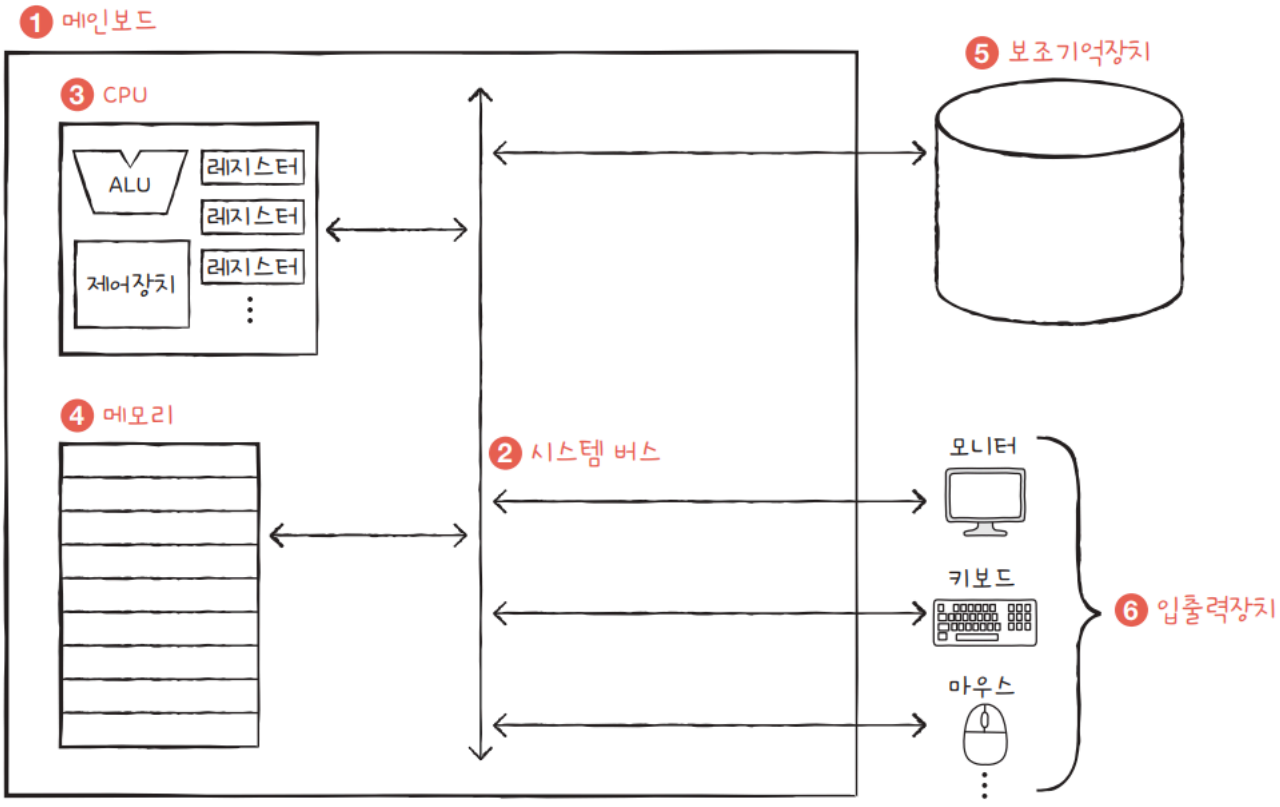
1. 컴퓨터 구조 시작하기

컴퓨터 구조를 알아야 하는 이유

컴퓨터 구조를 아는 것은 개발하는 데 있어 많은 도움이 된다. 컴퓨터 구조를 이해하고 있다면 문제 상황을 빠르게 진단하고, 이를 해결하기 위한 실마리를 다양하게 찾을 수 있다. 또, 단순하게 코드를 작성하는 것에서 그치지 않고, 성능, 용량, 비용까지 고려할 수 있는 능력을 갖출 수 있다.

