COP 4600 – Summer 2014

Introduction To Operating Systems

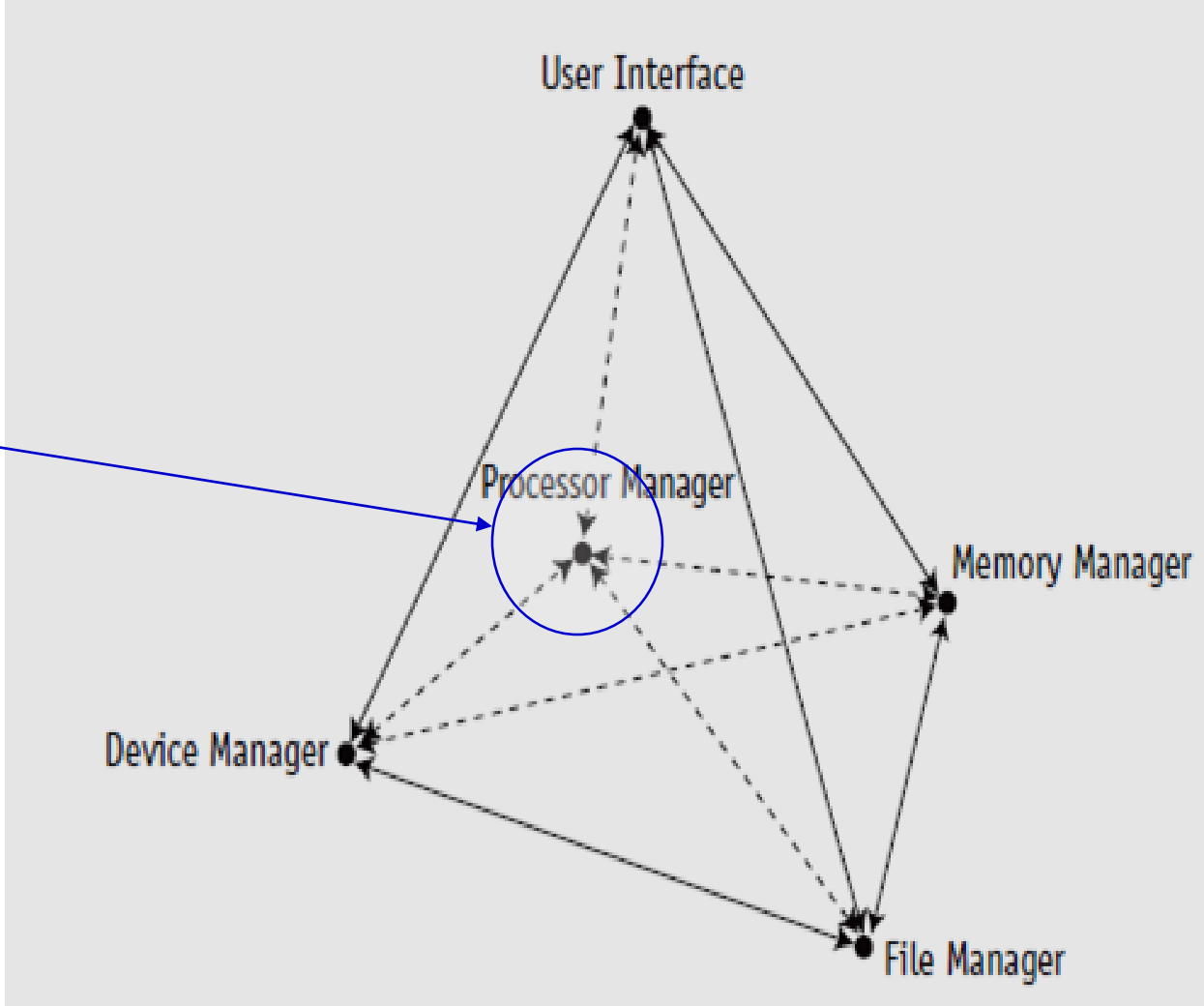
Chapter 3 – Process Description And Control

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cop4600/sum2014

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

Processor Management

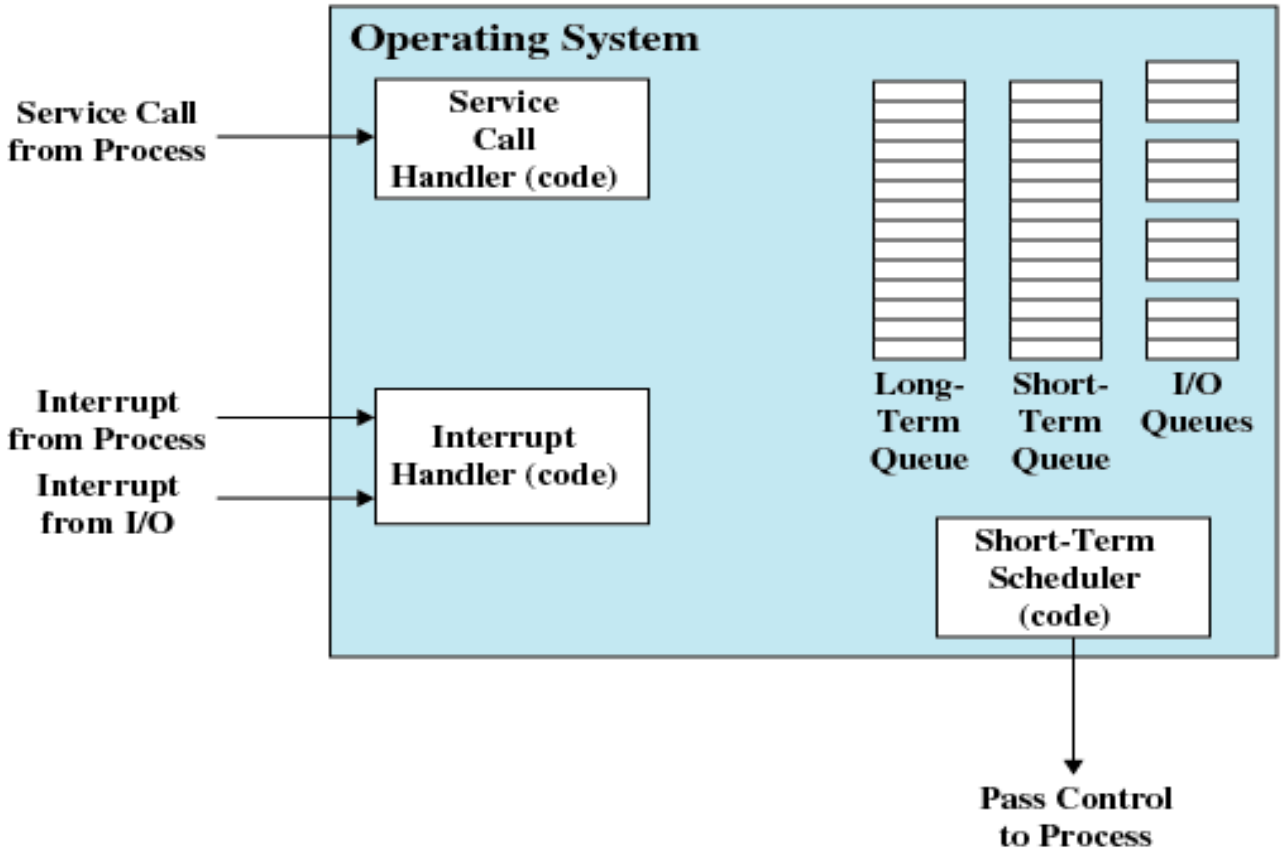
We'll begin our in-depth look at operating systems by focusing on the processor management component.



Scheduling and Resource Management

- Fairness
  - Give equal and fair access to resources
- Differential responsiveness
  - Discriminate among different classes of jobs
- Efficiency
  - Maximize throughput, minimize response time, and accommodate as many uses as possible

Key Elements of an Operating System



Elements of an Operating System for Multiprogramming

Interrupts

- Interrupts the normal sequencing of the processor.
- Most I/O devices are slower than the processor
  - Processor must pause to wait for device

Classes of Interrupts

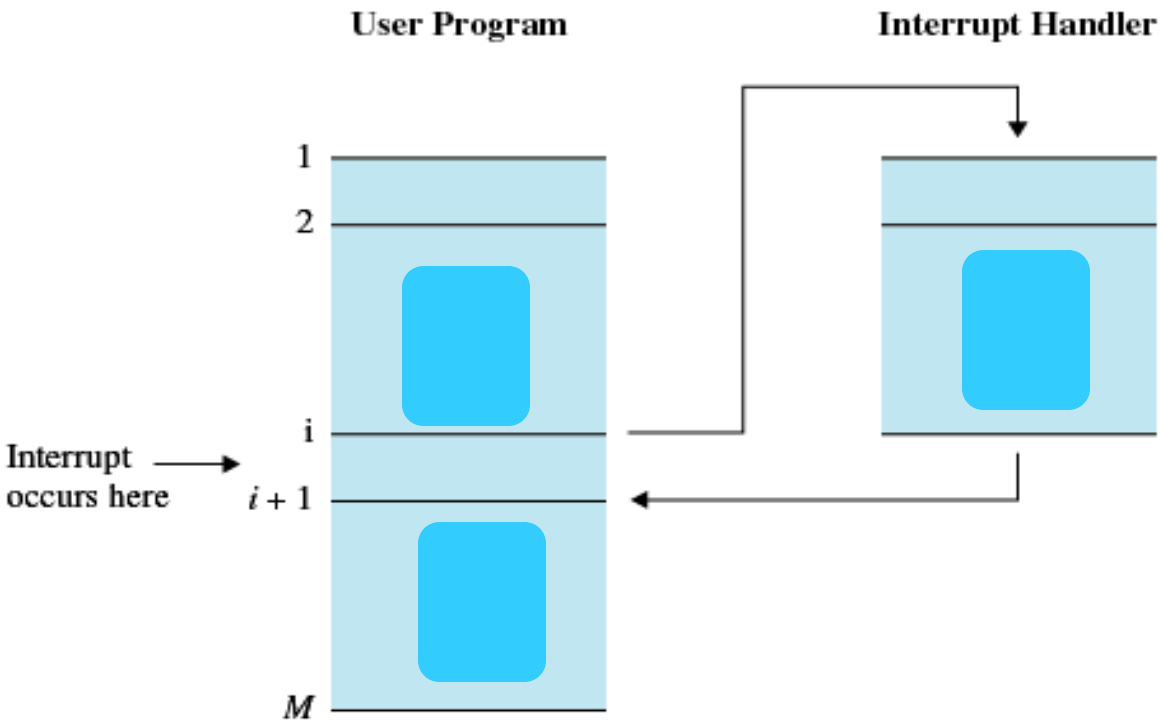
Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure, such as power failure or memory parity error.

