Computer Architecture

컴퓨터개론

(Introduction to Computer Systems)

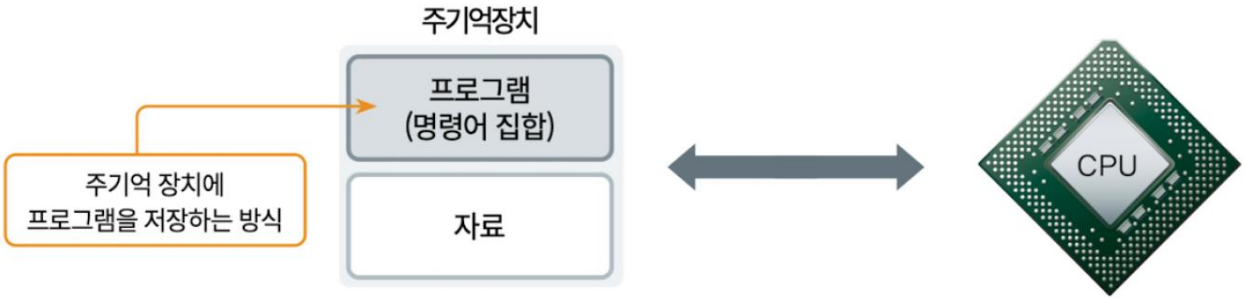
GEN1030

프로그램 내장 방식

프로그램 내장 방식

- 프로그램 내장(Stored program) 방식
  - 폰 노이만(John von Neumann) 방식
  - 메모리에 자료와 프로그램이 함께 저장
  - 주기억장치에 프로그램만 바꾸어 저장하면 프로그램 실행 가능 (cf. ENIAC 컴퓨터)

- Central Processing Unit (CPU)
  - 메모리에서 필요한 자료를 이용
  - 저장된 **명령어**를 순차적(Sequential) 실행

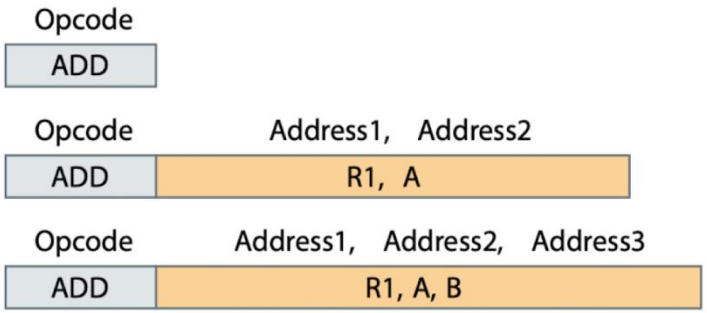


명령어 형식

- 명령어(instruction) 구성
  - 연산 부분(operation part): 명령어가 수행해야 할 기능을 의미
  - 피연산 부분(operand part): 연산에 참여하는 자료

- 명령어 크기
  - 8, 16, 32, 64 bit 등으로 구성 가능
  - Example) 16 bit
    - 4 bit: Opcode (연산 종류)
    - 12 bit: Address (피연산자의 메모리 주소)

- 피연산자
  - 메모리 주소 혹은 레지스터(CPU 내부에 있는 고속 메모리) 번호



명령어 종류

- 연산자
  - ADD (add)
  - LDA (load address)
  - STA (store address)
  - HLT (halt)

명령어	구문 형식	기능
ADD	ADD A	피연산자의 자료 A와 레지스터의 자료를 더하는 명령어
LDA	LDA B	피연산자의 자료 B(주소 B의 자료)를 레지스터에 가져오는 명령어
STA	STA C	레지스터의 내용을 피연산자 C(주소 C)에 저장하는 명령어
HLT	HLT	컴퓨터를 종료하는 명령어

