UNIT-1

OPERATING SYSTEM OVERVIEW

Prof. Dipak Thanki
Marwadi Education Foundation Group Of Institutions, Rajkot

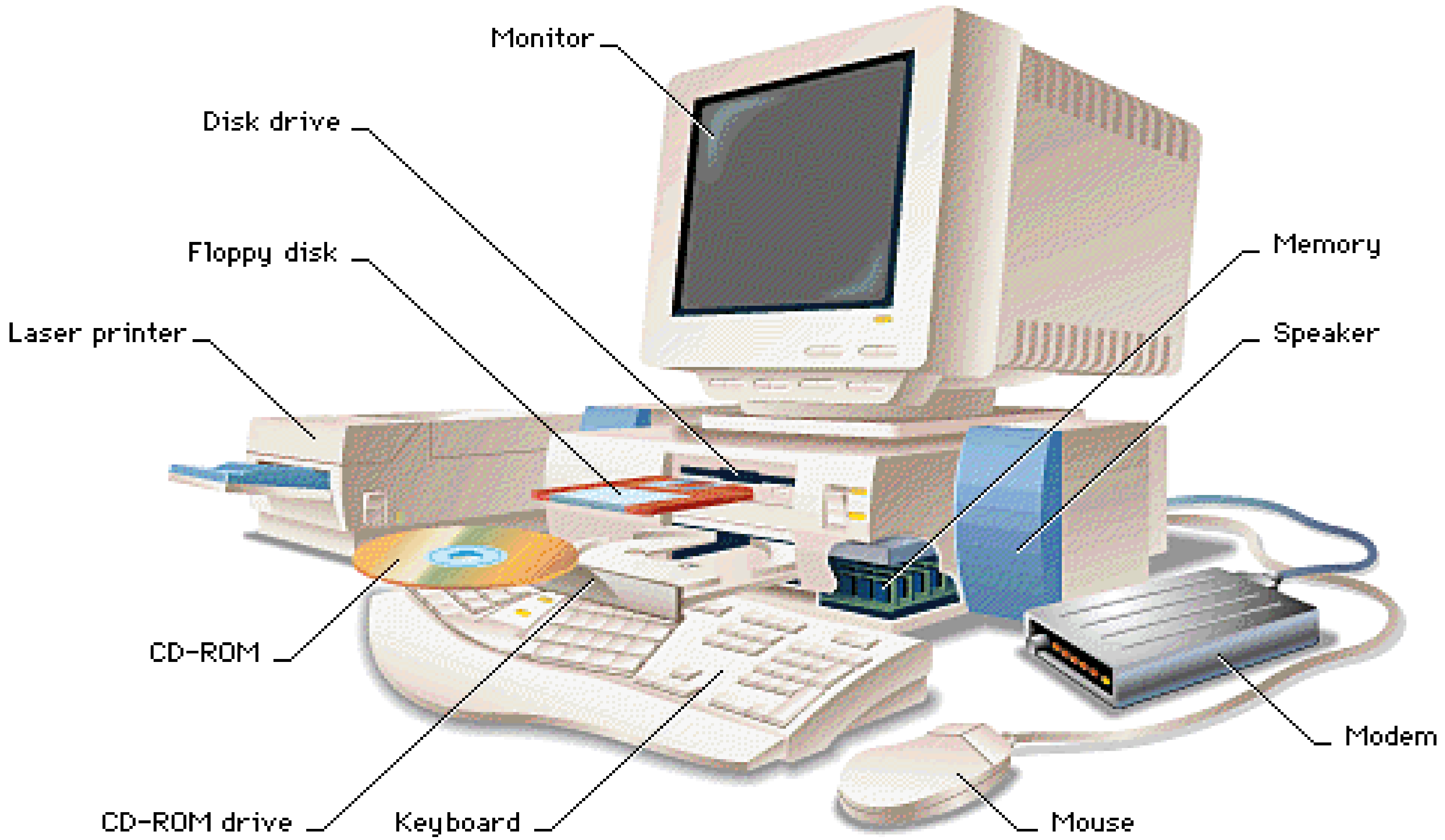
ROADMAP

- **Introduction to Operating system,**
- Operating System Objectives and functions,
- Evolution Of Operating System
- Major achievements

Operating System

- Definition:-
  - *" An Operating system is a set of programs that control the execution of application program and acts as an interface between application and computer hardware."*

- An OS is a program which acts as an *interface* between computer system users and the computer hardware.
- It provides a user-friendly environment in which a user may easily develop and execute programs.
- Otherwise, hardware knowledge would be mandatory for computer programming.
- So, it can be said that an OS hides the complexity of hardware from uninterested users.
- In general, a computer system has some resources which may be utilized to solve a problem. They are
  - *Memory*
  - *Processor(s)*
  - *I/O*
  - *File System*
  - *etc.*


Monitor
Disk drive
Floppy disk
Laser printer
Memory
Speaker
CD-ROM
Modem
CD-ROM drive
Keyboard
Mouse

processor

RAM

- The OS manages these resources and allocates them to specific programs and users.

- With the management of the OS, a programmer is rid of difficult hardware considerations.

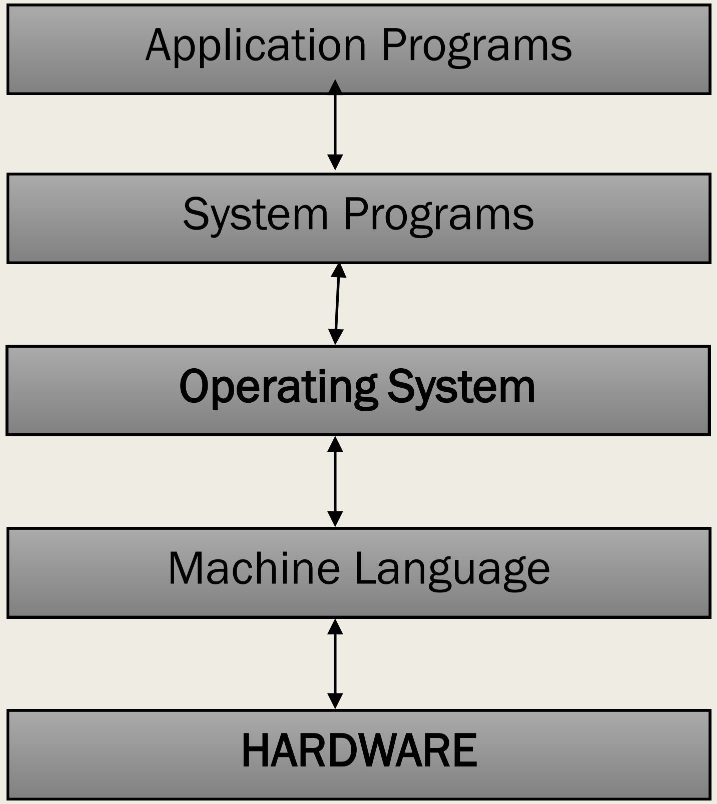
- An OS provides services for
  - *Processor Management*
  - *Memory Management*
  - *File Management*
  - *Device Management*
  - *Concurrency Control*

- Another aspect for the usage of OS is that; it is used as a *predefined library* for hardware-software interaction.

- This is why, system programs apply to the installed OS since they cannot reach hardware directly.

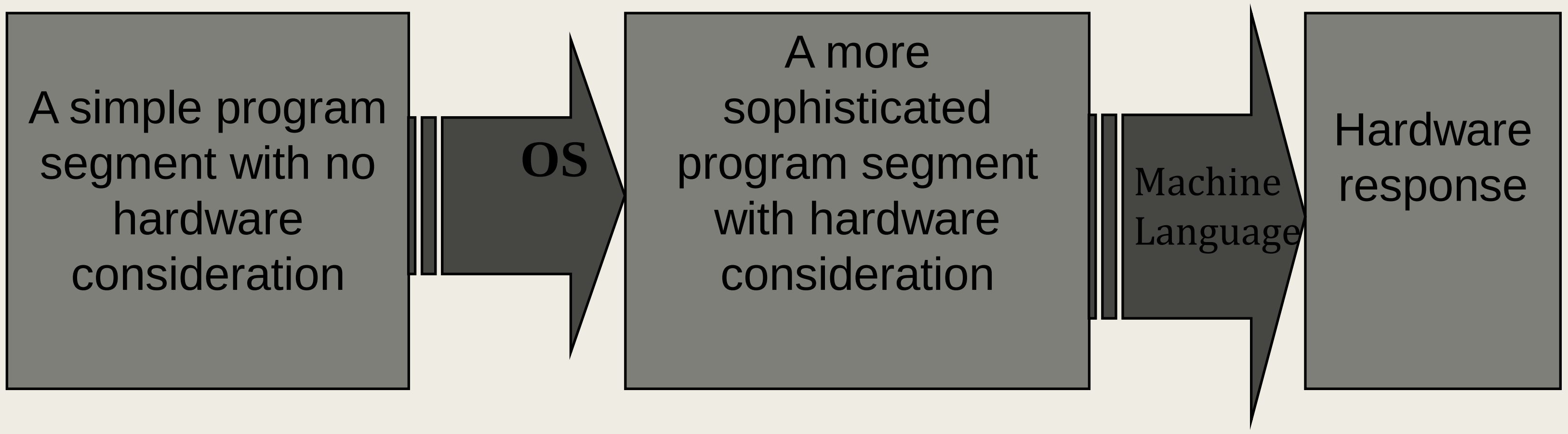
- Since we have an already written library, namely the OS, to add two numbers we simply write the following line to our program:

c = a + b ;



- In an OS installed machine, since we have an intermediate layer, our programs obtain *some advantage of mobility* by not dealing with hardware.
- For example, the above program segment would not work for an 8086 machine, where as the
- “c = a + b ;”
- syntax will be suitable for both.
- In a more simplistic approach, in fact, OS itself is a program.
- But it has a priority which application programs don’t have.
- OS uses the **kernel mode** of the microprocessor,
- whereas other programs use the **user mode**.
- The difference between two is that; all hardware instructions are valid in kernel mode, where some of them cannot be used in the user mode

PresenterMedia


A simple program segment with no hardware consideration
OS
A more sophisticated program segment with hardware consideration
Machine Language
Hardware response

ROADMAP

- Introduction to Operating system,
- **Operating System Objectives and functions,**
- Evolution Of Operating System
- Major achievements

Objectives of Operating System

- Convenience:-
  - *It makes a computer more convenient to use.*
  - *Tasks can be computed easily and in less time.*
- Efficiency:-
  - *Allows the computer system resources to be used in an efficient manner.*
- Ability to evolve:-
  - *Permits the effective development, testing and introduction of a new system functions without interfering with service.*