컴퓨터의 구성 요소

현대적 컴퓨터는 매우 복잡한 부품들로 구성되나, 단순하게 **메모리**(Memory), **입출력 장치**(I/O Device; Input and Output Device), **중앙처리장치**(CPU; Central Processing Unit)로 구분할 수 있다. 그리고 이런 부품들은 **메인보드**(Main Board)에 장착되어 **버스**(Bus)를 통해 서로 연결되어 있다. 이를 그림으로 나타내면 아래와 같다.



컴퓨터의 구성요소

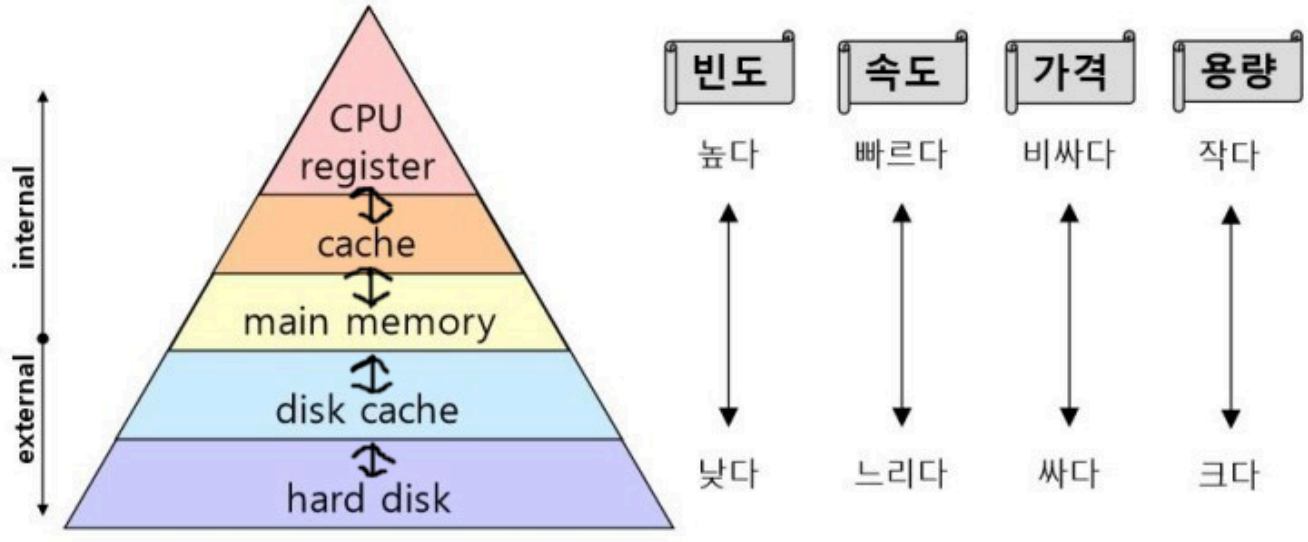
하나씩 살펴보도록 하자.

메모리

메모리는 정보를 저장하는 장치다. 메모리는 여러 주택이 있는 주상복합단지에 비유할 수 있는데, 각 집마다 **주소**(Address)가 있는 것처럼 메모리를 사용하려면 꼭 주소를 알아야 한다.

메모리는 주상복합단지와 비슷하다.

메모리는 여러 가지 종류가 존재하며, 각각의 쓰임새가 있다. 이를 표현한 계층도를 **메모리 계층 구조** (Memory Hierarchy)라 한다. 아래 그림에서 위에 위치할 수록 접근 빈도 수가 높으며 속도도 빠르지만, 가격이 비싸고 용량이 매우 작다.



메모리 계층 구조

각 메모리는 윗 계층의 메모리로부터 데이터를 받아 저장하거나(쓰기 요청), 데이터를 건네준다(읽기 요청). 읽기 요청을 받았을 때, 그 요청을 처리할 수 있는 데이터가 존재하지 않을 경우 아래 계층의 메모리로부터 복사하며, 쓰기 요청을 받았을 때는 윗 계층으로부터 전달 받은 정보로 갱신한다. 위에서부터 차례대로 가볍게 살펴보자.

