# Real-Time Operating System (Day 5 Lab)


**Jong-Chan Kim**

**Graduate School of Automotive Engineering**

KMU 국민대학교 KOOKMIN UNIVERSITY

# 25. MDemo

- 24. MTemplate 복사
- OIL 파일에 CPU_DATA 추가 및 TASK에 CPU_ID 정의

```
…

CPU_DATA = TRICORE {
    ID = 0x0;
};

CPU_DATA = TRICORE {
    ID = 0x1;
};

CPU_DATA = TRICORE {
    ID = 0x2;
};

…
```

CORE1 활성

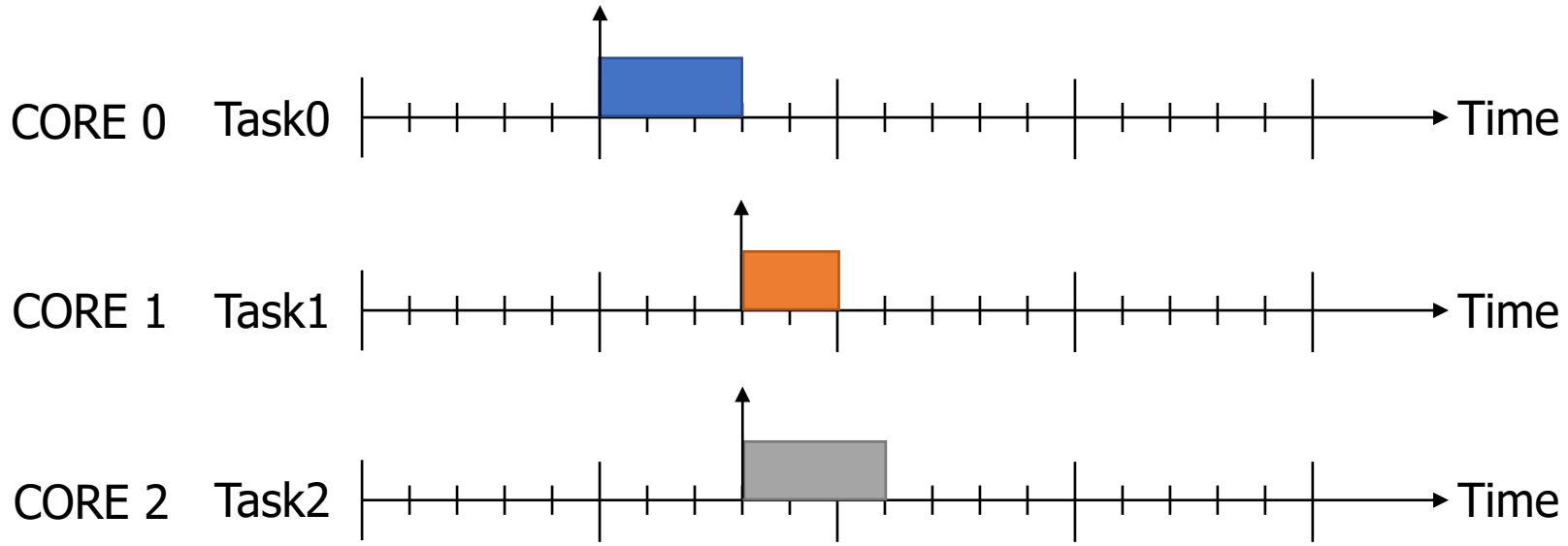
CORE2 활성

```
…

TASK Task1 {
    CPU_ID = 0x1;
    PRIORITY = 1;
    AUTOSTART = TRUE;
    STACK = SHARED
    ACTIVATION = 1;
    SCHEDULE = FULL;
};

…
```

