Principles of Operating Systems
CS 446/646

1. Introduction to Operating Systems

a. Role of an O/S

b. **O/S History and Features**
   - ✓ Serial processing
   - ✓ Simple batch systems
   - ✓ Multiprogrammed batch systems
   - ✓ Time-sharing systems
   - ✓ Personal computers

c. Types of O/S

d. Major O/S Components

e. System Calls

f. O/S Software Architecture

g. Examples of O/S

1.b Operating System History and Features

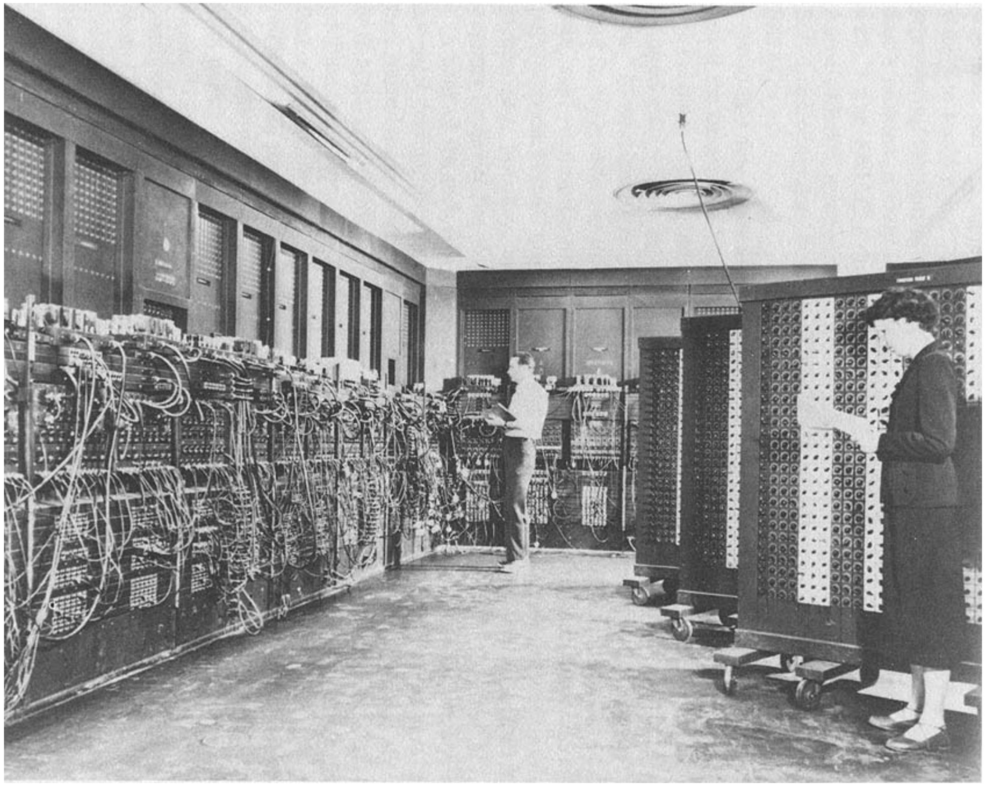
Note: This section is **not** a history of computer models or a year-by-year chronology of the origins of specific operating systems, like UNIX or Windows.

Our purpose is to highlight the **generic** features of operating systems through major periods in their evolution.

1.b Operating System History and Features

Serial processing

- **First generation: 1945–55**
  - ✓ room full of armoires: mechanical relays, then vacuum tubes



http://ftp.arl.mil/ftp/historic-computers/

The ENIAC (Electronic Numerical Integrator And Computer)

1.b Operating System History and Features

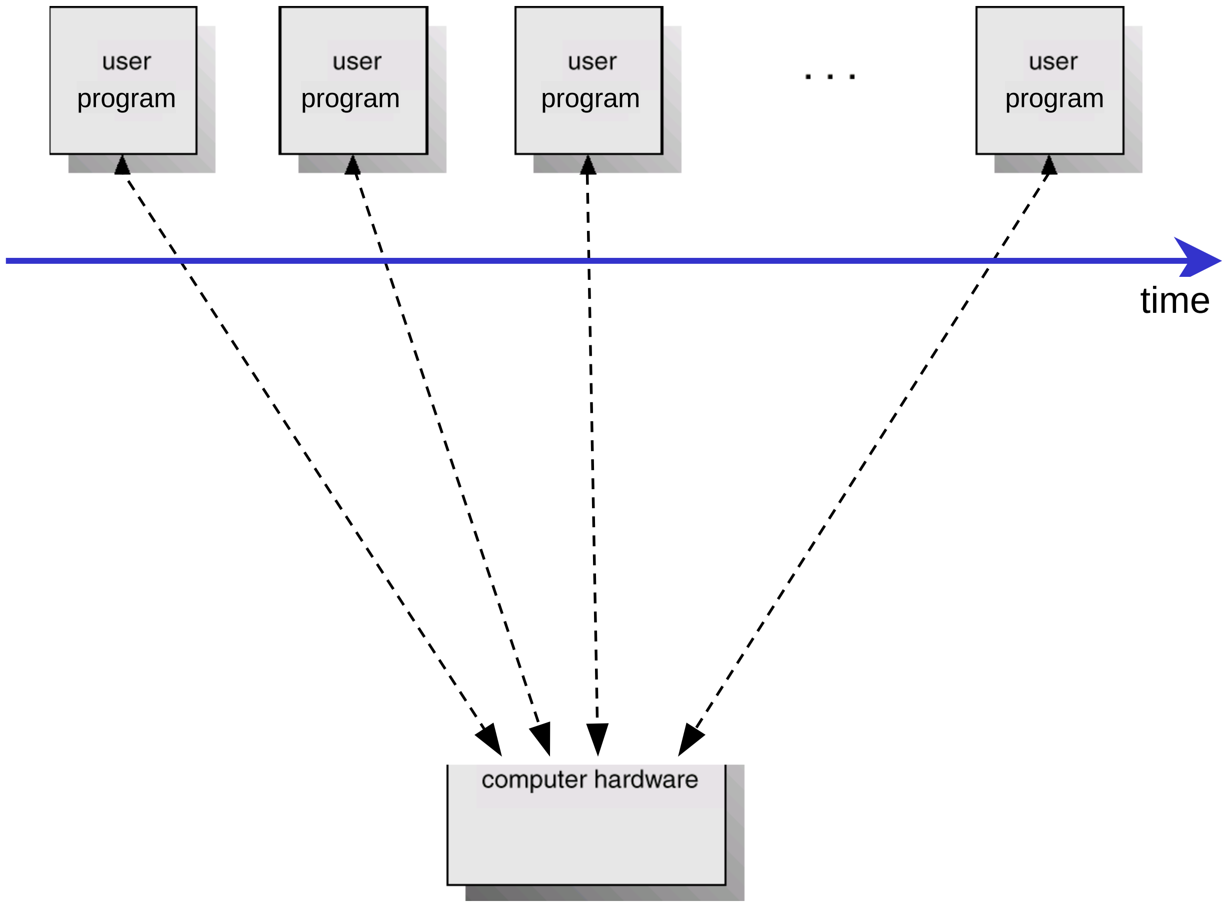
Serial processing

- Human operator-programmer-user
  - ✓ the machine was run from a console that had display lights, toggle switches, a plugboard or punched cards, a printer
  - ✓ the programmer also "operated" the machine as she/he interacted directly with the bare hardware
  - ✓ at first the computer was programmed by physically re-wiring it; later, through stored programs ("von Neumann architecture")
- Operating systems were unheard of
  - ✓ programs were entirely written in machine or assembly language
  - ✓ one running program had complete control of the entire computer

