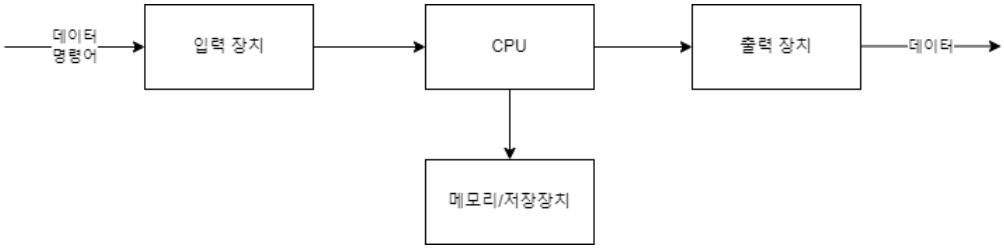
OS - 메모리 관리

메모리란

- 분류 : 컴퓨터 구조 중 CPU와 함께 필수 장치 중 하나



- 동작 : 폰노이만 구조에 의해, 모든 프로그램은 실행을 위해 메모리에 올라와야 한다.
- 주소 : 1B 단위로 나뉘어져 있다.
- 대상 : 데이터와 명령어를 저장한다.
- 관리자 : 메모리 관리 시스템이 관리를 담당한다.
- 명령어 : 데이터 적재, 저장을 지원한다.
- 종류 : 논리 주소(CPU가 지정), 물리 주소(실제 주소) 종류가 있다.
- 속도 : CPU 보다 속도가 느리다.

> 폰노이만 구조란?
>
> - 현대 컴퓨터 구조로, 소프트웨어 개념이 없었던 때 프로그래밍을 하드웨어로 하던 것 (스위치 조작, 배선 열선 변경 등)을 벗어나기 위해 나온 기술
> - 프로그램과 사용할 데이터를 메모리에 저장할 수 있는 구조

OS의 메모리 관리

프로그램 실행 시, OS는 메모리 관리를 다음 3가지로 진행한다.

1. 메모리로 가져오기

메모리로 가져오는 대상은 다음과 같다.

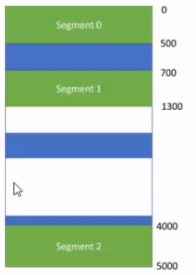
1. 실행할 프로그램(**명령어** 모음)을 저장장치에 접근하여 가져온다.
2. 연산에 사용될 **데이터**를 가져온다.

2. 메모리 배치하기

실행할 명령어와 데이터를 메모리의 어디에 배치할 지 결정한다. 방법은 다음 2가지가 있다.

1. 세그멘테이션 (가변 분할 방식)

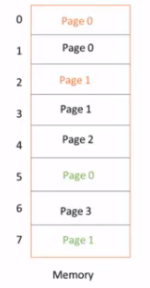
- 정의 : 프로세스 크기에 따라 메모리를 나누어서 데이터를 배치하는 기법.
- 방법 :



    - 문제 : 외부 단편화(빈 공간)의 발생
    - 해결 : 조각 모음(메모리 통합) 실행한다. 즉, 이미 배치된 프로세스를 옆으로 옮겨서 빈 공간을 합치는 작업이다.
      하지만, 프로세스의 메모리 위치를 바꿀 때, 프로세스의 동작을 멈춰서 오버헤드를 발생시키며, 메모리 관리가 복잡하다는 단점이 있다.
- 장점 : 하나의 프로세스를 연속된 공간에 배치할 수 있다.
- 단점 : 메모리 관리가 복잡하다.

2. 페이징 (고정 분할 방식)

- 정의 : 프로세스의 크기와 상관없이 메모리를 같은 크기로 나누어서 데이터를 배치하는 기법.
- 장점 : 메모리 관리가 편하다.
- 방법 :



    - 문제 : 내부 단편화의 발생
    - 해결 : 딱히 없다. 따라서, 페이지의 크기를 잘 정해야 한다.
- (페이지 테이블로 동적 주소 변환)

3. 메모리를 재배치 하기

- 기능 : 쓰지 않을 프로세스를 메모리에서 하드디스크로 내보내서 여유 공간을 만든다.
- 발생 조건 : 페이지 부재 (실행되는 프로세스가 페이지를 요청했는데, 메모리에 없는 상황)
- 문제 : 자주 발생하면 하드디스크의 잦은 입출력으로 스레싱(작업이 멈춘 것 같은 상태)가 발생한다.
- 해결
    1. 앞으로 쓰지 않을 페이지를 잘 선정하여 하드디스크에 보낸다. [페이지 교체 알고리즘]
    2. 동시에 실행하는 프로그램 수를 줄인다.

페이지 교체 알고리즘이란

- 정의 : 하드디스크로 보낼 페이지를 결정하는 알고리즘
- 목적 : 메모리에서 앞으로 사용할 가능성이 적은 페이지를 선정한다.

- 종류
    1. 무작위 알고리즘 : 무작위로 페이지를 선정한다.
    2. FIFO 알고리즘 : 처음 메모리에 올라온 페이지를 선정한다.
    3. 최적 알고리즘 : 미래 메모리 접근 패턴을 보고 선정한다. (구현 불가능)
    4. LRU 알고리즘 : 가장 오랫동안 사용되지 않았던 페이지를 선정한다.
        - 단점 : 시간을 기록하는 메모리 공간 필요
    5. LFU 알고리즘 : 가장 적게 사용되었던 페이지를 선정한다.
        - 단점 : 호출 횟수를 기록하는 메모리 공간 필요
    6. NUR 알고리즘 : LRU + LFU. 단, 시간과 빈도의 유무만 비트로 표현하여 메모리 공간을 확보한다.