EliceAcademy

핵심 운영체제

03 메모리

목차

운영체제가 메모리를 관리하는 방법에 대해서 알아봅니다.

01

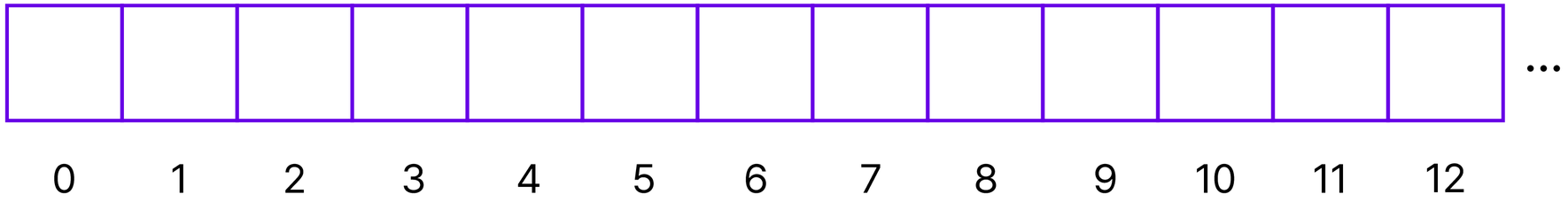
메모리 관리의 개념

메모리 관리의 개념

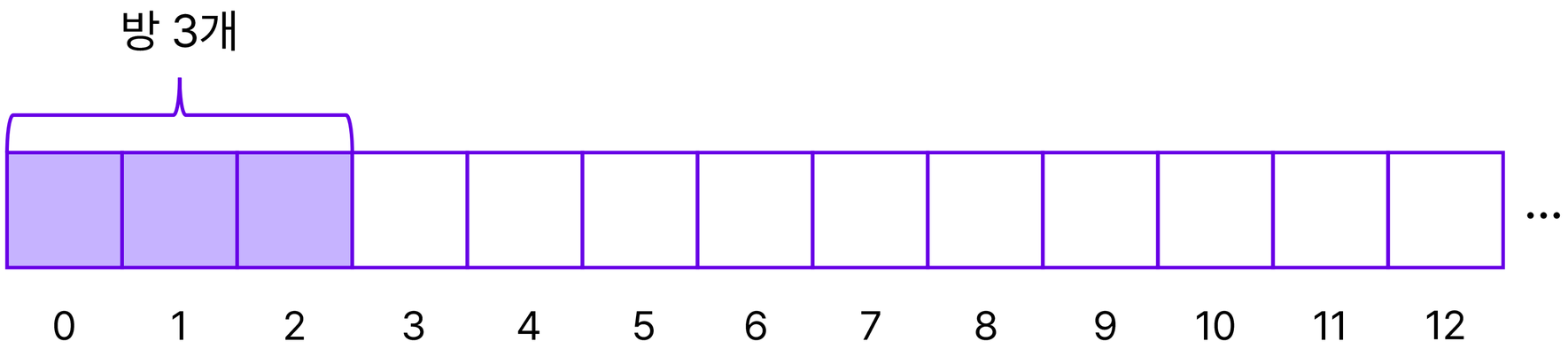
프로그램들이 잘 실행될 수 있도록 메모리를 할당하고, 해제하는 작업

- 각 프로그램의 리소스를 보호해야 한다.
- 여러 프로세스가 동시에 실행될 수 있도록 메모리 공간을 제공해야 한다.

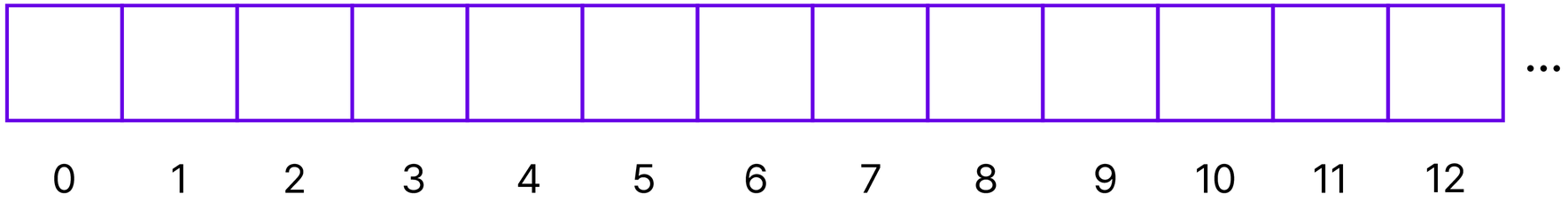
메모리의 주소 (memory address)



메모리 할당 (memory allocation)



메모리 해제 (memory release)



연속 할당과 불연속 할당

연속 할당
(continuous allocation)

1	1	1	4
4	4	4	7
7	7	7	7

한 손님에게는 연속된 방번호로 방을 주기

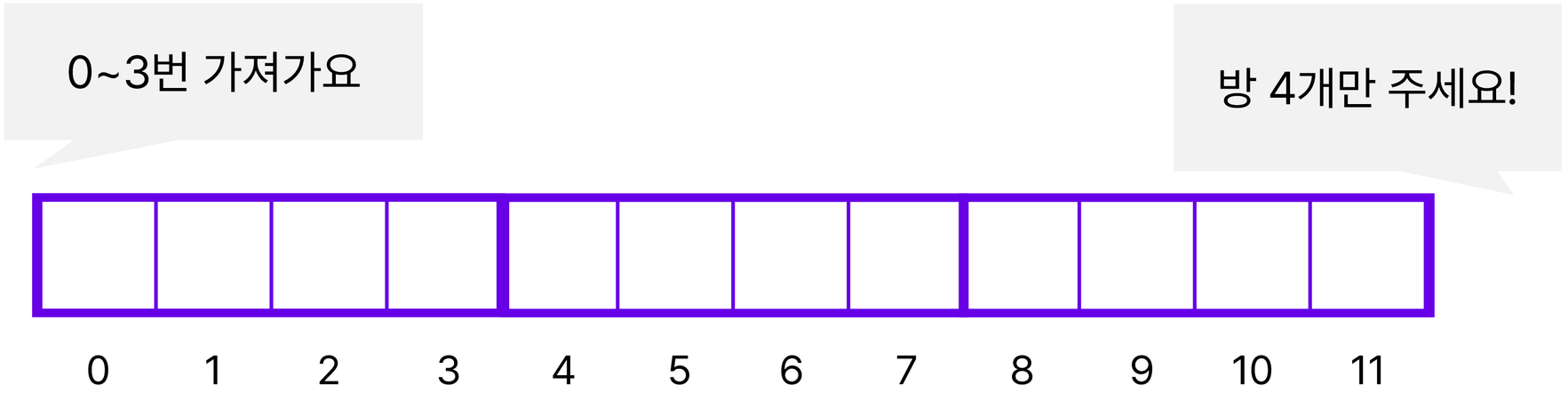
불연속 할당
(Noncontiguous allocation)

