운영체제의 기초:
Introduction to OS

2023년 3월 2, 7, 9일

홍 성 수
sshong@redwood.snu.ac.kr
SNU RTOSLab 지도교수
서울대학교 전기정보공학부 교수

Why Study OS?

- OS is an exciting field of study
  - Brings together many areas in Computer Science
    - Data structures, algorithms
    - Programming languages, compilers
    - Computer hardware, architecture
- Course goals
  - Learn theory and practice behind major OS features
  - Understand interworkings of OS internals
  - Apply your knowledge to develop better software or design new OS

Agenda

I. Evolution of OS
II. Functions of OS

I. Evolution of OS – Phase I
Up to Batch Monitor

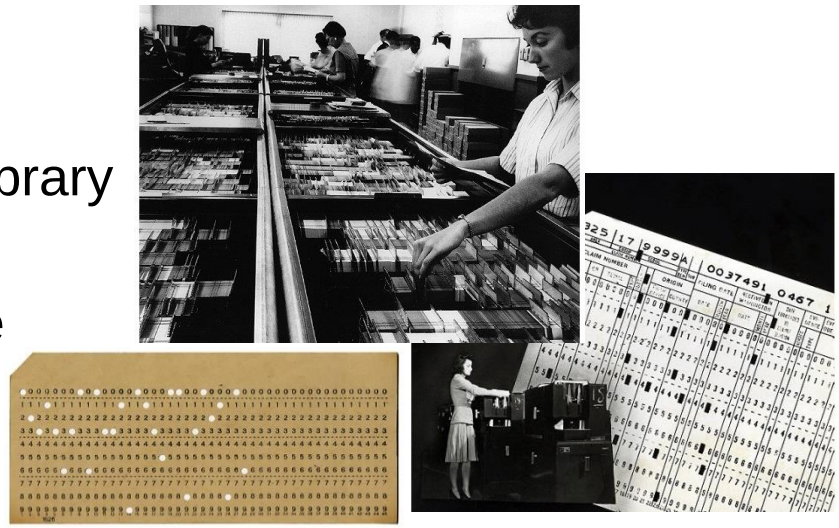
Three Phases of OS History

- Why study OS evolution?
  - Defining the term "*Operating System*" is difficult
    - Discipline arose historically from a set of problems

- Three Phases of OS History
  - Phase I: early '50s – mid '60s
  - Phase II: mid '60s – mid '90s
  - Phase III: mid '90s – present

Phase I (1)

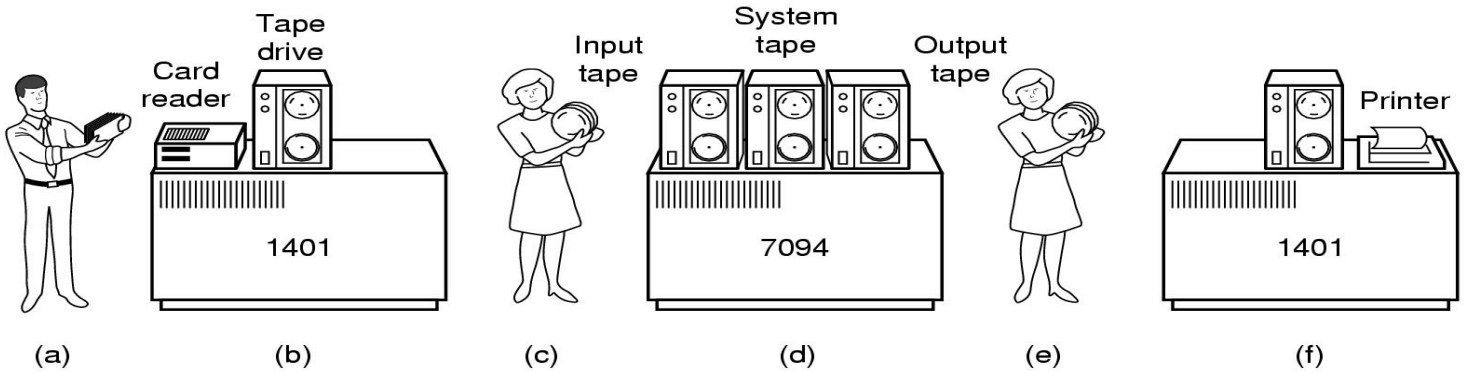
- Key observation
  - Hardware expensive, humans cheap
- Goal
  - Make efficient use of the hardware
- Phase 1-1: *Operator as OS*
  - OS was a shared subroutine library
    - Card decks in cabinet
  - Single user working at console
  - Debugging done interactively
  - Slow job-to-job transition



Source: http://www.computerhistory.org/

Phase I (2)

