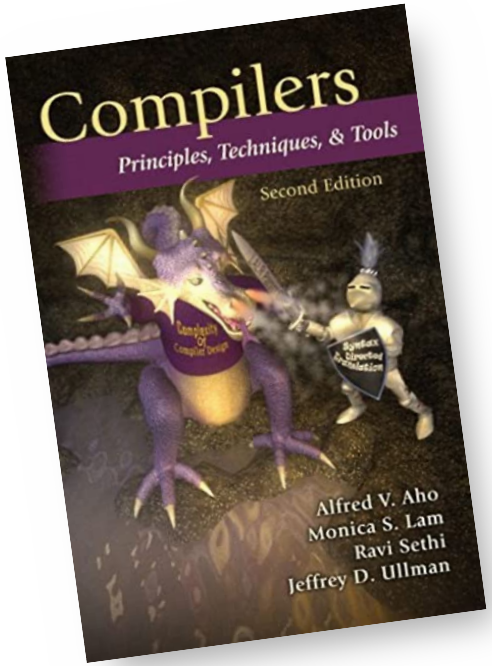
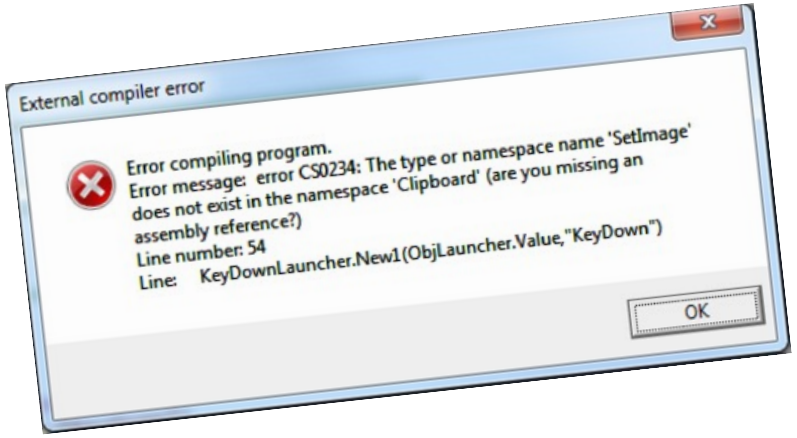
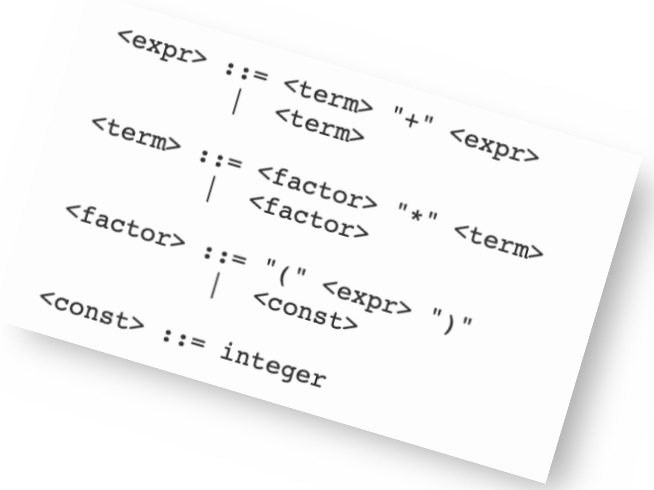
CSE110A: Fundamentals of Compiler Design

March 28, 2022







Hello!

- Professor Tyler Sorensen (he/him)
  - You can call me Tyler
- **Faculty** at UC Santa Cruz Since Summer 2020
  - *First time teaching this class*
- Previously
  - Post doc at Princeton
  - PhD Student at Imperial College London
  - BS/MS at University of Utah