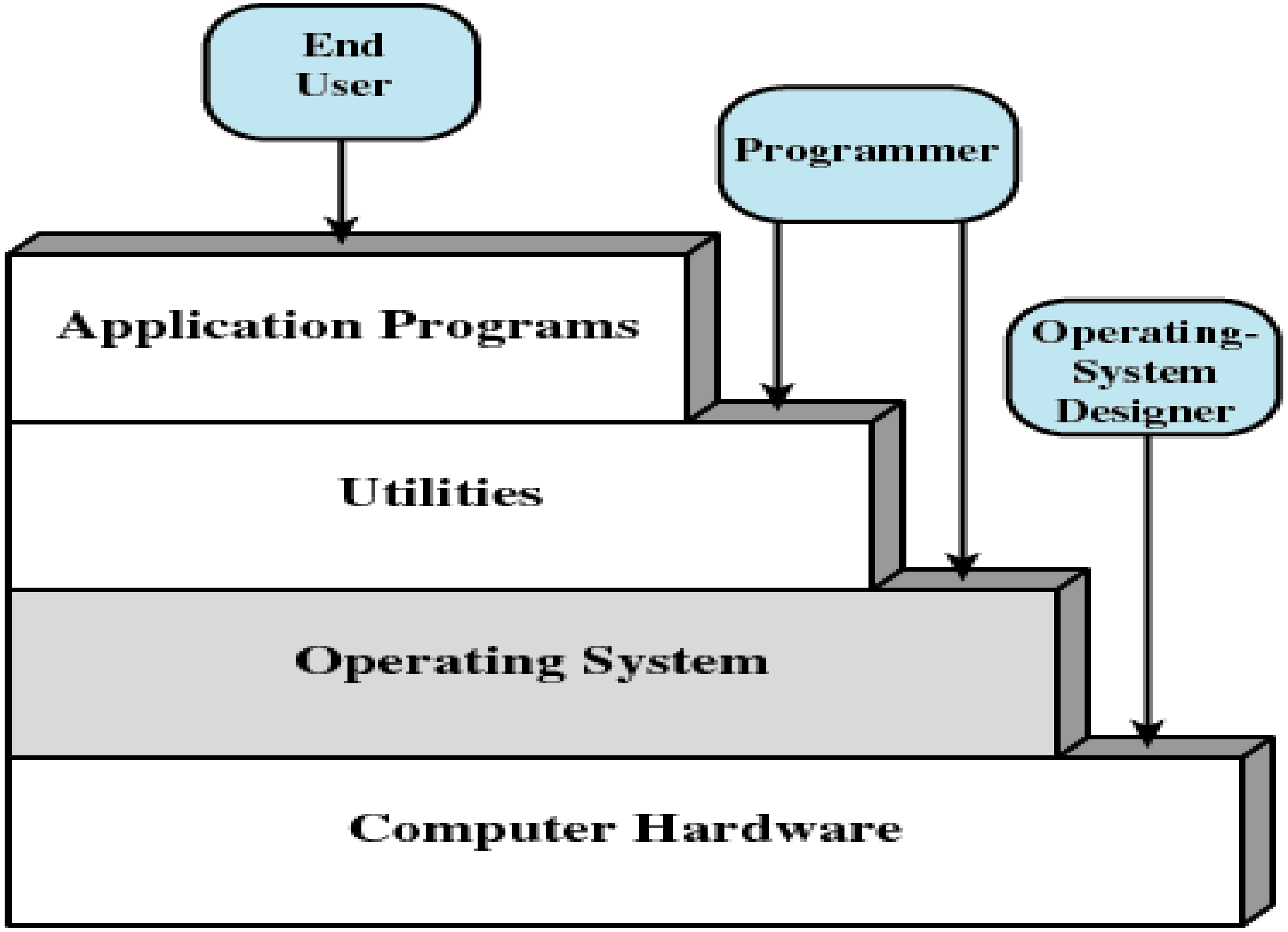


Figure 2.1 Layers and Views of a Computer System

- Let us examine these three aspects of an OS in turn

1) The Operating System as a User/Computer Interface (Convenience):-

- The end user views a computer system in terms of a set of applications.
- If an application program is to be built as a set of machine instructions than it will be more complex.
- To solve this, a set of system programs is provided.
- The OS hides the details of the hardware from the programmer and provides the programmer with a convenient interface for using the system.
- It acts as mediator, making it easier for the programmer and for application programs to access and use those facilities and services.

- Briefly, the OS typically provides services in the following areas:

- **Program development:-**
  - ✓ *It provides variety of facilities to assist the programmer E.g editors, debuggers*
  - ✓ *these services are in the form of utility programs that IS not strictly part of the OS AND supplied with the OS and are referred to as application Program development tools.*

- **Program Execution:-**
  - ✓ *A number of steps need to be performed to execute a program like Instructions and data must be loaded into main memory I/O devices and files must be initialized, and other resources must be prepared.*
  - ✓ *The OS handles these scheduling duties for the user.*

- **Access to I/O devices:-**
  - ✓ *Each I/O device requires its own peculiar set of instructions or control signals for operation.*
  - ✓ *The OS provides a uniform interface that hides these details*
  - – *so that programmers can access such devices using simple reads and writes..*
- **Controlled access to files:**
  - ✓ *For file access, the OS must reflect a detailed understanding of not only the*
  - – *nature of the I/O device (disk drive, tape drive) but also the structure of the*
  - – *data contained in the files on the storage medium.*
  - ✓ *In the case of a system with multiple users, the OS may provide protection*
  - – *mechanisms to control access to the files.*

- **System access:-**
  - ✓ *For shared or public systems, 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 like hardware*
  - – *errors(memory error, device failure) and various software errors(division by*
  - – *zero, attempt to access forbidden memory location).*
  - ✓ *The OS must provide a response that clears the error condition with the least*
  - – *impact on running applications.*
  - ✓ *Like ending the program that caused the error, to retrying the operation, to*
  - – *simply reporting the error to the application.*

- **<u>Accounting:</u>-**
  - ✓ ***A good OS will collect usage statistics for various resources and*** *monitor*
  - – *performance parameters such as response time.*
  - ✓ *On any system, this information is useful in anticipating the need for future*
  - – *enhancements and in tuning the system to improve performance.*
  - ✓ *On a multiuser system, the information can be used for*
  - – *billing purposes.*

- **2) The Operating System as Resource Manager (**Efficiency**)**
  - *A computer is a set of resources for the movement, storage, and processing of data and for the control of these functions.*
  - *The OS is responsible for managing these resources.*
  - *OS controls in two respects:*
    - The OS functions in the same way as ordinary computer software; that is, it is a program or suite of programs executed by the processor.
    - The OS frequently relinquishes control and must depend on the processor to allow it to regain control.
  - *The OS directs the processor in the use of the other system resources and in the timing of its execution of other programs.*

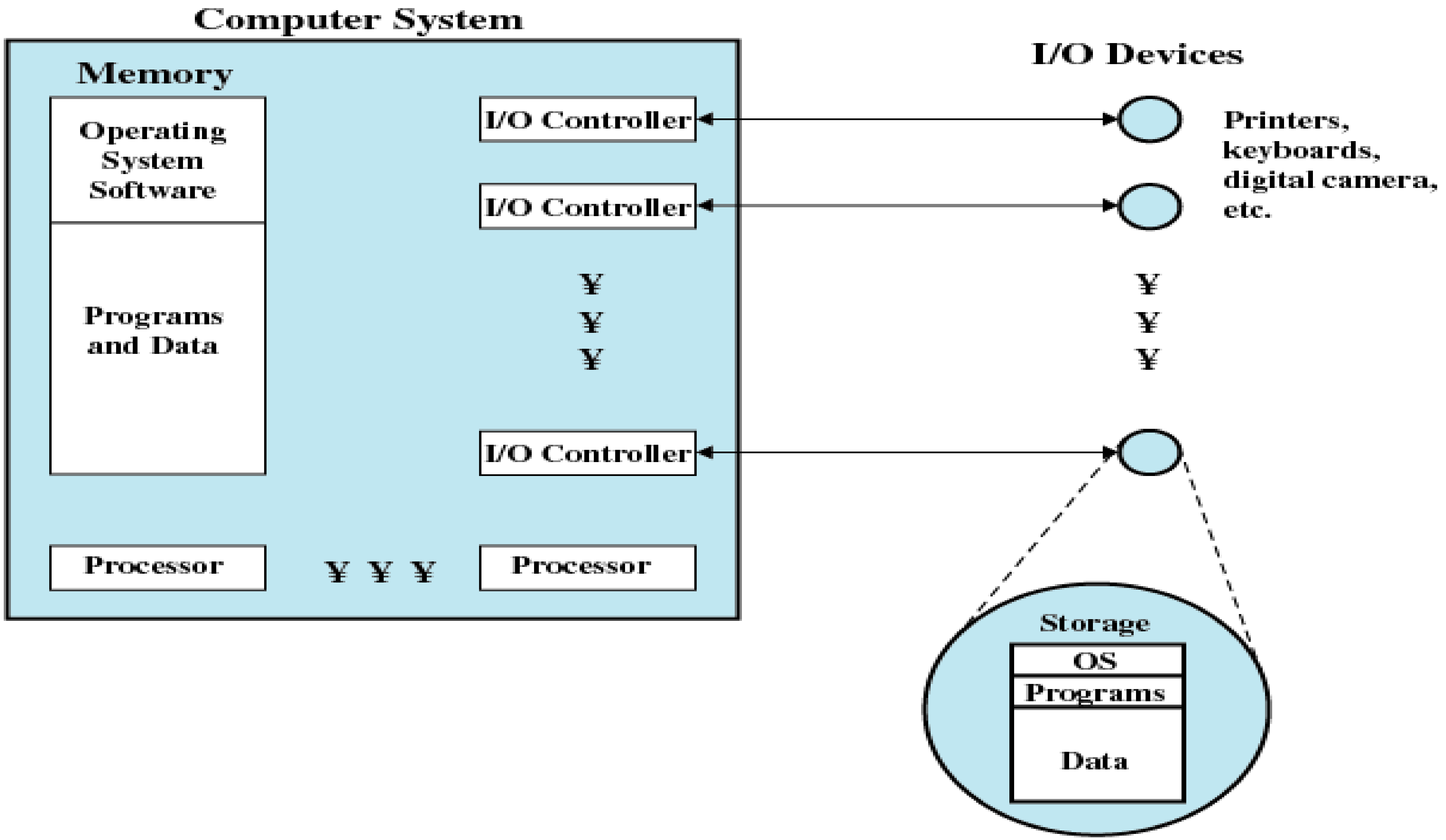


Figure 2.2 The Operating System as Resource Manager

- Figure 2.2 suggests the main resources that are managed by the OS
- A portion of the OS is in main memory.
- This includes the **kernel, or nucleus, which contains the** most frequently used functions in the OS and, at a given time, other portions of the OS currently in use.
- The remainder of main memory contains user programs and data.
- The allocation of this resource (main memory) is controlled jointly by the OS and memory management hardware in the processor.
- The OS decides when an I/O device can be used by a program in execution and controls access to and use of files.
- The processor itself is a resource, and the OS must determine how much processor time is to be devoted to the execution of a particular user program.

Operating System Evolution

- A major operating system will evolve over time for a number of reasons:-
  - ***Hardware upgrades plus new types of hardware:-***
    - In Early operating systems paging mechanism as well as graphics terminal and page mode terminals was not there.
    - In newer o.s this facility was introduced for e.g. a graphics terminal typically allows the user to view several applications at the same time through "windows" on the screen. This requires more sophisticated support in the OS.
  - ***New services:***
    - In response to user demand or in response to the needs of system managers, the OS expands to offer new services.
  - ***Fixes:***
    - Any OS has faults. These are discovered over the course of time and fixes are made.

ROADMAP

- Introduction to Operating system,
- Operating System Objectives and functions,
- **Evolution Of Operating System**
- Major achievements

EVOLUTION OF OPERATING SYSTEM

- The following sequence shows how operating systems have evolved over the years.