Interrupt Handler

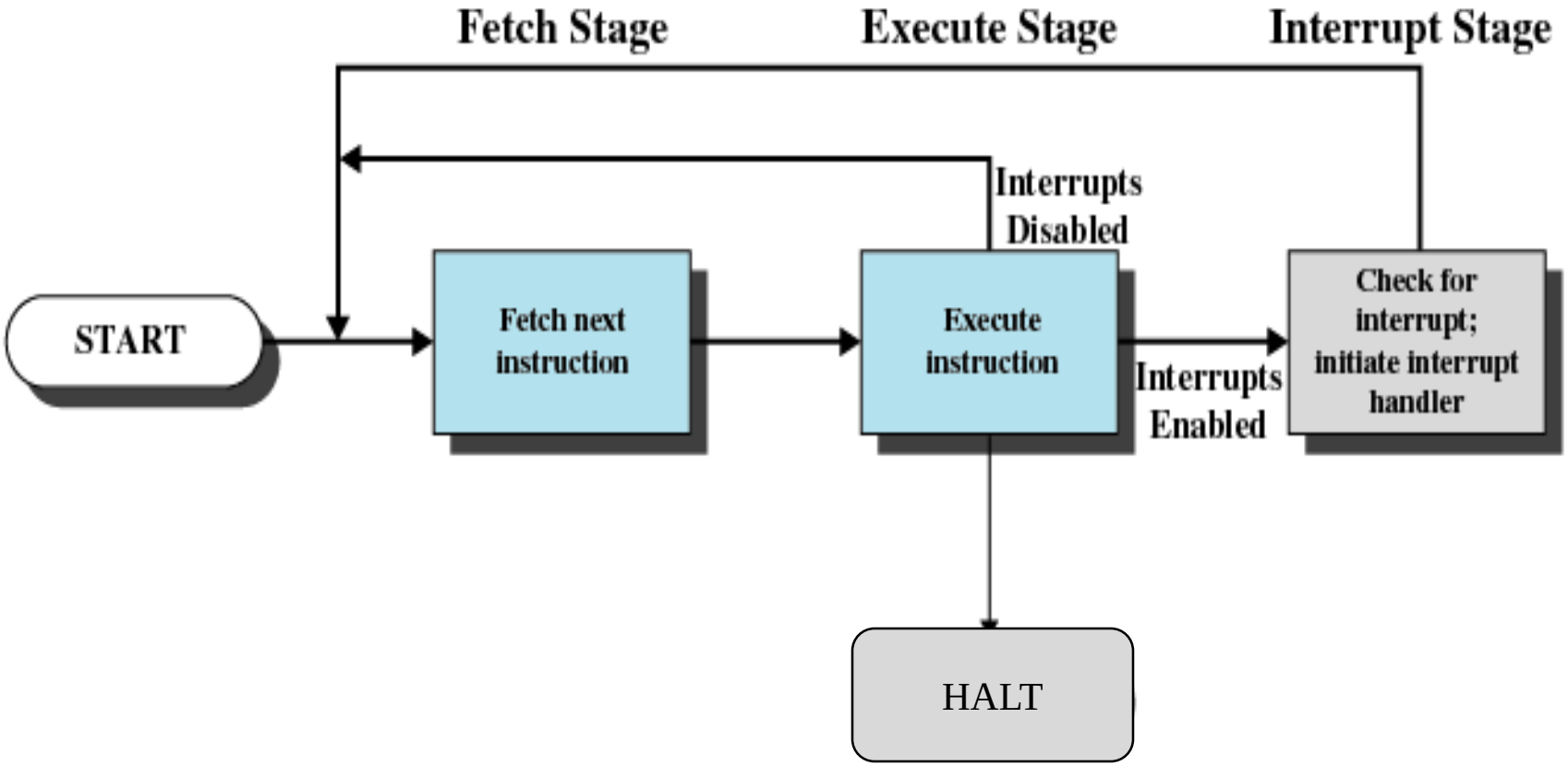
- Program to service a particular I/O device
- Generally part of the operating system

Interrupts

- Suspends the normal sequence of execution



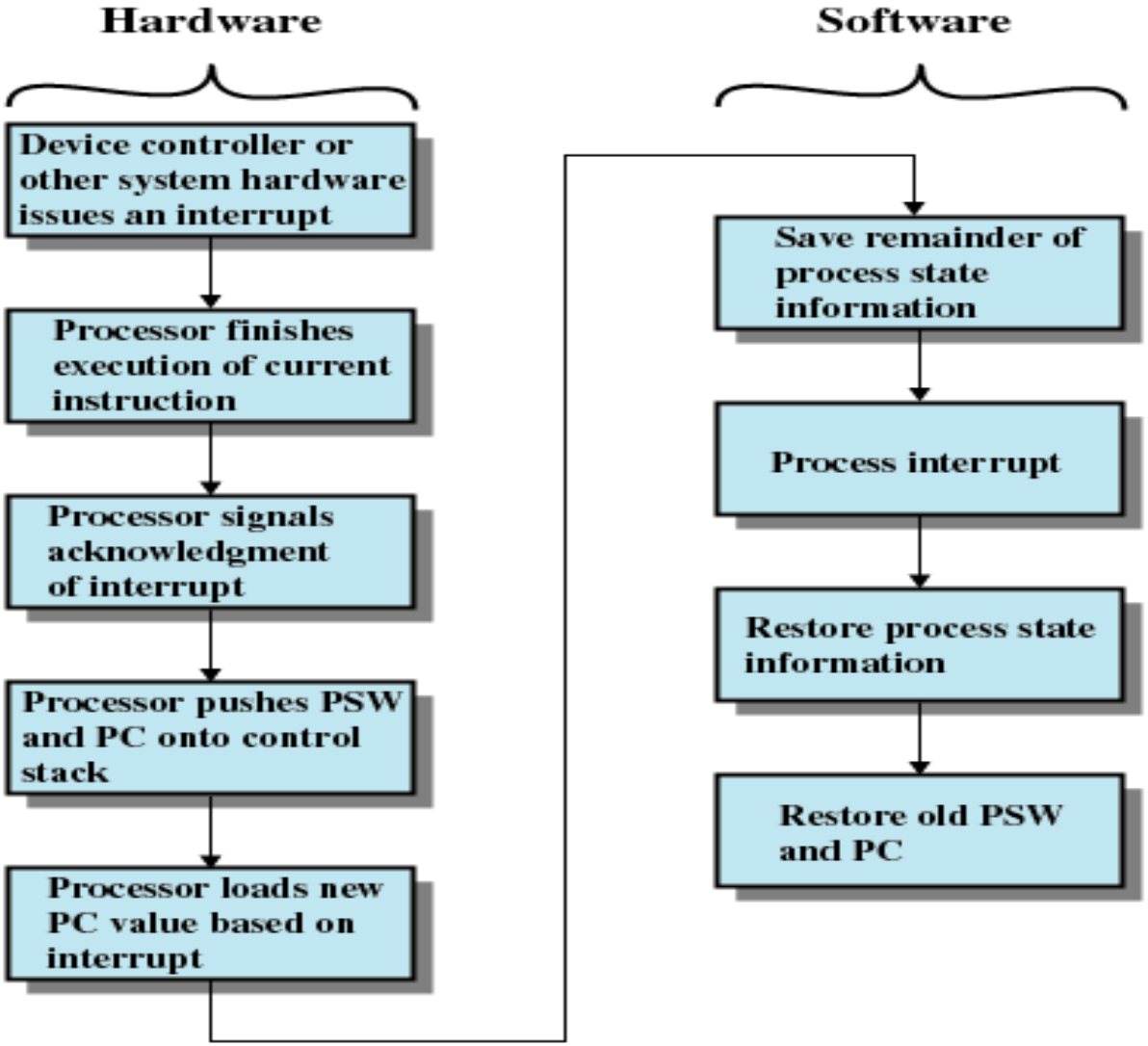
Interrupt Cycle



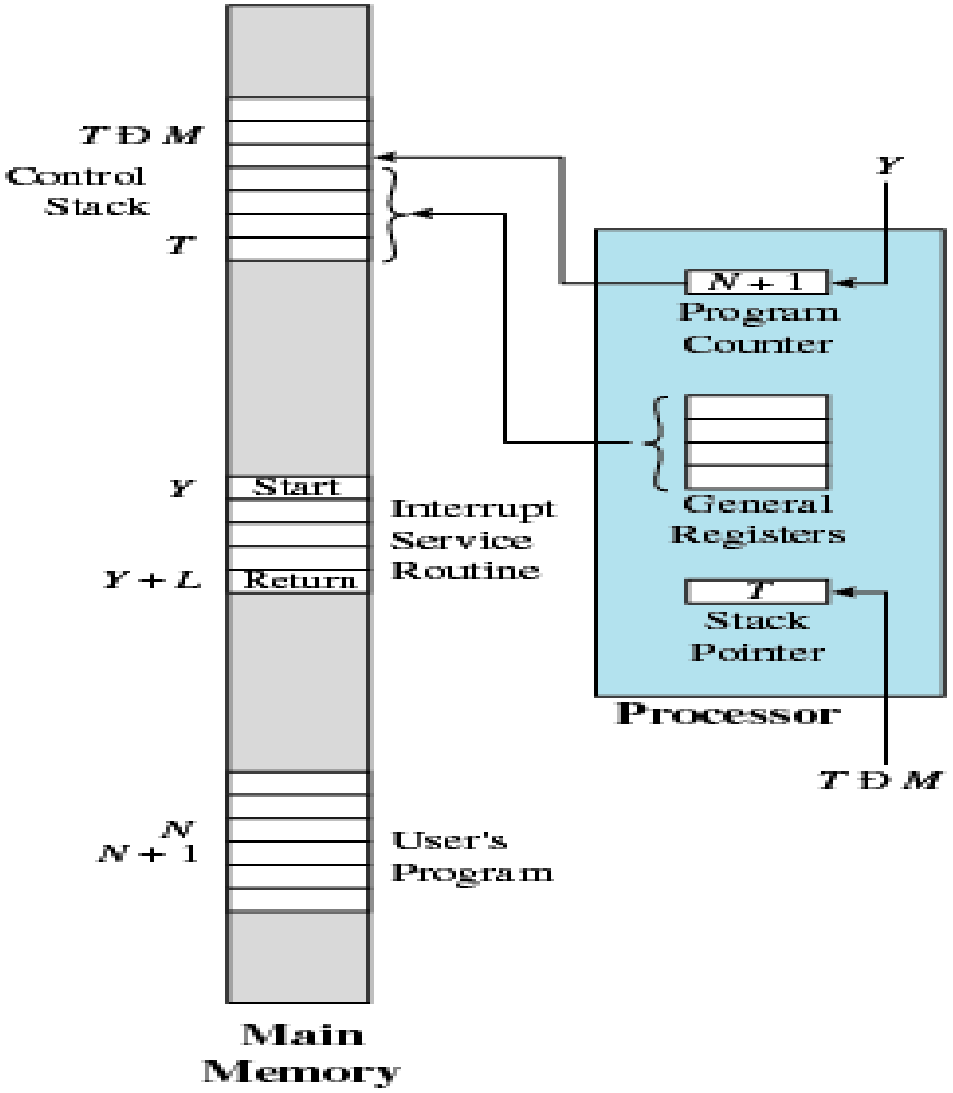
Interrupt Cycle

- Processor checks for interrupts
- If no interrupts, fetch the next instruction for the current program
- If an interrupt is pending, suspend execution of the current program, and execute the interrupt-handler routine

