**오퍼레이팅 시스템 3차 과제**

**Virtual Memory Manager**

컴퓨터공학과 12131593

전 준호

* **개요**

이번 과제를 핵심적인 내용으로 나눈 다면 총 다 섯 가지로 나눌 수 있다. 첫 번째로 logical address를 받아와

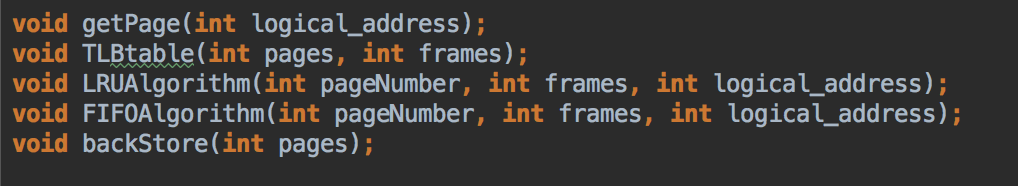
page, frame 테이블을 만드는 것, 두 번째로 page, frame테이블을 바탕으로 page, offset을 구하고 그

것들을 이용해 physical address를 Translation 하는 것, 세 번째로 TLB테이블 생성 및 관리, 네 번째로

replacement 알고리즘을 이용한 frame테이블 관리, 마지막으로 backing store 이용이다.

* **구조**

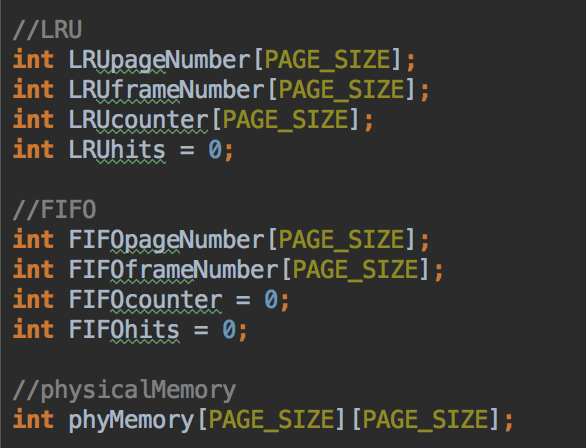
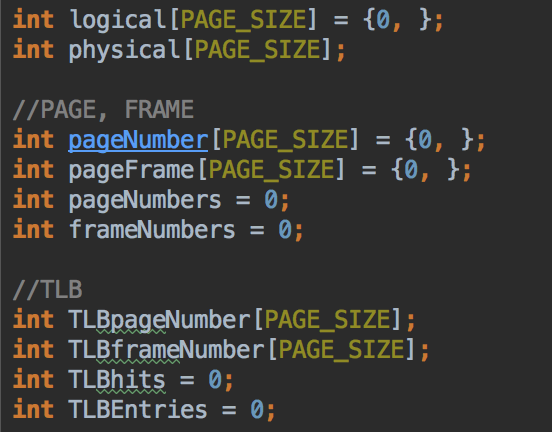
1. 함수

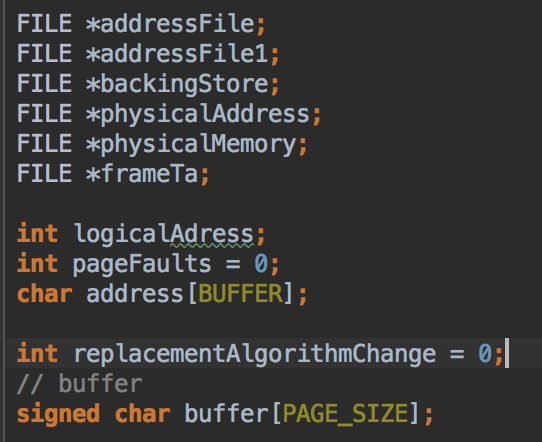


Virtual Memory Manager를 위한 함수로 총 다 섯 가지를 이용했으며 getPage는 logical\_address를 받아와 page,frame 테이블 생성 및 관리, TLB테이블 확인, 프레임 개수가 128개 초과가 되었을 때 일어나는 replacement 알고리즘, backingStore Swapping in, out을 기능으로 가지고 있으며 메모리 관리에 중심을 담당하고 있다.

