COMPUTER ORGANIZATION AND ARCHITECTURE (COA)

EET 2211

EET 2211

4TH SEMESTER – CSE & CSIT

CHAPTER 8, LECTURE 29

CHAPTER 8 – OPERATING SYSTEM SUPPORT

TOPICS TO BE COVERED

- Operating System Overview
- Scheduling
- Memory Management

LEARNING OBJECTIVES

- Summarize the key functions of the OS.
- Discuss the evolution of the OS form early simple batch systems to modern complex systems.
- Explain the different types of scheduling .
- Understand the reason for memory partitioning and explain the various techniques that are used.
- Assess the relative advantages of paging and segmentation.
- Define Virtual memory.

OPERATING SYSTEM OVERVIEW

- An OS is a program that controls the execution of application programs and acts as an interface between applications and the computer hardware.
- Popular OS include LINUX OS, WINDOWS OS, VMS, OS/400, Z/OS etc.
- **OBJECTIVES**: (i) Convenience (an OS makes a computer more convenient to use)

  (ii) Efficiency (an OS allows the computer system resources to be used in an efficient manner)
- **ASPECTS OF OS**: (i) the OS as a user/computer interface

  (ii) the OS as a resource manager.

(1) THE OS AS A USER/COMPUTER INTERFACE

✓The end user is not concerned with the computer's architecture.

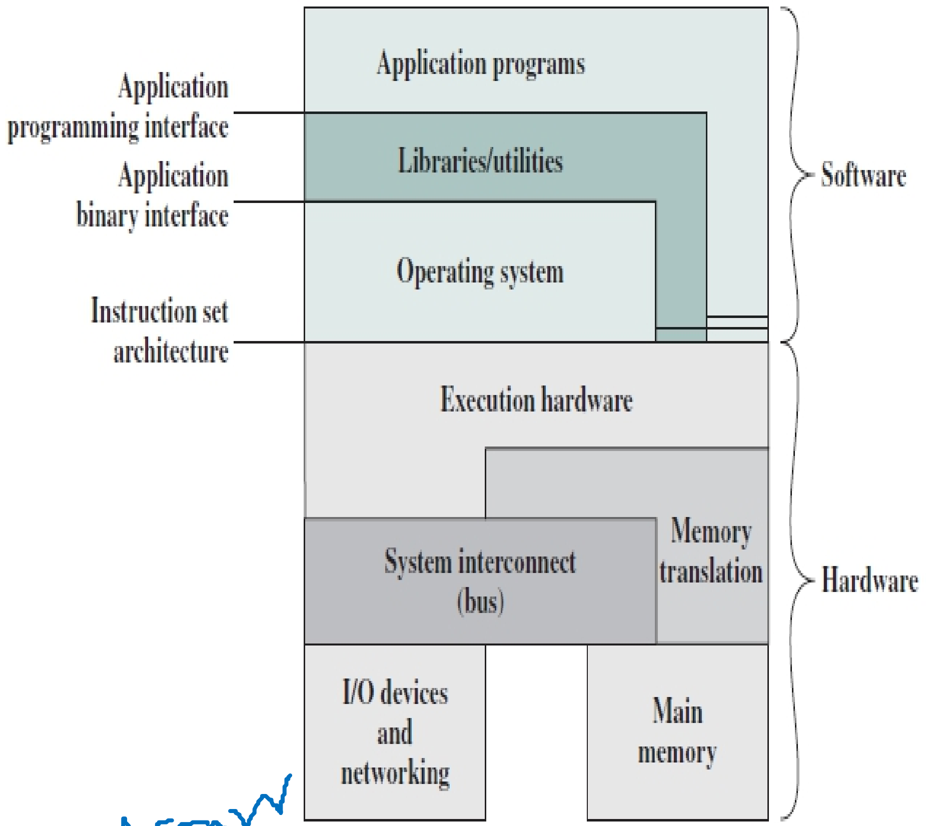
✓The application is expressed in a programming language.

✓Programs are referred as UTILITES.

✓The most important system program is the OS.

✓OS masks the details of the hardware from the programmer.

✓OS provides the programmer a convenient interface for using the system.


Application programs
Application programming interface
Libraries/utilities
Application binary interface
Operating system
Instruction set architecture
Software
Execution hardware
Memory translation
System interconnect (bus)
Hardware
I/O devices and networking
Main memory

Arora

✓Fig.1: Computer Hardware and Software Structure [Source: Computer Organization and Architecture by William Stallings]

Contd.

❖ ✓ The OS provides **SERVICES** in the following fields:

✓

1. Program creation
2. Program execution
3. Access to I/O devices
4. Controlled access to files
5. System access
6. Error detection and response
7. Accounting

Contd.

- **PROGRAM CREATION** : The OS provides a variety of facilities and services such as editors and debuggers to assist the programmer in creating programs. These programs in the form of utility programs that are not actually part of the OS but are accessible through the OS.

- **PROGRAM EXECUTION** : A number of steps need to be performed to execute a program. Instructions and adapt must be loaded into main memory, I/O devices and files must be initialized and other resources must be prepared.

- **ACCESS TO I/O DEVICES** : Each I/O device requires its own specific set of instructions or control signals for operation. The OS takes care of the details so that the programmer can think in terms of simple reads and writes.