Simple Interrupt Processing

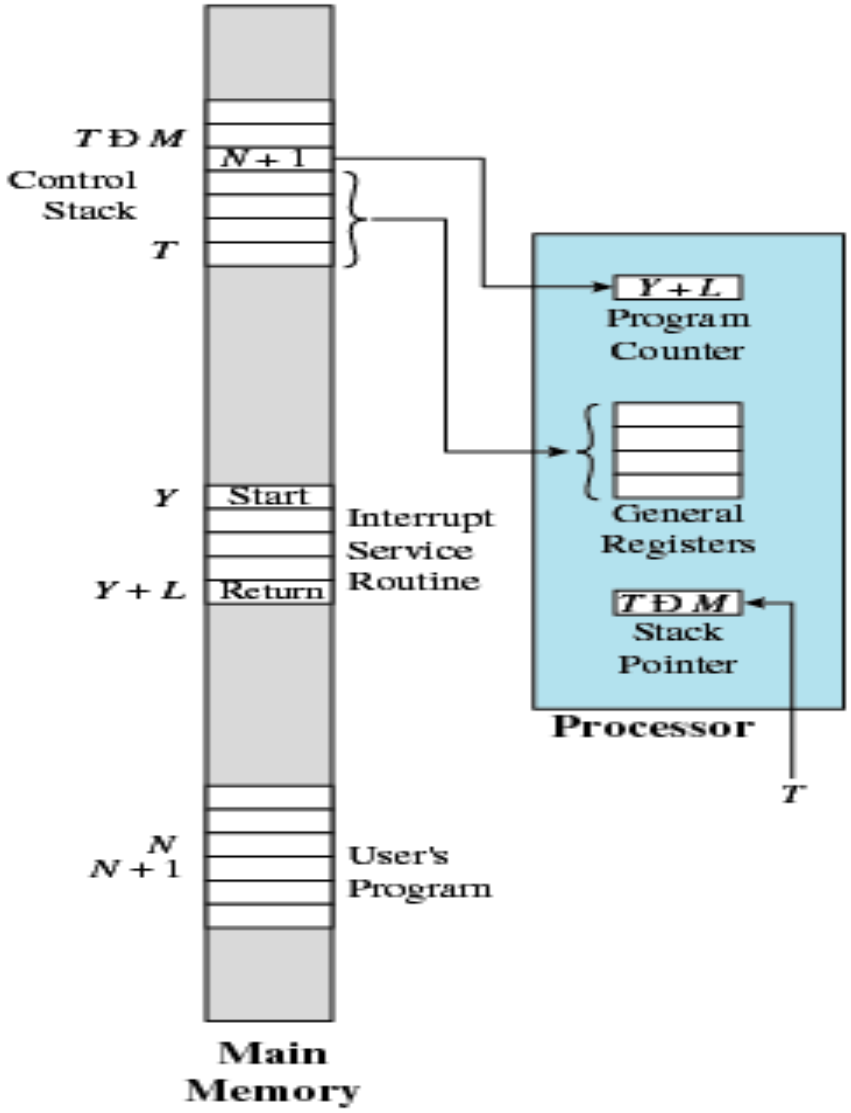


Changes in Memory and Registers for an Interrupt



- Interrupt occurs after instruction at location *N*

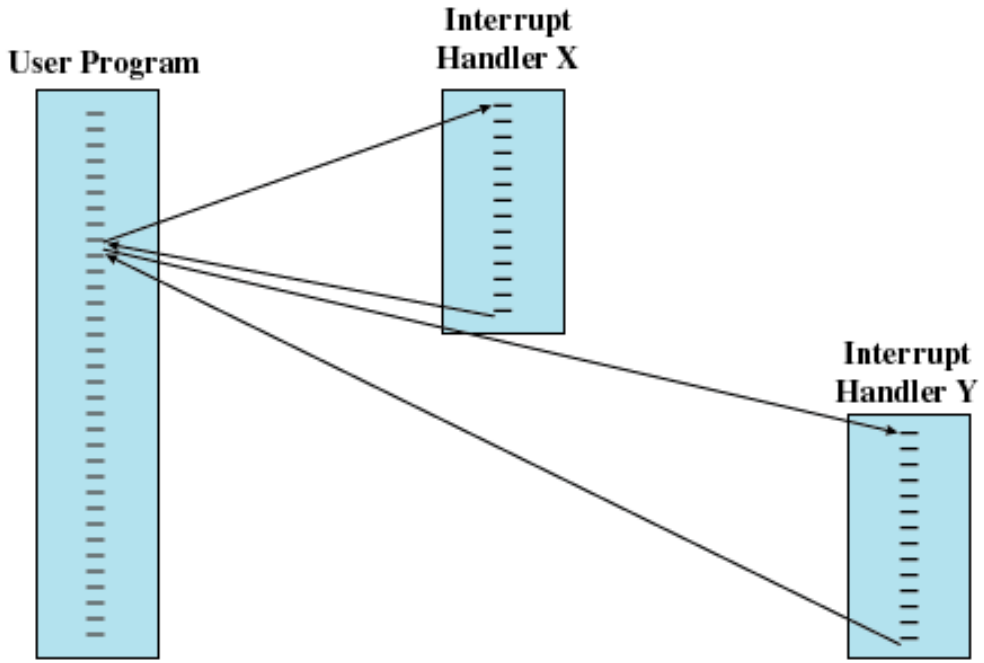
Changes in Memory and Registers for an Interrupt



Return from interrupt

Multiple Interrupts

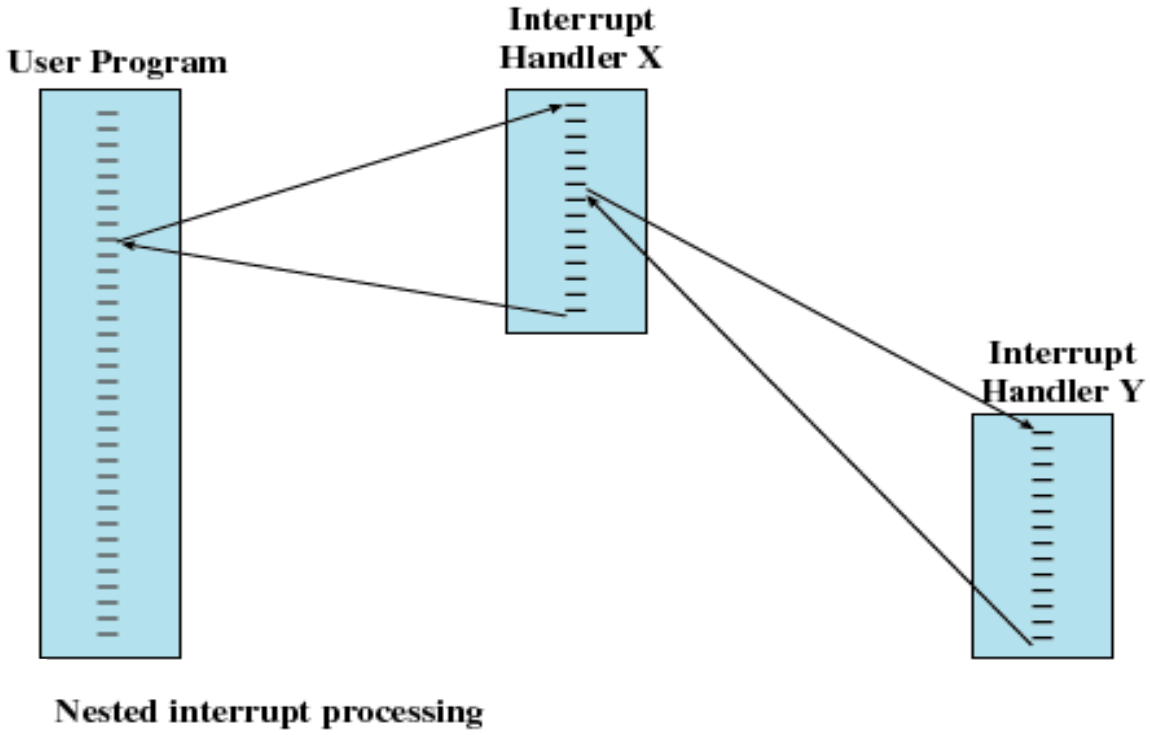
- Disable interrupts while an interrupt is being processed



