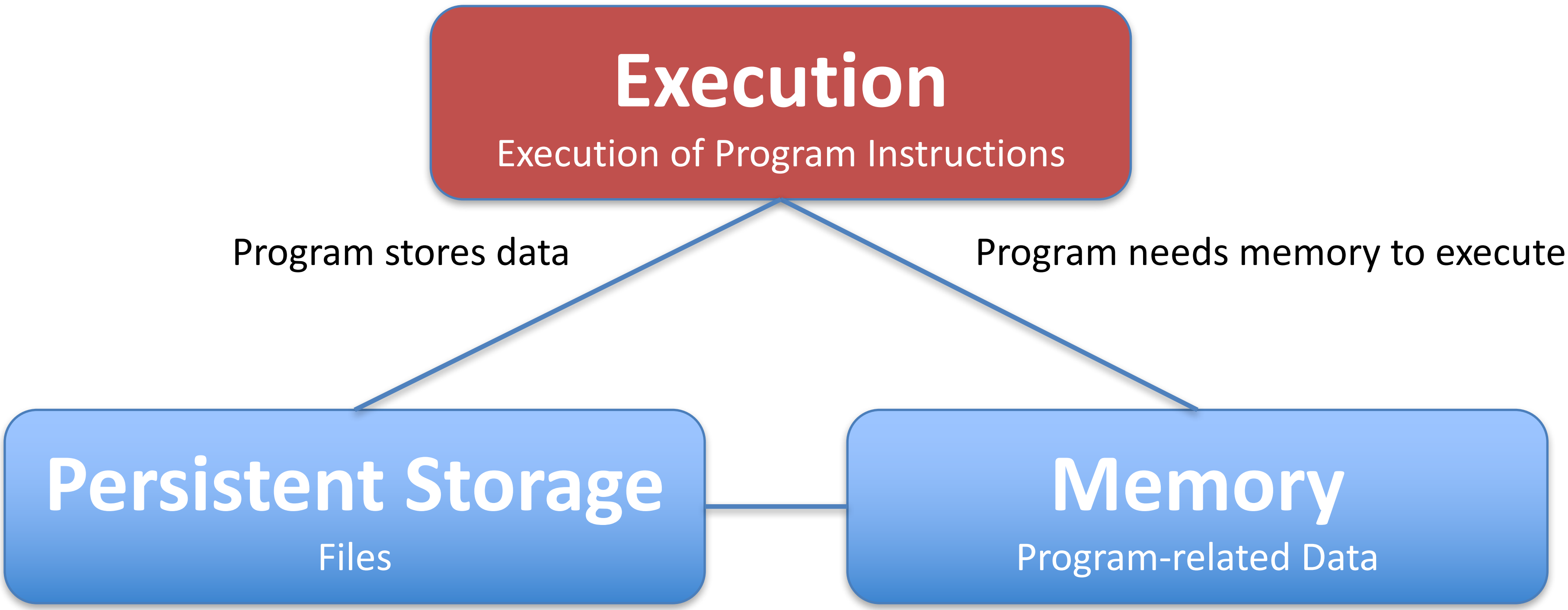
Processes

CS3026 Operating Systems

Lecture 04

Program Execution



Process

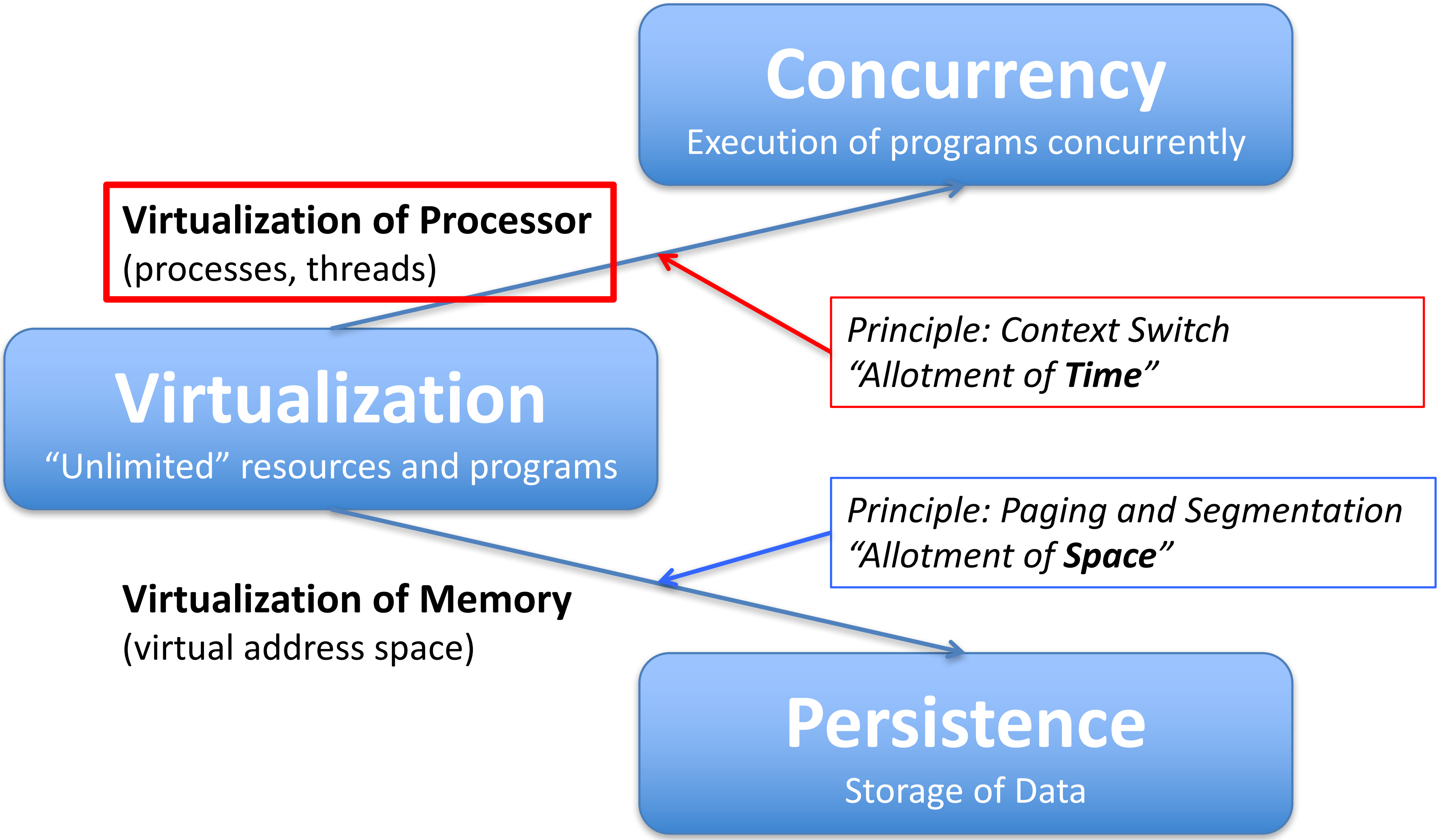
- Definition:
  - "***A process is a program in execution***"