(1) Serial Processing

(2) Simple Batch System

(3) Multiprogrammed Batch Systems

(4) Time-sharing Systems

1. Serial Processing

- Definition:-

  Processing information where only the one process of operation is carried out at one time.

- This mode of operation could be termed *serial processing, reflecting the fact t*hat users have access to the computer in series.
  - *No operating system*
  - *Machines run from a console with display lights, toggle switches, input device, and printer*
  - *Schedule time*
  - *Setup included loading the compiler, source program, saving compiled program, and loading and linking*
- As time went on, card readers, printers, and magnetic tape units were developed as additional hardware elements.
- Assemblers, loaders and simple utility libraries were developed as software tools.

1. Serial Processing

- Example of serial processing is ENIAC. It all started with computer hardware in about 1940s

ENIAC

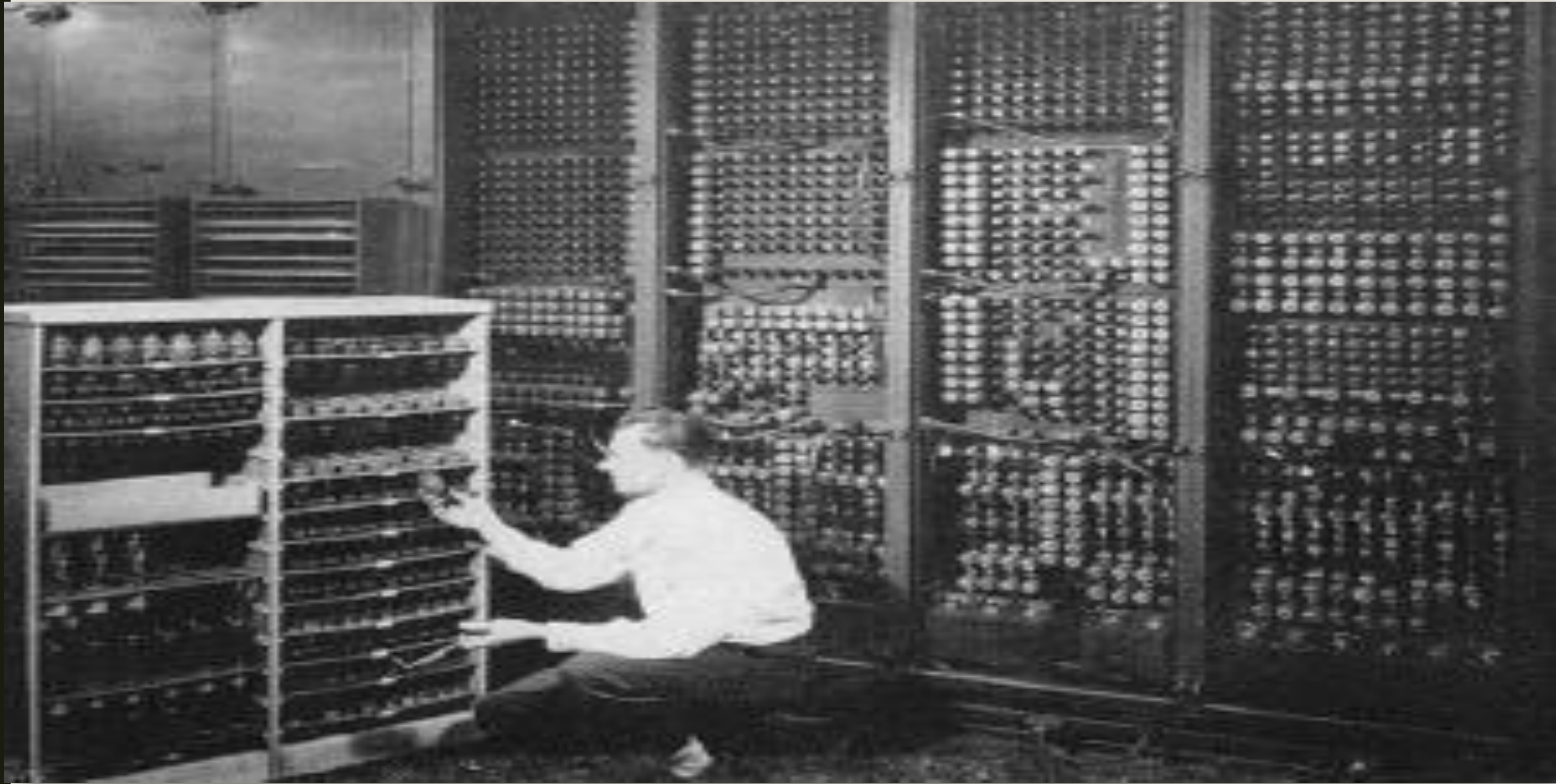
- ENIAC (Electronic Numerical Integrator and Computer), at the U.S. Army's Aberdeen Proving Ground in Maryland.
  - *Built in the 1940s,*
  - *Weighed 30 tons,*
  - *Was eight feet high, three feet deep, and 100 feet long*
  - *Contained over 18,000 vacuum tubes that were cooled by 80 air blowers.*

1. Serial Processing

- Computers were using vacuum tube technology

ENIAC’s vacuum tubes

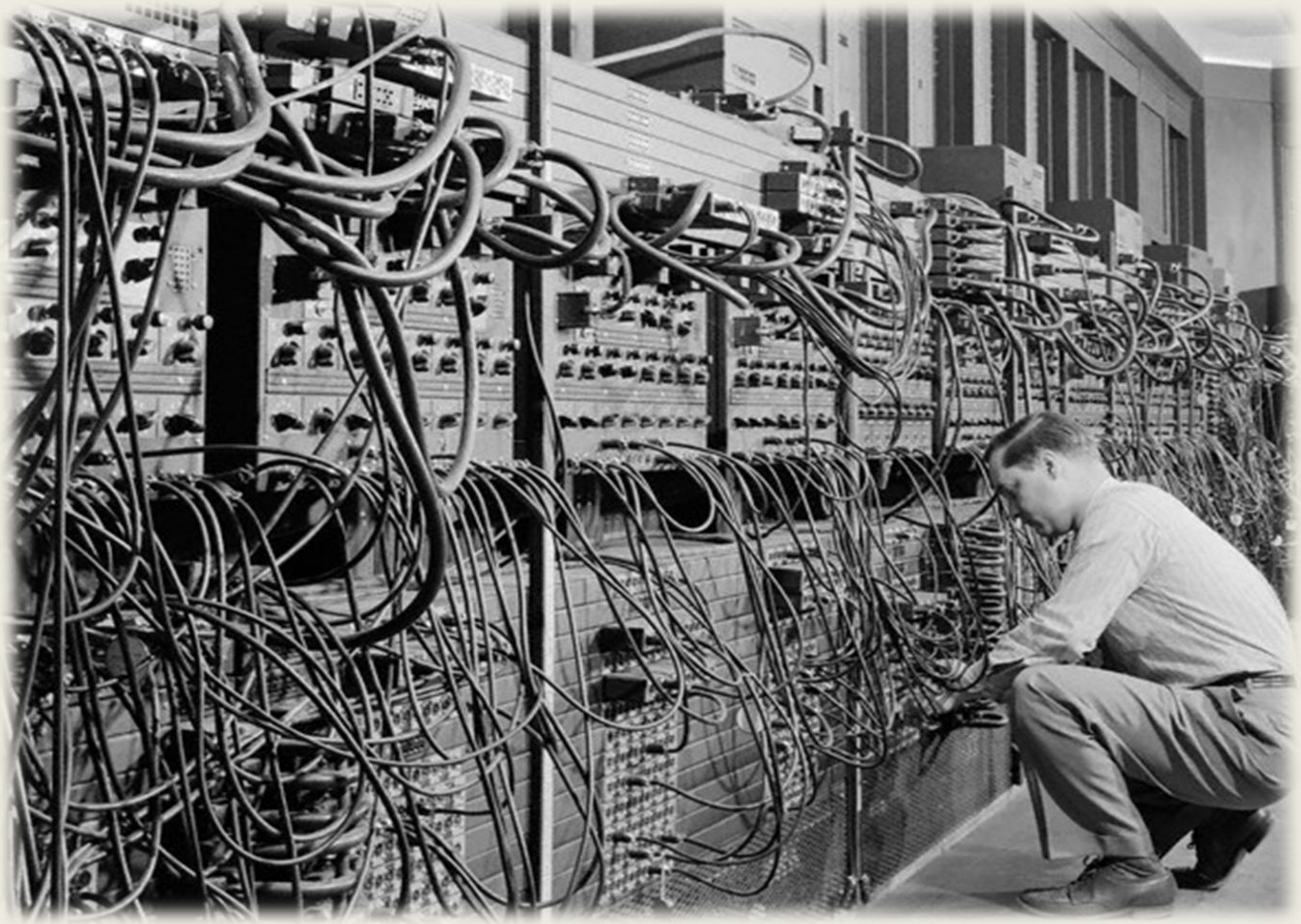
1. Serial Processing



ENIAC’s backside

F. Serial Processing

- Programs were loaded into memory manually using switches, punched cards, or paper tapes.



ENIAC : coding by cable connections

1. Serial Processing

- These early systems presented two main problems:

1) Scheduling:

- *Most installations used a hardcopy sign-up sheet to reserve computer time.*
- *Typically, a user could sign up for a block of time in multiples of a half hour or so.*
- *A user might sign up for an hour and finish in 45 minutes; this would result in wasted computer processing time.*

***2) Setup time:**-*

- *A single program (job) requires so many steps and if any error occurs, user typically had to go back to the beginning of the setup sequence.*
- *Thus, a considerable amount of time was spent just in setting up the program to run.*

2. Simple Batch System

- To minimize the problem of serial processing and to improve utilization, the concept of a batch operating system was developed.
- **Def :-** A batch system is a when a computer is programmed to batch together a number of transactions for processing at a specific time.
- The operator batched similar jobs together and then ran in the computer to speed up the processing.
- 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.
- **What is a MONITOR?**
    - It is the batch operating system.
    - Software that controls the sequence of events
    - Batch jobs together
    - Program branches back to monitor when finished

2. Simple Batch System

- With each job, instructions are included in a primitive form of **job control language (JCL).**
- Special type of programming language Provides instruction to the **monitor**
  - *What compiler to use*
  - *What data to use*

2. Simple Batch System

- To understand how this scheme works, let us look at it from two points of view:
- **Monitor point of view:**
  - *The monitor controls the sequence of events.*
  - *For this much of the monitor must always be in main memory for execution (Fig 2.3) and that portion is referred to as the* ***resident monitor.***
  - *The monitor reads in jobs one at a time from the input device*
    - As it is read in, the current job is placed in the user program area, and control is passed to this job.
    - When the job is completed, it returns control to the monitor, which immediately reads in the next job.
    - The results of each job are sent to an output device, such as a printer, for delivery to the user.

