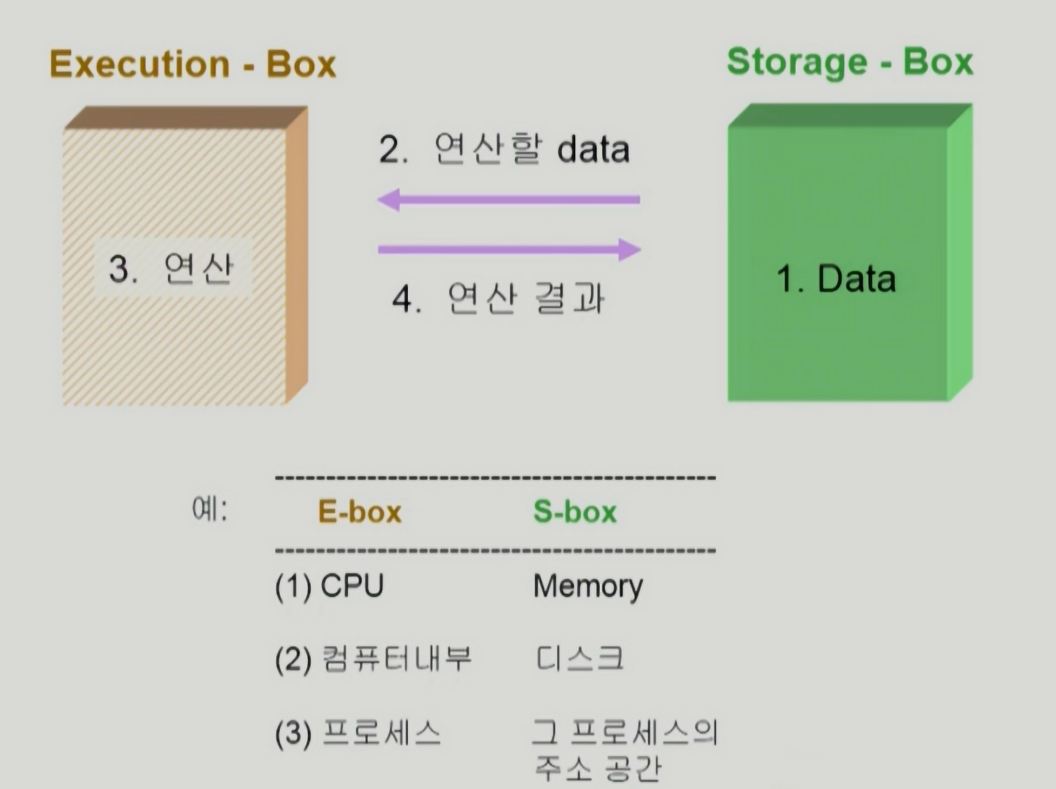
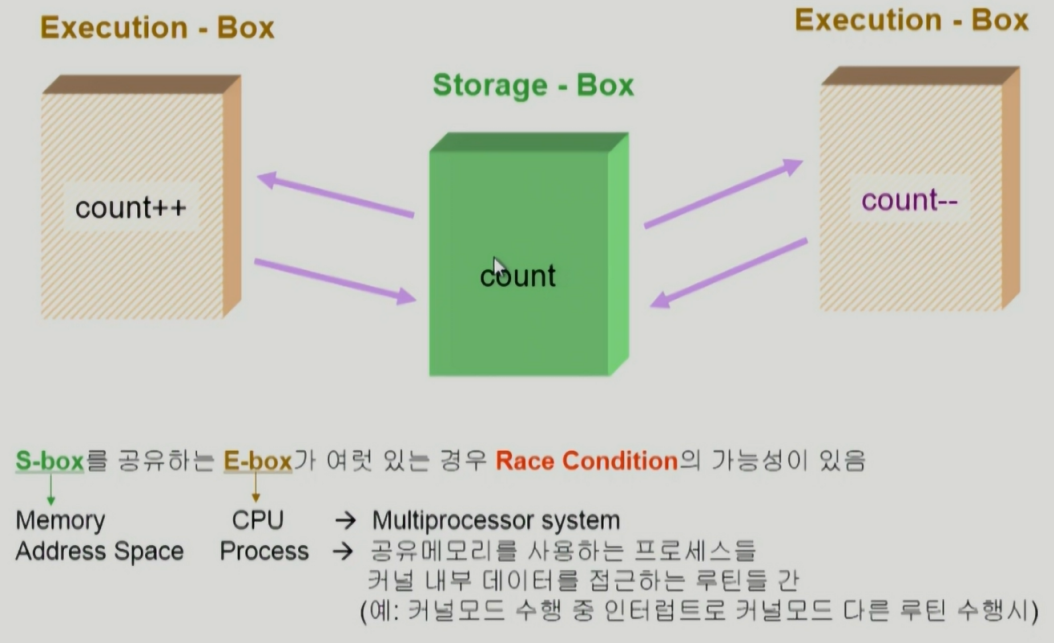
6. Process Synchronization

