COP 4600
Introduction To Operating Systems
Summer 2011

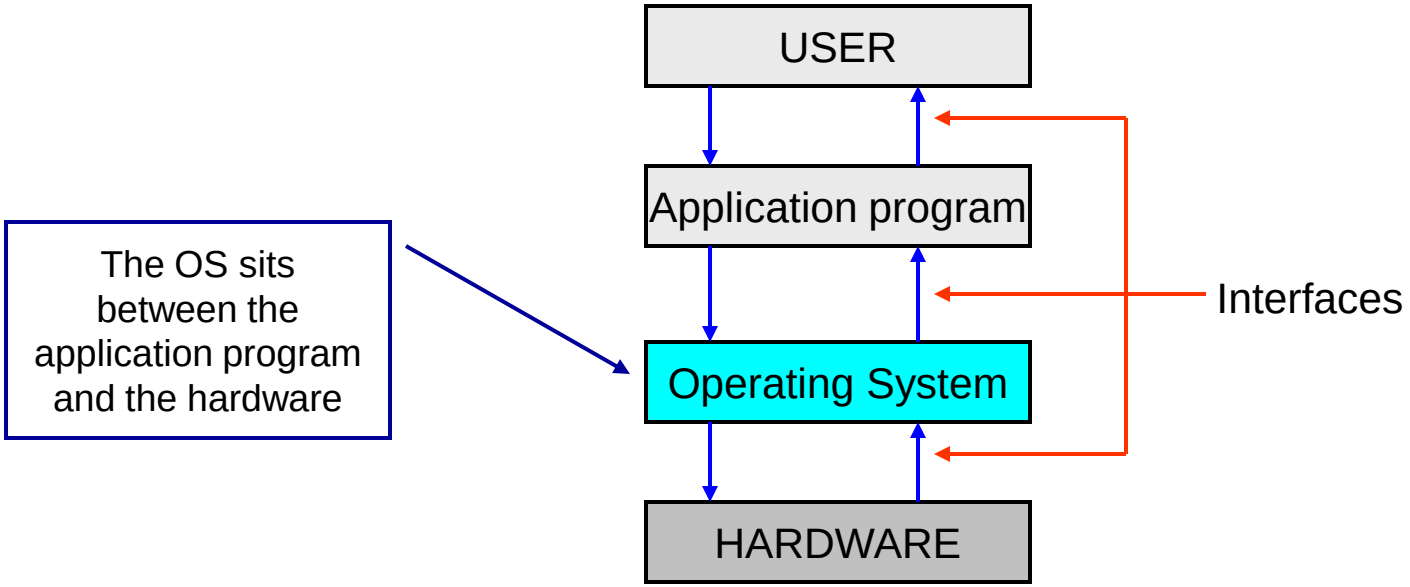
Introductory Material

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cop4600/sum2011

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida

What is an Operating System?

- In the most general sense an operating system is a collection of system software routines that sit between an application program and the computer hardware on which that application is to be executed.



What is an Operating System? (cont.)

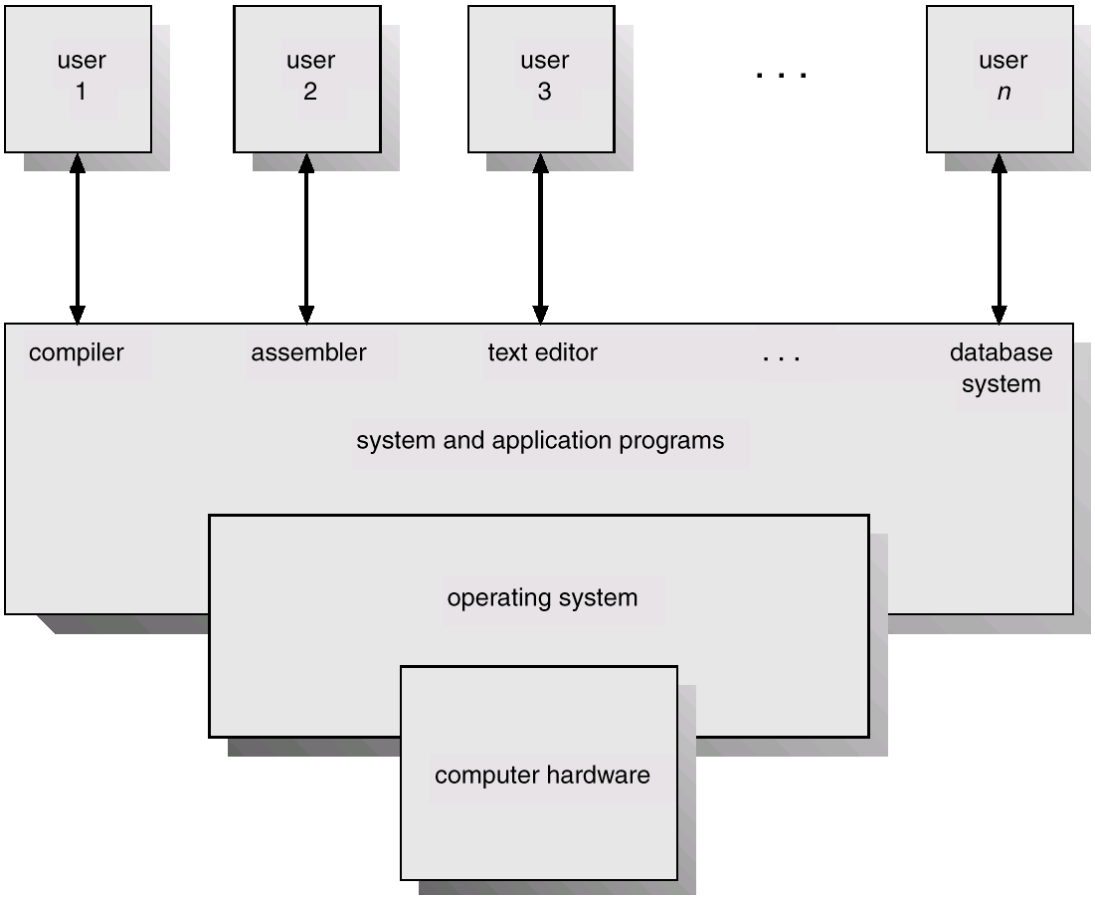
- For now, we can think of an OS as:

  1) is the interface or intermediary between a user/application and the computer hardware
  2) provides an environment in which the user can execute programs conveniently and
     - application and/or system software
  3) manages the computer's resources efficiently
     - memory, disk space, CPU time, I/O, software, etc.

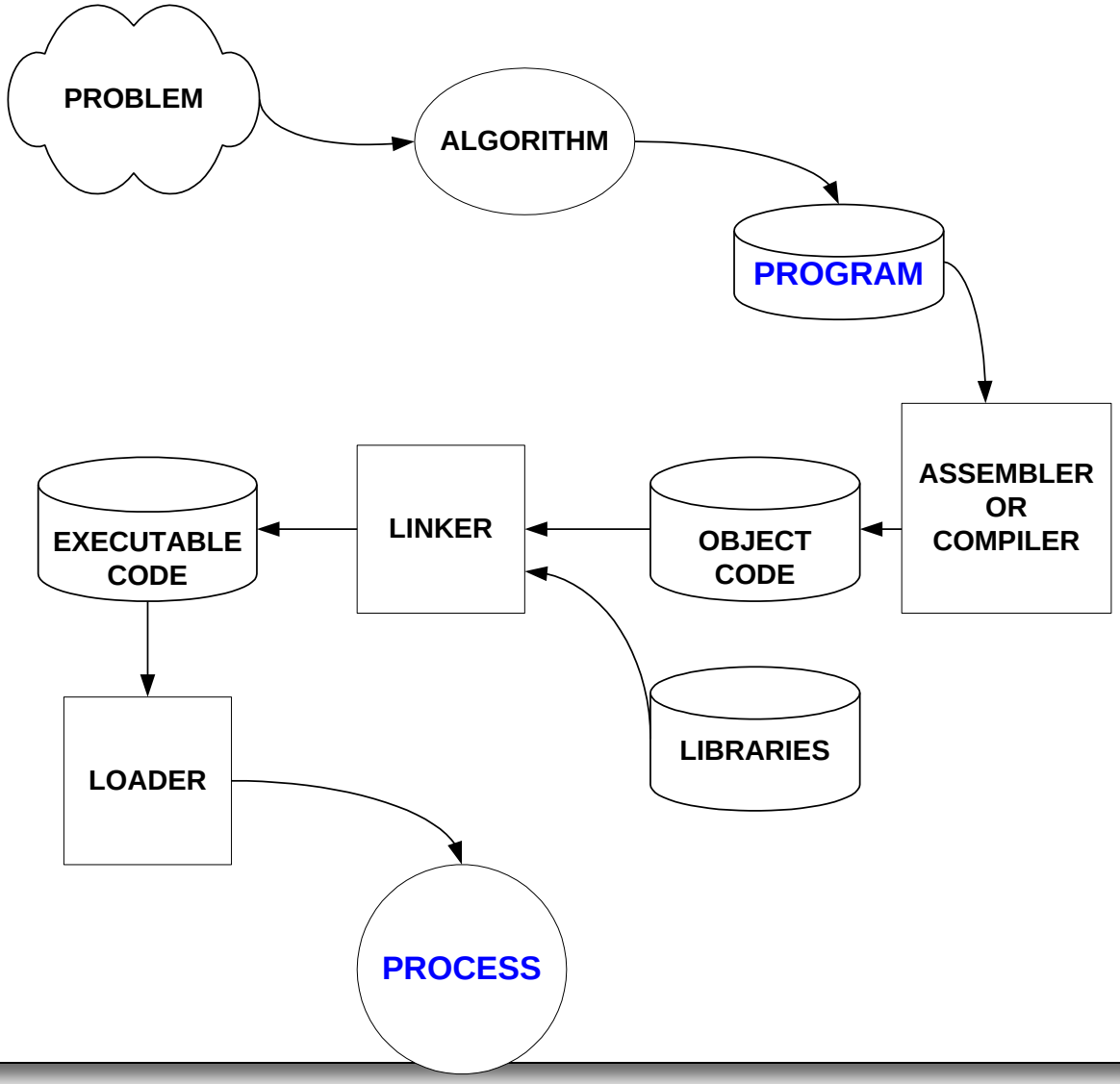
- Often an OS is a tradeoff between convenience and efficiency
  - Windows (GUI) vs. Unix (command interpreter)

The OS As An Intermediary

- What's an application?
  - Software to accomplish a task
    - Spread sheet, word processor, browser, email
  - What about system software?
    - Depending on who you ask, can be considered application programs, a computer resource, or part of the OS



What Is A Process?



What Is A Process (cont.)