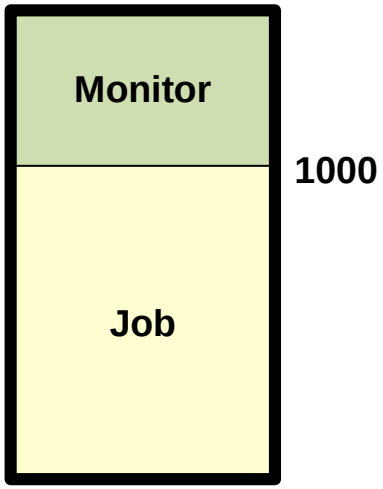
- Phase 1-2: (1) *Simple batch monitor*
  - OS loaded and ran a user job and took dump
    - *Simultaneous Peripheral Operations Online* (*SPOOL*)
      - I/O machine (IBM 1401) read in "a *batch of jobs" onto tape*
      - Main machine (IBM 7094) loaded "*a job from the batch" on tape*, did computing and took dump back to tape
      - I/O machine printed output from tape



Source: Tanenbaum and Woodhull, Operating Systems Design and Implementation, 2006

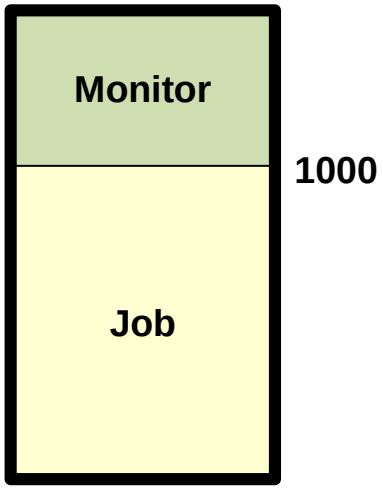
Phase I (3)

- Phase 1-2: (1) *Simple batch monitor* (cont'd)
  - Debugging done offline
  - Resolved limitations
    - CPU working with a faster I/O device than a card reader
    - Faster job-to-job transition within a batch
  - Unresolved issue
    - No overlap between I/O and computation



Phase I (4)

- Phase 1-3: (2) *Batch monitor*
  - Jobs *spooled* on “disk” or “drum”
    - Read jobs from cards to disk, loaded one into memory, and queued output to disk for printing
    - No need for costly I/O machines (advanced *SPOOLing*)
  - “Buffering” and “interrupt handling” added to OS
    - Overlap of computation with asynchronous I/O
  - Still single job, so utilization often bad

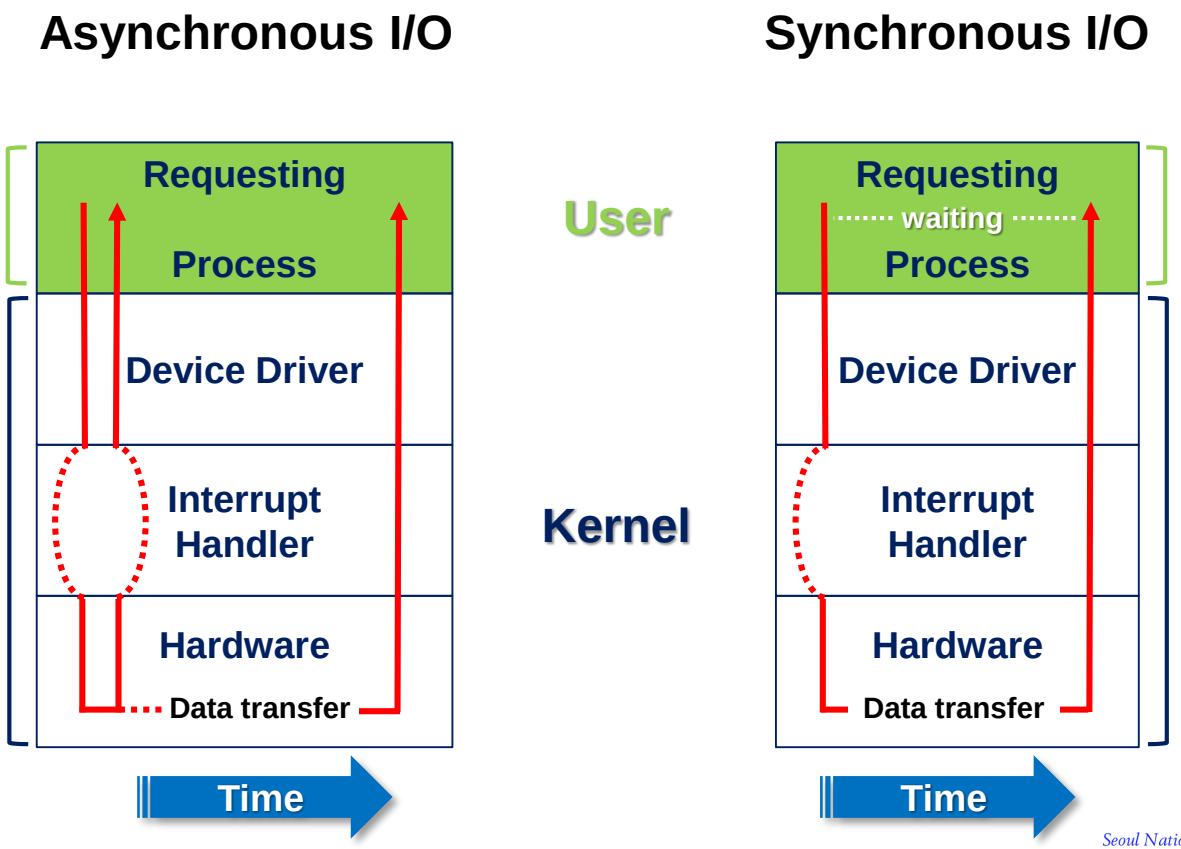


Phase I (4)

