제7장
주기억장치 관리

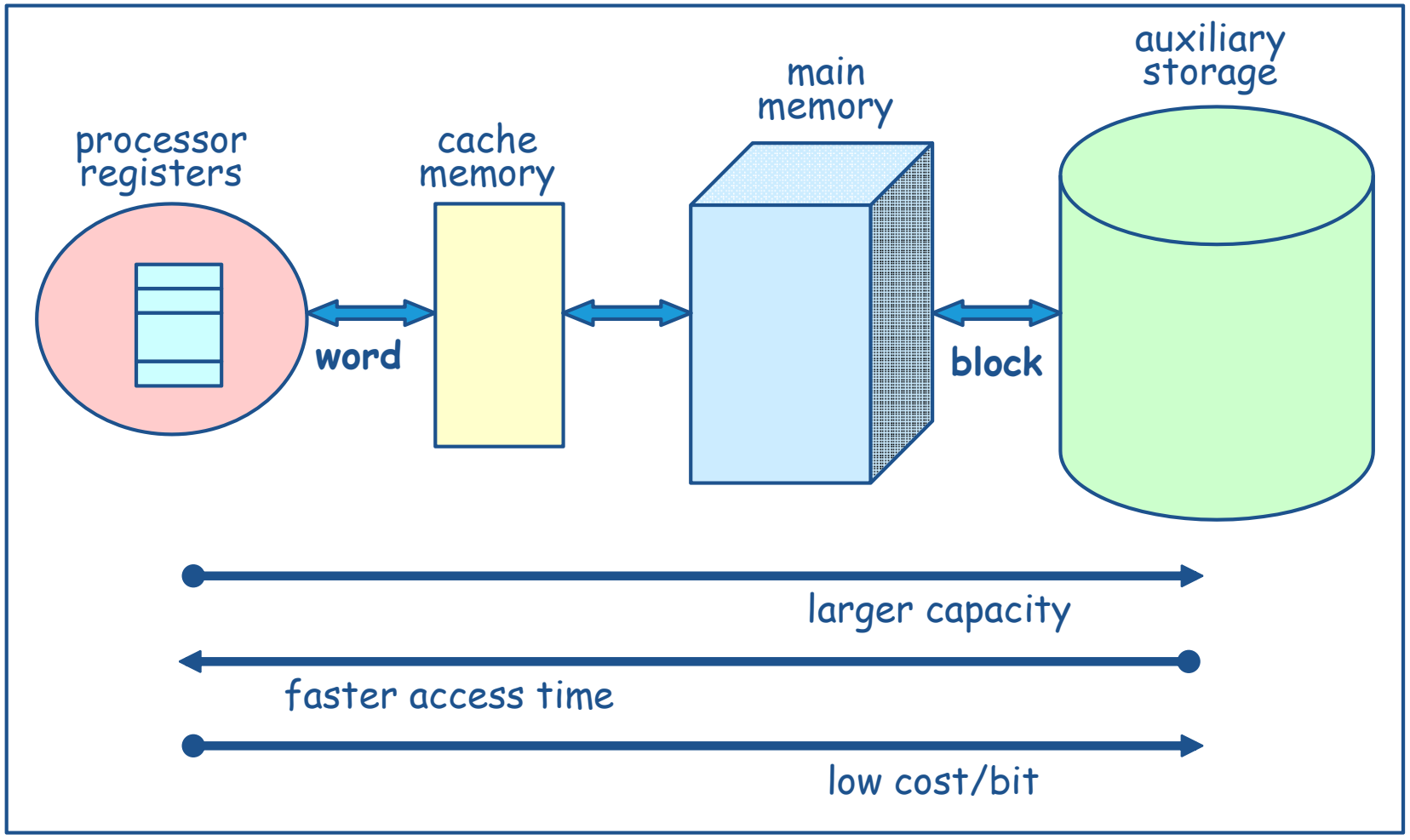
컴퓨터 운영체제 OS Operating Systems

Network Security Technology Lab.

기억장치 개요

- 컴퓨터 시스템의 기억장치 분류
  - 프로세서 레지스터 (processor registers)
  - 캐쉬 메모리 (cache memory)
  - 주기억장치 (main memory)
  - 보조 기억장치 (auxiliary memory)
- 프로세서 레지스터와 캐쉬 메모리
  - OS 관리 대상 아님, 하드웨어에 의해 제어됨
- 용어
  - 블럭(block)
    - 보조기억장치와 주기억장치간의 데이터 전송 단위
    - 크기 : 보통 1 KB내외 (128B ~ 4KB)
  - 워드(word)
    - 주기억장치와 프로세서 레지스터간의 데이터 전송 단위
    - 크기 : 보통 16 비트 ~ 64 비트

기억장치 계층구조


processor registers
cache memory
main memory
auxiliary storage
word
block
larger capacity
faster access time
low cost/bit

주기억장치 구성 및 관리 정책

- 가정
  - 프로세스의 실행에 필요한 데이터나 스택 등을 포함한 사용자 프로그램 context 전체가 주기억장치에 연속으로 적재되어 실행되는 환경
- 주기억장치 구성 정책
  - 주기억장치를 동시에 할당받을 수 있는 프로세스의 수
  - 각 프로세스에게 할당되는 주기억장치의 양
  - 주기억장치 분할 방법
  - 각 프로세스에게 할당된 분할영역의 교체 가능성
  - 프로세스에게 할당되는 주기억장치 영역의 연속성

주기억장치 구성 및 관리 정책

❖ 주기억장치 구성 정책

- 주기억장치를 동시에 할당받을 수 있는 프로세스의 수
  - 한 번에 한 사용자 프로그램만을 주기억장치에 적재 가능
    - 다중프로그래밍 정도 (multiprogramming degree) 가 1임
  - 동시에 여러 사용자 프로그램이 적재 가능
    - 다중프로그래밍 정도 (multiprogramming degree) 가 k임
      - » k개의 프로세스에게 동시에 주기억장치 할당 가능

주기억장치 구성 및 관리 정책

❖ 주기억장치 구성 정책

- 각 프로세스에게 할당되는 주기억장치의 양
  - 다중프로그래밍 정도가 2 이상인 경우
  - 동일한 양으로 또는 다른 양으로 할당할 지의 여부
    - 분할영역 (partition)의 크기를 같게/다르게 설정

주기억장치 구성 및 관리 정책

❖ 주기억장치 구성 정책

▪ 주기억장치 분할 방법

- 다중프로그래밍 정도가 2 이상인 경우
- 고정 분할 (fixed partition), 정적 분할 (static partition)
  - 초기의 분할 형태를 이후 변형 않는 방법
- 가변 분할 (variable partition), 동적 분할 (dynamic partition)
  - 초기의 분할 형태를 이후 필요에 따라 변형시키는 방법

주기억장치 구성 및 관리 정책

❖ 주기억장치 구성 정책

- 각 프로세스에게 할당된 분할영역의 교체 가능성
  - 실행중인 프로세스가 다른 분할 영역으로 이동하여 실행할 수 있도록 하는 정책
    - 유연성 (flexibility) 제공
    - 실행할 프로그램 코드가 재배치가능(relocatable)해야 함
    - 컴파일러, 어셈블러, 링커/로더 등의 기능 지원 필요
  - 실행중인 프로세스가 다른 분할 영역으로 이동하여 실행할 수 없도록 하는 정책
  - 할당받은 분할 영역의 교체가 필요한 경우
    - 프로세스가 주기억장치를 완전히 반납 후 지연 상태에 있다가 다시 주기억장치를 할당받고자 할 경우에 발생

주기억장치 구성 및 관리 정책

- 주기억장치 구성 정책
  - 프로세스에게 할당되는 주기억장치 영역의 연속성
    - 연속(contiguous) 할당 정책
    - 비연속(discontiguous) 할당 정책
      - 관리상의 오버헤드
      - 최근의 주기억장치 관리 정책 경향

