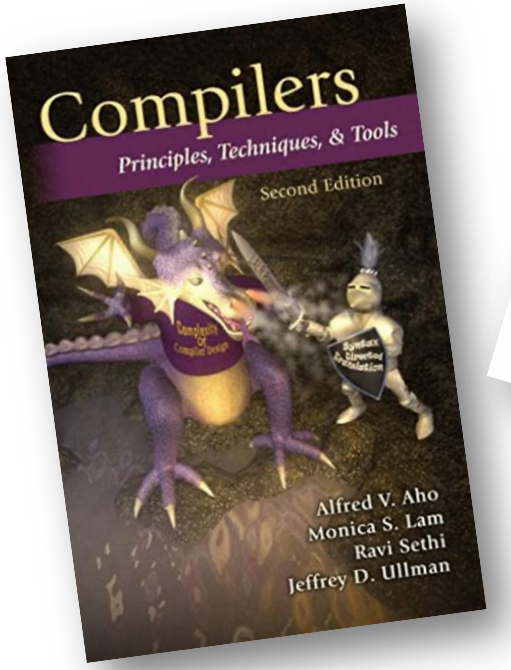
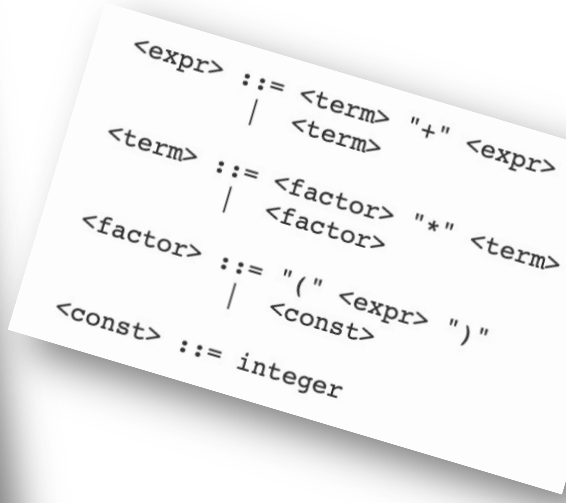
CSE211: Compiler Design

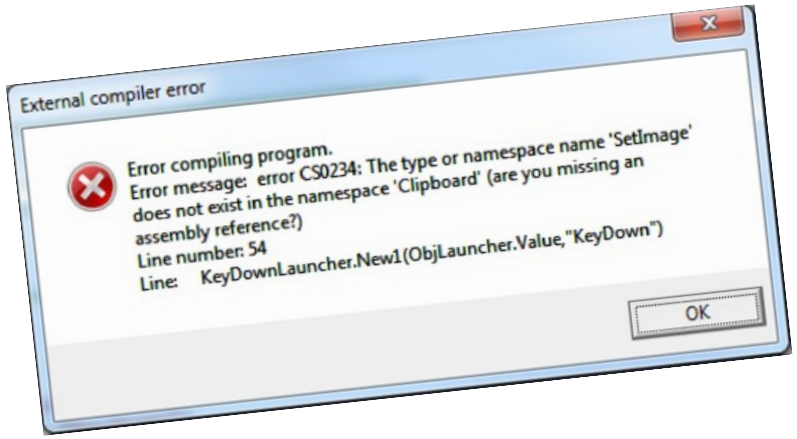
Sept. 24, 2020





- **Topic**: Course Introduction

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



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

Room E2-233

- Tyler Sorensen

Second year faculty at UCSC!

First year teaching in person

- From rural Utah
- Ph.D. at Imperial College London
- Work on parallel programming languages
- Invited member of the Khronos Group

Its been a tough year (and a half)…

Special Note for Fall 2021: The world is still in the midst of a global pandemic. Everyone has had different experiences over the last year and it has been difficult for everyone in different ways. As we return to campus, there will be new difficulties and new challenges. We will work together to solve them and we will have a productive and engaging quarter. That said, this class is designed to be *in person*, please see the COVID policy in the overview and email me if you have any questions or comments.

Please take care of yourself; support each other; find time for the things you enjoy. Please email me if you don't feel like you are performing at the best of your ability and we can discuss various accomodations.

Introductions

- There are 14 of us; let’s get to know each other!
- Discussions will be a big part of class.
- Let me know (e.g. through email) of any accessibility or attendance issues you might have

Today’s class

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

- High-level discussion on compilers
  - First time teaching in 65 minutes

Course resources

- Public course website: https://sorensenucsc.github.io/CSE211-fa2021/index.html
  - Schedule, slides, syllabus, homeworks, additional resources

- Private course website: Canvas
  - grades, discussions, announcements, SETs, homework solutions, tests, zoom links
  - ***Did you all get the email for the announcement?***

- Docker Image
  - Used for homework (instructions incoming)

Description

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. In practice, compilers are massive pieces of well-oiled software, 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*.

