Demand Paging

Minsoo Ryu

Operating Systems and Distributed Computing Lab.
Hanyang University

msryu@hanyang.ac.kr

Topics Covered

- ❐ **Memory Space Utilization** '과거의 메모리 효율을 높이는 방법'
- ❐ **Demand Paging**

Obtaining Better Memory-Space Utilization

- **In early approaches**
  - The entire program and data of a process must be in physical memory for the process to execute
  - The size of process is limited to the size of physical memory

- **For better memory-space utilization, we can use**
  - Dynamic loading
  - Overlays
  - Dynamic linking
  - Swapping

운영체제가
필요로하는 프로그램의 메모리를
제공해주는 기법

Dynamic Loading

- **A routine is loaded into memory only when it is called by some other routine**
  - **To start with, only the main routine is loaded into memory**
  - **When the main routine calls another routine, that is loaded into memory**
- **This can be quite significant**
  - **Very often a particular execution path in a program may not use all the routines**
- **The overhead is that it takes time to load a program into memory if it is not already there**

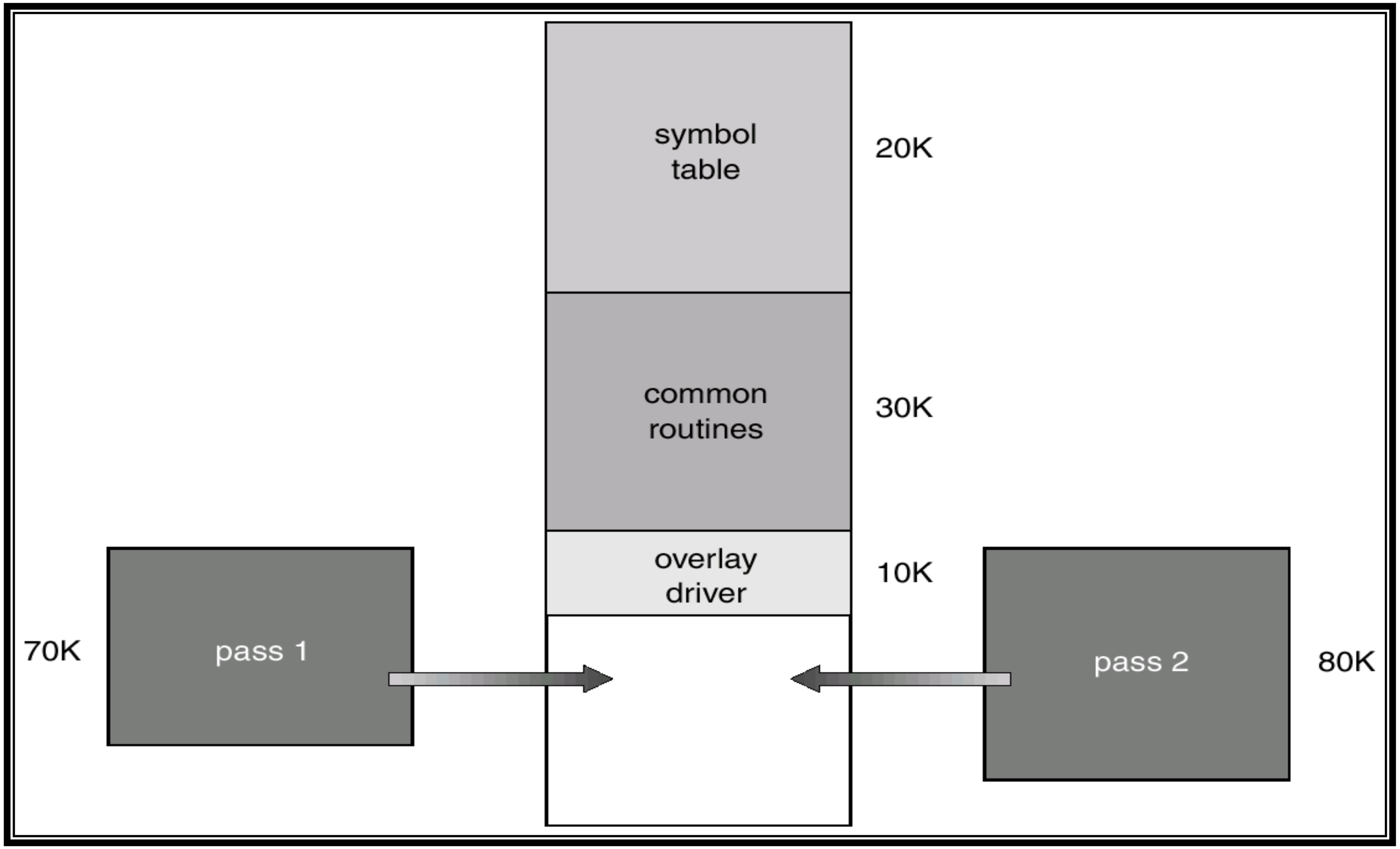
프로그램 개발자가 스스로
프로그램을 dynamic loading이 가능하도록
구성하는 방법

