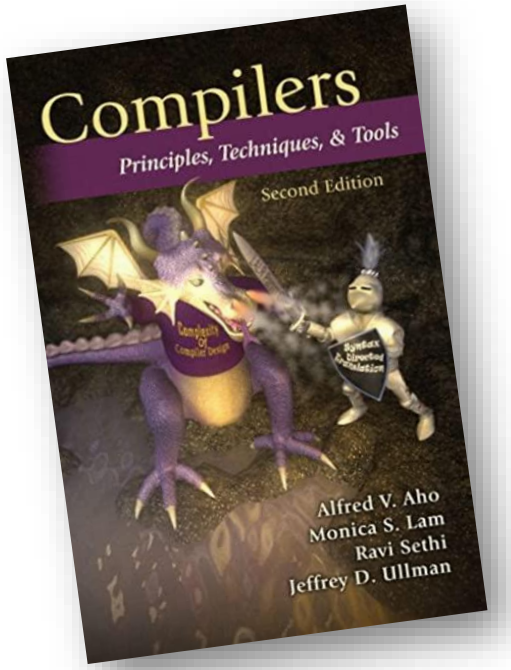
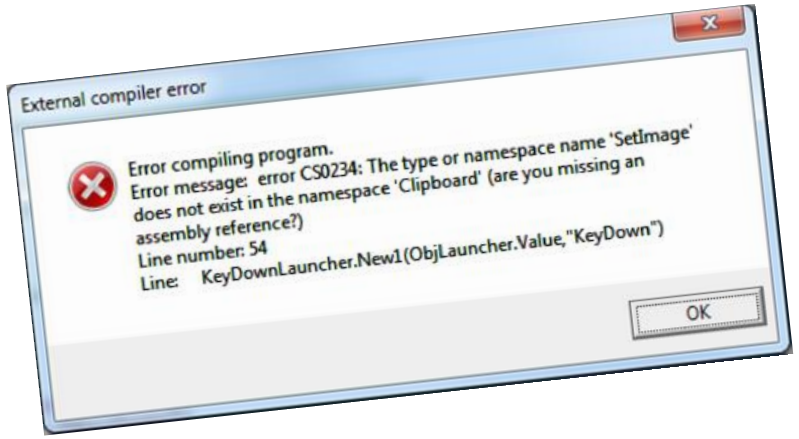
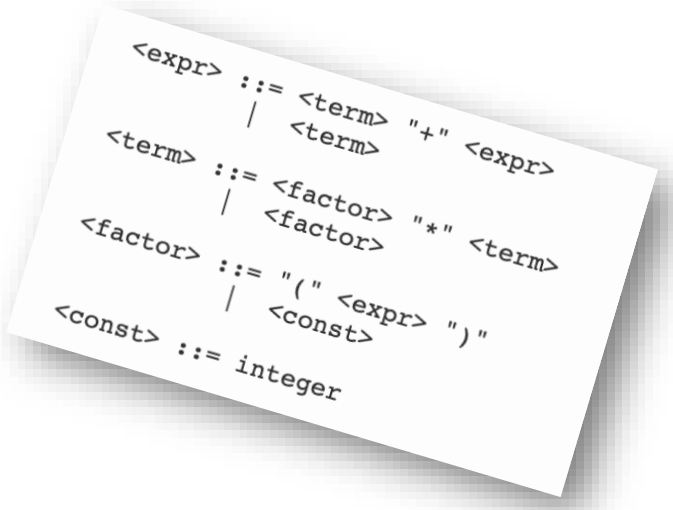
CSE110A: Fundamentals of Compiler Design



About This Course





Hello!

- Course Instructor:
  Marcelo Siero (he/him)
  Faculty at UCSC since Fall 2020
  You can call me Marcelo
  http://ideassiero.com

