동기화
문제

# 동기화 문제

## 동기화 문제

- 자원에 두 개 이상의 스레드가 동시접근 할 때 발생
- 자원 동시접근 발생 케이스
  1. 프로세스가 인터럽트 발생으로 인해 중단(부분적으로 실행 완료) -> 다른 프로세스가 자원에 접근
  2. 각각 다른 CPU코어에서 동작중인 프로세스가 동시에 접근
- 데이터 동시에 접근으로 인해 무결성이 깨지는 경우를 "**레이스 컨디션**"
- 레이스 컨디션을 유발되는 코드영역을 "**크리티컬 섹션**"

## 레이스 컨디션 해결방법

1. 상호배제, 프로세스 A가 코드 실행 중이면 **아무도 못 들어오게 함**

2. 다음에 누가 들어올지 결정, 임계영역에 들어올 프로세스들의 **순서를 정함**

3. 기다리는 프로세스는 일정시간 후 임계영역에 들어가야 함, 임계영역에 들어간 프로세스는 **일정 시간 후에는 임계영역에서 나와야 함**

4. 프로세스가 임계영역에 있으면 할당시간이 지나거나 인터럽트가 나도 문맥전환을 하지 않게 함(확실하지만 추천하지는 않음)

## 커널의 레이스 컨디션 해결방법

- 전적으로 커널 개발자의 의지에 의존함
- **해결방법 1.** 선점형 커널

  한번 점유한 자원은 스스로 안 놔줌, 응답성이 좋다는 특징이 있음
- **해결방법 2.** 비선점형 커널(레이스 컨디션 면역임)

  프로세스가 커널 모드에서 실행되지 않게 함, 자원을 양보할 수 있게 하기 위해서

# 피터슨 해결방법

# 피터슨 솔루션

```
while (true) {
        flag[i] = TRUE;
        turn = j;
        while ( flag[j] && turn == j);
            //CRITICAL SECTION
        flag[i] = FALSE;
            //REMAINDER SECTION
    }
```

프로세스 2개일 때만 적용가능

1. **상호배제 예방** 만족
2. **순서제어** 만족
3. **Bounded-waiting** 만족

## 피터슨 솔루션

- 오늘날 컴퓨터와는 맞지 않음
- 컴퓨터는 load / store 같은 기본적인 연산만 수행하기 때문에 틈이 생길 수 있다
- 실제 컴퓨터에서 완벽한 동작을 보증하지는 않는다(이론적으로는 괜찮음)