2. Simple Batch System

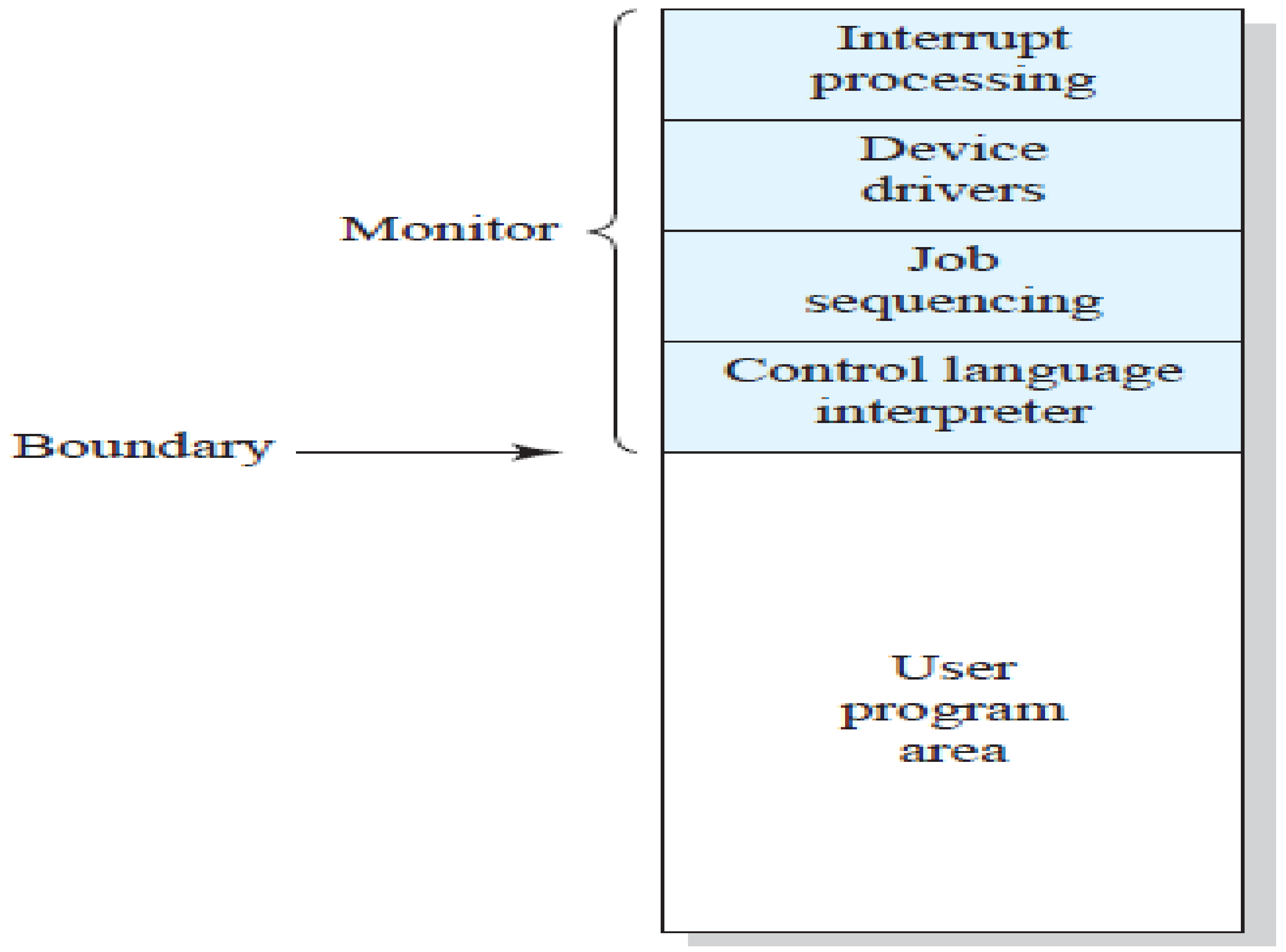


Figure 2.3 **Memory Layout for a Resident Monitor**

SIMPLE BATCH SYSTEM

- **Processor point of view:**
  - *Once a job has been read in, the processor will encounter a branch instruction in the monitor that instructs the processor to continue execution at the start of the user program.*
  - *The processor will then execute the instructions in the user program until it encounters an ending or error condition.*
  - *The monitor performs a scheduling function:*
    - A batch of jobs is queued up, and jobs are executed as rapidly as possible, with no intervening idle time.

Simple Batch System

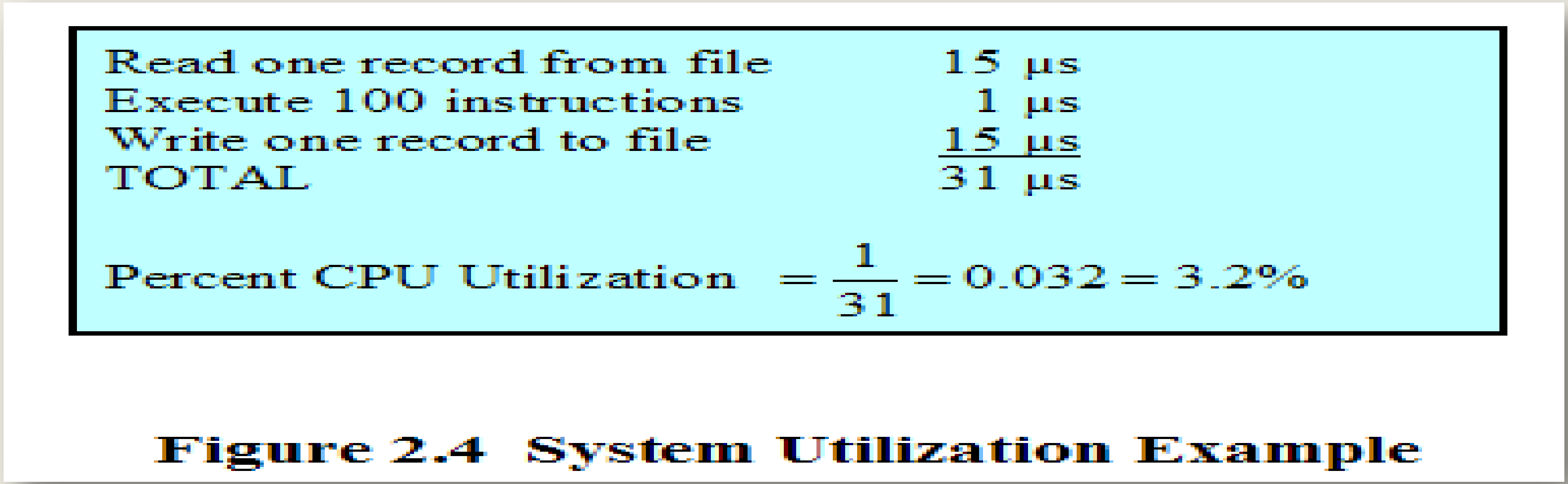
- Certain other hardware features are also desirable:
    - Memory protection
        - *Do not allow the memory area containing the monitor to be altered*
        - *by the user program*
    - Timer
        - *Prevents a job from monopolizing the system*
        - *The timer is set at the beginning of each job. If the timer expires, the*
        - *user program is stopped, and control returns to the monitor.*
    - Privileged instructions
        - *Certain machine level instructions can only be executed by the*
        - *monitor otherwise an error occurs causing control to be transferred*
        - *to the monitor.*
  - *Interrupts*
        - *Early computer models did not have this capability.*

        - *This feature gives the OS more flexibility in relinquishing control to and regaining control from user programs.*

SIMPLE BATCH SYSTEM

- User program executes in user mode
    - *Certain instructions may not be executed*
- Monitor executes in system mode
    - *Kernel mode*
    - *Privileged instructions are executed*
    - *Protected areas of memory may be accessed*
- With a batch operating system, processor time alternates between execution of user programs and execution of the monitor.
- **There have been two sacrifices:**
    - *Some main memory is now given over to the monitor and*
    - *some processor time is consumed by the monitor.*
    - *Despite this the simple batch system improves utilization of the computer.*

- Even with the automatic job sequencing provided by a simple batch operating system, the processor is often idle.
- The problem is that I/O devices are slow compared to the processor

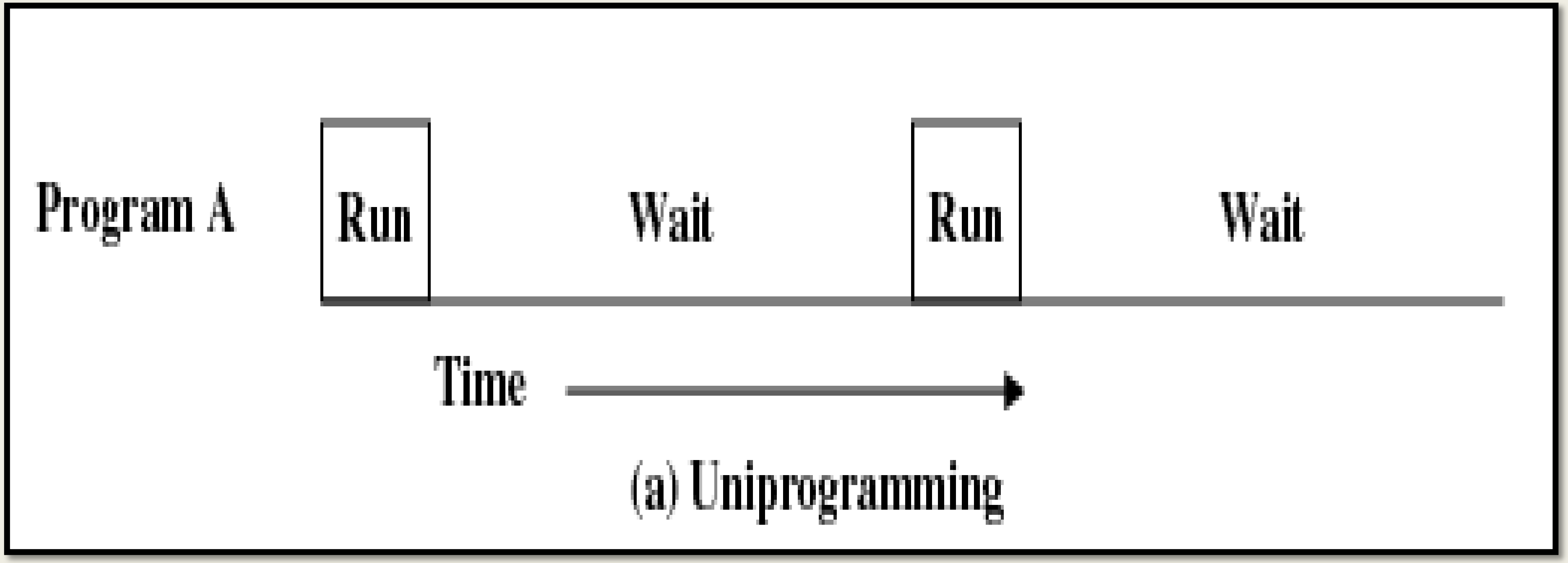


Read one record from file	15 µs
Execute 100 instructions	1 µs
Write one record to file	15 µs
TOTAL	31 µs

