

Introduction

CSCI 315 Operating Systems Design
Department of Computer Science



What is an **Operating System**?

What is an Operating System?

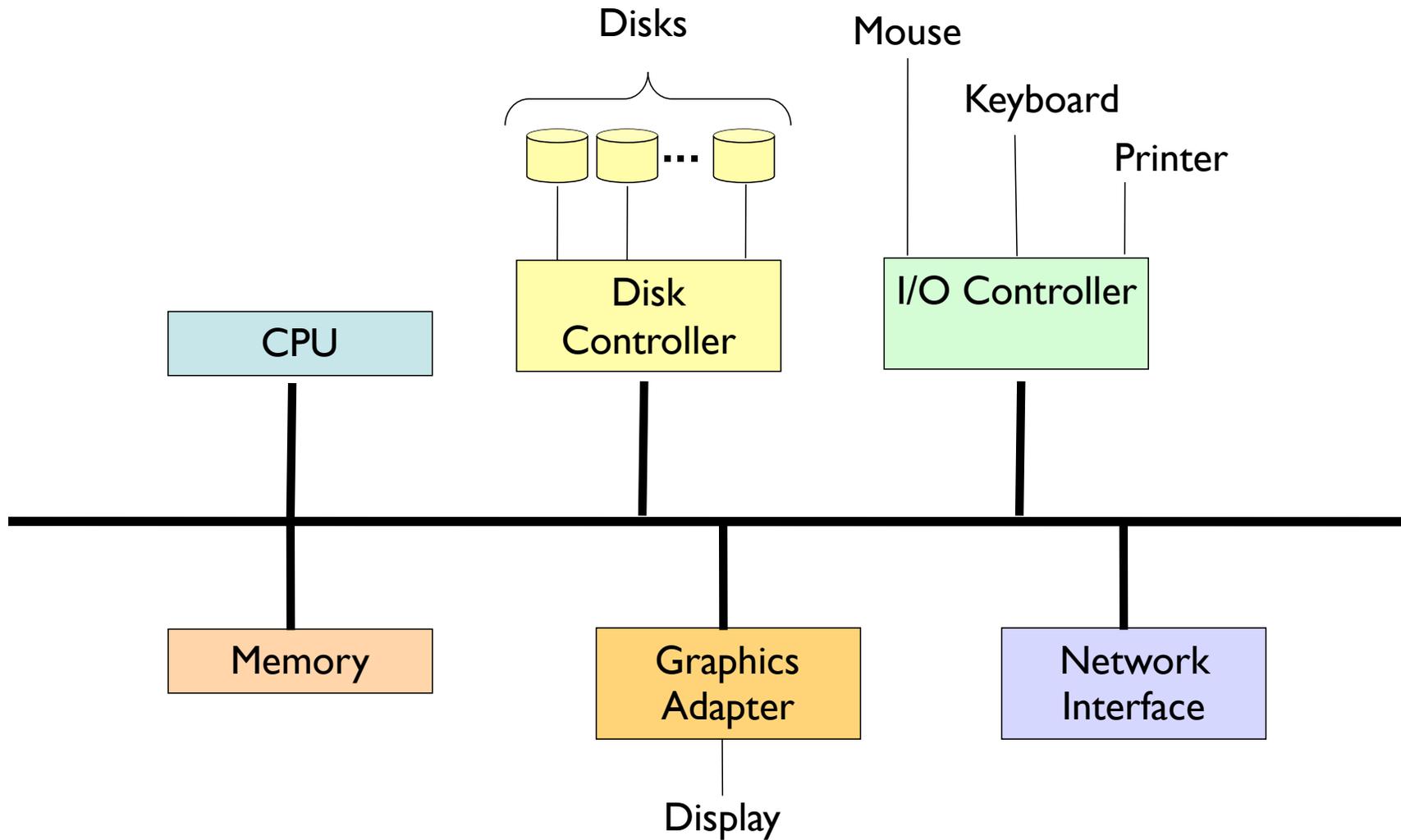
A "program" that acts as an intermediary between a user of a computer and the computer hardware.

- **The OS manages resources in the computer system.**
- **The OS controls the execution of programs.**

Operating System Definitions

- **Resource allocator** – manages and allocates resources.
- **Control program** – controls the execution of user programs and operations of I/O devices.
- **Kernel** – the one program “running” at all times (all else being application programs).

