# Operating Systems Overview

## Operating System

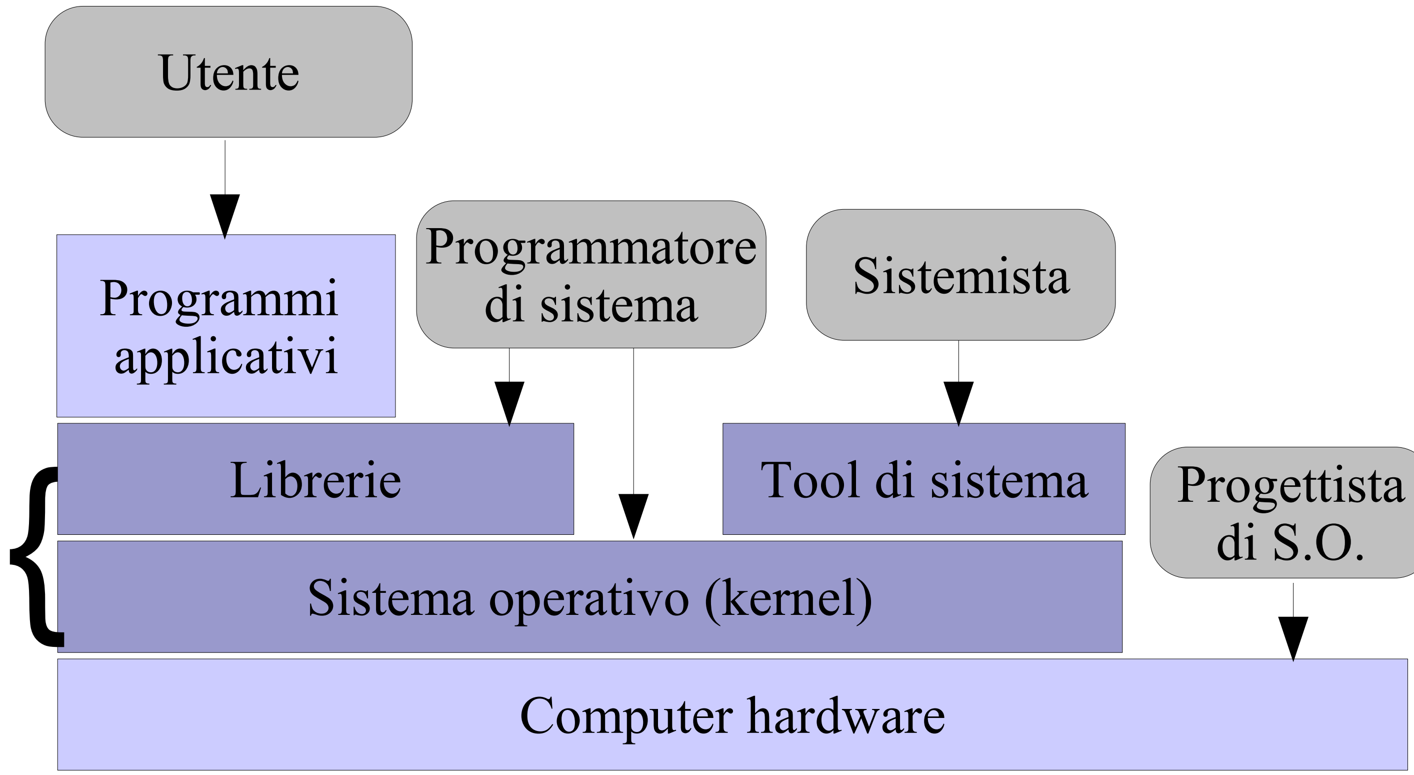
- A program that controls the execution of application programs
- An interface between applications and hardware
- A set of programs that provides basic functionalities

# Operating System Objectives

- Convenience
  - Makes the computer more convenient to use (for programmers and users)
- Efficiency
  - Allows computer system resources to be used in an efficient manner
- Ability to evolve
  - Permit effective development, testing, and introduction of new system functions without interfering with service

not on
the book

# Layers of Computer System



# Operating System as a Resource Manager

- **resource**: anything needed for program execution, e.g.
  - cpu time
  - I/O devices
  - memory
  - executable code
  - etc.
- an os manages resources

## services provided by the os

- services for users
  - program execution
  - security (user login, user confinement)
  - support for program development
  - accounting
  - error detection and response
- services for programs (or programmers)
  - resource management (es. memory and cpu time)
  - access to I/O devices (es. files)

