Stream), 단일 제어장치, 단일 프로그램 계수기(PC)로 구성한다.
  - 배열 형태의 계산을 위한 목적으로 만들어졌기 때문에 언어와 컴파일에 관하여 동일한 특성을 가진다.
  - 병렬성 단위는 명령문이다.
  - 슈퍼 컴퓨터에 속함

• 차이점

	벡터 프로세서	배열 프로세서
병렬성 유지 측면	• 프로세서 내부는 산술논리장치를 총괄하고, 동시에 실행이 가능하면서 구분이 가능한 특성화된 기능장치가 다수 개 존재 • 각 기능 장치의 파이프라인화에 의해서 이루어짐	• 범용 PE가 여러 개 있고, 각 PE는 산술논리장치로 구성되어 있음 • 프로세서-메모리 인터페이스는 단일 데이터 대열만 존재
데이터 공유와 통신적인 측면	• 공유 메모리와 공유 레지스터 이용 • 기능장치 간의 통신이 불가능 • 프로세서-메모리 인터페이스에 단일 데이터 대열만 존재	• PE들 간의 직접적인 전송이나 공유 • PE들 간의 통신은 공유 메모리 없이 직접 이루어짐

17. Section 076

데이터 흐름 컴퓨터(Data Flow Computer)

- 기존의 Von Neumann형인 제어 흐름(Control-Flow) 컴퓨터와 반대되는 개념의 컴퓨터 구조이다.
- 어떤 Instruction에서 필요한 피연산자가 모두 준비되었을 때 비로소 그 Instruction을 수행하고, 수행된 결과는 그 결과를 필요로 하는 Instruction에 보내주는 방식이다.
- 어떤 Instruction이 프로그램 상의 위치와 상관없이 그 Instruction이 처리할 피연산자가 모두 준비되기만 하면 수행되기 때문에 PC가 필요 없다.
- 각 Instruction은 한 개의 형판(Template)으로 구현되며, 각 형판은 연산자 부분, 피연산자의 수신 부분, 인스트럭션의 수행결과를 보낼 목적지에 대한 포인터 부분으로 구성되어 있다.

18. Section 076

다중 처리기(Multi Processor)

- 다수의 처리기들을 상호작용이 강하도록 연결한 Tightly Coupled System
- 두 개 이상의 Processor로 구성된다.
- 기억장치와 입·출력 채널, 주변장치들을 공유하여 접근한다.
- 단일의 복합 운영체제에 의해 제어된다.
- Processor들 간의 통신은 공유기억장치나 인터럽트 네트워크를 통해서 이루어진다.

19. Section 076

- 파이프라인 단계 수가 4이므로 동시에 4단계의 명령을 처리할 수 있다.
- 4단계 파이프라인에서는 하나의 명령이 4단계를 거쳐 처리된다.
- 파이프라인 클록 주파수가 1MHz이므로 1 클럭에 소요되는 시간은 1/1000000=0.000001=1마이크로 초다.
- 각각의 명령이 4단계의 파이프라인을 거쳐 수행되는 순서는 다음과 같다.
  - 1번 째 클록 : | 1 | | | |
    1번 째 명령이 파이프라인의 첫 번째 단계로 들어온다.
  - 2번 째 클록 : | 2 | 1 | | |
    1번 째 명령이 파이프라인의 두 번째 단계로 이동하고 2번 째 명령이 파이프라인의 첫 번째 단계로 들어온다.
  - 3번 째 클록 : | 3 | 2 | 1 | |
    1번 째 명령이 파이프라인의 3 번 째 단계로, 2번 째 명령이 2단계로 이동하고, 3번 째 명령이 파이프라인의 첫 번째 단계로 들어온다.
  - 4번 째 클록 : | 4 | 3 | 2 | 1 |
    1번 째 명령이 파이프라인의 4단계로, 2번 째 명령이 3단계로 이동하고, 3번 째 명령이 2단계로 이동하고 4번 째 명령이 파이프라인의 첫 단계로 들어온다.
  - 5번 째 클록 : | 5 | 4 | 3 | 2 |
    1번 째 명령이 끝나고 2번째 명령이 파이프라인의 4단계로, 3번 째 명령이 3단계로 이동하고, 4번 째 명령이 2단계로 이동하고 5번 째 명령이 파이프라인의 첫 단계로 들어온다.
  - 6번 째 클록 : | 6 | 5 | 4 | 3 |
  - 7번 째 클록 : | 7 | 6 | 5 | 4 |
  - 8번 째 클록 : | 8 | 7 | 6 | 5 |
  - 9번 째 클록 : | 9 | 8 | 7 | 6 |
  - 10번 째 클록 : | 10 | 9 | 8 | 7 |
  - 11번 째 클록 : | 11 | 10 | 9 | 8 |
  - 12번 째 클록 : | 12 | 11 | 10 | 9 |
  - 13번 째 클록 : | 13 | 12 | 11 | 10 |

총 13번의 클럭이 필요하므로 13 마이크로 초가 소요된다.

21. Section 076

파이프라인 프로세서(Pipeline Processor)

- 파이프라인은 CPU의 처리 속도를 높이기 위해 여러 개의 명령(Instruction)을 동시에 병렬 처리하는 장치로, 분업화의 원리를 활용하여 시간적 병렬 처리를 한다.
- 입력 태스크(Task)를 입력의 서브 태스크(Sub Task)로 나눈 다

음 서브 태스크별로 동시에 처리할 수 있도록 하여 처리능력을 크게 향상시킨다.

- **파이프라인 기법의 장 · 단점**
  - 일단 파이프라인이 차고 나면 연속적으로 결과를 얻을 수 있으므로 연산 속도가 빠르다.
  - 같은 연산이 여러 번 반복되어 사용되면 효율적이지만, 그렇지 않는 경우에는 구조가 복잡하고 시간이 오래 걸린다.

3과목 운영체제

1장 정답 및 해설 — 운영체제의 개요

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

1. Section 077

기차표 판매 시스템은 응용 소프트웨어의 일종이다.

2. Section 078

- 컴파일러(Compiler) : 고급 언어로 작성된 원시 프로그램을 목적 프로그램으로 번역하는 프로그램
- 매크로(Macro) : 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 것
- 로더(Loader) : 로드 모듈을 디스크 등의 보조기억장치에서 주기억장치로 적재하는 프로그램

3. Section 078

운영체제가 한 가지 기종의 시스템에 전문적인 기능을 가지게 되면 이식성이 없어 다른 시스템과의 호환성이 떨어지게 된다.

4. Section 078

두 개 이상의 목적 프로그램을 합쳐서 실행 가능한 프로그램을 만드는 것은 링커(Linker)의 기능이다.

5. Section 078

원시 프로그램을 목적 프로그램으로 변환하는 것은 언어 번역 프로그램의 역할이다.

6. Section 078

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 078

VP400은 컴퓨터 시스템의 명칭이다.

8. Section 078

운영체제의 성능 평가 기준

- 처리 능력(Throughput) : 일정 시간 내에 시스템이 처리하는 일의 양
- 반환 시간(Turn Around Time) : 시스템에 작업을 의뢰한 시간부터 처리가 완료될 때까지 걸린 시간
- 사용 가능도(Availability) : 시스템을 사용할 필요가 있을 때 즉시 사용 가능한 정도
- 신뢰도(Reliability) : 시스템이 주어진 문제를 정확하게 해결하는 정도

9. Section 078

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

10. Section 077

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

11. Section 078

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

※ 마이크로 프로그램은 ROM이나 PROM에 영구히 기록되는 프로그램을 의미한다.

12. Section 083

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, Boot Strap)이 수행된다.

13. Section 079

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 078

운영체제 설계 시 고려해야 할 사항에는 사용자와 컴퓨터 간의 인터페이스 제공, 자원의 효율적인 운영 및 자원 스케줄링, 데이터 공유 및 주변장치 관리, 처리 능력 및 신뢰도 향상, 사용 가능도 향상, 응답(경과) 시간 단축, 반환 시간 단축 등이 있다.

15. Section 079

시분할 시스템

- 여러 명이 사용하는 시스템에서 CPU가 사용자들의 프로그램을 번갈아 가며 처리해 줌으로써 각 사용자에게 독립된 컴퓨터를 사용하는 느낌을 주는 것이며 라운드 로빈(Round Robin) 방식이라고도 한다.
- 여러 사용자가 각자의 단말장치를 통하여 동시에 운영체제와 대화하면서 각자의 프로그램을 실행한다.
- 하나의 CPU는 같은 시점에서 여러 개의 작업을 동시에 수행할 수 없기 때문에, CPU의 전체 사용 시간을 작은 작업 시간량(Time Slice, Quantum)으로 나누어서 그 시간량 동안만 번갈아 가면서 CPU를 할당하여 각 작업을 처리한다.
- 다중 프로그래밍 방식과 결합하여 모든 작업이 동시에 진행되는 것처럼 대화식 처리가 가능하다.
- 시스템의 전체 효율은 좋아지나 개인별 사용자 입장에서는 반응 속도가 느려질 수 있다.
- 각 작업에 대한 응답 시간을 최소한으로 줄이는 것을 목적으로 한다.

16. Section 079

- 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 한번에 처리하게 되므로, 병행처리를 하여 반응 속도를 빠르게 하는 것과는 관련이 적다.

17. Section 079

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, 전화 교환장치의 제어, 은행의 On-Line 업무 등 시간에 제한을 두고 수행되어야 하는 작업에 사용된다.

18. Section 083

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

19. Section 082

구 분	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 실행 시간은 약간 느려짐	

20. Section 082

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 082

- Micro-programming : 하드웨어를 제어할 수 있는 제어 워드를 이용하여 펌웨어(FirmWare)에 저장될 마이크로 프로그램을 작성하는 것
- Function : 고급 프로그래밍 언어에서 미리 정의된 서브루틴으로서, 주어진 인수에 따라 어떤 값을 계산하여 돌려줌

22. Section 082

매크로는 어셈블리(Assembly) 언어로, 프로그램에서 반복되는 일련의 같은 연산을 효과적으로 프로그램하기 위해 작성해 놓은 코드 모음으로서, 고급 언어에서 부 프로그램을 호출하는 CALL 명령과 유사하다.

23. Section 082

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

24. Section 082

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

25. Section 082

Macro Processor는 Compiler와도 같이 사용될 수 있다.

26. Section 077

- Control Program : 운영체제의 제어 프로그램, 즉 감시 프로그램, 작업 제어 프로그램, 데이터 관리 프로그램을 의미
- 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, 이런 스케줄링 작업을 수행하는 것을 의미
- Bench Mark Program : 컴퓨터 속도 등의 성능을 평가하기 위한 프로그램을 이용해 사용자가 준비한 기본 문제들을 처리하여 문제 해결의 결과로서 성능을 판정하는 방법

27. Section 080

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

28. Section 081

- 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

29. Section 081

Pass-1의 기능

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

30. Section 081

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

31. Section 081

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

32. Section 083

재배치 로더의 수행 순서 : 주기억장치 할당(Allocation) → 연결(Linking) → 재배치(Relocation) → 적재(Loading)

33. Section 083

- 로더는 기억장치 할당, 연결, 재배치, 적재 기능을 수행하지만 최적화 기능을 수행하는 것은 아니다.
- 명령을 최적화하려면 명령어 최적화기 등을 이용해야 한다.

34. Section 083

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.

35. Section 080, 083

원시 프로그램이 수행되기까지의 순서

원시 프로그램은 번역 프로그램(Compiler)을 통해 번역된 후 링커(Lingker)를 통해 각 목적 프로그램이나 로드 모듈을 연결한 다음 적재기(Loader)에 의해 주기억장치에 적재된다.

36. Section 079

- 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 영상 표시장치(모니터)가 적합하다.

37. Section 079

일괄 처리 시스템(Batch Processing System)

- 초기의 컴퓨터 시스템에서 사용된 형태로, 일정량 또는 일정 기간 동안 데이터를 모아서 한꺼번에 처리하는 방식이다.
- 급여 계산, 지불 계산, 연말 결산 등의 업무에 사용된다.

38. Section 079

다중 처리(Multi-Processing)는 여러 개의 CPU와 하나의 주기억장치를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식이고, 다중 프로그래밍(Multi-Programming)은 하나의 CPU와 주기억장치를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식이다.

39. Section 080

- 프로그래밍 언어는 저급 언어와 고급 언어(컴파일러 언어)로 분류하는데, 다시 저급 언어는 기계어와 어셈블리어로 분류한다.
- **레지스터(Register)** : CPU 내부에서 처리할 명령어나 연산의 중간 결과값 등을 일시적으로 기억하는 임시기억장소

40. Section 082

리커션(Recursion, 재귀) 프로그램은 한 루틴이 다시 자기 자신을 호출하는 프로그램을 의미한다.

41. Section 078

목적 프로그램을 생성하는 것은 컴파일러의 역할이다.

42. Section 078

운영체제를 계층 구조로 나누어 나열하면 '하드웨어 – CPU 관리 – 기억장치 관리 – 프로세스 관리 – 주변장치 관리 – 파일 시스템 관리 – 사용자 프로세스' 순이다.

43. Section 078

운영체제를 자원 관리자(Resource Manager)라는 관점으로 보았을 때 자원을 관리하는 순서는 다음과 같다.

❶ 어떤 프로세스에게 언제, 어떤 자원을 할당할 것인가를 결정하는 분배 정책 수립 과정

❷ 시스템 내 모든 자원들의 상태를 파악하는 과정

❸ 자원을 배당하고 운영함으로써 수립된 정책을 수행하는 과정

❹ 프로세스에 배당된 자원을 회수하는 과정

44. Section 078

③번은 언어 번역 프로그램의 기능이다.

46. Section 083

- **Loading** : 실행 프로그램을 할당된 기억공간에 실제로 옮기는 기능
- **Relocation** : 디스크 등의 보조기억장치에 저장된 프로그램이 사용하는 각 주소들을 할당된 기억장소의 실제 주소로 배치시키는 기능
- **Linking** : 부 프로그램 호출 시 그 부 프로그램이 할당된 기억장소의 시작주소를 호출한 부분에 등록하여 연결하는 기능

47. Section 083

- **직접 연결 로더(Direct Linking Loader)** : 일반적인 기능의 로더로, 로더의 기본 기능 4가지를 모두 수행하며, 재배치 로더(Relocation Loader), 상대 로더(Relative Loader)라고도 함
- **동적 적재 로더(Dynamic Loading Loader)** : 프로그램을 한꺼번에 적재하는 것이 아니라 실행시 필요한 일부분만을 적재하는 것으로, 호출시 적재(Load- On-Call)라고도 하며, 프로그램의 크기가 주기억장치의 크기보다 큰 경우에 유리함

2장 정답 및 해설 — 프로세스 관리

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

1. Section 084

트랩 오류, 프로그램 요구, 입 · 출력 인터럽트에 대해 조치를 취하는 것은 운영체제의 역할이다.

2. Section 084

- PCB에 포함되는 정보 : 프로세스의 현재 상태, 포인터, 프로세스 고유 식별자, 스케줄링 및 프로세스의 우선순위, CPU 레지스터 정보 등
- 파일 할당 테이블(FAT) : 디스크에 저장된 파일에 대한 정보가 어느 위치에 저장되어 있는가를 표시해 놓은 영역

3. Section 085

대화형(시분할) 시스템은 여러 명의 사용자가 사용하는 시스템에서 컴퓨터가 사용자들의 프로그램을 번갈아가며 처리해 줌으로써 각 사용자에게 독립된 컴퓨터를 사용하는 느낌을 주는 것이다. 이러한 대화형 시스템에서는 각 사용자가 작업을 지시하고 수행하는 시간이 빨라야 하므로 각 사용자가 작업을 지시하고 반응하기 시작하는 시간, 즉 반응 시간(Response Time)이 가장 중요한 요소(인자)가 된다.

4. Section 084

- 실행 상태에서 준비 상태로 전이되는 것은 주어진 시간 할당량을 모두 사용한 경우이다.
- ②번은 실행 상태에서 대기(Wait, Block) 상태로 이동하는 경우이다.
- ④번은 대기 상태에서 준비 상태로 이동하는 경우이다.

5. Section 084

- 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 · 출력이 완료되어 대기 상태에서 다시 준비 상태로 전이되는 것

6. Section 084

다중 스레드 프로그램을 사용하면 응용 프로그램의 응답 시간을 단축시킬 수 있다.

7. Section 089

- 예방 기법(Prevention) : 교착상태가 발생하지 않도록 사전에 시스템을 제어하는 방법으로, 교착상태 발생의 4가지 조건 중에서 어느 하나를 제거(부정)함으로써 수행되며, 자원의 낭비가 가장 심한 기법임
- 회피 기법(Avoidance) : 교착상태 회피 기법은 교착상태가 발생할 가능성을 배제하지 않고 교착상태가 발생하면 적절히 피해나가는 방법으로, 주로 은행원 알고리즘(Banker's Algorithm)이 사용됨
- 회복 기법(Recovery) : 교착상태를 일으킨 프로세스를 종료하거나 교착상태의 프로세스에 할당된 자원을 선점하여 프로세스나 자원을 회복하는 것을 의미함

9. Section 084

- 국부성(Locality) : 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다는 이론
- 페이징(Paging) 기법 : 가상기억장치에 보관되어 있는 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 나눠진 프로그램을 주기억장치의 동일하게 나눠진 영역에 적재시켜 실행하는 기법
- 문맥 교환(Context Switching) : 하나의 프로세스에서 다른 프로세스로 CPU가 할당되는 과정에서 발생되는 것으로, 새로운 프

로세스에 CPU를 할당하기 위해 현재 CPU가 할당된 프로세스의 상태 정보를 저장하고, 새로운 프로세스의 상태 정보를 설정한 후 CPU를 할당하여 실행되도록 하는 작업

10. Section 084

- 프로세스의 모든 내용이 디스크에 접수되면 Job Scheduler(장기 스케줄러)가 그 프로세스를 스케줄링 큐(준비상태 큐)에 입력하여 실행 준비 상태로 만든다.
- **교통량 제어기(Traffic Controller)** : 프로세스의 상태에 대한 조사와 통보 담당
- **프로세서 스케줄러(Processor Scheduler)** : 준비 상태에서 실행 상태로의 전이 수행

11. Section 084

프로세서를 회수하는 것은 프로세스 종료나 오류 상황 또는 다른 작업에게 프로세서를 할당하기 위해서이다.

12. Section 085

스케줄링 정책을 수립하는 경우 고려할 요소 : 자원의 제한성, 자원의 요구도, 체제의 균형, 시한성, 자원의 유용도

13. Section 085

바인딩 시간(Binding Time) : 프로그램에서 어떤 요소의 이름을 그것이 나타내는 실제의 대상물과 연결하는 시간으로, 스케줄링 알고리즘과는 무관함

14. Section 087

- **비선점 스케줄링 기법** : FCFS, SJF, 우선순위, HRN, 기한부
- **선점 스케줄링 기법** : SRT, 선점 우선순위, Round Robin, 다단계 큐, 다단계 피드백 큐

15. Section 087

선점 기법은 우선순위가 높은 프로세스를 먼저 처리하므로 프로세스에 대한 요구를 공정히 처리하지 못한다.

16. Section 087

- 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

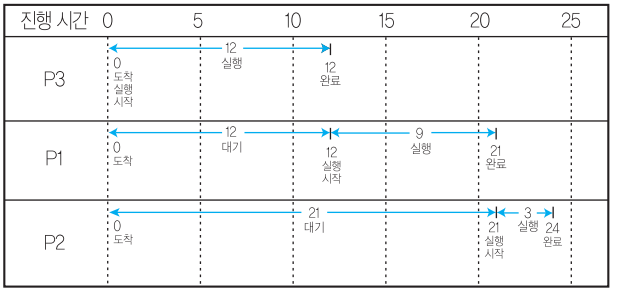
17. Section 085

TAT는 프로세서 스케줄링의 성능을 측정하기 위해 사용하는 것으로, 프로세스를 제출한 시간부터 실행이 완료될 때까지 걸리는 시간을 의미한다.

18. Section 086

임의의 작업 순서를 지정하여 최대 평균 반환 시간을 구하려면 실행 시간이 가장 큰 것부터 차례로 수행하면 되고, 최소 평균 반환 시간을 구하려면 실행 시간이 가장 작은 것부터 차례대로 수행하면 된다. 최대 평균 반환 시간과 최소 평균 반환 시간을 구하는 방법은 다음과 같다.

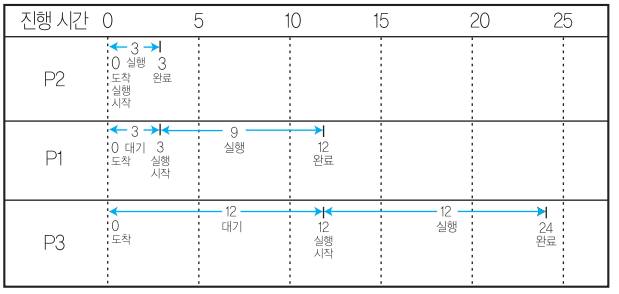
최대 평균 반환 시간



- P3 : 도착하자마자 실행하여 12에서 작업이 완료되므로 대기 시간은 0이고, 반환 시간은 12이다.
- P1 : 0에 도착하여 P3이 완료될 때까지 대기한 후 P3이 완료되는 12에서 실행을 시작하여 21에 작업이 완료된다. 그러므로 대기 시간은 12이고, 반환 시간은 21이다.
- P2 : 0에 도착하여 P1이 완료될 때까지 대기한 후 P1이 완료되는 21에서 실행을 시작하여 24에 작업이 완료된다. 그러므로 대기 시간은 21이고, 반환 시간은 24이다.

∴ 평균 반환 시간은 (12+21+24)/3 = 19 이다.

최소 평균 반환 시간



- P2 : 도착하자마자 실행하여 3에서 작업이 완료되므로 대기 시간은 0이고, 반환 시간은 3이다.

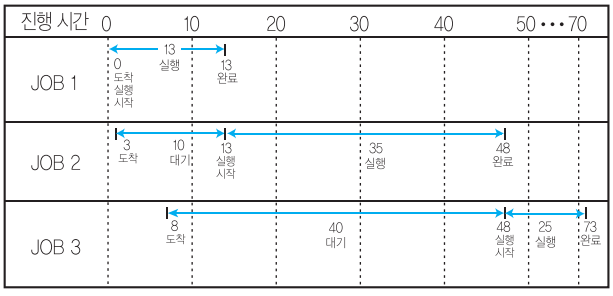
- P1 : 0에 도착하여 P2가 완료될 때까지 대기한 후 P2가 완료되는 3에서 실행을 시작하여 12에 작업이 완료된다. 그러므로 대기 시간은 3이고, 반환 시간은 12이다.
- P3 : 0에 도착하여 P1이 완료될 때까지 대기한 후 P1이 완료되는 12에서 실행을 시작하여 24에 작업이 완료된다. 그러므로 대기 시간은 12이고, 반환 시간은 24이다.

∴ 평균 반환 시간은 (3+12+24)/3 = 13 이다.

※ 최대 평균 반환 시간(T) – 최소 평균 반환 시간(t)은 19 – 13 = 6 이다.

19. Section 086

FIFO 스케줄링은 준비상태 큐에 도착한 순서에 따라 CPU를 할당하는 기법이다.



- JOB 1 : 도착하자마자 실행하여 13에서 작업이 완료되므로 대기 시간은 0이고, 반환 시간은 13이다.
- JOB 2 : 3에 도착하여 JOB 1이 완료될 때까지 대기한 후 JOB 1이 완료되는 13에서 실행을 시작하여 48에 작업이 완료된다. 그러므로 대기 시간은 10이고, 반환 시간은 45이다.
- JOB 3 : 8에 도착하여 JOB 2가 완료될 때까지 대기한 후 JOB 2가 완료되는 48에서 실행을 시작하여 73에 작업이 완료된다. 그러므로 대기 시간은 40이고, 반환 시간은 65이다.

