<Chapter 3. 기억장치 관리>

기억장치의 종류

프로그램과 데이터는 직접 실행되거나 참조되기 위하여 주기억장치 내에 있어야 한다.

1. 주기억 장치 : 용량이 제한되어 있고 값이 비싸다.

2. 보조기억장치 : 주기억장치보다 값이 싸며 대량의 프로그램과 데이터를 처리할 수 있지만, 처리 속 도가 느리다. ´ 자기테이프, 하드디스크, 플로피디스크, CD-ROM, DVD-ROM등의 광디스크, USB포트에 연결가능한 플래 시 메모리

3. SSD(Solid State Drive) : 하드디스크의 한계를 극복하기 위해 등장한 플래시가 들어가는 보조기억 장치.

**특징** : 반도체 메모리를 내장하고 있어 처리속도가 빠르고 소음이 없으며, 전력소모량이 적다. ex) 휴대용 컴 퓨터 ´ 전원이 꺼지더라도 기록된 데이터가 보존되는 플래시 메모리 기반의 모델

기억장치 관리의 발전

테이블이(가) 표시된 사진

자동 생성된 설명

주소 바인딩(address binding) : 주소를 변수로 표현하는 논리적 주소(logical address)가 실행되기 위해서 물리적 주소로 사상하는 것

바인딩 시점에 따른 구분

- **컴파일 시간(compile time) 바인딩**: 실행 시 위치가 바뀌면 다시 컴파일 해야하는 주소 바인딩 ´

- **적재 시간(load time) 바인딩** : 프로그램이 기억장치에 적재되는 시간에 적재기(loader)에 의해 이루어지 는 주소 바인딩

- **실행 시간(execution time) 바인딩** : 프로세스가 실행되는 동안에 기억장치의 한 세그먼트에서 다른 세그먼 트로 옮겨질 경우, 실행 시간에 이루어지는 주소 바인딩

논리적 주소와 물리적 주소

**v 논리적 주소** : 중앙처리장치가 생성하는 주소

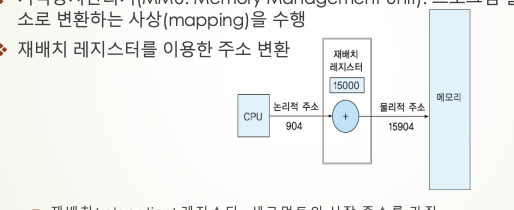
v **물리적 주소** : 기억장치가 취급하는 주소

**v 가상 주소(virtual address), 논리적 주소 공간(logical address space):** 프로그램에 의해 생성된 모든 논리 주소의 집합

v **물리적 주소 공간(physical address space**): 논리적 주소에 상응하는 모든 물리적 주소의 집합

v **기억장치관리기(MMU: Memory Management Unit)**: 프로그램 실행 중 논리적 주소를 물리적 주 소로 변환하는 사상(mapping)을 수행

v **재배치 레지스터를 이용한 주소 변환**



v **재배치(relocation) 레지스터** : 세그먼트의 시작 주소를 가짐

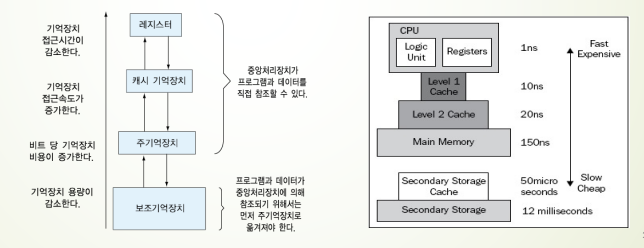
o 논리적 주소가 들어올 때마다 모든 주소에 재배치 레지스터가 가지고 있는 값을 더해서 물리적 주소를 만듦 ´ 사용자는 논리적 주소 사용하고 기억장치 하드웨어는 논리적 주소를 물리적 주소로 바꿈

