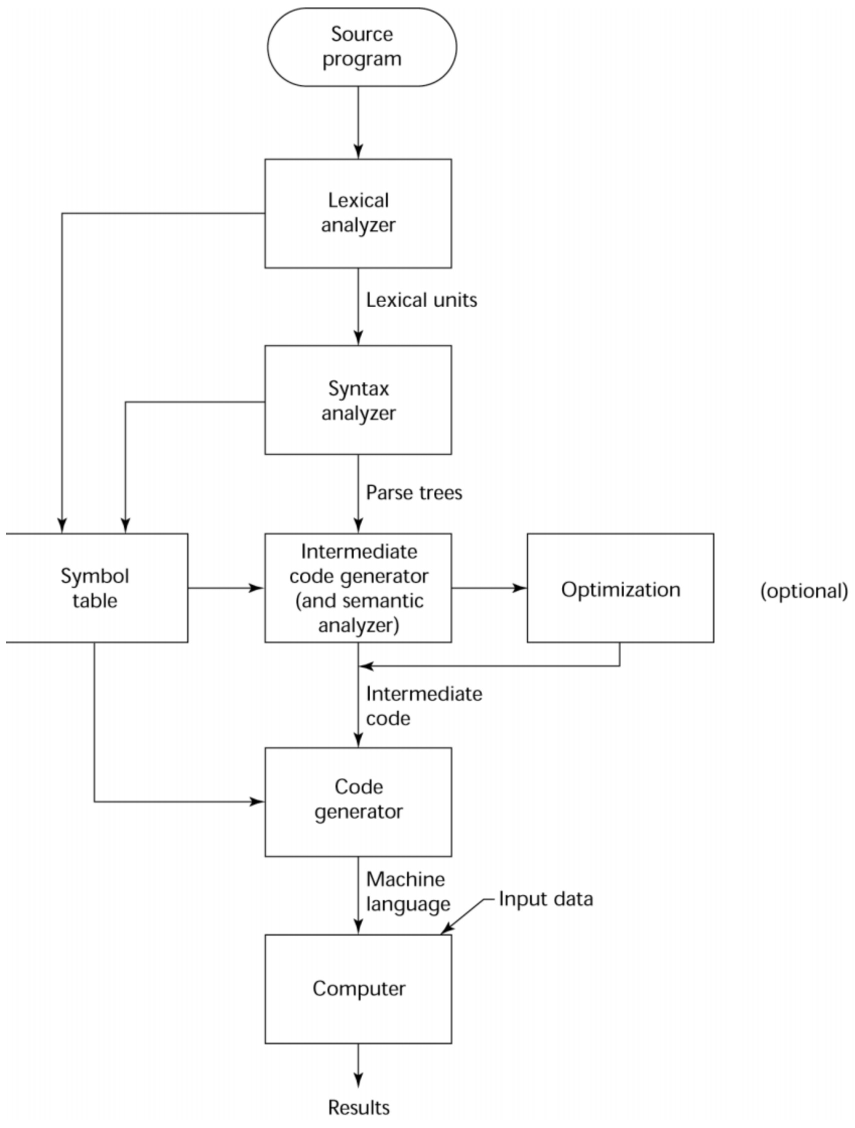
Week 5. Compilers

Compilers

- If you would like to get into the nitty gritty of compiler design, I recommend you take Alan Labouseur's Course on Compiler Design. This lecture only serves as a brief introduction to compilers and what you need to know for this course.
- The Compilation Process



  -
- GCC
  - GNU Compiler Collection

Reverse Engineering

- When it was first released in 1987, GCC 1.0 was named the GNU C Compiler since it only handled the C programming language. It was extended to compile C++ in December of that year.
  - Now Supports:
    - Objective-C
    - Objective-C++
    - Fortran
    - Java
    - Ada
    - Go
    - etc.
- Optimization Levels
  - A typical compiler has about three such levels, where level zero means that optimization is completely disabled.
  - Optimization can also be targeted towards code size or code speed. A non-optimizing compiler is faster and produces more understandable (albeit verbose) code, whereas an optimizing compiler is slower and tries to produce code that runs faster (but is not necessarily more compact).
  - In addition to optimization levels, a compiler can include some debug information in the resulting file, producing code that is easy to debug.
  - One of the important features of the´debug' code is that it might contain links between each line of the source code and its respective machine code address.
  - Optimizing compilers, on the other hand, tend to produce output where entire lines of source code can be optimized away and thus not even be present in the resulting machine code.
  - Why is this important?
    - Reverse engineers can encounter either version, simply because some developers turn on the compiler's optimization flags and others do not. We cannot take one compilers output to be the verbatim output of source -> machine code, it just mirrors the function of the source.
  - GCC Optimization Levels (O)

- Most optimizations are completely disabled at -O0 or if an -O level is not set on the command line, even if individual optimization flags are specified. Similarly, -Og suppresses many optimization passes.
- Link to all optimization flags:
    - https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

option	Optimization level	Execution time	Code size	Memory usage	Compile time
-O0	optimization for compilation time (default)	+	+	-	-
-O1 or -O	optimization for code size and execution time	-	-	+	+
-O2	optimization more for code size and execution time	--		+	++
-O3	optimization more for code size and execution time	---		+	+++
-Os	optimization for code size		--		++
-Ofast	O3 with fast none accurate math calculations	---		+	+++

- When the author of this book first started learning C and, later, C++, he used to write small pieces of code, compile them, and then look at the assembly language output. This made it very easy for him to understand what was going on in the code that he had written. 1. He did this so many times that the relationship between the C/C++ code and what the compiler produced was imprinted deeply in his mind. It's now easy for him to imagine instantly a rough outline of a C code's appearance and function. Perhaps this technique could be helpful for others.
    - We don't have to go to this extent, but it will serve as a learning technique.
- Print out or show and assembly instruction reference sheet for the class to use for the following.
- Navigate to: https://www.godbolt.org
    - Work through with the class exactly what the program is doing with the following examples:
        - The multiplication one pre-loaded on the site
            - Change the optimization levels to see if there is a change, ask for predictions before hand

G Leaden - Week 5

Reverse Engineering

- Hello World!
  - Do a few variations of this, ie:
    - Do not show the class the code side, and assign “Hello World!” To a variable, show them the machine code, ask what changed.
    - Change the code and ask what will or should change in the machine code section.
- Create a function then call it from a higher function
  - Word appending
  - Add 2 to whatever is passed
- Load a caesar cipher program, attempt to decipher it with the class with the compiled code.