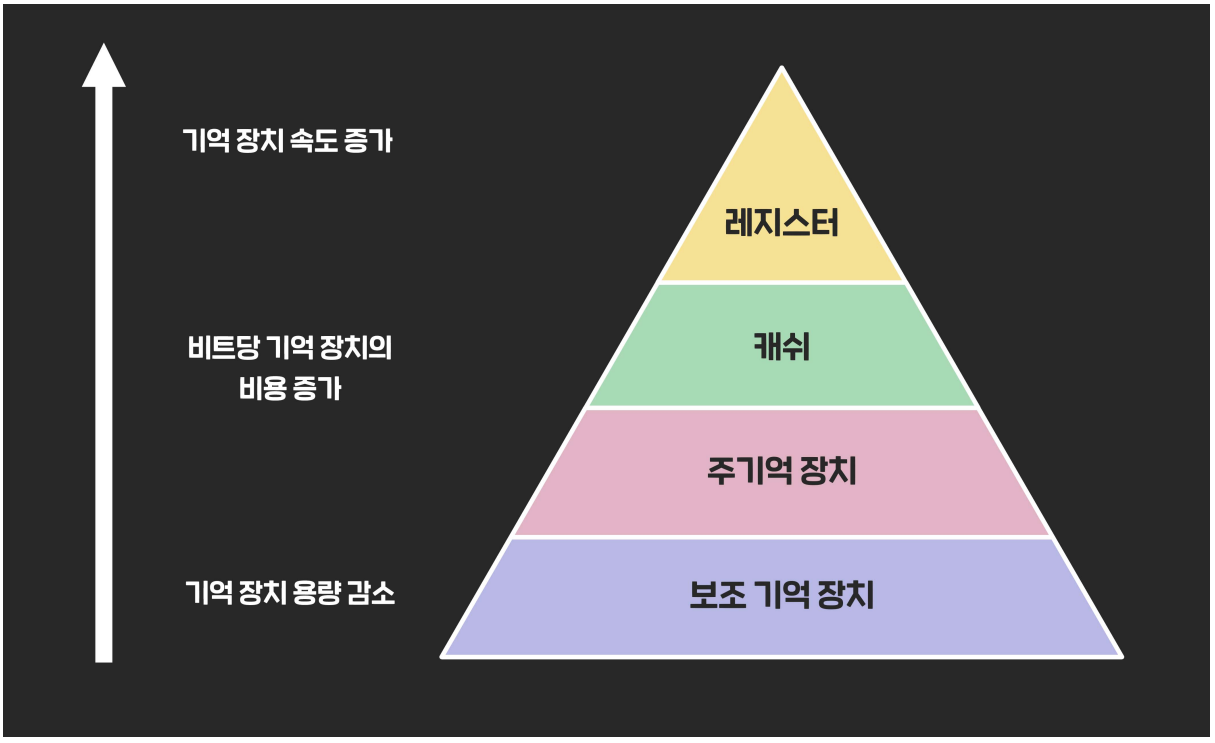
# Memory & Cache

| | |
|---|---|
| 태그 | OS |
| 주차 | 5주차 |

목차



## 0. 메모리 구조




메모리 계층 구조(Memory Hierarchy)

- 레지스터, 캐시, 주기억장치, 보조기억장치로 구성
  - 계층 위로 올라갈수록 가격↑, 용량↓, 속도↑
  - 계층의 존재 이유는 경제성과 캐시 때문

---

# 1. 레지스터

- CPU가 요청을 처리하는데 필요한 데이터를 일시적으로 저장하는 다목적 공간
  - CPU는 자체적으로 데이터를 저장할 수 없으므로 레지스터를 이용해 연산 처리 및 번지 지정
- 프로세스 내부에 있는 작은 공간으로 연산 제어, 디버깅 등의 목적으로 사용
- CPU 내부에 존재

