Course Wrap-up

# Lecture Topics

* Course review
* From LC-3 to modern microprocessor architecture

# Course review

* This course focuses on C programming, where each new C concept is introduced based on the fundamental concepts described in ECE120.
  + We cover basic programming concepts, functions, arrays, pointers, I/O, recursion, simple data structures, and concepts in object-oriented programming.
  + A bottom-up understanding of computing systems has proven more successful in helping students to understand advanced concepts in computing that follow in the ECE curriculum.
* In this course, we covered programming concepts & data structures from the bottom-up
  + Programming concepts in LC-3
    - Systematic decomposition (continuation from ECE 120)
    - Assembly language programming and process (continuation from ECE 120)
      * Operate, data movement, and control instructions
      * Pseudo-ops
      * Assembly process
    - I/O Abstractions
      * Memory-mapped I/O
        + input from the keyboard
        + output to the monitor
    - Subroutines
      * invocation
      * coller-/collee-save
    - Interrupts and exceptions
    - Stacks abstraction and its use in assembly
      * For problem solving
      * For function invocation
  + Introduction to C
    - built-in data types, operators, scope (review from ECE 120)
      * their LC-3 assembly language implementation
    - basic constructs (review from ECE 120)
      * sequential
      * conditional (if, if-else, switch)
      * iterative (for, while, do-while)
      * their LC-3 assembly language implementation
    - Functions
      * Syntax
      * Stack frames/activation records in LC-3 assembly
    - Pointers and Arrays
    - I/O in C
      * streams and buffers
      * file I/O
    - User-defined data types (structs)
    - Dynamic memory allocation
  + Testing and Debugging
    - Ad-hoc (using printf)
    - intro to gdb
    - intro to valgrind
  + Programming concepts in C
    - Recursion, recursion with backtracking
    - Sorting (sort by insertion) and searching (binary search)
    - Linked data structures
      * Linked lists
        + Traversal, search, insertion, deletion
        + Stack
        + queue
      * trees
        + Traversal, search, insertion, deletion
        + Binary-search tree (BST)
        + Depth-first search (DFS)
        + Breadth-first search (BFS)
  + Object Oriented (OO) concepts in C++
    - classes
    - inheritance, polymorphism
    - constructors and destructors
    - function and operator overload
    - templates
    - friends
* You have implemented several substantial software projects:
  + MP1 - Printing a Histogram
    - I/O in LC-3 assembly
  + MP2- Stack Calculator
    - Subroutines in LC-3 assembly
    - postfix expressions
    - stack
  + MP3 - Low Pass Filter
    - Implemented second-order low-pass filter using the finite difference method
    - Exercised iterative and conditional constructs in C
  + MP4 - Polynomial root finding
    - Functions in C
  + MP5 - Codebreaker game
    - Arrays and pointers
    - Proper code structure (header file, implementation file)
    - Random numbers generation in C
    - Debugging with *gdb*
  + MP6- Image Editor
    - 2D arrays
    - Row-major order
    - Image processing
    - Compiling with *make* utility
  + MP7 - Sudoku
    - 2D arrays
    - Recursion with backtracking
  + MP8 - 2048
    - Structures
  + MP9 - Maze Solver
    - recursive depth-first search
    - Dynamic memory allocation
    - Memory problems debugging with *valgrind*
  + MP10 - Sparse Matrix
    - Sparse matrix concept
    - Linked lists
    - File I/O
  + MP11 - VLSI Floorplan
    - Floorplan model
    - Tree data structure
    - Postfix tree traversal
  + MP12 - C++ with Inheritance
    - Classes
    - Inheritance
    - Operator overload
* You have learned to use industry-standard tools:
  + gcc
  + gdb
  + valgrind
  + make
  + svn
* We expect you to
  + Understand how statements written in high-level language such as C are transformed into machine code. Be able to perform such a transformation manually.
  + Understand the idea of scope and storage for variables, and the role of types in high-level languages in providing information to the compiler.
  + Understand the stack abstraction and the notion of a calling convention and its role in supporting the transfer of information between a caller and a subroutine.
  + Understand the concepts of arrays and pointers and their representations in memory. Be able to use arrays and pointers for problem solving.
  + Be able to develop and use data structures for representing and aggregating information.
  + Be able to use dynamic memory allocation for storing values and object sin memory.
  + Understand the value of recursion as a problem-solving tool and be able to apply it for solving math and logical problems.
  + Be able to test and debug programs written in C using standard debugging tools and techniques.
  + Be familiar with the concepts of object-oriented programming and be able to implement simple classes in C++ using such concepts as constructors, destructors, function and operator overload, inheritance.