Why study Operating Systems?

- Operating systems are a maturing field
  - Most people use a handful of mature OSes
  - Hard to get people to switch operating systems
  - Hard to have impact with a new OS
- High-performance servers are an OS issue
  - Face many of the same issues as OSes
- Resource consumption is an OS issue
  - Battery life, radio spectrum, etc.
- Security is an OS issue
  - Hard to achieve security without a solid foundation
- New “smart” devices need new OSes

Big Data, Networks, large-scale software systems, whatever --- Key to its success is Operating System Design Principles, its working and optimal use of its policies.

What is an OS?

- Tool to make programmer's job easy
- Resource allocator / manager
  - Must be fair; not partial to any process, specially for process in the same class
  - Must discriminate between different class of jobs with different service requirements
  - Do the above efficiently
    - Within the constraints of fairness and efficiency, an OS should attempt to maximize throughput, minimize response time, and accommodate as many users as possible
- Control program
- Tool to facilitate efficient operation of a computer system
- Virtual machine that is easier to understand and program
  - Encapsulates the complexities of hardware

OS as Resource Manager

- A computer is a set of resources for the movement, storage, and processing of data.
  - Data could be associated with programs, devices, and even the underlying hardware.
- The OS is responsible for managing these resources.

15

OS as Resource Manager

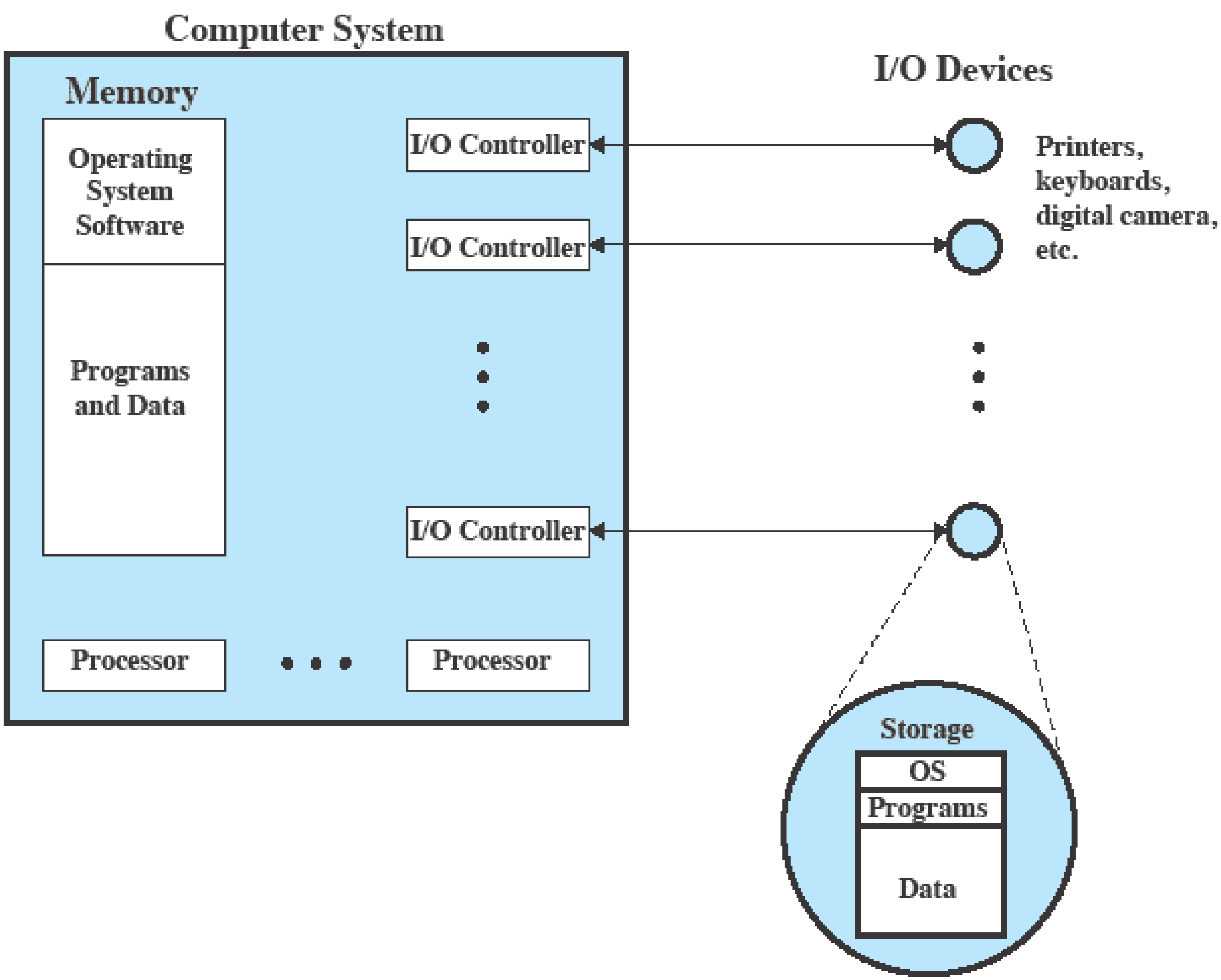


Figure 2.2 The Operating System as Resource Manager

Layers and Views

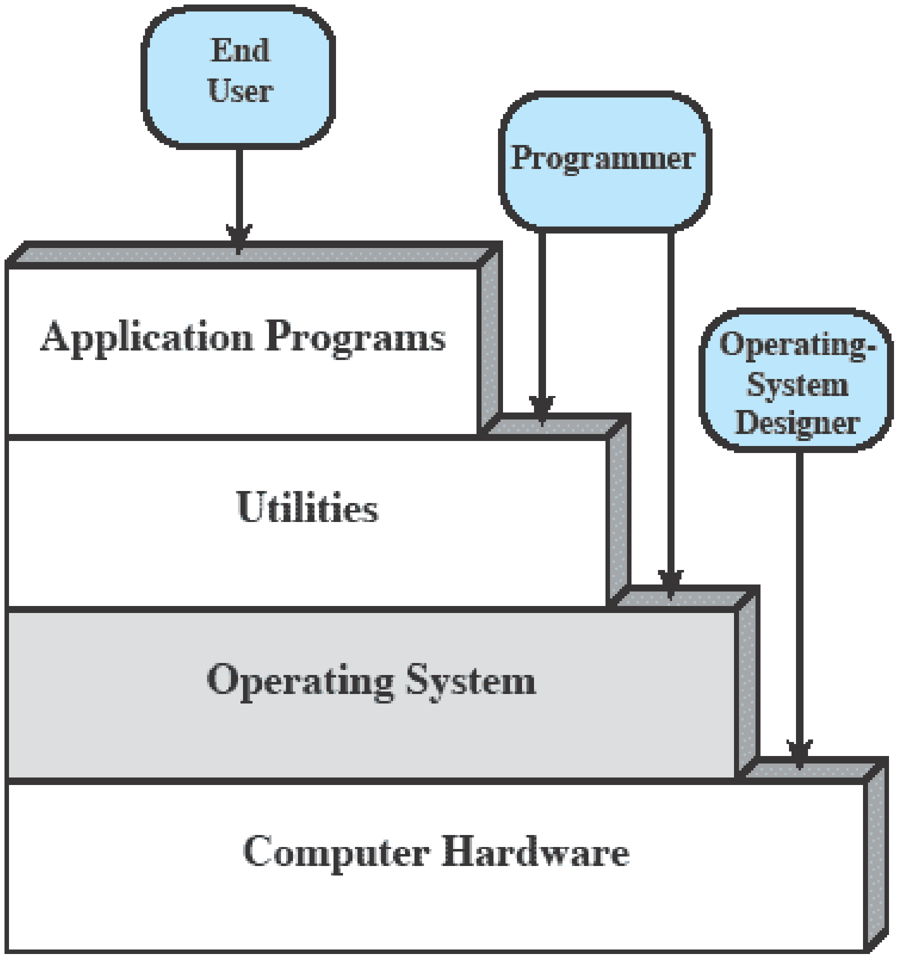


Figure 2.1 Layers and Views of a Computer System

Layered Architecture Design

Banking system	Airline reservation	Adventure games
Compilers	Editors	Command Interpreter
Operating System		
Machine Language		
Microprogramming		
Physical Devices		

This is our point of focus

- Typical OS structure
  - Application environment - shell, mail, text processing package, sccs
  - Operating system - support programs for applications

A microinstruction program that controls the functions of a central processing unit or peripheral controller of a computer.

A set of microinstructions that defines the individual operations that a computer carries out in response to a machine-language instruction.

..... many more definitions published and available in different literature.

Zooming into the OS Layer

Programs and Processes

Resource Usage – includes memory, disk etc.