TASK가 올라갈 CPU ID

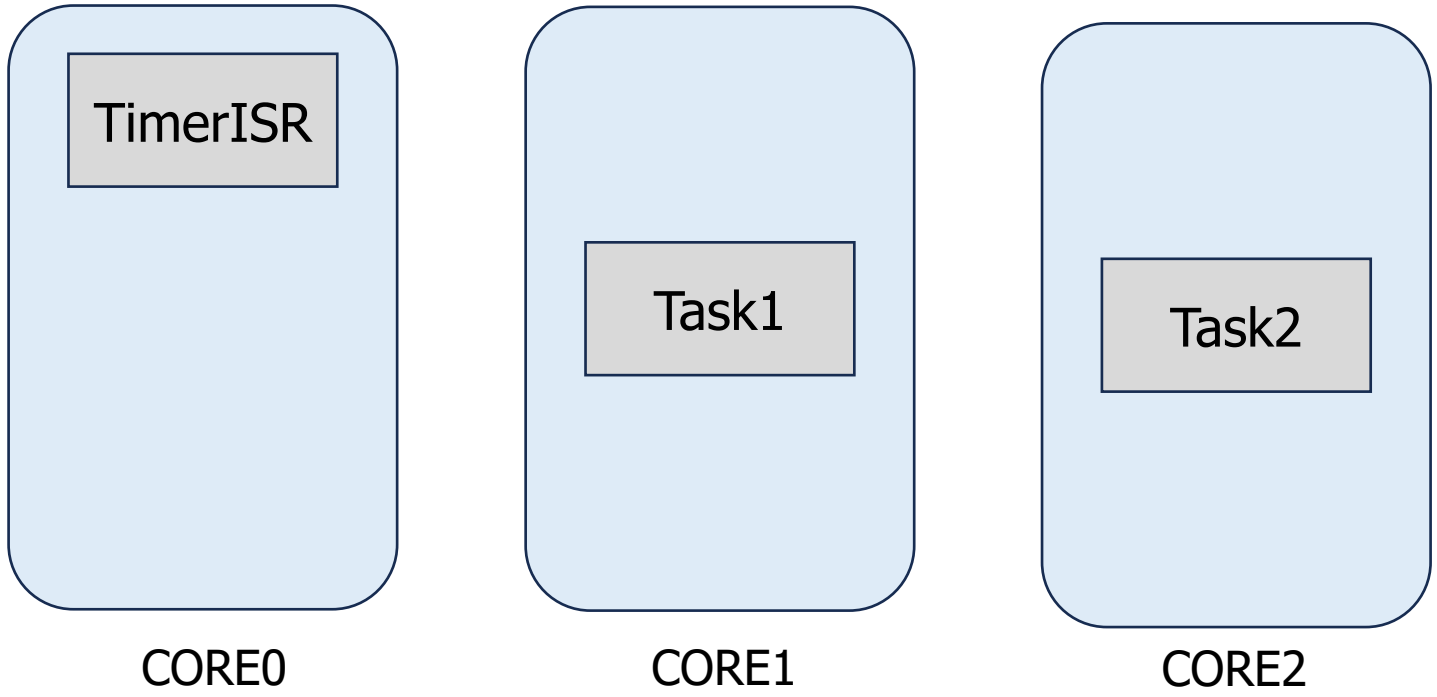
# 25. MDemo

- 06. Task Activation 참고하여 다음 스케쥴 재현
- Task0은 TimerISR에서 ActivateTask()
- Task1 과 Task2는 Task0에서 ActivateTask()
- Task 우선순위는 모두 1