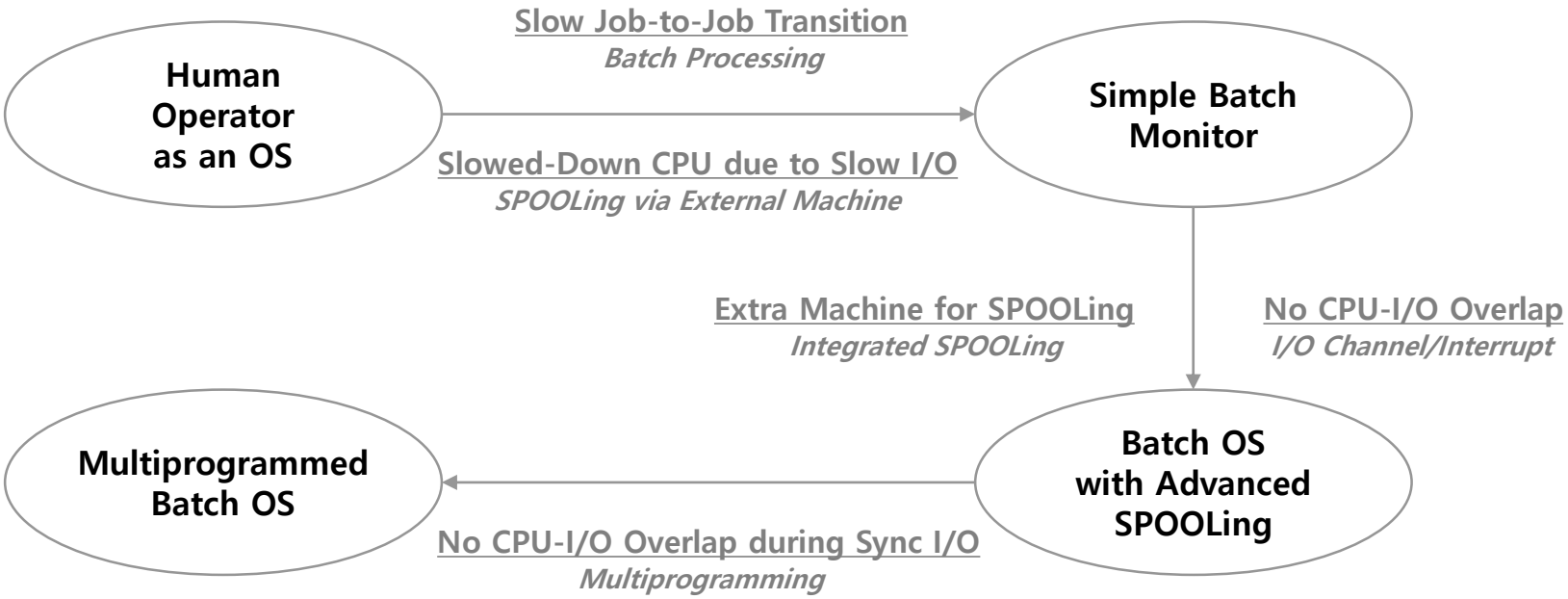
- Aside: Two types of I/O methods
  - Asynchronous I/O
    - After I/O starts, control returns to user program without waiting for I/O completion
  - Synchronous I/O
    - After I/O starts, control returns to user program only upon I/O completion
      - Wait instruction idles the CPU until the next interrupt
      - Wait loop
    - At most one I/O request is outstanding at a time

Phase I (5)

- Aside: Two types of I/O methods (cont'd)