Background

- Official prereq is an undergrad UCSC class (CMPS104A)

- But this is a graduate course…

Background

- Understand basics of parsing:
  - Regular expressions
  - Context free grammars
- Understand basics of programming languages
  - how are programs structured (e.g. basic blocks, expressions, statements)
- Comfortable using console coding (e.g. emacs or vim) and a terminal
  - Python
  - C/C++
- Some exposure to architecture, especially machine code (e.g. MIPS or X86)

Background

- First module and assignment will cover lots of compiler basics

Modules

- There are 5 modules
- We will aim to spend roughly two weeks on each
  - We will be flexible. This is a small class and if we need to spend more time on a topic, we will!
- The last module may be cut short
- Structure
  - first 3 modules will be given lecture style
  - last 2 modules are more like a reading group/discussion

Modules

Module 1: Parsing Overview: This module will go over parsing at a high-level. This includes tokenizing, parsing context-free grammars, parser generators, and how to quickly create a parser for a simple programming language.

This is not a class on implementing parsers! Instead I want you to learn how to easily implement a compilers using parser generators!

Modules

Module 2: Flow Analysis: This module will go over different flow analysis (aka static analysis). We will discuss different AST traversals, SSA form, and applications such as identifying use-before-initialized errors.

Modules

Module 3: Automatic Parallelization: This module will go over how programs written for a single thread can be automatically turned into a parallel program using compiler transformations. We will discuss ILP, do-all parallelism, and software pipelining.

Modules

Module 4: Domain Specific Languages: Here we will discuss various domain specific languages and how compilers can leverage the domain constraints to perform even more optimizations. We will look at widely used DSLs for array processing, DNNs, and graph analytics.

Modules

Module 5: Optimization impact and correctness: This module will explore the impact of compiler optimizations and ways to test and validate the compilers are correct.

Class Format

- in-person
- 65 minutes
- *Non-protected materials* are public
- *Protected materials* are on canvas
- *A few classes will be remote due to travel, see the schedule. I will provide Zoom links on Canvas*

Class Format

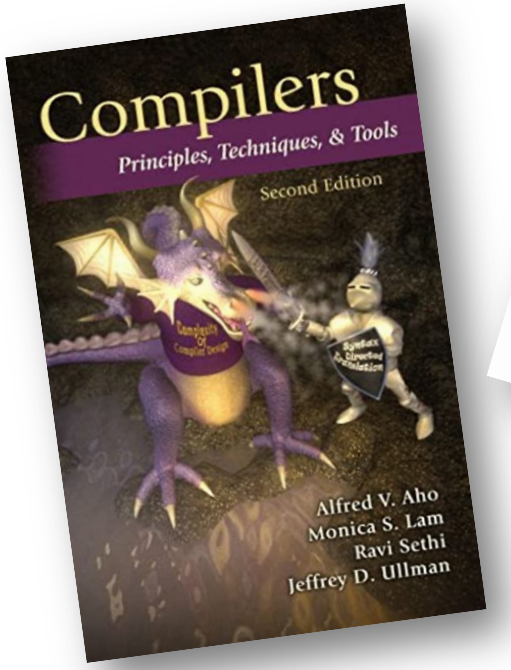
- If the class room is available, I will try to join 15 minutes before class and stay 15 minutes after.

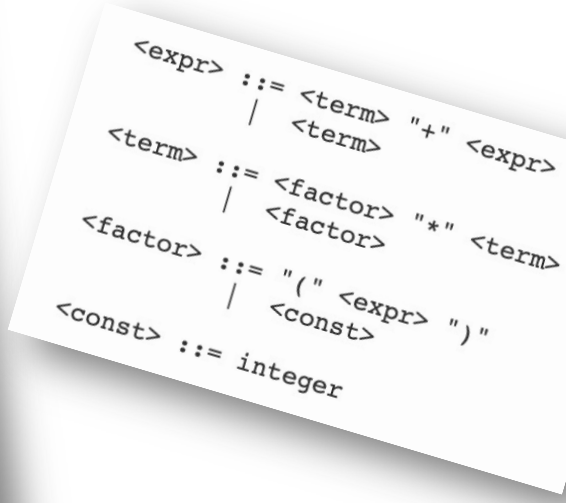
- Join to discuss further, discuss other topics (e.g. grad school), or even just socialize a little

