Operating System Structures

OS Structures

- Layered Structure
- Multi-kernels for Multi-core processors
- Microkernel Structure
- Virtual Machine Structure

Layered System Structure

- System structured as a series of levels - each level performs a related subset of functions
- Each level relies on the next lower level for more primitive functions
- Layer Concepts
  - kernel
  - Basic services and I/O drivers
  - Resource managements

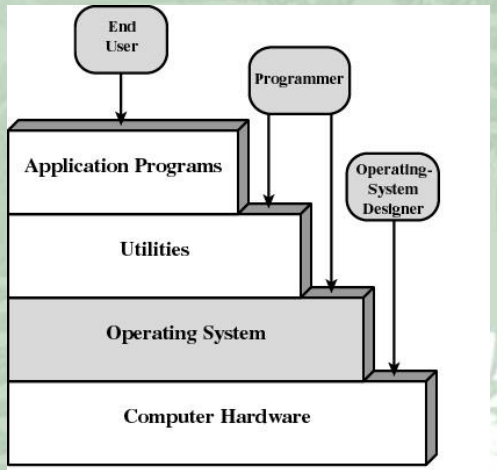
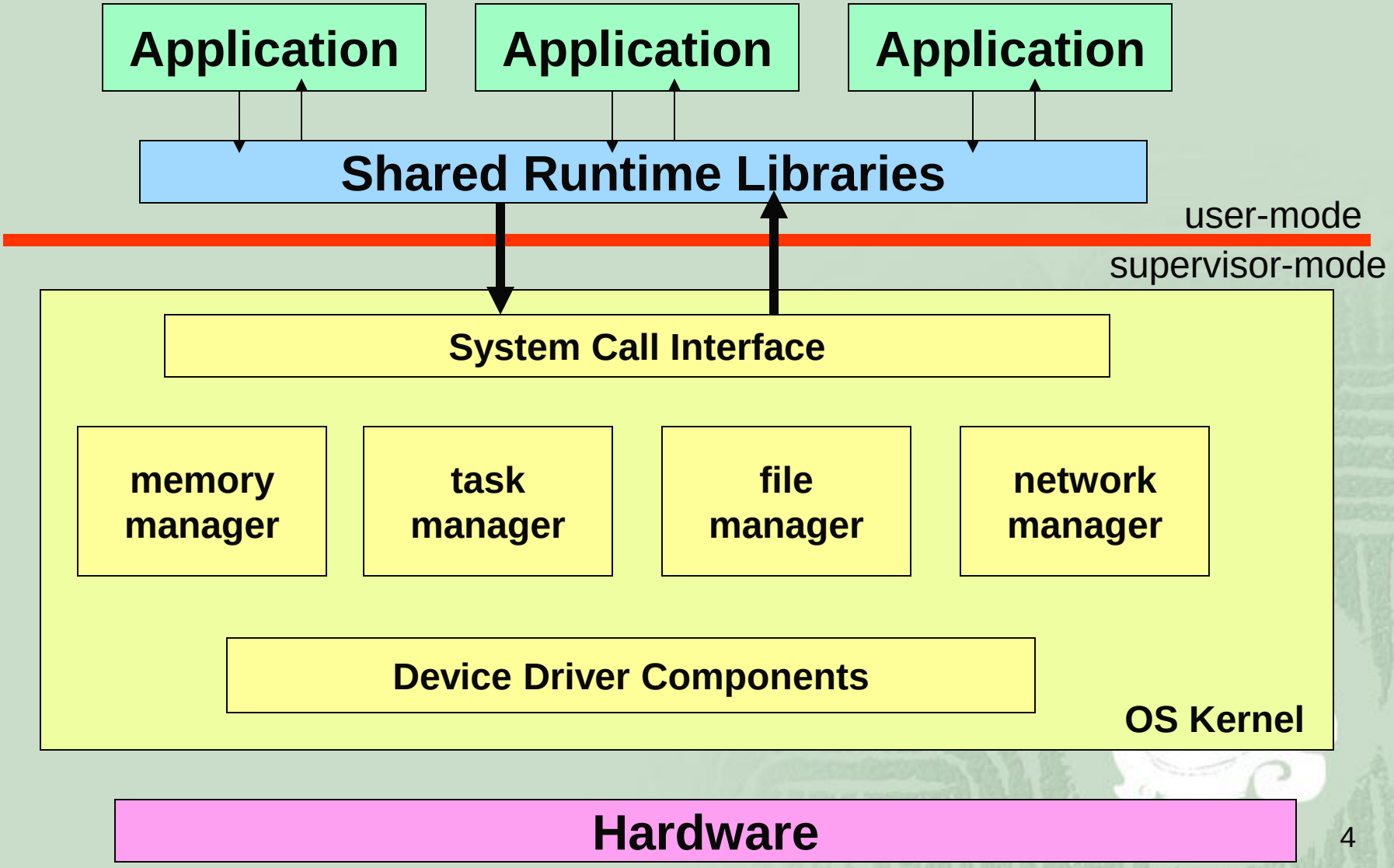