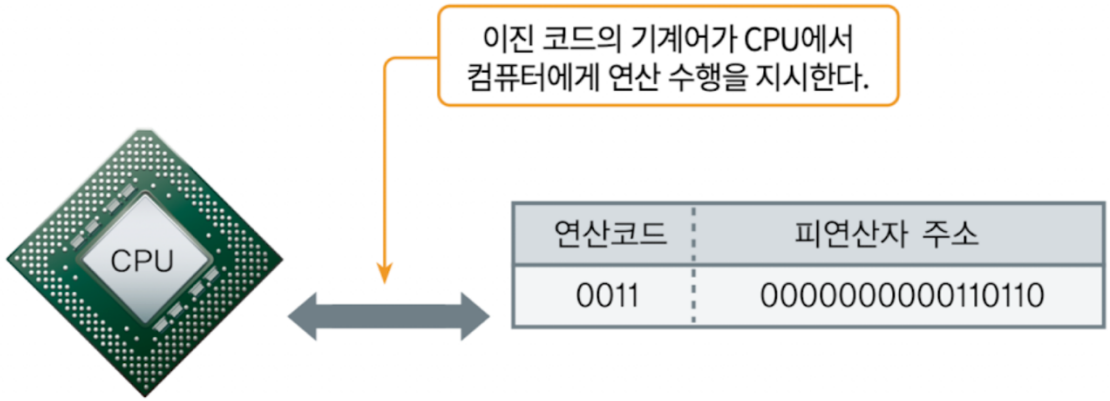
Low Level Language

- Low Level Language
  - 컴퓨터에게 작업을 지시할 수 있는 언어
  - 사람에게 친숙하지 않아 이해하기 어려움

- Machine language (기계어)
- Assembly language (어셈블리어)

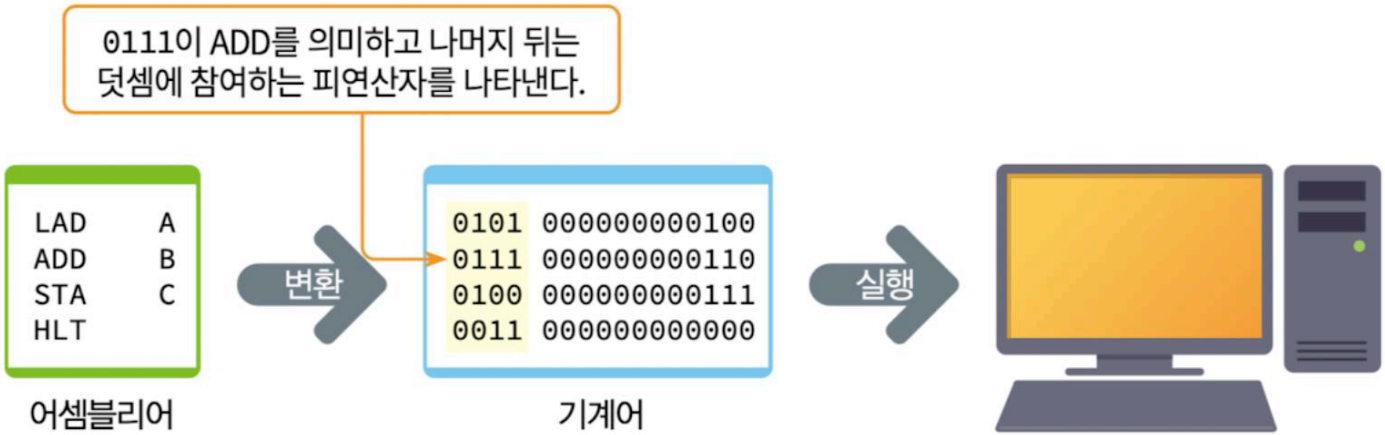
Machine Language

- 컴퓨터를 작동시키기 위해 Binary (0 or 1)로 표현한 컴퓨터 고유 명령 형식 언어
- 연산코드
  - 연산의 종류
- 피연산자
  - 처리 대상인 자료
  - 주로 메모리에 저장된 자료의 주소 나타냄



Assembly Language

- Machine language를 사람이 사용하는 자연어와 유사하게 만든 것
- 연산자와 피연산자를 몇 개의 문자 조합으로 기호화해서 나타냄



Low Level Language

- Machine language와 Assembly language는 각각의 CPU 종류에 따라 다름
  - 즉, CPU 아키텍쳐에 의존적
  - 특정 CPU에서 사용되기 위해 Machine language 혹은 Assembly language로 작성한 프로그램은 다른 종류의 CPU에서 동작하지 않을 수 있음
  - ex) Intel 프로세서 <-/-> Apple Silicon 프로세서