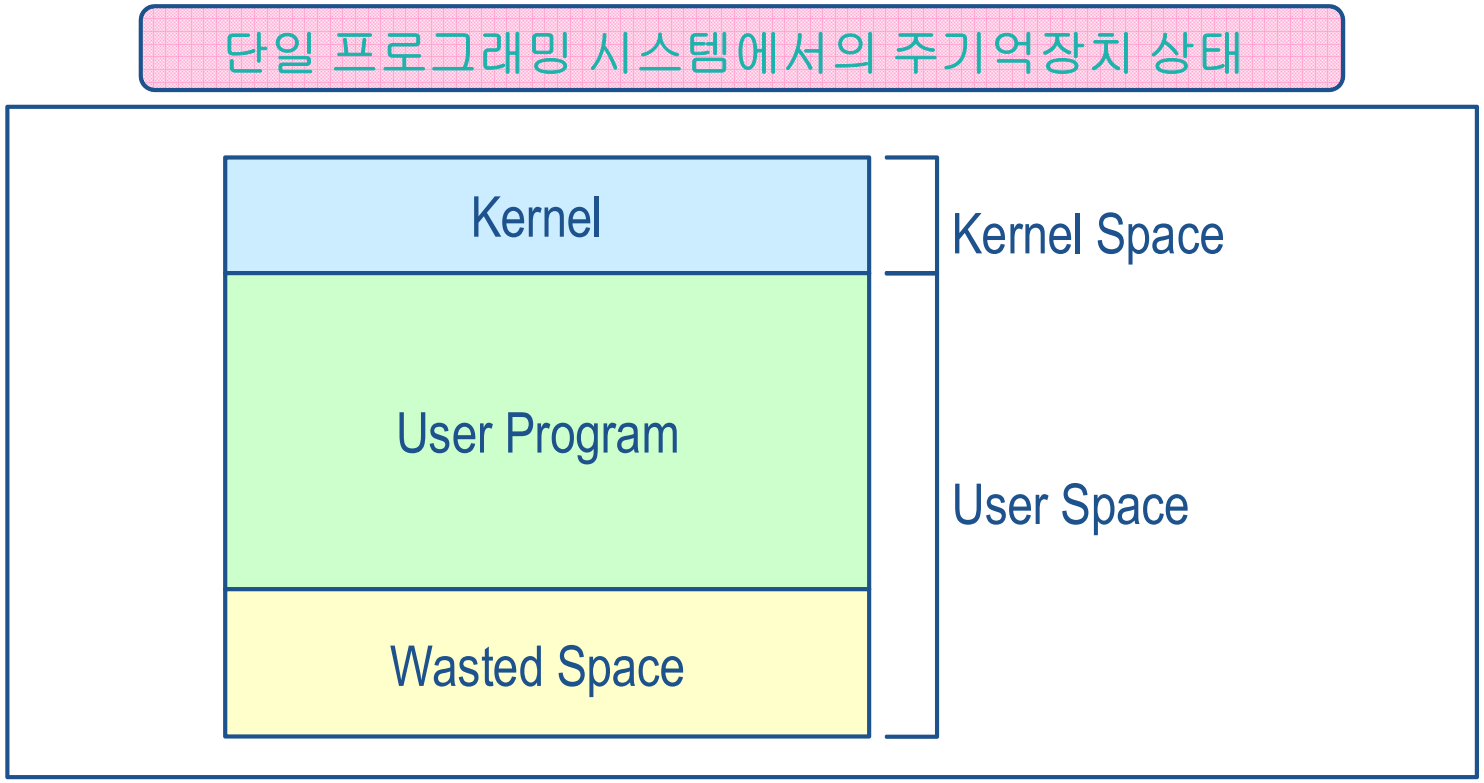
주기억장치 구성 및 관리 정책

주기억장치 관리 기법

- 호출 기법 (fetch strategy)
  - 새로 생성된 프로세스에 대한 주기억장치 할당 시기 결정
- 배치 기법 (placement strategy)
  - 주기억장치를 할당받고자 하는 프로세스에게 할당 공간의 위치 결정
  - first-fit, best-bit, worst-fit, next-fit 등
- 교체 기법 (replacement strategy)
  - 주기억장치 공간 부족시 어느 프로세스의 공간을 선점할지 결정
- 할당 기법 (allocation strategy)
  - 새로 생성되는 프로세스에 대한 공간 할당량 결정

단일프로그래밍 시스템

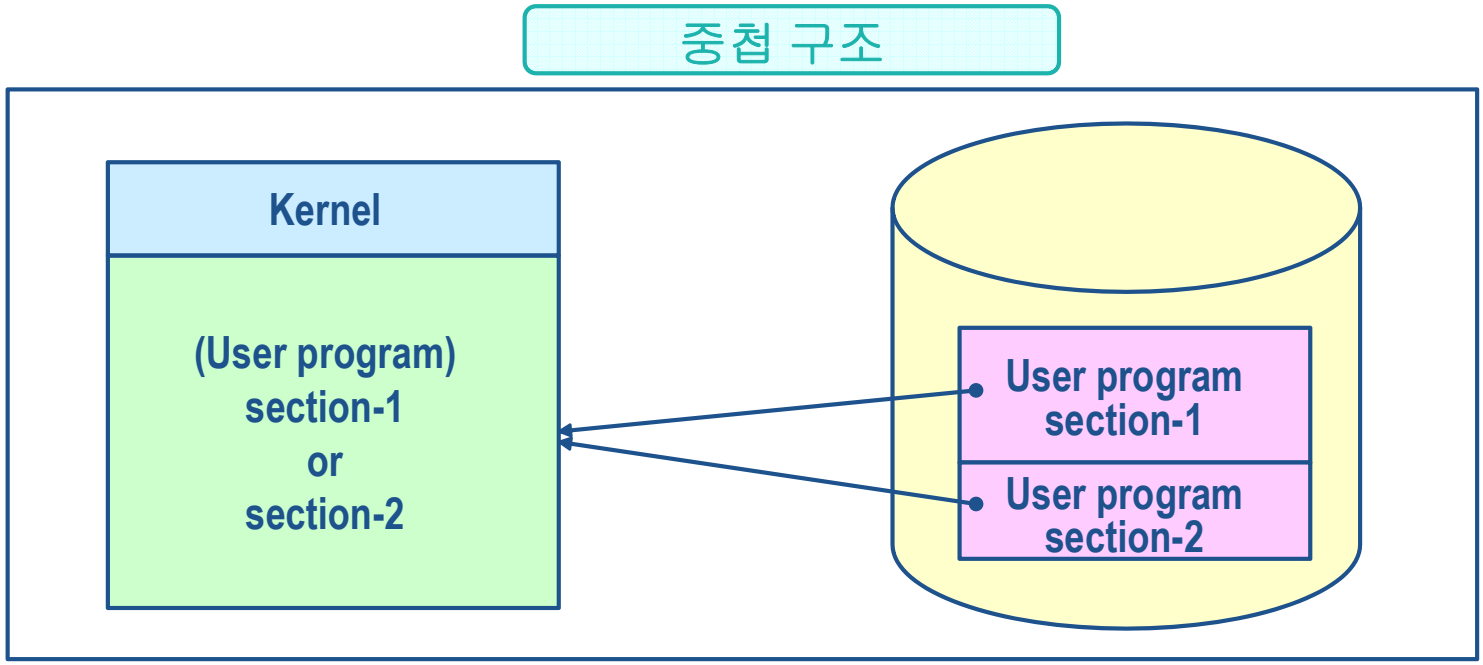
- 단일프로그래밍(uniprogramming) 시스템
  - 항상 시스템 내에 하나의 프로세스만 존재
  - 주기억장치 관리 기법이 매우 단순해 짐



