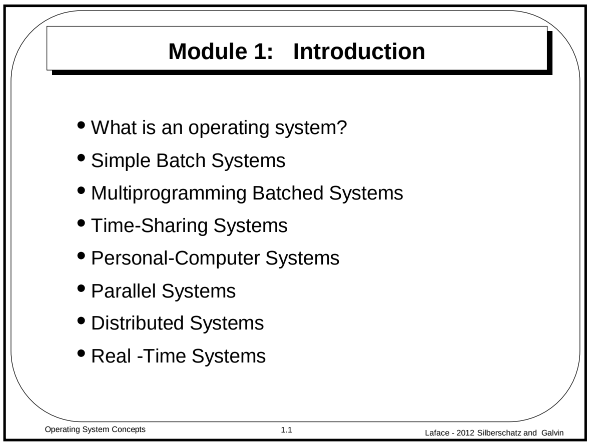
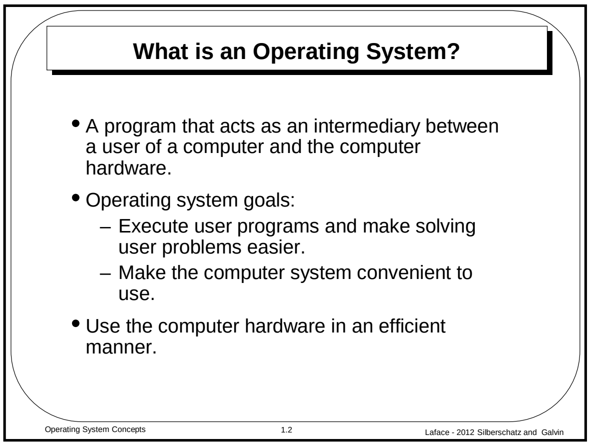


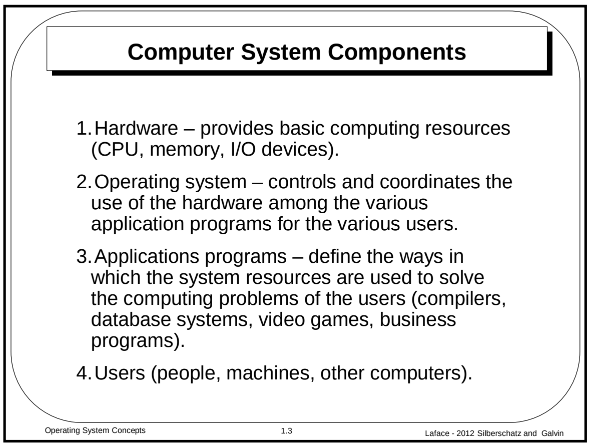
Module 1: Introduction

- What is an operating system?
- Simple Batch Systems
- Multiprogramming Batched Systems
- Time-Sharing Systems
- Personal-Computer Systems
- Parallel Systems
- Distributed Systems
- Real -Time Systems



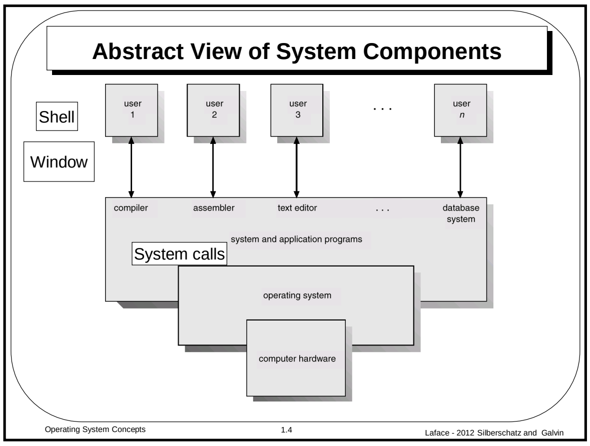
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
  - Execute user programs and make solving user problems easier.
  - Make the computer system convenient to use.
- Use the computer hardware in an efficient manner.