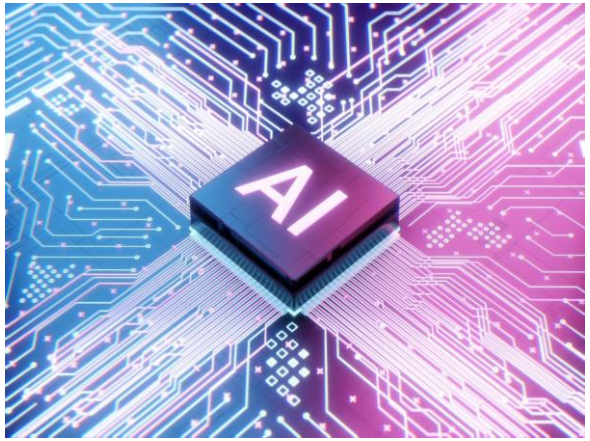
- Primary Course Author:
  Author/Professor Tyler Sorensen (he/him)
- **Faculty** at UC Santa Cruz Since Summer 2020
- Previously
  - Post doc at Princeton
  - PhD Student at Imperial College London
  - BS/MS at University of Utah

https://users.soe.ucsc.edu/~tsorensen/

Japanese Adage

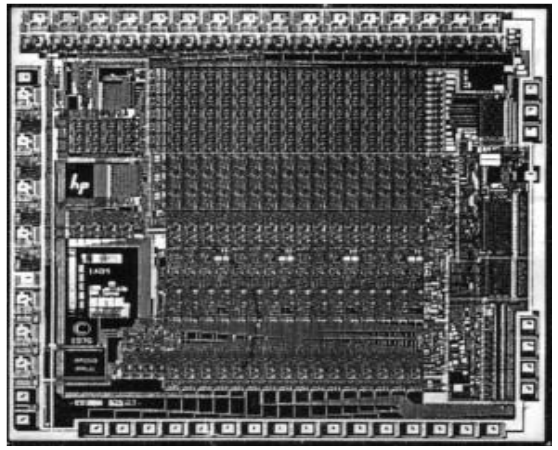
Tou Wa,
Ichigi No Haji
Towanu Wa,
Matsudai No Haji

Research Interests



AI Acceleration

Parallel Computation



VSLI Design / CAD

AI and Voice Technology

Compiler Technology

Computer Consulting

Today’s class

- Class syllabus (I apologize in advance for the text slides)

- High-level discussion on compilers

Description

In this class you will learn about compiler design and implementation. In the abstract, compilers explore many of the foundational problems in computer science. In practice, compilers are massive pieces of well-oiled software, and are some of the engineering marvels of the modern world.

Description

- We will explore how compiler techniques
  - transform high level languages into low-level languages, i.e. down to the instructions that processors can actually execute.
  - automatically make code more efficient and safe to execute.

- When you leave this class you should be comfortable with:
  - specifying programming language grammars,
  - how to efficiently parse these languages,
  - and how to convert complex high-level code into equivalent (and hopefully more performant) low-level code.

Course resources

- Public course website: https://siero.github.io/CSE110A-W26/index.html
  - Schedule, slides, syllabus, additional resources
- Private course website: Canvas
  - grades, announcements, SETs, homeworks, tests, zoom links (if needed)
- Docker Image
  - Used for homework
- Piazza
  - Used for questions, discussions, etc.

Teaching Staff Introductions

- Grad TA: Zheyuan Chen <zchen406@ucsc.edu>
  - PhD students who have compilers as one of the main components of their research!

- Undergrad mentors/graders:
  - TBD

They are all awesome! Please get to know them!

Desired Background

- CSE 12 (assembly)
  - We need to understand low level code (e.g. assembly)
- CSE 101 (data-structures and algorithms)
  - High-level code is represented as tree/graph data-structures.
  - Algorithms on these structures is how we will transform the code into a low-level
- *Optional* (CSE 103): programming languages are specified using regular expressions and context-free grammars
- *Optional* (CSE 120): understanding how low-level code executes on the processor can help us automatically apply optimizations

Background

- Officially supported homework environment:
  - Docker
  - Command line text editor, e.g. vim or emacs
  - many students like VSCode, but we will not be providing support

- *You should be comfortable using the command line (CLI)*

Background

- Languages used in this class:
  - Python - high-level language
  - C - low-level language

- We will provide support for Python (class examples, references, etc.)
  - It is a “friendly” language and you should be able to pick it up quickly
  - We will not use too many advanced features

- You should have learned basic Assembly Programming in CSE 12

Background

- Feel free to share your favorite docker or language resources!

