**오퍼레이팅시스템 – 3차 과제**

**12131819 육동현**

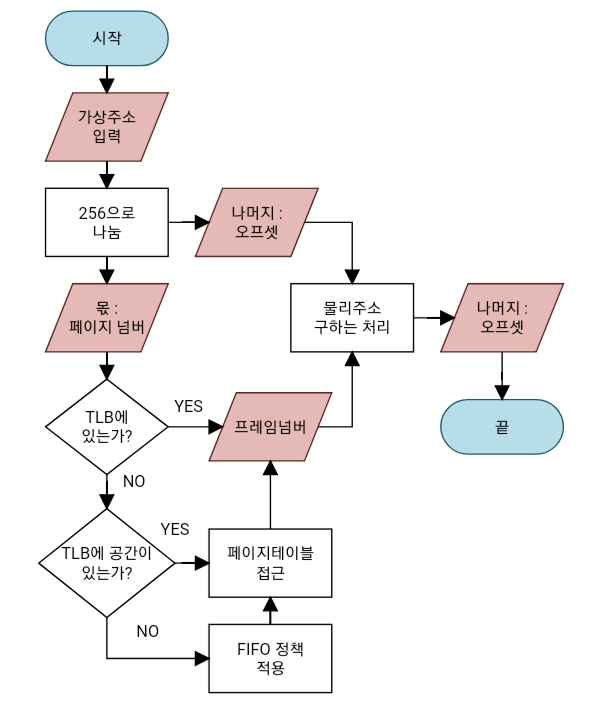
**01\_프로그램 개요**

**- addresses.txt를 읽어서 physical.txt를 생성한다.**

**- TLB를 관리한다.**

**- 프레임 테이블을 관리한다.**

**02\_설계 아이디어**

****

**저는 이번 프로그램을 위와 같은 흐름도를 작성한 후 이에 기반하여 작성하였습니다.**

**프로그램이 시작되면 addresses.txt에 저장된 논리주소를 읽어 들이고 해당 가상주소를 페이지 넘버와**

**오프셋으로 쪼갭니다. 이렇게 얻어 온 페이지 넘버를 TLB 내부 테이블에 존재하는지 탐색해봅니다.**

**만약 TLB 내부 테이블에 해당 페이지 넘버가 존재할 경우 이에 대응되는 프레임 넘버를 설정하고 이에**

**기반하여 물리주소를 구합니다. 만약 TLB 내부 테이블에 존재하지 않는 경우, TLB 내부에 공간이 있는**

**경우와 TLB 내부에 공간이 없는 경우로 나누어 처리합니다. TLB 내부에 공간이 있는 경우 즉시 페이지**

**테이블에 접근하여 프레임 넘버를 설정하고 이를 TLB 내부 테이블에 저장합니다. 만약 TLB 내부에 공간이**

**없는 경우 FIFO 정책에 의거하여 가장 먼저 들어 온 것을 victim으로 선택하고 쫓아낸 후 페이지 테이블에**

**접근하여 프레임 넘버를 설정하고 이를 TLB 내부 테이블에 저장합니다. 이렇게 하고 난 후 앞의 경우와**

**마찬가지로 설정된 프레임 넘버에 기반하여 물리주소를 얻어오는 처리를 합니다.**

**(프로그램을 설계하면서 스마트폰 메모장에 적은 내용을 보고서의 06번 부분에 참고자료로 제시합니다.)**

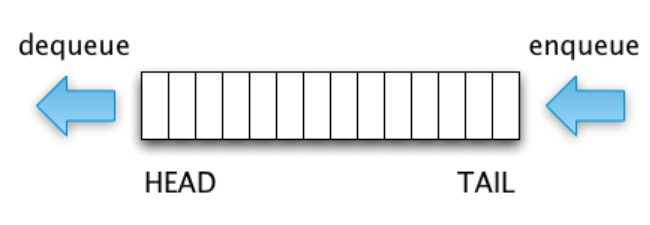
