OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

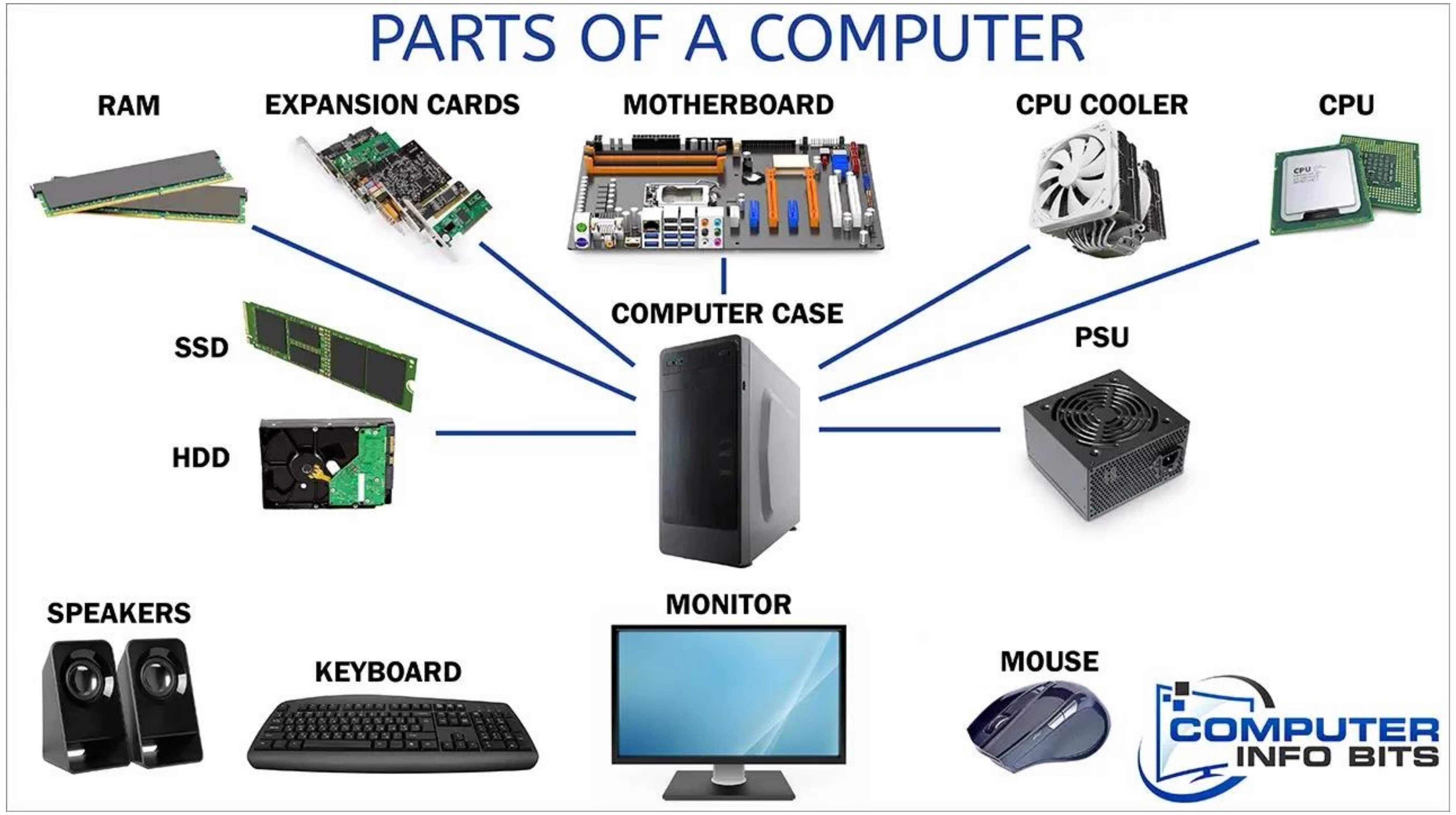
# CT101 Computing Systems

Dr. Bharathi Raja Chakravarthi
Lecturer-above-the-bar
Email: bharathi.raja@universityofgalway.ie

University
ofGalway.ie

# Hardware and Software

- Computer Hardware would include all the parts of a computer.
- Computer Software includes the programs that are installed.. Or the instructions for completing tasks.



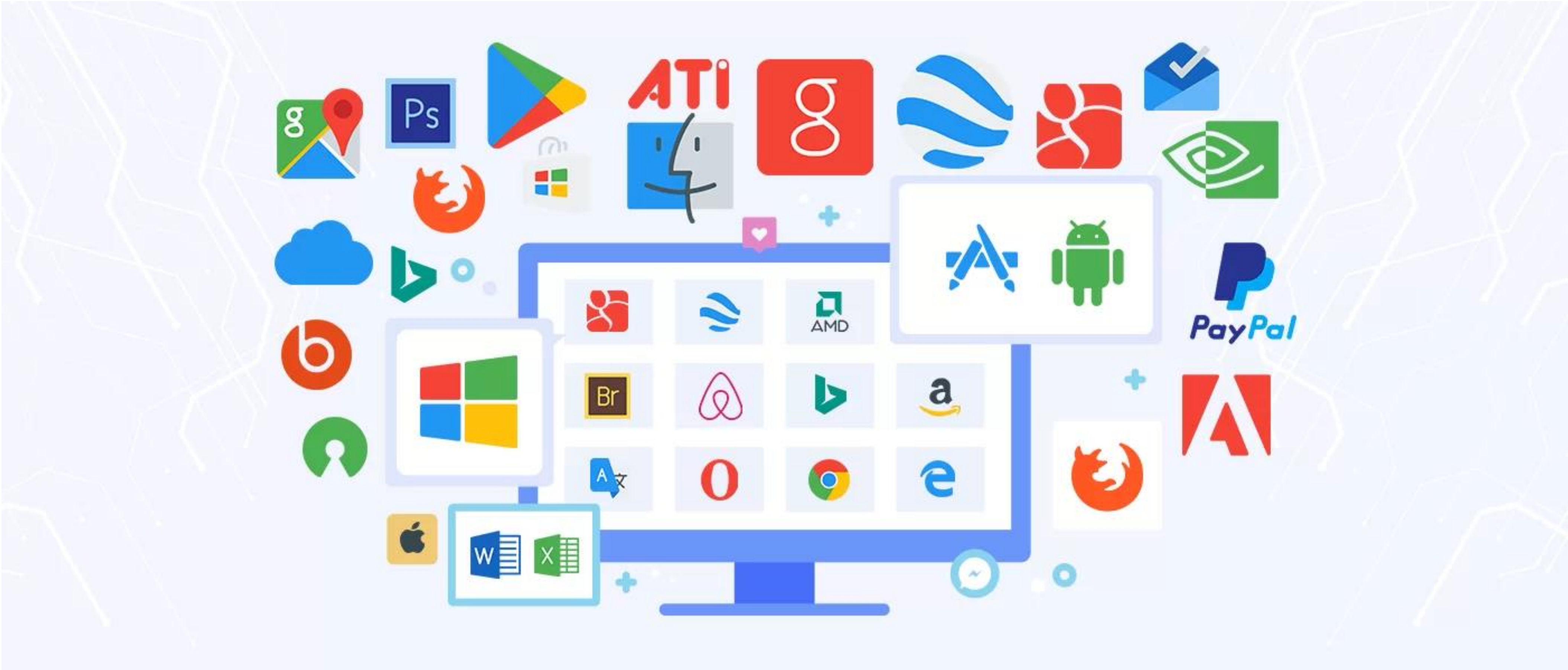


Image credits: https://computerinfobits.com/parts-of-computer-and-their-functions/
https://www.goodcore.co.uk/blog/types-of-software/

