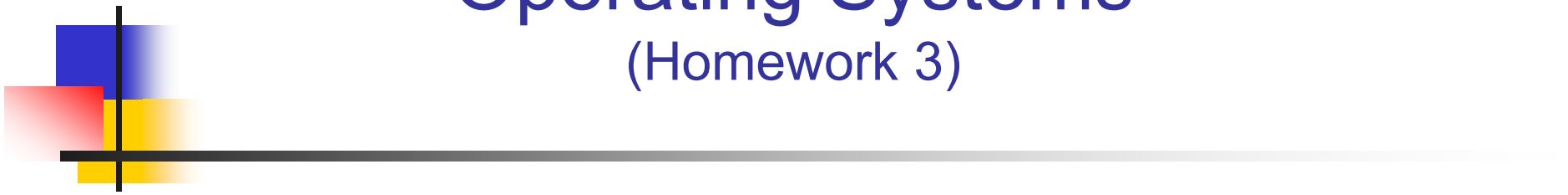
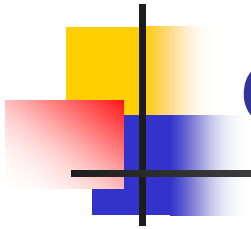


Operating Systems
(Homework 3)

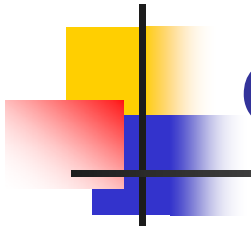
These lecture materials are modified from the source lecture notes written by A. Silberschatz, P. Galvin and G. Gagne.

Spring, 2017



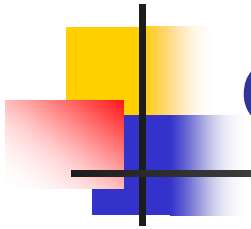
Outline

- Objectives
- Overview
- How to write a program ? (hints)



Objectives

- 목표
  - Designing a Virtual Memory Manager
- 숙제
  - Demand paging, address translation, page fault, page replacement
  - textbook 447 - 450
- 주의 사항
  - Copy 등 어떤 형태의 cheating 은 허용이 안되며 만약 적발 시에는, (예년에 조교들의 적발률은 매우 높았음)
    - copy 를 제공한 학생과 copy 한 학생 모든 숙제가 0 점이 됨

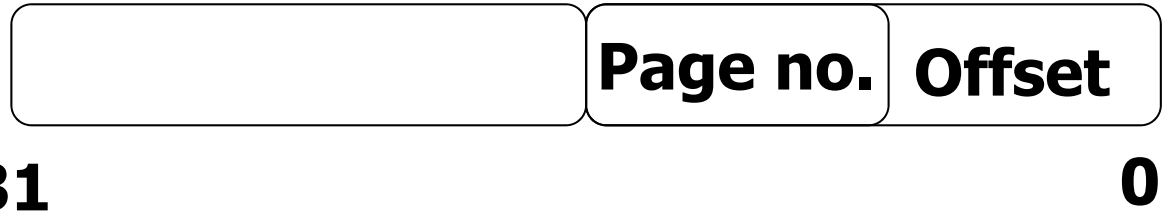


Overview

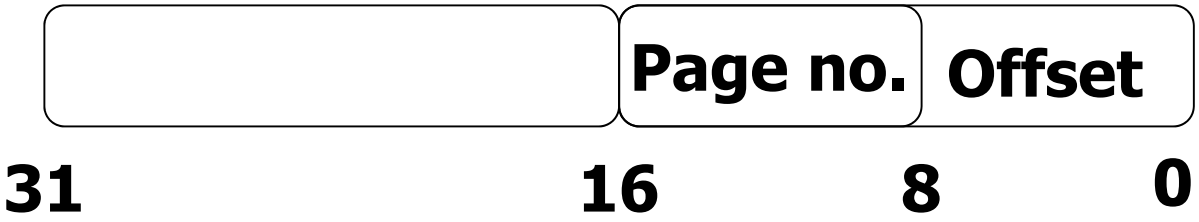
- Write a program that translates logical to physical addresses for a virtual address space of size 2^{16} = 65,536 bytes
  - 1. Read a file containing logical addresses
  - 2. Translate each logical address to its corresponding physical address
    - Using 1) page table and 3) frame table
  - Objective
    - To simulate the steps involved in translating from logical address to physical address