레지스터

레지스터(Register)는 CPU 안에 존재하는 메모리로, 프로그램을 실행하는 데 필요한 값을 임시로 저장할 수 있다. 레지스터는 여러 개가 존재하며, 각기 다른 이름과 용도를 갖고 있다.

캐시 메모리

캐시(Cache) 메모리는 주기억장치나 보조기억장치에 접근하는 시간을 줄이기 위한 임시 저장장치다.

주기억장치

주기억장치(Main Memory)는 현재 실행하고 있는 프로그램에 대한 정보가 저장되는 부품이다. 여기에는 **RAM**(Random Access Memory)과 **ROM**(Read Only Memory)이 있는데, 보통 RAM을 일컫는다.*

* ROM에는 컴퓨터가 부팅(Bootstrap)되기 위한 프로그램이 저장된다.

보조기억장치

주기억장치의 단점은 저장 용량이 작고, 전원이 꺼지면 저장된 내용을 잃는다는 것이다.* 그래서 이런 단점을

보완하기 위한 메모리가 **보조기억장치**(Secondary Memory)다. 하드 디스크, SSD, USB 메모리, DVD, CD-ROM 등이 여기에 속한다.

* 휘발성(Volatility)이라 한다.

CPU

프로그램을 실행한다는 것을 조금 더 자세하게 얘기하자면 프로그램에 저장된 여러 명령어를 실행한다는 것이다. CPU는 프로그램에 저장된 명령어를 읽어 들여, 이를 해석한 후 실행하는 부품이다. CPU의 내부도 컴퓨터처럼 매우 복잡하나, **산술논리장치**(ALU; Arithmetic Logic Unit), **레지스터**, **제어 장치**(CU; Control Unit)가 핵심 요소이다. 레지스터는 위에서 살펴봤기 때문에 ALU와 제어 장치만 살펴보도록 하자.

ALU

CPU의 부품 중 가장 핵심이 되는 부품으로 연산을 담당한다.

제어 장치

컴퓨터의 내부 부품들은 멋대로 동작하지 않는다. 복잡한 교차로에서 경찰이 도로의 질서를 정리하는 것처럼 컴퓨터 또한 이런 경찰의 역할을 하는 부품이 필요하다. 이런 신호를 **제어 신호**(Control Signal)라고 하며, 제어 장치는 제어 신호를 내보내고, 명령어를 해석하여 ALU에게 어떤 연산을 해야하는지 알려준다.

입출력장치

컴퓨터 외부에 연결 돼 컴퓨터 내부와 정보를 교환하는 장치다.* 이때, 사용자가 컴퓨터로 보내는 정보를 입력이라고 하며, 컴퓨터가 사용자에게 보내는 정보를 출력이라고 한다. 입출력장치에는 마이크, 스피커, 프린터, 마우스, 키보드 등이 있다.

* USB 메모리를 생각해보면 컴퓨터 내부에 존재하는 것이 아니라 외부에 연결된다고 볼 수 있다. 즉, 보조기억장치 또한 컴퓨터 외부에 연결된 장치로 볼 수 있다는 것인데, 그래서 보조기억장치와 입출력장치를 묶어 주변장치(Peripheral Device)라고도 한다. 다만, 보통은 구분하여 얘기한다.

메인보드와 버스

컴퓨터의 핵심 부품은 모두 **메인보드**(Main Board)*라는 판에 연결된다. 메인보드에는 여러 부품을 부착할 수 있는 슬롯과 연결 단자가 존재한다. 메인보드에 부착된 부품은 상술했듯 버스를 통해 서로 정보를 주고받는다. 버스에도 여러 가지 종류가 있으며, 이중 **시스템 버스**(System Bus)가 컴퓨터의 핵심 부품을 연결한

다.

