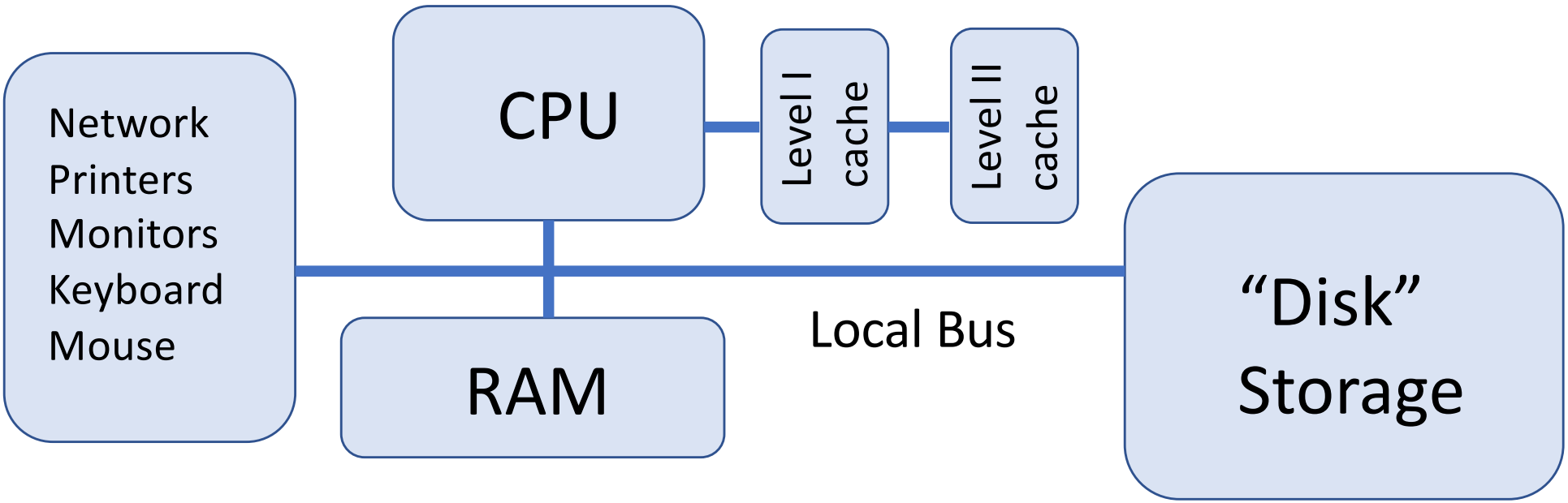
What does a computer do? and how does it do it?

Simply stated, a computer consists of three basic units

- The central processing unit (CPU)
- Memory
  - Cache (level I and level II)
  - Random access memory (RAM)
- Disk storage



The CPU is the ”brains” of the computer. Its performance is determined by the number of cycles it performs per second and what it can do each cycle.

In each cycle the CPU can

- Read (input) instructions/data from the cache
- Write (output) data to the cache
- Manipulate data (add, subtract, multiply etc)
- Send data and instructions along the local bus (i.e. input/output to RAM, disc, network, monitor)

The tasks the CPU can perform in each cycle is determined by the instruction set that is specific to a particular CPU or class of CPU.

Progress

- September 1984
- Processor: Motorola 680000
- Processor speed: 8 MHz
- System Bus Speed 8 MHz
- Memory registers: 16-bit
- Level 1 Cache: N.A.
- Level 2 Cache: N.A.
- Standard Ram: 512kB
- Hard Disk: N.A.
- Price $2000

- October 2009
- Processor: Intel Core Duo
- Processor speed: 3600 MHz
- System Bus Speed 1066 MHz
- Memory registers: 64-bit
- Level 1 Cache: 32kB
- Level 2 Cache: 3MB
- Standard Ram: 4GB (16GB)
- Hard Disk: 500GB
- Price $1200

- January 2021
- Processor: Intel Core i3 (quad)
- Processor speed: 3600 MHz
- System Bus Speed 2000 MHz
- Memory registers: 64-bit
- Level 1 Cache: 32kB per core
- Level 2 Cache: 256kB per core
- Level 3 Cache: 6MB shared
- Standard Ram: 8GB
- Hard Disk: 256GB SSD
- Price $1699

- All the data and instructions for the CPU are in the form of bits, with each bit having a value of 0 or 1. A byte is a sequence of 8 bits.
- An important aspect of a CPU's architecture is the maximum number of bits that can be stored in the registers (data-holding elements accessible to processor). When we talk about a 64-bit architecture, we refer to the size of the registers.
- The largest whole number that can be stored in a single register is 2^N -1. For a 64-bit register, the largest integer is 2^{64} -1 $\cong$ 1.84467 x 10^{19}.