# 26-1. MLoss

- 17-1. Loss 를 참고하여 다중 코어 공유 Data Loss 재현
- 다음과 같이 Task를 코어에 분배, Task0 제거




- TimerISR은 기본적으로 CORE0에서 작동

# 26-1. MLoss

- Timer 설정 변경

bsw.c

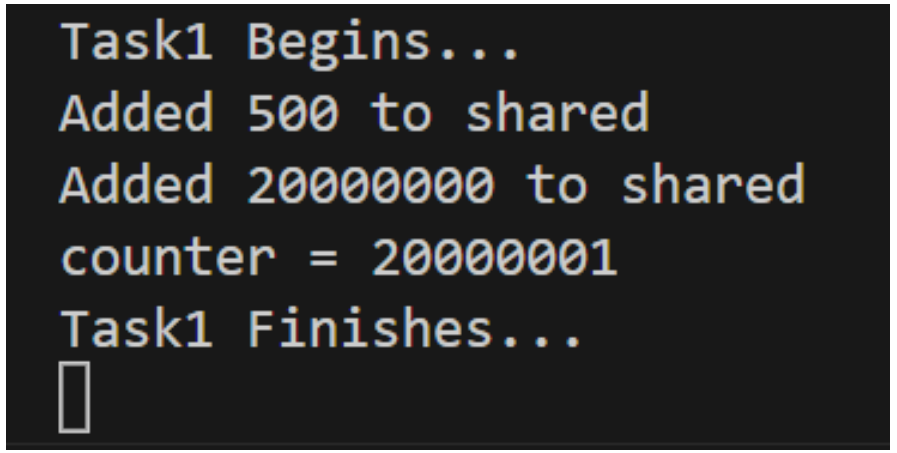
```
int main(void)
{
    …
    osEE_tc_stm_set_clockpersec();
    osEE_tc_stm_set_sr0(1000U, 1U);
```

asw.c

```
ISR2(TimerISR)
{
    osEE_tc_stm_set_sr0_next_match(1000U);
    IncrementCounter(mycounter);
}
```

# 26-1. MLoss

- SPINLOCK 을 이용해서 다중 코어 Date Loss 문제 해결 필요



# 26-1. MNoLoss

- OSEK의 SPINLOCK 기능을 이용하여 Data Loss 문제 해결

```
GetSpinlock(S1);
shared++;
ReleaseSpinlock(S1);
```