TLB함수는 TLB테이블 생성 및 관리를 LRUAlgorithm은 replacement알고리즘 중 하나인 LRU, FIFOAlogirothm은 FIFO를 관리한다. 마지막으로 backStore함수로 backing store를 관리한다.

1. 변수

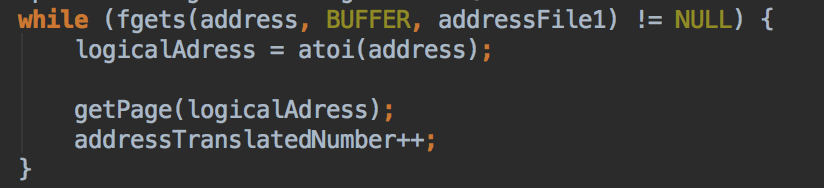




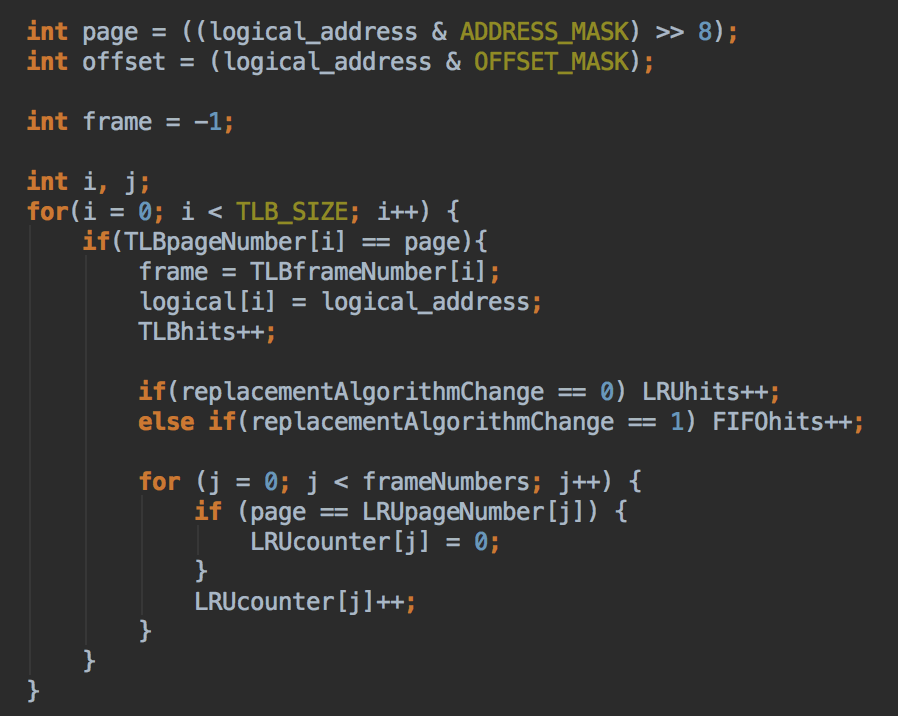
address -> fget함수를 이용해 logical address를 받아오기 위해 사용.