**03\_자료구조 설계 아이디어**

**저는 자료구조를 설계 할 때 교수님의 강의교재에 있는 자료를 참고하여 만들었습니다.**

**페이지 테이블은 배열의 인덱스가 페이지 넘버이고 배열의 저장 값은 프레임 넘버인 1차원 배열로**

**설계하였습니다. 또한 프레임 테이블은 배열의 인덱스가 프레임 넘버이고 배열의 첫 번째 열은 사용여부를 표현하는 부분, 배열의 두 번째 열은 페이지 넘버인 2차원 배열로 설계하였습니다. TLB 내부 테이블은**

**첫 번째 열은 페이지 넘버, 두 번째 열은 프레임 넘버인 2차원 배열로 설계하였습니다.**

****

**또한 위의 그림과 같이 TLB의 저장용량을 초과한 경우 victim을 선택하는 데 FIFO 정책을 적용하기**

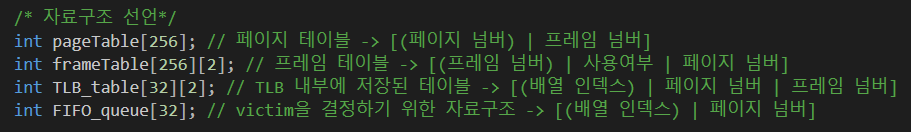
**때문에 이를 구현할 수 있게 지원하는 FIFO 큐를 설계하였습니다. FIFO 큐의 동작원리는 가장 먼저 들어온**

**원소가 가장 먼저 나가는, 즉 victim으로 설정되는 구조를 만들었습니다. 따라서 공간이 꽉 찬 경우 원소가 들어오게 된다면 가장 앞(HEAD)에 있는 원소가 나가고 그 바로 뒤에 있는 원소들을 앞으로 한 칸씩 당겨 오는 처리를 한 후 가장 뒤(TAIL)에 새로 들어 온 원소를 삽입하는 처리를 합니다. (이는 FIFO 큐와 TLB**

**내부 테이블에 모두 적용합니다.)**

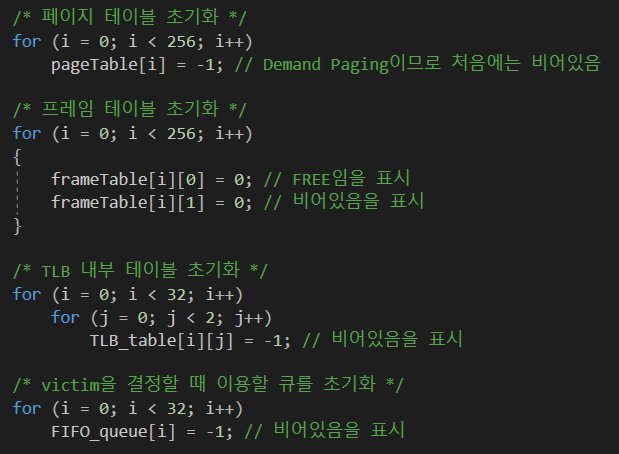
**04\_핵심코드 설명**

**1) 아래는 03에서 설명 드린 자료구조를 기반으로 코드를 구현하였습니다.**

****

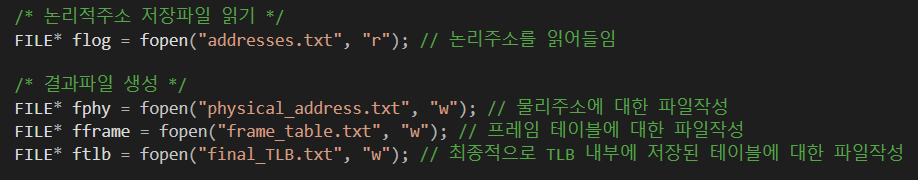
**2) 아래는 Demand Paging을 구현하기 위해 작성한 코드입니다.**

