SLIIT
Discover Your Future

IT2060/IE2061

Operating Systems and System Administration

Lecture 02

Introduction to Operating System

U. U. Samantha Rajapaksha

M.Sc.in IT, B.Sc.(Engineering) University of Moratuwa

Senior Lecturer SLIIT

Samantha.r@slit.lk

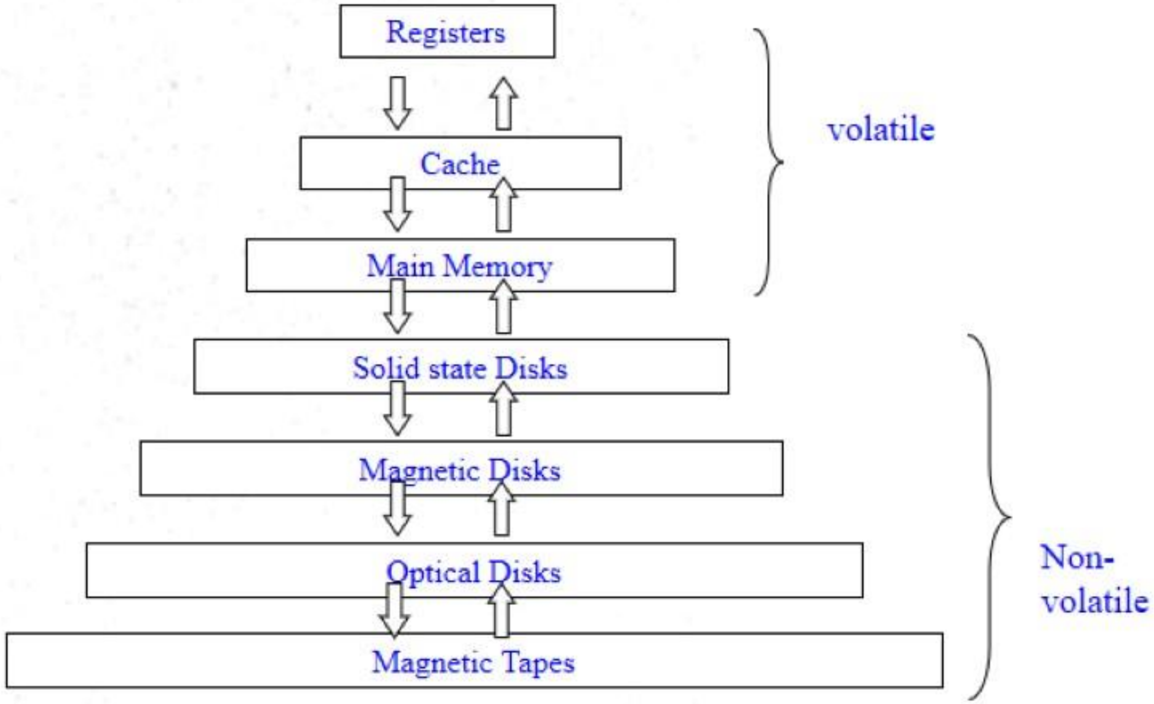
Content

- Storage Structure
- Booting Up process
- Multiprocessor System
- Interrupts handling
- Operating System Structures

Storage Structure

Storage systems are organized in hierarchy in terms of:

- Speed
- Cost
- Volatility
- Size/capacity



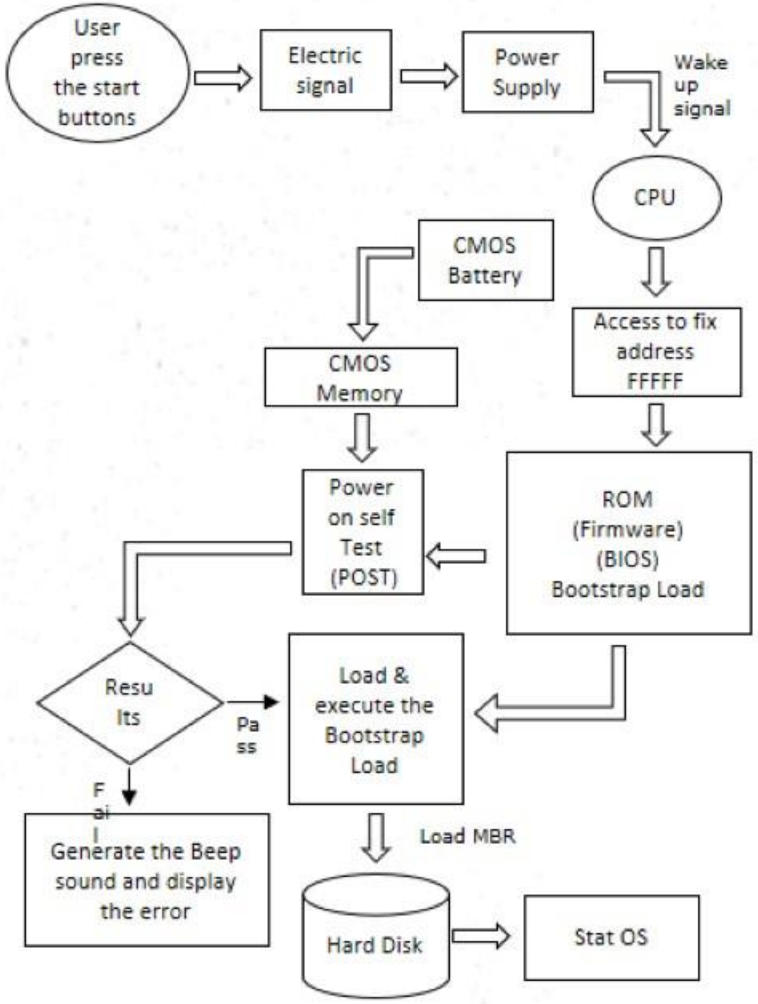
Performance of various levels of storage
Figure 1.11 (Textbook)

