Operating System (운영체제)

사전 발표 준비 및 전체적인 이론 학습

운영체제

운영체제

:: 비싼 hw의 자원을 자동으로 관리

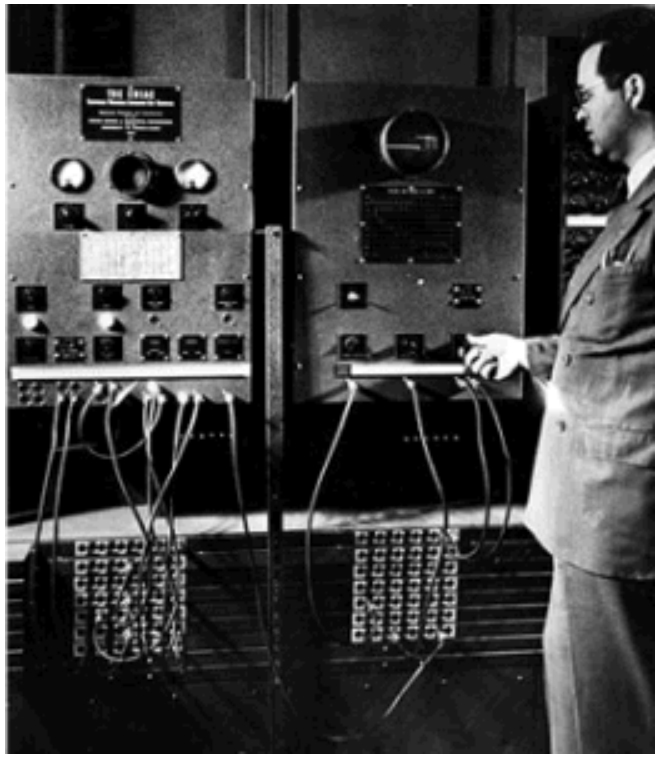
CPU, Memory, I/O

:: hw의 동작 과정을 몰라도 사용할 수 있도록 해줌

System Call

History of OS

1st Generation Vacuum Tubes and Plugboards

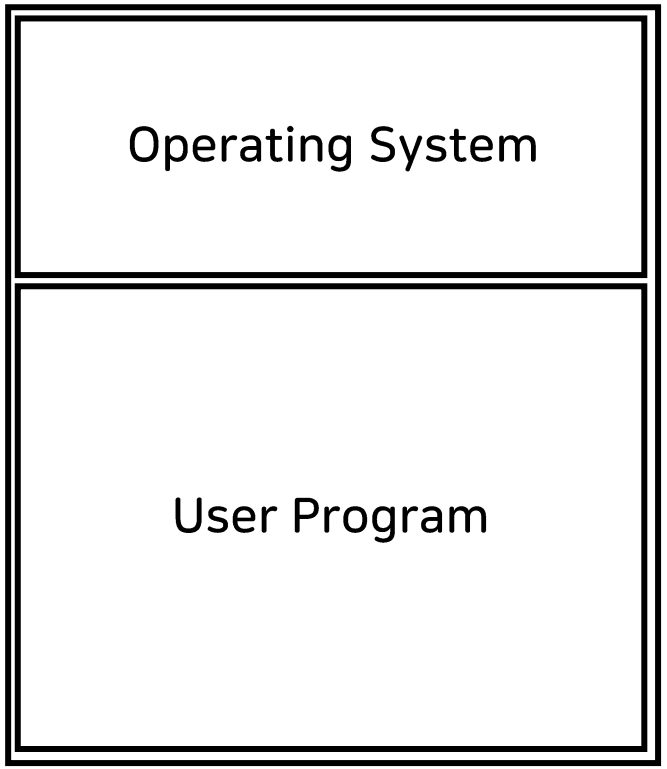


에니악/http://www.computerhistory.org

:: Eniac

No OS
No Programming Language

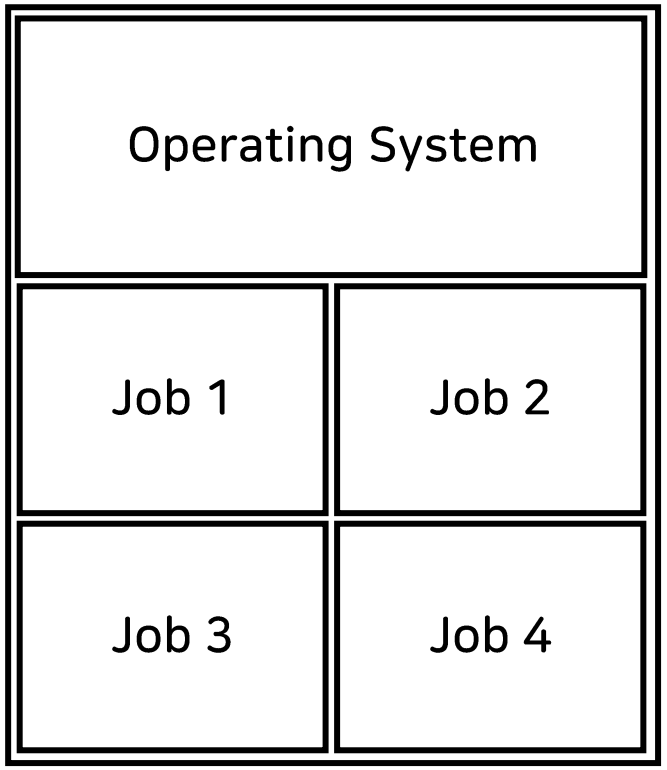
2nd Generation Transistors



:: IBM – Mainframes

OS : Batch System (Monitoring)

3rd Generation Integrated Circuits (ICs)



Notion of “Computer Architecture”

Multiprogramming System

Time-Sharing System

4th Generation Large Scale Integration (LSI)

고밀도 집적 회로에 의한 소형화 → Personal Computer

5th Generation Very Large Scale Integration (VLSI)

초고밀도 집적 회로

Classes of Computer

- Personal Computer (PC)

Cost
Individual
Third Party Program

- Personal Mobile Device (PMDs)