**캐시(cache)** : 주기억장치보다 접근 속도가 훨씬 빠른 기억 매체

v 기억장치 계층 구조의 성능과 유용성에 있어 상당한 증대를 이룬 부가적인 단계

v 가격이 주기억장치에 비해 상당히 비싸기 때문에 상대적으로 적은 용량의 캐시가 사용

**기억 장치의 계층 구조**



**v 인출(fetch) 기법** : 주기억장치에 적재할 다음 프로그램이나 데이터를 언제 가져올 것인가를 결정 하는 기법 ´

- **요구 인출(demand fetch) 기법** : 실행 프로그램에 의해 어떤 프로그램이나 데이터가 참조될 때 그것을 주 기억장치로 옮기는 방법

- **예상 인출(anticipatory fetch) 기법** : 앞으로 요구될 가능성이 큰 데이터 또는 프로그램을 예상하여 주기억 장치로 미리 옮기는 방법

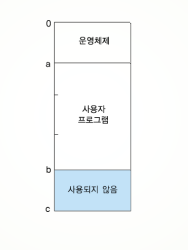
**v 배치(placement) 기법** : 새로 인출된 데이터나 프로그램을 주기억장치의 어디에 위치시킬 것인가 를 결정하는 기법

**최초 적합(first-fit), 최적 적합(best-fit) 및 최악 적합(worst-fit)** 등

**v 교체(replacement) 기법** : 새로 들어온 프로그램이 들어갈 장소를 마련하기 위해서 어떤 프로그 램 및 어떤 데이터를 제거할 것인가를 결정하는 기법

**단일 사용자 연속 기억장치 할당**

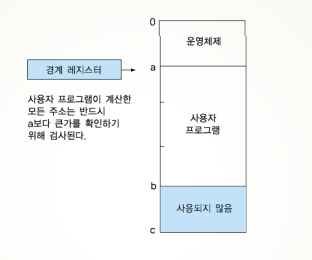
v **기억장치 구성**



v 사용자 프로그램의 크기는 주기억장치의 용량보다 클 수 없으나, 오버레이 기법에 의해 주기억장치 보다 더 큰 프로그램의 실행 가능

- **오버레이** : 프로그래머가 한정된 주기억장치를 확장하여 사용할 수 있는 방법을 제공하는 것

**단일 사용자 연속 기억장치 할당에서 기억장치의 보호**



v 중앙처리장치 내에 하나의 경계 레지스터를 두어 사용자 프로그램이 기억장치 주소를 참조할 때마 다 사용자가 운영체제를 파괴하지 않도록 확인하는 검사

v 경계 레지스터는 운영체제의 마지막 명령의 주소를 저장하고 있어 사용자가 운영체제에 들어가려고 하면 명령어의 실행은 중단되고 적절한 오류 메시지와 함께 작업을 끝냄

v 사용자는 슈퍼바이저 호출(SVC: Supervisor Call) 명령을 호출하여 운영체제가 제공하는 서비스를 요청

**단일 사용자 기억장치 할당 기법의 장단점**

**장점** : 설계 간단, 이해하기 쉬움

**단점** : 한 사용자만이 전체 시스템을 차지하여 기억장치와 주변장치의 낭비가 심함, 입출력 수행 동 안 중앙처리장치가 유휴(idle)상태에 놓여 컴퓨터 자원 낭비 초래함

**다중 프로그래밍 시스템**

- 여러 개의 작업이 동시에 컴퓨터의 주기억장치 내에 존재

- 여러 사용자들이 동시에 시스템의 자원을 획득하기 위하여 경쟁 ´ 고정 분할 기억장치 할당 기법 사용

- **고정 분할 기억장치 할당(fixed partition memory allocation)** : 주기억장치를 일정 수의 고정된 크기들로 분할하여 실행 중인 여러 프로세스에게 할당

- **분할(partition)**: 하나의 단일 작업이 적재될 수 있는 일정한 크기의 기억장치 영역

=> 분할의 수에 의해 다중 프로그래밍의 정도 결정 ´ 한 분할이 비어 있을 경우에만 새로운 작업 하나를 작업 큐에서 선택하여 빈 분할에 적재

**절대 번역 및 로딩 (고정분할)**

v 작업이 시스템에 들어 올 모든 작업들을 그들의 기억장치 요구량에 따라 분류

v 절대 어셈블러(absolute assembler)와 절대 컴파일러(absolute compiler)에 의해 번역되어서 지 정된 분할에서만 실행