단일프로그래밍 시스템

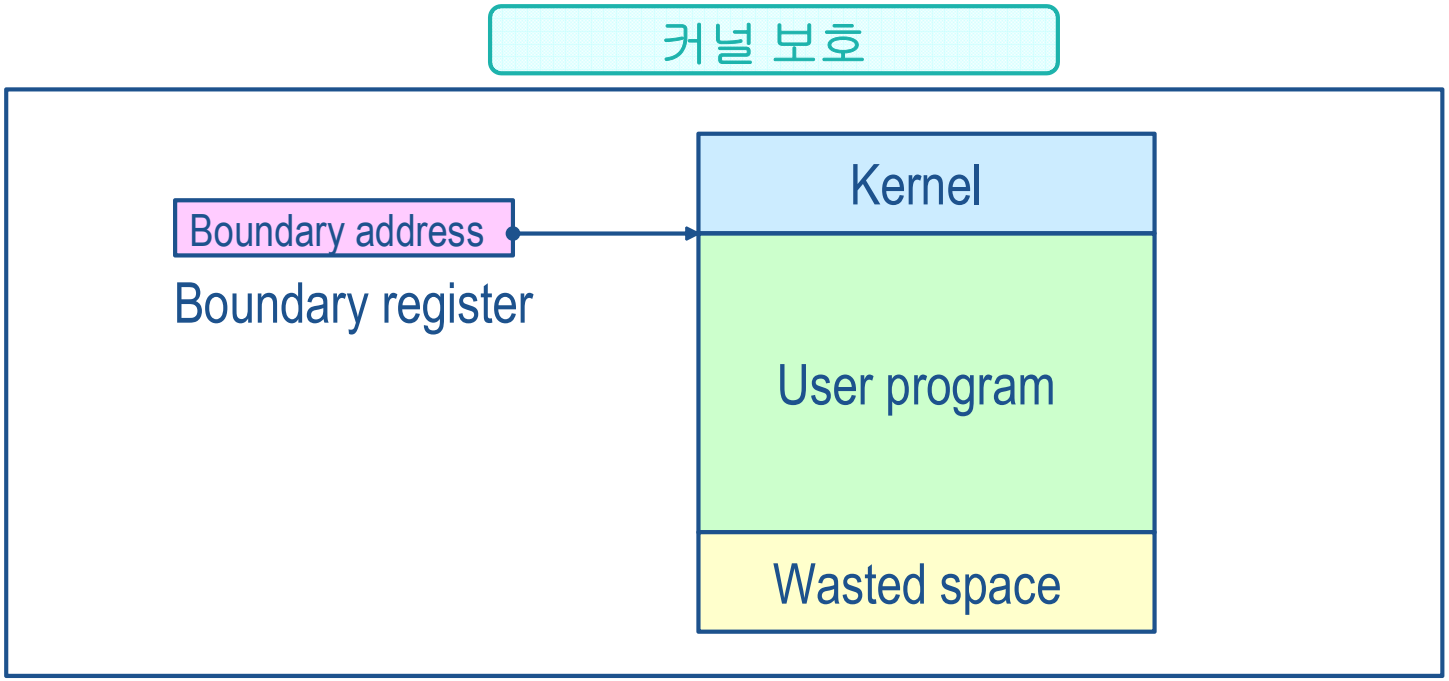
단일프로그래밍 환경에서의 문제점(1)

- 프로그램의 크기가 주기적장치의 가용 공간보다 클 경우
  - 해결
    - 중첩 구조 (overlay structure) 사용
    - 컴파일러 및 링커, 로더의 지원 필요



단일프로그래밍 시스템

- 단일프로그래밍 환경에서의 문제점(2)
  - 사용자 프로세스로부터 커널을 보호하는 기법 필요
    - 해결
      - 경계 레지스터 (boundary register) 사용



단일프로그래밍 시스템

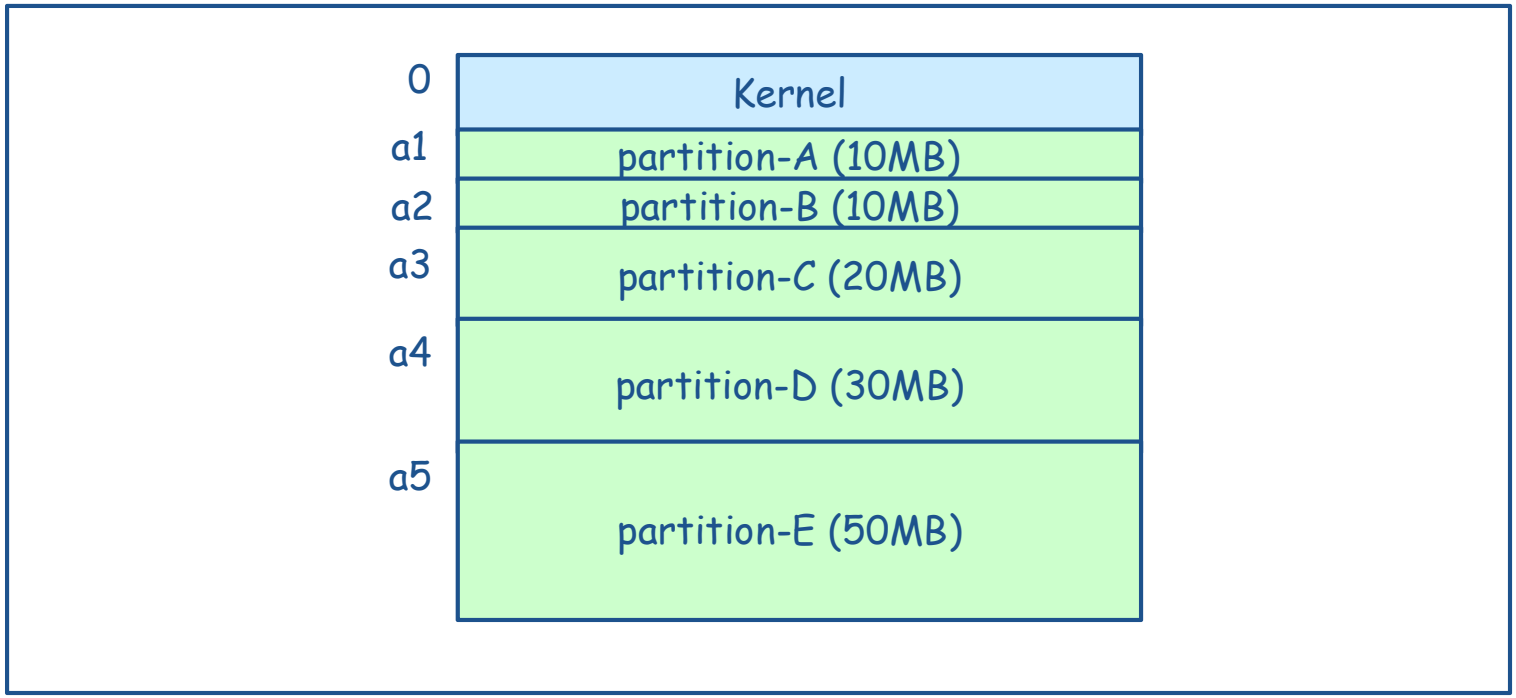
- 단일프로그래밍 환경에서의 문제점(3)
  - 시스템 자원의 낭비
  - 시스템 성능의 저하
    - 해결
      - 다중 프로그래밍 기법 사용
      - 동시에 여러 프로그램들 적재되도록 함

고정 분할 다중프로그래밍

- FPM : Fixed Partition Multiprogramming
  - 주기억장치의 사용자 공간을 미리 여러 개의 영역으로 분할
  - 각 분할 영역에는 항상 하나의 프로그램만 적재 가능
  - 하나의 프로그램이 두 개 이상의 분할 영역 사용 불가
  - 분할영역의 수가 k 일 경우
    - 이 시스템의 다중프로그래밍 정도 : 최대 k

