Computer-System Architecture

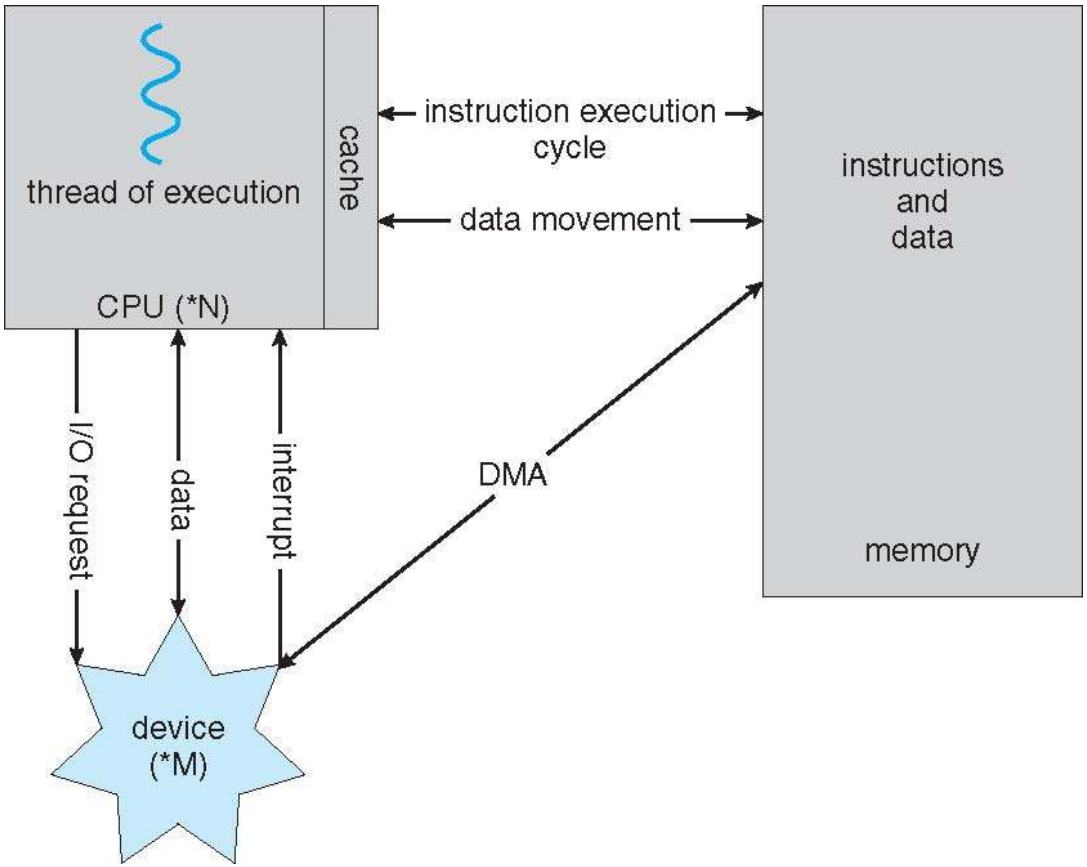
- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput**
    2. **Economy of scale**
    3. **Increased reliability** – graceful degradation or fault tolerance
  - Two types:
    1. **Asymmetric Multiprocessing** – each processor is assigned a specie task.
    2. **Symmetric Multiprocessing** – each processor performs all tasks

Multiprocessors Operating system

- Multiprocessor operating system allows the multiple processors, and these processors are connected with physical memory, computer buses, clocks, and **peripheral devices**.
- Main objective of using multiprocessor operating system is to consume high computing power and increase the execution speed of system.

- **Symmetric Multiprocessor**
- In this system, every processors have own identically copy of **operating system**, and they can make communication in between each other. In which all processors are connected each other with peer to peer relationship nature, it means no master & slave relation.
- **Asymmetric Multiprocessor**
- In this system, every processor is allotted predefined tasks, and master processor has power for controlling entire system. In which, It use the master- slave relationship.