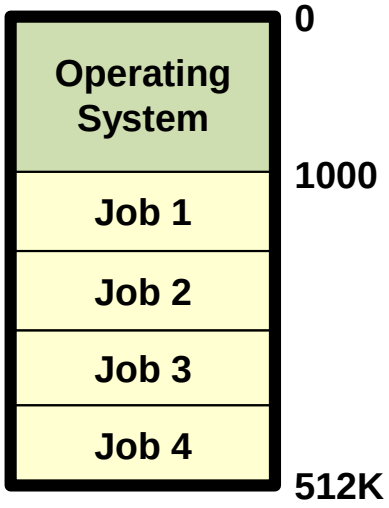
Evolution of Early-Day OS



I. Evolution of OS – Phase I
Multiprogrammed Batch Monitor

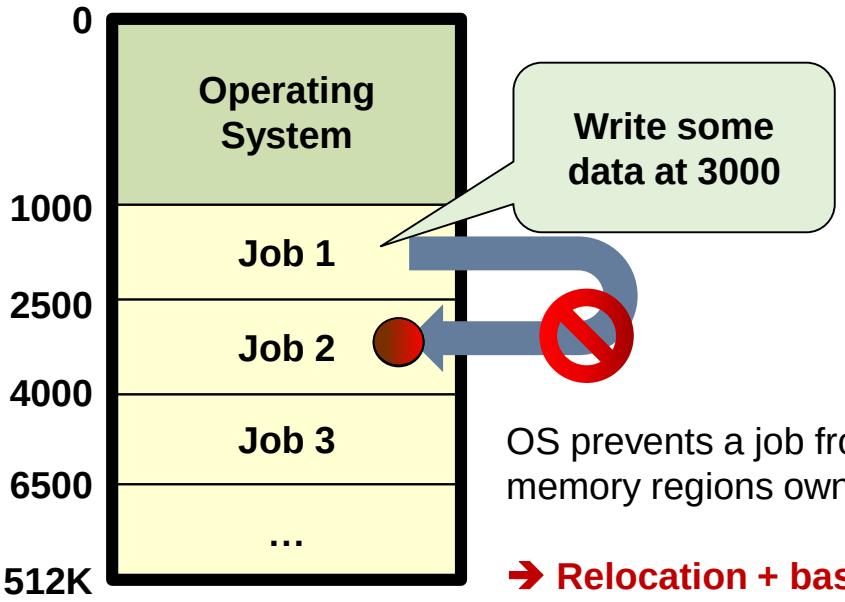
Phase I (6)

- Phase 1-4: (3) *Multiprogrammed batch monitor*
  - Several users shared the system
    - Degree of multiprogramming ≥ 1
  - OS became a focus of study
    - Memory protection and relocation added to OS
    - Higher utilization because of multiple jobs
    - Concurrent programming became necessary



Phase I (7)