고정 분할 다중프로그래밍

고정분할 다중프로그래밍 시스템의 주기억장치 분할 예



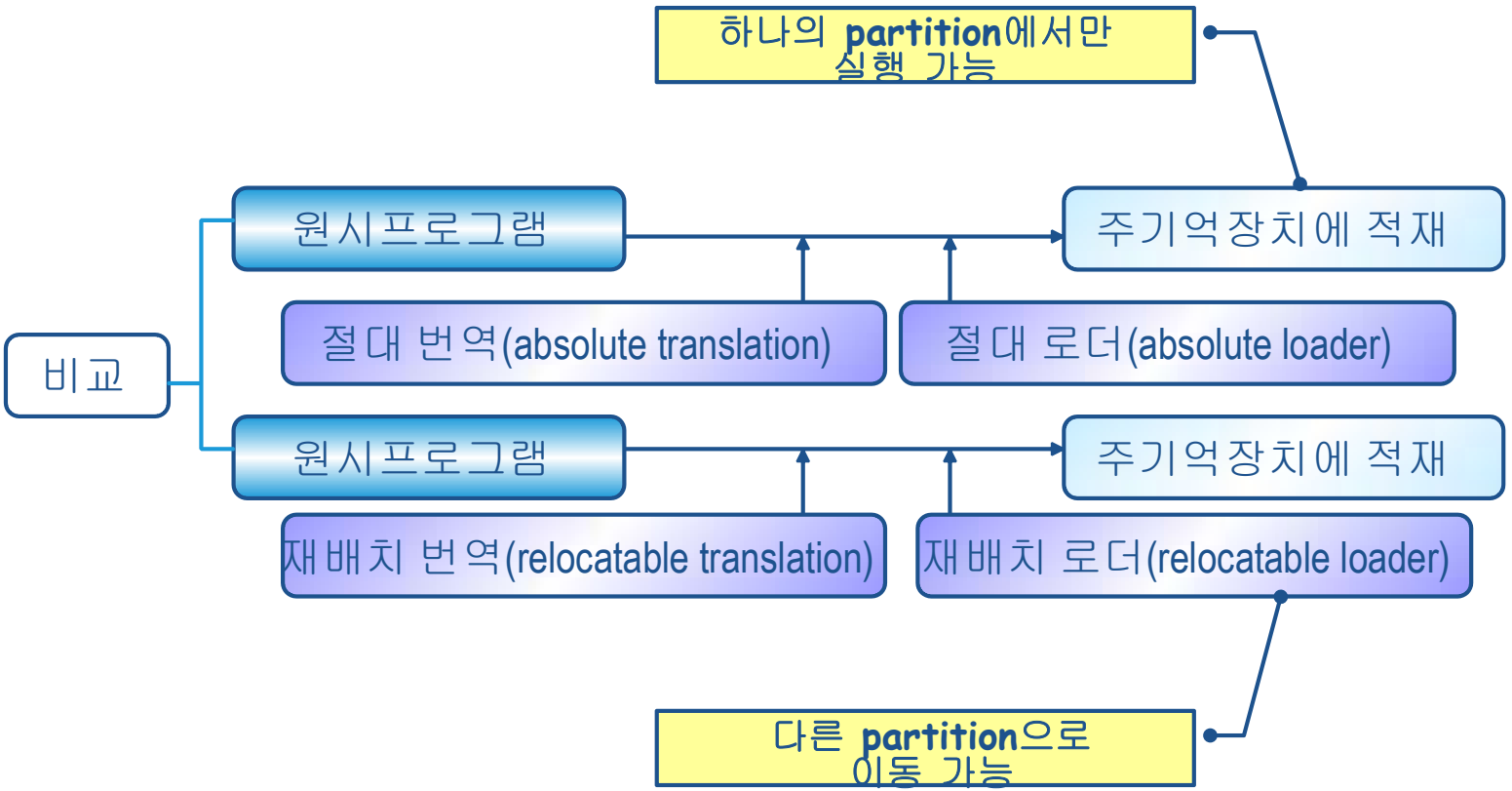
- 주기억장치 관리
  - 고정된 크기의 자료 구조 사용

고정분할 다중프로그래밍을 위한 자료구조

partition	start address	size	current process ID	other fields
A	a1	10 MB	-	···
B	a2	10 MB	-	···
C	a3	20 MB	-	···
D	a4	30 MB	-	···
E	a5	50 MB	-	···

◆ 적재할 프로그램 배치 시에 커널의 주기억장치 관리 모듈이 참조함

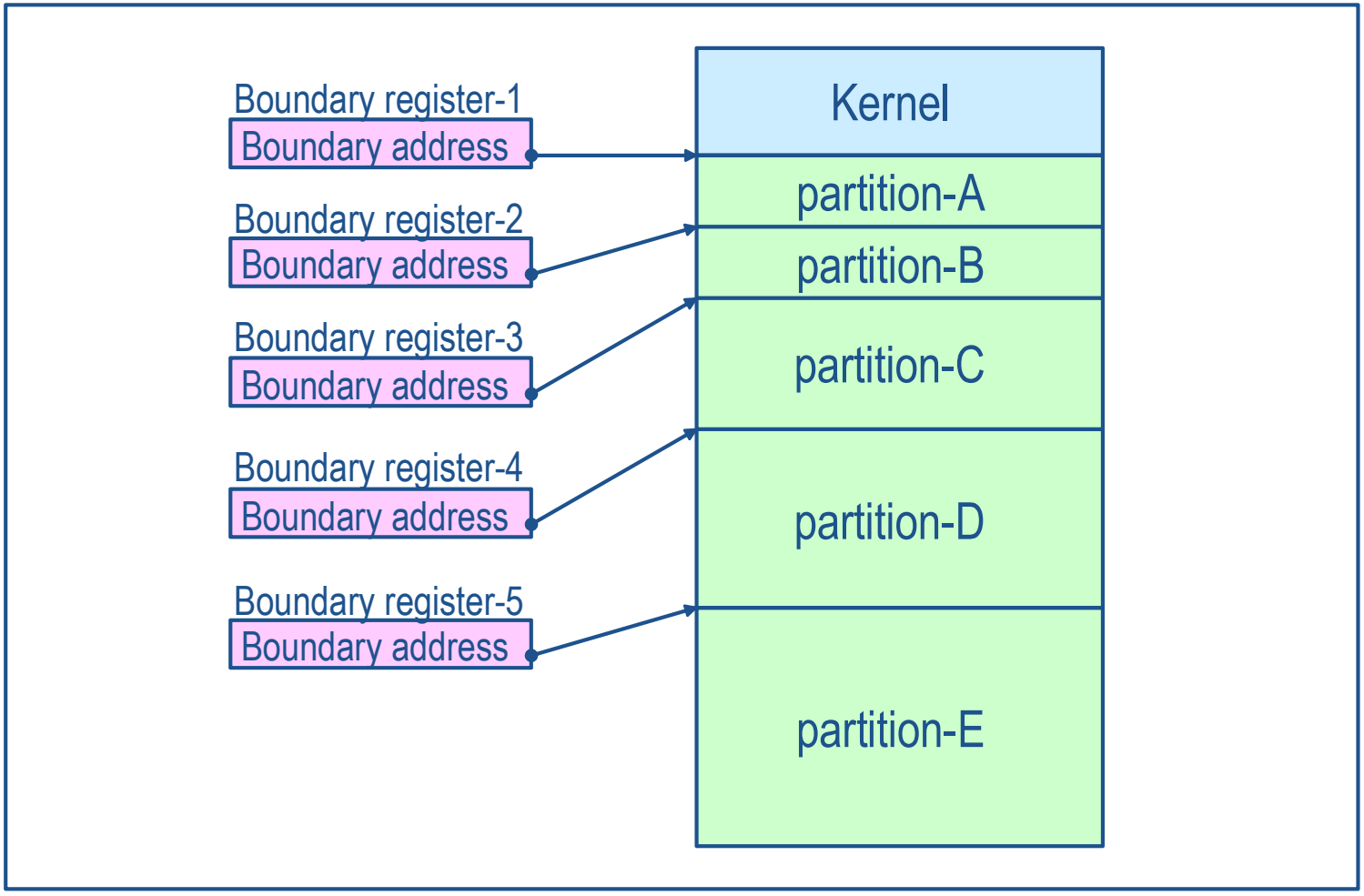
- 프로그램 재배치


하나의 partition에서만 실행 가능
원시프로그램
주기억장치에 적재
절대 번역(absolute translation)
절대 로더(absolute loader)
비교
원시프로그램
주기억장치에 적재
재배치 번역(relocatable translation)
재배치 로더(relocatable loader)
다른 partition으로 이동 가능

FPM 기법의 문제점

- 사용자 프로그램의 크기가 최대 분할 영역의 크기보다 큰 경우
  - 분할 영역 별로 중첩 구조를 사용하여 해결 가능
- 커널과 다른 프로세스들에게 할당된 분할 영역들에 대한 보호 필요
  - 여러 개의 경계 레지스터를 사용하여 해결 가능
- 각 분할 영역마다 낭비되는 공간 발생
  - 단편화(fragmentation)
    - 공간이 낭비되는 현상
  - 내부 단편화 (internal fragmentation)
    - 분할 영역 내에서 발생하는 공간의 낭비 현상
  - 외부 단편화 (external fragmentation)
    - 공간 용량의 문제로 한 분할 영역 전체가 낭비되는 현상

다중프로그래밍 시스템에서의 커널 및 사용자 영역 보호



고정 분할 다중프로그래밍 기법 요약

- 주기억장치 공간을 미리 분할
- OS 입장에서 주기억장치 관리 용이
- 오버헤드 작음
- 시스템 자원의 낭비 초래 가능
  - 분할영역의 개수가 고정되므로 작은 규모의 프로세스들만 실행되는 경우 비효율적임
- 각 분할 영역마다 내부 단편화 현상 발생 가능