Input / Output Operations

File Systems

Communication

Protection and Security

Accounting for all.

These are our point of focus

OS as Service Provider

- Program development
  - e.g., editors and debuggers
- Program execution
- Access I/O devices
- Controlled access to files
- System access for shared systems
- Error detection and response
  - e.g., memory error, device failure, division by zero
- Accounting for resources and performance monitoring

How about Operating System as a Service? - Think about it!
We are in the age of Cloud Computing – How OS changes?

Operating System

- A program that controls the execution of application programs
- An interface between applications and hardware
- Main objectives of an OS:
  - Convenience
  - Efficiency and Fairness
  - Ability to evolve

21

Early Systems

1945 - 1955
Bare machines - vacuum tubes and plugboards
No operating system, No protection
Black box concept - human operator
ENIAC - Electronic Numerical Integrator And Computer

Second Generation

Third Generation

Fourth Generation and beyond

Early Systems

1945 - 1955
Bare machines - vacuum tubes and plugboards
No operating system, No protection
Black box concept - human operator
ENIAC - Electronic Numerical Integrator And Computer

Second Generation

1956 - 1965
Transistors and batch systems
Clear distinction between designer, builders, operators, Programmers, and maintenance personnel
I/O channel
Read ahead / spooling,Interrupts / exceptions
Minimal protection, Libraries / JCL

Third Generation

Fourth Generation and beyond

1965 - 1980

ICs and Multiprogramming
System 360 and S/370 family of computers
Spooling (simultaneous peripheral operation on-line)
Time sharing
On-line storage for - system programs, user programs/data, and libraries.
Virtual memory
Multiprocessor configurations
MULTICS - Multiplexed Information and Computing Service

- Design started in 1965 and completed in 1972
- Collaborative effort between GE, Bell Labs, and Project MAC of MIT
- Aimed at providing - simultaneous computer access to large community of users, ample computation power and data storage, easy data sharing between users, if desired

Third Generation

Fourth Generation and beyond

Personal computers and workstations
MS-DOS and Unix
Massively parallel systems
- Pipelining
- Array processing / SIMD
- Multiprocessing / MIMD
- Symmetric multiprocessing
  - Any process and any thread can run on any available processor

Computer networks (communication aspect) - network operating systems
Distributed computing - distributed operating systems

