

Operating Systems and Databases

AE3B33OSD

RNDr. Petr Štěpán, Ph.D

Introduction

Operating System and Databases

Goal of course:

- To learn what is OS and how OS works
- To learn principles of OS design
- To learn algorithms and known solution for complicated problems
- Introduction to Databases
- How to use Databases

Material:

- <https://cw.fel.cvut.cz/wiki/courses/ae3b33osd>
- Book: Silberschatz A., Galvin P.B., Gange G.: Operating Systems Concepts — <http://codex.cs.yale.edu/avi/os-book/OS7/os7c/index.html>
- CS 162 – University of Berkeley, Youtube

Operating Systems and Networks

Examination:

- Lab exercise 10 points
- Test – quiz - select correct answer 8 points
- Test – 2 more general question 12 points

Topics for test will be listed on web.

Result:

- $A \geq 27$, $B \geq 24$, $C \geq 21$, $D \geq 18$, $E \geq 15$

Why Operating System?

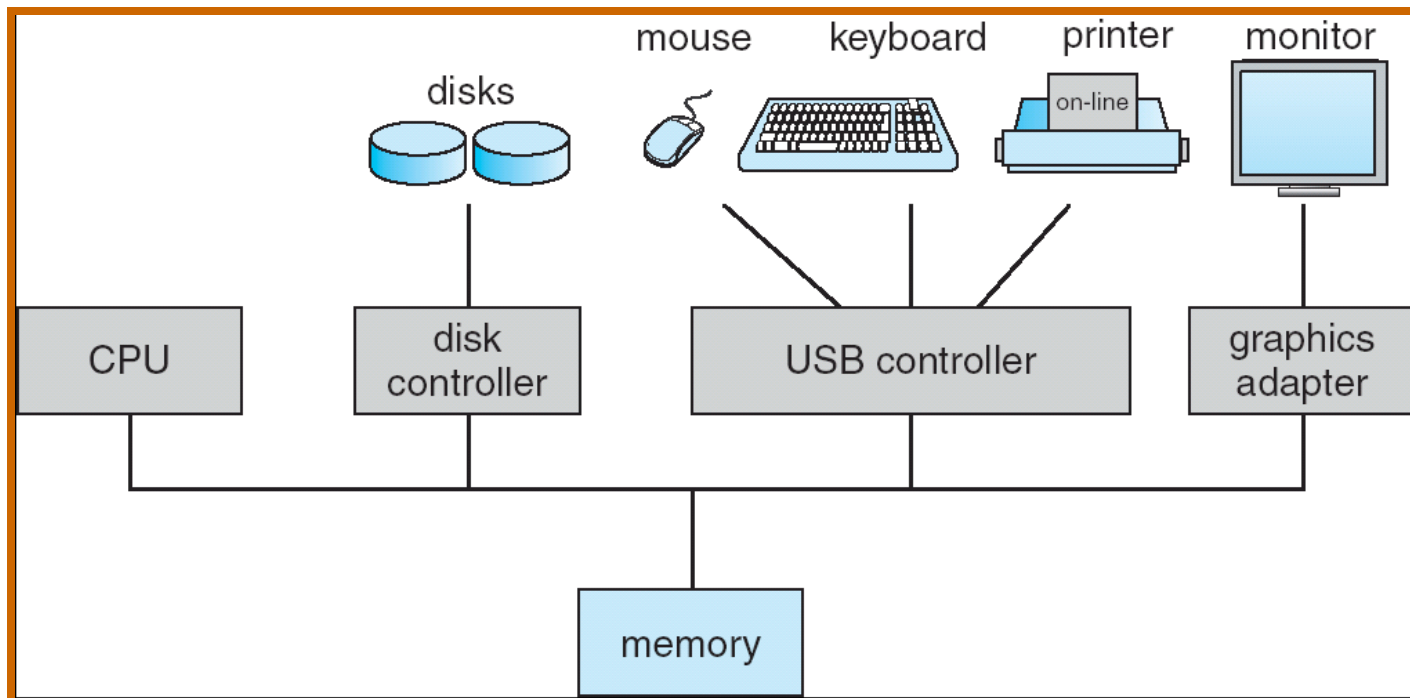
- OS is a program that acts as an intermediary between a user of a computer and a computer hardware.
- You cannot use PC without OS
- Operating system goals:
 - Execute user programs and make solving user problems easier.
 - Make the computer system convenient to use.
- Use the computer hardware in an efficient manner.
 - Exploits the hardware resources of one or more processors
 - Provides a set of services to system users
 - Manages memory storage and I/O devices

Where is Operating System?

- Operating system runs on a computer
- Operating system strongly depends on computer architecture
 - On CPU – type, number, instruction set
 - On bus – connection of components
 - On devices – drivers (programs that control the device)
- In this course we will suppose “general” operating system on “general” computer
- Some approaches will be documented on OS Linux, Windows, MacOS for PC like computer

Elements of General Computer

- Processor (one or more)
- Main Memory
 - Volatile, real memory or primary memory
- System bus
 - Communication among processors, memory, and I/O modules
- I/O modules
 - Secondary memory
 - Communications devices
 - Terminals
 - Printers ...