# Operating System

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

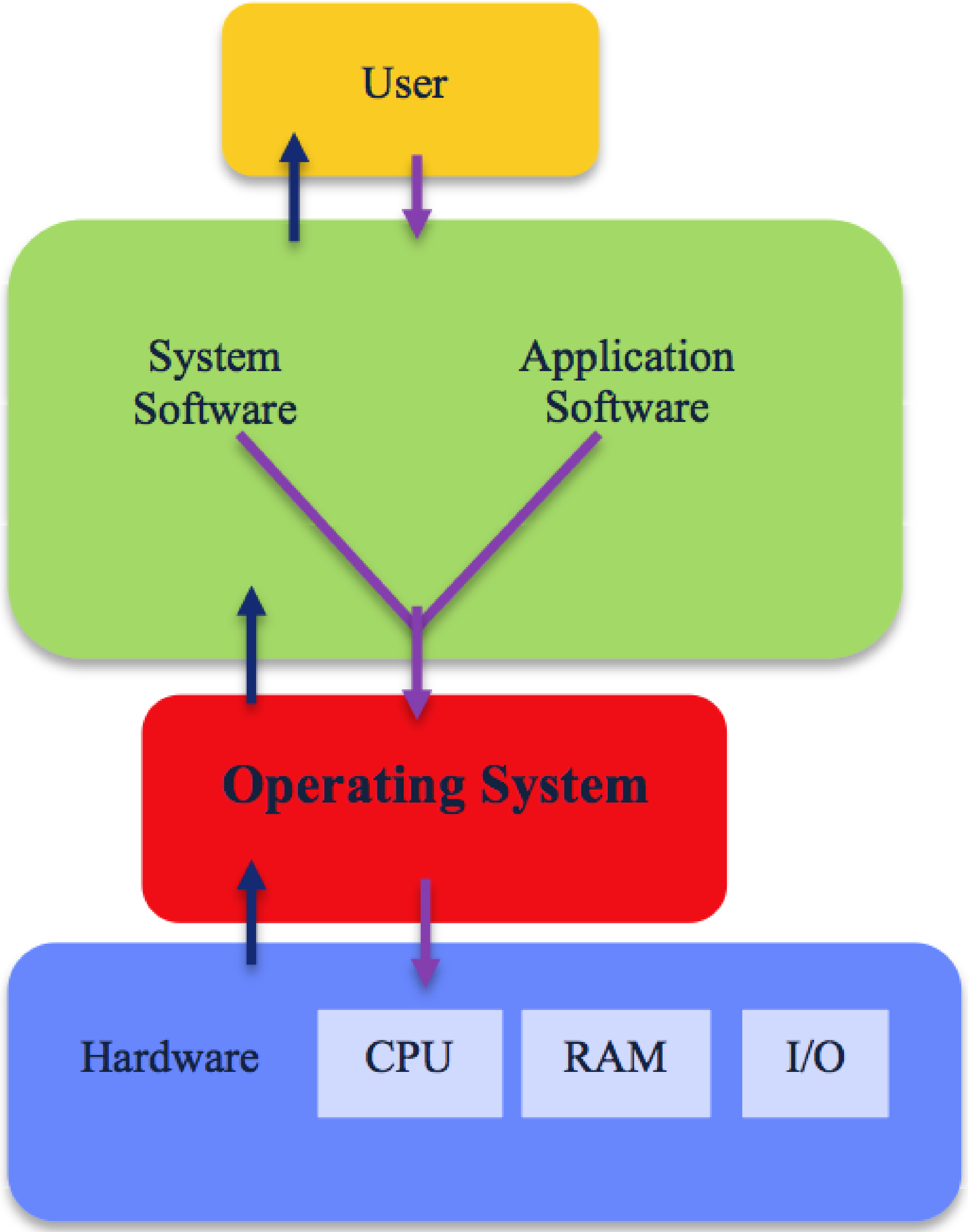
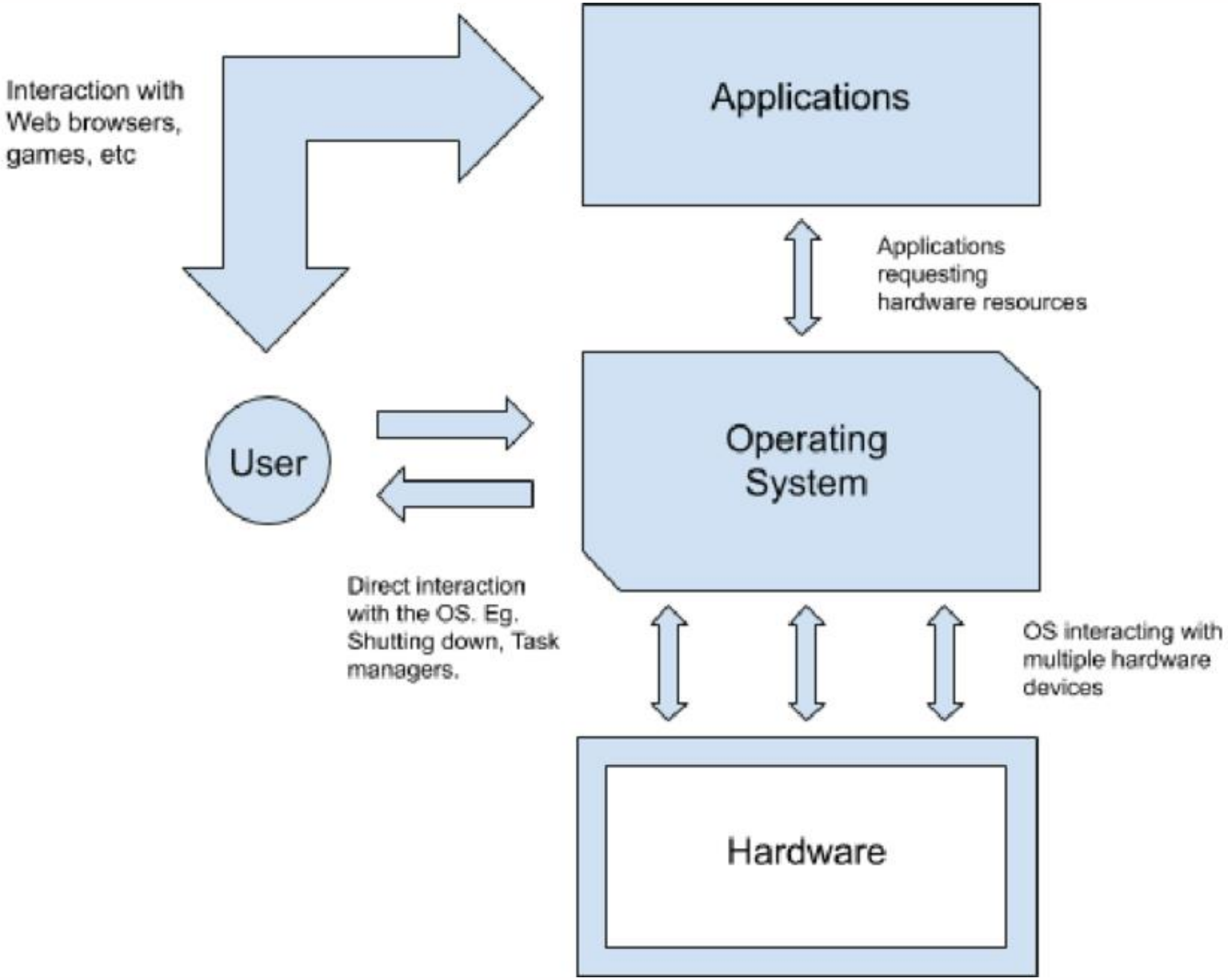


Image Credits: https://www.uow.edu.au/student/learning-co-op/technology-and-software/operating-systems

