Chapter 1

Introduction to Computers, the Internet and the Web

Computer Basics

- A computer consists of **hardware** and **software**
- Computer Hardware
  - Physical components of a computer
  - Processing unit, keyboard, monitor/screen, mouse, hard disk, memory, DVD drive, etc.
- Computer Software
  - The instructions you write to command computers to perform actions and make decisions

Computer Organization

- Computer hardware can be divided into various **logical units** or sections
  - Input Unit
  - Output Unit
  - Central Processing Unit (CPU)
    - Arithmetic logic unit (ALU)
  - Memory Unit
  - Secondary Storage Unit

Computer Organization

- Input Unit
  - Obtains information from the outside world through **input devices**
    - Places that information at the disposal of the other units for processing.
  - Examples of input devices
    - Keyboard, touch screen, mouse/touch pad, joystick
    - Microphones, scanners, cameras, bar code reader
    - GPS device, accelerometer

Computer Organization

- Output unit
  - Takes information the computer has processed and places it on various **output devices** to make it available for use outside the computer
  - Examples of output devices
    - Printer, plotter, screen, speaker, projector
    - Robot, “intelligent” appliance

Computer Organization

- Central processing unit (CPU)
  - The brain of the computer
  - Coordinates and supervises the operation of the other sections
  - Consists of
    - Arithmetic logic unit (ALU)
    - Registers (memory)
    - Control unit (CU)
  - Many of todays computers have multi-core processors
    - Multiple CPUs on a single integrated-circuit chip

Computer Organization

- Arithmetic logic unit (ALU)
  - Performs math operations (addition, subtraction, multiplication, division, etc.) and logic operations (AND, OR, NOT, etc.)
- Registers
  - Used for fast access of data by other CPU components
- Control unit (CU)
  - Controls the other parts of the CPU in order to execute instructions
    - *Fetches* instructions and data from main memory
    - Uses the ALU and registers to *execute* the instructions
    - Called a *fetch-execute* cycle

Computer Organization

- Memory unit (also called main or primary memory)
  - Rapid-access, relatively low-capacity “warehouse” section that is used to store computer instructions and computer information or data
    - Each location in memory has an address, so data in a particular location can be accessed
  - Data in memory is **volatile**
  - RAM (Random access memory)

Computer Organization

- Secondary Storage Unit
  - Long-term, high-capacity “warehousing” section
  - Examples of secondary storage devices
    - Hard drive, DVD drive, USB flash drive
    - Storage capacity usually expressed in terms of GB or TB
  - Data on secondary storage devices is **persistent**
  - Cheaper, but takes much longer for the computer to access than main memory
  - Input/output and secondary storage devices are also called **peripherals**

Computer Software

- The programs that allow the computer hardware to operate
- **Computer program**: A set of instructions for the computer to perform
- Two types of software
  - System software
    - Operating systems, utilities, language translators (assembler, compiler, interpreter)
  - Application software
    - Word processors, database management systems (DBMSs), graphics programs, games, payroll systems, etc.

Bits and Bytes

- Data and instructions – everything on a computer – is stored in memory in **bits**, or **b**inary dig**its**
- Bit – the smallest data item in a computer
  - Can be in one of two states (1 or 0)
- 8 bits is called a **byte**
- 4 bits is called a **nibble (or nybble)**
- Bytes can be combined to form **words**
  - The length of a word is system dependent

Bits and Bytes

- Letters, characters, numbers, instructions, etc., can all be represented by a combination of 0s and 1s.
  - For example, the letter F can be represented by 01000110, which is the number 70 in decimal (base 10)
  - This representation is based on ASCII (American Standard Code for Information Interchange) format
    - Each letter, number, or character is represented using 1 byte
    - See Appendix B for the ASCII character set

Bits and Bytes

- Unicode is another character coding system
  - Universal Character Encoding
  - Assigns a unique number to every character of every written language
  - Length depends on the specific encoding used
    - UTF-8 uses one byte for any ASCII character (same values) and up to 4 bytes for other characters

C Programming Language

- Developed by Dennis Ritchie at Bell Laboratories
  - Originally implemented in 1972
  - Evolved from B
    - B was developed and used by Ken Thompson to create early versions of the UNIX operation system
- Many of today's leading operating systems are written in C and/or C++
- C is *mostly* hardware independent
  - With careful design, it's possible to write C programs that are portable to most computers