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

https://traveleering.com/
https://wikipedia.com
https://Princeton.edu

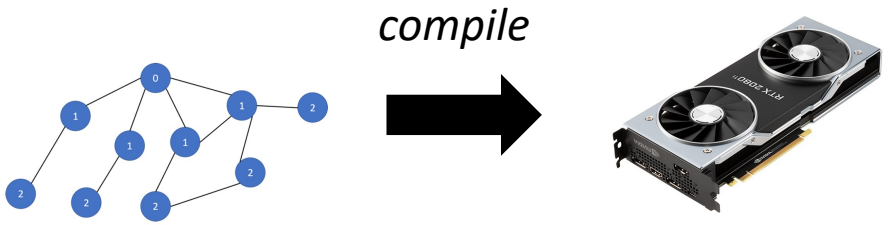
Research Interests

MS: Utah

Bugs in GPU compilers and programming languages

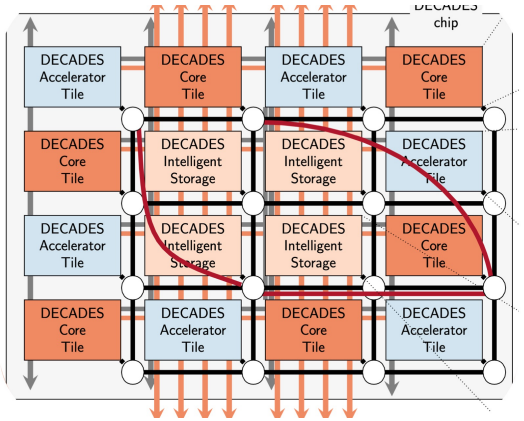
PhD: London

GPU synchronizations, including a DSL for graph analytics on GPUs



Post Doc: Princeton

Compilers targeting new architectures



Research Interests

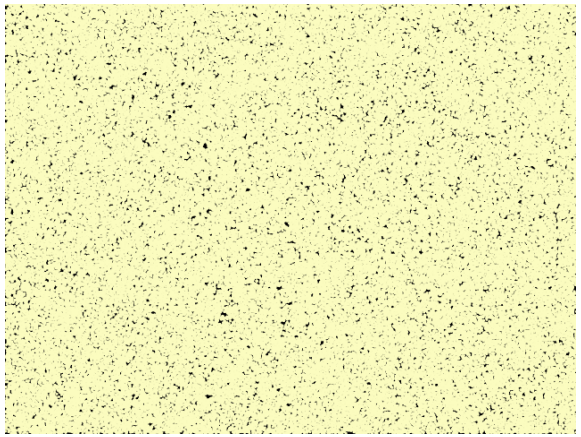
Faculty at UCSC

individual Contributor to

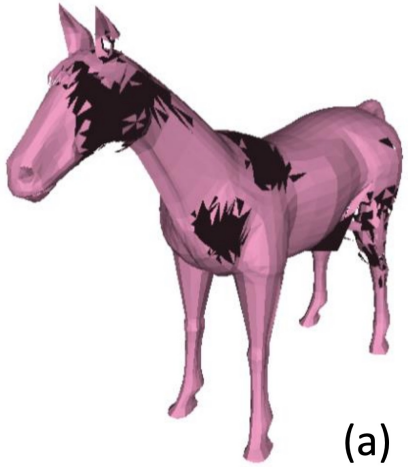
KHRONOS GROUP

Vulkan

parallel particle simulations



GPU memory model testing



(a)

Research Interests

- Compilers!
  - Especially targeting new architectures (like GPUs)
  - Especially targeting difficult application domains (like graphs and simulations)

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. closer to the instructions that processors can actually execute.
  - We will study how compilers can 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://sorensenucsc.github.io/CSE110A-sp2022/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 (instructions incoming)
- Piazza
  - Used for questions, discussions, etc.

Teaching Staff Introductions

- Grad TA: Yanwen Xu
  - PhD student working on particle simulations with me
  - Has worked on compilers in the past

- Undergrad mentors/graders (have taken this class in the past):
  - Arrian Chi
  - Neal Chokshi

They are all awesome! Please get to know them!

A note on COVID

COVID Note : The last few years have been difficult due to the COVID pandemic. Public health concerns and policies remain volatile. The first priority in this class in your health and well-being. We will approach any challenges that arise with compassion and understanding. I expect that you will do the same, both to the teaching staff and to your classmates. We will follow university guidelines and work together to have a productive and fun quarter.

That said, this class is scheduled to be *in person*. This is not an asynchronous class, and it is known that learning outcomes will suffer if you do not attend the synchronous lectures. I may post lectures online for extra study materials, but you should not expect them to be an equitable substitute for in-person attendence (e.g. they do not capture discussions well, and occasionaly the equiptment has errors). If your situation requires asynchronous courses, I suggest you contact an undergraduate adviser to discuss alternative options.

Background

- CSE 12 (systems)
  - 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
  - Ubuntu command line
  - command line text editor, e.g. vim or emacs
  - many students like VSCode. It seems like it can be set up to work with docker but it is not something we will officially support

- You should be comfortable using the command line

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 C in CSE 12

Class Format

- **4:00 - 5:05 MWF: 65 minutes**
  - Porter Acad 144
  - I will try to stay ~10 minutes afterwards to answer questions

- 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.)