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Computer Startup

[redacted]

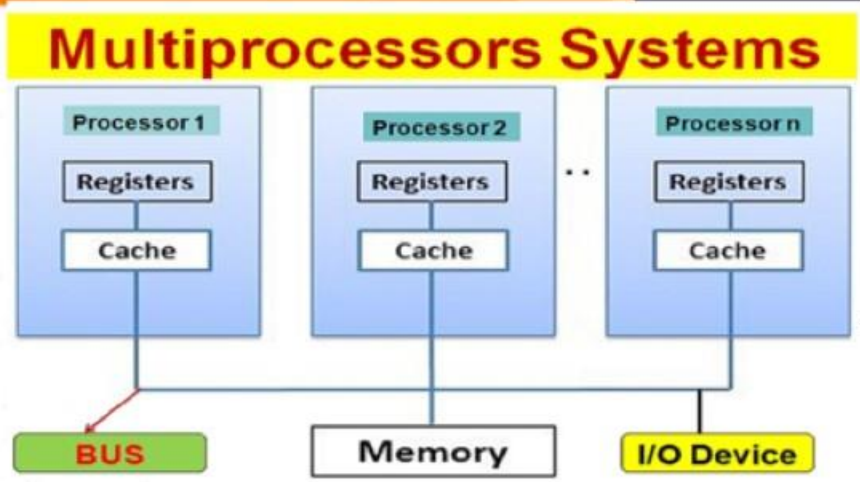
- **Booting Up the Computer:** Booting is a process or set of operations that loads and hence starts the operating system, starting from the point when user switches on the power button.
- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution


User press the start buttons
Electric signal
Power Supply
Wake up signal
CPU
Access to fix address FFFFF
CMOS Battery
CMOS Memory
Power on self Test (POST)
ROM (Firmware) (BIOS) Bootstrap Load
Resu lts
Pa ss
F ai l
Load & execute the Bootstrap Load
Generate the Beep sound and display the error
Load MBR
Hard Disk
Stat OS

Computer-System Architecture

- A single-processor system contains only one CPU to execute general purpose instructions
  - However, it also contains special purpose processors
    - E.g., disk, keyboard, DMA, graphic controllers
    - These specialized processors do not run user processes
    - OS may be able to manage the processors, e.g., task disk controllers to use given scheduling algorithms.
- A multiprocessor system contains two or more processors working together
  - They may share computer bus, system clock, memory, I/O devices
  - Also called parallel system or multicore system

Multiprocessor (cont.)

