

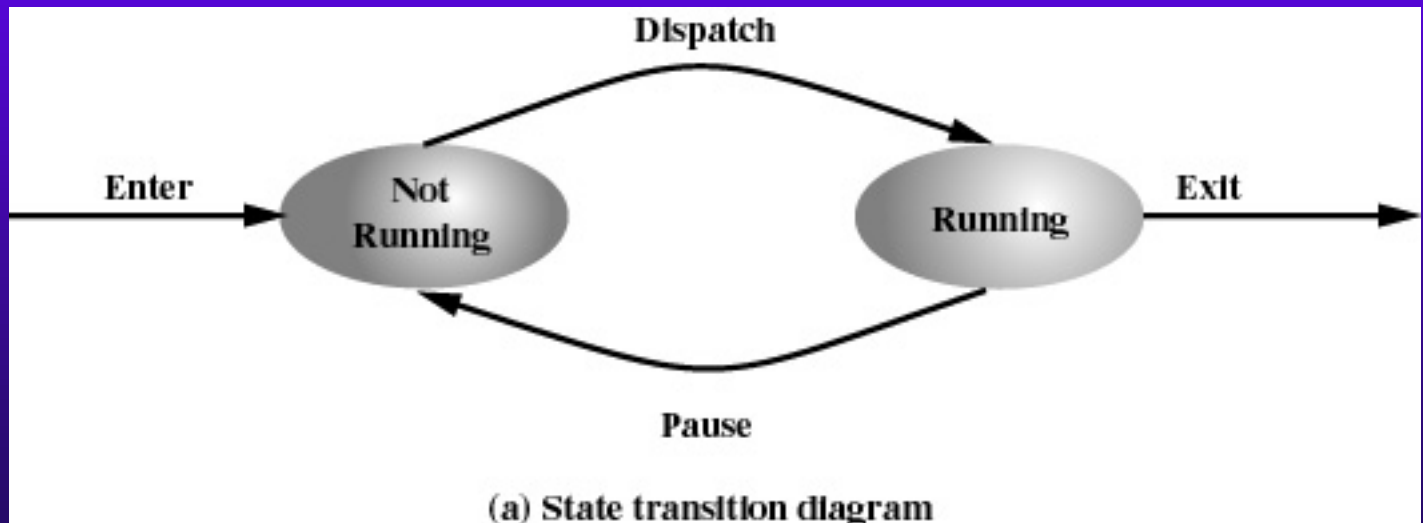


Processes

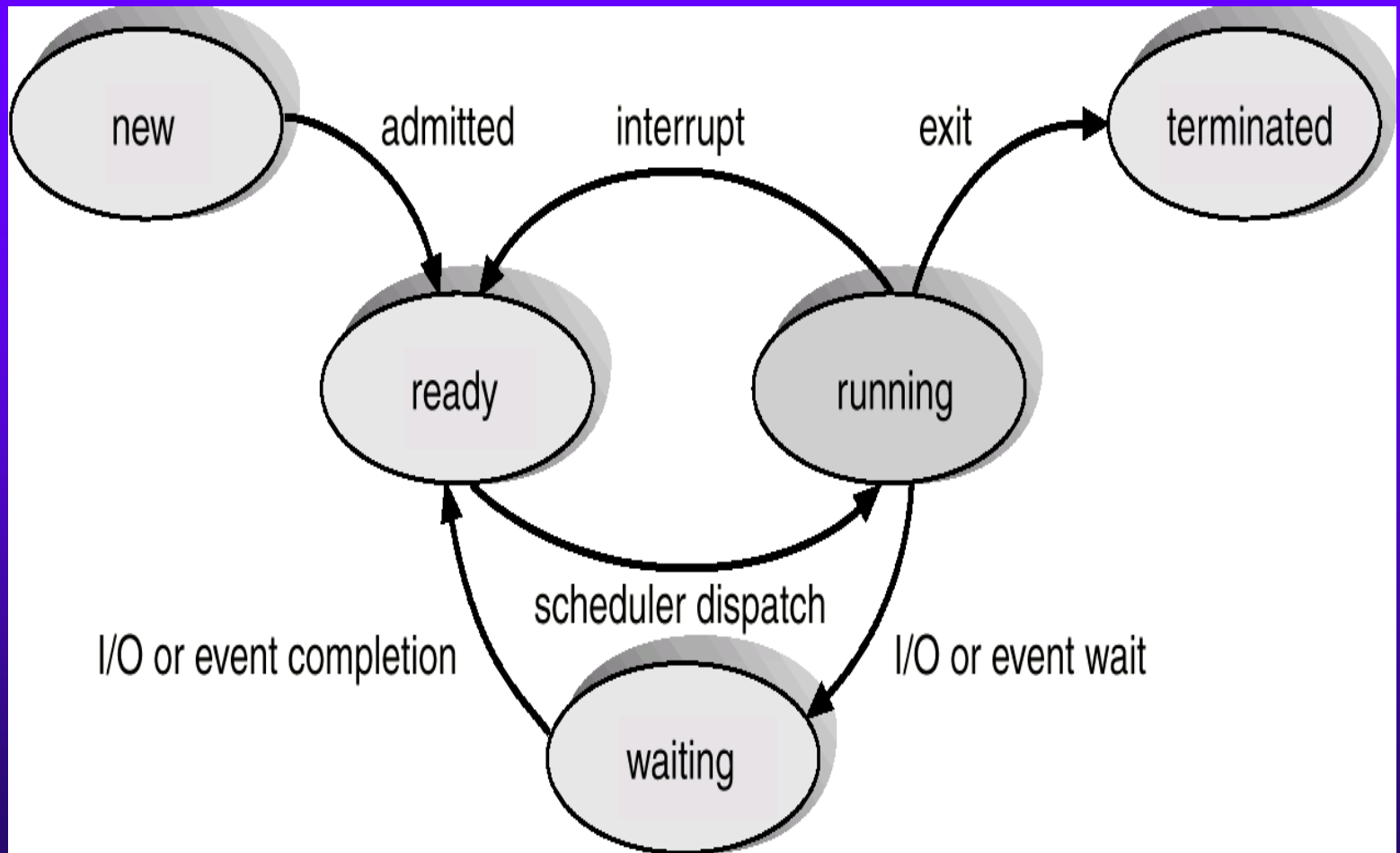
27/01/2004

Two state process model

- ◆ Process may in any of the 2 states
 - Running
 - Not running

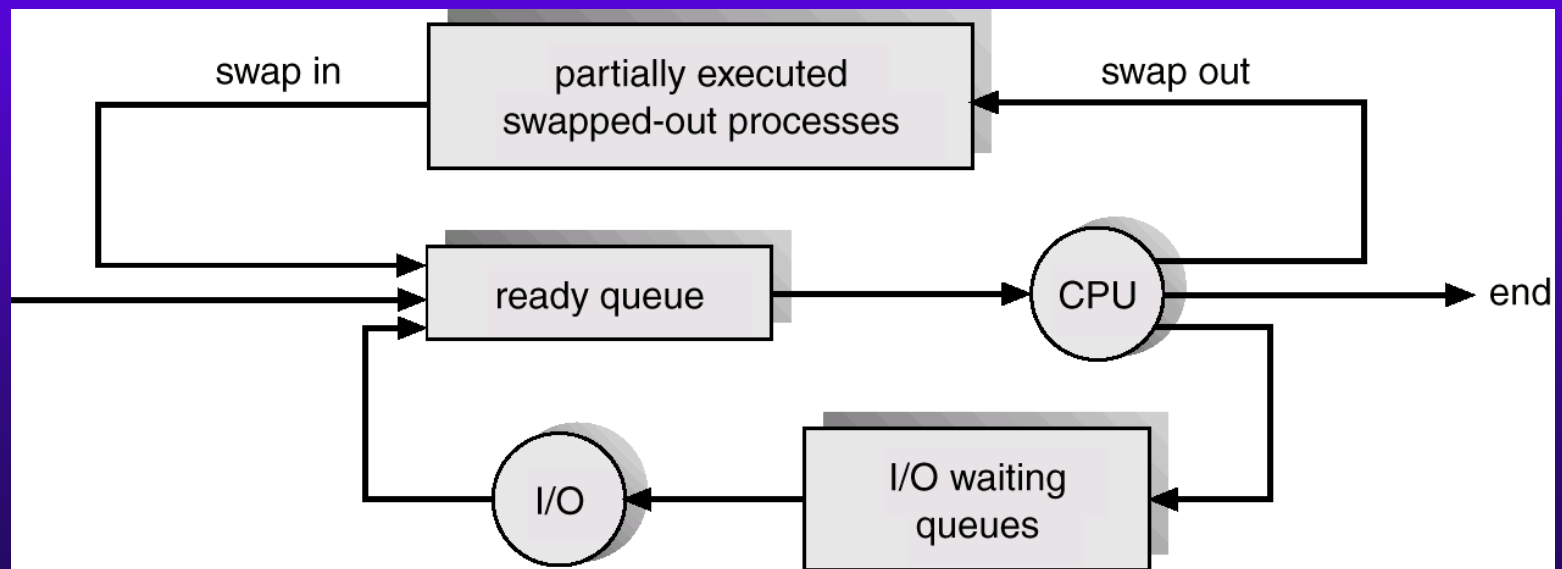


Process states



Schedulers

- Long-term scheduler (or job scheduler) – selects which processes should be brought into the ready queue.
- Short-term scheduler (or CPU scheduler) – selects which process should be executed next and allocates CPU.
- Mid term scheduler