**처음에는 아무 것도 올라와 있지 않으므로 -1(Invalid)로 설정합니다.**

****

**3) 아래의 첫 번째 코드는 논리적 주소가 저장된 파일을 읽어 들이는 처리입니다.**

**그 아래 세 줄은 각각 물리주소, 프레임테이블, TLB에 대한 파일을 작성하기 위한 처리입니다.**

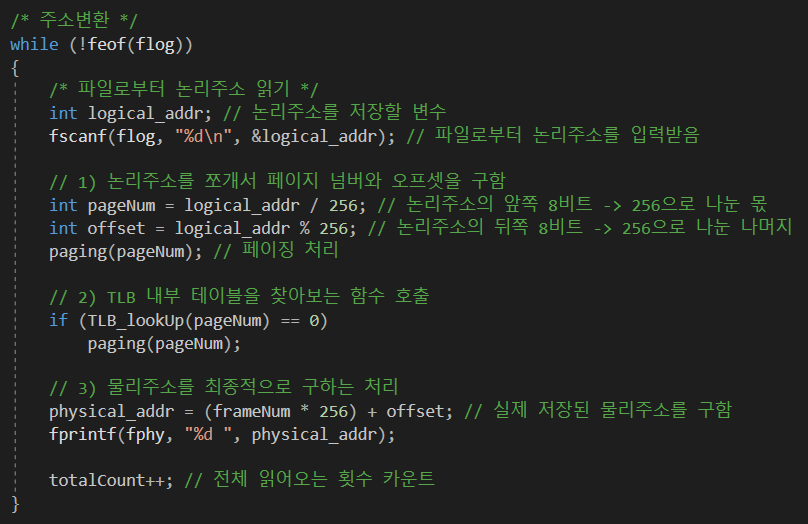
****

**4) 아래는 주석에 작성해 놓은 것과 같이 주소변환을 위한 처리를 하는 코드입니다.**

**이는 02번의 설계 아이디어에서도 말씀 드린 것과 같이 논리주소를 읽어오는 처리,**

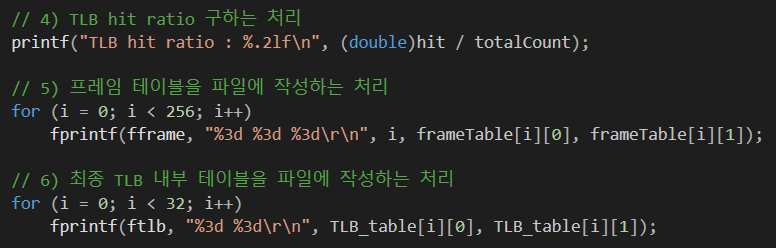
**논리주소를 쪼개서 페이지넘버와 오프셋을 구하는 처리, 그리고 TLB 내부 테이블을 찾아보는 처리**