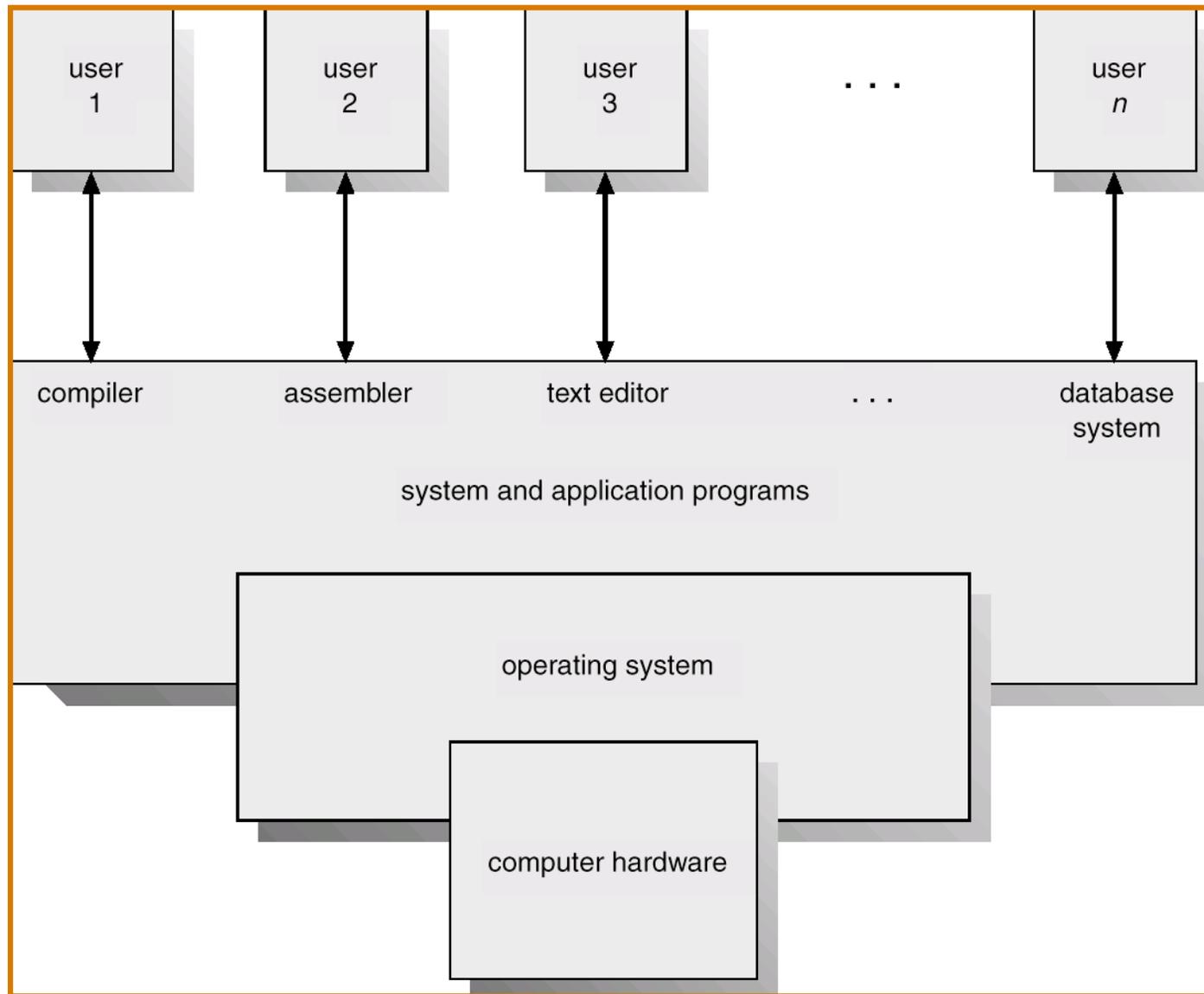
A Modern Computer System



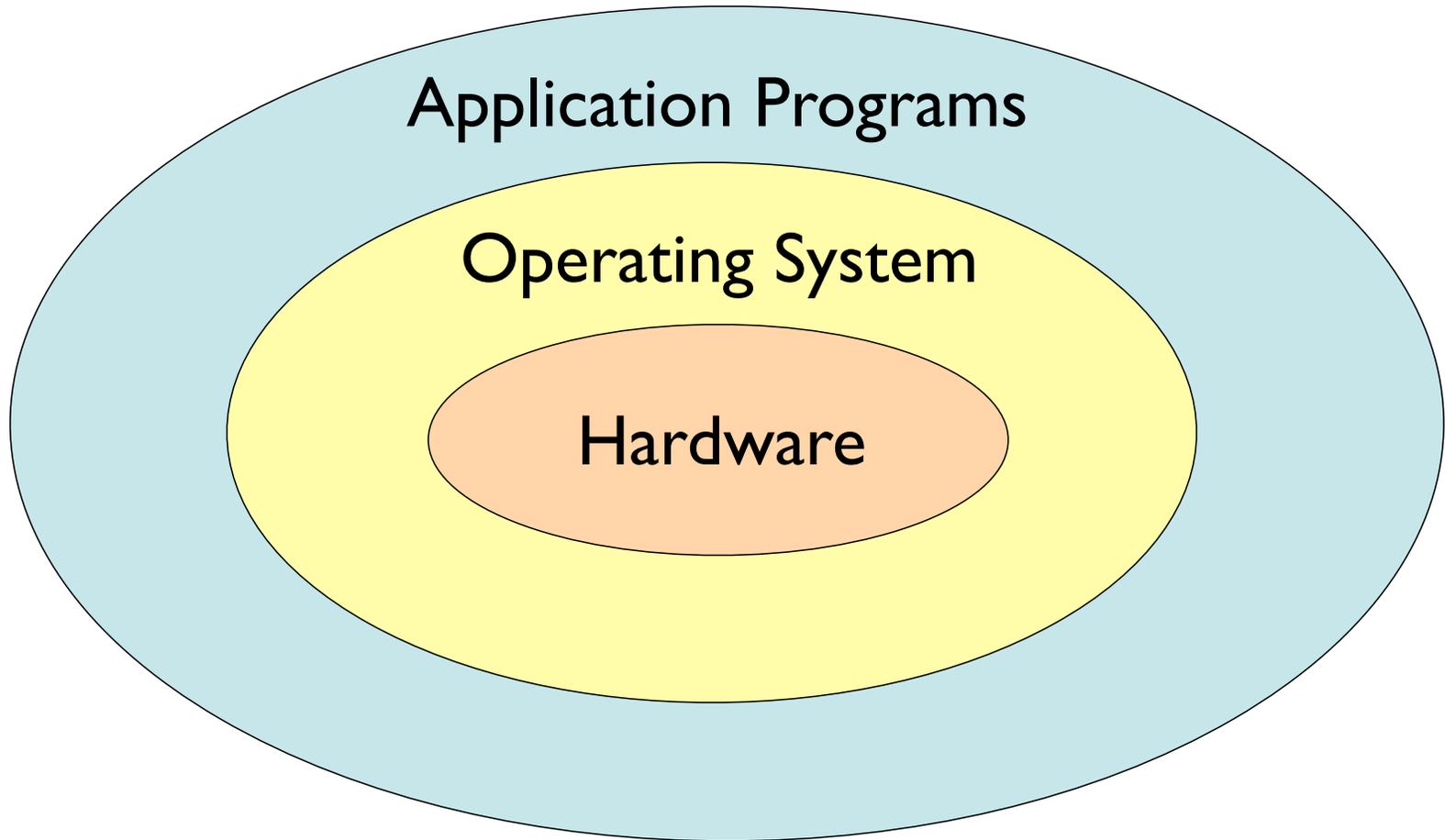
Computer System Components

1. **Hardware** – provides basic computing resources (CPU, memory, I/O devices).
2. **Operating system** – controls and coordinates the use of the hardware among the various application programs for the various users.
3. **Applications programs** – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).
4. **Users** (people, machines, other computers).