1	2	2	3
2	1	1	7
3	7	7	1

방 번호에 상관없이 제공하기

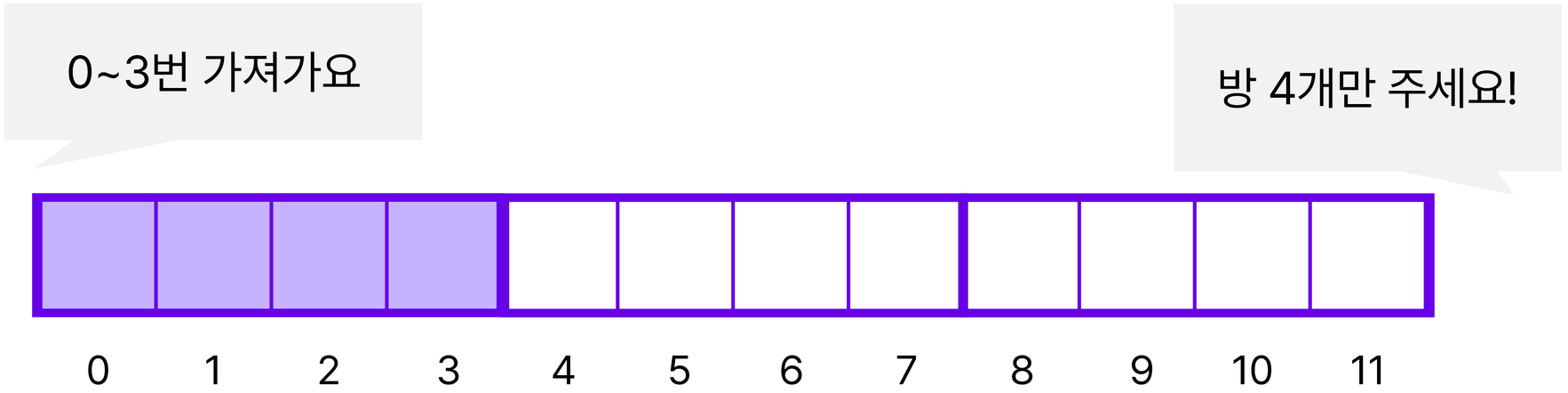
고정 분할 방식

미리 방을 고정된 길이로 나눠 놓고 손님에게 주는 방식



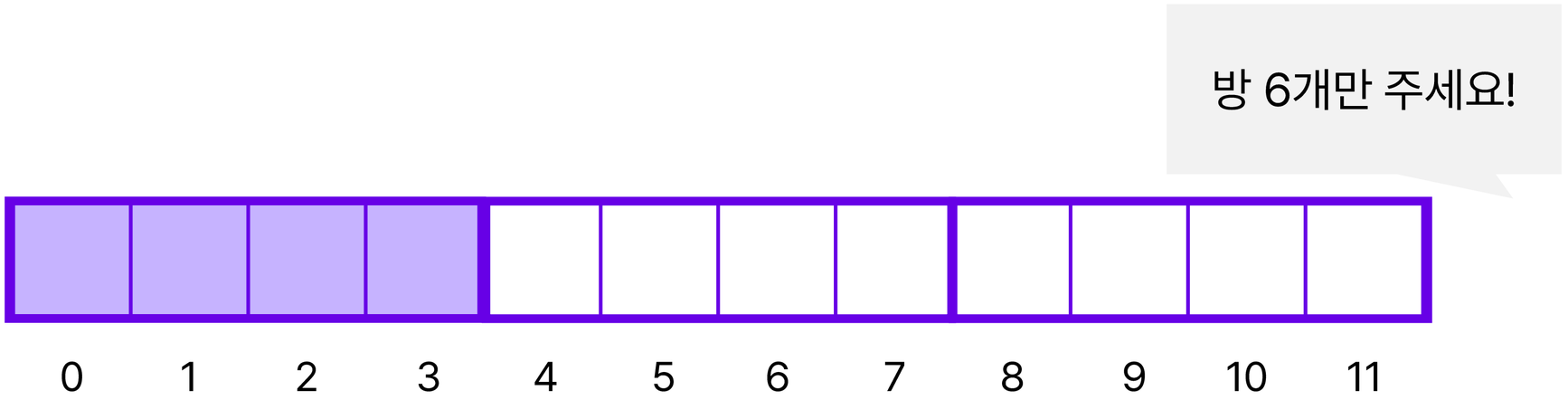
고정 분할 방식

미리 방을 고정된 길이로 나눠 놓고 손님에게 주는 방식



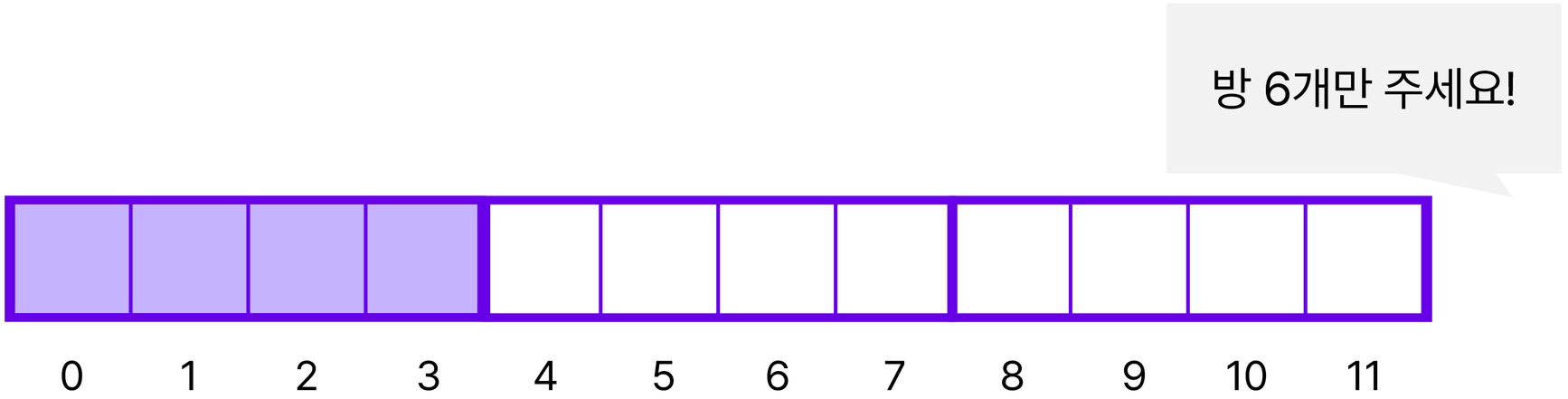
고정 분할 방식

미리 방을 고정된 길이로 나눠 놓고 손님에게 주는 방식



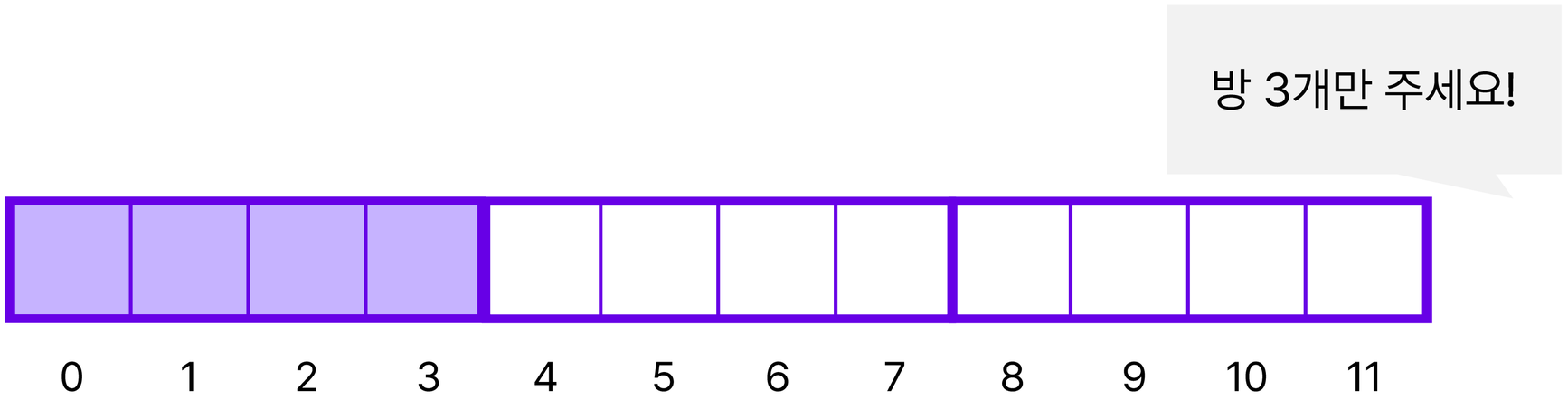
외부 단편화

방은 충분히 남지만 규칙 때문에 방을 줄 수 없는 상태



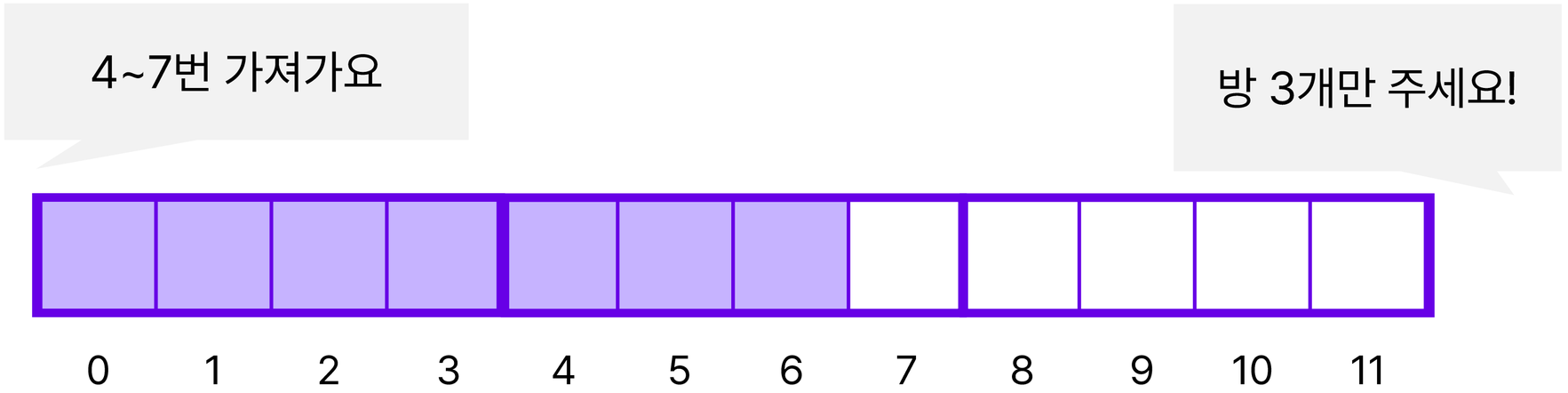
고정 분할 방식

미리 방을 고정된 길이로 나눠 놓고 손님에게 주는 방식



