Memory Management

Jo, Heeseung

Today's Topics

Why is memory management difficult?

Old memory management techniques:

- Fixed partitions
- Variable partitions
- Swapping

Introduction to virtual memory

Memory Management (1)

Goals

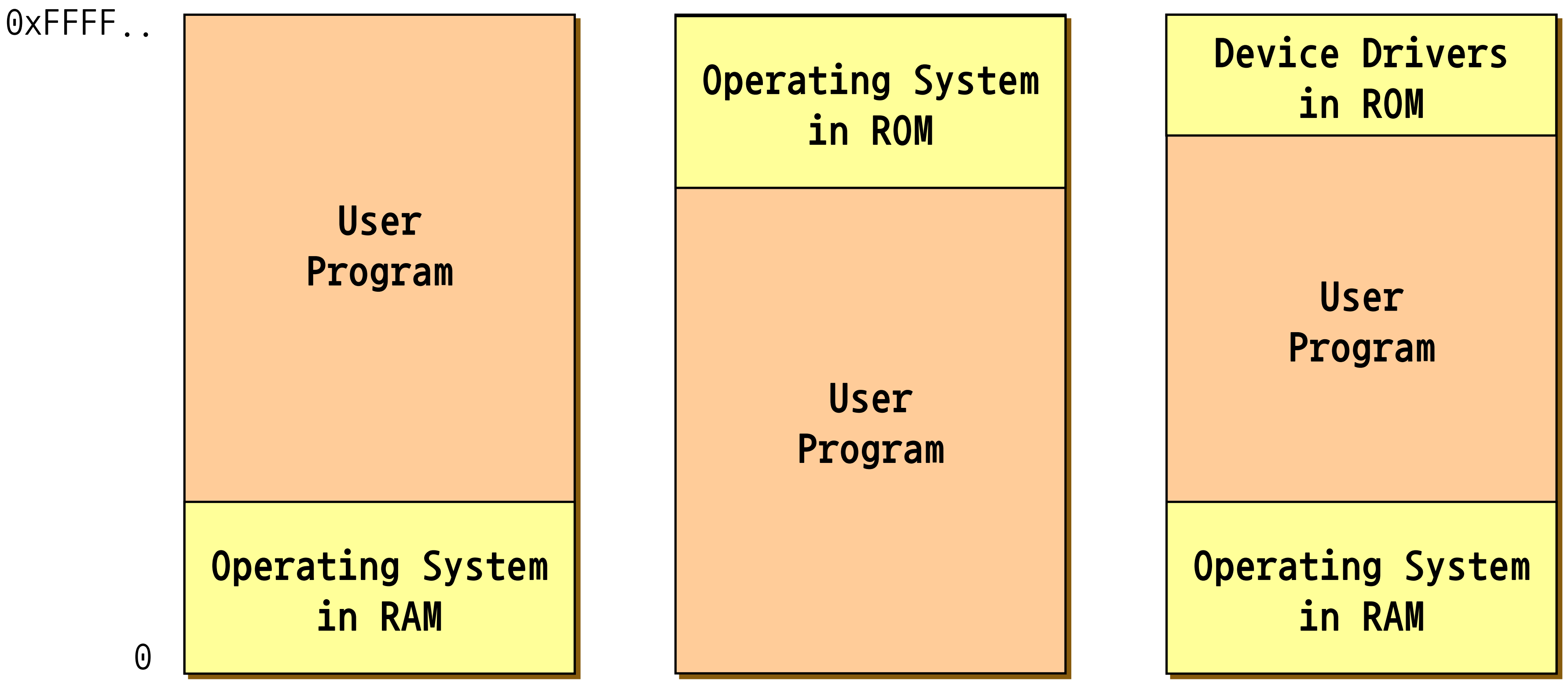
- To provide a convenient abstraction for programming
- To allocate scarce memory resources among competing processes
  - To maximize performance with minimal overhead
- To provide isolation between processes

Why is it so difficult?