Figure 2.1 Layers and Views of a Computer System

A Modern OS Design



Modern OS Architecture

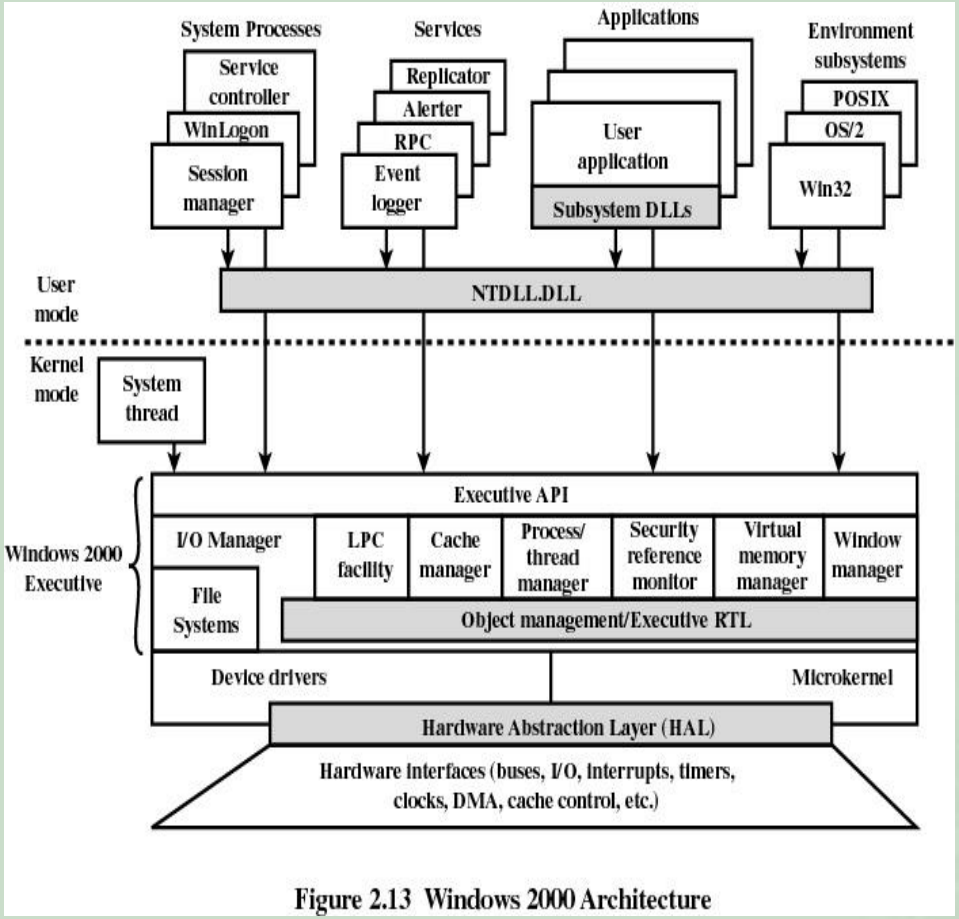


Figure 2.13 Windows 2000 Architecture

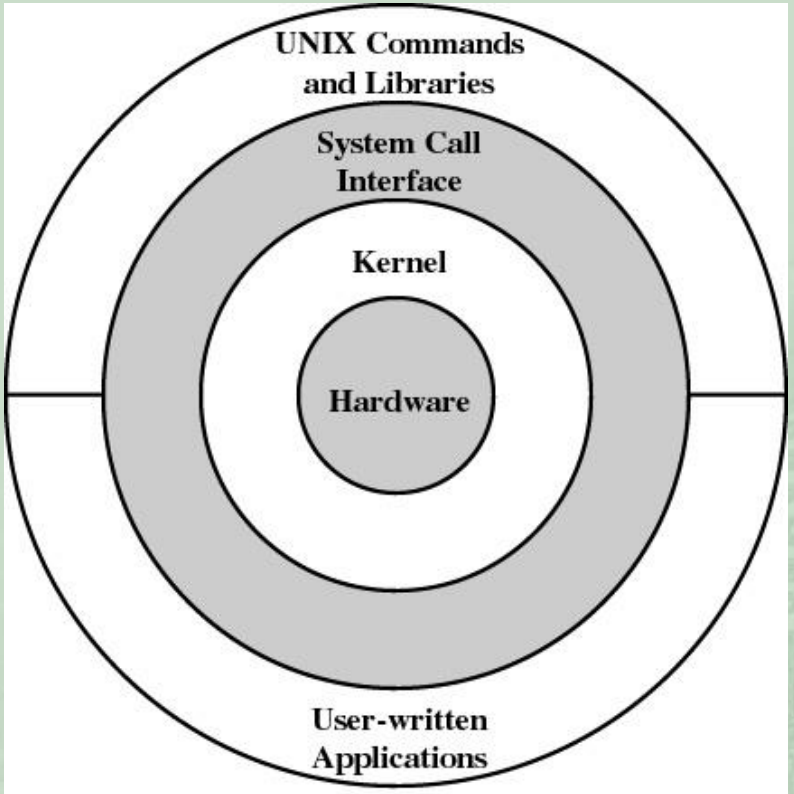
