전자계산기 구조

1 논리회로

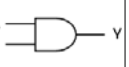
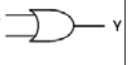
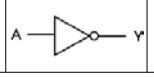
1.1 불 대수

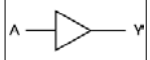
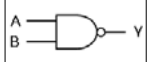
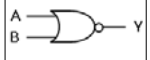
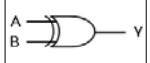
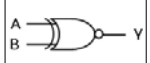
불대수의 기본 공식

용어	설명
교환법칙	• $A+B=B+A$ • $A \cdot B = B \cdot A$
결합법칙	• $A+(B+C)=(A+B)+C$ • $A \cdot (B \cdot C)=(A \cdot B) \cdot C$
분배법칙	• $A \cdot (B+C)=A \cdot B + A \cdot C$ • $A+(B \cdot C)=(A+B) \cdot (A+C)$
멱등법칙	• $A+A=A$ • $A \cdot A = A$
보수법칙	• $A+\overline{A}=1$ • $A \cdot \overline{A}=0$
항등법칙	• $A+0=A$ • $A+1=1$ • $A \cdot 0=0$ • $A \cdot 1=A$
콘센서스	• $AB+BC+C\overline{A}=AB+C\overline{A}$
드모르강	• $\overline{A+B}=\overline{A} \cdot \overline{B}$ • $\overline{A \cdot B}=\overline{A}+\overline{B}$

1.2 논리 게이트 (Logic gate)

논리 게이트의 종류

이 름	기 호	논리식	의 미	진리표
AND		$Y=A \cdot B$ $Y=AB$	• 입력 정보의 값이 모두 1일 때 만 결과가 1이 됨 • 비수치 데이터에서 마스크를 이용하여 불필요한 부분을 제거하기 위한 연산	A B Y 0 0 0 0 1 0 1 0 0 1 1 1
OR		$Y=A+B$	• 입력 정보의 값 중 한 개라도 1이면 결과가 1이 됨 • 두 개의 데이터를 섞거나 일부에 삽입하는 데 사용	A B Y 0 0 0 0 1 1 1 0 1 1 1 1
NOT		$Y=\overline{A}$ $Y=A'$	• 입력 정보의 반대값이 출력됨	A Y 0 1 1 0

이 름	기 호	논리식	의 미	진리표
BUFFER		$Y=A$	• 입력 정보를 그대로 출력	A Y 0 0 1 1
NAND		$Y=\overline{A \cdot B}$ $Y=\overline{AB}$	• NOT + AND 즉, AND의 부정	A B Y 0 0 1 0 1 1 1 0 1 1 1 0
NOR		$Y=\overline{A+B}$	• NOT + OR 즉, OR의 부정	A B Y 0 0 1 0 1 0 1 0 0 1 1 0
XOR		$Y=A \oplus B$ $Y=\overline{A}B+A\overline{B}$	• 입력 정보가 모두 같으면 0, 한 개라도 다르면 1이 출력 • 자료의 특정 비트를 반전시키고자 하는 경우에 사용	A B Y 0 0 0 0 1 1 1 0 1 1 1 0
XNOR		$Y=A \odot B$	• NOT + XOR 즉, XOR의 부정	A B Y 0 0 1 0 1 0 1 0 0 1 1 1

1.3 조합논리회로

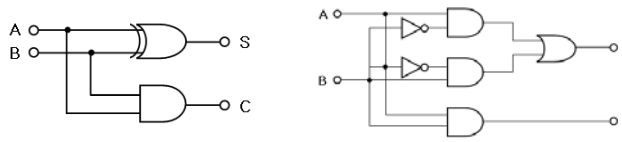
반가산기 (Half Adder)

• 1 Bit짜리 2진수 2개를 덧셈(가산)한 합(S)과 자리올림수(C)를 구하는 조합논리회로

• 진리표

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

• 논리회로: 한 개의 AND 회로와 Exclusive-OR(=XOR) 회로를 조합한 회로



• 논리식
- $S = A\overline{B} + \overline{A}B = A \oplus B$
- $C = AB$

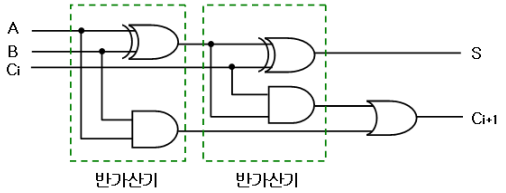
전가산기 (Full Adder)

• 반가산기의 회로에 뒷자리에서 발생한 자리올림수를 처리할 수 있도록 한 회로

• 진리표

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

• 논리회로: 한 개의 전가산기를 구성하는데 최소한 2개의 반가산기가 필요함



• 논리식
- $S = (A \oplus B) \oplus C$
  전가산기의 합의 동작을 얻을 수 있는 것은 **배타적 OR**
- $C_{i+1} = AB + (A \oplus B)C_i$

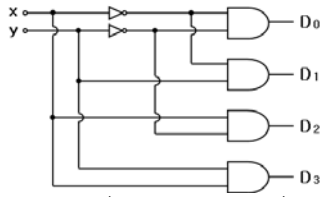
디코더 (Decoder)

• N비트 입력단자를 통하여 들어온 2진 신호를 최대 2^N개의 출력단자 중 하나를 선택하는 회로

• 진리표: 디코더의 출력이 4개일 때, 입력은 보통 2개임

x	y	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

• 논리회로: 디코더는 주로 AND 게이트의 집합으로 구성됨



멀티플렉서 (MUX, Multiplexer)

• N개의 입력 데이터에서 입력선을 선택하여 단일 채널로 송신하는 것

• 버스(bus)를 구성하는데 사용할 수 있는 논리회로

디멀티플렉서(DeMUX, DeMultiplexer)

• 멀티플렉서의 반대 동작을 함

1.4 순서논리회로

플립플롭 (FF, Flip-Flop)

• 1비트(bit)를 기억하는 메모리 소자

RS 플립플롭 (Reset-Set FF)

• 임의의 Bit 값을 그대로 유지시키거나 무조건 0 또는 1의 값을 기억시키기 위해서 사용

S	R	$Q_{(t+1)}$	
0	0	$Q_{(t)}$	불변
0	1	0	Reset
1	0	1	Set
1	1	동작 안 됨	부정

Q(t)	Q(t+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

JK 플립플롭

• RS FF에서 S=R=1일 때 동작되지 않는 결점을 보완한 플립플롭

J	K	$Q_{(t+1)}$	
0	0	$Q_{(t)}$	불변
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q_{(t)}}$	반전

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

T 플립플롭

• JK FF의 두 입력선을 묶어서 한 개의 입력선으로 구성한 플립플롭

마스터-슬레이브 플립플롭

• 출력측의 일부가 입력측에 궤환되어 유발되는 레이스 현상을 없애기 위해 고안된 플립플롭

2 자료의 표현

2.1 정보의 단위

정보의 단위

• 비트 < 니블 < 바이트 < 워드 < 필드 < 레코드 < 블록 < 파일 < 데이터베이스

2.2 진법

진법의 종류

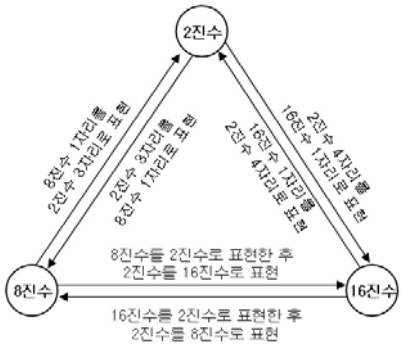
2진법(Binary)	0과 1 두 개의 숫자로 표현
8진법(Octal)	0~7까지의 숫자로 표현하며, 2진수 3자리를 묶어서 하나의 수로 표현
10진법(Decimal)	0~9까지의 숫자로 표현
16진법(Hexadecimal)	0~9까지의 숫자와 A~F까지의 문자(10~15까지를 의미)로 표현

진법 변환

• 10진수 ⇨ 2진수/8진수/16진수
- 정수 부분: 10진수의 값을 변환할 진수로 나누어 더 이상 나눠지지 않을 때까지 나누고, 나머지를 역순으로 표시
- 소수 부분: 10진수의 값에 변환할 진수를 곱한 후 결과의 정부 부분만을 차례대로 표기하되, 소수 부분이 0 또는 반복되는 수가 나올 때까지 곱하기를 반복

• 2진수/8진수/16진수 ⇨ 10진수
- 정수 부분과 소수 부분을 나누어서 변환하려는 각 진수의 자리 값과 자리의 지수 승을 곱한 결과 값을 모두 더함

• 2진수/8진수/16진수 상호 변환



2.3 보수

보수(Complement, 補數)의 개념

• 각 자리 숫자에 대해 N + N' = r 일 때, N'를 N에 대한 r의 보수(Complement)라고 함

• 10진법의 수 274의 9의 보수는?
  274 + ??? = 999 ⇨ 274의 9의 보수는 725

• 보수의 사용 용도: 덧셈과 뺄셈을 덧셈 회로로 처리 가능

보수의 종류: n진법에는 (n-1)의 보수와 n의 보수가 존재함

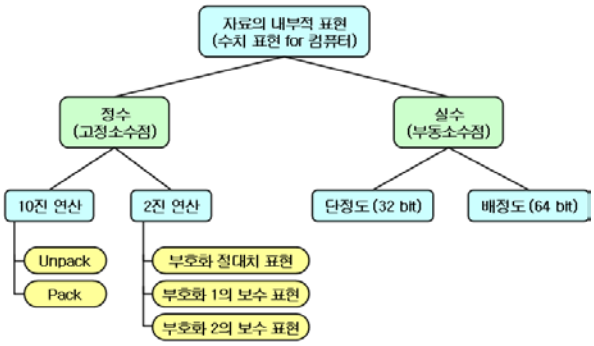
• (n-1)의 보수: 같은 자릿수에서 가장 큰 값이 되기 위해 필요한 수
- 10진법의 수 274의 9의 보수: 725 (274+725=999)
- $(1001011)_2$의 1의 보수: 1001011 + 0110100 = 1111111 ⇨ $(0110100)_2$
  1의 보수는 주어진 각 자리값을 0→1, 1→0으로 변환해도 됨

• n의 보수: 자릿수를 한 자리 늘리기 위해 필요한 수
- 10진법의 수 274의 10의 보수: 726 (274+726=1000)
- $(1001011)_2$의 2의 보수: 1001011 + 0110101 = 10000000 ⇨ $(0110101)_2$
  n의 보수는 (n-1)의 보수에 1을 더해도 됨

보수를 이용한 뺄셈

• A-B는 A+(-B)이므로 B에 대한 보수를 구하여 덧셈 연산으로 뺄셈을 수행

2.4 자료의 내부적 표현



고정 소수점(Fixed Point) 표현

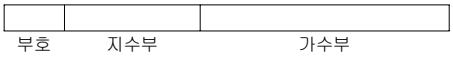
- 10진 연산: 10진수 1자리를 2진수 4자리로 표현하는 방식
  - 10진수 46을 2진화 10진수로 표현하면? ⇨ 01000110
  - 종류 및 용도
    Unpacked format: 10진수 입·출력 형식 (연산은 불가능)
    Packed format: 10진수 연산 형식 (데이터의 입·출력은 불가능)
- 2진 연산
  - 10진수 전체 값을 2진수로 변환하여 표현하는 방식
  - 양수: 부호 비트에 0을 넣고, 변환된 2진수 값을 Data Bit의 오른쪽에서 왼쪽 순으로 차례로 채우고 남는 자리에 0을 채움
  - 음수

종 류	표현 방법	예 제 (-25의 표현)	비 고	표현 범위 (n: 비트 개수)
부호화 절대치 (=부호 및 크기) (Signed Magnitude)	양수 표현 ⇩ 부호비트 0→1	00011001 ⇩ 10011001	2가지 형태의 0 존재 (-0, +0)	$-2^{n-1}+1$ ≀ $+(2^{n-1}-1)$
부호화 1의 보수 (Signed 1's Complement)	양수 표현 ⇩ 1의 보수	00011001 ⇩ 11100110		
부호화 2의 보수 (Signed 2's Complement)	양수 표현 ⇩ 2의 보수	00011001 ⇩ 11100111	한 가지 형태의 0만 존재 (+0)	-2^{n-1} ≀ $+(2^{n-1}-1)$

- 2의 보수 표현 방식으로 8비트의 기억 공간에 정수를 표현할 때 표현 가능 범위
  ⇨ $-2^7 \sim +(2^7-1)$

부동 소수점(Floating Point) 표현

- 부동 소수점 표현의 특징
  - 고정 소수점 표현보다 표현의 정밀도를 높일 수 있음
  - 아주 작은 수와 아주 큰 수의 표현에 적합함
  - 수 표현에 필요한 자리 수에 있어 효율적임
  - 과학이나 공학, 수학적인 응용에 주로 사용되는 수 표현
  - 부동 소수점 수의 연산은 고정 소수점 연산에 비해 복잡하며, 연산 시간이 많이 걸림
- 형식



  부호: 0(양수), 1(음수)
- 부동 소수점의 연산 방법
  - 덧셈, 뺄셈
    · 0인지 여부 조사
    · 가수의 위치 조정: 지수 비교 후 지수 크기가 다르면 지수가 큰 쪽에 일치시킴
    · 가수부 값끼리 더하거나 뺌
    · 결과의 정규화 (지수부와 가수부 분리)
  - 곱셈
    · 0인지의 여부 조사 (한쪽이 0이면 결과가 0이 됨)
    · 지수를 더함 (아래의 5+3)
    · 가수를 곱함 (아래의 0.27×0.18)
    · 결과의 정규화

2.5 자료의 외부적 표현

ASCII 코드 (American Standard Code for Information Interchange)

- 자료의 외부적 표현방식으로 가장 흔히 사용되는 코드
- 영문자(Alphanumeric) 코드에 해당
- IBM사에서 개발한 것으로 데이터 통신 및 마이크로컴퓨터에서 많이 채택되고 있는 코드
- ASCII 코드를 사용하여 통신을 할 때 1 Bit의 패리티 비트를 추가하여 통신함

BCD 코드 (Binary Coded Decimal, 2진화 10진 코드)

- 10진수 1자리의 수를 2진수 4 Bit로 표현
- 4 Bit의 2진수 각 Bit가 8(2^3), 4(2^2), 2(2^1), 1(2^0)의 자리값을 가지므로 8421코드라고도 함
- 대표적인 가중치 코드 (Weight Code)
- BCD코드를 사용하는 이유: 10진수 입·출력이 간편함
- 자체 보수화(self-complementary)는 불가능함
- 예제
  - 10진수 956을 BCD 코드로 변환: 1001 0101 0110
  - 10진수 634를 BCD 코드로 변환: 0110 0011 0100

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

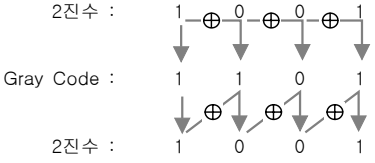
- 각각의 문자에 대하여 8개의 비트(4개의 Zone Bit + 4개의 Digit Bit)와 1개의 패리티 비트로 구성됨

3 초과 코드 (Excess-3 Code)

- 10진수를 표현하기 위한 부호
- BCD 부호에 3을 더한 것과 같음
- 부호를 구성하는 어떤 비트 값도 0이 아님
- BCD code 중에서 산술 연산 작용에 가장 적합
- 대표적인 자기 보수 코드, 비가중치 코드
- 예제
  - 10진수 8을 Excess-3 코드로 표시하면? ⇨ 1000 + 0011 = 1011 (E)
  - 10진수 9를 Excess-3 코드로 표시하면? ⇨ 1001 + 0011 = 1100 (E)

그레이 코드 (Gray Code)

- BCD 코드의 인접하는 비트를 XOR 연산하여 만든 코드
- 이웃하는 코드가 한 비트만 다르기 때문에 코드 변환이 용이
- A/D 변환, 입/출력 장치 등에 주로 사용됨
- Hardware error를 최소로 하는데 적합 (1 Bit만 변화시켜 다음 수치로 증가시키므로)
- 변환 방법



- 예제
  - 2진수 $(1010)_2$ 을 그레이 코드 변환하면? ⇨ 1111
  - Gray code (011011)G을 binary number로 변환시키면? ⇨ $(010010)_2$

패리티 검사 코드 (Parity Check Code)

- 코드의 오류(error, 에러, 착오)를 검사하기 위해서 Data Bit 외에 1 Bit의 패리티 체크 비트를 추가하는 것
- 패리티 비트(Parity Bit): 오류 정보를 검출하기 위해 사용하는 비트 (error 검출용 비트)
- 1 Bit의 오류만 검출 가능
- 종류
  - Odd Parity (기수 패리티)
  - Even Parity (우수 패리티)

해밍 코드 (Hamming Code)

- 오류 검출 및 교정이 가능한 코드
- 해밍 코드는 2 Bit의 오류를 검출할 수 있고, 1 Bit의 오류를 교정할 수 있음
- 해밍 코드 중 1, 2, 4, 8, 16, … 2^n 번째는 오류 검출을 위한 패리티 비트임

코드의 분류

분 류	코드 종류
가중치 코드 (Weighted Code)	BCD(8421) 코드, 2421 코드, Biquinary 코드, 51111 코드
비가중치 코드 (Non-Weighted Code)	Excess-3 코드, Gray 코드, 2 out-of 5 코드
자기 보수 코드 (Self-Complementary Code)	Excess-3 코드, 2421 코드, 51111 코드
오류 검출용 코드	Hamming 코드, 패리티 검사 코드, Biquinary 코드, 2 out-of 5 코드

3 프로세서

3.1 중앙처리장치 (CPU, Central Processing Unit)

중앙처리장치의 구성 및 기능

- 제어장치: 제어 기능
- 연산장치: 연산 기능
- 레지스터: 기억 기능
- 버스: 전달(전송) 기능

제어 장치 (Control Unit)

- 주기억장치에 기억된 명령을 꺼내서 해독하고, 시스템 전체에 지시 신호를 내는 장치
- 명령 코드가 명령을 수행할 수 있게 필요한 제어 기능을 제공

- 제어장치의 구성
  - 프로그램 카운터 (PC, Program Counter)
  - 명령어 레지스터 (IR, Instruction Register)
  - 부호기 (Encoder)
  - 명령 해독기 (Instruction Decoder)

연산장치 (ALU, Arithmetic and Logic Unit)

- 제어장치의 명령에 따라 실제로 연산을 수행하는 장치
- 산술연산, 논리연산, 관계연산, 이동(Shift)등을 수행
- 가산기(Adder), 누산기(AC, Accumulator), 보수기(Complementor), 데이터 레지스터, 오버플로우 검출기, 시프트 레지스터(Shift Register)등으로 구성

레지스터 (Register)

- 중앙처리장치 내부에서 처리할 명령어나 연산의 중간 결과값 등을 일시적으로 기억하는 임시 기억 장소
- 레지스터의 종류 및 기능

종 류	기 능
프로그램 카운터 (PC, Program Counter)	• 차기 명령(Next Instruction)의 번지를 지시 (다음에 실행할 명령의 번지를 가짐) • 프로그램 실행 도중 분기가 발생하면 CPU 내의 PC의 내용을 먼저 변화시켜야 함 (분기명령어가 실행되는 경우에는 그 목적지 주소로 갱신됨)
명령 레지스터 (IR, Instruction Register)	• 현재 실행중인 명령의 내용을 기억 • OP code 명령호출이 IR로 이동
누산기 (AC, Accumulator)	• 연산장치에 있는 레지스터(register)의 하나로 연산 결과를 일시적으로 기억하는 장치 • 주소 부분이 하나밖에 없는 1-주소 명령 형식에서 결과 자료를 넣어 두는데 사용하는 레지스터
상태 레지스터 (Status Register)	• 컴퓨터의 내부 상태를 나타냄
PSWR (Program Status Word Register)	• PSW를 저장하고 있는 레지스터 (PSW: 시스템 내부의 순간순간의 상태를 기록하고 있는 정보)
플래그 레지스터	• CPU 내부에서 방금 행한 연산의 결과로 나타나는 상태 (결과가 0인지여부, 부호(음수인지 양수인지), 캐리 및 오버플로의 발생 여부 등의 상태)를 나타내는 플립플롭
메모리 주소 레지스터 (MAR, Memory Address Register)	• 기억장치를 출입하는 데이터의 번지를 기억하는 레지스터

종 류	기 능
메모리 버퍼 레지스터 (MBR, Memory Buffer Register)	• 기억장치를 출입하는 데이터가 잠시 기억되는 레지스터
인덱스 레지스터 (Index Register)	• 어드레스의 수정, 서브루틴의 연결, 반복 계산 수행 등의 역할을 하는 레지스터
Shift Register	• 자료의 병렬 전송을 직렬 전송으로 변환 • 2배 길이 레지스터(double-length register)라고도 불림
Major State Register	• CPU가 무엇을 하고 있는지 나타냄

버스 (Bus)

- CPU, 메모리, I/O 장치 등과 상호 필요한 정보를 교환하기 위해 연결하는 공동의 전송선
- 전송하는 정보에 따른 버스의 분류
  - 제어버스 (Control Bus)
  - 주소버스 (Address Bus)
  - 데이터버스 (Data Bus)

  CPU와 메모리 혹은 I/O 장치 사이에서 데이터를 전송하는 양방향 버스

3.2 명령어

명령어(Instruction)의 구성

연산자(Operation Code)부	자료(Operand)부

- 연산자부(Op-Code, Operation Code부)
  - 실행할 명령이 들어 있음
  - 명령어의 연산자 부분이 나타낼 수 있는 것
    - 명령어의 형식
    - 연산자
    - 자료의 종류
  - 연산자부의 비트수가 n Bit → 2^n개의 명령어(연산자) 수행 가능
- 자료(Operand)부
  - 자료부 = 어드레스 필드(Address Field) = 주소부
  - 실제 데이터에 대한 정보를 표시하는 부분
  - 어드레스 필드(주소부)의 크기 = 최대 메모리 용량

연산자(Op-Code, Operation Code)의 기능

- 함수 연산 기능
  - 중앙처리장치에서 데이터를 처리하는 기능
  - 산술 · 논리 연산 명령(ADD, AND, CPA, CPC, CLC, ROR, ROL 등)

- 자료 전달 기능
  - 중앙처리장치와 기억장치 사이에서 정보를 교환하는 기능
  - Load: 기억장치(메모리)의 내용을 중앙처리장치(레지스터)에 전달하는 명령
  - Store: 중앙처리장치(레지스터)의 정보를 기억장치(메모리)에 기억시키는 명령
  - Move: 특정 레지스터의 내용을 다른 레지스터로 옮기는 명령
  - Push, Pop: 스택에 자료를 저장, 인출하는 명령
- 제어 기능
  - 프로그램의 수행 흐름을 제어하는데 사용
  - 무조건 분기 명령: GOTO, JMP(Jump)
  - 조건 분기 명령: IF, SPA, SNA, SZA
  - 부프로그램 호출 및 복귀: Call, Return
- 입 · 출력 기능
  - 중앙처리장치와 입출력장치, 또는 기억장치와 입출력장치 사이에서 자료를 전달하는 기능

단항연산자와 이항연산자

- 단항연산자 (Unary Operator)
  - 피연산자가 1개만 필요한 연산자
  - NOT, Complement, Shift, Rotate, MOVE 등
- 이항연산자 (Binary Operator)
  - 피연산자가 2개 필요한 연산자
  - 사칙연산, AND, OR, XOR, XNOR 등

명령어 설계 시 고려할 사항

- 연산자의 종류
- 주소 지정 방식
- 해당 컴퓨터 시스템 단어(word)의 크기(비트수)

명령어 형식 종류

- Operand부의 개수에 따라, 3 / 2 / 1 / 0주소 명령어 형식이 있음

3-주소 명령어

- Operand부가 3개로 구성 (연산의 결과는 Operand 3에 기록됨)

OP-Code	Operand 1	Operand 2	Operand 3
	자료1의 주소	자료2의 주소	결과의 주소

- 여러 개의 범용 레지스터를 가진 컴퓨터에 사용됨
- 장점
  - 연산 후에 입력 자료가 변하지 않고 보존됨
  - 전체 명령어를 읽어오는 시간 단축
  - 프로그램의 길이가 짧아짐
- 단점
  - 명령어 한 개의 길이가 길어짐
  - 하나의 명령을 수행하기 위해서 최소한 4번 기억장소에 접근해야 하므로 전체적인 수행시간 길어짐

2-주소 명령어

- Operand부가 2개로 구성 (연산의 결과는 Operand 1에 기록됨)

OP-Code	Operand 1	Operand 2
	자료1의 주소 결과의 주소	자료2의 주소

- 여러 개의 범용 레지스터를 가진 컴퓨터에 사용됨
- 장점
  - 3주소 명령에 비해 명령어의 길이가 짧음
  - 계산 결과를 시험할 필요가 있을 때 계산 결과가 기억장치에 기억 될 뿐 아니라 중앙처리장치에도남아 있어서 중앙처리장치 내에서 직접 시험이 가능하므로 시간이 절약
- 단점
  - Operand 1에 있던 원래의 자료가 파괴됨
  - 전체 프로그램 길이 증가

1-주소 명령어

- Operand부가 1개로 구성 (연산의 결과는 Operand 1에 기록됨)

OP-Code	Operand 1
	자료1의 주소

- 반드시 누산기(Accumulator)가 필요한 주소지정방식 (하나의 operand가 누산기 속에 포함되고, 연산 결과를 항상 누산기에 저장)
- 1-주소 명령의 예 (C = A + B)

```
LOAD A
ADD B
STORE C
```

0-주소 명령어

- Operand부가 없이 OP-code부만으로 구성 (자료의 주소를 지정할 필요가 없음)

OP-Code

- 모든 연산은 스택(Stack)에 있는 자료를 이용하여 수행
- 주소의 사용 없이 스택에 연산자와 피연산자를 넣었다 꺼내어 연산한 후 결과를 다시 스택에 넣으면서 연산하기 때문에 원래의 자료가 남지 않음
- 수식을 계산할 때 수식을 미리 처리되는 순서인 역 polish(또는 postfix) 형식으로 바꾸어야 함
- Instruction cycle time이 가장 짧은 명령어 형식
- 스택 머신(Stack Machine)이라고도 함 (0-주소 명령형을 갖는 컴퓨터 구조 원리: Stack architecture)

3.3 주소지정방식

주소지정방식 (Addressing Mode)
- 프로그램이 수행되는 동안 사용될 데이터의 위치를 지정하는 방법

주소 설계 시 고려 사항
- 표현의 효율성: 주소를 효율적으로 나타내야 함
- 사용의 편리성: 사용자에게 편리하도록 해야 함
- 주소공간과 기억공간의 독립성: 주소공간과 기억공간을 독립시킬 수 있어야 함

주소지정 방식의 종류
- 암시적 주소지정 방식 (Implied Addressing Mode)
  - 주소를 지정하는 필드가 없는 0번지 명령어에서 Stack의 Top 포인터가 가리키는 Operand를 암시하여 이용함
- 즉시적 주소지정 방식 (Immediate Addressing Mode)
  - Operand 부분에 데이터를 기억하는 방식
  - 레지스터의 값을 초기화할 때 주로 사용됨
  - 별도의 기억장치를 접근하지 않고 CPU에서 곧바로 자료를 이용
  - 메모리의 참조 횟수를 줄일 수 있으므로 실행속도가 가장 빠름
- 직접 주소지정 방식 (Direct Addressing Mode)
  - 명령어 주소 부분에 유효 주소 데이터가 있음
  - 명령어의 길이에 영향을 받으므로 표현할 수 있는 데이터 값의 범위가 제한적
- 간접 주소지정 방식 (Indirect Addressing Mode)
  - 명령문 내의 번지는 실제 데이터의 위치를 찾을 수 있는 번지가 들어 있는 장소를 표시
  - 인스트럭션의 길이가 짧고 제한되어 있어도 이것을 이용하여 긴 주소를 찾아갈 수 있음
- 계산에 의한 주소지정 방식
  - Operand부와 CPU의 특정 레지스터의 값이 더해져서 유효주소를 계산
  - 종류 (사용하는 레지스터의 종류에 따라 구분)

종 류	설 명
상대 주소 지정 방식 (Relative Addressing Mode)	• 유효주소: 명령어의 주소 부분 + Program Counter • 명령어 자신의 기억장소를 기준으로 하여 데이터의 위치를 지정하는 방식
베이스 레지스터 주소 지정 방식 (Base Register Addressing Mode)	• 유효주소: 명령어의 주소 부분 + Base Register • 프로그램을 재배치(Relocation)할 때 이용 • 다중 프로그래밍 기법에 많이 사용 • 베이스 레지스터 명령이 시작되는 최초의 번지를 기억하고 있는 레지스터
인덱스 레지스터 주소 지정 방식 (Indexed Addressing Mode)	• 유효주소: 명령어의 주소 부분 + Index Register

⑥ 리엔트란시 (Re-entrancy)
- 한 사람 이상의 사용자들이 동일한 명령어를 동시에 실행할 수 있도록 제공
- 인덱스 레지스터(Index Register)와 간접번지 방법(Indirect addressing)으로 이용

3.4 연산 (Operation)

논리연산과 산술연산
- 논리연산
  - 연산의 대상 및 결과가 '0'또는 '1'중 하나의 값을 취하는 연산 (비수치적인 연산)
  - 예: MOVE, NOT, AND, OR, 논리 SHIFT, ROTATE, COMPLEMENT, EXCLUSIVE OR 등
- 산술연산
  - 연산의 대상을 수치 데이터로 간주하고 행하는 연산 (수치적인 연산)
  - 예: ADD, SUBTRACT, MULTIPLY, DIVIDE, 산술 SHIFT 등

AND (Masking Operation)
- 비수치 데이터에서 마스크를 이용하여 불필요한 부분(일부분 혹은 전체)을 제거하기 위한 연산
- 삭제할 부분의 비트를 0과 AND시켜서 삭제

OR (Selective-Set)
- 두 개의 데이터를 섞거나 일부에 삽입하는데 사용되는 연산
- 특정 비트에 1을 세트(set)시키는 연산

XOR (비교, Compare)
- 자료의 특정 비트를 반전시키고자 하는 경우에 사용
- 비교(compare) 동작과 같은 동작을 하는 논리 연산
- 연산에서 overflow가 발생했을 경우 이것을 검출할 때 사용되는 논리 게이트

NOT (보수, Complement)
- 각 비트의 값을 반전시키는 연산
- 보수를 구할 때 사용

논리 Shift
- 왼쪽 또는 오른쪽으로 1 Bit씩 자리를 이동시키는 연산
- 데이터의 직렬 전송에 사용

Rotate
- 논리 Shift에서 밀려 나가는 비트의 값을 반대편 값으로 입력하는 연산
- 문자 위치 변환 시 사용

산술 Shift
- 부호 비트(Sign Bit)를 제외한 나머지 비트만 Shift (부호비트는 불변)
- 왼쪽으로 n Bit Shift: 원래 자료에 2^n을 곱한 값과 같음
- 오른쪽으로 n Bit Shift: 원래 자료를 2^n으로 나눈 값과 같음

Shift	수치 표현법	Padding Bit	
		음수	양수
Shift Left	부호화 절대치	Padding Bit : 0	모든 Padding Bit는 0
	1의 보수법	Padding Bit : 1	
	2의 보수법	Padding Bit : 0	
Shift Right	부호화 절대치	Padding Bit : 0	
	1의 보수법	Padding Bit : 1	
	2의 보수법	Padding Bit : 1	

※ Padding Bit : Shift에서 자리를 이동한 후 생기는 왼쪽 혹은 오른쪽 끝의 빈자리에 채워지는 비트

4 명령실행과 제어

4.1 마이크로 오퍼레이션 (Micro Operation)

마이크로 오퍼레이션 (동작)
- 명령을 수행하기 위해 CPU 내의 레지스터와 플래그의 상태 변환을 일으키는 작업
- 레지스터에 저장된 데이터에 의해서 이루어지는 동작
- 마이크로 오퍼레이션은 Clock 펄스에 기준을 두고 실행
- 동기 디지털 시스템에 내장되어 있는 모든 레지스터의 타이밍은 마스트 클록 발생기에 의하여 제어됨
- 제어 신호: 마이크로 오퍼레이션을 순서적으로 일어나게 하는데 필요한 신호
- 마이크로 사이클 타임(Micro Cycle Time): 마이크로 오퍼레이션 수행에 필요한 시간

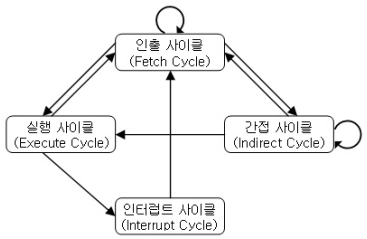
마이크로 사이클 타임 부여 방식

종 류	설 명
동기 고정식 (Synchronous Fixed)	• 마이크로 오퍼레이션 중에서 수행시간이 가장 긴 것을 정의한 방식 • 수행 시간이 가장 긴 마이크로 오퍼레이션의 사이클 타임을 클럭 주기로 정함
동기 가변식 (Synchronous Variable)	• 마이크로 오퍼레이션에 따라서 수행시간을 다르게 하는 것 • 각 마이크로 오퍼레이션의 사이클 타임이 현저한 차이를 나타낼 때 사용 • 중앙처리장치의 시간을 효율적으로 이용할 수 있음 • 마이크로 오퍼레이션에 대하여 서로 다른 사이클을 정의 할 수 있음
비동기식 (Asynchronous)	• 모든 마이크로 오퍼레이션에 대하여 서로 다른 마이크로 사이클 타임을 정의하는 방식

4.2 메이저 스테이트

메이저 스테이트 (메이저 상태, Major State)
- CPU가 어떤 일을 하고 있느냐에 따라 명령의 수행 단계를 4단계 (Fetch, Indirect, Execute, Interrupt)로 나눔

메이저 스테이트의 변화 과정



종 류	설 명
인출 단계 (Fetch Cycle)	• 주기억장치의 지정 장소(Address)로부터 명령을 읽어서 중앙처리장치에 가지고 오는 단계 • 인터럽트를 처리한 후 다음으로 전환해야 될 메이저 스테이트 • Fetch Cycle에서 일어나는 마이크로 오퍼레이션 T1 : MAR ← PC — • PC에 있는 번지를 MAR에 전송 T2 : MBR←M[MAR], PC ← PC+1 — • 메모리에서 MAR이 지정하는 위치의 값을 MBR에 전송 • 다음에 실행할 명령의 위치를 지정하기 위해 PC의 값을 1증가시킴 T3 : OPR ← MBR[OP], I ← MBR[I] — • 명령어의 Op-Code 부분을 명령 레지스터에 전송 • 명령어의 모드 비트를 플립플롭 I에 전송 T4 : F ← 1 또는 R ← 1 — • I가 0이면 F 플립플롭에 1을 전송하여 실행 사이클로 변하고, I가 1이면 R 플립플롭에 1을 전송하여 간접 사이클로 변함
간접 단계 (Indirect Cycle)	• 인스트럭션의 수행 시 유효 주소를 구하기 위한 메이저 상태 • 간접 단계(Indirect cycle) 동안에 기억장치로부터 오퍼랜드(데이터)의 번지(Address)를 인출

종 류	설 명
실행 단계 (Execute Cycle)	• 실제로 명령을 이행하는 단계 • Execute 단계에서는 Interrupt 요청신호를 나타내는 플래그 레지스터의 상태 변화를 검사하여 Interrupt 단계로 변천할 것인지 Fetch 단계로 변천할 것인지 판단함
인터럽트 단계 (Interrupt Cycle)	• 하드웨어로 실현되는 서브루틴의 호출이라고 볼 수 있음 • 인터럽트 발생 시 복귀주소(PC)를 저장시키고, 제어 순서를 인터럽트 처리 프로그램의 첫 번째 명령으로 옮기는 단계 • Interrupt Cycle에서 일어나는 마이크로 오퍼레이션 1. MBR(AD) ← PC, PC ← 0 2. MAR ← PC, PC ← PC+1 3. M ← MBR, IEN ← 0 4. F ← 0, R ← 0

주요 명령의 마이크로 오퍼레이션

• AND
- AC(누산기) 내용과 메모리 내용을 AND(논리곱) 연산하여 결과를 AC에 저장하는 연산 명령
- 마이크로 오퍼레이션

```
MAR ← MBR
MBR ← M(MAR)
AC ← AC AND MBR
```

• ADD
- AC와 메모리의 내용을 더하여 결과를 AC에 저장하는 연산 명령
- 마이크로 오퍼레이션

```
MAR ← MBR(ADDR)
MBR ← M(MAR)
AC ← AC + MBR
```

• LDA (Load to AC)
- 메모리의 내용을 AC로 가져오는 명령
- 마이크로 오퍼레이션

```
MAR ← MBR(AD)
MBR ← M(MAR), AC ← 0
AC ← AC + MBR
```

• STA (Store AC)
- AC의 내용을 메모리에 저장하는 명령
- 마이크로 오퍼레이션

```
MAR ← MBR(AD)
MBR ← AC
M ← MBR
```


