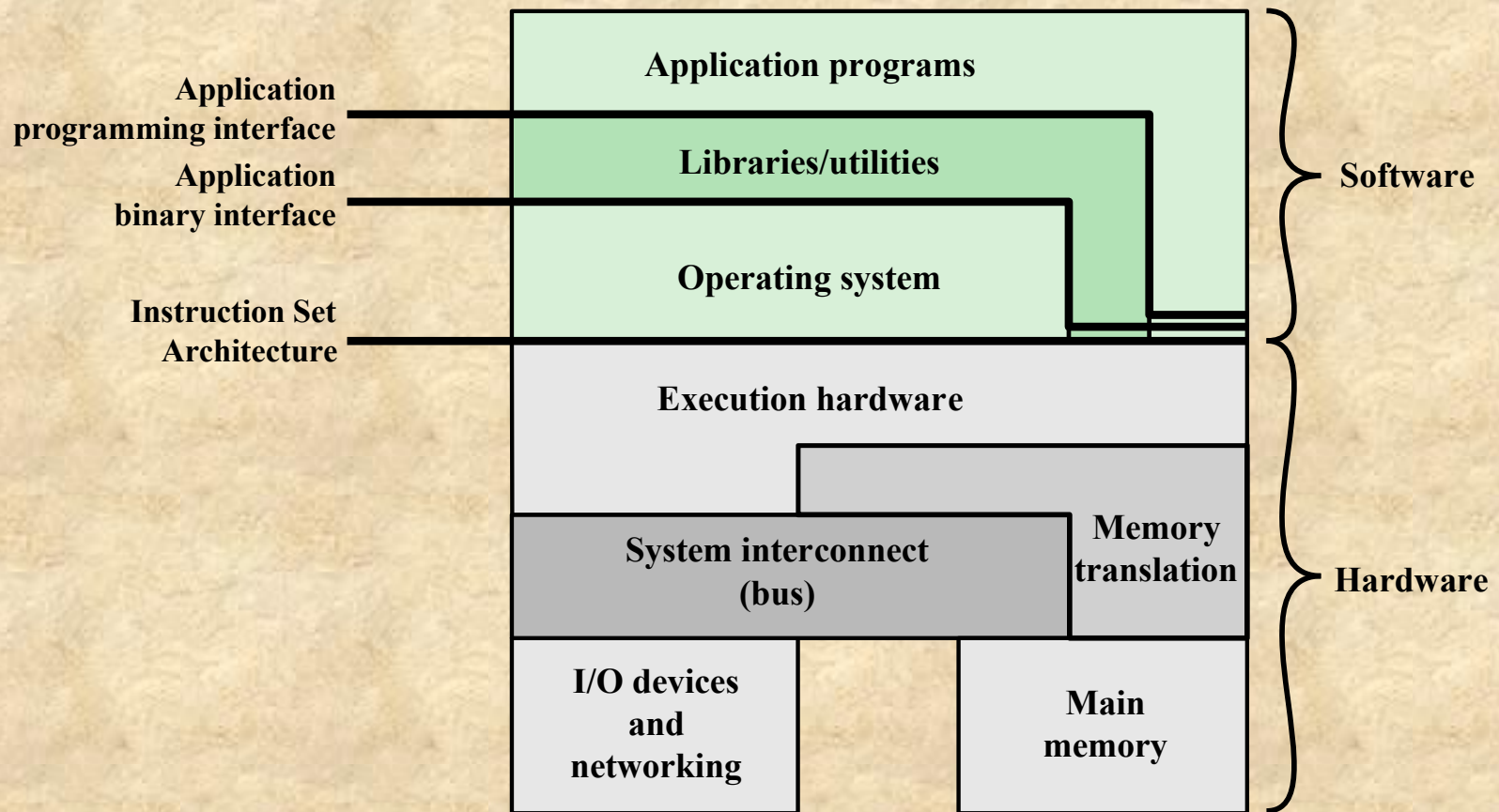


# Operating System

- A program that controls the execution of application programs
- An interface between applications and hardware

## Main objectives of an OS:

- Convenience
- Efficiency
- Ability to evolve



**Figure 2.1 Computer Hardware and Software Structure**

# Operating System Services

- Program development
- Program execution
- Access I/O devices
- Controlled access to files
- System access
- Error detection and response
- Accounting

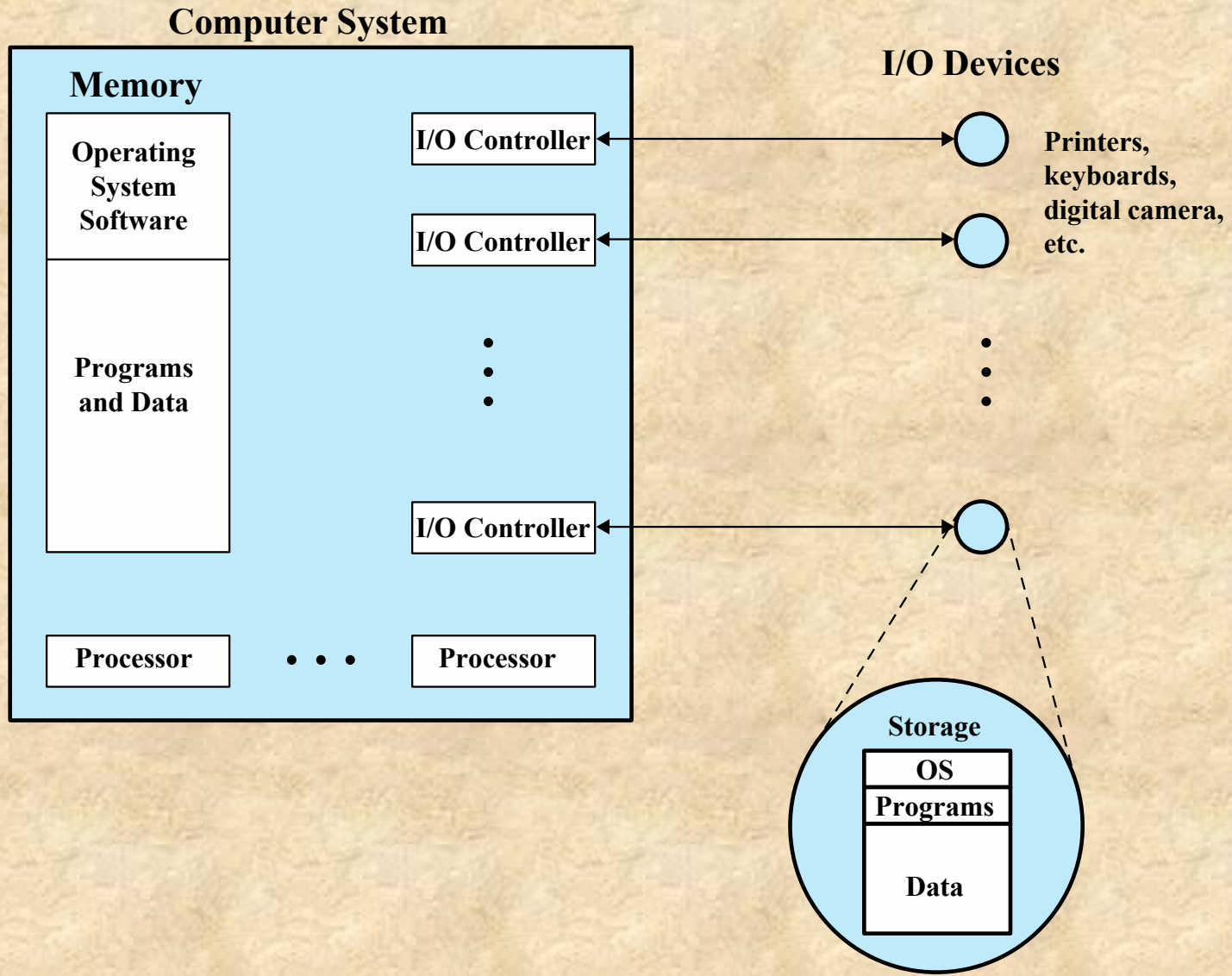


# The Operating System as Resource Manager

- The OS is responsible for controlling the use of a computer's resources, such as I/O, main and secondary memory, and processor execution time