Abstract View of System Components

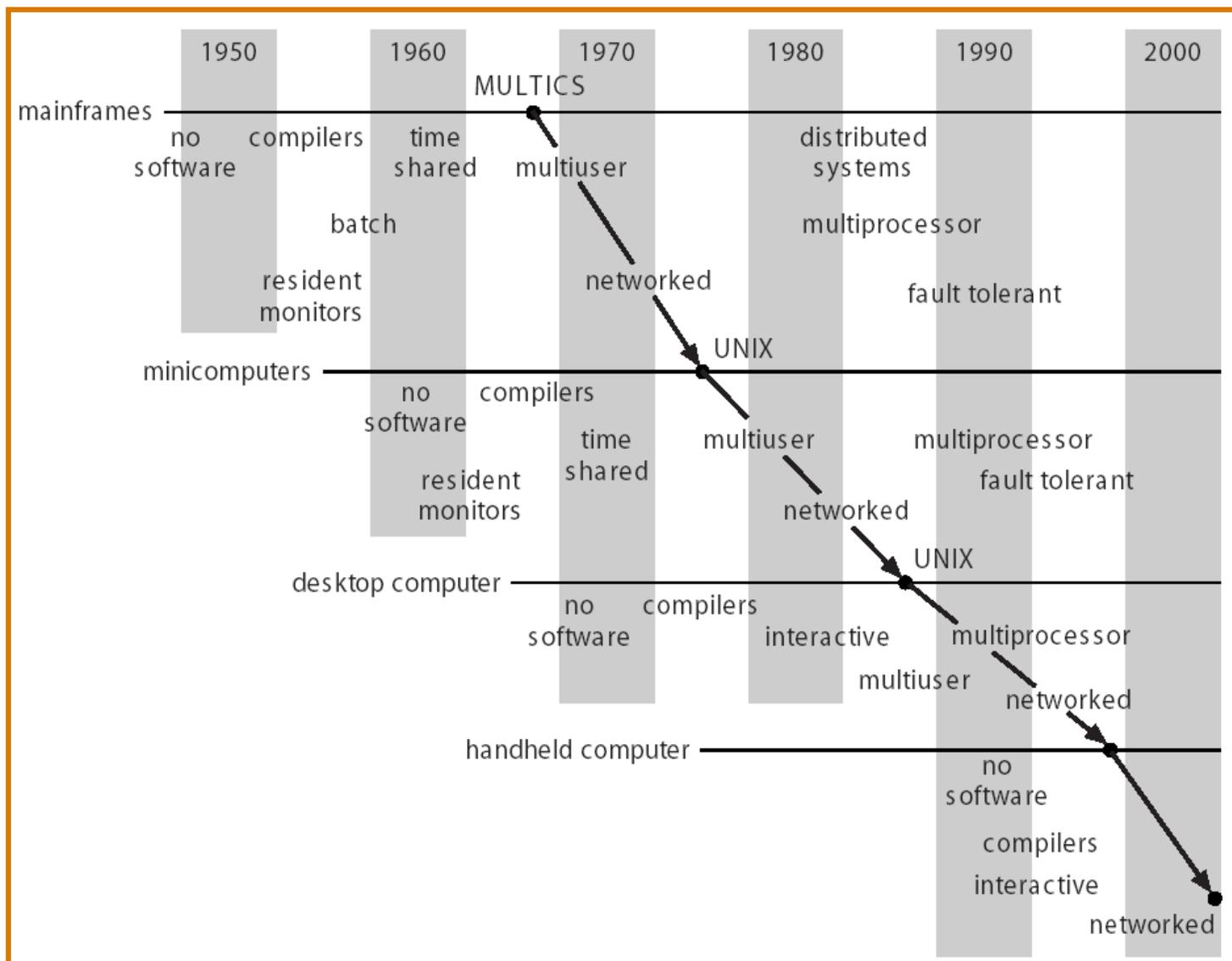


Macroscopic Abstract View of the Computer System

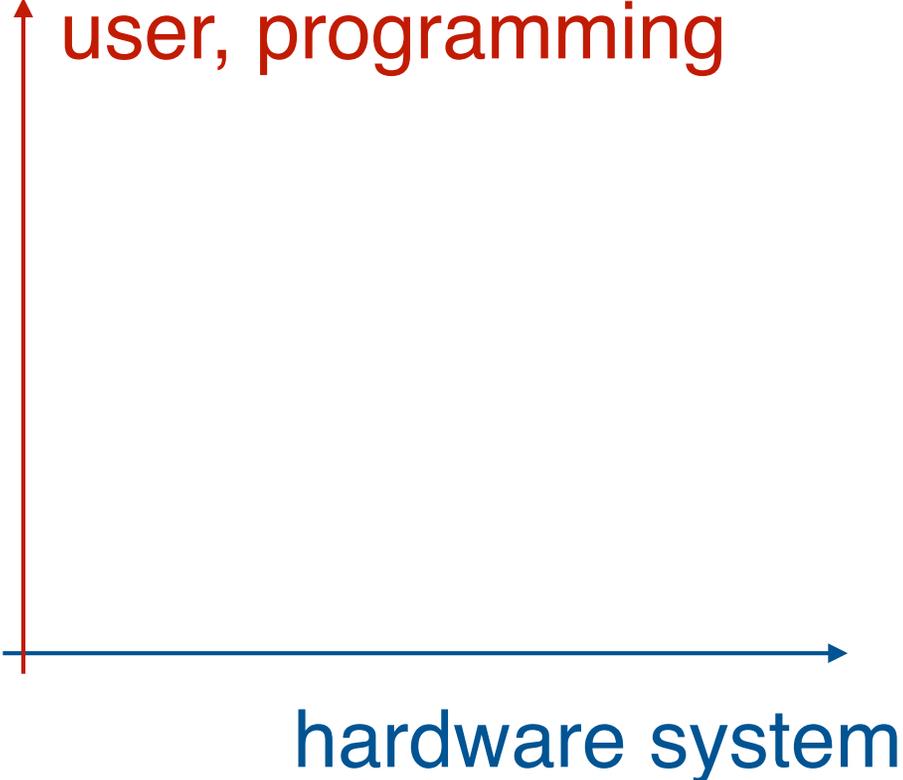


Why talk about **Design**?

Evolution



Not all systems are the same

- general purpose
 - mobile system
 - embedded system
 - real time
 - mainframe
 - server
 - cluster (HPC)
 - distributed
 - parallel
- single user, single program
 - single user, multiprogrammed
 - multiple user, multiprogrammed
- 
- A 2D coordinate system is shown. The vertical axis is a red arrow pointing upwards, labeled "user, programming" in red text. The horizontal axis is a blue arrow pointing to the right, labeled "hardware system" in blue text.

Consider the needs

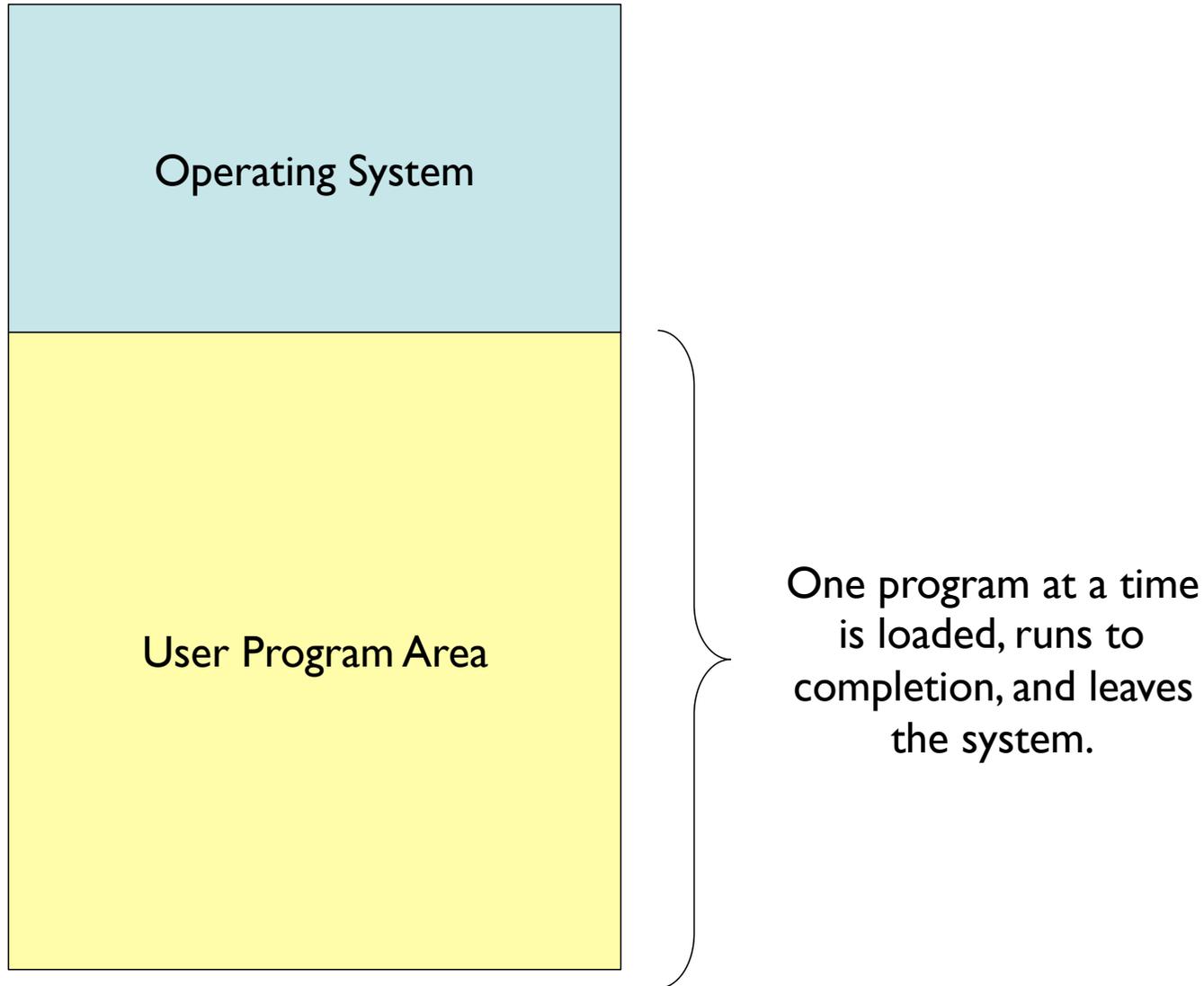
- What is the goal of the system?
- What is the hardware for this system?
- What resources are available?
- How many users?
- Do programs need to run “simultaneously” ?
- **What are the most essential functions that all users or programs will need in the system?**