- In-class discussion questions on start of each lecture slide

CSE211: Compiler Design

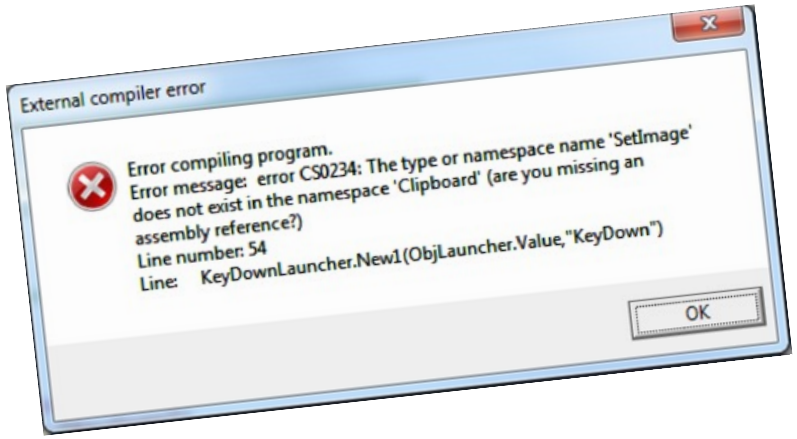
Oct. 1, 2020





- **Topic**: Course Introduction

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



Class Format

- Canvas is the official discussion board.
  - I will check and try to reply within 24 hours to discussion questions
  - Not guaranteed to reply in off-hours or weekends
  - Please start class-wide discussions, link to interesting articles, etc. etc.
  - Please DO NOT publicly share homework solutions before the due date or ask for detailed help in the class-wide channels. Ask privately.
  - I will moderate.

- Do we want something else (Zulip, Discord, Slack, Piazza?)? We can have a vote

Attendance

- Small class means lots of discussion! Please try your best to attend
- Message me before hand if you will miss a lecture
- 25 out of 30 lecture attendances required; more than that will be evaluated on a case-by-case basis.
- Please participate! In-person, Canvas discussions, interacting with your classmates, etc.

Office Hours

- Thursday from 2 – 3 pm Pacific Time.
  - Remote or in-person
  - I will create a google doc sign up sheet with a zoom link
  - Room is E2 233
- I will also start/end classroom lectures 15 minute before/after
  - Not private
  - feel free to join just to socialize!
- Office hours are also available by appointment
- Post clarifying questions as discussion topics so that everyone can benefit!
- If 1 hour is not enough, I can try to add more.

Homeworks

1 Homework per module, total of 5 homeworks.

- 2 weeks for each homework, posted midway through module.
- You can develop however you like (Python3 is widely supported) but final solution must run in the docker
  - Instructions incoming!
- Ask questions on Canvas, discuss concepts.
- Start early!

Homeworks

Message me if you want to use languages/tools/etc. not mentioned in the homework. I can also install them in the Docker image

There is a cult of people who believe all compiler work should be done in functional languages! (I am a ex-member ☺)

Tests

- Designed to take ~120 minutes (not including studying)
  - No strict timer
  - Midterm will have 1 week
  - Final will have 1 day (8 am to 8 pm)
  - Scheduled time for final is: 8 am to 11 am
  - ***Often students take longer! Plan accordingly***

- Open book and open note

- Please:
  - Do not collaborate with classmates
  - Do not google for exact answers (googling for concepts is fine)
  - Ask any clarifying questions through private email to me (not on Canvas discussions)

Paper Assignment

- Propose an academic paper of interest (or I can suggest one) no later than 3 weeks before the end of classes.

- Write a 4 page double spaced review of the paper (due before the final)

Final Project

You can choose to do a final project instead of a final exam

- Final projects must be approved by me no later than 2 weeks before the end of the semester
- 6 page double spaced report on the project (due at the date of final)
- 15 minute presentation to the class (due last week of class)
- If you are a grad student, I suggest thinking about a project you can incorporate into your thesis/project.

Rubric

- 10% Attendance
- 40% Homework (8% each)
- 10% paper assignment
- 10% Midterm
- 30% Final

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

Resources

- Slides and lectures:
  - I will post the slides for each lecture by the day after
- Textbooks:
  - **Engineering: A Compiler 2nd Edition**: unlimited online availability from the library. Links in Resources tab of webpage.
  - **Compilers: Principles, Techniques, and Tools 2nd Edition (Dragon Book)**: Limited availability from the library. Ask me for links, or to borrow physical/kindle editions
- Academic papers/blogs:
  - I will link to them as needed on the Resources/schedule site

Walls of text incoming…

Academic Integrity

- One of the joys of university life is socializing and working with your classmates. I want you to make friends with each other and discuss the material. This is an advanced topics course; there is a high chance that your classmate will be your colleague throughout your career!
- **That said, I expect all assignments (homeworks, tests, paper reviews, presentations, final projects) to be your own original work.**
- If you work together with a classmate on an assignment, please mention this, e.g. in the comments of your code. If you use a figure you didn't create in a presentation, then it needs a citation. Please review the universities policy on plagiarism
- 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.