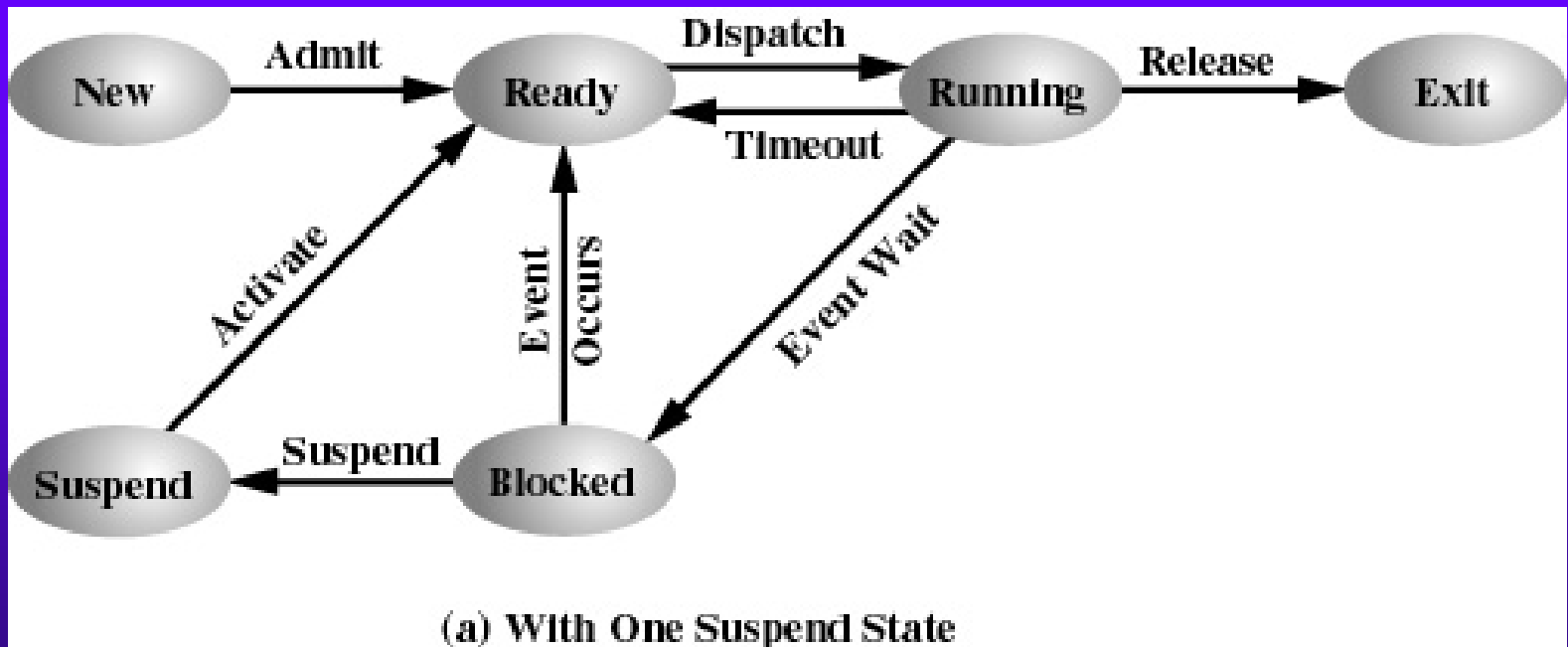
- Short-term scheduler is invoked very frequently (milliseconds) \Rightarrow (must be fast).
- Long-term scheduler is invoked very infrequently (seconds, minutes) \Rightarrow (may be slow).
- The long-term scheduler controls the *degree of multiprogramming*.
- Processes can be described as either:
 - *I/O-bound process* – spends more time doing I/O than computations, many short CPU bursts.
 - *CPU-bound process* – spends more time doing computations; few very long CPU bursts.
- The mid-term scheduler reduces the *degree of multiprogramming*.



