운영체제

2019.7.5

컴퓨터공학과 이병문
가천대학교 Gachon University

강의일정

- 7/2 Deadlock
- 3 CPU스케줄링1(온라인수행과
- 4 CPU스케줄링2
- 5 메모리관리1
- 7/8 중간고사2

메모리관리

- 메모리주소
- 동적적재기법
- 중첩기법
- 교체기법

연속메모리할당

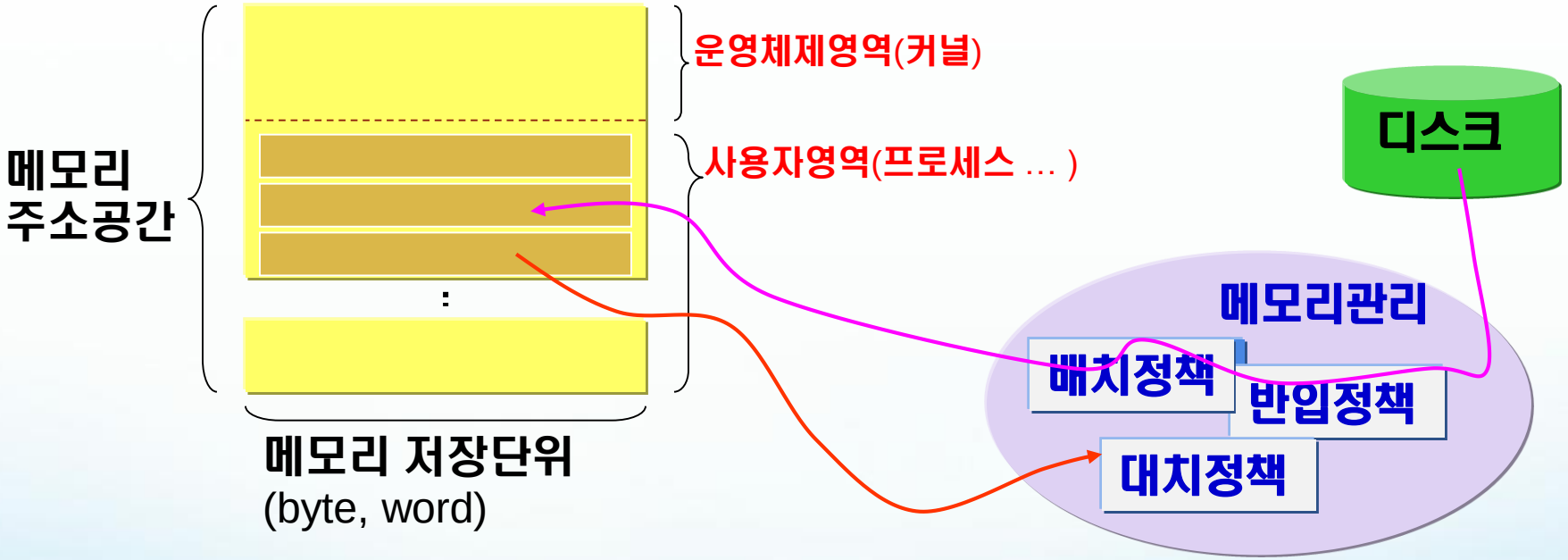
- 연속적재
- 고정분할/가변분할
- 단편화(내부, 외부)
- 버디시스템

