**메모리 관리 (Memory Management)**

**계층구조**

Register -> cache -> main memory -> storage

**Block** – 보조기억장치와 주기억장치 사이의 데이터 전송 단위

**Word** – 주기억장치와 레지스터 사이의 데이터 전송 단위

**Address Binding**-프로그램의 논리 주소를 실제 메모리의 물리 주소로 매핑하는 작업

**Compile time binding** –컴파일러가 메모리에 적재될 위치를 알 수 있는 경우

**Loading time binding** – 메모리 적재 위치를 컴파일 시점에서 모르면 대체 가능한 상대 주소를 생성

**Run-time binding** – address binding을 수행시간까지 연기, Hw의 도움의 필요

**Dynamic Loading**-모든 루틴 교체 가능한 형태로 디스크에 저장, 실제 호출 전까지 루틴을 적재하지 않음

**Swapping**- swap in, swap-out

**Memory allocation**

1. **Continuous memory allocation**: 프로세스를 하나의 연속된 메모리 공간에 할당

**- Uni-programming**: 프로세스가 하나만 올라가는 경우

단점: 프로그램 크기가 메모리 크기보다 클 때, 커널 보호, 공간 낭비

해결법: overlay structure, 경계 register 사용, multi-programming 사용

* **Multi-programming**:

a. fixed partition multiprogramming: 메모리를 공간을 고정된 크기로 분할

-Fragmentation (단편화): internal fragmentation- 내부 단편화, external fragment- 외부 단편화(메모리 낭비)-> 시스템 자원 낭비

b. variable partition multiprogramming- 초기에 전체가 하나의 영역, 공간을 동적으로 분활

배치 전략: first-fit-충분한 크기를 가진 첫번쨰 partition을 선택, 공간 활용은 떨어짐

Best-fit: 가장 작은 partition을 선택, 탐색 시간이 오래 걸림

Worst-fit: 가장 큰 partition을 선택, 탐색 시간이 오래 걸림

Next-fit: 마지막으로 탐색한 위치부터 탐색

Coalescing holes(공간 통합), Storage compaction(메모리 압축)

1. **Non-continuous allocation:**

**Virtual Storage(Memory)-**프로그램을 여러 개의 block으로 분할, 필요한 block들만 메모리에 적재

**Addressing mapping:** virtual address-> real address

**Block mapping:** BMT 커널 안에 존재

1. Paging system: 프로그램을 같은 크기의 블록으로 분할/ 메모리를 block size로 미리 분할

필요한 page만 page frame에 적재하여 사용

Paging mapping overhead-추가적인 메모리 접근 사용

Associate mapping TLB

Directing mapping – PMT (page map table)

Frame table

Page sharing- 여러 프로세스가 특정 page를 공유 가능, 공유 가능 page, protection bit 사용

1. Segmentation system: 논리적으로 프로그램을 block으로 분할(segment) 크기는 서로 다를 수 있음

미리 분할 하지 않음 No internal fragmentation but external fragmentation은 발생 가능

SMT (segment map table)

Segment sharing/protection 논리적으로 분할되어 있어 공유 및 보호가 용이함

1. Hybrid paging/segmentation system: HW 비용을 줄이고 associative mapping의 장점, 작은 TLB사용