- **CONTROLLED ACCESS TO FILES** : OS takes care about the details of the control that includes the nature of I/O devices (disk drive, tape drive) and also the file format on the storage medium. With multiple simultaneous users the OS can provide protection mechanisms to control access to the files.

- **SYSTEM ACCESS** : In case of a shared or public system, the OS controls access to the system as a whole and to specific system resources. The access function must provide protection of resources and data from unauthorized users and must resolve conflicts for resource contention.

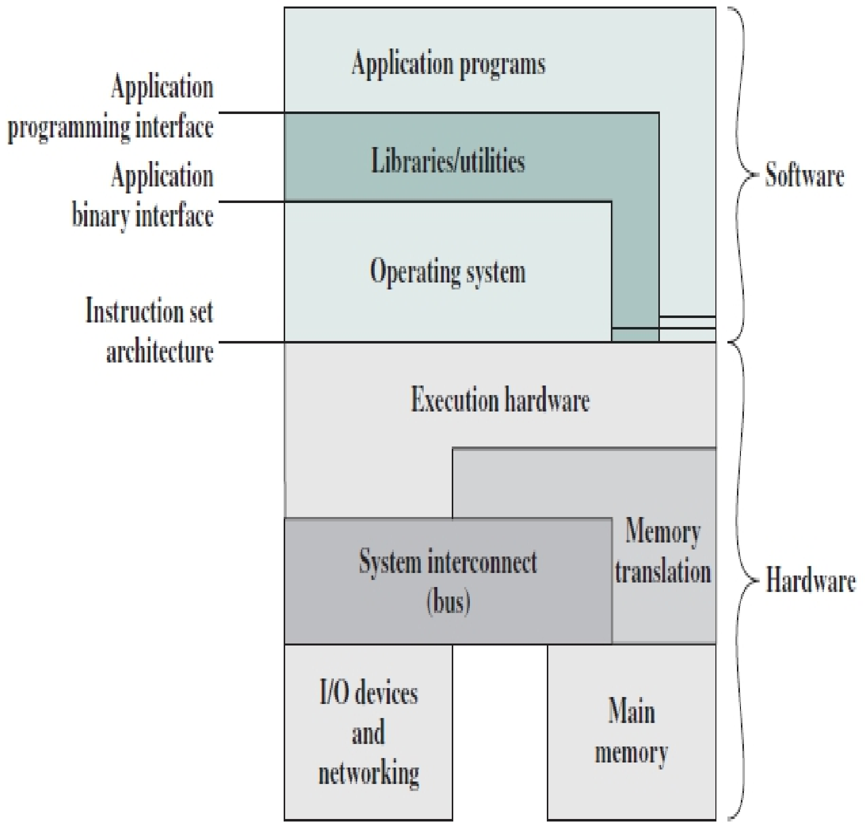
- **ERROR DETECTION AND RESPONSE** : A variety of errors can occur while a computer system is running. These include internal and external hardware errors, such as memory error or a device malfunction or failure, and various software errors such as arithmetic overflow, attempt to access forbidden memory location and inability of the OS to grant the request of an application. The response may range from ending the program that caused the error, retrying the operation and simply reporting the error to the application.

- **ACCOUNTING** : A good OS collects usage statistics for various resources and monitors performance such as response time.

Contd.

- Interfaces in a typical computer system

1. Instruction set architecture (ISA)
2. Application binary interface (ABI)
3. Application programming interface (API)


Application programs
Application programming interface
Libraries/utilities
Application binary interface
Operating system
Instruction set architecture
Software
Execution hardware
Memory translation
System interconnect (bus)
Hardware
I/O devices and networking
Main memory

✓THE OS AS A RESOURCE MANAGER

✓OS is responsible for managing the resources for the movement, storage and processing of data.

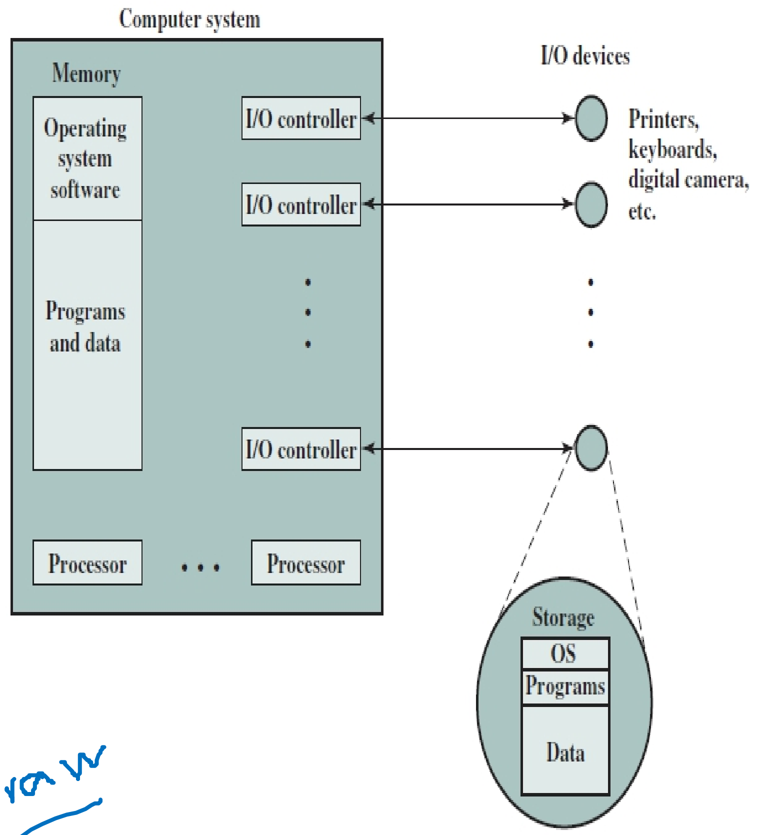
✓OS provides instructions for the processor.