- A *process*:
  - is a program in execution.
  - has a process control block (PCB)
  - has a program counter (PC)

- A process can have one or more *threads*.
  - A thread is sometimes known as a *lightweight process*

Types of Operating Systems

- Focus on two system resources
  - CPU (processor) Utilization
  - Main Memory Utilization
- Utilization is measure of busy time over total study time (T_{busy}/T_{Total})
- In the old days computers were
  - physically very large
  - but very small in terms of resources and capabilities
  - also very, very expensive
- Important to achieve high utilization of resources

Early Systems

- Instructions and data written in binary
- Loaded using switches on front panel
- Computers also had a few buttons
  - Halt, Run, Load, Set PC (displayed contents), Increment PC
- Everything done by programmer there was no real "user" as we know them today
- Very slow set up time
- Very limited output (set PC and check lights)
- CPU sat idle much of the time.
- Very little wasted memory since RAM was so small.
  - Programmers always "squeezed" program into Main Memory

Early Systems - Hardware Innovations

- Needed way to speed up Input & Output (I/O)
- Paper Tape
  - Data entry difficult, splicing needed or recopy tape
  - Paper tape output faster than looking at lights but hard to read.
- Punch Cards
  - Faster form of input.
  - Offline Card-to-Printer improved readability of output
- Magnetic Tape
  - Input even faster
  - Card-to-tape, tape-to-CPU, CPU-to-tape, tape-to-printer.
- Disk drives as a replacement for tapes

Early Systems - Software Innovations

- Assemblers
  - Symbolic programming rather than 1s and 0s
  - 1:1 relationship between assembler statements and machine instructions
- Linkers & Loaders
  - allowed the use of code libraries
  - didn't have to rewrite common code
- Compilers
  - Programming in High Level Languages (Fortran, COBOL)
  - 1:n relationship between a program statement and machine instructions
  - Eased programming task and improved operational efficiency.

Early Systems - People/Procedural Changes

- Too much work for programmer
- Division of labor became necessary.
  - Divided tasks between a programmer and professional operator
  - Operator could now organize the work more effectively and "batch" jobs
- Batching
  - allows similar jobs to run sequentially
  - efficient use of system software
  - today, refers more to jobs which lack user interaction
    - Example, billing systems used by companies
  - still took long time to set up jobs
  - CPU frequently sat idle

Running Multiple Programs

- Serial/Sequential Execution
  - One job must finish before next job starts
  - Results in very low utilization of resources
- Concurrent Execution
  - Two or more processes executing at the same time but doing different activities
  - Processes take turns using single shared resource
  - Gives the illusion of parallel processing
- Parallel/Simultaneous Execution
  - Two or more processes performing the same activity at the same time
  - Requires two or more of the same resource (e.g., processors, printers, disk drives)

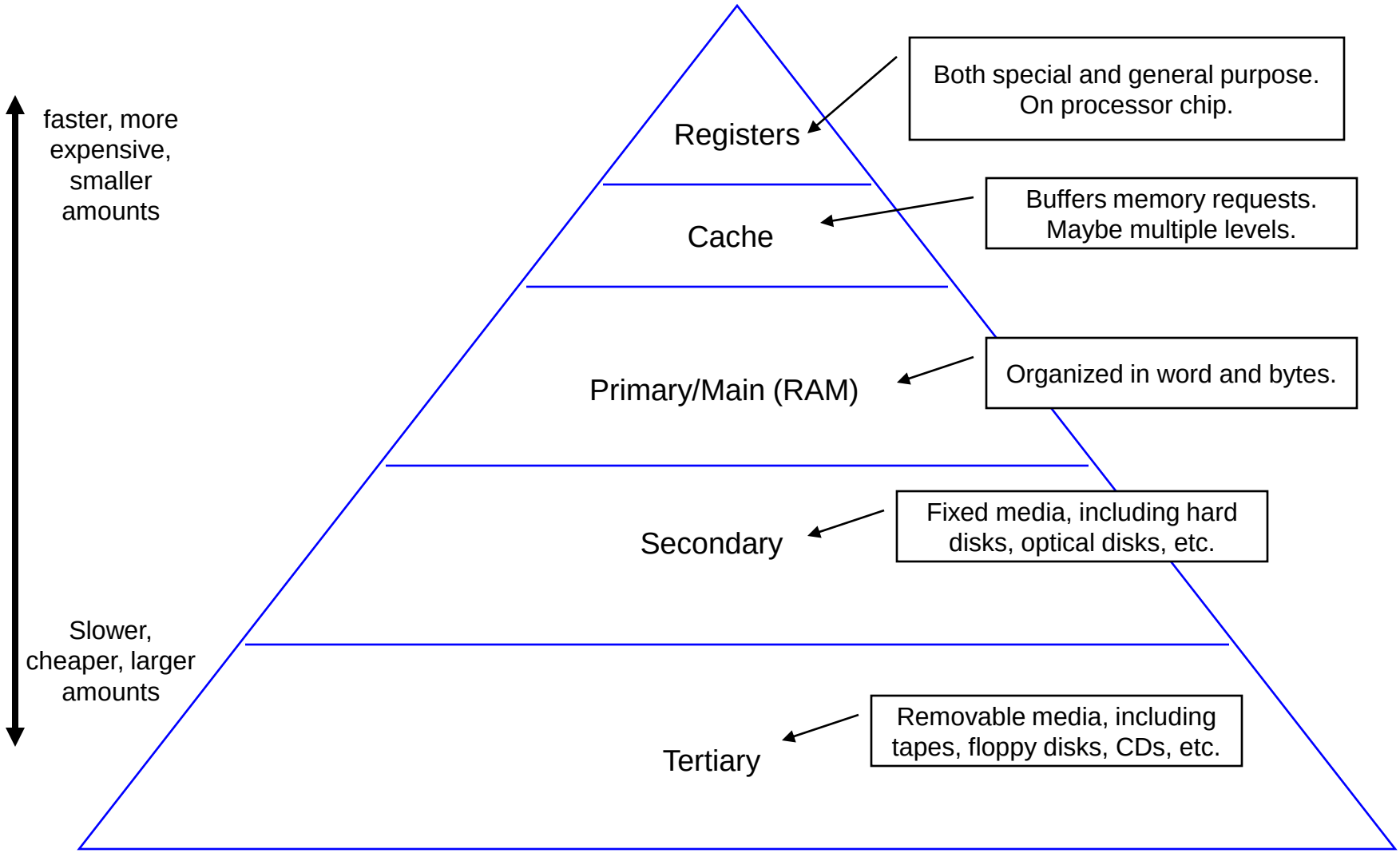
Resident Monitor

- Precursor to the Operating System
  - The beginnings of computer "self-governance"
- Resident in memory all the time, Monitored operations
- Primary task was job sequencing
  - In beginning read jobs sequentially from tape already prepared off-line
  - With disks, could select which jobs to run and when
    - Job Scheduling
- Resident Monitors Improved:
  - CPU Utilization: Faster setup, less idle time.
  - Memory Utilization: Sharing of I/O drivers/code
  - Functionality: Accounting and run time & I/O limits.

The Beginnings of Multiprogramming

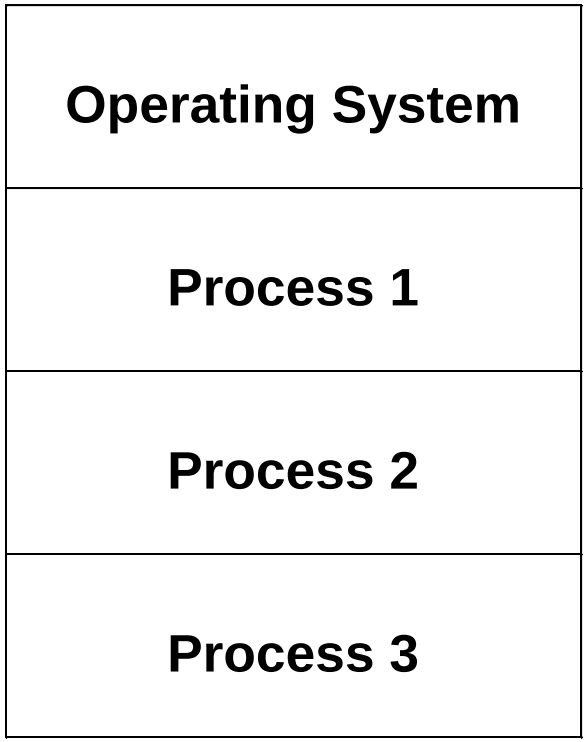
- With RM, had two programs running serially:
  - Application - RM - Application - RM - Application, etc.
  - Application ran to completion before RM took over
- Reduced CPU idle time between jobs but not between I/O operations
  - I/O speed very slow compared to CPU speed
  - Much of I/O deals with accessing data (tape, disk)

An Aside On The Memory Hierarchy



Multiprogramming (cont.)

- Need some way to perform CPU and I/O operations at the same time
- Keep more than one user program in memory
- Switch between programs during I/O operations



Multiprogramming (cont.)

- More efficient use of system, less idle time.
- Made possible because of reduced memory costs.
- Created a whole new set of problems:
  - Memory Management
  - Protection Mechanisms
  - Job Scheduling
  - CPU/Process Scheduling
- Improves resource utilization but not user interaction
  - ***"...manages the computer's resources efficiently..."***
- What about conveniences
  - ***"...the user can execute programs conveniently..."***

Time Sharing

- Introduced to improve the interaction with computer
  - Use of terminals and teleprinters
  - Allows user to input data and interact with program
- Give each process a slice of CPU time
  - Don't wait for next I/O operation
  - Similar to multiplexing
- Possible because user input is so slow
  - 1 char takes 1000 milliseconds to enter while only 2 milliseconds required for an interrupt handler
- Required the development of virtual memory, online file systems and directory structures
- Today's systems generally support a combination of batch and time sharing.

Other Operating System Developments

- Real Time - Response time is critical
  - Hard Real Time
    - Industrial & Robotic controls.
    - Very small bounded delays.
    - Little use of swapping or secondary storage.
  - Soft Real Time
    - Immediate response not as critical.
    - Monitoring devices, etc.
    - Longer delays tolerated
    - Use of secondary storage & priority scheduling.

Other Developments (cont.)

- Personal Computers
  - Many of the OS features we'll study are incorporated in today's PCs.
  - Change in priority since hardware is cheap and only a single user.
  - Greater emphasis on convenience than on efficiency
  - Ultimate in interactive – in a way, we've gone back to the early days
  - Adds new demands also – networking and multimedia.
- Handheld Systems

Other Developments (cont.)