Overlays

- **Keep in memory only those instructions and data that are needed at any given time**

- **Implemented by user**
  - **A programmer writes an overlay driver in addition to its core functionality**
  - **The function of the** overlay **driver is to simply load the relevant code of each phase into memory before the phase starts**
  - **No special support needed from operating system**
  - **Some microcomputer compilers provide the programmer with support for overlays to make the task easier**

Overlays for a Two-Pass Assembler



link 작업을 보류해뒀다가
나중에 실행할때 필요한
라이브러리를 link 하는 기법

Dynamic Linking

- **Linking is postponed until execution time**
  - **Small piece of code, stub, used to locate the appropriate memory-resident library routine**
  - **Stub replaces itself with the address of the routine, and executes the routine**

- **Dynamic linking generally requires help from the OS**
  - **OS should check whether the needed routine is in another process’s memory space or allow multiple processes to access the same memory addresses**

- **Dynamic linking is particularly useful for shared libraries**

메모리가 부족해지면 OS가 active 하지않은
몇 개의 프로세스의 실행을 중단하고
그 프로세스들의 이미지를 하드디스크 드라이브에 write : swap out

제한된 physical memory를 효율적으로 활용하는 기법

write : swap out

실행 중이던 프로세스 이미지 하드디스크 드라이브

copy : swap in

Swapping (1)

- ❒ Swap a process temporarily out of memory to a backing store
  - Bring it back into memory later for continued execution
  - (e.g.) In round-robin scheduling, when a quantum expires, the memory manager will swap out the process that just finished and swap in another process to the memory
- ❒ Backing store
  - Fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images
  - Chunk of disk separate from the file system

