Fast, highly sophisticated code analysis and code transformation tools are essential for modern software development. Before releasing its mobile apps, Facebook submits them to a tool called Infer that finds bugs by static analysis, i.e., without even having to run the code, and guides developers in fixing them. Google Chrome and Mozilla Firefox analyze and optimize JavaScript code to make browsers acceptably responsive. Performance-critical systems and application software would be impossible to build and evolve without compilers that derive highly optimized machine code from high-level source code that humans can understand and reason about efficiently. Understanding what modern code analysis and transformation techniques can and can’t do is a prerequisite for research on both software engineering and computer architecture, since hardware relies on software to realize its potential. In this class, you will learn the fundamentals of code analysis and transformation, and you will apply them by extending LLVM, a compiler framework now in production use by Apple, Adobe, Intel and other industrial and academic enterprises.

Course Code: ??

Course Times: ??

Instructor:
Simone Campanoni, simonec@eecs.northwestern.edu

Assignments:
In this class, you will learn how to design code analysis and optimization passes in a production compiler (LLVM). You will have 7 assignments; the first is a raw implementation of a trivial code analysis and related optimization. Subsequent assignments will involve incremental improvement of your original work. The end result of these assignments will comprise your final project that must include:

- a state-of-the-art code analysis and optimization implemented in LLVM
- a paper that describes your work, including experimental results obtained in a real system (e.g., your laptop)
- slides that you will use during your presentation at the end of the quarter where you will describe your work to the rest of the class

Materials:
Selected papers for topics not covered by the mentioned book
Grading Policy:

Homework assignments:     70%  This is cumulative to all homeworks assigned during the class
Final project:             20%  Quality of the end-result of your final project
Final presentation:       10%  Your presentation during the last week of the class

Pre-reqs:
EECS 213: Introduction to Computer Systems (or equivalent)

Closely linked classes:
EECS 322: Compiler Construction
EECS 361: Computer Architecture 1