Privacy

If we are recording lectures to accommodate a remote student:

- Things you do or say will be recorded. I doubt that this will be an issue, but if you want me to remove any part of the recording, please just let me know.

- Canvas is secure™: other communications are less so

- Let’s do our best to protect our privacy and respect the privacy of others.

Disability Accommodation

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.

COVID Policy

- Please keep your badges up to date...
- Notify me directly if you test positive or need to quarantine so we can plan
- If someone in the class tests positive, the health services will contact us with recommendations about quarantining.

COVID Policy

- We are still in the midst of a global pandemic. This quarter will surely have challenges related to this. I hope we can all approach our interactions with empathy and understanding. I will do my best to accommodate the various individual challenges that may arise. Please communicate with me early and often!

- Currently, this is designated as an *in-person* class. It is **not** a hybrid class, that is, you do not have the option to attend remotely without an approved reason. As explained in the attendance section, I do expect you to plan on primarily attending in person. If you cannot attend in person for reasons related to COVID (e.g. if you or someone close to you gets sick), let me ASAP to discuss accommodations. You will not lose attendance points in these cases and I will do my best to provide lecture materials to you.

- If I am unable to attend (e.g. if I get COVID), then I will aim to teach remotely as long as I am feeling well enough.

Now back to something a little lighter…

LSD Seminar

- CSE-280O-01: LSD Seminar (co-organized with Lindsey Kuper)
  - Friday's at Noon
  - 1 hour talk + 15 minutes of social
  - Student speakers!
  - https://lsd-ucsc.github.io/lsd-seminar/2021fa/
- Topics will be highly relevant to this class; please consider joining!

Website tour

- Class website

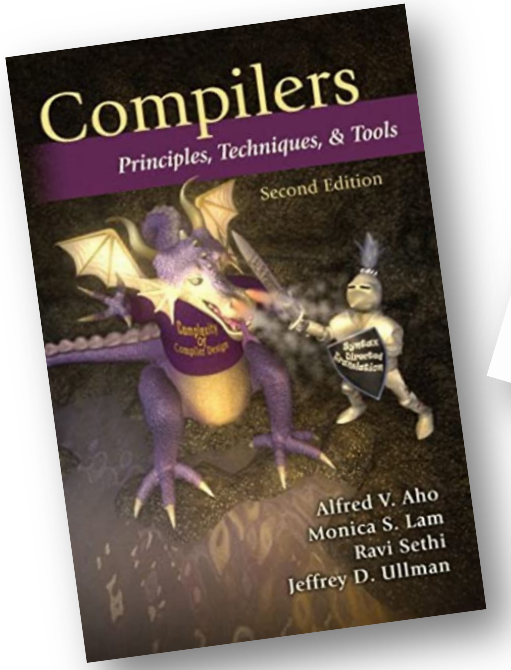
- Canvas

Made it through the syllabus…

The slides should start to get more interesting now

CSE211: Compiler Design

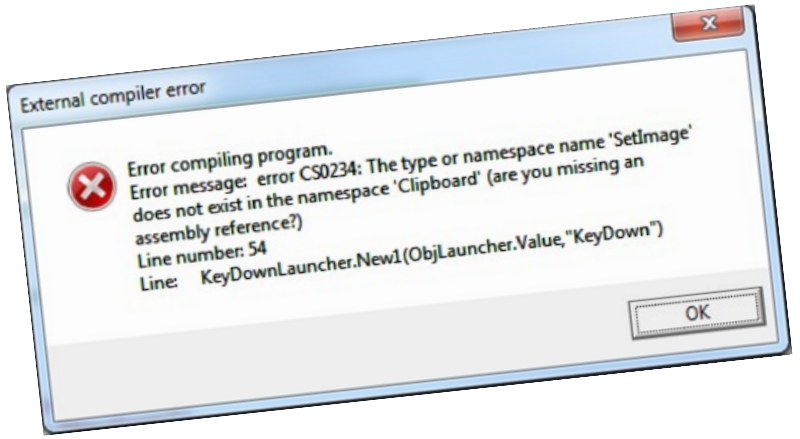
Oct. 1, 2020




```
<expr> ::= <term> "+" <expr>
         | <term>
<term> ::= <factor> "*" <term>
         | <factor>
<factor> ::= "(" <expr> ")"
           | <const>
<const> ::= integer
```

