프로세스 개념

4th Week

Kim, Eui-Jik

Contents

- 소개
- 프로세스 상태: 프로세스 생명 주기
- 프로세스 관리
- 인터럽트

소개

- 컴퓨터는 동시에 여러 기능을 수행
  - 예를 들어, 프로그램을 컴파일하고, 파일을 프린터에 보내고, 웹 페이지를 화면에 보여주면서 동시에 이메일을 받고 비디오를 상영
  - 프로세스는 시스템을 동작시키고 동시에 수행되는 많은 활동을 관리
  - 프로세스는 프로세스 상태(process state)를 변화
  - 운영체제는 프로세스 생성, 종료, 일시 정지, 재 시작, 깨우기 등과 같은 프로세스가 서비스할 때 수행하는 다양한 연산 제공

소개

- 프로세스 정의
  - '실행 중인 프로그램'(program in execution)
  - 각 프로세스는 자신의 주소공간을 가지고 있다
    - 텍스트 영역(Text region)
      - 프로세서가 실행하는 코드를 저장하는 영역
    - 데이터 영역(Data region)
      - 변수들을 저장하는 영역과 프로세스가 실행 중에 사용하려고 동적으로 할당 받은 메모리 공간
      - 전역 변수 저장
    - 스택 영역(Stack region)
      - 호출된 프로시저용으로 지역 변수와 명령어들을 저장하는 공간

프로세스 상태 : 프로세스 생명 주기

- 프로세스 생명 주기 동안의 구분된 프로세스 상태(process state)
  - 실행 상태(running state)
    - 프로세스가 프로세서에서 실행 중
  - 준비 상태(ready state)
    - 프로세스가 프로세서에서 실행 가능
  - 블록 상태(block state)
    - 프로세스가 작업을 진행하기에 앞서 특정 이벤트 발생을 대기 (ex. 입출력 완료 이벤트)
- 준비 리스트(ready list)와 블록 리스트(blocked list)
  - 준비 리스트
    - 우선순위 정보를 포함
    - 리스트에서 우선순위가 가장 높은 첫 번째 프로세스가 프로세서를 할당
  - 블록 리스트
    - 블록된 프로세스가 기다리는 이벤트가 발생하는 순서로 블록 해제

