2025-1-6-2

# 정보처리기능사

## 1. 전자계산기 일반

**컴퓨터의 특징**  
  
- 대용량성  
- 범용성  
- 호환성  
- 정확성  
- 신뢰성

**마이크로프로세서**  
CPU 기능을 수행하기 위해 만든 고밀도 직접 회로.  
주로 제어장치, 연산장치, 레지스터로 구성된다.  
  
**설계방식**

|  | RISC | CISC |
| --- | --- | --- |
| 명령어 개수 | 적음 | 많음 |
| 명령어 길이 | 고정 | 가변 |
| 실행속도 | 빠름 | 상대적으로 느림 |
| 하드웨어 | 단순 | 복잡 |
| 가격 | 저렴함 | 비쌈 |
| 사용 | 서버, 워크스테이션 | 일반PC |

* 중앙처리장치(CPU)의 구성   
  - 제어장치  
  - 부호기  
  - 명령 해독기  
  - 제어신호 발생기  
    
  - 연산장치(ALU)  
  - 누산기(ACC) : 연산 결과를 일시적으로 기억  
  - 가산기(Adder) : 누산기 + 데이터레지스터 값 ⇒ 누산기  
  - 보수기(Complementer) : 뺄셈/나눗셈 연산을 위한 보수변환  
  - 데이터 레지스터 : 연산 데이터 일시 저장  
  - 상태 레지스터 : 현재 상태를 나타내는 레지스터, 부호 저장  
    
  - 레지스터 : CPU 내부의 임시저장장치

| 레지스터 종류 | 주요 기능 | 특징 |
| --- | --- | --- |
| 프로그램 카운터(PC) | 다음 실행할 명령어의 주소 저장 | 명령어 인출 시 자동으로 증가 |
| 명령 레지스터(IR) | 현재 실행 중인 명령어 내용 저장 | 해독기가 참조하는 레지스터 |
| MAR | 메모리 접근할 주소 저장 | Memory Address Register |
| MBR | CPU-메모리 간 데이터 임시 저장 | Memory Buffer Register |
| 입출력 버퍼 레지스터 | CPU-I/O 장치 간 데이터 임시 저장 | 속도차 해결용 |
| 범용 레지스터 | 일반적인 데이터 저장 | 다목적으로 사용 가능 |
| 시프트 레지스터 | 비트를 좌/우로 이동 | 곱셈/나눗셈 연산에 활용 |
| 인덱스 레지스터 | 인덱스 주소 지정에 사용 | 배열 처리 시 유용 |
| 베이스 레지스터 | 기준 주소 저장 | 프로그램 재배치에 사용 |

데이터 레지스터, 상태 레지스터, 누산기는 레지스터로 분류하기도 하지만, 정보처리기능사에서는 연산 장치로 분류합니다.

명령 해독기와 부호기는 제어장치의 일부로, 레지스터가 아닙니다.

### 불 대수

1. 기본 연산

* AND (논리곱) : • 또는 ∧
  + 1 AND 1 = 1
  + 나머지는 모두 0
  + (x가 1이고 y가 1이면 1)
* OR (논리합) : + 또는 ∨
  + 0 OR 0 = 0
  + 나머지는 모두 1
  + (x나 y중 하나라도 1이면 1)
* NOT (보수) : ￣ 또는 '
  + NOT 1 = 0
  + NOT 0 = 1
  + (반대값)

1. 기본 법칙

* 교환법칙
  + X + Y = Y + X
  + X • Y = Y • X
* 결합법칙
  + (X + Y) + Z = X + (Y + Z)
  + (X • Y) • Z = X • (Y • Z)
* 분배법칙
  + X • (Y + Z) = (X • Y) + (X • Z)
  + X + (Y • Z) = (X + Y) • (X + Z)

1. 기본 정리  
   - X + 0 = X  
   - X + 1 = 1  
   - X • 0 = 0  
   - X • 1 = X  
   - X + X = X  
   - X • X = X  
   - X + X' = 1  
   - X • X' = 0  
   - (X')' = X  
   ​​

### 논리 회로

1. 가산기

* 반가산기(Half Adder)
  + A + B → Sum, Carry
  + XOR → Sum
  + AND → Carry
* 전가산기(Full Adder)
  + A + B + Carry → Sum, Carry
  + 반가산기 2개 + OR 게이트
* 멀티플렉서(MUX)
  + 개 입력 → 1개 출력
* 디멀티플렉서(DEMUX)
  + 1개 입력 → 개 출력
* 인코더(Encoder)
  + 개 입력 → 개 출력
* 디코더(Decoder)
  + 개 입력 → 개 출력
* 플립플롭(Flip-Flop)
* SR 플립플롭

S R | Q(next)  
0 0 | Q(현재값 유지)  
0 1 | 0  
1 0 | 1  
1 1 | 금지

* JK 플립플롭

J K | Q(next)  
0 0 | Q(현재값 유지)  
0 1 | 0  
1 0 | 1  
1 1 | Q'(토글)