Class Format

- **7:10 – 8:45PM T/Th: 90 minutes**
  - Stevenson Acad 175
  - I will try to stay ~10 minutes afterwards to answer questions
  - *Please be respectful of time*

- I will record class lectures
  - This is meant to be used as a study supplement, not as a replacement for attendance.
  - Recordings are not an equitable replacement (discussions, unable to see white board, etc.)
  - Recordings are not guaranteed (equipment failures, etc.)

- Note that the lectures are being recorded. Please keep in mind, since editing the recording is quite time consuming.

Class Format

- This is a smaller class: please ask questions and engage!

- Do not come to class sick! Be mindful of others. The recordings can work as a substitute during those times.

Typical Class Format

- **First** part of class will be announcements, upcoming homeworks, tests, etc. Announcements will also be sent out by email from Canvas.

- **Second** part of class will be overviewing the quiz from the previous class period.

- **Third** part will typically be a review of the material from the previous class

- **Fourth** part is new material

Asynchronous Discussion

- **Piazza**
  - Private message (to teaching staff) technical homework questions
  - Programming and framework questions (global)
  - Tech news (global)
  - Discussions on class material (global)

- Please do not email directly!
  - Email easily gets buried

- Do not expect replies off-hours (after 5 pm, weekends, holidays)

We will try to answer in 24 hours
Please try to help your peers!

Asynchronous Discussion

- **Additional forums**
  - You are welcome to create one yourselves (e.g. discord)
  - Please make it open and available to all your classmates
  - Please provide sufficient moderation (e.g. be nice to each other!)
  - Please do not cheat
  - Please remember that anything that is not in Canvas may not be private

  - If there are issues, please let me or a TA know!

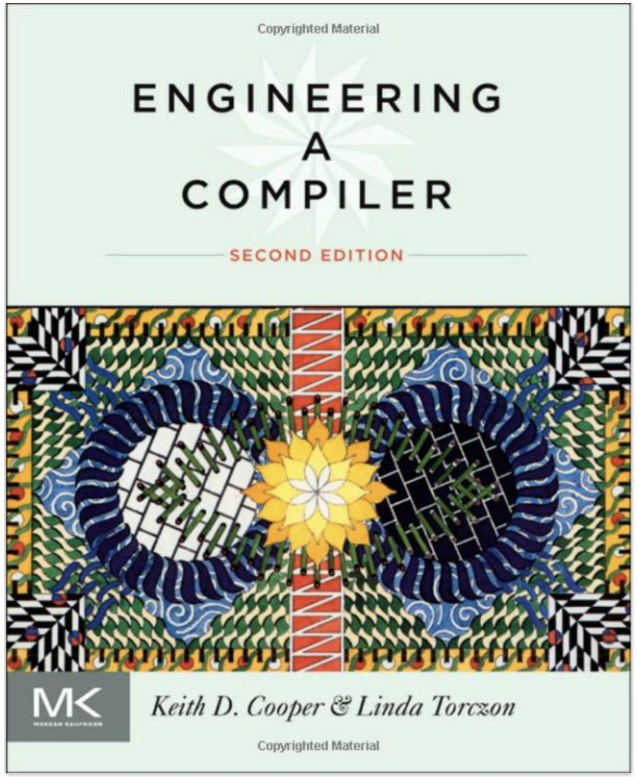
Class Content

- **19** classes, split into
- **4** modules, so there are
- **~5** classes per module