---

# 2. 캐시

- CPU의 처리속도와 주기억장치의 접근 속도 차이를 줄이기 위해 사용하는 고속 Buffer Memory

- CPU가 주기억장치에서 저장된 데이터를 읽어 올 때, 자주 사용하는데이터를 캐시 메모리에 저장한 뒤 다음 이용 시 이곳에서 가져오면서 속도 향상
  - 계층 구조에서 가장 빠른 소자이며, 처리 속도가 CPU의 속도와 비슷함
  - 캐시 메모리 사용 시 주 기억장치를 접근하는 횟수가 줄어 컴퓨터의 처리 속도 향상
- CPU 내부에 존재

## 2-1. 작동 원리

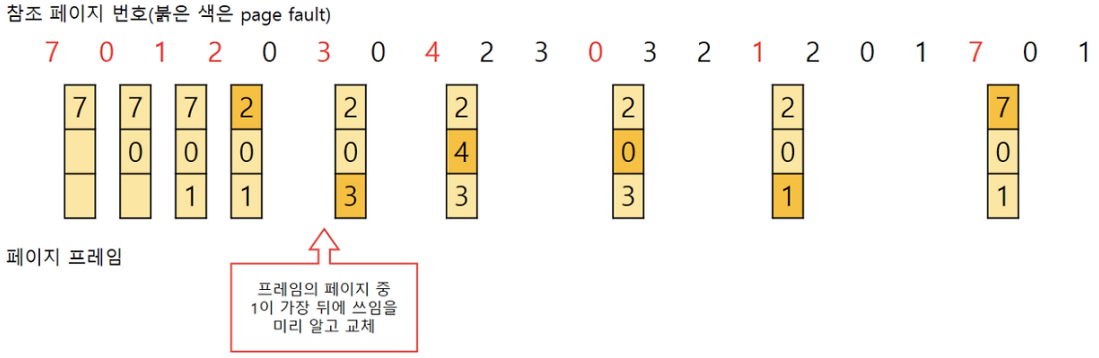
- 시간 지역성
  - 특정 데이터가 한번 참조된 경우, 가까운 미래에 또 한번 참조될 가능성이 높음
  - ex) `for`, `while`문에 사용되는 데이터는 참조 가능성이 높으므로 저장
- 공간 지역성
  - 액세스된 기억장소와 인접한 기억장소가 액세스될 가능성이 높음
  - ex) 배열에서 참조된 데이터 근처에 있는 데이터가 참조 가능성이 높으므로 저장

## 2-2. 캐시 미스(Cache Miss)

- **Cache Hit:** CPU가 요청한 데이터가 캐시에 있는 것
- **Cache Miss:** CPU가 요청한 데이터가 캐시에 없어 주기억장치에서 가져오는 것
  1. **Cold Miss**: 해당 메모리 주소를 처음 불러서 나는 Miss
  2. **Conflict Miss**: A와 B를 저장해야 하는데, A와 B가 같은 주소에 할당되어 있는 경우 나는 Miss
  3. **Capacity Miss**: 공간이 부족해서 나는 Miss

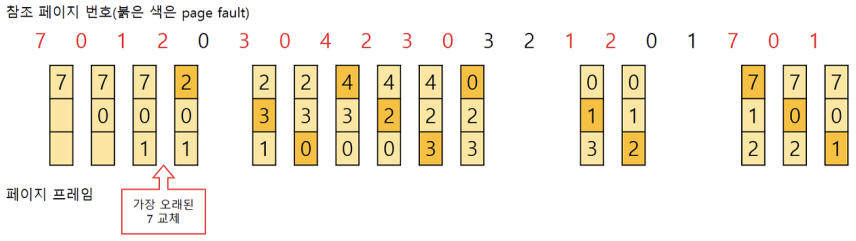
### 1) 페이지 교체 기법

- OPT(Optimal)