메모리 관리

■ 운영체제의 중요한 기능, 메모리 관리

- ☑ 메모리 관리정책
- ☑ 연속메모리 할당기법
- ☑ 분산메모리 할당기법
- ☑ 세그먼트 메모리 관리기법

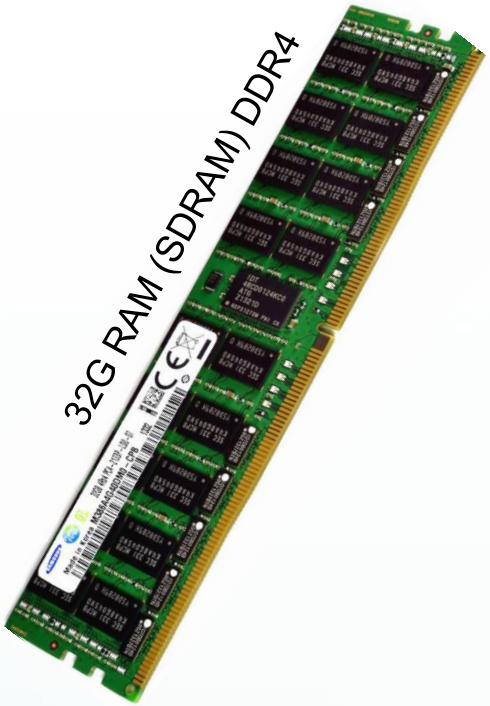
■ 메모리 관리개념

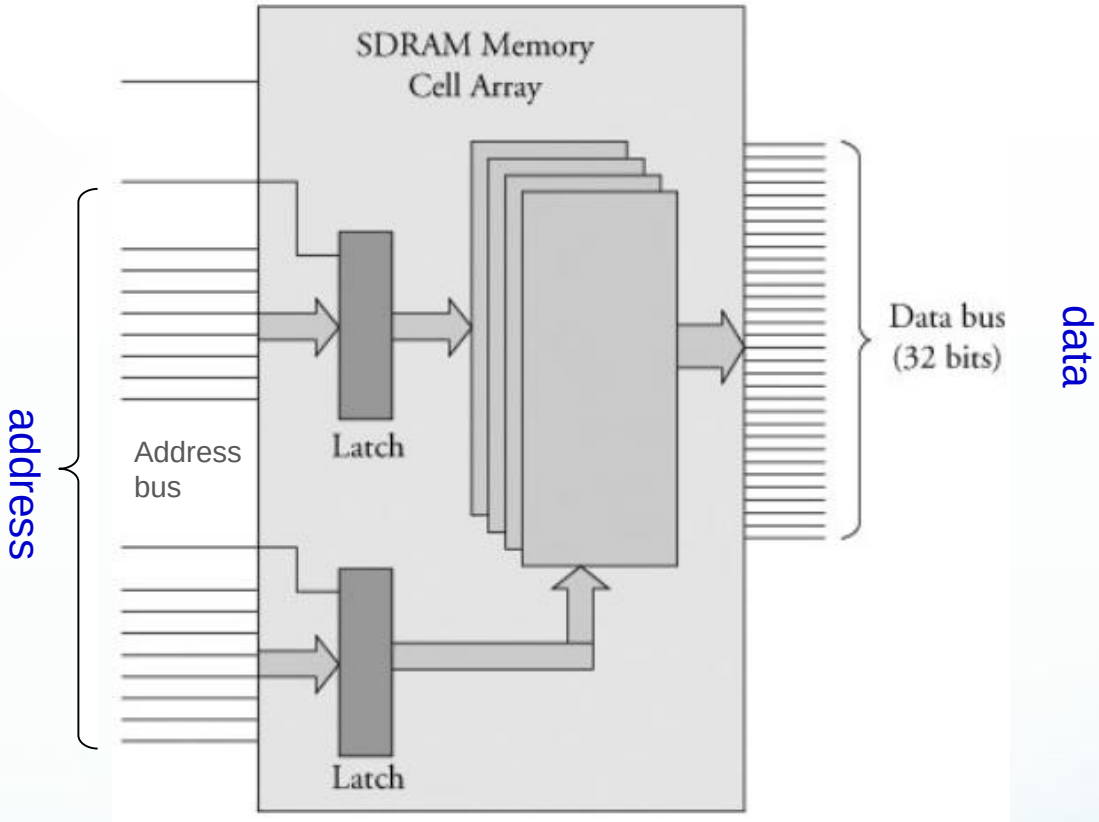


메모리 관리

■ 메모리

☑ 물리적 메모리

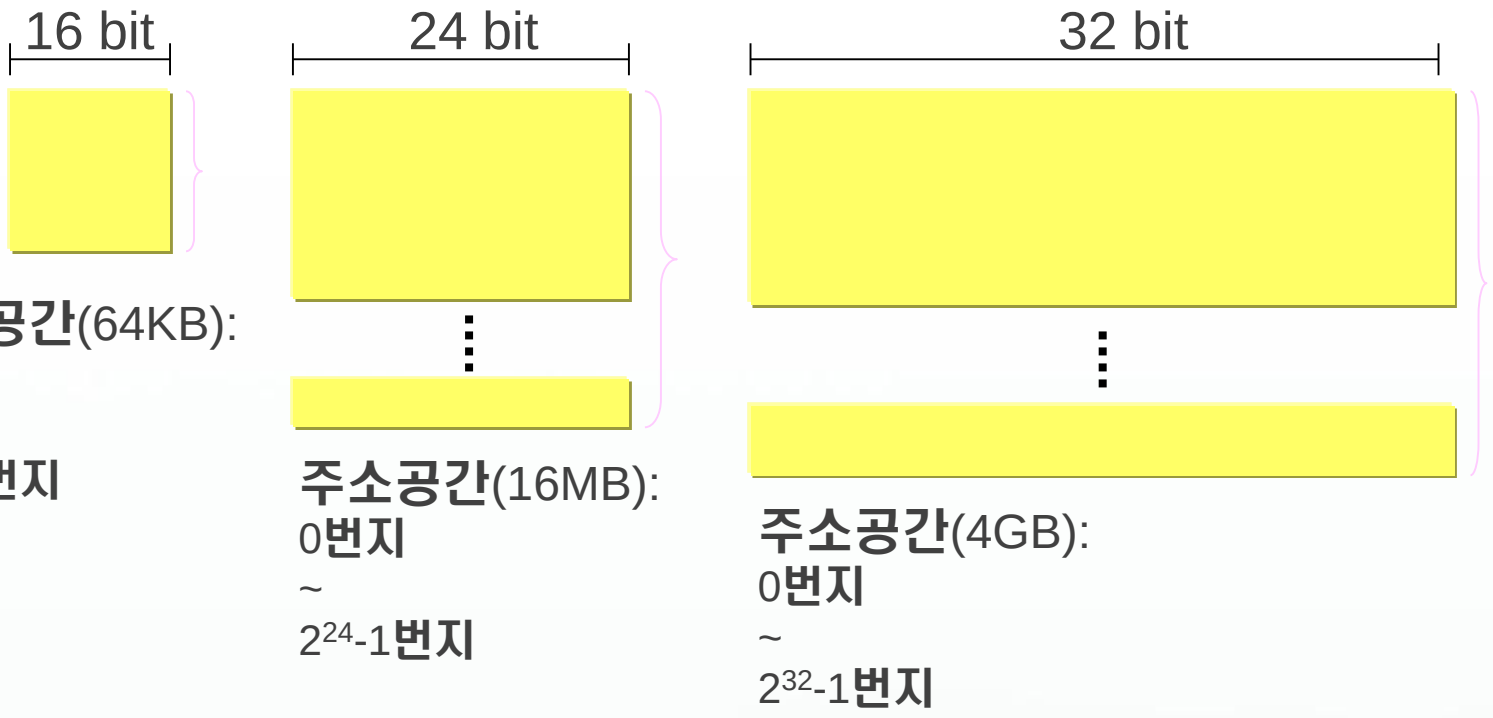




메모리 관리

■ 메모리 주소개념

☑ 주소공간의 분리(물리적, 논리적)



메모리 관리

■ 메모리 주소개념

☑ 물리적 주소 vs 논리적 주소

- 물리적 공간(물리적 주소) - 실제 자료나 프로그램이 저장되는 공간
- 메모리 칩(chip) 또는 디스크 공간 : 바이트(byte) 단위
- 논리적 공간(논리적 주소) - 프로그래머가 프로그래밍시 사용하는 공간
  목적코드(object code)가 저장된 공간과 프로그램에서 사용하는 자료구조

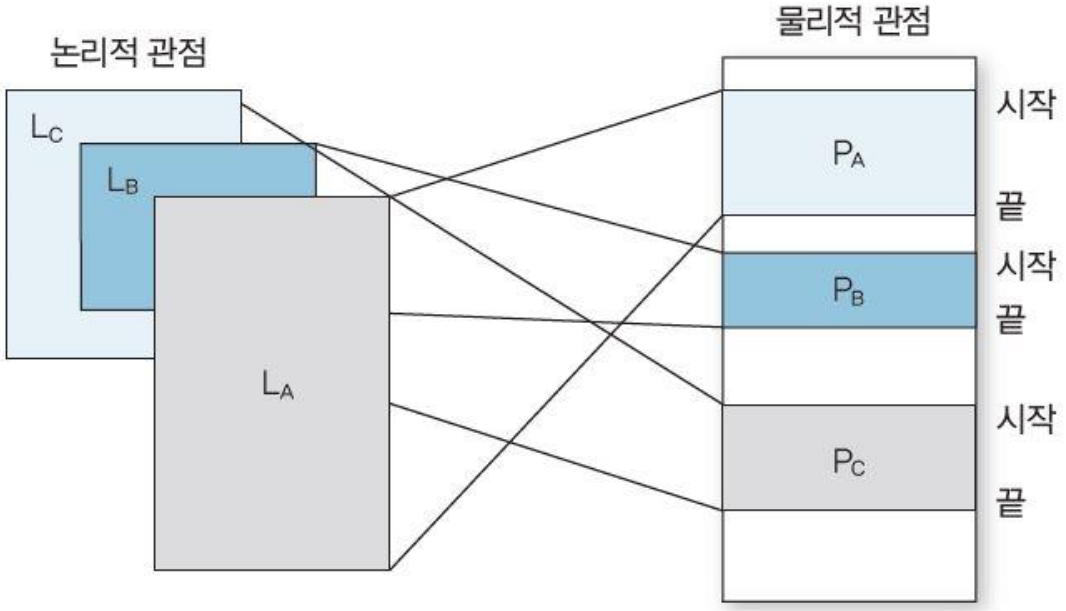
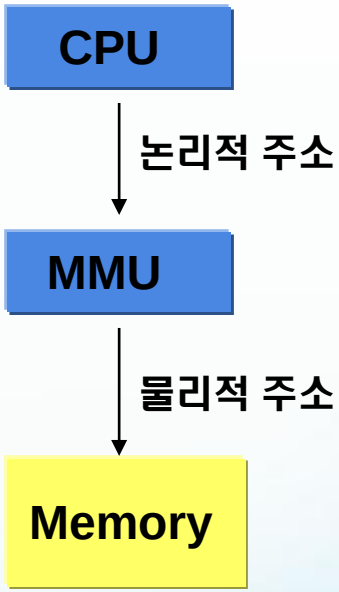