# Operating System as Resource Manager

- Functions in the same way as ordinary computer software
- Program, or suite of programs, executed by the processor
- Frequently relinquishes control and must depend on the processor to allow it to regain control



**Figure 2.2 The Operating System as Resource Manager**



# Processor Point of View

- Processor executes instruction from the memory containing the monitor
- Executes the instructions in the user program until it encounters an ending or error condition
- “*Control is passed to a job*” means processor is fetching and executing instructions in a user program
- “*Control is returned to the monitor*” means that the processor is fetching and executing instructions from the monitor program

# Job Control Language (JCL)

Special type of programming  
language used to provide  
instructions to the monitor



What compiler to use



What data to use



# Desirable Hardware Features

## Memory protection

- While the user program is executing, it must not alter the memory area containing the monitor

## Timer

- Prevents a job from monopolizing the system

## Privileged instructions

- Can only be executed by the monitor

## Interrupts

- Gives OS more flexibility in controlling user programs

# Modes of Operation

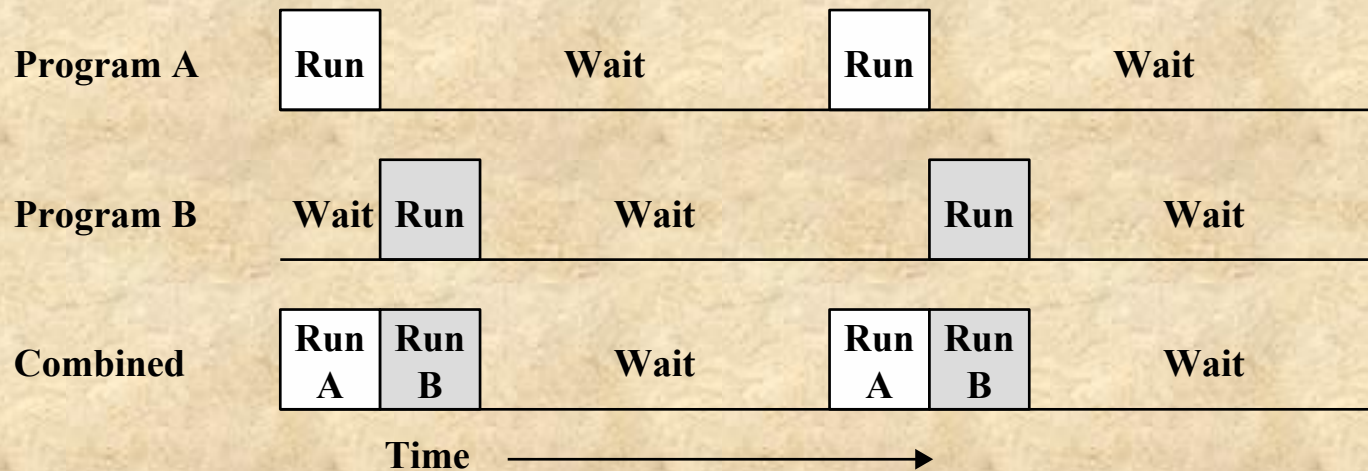
## User Mode

- User program executes in user mode
- Certain areas of memory are protected from user access
- Certain instructions may not be executed

## Kernel Mode

- Monitor executes in kernel mode
- Privileged instructions may be executed
- Protected areas of memory may be accessed

# Multiprogramming

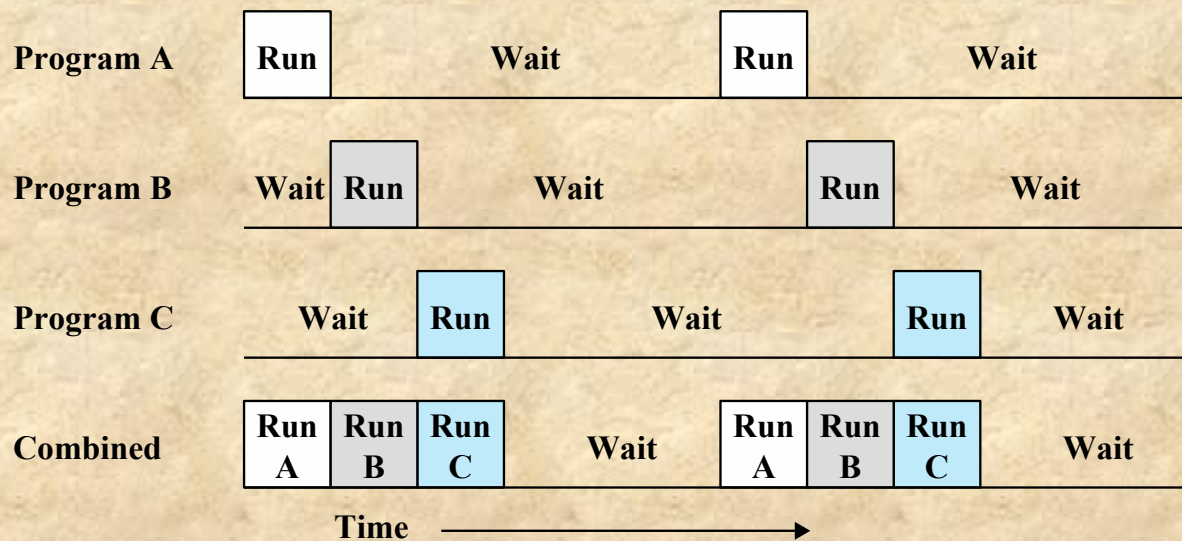


(b) Multiprogramming with two programs

- There must be enough memory to hold the OS (resident monitor) and one user program
- When one job needs to wait for I/O, the processor can switch to the other job, which is likely not waiting for I/O



# Multiprogramming



(c) Multiprogramming with three programs

- Also known as multitasking
- Memory is expanded to hold three, four, or more programs and switch among all of them

# Time-Sharing Systems

- Can be used to handle multiple interactive jobs
- Processor time is shared among multiple users
- Multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation

# Major Achievements

- Operating Systems are among the most complex pieces of software ever developed
- Major advances in development include:
  - Processes
  - Memory management
  - Information protection and security
  - Scheduling and resource management
  - System structure



# Process

- Fundamental to the structure of operating systems

A *process* can be defined as:

A program in execution

An instance of a running program

The entity that can be assigned to, and executed on, a processor

A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources

# Causes of Errors

## ■ Improper synchronization

- It is often the case that a routine must be suspended awaiting an event elsewhere in the system
- Improper design of the signaling mechanism can result in loss or duplication

## ■ Failed mutual exclusion

- More than one user or program attempts to make use of a shared resource at the same time
- There must be some sort of mutual exclusion mechanism that permits only one routine at a time to perform an update against the file

## ■ Nondeterminate program operation

- When programs share memory, and their execution is interleaved by the processor, they may interfere with each other by overwriting common memory areas in unpredictable ways
- The order in which programs are scheduled may affect the outcome of any particular program

## ■ Deadlocks

- It is possible for two or more programs to be hung up waiting for each other



# Components of a Process

- A process contains three components:
  - An executable program
  - The associated data needed by the program (variables, work space, buffers, etc.)
  - The execution context (or “process state”) of the program
- The execution context is essential:
  - It is the internal data by which the OS is able to supervise and control the process
  - Includes the contents of the various process registers
  - Includes information such as the priority of the process and whether the process is waiting for the completion of a particular I/O event