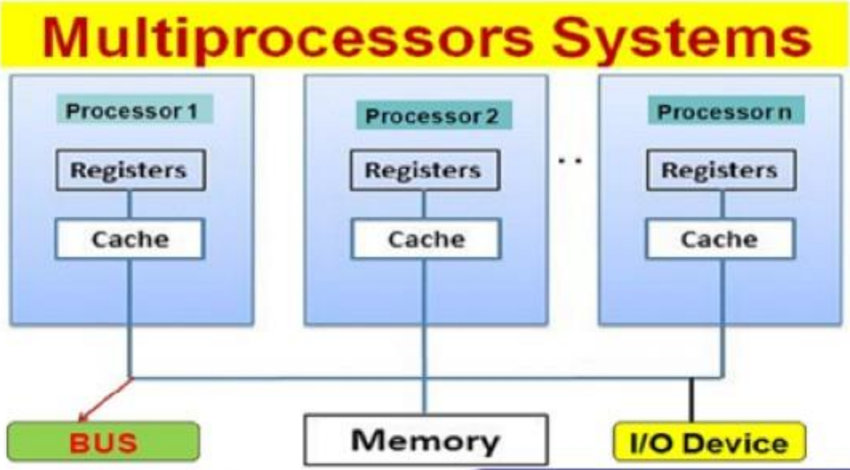


•Three main advantages of multiprocessor system

- **Increased throughput**: get more work done in less time
  - The speed up in *n* processor system is NOT *n* due to overhead
- **Economy of scale**: I/O devices, memory storage, and power supplies can be shared
- **Increased reliability**: failure of one processor does not make the whole system down
  - **Graceful degradation:** the system can perform operations proportional to the level of operations of the surviving parts of the system
  - **Fault-tolerant:** the system can continue its function in the event of component failures
    - Require failure detection, diagnoses, and even correction
    - May use multiple hardware and software duplicates to execute the same tasks in parallel, and take as output the result from the majority of the duplicates

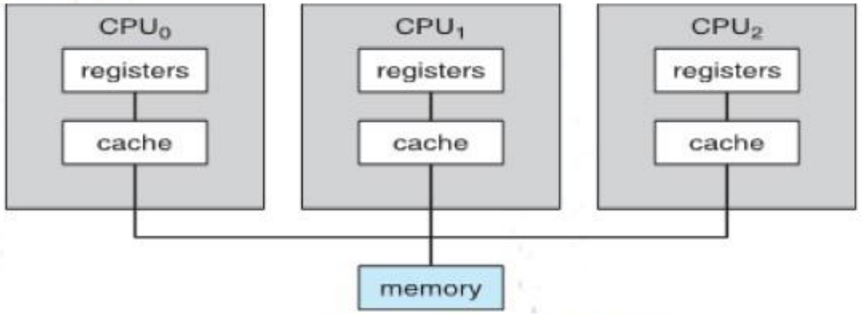
Multiprocessor (cont.)

- Two types of multiprocessor system:
  - Asymmetric multiprocessing
  - Symmetric multiprocessing (SMP)
- Asymmetric multiprocessing:
  - Use a **master** processor to schedule and allocate work to (**slave**) processors.
  - Each slave processor waits for instruction from the master or has predefined task
  - More common in extremely large systems



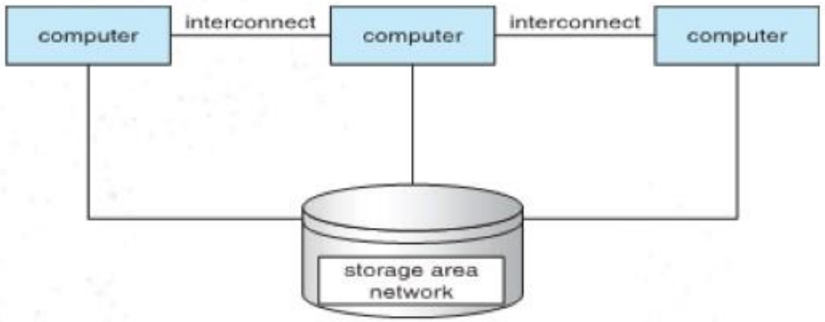
Multiprocessor Systems (cont.)

- SMP – not the master-slave model; a more common system
  - Each processor runs an identical copy of the OS.
  - Each processor has its own registers and cache
    - However, they share the same memory
  - Many processes can run at once without significant performance deterioration
    - Need load balancing to improve performance
- Symmtric and Asymmetric may be the result of either hardware or software.



Clustered Systems

- A clustered system consists of multiple CPUs, like multiprocessors
  - However they are individual systems or nodes
  - Each node can be a single processor or a multicore
  - They share storage and communicate via LAN
  - It offers high-availability service

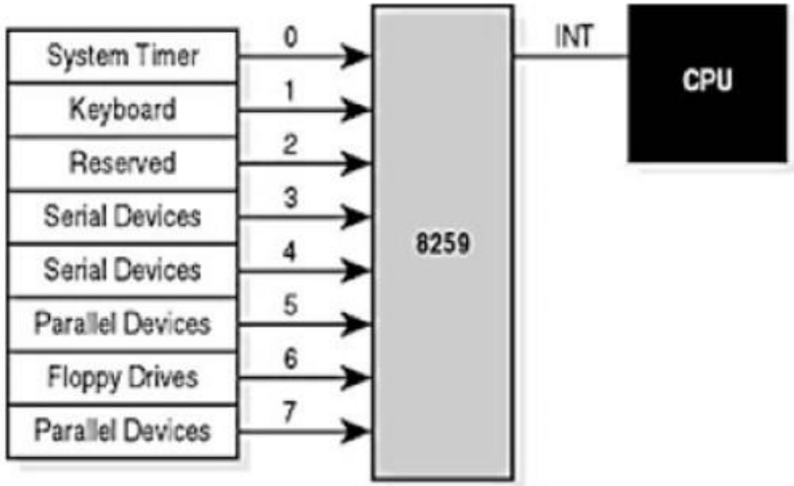


Interrupts

- **Interrupts** are signals sent to the CPU by external devices, normally I/O devices. They tell the CPU to stop its current activities and execute the appropriate part of the operating system.