∴ 평균 반환 시간은 (13+45+65)/3 = 41

20. Section 087

RR 방식은 FIFO 방식의 선점형으로, 시간 간격이 크면 FIFO와 비슷한 스케줄링이 된다.

22. Section 087

RR에서 사용되는 일정한 시간량을 Time Slice 또는 Quantum이라고 한다.

23. Section 086, 087

- 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

24. Section 086

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.

25. Section 087

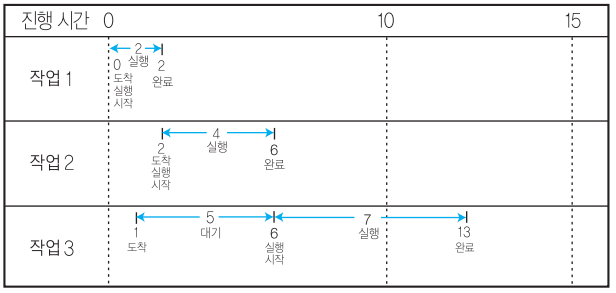
- 선점 우선순위 : 준비상태 큐의 프로세스들 중에서 우선 순위가 가장 높은 프로세스에게 먼저 CPU를 할당하는 기법
- SRT : 현재 실행중인 프로세스의 남은 시간과 준비상태 큐에 새로 도착한 프로세스의 실행 시간을 비교하여 가장 짧은 실행 시간을 요구하는 프로세스에게 CPU를 할당하는 기법
- FCFS : 준비상태 큐(대기 큐, 작업준비 큐, 스케줄링 큐)에 도착한 순서에 따라 차례로 CPU를 할당하는 기법
- RR : FCFS 기법과 같이 준비상태 큐에 먼저 들어온 프로세스가 먼저 CPU를 할당받지만 각 프로세스는 시간 할당량(Time Slice, Quantum) 동안만 실행된 후 다음 프로세스에게 CPU를 넘겨주는 기법

※ 시분할 시스템에는 선점 스케줄링 기법이 적당하고, 그 중에서 일정 시간 할당량만큼씩 동작해 주는 기법이 가장 적당하다.

26. Section 086

SJF는 준비상태 큐에서 기다리고 있는 프로세스들 중에서 실행 시간이 가장 짧은 프로세스에게 먼저 CPU를 할당하는 기법이다. 그러므로 수행 순서와 평균 반환 시간은 다음과 같다.

반환 시간 계산



- 작업 1 : 도착하자마자 실행하여 2에서 작업이 완료되므로 대기 시간은 0이고, 반환 시간은 2이다.
- 작업 2 : 2에 도착하여 작업 1이 완료된 2에서 실행을 시작하여 6에 작업이 완료되므로 대기 시간은 0이고, 반환시간은 4이다.
- 작업 3 : 1에 도착하여 작업 2가 완료될 때까지 대기한 후 작업 2가 완료되는 6에서 실행을 시작하여 13에 작업이 완료되므로 대기 시간은 5이고, 반환시간은 12이다.

∴ 평균 반환 시간은 (2+4+12)/3 = 6 이다.

27. Section 087

- 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

28. Section 084, 087, 089

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.

29. Section 086

- 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 088

- 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

32. Section 088

상호 배제 문제를 해결하기 위한 것에는 데커 알고리즘, 피터슨 알고리즘, 빵집 알고리즘, 세마포어, 모니터 등이 있다.

33. Section 089

교착상태 발생의 필요 충분 조건 : 상호 배제(Mutual Exclusion), 점유와 대기(Hold & Wait), 비선점(Non-Preemption), 환형 대기(Circular Wait)

34. Section 089

교착상태 예방 기법

- 상호 배제(Mutual Exclusion) 부정 : 한 번에 여러 개의 프로세스가 공유자원을 사용할 수 있도록 함
- 점유 및 대기(Hold & Wait) 부정 : 프로세스가 실행되기 전 필요한 모든 자원을 할당하여 프로세스 대기를 없애거나 자원이 점유되지 않은 상태에서만 자원을 요구하도록 함
- 비선점(Non-preemption) 부정 : 자원을 점유하고 있는 프로세스가 다른 자원을 요구할 때 점유하고 있는 자원을 반납하고, 요

구한 자원을 사용하기 위해 기다리게 함

- **환형 대기(Circular Wait) 부정** : 자원을 선형 순서로 분류하여 고유 번호를 할당하고, 각 프로세스는 현재 점유한 자원의 고유 번호보다 앞이나 뒤 어느 한쪽 방향으로만 자원을 요구하도록 하는 것

35. Section 089

교착상태의 해결 방법에는 교착상태 예방 기법(Prevention), 교착상태 회피 기법(Avoidance), 교착상태 발견 기법(Detection), 교착상태 회복 기법(Recovery)이 있다.

36. Section 088

임계 영역에서는 하나의 프로세스가 처리하는 동안 다른 프로세스의 접근이 허용되지 않으므로, 임계 영역 안에서는 인터럽트가 발생되지 않도록 해야 한다.

38. Section 089

교착 상태 예방 기법은 교착 상태가 발생하지 않도록 사전에 시스템을 제어하는 방법으로, 교착 상태 발생의 4가지 조건 중에서 어느 하나를 제거(부정)함으로써 수행된다. 그러나 상호 배제 조건의 부정은 실제로 구현하지 않는데, 그 이유는 공유 자원은 어느 한 시점에서 하나의 프로세스만 사용할 수 있어야 하기 때문이다.

39. Section 089

- 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

40. Section 089

세마포어는 프로세스 동기화를 위해 사용하는 기법이다.

41. Section 084

스레드(Thread)는 프로세스 내에서의 작업 단위이므로 스레드는 외부에 존재할 수 없다.

43. Section 084

CPU를 할당받은 프로세스는 프로그램 실행이 종료되거나 교착상태, 인터럽트, 오류 등이 발생하는 것과 같이 더 이상 프로그램 실행을 진행할 수 없는 경우를 제외하고는 자기에게 배당된 Time Slice 동안 CPU를 독점하여 사용하게 된다.

44. Section 088

문제에 주어진 프로세스의 선행 순서를 정리하면 P1〉P2〉P4, P3〉P4가 된다. P1 프로세스가 가장 먼저 실행되고, 그 다음 P2 프로세스, 그 다음은 P4 프로세스가 실행된다. 또한 P4 프로세스가 실행되기 전에 P3 프로세스가 실행되어야 한다. 그러므로 P1과 P3, P2와 P3은 병행 처리가 가능하게 된다.

45. Section 088

세마포어에서 P 연산은 자원을 사용하려는 프로세스들이 진입 여부를 자원의 개수(S)를 통해 결정하는 것으로, 자원의 개수를 감소시켜(S = S − 1) 자원이 점유되었음을 알린다. V 연산은 대기중인 프로세스를 깨우는 신호로서 자원의 개수를 증가시켜(S = S + 1) 자원이 반납되었음을 알린다.

46. Section 088

데커(Dekker) 알고리즘은 소프트웨어적으로 상호 배제 기법을 구현하는 방법이며, 임계 영역은 특정 프로세스가 독점할 수 없기 때문에 임계 영역에 들어가는 것이 무한정 지연될 수는 없다.

47. Section 089

순환대기(Circular Wait) 상황을 허용하지 않는다는 것은 각 프로세스가 한 쪽 방향의 자원만을 요청하도록 하는 것이다. 그러므로 프로세스가 가지고 있는 자원의 앞쪽이나 뒤쪽 중 한 쪽 방향의 자원만을 사용하게 해야지 앞 또는 뒤쪽에 있는 자원들을 자유롭게 요청하게 해서는 안된다.

3장 정답 및 해설 — 기억장치 관리

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

1. Section 091

- 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 095

실행 중인 프로세스가 일정 시간 동안에 참조하는 페이지의 집합은 워킹 셋(Working Set)이다.

3. Section 091

- 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 092

- 교착상태(Dead Lock)는 둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상을 의미한다.
- ①번은 압축, ②번은 재배치, ③번은 통합을 의미한다.

5. Section 092

- 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 092, 093

페이징 기법에서는 외부 단편화가 존재할 수 없다.

7. Section 092

- 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 090

- 반입(Fetch) 전략 : 보조기억장치에 보관중인 프로그램이나 데이터 등을 언제 주기억장치로 적재할 것인지를 결정하는 전략
- 압축(Compaction) : 주기억장치 내에 분산되어 있는 단편화된 빈 공간을 결합하여 하나의 큰 기억 공간을 만드는 작업
- 교체(Replacement) : 주기억장치의 모든 영역이 이미 사용중인 상태에서 새로운 프로그램이나 데이터를 주기억장치에 배치하려고 할 때, 이미 사용되고 있는 영역 중에서 어느 하나를 선택하여 사용할 것인지를 결정하는 전략

9. Section 091

- 스와핑(Swapping) : 하나의 프로그램 전체를 주기억장치에 할

당하여 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 090

Best Fit은 기억장치 배치(Placement) 전략 중 하나로, 내부 단편화 공간이 가장 적게 발생되는 영역에 배치하는 기법이다.

11. Section 090

- 최초 적합(First Fit) : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치시키는 방법
- 최적 적합(Best Fit) : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 작게 남기는 영역에 배치시키는 방법

12. Section 090

- 최초 적합(First Fit) : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치시키는 방법
- 최악 적합(Worst Fit) : 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 크게 남기는 분할 영역에 배치시키는 방법

13. Section 093

가상기억장치

- 보조기억장치(하드디스크)의 일부를 주기억장치처럼 사용하는 것으로, 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법이다.
- 가상기억장치의 일반적인 구현 방법에는 블록의 종류에 따라 페이징 기법과 세그먼테이션 기법으로 나눌 수 있다.
- 페이징 기법 : 프로그램을 동일한 크기로 나눈 단위를 페이지라 하며 이 페이지를 블록으로 사용하는 기법
- 세그먼테이션 기법 : 프로그램을 가변적인 크기로 나눈 단위를 세그먼트라하며 이 세그먼트를 블록으로 사용하는 기법

14. Section 090

13K 작업을 최초 적합으로 배치하면 16K 영역에, 최적 적합으로 배치하면 14K 영역에, 최악 적합으로 배치하면 30K 영역에 배치된다.

15. Section 093

매핑 작업은 가상주소를 실기억주소로 바꾸는 것을 의미하며, 이러한 매핑은 페이지 맵 테이블이나 세그먼트 맵 테이블을 이용하여 운영체제에 의해 이루어진다.

16. Section 093

가상 메모리를 사용하면 CPU의 처리율이 높아지며, 주기억장치보다 큰 프로그램을 실행할 수 있고, 가상기억장치의 페이징 기법이나 세그먼테이션 기법을 사용하므로 오버레이 기법을 구현할 필요가 없다.

17. Section 091, 093

교체(Swapping) 기법은 하나의 프로그램 전체를 주기억장치에 할당하여 사용하다가 필요에 따라 다른 프로그램과 교체하는 기법이다.

18. Section 093

페이지 테이블은 기억장치에 저장되기 때문에 페이지 테이블을 사용함으로써 기억장소를 낭비하게 된다.

19. Section 093

가상 메모리 기법에서 보조기억장치를 같은 크기인 페이지로 나누어서 활용하는 시스템을 Paging system이라 한다. 같은 크기로 나누어진 물리적 메모리(주기억장치)는 블록이라고 하므로 Paging system이란 보조기억장치가 여러 개의 Page로 구분된 것을 말한다.

20. Section 093

일정한 크기의 페이지로 프로그램을 나눈다고 하더라도 해당 프로그램이 항상 페이지 크기의 승수로 끝나는 것이 아니므로 내부 단편화는 발생할 수 있다. 만약 4K 단위로 페이지를 나누었는데 프로그램의 총 크기가 22K이면 2K의 내부 단편화가 발생된다.

21. Section 093

1MB = 1,000KB이므로 8MB의 주기억장치는 8,000 KB를 1KB로 나눈 만큼의 페이지를 갖게 된다.

22. Section 093

- 페이지 테이블에 존재해야 하는 항목의 개수는 페이지의 개수와 같다.
- 페이지 개수는 다음과 같이 계산할 수 있다. 논리적 공간의 크기 ÷ 페이지의 크기 = 페이지의 개수($2^{32} \div 2^{21} = 2^{11}$)

23. Section 094

SCR

- 가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지의 교체를 방지하기 위한 것으로 FIFO 기법의 단점을 보완하는 기법이다.
- 각 페이지마다 LRU 근사 알고리즘과 같이 참조 비트를 두고, FIFO 기법을 이용하여 페이지 교체 수행중 참조 비트가 0일 경우에는 교체하고 참조 비트가 1일 경우에는 참조 비트를 0으로 지정한 후 FIFO 리스트의 맨 마지막으로 피드백시켜 다음 순서를 기다리게 한다.

24. Section 093

페이징 기법에서 외부 단편화는 발생하지 않지만 내부 단편화는 발생할 수 있다.

25. Section 093

세그먼트는 사용자의 메모리 보는 관점을 지원하는 메모리 관리 방식이다.

26. Section 093

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.

27. Section 095

- 구역성(Locality) : 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다는 이론
- 단편화(Fragmentation) : 분할된 주기억장치에 프로그램을 할당하고 반납하는 과정을 반복하면서 사용되지 않고 남는 기억장치의 빈 공간 조각
- 세그먼트(Segment) : 프로그램을 가변적인 크기로 나눈 단위

29. Section 094, 095

- 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 집합
- PFF(Page Fault Frequency) : 페이지 부재가 일어나는 횟수

30. Section 094

- FIFO(First In First Out) : 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법
- LFU(Least Frequently Used) : 사용 빈도가 가장 적은 페이지를 교체하는 기법

31. Section 094

- LRU(Least Recently Used) : 최근에 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

32. Section 094

FIFO 알고리즘에서는 페이지가 활동적으로 사용되는데도 불구하고 가장 먼저 들어왔다고 해서 교체되기 때문에 페이지 프레임을 더 많이 할당하여도 페이지 부재율이 증가하는 모순 현상이 나타난다.

33. Section 096

①번은 SSTF, ②번은 SCAN, ④번은 FCFS 기법에 대한 설명이다.

34. Section 094

- ①번은 LFU, ③번은 LRU, ④번은 FIFO에 대한 설명이다.
- 클록(Clock) 알고리즘 : 주기억장치에 적재된 페이지들을 환형 리스트로 보고 이 페이지들을 따라 움직이는 포인터를 시계 방향으로 이동시키면서 교체될 페이지를 선정하는 알고리즘

35. Section 094

- 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 사이에 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

36. Section 095

Semaphore는 병행 프로세스의 동기화를 구현하기 위해 사용된다.

37. Section 094

- FIFO(First In First Out) : FIFO는 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법으로, 이해하기 쉽고 프로그래밍 및 설계가 간단함
- LFU(Least Frequently Used) : 사용 빈도가 가장 적은 페이지를 교체하는 기법
- LRU(Least Recently Used) : 최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법으로, 각 페이지마다 계수기나 스택을 두어 현 시점에서 가장 오랫동안 사용하지 않은, 즉 가장 오래 전에 사용된 페이지를 교체함

38. Section 094

- 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(Dead Lock) : 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

39. Section 095

- 구역성(Locality) : 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다는 이론
- 단편화(Fragementation) : 분할된 주기억장치에 프로그램을 할당하고 반납하는 과정을 반복하면서 사용되지 않고 남는 기억장치의 빈 공간 조각
- 워킹 셋(Working Set) : 실행중인 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합

40. Section 095

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.

41. Section 095

스래싱은 프로세스의 처리 시간보다 페이지 교체 시간이 더 많아지는 현상으로, 다중 프로그래밍의 정도가 높아지면 어느 시점까지는 CPU의 이용율이 높아지고, 스래싱의 발생 빈도가 낮아지지만 어느 시점을 넘어서면 스래싱의 발생 빈도가 높아져 CPU의 이용율이 낮아진다. 그러므로 다중 프로그래밍의 정도가 높다고 무조건 좋은 것은 아니다.

42. Section 095

스래싱이 발생하면 유효 메모리 접근 시간이 증가하게 된다.

43. Section 095

- 시간 구역성 : 루프(Loop), 스택(Stack), 부 프로그램(Sub-routine), Counting(1씩 증감) 등
- 공간 구역성 : 배열 순회(Array Traversal), 순차적 코드의 실행, 프로그래머들이 관련된 변수를 서로 근처에 선언하여 할당되는 기억장소 등

44. Section 095

- 페이지의 크기가 클 경우 적은 수의 페이지가 존재하게 되어 작은 페이지 테이블 공간을 필요로 하고, 참조되는 정보와 무관한 많은 양의 정보가 주기억장치에 남게 되며, 전체적인 입·출력 효율이 증가된다.
- 페이지 크기가 작을 경우 많은 수의 페이지가 존재하여 큰 페이지 테이블 공간을 필요로 하고 우수한 Working Set을 가질 수 있다.

45. Section 095

워킹 셋(Working Set)

- 실행중인 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.

46. Section 096

- SSTF는 탐색 거리(Seek Distance)가 가장 짧은 트랙에 대한 요청을 먼저 서비스하는 기법으로, 헤드에서 먼 곳의 요청은 기아상태(Starvation)를 일으킬 수 있다.
- ②번은 SCAN, ④번은 C-SCAN에 대한 설명이다.

47. Section 096

- FCFS : 가장 간단한 스케줄링으로, 디스크 대기 큐에 들어온 순서대로 서비스하는 기법
- SSTF : 탐색 거리가 가장 짧은 요청을 먼저 서비스하는 기법
- 에센바흐 : 부하가 매우 큰 항공 예약 시스템을 위해 개발된 것으로, 탐색 시간과 회전 지연 시간을 최적화하기 위한 최초의 기법

48. Section 096

- 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 처리함

49. Section 096

입 · 출력 채널이 복잡하면 채널을 추가하거나 그 채널에 부착된 장치를 다른 채널로 옮겨 병목 현상을 제거할 수 있다.

50. Section 093

Backing Store는 보조기억장치로, 여기서는 가상기억장치를 의미한다.

51. Section 096

- FCFS : 디스크 큐에 들어온 요청대로 헤드를 옮겨가며 서비스하는 기법
- 이동 순서는 50 → 100 → 180 → 40 → 120 → 0 → 130 → 70 → 80 → 150 → 200이다.
- 이동 거리는 50 + 80 + 140 + 80 + 120 + 130 + 60 + 10 + 70 + 50 = 790이다.

52. Section 096

②번은 FCFS에 대한 설명이다.

53. Section 096

SCAN

- 현재 헤드의 위치에서 진행 방향이 결정되면 탐색 거리가 짧은 순서에 따라 그 방향의 모든 요청을 서비스하고, 끝까지 이동한 후 역방향의 요청 사항을 서비스하는 기법이므로, 이동 순서는 50 → 40 → 0 → 70 → 80 → 100 → 120 → 130 → 150 → 180 → 200이다.
- 이동 거리는 10 + 40 + 70 + 10 + 20 + 20 + 10 + 20 + 30 + 20이므로 250이다.

54. Section 096

C-SCAN

- 항상 바깥쪽에서 안쪽으로 움직이면서 가장 짧은 탐색 거리를 갖는 요청을 서비스하는 기법이다.
- 트랙의 진행 방향이 주어지지 않았으므로 모든 방향으로 수행해 보면 다음과 같다.

① 0이 안쪽일 경우 : 50 → 40 → 0 → 200 → 180 → 150 → 130 → 120 → 100 → 80 → 70 순으로 총 이동 거리는 380이다.
② 200이 안쪽일 경우 : 50 → 70 → 80 → 100 → 120 → 130 → 150 → 180 → 200 → 0 → 40 순으로 총 이동 거리는 390이다.

55. Section 090

최초 적합(First Fit)은 프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치시키는 방법으로, 10K가 배치되는 영역은 2번이다.

56. Section 095

워킹 셋(Working Set)은 프로세스가 일정 시간(t-w 시간부터 t까지) 동안 참조하는 페이지들의 집합을 의미하는 것으로, 참조된 페이지가 {2, 3, 5, 6, 7}이므로 워킹 셋은 {2, 3, 5, 6, 7}이 된다.

57. Section 096

LRU는 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법으로, 두 번째 프레임의 2가 3으로 교체된다. 그러므로 7, 3, 1 순으로 표시된다.

58. Section 095

페이지 크기가 클 경우 페이지의 수가 줄어들기 때문에 페이지 테이블의 크기는 작아진다.

① 페이지 크기가 작을 경우 – 참조되는 정보와는 무관한 정보가 페이지 크기가 큰 경우보다 더 적게 주기억장치에 적재 될 수 있다.
② 페이지 크기가 작을 경우 – 마지막 페이지의 내부 단편화는 줄어든다.
③ 페이지 크기가 클 경우 – 마지막 페이지의 내부 단편화는 늘어난다.

59. Section 096

디스크 스케줄링의 목적에는 처리량 최대화, 응답 시간의 최소화, 응답 시간 편차의 최소화가 있다.

4장 정답 및 해설 — 정보 관리

1. ③ 2. ④ 3. ① 4. ④ 5. ④ 6. ④ 7. ③ 8. ③ 9. ③ 10. ① 11. ② 12. ④ 13. ③ 14. ③ 15. ③
16. ② 17. ④ 18. ④ 19. ③ 20. ③ 21. ④ 22. ③ 23. ③ 24. ③ 25. ④ 26. ① 27. ④ 28. ② 29. ④ 30. ④
31. ②

1. Section 097

- 파일 시스템 : 파일의 저장, 액세스, 공유, 보호 등 보조기억장치에서의 파일을 총괄하는 파일 관리 기술
- 디렉터리(Directory) : 파일 시스템 내부에 있는 것으로 효율적인 파일 사용을 위해 파일에 대한 여러 정보를 가지고 있는 특수한 형태의 파일
- 장치 상태 테이블(Device Status Table) : 시스템에 있는 장치의 상태를 나타내는 테이블

2. Section 097

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 097

- 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 097

Read는 파일 내의 레코드를 단위로 작업하는 것이고, Open, Create, Copy는 파일 단위로 작업하는 것이다.

5. Section 097

- 파일을 복사하는 명령은 Copy이다.
- Destroy는 파일을 디스크에서 삭제하는 명령이다.

6. Section 097

CPU 스케줄링은 운영체제의 핵심 기능이다.

7. Section 100

섹터 단위 할당 기법은 하나의 파일에 속하는 섹터들이 연결 리스트(Linked List)로 구성되어 있다.

8. Section 100

링크를 이용한 기법에서 레코드를 검색할 경우 파일이 속한 레코드를 링크를 이용하여 순차적으로 검색해야 하므로, 탐색 시간이 오래 걸리고 직접 접근이 불가능하다.

9. Section 100

- 디스크 할당 기법에는 연속 할당 기법과 불연속 할당 기법이 있다.
- 불연속 할당 기법에는 섹터 단위 할당 기법과 블록 단위 할당 기법이 있으며, 블록 단위 할당 기법에는 블록 체인 기법, 색인 블록 체인 기법, 블록 지향 파일 사상 기법이 있다.

10. Section 097

Update는 데이터를 갱신하는 것이고, 데이터 삽입은 Insert이다.

11. Section 098

- **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

12. Section 098

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.

13. Section 098

색인 순차 파일은 인덱스를 이용하여 참조하기 때문에 직접 참조하는 직접 파일보다 접근 시간이 느리다.

