22.12.12

📌 데이터 저장구조 방식

캐시는 프로세서의 로드나 스토어 명령이 요구하는 1~8 Byte 데이터만을 메모리에서 가져오는 것이 아니라 캐시라인 단위로 모아서 가져온다.

- 라인 : 주메모리의 데이터를 읽어들이는 최소단위

1) Full Associate

- 태그 일치 체크 (프로세서로부터 받은 주소와 태그에 있는 주소 일치 확인)
- 전체 체크는 속도가 느리기 때문에 비교회로 장착해서 일치여부를 병렬적으로 체크

2) Direct Map

- 메모리 주소의 중간 비트값 사용
- 단점 : Conflick miss (캐시라인 쟁탈에 의한 성능 저하)
    - 캐시라인의 인덱스 = 메인 메모리주소 mod 캐시의 라인 개수

3) Set Associate 방식

메모리 주소에 대응하는 캐시라인 여러개인 것

📌 SOLID

1) Single responsibity priciple 단일 책임 원칙

> 클래스는 하나의 기능만을 가지고 클래스가 제공하는 모든 서비스는 하나의 책임을 수행하는데 집중되어 있어야 한다.

2) Open close principle 개방폐쇄원칙

확장에는 개방되어 있고, 변경에는 폐쇄되어야 한다.

3) Liskov Substitution Principle 리스코프 치환법칙

서브 타입은 언제나 기반 타입으로 교체할 수 있어야 한다.

4) Interface segregation principle 인터페이스 분리법칙

하나의 일반적인 인터페이스보다는 여러개의 구체적인 인터페이스가 낫다

5) Dependency inversion principle 의존역전법칙

추상화에 의존해야지 구체화에 의존하면 안된다.

📌 HashMap에서 충돌을 핸들링하는 방법

- Open address

1) 다음칸을 계속 확인하며 비어있는 칸 찾기

2) 제곱수의 칸을 확인하며 비어있는 칸 찾기

3) 2차 해시함수 적용하기

- Seperate chaining

1) linkedlist 사용하기

2) Tree 사용하기

📌 HashMap vs LinkedHashMap

HashMap

- Key, Value 구조
- 시간 복잡도 O(1)

LinkedHashMap

- Linkedlist + HashMap
- 순서를 가진 HashMap
- HashMap보다 메모리 사용량이 높다.

📌 Linkedlist vs ArrayList

ArrayList

- 삽입, 삭제시 새로운 배열 생성

- 메모리 사용량 높다.

- 기존의 ArrayList는 가비지 컬렉션에 의해 메모리에서 제거된다.

- 접근에 용이하다.

LinkedList

- 삽입, 삭제시 빠르다.
- 접근시 순차적으로 확인해야 하므로 느리다.