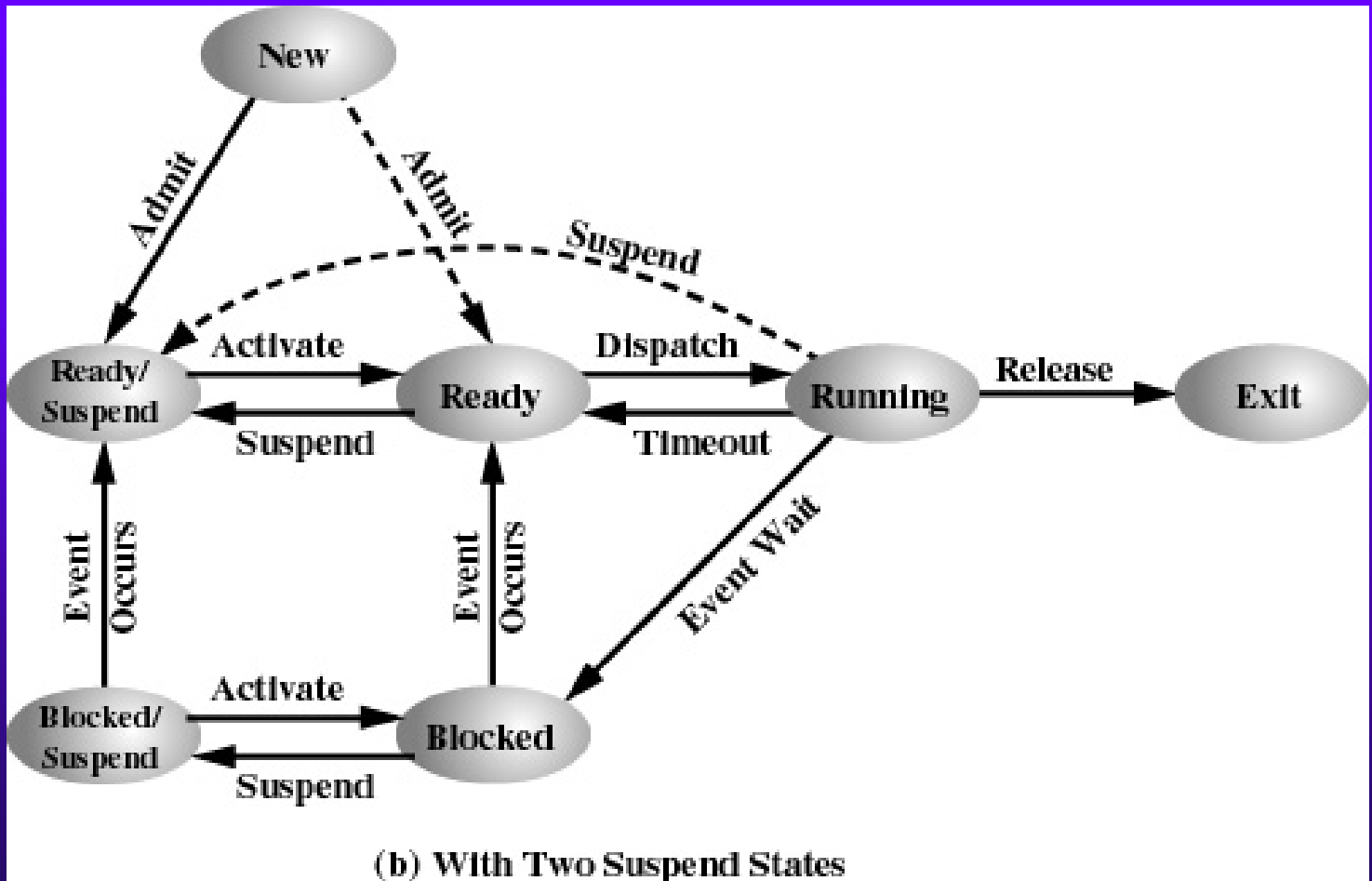
Suspended Processes

- ◆ Processor is faster than I/O so all processes could be waiting for I/O
- ◆ Swap these processes to disk to free up more memory
- ◆ Blocked state becomes suspend state when swapped to disk
- ◆ Two new states
 - Blocked, suspend
 - Ready, suspend

One Suspend State



Two Suspend State





Reasons for Process Suspension

Swapping

The operating system needs to release sufficient main memory to bring in a process that is ready to execute.

Other OS reason

The operating system may suspend a background or utility process or a process that is suspected of causing a problem.