Single/Batch Programming

An OS with one user process

- Programs use physical addresses directly
- OS loads job, runs it, unloads it

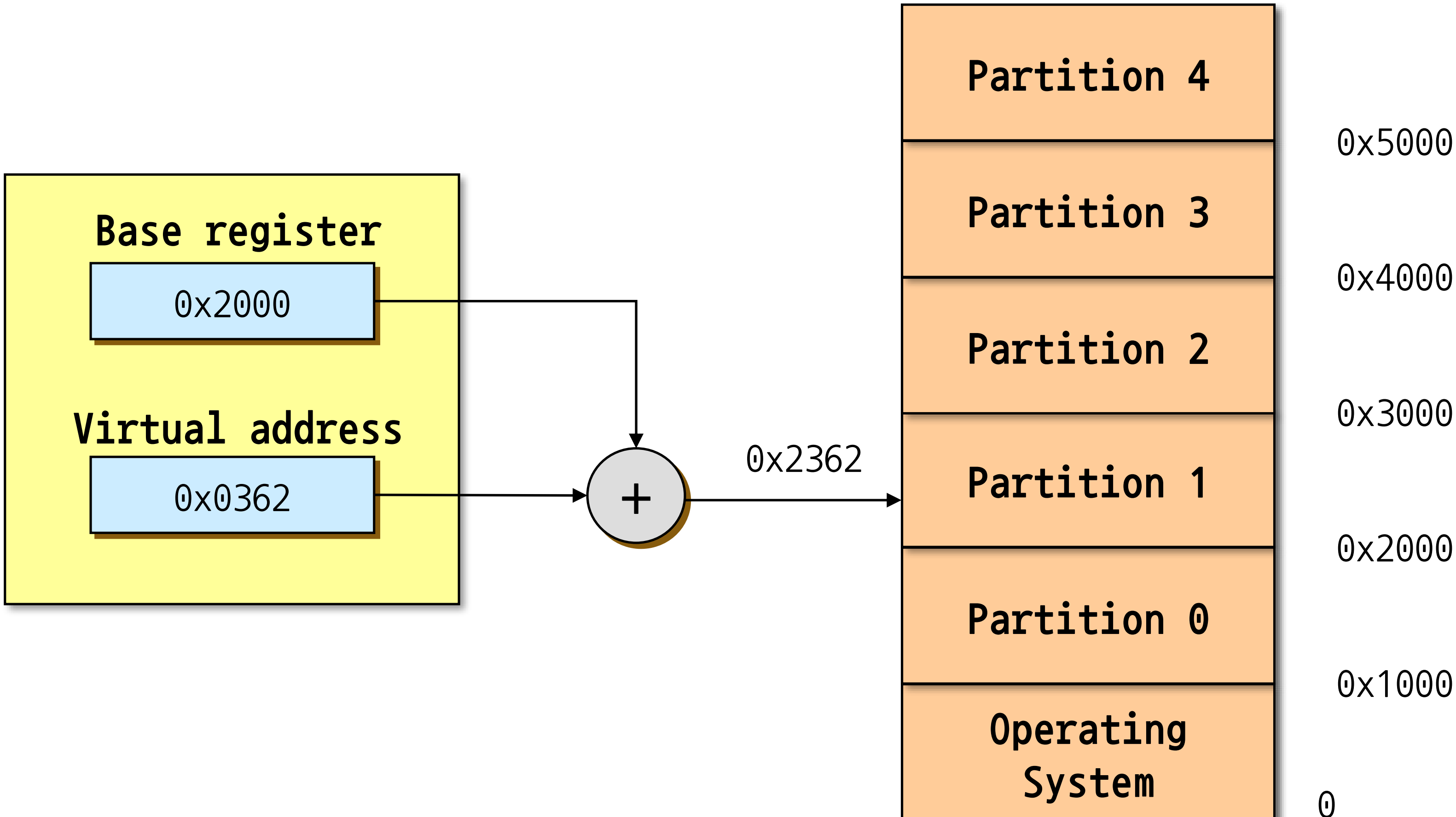


Multiprogramming

Multiprogramming

- Need multiple processes in memory at once
  - To overlap I/O and CPU of multiple jobs
  - Each process requires variable-sized and contiguous space
- Requirements
  - Protection: restrict which addresses and processes can use
  - Fast translation: memory lookups must be fast, in spite of protection scheme
  - Fast context switching: updating memory hardware (for protection and translation) should be quick

Fixed Partitions (1)



Fixed Partitions (2)

Physical memory is broken up into fixed partitions

- Size of each partition is the same and fixed
- The number of partitions = degree of multiprogramming
- Hardware requirements: base register
  - Physical address = virtual address + base register
  - Base register loaded by OS when it switches to a process

Advantages

- Easy to implement, fast context switch

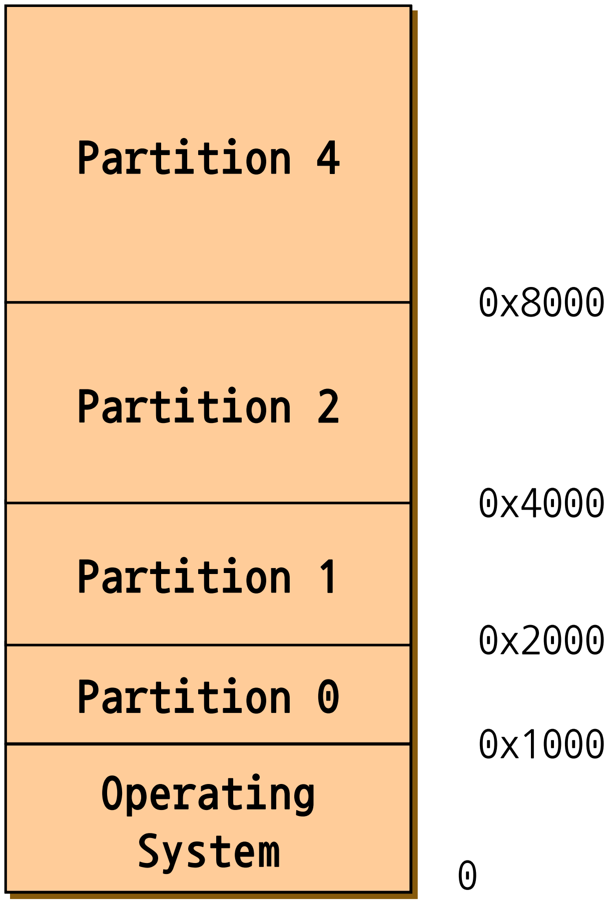
Problems

- Internal fragmentation: memory in a partition not used by a process is not available to other processes
- Partition size: one size does not fit all
  - Fragmentation vs. Fitting large programs