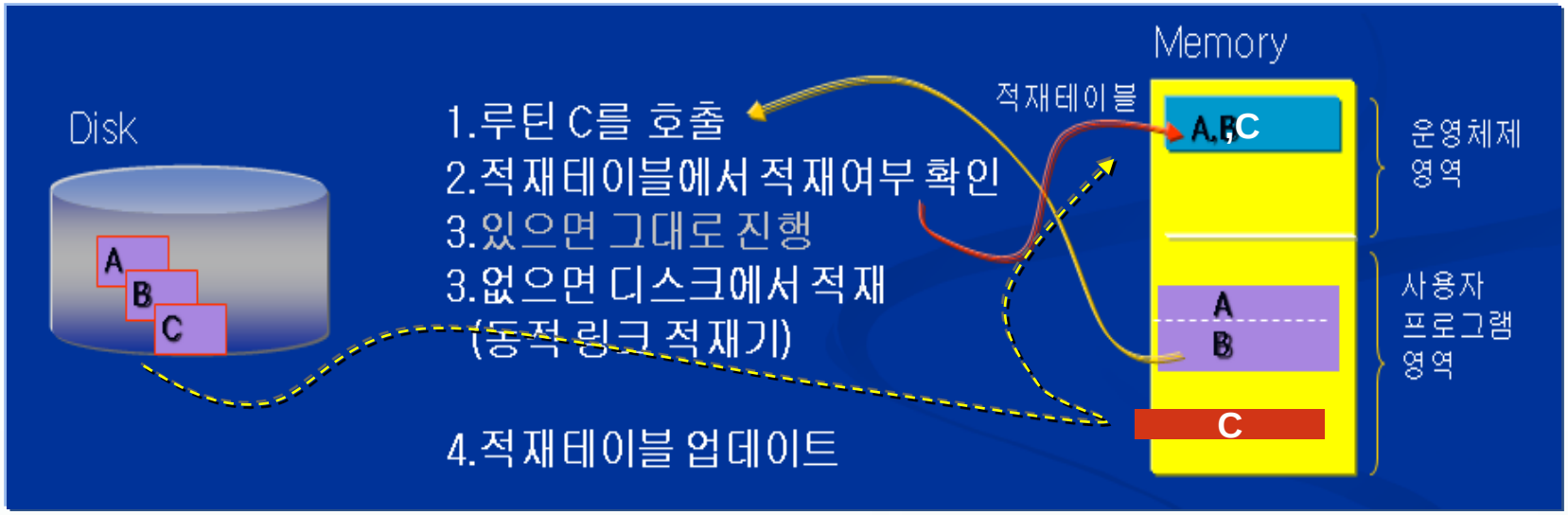
그림 7-1 논리적 주소와 물리적 주소



메모리 관리

■ 동적적재(Dynamic Loading) 기법

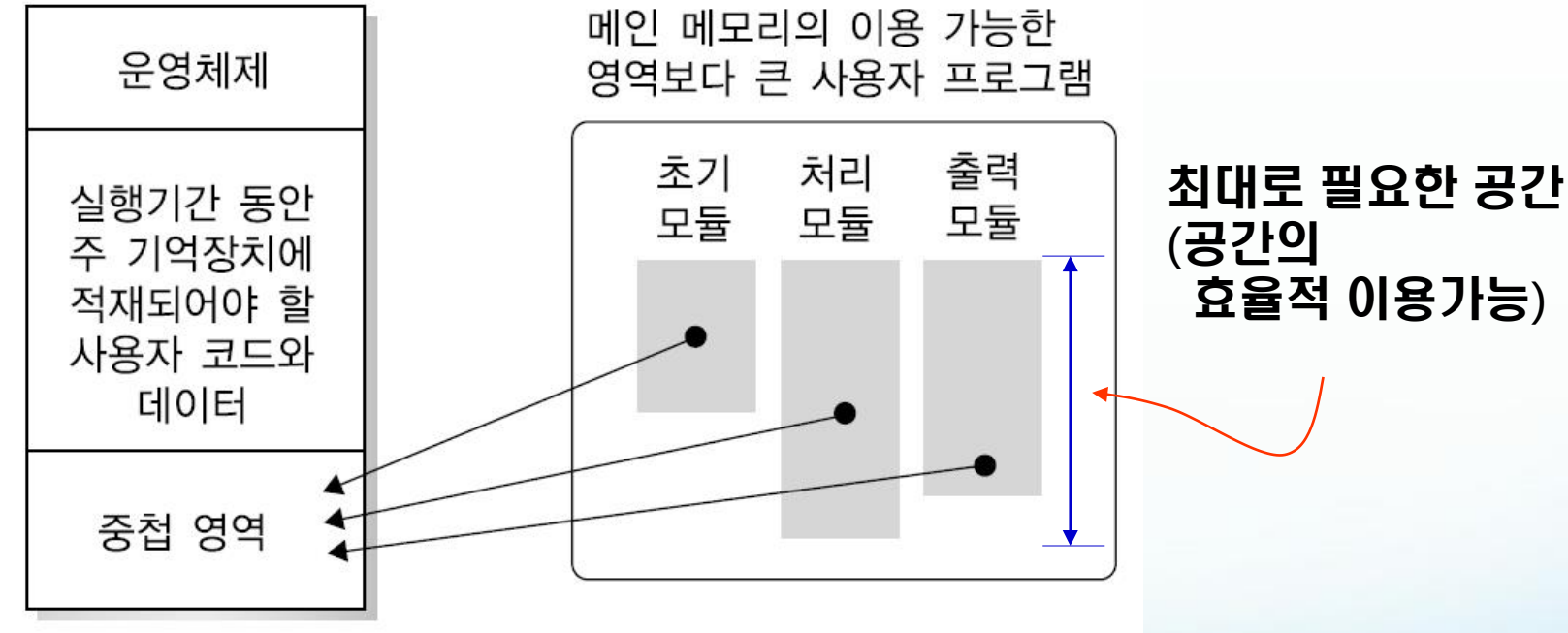
- 바인딩 시점을 실행시간으로, Late Binding
- 루틴(함수)이 호출되는 시점에 Loading, 즉. 호출되야 Loading
- 메모리공간의 효율적 운영이 가능



메모리 관리

■ 중첩(Overlay) 기법

- 프로그램 용량이 메모리 공간보다 큰 경우
  예, 프로그램(1000 KB) > 메모리 공간(512 KB)
- 우선적재 ; 프로그램 실행 시 반드시 필요한 명령과 데이터
- 중첩영역 ; 필요한 경우에 한하여 필요모듈을 필요한 시기에 적재
- 메모리공간의 **효율적 운영이** 가능

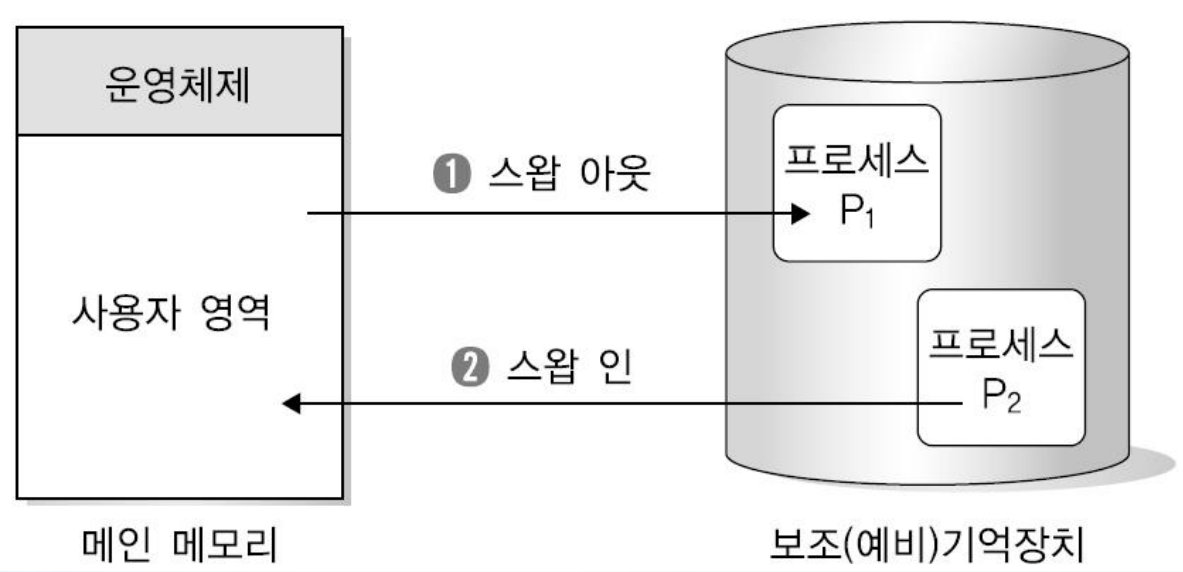


메모리 관리

■ 교체(Swapping) 기법

- 다중프로그래밍 환경-사용자 프로그램 완료까지 메인 메모리에 저장
- Round-Robin, Priority Scheduling 경우는 다소 문제발생
- 실행중인 프로세스 -> 메모리에 존재,
- 실행보류 프로세스-> 보조 기억장치(Swap영역)로 이동

(공간의 효율적 운영)



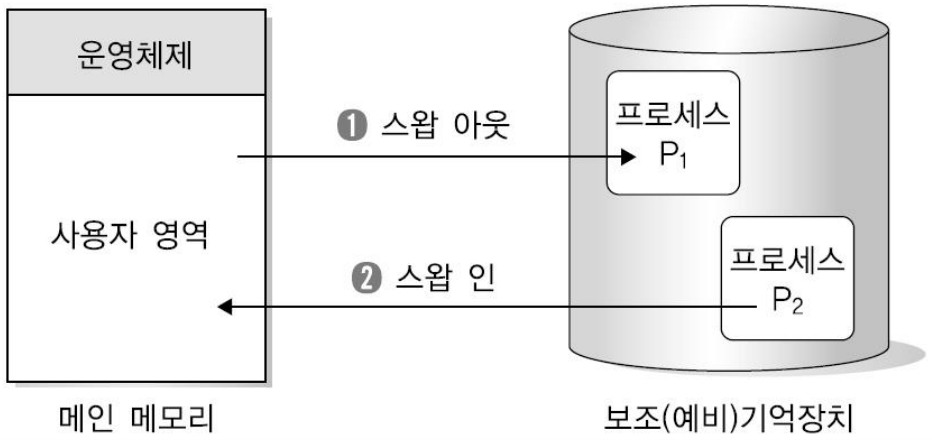
Swap Out
Swap In ;

메모리 관리

■ 교체(Swapping)시간

사용자 프로세스 용량 ; 100KB
디스크 전송률 ; 1MB
회전지연시간 ; 8ms

교체시간 = Time(Swap out)
+ Time(Swap in)
+ Time(**회전지연시간**)



Time(Swap out)
= Time(Swap in)