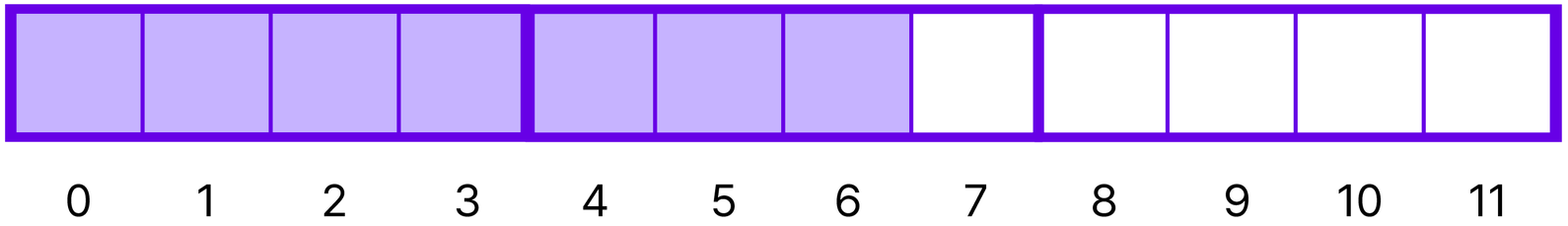
고정 분할 방식

미리 방을 고정된 길이로 나눠 놓고 손님에게 주는 방식



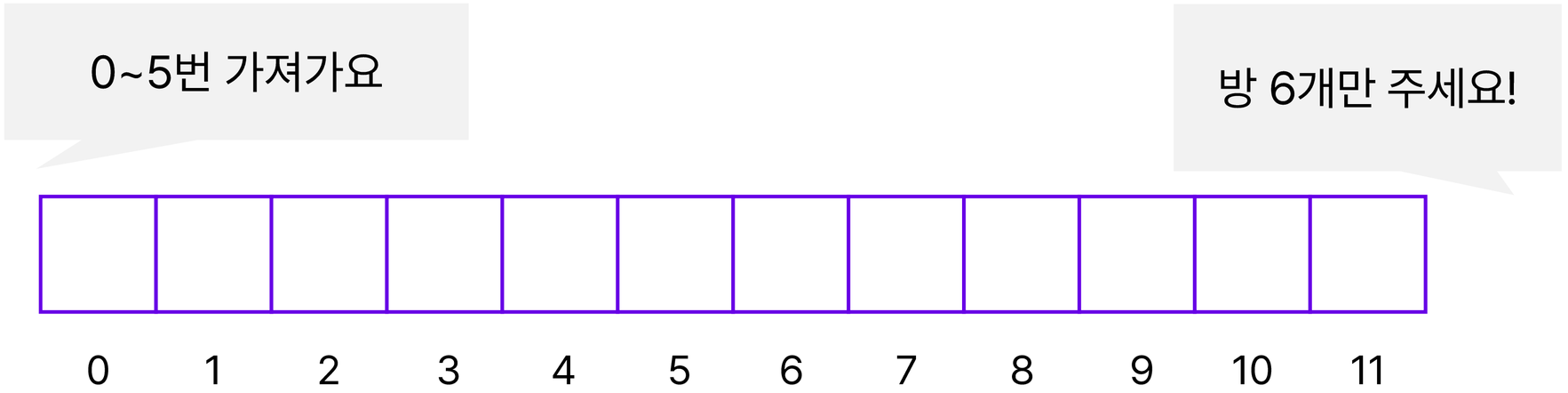
내부 단편화

할당은 해주었지만 사용하지 않고 남는 공간이 발생



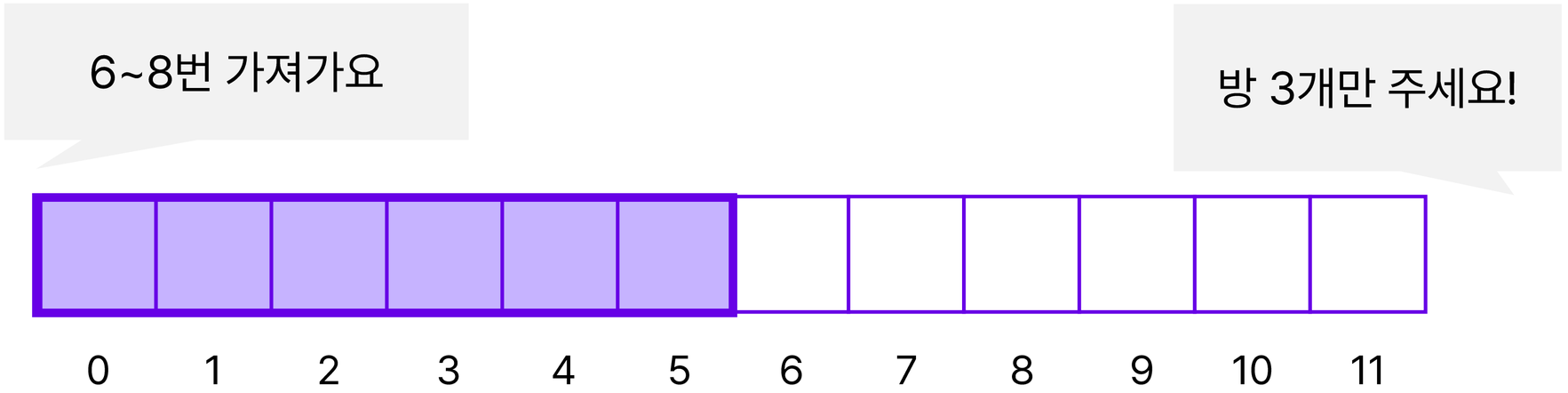
가변 분할 방식

손님마다 다른 방 길이를 받도록 하는 방식



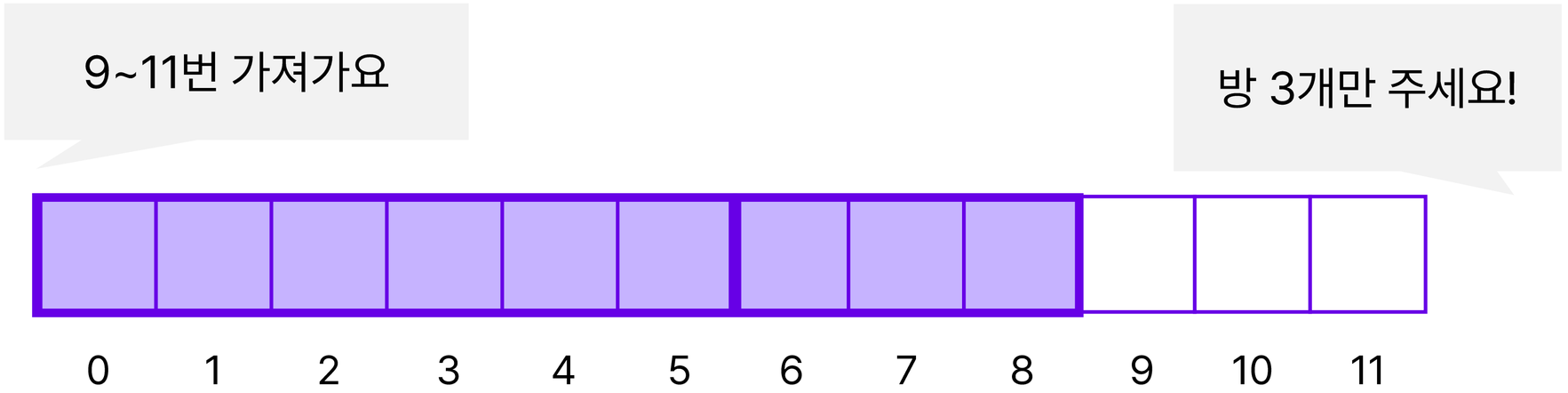
가변 분할 방식

손님마다 다른 방 길이를 받도록 하는 방식



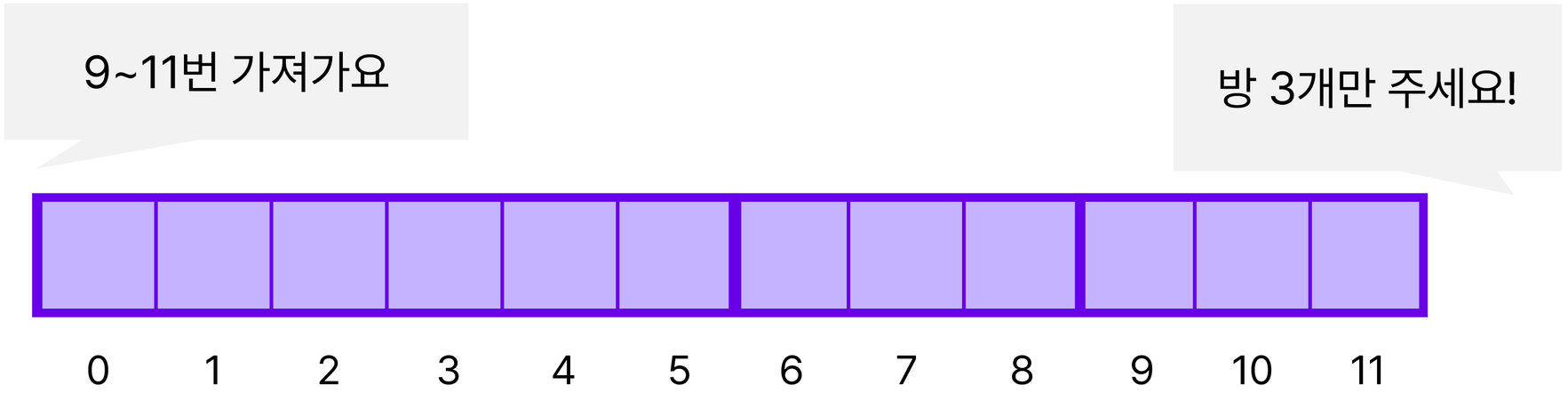
가변 분할 방식

손님마다 다른 방 길이를 받도록 하는 방식



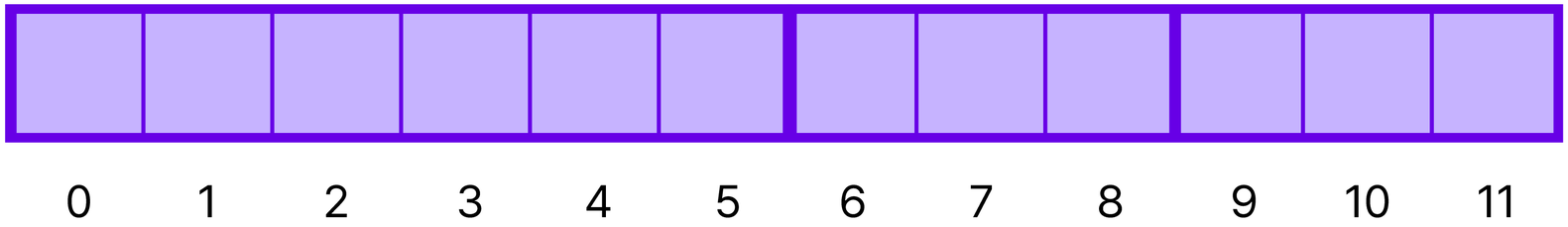
가변 분할 방식

손님마다 다른 방 길이를 받도록 하는 방식