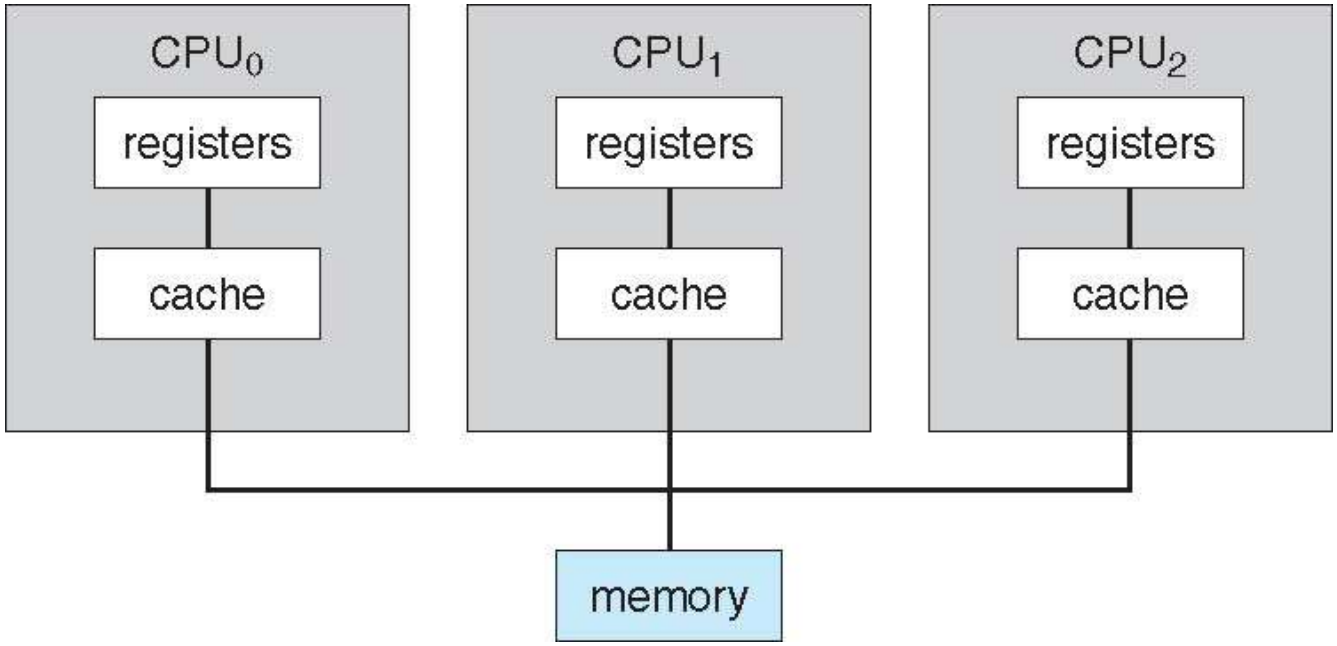
How a Modern Computer Works



A von Neumann architecture

BASIS FOR COMPARISON	LOOSELY COUPLED MULTIPROCESSOR SYSTEM	TIGHTLY COUPLED MULTIPROCESSOR SYSTEM
Basic	Each processor has its own memory module.	Processors have shared memory modules.
Efficient	Efficient when tasks running on different processors, has minimal interaction.	Efficient for high-speed or real-time processing.
Memory conflict	It generally, do not encounter memory conflict.	It experiences more memory conflicts.
Interconnections	Message transfer system (MTS).	Interconnection networks PMIN, IOPIN, ISIN.
Data rate	Low.	High.
Expensive	Less expensive.	More expensive.

Symmetric Multiprocessing Architecture

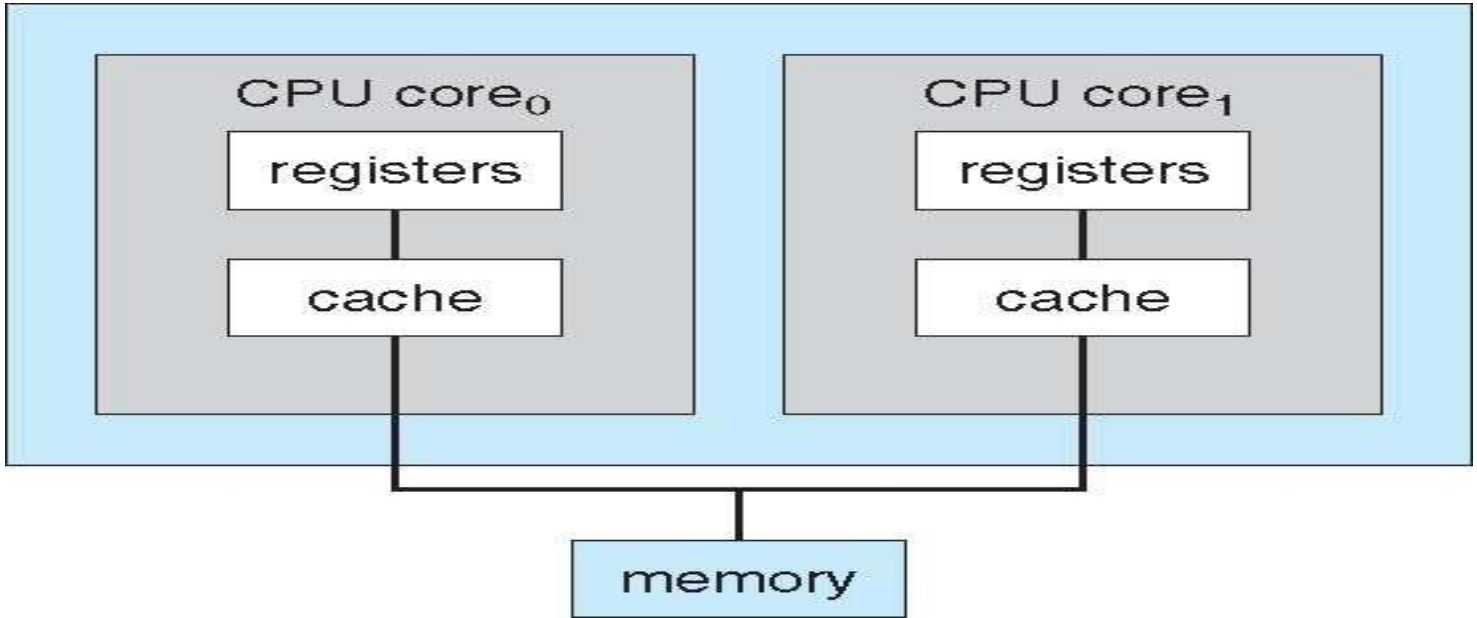


Symmetric Multiprocessing Architecture

- Symmetric multiprocessing is the use of two or more self-scheduling processors sharing a common memory space.
- Each processor has access to I/O and memory devices. SMP applies multiple CPUs to a task to complete in parallel and faster fashion.