conf.oil

```
…
SPINLOCK S1 {};
…
```

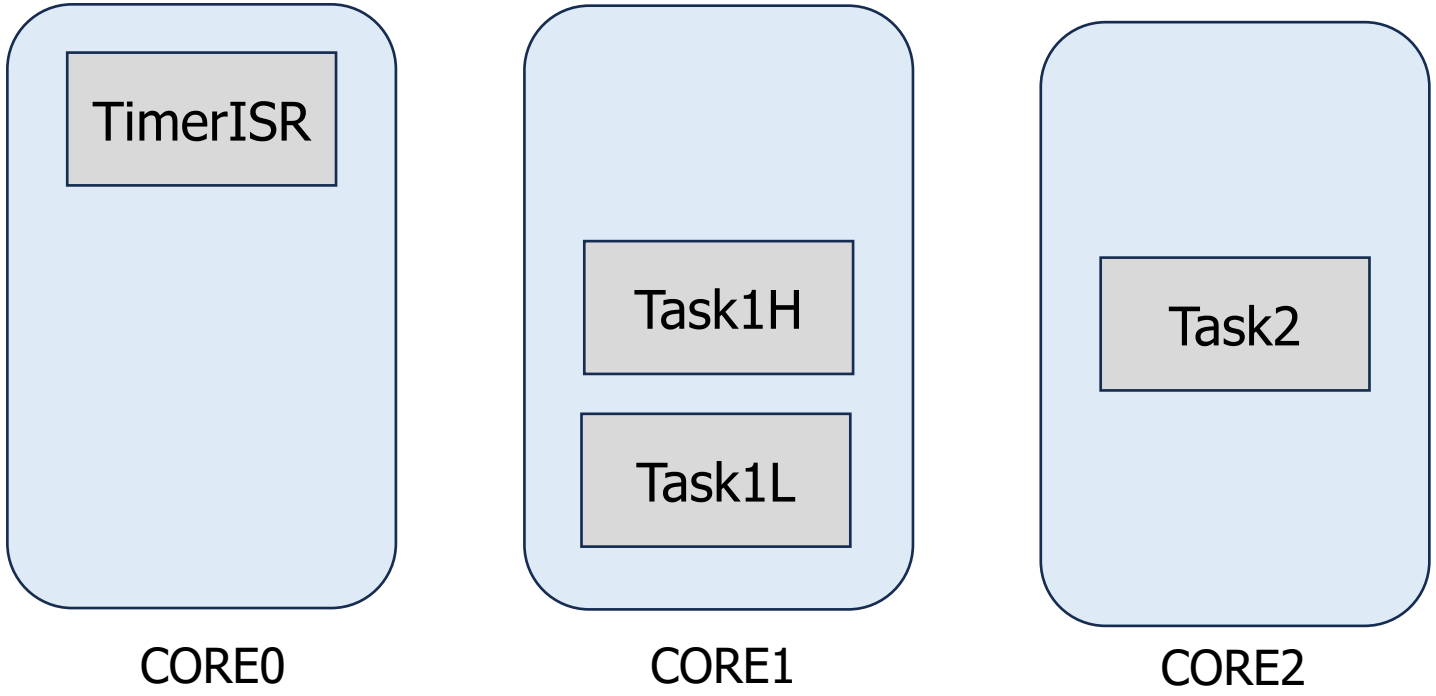
# 26-2. MNoLoss

- OSEK의 SPINLOCK 기능을 이용하여 Data Loss 문제 해결

```
Task1 Begins...
Added 500 to shared
Added 20000000 to shared
counter = 20000500
Task1 Finishes...
```

# 27-1. MDeadlock1

- 단일코어에서 같은 SPINLOCK 사용 시 발생하는 Deadlock 재현
- 24. MTemplate 복사하여 재현
- Task2는 Dummy Task (MDeadlock2 에서 사용 예정)