프로세스 관리

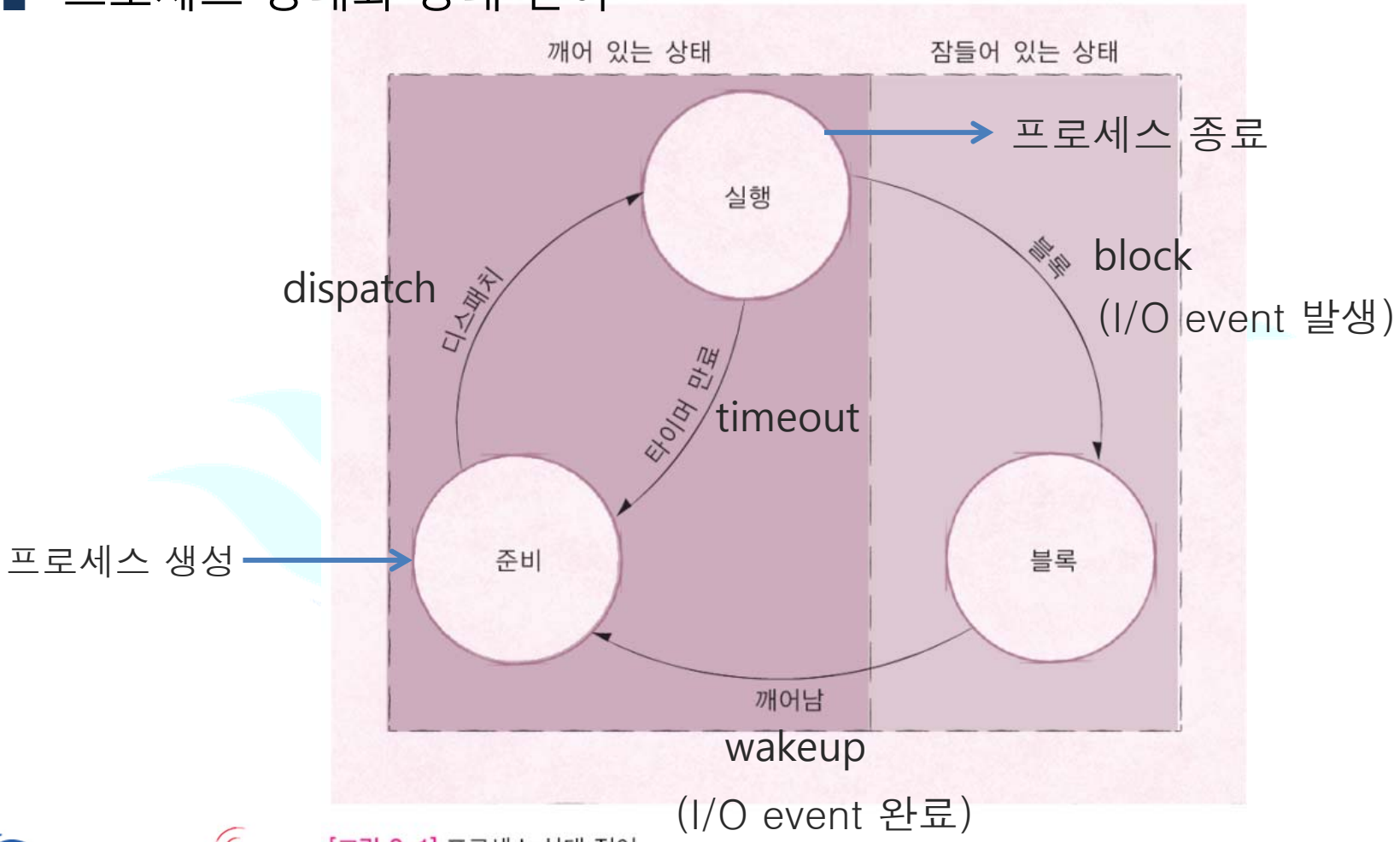
- 운영체제는 프로세스에 기본적인 서비스를 수행
  - 프로세스 생성(create)
  - 프로세스 소멸(destroy)
  - 프로세스 일시 정지(suspend)
  - 프로세스 재 시작(resume)
  - 프로세스 우선순위 변경(change priority)
  - 프로세스 블록(block)
  - 프로세스 깨우기(wake up)
  - 프로세스 디스패치(dispatch)
  - Inter-Processes Communication(IPC)

프로세스 관리

- 프로세스 상태와 상태 전이
  - 사용자가 프로그램을 하나 실행할 때 프로세스가 생성되고 준비리스트에 추가됨
  - 프로세스 상태(process state)
    - 준비 리스트에 있는 첫 번째 프로세스에 프로세서를 할당하는 것을 디스패칭이라 함(dispatching)
    - 운영체제는 한 프로세스가 시스템의 프로세서를 독점하는 일을 방지하려고 하드웨어 인터럽팅 클록(간격 타이머; interrupting clock)을 두어, 프로세스가 특정 시간 간격 또는 퀀텀(quantum) 동안만 실행할 수 있게 함
- 상태 전이(state transitions)
  - 네 가지 상태 전이 정의
    - 프로세서를 할당 받으면 프로세스의 상태가 준비상태에서 실행 상태로 전이
    - 할당 받은 시간이 만료되면 실행 상태에서 준비상태로 전이
    - 프로세스가 블록 되면 실행 상태에서 블록 상태로 전이
    - 대기하던 이벤트가 완료되면 프로세스가 깨어나고 블록 상태에서 준비 상태로 전이