• BUN (Branch UNconditionally)
- PC에 특정한 주소를 전송하여 실행명령의 위치를 변경하는 무조건 분기 명령
- 마이크로 오퍼레이션

```
PC ← MBR(AD)
```

• BSA (Branch and Save Return Address)
- 복귀주소를 저장하고 부 프로그램을 호출하는 명령

• ISZ (Increment and Skip if Zero)
- 메모리의 값을 읽고 그 값을 1 증가시킨 후 음수에서 시작한 그 값이 0이면 현재 명령을 건너 띄고 다음 명령으로 이동
- 마이크로 오퍼레이션

```
MAR ← MBR(AD)
MBR ← M
MBR ← MBR + 1
M ← MBR, IF(MBR=0) THEN (PC←PC+1)
```

4.3 제어장치와 마이크로프로그램

제어 데이터의 종류

• 각 메이저 스테이트 사이의 변천을 제어하는 제어 데이터
• 중앙처리장치의 제어점을 제어하는데 필요한 제어 데이터
• 인스트럭션의 수행순서를 결정하는데 필요한 제어 데이터

제어 데이터가 될 수 있는 것

• 연산자의 종류
• 인스트럭션의 주소지정방식
• 연산 결과에 대한 상태 플래그 내용

제어장치(제어기)의 구현

• 고정 배선 방식 (Hard-wired Control Unit)
- 속도가 빠름
- 마이크로프로그램 방식에 비해 비쌈
• 마이크로프로그램 방식 (Micro Programmed Control Unit)
- 마이크로프로그램을 이용한 방식
- 하드와이어드 방식에 비해 속도가 느림

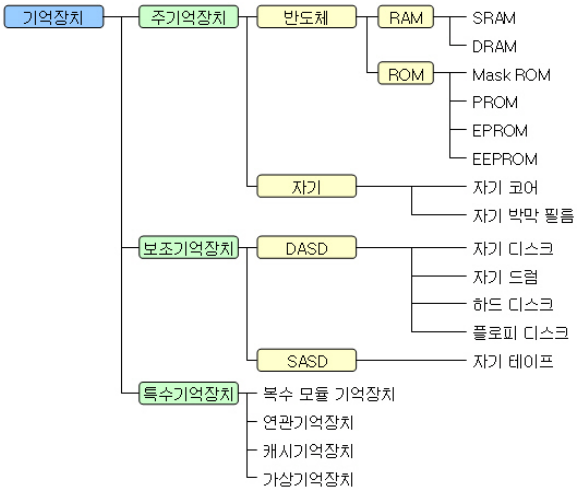
마이크로프로그램 (Microprogram)

• 어떤 명령을 수행할 수 있도록 된 일련의 제어 워드가 특수한 기억 장치 속에 저장된 것으로 각종 제어 신호를 발생시킴
• 마이크로프로그램이 저장되는 제어 메모리는 ROM이 주로 사용되고 사용자가 변경시킬 수 없음
• 별도의 번역과 RAM으로의 접근이 필요하지 않기 때문에 일반적인 소프트웨어와 구별하여 펌웨어(Firmware)라고도 함

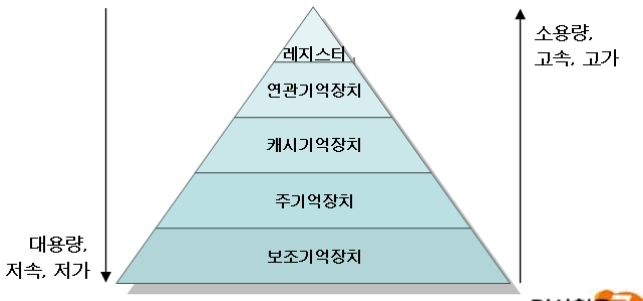
5 기억장치

5.1 기억장치의 개요

기억장치의 분류



기억장치 계층 구조



기억장치의 구분

• 내용의 보존 여부
- 파괴성 메모리 (Destructive Memory)
- 비파괴성 메모리: 판독 후에도 저장된 내용이 그대로 유지됨
• 전원 단절 시 내용 소멸 여부
- 휘발성 메모리 (Volatile Memory)
- 비휘발성 메모리: 전원이 단절되더라도 기억된 정보가 보존되는 메모리

5.2 주기억장치

주기억장치 개요

• CPU가 직접 접근하여 처리할 수 있는 기억장치
• 현재 수행되는 프로그램과 데이터를 저장
• 주기억장치에 사용되는 양극 소자나 MOS형 기억 소자는 보조기억장치에 비해 동작속도가 빠르고, 가격이 비쌈
• 주기억장치는 Main Storage를 의미함
• 주기억장치 밴드폭 (Bandwidth)
- 하드웨어의 특성상 주기억장치가 제공할 수 있는 정보 전달능력의 한계
• 주기억장치 종류: ROM, RAM

ROM

• Read Only Memory
• 기억된 내용을 임의로 변경 시킬 수 없음 (Read만이 가능)
• 전원이 꺼져도 기억된 내용이 지워지지 않는 비휘발성 메모리
• 마이크로프로그램을 저장하는 제어 메모리는 주로 ROM 메모리를 사용함
• 주로 기본 입·출력 시스템(BIOS), 자가 진단 프로그램(POST) 같이 변경 가능성이 희박한 시스템 소프트웨어를 기억시키는데 이용
• ROM의 종류

종 류	설 명
Mask ROM	• 반도체 공장에서 내용이 기입됨 • 읽기 전용으로 내용을 변경할 수 없음
PROM	• 사용자가 한번만 내용을 기입을 할 수 있으나, 지울 수 없는 것
EPROM	• 이미 기억된 내용을 자외선을 이용하여 지우고, 다시 사용할 수 있는 메모리
EEPROM	• 이미 기억된 내용을 전기적인 방법을 이용하여 지우고, 다시 사용할 수 있는 메모리

RAM
- Random Access Memory
- 자유롭게 읽고 쓸 수 있는 기억장치
- RAM의 종류

구 분	DRAM (Dynamic RAM, 동적 램)	SRAM (Static RAM, 정적 램)
구성 소자	콘덴서	플립플롭
특징	· 각 비트(Bit)를 전하(charge)의 형태로 저장하며, 주기적으로 재충전이 필요함 · 미소의 콘덴서에 전하를 충전하는 형태의 원리를 이용하는 메모리	· 전원이 공급되는 동안에는 기억 내용이 유지됨
전력 소모	적음	많음
접근 속도	느림	빠름
직접도	높음	낮음
가격	저가	고가
용도	일반적인 주기억장치	캐시 메모리

자기 코어
- 자기 코어는 중심을 통과하는 전선에 흐르는 전류의 방향에 따라 1혹은 0의 값을 가짐
- 전류 일치 기술(coincident-current technique)에 의하여 기억장소를 선별하는 기억장치
- 전자계산기 메모리에서 지움성 읽음(Destructive Read-Out) 성질을 갖고 있음

반도체 기억소자 구성
- RAM / ROM의 용량계산법

$$\text{기억장치 용량} = 2^{\text{워드의 수}} \times \text{워드의 크기}$$

- 워드의 수 = 입력 번지선의 수 = 주소선의 수 = MAR = PC
- 워드의 크기 = 출력 데이터선의 수 = Data Bus의 비트 수 = MBR = DR = IR
- 예제
  - 기억용량이 1MByte일 때 필요한 주소선의 수?
    ⇨ 워드의 크기에 대한 언급이 없으면 워드의 크기로 1Byte로 보면 됨
    1MByte = 220이므로 20개의 주소선이 필요함
  - 메인 메모리의 용량이 1,024K × 24Bit 일 때, MAR과 MBR의 길이는 각각 몇 비트?
    ⇨ 1,024K × 24Bit 는 용량이 1,024K 워드이고, 워드 길이가 24Bit이므로
    워드의 크기 = MBR = 24, 1,024K = $2^{10} \times 2^{10}$($\because$ K = 2^{10} = 1,024)
    = 2^{20} ∴ MAR = 20
  - 컴퓨터의 메모리 용량이 16K×32bit라 하면 MAR(Memory Address Register)와 MBR(Memory Buffer Register)은 각각 몇 비트?
    ⇨ 16K = 16 × 2^{10} = $2^4 \times 2^{10}$ = 2^{14} ∴ MAR = 14
    용량이 16K×32bit이므로, 워드의 크기 = MBR = 32

5.3 보조기억장치

자기 테이프 (Magnetic Tape)
- 자기 테이프는 주소의 개념을 사용하지 않고, 처음부터 차례대로 처리하는 순차처리(SASD)만 할 수 있는(랜덤 처리가 되지 않음) 대용량 저장 매체
- 대량의 자료를 장시간 보관하는데 가장 유리한 장치
- 자기 테이프 구조

- 관련 용어
  - 블록 레코드(물리 레코드)
    · 테이프에서 갭(gap)과 갭 사이에 존재하는 레코드
    · 1개 이상의 논리 레코드의 집합으로 입·출력 단위
  - IBG(Inter Block Gap): 블록과 블록 사이의 갭
  - 블록 팩터(Block Factor)
    · 하나의 블록 내에 통합되어 있는 논리 레코드의 개수
    · 블록 팩터 = 블록 크기 / (논리)레코드 크기
    예) 자기 테이프 Record 크기가 80자로서 블록(Block)의 크기가 2,400자일 경우 블록 팩터(Block Factor)?
    블록 팩터 = 블록 크기 / 레코드 크기 = 2,400 / 80 = 30

자기 드럼 (Magnetic Drum)
- 원통 표면에 Track과 Sector를 구성하고, 각 Track마다 고정된 R/W Head를 두고 있음
- 자기 디스크에 비해 속도가 빠름
- 자기 드럼은 순차, 비순차(직접) 처리가 모두 가능한 DASD(Direct Access Storage Device)방식으로 데이터를 처리함
- 기억 용량 = 드럼 표면의 트랙당 셀 수 × 트랙 수

자기 디스크 (Magnetic Disk)
- 자성 물질을 입힌 금속 원판을 여러 장 겹쳐서 만든 기억매체
- 용량이 크고, 접근속도가 빠름
- 순차, 비순차(직접) 처리가 모두 가능한 DASD(Direct Access Storage Device)방식으로 데이터를 처리함
- 자기 디스크는 디스크 번호, 디스크 표면 번호, 트랙 번호, 섹터 번호를 표현하는 번지 비트를 가지고 디스크의 기억공간을 Access함

5.4 특수기억장치

연관기억장치 (Associative Memory)
- 연관기억장치의 개념
  - 연관기억장치는 기억장치에서 자료를 찾을 때 주소에 의해 접근하지 않고, 기억된 내용의 일부를 이용하여 Access할 수 있는 기억장치로, CAM(Content Addressable Memory)이라고도 함
  - 기억된 정보의 일부분을 이용하여 원하는 정보가 기억된 위치를 알아낸 후 그 위치에서 나머지 정보에 접근하는 기억장치
- 연관기억장치의 특징
  - 주소에 의해서만 접근이 가능한 기억장치보다 정보 검색이 신속함
  - 캐시 메모리나 가상 메모리 관리 기법에서 사용하는 Mapping Table에 사용됨
  - 외부의 인자와 내용을 비교하기 위한 병렬 판독 논리회로를 갖고 있기 때문에 하드웨어 비용이 증가함
  - 메모리의 내용으로 접근(access) 할 수 있는 메모리
  - 기억된 정보의 일부분을 이용하여 원하는 정보가 기억된 위치를 알아낸 후 나머지 정보에 접근함
  - 기억된 여러 개의 자료 중에서 주어진 특성을 가진 자료를 신속히 찾을 수 있음
  - 비파괴적으로 읽을 수 있어야 함
- 연관기억장치에 사용되는 기본요소
  - 일치 지시기: 내용의 일부가 같은 워드를 찾았으면 1로 세트함
  - 마스크 레지스터: 비교할 비트를 정해 1로 세트함
  - 검색 데이터 레지스터: 비교할 내용이 들어 있음

복수 모듈 기억장치
- 독자적으로 데이터를 저장할 수 있는 기억장치 모듈을 여러 개 가진 기억장치
- 메모리 인터리빙 (Memory Interleaving)
  - 여러 개의 독립된 모듈로 이루어진 복수 모듈 메모리와 CPU 간의 주소 버스가 한 개로만 구성되어 있으면 같은 시각에 CPU로부터 여러 모듈들로 동시에 주소를 전달할 수 없기 때문에, CPU가 각 모듈로 전송할 주소를 교대로 배치한 후 차례대로 전송하여 여러 모듈을 병행 접근하는 기법
  - CPU가 버스를 통해 주소를 전달하는 속도는 빠르지만 메모리 모듈의 처리 속도가 느리기 때문에 병행 접근이 가능
  - 기억장치의 접근 시간을 효율적으로 높일 수 있음
  - 캐시 기억장치, 고속 DMA 전송 등에서 많이 사용
  - 각 모듈을 번갈아가면서 접근(Access)할 수 있음
  - Instruction의 빠른 처리속도를 위해 중앙처리장치의 속도와 기억장치의 속도를 유효 Cycle동안 병행 실행한다는 것과 관련 있는 것
  - 중앙처리장치와 기억장치 사이에 실질적인 대역폭(bandwidth)을 늘리기 위한 방법
- 디스크 인터리빙
  - 독립된 디스크(disk)를 2개 이상 나누어 연결하고 독립된 디스크(disk)를 번갈아 가면서 연속적으로 액세스가 이루어지도록 구현하는 방법

캐시 메모리 (Cache Memory)
- 캐시 메모리(Cache Memory)의 개념
  - 캐시 메모리는 CPU의 처리 속도와 주기억장치의 접근 속도 차이를 줄이기 위해 사용하는 고속 Buffer Memory임
  - 중앙 처리장치(CPU)의 속도와 주기억장치의 속도차가 클 때 명령어(Instruction)의 수행 속도를 중앙 처리 장치의 속도와 비슷하도록 하기 위하여 사용하는 메모리
  - 성능을 향상시키기 위하여 주기억 장치와 CPU 레지스터 사이에서 데이터를 이동시키는 중간 버퍼로 작용하는 기억장치
- 적중률 (hit ratio)
  - 캐시에 찾는 내용이 있을 확률
- 참조의 국한성 (locality of reference)
  - CPU가 기억장치를 접근할 때는 일부 특정 위치를 계속 참조한다는 이론
- 매칭 (matching)
  - 내용의 일부를 이용하여 자료를 찾는 기억장치에서 내용이 같은지 비교하는 것

가상기억장치 (Virtual Memory)
- 가상기억장치의 특징
  - 주기억장치를 확장한 것과 같은 효과를 제공
  - 실제로는 보조기억장치를 사용하는 방법
  - 사용자가 프로그램 크기에 제한 받지 않고 실행이 가능
  - 컴퓨터속도는 문제시되지 않음
  - 주소공간의 확대(용량의 확대)가 가장 큰 목적
  - 사용할 수 있는 보조기억장치는 DASD이어야 함
  - 가상기억공간의 구성은 프로그램에 의해서 수행됨
  - 보조기억장치는 자기 디스크를 많이 사용함
  - 보조기억장치의 접근이 자주 발생되면 컴퓨터 시스템의 처리 효율이 저하될 수 있음
  - 주기억장치와 보조 기억장치가 계층 기억 체제를 이루고 있음
  - 하드웨어에 의한 것이 아니라 소프트웨어에 의해 실현됨
- 기억장치의 관리전략 방법
  - 반입(Fetch) 전략
  - 배치(Placement) 전략
  - 교체(Replacement) 전략
- 가상기억장치의 관리 기법
  - 페이징(Paging) 기법
  - 세그먼트(Segmentation) 기법
- 주소 매핑 (주소 변환)
  - 가상주소를 실기억주소로 변환하는 작업
  - 보조기억장치에 보관 중이던 프로그램을 실행하기 위하여 각 Page들을 주기억장치에 Load했다 하더라도 프로그램을 구성하는 각 기계명령에 포함된 주소는 가상주소로 남아 있기 때문에, CPU에서 주기억장치를 Access하기 위해서는 가상주소를 실주소로 변환해야 함

6 입력 및 출력

6.1 입·출력의 기본

입·출력 장치의 구성

- 입·출력 제어장치:
  - 입·출력 장치와 컴퓨터 사이의 자료 전송을 제어하는 장치
- 입·출력 인터페이스
  - 메모리나 CPU 레지스터와 같은 내부 저장 장치와 외부 I/O 장치 간에 2진 정보를 전송하는 방법을 제공
- 입·출력 버스
  - 주기억장치와 입·출력 장치 사이의 데이터 전송을 위해 모든 주변장치의 인터페이스에 공통으로 연결된 버스

입·출력장치의 종류

- 입력장치: OMR, OCR, MICR, 스캐너, 마우스, 라이트 펜, 디지타이저, 키보드 등
- 출력장치: 모니터, 프린터, 플로터 등
- 보조기억장치(입·출력 겸용 장치): 자기 테이프, 자기 디스크, 자기 드럼 등

입·출력 장치와 기억장치의 동작 차이점

구 분	입·출력 장치	기억장치
동작의 속도 (가장 중요 항목)	느림	빠름
동작의 자율성	타율 / 자율	타율
정보의 단위	Byte(문자)	Word
착오 발생률	많음	적음

비동기 데이터 전송

- 핸드셰이킹(Handshaking) 방식
- 스트로브 펄스(Strobe Pulse) 방식

버퍼링과 스풀링

- 버퍼링 (Buffering)
  - 저속의 입·출력 장치와 고속의 CPU의 처리 속도 차이를 개선하기 위한 방법
  - 주기억 장치의 일부 공간을 버퍼공간으로 할당하여 처리할 데이터를 임시 기억하여 처리하는 방식
  - 한번 나와 있는 데이터가 CPU에서 여러 번 사용함
  - 버퍼의 위치는 주기억장치
  - 많은 데이터를 주기억장치에서 한 번에 가져 나감
  - 데이터를 주기억장치에서 읽어 내거나 주기억장치에 저장하기 위해 임시로 자료를 기억하는 공간
- 스풀링 (Spooling)
  - I/O 효율을 높이기 위해 I/O의 내용을 디스크 등에 모아두었다가 처리하는 방식
  - 디스크 일부를 매우 큰 버퍼처럼 사용
  - 스풀의 위치는 보조기억장치
  - 어떤 작업의 입/출력과 다른 작업의 계산을 병행 처리하는 기법

6.2 입·출력 제어방식

CPU의 관여 여부에 따라 나누어짐

- CPU 관여 O: Program에 의한 I/O, Interrupt에 의한 I/O
- CPU 관여 X: DMA에 의한 I/O, Channel에 의한 I/O

Programmed I/O

- 프로그램을 통한 입·출력 방식에서 입·출력 장치 인터페이스에 포함되어야 하는 하드웨어
- 데이터 레지스터
- 장치의 동작 상태를 나타내는 Flag
- 장치 번호 디코더

Interrupt I/O

- CPU가 계속 flag를 검사하지 않고 데이터가 준비되면 인터페이스가 CPU에 입·출력을 요구하고, 입·출력 전송이 완료되면 CPU는 수행 중이던 프로그램으로 되돌아가서 수행을 재개하는 입·출력 방식
- CPU가 계속 Flag를 검사하지 않아도 되기 때문에 Programmed I/O보다 효율적임

DMA에 의한 I/O

- DMA (Direct Memory Access)
  - 데이터 입출력 전송이 CPU를 통하지 않고 직접 주기억장치와 주변장치 사이에서 수행되는 방식
  - CPU를 거치지 않고 메모리와 입·출력장치가 직접 통신하기 때문에 CPU에 부하가 증가되지 않음
  - CPU와 DMA 제어기는 메모리와 버스를 공유함
  - DMA는 기억장치와 주변장치 사이의 직접적인 데이터 전송을 제공
  - DMA는 블록으로 대용량의 데이터를 전송할 수 있음
  - 자료를 입·출력 할 때 가장 효과적인 방법
  - 메모리 장치외의 통신에서 CPU보다 우선권을 가지고 있음
  - 보다 빠른 데이터의 전송이 가능함
  - 데이터 대량전송(burst transfer) 및 사이클 스틸링(cycle stealing)과 관계있는 것
- 사이클 스틸링 (Cycle Stealing)
  - DMA 제어기가 한 번에 한 데이터 워드를 전송하고, 버스의 제어를 CPU에게 돌려주는 방법
  - Cycle Steal을 이용하면 입·출력 자료의 전송을 빠르게 처리해 주는 장점이 있음
  - 사이클 스틸은 DMA 인터페이스에 의해서 이루어짐
  - 사이클 스틸(Cycle Steal)과 인터럽트(Interrupt)의 차이점

사이클 스틸 (Cycle Steal)	인터럽트 (Interrupt)
CPU 상태를 보존할 필요가 없음	CPU(중앙처리장치) 상태를 보존해야 함
잠시 CPU가 쉼	CPU는 인터럽트를 처리해야 함
아무 사이클이나 상관없이 훔치는 것이 가능함	항상 실행 사이클 이후에만 인터럽트가 인지됨

채널에 의한 I/O

- 채널 (Channel)
  - 신호를 보낼 수 있는 전송로(입·출력 장치와 주기억장치를 연결하는 중개 역할을 담당하는 부분)
  - 입·출력은 DMA 방법으로도 수행함
  - 입·출력 수행 중 어떤 오류조건에서 중앙처리장치에 인터럽트를 걸 수 있음
  - CPU의 명령을 받고 입·출력 조작을 개시하면 CPU와는 독립적으로 조작함
- 채널의 종류

종 류	설 명
Selector Channel (선택 채널)	• 입·출력이 실제로 일어나고 있을 때는 제어기가 임의의 시점에서 볼 때 마치 입·출력 장치의 전용인 것처럼 운영되는 • 특정한 한 개의 장치를 독점하여 입·출
Multiplexer Channel (다중 채널)	• 동시에 여러 개의 입출력 장치를 제어 있는 채널 • 저속 입·출력장치 제어
Block Multiplexer Channel	• 동시에 여러 개의 입·출력 장치를 제어 있는 채널 • 고속 입·출력장치 제어

6.3 인터럽트

인터럽트(Interrupt)의 정의

- 인터럽트는 프로그램을 실행하는 도중에 예기치 않은 상황이 발생할 경우, 현재 실행 중인 작업을 즉시 중단하고 발생된 상황을 우선 처리한 후 실행 중이던 작업으로 복귀하여 계속 처리하는 것을 말함
- Computer system에 예기치 않은 일이 발생했을 때 제어프로그램에게 알려주는 것
- 인터럽트는 외부 인터럽트, 내부 인터럽트, 소프트웨어 인터럽트로 분류하는데, 외부나 내부 인터럽트는 CPU의 하드웨어에서의 신호에 의해 발생하고 소프트웨어 인터럽트는 명령어의 수행에 의해 발생함

인터럽트의 종류 및 발생원인

종 류		설 명
외부 인터럽트	전원 이상 인터럽트 (Power Fail Interrupt)	• 정전이 되거나 전원 이상이 있는 경우
	기계 착오 인터럽트 (Machine Check Interrupt)	• 하드웨어의 기능적인 오류 동작이 발생한 경우
	외부 신호 인터럽트 (External Interrupt)	• 타이머에 의해 규정된 시간(Time Slice)을 알리는 경우 • 키보드로 인터럽트 키를 누를 경우 • 외부 장치로부터 인터럽트 요청이 있는 경우
	입·출력 인터럽트 (Input-Output Interrupt)	• Data의 오류나 이상 현상이 발생한 경우
내부 인터럽트	불법 명령어 사용 인터럽트 (Use Bad Command Interrupt)	• 정의되지 않은 명령이나 불법적인 명령을 사용했을 경우 혹은 보호되어 있는 기억공간에 접근하는 경우 발생하는 인터럽트
	프로그램 검사 인터럽트 (Program Check Interrupt)	• 프로그램의 실행 오류로 발생 - 0에 의한 나눗셈 - overflow 또는 underflow시 - 불법적인 명령 - 보호 영역 내의 메모리 어드레스를 Access 하는 경우
	SVC 인터럽트 (Supervisor Call Interrupt)	• 사용자 프로그램에서 SVC 명령을 호출하였을 경우 발생하는 인터럽트

인터럽트의 동작 원리

- 인터럽트 수행 순서
  - 인터럽트 요청 신호 발생(CPU에게 인터럽트 요청)
  - 현재 수행중인 명령을 완료하고, 상태를 기억시킴(현재 작업 중인 주소를 메모리에 저장)
  - 어느 장치가 인터럽트를 요청했는지 찾음(인터럽트 인지신호 발생)
  - 인터럽트 취급 루틴을 수행(벡터 인터럽트 처리)
  - 보존한 프로그램 상태를 복귀(리턴에 의한 복귀)
- 인터럽트 발생 시 CPU가 확인할 사항
  - 프로그램 카운터 내용
  - 상태 조건 내용 (PSW)
  - 사용한 모든 레지스터 내용

요청한 인터럽트를 처리하기 위해서 이전 프로그램의 상태 보존이 필요한 경우

- Cache memory에서 캐시 miss나 가상 메모리 시스템에서 page fault가 발생한 경우

인터럽트 반응시간 (interrupt response time)
- 인터럽트 요청신호를 발생한 후부터 인터럽트 취급 루틴의 수행이 시작될 때까지의 시간

인터럽트 발생 시 운영체제가 가장 먼저 하는 일
- 현재까지의 모든 프로그램 상태를 저장

벡터 인터럽트 (Vectored Interrupt)
- 인터럽트 발생 시 프로세서의 인터럽트 서비스가 특정의 장소로 점프(분기)하여 서비스 할 수 있게 함
- 인터럽트를 발생한 장치가 프로세서에게 분기할 곳의 정보를 제공해 주는 것
- 인터럽트 벡터(= 벡터 인터럽트)에 필수적인 것: 분기번지
- 하드웨어 신호에 의하여 특정 번지의 서브루틴을 수행하는 것

Program Counter (PC)
- 인터럽트 처리 루틴에서 반드시 사용되는 레지스터
- 인터럽트 발생 시에 반드시 보존되어야 하는 레지스터

인터럽트 우선순위

전원이상(Power Fail) ➡ 기계 이상 ➡ 외부 신호 ➡ 입·출력 ➡ 명령어 잘못 ➡ 프로그램 검사 ➡

높음 <---> 낮

인터럽트 우선순위 판별 방법
- 소프트웨어적인 방법: Polling
- 하드웨어적인 방법: 직렬 우선순위 부여방식(Daisy-Chain), 병렬 우선순위 부여 방식

폴링(polling) 방법
- 인터럽트 우선순위 가운데 소프트웨어적 처리기법
- 인터럽트 요청신호 플래그를 차례로 검사하여 인터럽트의 원인을 판별하는 방식

데이지 체인 (Daisy-Chain)
- 직렬 우선순위 부여 방식
- 인터럽트를 발생하는 모든 장치들을 직렬로 연결
- 인터럽트 처리 과정 중 하드웨어를 이용하여 우선순위를 결정하는 장치
- 벡터에 의한 인터럽트 처리 방법
- 인터럽트 된 모든 장치들은 벡터를 동시에 보낼 수 있음
- 인터럽트 신호 선을 공유하면서 연결 순서에 따라 우선순위가 결정되는 것

병렬 우선순위 부여 방식
- 인터럽트가 발생하는 각 장치를 개별적인 회선으로 연결
- 마스크 레지스터(Mask Register)를 갖고 있음
- 마스크 레지스터는 우선순위가 높은 것이 서비스 받고 있을 때 우선순위가 낮은 것을 비활성화 시킬 수 있음
- 우선순위는 레지스터의 Bit의 위치에 따라 결정될 수 있음

하드웨어 우선순위 인터럽트의 특징
- 응답속도가 빠름

7 병렬 컴퓨터 구조

병렬 처리 개념
- 병렬 처리는 폰노이만 컴퓨터 구조의 순차처리에 반대되는 구조로, I/O 채널 또는 Processor와 같은 다수의 Processor(처리기)에서 동시에 여러 작업(Process)을 처리하는 것
- 다수의 프로세서를 연결하여 동시에 수행을 하게 함으로서 연산 속도를 향상시키고, 다수의 프로세서를 관리하기 위한 시스템

병렬 컴퓨터의 분류
- 팽(Feng)의 분류
  - 컴퓨터의 구조를 병렬 수행의 정도에 따라 분류한 방식
  - WSBS, WPBS, WSBP, WPBP로 분류
- 플린(Flynn)의 분류
  - 플린은 명령 흐름과 자료 흐름을 고려하여 병렬 컴퓨터 구조를 분류함
  - 처리기가 동시에 수행하는 명령과 데이터의 수에 따라 구분하는 방법
  - 플린의 4가지 분류
    - SISD (Single Instruction stream Single Data stream)
    - SIMD (Single Instruction stream Multiple Data stream)
    - MISD (Multiple Instruction stream Single Data stream)
    - MIMD (Multiple Instruction stream Multiple Data stream)

병렬처리기의 종류
- Pipeline processor
- Vector processor
- Multi processor

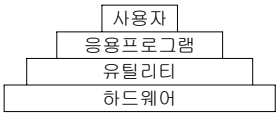
배열 처리기 (Array Processor)
- PE(Processing element)라고 불리는 다수의 연산기를 갖는 형태로 PE들을 동기 적으로 병렬처리를 수행하는데 동시에 같은 기능을 수행하도록 되어 있음
- 명령 해독 및 제어는 제어장치가 하고, PE들은 명령 해독 능력이 결여된 수동적 장치로서 명령 처리만 함
- 벡터 계산이나 행렬 계산에 적합

운영체제

1 운영체제의 개념 및 종류

1.1 운영체제의 개념 및 종류

운영체제의 개념
- 운영체제는 컴퓨터 시스템의 자원들을 효율적으로 관리
- 사용자가 컴퓨터를 편리하고 효과적으로 사용할 수 있도록 환경을 제공
- 사용자와 컴퓨터 간의 인터페이스로서 동작하는 시스템 소프트웨어
- 운영체제의 계층은 하드웨어와 유틸리티 사이임



운영체제의 목적
- 사용자와 컴퓨터 간의 인터페이스 제공
- 자원의 효율적인 운영 및 자원 스케줄링
- 데이터 공유 및 주변장치 관리
- 처리 능력 및 신뢰성 향상
- 응답시간 단축, 반환시간 단축
- 시스템의 오류를 처리

운영체제의 성능 평가 기준
- 처리능력(Throughput)
- 반환시간(Turnaround time)
- 사용 가능도(Availability)
- 신뢰도(Reliability)

운영체제의 성능 평가 방법
- 벤치마크(Benchmark): 프로그램을 수행하여 성능을 측정
- 시뮬레이션(Simulation): 시스템의 내부 특성을 프로그램으로 표현하여 성능 측정
- 수학적 모델: 수학적 공식으로 성능을 측정

운영체제의 운용기법 종류

종 류	설 명
일괄 처리 시스템 (Batch Processing System)	• 초기의 컴퓨터 시스템에서 사용된 형태로, 일정량의 데이터를 모아서 한꺼번에 처리하는 방식 • 컴퓨터 시스템을 효율적으로 사용 • 반환 시간이 늦지만 하나의 작업이 모든 자원을 독점하므로 CPU 유휴 시간을 줄임
다중 프로그래밍 시스템 (Multi Programming System)	• 하나의 CPU와 주기억장치를 이용하여 여러 개의 프로그램을 동시에 처리하는 방식 • CPU의 사용률과 처리량이 증가
시분할 시스템 (Time Sharing System)	• 여러 명의 사용자가 사용하는 시스템에서 컴퓨터가 사용자들의 프로그램을 번갈아 가며 처리해 줌으로써 각 사용자에게 독립된 컴퓨터를 사용하는 느낌을 받음 • 시스템의 전체 효율은 좋아지나 개인별 사용자 입장에서는 반응 속도가 느려질 수 있음 • 긴 작업에 대한 응답 시간을 최소한으로 줄이는 것을 목적 • 각 사용자는 기억 장치에 독립된 프로그램 • 라운드 로빈(Round Robin)방식을 사용
실시간 처리 시스템 (Real Time Processing System)	• 데이터 발생 또는 데이터에 대한 처리 요구가 있는 즉시 처리하여 응답해 주는 시스템 • 주어진 적정 시간 내에 답을 주어야 함 • 우주선 운행이나 레이더 추적기, 은행의 온라인 업무 등 시간에 제한을 두고 수행되어야 하는 작업에 사용

운영체제의 운용 기법 발달 과정

일괄처리시스템 ⇨ ·다중프로그램 ·시분할시스템 ·다중처리시스템 ·실시간처리시스템 ⇨ 다중모드시스템 ⇨ 분산처리시스템

1.2 시스템 소프트웨어의 종류

시스템 소프트웨어(System Software)

- 시스템 전체를 작동시키는 프로그램으로
- 프로그램을 주기억 장치에 적재시키거나 인터럽트 관리, 장치 관리, 언어 번역 등의 기능을 담당
- 대표적인 프로그램으로 운영체제가 있으며, 그 외에는 언어 번역 프로그램, 매크로 프로세서, 링커, 라이브러리, 로더 등이 있음

시스템 소프트웨어의 구성

- 제어 프로그램
  - 감시 프로그램 (Supervisor Program)
  - 작업 제어 프로그램 (Job Control Program)
  - 데이터 관리 프로그램 (Data Management Program)
- 처리 프로그램
  - 언어 번역 프로그램(Language Translate Program): 어셈블러, 컴파일러, 인터프리터
  - 서비스 프로그램(Service Program): 연결 편집기, 라이브러리
  - 문제 프로그램(Problem Program): 사용자가 작성한 프로그램

어셈블리어와 어셈블러

- 어셈블리어(Assembly Language)
  - 어셈블리어는 사용자가 이해하기 어려운 기계어 대신에 명령 기능을 쉽게 연상할 수 있는 기호를 기계어와 1:1로 대응시켜 코드화한 기호언어
  - 프로그램에 기호화된 명령 및 주소를 사용
  - 어셈블리어의 기본 동작은 동일하지만 CPU마다 사용되는 어셈블리어가 다를 수 있음
  - 기계어와 비교하여 읽기 쉽고 프로그램에 데이터를 사용하기 쉬움
  - 기계어로 번역하는 과정이 필요
- 어셈블러(Assembler)
  - 어셈블러는 어셈블리어로 작성된 원시 프로그램을 기계어로 된 목적 프로그램으로 번역하는 언어 번역 프로그램
  - 단일 패스 어셈블러와 이중 패스 어셈블러가 있음
  - 두개의 Pass로 구성하면 기호를 정의하기 전에 사용할 수 있어 프로그램 작성이 용이함

컴파일러 (Complier)

- 고급 언어로 작성된 프로그램 전체를 목적 프로그램으로 번역한 후 링킹 작업을 통해 실행 가능한 프로그램을 생성함
- 번역 시간이 오래 걸리지만 실행 속도가 빠름
- 사용언어: FORTRAN, COBOL, C, C++ 등이 있음

인터프리터 (Interpreter)

- 한줄 단위로 번역과 실행을 함
- 프로그램이 직접 실행되므로 목적 프로그램이 생성되지 않음
- 번역 속도는 빠르지만 실행 속도가 느림
- 사용언어: BASIC, LISP, APL 등이 있음

링커 (Linker)

- 언어 번역 프로그램이 생성한 목적 프로그램과 라이브러리, 또 다른 실행 프로그램 등을 연결하여 실행 가능한 로드 모듈을 만드는 시스템 소프트웨어이며 연결 편집기라고도 함

로더 (Loader)

- 프로그램을 실행시키기 위해 보조기억장치로부터 컴퓨터 주기억장치에 프로그램을 적재하는 시스템 소프트웨어
- 로더의 기능
  - 할당(Allocation): 프로그램을 실행시키기 위해 기억장치 내에 옮겨 놓을 공간을 확보하는 기능
  - 연결(Linking): 프로그램을 할당된 주소에 연결하는 기능
  - 재배치(Relocation): 디스크 등의 보조기억장치에 저장된 프로그램이 사용하는 주소들을 할당된 기억 장소의 실제 주소로 배치시키는 기능
  - 적재(Loading): 프로그램을 할당된 기억 공간에 실제로 옮기는 기능
- 절대 로더 (Absolute Loader)
  - 목적 프로그램을 기억 장소에 적재시키는 기능만 수행하는 로더
  - 할당 및 연결 작업은 프로그래머가 프로그램 작성 시 수행하며, 재배치는 언어 번역 프로그램이 담당
- 로더의 실행 순서
  - 할당 (Allocation)→연결 (Linking)→재배치 (Relocation)→적재 (Load)

매크로 (Macro)

- 프로그램 작성 시 한 프로그램 내에서 동일한 코드가 반복될 경우 반복되는 코드를 한번만 작성하여 특정이름으로 정의한 후 정의된 이름이 사용될 때 마다 작성된 코드를 삽입해서 실행
- 개방 서브루틴이라고 함
- 매크로 정의 내에 또 다른 매크로를 정의할 수 있음