성능 비용감수 필요

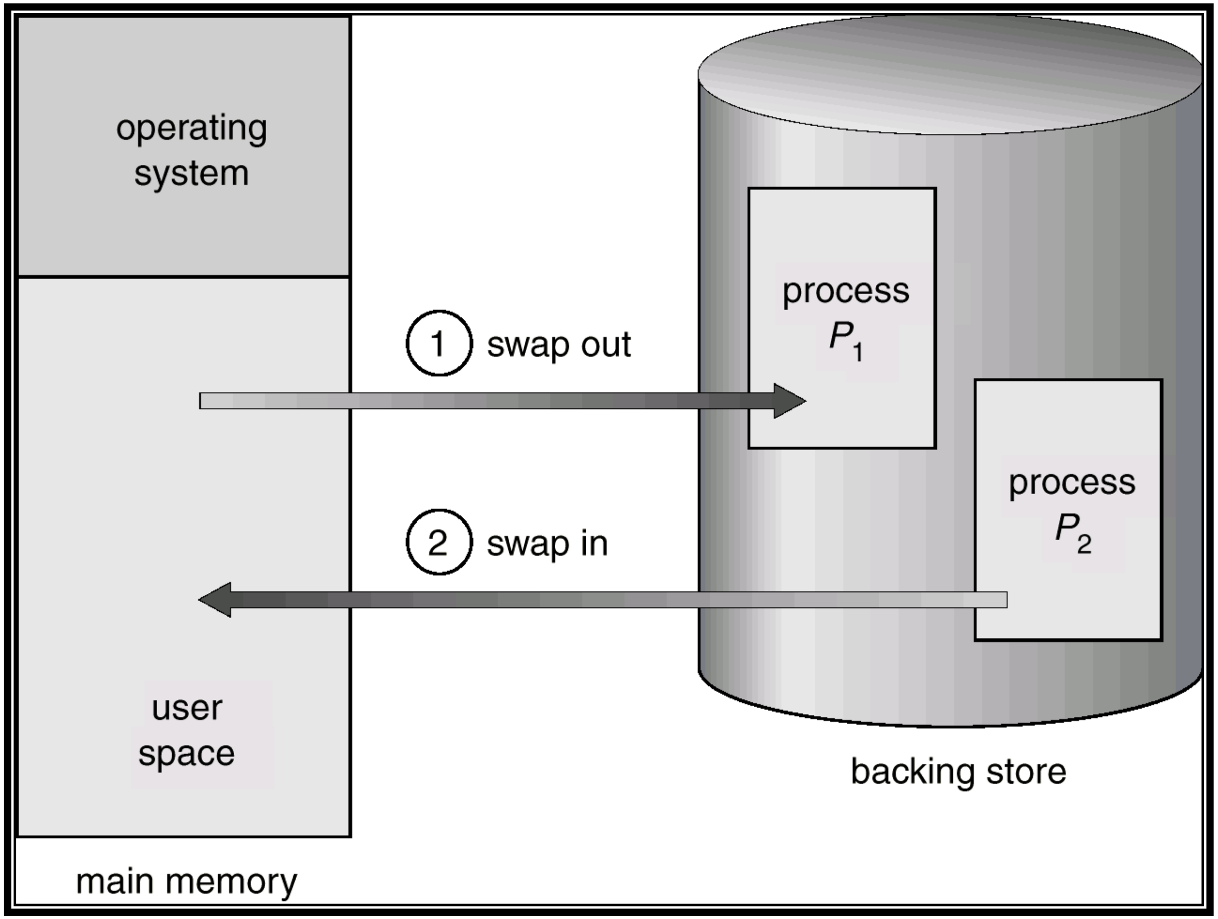
Swapping (2)

Disk I/O 작업은
일반적인 CPU 작업에 비해 굉장히 느리다

- **Major part of swap time is transfer time** 시간이 걸림
  - **Total transfer time is directly proportional to the *amount* of memory swapped**

- **Modified versions of swapping are found on many systems, i.e., UNIX, Linux, and Windows**
  - **In UNIX, swapping is normally disabled**
  - **Enabled when many processes are running**

Schematic View of Swapping



Topics Covered

- Memory Space Utilization
- Demand Paging

프로그램이 필요로 하는 메모리 공간만 배정

Demand Paging vs. Swapping

swap 단위 : page process 전체

❐ Demand paging

- A page is not loaded into main memory until it is needed
- Swap pages out of memory to a backing store
- Swap pages into memory when they are needed

❐ Swapping

- Swap processes temporarily out of memory to a backing store when the system runs out of physical memory

프로세스가 당시에 필요로 하지 않는 page만 선별해서 하드디스크 드라이브로 옮기기 때문에
프로세스 실행에는 큰 영향을 주지 않으면서 메모리를 절약하는 기법

