Advanced Topics in Compilers

Simone Campanoni
simone.campanoni@northwestern.edu

Welcome!

Prerequisites

Source code (e.g., C++)

i	n	t		m	a	i	n		…

Front-end — ***EECS 322: Compiler Construction***

IR

```
myVarX = 40
myVarY = myVarX + 2
```

Middle-end — ***EECS 323: Code analysis and transformation***

IR

```
myVarY = 42
```

Back-end — ***EECS 322: Compiler Construction***

Machine code

```
0101011101010101
```

Goal and mindset of ATC

- The goal is to expose you to compiler research
  - An example of how we identify problems that require novelty
  - How we create novelty
  - How we implement solutions
  - And how we test it

- It requires a lot of independence on your end
  - You don’t know something?
    Check the makefiles, sources, documentation (when it exists)
  - Is something not working?
    Understand why, debug it, fix it, send pull requests

Outline

- Structure of the course

- This year’s topics

ATC

- You learned the internals of modern production-quality compilers
  - E.g., Data-flow analysis, constant propagation, (CS 323)
  - E.g., Instruction selection, register allocation (CS 322)
- Research labs include advanced techniques not yet included in production-quality compilers
  - They are not (yet) as robust as production-quality compilers need to be
- ATC:
  - You will learn some of these advanced techniques
    - They are organized into topics
    - Each year we look at techniques of only some topics
    - Each year ATC is different

ATC on Canvas

2023 Spring

- Home
- Announcements
- Assignments
- Grades
- People
- Files
- Syllabus
- CTEC
- NameCoach
- Worldwide
- Learning Apps
- Discussions
- Pages
- Rubrics
- Outcomes
- Quizzes
- Modules
- Collaborations

ATC 2023

Edit

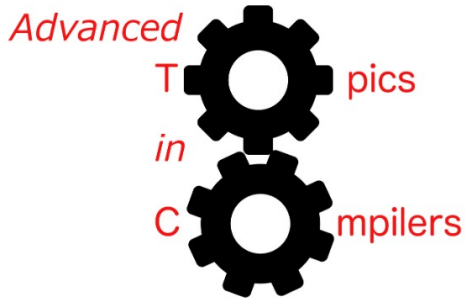
Syllabus.pdf

Lectures and files

Tutorials

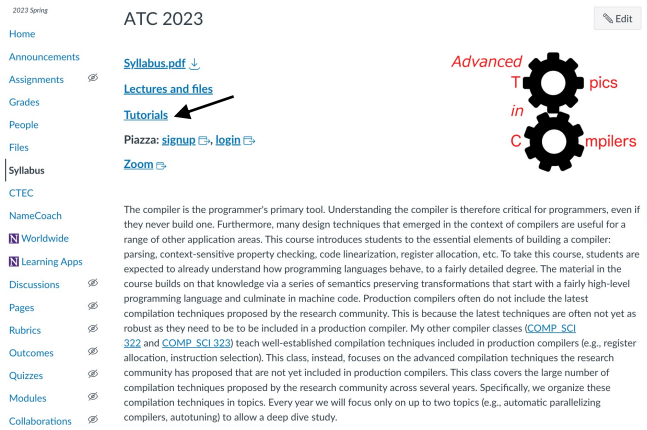
Piazza: signup, login

Zoom



The compiler is the programmer's primary tool. Understanding the compiler is therefore critical for programmers, even if they never build one. Furthermore, many design techniques that emerged in the context of compilers are useful for a range of other application areas. This course introduces students to the essential elements of building a compiler: parsing, context-sensitive property checking, code linearization, register allocation, etc. To take this course, students are expected to already understand how programming languages behave, to a fairly detailed degree. The material in the course builds on that knowledge via a series of semantics preserving transformations that start with a fairly high-level programming language and culminate in machine code. Production compilers often do not include the latest compilation techniques proposed by the research community. This is because the latest techniques are often not yet as robust as they need to be to be included in a production compiler. My other compiler classes (COMP_SCI 322 and COMP_SCI 323) teach well-established compilation techniques included in production compilers (e.g., register allocation, instruction selection). This class, instead, focuses on the advanced compilation techniques the research community has proposed that are not yet included in production compilers. This class covers the large number of compilation techniques proposed by the research community across several years. Specifically, we organize these compilation techniques in topics. Every year we will focus only on up to two topics (e.g., automatic parallelizing compilers, autotuning) to allow a deep dive study.