Specifics

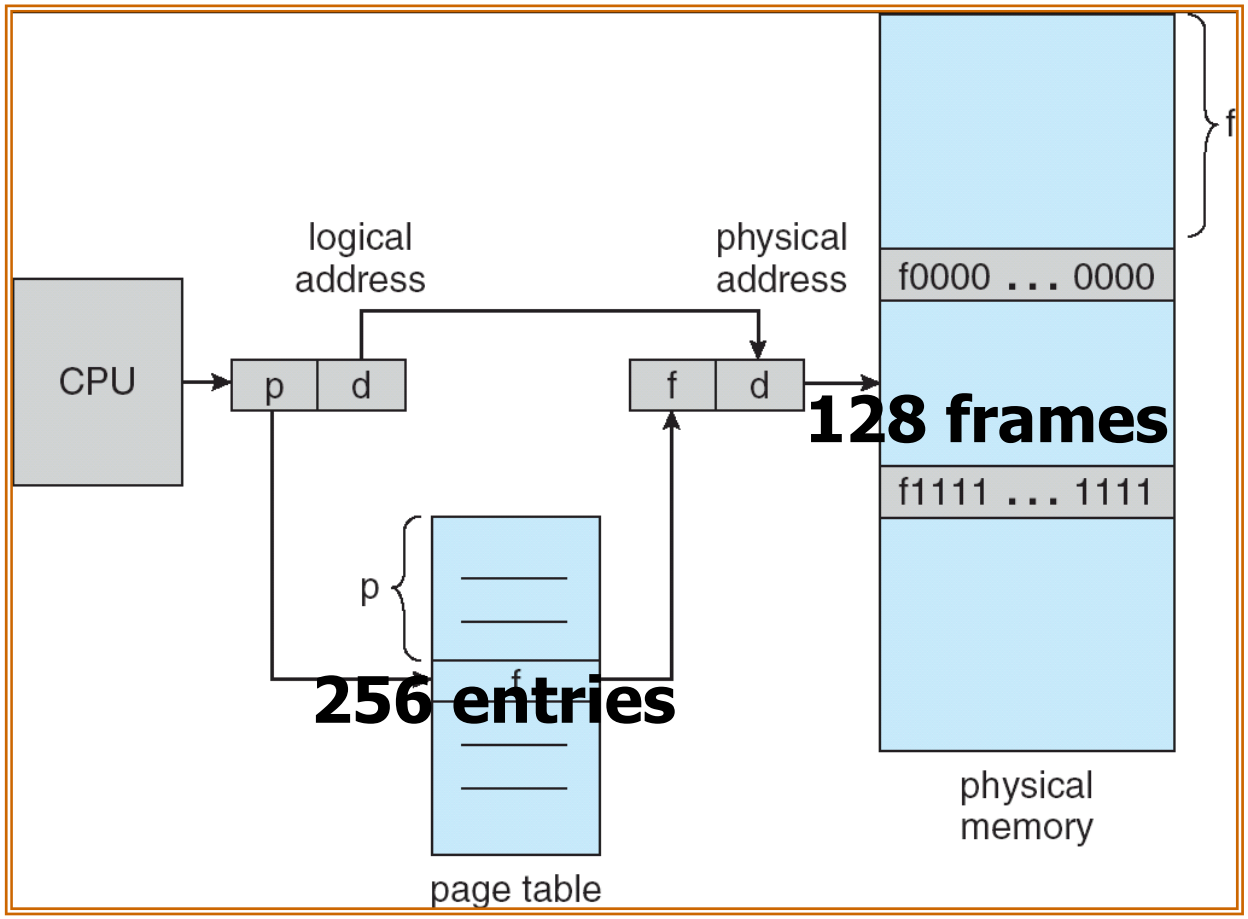
- 1. Read a file containing several 32-bit integer numbers that represent logical addresses
  - You need to be concerned about only 16-bit addresses, so you must mask the rightmost 16 bits of each logical address
  - 16bits are divided into (1) an 8-bit page number and (2) 8-bit page offset

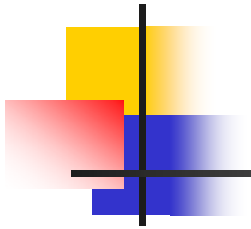


- 2. Other specifics
  - 2^8 entries in page table
  - Page size: 2^8 bytes
  - 16 entries in TLB
  - Frame size: 2^8 bytes
  - 128 frames
    - 32768 bytes of frame size (128 X 256 bytes)