14. Section 098

- **순차 파일** : 레코드를 논리적인 처리 순서에 따라 연속된 물리적 저장공간에 기록하는 방법
- **인덱스된 순차 파일** : 순차 파일과 직접 파일에서 지원하는 편성 방법이 결합된 형태
- **분할 파일** : 하나의 파일을 여러 개의 파일로 분할하여 저장하는 형태

15. Section 098

①번과 ②번은 색인 순차 파일(Indexed Seqential File), ④번은 직접 파일(Direct File)에 대한 설명이다.

16. Section 098

- 직접 접근은 해싱 함수를 이용하여 주소를 계산해야 하므로 가변 길이 레코드에서는 사용하기 어렵다.
- **가변 길이 레코드** : 레코드마다 길이가 다른 것을 의미

17. Section 103

- 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.

18. Section 101

파일 보호 기법에는 파일 이름 부여(파일의 명명), 암호, 접근 제어가 있다.

19. Section 101

- **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 101

영역 3의 프로세스는 파일 2에 대해 실행이 가능, 파일 3에 대해 읽기/쓰기가 가능하다.

21. Section 101

전역 테이블은 크기 때문에 일반적으로 가상기억장치에 보관한다.

22. Section 101

파일 3 = { (영역2, {W}), (영역3, {R, W}) }

23. Section 102

보안과 신뢰도는 상관관계가 적다.

24. Section 103

- 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

25. Section 103

공용키 시스템에서 암호화키는 공개되고, 해독키는 보안되어야 한다.

26. Section 103

대칭키 암호화 기법으로 대표적인 것은 DES, 비대칭키 암호화 기법으로 대표적인 것은 RSA이다.

27. Section 103

- 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

28. Section 102

백업(Backup)

원본 데이터의 손실을 대비하여 중요한 데이터를 외부 저장장치에 하나 더 만들어 두는 기능

29. Section 097

파일 시스템은 파일의 저장, 액세스, 공유, 보호 등 보조기억장치에서의 파일을 총괄하는 파일 관리 기술로 Interrupt에 자동으로 대처하는 능력은 없다.

30. Section 097

파일 디스크립터는 시스템을 운영하는 운영체제의 종류에 따라 서로 다른 자료 구조를 갖는다.

31. Section 098

순차 파일은 레코드를 삽입하거나 삭제할 때 파일 전체를 복사한 후 수행해야 한다.

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. ④ 26. ④ 27. ① 28. ④ 29. ④ 30. ②
31. ④ 32. ③ 33. ③ 34. ④ 35. ② 36. ④ 37. ③ 38. ② 39. ② 40. ④ 41. ②

1. Section 104

분리 실행 처리기 구조

주/종 처리기의 비대칭성을 보완하여 각 프로세서가 독자적인 운영체제를 가지고 있도록 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 104

크로스바 교환 행렬은 시분할 및 공유 버스 구조에서 버스의 수를 기억장치 수만큼 증가시켜 연결한 방식으로, 장치의 연결이 복잡하다.

3. Section 105

다중 처리기 운영체제의 구조에는 주/종 처리기, 분리 실행 처리

기, 대칭적 처리기가 있다.

4. Section 106

분산 시스템은 통신선을 이용하여 자원을 공유할 수 있다.

5. Section 106

- 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 보안에 관련된 내용은 분산 시스템의 단점에 해당된다.

6. Section 107

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 106

분산된 시스템에서 프로세서 간의 통신은 필수적이다.

8. Section 107

분산 처리 시스템의 위상에 따른 분류에는 완전 연결형, 부분 연결형, 트리형, 스타형, 다중 접근 버스형 등이 있다.

9. Section 106

신뢰성 : 시스템 안의 한 사이트가 고장나더라도 다른 사이트들이 정상적으로 작동하는지의 여부

10. Section 107

기본 비용은 초기의 설치 비용을 의미하는 것으로, 완전 연결은 모든 사이트를 직접 연결해야 하므로 기본 비용이 많이 든다.

11. Section 107

- **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 107

하나의 상위(부모) 사이트와 연결된 형제 사이트 간에는 상위(부모) 사이트를 통해서만 통신할 수 있다.

13. Section 107

스타형 연결 구조는 모든 사이트가 하나의 중앙 사이트에 직접 연결되어 있고, 그 외의 다른 사이트와는 연결되어 있지 않은 구조이다.

14. Section 107

링형 연결은 목적 사이트에 데이터를 전달하기 위해서 링형 구조에 따라 순환해야 하므로 통신 시간이 많이 걸릴 수 있다.

15. Section 107

다중 접근 버스 연결형은 시스템 내의 모든 사이트들이 공유 버스에 연결된 구조로, 한 사이트의 고장은 다른 사이트의 작동에 영향을 주지 않는다.

16. Section 107

광대역 통신망은 사이트 간의 거리가 멀기 때문에 에러 발생률이 높다.

17. Section 107

분산 운영체제에서 제공하는 이주 방법에는 데이터 이주, 연산 이주, 프로세스 이주가 있다.

18. Section 106

- 분산 파일 체제를 설계할 때 중요한 항목은 투명성이다.
- **투명성** : 분산 처리 운영체제에서 구체적인 시스템 환경을 사용자가 알 수 없도록 하며, 또한 사용자들로 하여금 이에 대한 정보가 없이도 원하는 작업을 수행할 수 있도록 지원하는 개념

19. Section 106

- 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 104

- 시분할 및 공유 버스 : 프로세서, 주변장치, 기억장치 등의 각종 장치들 간을 버스라는 단일 경로로 연결한 방식
- 크로스바 교환 행렬 : 시분할 및 공유 버스 방식에서 버스의 수를 기억장치 수만큼 증가시켜 연결한 방식
- 다중 포트 기억장치 : 시분할 및 공유 버스 방식과 크로스바 교환 행렬 방식을 혼합한 형태의 방식

21. Section 104

$2^n = 256$이므로 n = 8이 된다.

22. Section 105

다중 처리 시스템은 전체 시스템이 독립된 여러 운영체제나 하나의 운영체제에 의해서 관리된다.

23. Section 107

서버(Server)란 서비스를 제공하는 시스템을 의미한다.

24. Section 107

'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 106

분산 처리 시스템은 투명성에 의해 하드웨어나 소프트웨어와 같은 자원의 물리적 위치를 모르더라도 사용자가 자원에 접근할 수 있다.

26. Section 106

분산 처리 시스템은 독립적인 처리 능력을 가진 컴퓨터 시스템을 통신망으로 연결한 것으로, 시스템 설계가 복잡하다.

27. Section 105

약결합(Loosely Coupled) 시스템은 각 시스템마다 독립적인 운영체제와 기억장치를 가지므로 독립적으로 작동할 수 있다.

28. Section 105, 106

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 105, 106

분산 처리 시스템은 각 구성 요소의 추가나 제거가 용이하다.

30. Section 107

비분산 시스템은 중앙 집중식 시스템을 의미하는 것으로, 분산 응용의 설계는 중앙 집중식 시스템의 설계보다 어렵다.

32. Section 107

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 106

분산 처리 시스템에는 접근 투명성, 병행 투명성, 이주 투명성, 위치 투명성, 복제 투명성, 성능 투명성, 규모 투명성, 고장 투명성 등이 있다.

34. Section 106

분산 네트워크는 운영 조직이 복잡하다.

35. Section 107

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 106

분산 시스템은 다양한 형태의 하드웨어 결함으로 피해를 입을 수 있으며 링크의 결함, 사이트의 결함, 메시지의 분실 등이 가장 일반적인 결함이다.

- 링크 결함 : 두 개의 사이트 간에 연결이 잘못되어 발생하는 결함

- 사이트 결함 : 사이트 자체에서 발생하는 결함
- 메시지의 분실 : 통신 회선을 통해 각 사이트로 메시지가 전달되는 과정에서 여러 이유로 메시지를 잃어버리는 결함

37. Section 106

분산 처리 시스템에서는 하드웨어와 제어뿐만 아니라 자료도 분산의 대상이 된다.

38. Section 105

Master/Slave(주/종) 처리기의 주/종 프로세서의 역할

주 프로세서	• 입·출력과 연산을 담당함 • 운영체제를 수행함
종 프로세서	• 연산만 담당함 • 입·출력 발생 시 주프로세서에게 서비스를 요청함

39. Section 104

공유 기억장치 다중 프로세서 시스템이란 강결합(Tightly Coupled) 시스템을 말한다. 큐브는 약결합(Loosely Coupled) 시스템의 종류이다.

40. Section 106

분산 처리 시스템의 설계 목적

- 자원 공유(Resource Sharing) : 각 시스템이 통신망을 통해 연결되어 있으므로 유용한 자원을 공유하여 사용할 수 있음
- 연산 속도 향상 : 하나의 일을 여러 시스템에 분산시켜 처리하므로 연산 속도가 향상됨
- 신뢰도(Reliability) 향상 : 여러 시스템 중 하나의 시스템에 오류가 발생하더라도 다른 시스템은 계속 일을 처리할 수 있으므로 신뢰도가 향상됨
- 컴퓨터 통신 : 지리적으로 멀리 떨어져 있더라도 통신망을 통해 정보를 교환할 수 있음

41. Section 107

완전 연결 구조는 모든 사이트들을 직접 연결해야 하므로 기본 비용이 많이 들지만 각 사이트가 직접 연결되므로 통신 비용은 적게 든다.

6장 정답 및 해설 — 운영체제의 실제

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

1. Section 108

UNIX 파일 시스템의 디렉터리 구조는 트리 구조이다.

2. Section 108

- UNIX는 주로 네크워크용 시스템에 사용된다.
- Stand Alone 시스템은 단독으로 운영되는 시스템을 의미한다.

3. Section 108

- UNIX에서 명령어 해석기 기능을 수행하는 것은 Shell(쉘)이다.
- 커널은 UNIX의 핵심 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 108

- 쉘(Shell) : 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기
- 유틸리티(Utility) : 일반 사용자가 작성한 응용 프로그램을 처리하는 데 사용함

5. Section 109

- UNIX 파일 시스템에서 각 파일에 대한 정보(소유자, 크기, 생성 시간 등)를 저장하고 있는 것은 I-node이다.

- 데이터 블록에는 디렉터리별로 디렉터리 엔트리와 실제 파일에 대한 데이터가 저장되어 있다.

6. Section 108

- **커널(Kernel)** : UNIX의 가장 핵심적인 부분으로, 하드웨어를 보호하고 프로그램과 하드웨어 간의 인터페이스 역할을 담당
- **스케줄러(Scheduler)** : 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업을 하는 것

7. Section 108

- pipe는 프로세스 간에 단방향 통신 채널을 설정하여 프로세스 간에 통신이 가능하도록 하는 명령으로, 전송한 데이터는 FIFO 방식으로 상대에게 전달된다.
- FIFO 방식은 먼저 들어온 데이터를 먼저 전달하는 것이다.

8. Section 108

- 도스에서 COMMAND.COM 파일은 명령을 해독하여 실행하는 명령어 해석기이므로 UNIX에서 쉘(Shell)과 같은 기능을 수행한다.
- **커널(Kernel)** : UNIX의 핵심 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
- **데몬(Demon)** : 백그라운드 상태에서 실행하는 프로그램을 의미

11. Section 109

- chown : 소유자를 변경함
- cat : 파일 내용을 화면에 표시함
- mount : 파일 시스템을 마운팅함

12. Section 108

- **커널의 기능** : 프로세스 관리, 기억장치 관리, 파일 관리, 입·출력 관리, 프로세스 간 통신, 데이터 전송 및 변환, 프로그램과 하드웨어 간의 인터페이스 역할
- 사용자 인터페이스는 쉘(Shell)의 기능이다.

13. Section 109

```
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 옵션
- 파일 정보 구조

①	파일 유형	-
②	소유자 권한	rwx
	그룹 권한	r-x
	전체 권한	r-x
③	링크 수	생략
④	소유자명	aaa
	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 표시함

② **소유자·그룹·전체 권한** : 소유자 권한, 그룹 권한, 전체 권한을 의미하며, 각 권한은 세 개의 문자로 표시함

r(read, 읽기 가능)	파일을 읽을 수만 있다.
w(write, 쓰기 가능)	파일의 내용을 읽고, 수정할 수 있다.
x(excute, 실행 가능)	파일을 실행시킬 수 있다.

※ rwxr-xr-x : 소유자는 읽기, 쓰기, 실행이 가능, 그룹은 읽기, 실행만 가능, 전체 사용자는 읽기, 실행만 가능함

③ **링크 수** : 링크된 수를 표시함

④ 소유자명과 그룹명을 표시한다.

⑤ 파일 크기를 표시한다.

⑥ 최종적으로 변경된 일시를 표시한다.

⑦ 파일명을 표시한다.

14. Section 109