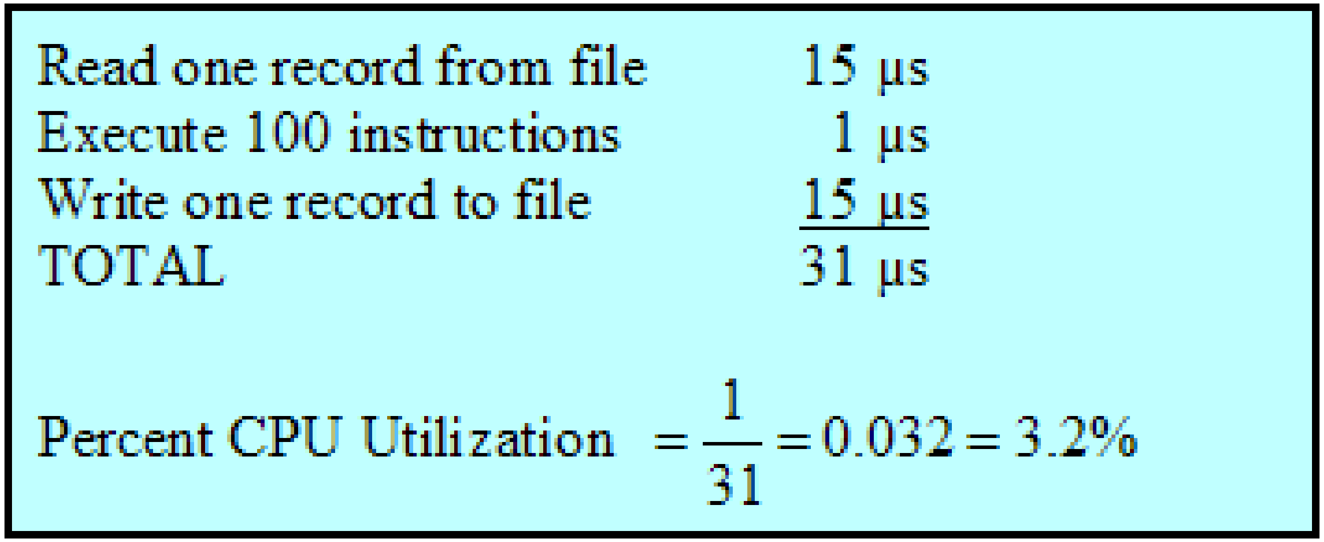
## Kernel

- Portion of operating system that is in main memory
- Contains most frequently used functions
- Also called “nucleus”

## User/Kernel mode

- User program executes in **user mode**
  - Certain *privileged* instructions may not be executed
  - only a ristricted part of main memory can be accessed (**user space**)
- Kernel executes in **system mode**
  - a.k.a. **kernel mode** or **supervisor mode**
  - Privileged instructions can be executed
  - Protected areas of memory may be accessed (**kernel space**)

## I/O Devices are Slow



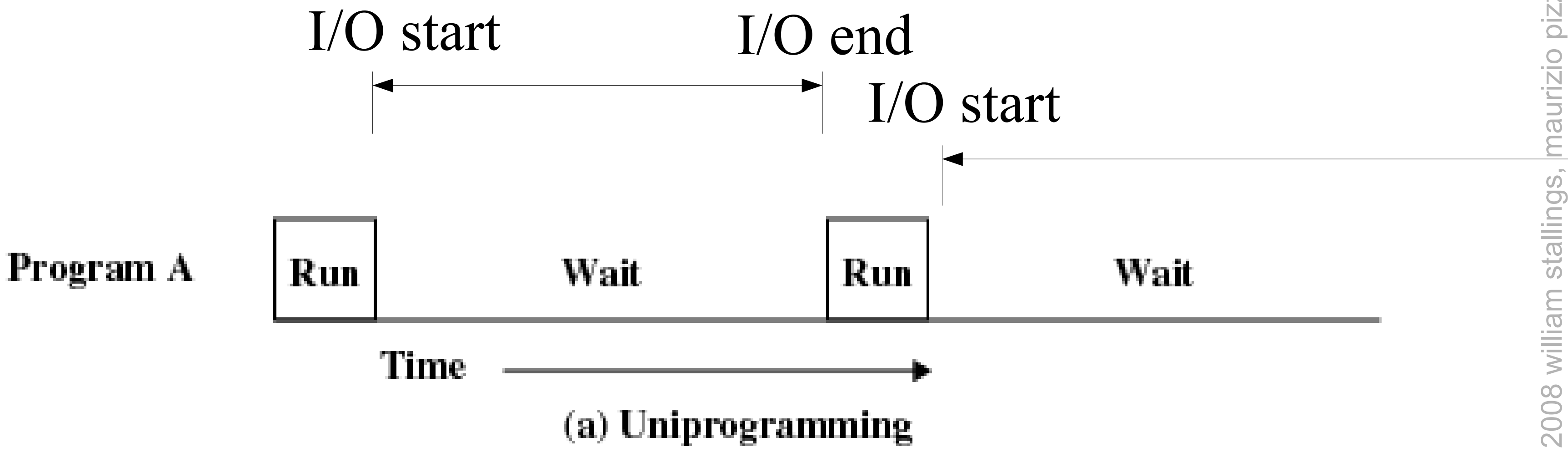
| | |
|---|---|
| Read one record from file | 15 µs |
| Execute 100 instructions | 1 µs |
| Write one record to file | 15 µs |
| TOTAL | 31 µs |

