INTRODUCTION TO COMPUTATION

Spring 2015

Computation as a Research Tool

- What is “computation”?
  - The goal of computation is to solve a problem (which may or may not be mathematical)
  - Computation requires …
    - The design of an algorithm (a set of general steps) to solve the problem.
    - The use of a programming language(s) to create a list of instructions that implement the algorithm.
    - A machine for running the program.

$\Big\}$ The focus of this class

A QUICK LOOK AT THE “MACHINE”

Tools for Computation

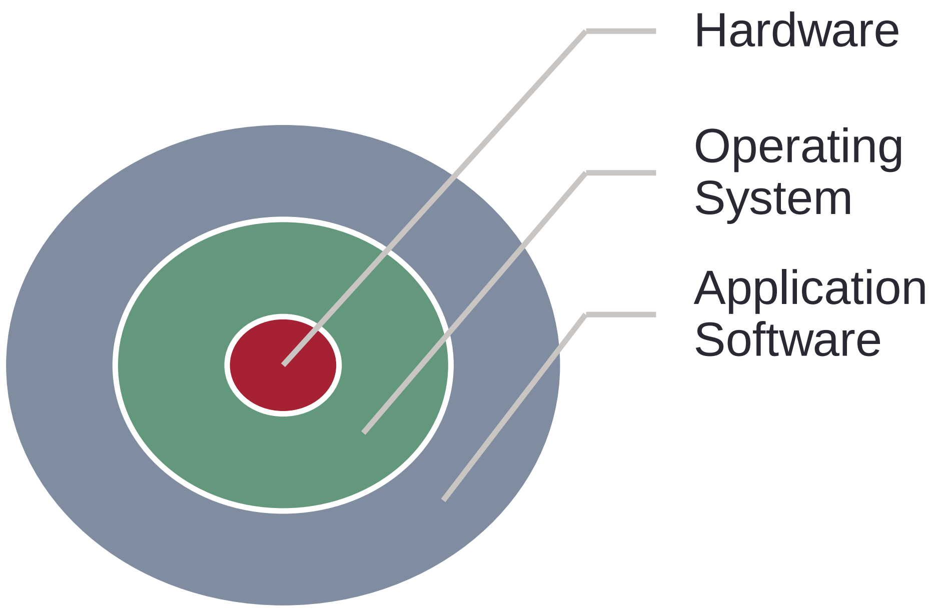
- Computers have . . .
  - Hardware:

    The collection of physical components that comprise the machine

  - Software:

    The intangible instructions that make the hardware work

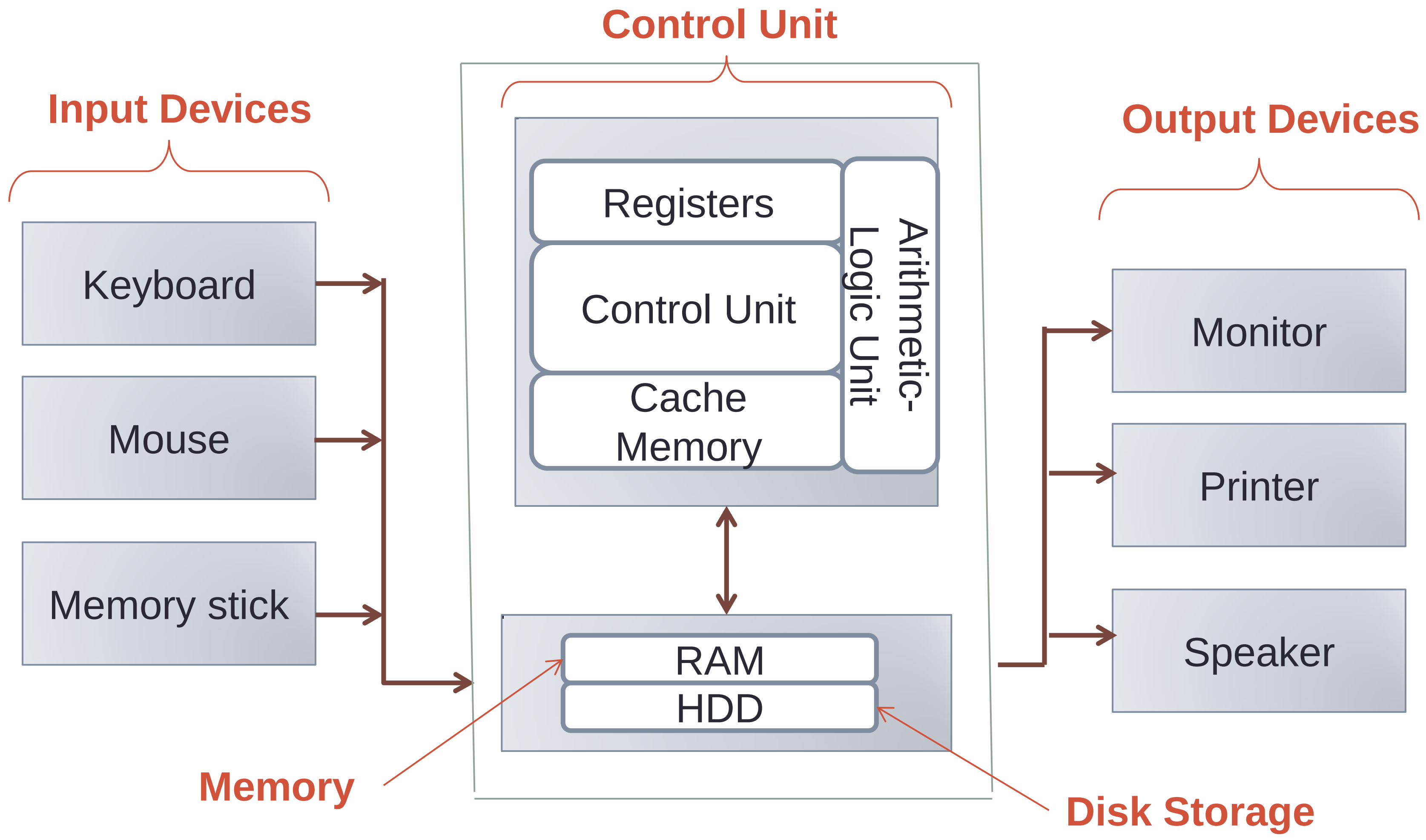
    - Operating System
    - Application Software