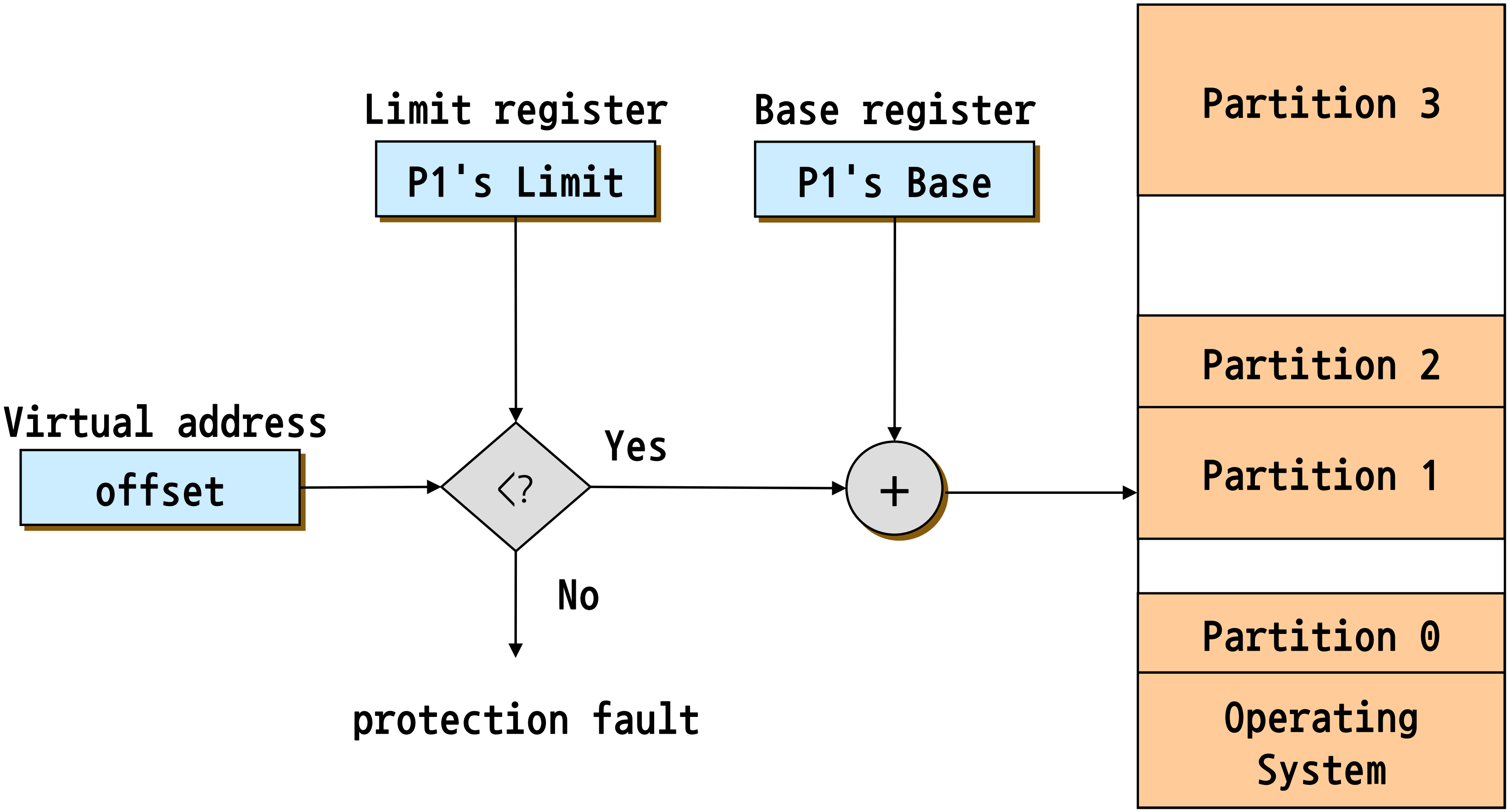
Fixed Partitions (3)

Improvement

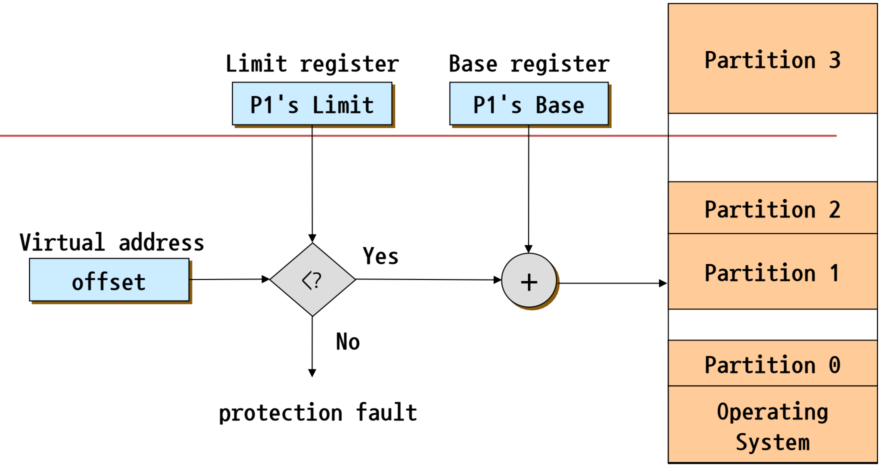
- Partition size need not be equal
- First fit allocation
  - Allocate to the closest job whose size fits in an empty partition
  - Need scanning
- Best fit allocation
  - Pick the largest job that fits in an empty partition
  - Need more scanning (more overhead)
- IBM OS/MFT (Multiprogramming with a Fixed number of Tasks)



Variable Partitions (1)



Variable Partitions (2)



Physical memory is broken up into variable-sized partitions

- IBM OS/MVT
- Hardware requirements: base register and limit register
  - Physical address = virtual address + base register
  - Base register loaded by OS when it switches to a process
- The role of limit register: protection
  - If (physical address > base + limit), then raise a protection fault

Allocation strategies

- First fit: Allocate the first hole that is big enough
- Best fit: Allocate the smallest hole that is big enough
- Worst fit: Allocate the largest hole

Variable Partitions (3)