ADD eax, 3 **ADD r0, r0, #3**

- 처리 속도가 중요하거나 High level language에서 지원되지 않는 기능을 사용해야 하는 경우 Assembly language 사용하는 경우 있음

기억장치

주기억장치

- Main memory
- 컴퓨터가 작동하는 동안 CPU가 처리 할 프로그램 명령어와 프로그램에서 이용할 자료를 저장하고 있음
  - 실행 중인 프로그램이 올라감

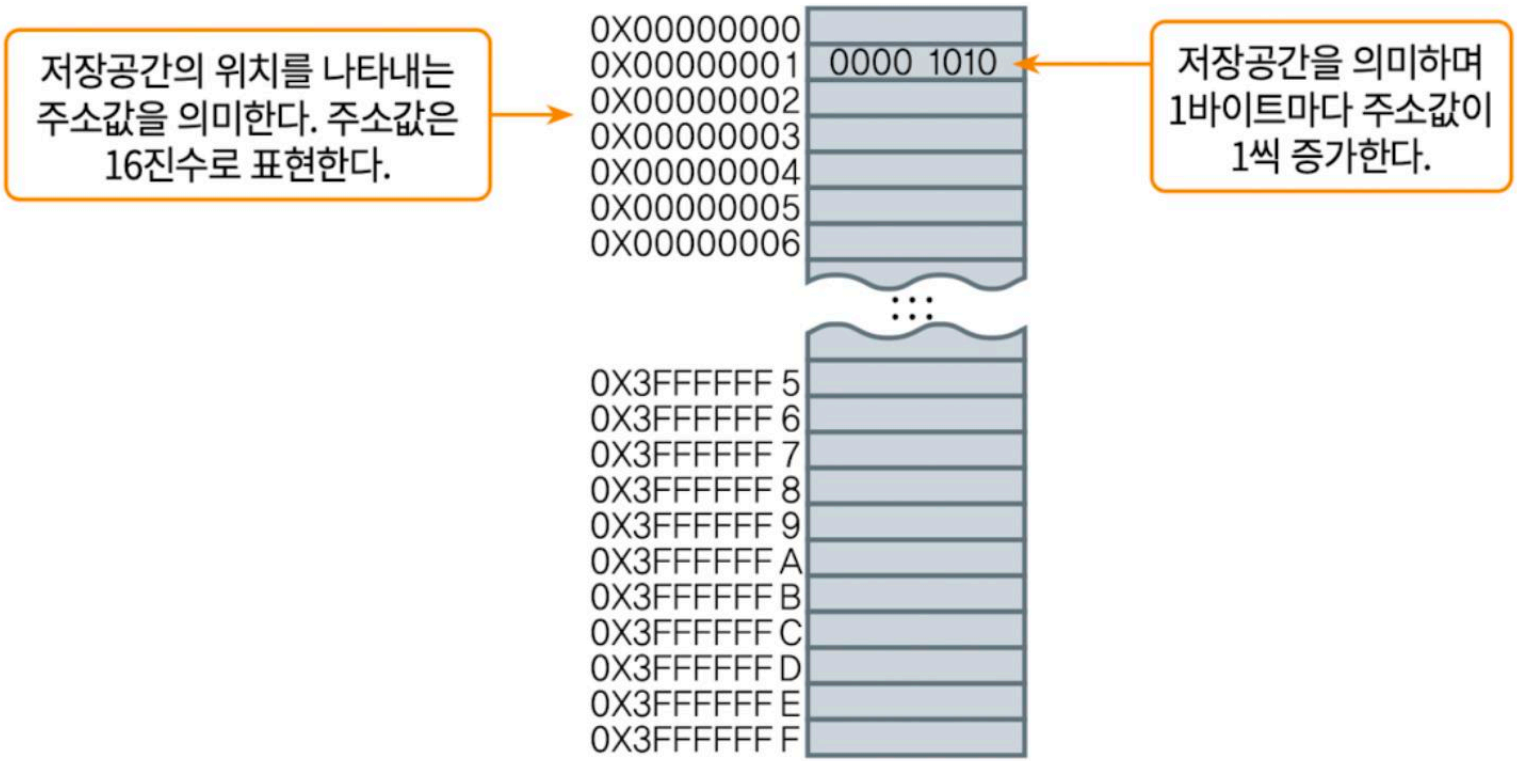
주기억장치 – 주소

- 메모리 주소(Address)
  - 메모리는 주소를 이용해 byte 단위로 고유하게 식별
  - 예를 들어,
    - 메모리 용량이 1,024MB? 주소는 0 ~ (1024×2^{20} -1 까지 존재 ($1M = 2^{20}$)