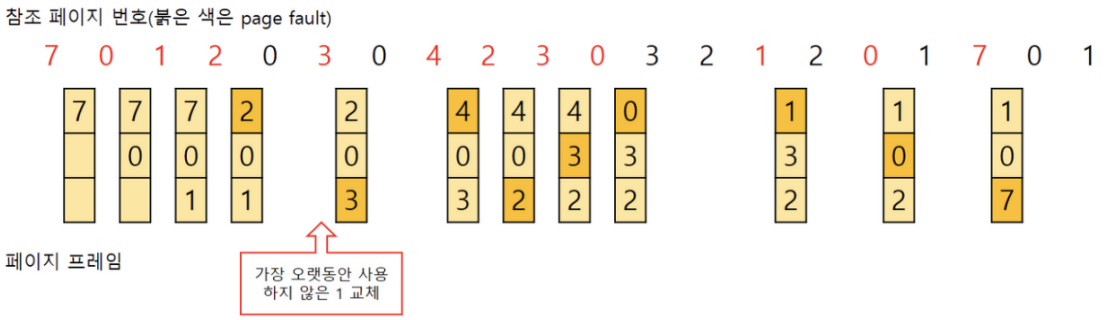

- 앞으로 가장 오랫동안 사용하지 않을 페이지 교체
- 가상 이상적
- 현실적으로 불가능 → 비교연구 목적을 위해 사용

- FIFO(First In First Out)




- 가장 먼저 올라온 페이지 교체
- 간단하고, 초기화 코드에 대해 적절함
- 들어온 시간을 저장하거나 올라온 순서를 큐에 저장
- 직관적으로는 프레임 수가 많아질 수록 페이지 결함의 횟수가 감소할 것 같으나, 실제로는 그렇지 않게 되는 현상이 발생할 수 있음
  - Belady's Anomaly(FIFO anomaly)

- LRU(Least Recently Used)




- 가장 오랫동안 사용되지 않은 페이지 교체
  - 시간 지역성 성질 고려
- 사용된 시간을 알 수 있는 부분을 저장해 가장 오랫동안 참조되지 않은 데이터 제거 (페이지마다 카운트 필요)
- 큐로 구현 가능. 사용한 데이터를 큐에서 제거해 맨 위로 다시 올리고, 프레임이 모자랄 경우 맨 아래에 있는 데이터 삭제
- 프로세스가 주기억장치에 접근할 때 마다 참조된 페이지 시간을 기록해야 하므로 막대한 오버헤드 발생
  - 카운터나 큐, 스택과 같은 별도의 하드웨어 필요
  - 카운터: 각 페이지 별로 존재하는 논리적인 시계(Logical clock)로, 해당 페이지가 사용될 때 마다 0으로 초기화 후 시간을 증가시켜 시간이 가장 오래된 페이지를 교체

- LFU(Least Frequently Used)
  - 참조 횟수가 가장 작은 페이지 교체
  - LRU는 직전 참조된 시점만을 반영하지만, LFU는 참조 횟수를 통해 장기적 시간 규모에서의 참조 성향을 고려할 수 있음
  - 가장 최근에 불러온 페이지가 교체될 수 있어 구현이 더 복잡하고 오버헤드가 큼
- MFU(Most Frequently Used)
  - 참조 횟수가 가장 많은 페이지 교체
  - 가장 많이 사용된 페이지는 앞으로는 사용되지 않을 것이라고 가정
- NRU(Not Recently Used) = NUR(Not Used Recently), 클럭 알고리즘
  - 최근에 사용되지 않은 페이지 교체 (LRU와 근사)
  - 교체되는 페이지의 참조 시점이 가장 오래되었다는 것을 보장하지는 못함

- 적은 오버헤드로 적절한 성능
- 동일 그룹 내에서 선택 무작위
- 각 페이지마다 두 개의 참조 비트(Reference Bit)와 변형 비트(Modified Bit, Dirty Bit)가 사용됨
  - 참조 비트: 페이지가 참조되었을 때 1, 아닐 때 0 (모든 참조비트를 주기적으로 0으로 변경)
  - 변형 비트: 페이지 내용이 변경되었을 때 1, 아닐 때 0
- 우선순위: 참조비트 > 변형비트