$$\text{Percent CPU Utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

**Figure 2.4 System Utilization Example**

# Uniprogramming

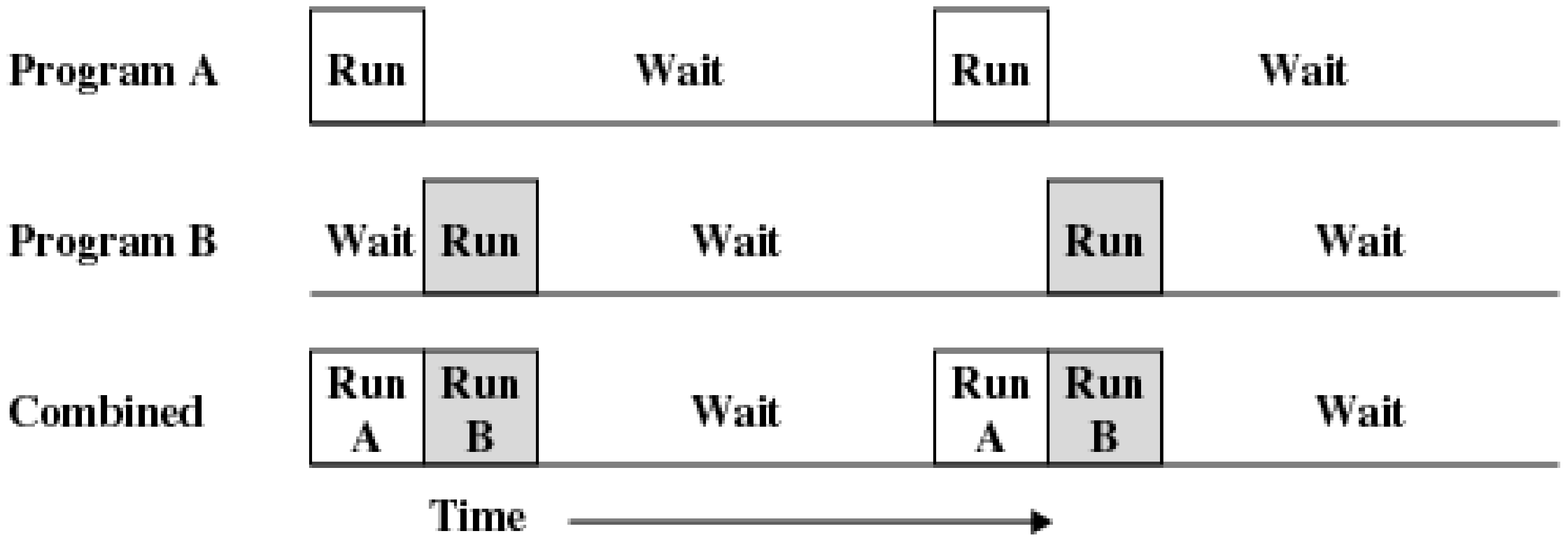
- Processor must wait for I/O instruction to complete before preceding



(a) Uniprogramming

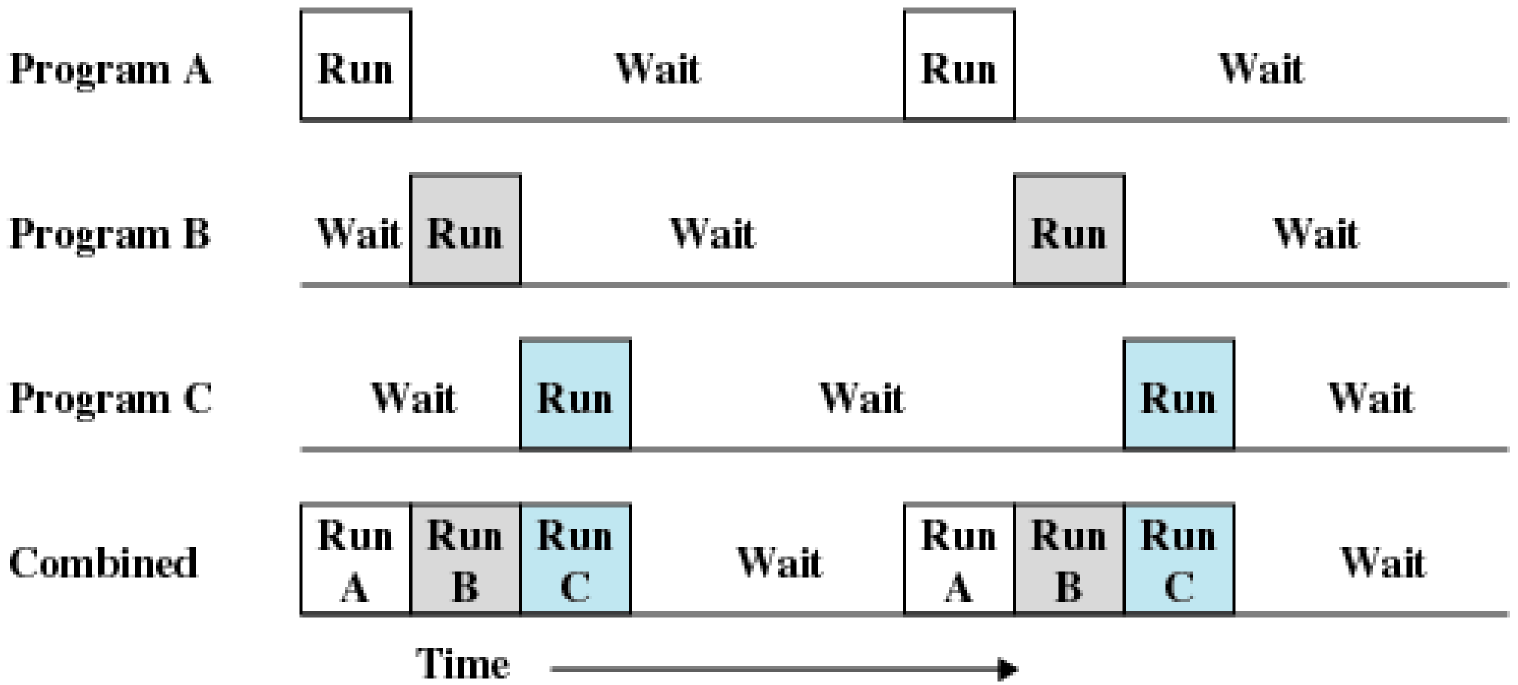
## Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs

# Multiprogramming

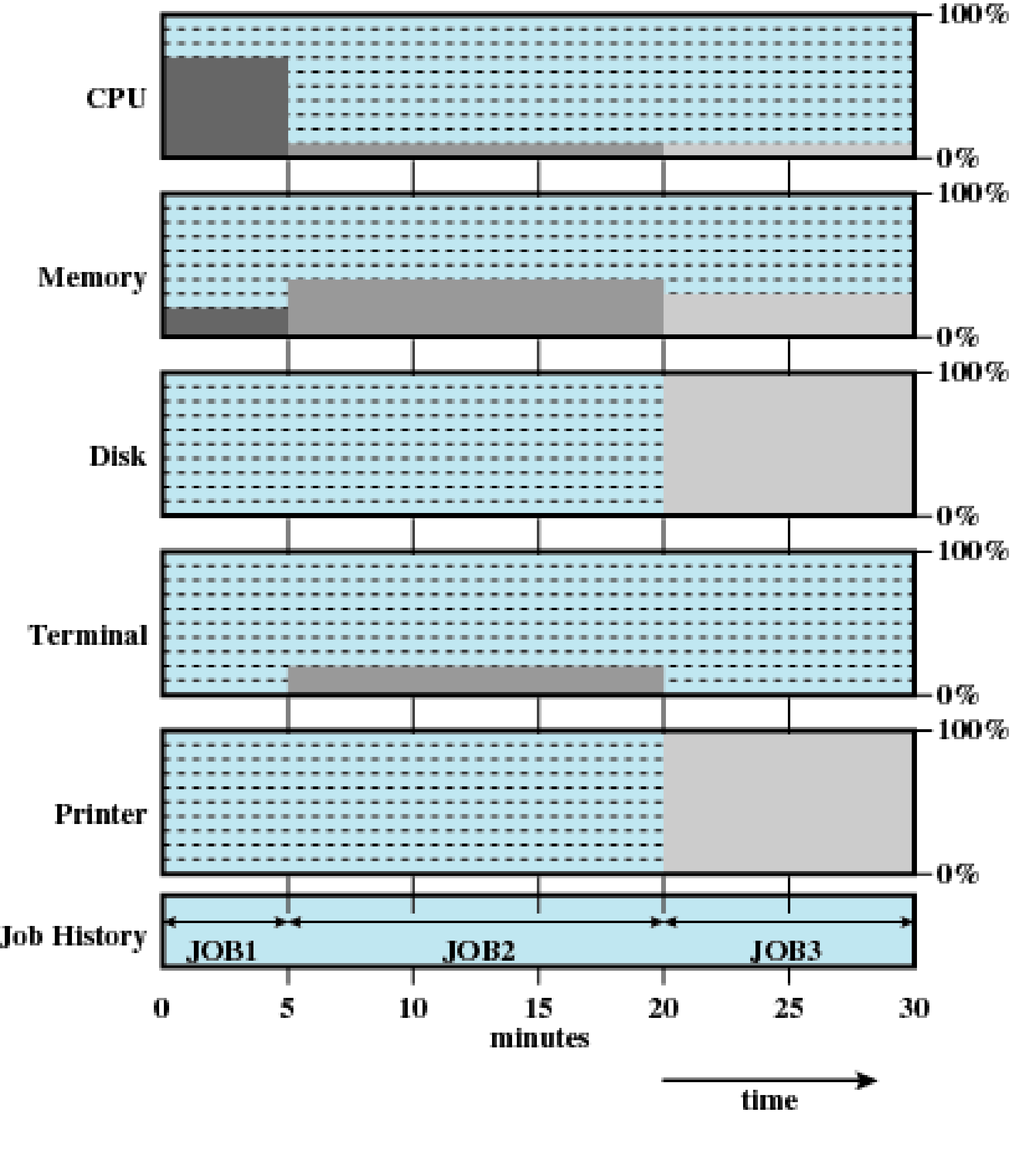


(c) Multiprogramming with three programs

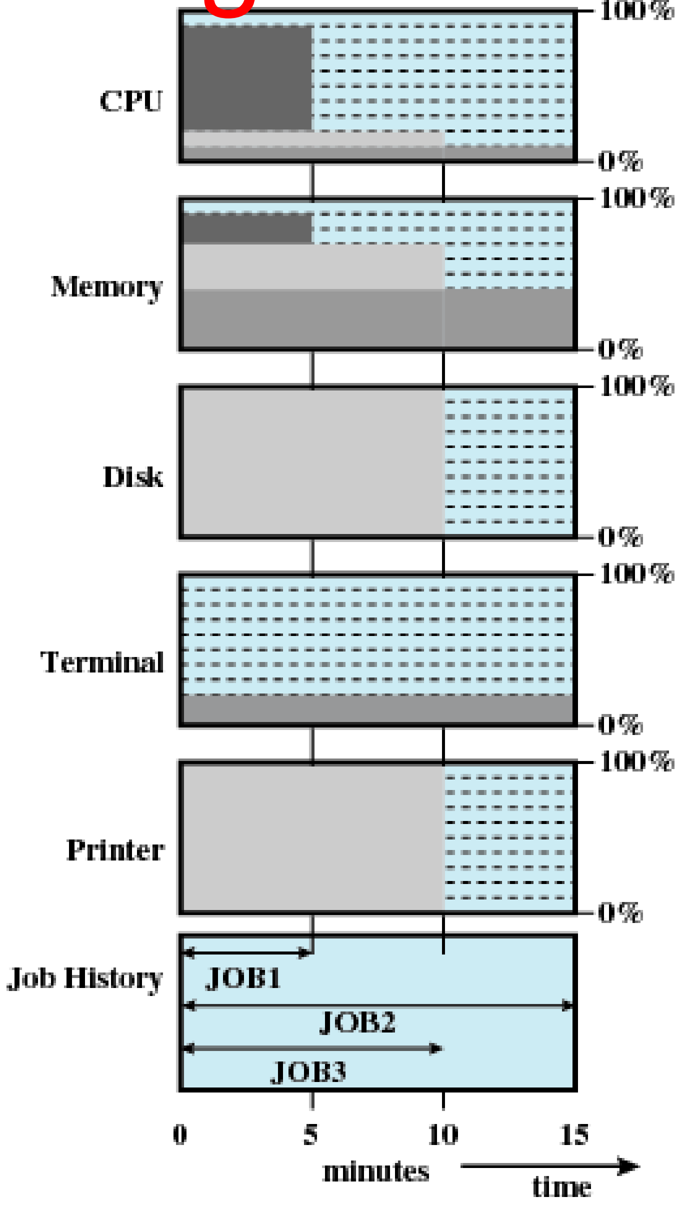
# I/O bound vs. CPU bound processes

| | JOB1 | JOB2 | JOB3 |
|---|---|---|---|
| **Type of job** | Heavy compute | Heavy I/O | Heavy I/O |
| **Duration** | 5 min | 15 min | 10 min |
| **Memory required** | 50 M | 100 M | 75 M |
| **Need disk?** | No | No | Yes |
| **Need terminal?** | No | Yes | No |
| **Need printer?** | No | No | Yes |
| | CPU bound | I/O bound | I/O bound |

# Utilization Histograms



(a) Uniprogramming



(b) Multiprogramming

# Time Sharing

- Using multiprogramming to handle **multiple interactive jobs**
- Processor's time is shared among multiple users
- Multiple users simultaneously access the system through terminals

## Major Achievements of Modern OSes

- Processes
- Memory Management
- Information protection and security
- Scheduling and resource management
- System structure

△
not on
the book

# Processes

- A program in execution (running) on a computer
- A unit of activity characterized by
  - an associated set of system resources
    - memory regions
    - open files
    - etc.
  - at least one execution **thread** with its current state of CPU