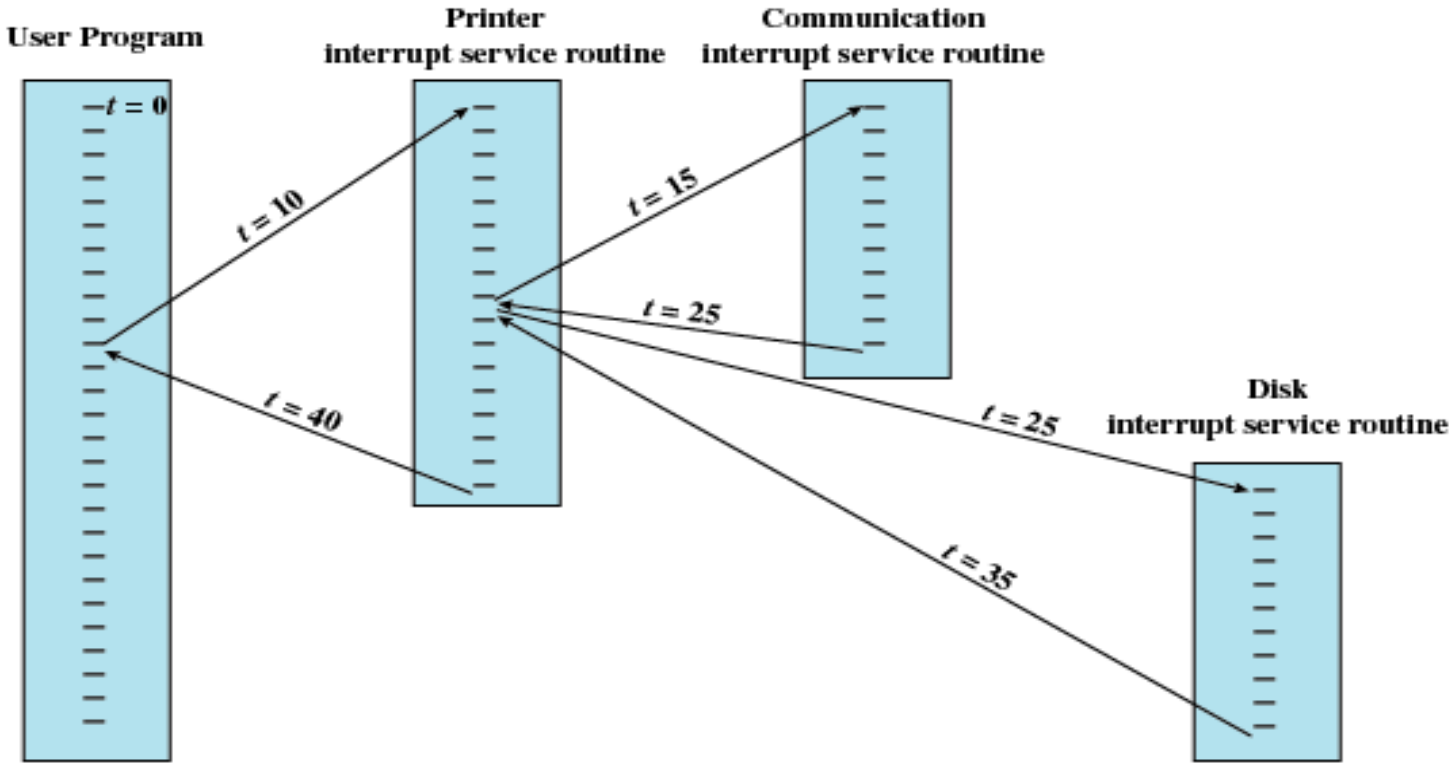
Sequential interrupt processing

Multiple Interrupts

- Define priorities for interrupts



Multiple Interrupts



Example Time Sequence of Multiple Interrupts

System Structure

- Over the years as more and more features have been added to the OS, and the underlying hardware has become more capable and versatile, the size and complexity of operating systems has grown.
- IBM's OS/360 (1964) contained just over 10^6 machine instructions. By the mid-1970s the Multics OS contained more than $20x10^6$ machine instructions. Windows 2000 contains more than $32x10^6$ lines of code. Windows XP has more than $40x10^6$ lines of code. The Linux Fedora 9 kernel contains about $7x10^6$ lines of code with the entire distribution just over $204x10^6$ lines of code. Windows 7 contains about $50x10^6$ lines of code
- Modular programming alone is not sufficient to manage the development of such large systems of code. There has been an increasing use of hierarchical layers and information abstraction in the design of modern OS.
- The hierarchical approach views the OS as a series of levels where each level performs a related subset of functions
- Each level relies on the next lower level to perform more primitive functions. This decomposes a problem into a number of more manageable subproblems

System Structure

- In general, lower layers deal with a far shorter time scale.
- Some parts of the OS must interact directly with the computer hardware, where events can have a time scale as brief as a few billionths of a second.
- At the other end of the spectrum, parts of the OS communicate with the user, who issues commands at a much more leisurely pace, perhaps one every few seconds.

External Objects

Processor

Hardware

Level	Name	Objects	Example Operations
13	Shell	User programming environment	Statements in shell language
12	User processes	User processes	Quit, kill, suspend, resume, …
11	Directories	Directories	Create, destroy, attach, search, …
10	Devices	External devices	Open, close, read, write, …
9	File system	Files	Create, destroy, open, close, read, …
8	Communications	Pipes	Create, destroy, open, write, …
7	Virtual memory	Segments, pages	Read, write, fetch, …
6	Local secondary store	Blocks, device channels	Read, write, allocate, free, …
5	Primitive processes	Semaphores, ready list	Suspend, resume, wait, signal, …
4	Interrupts	Interrupt handlers	Invoke, mask, unmask, retry, …
3	Procedures	Call stack, display	Mark, stack, call, return, …
2	Instruction set	Evaluation stack, micro-programs	Load, store, add, subtract, branch, …
1	Electronic circuits	Registers, gates, buses, etc.	Clear, transfer, activate, complement,

Process Hardware Levels

- Level 1
  - Electronic circuits
  - Objects are registers, memory cells, and logic gates
  - Operations are clearing a register or reading a memory location
- Level 2
  - Processor's instruction set
  - Operations such as add, subtract, load, and store

Process Hardware Levels

- Level 3
  - Adds the concept of a procedure or subroutine, plus call/return operations
- Level 4
  - Interrupts

Concepts with Multiprogramming

- Level 5
  - Process as a program in execution
  - Suspend and resume processes
- Level 6
  - Secondary storage devices
  - Transfer of blocks of data
- Level 7
  - Creates logical address space for processes
  - Organizes virtual address space into blocks

Deal with External Objects

- Level 8
  - Communication of information and messages between processes
- Level 9
  - Supports long-term storage of named files
- Level 10
  - Provides access to external devices using standardized interfaces

Deal with External Objects

- Level 11
  - Responsible for maintaining the association between the external and internal identifiers
- Level 12
  - Provides full-featured facility for the support of processes
- Level 13
  - Provides an interface to the operating system for the user

Modern Operating Systems

- Monolithic kernel
  - Most of what is viewed as OS functionality is provided in a large kernel
- Microkernel architecture
  - Assigns only a few essential functions to the kernel
    - Address spaces, interprocess communication (IPC), and basic scheduling
  - Other OS functionality (services) are provided by processes, sometimes called servers, that run in user mode and are treated like any other application by the microkernel.
  - Microkernel approach simplifies implementation, provides flexibility, and is well suited to a distributed environment. In essence, a microkernel interacts with local and remote server processes in the same way, facilitating construction of distributed systems.

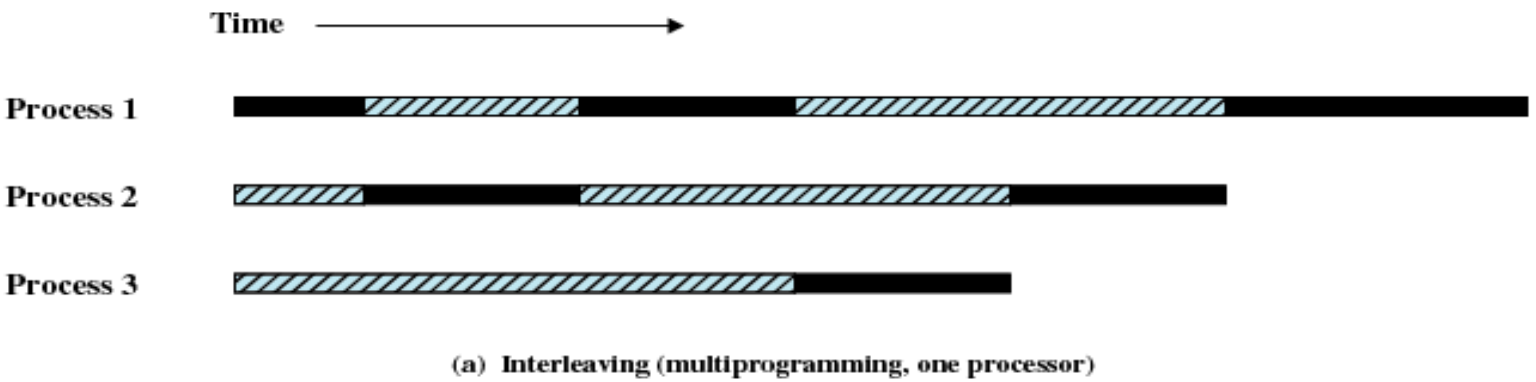
Modern Operating Systems

- Multithreading
  - A process is divided into threads that can run concurrently
    - A thread is a dispatchable unit of work. Includes a processor context (which includes the program counter and stack pointer) and its own data area for a stack (to enable subroutine branching).
      - Executes sequentially and is interruptible so that the processor can run another thread.
    - A process is a collection of one or more threads and associated system resources (such as memory containing both code and data, open files, and devices). This corresponds closely to the concept of a program in execution.