The needs define what the OS
will do for the system

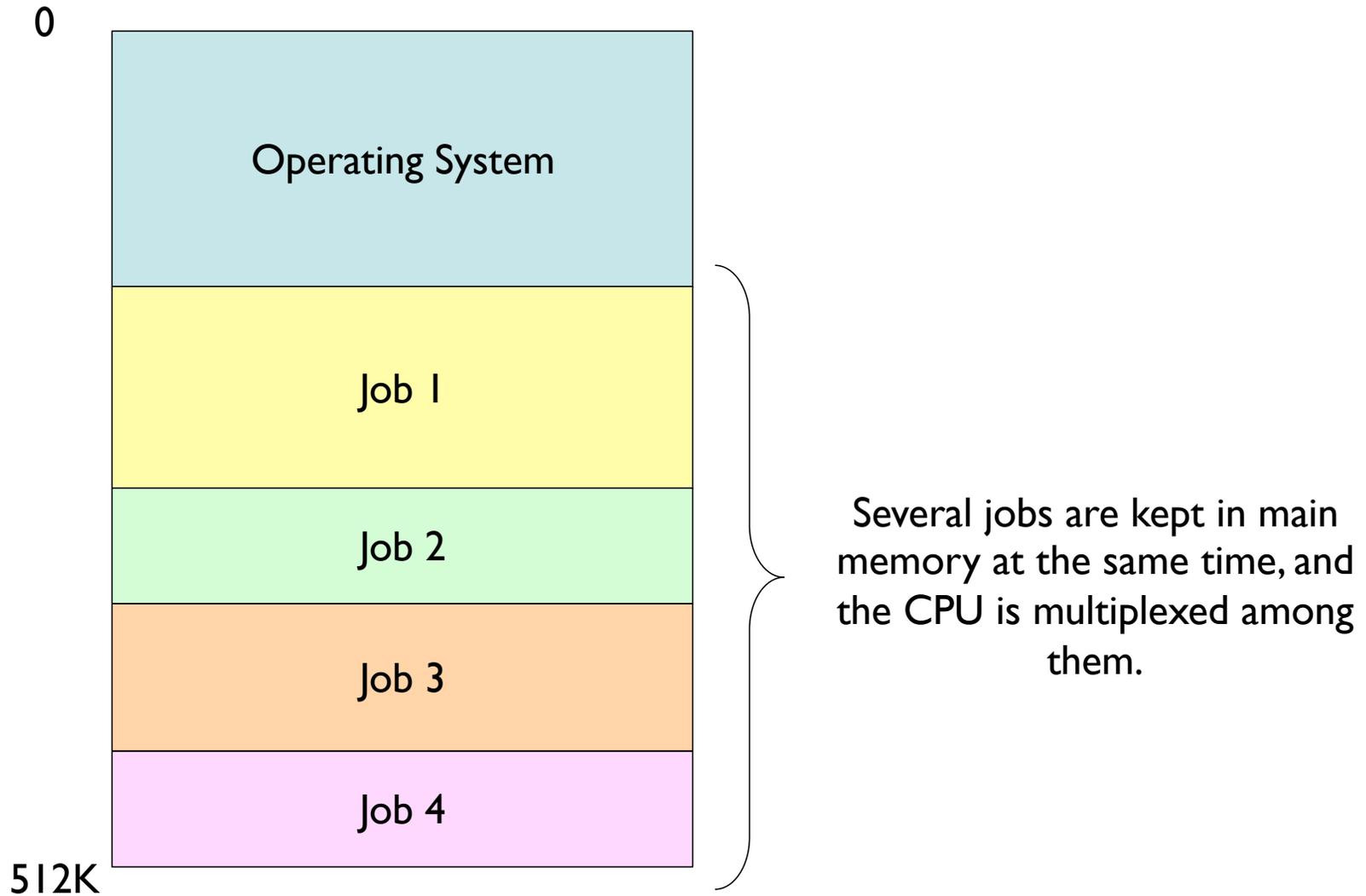
Mainframe Systems

- Reduce setup time by batching similar jobs.
- Automatic job sequencing – automatically transfers control from one job to another. First rudimentary operating system.
- Resident monitor:
 - initial control in monitor,
 - control transfers to job,
 - when job completes control transfers back to monitor.

Memory Layout for a Simple Batch System



Multiprogrammed Batch Systems



OS Features Needed for Multiprogramming

- I/O routines supplied by the system.
- Memory management – allocate memory to each of several jobs.
- CPU scheduling – determine which job runs when.
- Control access to multiple devices.

Time-Sharing Systems

Interactive Computing

- The CPU is multiplexed among several jobs that are kept in memory and on disk (the CPU is allocated to a job only if the job is in memory).
- A job swapped in and out of memory to the disk.
- On-line communication between the user and the system is provided:
 - When the operating system finishes the execution of one command, it seeks the next “control statement” from the user’s keyboard
- On-line system must be available for users to access data and code.

Desktop Systems

- *Personal computers* – computer system dedicated to a single user.
- I/O devices – keyboards, mice, display screens, small printers.
- User convenience and responsiveness.
- Can adopt technology developed for larger operating system:
 - Often individuals have sole use of computer and do not need advanced CPU utilization or protection features.
- May run several different types of operating systems (Windows, MacOS, UNIX, Linux).

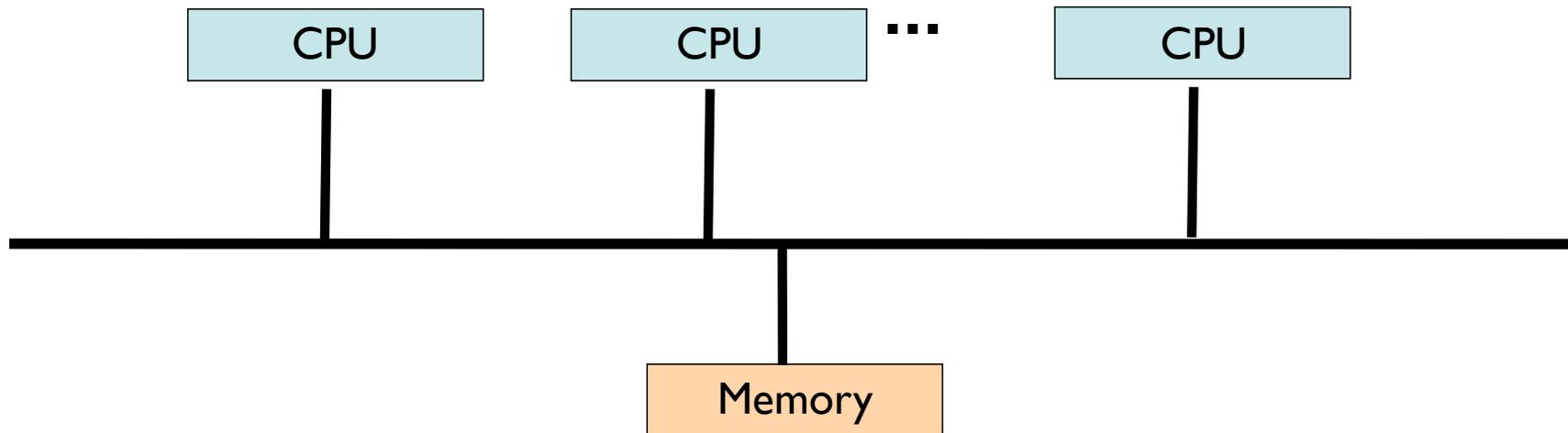
Parallel Systems

- Systems with more than one CPU in close communication (also known as *multiprocessor systems*).
- *Tightly coupled system* – processors share memory and a clock; communication usually takes place through the shared memory.
- Advantages of parallel system:
 - Increased *throughput*
 - Economical
 - Increased reliability (in some cases)
 - graceful degradation
 - fail-soft systems

Parallel Systems (Cont.)

