**Chapter 02 데이터**

***02-1*** **0과 1로 숫자를 표현하는 방법**

컴퓨터: **0**과 **1**로 모든 정보를 표현, 0과 1로 표현된 정보만을 이해

**<정보단위>**

**비트(bit)**:0과 1을 나타내는 가장 작은 정보 단위

n비트는 2ⁿ가지 정보를 표현

ex) 2비트로 표현 가능한 정보: 00, 01, 10, 11

프로그램의 크기:표현의 편의를 위해 비트보다 큰 단위 사용

-> 바이트, 킬로바이트, 메가바이트, 기가바이트, 테라바이트

**바이트(byte)**:8개의 비트를 묶은 단위(1바이트=8비트) -> 2⁸=256개의 정보 표현 가능

**킬로바이트(kB)**:1바이트 1,000개를 묶은 단위(1,000byte)

**메가바이트(MB)**:1킬로바이트 1,000개를 묶은 단위(1,000kB)

**1기가바이트(GB)**:1메가바이트 1,000개를 묶은 단위(1,000MB)

**1테라바이트(TB)**:1기가바이트 1,000개를 묶은 단위(1,000GB)

**워드(word)**:CPU가 한 번에 처리할 수 있는 데이터 크기

-> 현대 컴퓨터의 워드 크기는 대부분 32비트 또는 64비트

하프 워드(half word):정의된 워드의 절반 크기

풀 워드(full word):워드의 1배 크기

더블 워드(double word):워드의 2배 크기

**<이진법>**

**이진법**:수학에서 0과 1만으로 모든 숫자를 표현하는 방법

**이진수**:이진법으로 표현한 수 -> 컴퓨터에 어떤 숫자를 알려주려면 이진수로 알려주어야 함

표기 시 이진수 끝에 아래첨자₍₂₎ (수학적표기)를 붙이거나 이진수 앞에 0b(코드 상)를

붙임

ex) 1000₍₂₎ , 0b1000

이진수의 **음수 표현**:2의 보수를 구해 이 값을 음수로 간주

**2의 보수**:**사전적 의미**는 ‘어떤 수를 그보다 큰 2ⁿ에서 뺀 값

ex) 11₍₂₎의 2의 보수: 100₍₂₎에서 11₍₂₎을 뺀 01₍₂₎

**쉽게 표현**하면 ‘모든 0과 1을 뒤집고, 거기에 1을 더한 값’

ex) 11₍₂₎을 뒤집으면 00₍₂₎이고 거기에 1을 더한 값은 01₍₂₎

Quiz> 1011₍₂₎의 음수를 구해봅시다

Answer> 0101₍₂₎

**플래그**:이진수가 양수인지 음수인지 구분하기 위해 사용(컴퓨터가 부호를 헷갈릴 일은 없음)

-> 관련한 내용은 04장에서 자세히 다룰 것

**<십육진법>**

**십육진법**:수가 15를 넘어가는 시점에 자리 올림을 하는 숫자 표현 방식

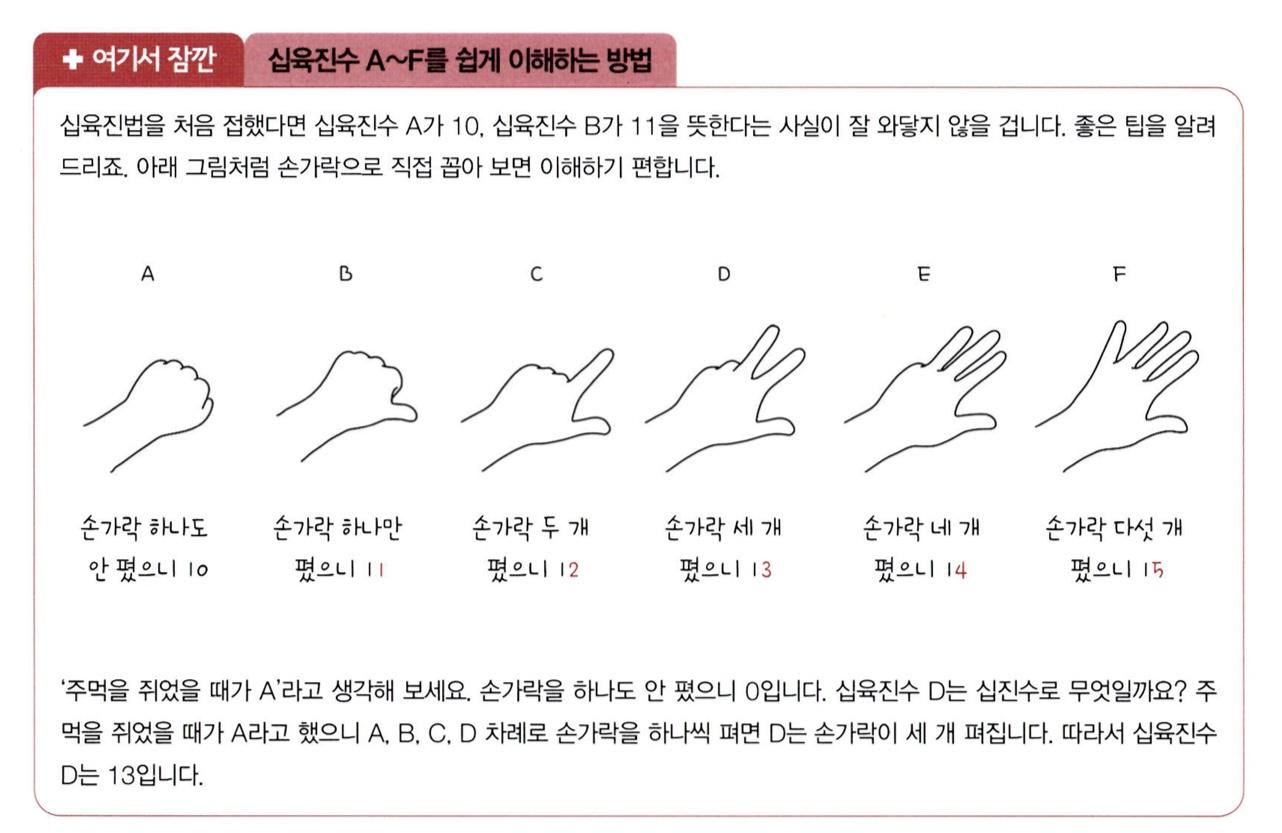
이진법으로 표현 시 숫자의 길이가 너무 길어지는 단점을 보완하여 더 적은