- **Topic**: Course Introduction

- **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

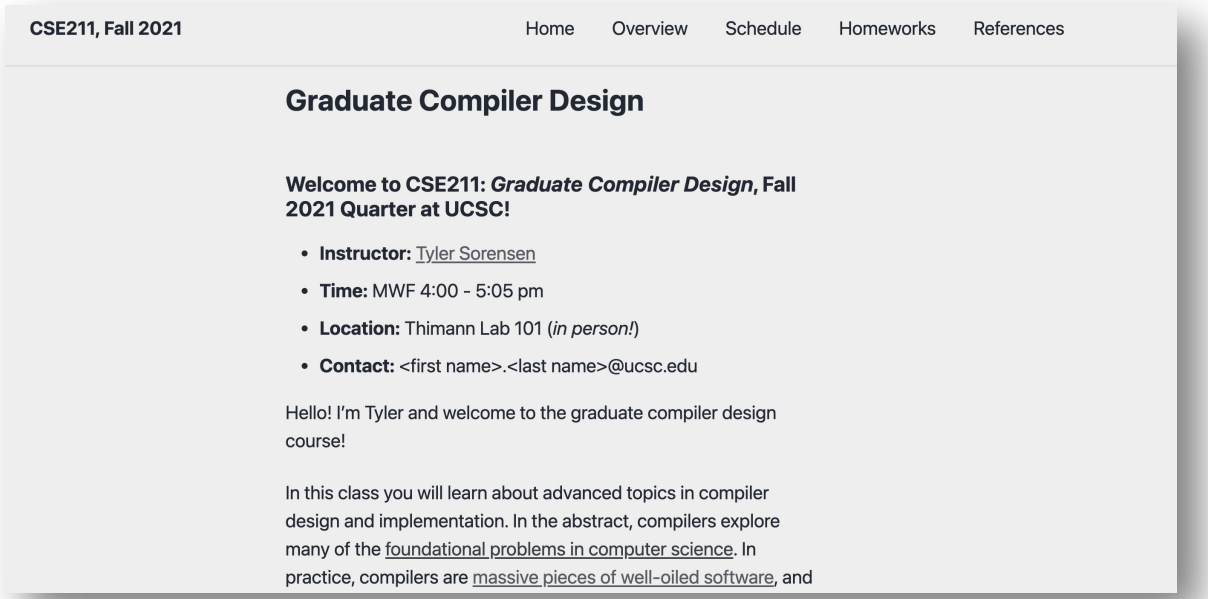
```
---
title: "Graduate Compiler Design"
layout: single
---


### 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

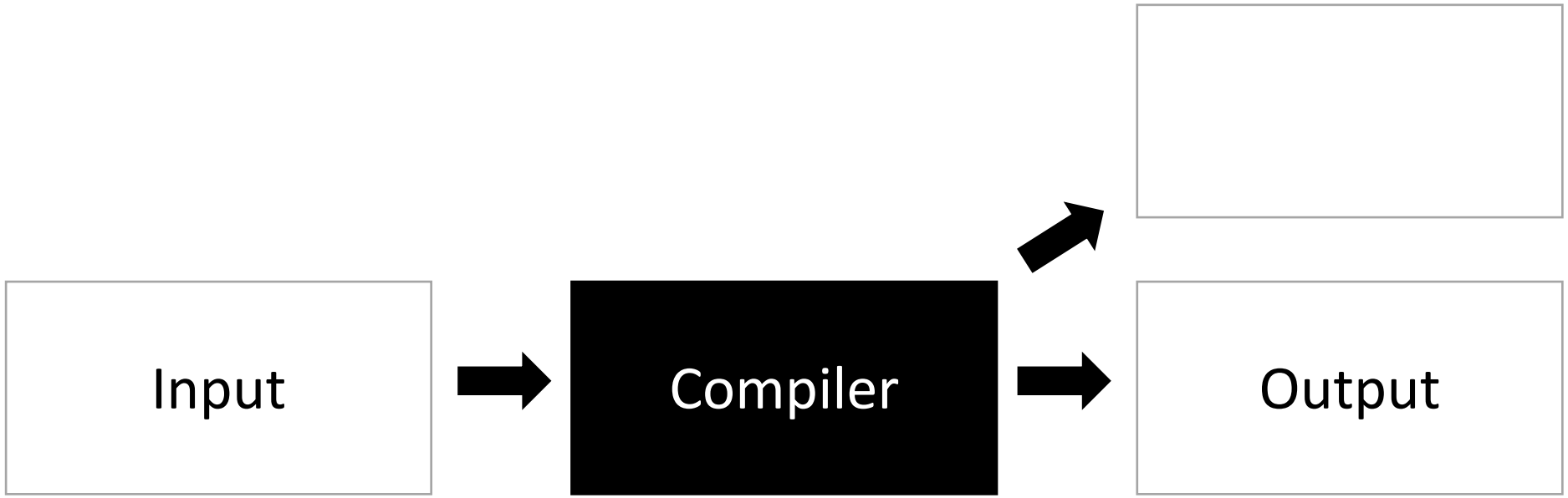
A program written in C++

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

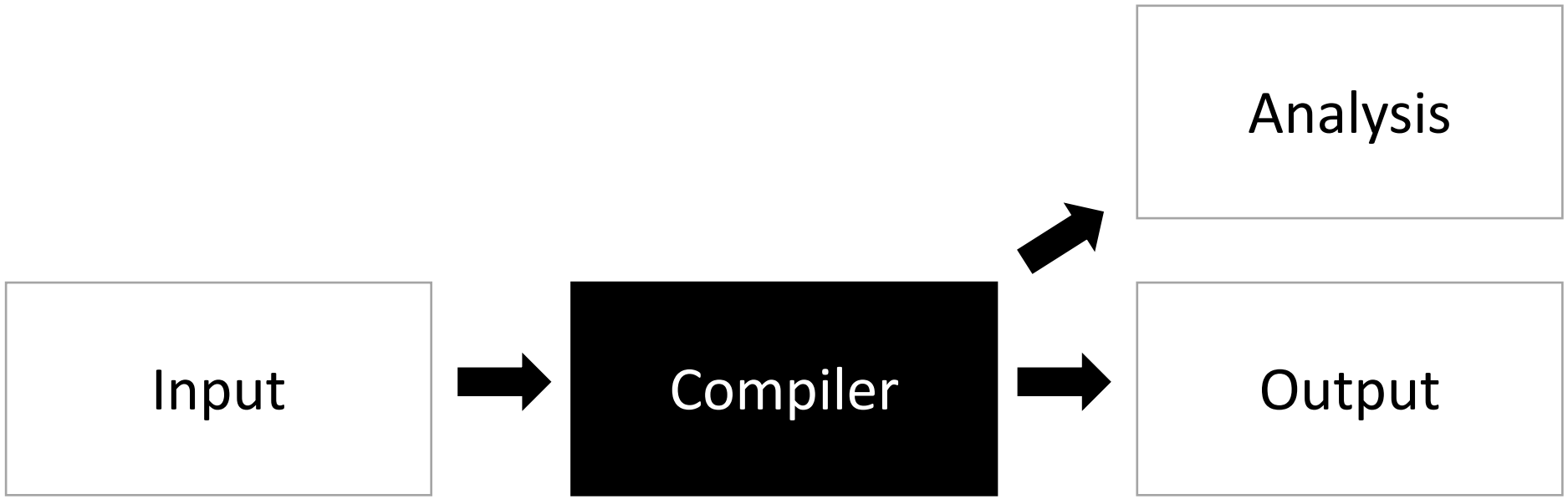
A program written in C++