* D 플립플롭

D | Q(next)  
0 | 0  
1 | 1

(클럭 상승 에지에서만 동작)

* T 플립플롭

T | Q(next)  
0 | Q(현재값 유지)  
1 | Q'(토글)

* **특징**
  + SR: 1-1 입력 금지
  + JK: SR의 개선버전, 모든 입력 허용
  + D: 가장 단순, 입력 그대로 출력
  + T: 토글용, 카운터에 많이 사용​​​​​​​​​​​​​​​​

### 자료 구성의 단위

| 단위 | 설명 |
| --- | --- |
| 비트(Bit) | - 0과 1의 2진수로 표현되는 최소 정보 단위 |
| 니블(Nibble) | - 4비트 - 16진수 한 자리 표현 가능 |
| 바이트(Byte) | - 8비트 - 문자 하나 표현 가능 - 기억장치의 주소 지정 최소 단위 |
| 워드(Word) | - CPU가 한 번에 처리할 수 있는 데이터 크기 - 전 워드(2바이트), 풀 워드(4바이트), 더블 워드(8바이트) 등 |
| 필드(Field) | - 데이터의 최소 의미 단위 - 하나의 항목을 정의하는 가장 작은 단위 (예: 이름, 주소) |
| 레코드(Record) | - 관련 있는 필드들의 모임 |
| 파일(File) | - 동일한 성격의 레코드들의 모임 |
| 데이터베이스(Database) | - 파일들의 집합체 |

추가:

* KB(1024B) → MB → GB → TB → PB
* 워드 크기는 CPU 아키텍처에 따라 다름  
  ​​​​​​​​​​​​​​​​

### 진법 변환

8진수와 16진수는 바로 변환하기 어려우므로 2진수나 10진수를 거쳐서 변환한다.

**10진수 → 2진수 변환**

* 2로 계속 나누면서 나머지를 아래에서 위로 읽는다.

149(2) → 2진수로 변환하기

149 ÷ 2 = 74 나머지 1  
74 ÷ 2 = 37 나머지 0  
37 ÷ 2 = 18 나머지 1  
18 ÷ 2 = 9 나머지 0  
9 ÷ 2 = 4 나머지 1  
4 ÷ 2 = 2 나머지 0  
2 ÷ 2 = 1 나머지 0  
1 ÷ 2 = 0 나머지 1  
  
나머지를 아래에서 위로 읽으면  
⇒ 10010101

**10진수 → 8진수 변환**

* 8로 계속 나누면서 나머지를 아래에서 위로 읽는다

149(10) → 8진수로 변환하기

149 ÷ 8 = 18 나머지 5  
18 ÷ 8 = 2 나머지 2  
2 ÷ 8 = 0 나머지 2  
  
나머지를 아래에서 위로 읽으면  
⇒ 225

**10진수 → 16진수 변환**

* 16으로 계속 나누면서 나머지를 아래에서 위로 읽는다
* 10~15는 A~F로 표시

149(10) → 16진수로 변환하기

149 ÷ 16 = 9 나머지 5  
9 ÷ 16 = 0 나머지 9  
  
나머지를 아래에서 위로 읽으면  
⇒ 95

**2진수 → 8진수**

* 오른쪽부터 3자리씩 끊어서 각각을 8진수로 변환

10010101(2) → 8진수로 변환하기

10|010|101  
↓ ↓ ↓  
2 | 2 | 5

**2진수 → 16진수**

* 오른쪽부터 4자리씩 끊어서 각각을 16진수로 변환

10010101(2) → 16진수로 변환하기

1001|0101  
 ↓ ↓  
 9 5

### 8진수/16진수 → 2진수 변환

* 각 자리를 2진수로 변환 (8진수는 3자리, 16진수는 4자리로)

225(8) → 2진수로 변환하기

2 2 5  
↓ ↓ ↓  
010 010 101

9A(16) → 이진수로 변환하기

9 A  
↓ ↓  
1001 1010

### 보수

**1의 보수 (1's Complement)**

* 모든 비트를 반대로 변환 (0→1, 1→0)
* 예시)
  + 00110 → 11001
  + 10101 → 01010
* 두 번 반전하면 원래 수로 돌아옴

**2의 보수 (2's Complement)**

* 1의 보수에 1을 더함
* 예시)
  + 00110 → 11001(1의보수) → 11010(2의보수)
  + 10101 → 01010(1의보수) → 01011(2의보수)
* 음수 표현에 사용

### 자료 표현 방식 : 내부적 표현

컴퓨터 내부에서 저장 공간을 효율적으로 사용하기 위한 방식

* 팩 10진 (Packed Decimal)  
  2자리를 1바이트에 꽉 채워 저장하기 때문에 **Pack**
  + 1 바이트에 2자리 저장
  + 연산은 가능하나 출력은 불가능 (언팩 10진으로 변경하여 출력)
  + 마지막 자리에는 부호 저장 (+ → C (1100), - → D (1101))