* 마더보드(Mother Board)라고도 한다.

시스템 버스에는 **주소 버스**(Address Bus), **데이터 버스**(Data Bus), **제어 버스**(Control Bus) 3가지로 구성되어 있다. 주소 버스는 메모리에 접근하기 위한 주소를 주고받는 통로이고, 데이터 버스는 메모리에 저장된 여러 정보를 주고받는 통로이며, 제어 버스는 제어 신호를 주고받는 통로이다.

2. 데이터

비트

메모리에 저장된 정보는 **명령어**와 명령어에 사용되는 **데이터**로 구분된다. 이러한 정보는 **비트**(Bit)로 저장되는데, 비트는 Binary와 Digit의 합성어로 2가지 상태를 나타낼 수 있는 숫자다. 비트에 저장할 수 있는 데이터는 무엇이든지 상관 없다. 그냥 수로서 0과 1이 될 수도 있고, 화재가 발생했다 / 발생하지 않았다, 왼쪽 / 오른쪽, 존재한다 / 존재하지 않는다 등 추상적인 의미도 저장할 수 있다.

비트에 저장할 수 있는 정보는 2가지 밖에 존재하지 않기 때문에 실제로는 비트를 여러 개 묶어 사용한다. 보통 **바이트**(Byte)라는 단위를 사용하며, 이는 8개의 비트를 의미한다.* 간혹, **워드**(Word)라는 단위를 사용하기도 하는데, 이는 CPU가 한번에 처리할 수 있는 데이터 크기를 의미한다. CPU를 구매할 때나 프로그램을 설치할 때 32bit, 64bit 등의 용어를 확인할 수 있는데, 이것이 CPU가 한번에 처리할 수 있는 데이터 크기**라고 할 수 있다. 워드의 절반을 **하프 워드**(Half Word), 워드의 1배를 **풀 워드**(Full Word), 워드의 2배를 **더블 워드**(Double Word)라고 한다.

* 하지만 바이트도 사용자의 입장에서는 무척 작기 때문에 실제로는 KB, MB, GB, TB나 KiB, MiB, GiB, TiB 라는 단위를 사용한다.

** 버스의 대역폭(Bandwidth)이라고 한다.

숫자를 표현하는 법

우리가 사용하는 수 체계는 10진법이다. 각 자리수를 0에서 9까지 사용하고, 자리수가 올라갈수록 10배씩 커지게 된다. 하지만 비트는 0과 1만 사용하기 때문에 비트로 수를 표현하려면 2진법을 사용해야 한다. 2진수의 표기법은 (1) 아래첨자 (2)를 붙이거나, (2) `0b`를 2진수 앞에 붙인다. 10진수를 2진수로 표현하기 위한 단계는 아래와 같다.

1. 임의의 수 N을 선택하고, N보다 작은 2의 제곱수를 나열한다.

> 13 --> 8, 4, 2, 1

2. 2의 제곱수 중 모두 더했을 때 N과 같아지도록 선택한다.

> **8**, **4**, 2, **1**

3. 선택한 수는 1로, 선택하지 않은 수는 0으로 표기한다.

> **8**, **4**, 2, **1** --> 0b1101

그러면 음수는 어떻게 나타낼까? 2진법으로 음수를 표기할 때는 2의 보수법을 사용한다. 2의 보수법으로 음수를 표현하기 위한 단계는 아래와 같다.

1. 0을 1로, 1을 0으로 표현한다. 이를 NOT 연산이라고 한다.

> 0b1011 --> 0b0100

2. 그 값에 1을 더해준다.

> 0b0100 --> 0b0101

음수까지 해결된 것으로 보이지만, 사실 한 가지 더 문제는 있다. `0b0101`이라는 2진수가 있을 때, 이는 `0b1011`의 음수일까, 아니면 `3`일까? 컴퓨터는 이를 구분하기 위해서 **플래그 레지스터**(Flag Register)를 사용하고 있다.