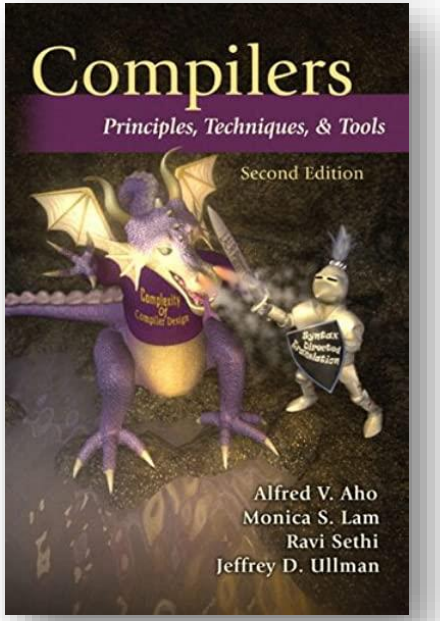
- **Reference book:**

Available online from the library

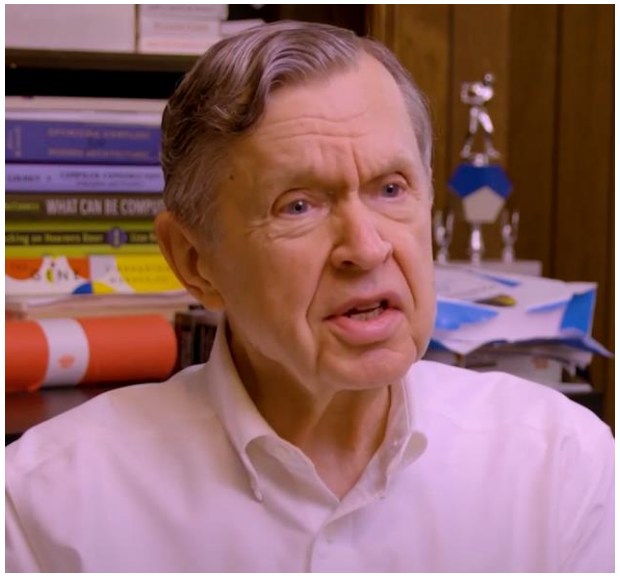
Link on the public webpage

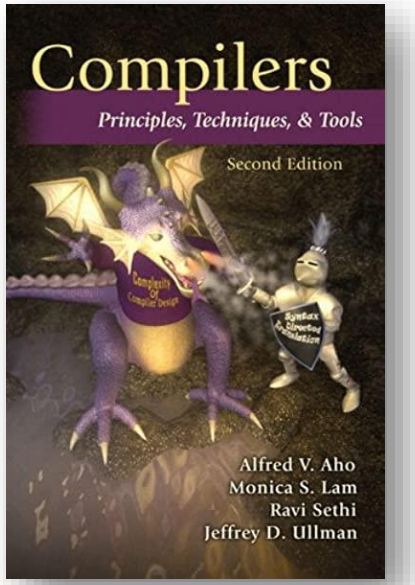


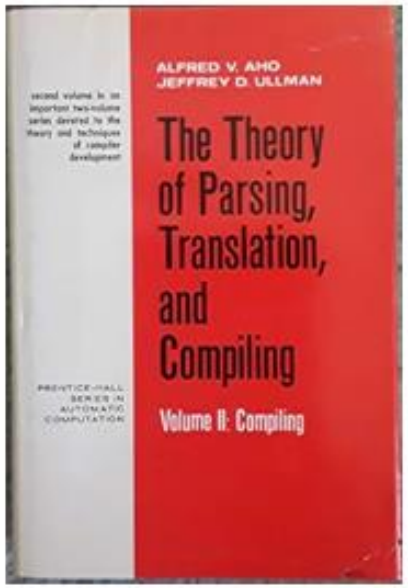
optional extra book



About the Math Behind Languages and Parsing







Alfred V. Aho

Jeffrey D. Ullman

Dragon Book

The Theory of Parsing and Compilers Vol. I & II (not covered in this class)

ACM named Alfred Vaino Aho and Jeffrey David Ullman recipients of the 2020 ACM AM Turing Award for fundamental algorithms and theory underlying programming

https://www.youtube.com/watch?v=ir9Mwl9zhIM&t=2789s

http://infolab.stanford.edu/~ullman/

Class Content

- **Module 1: Introduction, Regular Expressions and Lexing**

  This module will introduce the class, present regular languages and how to express them using regular expressions. We will discuss how to implement a lexer using regular expressions, and how to use the lexer to tokenize a string of input.

Class Content

- **Module 2: Context-free Grammars and Parsing**

  This module will present context-free grammars and how to express them using BNF notation. We will discuss several parser implementations that can be used for different grammars.

Class Content

- **Module 3: Intermediate Representations**

  In this module we will discuss how parsers can produce parse trees, and how parse trees can be operated on to convert complex expressions in a high-level language to a simpler intermediate representation (IR).

Class Content

- **Module 4: Optimizations**

  In this module we will discuss several simple optimizations that exploit the IR structure to make code more efficient. We will discuss loop unrolling, and other optimizations, as well as bonus material as time permits.