# Process Management

- The entire state of the process at any instant is contained in its context
- New features can be designed and incorporated into the OS by expanding the context to include any new information needed to support the feature

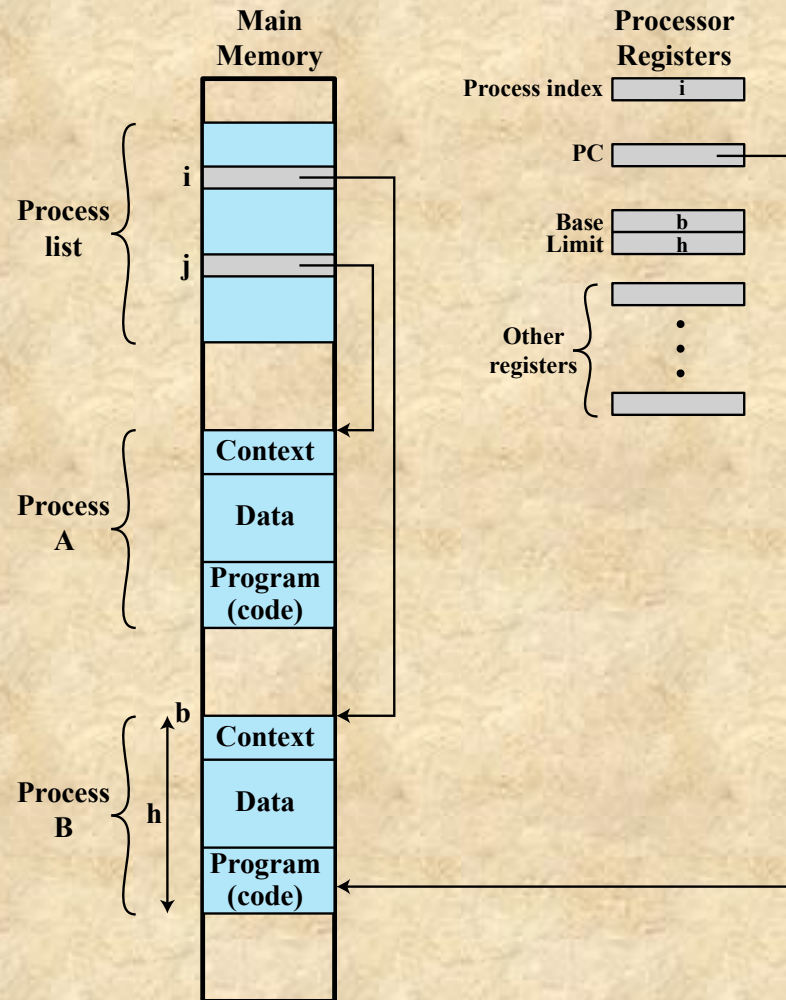
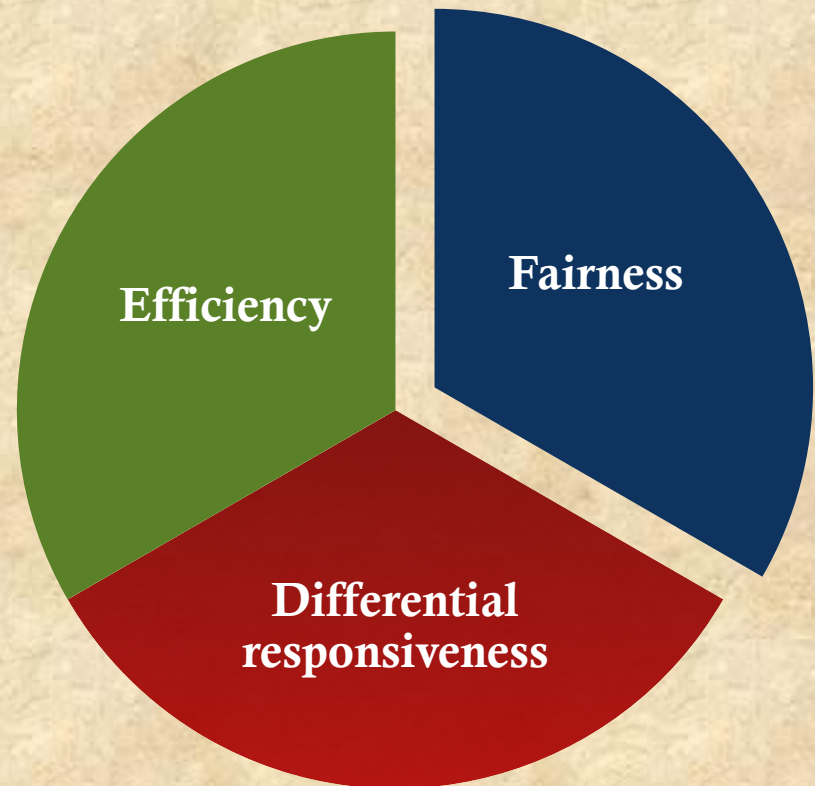
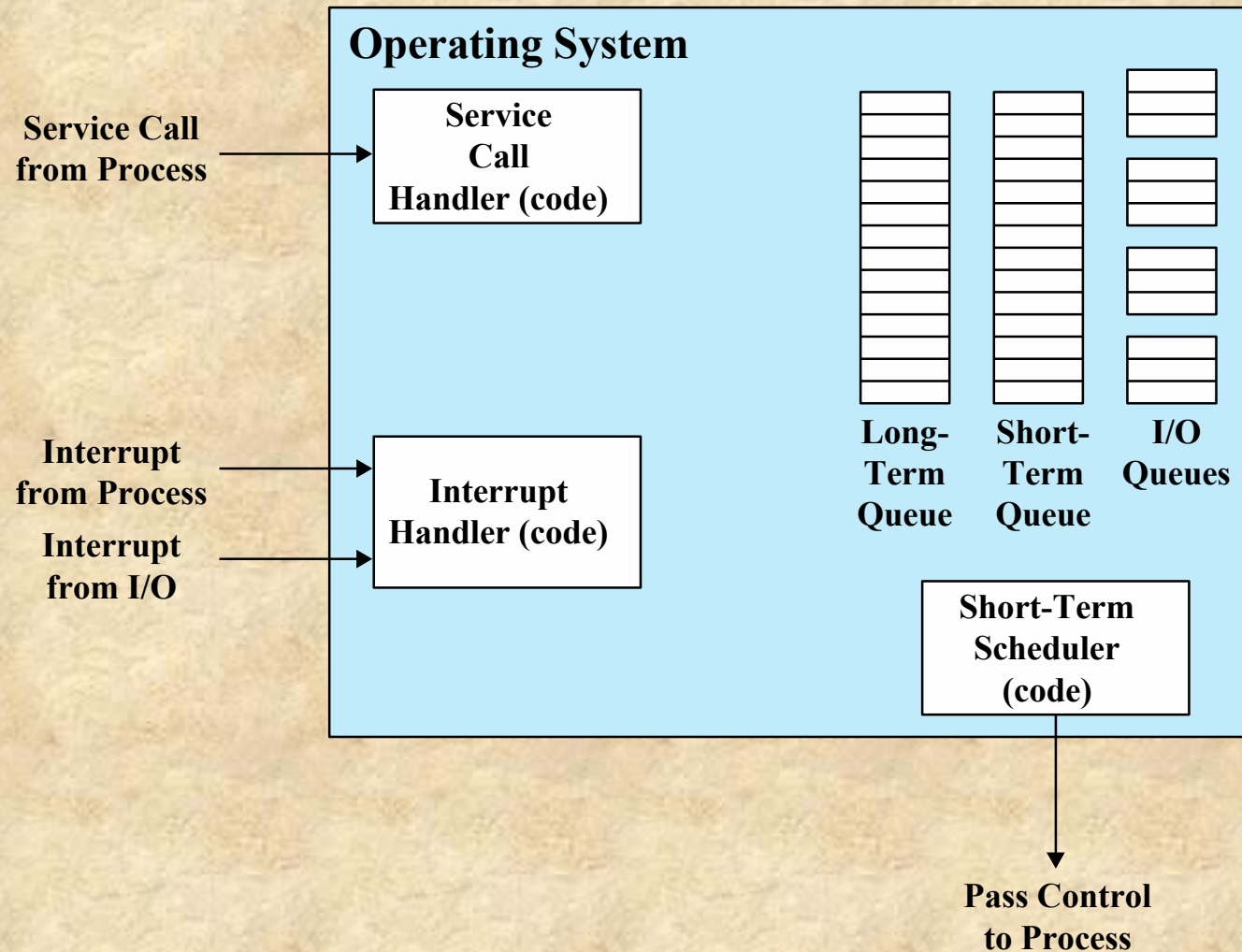


