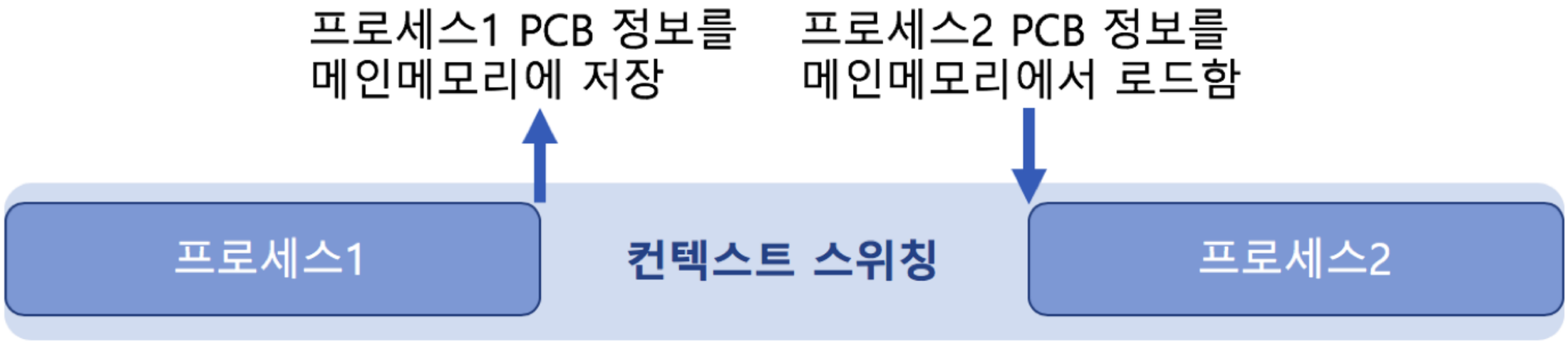
프로세스와 컨텍스트 스위칭

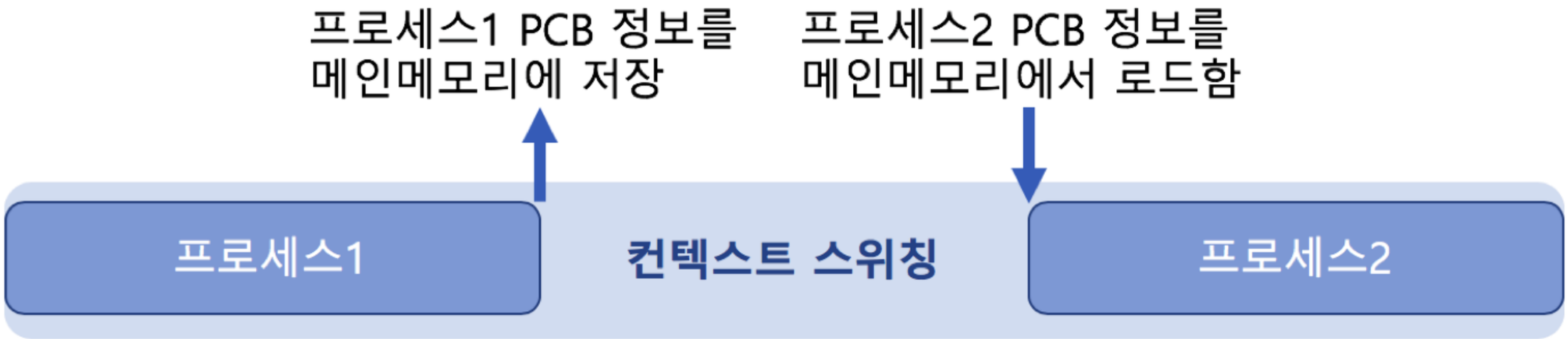
- Context Switching (문맥 교환)
  - CPU에 실행할 프로세스를 교체하는 기술

1. 실행 중지할 프로세스 정보를 해당 프로세스의 PCB에 업데이트해서, 메인 메모리에 저장
2. 다음 실행할 프로세스 정보를 메인 메모리에 있는 해당 PCB 정보를 CPU에 넣고, 실행



프로세스와 컨텍스트 스위칭

1. 실행 중지할 프로세스 정보를 해당 프로세스의 PCB에 업데이트해서, 메인 메모리에 저장
2. 다음 실행할 프로세스 정보를 메인 메모리에 있는 해당 **PCB 정보(PC, SP)를 CPU의 레지스터에 넣고**, 실행