- Keep in mind that this means you are being recording and if you ever want me to edit anything out of the recordings, please let me know!

Class Format

- **4:00 - 5:05 MWF: 65 minutes**
  - Porter Acad 144
  - I will try to stay ~10 minutes afterwards to answer questions

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

- Do not come to class sick! These recordings can work as a substitute during those times.

Class Format

- **4:00 - 5:05 MWF: 65 minutes**
  - Porter Acad 144
  - I will try to stay ~10 minutes afterwards to answer questions

- **First** part of class will be announcements, upcoming homeworks, tests, etc.

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

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

- **Fourth** part is new material

Office Hours

- **My office hours:**
  - 3 - 5 PM on Thursdays
  - I will share a google sign up sheet (it will contain a zoom link)
  - Slots are 10 minutes
  - link will be posted in Canvas around noon that day
    - don’t sign up before the Canvas announcement
    - don’t sign up unless you have a question
    - sign up for 1 slot at a time
  - Strict with timing to make sure it is fair
  - No office hours this week

Office Hours

- Yanwen, Arrian, and Neal will decide on theirs in the next few days.
- We hope to get good coverage across days and across in-person and virtual.
- I will update the website when this is decided:
  - https://sorensenucsc.github.io/CSE110A-sp2022/overview.html#teaching-staff

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!

Masks?

- Announcements that I have seen have that masking will no longer be required after April 10.
  - But it will be strongly recommended.
  - Could change given new variants, etc.

- Many people have strong feelings about this.
  - We will treat each other with respect
  - We will find a solution that hopefully everyone can be comfortable with

Masks?

- Personally, I will continue wearing mine for awhile

- Masks make a few things more difficult:
  - Harder to hear questions
  - Harder to recognize faces/names
  - Slightly exhausting to lecture in

- Just like we've done for the last year, we will accommodate each other and have a great quarter!

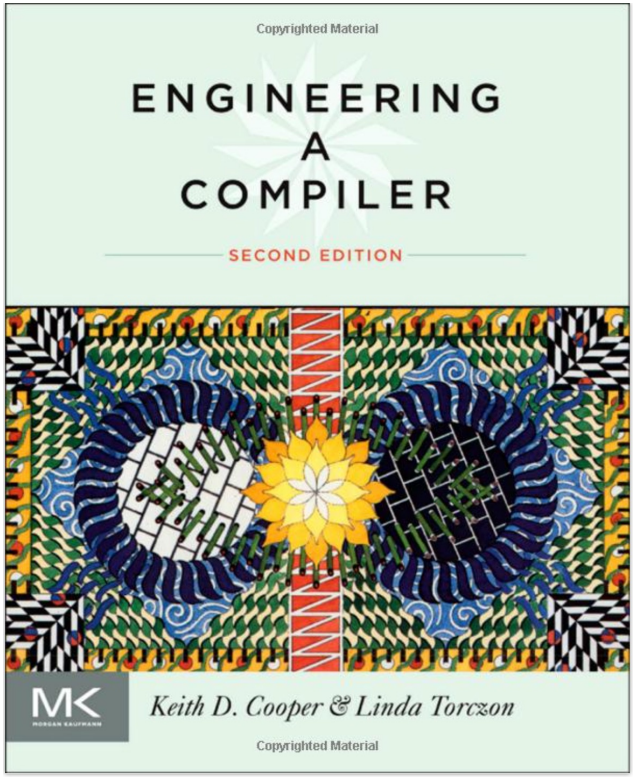
Class Content

- **30** classes, split into
- **5** modules, so there are
- **6** classes per module

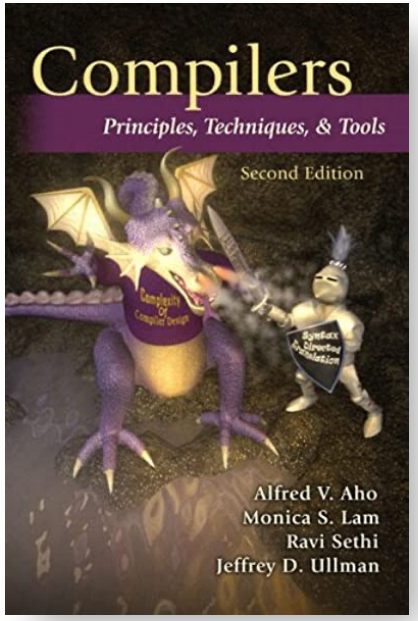
- **Reference book:**

Available online from the library

Link on the webpage



optional extra book



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 local value numbering and common subexpression elimination.