CPU - Top-Level Components

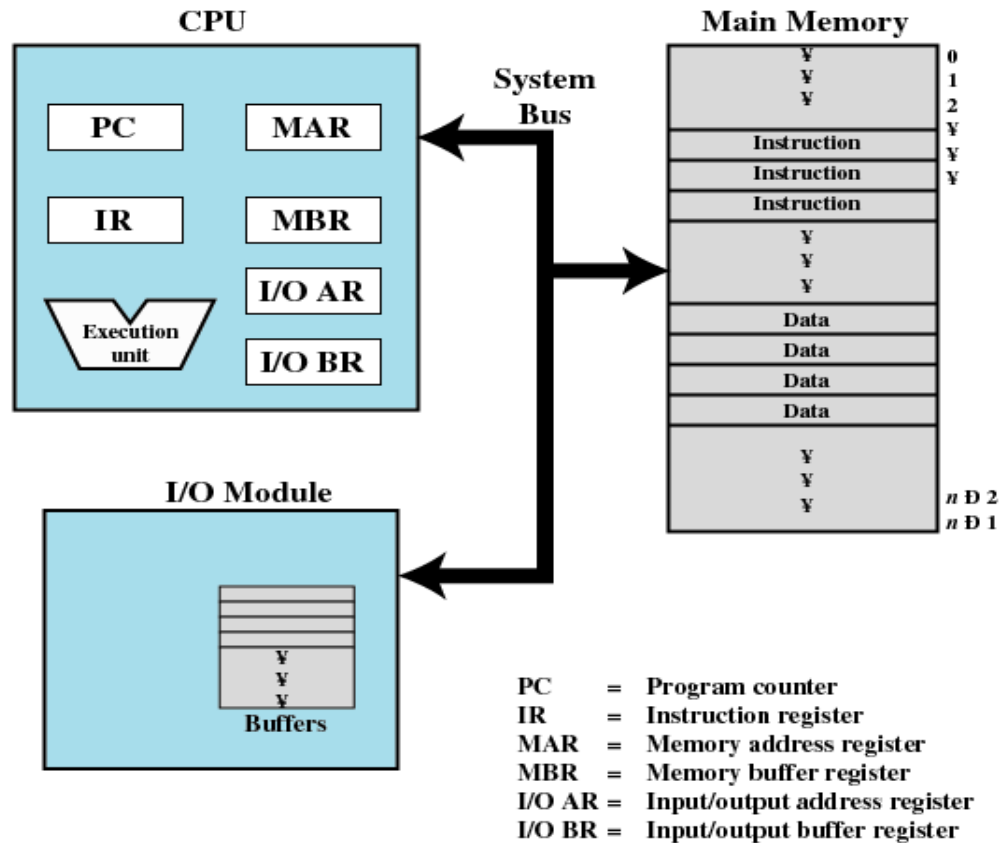


Figure 1.1 Computer Components: Top-Level View

Control and Status Registers

- Program Counter (PC)
 - Contains the address of an instruction to be fetched
- Instruction Register (IR)
 - Contains the instruction most recently fetched
- Program Status Word (PSW)
 - Condition codes
 - Interrupt enable/disable
 - Supervisor/user mode
 - Used by privileged operating-system routines to control the execution of programs

User-Visible Registers

- May be referenced by machine language
- RISC - Reduced Instruction Set Computing (ARM, Power) vs. CISC - Complex Instruction Set Computers (Intel)
- Available to all programs - application programs and system programs
- Types of registers
 - Data
 - Address
 - ▶ Index
 - ▶ Stack pointer

Interrupt Cycle

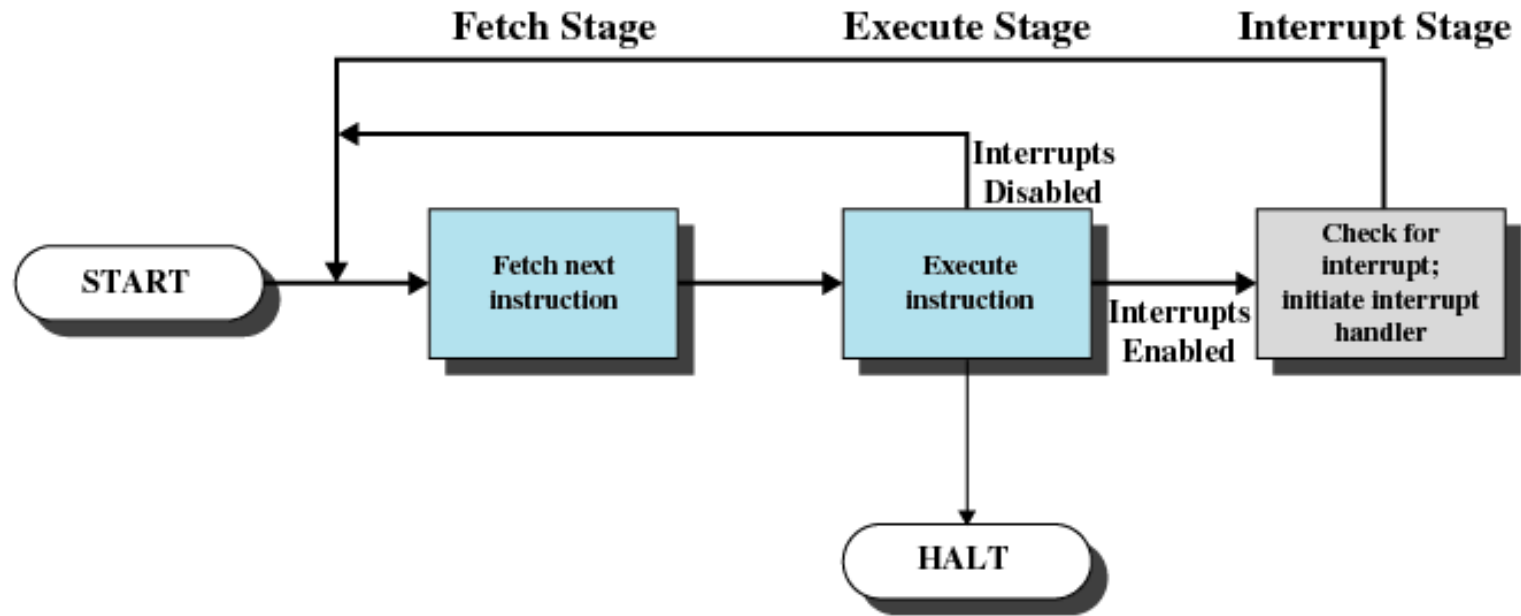


Figure 1.7 Instruction Cycle with Interrupts

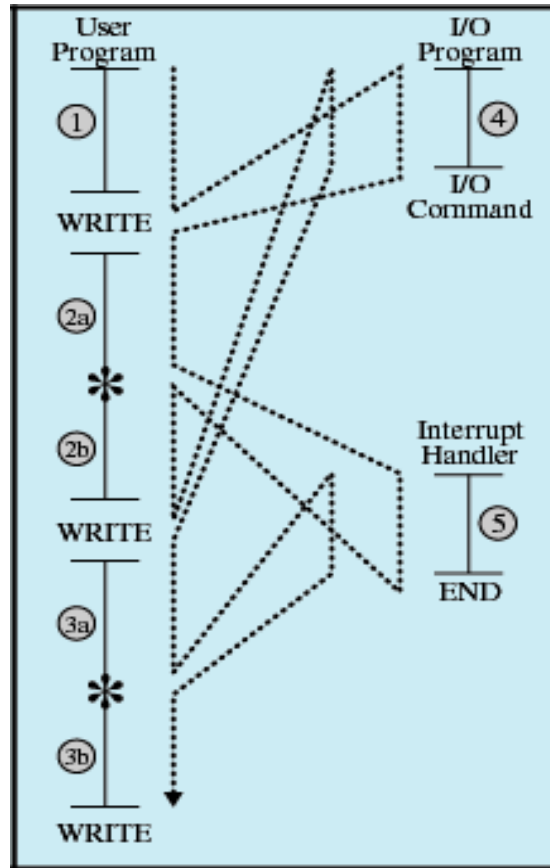
Interrupts

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

Table 1.1 Classes of Interrupts

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure, such as power failure or memory parity error.

