Chapter One: Introduction

A SHORT INTRODUCTION TO HARDWARE, SOFTWARE, AND ALGORITHM DEVELOPMENT

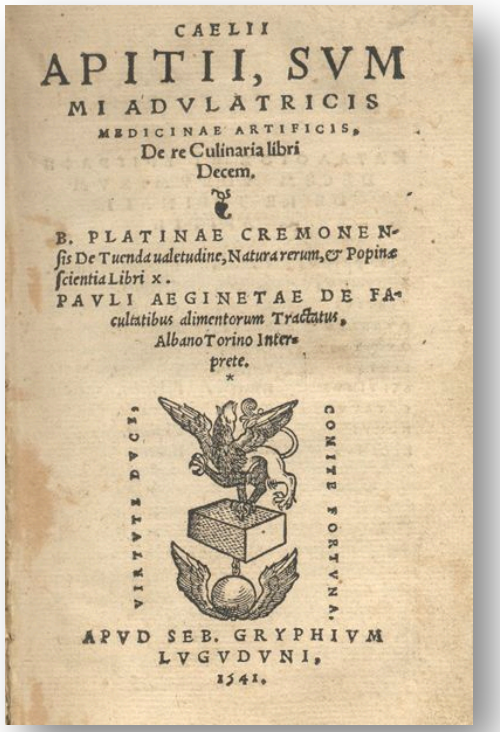
Chapter Goals

- In this chapter you will earn:
  - About computer hardware, software and programming
  - How to write and execute your first Python program
  - How to diagnose and fix programming errors
  - How to use pseudocode to describe an algorithm

Our First Definition

Algorithm:

- An ***algorithm*** is a step by step description of how to solve a problem.


CAELII
APITII, SVM
MI ADVLATRICIS
MEDICINAE ARTIFICIS,
De re Culinaria libri
Decem.
B. PLATINAE CREMONEN-
sis De Tuenda ualetudine, Natura rerum, & Popinæ
scientia Libri x.
PAVLI AEGINETAE DE FA-
cultatibus alimentorum Tractatus,
Albano Torino Inter-
prete.
VIRTVTE DVCI, COMITE FORTVNA.
APVD SEB. GRYPHIVM
LVGVDVNI,
1541.

The large collection of recipes *De re coquinaria*, conventionally entitled 'Apicius', appeared in the 4th or 5th century and is the only more or less complete surviving cookbook from the classical world. It lists the courses served in a meal as
'Gustatio' (appetizer),
'Primae Mensae' (main course)
'Secundae Mensae' (dessert).

Computer Programs

- A computer program tells a computer the sequence of steps needed to complete a specific task
  - The program consists of a very large number of primitive (simple) instructions
- Computers can carry out a wide range of tasks because they can execute different programs
  - Each program is designed to direct the computer to work on a specific task

Programming:

- The act of designing, implementing, and testing computer programs

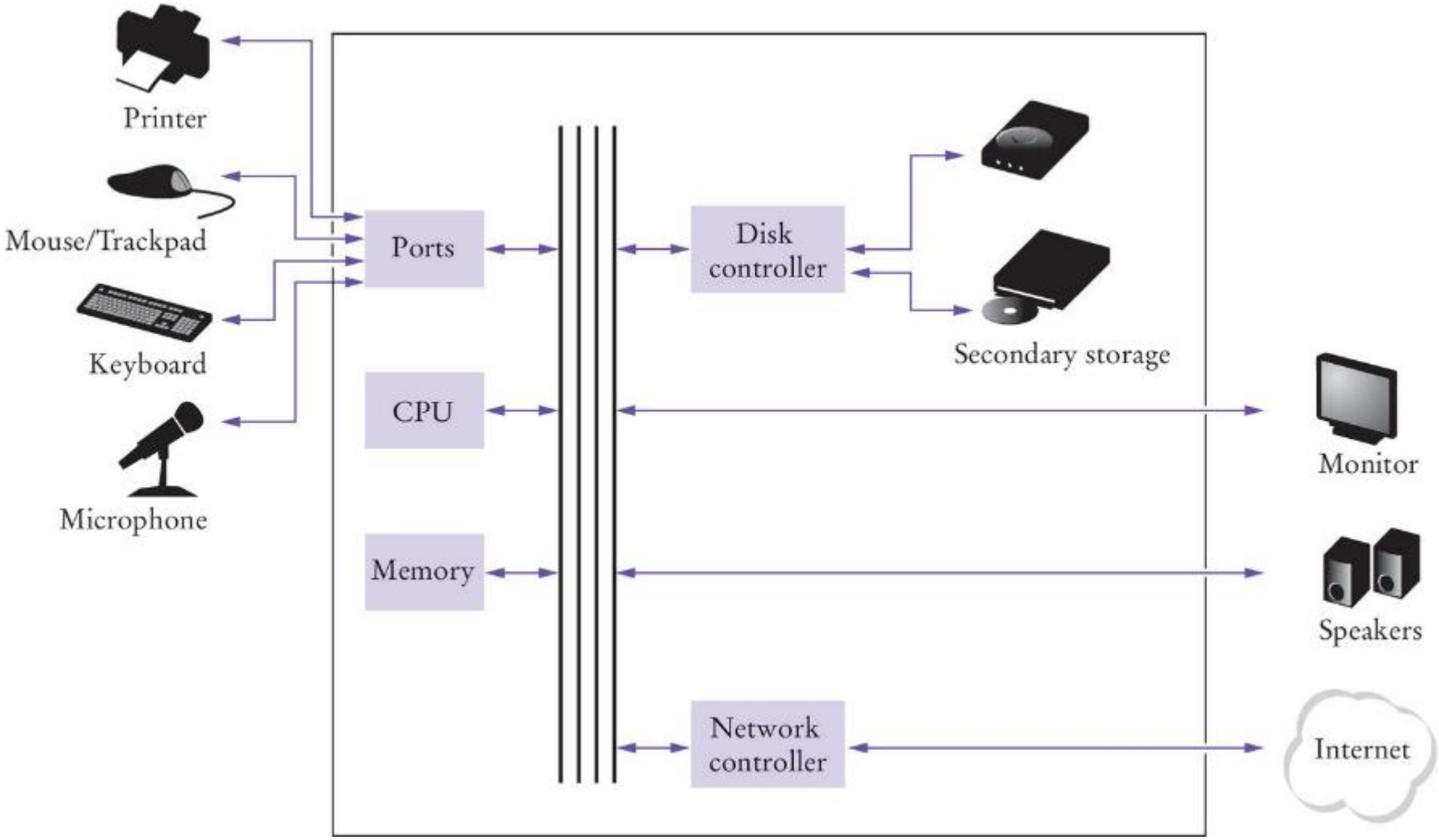
Hardware and Software

THE BUILDING BLOCKS THAT MAKE UP A COMPUTER

Hardware

- ***Hardware*** consists of the physical elements in a computer system.
  - Some very visible examples are the monitor, the mouse, external storage, and the keyboard.
- The ***central processing unit*** (CPU) performs program control and data processing
- Storage devices include memory (RAM) and secondary storage
  - Hard disk
  - Flash drives
  - CD/DVD drives
- Input / output devices allow the user to interact with the computer
  - Mouse, keyboard, printer, screen…