Class Content

- **Module 5: Backend Compilation**

  In this module we will discuss the final step of compilation: turning the IR into an ISA that a processor can execute. We will discuss register mapping algorithms, and subtle issues in IR to ISA mappings.

Class Content

- **Schedule:**

https://sorensenucsc.github.io/CSE110A-sp2022/schedule.html

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

Slides and Readings will be uploaded before the lecture

Assignments and Tests

Assignments and Tests

- **Assignments**:
  - 1 assignment per module
  - halfway through the module
  - due halfway through the next module
  - Each homework is worth 10% of your grade (total of 50%)

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 and a little C
  - We will provide a docker image that you should be able to run locally.
  - *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!

Assignments and Tests

Two tests: Final and Midterm

- So far I've been able to asynchronous tests. Let's plan on that again this year.
- Designed to take ~180 minutes
- As a warning: people take much longer on take-home tests than in-person tests!

Assignments and Tests

Midterm

- Assigned halfway through module 3
- One work week
- Assigned May 2, Due May 6
- Designed to take 3 hours
- Worth 10% of grade
- *Review slides and readings*

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

Assignments and Tests

Final

- 1 Day (16 hours)
- Assigned 8 AM on June 7
- Due before 11:59 PM on June 7
- Designed to take 3 hours, we will monitor Piazza
- Scheduled time is 7:30 PM to 10:30 PM
- Worth 30% of grade
- *inclusive: slide material from all year*

- Do not expect help off-hours (before 9 am, after **7 pm**)

Assignments and Tests

What you can use for tests:

You are free to consult notes, books, or the internet. While the test is active, you are not allowed to discuss the test with another person (either in the class or online). For example, you *can* google concepts that are on the test. You *cannot* post a test question to stackoverflow.

Please do not cheat! I like asynchronous tests but if we catch students cheating then I will have to move to synchronous tests!

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 trickey 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.

I prefer that you do not 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:
  - 10% subtracted per day
  - will not be accepted after 3 days
  - Homework 5 will not be accepted after the final
    - University policy, not mine
- Tests:
  - Will not be accepted late

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 not graded! only if you submit it
  - That said, we reserve the right to remove credit if it looks like the quiz wasn’t taken seriously.
- 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:

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

Letter Grade	Percentage	GPA
A+	97–100 %	4.33 or 4.00
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	73–76 %	2.00
C–	70–72 %	1.67
D+	67–69 %	1.33
D	63–66 %	1.00
D–	60–62 %	0.67
F	0–59 %	0.00

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 impacted
  - We have a higher student/staff ratio than normal
  - This might affect office hours, grading time, etc.

- This class is “new” (especially for me)
  - 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

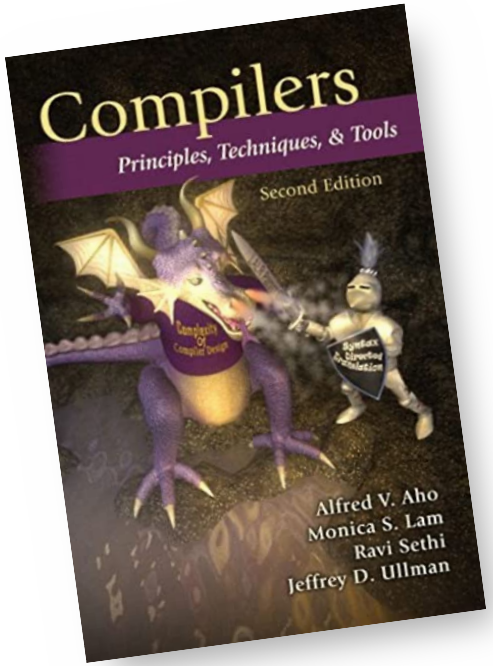
We will do our best and make sure to stay organized and communicate clearly!