가변 분할 다중프로그래밍

- VPM : Variable Partition Multiprogramming
  - 초기에 주기억장치의 사용자 공간 전체를 하나의 분할 영역으로 설정
  - 프로세스들의 활동에 따라 분할 형태를 동적으로 변화 시킴
  - 분할 영역 내의 단편화 현상 발생 않음
  - 프로세스들은 연속 공간을 할당 받음
  - 관리 오버헤드 증가

가변 분할 다중프로그래밍

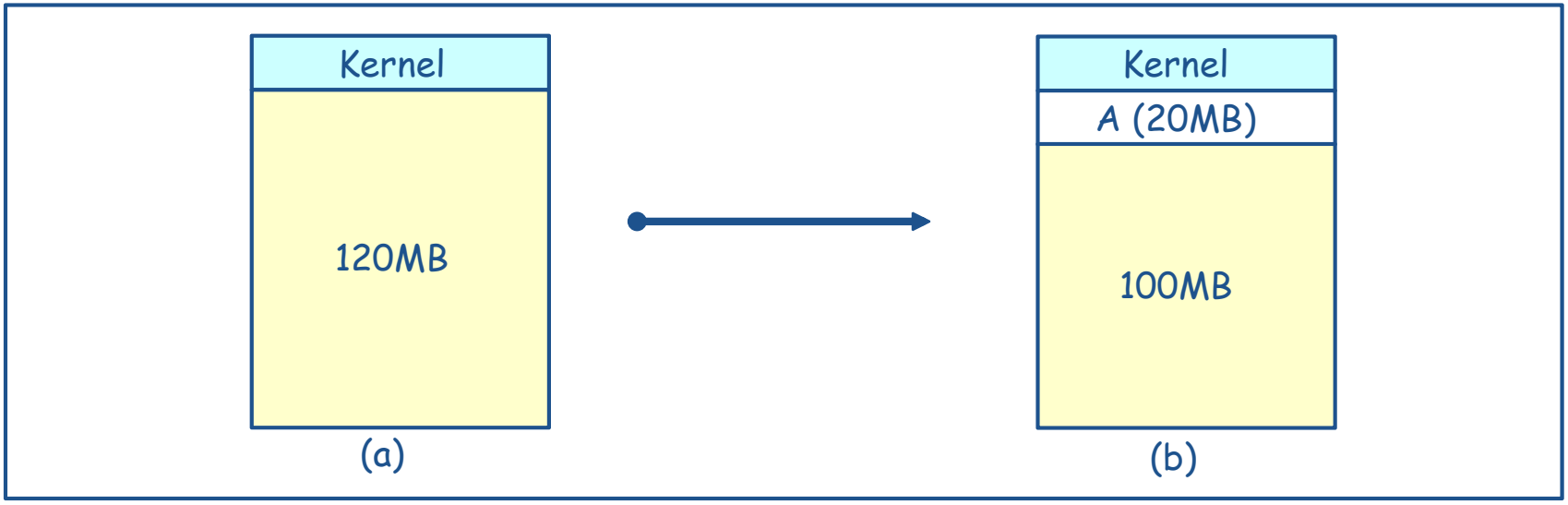
주기억장치 할당 과정 및 분할 과정의 예

- 시스템의 가정

  - 사용자 공간 크기 : 120 MB

  - (a) : 초기 상태
  - (b) : 프로세스 A(20MB) 가 적재 된 후
  - (c) : 프로세스 B(10MB) 가 적재 된 후
  - (d) : 프로세스 C(25MB) 가 적재 된 후
  - (e) : 프로세스 D(20MB) 가 적재 된 후
  - (f) : 프로세스 B가 주기억장치를 반납한 후
  - (g) : 프로세스 E(15MB) 가 적재 된 후
  - (h) : 프로세스 D가 주기억장치를 반납한 후

가변 분할 다중프로그래밍의 주기억장치 할당 예



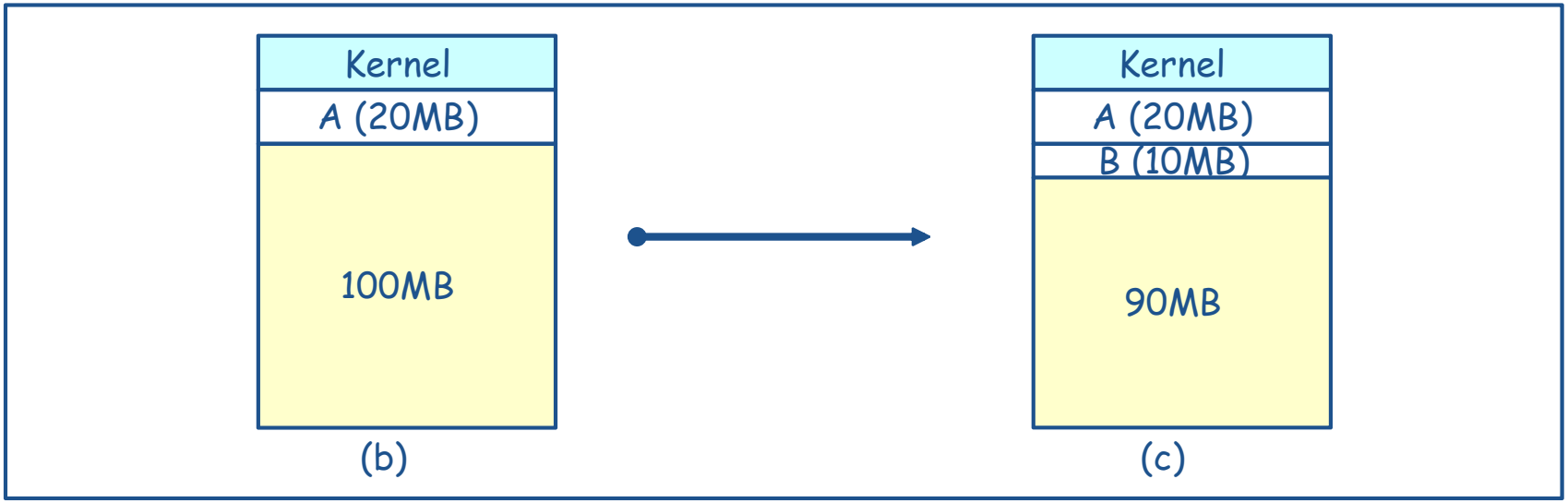
partition	start address	size	current process ID	other field
1	u	120	none	· · ·

(a)

partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	100	none	· · ·

(b)

가변 분할 다중프로그래밍의 주기억장치 할당 예



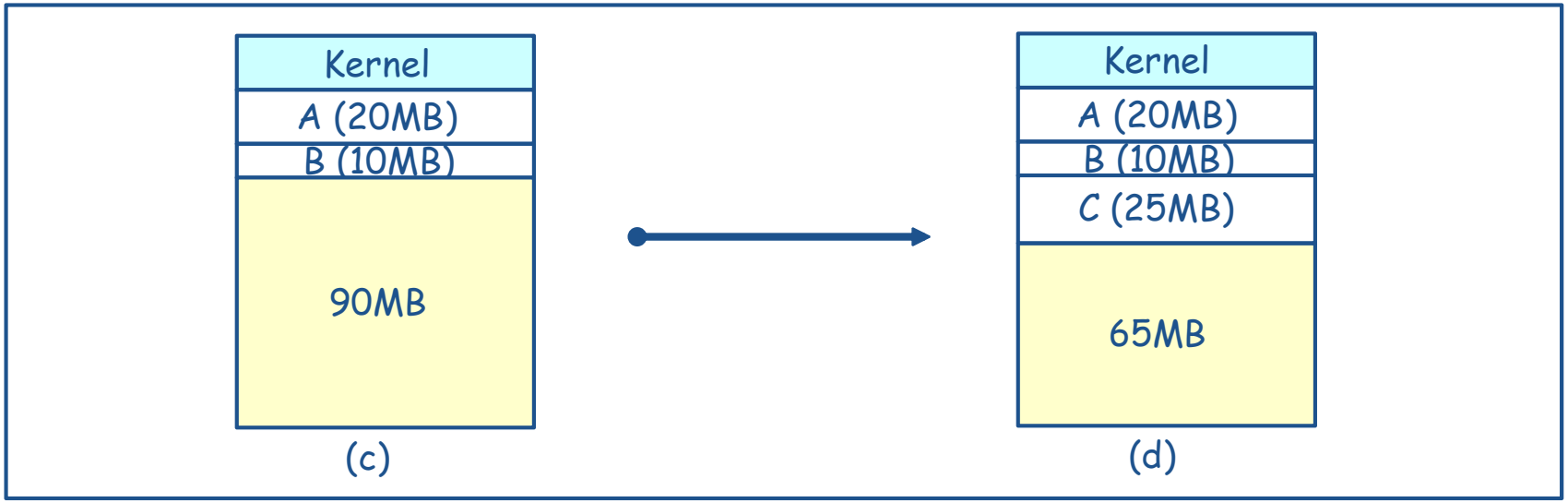
partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	100	none	· · ·