- Memory protection

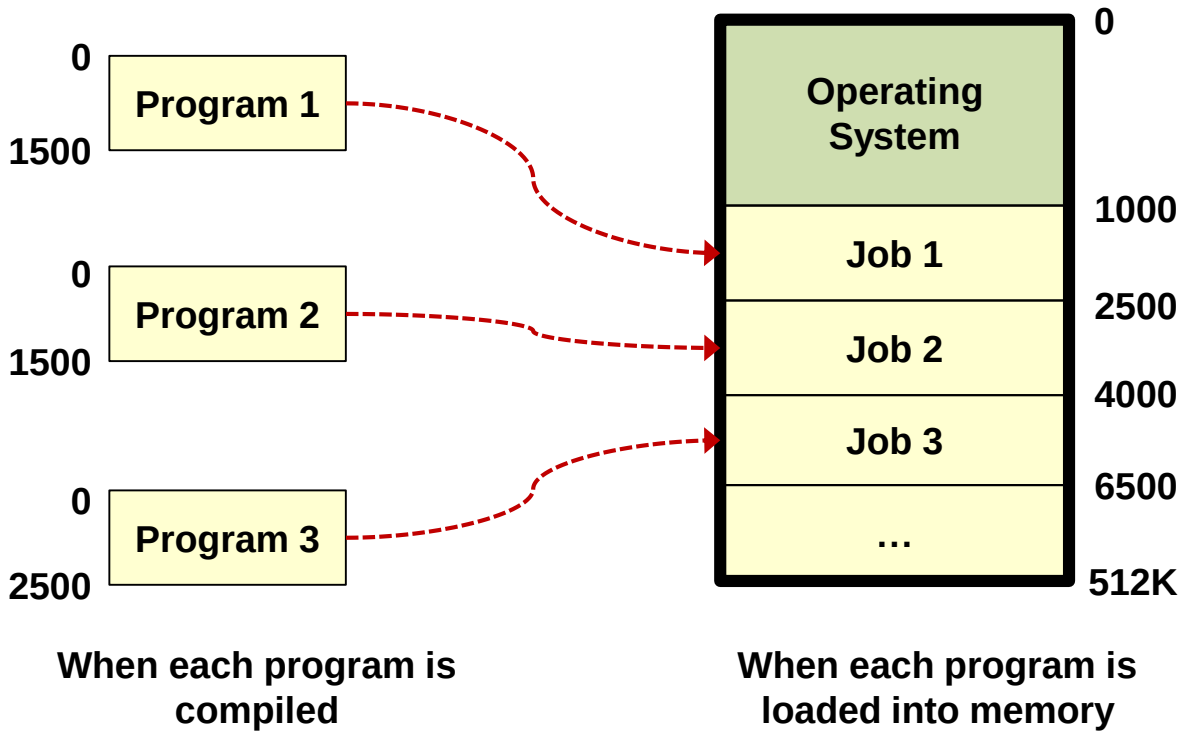


OS prevents a job from accessing memory regions owned by other jobs

➔ **Relocation + base/bound registers**

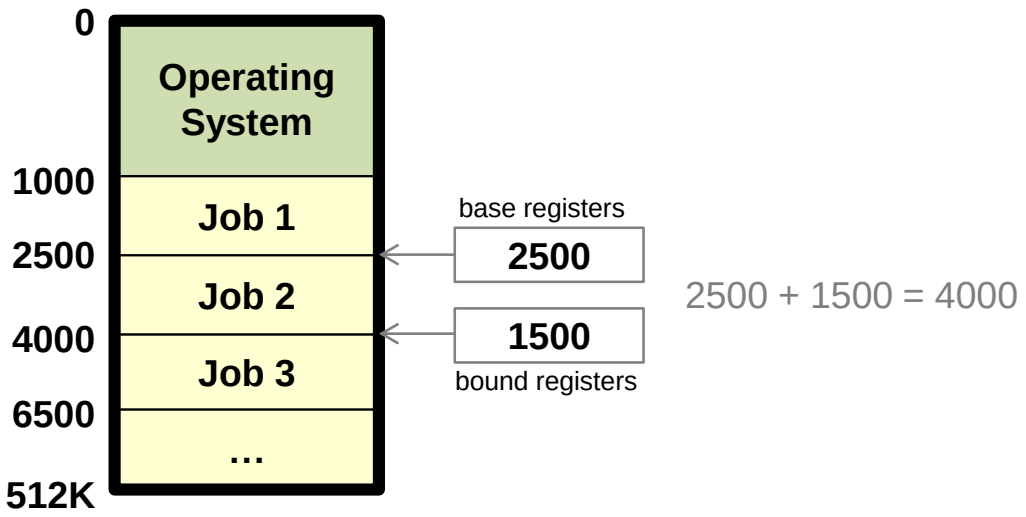
Phase I (8)

- Relocation



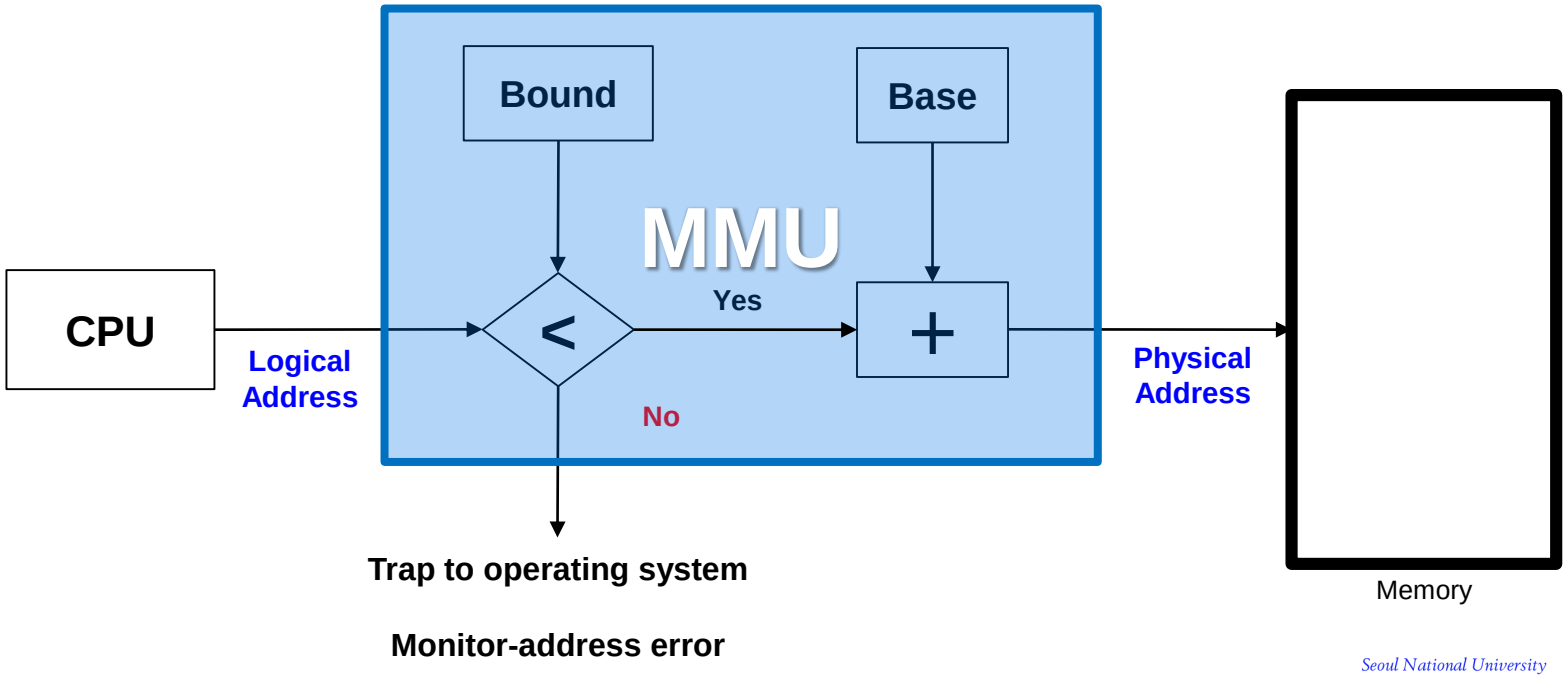
Phase I (9)

- Base/Bound registers
  - Primitive form of MMU (memory management unit)