A Dual-Core Design

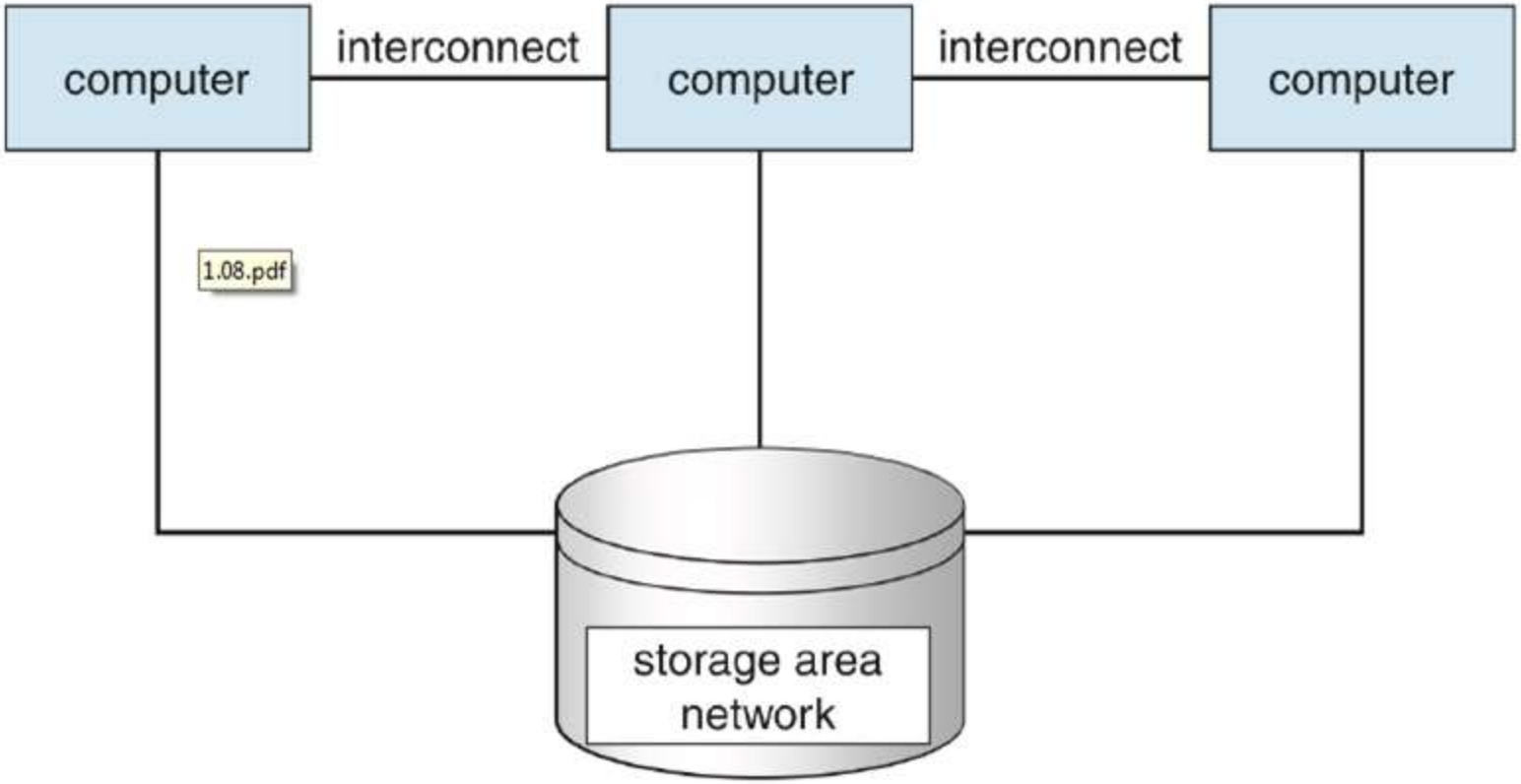
- Multi-chip and **multicore**
- Systems containing all chips
  - A dual-core processor is a CPU with two processors or "execution cores" in the same integrated circuit. Each processor has its own cache and controller, which enables it to function as efficiently as a single processor. However, because the two processors are linked together, they can perform operations up to twice as fast as a single processor can.



Clustered Systems

- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a **storage-area network (SAN)**
  - Provides a **high-availability** service which survives failures
    - **Asymmetric clustering** has one machine in hot-standby mode
    - **Symmetric clustering** has multiple nodes running applications, monitoring each other
  - Some clusters are for **high-performance computing (HPC)**
    - Applications must be written to use **parallelization**
  - Some have **distributed lock manager** (**DLM**) to avoid conflicting operations

Clustered Systems



What is a SAN?