Third Generation

Fourth Generation and beyond

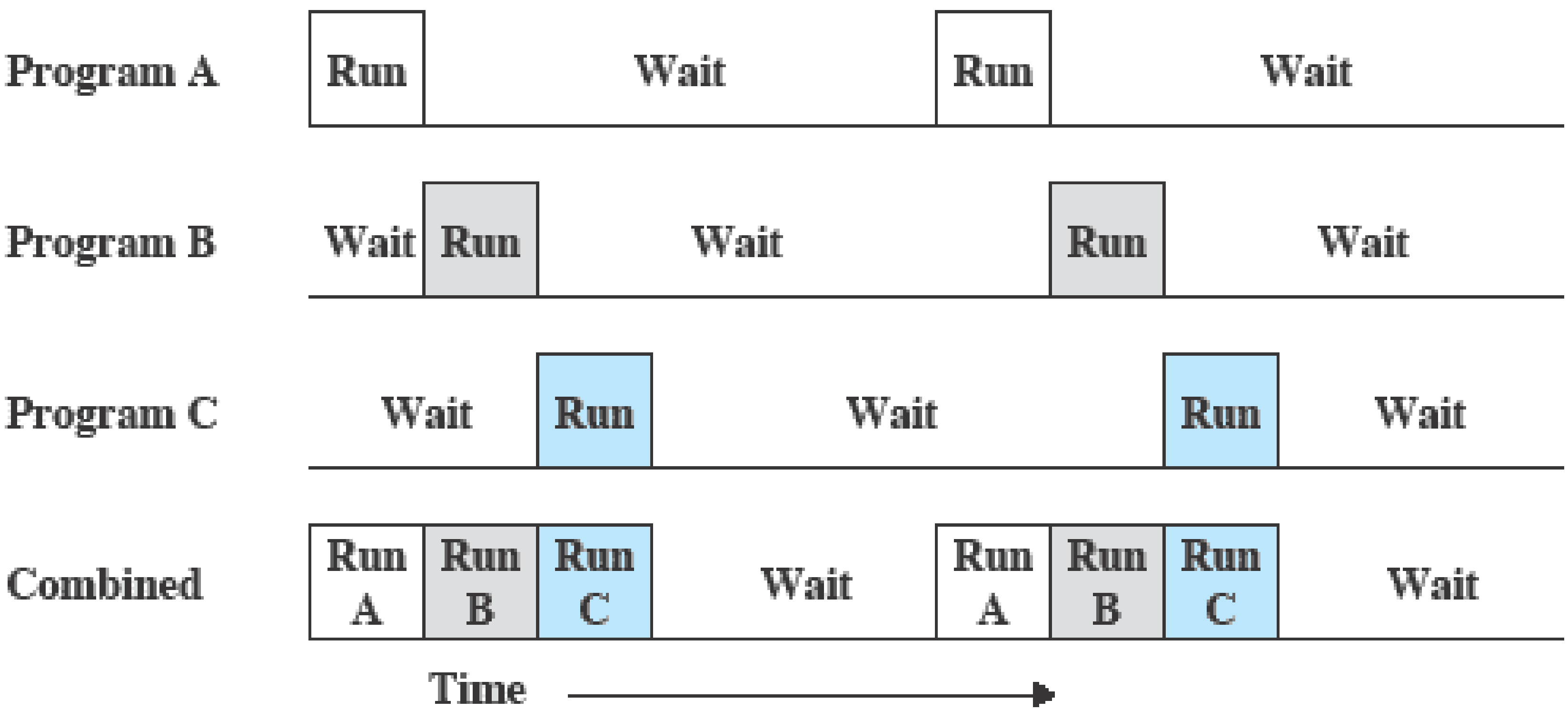
Uniprogramming

- Processor must wait for I/O instruction to complete before preceding

(a) Uniprogramming

Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job



(c) Multiprogramming with three programs

Idea of degree of multiprogramming – solution to avoiding CPU being idle
But what is the effect on Response time?

32

Time Sharing Systems

- Using multiprogramming to handle multiple interactive jobs
- Processor’s time is shared among multiple users

33

OS Concepts

- Kernel
  - Permanently resides in the main memory
  - Controls the execution of processes by allowing their creation, termination or suspension, and commn.
  - Schedules processes fairly for execution on the CPU
  - Allocates main memory for an executing process
  - File system maintenance
  - Allows processes controlled access to peripheral devices such as terminals, tape drives, disk drives, and network devices.
  - Design options – Monolithic kernels, Layered, Micro kernels.

- Processes share the CPU in a time-shared manner
  - CPU executes a process
  - Kernel suspends it when its time quantum elapses
  - Kernel schedules another process to execute
  - Kernel later reschedules the suspended process

- Kernel
  - Permanently re[illegible]ne main memory
  - Controls the [illegible]tion of processes by allowing their creation, ter[illegible]ation or suspension, and commn.
  - Schedules processes fairly for execution on the CPU
  - Allocates main memory for an executing process
  - File system maintenance
  - Allows processes controlled access to peripheral devices such as terminals, tape drives, disk drives, and network devices.
  - Design options – Monolithic kernels, Layered, Micro kernels.

OS Concepts

Allows processes to share portions of their address space under certain conditions, but protects the private address space of a process from outside tampering

If system runs low on free memory, the kernel frees memory by writing a process temporarily to secondary memory, or swap device

If kernel writes entire processes to a swap device, then the implementation is called a swapping system; if it writes pages of memory to a swap device, it is called a paging system.