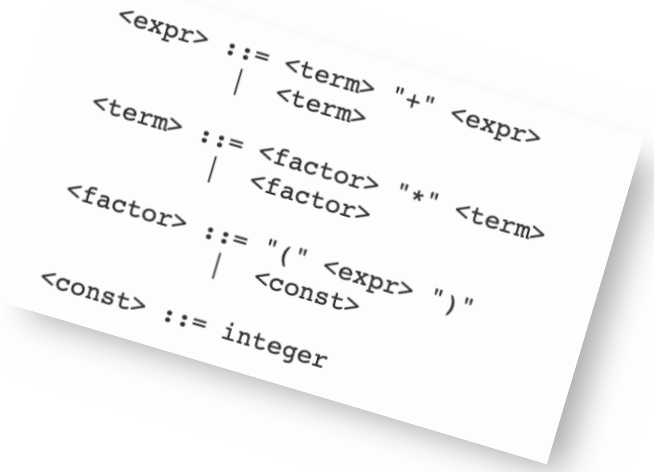
Today’s class

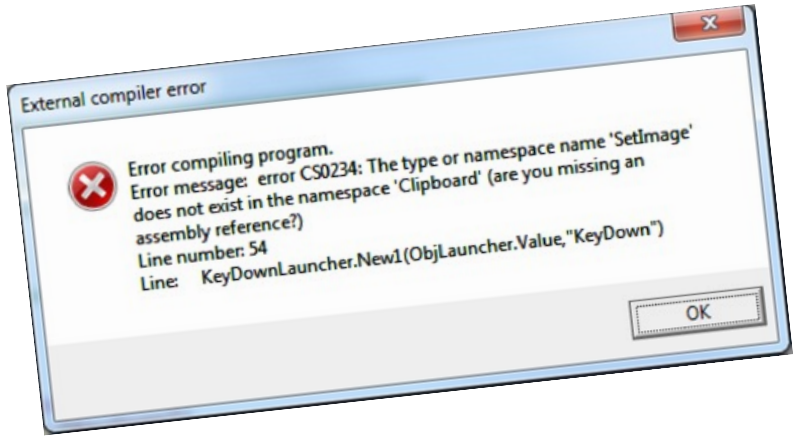
- Class syllabus (I apologize in advance for the text slides)

- **High-level discussion on compilers**

- **Questions**:
  - *What is a compiler?*
  - *What are some of your favorite compilers?*
  - *Have you ever built a compiler?*







What is a compiler?

What are some of your favorite compilers

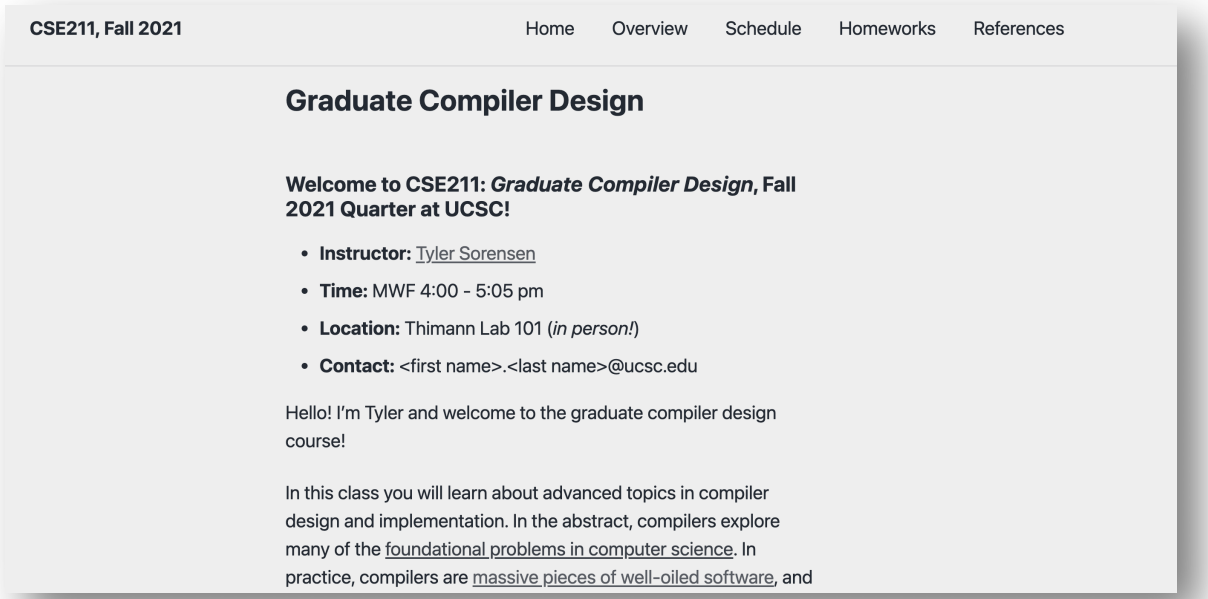
```
---
title: "Graduate Compiler Design"
layout: single
---


### Welcome to **CSE211:** _Graduate Compiler Design_, Fall 2021 Quarter at UCSC!

- **Instructor:** [Tyler Sorensen](https://users.soe.ucsc.edu/~tsorensen/)
- **Time:** MWF 4:00 - 5:05 pm
- **Location:** Thimann Lab 101 (_in person!_)
- **Contact:** \<first name\>.\<last name\>@ucsc.edu


Hello! I'm Tyler  and welcome to the graduate compiler design course!

In this class you will learn about advanced topics in compiler design and implementation. In the abstract, compilers explore many of the [foundational problems in computer science](https://en.wikipedia.org/wiki/Halting_problem). In practice, compilers are [massive pieces of well-oiled software](https://www.phoronix.com/scan.php?page=news_item&px=MTg3OTQ), and are some of the engineering marvels of the modern world. Given the end of Dennard's scaling, compilers will play an increasingly important role to achieve further computational gains. _The main focus of this class is how compilers can make your code more efficient and safe on modern (and near-future) processors_.
```