✓OS directs the processor in the use of other system resources.

✓A portion of the OS is in the main memory which includes **KERNEL/NUCLEUS**.

✓The rest of the main memory contains the user programs and data.

✓OS decides when an I/O device can be used by a program in execution.



draw ✓

Fig.2: OS as Resource Manager [Source: Computer Organization and Architecture by William Stallings]

✓ TYPES OF OS

- **<u>TYPES OF OS</u>**

1. Interactive and batch systems
2. Mulitprogramming and uniprogramming systems

- ✓ **INTERACTIVE SYSTEM**: The user/programmer interacts directly with the computer, usually through a keyboard/display terminal, to request the execution of a job or to perform a transaction.
- ✓ **BATCH SYSTEM** : The user's program is batched together with programs from other users and submitted by a computer operator.
- ✓ **MULTIPROGRAMMING SYSTEM** : The processor works on more than one program at a time. The processor is kept as busy as possible. Several programs are loaded into memory and the processor switches rapidly among them.
- ✓ **UNIPROGRAMMING SYSTEM** : It works on only one program at a time.

Contd.

- **EARLY SYSTEMS**
- ✓ No OS.
- ✓ The programmer interacted directly with the system.
- ✓ Processors were run from a console (consisting of display light, toggle switches, input devices and printer).
- ✓ Programs in the processor code were loaded through the input devices.
- ✓ Error condition was indicated through lights.
- ✓ The programmer checks the registers and main memory to determine the cause of error.
- ✓ Normal completion of the program appeared on the printer.

Contd.

- ✓ **The Early systems presented two main problems:**

1. Scheduling
2. Setup time

- ✓ **SCHEDULING** : Most installations used a sign-up sheet to reserve processor time. A user could sign up for a block of time in multiples of half hour. A user might sign up for an hour and finish in 45 minutes resulting in computer idle time. Similarly the user might run into problems, not able to finish in the allotted time, and be forced to stop before resolving the problem.

- ✓ **SET UP TIME** : A single program called a job involves loading the compiler plus the HLL program (source program) into memory, saving the compiled program (object program), and loading and linking together the object program and common functions. If an error occurs then the user goes back to the beginning of the setup sequence. Thus a considerable amount of time was spent just in setting up the program to run.

Contd.

- **SIMPLE BATCH SYSTEMS**
- ✓ Early processors were expensive and hence we need to maximize the processor utilization. So
- ✓ Simple batch systems were developed to improve processor utilization.
- ✓ Also known as **MONITOR**.
- ✓ The user has no direct access to the processor.
- ✓ The user submits the job on cards or tape to a computer operator, who batches the jobs together sequentially and places the entire batch on an input device for use by the monitor.

Contd.

RESIDENT MONITOR

- ✓ The portion of monitor always present in the main memory and available for execution is known as resident monitor.
- ✓ The monitor reads in jobs one at a time from the input device.
- ✓ Now the current job is placed in the user program area and the control is passed to this.
- ✓ After completion of the job the control is returned to the monitor which reads the next job.
- ✓ The result of each job are printed out for delivery to the user.

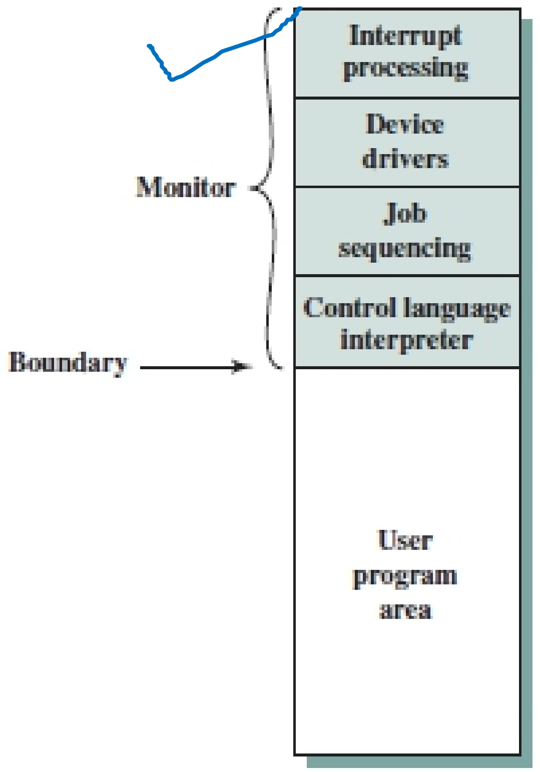


Fig.3: Memory Layout for a Resident Monitor [Source: Computer Organization and Architecture by William Stallings]

Contd.