Advantages

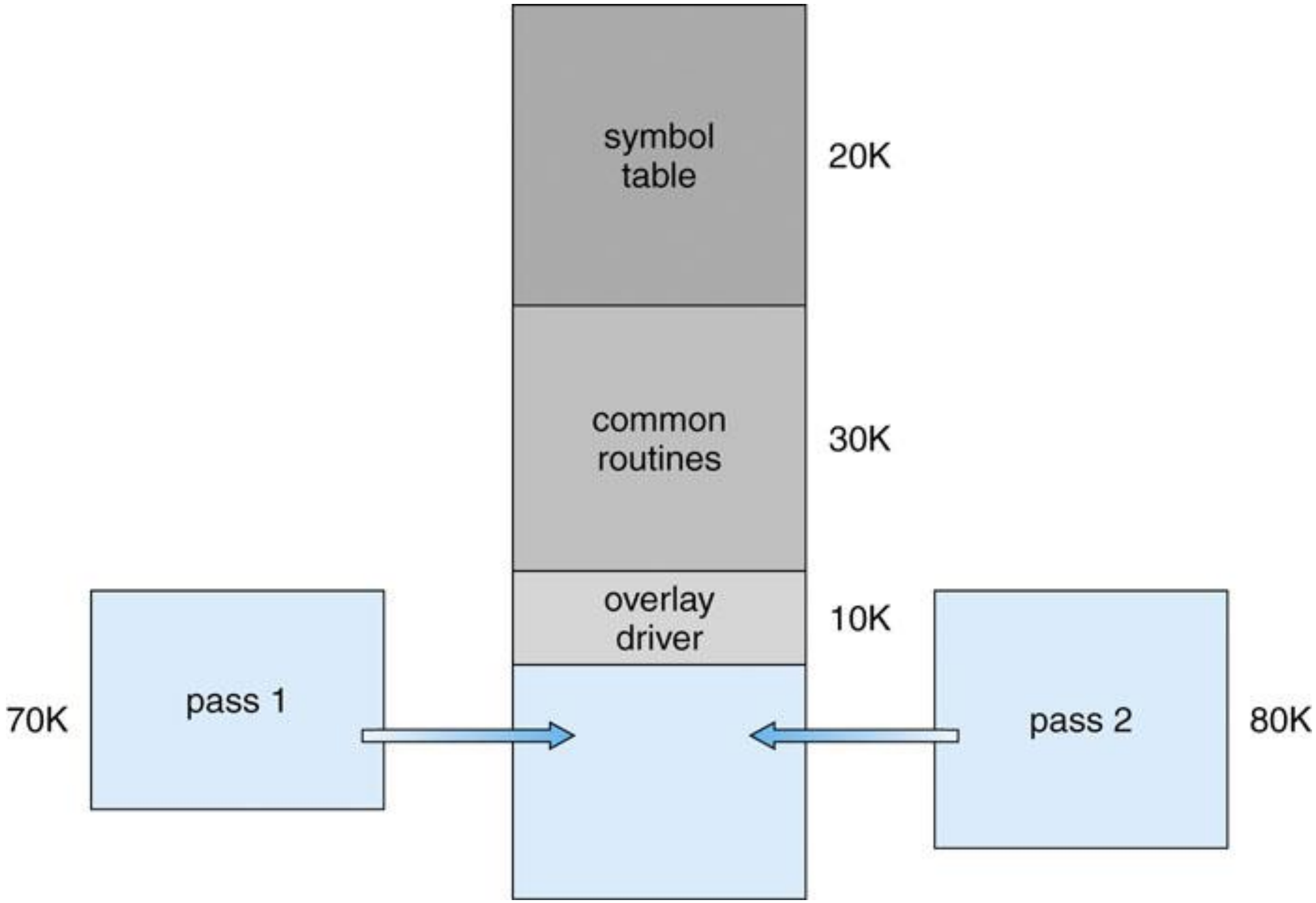
- No internal fragmentation
  - Simply allocate partition size to be just big enough for process
- But, if we break the physical memory into fixed-sized blocks and allocate memory in unit of block sizes (in order to reduce bookkeeping), we have internal fragmentation

Problems

- External fragmentation
  - As we load and unload jobs, holes are left scattered throughout physical memory
- Solutions to external fragmentation:
  - Compaction
  - Paging and segmentation

Overlays (1)

Overlays for a two-pass assembler



Overlays (2)

Overlays

- Keep in memory only those instructions and data that are needed at any given time
- Normally implemented by user