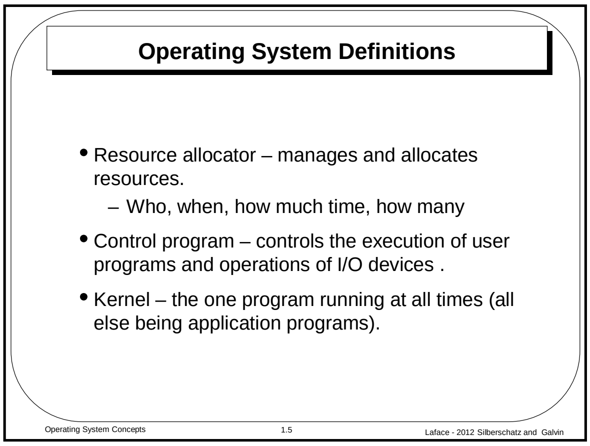


Computer System Components

1. Hardware – provides basic computing resources (CPU, memory, I/O devices).
2. Operating system – controls and coordinates the use of the hardware among the various application programs for the various users.
3. Applications programs – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).
4. Users (people, machines, other computers).

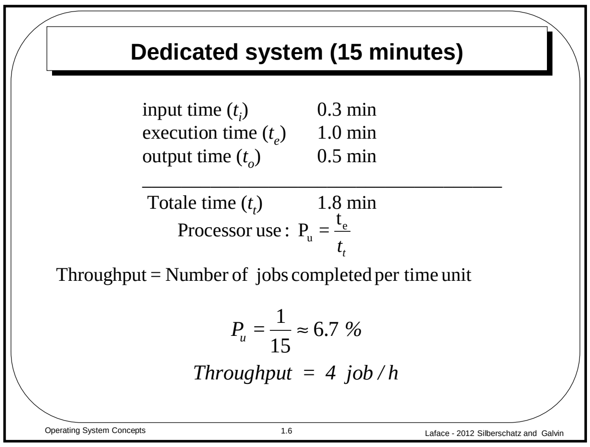


Abstract View of System Components



Operating System Definitions

- Resource allocator – manages and allocates resources.
  - Who, when, how much time, how many
- Control program – controls the execution of user programs and operations of I/O devices .
- Kernel – the one program running at all times (all else being application programs).



Dedicated system (15 minutes)

input time (t_i)	0.3 min
execution time (t_e)	1.0 min
output time (t_o)	0.5 min
Totale time (t_t)	1.8 min

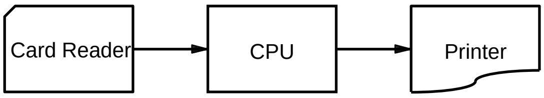
$$\text{Processor use: } P_u = \frac{t_e}{t_t}$$

Throughput = Number of jobs completed per time unit

$$P_u = \frac{1}{15} \approx 6.7\ \%$$

$$Throughput\ = \ 4\ \ job / h$$

Simple Batch Systems

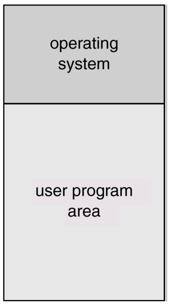


- Hire an operator
- User ≠ operator
- Add a card reader
- Automatic job sequencing – automatically transfers control from one job to another. First rudimentary operating system.

Simple Batch Systems

- Resident monitor
  - – initial control in monitor
  - – control transfers to job
  - – when job completes control transfers back to monitor

Memory Layout for a Simple Batch System



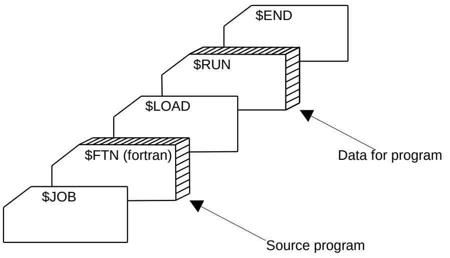
Control Cards

- Problems
  1. How does the monitor know about the nature of the job (e.g., Fortran versus Assembly) or which program to execute?
  2. How does the monitor distinguish
     (a) job from job?
     (b) data from program?
- Solution
  – Introduce control cards