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

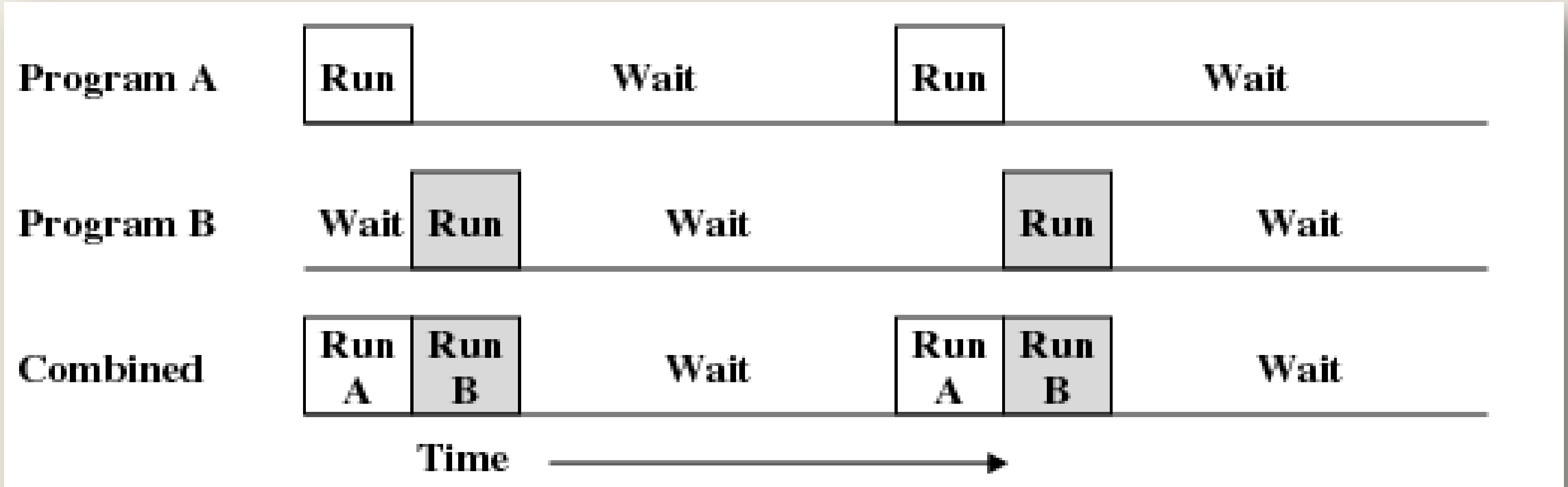
Figure 2.4 System Utilization Example

- **<u>Uniprogramming</u>:-**
  - ➢ *In a single application system, uniprogramming, the processor spends a certain amount of time executing, until it reaches an I/O instruction.*
  - ➢ *It must then wait until that I/O instruction concludes before proceeding.*
  - ➢ *This inefficiency is not necessary*



Multiprogrammed Batch System

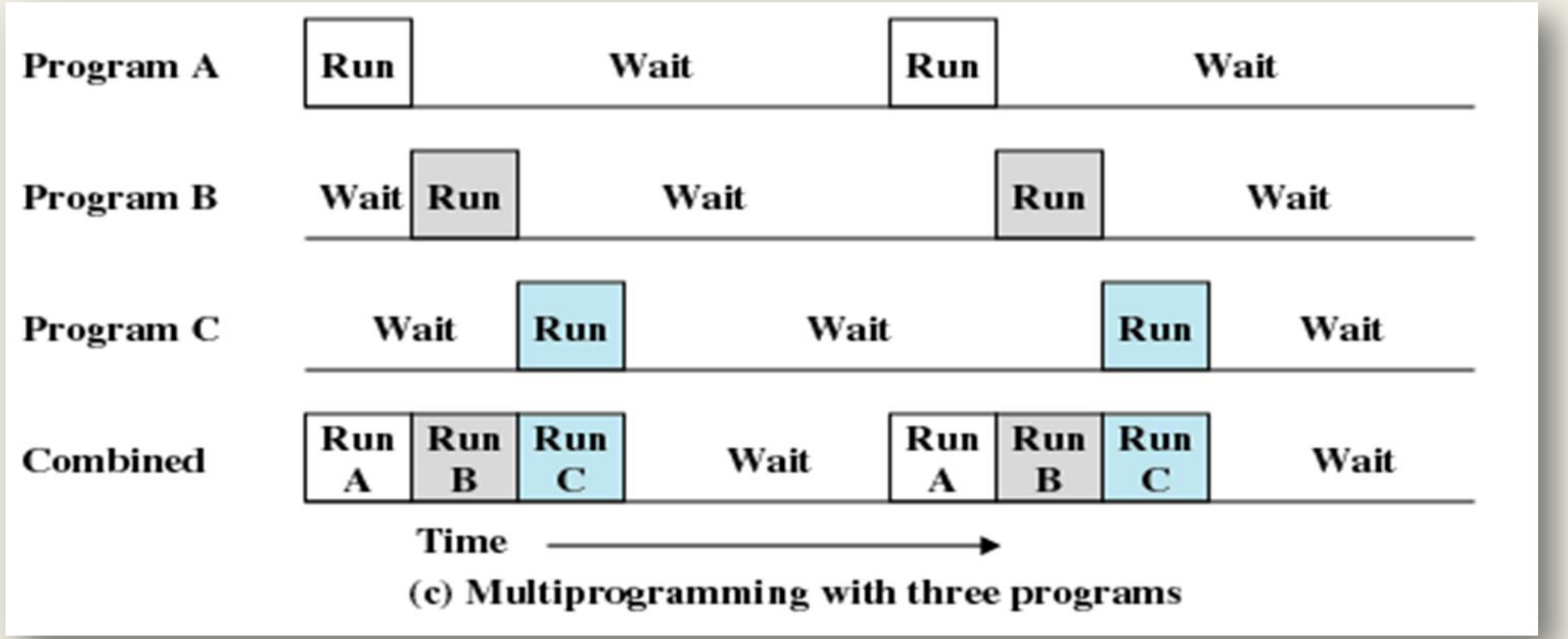
- **<u>Multiprogramming:-</u>**
  - *We know that there must be enough memory to hold the OS (resident monitor) and one user program.*
  - *Suppose that there is room for the OS and two user programs.*
  - *When one job needs to wait for I/O, the processor can switch to the other job, which is likely not waiting for I/O (Figure 2.5b).*



(b) Multiprogramming with two programs

MULTIPROGRAMMED BATCH SYSTEM

- **Multiprogramming with three programs**:-
- Further-more, we might expand memory to hold three, four, or more programs and switch among all of them (Figure 2.5c).
- The approach is known as multiprogramming, or multitasking.
- It is the central theme of modern operating systems.



(c) Multiprogramming with three programs

Multiprogrammed Batch System

- To illustrate the benefit of multiprogramming, we give a simple example.
- Consider a computer with 250 Mbytes of available memory (not used by the OS), a

  disk, a terminal, and a printer.
- Three programs, JOB1, JOB2, and JOB3, are submitted for execution at the same time, with the attributes listed in Table 2.1.

MULTIPROGRAMMED BATCH SYSTEM

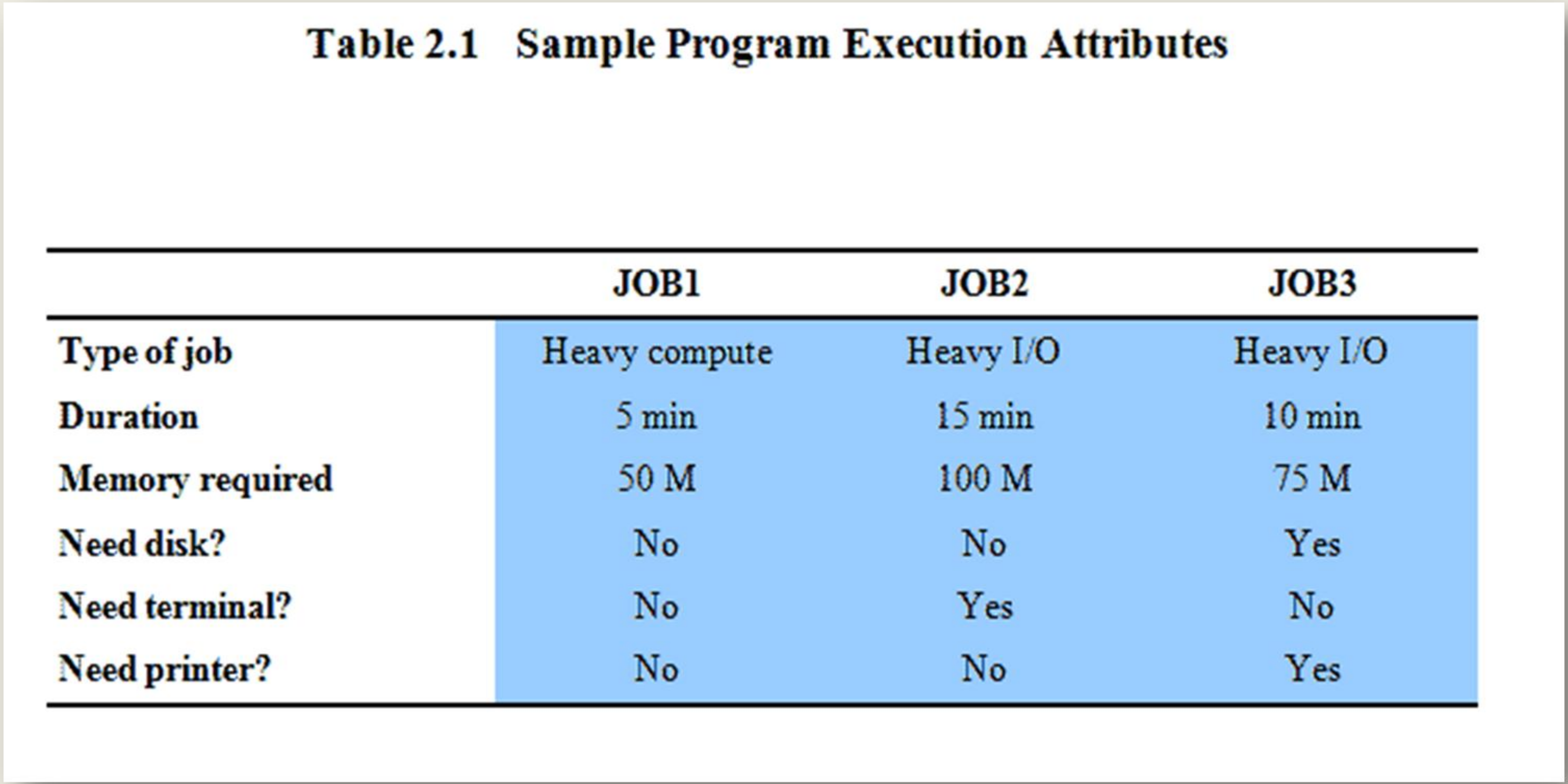


Table 2.1 Sample Program Execution Attributes

	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

MULTIPROGRAMMED BATCH SYSTEM

- **Utilization Histograms:**
- Device-by-device utilization of the previous example is illustrated in Figure 2.6a.
- 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 operating system.
- 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
- JOB1 will still require 5 minutes to complete, but at the end of that time,
- JOB2 will be one-third finished and JOB3 half finished.
- All three jobs will have finished within 15 minutes
- The improvement is evident when examining the multiprogramming column of Table 2.2, obtained from the histogram shown in Figure 2.6b.