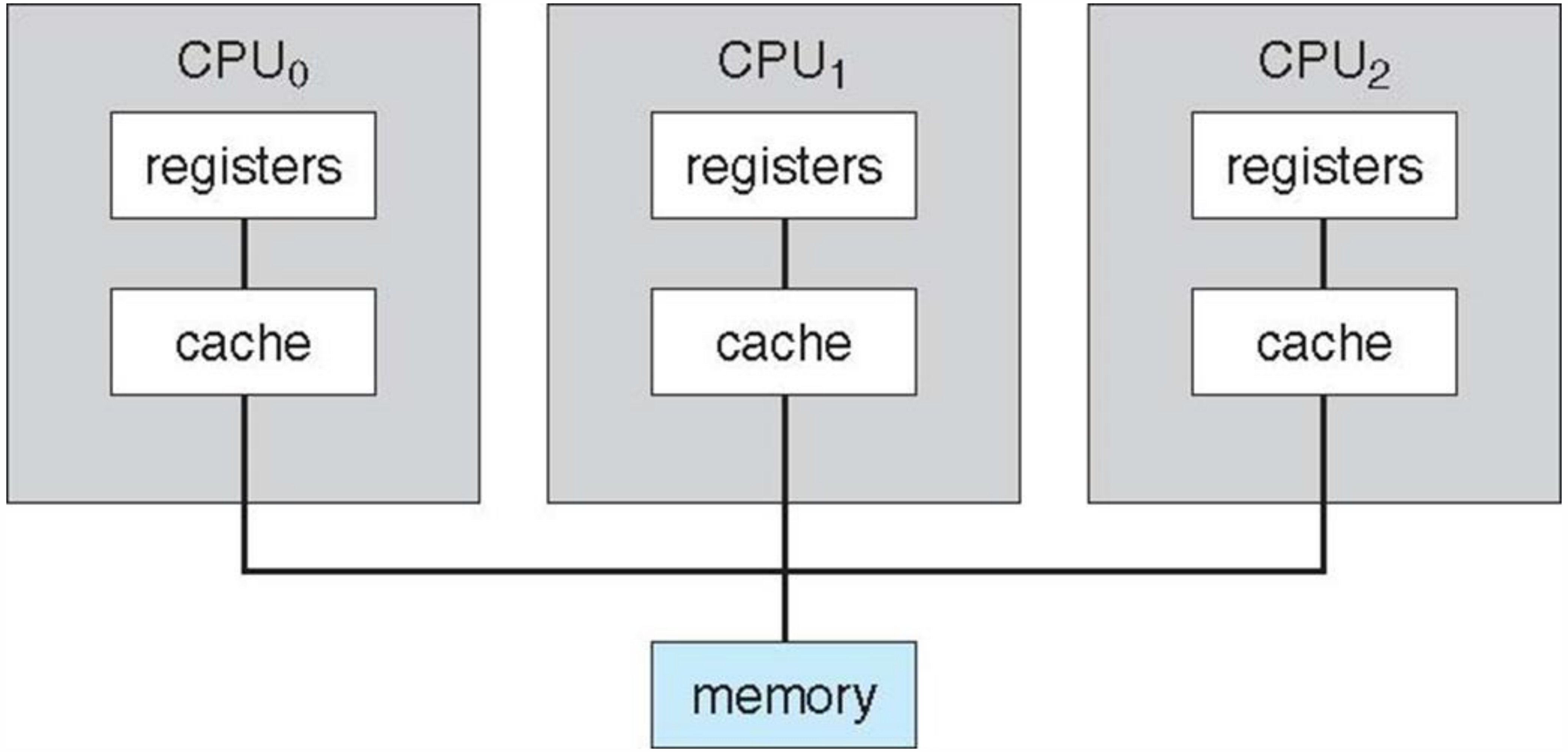
# Operating System (OS)

- OS is a resource allocator
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a control program
  - Controls execution of programs to prevent errors and improper use of the computer

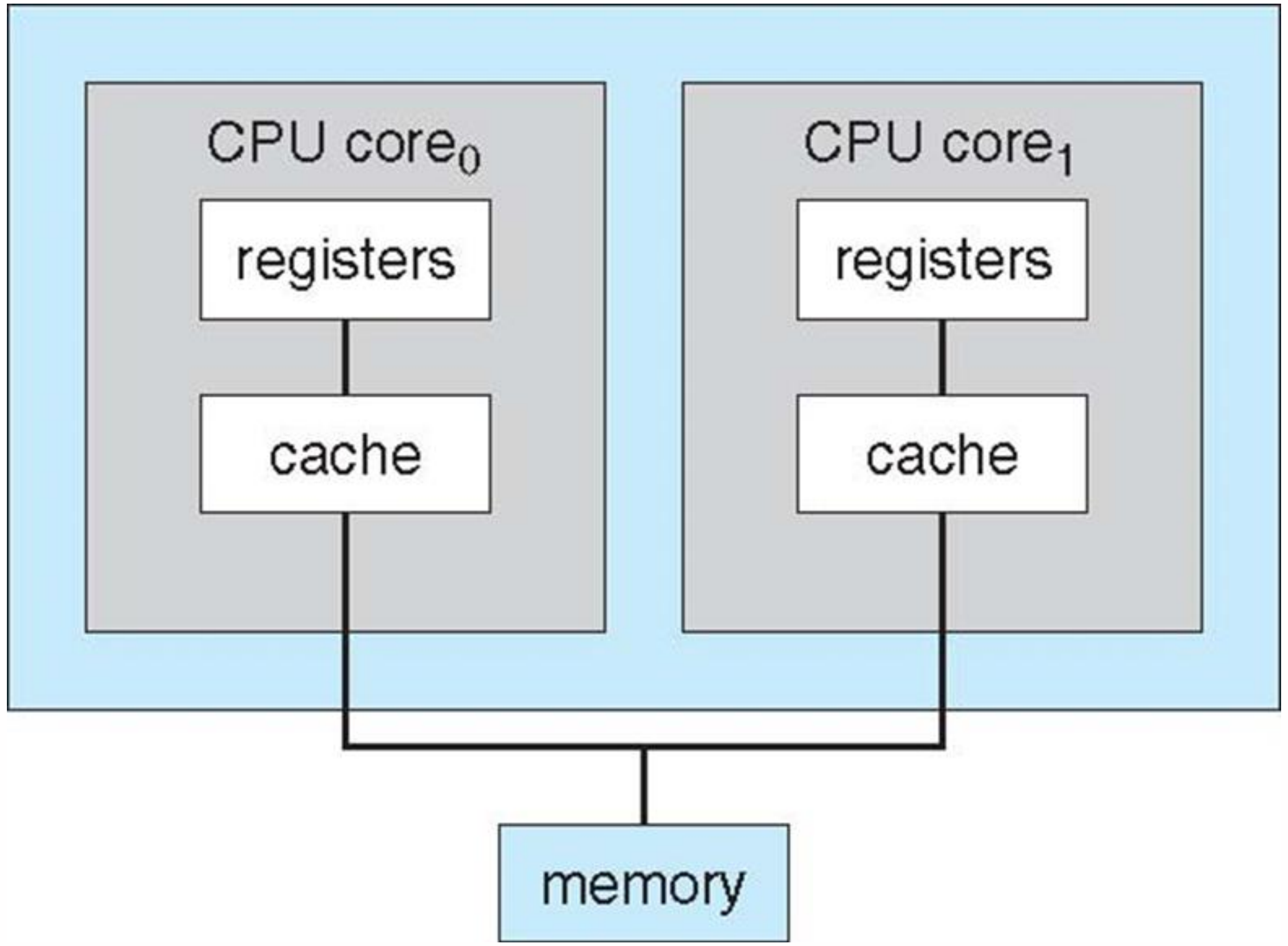

Interaction with Web browsers, games, etc
Applications
Applications requesting hardware resources
User
Operating System
Direct interaction with the OS. Eg. Shutting down, Task managers.
OS interacting with multiple hardware devices
Hardware

# Symmetric Multiprocessing Architecture

- Symmetrical multiprocessing (SMP) is a parallel computer architecture in which multiple processors share memory and other resources while running a single copy of the operating system.
- Memory, I/O, and external interrupts are accessible to all processors equally.