Page Fault

❐ Page fault

메모리 주소 참조 시 유효하지 않을때 발생하는 exception

- **The first memory access would cause an exception, called page fault**

page table에 존재하는 기능

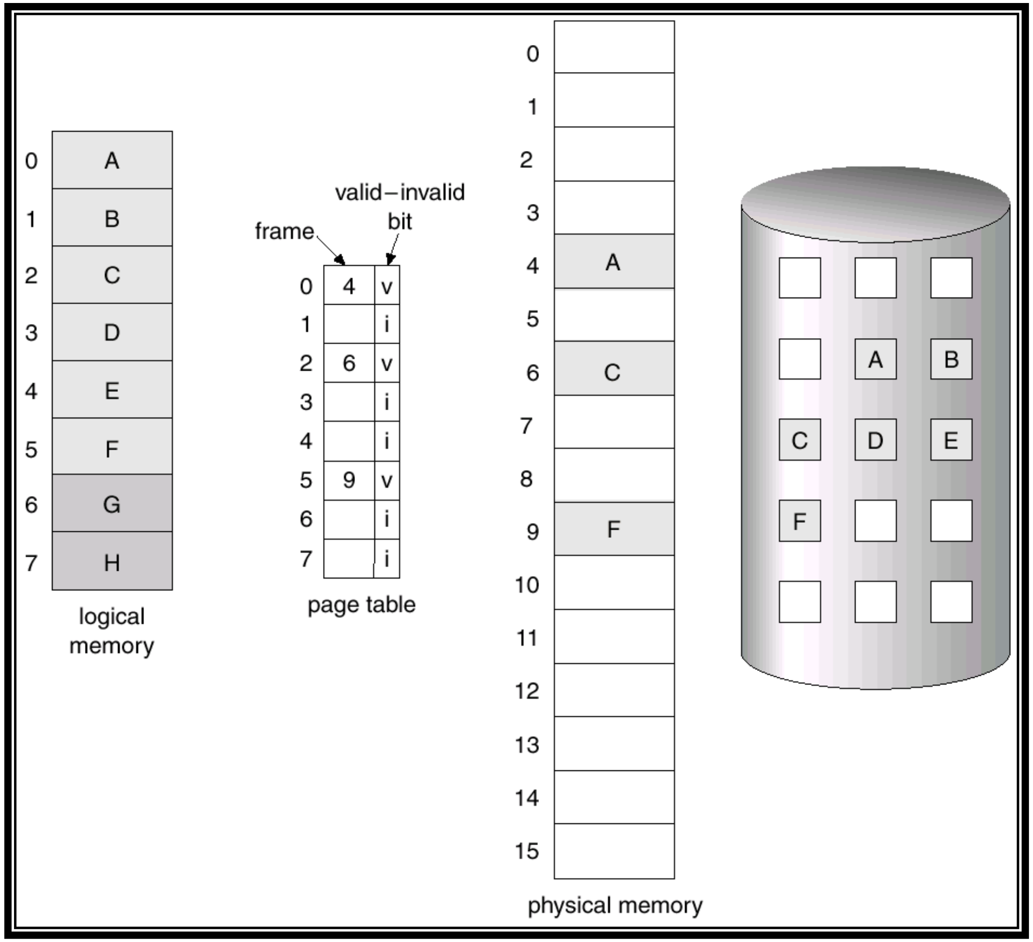
- **Exception is raised when valid-invalid bit is invalid**
  - Valid–invalid bit is initially set to 0 (invalid) on all entries
  - Valid-invalid bit indicates the page table entry is valid or not