매크로 프로세서(Macro Processor) 기능

- 매크로 정의 인식
- 매크로 정의 저장
- 매크로 호출 인식
- 매크로 호출 확장

2 프로세스 관리

2.1 프로세스의 개요

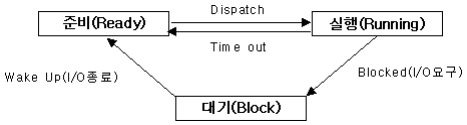
프로세스의 정의

- 실행중인 프로그램
- 실기억장치에 저장된 프로그램
- 프로세서(CPU)가 할당되는 실체
- 프로시저가 활동 중인 실체
- PCB를 가진 프로그램
- 비동기적 행위를 일으키는 주체
- 운영체제가 관리하는 실행 단위

프로세스 제어 블록 (PCB, Process Control Block)

- PCB는 운영체제가 프로세스에 대한 중요한 정보를 저장해 놓는 곳
- 각 프로세스가 생성될 때마다 고유의 PCB가 생성되고, 프로세스가 종료되면 PCB는 제거됨
- PCB에 저장되어 있는 정보
  - 프로세스의 현재 상태
  - 프로세스 고유 식별자 (PID , Process IDentifier)
  - 스케줄링 및 프로세스의 우선순위
  - 프로그램의 위치
  - CPU 레지스터 정보
  - 각종 자원의 포인터
  - 주기억장치 관리 정보
  - 입/출력 상태 정보
  - 계정 정보

프로세스 상태 전이



- 준비(Ready) 상태
  - 프로세스가 프로세서를 할당받기 위해 기다리고 있는 상태
  - 프로세스는 준비상태 큐에서 실행을 준비
- 실행(Running) 상태
  - 준비상태 큐에 있는 프로세스가 프로세서를 할당받아 실행되는 상태
  - 프로세스 수행이 완료되기 전에 프로세서에게 주어진 프로세서 할당 시간이 종료되면 프로세스는 준비 상태로 전이

- 실행 중인 프로세서에 입/출력처리가 필요하면 실행 중인 프로세스는 대기상태로 전이
- 준비 상태에서 실행 상태로의 전이는 CPU 스케줄러에 의해 수행

• 대기(Block) 상태
- 프로세스에 입/출력 처리가 필요하면 현재 수행 중인 프로세스가 중단되고 대기 상태로 전이
- 입/출력 처리가 완료되면 대기 상태에서 준비 상태로 전이

스레드 (Thread)
• 프로세스 내에서의 작업 단위로 시스템의 여러 자원을 할당받아 실행하는 프로그램의 단위
• 스레드의 분류
- 사용자 수준의 스레드
  · 사용자가 만든 라이브러리를 사용하여 스레드를 운용
  · 속도는 빠르지만 구현이 어려움
- 커널 수준의 스레드
  · 운영체제의 커널에 의해 스레드를 운용
  · 구현이 쉽지만 속도가 느림

문맥 교환 (Context Switching)
• CPU가 할당되는 프로세스를 변경하기 위하여 현재 CPU를 사용하여 실행되고 있는 프로세서의 상태 정보를 저장하고 제어권을 인터럽트 서비스 루틴에게 넘기는 작업

2.2 병행 프로세스와 상호 배제

병행 프로세스 (Concurrent Process)
• 두 개 이상의 프로세스들이 동시에 존재하며 실행 상태에 있는 것을 의미
• 독립적 병행 프로세스: 여러 프로세스들이 독립적으로 실행되는 것
• 협동적 병행 프로세스: 서로 협력하며 동시에 실행되는 것

임계 구역 (Critical Section)
• 다중 프로그래밍 운영체제에서 여러 개의 프로세스가 공유하는 데이터 및 자원에 대하여 어느 한 시점에서는 하나의 프로세스만 자원 또는 데이터를 사용하도록 지정된 공유 자원(영역)을 의미

상호 배제 기법 (Mutual Exclusion)
• 공유자원을 어느 시점에서 단지 한 개의 프로세스만이 사용할 수 있도록 하며, 다른 프로세스가 공유자원에 대하여 접근하지 못하게 제어하는 기법

동기화 기법 (Synchronization)
• 두 개 이상의 프로세스를 한 시점에서는 동시에 처리할 수 없으므로 각 프로세스에 대한 처리 순서를 결정하는 것으로 상호 배제의 한 형태

• 동기화 구현방법: 세마포어, 모니터 기법
• 세마포어 (Semaphore)
- E. J. Dijkstra 가 제안하였으며, P와 V라는 2개의 연산에 의해서 동기화를 유지하며 상호 배제의 원리를 보장
- S는 P와 V 연산만으로만 접근 가능한 세마포어 변수로, 공유 자원의 개수를 나타내며 0과 1 혹은 0과 양의 값을 가질 수 있음
• 모니터 (Monitor)
- 모니터 내의 공유 자원을 사용하려면 프로세스는 반드시 모니터의 진입부를 호출해야 함
- 모니터 외부의 프로세스는 모니터 내부의 데이터를 직접 액세스 할 수 없음
- 모니터의 경계에서 상호 배제가 시행됨
- 모니터에서는 한순간에 하나의 프로세스만 진입하여 자원을 사용할 수 있음
- 모니터에서는 Wait와 Signal 연산이 사용
- 특정의 공유자원을 할당하는데 필요한 데이터 및 프로시저를 포함하는 병행성 구조임

2.3 교착 상태 (Deadlock)

교착 상태(Deadlock)의 개념
• 교착상태는 상호 배제에 의해 나타나는 문제점으로, 둘 이상의 프로세스들이 자원을 점유한 상태에서 서로 다른 프로세스가 점유하고 있는 자원을 요구하며 무한정 기다리는 현상을 의미

교착 상태 발생 조건
• 상호 배제(Mutual Exclusion): 한 번에 한 개의 프로세스만이 공유 자원을 사용할 수 있어야 함
• 점유 및 대기(Hold and Wait): 프로세스가 이미 자원을 갖고 있으면서 다른 자원의 할당을 요구
• 비선점(Non-Preemption): 프로세스에 할당된 자원은 사용이 끝날 때까지 강제로 빼앗을 수 없음
• 환형 대기(Circular Wait): 프로세스는 자신이 가지고 있는 자원을 점유하면서 앞이나 뒤에 있는 프로세스의 자원을 요구

교착 상태 해결 방법
• 예방 기법 (Prevention)
- 교착 상태가 발생하지 않도록 사전에 시스템을 제어하는 방법으로, 교착 상태 발생의 4가지 조건 중에서 어느 하나를 제거함으로써 수행 (자원낭비가 가장 심함)
• 회피 기법 (Avoidance)
- 교착 상태가 발생할 가능성을 배제하지 않고 교착 상태가 발생하면 적절히 피해나가는 방법

- 은행원 알고리즘 (Banker's Algorithm)
  · 은행원 알고리즘은 E.J. Dijkstra가 제안
  · 각 프로세스에게 자원을 할당하여 교착 상태가 발생하지 않으며 모든 프로세스가 완료될 수 있는 상태를 안전 상태, 교착 상태가 발생할 수 있는 상태를 불안전 상태라 함
  · 은행원 알고리즘을 적용하기 위해서는 자원의 양과 사용자(프로세스)수가 일정해야 함
  · 은행원 알고리즘은 프로세스의 모든 요구를 유한한 시간 안에 할당하는 것을 보장

• 발견 기법 (Detection)
- 시스템에 교착 상태가 발생했는지 점검하여 교착 상태에 있는 프로세스와 자원을 발견

• 회복 기법 (Recovery)
- 교착 상태를 일으킨 프로세스를 종료하고 교착 상태의 프로세스에 할당된 자원을 회수하여 프로세스나 자원을 회복

2.4 스케줄링

스케줄링(Scheduling)
• 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업

스케줄링의 목적
• 처리율 증가
• CPU 이용률 증가
• 오버헤드 최소화
• 응답시간 최소화
• 반환시간 최소화
• 대기시간 최소화

비선점 스케줄링
• 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없는 스케줄링 기법
• 모든 프로세스에 대한 요구를 공정하게 처리할 수 있음
• 프로세스 응답 시간 예측이 용이
• 중요한 작업이 중요하지 않은 작업을 기다리는 경우가 발생할 수 있음
• 비선점 스케줄링의 종류에는 FIFO, SJF, 우선순위, HRN, 기한부 등의 알고리즘이 있음
• 대화형 시스템에 부적합

비선점 기법 종류
• FIFO (First In First Out) = FCFS
- 가장 간단한 방식이고 비선점 방식의 스케줄링
- 중요하지 않은 작업이 작업을 기다리게 할 수 있음
- 대화식 시스템에 부적합

• SJF (Shortest Job First)
- 준비상태 큐에서 기다리고 있는 프로세스들 중에서 실행시간이 가장 짧은 프로세스에게 먼저 CPU를 할당하는 기법
- 가장 적은 평균 대기 시간을 제공하는 최적 알고리즘
- 실행 시간이 긴 프로세스는 실행 시간이 짧은 프로세스에게 할당 순위가 밀려 무한 연기 상태가 발생 가능

• HRN (Highest Response-ratio Next)
- 실행시간이 긴 프로세스에 불리한 SJF 기법을 보완하기 위한 것으로 대기 시간과 서비스 시간을 이용하는 기법
- 우선순위를 계산하여 그 숫자가 가장 높은 것부터 낮은 순으로 우선순위가 부여
- 우선순위 계산식

$$\frac{\text{대기 시간} + \text{서비스 시간}}{\text{서비스 시간}}$$

에이징(Aging) 기법
• 프로세스가 자원을 기다리고 있는 시간에 비례하여 우선순위를 부여함으로써 무한연기 문제를 방지

선점 기법
• 하나의 프로세스가 CPU를 할당받아 실행하고 있을 때 우선순위가 높은 다른 프로세스가 CPU를 강제로 빼앗아 사용할 수 있는 스케줄링 기법
• 우선순위가 높은 프로세스를 빠르게 처리할 수 있음
• 주로 빠른 응답 시간을 요구하는 대화식 시분할 시스템에서 사용
• 선점 스케줄링의 종류에는 SRT, Round Robin, 선점 우선순위, 다단계 큐, 다단계 피드백 큐 등의 알고리즘이 있음

선점 기법 종류
• SRT (Shortest Remaining Time)
- 비선점 스케줄링인 SJF 기법을 선점 형태로 변경한 기법
- 현재 실행 중인 프로세스의 남은 시간과 준비상태 큐에 새로 도착한 프로세스의 실행 시간을 비교하여 가장 짧은 실행 시간을 요구하는 프로세스에게 CPU를 할당하는 기법으로 시분할 시스템에 유용

• RR (Round Robin)
- 시분할 시스템을 위해 고안된 방식으로 FIFO 기법을 선점 형태로 변형한 기법
- 할당되는 시간이 클 경우 FIFO 기법과 같아짐
- 할당되는 시간이 작은 경우 문맥 교환 및 오버헤드가 자주 발생
- 프로세스들이 배당 시간 내에 작업되지 못하면 준비상태 큐의 맨 뒤로 이동

- 선점 우선순위
  - 준비상태 큐의 프로세스들 중에서 우선순위가 가장 높은 프로세스에게 먼저 CPU를 할당하는 기법
- 다단계 큐(MQ , Multi-level Queue)
  - 프로세스를 특정 그룹으로 분류할 수 있을 경우 그룹에 따라 각기 다른 준비상태 큐를 사용하는 기법
  - 프로세스가 특정 그룹의 준비상태 큐에 들어갈 경우 다른 준비상태 큐로 이동할 수 없음
  - 하위 준비 상태 큐에 있는 프로세스를 실행하는 도중이라도 상위 준비 상태 큐에 프로세스가 들어오면 상위 프로세스에게 CPU를 할당해야 함
- 다단계 피드백 큐(MFQ, Multi-level Feedback Queue)
  - 특정 그룹의 준비상태 큐에 들어간 프로세스가 다른 준비 상태 큐로 이동할 수 없는 다단계 큐 기법을 준비상태 큐 사이를 이동할 수 있도록 개선한 기법

3 기억장치 관리

3.1 주기억 장치

주기억장치 관리 전략

- 반입(Fetch) 전략
  - 보조기억장치에 보관 중인 프로그램이나 데이터를 언제 주기억장치로 적재할 것인지를 결정하는 전략
  - 요구반입: 실행프로그램이 요구할 때 비로소 적재하는 방법
  - 예상반입: 앞으로 요구될 가능성이 큰 데이터 또는 프로그램을 예상하여 주기억장치로 미리 옮기는 방법
- 배치(Placement) 전략
  - 새로 반입되는 프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정하는 전략
    - · 최초 적합(First Fit): 적재 가능한 공간 중 첫 번째 공간에 배치
    - · 최적 적합(Best Fit): 적재 가능한 공간 중 단편화(남는 기억 공간)가 가장 적은 공간에 배치
    - · 최악 적합(Worst Fit): 적재 가능한 공간 중 가장 큰 공간에 배치
  - 방법에 따른 배치전략 예
    - · First Fit, Best Fit, Worst Fit 방법을 사용하여 10K의 프로그램이 할당되는 영역의 번호

영역	영역 크기 및 상태
A	5K 공백
B	14K 공백
C	10K 사용 중
D	12K 공백
E	16K 공백

  - * First Fit: 10K의 프로그램이 배치될 수 있는 첫 번째 영역인 B에 할당
  - * Best Fit: 단편화가 가장 작은 영역인 D에 할당
  - * Worst Fit: 단편화가 가장 큰 영역인 E에 할당
- 교체(Replacement) 전략
  - 주기억장치의 모든 영역이 이미 사용 중인 상태에서 새로운 프로그램이나 데이터를 주기억장치에 배치하려고 할 때 사용되고 있는 영역 중에서 어느 영역을 교체하여 사용할 것인지를 결정하는 전략
  - 교체 전략에는 FIFO, OPT, LRU, LFU, NUR 등이 있음

주기억장치 할당 기법

- 단일 분할 할당 기법
  - 주기억장치를 운영체제 영역과 사용자 영역으로 나누어 한 순간에는 오직 한 명의 사용자만이 주기억장치의 사용자 영역을 사용하는 기법
  - 오버레이(Overlay) 기법: 작업의 모든 부분들이 동시에 주기억 장소에 상주해 있을 필요가 없을 때 작업을 분할하여 필요한 부분만 교체
  - 스와핑(Swapping) 기법: 하나의 프로그램 전체를 주기억장치에 할당하여 사용하다 필요에 따라 다른 프로그램과 교체
- 다중 분할 할당 기법
  - 고정 분할 할당 기법
  - 가변 분할 할당 기법

주기억장치 관리 기법의 문제점과 해결 방법

- 단편화 (Fragmentation)
  - 분할된 주기억장치에 프로그램을 할당하고 반납하는 과정을 반복하면서 사용되지 않고 남는 기억장치의 빈 공간 조각을 의미하며 내부단편화와 외부단편화가 있음
  - 내부 단편화(Internal Fragmentation): 분할된 영역이 할당될 프로그램의 크기보다 크기 때문에 프로그램이 할당된 후 사용되지 않고 남아 있는 빈 공간
  - 외부 단편화(External Fragmentation): 분할된 영역이 할당될 프로그램의 크기보다 작기 때문에 프로그램이 할당될 수 없어 사용되지 않고 빈 공간으로 남아 있는 분할된 전체 영역
- 단편화 해결 방법
  - 주기억장치를 재사용할 수 있도록 단편화된 공간을 모아서 하나의 사용할 수 있는 공간으로 만드는 기법으로 통합 기법과 압축 기법이 있음
  - 통합(Coalescing) 기법: 주기억장치 내에 인접해 있는 단편화된 공간을 하나의 공간으로 통합하는 작업
  - 압축(Compaction) 기법: 주기억장치내에 서로 떨어져 있는 여러 개의 낭비 공간을 모아서 하나의 큰 기억 공간을 만드는 작업으로 쓰레기 수집(Garbage Collection)이라고도 함

3.2 가상기억장치

가상기억장치 개요

- 가상기억장치는 보조기억장치의 일부를 주기억장치처럼 사용하는 것으로, 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법
- 프로그램을 여러 개의 작은 블록 단위로 나누어서 보관해놓고, 프로그램 실행 시 요구되는 블록만 주기억장치에 불연속적으로 할당하여 처리
- 주기억장치의 크기보다 큰 프로그램을 실행하기 위해 사용
- 주기억장치의 이용률과 다중 프로그램의 효율을 높일 수 있음
- 가상기억장치에 저장된 프로그램을 실행하기 위해서 가상기억장치의 주소를 주기억장치의 주소로 바꾸는 주소 변환(Mapping) 작업이 필요함
- 페이징 기법과 세그먼테이션 기법으로 나눔

페이징(Paging) 기법

- 가상기억장치에 보관되어 있는 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 나눠진 프로그램(페이지)을 동일하게 나눠진 주기억장치의 영역(페이지프레임)에 적재시켜 실행하는 기법
- 프로그램을 일정한 크기로 나눈 단위를 페이지(Page)라고 하고, 페이지 크기로 일정하게 나누어진 주기억장치의 단위를 페이지 프레임(Page Frame)이라 함
- 외부 단편화는 발생하지 않으나 내부 단편화는 발생할 수 있음
- 주소 변환을 위해서 페이지 맵 테이블이 필요

세그먼테이션(Segmentation) 기법

- 가상기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행시키는 기법
- 프로그램을 배열이나 함수 등과 같은 논리적인 크기로 나눈 단위를 세그먼트라고 하며, 각 세그먼트는 고유한 이름과 크기를 가짐
- 세그먼테이션 기법을 이용하는 궁극적인 이유는 기억 공간을 절약하기 위함
- 세그먼트가 주기억장치에 적재될 때 다른 세그먼트에게 할당된 영역을 침범할 수 없으며, 이를 위해 기억장치 보호키(Storage Protection Key)가 필요
- 외부 단편화가 발생할 수 있음

페이지 교체 알고리즘

- 최적 교체(OPT, Optimal replacement)
  - 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법
  - Belady가 제안한 것으로 페이지 부재 횟수가 가장 적게 발생하는 가장 효율적인 알고리즘
  - 각 페이지의 호출 순서와 참조 상황을 미리 예측해야 하므로 실현 불가

- FIFO (First In First Out)
  - 주기억장치에 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법
  - 이해하기 쉽고 프로그래밍 및 설계가 간단
  - 프로세스에 할당된 페이지 프레임 수가 증가하면 페이지 부재의 수가 감소하는 것이 당연하지만 페이지 프레임 수가 증가할 때, 현실적으로 페이지 부재가 더 증가하는 모순(anomaly)현상이 발생
- LRU (Least Recently Used)
  - 최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법
  - 각 페이지마다 계수기나 스택을 두어 현 시점에서 가장 오랫동안 사용하지 않은 페이지를 교체
- LFU (Least Frequently Used)
  - 사용 빈도가 가장 낮은 페이지를 교체하는 기법
- NUR (Not Used Recently)
  - LRU와 비슷한 알고리즘으로 최근에 사용하지 않은 페이지를 교체하는 기법
  - 최근의 사용여부를 확인하기 위해 각 페이지 마다 2개의 비트를 사용, 참조 비트와 변형 비트를 사용
  - 참조 비트와 변형 비트의 값에 따라 순서가 결정되고 페이지 교체
- SCR (Second Chance Replacement)
  - 가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지의 교체를 방지하기 위한 것으로, FIFO 기법의 단점을 보안한 기법

페이지 크기

- 페이지 크기가 작을 경우
  - 페이지 단편화가 감소되고, 한 개의 페이지를 주기억장치로 이동하는 시간이 줄어듦
  - 프로그램 수행에 필요한 내용만 주기억장치에 적재할 수 있고, 지역성(Locality)에 더 일치할 수 있기 때문에 기억장치 효율이 높아짐
  - 페이지 정보를 갖는 페이지 맵 테이블의 크기가 커지고, 맵핑 속도가 늦어짐
  - 디스크 접근 횟수가 많아져서 전체적인 입/출력 시간은 늘어남
  - 더 많은 페이지가 존재
- 페이지 크기가 클 경우
  - 페이지 정보를 갖는 페이지 맵 테이블의 크기가 작아지고, 맵핑 속도가 빨라짐
  - 디스크 접근 횟수가 줄어들어 전체적인 입/출력의 효율성이 증가됨
  - 페이지 단편화가 증가되고, 한 개의 페이지를 주기억장치로 이동하는 시간이 늘어남
  - 프로그램 수행에 불필요한 내용까지도 주기억장치에 적재될 수 있음

페이지 부재 (Page Fault)
- 프로세스 실행 시 참조할 페이지가 주기억장치에 없는 현상을 의미
- 페이지 부재율(Page Fault Rate): 페이지 부재가 일어나는 횟수
- 페이지 부재율에 따라 주기억장치에 있는 페이지 프레임의 수를 늘리거나 줄여 페이지 부재율을 적정 수준으로 유지
- 페이지 부재를 처리하는 순서
  - 운영체제에서 트랩을 요청
  - 사용자 레지스터와 프로그램 상태를 저장
  - 사용 가능한 프레임을 프레임 리스트에서 찾음
  - backing store에 있는 페이지를 물리적 메모리로 가져옴

3.3 디스크 스케줄링

디스크 스케줄링의 개요
- 사용할 데이터가 디스크상의 여러 곳에 저장되어 있을 경우 데이터를 액세스하기 위해 디스크 헤드가 움직이는 경로를 결정하는 기법
- 목적
  - 처리량의 최대화
  - 응답 시간의 최소화
  - 응답 시간 편차의 최소화

FIFO (=FCFS)
- 디스크 대기 큐에 가장 먼저 들어온 트랙에 대한 요청을 먼저 서비스하는 디스크 스케줄링 기법
- 디스크 대기 큐에 있는 트랙 순서대로 디스크 헤드를 이동
- 디스크 대기 큐에 들어온 순서대로 서비스를 하기 때문에 공평성이 보장됨

SSTF (Shortest Seek Time First)
- 탐색거리가 가장 짧은 트랙에 대한 요청이 먼저 서비스 받는 디스크 스케줄링 기법
- 현재 헤드 위치에서 가장 가까운 거리에 있는 트랙으로 헤드를 이동시킴
- 처리량이 많은 일괄 처리 시스템에 유용
- 현재 서비스한 트랙에서 가장 가까운 트랙에 대한 서비스 요청이 계속 발생하는 경우, 먼 거리의 트랙에 대한 서비스는 무한정 기다려야 하는 기아상태가 발생할 수 있음
- 응답 시간의 편차가 크기 때문에 대화형 시스템에는 부적합

SCAN
- 현재 진행 중인 방향으로 가장 짧은 탐색 거리에 있는 요청을 먼저 서비스하는 디스크 스케줄링 기법
- 현재 헤드의 위치에서 진행 방향이 결정되면 탐색 거리가 짧은 순서에 따라 그 방향의 모든 요청을 서비스하고, 끝까지 이동한 후 역방향으로 서비스 함
- 헤드의 진행 방향에 있는 대기 요청뿐만 아니라 새로운 요청도 서비스하며, 현재의 진행 방향에 더 이상의 요청이 없을 때에만 이동 방향을 바꿈
- SSTF에서 발생하는 응답 시간의 편차를 줄임

C-SCAN
- 항상 바깥쪽에서 안쪽으로 움직이면서 가장 짧은 탐색 거리를 갖는 요청을 서비스 하는 디스크 스케줄링 기법
- 헤드는 트랙의 바깥쪽에서 안쪽으로 한 방향으로만 움직이며 서비스하여 끝까지 이동한 후, 안쪽에 더 이상의 요청이 없으면 헤드는 가장 바깥쪽의 끝으로 이동한 후 다시 안쪽으로 이동하면서 요청을 서비스함
- 요청을 서비스하는 도중 새로운 요청이 도착하면 다음 헤드가 진행할 때 서비스함
- 트랙의 안쪽과 바깥쪽의 요청에 대한 서비스가 공평함

N-step SCAN
- SCAN 기법을 기초로 하며 어떤 방향의 진행이 시작될 당시에 대기 중이던 요청에 대해서만 서비스하고 진행 도중 도착한 요청들은 반대 방향 진행 때 서비스하는 디스크 스케줄링 기법
- SSTF나 SCAN 기법보다 응답 시간의 편차가 적음
- 특정 방향에 많은 수의 요청이 도착할 경우 반대 방향에서의 무한 지연을 방지함

Look
- SCAN 기법을 사용하되 진행 방향의 마지막 요청을 서비스한 후 그 방향의 끝으로 이동하는 것이 아니라 바로 역방향으로 진행하는 디스크 스케줄링 기법

4 정보관리

4.1 파일과 파일시스템

파일
- 파일은 사용자가 작성한 서로 관련 있는 레코드의 집합체를 의미
- 프로그램 구성의 기본 단위가 되며, 보조기억장치에 저장
- 각 파일마다 위치, 크기, 작성 시기 등의 여러 속성을 가짐

파일시스템의 기능 및 특징
- 사용자와 보조기억장치 사이에 인터페이스를 제공
- 사용자가 파일을 생성, 수정, 제거할 수 있게 함
- 불의의 사태에 대비하여 파일의 백업(Backup)과 복구(Recovery)등의 기능을 제공
- 파일을 안전하게 사용할 수 있도록 하고 파일이 보호되어야 함
- 파일의 정보가 손실되지 않도록 데이터 무결성을 유지해야 함
- 여러 가지 접근 제어 방법을 제공

파일 디스크립터(File Descriptor)의 개요
- 파일을 관리하기 위한 시스템이 필요로 하는 파일에 대한 정보를 갖는 제어 블록을 의미
- 파일 디스크립터는 파일마다 독립적으로 존재하며 시스템에 따라 다른 구조를 가질 수 있음
- 보통 파일 디스크립터는 보조기억장치 내에 저장되어 있다가 해당 파일이 오픈될 때 주기억장치로 이동
- 파일 디스크립터는 파일 시스템이 관리하므로 사용자가 직접 참조할 수 없음
- 파일 제어 블록(FCB , File Control Block)이라고도 함

파일 디스크립터의 정보
- 파일 이름
- 보조기억장치에서의 파일 위치
- 파일 구조
- 보조기억장치의 유형
- 액세스 제어 정보
- 파일 유형
- 생성 날짜와 시간, 제거 날짜와 시간
- 최종 수정 날짜 및 시간
- 액세스한 횟수

4.2 파일의 구조

순차 파일 (Sequential File)
- 순차 파일은 레코드를 논리적인 처리 순서에 따라 연속된 물리적 저장 공간에 기록
- 파일의 레코드들이 순차적으로 기록되어 판독할 때 순차적으로 접근
- 순차 접근 방식(SAM , Sequential Access Method)이라고 함

직접 파일 (Direct File)
- 직접 파일은 파일을 구성하는 레코드를 임의의 물리적 저장 공간에 기록
- 레코드에 특정 기준으로 키를 할당
- 직접 접근 방식(DAM , Direct Access Method)이라고 함

색인 순차 파일 (Indexed Sequential File)
- 색인을 이용한 순차적인 접근 방법을 제공
- 색인 순차 접근 방식(ISAM , Index Sequential Access Method)이라고 함
- 물리적 특성에 따른 색인 구성
  - 트랙 색인 (track index)
  - 실린더 색인 (cylinder index)
  - 마스터 색인 (master index)

4.3 디렉토리 구조

1단계(단일) 디렉토리 구조
- 가장 간단하고, 모든 파일이 하나의 디렉토리 내에 위치하여 관리

2단계 디렉토리 구조
- 중앙에 마스터 파일 디렉토리가 있고, 그 아래에 사용자별로 서로 다른 파일 디렉토리가 있음

계층적(트리) 디렉토리 구조
- 하나의 루트 디렉토리와 여러 개의 서브 디렉토리로 구성

비순환(비주기) 그래프 디렉토리 구조
- 하위 파일이나 하위 디렉토리를 공동으로 사용할 수 있는 구조로 사이클이 허용되지 않는 구조

일반적인 그래프 디렉토리 구조
- 트리 구조에 링크를 이용하여 순환을 허용하는 그래프 구조

4.4 디스크 공간 할당 방법

연속 할당 (Contiguous Allocation)
- 파일을 디스크의 연속된 기억 공간에 할당하는 방법으로 생성되는 파일 크기만큼의 공간이 요구됨
- 논리적으로 연속된 레코드들이 물리적으로 인접한 공간에 저장되기 때문에 접근 시간이 빠름
- 디렉토리가 단순하고, 관리 및 구현이 용이
- 파일 크기에 알맞은 연속 공간이 없을 경우 파일이 생성되지 않음
- 파일의 생성과 삭제가 반복되면서 단편화가 발생
- 단편화를 줄이기 위해서 주기적인 압축이 필요
- 사용자는 만들고자 하는 파일의 크기에 해당하는 디스크 공간을 미리 지정해 주어야 함

불연속 할당
- 디스크 공간을 일정 단위로 나누어 할당하는 기법으로 섹터 단위 할당과 블록 단위 할당이 있음
- 블록 단위 할당
  - 하나의 파일이 연속된 여러 개의 섹터를 묶은 블록 단위로 할당되는 방법
  - 블록 체인 기법
  - 색인(인덱스) 블록 체인 기법
  - 블록 지향 파일 사상 기법
  - 파일 할당 테이블(FAT , File Allocation Table): 사용자가 해당 블록의 포인트를 실수로 지워지게 하는 것을 예방하고 블록 접근을 빠르게 하기 위하여 포인터를 모아 놓은 곳

4.5 자원 보호

자원 보호 기법

- 접근 제어 행렬
  - 자원 보호의 일반적인 모델로 자원에 대한 접근 권한을 행렬로 표시한 기법
  - 접근 제어 행렬 예

파일사용자	A	B	C
인사 파일	E	REW	E
급여 파일	RW	-	R

(E: 실행가능, R: 판독가능, W: 기록가능)

- A는 인사파일 실행이 가능, 급여파일은 판독과 기록이 가능
- B는 인사파일 판독, 실행, 기록이 가능, 급여파일은 접근 불가
- C는 인사파일 실행이 가능, 급여파일은 판독만 가능

- 전역 테이블
  - 가장 단순한 구현 방법으로 영역, 객체, 접근 권한의 집합을 목록 형태로 구성한 기법
- 접근 제어 리스트
  - 자원을 중심으로 접근 리스트를 구성
  - 접근 권한이 없는 영역은 제외됨
  - 사용자에 의해 간접적으로 액세스되는 기법

파일 보호 기법

- 자원 보호 기법과 마찬가지로 파일에 대한 일반적인 접근과 손상을 방지하기 위한 기법
- 파일의 명명(Naming)
- 비밀번호(Password)
- 접근 제어(Access Control)

비밀번호(Password) 설정

- 추출 가능한 전화번호, 생년월일용으로는 구성하지 않는 것이 좋음
- 암호는 자주 변경하는 것이 좋음
- 불법 액세스를 방지하는데 사용

4.6 보안

보안 요건

- 기밀성: 시스템 내의 정보와 자원은 인가된 사용자에게만 접근이 허용
- 무결성: 시스템 내의 정보는 오직 인가된 사용자만 수정 가능
- 가용성: 인가받은 사용자는 언제든지 사용 가능
- 인증: 컴퓨터 시스템에서 전송 정보가 오직 인가된 당사자에 의해서만 수정될 수 있도록 통제하는 것
- 부인방지: 데이터를 송/수신한 자가 송/수신한 사실을 부인할 수 없도록 송/수진 증거를 제공

보안 유지 기법

- 외부 보안
  - 시설 보안
  - 운용 보안
- 내부 보안
  - 하드웨어나 운영체제에 내장된 보안 기능을 이용하여 보안 문제를 해결하는 기법
  - 프로그램의 신뢰성 있는 운영과 데이터의 무결성을 보장
- 사용자 인터페이스 보안
  - 운영체제가 사용자의 신원을 확인한 후 권한이 있는 사용자에게만 사용할 수 있게 하는 보안 기법

정보 보안 기법

- 비밀키 시스템 (Private Key System)
  - 동일한 키로 데이터를 암호화하고 해독하는 대칭 암호화 기법
  - 키를 아는 사람은 누구나 해독 가능 하므로 키의 비밀성을 유지하는 것이 중요
  - 암호화/복호화 속도가 빠르고 알고리즘이 단순함
  - 키의 분배가 어려움
  - 대표적인 방식에는 DES(Data Encryption Standard)가 있음
    - DES 기법: 평문을 64비트로 블록화 하고, 실제 키의 길이는 56비트를 이용
- 공개키 시스템 (Public Key System)
  - 서로 다른 키로 데이터를 암호화하고 해독하는 비대칭 암호화 기법
  - 암호키는 공개하고 해독키는 비밀로 함으로써 해독키를 가진 사람만이 해독 가능
  - 키의 분배가 용이
  - 암호화/복호화 속도가 느리고 알고리즘이 복잡함
  - 대표적인 방식에는 RSA(Rivest Shamir Adleman)가 있음
- 디지털 서명 기법
  - 손으로 쓴 서명과 같이 고유의 전자 서명으로 송신자가 전자 문서 송신 사실을 나중에 부인할 수 없도록 하고, 작성 내용이 송/수신 과정에서 변조된 사실이 없음을 증명하는 기법
- 인증 교환 기법
  - 수신자가 메시지 전송 도중에 변경되지 않았음을 확인할 수 있으며, 메시지가 정당한 상대방으로부터 전달된 것임을 확인할 수 있는 기법
- 접근 제어 기법
  - 데이터에 접근이 허가된 자에게만 데이터 사용을 허용하는 정책을 강화하기 위해 사용하는 기법

보안 메커니즘의 설계 원칙에서 개방된 설계의 의미

- 알고리즘은 알려졌으나, 그 키는 비밀인 암호 시스템의 사용을 의미

5 분산 운영체제

5.1 다중 처리기 (Multi-processor)

프로세서 연결 방식

- 공유 버스
  - 장치 연결이 단순
  - 장치 추가가 용이
  - 한 시점에 하나의 전송만이 가능
  - 버스에 이상이 발생하면 전체 시스템이 중단
  - 시스템의 전체 전송량이 버스의 전송률에 의해 제한됨
- 크로스바 교환 행렬
  - 각 기억장치마다 다른 경로를 사용할 수 있음
  - 두 개의 서로 다른 기억 장치를 동시에 참조 가능
  - 하드웨어가 복잡해짐

다중 처리기의 운영체제 구조

- 주/종(Master/Slave) 처리기
  - 하나의 프로세서를 Master로 지정하고, 나머지들은 Slave로 지정하는 구조
  - 주프로세서가 고장나면 전체 시스템이 중단
  - 주프로세서만 입/출력을 수행하므로 비대칭 구조를 가짐
  - 주프로세서의 역할: 입/출력과 연산을 담당, 운영체제를 수행
  - 종프로세서의 역할: 연산만 담당, 입/출력 발생 시 주프로세서에게 서비스를 요청
- 분리 수행 처리기
  - 주/종 처리기의 비대칭성을 보안하여 각 프로세서가 독자적인 운영체제를 가진 구조
  - 프로세서별 인터럽트는 독립적으로 수행
- 대칭적 처리기
  - 여러 프로세서들이 완전한 기능을 갖춘 하나의 운영체제를 공유하여 수행하는 구조

프로세서의 결합도

- 약결합 시스템 (Loosely Coupled System)
  - 각 프로세서마다 독립된 메모리를 가진 시스템으로 분산 처리 시스템이라 함
  - 둘 이상의 독립된 컴퓨터 시스템을 통신망을 통하여 연결한 시스템
  - 각 시스템마다 독자적인 운영체제가 존재
  - 프로세서 간의 통신은 메시지 전달이나 원격 프로시저 호출을 통하여 이루어짐
  - 각 시스템마다 독자적인 운영이 가능하여 프로세서간의 결합력이 약함

- 강결합 시스템 (Tightly Coupled System)
  - 동일 운영체제 하에서 여러 개의 프로세서가 하나의 메모리를 공유하여 사용하는 시스템으로 다중 처리 시스템이라 함
  - 하나의 운영체제가 모든 프로세서와 시스템을 제어
  - 프로세서 간의 통신은 공유 메모리를 통하여 이루어짐
  - 하나의 메모리를 사용하므로 프로세서 간의 결합력이 강함
  - 공유 메모리를 차지하려는 프로세서 간의 경쟁을 최소화해야 함

Flynn의 분류

- SISD (Single Instruction Single Data)
  - 단일 명령 흐름에 대한 단일 데이터 흐름
- SIMD (Single Instruction Multiple Data)
  - 단일 명령 흐름에 대한 다중 데이터 흐름, 벡터 처리기 혹은 배열 컴퓨터
- MISD (Multiple Instruction Single Data)
  - 다중 명령 흐름에 대한 단일 데이터 흐름, 이론적일 뿐 실질적인 처리 방식으로 사용되지 않는 구조
- MIMD (Multiple Instruction Multiple Data)
  - 다중 명령 흐름에 대한 다중 데이터 흐름, 진정한 의미의 병렬 처리 구조

5.2 분산 시스템

분산 시스템 설계 목적

- 자원 공유
  - 각 시스템이 통신망을 통해 연결되어 있으므로 유용한 자원을 공유하여 사용
- 연산 속도 향상
  - 하나의 일을 여러 시스템에 분산시켜 처리함으로써 연산 속도가 향상
- 신뢰도 향상
  - 여러 시스템 중 하나의 시스템에 오류가 발생하더라도 다른 시스템은 계속 일을 처리할 수 있으므로 신뢰도가 향상