1000KB : 1초 = 100KB : x초
x초 = 100KB / 1000KB = 0.1초 = 100ms

교체시간 = 100ms + 8ms + 100ms + 8ms

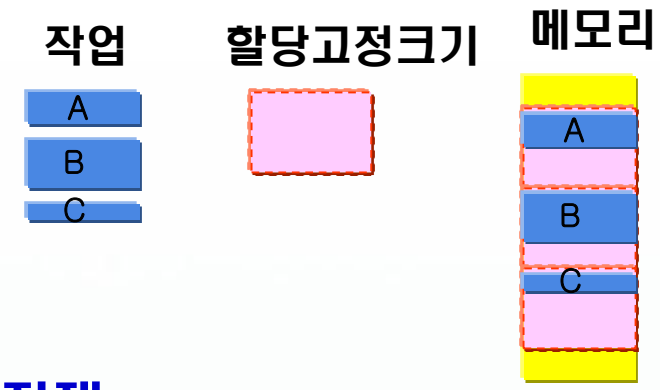
메모리 관리

■ 메모리관리의 변천

- **연속적재** ; 프로세스를 메모리에 연속적으로 할당하는 방식

  직접배치, 중첩배치, 분할기법

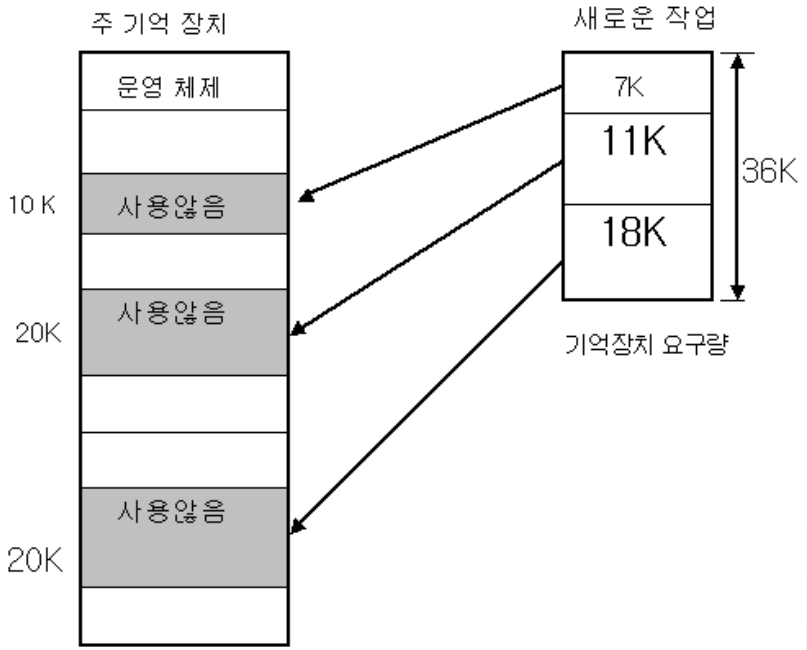
  작업 용량과는 상관없이 고정크기로 할당 -> 메모리낭비발생



- **분산적재** ;

  프로세스 용량에 맞게 메모리를 동적으로 분할하여 할당

  -> 가상메모리기법으로 발전



메모리 관리(연속메모리할당)

■ 단일사용자의 연속메모리 할당

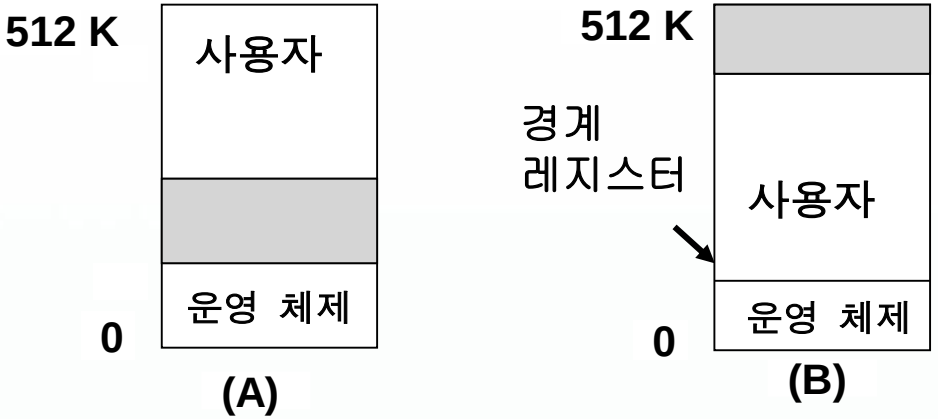
- **초기컴퓨터의 할당방식**(MS-DOS)

가장 원시적인 형태, 사용자가 메모리를 직접사용,

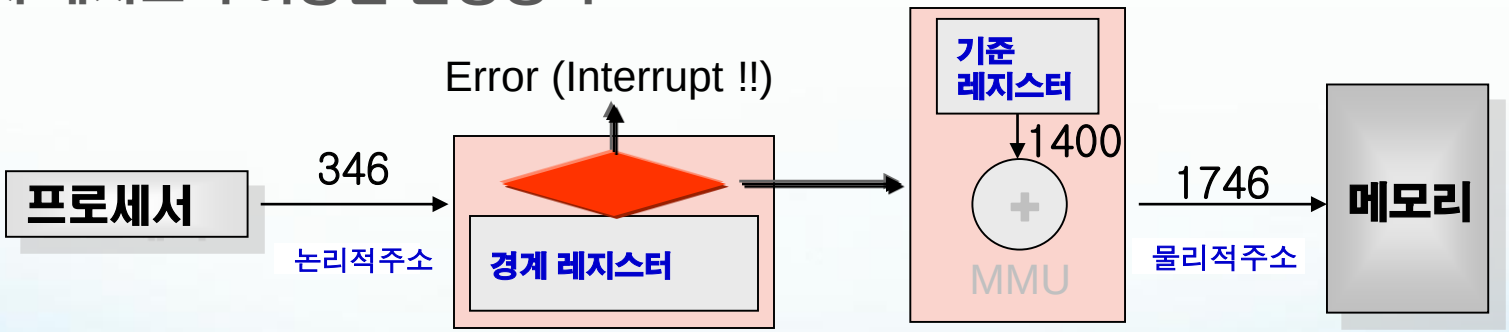
실행가능:작업용량 < 메모리공간, 실행불가능: 작업용량 > 메모리공간

기준 레지스터 이용한 할당방식

단일사용자



재배치 레지스터 이용한 할당방식

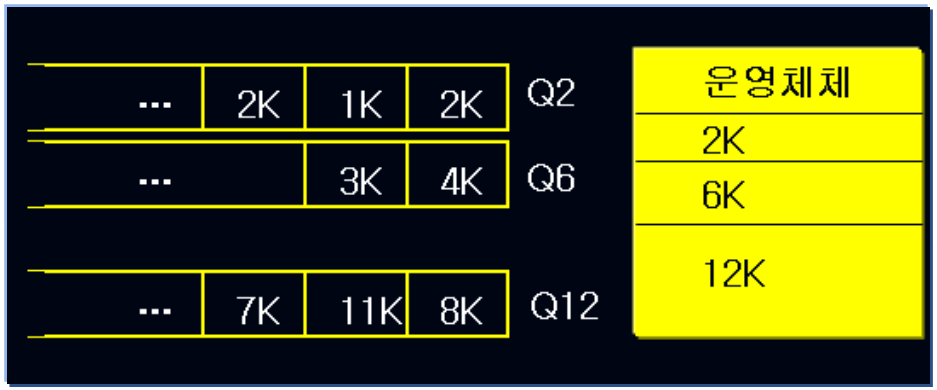


메모리 관리(연속메모리할당)

■ 고정분할 다중프로그래밍

- 메모리를 고정된 크기로 분할(정적 분할)하여 프로세스에 할당
- 다중프로그래밍 정도 = 분할 정도,

  예) 5개로 분할 -> 5개 multi programming

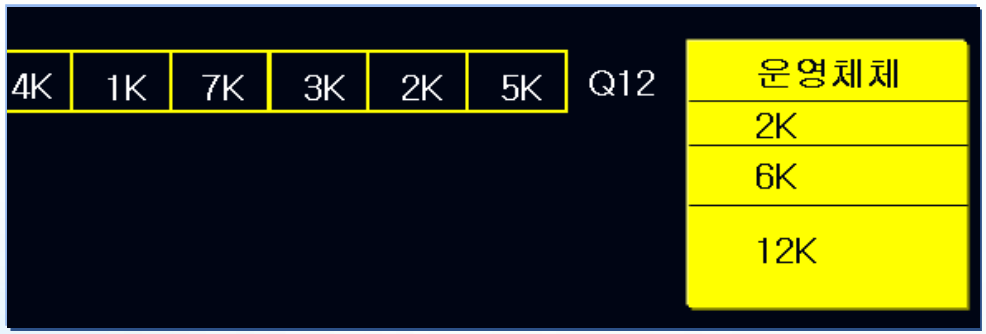


문제점

Q12 큐가 가득 차 있고 다른 큐**(Q2, Q6)**가 비어 있더라도 이용할 수 없음

개선 방법

모든 작업을 통합 큐에 넣고 운영



문제점

5K->2K-> Wait (6K Q)