(b)

partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	10	B	· · ·
3	u+30	90	none	· · ·

(c)

가변 분할 다중프로그래밍의 주기억장치 할당 예



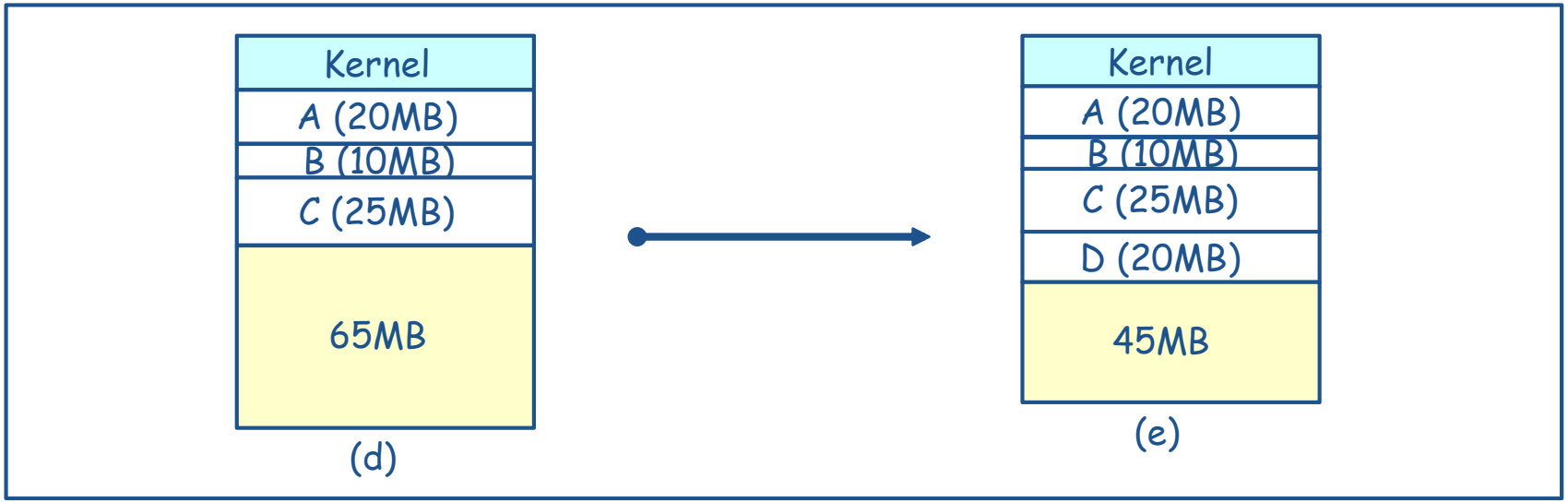
partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	10	B	· · ·
3	u+30	90	none	· · ·

(c)

partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	10	B	· · ·
3	u+30	25	C	· · ·
4	u+55	65	none	· · ·

(d)

가변 분할 다중프로그래밍의 주기억장치 할당 예



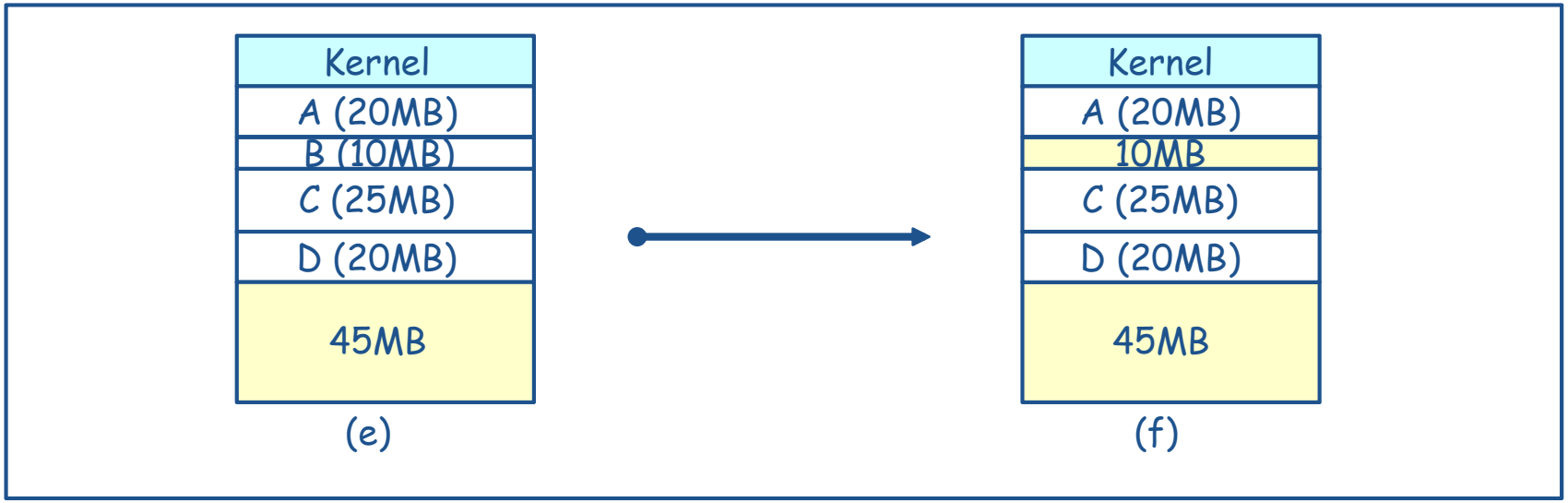
partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	10	B	· · ·
3	u+30	25	C	· · ·
4	u+55	65	none	· · ·

(d)

partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	10	B	· · ·
3	u+30	25	C	· · ·
4	u+55	20	D	· · ·
5	u+75	45	none	· · ·

(e)

가변 분할 다중프로그래밍의 주기억장치 할당 예



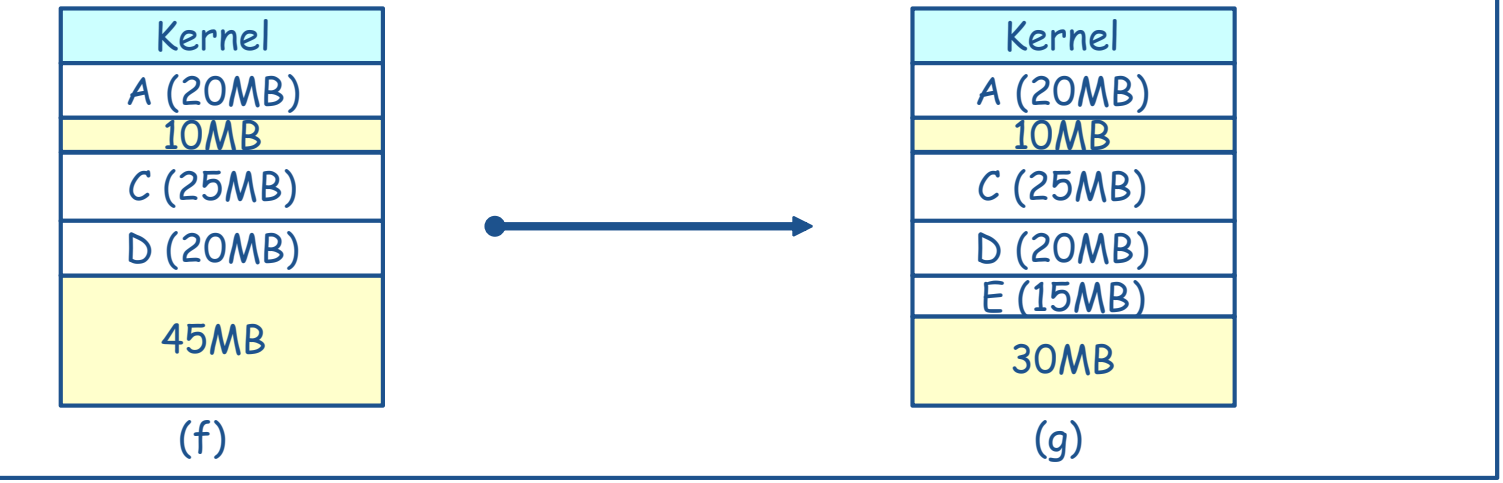
partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	10	B	· · ·
3	u+30	25	C	· · ·
4	u+55	20	D	· · ·
5	u+75	45	none	· · ·