△
not on
the book

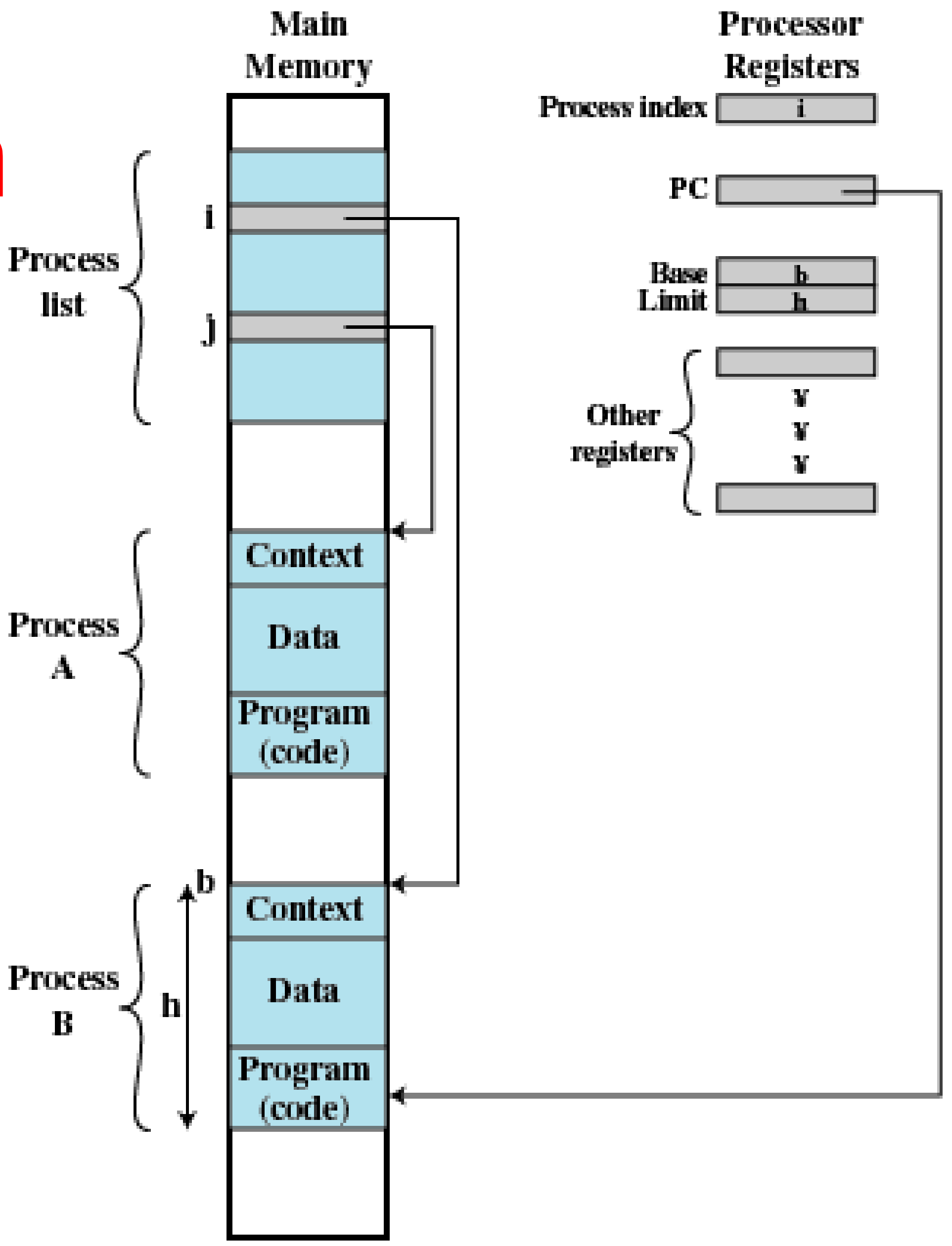
## Threads

- The entity that can be assigned to and executed on a processor
  - concept meaningful only within a process
  - described by
    - the value of the program counter
    - the value of the CPU regisers
- In modern operating systems a process may contains or more thread
- we always assume it contains one thread

## execution context

- cpu registers
- priority of the process
- is the process waiting for I/O? on which device?
- etc.
- etc.
- etc.
- etc.
- ...

# Processes Representation



## Memory Management

- Process isolation
- Automatic allocation and management
- Support of modular programming
- Protection and access control
- Long-term storage