Coordinates with the machine hardware to set up a virtual to physical address that maps the compiler generated addresses to their physical addresses

- Allocates main memory for an executing process
- File system maintenance
- Allows processes controlled access to peripheral devices such as terminals, tape drives, disk drives, and network devices.
- Design options – Monolithic kernels, Layered, Micro kernels.

OS Concepts

- Kernel
  - Permanently resides in the mai
  - Controls the execution of proce
    creation, termination or suspen
  - Schedules processes fairly for
  - Allocates main memory for
  - File system maintenance

    > Allocates secondary memory for efficient storage and retrieval of user data
    >
    > Allocates secondary storage for user files
    >
    > Reclaims unused storage
    >
    > Structures the file system in a well understood manner
    >
    > Protects user files from illegal access

  - Allows processes controlled access to peripheral devices such as terminals, tape drives, disk drives, and network devices.
  - Design options – Monolithic kernels, Layered, Micro kernels.

OS Concepts - Program

- Program
  - Collection of source code instructions and any associated data kept in a disk file
  - Source program, or a human-readable text file.
  - Machine language translation of the source program, or object file
    - The file is marked as executable.
    - File contents are arranged according to rules established by the kernel
    - Executable program, complete code output by linker / loader, with input from libraries

Example of Program Execution

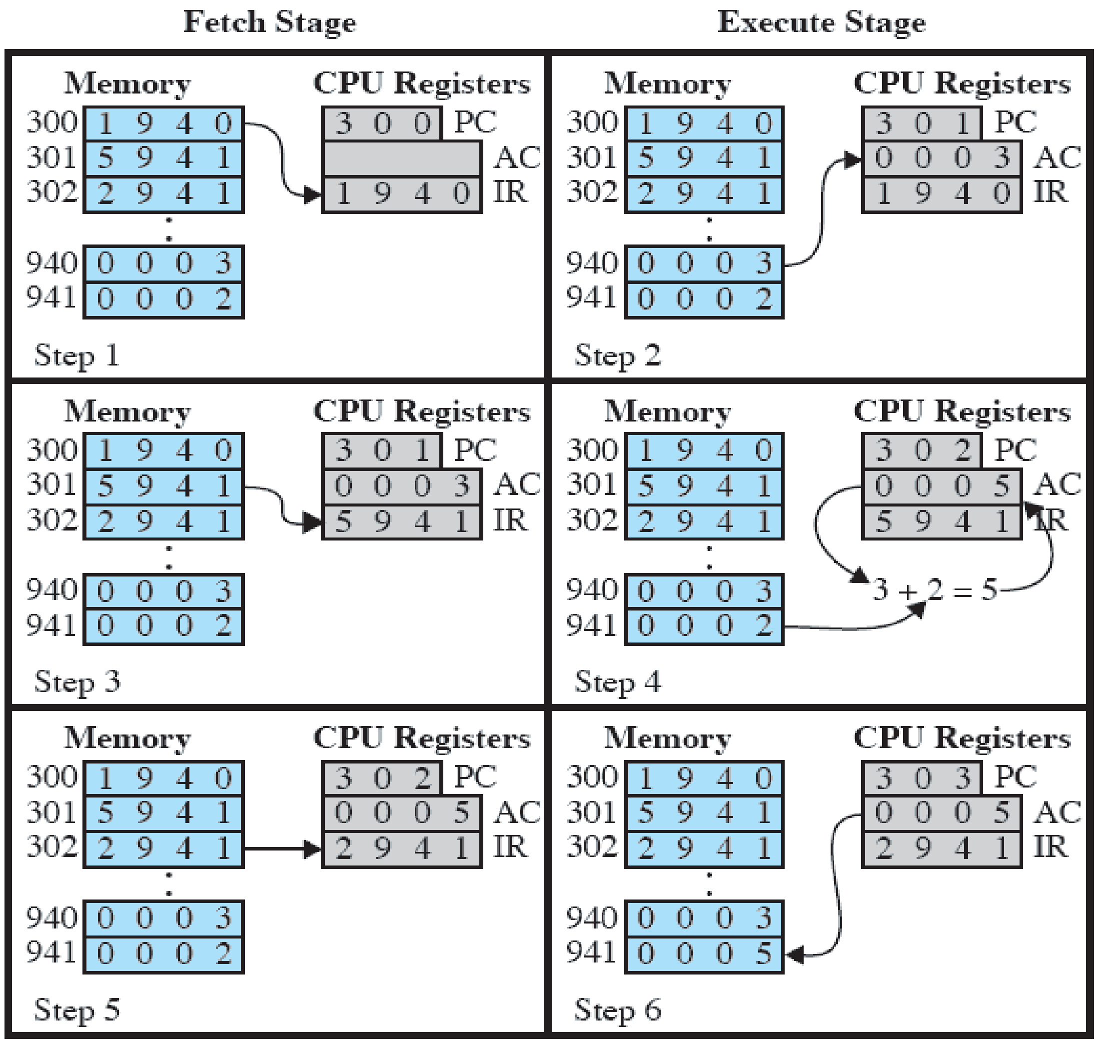


Figure 1.4 Example of Program Execution
(contents of memory and registers in hexadecimal)

40

Instruction Execution

- A program consists of a set of instructions stored in memory
- Two steps
  - Processor reads (fetches) instructions from memory
  - Processor executes each instruction