Control Cards

- Special cards that tell the resident monitor which programs to run
  $JOB
  $FTN
  $RUN
  $DATA
  $END
- Special characters distinguish control cards from data or program cards:
  $ in column 1
  // in column 1 and 2
  709 in column1

Control Cards



$END

$RUN

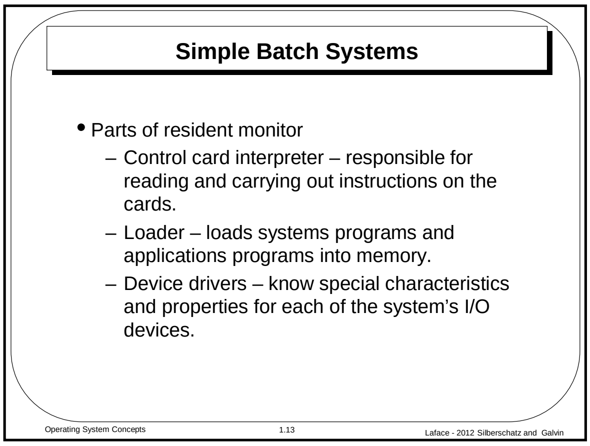
$LOAD

$FTN (fortran)

$JOB

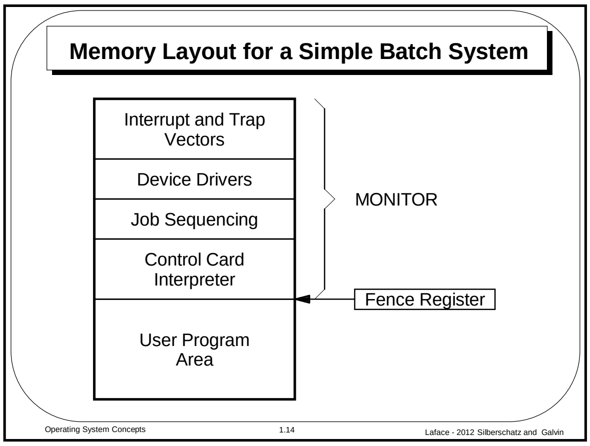
Data for program

Source program



Simple Batch Systems

- Parts of resident monitor
  - Control card interpreter – responsible for reading and carrying out instructions on the cards.
  - Loader – loads systems programs and applications programs into memory.
  - Device drivers – know special characteristics and properties for each of the system's I/O devices.



Memory Layout for a Simple Batch System

Interrupt and Trap Vectors

Device Drivers

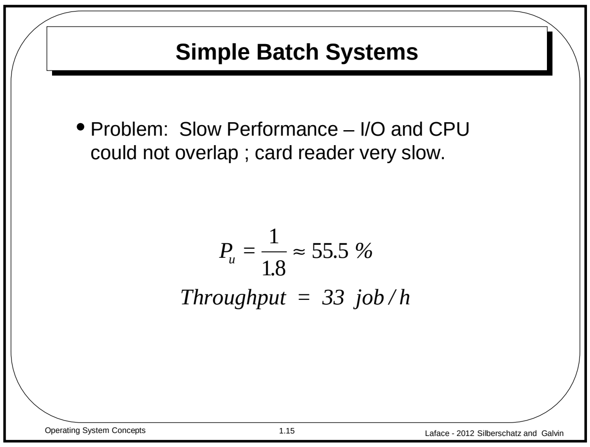
Job Sequencing

Control Card Interpreter

User Program Area

MONITOR

Fence Register



Simple Batch Systems

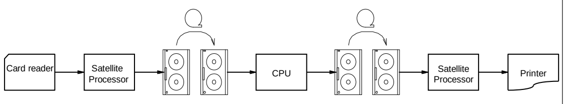
- Problem: Slow Performance – I/O and CPU could not overlap ; card reader very slow.