- Processes are a fundamental concept of an operating system
- All the running software on a computer system is organised as a set of processes
- Process is defined as the unit of resource ownership and protection

Process

- *Unit of Execution*
  - enables the concurrent execution of multiple programs on a system
  - CPU switches between processes
- *Unit of Protection*:
  - Protected access to processors, other processes, files, I/O
  - Operating system protects process to prevent unwanted interference between processes
- *Unit of Resource ownership*:
  - Processes own address space (Virtual memory space to hold the process image)
  - Processes own a set of resources (I/O devices, I/O channels, files, main memory)

Core Concepts



Process Control

Process Representation

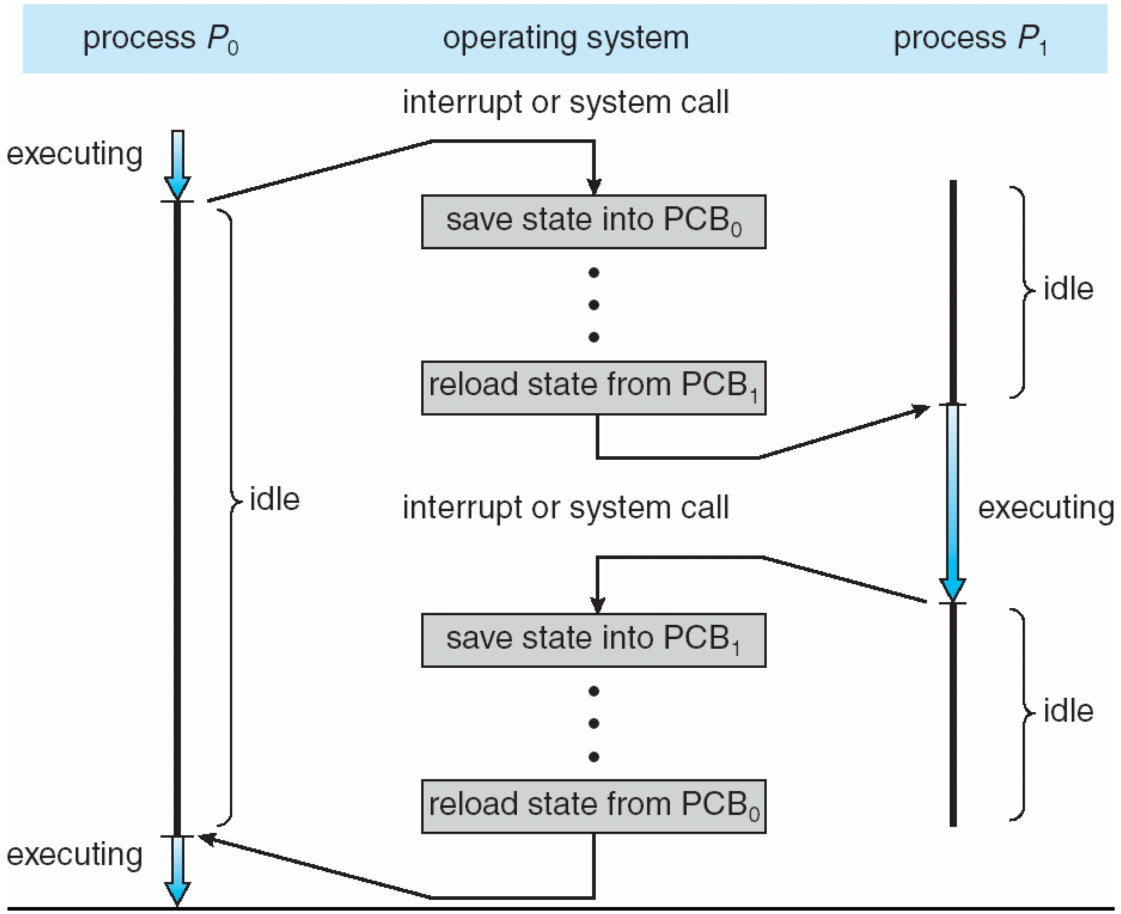
- Three basic components
  - Executable program code
  - Data related to the program
  - Execution context
    - Process ID, group ID, user ID
    - Stack pointer, program counter, CPU registers
    - File descriptors, locks, network sockets
- Execution context recorded in the Process Control Block

Process Control Block

Process Identifier	A unique identifier associated with the process
Process State	The state of the process, when it was interrupted
Priority	Priority level relative to other processes
Program counter	Address of next instruction to be executed
Memory pointers	Pointers to program code and data
Context data	Register content
I/O status information	Outstanding I/O requests, assigned I/O devices, list of open files
Accounting information	Used processor time, time limits etc.
. . .	

- The Process Control Block (PCB) contains all the information necessary to suspend a process and later resume its execution

Context Switch



Context Switch

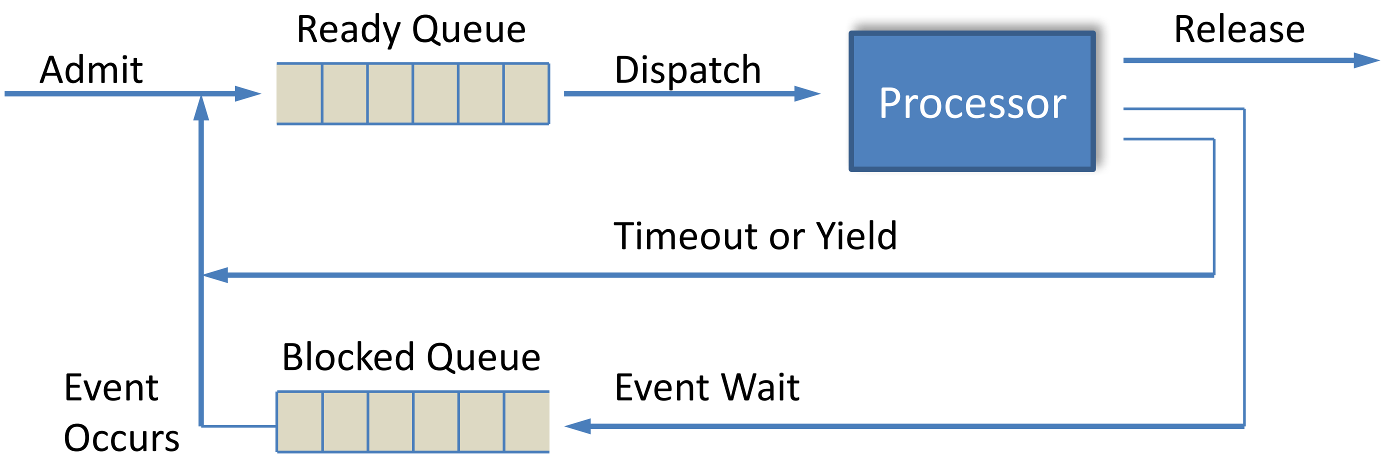
- The current state of the CPU (all its registers) are stored in the PCB of the process currently executing
- Another process is selected for execution – this requires a scheduling decision
- The state of this process, held the process PCB, is loaded into the CPU