**재배치 가능 번역 및 로딩 (고정분할)**

v 모든 작업을 하나의 작업 큐에 넣음

v 사전에 어떤 프로그램이 수행될 분할을 결정하지 않고, 그 프로그램이 수행될 시점에 충분한 크기 의 분할을 선택하여 그곳에 적재하는 것

v 작업은 적합한 크기의 사용 가능한 분할이면 어느 분할이든 배치 가능

**시스템 보호** **(고정분할)**

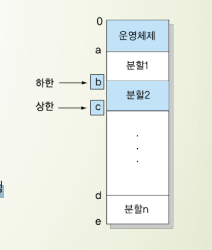
v 여러 사용자가 시스템을 공유하는 상황에서는 시스템에 대한 보호 방안

v 주로 여러 개의 경계 레지스터(boundary register)를 이용하여 보호

작업이 적재된 분할은 두 개의 경계 레지스터를 통해 하한과 상한을 나타냄

운영체제를 호출할 필요가 있는 사용자는 슈퍼바이저 호출 명령을 사용하여 수행 ´

다중 프로그래밍 시스템에서의 프로그램 및 데이터 보호



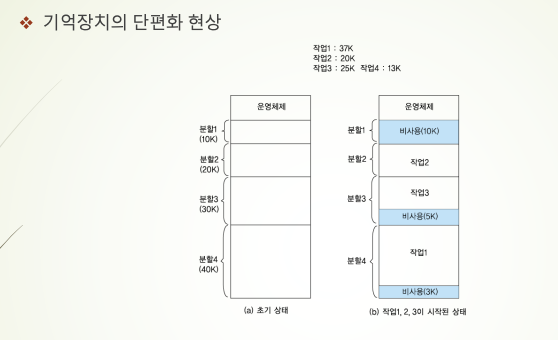
**분할 공간의 크기와 단편화 (문제점) (단점) (고정분할)**

v 고정 분할 기억장치 할당 기법의 운영 시에 고려해야 할 문제 중의 하나는 분할의 크기를 결정하는 것

즉, 얼마나 큰 영역을 몇 개나 만들 것인가를 결정하는 것

v **기억장치 단편화(fragmentation)** 현상이 발생 ´ 사용자 작업의 크기가 분할에 정확히 맞지 않거나, 또는 분할이 너무 작아서 대기 중인 어떤 작업도 이 분할 에 적재될 수 없는 경우에 발생

기억장치의 단편화 현상 **(고정분할)**



**내부 단편화(internal fragmentation)**: 필요한 양보다 더 큰 메모리가 할당이 되어서 할당 된 메모리 내부에 사용하는 메모리 공간 이외에 사용하지 않는 메모리 공간이 발생 o 작업3과 작업1이 각각 적재된 분할3과 분할4에서 각각 사용되지 않은 비사용 부분 5K와 3K가 발생

**외부 단편화(external fragmentation):** 분할1과 같이 어떤 분할이 사용되지 않고 이용 가능하지만 대기 중 인 작업에게는 너무 작아서 사용할 수 없을 때 발생하는 현상

**고정 분할 기억장치 할당 기법에서의 주된 문제점**

v 단편화 현상을 줄이기 위하여 가장 합리적인 분할의 크기를 결정하는 것

**가변 분할 기억장치 할당 (variable partition memory allocation)** : 작업들이 필요로 하는 만큼의 기억 공간을 할당

**기억 공간의 효율화**

v 가변 분할 다중 프로그래밍에서의 단편화 현상

테이블이(가) 표시된 사진

자동 생성된 설명

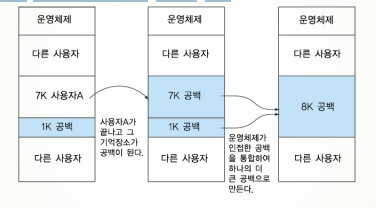
**공백의 합병**

공백이 비어 있는 다른 기억 공간과 인접되어 있는지를 점검

o 만일 인접되어 있다면, 빈 기억 장소 리스트에 새로운 공백으로 기록

o 이미 있는 공백과 새로운 인접한 공백을 합병하여 하나의 공백으로 기록

**가변 분할 다중 프로그래밍에서의 공백의 합병**



**기억 장소의 집약**

현재 사용되고 있는 모든 기억 공간을 주기억장치의 한쪽으로 모음으로써, 가변 분할 다중 프로그래밍에서 존재하는 여러 개의 작은 공백들을 하나의 커다란 기억 공간으로 통합