ATC on Canvas



Tutorials

Next are tutorials that show how to use common developing tools you (as well as every developer and system researcher) should be aware of.
Please consider these tutorials to be examples. Feel free to find on the web more (and perhaps better) ones.
Finally, please consider the links below to be the starting point (so follow the links included in them).

Perf

- Tutorial 0
- Tutorial 1
- Tutorial 2

Valgrind and tools built in it

- Tutorial 0
- Tutorial 1
- Tutorial 2
- Tutorial 3
- Tutorial 4
- Tutorial 5
- Tutorial 6

Gdb

- Tutorial 0
- Tutorial 1
- Tutorial 2
- Tutorial 3

Git

- Tutorial 0
- Tutorial 1
- Book

Makefile

- Tutorial 0

ATC on Canvas

2023 Spring

- Home
- Announcements
- Assignments
- Grades
- People
- Files
- Syllabus
- CTEC
- NameCoach
- Worldwide
- Learning Apps
- Discussions
- Pages
- Rubrics
- Outcomes
- Quizzes
- Modules
- Collaborations

ATC 2023

Edit

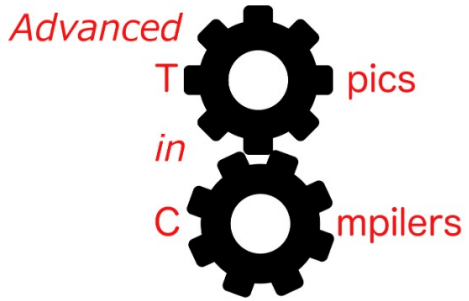
Syllabus.pdf

Lectures and files

Tutorials

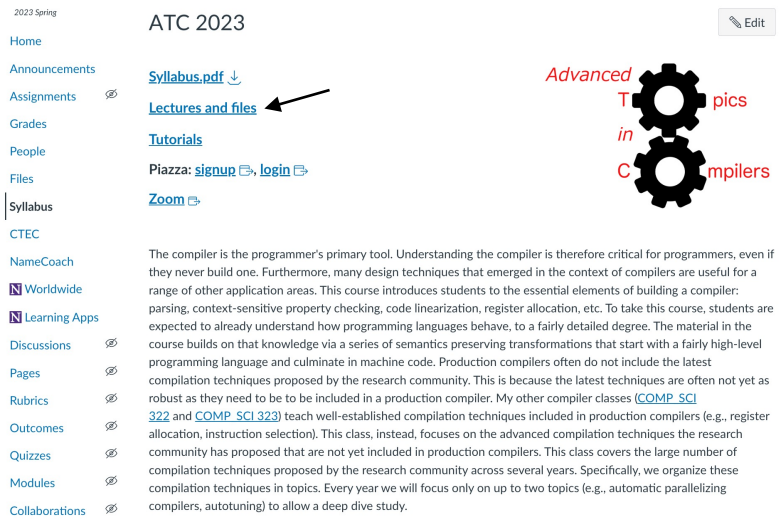
Piazza: signup, login

Zoom



The compiler is the programmer's primary tool. Understanding the compiler is therefore critical for programmers, even if they never build one. Furthermore, many design techniques that emerged in the context of compilers are useful for a range of other application areas. This course introduces students to the essential elements of building a compiler: parsing, context-sensitive property checking, code linearization, register allocation, etc. To take this course, students are expected to already understand how programming languages behave, to a fairly detailed degree. The material in the course builds on that knowledge via a series of semantics preserving transformations that start with a fairly high-level programming language and culminate in machine code. Production compilers often do not include the latest compilation techniques proposed by the research community. This is because the latest techniques are often not yet as robust as they need to be to be included in a production compiler. My other compiler classes (COMP_SCI 322 and COMP_SCI 323) teach well-established compilation techniques included in production compilers (e.g., register allocation, instruction selection). This class, instead, focuses on the advanced compilation techniques the research community has proposed that are not yet included in production compilers. This class covers the large number of compilation techniques proposed by the research community across several years. Specifically, we organize these compilation techniques in topics. Every year we will focus only on up to two topics (e.g., automatic parallelizing compilers, autotuning) to allow a deep dive study.