1.b Operating System History and Features

Serial processing

- Programs directly access the hardware, one at a time



1.b Operating System History and Features

Serial processing

- **Problem 1: scheduling**
  - ✓ users had access to the computer one by one in series
  - ✓ machine time was reserved in blocks of half hours with a hard-copy sign-up sheet
  - ✓ either the user was finished early and computer processing time was wasted
  - ✓ or, more frequently, the user could not finish debugging her/his program during the allotted time
- **Problem 2: duplication of programming efforts**
  - ✓ user wrote again and again the same routines (ex: I/O devices)
  - ✓ no concept of libraries

1.b Operating System History and Features

Simple batch systems

- **Second generation: 1955–65**
  - ✓ advent of transistors and printed circuits

http://www.columbia.edu/acis/history/1965.html

The IBM 7094 at Columbia University

1.b Operating System History and Features

Simple batch systems

- Separation between operators and programmers
  - ✓ first commercially viable machines
  - ✓ the programmer prepares her/his job off-line on punched cards, brings the card deck to the machine room and waits for results
  - ✓ the human operator runs the job and delivers a printed output
- New problem: still basically serial processing
  - ✓ one single **job** at a time
  - ✓ huge setup time for each job: loading the compiler, the source program, saving the compiled program, loading, linking, etc.
  - ✓ also mounting and dismounting tapes, handling card decks, etc.
  - ✓ a lot of time was wasted manipulating things and walking around

1.b Operating System History and Features

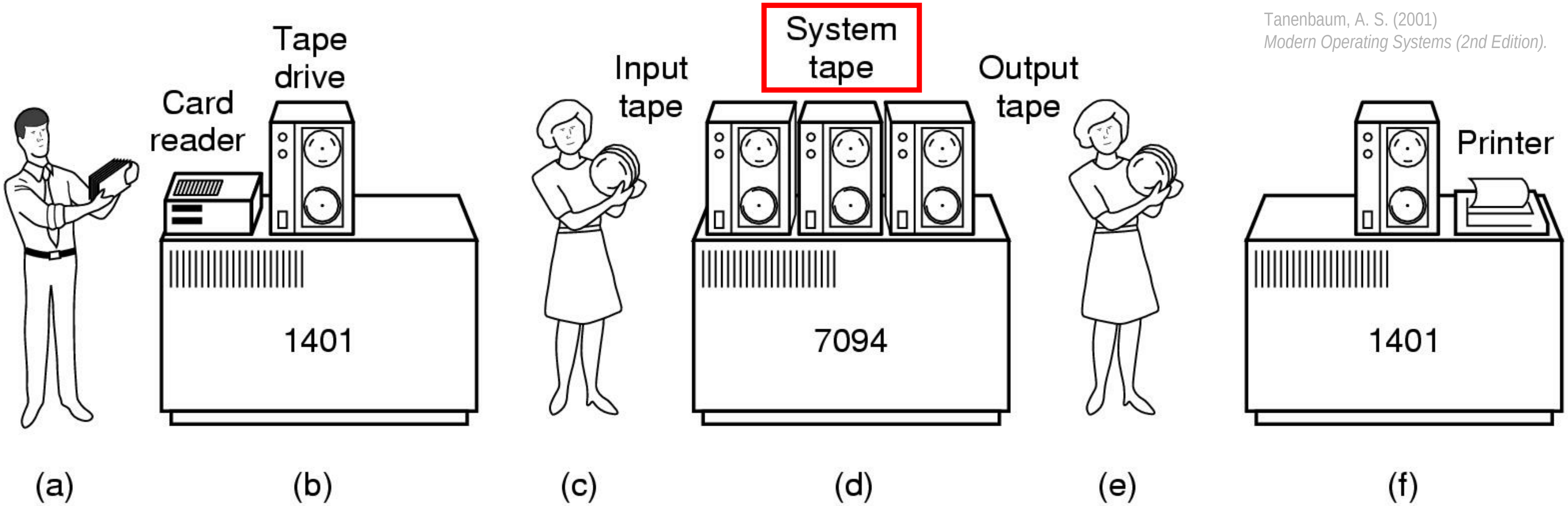
Simple batch systems

- Solution: batch the jobs together
  1. the human operator pre-reads a tray full of jobs onto a magnetic tape
  2. the human operator loads a special program, the **monitor**, that will automatically read the jobs from the tape and run them sequentially
  3. the effect of the monitor program is to write the output of each job on a second magnetic tape
  4. finally, the human operator brings the full output tape for offline printing

1.b Operating System History and Features

Simple batch systems

Tanenbaum, A. S. (2001)
Modern Operating Systems (2nd Edition).



a) programmer brings cards to IBM 1401
b) 1401 reads batch of jobs onto tape
c) operator carries input tape to IBM 7094
d) 7094 does computing
e) operator carries output tape to 1401
f) 1401 prints output