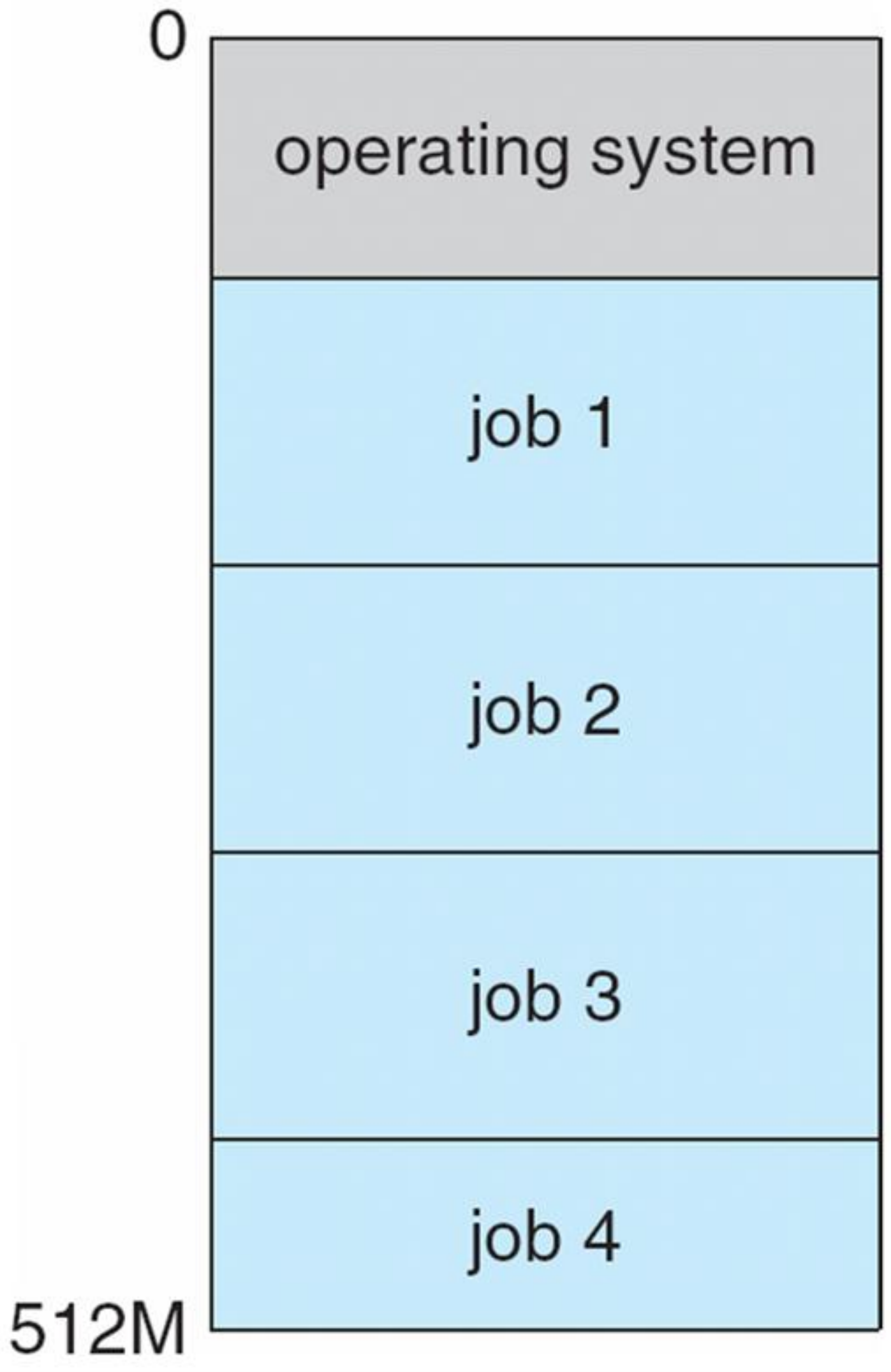
# Dual-Core


CPU core$_0$
registers
cache
CPU core$_1$
registers
cache
memory

# Operating System Structure

Multiprogramming needed for efficiency

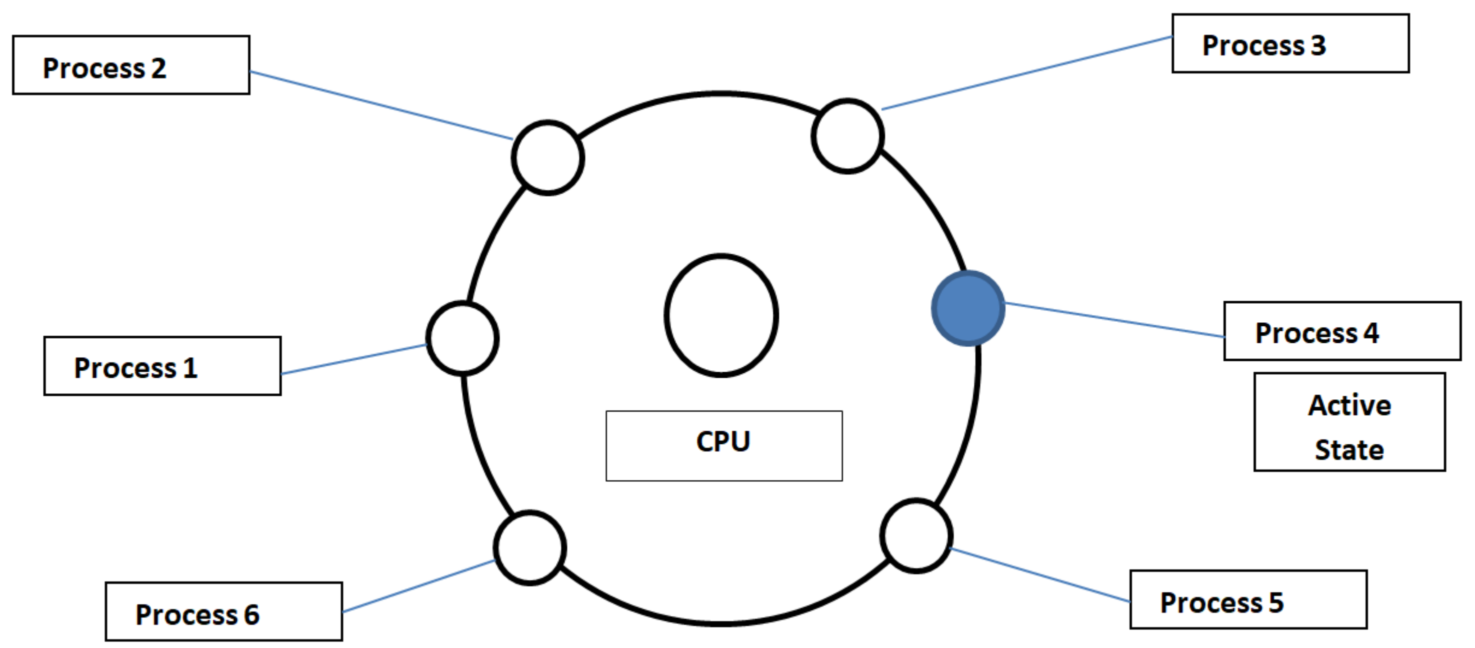
- Single user cannot keep CPU and I/O devices busy at all times
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- One job selected and run via job scheduling
- When it has to wait (for I/O for example), OS switches to another job



# Operating System Structure

Timesharing (multitasking) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing

- Response time should be < 1 second
- Each user has at least one program executing in memory [process
- If several jobs ready to run at the same time [ CPU scheduling
- If processes don’t fit in memory, swapping moves them in and out to run
- Virtual memory allows execution of processes not completely in memory