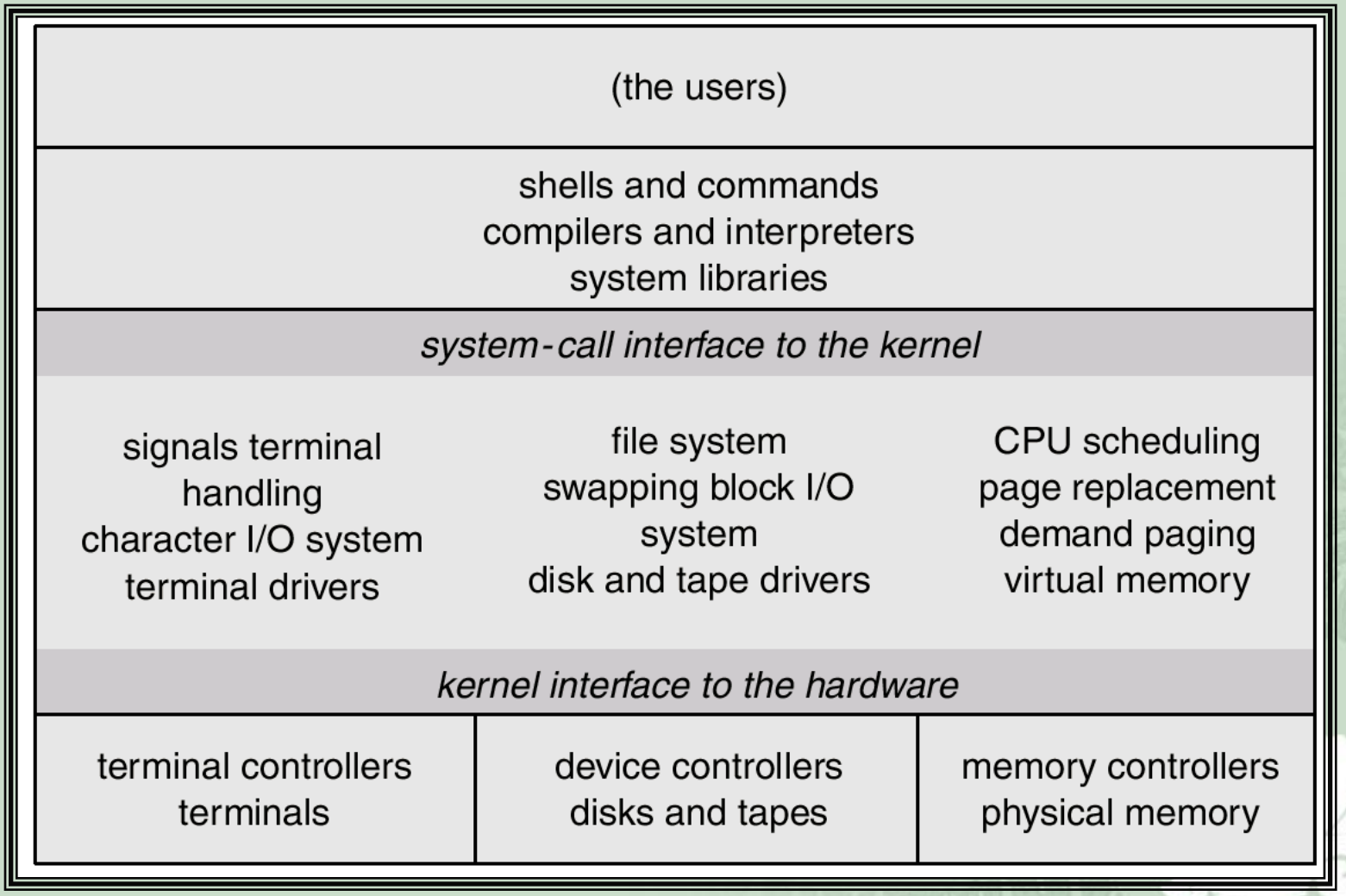
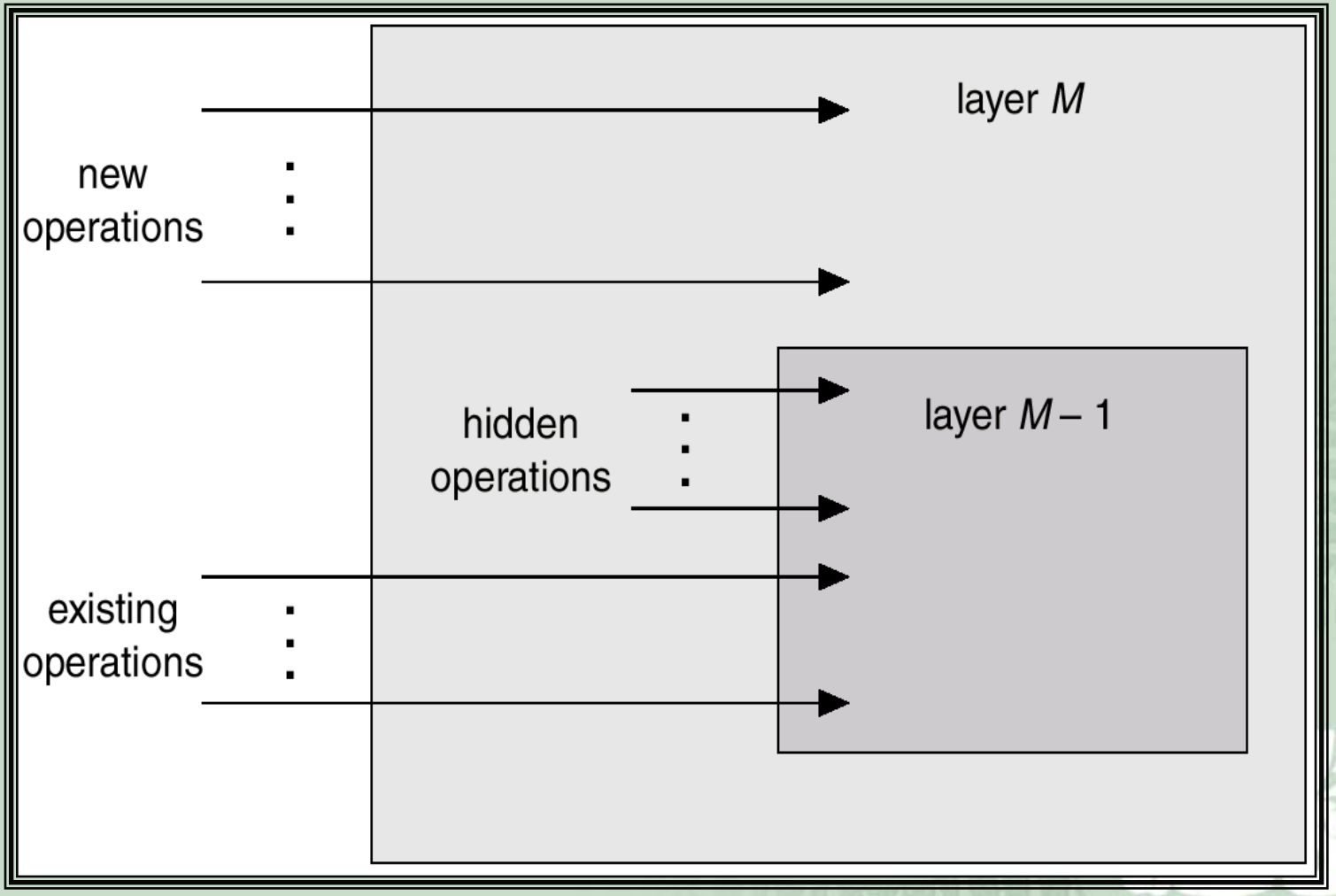