분산 시스템 특징

- 약결합(loosely-coupled) 시스템
- 업무량 증가에 따른 점진적인 확장이 용이
- 제한된 자원을 여러 지역에서 공유 가능
- 작업을 병렬적으로 수행함으로써 사용자에게 빠른 반응 시간과 작업 처리량이 향상
- 작업의 부하를 균등하게 유지
- 일부가 고장 나더라도 나머지 일부는 계속 작동 가능
- 계층구조는 하드웨어계층 - 기억 장치계층 - 프로세스계층 - 파일 시스템계층 - 사용자 프로그램 계층으로 분류됨

분산 시스템 장 / 단점
- 장점
  - 여러 사용자들 간의 통신이 용이
  - 제한된 장치를 여러 지역의 사용자가 공유
  - 여러 사용자들이 데이터를 공유
  - 중앙 컴퓨터의 과부하가 적음
  - 사용자는 각 컴퓨터의 위치를 몰라도 자원 사용 가능
  - 하나의 일을 나누어 처리함으로써 연산 속도, 신뢰도, 사용 가능도가 향상, 결함 허용이 가능
  - 시스템의 점진적 확장이 용이
- 단점
  - 중앙 집중형 시스템에 비해 소프트웨어 개발이 어려움
  - 보안의 어려움

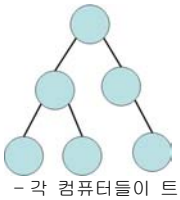
분산 시스템 투명성
- 위치 투명성: 자원의 물리적인 위치를 모르더라도 자원에 접근 가능
- 이주 투명성: 사용자나 응용프로그램의 동작에 영향을 받지 않고 시스템 내에 있는 자원을 이동 가능
- 복제 투명성: 사용자에게 통지할 필요 없이 자원들의 부가적인 복사를 자유로이 수행 가능
- 병행 투명성: 여러 다중 사용자들이 자원들을 병행하여 처리하고 공유가 가능

분산 시스템에서 발생하는 결함
- 링크 결합: 연결 부분에 발생할 수 있는 결함
- 사이트 결함: 각 사이트(시스템)에서 발생할 수 있는 결함
- 메시지 분실: 사이트에서 다른 사이트로 메시지 전송 시 발생할 수 있는 결함

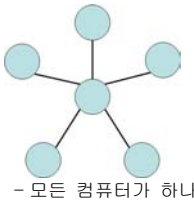
클라이언트 / 서버 모델
- 중앙 컴퓨터에서 대부분의 작업을 처리하던 예전의 호스트/터미널 모델과는 달리 클라이언트 쪽에서 더 많은 작업을 처리하여 서버의 부하를 줄이는 방식
- 통신 응용을 위한 표준 모델
- 시스템 확장이 용이하고 유연함
- 서버는 공유된 다양한 시스템 기능과 자원을 제공
- 처리할 자료의 출처 가까이에서 처리 작업이 진행
- 프로그램의 모듈성과 융통성을 증대시킴
- 개방형 시스템으로 다양한 하드웨어와 소프트웨어 선택이 가능

위상에 따른 분류
- 망형 - 완전 연결형

  - 각 컴퓨터는 시스템내의 모든 컴퓨터들과 직접 연결이 존재
  - 신뢰성이 높음
  - 설치비용이 많이 듦
- 트리(tree)형 - 계층형

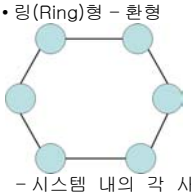


  - 각 컴퓨터들이 트리 형태로 연결된 구조
  - 상위 컴퓨터가 고장 나면 하위 컴퓨터들은 통신이 불가능
- 성(Star)형

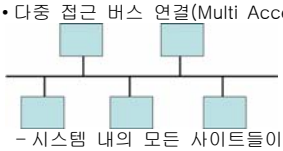


  - 모든 컴퓨터가 하나의 중앙 컴퓨터만 직접 연결되어 있음
  - 구조가 간단
  - 중앙 컴퓨터에 발생하면 전체시스템이 마비
  - 응답이 빠르고 통신비용이 적음
  - 터미널의 증가에 따라 통신 회선수도 증가

- 링(Ring)형 - 환형



  - 시스템 내의 각 사이트가 인접하는 다른 두 사이트와만 직접 연결된 구조
  - 정보는 단방향 또는 양방향으로 전달
  - 새로운 노드를 추가할 경우 통신회선을 절단해야 함
  - 목적지에 도달하는데 단방향인 경우 최대 n-1개의 노드를 거침
  - 기본비용은 사이트 수에 비례함
  - 메시지가 링을 순환할 경우 통신비용은 증가함
- 다중 접근 버스 연결(Multi Access Bus Connection)형



  - 시스템 내의 모든 사이트들이 공유 버스에 연결된 구조
  - 물리적 구조가 단순하고 사이트의 추가와 삭제가 용이
  - 사이트의 고장은 통신에 영향을 주지 않지만 공유 버스의 고장은 전체 시스템에 영향을 줌

운영체제에 따른 분류
- 네트워크 운영체제
  - 독자적인 운영체제를 가지고 있는 시스템을 네트워크로 구성한 것으로 사용자가 원격 시스템으로 로그인하거나 원격으로 자원을 전달받아 사용하는 방식
- 분산 운영체제
  - 하나의 운영체제가 모든 시스템 내의 자원을 관리하는 것으로 원격에 있는 자원을 자신의 자원처럼 쉽게 접근하여 사용할 수 있는 방식

6 운영체제의 실제

6.1 유닉스(UNIX)의 개요
UNIX의 특징
- 시분할 시스템을 위해 설계된 대화식 운영체제
- 소스가 공개된 개방형 시스템
- 대부분 C언어로 작성되어 이식성과 확장성이 높음
- 멀티 유저, 멀티 태스킹을 지원
- 트리구조의 파일 시스템
- 프로세스 간 통신을 위하여 주로 소켓을 사용

커널 (Kernel)
- UNIX 시스템의 가장 핵심적인 부분
- 항상 주기억장치에 상주
- 하드웨어와 프로그램간의 인터페이스 역할을 담당
- 프로세스 관리, 기억장치 관리, 입출력 관리, 파일 관리, 프로세스간의 통신 등을 수행
- 하드웨어를 보호하고 응용 프로그램들에게 서비스를 제공

쉘 (Shell)
- 명령어 해석기로 사용자의 명령어를 인식하여 필요한 프로그램을 호출하고 그 명령을 수행
- 사용자와 시스템 간의 인터페이스를 담당
- Bourne shell, C shell등을 사용
- 도스의 "command.com"과 같은 역할을 수행

UNIX 파일 시스템 특징
- UNIX 파일 시스템의 디렉토리 구조는 트리 구조
- 디렉토리나 주변장치를 파일로 간주하여 처리
- UNIX 시스템 구조는 사용자-쉘-커널-하드웨어

UNIX 파일 시스템 구조
- 부트 블록: 부팅 시 필요한 코드를 저장하고 있는 블록
- 슈퍼 블록: 전체 파일 시스템에 대한 정보를 저장하고 있는 블록
  - 사용가능한 i-node의 개수를 알 수 있음
  - file 시스템마다 각각의 슈퍼 블록을 가지고 있음
  - 사용 가능한 디스크 블록의 개수를 알 수 있음
- I-node 블록: 각 파일이나 디렉토리에 대한 모든 정보를 저장하고 있는 블록
  - 파일 소유자의 사용자 번호 및 그룹 번호
  - 파일 크기
  - 파일 타입
  - 파일 생성 시기
  - 파일 최종 변경 시기
  - 파일 최근 사용 시기
  - 파일의 보호 권한
  - 파일 링크 수
  - 데이터가 저장된 블록의 시작 주소

파일 시스템 보호 예
- 사용 예 (-rwxrwxr--)
  - 첫 번째 기호는 파일인지 디렉토리인지 구분 (d: 디렉토리, -: 파일)
  - 파일 소유자에게 rwx 허용, Group에게 rwx 허용, Other에게 r 허용
  - ⇨ 파일이며 소유자와 그룹은 읽고, 쓰고 실행하는 것이 가능하지만 기타 사용자에게는 읽기만 가능
- 파일 보호 기법 중 접근 제어(Access control)에 해당

UNIX 명령어

명령어	기능
Fork	• 새로운 프로세스를 생성 (프로세스 복제)
Exec	• 새로운 프로세스를 수행
Chmod	• 파일에 대한 액세스(읽기, 쓰기, 실행) 권한을 설정하여 파일의 사용 허가를 지정
Pipe	• 프로세스 간 통신을 위한 경로를 설정하여 프로세스 간 정보교환이 가능하게 함 • 전송되는 데이터는 FIFO(First In First Out) 방식으로 상대에게 전달 • 프로세스 간의 생산자 - 소비자 모델의 데이터 전달을 위한 큐
Wait	• 하위 프로세스 중의 하나가 종료될 때 까지 상위 프로세스를 임시 중지시킴
Mount	• 기존 파일 시스템에 새로운 파일 시스템을 서브디렉토리에 연결할 때 사용
Cat	• 유닉스 시스템에서 파일의 내용을 화면에 출력할 때 사용

명령어를 백그라운드로 수행시킬 때 가장 큰 장점

- 수행중인 명령문이 끝나기 전에 다른 명령문을 입력할 수 있음 (명령어 끝에 &을 입력함)

6.2 윈도우(Windows) 및 MS - DOS

윈도우(Windows)의 특징

- 그래픽 사용자 인터페이스 (GUI, Graphic User Interface)
- 선점형 멀티태스킹
- FAT32 파일 시스템 사용
- PnP(Plug and Play) 사용
- OLE(Object Linking and Embedding) 사용
- 시스템이 작동하지 않아 Control-Alt-Delete를 눌러 사용하는 것은 인터럽트와 관련됨

장치 구동기 (device driver)

- 컴퓨터 주변 장치를 만드는 업체에서 입/출력 장치를 제어하는 위해 공급하는 프로그램

MS-DOS의 특징

- 문자 중심의 사용자 인터페이스 (CUI , Character User Interface)
- Single-User, Single-Tasking
- 파일 시스템의 디렉토리 구조는 트리 구조
- "MSDOS.SYS"의 기능은 파일관리, 주변장치 관리들임
- "IO.SYS"의 기능은 입, 출력 관리임
- '가상디스크(virtual disk)'의 기능은 주기억장치의 일부를 디스크처럼 사용
  - 가상디스크의 기능을 위해 장치 제어기가 필요하며 이름은 RAMDRIVE.SYS임

시스템 부팅 시 반드시 필요한 파일

- COMMAND.COM
- MSDOS.SYS
- IO.SYS

MS-DOS 명령어

- 내부 명령어
  - 내부 명령어는 실행 과정이 간단하고 기본적인 기능을 수행하는 것으로 메모리에 항상 상주
  - Command.com에 포함되어 있으며 처리 속도가 빠름
  - 종류

명령어	기 능	UNIX 명령어
DIR	파일의 목록을 표시	LS
COPY	파일을 복사	CP
TYPE	파일의 내용을 표시	CAT
MD	디렉토리를 생성	MKDIR
CD	디렉토리의 위치를 변경	CHDIR
REN	파일을 이름을 변경	MV
DEL	파일을 삭제	RM

- 외부 명령어
  - 실행 과정이 복잡하거나 자주 사용하지 않는 것으로, 디스크에 파일로 저장되어 있음
  - 실행 파일을 디스크에서 찾아 메모리로 옮긴 후 실행하므로 처리 속도가 느림
  - 종류

명령어	기 능	UNIX 명령어
ATTRIB	파일의 속성을 변경	CHMOD
MOVE	파일을 이동	MV
FIND	파일을 찾음	FIND

데이터베이스

1 데이터베이스의 개념

1.1 정보처리시스템 과 데이터베이스의 개념

정보 시스템 (=정보처리 시스템)

- 한 조직체에서 필요한 DATA를 수집, 저장해 두었다가 필요시에 처리해서 의사결정에 유용한 정보를 생성하고 분배하는 수단

자료와 정보

- 자료: 현실 세계로부터 단순한 관찰이나 측정을 통해 수집된 사실이나 값
- 정보: 입력 받은 자료 처리 후 출력하는 값, 의사결정을 위해 쓰임

자료처리 시스템의 종류

- 일괄처리 시스템
- 온라인처리 시스템
- 분산처리 시스템

데이터베이스의 등장 배경

- 여러 사용자가 데이터를 공유해야 할 필요가 생김
- 데이터의 수시적인 구조 변경에 대해 응용 프로그램을 매번 수정하는 번거로움을 줄이고 싶음
- 물리적인 주소가 아닌 데이터 값에 의한 검색을 수행하고 싶음

데이터베이스의 정의

- 통합된 데이터
- 공용 데이터
- 운영 데이터
- 저장된 데이터

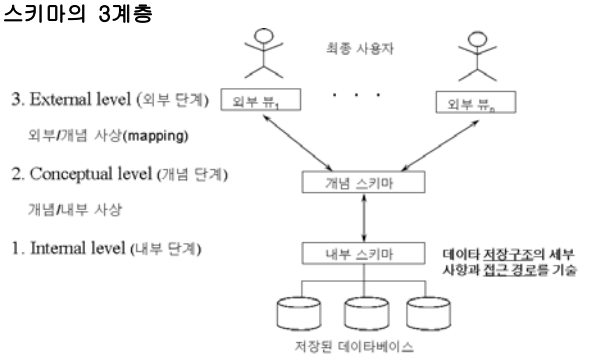
데이터베이스의 특징

- 데이터의 중복 최소화
- 데이터의 독립성 유지
- 데이터의 동시 공유
- 데이터의 보안성 유지
- 데이터의 일관성 유지
- 데이터의 표준화
- 데이터의 무결성(정확성) 유지

데이터베이스 생명주기

- 요구조건 분석→설계→구현→운영→감시 및 개선

스키마의 3계층



종 류	설 명
외부 스키마 (External Schema)	• 전체 데이터베이스의 한 논리적인 부분으로 볼 수 있으므로 서브스키마(subschema)라고도 함 • 응용의 의미보다는 어느 개인이나 특정 응용에 한정된 논리적 데이터 구조 • 데이터베이스의 개별 사용자나 응용 프로그래머가 접근하는 데이터베이스를 정의
개념 스키마 (Conceptual Schema)	• 데이터베이스 접근권한, 보안정책, 무결성 규칙을 명세화함 • 모든 응용시스템과 사용자가 필요로 하는 데이터를 통합한 조직 전체의 데이터베이스로 하나만 존재함 • 범기관적 입장에서 데이터베이스를 정의한 것
내부 스키마 (Internal Schema)	• 데이터베이스의 물리적 저장 구조를 설명한 것 • 시스템 프로그래머나 시스템 설계자가 보는 관점의 스키마

데이터베이스 사용자

종 류	설 명
데이터베이스 관리자 (DBA, DataBase Administrator)	• 데이터베이스 시스템의 모든 관리와 운영에 대한 책임을 지고 있는 사람이나 그룹
응용 프로그래머 (Application Programmer)	• 응용프로그램의 설계 및 개발 • 주로 데이터 조작어(DML)를 삽입시켜 데이터베이스 기반 응용 시스템을 작성하는 사람
일반 사용자 (End User)	• 터미널을 통해 응용 프로그램이나 질의어를 사용하여 데이터베이스에 접근하는 사람

DBA의 기능

- 데이터베이스 설계와 운영
- 행정 및 불편 해결
- 시스템 감시 및 성능 분석

2 데이터모델링 및 설계

2.1 데이터 모델 개념

데이터 모델(Data Model)

- 현실 세계의 데이터 구조를 컴퓨터 세계의 데이터 구조로 기술하는 개념적인 도구
- 현실 세계를 데이터베이스에 표현하는 중간 과정. 즉 데이터베이스 설계과정에서 데이터의 구조를 표현하기 위해 사용되는 도구

데이터 모델의 종류

종 류	설 명
개념적 데이터 모델	현실 세계에 대한 사람의 이해를 돕기 위하여 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정
논리적 데이터 모델	필드로 기술된 데이터 타입과 이 데이터 타입들 간의 관계를 이용하여 현실 세계를 표현하는 방법
물리적 데이터 모델	레코드의 형식, 순서, 접근 경로와 같은 정보를 사용하여 데이터가 컴퓨터에 저장되는 방법을 묘사

데이터 모델의 구성요소

- 논리적으로 표현된(추상적인) 데이터 구조
- 구성요소의 연산
- 구성요소의 제약조건

데이터 모델, 스키마, 인스턴스의 관계

- 모델 → 스키마 → 인스턴스

데이터 모델 구성요소

종 류	설 명
개체 (Entity)	• 데이터베이스에서 표현하려고 하는 정보 단위와 같은 현실 세계의 대상체
속성 (Attribute)	• 데이터베이스를 구성하는 가장 작은 논리적 단위
관계 (Relationship)	• 개체간 또는 속성간의 관계 • 1:1 / 1:N / N:M 관계

2.2 개체-관계(E-R) 모델

E-R 모델의 특성

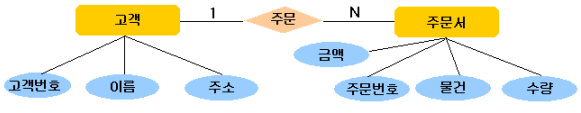
- E-R 다이어그램으로 표현하며 P.Chen이 제안
- 개체 타입과 이들 간의 관계 타입을 이용해 현실 세계를 개념적으로 표현한 방법
- E-R 다이어그램은 E-R 모델을 그래프 방식으로 표현한 것
- 데이터를 엔티티, 관계, 속성으로 묘사함
- 개념적 설계에 가장 많이 사용되는 모델
- 최초에는 entity, relationship, attribute와 같은 개념들로 구성되었으나 나중에 일반화 계층 같은 복잡한 개념들이 첨가되어 확장된 모델로 발전
- 정보모델링 과정에서 개념 세계의 정보구조로 표현하기 위한 규약
- 각 개체 집합은 여러 개의 속성으로 표현되며, 각 속성은 현실 세계의 객체들이 갖는 특성

E-R 다이어그램

- 구성요소

기 호	의 미
	개 체
	관 계
	속 성
	기본키 속성
	복합 속성
	개체 타입과 속성 연결

예)



2.3 논리적 데이터 모델

논리적 데이터 모델의 정의

- 필드로 기술된 데이터 타입과 이 데이터 타입들 간의 관계를 이용하여 현실 세계를 표현하는 방법
- 관계의 표현 방법으로 관계 데이터 모델, 계층 데이터 모델, 네트워크 데이터 모델 등으로 구분됨

관계형 데이터 모델

- 데이터베이스를 테이블(릴레이션)의 집합으로 표현
- 사용자 파일의 레코드들 사이의 논리적인 관계를 2차원의 테이블 집합으로 나타낸 구조

계층형 데이터 모델

- 개체 집합에 대한 속성 관계를 표시하기 위해 개체를 노드로 표현하고 개체 집합들 사이의 관계를 링크로 연결한 형태
- 데이터 구조도가 부자관계(parent-child relationship)를 나타내는 트리(tree) 형태의 자료 구조
- 하나의 루트 레코드 타입과 다수의 종속 레코드 타입으로 구성된 순서 트리
- 부모 레코드가 되지 못한 레코드 타입은 계층 정의 트리의 단말 노드임
- 두 레코드 타입 간에는 하나의 관계만 허용됨

네트워크(망)형 데이터 모델

- 데이터베이스의 논리적 구조 표현을 그래프 형태로 표현
- 일대다(1:n) 관계에 연관된 레코드 타입들을 각각 오너(owner), 멤버(member)라고 하고, 이들의 관계를 오너-멤버 관계라고도 일컫는 데이터 모델

객체지향 데이터 모델

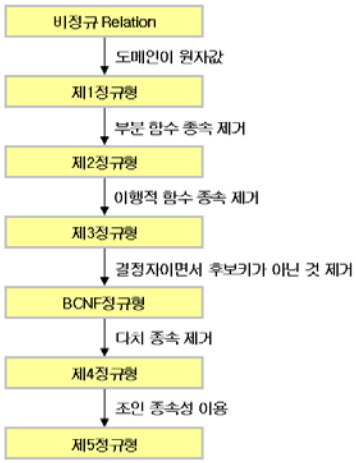
- 캡슐화(Capsulation), 상속(Inheritance), 다형성(Polymorphism)의 개념을 가지는 데이터 모델

2.4 관계 데이터베이스의 정규화

이상 (anomaly)

- 이상의 정의
  - 관계 모델에서는 애트리뷰트들 간에 존재하는 여러 종속관계를 하나의 릴레이션에 표현하기 때문에 릴레이션 조작 시 이상(anomaly)발생
  - 데이터의 중복으로 인하여 관계연산을 처리하기 곤란한 현상
- 이상의 종류
  - 삽입 이상 : 데이터를 삽입할 때 불필요한 데이터가 함께 삽입되는 현상
  - 삭제 이상 : 한 튜플을 삭제함으로써 연쇄 삭제 현상으로 인한 정보의 손실
  - 갱신 이상 : 튜플에 있는 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상

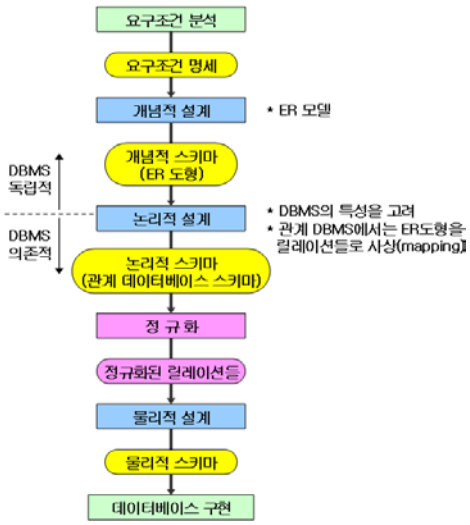
정규화 과정



종 류	설 명
제1정규형 (1NF)	• 어떤 릴레이션에 속한 모든 도메인이 원자값(atomic value)만으로 되어 있는 릴레이션
제2정규형 (2NF)	• 제1정규형이고, 모든 속성들이 기본키에 완전 함수적 종속 관계를 만족
제3정규형 (3NF)	• 제2정규형이고, 모든 속성들이 기본키에 이행적 종속 관계가 아닌 경우
BCNF	• 릴레이션의 모든 결정자가 후보키인 릴레이션
제4정규형 (4NF)	• • 릴레이션에서 다치 종속 관계가 성립되는 경우
제5정규형 (5NF)	• 어떤 릴레이션에 조인 종속이 성립하는 경우

2.5 물리적 데이터베이스 설계

데이터베이스 설계 과정



개념적 설계
- 산출물로 ER-다이어그램이 만들어짐
- DBMS에 독립적인 개념 스키마를 설계

논리적 설계
- DBMS에 따라 서로 다른 논리적 스키마를 정의
- 현실 세계를 표현하기 위한 데이터베이스의 논리적 구조, 즉 정규화 과정을 이용한 릴레이션의 속성을 결정하는 단계
- 논리적 설계 단계에서 수행되는 작업
  - 논리적 데이터 모델로 변환
  - 트랜잭션 인터페이스 설계
  - 스키마의 평가 및 정제

물리적 설계
- 목표 DBMS에 맞는 물리적 구조 설계
- 물리적 설계 단계에서 수행되는 작업
  - 저장레코드 양식 설계
  - 접근 경로 설계
  - 레코드 집중의 분석 및 설계
  - 파일의 저장 구조 및 탐색 기법
- 물리적 설계 단계의 고려사항
  - 어떤 인덱스를 만들 것인지에 대한 고려
  - 성능 향상을 위한 개념 스키마의 변경 여부 검토
  - 레코드의 크기
  - 파일과 구조 저장을 위한 최소한의 효율적 공간

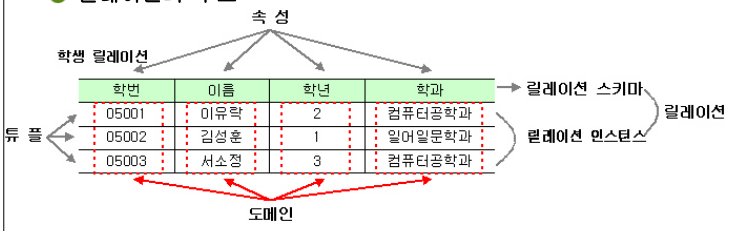
구현
- 목표 DBMS DDL로 스키마 작성
- 응용프로그램을 위한 트랜잭션 작성

3 관계 데이터베이스 모델과 언어

3.1 관계 데이터 모델

관계 데이터 모델 정의
- 개체 집합에 대한 속성 관계를 표현하기 위해 개체를 테이블(table)로 사용하고 개체 집합들 사이의 관계를 공통속성으로 연결하는 독립된 형태의 데이터 모델
- 논리적인 데이터 모델에서 데이터간의 관계를 기본키(primary key)와 이를 참조하는 외래키(foreign key)로 표현하는 데이터 모델

릴레이션의 구조



- 개체 (Entity)
  - 데이터베이스가 표현하려고 하는 유형, 무형의 정보대상으로 존재하면서 서로 구별될 수 있는 것
  - 데이터베이스에 표현하려고하는 현실 세계의 대상체
- 속성 (attribute)
  - 개체의 특성을 기술
  - 데이터의 가장 작은 논리적 단위로서 파일 구조상의 데이터 항목 또는 데이터 필드에 해당됨
- 도메인 (domain)
  - 하나의 애트리뷰트가 취할 수 있는 같은 타입의 원자(atomic) 값들의 집합
  - 표현되는 속성 값의 범위를 나타냄
  - 실제 애트리뷰트 값이 나타날 때 그 값의 합법여부를 시스템이 검사하는 데에도 이용됨
- 튜플 (Tuple)
  - 데이터베이스의 관계모형에서 사용하는 테이블의 행을 구성하는 애트리뷰트 값들의 집합
  - 테이블에서 행(레코드)과 유사
- 관계 (relation)
  - 개체간의 상호 작용을 나타냄
- 차수 (degree)
  - 릴레이션에서 속성의 수
- 카디널리티 (cardinality)
  - 릴레이션에 포함되어 있는 튜플의 수

릴레이션의 특성
- 한 릴레이션에 포함된 튜플들은 모두 상이함 (유일성을 가짐)
- 한 릴레이션에 포함된 튜플 사이에는 순서가 없음
- 모든 속성 값은 원자값
- 속성은 릴레이션 내에서 유일한 이름을 가짐
- 속성들 간에는 순서가 없음

키(KEY)의 개념
- 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 기준이 되는 속성

키(KEY)의 종류
- 기본키 (primary key)
  - 테이블의 유일한 식별자
  - 후보키 중에서 특별히 선정된 키로 중복값을 가질 수 없음
  - 후보키의 성질을 가짐 (유일성, 최소성 모두 만족)
  - NULL 값을 가질 수 없음
  - 외래키로 참조됨

- 외래키 (foreign key)
  - 어떤 릴레이션 R1의 기본키의 값들과 일치함을 요구하는 다른 릴레이션 R2의 한 속성
  - 외래키는 참조 릴레이션의 기본키와 동일한 키 속성을 가짐
- 후보키 (candidate key)
  - 릴레이션에 있는 모든 튜플들을 유일하게 식별할 수 있는 하나 또는 몇 개의 애트리뷰트 집합
  - 튜플을 유일하게 구분할 수 있는 최소 슈퍼키(하나의 속성으로 이루어짐)
  - 유일성과 최소성 모두 만족
- 대체키 (alternate key)
  - 후보키가 둘 이상 되는 경우에 그 중에서 어느 하나를 선정하여 기본키라 지정하면, 나머지 후보키들은 대체키가 됨(후보키 - 기본키 = 대체키)
- 슈퍼키 (super key)
  - 유일성만 있고 최소성이 없는 애트리뷰트 집합
  - 한 릴레이션 내에 있는 속성들의 집합으로 구성된 키

무결성 (Integrity)

- 데이터베이스를 정확하고 유효하게 유지 (데이터의 정확성)
- 제약조건에 의해 무결성 유지
- 무결성 규정에는 규정이름, 검사 시기, 제약조건 등을 명시

무결성 제약조건

- 개체 무결성 제약조건
  - 한 릴레이션의 기본키를 구성하는 어떠한 속성 값도 널(NULL)값이나 중복 값을 가질 수 없다는 것을 의미
- 참조 무결성 제약조건
  - 릴레이션 R1에 저장된 튜플이 릴레이션 R2에 있는 튜플을 참조하려면 참조되는 튜플이 반드시 R2에 존재해야 한다는 데이터 무결성 규칙 (R2의 기본키를 R1의 외래키로 참조함)
  - 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없음

3.2 관계 데이터 언어

관계대수 정의

- 원하는 정보와 그 정보를 어떻게 유도하는가를 기술하는 절차적인 방법
- 주어진 관계로 부터 원하는 관계를 얻기 위해 연산자와 연산규칙을 제공하는 언어

순수 관계 연산자

- 디비전 (Division)
  - X ⊃ Y인 2개의 R에서 R(X)와 S(Y)가 있을 때 R의 속성이 S의 속성값을 모두가 가진 튜플에서 S가 가진 속성을 구하는 연산자
  - 표기: R[속성r ÷ 속성s]S

- 프로젝션 (Projection)
  - 테이블에서 특정 속성에 해당하는 열을 선택하는데 사용되며 결과로는 릴레이션의 수직적 부분 집합에 해당하는 관계 대수 연산자
  - 표기 : $\pi_{(속성리스트)}(R)$
- 조인 (Join)
  - 공통된 속성 기준 2개의 R ⇨ 하나로 만듦
  - 표기 : $R \bowtie_{귀속성r = 귀속성s} S$
- 실렉션 (Selection)
  - 조건을 만족하는 릴레이션의 수평적 부분집합(행을 구하는 연산) 으로 구성하여, 연산자의 기호는 그리스문자 시그마(σ)를 사용
  - 표기 : $\sigma_{<조건>}(R)$

카티션 곱 (Cartesian Product)

- 두 릴레이션(Relation)의 교차 곱을 수행
- 릴레이션 R1과 R2를 카티션 곱을 하면 각 릴레이션의 튜플 수를 곱한 것과 같은 개수의 결과 튜플이 생김

관계해석(Relational Calculus)의 특징

- 튜플 관계 해석과 도메인 관계 해석이 있음
- 기본적으로 관계 해석과 관계 대수는 관계 데이터베이스를 처리하는 기능과 능력 면에서 동등함
- 수학의 predicate calculus에 기반을 두고 있음
- 관계 해석으로 질의어를 표현
- 원하는 릴레이션을 정의하는 방법을 제공, 즉 원하는 정보가 무엇이라는 것만 정의하는 비절차적인 언어
- 연산들의 절차(sequence)를 사용하여 데이터를 가져옴
- 기본적인 연산자로 UNION, INTERSECTION, DIFFERENCE를 사용함
- 전체 관계를 조작하는데 사용되는 연산들의 집합

3.3 SQL 언어

SQL 분류

- DDL (Data Definition Language, 데이터 정의어)

명령어	기 능
CREATE	SCHEMA, DOMAIN, TABLE, VIEW, INDEX 정의
ALTER	TABLE 정의 변경
DROP	SCHEMA, DOMAIN, TABLE, VIEW, INDEX 삭제

- DML (Data Manipulation Language, 데이터 조작어)

명령어	기 능
SELECT	튜플 검색
INSERT	튜플 삽입
DELETE	튜플 삭제
UPDATE	튜플 변경

- DCL (Data Control Languagem, 데이터 제어어)

명령어	기 능
COMMIT	정상적인 완료
ROLLBACK	트랜잭션 실패, 원래의 상태로 돌아감
GRANT	사용권한 부여
REVOKE	사용권한 취소

DDL

- CREATE TABLE
  - 테이블을 정의
  - 기본 구조

```
Create Table 테이블이름 {
 (속성명 data-type),
 Primary key(기본키 속성명),
 Unique(대체키 속성명),
 Foreign key(외래키 속성명),
       References 참조테이블(키본키 속성명),
 Check (조건식 );
};
```

- CREATE VIEW
  - 뷰를 정의
  - 기본 구조

```
Create View 뷰이름(속성명, ...)
 As Select 속성명, ...
 From 테이블명
 where 조건;
[With Check option]
```


- ALTER TABLE
  - 테이블에 대한 정의를 변경
  - 기본 구조

```
Alter Table 테이블이름  ADD    속성이름  data-type [default 값];
Alter Table 테이블이름  ALTER 속성이름  data-type [set default 값];
Alter Table 테이블이름  DROP   속성이름  data-type [cascade];
```

- DROP
  - 스키마, 도메인, 테이블, 뷰, 인덱스를 제거하는 명령문
  - 기본 구조

```
Drop [ SCHEMA | DOMAIN | TABLE | VIEW | INDEX ]  이름 [CASCADE | RESTRICT]
```

  - CASCADE: 제거할 개체를 참조하는 다른 모든 개체를 함께 제거
    RESTRICT: 다른 개체가 제거할 개체를 참조중일 경우 제거가 취소됨⇨참조 무결성 위배 방지

DML

- SELECT
  - 테이블을 구성하는 튜플들 중에서 조건을 만족하는 튜플을 검색하여 임시 테이블을 구성하는 명령
  - 기본 구조

```
SELECT   속성명, ...
FROM     테이블명, ...
[WHERE 조건]
```

  - 확장 구조

```
SELECT   속성명, ...
FROM     테이블명, ...
[WHERE 조건]
[GROUP BY 속성명, ...
[HAVING 조건]
[ORDER BY 속성명[ASC|DESC];
```

- INSERT
  - 데이터베이스에 저장된 자료(튜플)를 검색 삽입 삭제 갱신 재구성하기 위한 언어
  - 기본 구조

```
INSERT  INTO  테이블명(속성명,…)  VALUES(데이터값,…);
```

- UPDATE
  - 테이블에 있는 튜플들 중에 특정 튜플의 내용을 갱신할 때 사용
  - 기본 구조

```
UPDATE 테이블명 Set 속성명 = 데이터값 Where 조건;
```

- DELETE
  - 테이블에 있는 튜플들 중에서 특정 튜플을 삭제할 때 사용하는 명령문
  - 기본 구조

```
DELETE FROM 테이블명 WHERE 조건;
```

DCL

- GRANT
  - 데이터베이스 사용자에게 사용권한 부여
  - 기본 구조

```
GRANT 권한 ON 테이블명 TO 사용자명 [WITH GRANT OPTION];
```

- REVOKE
  - 데이터베이스 사용자의 사용 권한을 취소
  - 기본 구조

```
REVOKE 권한 ON 테이블명 FROM 사용자명 [CASCADE];
```

내장(삽입) SQL

- 응용 프로그램 내에 데이터를 정의하거나 질의하는 SQL 문장을 내포하여 프로그램이 실행될 때 함께 실행되도록 함
- Host Program의 컴파일 시 선행처리기에 의해 내장 SQL문은 분리되어 컴파일 됨 (내장 SQL 프로그램은 컴파일보다 우선하는 전 처리기에 의해 처리됨)
- 호스트 변수와 데이터베이스 필드의 이름은 같아도 됨
- 호스트 언어의 실행문이 나타날 수 있는 곳이면 어디든지 나타날 수 있음
- 삽입 SQL문은 호스트 변수를 포함할 수 있음
- SQL 문장의 식별자로서 EXEC SQL을 앞에 기술함
- SQL에 사용되는 호스트 변수는 콜론(:)을 앞에 붙임
- SQL code의 값이 영(제로)이면 성공적으로 수행되었음을 의미
- 호스트 변수의 데이터 타입은 이에 대응하는 데이터베이스 필드의 SQL 데이터 타입과 일치해야 함

3.4 시스템 카탈로그와 뷰

시스템 카탈로그(System Catalog)

- 데이터베이스 시스템에서 데이터가 실제로 읽혀지거나 수정되기 전에 먼저 참고 되는 파일
- 카탈로그에 저장된 데이터를 메타데이터라고 함
- 카탈로그가 생성되면 자료 사전(Data Dictionary)에 저장 되므로 좁은 의미로 자료사전이라 함

시스템 카탈로그의 특징

- 시스템 그 자체에 관련이 있는 다양한 객체들에 관한 정보를 포함하는 파일 시스템
- 분산 시스템에서 카탈로그는 보통의 릴레이션, 인덱스, 사용자 등의 정보를 포함할 뿐 아니라 위치 단편화 및 중복 독립성을 제공하기 위해 필요한 모든 제어 정보를 가짐
- 카탈로그 자체도 시스템 테이블로 구성되어 있어 일반 이용자도 SQL을 이용하여 내용을 검색해 볼 수 있음
- DBMS가 스스로 생성하고, 유지하는 데이터베이스 내의 특별한 테이블의 집합체
- 데이터베이스 스키마에 대한 정보를 제공
- 테이블정보, 인덱스 정보, 뷰 정보 등을 저장하는 시스템 테이블
- 시스템 카탈로그에 대한 갱신은 DBMS가 자동적으로 수행
- 개체들로서는 기본 테이블, 뷰, 인덱스, 데이터베이스, 패키지, 접근 권한 등이 있음