기억 장소의 집약을 일반적으로 쓰레기 수집(garbage collection)이라고 부르기도 함

기억 장소의 집약 방법에 따른 비교

테이블이(가) 표시된 사진

자동 생성된 설명

**기억장치 배치 기법**

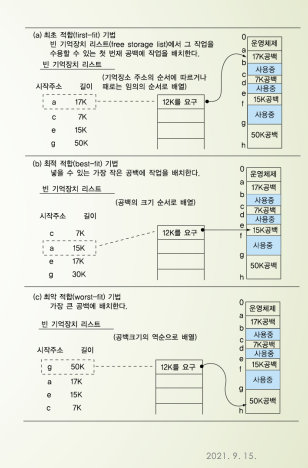
v 입력되는 프로그램과 데이터를 주기억장치의 어느 곳에 넣을 것 인가를 결정

v **최초 적합 기법(first-fit strategy):** 작업을 적재함에 있어, 그것을 수용할 수 있는 주기억장치의 첫 번째 유용한 공백을 우선적으 로 선택하는 방법

v **최적 적합 기법(best-fit strategy):** 어떤 작업을 적재할 기억 공 간을 선택함에 있어, 그 작업에 가장 적합한 공간을 선택함으로 써 기억장치의 단편화를 최소로 하는 방법

v **최악 적합 기법(worst-fit strategy):** 프로그램을 주기억장치 내 에서 가장 알맞지 않은 공백, 즉 가장 큰 공백에 배치하는 것

v **다음 적합 기법(next-fit strategy):** 최초 적합은 항상 메모리의 시작부터 검색을 시작하기 때문에 목록 상단 근처에 작은 공백 을 모으는 경향이 있다. 최초 적합 전략의 간단한 수정은 이전 검색이 중지 된 지점에서 검색을 시작하는 것이다. 이 방법을 회 전 최초 적합(rotating-first-fit)이라고도 한다.

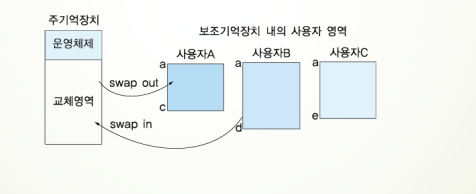


**기억장치 교체 (swapping)**

기억장치를 차지한 작업은 입출력이나 인터럽트 등이 발생하여 더 이상 계속될 수 없을 때 까지 짧은 기간 동안 사용한 후, 필요에 따라 그 작업은 제거(swap out)되고 다시 다음 작업 이 적재(swap in)된다.

하나의 작업은 완료되기까지의 주기억장치와 보조기억장치를 반복하는 교체 과정을 여러 번 거침으로써 수행

다중 프로그래밍 시스템에서의 기억장치 교체



**주기억장치(main memory) 구성 및 관리를 위한 방법**

v 단일 사용자 연속 기억장치 할당

v 고정 분할 기억장치 할당: 절대 번역 및 로딩

v 고정 분할 기억장치 할당: 상대 번역 및 로딩

v 가변 분할 기억장치 할당

v 기억장치 교체 q 주소 바인딩

v 시점에 따른 구분 : 컴파일시간 바인딩, 적재시간 바인딩, 실행시간 바인딩

**기억장치 관리 기법**

v 인출(fetch) 기법 v 배치(placement) 기법 v 교체(replacement) 기법

**인출기법** v 요구 인출(demand fetch) 기법 v 예상 인출(anticipatory fetch) 기법

**배치 기법**

v 최초 적합(first-fit) 기법 v 최적 적합(best-fit) 기법 v 최악 적합(worst-fit) 기법 v 다음 적합(next-fit) 기법

**교체 기법**