**그리고 마지막으로 물리주소를 최종적으로 구하는 처리를 각각 코드로 구현하였습니다.**

****

**5) 아래는 각각 TLB hit ratio를 구하는 처리, 그리고 프레임테이블과 TLB 내부 테이블을 파일에 작성하는**

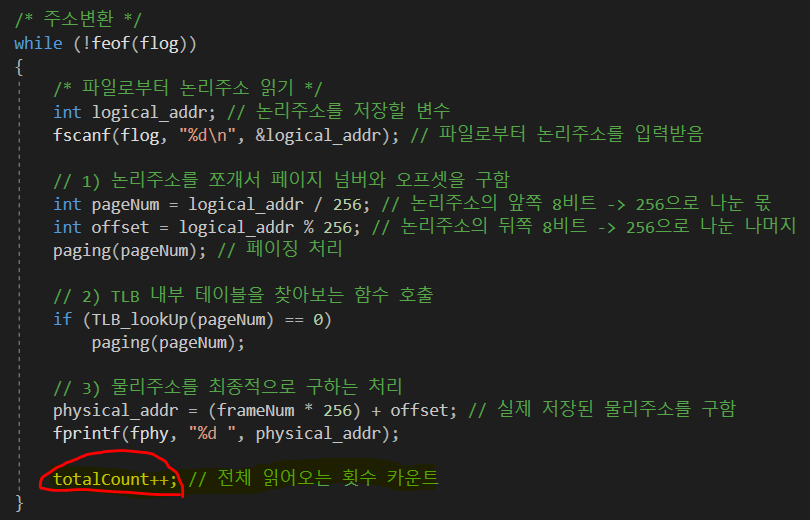
**처리를 구현한 코드입니다.**

****

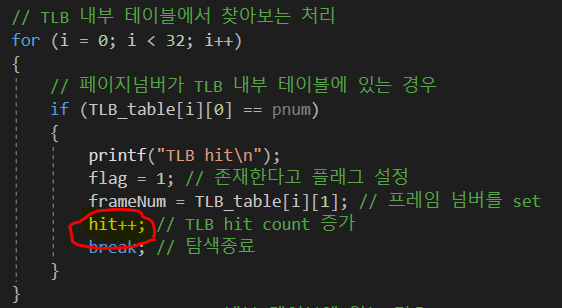
**TLB hit ratio의 경우 아래의 처리에 의해 산출됩니다.**

****

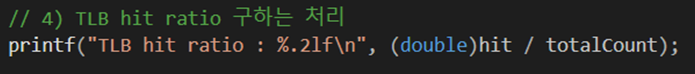
**논리주소를 읽어올 때마다 카운트를 하나씩 증가시킵니다.**

****

**TLB 내부 테이블에 해당 페이지넘버가 존재하는 경우 hit 카운트를 증가시킵니다.**

****

**위와 같은 방법을 통해 얻은 hit와 totalCount를 통해 TLB hit ratio를 산정하여 화면에 출력합니다.**

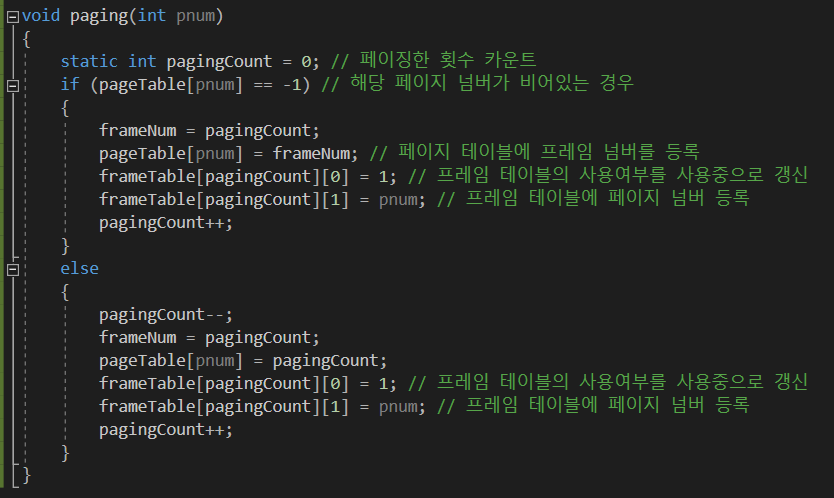
****

**6) 아래는 페이지테이블에 해당하는 페이지에 대응되는 프레임넘버가 없는 경우 이를 페이지테이블에**

**올리는 처리를 구현한 코드입니다. 프레임넘버는 0부터 시작해서 하나씩 증가시켜가면서 대응하고**

**이에 따라 페이지테이블과 프레임테이블을 업데이트하는 처리를 수행합니다. 만약 페이지테이블에 해당**

**페이지넘버가 존재한다면 카운트를 하나 줄인 후 위와 같은 처리를 하도록 설계하였습니다.**

****

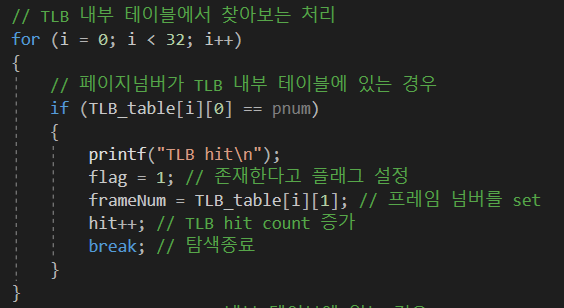
**7) TLB 내부 테이블을 찾아보는 함수는 아래와 같은 처리로 구성되어 있습니다. (아이디어는 02번 참고)**

**TLB 내부 테이블에 해당 페이지넘버가 존재하는지 여부를 판단하기 위해 플래그를 만듭니다.**

****

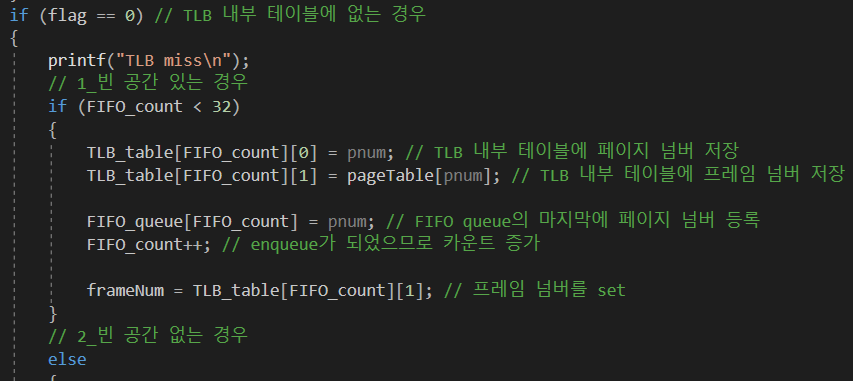
**우선 아래와 같이 TLB 내부 테이블에 해당 페이지넘버가 존재하는지 찾아봅니다.**