→ 프로세스가 필요로 하는 page 만 선별해서 메모리에 로딩하는것이 가능해짐

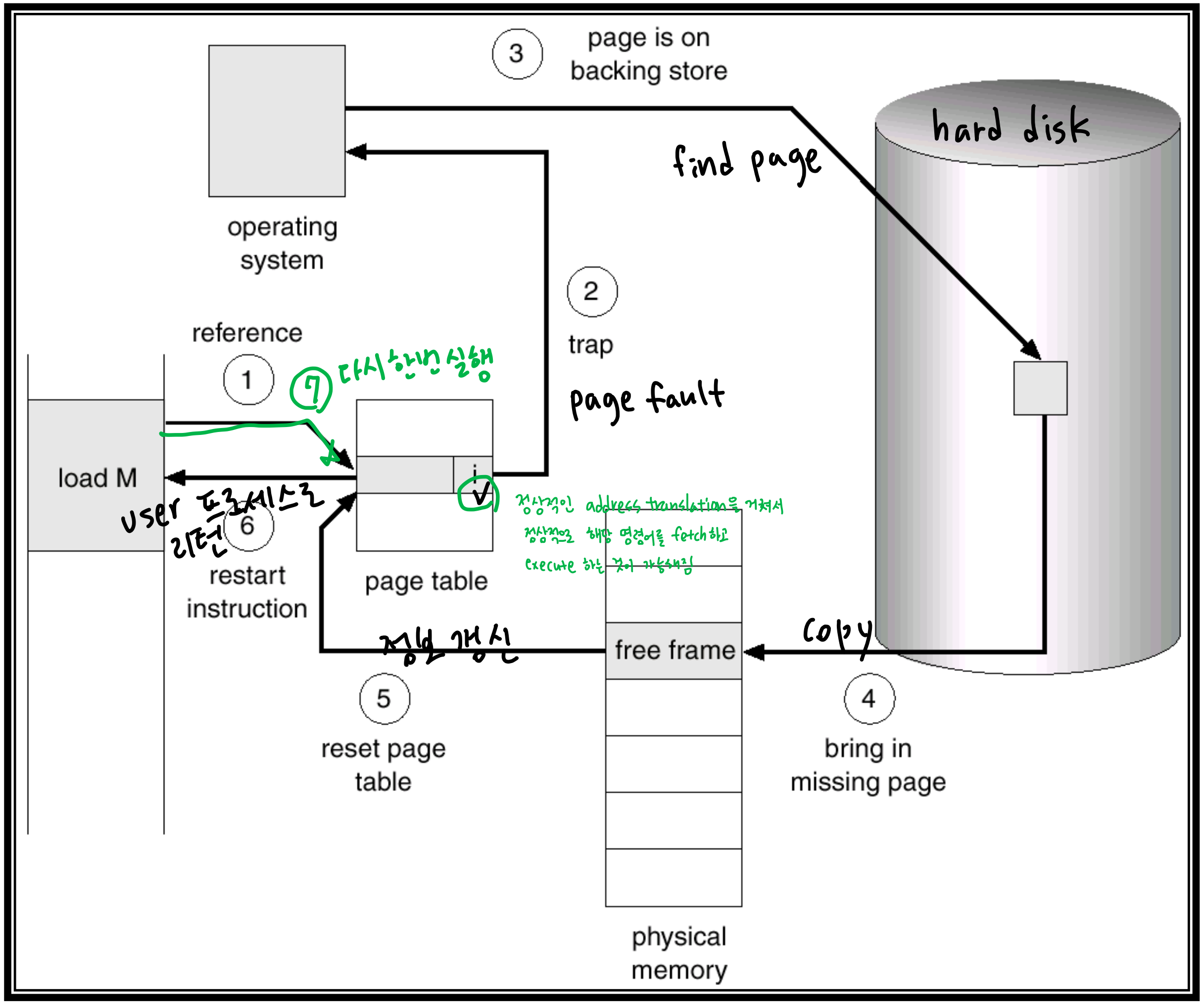
❐ Page fault handling

- **If there is a reference to an invalid page, the reference will trap to OS**
- **OS gets an empty frame, brings the desired page into the frame, and sets the valid-invalid bit to 1**
- **Restart the instruction that was interrupted by the trap**