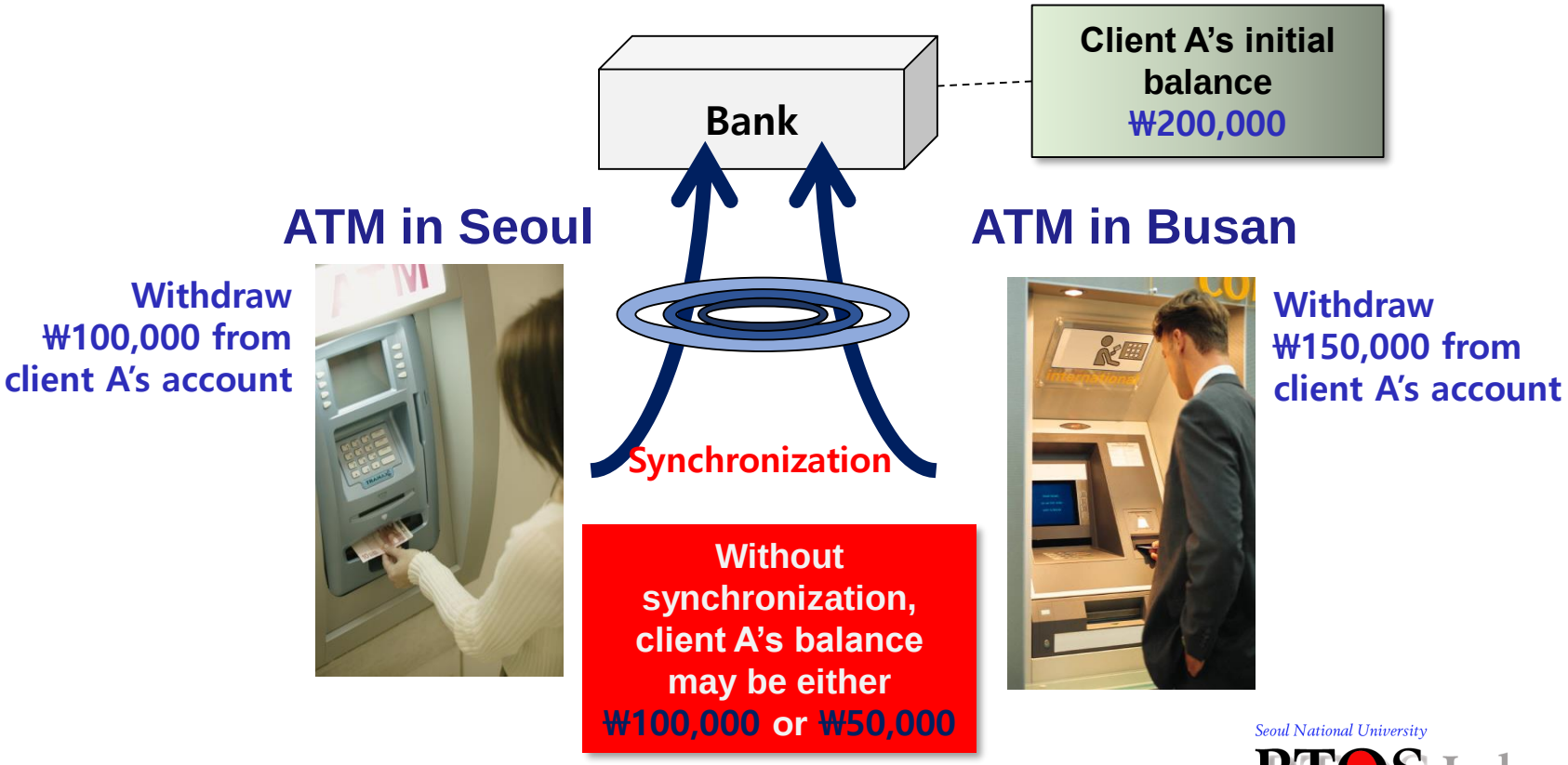
Phase I (10)

- Base/Bound registers (cont'd)
  - Primitive form of MMU (memory management unit)



Phase I (11)

- Concurrency and synchronization



I. Evolution of OS – Phase II

Phase II (1)

- Key observation
  - Hardware cheap, humans expensive
- Goal
  - Make efficient use of people's time
- Phase 2-1: *Interactive time-sharing OS*
  - Terminals were cheap
  - Users interacted with the system again
  - Fancy filing systems added to OS
  - Response time and protection became important

Phase II (2)

- Phase 2-2: *PC OS*
  - Computers were cheap
    - Computer in every terminal
  - OS becomes a subroutine library again

- Phase 2-3: *OS with Internet Access*
  - Allowed different machines to share resources easily
    - Remote procedure calls (RPC)
    - Network file system (NFS)

I. Evolution of OS – Phase III

Phase III (1)

- Key observation
  - Connectivity matters; things get connected
- Goal
  - Provide connected multimedia services for users
- Phase 3-1: *OS with built-in Internet Access*
  - Internet protocols added to PC OS
  - Internet programming is important (Web, CGI, Java, ...)
  - Multitasking became important again

Phase III (2)

- Phase 3-2: *Sophisticated PC OS*
  - Computers are extremely cheap
    - Even PC has sophisticated architecture
  - OS became complex again
- Phase 3-3: *OS with Multimedia Support*
  - Demands lots of computer and network resources
  - Human perception became the center of the universe
    - QoS (Quality of Service), RTOS (Real-Time OS)
  - Home appliances and computers got merged