SAN

It is a high-speed, dedicated network of servers and shared storage devices.

- Centralizes storage and management
- Enables sharing of storage resources across multiple servers at block level
- Meets increasing storage demands efficiently with better economies of scale
- Common SAN deployments are:
  - Fibre Channel (FC) SAN: uses FC protocol for communication
  - IP SAN: uses IP-based protocols for communication

Common System Components

- Process Management
- Main Memory Management
- File Management
- I/O System Management
- Secondary Management
- Networking
- Protection System
- Command-Interpreter System

Process Management

- A process is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- The operating system is responsible for the following activities in connection with process management.
  - Creating & deleting both user & system processes
  - process suspension and resumption.
  - Providing mechanisms for:
    - process synchronization
    - process communication
    - Deadlock handling

Main-Memory Management

- It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a volatile storage device. It loses its contents in the case of system failure.
- The operating system is responsible for the following activities in connections with memory management:
  - Keep track of which parts of memory are currently being used and by whom.
  - Decide which processes to load when memory space becomes available.
  - Allocate and deallocate memory space as needed.

File Management

- A file management system is a type of software that manages data files in a computer system. It has limited capabilities and is designed to manage individual or group files, such as special office documents and records. It may display report details, like owner, creation date, state of completion and similar features useful in an office environment.
- A file management system is also known as a file manager.

File Management

- A file is a collection of related information. Commonly, files represent programs and data.
- OS implements file by mass storage media (disks, tape) & devices that control them
- The operating system is responsible for the following activities in connections with file management:
  - File creation and deletion.
  - Directory creation and deletion.
  - Support of primitives for manipulating files and directories.
  - File backup on stable (nonvolatile) storage media.

I/O System Management

- One of the purposes of an OS is to hide the peculiarities of specific hardware devices from the user.
- This can be done by I/O subsystem. The I/O subsystem consists of:
  - A general device-driver interface
  - Drivers for specific hardware devices
  - Only the device driver knows the peculiarities of the specific device to which it is assigned

Secondary-Storage Management

- Since main memory (primary storage) is volatile and too small to accommodate all data and programs permanently, the computer system must provide secondary storage to back up data.
- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- The operating system is responsible for the following activities in connection with disk management:
  - Free space management
  - Storage allocation
  - Disk scheduling

Networking (Distributed Systems)

- The processors in the system are connected through a communication network.
- Communication takes place using a protocol.
- A distributed system provides user access to various system resources.
- Access to a shared resource allows:
  - Computation speed-up
  - Increased data availability
  - Enhanced reliability

Protection System

- Computer has multiple users & allow concurrent execution of multiple processes, then these processes must be protected from one another's activities.
- Protection refers to a mechanism for controlling access of programs, processes, or users to both system and user resources.
- The protection mechanism must:
  - distinguish between authorized and unauthorized usage.
  - specify the controls to be imposed.

Command-Interpreter System

- One of the most important system program for an operating system is the command interpreter, which is the interface between the user and the OS.
- A command interpreter is the part of a computer operating system that understands and executes commands that are entered interactively by a human being or from a program. In some operating systems, the command interpreter is called the shell.
- Command interpreters serve many purposes and are more useful than graphical user interfaces in some cases. Details about these cases are given as follows:

- Command interpreters have a large range of commands and queries available for different operations. Also, it is much faster to type than to click as is done using graphical user interfaces..

What Is a Shell?

- A shell is a program that takes commands typed by the user and calls the operating system to run those commands.
- A shell is a program that acts as the interface between you and the Linux system, allowing you to enter commands for the operating system to execute.
- Shell accepts your instruction or commands in English and translate it into computers native binary language

Command-Interpreter System

- The command statements deals with the following
  - process creation and management
  - I/O handling
  - secondary-storage management
  - main-memory management
  - file-system access , protection and networking

Command-Interpreter System

- Modern use of Command Interpreters
- Many advanced users use powerful command interpreters even though graphical user interfaces are more common. Some of the systems where command interpreters are used are:
- PHP has a shell for interactive use which is called php-cli.
- Ruby has command shell for interactive use.
- Some Linux distributions have the bash implementation of the Unix shell.
- Junos and Cisco IOS routers are configured using command line interpreters.
- Windows has Windows Command Prompt with a CLI environment.

User Operating System Interface - GUI

- User-friendly **desktop** metaphor interface
  - Usually mouse, keyboard, and monitor
  - **Icons** represent files, programs, actions, etc
  - Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a **folder**)
  - Invented at Xerox PARC
- Many systems now include both CLI and GUI interfaces
  - Microsoft Windows is GUI with CLI "command" shell
  - Apple Mac OS X is "Aqua" GUI interface with UNIX kernel underneath and shells available
  - Unix and Linux have CLI with optional GUI interfaces (CDE, KDE, GNOME)

Touchscreen Interfaces

- Touchscreen devices require new interfaces
  - Mouse not possible or not desired
  - Actions and selection based on gestures
  - Virtual keyboard for text entry
- Voice commands.

Operating System Services

- Operating systems provide an environment for execution of programs and services to programs and users
- One set of operating-system services provides functions that are helpful to the user:
  - **User interface** - Almost all operating systems have a user interface (**UI**).
    - Varies between **Command-Line (CLI)**, **Graphics User Interface (GUI)**, **Batch**
  - **Program execution** - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
  - **I/O operations** - A running program may require I/O, which may involve a file or an I/O device

Operating System Services (Cont.)