Program Flow of Control With Interrupts, Short I/O Wait



(b) Interrupts; short I/O wait

Interrupts

- Interrupt handler:
 - Program to service a particular I/O device
 - Generally part of the operating system
 - Suspends the normal sequence of execution

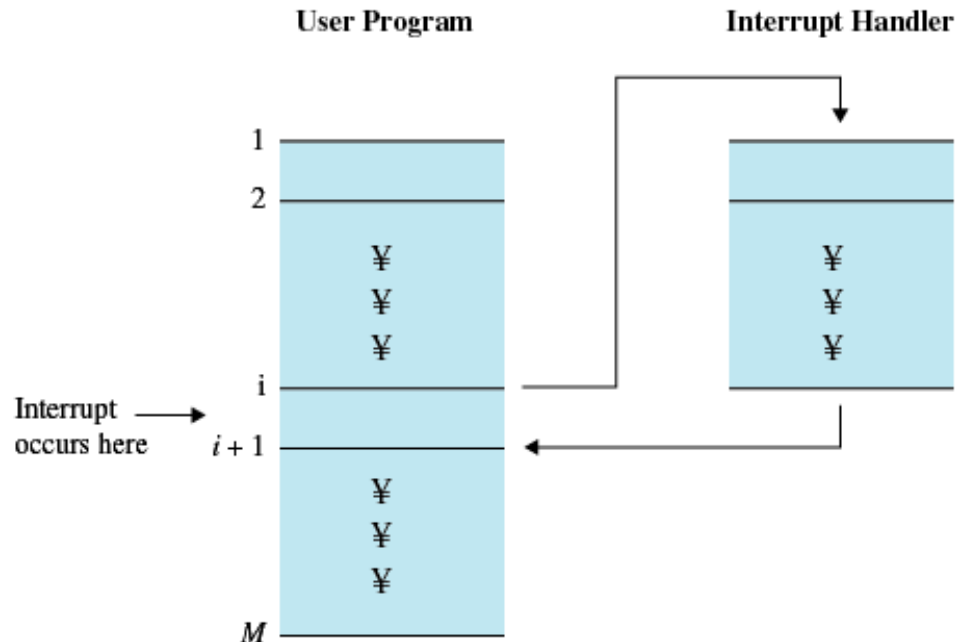


Figure 1.6 Transfer of Control via Interrupts

Simple Interrupt Processing

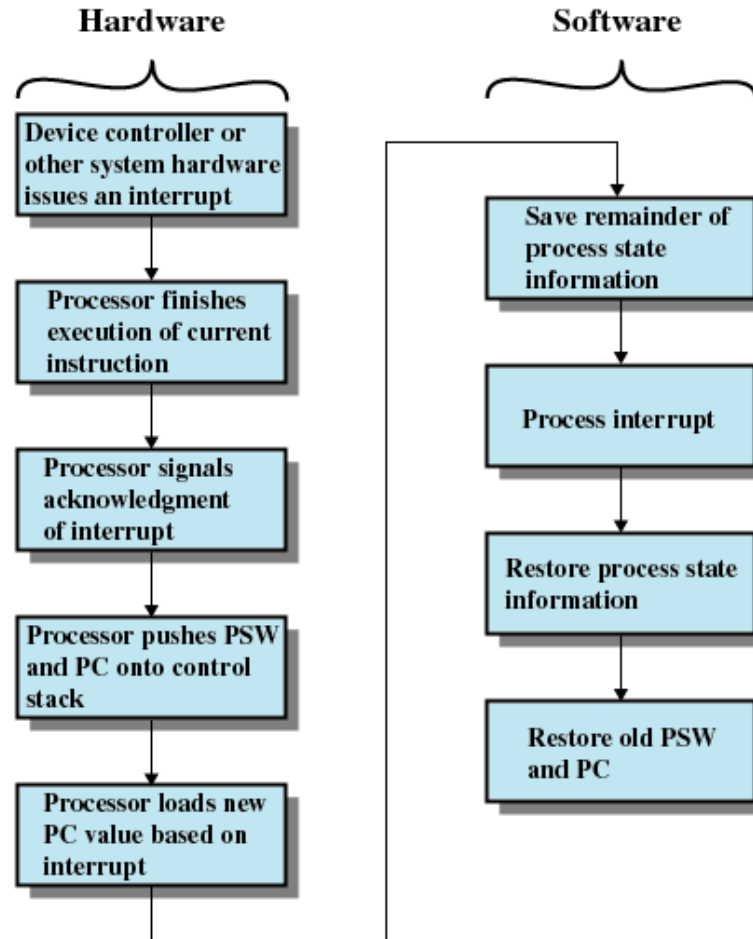
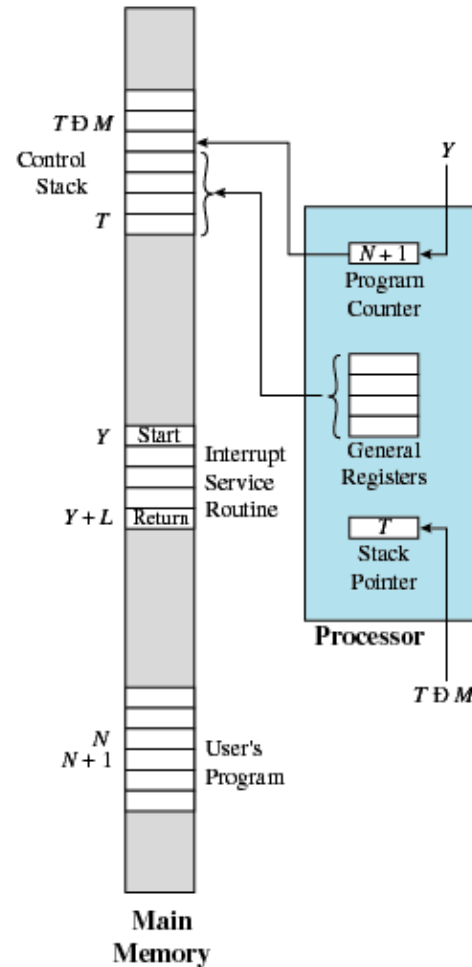


Figure 1.10 Simple Interrupt Processing

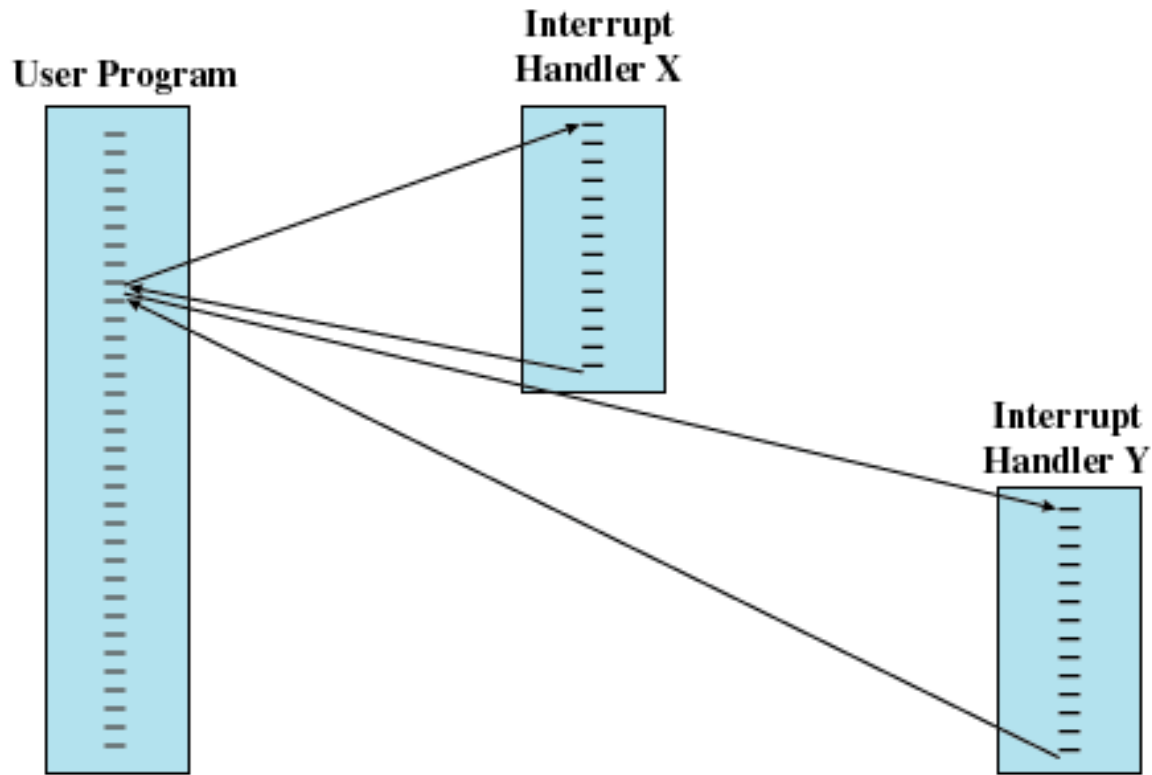
Changes in Memory and Registers for an Interrupt