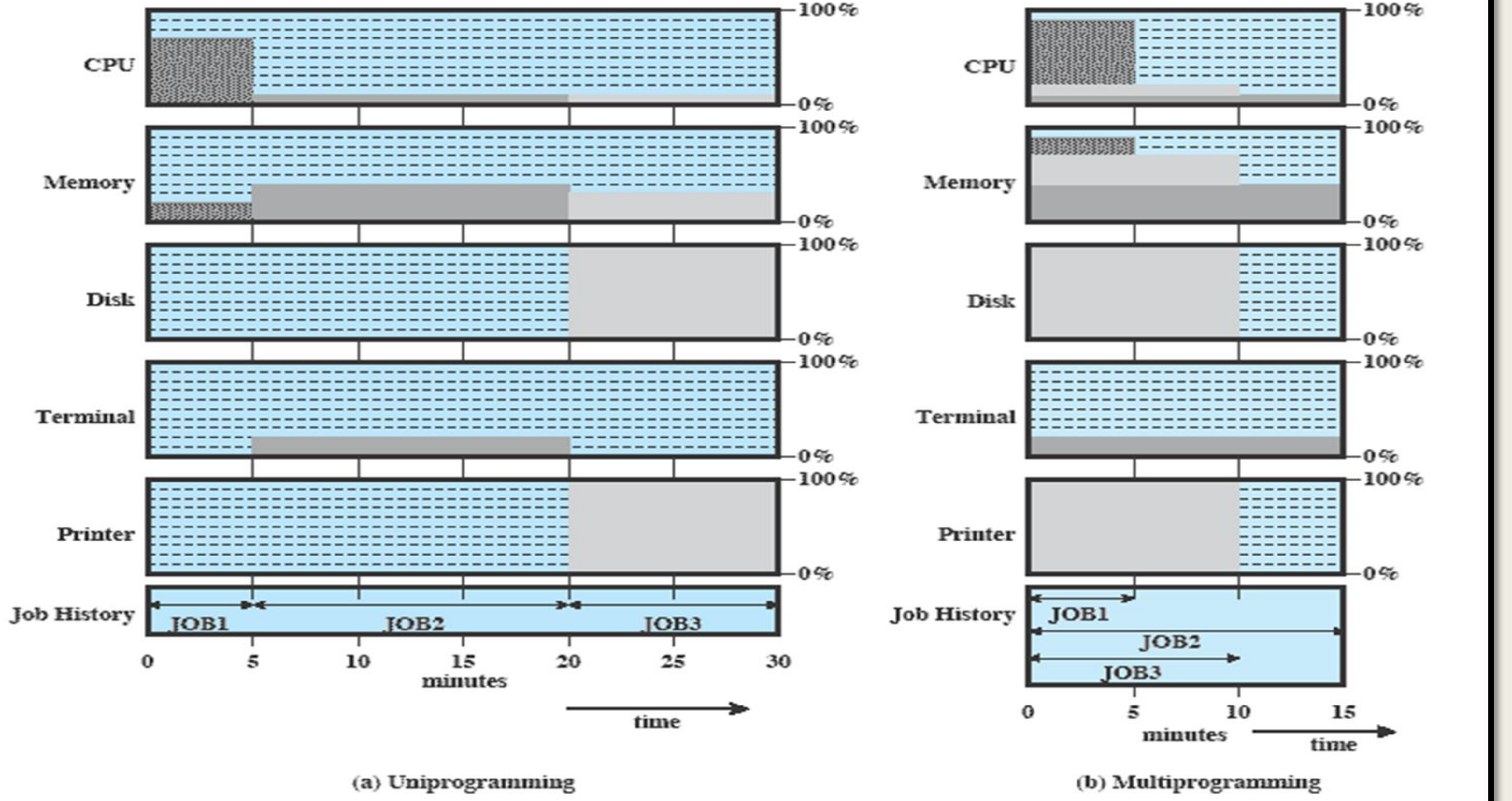


Figure 2.6 Utilization Histograms

Multiprogrammed Batch System

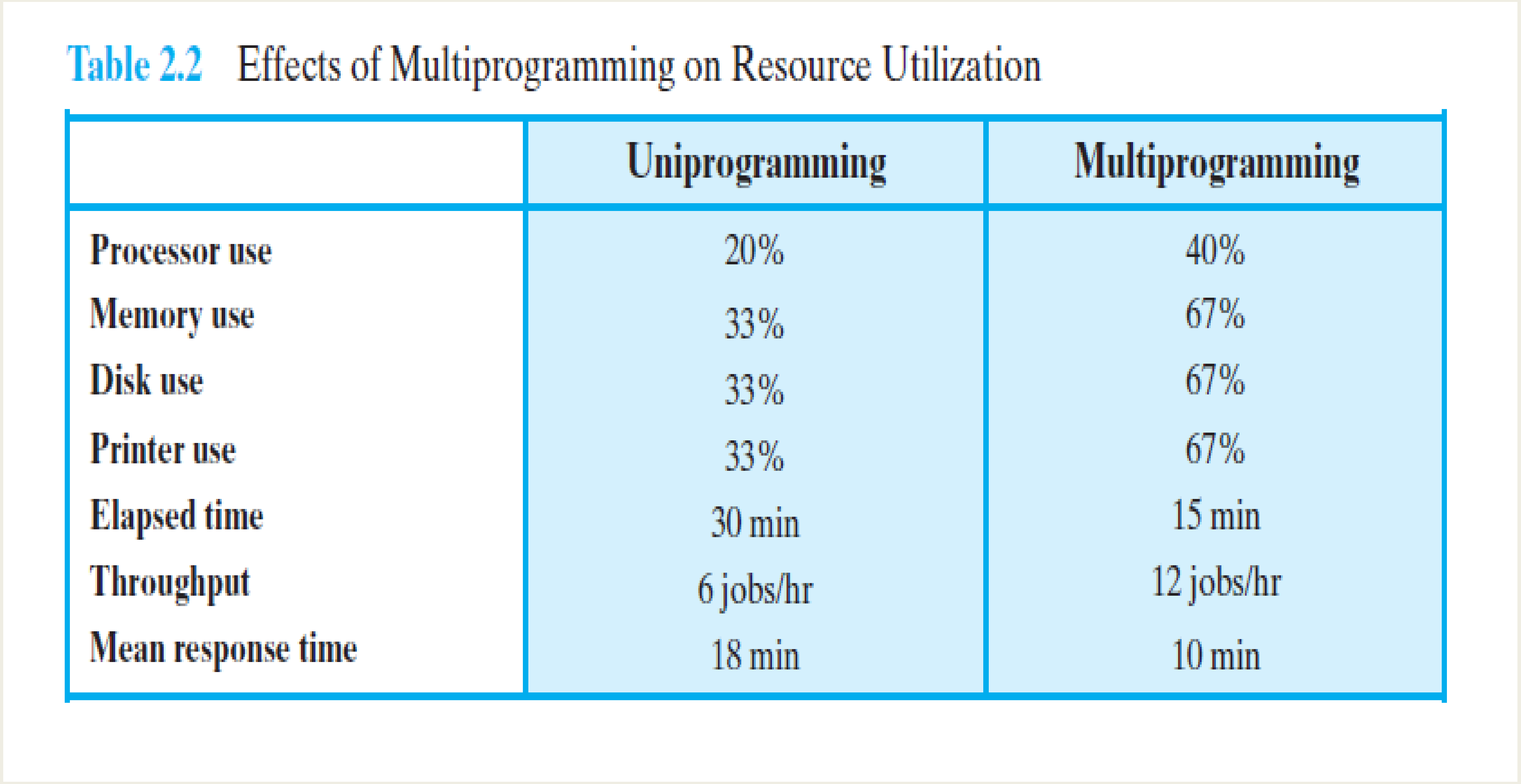


Table 2.2 Effects of Multiprogramming on Resource Utilization

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

TIME SHARING SYSTEM

- With the use of multiprogramming, batch processing can be quite efficient.
- For many jobs, it is desirable to provide a mode in which the user interacts directly with the computer.
- Today, this requirement met by the use of a dedicated personal computer.
- That option was not available in the 1960s, when most computers were big and costly.
- So Time sharing was developed.
- Multiprogramming can also be used to handle multiple interactive jobs.
- This technique is referred to as **time sharing**, because processor time is shared among multiple users.

Time Sharing System

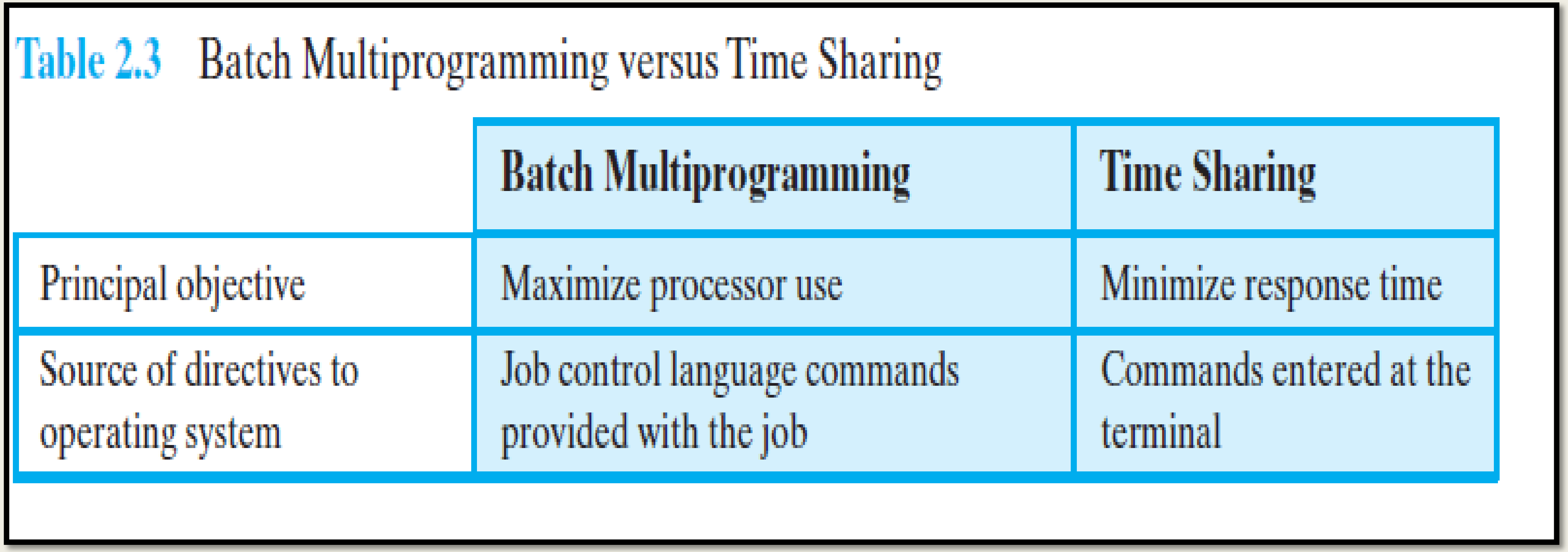


Table 2.3 Batch Multiprogramming versus Time Sharing

	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

TIME SHARING SYSTEM

- Programs queued for execution in FIFO order.
- Like multiprogramming, but timer device interrupts after a quantum (time slice).
    - Interrupted program is returned to end of FIFO
    - Next program is taken from head of FIFO
- Control card interpreter replaced by command language interpreter.
- When OS finishes execution of one command, it seeks the next control statement from user.