Figure 2.8 Typical Process Implementation

# Scheduling and Resource Management

- Key responsibility of an OS is managing resources
- Resource allocation policies must consider:





**Figure 2.11 Key Elements of an Operating System for Multiprogramming**



# Different Architectural Approaches

- Demands on operating systems require new ways of organizing the OS

Different approaches and design elements have been tried:

- Microkernel architecture
- Multithreading
- Symmetric multiprocessing
- Distributed operating systems
- Object-oriented design

# Multithreading

- Technique in which a process, executing an application, is divided into threads that can run concurrently

## Thread

Dispatchable unit of work

Includes a processor context and its own data area for a stack

Executes sequentially and is interruptible

## Process

A collection of one or more threads and associated system resources

By breaking a single application into multiple threads, a programmer has greater control over the modularity of the application and the timing of application-related events



# OS Design

## Distributed Operating System

- Provides the illusion of a single main memory space and a single secondary memory space plus other unified access facilities, such as a distributed file system
- State of the art for distributed operating systems lags that of uniprocessor and SMP operating systems

## Object-Oriented Design

- Lends discipline to the process of adding modular extensions to a small kernel
- Enables programmers to customize an operating system without disrupting system integrity
- Also eases the development of distributed tools and full-blown distributed operating systems



# Operating System Mechanisms

- A number of techniques can be incorporated into OS software to support fault tolerance:
  - Process isolation
  - Concurrency controls
  - Virtual machines
  - Checkpoints and rollbacks

# Symmetric Multiprocessor OS Considerations

- A multiprocessor OS must provide all the functionality of a multiprogramming system plus additional features to accommodate multiple processors
- Key design issues:

## Simultaneous concurrent processes or threads

Kernel routines need to be reentrant to allow several processors to execute the same kernel code simultaneously

## Scheduling

Any processor may perform scheduling, which complicates the task of enforcing a scheduling policy

## Synchronization

With multiple active processes having potential access to shared address spaces or shared I/O resources, care must be taken to provide effective synchronization

## Memory management

The reuse of physical pages is the biggest problem of concern

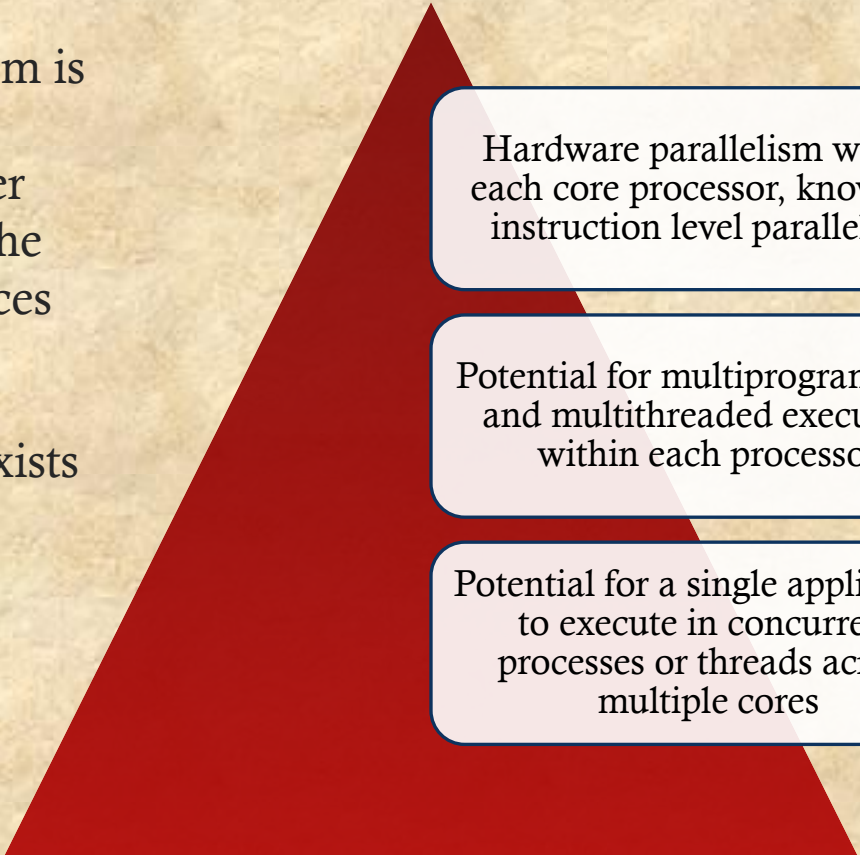
## Reliability and fault tolerance

The OS should provide graceful degradation in the face of processor failure



# Multicore OS Considerations

- The design challenge for a many-core multicore system is to efficiently harness the multicore processing power and intelligently manage the substantial on-chip resources efficiently
- Potential for parallelism exists at three levels:



Hardware parallelism within each core processor, known as instruction level parallelism

Potential for multiprogramming and multithreaded execution within each processor

Potential for a single application to execute in concurrent processes or threads across multiple cores



# OS Management of Application Execution

- Resources are made available to multiple applications
- The processor is switched among multiple applications so all will appear to be progressing
- The processor and I/O devices can be used efficiently

# Process Elements

- Two essential elements of a process are:

## Program code

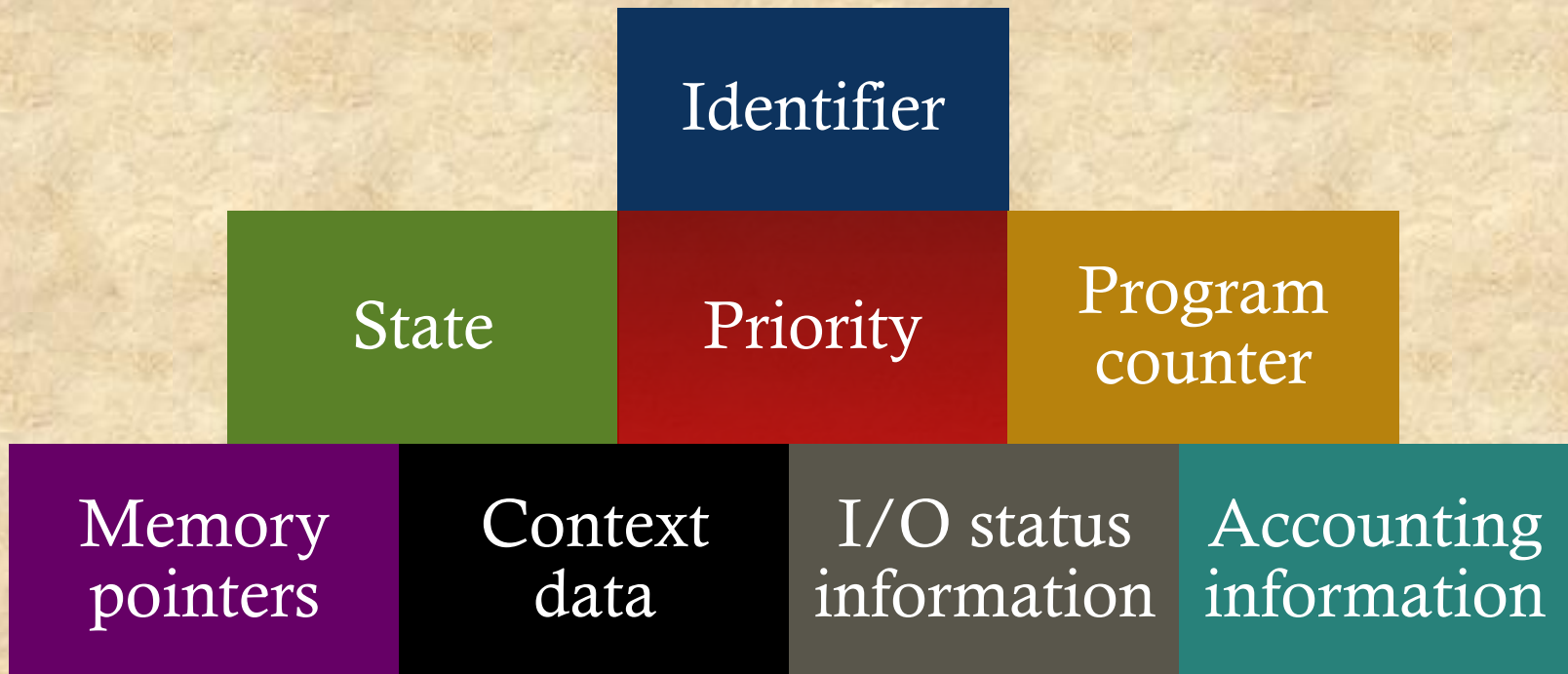
- which may be shared with other processes that are executing the same program