- ✓ The monitor handles the scheduling problem as well as the job setup time.
- ✓ Each job instruction is included in a JCL(job control language) – a special type of programming language used to provide instructions to the monitor.
- ✓ Other desirable hardware features

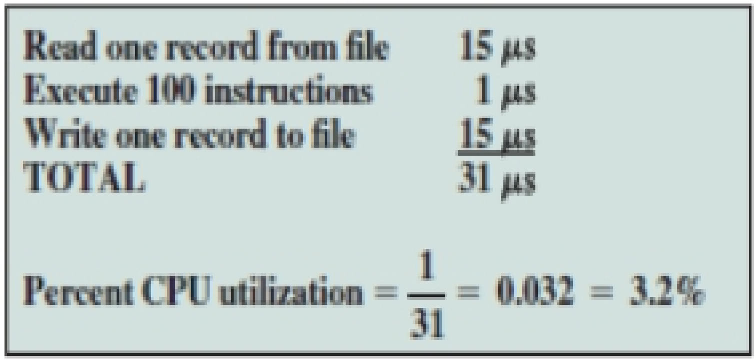
1. Memory protection
2. Timer
3. Privileged instruction
4. Interrupts

- ✓ Processor time alternates between execution of user programs and execution of the monitor.

Contd.

❖ MULTIPROGRAMMED BATCH SYSTEMS

- ✓ The processor utilization time increases.
- ✓ I/O devices are slower than the processor.
- ✓ E.g. A program processes a file of records and executes 100 processor instructions per record. Computer spends over 96% of its time waiting for the I/O devices to finish transferring data.



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

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

Fig.4: System utilization example [Source: Computer Organization and Architecture by William Stallings]

Contd.

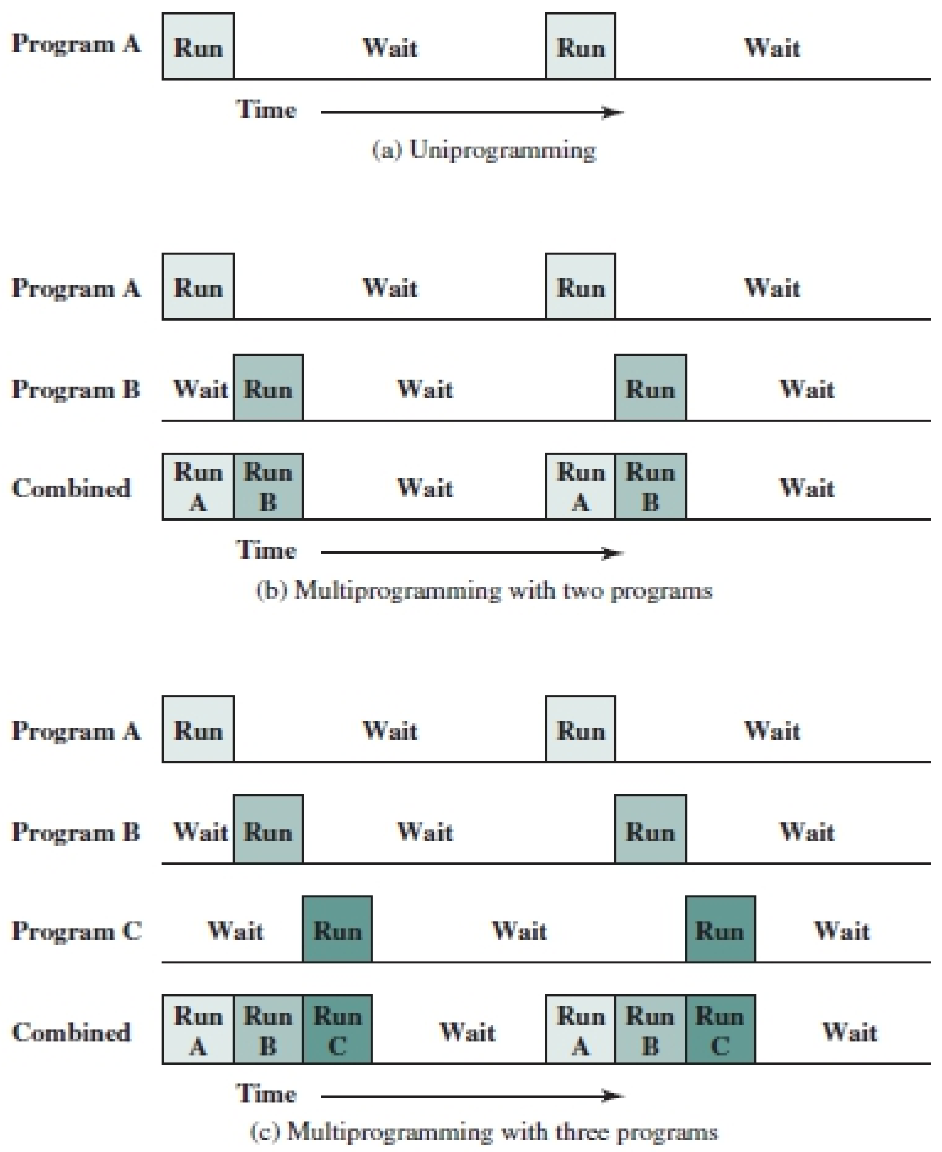


Fig.5: Multiprogramming example [Source: Computer Organization and Architecture by William Stallings]