Figure 2.15 General UNIX Architecture

UNIX System Structure



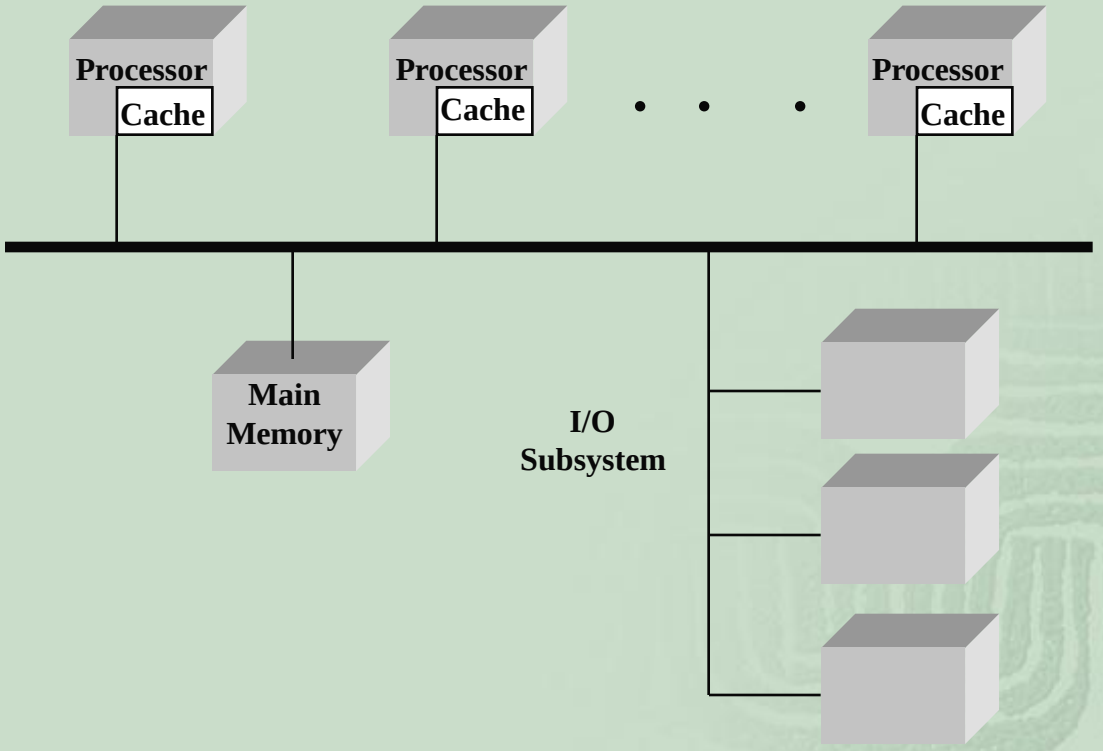
An Operating System Layer



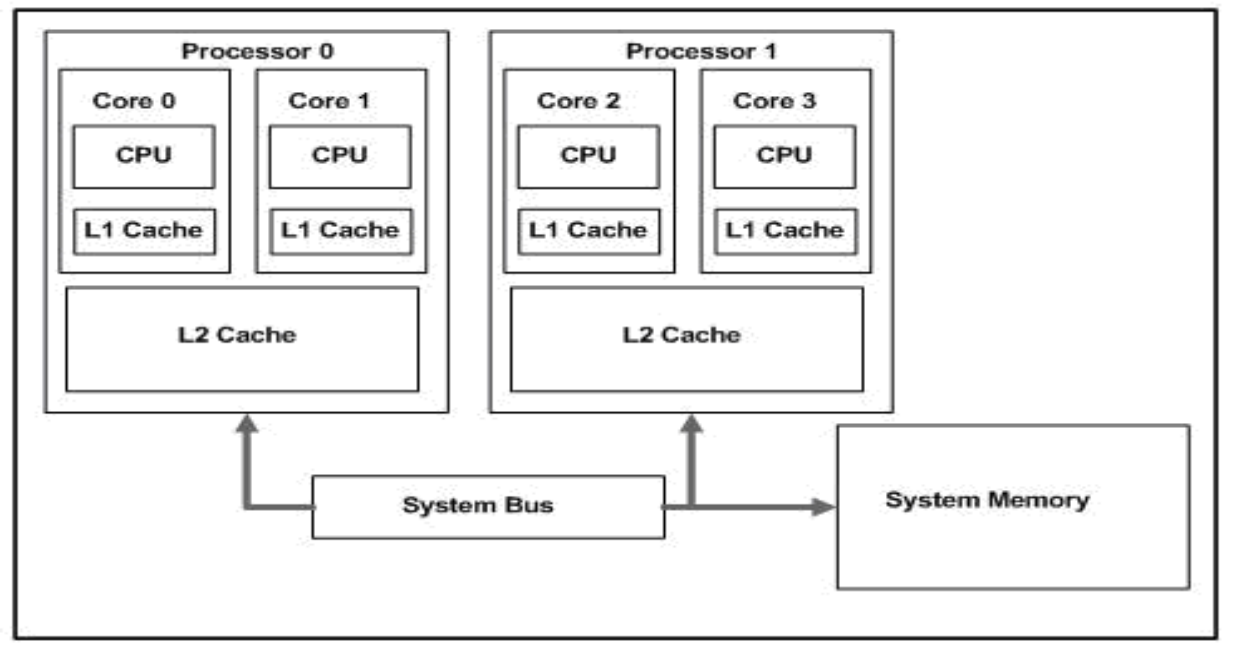
OS Structures

- Layered Structure
- Multi-kernels for Multi-core processors
- Microkernel Structure
- Virtual Machine Structure

Multiprocessor Organization



Multi-core Processors



- In a multi-core environment, the control over which core to run a specific thread or application is essential.
- Without this control, the threads/applications may get assigned to the wrong processors or cores and cause unnecessary performance degradation.

Multi-core Processors

- Typically each processor does self-scheduling from the pool of available process or threads
  - Timer interrupt
  - Ready queue
- OS Support
  - Any thread (including kernel threads) can run on any processor
  - *Soft affinity (close relation)* – Try to reschedule a thread on the same processor
  - *Hard affinity* – Restrict a thread to certain processors

Multi-core OS Design Issues

- Generally each processor has its own cache, shared cache, shared memory and I/O
- Design issues
  - Single kernel or Multi-kernel
    - Kernel routines must be reentrant to allow multiple threads to execute them
  - Scheduling
    - Must avoid conflicts
    - May be able to run processes or threads concurrently
  - Synchronization
    - Mutual exclusion, event ordering
  - Memory management
    - Deal with multiport memory
    - Have a unified paging scheme
  - Reliability and fault tolerance

OS Structures

- Layered Structure
- Multi-kernels for Multi-core processors
- Microkernel Structure
- Virtual Machine Structure