예) +129 표현

| Digit | Digit | Digit | Sign |
| --- | --- | --- | --- |
| 1 | 2 | 9 | C |
| 0001 | 0010 | 1001 | 1100 |

* 언팩 10진 (Unpacked Decimal, Zoned Decimal)  
  1자리에 1바이트로 여유를 두고 저장하기 때문에 압축을 풀었으니 **Unpack**
  + 1 바이트에 1자리 저장
  + 출력은 가능하나 연산은 불가능 (팩 10진으로 변경하여 연산)
  + 숫자 표현 시 Zone은 1111(F)로 채움
  + ASCII 코드와 호환

예) -2007 표현

| Zone | Digit | Zone | Digit | Zone | Digit | Sign | Digit |
| --- | --- | --- | --- | --- | --- | --- | --- |
| F | 2 | F | 0 | F | 0 | D | 7 |
| 1111 | 0010 | 1111 | 0000 | 1111 | 0000 | 1101 | 0111 |

* 음수 표현
  + 부호화 절대치 : 최상위 1비트를 양수는 0, 음수는 1로 표현
  + 부호화 1의 보수 : 부호화 절대치의 부호 비트를 제외한 나머지 비트 반전
  + 부호화 2의 보수 : 부호화 1의 보수에 1을 더함

부호화 절대치 방식과 부호화 1의 보수 방법은 0의 표현이 두개가 존재합니다.  
이는 연산을 복잡하게 하기때문에 대부분의 컴퓨터는 부호화 2의보수 방식을 채택합니다.

2의 보수는 0의 표현이 유일합니다.  
뺄셈을 덧셈으로 통일 가능하고, 자리올림 처리가 단순해 하드웨어 구현에 용이하며 연산을 효율적으로 수행합니다.

* 고정 소수점
  + 2진 정수 데이터 표현
  + 부호부, 정수부로 구성
  + 표현 범위는 적으나 연산 속도가 빠름
* 부동 소수점
  + 2진 실수 연산에 사용
  + 부호부, 지수부, 가수부(소수부)로 구성
  + 매우 큰 수나 작은 수의 표현에 용이하나 연산 속도가 느림

### 자료 표현 방식 : 외부적 표현

사용자와 컴퓨터 간 데이터 교환을 위해 사용되는 방식

#### 코드의 분류

* 가중치 코드 : 각 비트 자리에 가중치를 더해 계산하는 방식
* 비 가중치 코드 : 가중치가 없는 코드
* 에러 검출 코드 : 데이터 전송에서 발생하는 오류 검출 가능
* 자기 보수 코드 : 보수를 쉽게 구할 수 있는 코드

**주요 코드의 비트 구성**

| 코드 | Zone 비트 | Digit 비트 | 총 비트 | 표현 가짓수 |
| --- | --- | --- | --- | --- |
| BCD | - | 4비트 | 4비트 |  |
| ASCII | 3비트 | 4비트 | 7비트 |  |
| EBCDIC | 4비트 | 4비트 | 8비트 |  |

#### BCD 코드 (Binary Coded Decimal : 2진화 10진코드)

* 가중치 코드, 자기보수 코드
* 10진수의 각 자릿수를 2진수로 변환하는 코드
* 영문자의 대소문자를 구별하지 못함
* 8421, 2421 등 코드가 BCD 코드에 포함되며, 일반적으로 8421 코드를 말한다

예) 10진수 9 → 8421 코드

| x8 | x4 | x2 | x1 |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 1 |

예) 10진수 9 → 2421 코드

| x2 | x4 | x2 | x1 |
| --- | --- | --- | --- |
| 1 | 1 | 1 | 1 |

#### EBCDIC (Extended BCD Interchange Code : 확장 2진화 10진코드)

* 확장된 BCD 코드
* 대형 컴퓨터에서 사용되는 범용 코드이다

#### 패리티 비트 (parity Bit)

* 기본적인 에러 검출 코드 (교정은 불가능)
* 기존 코드에 1비트를 추가
* 1의 개수가 홀수 개인지 체크 → 기수검사
* 1의 개수가 짝수 개인지 체크 → 우수검사
* 에러 비트가 짝수 개일 경우 검출 불가

#### 해밍 코드 (Hamming Code)

* 1비트의 에러 검출 + 교정까지 가능
* 일반적으로 8421코드에 3비트 짝수 패리티로 구성

#### Gray Code

* 비 가중치 코드
* 아날로그/디지털 코드 변환기, 입출력 장치 코드로 사용
* 연산에 적합하지 않음
* **다음 숫자로 증가할 때 한 비트만 변환되는 특징이 있다**

#### 그 외 다양한 코드

* ASCII Code(American Standard Code for Information Interchange : 미국 표준 코드)
  + 비가중치 코드
  + 단일 문자를 표현하는 데이터 통신용 코드
  + 대소문자 구별이 가능하며, 자주 사용되는 특수문자를 포함한다
  + Excess-3 Code (3초과 코드)
* Excess-3 Code (3초과 코드)
  + 대표적인 비가중치 코드, 자기보수 코드
  + 8421 코드값에 3을 더함