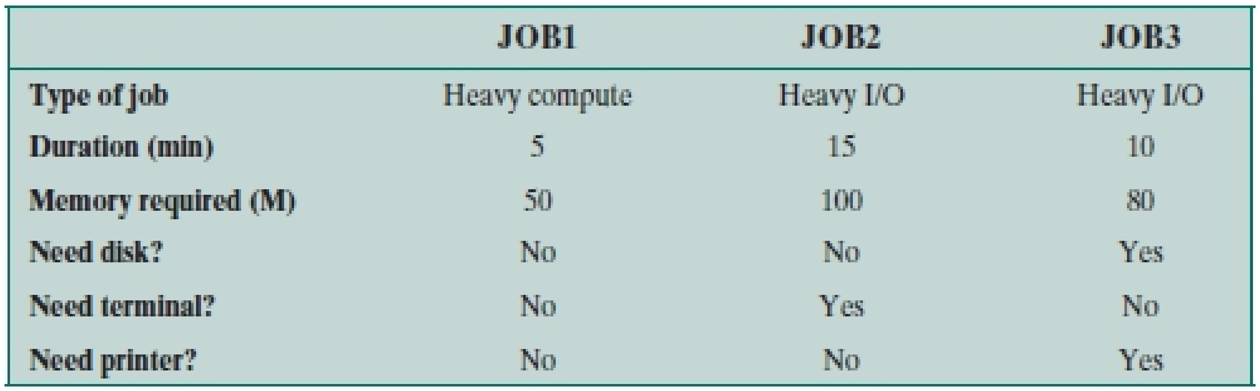
Contd.

EXAMPLE 1: this example illustrates the benefit of multiprogramming. Consider a computer with 250 Mbytes of available memory (not used by OS), a disk, a terminal, and a printer. Three programs JOB1, JOB 2, and JOB3 are submitted for execution at the same time with the attributes listed in table 1. We assume minimal processor requirements for JOB1 and JOB2 and continuous disk and printer use by JOB3. For a simple batch environment, these jobs will be executed in sequence. Thus JOB1 completes in 5 minutes. JOB2 must wait until the 5 minutes is over and then completes 15 minutes after that. JOB3 begins after 20 minutes and completes at 30 minutes form the time it was initially submitted. The average resource utilization, throughput and response times are given in uniprogramming column of table 2. device-by-device utilization is illustrated in figure6a. It is evident that there is gross underutilization for all resources when averaged over the required 30-minute time period.

Now suppose that the jobs are run concurrently under a multiprogramming OS. Because there is little resource contention between the jobs, all three can run in nearly minimum time while coexisting with the others in the computer (assuming that JOB2 and JOB3 are allotted enough processor time to keep their input and output operations active). JOB1 will still require 5 minutes to complete but at the end of that time, JOB2 will be one-third finished, and JOB3 will be half finished. All three jobs will have finished within 15 minutes. The improvement is evident when examining the multiprogramming column of table 2, obtained form the histogram of figure 6b.

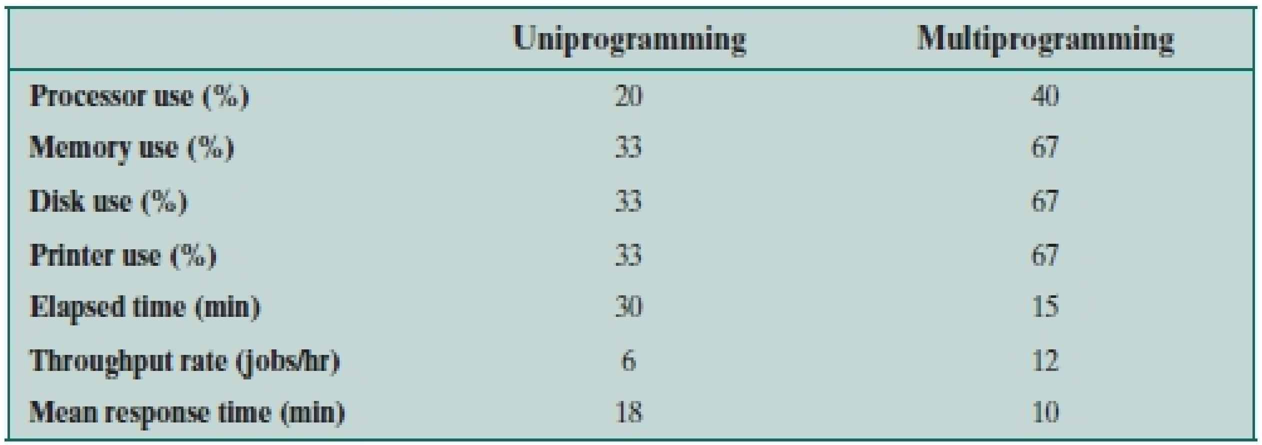
Contd.