Process Control Block

- Process identification
  - Process ID
  - Identifier of creating process (parent process), user ID, group ID
- Processor state information
  - CPU registers: user-visible registers, control and status registers, stack pointers, program counter
- Process control information
  - CPU scheduling information: process state, priority, scheduling-related information
  - Process privileges: access privileges to memory, allowed instructions
  - Memory management information: information about the virtual memory assigned to this process (segment, paging tables, various control registers)
  - Accounting information: time limits, accumulated CPU time etc.
  - I/O status information: resource ownership, open files, allocated I/O devices

Process Execution

Dispatcher

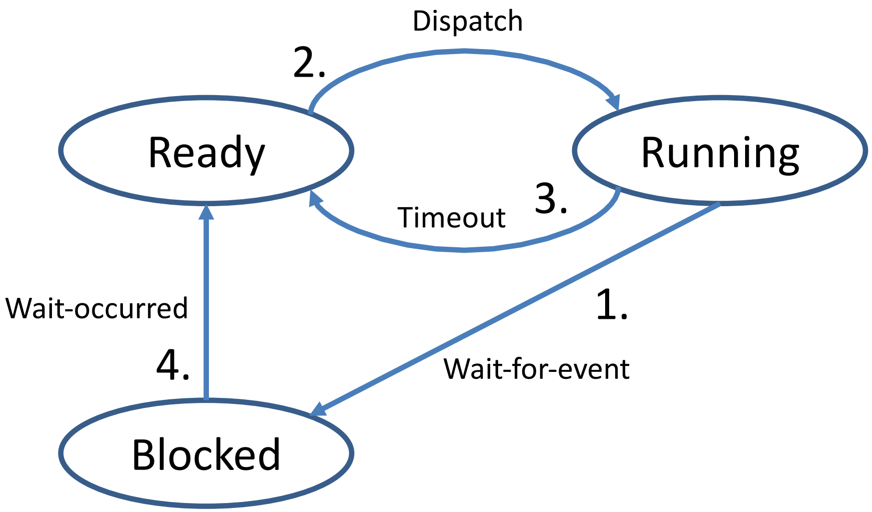


- Implementation:
  - Using one “Ready” and one “Blocked” queue
  - Weakness:
    - When a particular event occurs, ALL processes waiting for this event have to be transferred from the “Blocked” queue to the “Ready” queue
    - Operating system has to look through all the entries in the Blocked queue to select the right processes for transfer

Dispatcher

- Central part of the operating system
- Dispatcher is the OS function that allocates CPU to processes, switches CPU from one process to the next
  - Processes are "dispatched" for execution by the operating system
- Performs context switch
- Manages a "Ready" queue, dispatches next process from this queue

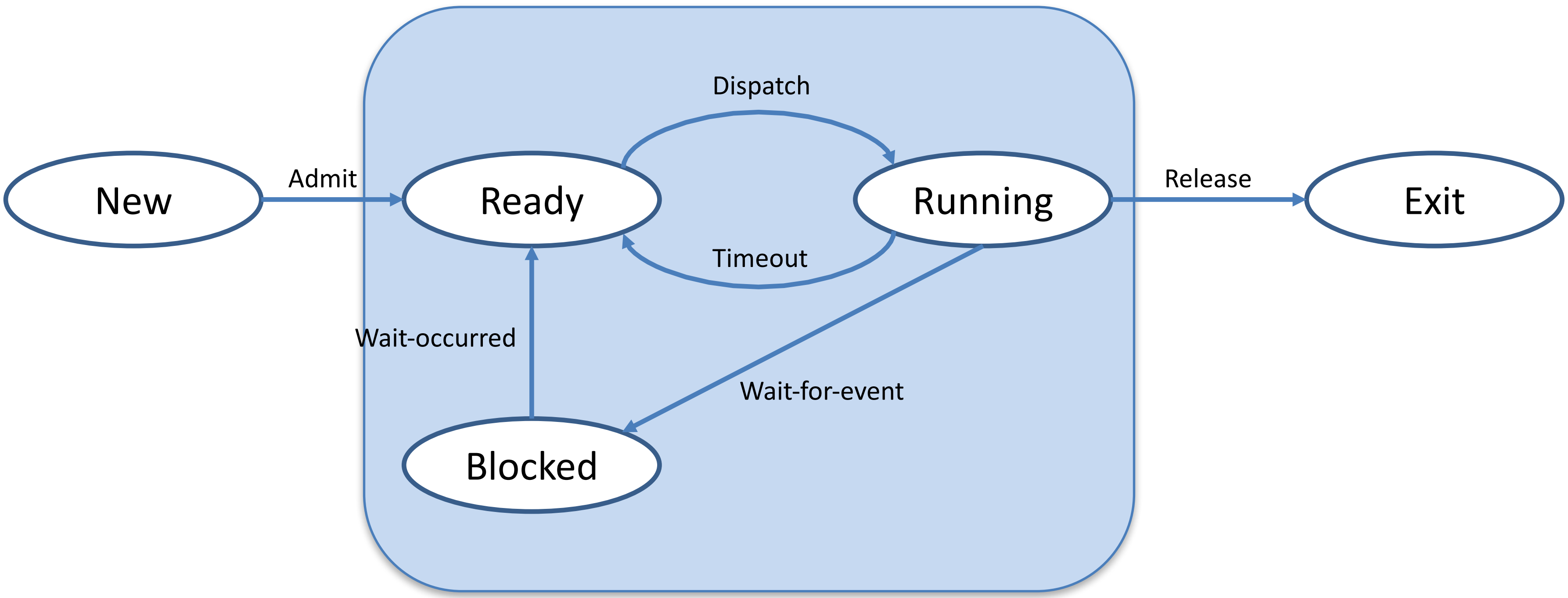
Process Execution (Dispatch)



1. Process blocked for I/O
2. Dispatcher schedules another process
3. Dispatcher interrupts process because its time slice expired
4. Input becomes available, blocked process made ready

- We can distinguish three basic process states during execution
  - Running: actually using the CPU
  - Ready: being runnable, temporarily stopped (time-out) to let another process execute
  - Blocked/Waiting: unable to run until some external event happens, such as I/O completion

Five-State Process Model



Process Execution (Dispatch)