Small, Wireless
Access as Internet
Screen, Touch, Speech
Smartphone, Tablet

- Server

Multiple Users
Access as Network
Dependability

- Embedded Computer

Limited Function
Predetermined
Minimizing power

- Supercomputer

A lot of processor
Expensive
Scientific / Engineering Calculation → Simulation

- Cloud Computing

Provide services over the Internet
IaaS, PaaS, SaaS

Computing Environments

- Client-Server computing

- Peer-to-Peer computing

- Cloud computing

- Virtual Machine

- Distributed computing

System Call

Process

Computer 내부의 실행 단위
가장 작은 단위? ( )

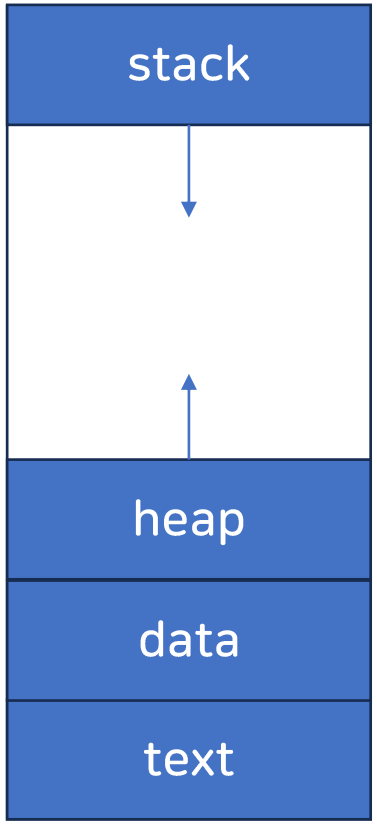
Mainframe : job
IBM, RTOS : task
Unix, Linux : process

Process

문강민

Process address space

메모리 주소 공간은 운영체제에서 프로세스가 사용할 수 있는 메모리 공간을 의미합니다.
각 프로세스는 고유한 주소 공간을 가지며, 다른 프로세스의 메모리 공간에 접근할 수 없습니다.



Stack : 함수의 호출과 관계되는 지역 변수와 매개변수가 저장되는 영역으로, 함수의 호출과 함께 할당되고 함수의 호출이 종료되면 사라진다. 재귀함수를 너무 깊게 호출되어 stack의 영역을 초과하면 stack overflow 에러가 일어날 수 있다.

Heap : 동적 메모리 할당을 위해 사용되는 공간으로 런타임 동안 메모리를 할당/해제할 수 있다. 주로 참조형 데이터(클래스, 배열, 문자열)가 할당된다.

Data : 전역 변수나 static 변수 등 프로그램이 사용할 수 있는 데이터를 저장하는 영역으로 프로그램의 시작과 함께 할당되며, 프로그램이 종료되면 소멸한다. 프로그램에서 전역/static 변수를 사용한다면 컴파일 후 data영역을 참조하게 된다.

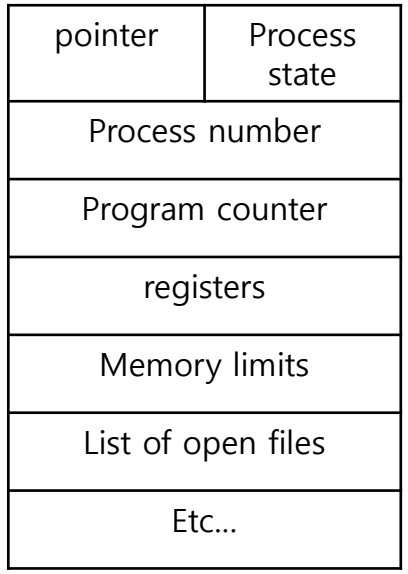
Text : 프로그램이 실행될 수 있도록 cpu가 해석 가능한 기계어 코드가 저장되어 있는 공간이다.

Process 란?

- 프로세스는 실행중인 프로그램을 의미하며, 컴퓨터에서 실행되고 있는 프로그램의 인스턴스입니다.
- 운영체제는 프로세스가 실행되는 동안 필요한 자원을 할당해주며, 프로세스는 자원을 사용하여 작업을 수행하고, 작업이 끝나면 자원을 운영체제에 반환합니다.
- 하나의 프로세스는 하나 이상의 쓰레드를 포함할 수 있습니다. 모든 쓰레드는 동일한 프로세스 자원을 공유하지만, 독립된 실행 흐름을 가집니다.

Process control block 이란?

운영체제가 각 프로세스를 관리하기 위해 사용되는 데이터 구조입니다. 운영체제에서 프로세스는 PCB로 나타내어집니다.
프로세스가 cpu를 점유하는 상황에서 상태가 전이되면, 진행하던 작업 내용들을 모두 PCB에 저장합니다.



Process number: 프로세스의 id
Process state: 프로세스의 상태(create, ready, running, waiting, terminated)
Program counter: 해당 프로세스 다음에 실행될 명령어 주소 저장
Registers: 데이터를 저장하는 공간
Memory lists: 프로세스가 사용하는 메모리에 관한 정보 저장
Pointer: 부모 프로세스에 대한 포인터, 자식 프로세스에 대한 포인터 등 필요한 자원의 포인터 정보를 저장합니다.

Stack Pointer 란?

스택의 최상단 주소를 저장하는 레지스터입니다. 스택에 데이터를 추가하게 되면 스택 포인터는 위로 이동하여 새로운 데이터의 위치를 가리킵니다.
스택에서 데이터를 제거하게 되면 스택 포인터는 아래로 이동하여 이전 데이터의 위치를 가리킵니다.

함수가 호출되면 호출된 함수의 반환 주소가 스택에 푸시되고 스택 포인터가 업데이트됩니다. 그 후 함수의 지역 변수가 스택에 할당되게 됩니다.
함수의 실행이 끝나면 스택에서 반환 주소를 팝하고 이전 함수로 돌아가며 스택 포인터가 업데이트 됩니다.

Program counter란?

프로그램 카운터는 메모리에서 실행할 다음 명령어의 주소를 저장하는 레지스터입니다.
프로그램을 실행시키면 프로그램 카운터가 코드를 한줄씩 읽어 나가면서 다음줄의 코드 주소를 프로그램 카운터에 저장합니다.