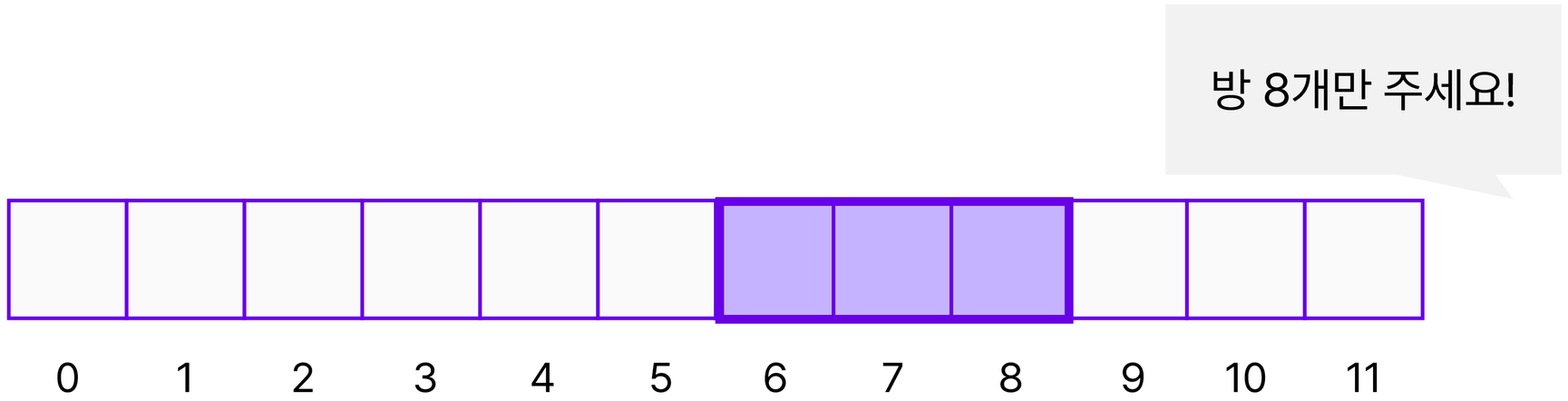
가변 분할 방식

내부 단편화가 발생하지 않는다



가변 분할 방식

외부 단편화는 여전히 발생한다

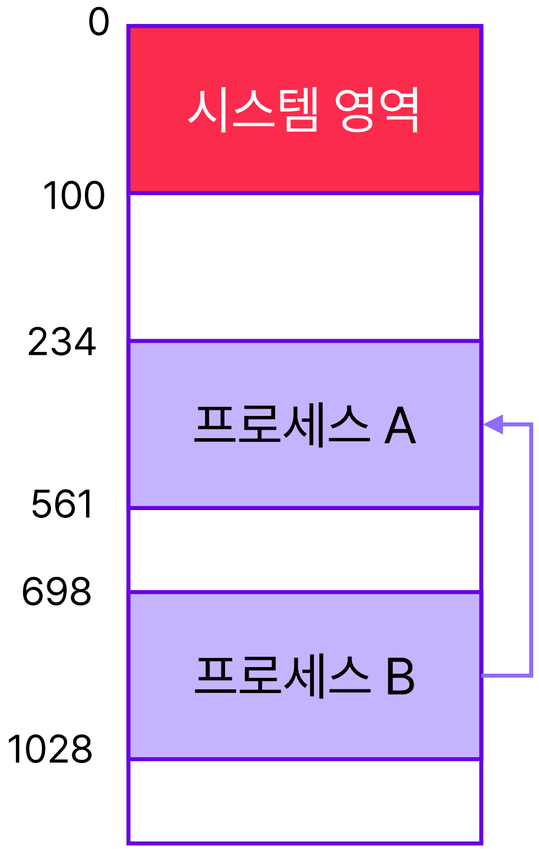


EliceAcademy

02

가상 메모리

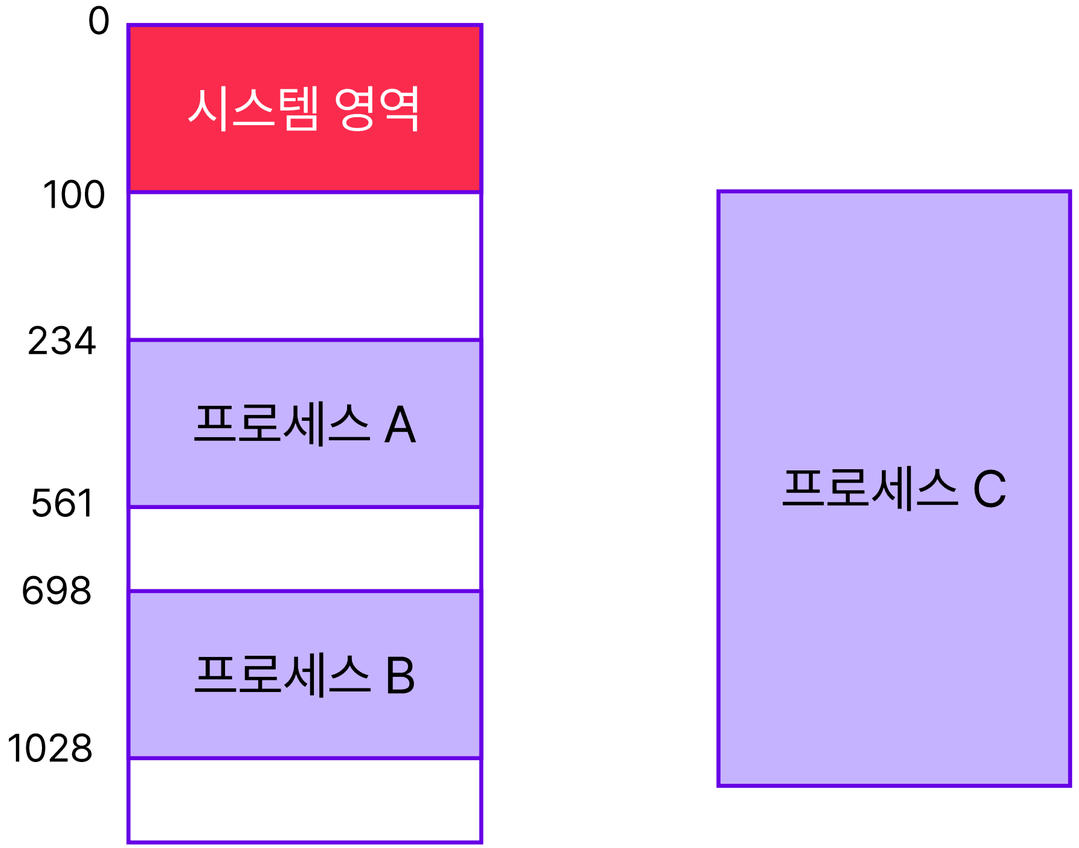
가상 메모리가 필요한 이유



잘못된 주소에 데이터를 삽입

EliceAcademy

가상 메모리가 필요한 이유



가상 메모리의 개념

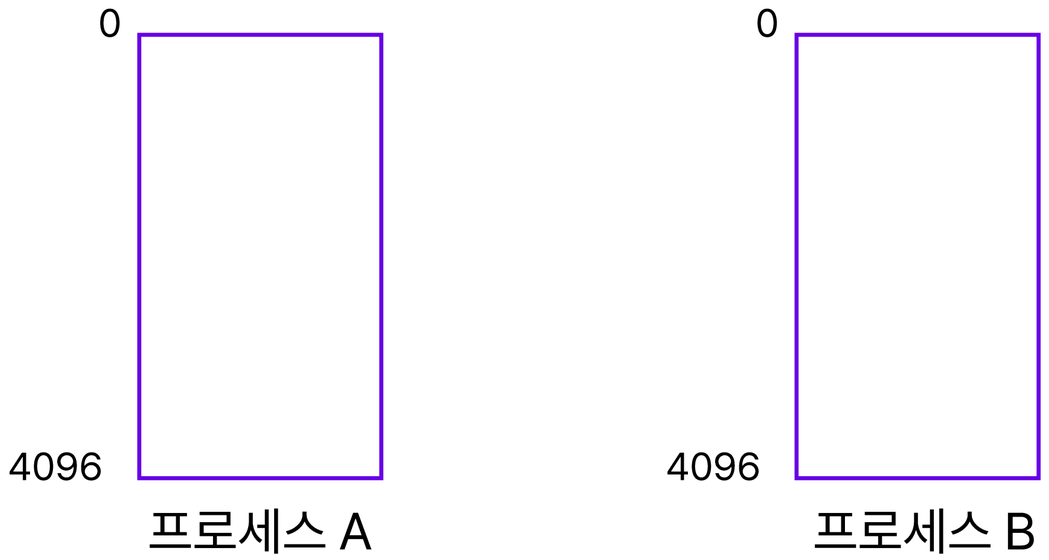
실제 물리적 주소가 아닌 가상의 주소인

논리적 주소를 프로세스가 사용

- 모든 프로세스가 고정된 주소를 가질 수 있도록 한다.
- 실제 물리적인 주소는 운영체제만 접근할 수 있어 리소스를 보호한다.
- 보조 기억 장치 일부를 메모리처럼 사용할 수 있도록 한다.