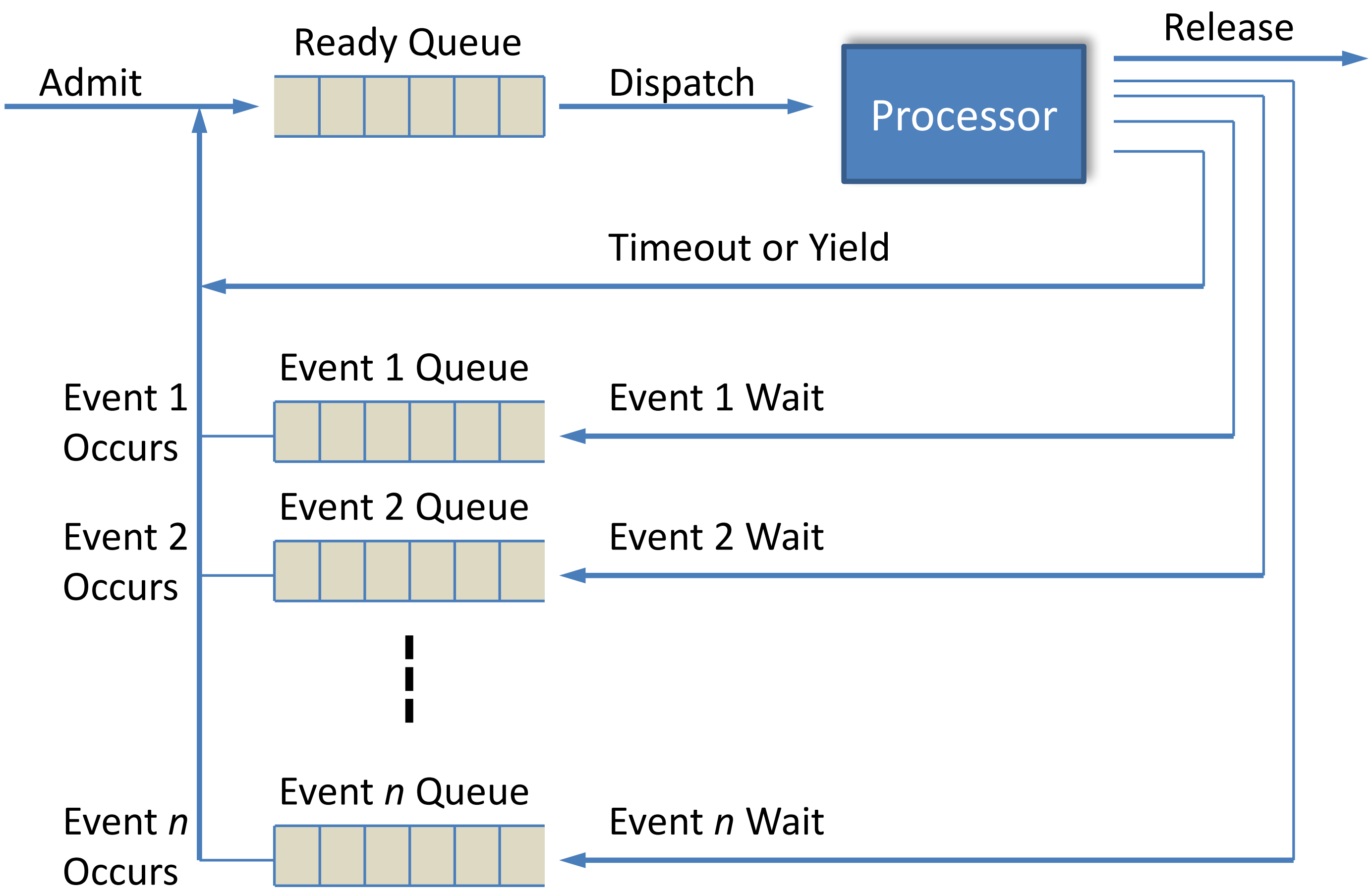
- Adding an ”admission” and “release” state of a process to the basic dispatch cycle of process execution

Five-State Process Execution Model

- ***New***: process is created, but not yet admitted to the pool of executable processes by the operating system
    - Process Control Block created
    - Program may not be loaded yet into main memory
- ***Ready***: process is prepared for execution and waits to be assigned to a processor
- ***Running***: process is executing
- ***Blocked***: a process waits for some event to occur (such as the reception of a signal or the completion of an I/O operation)
- ***Exit***: process is released from pool of executable processes, because it halted or aborted

Process Management

- Multiple Event Queues



Process Scheduling / Dispatch Queues

- Operating system maintains a set of queues to manage process scheduling
  - Job queue: set of all processes in the system
  - Ready queue: set of all processes residing in memory and ready for execution
    - Memory allocated for process image
  - Queues for waiting processes
    - Either one Blocked queue or different queues for processes waiting for particular events
      - E.g.: Device queue: processes that wait for an I/O device to become available
- Processes migrate between these queues

Dispatch Events

- When does a context switch occur?
- Clock interrupt, occurs after a specified time interval, usually 3-10ms
  - Execution of processes interrupted, control goes back to operating system
  - Process added to the ready queue
  - Frequency of such an interrupt important system parameter
    - Balance overhead of context switch vs. responsiveness

Dispatch Events

- I/O Interrupt
  - I/O interrupt occurs, when I/O action has occurred and data is loaded into memory
  - Currently executing process (Running state) is interrupted
  - All blocked processes waiting for this I/O action to be completed are moved into the Ready queue

Dispatch Events

- Memory fault / Page fault
  - Executing process refers to a virtual memory address that is not allocated a physical memory location (data still on hard disk)
  - Currently executing process (Running state) is interrupted
  - I/O request for bringing in data from secondary storage is issued
  - Currently executing process in Running state is switched to blocked
  - Switch to another process of the Ready queue
  - When I/O completed, blocked process moved back to Ready queue