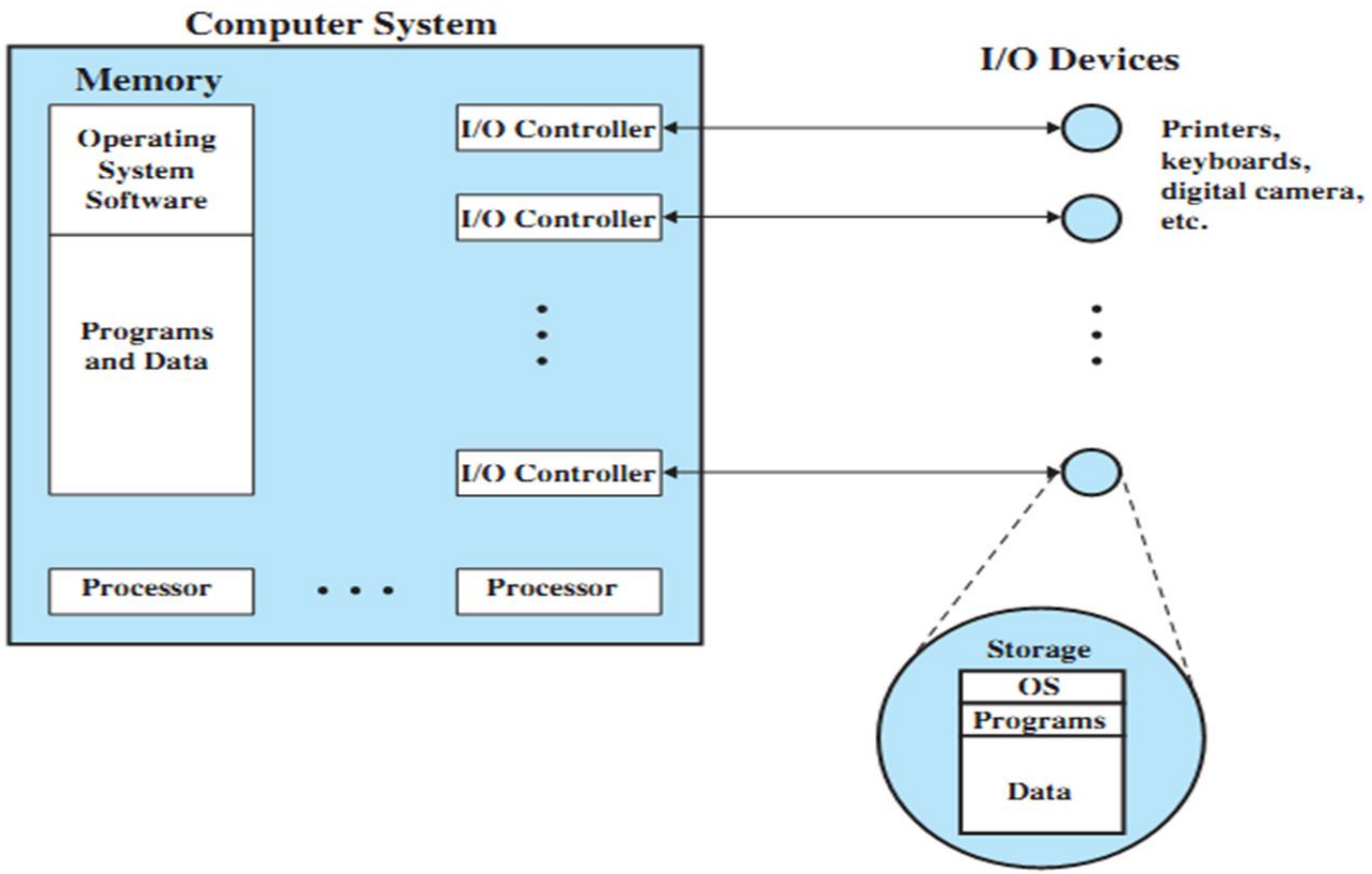


# OS as a Resource Manager

–Manages and protects multiple computer resources: CPU, Processes, Internal/External memory, Tasks, Applications, Users, Communication channels, etc…

–Handles and allocates resources to multiple users or multiple programs running at the same time and space (e.g., processor time, memory, I/O devices).

–Decides between conflicting requests for efficient and fair resource use (e.g., maximize throughput, minimize response time).



# Process Management

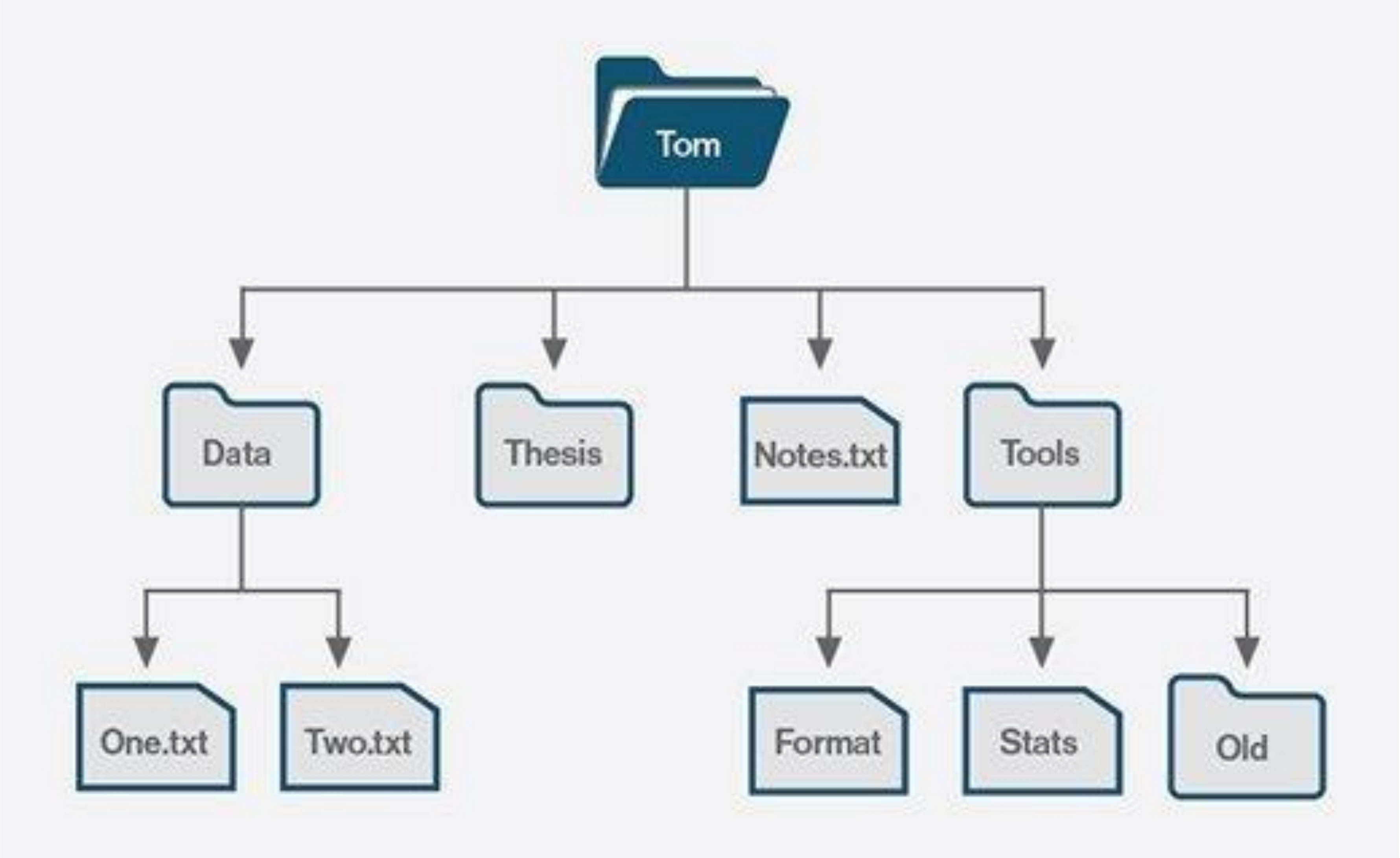
- A process is a program in execution. It is a unit of work within the system. Program is a passive entity, process is an active entity.
- Process needs resources to accomplish its task
- CPU, memory, I/O, files
- Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one program counter specifying location of next instruction to execute
- Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
- Concurrency by multiplexing the CPUs among the processes / threads