```
- rwxr - x - - x
```

- - : 파일 유형은 일반 파일
- rwx : 소유자는 읽기, 쓰기, 실행이 가능함
- r-x : 그룹은 읽기, 실행이 가능함
- --x : 전체 사용자는 실행만 가능함

15. Section 109

- | : 파이프 라인을 생성함
- [ ] : [a-z]와 같이 [ ] 사이에 문자의 범위를 지정함
- >> : 표준 입·출력장치의 방향을 바꾸어 주는 특수문자로, '>>' 다음에 파일이 존재하면 파일 끝에 데이터를 추가로 저장하고, 파일이 존재하지 않으면 새로운 파일을 만들어 데이터를 저장하라는 의미

16. Section 109

- **부트 블록** : 부팅 시 필요한 코드를 저장하고 있는 블록
- **슈퍼 블록** : 전체 파일 시스템에 대한 정보를 저장하고 있는 블록
- **디렉터리 및 데이터 블록** : 디렉터리별로 디렉터리 엔트리와 실제 파일에 대한 데이터가 저장된 블록

17. Section 109

UNIX는 파일과 디렉터리를 구분하지 않고 동일하게 취급한다.

18. Section 109

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

20. Section 109

getppid는 부모 프로세스의 id를 얻는 명령어이다.

21. Section 109

부모 프로세스와 자식 프로세스는 주기억장치의 공간을 공유하지 않는다.

22. Section 109

- creat : 파일을 생성함
- ls : 현재 디렉터리 내의 파일 목록을 확인함
- mkfs : 파일 시스템을 생성함
- cat : 파일 내용을 화면에 표시함

23. Section 109

- **슈퍼 블록** : 전체 파일 시스템에 대한 정보를 저장하고 있는 블록으로, 사용 가능한 I-node의 개수와 디스크 블록의 개수 등을 알 수 있음
- 부트 스트랩 시에 사용되는 코드를 갖고 있는 것은 부트 블록이다.

24. Section 109

- mkfs : 파일 시스템을 생성함
- kill : 프로세스를 제거함
- mknod : 특수 파일을 생성함

25. Section 110

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

26. Section 110

- **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

27. Section 110

〈시작〉 단추를 클릭할 경우 나타나는 메뉴에는 프로그램, 즐겨찾기, 문서, 설정, 찾기, 도움말, 실행 등이 있다.

29. Section 111

①번은 Autoexec.bat 파일, ②번은 Config.sys 파일, ③번은 Command.com 파일에 대한 설명이다.

30. Section 111

- **실행 가능한 파일을 나타내는 확장자** : COM, BAT, EXE
- OBJ : 언어번역 프로그램으로 번역된 목적 파일
- ASM : 어셈블리어를 이용하여 작성된 파일
- HWP : 흔글 파일

31. Section 111

- 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

32. Section 111

- 파일 삭제 명령은 MS-DOS에서는 DEL, UNIX에서는 rm이다.
- 디렉터리 삭제 명령은 MS-DOS에서는 RD, UNIX에서는 rmdir이다.

33. Section 109

UNIX에서 사용 허가에 대해서는 r(읽기 가능), w(쓰기 가능), x(실행 가능) 중 하나를 지정할 수 있다.

34. Section 110

Windows 98은 선점형 멀티태스킹을 사용하므로, 특정한 응용 프로그램에 문제가 발생되면 해당 프로그램만 종료시킬 수 있다.

35. Section 111

TYPE은 파일의 내용을 표시하는 내부 명령어이다.

36. Section 111

- CHKDSK : 디스크 상태를 점검함
- SYS : 시스템 파일을 복사함
- ATTRIB : 파일의 속성을 변경함

37. Section 108

프로세스 관리, 기억장치 관리, 파일 관리, 입 · 출력 관리, 프로세스 간 통신, 데이터 전송 및 변환 등을 수행하는 것은 커널(Kernel)이다.

38. Section 108

UNIX는 다중 태스킹(Multitasking) 환경을 지원한다.

39. Section 109

- cp : 파일을 복사함
- cat : 파일 내용을 화면에 표시함
- ls : 현재 디렉터리 내의 파일 목록을 확인함

40. Section 109

TYPE은 파일의 내용을 표시하는 도스 명령으로 UNIX에서 cat과 같은 명령어이다.

- cp : 파일을 복사함
- ls : 현재 디렉토리 내의 파일 목록을 확인함
- rm : 파일을 삭제함

41. Section 109

- cat : 파일 내용을 화면에 표시함
- fsck : 파일 시스템을 검사하고 보수함
- cp : 파일을 복사함

42. Section 109

- **슈퍼 블록** : 전체 파일 시스템에 대한 정보를 저장하고 있는 블록
- **데이터 블록** : 디렉터리별로 디렉터리 엔트리와 실제 파일에 대한 데이터가 저장된 블록
- **I-node 블록(Index-node)** : 각 파일이나 디렉터리에 대한 모든 정보를 저장하고 있는 블록

43. Section 109

- mkfs : 파일 시스템 생성
- type : 파일의 내용을 표시하는 명령어로, DOS 명령어임
- chmod : 파일의 보호 모드를 설정하여 파일의 사용 허가 지정

44. Section 109

- mkdir : 디렉터리 생성
- mv : 파일 이동 또는 이름 변경

4과목 소프트웨어 공학

1장 정답 및 해설 — 소프트웨어 공학의 개요

1. ① 2. ② 3. ④ 4. ① 5. ④ 6. ① 7. ④ 8. ② 9. ③ 10. ③ 11. ④ 12. ③ 13. ② 14. ③ 15. ③
16. ④ 17. ④ 18. ① 19. ① 20. ① 21. ④ 22. ① 23. ③ 24. ① 25. ② 26. ④ 27. ① 28. ③ 29. ① 30. ④
31. ④ 32. ③ 33. ① 34. ②

1. Section 112

소프트웨어는 유형의 매체에 저장되지만 개념적이고 무형적이므로 사용에 의해 마모되거나 소멸되지 않는 비마모성의 특징이 있다.

2. Section 112

소프트웨어는 변경 요구가 발생하면 언제라도 유지보수할 수 있지만 비용이 많이 들고, 하드웨어 유지보수보다 어렵다.

3. Section 112

소프트웨어 제품의 성능 평가 기준 : 처리 능력, 반환시간, 반응시간, 신뢰도

4. Section 112

사용 분야에 따라 분류한 소프트웨어

- 프로그래밍용 : 프로그램을 작성하는 데 사용되는 소프트웨어(에디터, 프로그래밍 언어 및 컴파일러)
- 문서 작성용 : 보고서 작성이나 형식을 만들기 위한 소프트웨어
- 통신용 : 인터넷이나 PC 통신을 지원하기 위한 소프트웨어
- 분산 처리용 : 분산 처리 시스템에서 자료가 발생한 지역에서 자료를 처리하도록 지원하는 소프트웨어
- 멀티미디어용 : AUDIO나 VIDEO적인 내용을 수행할 수 있도록 지원하는 소프트웨어
- 소프트웨어 개발용 : 소프트웨어 분석과 설계, 코딩 등 생명 주기의 각 단계에서 생산되는 산출물을 자동으로 생성해 주는 자동화 도구(CASE)
- 인공지능용 : 전문가 시스템, 인공지능 시스템 등

※ 시스템 소프트웨어는 기능에 의한 분류에 해당한다.

6. Section 112

개발 과정의 성격에 따라 분류한 소프트웨어

- 프로토타입(Prototype) : 사용자의 요구사항을 정확히 분석하고 쉽게 이해할 수 있도록 지원하는 견본품으로, 시제품이라고도 함
- 프로젝트(Project) 산출물 : 아직 상품화되지 않은 연구 과정에서 생산된 소프트웨어
- 패키지(Package) : 소프트웨어 개발이나 업무 처리를 지원하기 위해서 개발된 상품형 소프트웨어로 훈글 2005, MS-OFFICE 등이 있음

9. Section 113

소프트웨어 공학은 관리적인 측면과 기술적인 측면 모두를 중요시하는 학문이다.

10. Section 113

①번은 Bauer의 정의, ②번은 Boehm의 정의, ④번은 IEEE의 소프트웨어 공학 표준 용어사전의 정의이다.

11. Section 112

소프트웨어의 수요가 점점 증가하고 있는 상황에서 소프트웨어 개발 비용의 증대, 소프트웨어 개발 기술자 부족, 프로젝트 관리 기술의 부재, 프로그래밍에만 치중하는 행위 등의 원인으로 인해 소프트웨어의 위기가 발생했다고 볼 수 있다.

12. Section 112

컴퓨터 보급의 증가, 컴퓨터 기술 발전 등으로 다양하고 복잡한 소프트웨어의 수요가 증가하고 있는 반면, 이에 대한 공급(생산)이 제대로 이루어지지 않고 있다.

13. Section 112

소프트웨어의 특징

- 상품성 : 개발된 소프트웨어는 상품화되어 판매됨
- 견고성 : 일부 수정으로 소프트웨어 전체에 영향을 줄 수 있음
- 복잡성 : 개발 과정이 복잡하고, 비표준화되어 이해와 관리가 어려움
- 순응성 : 사용자의 요구나 환경 변화에 적절히 변경할 수 있음
- 비가시성 : 소프트웨어의 구조가 외관으로 나타나지 않고, 코드 속에 숨어 있음
- 비제조성 : 하드웨어처럼 제작되는 것이 아니라 논리적인 절차에 맞게 개발됨
- 비마모성 : 사용에 의해 마모되거나 소멸되지 않음
- 비과학성 : 소프트웨어 개발 자체는 수학적이거나 과학적인 것이 아니라 조직, 인력, 시간, 비용, 절차 등이 중심이 됨

14. Section 116

- 프로토타입(Prototype) 모델 : 사용자의 요구사항을 정확히 파악하기 위해 실제 개발될 소프트웨어에 대한 견본품을 만들어 최종 결과물을 예측하는 모형
- 폭포수(Waterfall) 모델 : 폭포에서 한번 떨어진 물은 거슬러 올라갈 수 없듯이 소프트웨어 개발도 각 단계를 확실히 매듭짓고 그 결과를 철저하게 검토하여 승인 과정을 거친 후에 다음 단계를 진행하며 이전 단계로 넘어갈 수 없는 방식
- RAD 모델 : 소프트웨어의 구성 요소를 사용하여 매우 빠르게 선형 순차적 모델을 적용시킴으로써 빠른 개발 주기를 가지는 점진적 소프트웨어 개발 방식

15. Section 113

계층화 기술의 3대 구성 요소 : 도구(Tool), 방법(Method), 절차(Process)

16. Section 115

나선형 모형을 점진적 모형이라고도 한다. 나선형 모형은 나선을 따라 돌듯이 여러 번의 소프트웨어 개발 과정을 거쳐 점진적으로(프로토타입을 지속적으로 발전시켜) 완벽한 최종 소프트웨어를 개발한다.

17. Section 115

소프트웨어 공학 패러다임 : 폭포수 모형(Waterfall Model), 프로토타입 모형(Prototype Model), 나선형 모형(점증적 모형, Spiral Model), 4세대 기법(4GT)

18. Section 115

- 폭포수 모형(Waterfall Model) : 폭포에서 한번 떨어진 물은 거슬러 올라갈 수 없듯이, 소프트웨어 개발도 각 단계를 확실히 매듭짓고 그 결과를 철저하게 검토하여 승인 과정을 거친 후에 다음 단계를 진행하며 이전 단계로 넘어갈 수 없는 방식으로 가장 오래, 가장 폭넓게 사용되고 있음
- 프로토타입 모형(Prototype Model) : 사용자의 요구사항을 정확히 파악하기 위해 실제 개발될 소프트웨어에 대한 견본품(Prototype)을 만들어 최종 결과물을 예측하는 모형
- 나선형 모형(Spiral Model, 점증적 모형) : 폭포수 모형과 프로토타입 모형의 장점에 위험 분석 기능을 추가한 모형
- 4세대 기법(4GT) : 사용자와 개발자가 쉽게 접근하고 사용할 수 있는 4세대 언어(4th Generation Language)를 이용하여 개발자가 조사한 요구사항 명세서로부터 원시 코드를 자동 생성할 수 있게 해주는 모형

19. Section 115

② 개발 중에도 고객의 요구사항에 맞게 수정 작업을 하려면 폭포수 모형이 아니라 개발 중간에 수정이 가능한 프로토타입 모형을 선택해야 한다.

③ 위험 분석을 통해 점증적으로 시스템을 개발하려면 폭포수 모형이 아니라 나선형 모형을 선택해야 한다.

④ 응용 분야가 단순하고 설치 시점에 제품 설명서가 요구되었을 경우에는 나선형 모형이 아니라 폭포수 모형을 선택해야 한다.

20. Section 113

소프트웨어 공학의 여러 정의

- IEEE의 소프트웨어 공학 표준 용어사전 : 소프트웨어의 개발, 운용, 유지보수, 폐기 처분에 대한 체계적인 접근 방안
- Fairley : 지정된 비용과 기간 내에 소프트웨어를 체계적으로 생산하고 유지보수 하는 데 관련된 기술적이고 관리적인 원리
- Boehm : 과학적인 지식을 소프트웨어 설계와 제작에 응용하는 것이며 이를 개발, 운용, 유지보수하는 데 필요한 문서 작성 과정
- 소프트웨어 공학은 제품을 단지 생산하는 것이 아니라 가장 경제적인 방법으로 양질의 제품을 생산하는 것이다.
- 소프트웨어 공학은 안정적이며 효율적으로 작동하는 소프트웨어를 생산하고, 유지보수 활동을 체계적이고 경제적으로 수행하기 위해 계층화 기술을 사용한다.
- 소프트웨어 공학은 소프트웨어의 제작부터 운영까지 생산성을

높이기 위해 기술적, 인간적인 요소에 대한 방법론을 제공한다.

21. Section 113

공학적으로 잘 작성된 소프트웨어

- 사용자가 요구하는 대로 동작해야 한다.
- 하드웨어 자원을 효율적으로 이용할 수 있어야 한다.
- 일정 시간 내에 주어진 조건하에서 원하는 기능을 실행할 수 있어야 한다.
- 애매모호함이 없이 처리 절차에 맞게 수행되어 정확하게 결과가 산출되어야 한다.
- 소프트웨어의 개발, 유지보수 등이 초기 예상한 비용 이내에서 수행되어야 한다.
- 적당한 사용자 인터페이스 제공으로 사용하기가 편리해야 한다.
- 유지보수가 용이해야 한다.
- 잠재적인 에러가 가능한한 적어야 한다.
- 신뢰성 및 효율성이 높다.
- 소프트웨어의 사용법, 구조의 설명, 성능, 기능 등이 이해하기 쉬워야 한다.

22. Section 115

- 프로토타입(Prototype) : 사용자의 요구사항을 정확히 분석하고 쉽게 이해할 수 있도록 지원하는 견본품
- 프로토타이핑(Prototyping) : 프로토타입 모형을 통해 요구 분석을 효과적으로 하면서 명세서를 산출하는 작업

23. Section 115

프로토타입 모형(Prototype Model, 원형 모형)은 사용자의 요구사항을 정확히 파악하기 위해 실제 개발될 소프트웨어에 대한 견본(시제)품(Prototype)을 만들어 최종 결과물을 예측하는 모형이다.

24. Section 115

프로토타입 모형(Prototype Model) 개발 과정

요구사항 수집 → 신속한(빠른) 설계 → 프로토타입 작성(구축) → 프로토타입에 대한 사용자 평가 → 프로토타입의 정제(조정) → 구현

25. Section 115

- 폭포수 모형(Waterfall Model) : 소프트웨어 개발의 각 단계를 확실히 매듭짓고 그 결과를 철저하게 검토하여 승인 과정을 거친 후에 다음 단계를 진행하는 방식으로 가장 오래, 가장 폭넓게 사용되고 있다.
- 4세대 기법(4GT) : 사용자와 개발자가 쉽게 접근하고 사용할 수 있는 4세대(4th Generation Language) 언어를 사용하는 모형

26. Section 116

나선형 모형(Spiral Model)은 폭포수 모형과 프로토타입 모형의 장점에 위험 분석 기능을 추가한 모형으로, 나선을 따라 돌듯이 여러 번의 소프트웨어 개발 과정을 거쳐 점진적으로(프로토타입을 지속적으로 발전시켜) 완벽한 최종 소프트웨어를 개발하는 것이다. 소프트웨어 개발 과정의 앞 단계가 끝나야만 다음 단계로 넘어갈 수 있는 선형 순차적 모형은 폭포수 모형이다.

27. Section 114

- 관리자는 실제 소프트웨어 제작과는 직접적인 관계가 없고 프로젝트 작업에 필요한 비용과 투입 인원, 개발 기간을 결정하는 사람으로, 소프트웨어 개발 계획과 요구사항 분석 단계, 즉 정의 단계에 가장 많이 참여한다.
- 설계 단계 : 소프트웨어의 구조, 알고리즘, 자료 구조 등을 작성하는 단계
- 구현 단계 : 설계 단계에서 작성된 문서를 기초로 하여 코딩하고 번역하는 단계
- 테스트 단계 : 구현된 소프트웨어에 내재되어 있는 오류를 찾아주는 단계

28. Section 114

- 개발 계획 단계 : 소프트웨어 개발에 사용될 자원과 비용을 측정하는 단계
- 분석 단계 : 사용자가 요구한 문제를 정확히 분석하는 단계
- 설계 단계 : 소프트웨어의 구조, 알고리즘, 자료 구조 등을 작성하는 단계로 에러가 가장 많이 발생함
- 유지보수 단계 : 소프트웨어를 직접 운용하고, 여러 환경의 변화에 따른 소프트웨어의 적응 및 유지를 위한 단계

29. Section 113

소프트웨어 공학은 소프트웨어의 위기를 극복하기 위한 방안으로 연구된 학문이며 여러 가지 방법론과 도구, 관리 기법들을 통하여 소프트웨어의 품질과 생산성 향상시킨다. 그러므로 예술성은 소프트웨어 공학과 관련이 없다.

31. Section 116

나선형 모형(Spiral Model, 점진적 모형)의 활동 과정은 계획 및 정의(Planning) → 위험 분석(Risk Analysis) → 공학적 개발(Engineering) → 고객 평가(Customer Evaluation) 순이다.

32. Section 113

어떠한 용도로도 타 기업의 시스템에 몰래 접속하여 새로운 소프트웨어 개발에 관한 정보를 획득하는 것은 올바른 자세가 아니다.

33. Section 114

소프트웨어 실패율(고장률, 오류율)을 나타내는 그림에서 '?'이 지적하는 것은 소프트웨어 변경이다. 문제의 그림은 소프트웨어에 대한 사용 시간과 실패율에 대한 관계를 나타내는데, 소프트웨어를 변경함으로써 예상치 못한 오류(실패)가 발생할 수 있음을 확인할 수 있다.

2장 정답 및 해설 — 소프트웨어 프로젝트 관리

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

1. Section 118

- **프로젝트 계획 수립** : 프로젝트가 수행되기 전에 소프트웨어 개발 영역 결정, 필요한 자원, 비용, 일정 등을 예측하여 수립하는 작업
- ①번은 프로젝트 관리에 대한 개념이다.

2. Section 118

소프트웨어 영역 결정 사항에 포함되는 사항 : 처리될 데이터와 소프트웨어에 대한 기능, 성능, 제약 조건, 인터페이스 및 신뢰도

3. Section 118

프로젝트 계획서에 포함되는 주요 내용 : 시스템의 목적 및 개발 지침, 전산화 규모 및 조건, 기본 일정, 프로젝트 예산, 프로젝트 팀의 구성, 프로젝트 관리 기준, 프로젝트 완료 조건 등

5. Section 127

'개발 비용'은 소프트웨어 형상 관리 대상에 포함되지 않는다.

6. Section 124

- 신뢰성(Reliability) : 정확하고 일관된 결과를 얻기 위해 요구된 기능을 오류 없이 수행하는 정도
- 효율성(Efficiency) : 요구되는 기능을 수행하기 위해 필요한 자원의 소요 정도
- 무결성(Integrity) : 허용되지 않는 사용이나 자료의 변경을 제어하는 정도

7. Section 120

- **상향식 비용 산정 기법** : 프로젝트의 세부적인 작업 단위별로 비용을 예측한 후 집계하여 전체 비용을 산정하는 방법
- **수학적 산정 기법** : 상향식 비용 산정 기법으로 경험적 추정 모형, 실험적 추정 모형이라고도 하며, 개발 비용 산정의 자동화를 목표로 함
- **LOC 기법** : 소프트웨어 각 기능의 원시 코드 라인 수의 비관치, 낙관치, 기대치를 측정하여 예측치를 구하고 이를 이용하여 비용을 산정하는 기법

8. Section 120, 121

홀스테드(Halstead)의 노력 방정식은 코드 생성 후 피연산자와 연산자 수를 더하여 복잡도를 측정하는 데 사용되는 평가 척도 중의 하나이다.

9. Section 120

전문가의 감정에 의한 기법은 전문가의 주관이 많이 개입될 수

있다.

10. Section 120

- 기능 점수 기법 : 소프트웨어의 기능을 고려하는 FP를 사용하는 기법
- LOC 기법 : 소프트웨어 각 기능의 원시 코드 라인 수의 비관치, 낙관치, 기대치를 측정하여 예측치를 구하고 이를 이용하여 비용을 산정하는 기법
- COCOMO 기법 : 원시 프로그램의 규모(LOC, 원시 코드 라인 수)에 의한 비용 예측 기법

11. Section 121

소프트웨어 개발 유형은 소프트웨어의 복잡도, 원시 프로그램의 규모에 따라 조직형, 반분리형, 내장형으로 분류한다.

12. Section 121

노력 승수 분류

- 제품의 특성 : 요구되는 신뢰도, 데이터베이스 크기, 제품의 복잡도
- 컴퓨터의 특성 : 수행 시간의 제한, 기억장소의 제한, 가상 기계의 안정성, Turn Around Time
- 개발 요원의 특성 : 분석가의 능력, 개발 분야의 경험, 가상 기계의 경험, 프로그래머의 능력, 프로그래밍 언어의 경험
- 프로젝트 특성 : 소프트웨어 도구의 이용, 프로젝트 개발 일정, 최신 프로그래밍 기법의 이용

14. Section 120

- 노력(PM) = LOC/1인당 월평균 생산 라인 수 = 30000/300 = 100
- 개발 기간 = 노력/투입 인원 = 100/5명 = 20개월

15. Section 121

- LOC 기법 : 소프트웨어 각 기능의 원시 코드 라인 수의 비관치, 낙관치, 기대치를 측정하여 예측치를 구하고 이를 이용하여 비용을 산정하는 기법
- 델파이 기법 : 전문가에 의한 주관적인 편견을 보완하기 위해 많은 전문가의 의견을 종합하여 산정하는 기법
- 기능 점수 모형 : 소프트웨어의 기능을 고려하는 FP를 사용하는 기법

16. Section 121

- LOC 기법 : 소프트웨어 각 기능의 원시 코드 라인 수의 비관치, 낙관치, 기대치를 측정하여 예측치를 구하고 이를 이용하여 비용을 산정하는 기법
- 델파이 기법 : 전문가에 의한 주관적인 편견을 보완하기 위해 많은 전문가의 의견을 종합하여 산정하는 기법
- Putnam 모형 : Putnam이 제안한 것으로 소프트웨어 생명 주기의 전 과정 동안에 사용될 노력의 분포를 가정해 주는 모형

17. Section 125

MTBF와 MTTF를 구분하지 않았으므로 MTBF와 MTTF를 동일한 개념, 즉 평균 가동 시간이라 가정하고 푼다. 평균 가동 시간은 (14+16+12)/3이므로 14가 된다.

18. Section 122

프로젝트의 종료 일정은 소프트웨어 공학 그룹에 속한 사람들이 정하는 것이 아니라 개발 의뢰인에 의해 정해진다.

19. Section 122

- CPM : 프로젝트 완성에 필요한 작업을 나열하고 작업에 필요한 소요 기간을 예측하는 데 사용하는 기법
- PERT : 프로젝트에 필요한 전체 작업의 상호 관계를 표시하는 네트워크로, 각 작업별로 비관적인 경우, 낙관적인 경우, 그리고 가장 가능성이 있는 경우로 나누어 각 단계별 종료 시기를 결정하는 방법
- KLOC : 원시 코드 라인 수(LOC)를 1,000라인씩 묶은 단위

20. Section 122

WBS

- 개발 프로젝트를 여러 개의 작은 관리 단위(소작업)로 분할하여 계층적으로 기술한 업무 구조이다.
- 일정 계획의 첫 단계에서 작업을 분할할 때 사용되는 방법이다.
- 계획 관리 단계에서 일정 계획과 인력 계획, 비용 산정의 기준이 된다.
- 프로젝트 진행중에 발생하는 모든 작업을 알 수 있다.
- 제품의 계층 구조 또는 프로세스의 계층 구조로 나타낸다.

21. Section 122

- WBS : 개발 프로젝트를 여러 개의 작은 관리 단위(소작업)로 분

할하여 계층적으로 기술한 업무 구조

- 간트 차트(Gantt Chart) : 프로젝트의 각 작업들이 언제 시작하고 종료되는지에 대한 작업 일정을 막대 도표를 이용하여 표시하는 프로젝트 일정표

22. Section 122

PERT/CPM

- 대단위 계획의 조직적인 추진을 위해 자원의 제약하에 비용을 적게 사용하면서 초단시간 내 계획 완성을 위한 프로젝트 일정 방법이다.
- 프로젝트 개발 기간을 결정하는 임계 경로(CP, Critical Path)를 제공한다.
- 통계적 모델을 적용해서 개별 작업에 대한 가장 근접한 시간 측정의 기준이 된다.
- 각 작업에 대한 시작 시간을 정의하여 작업들 간의 경계 시간을 계산할 수 있게 한다.

23. Section 122

PERT/CPM의 단점

- 작업에 소요되는 정확한 시간을 알 수 없다.
- Project가 대규모일 때 임계 경로 이외의 경로상에 있는 공정들은 작업이 끝났는데도 불구하고 다음 공정이 시작되기 전까지 기다려야 하는 시간이 많아진다.
- 공정 상호간에 별도의 시간 제약 조건 부가가 어렵다.

24. Section 122

간트 차트(Gantt Chart)