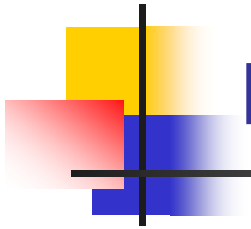


- Address translation





- Understanding
  - 1. Demand paging
  - 2. Paging
  - 3. LRU



Running your program

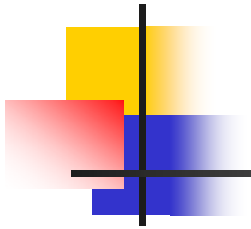
- 개요
  - 1. addresses.txt 읽어서 physical.txt 를 생성하기
  - 2. page fault rate 출력하기
  - 3. frame table 관리하기
  - 4. Backing store 관리하기

- Addresses.txt
  - 1 256 32768 32769 …
    - 1-> 00000000 00000001 → Page 0 offset: 1
    - 256-> 00000001 00000000 → Page 1 offset: 0
    - 32768 -> 10000000 00000000 → Page 128 offset: 0
    - 32769 -> 10000000 00000001 → Page 128 offset: 1

page	Frame
0	0
1	1
…	Invalid
128	2
…	Invalid

Physical_address.txt

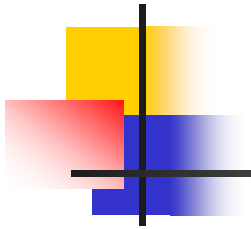
1 256 512 513



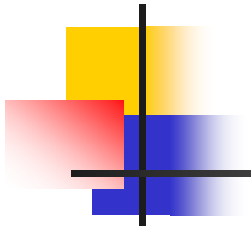
- 256 pages but 128 frames
  - Page replacement algorithm is needed !
  - LRU 교체 전략
  - FIFO 교체 전략

LRU hit ratio : 총 ( ) 중 ( ) hit 했음

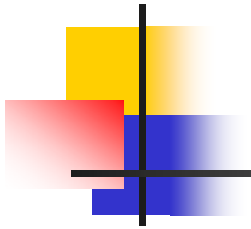
FIFO hit ratio : 총 ( ) 중 ( ) hit 했음



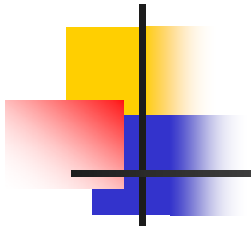
- 16 entries in TLB
  - Replacement is required
  - MRU (Most recently used) replacement policy



- Management of backing store
  - The backing store is represented by the file BACKING_STORE.bin
  - When a page fault occurs, you will read in a 256-byte page from the file BACKING_STORE.bin and store it in an available frame in physical memory
    - Physical memory = array
    - If a logical address with page number 15 resulted in a page fault, your program must read in page 15 from BACKING_STORE and store it in a page frame in physical memory



- BACKING_STORE.bin
  - A random access file so that you can randomly seek to certain positions of the file for reading
    - Use the following functions
      - fopen, fread, fseek, fclose



- 조교가 테스트하는 방법
  - memory_manager addresses.txt

실행화일 명

Virtual address 화일

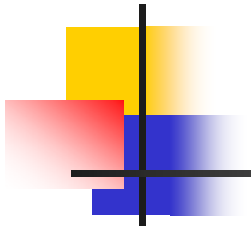
1 256 32768 32769

  - Output
    - 1. Physical.txt (파일)
    - 2. Page fault ratio (출력)
    - 3. Frame_table.txt (파일)
    - 4. Physical memory (파일)

TLB hit ratio : ( ) hits out of ( )

LRU hit ratio : ( ) hits out of ( )

FIFO hit ratio : ( ) hits out of ( )



- Deadline : 6월 6일 11시 59분
  - document 에서 프로그램 설계 원칙, 아이디어 등을 기술할 것
    - 별도 평가 예정이니 대충 쓰지 말 것
  - No submission after the deadline

- Frame table format

```
struct page {
        page_flags_t            flags;
        atomic_t                _count;
        atomic_t                _mapcount;
        unsigned long           private;
        struct address_space    *mapping;
        pgoff_t                 index;
        struct list_head        lru;
        void                    *virtual;
};
```

Linux

- 최종 frame table 출력