Basic Computer Architecture

- At the most basic level, a computer consists of
  - A Control Unit
    - Central Processing Unit (cpu)
    - Brains of the computer – oversees most operations
  - Memory
    - A place to store information while the computer is running.
    - Volatile – information goes away when power is turned off.
  - Disk storage (or Hard Drive)
    - Stores everything that is not in use (plus some things that are in use)
    - Non-volatile memory – information stays when power is off
  - Input/Output Devices
    - Items such as keyboard, screens that allow us to get information into and out of the computer.

The Hardware: single core

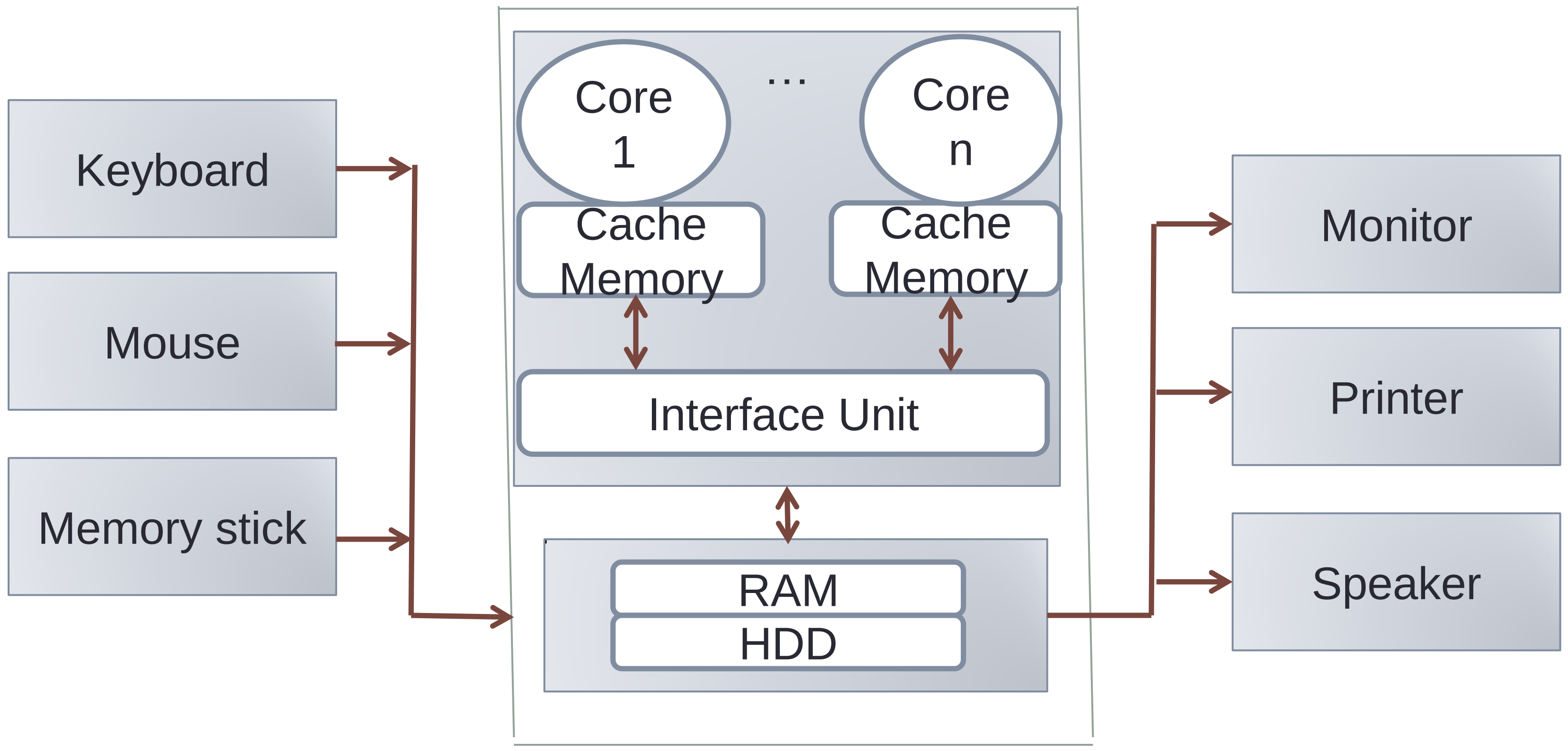


Hierarchy of Memory

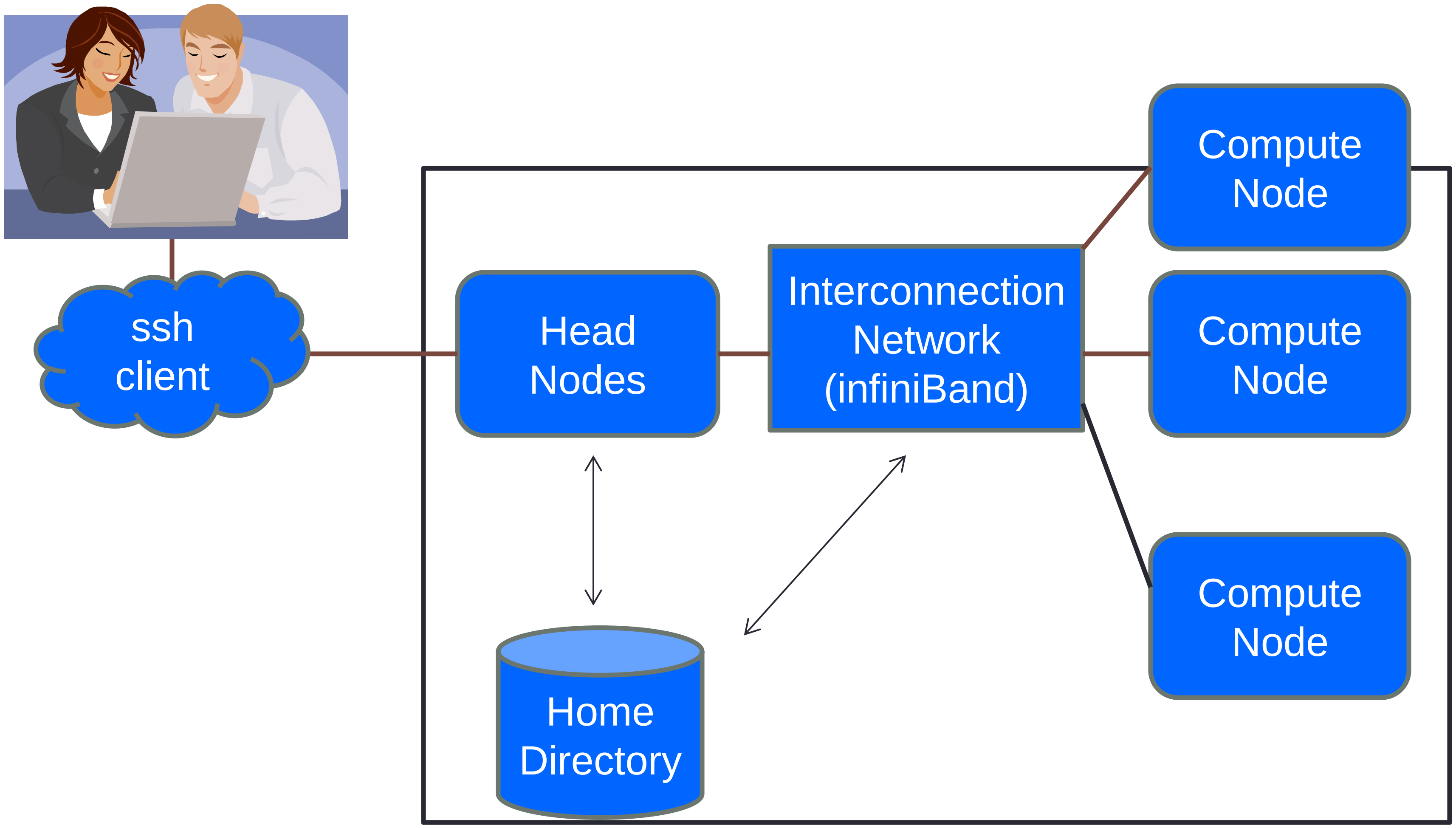
- Register
  - Very fast, but very expensive memory.
  - Only a few per core and they are integrated on the cpu chip. Used to store the items on which the cpu is currently working.
- Cache memory
  - Fast, but expensive memory
  - Stores values that are currently in use or might be used very soon.
  - Most processors nowadays have multiple levels of cache (L1, L2, L3) that become more capacious and slower as the level increases.
- Random Access Memory (RAM)
  - Main memory
  - Slow, but cheap
- Disk storage
  - Not “memory” but can be used to store values for which there isn’t enough room in RAM in a section called swap.
  - Very, very slow compared to everything else. This is why just upgrading your RAM quantity often speeds up an older system considerably.