- One set of operating-system services provides functions that are helpful to the user (Cont.):
  - **File-system manipulation** - The file system is of particular interest. Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.
  - **Communications** – Processes may exchange information, on the same computer or between computers over a network
    - Communications may be via shared memory or through message passing (packets moved by the OS)

- **Error detection** – OS needs to be constantly aware of possible errors
  - May occur in the CPU and memory hardware, in I/O devices, in user program
  - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
  - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing

Resource allocation - When multiple users or multiple jobs running concurrently, resources must be allocated to each of them

- Many types of resources - CPU cycles, main memory, file storage, I/O devices.

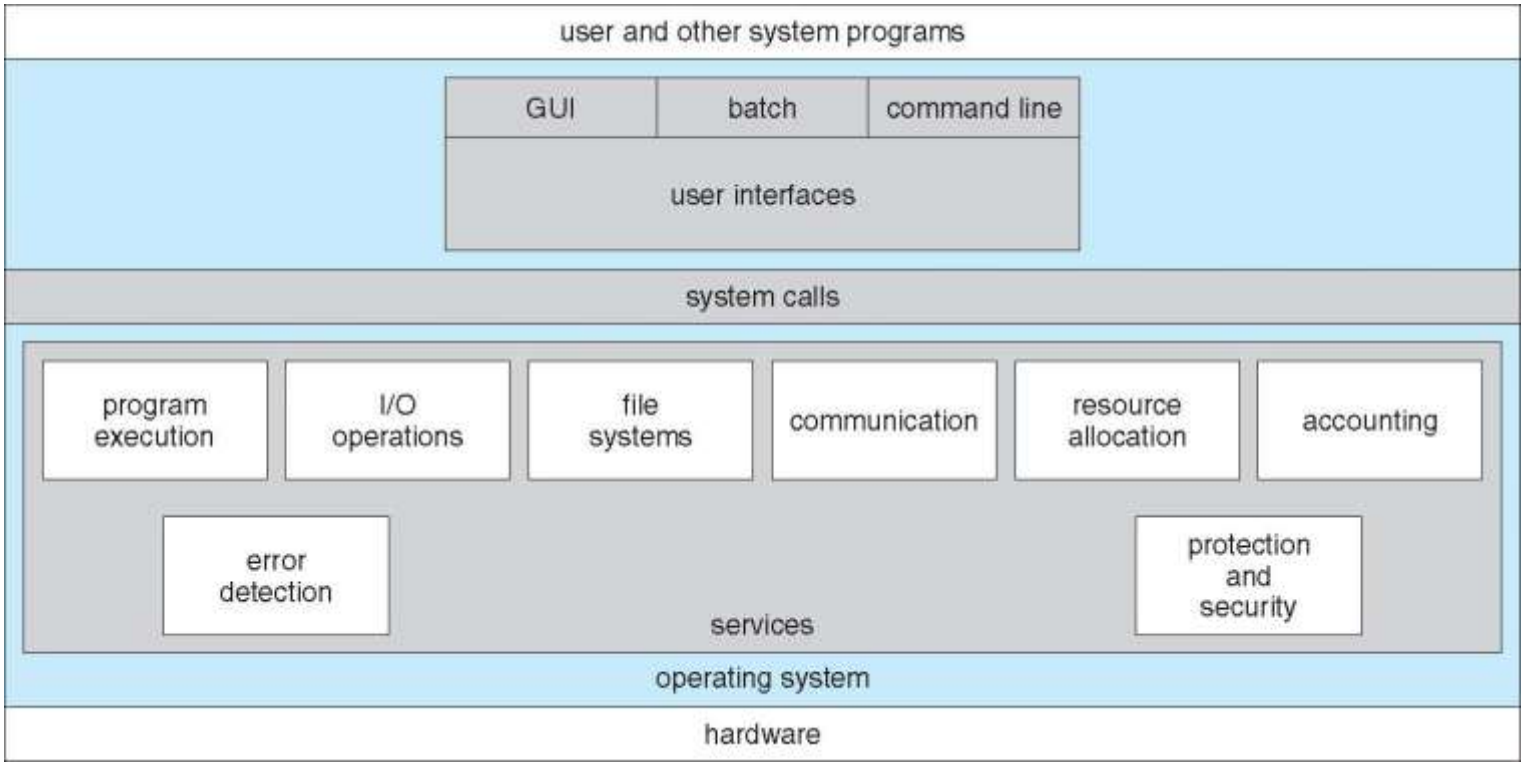
Operating System Services (Cont.)

Accounting - To keep track of which users use how much and what kinds of computer resources

Protection and security - The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other

- **Protection** involves ensuring that all access to system resources is controlled
- **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

A View of Operating System Services



System calls

- In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on.
- A system call is a way for programs to **interact with the operating system**. A computer program makes a system call when it makes a request to the operating system's kernel.
- System call **provides** the services of the operating system to the user programs via Application Program Interface(API).
- It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. All programs needing resources must use system calls.