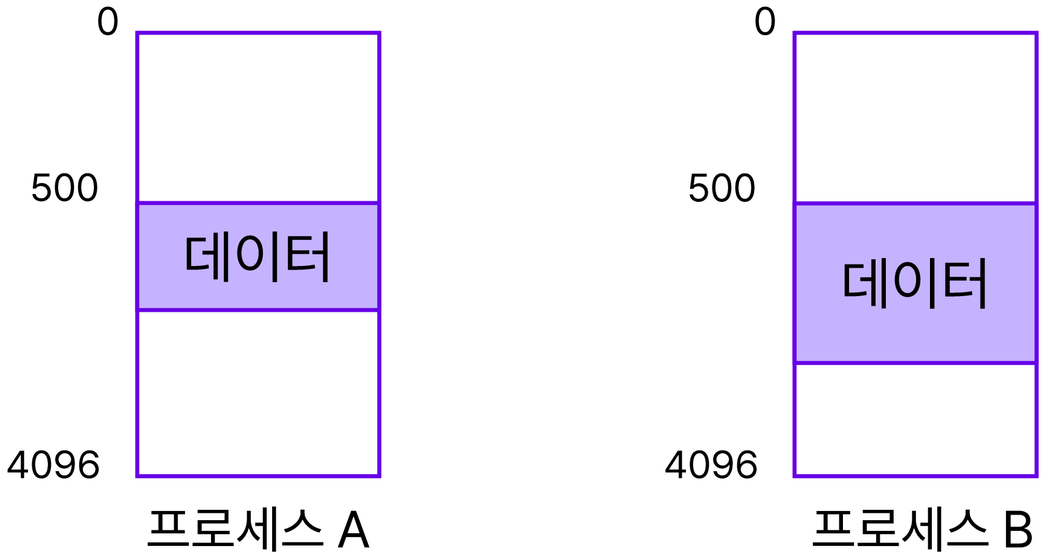
가상 메모리의 장점

모든 프로세스가 고정된 주소를 가질 수 있도록 한다.



