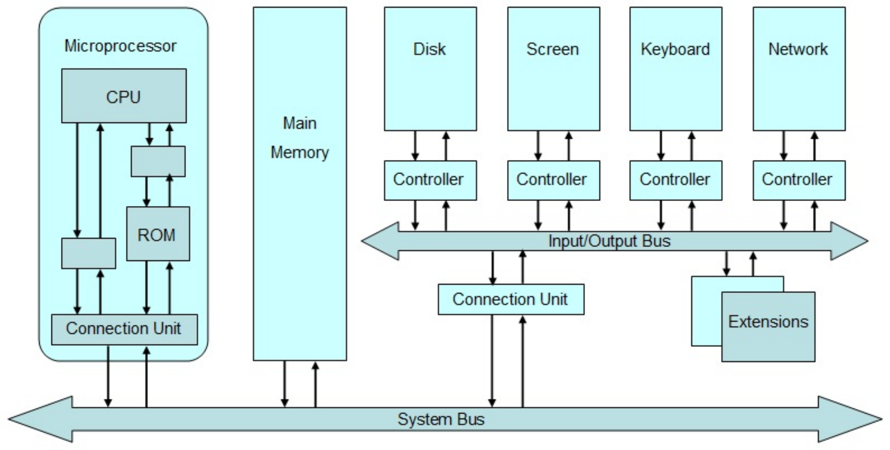
C++ for Science and Engineering COSC3000/6000

2018 Spring Semester

1 Computer Systems

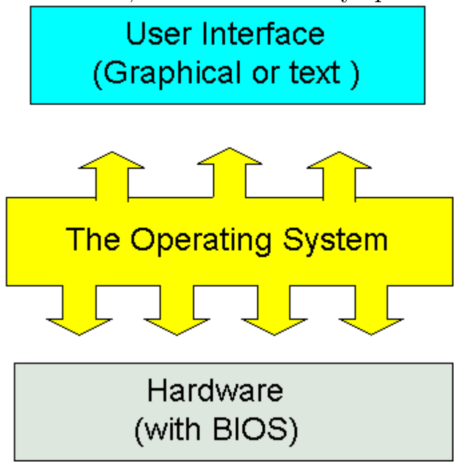


Typical Computer Architecture

- The **CPU** (Central Processing Unit) performs most of the calculations which enable a computer to function, and is sometimes referred to as the "brain" of the computer. It is usually cooled by a heat sink and fan. Most newer CPUs include an on-die Graphics Processing Unit (GPU).
  - Typical capabilities of CPU include
    - add, subtract, multiply, divide, move data from location to location
- The Chipset, which includes the north bridge, mediates communication between the CPU and the other components of the system, including main memory.
- The Random-Access Memory (**RAM**) stores the code and data that are being actively accessed by the CPU.
- The Read-Only Memory (**ROM**) stores the BIOS that runs when the computer is powered on or otherwise begins execution, a process known as Bootstrapping, or "booting" or "booting up". The BIOS (Basic Input Output System) includes boot firmware and power management firmware. Newer motherboards use Unified Extensible Firmware Interface (UEFI) instead of BIOS.
- A computer might have any of types of secondary memory
  - Hard disk
    - Fast
    - Fixed in the computer and not normally removed
  - Floppy disk
    - Slow
    - Easily shared with other computers

- Compact disk
    - Slower than hard disks
    - Easily shared with other computers
    - Can be read only or re-writable

The operating system and the BIOS program routines form the layer on which the user programs. When the PC has to work, an operating system has to be read from a disk. There are many different operating systems to choose from. However, the BIOS is always placed firmly and centrally in the PC hardware.



Common operating systems:

- Windows
- Mac OS (based on Unix)
- Linux/Unix
- iOS
- Android (based on linux)
- Chrome OS (based on linux)
- DOS
- etc.

2 Computer Programs

- A computer program is a set of instruction for a computer to follow.
- The operating system and the BIOS are programs.
- Computer program is often called "**code**".
- The code that CPU can follow is a set of binary numbers (zeros or ones), which is called "**machine language**".
- Human hardly understand machine language.

High-level Languages: C/C++, Java, Pascal, Visual Basic, FORTRAN, COBOL, Lisp, ...

- These high – level languages
    - **Resemble human languages**
    - Are designed to be easy to read and write
    - Use more complicated instructions than the CPU can follow
    - Must be translated to zeros and ones for the CPU to execute a program

2.1 Compile and Link

Compilers translate high-level language to machine language

- Source code
    - the original program in a high level language
- Object code
    - the translated version in machine language

A **Linker** combines

- The object code for the programs we write
  and
- The object code for the pre-compiled routines
  into
- The machine language program the CPU can run

3 Programing and Problem Solving

- Algorithm
  - A sequence of precise instructions which leads to a solution
- Program
  - An algorithm expressed in a language the computer can understand

Program Design

- Programming is a creative process
  - No complete set of rules for creating a program
- Program Design Process
  1. Problem Solving Phase
     - Result is an algorithm that solves the problem
  2. Implementation Phase
     - Result is the algorithm translated into a programming language

3.1 Problem Solving Phase

- Be certain the task is completely specified
  - What is the input?
  - What information is in the output?
  - How is the output organized?
- Develop the algorithm before implementation
  - Experience shows this saves time in getting your program to run.
  - Test the algorithm for correctness

3.2 Implementation Phase

- Translate the algorithm into a programming language
  - Easier as you gain experience with the language
- Compile the source code
  - Locates errors in using the programming language
- Run the program on sample data
  - Verify correctness of results
- Results may require modification of the algorithm and program

4 Overview of C++

- Where did C++ come from?
  - Derived from the C language
  - C was derived from the B language
  - B was derived from the BCPL language
- Why the '++'?
  - ++ is an increment operator in C

```
#include <iostream>
int main(int argc, const char * argv[]) {
        std::cout << "Hello C++\n";
        return 0;
}
```

4.1 #include Directives

- Tells compiler where to find information about items used in the program.
- **iostream** is a library containing definitions of most common functions
- Standard C++ Library reference `http://www.cplusplus.com/reference/`

4.2 main Function

To begin the main function of the program

```
int main(int argc, const char * argv[]) {
```

To end the main function

```
        return 0;
}
```

Main function ends with a return statement

4.3 Program Layout

- Compiler accepts almost any pattern of line-breaks and indentation
- Programmers format programs so they are easy to read
  - Place opening brace '{' and closing brace '}' on a line by themselves
  - Indent statements
  - Use only one statement per line

```
#include <iostream>
int main(int argc, const char * argv[])
{
        std::cout << "Hello C++\n";
        return 0;
}
```

This also works but hard to read.

```
#include <iostream>
int main(int argc, const char * argv[]) {std::cout << "Hello C++\n";return 0;}
```

4.4 Running a C++ Program

- C++ source code is written with a text editor.
  - Any text editor works, but better to use editors having syntax highlighting capability, such as emacs gedit, (or vi). IDE (Integrated Development Environment) is much better.
- The compiler on your system converts source code to object code.
- The linker combines all the object code into an executable program.

5 Testing and Debugging

- Bug
  - A mistake in a program
- Debugging
  - Eliminating mistakes in programs

5.1 Program Errors

- Syntax errors
  - Violation of the grammar rules of the language
  - Discovered by the compiler
    - Error messages may not always show correct location of errors
- Run-time errors
  - Error conditions detected by the computer at run-time
- Logic errors
  - Errors in the program's algorithm
    - Or errors in implementation but the code looks working.
  - Most difficult to diagnose
  - Computer does not recognize an error