Modern Operating Systems

- Until fairly recently, virtually all single-user personal computers and workstations contained a single general-purpose microprocessor. As demands for performance has increased and the cost of microprocessors has continued to drop, vendors have introduced computers with multiple microprocessors.
- To achieve greater efficiency and reliability one technique is to employ symmetric multiprocessing (SMP), which describes both a hardware architecture and a OS behavior.
- In SMP
  - There are multiple processors
  - These processors share same main memory and I/O facilities, interconnected by a communication bus or other internal connection scheme
  - All processors can perform the same functions (hence the term symmetric)

Multiprogramming and Multiprocessing



(a) Interleaving (multiprogramming, one processor)

(b) Interleaving and overlapping (multiprocessing; two processors)

Blocked Running

Multiprogramming and Multiprocessing

Modern Operating Systems

- Distributed operating systems
  - Provides the illusion of a single main memory space and single secondary memory space
- Object-oriented design
  - Used for adding modular extensions to a small kernel
  - Enables programmers to customize an operating system without disrupting system integrity

Threads and SMP

- Operating system routines can run on any available processor
- Different routines can execute simultaneously on different processors
- Multiple threads of execution within a single process may execute on different processors simultaneously
- Server processes may use multiple threads
- Share data and resources between process

Requirements of an Operating System

- Interleave the execution of multiple processes to maximize processor utilization while providing reasonable response time
- Allocate resources to processes
- Support inter-process communication and user creation of processes

Process

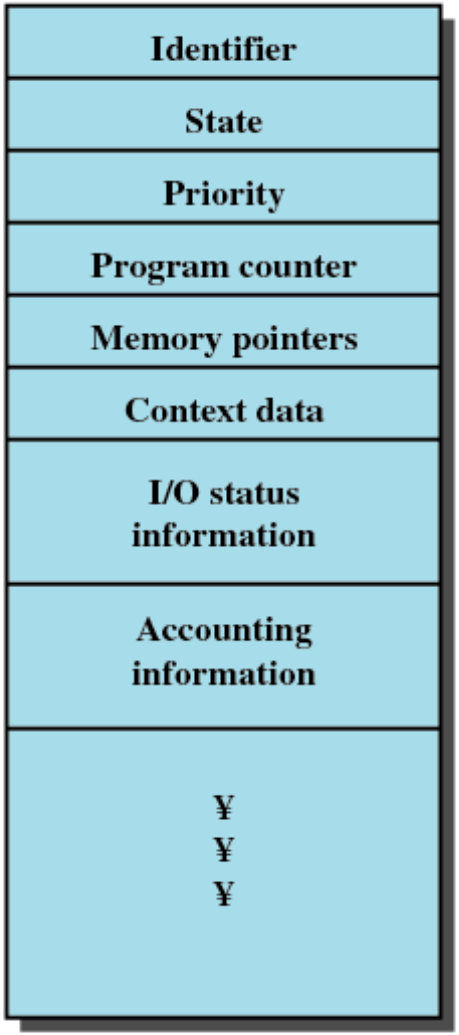
- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of activity characterized by the execution of a sequence of instructions, a current state, and an associated set of system instructions.
- Historically referred to as a job.

Process Elements

- Identifier – a unique process id
- State – running, suspended, etc. (more later)
- Priority – priority relative to other processes
- Program counter – address of next instruction to be executed.
- Memory pointers – pointers to program code and data plus any shared with other processes.
- Context data – data present in registers while the process is executing
- I/O status information – all outstanding I/O request, I/O devices assigned to this process
- Accounting information – processor time used, clock time used, account numbers, etc.

Process Control Block (PCB)

- Contains the process elements
- Created and manage by the operating system
- Allows support for multiple processes

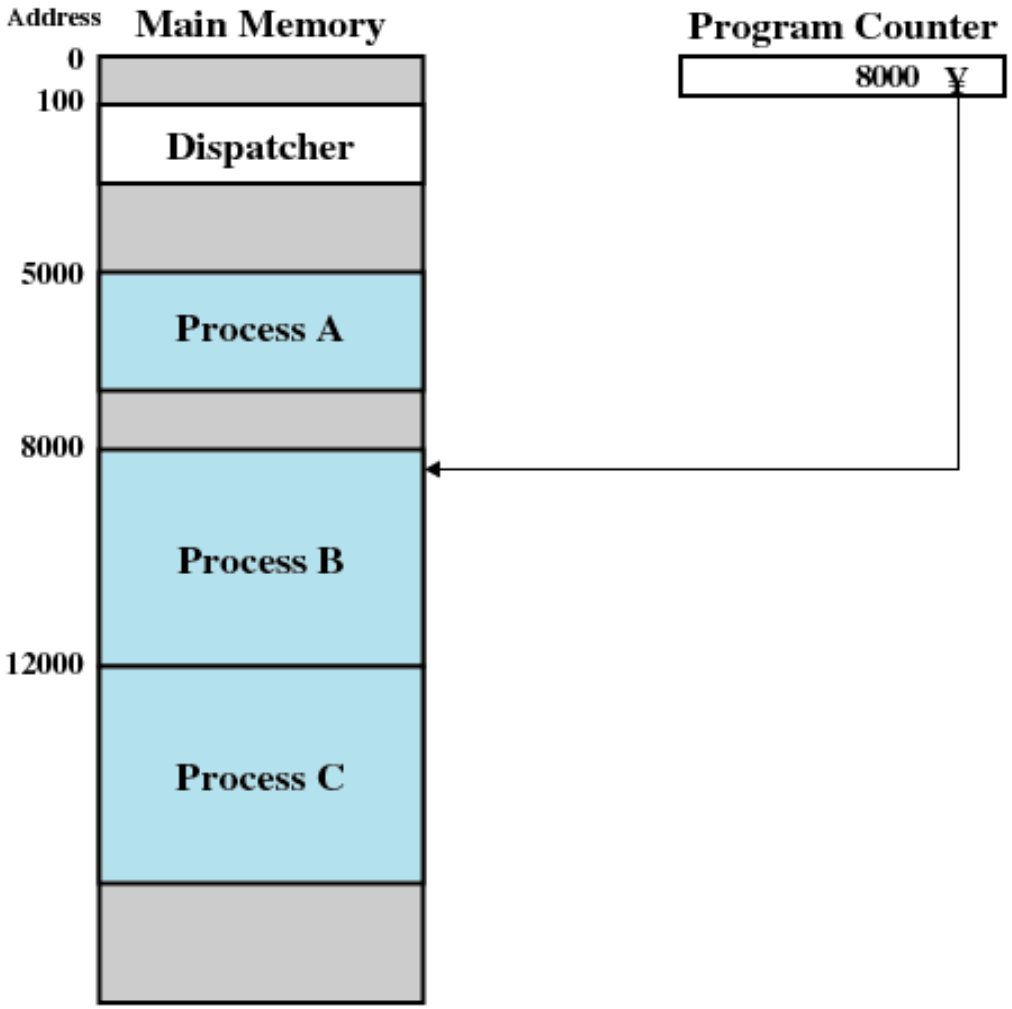


Simplified Process Control Block

Trace of Process

- The behavior of an individual process can be characterized by listing the sequence of instructions that execute for that process. Such a listing is referred to as a trace.
- The dispatcher switches the processor from one process to another.
- We can characterize the behavior of the processor by showing how the traces of the various processes are interleaved.

Example Execution



Snapshot of Example Execution
at Instruction Cycle 13

Trace of Processes

(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C
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

5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Instruction Cycle	Instruction Address
1	5000
2	5001
3	5002
4	5003
5	5004
6	5005
----------	Time out
7	100
8	101
9	102
10	103
11	104
12	105
13	8000
14	8001
15	8002
16	8003
----------	I/O request
17	100
18	101
19	102
20	103
21	104
22	105
23	12000
24	12001
25	12002
26	12003

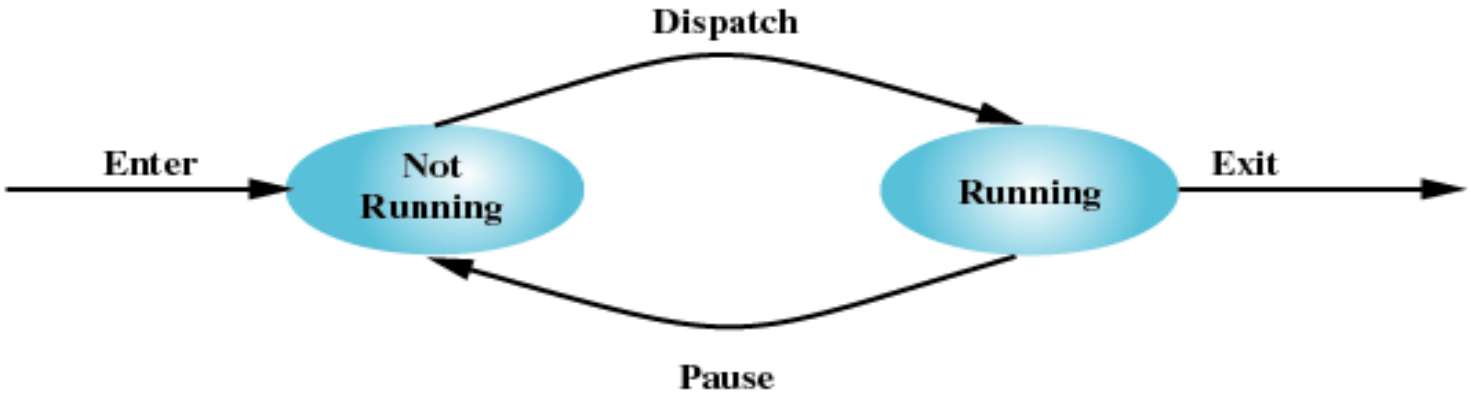
Instruction Cycle	Instruction Address
27	12004
28	12005
----------	Time out
29	100
30	101
31	102
32	103
33	104
34	105
35	5006
36	5007
37	5008
38	5009
39	5010
40	5011
----------	Time out
41	100
42	101
43	102
44	103
45	104
46	105
47	12006
48	12007
49	12008
50	12009
51	12010
52	12011
----------	Time out

100 = Starting address of dispatcher program

shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

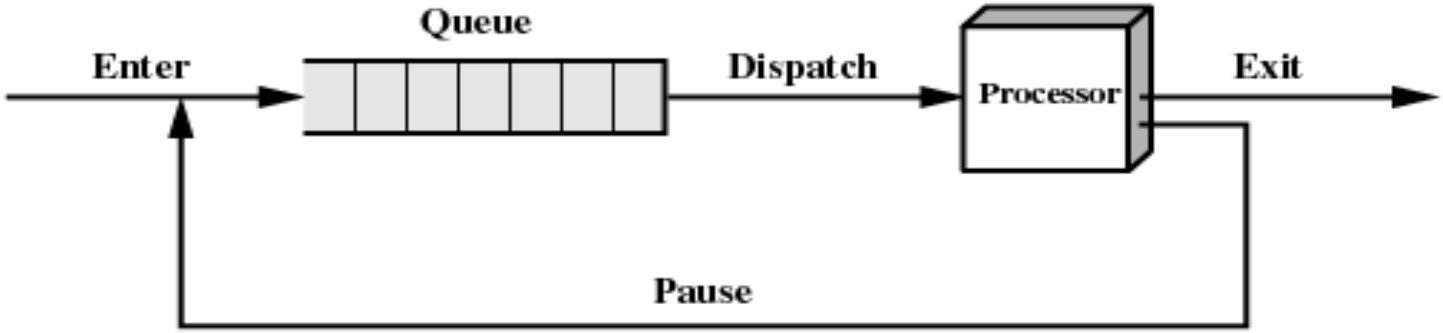
Two-State Process Model

- Process may be in one of two states
  - Running
  - Not-running



State transition diagram

All Not-Running Processes in a Queue



Queuing diagram

Reasons Process Creation Occurs

New batch job	The operating system is provided with a batch job control stream, usually on tape or disk. When the operating system 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 operating system 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.