## A set of data associated with that code

- when the processor begins to execute the program code, we refer to this executing entity as a *process*

# Process Elements

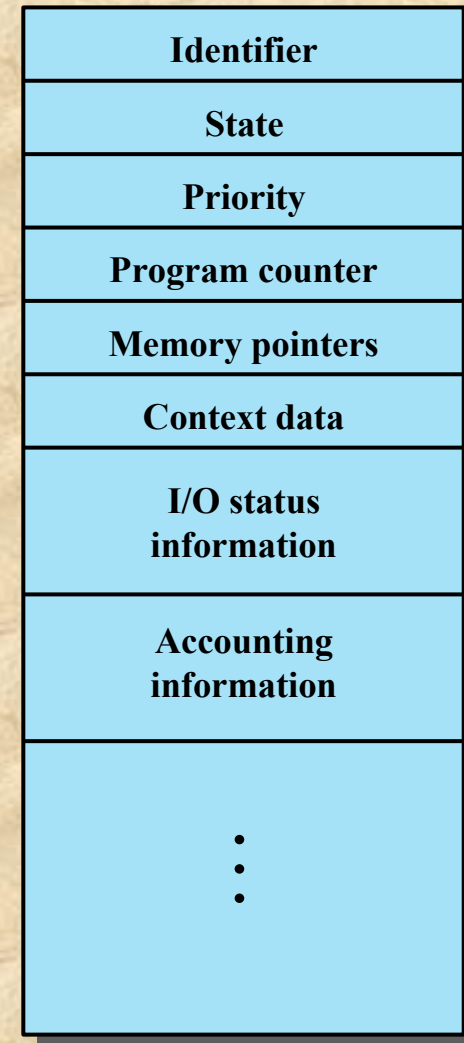
- While the program is executing, this process can be uniquely characterized by a number of elements, including:





# Process Control Block

- Contains the process elements
- It is possible to interrupt a running process and later resume execution as if the interruption had not occurred
- Created and managed by the operating system
- Key tool that allows support for multiple processes



**Figure 3.1 Simplified Process Control Block**

# Process States

## *Trace*

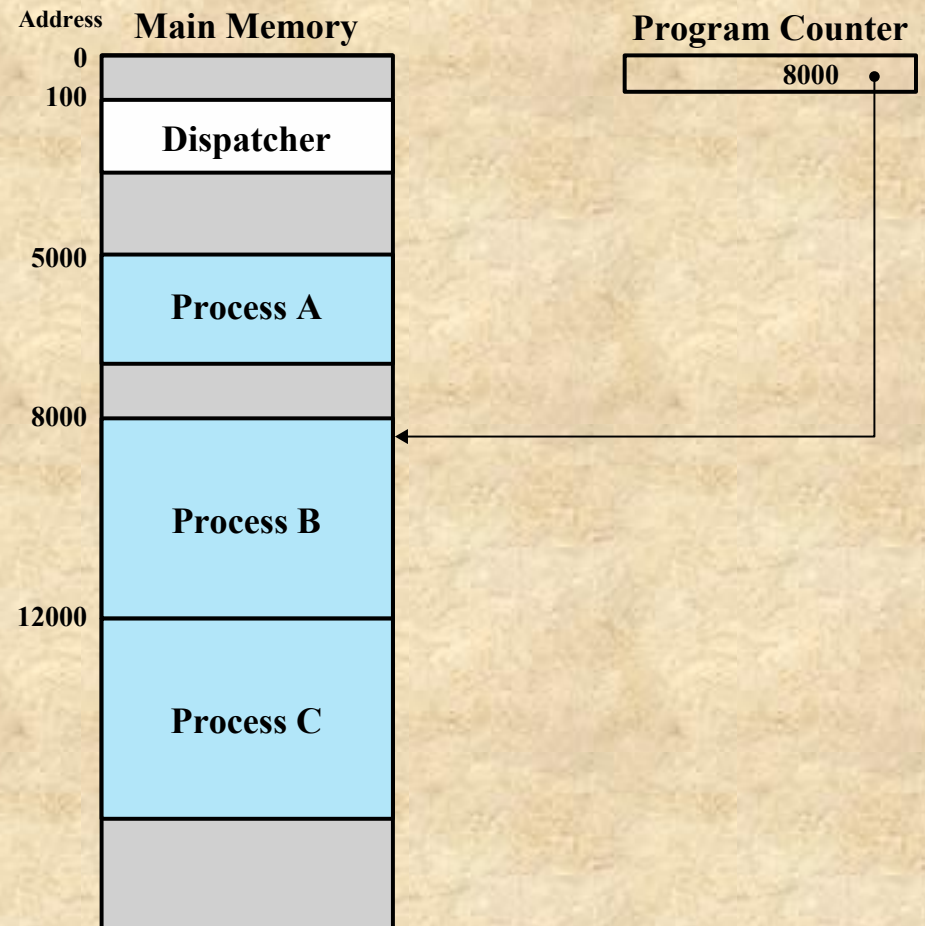
The behavior of an individual process by listing the sequence of instructions that execute for that process

The behavior of the processor can be characterized by showing how the traces of the various processes are interleaved