---

# 3. 주기억장치 (메모리)

- 컴퓨터 내부에서 현재 CPU가 처리하고 있는 내용을 저장하고 있는 기억 장치
- 즉, CPU의 명령에 의해 기억된 장소에 직접 접근하여 읽고 쓸 수 있음

## 3-1. 종류

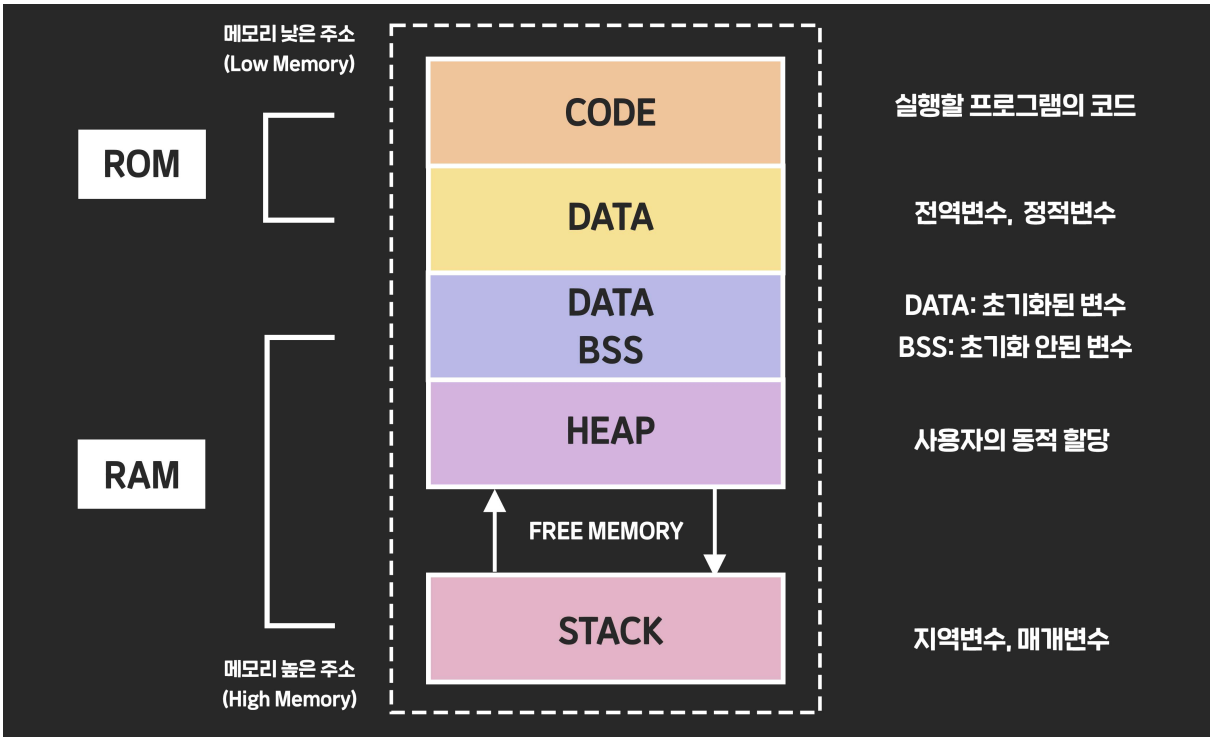
### 1) RAM(Random Access Memory)

- 읽고 쓰기가 가능하며, 응용 프로그램, OS 등을 불러와 CPU가 작업할 수 있도록 하는 기억장치
- Read와 Write 속도가 같고, 프로그램을 로딩하거나 데이터를 임시저장할 때 사용
- **휘발성 메모리(Volatile Memory)**
  - 전원이 끊어지면 데이터가 전부 지워짐
  - 따라서 실행중인 파일은 항상 보조기억장치에 저장해줘야 함