- Parallel vs. Distributed Systems vs. Clustered
  - Needed for various reasons: larger problems, larger data requirements,
  - Parallel/Tightly coupled –share resources (e.g., memory, clock, bus, OS, disks)
    - Asymmetric multiprocessing – master/slave
    - Symmetric multiprocessing – each processor has copy of OS
    - Typically used for scientific applications and simulations.
  - Distributed/Loosely coupled – Separate computers linked via network.
    - Used for client/server applications
    - Peer-to-Peer applications
  - Clustered Systems
    - Computers linked by network but sharing storage

Performance Measurements

- Utilization (maximize)
  - $U = T_{busy} / T_{total}$ where T_{total} is total study time, or
  - $U = T_{used} / T_{available}$

- Throughput (maximize)
  - X = C / T where C is number of completed jobs/processes and T is time frame
  - The rate at which requests are processed

Performance Measurements (cont.)

- Turnaround Time (minimize)
  - Typically used in reference to batch systems
  - The time it takes to complete/execute a job/process

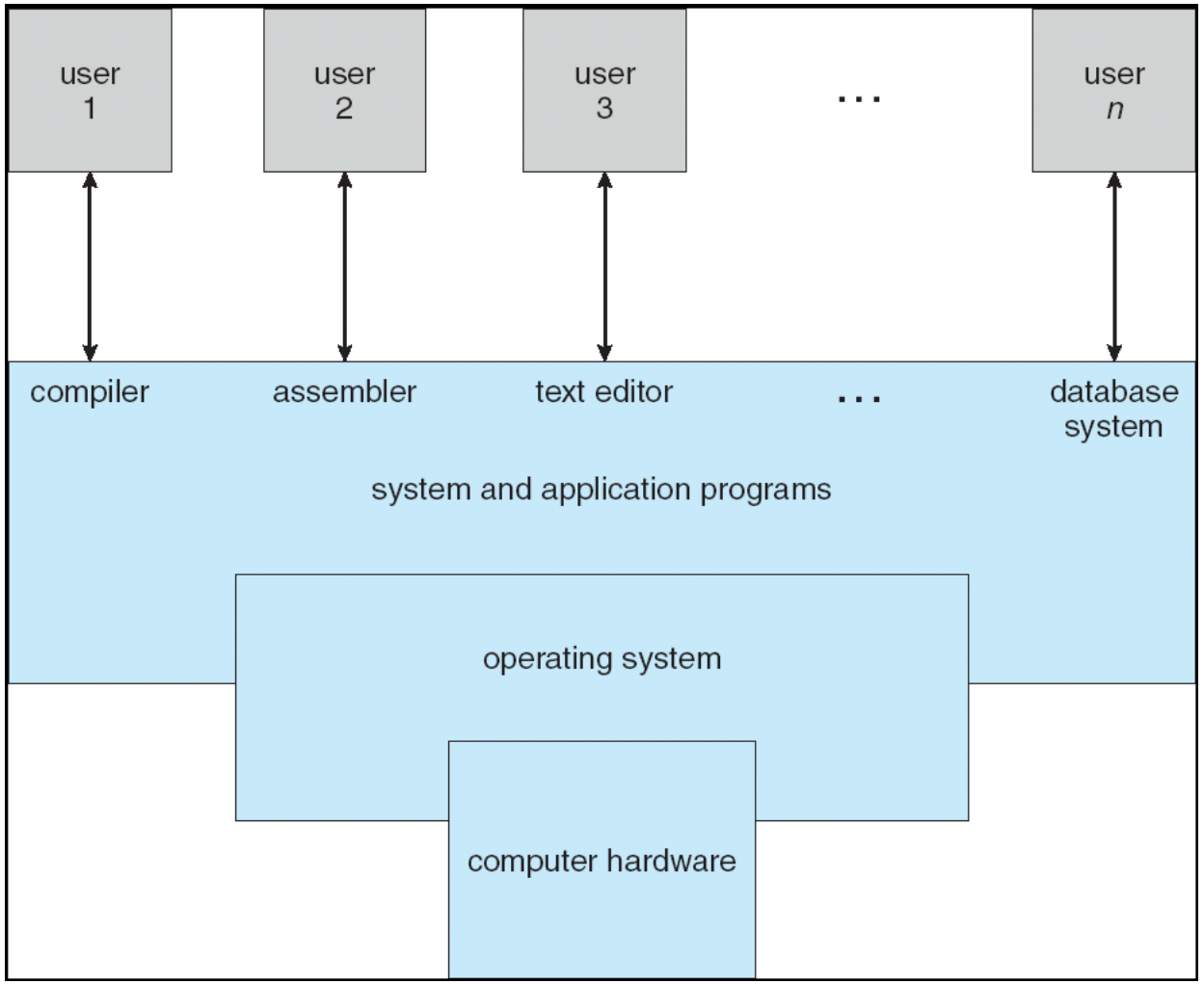
- Response Time (minimize)
  - Typically used in reference to interactive systems
  - The time it takes for the system to respond to a user request from submission to start of a response

Computer System Structure

Computer system can be divided into four components

- Hardware – provides basic computing resources
  - CPU, memory, I/O devices
- Operating system
  - Controls and coordinates use of hardware among various applications and users
- Application programs – define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, compilers, web browsers, database systems, video games
- Users
  - People, machines, other computers

Four Components of a Computer System



Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

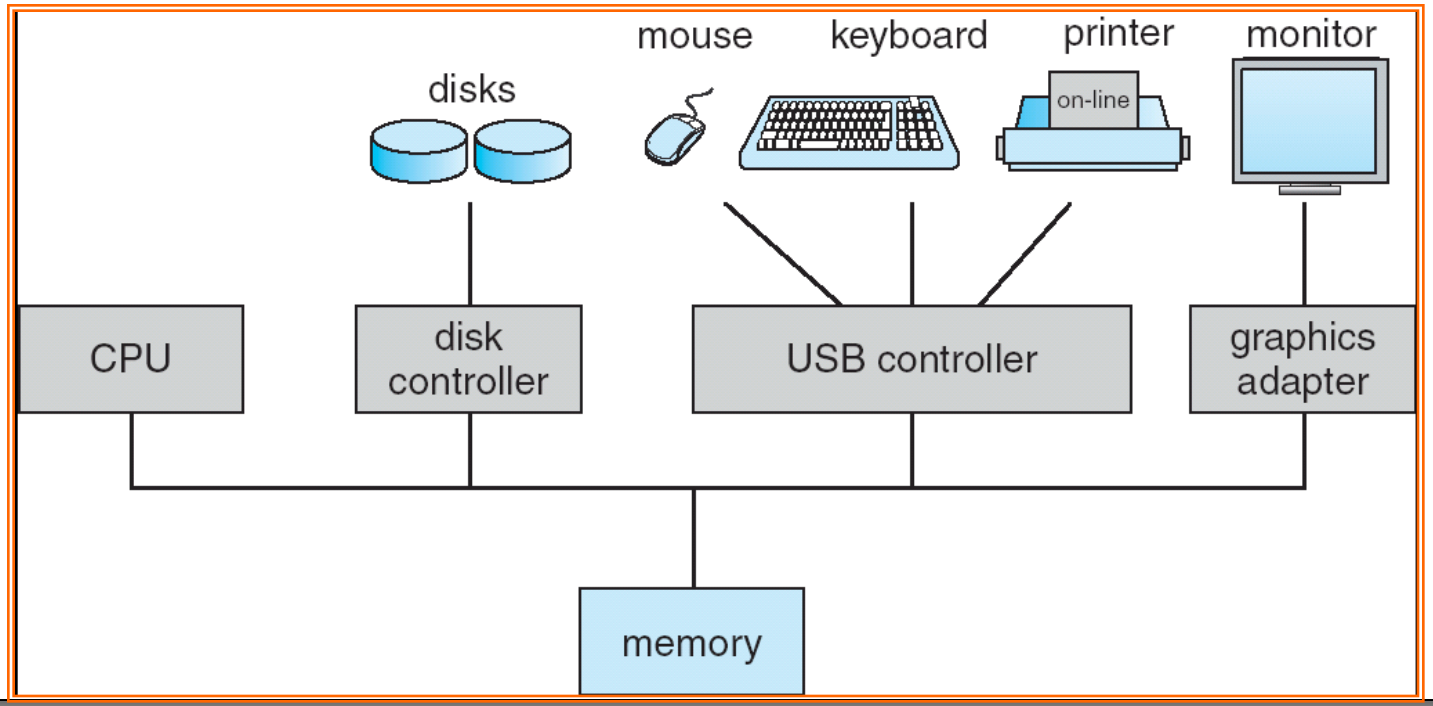
Operating System Definition (cont.)

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly, some OS require less than 1MB and do not even have a full-screen editor, while others require many GBs and are entirely based on graphical windowing systems.
  - Recall that in 1998 the U.S Department of Justice filed suit against Microsoft, in essence claiming that Microsoft included too much functionality in its OS and thus prevented vendors from competing.
- "The one program running at all times on the computer" is the **kernel.** Everything else is either a system program (ships with the operating system) or an application program

Computer System Organization

Computer-system operation

- One or more CPUs, device controllers connect through common bus providing access to shared memory
- Concurrent execution of CPUs and devices competing for memory cycles



Computer-System Operation

- 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.
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller.
- Device controller informs CPU that it has finished its operation by causing an *interrupt*.

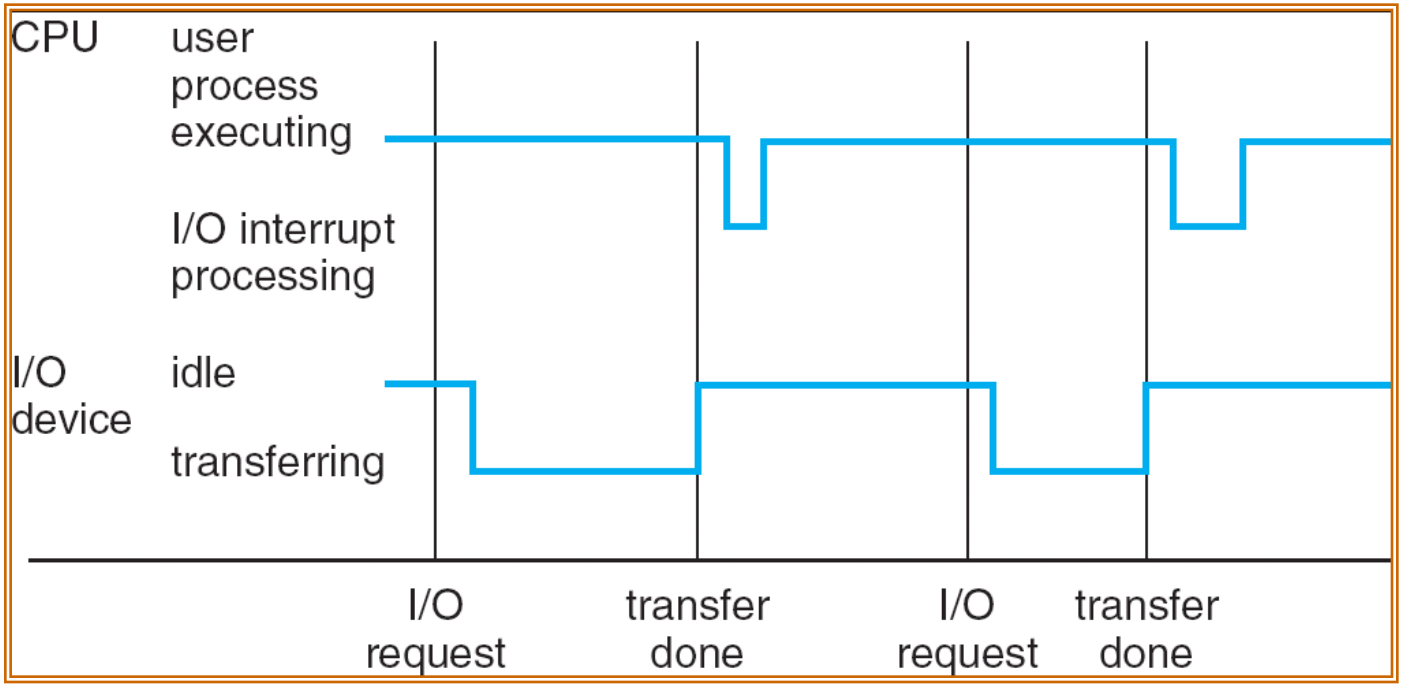
Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the *interrupt vector*, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted instruction.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*.
- A *trap* is a software-generated interrupt caused either by an error or a user request.
- An operating system is *interrupt driven*.