- 워드(Word)
  - 컴퓨터가 한 번에 작업할 수 있는 데이터의 단위
  - CPU 아키텍쳐 따라 다름 – 32 bit (4 byte) or 64 bit (8 byte)
  - 자료는 주로 워드 단위로 기억장치와 CPU 사이를 이동

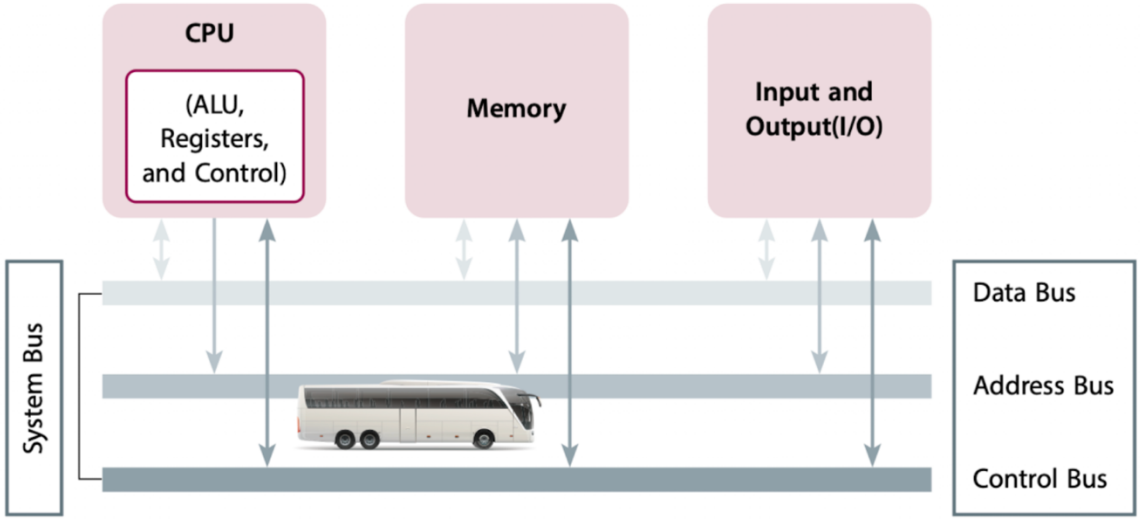
주기억장치 – 주소

- 메모리 주소 예시



주기억장치 – 버스

- 시스템 버스(System Bus)
  - CPU, 메모리, 입출력 장치의 제어기(I/O Controller) 사이를 연결해주는 통로
- 보통 세 종류의 버스로 구성
  - 주소 버스(Address Bus): 주기억장치의 주소를 전달
  - 자료 버스(Data Bus): 실제 데이터를 전달
  - 제어 버스(Control Bus): 읽기, 쓰기 같은 명령 제어 신호(무슨 동작을 할지)를 전달



RAM

- Random Access Memory (RAM)
  - 휘발성(Volatile) 메모리
  - 메모리 주소를 사용하여 메모리의 특정 위치의 내용을 참조
  - 쓰기와 읽기 모두 가능
  - RAM 종류 – SRAM, DRAM

RAM

- Dynamic RAM (DRAM)
  - 전원이 연결된 상태에서 일정 주기마다 전기적으로 재충전(Refresh) 해주어야 데이터가 지워지지 않음
  - Synchronous DRAM (SDRAM): 주로 주기억장치로 사용

- Static RAM (SRAM)
  - 전원만 연결되어 있으면 데이터가 지워지지 않음
  - DRAM보다 빠름
  - CPU 내부의 캐시 메모리(Cache memory)에 주로 사용