An early IBM batch system

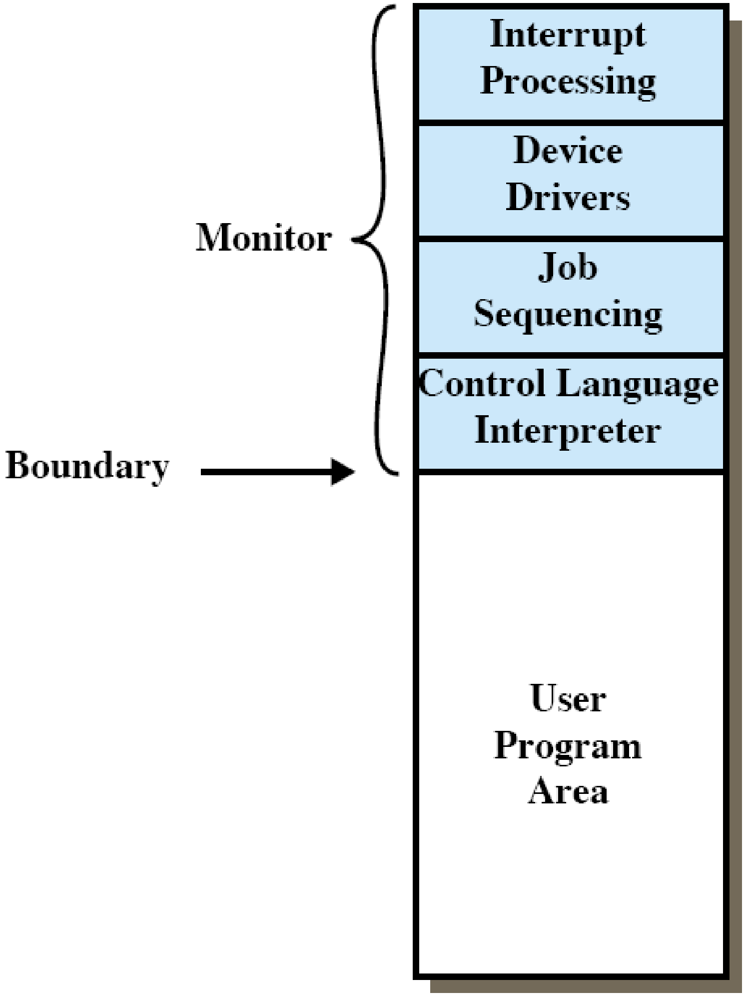
1.b Operating System History and Features

Simple batch systems

- The monitor program automates some of the human operator's tasks and is the ancestor of modern O/S
  - ✓ the monitor is a special program that controls the sequence of events
  - ✓ it always resides in main memory
  - ✓ it reads in jobs one at a time, places a job in the user program area of the memory, and passes control to it
  - ✓ upon completion, the user program branches back to the monitor, which immediately loads and executes the next job
  - ✓ therefore, the CPU alternates between fetching/executing instructions from the monitor program and fetching/executing instructions from the user program

1.b Operating System History and Features

Simple batch systems



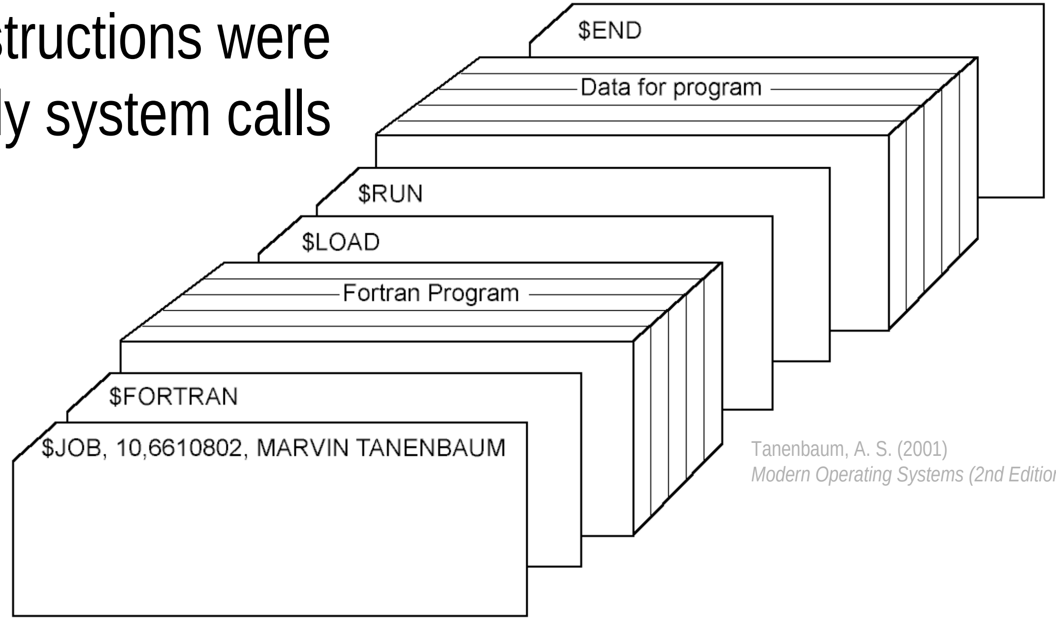
Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Memory layout for a resident monitor

1.b Operating System History and Features

Simple batch systems

- A Job Control Language (JCL) operates the monitor
  - ✓ primitive programming language: **`$JOB`**, **`$FTN`**, **`$LOAD`**, **`$RUN`**
  - ✓ gives instructions to the monitor about what compiler to use, what data to work on, etc.
  - ✓ JCL instructions were the early system calls



Tanenbaum, A. S. (2001)
Modern Operating Systems (2nd Edition).

Structure of a typical Fortran Monitor System (FMS) job's card deck

1.b Operating System History and Features

Simple batch systems

- A batch system monitor is a special program that cohabitates in memory with the job being executed
  - ✓ the monitor must alternate between seize and relinquish control
  - ✓ the monitor must switch among various portions of memory
- Hardware features needed to support batch
  - ✓ memory protection
  - ✓ timer (long-term, NOT preemptive)
  - ✓ privileged instructions
  - ✓ voluntary interrupts (monitor calls)
  - → this led to the concept of “modes of operation”

1.b Operating System History and Features

Simple batch systems

- **Memory protection**
  - ✓ a job must not alter the memory area containing the monitor
- **Timer (long-term)**
  - ✓ a global time limit prevents a job from monopolizing the system
- **Privileged instructions**
  - ✓ certain machine-level instructions can only be executed by the monitor, for ex: access to all CPU registers or I/O access
- **Voluntary interrupts (traps to monitor)**
  - ✓ make it easier to relinquish control to, and regain control from user programs (early computers did not have this capability)

1.b Operating System History and Features

Simple batch systems

- This led to the concept of “modes of operation”
  - ✓ user programs execute in **user mode**: certain instructions may not be executed
  - ✓ monitor executes in **monitor mode** (or “system mode”, “control mode”, “kernel mode”, “privileged mode”, “supervisor mode”, etc.)
  - ✓ in monitor mode, privileged instructions can be executed, protected areas of memory and I/O devices may be accessed
  - ✓ user/monitor mode is generally flagged by a bit in the Program Status Word (PSW) register of the CPU