Simple View of a Computer's Components

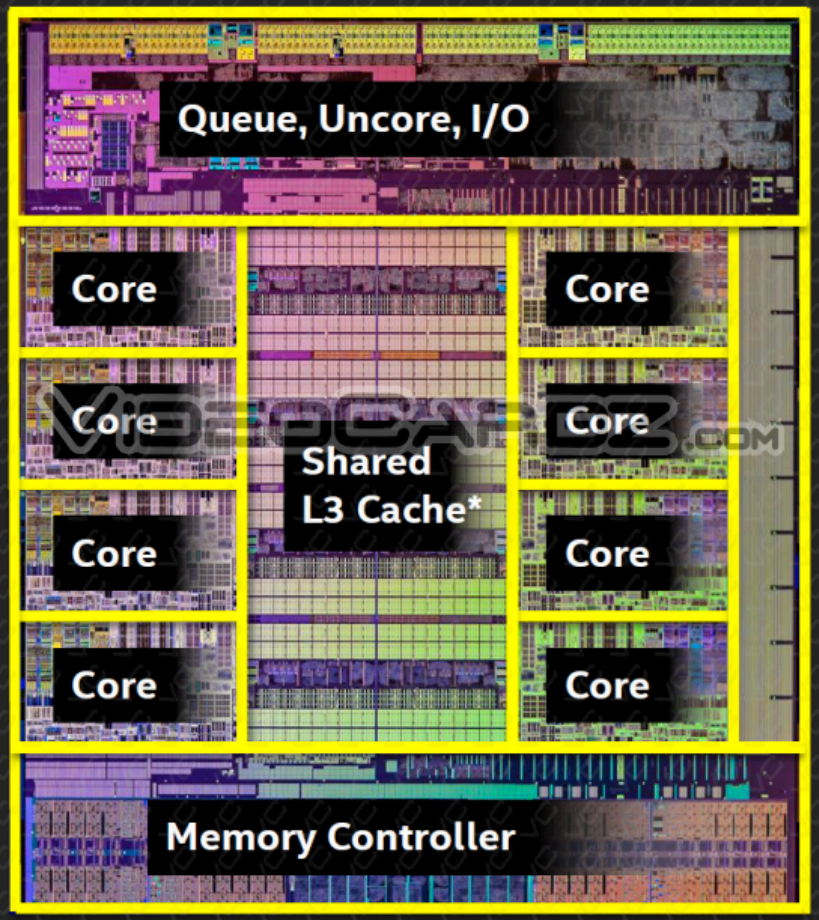


The CPU (aka “a core”)

- The CPU has two components, the **control unit** and the **arithmetic logic unit**
- The ***control unit*** directs operation of the processor.
  - All computer resources are managed by the **control unit**.
  - It controls communication and co-ordination between input/output devices.
  - It reads and interprets instructions and determines the sequence for processing the data.
  - It provides timing and control signals
- The ***arithmetic logic unit*** contains the circuitry to perform calculations and do comparisons.
  - It is the workhorse portion of the computer and its job is to do precisely what the control unit tells it to do.

Intel® Core™ i7-5960X Processor Die Map

22nm Tri-Gate 3-D Transistors



- Transistor count: 2.6 Billion
- Die size: 17.6mm x 20.2mm

*20MB of cache is shared across all 8 cores

nder Embargo until
00am PST, August 29, 2014

intel Look In

Storage

- There are two types of storage:
  - Primary Storage
  - Secondary Storage
- Primary storage is composed of memory chips: electronic circuits that can store data as long as it is provided electric power
- Secondary storage provides a slower, less expensive storage that is persistent: the data persists without electric power
- Computers store both data and programs
  - The data and program are located in secondary storage and loaded into memory when the program is executed

Memory

- A simple way to envision primary memory is a table of cells all the same size, one byte, and each containing a unique address beginning with 0.
  - The “typical” computer has a main memory ranging from 4 gigabytes (GB), to 32 GB.
- How big is a gigabyte?
  - A byte is 8 bits.
  - A kilobyte, KB, is 1024 bytes, or “about 1 thousand bytes.”
  - A megabyte, MB, is 1,048,576 bytes, or “about 1 million bytes.”
  - A ***gigabyte***, GB, is 1,073,741,824 bytes or “about 1 billion bytes.”

Executing a Program

- Program instructions and data (such as text, numbers, audio, or video) are stored in digital format
- When a program is started, it is brought into memory, where the CPU can read it.
- The CPU runs the program one instruction at a time.
  - The program may react to user input.
- The instructions and user input guide the program execution
  - The CPU reads data (including user input), modifies it, and writes it back to memory, the screen, or secondary storage.

Software

- **Software** is typically realized as an application program
  - Microsoft Word is an example of software
  - Computer Games are software
  - Operating systems and device drivers are also software
- Software
  - Software is a sequence of instructions and decisions implemented in some language and translated to a form that can be executed or run on the computer.
- Computers execute very basic instructions in rapid succession
  - The basic instructions can be grouped together to perform complex tasks
- Programming is the act of designing and implementing computer programs

Algorithms

Introduction to Algorithms

- If you want a computer to perform a task, you start by writing an algorithm
- An ***Algorithm*** is:
  - a sequence (the order mattering) of actions to take to accomplish the given task
  - An algorithm is like a recipe; it is a set of instructions written in a sequence that achieves a goal
- For complex problems software developers write an algorithm before they attempt to write a computer program
- For this class our goal is to ALWAYS write an algorithm for each project
- Developing algorithms is a fundamental problem solving skill
  - It has uses in many fields outside of Computer Science

Algorithm: Formal Definition

An ***algorithm*** describes a sequence of steps that is:

1. Unambiguous
   a. No “assumptions” are required to execute the algorithm
   b. The algorithm uses precise instructions
2. Executable
   a. The algorithm can be carried out in practice
3. Terminating
   a. The algorithm will eventually come to an end, or halt

Problem Solving: Algorithm Design

- Algorithms are simply plans
  - Detailed plans that describe the steps to solve a specific problem
- You already know quite a few
  - Calculate the area of a circle
  - Find the length of the hypotenuse of a triangle
- Some problems are more complex and require more steps
  - Calculate PI to 100 decimal places
  - Calculate the trajectory of a missile

A Simple Example

- A simple algorithm to get yourself a drink of orange juice
  - For simplicity, the following are true:
    - You have a clean glass in the cabinet
    - You have orange juice in your refrigerator
- So one valid algorithm is:
  1. get a glass from your cabinet
  2. go to the refrigerator and get the orange juice container
  3. open the orange juice container
  4. pour the orange juice from the container into the glass
  5. put the orange juice container back in the refrigerator
  6. drink your juice

Second Example: Selecting a Car

Problem Statement:

- You have the choice of buying two cars.
- One is more fuel efficient than the other, but also more expensive.
- You know the price and fuel efficiency (in miles per gallon, mpg) of both cars.
- You plan to keep the car for ten years.
- Which car is the better deal?

Developing the Algorithm

Determine the inputs and outputs

From the problem statement we know:

- Car 1: Purchase price, Fuel Efficiency
- Car 2: Purchase price, Fuel Efficiency
- Price per gallon = $4.00
- Annual miles driven= 15,000
- Length of time = 10 years

For each car we need to calculate:

- Annual fuel consumed for each car
- Annual fuel cost for each car
- Operating cost for each car
- Total cost of each Car

- Then we select the car with the lowest total cost

Translating the Algorithm to pseudocode

- Break down the problem into smaller tasks
  - 'Calculate total cost' for each car
  - To calculate the total cost for each year we need to calculate the operating cost
  - The operating cost depends on the annual fuel cost
  - The annual fuel cost is the price per gallon * the annual fuel consumed
  - The annual fuel consumed is the annual miles drive / fuel efficiency
- Describe each subtask as pseudocode
  - total cost = purchase price + operating cost

The Psuedocode

For each Car, compute the total cost

- Annual fuel consumed = annual miles driven / fuel efficiency
- Annual fuel cost = price per gallon * annual fuel consumed
- Operating cost = Length of time * annual fuel cost
- Total cost = purchase price + operating cost

If total cost1 < total cost2

- Chose Car1

Else

- Choose Car2

Bank Account Example

- Problem Statement:
  - You put $10,000 into a bank account that earns 5 percent interest per year. How many years does it take for the account balance to be double the original?
- How would you solve it?
  - Manual method
    - Make a table
    - Add lines until done
  - Use a spreadsheet!
    - Write a formula
      - Per line, based on line above

year	balance
0	10000
1	10000.00 x 1.05 = 10500.00
2	10500.00 x 1.05 = 11025.00
3	11025.00 x 1.05 = 11576.25
4	11576.25 x 1.05 = 12155.06

Develop the algorithm steps

- You put $10,000 into a bank account that earns 5 percent interest per year. How many years does it take for the account balance to be double the original?

year	balance
0	10000

- Break it into steps
  - Start with a year value of 0 and a balance of $10,000
  - Repeat the following while the balance is less than $20,000
    - Add 1 to the year value
    - Multiply the balance by 1.05
      - (5% increase)
  - Report the final year value as the answer

year	balance
0	10000
1	10500
14	19799.32
15	20789.28

Translate to pseudocode

- Pseudocode
  - Half-way between natural language and a programming language
- Modified Steps
  - Set the year value of 0
  - Set the balance to $10,000
  - While the balance is less than $20,000
    - Add 1 to the year value
    - Multiply the balance by 1.05
  - Report the final year value as the answer
- The pseudocode is easily translated into Python

The Python Language

- In the early 1990's, Guido van Rossum designed what would become the Python programming language
- Van Rossum was dissatisfied with the languages available
  - They were optimized to write large programs that executed quickly
- He needed a language that could not only be used to create programs quickly but also make them easy to modify
  - It was designed to have a much simpler and cleaner syntax than other popular languages such as Java, C and C++ (making it easier to learn)
  - Python is interpreted, making it easier to develop and test short programs
- Python programs are executed by the Python interpreter
  - The interpreter reads your program and executes it