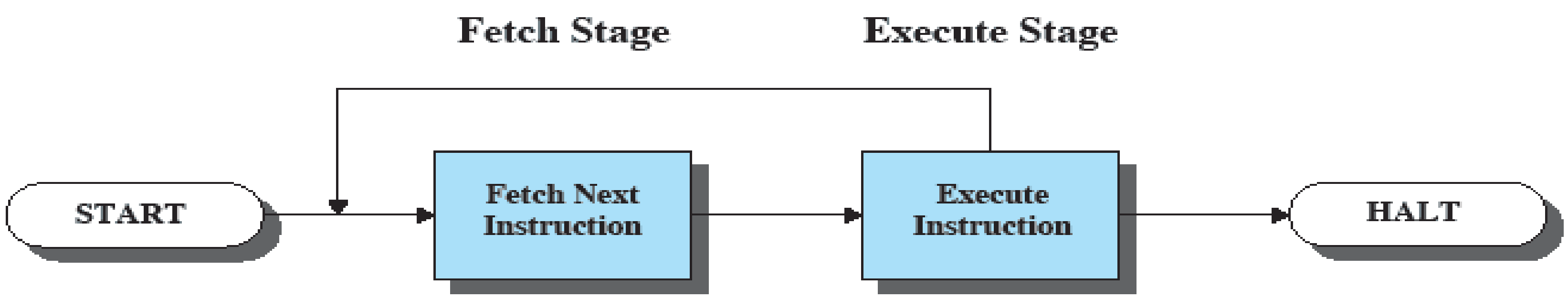


Figure 1.2 Basic Instruction Cycle

41

OS Concepts

- Memory (Volatility)
  - Memory hierarchy based on storage capacity, speed, and cost
  - Higher the storage capacity, lesser the speed, and lesser the cost
  - Different memory levels, in decreasing cost per byte of storage
  - Use hierarchical memory to transfer data from lower memory to higher memory to be executed
  - Locality of reference
    - Most of the references in the memory are clustered and move from one cluster to the next
  - Cache memory
    - Use of very fast memory (a few kilobytes) designated to contain data for fast access by the CPU
  - Virtual memory or extension of main memory
  - Disk cache
    - Data-intensive applications (generally rotational speed and seek time)

OS Concepts

Registers	Few bytes	Almost CPU speed
Cache memory	Few kilobytes	Nanoseconds
Main memory	Megabytes	Microseconds
Magnetic disk	Gigabytes	Milliseconds
Magnetic tape/Optical disk	No limit	Offline storage

- Memory (Volatility)
  - Memory hierar
  - Higher the st
  - Different memory
  - Use hierarchical memory to transfer data from lower memory to higher memory to be executed
  - Locality of reference
    - Most of the references in the memory are clustered and move from one cluster to the next
  - Cache memory
    - Use of very fast memory (a few kilobytes) designated to contain data for fast access by the CPU
  - Virtual memory or extension of main memory
  - Disk cache
    - Data-intensive applications (generally rotational speed and seek time)

OS Concepts

- Memory (Volatility)
  - Mem
  - High
  - Diffe
  - Use
    high
  - Loca
    -
  - Cache m
    - Use of v y (a few kilobytes) designated to contain data for fast acc CPU
  - Virtual me or extension of main memory
  - Disk cache
    - Data-intensive applications (generally rotational speed and seek time)

A disk cache is a mechanism for improving the time
it takes to read from or write to a hard disk.

Today, the disk cache is usually included as part of the hard disk.
(A disk cache is RAM built into your hard disk)
(Disk specifications : on-board cache)
OR
A disk cache can also be a specified portion of system RAM (disk buffer).

Disk cache holds data that has recently been read and, in
some cases, adjacent data areas that are likely to be accessed next.

Write caching (accumulated) is also provided with some disk caches.

OS Concepts – System Calls

Interface between user program and operating system - Set of extended instructions provided by the operating system

- Applied to various software objects like processes and files
- Invoked by user programs to communicate with the kernel and request services
- Access routines in the kernel that do the work
- Library procedure corresponding to each system call

Why modes?
Privileges?
- indicates power for System control

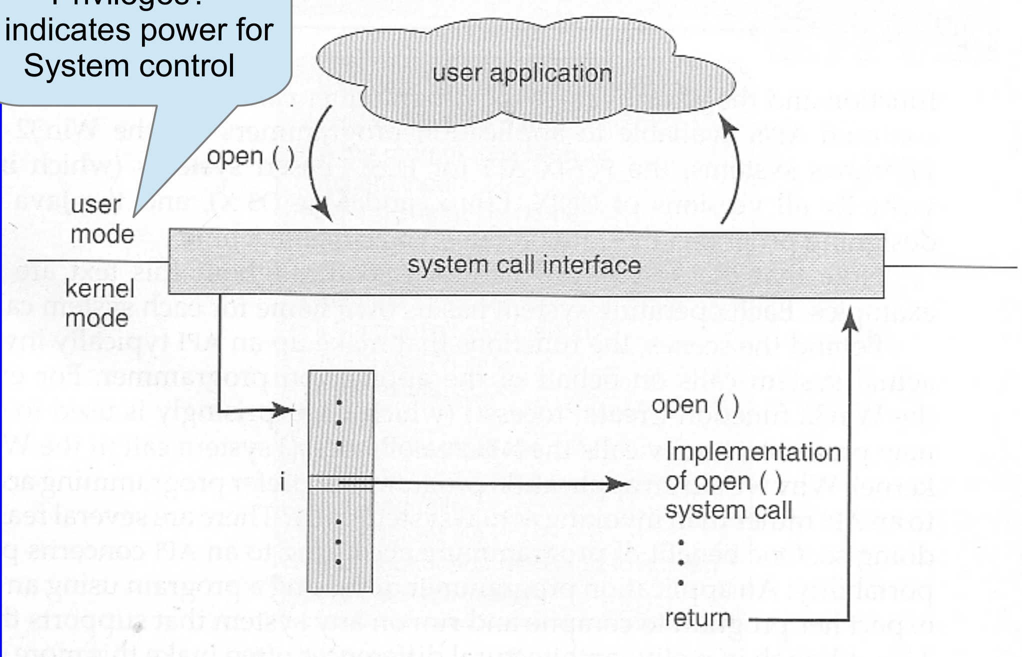


Figure 2.3 The handling of a user application invoking the open() system call.