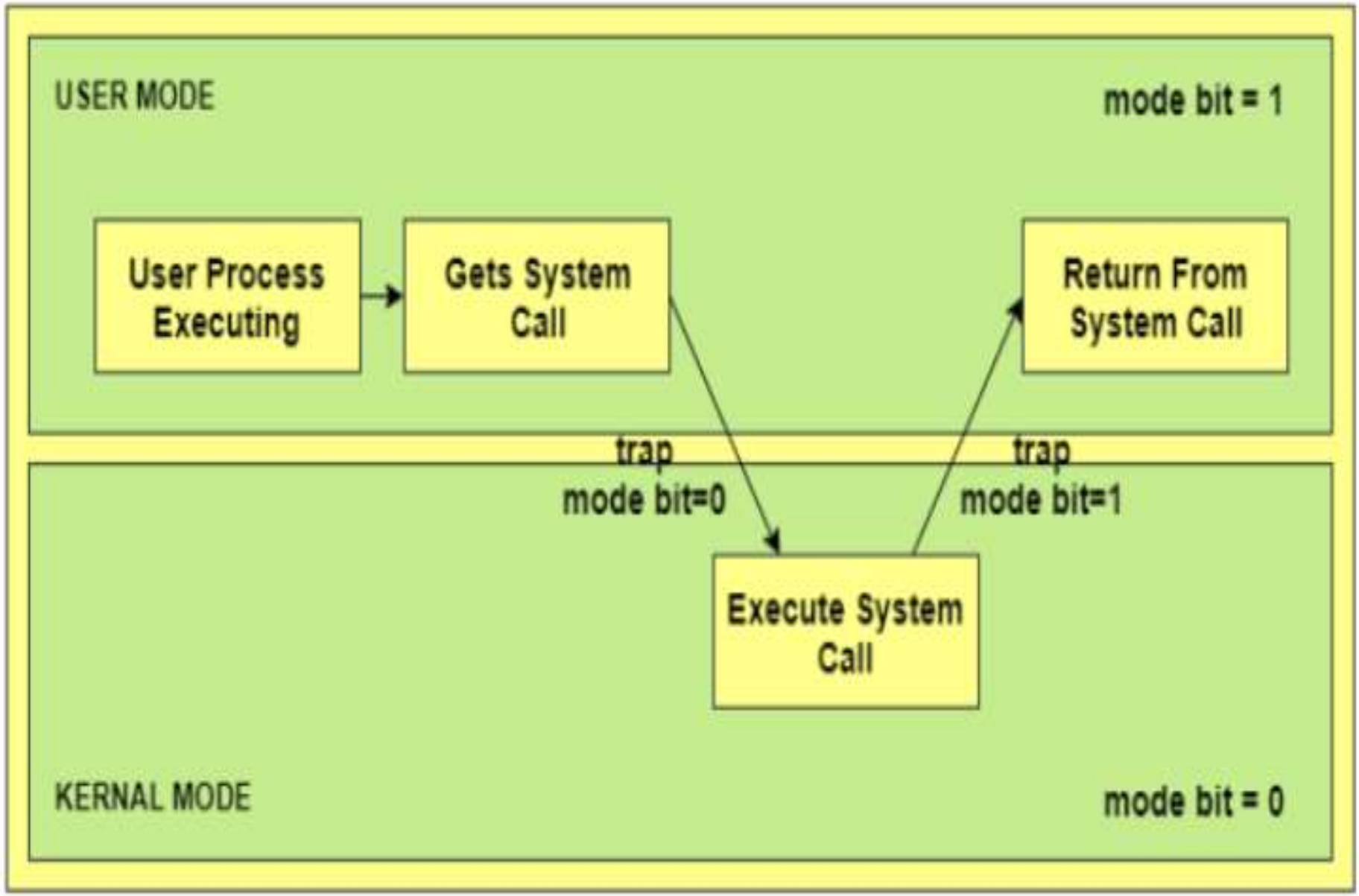
System Call Implementation

- Typically, a number associated with each system call
  - **System-call interface** maintains a table indexed according to these numbers
- The system call interface invokes the intended system call in OS kernel and returns status of the system call and any return values
- The caller need know nothing about how the system call is implemented
  - Just needs to obey API and understand what OS will do as a result call
  - Most details of OS interface hidden from programmer by API
  - Managed by run-time support library (set of functions built into libraries included with compiler)

