CST8152 - Compilers
Article #1
What is a Compiler? What is a Programming Language?
Brief History

A Compiler (the term was coined by Grace Murrey Hopper in early 50s) is a program that runs on some computer architecture under some operating system and transforms (translates) an input program (source program) written in some programming language into an output program (target program) expressed in different programming language.

Programming language compilers are part of a more general category of language manipulation tools or programs – Language Processors. Language processors include natural language translators and interpreters, text editors, text-to-speech and speech to text coverers, spell and grammar checkers and some others language tools. Programming language interpreters are subset of programming language compilers.

A Programming Language is a notational system for describing computations in machine-readable and human-readable form.

Computation in general is any process that can be carried by a computer. Programming Languages must provide two types of abstractions: data abstractions and control abstractions.

Why Study Compilers?

Compiler construction is a very specialized computer science and system programming field – there are relatively limited number of programmers involved in writing compilers. So why should a computer science student spend time to learn exactly how a compiler is designed and written? There are three main reasons:

- **Compilers are used by all programmers**. While most programmers will not ever write a full-fledged compiler, they use one every day. Knowledge about how compilers do their work can be used to write more efficient and error free code in a high-level language. As they say "*A good craftsman should know his tools.*" This is one of the most compelling to study compilers.

- **Compilers elements and techniques are used in almost every application**. Many programmers use compiler components for programming other applications. There is a good chance that a programmer will need to write a compiler or interpreter for a domain-specific language. Writing a parser for XML, HTML, or some other structured data file is a common task. Scanning and parsing a command or user input line is a very common task. Looking for a specific word or sentence in a text is a very common task.

- **Compilers are an excellent "capstone" or "focal" programming project.** Writing a compiler requires an understanding of almost all of the basic computer science subfields. To write a compiler, a programmer need to know regular expressions, grammars, finite automata theory, programming paradigms, operating systems, computer architecture, a large range of data structures and algorithms, some programming languages, and a good and sound software engineering principles.

- **Finally**, it is considered a topic that you should know in order to be "well-educated" or "well-versed" in computer science.

<u>Brief History:</u>

First computer and first programmer:
1830-40 – Analytical Engine invented by Charles Babbage. His wife Ada Augusta, Countess Lovelace (daughter of Lord Byron) wrote the first programs.

1950 - FORTRAN (1954-1957), COBOL (1959-1960). Algol60, LISP (LISt Processor)
1960 – PL/1, SNOBOL (StriNg Oriented symbolic Language), Simula, BASIC
1970 – Pascal (1971), C (1972)
1980 - Ada, Modula, Smalltalk-80 (1972-1980), C++ (1980-1985), Objective C, Object Pascal, Eiffel, Oberon, Scheme
1990 – Java, Haskell, Javascript, php, Perl, Python, Ruby, Lua…
2000 - C#, Scala , F#, Groovy, Go, D, R, Clojure, Swift…

If you want to learn more about the history of programming languages, visit the following web site:
http://www.oreilly.com/pub/a/oreilly/news/languageposter_0504.html

Programming Languages classifications: general-purpose and special purpose.

General-purpose

- Computational Paradigms
  - **Imperative** or **Procedural Programming** (based on John von Neumann's computer architecture). Examples:
    FORTRAN, COBOL, ALGOL, BASIC, PL, SNOBOL, C, ADA, Modula.
  - **Functional Programming** – based on functions as data values and lambda calculus. Examples:
    LISP, Scheme, ML (Meta Language), Miranda, Haskell, F#, Clojure, Java.
  - **Logic Programming** – based on symbolic logic or first-order predicate calculus. A logic programming language is a notational system for writing logical statements together with specified algorithms for implementing inference rules. Examples:
    Prolog
  - **Object-Oriented Programming** – based on the concept of an object, which can be described as a collection of memory locations together with all the operations (methods) that can change the value of these memory locations. Examples:
    SIMULA, Smaltalk, C++, Objective C, Modula-3, Oberon, Eiffel, Object Pascal, Java, C#.
  - **Parallel Programming Languages** – allow for concurrent execution of computational processes. Example: ADA
  - **Scripting Languages** – combine together utilities, libraries, operating system commands into a program.
    Perl, Python, Tcl/Tk, Javascript, Rexx, Visual Basic,php
  - **Markup Languages –** SGML, HTML and XML
  - **Specification Languages –** dream of the future

Special-purpose Languages

- Database Query Languages – SQL
- Simulation Languages – Simula, GPSS, SIMSCRIPT
- Silicon Design Languages – VRML, VHDL,SystemC (C++), SpecC(C)
- Graphics Design Languages – GRAF
- Real-time Languages – RT-FORTRAN, BCL, Embedded-C, Embedded Java

Compiler Input:

- Source program
- Configuration parameters or pragmatics (#pragma directives)
- *Source and Target Language Definitions*

Compiler Output:

- Target program
- Error messages
- Information accompanying the target program – external symbol tables, cross-reference tables.

Target Program:

- High-Level Language
- Low-Level Code (Language)

Target Low-Level Code Type:

- Pure Machine Code
- **Augmented Machine Code**
- Virtual Machine Code

Target Low-Level Code Format:

- Assembly or Pseudo-assembly Language Format,
- **Relocatable Binary Format**
- Memory-Image Format (Load & Go)

Run-time Environment:

- Fully Static Environment
- Fully Dynamic Environment
- Mixed Environment – Stacked-based environment

Compiler Related Applications

Editors, Word Processors, Command Interpreters, Formatting Printers, XML Parser, and actually almost all applications – big and small.