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
  - *polling*
  - *vectored* interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

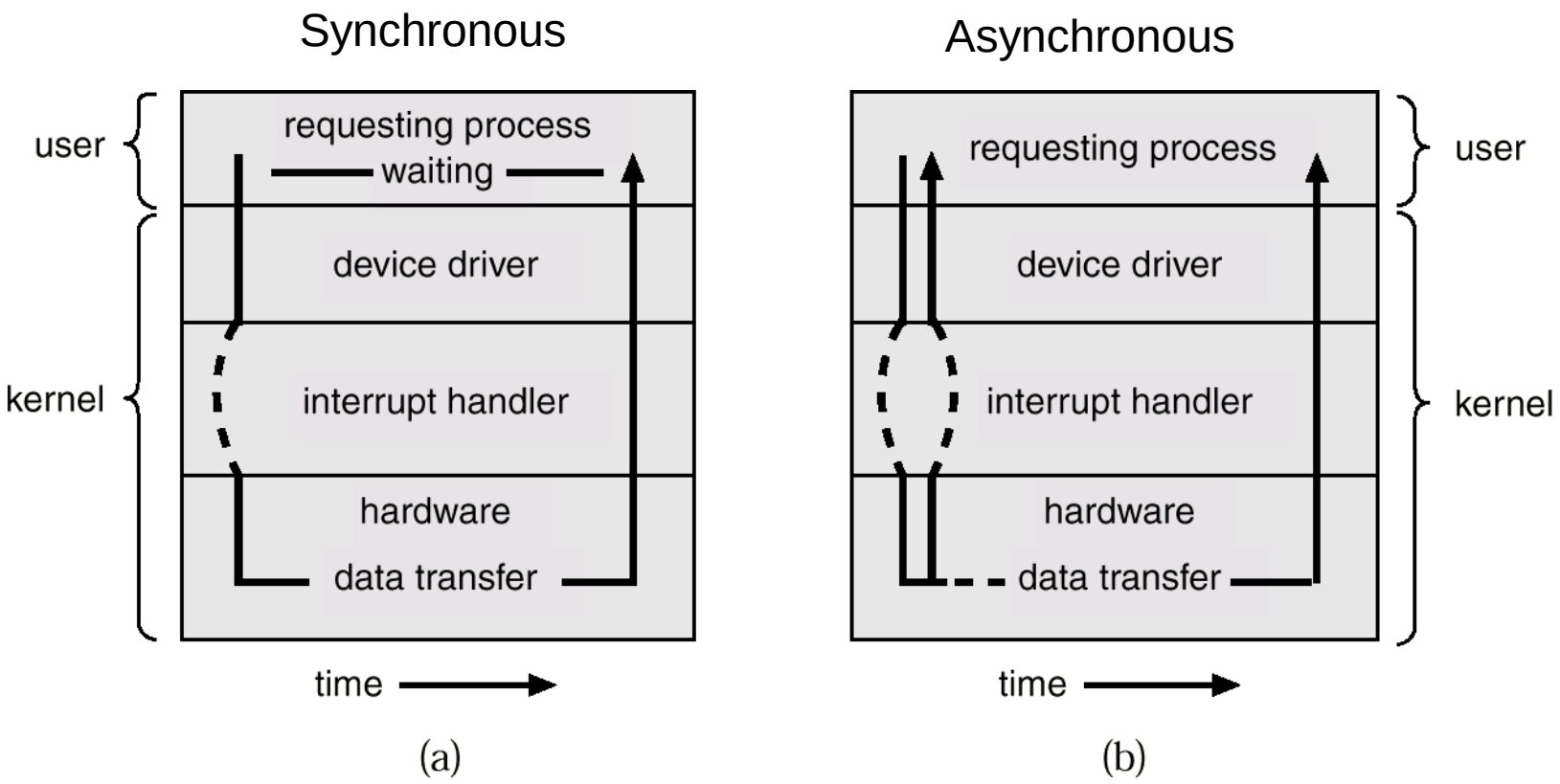
Interrupt Timeline



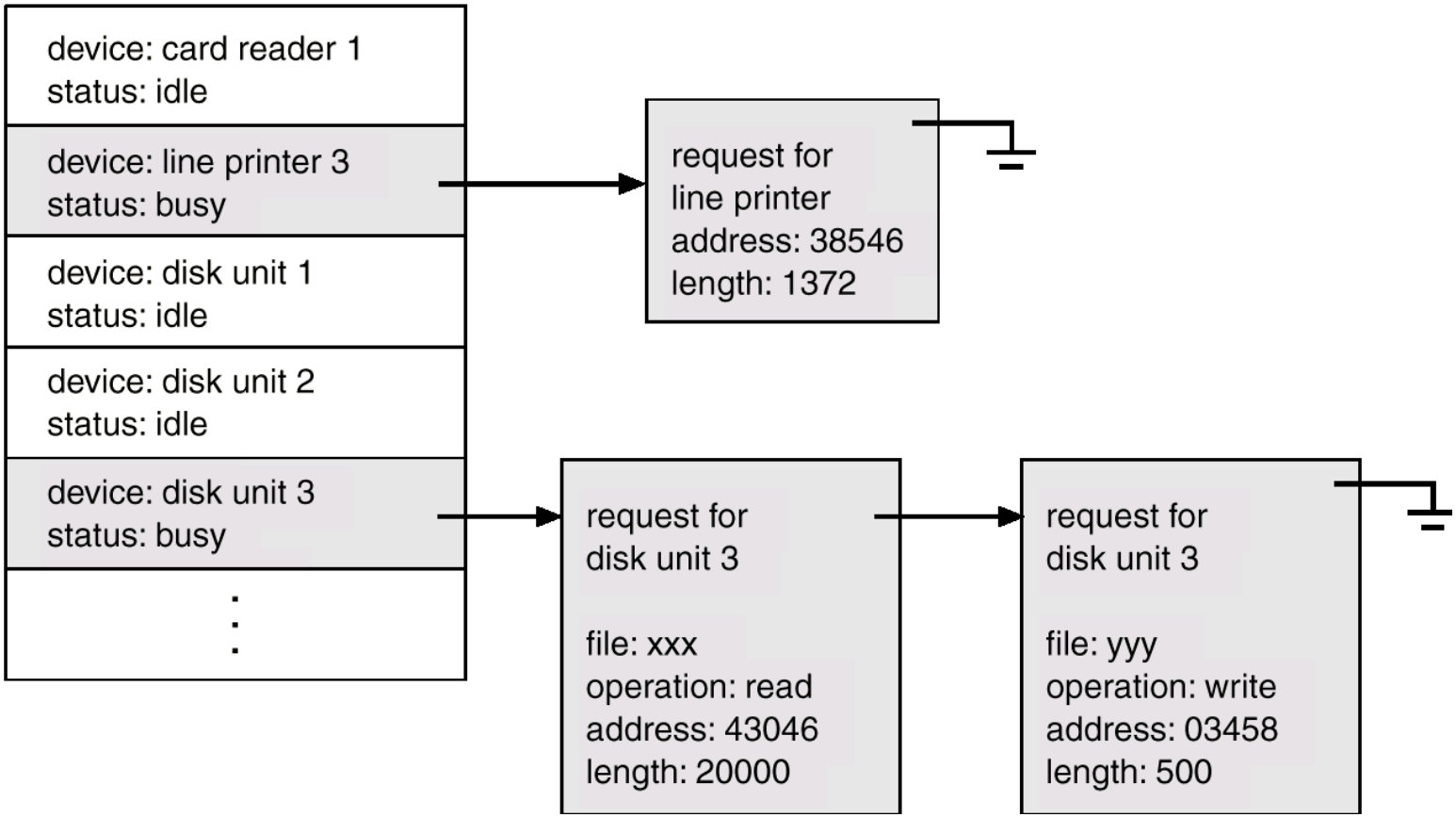
I/O Structure

- After I/O starts, control returns to user program only upon I/O completion.
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access).
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing.
- After I/O starts, control returns to user program without waiting for I/O completion.
  - *System call* – request to the operating system to allow user to wait for I/O completion.
  - *Device-status table* contains entry for each I/O device indicating its type, address, and state.
  - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt.