데이터 디렉토리

- 데이터 사전에 수록된 데이터를 실제로 접근하는데 필요한 정보를 관리 유지하는 시스템
- 시스템만 접근할 수 있음

뷰(VIEW)

- 하나 이상의 기본 테이블로부터 유도된 가상 테이블(기본 테이블의 열들로 구성)
- 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위한 테이블임

뷰(VIEW)의 특징

- 뷰를 이용한 또 다른 뷰의 생성이 가능
- 뷰의 활용은 테이블과 동일
- 뷰는 create view 명령을 사용하여 정의함
- 삽입, 갱신, 삭제 연산에는 제약이 따름(뷰의 정의는 ALTER 문을 이용하여 변경할 수 없음)
- 논리적 데이터에 대한 독립성이 보장
- 논리적 데이터 독립성을 제공함
- 접근 제어를 통한 보안을 제공함
- 사용자의 데이터 관리를 간단하게 해줌
- 여러 사용자의 상이한 응용이나 요구를 편리하게 지원해 줌
- 숨겨진 데이터를 위한 자동 보안이 제공됨
- 동일 데이터를 다양하게 표현할 수 있음

4 관계 데이터베이스 고급 기능

4.1 트랜잭션

트랜잭션(Transaction)의 정의

- 트랜잭션은 데이터베이스에서 복구 및 병행 시행시 처리되는 작업의 논리적 단위
- 하나의 트랜잭션은 Commit(완료)되거나 Rollback(복귀)되어야 함

트랜잭션의 속성

종 류	설 명
원자성 (atomicity)	• 완전하게 수행 완료되지 않으면 전혀 수행되지 않아야 함 (ALL or NOTHING) • 트랜잭션은 일부만 수행된 상태로 종료되어서는 안 됨
일관성 (consistency)	• 트랜잭션의 실행은 데이터베이스의 일관성을 유지해야 함
독립성 (isolation, 격리성)	• 임의의 트랜잭션은 동시에 수행되는 다른 트랜잭션에 방해 받아서는 안 됨
영속성 (durability)	• 트랜잭션이 일단 그 실행을 성공적으로 완료하면 그 결과는 영속적이어야 함

COMMIT

- 한 작업의 논리적 단위가 성공적으로 끝났고, 데이터베이스가 다시 일관된 상태에 있으며 이 트랜잭션이 행한 갱신 연산이 완료된 것을 트랜잭션 관리자에게 알려주는 연산
- SQL 명령어로 수행된 결과를 실제 물리적 디스크로 저장하는 SQL 명령

ROLLBACK

- 트랜잭션의 실행이 실패하였음을 알리는 연산자로 트랜잭션이 수행한 결과를 원래의 상태로 원상 복귀시키는 연산

4.2 데이터베이스 제어

보안의 특성

- 보안을 위한 데이터 단위는 테이블 전체로부터 특정 테이블의 특정한 행과 열 위치에 있는 특정한 데이터 값에 이르기까지 다양함
- 각 사용자들은 일반적으로 서로 다른 객체에 대하여 다른 접근권리 또는 권한을 갖게 됨
- SQL의 경우에는 보안규정에 포함된 독립적인 기능으로 뷰 기법(view mechanism)과 권한인가 서브시스템(authorization subsystem)이 있음

병행제어의 목적

- 데이터베이스의 공유 최대화
- 시스템의 활용도 최대화
- 사용자에 대한 응답시간 최소화

병행제어 기법에 의한 트랜잭션 제어로 막을 수 있는 문제점

- 갱신 분실 (lost update)
- 모순성 (inconsistency)
- 연쇄 복귀 (cascading rollback)
- 비완료 의존성문제 (uncommitted dependency problem)
- 불일치 분석문제 (inconsistent analysis problem)

로킹 (Locking)

- 데이터의 접근을 상호 배타적으로 만들어 병행 제어를 하는 방법
- 로킹 단위가 크면 병행성 수준이 낮아짐
  - 로킹의 병행성 수준
    페이지 차원(Page-level)의 잠금 > 테이블 차원(table-level)의 잠금 > 행 차원(row-level)의 잠금

4.3 분산 데이터베이스

분산 데이터베이스 정의

- 논리적으로는 하나의 시스템에 속하지만, 물리적으로는 네트워크에 연결하여 여러 개의 컴퓨터 사이트에 분산되어 있는 데이터의 집합

분산 데이터베이스 목표

종 류	설 명
위치 투명성	• 데이터가 물리적으로 저장되어 있는 곳을 알 필요 없이 논리적인 입장에서 데이터가 모두 자신의 사이트에 있는 것처럼 처리하는 특성
중복 투명성 (replication transparency)	• 트랜잭션이 데이터의 중복 개수나 중복 사실을 모르고도 데이터 처리가 가능함
병행 투명성 (concurrency transparency)	• 분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실현되더라도 그 트랜잭션의 결과는 영향을 안 받음
장애 투명성 (failure transparency)	• 트랜잭션, DBMS, 네트워크, 컴퓨터 장애에도 불구하고 트랜잭션을 정확하게 처리함

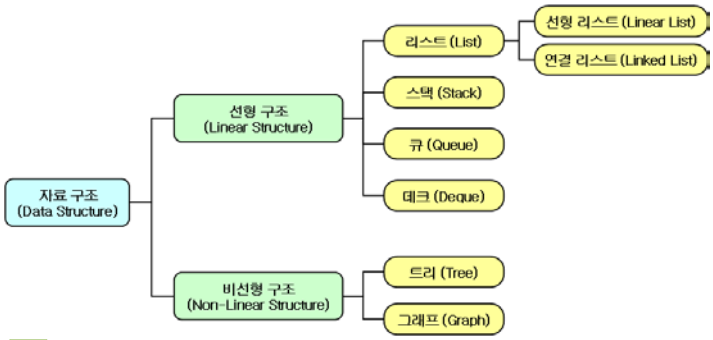
분산 데이터베이스 특징

- 지역 자치성이 높음
- 효용성과 융통성이 높음
- 점진적 시스템 용량 확장이 용이
- 신뢰성과 가용성이 높음
- 특정한 사이트에서 장애가 발생하더라도 다른 사이트는 계속 운용할 수 있음
- 데이터의 공유성 향상
- 질의처리(query processing) 시간의 단축
- 분산제어가 가능하고 시스템의 성능이 향상됨

5 자료구조

5.1 자료구조의 개념

자료구조의 분류



5.2 선형 구조

선형 리스트 (Linear List)

- 연속적인 기억장소에 저장된 리스트
- 형태: 임의의 노드에 접근을 할 때는 인덱스(Index)를 사용하므로 포인터가 없음
- 장점
  - 간단한 자료구조
  - 저장 효율이 뛰어남 (기록밀도: 1)
  - 접근 속도(Access Time)가 빠름
- 단점
  - 삽입과 삭제가 어려움(삽입 및 삭제 시 삽입하거나 삭제할 위치 이후의 모든 자료의 이동이 필요)

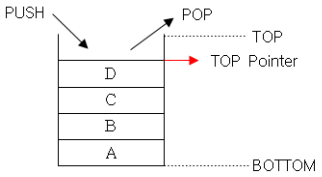
연결 리스트 (Linked List)

- 자료들을 임의의 기억공간에 기억시키고, 자료 항목의 순서에 따라 노드의 포인터 부분을 이용하여 서로 연결시킨 자료구조
- 형태
  - 각 노드는 다음 노드를 가리키는 링크(Link , Pointer) 정보를 가짐
  - 마지막 노드는 포인터 정보를 Null 값을 가짐

- 장점
  - 자료의 삽입 및 삭제가 용이
  - 비연속적(한 리스트를 여러 개의 리스트로 분리하기 쉬움)
  - 희소 행렬(행렬의 요소들 중에서 많은 부분이 0으로 되어 있는 행렬)을 연결리스트로 표현 시 기억장소 이용효율이 좋음
- 단점
  - Access Time이 느림
  - 기억장소 이용 효율이 나쁨 (저장되지 않은 빈 공간 발생)
  - 포인터를 위한 추가 공간이 필요
  - 중간 노드 연결이 끊어지면 그 다음 노드를 찾기 힘듦

스택(Stack)

- 리스트 내의 자료 삽입, 삭제가 한쪽 끝에서 이루어지는 자료 구조
- 자료의 후입선출(last-in-first-out) 방법

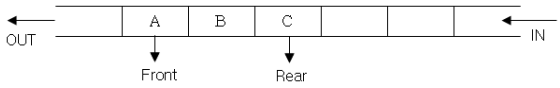


Top : 자료의 삽입과 삭제가 이루어지는 스택공간의 위치를 표시하는 포인터
Push : 스택에서 자료의 삽입
Pop : 스택에서 자료의 삭제

- 자료의 삽입: TOP = TOP + 1
- 자료의 삭제: TOP = TOP - 1
- Overflow 발생: 스택의 크기가 M일 때 , TOP > M 이면 Overflow 발생
- 용도
  - 인터럽트의 처리
  - 수식의 계산 (산술식 표현)
  - 서브루틴의 복귀번지 저장 (함수 호출의 순서 제어)

큐 (Queue)

- 삽입 작업이 선형 리스트의 한쪽 끝(rear)에서 이루어지고, 삭제 작업은 다른쪽 끝(front)에서 수행되는 자료 구조
- 운영체제의 작업 스케줄링 등에 응용되는 것으로 가장 적합한 자료구조
- FIFO 구조
- 형태



데크 (Deque, Double Ended Queue)

- 서로 다른 방향에서 입·출력이 가능한 구조 (삽입과 삭제가 양쪽 끝에서 일어남)
- 입력이 한쪽에서만 발생하고 출력은 양쪽에서 일어날 수 있는 입력제한과 입력은 양쪽에서 일어나고 출력은 한곳에서만 이루어지는 출력제한이 있음
- 스택과 큐를 복합한 형태

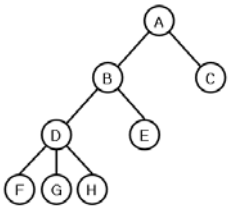
- 입력제한데크 : 입력이 한 쪽 끝으로 제한 (Scroll)
- 출력제한데크 : 출력이 한 쪽 끝으로 제한 (Shelf)

5.3 비선형 구조

트리(Tree)

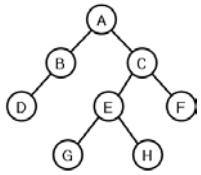
- Linked List로 표현할 때 가장 효율적임
- 계층형 구조(hierarchical structure)를 나타내기 편리함

트리 용어정리



용어	설명
노드 (Node)	Tree 의 기본 구성요소 (A, B, C, D, E, F, G, H)
근노드 (Root Node)	가장 상위에 위치한 노드 (위 트리에서는 A)
레벨 (Level)	근노드를 기준으로 특정 노드까지의 경로길이 (E의 레벨은 3)
조상노드 (Ancestors Node)	특정 노드에서 루트에 이르는 경로상의 모든 노드 (D의 조상노드는 B, A)
자식노드 (Son Node)	특정 노드에 연결된 다음 레벨의 노드 (B의 자식노드는 D, E)
부모노드 (Parent Node)	특정 노드에 연결된 이전 레벨의 노드 (F의 부모노드는 D)
형제노드 (Sibling)	같은 부모를 가진 노드 (F의 형제노드는 G , H)
깊이 (Depth, Height)	트리의 최대 레벨 (위 트리의 깊이는 4)
차수 (Degree)	특정 노드에 연결된 자식노드의 수 (D의 차수는 3)
단말노드 (Terminal Node, Leaf Node)	트리의 제일 마지막에 위치한 노드 위 트리에서는 F, G, H, E, C
트리의 차수	트리의 노드 중 가장 큰 차수 위 트리의 차수는 3 (D의 차수가 가장 크므로 D의 차수가 트리의 차수)

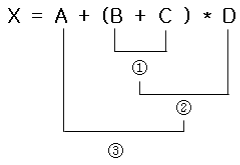
트리의 운행법(Traversal)의 종류

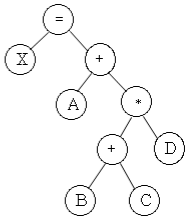


- preorder: root→left→right (A B D C E G H F)
- inorder: left→root→right (D B A G E H C F)
- postorder: left→right→root (D B G H E F C A)

수식의 표기법

- 표기법의 종류
  - 중위 표기법(Infix notation): 연산자가 피연산자 사이에 있는 표기법
  - 후위 표기법(Postfix notation, reverse Polish notation): 피연산자 뒤에 연산자가 표기되는 표기법
  - 전위 표기법 (Prefix notation): 피연산자들 앞에 연산자를 표시하는 표기법





▷ 중위 표기법
A + B + C * D
▷ 후위 표기법
A B C + D * +
▷ 전위 표기법
+ A * + B C D

- 중위 표기법 → 후위 표기법
  예) A/B-(C*D)/E
  AB/ CD*
  CD*E/
  AB/CD*E/-

이진 트리
- 정의: Degree가 2이하로 구성된 트리
- 특성
  - 깊이가 k인 이진 트리의 최대노드의 수 : 2^k-1
  - 이진 트리의 레벨 i 에서 최대노드의 수 : $2^{(i-1)}$
  - $n_0 = n_2 + 1$ (n_0: 이진 트리의 터미널노드, n_2: 차수가 2인 노드 수)

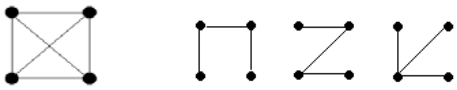
스레드(thread) 이진 트리
- 비순환적인 이진트리 운행 알고리즘에서 스택을 사용하지 않고 낭비되는 널(NULL) 연결 필드를 활용하여 트리 운행에 필요한 다른 노드의 포인터로 사용하도록 고안된 이진트리

B-트리
- 한 노드 안에 있는 키 값은 오름차순을 유지함
- 루트와 리프(leaf)를 제외한 모든 노드는 최소 (m/2)개, 최대 m개의 서브트리를 가짐
- 루트(root) 노드는 리프가 아닌 이상 적어도 두 개의 서브트리를 가짐
- 탐색, 추가, 삭제는 루트로부터 시작함
- 삽입과 삭제를 하여도 데이터 구조의 균형을 유지해야 함
- 순차 탐색은 각 노드를 중위 순회함으로써 좋은 성능을 발휘하지 못함
- B-트리는 인덱스 파일에서 인덱스를 구성하는 방법 중의 하나임

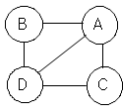
B+트리
- B-트리의 추가, 삭제 시 발생하는 노드의 분열과 합병 연산 과정을 줄일 수 있는 트리구조
- 단말노드가 아닌 인덱스 세트와 단말로드로 구성된 데이터 세트로 이루어짐

신장트리
- 그래프에서 간선들이 사이클이 되지 않도록 만든 트리

예)



그래프 (Graph)
- 그래프의 정의
  - 각각의 단위 정보를 링크로 연결하여 구조화시킨 자료 구조
  - 정점(vertex): 노드들의 집합
  - 간선(edge): 정점들 사이의 상호 연결의 집합, 임의의 점들의 쌍을 연결
- 인접행렬 (Adjacency Matrix)



	A	B	C	D
A	0	1	1	1
B	1	0	0	1
C	1	0	0	1
D	1	1	1	0

5.4 정렬

내부정렬기법
- 정의: 데이터량이 적을 때 주기억장치 내에서 정렬하는 방법
- 특징: 속도는 빠르나, 정렬할 자료의 양이 많은 경우 부적합
- 종류
  - 버블 정렬(bubble sort)
  - 삽입 정렬 (insertion sort)
  - 선택 정렬(selection sort)
  - 히프 정렬(heap sort)
  - 퀵 정렬(quick sort)
  - 기수 정렬(radix sort)
  - 2-way 합병 정렬

외부정렬기법
- 정의: 대용량의 데이터를 몇 개의 서브 파일로 나누어 각각 내부 정렬을 한 후에, 테이프나 디스크 내에서 각 서브 파일을 합병하는 방법
- 특징: 속도는 느리지만 정렬하고자 하는 자료의 양이 많을 경우 효과적 (주기억장치 + 보조기억장치)
- 종류
  - 진동 병합 정렬(oscillating merge sort)
  - 밸런스 병합 정렬(balanced merge sort)
  - 캐스캐이드 병합 정렬(cascade merge sort)
  - 폴리파즈 병합 정렬(polyphase merge sort)

정렬 알고리즘의 선택 시 고려사항
- 키 값들의 분포상태
- 소요 공간 및 작업시간
- 정렬에 필요한 기억공간의 크기
- 데이터의 양
- 초기 데이터의 배열상태
- 사용 컴퓨터 시스템의 특성

5.5 검색

이진 검색 (binary search)
- 일정한 순서로 배열된 레코드를 2개 부분으로 되풀이하여 나누어서, 한 부분은 버리고 남은 부분을 검색하는 방법
- 자료가 반드시 정렬되어 있어야 함 (이진 검색의 선행 조건)

보간(Interpolation) 검색
- 찾고자 하는 레코드 키가 있음직한 위치를 추정하여 검색하는 방법

해싱 (Hashing)
- 키 값으로부터 레코드가 저장되어 있는 주소를 직접 계산하여, 산출된 주소로 바로 접근하는 방법, 키-주소 변환 방법이라고도 함
- 검색 방법 중 속도는 가장 빠르지만 충돌현상 시 오버플로 해결의 부담이 과중되며, 많은 기억공간을 요구하는 검색 방법
- 버킷 (bucket): 하나의 주소를 갖는 파일의 한 구역을 의미하며, 버킷의 크기는 같은 주소에 포함될 수 있는 레코드 수를 의미
- 슬롯(slot): 한 개의 레코드를 저장할 수 있는 공간으로 n개의 슬롯이 모여 하나의 버킷을 형성
- 충돌(collision): 레코드를 삽입할 때 2개의 상이한 레코드가 똑같은 주소로 해싱 되는 것을 의미
- 동의어(synonym): 해싱 함수의 값이 같은 키들의 집합

해싱함수의 종류
- 중간제곱(mid-square) 방법
- 숫자분석(digit analysis) 방법
- 제산(division) 방법
- 중첩(접지) 방법(Folding method)
- 계수 분석 방법(Digit Analysis Method)
- 기수(Radix) 변환법

5.6 파일조직 기법

인덱스 (Index)
- 검색을 빠르게 하기 위해 만든 보조적인 데이터 구조
- 특징
  - 인덱스는 하나 이상의 필드로 만들어도 됨
  - 인덱스를 통해서 테이블의 레코드에 대한 액세스를 빠르게 수행할 수 있음

트라이 (trie)
- 검색을 위한 킷값을 직접 표현하지 않고 키를 구성하는 문자나 숫자 자체의 순서로 킷값을 구성하는 구조
- 트라이의 차수는 키 값을 표현하기 위해 사용하는 문자의 수(radix)에 의해 결정함
- 키 값의 분포를 미리 예측할 수 있다면 기억장소를 절약할 수 있음

인덱스 구분
- 정적 인덱스
  - 데이터 파일의 레코드가 삽입되거나 삭제됨에 따라 인덱스의 내용은 변하지만 구조자체는 변하지 않음
- 동적 인덱스
  - 인덱스나 데이터파일을 블록으로 구성하고 각 블록에는 추가로 삽입될 레코드를 감안하여 빈 공간을 미리 예비해두는 인덱스 방법

순차 파일 (Sequential file)
- 생성되는 순서에 따라 레코드를 순차적으로 저장하므로, 저장 매체의 효율이 가장 높음
- 프로그래밍이 쉬우며, 여러 개의 기록 매체에 기록이 가능

직접파일 (Direct file)
- 특정 레코드에 접근하기 위해서 디스크의 물리적주소로 변환할 수 있는 함수를 사용
- 해싱을 이용한 파일구조

직접 접근 방식 (DAM: Directed Access Method)
- 데이터의 입/출력이 빈번히 발생하는 곳에 응용
- 해싱 함수를 이용하여 레코드의 저장 위치를 결정
- 다른 레코드를 참조하지 않고 어떤 레코드를 접근할 수 있음

인덱스 순차 파일 (ISAM, Indexed sequential access-method)
- 키 값에 따라 순차적으로 정렬된 데이터를 저장하는 데이터 지역(Data Area)과 이 지역에 대한 포인터를 가진 색인 지역(Index Area)으로 구성된 파일
- 인덱스를 저장하기 위한 공간과 오버플로우 처리를 위한 별도의 공간이 필요
- 기본 구역(Prime data area), 색인 구역(Index area), 오버플로 구역(Overflow area)으로 구성
  - 기본 데이터 구역은 데이터 레코드를 저장
  - 인덱스 구역은 데이터 구역에 대한 인덱스를 저장
  - 독립된 오버플로우 구역은 기본 데이터 구역에서 오버플로우 된 레코드를 저장
- 데이터 파일은 기본구역과 오버플로 구성으로 구성
- 실제 데이터 처리 외에 인덱스를 처리하는 추가적인 시간이 소모되므로 파일 처리 속도가 느림
- 순차 처리와 직접(랜덤) 처리가 모두 가능하지만 기억장소의 낭비 초래
- 인덱스 영역 구분
  - 트랙 인덱스
  - 마스터 인덱스
  - 실린더 인덱스

데이터통신

1 데이터 통신의 개념

1.1 데이터 통신의 개요

데이터(Data)
- 관찰, 측정을 통해 수집한 단순한 사실이나 결과값

정보 (Information)
- 여러 가지 데이터를 처리한 후, 특정 목적 수행을 위하여 체계화한 것

정보화
- 정보의 생성, 가공, 축적 및 활용 등의 정보 행위를 의도적으로 행하여 그 유용가치를 높이는 활동

데이터와 정보의 진화과정
- 데이터(Data) - 정보(Information) - 지식(Knowledge) - 지능(Intelligence)

통신 처리와 데이터 전송
- 통신 처리
  - 기계 대 기계의 통신에서 일어날 수 있는 과정으로써 속도변환, 프로토콜 변환, 포맷변환 등을 총칭
- 데이터 전송
  - 컴퓨터나 데이터 단말기에 의해 처리할 또는 처리된 정보의 전송
- 데이터 전송에 가장 많이 사용되는 부호
  - EBCDIC(확장 2진화 십진 코드), ASCII(아스키코드)

정보 통신(Information Communication)
- 컴퓨터와 통신기술의 결합에 의하여 통신처리기능은 물론이고, 정보처리기능에 정보의 변환, 저장과정이 추가된 형태의 통신
- 통신 기술의 발전, 정보량의 증대, 컴퓨터의 개발 등으로 정보통신이 급속히 발달

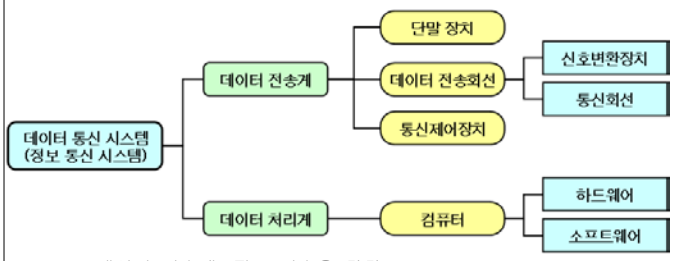
데이터 통신 (Data Communication)
- 정보기기 사이에서 디지털 신호형태로 표현된 정보를 송·수신하는 통신
- 정보처리장치 등에 의하여 처리된 정보를 전송하는 통신으로 기계장치 간의 통신
- 전기통신회선을 이용, 회선에 입·출력장치를 접속해서 정보를 송·수신하는 통신
- ITU-T의 데이터 통신에 관한 정의: 정보를 기계로 처리 하거나 처리한 정보를 전송 하는 것

통신의 구성 3요소
- 정보를 보내는 장소 (Source)
- 전송 매체 (Transmission Media)
- 정보를 수신하는 장소 (Destination)

정보 통신망 (Information Communication Network)
- 정보의 전달체계
- 정보 통신망의 3대 동작 기능: 전달 기능, 신호 기능, 제어 기능
- 정보 통신망의 3대 구성 요소: 단말 장치, 교환 장치, 전송 장치

1.2 데이터 통신 시스템

데이터 통신 시스템의 기본 구성 요소



- 데이터 전송계: 정보 전송을 담당
  - 단말장치(DTE): 통신 시스템과 사용자의 접점에 위치하여 데이터를 입력하거나 처리된 결과를 출력하는 기능을 하는 장치
  - 데이터 전송 회선(신호 변환 장치(DCE) + 통신 회선): 전송 신호를 송·수신하기 위한 통로
    - 신호 변환 장치: 단말 장치와 통신 회선 사이에서 적합한 신호나 데이터로 변환시켜주는 장치로서 데이터 회선 종단 장치라고도 함
    - 통신 회선: 데이터가 실질적으로 전송되는 선로로서, 꼬임선, 동축케이블, 광섬유 케이블, 라디오파, 마이크로파(Microwave)등의 전송매체가 있음
  - 통신 제어 장치(CCU)
    - 데이터 전송 회선과 컴퓨터를 연결하는 장치
    - 전송 오류 검출, 회선 감시등과 같은 통신제어 기능을 수행
- 데이터 처리계: 정보의 가공, 처리, 저장 등을 담당
  - 하드웨어 (중앙처리장치(CPU) + 주변장치)
  - 소프트웨어 (시스템 소프트웨어 + 응용 소프트웨어)

정보 통신 시스템의 3대 구성 요소
- 단말 장치, 전송 장치, **컴퓨터**

데이터 통신 시스템의 3가지 구성 요소
- 단말 장치, 전송 장치, **통신 제어 장치**

데이터 통신 시스템의 특징
- 고속 · 고품질의 통신 서비스 제공
- 고성능의 에러 제어 방식을 사용하여 시스템 신뢰도가 높음
- 거리와 시간의 한계 극복
- 대형 컴퓨터의 공동 이용
- 대용량 파일의 공동 이용
- 분산 처리 방법 활용
- 원격지의 정보처리기기 사이의 효율적 정보교환
- 정보통신망의 초고속화 및 글로벌화

데이터 통신 시스템의 발달 과정
- SAGE (Semi-Automatic Ground Environment)
- SABRE (Semi-Automatic Business Research Environment)
- CTSS (Compatible Time Sharing System)
- ARPANET (Advanced Research Project Agency Network)
- ALOHA (Additive Links On-line Hawaii Area)
- SNA (System Network Architecture)

2 데이터 통신 기기

2.1 단말 장치 (터미널)

단말 장치 (DTE, Data Terminal Equipment)
- 통신 시스템과 사용자의 접점에 위치하여 데이터를 입력하거나 처리된 결과를 출력하는 기능을 하는 장치
- 전화기, FAX, 휴대폰, 리모콘, 컴퓨터 등 다양함

단말 장치의 기능
- 입 · 출력 기능
  - 입력된 데이터를 2진 신호로 변환하고(입력), 처리된 데이터를 문자, 숫자, 영상의 형태로 변환(출력)하는 기능
- 전송 제어 기능
  - 통신망에 접속된 컴퓨터와 단말 장치 간에 효율적이고 원활한 정보를 교환하기 위하여 갖추어야 할 제어기능과 방식을 총칭
- 기억 기능
  - 입력된 데이터나 출력된 데이터를 잠시 보관하는 기능

단말 장치의 분류
- 일반적인 분류
  - 범용 단말 장치
  - 전용 단말 장치
  - 복합 단말 장치
    - · 리모트 배치 터미널(Remote Batch Termanal): 원거리에서 일괄 처리하는 시스템 터미널
- 프로그램 내장 유무에 따른 분류
  - 지능형
  - 비지능형

2.2 통신 제어 장치

통신 제어 장치의 기능
- 통신회선을 통하여 송 · 수신되는 자료를 전체적으로 제어하고 감시함
- 전송 제어: 통신의 시작과 종료 제어, 송신권 제어, 교환, 분기
- 동기 제어: 통신회선의 전송속도와 중앙처리장치의 처리속도 사이에서 조정을 수행
- 오류 제어: 데이터 전송 중 오류 검출 및 정정을 수행
- 흐름 제어, 응답 제어
- 제어 정보의 식별, 기밀 보호, 관리 기능

통신 제어 장치의 종류
- 통신 제어 장치 (CCU, Communication Control Unit)
  - 데이터 전송회선과 컴퓨터와의 전기적 결합과 전송문자를 조립, 분해하는 장치
- 통신 제어 처리 장치 (CCP, Communication Control Processor)
  - 문자 외에 메시지의 조립과 분해 기능을 수행하는 장치
- 전처리기 (FEP, Front End Processor)
  - 호스트 컴퓨터와 단말기 사이에 고속 통신 회선으로 설치
  - 통신 회선 및 단말기 제어, 메시지 조립과 분해, 전송 메시지 검사 등을 수행
  - 경쟁(Contention) 방식, 순서적 할당 방식, 시간 분할 방식으로 동작
  - 전단제어장치라고도 함

통신 제어 장치의 분류
- 비트 버퍼(Bit Buffer) 방식
- 문자 버퍼(Character Buffer) 방식
- 블록 버퍼(Block Buffer) 방식
- 메시지 버퍼(Message Buffer) 방식

2.3 신호 변환 장치

신호 변환 장치 (DCE, Data Circuit Equipment)
- 단말장치와 통신제어장치를 통신회선에 접속하는 장치
- 컴퓨터나 단말 장치의 데이터를 통신 회선에 적합한 신호로 변경하거나, 그 반대의 기능을 수행

모뎀 (MODEM)
- 디지털 신호를 음성대역(0.3 ~ 3.4㎑)내의 아날로그 신호로 변환(변조)한 후 음성 전송용으로 설계된 전송로에 송신한다든지 반대로 전송로부터의 아날로그 신호를 디지털 신호로 변환(복조) 하는 장치

코덱 (CODEC)
- 아날로그 데이터를 전송하기 위해 디지털 신호 형태로 변환시키고 또 이러한 디지털 형태를 원래의 아날로그 데이터로 복구시키는 장치

DSU (Digital Service Unit)
- 전송회선 양단의 데이터 회선 종단장치로서 단말에서 출력되는 디지털 데이터를 디지털 전송에 적합한 신호형식으로 변화하거나 또는 그 반대의 동작을 하는 장치
- 디지털 전송로에서 단극성(유니폴라) 신호를 쌍극성(바이폴라) 신호로 변환이 가능
- 데이터 전송을 위해서 필요성이 증대되고 있음

2.4 DTE / DCE 접속 규격

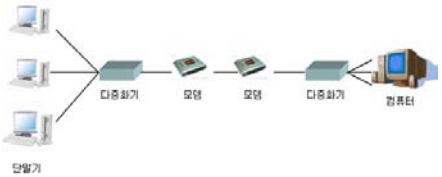
접속 규격 표준안
- ITU-T (국제전기통신연합 전기통신표준화 부문, International Telecommunication Union - Telecommunication)
  - V 시리즈
  - X 시리즈
- EIA (전자공업협회, Electronic Industries Association)
  - RS-232C
    - · 공중전화 교환망(PSTN)을 통한 DTE / DCE 접속 규격
    - · ISO2110, V.24, V.28을 사용하는 접속 규격
- ISO (국제표준화기구, International Standards Organization)

RS-232C
- 데이터 회선종단장치(신호변환장치, DCE)와 단말장치(터미널, DTE)사이의 물리적 전기적 접속규격
- 25핀(PIN)으로 구성된 커넥터

2.5 다중화기

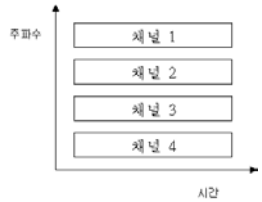
다중화 (Multiplexing)
- 하나의 통신 회선에 다수의 터미널이 공유할 수 있도록 하는 것
- 다중화의 가장 큰 장점: 선로의 공동 이용이 가능하므로 전송 효율이 높아짐
- 다중화를 위한 장치: 다중화기, 집중화기, 선로 공동 이용기

다중화기 (MUX, MUltipleXer)

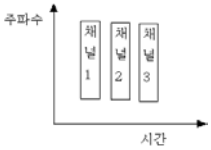


- 여러 개의 터미널 신호를 하나의 통신회선을 통해 전송할 수 있도록 하는 장치
- 여러 개의 채널들이 하나의 통신 회선을 통하여 결합된 신호의 형태로 전송되고 수신측에서 다시 이를 여러 개의 채널 신호로 분리하는 역할을 수행

다중화 기법
- 주파수 분할 다중화 (FDM, Frequency-Division Multiplexing)
  - 통신 회선의 주파수를 여러 개로 분할하는 방식



- 시분할 다중화 (TDM, Time-Division Multiplexing)
  - 한 전송로의 데이터 전송 시간을 일정한 시간 폭(Time Slot)으로 나누어 각 부 채널에 차례로 분배하는 방식



- 부호 분할 다중화 (CDM, Code-Division Multiplexing)
  - 부호를 분리시켜 하나의 연결선을 통하여 여러 신호를 전송하는 방식
- 공간 분할 다중화(SDM, Space-Division Multiplexing)

주파수 분할 다중화기 (FDM, Frequency Division Multiplexer)
- 통신 회선의 주파수를 여러 개로 분할하여 여러 대의 단말기가 동시에 사용할 수 있도록 하는 장치
- 주파수 분할 다중화기의 특징
  - 주파수 대역폭을 작은 대역폭으로 나누어 사용
  - 하나의 채널에 주파수 대역별로 전송로가 구성
  - 전송하려는 신호의 필요한 대역폭보다 전송매체의 유효대역폭이 클 때 사용
  - 전송에 있어 시간의 지연 없이 실시간 전송
  - 주파수 분할은 변·복조 기능도 포함하기 때문에 별도의 모뎀을 필요로 하지 않음
  - 여러 개의 정보 신호를 한 개의 전송선로에서 동시에 전송 가능
  - 전송매체를 지나는 신호는 아날로그 신호임
  - 전화 회선에서는 1200[baud] 이하의 비동기식에서만 이용
  - 시분할 다중화 장비에 비해 가격이 쌈
  - 멀티 포인트 방식 구성에 적합
  - 구조가 간단하고 주로 저속도의 장비에 이용 가능
  - 케이블 혹은 TV 공중파 텔레비전에 적용
  - 채널간의 누화(Crosstalk) 및 상호변조잡음(Intermodulation noise)을 막기 위해(대역폭이 겹치지 않도록) 완충지역으로 보호대역(가드 밴드, Guard Band)이 필요
  - 가드 밴드의 이용으로 채널의 이용률이 낮아짐으로써 시분할 다중화기에 비해 비효율적

시분할 다중화기 (TDM, Time Division Multiplexer)
- 통신 회선의 대역폭을 일정한 시간 폭(Time Slot)으로 분할한 것
- 시분할 다중화기의 특징
  - 대역폭의 이용도가 높아 고속 전송에 용이
  - 동기식 시분할 다중화기와 비동기식 시분할 다중화기가 있음
- 동기식 시분할 다중화기 (STDM, Synchronous TDM)
  - 전송 매체상의 전송 프레임마다 해당 채널의 타임 슬롯이 고정적으로 할당되는 다중화 방식
  - 매체의 데이터 전송률이 전송 디지털 신호의 데이터 전송을 능가할 때 사용
  - 송·수신 스위치가 서로 정확히 동기 되도록 하기 위해서 이를 위한 동기 비트가 더 필요함
  - 전송할 데이터가 없는 단말장치에도 타임 슬롯을 할당
  - 타임 슬롯을 고정적으로 할당하여 타임 슬롯이 낭비될 수 있음

- 비동기식(통계적) 시분할 다중화기 (ATDM, Asynchronous TDM)
  - 사용자의 요구에 따라 타임 슬롯을 동적으로 할당하여 데이터를 전송하는 다중화 방식
  - 각 채널 할당 시간이 공백인 경우(idle time) 다음 차례에 의한 연속 전송이 가능하여 전송 전달 시간을 빠르게 하는 방식
  - 실제로 전송할 데이터가 있는 단말장치에만 타임 슬롯을 할당함으로써 전송 효율을 높임
  - 다중화 회선의 데이터 전송률을 회선에 접속된 스테이션들의 전송률의 합보다 작게 할 수 있음
  - 같은 속도일 경우 동기 시분할 방식에 비해 많은 스테이션(터미널)을 수요할 수 있음
  - 주소 회로, 흐름 제어, 오류 제어 등의 기능을 함
  - 데이터를 잠시 저장할 버퍼와 주소 제어 회로 등이 별도로 필요
  - 제어 회로가 복잡함
  - 지능 다중화기, 통계적 시분할 다중화기라고도 불림

역 다중화기 (Demultiplexer)
- 두개의 음성 대역폭을 이용하여 광대역에서 얻을 수 있는 통신 속도를 얻을 수 있는 기기
- 역 다중화기의 특징
  - 여러 가지 변화에 대응해 여러 가지 전송속도를 얻을 수 있음
  - 한 채널 고장 시 나머지 한 채널을 1/2의 속도로 계속 운영 가능
  - 비용을 절감할 수 있음
  - 전용회선의 고장 시 DDD망을 이용할 수 있음