Reasons For Process Termination

Normal completion	The process executes an OS service call to indicate that it has completed running.
Time limit exceeded	The process has run longer than the specified total time limit. There are a number of possibilities for the type of time that is measured. These include total elapsed time ("wall clock time"), amount of time spent executing, and, in the case of an interactive process, the amount of time since the user last provided any input.
Memory unavailable	The process requires more memory than the system can provide.
Bounds violation	The process tries to access a memory location that it is not allowed to access.
Protection error	The process attempts to use a resource such as a file that it is not allowed to use, or it tries to use it in an improper fashion, such as writing to a read-only file.
Arithmetic error	The process tries a prohibited computation, such as division by zero, or tries to store numbers larger than the hardware can accommodate.

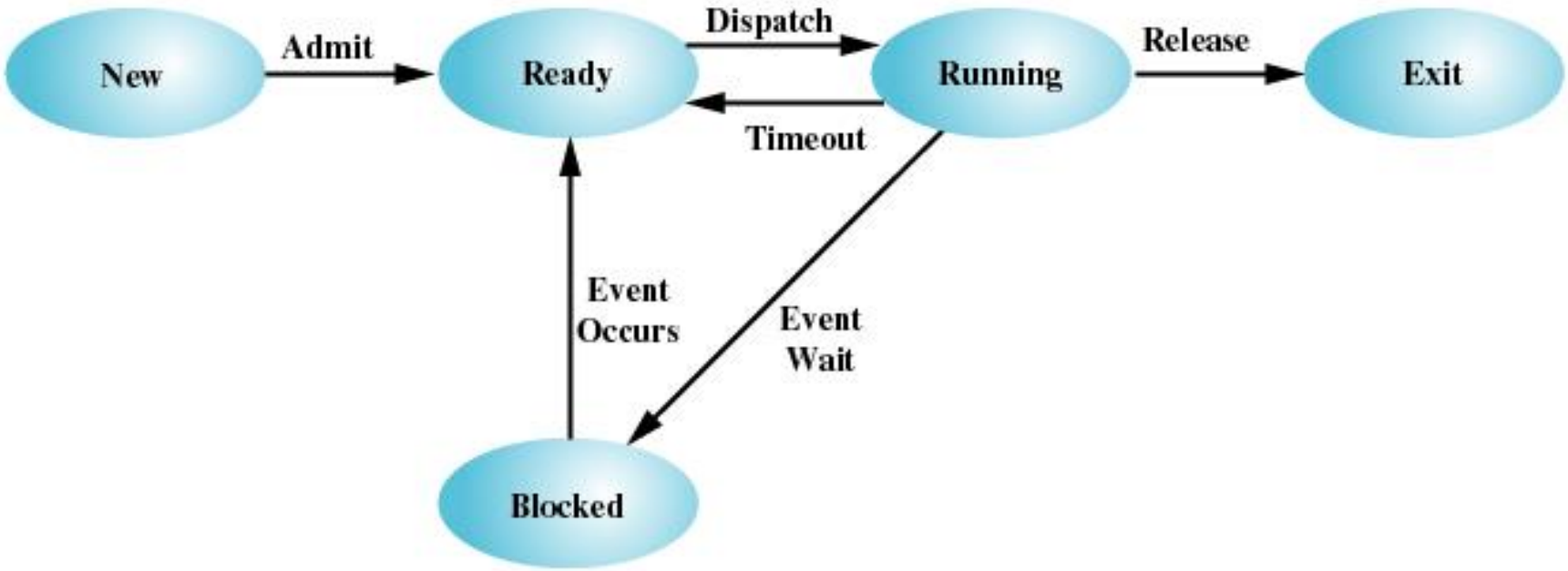
Reasons For Process Termination

Time overrun	The process has waited longer than a specified maximum for a certain event to occur.
I/O failure	An error occurs during input or output, such as inability to find a file, failure to read or write after a specified maximum number of tries (when, for example, a defective area is encountered on a tape), or invalid operation (such as reading from the line printer).
Invalid instruction	The process attempts to execute a nonexistent instruction (often a result of branching into a data area and attempting to execute the data).
Privileged instruction	The process attempts to use an instruction reserved for the operating system.
Data misuse	A piece of data is of the wrong type or is not initialized.
Operator or OS intervention	For some reason, the operator or the operating system has terminated the process (for example, if a deadlock exists).
Parent termination	When a parent terminates, the operating system may automatically terminate all of the offspring of that parent.
Parent request	A parent process typically has the authority to terminate any of its offspring.

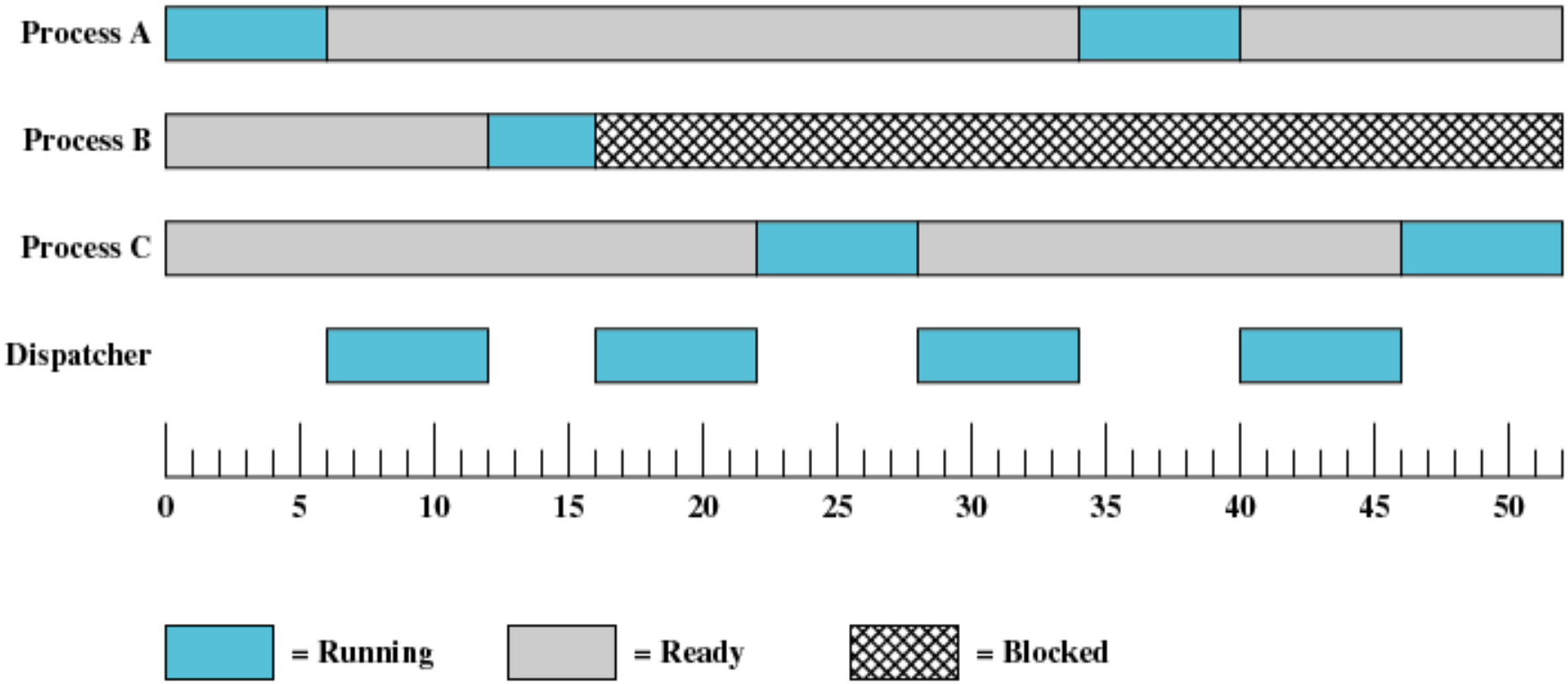
Processes

- Not-running
  - ready to execute
- Blocked
  - waiting for I/O
- Dispatcher cannot just select the process that has been in the queue the longest because it may be blocked