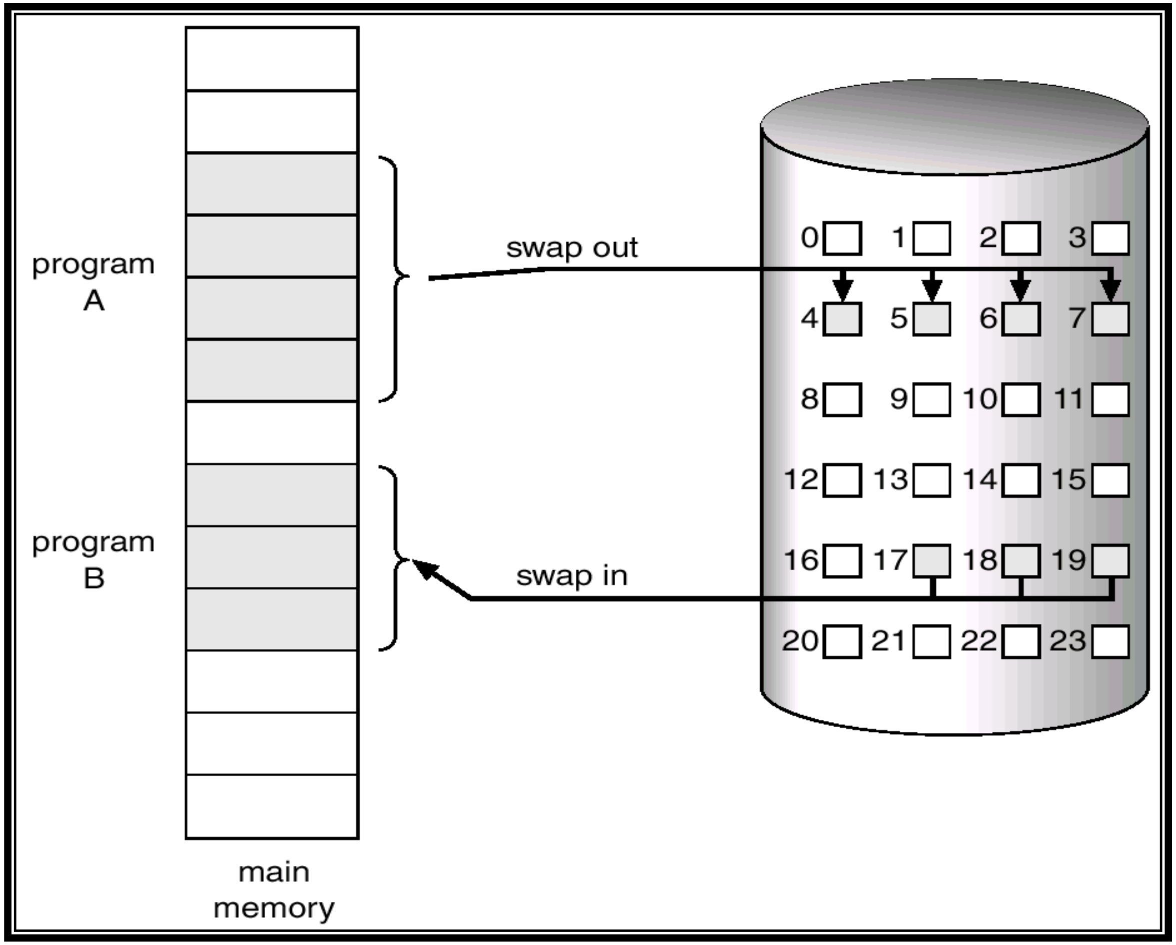
Page Table When Some Pages Are Not in Memory