집중화기 (Concentrator)
- 여러 개의 채널을 몇 개의 소수 회선으로 공유화시키는 장치

3 데이터 전송 기술

3.1 데이터 전송의 개요

주파수, 진폭, 위상
- 주파수(Frequency): 신호의 주기로 단위는 Hz
- 진폭(Amplitude): 신호의 높낮이
- 위상(Phase): 신호의 시작 위치
- 대역폭 (Bandwidth): 최고 주파수와 최저 주파수 사이 간격

통신 속도
- 변조 속도 (보오, baud)
  - 통신 속도의 단위
  - 1초 동안 몇 개의 신호 변화가 있었는지를 나타냄 (단위시간당 변조율)
  - 1초당 보내지는 코드의 개수
  - baud = 1 / T (변조되는 시간)
  - 변조 속도(baud) = bps / 변조 시 상태 변화 수

  - 변조 시 상태 변화 수
    - 1비트: 모노비트(Monobit)
    - 2비트: 디비트(Dibit)
    - 3비트: 트리비트(Tribit)
    - 4비트: 쿼드비트(Quadbit)
- 데이터 신호 속도 (bps, bit per second)
  - 데이터 통신망에서 사용되는 일반적인 전송 속도 단위로서 1초간에 운반할 수 있는 데이터의 비트 수
  - 데이터 신호 속도(bps) = baud × 변조 시 상태 변화 수
- 베어러 속도
  - 기저대 전송방식에서 데이터 신호 이외에 동기 신호, 상태신호 등을 포함하는 데이터 전송속도
  - 단위는 bps(bit/sec)를 사용

전송 용량
- 샤논(Shannon)의 정의: 백색 가우스 잡음(white gauss noise)이 발생되는 통신로의 용량(C[bit/sec])을 정의

$C=W\log_2(1+S/N)$ [bps] C: 통신용량, W: 대역폭, S: 신호전력, N: 잡음전력

- 통신 회선의 전송 용량을 증가시키기 위한 방법
  - 주파수 대역폭을 증가시킴
  - 신호 세력을 높임
  - 잡음 세력을 줄임

3.2 데이터 전송 매체

유선 매체
- 꼬임선 (이중나선, Twisted Pair Wire)
- 동축 케이블 (Coaxial Cable)
- 광섬유 케이블 (Optical Fiber Cable)

무선 매체
- 라디오파
- 지상 마이크로파

3.3 데이터 전송 방식

아날로그 전송
- 아날로그 신호 형태로 전송되는 방식
- 신호 감쇠 현상이 심하고 에러가 발생할 가능성이 높음

디지털 전송
- 디지털 신호 형태로 전송되는 방식
- 디지털 신호 변환에 의해 아날로그나 디지털 정보의 암호화가 쉽게 구현 가능
- 전송 용량을 다중화 함으로써 효율성이 높음
- 디지털 전송의 각 재생기는 잡음이 없는 새로운 펄스를 재생할 수 있어, 원래의 신호와 동일한 신호의 전달이 가능
- 장거리 전송 시 데이터의 감쇠 및 왜곡 현상을 방지하기 위해서 리피터(Repeater)를 사용
- 적당하게 재생기(리피터)만 설치되면 장거리 전송이 용이
- 패킷전송방식이 주로 이용
- 국과 국간의 전송로는 디지털 방식으로 구성
- LSI, VLSI로 이어지는 기술의 진보로 더욱 발전
- 디지털 기술의 발전으로 전송 장비의 소형화가 가능하며, 가격도 저렴화 되고 있음
- 전송매체는 M/W(Micro Wave), 광케이블, UTP 케이블 등이 있음
- 신호변환기로 DSU 혹은 코덱을 사용함

통신방식에 따른 종류
- 단방향(Simplex) 통신
  - 한쪽 방향으로만 전송이 가능한 방식
  - 현재의 라디오나 공중파 TV방송에 적용되는 통신 방식
- 반이중(Half-Duplex) 통신
  - 2선식 선로를 송신과 수신을 교대로 바꿔가며 전송 하는 방식
  - 데이터를 양쪽방향으로 모두 전송할 수 있으나 동시에 양쪽방향에서 전송할 수 없는 통신 방식
  - 한 통신로를 이용하여 송신과 수신 중 한 가지 기능만으로 사용하되, 송·수신 기능을 번갈아 사용함으로써 상호정보를 교환하는 방법
  - 예: ON-OFF 무전기
- 전이중(Full-Duplex) 통신
  - 송수신 쌍방향으로 동시에 통신이 가능한 전송방식
  - 전송량이 많고 통신 회선의 용량이 클 때 사용

동기식 전송과 비동기식 전송

동기식(Synchronization) 전송	비동기식(Asynchronization) 전송
·전송할 데이터를 여러 블록으로 나누어 블록 단위로 전송 ·휴지시간이 없으므로 전송 효율이 좋음 ·전송속도가 빠름(프레임 단위로 전송) ·주로 원거리 전송에 사용	·한 번에 한 문자씩 전송 ·스타트 비트와 스톱 비트를 사용하여 글자와 글자 사이를 구분 ·문자와 문자 사이의 휴지시간이 일정치 않음 ·2,000bps 이하의 저속, 단거리 전송에 사용

3.4 신호 변환

베이스 밴드 전송 방식

- 원래의 신호(펄스 파형, 디지털 데이터)를 다른 주파수대역으로 변조하지 않고 전송하는 방식
- 정보를 0과 1로 표시하고, 이것을 직류의 전기 신호로 전송
- 단거리 전송에 적합
- 베이스 밴드 전송 방식의 유형
  - 단류 NRZ(Non Return to Zero): 신호 1에 대해 양(+)의 전압을 주고, 0이면 전압을 주지 않음
  - 복류 NRZ: 0은 음(-), 1은 양(+)의 전압을 표현
  - 단류 RZ(Return to Zero): 신호 1에 대해 양(+)의 전압을, 신호 0이면 전압을 주지 않고 신호 간에는 반드시 전압이 0의 상태를 취하는 방식
  - 복류 RZ: 신호 1에 대해 양(+)의 전압을, 신호 0에 대해 음(-)의 전압을 주어 신호 간에는 반드시 전압이 0의 상태를 취하는 방식
  - Bipolar(바이폴라, 양극성) 방식: 신호 0에 대해서는 0V를 유지하고, 1일 때는 양(+), 음(-)을 교대로 표현
  - 맨체스터(Manchester): 신호 0에 대해서는 음(-)에서 양(+)으로, 1일 때는 양(+)에서 음(-)으로 상태가 변화는 방식
  - CM (Code Mark Inversion)

아날로그 데이터를 아날로그 신호로 변환 (아날로그 변조)

- 진폭 변조(AM, Amplitude Modulation): 변조 파형에 따라 진폭을 변조하는 방식
- 주파수 변조(FM, Frequency Modulation): 변조 파형에 따라 주파수를 변조하는 방식
- 위상 변조(PM, Phase Modulation): 변조 파형에 따라 위상을 변조하는 방식

아날로그 데이터를 디지털 신호로 변환

- 펄스 변조
  - 펄스파의 진폭, 위상 등을 변화시키는 변조 방식
  - 분류
    - 연속 레벨(아날로그) 변조 전송 방식: PAM, PWM, PPM
    - 불연속 레벨(디지털) 변조 전송 방식: PNM, PCM
- 펄스 코드 변조 (PCM, Pulse Code Modulation)
  - 화상, 음성, 동영상 비디오, 가상현실 등과 같이 연속적인 시간과 진폭을 가진 아날로그 데이터를 디지털 신호로 변환하는 것
  - 펄스 코드 변조 순서

표본화(Sampling) ⇨ 양자화(Quantizing) ⇨ 부호화(Encoding)

  - 표본화(Sampling): 연속적인 신호 파형을 일정 시간 간격으로 검출하는 단계
    - 샤논의 표본화 이론: 어떤 신호 f(t)를, f(t)가 가지는 최고 주파수의 2배 이상으로 채집하면, 채집된 신호는 원래의 신호가 가지는 모든 정보를 포함한다는 이론
    - 표본화 횟수: 2 × 최고 주파수
    - 표본화 간격: 1 / 표본화 횟수
  - 양자화(Quantizing): 표본화에 의해 얻어진 신호를 평준화 시키는 단계
    - 양자화 레벨: PAM 신호(표본화에 의해 검출된 신호)를 부호화할 때 2진수로 표현할 수 있는 레벨
    - 양자화 레벨 = $2^{\text{표본당전송비트}}$
  - 부호화(Encoding): 펄스 진폭의 크기를 디지털량으로 변환하는 것
  - 복호화(Decoding): PCM 신호를 PAM 신호로 되돌리는 것

디지털 데이터를 아날로그 신호로 변환 (브로드밴드(Broadband) 변조 방식)

- 진폭 편이 변조 (ASK, Amplitude Shift Keying)
  - 2진수 0과 1을 각각 서로 다른 진폭의 신호로 변조하는 방식
- 주파수 편이 변조 (FSK, Frequency Shift Keying)
  - 2진수 0과 1을 각각 서로 다른 주파수로 변조하는 방식
- 위상 편이 변조 (PSK, Phase Shift Keying)
  - 2진수 0과 1을 각각 서로 다른 위상을 가진 신호로 변조하는 방식
  - 반송파로 사용하는 정현파의 위상에 정보를 실어 변조
  - 일정 주파수, 일정 진폭의 정현파 위상에 1Bit(2위상), 2Bit(4위상), 3Bit(8위상)을 대응시켜 각각 다른 위상에 "1" 혹은 "0"을 할당하거나 두 비트 혹은 세 비트를 한꺼번에 할당(위상 변환)하여 전송하므로 속도 향상에 유리
  - 동기식 변·복조기(Synchronous MODEM)에서 주로 사용
- 진폭 위상 편이 변조 (QAM, Quadrature Amplitude Modulation)
  - 일정 진폭 및 위상을 상호 변환하여 신호를 싣는 변조 방식
  - 비트 전송률을 높이기 위해 각각의 벡터를 위상 변화뿐만 아니라 진폭 변화도 시키는 방식
  - 반송파의 위상과 진폭을 상호 변환하여 신호를 전송함으로써 4개의 위상과 2개의 진폭으로 한번에 3비트가 전송 가능한 방식
  - 고속데이터 전송에 이용되며, 주로 9600[bps]의 속도에서 운용되는 변조방식

4 전송 제어 방식

4.1 전송 제어의 개요

전송 제어 (Transmission Control)

- 데이터의 원활한 흐름을 위해 입·출력 제어, 동기 제어, 오류 제어, 회선제어, 흐름 제어 등을 수행하는 것

전송 제어 프로세스

데이터 통신 회선의 접속
⇩
데이터 링크 설정(확립)
⇩
정보 메시지 전송
⇩
링크의 종료(해제)
⇩
데이터 통신 회선의 절단

4.2 데이터 링크 제어 프로토콜

문자 위주 동기 방식 데이터 링크 프로토콜

- BSC (Binary Synchronous Control)
  - 프레임에 전송 제어 문자를 삽입하여 전송을 제어하는 문자 위주의 프로토콜
  - 에러제어와 흐름제어를 위해서는 정지-대기 방식을 사용
  - 점-대-점(Point-to-Point)링크 뿐만 아니라 멀티 포인트 링크에서도 사용될 수 있음
  - 주로 동기전송을 사용하나 비동기 전송방식을 사용하기도 함
  - 반이중(Half Duplex)전송만 지원
  - 프레임 구조

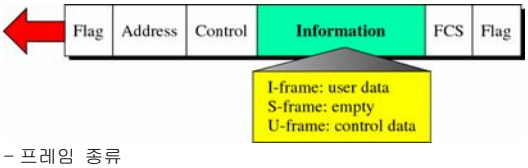
SYN (동기문자)	SYN (동기문자)	SOH (헤딩의 시작)	Heading (프레임순서 및 수신국 주소)	STX (헤딩의종료 및 본문시작)	TEXT (본문)	ETX (본문의종료)	BCC (오류검출)

- 제어 문자의 종류
  - 전송 제어 문자
  - 장치 제어 문자
  - 포맷 제어 문자
  - 정보 분리 문자

- 전송 제어 문자
  - SYN(SYNchronous idle): 동기 맞춤 문자
  - SOH(Start Of Heading): 헤딩의 개시를 표시
  - STX(Start of Text): 실제 전송할 데이터 집합의 시작임을 의미
  - ETX(End Of TeXt): 본문의 종료
  - ETB(End of Transmission Block): 블록의 종료
  - EOT(End Of Transmission): 한 개 또는 그 이상의 전송 종료를 표시
  - DLE (Data Link Escape)
    - 인접하여 뒤따르는 제한된 수의 문자나 의미를 바꾸는 통신제어문자로서 데이터통신 네트워크에 보조적인 제어의 목적으로만 사용
    - 데이터 투과성(Data Transparent)을 위해 삽입되는 제어문자
  - ACK(ACKnowledge): 수신 측에서 송신 측으로 긍정 응답을 보내는 문자
  - NAK(Negative AcKnowledge): 수신 측에서 송신 측으로 부정 응답으로 보내는 문자
  - ENQ(EnQuiry): 링크 설정 요청, 상대국의 응답 요청

비트 위주 동기 방식 데이터 링크 프로토콜

- HDLC (High-level Data Link Control)
  - 각 프레임에 데이터 흐름을 제어하고 오류를 검출할 수 있는 비트열을 삽입하여 전송하는 비트 위주의 프로토콜
  - 전송 효율과 신뢰성이 높음
  - 정보 전송 단위가 프레임
  - 전송 제어상의 제어를 받지 않고 문자 코드 종류와 무관하게 투명하게 동작 (비트 투과성)
  - 단방향, 반이중, 전이중 모두 사용 가능
  - Go-Back-N ARQ 에러 제어 방식을 사용
  - 데이터 링크 형식은 Point-to-Point, Multi-point, Loop 모두 가능
  - 프레임 구조



  - 프레임 종류
    - 정보 프레임 (I-frame: Information Frame)
    - 감독 프레임 (S-frame: Supervisory Frame)
    - 비번호(무번호) 프레임 (U-frame: Unnumbered Frame)
  - HDLC의 수행 국 (Station): 주국, 종국, 혼합국

- 데이터 전송 모드
  - · 표준(정규) 응답 모드(NRM, Normal Response Mode)
  - · 비동기 응답 모드(ARM, Asynchronous Response Mode)
  - · 비동기 평형(균형) 모드(ABM, Asynchronous Balanced Mode)
- SDLC (Synchronous Data Link Control)
  - HDLC와 동일한 프레임 구조를 가짐
  - IBM에서 개발한 비트 방식의 프로토콜로 HDLC의 기초가 됨

4.3 에러

에러의 발생원인

- 감쇠 (Attenuation)
  - 전송 신호가 전송 매체를 통과하는 과정에서 거리에 따라 점차 약해지는 현상
- 지연 왜곡 (Delay Distortion)
  - 주로 하드와이어 전송 매체에서 발생되며, 전송 매체를 통한 신호 전달이 주파수에 따라 그 속도를 달리 함으로써 유발되는 신호 손상
- 상호 변조 잡음 (Intermodulation Noise)
  - 서로 다른 주파수들이 똑같은 전송 매체를 공유할 때 이 주파수들이 서로의 합과 차의 신호를 발생함으로써 발생되는 잡음
- 충격 잡음 (Impulse Noise)
  - 비연속적이고 불규칙한 진폭을 가지며, 순간적으로 높은 진폭이 발생하는 잡음
  - 외부의 전자기적 충격이나 기계적인 통신 시스템에서의 결함 등이 원인
  - 디지털 데이터를 전송하는 경우 중요한 오류발생의 원인이 됨

에러 검출 방식

- 패리티(Parity) 검사
  - 데이터 한 블록 끝에 1비트의 검사 비트인 패리티 비트(Parity Bit)를 추가하여 전송에러를 검출하는 방식
  - 짝수(우수) 패리티
    전송 비트 내에 1의 개수가 짝수가 되도록 하는 것
  - 홀수(기수) 패리티
    전송 비트 내에 1의 개수가 홀수가 되도록 하는 것
  - 수직 패리티 체크 방식 (VRC, Vertical Redundancy Check)
    전송 비트들 중 수직에 대한 1의 bit수를 짝수 혹은 홀수가 되도록 하는 방식
  - 수평 패리티 체크 방식 (LRC, Longitudinal Redundancy Check)
    전송 비트를 일정량의 블록으로 묶어 블록의 맨 마지막에 패리티를 부여하는 방식

- 순환 중복 검사 (CRC, Cyclic Redundancy Check)
  - 특정 다항식에 의한 연산 결과를 데이터에 삽입하여 전송하는 에러검출 방법
  - 동기 전송에서 주로 사용되는 에러 검출 방식
  - HDLC 프레임의 FCS에 사용되는 방식
- 해밍 코드(Hamming Code) 방식
  - 자기 정정 부호의 하나로 비트 착오를 검출해서 1 bit 착오를 정정하는 부호 방식
  - 송신한 데이터와 수신한 데이터의 각 대응하는 비트가 서로 다른 비트의 수를 해밍 거리(Hamming Distance)라고 함
- 상승 부호(코드) 방식
- 궤환 전송 방식
- 연속 전송 방식 (자동 연속 방식)

에러 제어 방식

- 자동 반복 요청 (ARQ, Automatic Repeat reQuest)
  - 통신 경로에서 에러 발생 시 수신측은 에러의 발생을 송신 측에 통보하고 송신측은 에러가 발생한 프레임을 재전송
  - 정지-대기(Stop-and-Wait) ARQ
    - · 송신 측이 하나의 블록을 전송한 후 수신 측에서 에러의 발생을 점검한 다음 에러 발생 유무 신호를 보내올 때까지 기다리는 방식
    - · 수신 측에서 에러 점검 후 제어 신호를 보내올 때까지 오버헤드(overhead)가 효율 면에서 가장 부담이 큼
  - 연속(Continuous) ARQ
    - · Go-Back-N ARQ: 에러가 발생한 블록 이후의 모든 블록을 다시 재전송하는 방식
    - · 선택적 재전송(Selective-Repeat) ARQ: 수신 측에서 NAK를 보내오면 에러가 발생한 블록만 재전송
  - 적응적(Adaptive) ARQ: 데이터 블록의 길이를 채널의 상태에 따라 동적으로 변경시키는 방식

전송 에러 제어 방식

- 전진 에러 수정 (FEC, Forward Error Correction)
  - 송신 측에서 정보비트에 오류 정정을 위한 제어 비트를 추가하여 전송하면 수신 측에서 이 비트를 사용하여 에러를 검출하고 수정하는 방식
  - ARQ 방식과는 달리 재전송 요구가 없으므로 역 채널이 필요 없고 연속적인 데이터 흐름이 가능
  - ARQ에 비해 기기와 코딩이 더 복잡함
  - ARQ 방식과 마찬가지로 데이터와 함께 잉여 비트들을 함께 전송함
  - 잉여 비트들이 데이터 시스템 효율의 개선을 저해함
  - 대표적인 예로 해밍(Hamming) 코드 방식과 상승 코드 방식이 있음
- 후진 에러 수정 (BEC, Backward Error Correction)
  - 데이터 전송 과정 중 에러가 발생하면 송신 측에 재전송을 요구하는 방식

4.4 전송 트래픽 제어

흐름 제어 (Flow Control)

- 통신망 내의 트래픽 제어의 원활한 흐름을 위해 망 내의 노드와 노드 사이에 전송하는 프레임(패킷)의 양이나 속도를 규제하는 제어
- 정지-대기 (Stop-and-Wait)
  - 수신 측으로부터 ACK를 받은 후에 다음 프레임을 전송하는 방식
- 슬라이딩 윈도우 (Sliding Window)
  - 수신 측의 확인 신호를 받지 않더라도 미리 정해진 프레임 수만큼 연속적으로 전송하는 방식
  - 한 번에 여러 개의 프레임을 나누어 전송할 경우 효율적인 기법
  - 수신 측으로부터 이전에 송신한 프레임에 대한 긍정 수신 응답(ACK)이 왔을 때 송신 윈도우 증가
  - 윈도우(Window): 전송할 수 있는 프레임의 개수

혼잡(폭주) 제어 (Congestion Control)

- 네트워크 내에서 패킷의 대기 지연(Queuing delay)이 너무 높아지게 되어 트래픽이 붕괴되지 않도록 네트워크 측면에서 패킷의 흐름을 제어하는 트래픽 제어

교착 상태 회피 (Deadlock Avoidance)

- 교착 상태: 교환기 내에 패킷들을 기억할 수 있는 공간이 포화 상태에 있을 때 다음 패킷들이 기억 공간에 들어가기 위해 무한정 기다리는 현상
- 교착 상태 발생 시 교착 상태에 있는 한 단말 장치를 선택하여 패킷 버퍼를 폐기함

5 데이터 회선망

5.1 교환 회선과 전용 회선

교환 회선 (Switched Line)

- 교환기에 의해서 연결되는 방식
- 전용 회선에 비해 전송 속도가 느림
- 전송할 데이터양이 적고, 회선 사용 시간이 짧을 때 효율적

전용 회선 (Leased Line)

- 회선이 단말기 상호 간에 항상 고정되어 있는 회선 방식
- 전송 속도가 빠르고 전송 에러가 적음
- 전송할 데이터양이 많고, 회선 사용 시간이 길 때 효율적
- 직통 회선(Point-to-Point)방식과 분기 회선(Multi-Point 혹은 Multi-Drop)방식이 있음

5.2 회선 구성 및 제어 방식

회선 구성 방식

방 식	설 명
포인트 투 포인트 (Point-to-Point)	· 중앙 컴퓨터와 각 단말기가 독립적인 회선으로 일대일로 연결되어 있는 방식
멀티 포인트 (Multi-Point, 멀티 드롭)	· 한 개의 통신 회선에 여러 개의 단말기들을 연결하는 방식
회선 다중 (Line Multiplexing, 다중화 방식)	· 여러 대의 단말기들을 다중화 장치를 이용하여 중앙 컴퓨터와 연결하는 방식

회선 제어 방식

- 경쟁(Contention) 방식
  - 회선 제어 방식 중 가장 간단한 형태
  - 회선의 접근을 위해 서로 경쟁하는 방식으로 송신 요구를 먼저 한 쪽이 송신권을 갖는 방식
  - 송신측이 전송할 메시지가 있을 경우 사용 가능한 회선이 있을 때까지 기다려야 함
  - Point-to-Point 회선으로 접속되어 있어 관계가 대등한 단말에 많이 이용
  - 주 통신국과 종속 통신국이 따로 없고 데이터 링크를 설정하고자 하는 단말장치가 주국이 되어 시행
  - 대표적인 시스템으로 ALOHA가 있음
- 폴링과 셀렉션(Polling & Selection) 방식

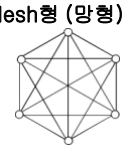
방 식	설 명
폴링(Polling)	· 데이터 통신에서 컴퓨터가 단말기에게 전송할 데이터의 유무를 묻는 방식
셀렉션(Selection)	· 멀티 포인트 방식에 있어서 중앙 컴퓨터가 주변의 터미널로 데이터를 전송하고자 하는 경우, 수신측 터미널의 상태를 확인하는 절차

5.3 네트워크의 구조와 구성 형태

네트워크의 논리적 구성 요소

- 사용자 프로세스(User Process): 정보처리나 통신을 하는 장치를 모델화
- 노드(Node): 원격처리장치 혹은 교환기를 모델화
- 링크(Link): 일반적으로 통신회선, 즉 전기신호를 운반하는 매체를 모델화

Mesh형 (망형)

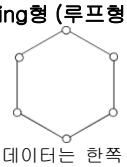


- 보통 공중 데이터통신 네트워크에서 주로 사용되며, 통신 회선의 총 경로가 다른 네트워크 형태와 비교하여 가장 길게 소요되는 네트워크 구성 형태
- node의 연결성이 높음
- 많은 단말기로부터 많은 양의 통신을 필요로 하는 경우에 유리
- 모든 노드를 망형으로 연결할 경우: 노드수가 n개 일 때, n(n-1) / 2개의 회선이 필요

Star형 (성형, 중앙 집중형)

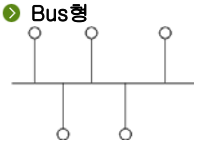
- 중앙에 Host Computer가 있고 이를 중심으로 터미널들이 연결되는 중앙집중식의 망 구성 형태
- 모든 스테이션이 중앙 스위치에 연결된 형태로 두 스테이션은 회선교환에 의해 통신을 행함
- 주프로세서(Host processor)를 통하여 데이터를 교환하며 통신망제어를 가장 간편하게 할 수 있음
- 교환 node 수가 가장 적음

Ring형 (루프형)



- 데이터는 한쪽 방향으로만 흐르고 병목 현상이 드묾
- 양쪽 방향으로 접근이 가능하여 통신 회선 장애에 대한 융통성이 있음
- 한 노드(node)가 절단되어도 우회로를 구성하여 통신이 가능
- 트래픽이 일정한 시스템에 적합
- 노드의 추가와 변경이 비교적 어려움
- 단방향 링의 경우 두 노드 사이의 채널이 고장 나면 전체 네트워크가 손상될 수 있는 단점을 가짐
- 중계기 수가 많아짐
- 근거리 네트워크(LAN)에서 가장 많이 채택되고 있는 방식

Bus형



- 한 개의 통신 회선에 여러 대의 단말 장치가 연결
- 단말 장치가 고장 나더라도 통신망 전체에 영향을 주지 않으므로 신뢰성이 높음

Tree형 (계층형, 분산형)

- 분산 처리 시스템을 구성하는 방식

5.4 데이터 교환 기술

회선 교환 방식 (Circuit Switching)

- 음성 전화망과 같이 메시지가 전송되기 전에 발생지에서 목적지까지의 물리적 통신 회선 연결이 선행되어야 하고 이 물리적인 연결이 정보 전송이 종료될 때 까지 계속 유지 되는 교환 방식
- 고정 대역폭을 사용하고 각 전문은 동일한 물리적 경로를 따름
- 시분할 교환 방식 (TDS, Time Division Switching)
  - TDM 버스 교환, 타임 슬롯 상호 교환, 시간 다중화 교환이 있음
- 공간 분할 교환 방식 (SDS, Space Division Switching)
  - 일반 전화 회선 교환에 사용 되는 방식으로 데이터 전송에 필요한 시간이 가장 김
- 회선 교환 방식에서 제어 신호의 종류
  - 감시 제어 신호 (Supervisory Control Signal)
  - 주소 제어 신호 (Address Control Signal)
  - 통신망 관리 제어 신호 (Communication Management Control Signal)
  - 호 정보 제어 신호 (Call Information Control Signal)

메시지 교환 방식 (Message Switching)

- 하나의 메시지 단위로 축적-전달(store-and-forward) 방식에 의해 데이터를 교환하는 방식
- 수신측이 준비 안 된 경우에도 지연 후 전송이 가능
- 각 메시지마다 전송 경로가 다르고 수신 주소를 붙여서 전송
- 네트워크에서 속도나 코드 변환이 가능
- 전송 지연 시간이 가장 김
- 응답시간이 느려 대화형 데이터 전송을 위해서는 부적절

패킷 교환 방식 (Packet Switching)

- 메시지를 일정한 길이의 패킷으로 잘라서 전송하는 방식
- 패킷 (Packet): 전송 혹은 다중화의 목적으로 메시지를 정해진 크기의 비트 수로 나눈 다음 정해진 형식에 맞추어 만들어진 데이터의 블록
- 패킷을 일시 저장했다 수신처에 따라 적당한 경로를 선택해서 전송 (Store-and-Forward)하는 방식
- 가상 회선 방식과 데이터 그램 방식이 있음
- 가상 회선 방식
  - 송수신국 사이에 논리적 연결이 설정됨
  - 정보 전송 전에 제어 패킷에 의해 경로가 설정됨
  - 패킷의 발생 순서대로 전송
  - 통신 과정: 호(Call) 설정 ⇨ 데이터(패킷) 전송 ⇨ 호(Call) 해제
  - 별도의 호(Call) 설정 과정이 있다는 것이 회선 교환 방식과의 공통점임
- 데이터 그램 방식
  - 데이터의 전송 시에 일정 크기의 데이터 단위로 쪼개어 특정 경로의 설정 없이 전송되는 방식
  - 수신 측에서 도착한 패킷들의 순서를 재정리해야 함

패킷 교환망 (PSDN, Packet Switched Data Network)

- 정보를 패킷 단위로 전송
- 부호가 다른 단말장치 사이의 통신이 가능
- 처리속도가 다른 단말장치 사이의 통신이 가능
- 축적 전송기능에 의해 패킷 다중전송이 가능
- 회선 이용효율의 극대화
- 전송 품질이 우수하며 고신뢰성
- 전송량 제어와 전송속도 변환 가능
- 장애발생시 대체 경로 선택이 가능
- 표준화된 프로토콜 적용
- 대량의 데이터 전송 시 전송 지연
- 패킷 교환망의 기능
  - 순서 제어
  - 경로 선택 제어(Routing Control)
  - 트래픽 제어(Traffic Control)
  - 에러 제어(Error Control)
  - 패킷 다중화
  - 논리 채널

경로 설정 (Routing)

- 데이터 패킷을 출발지에서 목적지까지 이용 가능한 전송로를 찾아본 후에 가장 효율적인 전송로를 선택하는 것
- 경로 배정 요소 (Parameter)
  - 성능 기준
  - 경로의 결정 시간과 장소
  - 네트워크 정보 발생지
- 경로 설정 프로토콜
  - IGP (내부 게이트웨이 프로토콜, Interior Gateway Protocol)
  - EGP (외부 게이트웨이 프로토콜, Exterior Gateway Protocol)
  - BGP (Border Gateway Protocol)

④ 경로 설정 알고리즘
  - 범람 경로 제어 (Flooding)
  - 고정 경로 제어 (착국 부호 방식, Static Routing)
  - 적응 경로 제어 (Adaptive Routing)
  - 임의 경로 제어 (Random Routing)

6 통신 프로토콜

6.1 프로토콜

프로토콜 (Protocol)

- 둘 이상의 컴퓨터 사이에 데이터 전송을 할 수 있도록 미리 정보의 송·수신측에서 정해둔 통신 규칙
- 정보통신을 위해 통신을 원활하게 수행할 수 있도록 해주는 통신 규약
- 통신을 제어하기 위한 표준적인 규칙과 절차의 집합
- 서로 다른 시스템 간에 존재하는 두 실체(Entity)간에 정확하고 효율적인 정보전송을 위한 일련의 절차나 규범의 집합

프로토콜의 기본요소

- 구문(Syntax): 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정
- 의미(Semantic): 전송제어와 오류관리를 위한 제어정보를 포함
- 타이밍(Timing): 기기 간의 통신 속도, 메시지 순서 제어 등을 규정

프로토콜의 기능

- 동기 제어 (Synchronization Control)
- 분리와 재결합
- 흐름 제어 (Flow Control)
- 순서 제어 (Sequencing)
- 주소 지정 (Addressing)

- 요약화 (캡슐화, Encapsulation)
  - 분리된 데이터에 프로토콜 제어 정보, 에러 검출 코드, 송·수신지 주소 등의 제어 정보를 부가하는 것
  - 송신기에서 발생된 정보의 정확한 전송을 위해 사용자 정보에 헤더(header)와 트레일러(trailer)를 부가하는 과정
- 에러 제어 (Error Control)
- 경로 제어 (Routing)
- 다중화 (Multiplexing)

프로토콜의 전송 방식

- 문자 방식
- 바이트 방식
- 비트 방식

6.2 OSI 참조 모델 7계층

OSI(Open System Interconnection) 참조 모델 7계층

- 서로 다른 시스템 간의 원활한 통신을 위해 ISO(국제표준화기구)에서 제안한 통신 규약으로, 7단계로 표준화 하여 규정

OSI 7계층 구조

상위레벨	7. 응용 계층
	6. 프레젠테이션(표현) 계층
	5. 세션 계층
	4. 트랜스포트(전송) 계층
하위레벨	3. 네트워크 계층
	2. 데이터 링크 계층
	1. 물리 계층

- Layer1: 물리 계층 (Physical Layer)
  - 전기적, 기능적, 절차적 기능 정의
  - 장치와 전송매체 간의 인터페이스 특성 규정, 전송 매체의 유형 규정, 전송로의 연결·유지 및 해제를 담당
  - 관련 표준: RS-232C, X.21 등
- Layer2: 데이터 링크 계층 (Data Link Layer)
  - 2개 인접된 호스트(Host) 간에 데이터의 전송을 행하고 전송에러를 제어
  - 신뢰성 있고 효율적인 프레임 데이터 전송
  - 논리 링크 제어 및 매체 액세스 제어
  - 프레임 동기(순서 제어), 흐름 제어, 전송 제어를 통해 링크의 효율성을 향상
  - 오류제어 (에러 검출 및 정정)
  - 관련 표준: HDLC, LAP-B, LLC, LAPD, ADCCP 등

- Layer3: 네트워크 계층 (Network Layer)
  - 네트워크 연결을 설정, 유지, 해제하는 기능
  - 통신 시스템간의 경로 설정 및 네트워크 연결 관리
  - 패킷 정보를 전송
  - 정보 교환과 중계 기능
  - 통신 트래픽의 흐름을 제어
  - 통신 중에 패킷의 분실로 재전송을 요청할 수 있는 오류제어 기능
  - 관련 표준: X.25, IP 등
- Layer4: 트랜스포트(전송) 계층 (Transport Layer)
  - 네트워크 종단(end)시스템간의 데이터를 일관성 있게 전송
  - 종점 간(end-to-end) 전송 연결 설정, 데이터 전송, 전송 연결 해제
  - 오류 수정과 흐름 제어를 수행
  - 신뢰성 있고 투명한 데이터 전송을 제공
  - 전송 데이터의 다중화 및 중복 데이터 검출, 누락 데이터 재전송
  - 네트워크를 A, B, C형의 3개의 타입으로 나누고, 서비스 등급인 Class를 0~4까지 5개로 나누어 응용프로세스에게 일정한 전송 품질(QoS)을 제공 (예를 들어 Class 0의 경우 기본 커널 기능만 수행)
  - 관련 표준: TCP, UDP 등
- Layer5: 세션 계층 (Session Layer)
  - 프로세스 간에 연결을 확립, 관리, 단절시키는 수단을 제공
  - 대화를 구성하고, 동기를 취함 (통신 시스템간의 회화 기능 제공)
  - 데이터교환을 관리하기 위한 수단을 제공
  - 전송하는 정보의 일정한 부분에 체크 점(check point)을 둠
  - 소동기점과 대동기점을 이용하여 회화 동기를 조절
- Layer6: 프레젠테이션 계층 (Presentation Layer)
  - 접속 설정 기능
  - 문맥 관리 기능
  - 정보 전송 기능
  - 데이터 암호화 및 압축 수행
  - 데이터 표현 형식의 설정 및 제어
  - 코드 변환
- Layer7: 응용 계층 (Application Layer)
  - 사용자에게 응용서비스를 제공

6.3 X.25 패킷 교환 네트워크

X.25

- 패킷망으로 정보를 전송할 때 패킷 터미널을 제안한 표준 규격안
- 공중 데이터망에서의 패킷형태를 위한 DTE와 DCE의 인터페이스 규격을 포함하고 있는 ITU-T 권고안

X.25의 특징

- 사용자 장치(DTE)와 패킷 네트워크 노드(DCE) 간의 데이터 교환 절차를 정의
- 1976년에 처음 승인한 국제 표준 프로토콜로, 호환성이 뛰어남
- X.25의 가장 중요한 사항은 패킷들이 하나의 경로를 공유할 수 있도록 하는 다중화 기능임
- 에러 검출 기능이 뛰어나 신뢰성이 높음

X.25의 계층 구조

		응용 계층
		표현 계층
		세션 계층
		전송 계층
패킷 계층		네트워크 계층
프레임 계층		데이터링크 계층
물리계층		물리계층
<X.25>		<OSI 7 계층>

- 물리 계층 (물리 레벨 프로토콜)
  - 단말장치와 패킷 교환망 간의 물리적 접속에 관한 인터페이스 정의
  - X.21을 사용
- 프레임(링크) 계층 (프레임 레벨 프로토콜)
  - 패킷의 원활한 전송을 위해 데이터 링크의 제어를 수행
  - 전송 제어를 위해 HDLC의 변형인 LAP-B(Link Access Procedure-Balanced) 사용
- 패킷 계층 (패킷 레벨 프로토콜)
  - OSI 7계층의 네트워크 계층에 해당
  - 패킷 계층의 수행 단계: 호 설정(Call Setup) ⇨ 데이터 전송(Data Transfer) ⇨ 호 제거(Call Cleaning)

6.4 TCP / IP

TCP / IP (Transmission Control Protocol/Internet Protocol)

- 인터넷에서 사용하고 있는 프로토콜로서 서로 다른 기종의 컴퓨터들 간에 데이터 송·수신이 가능하도록 해주는 표준 프로토콜

TCP / IP 계층 구조

응용계층		응용 계층 표현 계층 세션 계층
TCP/UDP(전송계층)		전송 계층
IP(인터넷계층)		네트워크 계층
링크계층		데이터 링크 계층 물리 계층
<TCP/IP>		<OSI 7 계층>