(e)

partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	10	none	· · ·
3	u+30	25	C	· · ·
4	u+55	20	D	· · ·
5	u+75	45	none	· · ·

(f)

가변 분할 다중프로그래밍의 주기억장치 할당 예



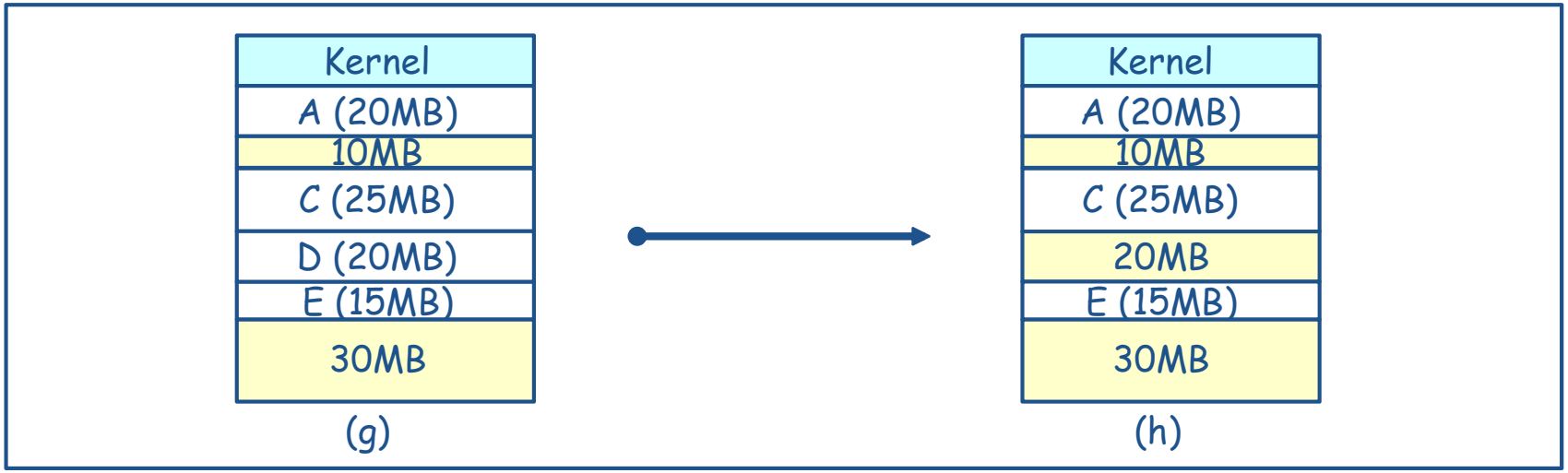
partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	10	none	· · ·
3	u+30	25	C	· · ·
4	u+55	20	D	· · ·
5	u+75	45	none	· · ·

(f)

partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	10	none	· · ·
3	u+30	25	C	· · ·
4	u+55	20	D	· · ·
5	u+75	15	E	· · ·
6	u+90	30	none	· · ·

(g)

가변 분할 다중프로그래밍의 주기억장치 할당 예



partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	10	none	· · ·
3	u+30	25	C	· · ·
4	u+55	20	D	· · ·
5	u+75	15	E	· · ·
6	u+90	30	none	· · ·

(g)

partition	start address	size	current process ID	other field
1	u	20	A	· · ·
2	u+20	10	none	· · ·
3	u+30	25	C	· · ·
4	u+55	20	none	· · ·
5	u+75	15	E	· · ·
6	u+90	30	none	· · ·

(h)

가변 분할 다중프로그래밍

- 배치 기법 (placement policies)
  - 최초 적합 (first-fit) 전략
    - 상태 테이블의 처음부터 차례로 각 분할 영역 정보를 검사
    - 프로그램의 용량보다 크면서 비어 있는 첫번째 분할 영역에 적재
    - 매우 단순하고 오버헤드 적음
  - 최적 적합 (best-fit) 전략
    - 모든 빈 분할 영역들을 검사하여 용량이 새로운 프로그램의 크기보다 큰 분할 영역들 중 가장 작은 용량의 분할 영역에 적재
    - 알맞은 분할 영역 찾는 시간이 오래 걸림
    - 용량이 큰 빈 공간들을 확보 가능
    - 외부 단편화 현상 발생

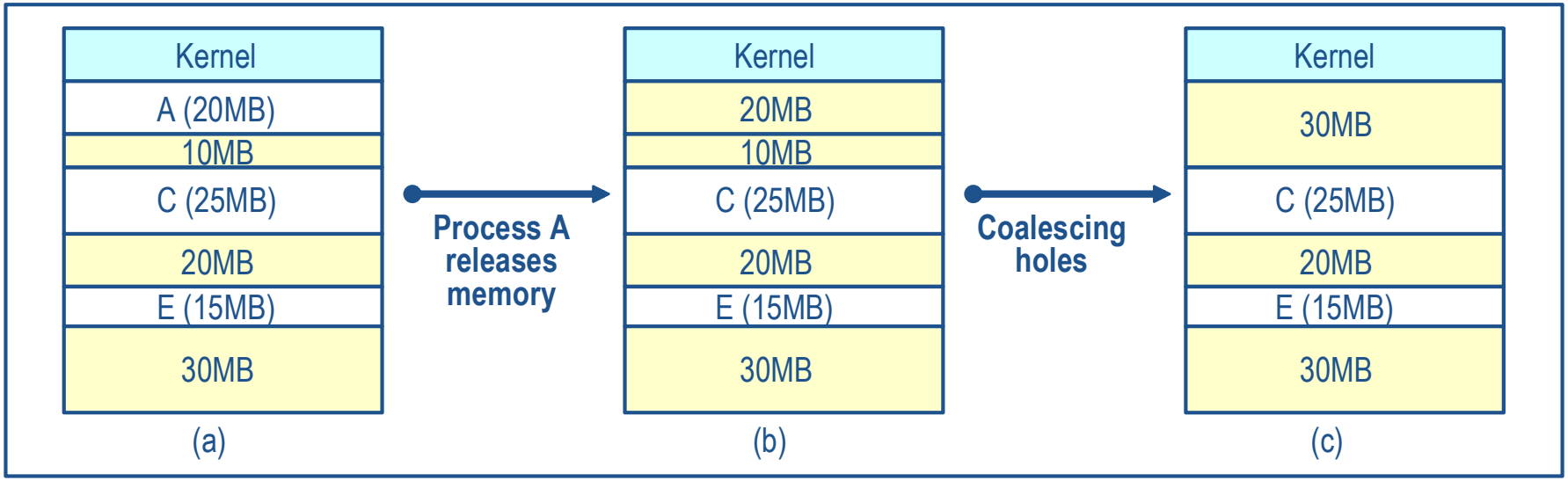
가변 분할 다중프로그래밍

- 배치 기법 (placement policies)
  - 최악 적합 (worst-fit) 전략
    - 주기억장치 상태 테이블 전체를 검사
    - 모든 비어 있는 분할 영역들 중 가장 용량이 큰 분할 영역에 적재
    - 단편화 현상 극소화 가능
    - 대용량의 빈 분할 영역 확보 불가능
  - 순환 최초 적합 (next-fit) 전략
    - 최초 적합 전략과 유사
    - 주기억장치 상태 테이블의 직전 검사 마지막 부분부터 검사 시작
    - 테이블의 마지막에 도달 시 다시 테이블의 처음부터 검사
    - 주기억장치의 각 영역 사용 빈도 균등화
    - 오버헤드 적음