Five-State Process Model

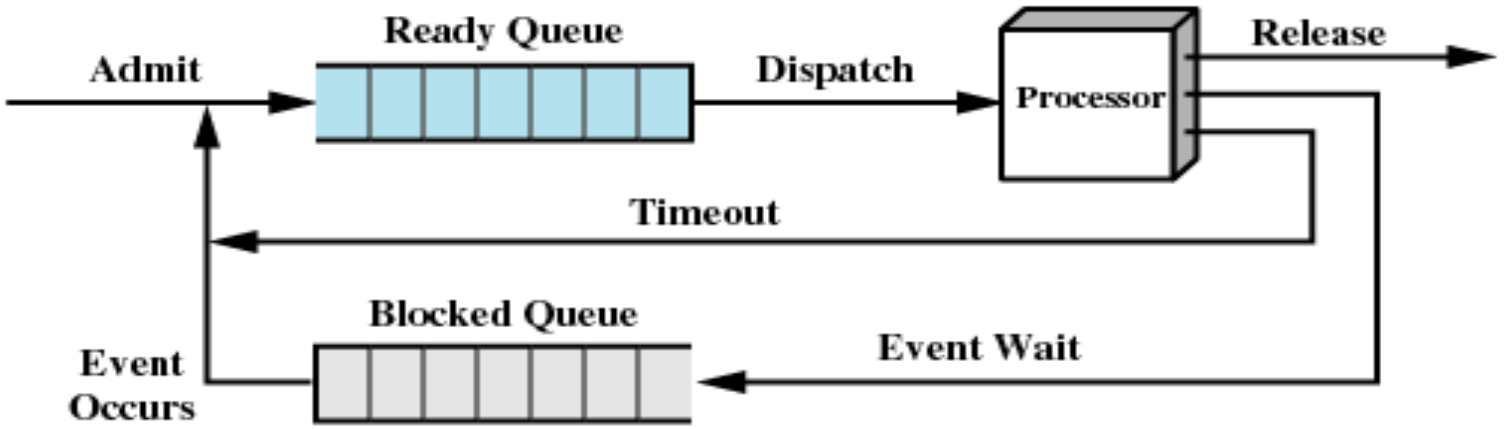


Process States



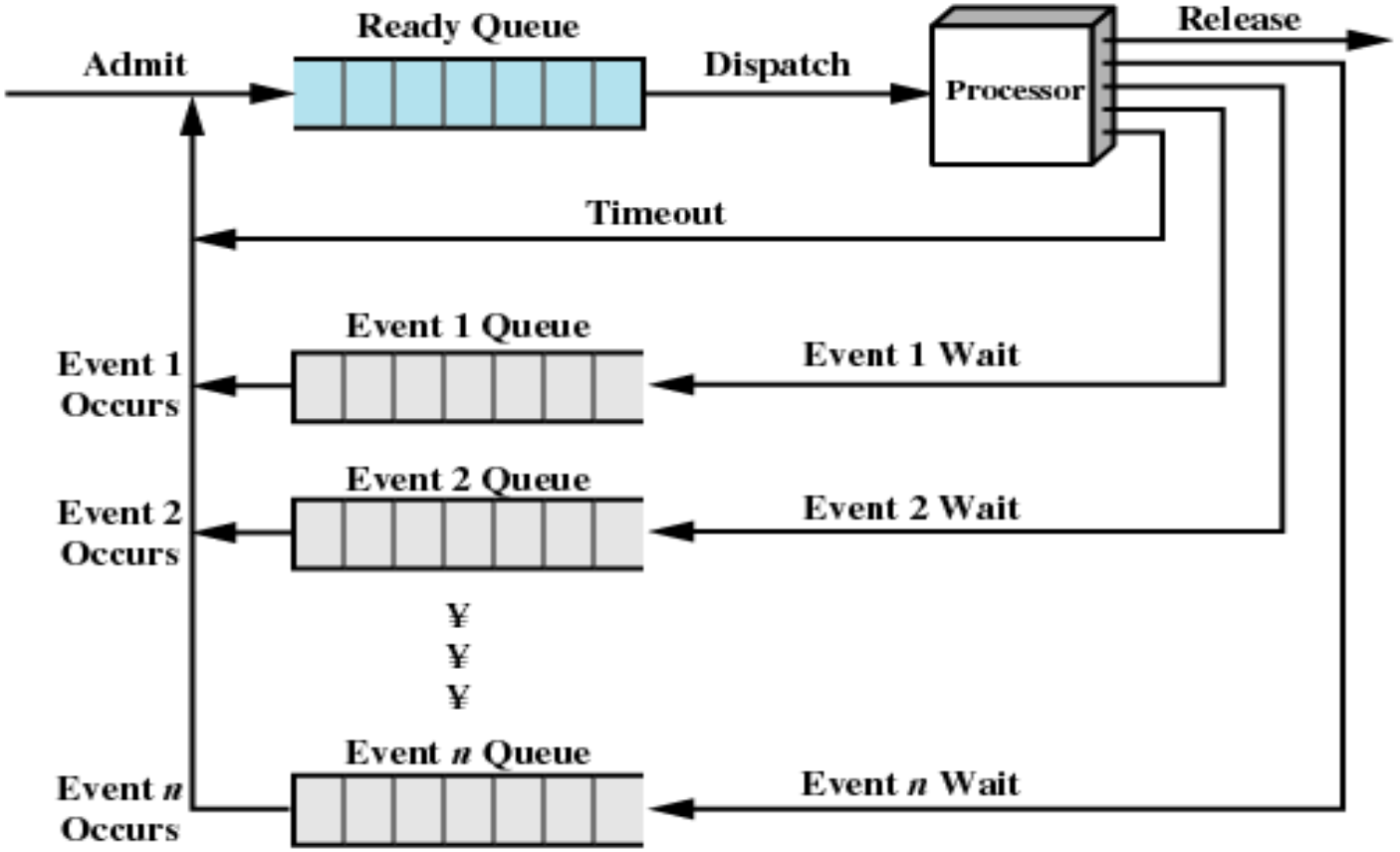
Process states corresponding to trace on page 36