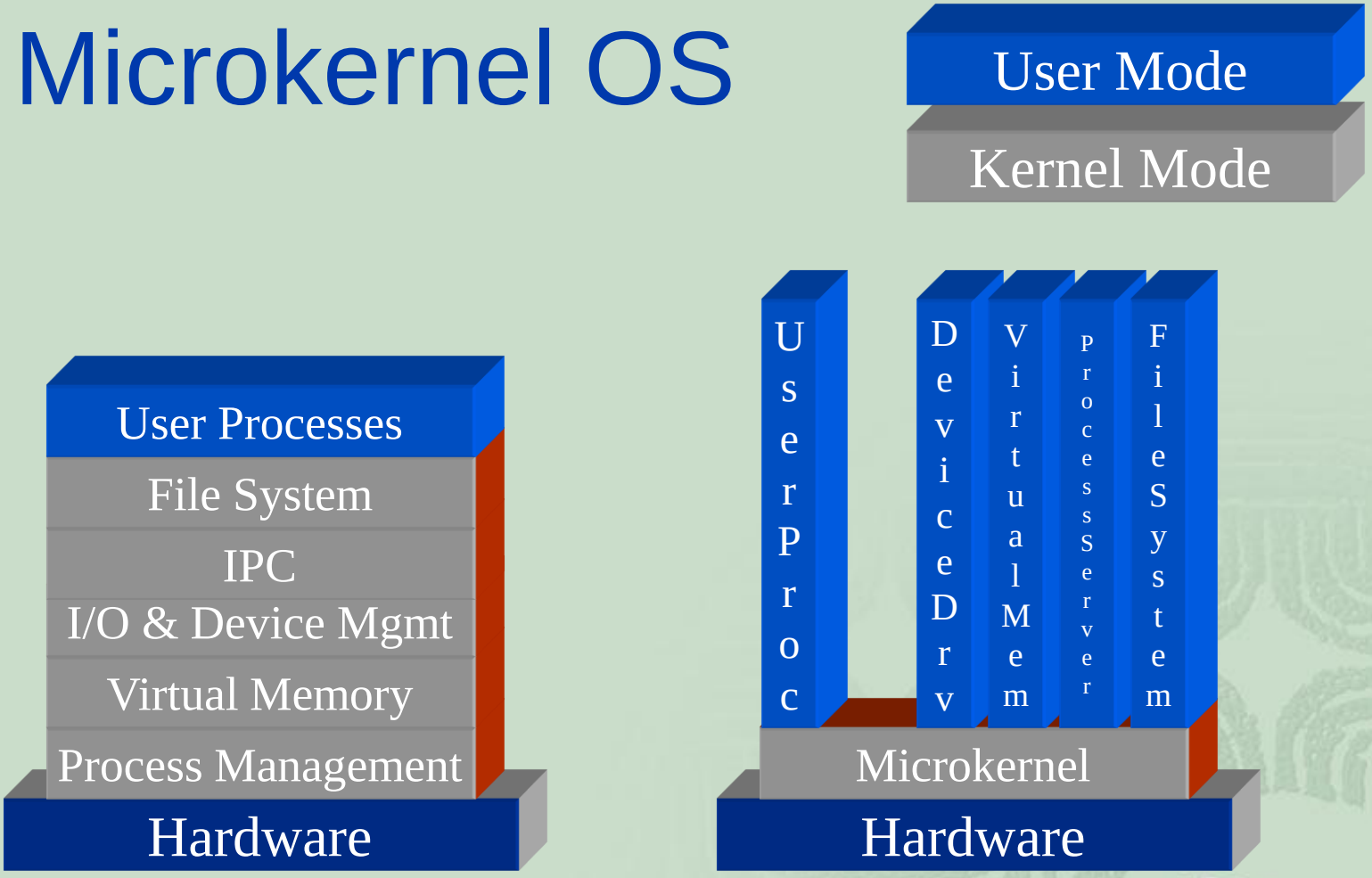
Microkernel Structure

- Monitors (1960s):
  - Built as a single large program, any routine can call any other routine
  - Used in most early systems
- Layered O.S (1970s – now):
  - Based on modular programming
  - Major changes still had wide-spread effects on other layers
- Microkernel (1980s – now):
  - Only essential functions in the kernel
  - File System, Device drivers, etc., are now external subsystems/processes
  - Processes interact through (IPC) messages passed through the kernel

Microkernel

- Identify and isolate a small operating system core that contains only essential OS functions
- Move many services included in the traditional kernel OS to external subsystems
  - device drivers
  - file systems
  - virtual memory manager
  - windowing system and security services

Microkernel OS



Traditional OS

Microkernel OS

Microkernel Design

- Primitive Memory Management
  - Kernel handles virtual → physical mapping, rest is a user mode process
    - VM module can decide what pages to move to/from disk
    - Module can allocate memory
  - Three microkernel memory operations
    - Grant – Grant pages to someone
    - Map – Map pages in another space
    - Flush – Reclaim pages granted or mapped
- Inter-process Communication (IPC)
  - Based on messages
- I/O and Interrupts
  - Handle interrupts as messages

Microkernel Benefits

- Uniform Interface
  - Same message for user/system services
- Extensibility
  - Easy to add new services
  - Modifications need only change directly affected components
  - Could have multiple file services
- Flexibility
  - Can customize system by omitting services
- Portability
  - Isolate nearly all processor-specific code in the kernel

Microkernel Benefits (continued…)

- Reliability
  - Easy to test kernel
  - Fewer system calls to master
  - Less interaction with other components
- Distributed System Support
  - Just as easy to send a message to another machine as this machine
    - Need system-wide unique Ids
  - Processes don’t have to know where a service resides
- Object-Orientated O.S.
  - Lends OO disciplines to the kernel

Kernel Performance

- Sending a message generally slower than simple kernel call (soft interruption)
- Depends on size of the microkernel
  - First generation systems slower
  - Then tried to include critical system items into kernel (Mach OS)
    - Fewer user/system mode switches
    - Lose some microkernel benefits
  - Trying approach of very small kernel
    - Mach OS L4 kernel (version 2) - 12K code, 7 system calls. Speed seems to match Unix
    - Mac OS X adopt this concept