In addition to whole numbers, a number of other important data structures can be expressed in terms of bit. The set of integers (signed whole numbers) {0, ±1, ±2, ±3, …} can be represented in a number of ways as N-bit binary numbers. E.g.

Binary	Unsigned	Sign and magnitude	Ones complement	Twos complement
00000000	0	+0	+0	0
00000001	1	1	1	1
…	…	…	…	…
01111101	125	125	125	125
01111110	126	126	126	126
01111111	127	127	127	127
10000000	128	-0	-127	-128
10000001	129	-1	-126	-127
10000010	130	-2	-125	-126
…	…	…	…	…
11111110	254	-126	-1	-2
11111111	255	-127	-0	-1

Floating Point: An approximation to the set of real numbers in terms of N-bit binary number.

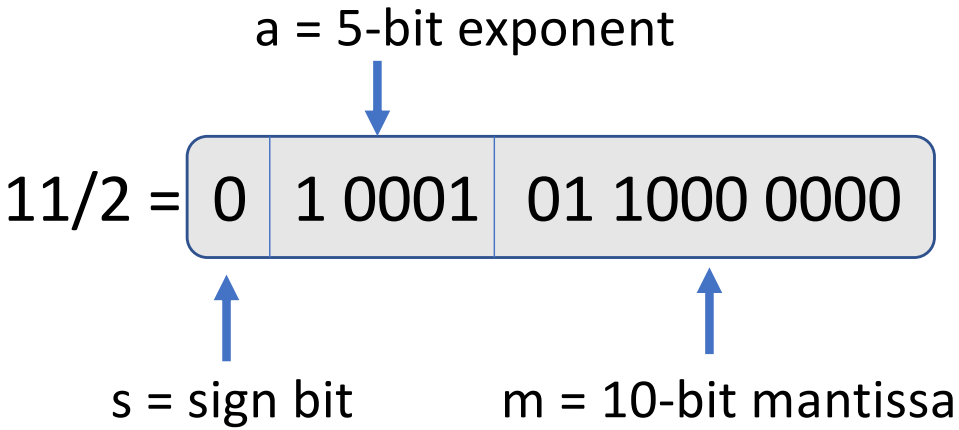
$$x_{float} = (-1)^s \times m \times (10)_2^E$$

	s	m	a	bias
16-bit	1	10	5	15 ($2^4 - 1$)
32-bit	1	23	8	127 ($2^7 - 1$)
64-bit	1	52	11	1023 ($2^{10} - 1$)

m = mantissa
= 1.0111011000
10-bit mantissa

E = exponent
= a - bias

For example, x = 11/2 = $(5.5)_{10} = (1.011)_2 \times (10)_2^2$ may be expressed in terms of a 16-bit binary string as



Floating point representation

- Preserves precision when multiplying and dividing
- Loses precision when adding or subtracting

The set of binary instructions that the machine reads is generally referred to as machine code. While in principle it is possible to write machine code directly, it is usually written in assembly language or a higher level language.

- Assembly Language: Consists of a set of commands written in alphanumeric form that can translated (by an application known as a “assembler”) into instructions in binary format that the CPU can process. Assembly language is typically very CPU specific.
- Programmers will sometimes write critical pieces of code in an application in assembly language to maximize the speed and efficiency of the application.

Higher level languages come in two forms: interpreted and compiled languages:

- Interpreted Languages: These are languages in which the commands are written in a syntax that bears some resemblance to mathematics or standard English. Commands are read in line by line to an application called an “interpreter”. As the interpreter reads in each line, it translates it into machine code and executes the instructions.
- Examples of interpreted languages are Python, perl, awk, lua etc. Interpreted languages are very convenient, but because each line has to be interpreted into machine code each time through, they are not very efficient for large-scale numerical calculations.

- Compiled languages are also written in a syntax that roughly approximates mathematics or English.
- “Code” consisting of one or more text files containing a sequence of commands is read into a compiler that translates the commands into machine code. This is typically a two-step process in which the compiler first generates “object” files. These object files may be “linked” to other pre-existing object files (e.g. from a library or from a previous compilation) and translated into machine code.
- Compiled code can be very efficient. Examples of compiled languages are FORTRAN, C and C++.