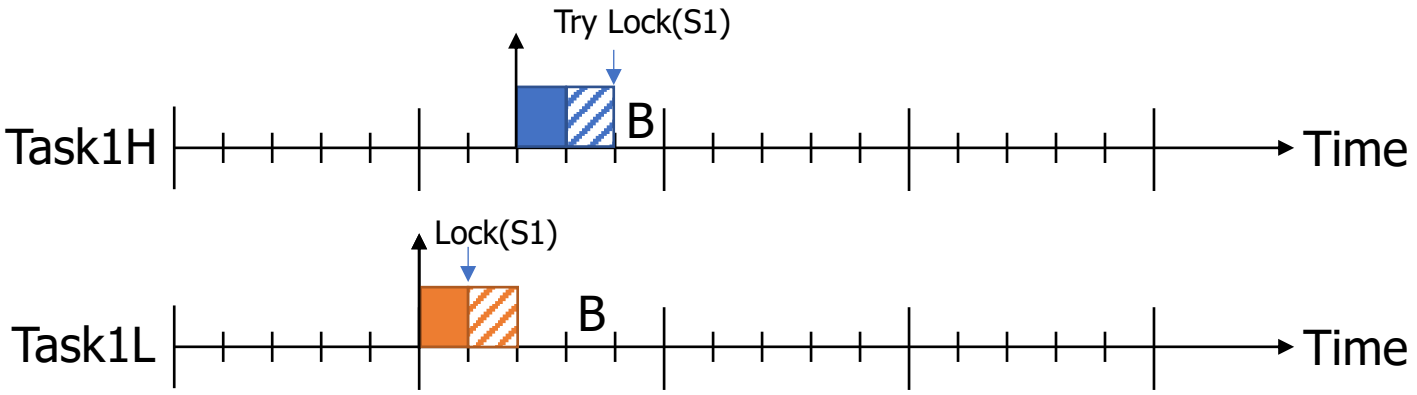

- 19-2. No Priority Inversion 참고하여 재현

# 27-1. MDeadlock1

- Task2 (Dummy Task) 작성
- 오른쪽 스케쥴 재현하고 Deadlock 확인

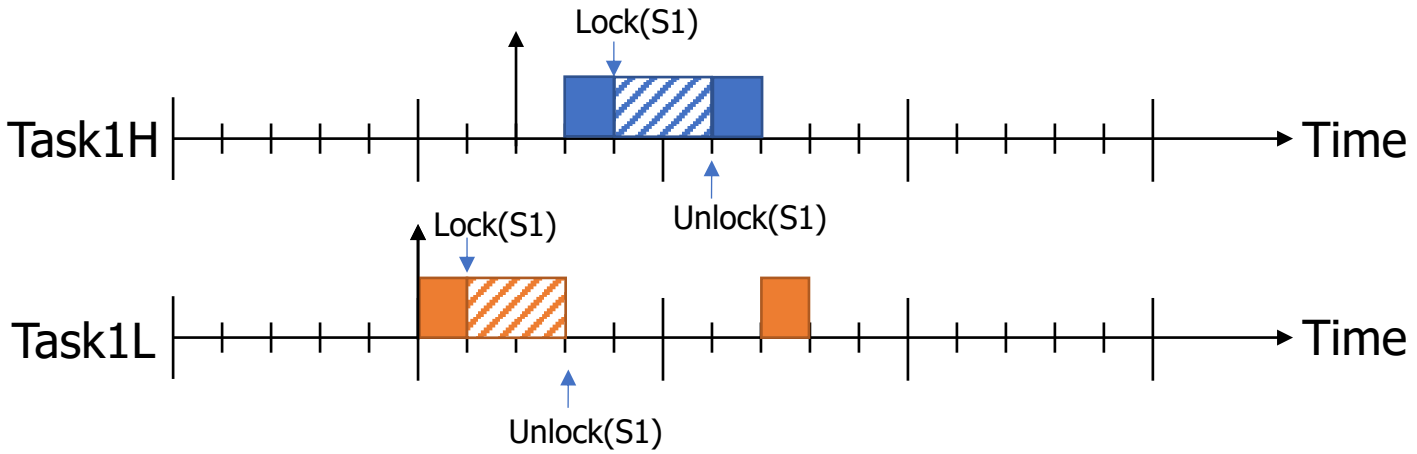
```
TASK(Task2)
{
    TerminateTask();
}
```

```
TASK Task2 {
    PRIORITY = 1;
    STACK = SHARED;
    SCHEDULE = FULL;
    AUTOSTART = FALSE;
    ACTIVATION = 1;
};
```