한편, 2진수는 너무 길기 때문에 2진수를 그대로 다루기 보단 이를 16진수로 변환하여 사용하는 것이 일반적이다. 2진수와 마찬가지로 표기할 때는 (1) 아래첨자 $_{(16)}$을 붙이거나, (2) `0x`를 16진수 앞에 붙인다. 2진수를 16진수로 변환하고 싶다면 아래의 단계를 따르면 된다.

1. 2진수를 4자리씩 나눈다.

> 0b1011011 --> 101 | 1011

2. 나눠진 구역을 16진수로 변환한다. 16진수의 각 자릿수 범위는 10 미만은 0 ~ 9를 사용하되 10 이상부터는 A ~ F로 표기한다.

> 101 | 1011 --> 5 | B --> 0x5B

문자를 표현하는 법

앞서 비트로 숫자를 표현하는 법에 대해서 살펴봤다. 그럼 문자는 어떻게 표현할 수 있을까? 문자를 표현하기 위해선 문자 집합, 인코딩, 폰트에 대해서 알아야 한다. **문자 집합**(Character Set)은 컴퓨터가 인식하고 표현할 수 있는 문자의 모음*으로 각 문자에는 이진수**가 할당되어 있다. 그래서 문자를 이진수로 변환하는 과정을 **문자 인코딩**(Character Encoding), 반대로 이진수를 문자로 변환하는 과정을 **문자 디코딩**(Character Decoding)이라 한다. 그리고 문자의 모양을 정의한 것이 **폰트**(Font)라고 할 수 있다. 대표적인 문자 집합에 대해서 살펴보자.

* 코드 페이지(Code Page)라고도 한다.

** 코드 포인트(Code Point)라고 한다.

아스키 코드

아스키(ASCII; American Standard Code for Information Interchange) 코드는 알파벳과, 아라비아 숫자, 일부 특수 문자를 포함하는 문자 집합이다. 7비트를 사용해 총 128개의 문자를 표현한다.

EUC-KR

아스키 코드는 인코딩이 매우 간단하지만, 영어밖에 표현할 수 없다. 따라서 각 나라에서는 그들의 언어를 표현하기 위한 인코딩이 필요했다. 한국도 마찬가지였다. 한글은 각 음절이 초성, 중성, 종성의 조합으로 구성되는데, 이를 표현하는 방법에는 완성된 하나의 글자에 코드 포인트를 부여하는 완성형과 초성, 중성, 종성을 위한 각각의 비트열을 할당하여 하나의 글자를 만드는 조합형이 있다. **EUC-KR**은 KS X 1001, KS X 1003의 문자 집합을 기반으로 하는 완성형 인코딩으로, 2바이트를 사용하여 2,350개 정도의 한글 단어를 표현할 수 있었다. 하지만, 이 역시도 11,172자나 되는 모든 한글 조합을 표현하기엔 부족했기에 때때로 문제가 발생하기도 했다.*

* 실제 청원이 올라온 적이 있다. 마이크로소프트도 이런 문제점을 알고 있었기에 남은 8,000여자를 지원하는 EUC-KR의 확장형 격인 CP949 인코딩을 내놓았다.

유니코드

각 나라마다 다른 인코딩 방법을 사용하는 것은 국제화에 장애물이었다. 따라서 대부분의 언어를 아우르는 문자 집합과 통일된 표준 인코딩 방식이 필요했고 이것이 **유니코드**(Unicode)이다. 유니코드 문자에는 각 나라의 문자 외에도 이모티콘이나 여러 특수 기호도 포함되어 있다. 인코딩 방법에는 UTF-8, UTF-16, UTF-32 등이 있는데, 이중 UTF-8을 가장 많이 사용하고 있다.

참고자료

- https://www.geeksforgeeks.org/introduction-of-secondary-memory/
- https://superuser.com/questions/357530/how-are-character-encodings-related-to-fonts