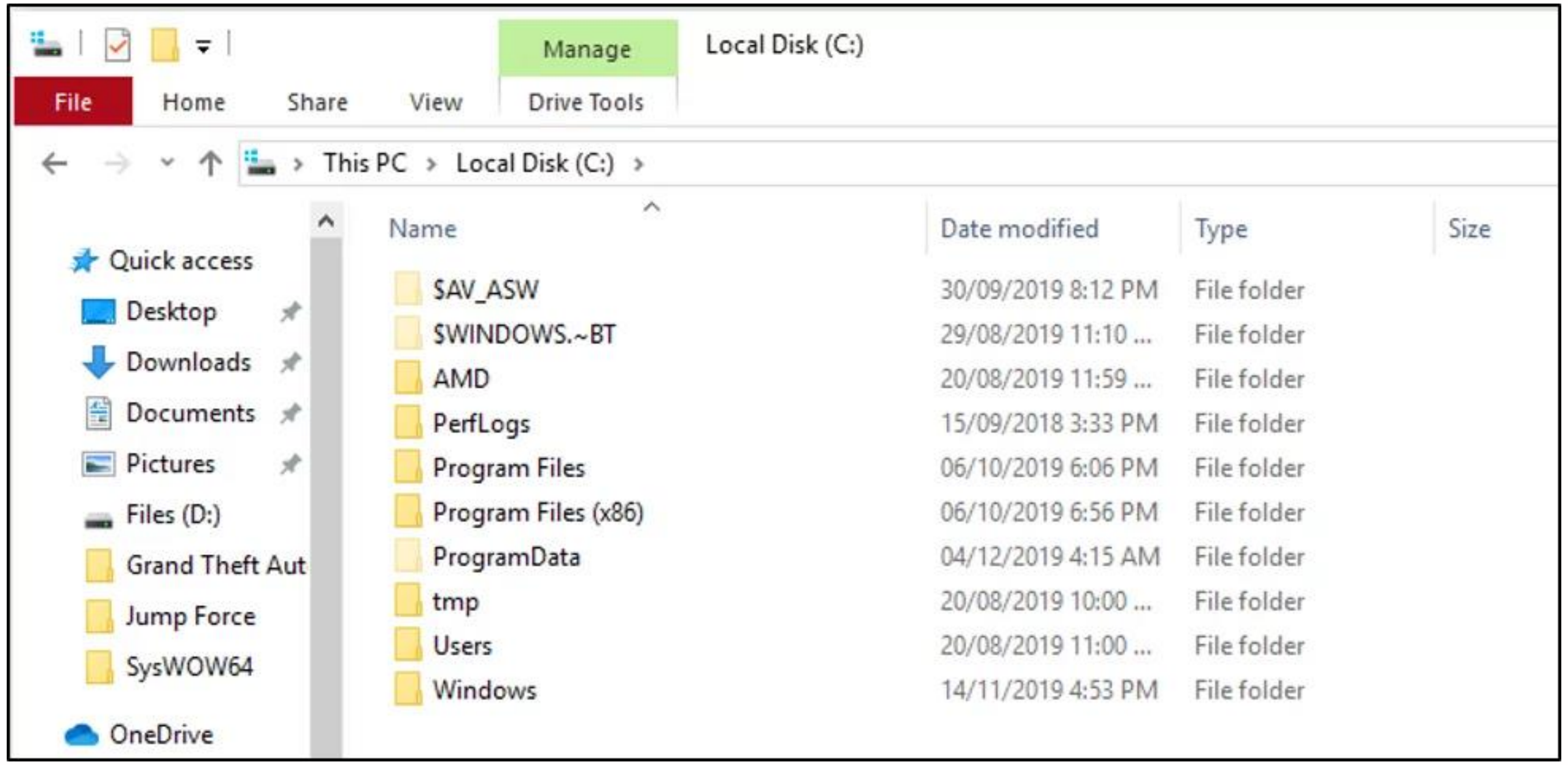
# Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
- Optimizing CPU utilization and computer response to users
- Memory management activities
- Keeping track of which parts of memory are currently being used and by whom
- Deciding which processes (or parts thereof) and data to move into and out of memory
- Allocating and deallocating memory space as needed

# File System

- A computer's file system determines how files are named and where they are logically arranged for storage and retrieval.
- Without a file system, information stored would not be separated into individual files, making it difficult to identify and retrieve.
- Operating systems (OS) such as Microsoft Windows, macOS, and Linux-based systems can have distinct file systems.





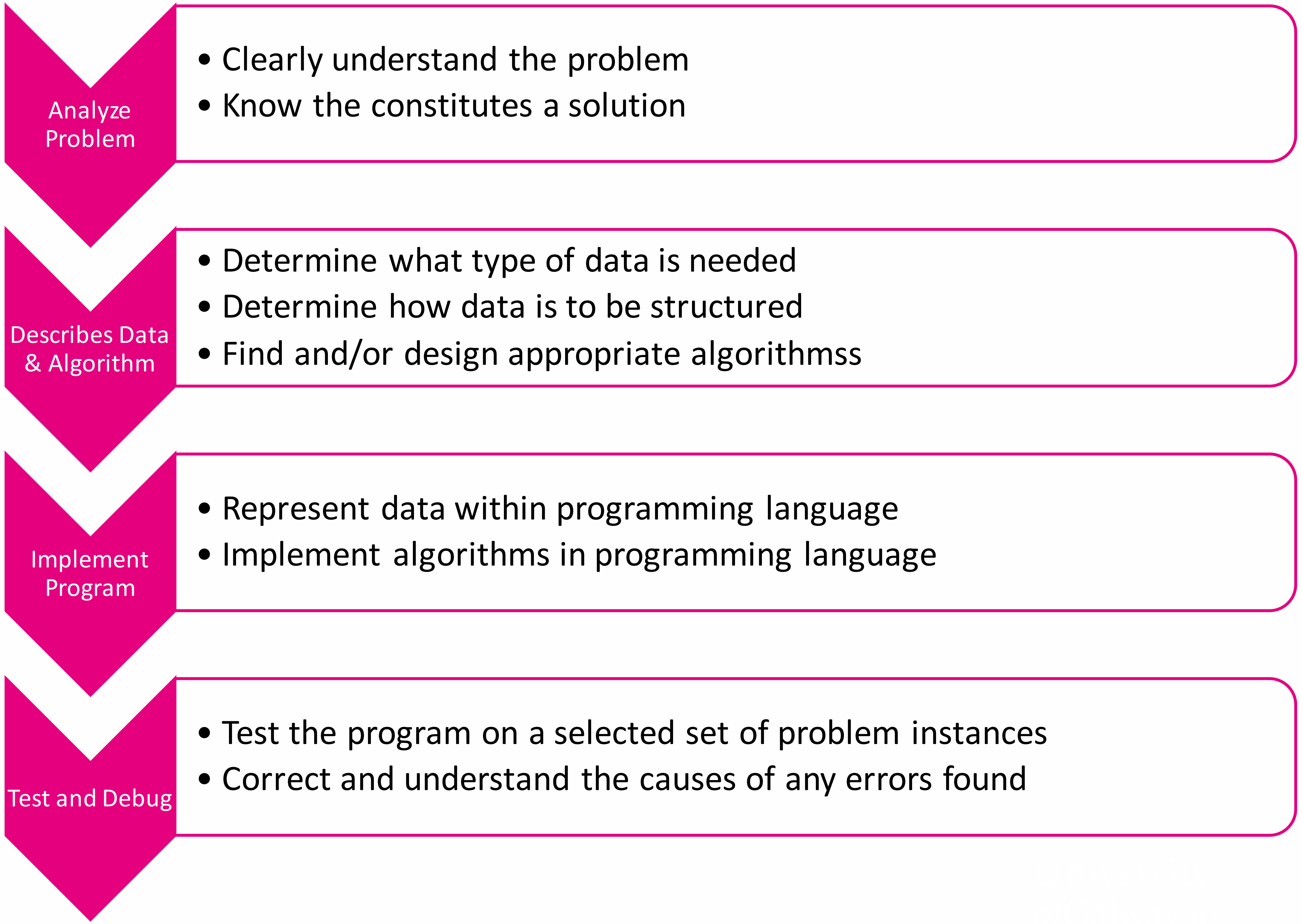
| Name | Date modified | Type | Size |
|---|---|---|---|
| $AV_ASW | 30/09/2019 8:12 PM | File folder | |
| $WINDOWS.~BT | 29/08/2019 11:10 ... | File folder | |
| AMD | 20/08/2019 11:59 ... | File folder | |
| PerfLogs | 15/09/2018 3:33 PM | File folder | |
| Program Files | 06/10/2019 6:06 PM | File folder | |
| Program Files (x86) | 06/10/2019 6:56 PM | File folder | |
| ProgramData | 04/12/2019 4:15 AM | File folder | |
| tmp | 20/08/2019 10:00 ... | File folder | |
| Users | 20/08/2019 11:00 ... | File folder | |
| Windows | 14/11/2019 4:53 PM | File folder | |

