THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

COMP 110
Introduction to Programming

Fall 2014

Time: TR 9:30 – 10:45

Room: G100 (Genome Sciences Bldg.)

Jay Aikat

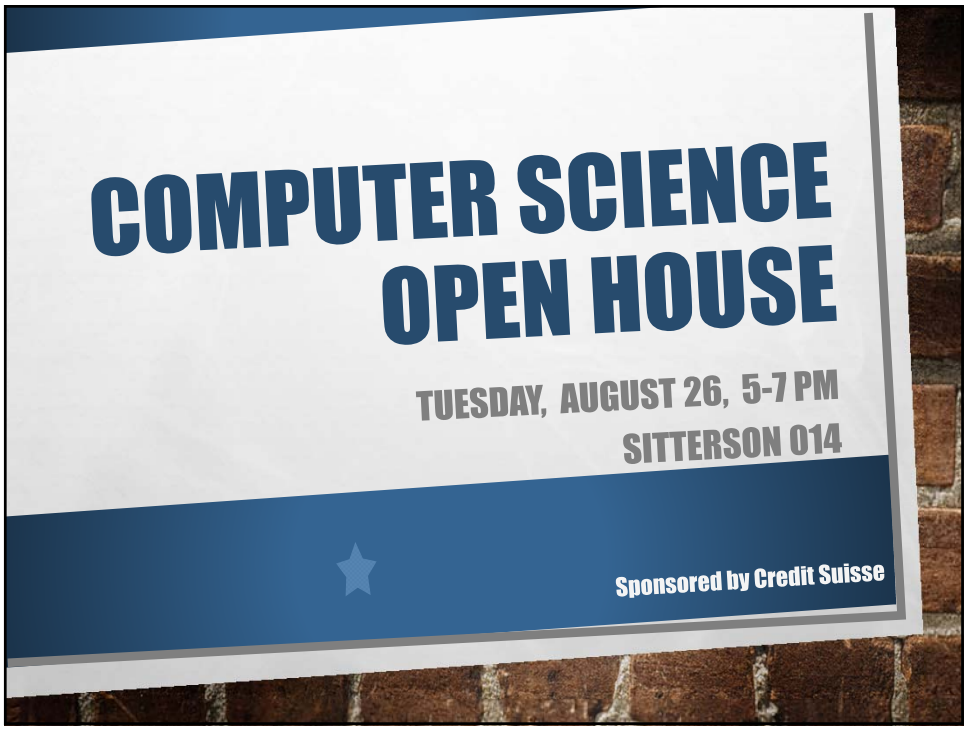
FB 314, *aikat@cs.unc.edu*

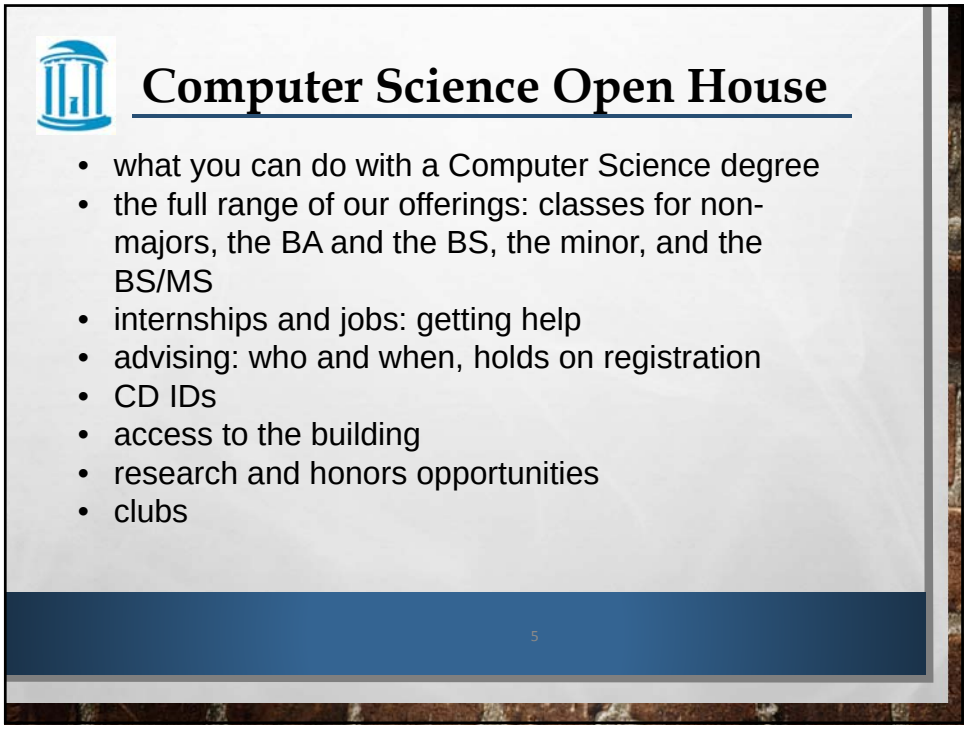
Previous Class

- What did we discuss?

Today

- Announcements
  - Look out for TA office hours and lab timings on email (did you receive one yesterday?)
  - Assignment1: **due Wed, Aug 27 11:55 PM**
  - **Sakai**
- CS Club – Matthew Lemming
- Computer basics
- Programming basics
- Your first program



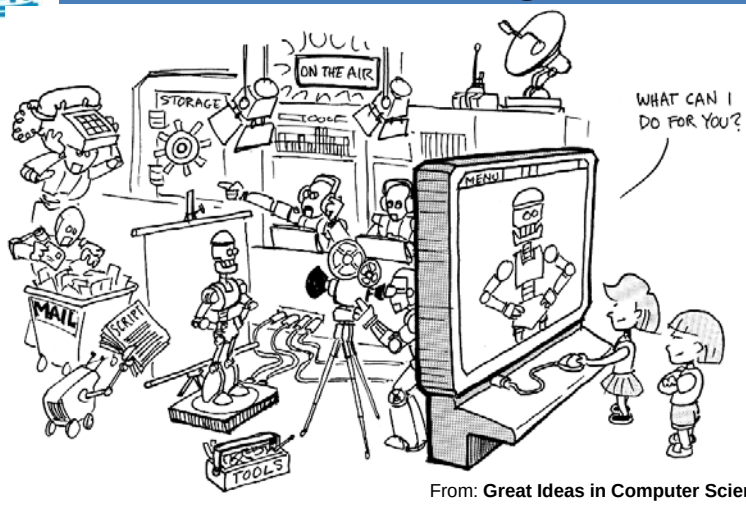


Software

- Java and Eclipse
  -- See the course website for detailed installation instructions

- How many got this installed?

Computers – not Magic



From: **Great Ideas in Computer Science by JAVA,**
by *A. W. Biermann and D. Ramm*,
The MIT Press 2002.

Hardware vs Software (concretely)

- Hardware
  - CPU, Memory, disks
- Software - programs that give instructions to the computer
  - Windows 7, Google Chrome, Games, Eclipse
  - Microsoft Word

Hardware vs Software (abstractly)

- Software
  - An organized collection of instructions
- Hardware
  - Circuits that execute, store and interact with instructions
    - Execution: CPU
    - Storage: Memory
    - Interaction: Peripherals, like keyboards, monitors, networks

Software Categories

- Operating System
  - first program to load when a computer is turned on
  - controls all machine activities
  - provides the user interface to the computer
  - manages computer resources, such as the CPU, memory, and hard drive
  - examples: Windows 8, Linux, Mac OS X
- Application
  - generic term for any other kind of software
  - examples: word processors, missile control systems, games

Operating System (OS)

- OS monitors overall activity of the computer and provides services
- Written using programming language
- Example services:
  - memory management
  - input/output
  - storage management

Application Programs

- Written using programming languages
- Perform a specific task
- Run by the OS
- Example programs:
  - Browsers
  - Word Processors
  - Spreadsheets
  - Games

Questions

Classify the following pieces of software as *operating system* or *application*:

1. Microsoft Windows 8
2. Microsoft PowerPoint
3. Linux
4. Your COMP 110 programs

Instructions

- An instruction is a sequences of 0's and 1's that represents a single operation on the computer

	Instruction	*Data*	*Data*
– Example:	00000101	00000001	00000010
– Means:	ADD	1	2

  - What is the output? -- 3
- These 0's and 1's are called **bits**
  - Why only 0 and 1?
    - Because it is easier to make an electrical device that has only two stable states