Building this website started with:

- Markdown to describe the page
- compiled with Jekyll to a static webpage
- static webpage is in HTML and javascript

Canon

Have you ever built a compiler?

What is a compiler?

What is a compiler?

Strings belonging to language L

Strings belonging to language L’

What is a compiler?

Strings belonging to language L

A series of statements in programming language L

Strings belonging to language L’

An executable binary file in an ISA language

What is a compiler?

Strings belonging to language L

A series of statements in programming language L

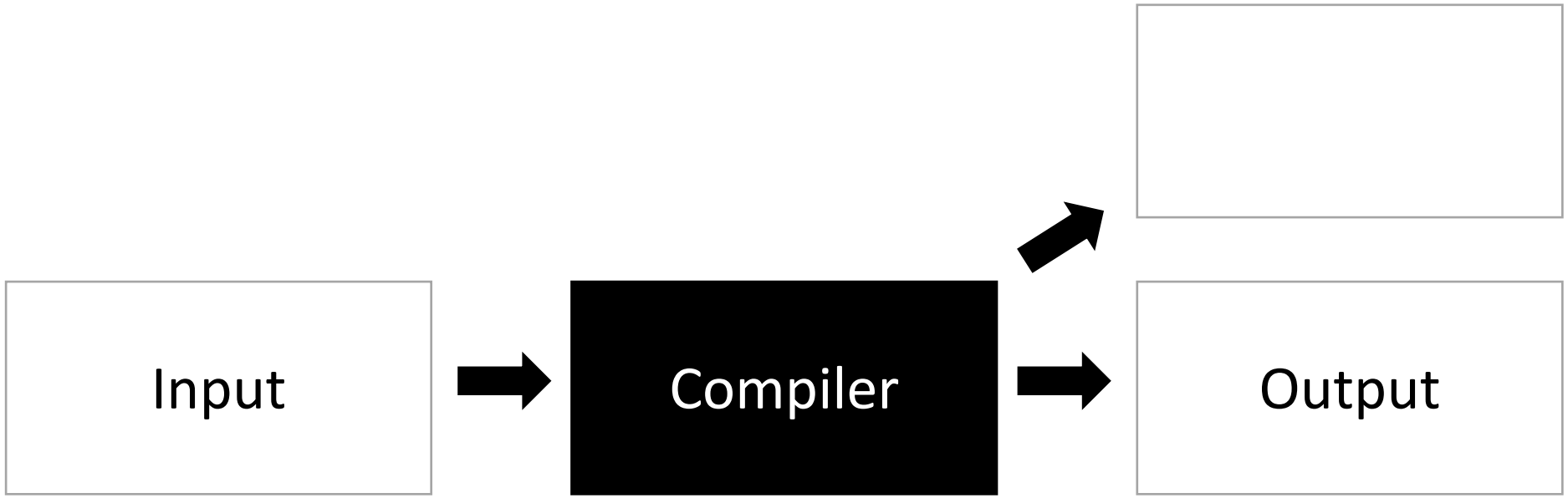
A program written in C++

Strings belonging to language L’

An executable binary file in an ISA language

An x86 Binary executable

What is a compiler?