Class Content

- **Schedule:**

https://siero.github.io/CSE110A-W26/schedule.html

Readings are highly *suggested* and will be a useful reference for test studying and homeworks

Slides will be uploaded before the lecture

Assignments and Tests

Assignments and Tests

- **Assignments**:
  - 1 or 2 assignment per module
  - All homeworks will be worth 50% combined

- Homeworks build on each other and we will experiment with different granularities
  - By the end, you will have a little compiler that you have written by yourself!

- Do not expect help off-hours (after 5 pm, weekends, holidays)

- We will try to make homeworks due at midnight. If this is an issue, we will move earlier

Assignments and Tests

- **Format**:
  - Coding assignments in Python
  - We will provide a docker image that you should be able to run locally, but we won't use too many external libraries
  - *It must run on the docker to be graded*
  - The homework will specify constraints on the code format and submission format. It must adhere to this format to be graded!
  - We plan to use github classroom for some automatic feedback
    - It will tell you if the format is correct
    - It may run a small number of tests
    - It probably doesn't run all of the tests
  - Github classroom is a new system for this class and there may be some friction. Please be patient with us!

Assignments and Tests

I expect submitted assignments to contain your own original work. You can refer to notes, slides, internet, etc. But do not blindly copy code.

Consulting the internet is a tricky component to constrain. Especially with learning a new language.

- Okay example: “How do you concatenate an array in Python?”
- Not okay example: “How do you implement a compiler in Python?”

Any part of your submission that is not your original work (e.g. code snippets from the internet) need a citation. My aim is to be lenient with cited code, but we may remove some points based on the extent. A few missing points is better than a referral for academic misconduct.

Please do not closely collaborate on homework with classmates. In the case that you do, please mention in the submission. Again, a few missing points is better than a misconduct referral.

Assignments and Tests

This class has a zero tolerance policy on cheating. Please don’t do it. I would much rather get a hundred emails asking for help than have to refer anyone for academic misconduct.

Cheating harms you: this is the best chance in your career to take the time to really learn the class material. If you do not learn the material you will not be successful in a tech career.

Late policy

- Assignments:
  - Will not be accepted late

- Why? Because the assignments in this class build on each other.
  - The next assignment depends on the one before it.
  - We will release reference solutions to previous assignments so that people don’t get stuck
  - Upload check points, plan on getting work done early

- Tests:
  - Only accommodations will be through DRC

Reviewing Grades

- For assignments and tests:
  - You have 1 week from when the grade is posted to discuss grades with teaching staff

Quizzes and Lecture

- Small canvas “quiz” every lecture - take the quiz to get the points
- Quiz answers are graded on engagement, not correctness!
  - All multiple choice questions are free as long as you answer
  - Not always 1 right answer
  - Last question is usually a reflection question. Answer in a few questions
  - Meant to let you reflect on the material
- Quizzes are released right after class and due before the next class.
- *Please only take the quiz if you attended (or watched) the lecture.*
- You can have 3 free missed quizzes

Assignments and Tests

Grade Breakdown:

- homeworks: 50%
- 1 midterm: 10%
- 1 final: 30%
- quiz: 10%

Letter Grade	Percentage	GPA
A+	97-100%	4.33
A	93-96%	4.00
A-	90-92%	3.67
B+	87-89%	3.33
B	83-86%	3.00
B-	80-82%	2.67
C+	77-79%	2.33
C	70-76%	2.00
D+	67-69%	1.67
D	63-66%	0.67
D-	60-62%	0.00
F	0-59%	

From: https://en.wikipedia.org/wiki/Academic_grading_in_the_United_States

Accessibility

UC Santa Cruz is committed to creating an academic environment that supports its diverse student body. If you are a student with a disability who requires accommodations to achieve equal access in this course, please submit your Accommodation Authorization Letter from the Disability Resource Center (DRC) to me by email, preferably within the first two weeks of the quarter. I would also like us to discuss ways we can ensure your full participation in the course. I encourage all students who may benefit from learning more about DRC services to contact DRC by phone at 831-459-2089 or by email at drc@ucsc.edu.

Website tour

Final notes

- This class is constantly evolving
  - Material is still being adapted.
  - There may be issues on HWs and tests (please let us know if you find any!)
  - There may be schedule changes

We will do our best and make sure to stay organized and communicate clearly!

Topics

- About this course

- **High-level discussion on compilers**