CPU (Central Processing Unit)

- It is the "brain" of the computer
  - CPU executes the instructions
  - CPU's working routine
    - read instructions and data from memory
    - do calculation
    - write calculation results back to memory
- Intel Core i7 **3.4 GHz**
  - Executes *at most* 3,400,000,000 instructions per second

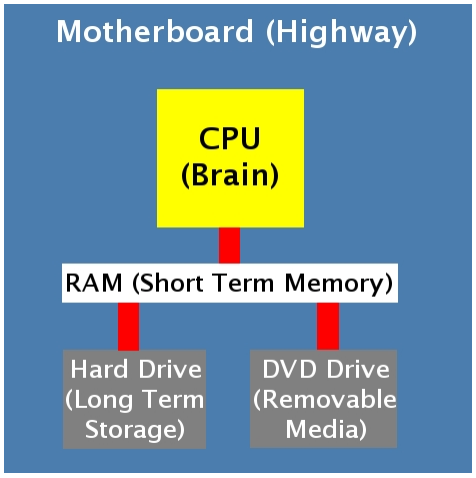
Central Processing Unit (CPU)

- Control Unit (CU)
  - "the brain" of the CPU
- Program Counter (PC)
  - points to the next instruction to be executed
- Instruction Register (IR)
  - holds the currently executing instruction
- Arithmetic Logic Unit (ALU)
  - carries out all arithmetic and logical ops
- Accumulator (ACC)
  - holds the results of the operations performed by the ALU

Memory

- Holds instructions and data for the computer
  - How much the "brain" can remember
- Main Memory
  - For intermediate calculations (program you are running)
  - Disappears when you shut down your computer
- Secondary Memory
  - Hard drives, CDs, Flash drives
  - Exists until you delete it

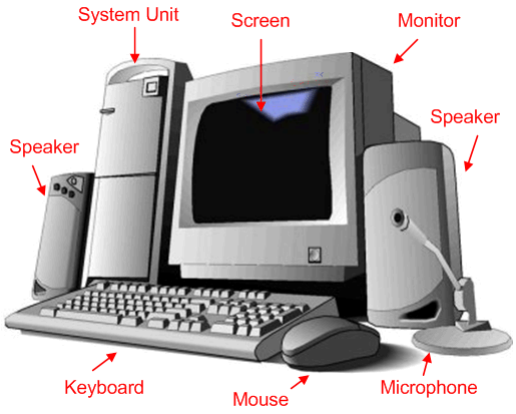
Memory



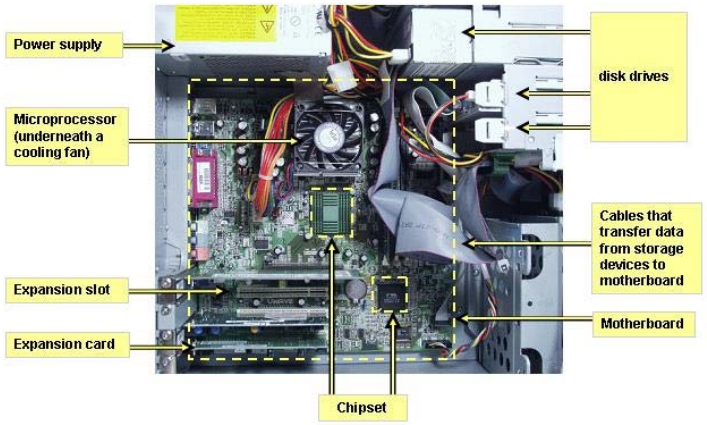
GB? MB? KB?

- 1 bit = 0 or 1
- 1 byte = 8 bits
  - Smallest addressable unit of memory
- Kilo, Mega, Giga, Tera
  - 1 KB = 1,000 bytes (1 thousand bytes)
  - 1 MB = 1,000 KB = 1,000,000 bytes (1 million bytes)
  - 1 GB = 1,000 MB = 1,000,000,000 bytes (1 billion bytes)
    - The same for GHz (1 Giga Hertz)
  - 1 TB = 1,000 GB = 1,000,000,000,000 bytes!