Working of a System Call – How?

EXAMPLE OF STANDARD C LIBRARY

The standard C library provides a portion of the system-call interface for many versions of UNIX and Linux. As an example, let's assume a C program invokes the `printf()` statement. The C library intercepts this call and invokes the necessary system call(s) in the operating system—in this instance, the `write()` system call. The C library takes the value returned by `write()` and passes it back to the user program. This is shown in Figure 2.6.

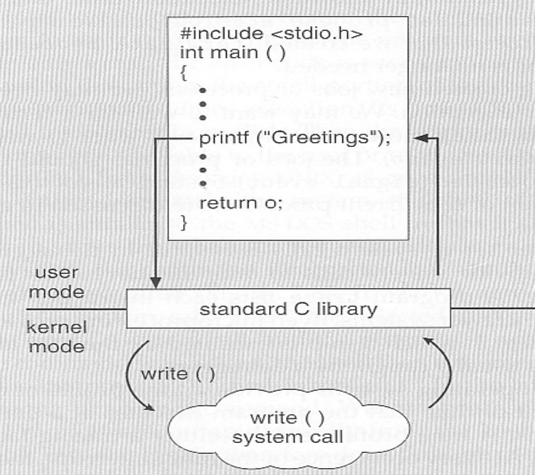


Figure 2.6 C library handling of `write()`.

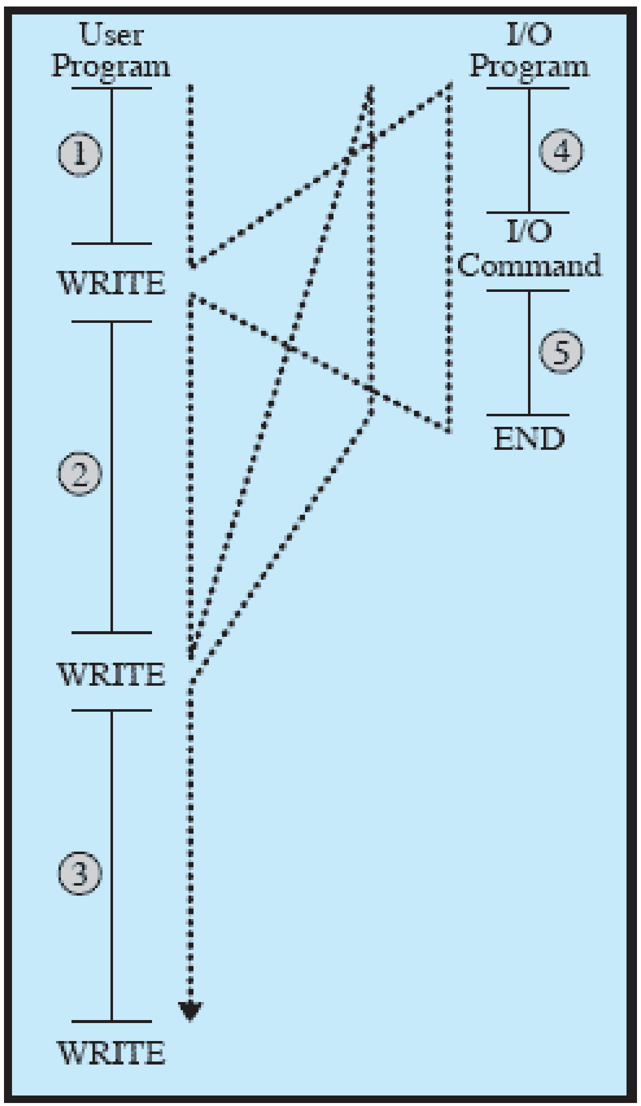
Interrupts

- Interrupt the normal sequencing of the processor
- Provided to improve processor utilization
  - Most I/O devices are slower than the processor
  - Processor must pause to wait for device