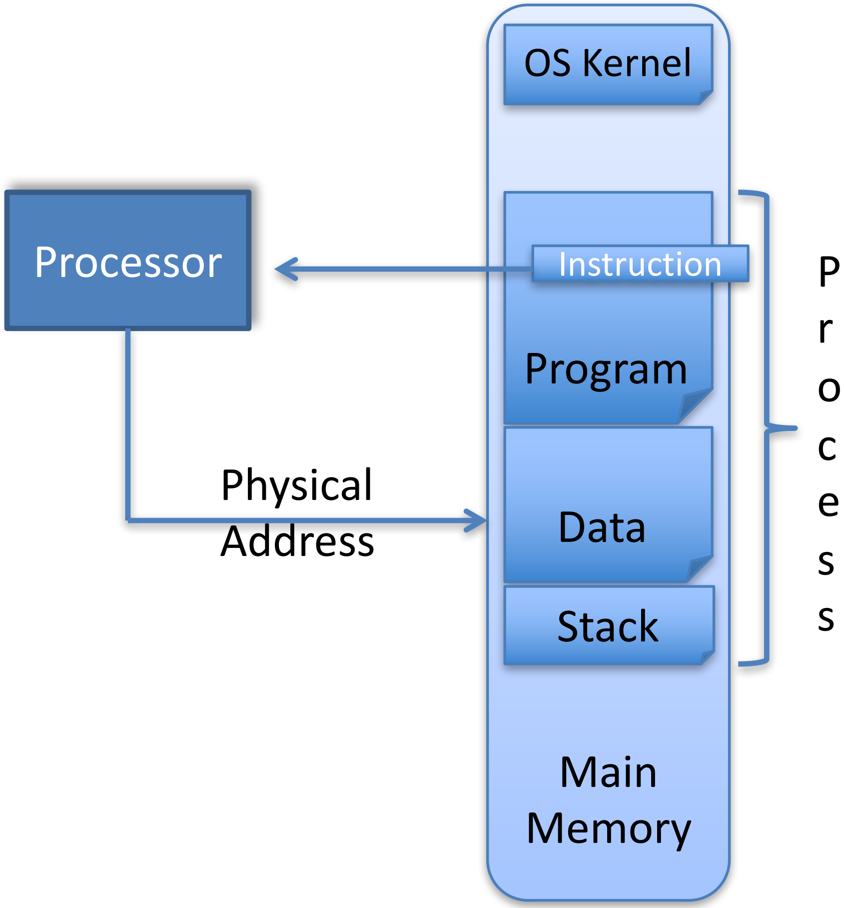
Process Memory

Virtual Memory Management

- We want to execute processes that are larger than actual physical memory
- We want to run multiple processes concurrently
- Problem of size
- Solution: Virtual memory management

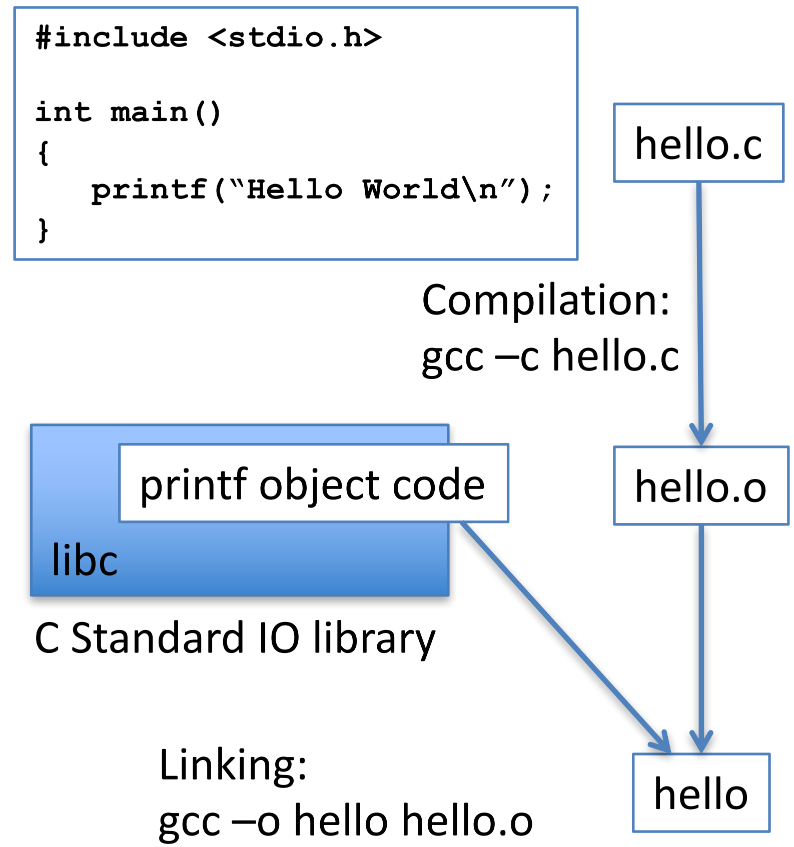
Program Execution

- Programs need memory for execution
  - Programs have to be loaded by the operating system into physical memory for execution
- During execution, programs manipulate data
  - Program instructions refer to memory addresses
    - Load data into register, or write register back to memory
- Program code has to contain information about memory locations



Problem: Addressing Memory

- A programmer cannot know in advance where in memory a program will be loaded for execution
- Compiler produces code that refers to memory locations (which ones?)
- Linker combines pieces of a program (with libraries) into a loadable and executable “image”
- Where in memory will the program reside?
- Which memory locations will it address?