C Programming Language

- C was built for systems that demand performance
  - Operating systems, embedded systems, real-time systems, and communication systems
  - C remains the most widely used embedded programming language
- C is a **high-level** language
  - The 3 language types are machine, assembly, and high-level
- C is a compiled language
  - A **compiler** converts the source code into machine language

C Development Process

- Phases of development:
  - **Edit**: Programmer writes source code in an editor (.c)
  - **Preprocess**: Preprocessor processes the source code
  - **Compile**: Compiler translates the pre-processed source code into object code (.obj) – machine language
  - **Link**: Linker links the object code with libraries and creates an executable file (.exe)
- Phases of execution:
  - **Load**: The loader moves the executable program from secondary storage into memory
  - **Execute**: The CPU takes each instruction and executes it

C Programming Language

- “Writing in C or C++ is like running a chain saw with all the safety guards removed.”

  (Bob Gray)

Chapter 2

Introduction to C Programming

A Simple C Program

```
#include <stdio.h>

int main( void )

{

     printf(“Hello, world!”);

     return 0;

}
```

```
#include <stdio.h>
int main( void )
{
      printf("Hello, world!");
      return 0;
}
```

- Preprocessor directive
  - Tells the compiler your program uses a function defined in the header file `stdio.h`
  - The preprocessor takes the appropriate code from `stdio.h` and combines it with your program, which is sent to the compiler
  - Preprocessor directives always start with #

```
#include <stdio.h>
int main( void )
{
      printf("Hello, world!");
      return 0;
}
```

- Preprocessor directive
  - `stdio.h` is a built-in header file, which provides standard input/output functions
  - For built-in C header files, use `< >` around the name of the file
  - If you try to use `printf` in a program without the `#include <stdio.h>` statement, you will get a compiler error

```
#include <stdio.h>
int main( void )
{
      printf("Hello, world!");
      return 0;
}
```

- Function declaration
  - Every C program must have a `main` function
  - Note: C is **case sensitive**
    - `int Main( void )` will not work!

```
#include <stdio.h>
int main( void )
{
      printf("Hello, world!");
      return 0;
}
```

- Pairs of opening and closing braces are used to signal where a block of code (such as a function) begins and ends
- Every opening brace { must have a corresponding closing brace } or you will get a compiler error

```
#include <stdio.h>
int main( void )
{
      printf("Hello, world!");
      return 0;
}
```

- This is a C **statement**
  - A C program is a series of C statements
  - Every C statement ends in a semi-colon ;
  - The `printf("Hello, world!");` statement is a function call
  - `printf` is used to print information to the screen

```
#include <stdio.h>
int main( void )
{
      printf("Hello, world!");
      return 0;
}
```

- `return 0;` is a C statement
  - Used to indicate our main function completed execution with no problem
- What is the output of the program?

Printing to the screen

- The `printf` function is used in C to print text to the screen
- Simplest form of `printf`

  `printf(`*string*`);`

  - *string* is any text you would like to print
  - Text must be in double quotes
- Example

  `printf("C programming is fun!");`

Printing to the screen

- Each `printf` statement does **not** automatically advance to a new line
  - You must explicitly tell the program when you want to move to a new line in the output
  - For example, what is the output of the following program?

```
#include <stdio.h>
int main( void )
{
        printf("Hello!");
        printf("Goodbye!");
        return 0;
}
```

Printing to the screen

- To move to a new line when printing:
  - Use the *escape sequence* **\n** inside the double quotes
- Modify program to print a new line between Hello! and Goodbye!

```
#include <stdio.h>
int main( void )
{
        printf("Hello!\n");
        printf("Goodbye!");
        return 0;
}
```

Escape Sequences

- Whenever a backslash (`\`) is encountered in a `printf` statement, it indicates the character following the `\` has a special meaning.
- Commonly used escape sequences

Escape Sequence	Meaning
`\n`	Advance to a new line
`\a`	Beep
`\t`	Print a tab
`\\`	Print a single backslash
`%%`	Print a percent sign

Escape Sequences

- So, what if you want to print a \?
  - If you want to print a \, you must use the escape sequence \\
  - What is the output of the following program?

```
#include <stdio.h>
int main( void )
{
        printf("\\\\\\\");
        return 0;
}
```

In-Class Assignment #1