Interactive user request

A user may wish to suspend execution of a program for purposes of debugging or in connection with the use of a resource.

Timing

A process may be executed periodically (e.g., an accounting or system monitoring process) and may be suspended while waiting for the next time interval.

Parent process request

A parent process may wish to suspend execution of a descendent to examine or modify the suspended process, or to coordinate the activity of various descendents.



- ◆ Once if the process leaves from new, it will not come back.
- ◆ Swapped ready to Ready
 - Reasons
 - mid term scheduler swaps other processes
 - Processes released memory previously allocated to them
 - Processes terminated releasing all its memory.
 - Amount of memory allotted to various processes released based on changing system conditions.
- ◆ Block to Swapped block
 - Not all OS does this.
 - A blocked process can not execute anyway, so placing it in memory appears to waste that resource.



- ◆ Swapped blocked to Blocked
 - This allows the process to execute sooner, once the blocked condition satisfied.
- ◆ Ready to Swapped ready OR Blocked to Swapped blocked
 - Mid term scheduler swaps out processes to release memory resources for all the processes.
- ◆ Swapped blocked to Swapped ready, Blocked to Ready
 - When the event occurs the process moves to the next state.
- ◆ Running to Blocked
 - I/O operations, communication with other processes, waiting on system conditions are controlled by factors external to the process.
- ◆ Any state to Termination
 - Possible but no guarantee about the result.
- ◆ Running to Ready --- time slice expire.

Process Control Block(PCB)

pointer	process state
process number	
program counter	
registers	
memory limits	
list of open files	
• • •	

Process Attributes

- ◆ Divided in to 3 parts
 - Process Identification
 - Process state information
 - Process Control Information
- ◆ 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. 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

◆ Processor State Information

– 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.

◆ 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



◆ Process Control Information

– 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.



◆ Process Control Information

– Inter process 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.



◆ Process Control Information

– 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.