프로세스 관리

- 프로세스 상태와 상태 전이



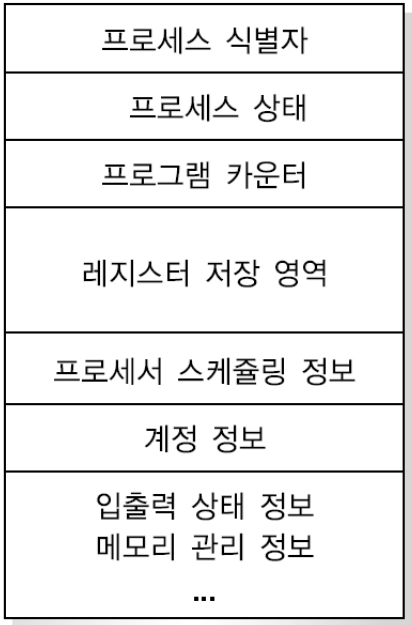
[그림 3-1] 프로세스 상태 전이

프로세스 관리

- 프로세스 제어 블록 (PCB: Process Control Block)
  - PCB는 운영체제가 생성한 프로세스를 관리하는데 필요한 정보 보관
  - 프로세스를 관리하기 위해 유지되는 데이터블록 또는 레코드의 데이터구조
    - Process identification number(PID)
    - 프로그램 상태(process state)
    - 프로그램 카운터(program counter)- 프로세서가 다음에 실행할 명령어를 가르키는 값
    - 스케줄링 우선순위(scheduling priority)
    - 권한(credential)- 프로세스가 접근할 수 있는 자원을 결정
    - 프로세스의 부모 프로세스(parent process)를 가르키는 포인터
    - 해당 프로세스를 생성한 프로세스
    - 프로세스의 자식 프로세스(child process)를 가르키는 포인터
      - 해당 프로세스가 생성한 프로세스
    - 프로세스의 데이터와 명령어가 있는 메모리 위치를 가리키는 포인터
      - 프로세스에 할당된 자원들을 가리키는 포인터
  - PCB 구조는 운영체제의 구현에 따라 다르다
  - PCB는 실행 프로세스가 실행상태에서 빠져나올 때 마지막으로 실행한 프로세서의 레지스터 내용 (실행문맥, execution context)을 저장
  - 운영체제는 프로세스가 상태전이를 할 때, 해당 프로세스의 PCB 안에 있는 상태정보도 갱신

[참고] 프로세스 제어 블록 (PCB, Process Control Block)

- 프로세스는 운영체제 내에서 프로세스 제어 블록이라 표현하며, 작업 제어 블록이라고도 함.
- 프로세스를 관리하기 위해 유지되는 데이터 블록 또는 레코드의 데이터 구조.
- 프로세스 식별자, 프로세스 상태, 프로그램 카운터 등의 정보로 구성.
- 프로세스 생성 시 만들어지고 메인 메모리에 유지, 운영체제에서 한 프로세스의 존재를 정의.



프로세스 제어 블록 (PCB)