## *Dispatcher*

Small program that switches the processor from one process to another

# Process Execution



**Figure 3.2 Snapshot of Example Execution (Figure 3.4)  
at Instruction Cycle 13**



5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

**(a) Trace of Process A**

**(b) Trace of Process B**

**(c) Trace of Process C**

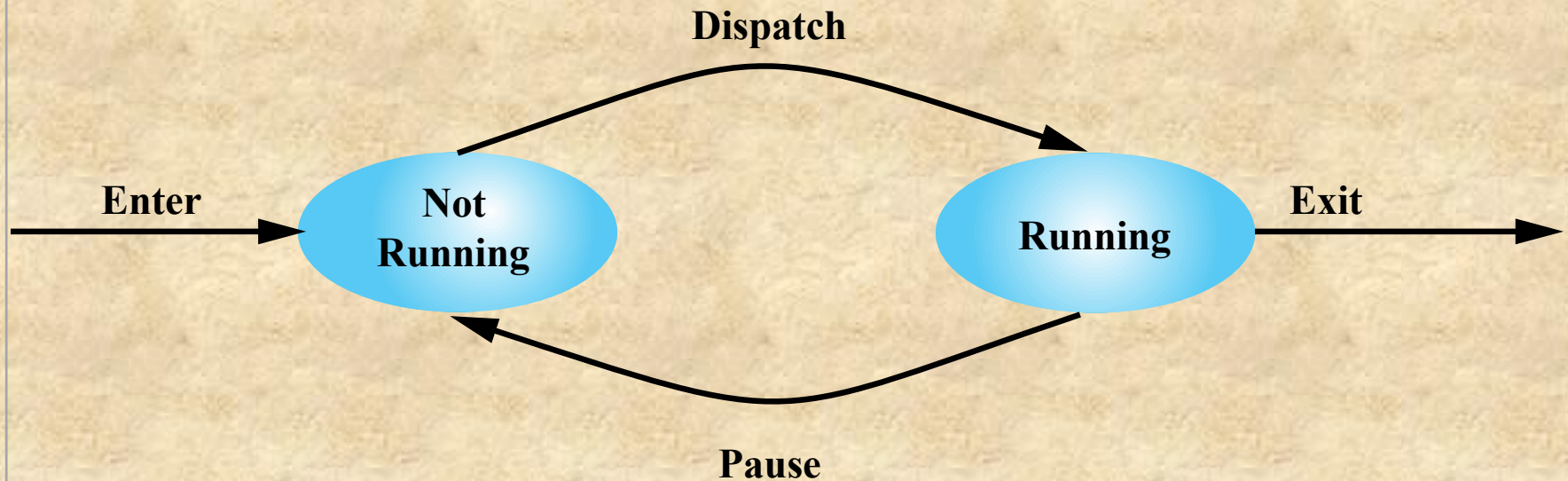
5000 = Starting address of program of Process A

8000 = Starting address of program of Process B

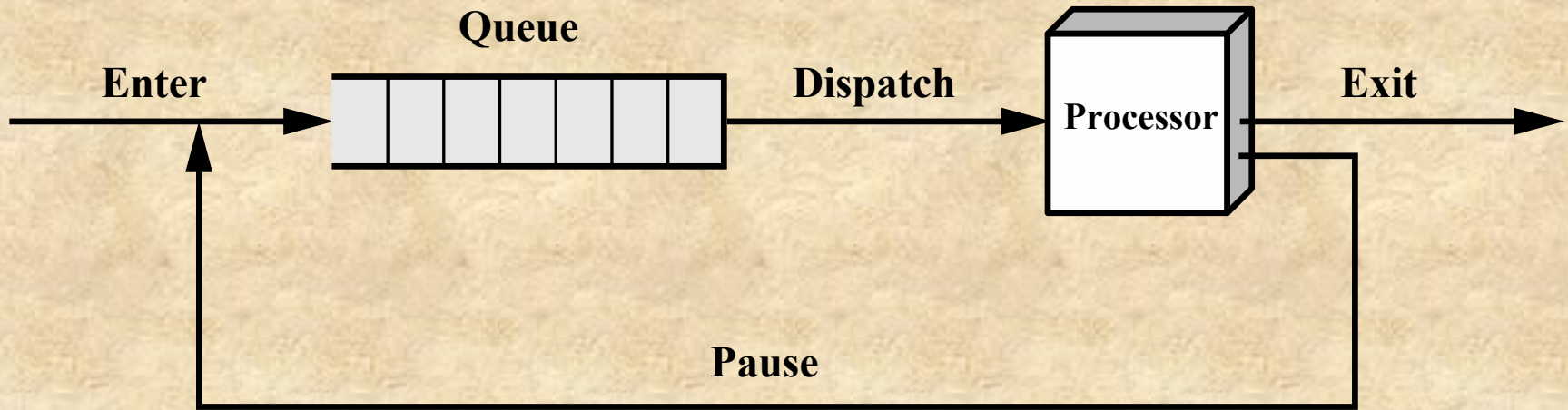
12000 = Starting address of program of Process C

**Figure 3.3 Traces of Processes of Figure 3.2**

# Two-State Process Model



(a) State transition diagram



**(b) Queuing diagram**

**Figure 3.5 Two-State Process Model**



## Table 3.1 Reasons for Process Creation

New batch job	The OS is provided with a batch job control stream, usually on tape or disk. When the OS is prepared to take on new work, it will read the next sequence of job control commands.
Interactive logon	A user at a terminal logs on to the system.
Created by OS to provide a service	The OS can create a process to perform a function on behalf of a user program, without the user having to wait (e.g., a process to control printing).
Spawned by existing process	For purposes of modularity or to exploit parallelism, a user program can dictate the creation of a number of processes.