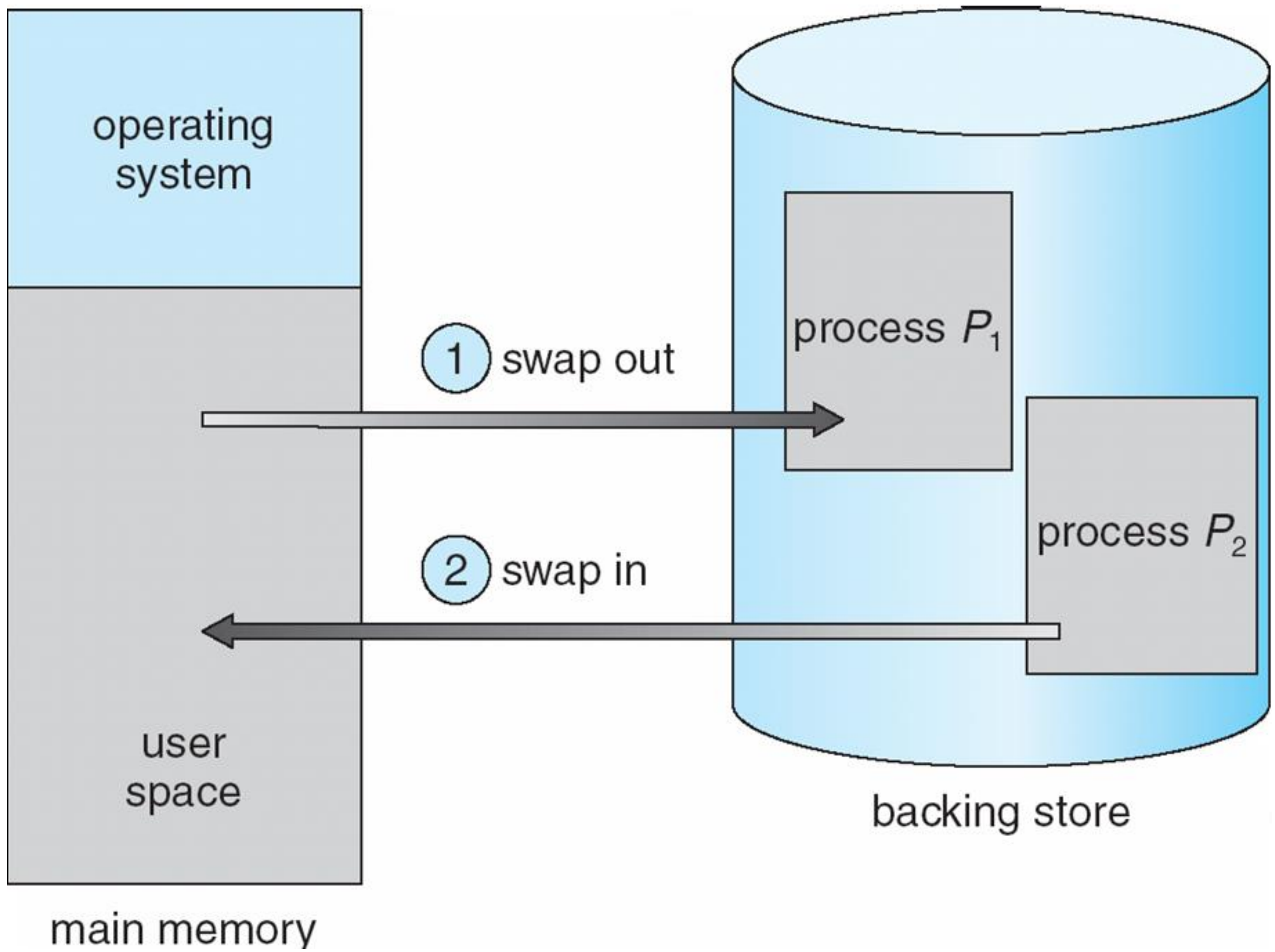
Advantages

- Needed when a process is larger than the amount of memory allocated to it
- No special support needed from operating system

Problems

- Complex
  - Programming design of overlay structure

Swapping (1)



operating
system
1 swap out
2 swap in
user
space
main memory
process P_1
process P_2
backing store

Swapping (2)

Swapping

- Temporarily swapping out of memory to a backing store
- Bringing back into memory later for continued execution
- Backing store
  - Fast disk
  - Large enough to accommodate copies of all memory images
  - Must provide direct access to these memory images

Problems

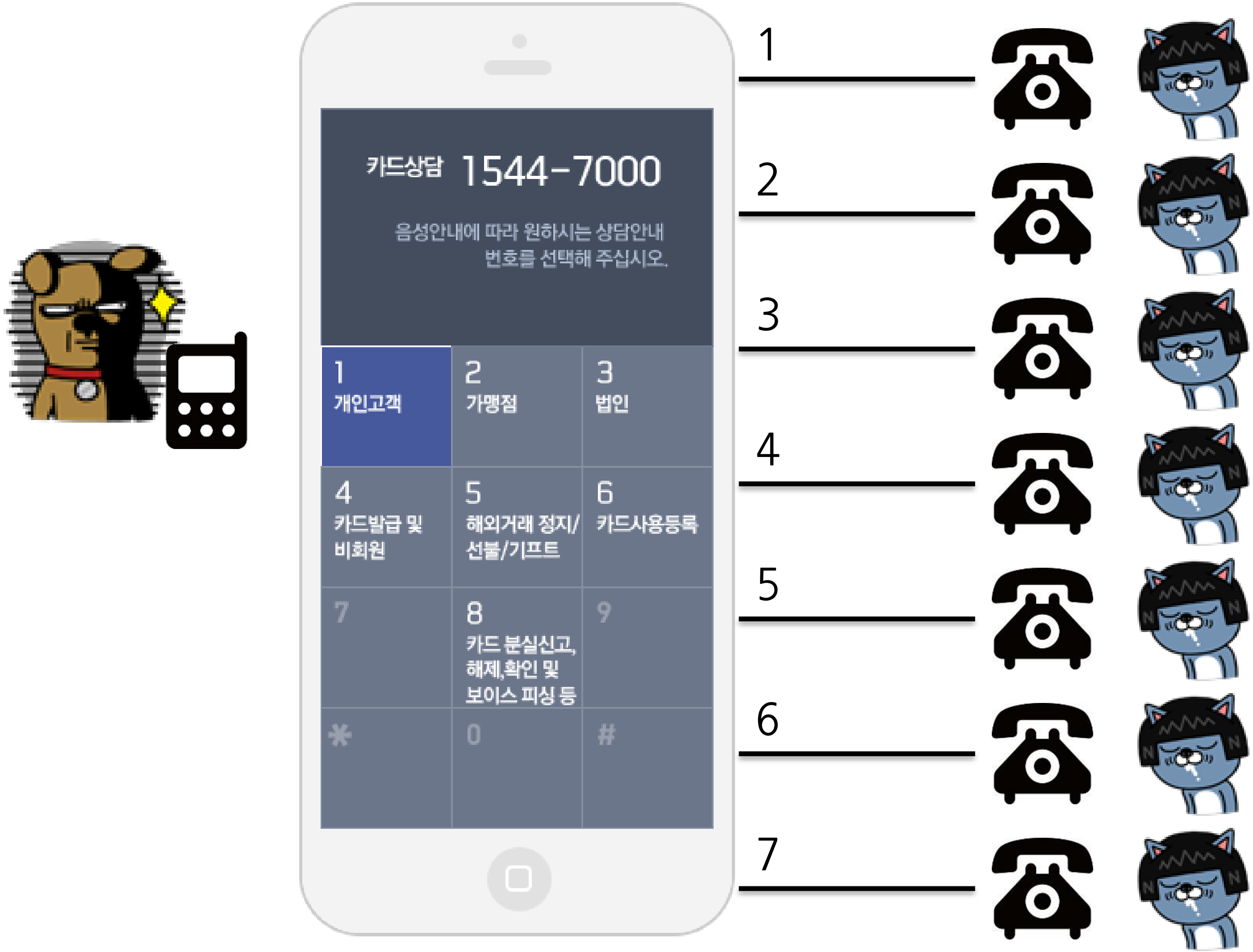
- Major part of swap time is transfer time
  - Directly proportional to the amount of memory swapped
- Swapping a process with a pending I/O ?
  - Do not swap a process with pending I/O
- Modern OS uses modified swapping mechanisms (demand paging) with virtual memory