▼ 목차

0. 데이터 접근 패턴



하나의 E-box와 S-box 만 존재한다면 괜찮지만



하나의 S-box에 대해 두 개 이상의 E-box가 접근한다고 하면 연산이 꼬일 수 있다 (Race-condition)
여러 프로세스들이 동시에 데이터에 접근하는 상황에서, 어떤 순서로 데이터에 접근하느냐에 따라 결과 값이 달라질 수 있는 상황

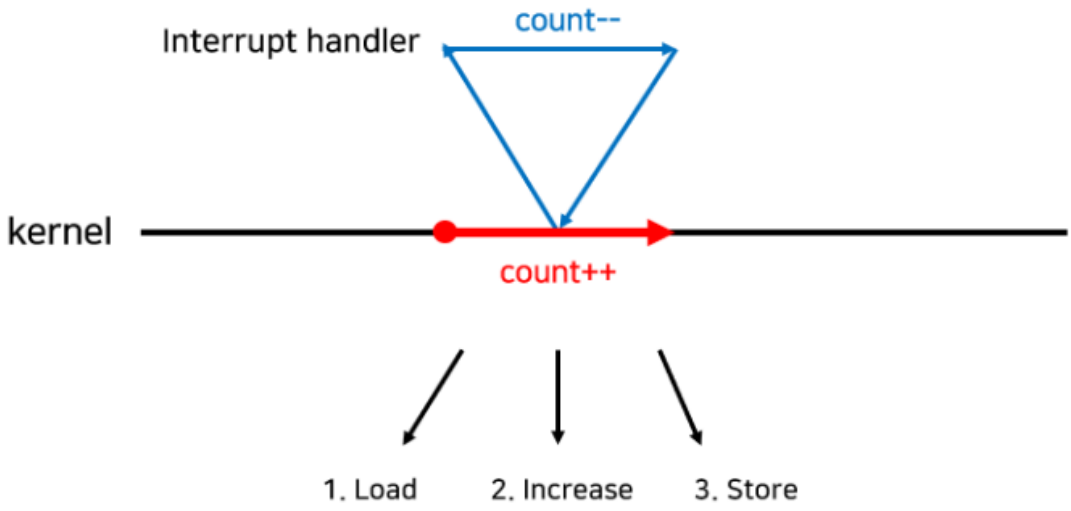
공유 데이터의 동시 접근(Concurrent access)은 데이터의 불일치 문제를 발생시킬 수 있다.

따라서, Race condition을 막고 일관성을 유지하기 위해서는 협력 프로세스 간의 실행 순서를 정해주는 메커니즘인 동기화(Synchronization)가 필요하다.