Two I/O Methods



Device-Status Table



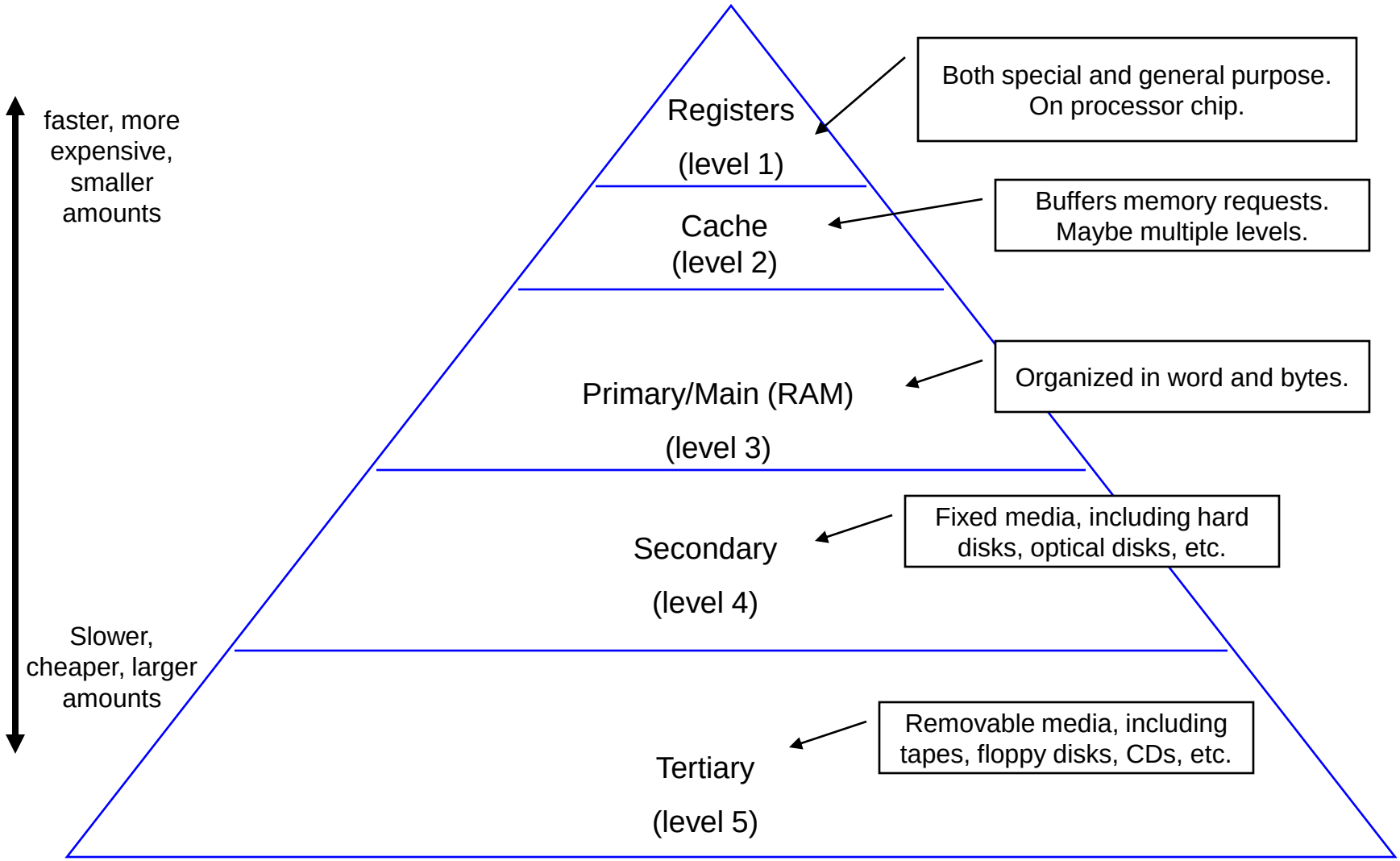
Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds.
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
- Only one interrupt is generated per block, rather than the one interrupt per byte.

Storage Structure

- Main memory – only large storage media that the CPU can access directly.
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity.
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into *tracks*, which are subdivided into *sectors*.
  - The *disk controller* determines the logical interaction between the device and the computer.

The Memory Hierarchy

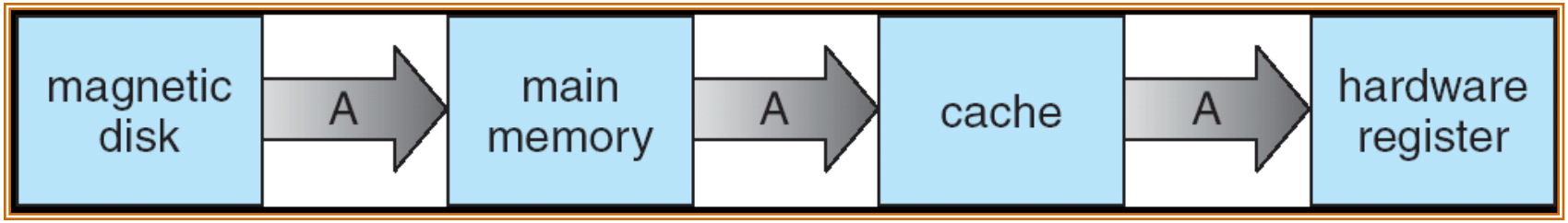


Performance of Various Levels of Storage

- Movement between levels of storage hierarchy can be explicit or implicit

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Migration of Integer A from Disk to Register



- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy.
- Multiprocessor environments must provide cache coherency in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist
  - We'll deal with this more complex situation later in the course.

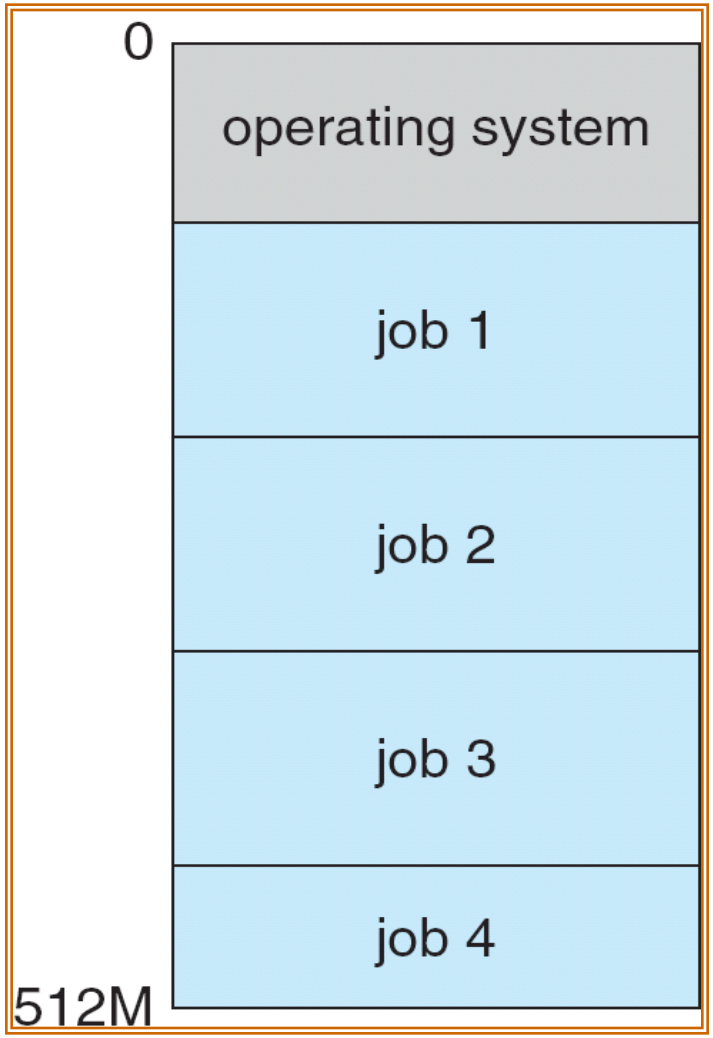
Operating System Structure

- **Multiprogramming** needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job

Operating System Structure (cont.)

- **Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing.
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory ⇨**process**
  - If several jobs ready to run at the same time ⇨ **CPU scheduling**
  - If processes don't fit in memory, **swapping** moves them in and out to run
  - **Virtual memory** allows execution of processes not completely in memory

Memory Layout for Multiprogrammed System

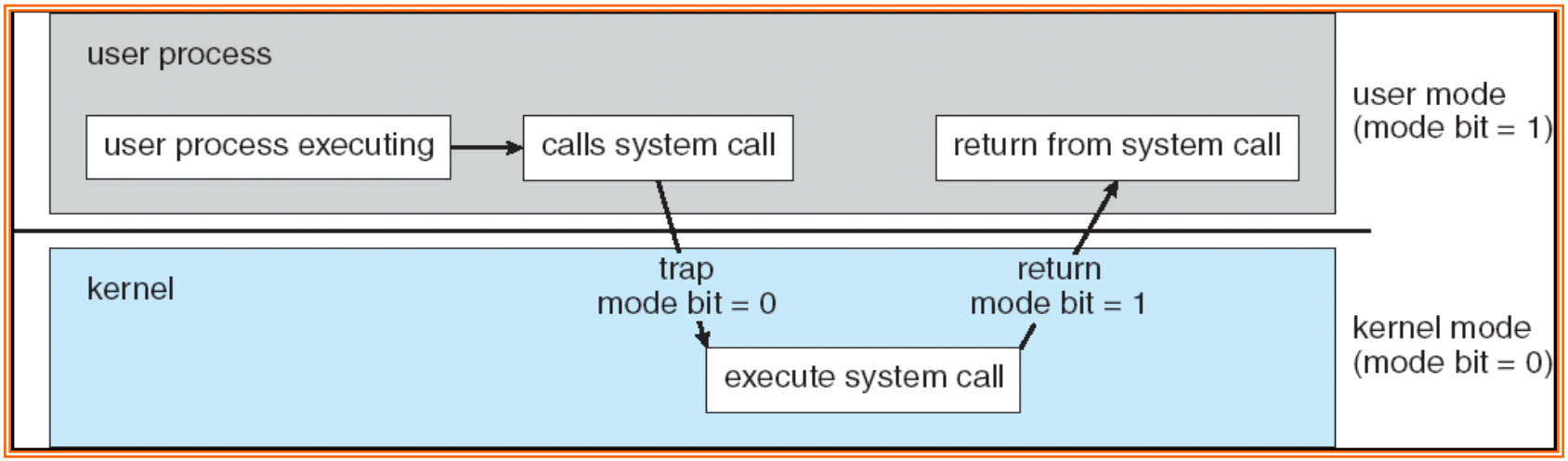


Operating-System Operations

- Interrupt driven by hardware
- Software error or request creates **exception** or **trap**
  - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- **Dual-mode** operation allows OS to protect itself and other system components. The two modes are: **User mode** and **kernel mode**
  - The **Mode bit** is provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



Process Management

- A process is a program in execution. It is a unit of work within the system. A program is a *passive entity*, a process is an *active entity*.
- A 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 directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time.
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed
  - Both WORM (write-once, read-many-times) and RW (read-write) are used

I/O Subsystem

- One purpose of OS is to hide peculiarities 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

Protection and Security

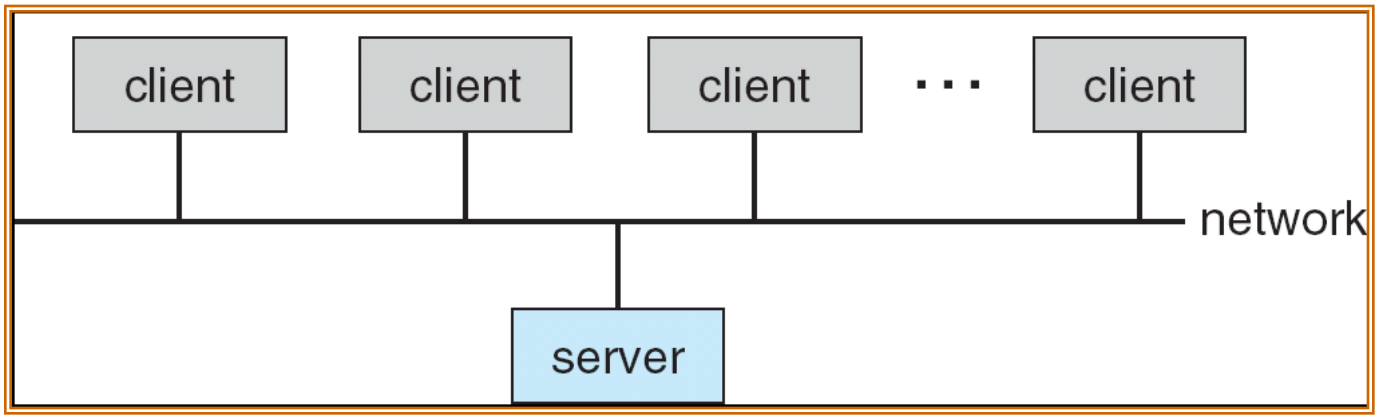
- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (g**roup ID**) allows set of users to be defined and controls managed, then also associated with each process, file
  - **Privilege escalation** allows user to change to effective ID with more rights

Computing Environments

- Traditional computer
  - Blurring over time
  - Office environment
    - PCs connected to a network, terminals attached to mainframe or minicomputers providing batch and timesharing
    - Now portals allowing networked and remote systems access to same resources
  - Home networks
    - Used to be single system, then modems
    - Now firewalled, networked

Computing Environments (cont.)

- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now **servers**, responding to requests generated by **clients**
    - **Compute-server** provides an interface to client to request services (i.e. database)
    - **File-server** provides interface for clients to store and retrieve files



Peer-to-Peer Computing

- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via *discovery protocol*
  - Examples include *Napster* and *Gnutella*

Web-Based Computing

- Web has become ubiquitous
- PCs most prevalent devices
- More devices becoming networked to allow web access
- New category of devices to manage web traffic among similar servers: **load balancers**
- Use of operating systems like Windows 95, client-side, have evolved into Linux and Windows 7, which can be clients and servers

Operating System Services

- An OS provides a number of services to a user or application program including:
  - Communications
  - Resource Allocation
  - Accounting
  - Protection
  - Error Detection
  - Program Execution
  - I/O Operations (explicit requests)
  - File System Manipulation

An Event Driven System

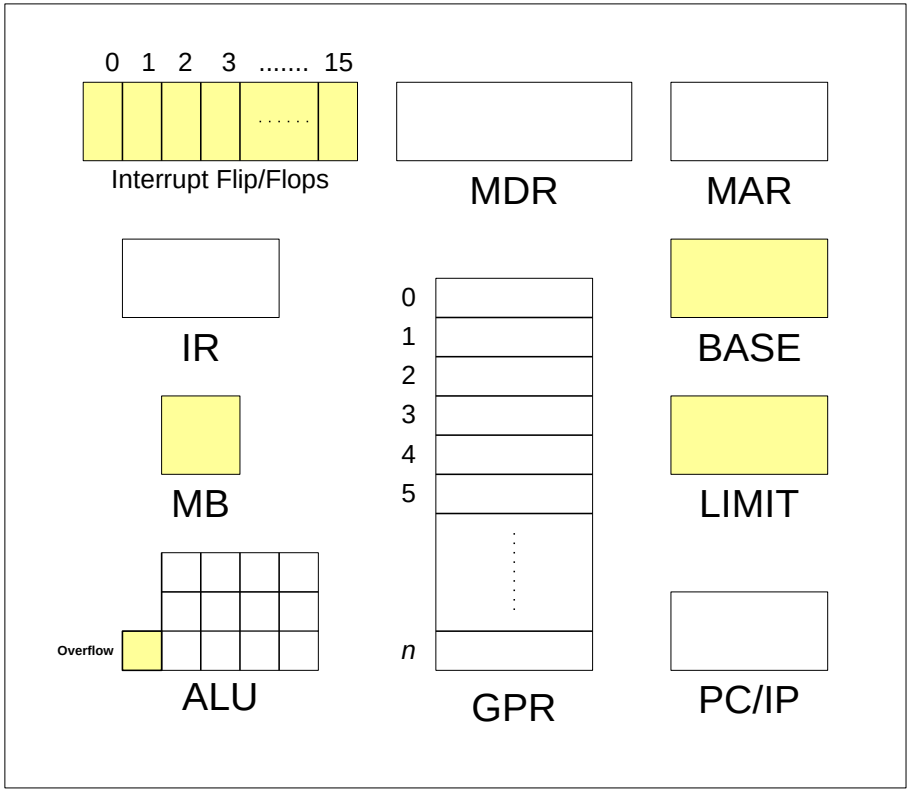
- OS services are performed on behalf of the user or application program in response to specific events.
  - The OS does not perform any "useful" work (e.g., solve a user's problem)
  - Unless prompted, the OS just stays out of the way as much as possible
- Events include:
  - Hardware Interrupts from external (non-processor) devices
  - Interrupts caused by execution of a program (aka software interrupts)
    - Supervisor Calls - User/Application request OS service
    - Trap - Error occurs during execution

Computing Cycle

- Modern computers operate using a three-step cycle:
  - Fetch
    - Get next instruction pointed to by the program counter
    - Increment the program counter
    - Decode the instruction and operands
  - Execute
    - Execute the instruction (if possible)
    - If instruction is a supervisor call, turn control over to OS
  - Interrupt
    - Inspect interrupt flags to determine if error has occurred (e.g., overflow trap) or hardware device requires attention (e.g., printer hardware interrupt)
    - If no interrupt, repeat cycle
    - If interrupt, turn control over to OS which executes appropriate interrupt handler

OS / Hardware Linkage

- Modern operating systems required the development of special hardware components to perform various services.



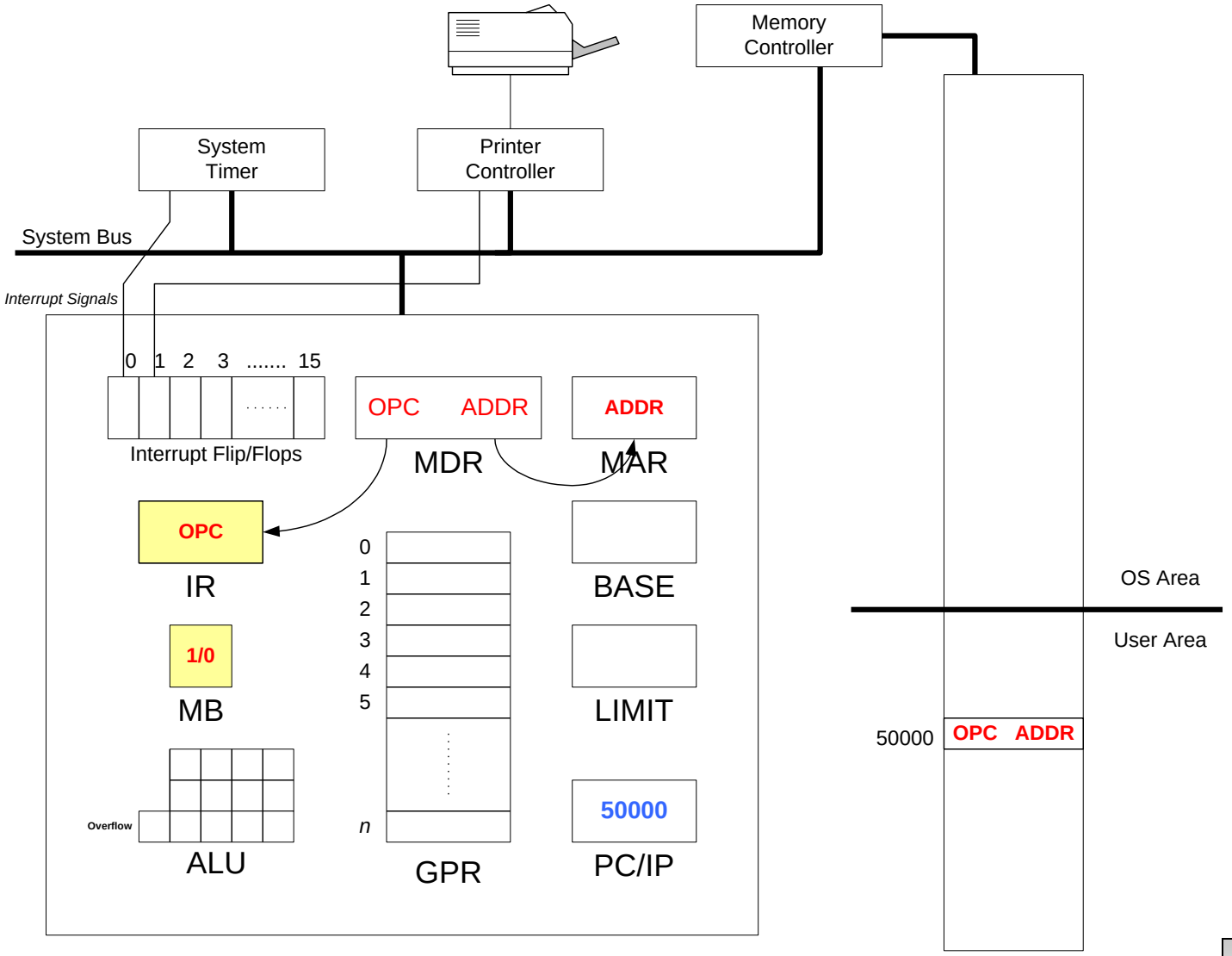
OS / Hardware Linkage (cont.)

MDR	**Memory Data Register** Instructions or data are loaded in this special purpose register from memory one word at a time over the system bus. Data to be stored in memory is placed here first, then sent to memory over the system bus.
MAR	**Memory Address Register** Contains the address of the word to be stored in / retrieved from memory.
BASE	**Base Address Register** Lowest physical memory address allocated to a process.
LIMIT	**Limit Address Register** Highest physical memory address allocated to a process.
PC/IP	**Program Counter / Instruction Pointer** Contains the address of the next instruction to be executed.
GPR	**General Purpose Registers** These registers are programmer accessible. Program data is stored here temporarily for use by the ALU.
IR	**Instruction Register** Operation codes are extracted from the MDR and copied to this register. During decoding the processor determines what operation to perform and what operands are required.
MB	**Mode Bit** Used in dual mode operation to indicate whether CPU is in user mode (1) or operating system / supervisor mode (0)
ALU	**Arithmetic Logic Unit** Performs the actual operations (add, sub, compare, etc) on program data.