> 마치 레코드판 같은?
>
> 출처: https://youtu.be/vvNqDRLYN64

프로세스와 컨텍스트 스위칭

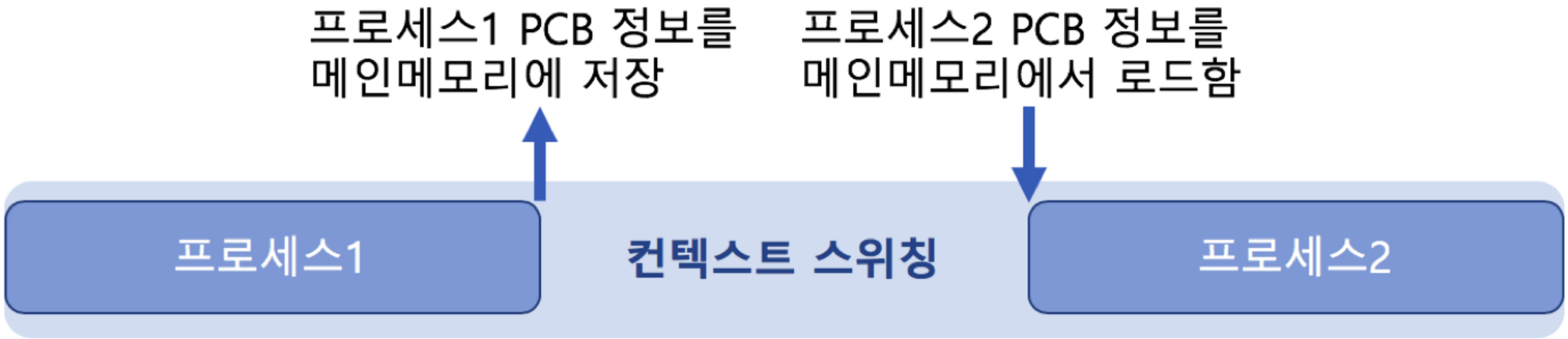
2. 다음 실행할 프로세스 정보를 메인 메모리에 있는 해당 **PCB 정보(PC, SP)를 CPU의 레지스터에 넣고**, 실행

> 디스패치 (dispatch): ready 상태의 프로세스를 running 상태로 바꾸는 것

> OS.xlsx -> ProcessWithCS

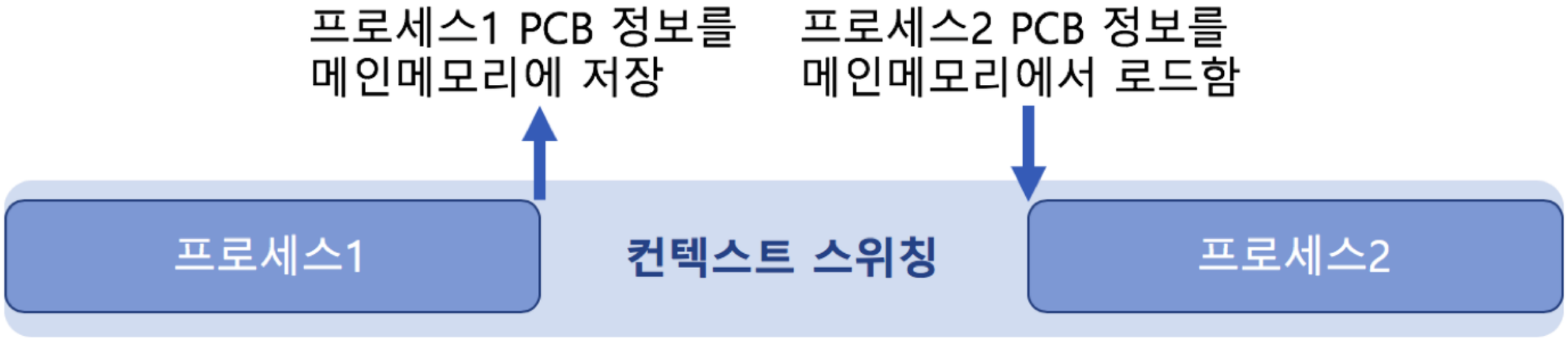
프로세스와 컨텍스트 스위칭

> 컨텍스트 스위칭 시간이 오래 걸리면...
> 실제로는 굉장히 짧은 시간 (ms) 단위로, 프로세스 스위칭이 일어남



프로세스와 컨텍스트 스위칭

> 어떻게 하면 조금이라도 컨텍스트 스위칭 시간을 짧게 할 수 있을까?



> C언어가 아니라, 어셈블리어로 컨텍스트 스위칭 코드를 작성하면...

여기서 잠깐! - 컴파일러

- 초기 컴퓨터 프로그램들은 어셈블리어로 작성
  - 서로 다른 CPU 아키텍처가 등장할 때마다 매번 똑같은 프로그램 작성
  - 어셈블리어로는 프로그램 작성 속도가 매우 떨어짐
- 컴파일러 등장
  - CPU 아키텍처에 따라서는 컴파일러 프로그램만 만들면 됨, 기존 코드는 재작성할 필요 없음
  - 그러나, 어셈블리어로 작성한 코드보다는 속도가 떨어질 수 있음

여기서 잠깐! - 컴파일러

- 어셈블리어로 작성했다면...

> 리눅스의 경우 컨텍스트 스위칭 코드는 각 CPU마다 별도로 존재

여기서 잠깐! - 어셈블리어

> 아니 그럼? 나도 어셈블리어를 알아야 하나?

정리

- 프로세스 구조
  - Stack, HEAP, DATA(BSS, DATA), TEXT(CODE)
- PCB
  - 프로세스 상태 정보 - PC, SP, 메모리, 스케쥴링 정보등
- 컨텍스트 스위칭
  - 프로세스 상태 정보를 PCB로부터 CPU에 로드하고, 실행

> 레코드판 가운데에 핀을 놓는 느낌!