- 프로젝트의 각 작업들이 언제 시작하고 종료되는지에 대한 작업 일정을 막대 도표를 이용하여 표시하는 프로젝트 일정표이다.
- 중간 목표 미달성 시 그 이유와 기간을 예측할 수 있게 한다.
- 사용자와의 문제점이나 예산의 초과 지출 등도 관리할 수 있게 한다.
- 자원 배치와 인원 계획에 유용하게 사용된다.
- 다양한 형태로 변경하여 사용할 수 있다.
- 작업 경로는 표시할 수 없으며, 계획의 변화에 대한 적응성이 약하다.

※ 프로젝트 내에 포함된 작업들 간의 관계를 알 수 있는 것은 PERT/CPM이다.

25. Section 123

간트 차트는 막대 도표로 작업 일정을 표시하는 프로젝트 일정표로, 작업 경로를 표시할 수는 없다.

26. Section 123

프로젝트 조직 구성 : 민주주의식 팀(분산형 팀), 책임 프로그래머 팀(Chief Programmer Team, 중앙 집중형 팀), 계층적 팀(혼합형 팀)

27. Section 123

④번은 중앙 집중형, 즉 책임 프로그래머 팀 구성에 대한 설명이다.

28. Section 123

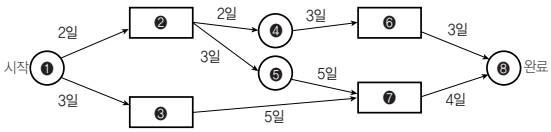
중앙 집중형(책임 프로그래머) 팀 구성은 한 관리자가 의사 결정을 하고, 팀 구성원들은 그 결정을 따르는 구성 방식이므로 의사 결정이 빠르고, 의사 교환 통로를 줄일 수 있다.

29. Section 123

중간 관리층은 고급 프로그래머로 구성하며, 의사 교환은 초급 프로그래머와 중간 관리층으로 분산된다.

30. Section 122

임계경로는 최장 경로를 의미한다. 문제에 제시된 그림을 보고 각 경로에 대한 소요기일을 계산한 후 가장 오래 걸린 기일을 찾으면 된다.



- 경로 1 : ❶→❷→❹→❻→❽
  = 2+2+3+3 = 10일
- 경로 2 : ❶→❷→❺→❼→❽
  = 2+3+5+4=14일
- 경로 3 : ❶→❸→❼→❽ = 3+5+4 =12일

그러므로 임계경로는 경로 2이며, 소요기일은 14일이다.

31. Section 123

민주주의식 팀 구성

- 팀 구성원들이 모두 자신의 의견을 서로에게 전달할 수 있는 팀 구성 방식이다.
- 의사 경로의 수는 n(n−1)/2로 가장 많다(n : 인원수).

33. Section 124

- 품질 관리는 품질 통제와 품질 보증이 유기적으로 연계되어 이루어진다.
- **품질 통제** : 개발, 운영, 유지보수 과정에서 품질을 유지하기 위해 필수적으로 고려해야 하는 기본 프로세스로 소프트웨어의 개발 조직, 운영 조직 및 유지보수 조직 내에서 자체 수행함
- **품질 보증** : 해당 소프트웨어의 신뢰성을 보장해 주기 위해 기본 프로세스를 지원해 주는 프로세스로, 제3자의 입장에서 수행함

34. Section 124

- 용이성(Usability) : 사용에 필요한 노력을 최소화하고 쉽게 사용할 수 있는 정도
- 신뢰성(Reliability) : 정확하고 일관된 결과를 얻기 위해 요구된 기능을 오류 없이 수행하는 정도
- 무결성(Integrity) : 허용되지 않는 사용이나 자료의 변경을 제어하는 정도

35. Section 124

- 사용 용이성(Usability) : 사용에 필요한 노력을 최소화하고 쉽게 사용할 수 있는 정도
- 유연성(Flexibility) : 소프트웨어를 얼마만큼 쉽게 수정할 수 있는가 하는 정도
- 유지보수성(Maintainability) : 변경 및 오류 사항의 교정에 대한 노력을 최소화하는 정도

37. Section 124

정형 기술 검토의 목적

- 검토중인 소프트웨어가 해당 요구사항과 일치하는지를 검증한다.
- 소프트웨어가 미리 정해진 표준에 따라 표현되고 있는지를 확인한다.
- 소프트웨어가 균일한 방식으로 개발되도록 한다.
- 프로젝트를 보다 용이하게 관리하도록 한다.

38. Section 124

- 검열(Inspections, 심사) : 검토 회의(Walkthrough)를 발전시킨 형태로, 소프트웨어 개발 단계에 걸쳐서 산출된 결과물의 품질을 평가하며 이를 개선시키는 데 사용되는 기법
- 정형 기술 검토(Formal Technical Review) : 소프트웨어 기술자들에 의해 수행되는 소프트웨어 품질 보증 활동
- 검증(Verification) : 설계의 각 과정이 올바른지, 프로그램이나 하드웨어에 오류가 있는지 검사하는 활동
- 확인(Validation) : 올바른 제품을 생산할 수 있도록 정의, 분석이 잘 되었는지를 검사하는 활동

39. Section 124

품질 보증을 위한 감시 활동들은 제품이 완성된 후에만 하는 것이 아니라 모든 개발 과정에 필요한 활동이다.

40. Section 126

위험 관리 활동 : 위험 식별, 위험 분석 및 평가, 위험 관리 계획, 위험 감시 및 조치

41. Section 126

위험 요소(Risk Component) : 인력 부족, 비현실적 일정 및 예산, 부정확한 기능의 소프트웨어 개발, 잘못된 인터페이스의 개발, 계속적인 요구 변경, 외부 요소의 빈약, 외부 기능의 빈약, 실시간 성능 문제, 기술적 취약(교육상의 문제)

43. Section 127

형상 관리(SCM, Software Configuration Management)

- 소프트웨어의 변경되는 사항을 관리하기 위해 개발된 일련의 활동이다.
- 소프트웨어 변경의 원인을 알아내고 제어하며 적절히 변경되고 있는지 확인하여 해당 담당자에게 통보하는 작업이다.
- 형상 관리는 소프트웨어 생명 주기 전 과정에 적용되는 활동으로, 유지보수 단계에서 수행된다.

44. Section 125

MTBF와 MTTF를 구분하지 않았으므로 MTBF와 MTTF를 동일한 개념, 즉 평균 가동 시간이라 가정하고 문제를 풀면 평균 가동 시간(MTBF)은 (144 + 178 + 216 + 252)/4 = 197.5시간이 된다.

47. Section 119

소프트웨어 개발의 생산성에 영향을 미치는 요소에는 제품의 복잡도, 시스템의 크기, 개발자의 능력, 팀 의사 전달, 개발 기간 등이 있다.

48. Section 118

유지보수는 개발된 소프트웨어의 품질을 항상 최상의 상태로 유지

하기 위한 것으로, 유지보수 비용은 프로젝트가 완성된 후에 사용되는 비용이다. 그러므로 프로젝트를 시작하기 전에 추정해야 하는 항목과는 거리가 멀다.

49. Section 119

프로젝트 비용과 측정을 위해서는 프로젝트를 상대적으로 잘게 분리하며 예측하는 분해 기술을 이용해야 한다.

50. Section 122

CPM은 간선을 나타내는 화살표의 흐름에 따라 각 작업이 진행되고, 전 작업이 완료된 후 다음 작업을 진행할 수 있으므로 작업의 선후 관계가 명확히 파악되어야 한다.

51. Section 124

- 신뢰성(Reliability) : 정확하고 일관된 결과를 얻기 위해 요구된 기능을 오류 없이 수행하는 정도
- 유지보수성(Maintainability) : 변경 및 오류 사항의 교정에 대한 노력을 최소화하는 정도

52. Section 124

- 신뢰성(Reliability) : 정확하고 일관된 결과를 얻기 위해 요구된 기능을 오류 없이 수행하는 정도
- 사용 용이성(Usability) : 사용에 필요한 노력을 최소화하고 쉽게 사용할 수 있는 정도
- 유연성(Flexibility) : 소프트웨어를 얼마만큼 쉽게 수정할 수 있는가 하는 정도

53. Section 124

검토 회의(Walkthrough)는 소프트웨어 개발의 각 단계에서 개최하는 기술 평가 회의로, 소프트웨어 구성 요소와 같은 작은 단위를 검토하는 것으로, ④번과는 관련이 없다.

3장 정답 및 해설 — 전통적 S/W 개발 방법론

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

1. Section 128

- 프로젝트 계획 단계 : 프로젝트가 수행되기 전에 소프트웨어 개발 영역, 필요한 자원, 비용, 일정 등을 예측하는 작업
- 설계 단계 : 요구사항 분석 단계의 산출물인 요구사항 분석 명세서의 기능이 실현되도록 알고리즘과 그 알고리즘에 의해 처리될 자료 구조를 문서화하는 것
- 유지보수 단계 : 개발된 소프트웨어의 품질을 항상 최상의 상태로 유지하기 위한 것

2. Section 128

- 기능 요구 : 절차, 입 · 출력
- 비기능 요구 : 성능(Performance), 신뢰도(Reliability), 기밀 보안성(Security), 운용 제약성(Operating Constraints), 개발 계획(Development Plan), 개발 비용(Development Cost), 개발 환경(Development Environments), 상충 효과(Trade off)

3. Section 128

요구사항 분석이 어려운 이유

대화 장벽, 시스템의 복잡도, 요구의 변경, 요구 명세화의 어려움

4. Section 128

요구사항 분석 기법

- 자료 흐름 중심 분석 : 자료 흐름도, 자료 사전, 소단위 명세서 등
- 자료 구조 중심 분석 : 워니어-오 분석 기법, 잭슨 분석 기법

5. Section 128

구조적 분석은 하향식 원리를 적용하기 때문에 분석의 중복성을 배제할 수 있다.

6. Section 129

자료 흐름은 실선의 화살표로 표시한다.

7. Section 129

배경도는 기본 시스템 모델로, 전체 소프트웨어 요소를 표시하는 하나의 프로세스와 입 · 출력을 나타내는 화살표로 표현한다.

8. Section 129

- **자료 흐름도** : 요구사항 분석에서 자료의 흐름 및 변환 과정과 기능을 도형 중심으로 기술하는 방법
- **소단위 명세서** : 세분화된 자료 흐름도에서 최하위 단계 버블(프로세스)의 처리 절차를 기술한 것
- **상태 전이도** : 시스템에 어떤 일이 발생할 경우 시스템의 상태와 상태 간의 전이를 모형화하는 것

9. Section 129

자료 흐름도에서 자료 저장소(Data Store)는 삼각형이 아니라 평행선으로 표시한다.

10. Section 129

자료 사전 작성 시 고려 사항

- 이름을 가지고 정의를 쉽게 찾을 수 있어야 하므로, 이름이 중복되어서는 안 된다.
- 갱신하기 쉬워야 하며, 정의하는 방식이 명확해야 한다.

11. Section 135

조건 검사, 루프 검사, 데이터 흐름 검사는 화이트 박스 검사 기법에 해당한다.

12. Section 129

소단위(프로세스) 명세서는 구조적 언어, 의사 결정표(판단표), 의사 결정도 등을 이용하여 기술한다.

13. Section 129

소단위 명세서는 세분화된 자료 흐름도에서 최하위 단계 버블(프로세스)의 처리 절차를 기술한 것이다.

14. Section 131

구조적 설계의 기본 원리 : 모듈화, 추상화, 단계적 정제, 정보 은닉, 프로그램 구조, 자료 구조 등

15. Section 132

모듈

하나 또는 그 이상의 논리적 기능들을 수행하는 컴퓨터 지시어들의 집합으로, 호출 모듈과 피호출 모듈로 구성되어 있으며 내부 자료 속성을 갖는다.

16. Section 132

- **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
- **스탬프(검인) 결합도(Stamp Coupling)** : 모듈 간의 인터페이스로 배열이나 레코드 등의 자료 구조가 전달될 때의 결합도

17. Section 132

모듈의 속성

- **입·출력 요소** : 자료를 받아들이고, 자료를 내보내는 요소
- **기능 요소** : 입력을 출력으로 바꾸는 요소, 즉 모듈의 처리 결과
- **기관 요소** : 기능을 수행하기 위한 절차상의 코드 또는 논리
- **내부 자료 요소** : 모듈 자체의 작업장, 모듈이 스스로 참조하는 자료

18. Section 133

설계 도구

- **계층 구조 표현 도구** : 구조 도표
- **모듈 명세서 작성 시 문서화 도구** : 의사 코드(PDL), N-S 차트, HIPO, 의사 결정표, 의사 결정도 등

19. Section 132

모듈의 독립성 : 소프트웨어를 구성하는 각 모듈의 기능이 독립됨을 의미하는 것

20. Section 132

결합도 정도(강함 〉 약함)

내용 결합도(Content Coupling) 〉 공통 결합도(Common Coupling) 〉 외부 결합도(External Coupling) 〉 제어 결합도(Control Coupling) 〉 스탬프 결합도(Stamp Coupling) 〉 자료 결합도(Data Coupling)
※ 모듈 설계에서 가장 좋은 결합도 상태는 결합도가 약한 것이다.

21. Section 132

응집도(Cohesion)

- 응집도는 정보 은닉 개념을 확장한 것으로 모듈 안의 요소들이 서로 관련되어 있는 정도, 즉 모듈이 독립적인 기능으로 정의되어 있는 정도를 의미한다.
- 독립적인 모듈이 되기 위해서는 각 모듈의 응집도가 강해야 한다.

22. Section 132

① 결합도는 모듈 간에 상호 의존하는 정도를 의미한다.
※①번의 설명은 응집도에 대한 설명이다.
② 결합도가 높으면 시스템을 구현하고 유지보수하는 작업이 어렵다.
④ 자료 결합도는 내용 결합도보다 결합도가 낮다.

25. Section 132

하나의 프로그램을 일정한 간격으로 나누어 모듈을 만들면 각 모듈은 서로 뚜렷한 관계를 갖지 못하므로 우연적 응집도에 해당된다.

26. Section 132

- **순차적 응집도(Sequential Cohesion)** : 모듈 내의 하나의 활동으로부터 나온 출력 데이터를 그 다음 활동의 입력 데이터로 사용할 경우의 응집도
- **교환적 응집도(Communication Cohesion)** : 동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 구성 요소들이 모였을 경우의 응집도로, 통신적 응집도라고도 함
- **논리적 응집도(Logical Cohesion)** : 유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들로 하나의 모듈이 형성되는 경우의 응집도

27. Section 133

거래 분해 접근법 : 자료 흐름도에서 거래에 해당하는 부분을 찾아내어 이 거래 중심 자료 흐름도를 거래 중심 구조 도표로 변환하는 일련의 단계

28. Section 131, 132

모듈 B, C는 A에 종속하고 모듈 D, E는 A에 간접 종속한다.

29. Section 130

HIPO 사용의 이점

- 문서의 체계화가 가능하다.
- 하향식(Top-Down) 개발이 가능하다.
- 보기 쉽고 알기 쉽다.
- 기능과 자료의 의존 관계를 동시에 표현할 수 있다.
- 변경, 유지보수가 용이하다.

31. Section 130

총괄 도표(개요 도표) : 프로그램을 구성하는 기능을 기술한 것으로 입력, 처리, 출력에 대한 전반적인 정보를 제공하는 도표

32. Section 133

프로그램 설계 언어(PDL, Program Design Language)

영어 단어를 이용하여 구조적 프로그래밍의 제어 구조를 기술하는 것으로, 프로그래밍 언어를 잘 몰라도 사용이 가능하다.

33. Section 133

N-S 차트의 특징

- GOTO나 화살표를 사용하지 않는다.
- 조건이 복합되어 있는 곳의 처리를 시각적으로 명확히 식별하는 데 적합하다.
- 선택과 반복 구조를 시각적으로 표현한다.
- 이해하기 쉽고, 코드 변환이 용이하다.
- 읽기는 쉽지만 작성하기가 어려우며, 임의로 제어를 전이하는 것이 불가능하다.
- 총체적인 구조 표현과 인터페이스를 나타내기는 어렵다.
- 단일 입구와 단일 출구로 표현한다.

34. Section 133

인터페이스 설계

- 소프트웨어와 상호 작용하는 시스템, 사용자 등과 어떻게 통신하는지를 기술하는 과정이다.
- 소프트웨어와 모듈 사이, 소프트웨어와 정보 생산자 · 소비자 사이, 소프트웨어와 사용자 사이의 인터페이스 설계로 나눌 수 있다.
- 변화 요인이 다양한 사용자와의 인터페이스에 중점을 두고 설계한다.
- 사용자 인터페이스는 소프트웨어 외부 설계에 해당된다.

35. Section 135

- 경계값 분석(Boundary Value Analysis) : 입력 자료에만 치중한 동치 분할 기법을 보완하기 위한 기법
- 원인-효과 그래프 검사(Cause-effect graphing testing) : 입력 데이터 간의 관계와 출력에 영향을 미치는 상황을 체계적으로 분석하여 효용성 높은 검사 사례를 선정하여 검사하는 기법
- 동치 분할 검사(Equivalence Partitioning Testing) : 입력 자료에 초점을 맞춰 검사 사례를 만들고 검사하는 방법으로 동등 분할 기법이라고도 함

36. Section 134

구조적 프로그래밍의 특징

- 하향식(Top Down) 접근 기법이다.
- 모듈(Module) 단위의 프로그래밍이다.
- 세 가지 제어 구조(순차, 선택, 반복)만을 사용한다.
- 제어 흐름은 단일 입구와 단일 출구를 가진다.
- 무조건 분기하는 GOTO문은 사용하지 않는다.

37. Section 135

제어 흐름도에서 순환 복잡도 계산 방법

① 영역 수를 계산한다.
② V(G)=E-N+2(E는 화살표 수, N은 노드 수)
∴ V(G)=6-4+2=4개임

38. Section 136

- 개발 단계에 따른 테스트 종류 : 단위 테스트, 통합 테스트, 검증 테스트, 시스템 테스트
- 화이트 박스 테스트는 최소한의 시간과 노력으로 대부분의 오류를 찾아내기 위해 검사 사례(Test Case)를 설계하는 방법 중 하나이다.

39. Section 136

깊이 우선 통합법 또는 넓이 우선 통합법에 따라 스터브를 실제 모듈로 대치하는 것은 하향식 통합 검사이다.

40. Section 135

복잡한 논리는 반드시 필요할 때만 사용하고 가능한 한 배제시킨다.

41. Section 135

테스트에 임하는 원칙

- 개발자는 자신이 작성한 프로그램을 테스트하는 것을 피해야 한다.
- 각 테스트 사례는 기대되는 출력 내용을 포함해야 한다.
- 결함이 추가로 발견될 확률은 이미 발견된 결함수에 정비례한다는 사실을 인지한다.
- 사용상 오류나 기대되지 않은 입력 조건에 관해서도 테스트 사례가 준비되어야 한다.
- 결함이 발견되지 않을 것이라는 묵시적 가정하에서 테스트 계획을 수립하면 안 된다.

42. Section 135

- 화이트 박스 검사(White Box Test) : 모듈의 원시 코드를 오픈시킨 상태에서 원시 코드의 논리적인 모든 경로를 검사하여 검사 사례를 설계하는 방법
- 블랙 박스 검사(Black Box Test) : 제품이 수행할 특정 기능을 알기 위해서 각 기능이 완전히 작동되는 것을 입증하는 검사로서, 기능 검사라고도 함
- 샌드위치 검사(Sandwich Test) : 하위 수준에서는 상향식 통합, 상위 수준에서는 하향식 통합을 사용하여 최적의 검사를 지원하는 방식
- 빅뱅 검사(Big Bang Test) : 단계적으로 통합하는 절차 없이 모든 모듈이 미리 결합되어 프로그램 전체를 검사하는 방법

43. Section 135

화이트 박스 테스트 기법에는 기초 경로 검사, 제어 구조 검사(조건 검사, 루프 검사, 데이터 흐름 검사) 등이 있다. 비교 검사는 블랙 박스 테스트 기법의 한 종류다.

44. Section 129

자료 흐름의 구성 요소

- 프로세스(Process) : 자료를 변환시키는 시스템의 한 부분(처리 과정)으로 처리, 기능, 변환, 버블이라고도 함. 원이나 둥근 사각형으로 표시하고 그 안에 프로세스 이름을 기입함
- 자료 흐름(Flow) : 자료의 이동(흐름)이나 연관관계를 나타내며, 화살표 위에 자료의 이름을 기입함
- 자료 저장소(Data Store) : 시스템에서의 자료 저장소(파일, 데이터베이스)를 나타내며, 도형 안에 자료 저장소 이름을 기입함
- 단말(Terminator) : 시스템과 교신하는 외부 개체로, 입력 데이터가 만들어지고 출력 데이터를 받음(정보의 생산자와 소비자), 이중선 도형 안에 이름을 기입함

45. Section 135

- 블랙 박스 테스트를 이용하여 발견할 수 있는 오류에는 부정확하거나 누락된 기능, 인터페이스 오류, 자료 구조나 외부 데이터베이스 접근에 따른 오류, 행위나 성능 오류, 초기화와 종료 오류 등이 있다.
- 반복 조건을 만족하는데도 루프 내의 문장이 수행되지 않는 경우는 화이트 박스 테스트를 이용하여 발견할 수 있다.

46. Section 136

- **통합 검사** : 단위 검사가 완료된 모듈들을 결합하여 하나의 시스템으로 완성시키는 과정에서의 검사
- **검증 검사** : 소프트웨어가 사용자의 요구사항을 충족시키는가에 중점을 두는 검사
- **시스템 검사** : 개발된 소프트웨어가 해당 컴퓨터 시스템에서 완벽하게 수행되는가를 검사하는 것

47. Section 136

단위 검사

- 코딩이 이루어진 후 소프트웨어 설계의 최소 단위인 모듈에 초점을 맞추어 검사하는 것
- 화이트 박스 테스트 기법을 사용한다.
- 단위 검사에서는 인터페이스, 외부 I/O, 자료 구조, 독립적 기초 경로, 오류 처리 경로, 경계 조건 등을 검사한다.

48. Section 136

빅뱅 검사(Big Bang Test)는 단계적으로 통합하는 절차 없이 모든 모듈이 미리 결합되어 프로그램 전체를 검사하는 방법으로, 비점진적 통합 방식이다.

49. Section 136

하향식 통합 기법은 처음부터 독립된 프로그램 구조를 갖춘다.

50. Section 135

- **기초 경로 검사(Basic Path Testing)** : 대표적인 화이트 박스 테스트 기법으로, 검사 사례 설계자가 절차적 설계의 논리적 복잡성을 측정할 수 있게 해주고 이 측정 결과는 실행 경로의 기초를 정의하는 데 지침으로 사용됨
- **조건 검사(Condition Testing)** : 프로그램 모듈 내에 있는 논리적 조건을 검사하는 검사 사례 설계 기법
- **루프 검사(Loop Testing)** : 프로그램의 반복(Loop) 구조에 초점을 맞춰 실시하는 검사 사례 설계 기법

51. Section 136

알파 검사와 베타 검사는 사용자가 개발된 소프트웨어를 인수하기 위해 사용자 관점에서 테스트하는 기법이다.

52. Section 136

시스템 검사의 종류

- **복구 검사** : 소프트웨어에 여러 가지 결함을 주어 실패하도록 한 후 올바르게 복구되는지를 확인하는 검사
- **보안 검사** : 시스템 내에 설치된 보호 도구가 부적당한 침투로부터 시스템을 보호할 수 있는지를 확인하는 검사
- **강도 검사** : 비정상적인 상황에서 소프트웨어를 실행시키기 위한 검사로 비정상적인 양, 빈도 등의 자원을 요구하는 환경에서 소프트웨어를 실행시킴
- **성능 검사** : 통합된 시스템에서 소프트웨어의 실행 시간을 검사하기 위한 것으로, 검사 단계의 전 과정에 걸쳐 수행됨

53. Section 136

디버깅 접근법에는 맹목적 강요, 역추적, 원인 제거가 있다.

54. Section 137

유지보수 작업 : 완전화 유지보수, 적응 유지보수, 수정(교정, 수리) 유지보수, 예방 유지보수

55. Section 137

유지보수

소프트웨어 개발이 성공적으로 완료되어 사용자에게 인도된 후 발생되는 여러 가지 사항에 맞게 수정, 보완, 적응시켜 나가기 위한 일련의 소프트웨어 관리, 보수 활동이다.

56. Section 129

자료 사전에서 사용되는 표기 기호

- = : 자료의 정의
- + : 자료의 연결
- ( ) : 자료의 생략
- [ ] : 자료의 선택
- { } : 자료의 반복

57. Section 129

- 자료 흐름도(DFD, Data Flow Diagram)의 구성 요소 중 종착지는 사각형으로 표시한다.
- 원은 프로세스를 의미한다.

58. Section 137

소프트웨어 문서가 표준화되었다고 해서 프로그램 개발 인력이 감소되지는 않는다.

59. Section 132

① 모듈의 독립성을 높여주기 위해서는 각 모듈 간의 관련성을 최소로 하며, 이 경우에 결합도가 최소가 된다.

② 모듈 간의 관련성을 최대로 하면 모듈의 독립성은 저하되며, 이 경우 모듈의 결합도(Coupling)는 최대가 된다.

③ 복잡성을 감소시키는 수단으로 독립성의 개념이 많이 적용되고 있으며, 모듈의 독립성 척도로서는 결합도와 응집도가 모두 사용된다.

4장 정답 및 해설 — 객체지향 S/W 공학

1.①	2.③	3.④	4.②	5.④	6.③	7.④	8.②	9.③	10.②	11.④	12.②	13.③	14.④	15.②
16.②	17.④	18.③	19.④	20.②	21.④	22.①	23.③	24.④	25.①	26.④	27.③	28.②	29.③	30.①
31.④	32.②	33.④	34.④	35.①	36.②	37.①	38.④	39.①	40.②	41.②	42.③			

1. Section 138