There are three types of interrupts:

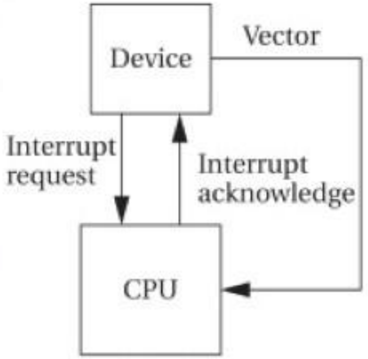
- **Hardware Interrupts** are generated by hardware devices to signal that they need some attention from the OS. They may have just received some data (e.g., keystrokes on the keyboard or an data on the ethernet card); or they have just completed a task which the operating system previous requested, such as transfering data between the hard drive and memory.
- **Software Interrupts** are generated by programs when they want to request a [system call]() to be performed by the operating system.
- **Traps** are generated by the CPU itself to indicate that some error or condition occured for which assistance from the operating system is needed.



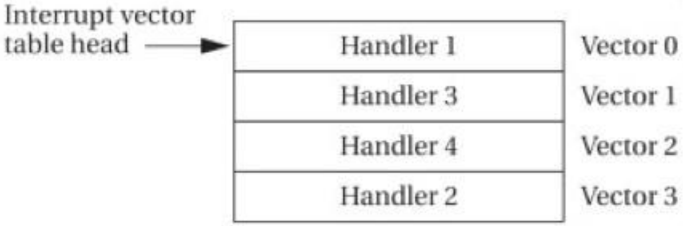
Interrupt Handling

Common functions of interrupts

- An interrupt is an event that suspends the execution of one program and begins the execution of another program.
- For each type of interrupt, separate segments of code in the OS must determine what action should be taken
- This code is called **INTERRUPT SERVICE ROUTINE.**
- Associated with each I/O device there is a location near the bottom of memory called **INTERRUPT VECTOR**
- This contains the address of the interrupt service routine for the various devices.