### 명령어

#### 명령어 구성

| OP-Code | Mode | Register | Address |
| --- | --- | --- | --- |

* 연산자부, 명령 코드부(Operation Code, OP-Code)
  + 함수 연산기능 : 산술 / 논리 연산
    - 산술 연산 : ADD, SUB, MUL, DIV ,산술Shift
    - 논리 연산 : AND, NOT, OR, XOR, 논리Shift 둥
  + 전달 기능 : 중앙처리장치-주기억장치 간 정보 이동
    - 중앙처리장치 → 주기억장치 : LOAD
    - 주기억장치 → 중앙처리장치 : STORE
  + 제어 기능 : 프로그램 순서 분기 명령
    - 무조건 분기 : GOTO, JUMP 등
    - 조건 분기 : IF, ON GOTO 등
  + 입출력 기능 : 입출력 포트를 통한 입출력
    - INPUT, OUTPUT
* 자료부(Operand, 또는 주소부/번지부)
  + Mode : 주소 지정 방식
  + Register : 레지스터 지정
    - PC, IR, AC, MAR, MBR 등
  + Address : 실제 데이터 위치

#### 명령어 형식

* 0주소 명령어: OP
* 1주소 명령어: OP, ADDRESS
* 2주소 명령어: OP, ADDRESS1, ADDRESS2
* 3주소 명령어: OP, ADDRESS1, ADDRESS2, ADDRESS3

#### 주소 지정 방식

* **접근 방식에 의한 주소 지정 방식**

| 방식 | 메모리 참조 횟수 | 명령어 길이 | 처리 속도 |
| --- | --- | --- | --- |
| 묵시적 주소 지정 | 0 | 가장 짧다 | 가장 빠름 |
| 즉시 주소 지정 | 0 | 가장 길다 | 빠름 |
| 직접 주소 지정 | 1 | 길다 | 보통 |
| 간접 주소 지정 | 2 | 짧다 | 가장 느림 |

* **계산에 의한 주소 지정 방식**

| 방식 | 설명 |
| --- | --- |
| 상대 주소 지정 | 프로그램 카운터(PC)와 주소 부분의 값을 더해 주소를 지정 |
| 인덱스 주소 지정 | 인덱스 레지스터 값과 주소 부부의 값을 더해서 주소를 지정 |
| 베이스 주소 지정 | 베이스 레지스터 값과 주소 부분의 값을 더해서 주소를 지정 |

* **실제 기억 장소와 연관성이 있는 주소 지정 방식**
  + 절대 주소  
    : 기억 장치 고유의 16진수 주소
    - 장점 : 이해하기 쉽고 간편하다
    - 단점 : 공간 효율성이 떨어질 수 있다
  + 상대 주소  
    : 특정 주소를 기준으로 상대적인 위치
    - 장점 : 주소 지정이 용이하므로 공간 효율성이 좋다
    - 단점 : 자료 접근에 따른 계산 절차가 필요하다

### 입출력

* 입력장치와 출력장치의 종류
  + 입력장치 : 키보드, 마우스 ,OMR, OCR, 스캐너 등
  + 출력장치 : 모니터, COM, 프린터, X-Y 플로터 등
* 단말장치
  + 원격지에 설치되어 입출력 기능을 수행
* **채널 (입출력 채널)**
  + 주기억 장치와 입출력 장치의 중간에 위치
  + 처리 속도가 다른 두 장치의 처리 효율을 위해 존재
  + CPU의 간섭을 필요로 하지 않음 → CPU의 효율이 증가
  + 자체 메모리 공간은 없음

| 기법 | 설명 | 주요 특징 |
| --- | --- | --- |
| DMA(Direct Memory Access) | CPU 개입 없이 입출력 장치와 메모리 간 직접 데이터 전송 | - CPU 부하 감소 - 데이터 전송 효율 증가 - 하드웨어적 방식 |
| 스풀링(Spooling) | 느린 입출력 장치(프린터 등)과 CPU의 속도 차이 해소 | - 보조기억장치 사용 - 다중 작업 가능 - 소프트웨어적 방식 |
| 버퍼링(Buffering) | 스풀링과 목적 동일 | - 주기억장치 사용 - 단일 작업 - 하드웨어적 방식 |
| 인터럽트(Interrupt) | 예기치 않은 상황 발생 시 CPU에 알리는 방식 | - 현재 작업 중단 - 발생한 상황 우선 처리 (끼어들기) |

**추가 설명**

채널은 DMA, 스풀링, 버퍼링 방식을 모두 사용하며 CPU의 부담을 줄여줍니다.

DMA는 대량의 데이터 전송에 효율적이고, 스풀링은 프린터 작업, 버퍼링은 키보드 입력 등에 많이 사용합니다.

채널은 인터럽트를 사용하여 작업을 완료하거나 오류가 발생했을 때 CPU에게 보고하는 용도로 사용합니다. 인터럽트에 대해서는 바로 다음 부분에서 자세히 설명합니다.