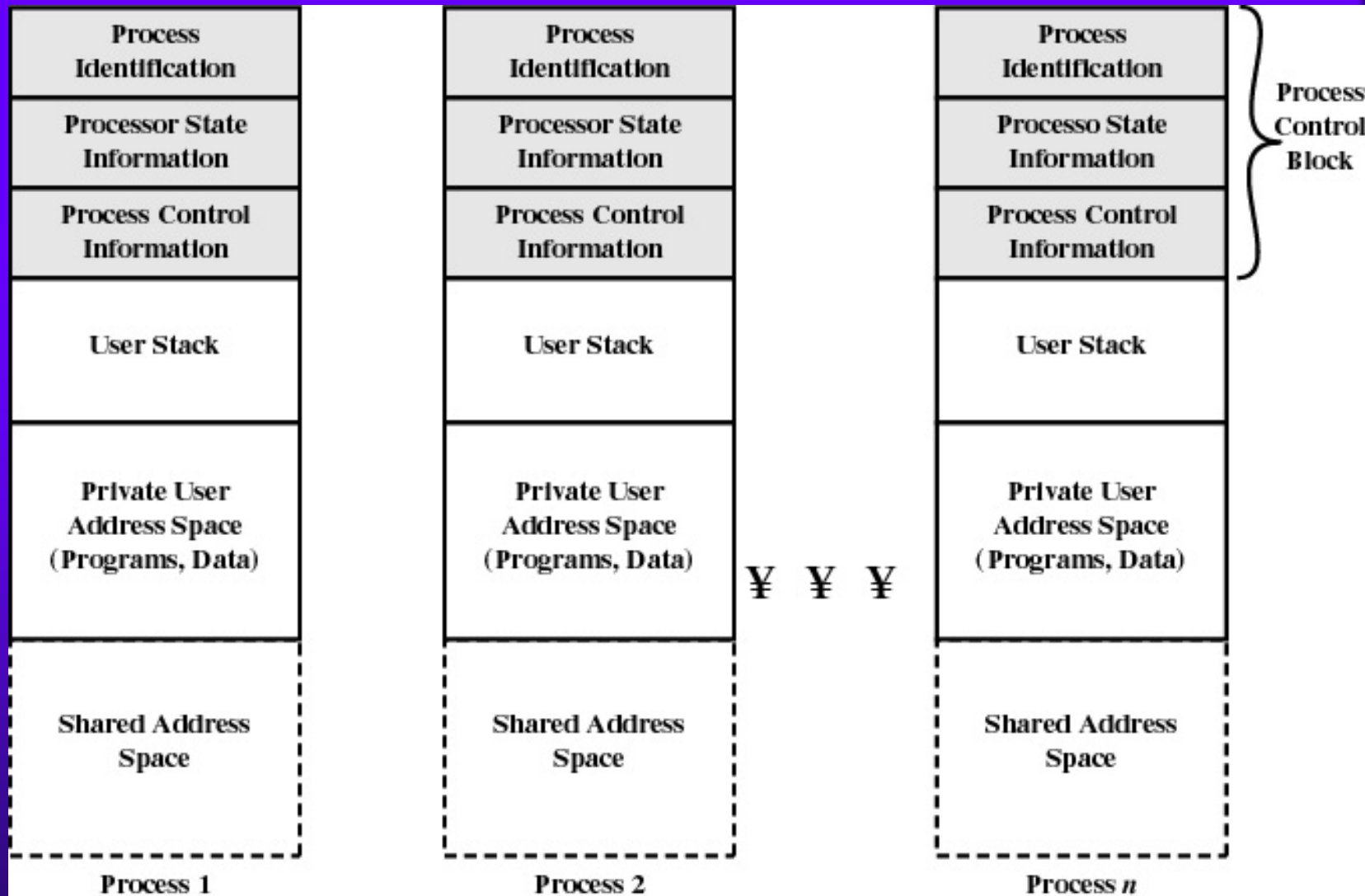


Figure 3.12 User Processes in Virtual Memory



Process state information

- ◆ Contents of processor registers
 - User-visible registers
 - Control and status registers
 - Stack pointers
- ◆ Program status word (PSW)
 - contains status information
 - Example: the EFLAGS register on Pentium machine.

Mode of execution

- ◆ User mode
 - Less-privileged mode
 - User programs typically execute in this mode
- ◆ System mode, control mode, or kernel mode
 - More-privileged mode
 - Kernel of the operating system
- ◆ Done by using hardware bit (0/1).



Process Creation

- ◆ Assign a unique process identifier
- ◆ Allocate space for the process
- ◆ Initialize process control block
- ◆ Set up appropriate linkages
 - Ex: add new process to linked list used for scheduling queue
- ◆ Create or expand other data structures
 - Ex: maintain an accounting file

When to Switch a Process

- ◆ Interrupt – Event that is external to & independent of currently running process.
 - Clock interrupt
 - process has executed for the maximum allowable time slice
 - I/O interrupt
 - Memory fault
 - memory address is in virtual memory so it must be brought into main memory
- ◆ Trap – Error or exception generated by the currently running process.
 - error occurred
 - may cause process to be moved to Exit state
- ◆ Supervisor call
 - such as file open



Mode switching

- ◆ Save the context of the current program being executed.
- ◆ Sets the program counter to the starting of the interrupt handler program.
- ◆ It switches from user mode to kernel mode to start the execution of interrupt routine.
- ◆ Switches between User mode & Kernel mode.