(a) Interrupt occurs after instruction at location N

Multiple Interrupts

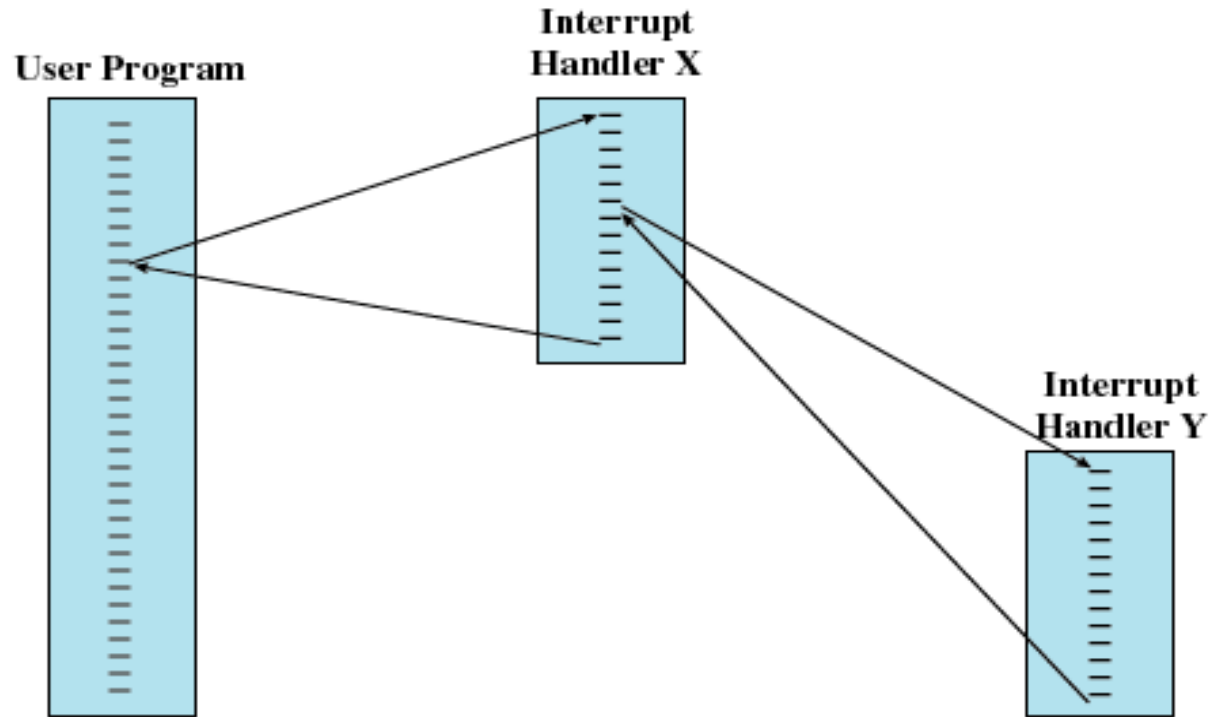
- Disable interrupts while an interrupt is being processed



(a) Sequential interrupt processing

Multiple Interrupts

- Define priorities for interrupts



(b) Nested interrupt processing

Memory Hierarchy

- Faster access time, greater cost per bit
 - Cache memory is fast but it is small because it is expensive
- Greater capacity, smaller cost per bit & slower access speed
 - DVD memory is cheap but the CPU need first to read data into main memory – it is slow

Memory Hierarchy

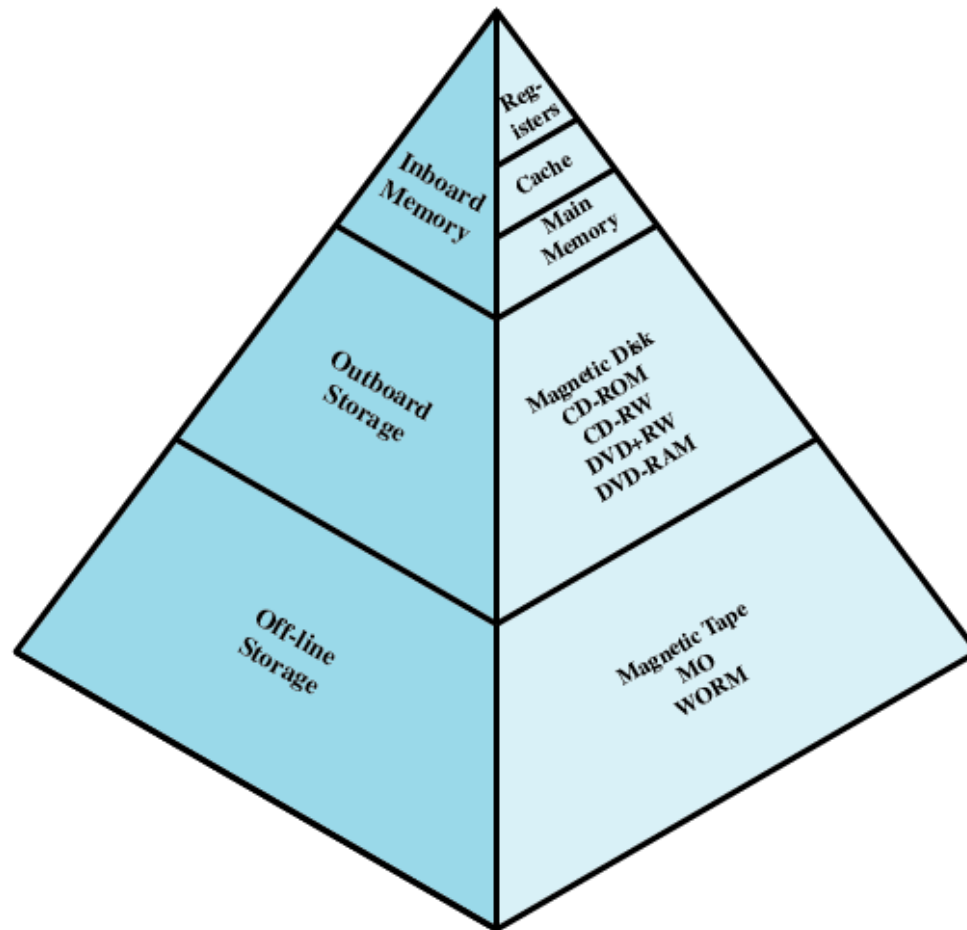


Figure 1.14 The Memory Hierarchy

Going Down the Hierarchy

- Decreasing cost per bit
- Increasing capacity
- Increasing access time
- Decreasing frequency of access of the memory by the processor
 - Locality of reference