Steps in Handling a Page Fault

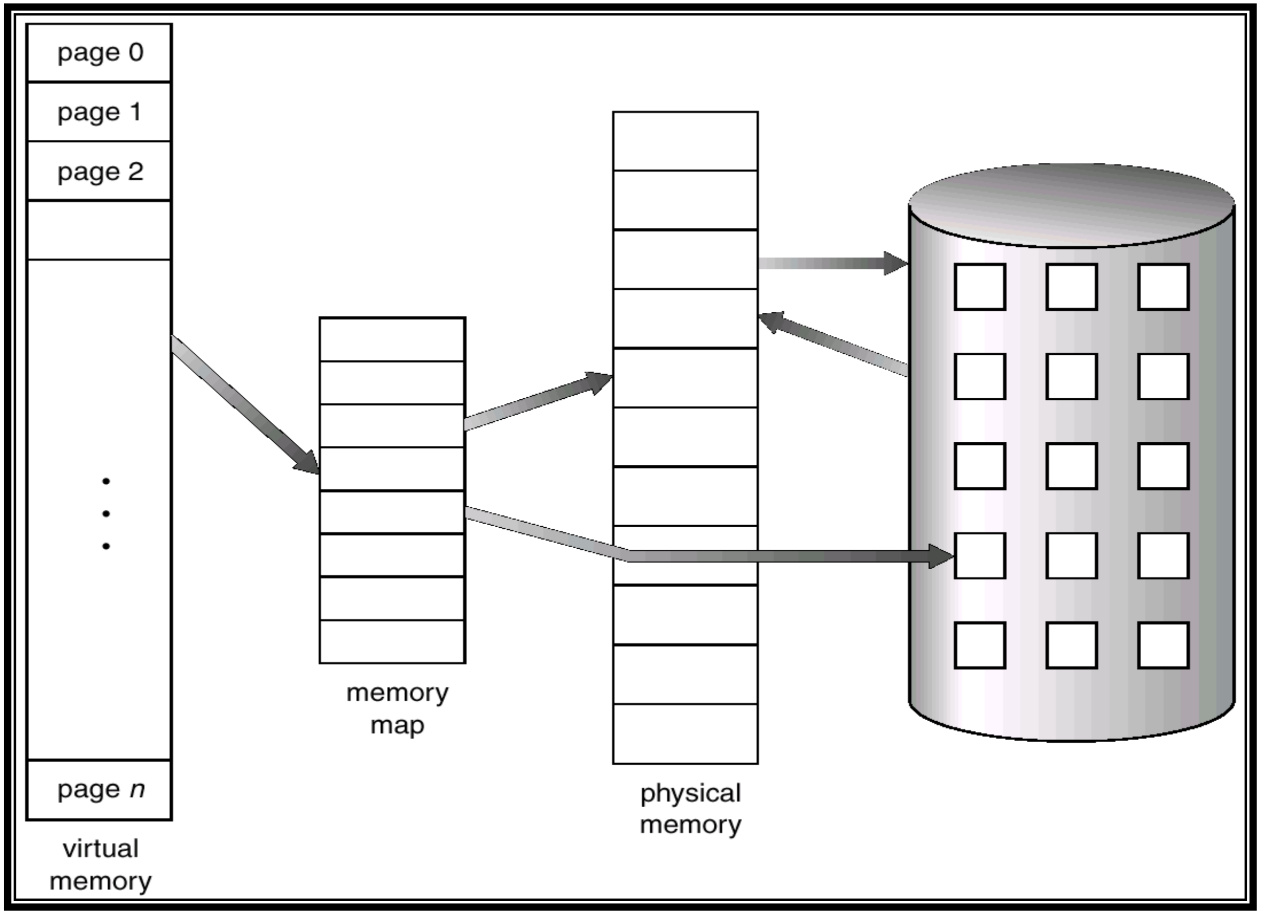


Swapping



program
A
program
B
swap out
swap in
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
main
memory

Demand Paging



Performance of Demand Paging

- ❒ **Page Fault Rate $0 \le p \le 1.0$**
  - **if $p = 0$ no page faults**
  - **if $p = 1$, every reference is a fault**

page fault의 학률을 줄이는것이
성능면에서 매우 중요한 일이 된다

- ❒ **Effective Access Time (EAT)**
  - **EAT = (1-p)m + pf**
    - Memory access time (m)
    - Page fault handling time (f)
  - **e.g.,**
    - m = 100 nanosec
    - f = 25,000,000 nanosec → 하드디스크에 데이터를 쓰고 읽는 작업이므로 메모리의 동작속도보다 훨씬 느림
    - EAT = (1-p)100 + 25,000,000 x p = 100 + 24,999,900 x p
    - What if we want less than 10% degradation ? – demand paging을 썼을 때 성능저하가 10% 이하
      - ✓ 110 > 100 + 24,999,900 x p
      - ✓ 0.0000004 > p


Q
A
thank you!