ROM

- Read Only Memory (ROM)
  - **읽기 전용** 메모리
    - 보통 데이터가 한 번 저장 된 이후에는 다시 쓸 수 없음
  - 메모리에 임의 접근 가능
  - 전원이 공급되지 않아도 정보가 지워지지 않음
    - **비휘발성(non-volatile)**
  - 컴퓨터가 전원 공급받아 처음 수행해야 할 **바이오스(BIOS)** 저장에 사용
  - 주기억장치의 일종으로 봄

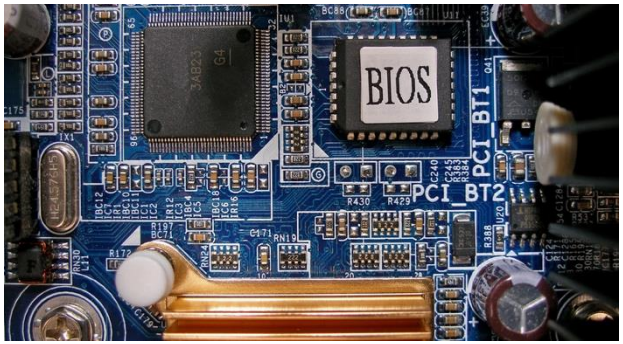
ROM 종류

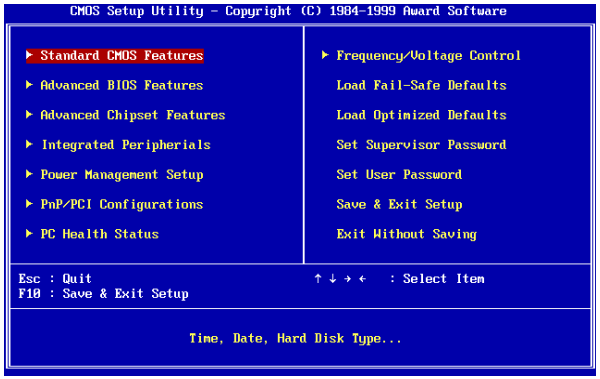
- Mask ROM
  - 기억된 데이터를 지우거나 변경하는 것이 불가능
  - 보통 제조 공정에서 이미 프로그램이 들어감
- Programmable ROM (PROM)
  - 임의의 프로그램을 쓸 수 있음
  - 한 번만 기록 가능
- Erasable Programmable ROM (EPROM)
  - 여러 번 지우고 다시 쓸 수 있음
  - 별도의 방법(자외선 또는 X선 등)을 이용해 지움

BIOS

- Basic Input/Output System (BIOS)
  - 컴퓨터의 부팅을 위한 펌웨어(firmware)
  - 컴퓨터 전원이 들어오면 바이오스는 먼저 POST 실행
- POST (Power-On Self Test)
  - 일련의 시동자체시험 과정
  - 주기억장치와 보조기억장치, 컴퓨터 키보드, 그래픽 카드 등, 주변장치 연결 확인
  - 보조기억장치의 연결에 문제가 있거나 보조기억장치 내의 운영체제 파일에 문제가 있다면 → 부팅 불가 메시지 출력
  - 문제가 없으면 → OS를 메모리에 로드(load)