Strings belonging to language L

A series of statements in programming language L

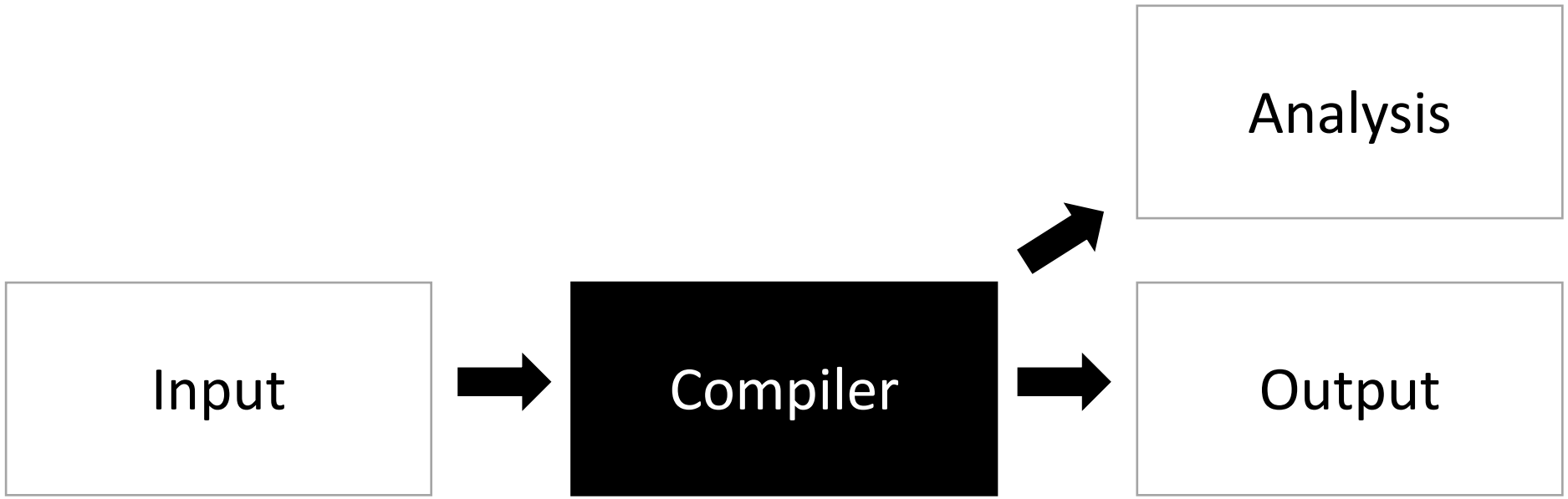
A program written in C++

Strings belonging to language L’

An executable binary file in an ISA language

An x86 Binary executable

What is a compiler?



Strings belonging to language L

A series of statements in programming language L

A program written in C++

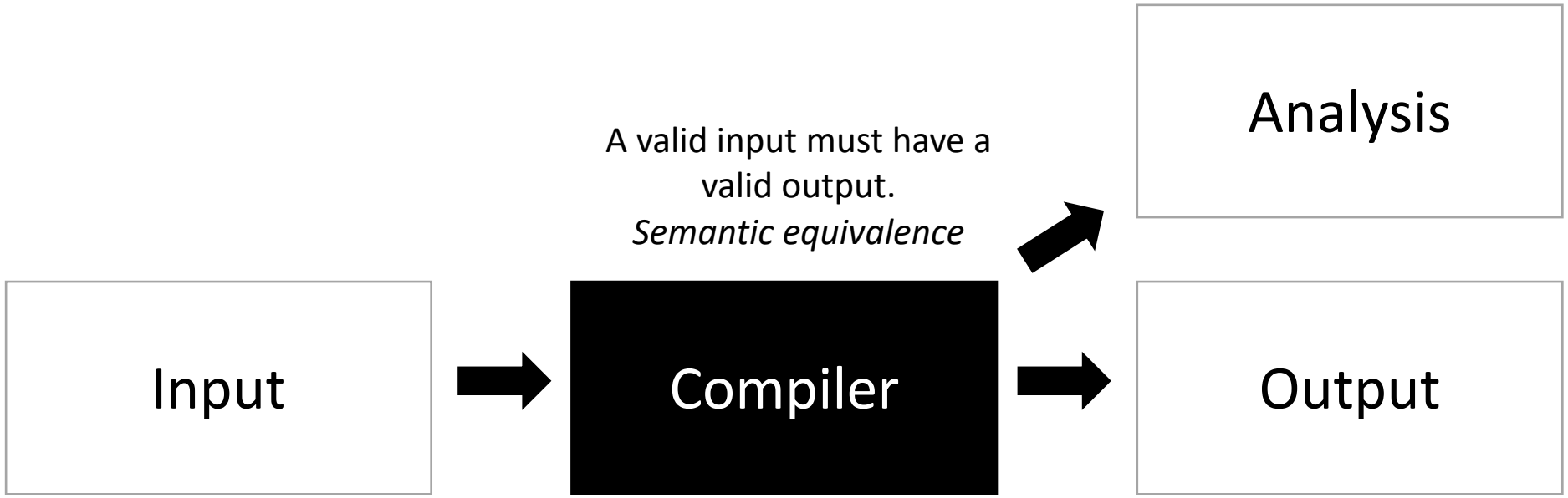
Strings belonging to language L’