A Computer



A Computer Motherboard



21

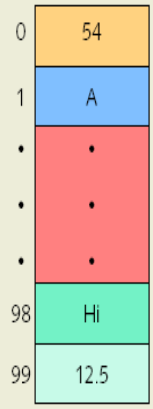
Main Memory

- Memory address
  - To locate certain memory positions
  - CPU fetches data according to memory address
- *Another interesting fact: characters are also saved in bits, and so does everything*

Memory address	Memory content	
.	.	
.	.	
.	.	
2000	01001010	Encoding for character 'J'
2001	01100001	Encoding for character 'a'
2002	01110110	Encoding for character 'v'
2003	01100001	Encoding for character 'a'
2004	00000011	Encoding for number 3
.	.	

Main Memory with 100 Cells

Each memory cell has a numeric *address*, which uniquely identifies it



CPU and Main Memory

All programs must be brought into main memory before execution

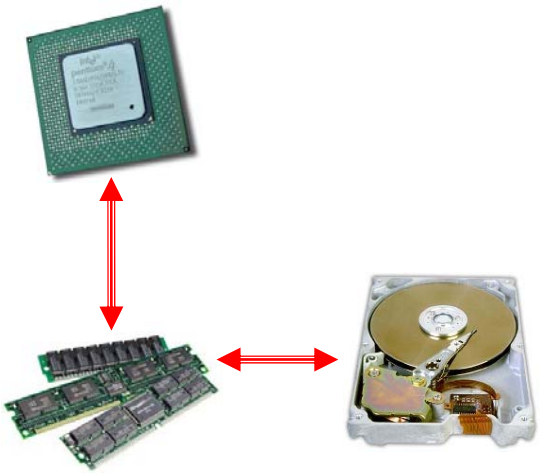
Chip that executes program instructions

Primary storage area for programs and data that are in active use (RAM)

Secondary Storage

Secondary memory devices provide long-term storage

Information is moved between main memory and secondary memory as needed



Secondary Storage

- Provides permanent storage for information
- Retains information even when power is off
- Examples of secondary storage:
  - Hard Disks
  - USB Drives
  - DVDs
  - CDs
  - Tapes

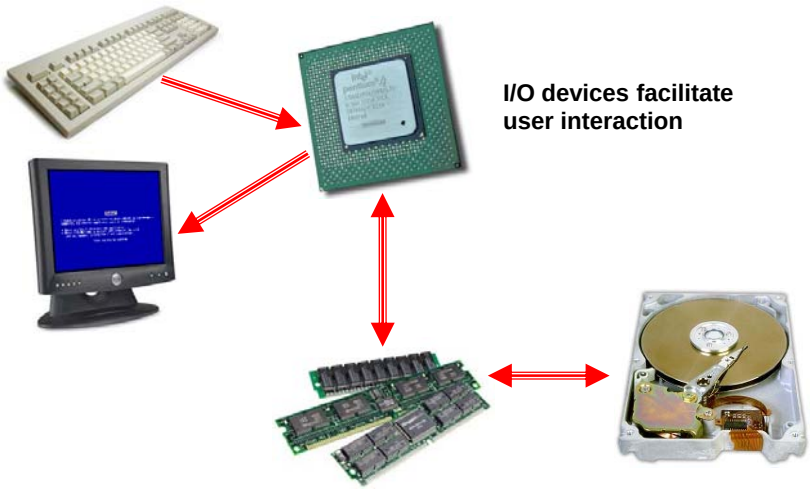
Peripherals





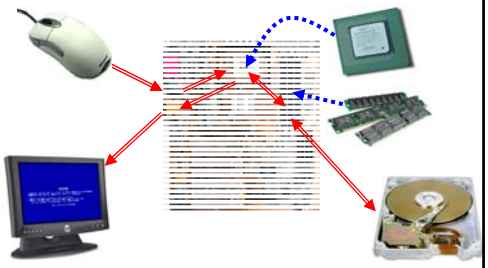

- Input devices
  - Keyboard, mouse, game controller……
  - When they get input, they save that at certain memory addresses
- Output devices
  - Monitor, speaker, printer……
  - They are projected to certain memory addresses
  - When CPU wants to output, it writes to those addresses
- **CPU sees everything as memory**