Hardware structure

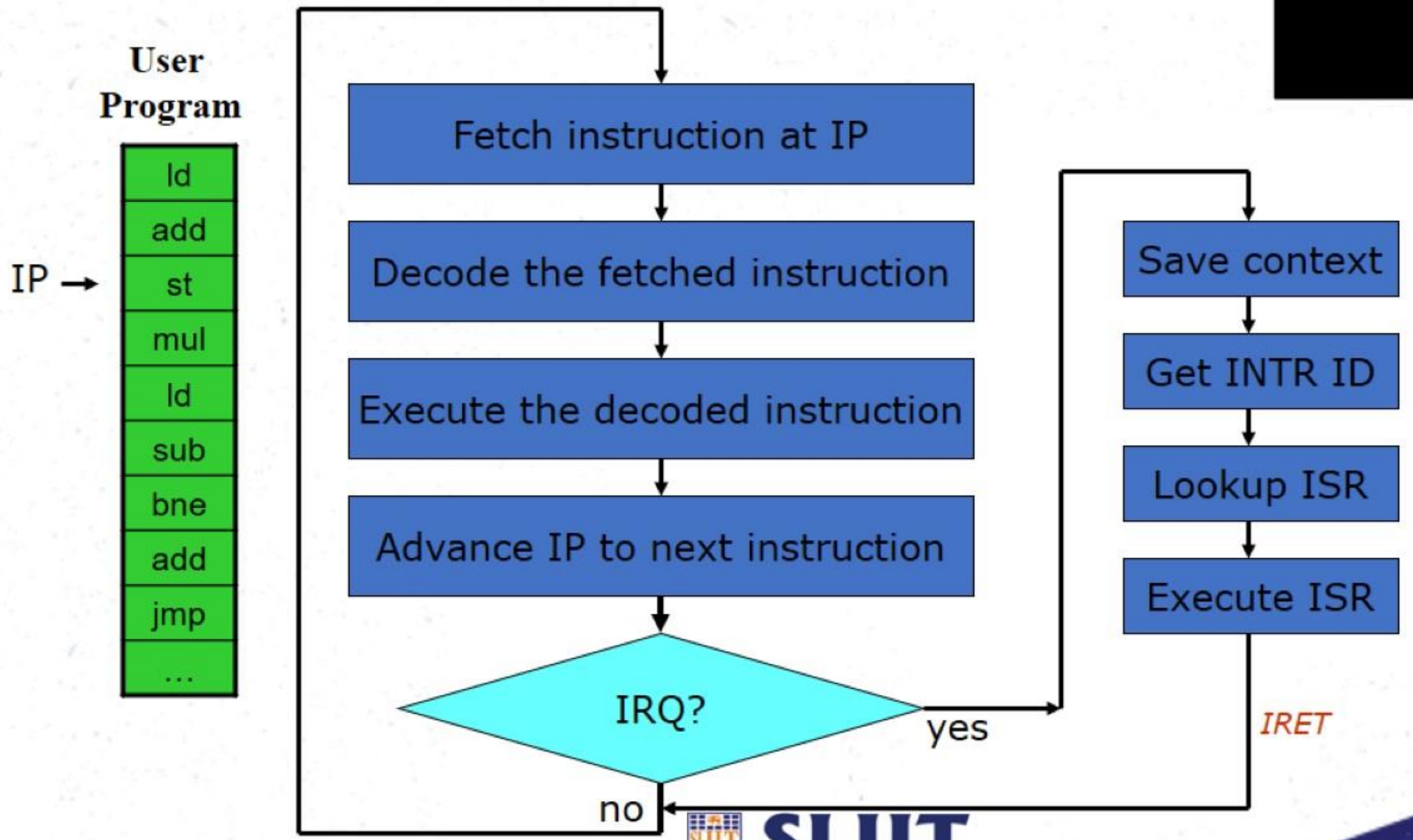


Interrupt vector table

Interrupt Handling

- When an interrupt (or trap) , hardware transfers control to OS
- The OS preserves the state of the CPU by storing registers and the Program Counter.
- Separate segments of code (Interrupt Service Routine) determine what action should be taken for each type of interrupt.

CPU's 'fetch-execute' cycle



Intel Pentium interrupt vector table

Figure 13.4 (textbook)

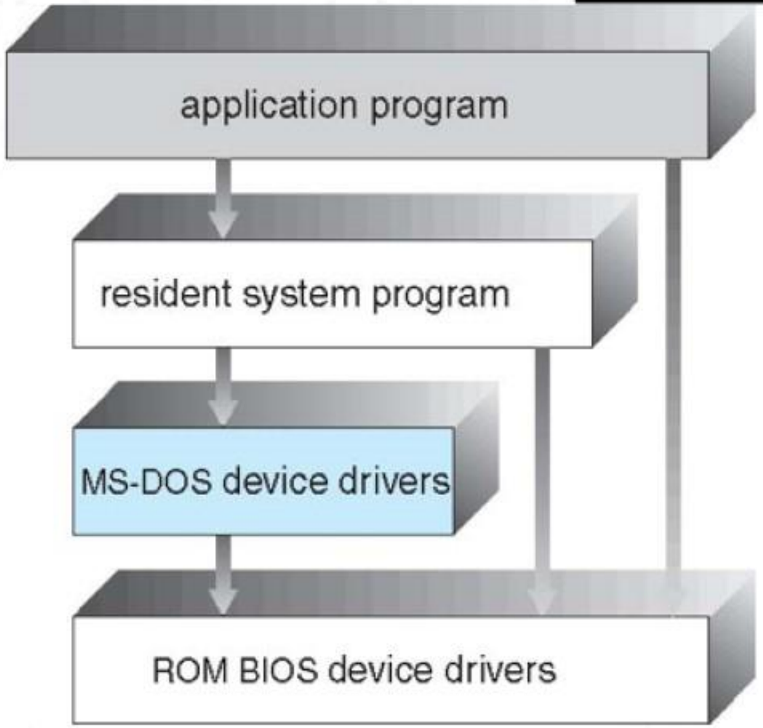
vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Operating System Structure

- General-purpose OS is very large program
- Various ways to structure ones
  - Simple structure – MS-DOS
  - More complex -- UNIX
  - Layered – an abstrcation
  - Microkernel -Mach

Simple Structure -- MS-DOS

- MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated



[redacted]