BIOS가 저장된 ROM

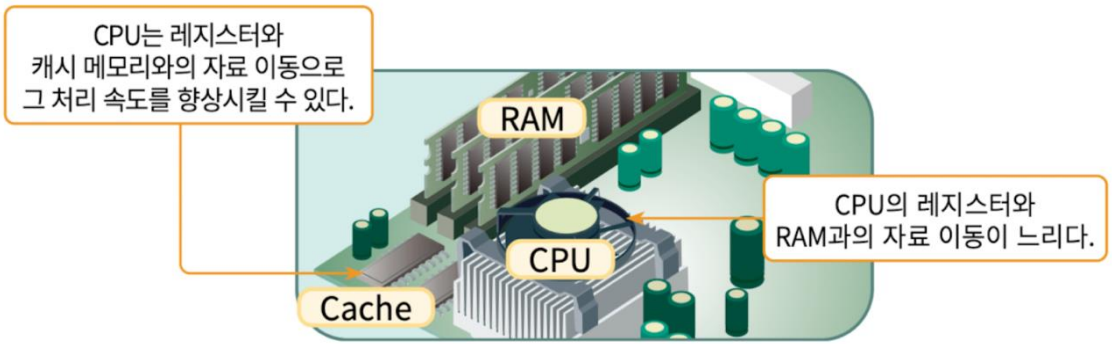




캐시(Cache) 메모리

- 캐시의 사용 이유
  - CPU에 비해 주변기기의 속도는 상대적으로 느림
  - 주기억장치와 CPU 사이의 데이터 이동이 오래 걸림
  - CPU만 빠르고 다른 기억장치의 속도가 느리다면 빠른 CPU의 장점이 없음

- 캐시(Cache)
  - 캐시는 메인메모리보다 10배 이상 빠름
  - 캐시는 주기억장치와 CPU 간 처리속도 차이로 인해 발생하는 **병목(bottleneck) 현상**을 해결
  - 메모리 소자로 저장 속도가 빠르고 고가인 Static RAM (SRAM) 사용



캐시(Cache) 메모리

- 참조 지역성(Locality of reference)
  - 프로그램이 메모리에 접근할 때 일정한 패턴이 있음
  - 기억장치에 저장된 자료는 **시간적**으로나 **공간적**으로 곧 다시 사용할 가능성이 높음
- 시간 지역성(Temporal locality)
  - 최근에 참조 된 자료를 곧 다시 사용할 가능성이 높음
- 공간 지역성(Spatial locality)
  - 한 번 참조된 자료의 주변 자료가 참조 될 가능성이 높음

캐시(Cache) 메모리

- 참조 지역성(Locality of reference)
  - 프로그램이 메모리에 접근할 때 일정한 패턴이 있음
  - 기억장치에 저장된 자료는 **시간적**으로나 **공간적**으로 곧 다시 사용할 가능성이 높음

- **시간 지역성(Temporal locality)**
  - 최근에 참조 된 자료를 곧 다시 사용할 가능성이 높음

- 공간 지역성(Spatial locality)
  - 한 번 참조된 자료의 주변 자료가 참조 될 가능성이 높음

```
sum = 0;
A = 2;
for (i = 0; i < 100; i++)
   sum += A;
```

i = 0 → sum = sum + A
= 0 + 2 = 2
i = 1 → sum = sum + A
= 2 + 2 = 4
⋮
i = 99

캐시(Cache) 메모리

- 참조 지역성(Locality of reference)
  - 프로그램이 메모리에 접근할 때 일정한 패턴이 있음
  - 기억장치에 저장된 자료는 **시간적**으로나 **공간적**으로 곧 다시 사용할 가능성이 높음

i	0	1	2	...	99
array[i]	1	2	3	...	100

- 시간 지역성(Temporal locality)
  - 최근에 참조 된 자료를 곧 다시 사용할 가능성이 높음

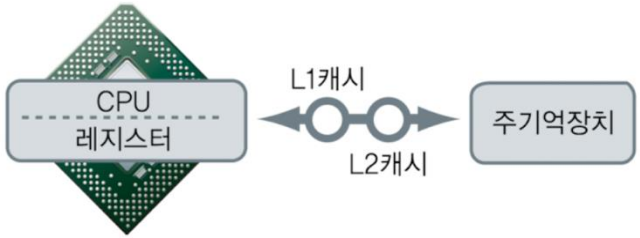
```
array[100] = {1, 2, 3, ...};
sum = 0;
for (i = 0; i < 100; i++)
   sum += array[i];
```

- **공간 지역성(Spatial locality)**
  - 한 번 참조된 자료의 주변 자료가 참조 될 가능성이 높음

i = 0 → sum = sum + array[0]
= 0 + 1 = 1
i = 1 → sum = sum + array[1]
= 1 + 2 = 3
⋮

캐시 계층 구조

- 캐시 레벨(Level)



- 속도, 크기, 비용의 trade-off
  - 캐시 크게 만들면 → 비싸고, 느려짐 (L1의 경우 CPU 면적도 증가)
    *메모리는 크기가 커질수록 접근 시간이 늘어남
  - 캐시 작게 만들면 → 캐시에 원하는 데이터가 없을 확률 증가 (Miss 많아짐)