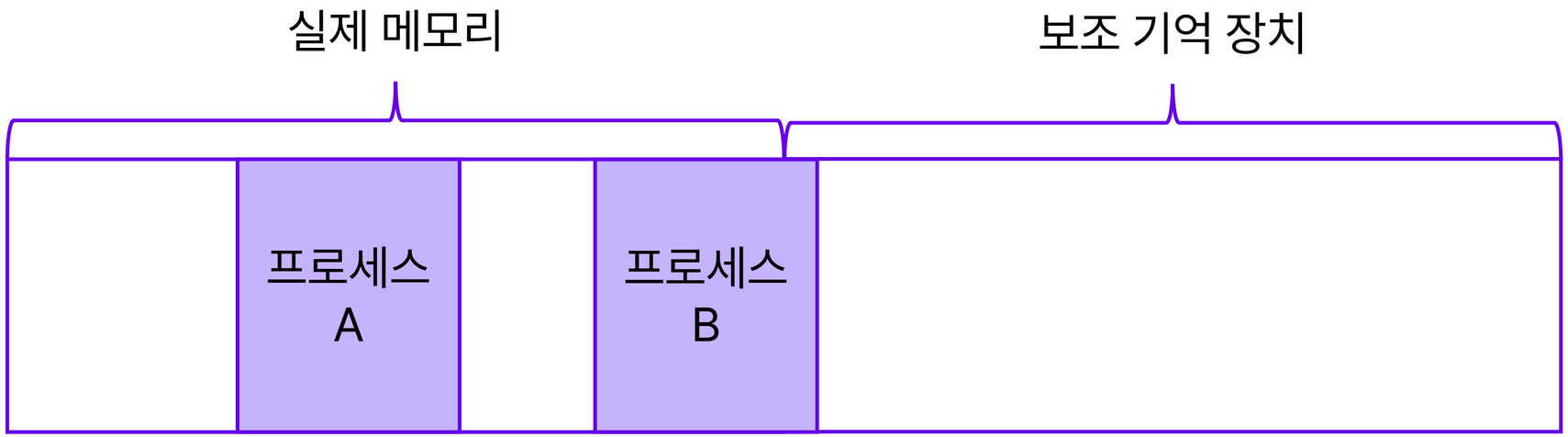
가상 메모리의 장점

실제 물리적인 주소는 운영체제만 접근할 수 있어 리소스를 보호한다



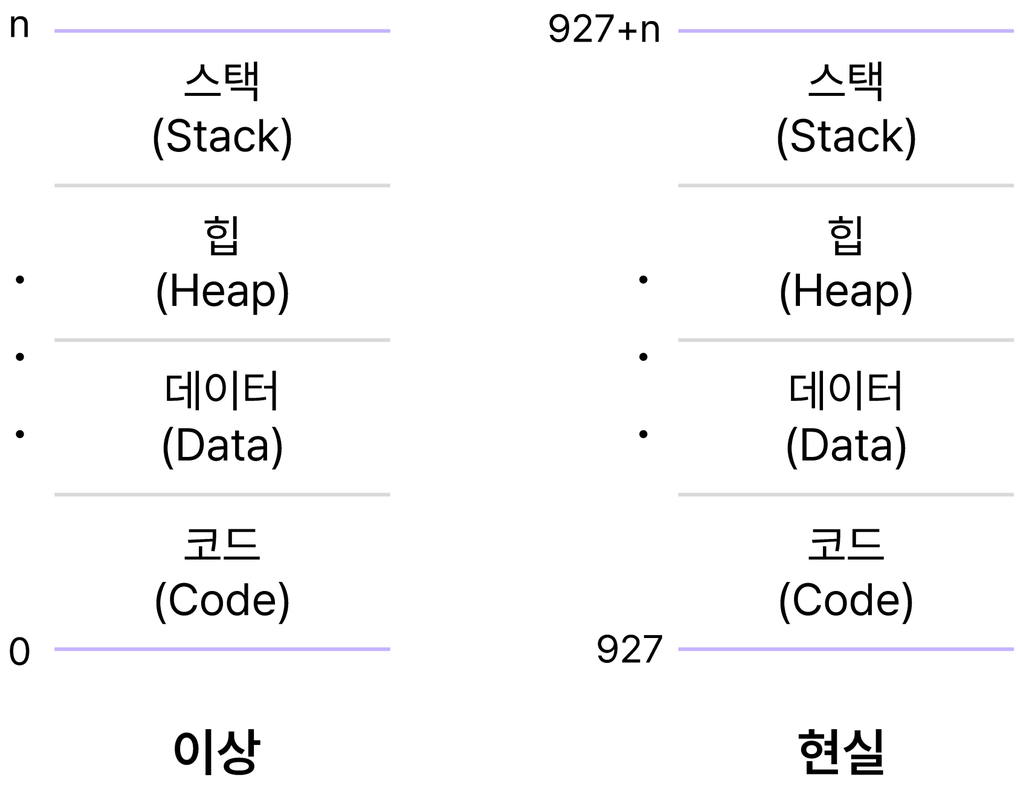
가상 메모리의 장점

보조 기억 장치 일부를 메모리처럼 사용할 수 있도록 한다

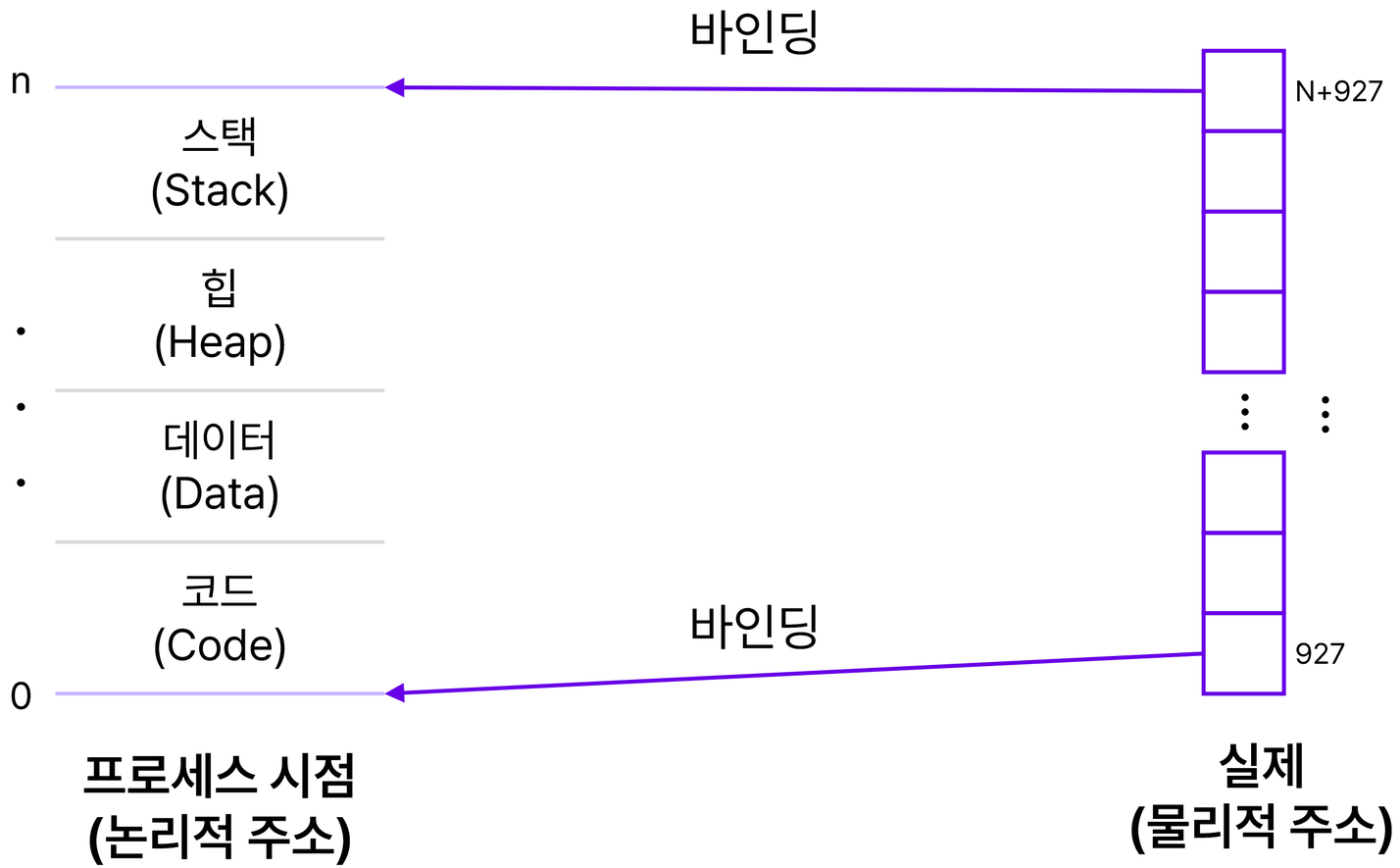


EliceAcademy

프로세스의 메모리 구조

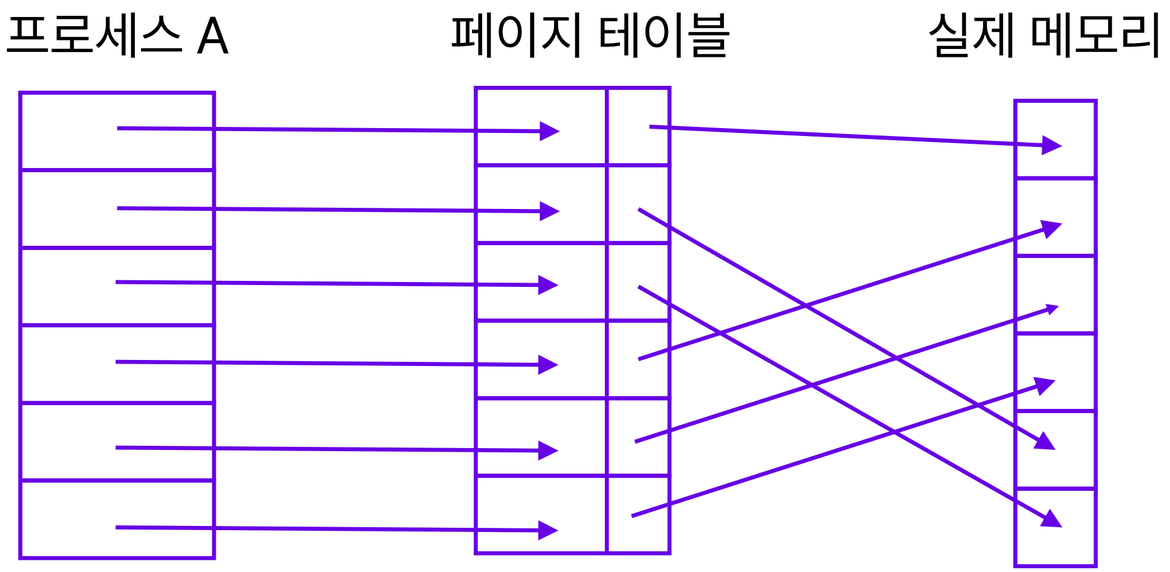


논리적 주소와 물리적 주소



가상 메모리의 원리

가상 메모리의 페이지와 물리 메모리를 매핑하는 표를 유지



EliceAcademy

03

페이징

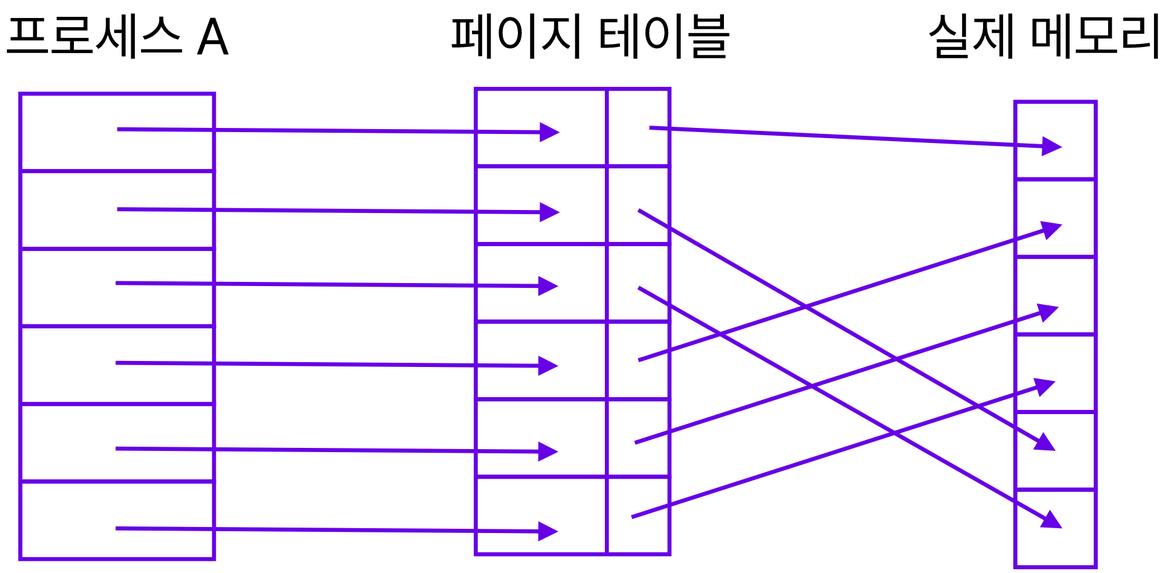
페이지(Page)

물리적 메모리와 가상 주소를 연결하는 단위

- 가상 주소와 물리적 주소를 매핑하는 단위
- 일반적으로 4KB 또는 8KB의 크기
- 메모리는 이 페이지 단위로 나누고 사용

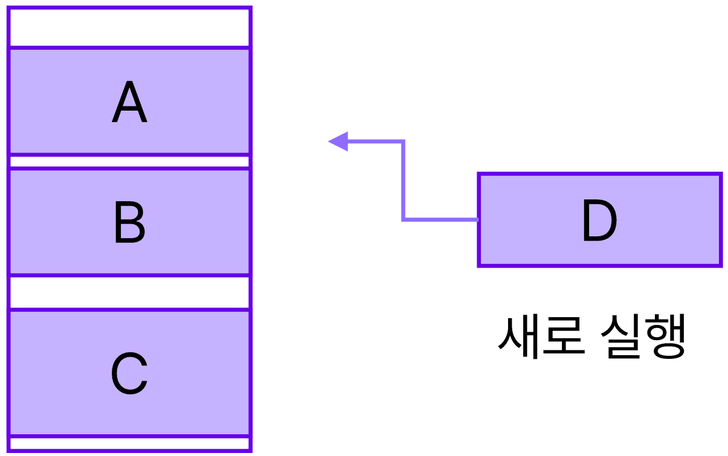
페이지 테이블 (Page table)