Cache Memory

- Invisible to operating system
- Increase the speed of memory
- Processor speed is faster than memory speed
- Exploit the principle of locality

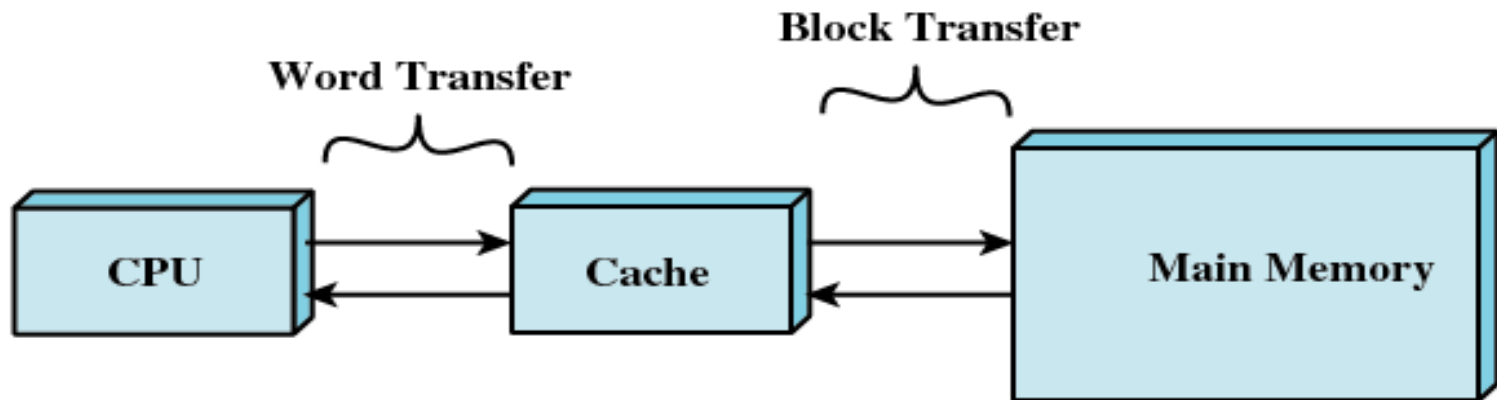


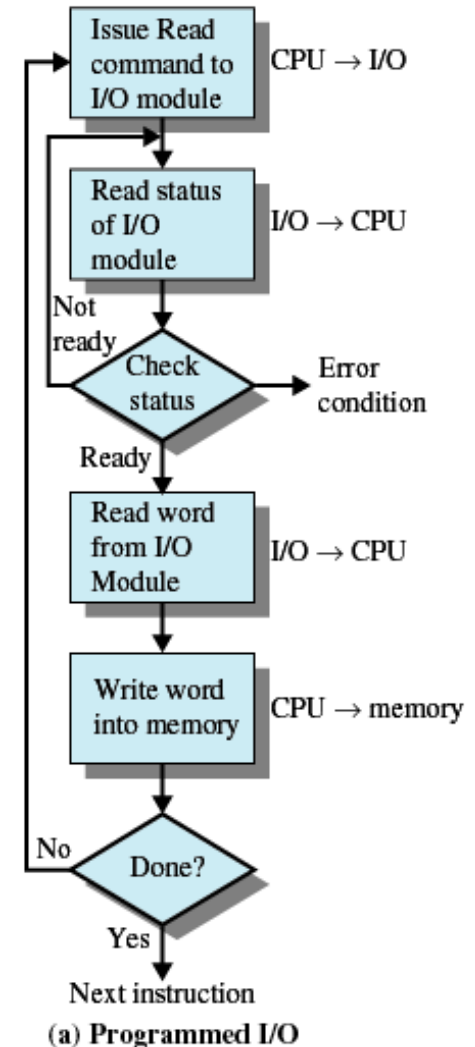
Figure 1.16 Cache and Main Memory

Cache Memory

- Contains a copy of a portion of main memory
- Processor first checks cache
- If not found in cache, the block of memory containing the needed information is moved to the cache and delivered to the processor

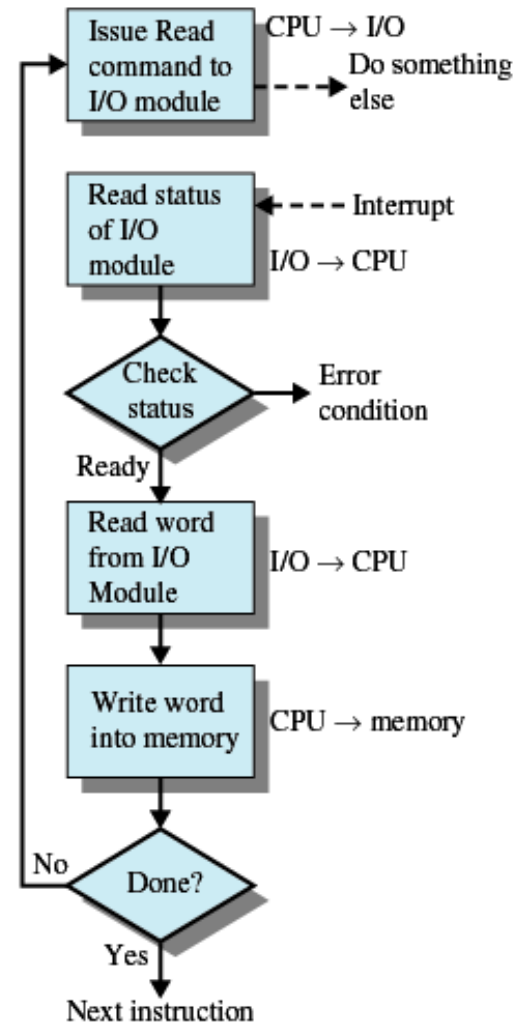
Programmed I/O

- I/O module performs the action, not the processor
- Sets appropriate bits in the I/O status register
- No interrupts occur
- Processor checks status until operation is complete



Interrupt-Driven I/O

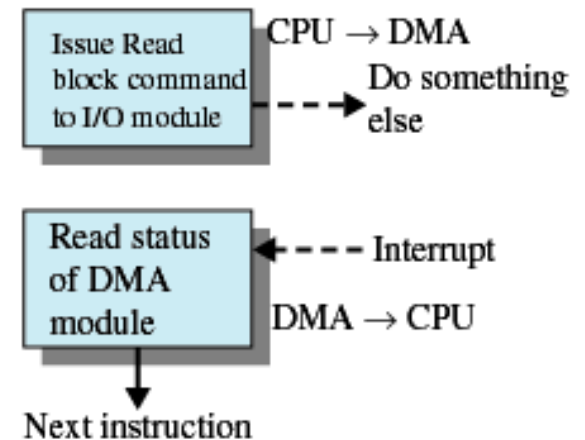
- Processor is interrupted when I/O module ready to exchange data
- Processor saves context of program executing and begins executing interrupt-handler
- No needless waiting
- Consumes a lot of processor time because every word read or written passes through the processor