1.b Operating System History and Features

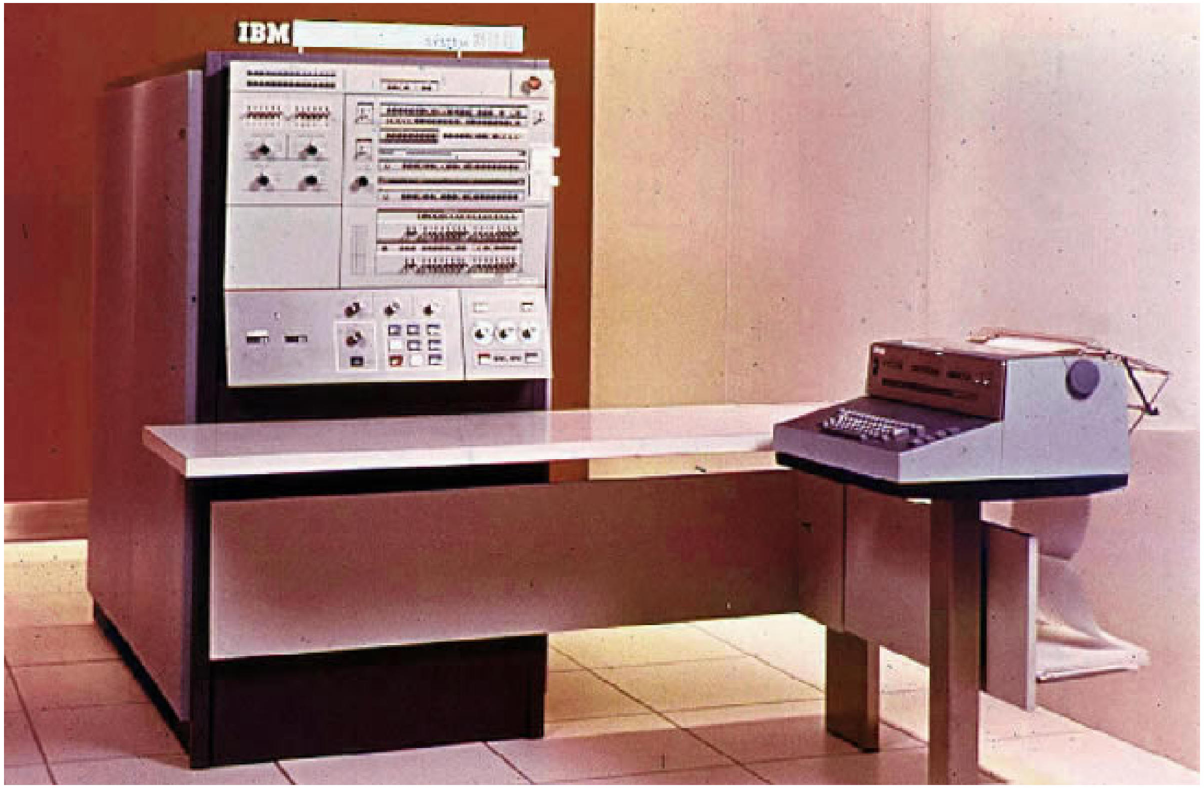
Note: The history of computer systems reveals the problems faced by the early designers and operators, and the key ideas and solutions still around today.

Computer science is fundamentally an **empiric** science: features have mainly evolved from needs.

1.b Operating System History and Features

Multiprogrammed batch systems

- Third generation: 1965-80
  - ✓ first major use of small-scale Integrated Circuits (ICs)



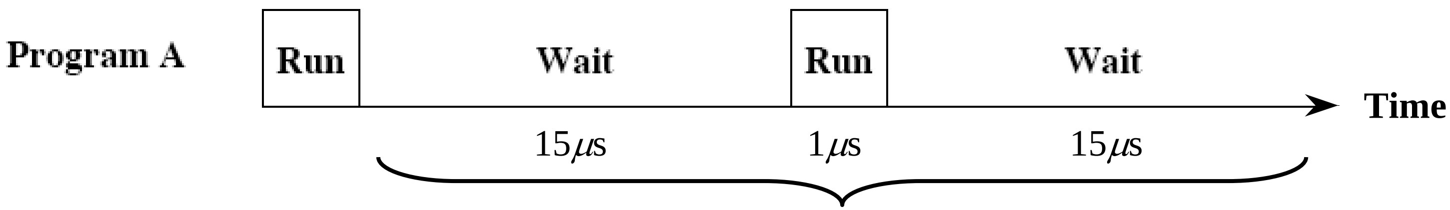
http://www.thocp.net/hardware/pictures/

The IBM 360

1.b Operating System History and Features

Multiprogrammed batch systems

- **Problem: despite batching, a lot of CPU time is still wasted waiting for I/O instructions to complete**
  - ✓ I/O devices much slower than processor (especially tapes!)



Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
TOTAL	31 μs

$$\text{Percent CPU Utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

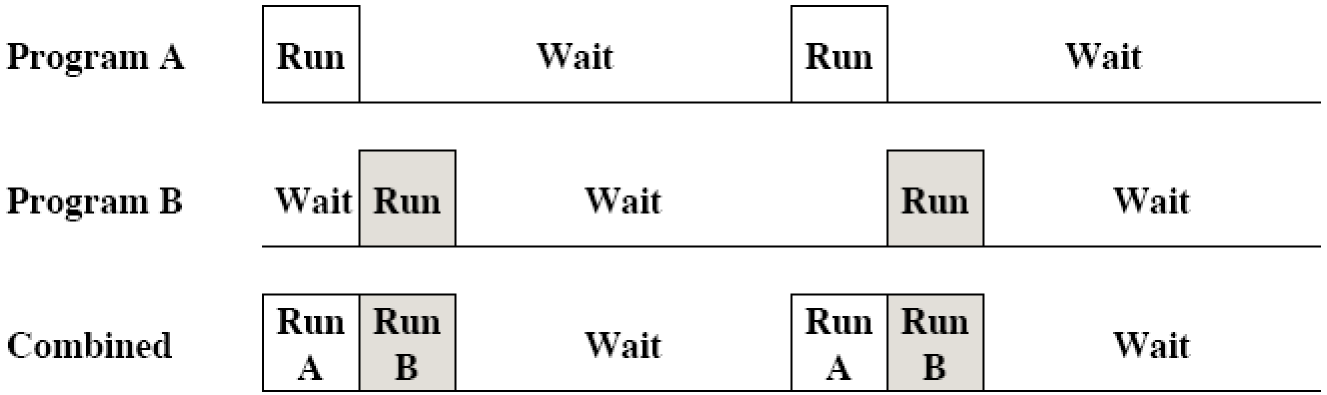
Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Example of system utilization with uniprogramming

1.b Operating System History and Features

Multiprogrammed batch systems

- **Solution: load two jobs in memory**
  - ✓ while one job is waiting for I/O, the processor could switch to the other job