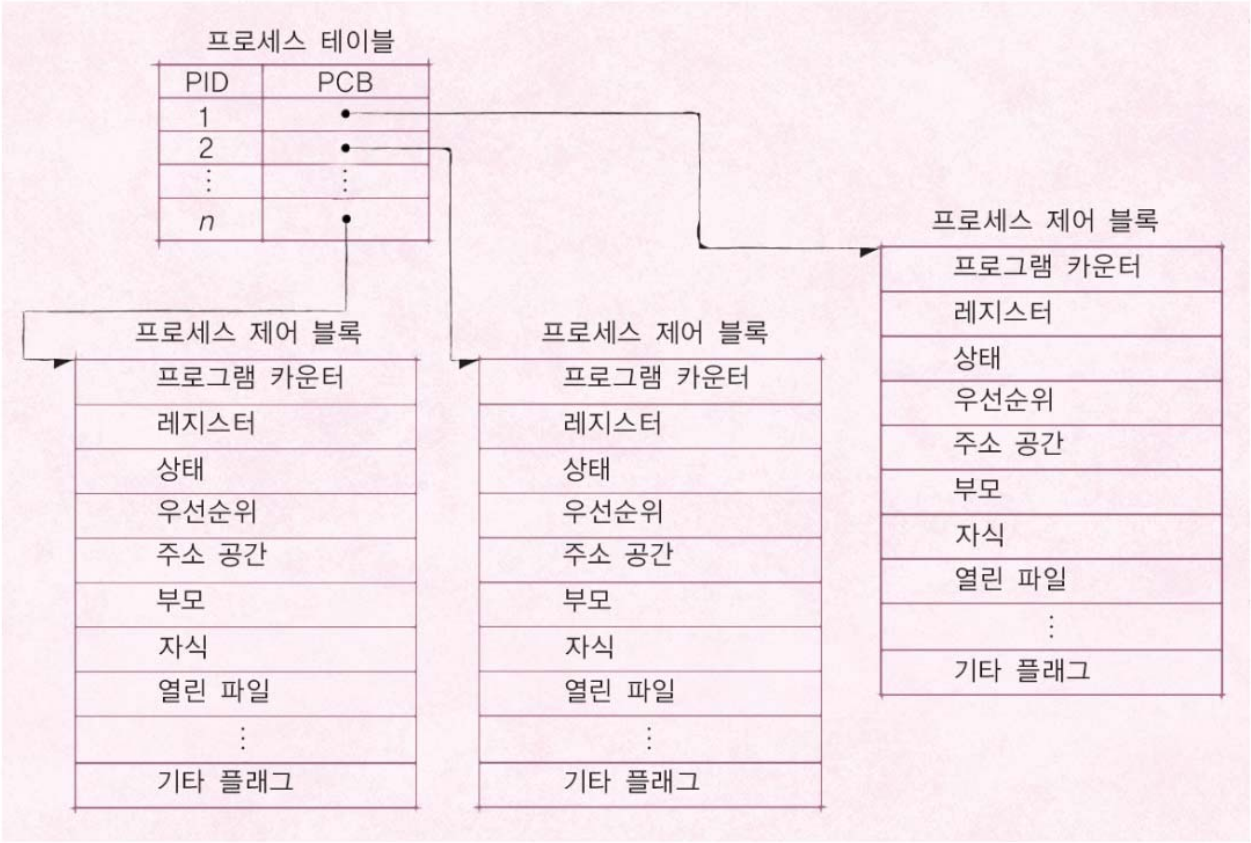
- 프로세스 식별자 : 각 프로세스에 대한 고유 식별자 지정.
- 프로세스 상태 : 생성, 준비, 실행, 대기, 중단 등의 상태 표시.
- 프로그램 카운터 : 프로그램 실행을 위한 다음 명령의 주소 표시.
- 레지스터 저장 영역 : 누산기, 인덱스 레지스터, 범용 레지스터, 조건 코드 등에 관한 정보로 컴퓨터 구조에 따라 수나 형태가 달라짐.
- 프로세서 스케줄링 정보 : 프로세스의 우선순위, 스케줄링 큐에 대한 포인터, 그 외 다른 스케줄 매개변수를 가짐.
- 계정 정보 : 프로세서 사용시간, 실제 사용시간, 사용상한시간, 계정번호, 작업 또는 프로세스 번호 등.
- 입출력 상태 정보 : 특별한 입출력 요구 프로세스에 할당된 입출력장치, 개방된(Opened) 파일의 목록 등.
- 메모리 관리 정보 : 메모리 영역을 정의하는 하한 및 상한 레지스터(경계레지스터) 또는 페이지 테이블 정보.

프로세스 관리

- 프로세스 제어 블록
  - 프로세스 테이블(process table)
    - 운영체제는 각 프로세스의 PCB를 가리키는 포인터를 시스템 전체 혹은 사용자별 프로세스 테이블에 유지
    - PCB에 빠르게 접근할 수 있도록 함
    - 프로세스가 종료하면, 운영체제는 프로세스의 메모리와 기타 자원을 해제해 다른 프로세스가 사용할 수 있게 하고, 프로세스 테이블에서 해당 프로세스를 제거