replacementAlgorithmCahnge -> 0 일 때 LRU 1 일 때 FIFO

* **핵심 코드 소개(진행 순서대로)**

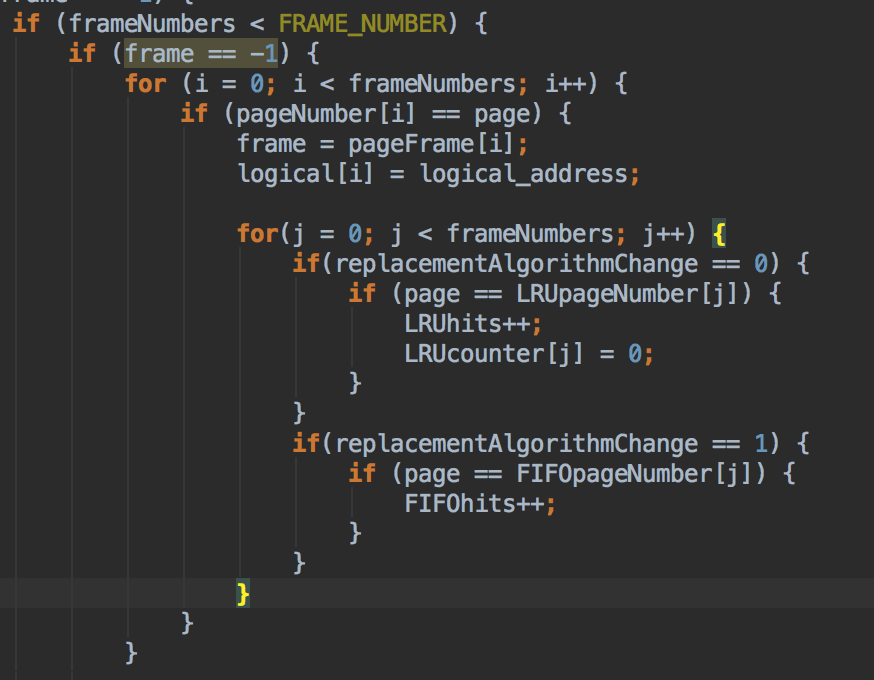


addressFile로 addresses.txt를 오픈하여 fgets함수로 한 줄씩 받아와 logicalAdress변수에 저장 하여 getPage로 이동하며 Translation한 페이지 수를 증가한다.

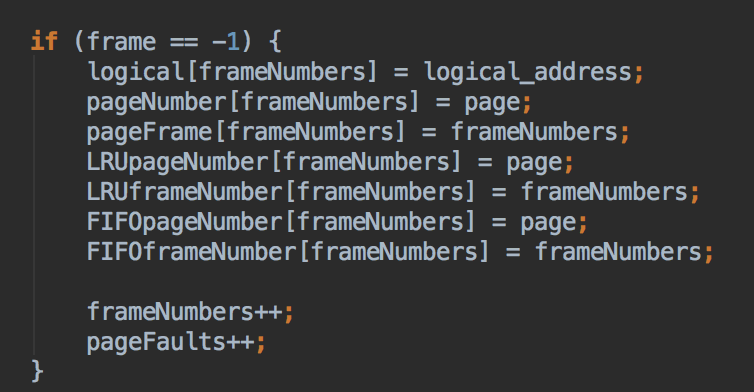


getPage함수로 page와 offset을 구하며 frame을 초기에 -1로 두며 Memory Protection을 한다.

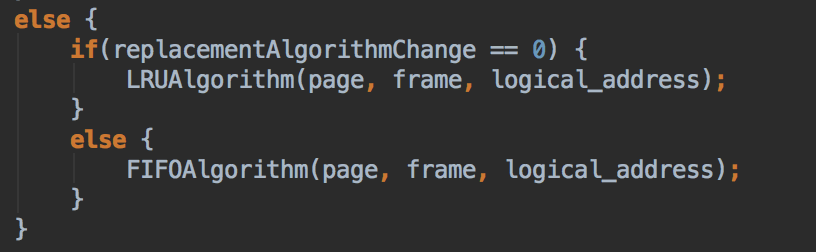
그 후 구한 page No가 TLB테이블에 있다면 frame No를 가져온다.



TLB테이블에 page No가 없다면 계속해서 frame변수가 -1 이므로 page테이블에 접근해 해당하는 frame No를 찾는다.