Dual Mode Operation

- Uses a Mode Bit to indicate whether a user process or OS process is running
  - 0 = Supervisor Mode (aka monitor mode, system mode)
  - 1 = User Mode
- Allows OS to protect itself and user processes
- Only privileged instructions can be executed in supervisor mode
  - e.g., set mode bit, set timer, reset interrupts
- Can also be used for I/O protection
  - Make I/O instructions privileged.
  - Require system calls (SVC) for users to request I/O operations

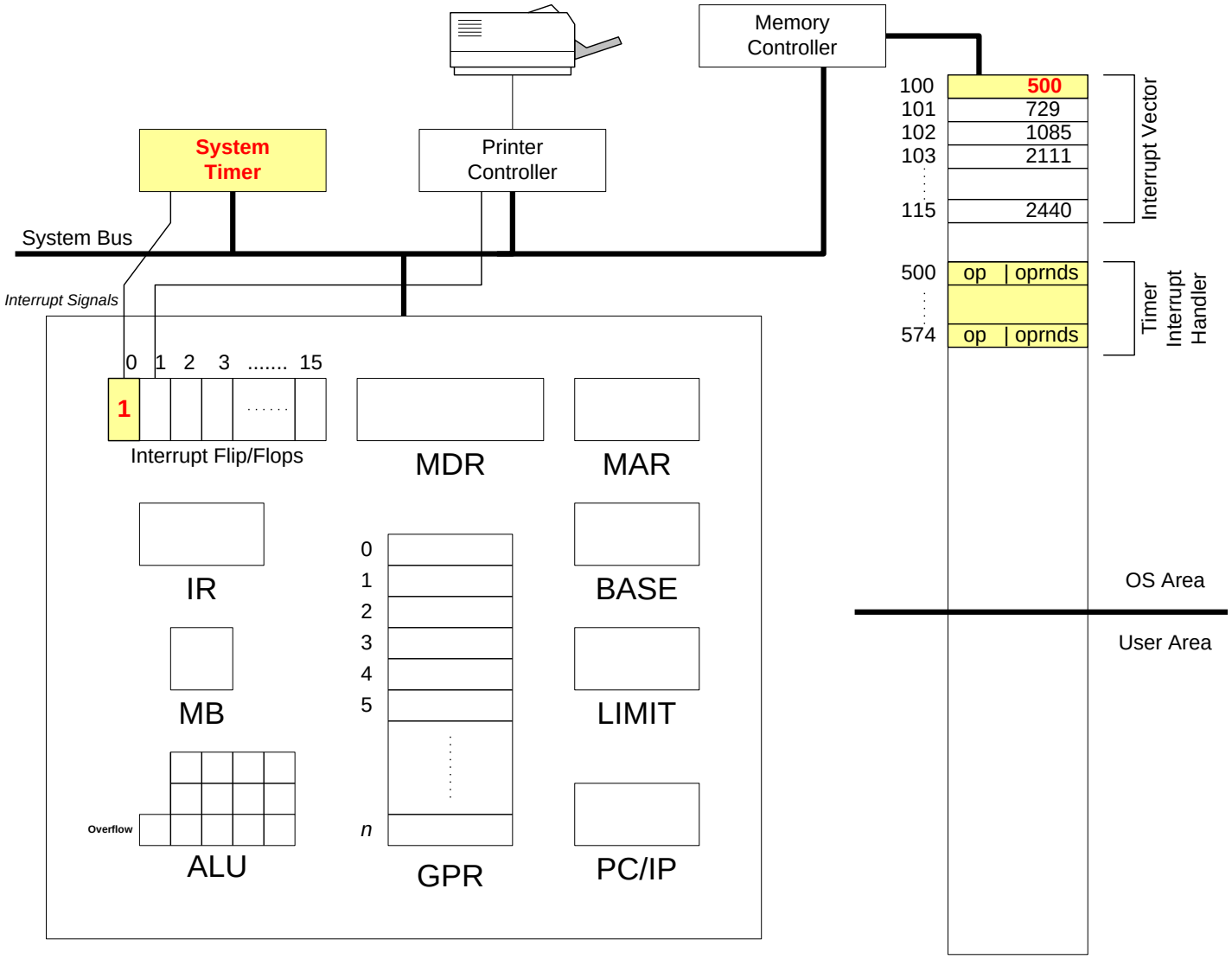
Dual Mode Operation (cont.)



CPU Protection

- Need to prevent a user process from monopolizing the CPU
  - e.g., catch infinite loops
- Uses a System Timer connected to an interrupt flip/flop
  - When timer counts down to zero, interrupt is raised
- Can also be used to implement time sharing

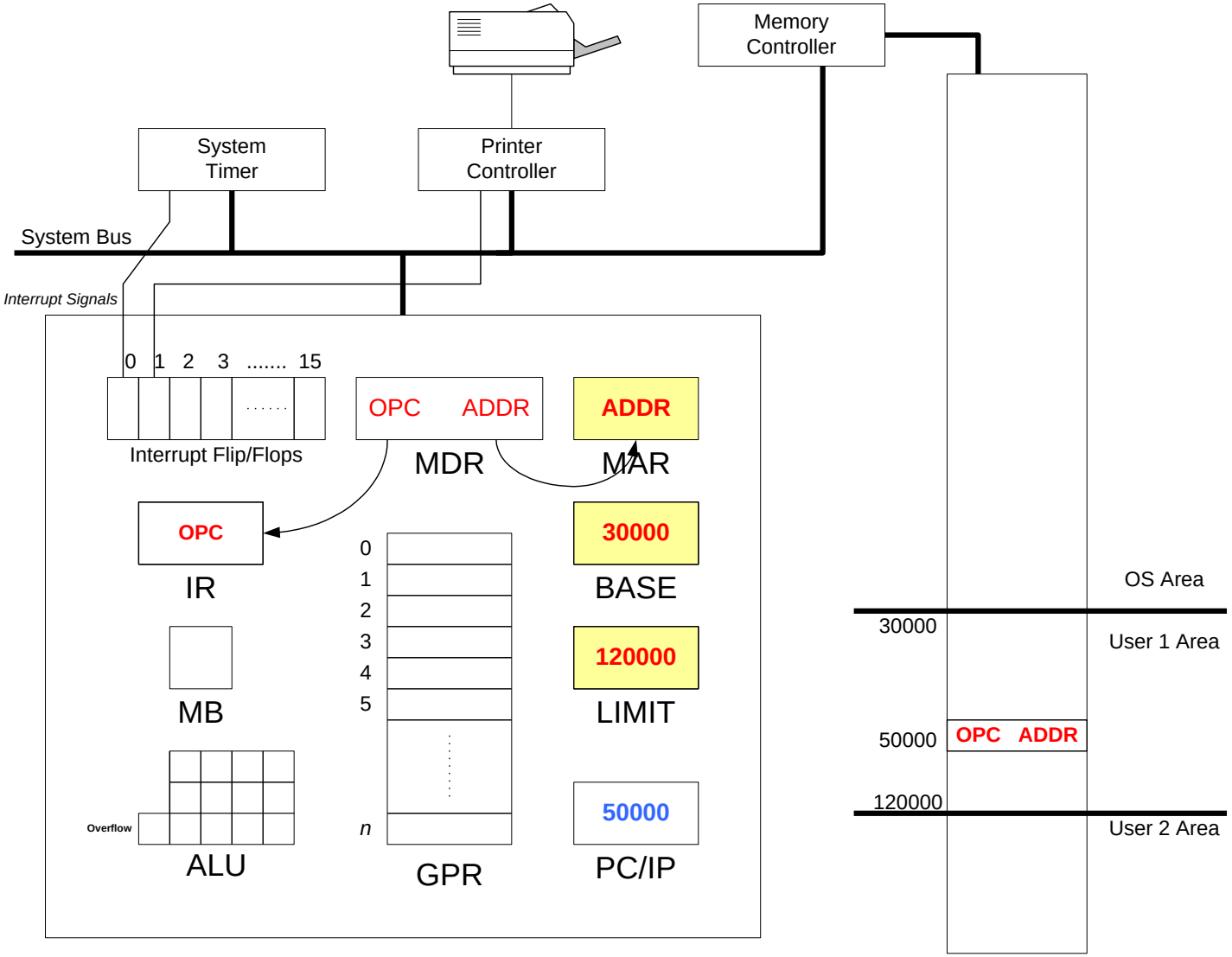
CPU Protection (cont.)



Memory Protection

- Required to prevent user processes from accessing / writing to memory outside of their allocated portion of RAM.
- In single-program environment uses a fence register
- For multi-programming, use base and limit
  - Base - Lowest address in process' memory allocation
  - Limit - Highest address in process' memory allocation
- Address of each memory location accessed / written to by user process is compared with base and limit (or fence)
  - If "out of bounds" traps to OS

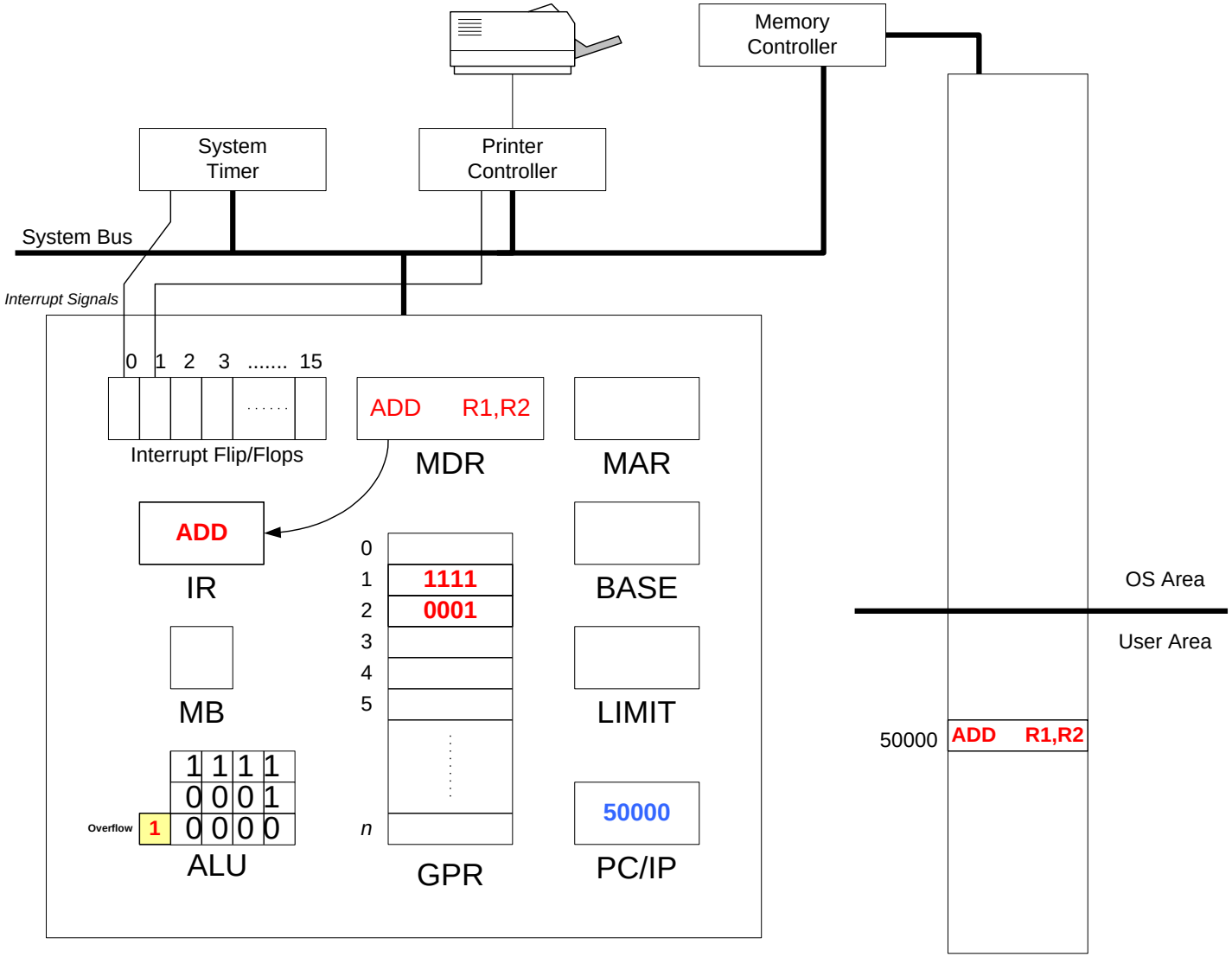
Memory Protection (cont.)