Table .1 Sample Program Execution Attributes



	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration (min)	5	15	10
Memory required (M)	50	100	80
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Table 2 Effects of Multiprogramming on Resource Utilization



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

Contd.

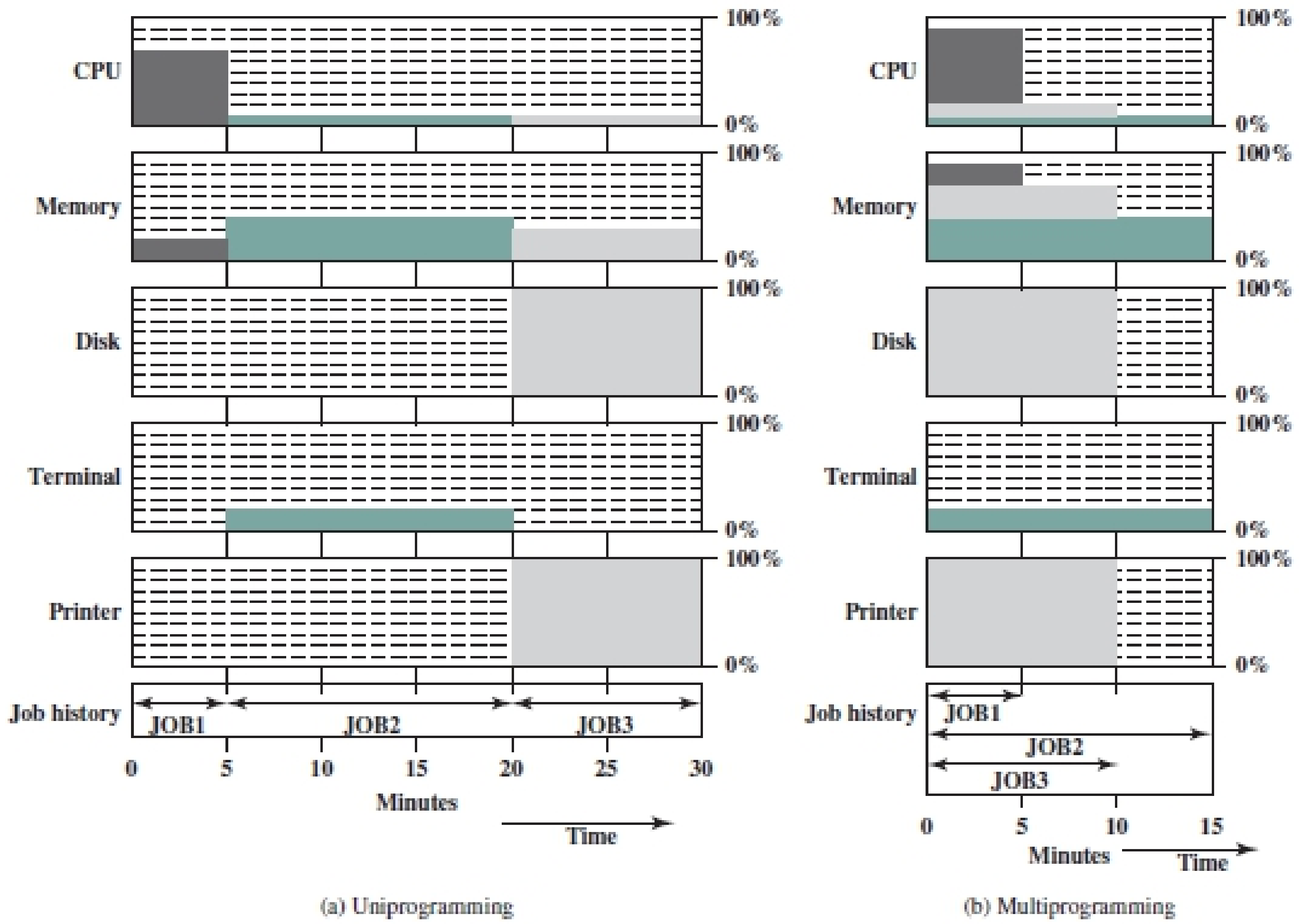
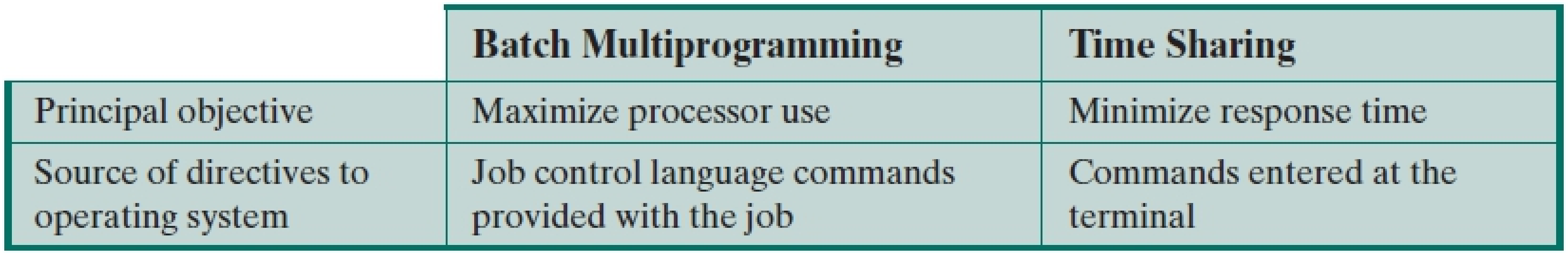


Fig.6: Utilization histograms [Source: Computer Organization and Architecture by William Stallings]

Contd.