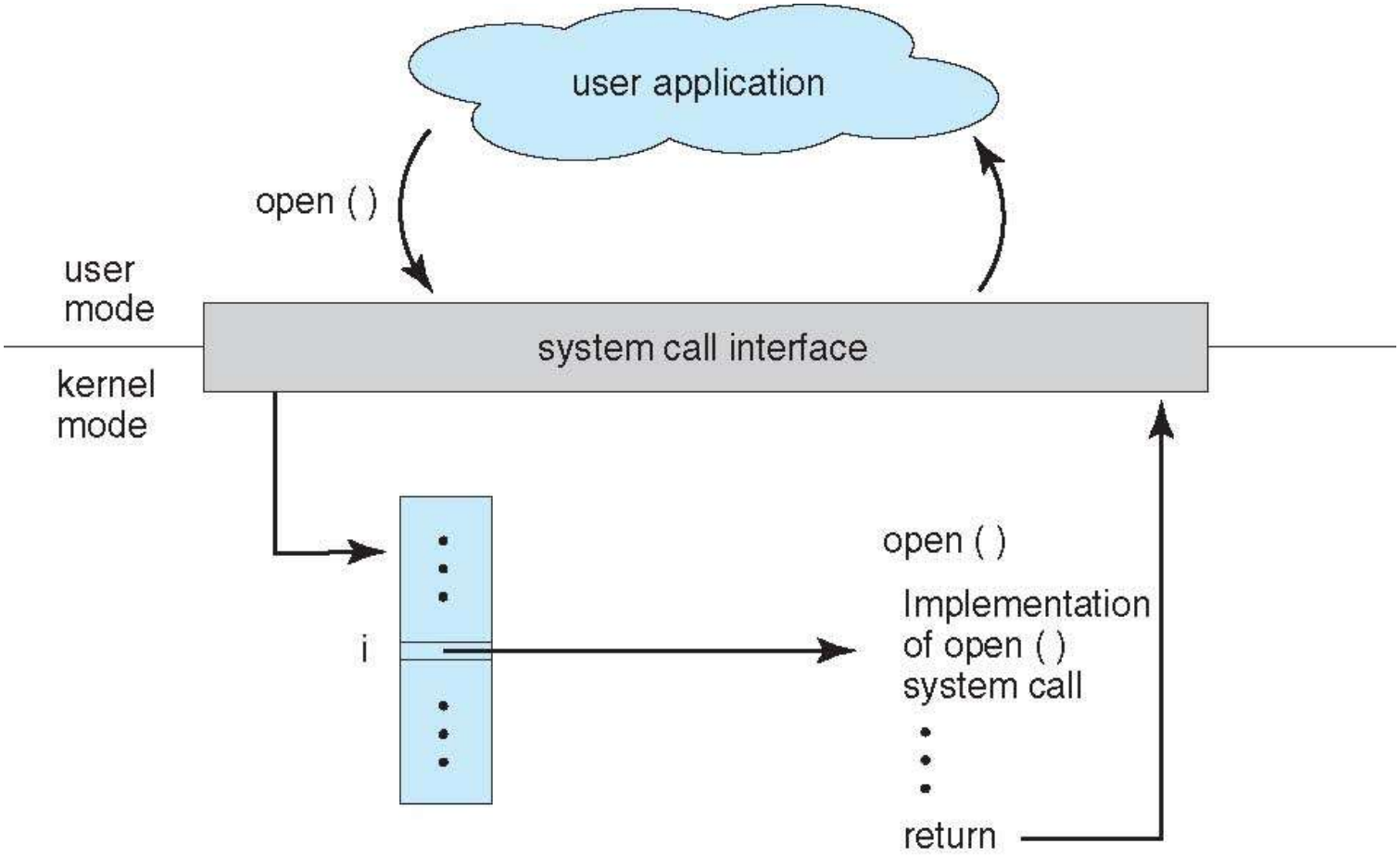
User Mode vs Kernel Mode

- **User Mode**
- The system is in user mode when the operating system is running a user application such as handling a text editor. The transition from user mode to kernel mode occurs when the application requests the help of operating system or an interrupt or a system call occurs.
- The mode bit is set to 1 in the user mode. It is changed from 1 to 0 when switching from user mode to kernel mode.

- **Kernel Mode**
- The system starts in kernel mode when it boots and after the operating system is loaded, it executes applications in user mode. There are some privileged instructions that can only be executed in kernel mode.
- These are interrupt instructions, input output management etc. If the privileged instructions are executed in user mode, it is illegal and a trap is generated.
- The mode bit is set to 0 in the kernel mode. It is changed from 0 to 1 when switching from kernel mode to user mod


USER MODE
mode bit = 1
User Process Executing
Gets System Call
Return From System Call
trap
mode bit=0
trap
mode bit=1
Execute System Call
KERNAL MODE
mode bit = 0