Phase III (3)

- Phase 3-4: *OS as Commodity*
  - Common OS used in desktop, mobile, cloud systems
  - Multicore support added to OS
  - Virtualization
  - OS became software platform
    - Android, webOS, …

II. Functions of OS

OS Characteristics

- Characteristics of current OS
  - Large
    - 10M's of lines of code, 100-1000 man-years of work
  - Complex
    - Asynchronous behaviors
    - Hardware idiosyncrasies
    - Conflicting needs of different users and performance goals
  - Poorly understood
    - The system outlives any of its builders
    - Too complex to totally debug – often unreliable
    - Behavior is hard to predict
    - Tuning is done by guessing

Functions of OS

1

Coordinator

- Allow things to work together efficiently and fairly

2

Illusion Generator

- Exports cleaner, higher level interface to hardware

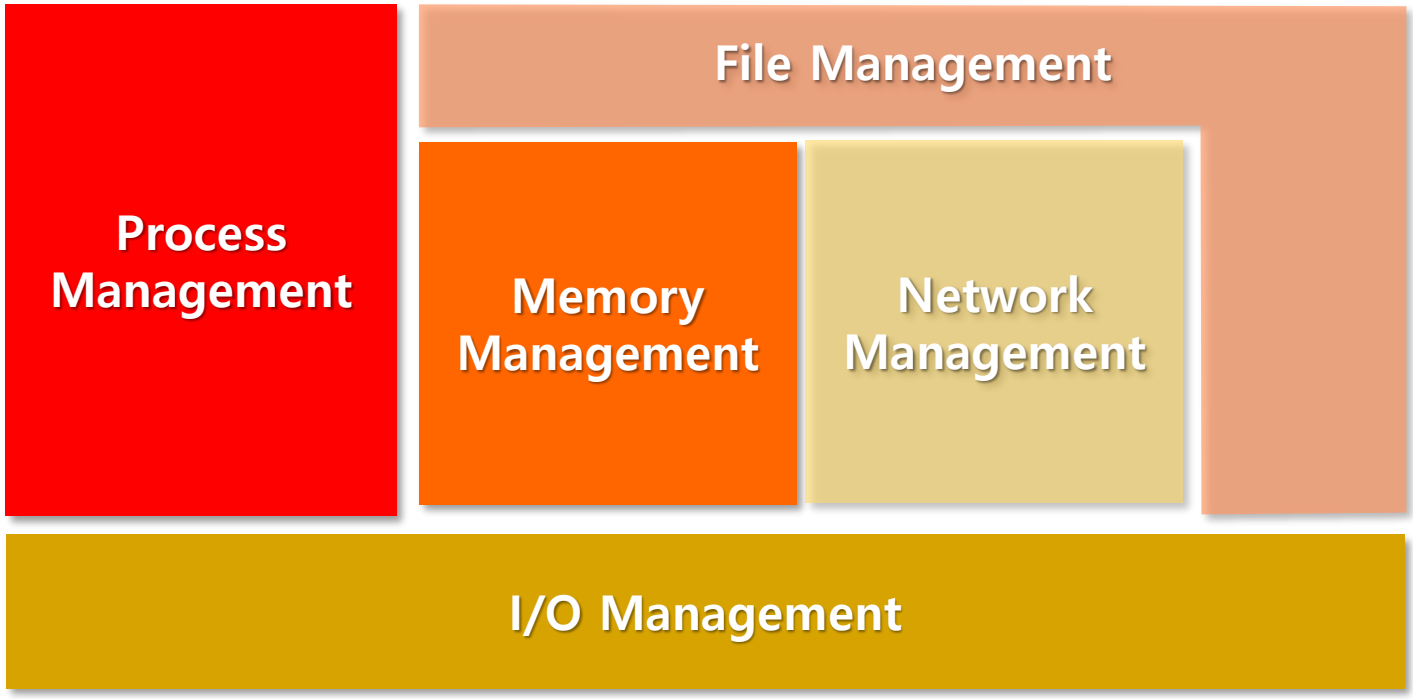
3

Standard Library

- Provide standard facilities that everyone needs

OS as Coordinator (1)

- Make many things work well together



OS as Coordinator (2)