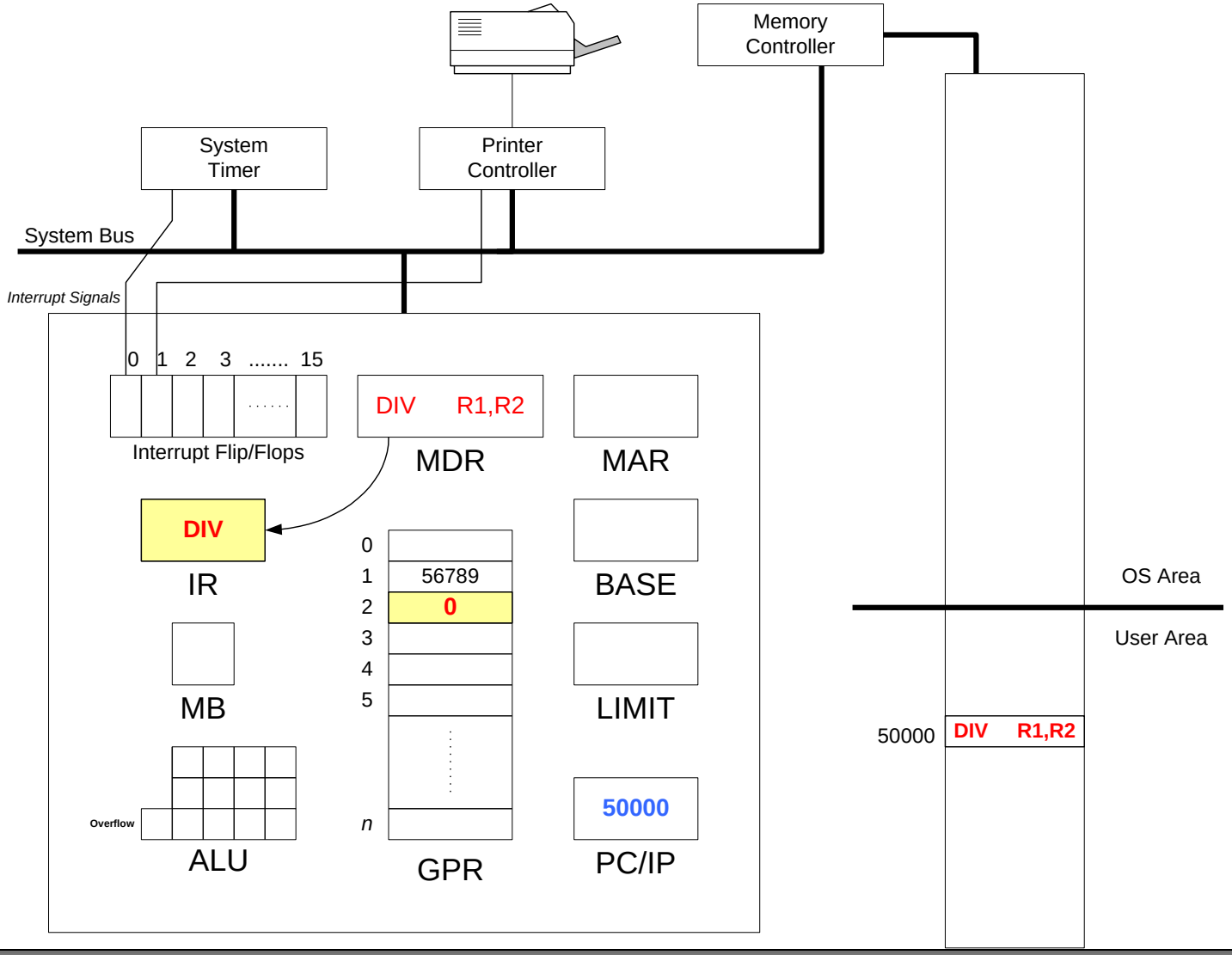
Error Detection

- OS can trap various errors that occur during execution of a user process
- Math Overflow:
  - Occurs when adding two numbers and result too big for accumulator
  - Overflow bit changes to “1” if error occurs
- Divide by Zero
  - Occurs when dividing by zero value
  - Hardware compares opcode with value of divisor.
  - If opcode is DIV and divisor = 0 then trap to OS

Error Detection Math Overflow



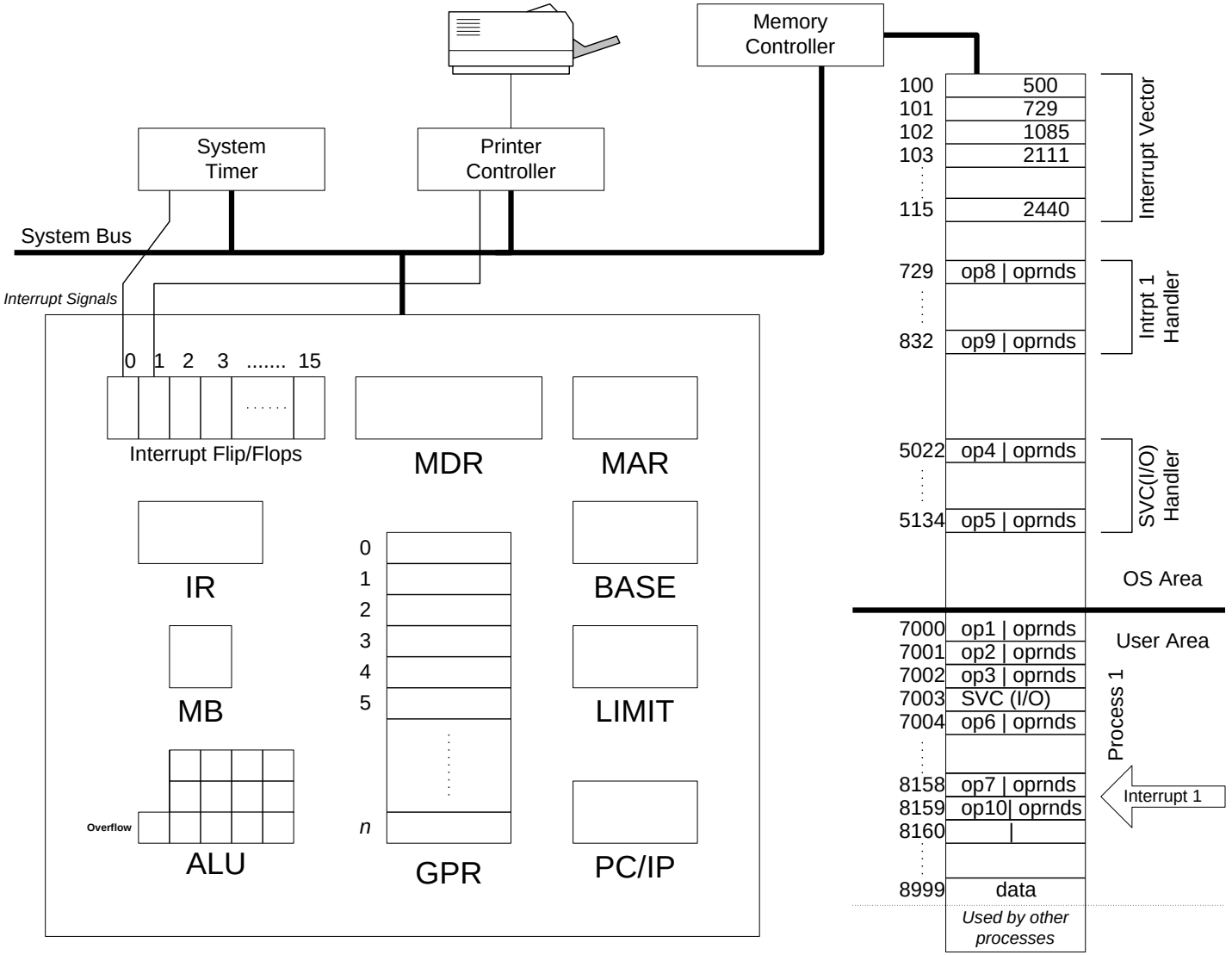
Error Detection - Divide by Zero



Handling I/O Operations

- I/O operations typically initiated by user request
  - System or Supervisor Calls (SVC)
- OS communicates user's request to appropriate I/O device controller
  - Synchronous I/O - control returned to user process after I/O completes
  - Asynchronous I/O - control returned immediately back to user process
- OS notified of I/O completion when device controller raises hardware interrupt flag
  - Device can also use interrupt to signal status (e.g., out of paper, no media)

Handling I/O Operations (cont.)



Processing Interrupts

- Interrupt Vector - a series or array of addresses stored in lower memory which point to each interrupt handler.
- Interrupt Handlers - portions of OS program specifically written to deal with each interrupt type

- Remember.....
  - A process is a program in execution
  - An OS is a program
  - Therefore, OS is also a process
  - Very large OS, may spawn multiple processes

When An Interrupts Occurs:

- Mode bit set to supervisor mode (0)
- Based on which flip/flop raised, retrieves address of interrupt handler from the interrupt vector and places in PC
- OS begins executing:
  - Handler must save PC and Registers for current user process
    - Context Switch
  - Run the interrupt handler (take care of problem)
  - Interrupt handler may disable or mask interrupts during processing (don't want to interrupt the interrupt handler)
- Upon completion, OS:
  - Resets, unmasks and/or enables interrupt flip/flops
  - Restores user process registers and program counter
  - Sets mode bit to 1 (user mode)
  - Loads PC with next instruction in user process