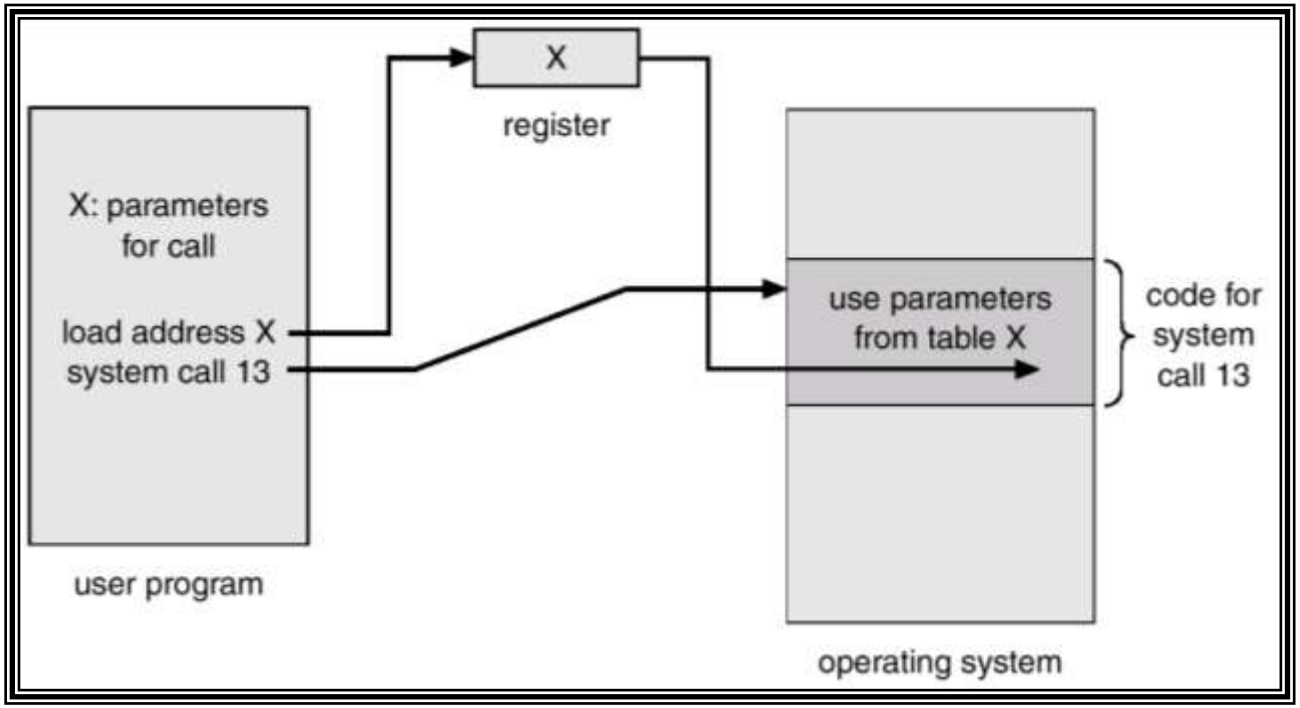
API – System Call – OS Relationship



System Call Parameter Passing

- Three general methods used to pass parameters to the OS
  - Simplest: pass the parameters in registers
    - In some cases, may be more parameters than registers
  - Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register
    - This approach taken by Linux and Solaris
  - Parameters placed, or **pushed**, onto the **stack** by the program and **popped** off the stack by the operating system
  - Block and stack methods do not limit the number or length of parameters being passed

Passing of Parameters As A Table



Types of System Calls

- Process control
  - End, abort, load, create, terminate, wait event, signal event, allocate and free memory
- File management
  - Create file, delete, open, close, read, write, get file attributes, set file attributes
- Device management
  - Request device, release device
- Information maintenance
  - Get time or date, set time, get process attributes, Set process attributes , get and set system data
- Communications
  - Send , receive message, transfer status information, create and delete communication connection

Services Provided by System Calls :

- Process creation and management
- Main memory management
- File Access, Directory and File system management
- Device handling(I/O)
- Protection
- Networking, etc

Examples of Windows and Unix System Calls

	Windows	Unix
Process Control	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
File Manipulation	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
	CloseHandle()	close()
Device Manipulation	SetConsoleMode()	ioctl()
	ReadConsole()	read()
	WriteConsole()	write()
Information Maintenance	GetCurrentProcessID()	getpid()
	SetTimer()	alarm()
	Sleep()	sleep()
Communication	CreatePipe()	pipe()
	CreateFileMapping()	shmget()
	MapViewOfFile()	mmap()
Protection	SetFileSecurity()	chmod()
	InitlializeSecurityDescriptor()	umask()
	SetSecurityDescriptorGroup()	chown()

System Programs

- System programs provide a convenient environment for program development and execution. They can be divided into:
  - File manipulation
  - Status information sometimes stored in a File modification
  - Programming language support
  - Program loading and execution
  - Communications
  - Background services
  - Application programs
- Most users' view of the operation system is defined by system programs, not the actual system calls

System Programs

- Provide a convenient environment for program development and execution
  - Some of them are simply user interfaces to system calls; others are considerably more complex
- **File management** - Create, delete, copy, rename, print, dump, list, and generally manipulate files and directories
- **Status information**
  - Some ask the system for info - date, time, amount of available memory, disk space, number of users
  - Others provide detailed performance, logging, and debugging information
  - Typically, these programs format and print the output to the terminal or other output devices
  - Some systems implement a **registry** - used to store and retrieve configuration information

System Programs (Cont.)

- **File modification**
  - Text editors to create and modify files
  - Special commands to search contents of files or perform transformations of the text
- **Programming-language support** - Compilers, assemblers, debuggers and interpreters sometimes provided
- **Program loading and execution**- Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language
- **Communications** - Provide the mechanism for creating virtual connections among processes, users, and computer systems
  - Allow users to send messages to one another's screens, browse web pages, send electronic-mail messages, log in remotely, transfer files from one machine to another

System Programs (Cont.)

- **Background Services**
  - Launch at boot time
    - Some for system startup, then terminate
    - Some from system boot to shutdown
  - Provide facilities like disk checking, process scheduling, error logging, printing
  - Run in user context not kernel context
  - Known as **services**, **subsystems**, **daemons**
- **Application programs**
  - Don't pertain to system
  - Run by users
  - Not typically considered part of OS
  - Launched by command line, mouse click, finger poke