Hardware: multicores

- Each core acts like an individual CPU.



Hardware: computer cluster



The Operating System

- The operating system is the interface between the user of the computer and the hardware.

- It oversees and coordinates all activities performed by the hardware, including
    - Process scheduling and synchronization
    - Memory management
    - File system allocation
    - I/O interface management
    - Network & Security management

- There are three commonly-used operating systems:
    - Unix/Linux
    - Windows
    - OS X

The Operating System

- The operating system is comprised of two parts:
  - Kernel – which handles all of the allocations and managements
  - Shell – which provides the interface with the user (e.g., command-line or GUI)

- We will be working with a Unix/Linux operating system. In particular, we will be using a bash shell.

Application Software

- Any program that allows the user to perform specific tasks.
  - Note: Most of the programs that you use day-to-day are applications
    - Apps on your cell phone
    - Collab
    - Google
- All applications have to be written in a programming language (e.g., C/C++, Fortran, Mathematica, Matlab, Python, R).
  - The languages that we will use for this class fall into two categories: compiled languages and interpretive language.
  - Some languages may have an application, called an IDE (Integrated Development Environment) that combines a file editor, the compiler or interpreter, and other useful tools.

GETTING STARTED: ALGORITHMS

Returning to Algorithms . . .

- Recall that an algorithm is a set of steps used to achieve a goal.

- We work through algorithms everyday.
    - Classic example: How to tie your shoestrings
    - How to make popcorn
    - How to add two fractions: $\frac{1}{2} + \frac{1}{5}$
    - How to get to the MEC building
    - How to install a program on your laptop

- Note: There can be more than one way to achieve a goal. We usually want to find the clearest and most concise algorithm.

Simplest Algorithms

- In programming, the simplest algorithm you will see is

  Get a value (often input by the user).
  Perform an “operation” on the value.
  Report the result.

- Interesting fact: In some texts, an algorithm is defined as the sequence of steps needed to transform the input into the output.

Algorithm Examples

- Problem #1: The weatherman keeps giving me the current temperature in degrees Celsius, but I need it in degrees Fahrenheit. How would I write an algorithm to do this?

```
Prompt the user to type in the Celsius temperature
Convert to Fahrenheit using the formula: $F = (9.0/5.0)*C + 32.0$
Write the result to the screen.
```

- Problem #2: I have a long list of values stored in a file, and I need to find the largest in the list.

```
Prompt the user for the name of the file where the values are listed.
Read in the first value of the list.
Assume it is the max until a larger value is found; so, save it in a special location.
Read the next value from the list and compare it with the current max value.
   If the new value is larger, save it in the special location.
 Repeat until we've read all values in the list.
```

Algorithm Examples (con't)

- Problem #3: I have drawn three random numbers (all different values), and I need to order them from smallest to largest. The three numbers are labeled a, b, and c. How do I order them?

```
If a < b, then I just need to find where c goes.
    if c < a, then the order is c – a – b
    otherwise, if c < b, then the order is a – c – b
    otherwise, the order is a – b – c
Otherwise, I know that b < a; again, I need to find where c goes.
    if c < b, then the order is c – b – a
    otherwise, if c < a, then the order is b – c – a
    otherwise, the order is b – a – c
```

Summary

- To use computation as a research tool, we will need to
  - Develop algorithms that describe how to solve the computational problem(s)
  - Write the algorithm as a set of instructions (using a programming language) so that the computer can perform the desired operations.
- Often, there is more than one way to solve a problem.
  - We want our algorithms to be clear and concise.
  - The algorithm is the “blueprint” for our code.
- The next step is to learn the syntax (i.e., the grammatical rules) for the programming language of choice.