고려사항

분할영역의 적정크기

영역의 배정

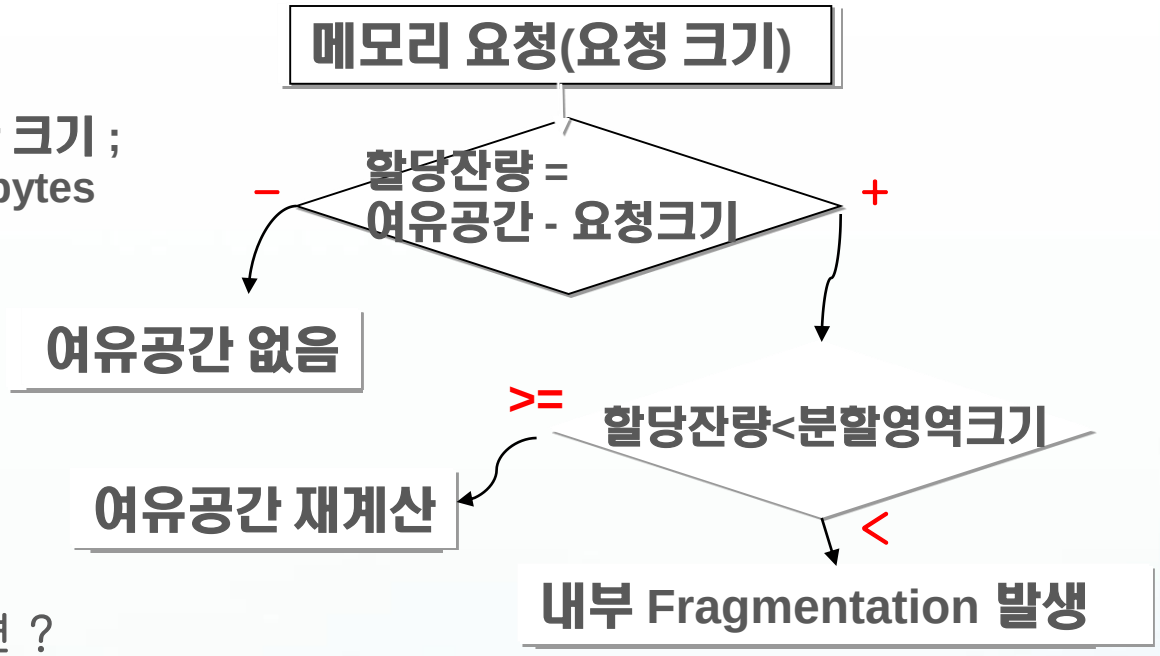
메모리 관리(연속메모리할당)

■ 고정분할 다중프로그래밍의 단편화

- **외부단편화** ; 할당할 수 있지만 연속적으로는 안 되는 경우
- **내부단편화** ; 사용자 작업크기 ≠ 메모리 분할영역 크기

  분할영역 크기가 너무 작아 사용자 작업을 하나도 할당 할 수 없는 경우

여유공간 크기 ;
18,646 bytes



예) 18,462 bytes 요청하면 ?

184 byte internal fragmentation 발생 !

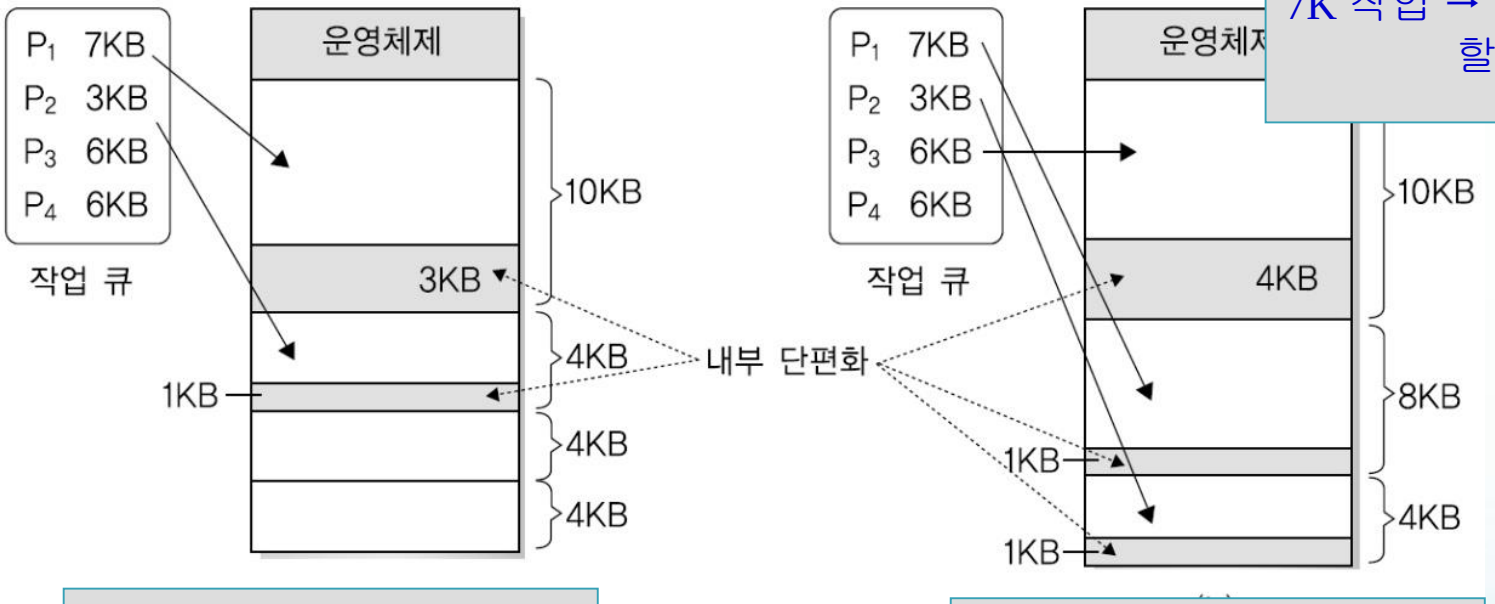
메모리 관리(연속메모리할당)

■ 사례) 고정분할 다중프로그래밍의 단편화 문제

-10K영역과 3개의 4K영역으로 나눔
7K 작업 →10K영역에[3K 내부 단편화]
3K 작업 → 4K 영역 하나에 [**1K 내부 단편화**] 할당
6K 작업 → 할당할 수 없음

10K, 8K, 4K 영역으로 나눔
7K 작업 → 8K 영역에
3K 작업 → 4K영역에
6K 작업 → 10K 영역에 할당 수행

6K의 내부 단편화 발생
7K 작업 → 10K 영역에 할당 가능



4K 단편화문제, P_3/P_4 대기 발생!!!

6K 단편화문제, P_4 대기 발생!!!

메모리 관리(연속메모리할당)

■ 가변분할 다중프로그래밍

- 작업이 필요한 만큼의 메모리 차지 -> 단편화 개선 가능
  고정된 경계 제거: 가변분할 다중프로그래밍
- 가변분할 할당 : 테이블 유지
  메모리의 어떤 부분이 사용될 수 있고, 어떤 부분이 사용 중인가 정보 필요