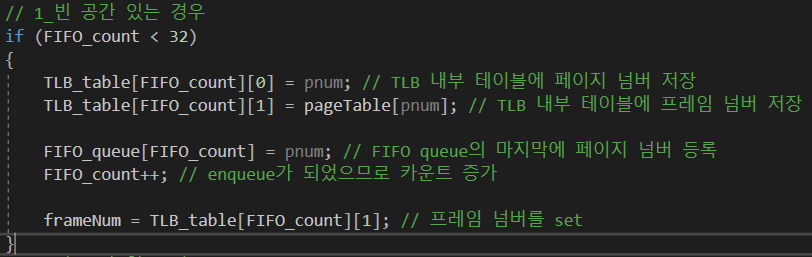
**만약에 페이지넘버가 존재하는 경우 프레임넘버를 설정하고 TLB hit를 증가시킵니다.**

****

**위의 처리를 수행하고도 플래그가 0인 경우, 즉, TLB 내부 테이블에 페이지넘버가 없는 경우**

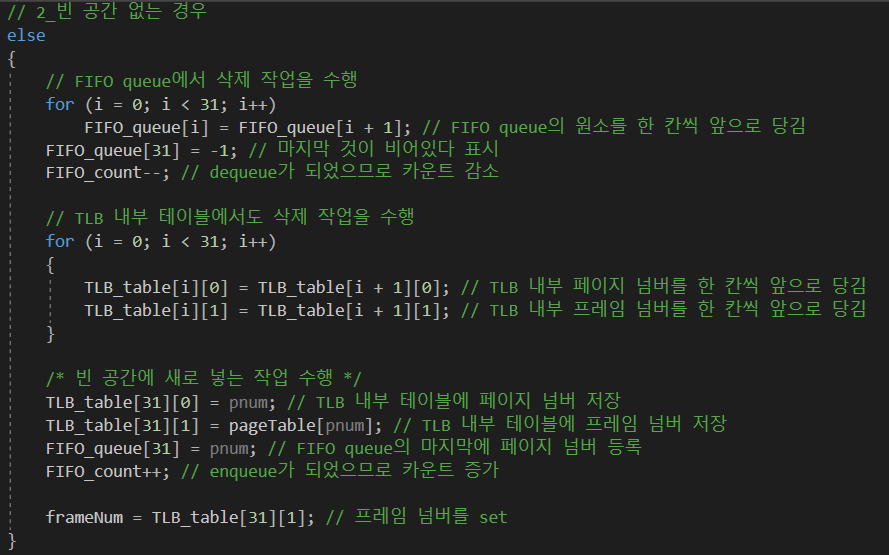
**아래와 같이 페이지테이블에 접근하기에 앞서 TLB에 저장 가능한 공간이 있는지 없는지 따져봅니다.**

