01.11

📌제네릭이란?

형 변환 시, 타입 검증을 위해 사용되는 자바 문법입니다.(jdk 1.5~) 제네릭을 사용할 경우, 제네릭 타입에 대한 추가 형변환이 필요 없습니다.

메서드에서 제네릭 타입 파라미터를 받을 때

- Wildcard(?) or Bounded Wildcard(? extends T, ? super T) 사용
  - 그러나 Wildcard는 조회용 매개 변수에만 사용하여 변수의 값을 가져올 수는 있지만, 특정 타입으로 값을 정하는 것은 불가하다.
- 메소드를 제네릭하게 선언
  - 리턴 타입 앞에 제네릭 타입을 선언한다. 이 때 와일드카드는 함께 사용될 수 없다.

    ex) public <T> void methodA(T t) {}

제네릭의 가변성

1) 공변성

T'가 T의 서브 타입이면, C<T'>는 C<T>의 서브 타입이다.

2) 반공변성

T'가 T의 서브 타입이면, C<T>는 C<T'>의 서브 타입이다.

3) 무변성

T'가 T의 서브 타입일 때, C<T'>는 C<T>와 아무 관련이 없다.

- 제네릭은 기본적으로 무변성이다.

  ex) T'가 T의 서브 타입일 때, ArrayList<T'>는 ArrayList<T>와 아무 관련이 없다.
- 이 때, extends나 super를 사용하여 타입 경계를 정하여 공변성, 반공변성을 띄도록 할 수 있다.

📌가상 메모리

다중 프로그래밍 환경에서의 프로세스 간 메모리 분할 방식

- **가변 분할 방식**
  - **연속** 메모리 할당
  - **외부** 단편화
    - 최초 배치, 최적 배치, 최악 배치
    - **조각 모음**
  - **버디 시스템**
    - 가변 분할 but, 공간을 1/2로 나누는 방식으로 내부 단편화로 이끌어냄.
    - 조각 모음 없이 간단하게 큰 덩어리 만들 수 있음
- **고정 분할 방식**
  - **비연속** 메모리 할당
  - 관리 수월, 메모리 통합 같은 부가적인 작업 x, **내부** 단편화

동적 주소 변환(DAT, Dynamic Address Translation)

- 물리 메모리 + 스왑 영역의 가상 주소 → 실제 메모리의 물리 주소로 변환하는 것
- 32bit → 4gb 주소 표현 한계, 64bit는 사실상 한계 없음

세그멘테이션 기법

- 세그멘테이션 매핑 테이블 - limit(세그먼트 크기, 메모리 보호하는 역할), address(물리 메모리 상의 시작 주소)
  - swap → address는 Invalid
- VA=<S,D>
  - limit 초과 시 trap(자신의 영역을 벗어나는 주소에 접근하거나 숫자를 0으로 나누는 것과 같이 사용자가 **의도치 않게** 일으키는 인터럽트를 말한다)

페이징 기법

- 페이징 매핑 테이블 - 물리주소는 프레임, 가상 주소는 페이지
  - 각각의 한 줄(페이지 인덱스, 프레임 번호)을 페이지 테이블 엔트리(PTE)라고 함.
- VA=<P,D> ⇒ PA=<F,D>
  - P=가상 주소/한 페이지의 크기 의 몫, D는 나머지
  - 주소 변환 예시 - 16bit 컴퓨터, 페이지 크기 2^10
    - 가질 수 있는 주소의 최대 수는 2^16 , 총 번지 수는 0-65535
    - 페이지가 2^10이므로 페이지의 주소를 나타낼 수 있는 크기는 2^6. 이후 페이지를 찾아, 페이지 내의 주소를 2^10까지 검색할 수 있다.
      - 16bit 있으면, 6bit는 페이지 주소, 10bit는 페이지 내 거리를 나타낸다는 것
    - 프로세스는 각 프로세스의 가상 주소를 요청한다
      - 908번 주소 요청 → 페이지 크기 나누기 → 몫과 나머지 구함 VA<P,D> → 매핑 테이블 가서 PA=<F,D>로 교체
- PTBR(Page Table Base Register) - 페이지 테이블의 시작 주소 - PCB에 저장됨

페이지 매핑 방식

- **직접 매핑**
  - 페이지 테이블 **전체**가 물리 메모리의 운영체제 영역에 존재. 부가 작업 X
  - index - 페이지, 내용 - 프레임
  - PCB → PTBR → DAT
- **연관 매핑**
  - 모든 페이지 테이블을 저장 장치의 스왑 영역에 저장
  - 일부의 무작위 테이블 부분만 물리 메모리로 가져옴
    - 이 일부분의 테이블 부분을 연관 레지스터, TLB(Translation Look-aside Buffer)에 저장
    - **TLB** hit | TLB miss → 미스 알게 되는 시점은 모든 검색이 끝나고 나서임
  - 페이지, 프레임 | 두 열 가짐

    - 메모리 상의 모든 테이블 요소를 찾아야 하고, 없으면 스왑 영역에서 다시 찾아야 해서 시간 낭비

- **집합-연관 매핑**
    - VA=<P1,P2,D>, 멀티페이지 매핑
    - TLB에 디렉터리 매핑 정보 저장
    - 페이지 테이블을 일정한 집합으로 자르고, 자른 테이블들을 관리하는 집합 테이블을 물리 메모리로 가져옴.
    - 집합 테이블은 자른 페이지 테이블이 물리 메모리에 있는지 스왑 영역에 있는지 위치 정보를 표시
        - 원하는 페이지 엔트리가 물리 영역에 있는지 스왑에 있는지 파악 가능

          → 물리 메모리의 모든 페이지 테이블을 검사할 필요가 없어 주소 변환 시간 단축됨, 물리 메모리 낭비X
- **역매핑**
    - 물리 메모리의 프레임 번호를 기준으로 테이블을 구성함.
    - 프로세스의 수와는 상관 없이 항상 일정 크기의 페이지 테이블을 유지하여 크기가 작다.
    - 프레임 | PID | 페이지 번호
        - 모두 검사하고 나서야 저장장치에 접근하여 검색 시간 낭비할 수 있음.

세그먼테이션-페이징 혼용 기법

- 메모리에 있는 데이터로의 접근 권한(R, W, E) 소유 권한을 나타내기 위해 프로세스에 세그먼트 테이블 사용
    - 세그먼트 테이블은 각 권한에 따라 나누어져 연결된 테이블의 시작 주소를 가진다.