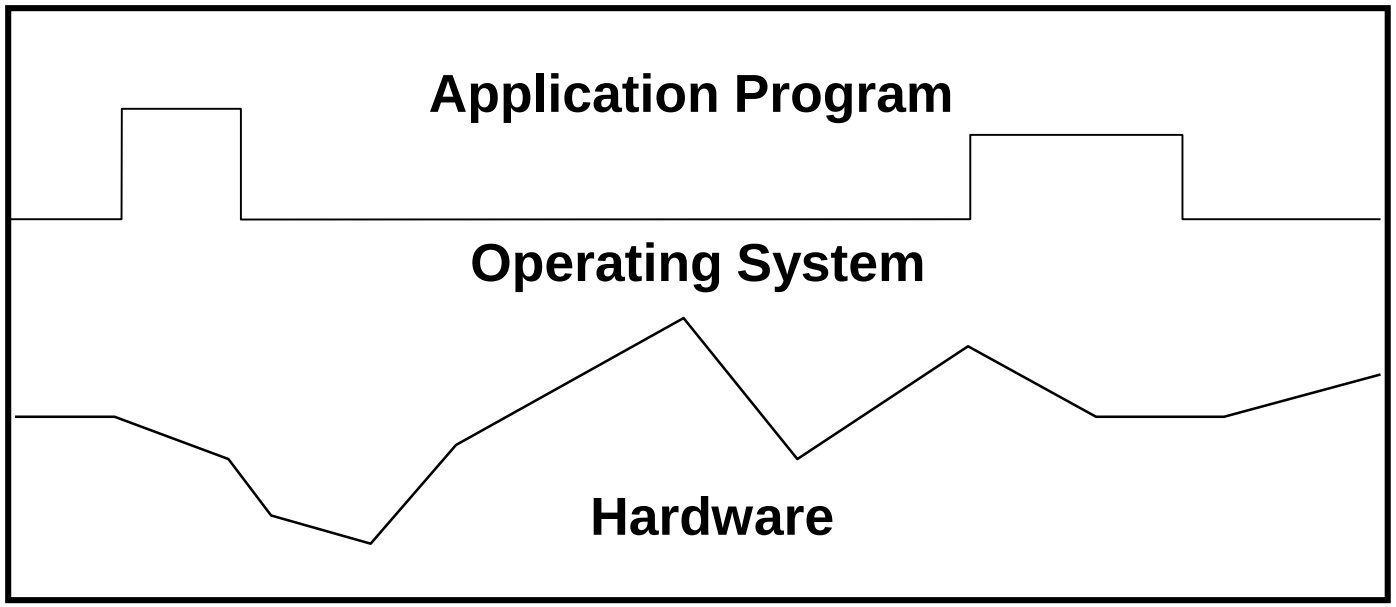
- Make many things work well together (cont'd)
  - Concurrency: Notion of process
    - Several users working at the same time
    - One user doing many things at the same time
  - I/O devices: I/O devices run concurrently with the CPU
    - Devices interrupts CPU when done
    - Interrupt processing complicates the OS
  - Memory: Each process needs some memory to execute
    - OS must coordinate the memory usage
    - Swap information between memory and disk

OS as Coordinator (3)

- Make many things work well together (cont'd)
  - Files: Each user owns a collection of files
    - OS must coordinate how space is allocated
    - Control shared accesses to files
  - Network: Allow groups of computers to work together

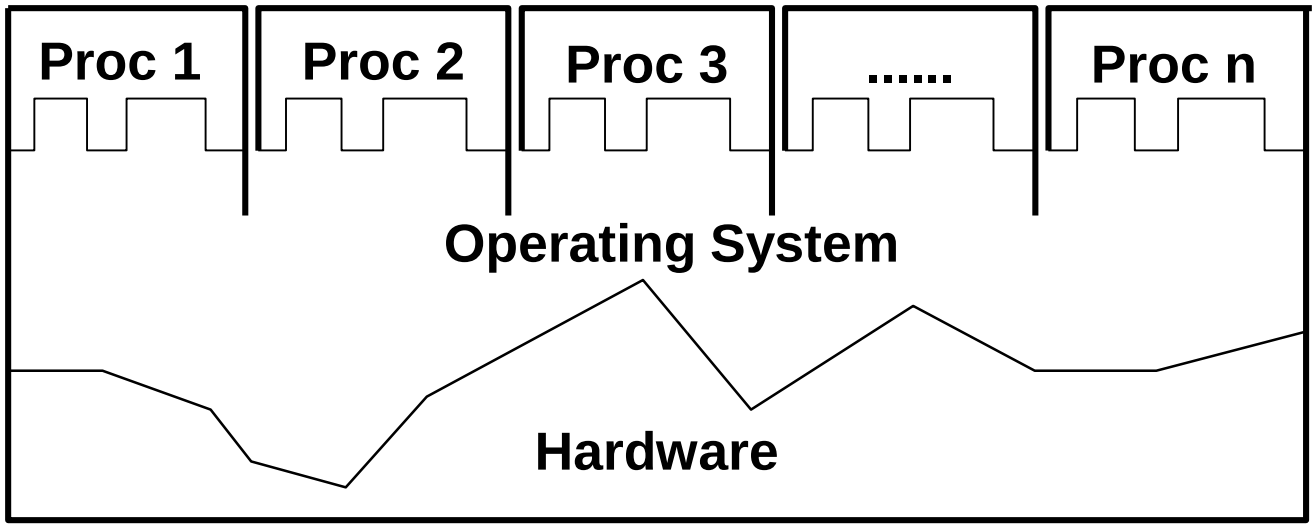
OS as Illusion Generator (1)

- OS presents an illusion: “Cleaner abstraction”



OS as Illusion Generator (2)

- OS presents an illusion: “Multiple processors”



OS as Illusion Generator (3)

- Examples that work
  - Timesharing, virtual memory
- Sometimes the illusions fail
  - You can’t fake what you don’t get
  - Thrashing