If, goto 명령어가 있다면 프로그램 카운터는 조건에 따라 다른 주소를 가리키게 됩니다.

Example

```python
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)

result = fibonacci(5)  # F(5) = 5
```

Process address space

텍스트 영역: 피보나치 함수의 코드 저장
데이터 영역: null
힙 영역: null
스택 영역: Fibonacci(n) 호출마다 새로운 스택 프레임이 생성됩니다. 해당 스택 프레임에는 매개변수의 값, 지역 변수의 값, 반환 주소가 저장됩니다.

PCB

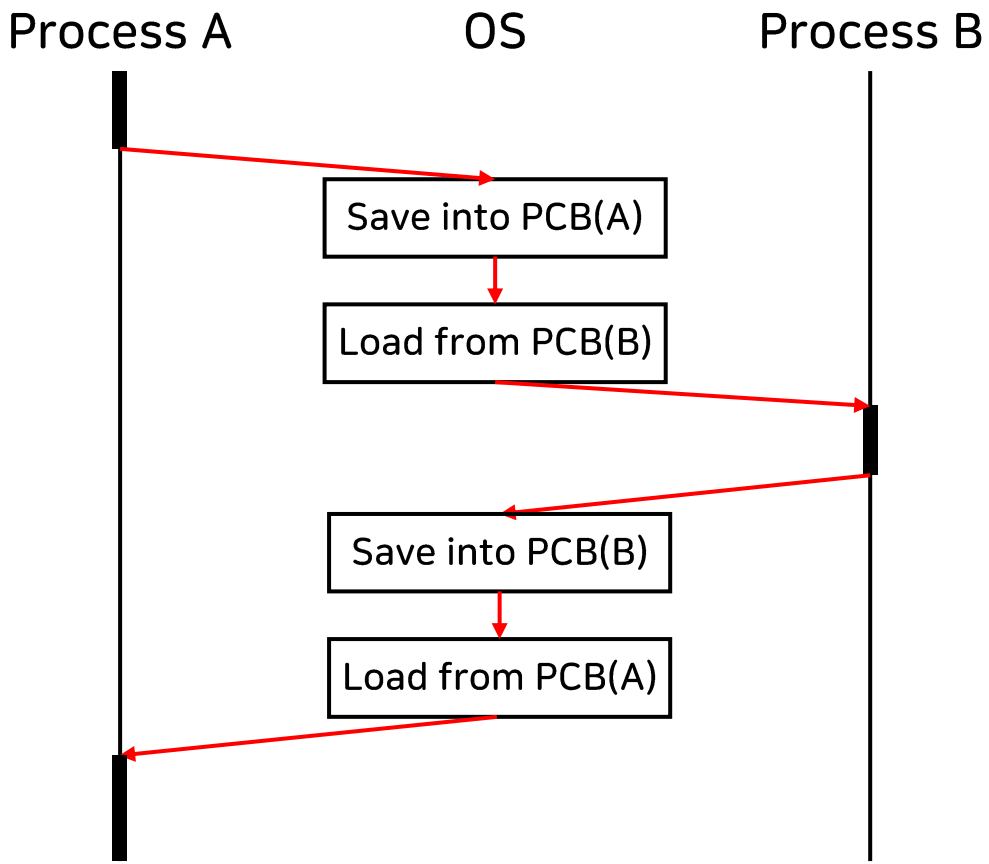
프로세스 식별 정보: 프로세스ID, 부모 프로세스ID가 저장됩니다.
프로세스 상태: 현재 프로세스의 상태를 저장합니다. 재귀 호출시 대기 상태로 전환됩니다.
CPU 레지스터: 피보나치 함수의 매개변수와 관련된 값을 저장합니다.

PC

다음에 실행될 명령어의 주소를 저장합니다. 피보나치 함수 내에서 조건문이나 재귀 호출이 위치하는 주소가 저장됩니다.

Context Switch (CPU Switch)

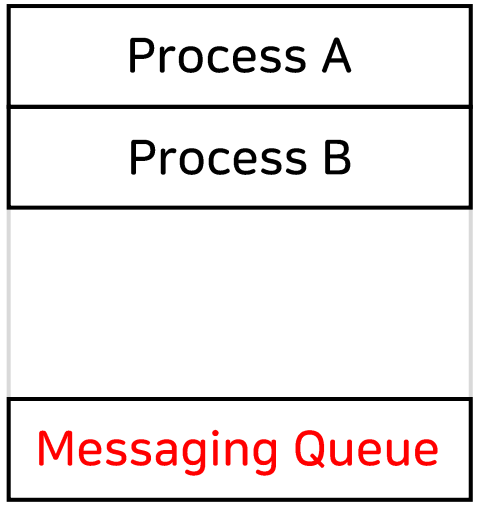
A process를 수행하다가 동일한 CPU에서 B process 수행하기



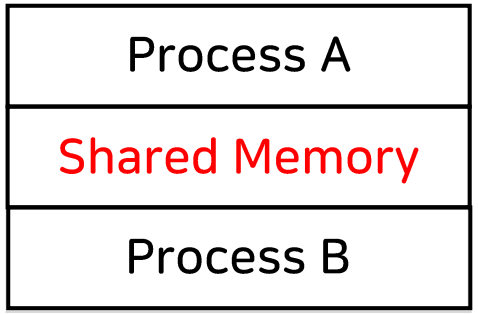
Inter Process Communication (IPC)

Process끼리 Data를 공유하는 방법

- Message Passing



- Shared Memory



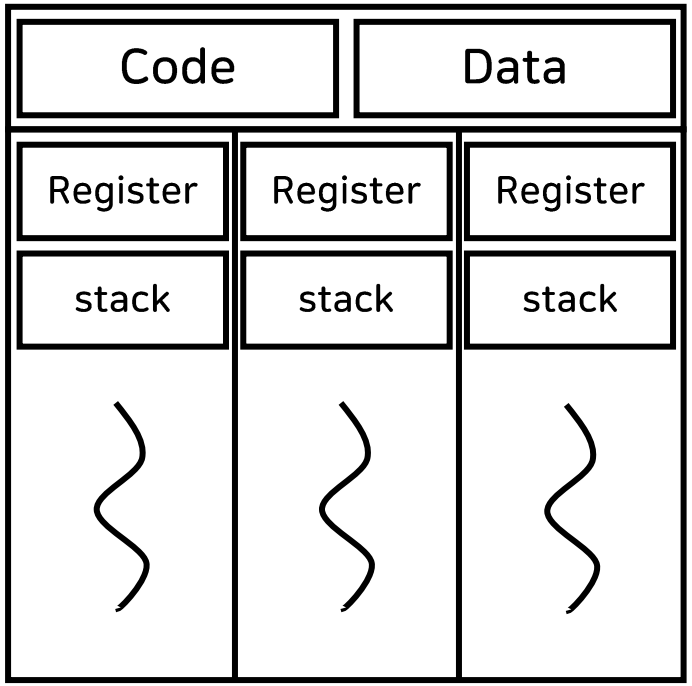
Issue 1. Synchronization
Issue 2. Performance