Example Time Sharing System

- Early Example: CTSS
- • Compatible Time-Sharing System (CTSS)
  - ✓ *Developed at MIT as project MAC*
- • Time Slicing:
- – When control was passed to a user
- – User program and data loaded
- – Clock generates interrupts about every 0.2 sec
- – At each interrupt OS gained control and could assign processor to another user

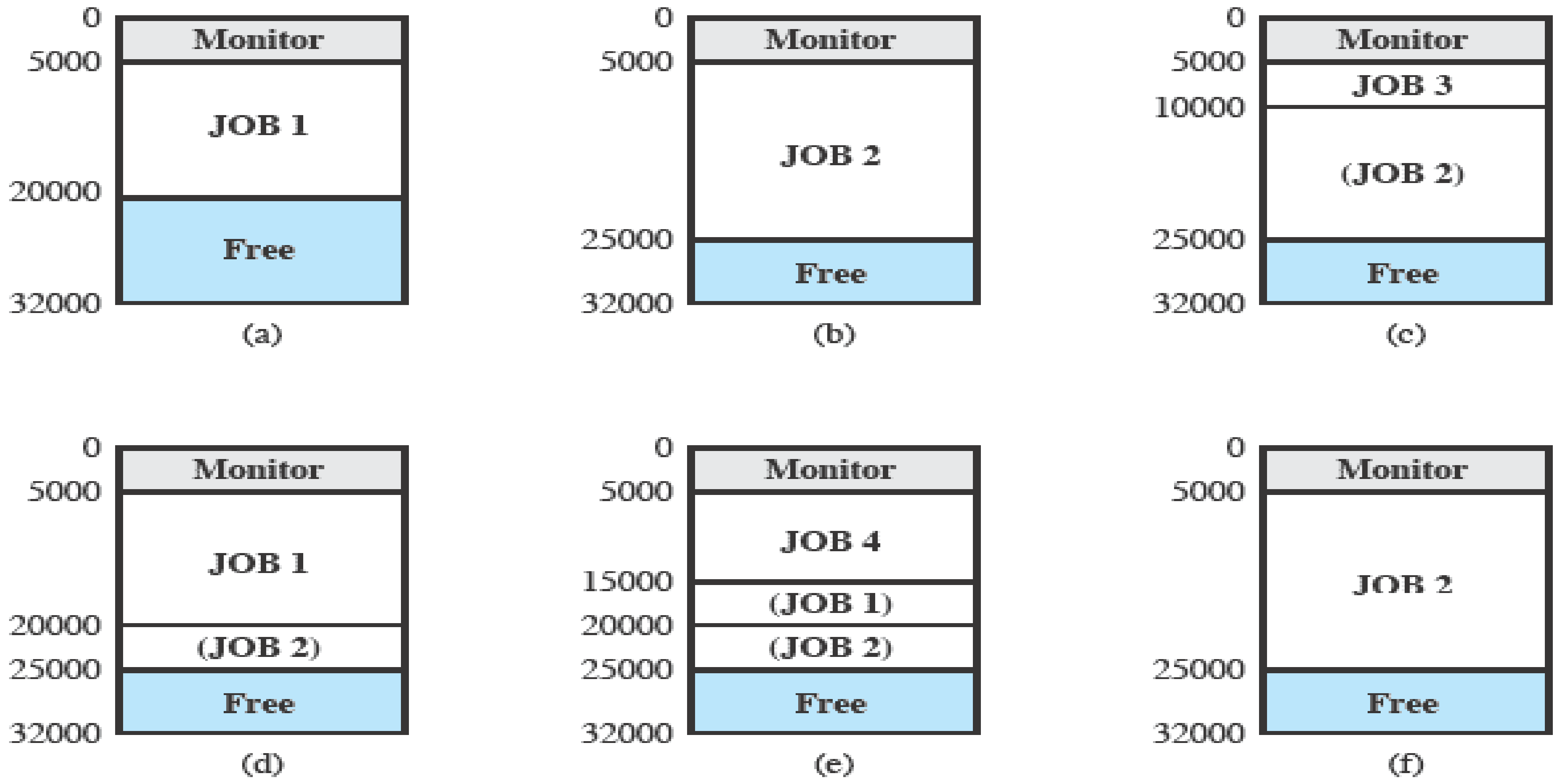


Figure 2.7 CTSS Operation

- Assume that there are four interactive users with the following memory requirements, in words:
- • JOB1: 15,000 • JOB2: 20,000
- • JOB3: 5000 • JOB4: 10,000

- The monitor loads JOB1 and transfers control to it **(a)**.
- Later, the monitor decides to transfer control to JOB2 as it requires more memory than JOB1. JOB1 must be written out first, and then JOB2 can be loaded **(b).**
- Next, JOB3 is loaded in to be run. Because JOB3 is smaller than JOB2, a portion of
- JOB2 can remain in memory, reducing disk write time **(c)**.
- Later, the monitor decides to transfer control back to JOB1.An additional portion of JOB2 must be written out when JOB1 is loaded back into memory **(d)**.
- When JOB4 is loaded, part of JOB1 and the portion of JOB2 remaining in memory are retained **(e)**.
- At this point, if eitherJOB1 or JOB2 is activated, only a partial load will be required.
- In this example, it is JOB2 that runs next.
- This requires that JOB4 and the remaining resident portion of JOB1 be written out and that the missing portion of JOB2 be read in (f).

TIME SHARING SYSTEM

- Problems and Issues
  - *Multiple jobs in memory must be protected from each other's data*
  - *File system must be protected so that only authorised users can*
  - – *access*
  - *Contention for resources must be handled*
    - Printers, storage etc

ROADMAP

- Introduction to Operating system,
- Operating System Objectives and functions,
- Evolution Of Operating System
- **Major achievements**

Major Achievement

- Operating Systems are among the most complex pieces of software ever developed.
- Major Advances include

  I. Processes

  II. Memory management

  III. Information protection and security

  IV. Scheduling and resource management

  V. System structure

I. Process:

- Fundamental to the structure of OS IS A PROCESS..
  - ✓ *A program in execution*
  - ✓ *An instance of a program running on a computer*
  - ✓ *A single sequential thread of execution*

- Causes of Errors when Designing System Software
  - ✓ *Error in designing an OS are often subtle and difficult to diagnose*

- Errors typically include:
  - ✓ *Improper synchronization*
  - ✓ *Failed mutual exclusion*
  - ✓ *Non-determinate program operation*
  - ✓ *Deadlocks*
- **Components of a Process :-**
- A process consists of

■ – An executable program

■ – Associated data needed by the program

■ – Execution context of the program (or "process state")

- The execution context contains all information the operating system needs to manage the process

PROCESS MANAGEMENT

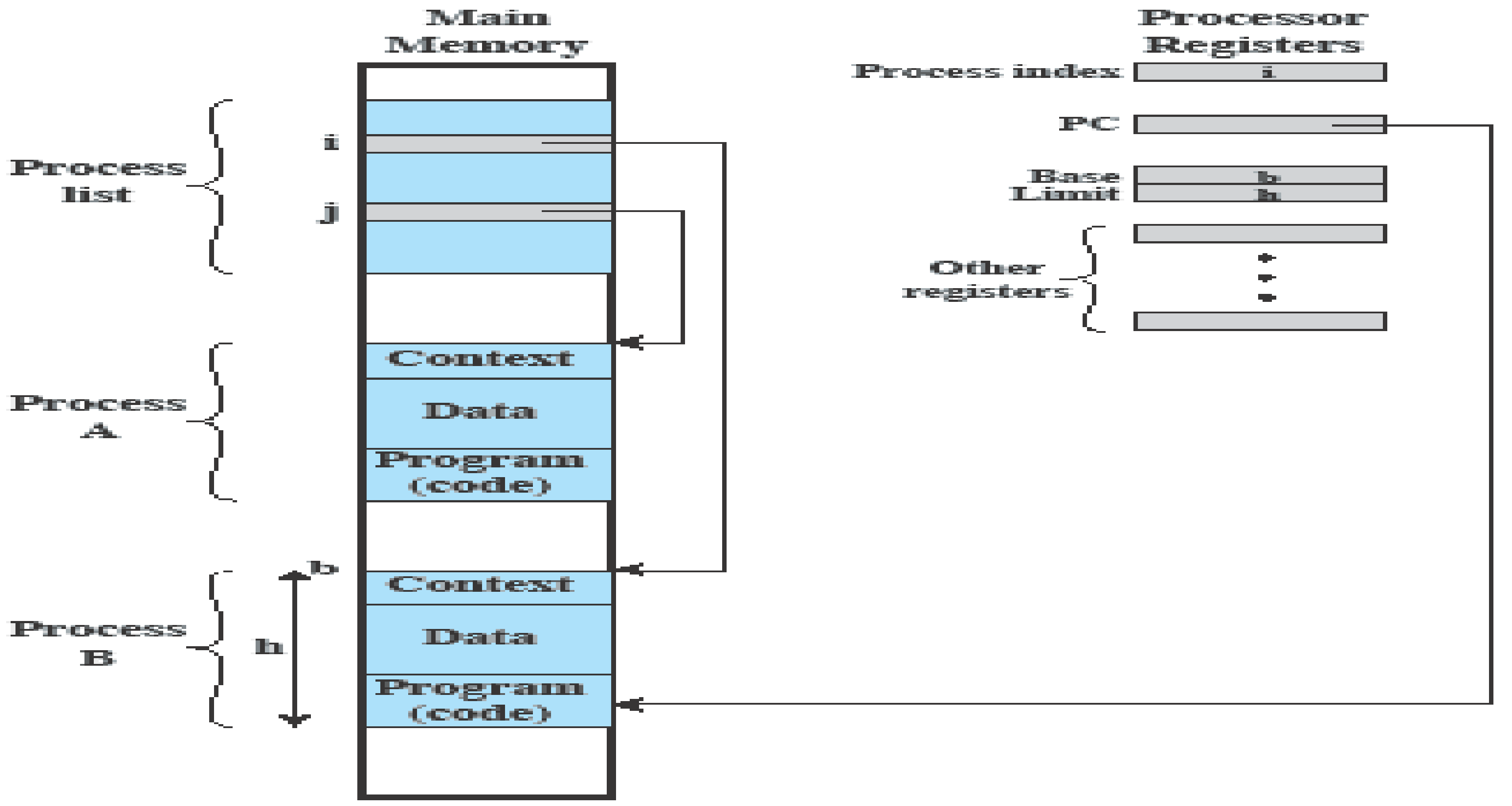


Figure 2.8 Typical Process Implementation

- In Figure 2.8, the process index register indicates that process B is executing.
- Process A was temporarily interrupted.
- The contents of all the registers at the moment of A’s interruption were recorded in its execution context.
- The process switch consists of storing the context of B and restoring the context of A.
- When the program counter is loaded with a value pointing into A’s program area, process A will automatically resume execution.

II. Memory Management

- The OS has 5 principal storage management responsibilities

1. Process isolation:-

- The OS must prevent independent processes from interfering with each other's memory, both data and instructions.

2. Automatic allocation and management:-

- Programs should be dynamically allocated across the memory hierarchy as required.

3. Support of modular programming:-

- Programmers should be able to define program modules, and to create, destroy, and alter the size of modules dynamically.

4. Protection and access control:-

- Sharing of memory, at any level of the memory hierarchy, creates the potential for one program to address the memory space of another.