$$P_u = \frac{1}{1.8} \approx 55.5\ \%$$

$$Throughput\ =\ 33\ \ job/h$$

Simple Batch Systems

- Solution: Off-line operation – speed up computation by loading jobs into memory from tapes
- card reading and line printing done off-line



Off-line batch procesing

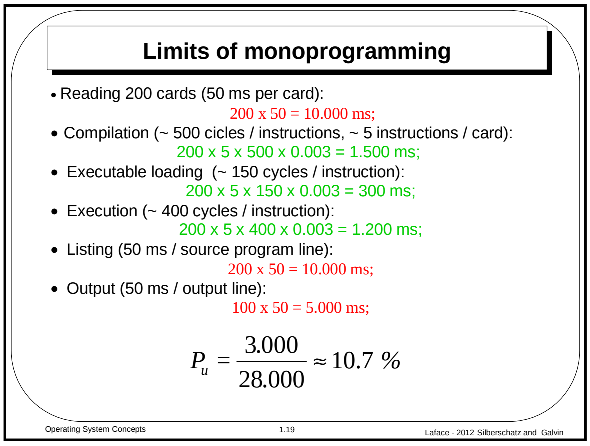
Job collection	50 job	30 min
Conversion cards → tape		15 min
Tape mounting		5 min
Batch execution		50 min
Conversion tape → printer		25 min
Output separation		15 min
Batch response time		140 min

$$P_u = \frac{50}{55} \approx 90.9\ \%$$

$$Throughput\ =\ \frac{50\ \ job}{55\ min}\ \approx\ 55\ \ job/h$$

Off-line batch processing

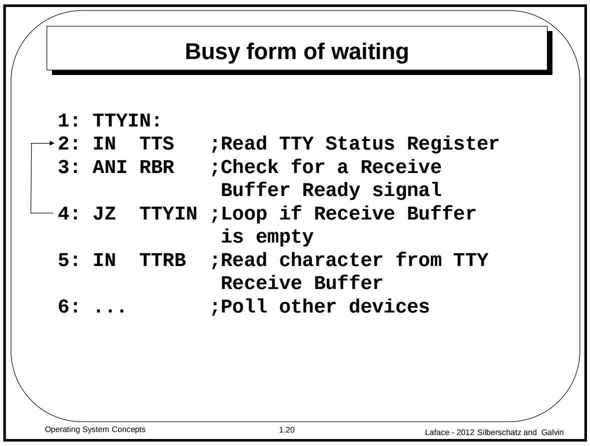
Job mean duration	Mean response time
< 1 min	2 - 3 h
< 5 min	8 - 10 h
> 5 min	1- 7 days



Limits of monoprogramming

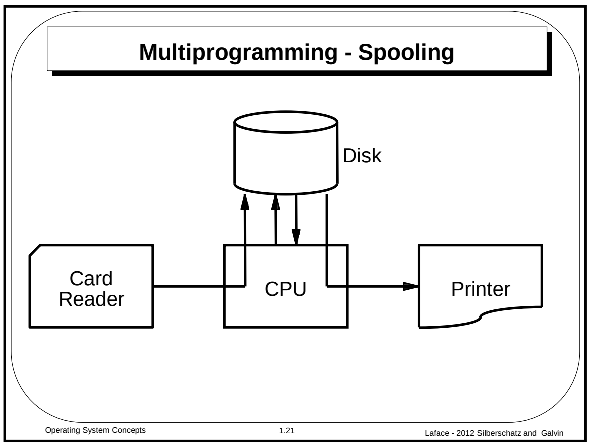
- Reading 200 cards (50 ms per card):
  200 x 50 = 10.000 ms;
- Compilation (~ 500 cicles / instructions, ~ 5 instructions / card):
  200 x 5 x 500 x 0.003 = 1.500 ms;
- Executable loading (~ 150 cycles / instruction):
  200 x 5 x 150 x 0.003 = 300 ms;
- Execution (~ 400 cycles / instruction):
  200 x 5 x 400 x 0.003 = 1.200 ms;
- Listing (50 ms / source program line):
  200 x 50 = 10.000 ms;
- Output (50 ms / output line):
  100 x 50 = 5.000 ms;

$$P_u = \frac{3.000}{28.000} \approx 10.7\ \%$$



Busy form of waiting

```
TTYIN:
IN   TTS   ;Read TTY Status Register
ANI  RBR   ;Check for a Receive
            Buffer Ready signal
JZ   TTYIN ;Loop if Receive Buffer
            is empty
IN   TTRB  ;Read character from TTY
            Receive Buffer
...        ;Poll other devices
```