[예] 256K - 메모리
40K - 운영체제

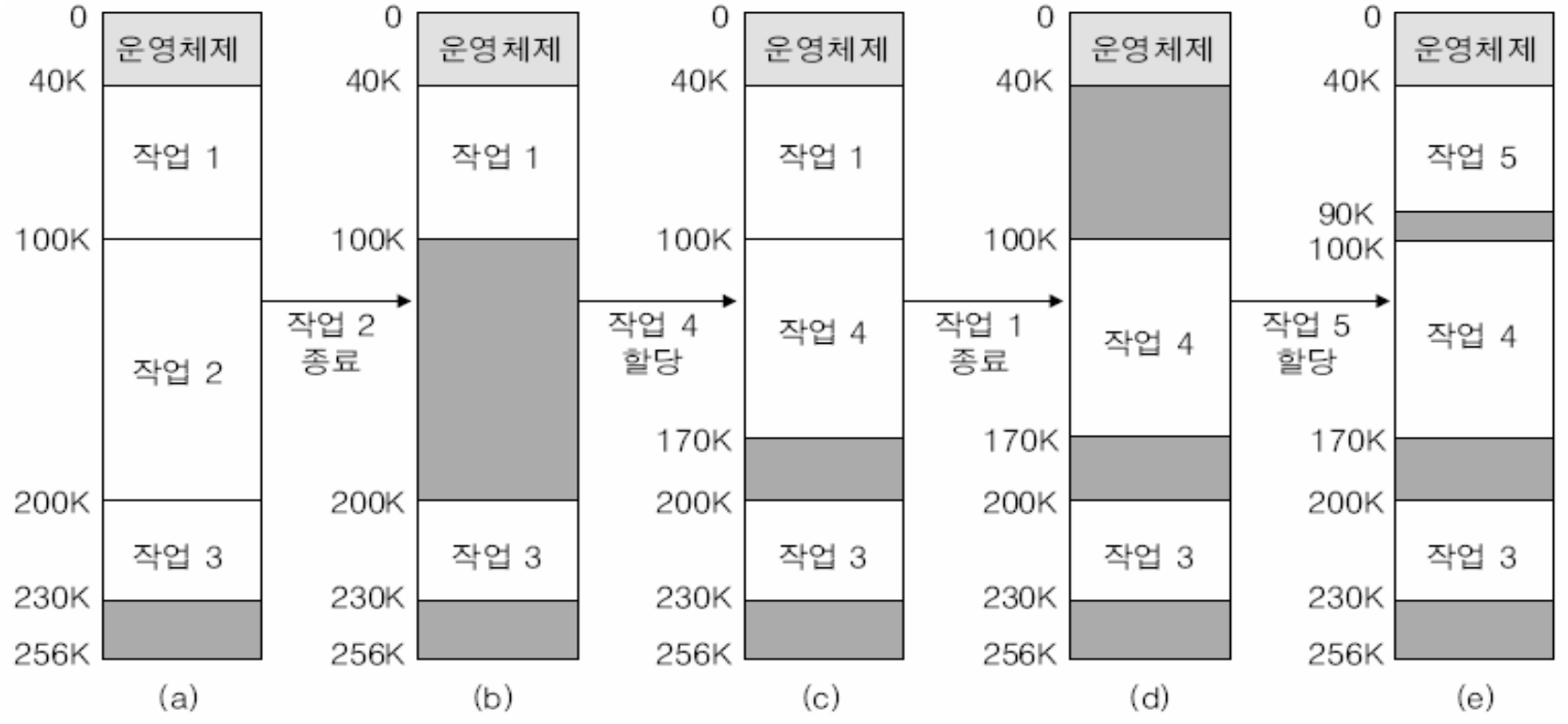
작업 큐

작업	메모리	시간
1	60KB	10
2	100KB	5
3	30KB	20
4	70KB	8
5	50KB	15

메모리 관리(연속메모리할당)

■ 가변분할 다중프로그래밍

-작업1, 작업2, 작업3에 메모리 할당 (a)
-작업2는 5 시간단위 후 끝나게 되어 사용 중이던 메모리 할당량 해제(b)
-작업4가 스케줄 되어 할당(c)
-작업1은 10 시간단위 후 끝나게 되어 사용 중이던 메모리 할당량 해제(d)
-작업5가 스케줄 되어 할당(e)



메모리 관리(메모리 배치기법)

■ 메모리배치 정책

☑ 메모리 배치기법

- 가변분할 메모리 할당 시 프로세스의 메모리 할당위치 선정방법

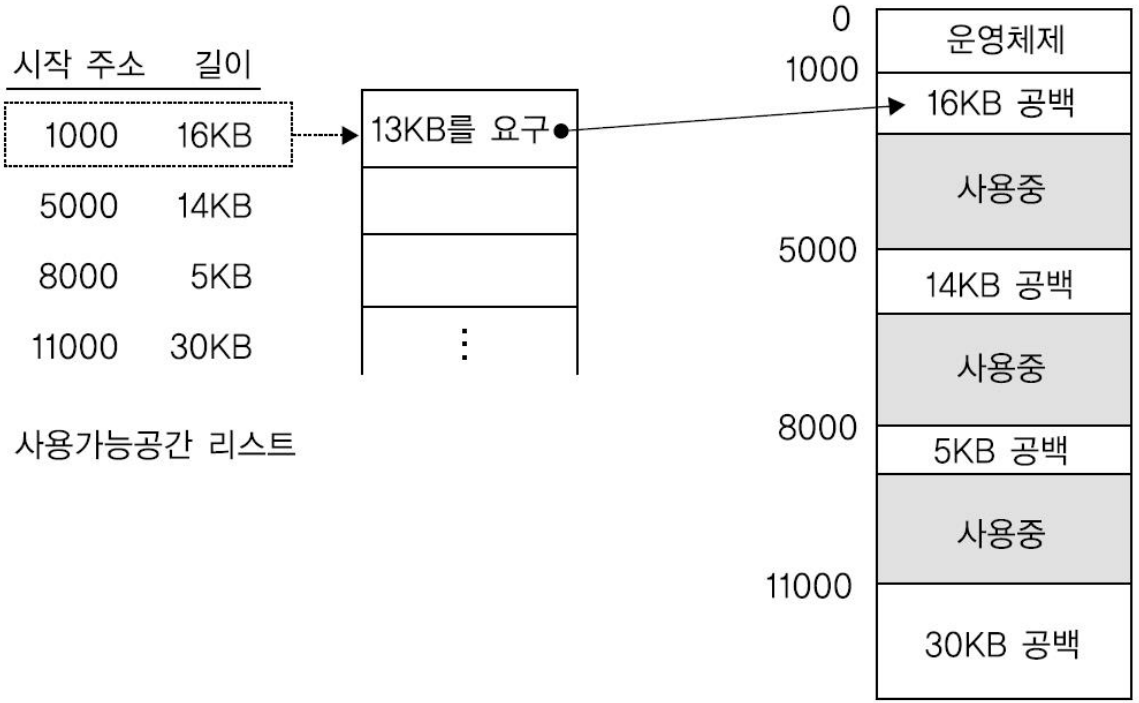
1) 최초적합(First-Fit)

2) 최상적합(Best-Fit)

3) 최악적합(Worst-Fit)

☑ **최초적합 기법**

- 사용가능공간리스트 검색하여 **첫 번째로 찾은** 공간을 우선할당!
- 검색&할당 시간빠름
- 공간활용률 ▼

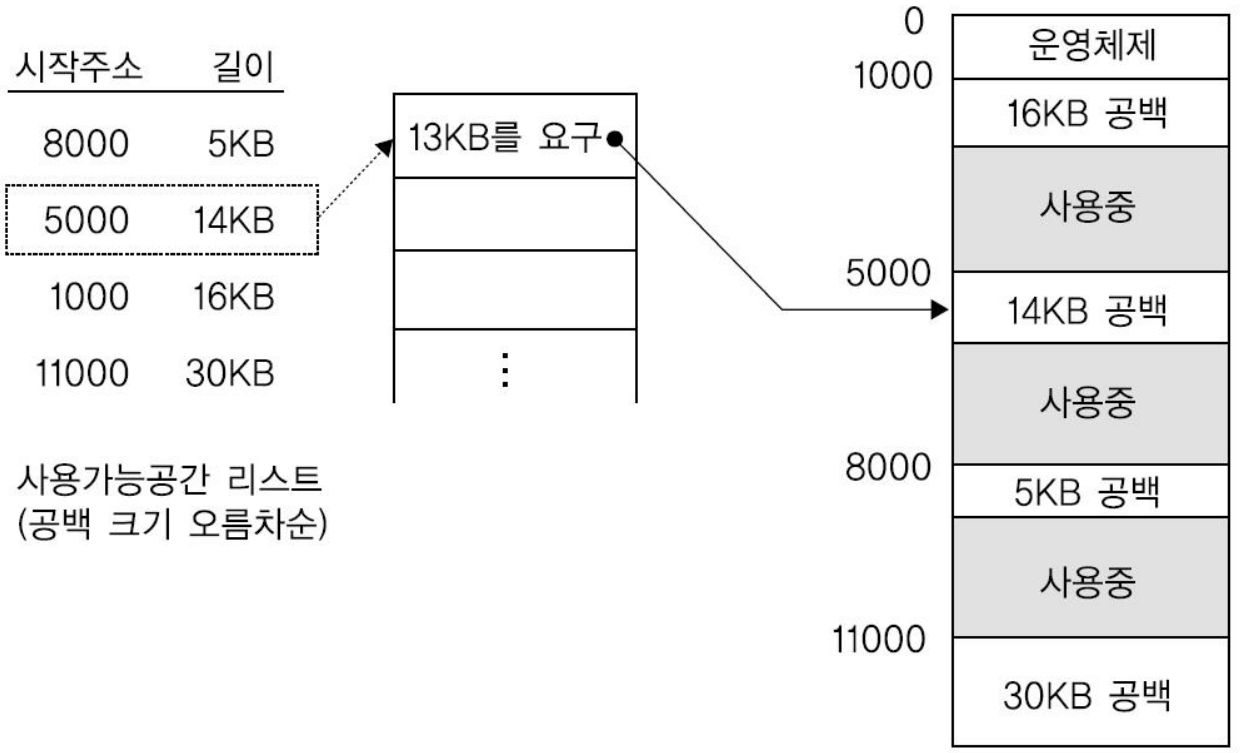


메모리 관리(메모리 배치기법)

