CASLock을 할 때의 쓰이는 int 변수도 최적화를 하기 때문에 이상한 값이 나온다. 그러므로 이 변수를 직접 메모리에 접근하여 최적화를 하지 못하도록 volatile을 쓴다.

이제 제대로 나온다. 하지만 방심하면 안된다. 주의를 기울여야 한다.

멀티스레드 프로그래밍을 잘 하기 위해서는 디스어셈블리를 통한 디버깅을 사용한다. (어셈블러 공부 필요)

5장 Non-Blocking 알고리즘 – LIST

링크드 리스트

* 목표 자료구조

Lock Free Linked List

Linked List 사용이유? SET

\* SET

1. 아이템의 중복 허용 X

2. 아이템 정렬로 검색 효율 증대

3. 삽입삭제 효율을 위해 Linked List로 구현

-> 구현할 method >> add(need search), remove(need search), contains

\*필드(Node)

- key, next

\*보초 노드

링크드 리스트를 사용할 때 편함

>> 예외상황을 줄여서 알고리즘을 최적화 하기 위함

앞, 뒤 넣는 다른 방법을 중간에 끼우는 방법과 통일이 가능해진다.

* 성긴 동기화
* Lock하나로 동기화 객체 전체를 감싸는 경우
* 추후 점점 보완된 동기화를 사용할 것이다.

-구현

하나의 잠금, 모든 메서드 호출은 이 잠금을 통해 Critical Section으로 진행된다.

-모든 메서드는 잠금을 가지고 있는 동안에만 리스트에 접근한다.

Head는 변화가 없으므로 lock범위 안에 포함시키지 않게하여 미세하게 성능향상을 꾀한다.

* 세밀한 동기화

Node에만 락