프로세스 관리

- 프로세스 제어 블록



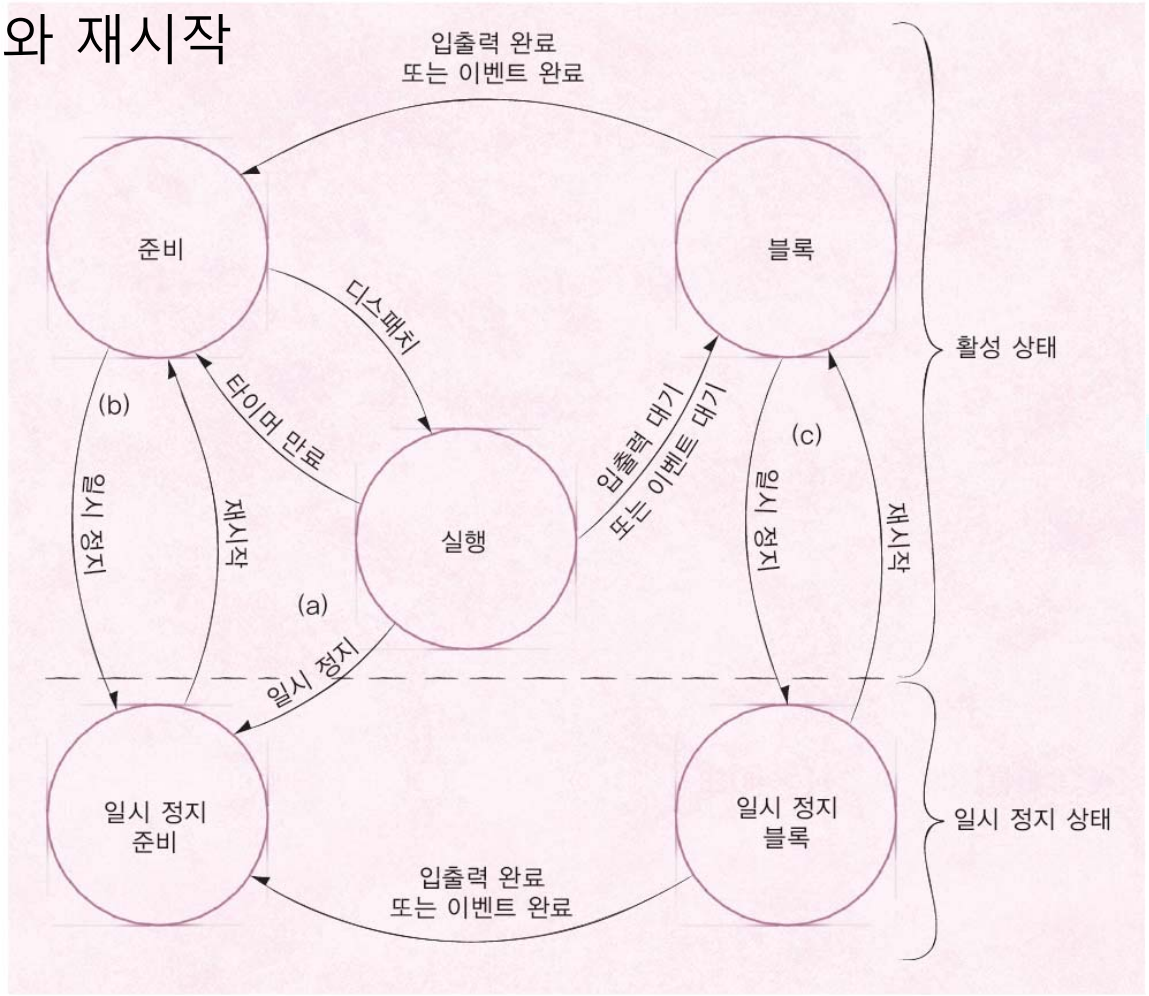
[그림 3-2] 프로세스 테이블과 프로세스 제어 블록

프로세스 관리

- 일시정지와 재시작
  - 프로세스 일시정지
    - 아직 소멸되지는 않고 프로세서를 차지하려고 하는 경쟁 대열에서 무기한으로 배제
    - 악성 코드 실행과 같은 보안 위협 요인을 추적하거나 디버깅할 때 유용
      - 운영자나 사용자가 특정 프로세스의 실행결과가 의심스럽다고 느낄 때 일시정지 사용가능
    - 일시정지는 해당 프로세스 혹은 다른 프로세스에 의해 발생
    - 일시정지 블록 상태 프로세스는 다른 프로세스에 의해 재시작 가능
    - 일시정지 상태
      - 일시 정지 준비(suspendedready)
      - 일시 정지 블록(suspendedblocked)