### 인터럽트

컴퓨터에서 예기치 못했거나 특수한 상황이 발생했을 때 현재 진행중인 작업을 일시 중단하고 발생한 상태를 처리한 후 원래의 프로그램으로 돌아가는 일련의 처리 과정을 의미한다

* 인터럽트 동작 순서
  1. CPU에 인터럽트 발생 요청
  2. CPU는 현재 수행중인 프로그램을 저장 (**스택(Stack)** 사용)
  3. 인터럽트 처리
  4. 수행중이던 프로그램으로 복귀
* 인터럽트 종류  
  **하드웨어**
  + 정전 인터럽트
  + 기계고장 인터럽트
  + 외부 인터럽트
* **소프트웨어**
  + 입출력 인터럽트
  + 프로그램 인터럽트
  + 슈퍼바이저 콜 인터럽트

### 주기억장치

* **ROM (Read Only Memory)**
* - 읽기 가능, 쓰기 불가  
  - 전원 공급이 끊어져도 내용을 기억하는 **비휘발성**
  + 롬 바이오스 등 컴퓨터 구동에 필수적인 내용 저장
* **종류**
  + Mask ROM :수정 불가능
  + PROM : 한 번에 한해서 기록, 이후 수정 불가능
  + EPROM : 자외선을 이용하여 수정 가능
  + EEPROM : 전기를 이용하여 수정 가능
* **RAM (Random Access Memory)**
  + 읽기 / 쓰기가 가능
  + 전원 공급이 끊어지면 내용을 잃어버리는 **휘발성**
  + 일반 프로세스 / 데이터 등을 저장
* **종류**
  + SRAM : 캐시 메모리로 사용, 전력 소비가 많고 속도 빠름
  + DRAM : 주기억장치로 사용 , 일정 시간이 지나면 재충전이 필요하며 SRAM보다 느림

### 보조 기억 장치

* **자기 디스크 (Magnetic Disk)**
  + 자성을 띄는 금속 원판
  + 하드 디스크, 플로피 디스크가 이에 해당
* **주요 용어**
  + 트랙(Track) : 디스크의 일정 반경에 형성된 동심원. 바깥부터 0번
  + 섹터 (Sector) : 트랙을 일정한 크기로 나눈 구역. 정보 저장 기본 단위
  + 실린더 (Cylinder) : 동일한 수직선 상의 트랙들의 집합, 트랙 수와 동일
* **디스크에 저장 가능한 워드 수**  
  : 트랙 당 섹터 수 x 섹터 당 워드 수 x 트랙 수 x 면 수
* **자기 테이프 (Magnetic Tape)**
  + 순차 접근 기억 장치(SASD) : 순차 처리만 가능
  + 대용량 데이터 백업 용으로 사용
  + 접근 속도가 느리고 가격이 저렴함
* **주요 용어**
  + IRG : 논리 레코드 사이의 공백
  + IBG : 실제 데이터 입출력의 기본 단위
  + 블로킹 : 여러 레코드를 묶어서 블록으로 만드는 것
  + 블로킹 인수(BF) : 블록 안에 포함된 레코드 수

### 기타 기억 장치

* 플래시 메모리
  + 휴대용 기기의 저장장치로 사용
  + 충격에 강함, 비휘발성
* 캐시 메모리
  + CPU와 주기억장치 사이에 위치
  + 컴퓨터 처리속도 향상
  + SRAM 사용
* 가상메모리
  + 보조기억장치의 일부를 주기억장치처럼 사용하는 기법
  + 주기억장치의 크기보다 큰 프로그램을 불러와 실행 시 사용

## 2. 패키지 활용

### 데이터베이스

서로 관련있는 데이터의 집합체

**장점**

* 데이터 중복 최소화
* 데이터 공유 가능
* 일관성 유지
* 무결성 유지

**단점**

* 전산화 비용 증가
* 자료 처리 복잡성

**데이터베이스 디자인 단계**

1. 목적 정의
2. 테이블 정의
3. 필드 정의
4. 관계 정의

**데이터베이스 설계 단계**

1. 요구 조건 분석
2. 개념적 설계
3. 논리적 설계
4. 물리적 설계
5. 구현

### DBMS

DataBase Management System, 데이터베이스 관리 시스템  
사용자와 데이터베이스 사이에 위치하여 데이터베이스를 관리하고 사용자가 요구하는 데이터를 데이터베이스에서 찾아 제공하는 소프트웨어의 총칭

**필수기능**  
- **정**의 : 물리적인 구조를 정의  
- **조**작 : 데이터의 삽입, 삭제, 검색 등  
- **제**어 : 권한검사, 병행제어, 보안 등

### 스키마

Schema  
데이터베이스를 구성하는 개체, 속성, 관계의 형식과 상호 관계 전체를 정의하는 것  
데이터베이스 구조에 대한 정의와 제약조건을 기술하는 것