Input/Output Devices



Opening Notepad

- Use the mouse to select Notepad
- The CPU requests the Notepad application
- Notepad is loaded from the hard drive to main memory
- The CPU reads instructions from main memory and executes them one at a time
- Notepad is displayed on your monitor

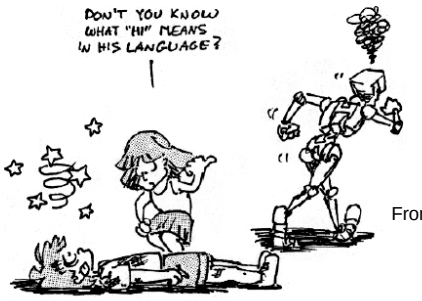


Programs

- Set of instructions for a CPU to follow
  - Also known as software
- You will be writing programs
  - We will write one today
- Our programs will be in **Java**

Programming Languages

- Why do we need languages when we have instructions?
  - Too hard for humans to write bits directly



From: **Great Ideas in Computer Science by JAVA,** by *A. W. Biermann and D. Ramm*, The MIT Press 2002.

Programming Languages

- Different languages are good at different aspects
  - C/C++: close to instructions, runs fast
  - Matlab: good at scientific computation
  - Python: relatively easy, fast development
- We choose Java
  - Not because…
    - It's the best language (there is no such thing!), or
    - It's easiest to learn (not!)
  - But because
    - It's widely used, incorporates (most) modern features

Programming languages

- Java - General purpose. Best for writing larger programs.
- Javascript - No relationship to Java! Very specialized. Runs in your web browser and adds advanced behavior to web pages.
- Python - General purpose but a scripting language. Much easier to write small programs (compared to Java), but much less appropriate for larger ones.
- Matlab - Very different from the other 3 languages. Very powerful but highly specialized. Excellent for solving equations, graphing data, etc. since much less programming is required.

From Languages to Instructions

- The translator is called a **compiler**
  - It is also a program
  - From human-readable to machine-readable

From Java to Machine Language

- Computers understand only 0 and 1 (*machine language*)
- *Compiler* translates source code into machine code

- Java *compiler* translates source code (file ending in .java) into *bytecode* (file ending in .class)
  - bytecode is portable (not machine-specific)
- Java *interpreter* reads and executes bytecode
  - different Java interpreters for different types of CPUs and operating systems (OS)
    - Intel/Windows, Motorola/Mac OS X, Intel/Linux

Programming Languages

- Programming languages have rules of grammar just as English does

- ***syntax rules*** **-** which statements are legal and which are not

- ***semantic rules*** - determine the meaning of the instructions

- ***token*** - smallest individual unit of a program
  - special symbols
  - word symbols
  - identifiers

Special Symbols

```
+     -     *     /

.     ;     ?     ,

<=    !=    ==    >=
```


Word Symbols (aka reserved words or keywords)

- `int`
- `float`
- `double`
- `char`
- `void`
- `public`
- `static`
- `throws`
- `return`

- reserved words are always all lowercase
- each word symbol is considered to be a single symbol
- cannot be used for anything other than their intended purpose in a program

Identifiers

- Names of things (variables, constants, methods) in your programs
- Can be composed of any combination of letters, digits, underscore (_), and dollar sign ($)
- Cannot begin with a digit
- May be any length
- Java is ***case-sensitive***
  - Total, total, and TOTAL are different identifiers

Illegal Identifiers

Identifier	Description
`employee Salary`	There can be no space between `employee` and `Salary`.
`Hello!`	The exclamation mark cannot be used in an identifier.
`one+two`	The symbol + cannot be used in an identifier.
`2nd`	An identifier cannot begin with a digit.

Questions

Classify the following as legal or illegal identifiers:

1. My First Program — illegal
2. my1stProgram — legal
3. 1stProgram — illegal
4. $money — legal
5. an_identifier — legal
6. Jane'sProgram — illegal

Primitive Data Types

What is a Data Type?