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++

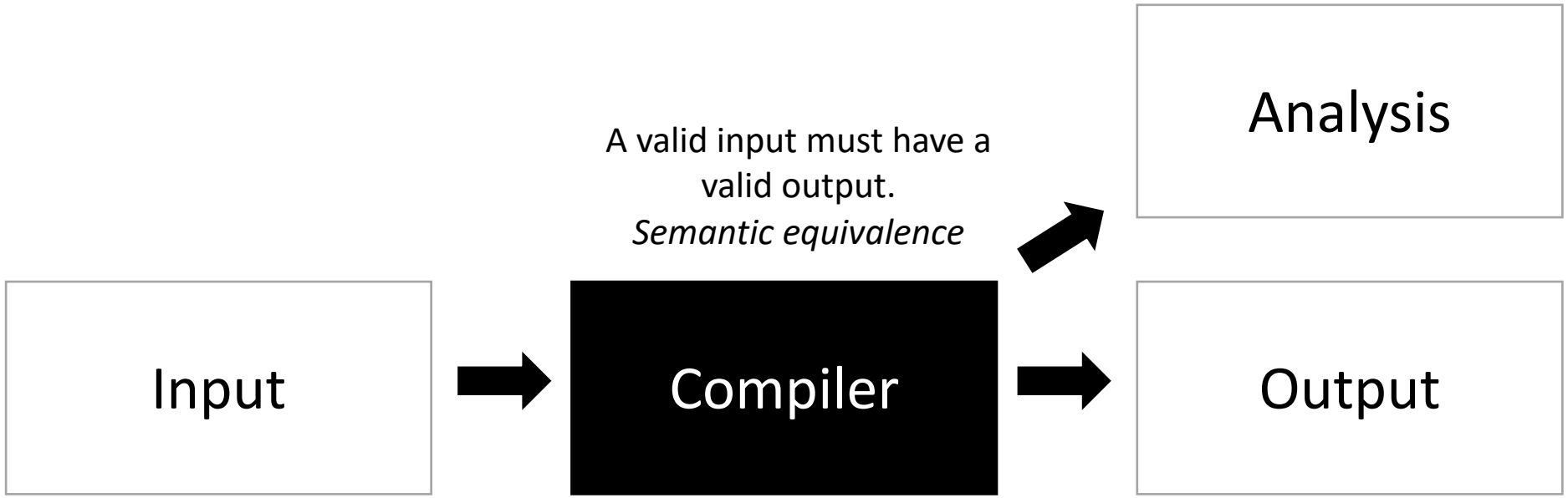
Strings belonging to language L’

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

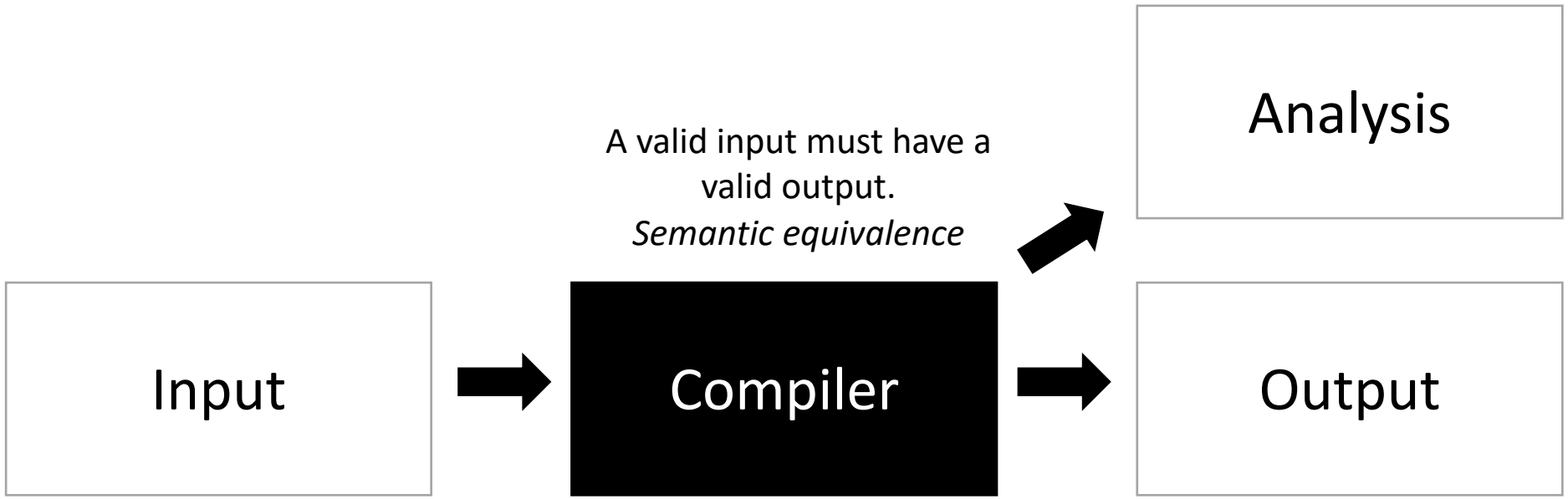
A program written in C++

Strings belonging to language L’

An executable binary file in an ISA language

An x86 Binary executable

What is a compiler?



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 clang; you will get an error and a suggestion!

What can happen when the Input isn't valid?

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

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

What about this one?

What is the compiler allowed to do?

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

Try running this on https://godbolt.org/
change the optimization level to -O3 and see what happens!

What is the compiler allowed to do?