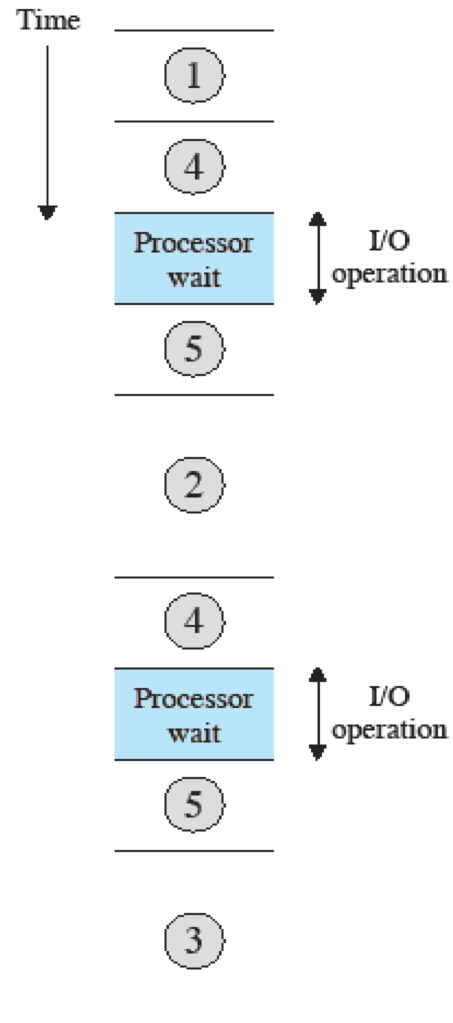
47

Flow of Control w/o Interrupts and I/O Wait

Figure 1

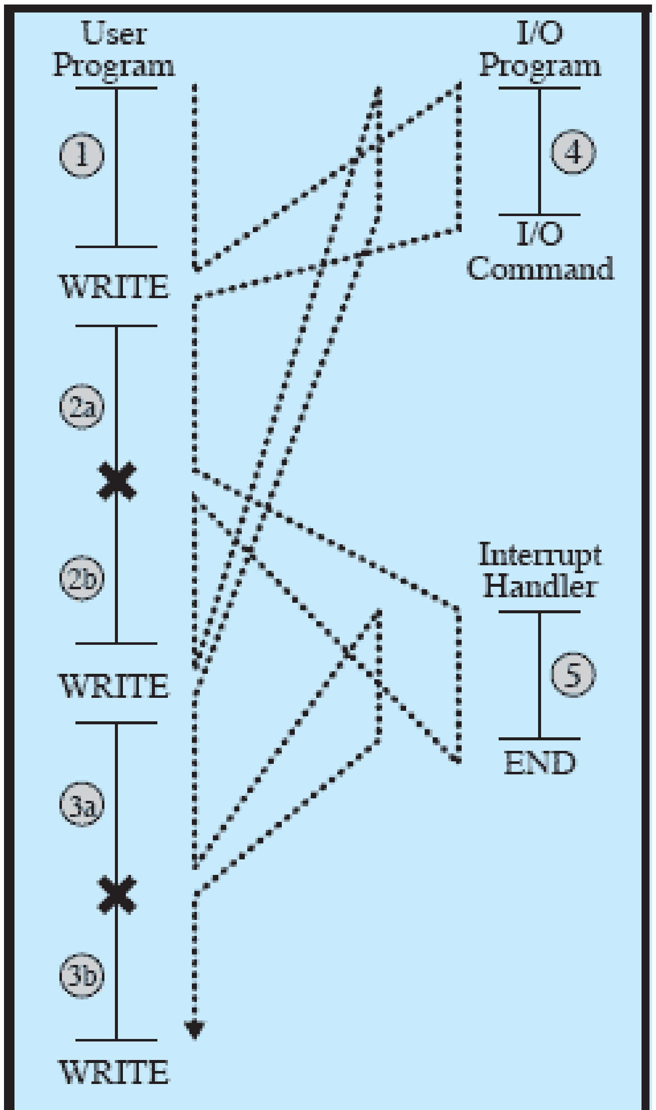


(a) No interrupts

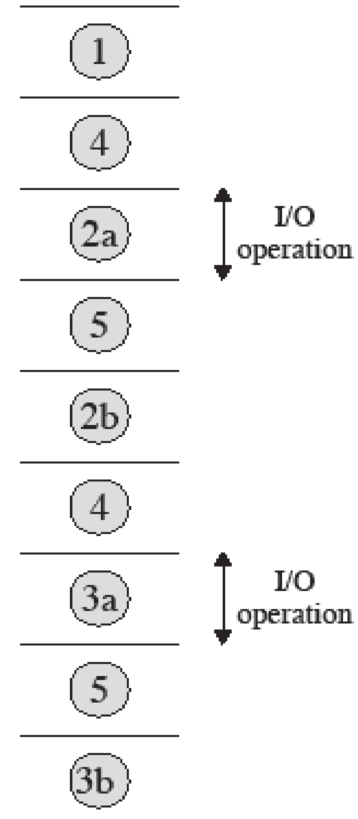


Flow of Control with Interrupts and I/O Wait

Figure 2