(b) Interrupt-driven I/O

Direct Memory Access

- Transfers a block of data directly to or from memory
- An interrupt is sent when the transfer is complete
- Processor continues with other work

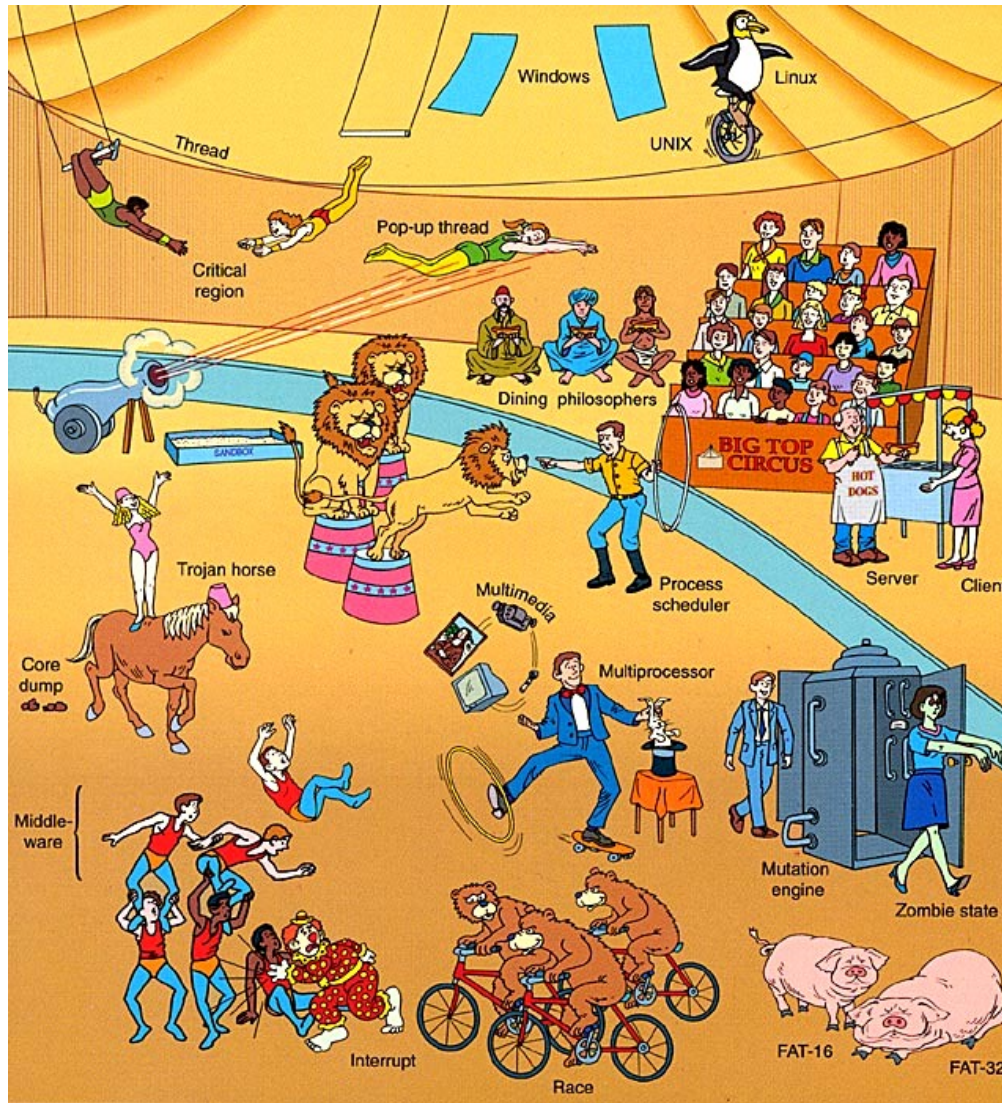


(c) Direct memory access

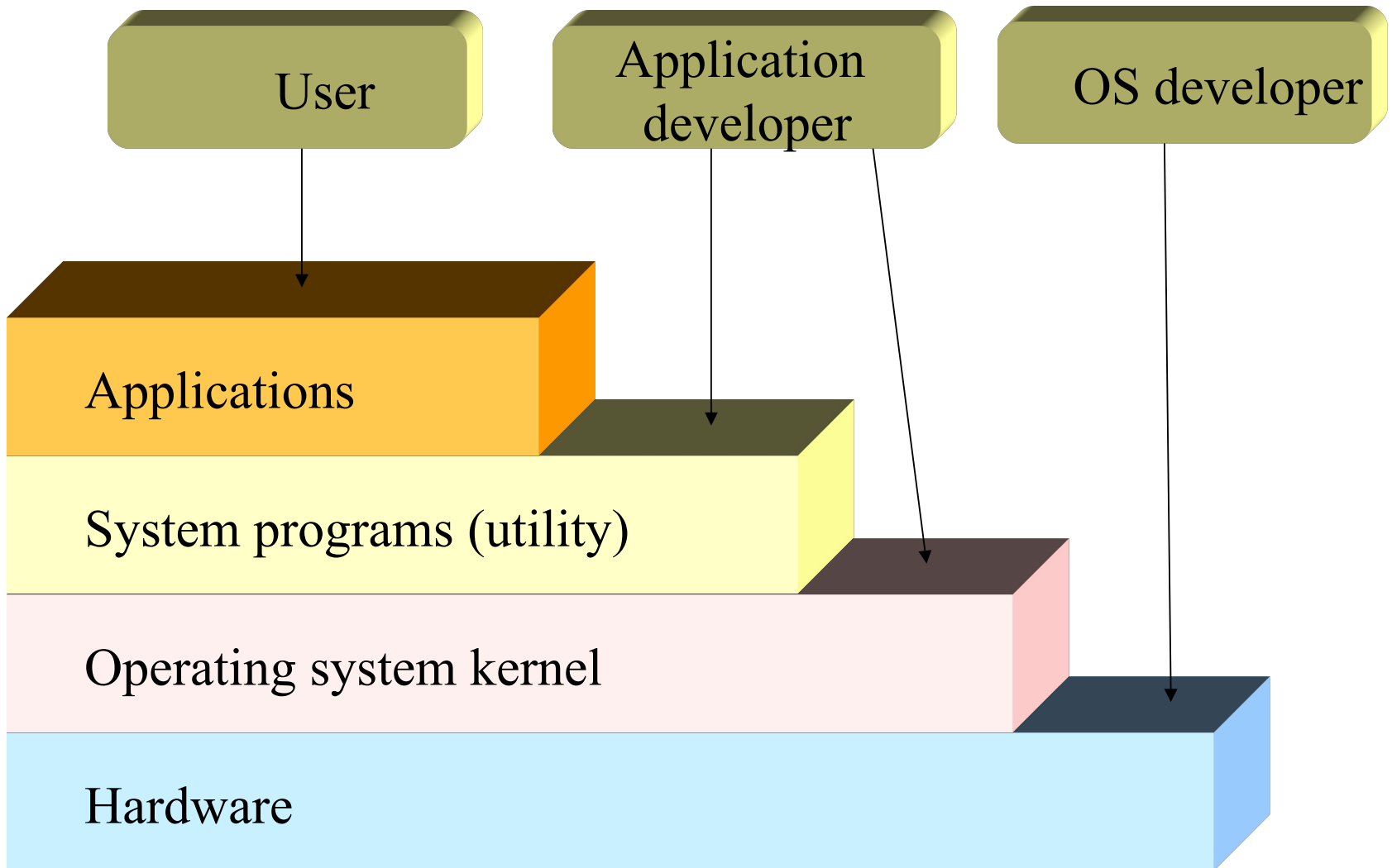
Direct Memory Access (DMA)