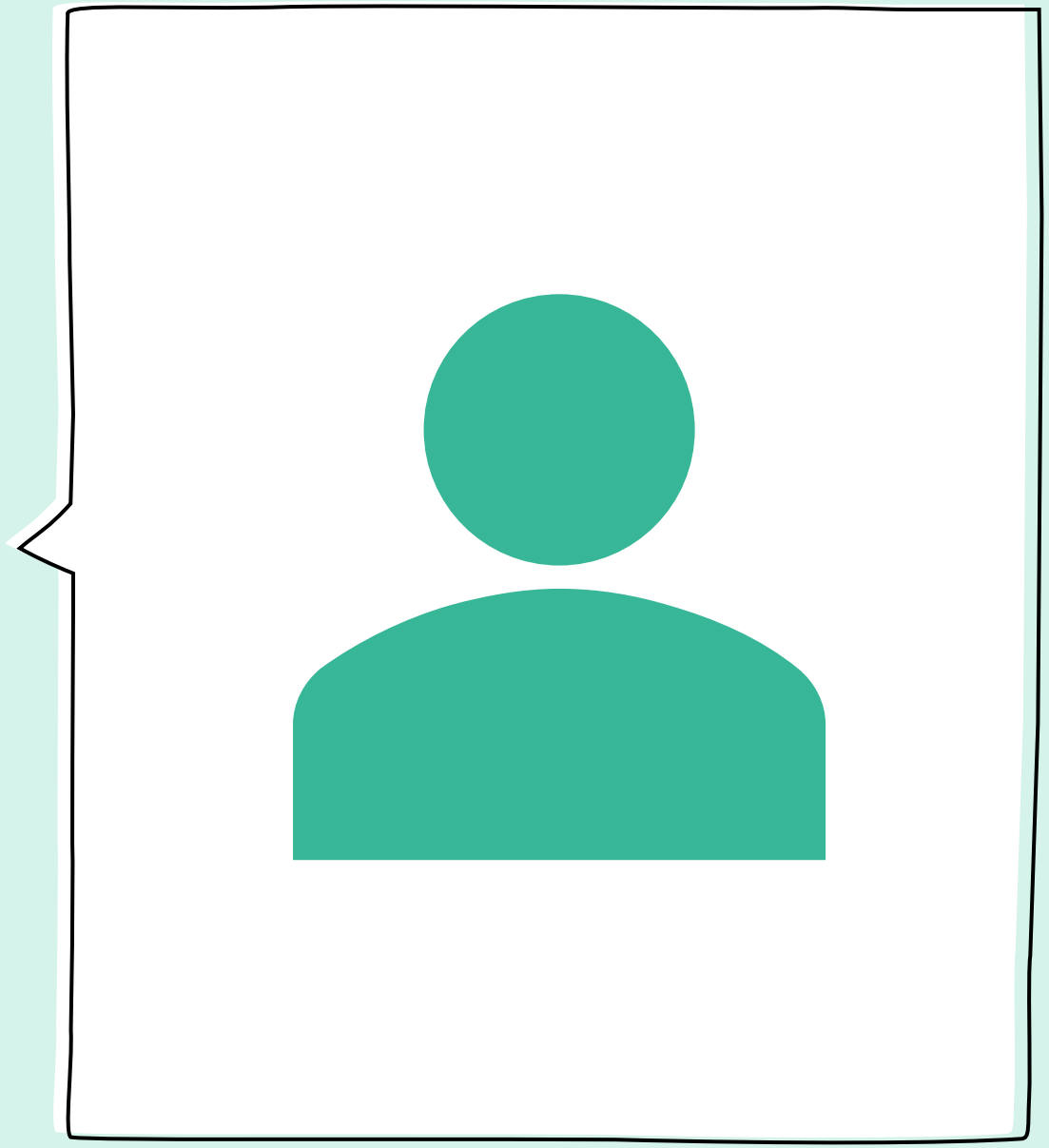
# 하드웨어 동기화

## 하드웨어 동기화

- 피터슨 솔루션 같은 소프트웨어 기반 해결 방법을 보완
- Lock을 거는 것을 전제로 함
- 싱글 프로세스 사용, 인터럽트 비활성화, 명령어 원자적 실행 등이 있음

# 싱글 프로세스 사용

- 싱글 프로세서에서 선점없이 코드실행 함으로서 공유자원문제해결
- 다른 프로세스가 실행 안되니까 다른 프로세스의 예상치 못한 명령어가 실행될 일도 없음