■ 메모리배치 정책

☑ 최상적합 기법

- 사용가능 공간 리스트중에서 **가장 작은** 크기의 사용공간 할당
- 공간이용률 ▲, 크기에 따른 주기적인 정렬(Sorting)필요
- 시간소요 ▲, 단편화(Fragmentation) 발생

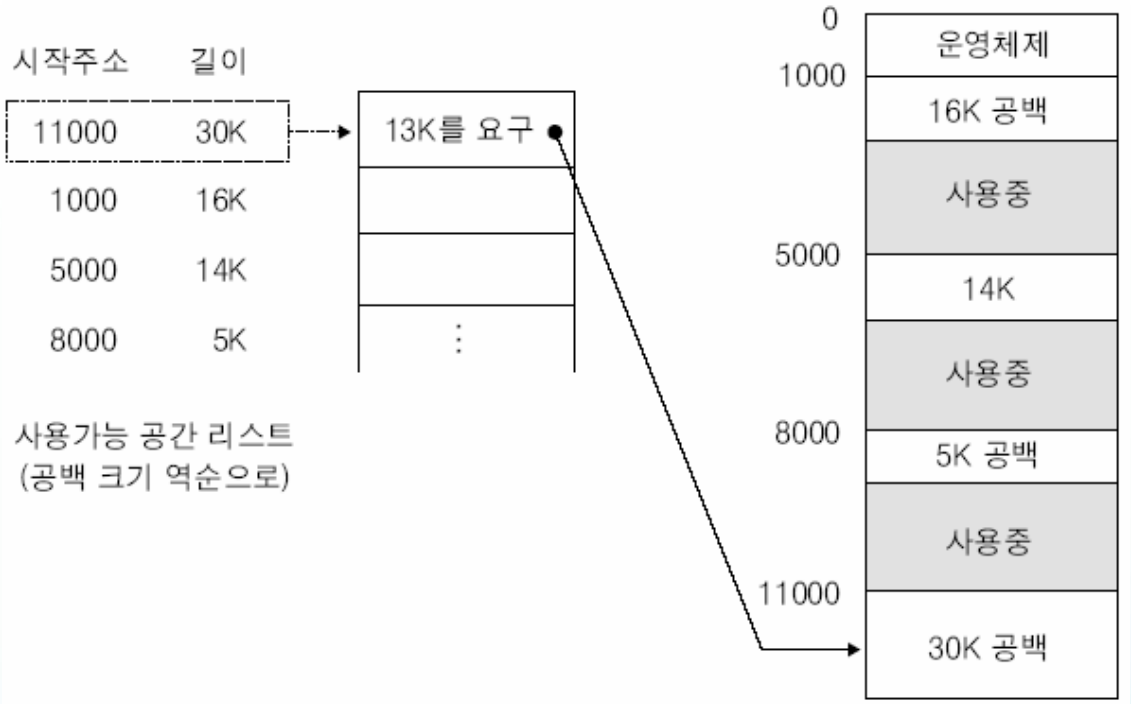


메모리 관리(메모리 배치기법)

■ 메모리배치 정책

☑ 최악적합 기법

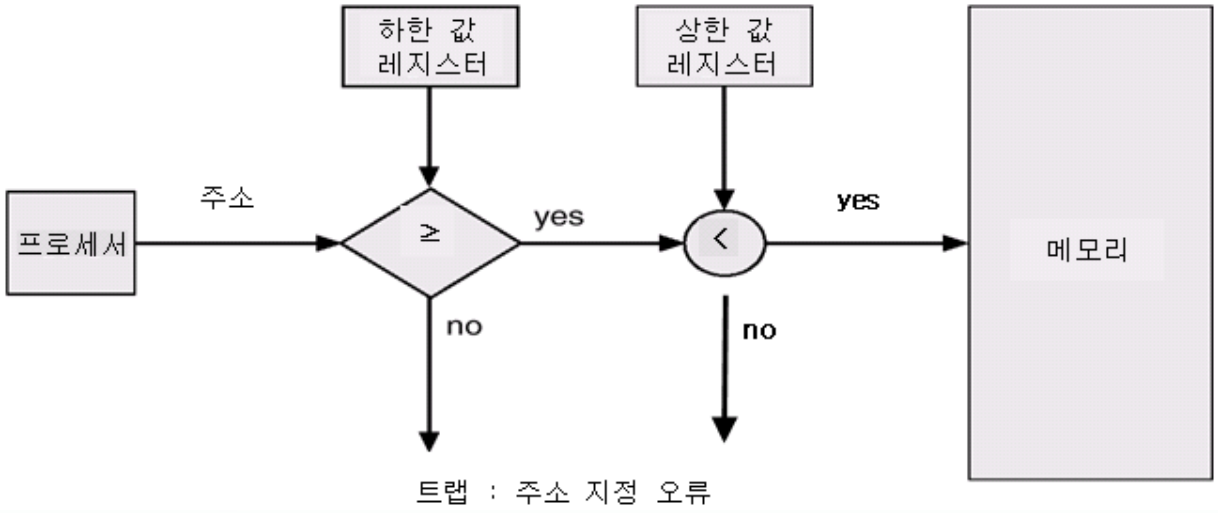
- 사용가능 공간 리스트중에서 **가장 큰** 크기의 사용공간 할당
- 공간이용률 ▲, 크기에 따른 주기적인 정렬(Sorting)필요
- 시간소요 ▲, 단편화(Fragmentation) 발생 정도를 줄임



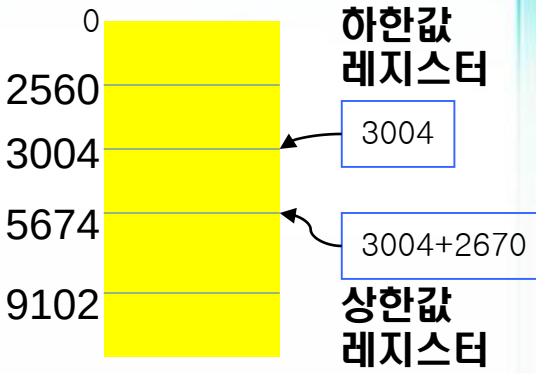
- 평가(시간, 메모리이용률) ; 최초적합 > 최상적합 > 최악적합

메모리 관리(메모리 배치기법)

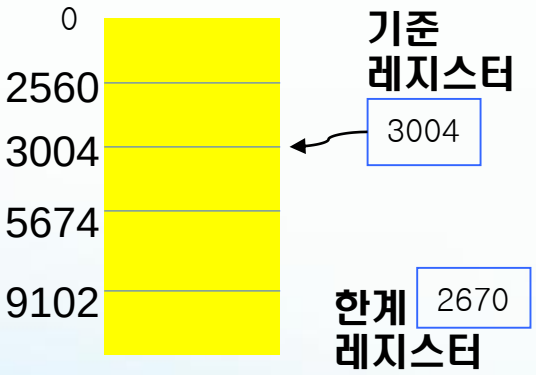
■ 가변분할에서의 메모리 보호

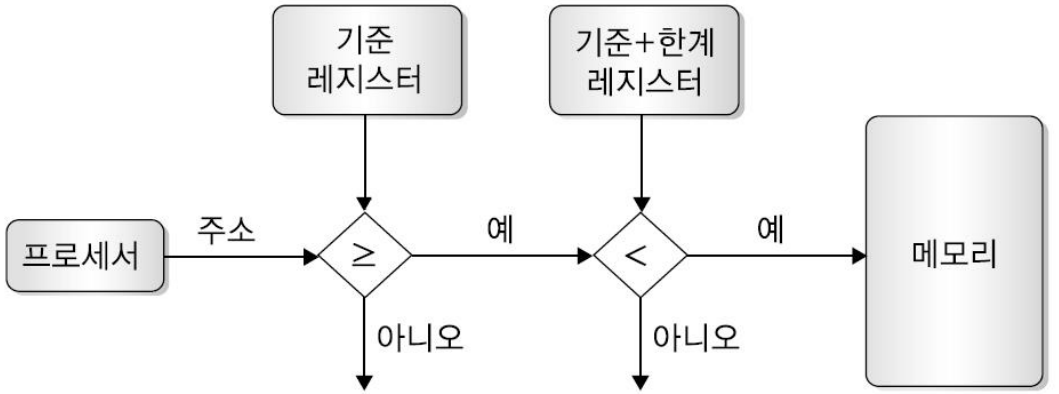


트랩 : 주소 지정 오류



상한값=하한값+프로세스크기
(2670)