프로세스 관리

- 일시정지와 재시작



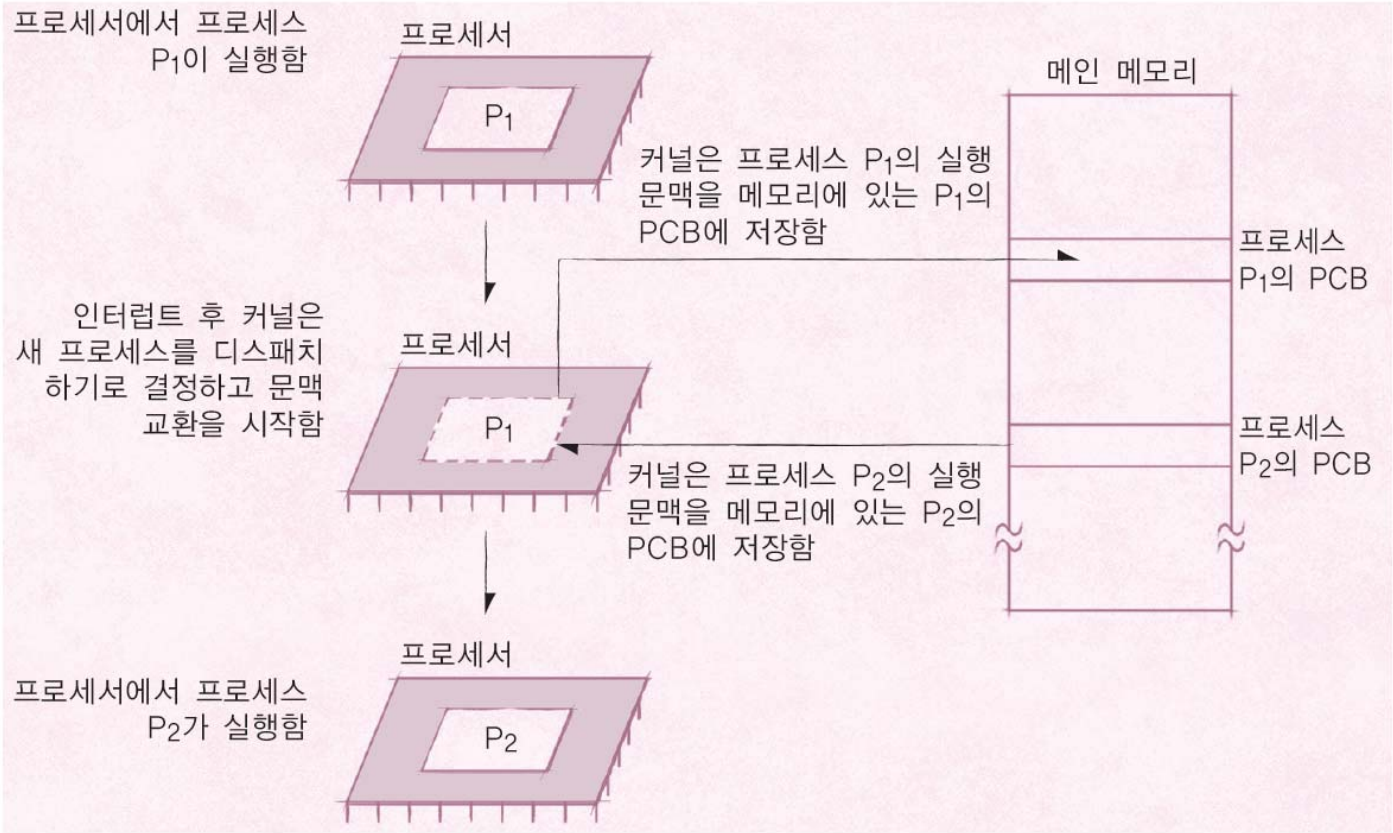
[그림 3-5] 일시 정지와 재시작을 포함한 프로세스 상태 전이도

프로세스 관리

- 문맥 교환(context switch)
  - 운영체제에 의해 실행 중인 프로세스를 멈추고 준비 상태에 있던 다른 프로세스를 실행
  - 실행 중인 프로세스의 실행 문맥을 해당 프로세스의 PCB에 저장
  - 실행할 준비 상태 프로세스의 이전 실행 문맥을 PCB에서 읽어 로드
  - 문맥 교환이 일어나는 동안 프로세서는 '의미 있는'작업을 전혀 수행 하지 못함
    (프로세서는 프로세스들의 명령어를 실행 못함)
    - 운영체제는 문맥 교환에 드는 시간을 최소화 해야 함