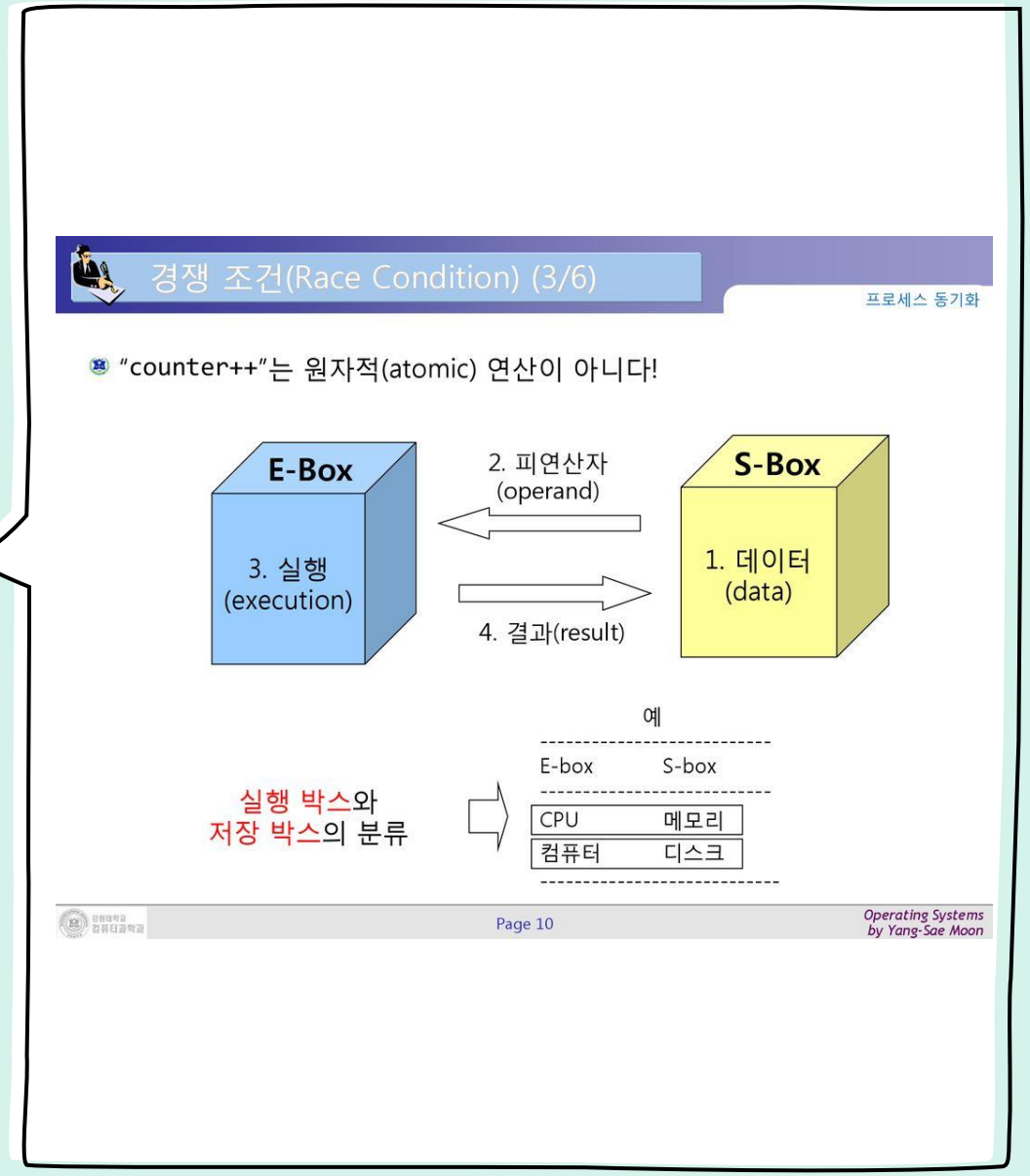


# 인터럽트 비활성화

- 공유데이터 수정 중 일어나는 인터럽트 무시
- 비선점 커널에서 사용
- 모든 프로세서에게 인터럽트 무시 메시지를 보내야 해서 시간 소모가 큼

# 명령어 원자적 실행

- Atomic_t 같은 원자적 명령어 사용
- Load / store 수행 중 방해받지 않게 함



# Lock 종류

## Lock 종류

1. Spin lock
   - Cpu가 뺑뺑이 돌면서 자원이 사용가능한지 체크
   - 임계영역이 짧을 때 사용(= 자원이 금방 반환될 거 같을 때 사용)
   - 문맥전환을 하지 않고 자원을 기다린다.
   - Cpu 사이클이 낭비된다는 단점이 있음
2. Mutex
   - 임계영역 진입하기 전 lock얻고 나올 때 lock 반환
3. semaphore

## Lock 종류

3. Semaphore

- 두 프로세스가 동시에 세마포어 수정 불가능
- 세마포어 값 테스트 중 인터럽트 비활성화
- 임계영역에 들어가기 원하면 wait 함수 호출, 세마포어 값이 0 보다 크면 임계영역에 들어감
- 임계영역에 들어갈 때 세마포어 값--
- 임계영역에서 나올 때 signal 함수 호출해서 세마포어 값++
- 실행순서 제어에도 사용가능