Process switching

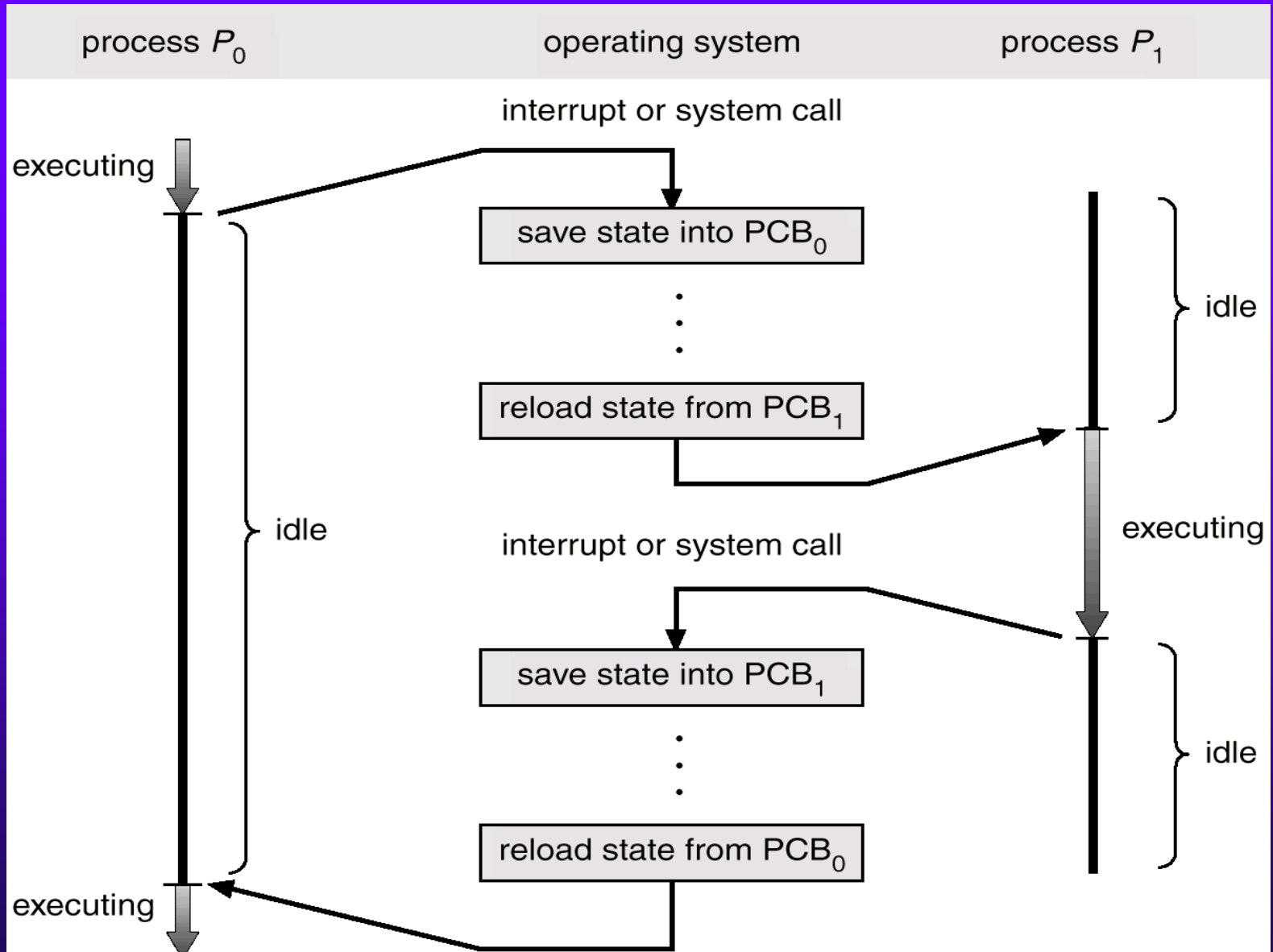
- ◆ Save context of processor including program counter and other registers
- ◆ Update the process control block of the process that is currently running
- ◆ Move process control block to appropriate queue - ready, blocked
- ◆ Select another process for execution
- ◆ Update the process control block of the process selected
- ◆ Update memory-management data structures
- ◆ Restore the program counter & Register values of new process.



Context switching

- When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process.
- Context-switch time is overhead; the system does no useful work while switching.
- Time dependent on hardware support.

CPU switch from process₀ to Process₁





Execution of the Operating System

◆ Non Process Kernel

- OS has its own region of memory, stack for controlling procedure call and return. When currently running process interrupts, the mode context of this process is saved.
- In this case concept of process is considered to apply only to user program. OS code is executed as a separate entity in privileged mode

◆ Execution Within User Processes

- Operating system software within context of a user process (code and data are in the shared address space and are shared by all user processes).
- Process executes in privileged mode when executing operating system code