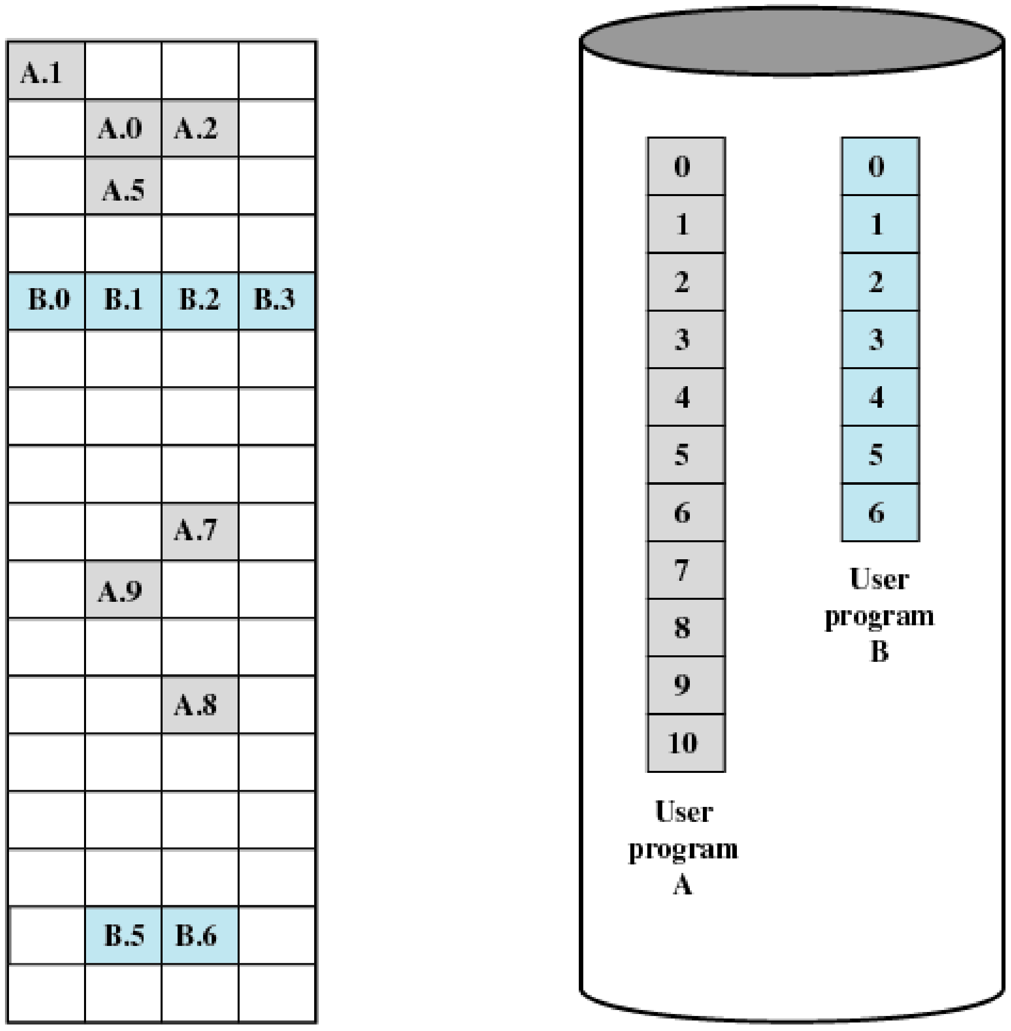
## Virtual Memory

- Allows programmers to address memory from a logical point of view
- Virtual memory is much larger than Real Memory
  - processes see a large virtual address space
- Real Memory is used only for processes that are in execution

# Paging

- Allows process to be comprised of a number of fixed-size blocks, called pages
- Virtual address is a page number and an offset within the page
- Each page may be located any where in main memory
- Real address or physical address in main memory are managed only by the kernel

## Virtual Memory and Main Memory



**Main Memory**

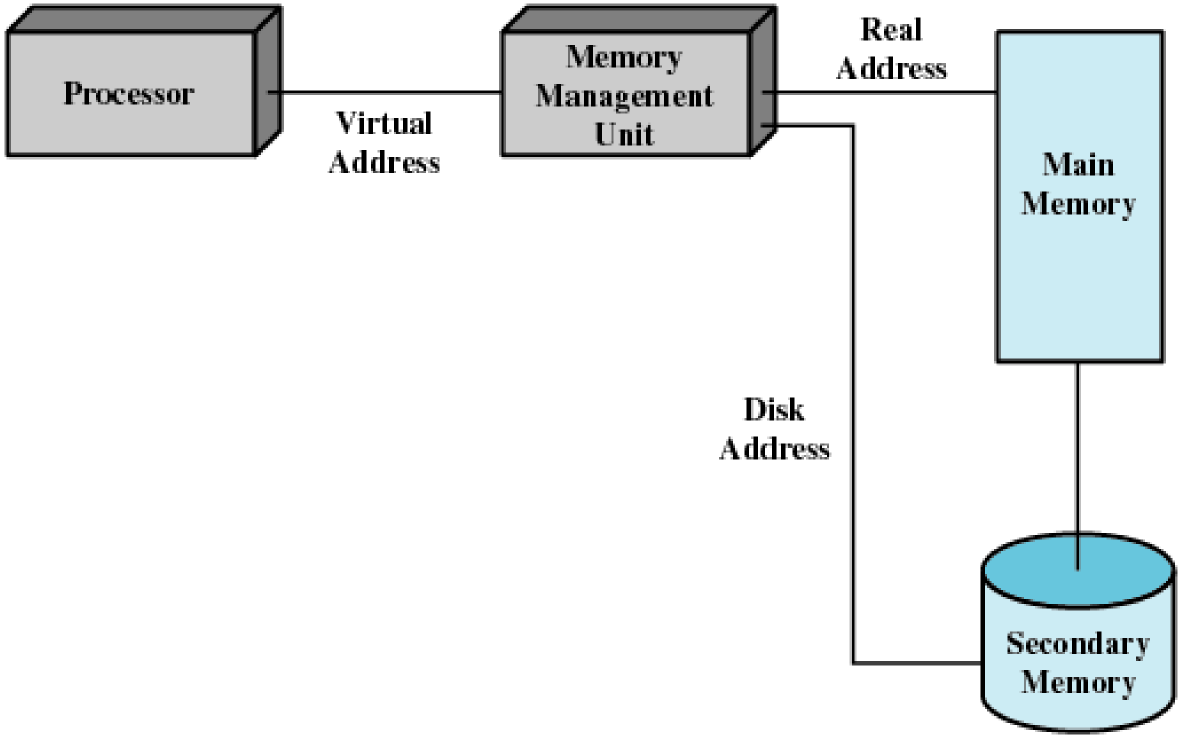
Main memory consists of a number of fixed-length frames, each equal to the size of a page. For a program to execute, some or all of its pages must be in main memory.

**Disk**

Secondary memory (disk) can hold many fixed-length pages. A user program consists of some number of pages. Pages for all programs plus the operating system are on disk, as are files.

# Virtual Memory Addressing

- Virtual is managed by
  - a Memory Management Unit
  - ...and the kernel



## Information Protection and Security

- Availability
  - Concerned with protecting the system against interruption
- Confidentiality
  - Assuring that users cannot read data for which access is unauthorized

## Information Protection and Security

- Data integrity
  – Protection of data from unauthorized modification
- Authenticity
  – Concerned with the proper verification of the identity of users and the validity of messages or data