자릿수로 더 많은 정보를 표현 가능

표기 시 숫자 뒤에 아래첨자 ₍₁₆₎(수학적 표기)을 붙이거나 숫자 앞에 0x(코드 상)를

붙여 구분 ex) 15₍₁₆₎ , 0x15

10,11,12,13,14,15를 A,B,C,D,E,F로 표기



**십육진수 -> 이진수**

: 십육진수 한 글자를 4비트의 이진수로 간주

ex) 1A2B₍₁₆₎ -> 0001₍₂₎ 1010₍₂₎ 0010₍₂₎ 1011₍₂₎ -> 0001101000101011₍₂₎

**이진수 -> 십육진수**

**:**이진수 숫자를 네 개씩 끊고 끊어 준 네 개의 숫자를 하나의 십육진수로 변환

ex) 11010101₍₂₎ -> 1101₍₂₎ 0101₍₂₎ -> D₍₁₆₎ 5₍₁₆₎ -> D5₍₁₆₎

***02-2*** **0과 1로 문자를 표현하는 방법**

**<문자 집합과 인코딩>**

**문자 집합**: 컴퓨터가 인식하고 표현할 수 있는 문자의 모음

**문자 인코딩**: 문자를 0과 1로 변환하는 과정

**문자 코드**: 인코딩 후 0과 1로 이루어진 결과값

**문자 디코딩**:0과 1로 이루어진 문자 코드를 사람이 이해할 수 있는 문자로 변환하는 과정

**<아스키 코드>**

**아스키**:초창기 문자 집합 중 하나로 영어 알파벳,아라비아 숫자, 일부 특수 문자 포함

7비트로 표현. 2⁷=128개의 문자를 표현 가능

(실제는 8비트를 사용하지만 1비트는 오류 검출을 위해 사용되는 패리티 비트)

**아스키 코드**:아스키 문자에 대응된 고유한 수.

아스키 코드를 이진수로 표현함으로써 아스키 문자를 0과 1로 표현 가능(인코딩)

한글 및 영어권 외의 나라의 언어는 표현 불가

**확장 아스키**:더 다양한 문자 표현을 위해 1비트를 추가한 8비트의 확장 아스키가 등장했지만

표현 가능한 문자의 수는 256개 뿐

**<EUC-KR>**

**EUC-KR**:한글 인코딩 방식. 두가지 방식인 완성형과 조합형이 존재

초성,중성,종성이 모두 결합된 한글 단어에 2바이트(16비트) 크기의 코드를 부여

-> 네 자리 십육진수로 표현 가능

**완성형 인코딩**:초성,중성,종성의 조합으로 이루어진 완성된 하나의 글자에 고유한 코드 부여

**조합형 인코딩**:초성,중성,종성 각각 따로 비트열을 할당하여 그것을 조합해 하나의

글자코드를 완성하는 인코딩 방식

**CP949**:EUC-KR의 확장된 버전으로 모든 한글을 표현할 수 없는 EUC-KR의 문제를 해결하기

위해 등장, 그러나 이마저도 한글 전체를 표현하기에 넉넉하지는 않음

**<유니코드와 UTF-8>**

**유니코드**:EUC-KR보다 훨씬 다양한 한글을 포함하며 대부분 나라의 문자, 특수문자,화살표나

이모티콘까지도 코드로 표현할 수 있는 통일된 문자 집합

글자에 부여된 값 자체를 인코딩된 값으로 삼지 않고 이 값을 다양한 방법으로 인코딩

인코딩 방법에는 크게 UTF-8, UTF-16, UTF-32등이 있음. 이중 UTF-8이 가장 대중적

<https://www.unicode.org/charts/PDF/UAC00.pdf>

-> 유니코드 문자 집합에서 한글 문자마다 고유한 값이 부여되는 것 확인 가능

ex) 한 -> D55C₍₁₆₎ , 글 -> AE00₍₁₆₎

**UTF-8**:통상 1바이트부터 4바이트까지의 인코딩 결과를 만들어 냄

인코딩한 결과가 몇 바이트가 될지는 유니코드 문자에 부여된 값의 범위에 따라 결정



한(D55C₍₁₆₎) 글(AE00₍₁₆₎) 두 글자 모두 0800₍₁₆₎에서 FFFF₍₁₆₎사이 -> 각각 3바이트로 표현

D₍₁₆₎ 5₍₁₆₎ 5₍₁₆₎ C₍₁₆₎ -> 1101₍₂₎ 0101₍₂₎ 0101₍₂₎ 1100₍₂₎

1바이트:1110XXXX 2바이트:10XXXXXX 3바이트:10XXXXXX

-> 11101101 10010101 10011100₍₂₎

A₍₁₆₎ E₍₁₆₎ 0₍₁₆₎ 0₍₁₆₎ -> 1010₍₂₎ 1110₍₂₎ 0000₍₂₎ 0000₍₂₎

-> 11101010 10111000 10000000₍₂₎