프로세스 관리

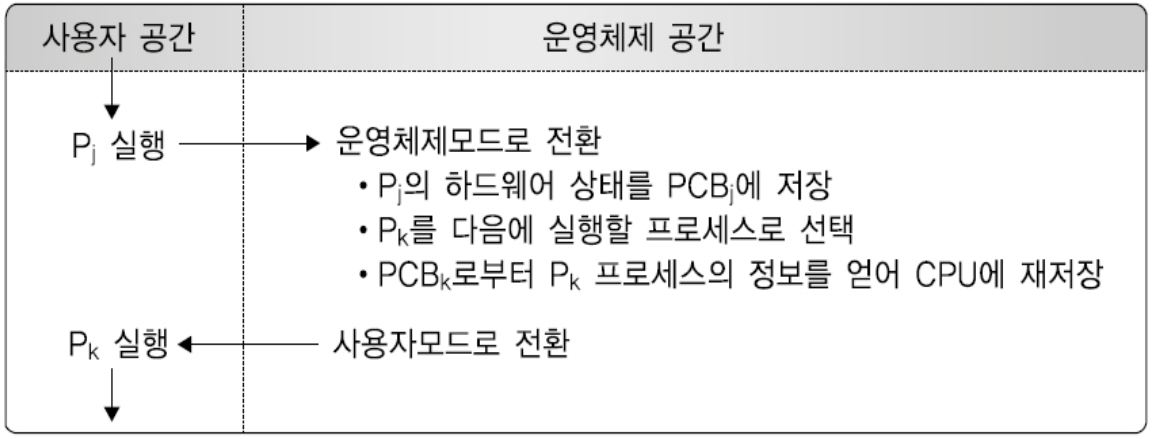
- 문맥 교환(context switch)



[그림 3-6] 문맥 교환

프로세스 관리

- 문맥 교환 (정리)
  - 프로세스를 다른 프로세스로 교환하기 위해 이전 프로세스의 상태 레지스터 내용을 보관하고 다른 프로세스의 레지스터를 적재하는 일련의 과정.
    - 프로세스가 "준비→실행", "실행→준비", "실행→블록"상태로 변할 때 발생.
  - 오버헤드가 발생하며 오버헤드는 메모리 속도, 레지스터 수, 특수 명령어의 존재에 따라 다르므로 시스템마다 다름.