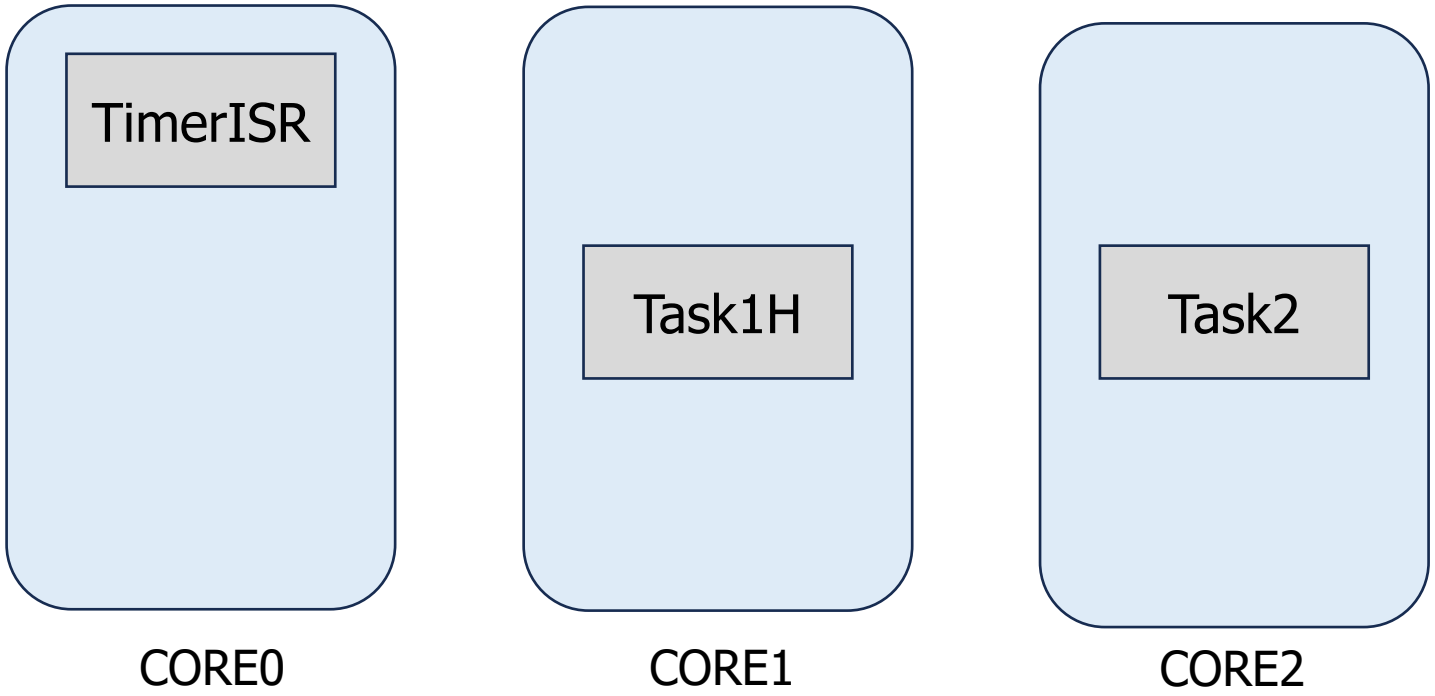
# 27-2. MNoDeadlock1

- SuspendAllInterrupts() 사용하여 Deadlock 해결
  - 코어 안에서의 스케쥴링을 방지



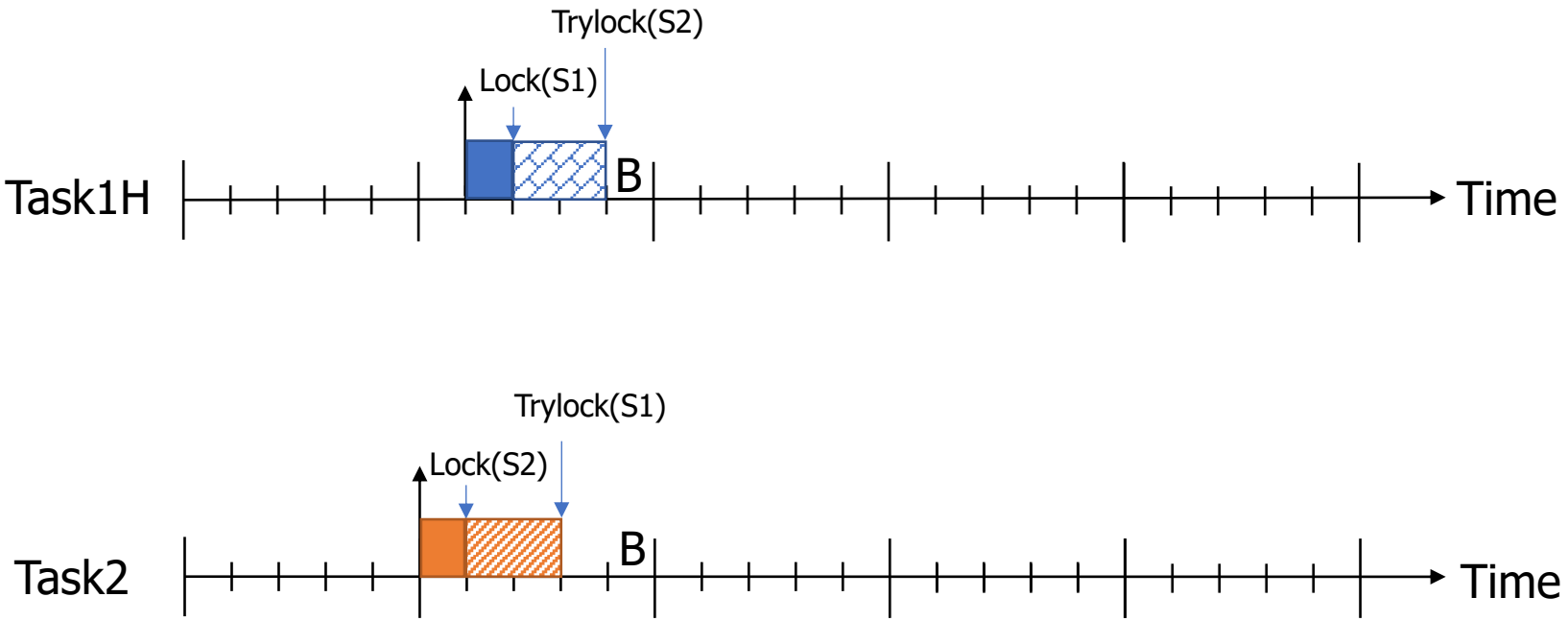
# 28-1. MDeadlock2

- 멀티코어에서 중첩 SPINLOCK 사용 시 발생하는 Deadlock 재현