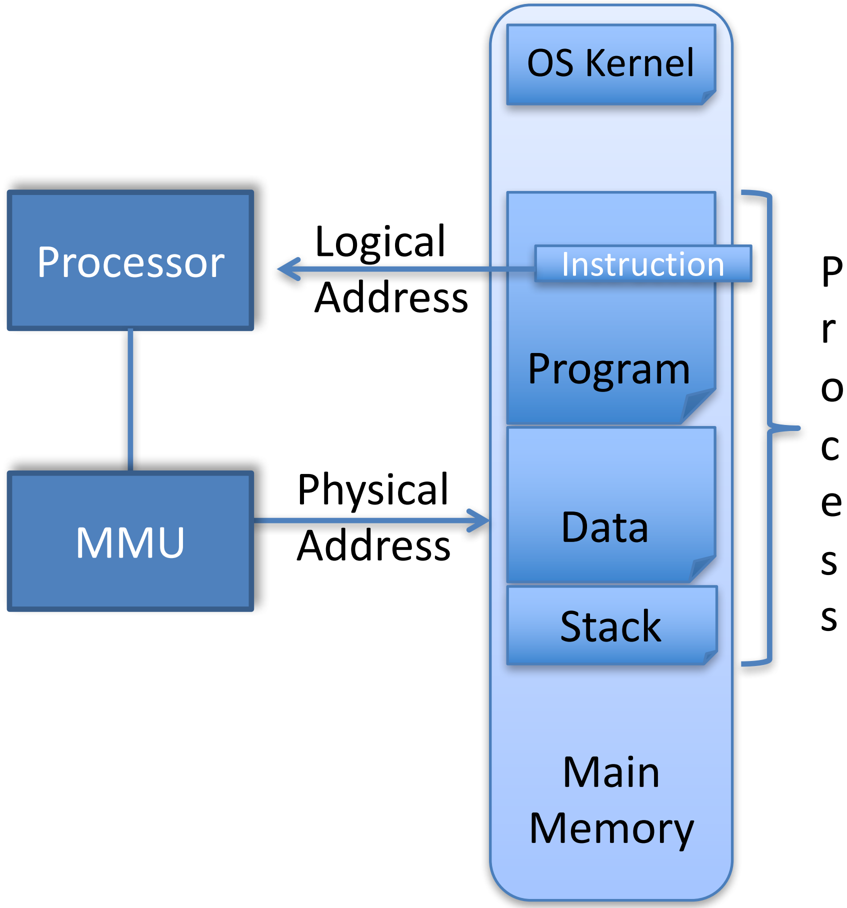


Program Relocation

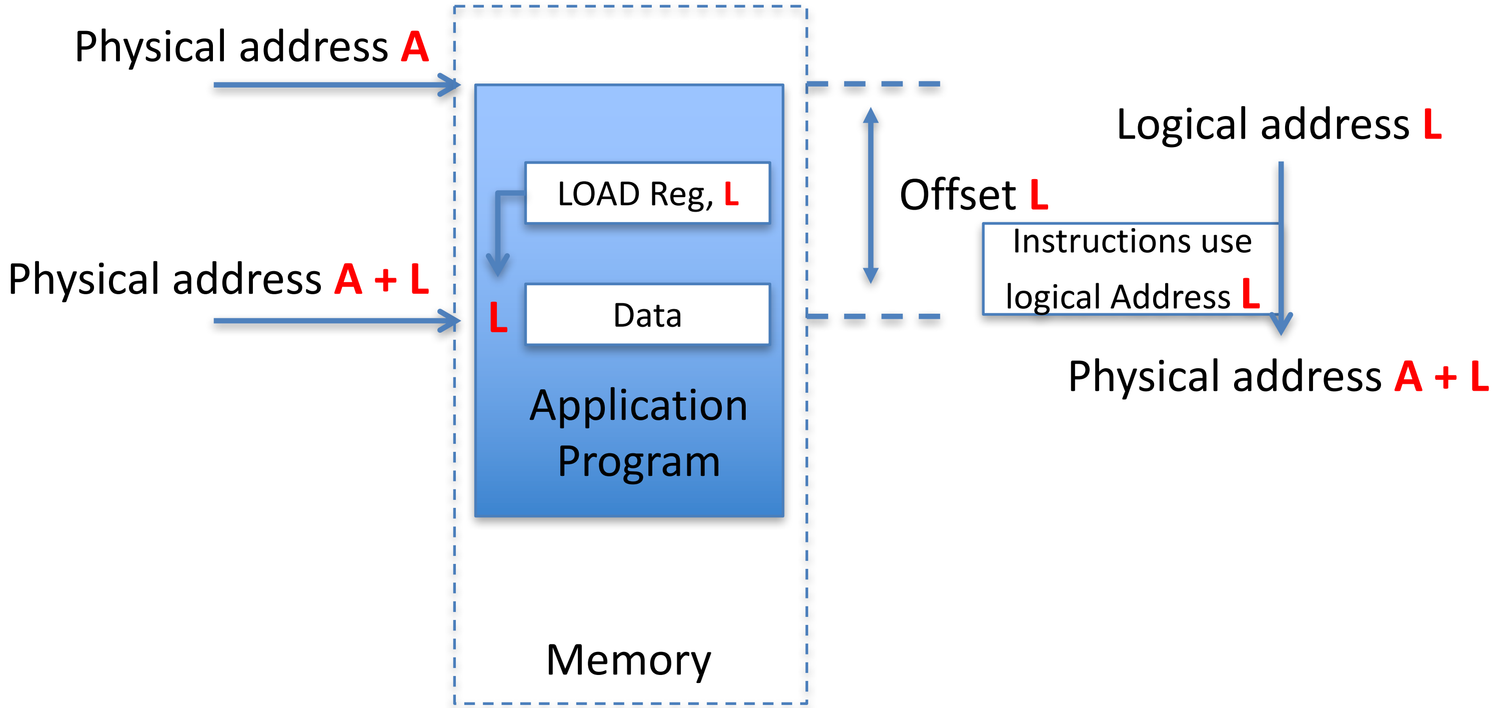
- A compiled program should be independent of physical memory locations
  - Programs have to be executable at any location in memory
  - Program instructions refer to memory addresses
    - Load data into register, or write register back to memory
- With dynamic schemes, programs can be relocated in memory
- Addresses used in a program cannot be actual physical addresses
  - makes program relocation impossible
  - Use relative addressing: addresses used in program are offsets relative to the program's start location

Logical vs Physical Addressing

- Programs should be able to reside at any location in memory
- This is achieved by a **virtual** or "relative" addressing scheme, that calculates a "physical" memory address from the program-specific "logical" address
- Logical address (also called "virtual" or "relative" address)
  - A value that specifies a generic location relative to the start of the program
- Physical address
  - The actual address of a location in the physical memory of a computer system



Relative Addressing



- Solution:
  - Programs operate as if they all start at address 0
- Programs operate with relative addressing mechanism:
  - Reference to memory in program independent of actual physical location
  - Addresses in program are offsets from its base address (the starting address of the process image)
  - add the base address whenever there is an access to memory

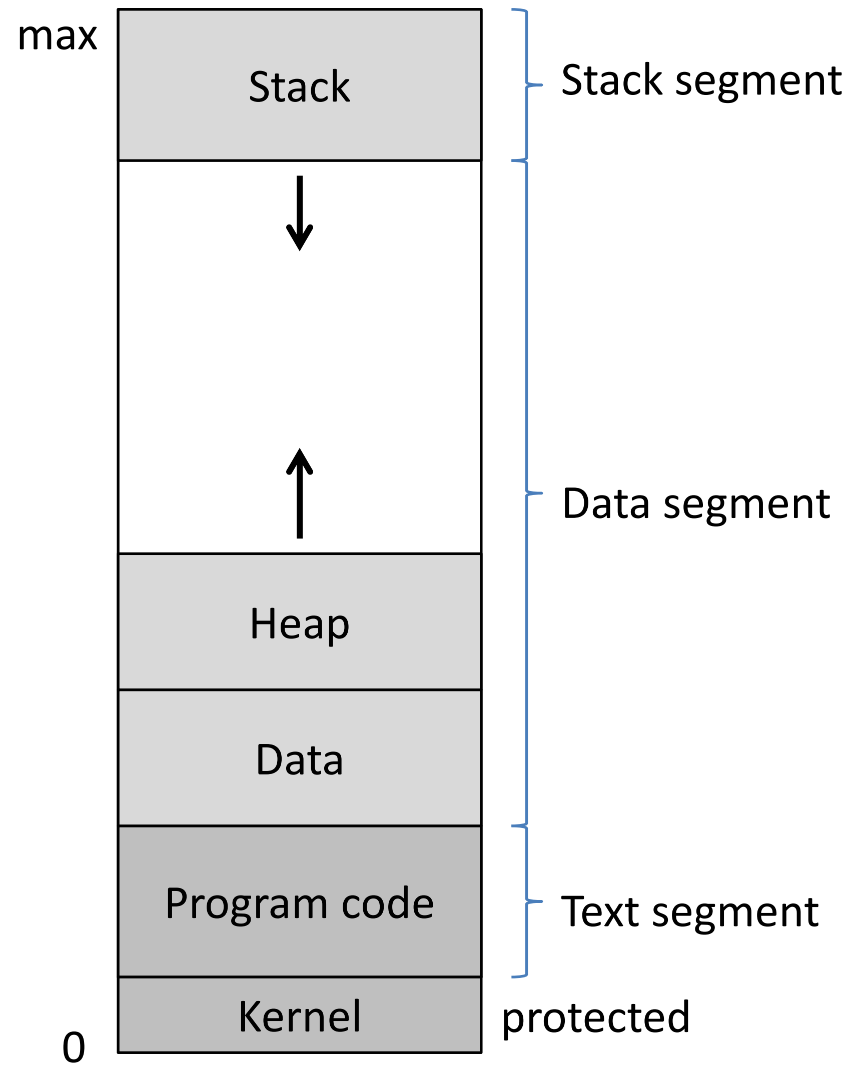
Address Space

- Programs operate within the address space of the processor, uses logical or “virtual” addresses
  - All (theoretically) addressable memory locations
  - Depends on size of address register
    - 32-bit architecture: 4GB of addressable memory locations
    - 64-bit architecture: 16ExaBytes of addressable memory locations