- I/O exchanges occur directly with memory
- Processor grants I/O module authority to read from or write to memory
- Relieves the processor responsibility for the exchange

What OS do?



Structure of computer



Operating System structure

Graphical user interface (GUI)

Command-line interface (CLI)

Networking

File system mng.

Storage mng.

I/O mng.

Memory mng.

Process mng.

not sys

not core p

OS

kernel

Process Management

A process is a program in execution. It is a unit of work within the system. Program is a *passive entity*, process is an *active entity*.

- Process needs resources to accomplish its task
 - CPU, memory, I/O, files
 - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
 - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
 - Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
 - Optimizing CPU utilization and computer response to users
- Memory management activities
 - Keeping track of which parts of memory are currently being used and by whom
 - Deciding which processes (or parts thereof) and data to move into and out of memory
 - Allocating and deallocating memory space as needed

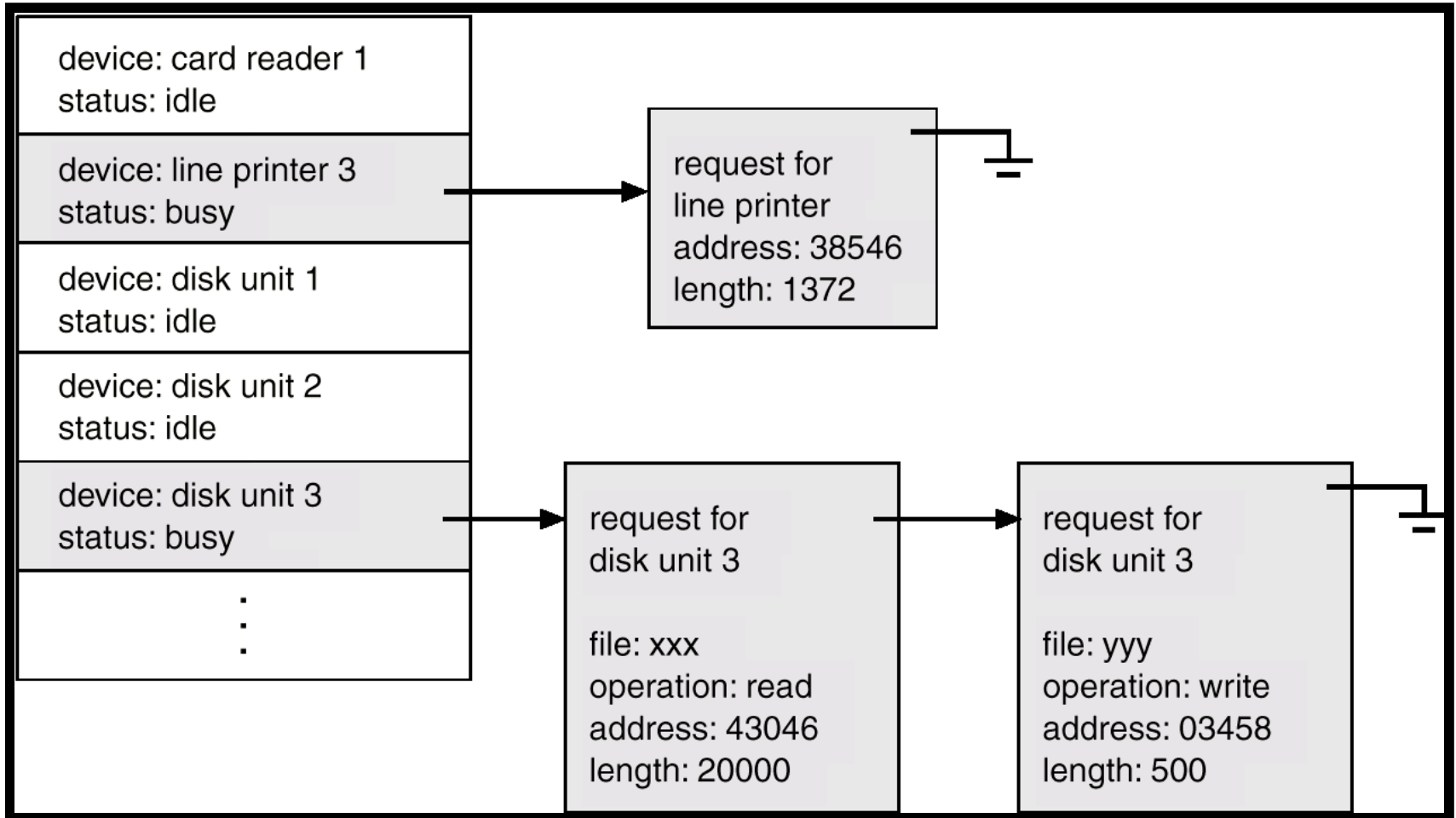
Storage Management

- OS provides uniform, logical view of information storage
 - Abstracts physical properties to logical storage unit – **file**
 - Each medium is controlled by device (i.e., disk drive, tape drive)
 - ▶ Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
 - Files usually organized into directories
 - Access control on most systems to determine who can access what
 - OS activities include
 - ▶ Creating and deleting files and directories
 - ▶ Primitives to manipulate files and dirs
 - ▶ Mapping files onto secondary storage
 - ▶ Backup files onto stable (non-volatile) storage media

I/O Subsystem

- One purpose of OS is to hide specialities of hardware devices from the user
- I/O subsystem responsible for
 - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
 - General device-driver interface
 - Drivers for specific hardware devices