## Lock 종류

### Binary Semaphore

- 세마포어는 0과 1 두가지 값만 가짐
- 뮤텍스 처럼 동작함.
- 뮤텍스 제공안하는 운영체에서 뮤텍스 대용으로 사용

### Counting Semaphore

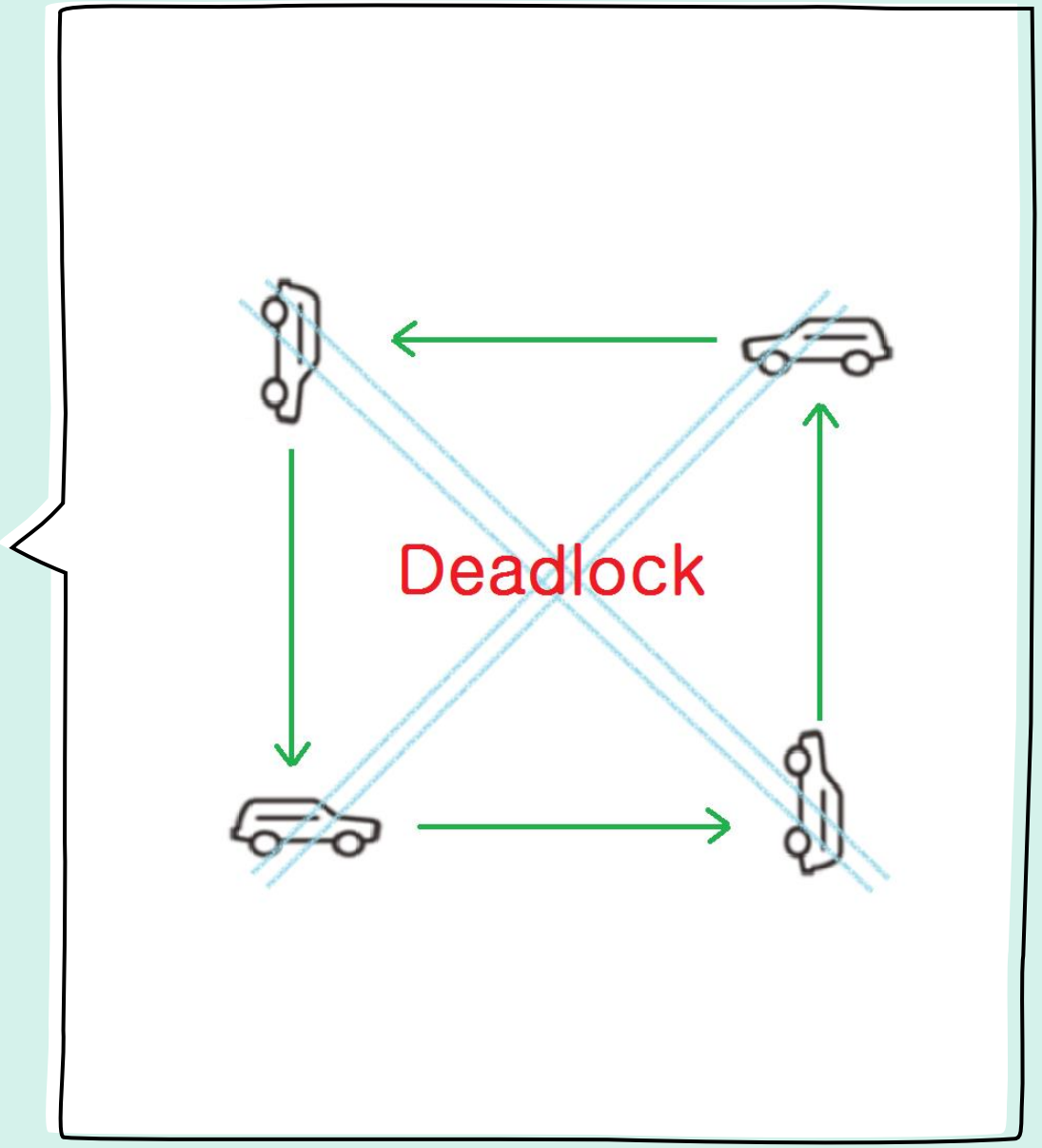
- 세마포어는 0 ~ N 까지 여러가지 값을 가짐
- 유한한 자원에 접근을 제한하는 데 사용

# 세마포어 문제

세마포어 문제

# 데드락

- 여러 개의 프로세스가 서로 가지고 있는 자원을 기다림
- 프로세스간 문맥전환만 하다가 끝남
- 무한 블로킹, starvation



세마포어 문제

# 우선순위 역전

- 우선순위 낮은 애가 공유자원 가지고 있을 때 우선순위 높은 애들이 계속 와서 우선순위 낮은 애 실행불가능
- 우선순위 낮은 애가 가지고 있는 자원을 기다리는 프로세스는 계속 대기하게 됨