**종류**  
- 외부 스키마 (서브스키마)  
- 사용자나 응용 프로그래머가 필요로 한다  
- 뷰, 사용자 권한, 제약 조건을 포함  
  
- 개념 스키마 (스키마)  
- 각 응용 시스템이 필요로 하는 구조  
- 테이블, 관계, 구조, 접근권한, 보안정책, 무결성 규칙 등을 포함  
  
- 내부 스키마 (물리적 스키마)  
- 데이터베이스가 실제로 어떻게 저장되는가  
- 저장 공간, 접근 경로, 물리적 특성 등을 포함

**데이터베이스 사용자**

 - 데이터베이스 관리자(DBA)  
- 응용 프로그래머  
- 일반 사용자

**관계형 데이터베이스의 구성 요소**

 - 테이블(=표, 릴레이션): 데이터들을 행과 열로 표현  
- 튜플(Tuple): 행(row), 개체(record)  
- 속성(Attribute), 열(column), 필드(field)  
- 도메인(Domain): 하나의 속성에서 취할 수 있는 **값의 범위**  
- 차수(Degree): 속성의 개수 = 필드의 개수  
- 기수(Cardinality) 튜플의 개수

### SQL

- DDL (데이터 정의어)   
  
- CREATE : 객체 생성  
- ALTER : 객체 수정  
- DROP : 객체 삭제   
- CASCADE (참조 시 함께 삭제)  
- RESTRICT (참조 시 삭제 거부)  
  
- DML (데이터 조작어)  
  
- SELECT : 데이터 검색  
- **SELECT - FROM - WHERE**  
- DISTINCT: 데이터 중복 제거  
- ORDER BY: 데이터 정렬 (ASC:오름차순, DESC:내림차순)  
  
- INSERT: 새로운 튜플을 삽입  
- **INSERT - INTO - VALUE**  
  
- DELETE: 조건에 맞는 튜플을 삭제  
- **DELETE - FROM - WHERE**  
  
- UPDATE: 조건에 맞는 튜플의 내용을 변경  
- **UPDATE - SET - WHERE**  
  
- DCL (데이터 제어어)  
  
- GRANT : 권한 부여  
- REVOKE : 권한 제거  
  
**TCL**  
- COMMIT : 트랜잭션 → 원본 적용  
- ROLLBACK :트랜잭션 삭제

## 3. PC **운영체제**

사용자들이 보다 쉽고 간편하게 컴퓨터 시스템을 이용할 수 있도록 해주는 소프트웨어  
사용자와 컴퓨터의 인터페이스 역할을 수행한다

**운영 체제의 목적**

* 신뢰성
* 편리성
* 효율성
* 용이성

**운영체제의 종류**

* 단일 작업 처리(DOS)
* 다중 작업 처리(Windows, UNIX, LINUX 등)

**운영체제의 구성**

* 제어 프로그램 : 감시, 작업 제어, 자료 관리
* 처리 프로그램 : 언어 번역, 서비스, 문제 해결

**운영체제의 발달 과정과 운용 방식**  
  
1. 일괄 처리 시스템(Batch Processing System)  
**Batch 처리** : 데이터를 모아서 한번에 처리  
  
2. 실시간 처리 시스템(Real Time Processing System)  
데이터 발생 즉시 처리  
우주선, 은행, 전화, 레이더 등 실시간 처리 필요한 시스템에 사용  
  
3. 시분할 시스템(Time Sharing System)  
하나의 프로세스가 CPU를 독점하는 것을 방지  
시간을 쪼개 여러 프로세스에 할당  
**Round Robin 방식**  
  
4. 다중 프로그래밍 시스템(Multi-Progamming System)  
하나의 CPU와 주기억장치  
**여러 개의 프로그램을 동시에 처리**  
  
5. 다중 처리 시스템(Multi-Processing System)  
두 개 이상의 CPU를 사용하여 병렬 처리  
**여러 개의 프로그램을 동시에 처리**

### 프로세스

실행중인 프로그램

**프로세스 생명 주기**  
생성 → 준비 -> 실행 → 대기 → 종료

**주요 용어**  
- Dispatch  
준비 → 실행  
CPU 할당

* Time out  
  실행 → 준비  
  CPU 시간 만료
* Block  
  실행 → 대기  
  입출력 작업 요청 등
* Wake Up  
  대기 → 준비  
  입출력 작업이 완료 등
* **교착상태 (Deadlock)**  
  둘 이상의 프로세스가 서로의 자원을 기다리며 무한 대기하는 상태

**프로세스 스케줄링 : 비선점 (Non-preemptive)**  
할당받은 CPU를 빼앗기지 않는 방식

* FCFS (First Come First Served)
  + 큐에 도착한 순서대로 CPU 할당
  + 가장 단순한 방식
* SJF (Shortest Job First)
  + CPU 사용시간이 가장 짧은 프로세스 우선 처리
  + 평균 대기시간 최소화
* HRN (Highest Response Ratio Next)
  + 대기시간과 서비스시간 고려
  + SJF의 단점 보완
  + 우선순위 = (대기시간 + 서비스시간) / 서비스시간