---

1. Race Condition 이 발생하는 경우

1-1. kernel 수행 중 인터럽트 발생할 경우

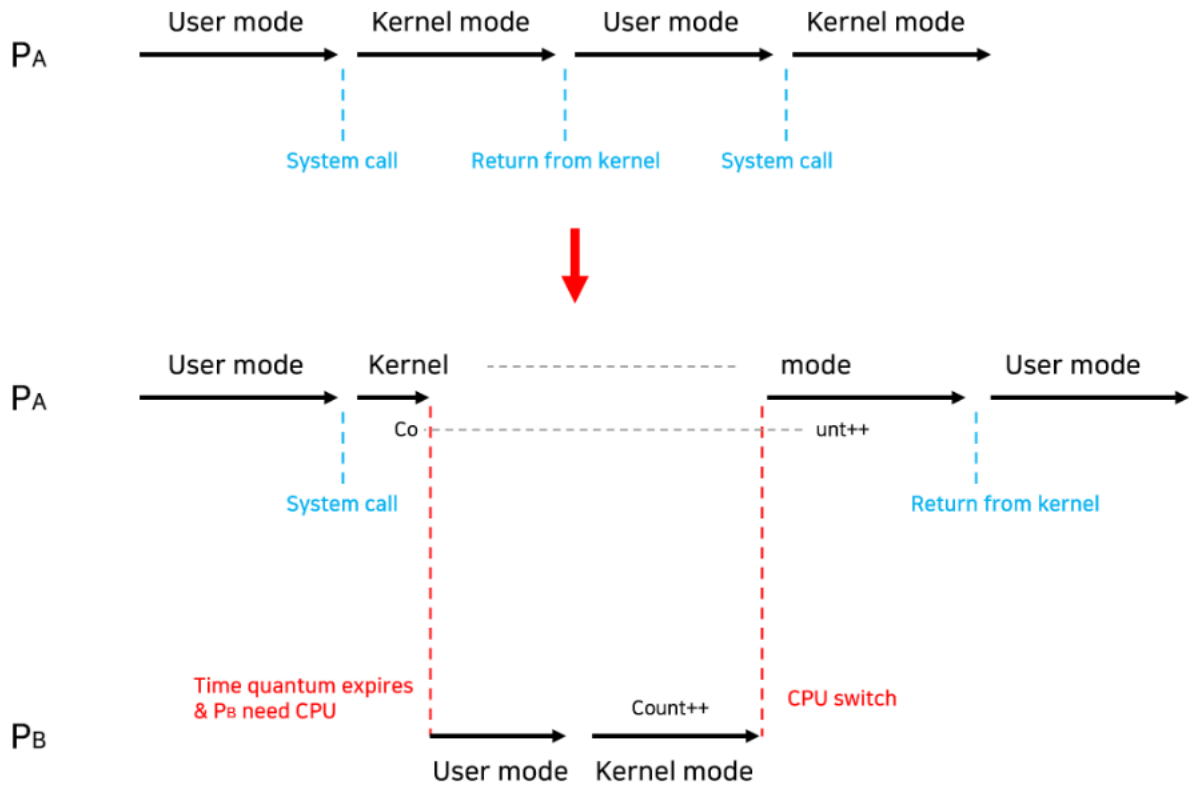


의도된 동작은 count++과 count--가 모두 반영되어 count가 초기값을 유지하는 것이지만, 만약 Load를 한 후에 인터럽트가 발생하는 경우 인터럽트의 결과는 반영되지 않고 count++만 반영

이는 커널 모드의 수행이 끝나기 전에는 인터럽트를 받지 않도록 하는 방법(disable/enable)으로 문제를 해결할 수 있다.

1-2. 프로세스가 시스템 콜을 호출해서 커널 모드 수행 중에 Context switch가 발생하는 경우

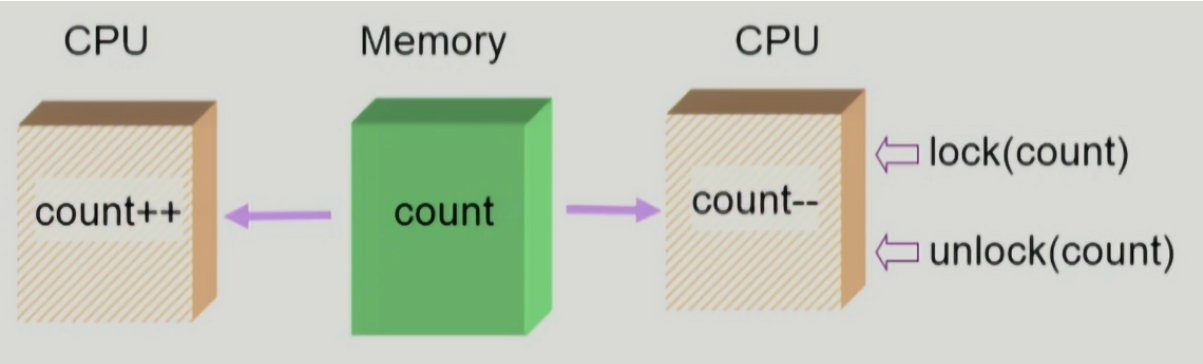
두 프로세스의 주소 공간에서는 데이터를 공유하지 않지만, 시스템 콜을 수행하는 동안에는 둘 다 커널 주소 공간의 데이터를 접근한다. 따라서 커널 주소 공간에서 작업을 수행하는 도중에 CPU를 빼앗으면 race condition이 발생한다.



커널 모드를 수행 중일 땐 CPU가 preempt 되지 않도록 하고,
커널 모드에서 유저 모드로 돌아갈 때 preempt 되도록 함으로써 해결할 수 있다.

1-3. 멀티 프로세서에서 공유 메모리 내의 커널 데이터에 접근하는 경우

CPU가 여러 개 있는 경우



어떤 CPU가 마지막으로 Count를 저장했는지에 따라 결괏값이 달라진다.

- 싱글 프로세서인 경우 1번에서와 같이 인터럽트 disable/enable 방법으로는 해결할 수 있지만 멀티 프로세서인 경우 인터럽트 제어로는 해결할 수 없다.

---

1. 한 번에 한 CPU만 커널에 들어갈 수 있도록 하는 방법
   비효율적이다. 만약 두 프로세서가 서로 다른 데이터에 접근하여 Race condition의 가능성이 없음에도 불구하고 한 번에 한 CPU만 커널에 들어갈 수 있기 때문이다.
2. lock/unlock 방법
   커널 내부에 있는 각 공유 데이터에 접근할 때마다 그 데이터에 대해서만 lock/unlock을 하는 방식으로 해결할 수 있다.