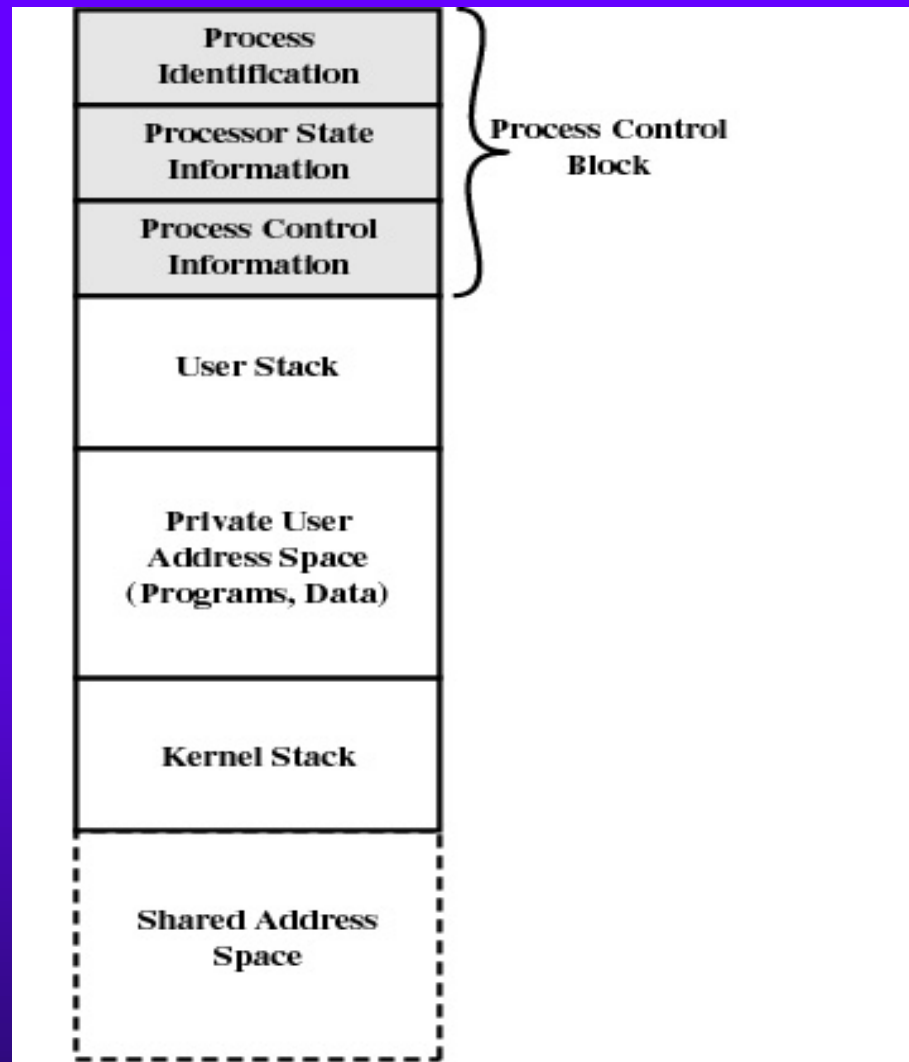
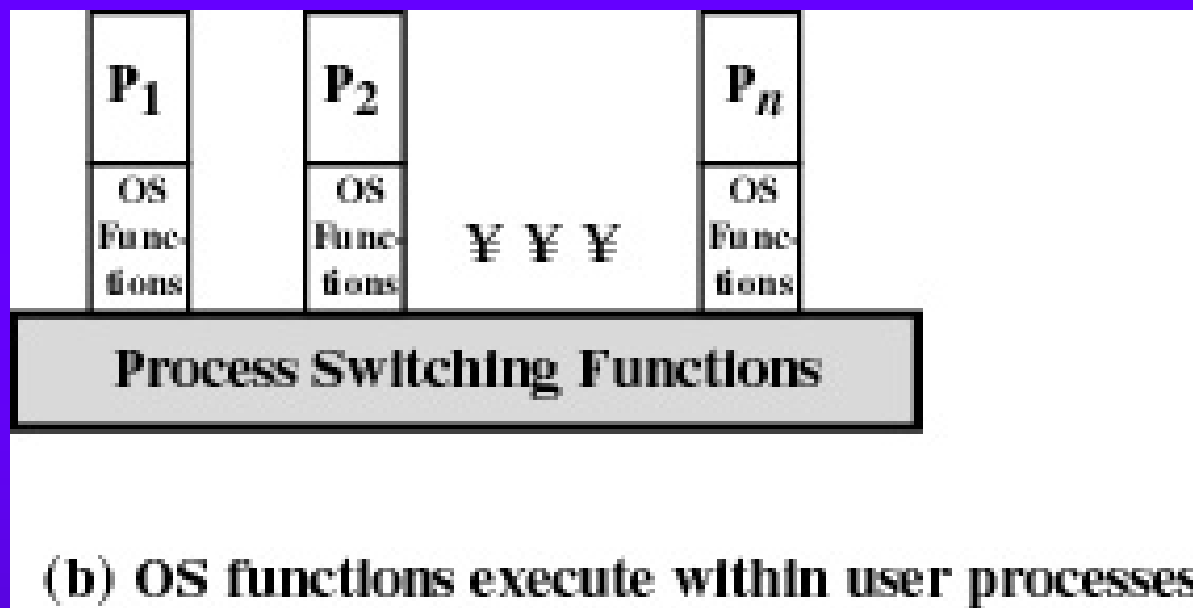


Figure 3.15 Process Image: Operating System Executes Within User Space



- ◆ **Process-Based Operating System**
 - major kernel functions are separate processes
 - Provides modularity with clean interfaces.
 - Non critical OS functions are conveniently implemented as separate process.
 - Useful in multi-processor or multi-computer environment