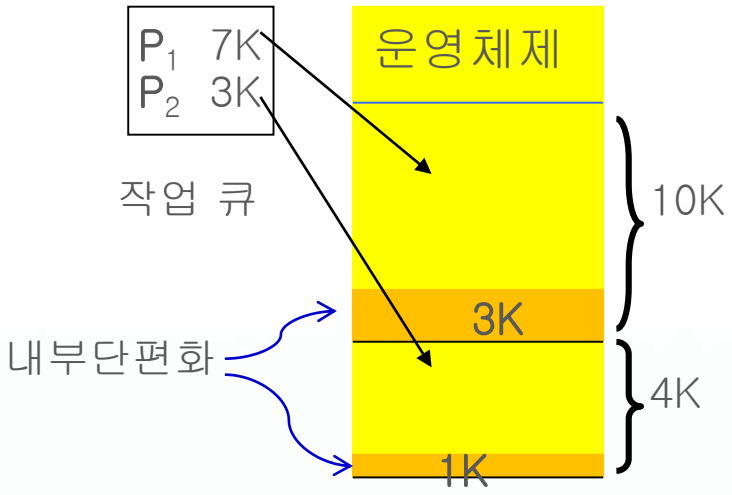


트랩: 주소 지정 오류

메모리 관리

■ 단편화(Fragmentation)

☑ 내부단편화(Internal Fragmentation)



☑ 외부단편화(External Fragmentation)

할당할 수 있지만 연속적으로는 안 되는 경우
가변분할 알고리즘은 내부단편화는 없지만 외부단편화가 발생될 수 있음

메모리 관리

■ 외부단편화(External Fragmentation)

☑ 프로세스들이 연속된 메모리 차지하는 과정에서 각각의 공백 발생됨

작업 큐

작업	메모리	시간
1	60K	10
2	100K	5
3	30K	20
4	70K	8
5	50K	15

단계1: 26K 외부단편화

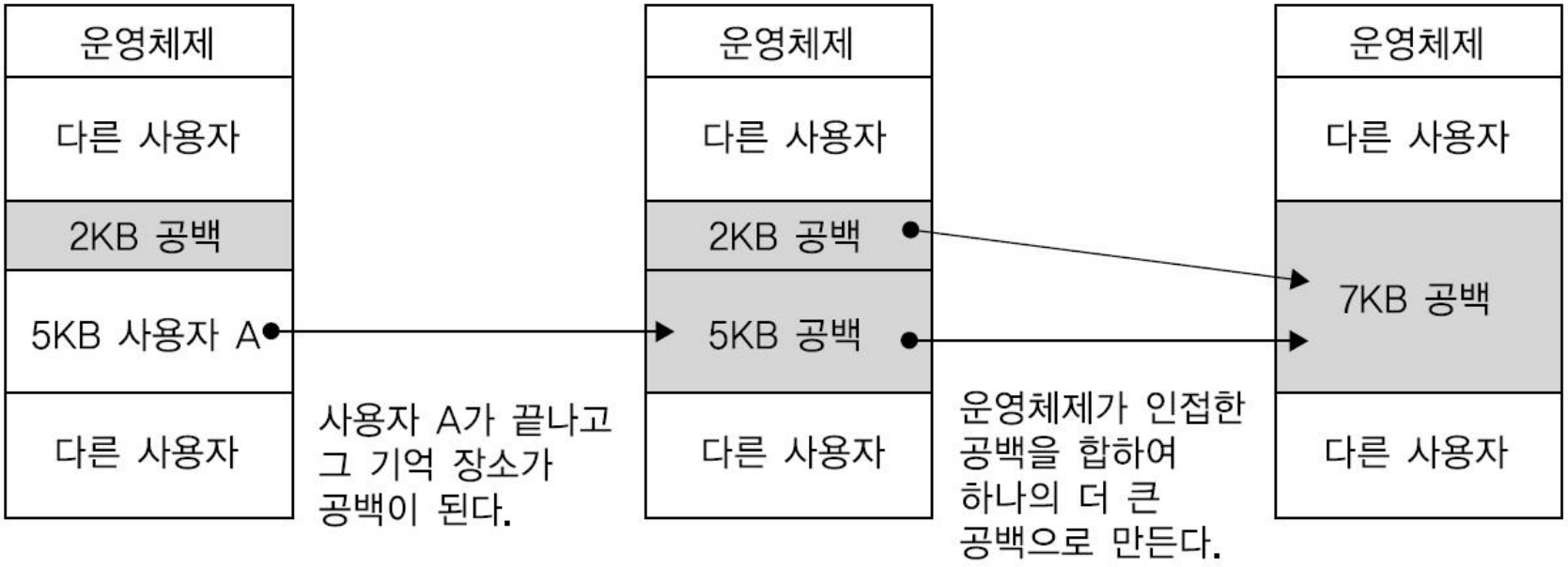
단계3 ; 30K 외부단편화

어떤 작업에도
할당할 수 없음
→ 외부단편화문제

메모리 관리

■ 단편화의 해결방법 -> 통합과 압축

☑ **통합** ; 메모리의 여유공간이 인접될 경우에 통합



메모리 관리

■ 단편화의 해결방법 -> 통합과 압축

☑ **압축** ; 메모리 내용 이동, 여유공간을 하나의 큰 블록으로 구성

압축의 특징

- 외부단편화를 해결
- 실행(runtime)시간에
  -> 주소재배치 필요
- 동적인 메모리 테이블을 유지
  ->과부하
- 압축되는 동안 모든작업 중단
- 압축으로 시스템 자원소모 발생

메모리 관리

■ 버디(Buddy) 시스템 알고리즘

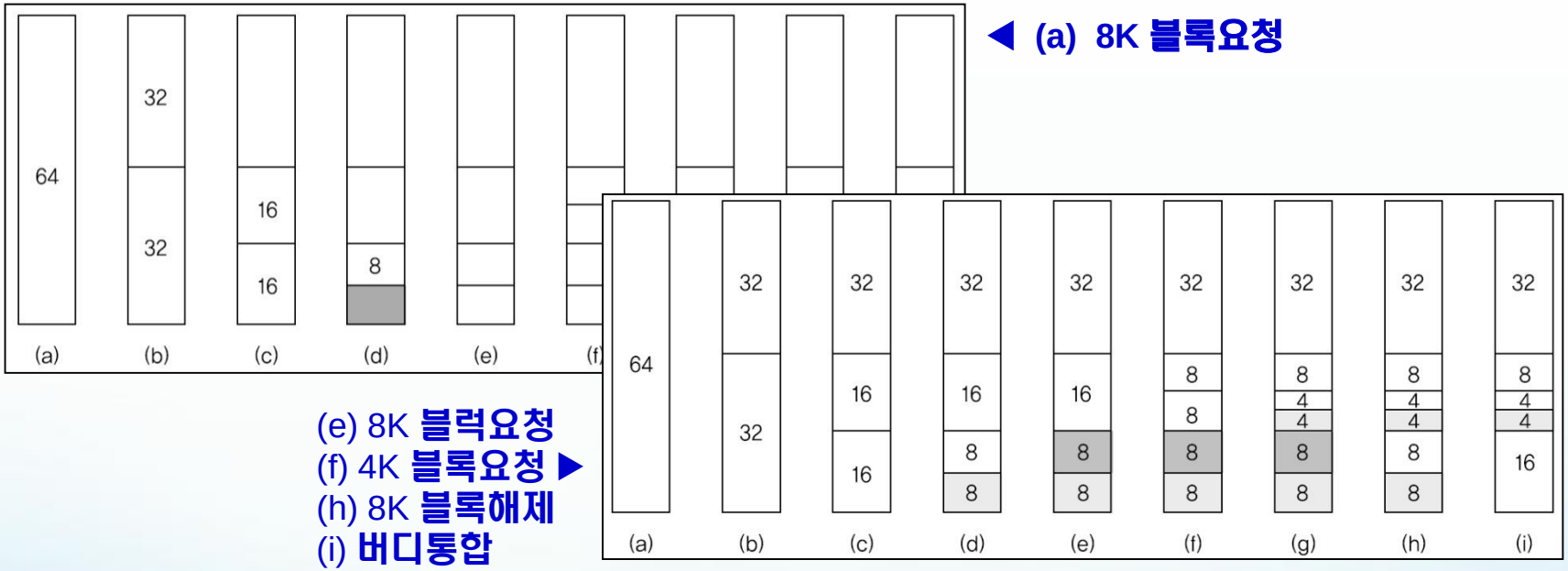
☑ 자원할당 과정에서 발생하는 단편화를 해결하는 방법

- 큰 버퍼를 반복적으로 이등분하여 작은 버퍼로 구성하고 틈틈히 통합
- 메모리 블록(K) ; $L \le K \le U$

L : 할당 가능한 가장 작은 블록, U : 가장 큰 블록(전체 메모리)

블록 : 전체크기 또는 2U-1의 크기를 갖는 2개의 버디로 나누어짐,

통합 ; 크기가 b인 빈 버디 블록 쌍 ⏲ 크기가 2b인 큰 블록 하나로 만듬



Q&A