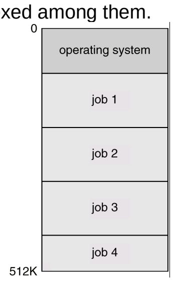
Multiprogramming - Spooling

Spooling

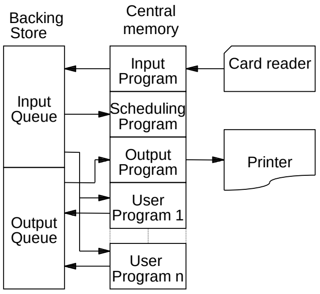
- Overlap I/O of one job with computation of another job. While executing one job, the OS
  - – Reads next job from card reader into a storage area on the disk (job queue).
  - – Outputs printout of previous job from disk to printer.
- Job pool – data structure that allows the OS to select which job to run next in order to increase CPU utilization.

Multiprogrammed Batch Systems

Several jobs are kept in main memory at the same time, and the
CPU is multiplexed among them.



Multiprogramming - Spooling



OS Features Needed for Multiprogramming

- I/O routine supplied by the system.
- Memory management – the system must allocate the memory to several jobs.
- CPU scheduling – the system must choose among several jobs ready to run.
- Allocation of devices.

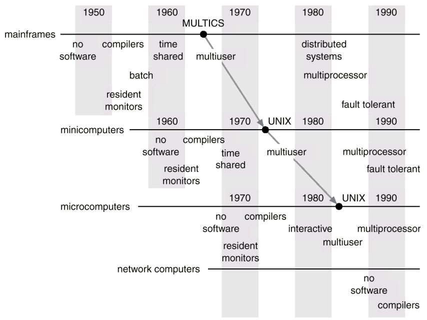
Time-Sharing Systems–Interactive Computing

- The CPU is multiplexed among several jobs that are kept in memory and on disk (the CPU is allocated to a job only if the job is in memory).
- A job is swapped in and out of memory to the disk.
- On-line communication between the user and the system is provided; when the operating system finishes the execution of one command, it seeks the next “control statement” not from a card reader, but rather from the user’s keyboard.
- On-line system must be available for users to access data and code.

Personal-Computer Systems

- Personal computers – computer system dedicated to a single user.
- I/O devices – keyboards, mice, display screens, small printers.
- User convenience and responsiveness.
- Can adopt technology developed for larger operating system’ often individuals have sole use of computer and do not need advanced CPU utilization of protection features.

Migration of Operating-System Concepts and Features



1950 1960 1970 1980 1990

MULTICS

mainframes

no software

compilers

time shared

multiuser

distributed systems

batch

multiprocessor

resident monitors

fault tolerant

minicomputers

1960 1970 UNIX 1980 1990

no software

compilers

time shared

multiuser

multiprocessor

resident monitors

fault tolerant

microcomputers

1970 1980 UNIX 1990

no software

compilers

interactive

multiprocessor

multiuser

resident monitors

network computers

no software

compilers

Parallel Systems

- Multiprocessor systems with more than one CPU in close communication.
- Tightly coupled system – processors share memory and a clock; communication usually takes place through the shared memory.
- Advantages of parallel system:
  - Increased throughput
  - Economical
  - Increased reliability
    - ❄graceful degradation
    - ❄fail-soft systems

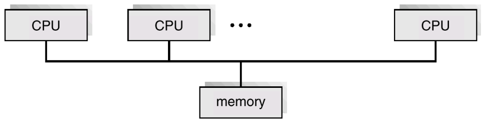
Parallel Systems (Cont.)