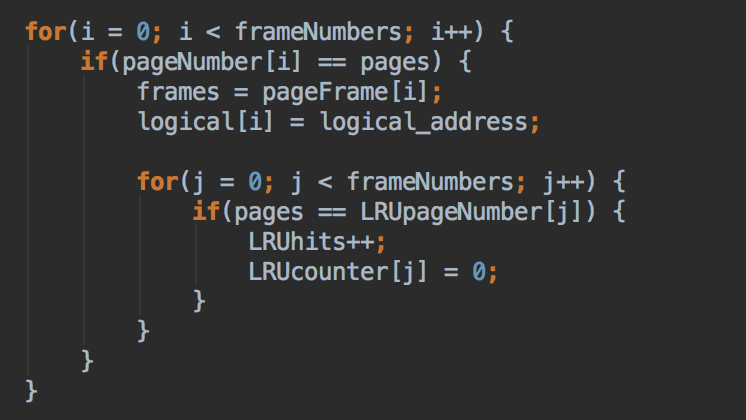


page테이블에서도 찾을 수 없고 현재 존재하는 frame의 개수가 128개가 넘지 않는다면 페이지 테이블에 구한 page와 frame No를 순서대로 증가시키며 페이지 테이블에 넣어주며 frame 개수를 증가.

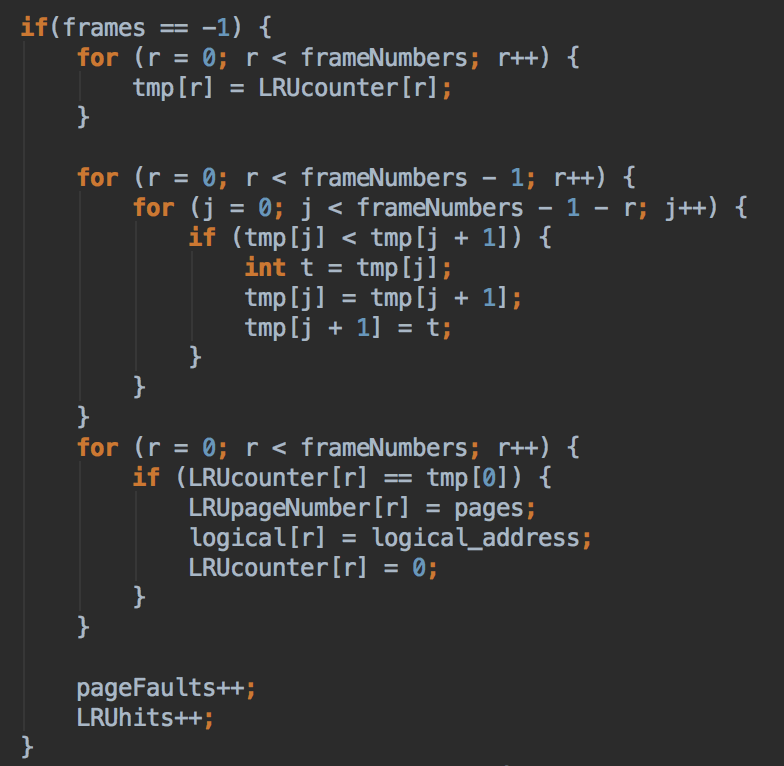


만약 frame의 개수가 128개가 초과 되었고 페이지 테이블에서 frame No를 못 찾았다면 replacement 알고리즘이 필요하며 사용한다.

* LRU 알고리즘(COUNTER 방식 사용)

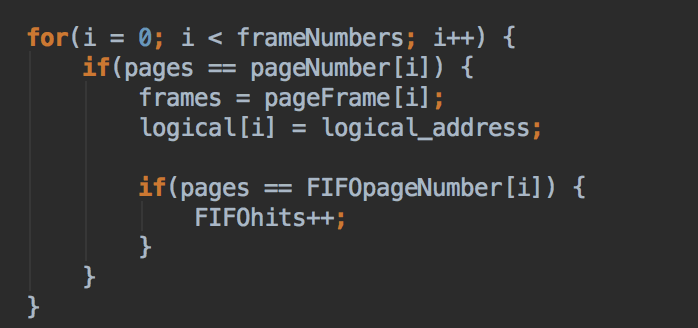


replacement알고리즘을 사용하기 전에 page테이블을 검색해 page가 있다면 frame No를 가져오고 종료.