Image credits https://softwaretested.com/windows-system-files/

Ollscoil na Gaillimhe
University of Galway

# The Process of Computational Problem Solving

- Computational problem solving does not simply involve the act of computer programming. It is a process , with programming being only one of the steps.
- Before a program is written, a design for the program must be developed.
- Before a design can be developed, the problem to be solved must be well understood.
- Once written, the program must be thoroughly tested.



Analyze Problem
- Clearly understand the problem
- Know the constitutes a solution

Describes Data & Algorithm
- Determine what type of data is needed
- Determine how data is to be structured
- Find and/or design appropriate algorithmss

Implement Program
- Represent data within programming language
- Implement algorithms in programming language

Test and Debug
- Test the program on a selected set of problem instances
- Correct and understand the causes of any errors found

# Problem Analysis

- Once a problem is clearly understood, the fundamental computational issues for solving it can be determined.
- Besides clearly understanding a computational problem, one must know what constitutes a solution.
- Thus, a program may be stated as finding,
  - A solution
  - An approximate solution
  - A best solution
  - All solutions

# Abstraction

- A representation that leaves out details of what is being represented is a form of abstraction.
- Is the color of the boat relevant?
- The width of the river?
- The name of the man?
- No, the only relevant information is where each item is at each step.
- The collective location of each item, in this case, refers to the state of the problem.
- Thus, the start state of the problem can be represented as follows

**man cabbage goat wolf**

[E, E, E, E]

# The Essence of Computational Problem Solving

- In order to solve a problem computationally, two things are needed:
  - A representation that captures all the relevant aspects of the problem, and
  - An algorithm that solves the problem by use of the representation.

A man lives on the east side of a river. He wishes to bring a cabbage, a goat, and a wolf to a village on the west side of the river to sell. However, his boat is only big enough to hold himself, and either the cabbage, goat, or wolf. In addition, the man cannot leave the goat alone with the cabbage because the goat will eat the cabbage, and he cannot leave the wolf alone with the goat because the wolf will eat the goat. How does the man solve his problem

Man, Cabbage, Goat, Wolf Problem

# Abstraction

- In this representation, the symbol E denotes that each corresponding object is on the east side of the river.
- If the man were to row the goat across with him, for example, then the representation of the new problem state would be

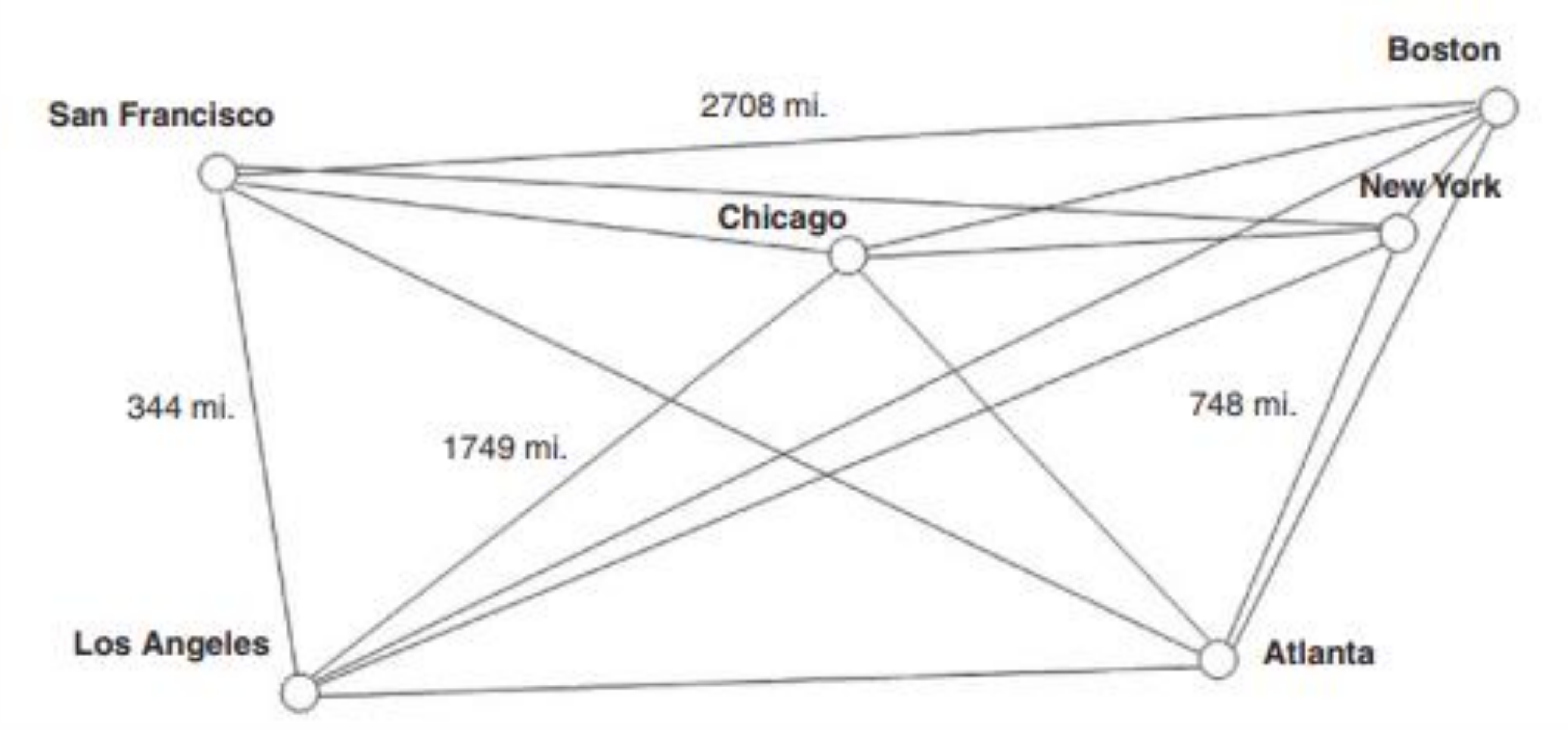
**man cabbage goat wolf**
[E, E, E, E]

- The symbol W indicates that the corresponding object is on the west side of the river—in this case, the man and goat.
- All objects are on the west side of the river

**man cabbage goat wolf**
[W, E, W, E]

# Travelling Salesman Problem

- The problem is to find the shortest route of travel for a salesman needing to visit a given set of cities.
- In a brute force approach, the lengths of all possible routes would be calculated and compared to find the shortest one.
- For ten cities, the number of possible routes is 10! (10 factorial), or over three and a half million (3,628,800). For twenty cities, the number of possible routes is 20!, or over two and a half quintillion (2,432,902,008,176,640,000).