### 2) ROM(Read Only Memory)

- 데이터 저장 시 반영구적으로 사용할 수 있는 기억장치
- 저장된 데이터를 읽기만 가능
- 변경되면 안되는 BIOS와 같은 주요 데이터를 저장할 때 사용
- PROM, EPROM, EEPROM 등 수정 가능한 ROM도 존재
- **비휘발성 메모리(Non-Volatile Memory)**
  - 전원이 끊어져도 데이터가 사라지지 않음

# 3-2. 구조




## 1) 코드 영역(Code Section)

- 텍스트 영역(Text Section)이라고도 함
- 실행할 프로그램의 코드가 저장되는 영역
- CPU는 이 영역에 저장된 명령어를 하나씩 가져가서 처리

## 2) 데이터 영역(Data Section)

- 프로그램의 시작과 동시에 할당되며, 프로그램 종료시 소멸
- 전역변수, 정적변수, 배열, 구조체 등이 저장됨
- data Section
  - 초기화된 변수 저장 (초기화된 전역변수 및 지역변수)
- rodata Section
  - 상수 데이터 저장
- BSS Section
  - 초기화되지 않은 변수 저장 (초기화되지 않은 전역변수 및 정적 변수)

## 3) 힙 영역(Heap)

- 사용자가 직접 관리할 수 있는 메모리 영역
- 메모리 주소 값에 의해서만 참조되고 사용
- 참조 타입(Reference Type)에 대한 저장 공간
- 메모리 공간이 동적으로 할당되고 해제됨
    - 낮은 주소에서 높은 주소의 방향으로 할당

### 4) 스택 영역 (Stack)

- 함수 호출과 관계되는 지역변수, 매개변수, 메소드, return값 등이 저장되는 영역
- 프로그램이 자동으로 사용되는 임시 메모리 영역
- 라이프 사이클은 호출되는 함수와 동일
- 스택 크기(Stack size)는 각 프로세스마다 할당됨
    - 프로세스가 메모리에 로드될 때 사이즈가 고정돼 런타임 시 변경은 불가
- 힙과 같은 공간을 공유하지만 시작점이 다름
    - 높은 주소에서 낮은 주소의 방향으로 할당
- push 동작으로 데이터를 저장하고, pop 동작으로 데이터를 인출

> 💡 **스택 프레임(Stack Frame)**
>
> 스택 영역에 저장되는 함수의 호출 정보

> 💡 **Stack/Heap Overflow**
>
> 스택과 힙 메모리가 겹치는 현상

---

# 4. 보조기억장치 (디스크)

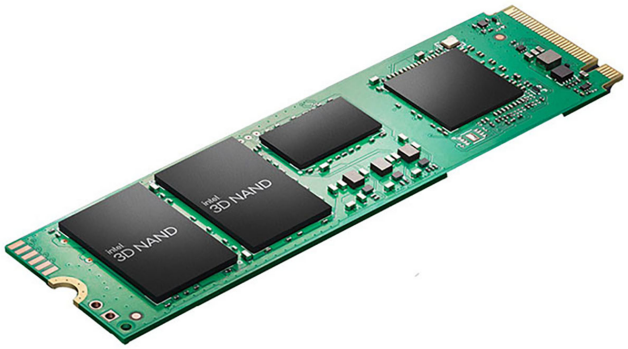
- 물리적인 디스크가 연결된 기억장치
- 주기억장치보다 느리지만 데이터가 영구적으로 저장됨

## 4-1. 종류

## 1) HDD(Hard Disk Driver)

- 하드디스크
- 물리적인 디스크를 고속으로 회전시켜 데이터를 저장
- 충격에 약하며 소음 발생
- 크기가 작고 처리속도가 향상된 SSD가 나옴에 따라 최근에는 적게 쓰이는 중

## 2) SSD(Solid State Driver)




- 반도체 기반으로 데이터를 저장
- 전기적으로 데이터를 저장하기 때문에 HDD에 비해 속도가 빠르고 소음 X
- 전력 소모가 적고 경량화, 소형화 가능
- HDD에 비해 비용이 큼
- 자기장에 약함