Virtual Memory (1)

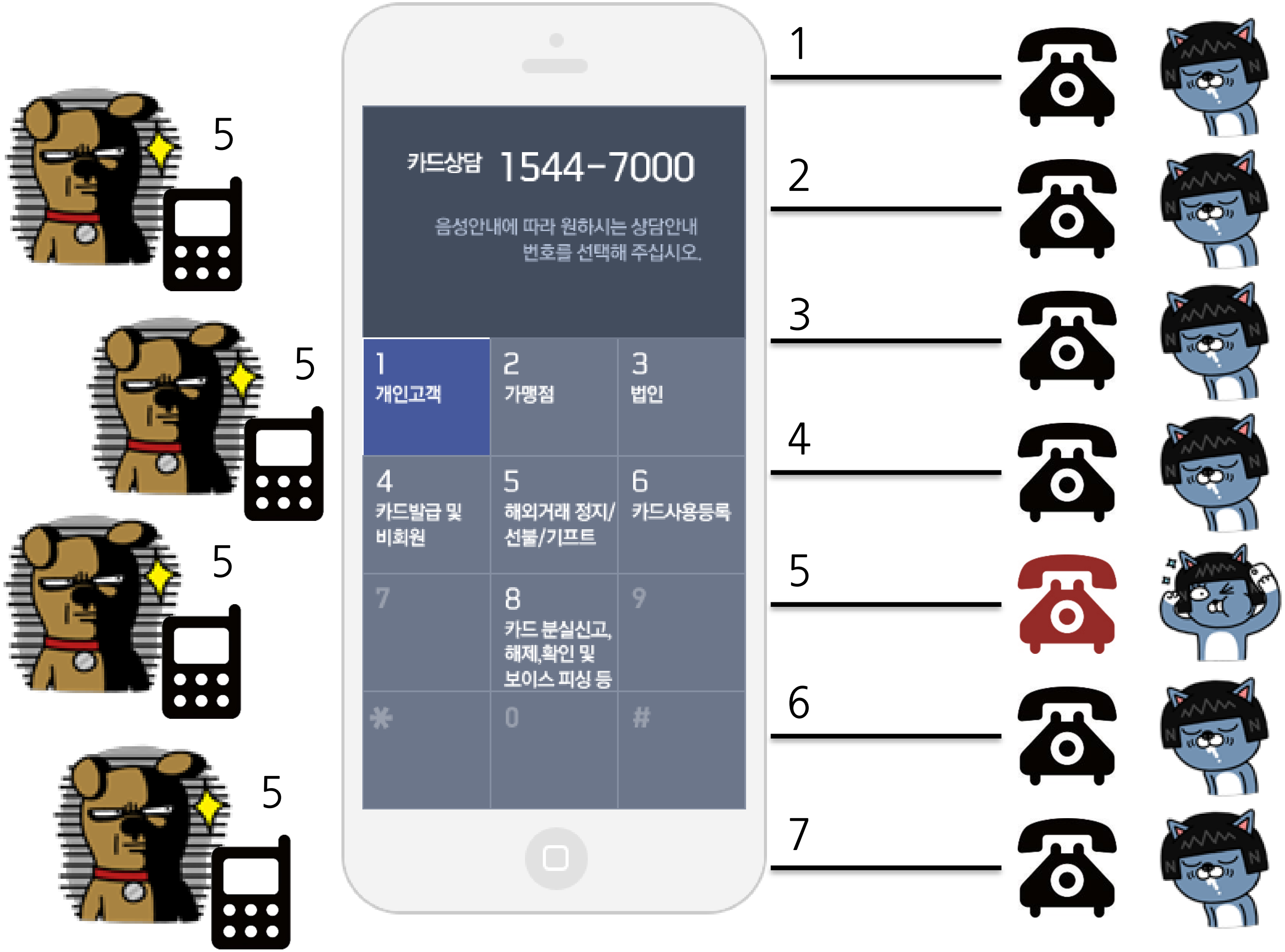
Example

```
#include <stdio.h>

int n = 0;

int main ()
{
     printf ( “ &n = 0x%08x\n ” , &n);
}

% ./a.out
&n = 0x08049508
% ./a.out
&n = 0x08049508
```