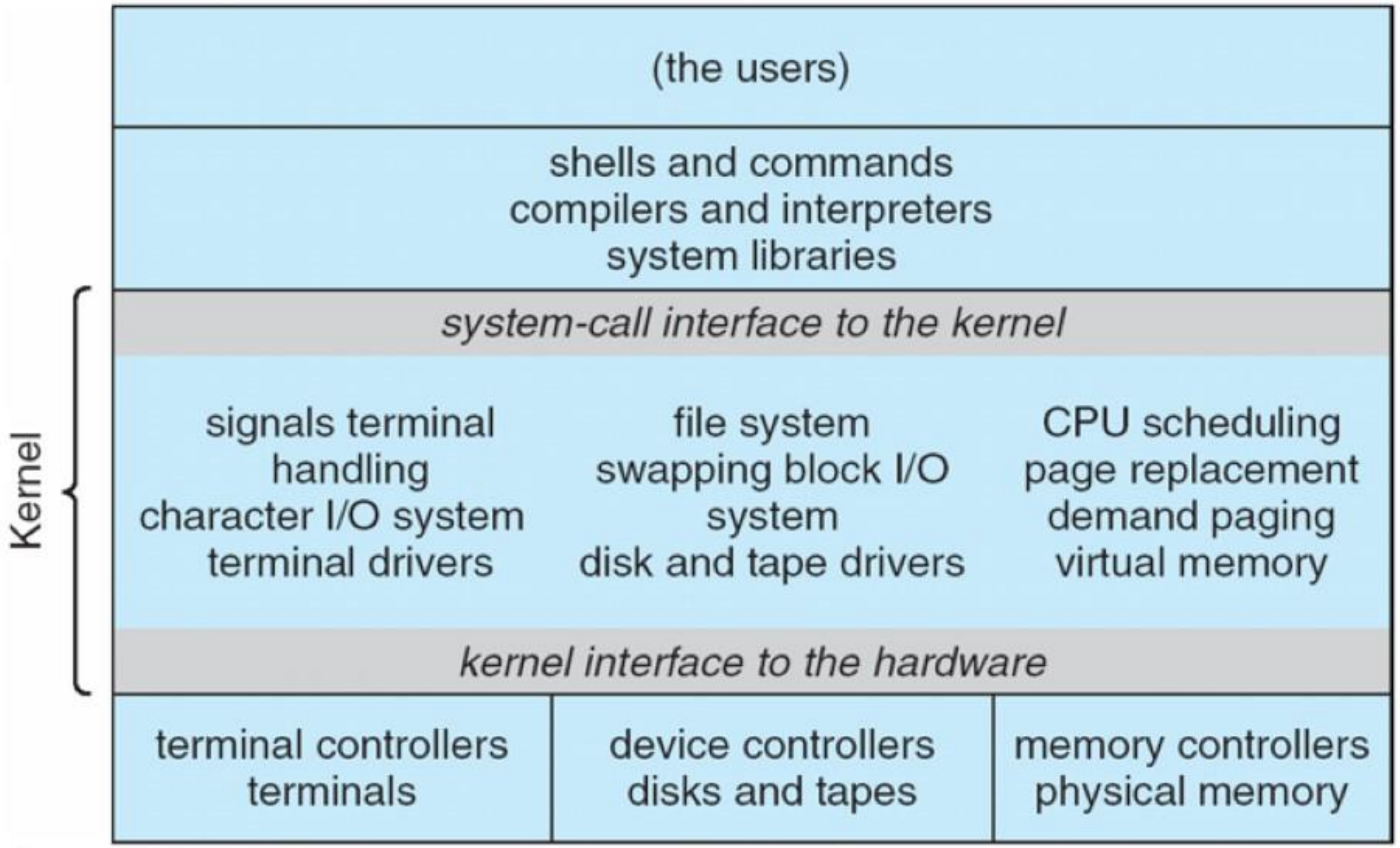
Non Simple Structure -- UNIX

UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts

- Systems programs
- The kernel
  - Consists of everything below the system-call interface and above the physical hardware
  - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