- *Asymmetric multiprocessing*
 - Each processor is assigned a specific task; master processor schedules and allocated work to slave processors.
 - More common in extremely large systems.
- *Symmetric multiprocessing (SMP)*
 - Each processor runs an identical copy of the operating system.
 - Many processes can run at once without performance deterioration.
 - Most modern operating systems support SMP.

Symmetric Multiprocessing Architecture



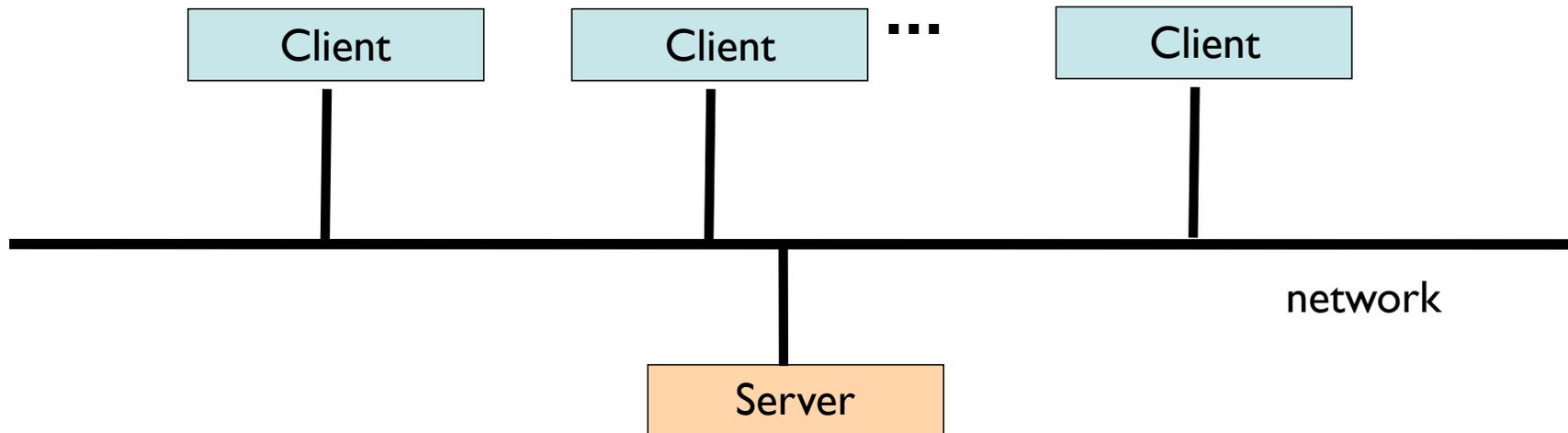
Distributed Systems

- Distribute the computation among several physical processors.
- *Loosely coupled system* – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- Advantages of distributed systems:
 - Resources Sharing,
 - Computation speed up – load sharing,
 - Reliability,
 - Communications.

Distributed Systems (cont.)

- Requires networking infrastructure.
- Local area networks (*LAN*) or Wide area networks (*WAN*).
- May be either *client-server* or *peer-to-peer* systems.

General Structure of Client-Server System



Clustered Systems

- Clustering allows two or more systems to share storage.
- Provides high reliability.
- *Asymmetric clustering*: one server runs the application or applications while other servers standby.
- *Symmetric clustering*: all N hosts are running the application or applications.

Real-Time Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints.
- Real-Time systems may be either *hard* or *soft* real-time.

Real-Time Systems (Cont.)

- Hard real-time:
 - Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM).
 - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- Soft real-time:
 - Limited utility in industrial control of robotics.
 - Can be integrated with time-shared systems.
 - Useful in applications (multimedia, virtual reality) requiring tight response times.

Embedded Systems

- Appliances.
- Smart sensors.
- Digital control systems.
- Issues:
 - Limited memory,
 - Slower processors,
 - Small display screens (if any).

Operating System Operations

Assumptions:

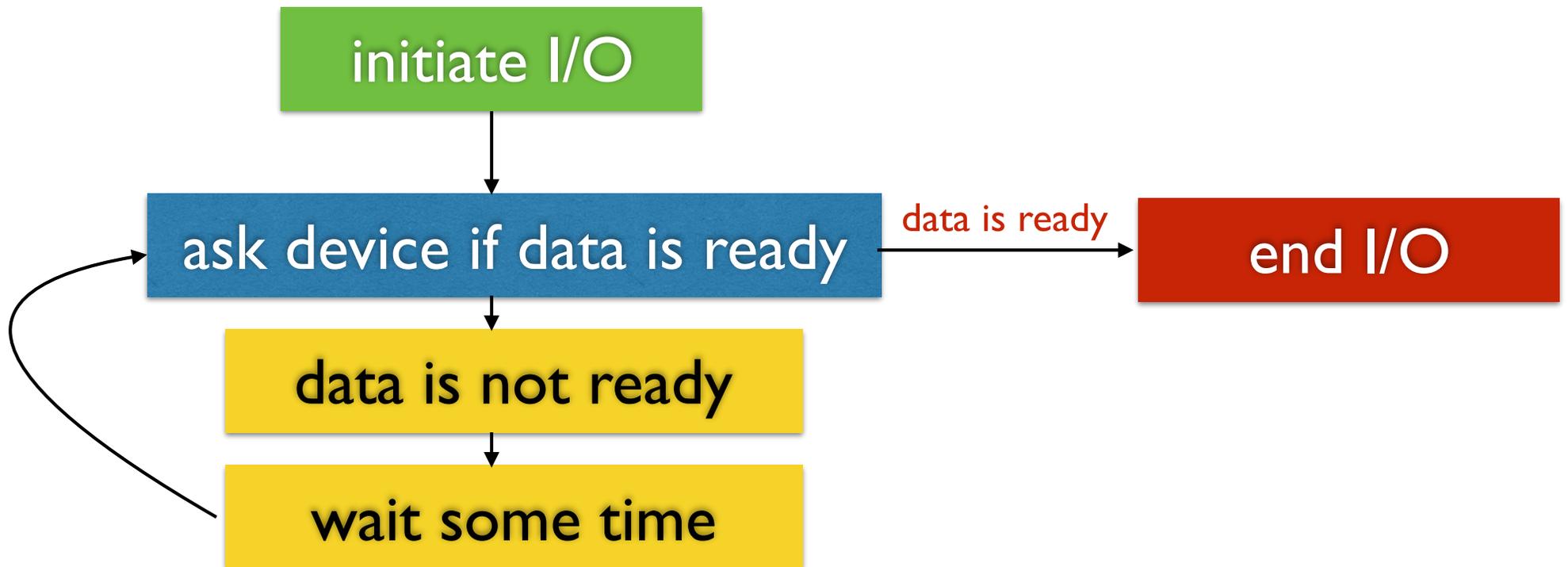
- I/O devices and the CPU can execute **concurrently**.
- Each *device controller* is in charge of a particular device type.
- Each device controller has a *local buffer*.
- There must be some mechanism to move data from/to main memory to/from local buffers.
- I/O operations move data from the device to a controller's local buffer.
- There must be some mechanism for the CPU to learn that an I/O operation has completed.