- A set of values and the operations that can be performed on those values
- *Primitive data* are fundamental values such as numbers and characters
- Operations are performed on primitive types using built-in operators

Primitive Data Types

- 8 primitive data types in Java
  - 4 represent integers
    - **byte, short, int, long**
  - 2 represent floating point numbers
    - **float, double**
  - 1 represents characters
    - **char**
  - 1 represents boolean values
    - **boolean**

Primitive Data Types (Numeric Types)

- The difference between the various numeric primitive types is their size, and therefore the values they can store:

Type	Storage	Min Value	Max Value
byte	8 bits	-128	127
short	16 bits	-32,768	32,767
int	32 bits	-2,147,483,648	2,147,483,647
long	64 bits	$< -9 \times 10^{18}$	$> 9 \times 10^{18}$
float	32 bits	$+/- 3.4 \times 10^{38}$ with 7 significant digits	
double	64 bits	$+/- 1.7 \times 10^{308}$ with 15 significant digits	

Integers

- Examples: -6728, -67, 0, 78, 36782
- Positive integers do not have a '+' sign in front of them (but they can)
- No commas are used in an integer
  - commas in Java are used to separate items in a list

Primitive Data Types (Characters)

- A **char** stores a single character from the *Unicode character set*
  - an ordered list of characters, and each character corresponds to a unique number
  - uses 16 bits per character, allowing for 65,536 unique characters
- Character literals are delimited by single quotes:

```
'a'   'X'   '7'   ' '   '$'   ','   '\n'
```

newline character
(we'll discuss later)

Primitive Data Types (Boolean)

- Only two valid values
  - true or false
  - uses 1 bit for storage
- Represent any situation that has 2 states
  - on - off
  - true - false
- **true** and **false** are reserved words

Arithmetic Expressions

- *Expression* - a combination of one or more operands and their operators
- *Arithmetic expressions* compute numeric results and make use of the arithmetic operators:

Addition	**+**
Subtraction	**-**
Multiplication	*****
Division	**/**
Remainder	**%**

- If either or both operands associated with an arithmetic operator are floating point, the result is a floating point

Division and Remainder

- If both operands to the division operator (/) are integers, the result is an integer (the fractional part is discarded)

```
14 / 3        equals?    4

8 / 12        equals?    0
```

- The remainder, or **mod**ulus, operator (%) returns the remainder after dividing the second operand into the first (only works with integer types)

```
14 % 3        equals?    2

8 % 12        equals?    8
```


Unary vs. Binary Operators

- Unary operators
  - has only one operand
  - example: - (negative, not subtraction)

    -5

- Binary operators
  - has two operands
  - example: - (subtraction)

    5 - 3

Operator Precedence

- Determines the order in which operators are evaluated:
  1. multiplication, division, and remainder
  2. addition, subtraction, and string concatenation
  3. arithmetic operators with the same precedence are evaluated from left to right
- Parentheses can be used to force the evaluation order (just like in math)

Operator Precedence (PEMDAS)

- Parentheses: 6 * (5 + 7) vs. 6 * 5 + 7
- Exponents (powers, roots – 2^5 $36^{1/2}$)
- Multiplication / Division / Mod
- Addition / Subtraction
- Left to right
- Which is these is correct?

  30 / 5 * 3 = 6 * 3 = 18 ← **this one!**

  30 / 5 * 3 = 30 / 15 = 2

Operator Precedence

- What is the order of evaluation in the following expressions?

```
a + b + c + d + e        a + b * c - d / e
  1   2   3   4            3   1   4   2

      a / (b + c) - d % e
        2     1     4   3

    a / (b * (c + (d - e)))
      4     3     2     1
```


Integral Expressions

- All operands are integers
- Result is an integer

- Examples:

  2 + 3 * 5

  3 + x – y / 7

  x + 2 * (y – z) + 18

Floating-point Expressions

- All operands are floating-point numbers
- Result is a floating-point

- Examples:

  12.8 * 17.5 – 34.50

  x * 10.5 + y - 16.2

  7.0 / 3.5

Mixed Expressions

- Operands of different types
- Examples:

  2 + 3.5

  6 / 4 + 3.9
- Integer operands yield an integer result
- Floating-point operands yield a floating-point result
- If both types of operands are present, the result is a floating-point number
  - **implicit type coercion**
- Precedence rules are followed

Our First Program

```
public class HelloWorld
{
  public static void main(String[] args)
  {
    System.out.println("Hello World");
  }
}
```


Next class (Tue, Aug 26)

- Binary representation
- Program in class: Adding two numbers
- Assignment1 DUE Wed, Aug 27

→ Reading Assignment: Chapter 1.1 and 1.2