Bytes	Exponent			How to calculate
1,024	2^{10}	1kb	1024bytes	
1,048,576	2^{20}	1MB	1024kb	1024 x 1024
1,073,741,824	2^{30}	1GB	1024MB	1024 x 1024 x 1024
4,294,967,296	2^{32}	4GB	4 x 1024MB	4 x 1024 x 1024 x 1024
1,099,511,627,776	2^{40}	1TB	1024GB	1024 x 1024 x 1024 x 1024
1,125,899,906,842,620	2^{50}	1PB	1024TB	1024 x 1024 x 1024 x 1024 x 1024
1,152,921,504,606,850,000	2^{60}	1EB	1024PB	1024 x 1024 x 1024 x 1024 x 1024 x 1024
18,446,744,073,709,600,000	2^{64}	16EB		16 x 1024 x 1024 x 1024 x 1024 x 1024 x 1024

Process Image

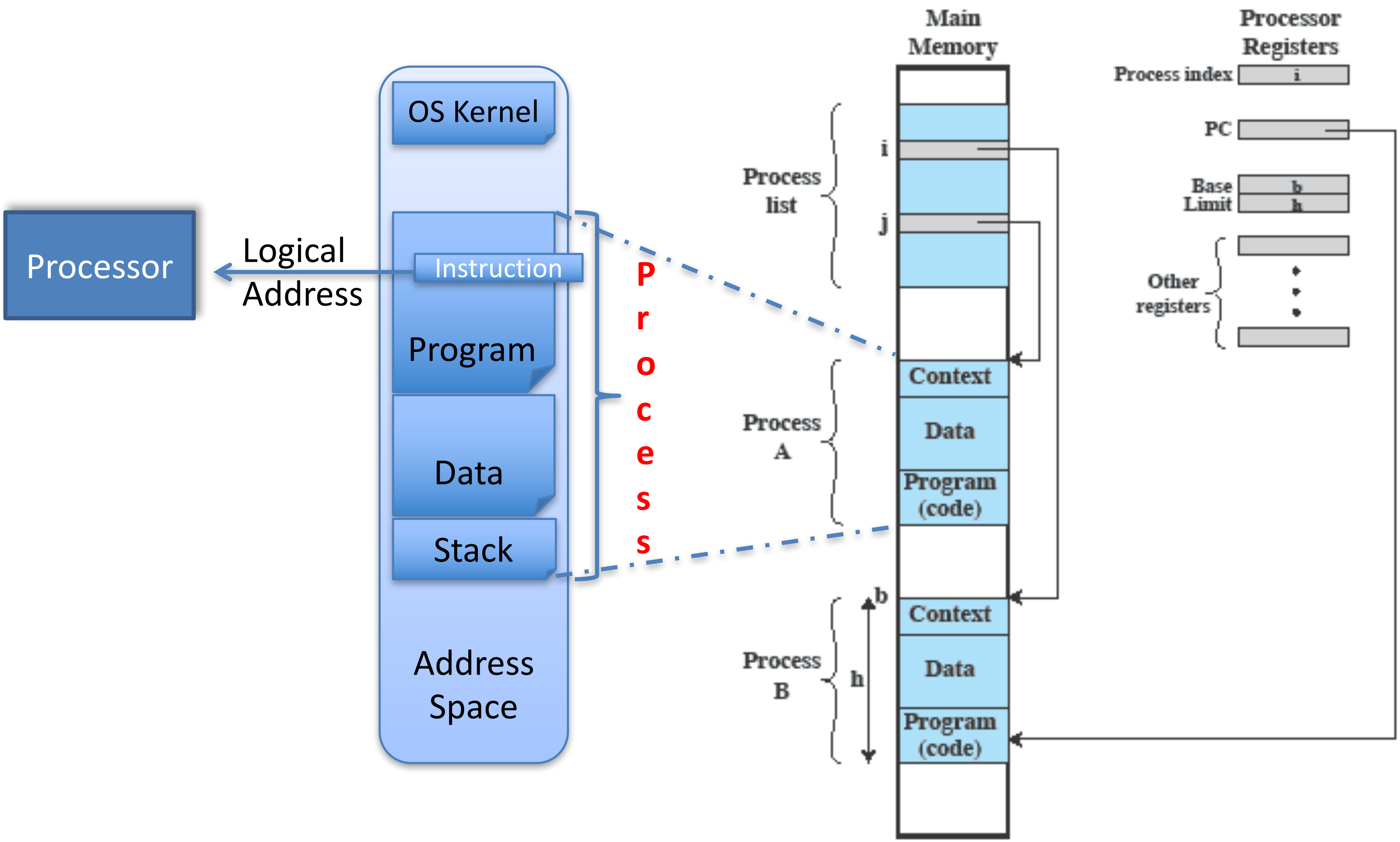
- Process Image
  - Layout of process in virtual memory
- Segments
  - Stack segment
    - Used for function calls
  - Data segment
    - Static variables, constants
    - Dynamic allocation of memory from the heap
  - Text segment
    - Contains the program code, shared between processes