I/O and the OS

Nothing happens without I/O:

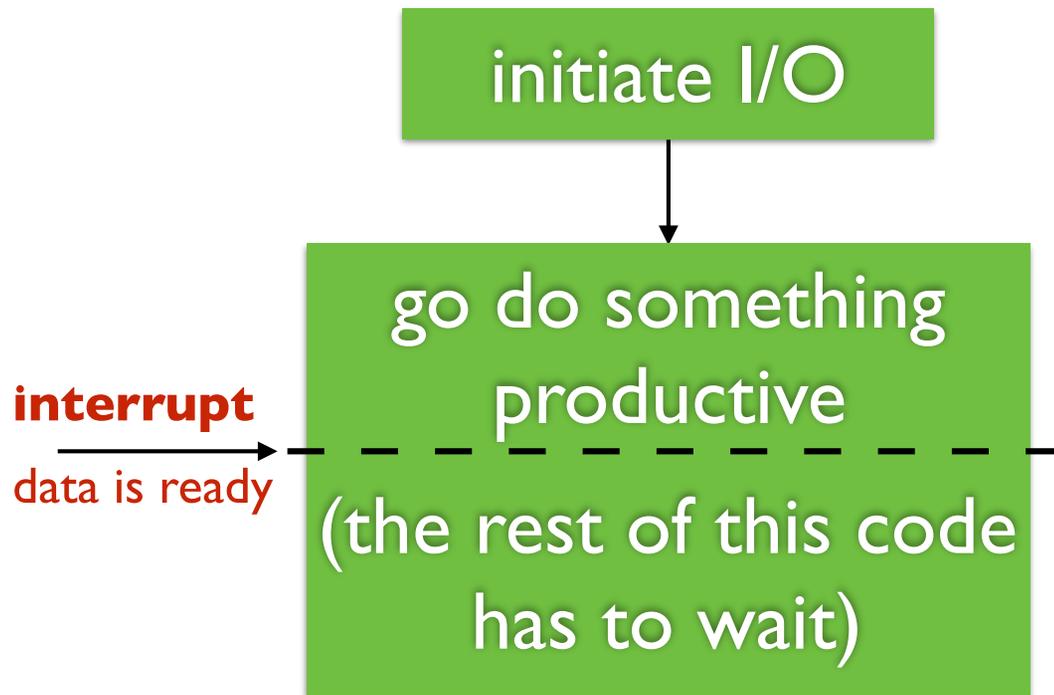
- Data comes in from input devices and goes out to output devices.
- Two big questions: (1) “Should application programmers work directly with I/O devices?” (2) “Is there any advantage in having the OS serve as intermediary to between applications programs and I/O devices?”
- Another BIG question: “how does the CPU interact with I/O devices to make data transfers happen?”
 - Ask yourself if and when it’s productive to have the CPU involved in data transfers.
 - Ask yourself how the CPU may be told when a data transfer between I/O device and memory terminates.