- 레벨별로 캐시 사이즈 다르게
  - Level 1 (L1) cache: 작음 → 매우 빠름
  - Level 2 (L2) cache: 조금 더 큼 → L1보다 조금 느림

캐시 메모리

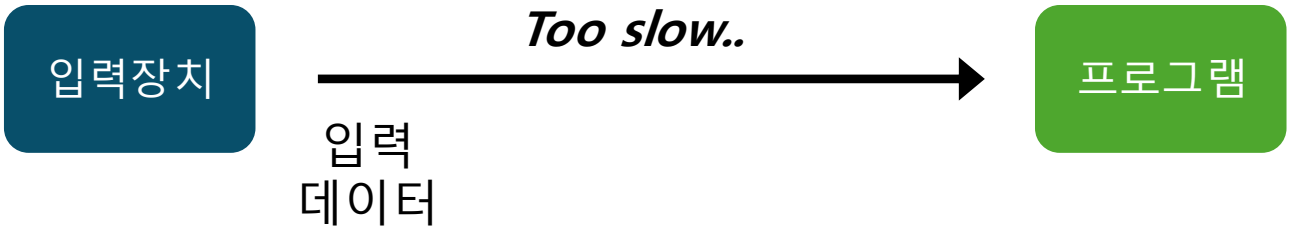
- 디스크(Disk) 캐시
  - 하드디스크(HDD) 속도는 RAM보다 느림
  - 매번 프로그램을 실행할 때마다 디스크를 읽어야 함

  - 보조기억장치와 주기억장치 사이의 캐시
  - 처음 프로그램을 실행할 때 RAM으로 올라가는 정보를 캐시에 저장
  - 보통 주기억장치의 일부를 디스크 캐시로 사용

버퍼(Buffer)

- 데이터를 잠시 저장하는 임시 메모리 공간
- 속도의 차이가 나는 두 장치 사이에서 데이터 흐름을 맞추기 위해 사용
  - CPU의 속도는 입출력 장치의 속도보다 훨씬 빠름
    - 입출력? 키보드 입력, 네트워크 통신 등

  → 주변장치의 응답을 기다리며 시간을 허비할 수 있음



버퍼(Buffer)

- 데이터를 잠시 저장하는 임시 메모리 공간
- 속도의 차이가 나는 두 장치 사이에서 데이터 흐름을 맞추기 위해 사용
  - CPU의 속도는 입출력 장치의 속도보다 훨씬 빠름
    - 입출력? 키보드 입력, 네트워크 통신 등

  → 주변장치의 응답을 기다리며 시간을 허비할 수 있음

- RAM의 일부 공간을 버퍼로 사용
  - 입력(input)/출력(output)장치로 전달 될 데이터를 임시로 저장
  - 그 동안 CPU는 다른 작업을 처리 할 수 있음

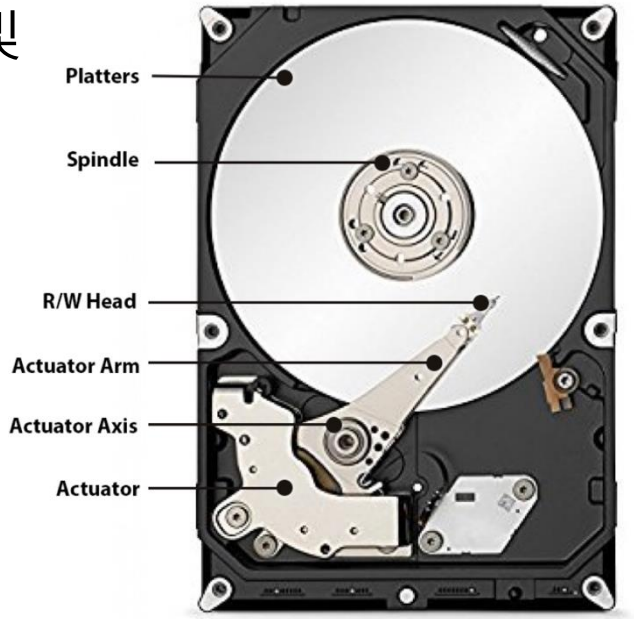
보조기억장치

- 전원이 공급되지 않아도 대용량의 자료를 계속해서 저장 가능
  - 주기억장치의 제한된 용량을 보조하기 위해 사용

- 접근 방식
  - 순차접근(Sequential access)
    - 순차적으로 데이터에 접근 가능
    - 자기 테이프
  - 직접접근(Direct access)
    - 원하는 위치에 바로 쓰고 읽을 수 있는 직접 접근이 가능
    - 자기 디스크, 자기 드럼

