

Process Synchronization in Multi Processor

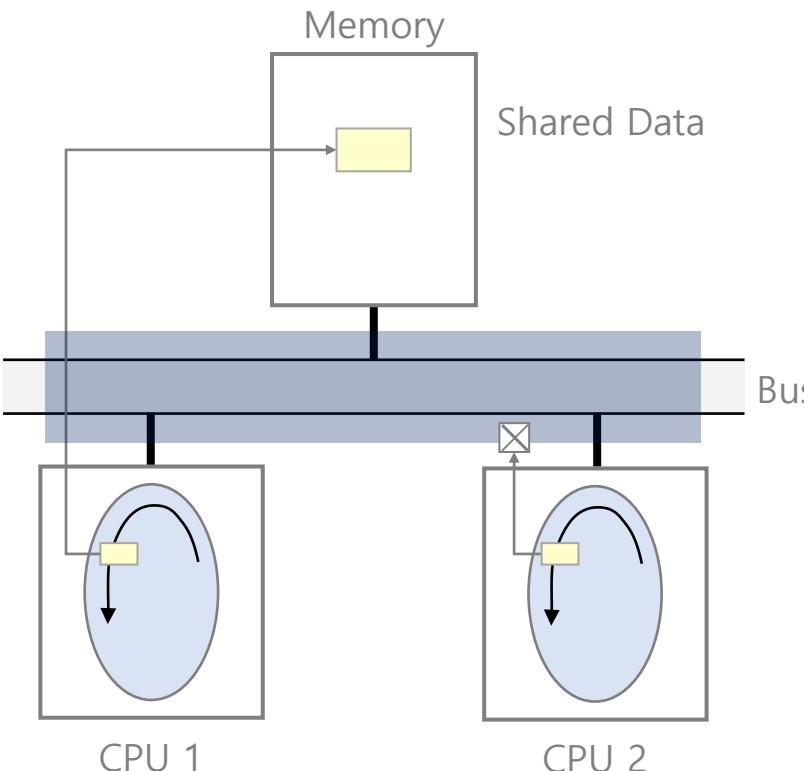
Problem

❖ Multi processor에서는 disable interrupt를 하더라도 다른 CPU에서 Thread가 동작하므로 다른 기능이 필요하다.

Solution

❖ 하나의 CPU가 Critical Section을 수행할 때, 다른 CPU가 해당 메모리에 접근하지 못하도록 해야 한다.

- Multi Processor에서는 Bus를 disable 시켜야 한다.



기존

1. Bus를 사용해서 Memory Read
2. CPU Register에서 **Modify**
3. Bus를 통해 Memory에 다시 **Write**

CPU1이 disable interrupt하든지 말든지
CPU2도 Memory에서 Read 가능하다.

해결

1. Bus를 사용해서 Memory Read
2. CPU Register에서 **Modify**
3. Bus를 통해 Memory에 다시 **Write**

Test and Set Operation

이 과정이 Atomic한 연산으로 만든다.

Process Synchronization in Multi Processor – Test and Set (TAS) instruction

Single Processor

```
disableInterrupt();  
//... Critical Section  
  
enableInterrupt();
```

TAS(특정 메모리 주소)

1. 메모리 주소의 값을 Read
2. 해당 값이 "0"이면 값을 1로 변경하고 Critical Section 진입
3. 만약 "1"이라면 Spin Lock 동작

⚠ 이 과정이 H/W에 Atomic 연산으로 지원 되어야 한다.
그렇지 않으면, CPU 1 & CPU 2이 동시에 메모리 접근 후 "0" 값을 읽어오고 Critical Section에 진입하게 된다.

Multi Processor

```
disableInterrupt();  
while (TAS(lockMem) != 0) continue;  
//... Critical Section
```

```
lockMem = 0;
```

```
enableInterrupt();
```

Multi Processor라도 당연히 Critical Section에서 CPU Scheduling을 막아한다.

Process Synchronization – Spin Lock & Scheduling

Spin Lock

- ISR(Interrupt Service Routine)의 경우 Scheduling의 대상이 될 수 없다.
 - 따라서, **ISR이 공유 자원에 접근한다면 Spin Lock 밖에 사용할 수 없다.**
- Spin Lock은 CPU를 점유하고 계속 요청하기 때문에 자원을 낭비하는 것 처럼 보인다.
 - 하지만, **Critical Section이 굉장히 작다면**, Scheduling 되면서 Context Switching이 발생하는 것보다 **Spin Lock으로 구현** 하는 것이 자원을 적게 소모한다.

Process Synchronization – Conditional Variable

❖ 상황 예시

Critical Section에 진입해서 공유 버퍼의 데이터를 Read하려고 했다.
하지만, 버퍼에 데이터가 없다면.. Block되어 버린다.
만약 이때, Critical Section에서 Sleep() 해버리면, 아무도 Critical Section에 들어 올 수가 없다.

따라서, Critical Section의 Lock을 Release 하고 **Conditional Variable이라는 Waiting Queue**에서 대기한다.