가상 메모리의 페이지와 물리 메모리를 매핑하는 표



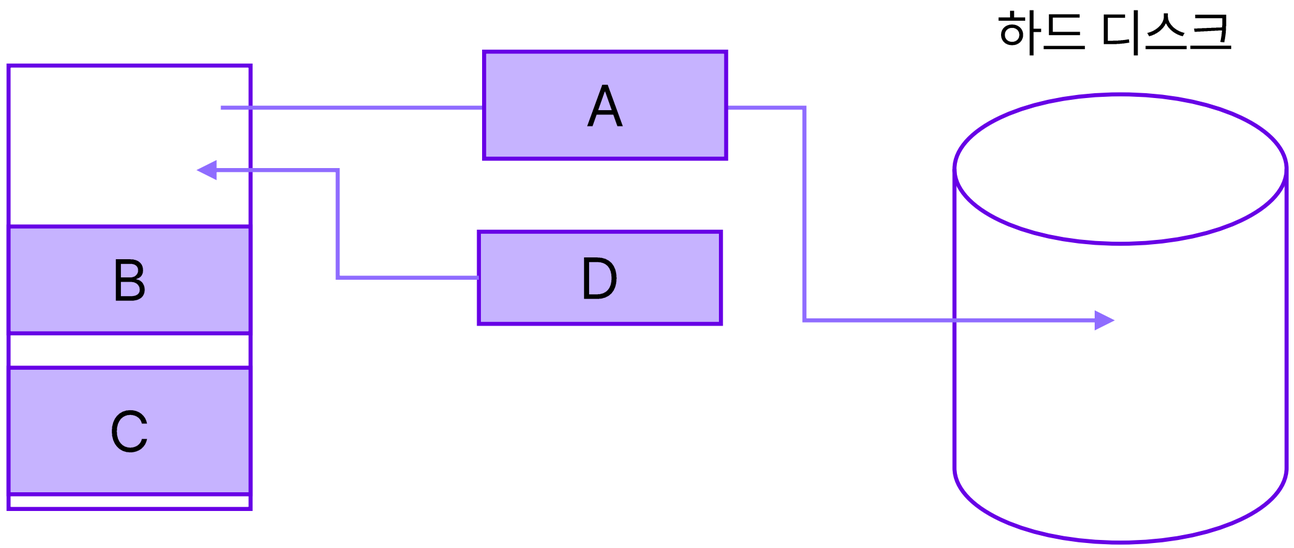
스와핑(swapping)

안 쓰는 프로세스는 잠시 하드 디스크에 넣어두기



스와핑(swapping)

안 쓰는 프로세스는 잠시 하드 디스크에 넣어두기



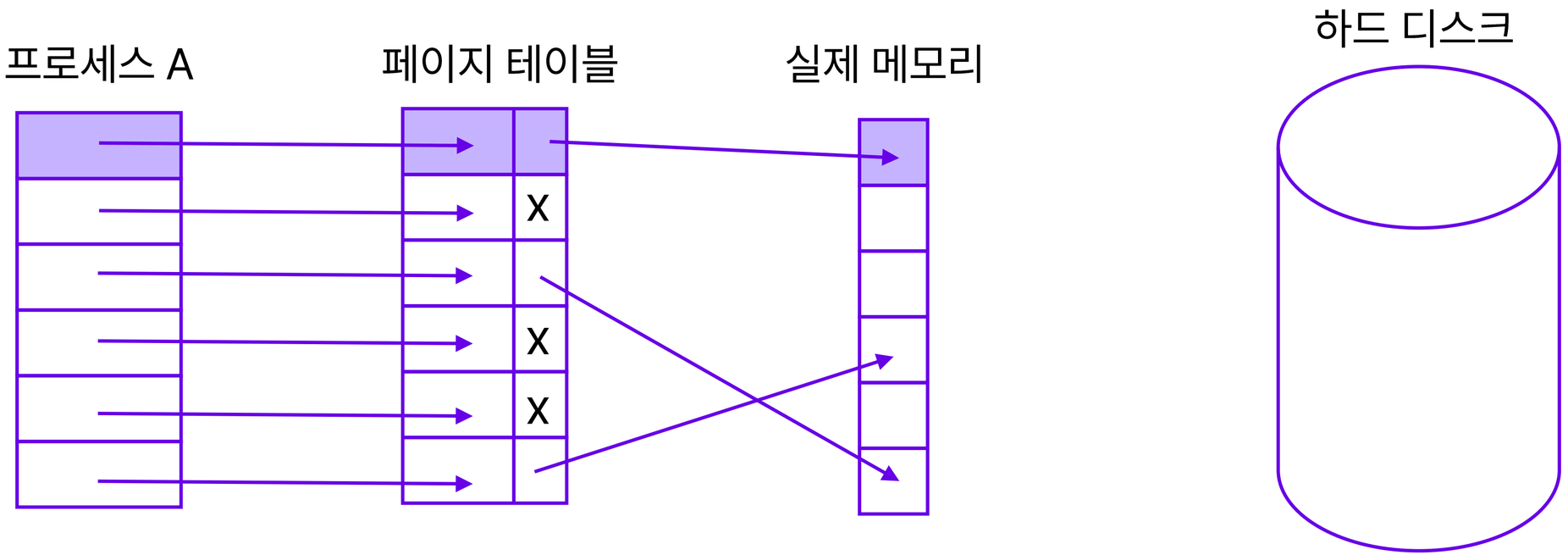
프로세스 스와핑(swapping)

안 쓰는 프로세스는 잠시 하드 디스크에 넣어두기

- 프로세스 단위로 메모리와 하드 디스크를 전환하는 것은 비효율적이고 단편화가 발생할 수 있음
- 최근에는 사용하지 않는 방법
- 하지만 더 많은 프로세스 실행을 위해서는 사용하지 않는 프로세스를 하드 디스크에 넣어둘 필요가 있음

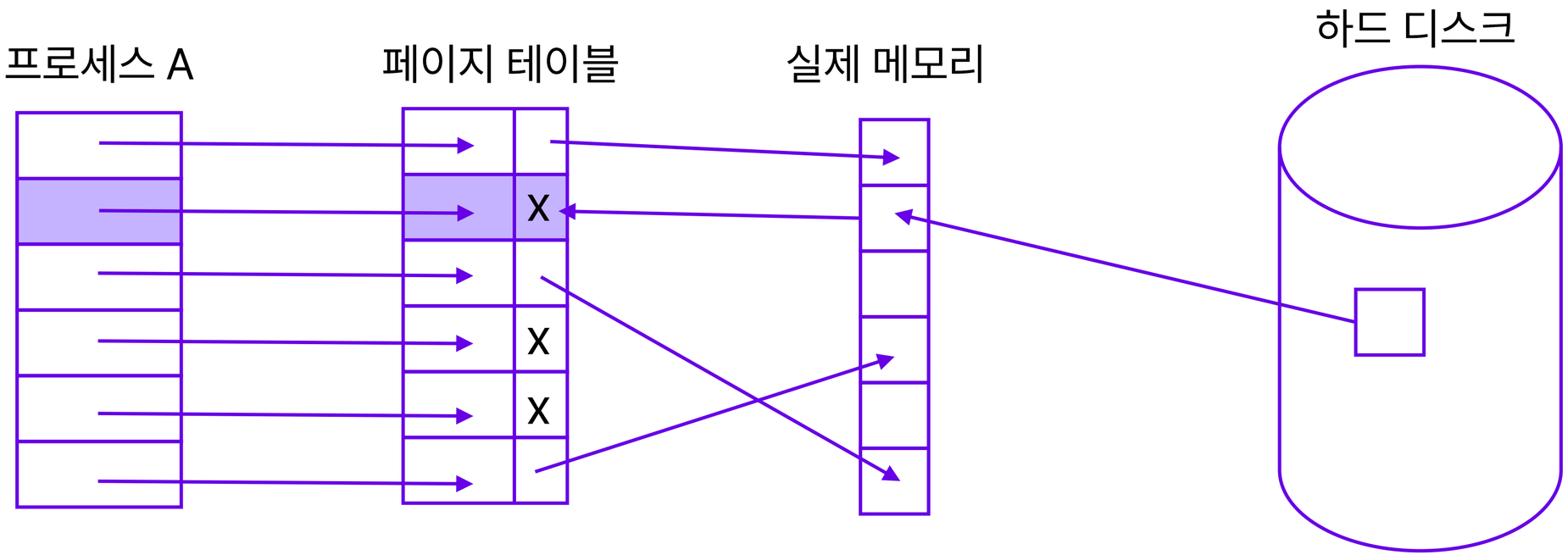
요구 페이징(demand paging)

페이지 단위로 스와핑하여 당장 필요한 페이지만 메모리에 남겨두는 방법

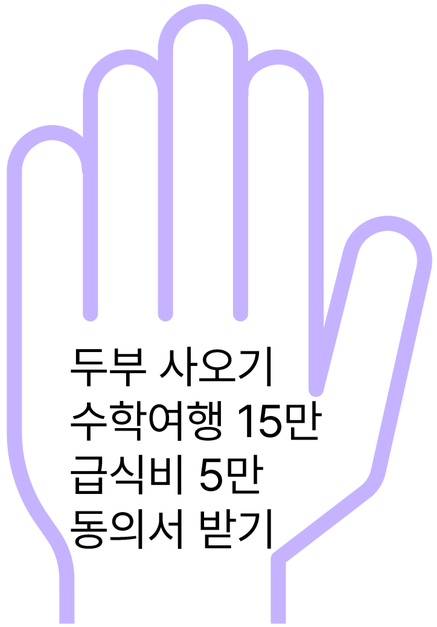


요구 페이징(demand paging)

페이지 단위로 스와핑하여 당장 필요한 페이지만 메모리에 남겨두는 방법



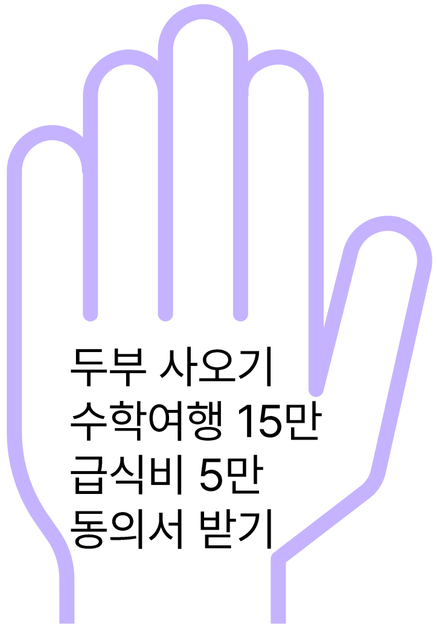
캐시 (cache)