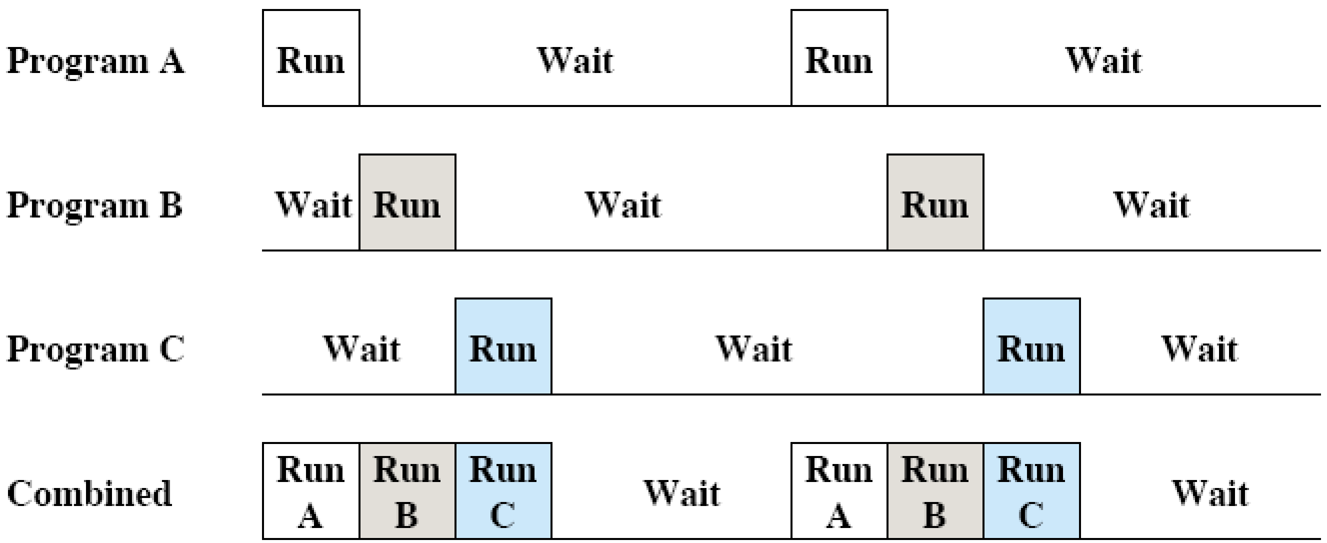
Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Multiprogramming with two programs

1.b Operating System History and Features

Multiprogrammed batch systems

- Expand to three, four or more jobs
  - ✓ jobs are kept in main memory at the same time and the CPU is multiplexed among them, or “multiprogrammed”
  - ✓ **multiprogramming** (“multitasking”) is a central O/S theme



Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Multiprogramming with three programs

1.b Operating System History and Features

Multiprogrammed batch systems

- Example of multiprogramming with three jobs

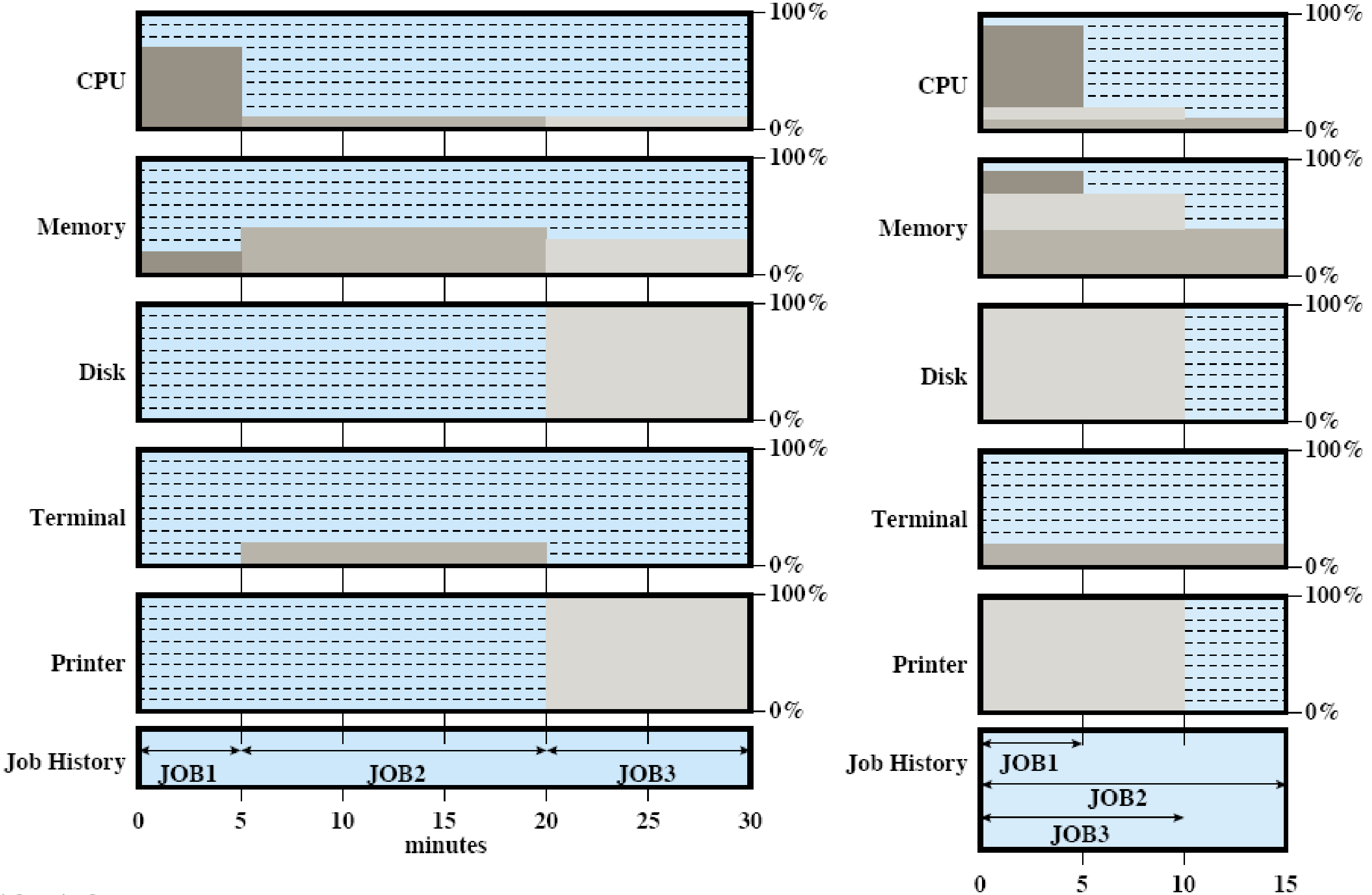
	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Example of program execution attributes

1.b Operating System History and Features

Multiprogrammed batch systems



Stallings, W. (2004) Operating Systems: Internals and Design Principles (5th Edition).