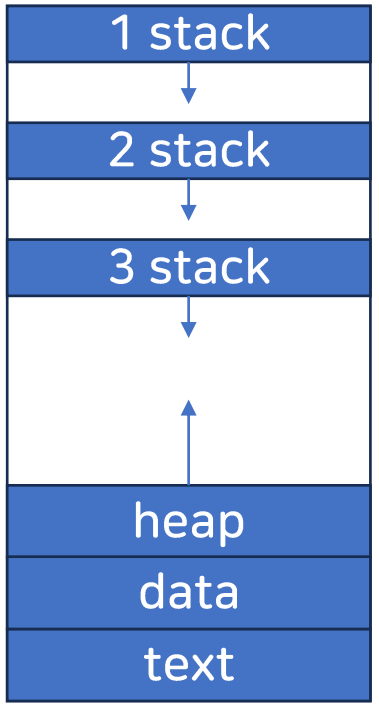
Thread

Computer 내부에서 가장 작은 단위
Process가 너무 costly

Thread
LWP (Light Weight Process)

Multithreaded Process





SP : ( )개
PC : ( )개

Synchronization

Bounded Buffer Problem

Readers-Writers Problem

Dining Philosopher Problem

Semaphore (numeric, binary)

Mutex Lock

Monitor & Condition Variables (in Java)

Deadlock

2개 이상의 process/thread에서 shared data에 접근하여 사용할 때, 발생 가능함.
다음 4가지 조건을 모두 만족해야만, 발생할 수 있음

Mutual Exclusion
(상호 배제) : 자원은 한번에 하나의 프로세스에서만 접근

Hold and wait
(점유 대기) : 자원을 선점한 상태에서 대기

No preemption
(비선점) : 강제로 빼앗을 수 없음

Circular wait
(순환 대기) : deadlock 상태에 있는 프로세스끼리 서로를 대기

Main Memory

Virtual Memory

< Program Execution >

Load : Storage → Main Memory

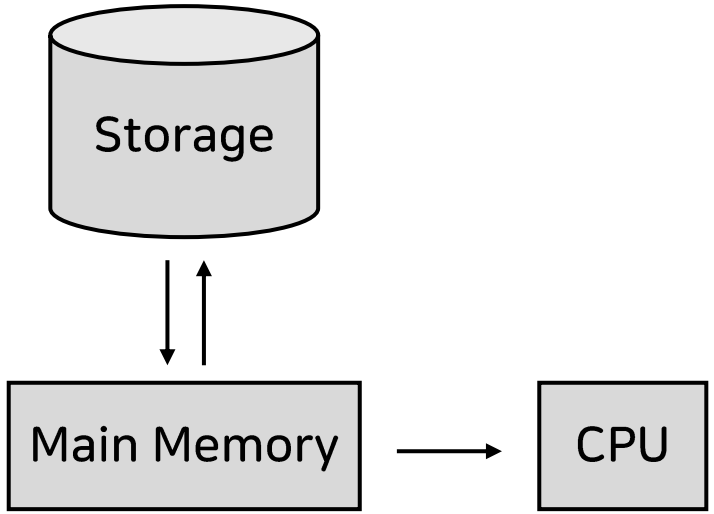
Fetch : Main Memory → CPU
(Ready Queue)