****

****

**위와 같이 빈 공간이 있는 경우 페이지테이블을 읽어 들여 TLB 내부 테이블의 가장 뒷부분(TAIL)에**

**(페이지넘버, 프레임넘버)의 투플을 추가해줍니다.**

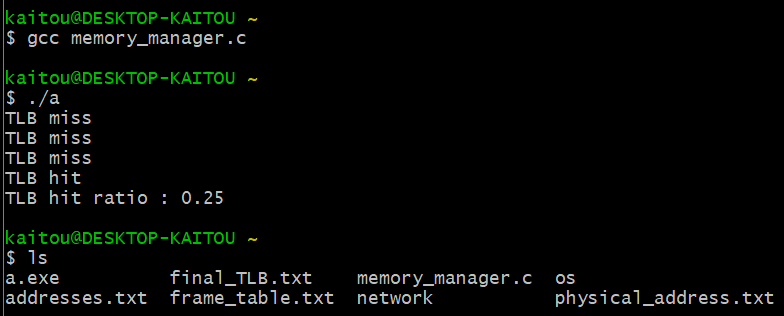
****

**만약 빈 공간이 없는 경우 03번에서 설명 드린 것과 마찬가지로 FIFO 정책을 통해 우선적으로 victim을**

**선택하여 쫓아내는 처리를 한 후 (FIFO 큐와 TLB 내부 테이블 모두에서 수행), 페이지테이블을 읽어 들여**

**위와 마찬가지로 TLB 내부 테이블의 가장 뒷부분(TAIL)에 (페이지넘버, 프레임넘버)의 투플을 추가해줍니다.**

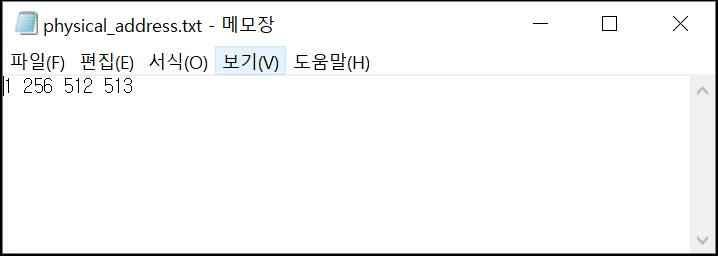
**05\_실행결과 설명**

****

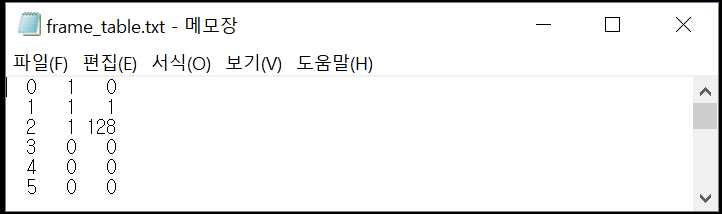
**프로그램을 컴파일하고 실행하면 위와 같이 TLB hit ratio가 나옵니다.**

**또한 미리 주어진 addresses.txt를 input으로 하여 final\_TLB.txt, frame\_table.txt,**

**그리고 physical\_addresses.txt를 생성합니다.**

****

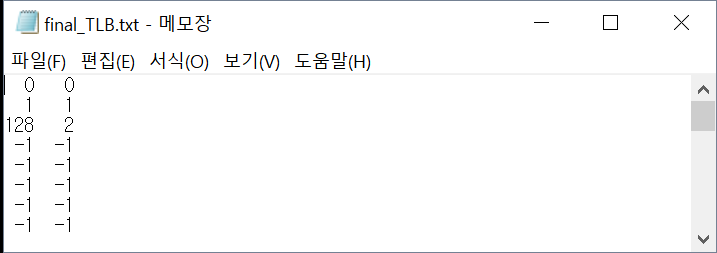
**위는 addresses.txt를 기반으로 산출된 물리주소를 파일로 작성한 것입니다.**

****

**위는 프레임테이블을 파일로 작성한 것입니다. 이는 일부만 캡처한 것으로 전체를 보시려면 .txt 파일을**

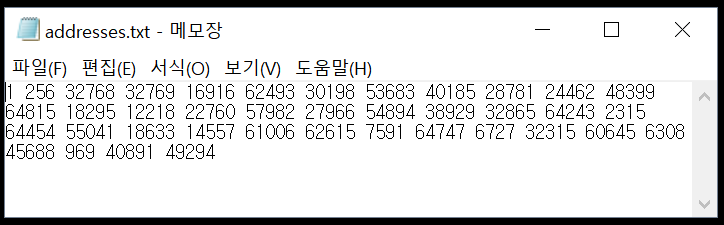
**읽어보셔야 합니다. 첫 번째 열은 프레임넘버, 두 번째 열은 사용여부 (USED의 경우 1, FREE의 경우 0),**