# Process Creation

## *Process spawning*

- When the OS creates a process at the explicit request of another process

## *Parent process*

- Is the original, creating, process

## *Child process*

- Is the new process

# Five-State Process Model

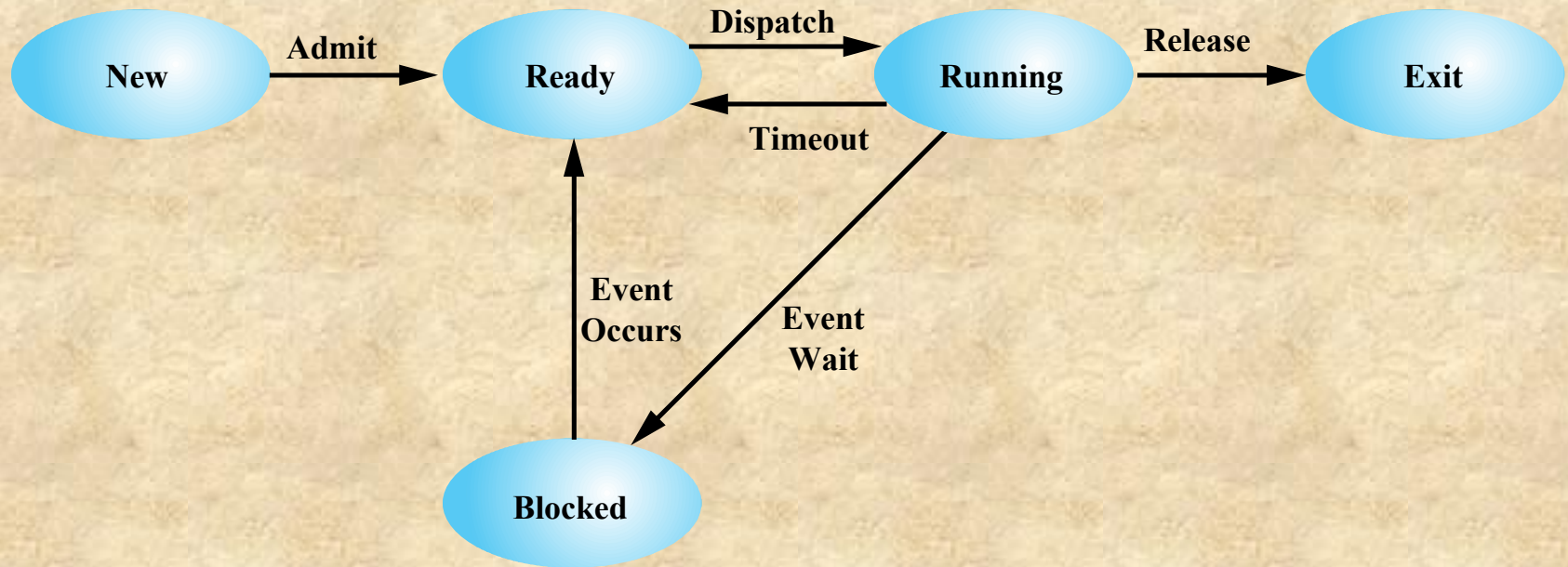


Figure 3.6 Five-State Process Model



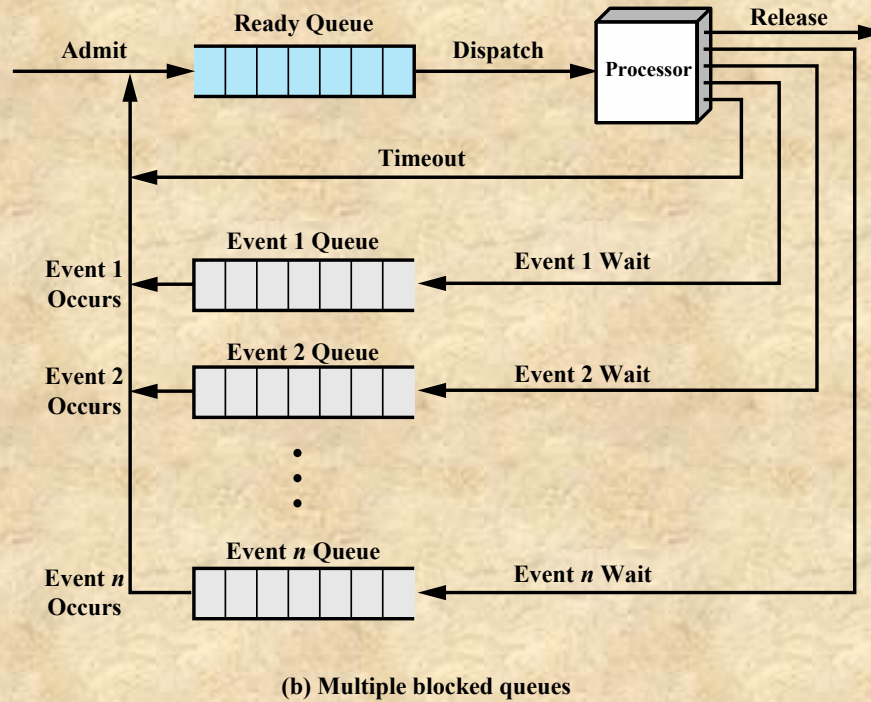
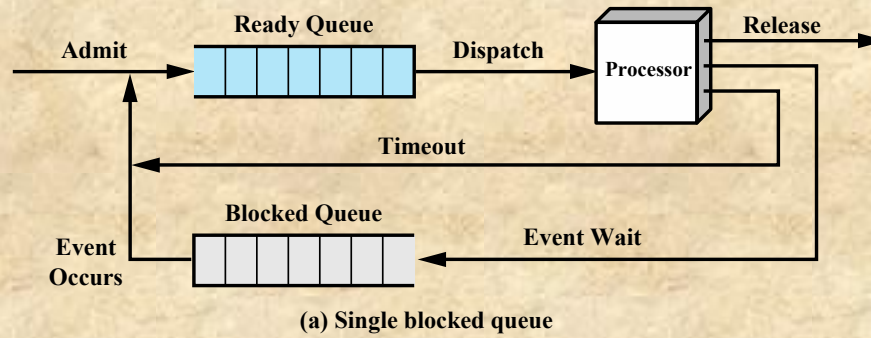


Figure 3.8 Queuing Model for Figure 3.6

# Process Control Structures

To manage  
and  
control a  
process the  
OS must  
know:

- Where the process is located
- The attributes of the process that are necessary for its management

# Process Control Structures

## Process Location

- A process must include a program or set of programs to be executed
- A process will consist of at least sufficient memory to hold the programs and data of that process
- The execution of a program typically involves a stack that is used to keep track of procedure calls and parameter passing between procedures

## Process Attributes

- Each process has associated with it a number of attributes that are used by the OS for process control
- The collection of program, data, stack, and attributes is referred to as the process image
- Process image location will depend on the memory management scheme being used



---

# Table 3.4

## Typical Elements of a Process Image

### **User Data**

The modifiable part of the user space. May include program data, a user stack area, and programs that may be modified.

### **User Program**

The program to be executed.

### **Stack**

Each process has one or more last-in-first-out (LIFO) stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls.

### **Process Control Block**

Data needed by the OS to control the process (see Table 3.5).

## Process Identification

### Identifiers

Numeric identifiers that may be stored with the process control block include

- Identifier of this process
- Identifier of the process that created this process (parent process)
- User identifier

## Processor State Information

### User-Visible Registers

A user-visible register is one that may be referenced by means of the machine language that the processor executes while in user mode. Typically, there are from 8 to 32 of these registers, although some RISC implementations have over 100.

### Control and Status Registers

These are a variety of processor registers that are employed to control the operation of the processor. These include

- **Program counter:** Contains the address of the next instruction to be fetched
- **Condition codes:** Result of the most recent arithmetic or logical operation (e.g., sign, zero, carry, equal, overflow)
- **Status information:** Includes interrupt enabled/disabled flags, execution mode

### Stack Pointers

Each process has one or more last-in-first-out (LIFO) system stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls. The stack pointer points to the top of the stack.

Table 3.5  
Typical  
Elements  
of a  
Process  
Control  
Block  
(page 1 of 2)

(Table is located  
on page 125 in the  
textbook)

## Process Control Information

### Scheduling and State Information

This is information that is needed by the operating system to perform its scheduling function. Typical items of information:

- **Process state:** Defines the readiness of the process to be scheduled for execution (e.g., running, ready, waiting, halted).
- **Priority:** One or more fields may be used to describe the scheduling priority of the process. In some systems, several values are required (e.g., default, current, highest-allowable)
- **Scheduling-related information:** This will depend on the scheduling algorithm used. Examples are the amount of time that the process has been waiting and the amount of time that the process executed the last time it was running.
- **Event:** Identity of event the process is awaiting before it can be resumed.

### Data Structuring

A process may be linked to other process in a queue, ring, or some other structure. For example, all processes in a waiting state for a particular priority level may be linked in a queue. A process may exhibit a parent-child (creator-created) relationship with another process. The process control block may contain pointers to other processes to support these structures.

### Interprocess Communication

Various flags, signals, and messages may be associated with communication between two independent processes. Some or all of this information may be maintained in the process control block.

### Process Privileges

Processes are granted privileges in terms of the memory that may be accessed and the types of instructions that may be executed. In addition, privileges may apply to the use of system utilities and services.

### Memory Management

This section may include pointers to segment and/or page tables that describe the virtual memory assigned to this process.

### Resource Ownership and Utilization

Resources controlled by the process may be indicated, such as opened files. A history of utilization of the processor or other resources may also be included; this information may be needed by the scheduler.

## Table 3.5 Typical Elements of a Process Control Block (page 2 of 2)

(Table is located  
on page 125 in the textbook)