**프로세스 스케줄링 : 선점 (Preemtive)**  
우선순위가 높은 프로세스가 CPU 강제 획득 가능

* SRT (Shortest Remaining Time)
  + 남은 실행시간이 가장 짧은 프로세스 우선
  + SJF의 선점형 버전
* Round Robin
  + 각 프로세스에 동일한 시간할당(Time Quantum)
  + 할당시간 종료 시 다음 프로세스에게 CPU 넘김
  + 시분할 시스템에 적합

### 언어 번역 프로그램

* **어셈블러(Assembler)**
  + 어셈블리어로 작성된 원시 프로그램을 기계어로 번역
* **컴파일러(Compiler)**
  + 고급 언어로 작성된 원시 프로그램을 목적 프로그램으로 번역
  + 전체 프로그램을 한번에 번역
  + 실행 파일 생성
* **인터프리터(Interpreter)**
  + 고급 언어나 중간 언어를 한 줄씩 해석하여 실행
  + 프로그램 생성 없이 직접 기계어로 변환하여 실행
  + 실행 속도는 느리지만 메모리 효율적

### 관련 서비스 프로그램

* **링커(Linker)**
  + 언어 번역 프로그램이 생성한 목적 프로그램들을 하나로 연결
  + 여러 목적 프로그램과 로드 모듈을 연결하여 실행 파일 생성
* **로더(Loader)**
  + 보조기억장치의 프로그램을 주기억장치로 적재
  + 주요 기능: 할당, 연결, 재배치, 적재

### 전체 언어 번역 과정

원시 프로그램 → (컴파일러) → 목적 프로그램 → (링커) → 로드 모듈 → (로더) → 실행

### DOS

**부팅**  
  
- 부팅 순서  
1. **IO.SYS** : 기본 입출력  
2. **MSDOS.SYS** : 파일관리/입출력 서비스  
3. **CONFIG.SYS** : 환경 설정  
4. **COMMAND.COM** : 명령어 처리기  
5. **AUTOEXEC.BAT** : 자동 실행  
  
- 웜부팅  
- Ctrl + Alt + Del  
- 일부 메모리 내용 유지  
- 콜드 부팅보다 빠름  
  
- 콜드부팅  
- 전원 연결 제거  
- 메모리 삭제

**DOS의 환경 설정 파일**

- CONFIG.SYS : 설정 파일, 반드시 루트 디렉토리에 존재해야 함

* LASTDRIVE : 드라이버의 최대 갯수 지정
* BUFFERS : 시스템이 사용 가능한 버퍼 수 지정

**DOS 명령어**

| 분류 | 명령어 | 설명 |
| --- | --- | --- |
| **내부 명령어** | CLS | 화면 지우기 |
|  | DIR | 디렉토리 내용 표시 |
|  | VER | DOS 버전 표시 |
|  | COPY | 파일 복사 |
|  | DATE | 날짜 확인/변경 |
|  | TIME | 시간 확인/변경 |
|  | MD | 디렉토리 생성 |
|  | CD | 디렉토리 이동 |
|  | RD | 디렉토리 삭제 |
|  | PROMPT | 프롬프트 설정 |
|  | VOL | 볼륨 정보 표시 |
|  | TYPE | 파일 내용 표시 |
| **외부 명령어** | ATTRIB | 파일 속성 변경 |
|  | FORMAT | 디스크 초기화 |
|  | CHKDSK | 디스크 상태 점검 |
|  | FDISK | 파티션 설정 |
|  | LABEL | 볼륨 레이블 설정 |
|  | SCANDISK | 디스크 검사 |
|  | DISKCOPY | 디스크 전체 복사 |
|  | XCOPY | 디렉토리 전체 복사 |
|  | DELTREE | 디렉토리 삭제 |
|  | SORT | 정렬 |
|  | SYS | 시스템 파일 복사 |
| **필터 명령어** | MORE | 한 화면씩 출력 |
|  | FIND | 문자열 검색 |
|  | SORT | 내용 정렬 출력 |
| **DIR 옵션** | /P | 한 화면씩 표시 |
|  | /W | 한 줄에 5개씩 출력 |
|  | /O | 정렬 방식 지정 |
|  | /S | 하위 디렉토리 포함 |
|  | /A | 속성별 표시 |
|  | /H | 숨김 파일 표시 |
| **디스크 관리** | FORMAT | 디스크 초기화 |
|  | DISKCOPY | 디스크 전체 복사 |
|  | CHKDSK | 상태 점검 |
|  | SCANDISK | 상태 검사 |
|  | FDISK | 파티션 설정 |
|  | DEFRAG | 조각 모음 |
| **디렉토리 관리** | MD | 디렉토리 생성 |
|  | RD | 디렉토리 삭제 |
|  | CD | 디렉토리 이동 |
|  | DELTREE | 디렉토리와 내용 삭제 |
|  | XCOPY | 하위 디렉토리 포함 복사 |

### UNIX / LINUX

서버용 컴퓨터에서 주로 사용된다. 이식성과 확장성이 뛰어난 오픈소스 시스템이다.  
시분할 시스템을 위해 설계되었으며, 다중 사용자와 다중 작업을 지원한다

C언어로 작성되었다.