❖ **TIME-SHARING SYSTEMS**

- ✓ Processor's time is shared among multiple users.
- ✓ Multiple users can simultaneously access the system through terminals.
- ✓ E.g. if there are n users actively requesting service at one time, each user will see on the average 1/n of the effective computer speed.



	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Table 3: Batch Multiprogramming versus Time Sharing [Source: Computer Organization and Architecture by William Stallings]

SCHEDULING

TYPES OF SCHEDULING

1. Long-term scheduling
2. Medium-term scheduling
3. Short-term scheduling
4. I/O scheduling

LONG-TERM SCHEDULING

✓ Scheduler determines the programs that are to be admitted to the system for processing.

✓ It controls the degree of multiprogramming.

Contd.

MEDIUM-TERM SCHEDULING

- ✓ It is a part of the swapping function.
- ✓ Swapping-in decision is based on the need to manage the degree of multiprogramming.

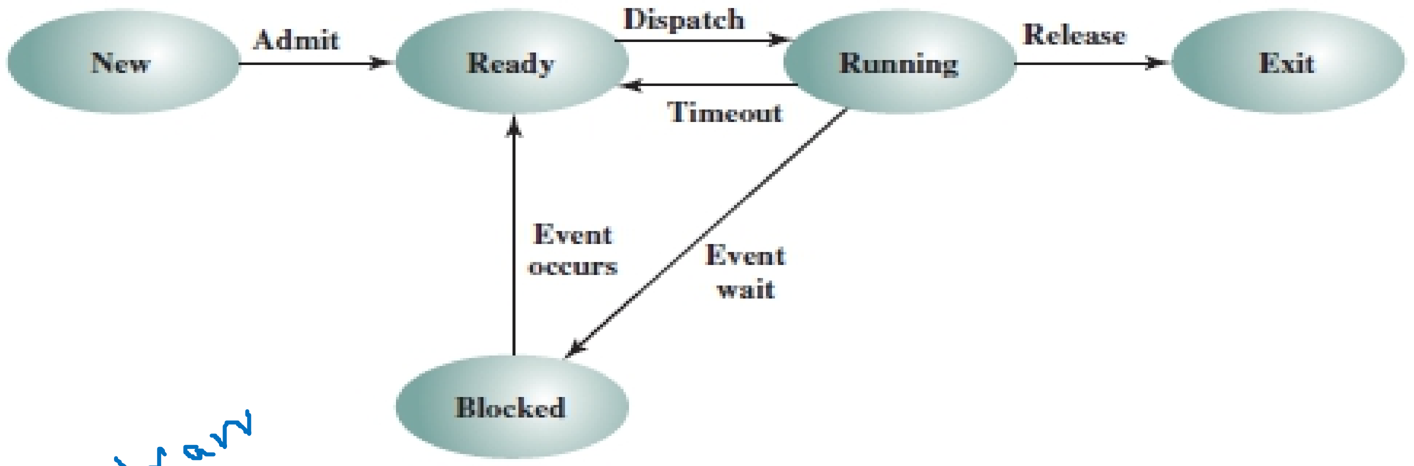
SHORT-TERM SCHEDULING

- ✓ Also known as dispatcher, executes frequently makes the fine-grained decision of which job to be executed next.

Contd.

PROCESS STATES

- ✓ During the life-time of a process its status changes a number of times.
- ✓ Its status at any point of time is known as a **state**.



draw

Fig.7: Five-State Process Model [Source: Computer Organization and Architecture by William Stallings]

Contd.

PROCESS CONTROL BLOCK

- ✓ For each process in the system, the OS must maintain information indicating the state of the process and for process execution.
- ✓ Each process is represented in the OS by a process control block.
- ✓ When the scheduler accepts a new job for execution it creates a blank process control block and places the associated process in the new state.