(a) uniprogramming

(b) multiprogramming

Utilization histograms

1.b Operating System History and Features

Multiprogrammed batch systems

- Multiprogramming results in more efficient resource utilization

	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

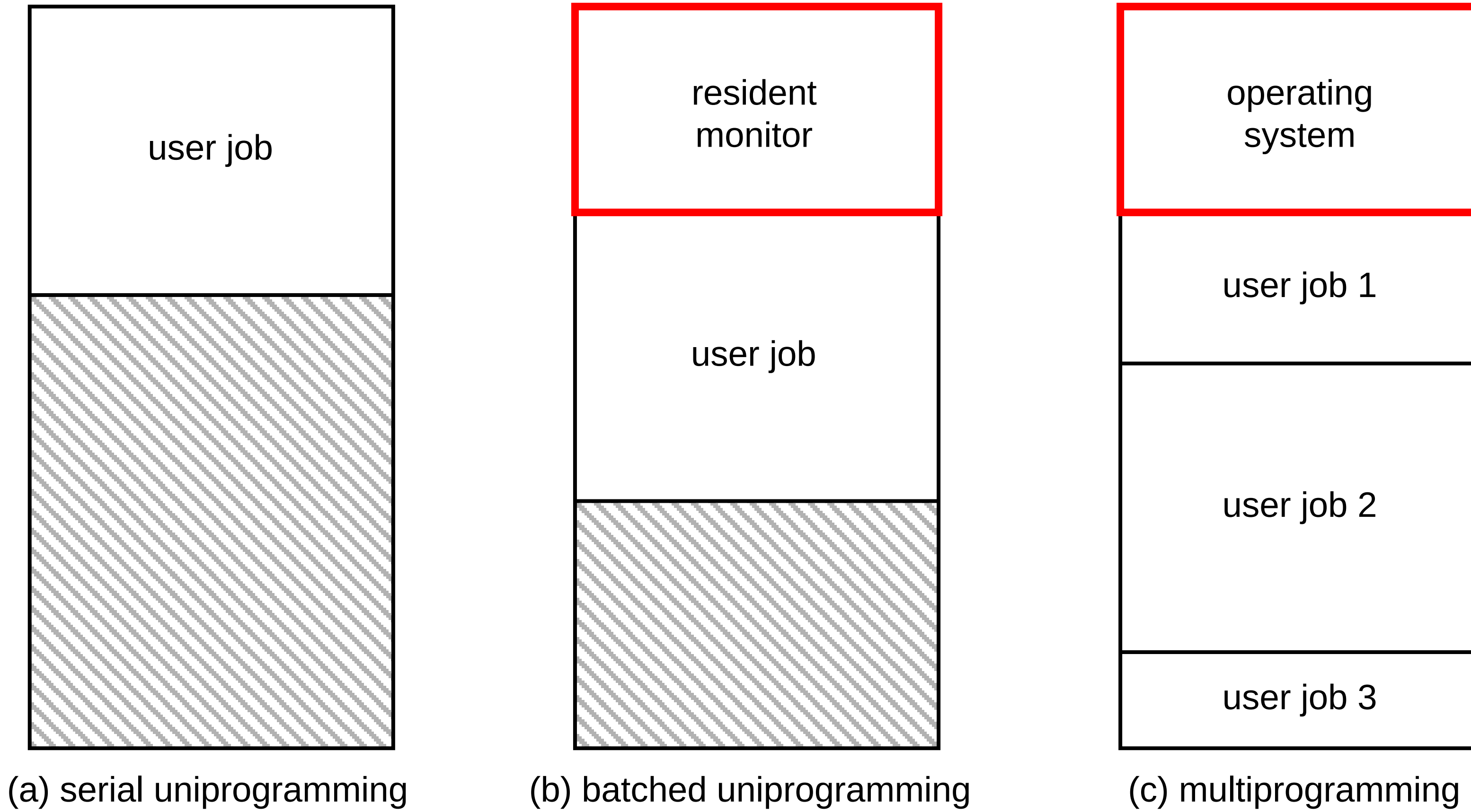
Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Effects of multiprogramming on resource utilization

1.b Operating System History and Features

Multiprogrammed batch systems

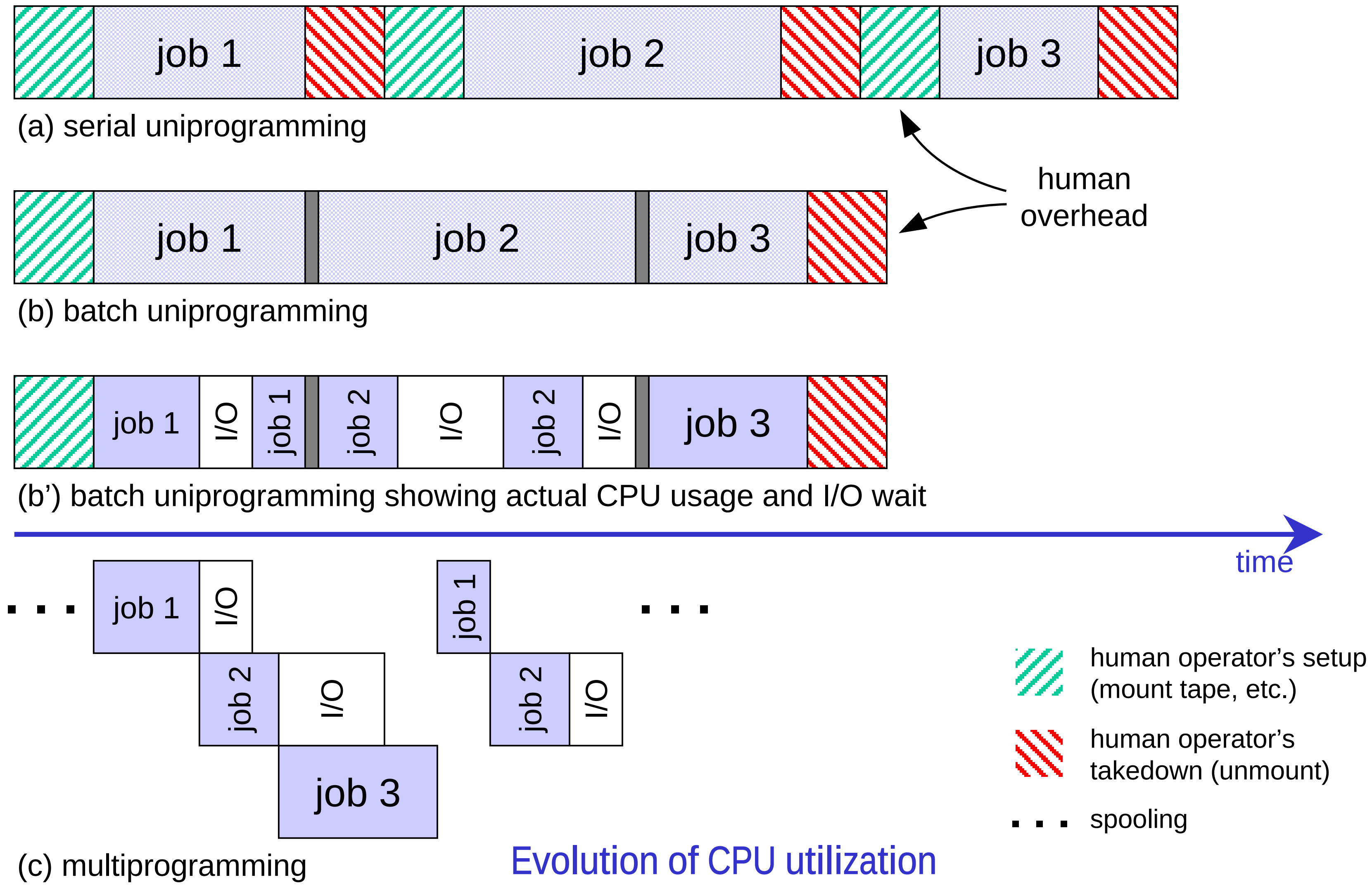
- Summary: serial, batched uni-, and multiprogramming