page테이블에서 frame No 못 가져왔다면 replacement 알고리즘(LRU)실행하여 가장 처음 접근 된 page를 교체

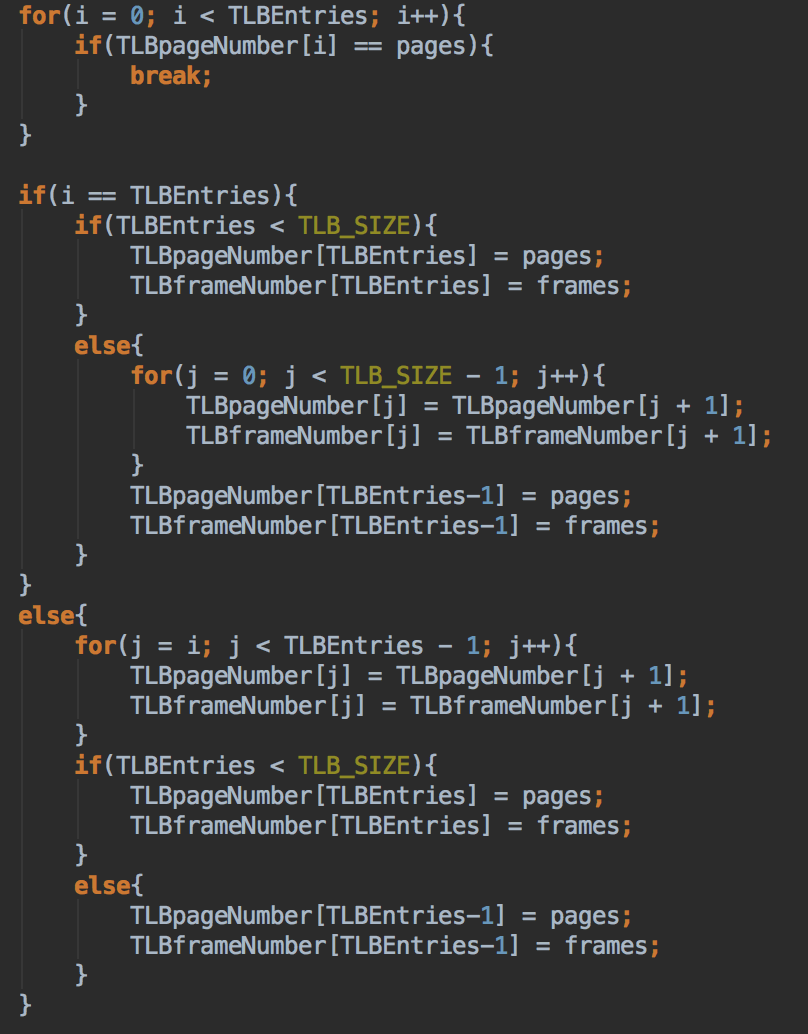
* FIFO 알고리즘



FIFO 알고리즘 실행 전에 page테이블을 탐색하여 같은 page가 있는지 확인 후 없으면 replacement 알고리즘(FIFO) 실행하여 가장 먼저 들어 온 page 교체