Using Two Queues



(a) Single blocked queue

Multiple Blocked Queues

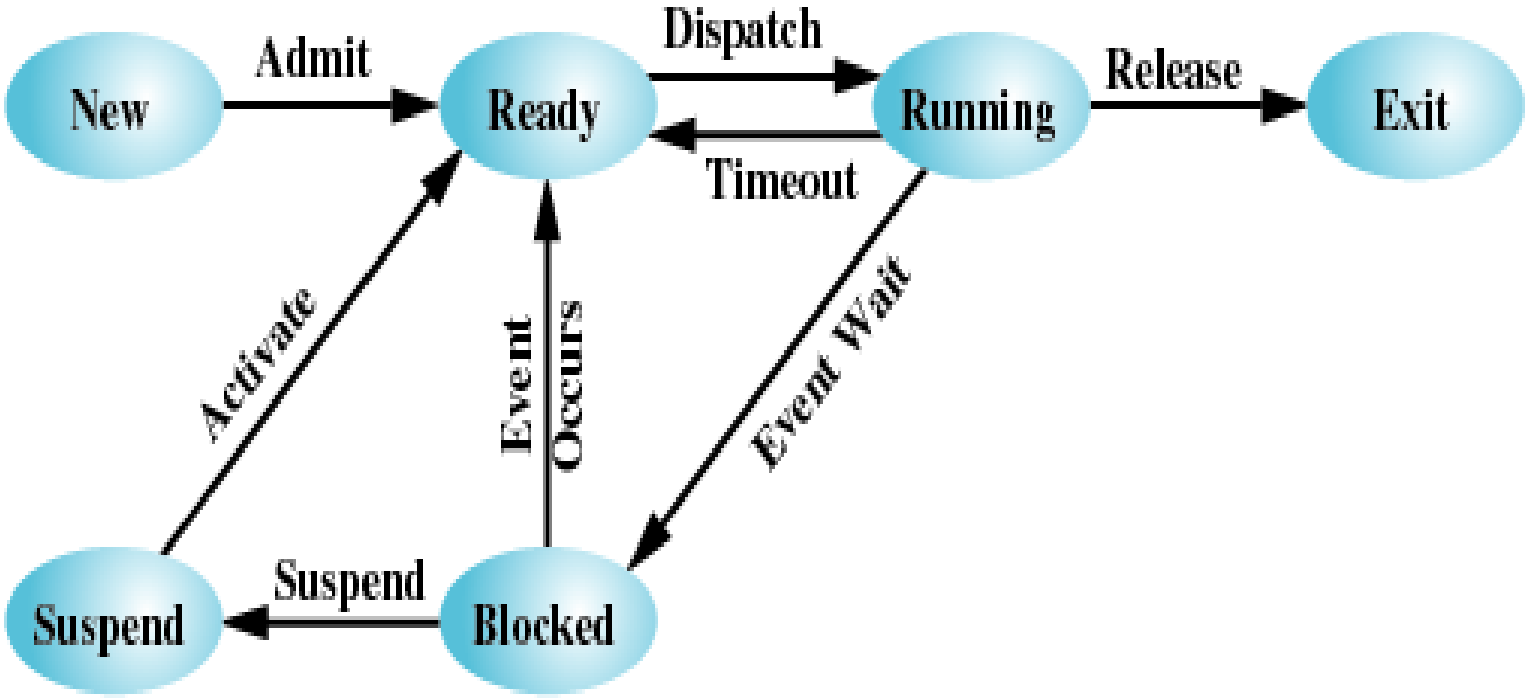


(b) Multiple blocked queues

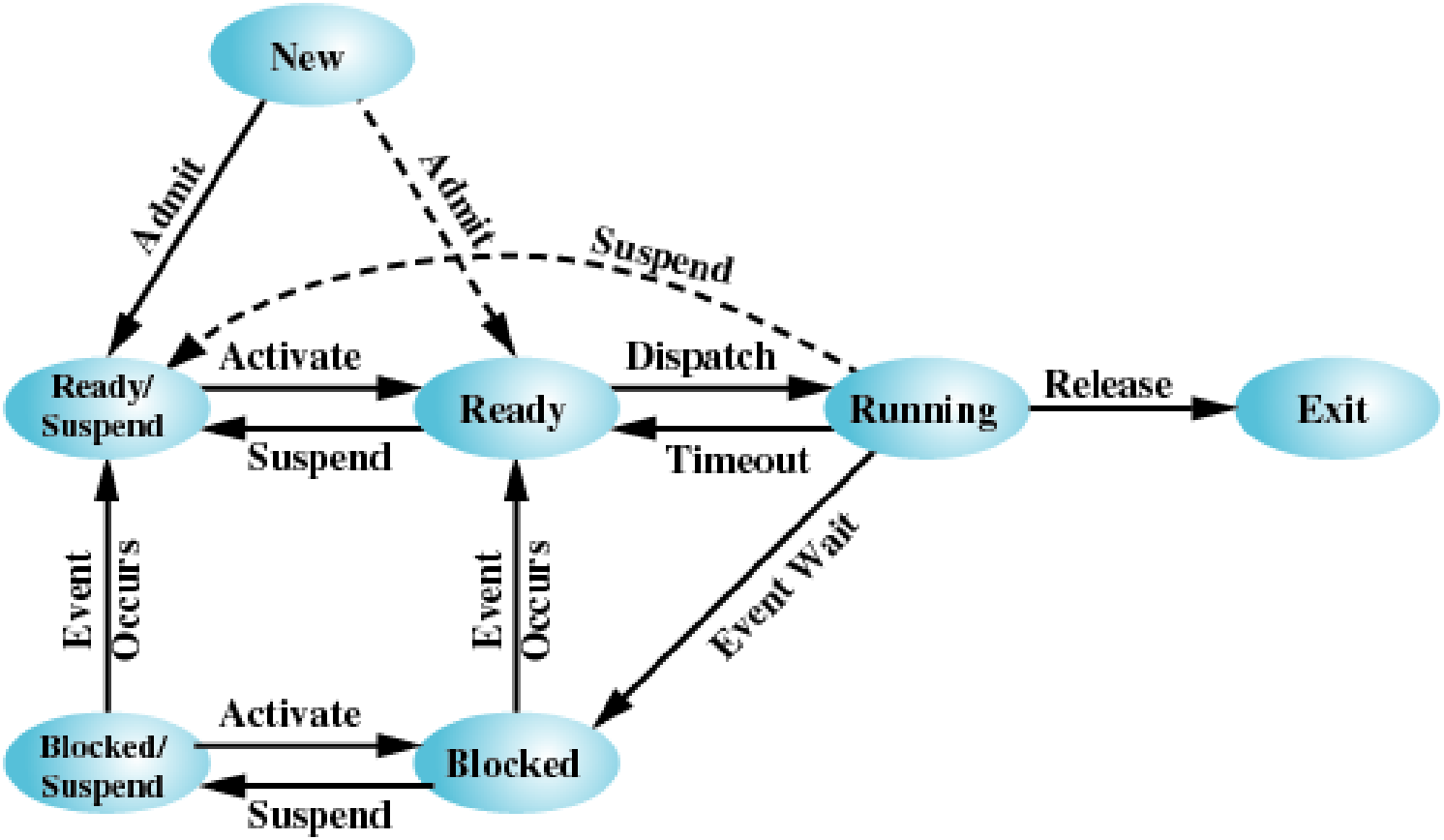
Suspended Processes

- 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



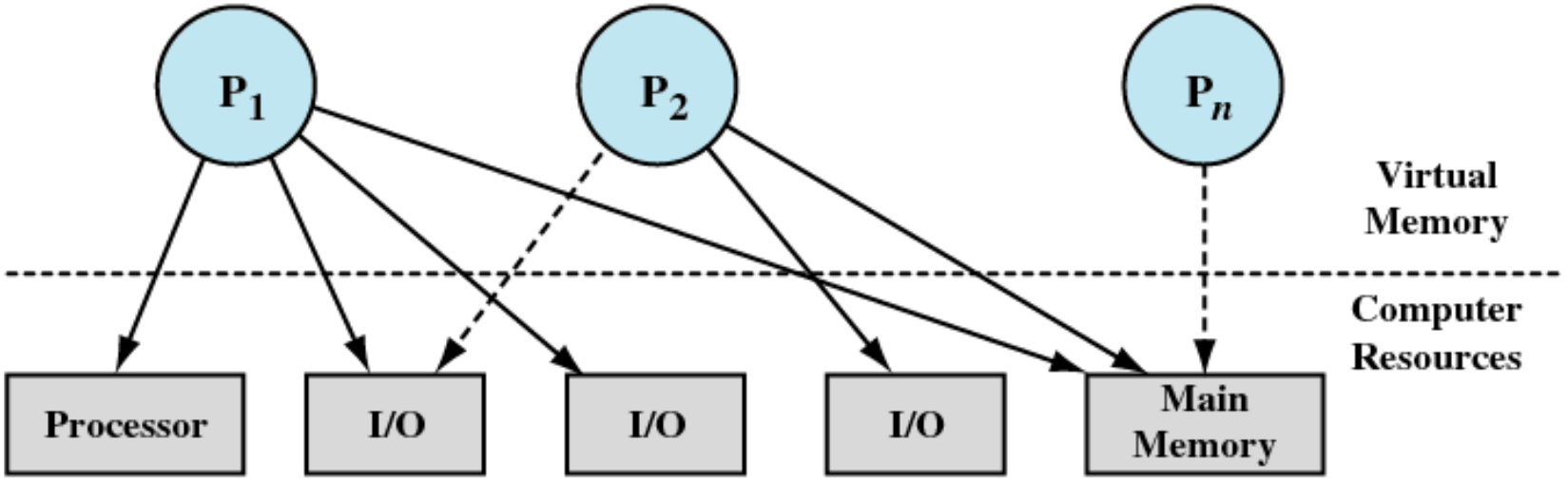
Two Suspend States



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.

Processes and Resources



Operating System Control Structures

- Information about the current status of each process and resource
- Tables are constructed for each entity the operating system manages

Memory Tables

- Allocation of main memory to processes
- Allocation of secondary memory to processes
- Protection attributes for access to shared memory regions
- Information needed to manage virtual memory

I/O Tables

- I/O device is available or assigned
- Status of I/O operation
- Location in main memory being used as the source or destination of the I/O transfer

File Tables

- Existence of files
- Location on secondary memory
- Current Status
- Attributes
- Sometimes this information is maintained by a file management system

Process Table

- Where process is located
- Attributes in the process control block
  - Program
  - Data
  - Stack

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

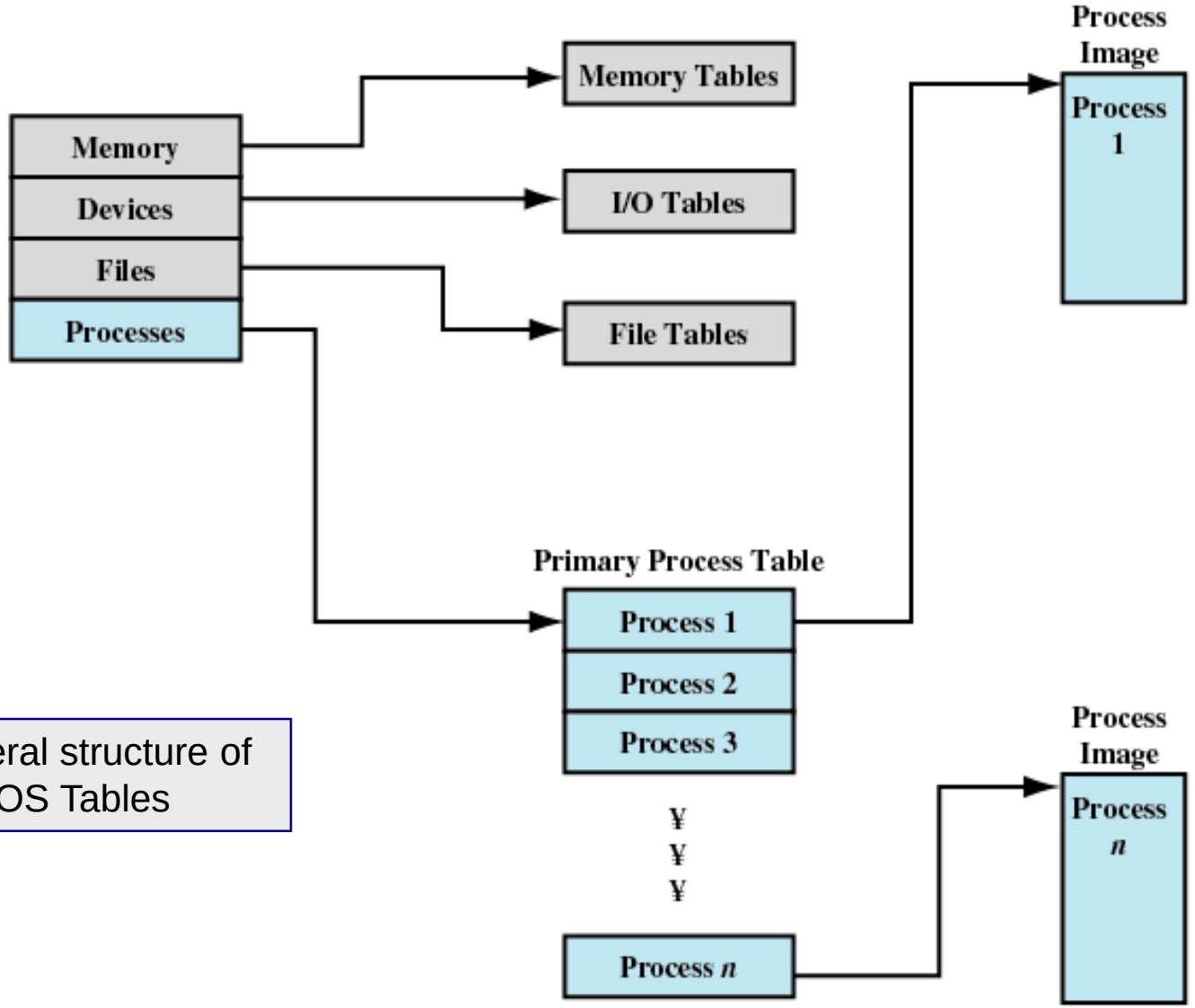
The program to be executed.

System Stack

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.

Process Control Block

Data needed by the operating system to control the process



General structure of OS Tables

Process Control Block

- 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

Process Control Block

- 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.

Process Control Block

- Processor State Information
  - 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

Process Control Block

- 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 Block

- 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 Block

- 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 Block

- Process Control Information
  - 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.

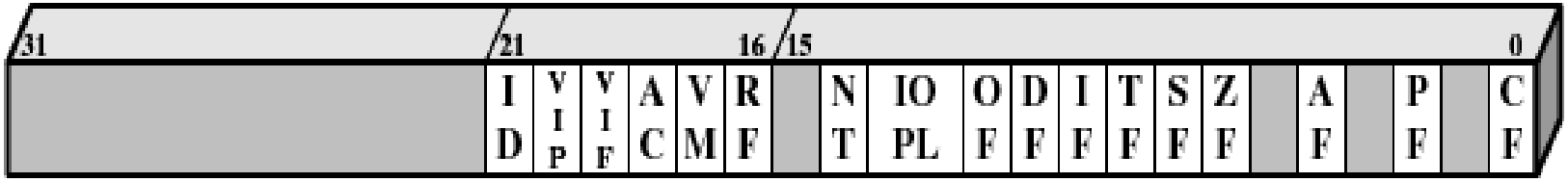
Process Control Block

- 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.

Processor 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 machines

Pentium II EFLAGS Register



ID = Identification flag
VIP = Virtual interrupt pending
VIF = Virtual interrupt flag
AC = Alignment check
VM = Virtual 8086 mode
RF = Resume flag
NT = Nested task flag
IOPL = I/O privilege level
OF = Overflow flag

DF = Direction flag
IF = Interrupt enable flag
TF = Trap flag
SF = Sign flag
ZF = Zero flag
AF = Auxiliary carry flag
PF = Parity flag
CF = Carry flag

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

- 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 occurred
  - may cause process to be moved to Exit state
- Supervisor call
  - such as file open

Change of Process State

- Save context of processor including program counter and other registers
- Update the process control block of the process that is currently in the Running state
- Move process control block to appropriate queue – ready; blocked; ready/suspend
- Select another process for execution
- Update the process control block of the process selected
- Update memory-management data structures
- Restore context of the selected process

CPU Switch From Process to Process