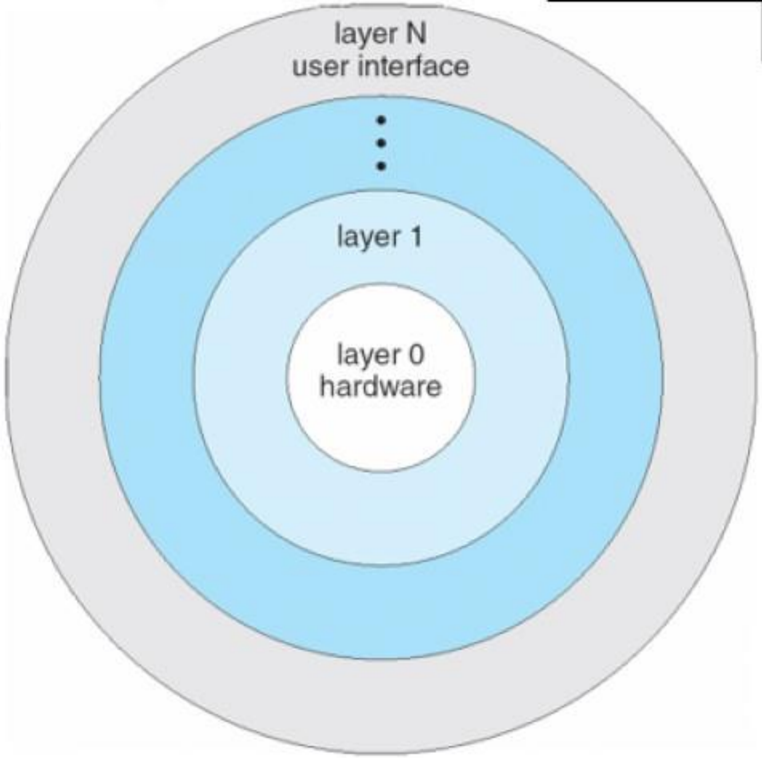
Traditional UNIX System Structure

Beyond simple but not fully layered



Layered Approach

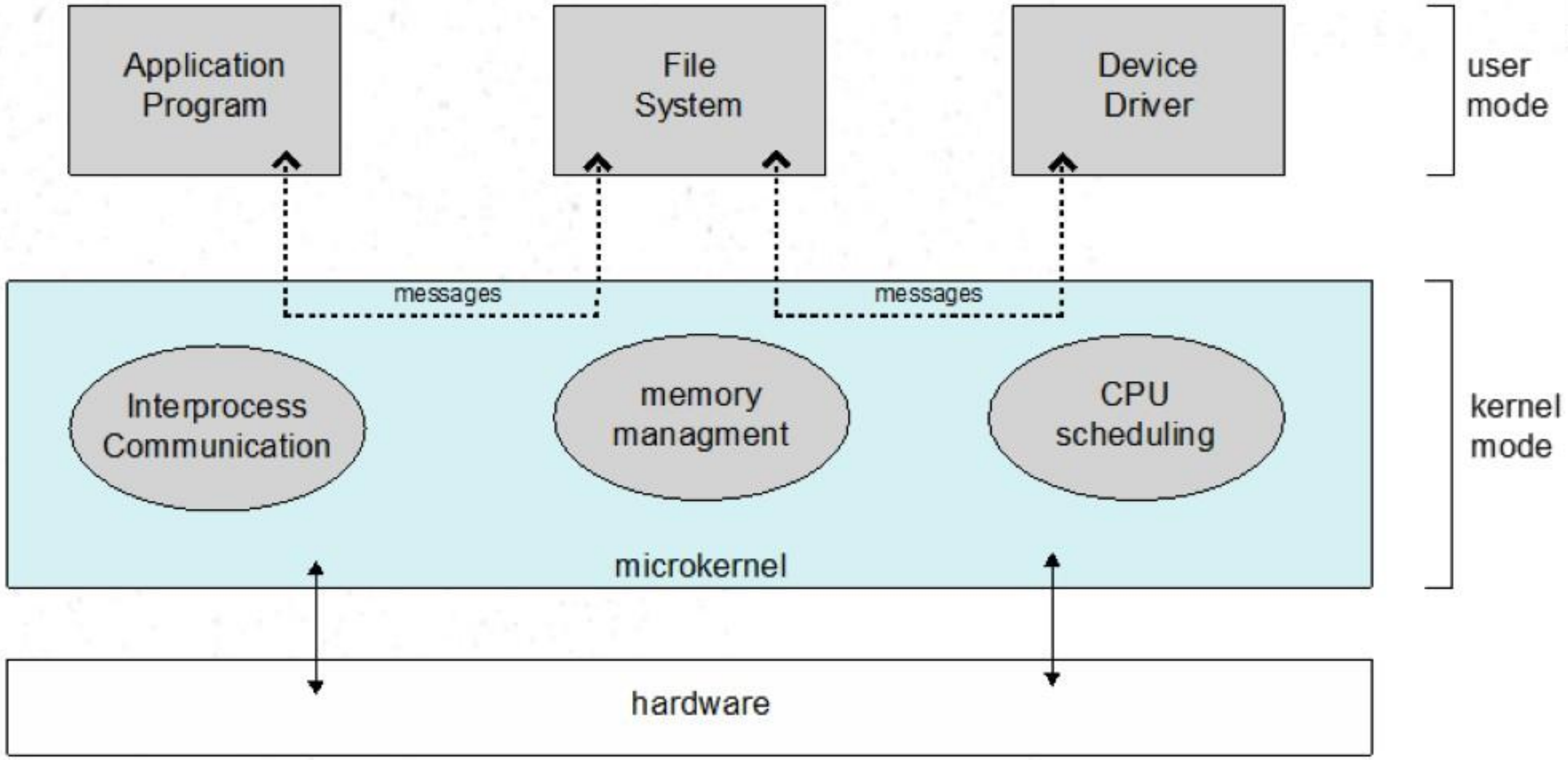
- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers



Microkernel System Structure

- Moves as much from the kernel into user space
- **Mach** example of **microkernel**
  - Mac OS X kernel (**Darwin**) partly based on Mach
- Communication takes place between user modules using **message passing**
- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Detriments:
  - Performance overhead of user space to kernel space communication

Microkernel System Structure

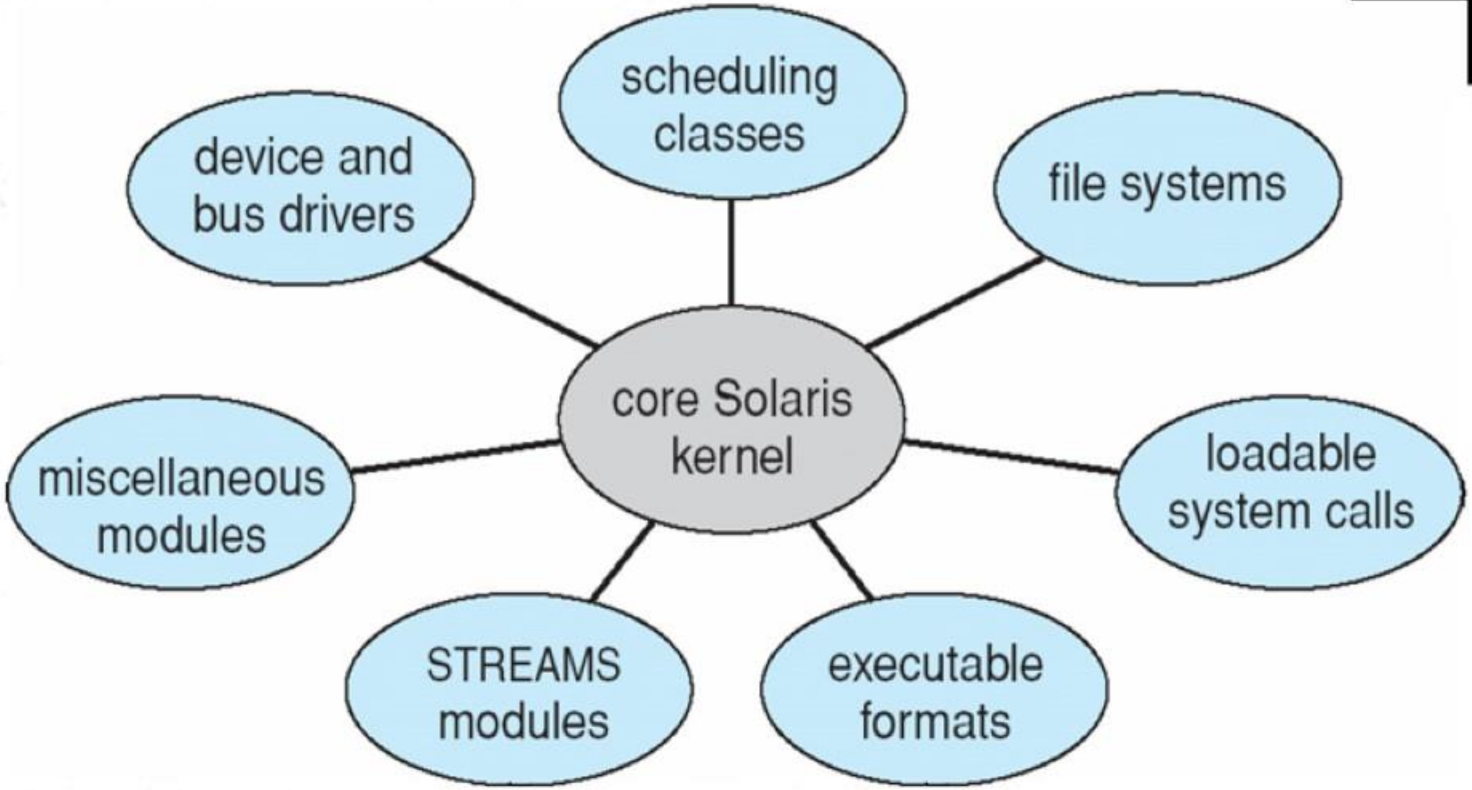


Modules

- Many modern operating systems implement **loadable kernel modules**
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexible
  - Linux, Solaris, etc

Solaris Modular Approach

[redacted]



Thank you very much