- 응용 계층(Layer4): 응용 프로그램 간의 데이터 송·수신 제공
  - FTP(File Transfer Protocol)
  - SMTP(Simple Mail Transfer Protocol)
  - SNMP(Simple Network Management Protocol)
  - TELNET(TELecommunication NETwork)
- 전송 계층(Layer3): 호스트들 간의 신뢰성 있는 통신 제공
  - TCP (Transmission Control Protocol)
  - UDP (User Datagram Protocol)
- 인터넷(네트워크) 계층(Layer2): 주소 지정, 경로 설정
  - IP(Internet Protocol): 여러 개의 패킷 교환망들의 상호 연결을 위한 범용 비연결성 프로토콜
    - 호스트의 주소 지정
    - 패킷 절단
    - 전송 경로의 논리적 관리
  - ICMP(Internet Control Message Protocol): 인터넷 제어 메시지 프로토콜
  - IGMP(Internet Group Management Protocol): 인터넷 그룹 관리 프로토콜
  - ARP(Address Resolution Protocol): 주소 분석 프로토콜
  - RARP(Reverse Address Resolution Protocol): 호스트의 물리적 주소로부터 IP 주소를 구할 수 있도록 하는 프로토콜
- 링크 계층(Layer1): 실제 데이터(프레임)를 송·수신하는 역할

6.5 정보 통신 관련 표준안 제정 기구

국제표준화기구 (ISO, International Organization for Standardization)

국제전기통신연합 전기통신표준화 부문 (ITU-T, International Telecommunication Union -Telecommunication Standardization Sector)

- 주요 ITU-T 권고안
  - I 시리즈: ISDN에 관한 권고
  - X 시리즈: 공중 데이터망(PSDN)을 통한 데이터 전송에 관한 권고
  - V 시리즈: 공중 전화망(PSTN)을 통한 데이터 전송에 관한 권고
  - T 시리즈: 텔레매틱 서비스를 위한 단말 장치와 프로토콜에 관한 권고
  - X.400 계열: 메시지 통신 처리 시스템(MHS)에 대한 권고안

- **국제전기표준협회 (IEC, International Electrotechnical Commission)**
  - 전기 전자 분야에서 국제 규격의 조정과 통일을 목적으로 설립
- **전기전자기술자협회 (IEEE, Institute of Electric and Electronic Engineers)**
  - ANSI에 의해 미국국가표준을 개발하도록 인증 받은 전문 기구
- **IETF (Internet Engineering Task Force)**
  - 변화하는 망 환경에 따라 새로운 기술을 제시하고 인터넷 표준안을 제정하기 위한 기술 위원회

7 정보 통신망과 인터넷

7.1 VAN

VAN (부가가치통신망, Value Added Network)

- 정보 제공시 통신회선을 공중 통신사업자로부터 임차하여 하나의 사설망을 구축하고 이를 통해 축적해 놓은 갖가지 정보를 유통시키는 정보통신 서비스망
- 단순한 정보의 수집 및 전달 기능뿐만 아니라 정보의 저장, 가공, 관리 및 검색 등과 같이 정보에 부가가치를 부여하는 통신망
- 공중 통신 회선에 교환설비, 컴퓨터 및 단말기 등을 접속시켜 새로운 부가 기능을 제공하는 통신망

VAN의 계층 구조

정보처리 계층
통신처리 계층
네트워크 계층
전송 계층

VAN의 기능

- 전송 기능 (전송 계층)
- 교환 기능 (네트워크 계층)
- 통신 처리 기능 (통신 처리 계층)
  - 축적 교환 기능
    - 전자 사서함 (Mail Box)
    - 데이터 교환
    - 동보 통신: 한 단말기에서 여러 단말기로 같은 내용을 동시에 전송하는 기능
    - 정시 수집
    - 정시 배달
  - 변환 기능
    - 프로토콜 변환: 회선제어, 접속 등의 통신 절차를 변환하는 기능
    - 속도 변환
    - 코드 변환
    - 데이터 형식 변환
    - 미디어 변환
- 정보 처리 기능 (정보 처리 계층)
  - 데이터베이스 구축, 정보 검색 서비스, 소프트웨어 개발 등

7.2 LAN

LAN (근거리통신망, Local Area Network)

- 정보통신 기술발전에 의해 출현한 정보화의 한 형태로서, 한 건물 또는 공장, 학교 구내, 연구소 등의 일정지역 내의 설치된 통신망으로서 각종 기기 사이의 통신을 실행하는 통신망
- 구내나 동일 건물 내(제한된 지역)에서 프로그램, 파일 또는 주변장치들을 공유할 수 있는 컴퓨터 통신망

LAN의 표준안

- OSI 7계층 구조상 물리 계층과 데이터링크 계층을 대상으로 함
- IEEE 802 주요 표준 규격
  - 802.5: 토큰 링 방식의 매체 접근 제어(MAC) 계층에 관한 규약
  - 802.11: 무선 LAN에 관한 규약

LAN의 특징

- 제한된 지역 내의 통신
- 근거리 상호통신을 지원하고 워크스테이션 간을 연결하는데 사용
- 단일 건물 내에 설치되고, 패킷 지연이 최소화됨
- 경로 선택이 필요하지 않고, 망에 포함된 자원을 공유함
- 네트워크 내의 모든 정보기기와 통신이 가능
- 광대역 전송 매체의 사용으로 고속 통신이 가능
- 확장성과 재배치성이 좋음
- 매우 낮은 오류율을 가지며, 방송 형태의 이용이 가능
- 소단위 고속정보통신망
- 공중망을 이용하는 광역 통신망에 대조되는 망
- 기본적인 회선망의 형태로 성형, 버스형, 링형, 계층형(트리형)이 있음
- 전송 매체로 꼬임선, 동축 케이블, 광섬유 케이블 등이 사용됨
- 전송 방식으로 베이스밴드와 브로드밴드 방식이 있음

LAN 시스템 장비

- CIU (Communication Interface Unit)
- BIU (Bus Interface Unit)
- MAU (Media Access Unit)

LAN의 확장 및 변형

- CO-LAN
- WAN (Wide Area Network)
- MAN (Metropolitan Area Network)
- PBX (사설 교환기, Private Branch Exchange)

매체 접근 제어(MAC; Media Access Control) 방식에 의한 분류

- CSMA / CD
  - 데이터의 충돌을 막기 위해 송신 데이터가 없을 때에만 데이터로 송신하고, 다른 장비가 송신중일 때에는 송신을 중단하며 일정시간 간격을 두고 대기하였다가 순서에 따라 다시 송신하는 방식

  - 전송 도중 충돌이 감지되면 즉시 전송을 멈추고 다른 스테이션에 충돌을 알리는 잼(Jam) 신호를 전송
  - 잼(Jam) 신호를 전송 후 일정 시간이 흐른 뒤 데이터를 재송신
  - 통신량이 적을 때 채널 이용률이 높음
  - 장애 처리가 쉬움
  - LAN에 연결되어 있는 어느 한 DTE가 고장이 나더라도 다른 DTE의 통신에는 전혀 영향을 미치지 않음
  - 일정길이 이하의 데이터를 송신할 경우 충돌을 검출할 수 없음
  - 일반적으로 지연시간을 예측할 수 없음
  - 버스형 또는 성형 근거리 통신망에 가장 일반적으로 이용
  - IEEE 802.3의 표준규약
- 토큰 버스 (Token Bus)
  - 버스(Bus) 형태를 갖는 LAN에서 사용하는 방식으로, 망을 구성하는 모든 노드들 사이에 논리적 링(Ring)이 형성되어 토큰을 잡은 노드만이 데이터 패킷을 송신할 권리를 가지며, 데이터 패킷을 전송한 후 토큰을 논리적 링 상의 다음 노드로 넘겨주는 방식
- 토큰 링 (Token Ring)
  - 링(Ring) 형태를 갖는 LAN에서 사용하는 방식으로, 물리적으로 연결된 링 형태의 망을 따라 한쪽 방향으로 순회하는 토큰에 의해 노드에게 데이터의 송신권이 주어지는 방식

이더넷 (Ethernet)

- CSMA / CD방식을 사용하는 LAN
- 고속 이더넷과 기가비트 이더넷이 있으며, 기존의 LAN과 같은 구성과 MAC 프로토콜을 그대로 사용할 수 있음
- 고속 이더넷(Fast Ethernet): 100Mbps의 전송 속도를 지원
- 기가비트 이더넷(Gigabit Ethernet): 1Gbps의 전송 속도를 지원

이더넷 시스템 규격

- 10 BASE T
  - -10: 10Mbps
  - -BASE: 베이스밴드 방식
  - -T: 전송매체로 꼬임선(Twisted Pair Wire)을 사용
- 10 BASE 2
  - -얇은 동축 케이블 이용
  - -2: 한 세그먼트의 케이블 길이가 최대 200m라는 의미
- 10 BASE 5
  - -굵은 동축 케이블을 이용
  - -5: 한 세그먼트의 케이블 길이가 최대 500m라는 의미

7.3 ISDN

ISDN (종합 정보 통신망, Integrated Service Digital Network)

- 컴퓨팅, 교환, 디지털 전송 장치간의 구분이 없어지고, 음성, 데이터 및 이미지 전송에 동일한 디지털 기술이 적용된 통합 시스템
- 모든 통신 서비스를 단일 통신망으로 통합한 것

ISDN의 특징

- 하나의 통신망에 접속되며, 디지털 전송기술을 이용하여 데이터, 음성, 화상정보 등 다양한 서비스를 제공
- 통신망의 경제성과 효율성을 증대시키고 통신처리 기능을 고도화시킴
- 사용자는 단일 / 복수의 다른 사용자와 동시에 교대로 통신서비스를 제공받을 수 있음
- 음성 신호와 컴퓨터 단말기에 사용되는 신호, 그리고 텔레비전의 영상 신호 등을 하나의 통신망으로 연결 가능
- 데이터베이스나 정보 처리 기능의 이용 범위가 넓어지게 되어 통신의 이용 가치를 높임
- 통신망 운용자도 많은 부가가치를 얻을 수 있음
- 64kbps 디지털 기본 접속 기능을 제공
- OSI 참조 모델에 정의된 계층화 된 프로토콜 구조가 적용됨
- 통신망의 교환 접속 기능에는 회선 교환방식과 패킷 교환 방식이 있음
- 하위 계층 기능만 제공하는 베어러 서비스와 상·하위 계층 기능을 모두 제공하는 텔레 서비스로 나뉨
- 채널은 B, D, H 등이 있음

ISDN의 통신 서비스

- 베어러 서비스 (Bearer Service)
  - 회선 교환, 패킷 교환 등 하위 계층 기능만을 제공하는 서비스
- 텔레 서비스 (Tele service)
  - 통신망과 단말 기능을 제공하는 서비스로 OSI 상위 4개 계층까지도 지원
  - 실제로 단말을 조작하고 통신하는 이용자 측에서 본 서비스

ISDN의 구조

- TDM을 이용해서 사용자 정보채널과 신호 정보채널을 구성
- 채널 종류
  - A 채널
  - B(Bearer) 채널
  - D(Data) 채널
  - H(Hybrid) 채널
- 사용자망 인터페이스
  - 기본 속도 인터페이스 (BRI, Basic Rate Interface)
  - 1차군 속도 인터페이스 (PRI, Primary Rate Interface)

- 기능 그룹
  - 장비 및 소프트웨어에 의해 구현되는 기능들의 집합으로 각 장비들의 경계점을 명확히 규정하기 위해 사용
  - 망 종단 장치 (NT, Network Termination)
  - 터미널 장비 (TE, Terminal Equipment)
  - TA(Terminal Adapter): TE2를 ISDN에서 이용 가능토록 하기 위한 접속 장치
  - LE(Local Exchange): 상대방 ISDN 교환기와 연결 제공
- 기준점 (참조점)
  - U(User): 내부망과 외부망을 구분
  - T(Terminal): 사용자 영역과 네트워크 영역 구분
  - S(System): 사용자 장비와 네트워크 장비를 구분
  - R(Rate): ISDN 장비와 비ISDN 장비를 구분

공중 통신망 (public switched network)

- ISDN(종합정보통신망, Integrated Service Digital Network)
- PSDN(공중데이터교환망, Public Switched Data Network)
- CSDN(회선교환데이터통신망, Circuit Switched Data Network)
- PSTN(공중전화교환망, Public Switched Telephone Network)

7.4 이동 통신망

다중 접속 기법

- 여러 사용자가 전송 매체를 공유하여 데이터를 송·수신하는 기법
- 시분할 다중 접속 (TDMA, Time Division Multiple Access)
- 주파수 분할 다중 접속 (FDMA, Frequency Division Multiple Access)
- 코드 분할 다중 접속 (CDMA, Code Division Multiple Access)

셀룰러(Cellular) 시스템

- 서비스 지역을 셀(Cell) 단위로 나눈 후 각 셀마다 기지국을 설치하여 서비스 영역을 담당하도록 하는 시스템

IMT-2000 (International Mobile Telecommunication-2000)

- 통신과 방송이 결합한 위성 멀티미디어 환경에서 가장 각광받을 것으로 기대되는 미래의 이동통신

7.5 위성 통신망

위성 통신망 (Satellite Transmission)

- 지상에서 쏘아올린 마이크로 주파수를 통신 위성을 통해 변환, 증폭한 후 다시 지상에 송신하는 방식

위성 통신의 특징

- 광범위한 지역에 서비스를 제공할 수 있음 (광대역 통신)
- 사용 주파수 대역은 3 ~ 30GHz의 극초단파
- 대역폭이 넓어 고속·대용량, 고품질의 정보 전송이 가능하고 통신비용도 저렴함
- 지상 무선 통신에 비해 에러율이 현저히 감소

- 장거리 통신이고 통신 위성을 거쳐야 하므로 전파 지연이 발생
- 사용 주파수가 높아질수록 기상 현상에 의한 신호 감쇠 현상이 심함
- 통신 위성의 궤도 위치는 적도 상공 약 35,800km 정도
- 위성 통신 시스템은 지구국, 채널, 통신 위성으로 구성

7.6 초고속 정보 통신망

초고속 정보 통신망 (정보 고속도로, Information Superhighway)

- 첨단 광섬유 케이블망을 이용하여 문자, 음성, 영상 등 대량의 멀티미디어 정보를 초고속으로 주고받을 수 있는 통신 시스템

초고속 정보 통신망 구축 기술

- ADSL (Asymmetric Digital Subscriber Line)
  - 기존의 설치된 전화의 동선케이블을 이용하여 고속 데이터 전송을 가능하게 하는 방식
  - 양쪽 방향의 전송속도가 서로 다름
  - 데이터통신과 일반 전화를 동시에 이용 가능
  - 최근에 고속 인터넷 통신을 위해 각광 받는 기술
- B-ISDN (광대역 종합 정보 통신망, Broadband-ISDN)
  - 신호의 전송 속도가 매우 높음
  - 서비스 신호 대역폭의 분포 범위가 넓음
  - 연속성 신호와 군집성 신호가 공존
- ATM (비동기 전송 모드, Asynchronous Transfer Mode)
  - B-ISDN의 핵심 기술이자 이의 실현 방안으로 적합한 통신방식
  - 디지털 정보를 다중 전송하는 방식
  - 정보는 셀(Cell) 단위로 나누어 전송

7.7 인터넷 (Internet)

백본망 (Backbone)

- 다른 네트워크 또는 같은 네트워크를 연결하여 그 중추역할을 하는 네트워크로 보통 인터넷의 주가 되는 기간망

인터넷 서비스

- WWW (World Wide Web): 웹 브라우저를 통해 전자 우편 서비스, FTP 서비스, HTTP 서비스 등 기존의 인터넷 서비스도 이용 가능
- 전자 우편 (E-Mail)
- 텔넷 (TELNET): 가상 터미널(VT, Virtual Terminal) 기능 수행
- FTP (File Transfer Protocol)

IP address

- 인터넷에 연결된 컴퓨터를 구분하기 위한 고유 주소
- IP address는 32bit 크기로 8bit 씩 4개의 필드로 분리 표기
- 네트워크 부분의 길이에 따른 분류
  - A class
    - 연결 가능 호스트 수: $2^{24} = 16,777,216$개

  - B class
    - 연결 가능 호스트 수: $2^{16} = 65,536$
  - C class
    - 연결 가능 호스트 수: $28 = 256$개
    - 실제로 할당할 수 있는 최대 IP 개수: 예약된 주소 2개를 제외한 254개

서브네팅 (Subnetting)

- 할당된 네트워크 주소를 여러 개의 작은 네트워크로 나누어 사용하는 것
- 서브넷 마스크(Subnet Mask): IP address에서 네트워크 ID와 호스트 ID를 구별하는 방식

도메인 네임 (Domain Name)

- 숫자로 된 IP address를 사람이 알아보기 쉬운 문자 형태로 표현한 것
- DNS(Domain Name System): IP 주소와 호스트 이름 간의 변환을 제공하는 분산 데이터베이스
- 도메인 네임의 구성

  www . hankook . co . kr
  - www: 호스트 컴퓨터 이름
  - hankook: 소속 기관
  - co: 소속 기관 종류
  - kr: 소속 국가

7.8 인터네트워킹

인터네트워킹 (Internetworking)

- 분산, 독립된 통신망 상호 간을 접속함으로써 통신망의 집합을 형성하거나 통신망을 광역화하는 것

관련 장치

- 리피터 (Repeater)
  - 신호의 감쇠 현상을 복원해 주는 장치
- 게이트웨이 (Gateway)
  - 프로토콜이 전혀 다른 네트워크 사이를 결합하는 장치
- 브리지 (Bridge)
  - 동종의 LAN과 LAN이 데이터 링크 계층(2계층)에서 서로 결합되어 있는 경우에 이들을 연결하는 요소
  - 로컬 네트워크 상호간 연결
  - Data의 움직임을 제어함으로써 내부와 외부 간 LAN의 정보량과 트래픽 양을 조절하는 기능이 있음
  - 서브넷(Subnet)을 브리지로 이용할 때 전송 가능 회선의 수: 브리지가 N개일 때, $N(N-1)/2$개

- 라우터 (Router)
  - 두 개의 서로 다른 형태의 네트워크를 상호 접속하는 3계층(네트워크 계층) 장비
  - 네트워크 계층에서 연동하여 경로를 설정하고 전달하는 기능을 제공하는 장비
  - 적절한 전송 경로를 선택하고 이 경로로 데이터를 전달
  - 게이트웨이 기능을 제공함

8 뉴 미디어와 멀티미디어

8.1 뉴 미디어

뉴 미디어의 특징

- 대용량 및 고속성
- 상호작용성 및 비동기성
- 쌍방향성 및 특정 다수자를 목표로 한 탈대중화
- 정보 형태의 다양화
- 네트워크화
- 분산적

뉴 미디어의 분류

- 유선계와 무선계로 분류
  - 유선계: CATV, 비디오텍스, 원격 회의, 팩시밀리, 퍼스널 컴퓨터 통신, LAN, VAN, ISDN 등
  - 무선계: 위성 통신, 텔레텍스트, HDTV, PCM 음성 방송, 팩시밀리 방송, 개인 휴대 통신 등
  - 패키지계: 비디오 디스크, 디지털 오디오 디스크, VTR, CD-ROM 등
- 방송계와 통신계로 분류
  - 방송계
  - 통신계

CATV (Cable Television)

- 공동시청안테나를 이용하는 텔레비전방식
- 난시청 지역에 고감도 안테나를 설치하여, 이를 통해 수신한 양질의 TV 신호를 일정한 전송로를 통하여 수요자에게 제공하는 뉴미디어시스템
- 수용자의 범위가 한정적
- 양방향 통신이 가능
- 기존 TV 방송과 컬러 색상 구조 및 주사방식이 동일함
- 다채널로서 방송뿐만 아니라 부가정보통신서비스가 가능
- 전송로는 동축케이블이나 광섬유케이블을 사용

아날로그 컬러 TV 방식의 국제 표준 규격

① NTSC (National Television System Committee)
② PAL (Phase Alternation Line)
③ SECAM (Séquentiel couleur à Mémoire)

비디오텍스 (Videotex)
- 화상정보가 축적된 정보센터의 데이터베이스를 TV수신기와 공중전화망에 연결해서 이용자가 화면을 보면서 상호대화 형태로 각종 정보검색을 할 수 있는 뉴미디어
- 양방향 통신이 가능

8.2 멀티미디어

멀티미디어 환경
- 멀티미디어 시스템에 두루 쓰이는 장비
  - CD-ROM 드라이버
  - MIDI 인터페이스
  - 스피커
- 멀티미디어 서비스 제공에 필요한 사항
  - 고속 통신망
  - Hypermedia
  - 신뢰도 높은 통신망

멀티미디어 기술
- 정지 영상 압축 기법
  - JPEG (Joint Photographer's Experts Group)
    정지 영상 압축의 국제 표준 방식
  - JBIG (Joint Bi-Level Image Coding Group)
    2차 화상을 대상으로 영상 압축 방식의 국제 표준
- 동영상 압축 기법
  - MPEG (Moving Picture's Experts Group)
    동영상 전문가 그룹에서 제정한 동영상 압축을 위한 국제 표준
  - MPEG-1
    가정용 VTR 품질(1.5Mbps)의 영상을 제공하기 위한 표준
  - H.261
    영상 전화나 영상 회의용 동화상 압축 부호화 방식의 국제 표준
  - DVI (Digital Video Interactive)
    동화상을 최고 1 / 120로 압축하는 데이터 압축 기술
- 오디오 압축 표준
  - WAVE
  - MIDI (Musical Instrument Digital Interface)
  - MP3 (MPEG Audio Player-3)
  - MHEG: 멀티미디어와 하이퍼미디어 정보에 관한 ISO 표준화 기술
- 기타 압축 기법
  - 허프만(Huffman) 압축 기법
  - LZW(Lempel-Ziv-Welch) 압축 기법

소프트웨어공학

1 소프트웨어 공학의 개념

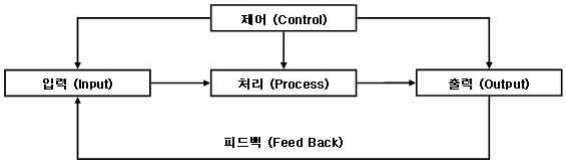
1.1 소프트웨어와 시스템

공학적으로 잘 작성된 소프트웨어(=좋은 소프트웨어)의 특성
- 사용자가 원하는 대로 동작해야 함
- 일정 시간 내에 주어진 조건하에서 원하는 기능을 실행할 수 있어야 함
- 신뢰성이 높아야 하며 효율적이어야 함
- 잠재적인 에러가 가능한 적어야 하며, 유지보수가 용이해야 함
- 적당한 사용자 인터페이스 제공으로 사용하기가 편리해야 함
- 문서화가 잘 되어 있어야 함

소프트웨어 생산성에 영향을 미치는 요소
- 개발자(프로그래머)의 능력
- 원활한 팀 의사 전달
- 제품의 복잡도

소프트웨어의 문서 표준이 되었을 때, 개발자가 얻는 이득
- 시스템 개발을 위한 분석과 설계가 용이함
- 프로그램 유지보수가 용이
- 프로그램의 확장성이 있음

시스템의 구성 요소



- 입력(Input): 처리 방법, 처리할 데이터, 조건을 시스템에 투입하는 것
- 처리(Process): 입력된 데이터를 처리 방법과 조건에 따라 처리하는 것
- 출력(Output): 처리된 결과를 시스템에서 산출하는 것
- 제어(Control): 자료를 입력하여 출력될 때까지의 처리 과정이 올바르게 진행되는지 감독하는 것
- 피드백(Feed Back): 출력된 결과가 예정된 목표를 만족시키지 못할 경우 목표 달성을 위해 반복 처리하는 것

1.2 소프트웨어 공학 (SE, Software Engineering)

소프트웨어 공학의 정의
- 가장 경제적으로 신뢰도 높은 소프트웨어를 만들기 위한 방법, 도구와 절차들의 체계

소프트웨어 공학의 공학(Engineering)이 가지는 의미
- 경제성
- 적시성
- 보편타당성

소프트웨어 공학이 나타나게 된 배경
- 소프트웨어(S/W) 비용의 증가
- 소프트웨어(S/W) 품질과 생산성의 재고
- 특정 개인에 의존한 시스템 개발

소프트웨어의 위기 (Crisis)
- 컴퓨터의 발달 과정에서 소프트웨어의 개발속도가 하드웨어의 개발속도를 따라가지 못해 사용자들의 요구사항을 감당할 수 없는 문제가 발생함을 의미
- 소프트웨어 위기의 원인
  - 소프트웨어의 특징에 대한 이해 부족
  - 소프트웨어의 관리 부재
  - 프로그래밍에만 치중
- 소프트웨어 위기의 현상
  - 개발 인력의 부족과 그로 인한 인건비 상승
  - 개발 기간의 지연
  - 하드웨어 비용을 초과하는 개발 비용 증가
  - 성능 및 신뢰성의 부족
  - 유지 보수의 어려움에 따른 엄청난 비용

소프트웨어 공학의 기본 원칙
- 현대적인 프로그래밍 기술 적용
- 지속적인 검증 시행
- 결과에 대한 명확한 기록 유지

1.3 소프트웨어 생명 주기

일반적인 소프트웨어 생명 주기
- 정의 단계
  - '무엇(What)'을 처리하는 소프트웨어를 개발할 것인지 정의하는 단계로, 관리자와 사용자가 가장 많이 참여하는 단계
- 개발 단계
  - '어떻게(How)'에 초점을 두고 실제적으로 소프트웨어를 개발하는 단계
- 유지보수 단계
  - 소프트웨어를 직접 운용하며, '변경(Change)'에 초점을 두고 여러 환경 변화에 따라 소프트웨어를 적응 및 유지시키는 단계
  - 소프트웨어 생명 주기 단계 중에서 시간과 비용이 가장 많이 요구되는 단계

폭포수 모형 (Waterfall Model)
- 폭포수 모형
  - 소프트웨어 개발 과정의 앞 단계가 끝나야만 다음 단계로 넘어갈 수 있는 선형 순차적 모형
  - 제품의 일부가 될 매뉴얼을 작성해야 함
  - 각 단계가 끝난 후 결과물이 명확히 나와야 함
  - 폭포수 모형은 소프트웨어 공학에서 가장 오래되고 가장 폭넓게 사용된 전통적인 소프트웨어 생명 주기 모형으로, 고전적 생명 주기 모형이라고도 함
- 폭포수 모형(Waterfall Model) 개발 순서

타당성 검토 ⇨ 계획 ⇨ 요구 분석 ⇨ 설계 ⇨ 구현(코딩) ⇨ 시험(검사, 테스트) ⇨ 유지보수

- 폭포수 모형의 장점
  - 모형의 적용 경험과 성공 사례가 많음
  - 단계별 정의가 분명하고, 전체 공조의 이해가 용이
  - 단계별 산출물이 정확하여 개발 공정의 기준점을 잘 제시
- 폭포수 모형의 단점
  - 개발 과정 중에 발생하는 새로운 요구나 경험을 설계에 반영하기 어려움
  - 처음부터 사용자들이 모든 요구사항들을 명확하게 제시해야 함
  - 단계별로 오류 없이 다음 단계로 진행해야 하는데 현실적으로 오류 없이 다음 단계로 진행하기는 어려움

프로토타입 모형 (Prototype Model)
- 프로토타입 모형
  - 시스템의 일부 혹은 시스템의 모형을 만드는 과정으로서, 요구된 소프트웨어의 일부를 구현하여, 추후 구현단계서 사용될 골격코드가 되는 모형
  - 실제 상황이 나오기 전에 가상으로 시뮬레이션을 통해 최종 결과물에 대한 예측을 할 수 있는 소프트웨어 수명 주기 모형
  - 최종 결과물의 만들어지기 전에 의뢰자가 최종 결과물의 일부 또는 모형을 볼 수 있음
  - 프로토타입은 발주자나 개발자 모두에게 공동의 참조 모델을 제공
  - 프로토타입은 구현 단계의 구현 골격이 됨
  - 구축하고자 하는 시스템의 요구사항이 불명확한 경우 가장 적절하게 적용될 수 있음
- 프로토타이핑(원형) 모형의 개발 작업 순서

요구 수집 ⇨ 빠른 설계 ⇨ 프로토타입 구축 ⇨ 고객평가 ⇨ 프로토타입 조정 ⇨ 구현

- 프로토타입 모형의 장점
  - 사용자 요구사항의 정확한 파악 및 충실한 반영
  - 요구사항의 변경이 용이
  - 정보문제의 본질에 대한 불확실성과 그 정보문제를 해결하기 위해 사용자가 제시하는 요구의 불확실성을 줄일 수 있음
- 프로토타이핑(Prototyping) 접근방법 채용에 따른 불확실성 결정요인
  - 지원이 필요한 일로부터의 요구연역(要求演繹)
  - 사용자와 분석자의 지식과 경험의 수준
  - 커뮤니케이션 문제가 일어날 가능성

나선형(Spiral) 모형
- 반복적인 작업을 수행하는 점증적 생명주기 모델
- 나선형 모형의 단계와 순서

Planning ⇨ Risk Analysis ⇨ Engineering ⇨ Customer Evaluation

  - 계획 수립(Planning): 목적, 다른 방안, 제약 조건을 결정
  - 위험 분석(Risk Analysis): 다른 방안을 분석하고 위험을 식별하고 해결
  - 공학적 개발(Engineering): 개선된 한 단계 높은 수준의 제품을 개발
  - 고객 평가(Customer Evaluation): 공학적 결과를 평가

2 소프트웨어 프로젝트 관리

2.1 소프트웨어 프로젝트 관리 개념
소프트웨어 프로젝트 관리의 개념
- 프로젝트 관리(Project Management)는 주어진 기간 내에 최소의 비용으로 사용자를 만족시키는 시스템을 개발하기 위한 전반적인 활동

프로젝트 관리 대상
- 계획 관리
- 품질 관리
- 위험 관리

효과적인 소프트웨어 프로젝트 관리를 위한 3P (3대 요소)
- People(사람): 인적자원
- Problem(문제): 문제인식
- Process(프로세스): 작업계획

2.2 프로젝트 계획 및 예측
프로젝트 계획 수립
- 프로젝트가 수행되기 전에 소프트웨어 개발 영역 결정, 필요한 자원, 비용, 일정 등을 예측하는 작업

소프트웨어 영역(Software Scope) 결정
- 개발될 소프트웨어의 영역을 결정하는 것
- 예측의 대상: 기능(Function), 성능(Performance), 신뢰도(Reliability), 비용, 일정, 참여인원 수, 요구되는 노력, 제약조건

소프트웨어 프로젝트를 신뢰성 있게 예측하는 방법
- 이미 수행된 유사 프로젝트를 참고함
- 프로젝트를 상대적으로 잘게 분리하여 예측함
- 경험적 예측 모델을 활용함
- 예측을 가능한 뒤로 미룸 (현실성이 부족함)
- 하나 이상의 자동화 측정 도구들을 이용함

프로젝트 계획 수립 시 고려사항
- 프로젝트 규모 파악: 제일 먼저 해야 할 작업
- 프로젝트 복잡도
- 구조적 불확실성의 정도
- 과거 정보의 가용성
- 위험성

프로젝트 비용 결정 요소

프로젝트 요소	자원 요소	생산성 요소
제품의 복잡도	인적 자원	개발자의 능력
시스템의 크기	하드웨어 자원	개발기간
요구되는 신뢰도	소프트웨어 자원	

2.3 비용 산정 기법
LOC(원시 코드 라인 수) 방법
- 프로그램의 라인 수를 평가하여 비용을 산정하는 방법
- 소프트웨어 각 기능의 LOC(원시 코드 라인 수)의 비관치, 낙관치, 기대치를 측정하여 예측치를 구하고, 이것으로 비용을 산정하는 방법
- 예측치 = { 낙관치 + (4 × 기대치) + 비관치 } / 6
- 산정 공식
  - 개발기간 = 노력(인월) / 투입인원
  - 개발비용 = 노력(인월) × 단위비용
  - 노력(인월) = 개발기간 × 투입인원 = LOC / 1인당 월평균 생산 코드 라인 수
  - 생산성 = LOC / 노력(인월)

COCOMO(COnstructive COst MOdel) 모형
- 보헴(Boehm)이 제안한 원시 프로그램의 규모에 의한 비용예측 모형
- 소프트웨어의 종류에 따라 다르게 책정되는 비용신장 방정식을 이용함
- 같은 규모의 프로그램이라도 그 성격에 따라 비용이 다르게 산정됨
- 비용 견적의 강도 분석 및 비용 견적의 유연성이 높아 소프트웨어 개발 비 견적에 널리 통용되고 있음
- 비용 산정 결과는 프로젝트를 완성하는데 필요한 노력(Man-Month)으로 나타냄
- 개발할 소프트웨어의 규모(LOC)를 예측한 후 이를 소프트웨어 종류에 따라 다르게 책정되는 비용 산정 방정식(공식)에 대입하여 비용을 산정

COCOMO 소프트웨어 프로젝트 모드 (=개발 유형)
- 소프트웨어 개발 유형은 소프트웨어의 복잡도 혹은 원시 프로그램의 규모에 따라 조직형(Organic Mode), 반분리형(Semi-Detached Mode), 내장형(Embedded Mode)으로 분류 할 수 있음
- 조직형 (Organic Mode)
  - 기관 내부에서 개발된 중·소규모의 소프트웨어로 일괄 자료 처리나 과학 기술 계산용, 비즈니스 자료 처리용으로 5만(50KDSI)라인 이하의 소프트웨어를 개발하는 유형
- 반분리형 (Semi-Detached Mode)
  - 조직형과 내장형의 중간형으로 트랜잭션 처리 시스템이나 운영체제, 데이터베이스 관리 시스템 등의 30만(300KDSI)라인 이하의 소프트웨어를 개발하는 유형
- 내장형 (Embedded Mode)
  - 내장형은 최대형 규모의 트랜잭션 처리 시스템이나 운영체제 등의 30만(300KDSI)라인 이상의 소프트웨어를 개발하는 유형

COCOMO 모형의 종류
- 기본(Basic)형 COCOMO
  - 기본형 COCOMO는 소프트웨어의 크기(생산 코드 라인 수)와 개발 유형(모드)만을 이용하여 비용을 산정하는 모형
- 중간(Intermediate)형 COCOMO
  - 중간형 COCOMO는 기본형 COCOMO의 공식을 토대로 사용하나, 여러 가지 다른 요인에 의해 비용을 산정하는 모형
- 발전(Detailed)형 COCOMO
  - 발전형 COCOMO는 중간(Intermediate)형 COCOMO를 보완하여 만들어진 방법으로 개발 공정별로 보다 자세하고 정확하게 노력을 산출하여 비용을 산정하는 모형

2.4 프로젝트 조직 구성
중앙집중형 팀 구성
- 한 관리자가 의사 결정을 하고 팀 구성원들은 그 결정에 따르는 구성 방식으로, 책임 프로그래머 팀 구성이라고도 함
- 중앙 집중형 팀 구성원의 역할
  - 책임 프로그래머
  - 프로그래머
  - 프로그램 사서
  - 보조 프로그래머

분산형 팀 구성
- 분산형 팀 구성은 팀원 모두가 의사 결정에 참여하는 비이기적인 구성 방식으로, 민주주의식 팀 구성이라고도 함

2.5 프로젝트 일정

프로젝트 일정 계획
- 프로젝트 일정(Scheduling) 계획은 프로젝트의 프로세스를 이루는 소작업을 파악하고 예측된 노력을 각 소작업에 분배하며, 소작업의 순서와 일정을 정하는 것

PERT / CPM(Program - Evaluation and Review technique / Critical Path Method)이 제공하는 도구
- 프로젝트 개발 기간을 결정하는 임계 경로(CP, Critical Path)를 제공
- 통계적 모델을 적용해서 개별 작업에 대한 가장 근접한 시간을 측정하는 기준이 됨
- 각 작업에 대한 시작 시간을 정의하여 작업들 간의 경계 시간을 계산할 수 있게 함

CPM (Critical Path Method, 임계 경로 기법)
- 프로젝트 완성에 필요한 작업을 나열하고 작업에 필요한 소요기간을 예측하는 데 사용함
- 프로젝트 작업 사이의 관계를 나타내며 최장경로를 파악
- 노드에서 작업을 표시하고 간선은 작업 사이의 전후 의존관계를 나타냄
- 박스노드는 프로젝트의 중간 점검을 뜻하는 이정표로 이 노드 위에는 예상완료 시간을 표시함
- 다른 일정계획안을 시뮬레이션 할 수 있음
- 병행작업이 가능하도록 계획할 수 있으며, 이를 위한 자원할당도 가능

브룩스(Brooks)의 법칙
- S/W Project 일정이 지연된다고 해서 Project 말기에 새로운 인원을 추가 투입하면 Project는 더욱 지연되게 된다고 주장하는 법칙
- 프로젝트 진행 중에 새로운 인력을 투입할 경우 작업 적응 기간과 부작용으로 인해 일정을 더욱 지연시키고, 프로젝트에 혼란을 가져오게 됨

간트 차트 (Gantt Chart)
- 프로젝트의 각 작업들이 언제 시작하고 종료되는지에 대한 작업 일정을 막대 도표를 이용하여 표시하는 프로젝트 일정표
- 수평 막대의 길이는 각 작업(Task)의 기간을 나타냄
- 시간선(Time-Line) 차트라고도 함
- 간트 차트에 포함되는 사항: 이정표, 작업 일정, 작업 기간, 산출물

2.6 소프트웨어 품질 보증

소프트웨어 공학에 적용되는 품질 표준(목표) 항목

품질 표준(목표)	의 미
정확성 (Correctness)	사용자의 요구 기능을 충족시키는 정도
신뢰성 (Reliability)	정확하고 일관된 결과를 얻기 위해 요구된 기능을 오류 없이 수행하는 정도 옳고 일관된 결과를 얻기 위하여 요구된 기능을 수행할 수 있는 정도
효율성 (Efficiency)	요구되는 기능을 수행하기 위해 필요한 자원의 소요 정도
무결성 (Integrity)	허용되지 않는 사용이나 자료의 변경을 제어하는 정도
사용 용이성 (Usability)	사용에 필요한 노력을 최소화하고 쉽게 사용할 수 있는 정도
유지보수성 (Maintainability)	변경 및 오류 사항의 교정에 대한 노력을 최소화하는 정도
유연성 (Flexibility)	소프트웨어를 얼마만큼 쉽게 수정할 수 있는가 하는 정도
시험 역량 (Testability)	의도된 기능을 수행하도록 보장하기 위해 프로그램을 시험할 수 있는 정도
이식성 (Portability)	다양한 하드웨어 환경에서도 운용 가능하도록 쉽게 수정될 수 있는 정도
재사용성 (Reusability)	전체나 일부 소프트웨어를 다른 목적으로 사용할 수 있는가 하는 정도
상호 운용성 (Interoperability)	다른 소프트웨어와 정보를 교환할 수 있는 정도

소프트웨어 품질 보증 (SQA, Software Quality Assurance)
- 어떤 항목이나 제품의 설정된 기술적 요구사항과 일치하는가를 적절하게 확인하는데 필요한 체계적이고도 계획적인 유형의 활동

정형 기술 검토 (FTR, Formal Technical Review)
- 가장 일반적인 검토방법으로 소프트웨어 기술자들에 의해 수행되는 소프트웨어 품질 보증 활동
- 유형
  - 검토 회의 (Walk-through, 워크스루)
    - 소프트웨어 개발의 각 단계에서 개최하는 기술 평가 회의
    - 소프트웨어 구성 요소와 같은 작은 단위를 검토하는 것
    - 검토를 위한 자료를 사전에 배포하여 검토하도록 함
    - 오류 검출에 초점을 두고 해결책은 나중으로 미룸
    - 발견된 오류는 문서화 함
  - 검열 (Inspections)
    - 검토 회의를 발전시킨 형태
    - 검열자는 검열 항목에 대한 체크리스트를 이용하여 작업 수행

- 정형 기술 검토의 지침사항
  - 논쟁과 반박을 제한하라.
  - 제품의 검토에 집중하라.
  - 참가자의 수를 제한하라.