Decode

Execution

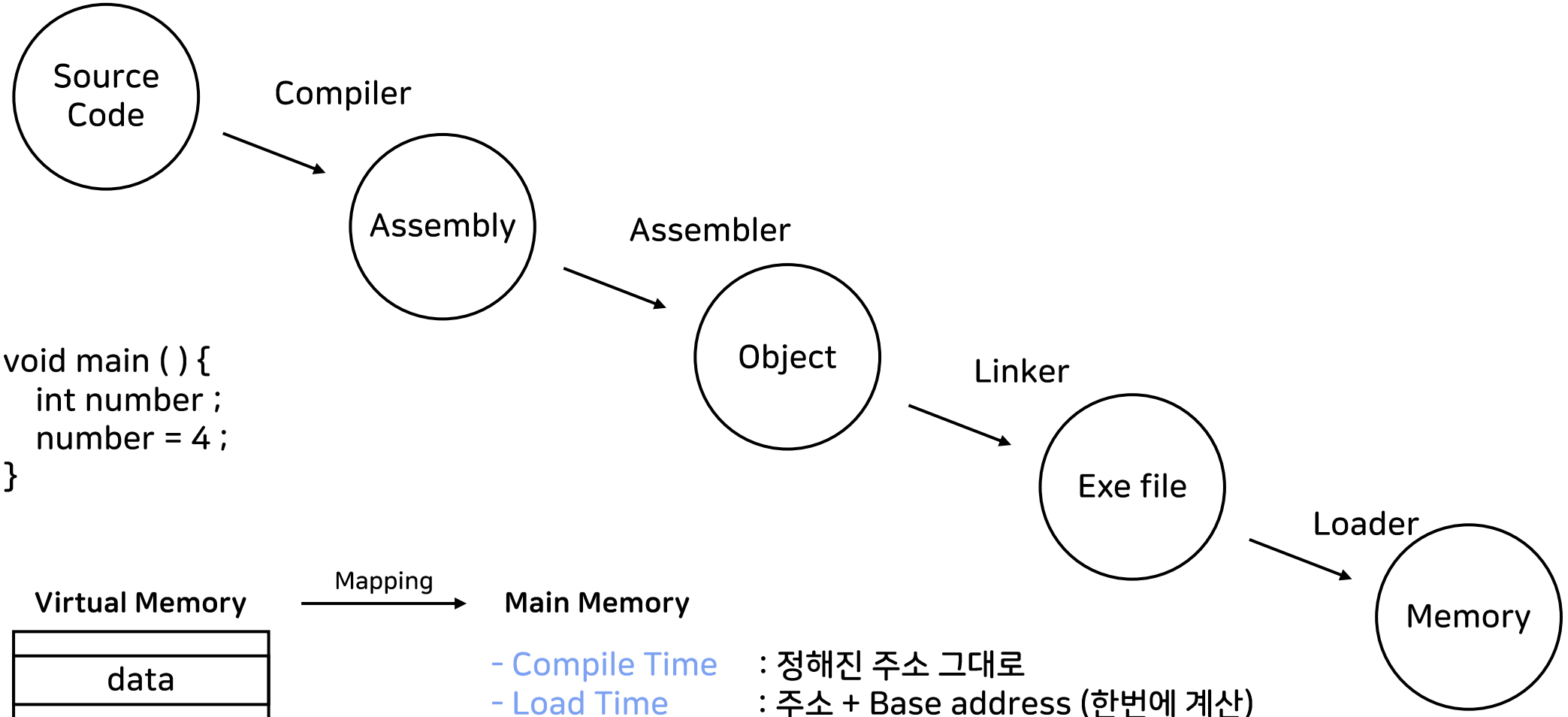
Write Back

Long term scheduler (Job Scheduler) : Load
Short term scheduler (CPU Scheduler) : Fetch
Medium term scheduler (swapper) : Reverse Load



Virtual Memory

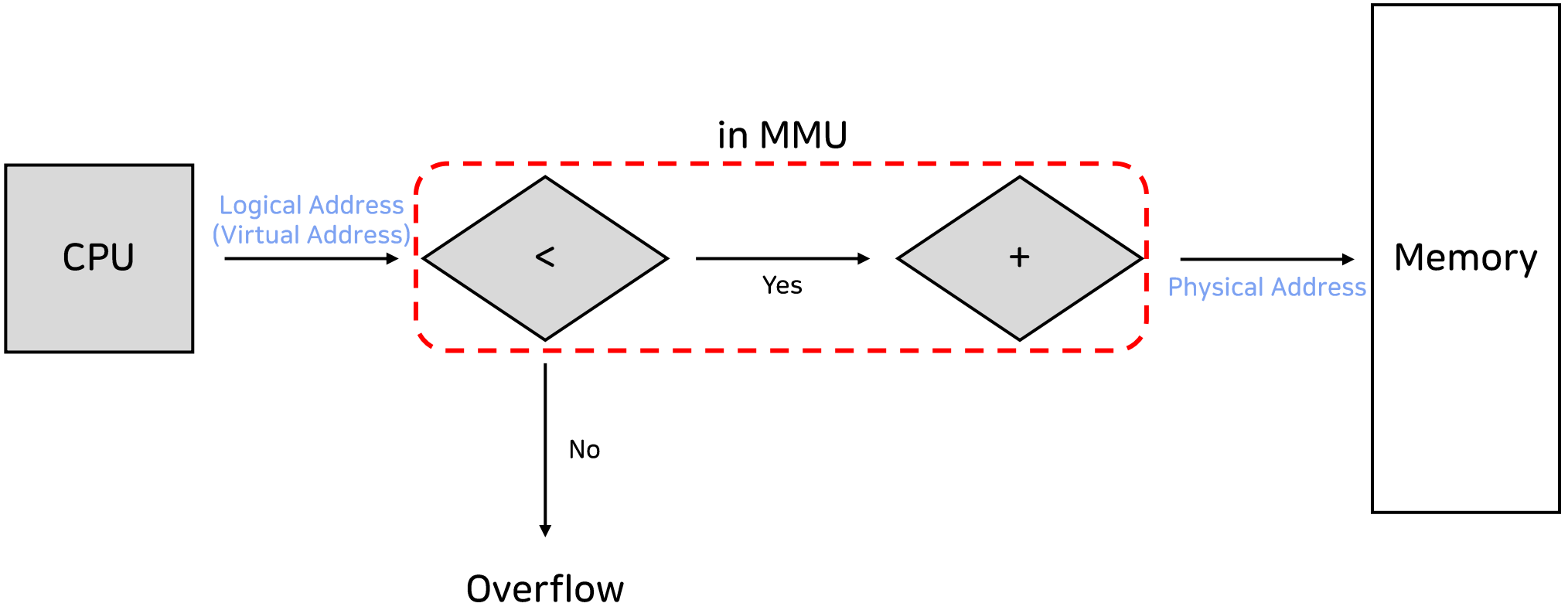
1. “Main Memory는 유한하다.”


Source Code
Compiler
Assembly
Assembler
Object
Linker
Exe file
Loader
Memory
void main ( ) {
int number ;
number = 4 ;
}
Virtual Memory
Mapping
Main Memory
100
data
store #100, 4
- Compile Time : 정해진 주소 그대로
- Load Time : 주소 + Base address (한번에 계산)
- Execution Time : 주소 + offset (매번 계산)
MMU

Virtual Memory

1. “Main Memory는 유한하다.”

2. Virtual Address → Physical Address
(Mapping / Translation)


in MMU
CPU
Logical Address
(Virtual Address)
<
Yes
+
Physical Address
Memory
No
Overflow

Virtual Memory

1. “Main Memory는 유한하다.”