Virtual Address Space

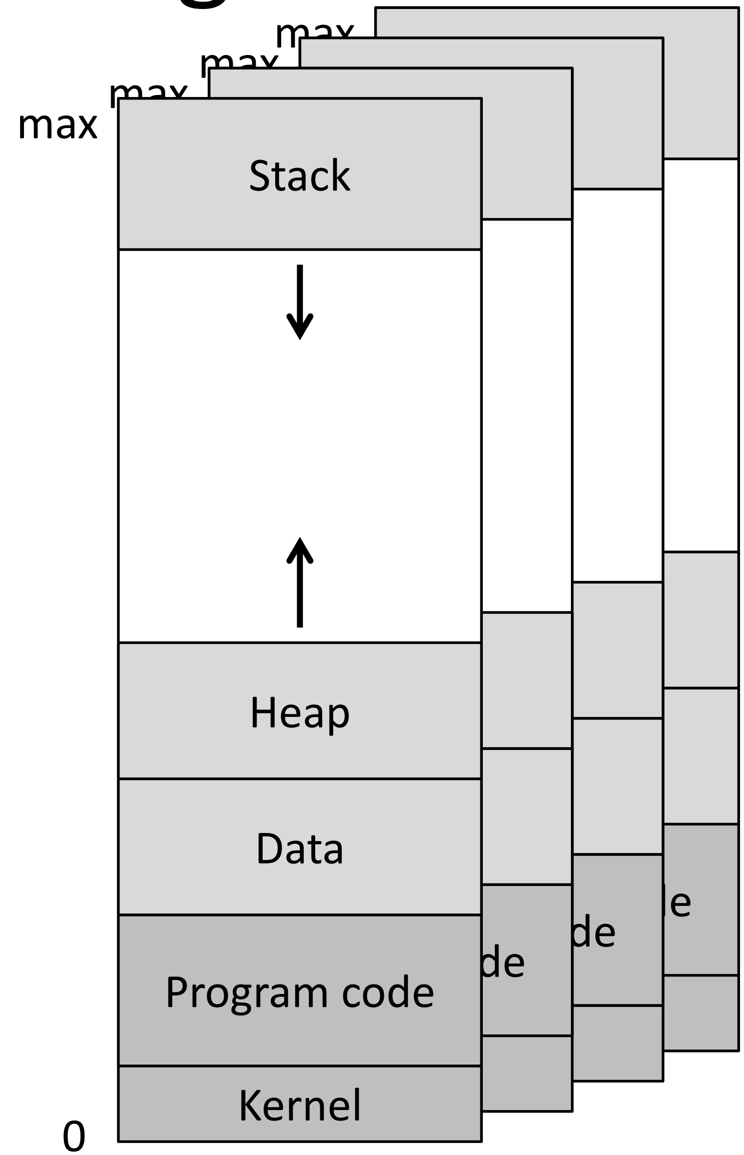
- Process image appears to be a huge contiguous memory area
  - Starts at address 0, up to the maximum possible memory address
  - Each of the segments starts at a particular address
- Virtual address space
  - Abstraction of physical memory
  - Independent of the actual size of physical memory
- Mechanisms needed to map addresses in the process image to actual physical memory locations

Address Translation



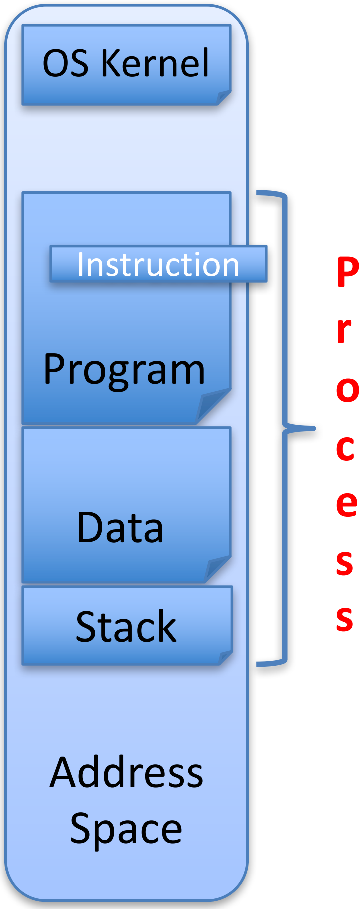
Virtual Memory Management

- Each program invoked results in the creation of a separate process, each with its own process image
- We want to execute processes that are larger than actual physical memory
- We want to run multiple processes concurrently
- Problem of size
- Solution:
  - Virtual memory management
    - Use persistent storage to complement the lack of physical memory
  - Scheduling / context switch
    - Run processes concurrently by switching between process images



Segmentation

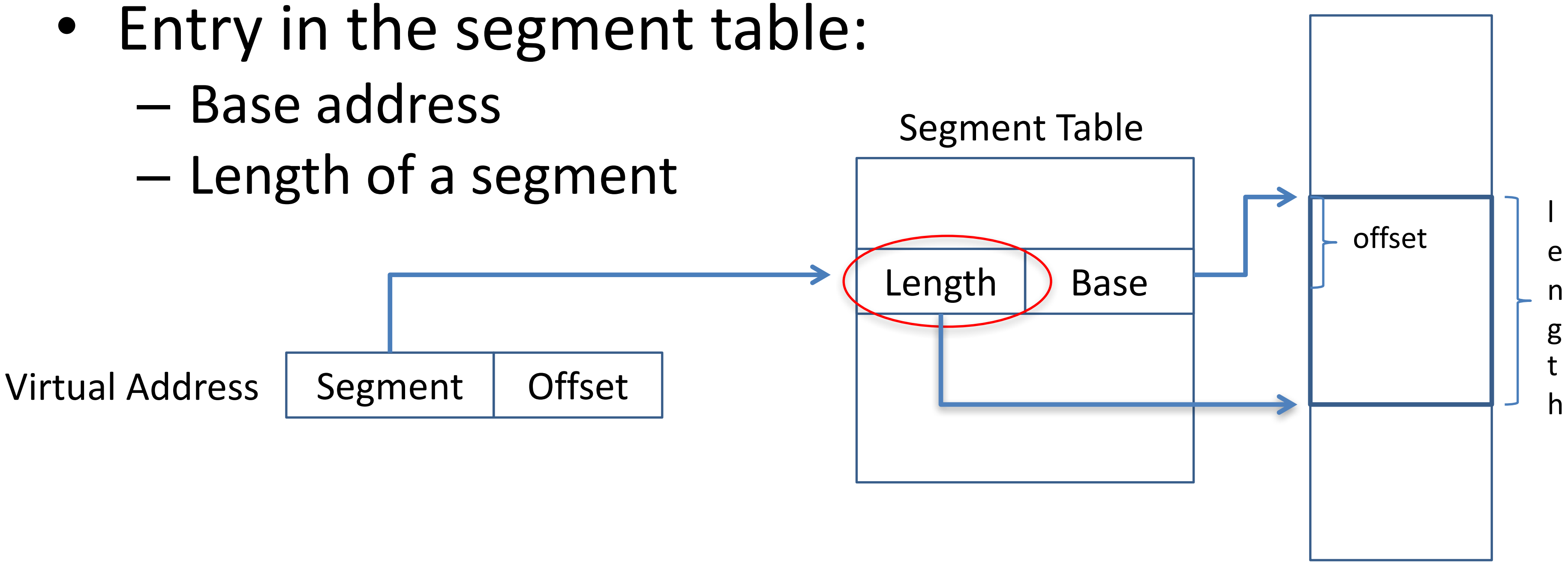
- Segments allow Programs to be divided into logical units of variable length
  - Process image regarded as a collection of segments
    - Code segment
    - Data segment
    - Stack, Heap, shared libraries etc.
- Each segment is a separate address space
  - Has a base address where the segment starts, memory within a segment is addressed relative to this base address
  - Allows parts of a program code to be altered and recompiled independently
    - Shared libraries
- Segments are visible to programmer
- In programs, a "virtual address" to a memory location consists of two parts: starting location of the segment and offset within the segment



Virtual Address	Segment ***i***	Offset

Segmentation

- Each process has a segment table
  - Each segment has a base address
  - Segments are of variable size
    - Segment table entry has to specify the **length** of a segment
  - Held in the process control block
- Entry in the segment table:
  - Base address
  - Length of a segment



Virtualization
Memory Management