An executable binary file in an ISA language

An x86 Binary executable

What is a compiler?

warnings



Strings belonging to language L

A series of statements in programming language L

A program written in C++

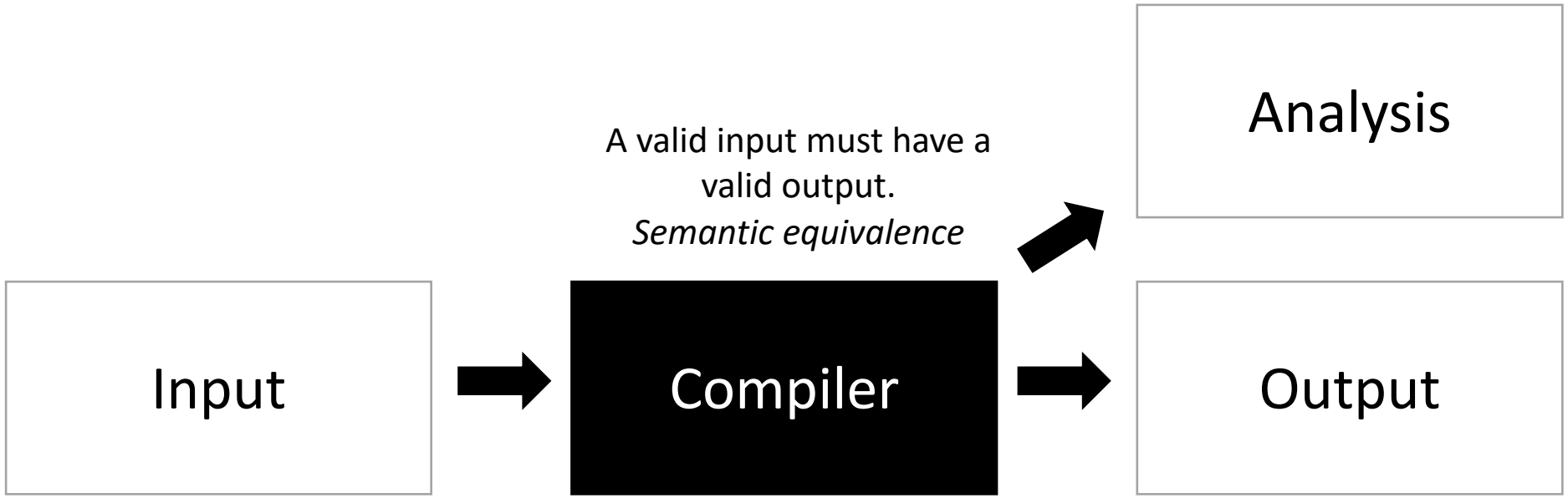
Strings belonging to language L’

An executable binary file in an ISA language

An x86 Binary executable

What is a compiler?

warnings



Strings belonging to language L

A series of statements in programming language L

A program written in C++

What can happen when the Input isn't valid?

Strings belonging to language L'

An executable binary file in an ISA language

An x86 Binary executable

What can happen when the Input isn’t valid?

```
int my_var = 5;
my_var = my_car + 5;
```

Try running this through a compiler; you will get an error and a suggestion!

What can happen when the Input isn’t valid?

```
int foo() {
  int *x = malloc(100*sizeof(int))
  return x[100];
}
```

What about this one? No error...

What can happen when the Input isn’t valid?

```
int my_var = 0;
for (int i = 0; i < 128; i++) {
    my_var++;
}
```

What about this one?

Thank you!

- Thank you for giving it a chance!
- Your experiences and feedback will help shape this class for future students.
- Email is always open for comments about class material, HW assignments, etc.

Next Class

- **Module 1: regular languages, regular expressions and lexing**