ATC on Canvas



Lectures

Next are the lectures of this class with the link to the related videos.

Week 0:

- Welcome, structure of the class, and projects (slides)

Week 1:

- Topics 0 and 1
- Topics 1 and 2

Week 2:

- Topic 3
- NOELLE (slides)

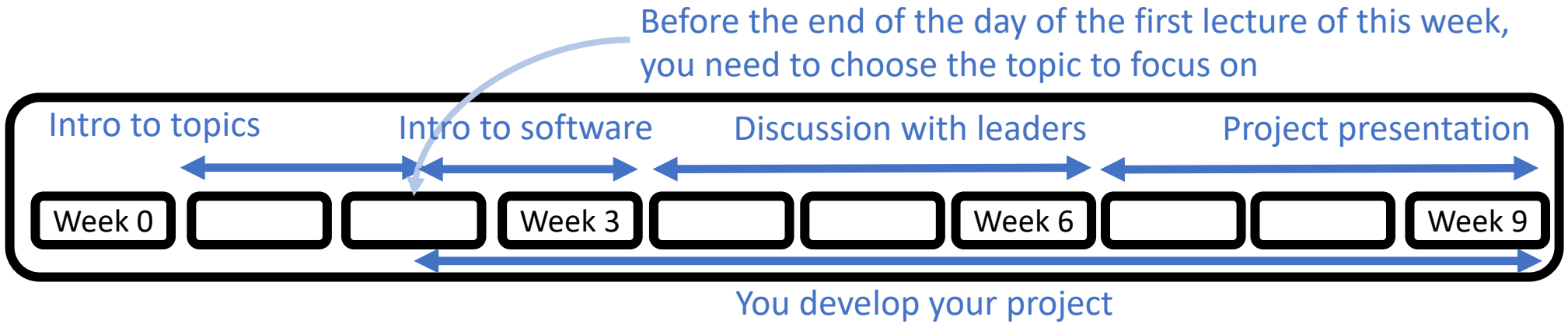
Week 3:

- MemOIR (slides)
- More about NOELLE and MemOIR

Week 4:

- Leading discussions

The ATC structure



- You need to read 1 paper per topic before the class that will introduce that topic
- After choosing a topic, you need to read papers about it, and do a project about it within our codebases (and optionally others)
- We'll only introduce the codebases in class
- Watch all videos about the codebases to use for your project on the ATC's webpage: http://users.cs.northwestern.edu/~simonec/ATC.html

Papers to read

- Everyone is required to read all papers that introduce a topic
  (those discussed in the second and third week)
- Only the discussion leader of a topic must read the remaining papers of that topic
- The discussion leader will lead the discussion about his/her topic
  - You will be the teacher for that topic and you will teach us
  - No slides
  - Use white/blackboard to present the concepts

- Expectation: everyone will learn all concepts explained by the leader

Software

- NOELLE: it can be downloaded from here
  A set of abstractions/transformations/analyses that can be used by an LLVM middle-end pass
- Gino: it can be downloaded from here
- MemOIR: it can be downloaded from here
- NOELLEGym: it can be downloaded from here
  infrastructure to test NOELLE-based optimizations on benchmarks typically used for testing research ideas
- VIRGIL: it can be downloaded from here
  our task engine that is used by (for example) the Gino parallelizing compiler
- Not ours: from the web

Outline

- Structure of the course

- This year’s topics

Topics

- Every year ATC covers different topics

- Topics we will cover this year:
  1. Equality Saturation
  2. Collection selection
  3. Sequential IR
  4. Parallel IR
  5. Sparse computation
  6. Scheduling languages
  7. LLM in compilation
  8. Vectorization

Always have faith in your ability

Success will come your way eventually

Best of luck!