```
void add_arrays(int *a, int *b)
for (int i = 0; i < 128; i++) {
   a[i] += b[i];
}
```

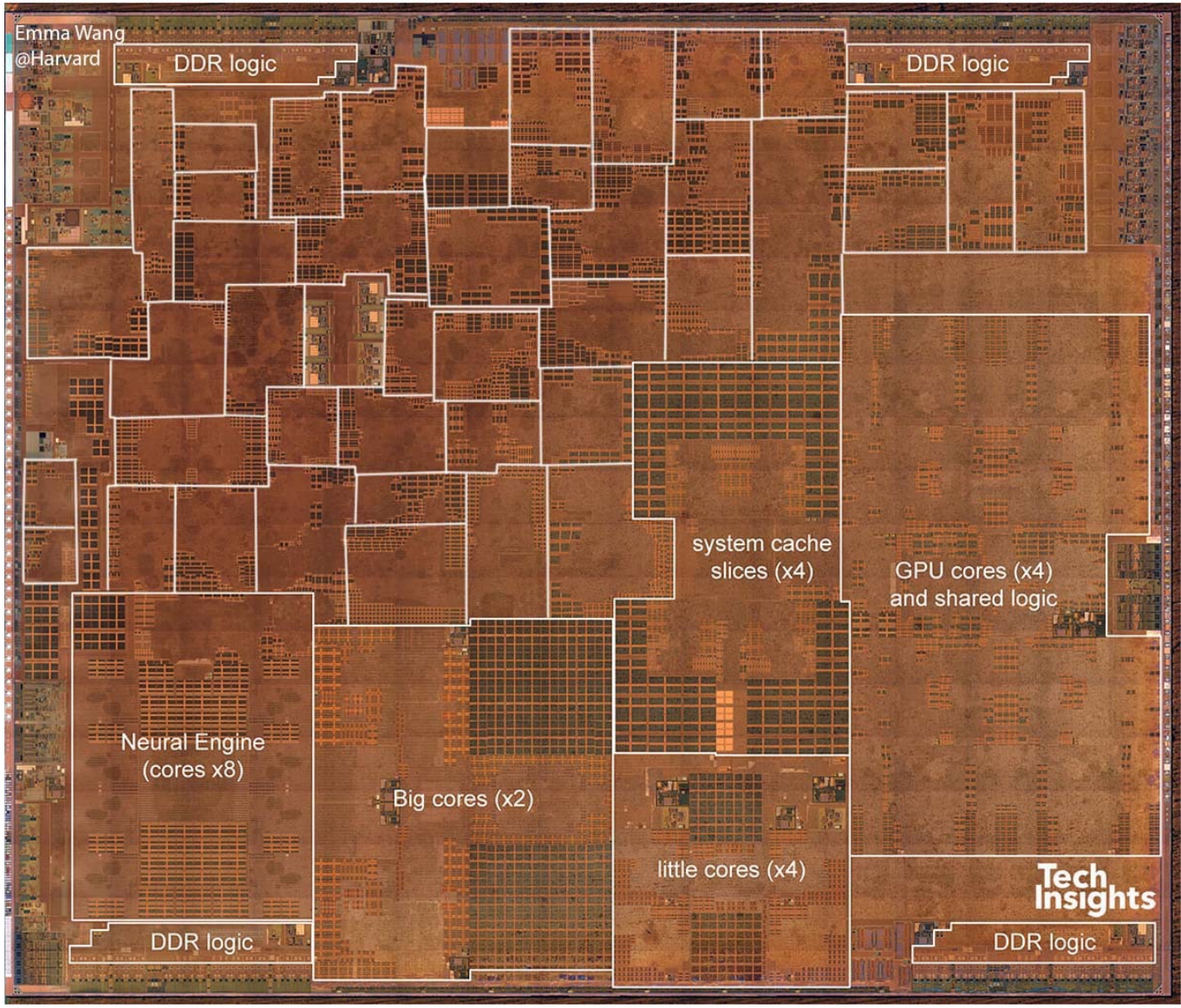
Try running this on https://godbolt.org/
change the optimization level to -O3 and see what happens!
Look for instructions like `paddd`. what does it do?

Going forward

- From David Brooks lab at Harvard:

  http://vlsiarch.eecs.harvard.edu/research/accelerators/die-photo-analysis/

- Compilers will need to be able to map software efficiently to a range of different accelerators



Looking forward to a fun semester!

- Next week we start module 1:

Module 1: Parsing Overview: This module will go over parsing at a high-level. This includes tokenizing, parsing context-free grammars, parser generators, and how to quickly create a parser for a simple programming language.

Readings:

- Chapter 1 EAC, nice overview and motivation of compilers
- Chapter 2 EAC, deeper than we will go, but good reading

What is the compiler allowed to do?

If there is time: can we make it so that the compiler doesn't optimize away this loop?

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

Try running this on https://godbolt.org/
change the optimization level to -O3 and see what happens!