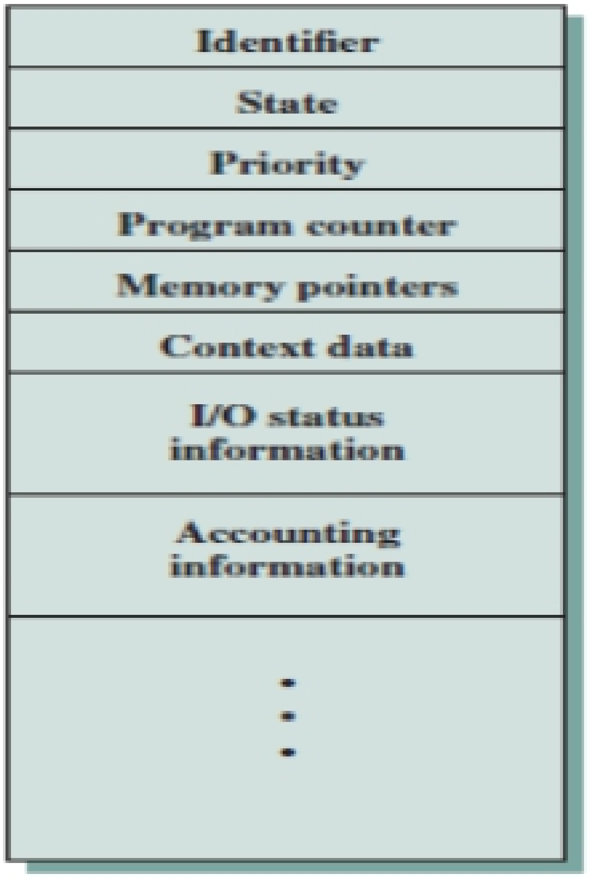


Fig.8: Process Control Block [Source: Computer Organization and Architecture by William Stallings]

Contd.

SCHEDULING TECHNIQUES

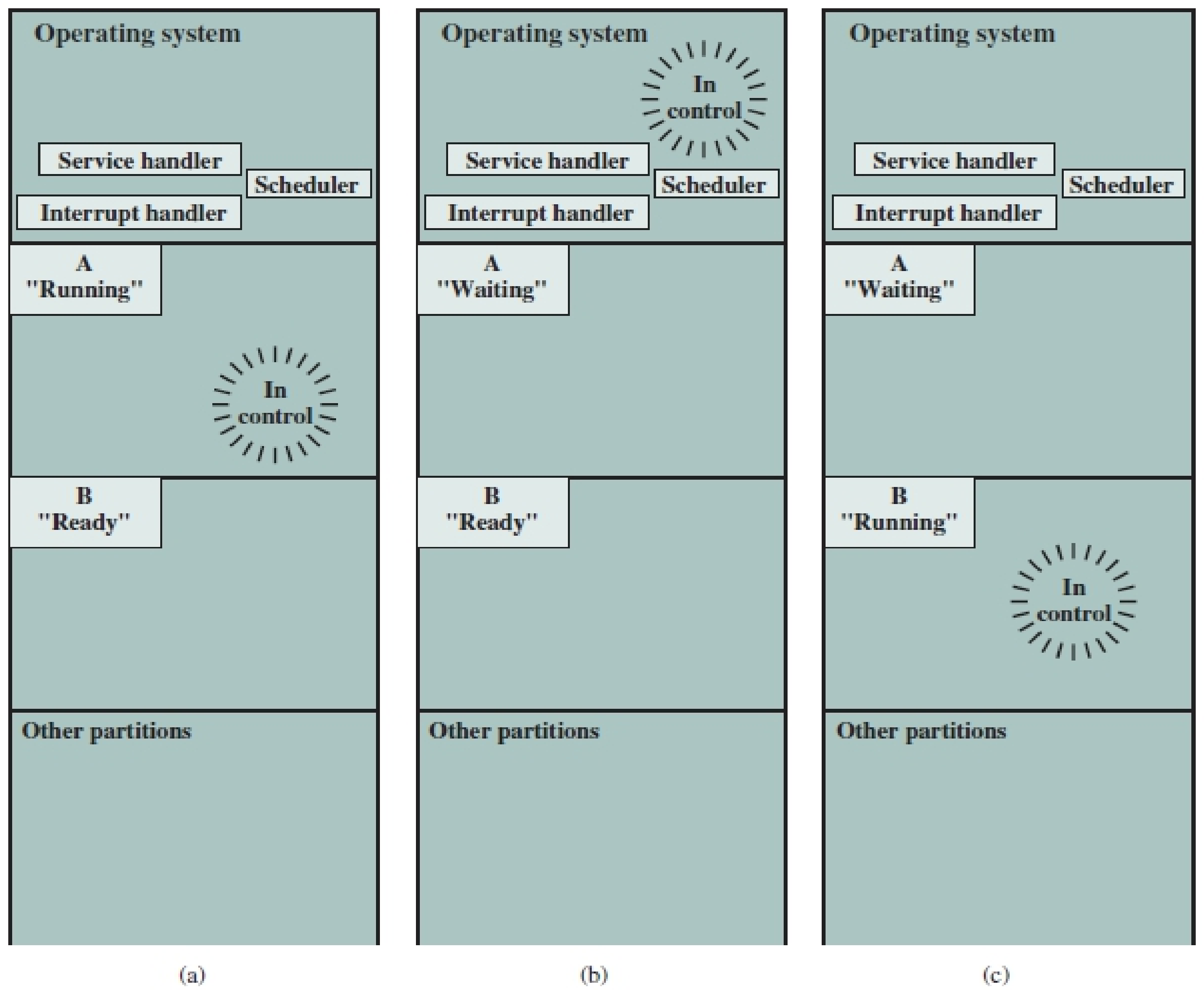


Fig.9: Scheduling Example [Source: Computer Organization and Architecture by William Stallings]

N/B ✓

QUESTIONS

1. What is an operating system?
2. List and briefly define the key services provided by an OS.
3. List and briefly define the major types of OS scheduling.

THANK YOU