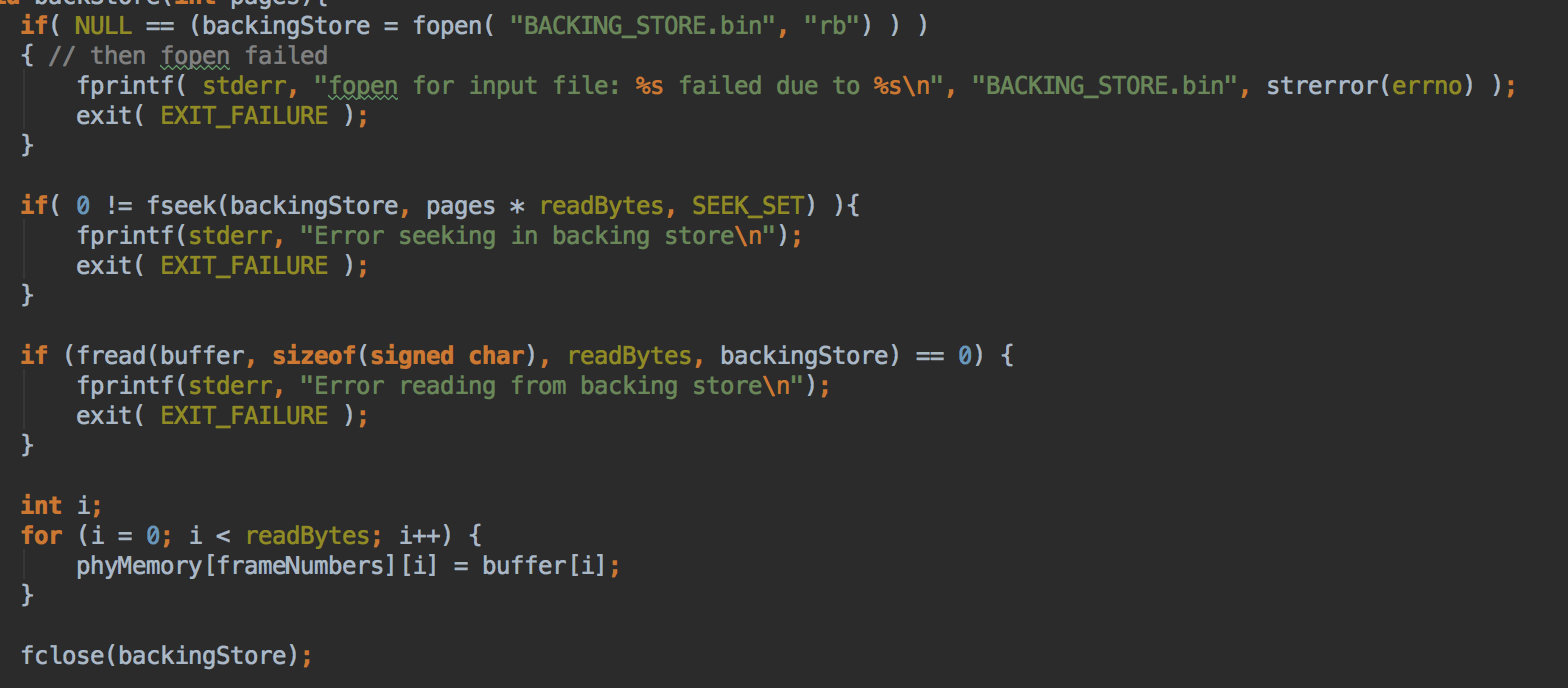
모든 과정을 거친 후 logcial address를 이용해 구해 낸 page와 그 page의 frame No를 가지고 TLB테이블에 입력



page를 받아 와 그 page가 이미 TLB테이블에 있다면 그 page를 테이블 가장 뒤로 옮겨 가장 최근에 사용 했다는 것을 확인. page를 찾을 수 없다면 TLB테이블에서 가장 처음에 접근 된 page를 교체. (STACK방식 사용)



TLB테이블을 채우고 backing store를 읽음.



fseek함수를 이용해서 pages \* readBytes(256)만큼 떨어진 곳의 데이터에 포인터를 옮기고

fread함수를 이용해서 readBytes(256)만큼 signed char(1)크기를 읽어와 buffer에 저장한 후

phyMemory배열에 저장한다.



마지막 과정으로 translation된 physical memory를 txt파일에 저장한다.

* **개발 환경**

온라인 통합 개발환경인 Cloud 9에서 c를 이용해 개발

* **평가 및 개선방향**

address translation은 과정이 어렵고 이해하기가 힘들었다. 하지만 이번 과제를 통해서 이해하는데 도움이 되었다. 하지만 backing store에 대해서는 인터넷을 찾아 해결하기는 했지만 완벽하게는 이해가 안되는점이 아쉽다.

또 LRU알고리즘을 할 때 TLB에서는 stack방식을 이용했고 page 테이블에서는 counter방식을 이용했는데 LRU 유사 알고리즘인 reference bit를 이용해 성능을 향상시킬 수 있지만 그렇게 하지 못해 개선해야 할 점이라고 생각한다.