(b) Interrupts; short I/O wait



Transfer of Control via Interrupts

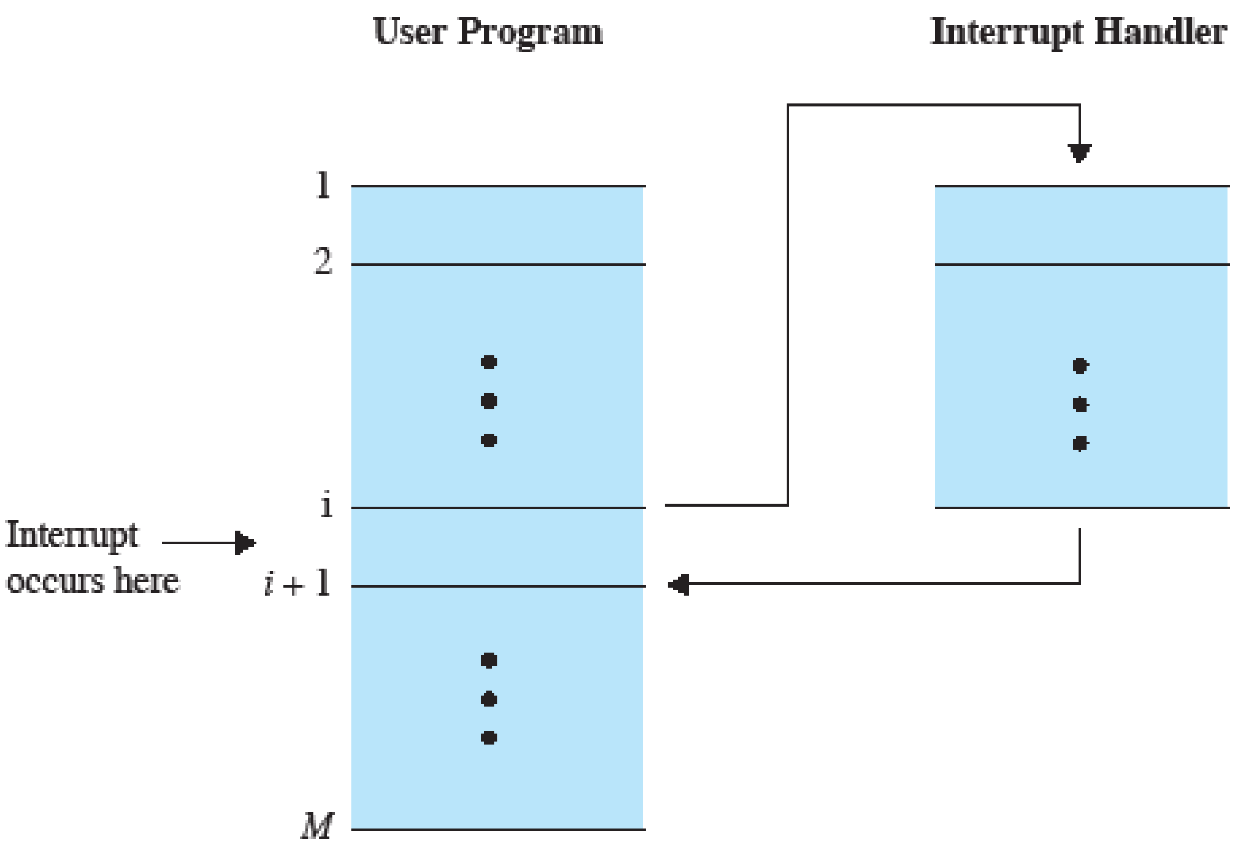


Figure 1.6 Transfer of Control via Interrupts

52

Instruction Cycle with Interrupts

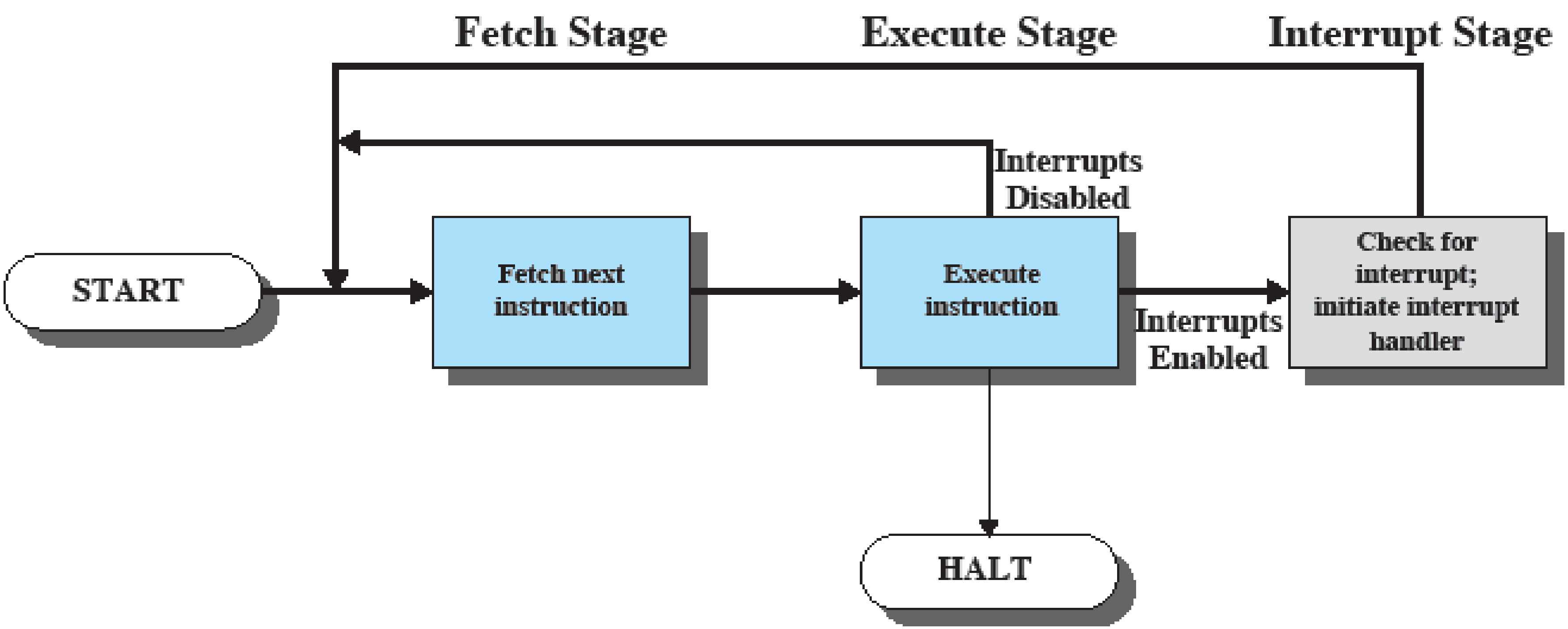


Figure 1.7 Instruction Cycle with Interrupts