## 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

## Scheduling Elements

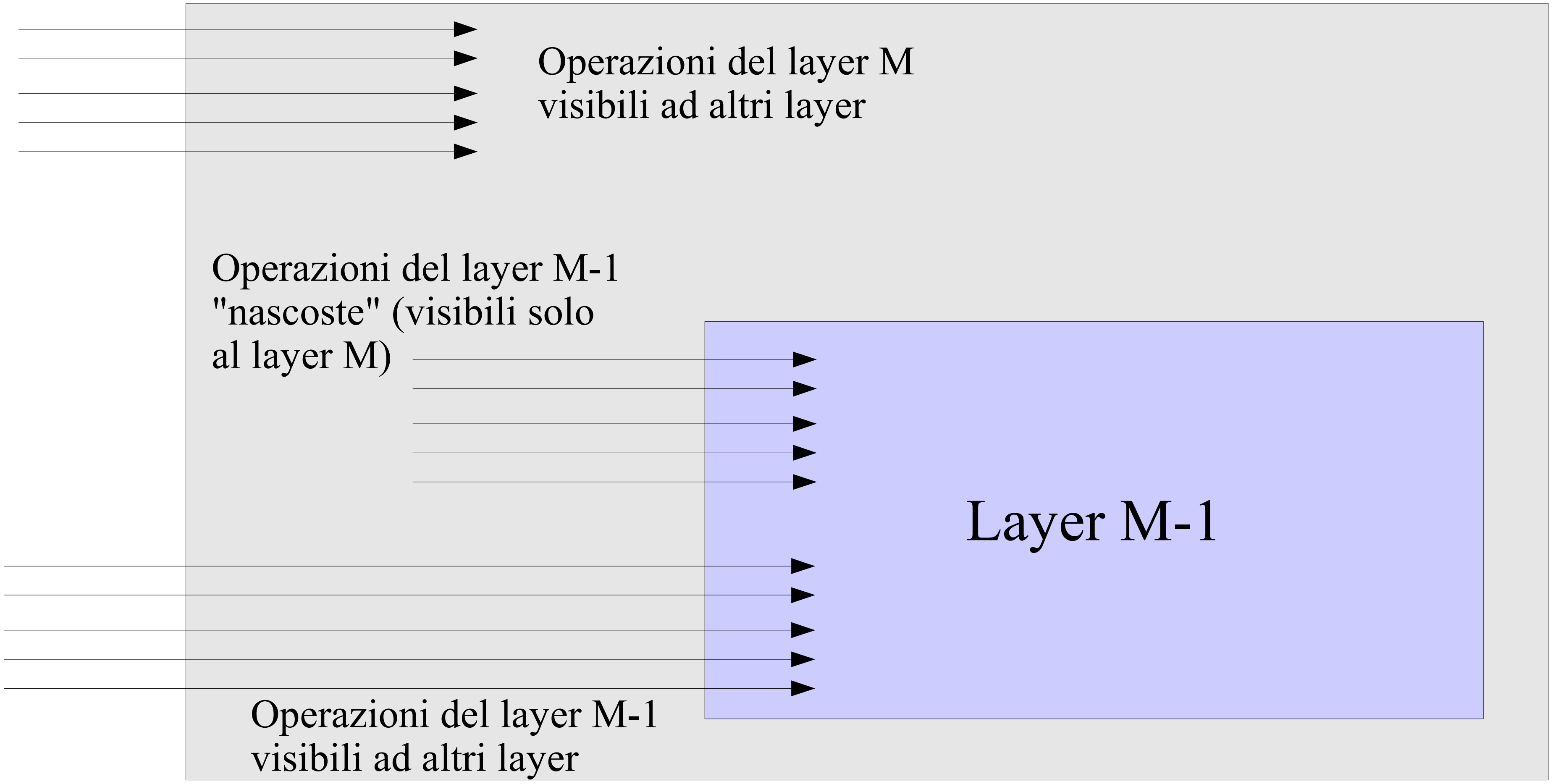
- queues
  - at least one for each resource
- CPU
  - short term
    - contains processes in main memory and ready to run
    - short term scheduler / dispatcher
      - simple approach: round robin (circular queue)
  - long term
    - new jobs waiting for the processor
    - long term scheduler
- I/O
  - at least queue for each device
  - interrupts

## System Structure

- View the system as a series of levels
- 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

not on
the book

# Layered Systems



## System Structure
*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

# System Structure
## *Hardware Levels*

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

## System Structure
### *Basic Multiprogramming and Memory Management*

- Level 5
  - Process management from the point of view of CPU (no I/O support)
  - 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

## System Structure
### *Process comunication, I/O, and Inter Process Comunication*

- 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

## System Structure

## *Process comunication, I/O, and Inter Process Comunication*

- Level 11
  - Responsible for maintaining the association between the external and internal identifiers
- Level 12
  - Provides full-featured facility for the support of processes

# System Structure
## *User Interface*

- Level 13
  - Provides an interface to the operating system for the user (shell)

# Modern Operating Systems

- Microkernel architecture
  - Assigns only a few essential functions to the kernel
    - Address spaces
    - Interprocess communication (IPC)
    - Basic scheduling
  - everything else is implemented as a process

# Modern Operating Systems

- Multithreading
  - Process is divided into threads that can run concurrently
    - Thread
      - Dispatchable unit of work
      - executes sequentially and is interruptable
    - Process is a collection of one or more threads

# Modern Operating Systems

- Symmetric multiprocessing (SMP)
  - There are multiple processors
  - These processors share same main memory and I/O facilities
  - All processors can perform the same functions