2. Virtual Address → Physical Address
   (Mapping / Translation)

3. MMU를 사용하여 Protection

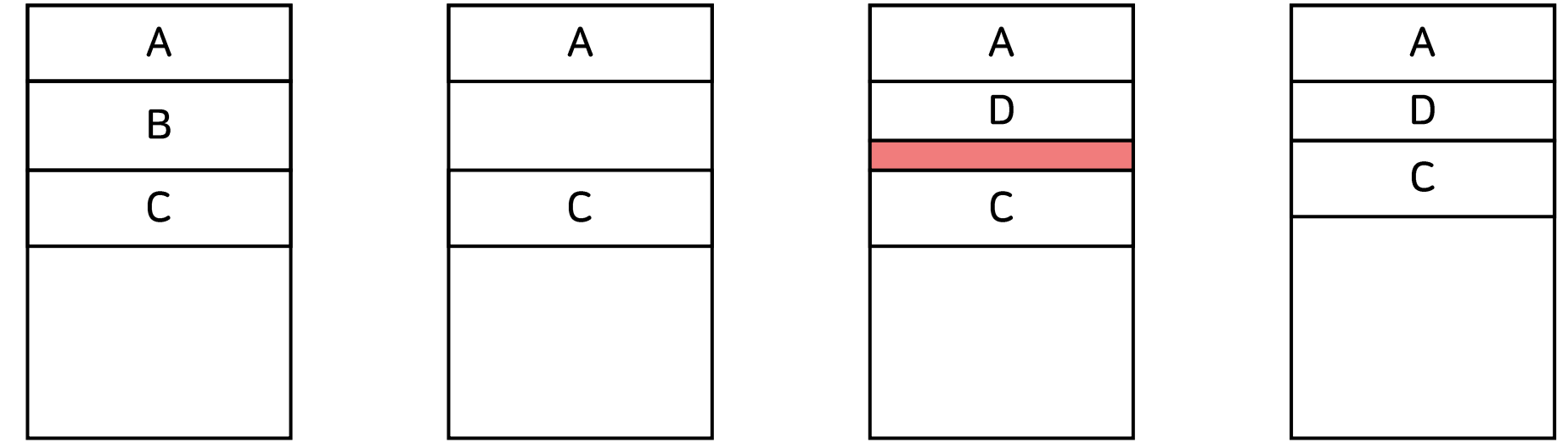
Virtual Address → Physical Address

Way / Solution Problem

Contiguous Allocation (연속적인 할당)

↳ Hole (External Fragmentation)

↳ Compaction (압축)