Evolution of memory management

1.b Operating System History and Features

Multiprogrammed batch systems



Evolution of CPU utilization

1.b Operating System History and Features

Multiprogrammed batch systems

- A multiprogramming O/S is fairly sophisticated compared to a uniprogramming O/S
  - ✓ it requires memory management: the system must allocate the memory to several jobs
  - ✓ it requires CPU scheduling: the system must choose among several jobs ready to run
- Multiprogramming also relies on I/O hardware features
  - ✓ I/O interrupts: <u>asynchronous</u> interrupts from I/O device, upon completion
  - ✓ Direct Memory Access (DMA)

1.b Operating System History and Features

Multiprogrammed batch systems

Forward Note: Three I/O Techniques (see 5.)

- ✓ **Programmed I/O** ("busy waiting"): the CPU is blocked and must poll the device to check if the I/O request completed
- ✓ **Interrupt-driven I/O**: the CPU can switch to other tasks and is (frequently) interrupted by the I/O device
- ✓ **Direct Memory Access (DMA)**: the CPU is involved only at the start and the end of the whole transfer; it delegates control to an independent I/O controller that accesses memory directly without bothering the CPU

} needed for multi-programming

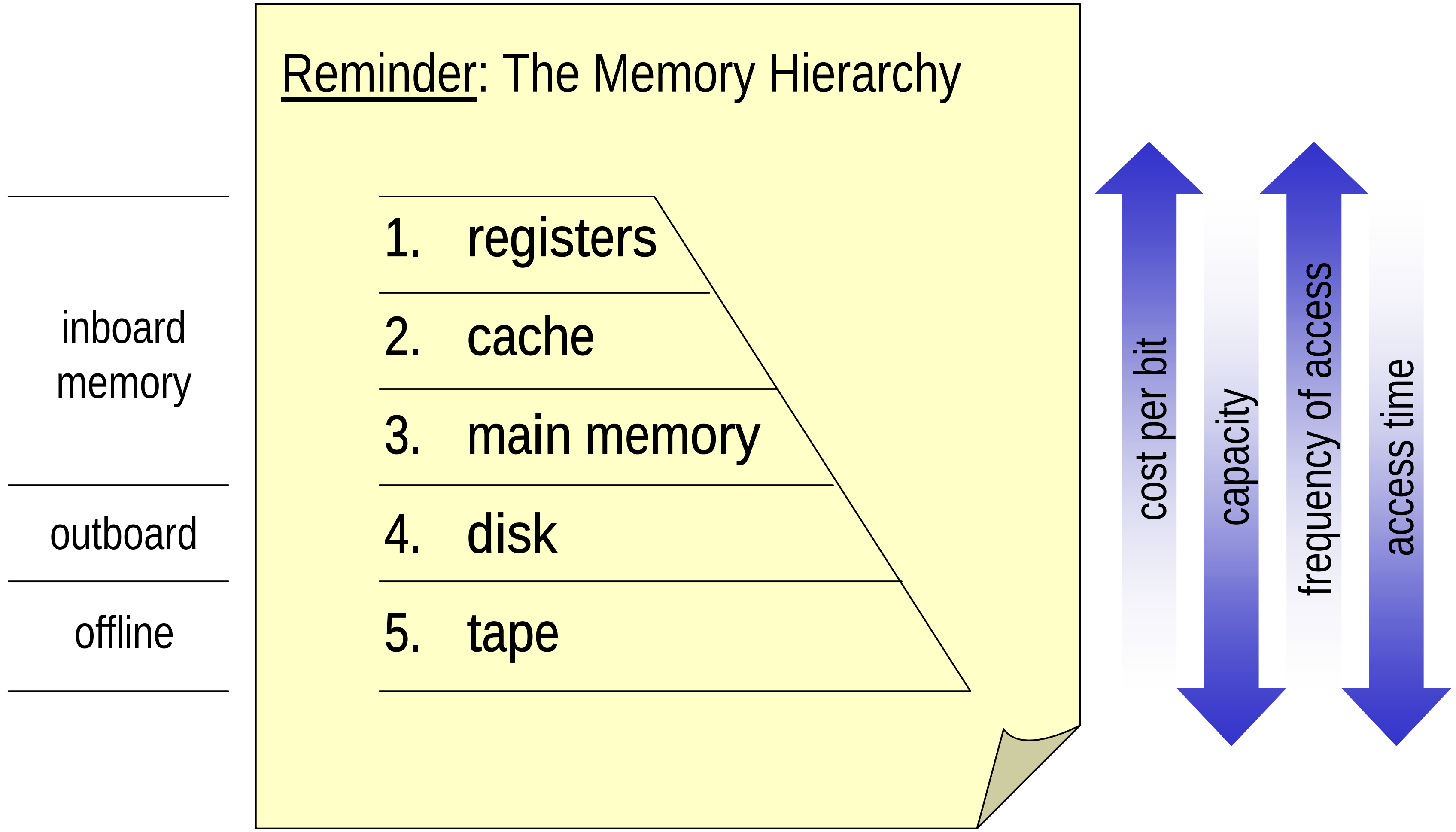
1.b Operating System History and Features

Multiprogrammed batch systems

- Spooling (Simultaneous Peripheral Operation On-Line) is an important I/O feature for multiprogramming
  - ✓ instead of preparing a batch of jobs on tape, new jobs are continuously and directly read in from cards onto disk as soon as they are brought to the computer room
  - ✓ producer/consumer scheme: spooling decouples job loading from job execution through a **buffer** (on disk or in memory)
  - ✓ useful because I/O devices access (read, write) data at different rates; the buffer provides a waiting station where data can rest while the slower device catches up
  - ✓ same in the output direction (printers)