파일시스템은 계층적 트리구조이다.

**구성**  
  
- 커널: 가장 핵심적인 부분, 하드웨어 보호, 인터페이스 역할  
  
- 쉘: 명령을 해석하여 커널로 처리할 수 있도록 전달하는 명령 인터프리터  
  
- 유틸리티: 일반 사용자가 작성한 응용 프로그램을 처리하는데 사용  
  
**주요 명령어**

| 분류 | 명령어 | 설명 | DOS 동일 명령어 |
| --- | --- | --- | --- |
| **시스템/프로세스** | kill | 프로세스 종료(삭제) | - |
|  | fork | 새로운 프로세스 생성(복제) | - |
|  | finger | 사용자 정보 조회 | - |
|  | ps | 프로세스 상태 정보 표시 | - |
|  | ping | 네트워크 문제 진단 | - |
| **디렉터리** | pwd | 현재 디렉터리 경로 출력 | - |
|  | ls | 현재 디렉터리의 파일 목록 표시 | dir |
|  | mkdir | 디렉터리 생성 | md |
|  | rmdir | 디렉터리 삭제 | rd |
|  | cd | 디렉터리 이동 | cd |
| **파일** | cp | 파일 복사 | copy, xcopy |
|  | rm | 파일 삭제 | del |
|  | cat | 파일 내용 화면 표시 | type |
|  | chmod | 파일 보호모드 설정 | attrib |

## 4. 정보 통신 일반

### 정보 통신의 기본 개념

* **정의**: 전기 통신(정보 전송) + 컴퓨터(정보 처리)
* **온라인 시스템 3대 구성**:
  + 단말장치(단말기)
  + 통신 회선(전송 매체)
  + 전송 제어 장치

### 전처리기(FEP)

* FEP(Front End Processor)의 기능:
  + 통신 회선 및 단말장치 제어
  + 메시지 조립과 분해
  + 전송 메시지 검사
  + 오류 검출 수행

### 정보 통신 시스템 처리 형태

1. **온/오프라인 시스템**
   * 오프라인: 임시 기록매체 필요
   * 온라인: 직접 데이터 처리
2. **처리 방식**
   * 일괄 처리(Batch): 데이터 축적 후 처리
   * 실시간 처리(Real Time): 즉시 처리
   * 시분할 처리(Time Sharing): 대화형 처리

### 통신 속도와 통신량

* **변조속도**: Baud (1초당 신호 변화 수)
* **신호속도**: Bps (1초당 전송 비트 수)
  + 모노비트(1비트), 디비트(2비트)
  + 트리비트(3비트), 쿼드비트(4비트)

### 통신 방식

1. **단방향(Simplex)**
   * 한쪽 방향만 전송
   * 예: 라디오, TV
2. **반이중(Half-Duplex)**
   * 양방향 가능, 동시 전송 불가
   * 예: 무전기, 모뎀
3. **전이중(Full-Duplex)**
   * 동시 양방향 전송 가능
   * 예: 전화

### 신호 변환 방식

1. **디지털 변조** (모뎀 사용)
   * ASK(진폭): 잡음에 약함
   * FSK(주파수): 잡음에 강함
   * PSK(위상): 위상 차이로 변조
2. **펄스 코드 변조(PCM)** (코덱 사용)
   * 아날로그→디지털 변환
   * 과정: 표본화→양자화→부호화→복호화→여과화

### 전송 제어

1. **절차**
   * 회선 연결→링크 확립→메시지→링크 종결→회선 절단
2. **제어 문자**
   * SYN: 문자 동기
   * SOH: 헤딩 시작
   * STX: 본문 시작
   * ETX: 본문 종료
   * ETB: 블록 종료
   * ACK: 긍정 응답
   * NAK: 부정 응답

### 오류 제어

1. **ARQ 방식**
   * 정지-대기 ARQ
   * 연속 ARQ (Go-Back-N, Selective Repeat)
   * 적응적 ARQ
2. **오류 검출**
   * 패리티 검사: 검출만 가능
   * 해밍코드: 검출 및 수정 가능

### 접속 규격

* **DTE**: 단말장치
* **DCE**: 회선종단장치
* **RS-232C**: 25핀 커넥터
  + 2(송신), 3(수신)
  + 4(송신 요청), 5(송신 준비)

### 네트워크 구성

1. **망 구성 형태**
   * 성형(Star): 중앙 집중형
   * 링형(Ring): 포인트-투-포인트
   * 버스형(Bus): 단일 회선
   * 계층형(Tree): 분산형
   * 망형(Mesh): n(n-1)/2 회선
2. **네트워크 장비**
   * 리피터: 신호 증폭
   * 브리지: 1:1 연결
   * 라우터: 최적 경로 선택
3. **네트워크 규모**
   * LAN → MAN → WAN

# Reference

* 이기적 정보처리기능사 필기
* 시나공 정보처리기능사 필기