객체지향적 개발 기술은 1980년대에 제기되어 본격적으로 개발되었다.

2. Section 141

- 객체지향 기법은 현실 세계의 개체(Entity)를 기계의 부품처럼 하나의 객체(Object)로 만들어, 기계적인 부품들을 조립하여 제품을 만들 듯이 소프트웨어를 개발할 때도 객체들을 조립해서 작성할 수 있도록 하는 기법이다.
- ③번은 구조적 기법에 대한 설명이다.

3. Section 138

- 객체지향(Object Oriented) 기법은 구조적 방법에서의 생산성 문제를 해결하기 위한 새로운 방법이다.
- 구조적 코딩 기능을 극대화하려면 구조적 기법을 사용해야 한다.

4. Section 138

- 인스턴스 : 클래스에 속한 각각의 객체
- 메시지 : 객체들 간에 상호작용을 하는데 사용되는 수단으로, 객체에게 어떤 행위를 하도록 지시하는 명령 또는 요구사항

5. Section 138

- 객체는 데이터(속성)와 데이터를 처리하는 함수(메소드)를 묶어 놓은(캡슐화한) 하나의 소프트웨어 모듈이다.
- 데이터 : 객체가 가지고 있는 정보로 객체의 속성이나 상태, 분류 등을 나타내며, 속성(Attribute), 상태, 변수, 상수, 자료 구조라고도 함
- 함수 : 객체가 수행하는 기능으로 객체가 갖는 데이터(속성, 상태)를 처리하는 알고리즘이며, 메소드(Method, 행위), 서비스(Service), 동작(Operation), 연산이라고도 함

6. Section 138

- 메시지 : 객체들 간에 상호작용을 하는 데 사용되는 수단으로, 객체에게 어떤 행위를 하도록 지시하는 명령 또는 요구사항
- 추상화 : 불필요한 부분을 생략하고 객체의 속성 중 가장 중요한 것에만 중점을 두어 개략화하는 것, 즉 모델화하는 것
- 객체 : 데이터(속성)와 데이터를 처리하는 함수를 묶어 놓은(캡슐화한) 하나의 소프트웨어 모듈

7. Section 138

객체는 일정한 기억장소를 가지고 있다.

8. Section 138

- 속성 : 객체가 가지고 있는 정보로 객체의 상태, 분류 등을 나타냄
- 메시지 : 객체들 간 상호작용을 하는 데 사용되는 수단으로, 객체에게 어떤 행위를 하도록 지시하는 명령 또는 요구사항
- 인스턴스 : 클래스에 속한 각각의 객체

9. Section 138

객체의 자료 구조는 데이터, 즉 속성을 의미한다.

10. Section 138

- 메소드(Method) : 객체가 수행하는 기능으로 객체가 갖는 데이터(속성, 상태)를 처리하는 알고리즘이며, 객체의 상태를 참조하거나 변경하는 수단이 되는 것
- 객체(Object) : 데이터(속성)와 데이터를 처리하는 함수를 묶어 놓은(캡슐화한) 하나의 소프트웨어 모듈
- 클래스(Class) : 공통된 속성과 연산(행위)을 갖는 객체의 집합으로 객체의 일반적인 타입(Type)

11. Section 139

- 인스턴스(Instance) : 클래스에 속한 각각의 객체
- 다형성(Polymorphism) : 메시지에 의해 클래스가 연산을 수행하게 될 때 하나의 메시지에 대해 각 클래스가 가지고 있는 고유한 방법으로 응답할 수 있는 능력
- 상속성(Inheritance) : 이미 정의된 상위 클래스(슈퍼 클래스 / 부모 클래스)의 모든 속성과 연산을 하위 클래스가 물려받는 것

12. Section 138

메타 클래스는 클래스의 클래스를 의미하는 것으로, 클래스 계층 트리의 최상단에 위치한다.

13. Section 138, 139

- 추상화(Abstraction) : 불필요한 부분을 생략하고 객체의 속성 중 가장 중요한 것에만 중점을 두어 개략화하는 것, 즉 모델화하는 것
- 메소드(Method) : 객체가 수행하는 기능으로 객체가 갖는 데이터를 처리하는 알고리즘이며 객체의 상태를 참조하거나 변경하는 수단이 됨
- 메시지(Message) : 객체들 간 상호작용을 하는 데 사용되는 수단으로, 객체에게 어떤 행위를 하도록 지시하는 명령 또는 요구사항

14. Section 138

- 객체지향 기법은 객체, 클래스, 속성, 연산, 추상화, 상속성 등의 개념을 지원받아 효율적인 코딩을 통해 재사용을 극대화한 기법이다.
- 응집도(Cohesion)는 구조적 방법에서 설계의 평가 척도가 된다.

15. Section 139

캡슐화와 정보 은닉의 개념을 적용하여 얻을 수 있는 장점에는 유지보수의 용이성, 확장성 등이 있다.

16. Section 139

캡슐화

- 데이터와 데이터를 처리하는 함수를 하나로 묶는 것
- 캡슐화된 객체의 세부 내용이 외부에 은폐되어, 변경이 발생할 때 오류의 파급 효과가 적다.
- 캡슐화된 객체들은 재사용이 용이하다.
- 객체들 간의 메시지를 주고받을 때 각 객체의 세부 내용은 알 필요가 없으므로 인터페이스가 단순해지고, 객체 간의 결합도도 낮아진다.

17. Section 138

객체의 속성

- 주체성 : 객체가 다른 객체들과 분리되어 식별이 가능한 성질
- 분류성 : 같은 속성과 행위를 갖는 객체들을 묶을 수 있는 성질
- 다형성 : 하나의 메시지에 대해 각 객체가 가지고 있는 고유의 방법으로 응답할 수 있는 능력
- 상속성 : 상위 클래스의 모든 속성과 연산을 하위 클래스에 물려주는 성질

18. Section 138

객체지향 기법의 장점

- 소프트웨어의 재사용 및 확장을 용이하게 함으로써 고품질의 소프트웨어를 빠르게 개발할 수 있으며 유지보수가 쉽다.
- 복잡한 구조를 단계적, 계층적으로 표현하고, 멀티미디어 데이터 및 병렬 처리를 지원한다.
- 현실 세계를 모형화하여 사용자와 개발자가 이해하기 쉽다.

19. Section 138

객체지향 분석의 필요성

- 사람이 사고하고 인지하는 틀 내에서 시스템의 요구사항을 정의하며 사용자와 정보를 교환할 수 있다.
- 사용자가 속해 있는 실세계의 문제 영역을 이해하는 데 중점을 둔다.
- 객체의 속성과 메소드가 하나의 엔티티가 되는 모델을 만든다.
- 한 객체와 다른 객체의 종속성을 최소화한다.
- 공통된 속성을 명백히 표현할 수 있다.
- 소프트웨어 생명 주기상에서 일관적으로 나타날 수 있다.
- 재사용을 높일 수 있다.

20. Section 140

객체지향 개발 단계 : 객체지향 분석(OOA, Object Oriented Analysis) → 객체지향 설계(OOD, Object Oriented Design) → 객체지향 프로그래밍(OOP, Object Oriented Programming)

21. Section 140

- **Booch(부치) 방법** : 미시적(Micro) 개발 프로세스와 거시적(Macro) 개발 프로세스를 모두 사용하는 분석 방법으로 클래스와 객체들을 분석 및 식별하고 클래스의 속성과 연산을 정의
- **Rumbaugh(럼바우) 방법** : 가장 일반적으로 사용되는 방법으로 분석 활동을 객체 모델, 동적 모델, 기능 모델로 나누어 수행하는 방법
- **Jacobson 방법** : Use Case를 강조하여 사용하는 분석 방법

22. Section 140

- **객체 모델링(Object Modelling)** : 정보 모델링이라고도 하며 시스템에서 요구되는 객체를 찾아내어 속성과 연산 식별 및 객체들 간의 관계를 규정하여 객체 다이어그램으로 표시하는 것
- **동적 모델링(Dynamic Modelling)** : 상태도를 이용하여 시간의 흐름에 따른 객체들 사이의 제어 흐름, 상호 작용, 동작 순서 등의 동적인 행위를 표현하는 것
- **기능 모델링(Functional Modelling)** : 자료 흐름도(DFD)를 이용하여 다수의 프로세스들 간의 자료 흐름을 중심으로 처리 과정을 표현한 것

23. Section 140

객체지향 분석의 방법론

- **Rumbaugh(럼바우) 방법** : 가장 일반적으로 사용되는 방법으로 분석 활동을 객체 모델, 동적 모델, 기능 모델로 나누어 수행하는 방법
- **Booch(부치) 방법** : 미시적(Micro) 개발 프로세스와 거시적(Macro) 개발 프로세스를 모두 사용하는 분석 방법으로 클래스와 객체들을 분석 및 식별하고 클래스의 속성과 연산을 정의
- **Wirfs-Brock 방법** : 분석과 설계 간의 구분이 없고, 고객 명세서를 평가해서 설계 작업까지 연속적으로 수행하는 기법
- **Jacobson 방법** : Use Case를 강조하여 사용하는 분석 방법
- **Coad와 Yourdon 방법** : E-R 다이어그램을 사용하여 객체의 행위를 모델링하며, 객체 식별, 구조 식별, 주제 정의, 속성과 인스턴스 연결 정의, 연산과 메시지 연결 정의 등의 과정으로 구성하는 기법

24. Section 140

객체지향 기술은 개발 과정 전반에 걸쳐 같은 용어와 개념을 사용하므로 분석, 설계, 구현 단계 사이의 전환이 쉬우며, 각 과정이 명확하게 구분되지 않는다.

27. Section 141

객체지향 설계의 설계 개념 : 추상화, 정보 은닉, 기능 독립성, 모듈화, 상속성

28. Section 141

시스템 기술서(정의서)에서의 명사는 객체를, 동사는 연산이나 객체 서비스를 나타낸다.

29. Section 141

- **시스템 설계** : 분석 단계의 분석 모델을 서브시스템(모듈)으로 분할하고, 시스템의 계층을 정의하며 분할 과정 중에서 성능의 최적 방안, 문제 해결 전략, 자원 분해 등을 확정하는 것
- 분석 단계에서 만들어진 클래스, 속성, 관계들을 설계 모델로 제작하고 상세화하는 것은 객체 설계에서 수행되는 작업이다.

32. Section 141

B●●ch의 객체지향 설계 표기법

- **객체 다이어그램(Object Diagram)** : 객체들 간에 전달되는 메시지 표현
- **클래스 다이어그램(Class Diagram)** : 클래스 간의 관계 표현
- **모듈 다이어그램(Module Diagram)** : 프로그램 구성 요소 표현
- **프로세스 다이어그램(Process Diagram)** : 특정 프로세서들 간의 관계 표현

34. Section 138

객체지향 언어도 제3세대 언어에 해당된다.

35. Section 141

객체지향 프로그래밍 언어에는 Simula, Smalltalk, C++, Java 등이 있으며 이 중 Simula는 1966년 객체지향 언어로는 가장 최초로 발표된 것이다.

36. Section 141

객체지향 언어

- **객체 기반 언어** : 객체의 개념만을 지원하는 Ada, Actor와 같은 언어

- **클래스 기반 언어** : 객체와 클래스의 개념을 지원하는 Clu와 같은 언어
- **객체지향성 언어** : 객체, 클래스, 상속의 개념을 모두 지원하는 가장 좋은 언어로 초기에 발표된 Simula로부터 Smalltalk, C++, Objective C와 같은 언어가 있음

37. Section 141

객체지향 언어의 분류

- **객체 기반 언어** : Ada, Actor
- **클래스 기반 언어** : Clu
- **객체지향성 언어** : Simula, Smalltalk, Objective C, C++

38. Section 141

UML에서 사용되는 그래프 : 사용 사례(Use Case) 다이어그램, 클래스(Class) 다이어그램, 순서(Sequence) 다이어그램, 상태(Status) 다이어그램, 활동(Activity) 다이어그램

39. Section 141

UML에서 사용되는 그래프 중 클래스 다이어그램은 럼바우 분석 활동 중 객체 모델링에, 사용 사례 다이어그램은 기능 모델링에, 순서 · 상태 · 활동 다이어그램은 동적 모델링에 사용된다.

40. Section 139

캡슐화는 데이터(속성)와 데이터를 처리하는 함수를 하나로 묶는 것을 의미하는 것으로, 캡슐화한다고 하여 분석 단계가 간단해지는 것은 아니다.

41. Section 139

- 지문의 내용은 A의 값(Integer, float, char)에 따라 각각 다른 작업을 수행하는 것을 의미하는 것으로, 이와 관련된 내용은 하나의 메시지에 대해 여러 가지 형태의 응답이 있다는 것을 의미하는 다형성이다.
- **클래스(Class)** : 공통된 속성과 연산(행위)을 갖는 객체의 집합으로 객체의 일반적인 타입(Type)
- **상속성(Inheritance)** : 이미 정의된 상위 클래스(부모 클래스)의 모든 속성과 연산을 하위 클래스가 물려받는 것
- **캡슐화(Encapsulation)** : 데이터(속성)와 데이터를 처리하는 함수를 하나로 묶는 것

42. Section 139

- 지문의 핵심은 특정 객체가 가진 고유의 특성을 사용하여 동작한다는 것을 의미하는 것으로, 다형성에 대한 설명이다.
- **추상화(Abstraction)** : 불필요한 부분을 생략하고 객체의 속성 중 가장 중요한 것에만 중점을 두어 개략화하는 것, 즉 모델화하는 것
- **캡슐화** : 데이터(속성)와 데이터를 처리하는 함수를 하나로 묶는 것
- **상속성(Inheritance)** : 이미 정의된 상위 클래스(부모 클래스)의 모든 속성과 연산을 하위 클래스가 물려받는 것

5장 정답 및 해설 — S/W 공학의 발전적 추세

1. ④ 2. ④ 3. ④ 4. ③ 5. ① 6. ② 7. ③ 8. ④ 9. ① 10. ③ 11. ④ 12. ④ 13. ① 14. ④ 15. ②
16. ④ 17. ② 18. ① 19. ③ 20. ③ 21. ④ 22. ① 23. ② 24. ④ 25. ① 26. ③ 27. ② 28. ② 29. ① 30. ④
31. ③ 32. ④

1. Section 142

재사용하기 가장 좋은 개발 기법은 객체나 클래스를 사용하는 객체지향 기법이다.

2. Section 142

소프트웨어 재사용

- 이미 개발된 인정받은 소프트웨어의 전체 혹은 일부분을 다른 소프트웨어 개발이나 유지에 사용하는 것이다.
- 소프트웨어 개발의 품질과 생산성을 높이기 위한 방법으로, 기존에 개발한 소프트웨어와 경험, 지식 등을 새로운 소프트웨어에 적용한다.
- 소프트웨어를 재사용함으로써 개발 시간과 비용을 단축시키고, 품질을 향상시킨다.

3. Section 142

소프트웨어 재사용의 이점

- 개발 시간과 비용을 단축시킨다.
- 소프트웨어 품질을 향상시킨다.
- 소프트웨어 개발의 생산성을 향상시킨다.
- 프로젝트 실패의 위험을 감소시킨다.
- 시스템 구축 방법에 대한 지식을 공유하게 된다.
- 시스템 명세, 설계, 코드 등 문서를 공유하게 된다.

※ 기존 소프트웨어에 새로운 개발 방법론을 도입하기가 어렵다는 것은 소프트웨어 재사용 도입의 문제점에 해당된다.

4. Section 142

- 정규화가 몇 회 이루어졌느냐는 별 의미가 없고, 어느 정도 이루어졌느냐가 중요하다.
- **재사용 평가 기준** : 부품의 크기, 복잡도, 정규화 정도, 재사용 빈도 수

5. Section 142

부품의 크기가 작고 일반적인 설계일수록 재사용 이용률이 높다.

7. Section 143

소프트웨어 재공학은 새로운 요구에 맞도록 기존 시스템을 이용하여 보다 나은 시스템을 구축하고, 새로운 기능을 추가하여 소프트웨어 성능을 향상시키는 것이지만 재개발은 소프트웨어를 다시 개발하는 것이다.

9. Section 143

소프트웨어 재공학의 목적 중 하나가 소프트웨어 요소를 추출하여 정보 저장소에 저장하는 것이다.

10. Section 143

- **분석** : 기존 소프트웨어의 명세서를 확인하여 소프트웨어의 동작을 이해하고, 재공학 대상을 선정하는 것
- **역공학** : 기존 소프트웨어를 분석하여 소프트웨어 개발 과정과 데이터 처리 과정을 설명하는 분석 및 설계 정보를 추출하거나 다시 만들어 내는 작업
- **이식** : 기존 소프트웨어를 다른 운영체제나 하드웨어 환경에서 사용할 수 있도록 변환하는 작업

11. Section 143

소프트웨어 재공학은 새로운 기능을 추가하여 소프트웨어 성능을 향상시키는 것이므로 예방 유지보수 측면에서 소프트웨어 위기를 해결하는 방법이다.

12. Section 143

소프트웨어 재공학은 새로운 요구에 맞도록 기존 시스템을 이용하여 보다 나은 시스템을 구축하고, 새로운 기능을 추가하여 소프트웨어 성능을 향상시키는 것으로, ①, ②, ③번과 같은 장점을 가지고 있다. 하지만 기존 시스템을 사용하므로 최신의 소프트웨어 공학 기법을 적용할 수는 없다.

13. Section 143

역공학은 기존 코드를 복구하는 방법을 말하는 것으로, 기존의 소프트웨어를 변경하는 것이 아니라 분석하는 것이다.

14. Section 143

코드 역공학 절차

코드 → 흐름도 → 자료 구조도 → 자료 흐름도 순으로 재생한다.

15. Section 143

- **재사용** : 이미 개발된 인정받은 소프트웨어의 전체 혹은 일부분을 다른 소프트웨어 개발이나 유지에 사용하는 것
- 코드 역공학과 데이터 역공학은 역공학의 종류에 해당된다.

17. Section 144

클라이언트/서버 시스템

- 상당한 계산력을 중대형 컴퓨터 비용의 몇 분의 일의 비용으로 제공한다.
- 고성능 워크스테이션에서 가능한 그래픽 사용자 인터페이스 기능이 향상되어 사용이 용이하다.
- 개방 시스템을 받아들이도록 참작하고 독려하여 특정 독점 구조에 얽매이지 않고 유용하게 사용된다.

18. Section 144

클라이언트/서버 시스템

- 분산 시스템의 가장 대표적인 모델로 정보를 제공하는 서버와 정보를 요구하는 클라이언트로 구성되어 있는 방식이다.
- LAN을 통하여 클라이언트와 서버가 하나의 작업을 분산, 협동 처리한다.
- **클라이언트/서버 처리 시 필요한 사항** : 클라이언트/서버 사이의 신뢰성 있는 통신, 클라이언트/서버의 협동적인 상호 접속, 클라이언트/서버 사이에 응용 처리 분배인 상호 접속

19. Section 144

미들웨어는 클라이언트가 서버 측에 어떠한 처리를 요구하고, 또 서버가 그 처리한 결과를 클라이언트에게 돌려주는 과정을 효율적으로 수행하도록 도와주는 소프트웨어이므로 미들웨어를 사용하면 서버와 클라이언트의 작업량이 감소된다.

20. Section 145

③번은 소프트웨어 재공학에 대한 설명이다.

21. Section 145

검토 회의(Walkthrough)는 소프트웨어 개발의 각 단계에서 개최하는 기술 평가 회의이다.

22. Section 145

소프트웨어 공학 과정을 자동화하면 소프트웨어 개발 기간을 단축하고, 유지보수 비용뿐만 아니라 개발 비용도 절감할 수 있다.

23. Section 145

- CAD(Computer Aided Design) : 컴퓨터를 이용한 설계
- CAE(Computer Aided Education) : 컴퓨터를 이용한 교육
- CAM(Computer Aided Manufacturing) : 컴퓨터를 이용한 생산

24. Section 145

IEF는 소프트웨어 개발 주기에 포함되는 전체 과정을 지원하기 위한 통합 CASE 도구이다.

25. Section 145

CASE 정보 저장소

소프트웨어를 개발하는 과정 동안에 모아진 정보를 보관하여 관리하는 곳으로 정보 저장소, CASE 데이터베이스, 요구사항 사전, 저장소 등이라고도 불린다.

26. Section 145

재사용 부품의 크기가 클수록 소프트웨어 재사용률은 낮아진다.

27. Section 145

CASE는 소프트웨어 생명 주기의 어느 부분을 지원하느냐에 따라 상위(Upper) CASE, 하위(Lower) CASE, 통합(Integrate) CASE로 분류할 수 있다.

28. Section 145

CASE(Computer Aided Software Engineering)는 소프트웨어 개발 과정에서 사용되는 요구 분석, 설계, 구현, 검사 및 디버깅 과정 전체 또는 일부를 컴퓨터와 전용 소프트웨어 도구를 사용하여 자동화하는 것으로, CASE를 사용하려면 설계 지식이 필요하다.

29. Section 145

CASE는 프로그램의 구현과 유지보수 작업만을 중심으로하는 것이 아니라 요구 분석, 설계, 구현, 검사 및 디버깅 과정 전체 또는 일부를 컴퓨터와 전용 소프트웨어 도구를 사용하여 자동화하는 것이다.

30. Section 143

소프트웨어 재공학은 자사 소프트웨어뿐만 아니라 다른 소프트웨어도 사용할 수 있다.

31. Section 145

CASE는 소프트웨어 개발 과정에서 사용되는 요구 분석, 설계, 구현, 검사 및 디버깅 과정 전체 또는 일부를 컴퓨터와 전용 소프트웨어 개발 도구를 사용하여 자동화하는 것으로, 언어 번역 기능은 없다.

32. Section 143

역공학이란 기존 소프트웨어를 분석하여 소프트웨어 개발 과정과 데이터 처리 과정을 설명하는 분석 및 설계 정보를 재발견하거나 다시 만들어 내는 작업이다.

5과목 데이터 통신

1장 정답 및 해설 — 데이터 통신의 기본

1. ③ 2. ④ 3. ③ 4. ① 5. ③ 6. ④ 7. ① 8. ③ 9. ④ 10. ③ 11. ④ 12. ④ 13. ③ 14. ② 15. ④
16. ① 17. ③ 18. ② 19. ④ 20. ④ 21. ③ 22. ① 23. ④ 24. ①

1. Section 146

데이터 통신은 2진 부호(0과 1)로 표현된 디지털 정보를 목적물로 하는 통신이다.

2. Section 146

- ARPANET(Advanced Research Project Agency NETwork) : 1969년 미 국방성에서 설치한 최초의 유선 패킷 교환 시스템으로, 인터넷의 효시가 된 통신 시스템
- SABRE(Semi-Automatic Business Research Environment) : 1963년 아메리칸 에어라인 항공사에서 도입한 항공기 좌석 예약 시스템으로, 최초의 상업용 데이터 통신 시스템
- CTSS(Compatible Time Sharing System) : 1964년 MIT 공과대학에서 대학 내 대형 컴퓨터의 공동 이용을 목적으로 시행한 최초의 시분할 시스템

3. Section 146

정보 통신의 발전 단계

광대역 데이터 전송 회선 구축 → 데이터 전용 교환망 구축 → 디지털 전용 회선 구축 → 종합 정보 통신망 구축

4. Section 149

- **흐름 제어** : 수신 측이 수신 가능한 데이터 양 등을 송신 측에 통지함으로써 정보 흐름이 원활하도록 제어
- **정보 전송 단위의 정합** : 전송 정보를 패킷 등의 적당한 길이 단위로 분할 또는 결합
- **투과성** : 전송할 실제 자료에 대한 비트 열에 확장 비트를 부가 또는 소거

5. Section 146

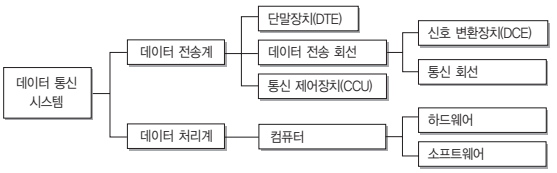
정보 처리의 효율화는 컴퓨터 시스템의 특징이다.

6. Section 146

데이터 통신 시스템은 광대역 전송이 가능하다.

7. Section 147

데이터 통신 시스템의 기본 구성



8. Section 147

주변장치는 데이터 처리계에 해당된다.

9. Section 146

노동 경제성은 노동의 투자로 인해 얻는 이득의 정도를 나타내는 것이다. 정보 통신을 이용한 정보의 이용은 노동보다는 컴퓨터 및 통신 기술을 이용해 수행되는 것으로, 노동 경제성의 향상으로 볼 수 없다.

10. Section 147

DTE(Data Terminal Equipment)는 단말장치를 의미하는 것으로, 데이터 통신 시스템과 외부 사용자의 접속점에 위치하여 최종적으로 데이터를 입 · 출력하는 기능을 한다.

11. Section 146

초기의 정보 통신 시스템 방식은 정보의 전송 과정에 사람이나 중간 매체가 개입된 오프라인(Off-Line) 방식이다.

12. Section 149

메시지 버퍼(Message Buffer) 방식

- 통신 제어장치에서 메시지의 형태로 조립 및 분해까지 하므로, 컴퓨터에 거의 부담을 주지 않으나, 큰 버퍼가 필요하고 구조가 복잡하여 가격이 비싸다.
- 대규모 데이터 통신 시스템에서 많이 사용한다.

13. Section 149

- **통신 방식 제어** : 단방향, 반이중, 전이중 등의 통신 방식 결정
- **다중 접속 제어** : 하나의 통신 회선을 여러 개의 단말장치가 공유하는 경우 전송 회선 선택
- **교환 접속 제어** : 데이터 송·수신을 위한 회선의 설정과 절단

14. Section 149

컴퓨터나 단말장치의 데이터를 통신 회선에 적합한 신호로 변환하는 것은 신호 변환장치의 기능이다.

15. Section 147

DSU : 디지털 데이터 → 디지털 회선

16. Section 148

광섬유 케이블은 감쇠율이 적으므로, 리피터 소요 개수가 적다.

17. Section 148

동축 케이블은 광섬유 케이블에 비해 부피가 크고 무겁다.

18. Section 148

광섬유 케이블은 가늘고 가벼워 취급하기 쉽지만, 광섬유 케이블 간의 연결이 어려워 설치 시 고도의 기술이 필요하다.

19. Section 149

교환 접속 제어는 통신 제어장치의 전송 제어 기능에 속한다.