1.b Operating System History and Features

Multiprogrammed batch systems



1.b Operating System History and Features

Multiprogrammed batch systems

Forward Note: Types of Scheduling

- ✓ **Long-term scheduling**: which jobs (stored on disk) will be considered for execution
- ✓ **Medium-term scheduling** (“swapping”): which jobs are actually loaded into memory
- ✓ **Short-term (CPU) scheduling** (“dispatching”): which job available in memory is run next

1.b Operating System History and Features

Time-sharing systems

- Batch multiprogramming was not fully satisfactory
  - ✓ multiprogramming alone does not give any guarantee that a program will run in a timely manner
  - ✓ users had a growing need to control more closely the execution of their jobs and intervene (fix, retry, etc.)
- There was a need for multiple-user interactivity
  - ✓ each user wants to see their program running as if it was the only program in the computer
  - ✓ but without reverting back to single-user signup sheets
  - ✓ therefore the advent of time-sharing or **preemptive multitasking** systems

1.b Operating System History and Features

Time-sharing systems

- In the original multiprogramming systems
  - ✓ tasks kept running until they performed an operation that required waiting for an external event such as I/O
  - ✓ multiprogramming was designed to <u>maximize CPU usage</u>
- In time-sharing systems
  - ✓ running tasks are required to relinquish the CPU on a regular basis through hardware interrupts (timer)
  - ✓ time-sharing is designed to <u>minimize response time</u> and allow several programs to execute apparently simultaneously
  - ✓ time sharing is a logical extension of multiprogramming for handling multiple **interactive** jobs among multiple users
  - ✓ birth of Unix in the 1960's: CTSS → MULTICS → UNIX

1.b Operating System History and Features

Personal computers

- Fourth generation: 1980-Present
  - ✓ Large Scale Integration (LSI) makes personal computing real

http://www.it.unr.edu/facilities/cordlab.asp

E. L. Cord computer lab at UNR

1.b Operating System History and Features

Personal computers

- **From multiple users back to a single user**
  - ✓ preemptive multitasking was developed in the 1960's to share big and costly mainframe computers among multiple users
  - ✓ since then, **single-user** interactive computing has become possible on dedicated personal computers (PCs)
- **Resource sharing not critical anymore, yet multitasking still a central feature of modern PC operating systems**
  - ✓ a single-tasking environment is tedious: one must close the drawing application before opening the word processor, etc.
  - ✓ multitasking makes it possible for a single user to run multiple applications at the same time (or "background" processes) while retaining control of the computer

1.b Operating System History and Features

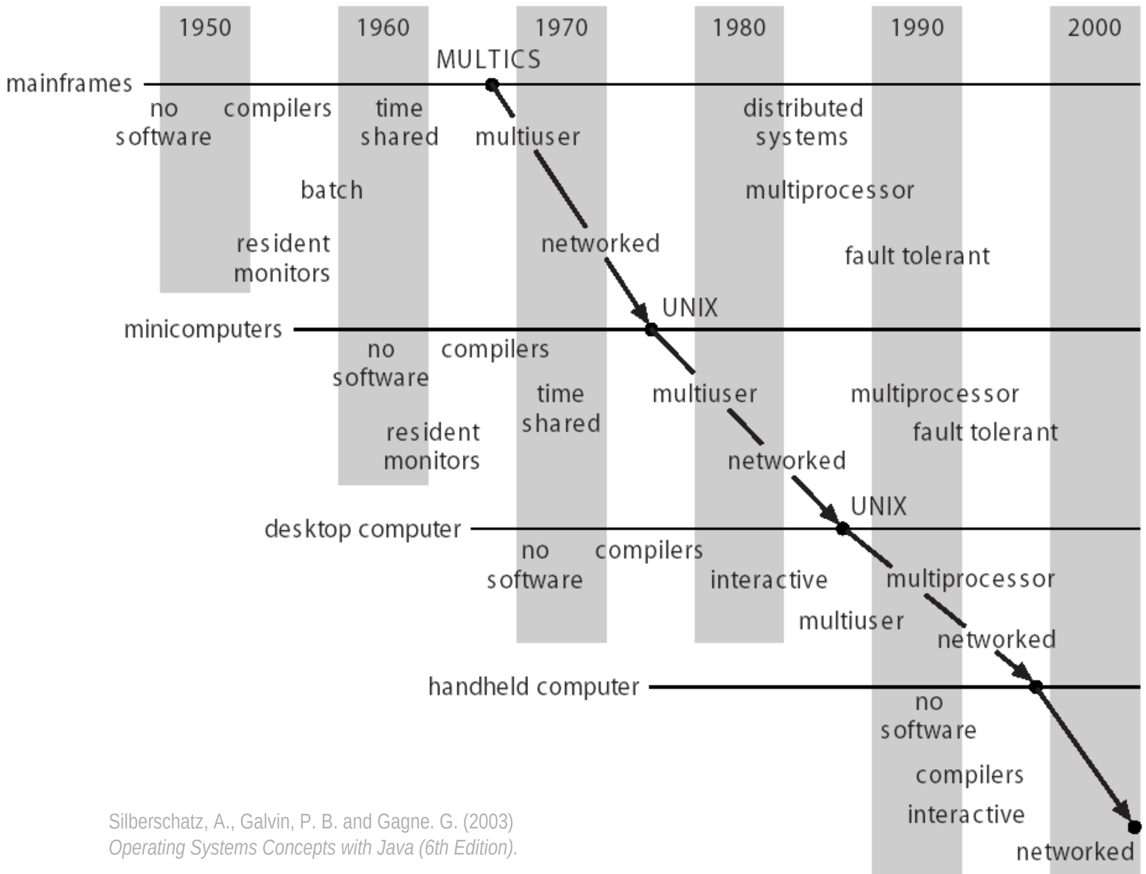
Personal computers

- Other mainframe system features have been integrated into PC systems, for example: file protection
  - ✓ in multi-user systems, file protection was critical
  - ✓ in single-user PCs, it was not considered necessary at first, but reappeared with the advent of networking
- PC systems emphasize user convenience
  - ✓ the primary goal of the mainframe multiprogrammed systems was to maximize CPU utilization
  - ✓ as in time-sharing systems, the primary goal of PC systems is rather to maximize user convenience and responsiveness

1.b Operating System History and Features

Personal computers

- "Ontogeny recapitulates phylogeny"



Silberschatz, A., Galvin, P. B. and Gagne. G. (2003)
Operating Systems Concepts with Java (6th Edition).

Migration of operating system concepts and features

Principles of Operating Systems
CS 446/646

1. Introduction to Operating Systems

a. Role of an O/S

b. **O/S History and Features**
   - ✓ Serial processing
   - ✓ Simple batch systems
   - ✓ Multiprogrammed batch systems
   - ✓ Time-sharing systems
   - ✓ Personal computers

c. **Types of O/S**

d. **Major O/S Components**

e. **System Calls**

f. **O/S Software Architecture**

g. **Examples of O/S**