**세 번째 열은 페이지넘버입니다.**

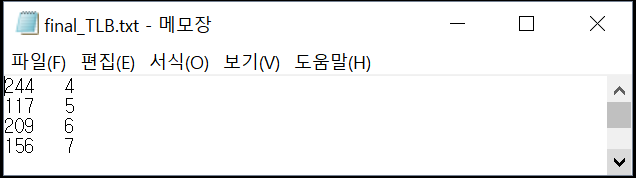
****

**위는 모든 처리를 마친 후 TLB 내부 테이블을 파일로 작성한 것입니다.**

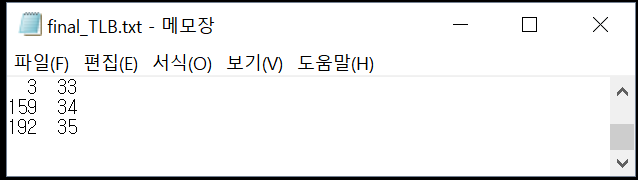
**-1의 경우 비어있다는 것을 나타냅니다. 첫 번째 열은 페이지넘버, 두 번째 열은 프레임넘버입니다.**

****

**위는 TLB에 있어 FIFO 정책이 제대로 동작하는지 확인하기 위해 임의의 논리주소를 넣은 파일입니다.**

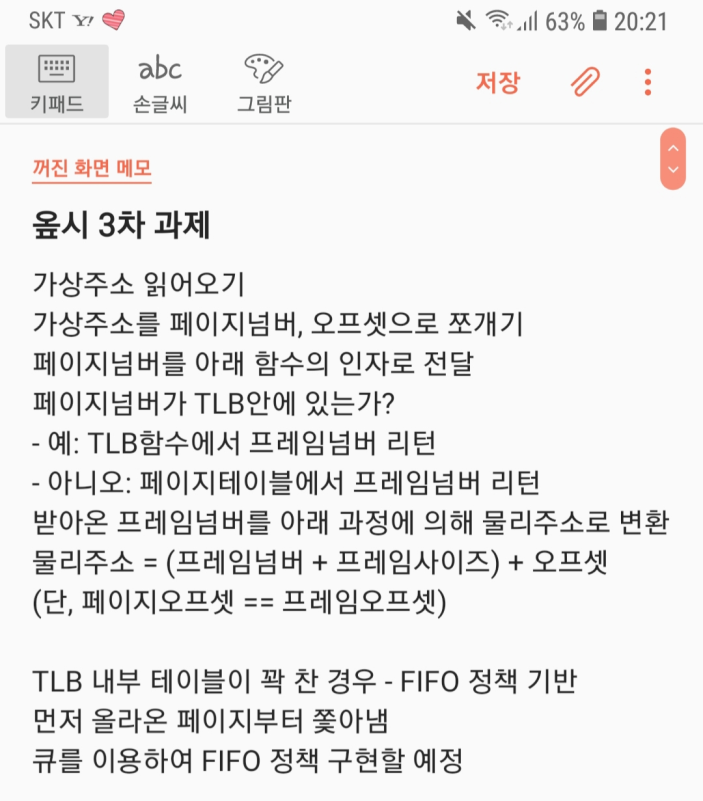
****

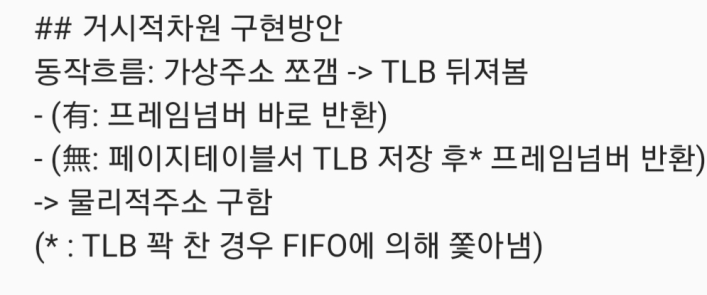