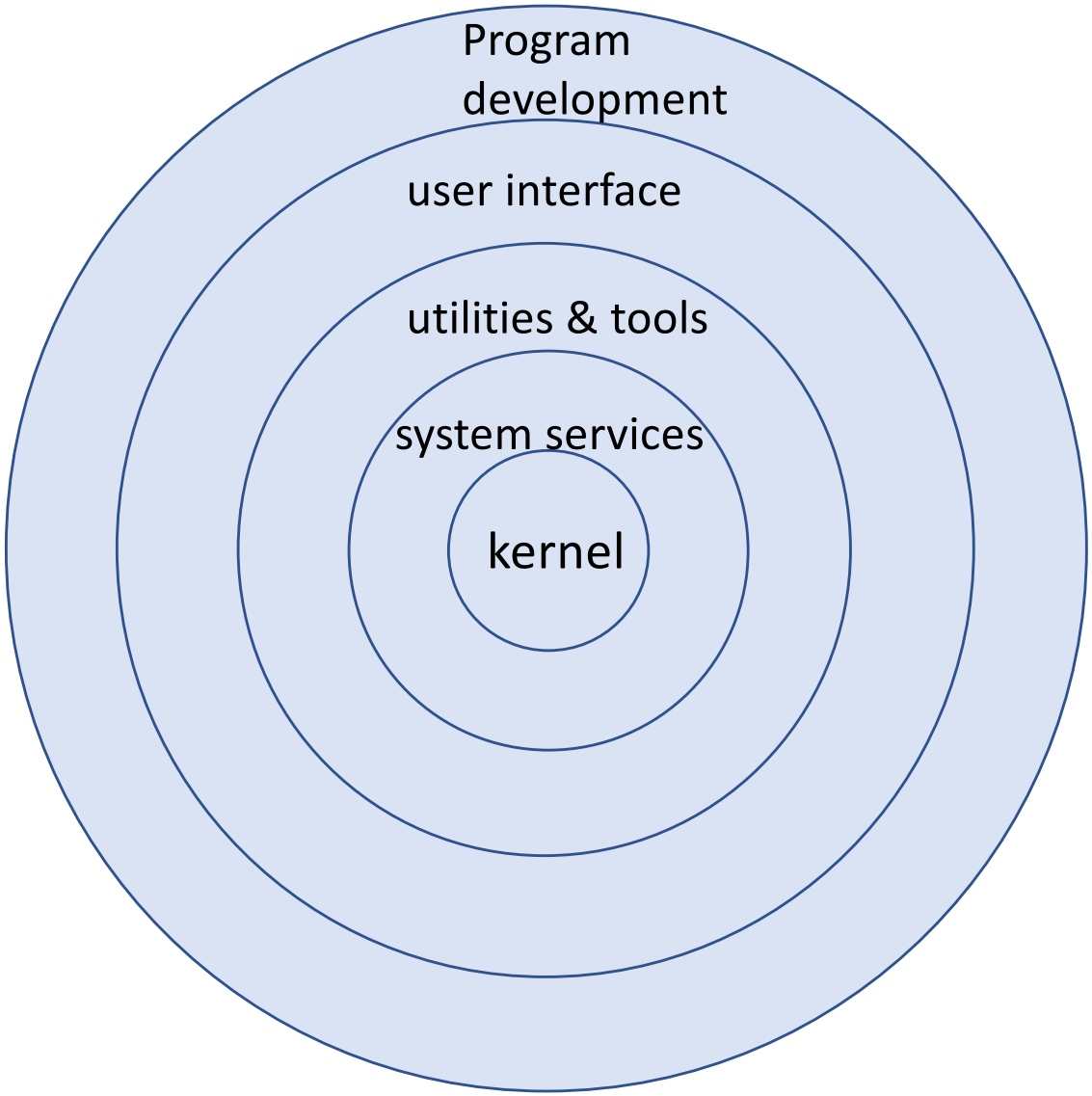
Both interpreted and compiled languages play an important role in scientific computing.

- In scientific computing it is essential that you have knowledge and experience of at least one compiled language like C, C++ or Fortran, and several interpreted languages (e.g. Python, bash, awk, perl, Matlab).
- The distinction between an interpreted language and a compiled language is not always straight forward, as many high-level languages contain elements of both. For example, Python library routines often have C or Fortran compiled code under the hood.

The Operating System

- The operating system (sometimes referred to as the OS for short) is a program that is launched when you start the computer.
- The OS serves as an interface between the peripherals (e.g. keyboard, monitor, network, disk etc).
- Some OS's are proprietary and sometimes tied to a particular class of CPU and/or manufacturer (MS Windows, VMS). Other OS's are partially proprietary (e.g. macOS, AIX, Solaris). Most of these latter ones are based on the UNIX OS.
- An important UNIX variant is Linux. This is non-proprietary and is governed by an open-source licence.

The Linux OS: We will be using this OS in this course.



- kernel: the central component of the Linux OS. It is a bridge between applications and the data processing done at the hardware level.
- System services: drivers for disks, network, printers, monitors etc.
- utilities and tools: assembler
- User interface: shells (bash, csh, tcsh, etc.) X-windows, desktop environment
- Program development: C, C++ and Fortran compilers, editors (vi, emacs, nedit), debugging and profiling tools (dbx, prof)