20. Section 149

- **단말장치** : 데이터 통신 시스템과 외부 사용자의 접속점에 위치하여 최종적으로 데이터를 입·출력하는 장치
- **신호 변환장치** : 컴퓨터나 단말장치의 데이터를 통신 회선에 적합한 신호로 변환하거나 통신 회선의 신호를 컴퓨터나 단말장치에 적합한 데이터로 변경하는 신호 변환 기능을 수행
- **교환 장비** : 통신에 참여하는 두 개체를 연결해 주는 역할

21. Section 148

위성 통신은 거리에 관계없이 요금이 일정하므로 원거리 통신에 경제적이다.

22. Section 148

꼬임선에 비해서는 덜하지만 동축 케이블에도 신호의 감쇠 현상이 나타나며, 신호의 감쇠 현상을 막기 위해 일정 간격마다 중계기를 설치한다. ④번의 베이스밴드용은 디지털 신호 전송에 사용된다는 의미이고, 브로드밴드용은 아날로그 신호 전송에 사용된다는 의미이다.

23. Section 146

- ALOHA(Additive Links On-line Hawaii Area) : 1970년 미 하와이 대학에서 실험적으로 설치한 최초의 무선 패킷 교환 시스템으로, 회선 제어 방식 중 경쟁 방식(CSMA, CSMA/CD)의 모체가 되었음
- SNA(System Network Architecture) : 1974년 IBM에서 발표한 컴퓨터 간 접속 네트워크 시스템 표준으로, 이로 인해 데이터 통신 시스템의 표준화가 시작되었음

24. Section 149

- **교환 접속 제어** : 회선 설정과 절단 기능
- **통신 방식 제어** : 단방향, 반이중, 전이중 등의 통신 방식을 설정하는 기능
- **우회 중계 회선 설정** : 통신 시스템 간에 복수 개의 통신 회선과 중계기가 있는 경우에 통신 회선의 장애가 발생할 때 우회 경로 회선을 설정하는 기능

2장 정답 및 해설 — 데이터 전송 이론

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

1. Section 151

- 디지털 전송(Digital Transmission)은 장거리 전송 시 데이터의 감쇠 및 왜곡 현상을 방지하기 위해서 리피터(Repeater)를 사용한다.
- ① 디지털 전송 시에도 신호의 감쇠 현상은 나타나지만 리피터로 원래의 신호를 복원한 후 전송하기 때문에 잡음에 의한 오류율이 낮다.
- ② 디지털 전송은 아날로그 전송보다 훨씬 큰 대역폭을 필요로 한다.
- ③ 디지털 전송은 디지털 기술의 발전으로 전송 장비의 소형화가 가능하므로 설치 및 유지 비용이 저렴해지고 있다.

2. Section 151

직렬 전송

- 정보를 구성하는 각 비트들이 하나의 전송 매체를 통하여 한 비트씩 차례로 전송되는 형태이다.
- 하나의 전송 매체만 사용하므로 전송 속도가 느리지만 구성 비용이 적게 든다.
- 대부분의 데이터 통신에 사용된다.

3. Section 151

직렬 전송은 주로 데이터 통신에 사용된다. 컴퓨터와 주변장치 사이의 데이터 전송에 사용되는 것은 병렬 전송이다.

4. Section 151

통신 방식은 데이터의 전송 방향에 따라 단방향 통신 방식, 반이중 통신 방식, 전이중 통신 방식으로 나뉜다.

5. Section 151

단방향(Simplex) 통신은 한쪽 방향으로만 전송이 가능한 방식으로, 라디오나 TV 방송에 적용된다.

6. Section 151

①번과 ④번은 단방향 통신에 대한 설명과 사용 예이고, ②번은 반이중 통신의 사용 예이다.

7. Section 151

반이중(Half Duplex) 통신은 2선식 선로를 두어 송신과 수신을 번갈아 수행하도록 하는 것이다.

8. Section 151

맨체스터 부호화 방식은 동기식 전송 중 비트 동기 방식에서 사용한다.

9. Section 151

동기식 전송 방식을 사용하는 단말장치는 반드시 버퍼 기억장치를 내장하여야 한다.

10. Section 151

동기 문자를 삽입하여 데이터 송신 전에 동기화하고, 수신 시점에서 동기화한다.

11. Section 153

DSU(Digital Service Unit)

- 컴퓨터나 단말장치로부터 전송되는 디지털 데이터를 디지털 회선에 적합한 디지털 신호로 변환하거나 그 반대의 기능을 수행한다.
- 디지털 데이터를 디지털 통신망(PSDN)을 이용하여 전송할 때 사용된다.
- 신호의 변조 과정 없이 단순히 유니폴라(단극성) 신호를 바이폴라(양극성) 신호로 변환하여 주는 기능만 제공하기 때문에 모뎀에 비해 구조가 단순하다.
- 속도가 빠르고 오류율이 낮다.

12. Section 154

- 위상 편이 변조는 잡음에 크게 영향을 받지 않는다.
- 신호 변동과 잡음에 약하여 데이터 전송용으로 거의 사용되지 않는 방식은 진폭 편이 변조이다.

13. Section 154

ITU-T에서 9,600bps 모뎀의 표준 방식으로 권고하고 있는 변조 방식은 직교 진폭 변조(QAM)이다.

14. Section 154

반송파의 진폭과 위상을 상호 변환하여 신호를 얻는 변조 방식은 직교 진폭 변조(QAM)이다.

15. Section 155

복호기(Decoder)는 수신된 디지털 신호를 PAM 신호로 변환한다.

16. Section 155

펄스 진폭 변조(PAM)에서 나타난 펄스 진폭의 크기를 디지털 부호(2진수 0과 1로)로 변환하는 것은 부호화(Encoding)이다.

17. Section 155

- 양자화(Quantizing) : 표본화된 PAM 신호를 유한 개의 부호에 대한 대표값으로 조정하는 과정
- 복호화(Decoding) : 수신된 디지털 신호를 PAM 신호로 변환
- 여파화(Filtering) : PAM 신호를 원래의 입력 신호인 아날로그 데이터로 복원하는 과정

18. Section 155

양자화 레벨(스텝) 수가 126단계이므로, 126을 포함할 수 있는 128(2^7)로 계산하면 된다. 양자화 레벨(스텝) = $2^{\text{표본당 전송 비트 수}}$ ⇒ 128 = 2^7이므로, 표본당 전송 비트 수는 7이 된다.

19. Section 158

2,400bps 회선을 4,800bps 회선으로 교체했다는 것은 1초에 2,400개의 비트를 전송하던 회선을 1초에 4,800개의 비트를 전송하는 회선으로 교체했다는 의미이다. 그러므로 전송량은 2배로 증가하고, 처리율이 향상되며, 응답 시간이 향상된다.

20. Section 158

변조 속도는 1초 동안 몇 개의 신호 변화가 있었는가를 나타내며, 단위는 보(baud)를 사용한다.

21. Section 158

변조 속도(baud) = $\frac{1}{T}$ 이므로, T = $\frac{1}{\text{baud}}$ 이 된다.

⇒ T = $\frac{1}{1{,}200}$ [sec]

22. Section 158

베어러 속도는 데이터 신호에 동기 문자, 상태 신호 등을 합한 속도로, 단위는 bps(bit/sec)를 사용한다.

23. Section 158

bps = baud × 변조시 상태 변화 수

- 한 번에 2개의 비트를 전송하므로, 변조 시 상태 변화 수는 2Bit
- bps = 2,400 × 2 = 4,800

24. Section 158

변조 시 상태 변화 수에 대한 언급이 없으면, 50보는 50bps와 같은 의미이다. 따라서 초당 50Bit를 전송한다는 뜻이다. 1분에는 50 × 60 = 3,000Bit를 전송하며, 한 문자는 6비트로 구성되었으므로, 3,000/6 = 500자를 전송할 수 있다.

25. Section 158

문제에서 C는 통신 용량, B는 대역폭, S는 신호 전력, N은 잡음 전력을 의미한다.

26. Section 157

- 주파수 분할 다중화는 전송 신호가 아날로그일 때 사용된다.
- 시분할 다중화에서 세 대의 터미널이 각각 3,600(bps)로 동작할 경우에는 10,800(bps)로 수신기에 전송된다(모든 입력 회선 속도의 합이 출력 회선 속도와 같다).
- 디지털 다중화기로 전송할 경우, 음성은 코덱을 통해 신호 변환한 후 전송한다.

27. Section 157

누화 및 상호 변조 잡음은 하나의 주파수 대역폭을 나누어 사용하는 채널들이 서로 겹치면서 생기는 오류이다.

28. Section 157

주파수 분할 다중화기는 하나의 채널당 하나의 신호만 전송할 수 있으므로, 시분할 다중화기에 비해 다중화 효율이 낮다.

29. Section 157

다중화기의 종류

- **주파수 분할 다중화기**
  - 저속의 데이터를 각각 다른 주파수에 변조하여 전송하는 방식이다.
  - 변조 기능을 하므로 별도의 모뎀이 필요 없으며 비동기식 전송에 적합하다.
  - 시분할 다중화기에 비해 구조가 간단하고 가격이 저렴하다.
  - 채널 간섭을 방지하기 위해 보호 대역(Guard Band)이 필요하다.
- **시분할 다중화기**
  - 전송 데이터를 일정한 시간폭(Time Slot)으로 나누어 차례로 전송하는 방식이다.
  - 동기식에 주로 사용되고 비동기식에도 이용된다.

30. Section 157

전송하려는 신호에 필요한 대역폭보다 통신 회선의 유효 대역폭이 클 때 사용하는 다중화 방식은 주파수 분할 다중화이다.

31. Section 157

하나의 음성은 음성 대역폭(3KHz) + Guard Band(1KHz) = 4KHz이고, 대역폭은 48KHz이다. 대역폭 ÷ 하나의 음성이므로, 48KHz ÷ 4KHz는 12(개)이다.

32. Section 157

매체(통신 회선)의 데이터 전송률이 전송 디지털 신호의 데이터 전송을 능가할 때 사용하는 다중화 방식은 동기 시분할 다중화 방식이다.

33. Section 157

시분할 다중화(TDM)는 포인트 투 포인트(Point-to-Point)에서, 주파수 분할 다중화(TDM)는 멀티 포인트(Multi-Point)에서 많이 사용한다.

34. Section 157

비동기 시분할 다중화는 동일한 조건인 경우 동기식 시분할 다중화 방식보다 많은 수의 단말기들을 전송 매체에 접속할 수 있다.

35. Section 157

비동기식 시분할 다중화기(ATDM, Asynchronous TDM)

- 마이크로프로세서를 이용하여 접속된 단말기 중 전송할 데이터가 있는 단말기에만 시간폭(Time Slot)을 제공한다.
- 비동기식 시분할 다중화기는 낭비되는 시간폭을 줄일 수 있고, 남는 시간폭을 다른 용도로 사용할 수 있으므로, 전송 효율이 높다.
- 동일한 조건일 경우 동기식 시분할 다중화기보다 많은 수의 단말기들을 전송 매체에 접속할 수 있다.
- 다중화기의 내부 속도와 단말기의 속도 차이를 보완해 주는 버퍼가 필요하다.
- 데이터 전송량이 많아질 경우 전송 지연이 생길 수 있다.
- 동기식 시분할 다중화기에 비해 접속에 소요되는 시간이 길다.
- 주소 제어, 흐름 제어, 오류 제어 등의 기능이 필요하므로 장비가 복잡하고, 가격이 비싸다.
- 지능 다중화기, 확률적 다중화기, 통계적 다중화기라고도 한다.

36. Section 157

지능 다중화기는 비동기식 시분할 다중화기를 의미하는 것으로, 회로가 복잡하고 데이터를 임시로 저장하기 위한 버퍼가 필요하므로 가격이 비싸지며, 동기식 시분할 다중화기에 비해 접속에 소요되는 시간이 길다.

37. Section 157

실제 전송할 데이터가 있는 단말기에만 시간폭(Time Slot)을 할당하는 것은 비동기식 시분할 다중화기 또는 집중화기이다.

38. Section 157

집중화기(Concentrator)는 하나 또는 소수의 고속 회선에 여러 대의 저속 단말기를 접속하여 사용할 수 있도록 하는 장치이다.

39. Section 157

시분할 다중화기는 통신 회선의 대역폭을 일정한 시간폭(Time Slot)으로 나누어 여러 대의 단말장치가 동시에 사용할 수 있도록 한 것으로 시간폭의 할당 방식에 따라 동기식 시분할 다중화기와 비동기식(통계적) 시분할 다중화기로 구분된다.

- **동기식 시분할 다중화기** : 모든 단말기에 균등한(고정된) 시간폭(Time Slot)을 제공함으로써 접속장치들의 데이터 전송률의 합과 다중화된 회선의 데이터 전송률이 같음(다중화된 회선의 데이터 전송률 = 접속장치들의 데이터 전송률의 합)
- **비동기식(통계적) 시분할 다중화기** : 전송할 데이터가 있는 단말기에만 시간폭(Time Slot)을 제공함으로써 접속장치들의 데이터 전송률의 합보다 다중화된 회선의 데이터 전송률이 작음(다중화된 회선의 데이터 전송률 < 접속장치들의 데이터 전송률의 합)

40. Section 157

집중화기(Concentrator)는 동적인 공동 이용 방식이다.

41. Section 151

- Control – 제어 정보, Error Check – 오류 검출, User Data – 전송 데이터 블록, Address – 도착지 주소, Sync – 동기 문자로 구성된 프레임 단위로 전송하는 방식을 동기식 전송이라고 한다.
- Sync는 송 · 수신 측의 동기 유지를 위해 사용되는 SYN(SYNchronous idle)을 의미한다.

42. Section 157

물리적인 전송로 수를 감소시키는 장치에는 다중화기(멀티플렉서), 집중화기, 공동 이용기가 있다.

43. Section 151

동기식 전송은 프레임 단위로 연속적으로 전송하므로 전송 효율이 좋고, 비동기식 전송은 한 문자씩 전송하므로 전송 효율이 나쁘다.

44. Section 154

문제에 제시된 그림은 8위상 2진폭 변조 방식을 나타낸 것으로, 동시에 4Bit씩 전송할 수 있다.

45. Section 157

μ는 10^{-6}이므로 125μs는 125/1,000,000 = 0.000125초이다. 0.000125초에 8Bit씩 전송하므로 1초에는 8/0.000125 = 64,000비트를 전송할 수 있다. 1초에 전송할 수 있는 비트의 수가 전송률이므로 전송률은 64,000bps이다. 이것을 K 단위로 나타내면 약 64Kbps이다.

※ $1K = 2^{10} = 1024 \fallingdotseq 1{,}000$

※ $64{,}000 \fallingdotseq 64K$

※ $1\mu s = 10^{-6}Sec$

46. Section 155

①번은 여파화, ②번은 부호화, ③번은 표본화 과정에 대한 설명이다.

3장 정답 및 해설 — 전송 제어 방식

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

1. Section 159

전송 제어 절차

데이터 통신 회선의 접속 → 데이터 링크 설정(확립) → 정보 메시지 전송 → 데이터 링크 종결 → 데이터 통신 회선의 절단

2. Section 159

데이터 통신 회선의 접속 단계는 통신 회선과 단말기를 물리적으로 접속하는 단계로, 교환 회선을 이용한 포인트 투 포인트(Point-to-Point) 방식이나 멀티 포인트(Multi-Point) 방식으로 연결된 경우에만 필요한 단계이다.

4. Section 159

수신 측에 정보를 전송하면서 오류 제어를 수행하는 것은 전송 제어 절차 3단계인 정보 메시지 전송 단계에서 수행되는 작업이다.

- **데이터 링크의 설정(확립)** : 수신 측 호출 → 정확한 수신 측인지 확인 → 수신 측의 데이터 전송 준비 상태 확인 → 송 · 수신 입장의 확인 → 수신 측 입 · 출력 기기 지정 순으로 구성됨

5. Section 160

DLE(Data Link Escape)

전송 제어 문자와 전송 데이터를 구분하기 위한 보조적 제어 문자로, 전송 제어 문자 앞에 삽입하여 뒤에 오는 문자가 전송 제어 문자임을 알린다.

6. Section 160

- SOH(Start of Heading) : 헤딩의 시작
- STX(Start of Text) : 본문의 시작 및 헤딩의 종료
- DLE(Data Link Escape) : 전송 제어 문자 앞에 삽입하여 전송 제어 문자임을 알림

7. Section 160

상대편에 데이터 링크 설정 및 응답을 요구하는 전송 제어 문자는 ENQ(ENQuiry)이다.

8. Section 160

BSC는 포인트 투 포인트(Point-to-Point) 방식과 멀티 포인트(Multi-Point) 방식에서 사용할 수 있다.

9. Section 160

- BSC(Binary Synchronous Control)는 각 프레임에 전송 제어 문자를 삽입하여 전송을 제어한다.
- 플래그(01111110)를 데이터의 처음과 끝에 표시하여 동기를 맞추는 방식은 HDLC이다.

10. Section 160

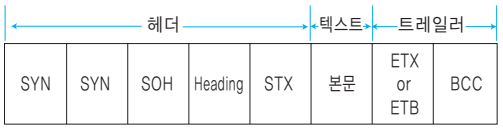
본문의 종료는 ETX이고, STX는 본문의 시작 및 헤딩의 종료를 의미한다.

11. Section 160

- NAK(Negative Acknowledge) : 수신된 메시지에 대한 부정 응답
- STX(Start of TeXt) : 본문의 시작 및 헤딩의 종료
- ENQ(ENQuiry) : 상대편에 데이터 링크 설정 및 응답 요구

12. Section 160

BSC 프레임 구조



13. Section 161

HDLC는 동기식 전송 방식을 사용한다.

14. Section 161

HDLC 프레임 구조

플래그	주소부	제어부	정보부	FCS	플래그
8Bit	8Bit (확장 가능)	8Bit	임의(n)Bit	16/32Bit	8Bit

15. Section 161

제어부의 첫 번째 비트가 '0'인 프레임은 사용자 데이터 전달을 위한 정보 프레임(Information Frame)이다.

16. Section 161

데이터의 투명성을 보장하기 위해 전송 데이터 중간에 의미 있는 비트(Bit)를 끼워 넣는 것을 비트 스터핑이라고 한다.

17. Section 160

PPP는 영역 연결 과정을 제어하는 LCP(Link Control Protocol) 프로토콜과 네트워크 계층의 다양한 프로토콜이 서로 전송할 수 있도록 제어하는 NCP(Network Control Protocol)에 의해 구현된다.

18. Section 161

제어부 프레임의 종류

- 정보 프레임(Information Frame) : 사용자 데이터를 전달하는 역할을 함
- 감독 프레임(Supervisor Frame) : 오류 제어와 흐름 제어를 위해 사용됨
- 비번호 프레임(Unnumbered Frame) : 링크의 동작 모드 설정과 관리 및 오류 회복을 위해 사용됨

19. Section 161

피기백킹(Piggybacking)이란 데이터 프레임에 확인 응답을 포함시켜 전송하는 것으로 I-프레임(정보 프레임)에서 사용한다.

20. Section 161

HDLC의 데이터 전송 모드

- 표준 응답 모드(NRM, Normal Response Mode) : 반이중 통신을 하는 포인트 투 포인트(Point-to-Point) 또는 멀티 포인트(Multi-Point) 불균형 링크 구성에 사용하는 것으로, 종국은 주국의 허가(Poll)가 있을 때에만 송신
- 비동기 응답 모드(ARM, Asynchronous Response Mode) : 전이중 통신을 하는 포인트 투 포인트(Point-to-Point) 불균형

링크 구성에 사용하는 것으로, 종국은 주국의 허가(Poll) 없이도 송신이 가능하지만, 링크 설정이나 오류 복구 등의 제어 기능은 주국만 함

- 비동기 균형 모드(ABM, Asynchronous Balanced Mode) : 포인트 투 포인트(Point-to-Point) 균형 링크에서 사용하는 것으로, 혼합국끼리 허가 없이도 송신이 가능하도록 설정

21. Section 161

HDLC의 국(Station)

- 주국(Primary Station) : 종속된 단말기의 오류를 제어하고 복구의 책임을 가지며, 필요한 정보를 제공해 주는 컴퓨터로, 주 스테이션, 1차국, 서버라고도 함
- 보조국(Secondary Station) : 주국으로부터 제어를 받고 정보를 제공받는 컴퓨터로, 종국, 2차국, 클라이언트라고도 함
- 복합국(Combined Station) : 상대국 컴퓨터의 제어를 받기도 하고 제어를 하기도 하는 동등한 위상을 가진 컴퓨터로, 혼합국이라고도 함

22. Section 161

비트 스터핑(Bit Stuffing)이란 전송 데이터 중간에 의미있는 비트(Bit)를 끼워넣는 것을 말한다. HDLC 프로토콜의 프레임 중 프레임의 시작과 끝을 알리는 고유한 비트 패턴인 플래그가 01111110인데, 실제 전송하는 데이터 중에도 01111110이 존재한다면 이는 플래그로 인식될 수 있는 상황이 초래한다. 이를 방지하기 위해 전송하는 데이터에 1이 5번 연속되면 강제로 11111 뒤에 0을 끼워넣어(Stuffing) 11110으로 만들어 전송하고 수신측에서는 0을 제거한 후 사용하도록 하는 것이다.

23. Section 161

HDLC의 데이터 전송 모드

- 표준 응답 모드(NRM, Normal Response Mode) : 반이중 통신을 하는 포인트 투 포인트(Point-to-Point) 또는 멀티 포인트(Multi-Point) 불균형 링크 구성에 사용하는 것으로, 종국은 주국의 허가(Poll)가 있을 때에만 송신
- 비동기 응답 모드(ARM, Asynchronous Response Mode) : 전이중 통신을 하는 포인트 투 포인트(Point-to-Point) 불균형 링크 구성에 사용하는 것으로, 종국은 주국의 허가(Poll) 없이도 송신이 가능하지만, 링크 설정이나 오류 복구 등의 제어 기능은 주국만 함
- 비동기 균형 모드(ABM, Asynchronous Balanced Mode) : 포인트 투 포인트(Point-to-Point) 균형 링크에서 사용하는 것으로, 혼합국끼리 허가 없이도 송신이 가능하도록 설정

24. Section 163

- 감쇠 : 전송 신호 세력이 전송 매체를 통과하는 과정에서 거리에 따라 약해지는 현상
- 누화 잡음 : 인접한 전송 매체의 전자기적 상호 유도 작용에 의해 생기는 오류
- 하드와이어 전송 매체란 유선 매체를 의미한다.

25. Section 163

- 번개와 같은 외부적인 충격 또는 통신 시스템의 결함이나 파손 등의 기계적인 충격에 의해 생기는 오류를 충격성 잡음(Impulse Noise)이라고 한다.
- 전송 매체 내부에서 온도에 따라 전자의 운동량이 변화함으로써 생기는 오류를 백색 잡음(White Noise)이라고 한다.
- 하나의 전송 매체를 통해 여러 신호를 전달했을 때 주파수에 따라 그 속도가 달라짐으로써 생기는 오류를 지연 왜곡(Delay Distortion)이라고 한다.

26. Section 163

누화 잡음 = 혼선(Cross Talk Noise)

- 인접한 전송 매체의 전자기적 상호 유도 작용에 의해 생기는 오류이다.
- 신호의 경로가 비정상적으로 결합된 경우 나타난다.
- 전화 통화중 다른 전화의 내용이 함께 들리는 현상을 말한다.

27. Section 163

CRC 방식을 이용하여 프레임 내용에 대한 오류를 검사하는 것은 FCS(Frame Check Sequence Field, 프레임 검사 순서 필드)의 역할이다.

28. Section 163

데이터 오류를 정정할 수 있는 기술에는 해밍 코드 방식과 상승 코드 방식이 있다.

29. Section 163

전진 오류 수정(FEC)은 데이터 전송 과정에서 발생한 오류를 검출하여, 검출된 오류를 재전송 요구 없이 스스로 수정하는 방식이다.

30. Section 164

해밍 코드 계산

- 전송 비트 중 1, 2, 4, 8, 16, 32, 64, … 2^n번째를 오류 검출을

위한 패리티 비트로 사용한다.

- 총 정보 비트 수가 4개이므로 패리티 비트가 들어갈 자리인 1, 2, 4 자리를 비운 나머지 자리에 정보 비트를 기입한다.

1	2	3	4	5	6	7	
?	?	1	?	0	0	1	**1번 비트 결정** : 3(1), 5(0), 7(1)의 1의 개수를 합하면 짝수이므로, 1번 비트를 1로 하여 전체를 홀수로 맞춘다.
1	?	1	?	0	0	1	

1	2	3	4	5	6	7	
1	?	1	?	0	0	1	**2번 비트 결정** : 3(1), 6(0), 7(1)의 1의 개수를 합하면 짝수이므로, 2번 비트를 1로 하여 전체를 홀수로 맞춘다.
1	1	1	?	0	0	1	

1	2	3	4	5	6	7	
1	1	1	?	0	0	1	**4번 비트 결정** : 5(0), 6(0), 7(1)의 1의 개수를 합하면 홀수이므로, 4번 비트를 0으로 하여 전체를 홀수로 맞춘다.
1	1	1	0	0	0	1	

1	1	1	0	0	0	1	전송 비트

31. Section 164

슬라이딩 윈도우(Sliding Window)는 트래픽 제어 방식의 하나로 수신 측의 확인 신호를 받지 않더라도 미리 정해진 패킷의 수만큼 연속적으로 전송하는 방식이다.

32. Section 164

'100 0101'에서 1의 개수는 세 개로 홀수이다. 홀수 패리티이므로 1의 개수를 홀수로 만들기 위해 패리티 비트는 0이 된다.