- Symmetric multiprocessing (SMP)
  - Each processor runs an identical copy of the operating system.
  - Many processes can run at once without performance deterioration.
  - Most modern operating systems support SMP

Parallel Systems (Cont.)

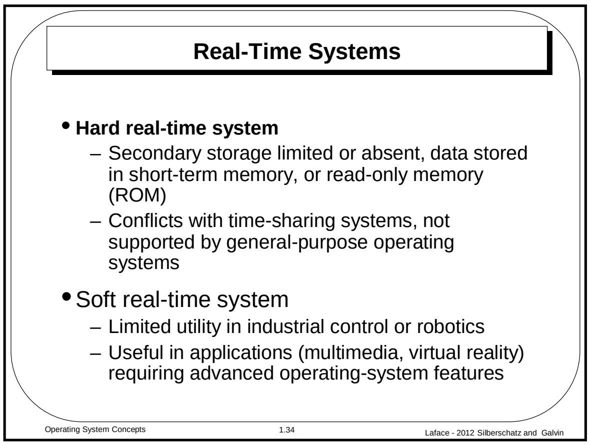
- Asymmetric multiprocessing
  - Each processor is assigned a specific task; master processor schedules and allocates work to slave processors.
  - More common in extremely large systems

Symmetric Multiprocessing Architecture



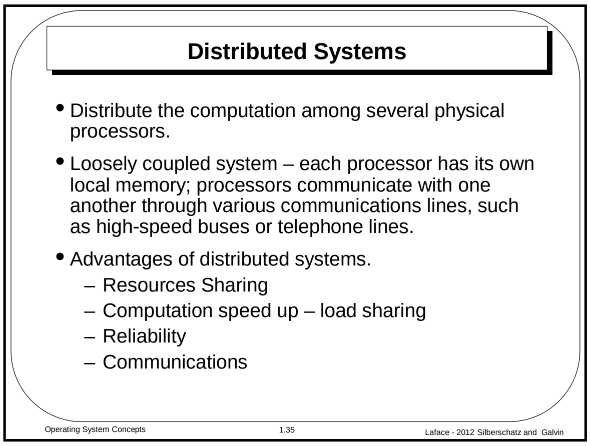
Real-Time Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints: the task correctness depends on when a given operation is performed



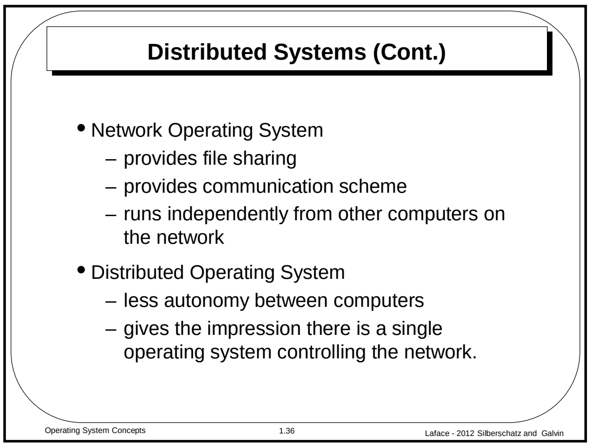
Real-Time Systems

- **Hard real-time system**
  - Secondary storage limited or absent, data stored in short-term memory, or read-only memory (ROM)
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems
- Soft real-time system
  - Limited utility in industrial control or robotics
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features



Distributed Systems

- Distribute the computation among several physical processors.
- Loosely coupled system – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- Advantages of distributed systems.
  - Resources Sharing
  - Computation speed up – load sharing
  - Reliability
  - Communications



Distributed Systems (Cont.)

- Network Operating System
  - provides file sharing
  - provides communication scheme
  - runs independently from other computers on the network
- Distributed Operating System
  - less autonomy between computers
  - gives the impression there is a single operating system controlling the network.