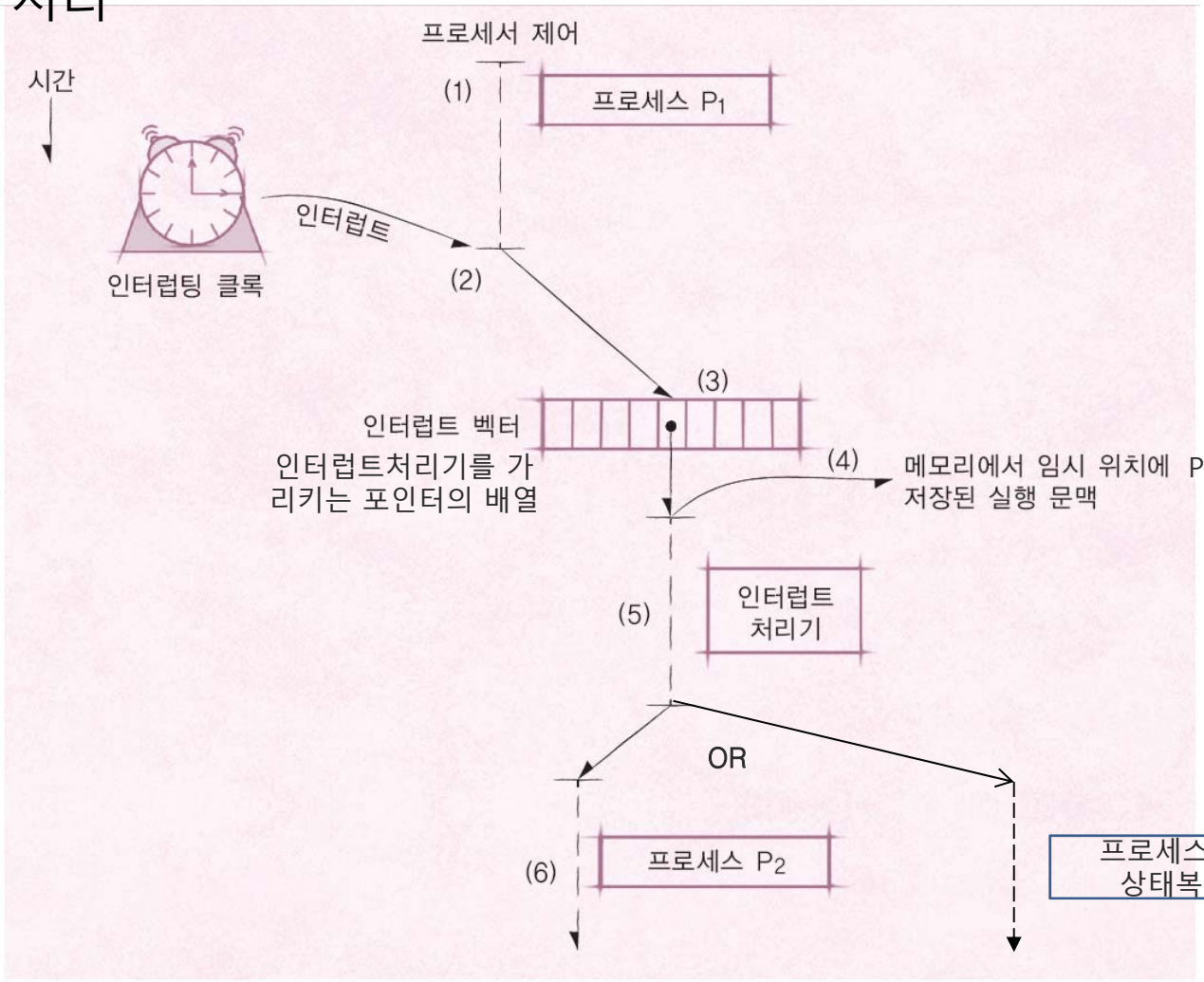
문맥 교환 과정

인터럽트

- 인터럽트(interrupt)
  - 인터럽트는 소프트웨어가 하드웨어로부터 오는 신호에 반응할 수 있게 함
    - 프로세서는 프로세스의 명령어를 실행한 결과로 인터럽트를 발생
      - 트랩(trap)
      - 프로세스의 작동과 동기(synchronous)
      - ex. 0을 나누거나 보호되는 메모리 위치를 참조하려고 하는 경우
    - 프로세서의 현재 명령어와 관련 없는 이벤트에 의해서도 인터럽트 발생
      - 프로세스 작동과 비동기(asynchronous)
      - ex. 사용자가 키보드를 누르거나 마우스를 움직이는 경우
    - 적은 오버헤드
- 폴링(polling)
  - 인터럽트의 대안
  - 프로세서가 각 장치의 상태를 반복적으로 확인
  - 컴퓨터 시스템의 복잡도가 증가할수록 오버헤드 증가

인터럽트

- 인터럽트 처리



[그림 3-7] 인터럽트 처리

프로세스 간 통신

- 신호(signal)
  - 프로세스에 이벤트가 발생했음을 알리는 소프트웨어 인터럽트
    - 프로세스들이 다른 프로세스와 교환할 데이터를 명시하지 않음
    - 운영체제는 신호가 발생할 때, 해당 신호를 받을 프로세스와 해당 프로세스가 신호에 반응할 방법을 결정함
  - 잡음(catch), 무시(ignore), 마스킹(masking)
    - 프로세스는 신호를 전달할 때 운영체제가 호출하는 루틴을 정함으로써 신호를 잡음
    - 프로세스는 신호 처리를 위한 운영체제의 기본 동작에 의존함으로써 신호 무시
      - 공통적인 기본동작은 중단, 메모리덤프, 무시, 일시정지/재시작 등
    - 프로세스가 특정 유형의 신호를 마스크하면, 해당 유형의 신호를 전달하지 않음