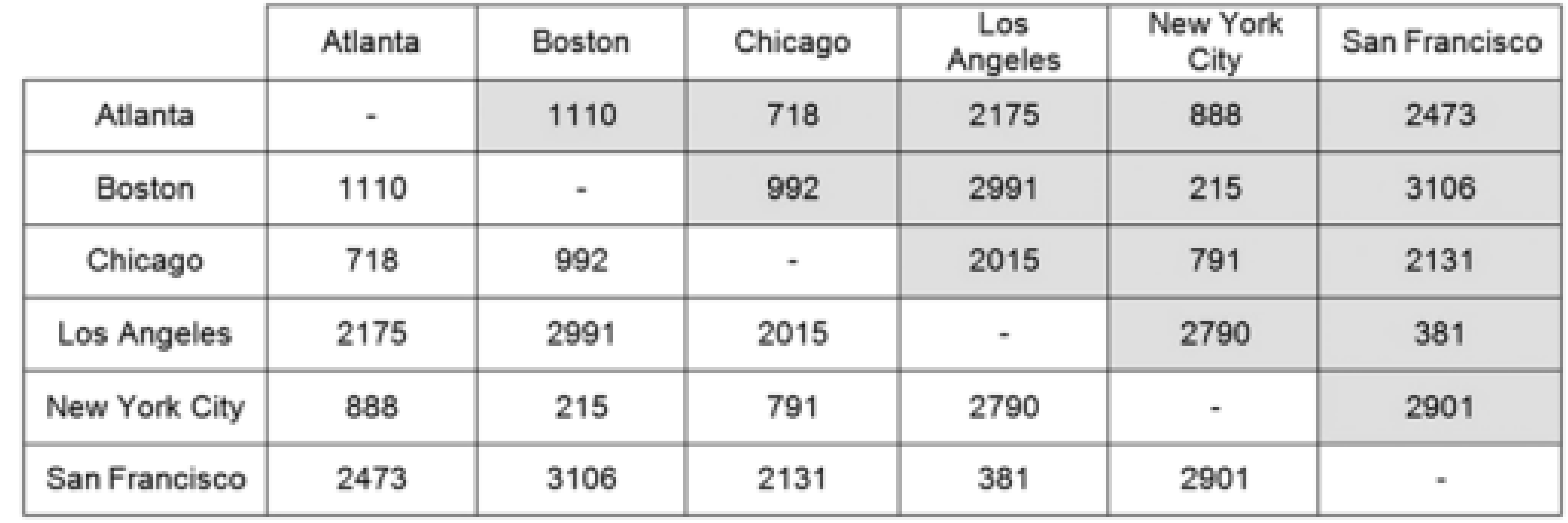
# Program Design

**man cabbage goat wolf**
[W, E, W, E]

- Describing the Data Needed

**[month , year ]**
**[31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]**
**[‘Sunday’, ‘Monday’, ‘Tuesday’, ‘Wednesday’, ‘Thursday’, ‘Friday’, ‘Saturday’]**

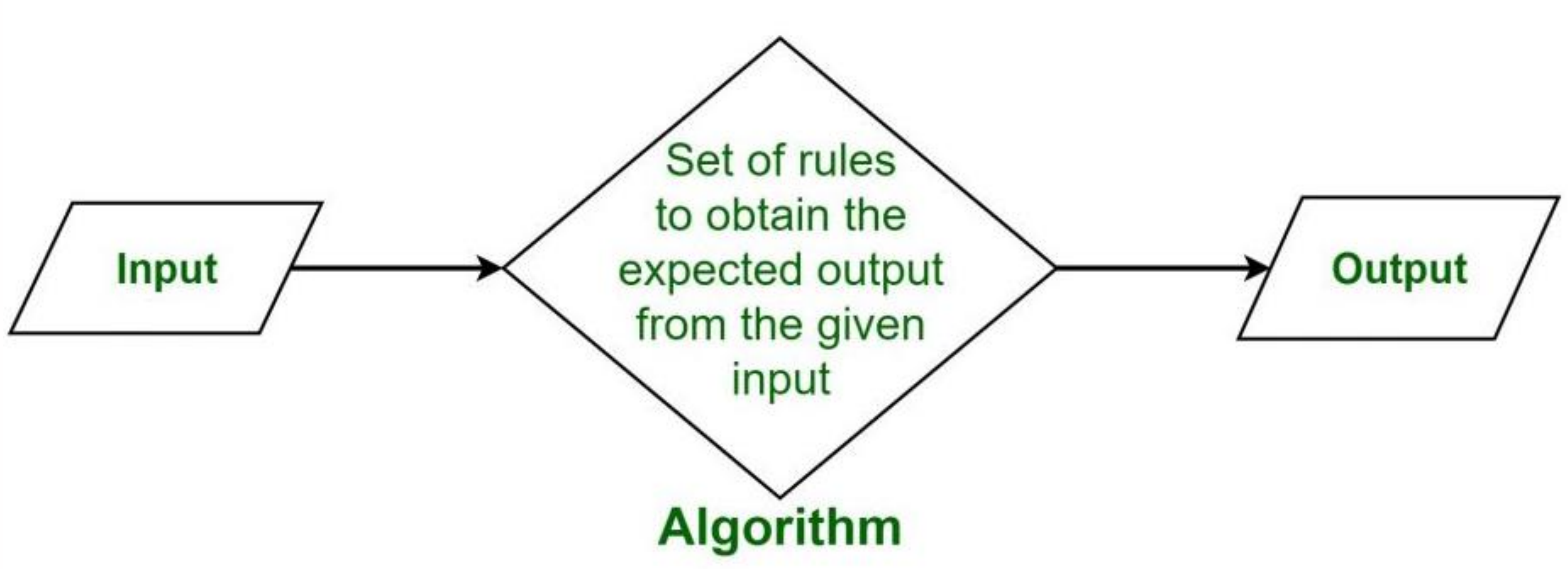
[ [‘Atlanta’, [‘Boston’, 1110], [‘Chicago’, 718], [‘Los Angeles’, 2175], [‘New York’, 888], [‘San Francisco’, 2473] ], [‘Boston’, [‘Chicago’, 992], [‘Los Angeles’, 2991], [‘New York’, 215], [‘San Francisco’, 3106] ], [‘Chicago’, [‘Los Angeles’, 2015], [‘New York’, 791], [‘San Francisco’, 2131] ], [‘Los Angeles’, [‘New York’, 2790], [‘San Francisco’, 381] ], [‘New York’, [‘San Francisco’, 2901] ] ]



| | Atlanta | Boston | Chicago | Los Angeles | New York City | San Francisco |
|---|---|---|---|---|---|---|
| Atlanta | - | 1110 | 718 | 2175 | 888 | 2473 |
| Boston | 1110 | - | 992 | 2991 | 215 | 3106 |
| Chicago | 718 | 992 | - | 2015 | 791 | 2131 |
| Los Angeles | 2175 | 2991 | 2015 | - | 2790 | 381 |
| New York City | 888 | 215 | 791 | 2790 | - | 2901 |
| San Francisco | 2473 | 3106 | 2131 | 381 | 2901 | - |

# Describing the Needed Algorithms

- When solving a computational problem, either suitable existing algorithms may be found or new algorithms must be developed
- Algorithms that work well in general but are not guaranteed to give the correct result for each specific problem are called heuristic algorithms


Input
Set of rules to obtain the expected output from the given input
Output
Algorithm

# Program Implementation

- Design decisions provide general details of the data representation and the algorithmic approaches for solving a problem.
- The details, however, do not specify which programming language to use, or how to implement the program.
- That is a decision for the implementation phase.

# Program Testing

- Programming errors are pervasive, persistent, and inevitable
- Software testing is an essential part of software development
- Any changes made in correcting a programming error should be fully understood as to why the changes correct the detected error.

# References

- Introduction to Computer Science Using Python by Charles Dierbach
- Computer Systems A Programmer's Perspective by Randal E. Bryant and David R. O'Hallaron
- Computer Organization and Architecture Designing for Performance Tenth Edition by William Stallings

OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

# Thank *you*

University
*of*Galway.ie