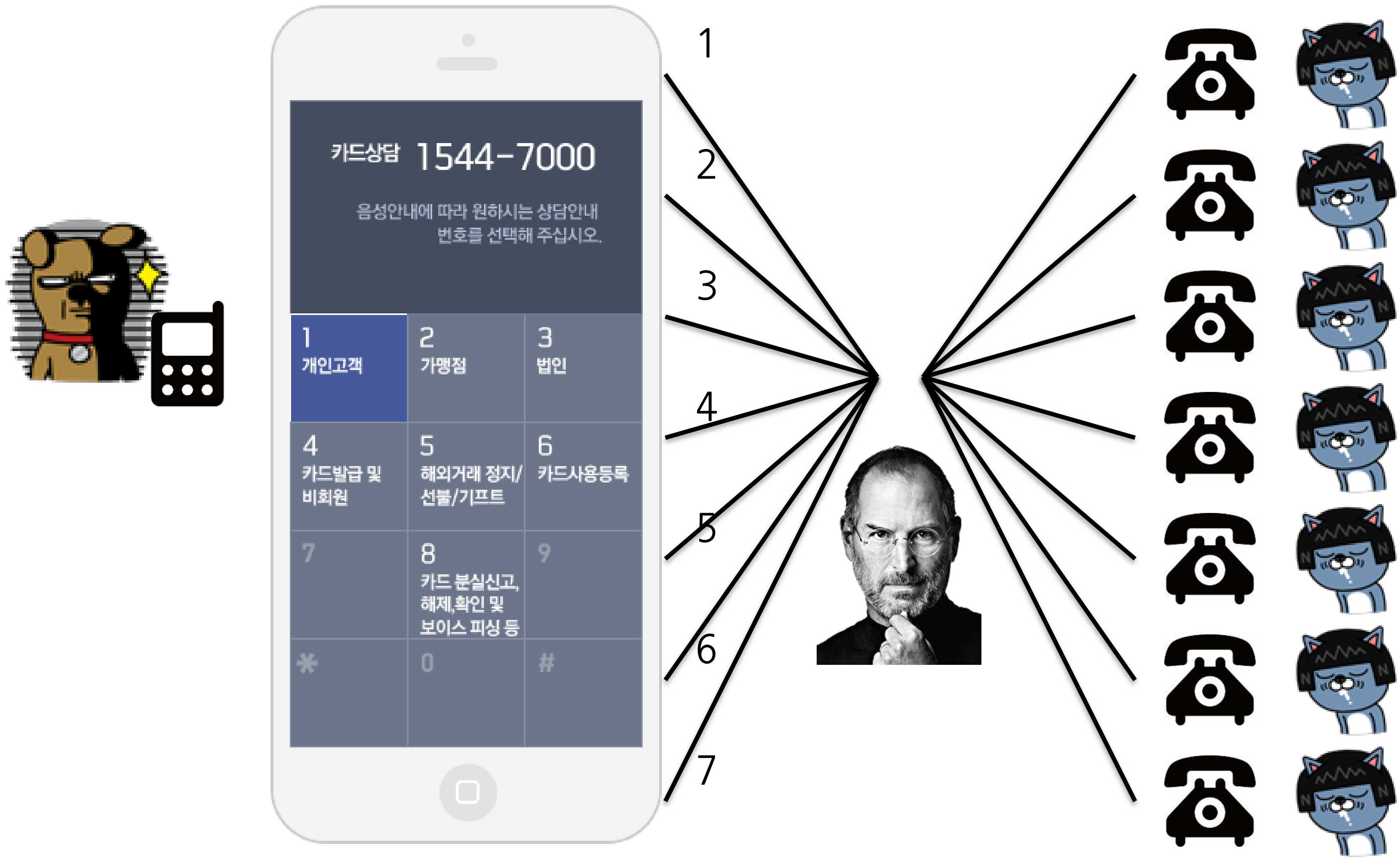
Virtual Memory Concept



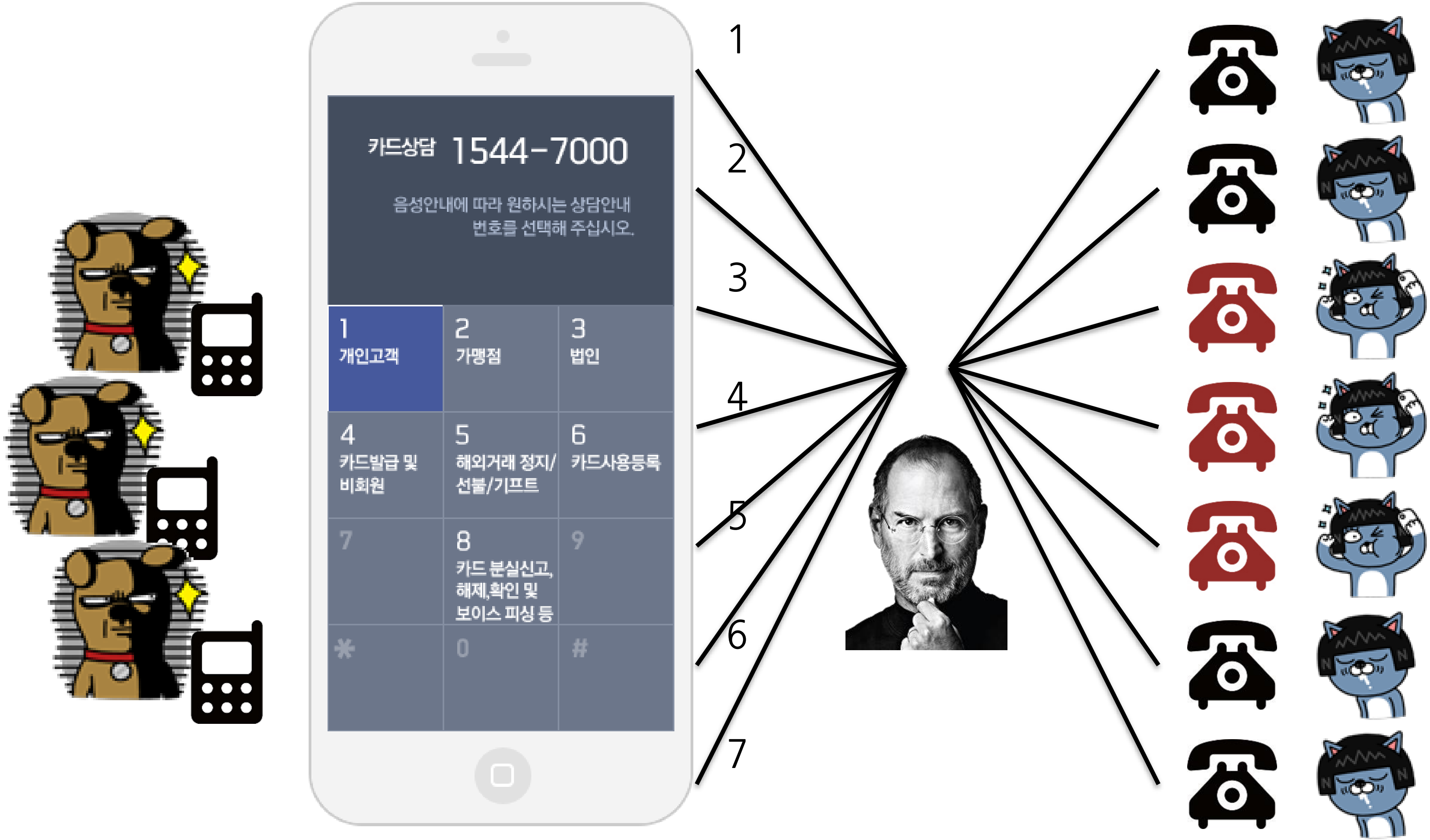
Virtual Memory Concept



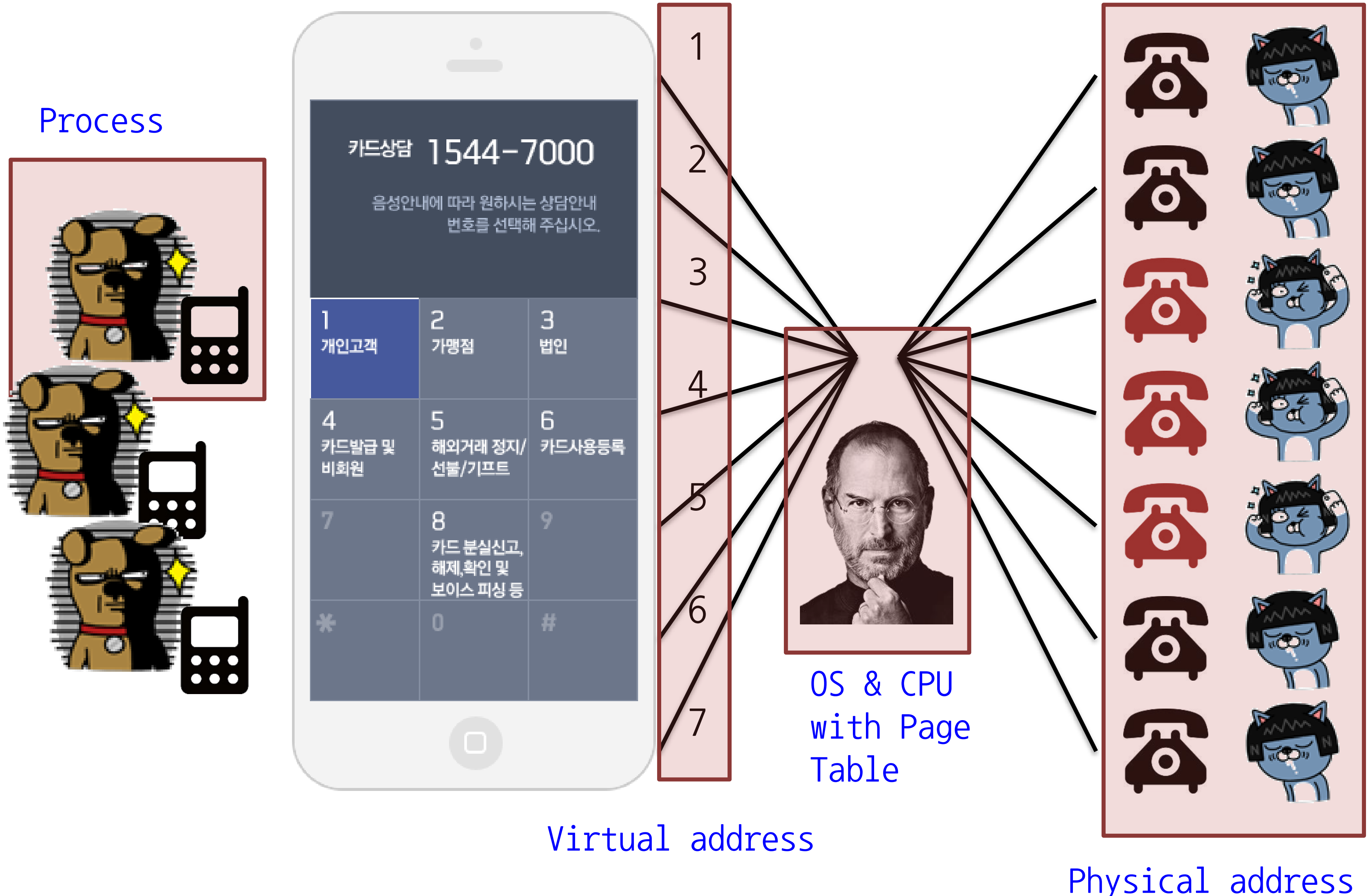
Virtual Memory Concept



Virtual Memory Concept



Virtual Memory Concept



Virtual Memory (2)

Virtual Memory (VM)

- Use virtual addresses for memory references
  - Large and contiguous
- CPU & OS perform address translation at run time
  - From a virtual address to the corresponding physical address
- Physical memory is dynamically allocated or released on demand
  - Programs execute without requiring their entire address space to be resident in physical memory
  - Lazy loading
- Virtual addresses are private to each process
  - Each process has its own isolated virtual address space
  - One process cannot name addresses visible to others

Virtual Memory (3)

Virtual addresses

- To make it easier to manage memory of multiple processes, make processes use virtual addresses (logical addresses)
- Virtual addresses are independent of the actual physical location of data referenced
- OS determines location of data in physical memory

Memory access procedures

- Instructions executed by the CPU issue virtual addresses
- Virtual addresses are translated by hardware into physical addresses (with help from OS)
- The set of virtual addresses that can be used by a process is its virtual address space

There are many ways to translate virtual addresses into physical addresses

Virtual Memory (4)

Advantages

- Separates user's logical memory from physical memory
  - Abstracts main memory into an extremely large, uniform array of storage
  - Frees programmers from the concerns of memory-storage limitations
- Allows the execution of processes that may not be completely in memory
  - Programs can be larger than physical memory
  - More programs could be run at the same time
  - Less I/O would be needed
    - to load or swap each user program into memory
- Allows processes to easily share files and address spaces
- Efficient for protection and process creation

Virtual Memory (5)

Disadvantages

- Performance!!!
  - In terms of time and space

Implementation

- Paging
- Segmentation