가변 분할 다중프로그래밍

- 인접 공간 통합 (coalescing holes) 작업
  - 인접한 빈 분할 영역들을 하나의 분할 영역으로 통합
- 기억장소 통합 (storage compaction) 작업
  - 모든 빈 분할 영역들을 하나로 통합
  - 프로그램의 적재 공간이 부족할 경우 수행
  - 주기억장치 내의 모든 프로세스들의 재배치 작업 수행 필요
  - 작업 시간 매우 길
  - 많은 시스템 자원을 소비하는 결과 초래

인접 공간 통합의 예 (1)



partition	start address	size	current process ID
1	u	20	A
2	u+20	10	none
3	u+30	25	C
4	u+55	20	none
5	u+75	15	E
6	u+90	30	none

(a)

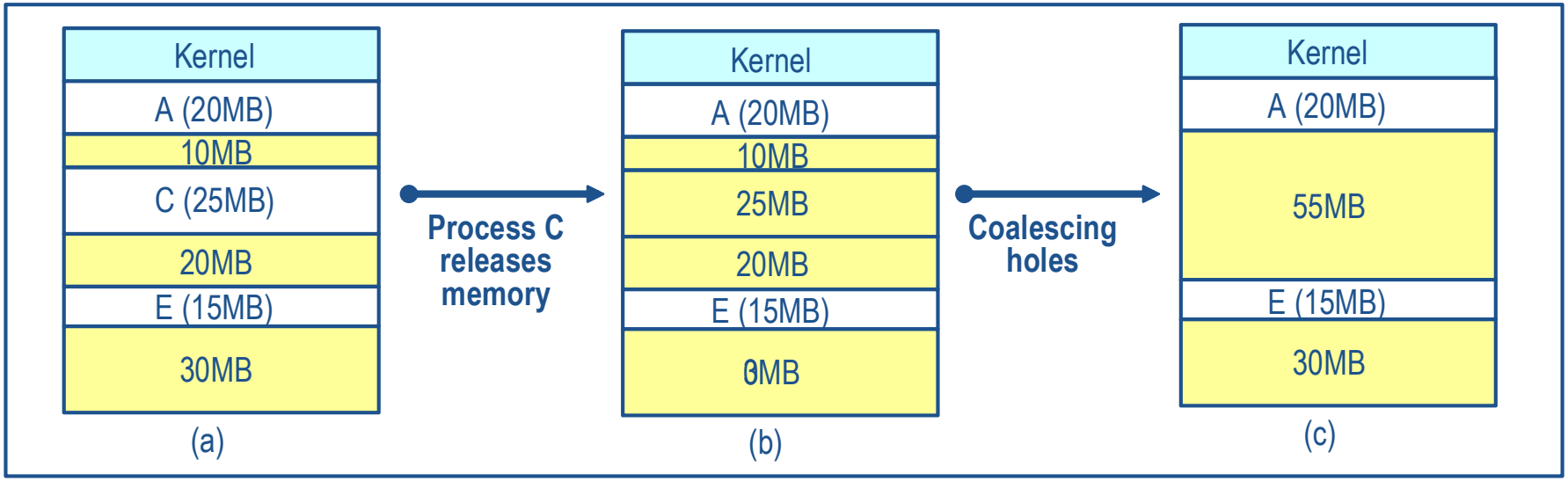
partition	start address	size	current process ID
1	u	20	none
2	u+20	10	none
3	u+30	25	C
4	u+55	20	none
5	u+75	15	E
6	u+90	30	none

(b)

partition	start address	size	current process ID
1	u	30	none
2	u+30	25	C
3	u+55	20	none
4	u+75	15	E
5	u+90	30	none

(c)

인접 공간 통합의 예 (2)



partition	start address	size	current process ID
1	u	20	A
2	u+20	10	none
3	u+30	25	C
4	u+55	20	none
5	u+75	15	E
6	u+90	30	none

(a)

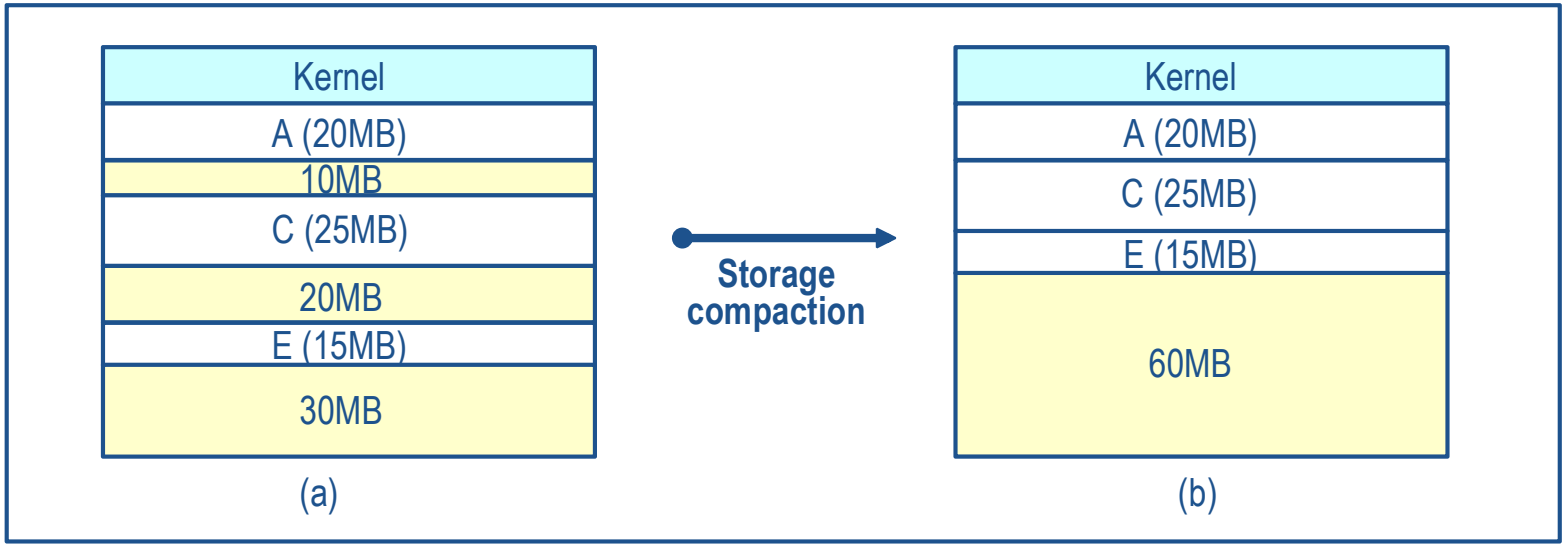
partition	start address	size	current process ID
1	u	20	A
2	u+20	10	none
3	u+30	25	none
4	u+55	20	none
5	u+75	15	E
6	u+90	30	none

(b)

partition	start address	size	current process ID
1	u	20	A
2	u+20	55	none
3	u+75	15	E
4	u+90	30	none

(c)

기억 장소 통합의 예



partition	start address	size	current process ID
1	u	20	A
2	u+20	10	none
3	u+30	25	C
4	u+55	20	none
5	u+75	15	E
6	u+90	30	none

(a)

partition	start address	size	current process ID
1	u	20	A
2	u+20	25	C
3	u+45	15	E
4	u+60	60	none

(b)

Storage Swapping

- 기억장소 교체 (storage swapping) 기법
  - 실행이 종료되지 않은 프로세스가 할당받은 기억장소를 중간에 반납받을 수 있도록 하는 기법
  - 종류
    - 프로세서 할당/반납 시 주기억장치와 함께 할당/반납
    - 새 프로세스에게 할당할 기억 공간이 부족할 경우에만 반납
      - 여러 프로세스들이 동시에 주기억장치 내 존재 가능

결론

- 주기억장치 관리 기법
  - 단일프로그래밍 시스템
  - 다중프로그래밍 시스템
    - 가정
      - 기억장소 연속적(contiguously) 할당 기법
      - 주기억장치 할당 시 프로그램 코드나 데이터 등 모두 적재
    - 주기억장치의 효율적인 관리 위한 기법
      - 고정 분할 다중프로그래밍 기법
      - 가변 분할 다중프로그래밍 기법
      - 기억장소 교체를 이용한 다중프로그래밍 기법