1 Introduction to Computer Systems

As far as this course is concerned:

> A computer is a *machine* that can be *programmed*.

1.1 Computer Hardware

Modern computers are built around *microprocessors* which are *integrated circuits* (IC) that encapsulate the first three of four basic types of components

1. Arithmetic Logic Unit (ALU)
2. Control Unit
3. Memory
4. Input/Output Devices (I/O)
   such as keyboards, graphic monitor, printer, mouse, network interfaces, etc

Most of the memory is external to the microprocessor. Access to integrated memory is much faster though. Modern processor have a hierarchy of memory with different sizes and access speed. Scientific software explores these for improved efficiency during calculations.

Modern processors may have *auxiliary add-hoc* co-processors, such as mathematical and graphics co-processors. Mathematical co-processors are mostly integrated in the same IC but graphics processors are often found in a separated IC.

It is now very common to find processors with multiple ALUs integrated in a single IC. Each unit is labeled a *core* and multiple cores often share some memory space and controlled by a higher level control unit. Designing general purpose software that can consistently take full advantage of such architectures is still a major research topic.

1.2 Computer Software

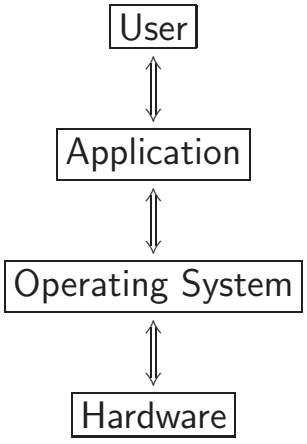
The most basic computer software is the *Operating System* (OS). It provides a common framework for managing diverse hardware and building *Application Software*. Common general purpose operating systems include:

1. Microsoft Windows
2. UNIX
   (a) System V, Solaris, IBM's AIX, HP UX
   (b) BSD, Mac OS X
   (c) Linux

Common tasks performed by application software include:

1. Text and spreadsheet processing
2. Database management
3. Email clients and web browsers
4. Games
5. Computer-Aided-Design (CAD)
6. Scientific calculators
   (a) Matlab
   (b) Mathematica
7. Development tools
   (a) Assemblers
   (b) Compilers and Interpreters
   (c) Debuggers

The following diagram represents the flow of information between these parts:



1.3 Programming languages

Programming languages specify programs to be run by a computer. There are *thousands* of different programming languages! Many more to come. Popular general-purpose high-level programming languages include:

1. BASIC
2. C
3. C++
4. Fortran
5. Java

Programs can be *compiled* or *interpreted*. A compiler or interpreter translates a program written in a programming language into *machine code*, i.e. code that can be run by the processor.

A *compiler* translates a program *before execution* (compile-time).

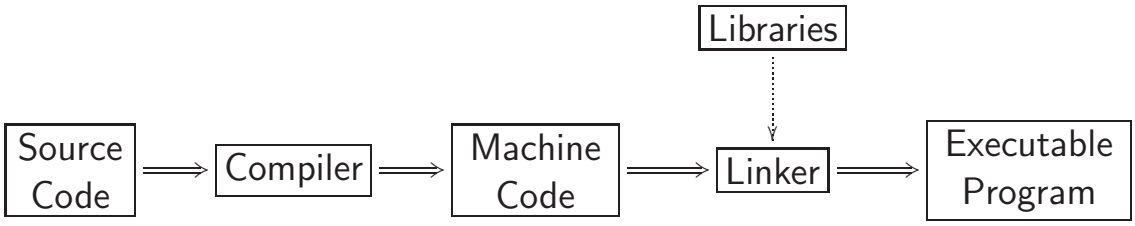
An *interpreter* translates a program *at the time of execution* (run-time).

Some modern programming languages and programmable application software use a combination of these two methods:

1. Interpreted
   (a) BASIC
2. Compiled
   (a) C
   (b) C++
   (c) Fortran
3. Hybrid
   (a) Java

1.4 How does a compiler work?

The following diagram represents the flow of information during program compilation:



An *Executable Program* is not truly executable in the sense that the actual memory address from which the program will be run and possibly the code for some library functions are not known to the compiler at *compile-time*. At *run-time*, the operating system will link missing code from *dynamic libraries* and resolve all memory addresses.