Option 1: Polling

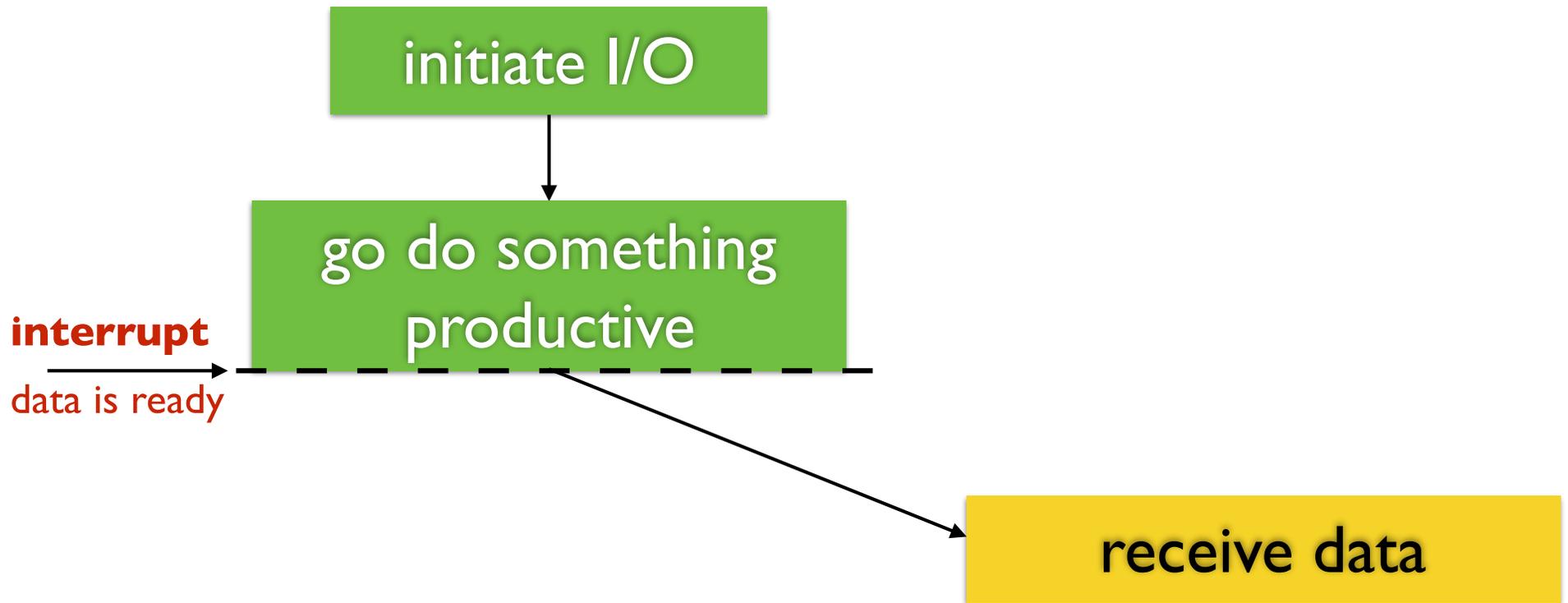


The Simpsons: <https://www.youtube.com/watch?v=18AzodTPG5U>

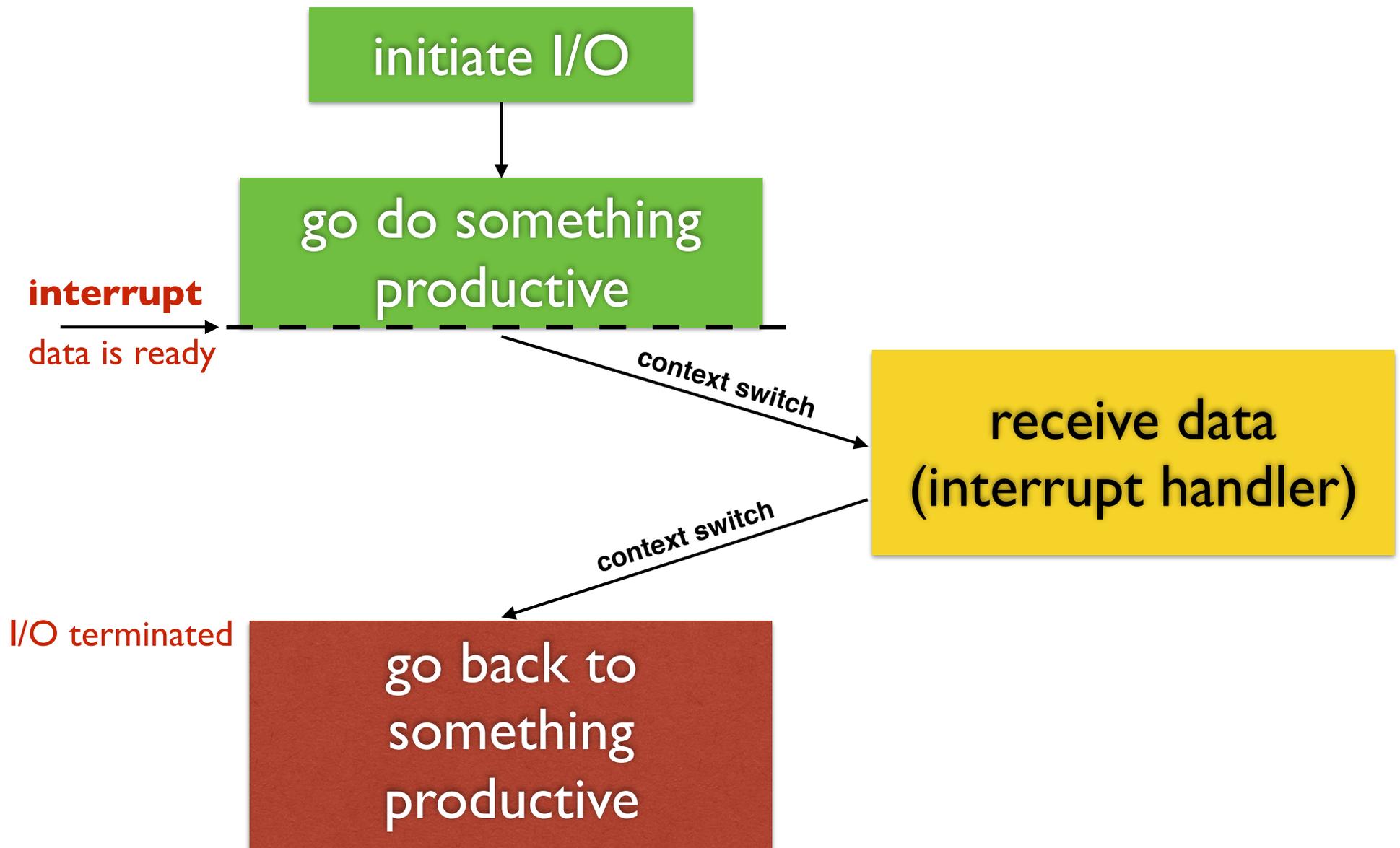
Option 2: Interrupt



Option 2: Interrupt



Option 2: Interrupt

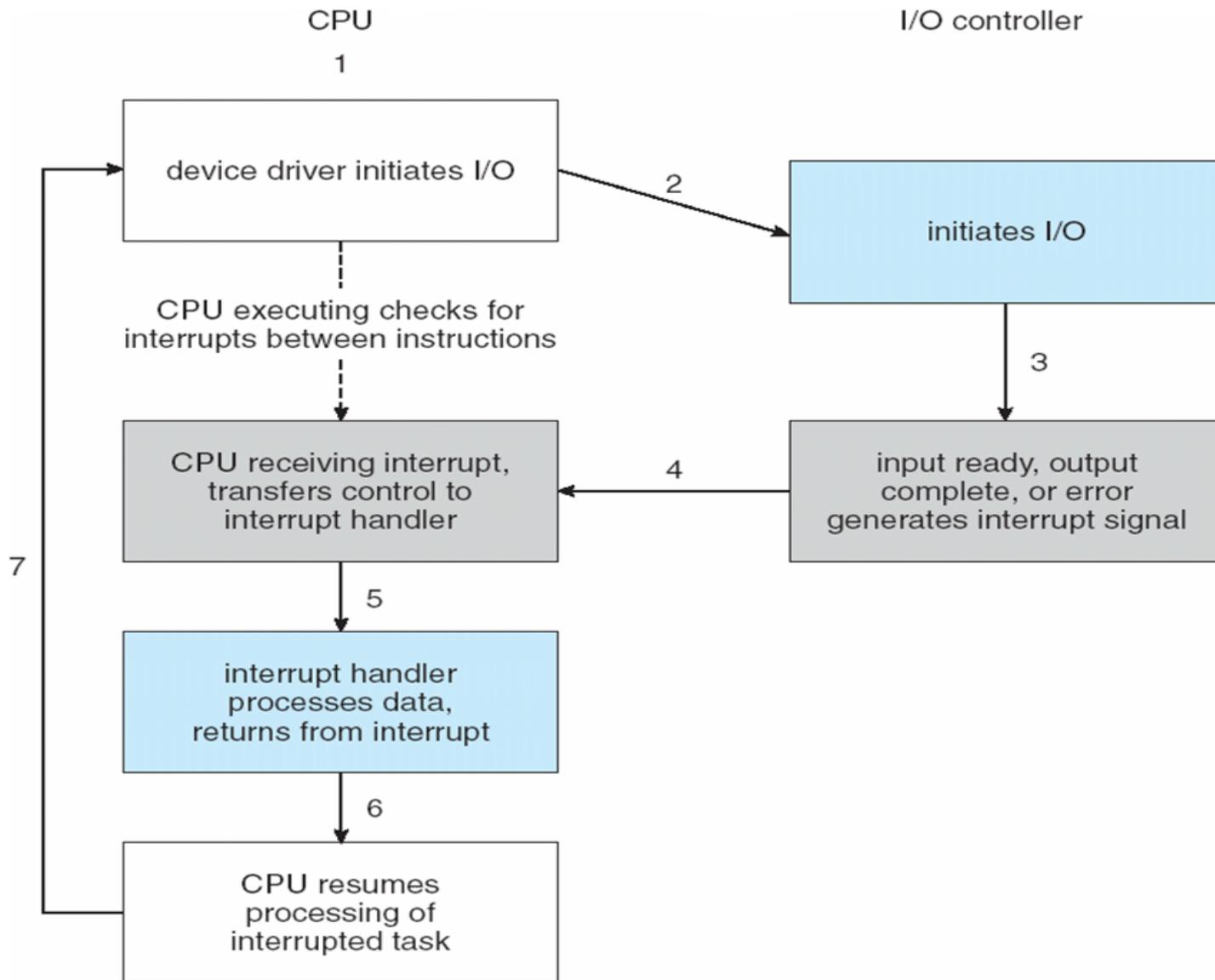


Interrupt Vector

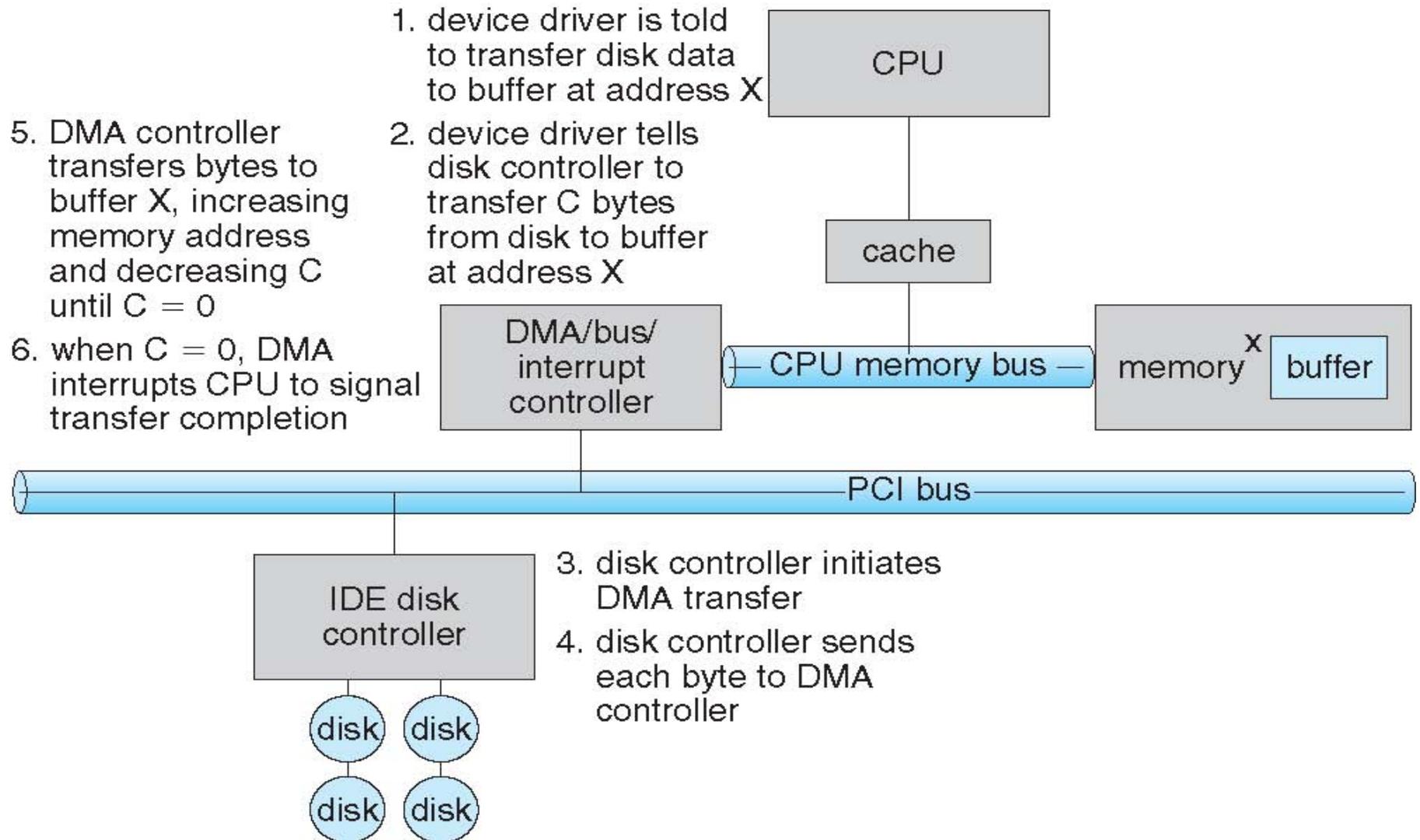
Each device is assigned an interrupt number or ID