손바닥에 메모하기

- 빠르게 내용을 확인할 수 있음
- 볼펜만 있다면 빠르게 쓰고 지울 수 있음
- 손바닥에 들어갈 정도의 내용만 적을 수 있음

캐시 (cache)

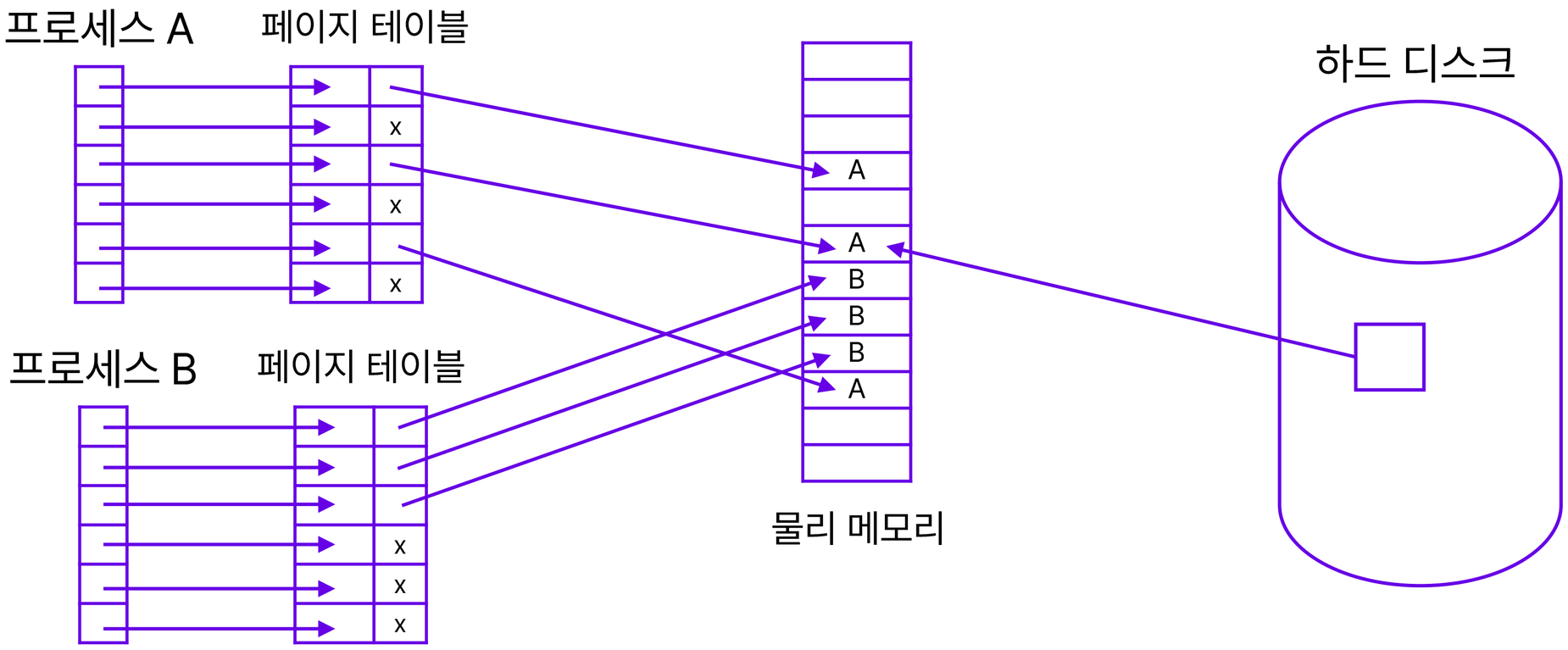


데이터나 값을 미리 복사해놓는 임시 장소

- 계산이 오래 걸리는 데이터를 자주 봐야 하는 경우
- 미리 값을 복사해둬서 시간을 절약
- 모든 데이터를 캐시에 넣을 수는 없음.

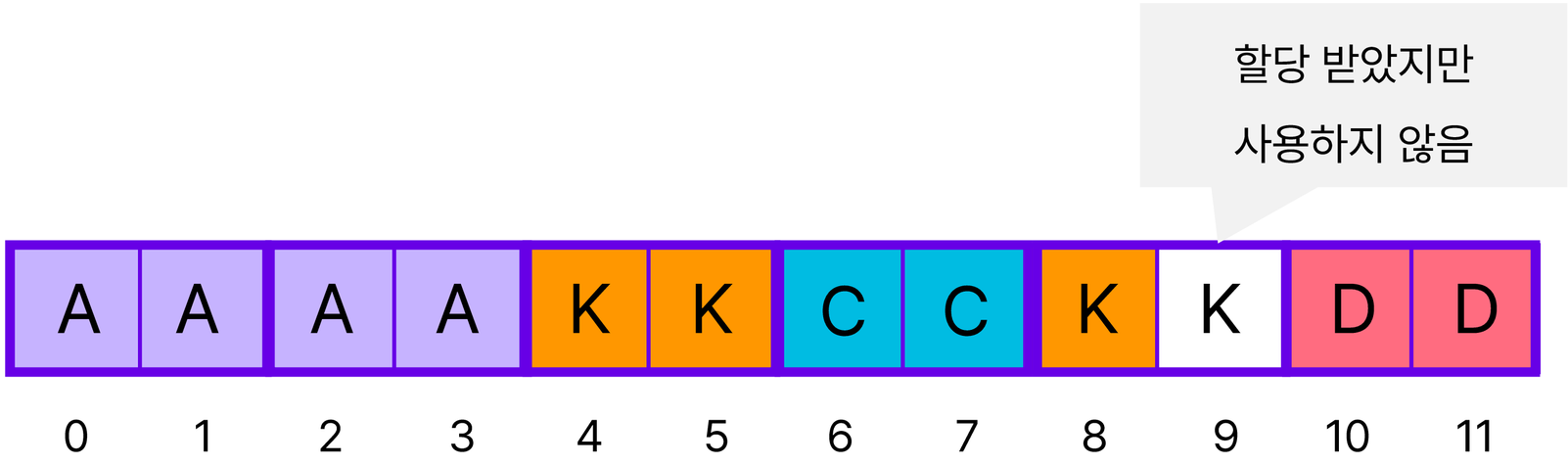
요구 페이징(demand paging)

실제 물리 메모리를 하드 디스크의 캐시로 사용하는 기법



페이징 기법의 문제점

일정한 크기로 자르기 때문에 내부 단편화가 발생



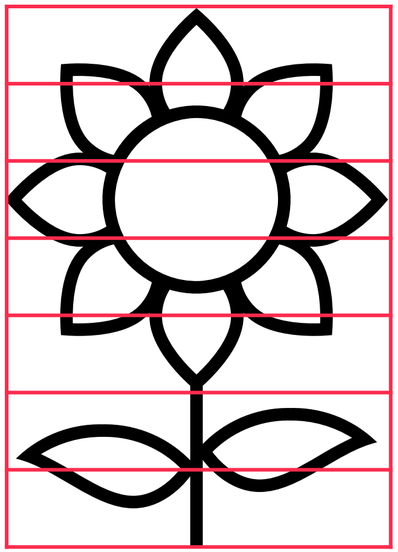
EliceAcademy

04

세그멘테이션

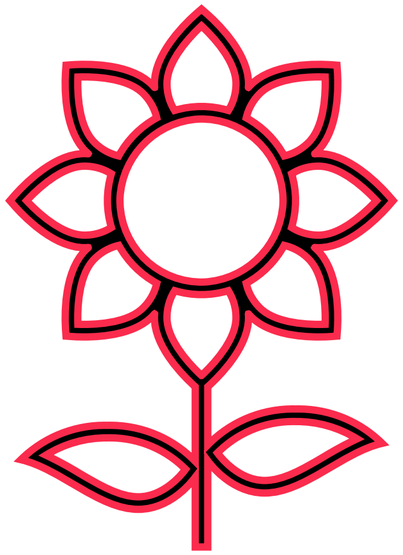
페이징 기법의 문제점

부위에 상관없이 일정한 크기로 자르는 방법



세그멘테이션

논리적으로 같은 역할을 하는 부분끼리 자르는 방법



세그멘테이션의 장점

중요한 부분과 아닌 부분을 분리하여 저장

- 중요한 부분과 그렇지 않은 부분을 분리할 수 있어서 효율적
- 같은 코드 영역은 하나만 저장하고 같은 곳을 가리키도록 하면 중복된 내용을 없앨 수 있음
- 내부 단편화 문제가 해결됨

세그멘테이션의 문제점

크기가 각각 다른 덩어리로 자름

- 길이가 다르게 자르기 때문에 외부 단편화가 발생할 수 있음
- 일반적으로 외부 단편화가 내부 단편화에 비해 훨씬 큰 공간이 낭비됨

페이징+세그멘테이션

부위별로 먼저 나눈 다음 각 부위를 일정 크기대로 자름

- 세그멘트를 페이징 기법으로 나누는 방법
- 세그먼트의 장점을 살리면서 외부 단편화를 방지할 수 있음
- 테이블을 두 번 거쳐야 하기 때문에 조금 느려질 수 있음