5. Long-term storage:-

- Many application programs require means for storing information for extended periods of time, after the computer has been powered down.

Virtual Memory:-

Virtual memory refers to the technology in which some space in hard disk is used as an extension of main memory. As shown in the fig 2.9

III. Information Protction & Security

- We are concerned with the problem of controlling access to computer
- systems and the information stored in them.
- Main issues are:
  - Availability:-

    Concerned with protecting the system against interruption
  - Confidentiality:-

    Assures that users cannot read data for which access is unauthorized
  - Data integrity:-

    Protection of data from unauthorized modification
  - Authenticity:-

    Concerned with the proper verification of the identity of users and the validity of messages or data

IV. Scheduling & Resource Management

- A key responsibility of the OS is to manage the various resources available to it and to schedule their use by the various active processes.

- Resource allocation policies must consider:

– Fairness:-

All processes that are competing for the use of a particular resource to be given approximately equal and fair access to that resource.

– Differential responsiveness:-

- *OS may need to discriminate among different classes of jobs with different service requirements.*
- *The OS should attempt to make allocation and scheduling decisions to meet the total set of requirements.*

– Efficiency:-

- *The OS should attempt to maximize throughput, minimize response time, and, in the case of time sharing, accommodate as many users as possible*

Key elements of o.s. in Multiprogramming

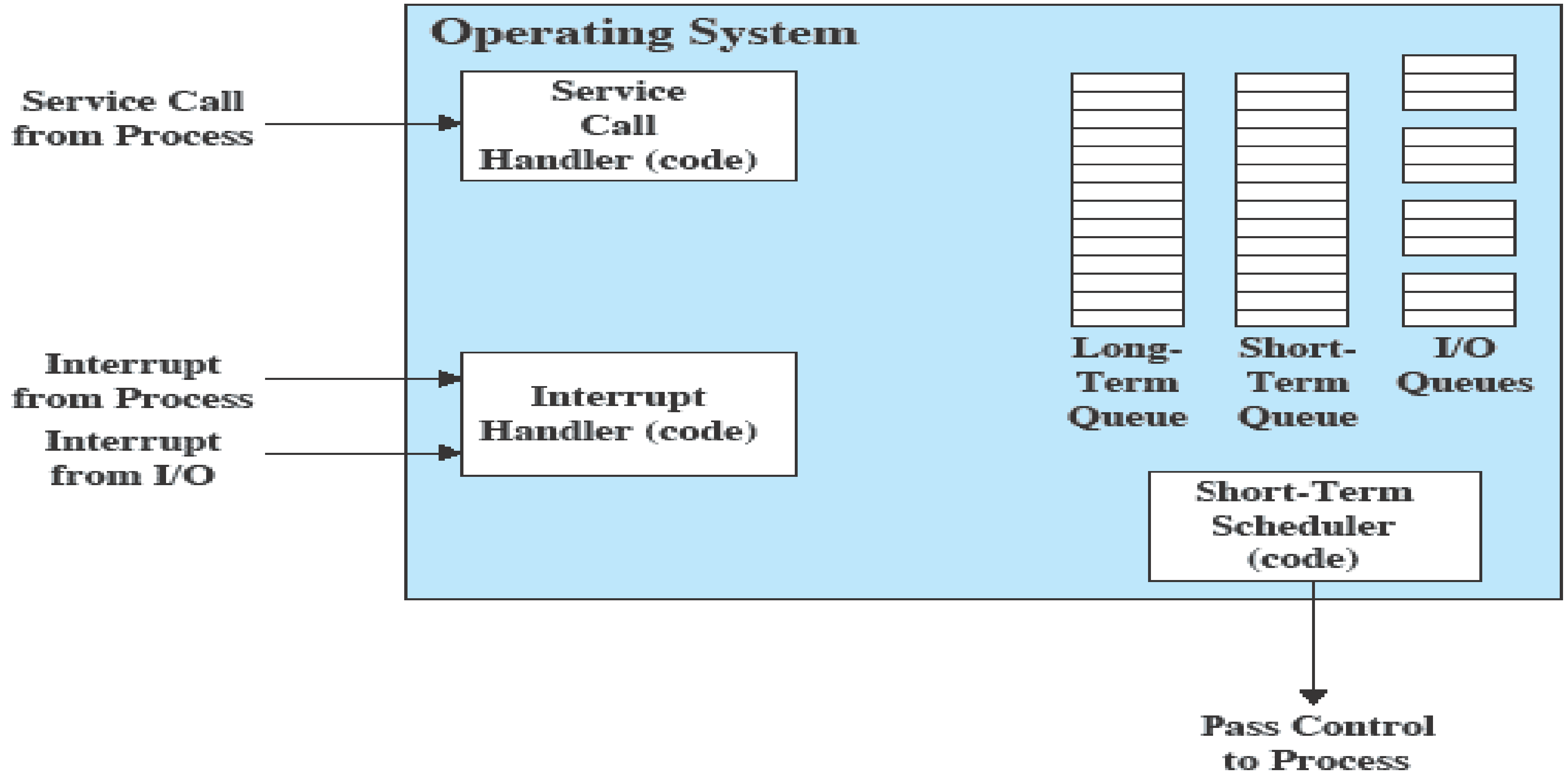


Figure 2.11 Key Elements of an Operating System for Multiprogramming

Key elements of o.s. in Multiprogramming

- Figure 2.11 suggests the major elements of the OS involved in the scheduling of processes and the allocation of resources in a multiprogramming environment.
- The OS maintains a number of queues, each of which is simply a list of processes waiting for some resource.
- The short-term queue:-
  - *Consists of processes that are in main memory and are ready to run as soon as the processor is made available.*
- The long-term queue:-
  - *Consists of a list of new jobs waiting to use the processor.*
- The I/O queue:-
  - *For each Input Output Device.*
  - *All processes waiting to use each device are lined up in that device's queue.*

V. System Structure

- View the system as a series of levels.
- Each level performs a related subset of functions.
- Each level relies on the next lower level to perform more primitive functions.
- This decomposes a problem into

- **Level 1:**

  Consists of electronic circuits, where the objects that are dealt with are registers, memory cells, and logic gates

- **Level 2:**

  The processor's instruction set

- **Level 3:**

  Adds the concept of a procedure or subroutine, plus the call/return operations.

- **Level 4:**

  Introduces interrupts, which cause the processor to save the current context and invoke an interrupt-handling routine.

- These first four levels are not part of the OS but constitute the processor hardware a number of more manageable sub problems.

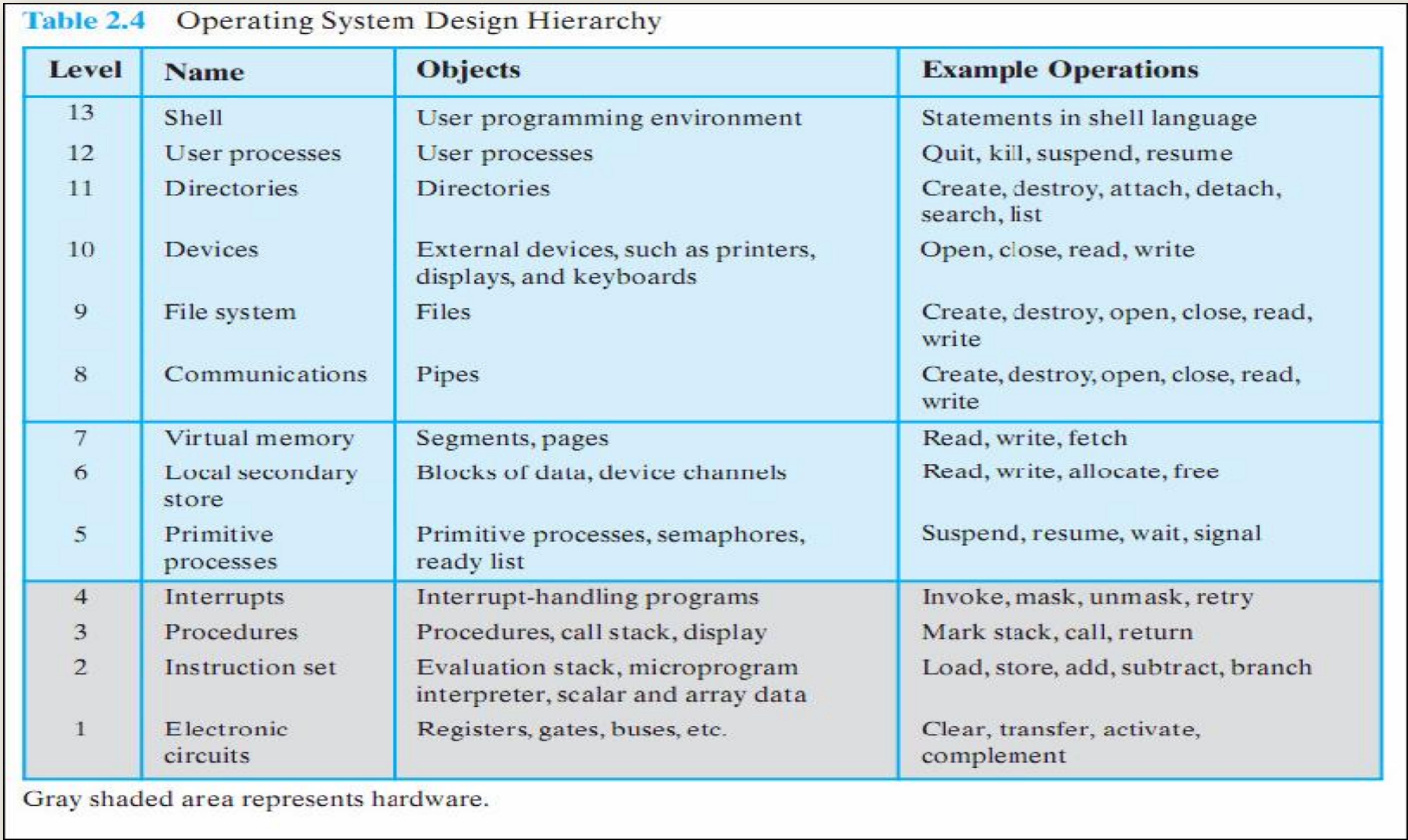


Table 2.4 Operating System Design Hierarchy

Level	Name	Objects	Example Operations
13	Shell	User programming environment	Statements in shell language
12	User processes	User processes	Quit, kill, suspend, resume
11	Directories	Directories	Create, destroy, attach, detach, search, list
10	Devices	External devices, such as printers, displays, and keyboards	Open, close, read, write
9	File system	Files	Create, destroy, open, close, read, write
8	Communications	Pipes	Create, destroy, open, close, read, write
7	Virtual memory	Segments, pages	Read, write, fetch
6	Local secondary store	Blocks of data, device channels	Read, write, allocate, free
5	Primitive processes	Primitive processes, semaphores, ready list	Suspend, resume, wait, signal
4	Interrupts	Interrupt-handling programs	Invoke, mask, unmask, retry
3	Procedures	Procedures, call stack, display	Mark stack, call, return
2	Instruction set	Evaluation stack, microprogram interpreter, scalar and array data	Load, store, add, subtract, branch
1	Electronic circuits	Registers, gates, buses, etc.	Clear, transfer, activate, complement

Gray shaded area represents hardware.

PresenterMedia