Advanced compiler construction

Summary
Students learn several implementation techniques for modern functional and object-oriented programming languages. They put some of them into practice by developing key parts of a compiler and run-time system for a simple functional programming language.

Content
Part 1: implementation of high-level concepts
- functional languages: closures, continuations, tail call elimination,
- object-oriented languages: object layout, method dispatch, membership test.

Part 2: optimizations
- compiler intermediate representations (RTL, SSA, CPS),
- inlining and simple optimizations,
- register allocation.

Part 3: run-time support
- interpreters and virtual machines,
- memory management (including garbage collection).

Keywords
compilation, programming languages, functional programming languages, object-oriented programming languages, code optimization, register allocation, garbage collection, virtual machines, interpreters, Scala.

Learning Prerequisites
Recommended courses
CS-320 Computer language processing

Important concepts to start the course
Excellent knowledge of Scala and C programming languages

Learning Outcomes
By the end of the course, the student must be able to:

- Assess / Evaluate the quality of a compiler intermediate representation
- Design compilers and run time systems for object-oriented and functional programming languages
- Implement rewriting-based compiler optimizations
- Implement efficient virtual machines and interpreters
- Implement mark and sweep or copying garbage collectors

Teaching methods
Ex Cathedra, mini-project

Assessment methods
Continuous control (mini-project 80%, final exam 20%)

Supervision
Office hours  No
Assistants  Yes
Forum  Yes

Resources
Virtual desktop infrastructure (VDI)
No

Websites
- https://cs420.epfl.ch/