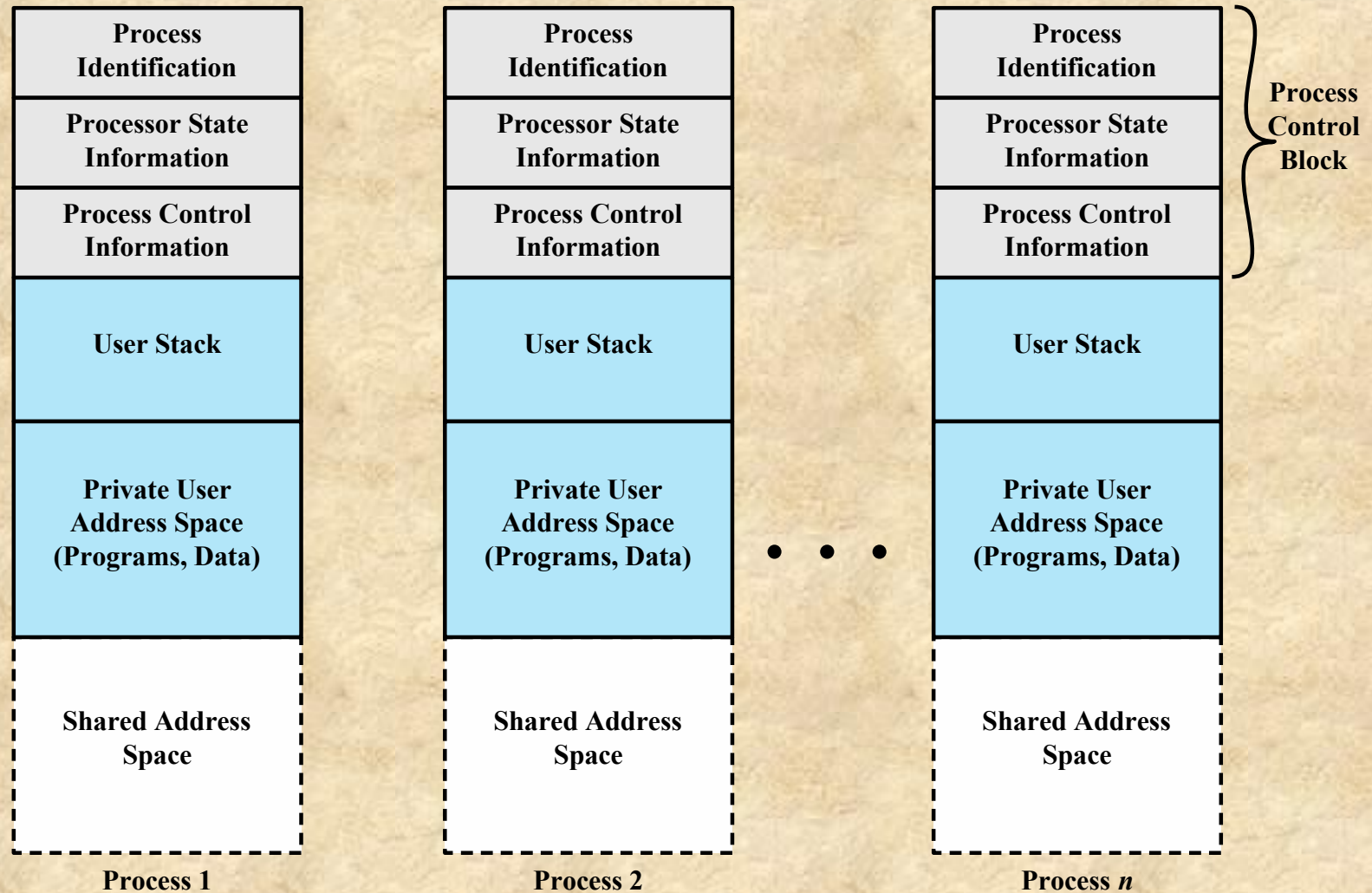


# Process Identification

- Each process is assigned a unique numeric identifier
  - Otherwise there must be a mapping that allows the OS to locate the appropriate tables based on the process identifier
- Many of the tables controlled by the OS may use process identifiers to cross-reference process tables
- Memory tables may be organized to provide a map of main memory with an indication of which process is assigned to each region
  - Similar references will appear in I/O and file tables
- When processes communicate with one another, the process identifier informs the OS of the destination of a particular communication
- When processes are allowed to create other processes, identifiers indicate the parent and descendents of each process

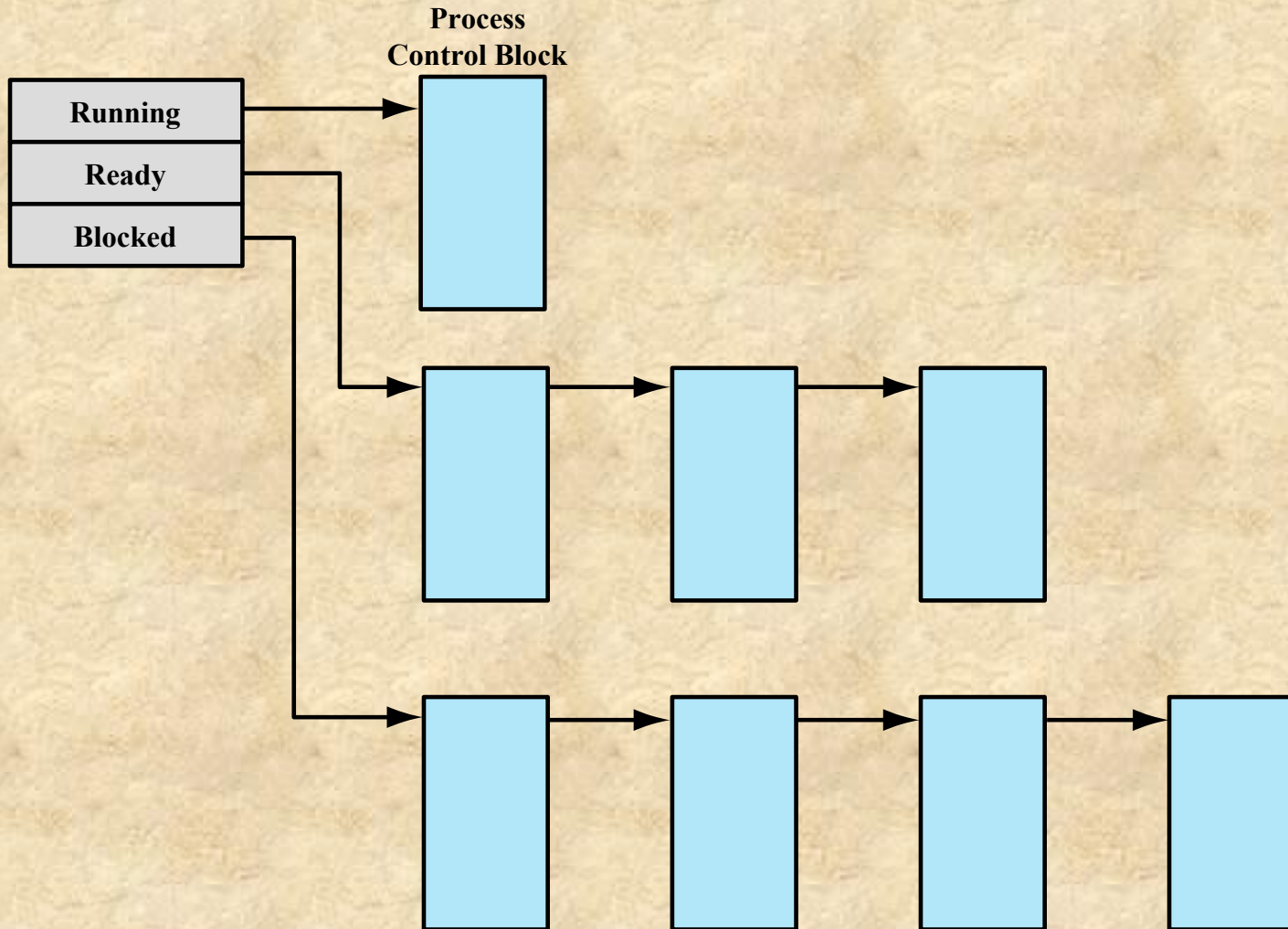
# Process Control Information

- The additional information needed by the OS to control and coordinate the various active processes



**Figure 3.13 User Processes in Virtual Memory**





**Figure 3.14 Process List Structures**

# Role of the Process Control Block

- The most important data structure in an OS
  - Contains all of the information about a process that is needed by the OS
  - Blocks are read and/or modified by virtually every module in the OS
  - Defines the state of the OS
- Difficulty is not access, but protection
  - A bug in a single routine could damage process control blocks, which could destroy the system's ability to manage the affected processes
  - A design change in the structure or semantics of the process control block could affect a number of modules in the OS