보조기억장치

- Hard Disk Drive (HDD)
  - 대표적인 보조기억장치 중 하나
  - 자성 물질의 디스크(Disk)를 회전시키고 그 위에 헤드(Head)를 접근 시켜 데이터를 읽거나 씀
  - 기계적인 방식이므로 소음, 진동 및 속도 제한 있음

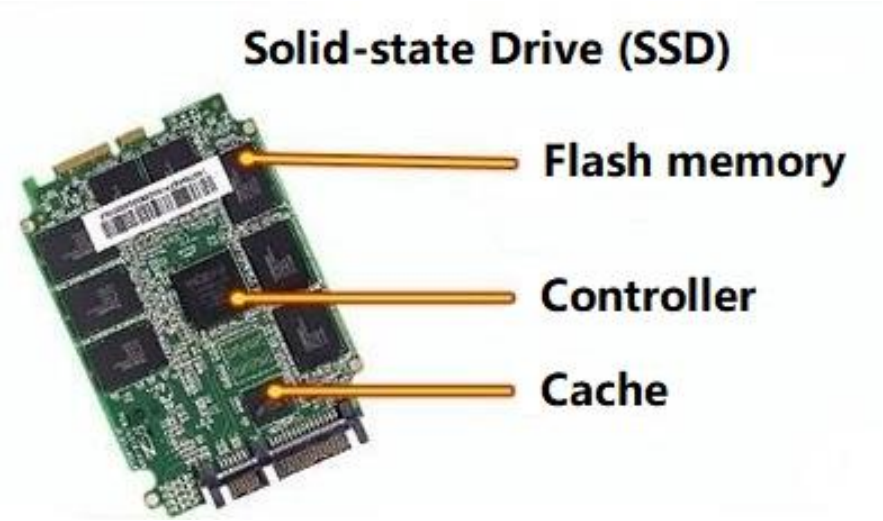


Why "Hard" Disk?

- 얇고 부드러운 재질의 저장장치인 플로피 디스크 (Floppy Disk)와 대비

보조기억장치

- Solid State Disk (SSD)
  - 플래시 메모리(Flash memory)와 이를 제어하는 컨트롤러로 구성
  - HDD와 비교했을 때
    - 읽고 쓰는 속도가 빠름
    - 전력 사용량 적음
    - 충격에 강함
    - 발열 및 소음 적음



보조기억장치

- 플래시 메모리(Flash memory)
  - RAM과 ROM의 장점을 합침
  - 정보의 입출력이 자유로움
  - 비휘발성(non-volatile)
  - 페이지(Page)/블록(Block) 단위로 데이터 접근
    - cf) RAM은 byte 단위로
  - 다양한 곳에 사용 됨
    - 휴대전화, 게임기
    - USB 메모리

SD Memory

- USB 메모리
  - Universal Serial Bus (USB): 컴퓨터와 주변장치 연결하기 위한 인터페이스 표준
    - 여러 장치를 하나의 표준으로 연결
  - 플래시 메모리 사용
  - USB 포트(Port) 사용하여 외부 연결

기억장치의 계층 구조

- 다양한 종류의 저장장치를 사용하는 이유는?
  - 저장장치의 속도, 용량, 가격, 사용 목적 등을 고려
  - 저장장치의 속도가 빠르면 가격이 상승
    - 동일한 비용으로 속도를 유지하기 위해선 – 용량이 작아져야 함
- 계층에 따른 사용
  - 보조기억 장치
    - 장기로 이용하려는 프로그램, 데이터
  - 주기억장치
    - 현재 실행 중인 프로그램, 데이터
  - 캐시
    - 현재 집중적으로 이용되는 프로그램, 데이터