33. Section 164

- **순환 중복 검사(CRC)** : 다항식 코드를 사용하여 오류를 검출하는 방식
- **해밍 코드** : 수신 측에서 오류가 발생한 비트를 검출한 후 직접 수정하는 방식
- **패리티 검사** : 전송 비트에 1비트의 검사 비트인 패리티 비트(Parity Bit)를 추가하여 오류를 검출하는 방식

34. Section 164

- **패리티 검사(Parity Check)** : 데이터 블록에 1비트의 검사 비트인 패리티 비트(Parity Bit)를 추가하여 오류를 검출하는 방식
- **ARQ(자동 반복 요청)** : 수신 측에서 오류 발생 정보가 송신 측에 전송되면 송신 측에서 오류 발생 블록을 재전송하는 방식
- **해밍 코드** : 수신 측에서 오류가 발생한 비트를 검출한 후 직접 수정하는 방식

35. Section 163

ARQ는 오류 검출 후 재전송을 위한 방식이다.

36. Section 163

정지-대기(Stop-and-Wait) ARQ

- 송신 측에서 한 개의 블록을 전송한 후 수신 측으로부터 응답을 기다리는 방식으로, 만일 긍정 응답(ACK)이면 다음 블록을 전송하고, 부정 응답(NAK)이면 앞서 송신하였던 블록을 재전송한다.
- 구현 방법이 가장 단순하지만 블록을 전송할 때마다 수신 측의 응답을 기다려야 하므로 전송 효율이 떨어진다.

37. Section 163

전송 데이터의 블록 길이를 채널의 상태에 따라 그때그때 동적으로 변경하는 방식은 적응적(Adaptive) ARQ이다.

38. Section 163

적응적 ARQ는 전송 효율이 제일 좋지만, 제어 회로가 매우 복잡하므로 현재 거의 사용되지 않는다.

39. Section 163

Go-Back-N ARQ는 오류가 발생한 프레임 이후의 모든 프레임을 재전송한다. 7번째 프레임까지 전송한 상태에서 4번째 프레임에 오류가 발생했으므로 재전송되는 프레임은 4, 5, 6, 7번 4개의 프레임이다.

40. Section 163

- 선택적 재전송(Selective Repeat) ARQ는 여러 블록을 연속적으로 전송하는 연속(Continuous) ARQ에 속한다.
- 한 번에 한 블록만 전송하고, 수신 측의 응답을 기다리는 것은 정지-대기(Stop-and-Wait) ARQ이다.

41. Section 160

- SOH : 헤딩의 시작
- ETB : 블록의 종료
- NAK : 부정 응답

42. Section 161

HDLC

- HDLC는 비트(Bit) 프레임 단위로 데이터를 전송하는 것으로 전송 효율과 신뢰성이 높다.
- 포인트 투 포인트, 멀티 포인트, 루프 등 다양한 데이터 링크 형

태에 동일하게 적용 가능하다.

- 단방향, 반이중, 전이중 통신을 모두 지원하며, 동기식 전송 방식을 사용한다.
- 전송 제어상의 제한을 받지 않고 자유로이 비트 정보를 전송할 수 있다(비트 투과성).

43. Section 163

시스템적 왜곡에는 손실, 감쇠, 하모닉 왜곡 등이 있다.

44. Section 163

시스템의 결함이나 기계적인 충격에 의해 순간적으로 발생하는 높은 진폭의 잡음을 충격성 잡음이라고 한다.

45. Section 164

❶ 총 비트 수가 7개이므로 패리티 비트는 1, 2, 4자리이다.

1	2	3	4	5	6	7
0	0	1	1	0	1	1
P1	P2		P3			

❷ P1은 1, 3, 5, 7번 비트를 이용하여 짝수 패리티를 확인한다. 1, 3, 5, 7번 비트는 0, 1, 0, 1, 즉 1의 개수가 짝수이므로 P1로 체크한 비트에는 오류가 없다. 오류가 없으면 0이다.

❸ P2는 2, 3, 6, 7번 비트를 이용하여 짝수 패리티를 확인한다. 2, 3, 6, 7번 비트는 0, 1, 1, 1, 즉 1의 개수가 홀수이므로 P2로 체크한 비트에는 오류가 있다. 오류가 있으면 1이다.

❹ P3는 4, 5, 6, 7번 비트를 이용하여 짝수 패리티를 확인한다. 4, 5, 6, 7번 비트는 1, 0, 1, 1, 즉 1의 개수가 홀수이므로 P3로 체크한 비트에는 오류가 있다. 오류가 있으면 1이다.

∴ ❷, ❸, ❹번의 체크로 발생된 결과를 역으로 나열하면 110, 즉 십진수로 6이다. 결과가 6이란 것은 6번째 비트가 오류란 의미이므로 6번째 비트 1을 0으로 변경하여 오류를 수정한다.

1	2	3	4	5	6	7
0	0	1	1	0	0	1

4장 정답 및 해설 — 데이터 회선망

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

1. Section 167, 168

회선 교환 방식은 수신 측이 준비되어 있지 않으면 데이터 전송이 불가능하고, 패킷 교환 방식은 응답 시간이 빠르므로 대화형 응용이 가능하다.

2. Section 165

전용 회선 방식은 전송할 데이터의 양이 많고, 회선의 사용 시간이 많을 때 효율적이다.

3. Section 166

멀티 드롭 방식은 회선을 공유하기 때문에 포인트 투 포인트 방식에 비해 회선 및 모뎀의 소요가 적다.

4. Section 169

상대방에 미리 붙여둔 번호를 해석한다는 것은 경로가 이미 결정되어 있다는 것이므로, 네트워크 내의 모든 쌍에 대해서 경로를 미리 정해 놓는 고정 경로 제어(착국 부호 방식)에 속한다.

5. Section 165

사용 방법이 간편하며 업무 적용이 쉽다는 것은 전용 회선(Leased Line)의 장점이다.

6. Section 165

회선 교환 방식은 접속에는 긴 시간이 소요되나 일단 접속되면 전송 지연이 거의 없어 실시간 전송이 가능하다.

7. Section 168

패킷 교환 방식은 음성 전송보다 데이터 전송에 더 적합하다.

8. Section 168

패킷 교환망에서 대량의 데이터 전송 시 전송 지연이 발생할 수 있지만, 전송량 제어 기능을 수행하여 전송 지연을 예방 또는 해결한다.

9. Section 172

토큰이란 채널을 사용할 수 있는 권한을 의미한다.

10. Section 168

패킷 교환망의 기능

- **패킷 다중화** : 물리적으로는 한 개의 통신 회선을 사용하면서도 패킷마다 논리 채널 번호를 붙여 동시에 다수의 상대 터미널과 통신을 수행하도록 하는 기능
- **경로 제어(Routing)** : 출발지에서 목적지까지 이용 가능한 전송로를 찾아본 후에 가장 효율적인 전송로를 선택하는 것
- **논리 채널** : 송·수신 측 단말기 사이에서 논리 채널(가상 회선)을 설정하는 기능
- **순서 제어** : 패킷을 여러 경로를 통해서 전송할 때 패킷의 순서가 송신 측에서 보낸 순서와 다르게 수신되는 것을 방지하기 위한 기능
- **트래픽 제어(Traffic Control)** : 망의 보호, 성능 유지, 망 자원의 효율적인 이용을 위해 전송되는 패킷의 흐름 또는 그 양을 조절하는 기능으로, 교착상태(Dead Lock)의 방지, 흐름 제어 등을 수행
- **오류 제어** : 오류를 검출하고 정정하는 기능

11. Section 168

패킷(Packet)은 전송 혹은 다중화를 목적으로, 메시지를 일정한 비트 수로 분할하여 송·수신 측 주소와 제어 정보 등을 부가하여 만든 데이터 블록이다. 프레임(Frame)과 같은 의미로 사용된다.

12. Section 168

데이터그램 방식은 각 패킷들이 순서에 상관없이 운반되므로, 데이터 전송의 안정성과 신뢰성을 보장하기가 어렵다.

13. Section 169

경로 제어(Routing)

- 송·수신 측 간의 전송 경로 중에서 최적 패킷 교환 경로를 설정하는 기능이다.
- 최적의 패킷 교환 경로란 어느 한 경로에 통화량이 집중하는 것을 피하면서, 최저의 비용으로, 최단 시간에 송신할 수 있는 경로를 의미한다.

14. Section 169

경로 배정 방법(경로 설정 방식)에는 고정 경로 제어(착국 부호 방식), 적응 경로 제어, 범람 경로 제어(플러딩, Flooding), 임의 경로 제어가 있다.

15. Section 169

- ARP(Address Resolution Protocol) : 호스트의 IP 주소를 호스트와 연결된 네트워크 접속장치의 물리적 주소(MAC Address)로 바꿈
- 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 막음
- HTTP(HyperText Transfer Protocol) : 하이퍼텍스트 문서를 전송하기 위해 사용하는 프로토콜

16. Section 169

RIP와 OSPF는 하나의 자율 시스템(AS) 내의 경로 설정에 사용되는 프로토콜인 IGP(Interior Gateway Protocol, 내부 게이트웨이 프로토콜)에 속한다.

17. Section 168

데이터그램 전송 방식은 연결 경로를 설정하지 않고 인접한 노드의 전송량을 감안하여 각각의 패킷을 순서에 관계없이 전송하므로 목적지에서는 패킷의 순서를 재구성해야 하지만 가상회선 전송 방식은 송신지와 수신지 사이의 연결 경로를 미리 설정한 후 각각의 패킷을 순서적으로 전송하므로 패킷의 순서를 재구성할 필요가 없다.

18. Section 169

슬라이딩 윈도우(Sliding Window)

- 수신 측이 수신할 수 있는 최대 패킷의 수(윈도우 사이즈)를 정해서, 수신 측으로부터 확인 신호를 받지 않더라도 정해진 패킷의 수만큼은 연속적으로 전송할 수 있는 방식이다.
- 한 번에 여러 개의 패킷을 전송할 수 있으므로, 전송 효율이 좋다.

19. Section 170

- LAN은 성형, 버스형, 링형, 계층형으로 구성된다.
- 망(Mesh)형은 공중 통신망에 사용되는 방식이다.

20. Section 168

- GFI(General Format Identifier) : 범용 형식 식별자로 패킷의 성격을 결정짓는 부분
- PTI(Packet Type Identifier) : 패킷 유형 식별자로 패킷의 유형을 결정짓는 부분
- SVC(Switched Virtual Circuit) : 가상 회선 방식의 하나로 데이터를 전송할 때마다 가상회선을 설정하는 방식으로 미리 지정된 상대방과 고정적으로 데이터를 전송하는 PVC(Permanent Virtual Circuit)와 구분됨

21. Section 170

망형(Mesh)은 보통 공중 데이터 통신 네트워크에서 사용된다.

22. Section 171

LAN(Local Area Network)은 공유 매체를 사용하여 매체에 연결된 모든 장치로 데이터를 전송하므로 경로 선택이 필요 없다.

23. Section 176

IPv4와 IPv6의 패킷 헤더의 비교

IPv4	IPv6
Version	Version
Header Length	삭제됨
Type Of Service	Traffic Class
Total Length	Payload Length
Identification	Fragment 헤더로 이동
Flags	Fragment 헤더로 이동
Fragment Offset	Fragment 헤더로 이동
Time To Live(TTL)	Hop Limit
Protocol	Next Header
Header Checksum	삭제됨
Options	Extension Headers

24. Section 172

CSMA/CD는 전송량이 적을 때 매우 효율적이고 신뢰성이 높은 방식으로, 전송량이 많은 경우 충돌이 잦아져서 지연 시간이 급격히 증가한다.

25. Section 173

- **속도 변환** : 축적 기능을 이용하여 속도가 빠른 컴퓨터로부터 데이터를 받아들여 축적해서, 상대방 단말기의 속도에 맞추어 보내주는 기능
- **포맷 변환** : 서로 다른 데이터 표현 형식을 일치시키는 기능

26. Section 169

- **고정 경로 제어** : 네트워크 내의 모든 쌍에 대해서 경로를 미리 정해 놓은 방식
- **적응 경로 제어** : 통신망 내에서 시시각각으로 변하는 통화량에 따라 교환기 및 통신망의 상태 등에 의해 전송 경로를 동적으로 결정하는 방식
- **임의 경로 제어** : 각 교환기가 전달받은 패킷을 인접하는 교환기 중에서 임의로 선택하여 전송하는 방식

27. Section 172

CSMA/CD의 의미

- CS(Carrier Sence) : 통신 회선이 사용중인지를 점검
- MA(Multiple Access) : 통신 회선이 비어 있으면 누구든지 사용 가능
- CD(Collision Detection) : 데이터 프레임을 전송하면서 충돌 여부를 조사

28. Section 174

ISDN은 기존의 회선 교환망이나 패킷 교환망을 이용한 통신 서비스도 포함한다.

29. Section 174

D 채널 : 16kbps 또는 64kbps

30. Section 169

OSPF는 경로 수(Hop)에 제한이 없으므로 대규모 네트워크에서 많이 사용된다.

31. Section 177

- **브리지** : LAN과 LAN을 연결하거나 LAN 안에서의 컴퓨터 그룹(세그먼트)을 연결하는 기능을 수행하며, 데이터 링크 계층에서 동작함
- **허브** : 네트워크를 구성할 때 한꺼번에 여러 대의 컴퓨터를 연결하는 장치로, 각 회선을 통합적으로 관리함

- 리피터 : 전송되는 신호가 전송 선로의 특성 및 외부 충격 등의 요인으로 인해 원래의 형태와 다르게 왜곡되거나 약해질 경우 원래의 신호 형태로 재생하여 다시 전송하는 역할을 수행하며, 물리 계층에서 동작함

32. Section 176

DNS 헤더

- ID : 클라이언트가 보낸 질의와 수신된 응답 간의 일치를 식별함(16Bit)
- 플래그(Flag)
  - QR : Query(0)인지 Response(1)인지를 구별(1Bit)
  - Opcode : Query의 종류(4Bit)
  - AA : 공식으로 인정된 네임 서버인지를 구별(1Bit)
  - TC : 정해진 크기(Size)가 초과됐는지를 구별(1Bit)
  - RD : 재귀(Recursion) 질의가 필요한지를 구별(1Bit)
  - RA : 응답한 네임 서버가 재귀 질의가 가능한지를 구별(1Bit)
  - Z : 예약된 필드(3Bit)
  - Rcode : 지정된 15가지 기능을 수행하는 필드(4Bit)
- QDCOUNT : Question 섹션의 개수
- ANCOUNT : Answer 섹션의 Resource 레코드 개수
- NSCOUNT : Authority 섹션에 있는 서버 Resource 레코드 개수
- ARCOUNT : Additional 섹션에 있는 Resource 레코드 개수

33. Section 176

IPv6은 16비트씩 8부분, 총 128비트로 구성되어 있다.

34. Section 172

토큰 패싱 방법은 회선에 대한 사용 권한을 의미하는 토큰이 각 노드를 차례로 옮겨 다니는 방식으로, 노드의 수가 많을수록 토큰의 순회 시간이 길어져 성능이 나빠지고 속도도 느려진다.

35. Section 169

송신 윈도우는 수신 측으로부터 확인 신호 없이도 수신 측으로 보낼 수 있는 패킷의 개수를 의미하는 것으로, 긍정 수신 응답이 있을 때 송신 윈도우를 증가시킬 수 있다.

36. Section 169

- 오류 제어 : 오류를 검출하고 정정하는 기능
- 순서 제어 : 패킷을 여러 경로를 통해서 전송할 때 패킷의 순서가 송신 측에서 보낸 순서와 다르게 수신되는 것을 방지하기 위한 기능
- 경로 제어 : 출발지에서 목적지까지 이용 가능한 전송로를 찾아본 후에 가장 효율적인 전송로를 선택하는 기능

37. Section 169

트래픽 제어 기법의 종류에는 흐름 제어, 폭주(혼합) 제어, 교착 상태(Dead Lock) 방지가 있다.

38. Section 176

- A 클래스 : 국가나 대형 통신망에 사용, 16,777,216개의 호스트 사용 가능
- B 클래스 : 중대형 통신망에 사용, 65,536개의 호스트 사용 가능
- C 클래스 : 소규모 통신망에 사용, 256개의 호스트 사용 가능
- E 클래스 : 실험용으로 사용

39. Section 177

브리지를 이용한 서브넷(Subnet) 구성 시 전송 가능한 회선 수는 브리지가 n개일 때, $\frac{n(n-1)}{2}$ 개이다. 즉 $\frac{5(5-1)}{2}=\frac{20}{2}$=10개이다.

40. Section 170

DCF(Distributed Coordination Function)는 무선 LAN에서 매체에 경쟁적으로 접근할 때 생길 수 있는 충돌을 방지하는 기능이다.

41. Section 169

통신망 내에서 시시각각으로 변하는 통화량에 따라 교환기 및 통신망의 상태 등에 의해 전송 경로를 동적으로 결정하는 방식을 적응 경로배정(Adaptive Routing)이라고 한다.

5장 정답 및 해설 — 통신 프로토콜

1. ① 2. ③ 3. ③ 4. ① 5. ③ 6. ④ 7. ③ 8. ④ 9. ① 10. ③ 11. ① 12. ④ 13. ③ 14. ③ 15. ②
16. ④ 17. ③ 18. ② 19. ① 20. ④ 21. ② 22. ② 23. ③ 24. ② 25. ② 26. ③ 27. ③ 28. ② 29. ③ 30. ④
31. ③ 32. ④ 33. ④

1. Section 178

통신 프로토콜의 기본 요소

- 구문(Syntax) : 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정
- 의미(Semantics) : 두 개체 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정
- 시간(Timing) : 두 개체 간의 통신 속도, 메시지의 순서 제어 등을 규정

2. Section 178

- 폴링/셀렉션은 회선 제어 방식이다.
- 회선 제어 방식은 하나의 통신 회선을 공유하는 여러 대의 단말 장치들이 통신 회선을 사용하는 방식에 대한 규약이다.

3. Section 178

캡슐화(Encapsulation)할 때 제어 정보에 포함되는 것은 주소, 오류 검출 코드, 프로토콜 제어 정보이다.

4. Section 178

- 캡슐화(Encapsulation) : 단편화된 데이터에 주소, 오류 검출 코드, 프로토콜 제어 정보 등을 부가하는 작업
- 동기화(Synchronization) : 송·수신 측이 같은 상태를 유지하도록 타이밍(Timing)을 맞추는 기능
- 다중화(Multiplexing) : 한 개의 통신 회선을 여러 가입자들이 동시에 사용하도록 하는 기능

5. Section 178

캡슐화(Encapsulation)

- 단편화된 데이터에 주소, 오류 검출 코드, 프로토콜 제어 정보 등을 부가하는 작업으로, HDLC 프레임이 대표적이다.
- 정보 데이터를 오류 없이 정확하게 전송하기 위해 캡슐화를 수행한다.

6. Section 179

- 정보 표현 형식을 구문 형식으로 변환하는 계층은 표현 계층이다.
- 데이터 링크 계층은 인접한 두 개의 통신 시스템 간에 신뢰성 있고 효율적인 프레임 데이터를 전송할 수 있도록 한다.

7. Section 179

한 계층을 수정할 때 다른 계층에 영향을 주지 않도록 하였다.

8. Section 179

서비스 접근점(SAP, Service Access Point)

- 상위 계층이 자신의 하위 계층으로부터 서비스를 제공받는 점(Point)을 말한다.
- OSI 7계층의 각 계층은 자신의 하위 계층으로부터 서비스를 제공받으며, 이때 하위 계층과 상위 계층의 통신 경계점을 서비스 접근점(SAP)이라고 한다.

9. Section 179

물리 계층(Physical Layer)

- 전송에 필요한 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성을 정의한다.
- 물리적 전송 매체와 전송 신호 방식을 정의한다.
- RS-232C, X.21 등의 표준이 있다.

10. Section 179

LAN은 물리 계층과 데이터 링크 계층으로 구성되며, 데이터 링크 계층은 매체 접근 제어(MAC, Medium Access Control) 계층과 논리 링크 제어(LLC, Logical Link Control) 계층으로 분류한다.

11. Section 179

경로 설정은 네트워크 계층의 기능이다.

12. Section 181

End to End의 통신 서비스는 전송 계층에서 제공한다.

13. Section 179

경로 설정(Routing), 데이터 교환 및 중계, 트래픽 제어, 패킷 정보 전송을 수행하는 것은 네트워크 계층의 기능이다.

14. Section 179

개체들 간의 관련성을 유지하고 대화 제어를 담당하는 계층은 세션 계층이다.

15. Section 181

직접 전송이란 현재 네트워크 상에 최종 목적지가 존재하여 해당 최종 목적지까지 직접 전송하는 것을 말하며, 간접 전송이란 현재 네트워크 상에 최종 목적지가 존재하지 않아 최종 목적지가 존재하는 네트워크의 라우터까지 전송하는 것을 말한다.

16. Section 180

X.25는 가상 회선 방식을 이용한 연결 지향성 프로토콜이다.

17. Section 180

프레임 릴레이

- 프레임 릴레이는 기존의 X.25가 갖는 오버헤드를 제거하여 고속 데이터 통신에 적합하도록 개선한 프로토콜이다.
- 데이터 패킷과 제어 패킷이 다른 채널을 통하여 전송되도록 하고, 전체 계층을 두 개로 구성하였다. 그리고 순서 제어, 저장 후 전송, 오류 복구 등의 처리 절차를 생략하는 등 데이터 처리 절차를 단순화함으로써 성능을 향상시켰다.

18. Section 180

패킷 교환기망과 단말장치 간의 물리적 접속에 관한 인터페이스를 정의하는 것은 물리 계층의 기능이다.

19. Section 181

TCP/IP의 특징

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

20. Section 181

SNMP, HTTP, SMTP는 응용 계층 프로토콜이다.

21. Section 181

경로 설정은 IP(Internet Protocol)의 기능이다.

22. Section 179

트랜스포트 계층의 전송 서비스는 전송 연결 설정 → 데이터 전송 → 전송 연결 해제 순이다.

23. Section 179

응용 계층은 사용자(응용 프로그램)가 OSI 환경에 접근할 수 있도록 서비스를 제공한다.

24. Section 180

- ① : X.25는 OSI 계층의 1, 2, 3계층에 속하며 네트워크 계층과 관련된 기능을 주로 수행함
- ③ : X.25는 패킷 교환망에서 사용함
- ④ : X.25는 패킷 단위로 전송이 이루어짐

25. Section 179

PSH(PuSH)는 빠른 데이터의 송신을 요청한다는 의미이다.

26. Section 179

흐름 제어와 오류 제어 기능 때문에 데이터 링크 계층이 답이라고 생각할 수도 있지만, 이 문제에서 중요하게 생각해야할 부분은 경로 선택 기능에 대한 것이다. 그러므로 답은 네트워크 계층이 된다.

27. Section 180

비트 중심이고, X.25의 2계층에서 사용하는 프로토콜은 LAPB이다.

- ADCCP(Advanced Data Communications Control Procedure, 고급 데이터 통신 제어 절차) : 미국표준협회(ANSI)에서 제정한 비트 중심 프로토콜로, HDLC, SDLC와 거의 같은 절차로 이루어짐

28. Section 181

SNMP는 응용 계층 프로토콜이다.

29. Section 181

RTP 헤더의 주요 필드

- Version(V) : RTP의 버전을 나타냄(2Bit)
- Padding(P) : Payload 끝 부분에 추가 데이터가 있음을 의미함(1Bit)
- Extension(X) : 헤더 다음에 헤더가 확장됨을 의미함(1Bit)
- CSRC count(CC) : 헤더 뒤에 나오는 CSRC identifier의 개수를 의미함(4bit)
- Marker(M) : 프레임의 경계를 표시하는데 사용됨(1Bit)
- Payload Type(PT) : Payload의 형식을 식별함(7Bit)
- Sequence Number : 패킷이 송신될 때마다 증가하며, 수신 측에서는 이 필드를 이용해 패킷 분실을 감지함(16Bit)

30. Section 181

ICMP의 헤더는 8Byte로 구성된다.

31. Section 181

- IP : 전송할 데이터에 주소를 지정하고, 경로를 설정하는 기능 수행
- TCP : 신뢰성 있는 경로를 확립하고 메시지 전송을 감독하는 기능 수행
- FTP : 컴퓨터와 컴퓨터 또는 컴퓨터와 인터넷 사이에서 파일을 주고받을 수 있도록 하는 원격 파일 전송 프로토콜

32. Section 181

- HTTP : 하이퍼텍스트 문서를 전송하기 위해 사용되는 프로토콜
- SMTP : 전자 우편을 교환하는 서비스

33. Section 181

Transmission(전송) 계층에서 사용되는 프로토콜은 TCP와 UDP이다.

광고
기사필기_정답131p광고.pdf

광고
기사필기_정답132p광고.pdf