Device-Status Table



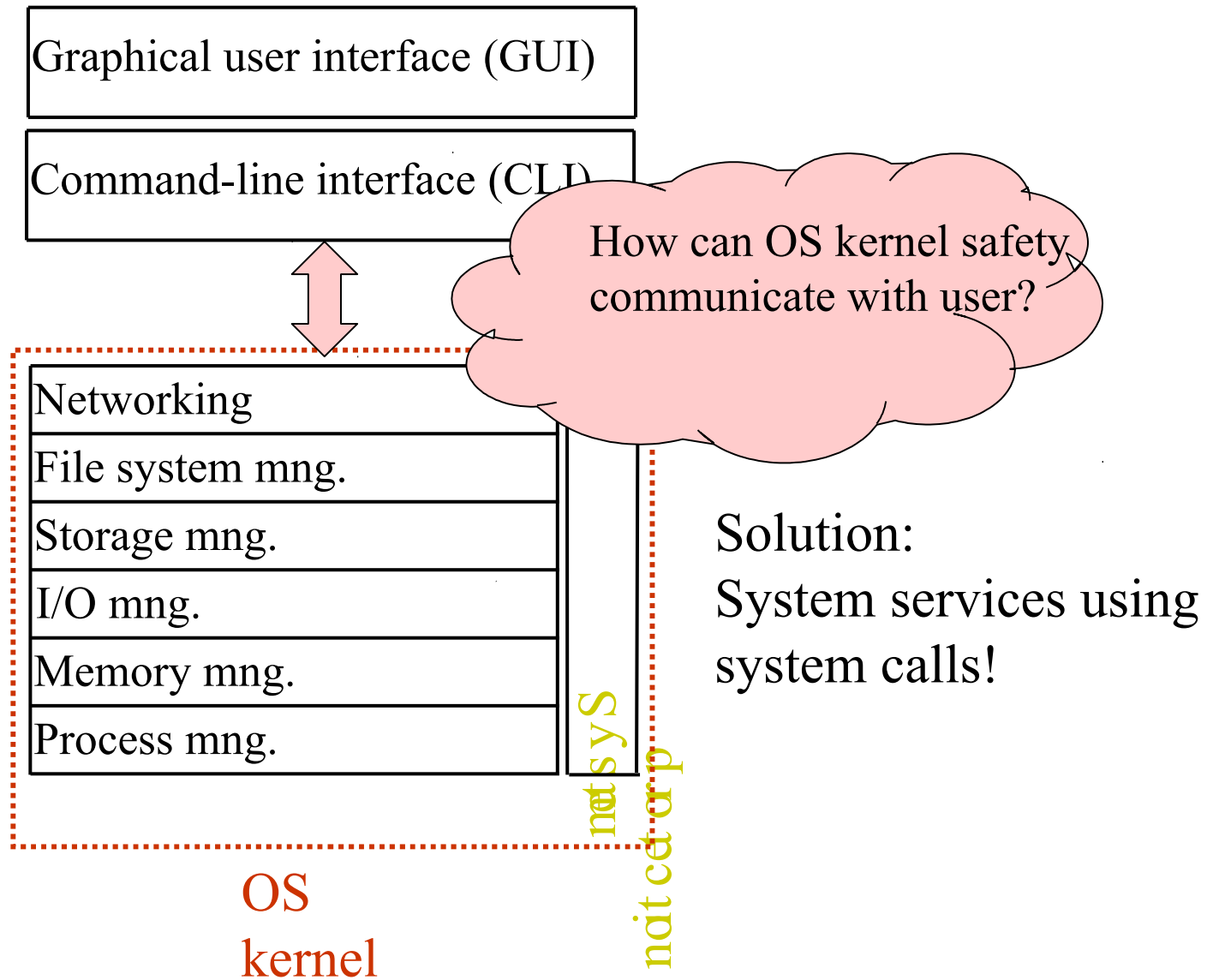
User Operating System Interface

- **CLI** allows direct command entry
 - Sometimes implemented in kernel, mostly by system programs
 - Sometimes multiple flavors implemented – **shells**
 - Primarily fetches a command from user and executes it
 - ▶ Some commands are built-in, sometimes just names of programs
 - ▶ If the latter, adding new features doesn't require shell modification

User Operating System Interface

- **GUI** – a 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 1981, followed by Apple 1983, X windows (client-server) 1984 and MS Windows 1.0 -1985
- Many systems include both CLI and GUI interfaces
 - Microsoft Windows is GUI with CLI “command” shell
 - Apple Mac OS X as “Aqua” GUI interface with UNIX kernel underneath and shells available
 - Solaris is CLI with optional GUI interfaces (Java Desktop, KDE)

How Operating System works?



Operating System Services

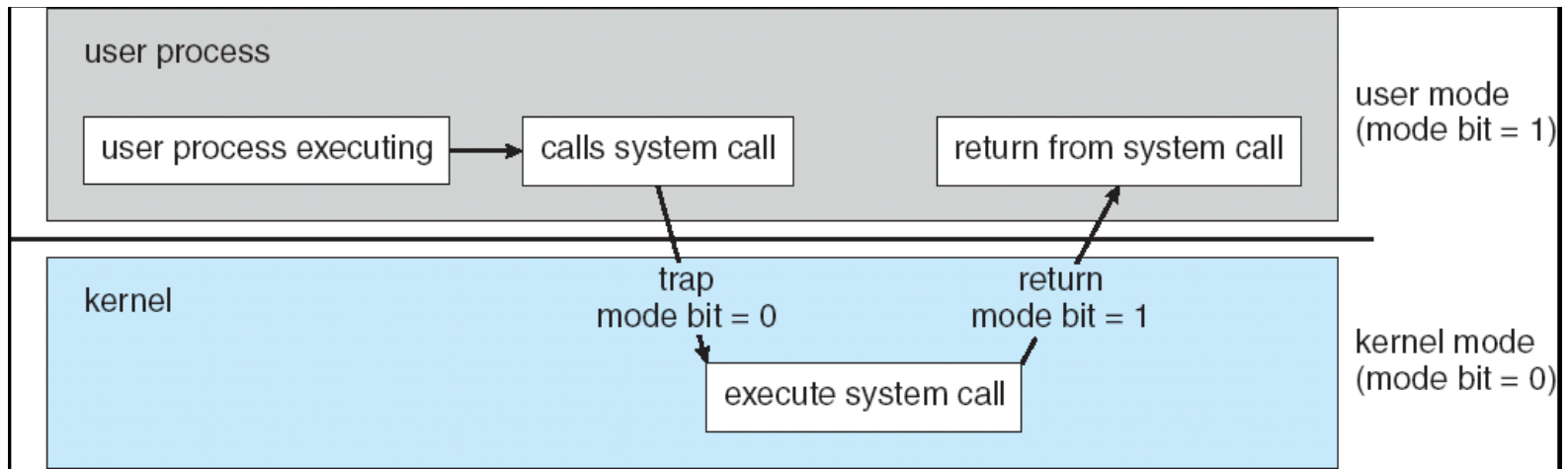
- Operating System Services are all function that OS kernel offers for user programs.
- There are several sets of OS Services
- One set of operating-system services provides functions that are **helpful to the user**:
 - User interface - Almost all operating systems have a user interface (UI)
 - ▶ Varies 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.
 - File-system manipulation - The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.

Operating System Services

- 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

OS protection by System services

Transition from User to Kernel Mode and back



- User processes are running in protected mode
- Kernel is running in supervisor mode

System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level **Application Program Interface (API)** rather than direct system call use
- Three most common APIs are
 - **Win32** API for Windows,
 - **POSIX** API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and
 - **Java API** for the Java virtual machine (JVM)
- Why to use APIs rather than system calls?

(Note that the system-call names used throughout this text are generic)