| Interrupt ID | Address of the corresponding handler in memory (think of a C function pointer) |
|--------------|--|
| 0 | address of handler to int 0 |
| 1 | address of handler to int 1 |
| 2 | address of handler to int 2 |
| ... | address of handler to int ... |

Interrupt Driven I/O Cycle

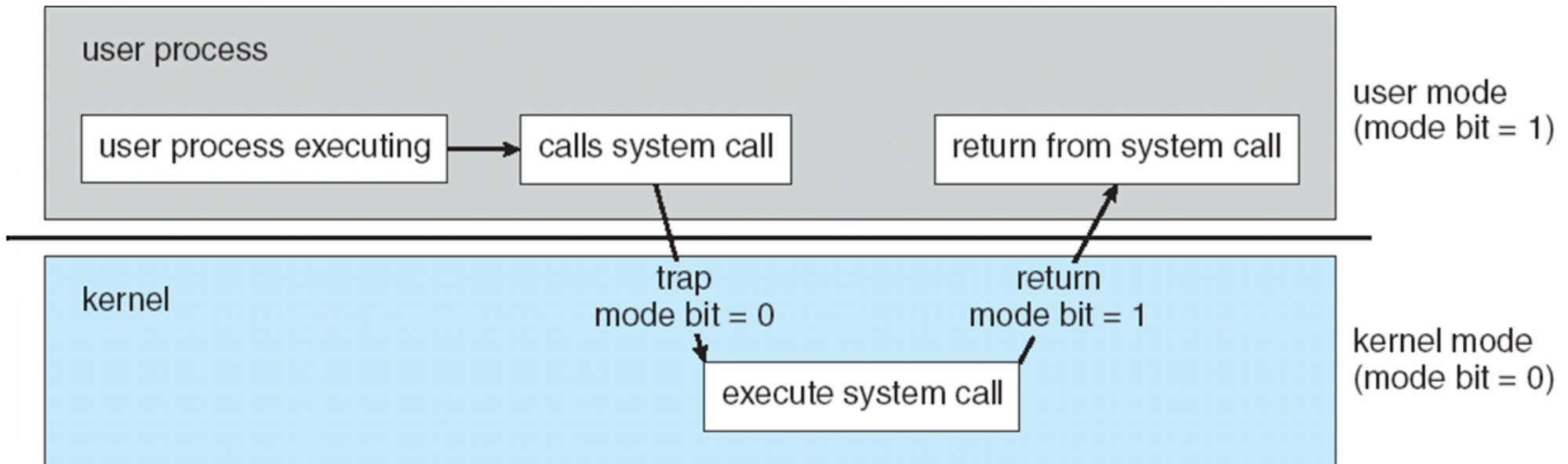


Data Transfer from I/O to RAM

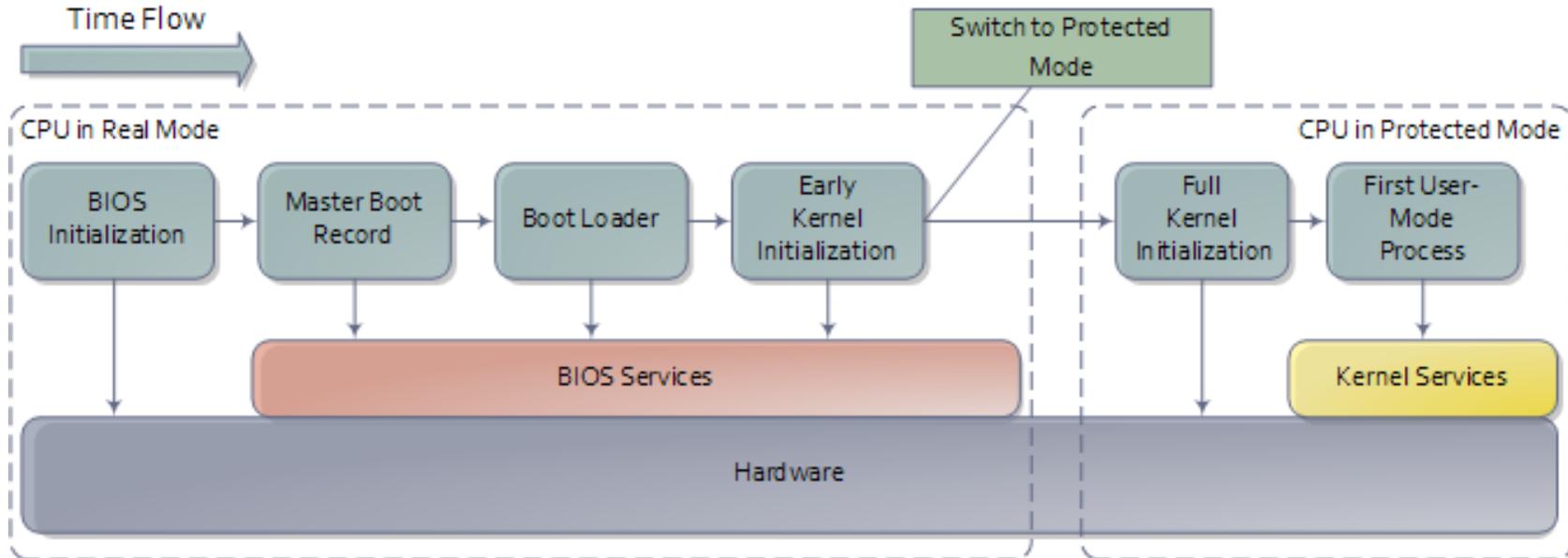


Hardware Support for the OS

- **Two classes of instructions:** one class for anyone to use, others with **privileged** use (for the OS kernel).
- Need to be able to switch between **user mode** and **kernel mode**.
- If a user runs a privileged instruction, an exception is raised.
- To switch to kernel mode, you need to trap to the kernel.



Booting up the OS



- BIOS is firmware (flash memory). Power on self tests (POST) check if machine is in shape to run.
- Every disk has an MBR, which contains a bootstrap program and a **partition table**. Each partition has a **boot sector** with the boot loader.
- How does the machine know the address of the first instruction to run???