- 다음과 같이 Task를 코어에 분배, Task1L 제거



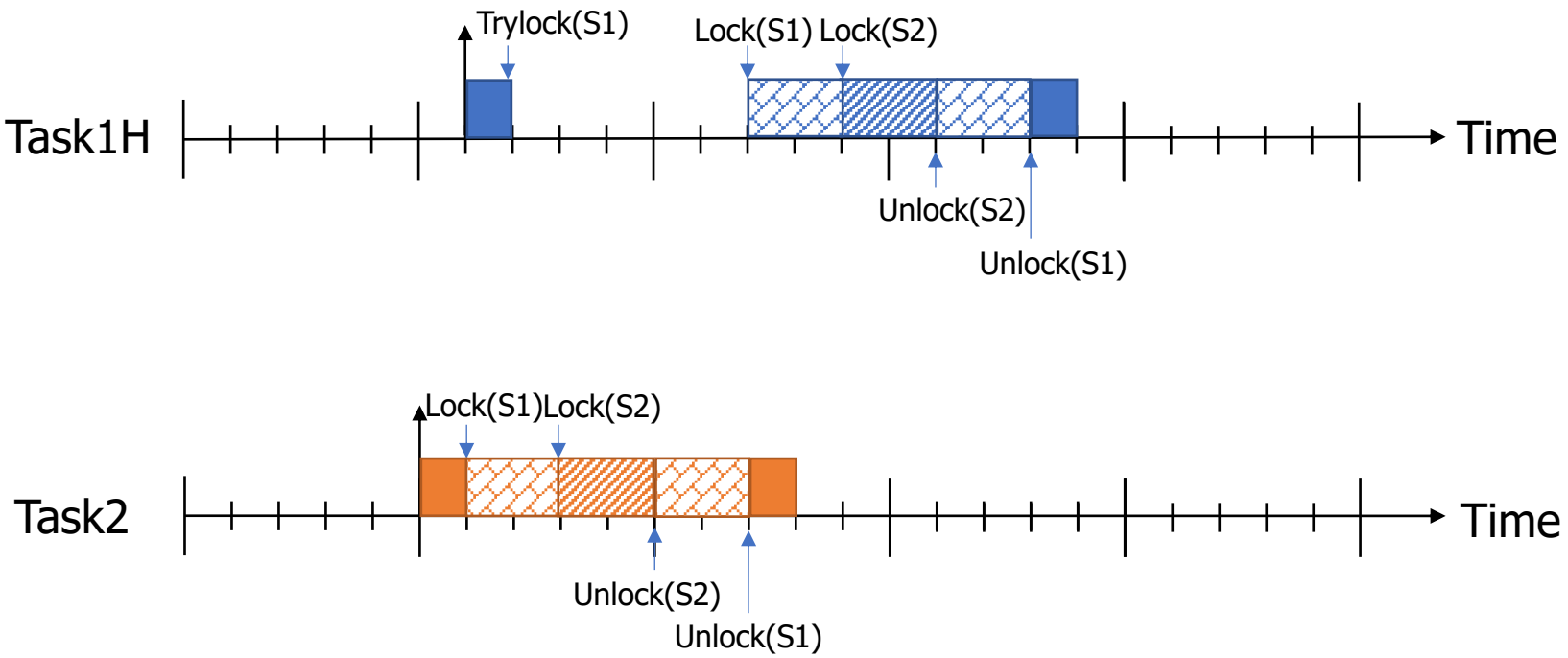
# 28-1. MDeadlock2

- 틀린 순서로 중첩 SPINLOCK 사용 시 발생하는 Deadlock 재현



# 28-2. MNoDeadlock

- SPINLOCK 획득 순서를 정하여 Deadlock 해결
- SPINLOCK 획득 순서 : S1 ➔ S2



# Questions