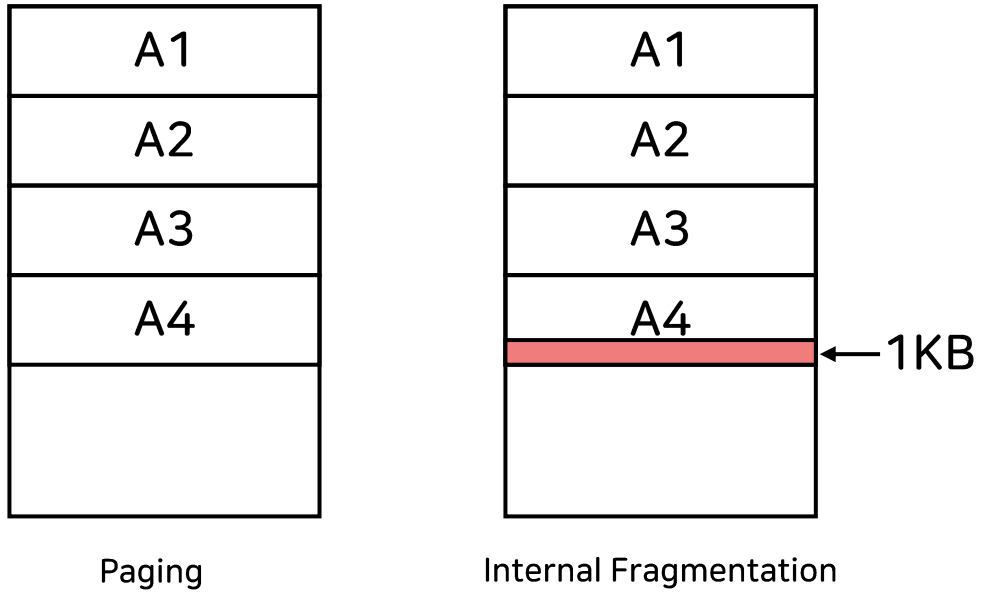


Contiguous Allocation External Fragmentation Compaction

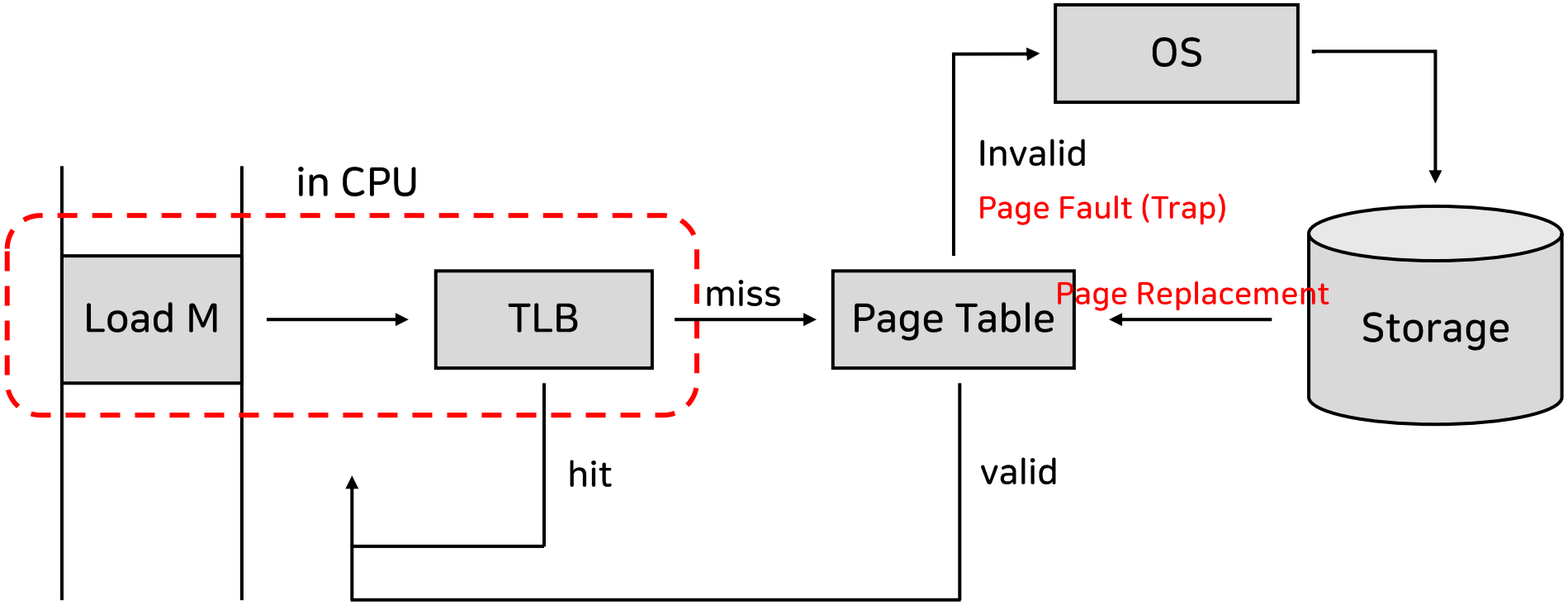
Paging : 특정 사이즈(4kb)로 자르기

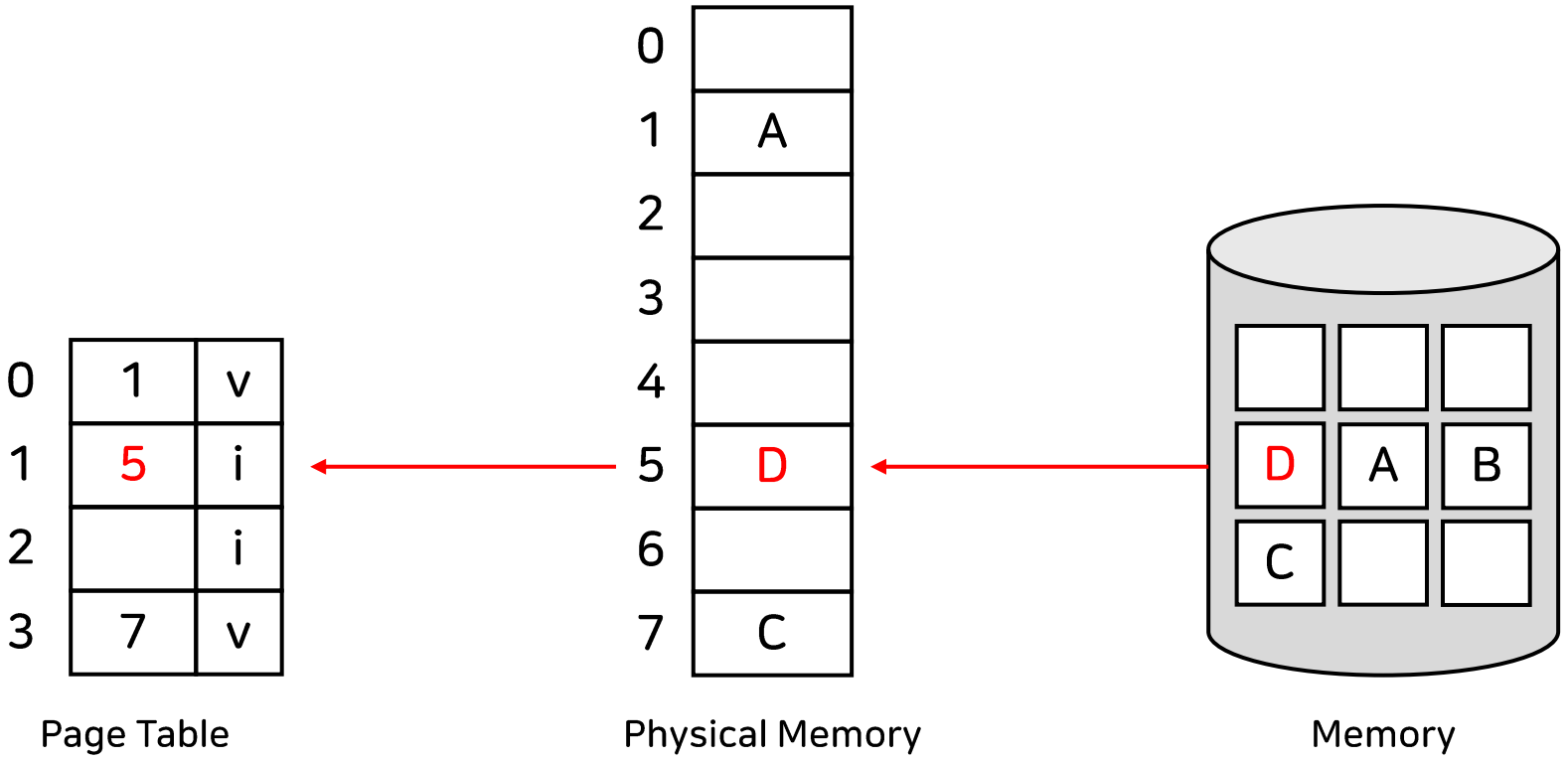
↳ Internal Fragmentation

↳ Segmentation (분할) : 의미있는 단위로 쪼개기 (Code, Data, Heap, Stack, …)



Segmentation + Paging


OS
Invalid
Page Fault (Trap)
in CPU
Load M
TLB
miss
Page Table
Page Replacement
Storage
hit
valid


0
1
2
3
0 1 v
1 5 i
2 i
3 7 v
Page Table
0
1 A
2
3
4
5 D
6
7 C
Physical Memory
D A B
C
Memory

Page Replacement Algorithms

FIFO (First In First Out)

Belady's anomaly : Frame이 많으면 Page Faults가 덜 발생된다? No!