소프트웨어의 신뢰성과 가용성
- 신뢰성: 프로그램이 주어진 환경에서 주어진 시간 동안 오류 없이 작동할 확률
- 가용성: 한 프로그램이 주어진 시점에서 요구사항에 따라 운영되는 확률
- 측정 방법
  - 소프트웨어의 간단한 신뢰성 측정은 MTBF로 가능함
  - MTBF (Mean Time Between Failure, 평균 고장 간격)
    - 수리가 가능한 시스템이 고장난 후부터 다음 고장이 날 때까지의 평균 시간
    - MTBF = MTTF + MTTR
    - MTTF (Mean Time To Failure, 고장에 대한 평균시간)
    - MTTR (Mean Time To Repair, 수선하기 위한 평균시간)
- 가용성 측정
  - 시스템의 총 운용 시간 중 정상적으로 가동된 시간의 비율

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

2.7 위험 관리

위험 관리 (Risk Analysis)
- 프로젝트 추진 과정에서 예상되는 각종 돌발 상황(위험)을 미리 예상하고, 이에 대한 적절한 대책을 수립하는 일련의 활동

위험의 종류
- 사용자 요구사항 변경: 가장 대표적인 위험 요소
- 인력 부족
- 예산 관리
- 일정 관리

위험 관리의 절차

위험 식별 ⇨ 위험 분석 및 평가 ⇨ 위험 관리 계획 ⇨ 위험 감시 및 조치

위험표 (Risk Table)

위험 내용	위험 종류	발생 확률	영향력	위험 감시 및 조치
인력이 부족하다.	BU	50%	2	
제품 규모 추정이 낮다.	PS	30%	3	
⋮	⋮	⋮	⋮	⋮

2.8 형상 관리

소프트웨어 형상 관리 (SCM, Software Configuration management)
- 개발 과정의 변화되는 사항을 관리하는 것
- 소프트웨어에 일어나는 수정이나 변경을 알아내고 제어하는 것
- 소프트웨어에 대한 변경을 관리하기 위해 개발된 일련의 활동
- 소프트웨어에 가해지는 연결을 제어 관리하는 것
- 소프트웨어의 생산물을 확인하고 소프트웨어 통제, 변경 상태를 기록하고 보관하는 일련의 관리 작업

형상 관리의 특징
- 유지보수 단계에서 행해짐
- 형상 관리에서 중요한 기술 중의 하나는 버전 제어 기술
- 형상 관리는 소프트웨어 개발의 전 단계에 적용되는 활동으로, 유지보수 단계에서 수행
- 형상 관리는 소프트웨어 개발의 전체 비용을 줄이고, 개발 과정의 여러 방해 요인이 최소화되도록 보증하는 것을 목적으로 함

형상관리(configuration management)의 관리 항목
- 정의 단계의 문서
- 개발 단계의 문서와 프로그램
- 유지보수 단계의 변경 사항

3 전통적 소프트웨어 개발 방법론

3.1 요구사항 분석

요구사항 분석 작업
- 문제 인식
- 평가와 종합
- 모델 제작
- 문서화 검토

요구사항 분석의 어려움
- 사용자의 요구사항이 모호하고, 부정확하며, 불완전함
- 개발자와 사용자 간의 지식이나 표현의 차이가 커서 상호 이해가 쉽지 않음
- 개발하고자 하는 시스템 자체가 복잡함

구조적 분석 도구 종류
- 자료 흐름도, 자료 사전, 소단위 명세서, 개체 관계도 등

자료 흐름도 (DFD, Data Flow Diagram)
- 자료 흐름도
  - 요구사항 분석에서 자료의 흐름 및 변환 과정과 기능을 도형 중심으로 기술하는 방법
- 자료 흐름도의 구성 요소 표기법
  - 프로세스(Process), 자료 흐름(Data Flow), 자료 저장소(Data Store), 단말(Terminator)

구성 요소	의 미	표기법
프로세스 (Process, 처리 공정)	·자료를 변환시키는 시스템의 한 부분(처리 공정)을 나타냄 ·버블이라고도 함	프로세스 이름
자료 흐름 (Data Flow)	·자료의 이동(흐름)을 나타냄	→
자료 저장소 (Data Store)	·시스템에서의 자료 저장소(파일, 데이터베이스)를 나타냄	자료 저장소 이
단말 (Terminator, 종착지)	·자료의 출처와 도착지를 나타냄 ·정보의 생산자와 소비자	단말 이름

- 자료 흐름도를 작성하는데 지침이 되는 항목
  - 어떤 처리(Process)가 출력자료를 산출하기 위해서는 반드시 입력 자료가 발생해야 함
  - 자료흐름은 처리(Process)를 거쳐 변환될 때마다 새로운 이름을 부여함
  - 처리(Process)와 하위 자료흐름도의 자료 흐름은 서로 일치해야 함

자료 사전 (DD, Data Dictionary)
- 자료 흐름도에 있는 자료를 더 자세히 정의하고 기록한 것
- 데이터를 데이터의 데이터 또는 메타 데이터(Meta Data)라고도 함

기 호	의 미	설 명
=	자료의 정의	~로 구성되어 있다.
+	자료의 연결	그리고 (and)
( )	자료의 생략	생략 가능한 자료 (Optional)
[|]	자료의 선택	또는 (or) 하나 이상의 선택이 필요할 때 사용하는 기호
{ }	자료의 반복	$\{\}_n$: n번 이상 반복 $\{\}^n$: 최대로 n번 반복 $\{\}_m^n$: m 이상 n 이하로 반복
* *	자료의 설명	주석 (Comment)

소단위 명세서 (Mini-Specification)
- 세분화된 자료 흐름도에서 최하위 단계 버블(프로세스)의 처리 절차를 기술한 것
- 프로세스 명세서라고도 함

개체 관계도 (ERD, Entity Relationship Diagram)
- 데이터 구조들과 그들 간의 관계들을 표현하기 위해 사용됨
- 개체 관계도의 작성 순서
  - 주요키를 포함하여 개체(Entity)의 속성을 모두 찾아냄
  - 기본적인 개체(Entity)와 주요키를 정의하며, 개체(Entity) 사이의 관계를 정의함
  - 1 : M 관계를 단순화하기 위해 속성 개체(Entity)를 추가하며, 연관 관계를 정의하여 M : N 관계를 표현함
  - 각 개체(Entity)를 정규화, 누락된 개체(Entity) 점검 및 클래스 구조가 필요한지 결정함

HIPO (Hierarchy Input Process Output)
- HIPO 다이어그램은 분석 및 설계 도구로서 사용됨
- 시스템의 분석 및 설계나 문서화할 때 사용되는 기법이며, 기본 시스템 모델은 입력, 처리, 출력으로 구성됨
- 기능과 자료의 의존 관계를 동시에 표현할 수 있음
- 구조도, 개요 도표 집합, 상세 도표 집합으로 구성됨
- 보기 쉽고 이해하기 쉬움

HIPO의 종류
- 가시적 도표 (Visual Table of Contents, 도식 목차)
  - 시스템의 전체적인 기능과 흐름을 보여주는 계층(Tree) 구조도
- 총체적 다이어그램 (Overview Diagram, 총괄도표 · 개요 도표)
  - 프로그램을 구성하는 기능을 기술한 것으로 입력, 처리, 출력을 기술
- 세부적 다이어그램 (Detail Diagram, 상세 도표)
  - 총체적 도표에 표시된 기능을 구성하는 기본 요소들을 상세히 기술하는 도표

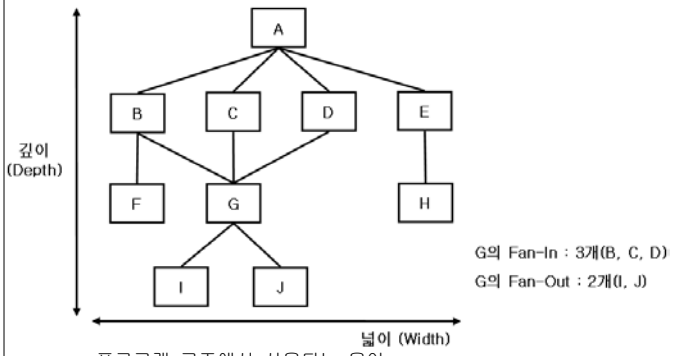
3.2 설계

설계(Design)
- 요구사항 분석 단계의 산출물인 요구사항 분석 명세서의 기능이 실현되도록 알고리즘과 그 알고리즘에 의해 처리될 자료구조를 문서화하는 것

소프트웨어 설계 모형
- 구성: 데이터 설계 – 아키텍처 설계 – 인터페이스 설계 – 절차 설계
- 설계 모형의 구조도

소프트웨어 설계에 사용되는 대표적인 3가지 추상화 기법
- 제어추상화: 제어의 정확한 메커니즘을 정의하지 않고 원하는 효과를 정하는 데 이용하는 방법
- 기능추상화: 입력 자료를 출력자료로 변환하는 과정을 추상화하는 방법
- 자료추상화: 자료와 자료에 적용될 수 있는 기능을 함께 정의함으로써 자료 객체를 구성하는 방법

프로그램 구조 (Program Structure)
- 소프트웨어의 구성 요소인 모듈의 계층적 구성을 나타내는 것으로, 제어 계층 구조라고도 함
- 프로그램의 순서, 선택, 반복과 같은 소프트웨어의 절차적인 처리 과정을 나타내지 않음
- 프로그램 구조는 일반적으로 트리 구조의 다이어그램으로 표기 (사각형은 모듈을 나타냄)



- 프로그램 구조에서 사용되는 용어

용 어	설 명
Fan-In (공유도, 팬-입력)	·얼마나 많은 모듈이 주어진 모듈을 호출(제어)하는가 ·주어진 한 모듈을 호출(제어)하는 상위 모듈 수
Fan-Out (제어도, 팬-출력)	·어떤 모듈에 의해 호출(제어)되는 모듈의 수 ·주어진 한 모듈이 호출(제어)하는 하위 모듈 수

바람직한 설계의 특징 (=좋은 설계에 대한 기준)
- 설계는 모듈적이어야 함 (독립적인 기능적 특성을 가진 요소(모듈)로 구성되어야 함)
- 설계는 자료와 프로시저에 대한 분명하고 분리된 표현을 포함해야 함
- 소프트웨어 요소(모듈)들 간의 효과적인 제어를 위해 설계에서 계층적 조직이 제시되어야 함
- 소프트웨어는 논리적으로 특별한 기능과 부기능을 수행하는 요소들로 나누어져야 함

모듈(Module)과 모듈화(Modularity)
- 모듈 (Module)
  - 소프트웨어를 각 기능별로 분할한 것
  - 소프트웨어 구조를 이루는 기본 단위
- 모듈화 (Modularity)
  - 소프트웨어를 각 기능별로 분할하는 것
  - 모듈화 장점: 융통성, 경제성, 확장성

결합도 (Coupling)
- 두 모듈 간의 상호 의존도를 나타낸 것
- 한 모듈과 다른 모듈 간의 상호 의존도 또는 두 모듈 사이의 연관 관계
- 독립적인 모듈이 되기 위해서는 각 모듈간의 결합도가 약해야 하며, 의존하는 모듈이 적어야 함
- 결합 정도에 따른 순서

데이터 결합도	스탬프 결합도	제어 결합도	외부 결합도	공통(공유) 결합도	내용 결합도

결합도 약함 <----------> 결합도 강함

응집도 (Cohesion)
- 모듈 안의 요소들이 서로 관련되어 있는 정도
- 응집 정도에 따른 순서

기능적 응집도	순차적 응집도	교환적 응집도	절차적 응집도	시간적 응집도	논리적 응집도	우연적 응집도

응집도 강함 <----------> 응집도 약함

효과적인 모듈화 설계 방안
- 응집도는 강하고, 결합도는 약해야 함
- 복잡도와 중복을 피함
- 모듈의 기능은 예측이 가능해야 하며 지나치게 제한적이어서는 안 됨
- 유지보수가 용이해야 함

설계 방법
- 데이터 설계
  - 요구사항 분석에서 생성된 여러 모델들을 소프트웨어를 구현하는 데 필요한 자료 구조로 변호나하는 것
- 아키텍처 설계
  - 프로그램의 구조를 개발하고, 소프트웨어 구성 요소들 간의 관계를 정의하는 것
- 인터페이스 설계
  - 소프트웨어와 상호 작용하는 시스템, 사용자 등과 어떻게 통신하는지를 기술하는 과정
- 프로시저 설계
  - 모듈이 수행할 기능을 절차적 기술로 바꾸는 것
  - N-S 차트 (Nassi-Schneiderman Chart, 나씨-슈나이더만 도표)
    - 구조적 프로그램을 표현하기 위해 고안됨
    - 논리의 기술에 중점을 둔 도형식 표현 방법
    - 조건이 복합되어 있는 곳의 처리를 시각적으로 명확히 식별하는 데 적합함
    - 알고리즘의 제어 구조
      - 반복 (Repeat ~ until, While, for)
      - 연속(순차) (Sequential)
      - 선택, 다중선택 (If ~ then ~ else, Case)

3.3 구현

구현 (Implementation)
- 설계 명세서가 컴퓨터가 알 수 있는 모습으로 변환되는 과정
- 시스템의 설계 명세서를 바탕으로 모듈 단위의 코딩과 디버깅 및 단위 테스트가 이루어지는 소프트웨어 개발 단계
- 프로그래밍 또는 코딩(Coding)이라고도 함

프로그래밍 언어 선정 기준
- 대상 업무의 성격
- 과거의 개발실적
- 개발담당자의 경험과 지식

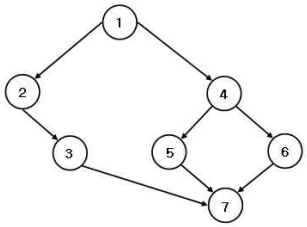
구조적 프로그래밍
- 신뢰성 있는 소프트웨어 생산과 코딩의 표준화 등을 위해 개발된 방법
- Dijkstra 방법론: 구조화 프로그래밍 방법론 중 선택과 반복 구조를 사용하는 것
- 구조적 프로그래밍의 기본 제어 구조
  - 순차(Sequence): 명령을 순서적으로 나열
  - 선택(Selection): 특정 논리에 기초하여 명령을 선택함
  - 반복(Iteration): 순환을 제공함

3.4 검사

검사(Test)
- 소프트웨어 품질을 평가하는 작업이며, 분석이나 설계, 코딩 결과를 최종적으로 점검하는 과정
- 검사 기법 종류: 화이트 박스 테스트, 블랙 박스 테스트

화이트 박스 테스트(White Box Test)
- 정의 및 특성
  - 모듈 안의 작동을 자세히 관찰 할 수 있음 (모듈 안의 논리적인 구조 검사)
  - 프로그램 원시 코드의 논리적인 구조를 커버(Cover)하도록 테스트 케이스를 설계하는 프로그램 테스트 방법
  - 원시 코드의 모든 문장을 한 번 이상 수행함으로써 수행됨
  - 프로그램의 제어 구조에 따라 선택, 반복 등의 부분들을 수행함으로써 논리적 경로를 제어함
  - 검사 대상의 가능한 경로를 어느 정도 통과하는지의 적용 범위성을 측정기준으로 함
  - Nassi-Shneiderman 도표를 사용하여 검정기준을 작성할 수 있음
- 기초 경로 검사 (Basic Path Test)
  - McCabe가 제안한 것으로 대표적인 화이트 박스 테스트 기법
  - 제어 흐름도: 제어 흐름을 표현하기 위해 사용되는 그래프
  - 순환 복잡도: 한 프로그램의 논리적인 복잡도를 측정하기 위한 소프트웨어의 척도로, 제어 흐름도 이론에 기초를 둠
  - McCabe에 의해 제안된 소프트웨어의 복잡성 측정
    - 영역은 그래프의 평면에서 둘러 싸여진 부분으로 묘사될 수 있음
    - 영역의 수는 경계된 영역들과 그래프 외부의 비경계지역의 수를 계산함
    - V(G)는 영역의 수를 결정함으로써 계산되어 짐

- 예: McCabe 방법에 의한 다음 그래프의 V(G)의 크기는?



⇨ 복잡도 V(G) = 화살표(E) - 노드 수(N) + 2 = 8 - 7 + 2 = 3
(또는 내부영역 + 외부영역 = 2 + 1 = 3 과 같이 구해도 됨)

블랙 박스 테스트(Black Box Test)
- 정의 및 특성
  - 제품이 수행할 특정 기능을 알기 위해서 각 기능이 완전히 작동되는 것을 입증하는 검사
  - 모듈의 구조보다 기능을 검사함
  - 기능 검사라고도 함
  - 동치 분할(Equivalence Partitioning)이라는 기법 사용
  - 원인-결과 그래프(Cause and Effect Graph)로 테스트 케이스 작성 가능
  - 블랙 박스 테스팅을 통해 발견 가능한 오류: 성능 오류, 부정확한 기능, 인터페이스 오류
- 블랙 박스 테스트 기법 종류
  - 동치 분할 검사 (Equivalence Partitioning Test)
  - 경계값 분석 (Boundary Value Analysis)
  - 원인-효과 그래프 검사 (Cause-Effect Graphing Testing)
  - 오류 예측 검사 (Fault Based Testing, Mutation Testing)
  - 비교 검사 (Comparison Testing)

소프트웨어 개발 단계와 테스트 전략
- 요구사항 분석 단계 ↔ 시스템 테스트
- 설계 단계 ↔ 통합 테스트
- 구현 단계 ↔ 단위 테스트
- 유지보수 단계 ↔ 시스템 테스트

소프트웨어 검사 단계 순서
단위(코드) 검사 ⇨ 통합(설계) 검사 ⇨ 검증(요구사항) 검사 ⇨ 시스템 검사

스터브 (Stub)
- 하향식 통합에 있어서 모듈 간의 통합 시험을 하기 위해 일시적으로 필요한 조건만을 가지고 임시로 제공되는 시험용 모듈

검증(요구사항) 검사 (Validation Test)
- 소프트웨어가 요구사항에 맞는지를 추적해 보는 데 중점을 두고 있는 시험 방법
- 검정 검사 기법의 종류: 형상 검사, 알파 검사, 베타 검사
  - 형상 검사: 소프트웨어 구성 요소, 목록, 유지보수를 지원하기 위해 필요한 모든 사항들이 제대로 표현되었는지를 검사
  - 알파 검사: 개발자의 장소에서 사용자가 시험하고 개발자는 뒤에서 결과를 지켜봄
  - 베타 검사: 실업무를 가지고 사용자가 직접 시험함

3.5 유지 보수

유지보수(Maintenance)의 개요
- 개발된 소프트웨어의 품질을 항상 최상의 상태로 유지하기 위한 것으로, 소프트웨어 개발 단계 중 가장 많은 노력과 비용이 투입되는 단계
- 유지보수는 소프트웨어가 사용자에게 인수되어, 설치된 후 발생하는 모든 공학적 작업
- 소프트웨어 유지보수를 용이하게 하려면 시험 용이성, 이해성, 수정 용이성, 이식성 등이 고려되어야 함
- 소프트웨어에 가해지는 연결을 제어 관리하는 것을 형상관리(configuration management)라고 함

유지보수 유형

유 형	설 명
수정(Corrective) 보수 (수리·교정·정정·하자 보수)	· 시스템을 운영하면서 검사 단계에서 발견하지 못한 오류를 찾아 수정하는 활동
적응(Adaptive) 보수 (환경 적응, 조정 보수)	· 소프트웨어 산물의 수명 기간 중에 발생하는 환경의 변화를 기존의 소프트웨어 산물에 반영하기 위하여 수행하는 활동
완전화(Perfective) 보수 (기능 개선, 기능 보수)	· 소프트웨어의 본래 기능에 새로운 기능을 추가하거나 성능을 개선하기 위해 소프트웨어를 확장 시키는 활동 · 유지보수 활동 중 가장 큰 업무 및 비용을 차지하는 활동
예방(Preventive) 보수	· 미래에 유지보수를 용이하게 하거나 기능을 향상시키기 위해소프트웨어를 변경하는 활동 · 예방 유지보수를 소프트웨어 재공학이라고도 함

유지보수 작업의 목적
- 하자 보수
- 환경 적응
- 예방 조치

외계인 코드 (Alien Code)
- 외계인 코드는 아주 오래 전에 개발되어 유지보수 작업이 어려운 프로그램을 의미
- 일반적으로 15년 전 또는 그 전에 개발된 프로그램을 의미하며, 문서화(Documentation)를 철저하게 해 두면 외계인 코드를 방지할 수 있음

4 객체지향 소프트웨어 공학

4.1 객체지향 소프트웨어 공학 개념

객체지향 기법 개념
- 현실 세계의 개체(Entity)를 기계의 부품처럼 하나의 객체(Object)로 만들어, 기계적인 부품들을 조립하여 제품을 만들듯이 소프트웨어를 개발할 때도 객체들을 조립해서 작성할 수 있도록 하는 기법을 말함
- 객체지향 프로그램의 장점
  - 자연적인 모델링이 가능
  - 소프트웨어의 재사용률이 높아짐
  - 소프트웨어의 유지보수성이 향상

객체 (Object)
- 필요한 자료 구조와 이에 수행되는 함수들을 가진 하나의 소프트웨어 모듈 (어트리뷰트+메소드)
- 어트리뷰트 (Attribute)
  - 객체가 가지고 있는 정보로 속성이나 상태, 분류 등을 나타냄
  - 데이터, 속성, 상태, 변수, 상수, 자료 구조라고도 함
  - 오브젝트의 상태는 어트리뷰트를 파악함으로써 알 수 있음
- 메소드 (Method)
  - 객체지향 시스템에서 전통적 시스템의 함수(Function) 또는 프로시저(Procedure)에 해당하는 연산 기능
  - 객체가 메시지를 받아 실행해야 할 객체의 구체적인 연산을 정의한 것
  - 객체가 갖는 데이터를 처리하는 알고리즘
  - 연산(Operation)이라고도 함
  - 오퍼레이션(Operation)은 속성(Attribute)을 변화시킴

클래스 (Class)
- 공통된 속성과 연산(행위)을 갖는 객체의 집합으로 객체의 일반적인 타입(Type)을 의미
- 하나 이상의 유사한 객체들을 묶어 공통된 특성을 표현한 데이터 추상화를 의미
- 모든 객체들은 더 큰 클래스의 멤버이고, 그 클래스에 대하여 이미 정의된 개별 자료구조와 연산이 상속이 되며, 그 때문에 개별 객체는 클래스의 인스턴스가 됨
- 메타 클래스(Meta Class): 클래스의 클래스를 의미하는 것으로 클래스 계층 트리의 최상단에 위치

메시지 (Message)
- 메시지는 객체(Object)들 간에 상호작용을 하는데 사용되는 수단
- 객체(Object)의 메소드(동작, 연산)를 일으키는 외부의 요구 사항임
- 메시지의 구성 요소
  - 메시지를 받는 객체(수신자)의 이름
  - 객체(Object)가 수행할 메소드 이름
  - 메소드를 수행할 때 필요한 인자(속성값)
- 메시지 전달은 오브젝트(object)에서 오브젝트로 이루어짐
- 메소드는 오브젝트로부터 메시지를 받을 때 시작되어 짐 (=오브젝트가 메시지를 받으면 메소드를 부른다(Invoke))

객체지향 기법의 기본 원칙
- 캡슐화 (Encapsulation)
- 정보 은닉 (Information Hiding)
- 추상화 (Abstraction)
- 상속성 (Inheritance)
- 다형성 (Polymorphism)

캡슐화 (Encapsulation)
- 캡슐화는 데이터(속성)와 데이터를 처리하는 함수를 하나로 묶은 것
- 객체지향 시스템에서 자료부분과 연산(또는 함수)부분 등 정보처리에 필요한 기능을 한 데두리로 묶는 것
- 객체 지향의 기본 원리인 정보은폐와 가장 밀접한 관계가 있음
- 객체지향 개념에서 연관된 데이터와 함수를 함께 묶어 외부와 경계를 만들고 필요한 인터페이스만을 밖으로 드러내는 과정
- 서로 관련 있는 데이터와 연산자를 하나로 묶어서 프로그램의 컴포넌트로 재사용할 수 있는 개념
- 객체지향의 캡슐화(encapsulation) 개념이 갖는 장점
  - 재사용이 용이
  - 인터페이스 단순화
  - 변경이 발생할 때 오류의 파급효과가 적음

정보 은닉 (Information Hiding)
- 캡슐화에서 가장 중요한 개념
- 다른 객체에서 자신의 정보를 숨기고 자신의 연산만을 통하여 접근을 허용하는 것
- 정보 은폐라고도 함
- 목적: 고려되지 않은 영향(Side Effect) 최소화

상속성 (Inheritance)
- 상속성은 이미 정의된 상위 클래스(부모 클래스)의 모든 속성과 연산을 하위 클래스가 물려받는 것을 말함
- 상위 클래스의 속성과 연산을 하위 클래스가 공유할 수 있기 때문에 객체(Object)와 클래스의 재사용, 즉 소프트웨어 재사용(Reuse)을 증대시키는 중요한 개념임

다형성 (Polymorphism)
- 객체지향 시스템에서 서로 다른 Class들이 같은 의미의 응답을 하는 특성
- 한 메시지가 객체에 따라 다른 방법으로 응답할 수 있는 것
- 많은 상이한 클래스들이 동일한 메소드 명을 이용하는 능력

4.2 객체지향 개발 단계와 분석

객체지향 소프트웨어 개발모형의 개발 단계
계획 ⇨ 분석 ⇨ 설계 ⇨ 구현 ⇨ 테스트 및 검증

객체지향 분석 (OOA, Object Oriented Analysis)
- 사용자의 요구사항을 분석하여 요구된 문제와 관련된 모든 클래스(객체), 이와 연관된 속성과 연산, 그들 간의 관계 등을 정의하여 모델링하는 작업

객체지향 분석의 방법론
- 럼바우의 분석 기법 (Rumbaugh Method)
- 부치(Booch) 방법
- Jacobson 방법
- Coad와 Yourdon 방법
- Wirfs-Brock 방법

럼바우의 분석 기법 (Rumbaugh Method)
- 모든 소프트웨어 구성 요소를 그래픽 표기법을 이용하여 모델링하는 기법
- 분석 절차: 객체 모델링 ⇨ 동적 모델링 ⇨ 기능 모델링
  - 객체 모델링 (Object Modeling)
    · 객체와 클래스를 연관화, 집단화, 일반화 관계를 중심으로 표현함
    · 정보 모델링이라고도 하며, 시스템에서 요구되는 객체를 찾아내어 속성과 연산 식별 및 객체들 간의 관계를 규정하여 객체 다이어그램(객체도)으로 표시하는 것
    · 분석 활동의 세 가지 모델 중 가장 중요하며 선행되어야 할 모델링
  - 동적 모델링 (Dynamic Modeling)
    · 객체지향 분석 과정 중 객체들의 제어 흐름, 상호 반응, 연산 순서를 나타내주는 과정
    · 동적 모델링에서는 객체나 클래스의 상태, 사건을 중심으로 다룸
  - 기능 모델링 (Functional Modeling)
    · 자료 흐름도(DFD)를 사용하여 프로세스들의 처리 과정을 기술하고, 처리 과정은 프로세스, 제어 흐름, 데이터 흐름, 데이터 저장소, 행위자를 가지고 표현함
    · 기능 모델링 순서
      * 입/출력 결정
      * 자료 흐름도 작성 (기능 의존 관계를 기술)
      * 기능의 내용을 상세히 기술
      * 제약 사항을 결정하고 최소화

- 각 모델에서 사용되는 그래픽 기법
  - 객체 모델링 - 객체도
  - 동적 모델링 - 상태도
  - 기능 모델링 - 자료 흐름도

4.3 객체지향 설계, 구현, 테스트

객체지향 설계 (OOD, Object Oriented Design)
- 객체지향 분석(OOA)을 사용해서 생성한 여러 가지 분석 모델을 설계 모델로 변환하는 작업으로, 시스템 설계와 객체 설계를 수행
- 최근 소프트웨어 제품의 전형적인 타입인 사용자 중심, 대화식 프로그램의 개발에 적합한 방식
- 객체의 속성과 자료구조를 표현함
- 구체적인 절차를 표현함
- 서브 클래스와 메시지 특성을 세분화하여 세부사항을 정제화함

객체지향 구현
- 객체지향 프로그래밍 (OOP, Object Oriented Programming)
  - 객체라는 단위를 이용하여 현실 세계에 가까운 방식으로 프로그래밍 함
  - 현실 세계에 가까운 방식이므로 이해하기 쉽고 조작하기 쉬운 프로그램을 개발할 수 있음
  - 객체모델의 주요 요소는 추상화, 캡슐화, 모듈화, 계층 등이 있음
  - 설계 시 자료와 자료에 가해지는 프로세스를 묶어 정의하고 관계를 규명함
  - 객체지향 프로그래밍 언어에는 Smalltalk, C++ 등이 있음
  - 유지보수가 쉽고 재사용 가능한 프로그램을 만들 수 있음
  - 이미 개발된 프로그램을 이용해 빠르고 확장된 프로그램을 개발할 수 있음

객체지향 테스트
- 클래스 테스트
  - 구조적 기법에서의 단위테스트와 같은 개념으로 가장 작은 단위, 즉 캡슐화 된 클래스나 객체를 검사하는 것 (단위 테스팅에 사용)
- 통합 테스트
- 확인 테스트
- 시스템 테스트

5 소프트웨어 공학의 발전적 주제

5.1 소프트웨어 재사용

소프트웨어 재사용 (Software Reuse)

- 이미 개발된 인정받은 소프트웨어의 전체 혹은 일부분을 다른 소프트웨어 개발이나 유지에 사용하는 것
- 1990년대의 클래스, 객체 등의 소프트웨어 요소는 소프트웨어 재사용성을 크게 향상시킴
- 소프트웨어 재사용에 가장 많이 이용되는 것은 프로그램, 즉 소스 코드(Source Code)임

소프트웨어를 재사용의 이점

- 개발시간과 비용을 단축
- 소프트웨어 개발의 생산성을 높임
- 프로젝트 실패의 위험을 줄여 줌
- 소프트웨어의 품질을 향상
- 시스템 구축 방법에 대한 지식을 공유하게 됨
- 시스템 명세, 설계, 코드 등 문서를 공유하게 됨

5.2 소프트웨어 재공학

소프트웨어 재공학 (Software Reengineering)

- 새로운 요구에 맞도록 기존 시스템을 이용하여 보다 나은 시스템을 구축하고, 새로운 기능을 추가하여 소프트웨어 성능을 향상 시키는 것
- 소프트웨어의 위기를 개발의 생산성이 아닌 유지보수의 생산성으로 해결하려는 방법을 의미
- 소프트웨어 재공학의 일반적인 개념은 데이터와 기능들의 개조 및 개선을 가해 유지보수 용이성을 향상시키자는 것임
- 재공학은 유지보수에 대한 장기적인 전략적 고려와 많은 비용, 시간, 자원을 요구함
- 재공학은 유지보수성, 생산성, 품질의 향상을 목적으로 함
- 재공학은 형식의 변경과 재설계 과정을 포함
- 소프트웨어 재공학도 자동화된 도구를 사용하여 소프트웨어를 분석하고 수정하는 과정을 포함
- 소프트웨어 재공학의 활동은 분석, 개조(재구성), 역공학, 이식 등으로 구분할 수 있음
- 유지보수의 문제로 인해 필요성이 대두됨

소프트웨어 재공학(Reengineering)의 목표

- 복잡한 시스템을 다루는 방법
- 다른 뷰의 생성
- 잃어버린 정보의 복구 및 제거
- 부작용의 발견
- 고수준의 추상
- 재사용 용이

역공학 (Reverse Engineering)

- 기존 소프트웨어를 분석하여 소프트웨어 개발 과정과 데이터 처리 과정을 설명하는 분석 및 설계 정보를 재발견하거나 다시 만들어 내는 작업
- 현재 프로그램으로부터 데이터, 아키텍처, 그리고 절차에 관한 분석 및 설계 정보를 추출하는 과정

5.3 CASE

CASE(Computer Aided Software Engineering)의 개념

- 소프트웨어 개발 과정에서 사용되는 요구 분석, 설계, 구현, 검사 및 디버깅 과정 전체 또는 일부를 컴퓨터와 전용 소프트웨어 도구를 사용하여 자동화하는 작업
- 소프트웨어 생명 주기의 전체 단계를 연결해 주고 자동화해 주는 통합된 도구를 제공해 주는 기술
- 소프트웨어 개발의 작업들을 자동화 하는 것
- 소프트웨어 도구와 방법론의 결합
- 통합 CASE는 소프트웨어 개발 주기 전체 과정을 지원함

CASE(Computer Aided Software Engineering) 분류

- 상위(Upper) CASE
  - 소프트웨어 생명 주기의 전반부에서 사용되는 것
  - 문제를 기술(Description)하고 계획하며 요구분석과 설계단계를 지원
  - 여러 가지 명세와 문서를 작성하는 데 사용
- 하위(Lower) CASE
  - 소프트웨어 생명 주기의 하반부에서 사용되는 것
  - 코드를 작성하고 테스트하며 문서화하는 과정을 지원
- 통합(Integrate) CASE
  - 소프트웨어 생명 주기에 포함되는 전체 과정을 지원
  - 공통의 정보 저장 장소와 통일된 사용자 인터페이스를 사용하여 도구들을 통합함

CASE(Computer Aided Software Engineering) 사용의 이점

- 소프트웨어 개발 기간을 단축하고 개발 비용을 절감할 수 있음
- 자동화된 기법을 통해 소프트웨어 품질이 향상
- 소프트웨어의 유지보수를 간편하게 수행
- 소프트웨어의 생산성이 향상되고 생산, 운용 활동을 효과적으로 관리·통제할 수 있음
- 품질과 일관성을 효과적으로 제어
- 소프트웨어 개발의 모든 단계에 걸친 표준을 확립
- 소프트웨어 모듈의 재사용성이 향상

기사친구 가족 여러분!!
자신감을 가지고 끝까지
최선을 다하셔서
기쁜 마음으로 합격수기를
올리시기 바랍니다~