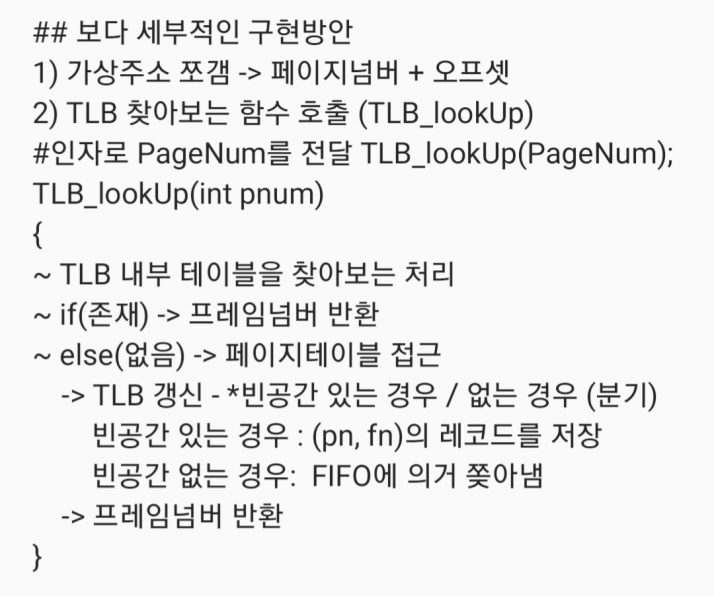
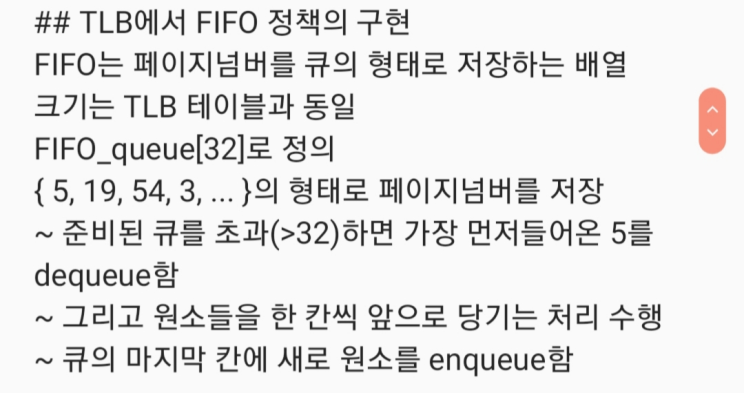
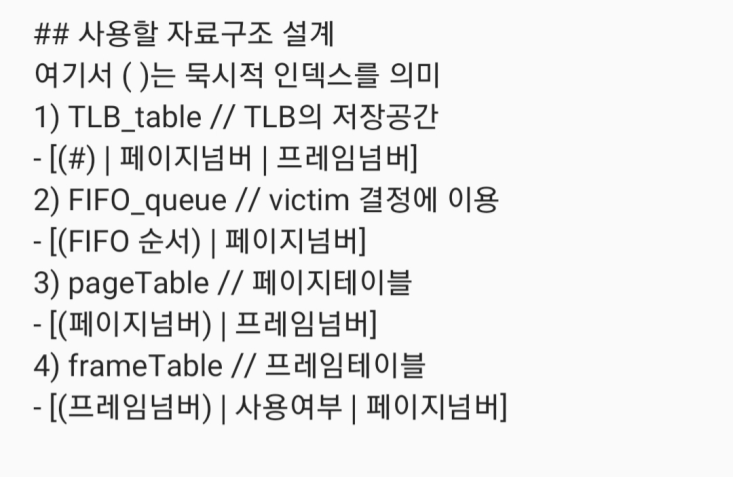
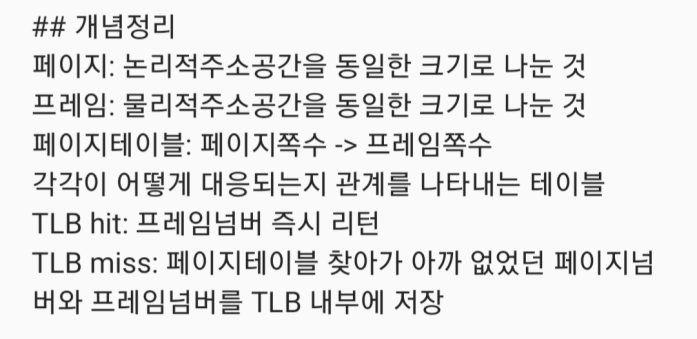
**(중간에도 있으나 공간부족으로 인해 필요한 부분만 캡처)**

****

**이를 실행하면 TLB에는 프레임넘버 0, 1, 2, 3에 해당하는 투플이 FIFO 정책에 의해 제거됨을 알 수**

**있습니다. (위의 캡처는 설명을 위해 일부만 캡처한 것입니다.)**

**06\_참고자료 (스마트폰 메모장에서 설계한 자료 캡처)**

** **