LRU (Least Recently Used)

참조된 시간을 담아둘 bit → Approximation 1 bit (Reference bit)

NRU (Not Recently Used)

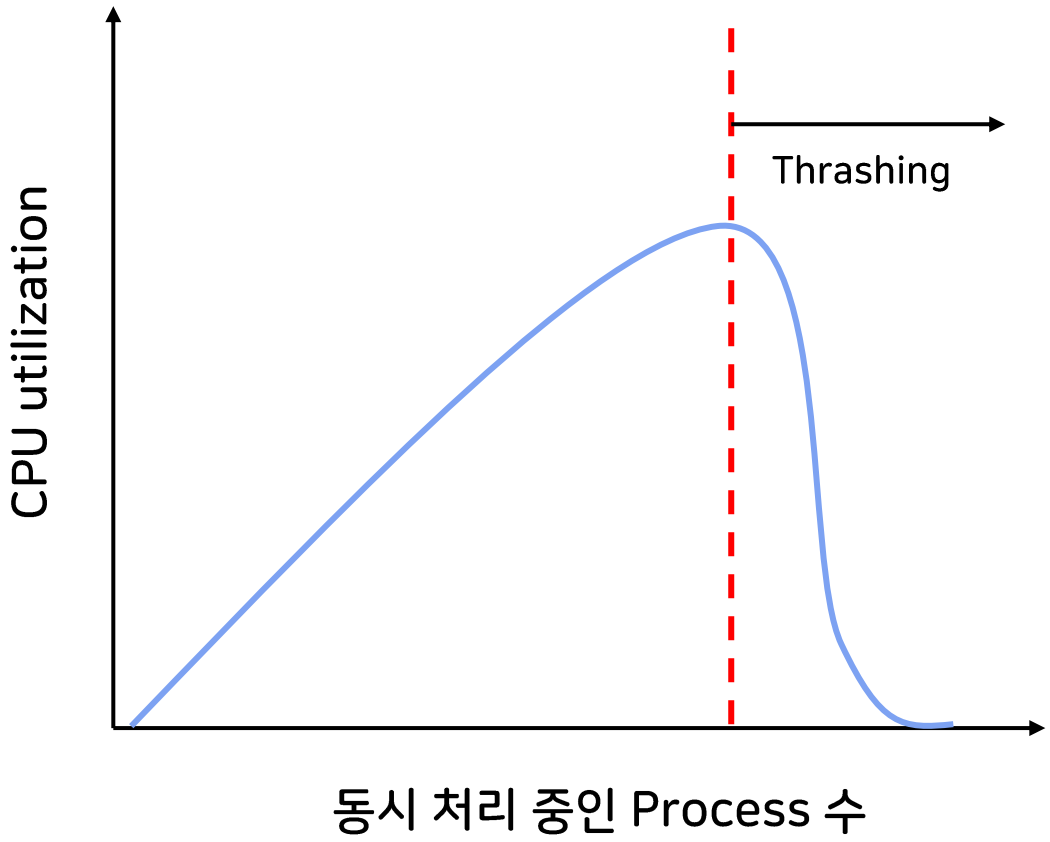
최근에 읽거나 변형이 일어나지 않은 page 교체 → R (reference bit) + M (modify bit)

LFU (Least Frequently Used)

횟수 기반 → Approximation 1 bit (Counter bit)

Thrashing : 동시에 처리하는 Process의 수가 너무 많아 Page Fault가 많이 발생해, 오히려 CPU 사용량이 줄어듦

따라서, 무조건 Multiprocessing이 좋은 것은 아니다.



Storage

HDD (Hard Disk Drive)

SSD (Solid State Disk)

RAM Disk

ex) iptime 공유기

NAS (Network Attached Storage)

IP network로 접근
File System

SAN (Storage Area Network)

ex) Database, 가상화 환경

Private network로 접근
Block 단위 access

HDD (Hard Disk Drive)

FCFS : 들어온 순서대로

SSTF (Shortest Seek Time First) : 현재 위치에서 가장 가까운 것

SCAN : 왼쪽으로 순환하면 왼쪽 탐색, 오른쪽으로 순환하면 오른쪽 탐색

C-SCAN (Circular) : 오른쪽으로만 순환

C-LOOK (Circular) : 한쪽 끝까지으로 찍지않고, 처리

I/O system

Buffering : 데이터를 생산하고, 사용하는 것에 대해 속도의 차이가 발생할 경우

Caching : 다시 사용될 가능성이 높은 데이터를 Cache에 저장

Spooling : 아주 느린 device를 위해 file을 image로 처리하여 전달 (ex. 프린터기)

File system

	Windows	Linux
확장자	O	X
Mount (용량추가)	Drive	Root
RFS Remote File System	CIFS	NFS

↳ Network Memory를 내 드라이브처럼 사용

$ chmod ___ {file_name}

Owner / Group / Others(public)

Read / Write / Execute

```
drwxrwxr-x 8 kyungsik kyungsik 73728 Nov  5 21:13 .
drwxrwxr-x 3 kyungsik kyungsik  4096 Oct 25 13:28 ..
drwxrwxr-x 7 kyungsik kyungsik  4096 Oct 30 22:51 experiments
drwxrwxr-x 8 kyungsik kyungsik  4096 Nov  1 22:01 .git
-rw-rw-r-- 1 kyungsik kyungsik  1242 Oct 25 13:28 .gitignore
drwxrwxr-x 2 kyungsik kyungsik  4096 Oct 25 13:28 img
-rw-rw-r-- 1 kyungsik kyungsik 11357 Oct 25 13:28 LICENSE
drwxrwxr-x 2 kyungsik kyungsik  4096 Oct 25 13:28 plots
-rw-rw-r-- 1 kyungsik kyungsik  2584 Oct 25 13:28 README.md
drwxrwxr-x 2 kyungsik kyungsik  4096 Nov  5 21:09 scripts
drwxrwxr-x 5 kyungsik kyungsik  4096 Oct 25 14:08 src
```

Task

T/F - 3문제

서술형 - 2문제

Computer Network

컴퓨터 네트워크

OSI 7계층 & TCP/IP 4계층	각 계층 설명, 예시, 둘의 관계, 예시 device (ex. 라우터, 스위치)
IP 주소, MAC 주소	각 주소 설명, ARP/RARP, Hop by Hop
HTTP	기본 설명, HTTP/1.0 ~ HTTP/3

Curriculum

1주차 : Orientation	(9/4)
2주차 : Data Structure	(9/11)
3주차 : Algorithm 1	(9/25)
4주차 : Algorithm 2	(10/2)
5주차 : Computer Architecture	(10/16)
6주차 : Operating System	(11/6)
7주차 : Computer Network	(11/13)
8주차 : DataBase	(11/20)
9주차 : Design Pattern	(11/27)
10주차 : Security	(12/4)