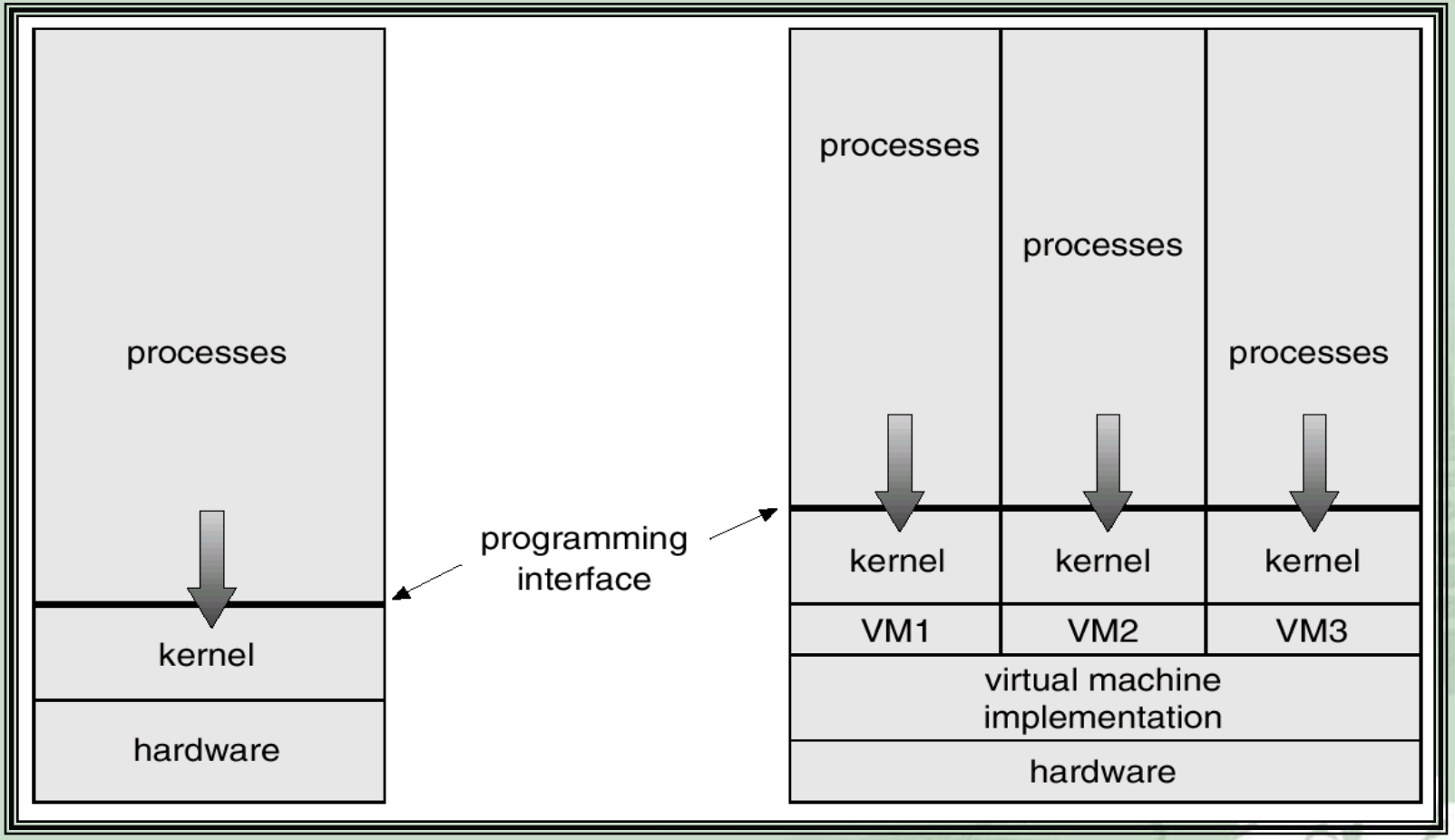
OS Structures

- Layered Structure
- Multi-kernels for Multi-core processors
- Microkernel Structure
- Virtual Machine Structure

Virtual Machine Structure

- A *virtual machine* takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware.
- A virtual machine provides an interface *identical* to the underlying bare hardware.
- The operating system creates the illusion of multiple processes, each executing on its own processor with its own (virtual) memory.

System Models



Non-virtual Machine

Virtual Machine

Advantages/Disadvantages of VM's

- The virtual-machine concept provides complete protection of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.
- A virtual-machine system is a perfect vehicle for operating-systems research and development.
- System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.
- The virtual machine concept is difficult to implement due to the effort required to provide an *exact* duplicate to the underlying machine.

Java Virtual Machine

- Compiled Java programs are platform-neutral bytecodes executed by a Java Virtual Machine (JVM).
- JVM consists of
  - class loader
  - class verifier
  - runtime interpreter
- Just-In-Time (JIT) compilers increase performance

Java Virtual Machine